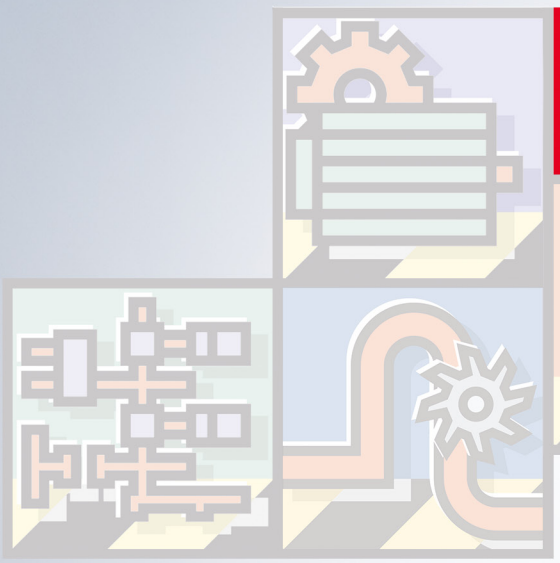
BECKHOFF New Automation Technology

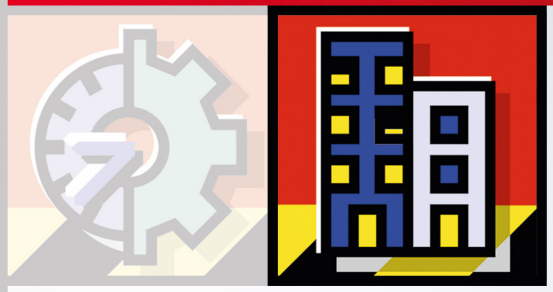
Handbuch | DE

TS8020

TwinCAT 2 | BACnet/IP



Supplement | Building Automation



11.08.2023 | Version: 1.3

Inhaltsverzeichnis

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, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche 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 weiter entwickelt.
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.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

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

Copyright

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 Einleitung

BACnet (Building Automation Control Network) ist ein standardisiertes, herstellerunabhängiges Kommunikationsprotokoll für die Gebäudeautomatisierung. Verwendung findet es in den Bereichen HLK, Lichtsteuerung, Sicherheits- und Brandmeldetechnik. Die Implementierung dieses Protokolls erfolgte sowohl als Server als auch als Client und ist auf allen Beckhoff Industrie-PCs und Embedded-PCs lauffähig. Unterstützt werden alle Dienste eines BBC (BACnet Building Controller), wie z. B. gemeinsame Datennutzung (DS), Alarm- und Ereignisverarbeitung (AE), Zeitplan (SCHED), Trendaufzeichnung (T) sowie Device- und Netzwerkmanagement (DM).

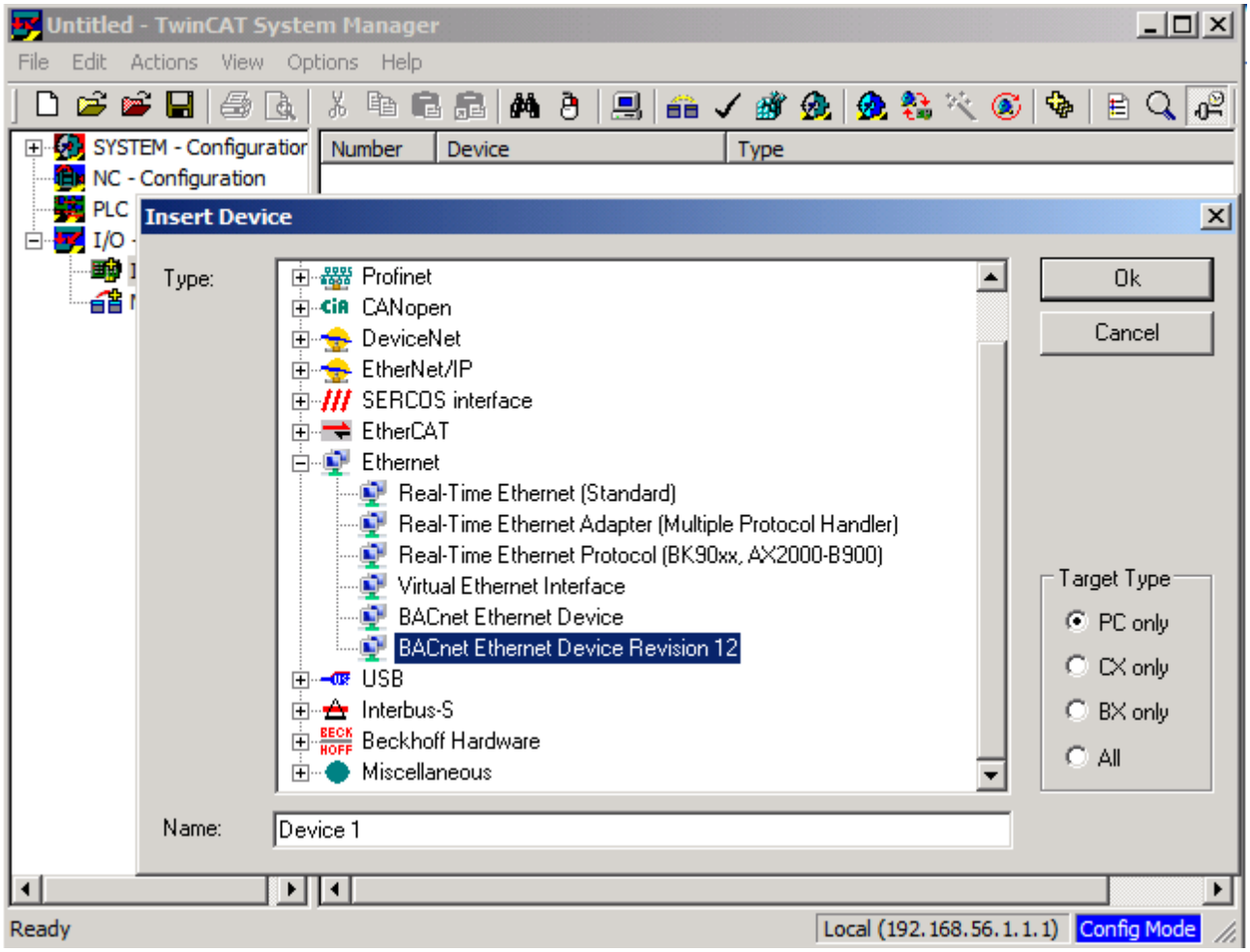
Die hier vorliegende Dokumentation bezieht sich ausdrücklich auf die technische Beschreibung von TwinCAT BACnet/IP und damit der Umsetzung des BACnet Standard in die TwinCAT Laufzeit- und Konfigurationsumgebung. Ausführliche Kenntnisse von BACnet werden vorausgesetzt. Es existiert zahlreiche Literatur auf die ggf. im Vorfeld zurückgegriffen werden kann:

- Buch: BACnet Gebäudeautomation 1.4, Hans R. Kranz, CCI Promotor Verlag 2. Auflage ISBN: 3-922420-09-5
- Buch: BACnet and BACnet/IP - A Clear Description, F. Tiersch, C.Kuhles, Desotron Verlag ISBN 978-3-932875-31-1
- Externer Link: ASHRAE BACnet Bibliography
- Externer Link:BACnet Interest Group - BACnet Literatur
- Externer Link: BACnet in öffentlichen Gebäuden (AMEV)

2.1 BACnet-Device

BACnet/IP ist ein Ethernet basierendes Protokoll und kann deshalb im TwinCAT System Manager in der Kategorie "Ethernet" projektiert werden. Über das Konfigurationselement "I/O Devices" kann einer Konfiguration ein BACnet-Device hinzufügt werden. Das BACnet-Device stellt die Verbindung zu einem Netzwerkadapter her.

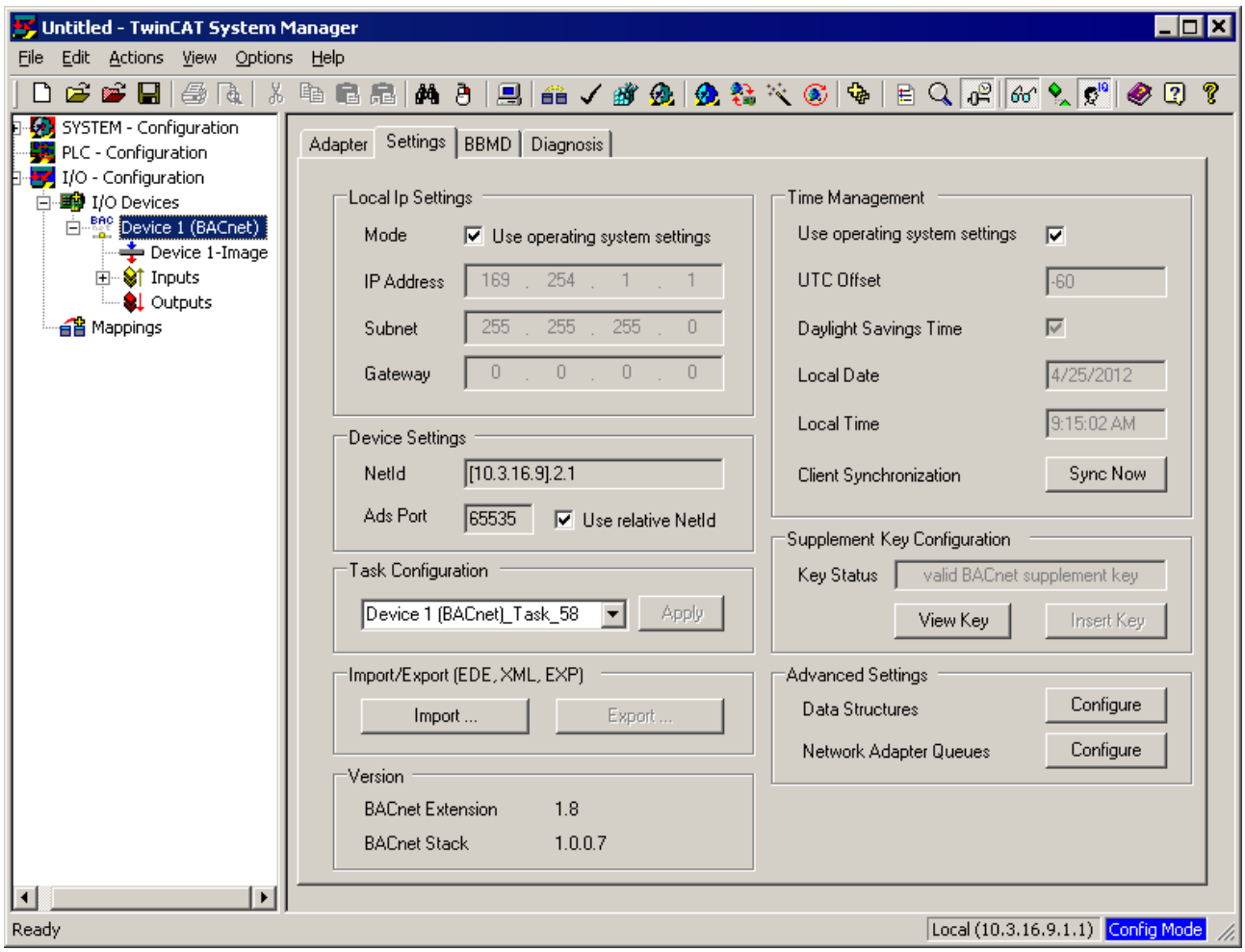
Momentan existieren zwei Versionen von BACnet, die aus Kompatibilitätsgründen parallel im System Manager konfiguriert werden können. Beim "*BACnet Ethernet Device*" handelt es sich um die ältere zertifizierte Version nach BACnet Revision 6. "*BACnet Ethernet Device Revision 12*" ist die aktuelle BACnet-Version nach Revision 12, die auch zertifiziert wurde. Für neue Projekte sollte immer die aktuelle Version; momentan Revision 12; verwendet werden.



BACnet-Device Einstellungen

IP-Einstellungen

Über den Reiter "Settings" eines BACnet-Device wird dessen IP-Adresse angezeigt und kann verändert werden. Die IP-Adresse wird beim Anlegen vom ausgewählten Netzwerkadapter übernommen. Die für BACnet verwendete IP-Adresse muss dabei nicht mit der IP-Adresse des Netzwerkadapters übereinstimmen, sondern kann in einem separaten IP-Adressraum angesiedelt sein. Auf einem TwinCAT-System können mehrere BACnet-Devices projektiert werden, wenn mehrere Netzwerkadapter vorhanden sind. Über die Option "Use operating system settings" kann die IP-Einstellung des Betriebssystems übernommen werden. Diese Option bietet sich z.B. an, wenn ein Gerät seine IP-Adresse über DHCP bezieht.



AmsNetID-Einstellungen

Nach dem Anlegen eines BACnet-Device können über den Settings-Reiter weitere Einstellungen vorgenommen werden. Jedes BACnet-Device verfügt über eine AmsNetId, die als Zugangspunkt für azyklische Dienste verwendet wird. Die AmsNetID besitzt 6 Stellen. Die ersten 4 Stellen werden durch die System-NetID gebildet, die 5. Stelle wird durch die Item-ID bestimmt (1.Gerät: 2 - 2.Gerät: 3 usw.) und die 6. Stelle ist immer 1. Soll eine BACnet-Konfiguration auf einem anderen Zielsystem aktiviert werden, müssen die ersten 4 Stellen der AmsNetID entsprechend angepasst werden. Hierzu erscheint beim Verbinden mit dem neuen Zielsystem ein entsprechender Dialog. Generell wird empfohlen den Modus "Use relative NetId" zu aktivieren. Hierbei werden intern die ersten 4 Stellen der NetId mit 0 initialisiert und beim Laden des BACnet-Device durch die aktuelle System-NetID ersetzt. Hierdurch ist es möglich eine auf einem System aktivierte Konfiguration durch Kopieren entsprechender Dateien (CurrentConfig.xml) auf ein anderes Zielsystem zu übertragen.

Task-Konfiguration

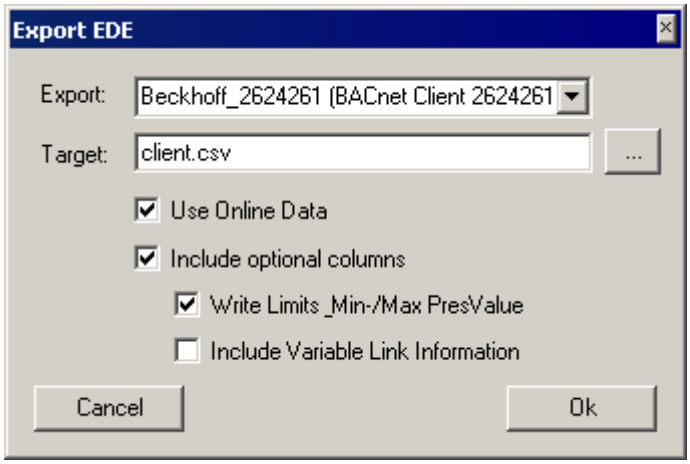
Ein BACnet-Device wird über ein asynchrones Mapping mit einer Task verbunden. Die entsprechende Task kann über das Feld "Task Configuration" eingestellt werden. Für ein BACnet-Device kann jede Task mit einem Port >350 verwendet werden. Wird ein BACnet-Device erstellt, wird automatisch eine Default-Task mit der Priorität 58 erstellt und dem BACnet-Device zugeordnet.

Import-/Export

Des Weiteren können über den "Settings"-Reiter des BACnet-Device verschiedene Datei-Formate importiert und exportiert werden. So kann eine .exp-Datei exportiert werden, die einer SPS eine BACnet-Konfiguration zur Verfügung stellt. Es werden Deklarationen von Funktionsbausteinen der TcBACnet.lib bzw. TcBACnetRev12.lib für alle projektierten BACnet-Server- und Client-Objekte erstellt, die dann im SPS-Programm verwendet werden können. So kann z.B. sehr komfortabel auf entfernte BACnet-Objekte zugegriffen werden, da im Rahmen des SPS-Automapping Prozessdatenvariablen automatisch verknüpft werden.

Der XML-Export und -Import verarbeitet BACnet-Konfigurationen in einem herstellerspezifischen Format.

Zusätzlich werden der Import und Export des *Engineering Data Exchange* (EDE) in der Version 2.2 unterstützt. Das EDE-Format ist ein Komma bzw. Semikolon-separierte Liste mit festen und optionalen Spalten. Ob Komma oder Semikolon als Trennzeichen verwendet wird, hängt von länderspezifischer Einstellung ab. Es wird der im Betriebssystem (Regions- und Spracheinstellungen) festgelegte Listenseparator verwendet. Beim Export kann selektiert werden, ob zusätzlich die optionalen Spalteneinträge exportiert werden sollen ("Include optional columns"). Zusätzlich können in der optionalen Spalte "vendor-specific-address" Informationen zu verknüpften Variablen der TwinCAT-Konfiguration in die EDE-Datei übernommen werden. Beim EDE-Export eines BACnet-Servers kann gewählt werden, ob Offline-Daten (der Settings-Dialoge) oder Online-Daten für den Export verwendet werden sollen.



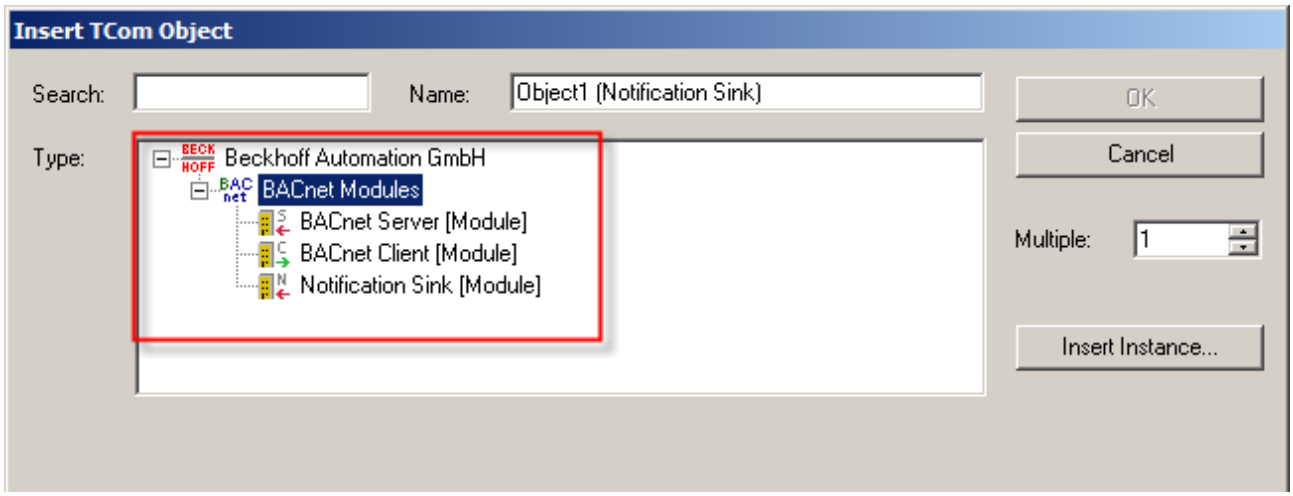
Zeitverwaltung

In der Gruppe "Time Management" des BACnet-Device "Settings"-Reiter kann eingestellt werden, ob die Uhrzeit vom unterliegenden Betriebssystem übernommen werden soll. Hierbei wird auch die Sommer- und Winterzeit automatisch aktualisiert, wenn eine Automatik im Betriebssystem konfiguriert wurde. Bei der manuellen Zeiteinstellung kann der UTC-Offset sowie der DayLightSavingsStatus (Sommer-/Winterzeit) über diesen Dialog vorgegeben werden. Bei der manuellen Konfiguration der Zeitparameter können der UTC-Offset und DayLightSavingsStatus über BACnet verändert werden. Die Umschaltung zwischen Sommer- und Winterzeit bewirkt dann eine Zeitumstellung von 60 Minuten.

Weitere Einstellmöglichkeiten des BACnet-Device werden in weiteren Abschnitten vorgestellt.

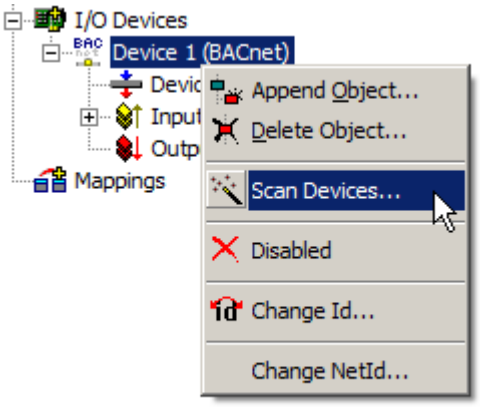
BACnet-Server, -Client

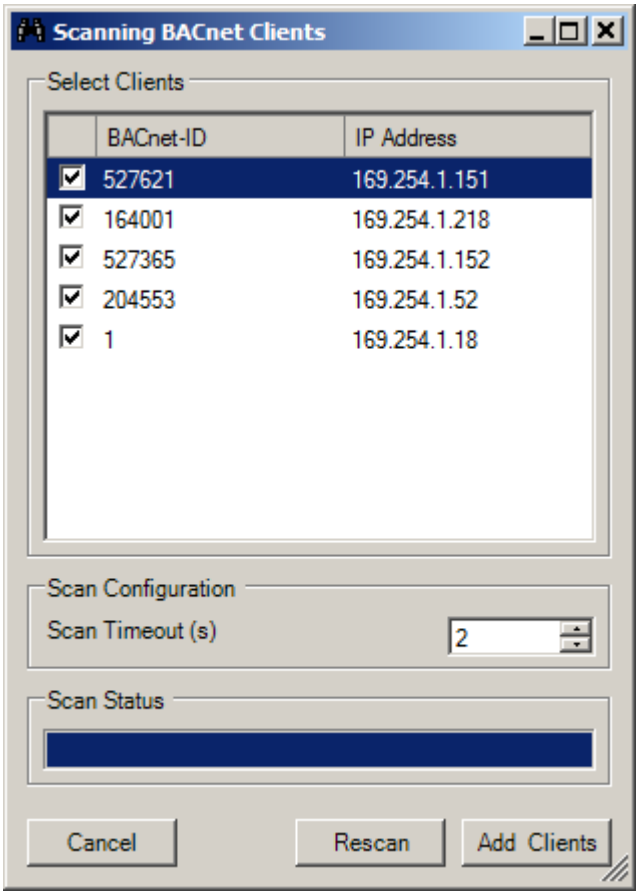
Unterhalb eines BACnet-Device können BACnet-Clients und -Server projektiert werden (siehe auch Beispiel: "BACnet Adapter und Server anlegen [▶ 96]"). Ein Server stellt BACnet-Objekte zur Verfügung. Ein BACnet-Client ermöglicht den Zugriff auf BACnet-Objekte anderer BACnet-Geräte (Server) im Netzwerk.



BACnet-Clients in einem Netzwerk können durch Scannen automatisch erkannt werden. Hierzu wird ein BACnet-Device angelegt und mit Hilfe der Scan-Funktion des TwinCAT System Manager zum Suchen nach anderen Geräten (Servern) veranlasst. Für jedes gefundene Gerät werden die geladenen BACnet-Objekte erkannt und im System Manager angelegt. Zudem können Clients mit deren BACnet-Device-ID manuell konfiguriert werden, wobei die entsprechende IP-Adresse mit dem BACnet-Dienst Who-Is automatisch ermittelt wird. Das Anwenden der Scan-Funktion auf einen selektierten BACnet-Client führt zum Erkennen der Objekte dieses Gerätes, so dass die manuelle Projektierung entfällt.

Das Scannen von Clients resp. deren Objekten und Properties setzt jedoch eine bestehende BACnet-Installation voraus.



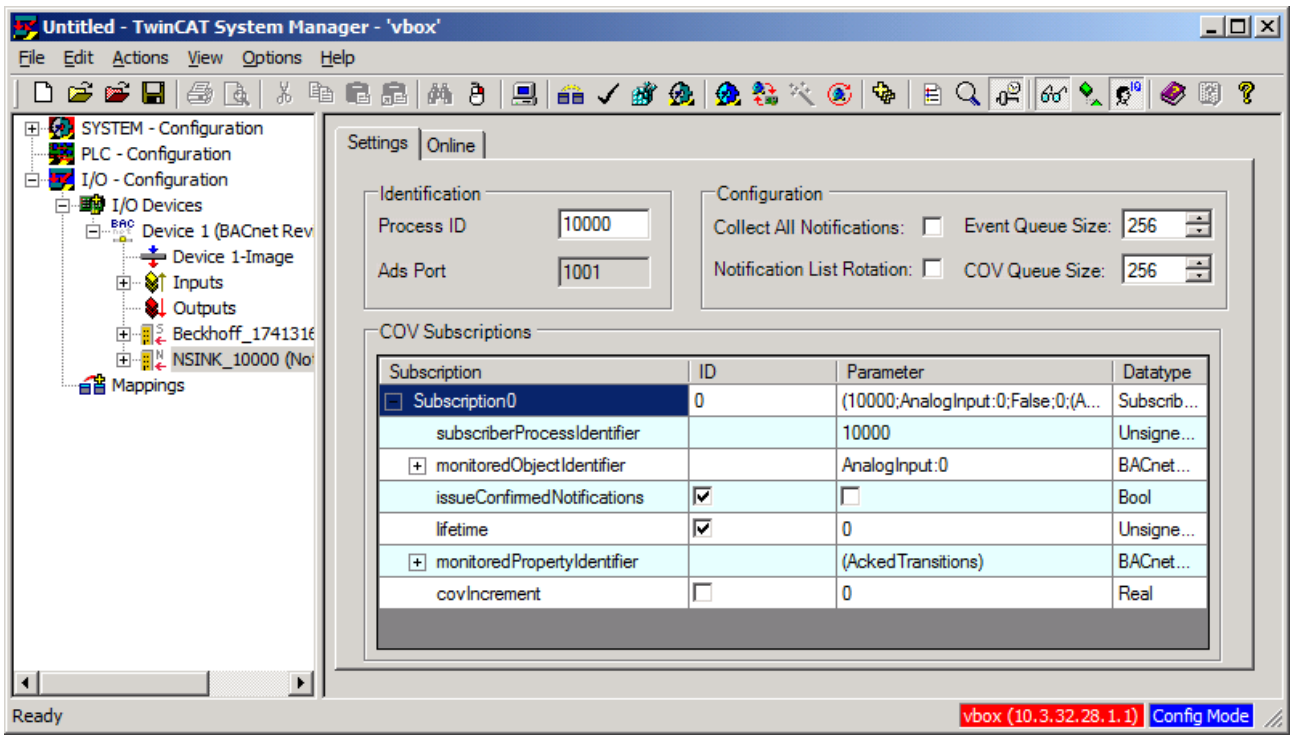


BACnet Notification Sink

Eine Besonderheit unter den BACnet-Modulen stellt die sogenannte Notification Sink dar. Mit Hilfe dieser können Events gesammelt und COV-*Subscriptions* abonniert werden. Funktionell handelt es sich dabei nicht um ein BACnet-Objekt, so dass die Notification Sink aus BACnet nicht für andere Clients sichtbar ist. Jedoch ermöglichen ADS-Dienste den Zugriff auf die Notification Sink (u.a. auch via TCP/IP, siehe Kapitel "ADS-Interface [▶ 520]").

Notification Sink und Zertifizierung

Die Notifikation Sink ist kein Bestandteil der Zertifizierung. Die Verwendung der Notifikation Sink obliegt der Verantwortung des Anwenders.



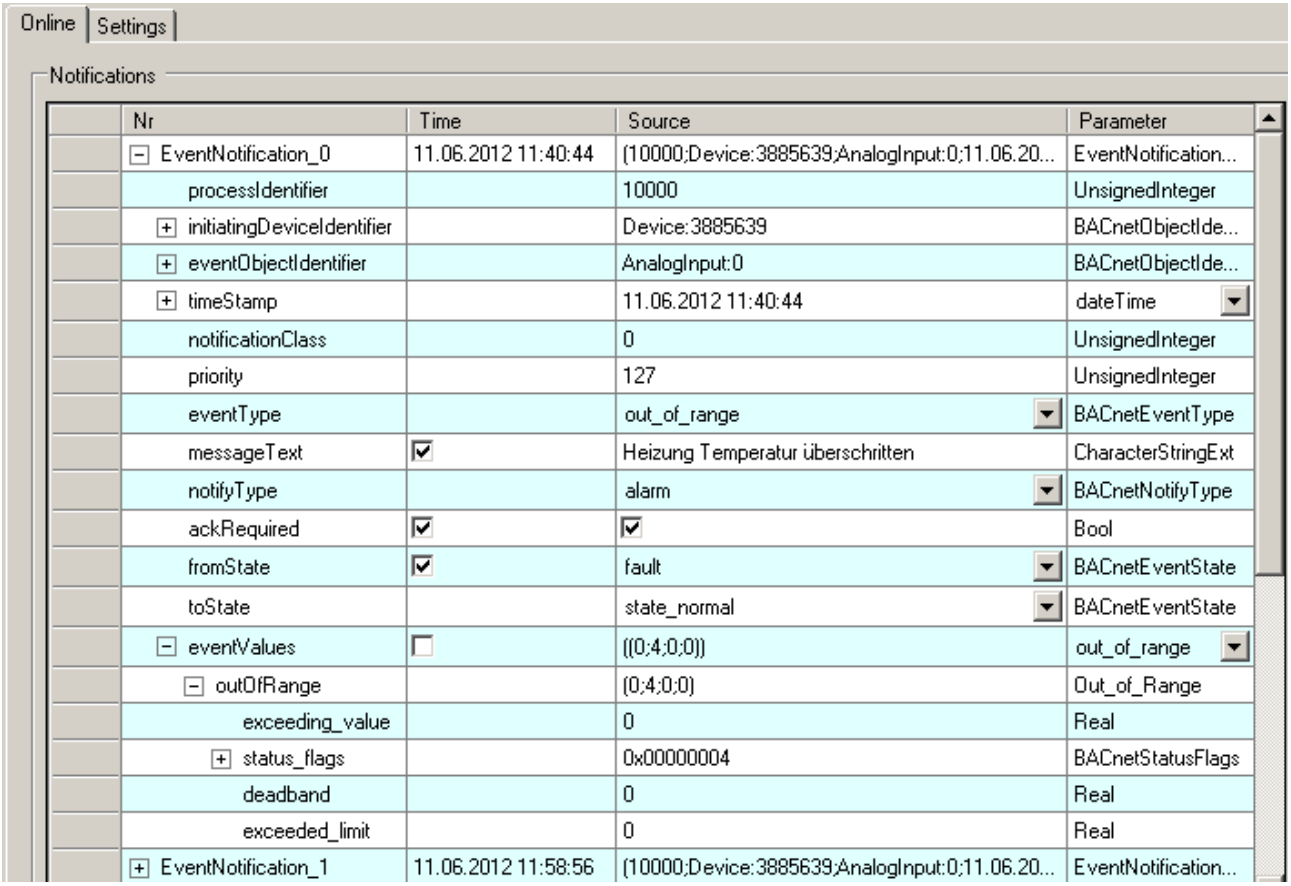
Pro BACnet-Device können mehrere Notification Sinks (mit verschiedenen Prozessidentifikatoren - Process ID) konfiguriert werden, um z.B. verschiedene Gewichtungen von Ereignisprioritäten zu unterscheiden. Für jede Notification Sink wird ein eindeutiger ADS-Port generiert, der unter "Identification" angezeigt wird. Über das Feld "Configuration" kann die Größe der Warteschlagen für COV- und Event-Notifications festgelegt werden. Ist eine Warteschlange voll, werden neu ankommende Notifications verworfen. Über Option "Notification List Rotation" kann dafür gesorgt werden, dass bei voller Warteschlange die älteste Nachricht verworfen wird (Ringpuffer). Ansonsten muss über die ADS-Schnittstelle sichergestellt werden, dass die Warteschlangen entsprechend schnell verarbeitet und Einträge (via ADS) gelöscht werden.

Über die Option "Collect All Notification" kann für eine Notification Sink festgelegt werden, dass alle Event- und COV-Notifications unabhängig ihrer Process-ID empfangen werden; also alle Meldungen die auf dem System eingehen.

Hauptverwendung für die Notification Sink ist die Verarbeitung bzw. Visualisierung von Alarmmeldungen (Event-Notifications). Abonnements von COV-Notifications sollten nur verwendet werden, wenn Zustandsänderungen von komplexen BACnet-Properties implementiert werden müssen. Z.B. kann hiermit überwacht werden, ob einem Gerät dynamisch neue BACnet-Objekte hinzugefügt wurden. Für Properties mit einfachen Datentypen sollten Prozessdaten von Remote-Objekten verwendet werden, da auch hier effiziente COV-Notifications umgesetzt werden.

Über ein Kontextmenü kann ein anderer projektierter Client als Ziel für COV-Subscriptions eingestellt werden und über den Property-Wizard komfortabel Properties selektiert werden.

Im Reiter "Online" können empfangene Notifications (inkl. Empfangszeitstempel) betrachtet werden. Zusätzlich können über ein Kontextmenü im Online-Reiter einzelne Event- und COV-Notifications gelöscht und auch alle Warteschlangen mit einem Befehl gelöscht werden. Auch das Bestätigen von Meldungen (AcknowledgeAlarm) ist über den Online möglich. Beim Bestätigen einer Meldung kann ein Meldungstext für die Bestätigungsnachricht angegeben werden, die bei allen Recipients dieser Notification angezeigt wird.



Die Abbildung zeigt eine EventNotification im Online-Reiter einer Notification Sink. Der Nachrichtentext (messageText) einer EventNotification wird in TwinCAT BACnet/IP-Geräten nachfolgendem Verfahren gebildet:

- Ist die Property EventMessageTexts aktiv (nicht optional deaktiviert) wird der im Settings-Dialog vorgegebene Text für den Zustandsübergang verwendet
- Sonst: Ist die Property Description vorhanden, wird dieser Text als messageText verwendet.
- Sonst: Wird der messageText nicht gesendet (optionales Feld deaktiviert)

Das Löschen von EventNotifications kann auch über den Funktionsbaustein FB_BACnet_NotificationSinkDelEntry [▶ 496] der SPS-Bibliothek TcBACnet.lib bzw. TcBACnetRev12.lib realisiert werden.

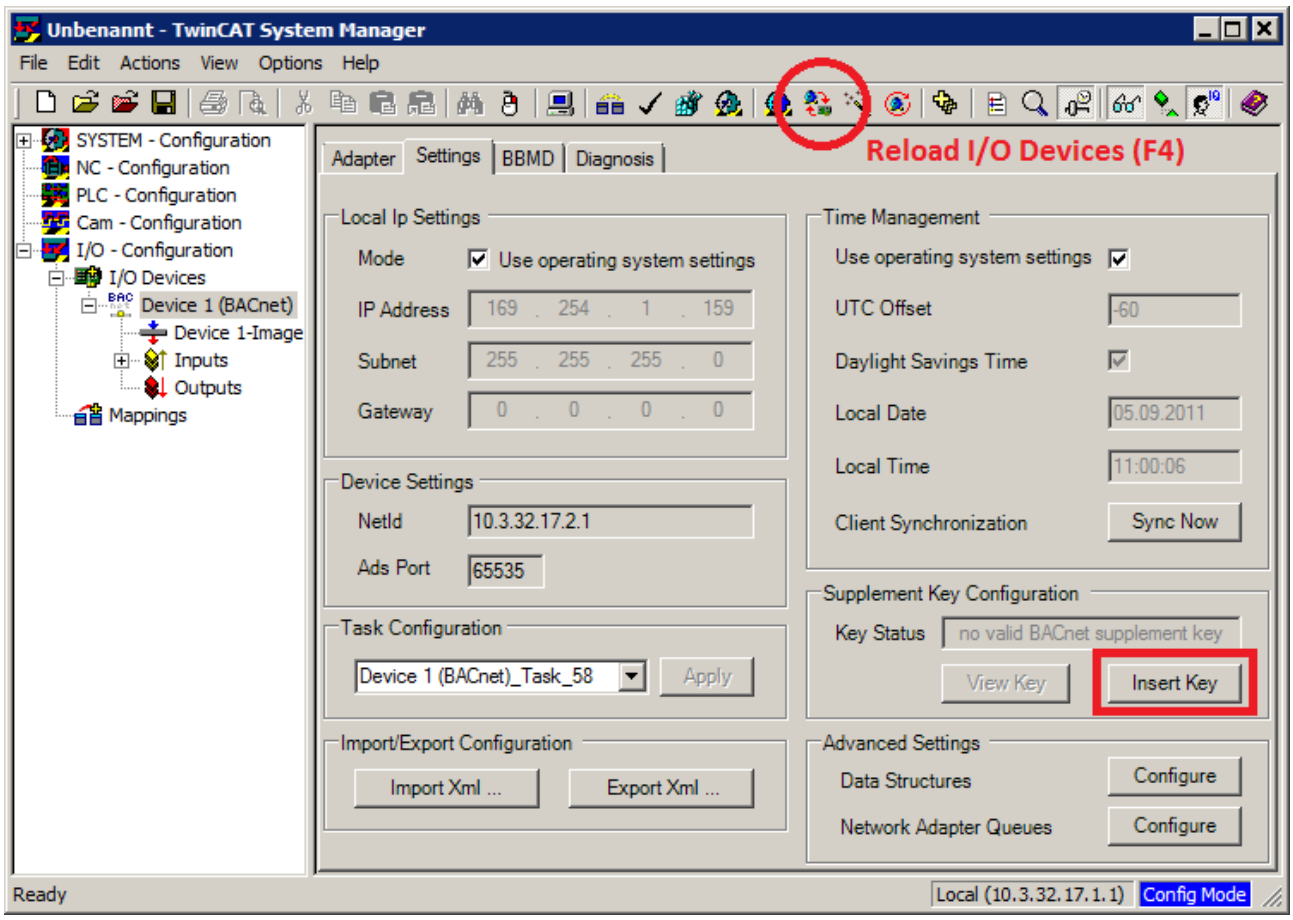
Ein Beispiel zur Verwendung der NotificationSink sowie der Generierung von EventNotifications findet sich im Abschnitt Anwendung: Beispiel: NotificationClass und NotificationSink [▶ 133].

2.2 BACnet Supplement Key einrichten

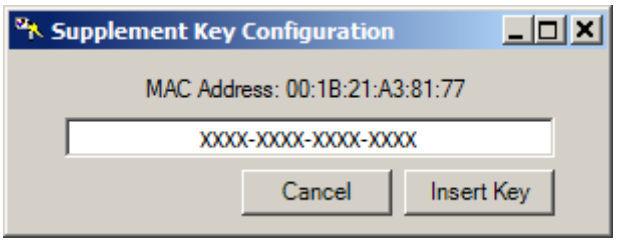
Um eine TwinCAT BACnet/IP Installation im Run Modus ausführen zu können, ist der Erwerb eines Supplement-Keys notwendig. Dieser Schlüssel ist an die MAC-Adresse des Netzwerkadapters gebunden. Es können Supplement-Keys für mehrere Netzwerkadapter auf einem Rechner eingerichtet werden.

Windows CE Images

Bei Windows CE basierenden Images ist in entsprechenden BACnet-Versionen (spezielle Bestellnummer) der Supplement-Key schon vorinstalliert und muss nicht nachträglich eingetragen werden.



Um einen Supplement-Key hinzuzufügen, muss ein BACnet-Device angelegt und geladen werden („Reload Devices“). Danach kann über den Reiter „Settings“ des BACnet-Device der Key mit dem Button „Insert Key“ hinzufügt werden.



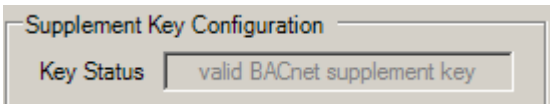
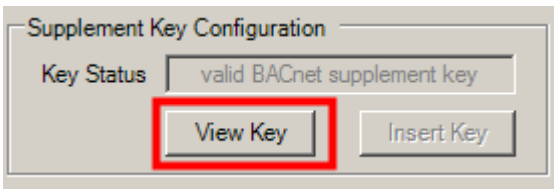


Image Updates

Ist ein gültiger Key eingetragen, kann dieser auch zu einem späteren Zeitpunkt mit Hilfe des Button "View Key" angezeigt werden. Bei Windows-Embedded basierten Geräten (CX5010, CX5020) kann ein Image-Update zum Verlust des Supplement-Key führen. Hier empfiehlt es sich vor dem Update den Supplement-Key zu notieren.

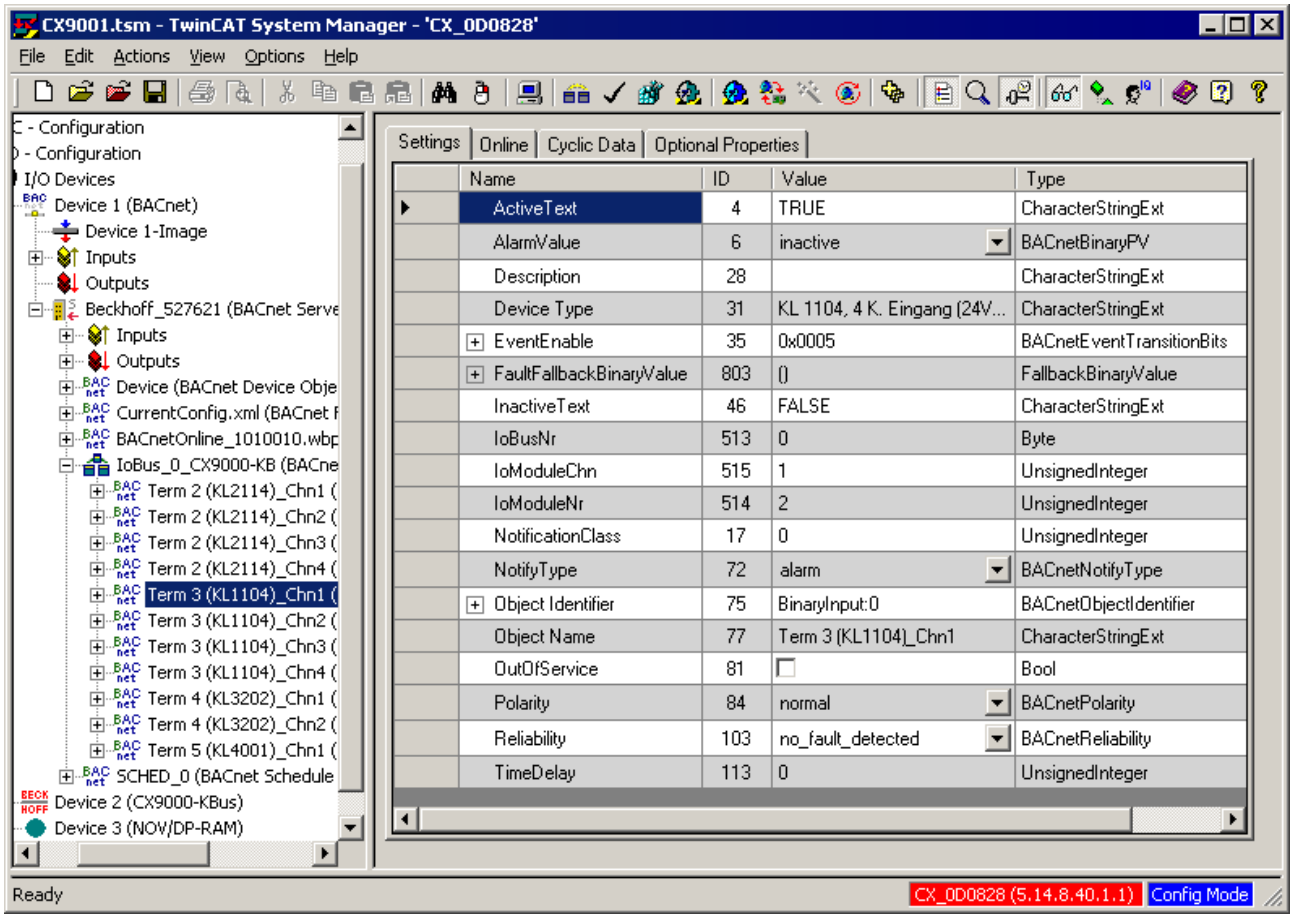


2.3 BACnet Objekte und Properties

Allgemeines zur Handhabung

Zu jedem BACnet-Objekt gehören eine definierte Anzahl Properties. Diese Properties werden auf BACnet-Serverseite offline konfiguriert und im Betrieb (RUN oder Free RUN) anderen BACnet-Teilnehmern im BACnet-Netzwerk zur Verfügung gestellt. Im Reiter "Settings" werden diese Offline-Daten, bzw. Initialwerte, der Objekt-Properties angezeigt. Im Reiter "Online" sind die aktuellen Werte der Objekt-Properties zur

Laufzeit zu sehen. Einige Properties sind schreibgeschützt und an dem Schloss-Symbol erkennbar , diese sind nicht änderbar.

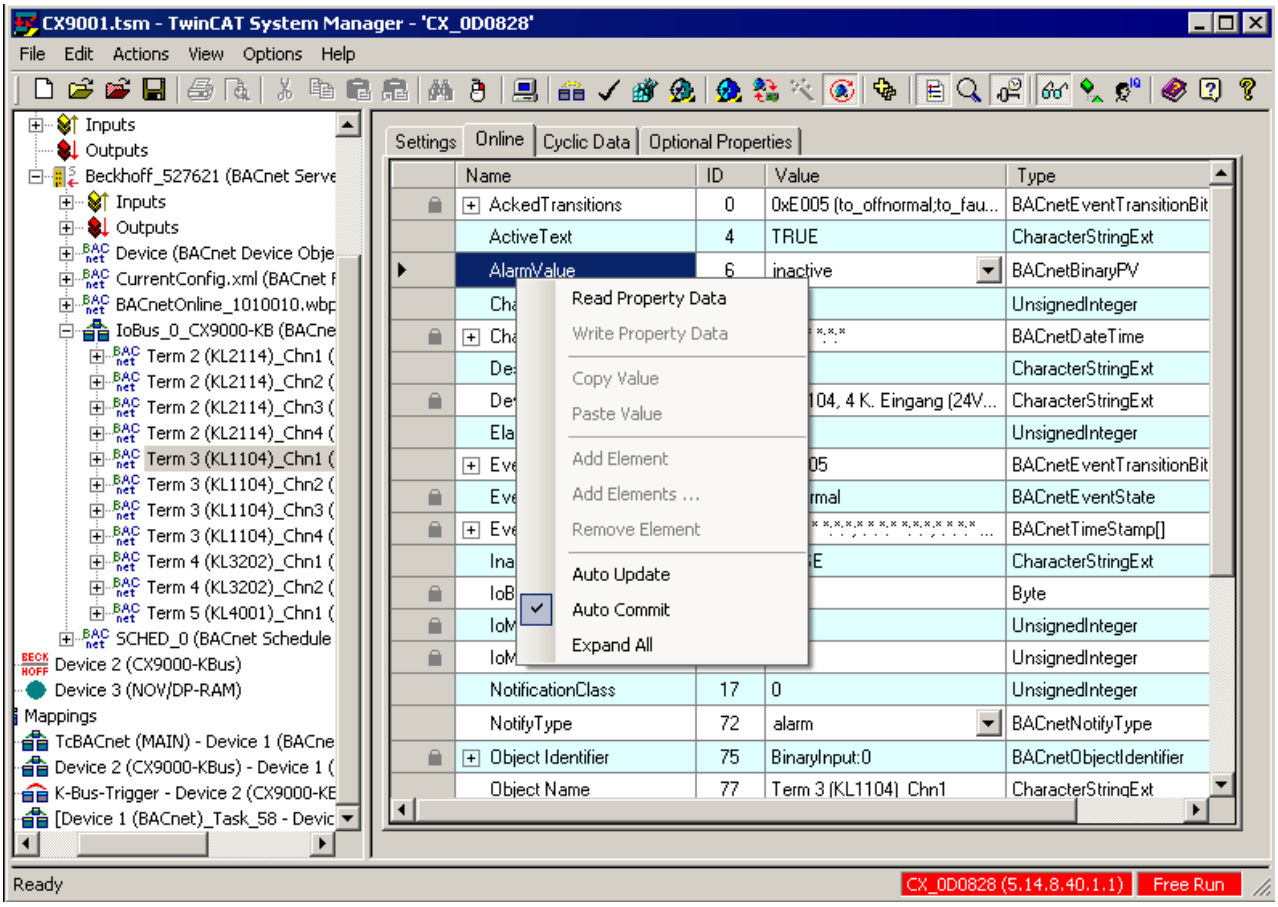


Jede Objekt-Property hat eine eindeutige Nummer. Anhand der Nummerierung werden Properties in TwinCAT BACnet/IP in folgende Gruppen unterteilt:

- Property-Identifier 0...511: BACnet standardkonforme Properties:
  - definiert in ISO 16484-5
  - festgelegtes definiertes herstellerübergreifendes Verhalten
  - teilweise offline projektierbar, teilweise nur Repräsentation eines online Zustands
- Property-Identifier 512...799: Herstellerspezifische Properties mit BACnet Sichtbarkeit:
  - im BACnet-Netzwerk als Properties der entsprechenden Objekte sichtbar
- Property-Identifier 800...1000: Herstellerspezifische Properties ohne BACnet Sichtbarkeit:
  - dienen der Konfiguration interner Parameter die ohne BACnet Relevanz sind, aber Objekten als Parameter zugeordnet sind
  - Konfigurierbar im Reiter "Settings" entsprechender Objekte; im Reiter "Online" nicht sichtbar
  - u.a. Fallbackwerte für I/O-Module
- Property-Identifier >1000: Interne Properties ohne BACnet Sichtbarkeit:
  - weder im "Settings"- noch "Online"-Dialog sichtbar
  - interne Verwendung für spezielle Prozessdaten (z.B. "Rawlo*")

- Verwendung im SPS-Automapping [▶ 75] für prioritätsbasierte Verknüpfungen sowie einer Device-ID

In der Online Ansicht kann mit einem Mausrechtsklick ein Kontextmenü geöffnet werden.



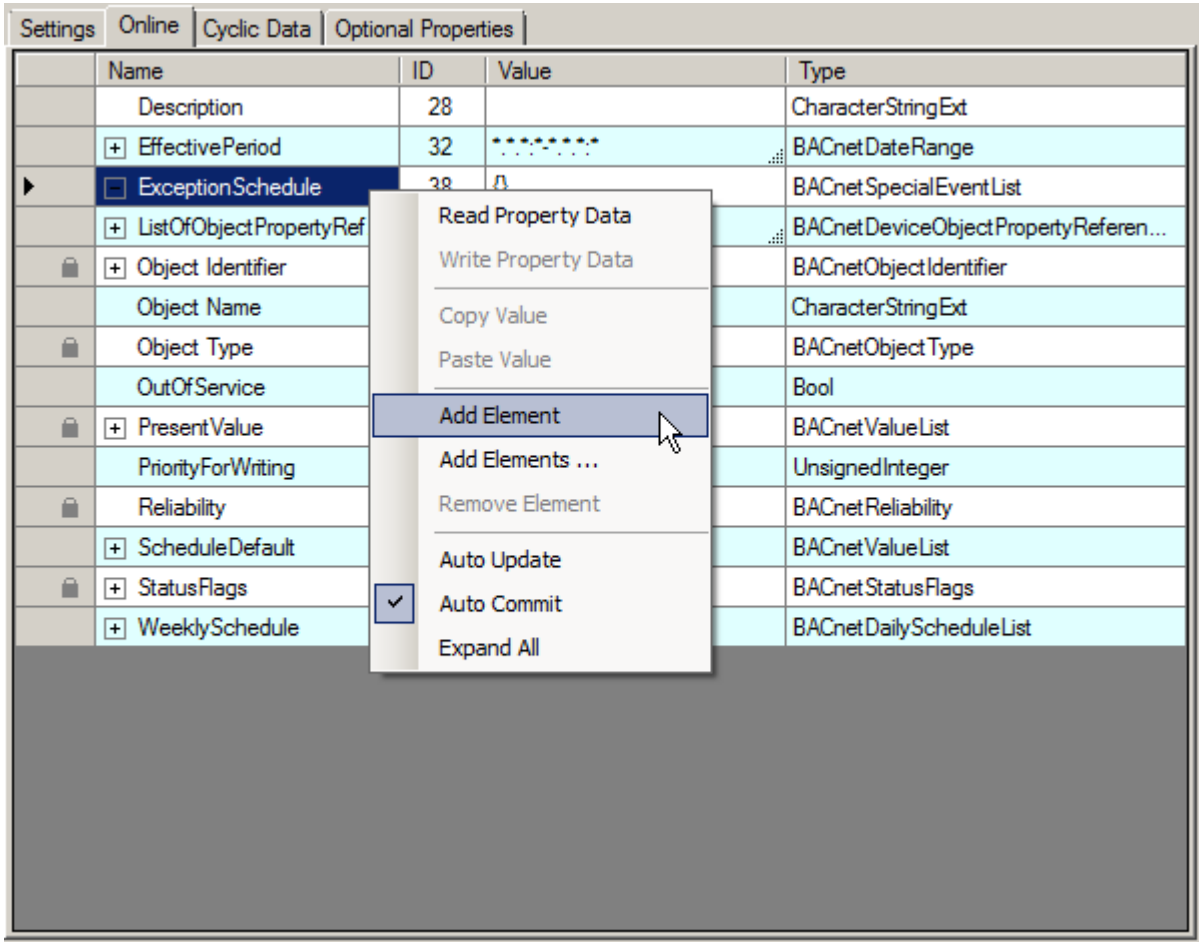
Mit dem Menüeintrag „Read Property Data“ kann der Wert einer einzelnen Property eingelesen werden.

ReadPropertyAll

Die Gesamtansicht wird mit dem BACnet-Dienst ReadPropertyAll gelesen. Wird dieser Dienst von einem Gerät nicht unterstützt, dann werden die Properties einzeln gelesen.

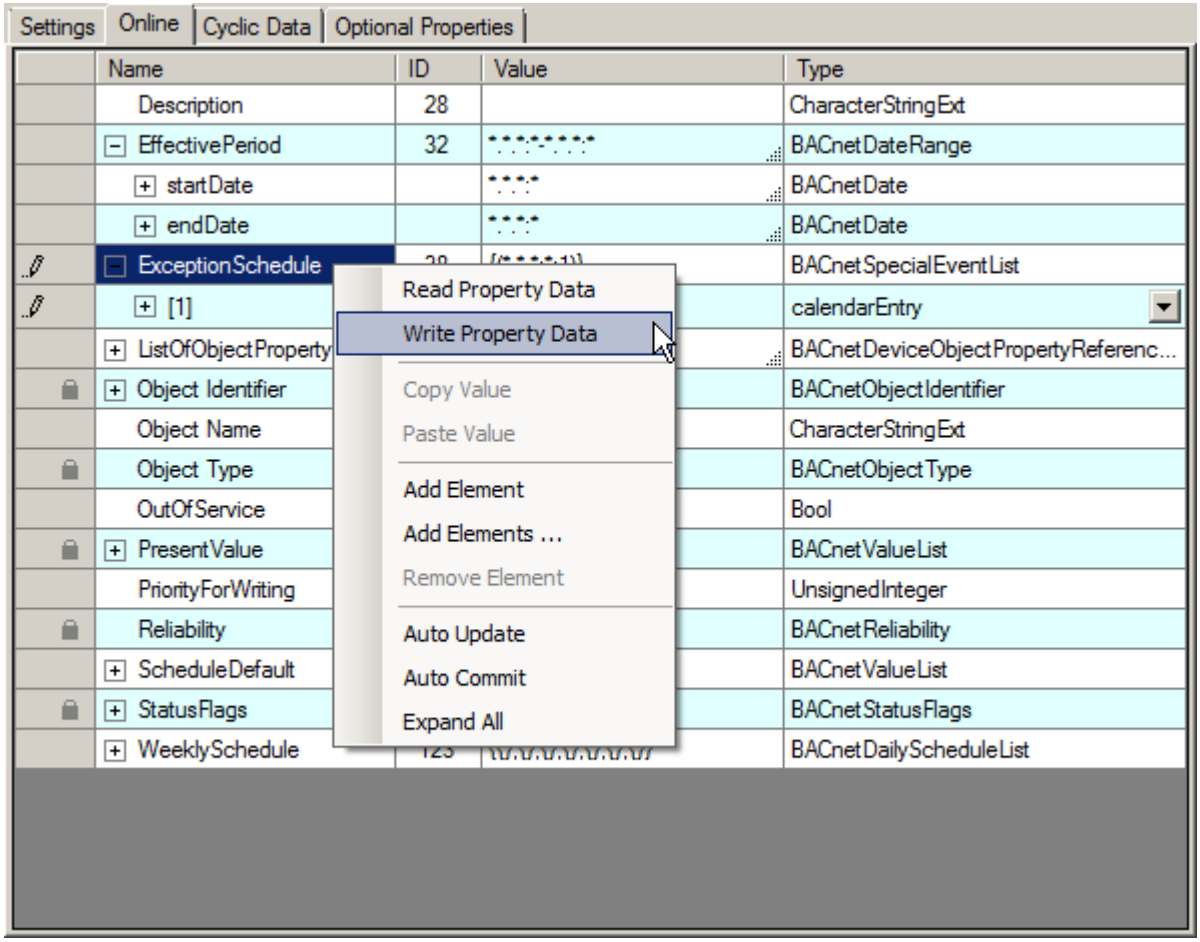
Mit der Funktion Auto-Update werden alle Properties in einem Intervall von 1 Sekunde neu eingelesen und dargestellt. Damit müssen die einzelnen Properties nicht mehr mit dem Menüeintrag "Read Property Data" explizit gelesen werden. Diese Funktion ist per Default aktiviert.

Bei Listen- und Array- Properties können mit Hilfe von „Add“ und „Remove“ Elemente hinzugefügt bzw. entfernt werden.



Im Default-Fall werden Änderungen der Property-Daten sofort geschrieben. Mit der Deaktivierung der Funktion „Auto Commit“, können Daten manipuliert werden, ohne ein sofortiges Schreiben auszulösen. Damit können komplexe Änderungen an Properties vorgenommen werden (z. B. an der Property ExceptionSchedule) und die Änderungen anschließend insgesamt geschrieben werden. Derart editierte

Properties sind mit einem Stift auf der linken Seite markiert bis die Änderungen geschrieben wurden. Mit dem Kontextmenü „Write Property Data“ können die Änderungen dann geschrieben werden. Bitte beachten: Momentan wird hier nicht WritePropertyMultiple verwendet, sondern einzelne WriteProperty-Aufrufe.

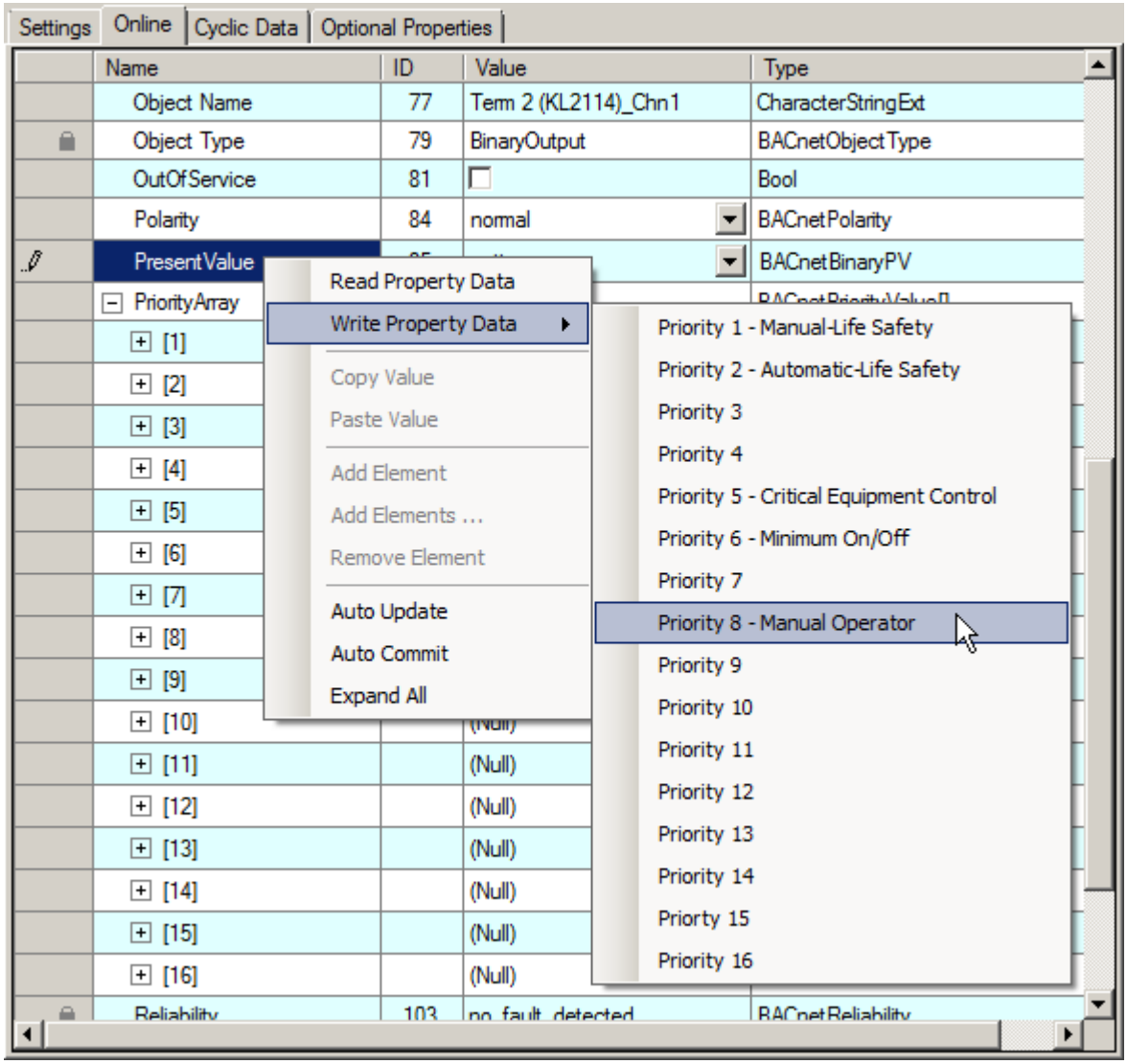


HINWEIS

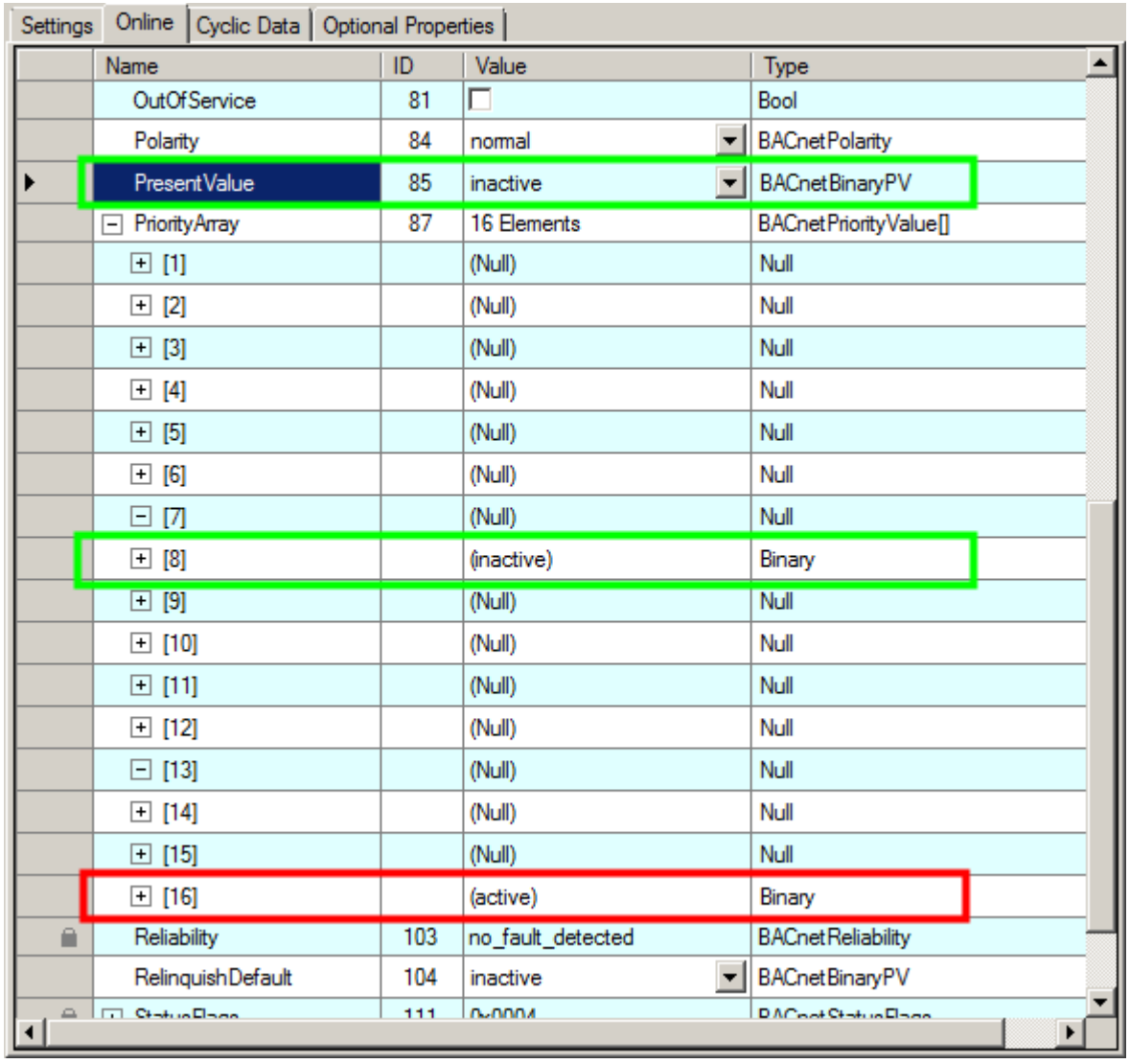
Auto Update deaktivieren!

Bei deaktiviertem „Auto Commit“ sollte auch die Funktion „Auto Update“ deaktiviert werden, da sonst schon editierte Daten beim Wiederauslesen überschrieben werden.

Bestimmte Objekte besitzen kommandierbare Properties. Bei diesen kann mit Angabe einer Priorität die Wertigkeit eines Schreibzugriffs festgelegt werden. Dieser Wert wird über den mit der Priorität verknüpften Index in das PriorityArray geschrieben.



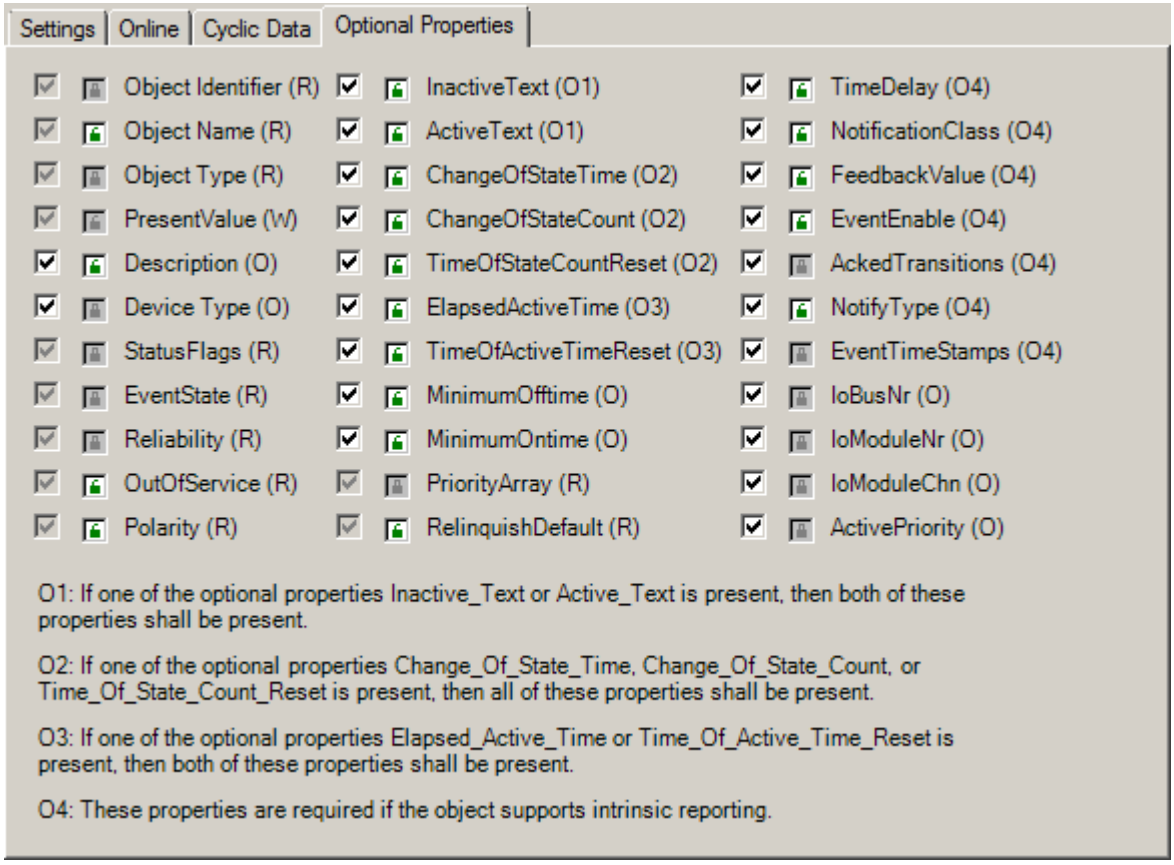
Das PriorityArray kann zudem direkt beschrieben werden. Alle Einträge mit dem Wert "(Null)" werden nicht berücksichtigt. Einträge mit konkretem Datentyp und gültigem Wert wirken, je nach Priorisierung, auf das zugehörige, kommandierbare Property oder werden von einem anderen Eintrag mit höherer Priorität übersteuert. Die Einträge aus dem PriorityArray bestimmen dann z. B. den Wert das PresentValue eines BinaryOutput-Objekts.



Um einen Eintrag aus dem PriorityArray zu entfernen, muss der Typ in der Spalte "Type" auf NULL gestellt werden. Zudem kann ein Eintrag aus dem PriorityArray mittels Kontextmenü der kommandierbaren Property gelöscht werden. Dazu muss als Typ NULL angewählt und mittels "Write Property Data" auf die entsprechende Priorität geschrieben bzw. mit "Auto Commit" automatisch geschrieben werden.

Konfiguration optionaler Properties und Property-Schreibschutz

Im BACnet-Standard sind die Standardobjekte und deren Properties definiert (Property-Identifier 0 - 511). Einige Properties sind dort als optional gekennzeichnet und andere als "Pflicht-Properties". Um eine möglichst hohe Flexibilität bei der Konfiguration von Objekten zu erreichen, besteht in der Objektansicht im Reiter "Optional Properties" die Möglichkeit, optionale Properties ein- bzw. auszublenden und die Zugriffsart, soweit erlaubt, einzuschränken (nur lesend oder lesend/schreibend).



Angaben hinter dem Property-Namen (R/W/O)

Die Angabe hinter dem Property-Namen (R/W/O) gibt die minimale Zugriffsberechtigung laut BACnet-Spezifikation an. Dabei steht "R" für lesbare (oder höhere) Properties, "W" für schreib- und lesbare und "O" für optional les- und/oder schreibbare Properties. Die Zahlenkennzeichnung von optionalen Properties gibt die Abhängigkeit zueinander an. Optionale Properties mit derselben Nummer können nur als Gruppe aktiviert oder deaktiviert werden. Abweichend von der BACnet-Spezifikation sind einige optionale Properties in TwinCAT BACnet/IP Pflicht und damit nicht abwählbar. Dies ist in der internen Implementierung begründet.

Bei einigen Properties kann zudem der Schreibzugriff festgelegt werden. Das Symbol "grünes, geöffnetes Schloss" symbolisiert Schreibberechtigung. Das Symbol "rotes, geschlossenes Schloss" bedeutet nur Leseberechtigung. Durch Klick auf das Symbol toggelt der Zustand. Bei "ausgegrautem" Symbol ist die Umstellung der Berechtigung nicht möglich. Die Zugriffsberechtigung ist an der Kennzeichnung "W" bzw. "R" oder am ausgegrauten Schlosssymbol erkennbar ("W" bzw. "geöffnetes, graues Schloss = Schreib- und Leseberechtigung / "R" bzw. geschlossenes, graues Schloss = nur Leseberechtigung).

Die Verfügbarkeit von optionalen Properties kann durch Klick auf das Checkbox-Symbol gesteuert werden. Durch Abwahl der Checkbox wird die zugehörige Property deaktiviert.

Herstellerspezifische Properties

TwinCAT BACnet/IP verwendet eine Reihe herstellerspezifischer Properties, die im Folgenden genauer erläutert werden sollen. Dabei werden herstellerspezifische Properties unterschieden durch ihre Property-IDs. Properties im Bereich 512 bis 799 sind BACnet-weit sichtbar. Properties in höheren Bereichen werden für die Konfiguration innerhalb von TwinCAT BACnet/IP verwendet; via BACnet aber nicht angezeigt. Diese Properties sind nur im Reiter "Settings" sichtbar - nicht aber im Reiter "Online".

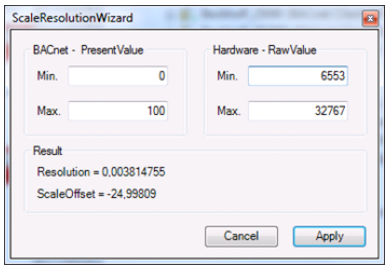
ScaleOffset (512)	ScaleOffset wird bei AnalogInput- und AnalogOutput-Objekten verwendet und bestimmt den Offset bei der Umrechnung vom Hardware-Wertbereich nach BACnet und umgekehrt.

	Für AnalogInput-Objekte gilt: PresentValue = RawIoSingedValue * Resolution + ScaleOffset Für AnalogOutput-Objekte gilt: RawIoSingedValue = PresentValue / Resolution - ScaleOffset Die gleichen Berechnungsalgorithmen gelten für die Prozessdaten-Properties RawIoUnsingedValue.
IoBusNr [0..255] (513)	IoBusNr wird für die Abbildung eines Feldbus-Zustands u.a. beim K-Bus I/O-Automapping verwendet. Details hierzu können in den Abschnitten IO Automapping [▶ 75] und Prozessdaten [▶ 43] nachgelesen werden. Der 255 gibt an, dass ein BACnet-Objekt keinem IoBus zugeordnet ist.
IoModuleNr [0..65535] (514)	IoModuleNr gibt die Position einer Klemme innerhalb eines I/O-Bus an. Beim I/O-Automapping wird mit 1 beginnend durchnummeriert. Diese Property hat rein informativen Charakter und wird im BACnet-Stack nicht verwendet.
IoModuleChn [0..255] (515)	IoModuleChn gibt die Kanalnummer einer I/O-Klemme an. Beim I/O-Automapping wird mit 1 beginnend durchnummeriert. Diese Property hat rein informativen Charakter und wird im BACnet-Stack nicht verwendet.
PersistentData (516)	PersistentData gibt an, ob seit dem letzten Speichern persistenter Daten Änderungen aufgetreten sind. Zusätzlich kann über diese Property via BACnet das Speichern persistenter Daten ausgelöst werden. Details zu dieser Property finden sich im Abschnitt Persistente Daten [▶ 60]. Diese Property ist nur sichtbar, wenn persistente Daten aktiviert wurden und die Konfiguration im RUN Modus geladen wurde.
ActivePriority (517)	ActivePriority gibt bei Objekten mit kommandierbaren PresentValue-Properties an, welche Prioritätsstufe gerade aktiv ist. Sind alle Einträge im PriorityArray = NULL ist der Wert von ActivePriority = 17 und ReliquishDefault wird als Wert für das PresentValue verwendet.
LastConfirmedServiceAccess (518)	Mit Hilfe der Property LastConfirmedServiceAccess des Device-Objekts wird in Form einer Zeichenkette die IP-Adresse des letzten erfolgreich ausgeführten bestätigten Dienstens (z.B. Property lesen, schreiben) angezeigt. Über ein TrendLog-Objekt im COV-Modus kann so ein Zugriffslog eingerichtet werden, der die Zugriffe im Netzwerk dokumentiert.
DataRequestMode (519) DataPollingInterval (520) EnrollmentCovResubscriptionInterval (521)	Beim EventEnrollment-Objekt externe Objekte überwacht werden. Mit Hilfe dieser Properties kann die Datenübertragungsmethode eines EventEnrollment-Objektes konfiguriert werden. Die Property DataRequestMode vom Datentyp Enum legt fest über welchen Dienst Daten gelesen werden (ReadProperty, COV, ReadPropertyMultiple). Die Property DataPollingInterval legt fest in welchem zeitlich Abstand bei den Übertragungsarten ReadProperty und ReadPropertyMultiple Daten angefordert werden. EnrollmentCovResubscriptionInterval legt das ResubscriptionIntervall für den Modus COV fest.
FileName (522)	Beim File-Objekt wird die zugehörige Datei über die Property ObjectName identifiziert. Wird aber vom Betreiber ein festes Namensschema auch für File-Objekt vorgeschrieben, ist die Zuordnung der Datei über die Property ObjectName nicht mehr möglich. Die Property FileName kann optional aktiviert werden. Wenn diese Property aktiv ist, wird über den Wert der Zeichenkette die zugehörige Datei, die sich immer relativ zum Boot-Ordner befindet, zugeordnet und die Property ObjectName kann frei verwendet werden.

Voraussetzungen

FaultFallbackRealValue (802)	FaultFallbackRealValue kann bei AnalogInput-Objekten verwendet werden, um im Falle eines I/O-Bus-Fehlers einer übergelagerten SPS bzw. BACnet gültige Werte zu liefern. Wird ein IoBus-Fehler festgestellt werden von TwinCAT BACnet/IP nicht mehr die entsprechenden Prozessdaten des Feldbus, sondern die konfigurierte Fallbackwert kopiert. ⊟ FaultFallbackRealValue 802 (100) FallbackRealValue fallbackValue ☑ 100 Real

	Der Fallbackwert kann im Reiter "Settings" aktiviert werden indem der entsprechende Haken des optionalen Wertes aktiviert wird. Ist ein Fallbackwert nicht aktiv, werden bei einem IoBus-Fehler keine Prozessdaten mehr kopiert und der letzte gültige Wert ist aktiv.
FaultFallbackBinaryValue (803)	FaultFallbackBinaryValue hat die gleiche Bedeutung wie FaultFallbackValueReal und wird bei BinaryInput-Objekten verwendet.
OutOfServiceFallbackRealValue (804)	OutOfServiceFallbackRealValue kann verwendet werden um gültige Ausgangsprozessdaten zu erzeugen wenn ein AnalogOutput-Objekt OutOfService geschaltet wird. Ist OutOfServiceFallbackRealValue aktiv wird der entsprechende Wert für die I/O-Prozessdaten verwendet, andernfalls wird beim aktivem OutOfService der letzte gültige PresentValue-Wert verwendet.
OutOfServiceFallbackBinaryValue (805)	OutOfServiceFallbackBinaryValue hat die gleiche Bedeutung wie OutOfServiceFallbackRealValue und wird bei BinaryOutput-Objekten verwendet.
TimeSynchronizationUseUTC (806) *Seit Revision 12 ist diese Property nicht mehr vorhanden. Es werden nun die spezifizierten Properties UTCTimeSynchonisationRecipients und TimeSynchronisationRecipients unterstützt. Die beschriebene Funktionalität bleibt erhalten.*	TimeSynchronizationUseUTC wird bei Konfiguration eines TimeMasters verwendet und gibt an ob beim Versenden von Synchronizationstelegrammen an die TimeSynchronizationRecipients der TimeSynchronization-Dienst oder UTCTimeSynchronization-Dienst verwendet wird. TimeMaster-spezifische Properties sind in der von TwinCAT BACnet/IP implementierten Revision 6 noch nicht vorhanden. In diesem Zusammenhang werden von TwinCAT BACnet/IP aber die standardkonformen Properties AlignIntervals, IntervalOffset und TimeSynchronizationInterval ohne BACnet-Sichtbarkeit implementiert. Die Property TimeSynchronizationInterval (204) gibt an in welchem Abstand (in Minuten) Zeitsynchronisationstelegramme versendet werden. Ist der Wert dieser Property 0 ist die TimeMaster-Funktion deaktiviert. Mit Hilfe der Property AlignIntervals (193) wird die Property IntervalOffset (195) freigeschaltet, über die festgelegt werden kann wann genau Zeitsynchronisationstelegramme versendet werden. Hat die Property TimeSynchronizationInterval einen durch 60 teilbaren Wert, werden Zeitsynchronisationstelegramme versendet, wenn die aktuelle Uhrzeit in Minuten Modulo dem Wert von TimeSynchronizationInterval dem Wert von IntervalOffset entspricht. Im folgenden Beispiel ist die Konfiguration eines Timemasters erläutert, der jeden Tag um 2 Uhr Nachts UTC-Zeitsynchronisationstelegramme an alle BACnet-Geräte im Netzwerk versendet: TimeSynchronizationUseUTC = TRUE AlignIntervals = TRUE TimeSynchronizationInterval = 1440 IntervalOffset = 120 TimeSynchronizationRecipients = {((FF:FF:FF:FF:FF:FF;0))}
PredictScheduleValueTime [sec] (807) PredictedScheduleBoolValue (808)	In der Gebäudeautomatisierung können heiz- und klimatechnische Anlagen durch ein "optimiertes/gleitendes" Einschalten Energie sparen. Wird z.B. eine Solltemperatur von 21°C um 8.00 Uhr gefordert, kann eine Heizung je nach Jahreszeit früher oder später aktiviert werden. Bei sehr niedrigen Außentemperaturen muss früher als bei höheren Außentemperaturen geheizt werden. Mit entsprechenden mathematischen Formel oder lernenden Systemen kann berechnet werden wieviel im Voraus eine Analage aktiviert werden muss. Für die Umsetzung dieser Funktionalität stehen im BACnet-Objekt Schedule die herstellerspezifischen Properties *PredictScheduleValueTime* und *PredictedScheduleBoolValue* zur Verfügung. Die berechnete vorzeitige Einschaltung der Anlage kann über die Prozessdaten-Property PredictScheduleValueTime von der SPS an BACnet übergeben werden. Über die Prozessdaten-Property PredictedScheduleBoolValue berechnet der BACnet-Stack den vorausgesagten Wert (PresentValue) des Schedule-Objekts in der Zukunft. Einheit von *PredictScheduleValueTime* ist Sekunde. Ist es z.B. gerade 7:51 Uhr und der Wert von PredictScheduleValueTime ist 540 (9 Minuten), dann wird PredictedScheduleBoolValue den *Wert desPresentValue* des Schedule-Objekts um 8.00 Uhr enthalten.

AccumulatorIntegrationMode (811)	Mit dem Accumulator-Objekt können über die Prozessdaten-Property RawIoAccumulatorValueUSINT Zählwerte akkumuliert werden. Im normalen Modus wird dabei der Zählwert des Accumulators um 1 erhöht wenn sich der Wert von RawIoAccumulatorValueUSINT um 1 erhöht. Für die Messung von Mengen pro Zeit kann der Integrationsmodus akiviert werden. Dann wird der Zählwert des Accumulators erhöht wenn der Wert von RawIoAccumulatorValueUSINT über eine Millisekunde 1 war. Mit dieser Funktionalität kann z.B. eine Luftmenge berechnet werden, wenn RawIoAccumulatorValueUSINT mit dem aktuellen Luftstrom verknüpft wird. Die Zeitbasis ist dabei 1 ms. Mit Hilfe der BACnet-Property *PreScale* können Skalierungen z.B. auf eine Stundenbasis vorgenommen werden.
AvgFilterCycles (812)	Bei analogen Eingangswerten kann es auf Grund von langen Leitungen oder Störeinflüssen leicht zu schwankenden Werten kommen. BACnet bietet bei AnalogInput-Objekten die Möglichkeit die Übertragung von Wertänderungen durch die Property CovIncrement zu begrenzen. Liegen die Schwankungen aber über der darzustellenden Auflösung von Werten kann es sinnvoll sein, die analogen Eingangswerte durch Filter zu glätten. Mit der Property AvgFilterCycles kann ein mittelwertbildender Filter für AnalogInput-Objekte aktiviert werden, der über AvgFilterCycles Zyklen den Mittelwert des Raw-Wertes bildet. Bei ein BACnet-Zykluszeit von 50ms, bedeutet ein Wert von AvgFilterCycles = 20 einen Filter über 1 Sekunde.
FaultDeadband (813)	Ein weiteres Problem von schwankenden Analogwerten, ist ein mögliches Verlassen eines als gültig definierten Wertebereichs. Beim AnalogInput-Objekten bestimmen die Properties MinPresValue und MaxPresValue ein gültiges Band innerhalb dessen sich der Wert der Property PresentValue befinden darf. Liegt der aus dem Raw*-Wert sklalierte Wert außerhalb des definierten Bereichs geht das BACnet-Objekt in den Fault-State. Da die Properties MinPresValue und MaxPresValue auch von der Leittechnik als Wertebereich ausgelesen werden, sollten diese Grenzwerte nicht verschoben werden, um leicht unter die Grenze schwankende Werte noch als gültig zu tolerieren. Mit Hilfe der Property FaultDeadband kann ähnlich der Property Deadband, die den Übergang in den OffNormal-Zustand bestimmt, eine erweiterte Grenze des Fault-Zustandes festgelegt werden. Unterschreitet der durch die Skalierung berechnete Wert den Wert von MinPresValue um weniger als FaultDeadband, wird das PresentValue den Wert MinPresValue annehmen und das Objekt nicht in den Fault-Zustand wechseln. Die genaue Berechnungsformel ist wie folgt implementiert: WENN MaxPresValue + FaultDeadband > PresentValue > MaxPresValue DANN PresentValue = MaxPresValue WENN MinPresValue - FaultDeadband < PresentValue < MinPresValue DANN PresentValue = MinPresValue Im Folgenden ist ein kleines Beispiel skizziert, welches die Anwendung dieser Property demonstriert: **Beispiel:** - Analoges Eingangssignal 2V-10V - Eingangsklemme 0V-10V [0..32767] - Abbildung auf 0% - 100% (PresentValue) - Es sollen Werte bis 1,8V als gültig toleriert werden ? **Rechnung:** - 2V = 6553 - 10V = 32767 - MinPresValue = 0 (0%) - MaxPresValue = 100 (100%) - FaultDeadband: 1,8V entspricht einem Wert von 5898. FaultDeadband wirkt im Wertebereich des PresentValue; Der skalierte Wert beträgt: 5898 *0,0038 – 24,99 = 2,5%. 

	Das FaultDeadband muss also einen Wert von 2,5 haben damit im Beispiel 1,8 V noch als gültiger Wert angezeigt wird.	
Pt1DampFactor [0..1] (814)	Ähnlich der Property AvgFilterCycles kann mit Hilfe der Property Pt1DampFactor ein analoger Eingangsfilter aktiviert werden. Der Pt1-Filter wird nach der Skalierung angewendet und dämpft den Analogwert. Der Dämpfungsfaktor kann Werte zwischen 0 und 1 annehmen. Bei 1 findet keine Filterung statt, bei einem Wert von 0 wird der neue Eingangswert erst im Unendlichen angenommen. Sinnvolle Werte können z.B. 0,01 oder 0,001 sein. Über ein TrendLog-Objekt kann die Wirkung des Filters leicht für eine einfache Impulsantwort überprüft werden. (Raw-Wert von 0 auf 1 ändern). Die implementierte Formel lautet: $PresentValue_{neu}$=Pt1DampFactor*(Eingang-PValt)+$PresentValue_{alt}$ PT1 und mittelwertbildender Filter können parallel betrieben werden. Der mittelwertbildende Filter ist laufzeiteffizienter, da Ganzzahlarithmetik zur Anwendung kommt.	
EnableInternalLoopCtrl (815)	Über die Property EnableInternalLoopCtrl kann ein interner Regelungsalgorithmus im Loop-Objekt aktiviert werden. Im Normalfall wurde das Loop als leere Hülle für einen in der SPS implementierten Regler umgesetzt. Um auch die dynamische Erzeugung von Loop-Objekten unterstützen zu können, wurde eine Basisregelung in den BACnet-Stack integriert. Dieser PID-Regler verknüpft die P,I und D-Anteile additive und setzt eine Begrenzung für den Integral- sowie den Summenanteil um. Im folgende ist die implementierte Formel des Loop-Objekts bei aktvierter (und auf TRUE gesetzer) Property EnableInternalLoopCtrl dargestellt: `IF (Action = = direct) E = W - X IF (Action = = reverse) E = X - W Yp = Kp * E // Proportional Yi = Yi(n-1) + (Ki * tCycle * E) // Integral Max >= Yi >= Min // Anti-Wind-Up Yd = Kd * (E - E(n-1)) / tCycle // Differenzial Y = Yp + Yi + Yd + Bias Max >= Y >= Min // Ausgangslimitierung`	
Pt1DerivativeDampFactor [0..∞] (816)	Um Sprünge bei der Sollwertvorgabe des intern implementierten Regelalgorithmus im Loop-Objekt abzufangen, kann der Fehlerwert (E) des D-Anteils über einen Pt1-Faktor gedämpft werden. Dies ermöglicht die Wirkung des D-Anteils bei Sollwertsprüngen zu verstärken.	

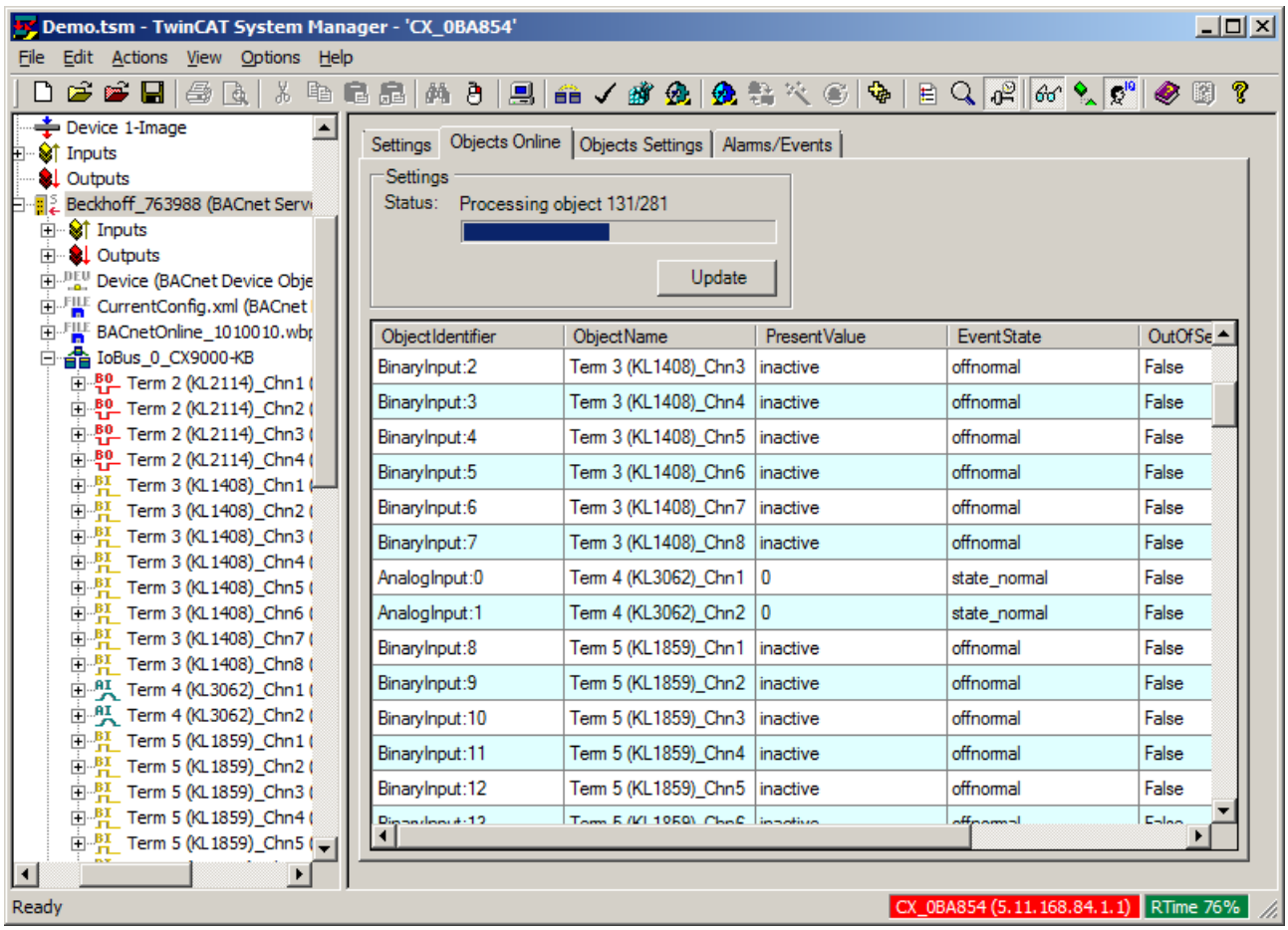
2.4 Hilfsfunktionen und Wizards

Zur vereinfachten Konfiguration und Übersicht von TwinCAT BACnet/IP wurden einige Wizards implementiert, die in diesem Abschnitt kurz vorgestellt werden sollen. Zu erkennen ist die Verfügbarkeit

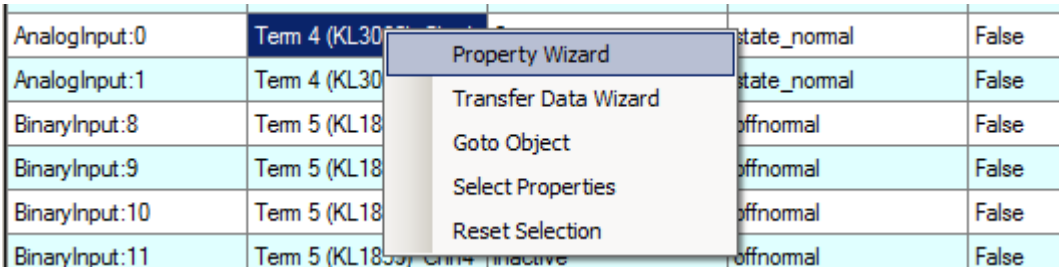
eines Wizards an einem kleinen Dreieck () rechts unten in der Property-Ansicht im Settings- und Online-Reiter der Objekte. Die Wizards funktionieren sowohl für Client- als auch Server-Objekte.

Objektübersicht

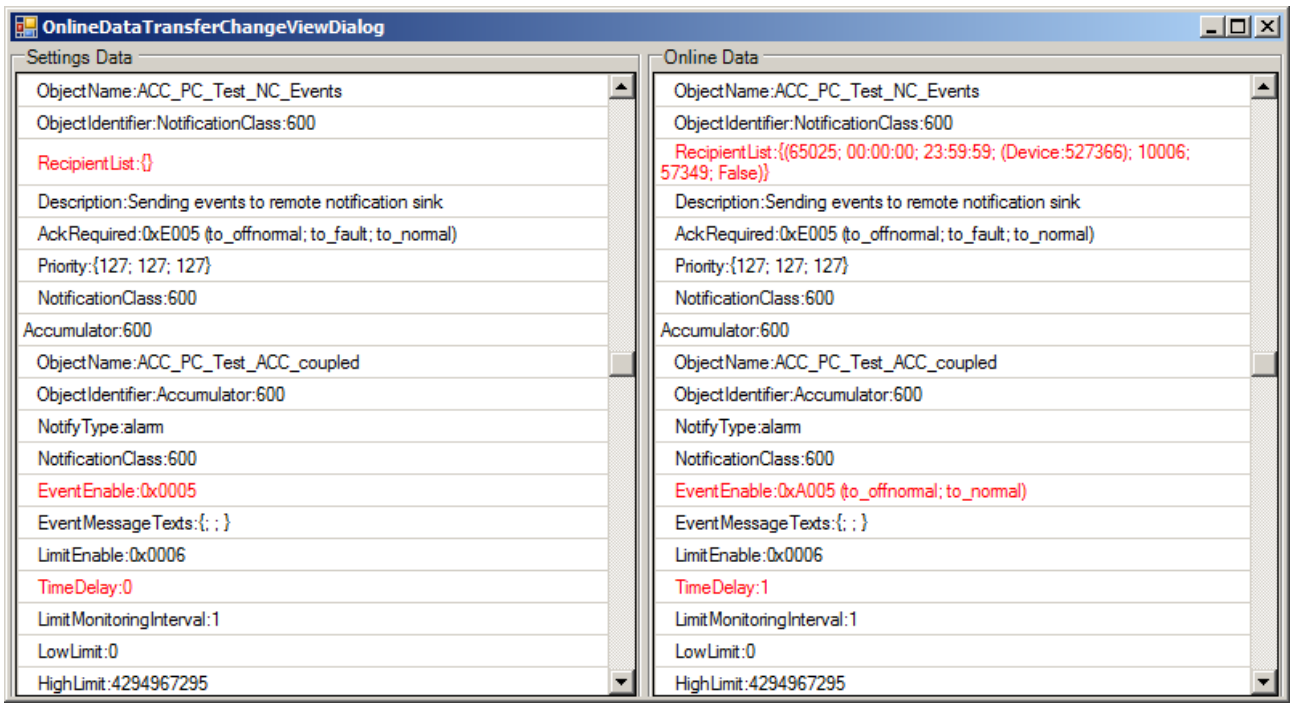
Unter BACnet-Client und -Server finden sich jeweils Reiter "Objects Online" und "Objects Settings". In dieser Ansicht werden ausgewählte Properties der Objekte eines BACnet-Client bzw. -Server dargestellt. BACnet-Objekte können durch Mausklick auf den Tabellenkopf nach verschiedenen Kriterien sortiert werden. Auf diese Weise ist es z. B. möglich, alle Objekte im Zustand *fault* zuerst anzuzeigen. Generell gilt, dass "Objects Online" alle Online-Werte der BACnet-Objekte dargestellt; "Objects Settings" stellt die Settingswerte dar. Mit Hilfe der Copy&Paste Funktionalität (Taste STRG-C, STRG-V) können Property-Werte in dieser Ansicht kopiert werden. Sind mehrere Zellen selektiert, werden bei STRG-V mehrere Property-Werte verändert.



Durch einen Rechtsklick auf die Objektübersicht kann in einem Kontextmenü für BACnet-Clients der "Prozessdaten-Wizard" bzw. für BACnet-Server der "Property-Wizard" aktiviert werden.



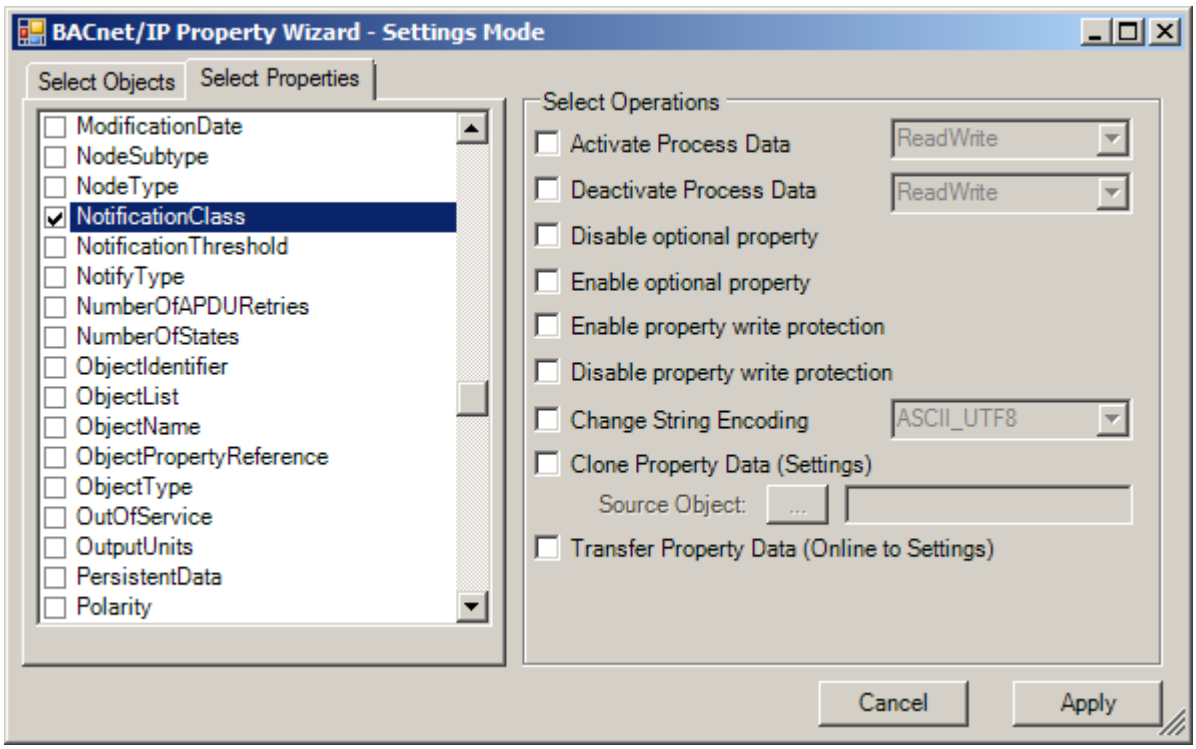
Über das Kontextmenü kann auch direkt zu einem selektierten Objekt im System Manager-Baum gesprungen werden ("Goto Object"). Über die Schaltfläche "Select Properties" kann ausgewählt werden, welche Properties in der Objektübersicht dargestellt werden. Über den Property-Transfer-Wizard können Unterschiede zwischen Settings- und Online-Daten dargestellt werden. Über ein Kontextmenü können Daten von Online nach Settings kopiert werden. Diese Funktionalität steht zusätzlich auch über den Property-Wizard zur Verfügung. Bei großen Konfigurationen kann das Öffnen des Transfer-Wizards einige Zeit in Anspruch nehmen.



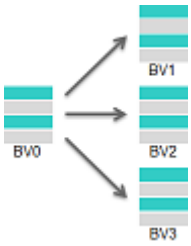
Property-Wizard

Der Property-Wizard ermöglicht das Aktivieren/Deaktivieren von Prozessdaten, Aktivieren/Deaktivieren optionaler Properties sowie das Aktivieren bzw. Deaktivieren des Schreibschutzes für mehrere Properties von mehreren Objekten. So kann z.B. sehr einfach bei allen BinaryValue-Objekten das Intrinsic Reporting (Alarme) deaktiviert werden indem die entsprechenden optionalen Properties deaktiviert werden. Für den Property-Wizard existert eine Online- und eine Settings-Variante; je nachdem über welche Objektübersicht der Wizard aktivert werden. Im Online-Modus werden alle Werte "Online" manipuliert; im Settings-Modus "Offline". Im Online-Modus stehen nicht alle Operationen zur Verfügung.

Zunächst muss festgelegt werden, für welche Objekte (Reiter "Select Objects") und welche Properties (Reiter "Select Properties") die Aktionen ausgeführt werden sollen. Durch Klick auf "Apply" werden die gewählten Operationen ausgeführt.



Mit Hilfe der Funktion "Change String Encoding" kann der Zeichensatz für multiple BACnet-Properties verändert werden. In der folgenden Tabelle wird der Unterschied zwischen den Funktionen "Clone" und "Transfer" erläutert.

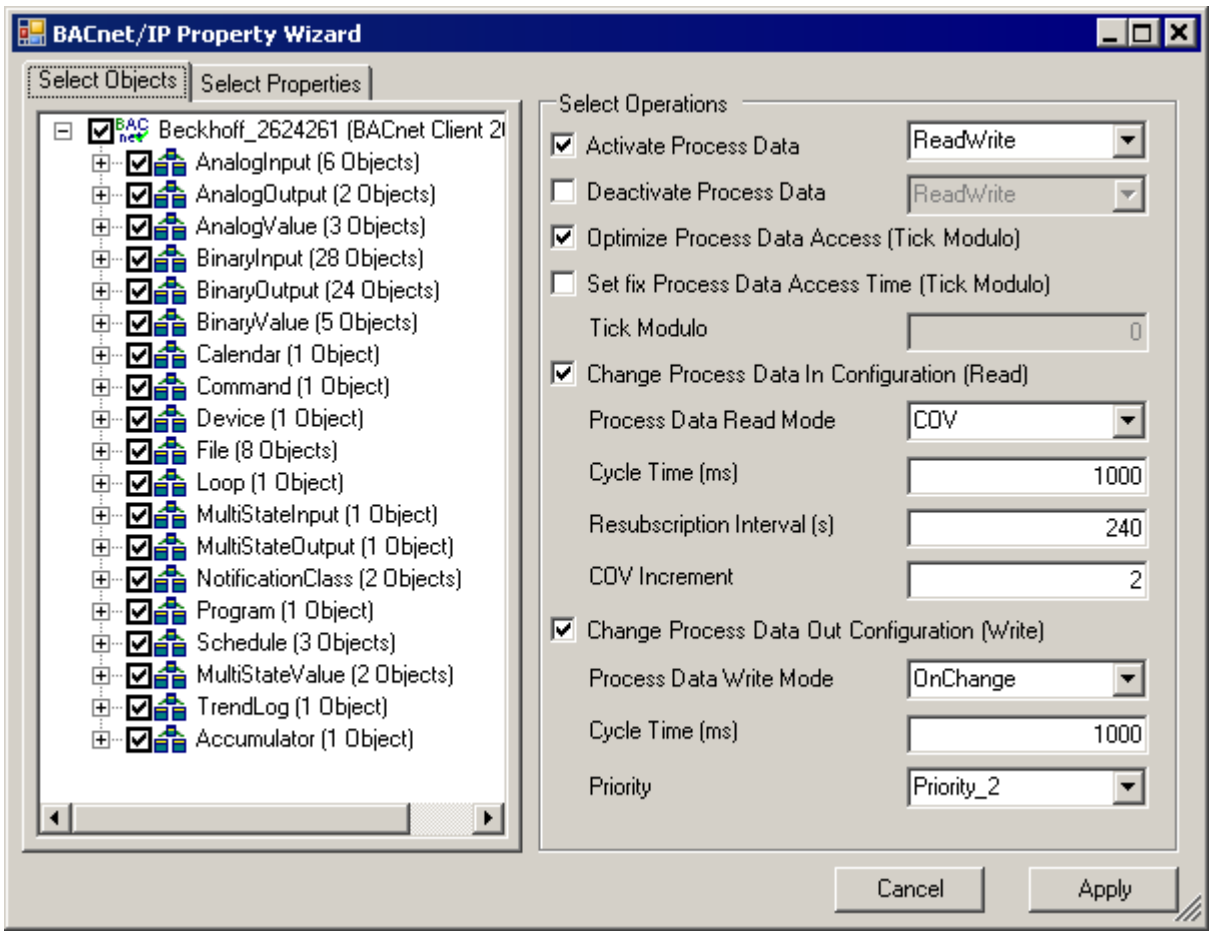
Clone 	Beim "Klonen" werden die Property-Werte eines Quellobjektes auf mehrere Ziel-Objekte übertragen. Diese Funktion kann z.B. dafür verwendet werden, die NotificationClass-Property mehrerer BACnet-Objekte gleichzeitig zu ändern. Hierfür wird ein BACnet-Objekt mit dem neuen Wert manuell beschrieben und dann als Quellobjekt ausgewählt. Ein anderes Beispiel ist das nachträgliche Aktivieren eines Glättungsfilters bei allen AnalogInput-Objekten. Da parallel zum Kopieren des Property-Wertes auch weitere Operationen parallel durchgeführt werden können, kann also die Property AvgFilterCycles aktiviert ("Enable optional Property") und gleichzeitig mit einem vorkonfigurierten Wert vom Quellobjekt initialisiert werden.
Transfer 	Beim "Transfer" werden Property-Werte bei einem Objekt (Quell- und Zielobjekt sind gleich) übertragen. Entweder von Settings nach Online (Online-Modus) oder von Online nach Settings (Settings-Modus). Diese Funktion kann z.B. verwendet werden um geänderte Online-Daten in die Konfiguration (.tsm) zu übernehmen. Die Werte aller selektierten Properties werden kopiert.

Prozessdaten-Wizard

Der Prozessdaten-Wizard ermöglicht das Aktivieren/Deaktivieren von Prozessdaten, Aktivieren/Deaktivieren optionaler Properties sowie das Aktivieren bzw. Deaktivieren des Schreibschutzes für mehrere Properties von mehreren Objekten. So kann z.B. sehr einfach bei allen BinaryValue-Objekten das *Intrinsic Reporting* (Alarme) deaktiviert werden indem die entsprechenden optionalen Properties deaktiviert werden. Für den Property-Wizard existert eine Online- und eine Settings-Variante; je nachdem über welche Objektübersicht der Wizard aktivert werden. Im Online-Modus werden alle Werte "Online" manipuliert; im Settings-Modus "Offline". Im Online-Modus stehen nicht alle Operationen zur Verfügung.

Der Prozessdaten-Wizard für BACnet-Clients ermöglicht die Konfiguration von Prozessdaten-spezifischen Einstellung von Client-Objekten. Dabei kann u.a. die Art der Prozessdatenbehandlung (COV usw.), Zykluszeiten, COV Increment usw. konfiguriert werden.

Bei sehr großen BACnet-Client-Konfigurationen (mit vielen COV-Prozessdaten) wird empfohlen die Funktion "Optimize Prozess Data Access" auf allen Objekten und Properties auszuführen. Hierbei werden automatisch die entsprechenden TickModulo-Faktoren für die Client Interaktionen berechnet. Dabei wird automatisch der verfügbare Raum an Modulo-Faktoren anhand der BACnet-Zykluszeit und der "Cycle Time" der Prozessdaten ermittelt. Bei einer BACnet-Zykluszeit von 10ms und einer Prozessdaten-"Cycle Time" von 1000 ms können die Client-Interaktionen auf 100 Zyklen aufgeteilt werden. Bei großer Konfiguration sollten die Zykluszeiten angepasst werden, um einen möglichst großen Modulo-Faktor zu erzeugen. Maximal können Client-Interaktionen über 255 Zyklen verteilt werden. Auch im COV-Modus wird die "Cycle Time" verwendet. Im Zusammenhang mit dem TickModulo wird die Anmeldung (COV-Subscription) im zugewiesenen Takt versendet. Damit können mit dem Verteilen der COV-Subscriptions über mehrere Takte, Bursts bei vielen COV-Prozessdaten reduziert werden. **Die Funktion "Optimize Prozess Data Access" wird seit Revision 12 automatisch beim Aktivieren einer Konfiguration durchgeführt und muss nicht mehr über den Wizard ausgelöst werden!**

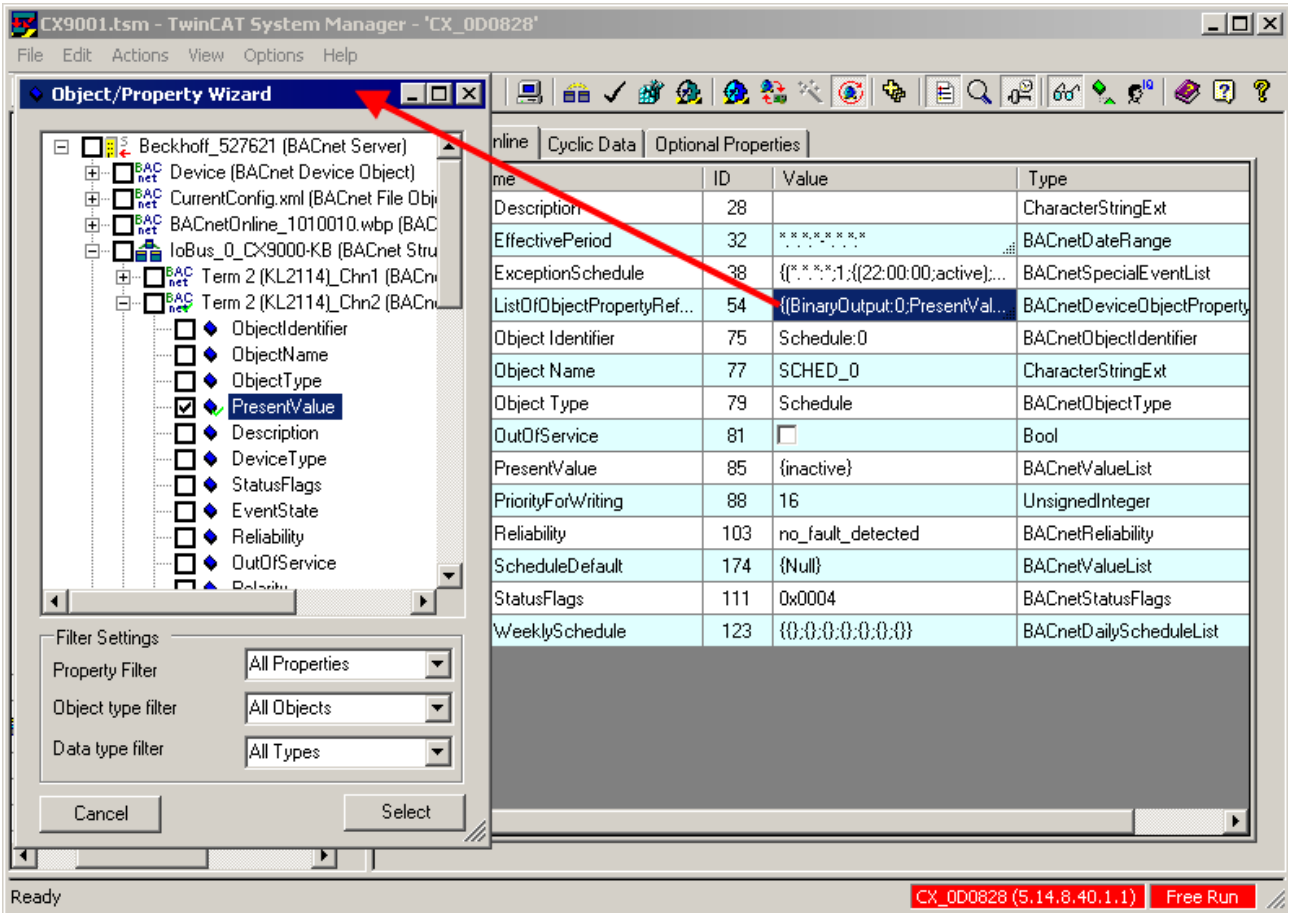


Der Prozessdaten-Wizard ist sehr gut geeignet die Art der Prozessdatenübertragung für mehrere ggf. alle BACnet-Remote-Objekte umzukonfigurieren.

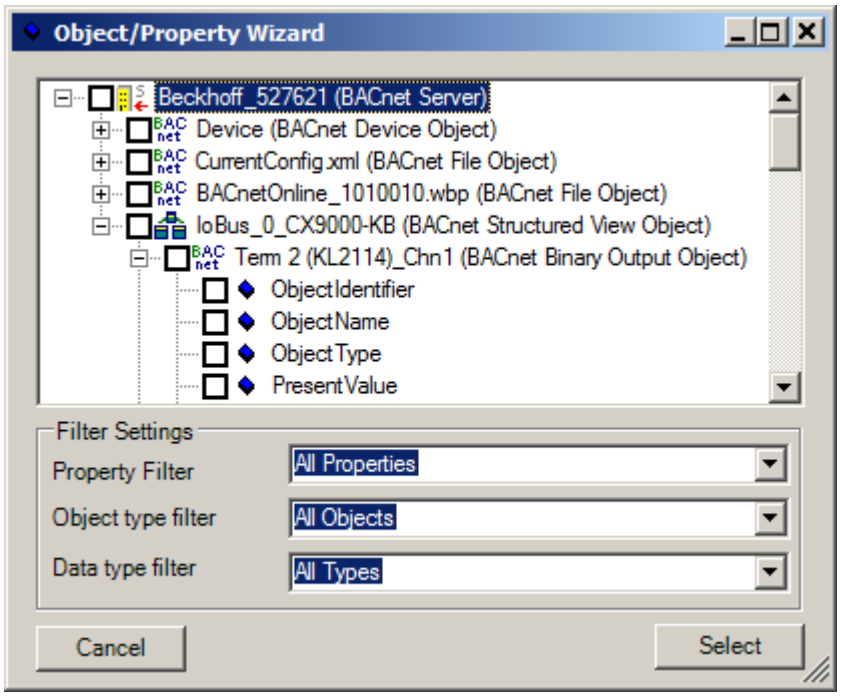
Property-Referenz Wizard

Um die Arbeit mit Property- und Objektreferenzen zu erleichtern, ist es möglich, diese Properties mit einem **Doppelklick auf ihr** Value zu konfigurieren. Nach dem Doppelklick wird ein Wizard-Dialog geöffnet. Die Verfügbarkeit dieses ist anhand eines Dreiecksymbols in der rechten unteren Ecke des Value-Feldes

erkennbar  . Beispiele sind die ListOfObjectPropertyReferences-Property im Schedule-Objekt oder die LogDeviceObject-Property im Trendlog-Objekt. Der Wizard-Dialog vereinfacht die Auswahl der entsprechenden Properties bzw. Objekte.

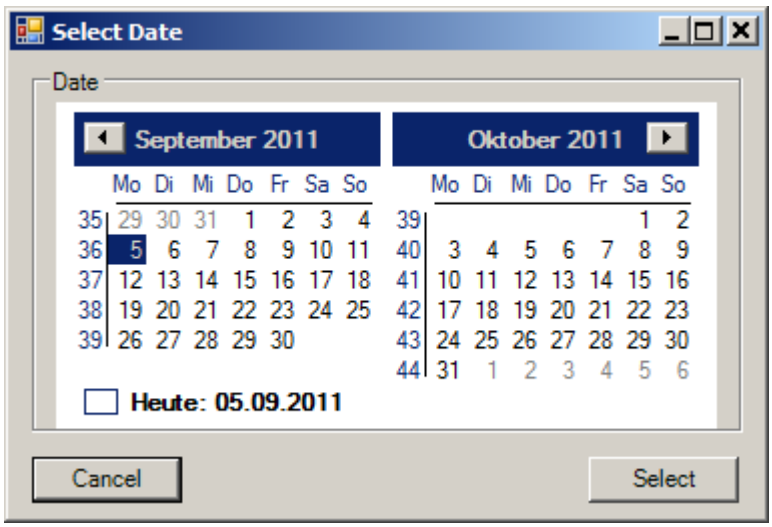


Durch das Einstellen von Filter-Optionen kann die Auswahl mehrerer Properties bzw. Objekte erleichtert werden. Bei Änderungen der Filter-Einstellungen wird die Auswahl möglicher Properties bzw. Objekte dynamisch angepasst.



Kalender-Wizard

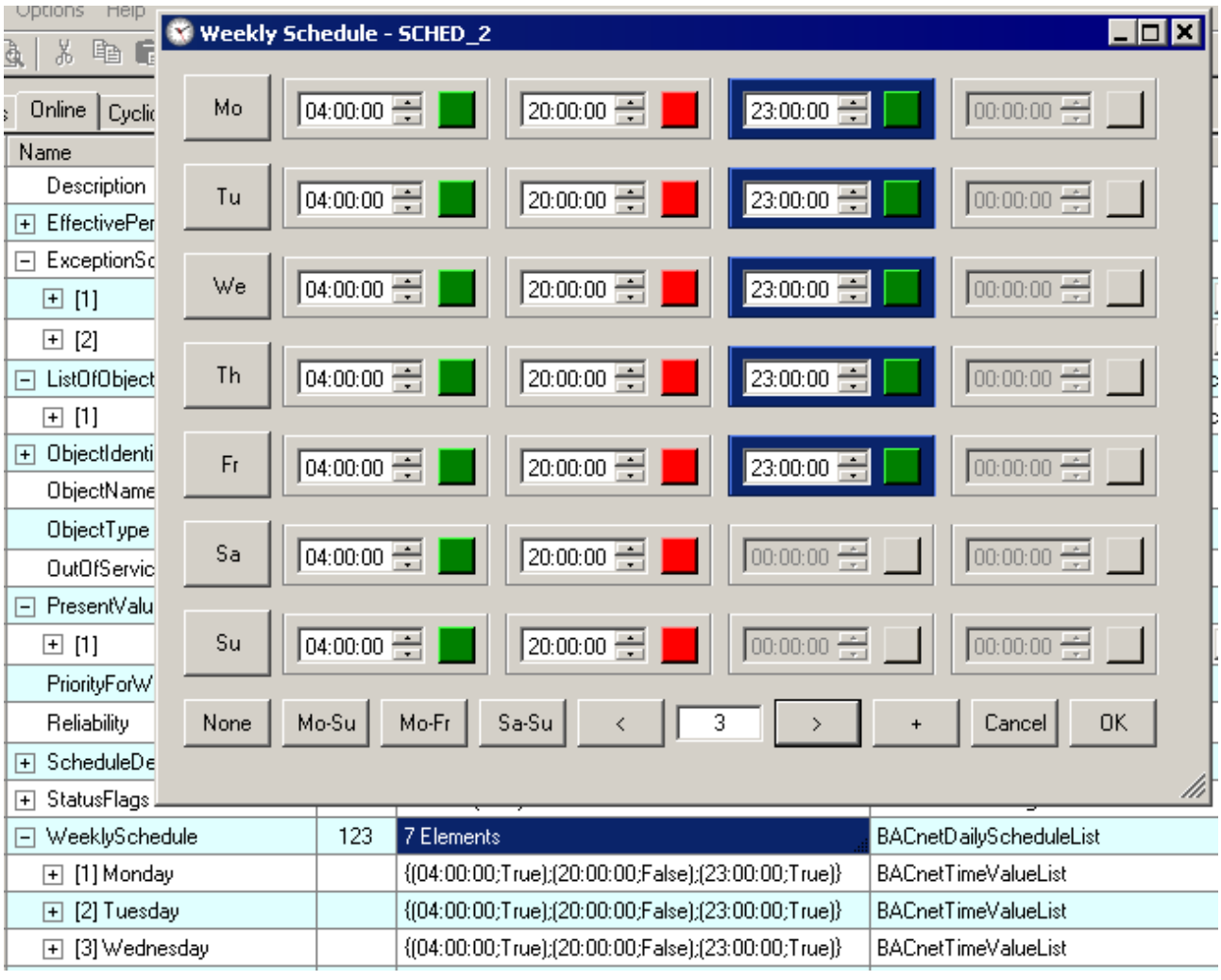
Properties mit dem Datentyp BACnetDate können über den Kalender-Wizard konfiguriert werden. Hierbei kann über einen eingeblendeten Kalender ein Tag selektiert werden. Der entsprechende Wochentag wird automatisch ermittelt. Für Daten mit dem Datentyp BACnetDateRange können auch Datumsbereiche mit dem Wizard konfiguriert werden.



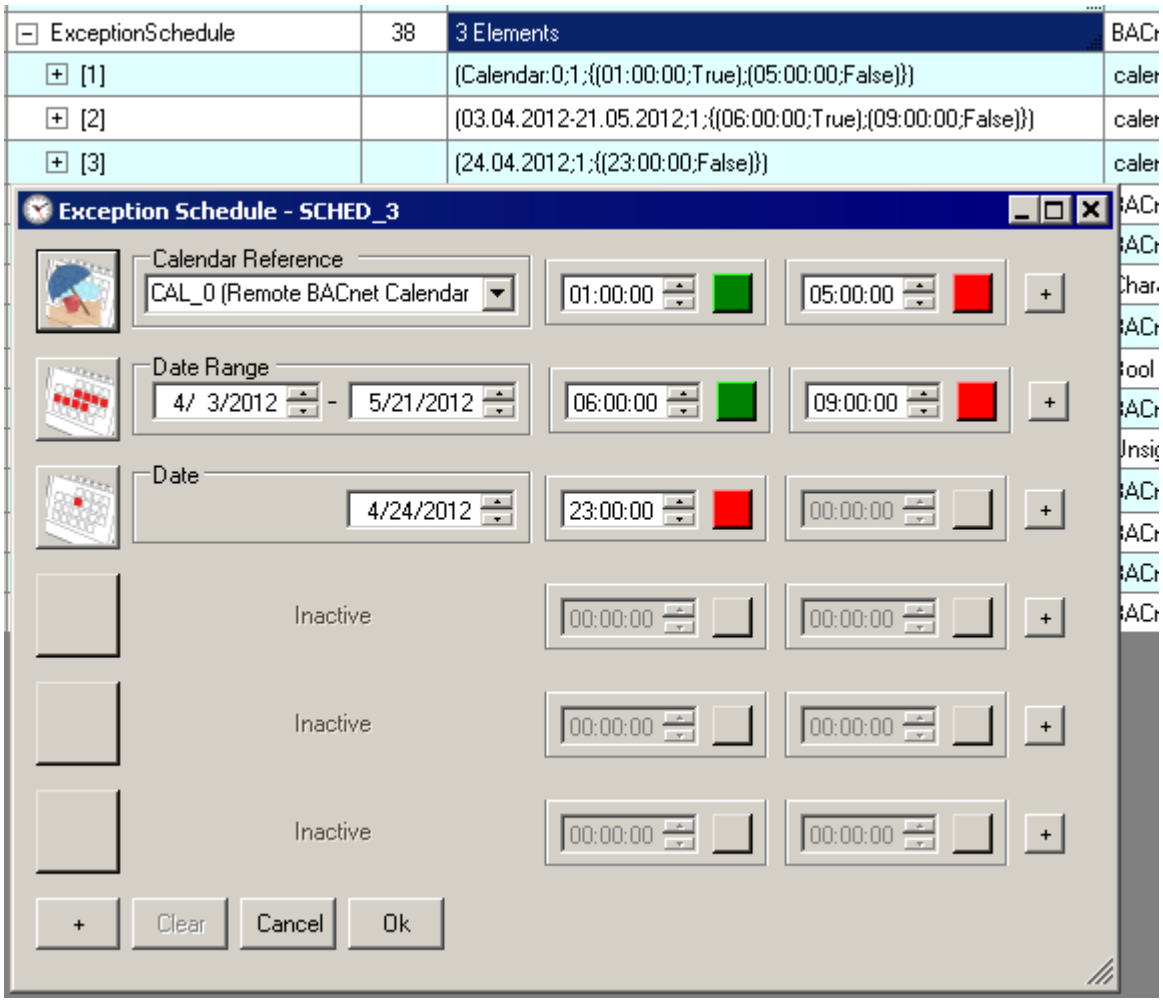
Schedule-Objekt Wizards

Für die Konfiguration von Zeitschaltplänen existieren Wizards für die Properties WeeklySchedule und ExceptionSchedule. Momentan sind diese Wizards nur verfügbar, wenn als Property-Referenz eine Property vom Datentyp BinaryPV oder Boolean eingestellt ist. Zusätzlich dürfen in den Zeiteinträgen keine Wildcards ("*") projektiert worden sein.

Der WeeklySchedule-Wizard unterstützt das Editieren von mehreren Tag-Eintragen auf einmal. Durch die Selektion von "Mo-Su", "Mo-Fr" bzw. "Sa-Su" können mehrere Wochentage selektiert werden. Änderungen eines Eintrags werden dann automatisch auf allem selektierten Eintrage übertragen. Per Mausklick können auch beliebige andere Einträge selektiert und deselektiert werden.

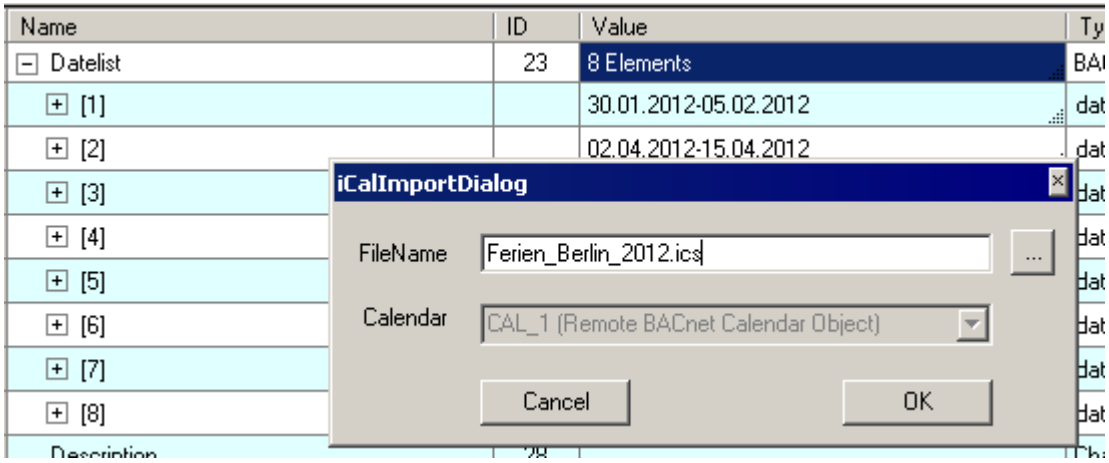


Im ExceptionSchedule-Wizard können Ausnahmeeinträge editiert werden. Dabei können 4 Modi selektiert werden: Calendar-Referenz, Date, DateRange und WeekNDay. Durch Klick auf die "+"-Symbole können weitere Einträge hinzugefügt werden.



Calendar/Schedule iCal/.ics Import

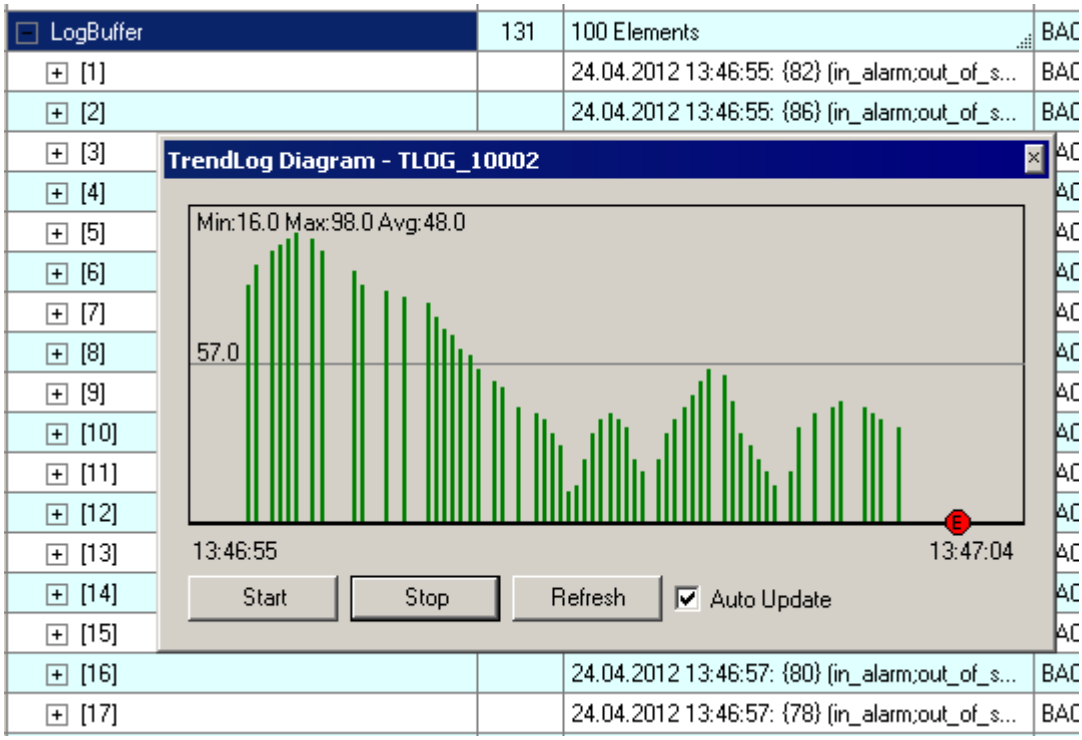
Für die effiziente Erstellung von Ferienkalendern wurde der .ics-Import implementiert. Im Internet können die entsprechenden .ics-Dateien für die Ferien heruntergeladen und importiert werden. Der .ics-Import kann als Wizard der Datelist-Property der Calendar-Objekte aktiviert werden. Zusätzlich kann über das BACnet-Device über die Schaltfläche "Import" .ics ausgewählt werden und die Einträge entweder in einen neuen oder existierenden Kalender übernommen werden.



Auch für die Properties ExceptionSchedule und WeeklySchedule wird der .ics-Import unterstützt. Über die Windows Drag&Drop-Funktion kann eine .ics-Datei auf eine der Properties gezogen werden. Es werden dann ensprechende boolesche Einträge für jeden enthaltenen Zeitraum importiert. .ics können z.B. über das Programm Microsoft Outlook erzeugt werden.

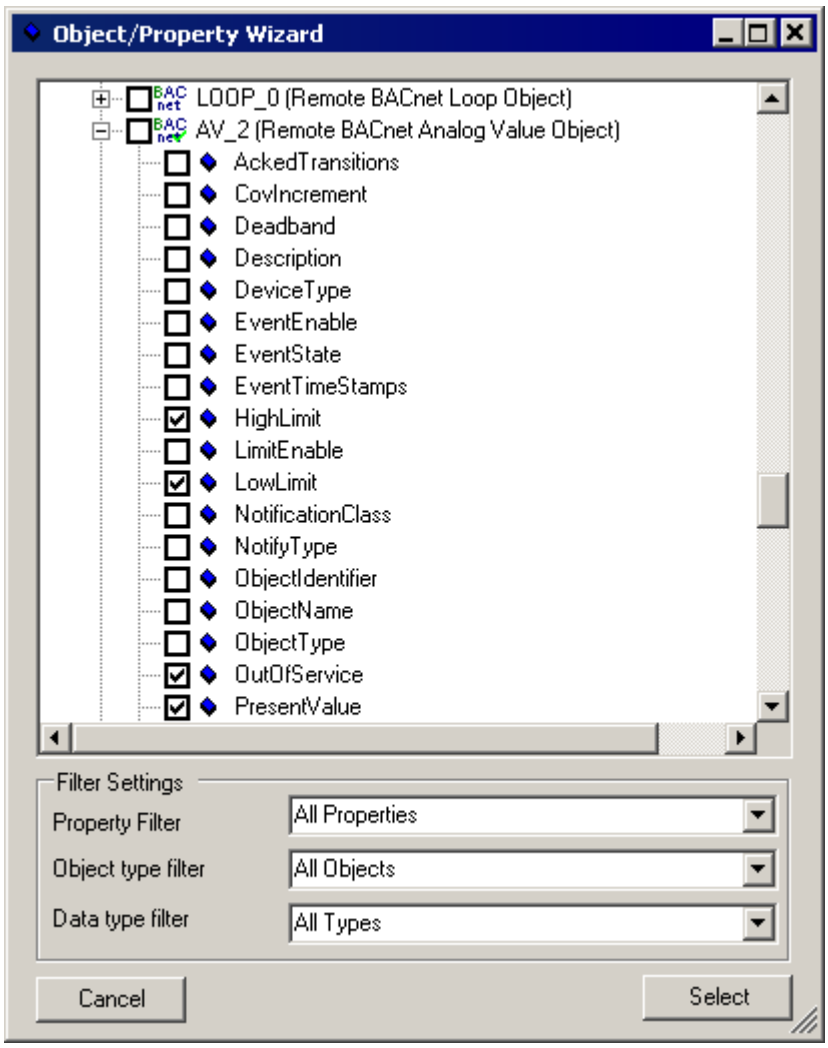
TrendLog-LogBuffer Wizard

Die Property LogBuffer der TrendLog-Objekte enthält die Logging-Einträge. Um die Übersicht zu erhöhen, wurde eine Diagramm-Darstellung für TrendLog-Objekte als Wizard integriert. Im Diagramm werden neben Durchschnitts-, Min-, Max-Werten die Einträge des LogBuffers als Linien dargestellt. Zusätzlich werden besondere Einträge (Start - B, Stop - E, Zeitsynchronisation - S) auf der X-Achse des Diagramms dargestellt. Eine Besonderheit dieses Wizard ist, dass mehrere Fenster parallel dargestellt werden können. Zusammen mit der Funktion "Auto Update" können so einfach bestimmte Sachverhalte in einer BACnet-Konfiguration visualisiert werden.

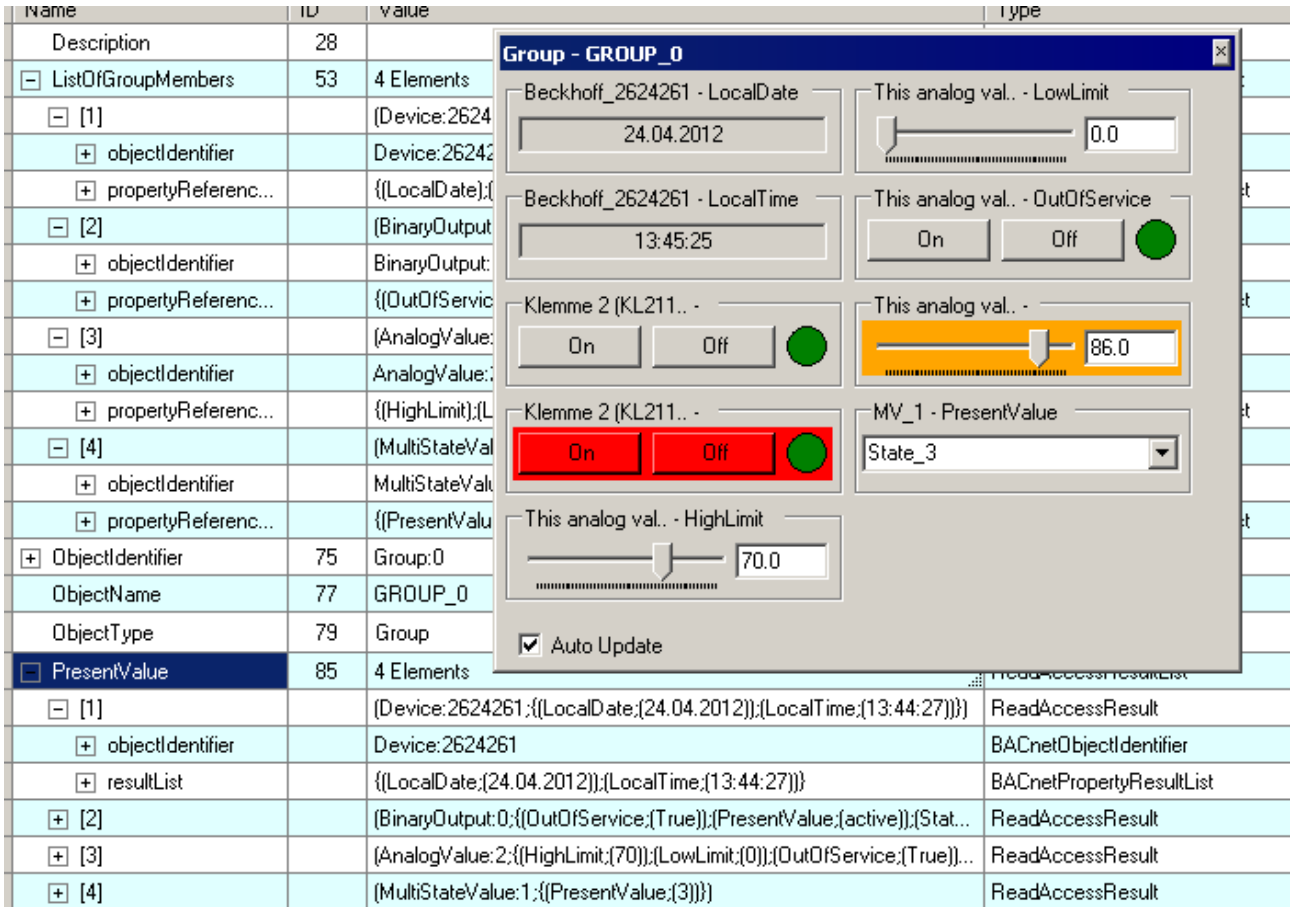


Group-Objekt Wizards

Group-Objekte ermöglichen die Zusammenfassung mehrerer Property-Werte multipler BACnet-Objekte. Über den im Abschnitt "Objekte und Properties" vorgestellten Property-Wizard kann die Property ListOfGroupMembers der Group-Objekte konfiguriert werden.



Zur Laufzeit enthält die Property PresentValue eines Group-Objekts die Werte der konfigurierten Properties. Der Group-PresentValue-Wizard stellt die enthaltenen Property-Werte in einem Fenster dar. Zusätzlich ist es möglich die Property-Werte zu verändern. Für Properties vom Datentyp Boolean und BinaryPV werden die Schaltflächen "On" bzw. "Off" sowie der aktuelle Zustand angezeigt. Properties von Typ Real werden als Schieberegler und Wert-Eingabe-Feld dargestellt. Die Zustände von Multistate*-Objekten sowie Enumerated-Properties (außer BinaryPV) werden als Combobox visualisiert. Ist zusätzlich zu einem PresentValue die Property StatusFlags eines Objekts konfiguriert worden, wird der Zustand *Offnormal* Orange und der Zustand *Fault* Rot hinterlegt.

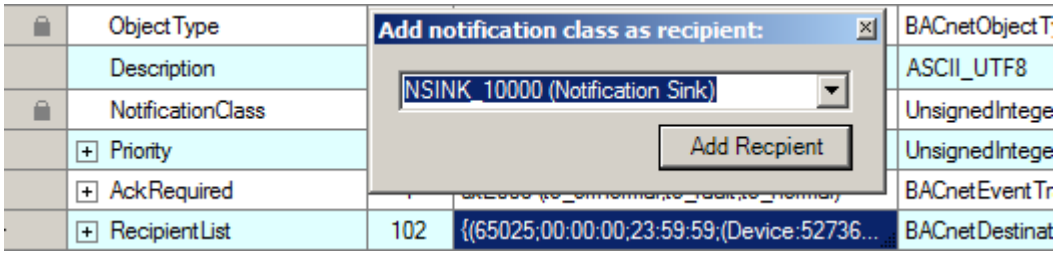


Name	ID	Value	Type
Description	28		
ListOfGroupMembers	53	4 Elements	
[1]		(Device:2624	
objectIdentifier		Device:26242	
propertyReferenc...		{(LocalDate);(	
[2]		(BinaryOutput	
objectIdentifier		BinaryOutput:	
propertyReferenc...		{(OutOfServic	
[3]		(AnalogValue:	
objectIdentifier		AnalogValue:1	
propertyReferenc...		{(HighLimit);(L	
[4]		(MultiStateVal	
objectIdentifier		MultiStateValu	
propertyReferenc...		{(PresentValu	
ObjectIdentifier	75	Group:0	
ObjectName	77	GROUP_0	
ObjectType	79	Group	
PresentValue	85	4 Elements	
[1]		(Device:2624261;{(LocalDate;(24.04.2012));(LocalTime;(13:44:27))})	ReadAccessResult
objectIdentifier		Device:2624261	BACnetObjectIdentifier
resultList		{(LocalDate;(24.04.2012));(LocalTime;(13:44:27))}	BACnetPropertyResultList
[2]		(BinaryOutput:0;{(OutOfService;(True));(PresentValue;(active));(Stat...	ReadAccessResult
[3]		(AnalogValue:2;{(HighLimit;(70));(LowLimit;(0));(OutOfService;(True))...	ReadAccessResult
[4]		(MultiStateValue:1;{(PresentValue;(3))})	ReadAccessResult

Durch Mausklick auf den Property-Namen einer GroupBox im Group-PresentValue-Wizard bei gedrückter SHIFT-Taste wird dynamisch ein Trendlog-Objekt auf dem zugehörigen BACnet-Client-/Server erstellt und gestartet. Hierdurch können sehr komfortabel bestimmte Properties bei einer Inbetriebnahme überwacht werden. Beim Schließen des erstellten Diagramm-Fensters wird auch die dynamisch erzeugten TrendLog-Objekte wieder gelöscht.

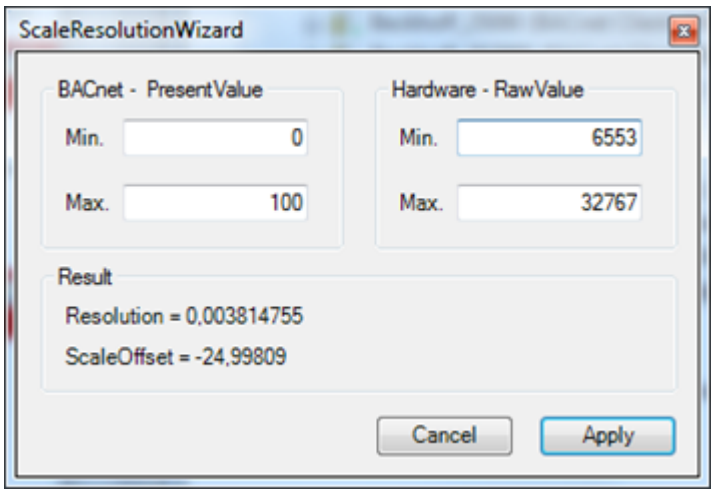
Notification Recipient Wizard

Mit Hilfe des Notification Recipient Wizard können Einträge in der RecipientList eines NotificationClass-Objektes leicht vorgenommen werden. So können lokale NotificationSinks auch auf entfernten BACnet-Stationen eingetragen werden. Im Wizard kann die einzutragende NotificationSink ausgewählt werden.



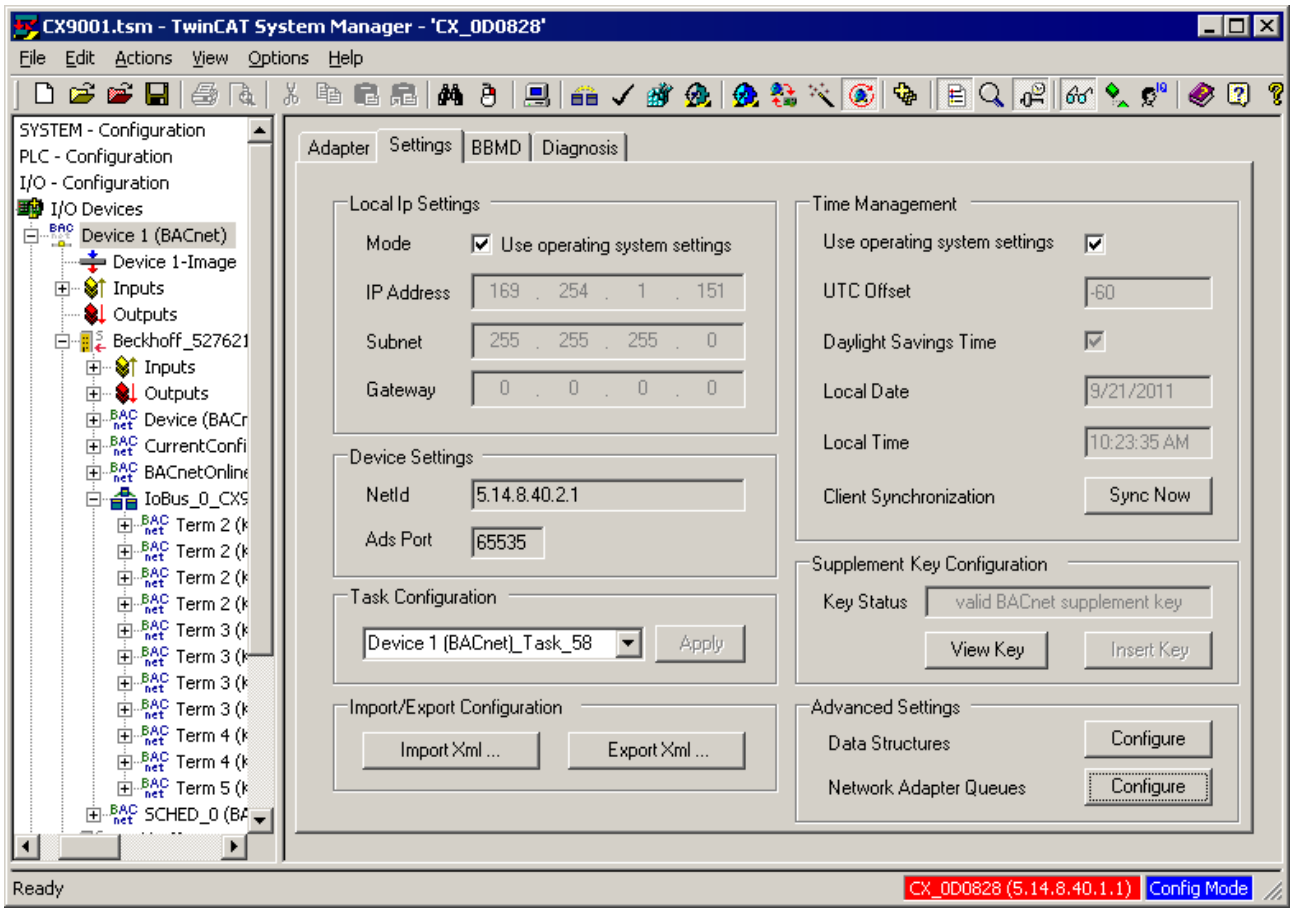
Scale-/Resolution Wizard

Mit Hilfe des ScaleResolution-Wizards können die Properties ScaleOffset und Resolution von AnalogInput- und AnalogOutput-Objekten berechnet werden. Hierbei wird der Werte-Bereich von BACnet und der verwendeten Hardware (also der Wertebereich der verknüpften Raw*-Variable) eingegeben. Beim Drücken von "Apply" werden die Werte automatisch übernommen. Im Online- oder im Settings-teil je nachdem wo der Wizard geöffnet wurde.



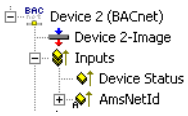
2.5 Prozessdaten

In diesem Abschnitt werden die Prozessdatenvariablen der einzelnen TwinCAT BACnet/IP Elemente beschrieben. Generell werden Prozessdaten eines BACnet/IP-Device über ein asynchrones Mapping mit der SPS bzw. einem Feldbus verknüpft. Dies ist notwendig, da BACnet azyklischer Natur ist und bestimmte Dienste viele Änderungen anderer BACnet-Properties und damit das Versenden vieler BACnet-Ethernet-Frames auslösen können. Damit Verknüpfungen mit BACnet-Prozessdaten nicht zu Zykluszeitüberschreitungen in der SPS bzw. zu Watchdog-Timeouts der Feldbusse führen, wird ein asynchrones Mapping verwendet; BACnet-, SPS- und Feldbus-Tasks laufen damit unabhängig voneinander.



Die Verknüpfung zu einer Task erfolgt unter dem BACnet-Device im Reiter "Settings" und "Task Configuration". Dort kann eine Task gewählt werden. Im Auswahldialog erscheinen alle Tasks mit einem Port größer 350. Links unten im Screenshot ist zu erkennen, dass eine SPS-Task ("BACnet_Demo") asynchron mit der BACnet-Task verknüpft ist, die per Default mit Priorität 58 beim Hinzufügen eines BACnet-Objekts angelegt wird.

BACnet Device

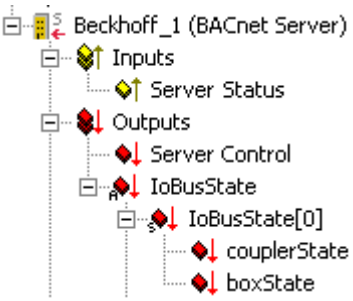


Unter einem BACnet-Device finden sich als Prozessdatenvariablen der Device Status, welcher den Link-Status des verknüpfen Ethernet-Adapters beinhaltet sowie den Zustand einer internen Statemachine. Der Device-Status wird in der anschließenden Tabelle erläutert.

Zusätzlich kann die AmsNetId des BACnet-Device mit einer SPS verknüpft werden, die via ADS auf BACnet-Funktionen zugreift.

Bit	Beschreibung
0	**Ethernet Link Status:** Repräsentiert den aktuellen Linkstatus des verknüpften Ethernet-Adapters: • 0 - Link ist vorhanden. • 1 - Kein Link ist vorhanden.
1	**Gateway Status:** Repräsentiert den aktuellen Status des IP-Gateways: • 0 - Gateway nicht vorhanden bzw. MAC-Addresse nicht aufgelöst • 1 - Gateway vorhanden und MAC-Addresse aufgelöst
8 - 15	**Device Statemachine**: Spiegelt den Zustand der internen BACnet-Device-Statemachine wieder: • Init (0) - Initialer Zustand • CheckIpAddress (1) - Initialisieren der IP-Adresse (u.a. Warten auf DHCP - Adresse) • CheckParameter (2) - Verifikation von Gateway und BBMD-Parametern • GetGatewayMAC (3) - Wenn ein Gateway konfiguriert wurde: Aussenden eines ARP-Request • WaitForGatewayMAC (4) - Warten auf Auflösen der Gateway-MAC-Adresse (ARP-Reply) • Complete (8) - Initialisierung beendet

BACnet Server



Der ServerStatus enthält unter anderem den aktuellen Zustand der internen Server-Statemachine. Der Serverzustand aus BACnet-Sicht kann mit Hilfe der Property SystemStatus ermittelt werden, die zyklisch verknüpft werden kann.

Über die IoBusState-Prozessdatenvariablen kann der Zustand einer I/O-Anbindung zentral überwacht werden. Bei der Verknüpfung eines K-Bus oder z.B. eines BK9000 gibt es einen CouplerState bzw. BoxState die den Zustand mehrerer I/O-Module abbilden. Über ein I/O-Automapping nach BACnet verknüpfte Feldbusse, können mit Hilfe dieser Prozessdaten die Reliability automatisch abbilden. Jedem Feldbusstrang wird eine IoBusNr zugeordnet, die auch eine herstellerspezifische BACnet-Property ist. Ist der Wert Prozessdatenvariable couplerState oder boxState ungleich 0, werden alle relevanten (I/O-) BACnet-Objekte des Servers mit der entsprechenden IoBusNr informiert und die Reliability entsprechend auf NO_SERVER bzw. NO_OUTPUT gesetzt. Per Default stehen 8 Io-Busse zur Verfügung.

BACnet Server Status

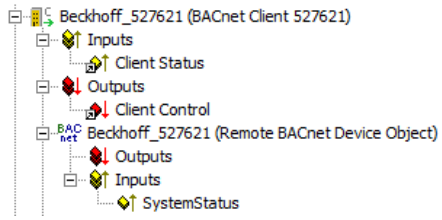
Bit	Beschreibung
2	**WriteOnChange disabled**: WriteOnChange ist über das Server-Control-Bit deaktiviert.
8 - 15	**BACnet Server Statemachine**: Spiegelt den Zustand der internen BACnet-Server-Statemachine wieder: • Init (0) - Initialer Zustand • Send I-Am Request (1) - Es wird ein initialer I-Am Request in das BACnet-Netzwerk gesendet. • Wait IO Timeout (2) - Warten auf die Freigabe der SPS-Prozessdaten und Objekt-Statemachines • Set System Status (3) - Property SystemStatus vom Device-Objekt wird auf Operational gesetzt

Bit	Beschreibung
	• Complete (4) - Der BACnet Server ist einsatzbereit und funktionsfähig. • Backup (5) - Es wird ein BACnet-Backup ausgeführt.

BACnet Server Control

Bit	Beschreibung
0	**Backup PLC feedback Enable:** Sollen bei einem BACnet-Backup von der SPS generierte Daten gesichert werden, kann über dieses Bit sichergestellt werden, dass die Backup-relevanten Dateien (Configuration Files) zunächst von der SPS geschrieben werden können. Ist dieses Bit gesetzt, wartet der BACnet-Stack mit dem Bestätigen des Backup-Request bis auch Bit 1 des Server-Control-Wortes gesetzt ist. Ist dieses Bit nicht gesetzt, wird vom BACnet-Stack dieses Hand-Shake Verfahren nicht ausgeführt. Weitere Informationen zum Thema Backup und Restore finden sich im entsprechenden Kapitel dieser Dokumentation.
1	**Backup PLC feedback Complete:** Wurden die Backup-relevante Daten der SPS gesichert, wird über dieses Bit der BACnet-Stack angewiesen mit dem Backup fortzufahren.
2	**Disable WriteOnChange**: Deaktiviert das Schreiben zyklischer Prozessdaten. Kann von der SPS verwendet werden um die schreibenden Prozessdaten zurückzusetzen ohne BACnet zu manipulieren
3	**Elapse IO startup timeout**: Beim Systemstart werden Prozessdaten im BACnet-Stack erst nach Ablauf der IO startup time verarbeitet. Diese ist als "Advanced Parameter" konfiguriert und ist als Default auf 2000 ms vorkonfiguriert. Dies ermöglicht der SPS schreibende Prozessdaten für BACnet vorzubereiten bevor potentiell Fehlermeldungen in das BACnet-Netz gesendet werden. Zum beschleunigten Start kann diese Zeit von der SPS verkürzt werden, indem dieses Bit auf den Wert 1 gesetzt wird. Die TcBACnet.lib verwendet diese Funktionalität automatisch.

BACnet Client



Zusätzlich zur Überwachung des Client-Status sollte immer auch der SystemStatus des Device-Objekts des Client als Prozessdatum aktiviert werden. Da Fehler bei der Client-Kommunikation im Client-Status nur angezeigt werden, wenn Fehler in der Kommunikation auftreten, sollte regelmäßig eine Kommunikation sichergestellt werden. Hierzu empfiehlt sich die Property SystemStatus zyklisch (mit einer entsprechend niedrigen Zykluszeit) auszulesen bzw. zu pollen.

Das Prozessdatum SystemStatus ist eine Enumeration mit dem Namen BACnetDeviceStatus und im BACnet-Standard definiert. Folgende Werte sind nach BACnet vorgesehen:

- operational (0)
- operational-read-only (1)
- download-required (2)
- download-in-progress (3)
- non-operational (4)
- backup-in-progress (5)
- ...

BACnet Client Status

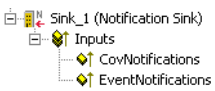
Bit	Beschreibung
0	**Client TimeOut:** Bei einem BACnet-Service-Request ist ein Timeout abgelaufen.
1	**Client Service Failure:** Ein Dienst mit dem projektierten Client konnte nicht erfolgreich durchgeführt werden. Dieses Fehlerbit wird gesetzt wenn: • Ein Client nicht erreichbar ist. • Eine COV-Subscription nicht erfolgreich durchgeführt werden konnte.

Bit	Beschreibung
	• Ein WriteProperty-/ReadProperty-Request ein negatives Ergebnis brachte.
2	**Client WriteOnChange State**: Aktivierungssteuerung für WriteOnChange, die im Client Control vorgegeben wird. • 0 - WriteOnChange ist aktiv. Bei Änderung der Client-Prozessdaten werden entsprechende BACnet-Netzwerkdienste ausgelöst. • 1 - WriteOnChange ist deaktiviert. Prozessdaten können geändert werden ohne das BACnet-Netzwerkdienste ausgelöst werden.
3	**Client COV Subscription Control State**: Zeigt den Zustand der COV-Subscription-Steuerung an. Ist das **Client COV Subscription Control** Bit gesetzt wird dieses Bit auf 1 gesetzt wenn die COV-Subscription-Datenstrukturen zurückgesetzt wurden. Ist das **Client COV Subscription Control** Bit nicht gesetzt ist auch State 0. Dieses Bit gibt keine Auskunft darüber, ob alle COV-Subscriptions erfolgreich durchgeführt wurden, sondern lediglich, dass alle COV-Subscriptions in Zukunft neu ausgeführt werden.
4	**Client Force Output Update State**: Force Output Update ist aktiv.
8 - 15	**Client Remote State**: Spiegelt den Zustand der internen Client-Statemachine wieder: • Init (0) - Intialer Zustand • Connecting (1) - Aussenden eines Who-Is-Request für die Client-ID • WaitingForConnection (2) - Warten auf I-Am und IP-Adresse des Client. Es wird 10 Sekunden auf eine Antwort gewarten, dann wird die WhoIs-Anfrage wiederholt (Connecting). • RequestDeviceInformation (3) - Auslesen der Properties ApduTimeout und NumberOfApduRetries und wenn Segmentierung unterstützt wird zusätzlich MaxSegmentsAccepted, ApduSegmentTimeOut • Complete (4) - Client vollständig initialisiert. Es kann mit dem BACnet-Client interagiert werden.

BACnet Client Control

Bit	Beschreibung
0	**Client TimeOut Reset:** Zurücksetzen des Client-TimeOut-Bit im Client Status.
1	**Client Service Failure Reset:** Zurücksetzen des Service-Failure-Bit im Client Status.
2	**Client WriteOnChange Control:** Aktivierungssteuerung für WriteOnChange. WriteOnChange ist deaktiviert, wenn im Client Status das entsprechende Bit gesetzt ist! • 0 - WriteOnChange ist aktiv. Bei Änderung der Client-Prozessdaten werden entsprechende BACnet-Netzwerkdienste ausgelöst. • 1 - WriteOnChange ist deaktiviert. Prozessdaten können geändert werden ohne das BACnet-Netzwerkdienste ausgelöst werden.
3	**Client COV Subscription Control:** Setzt den Status der COV-Subscriptions zurück bzw. löst die Neuanmeldung von COV-Subcriptions aus. Die internen Datenstrukturen werden genau einmal zurückgesetzt (und damit die COV-Subscriptions neu ausgelöst), wenn der die Wertänderung 0 -> 1 erkannt wird.
4	**Client Force Output Update State:** Bei einem Wechsel von 0 nach 1 werden zum nächsten möglichen Zeitpunkt alle schreibenden Prozessdaten auf einem BACnet Client aktualisiert. Dies kann verwendet werden, um sicherzustellen, dass Werte für die gesammte Querkommunikation auf einer Gegenstelle übernommen werden und nicht zwischenzeitlich von einem anderen Gerät überschrieben wurden.

Notification Sink



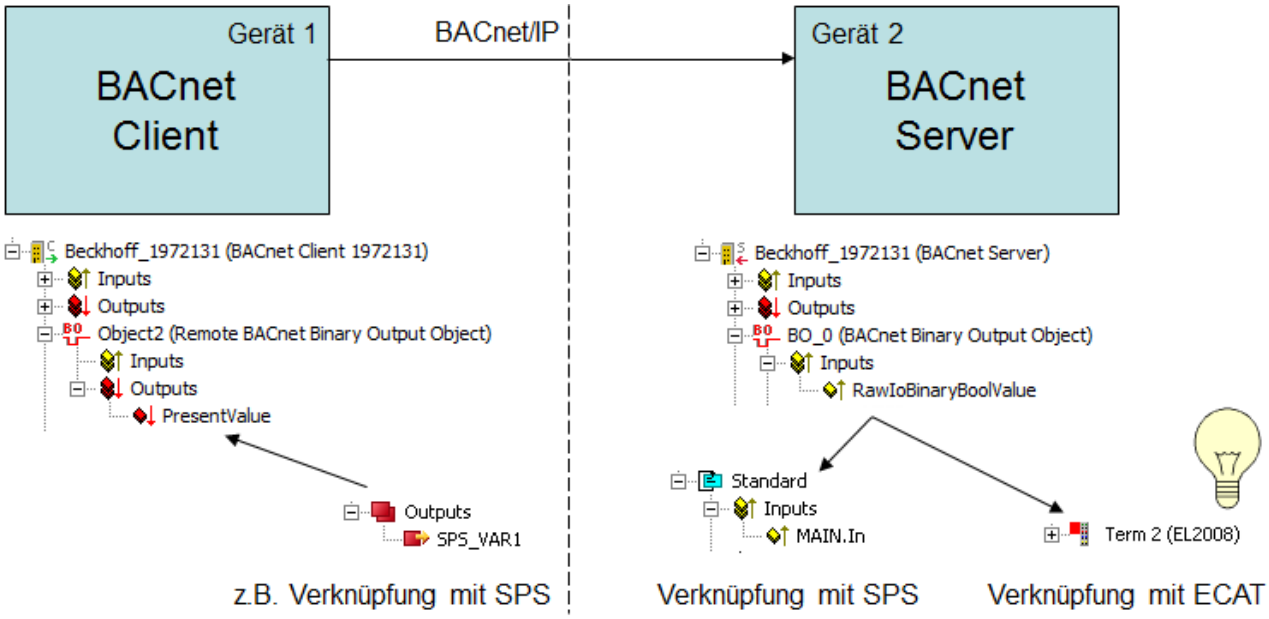
Die TwinCAT BACnet/IP Notification Sink ermöglicht den Empfang von COV- und Event-Notifications. Über die Prozessdaten CovNotifications und EventNotifications kann jeweils die Anzahl der empfangenen Notifications ermittelt werden. Über die ADS-Schnittstelle der Notification Sink können Notifications gelöscht und gelesen werden.

BACnet-Objekte - Properties als Prozessdaten

Properties mit nicht-komplexen Datentypen (Signed Integer, Unsigned Integer, BitField, Boolean, Real, Enumerated) können als zyklische Prozessdaten konfiguriert werden. Bei den Property-Prozessdaten muss zwischen Objekten unter einem Server (lokale Objekte) und Objekten unter einem Client (entfernte Objekte) unterschieden werden.

- Die Properties von lokalen Objekten werden unterhalb eines BACnet-Server konfiguriert. Über den Dialog "*Cyclic Data*" kann konfiguriert werden, welche Properties als Prozessdaten verknüpftbar sind. Verknüpfbare Properties erscheinen unter einem Objekt als Prozessdatenvariablen und können mit Variablen einer SPS oder auch mit anderen I/O-Geräten (z.B. EtherCAT-Klemmen oder K-Bus-Klemmen) verknüpft werden. Properties können als lesende bzw. schreibende Prozessdaten aktiviert werden:
  - Ausgangsdaten („*SPS/IO -> BACnet*“) sind Daten die aus einer SPS heraus in ein BACnet-Objekt kopiert werden. Diese sind als rote Variable dargestellt.
  - Eingangsdaten ("*BACnet -> SPS/IO*"); gelb dargestellt; werden aus BACnet heraus zur SPS kopiert. Dies sind z.B. die StatusFlags eines Objekts, die in der SPS verarbeitet werden können.

Auch bei entfernten (engl. "*remote*") Objekten können die Properties eines entfernten Gerätes als zyklische Prozessdaten bereitgestellt werden. Die Daten der Properties können dabei entweder via COV (Change of Value) azyklisch übertragen werden oder zyklisch in einem festen Intervall. Auch bei entfernten Objekten sind Daten, die von einer SPS konsumiert werden, gelb dargestellt; produzierte Daten rot.



Die Abbildung zeigt eine schematische Darstellung der Verwendung von Prozessdaten bei Client- und Server-Objekten. Im Beispiel ist auf einem Server ein BinaryOutput-Objekt angelegt. Hierbei ist die Property PresentValue als Prozessdatenvariable verknüpft. Das als *RawInBinaryBoolValue* umgesetzte PresentValue kann auf dem Server in einer SPS verwendet werden, um z.B. eine Funktionalität in einem SPS-Programm zu beeinflussen oder kann mit einer EtherCAT/KBus-Klemme verknüpft werden, um z.B. eine Beleuchtung zu aktivieren. *RawInBinaryBoolValue* ist in diesem Fall auf Server-Seite eine gelbe SPS-Eingangsvariable. In einem 2. Gerät ist das Objekt als entferntes Objekt konfiguriert. In diesem Beispiel soll der Wert des PresentValue des Serverobjekts mit Hilfe einer SPS auf dem Client verändert werden. Hierfür ist das PresentValue auf dem Client als rote Ausgangsvariable verknüpft und kann so mit der SPS verknüpft werden.

Die folgende Tabelle zeigt die **Abbildung von BACnet-Datentypen** zu SPS-Datentypen:

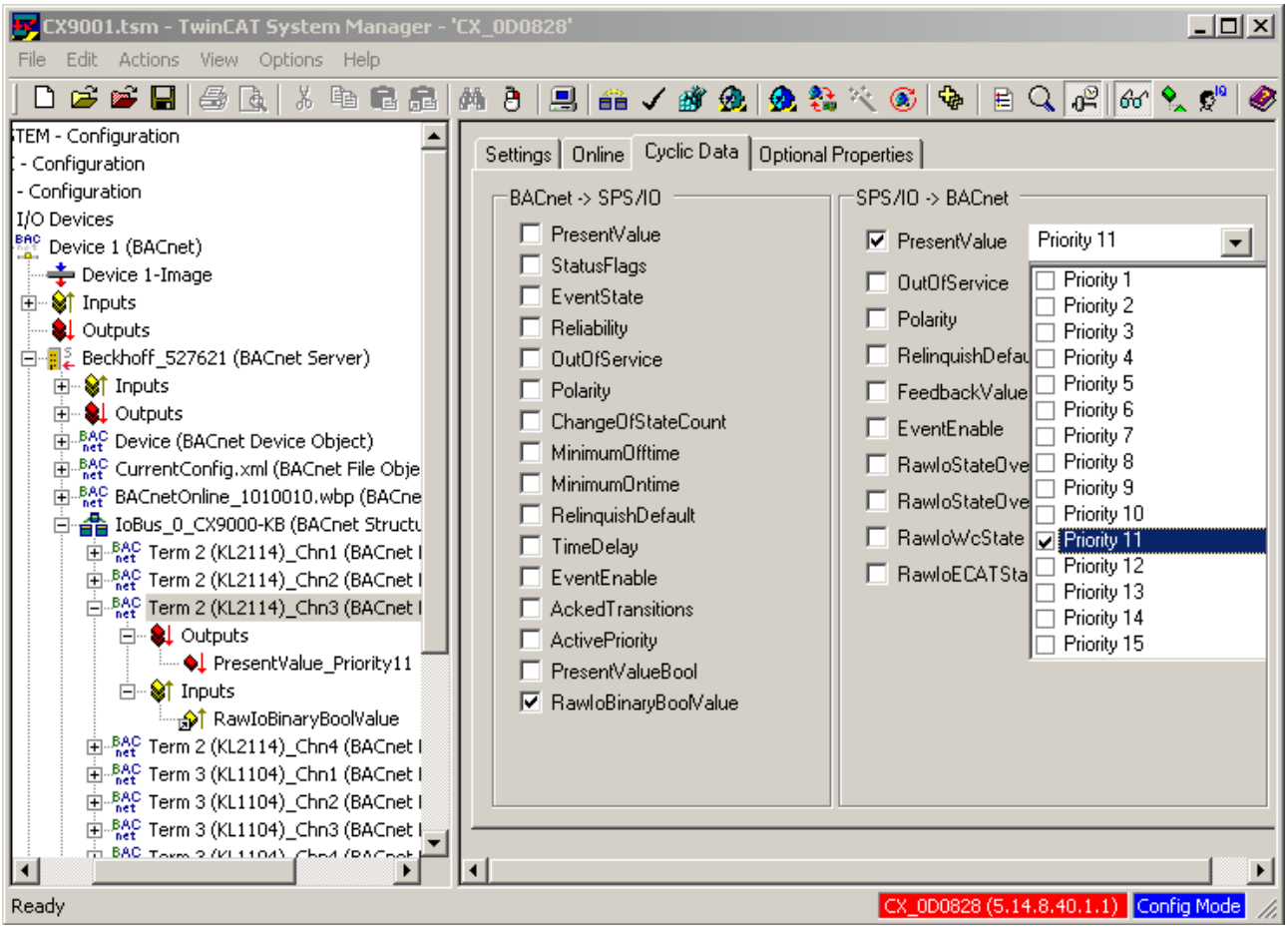
BACnet Datentyp	SPS Datentyp
Boolean	BOOL
Unsigned Integer	UDINT
Signed Integer	DINT
Real	REAL
Double	wird nicht verwendet

BACnet Datentyp	SPS Datentyp
Bit String (StatusFlags, EventEnable, LimitEnable usw.)	WORD (Bei Bit-Strings ist zu beachten, dass das niederwertigste Byte dazu verwendet wird, die Anzahl nicht verwendeter Bits des höherwertigsten Bytes anzuzeigen. Ein Beispiel ist 0xE005 einer EventEnable-Property: Die 5 gibt an, dass im höherwertigsten Byte 3 Bits verwendet werden [to_offnormal, to_fault, to_normal]. 0xE0 signalisiert, dass alle 3 Bits TRUE sind. Wird eine Bit-String-Property zyklisch von der SPS nach BACnet geschrieben, müssen diese Bits entsprechend codiert werden. Gültige Werte können im Reiter "Online" der BACnet-Objekte entnommen werden.)
Enumerated (EventState, Reliability, Polarity, BinaryPV usw.)	WORD

Prozessdaten von lokalen Objekten

Bei lokalen Objekten werden (schreibende) Prozessdaten generell einmalig beim Systemstart und danach bei Änderung geschrieben. Dies bedeutet, dass potenziell der von der SPS geschriebene Wert und der Wert der beschriebenen Property voneinander abweichen können. U.a. weil z.B. via BACnet oder ADS ein weiterer Akteur den Wert geändert hat. Deshalb empfiehlt es sich, für schreibende Prozessdaten eine Rückkopplung in die SPS umzusetzen. Dieses Verhalten wurde gegenüber der Revision 6 Implementierung geändert, bei der alle Properties in jedem Zyklus geschrieben wurden. Eine Ausnahme bilden die Raw*-Properties, die in jedem Zyklus geschrieben werden.

Kommandierbare Properties



BACnet bietet die Möglichkeit bestimmte (kommandierbare) Properties über einen Prioritätsmechanismus zu steuern. Der Wert der Property *PresentValue* der Objekte BO, BO, AO,AV, MV und MO wird dabei durch die Einträge der Property *PriorityArray* gesteuert. Die verschiedenen Einträge entsprechen unterschiedlicher Wichtigkeit und können von verschiedenen Akteuren im BACnet-Netzwerk beschrieben werden. Mit

TwinCAT BACnet/IP können Einträge im *PriorityArray* von der SPS beschrieben werden. Hierzu kann im Reiter "Cyclic Data" eine Prioritätsstufe für das Schreiben der Property *PresentValue* gewählt werden (Die Werte werden hierdurch intern auf die Property *PriorityArray* kopiert.)

Um niedrigere Prioritätsstufen zum Zuge kommen zu lassen, ist es möglich eine Prioritätsstufe mit dem Wert NULL zu schreiben (freizugeben).

- Von der SPS ist dies für BinaryOuput/-Value-Ojekte möglich. Wird in das PresentValue einer verknüpften Prioritätsstufe eines BinaryValue- oder BinaryOutput-Objekt ein Wert ungleich 0/1 geschrieben, wird der Wert NULL in das PriorityArray kopiert.
- Bei AnalogOutput/-Value-Objekten wird der Wert NULL im PriorityArray angenommen, wenn als Prozessdatum die spezielle Kodierung von REAL-Werten NaN (not a number) verwendet wird. Bei Server-Objekten wird außerdem NULL geschrieben, wenn der REAL-Wert der Prozessdaten-Variable außerhalb von Min-/MaxPresValue liegt.
- Bei MultistateOuput/-Value-Objekten wird der Wert NULL im PriorityArray angenommen, wenn als Prozessdatum eine 0 geschrieben wird. Bei Server-Objekten wird auch NULL angenommen wenn das Prozessdatum einen Wert größer als NumberOfStates annimmt.

Pro BACnet-Objekt können mehrere Prioritätsstufen gleichzeitig von der SPS beschreiben werden.

RawIoProperties

Prozessdaten-Properties dieser Kategorie werden für die Verknüpfung von BACnet-Objekten mit I/O-Modulen und SPS-Variablen verwendet. Diese Properties stellen direkt keine BACnet-sichtbaren BACnet-Properties dar, wirken aber auf standardisierte Properties der entsprechenden Objekte bzw. werden aus ihnen gebildet. Generell gilt für RawIo-Properties, dass die BACnet-Property OutOfService wirkt. Ein Objekt wird im Folgenden OutOfService genannt, denn die BACnet-Property OutOfService hat den Wert TRUE.

Property	Beschreibung
RawIoSignedValue	Vorzeichenbehafteter 16-Bit Wert (INT) der bei Analog*-Objekten Anwendung findet. Wenn ein AnalogInput-Objekt nicht OutOfService ist, bildet der Wert dieser Prozessdatenvariable den Wert des PresentValue. Dabei wird der Wert von RawIoSignedValue in einen REAL-Datentyp konvertiert, das Resultat mit der Property Resolution multipliziert und abschließend mit der Property ScaleOffset addiert. Für AnalogOutput-Objekte wird aus dem PresentValue der Wert von RawIoSignedValue nach dem inversen Verfahren gebildet. Überschreitet dabei der Wert des PresentValue den Wertebereich des Datentypes INT (-32768 bis 32767) hat RawIoSignedValue den Wert 0. Ist ein Objekt OutOfService bleibt je nach Konfiguration der Properties FaultFallbackValue bzw. OutOfServiceFallbackValue der letzte Wert bestehen bzw. der konfigurierte Wert wird verwendet.
RawIoUnsingedValue	Vorzeichenloser 16-Bit-Wert (UINT) der bei Analog*-Objekten Anwendung findet. Wenn ein AnalogInput-Objekt nicht OutOfService ist, bildet der Wert dieser Prozessdatenvariable den Wert des PresentValue. Dabei wird der Wert von RawIoSignedValue in einen REAL-Datentyp konvertiert, das Resultat mit der Property Resolution multipliziert und abschließen mit der Property ScaleOffset addiert. Für AnalogOutput-Objekte wird aus dem PresentValue der Wert von RawIoSignedValue nach dem inversen Verfahren gebildet. Überschreitet dabei der Wert des PresentValue den Wertebereich des Datentypes UINT (0 bis 65535) hat RawIoSignedValue den Wert 0. Ist ein Objekt OutOfService bleibt je nach Konfiguration der Properties FaultFallbackValue bzw. OutOfServiceFallbackValue der letzte Wert bestehen bzw. der konfigurierte Wert wird verwendet.
RawIoBinaryEnumValue	Aufzählungstyp (in der SPS als WORD verarbeitet) der bei Binary*-Objekten Anwendung findet. Wenn OutOfService nicht TRUE ist gilt. Bei **BinaryOutput**-Objekten bildet sich der Wert der Property RawIoBinaryEnumValue: • RawIoBinaryEnumValue ist 0 wenn PresentValueACTIVE und die Polarität NORMAL oder wenn PresentValueINACTIVE und die Polarität REVERSE ist.

Property	Beschreibung
	• RawIoBinaryEnumValue ist 1 wenn PresentValueINACTIVE und die Polarity NORMAL oder wenn PresentValueACTIVE und Polarity REVERSE ist. Ist OutOfServiceTRUE nimmt die Property RawIoBinaryEnumValue bei BinaryOutput-Objekten den Wert der Property OutOfServiceFallbackValue an wenn dieser aktiviert ist, ansonsten bleibt der letzte gültige Wert erhalten. Bei **BinaryInput**-Objekten bildet sich das PresentValue aus der Property RawIoBinaryEnumValue. • PresentValue ist ACTIVE wenn RawIoBinaryEnumValue 0 und die Polarität NORMAL oder wenn RawIoBinaryEnumValue größer 0 und die Polarität REVERSE ist. • PresentValue ist INACTIVE wenn RawIoBinaryEnumValue größer 0 und die Polarity NORMAL oder wenn RawIoBinaryEnumValue 0 und Polarity REVERSE ist. ist OutOfServiceTRUE nimmt PresentValue immer den Wert von FaultFallbackValue an wenn dieser aktiv ist andernfalls nimmt PresentValue den letzten Wert an als OutOfServiceFALSE war.
RawIoBinaryBoolValue	Binärer Wert (BOOL) der bei Binary*-Objekten Anwendung findet. Die Property PresentValue der Binary*-Objekte ist in BACnet ein Auszählungstyp. Für den optimierten Umgang mit binären Objekten wurde die Property RawIoBinaryBoolValue eingeführt, die an Stelle von RawIoBinaryEnumValue verwendet werden kann. Bei **BinaryOutput**-Objekten bildet sich der Wert der Property RawIoBinaryBoolValue: • RawIoBinaryBoolValue ist FALSE wenn PresentValueACTIVE und die Polarität NORMAL oder wenn PresentValueINACTIVE und die Polarität REVERSE ist. • RawIoBinaryBoolValue ist TRUE wenn PresentValueINACTIVE und die Polarity NORMAL oder wenn PresentValueACTIVE und Polarity REVERSE ist. Ist OutOfServiceTRUE nimmt die Property RawIoBinaryEnumValue bei BinaryOutput-Objekten den Wert der Property OutOfServiceFallbackValue an wenn dieser aktiviert ist, ansonsten bleibt der letzte gültige Wert erhalten. Bei **BinaryInput**-Objekten bildet sich das PresentValue aus der Property RawIoBinaryBoolValue. • PresentValue ist ACTIVE wenn RawIoBinaryBoolValueFALSE und die Polarität NORMAL oder wenn RawIoBinaryBoolValueTRUE und die Polarität REVERSE ist. • PresentValue ist INACTIVE wenn RawIoBinaryBoolValueTRUE und die Polarity NORMAL oder wenn RawIoBinaryBoolValueFALSE und Polarity REVERSE ist. ist OutOfServiceTRUE nimmt PresentValue immer den Wert von FaultFallbackValue an wenn dieser aktiv ist andernfalls nimmt PresentValue den letzten Wert an als OutOfServiceFALSE war.
RawIoStatus	Analoge Eingangsmodule können ein Prozessdatum besitzen, welches den Zustand des physikalischen Anschlusses anzeigt (z.B. Drahtbrucherkennung, Bereichsüber-/unterschreitungen. Ist die Property RawIoStatus (Datentyp USINT) aktiv, beeinflusst diese die Property Reliability. Die Property Reliability bildet sich aus RawIoStatus wenn das Objekt nicht OutOfService, kein IoBus-Fehler vorliegt (siehe RawIoECATState, IoBusState) und kein Status-Override aktiv ist, (siehe RawIoStateOverride*) wie folgt: • Ist Bit 0 gesetzt, ist Reliability UNDER_RANGE • Sonst ist wenn Bit 1 gesetzt Reliability OVER_RANGE • Sonst ist wenn Bit 6 gesetzt Reliability UNRELIABLE_OTHER

Property	Beschreibung
RawIoECATState	EtherCAT-Module besitzen einen Operationsstatus, der den aktuellen Betriebszustand abbildet. Mit EtherCAT-Modulen verknüpfte BACnet-Objekte können ihre Reliability über diesen Operationsstatus bilden. Mit Hilfe von RawIoECATState können so abgezogene Module (Module-Plug) sowie Teilbusausfälle erkannt und nach BACnet abgebildet werden. Die Property Reliability bildet sich aus RawIoECATState wenn das Objekt nicht OutOfService ist wie folgt: • Bei Ausgangsobjekten (BO, AO, MO) gilt: Hat der Wert von Bit 0 - Bit 3 ungleich 8 (OP), ist Reliability NO_OUTPUT. • Bei Eingangsobjekten (BI, AI, MI) gilt: Hat der Wert von Bit 0 - Bit 3 ungleich 8 (OP), ist Reliability NO_SENSOR.
RawIoWcState	Die WcState-Überwachung wird momentan noch nicht unterstützt.
RawIoStateOverride	Einige I/O-Module besitzen einen mechanischen Handbetrieb mit dem Feldbusprozessdaten lokal überschrieben (engl. override) können. Eine Handübersteuerung kann in BACnet über die Property StatusFlags mit dem Bit OVERRIDE abgebildet werden. Ist die Prozessdatenvariable RawIoStateOverrideTRUE ist beim zugeordneten BACnet-Objekt das Bit OVERRIDE in der Property StatusFlags gesetzt andernfalls nicht, genau wenn das BACnet-Objekt nicht OutOfService ist.
RawIoStateOverrideInverted	Einige I/O-Module besitzen einen mechanischen Handbetrieb mit dem Feldbusprozessdaten lokal überschrieben (engl. override) können. Eine Handübersteuerung kann in BACnet über die Property StatusFlags mit dem Bit OVERRIDE abgebildet werden. Ist die Prozessdatenvariable RawIoStateOverrideFALSE ist beim zugeordneten BACnet-Objekt das Bit OVERRIDE in der Property StatusFlags gesetzt andernfalls nicht, genau wenn das BACnet-Objekt nicht OutOfService ist. Beispiel für die Verwendung von RawIoStatusOverrideInverted sind die Module KM2642 und KM4602.
RawIoAccumulatorUnsignedValue	RawIoAccumulatorUnsignedValue ermöglich die Erfassung von Zählwerten über eine 16-Bit Prozessdatenvariable. Diese kann z.B. mit einer KL5101 verknüpft werden. bzw. können Zählimpulse von der SPS vorgegeben werden. Der BACnet-Stack führt hierbei eine Rotationserkennung bei 32767 durch. D.h. heißt beim Übergang von 32767 nach 1 werden 2 Pulse gezählt (nach 0 und nach 1). Um einem Datenverlust beim Stoppen der SPS vorzubeugen, werden bei einem aktuellen Wert von 0 keine Operationen durchgeführt. Erst bei einem neuen Wert von 1 würde entsprechend weiter akkumuliert. Über die Property AccumulatorIntegrationMode kann ein integrierender Modus aktiviert werden. Hat diese Property den Wert TRUE, werden nicht Änderungen der Variable RawIoAccumulatorUnsignedValue akkumuliert sondern das Produkt aus RawIoAccumulatorUnsignedValue * vergangener Zeit (in ms). Bei einem Wert von 1 wird also nach einer Sekunde 1000 akkumuliert. Diese Funktion ermöglicht z.B. die Erfassung von Luftmengen auf Basis einer aktuellen Luftmenge.
RawIoPulseConverterUnsignedValue	RawIoPulseConverterUnsignedValue hat die gleiche Funktionsweise wie RawIoAccumulatorUnsignedValue. Der Integrationsmodus wird bei PulseConverter-Objekt nicht unterstützt.

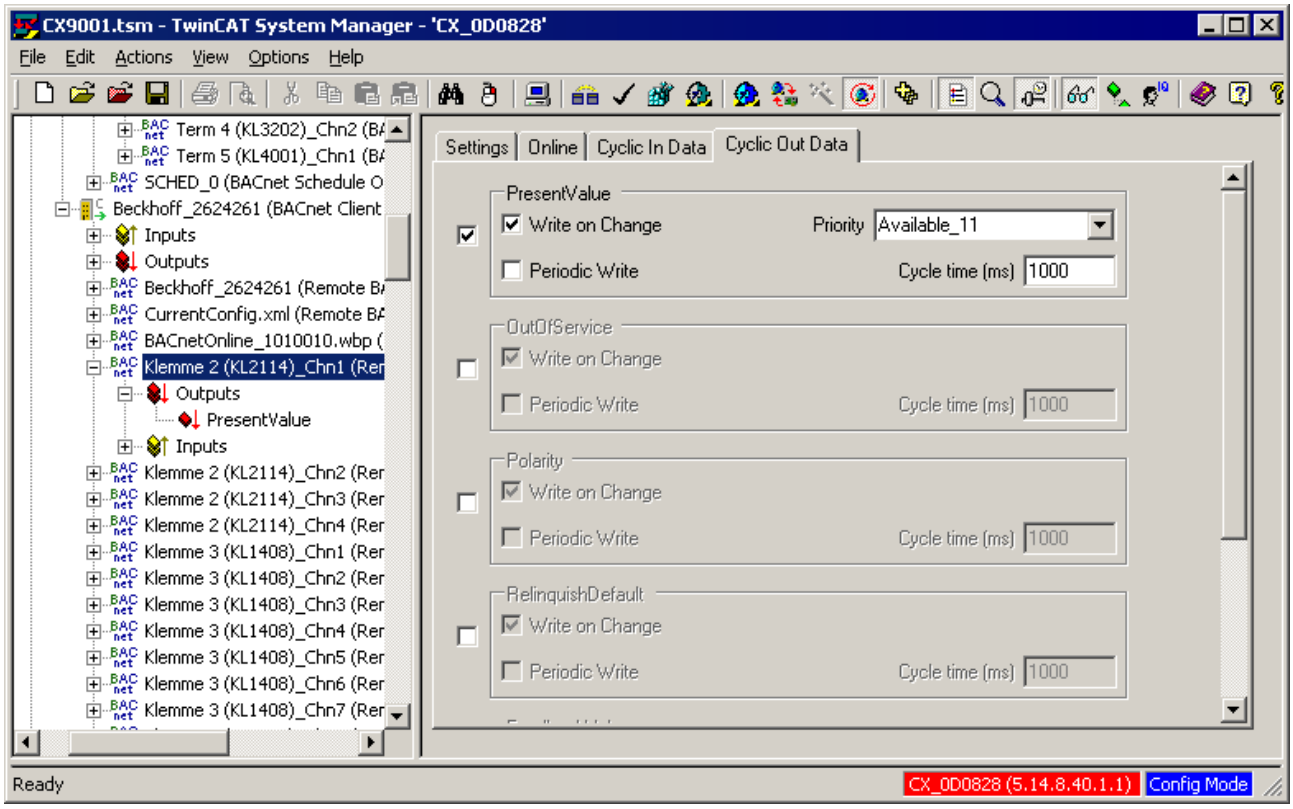
Spezielle Datenvarianten des PresentValue

Property	Beschreibung
PresentValueBool	**Schedule Objekt**: Ist das PresentValue eines Schedule-Objekts vom BACnet-Datentyp Boolean, spiegelt die Prozessdaten-Property PresentValueBool den aktuellen Wert der Property PresentValue wieder. **Binary*-Objekte**: Die Property PresentValue von Binary*-Objekten ist in BACnet vom Datentyp BinaryPV und damit eine Enumeration. Deshalb ist das zugehörige PresentValue als Prozessdatum vom Datentyp WORD, da Enumerationen in der SPS als 16-Bit-Wert verarbeitet werden. Als optimierte

Property	Beschreibung
	Variante kann das Prozessdatum PresentValueBool vom Datentyp BOOL verwendet werden. PresentValueBool ist TRUE wenn PresentValueACTIVE ist und FALSE wenn PresentValueINACTIVE ist.
PresentValueEnumerated	Ist das PresentValue eines Schedule-Objekts eine BACnet-Enumeration, spiegelt die Prozessdaten-Property PresentValueEnumerated den aktuellen Wert der Property PresentValue wieder.
PresentValueReal	Ist das PresentValue eines Schedule-Objekts vom BACnet-Datentyp Real, spiegelt die Prozessdaten-Property PresentValueBool den aktuellen Wert der Property PresentValue wieder.
PresentValueSigned	Ist das PresentValue eines Schedule-Objekts vom BACnet-Datentyp SignedInteger, spiegelt die Prozessdaten-Property PresentValueBool den aktuellen Wert der Property PresentValue wieder.
PresentValueUnsigned	Ist das PresentValue eines Schedule-Objekts vom BACnet-Datentyp UnsignedInteger, spiegelt die Prozessdaten-Property PresentValueBool den aktuellen Wert der Property PresentValue wieder.

Remote Prozessdaten

Bei Remote-Objekten müssen Daten zum entfernten BACnet-Server übertragen werden (mit Hilfe des BACnet-WriteProperty-Dienstes). Für die Übertragung der Daten stehen in TwinCAT BACnet/IP zwei Varianten zu Auswahl. Daten können „*On Change*“ also bei Veränderung geschrieben werden. Hierbei wird beim Verarbeiten der Prozessdaten innerhalb von TwinCAT BACnet/IP beim Erkennen einer Änderung des Prozessdatum ein Schreibzugriff auf die entsprechende BACnet-Property ausgelöst. Dieses Verfahren entspricht der Verarbeitung lokaler Objekte. Als 2. Variante können die Daten zyklisch geschrieben werden. Achtung: Bei "*Write On Change*" werden die Daten genau einmal bei einer Änderung geschrieben. Wird im Nachhinein der Wert der Property von einem anderen BACnet-Gerät überschrieben, kann der Wert der Client-SPS vom Property-Wert abweichen! Hier empfiehlt sich ggf. die Rückkopplung der BACnet-Properties als Eingangsvariable. Auch bei "Write On Change" werden Daten maximal mit der eingestellten Zykluszeit übertragen. Über den Dialog *"Cyclic Out Data*" können schreibende Prozessdaten für entfernte Objekte konfiguriert werden:



Bei kommandierbaren Properties kann für den Schreibzugriff eine Priorität gewählt werden. Diese Priorität wird dann im WriteProperty-Request übertragen.

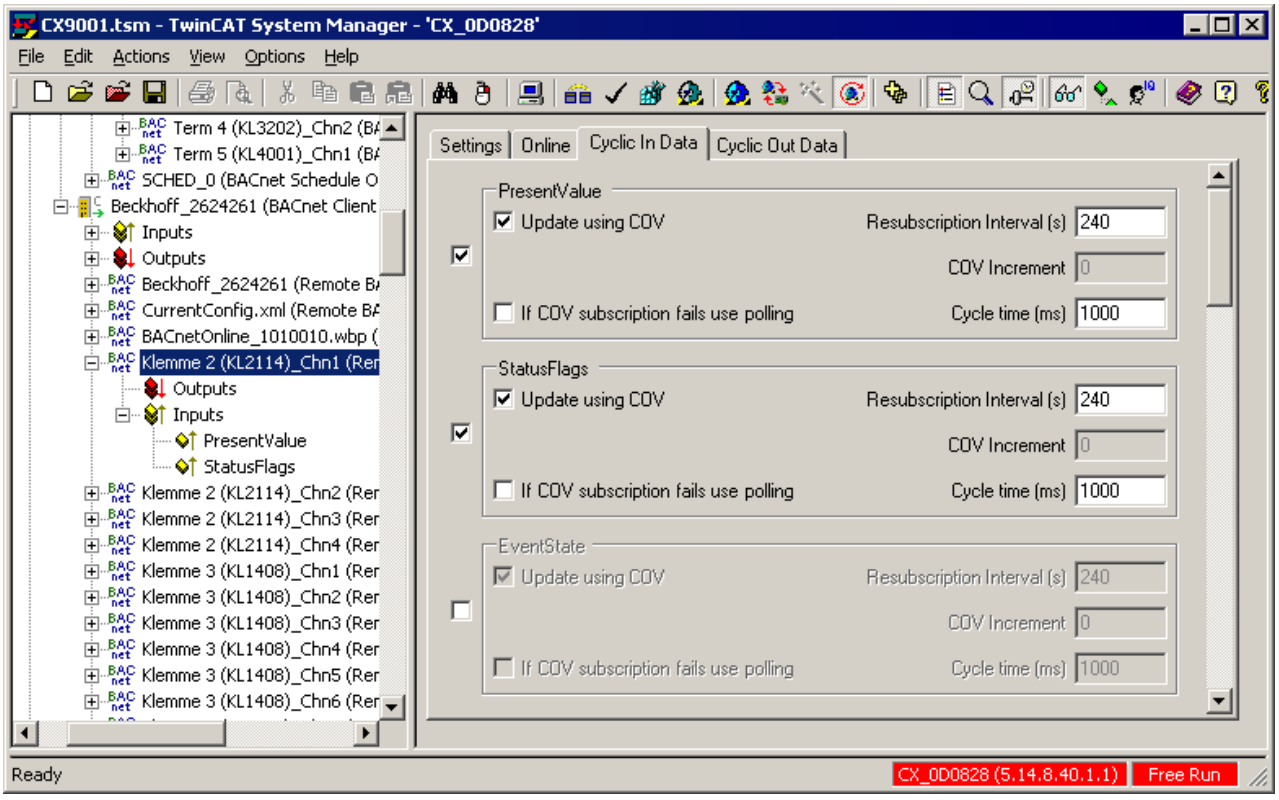
Eine Besonderheit bei schreibenden Remote-Prozessdaten ist eine automatische Verriegelung beim Stopp der SPS. Fällt im BACnet-Server-Steuerwort(Control) Bit3 (ElapseIoActiveTime) von 1 nach 0 wird automatisch für jeden BACnet-Client die *DisableWriteOnChange*-Funktion aktiviert. Erst bei einem erneuten Setzen dieses Bits auf 1 können wieder entfernte Prozessdaten geschrieben werden. Diese Verriegelung ist erforderlich um bei einem Stopp der SPS ein Fluten des BACnet-Netzes mit "0" zu verhindern.

Lesende Prozessdaten von entfernten BACnet-Objekten werden über den Reiter "*Cyclic In Data*" konfiguriert. Als Basiskonfiguration kann in diesem Dialog ausgewählt werden, ob Daten bei Änderung (COV) oder zyklisch übertragen werden sollen. Zusätzlich kann festgelegt werden, in welchem Abstand neue Abonnements für Änderungsmeldungen (*Resubscription Intervall*) durchgeführt werden sollen. Über das COV Increment kann bei Properties mit Gleitkommawerten vorgegeben werden, in welcher "Auflösung" Daten übertragen werden sollen. Die *CycleTime* gibt vor mit welcher Periodendauer zyklische Daten gelesen werden.

BACnet unterstützt zwei Dienste für die Übertragung von Änderungsmeldungen. Der Dienst COV-P erlaubt die Übertragung von Änderungen sämtlicher Properties. Zusätzlich kann bei Properties mit Gleitkommawerten pro *Subscription* ein COV-Increment vorgegeben werden. Der Dienst COV erlaubt nur die Übertragung von speziell festgelegten Properties bei ausgewählten Objekten (z.B. *PresentValue*, *StatusFlags*). COV wird von vielen Herstellern unterstützt; COV-P wird nur von einigen. Um maximale Kompatibilität zu erreichen wird, wenn immer möglich, versucht den Dienst COV zu verwenden.

COV-P wird verwendet wenn:

- eine Property nicht von COV unterstützt wird (z.B. *EventState*)
- ein COV Increment ungleich 0.0 eingegeben wurde



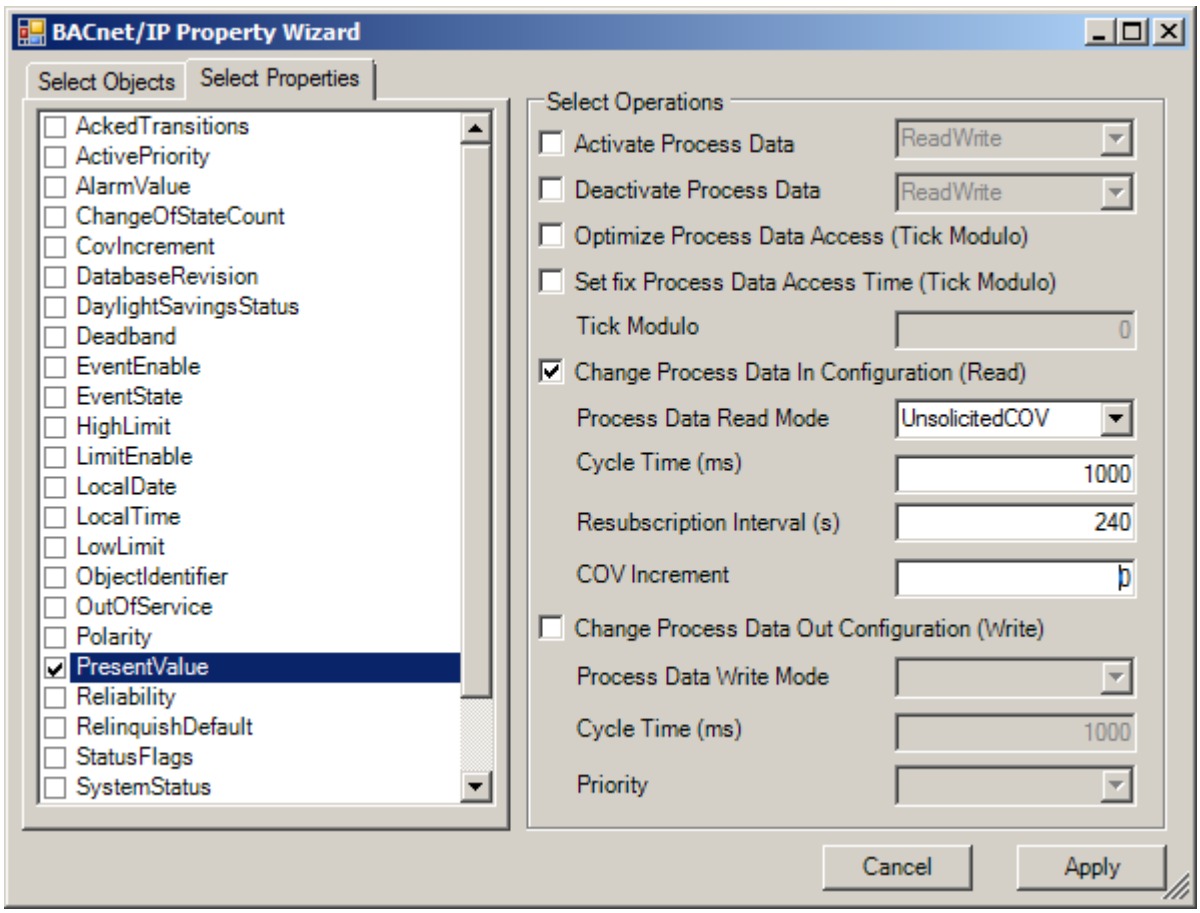
Der TwinCAT BACnet/IP Stack der Revision 12 unterstützt umfangreichere weitere Modi für die Übertragung von Prozessdaten entfernter BACnet-Objekte. Diese Übertragungsmodi können nicht im Reiter "*Cyclic In Data*" konfiguriert werden, da immer mehrere BACnet-Properties gleichzeitig betroffen sind. Einige der in der folgenden Tabelle dargestellten Modi können nur über das SPS-Automapping oder mit dem Prozessdaten-Wizard aktiviert werden.

Voraussetzungen

Übertragungsart	Beschreibung
Unconfirmed COV	Dieser Modus wird im Dialog "*Cyclic In Data*" aktiviert. Property-Werte werden als unbestätigte Änderungsmeldungen verschickt (*UnconfirmedCOV*). Dass heißt der Absender stellt nicht sicher, dass die Daten tatsächlich ausgeliefert werden. Um diesen Modus verwenden zu können, muss die Gegenstelle den Dienst COV bzw. COV-P für die aktivierte Property unterstützen. Dieser Modus kann in zuverlässigen Netzwerken verwendet werden.
Confirmed COV	Bei diesem Modus werden Änderungsmeldungen bestätigt ausgeliefert (*ConfirmedCOV*). Der Sender wartet nach dem Absenden einer Änderungsmeldung die konfigurierte *APDU-Timeout* (Device-Objekt) ab. Wurde die Änderungsmeldung nicht bestätigt, wird die Meldung nochmals gesendet bis die APDURetries (Device-Objekt) erreicht sind. Um diesen Modus verwenden zu können, muss die Gegenstelle den Dienst COV bzw. COV-P für die aktivierte Property unterstützen. Dieser Modus sollte in Netzwerken mit temporären Überlasten verwendet werden bzw. wenn sichergestellt sein muss, dass Änderungsmeldungen ausgeliefert wurde. Zu beachten ist, dass maximal 255 gleichzeitige Änderungen zu einer Gegenstelle übertragen werden können.
Unsolicited COV	Bei den Modi *Unconfirmed COV* und *Confirmed COV* werden Änderungen durch den Dienst *SubscribeCOV* abonniert. Die Gegenstelle speichert dieses Abonnement in einer Tabelle, die oft eine begrenzte Größe besitzt. Wenn sich in einem Netzwerk viele Geräte für Änderungen einer bestimmten Property registrieren (z.B. Wetterdaten), kann es sinnvoll sein, den Dienst *Unsolicited COV* zu verwenden. Hierbei kann auf eine Registrierung verzichtet werden. Änderungen werden vom Quellgerät als Broadcast (Nachricht an alle Teilnehmer im Netzwerk) automatisch versendet. Wenn dieser Modus für Client-Prozessdaten verwendet wird, wird beim TwinCAT-Start die entsprechende Property einmalig mittels *ReadProperty* gelesen (Initialwert), danach werden die zugehörigen Broadcast-Telegramme ausgewertet. Details zur Konfiguration von *Unsolicited COV* als Client und Server finden sich am Ende dieses Kapitels.
ReadProperty (Polling)	Dieser Modus wird im Dialog "*Cyclic In Data*" aktiviert. Property-Werte werden zyklisch mittels *ReadProperty* übertragen.
ReadPropertyMultiple pro Objekt (RPM-O)	Mit dem Dienst *ReadPropertyMultiple* können in BACnet mehrere Property-Werte mit einer Nachricht übertragen werden. Müssen von einem entfernten BACnet-Geräte viele Property-Daten als Prozessdaten übertragen werden, kann der Modus RPM-O verwendet werden. Hierbei werden zyklische Read-Zugriffe auf mehrere Properties in einer Nachricht pro Objekt zusammengefasst.
ReadPropertyMultiple pro Client (RPM-C)	Mit dem Dienst *ReadPropertyMultiple* können in BACnet mehrere Property-Werte mit einer Nachricht übertragen werden. Müssen von einem entfernten BACnet-Geräte viele Property-Daten als Prozessdaten übertragen werden, kann der Modus RPM-C verwendet werden. Im Gegensatz zu RPM-O werden bei RPM-C alle Property-Werte aller Objekte zu einer Nachricht zusammengefasst. Der BACnet-Stack optimiert die Anfragegröße automatisch, dass eine Segmentierung auf BACnet-Ebene vermieden wird (Nachrichten werden auf ca. < 1400 Byte pro Nachricht begrenzt. Diese Übertragungsart kann bei langsamen Netzwerken eingesetzt werden in denen viele Property-Daten übertragen werden müssen.

Unsolicited COV - Client Konfiguration

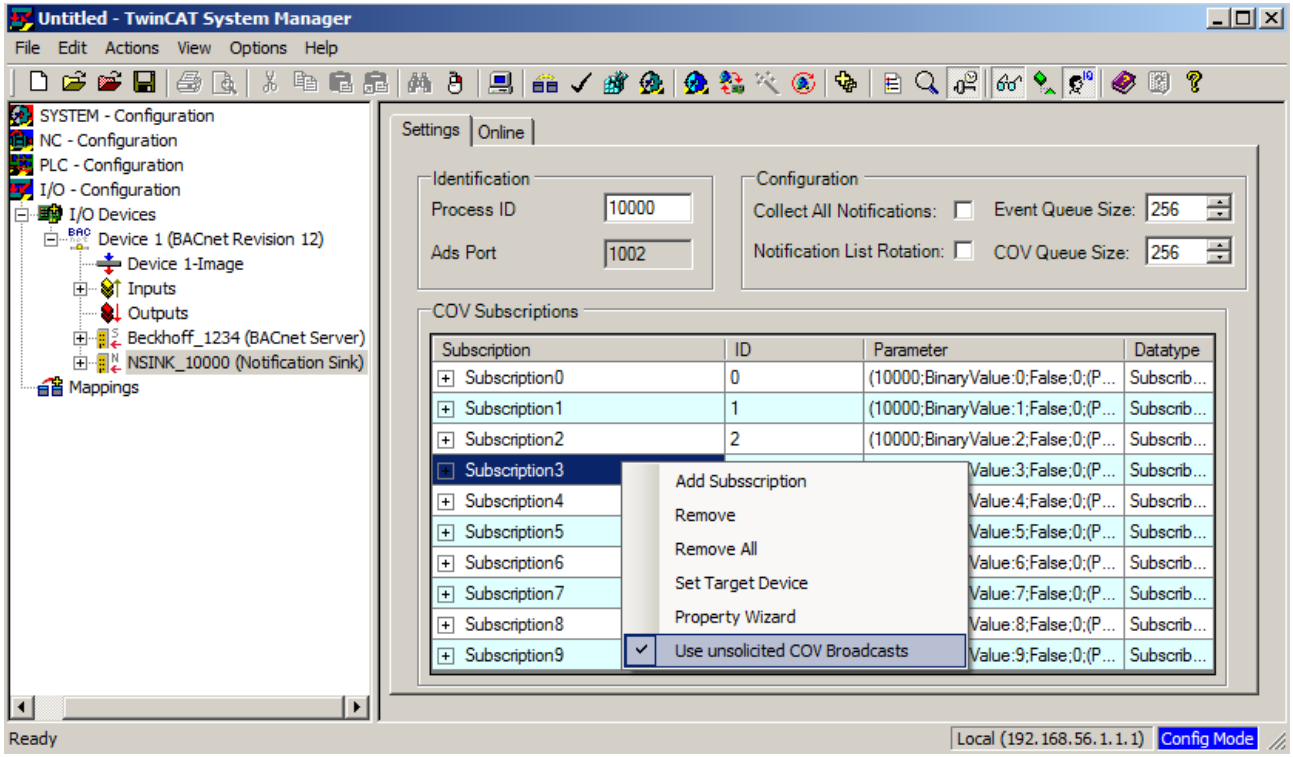
Bei der Funktionalität *Unsolicited COV* werden Änderungsmeldungen ohne vorherige Anmeldung (*COV-Subscription*) im BACnet-Netzwerk verschickt. Diese Funktion wird z.B. von Raumbediengeräten oder Wetterstationen verwendet. Um diese Änderungsmeldungen als Client-Prozessdaten zu konsumieren, kann der Modus Unsolicited COV verwendet werden. Über den Property-Wizard kann dieser Modus gewählt werden, indem alle gewünschten Objekte des BACnet-Client ausgewählt, Properties selektiert und abschließend der neue Modus im Feld "Change Process Data In Configuration (Read)" aktiviert wird. In diesem Modus wird einmalig beim Start ein Initialwert via *ReadProperty* gelesen und anschließend die Änderungsmeldungen verarbeitet.



Unsolicited COV - Server-Konfiguration

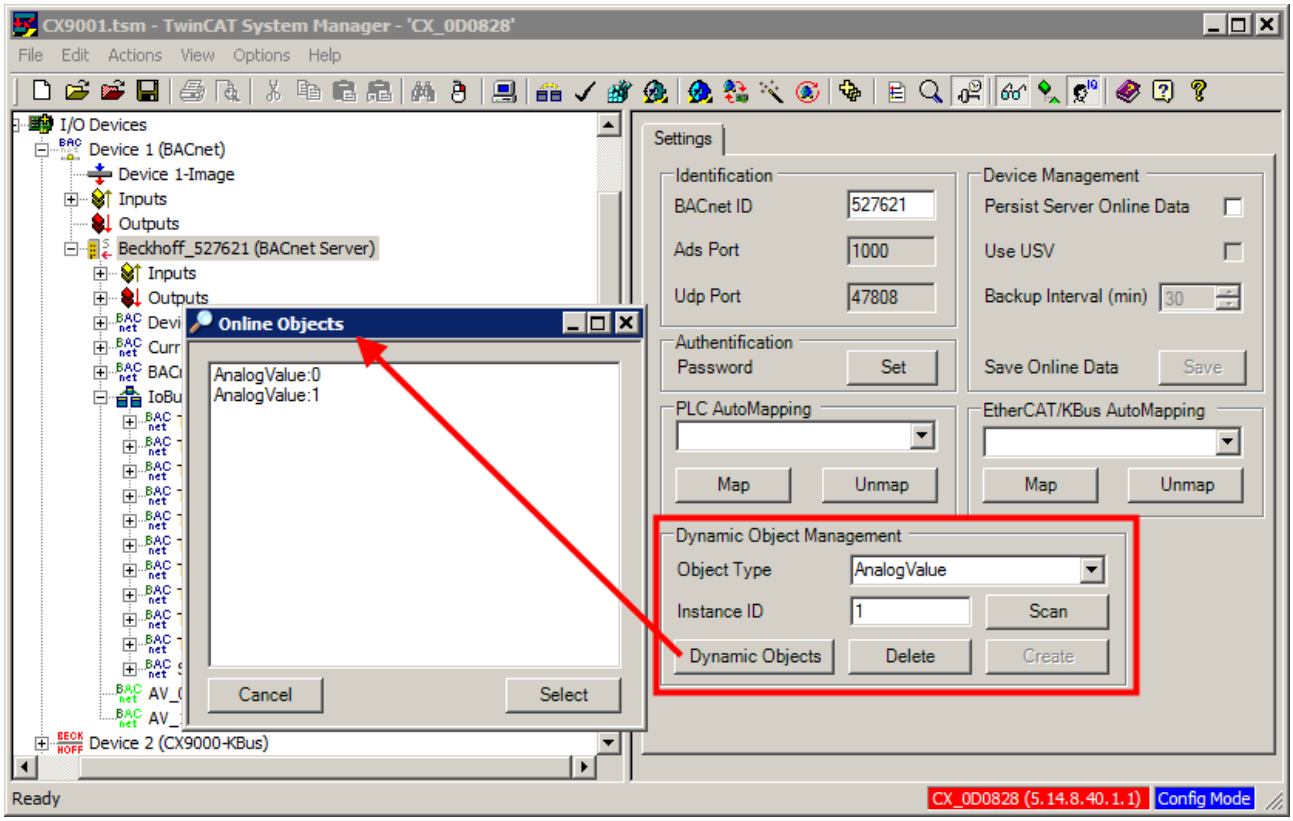
TwinCAT BACnet/IP unterstützt auch die *Unsolicited COV* Server Funktionalität. Das heißt, es können Änderungsmeldungen ohne vorherige Anmeldung als Broadcast im BACnet-Netz versendet werden. *Unsolicited COV* wird über eine (oder mehrere) Notification Sinks konfiguriert. Über das Kontextmenü im "Settings"-Reiter kann Unsolicited COV über die CheckBox "*Use unsolicited COV Broadcasts*" aktiviert werden. Dies bewirkt, das alle konfigurierten COV *Subscriptions* intern mit Broadcast-*Repicient* (Empfänger) gespeichert und verarbeitet werden. Über die COV-Subscription-Liste können auch Einstellung wie das COV-*Increment* festgelegt werden.

Über den Property-Wizard im Kontext-Menü können über eine Liste die Properties ausgewählt werden, bei denen bei Änderung Meldungen verschickt werden sollen.

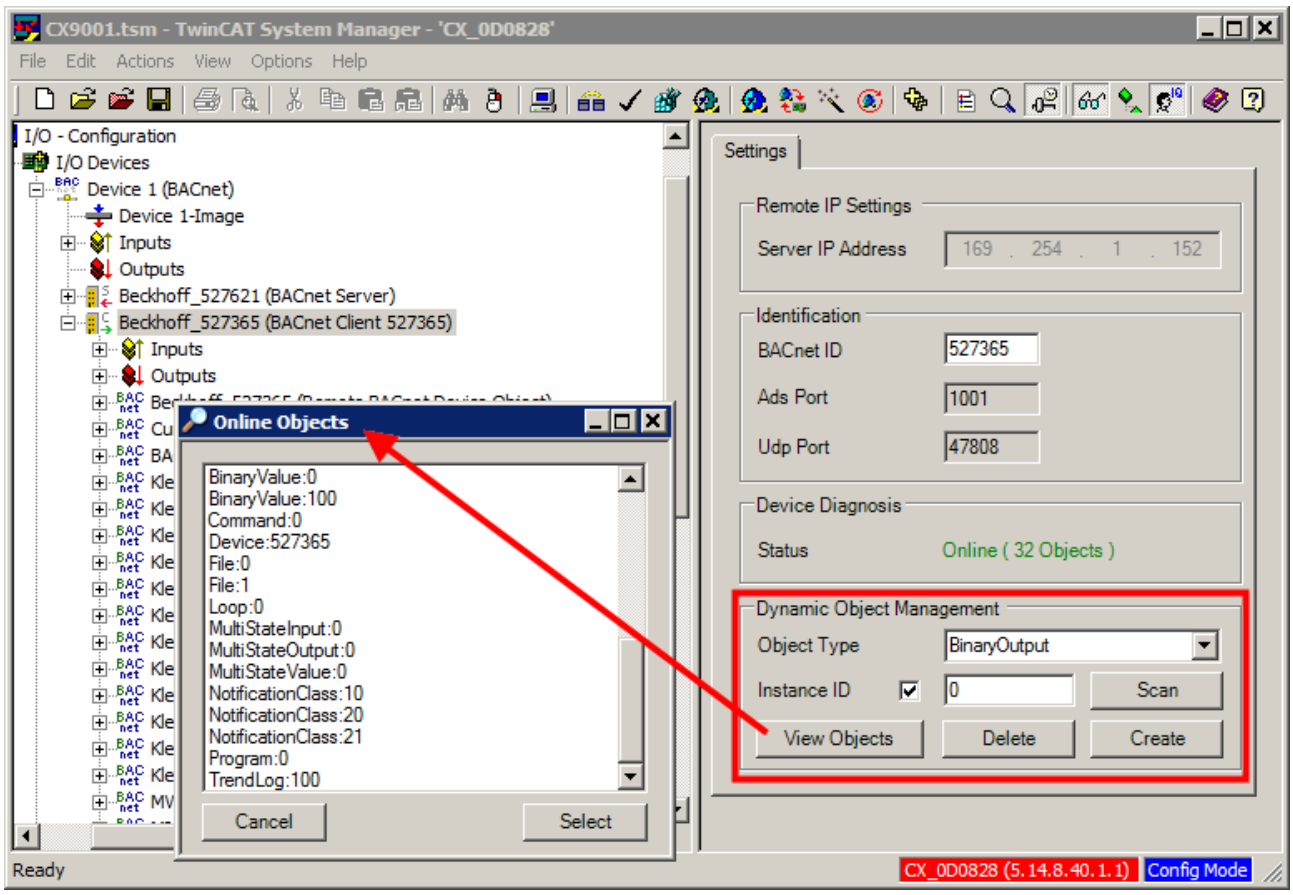


2.6 Verwaltung dynamischer Objekte

Dynamische Objekte sind BACnet-Objekte die während der Laufzeit (RUN oder Free RUN) auf einem Server (lokal) oder einem Client (remote) angelegt werden können, ohne TwinCAT neu zu aktivieren/starten zu müssen. Der Zugriff auf dynamische Objekte eines lokalen Servers erfolgt unter dem Reiter "Settings" des BACnet-Servers. Im Reiter "Settings" können die dynamischen Objekte angezeigt, erzeugt und gelöscht werden. Zudem besteht mit dem Button "Scan" die Möglichkeit bereits vorhandene, nicht angezeigte Objekte zu scannen und damit in der Baumdarstellung unterhalb des Servers einzubinden (z.B. dynamische Objekte, die von einem entfernten System erzeugt wurden). Um ein Objekt anzulegen muss der entsprechende Object Type (Objekttyp) und eine Instance ID (Objektnummer) eingegeben werden. Zum Löschen muss eine bereits vorhandene Objektnummer mit entsprechendem Objekttyp eingegeben oder mit der Schaltfläche „Dynamic Objects“ komfortabel selektiert werden.



Der Zugriff auf die dynamischen Objekte eines Clients (Remote-Zugriff) ist ähnlich. Die Verwaltungsfunktion befindet sich im Reiter "Settings" des entsprechenden BACnet-Clients. Der Unterschied zur Verwaltung eines Servers besteht in der Unterscheidung zwischen dynamischen und statischen (offline-konfigurierten) Objekten. Generell können alle dynamischen Objekte erzeugt/gelöscht werden, mit Ausnahme von: Loop-, Device- und Program-Objekten. Statische Objekte können zur Laufzeit weder erzeugt noch gelöscht werden. Da jedoch in der BACnet-Kommunikation nicht zwischen dynamischen und statischen Objekten unterschieden wird, gibt es diesbezüglich auch keinen Unterschied in der Anzeige von Objekten im Reiter "Settings" → "View Objects" des BACnet-Clients. Aufgrund dessen werden im Dialog "Online Objects" alle Objekte aufgelistet, die die Client-Seite zur Verfügung stellt. Der Versuch ein statisches Objekt zu löschen, wird mit einer Fehlerrückmeldung des BACnet-Clients (Error-Result) quittiert. Im Reiter "Settings" des BACnet-Clients gibt es die Möglichkeit mehrere dynamische Objekte gleichzeitig zu erzeugen. Dazu muss die Tastenkombination CTRL+SHIFT während der Betätigung des Buttons "Create" gedrückt bleiben.



Die Darstellung von dynamischen Objekten unter einem BACnet-Server erfolgt mittels "grünem" Symbol. Auf Client-Seite gibt es, wegen dem oben erwähnten Client-Verhalten, keine Unterscheidung in der Baumansicht.

Bei der dynamischen Erzeugung von Client-Objekten, kann konfiguriert werden, ob im CreateObject-Dienst eine Objekt-ID angegeben werden soll oder nicht. Hier ist zu beachten, dass die Geräte einiger Herstellung nur die dynamische Objekterzeugung ohne Angabe einer Objekt-ID unterstützen. In diesem Fall kann der Haken "Instance ID" abgewählt werden.

Da dynamische Objekte erst zur Laufzeit erzeugt werden, gibt es die Möglichkeit des Prozessdaten-Mappings [▶ 43] nicht. Die Verknüpfung von Prozessdaten muss bereits beim Laden der TwinCAT-Konfiguration bekannt sein. Dies wiederum schließt ebenfalls die Offline-Konfiguration von dynamischen Objekten aus. Die Konfiguration erfolgt deshalb ausschließlich unter dem Reiter "Online". Dabei ist zu beachten, dass die Konfiguration sämtlicher dynamischer Objekte und deren Properties nur bei aktivierter Daten-Persistenz nach einem Neustart erhalten bleiben (Informationen zu Daten-Persistenz: siehe Kapitel "Persistente Daten [▶ 60]").

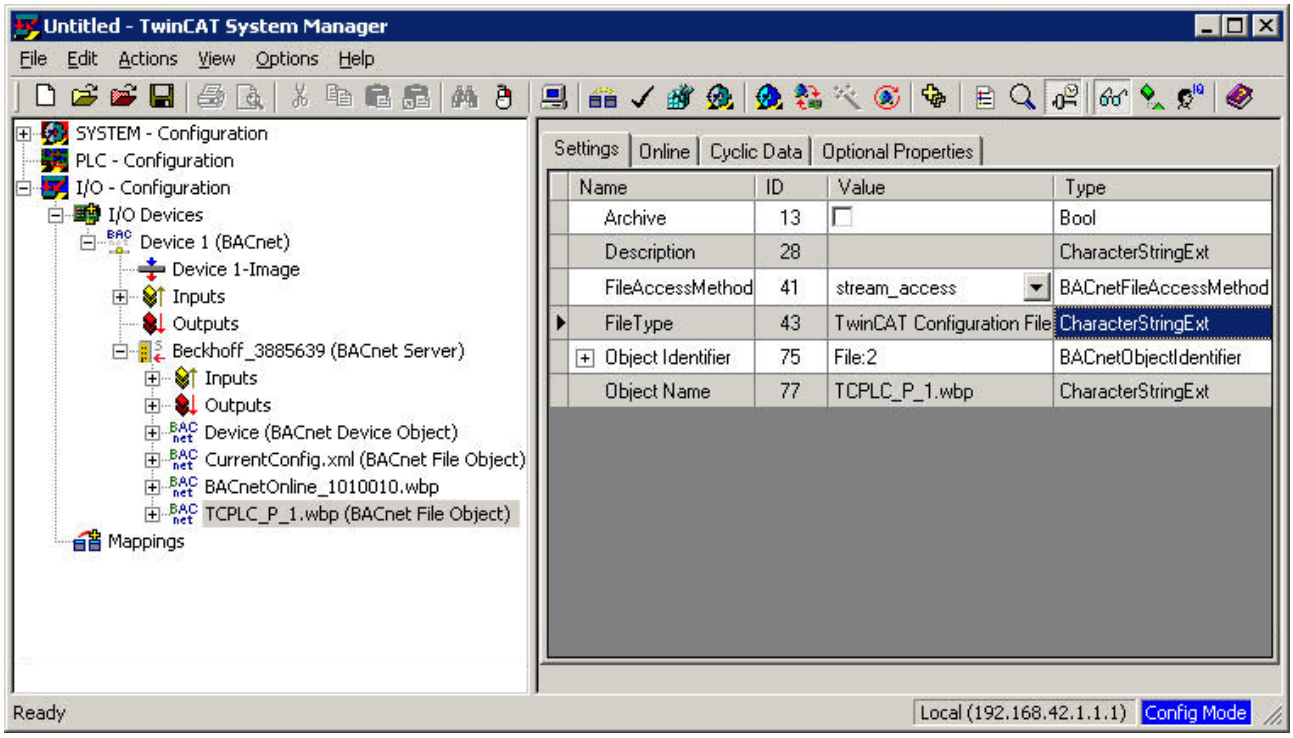
2.7 Backup und Restore

BACnet ermöglicht die Erstellung von Sicherungen (Backup) der Konfiguration eines BACnet-Controllers. In TwinCAT BACnet/IP kann die Konfiguration aus einer Reihe von Dateien bestehen. Die wesentlichen Dateien sind zum einen die Datei CurrentConfig.xml, welche die generelle TwinCAT Konfiguration enthält. Zusätzlich wird in TwinCAT BACnet/IP eine weitere BACnetOnline-Datei verwendet, welche die relevanten Daten enthält, die während der Laufzeit verändert wurden und von der Ausgangskonfiguration der CurrentConfig.xml abweichen. Potenziell können weitere Dateien eine Konfiguration bestimmen z.B. ein SPS Boot-Projekt oder projektspezifische Dateien.

Der BACnet-Backup und -Restore Mechanismus ermöglich eine flexible Anzahl von Dateien mit Hilfe der File-Dienste zu sichern und wiederherzustellen. Nachdem über die entsprechenden BACnet-Dienste ein Backup ausgewählt wurde, kann ein BACnet-Client über die Device-Objekt Property ConfigurationFiles auslesen welche BACnet-File-Objekte zu einer Backup-Konfiguration gehören. Die jeweiligen BACnet-File-

Objekte weisen auf die entsprechenden Dateiennamen. Zu beachten ist, dass in TwinCAT BACnet/IP alle Dateien relativ zum Boot-Projekt-Ordner (per Default "C:\TwinCAT\Boot") behandelt werden. Backup-relevante Dateien sollten deshalb in diesem Ordner gespeichert werden.

Beim Anlegen eines BACnet Server werden automatisch zwei File-Objekte angelegt, die auf die Datei CurrentConfig.xml sowie die BACnetOnline-Datei verweisen.



Zusätzlich bietet BACnet TwinCAT/IP die Möglichkeit weitere Dateien in eine Backup-Konfiguration aufzunehmen. Hierzu kann ein neues BACnet-File-Objekt hinzugefügt werden. In der Property ObjectName kann der Dateiname konfiguriert werden. Wird die Property FileType mit dem String "TwinCAT Configuration File" initialisiert, wird dieses File-Objekt bei einer Backup-Anforderungen in die Property ConfigurationFiles aufgenommen und kann somit von einem BACnet-Client gesichert und später wiederhergestellt werden. Im abgebildeten Screenshot ist dargestellt, wie das Boot-Projekt der SPS-Laufzeit 1 in die Backup-Konfiguration aufgenommen werden kann.

Des Weiteren besteht die Möglichkeit innerhalb der SPS Backup-relevante Daten zu speichern. Um sicherzustellen, dass die SPS Backup-Daten während einer Backup-Operation auf das Speichermedium geschrieben wurden, existiert ein Rückkopplungsmechanismus zwischen BACnet und einer SPS. Durch die zyklische Überwachung der Property SystemStatus des Server-Device-Objekts kann erkannt werden wann ein Backup eingeleitet wurde (BackupInProgress (5)). Über das Server Kontrollstatuswort (ServerControl) kann durch Setzen von Bit 0 ein Rückkopplungsverfahren aktiviert werden. Ist Bit 0 im Kontrollstatuswort gesetzt wartet der BACnet-Stack auf das Schreiben der SPS-Daten indem die Backup-Anforderung eines BACnet-Client erst weiterverarbeitet wird bis Bit 1 im Kontrollstatuswort gesetzt ist. Ein entsprechender Ablauf in der SPS ist dabei:

1. Beim Start Bit 0 im Kontrollstatuswort setzten
2. SystemStatus überwachen - ist der Wert BackupInProgress (5): Schreiben der relevanten Daten
3. Nach Abschluss des Schreibvorgangs setzen von Bit 1 im Kontrollstatuswort
4. Wenn SystemStatus wieder den Wert Operational (0) hat: Löschen von Bit 1

Eine weitere für das Backup relevante Datei ist "BACnet_LastRestoreTime.wbp", die im Boot-Ordner von TwinCAT gespeichert wird. In dieser Datei werden Daten abgelegt, die ein Restore überdauern müssen. Hierzu zählen die Property LastRestoreTime und DatabaseRevision.

Das Auslösen eines Backups bzw. Restore wird momentan von TwinCAT BACnet/IP nicht unterstützt. Mit einem entsprechenden GLT-Werkzeug können aber Backup und Restore der Beckhoff Controller erstellt werden. Es wird also hier nur die Server- und nicht die Client-Seite unterstützt.

2.8 Persistente Daten

Werden via BACnet bzw. über den Online-Dialog BACnet-Properties von Server-Objekten verändert, sind diese bei einem TwinCAT-Neustart zunächst verloren, da die entsprechenden Werte aus dem Settings-Dialog zur Initialisierung verwendet werden. Um Daten auch bei einem Geräteneustart persistent zu halten, kann im BACnet-Server im Reiter "Settings" die Funktion Persist Server Online Data verwendet werden. Ist diese Funktion aktiviert, werden beim Herunterfahren und ggf. zyklisch die Werte aller BACnet-Properties, die sich gegenüber der Initialkonfiguration verändert haben, in eine Datei gespeichert und beim Laden einer Konfiguration wieder eingespielt. Die Datei wird im Boot-Verzeichnis der TwinCAT-Installation angelegt ("C:\TwinCAT\Boot") und hat den Namen "BACnetOnline_1010010.wbp". Der Zahlen-Code identifiziert den Server über eine TwinCAT(TcCOM)-Objekt-ID.

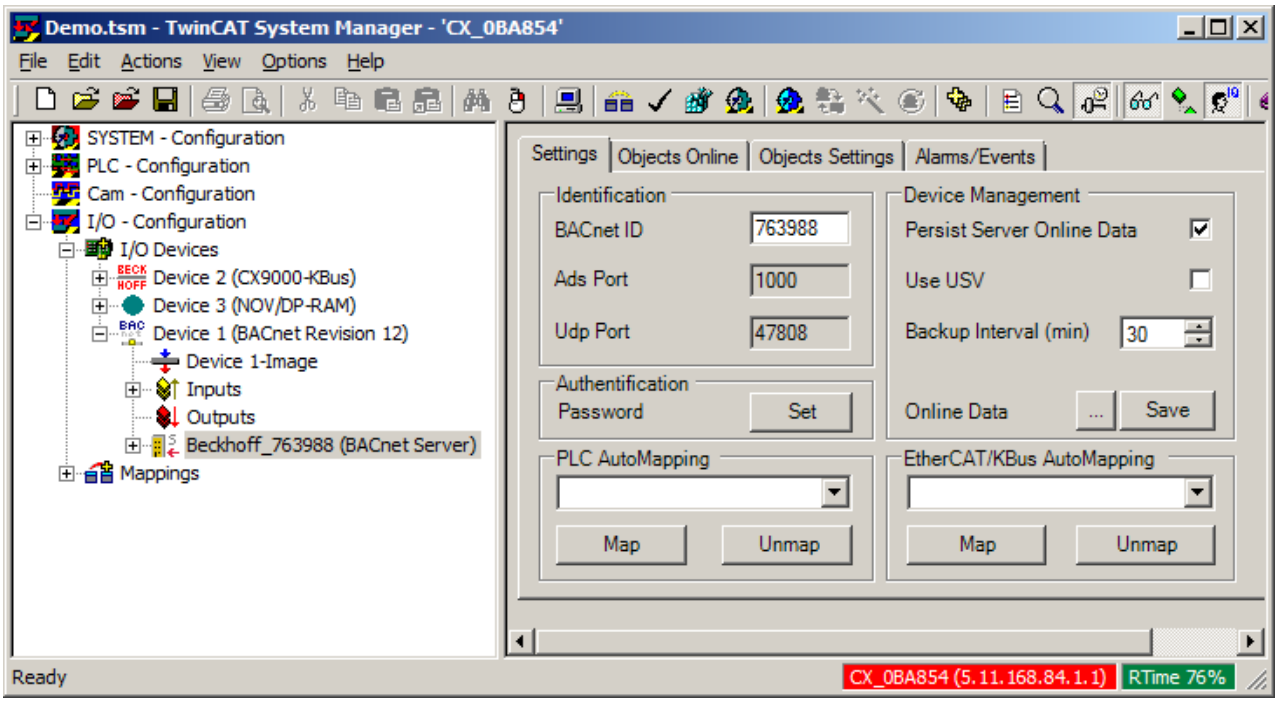
Geräte mit USV können auch bei einem Versorgungsspannungsverlust heruntergefahren werden und damit das Speichern persistenter Daten sicherstellen. Bei Geräten ohne USV muss zyklisch bzw. bei einer Veränderung von Daten gespeichert werden. Im Reiter "Settings" des BACnet-Servers kann über die Option "*Use USV*" gesteuert werden, ob auch während des Betriebs Daten in einem festgelegten Intervall persistiert werden. Über das Dialogfeld *"Backup Interval*" kann in Minuten konfiguriert werden in welchen Abständen veränderte BACnet-Properties gesichert werden. Dabei wird nach einem abgelaufenen Intervall nur gesichert, wenn seit der letzten Sicherung eine BACnet-Property verändert wurde.

Die Option "Use USV" realisiert nicht das automatische Herunterfahren eines Geräts bei einem Stromausfall. Die Überwachung des USV-Status und entsprechende Aktionen können in der SPS mit Hilfe entsprechender Bibliotheken implementiert werden. Die Option "Use USV" schaltet lediglich das periodische Speichern persistenter Daten ab. Das Schreiben persistenter Daten muss dann manuell bzw. durch die SPS via ADS oder die Property PersistentData erfolgen. In der SPS-Bibliothek TcBACnetRev12.lib stehen Funktionen zum Auslösen des Speicherns bereit.

HINWEIS

Anzahl maximaler Schreibzyklen beachten

Werden Daten zyklisch auf ein Flash-Speichermedium gesichert, kann dies bei kleinen Backup Intervallen zur Zerstörung des Speichermediums führen. Deshalb ist darauf zu achten, dass über die angestrebte Betriebsdauer die Anzahl der vom Flash-Hersteller spezifizierten Schreibzyklen nicht überschritten wird.



Zu beachten ist, dass bei dieser Funktionalität alle Einstellungen, die nach dem Aktivieren der aktuellen TwinCAT-Konfiguration in den Reitern "Settings" der Objekte vorgenommen werden hinfällig sind! Da alle Einstellungen aus der BACnetOnline-Datei übernommen werden. Muss eine Konfiguration nachträglich geändert werden, muss die Funktion deaktiviert und die BACnetOnline-Datei gelöscht werden. Achtung: Nach dem Deaktivieren der Funktion ist die Konfiguration noch mit aktivierter Persistenzfunktion geladen; es

wird also beim Herunterfahren noch einmal die BACnetOnline-Datei erzeugt. Erst bei aktivierter Konfiguration ohne die Persistenzfunktion wird die Datei nicht mehr geschrieben. Mit dem neuen Dialog zur Verwaltung persistenter Daten können Veränderungen dennoch leicht übernommen werden. Eine Beschreibung folgt am Ende dieses Kapitels.

Auch dynamisch erzeugte BACnet-Objekte werden in die persistenten Daten aufgenommen und bei einem Neustart wieder angelegt.

Beim Anlegen einer neuen TwinCAT BACnet/IP Konfiguration (genauer eines BACnet-Servers) wird innerhalb des Projekts eine eindeutige Config-ID generiert über die die BACnetOnline-Dateien einer Konfiguration (CurrentConfig.xml) zugeordnet werden. Beim Laden der Online-Daten wird diese überprüft. Stimmt die Config-ID nicht mit dem geladenen Projekt überein, werden die Online-Daten nicht übernommen. In diesem Fall sind die Daten der Reiter "Settings" ausschlaggebend.

Laden und Speichern persistenter Daten

Im Folgenden wird der Ablauf von Laden und Speichern der Online-Daten beschrieben, der auch bei einem Stromausfall sicherstellt, dass immer eine gültige Version der persistenten Daten vorliegt. Das Laden und Speichern persistenter Daten wird nur im RUN Modus durchgeführt und wenn persistente Daten aktiviert wurden.

Ablauf: Laden der BACnetOnline-Datei

1. Wenn Vorhanden: Laden der Datei "BACnetOnline_XXXXXXXX.wbp"
2. Nach erfolgreichem Laden: Umbenennen der Datei "BACnetOnline_XXXXXXXX.wbp" in "BACnetOnline_XXXXXXXX.wb~"
3. Ist "BACnetOnline_XXXXXXXX.wbp" nicht vorhanden: Laden der Datei "BACnetOnline_XXXXXXXX.wb~"

Ablauf: Schreiben der BACnetOnline-Datei:

1. Umbenennen einer vorhandenen Datei "BACnetOnline_XXXXXXXX.wbp" in "BACnetOnline_XXXXXXXX.wb~"
2. Schreiben der geänderten Property- und Objektdaten in die Datei "BACnetOnline_XXXXXXXX.wbp.tmp"
3. Nach Abschluss des Schreibvorgangs: Umbenennen der Datei "BACnetOnline_XXXXXXXX.wbp.tmp" in "BACnetOnline_XXXXXXXX.wbp"

Es ist darauf zu achten, dass durch das beschriebene Verfahren doppelter Speicherplatz auf dem Flashspeichermedium benötigt wird!

Relevante Änderungen

Wie schon zuvor erwähnt, werden Onlinedaten nur gesichert, wenn entsprechende Änderungen vorliegen. Änderungen können ausgelöst werden durch:

- Beschreiben einer BACnet-Property via WriteProperty bzw. WritePropertyMultiple
- Manipulation von Listen via AddListElement, RemoveListElement
- Anlegen dynamischer Objekte via CreateObject
- Löschen dynamischer Objekte via DeleteObject
- Änderung von BACnet-Properties durch objektinterne Statemachines (z.B. ElapsedActiveTime von Binary-Objekten, LogBuffer im TrendLog-Objekt)
- Änderung der Broadcast Distribution Table (BDT) wenn BBMD aktiviert ist

Generell führen Änderungen der folgenden BACnet-Properties nicht zu persistenzrelevanten Änderungen:

- Dauerhaft schreibgeschützte BACnet-Properties (z.B. ProtocolRevision, ObjectType, ObjectIdentifier)
- ActiveCOVSubscription: Vorgegeben durch die BACnet-Spezifikation werden durchgeführte COV-Subscriptions nicht persistiert
- BACnet-Properties die sich durch andere BACnet-Property oder interne Systemzustände ergeben (LocalDate, LocalTime, StatusFlags, EventState, DeviceAddressBinding, ObjectList, ModificationDate, SystemStatus, PersitentData, ActivePriority)

- DaylightSavingsStatus und UtcOffset wenn die Zeit-Betriebssystemeinstellungen verwendet werden (Time management "Use operating system settings")

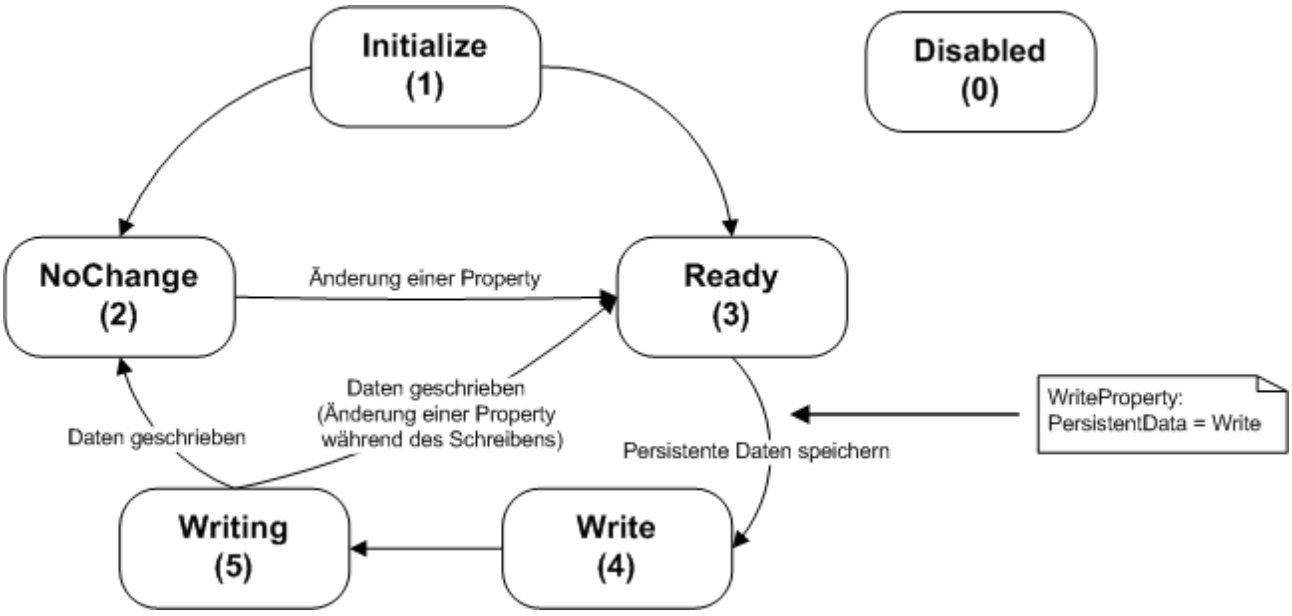
Manuelles Speichern persistenter Daten

Das Abspeichern persistenter Daten kann manuell/programmatisch ausgelöst werden. Es existiert eine azyklische (ADS) Schnittstelle, über die das Schreiben der BACnet-Online-Daten angestoßen werden kann:

IndexGroup	IndexOffset	Beschreibung
0x80C00000	immer 0	Abspeichern der persistenten Daten auslösen. Nur gültig für BACnet-Server, wenn Persist Online Data aktiv ist.

Mit Hilfe des System Managers kann über den BACnet-Server Reiter "Settings" über die Schaltfläche "Save" das Speichern via ADS ausgelöst werden.

Auch via BACnet-Netzwerk kann über die herstellerspezifische BACnet-Property PersistentData (des Device-Objekts) das Abspeichern der persistenten Daten ausgelöst werden. Die folgende Abbildung zeigt ein Zustandsdiagramm der BACnet-Property PersistentData.



Die BACnet-Property PersistentData ist nur sichtbar, wenn die persistenten Daten aktiviert und die Konfiguration im RUN Modus ausgeführt wird. Nach der Initialisierung zeigt die Property vom BACnet-Datentyp Enumerated, ob sich seit dem letzten Speichern persistenter Daten bzw. nach dem Projektstart eine relevante BACnet-Property geändert hat bzw. dynamische Objekte angelegt wurden. Bei unveränderten Daten hat die Property PersistentData den Wert NOCHANGE (2). Werden Daten via BACnet (WriteProperty, WritePropertyMultiple, CreateObject, DeleteObjekt, WriteBDT), via ADS-Write oder durch eine interne Objektzustandsmachine verändert, wird dies durch die Property PersitentData mit dem Wert Ready (3) angezeigt.

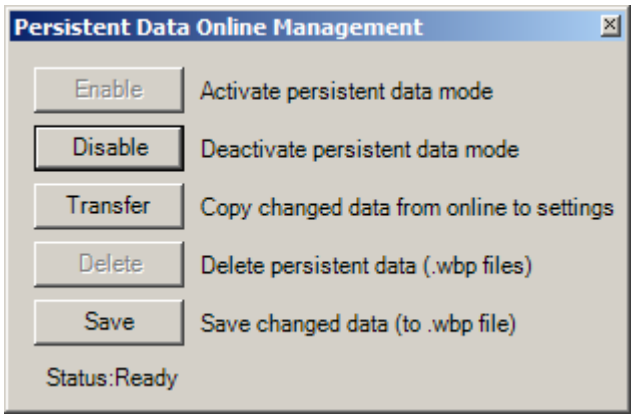
Hat die Property PersistentData den Wert Ready (3) kann durch Schreiben des Wertes Write (4) via BACnet das Schreiben ausgelöst werden. Während des Schreibens hat PersitentData den Wert Writing (5). Nach erfolgreichen Schreiben der persistenten Daten geht PersistentData in den Zustand NoChange (2) bzw. Ready (3), je nachdem ob sich während des Schreibens relevante Daten verändert haben.

Werden persistente Daten durch manuelles Auslösen des Schreibvorgangs gesichert, empfiehlt es sich das Backup Intervall auf ein entsprechend hohen Wertes zu setzten bzw. die periodische Sicherung mit der Funktion "*Use USV*" zu deaktivieren.

Änderungsmanagement persistenter Daten

Als Hilfestellung bei der Verwaltung persistenter Daten wurde in TwinCAT BACnet/IP ein neuer Dialog integriert. Über die Schaltfläche "..." neben dem Save-Button auf dem BACnet-Server - "Settings" - Reiter kann dieser Dialog aktiviert werden. Über diesen Dialog kann das Speichern der persistenten Daten temporär zur Laufzeit deaktiviert bzw. aktiviert (auch wenn es zuvor nicht in der Konfiguration aktiv war) werden. Das temporäre Deaktivieren wirkt dabei nicht über einen Neustart einer Konfiguration hinaus;

sondern nur bis zum nächsten Neustart bzw. temporären Aktivieren dieser Funktion. Mit der Transfer-Funktion können (online) geänderte Daten in den Settings-Dialog (und damit in die .tsm Datei) übernommen werden. Achtung: Im Gegensatz zur Transfer-Funktion des Property-Wizard, werden hier nur online-geänderte Daten übernommen. Property-Daten die im Settings-Dialog verändert wurden; online aber nicht; bleiben hier erhalten. Über "Delete" können die ".wbp"-Dateien gelöscht werden. Save löst das Speichern der peristenten Daten aus.



Sollen in einem Projekt mit aktivierten persistenten Daten, Änderungen eingespielt werden bzw. nach einer Inbetriebnahme online-geänderte Daten in der .tsm-Datei gesichert werden, empfiehlt sich folgende Vorgehensweise:

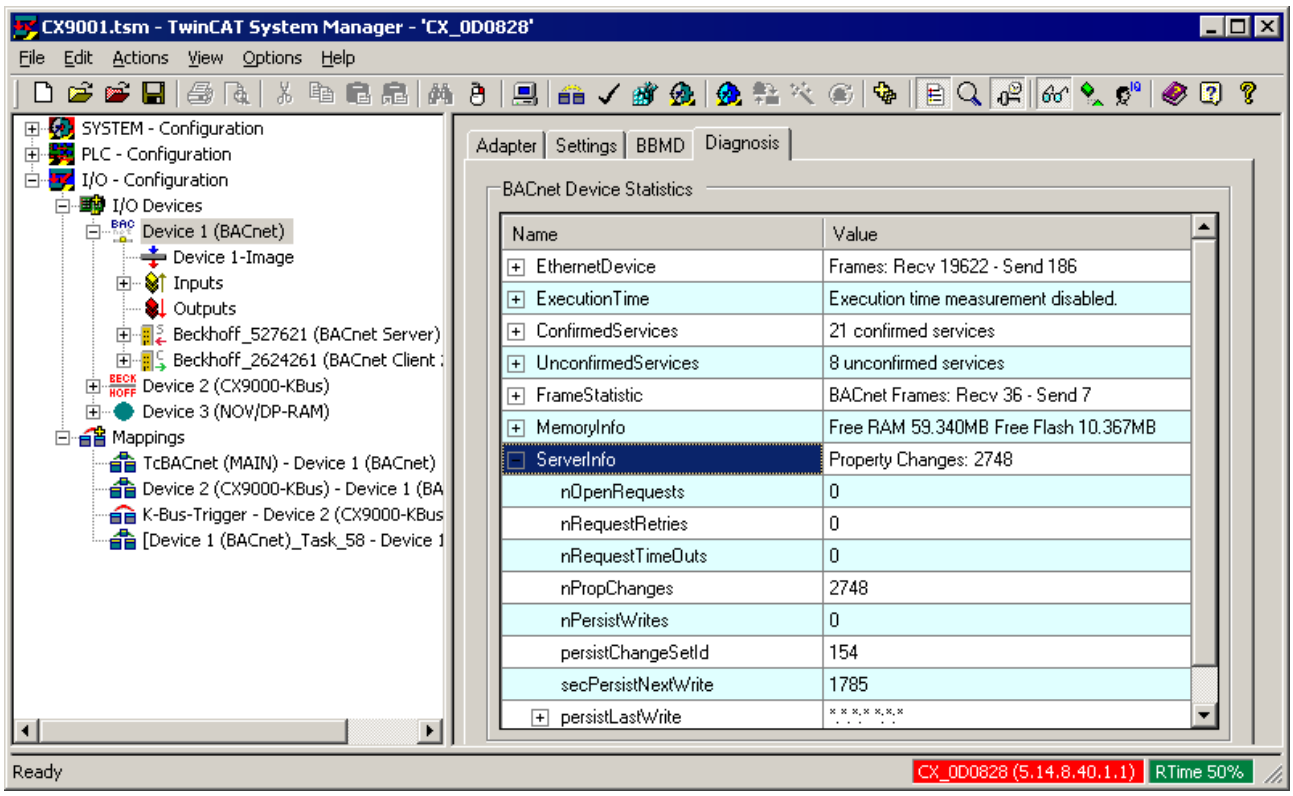
1. Auslösen der Transfer-Funktion (geändert Daten werden von Online nach Settings übernommen)
2. Deaktivieren des Schreibens persistenter Daten (Disable-Button)
3. Löschen der persistenten Daten, da sich die Änderungen nun im .tsm befinden (Delete-Button)
4. Abspeichern der Konfiguration (.tsm) (optional)
5. Aktivieren der Konfiguration (SystemManager - Actions - Activate Configuration ...)
6. Neustart der Steuerung

Diagnose persistenter Daten

Im Online-Reiter der BACnet-Objekte wird für jede BACnet-Property angezeigt, ob diese gegenüber dem Settings-Dialog verändert wurde. Ein * bedeutet, dass eine BACnet-Property geändert wurde. Ein zusätzliches Häkchen bedeutet, dass die Änderung in der .wbp-Datei gesichert wurde.

	InactiveText	46	FALSE
*	⊞ ChangeOfStateTime	16	20.06.2014 09:46:12
*	ChangeOfStateCount	15	1
*	⊞ TimeOfStateCountReset	115	20.06.2014 09:45:57
*	ElapsedActiveTime [s]	33	16
*	⊞ TimeOfActiveTimeReset	114	20.06.2014 09:45:57
	MinimumOfftime [s]	66	0

Mit Hilfe der Diagnosefunktionalität des BACnet-Device können einige Informationen zur Verarbeitung persistenter Daten betrachtet werden.



Unter ServerInfo befinden sich Persistenz spezifischen Informationen:

- **nPropChanges**: Gibt an wie oft BACnet-Properties seit dem Systemstart verändert wurden.
- **nPersistWrites**: Gibt an wie oft die BACnetOnline-Datei seit dem Systemstart geschrieben wurde.
- **persistChangeSetId**: Nach jedem Schreiben der Online-Datei wird in BACnet-Treiber eine ChangeSet-ID inkrementiert, um erkennen zu können ob sich seit dem letzten Schreiben BACnet-Properties verändert haben. Diese ChangeSet-ID wird in die Online-Datei geschrieben und zeigt damit an wie oft ein Projekt seit dem ersten Start gesichert wurde. Bei einem BACnet-Restore wird entsprechend die ChangeSet-ID der geschriebenen Online-Datei verwendet.
- **secPersistNextWrite**: Gibt an in wie viel Sekunden das nächste zyklische Schreiben persistenter Daten ausgelöst wird.
- **persistLastWrite**: Gibt an wann zuletzt die Online-Datei geschrieben wurde. Wurde sie seit dem Systemstart noch nicht geschrieben, ist dieser Zeitstempel mit *.*.*.* *.*.*.* initialisiert.

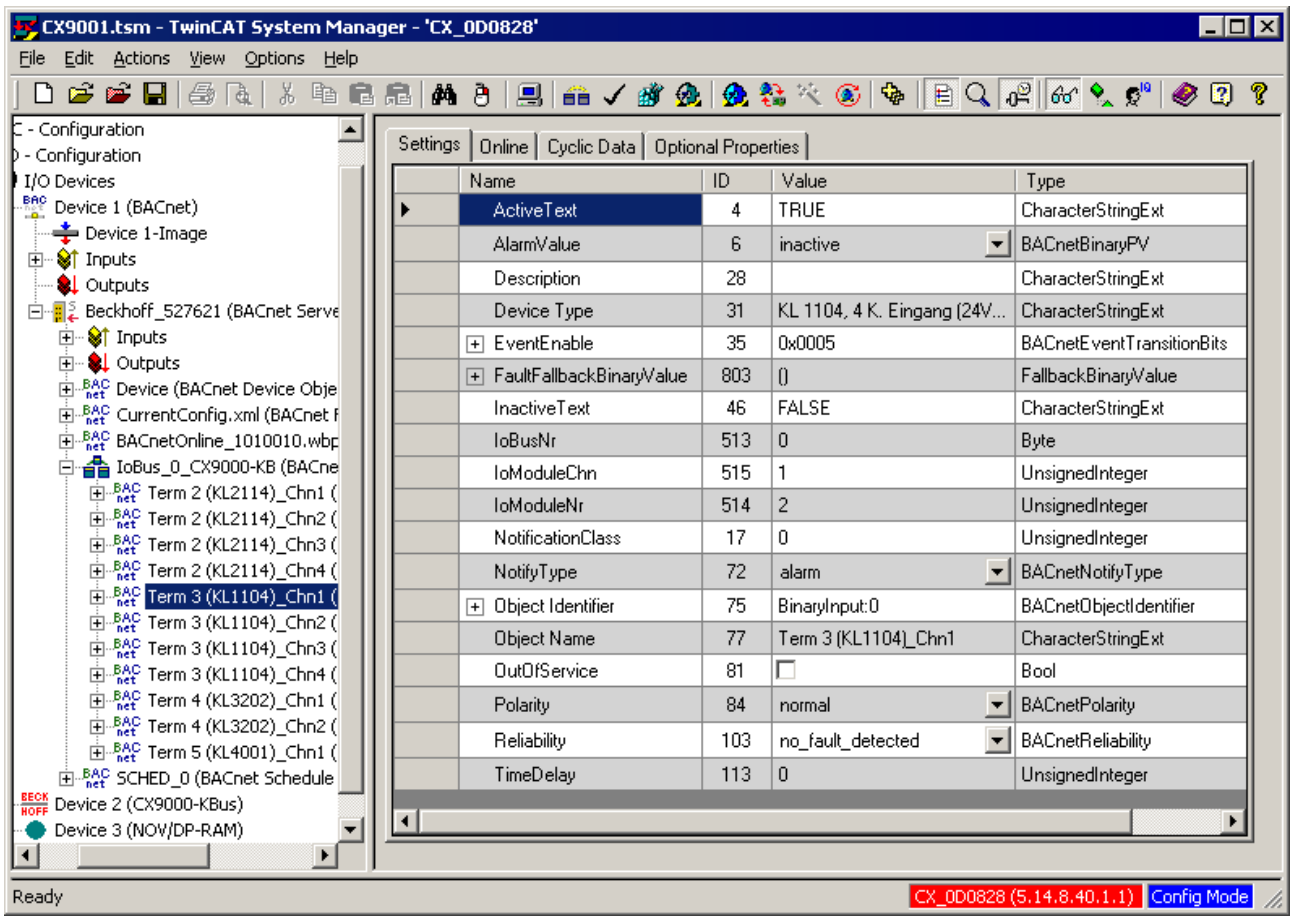
Eine genauere Analyse, welche Property-Daten verändert wurden und in der .wbp-Datei gespeichert sind, erlaubt der WbpViewDialog. Dieser kann als Wizard über das File-Objekt der .wbp-Datei aktiviert werden. Der Wizard zeigt in einer Text-Übersicht den Inhalt der binären wbp-Daten.

2.9 SPS-Automapping

Einführung

Mit Hilfe von speziellen Kommentaren, die hinter die SPS-Variablen-Deklaration eingefügt werden, kann eine automatisierte Konfiguration von BACnet-Objekten aus dem SPS-Programm vorgenommen werden. Dabei können BACnet-Objekte angelegt, BACnet-Property-Variablen mit SPS-Variablen verknüpft und BACnet-Properties initialisiert werden.

Das SPS-Automapping kann über den BACnet-Server im Reiter "Settings" durchgeführt werden, nachdem das SPS-Projekt in die Hardware-Konfiguration eingefügt wurde. Durch Betätigung des Buttons "Map", nach der Selektion des gewünschten SPS-Programms, wird das Automapping ausgelöst:



SPS-Automapping für BACnet-Server

Im folgenden Quellcodeauszug ist die Verwendung der Automapping-Kommentare veranschaulicht. Es wird eine lokierte Variable "av0" vom Typ REAL angelegt. Die verwendeten Kommentare definieren BACnet-Properties für die Verwendung des Automappings. Die generelle Kommentar-Syntax ist durch die SPS vorgegeben und wird durch "(*" eingeleitet und "*)" abgeschlossen. Kommentare mit dem speziellen Präfix "~" werden vom SPS-Compiler für die entsprechenden Symbole in die TPY-Datei überführt. Diese als "Property" bezeichneten speziellen Kommentare haben immer den Aufbau "(NAME : WERT : KOMMENTAR)". Für das BACnet-Automapping setzt sich der "NAME" aus "BACnet_[BACnet-Property-Name]" zusammen. "WERT" gibt je nach BACnet-Property einen entsprechenden Wert vor, die im Speziellen erläutert werden.

Im Beispiel wird das Symbol "av0" mit 3 BACnet Properties annotiert. Es wird festgelegt, dass es sich um ein AnalogValue-Objekt handelt (AV). Die Objekt-Instanz ist 100. Bei der Property ObjectIdentifier ist zu beachten, dass diese im SPS-Kommentar nur die Instanz-Nummer und nicht zusätzlich den kodierten Objekt-Typ wie in BACnet enthält. Im Beispiel wird beim Automapping also ein BACnet-Objekt vom Typ AnalogValue mit der Instanznummer 100 angelegt. Pro Symbol kann eine Verknüpfung mit einer zyklischen BACnet-Property-Variable vorgenommen werden. Variablen die nicht verknüpft werden sollen, können mit dem Kommentar "NOLINK" (siehe Anmerkung im Anhang) versehen werden. Pro Symbol darf genau eine Property zum Verknüpfen existieren. Im Beispiel wird die Property PresentValue des AnalogValue-Objekts mit der SPS-Variable "av0" verknüpft.

```
av0 AT %I* : REAL;          (* ~ (BACnet_ObjectType      :  AV         : NOLINK)
                      (BACnet_ObjectIdentifier   :  100        : NOLINK)
                      (BACnet_PresentValue       :          : )
                  *)
```

Im Folgenden ist ein weiteres Beispiel dargestellt. In diesem Fall wird ein BACnet-Objekt vom Typ BinaryInput angelegt. Eine Objektinstanz ist nicht definiert. Beim Automapping wird dann automatisch eine verfügbare Objektinstanznummer generiert. Im Beispiel ist demonstriert wie die Werte bestimmter BACnet-Properties vorinitialisiert werden können. Die BACnet-Properties Description und ObjectName werden mit den Werten "Hello, BACnet" bzw. "BinaryInput_1" initialisiert. Nach dem Automapping sind dann im erzeugten BACnet-Objekt im Reiter "Settings" die entsprechenden Werte übernommen und werden beim Aktivieren der Konfiguration "Online" sichtbar. Zusätzlich wird der Objektname im System-Manager-Baum

verwendet. Verknüpft wird im Beispiel eine Variable vom Typ BOOL mit der Property PresentValueBool. Während der BACnet-Datentyp der PresentValues von Binär-Objekten ein Aufzählungstyp ist, wurde für den effizienten Umgang mit der SPS die Hilfs-Property PresentValueBool eingeführt, um boolesche Variablen verknüpfen zu können. Details hierfür können im Kapitel "Prozessdaten [▶ 43]" nachgelesen werden.

```
bi0 AT %I* : BOOL;          (* ~ (BACnet_ObjectType     : BI          : NOLINK)
                    (BACnet_Description    : Hello, BACnet  : NOLINK)
                    (BACnet_ObjectName     : BinaryInput_1  : NOLINK)
                    (BACnet_PresentValueBool   :         : )
                *)
```

Auch bei Verwendung von Variablen primitiven Typs können mehrere Variablen mit einem BACnet-Objekt verknüpft werden. Hierzu müssen bei zwei Variablen identische Objekttyp- und Objektinstanz-Parameter angegeben werden. Im Beispiel werden die BACnet-Properties PresentValue und StatusFlags mit dem AnalogValue-Objekt "101" verknüpft. Um versehentlich doppelt vergebene Objektinstanznummer als Fehler melden zu können, muss beim 2. Symbol, welches sich auf ein zuvor erzeugtes BACnet-Objekt bezieht, der Kommentar "OBJREF" ergänzt werden. "OBJREF" muss hierbei immer bei der BACnet-Property "ObjectIdentifiert" deklariert werden. Soll ein Objekt über den Namen über 2 Symbole zugeorgnet werden, muss bei der Deklaration der Property *ObjectName* "OBJREF" verwendet werden.

```
av0 AT %I* : REAL;          (* ~ (BACnet_ObjectType     : AV          : NOLINK)
                    (BACnet_ObjectIdentifier   : 101         : NOLINK)
                    (BACnet_PresentValue       :         :)
                *)
av0_StatusFlags AT %I* : WORD; (* ~ (BACnet_ObjectType     : AV          : NOLINK)
                    (BACnet_ObjectIdentifier   : 101         : OBJREF,NOLINK)
                    (BACnet_StatusFlags    :         :)
                *)
```

Kommandierbare Properties

Von der SPS kann auch schreibend auf BACnet-Objekte zugegriffen werden. Kommandierbare BACnet-Properties sind mit einem PriorityArray verbunden und unterstützen Schreibzugriffe mit den Prioritäten 1 bis 16. Durch die Verwendung von Prioritäten können andere Systemkomponenten Signale der SPS übersteuern. Generell gilt beim schreibenden Zugriff von der SPS auf BACnet, dass nur beim Programmstart, sowie bei Änderung ein Wert von BACnet übernommen wird. Wird vom BACnet-Netzwerk konkurrierend auf eine gleiche Prioritätsstufe geschrieben, kann es sein, dass der SPS-generierte Wert vom BACnet-Wert abweicht. Bei priorisierten Zugriffen sollte aber in den meisten Fällen eine Prioritätsstufe exklusiv verwendet werden. Bei mehreren potentiellen gleichprioren Änderungsquellen kann auch die BACnet-Property *RelinquishDefault* verwendet werden.

Im Beispiel ist eine "Q"-lokierte Variable definiert, die mit dem PresentValue eines AnalogOutput-Objektes verknüpft wird. Dabei führt das Schreiben der SPS zu einem Eintrag in Prioritätsstufe "8" des PriorityArray. Kommandierbar sind jeweils die PresentValue-Properties von BinaryValue-, BinaryOutput-, AnalogValue-, AnalagOutput-, MultistateValue- und MultistateOutput-Objekten. Die jeweils höchste aktive Priorität (1=höchste, 16=niedrigste) resultiert als PresentValue. Eine Priorität im PriorityArray ist aktiv, wenn sie nicht NULL ist. Sind alle Elemente im PriorityArray NULL nimmt *PresentValue* den Wert der BACnet-Property *RelinquishDefault* an.

```
ao0_PresentValue8 AT %Q* : REAL;(*~ (BACnet_ObjectType     : AO          : NOLINK)
                    (BACnet_PresentValue_Priority8 :         :)
                *)
```

Eine Besonderheit kommandierbarer Properties ist der Eintrag NULL im PriorityArray, der in BACnet ein eigener Datentyp darstellt - im Gegensatz zu den jeweiligen Wertdatentypen (REAL, BINARY_PV, UNSIGNED). Der Wert NULL, der eine Prioritätsstufe im PriorityArray deaktiviert, kann für die jeweiligen Objekte wie folgt über die verknüpfte Variable geschrieben werden:

- **BinaryValue, BinaryOutput** : Prioritätsbasierte PresentValues werden hier als WORD (bzw. einem Aufzählungstypen z.B.: E_BACNET_BINARYPV) verknüpft. Eine 0 führt zu einem Eintrag INACTIVE im PriorityArray, eine 1 zu ACTIVE und Werte größer 1 zu NULL.
- **AnalogValue, AnalogOutput**: Um einen NULL-Eintrag im PriorityArray zu erzeugen kann eine spezielle Bit-Codierung von REAL-Werten verwendet werden, die als NaN (not a number) definiert sind. In der BACnet-SPS-Bibliothek "TcBACnet.lib" kann dieser Wert mit Hilfe der Funktion "F_BACnet_NAN()" erzeugt werden. Zu beachten ist, dass mit diesem Zahlenwert keine Berechnungen in der SPS durchgeführt werden, da ansonsten ein Floating-Point-Exception ausgelöst wird. Ein Wert von 0.0 führt auch im PriorityArray zu einem Wert 0.0. Bei AnalogOutput-Objekten gilt zusätzlich: Wird ein Wert außerhalb von Min-/MaxPresValue geschrieben, führt dies zu NULL im PriorityArray.

- **MultistateValue, MultistateInput**: Wird der Wert 0 oder ein Wert größer der BACnet-Property NumberOfStates geschrieben, wird im PriorityArray ein Eintrag NULL erzeugt.

Strukturierte Programmierung

Für die strukturierte Verknüpfung von BACnet-Objekten werden auch Funktionsblöcke (FBs) beim Automapping unterstützt. Der abgebildete Quellcodeausschnitt zeigt die Deklaration eines BACnet-Funktions-Bausteins. Um beim Automapping betrachtet zu werden, müssen diese Bausteine mit dem Präfix "FB_BACnet" beginnen. Im Beispiel wurde ein FB für ein AnalogOutput-Objekt definiert, der als Vorlage in der BACnet-SPS-Bibliothek "TcBACnet.lib" vorliegt. Dabei werden die Properties PresentValue, StatusFlags, Reliability und OutOfService als Eingangsvariablen zur SPS verknüpft, die Property PresentValue mit der Priorität 8 schreibend.

```
FUNCTION_BLOCK FB_BACnet_AnalogOutput
VAR_OUTPUT
    ObjectType: E_BACNETOBJECTTYPE;       (* ~ (BACnet_ObjectType       : AO   : NOLINK) *)
    PresentValue AT%I*: REAL;          (* ~ (BACnet_PresentValue     :      :) *)
    StatusFlags AT%I*: UINT;           (* ~ (BACnet_StatusFlags      :      :) *)
    Reliability AT%I*: E_BACNETRELIABILITY;   (* ~ (BACnet_Reliability      :      :) *)
    OutOfService AT%I*: BOOL;          (* ~ (BACnet_OutOfService     :      :) *)END_VAR
VAR_INPUT
    PresentValue_Priority8 AT%Q*: REAL;       (* ~ (BACnet_PresentValue_Priority8 ::) *)END_VAR
```

Im Hauptprogramm (bzw. einem anderen FB, Programm) kann dieser FB instanziiert werden. Auch bei der Instanziierung können zusätzlich Kommentare angegeben werden. Kommentare auf Instanziierungsebene gewinnen gegenüber Kommentaren auf FB-Deklaraktionsebene. Im Beispiel wird ein AnalogOutput-Objekt angelegt, die Objektinstanz und die BACnet-Property Description auf Instanziierungsebene initialisiert. Durch die FB-Deklaration werden bei dem erzeugten BACnet-Objekt die Properties PresentValue, StatusFlags, Reliability und OutOfService sowie PresentValue_Priority8 verknüpft.

```
ao1 : FB_BACnet_AnalogOutput;         (* ~ (BACnet_ObjectIdentifier : 100 :NOLINK)
                           (BACnet_Description      : FB-basiertes AO: NOLINK)
                          *)
```

BACnet-FBs können hierarchisch verwendet werden, das heißt, es kann z.B. ein "FB_BACnet_Pumpe" definiert werden, der mehrere FB_BACnet_AnalogOutput-Instanzen enthält. Für Schachtelungstiefen größer eins wird jeweils ein StructuredView-Objekt im System-Manager angelegt. BACnet-Kommentare werden dabei nicht auf mehrere Hierarchieebenen heruntergereicht. Nur Instanziierungs- und Deklarationsebene werden unterstützt. Eine Ausnahme bildet der BACnet-Kommentar BACnet_ObjectIdentifier, der über mehrere Hierarchieebenen aufgelöst wird, um eine ID-Vergabe für geschachtelte FB's zu unterstützen.

Wird auf einer oberen Ebene eine Objekt-Instanz angegeben, hat diese Auswirkung auf potenziell viele Objekte des gleichen Typs. Bei der Generierung einer Objekt-ID wird von der angegebenen ID ausgegangen (ist z. B. eine Objektinstanz von 1000 angegeben, wird ab 1000 eine neue freie ID gesucht). Ein FB_BACnet_Raum mit 2 AV- und 2 BV-Objekten und angegebener Start-ID von 1000 wird also die BACnet-Objekte AV:1000, AV1001, BV:1000 und BV:1001 erzeugen. Die resultierenden Objekt-IDs werden vor der Ausführung der eigentlichen Mapping-Operation berechnet. Existieren Objekte mit den ermittelten IDs z.B. durch ein IO-Mapping schon, werden die existierenden Objekte verwendet bzw. verknüpft und keine neuen Objekte erstellt.

Neben verschachtelten FBs werden auch eindimensionale Arrays unterstützt. Das folgende Beispiel zeigt die Deklaration eines Arrays *raeume* vom Typ FB_BACnet_Raum, welches weitere BACnet-Objekte enthält. Auf diese Weise können systematisch Einheiten gleicher Funktionalität verwaltet werden. Es werden auch Arrays auf verschiedenen Schachtelungsebenen unterstützt. So kann z.B. ein BACnet-Objekt *fbHaus[1].EtageU.Raum[10].tmpSoll* generiert werden.

```
raeume : ARRAY[0..10] OF FB_BACnet_Raum;
```

Strukturierte Programmierung: Konfiguration von FB-Instanzen

BACnet-Kommentare auf FB-Deklarations-Ebene gelten für alle Instanzen des FB gleichermaßen. Für bestimmte BACnet-Properties kann eine Definition auf Instanzebene Anwendung finden. So können z.B. instanzbasierte ObjectIdentifier definiert oder die Hardwareanbindung auf Io-Module pro Instanz konfiguriert werden. Im Folgenden ist ein Beispiel dargestellt bei dem BACnet-Properties auf Instanz-Ebene festgelegt werden.

```
FUNCTION_BLOCK FB_BACnet_SubSub
VAR
  bi  : FB_BACnet_BinaryInput; (*~(BACnet_OutOfService : TRUE: NOLINK)*)
  bi2 : FB_BACnet_BinaryInput;
END_VAR
```

```
FUNCTION_BLOCK FB_BACnet_Sub
VAR
  sub1   : FB_BACnet_SubSub;
  sub2   : FB_BACnet_SubSub;
  subArr : ARRAY[0..1] OF FB_BACnet_SubSub;
END_VAR
```

```
PROGRAM SIMPLE
VAR
s1 :FB_BACnet_Sub; (* ~ { BACnet_ObjectIdentifier<sub1/bi>    : 1017   : }
            { BACnet_ObjectIdentifier<sub2>       : 1028   : }
            { BACnet_ObjectIdentifier<subArr[0]/bi>   : 5555   : }
            { BACnet_ObjectName<sub2/bi>          : biname : NOLINK }
           *)
END_VAR
```

Durch die Definition eines Instanzpfades nach dem BACnet-Property-Namen können BACnet-Property-Definitionen auf Sub-Elemente eines FB angewendet werden. Hierfür kann der Instanzpfad durch Angabe in "<" und ">" vorgegeben werden. Im obigen Beispiel wird z.B. den BACnet-Objekt bi der Instanz sub1 ein ObjectIdentifier 1017 vorgegeben. Einzelne Sub-Ebenen werden dabei durch "/" abgetrennt. Es können auch BACnet-Property-Definitionen an Zwischenebenen konfiguriert werden. Im Beispiel wird der ObjectIdentifier 1028 an die FB-Instanz sub2 übergeben. Diese veranlasst in diesem Beispiel, dass die BACnet-Objekte s1.sub2.bi die Object-ID 1028 und s1.sub1.bi2 die Objekt-ID 1029 bekommen. Die Instanzpfadangabe kann auch Array-Elemente enthalten.

```
ARRAY[01..03] OF FB_BACnet_AnalogValue; (* ~
   (BACnet_Description<[1]>:1. OG Raum 1 Temp: NOLINK)
   (BACnet_Description<[2]>:1. OG Raum 2 Temp: NOLINK)
   (BACnet_Description<[3]>:1. OG Raum 3 Temp: NOLINK)
*)
```

Auch die direkte Parametrierung von Array-Elementen auf Programm- oder globaler Ebene wird unterstützt. Im Beispiel wird die BACnet-Property Description von 3 Array-Elementen initialisiert.

Strukturierte Programmierung: View-Konfiguration

Beim SPS-Automapping werden für jede Verschachtelungstiefe in Form von Programmen und Funktionsblöcken StructuredView-Objekte angelegt, um die Strukturierung auf innerhalb der SPS auf BACnet abzubilden. Je nach verwendetem Anlagenkennzeichungssystem (AKZ) kann es notwendig sein die Benennung von BACnet-Objekten und deren Struktur mit einem Namenspräfix anzupassen. Mit Hilfe des SPS-Kommentars "BACnet_StructuredViewPath" ist es möglich für das SPS-Automapping eigene Namenspräfixe zu definieren. "BACnet_StructuredViewPath" kann eine beliebige Variable auf den gewünschten Strukturierungsebenen annotieren.

```
PROGRAM MAIN
VAR
  viewDummy : BOOL; (* ~     ( BACnet_StructuredViewPath        : \/Building1\/Floor2 : )
( BACnet_StructuredViewPathDescription : \/Haus 1\/Flur 2 : )
            ( BACnet_StructuredViewPathObjectId    : \/1000\/9: )
             *)
  room1 : FB_BACnet_Room; (* ~( BACnet_Description : Description of structured view object room1: )
*)
  room2 : FB_BACnet_Room;
END_VAR
```

Im Beispiel wird für alle BACnet-Objekte innerhalb des Funktionsblock FB_BACnet_Room nicht der Name MAIN.room1.XXX sondern Building1.Floor2.room1.XXX verwendet. Neben der Verwendung für den Objektnamen der BACnet-Objekte wird auch für jede Hierarchieebene ein StructuredView-Objekt angelegt. Hierarchieebenen werden durch die Zeichenkette "\/" definiert.

Mit Hilfe der SPS-Kommentare "BACnet_StructuredViewPathDescription" und "BACnet_StructuredViewPathObjectId" können die BACnet-Property Description sowie die ObjectIdentifier der erzeugten StructuredView-Objekte festgelegt werden. StructuredView-Objekte, die über komplexe Funktionsbausteine erzeugt werden; Im Beispiel room1 und room2, können über die SPS-Kommentare BACnet_Description und BACnet_ObjectIdentifier konfiguriert werden.

Anlagenkennzeichnung

Je nach gefordertem Anlagenkennzeichnungssystem kann es notwendig sein, die generierten BACnet-Objektnamen anzupassen. Per Default werden für die Trennung von Strukturebenen Punkte (".") verwendet. Durch die Angabe eines "StructureSeparator" kann an Stelle des Punkts auch ein anderes Symbol verwendet werden bzw. das Trennsymbol auch vollständig entfernt werden. Das folgende Beispiel zeigt die Definition des Trennsymbols "_". Das Trennsymbol sollte einmalig pro Projekt z.B. an einer globalen Variable definiert werden.

```
akzDummy : BOOL; (* ~( BACnet_StructureSeparator :_: ) *)
```

Schreibschutz und optionale Properties

Für die Steuerung von Schreibschutz und optionalen Properties wurden spezielle Kommentaroptionen eingeführt. Im Kommentarteil der SPS-Kommentare können in Form einer ","-separierten Liste zusätzliche Optionen angegeben werden. DISABLE deaktiviert dabei eine optionale Property. WRITEPROTECT aktiviert den Schreibschutz einer schreibbaren Property. Es ist darauf zu achten, dass die Option NOLINK bei der Konfiguration multipler Properties angegeben wird. Das folgende Beispiel zeigt die Konfiguration von Schreibschutz und optionalen Properties.

```
bo1: FB_BACnet_BinaryOutput; (* ~(BACnet_NotifyType          :    : DISABLE,NOLINK)
                 (BACnet_ChangeOfStateTime      :    : DISABLE,NOLINK)
                 (BACnet_ChangeOfStateCount     :    : DISABLE,NOLINK)
                 (BACnet_ActiveText         : AN     : WRITEPROTECT,NOLINK)
                 (BACnet_InactiveText       : AUS    : WRITEPROTECT,NOLINK)
                  *)
```

Bei BACnet-Properties mit invertierter Optionallogik, also BACnet-Properties die beim Anlegen eines BACnet-Objektes zunächst deaktiviert sind, kann der Kommentar ENABLE verwendet werden. Ein Beipiel hierfür ist das Aktivieren eines mittelwertbildenen Filters im AnalogInput-Objekt mittels der herstellerspezifischen BACnet-Property *AvgFilterCycles:*

```
fbAI : FB_BACnet_AnalogInput; (* ~(BACnet_AvgFilterCycles     : 20    : ENABLE,NOLINK ) *)
```

Initialisierung komplexer Property-Werte

Der Initialisierung von Property-Werten über das SPS-Automapping sind durch die verwendete SPS-Kommentar-Syntax Grenzen gesetzt. So können die reservierten Zeichen "(", ")", ":", "*" und " nicht verwendet werden. Zusätzlich kann bei BACnet-Properties mit Auswahltyp (Choice) oder optionalen Feldern nicht immer eindeutig jede Zeichenkette analysiert werden, wenn eine optimierte Darstellungsform wie im System Manager verwendet wird.

Für bestimmte Properties wurden spezielle Parser implementiert, die eine vereinfachte Konfiguration von Property-Werten ermöglichen. Im folgenden Beispiel ist dargestellt, wie die Property StateText eines MultiStateValue-Objekts initialisiert werden kann.

```
mv : FB_BACnet_MultiStateValue;  (* ~(BACnet_StateText   : {Low;Medium;High}       : NOLINK )*)
```

BitField-Properties (EventEnable, AckRequired) können durch die Angabe einer Hexadezimalzahl initialisiert werden. Hier empfiehlt sich Übernahme des entsprechenden Werts im System Manager:

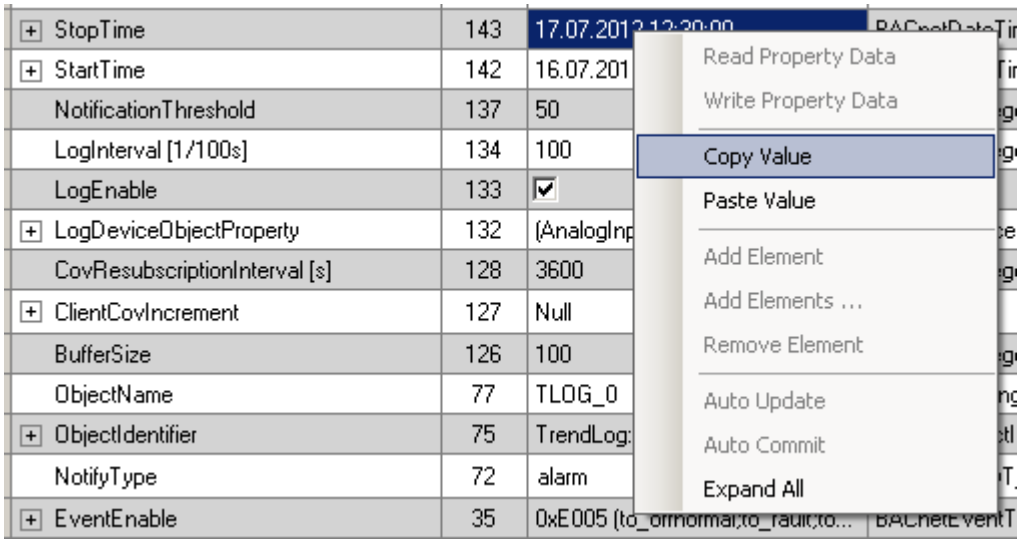
```
bi : FB_BACnet_BinaryInput;      (* ~(BACnet_EventEnable : 0xE005          : NOLINK )*)
```

Auch die Property Priority des Objekts NotificationClass kann verkürzt initialisiert werden. Hierfür wird diese Array-Property im Beispiel mit den Werten 12, 12 und 122 initialisiert. Für die Initialisierung von BitField-Properties wurde zusätzlich die übersichtlichere Form der Array-artigen Initialisierung implementiert. Im Beispiel werden in der Property AckRequired die Bits für "to_offnormal" und "to_fault" gesetzt. Eine BitField-Property mit keinem gesetzten Bit kann durch die Zeichenkette "{}" initialisiert werden.

```
nc : FB_BACnet_NotificationClass;(* ~(BACnet_Priority    : {12;12;222}         : NOLINK)
                     (BACnet_AckRequired : {to_offnormal;to_fault} : NOLINK)
                *)
```

Da nicht für alle komplexen Properties eigene Parser implementiert werden können, wurde die Möglichkeit der Initialisierung mit Hilfe von XML eingeführt. XML hat den Vorteil, dass die reservierten Zeichen der Kommentarsyntax nicht verwendet werden müssen und so keine Einschränkung bei der Initialisierung der Property-Werte besteht. Die XML-Werte für die Property-Initialisierung können sehr einfach durch die Copy&Paste Funktion von Property-Werten erstellt werden. Eine Property kann hierfür im System Manager unter Verwendung aller Komfort-Funktionen (Wizards) erstellt werden. Über das Kontextmenü in der

Property-Ansicht ("Online" bzw. "Settings") kann über "Copy Value" (bzw. die Tastenkombination STRG-C) der aktuelle Wert einer Property in die Zwischenablage eingefügt werden und im PLC-Control über "Einfügen" bzw. STRG-V verwendet werden.



Wird die SHIFT-Taste beim Auslösen der Funktion "Copy Value" gehalten, wird die XML-Zeichenkette als eine Zeile kopiert - sonst wird eine strukturierte Variante im Mehrzeilenmodus erzeugt. Die Verwendung des Einzeilenmodus ermöglicht eine übersichtlichere Variablen-Deklaration im SPS-Programm. Das nachfolgende Beispiel zeigt wie die Property StartTime eines TrendLog-Objekts mit Hilfe der XML-Syntax initialisiert wird:

```
tl : FB_BACnet_TrendLog; (* ~(BACnet_StartTime :
<BACnetDateTime>
  <date>
    <year>112</year>
    <month>7</month>
    <day>16</day>
    <dayOfWeek>1</dayOfWeek>
  </date>
  <time>
    <hour>12</hour>
    <minute>30</minute>
    <second>0</second>
    <hundredths>0</hundredths>
  </time>
</BACnetDateTime>
: NOLINK )*)
```

Property-Referenzen

Für die Initialisierung von Referenz-Properties wurde die Option REFERENCE eingeführt. Durch die automatische Generierung von Objekt-IDs können Referenzen auf Properties anderer SPS-BACnet-Objekte nicht durch die direkte Wert-Initialisierung von Properties erfolgen, da die Objekt-IDs zur Implementierungszeit des SPS-Programms unbekannt sind. Durch die Angabe der Option REFERENCE und einem Property-Namen als Suffix können automatisch generierte Objekte referenziert werden. Als Beispiel ist im Folgenden eine einfache Heizung skizziert. Die Angabe von Soll-, Ist- und Stellwert erfolgt über Analog*-BACnet-Objekte, die über Referenzen mit einem Loop-Objekt verknüpft werden.

```
fbSollwert   : FB_BACnet_AnalogValue;
fbStellwert  : FB_BACnet_AnalogOutput;
fbIstwert    : FB_BACnet_AnalogInput;
fbTempRegler : FB_BACnet_Loop; (* ~(BACnet_ManipulatedVariableReference : ./
fbStellwert : REFERENCE_PresentValue)
                  (BACnet_SetpointReference        : ./fbSollwert  : REFERENCE_PresentValue)
                  (BACnet_ControlledVariableReference  : ./fbIstwert   : REFERENCE_PresentValue)
               *)
```

Bei der REFERENCE-Funktionaliät wird zwischen relativer und absoluter Referenzierung unterschieden. Relative Referenzen beziehen sich immer auf den aktuellen Kontext und werden mit "./" eingeleitet. Der aktuelle Kontext bezieht sich immer auf die aktuelle Deklarationsebene im Programm bzw. Funktionsbaustein. Mit Hilfe relativer Referenzierung können auch Objekt in anderen Programmen bzw. Funktionsbausteinen verwendet werden. Ein Beispiel ist die Deklaration eines LOOP-Objekts im "MAIN"-Programm; um ein Objekt im Programm "IO_OBJECTS" zu referenzieren kann die Syntax "./../

IO_OBJECTS/fbAO" verwendet werden. Absolute Referenzen beginnen immer direkt mit dem Programmnamen (z.B. MAIN.fbBvI oder MAIN.haus1.raum1.uv24Vok). Bei globalen Variablen muss der "." am Beginn entfernt werden.

Generell beziehen sich alle Referenzen auf Symbolnamen in der SPS nicht auf den BACnet-Objektnamen!

Generell gilt, das Referenzen die potentiell BACnet-Objekte in anderen Geräten (BACnetDeviceObjectPropertyReference) enthalten können, auch mit der Funktion REFERENCE initialisiert werden können. Handelt es sich bei dem Zielsymbol um ein FB_BACnet_Remote_*-Baustein, wird die Ziel-Geräte-ID automatisch eingetragen.

Die Option REFERENCE funktioniert auch für Schedule- und TrendLog-Objekte:

```
bi      : FB_BACnet_BinaryInput;
bv      : FB_BACnet_BinaryValue;
tl      : FB_BACnet_TrendLog; (* ~(BACnet_LogDeviceObjectProperty       : ./
bi      : REFERENCE_PresentValue) *)
sched   : FB_BACnet_Schedule; (* ~(BACnet_ListOfObjectPropertyRefs      : ./
bv      : REFERENCE_PresentValue) *)
```

Für entfernte Objekte zeigt der folgenden Ausschnitt ein Beispiel. Im Beispiel wird eine absolute Referenz innerhalb des MAIN-Programms verwendet.

```
bvRemote : FB_BACnet_RemoteBinaryValue; (* ~ (BACnet_ObjectType       : BV : NOLINK)
                        (BACnet_DeviceID     : 32 : NOLINK)
                        (BACnet_ObjectIdentifier : 2 : NOLINK) *)
fbTLog   : FB_BACnet_TrendLog;      (* ~ (BACnet_LogDeviceObjectProperty : MAIN.bvRemote : REFERENCE
_PresentValue) *)
```

Mehrere Referenzen in der Property *ListOfObjectPropertyRefs* des Schedule-Objekts können über folgende Syntax festgelegt werden:

```
 schedMultiRef : FB_BACnet_Schedule;     (* ~ (BACnet_ListOfObjectPropertyRefs : Object=COOLING.bPrg
Cool;Property=PresentValue;
                                      Object=fbGlob;Property=PresentValue; : REFERENCE) *)
```

Auch für weitere komplexere BACnet-Properties mit Referenzen existiert eine spezielle Initialisierung, die im Folgenden anhand kurzen Beispiels vorgestellt werden soll. Zunächst folgt die Initialisierung der *Action*-Property des Command-Objekts, welche für die Steuerung von ablaufbasierten Vorgängen verwendet werden kann.

```
fbCmdBo  : FB_BACnet_BinaryOutput;
fbCmdBV19: FB_BACnet_BinaryValue; (* ~(BACnet_ObjectIdentifier : 19 : ) *)fbCmdAv  : FB_BACnet_Analo
gValue;
fbCmdMv  : FB_BACnet_MultiStateValue;
fbCmd    : FB_BACnet_Command; (* ~(BACnet_Action :
   Action=1;Command=1;Object=./
fbCmdBo;Property=PresentValue;Value=ACTIVE;Priority=16;Delay=0;Quit=TRUE;
   Action=1;Command=2;Object=./fbCmdBo;Property=PresentValue;Value=NULL;Delay=1;Quit=FALSE;
   Action=1;Command=3;Object=BinaryValue_19;Property=PresentValue;Value=NULL;Delay=1;Quit=FALSE;
   Action=1;Command=4;Object=./fbCmdAv;Property=PresentValue;Value=NULL;
   Action=1;Command=5;Object=./fbCmdAv;Property=PresentValue;Value=12.3;Delay=1;Quit=FALSE;
   Action=2;Command=1;Object=./fbCmdMv;Property=PresentValue;Value=2;Delay=1;Quit=FALSE;
: REFERENCE )*)
```

Im folgenden Beispiel wird die Initialisierung der ListOfGroupMembers-Property des Group-Objekts dargestellt.

```
fbGroupAv:  FB_BACnet_AnalogValue;
fbGroupBv : FB_BACnet_BinaryValue;
fbGroup   : FB_BACnet_Group;(* ~(BACnet_ListOfGroupMembers :
   Object=./
fbGroupAv;Property=PresentValue;Property=StatusFlags;Property=HighLimit;Property=LowLimit;
   Object=./fbGroupBv;Property=PresentValue;Property=StatusFlags;Property=OutOfService
: REFERENCE ) *)
```

Mit Hilfe des EventEnrollment Objektes können Alarmbedingungen definiert werden, die über die des Instrinsic Reporting hinausgehen. Im Folgenden wird die Initialisierung der EventEnrollment-Objekte mit Hilfe einer verkürzten Syntax demonstriert:

```
fbEEAv   : FB_BACnet_AnalogValue;
fbEEWarn : FB_BACnet_EventEnrollment;(* ~
   (BACnet_ObjectPropertyReference : ./fbEEAv : REFERENCE_PresentValue)
   (BACnet_EventParameters     : EventType=out_of_range;timeDelay=5;lowLimit=0;highlimit=10;deadband
=0;:REFERENCE )
   (BACnet_NotificationClass       : 10 : nolink )
   (BACnet_EventMessageTexts       : {Warning;Sensorfault;Normal operation} : nolink )
*)
```

```
fbEEBo   : FB_BACnet_BinaryOutput;
bEEHours : FB_BACnet_EventEnrollment; (* ~
   (BACnet_ObjectPropertyReference : ./fbEEBo : REFERENCE_ElapsedActiveTime)
   (BACnet_EventParameters     : EventType=unsigned_range;highlimit=60; :REFERENCE )
   (BACnet_NotificationClass       : 10 : nolink )
   (BACnet_EventMessageTexts       : {Maximum operating hours exceeded; ; } : nolink )
*)
```

Einbindung von Hardware-Modulen

Die Anbindung von Hardware-Modulen kann grundsätzlich über das Io-Automapping erfolgen und die Zuordnung zu SPS-Objekten durch die Angabe von ObjectIdentifiern im SPS-Code. Um eine effiziente Code-Generierung von BACnet-Projekten zu ermöglichen, ist es auch möglich die Hardware-Anbindung direkt hinter der Deklaration von BACnet-Objekten in der SPS zu konfigurieren. Hierfür existiert die LINKPATH-Option. Mit dieser Option ist es möglich eine BACnet-Property-Variable eines BACnet-Objekts direkt mit einem IO-Modul zu verknüpfen. Hier wird dann keine Verlinkung zu einer SPS-Variablen, sondern eine reine Device-Device-Verknüpfung erstellt.

Das folgende Beispiel zeigt die Verknüpfung eines BACnet-Objekt BinaryInput mit der 3. Klemme (einer KL1002) auf dem 1. Kanal. Über das PresentValue des FB_BACnet_BinaryInput kann dann im SPS-Programm direkt auf den Hardware-Wert zugegriffen werden.

```
bi : FB_BACnet_BinaryInput;(* ~{ BACnet_RawIoBinaryBoolValue : TIID^Rt-
Ethernet^BK9000^Term 3 KL1002^Channel 1^Input : LINKPATH } *)
```

Durch die Angabe eines zusätzlichen Parameters der LINKPATH-Option können Offset1, Offset2 und die Bitgröße für die Verknüpfung auch direkt vorgegeben werden (LINKPATH<[Offset1],[Offset2],[Bitgröße]>).. Ist die Bitgröße auf den Wert 0 festgelegt, wird automatische die maximal mögliche Anzahl von Bits verknüpft. Ohne die Angabe der zusätzlichen Parameter wird für Offset1 und Offset2 der Wert 0 verwendet. Offset1 bezieht sich auf die BACnet-Property-Variable und Offset2 auf den externen Link.

```
bx : FB_BACnet_BinaryInput;(* ~{ BACnet_RawIoBinaryBoolValue : TIID^Rt-
Ethernet^BK9000^Term 3 KL1002^Channel 1^Input : LINKPATH<0,0,1> } *)
```

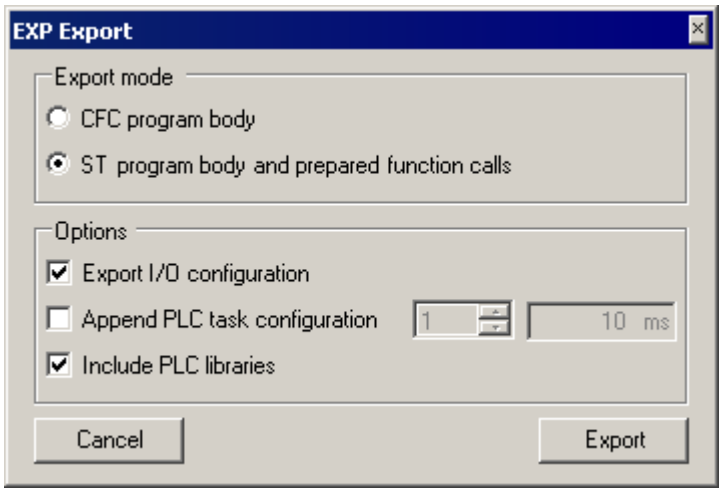
Im Zusammenhang mit Funktionsbausteinen können auch Sub-Elemente von FB-Instanzen direkt mit einem Hardware-Modul verbunden werden. Hierfür kann die Instanzpfad-Syntax "<[]>" nach der zu verknüpfenden BACnet-Property verwendet werden.

```
PROGRAM SIMPLE
VAR
s1 :FB_BACnet_Sub; (* ~ { BACnet_RawIoBinaryBoolValue<sub1/bi>  : TIID^Rt-
Ethernet^BK9000^Term 3 KL1002^Channel 1^Input : LINKPATH }
            { BACnet_RawIoBinaryBoolValue<sub2/bi>  : TIID^Rt-
Ethernet^BK9000^Term 3 KL1002^Channel 2^Input : LINKPATH }
            { BACnet_RawIoBinaryBoolValue<sub1/bi2> : TIID^Rt-
Ethernet^BK9000^Term 2 KL1002^Channel 2^Input : LINKPATH }
            *)
END_VAR
```

Zusätzlich ermöglicht das SPS-Automapping auch die Verknüpfung von Variablen außerhalb von BACnet. Mit der Option "ExtLink" kann ein beliebieges, lokiertes SPS-Symbol mit einer externen Variablen z.B. einem Hardware-Modul verknüpft werden. Mit der Option "ExtLink2" können 2 SPS-externe Variablen miteinander verknüpft werden. Über zusätzliche Parameter (wie auch bei LINKPATH) können Offset1, Offset2 und die Bitgröße für die Verknüpfung angegeben werden. Im Folgenden ist je ein Beispiel für die Verwendung dieser Optionen dargestellt.

```
b1 AT%I*: BOOL;   (* ~{ ExtLink  : TIID^Rt-Ethernet^BK9000^Term 2 (KL1002)^Channel 2^Input  : } *)
bDummy  : BOOL;   (* ~{ ExtLink2 : TIID^Rt-Ethernet^BK9000^Term 5 (KL2012)^Channel 1^Output :
                   TIID^Rt-Ethernet^BK9000^Term 2 (KL1002)^Channel 2^Input }
            *)
b1 AT%I*: BOOL;   (* ~{ ExtLink<0;0;1>  : TIID^Rt-
Ethernet^BK9000^Term 2 (KL1002)^Channel 2^Input  : } *)
bDummy  : BOOL;   (* ~{ ExtLink2<0;0;1> : TIID^Rt-
Ethernet^BK9000^Term 5 (KL2012)^Channel 1^Output :
                     TIID^Rt-Ethernet^BK9000^Term 2 (KL1002)^Channel 2^Input }
            *)
```

Als Ausgangspunkt für die Verknüpfung von Hardware-Modulen kann der EXP-Export verwendet werden. Im EXP-Export-Konfigurationsdialog kann mit der Option "Export I/O configuration" die BACnet-Objekt-Konfiguration für die TcBACnet.lib erzeugt werden. Hierbei wird für jeden im Projekt vorhandenen Io-Bus ein entsprechendes Programm mit den Hardware-BACnet-Objekten und ihrer Verknüpfung erstellt. Die Prozedur EXP-Export, SPS-Automapping erzeugt dann die gleiche BACnet-Konfiguration, die auch ein IO-Automapping erzeugt. Der Vorteil der SPS-Variante ist, dass Hardware-Objekte in strukturierte Funktionsbausteine der SPS integriert werden können bzw. Hardware-Kanäle, die nicht via BACnet sichtbar sein sollen, gelöscht werden können.



SPS-Automapping für BACnet-Clients

Sollen SPS-Variablen mit Prozessdaten-Properties von entfernten (remote) BACnet-Objekten verknüpft werden, dann kann als Kommentar zusätzlich zum Objekttyp und -instanz eine Device-ID angegeben werden. In diesem Fall werden SPS-Variablen mit zuvor angelegten BACnet-Objekten (z.B. durch Scannen von Clients) verknüpft. Ist ein Client mit noch nicht vorhanden wird dieser automatisch angelegt. Auch nicht vorhandende Objekte werden erzeugt. Die erzeugte Hierarchie ist dabei flach. StructuredView-Objekt für Clients werden vom SPS-Automapping momentan nicht unterstützt.

Im folgenden Quelltextausschnitt ist eine Variable vom Typ REAL angelegt. Diese wird beim Automapping mit der Property PresentValue des Objekts AnalogValue:100 unter dem Client mit der BACnet-ID "42" verknüpft. Die Prozessdatenvariable des remote BACnet-Objekts wird dabei automatisch aktiviert.

```
av0 AT %I* : REAL;          (* ~ (BACnet_ObjectType          : AV     : NOLINK)
                    (BACnet_DeviceID            : 42     : NOLINK)
                    (BACnet_ObjectIdentifier        : 100    : NOLINK)
                    (BACnet_PresentValue        :        :)
                *)
```

Für die Konfiguration der prozessdatenspezifischen Parameter von Client-Properties existieren Hilfs-Properties, um alle im System-Manager einstellbaren Optionen automatisiert konfigurieren zu können. Dabei handelt es sich um die Schreibpriorität bei PresentValue-Properties, die Zykluszeit, das COV-Increment und das COV-Resubscriptioninterval für COV-Subscriptions sowie Flags zur Konfiguration, ob Daten zyklisch, per COV oder OnChange verarbeitet werden sollen. Der folgende Quellcodeausschnitt zeigt die Verwendung dieser Hilfs-Properties.

```
ao0 AT %Q* : REAL;          (* ~ (BACnet_ObjectType          : AO     : NOLINK)
                    (BACnet_DeviceID            : 42     : NOLINK)
                    (BACnet_ObjectIdentifier        : 0      : NOLINK)
                    (BACnet_RemotePriority      : 8      : )
                    (BACnet_RemoteCycleTimeWrite    : 1000   : )
                    (BACnet_RemoteFlagsWrite        : WRITEONCHANGE : )
                    (BACnet_PresentValue        :    : )
                *)

ao1 AT %I* : REAL;          (* ~ (BACnet_ObjectType          : AO     : NOLINK)
                    (BACnet_DeviceID            : 42     : NOLINK)
                    (BACnet_ObjectIdentifier        : 0      : NOLINK)
                    (BACnet_PresentValue        :    : )
                    (BACnet_RemoteCycleTimeRead     : 1000   : )
                    (BACnet_RemoteFlagsRead     : COV    : )
                    (BACnet_RemoteCovincrement      : 0,5    : )
                    (BACnet_RemoteResubscriptionInterval: 60: )
                *)
```

Für die RemoteFlagsWrite können dabei die folgenden Werte - Für Ausgangsvariable (%Q) - Also vom projektieren Controller zum entfernten Gerät - verwendet werden:

- onchange, periodic

Für RemoteFlagsRead - Eingangsvariable (%I) - Lesend von einem entfernten Gerät gilt:

- cov, covpolling, polling, confirmedcov, rpm-o, rpm-c

Werden die Hilfs-Properties bei den Remote-Objekten nicht angegeben, dann werden die im System-Manager verwendeten Default-Einstellungen angewendet:

- RemotePriority : 16
- RemoteFlagsRead : 1 (COV)
- RemoteFlagsWrite : 1 (OnChange)
- RemoteCycleTimeRead : 1000
- RemoteCycleTimeWrite : 1000
- RemoteResubscriptionInterval : 240
- RemoteCovIncrement : 0.0

Bei Verwendung der Remote-Bausteine der TcBACnet.lib kann die Art der Prozessdatenübertragung auch bei der FB-Instanziierung angegeben werden. Die Parameter der Remote-Konfiguration beziehen sich dabei gemeinsam auf alle aktivieren BACnet-Properties. Im folgenden Beispiel wird also die Übertragungsart der Prozessdaten auf ConfirmedCOV gestellt bei den Properties des *EX-Bausteins: *StateOfChangeCount*, *PresentValue*, *StatusFlags*, *EventState* sowie *Reliability*.

```
bvRemote : FB_BACnet_RemoteBinaryValue_EX; (* ~ (BACnet_ObjectType : BV : NOLINK)
                        (BACnet_DeviceID   : 32 : NOLINK)
                        (BACnet_ObjectIdentifier : 20 : NOLINK)
                        (BACnet_RemoteFlagsRead : ConfirmedCOV : )
                        (BACnet_RemoteResubscriptionInterval : 3600 : )
                        *)
```

Bei prioritätsbasierten Properties (PresentValue) kann schreibend eine Priorität konfiguriert werden, um einen Zugriff auf höhere Prioritäten des PriorityArray zu aktivieren. Wie auch im Serverfall ist es möglich den Wert NULL in das zugeordnete PriorityArray zu schreiben, um diese Prioritätsstufe zu deaktivieren. Bei Analog- und Binary-Objekten geschieht das äquivalent zum Serverfall (Wert > 1 für Binary-Objekte; NaN für Analog-Objekte). Bei AnalogOutput-Objekten findet keine Überwachung des Min-/MaxPresValue-Bereichs statt - NULL kann hier also nur durch NaN erzeugt werden. Bei Multistate-Objekten führt abweichend vom Serverfall nur das Schreiben von 0 zu NULL im PriorityArray. Aus Performance-Gründen wird die Property NumberOfStates nicht vom entfernten Gerät ausgelesen.

Zusammen mit der TwinCAT-BACnet-SPS-Bibliothek "TcBACnet.lib" existiert eine effiziente Möglichkeit "gescannte" Client-Konfigurationen in der SPS zu verwenden. Nachdem die zu verwendenden Clients im System-Manger (z.B. durch Scannen eines BACnet-Netzwerks) eingefügt wurden, kann über den Reiter "Settings" des BACnet-Device via "Export" und Angabe einer Ziel-Datei mit der Endung "EXP" die Variablendeklaration und entsprechende Annotierung mit Automapping-Kommentaren generiert werden. Diese kann in ein SPS-Projekt im PLC Control importiert werden. Dabei werden die Basis-BACnet-FBs der Bibliothek verwendet (z.B. "FB_BACnet_RemoteAnalogInput").

Remap

Bei der Verwendung persistenter Daten existiert bei späteren Erweiterungen die Herausforderung, eine eindeutige Zuordnung der persitierten Daten und Ihrem Pendent im SPS-Programm herzustellen. Persistente Daten werden eindeutig über den ObjektIdentifier zugeordnet. Wird bei einer Erweiterung ein Objekt in ein Programm eingefügt, kann es vorkommen, dass nachfolgende, generierte BACnet-ObjectIdentifier verschoben werden und damit die Zuordnung der persistenten Daten zerstört wird. Um diese Problematik zu beheben, wurde die Funktion des Remap implementiert. Remap kann auf einem schon gemappten SPS-Projekt durchgeführt werden, wenn an diesen Änderungen/Erweiterungen vorgenommen wurden.

Das SPS-Projekt wird dabei auf Änderungen durchsucht und dabei:

- Neue Objekte hinzugefügt (Um Kollisionen mit alten Projekten zu vermeiden, werden neue ObjectIdentifier größer als die größte je generierte Instanznr. vergeben, siehe "System Manager Settings")
- Geänderte Property-Initialisierungen geschrieben (Unveränderte Initialisierungen werden nicht nochmals geschrieben)
- In der SPS entfernte, zuvor durch ein SPS-Automapping erzeugte Objekte entfernt
- Achtung bei Umbenennungen: Beim Verschieben von Objekten im Baum, kann es zum Verlust von Property-Werten kommen. (z.B. durch Veränderung des StructuredViewPath oder Symbolnamen)

Anhang

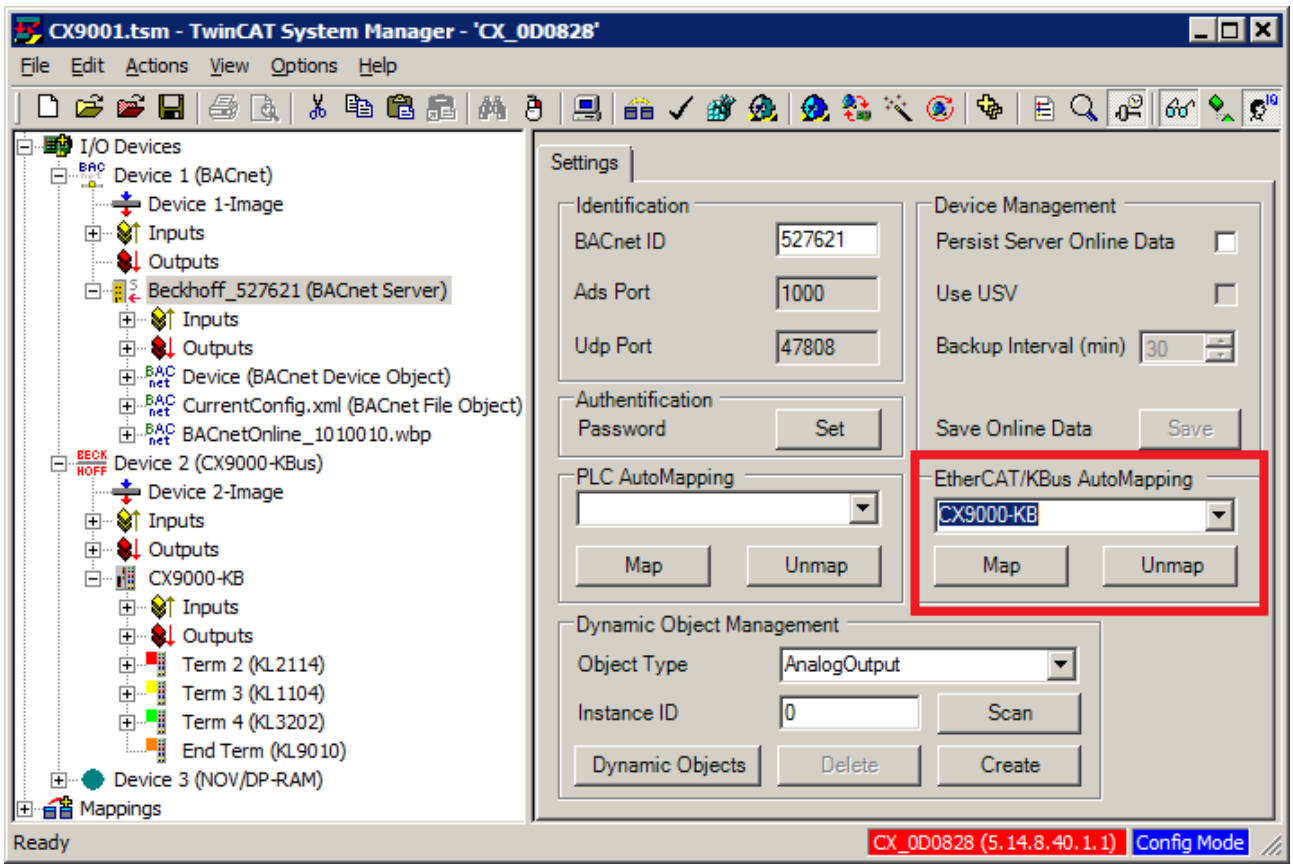
- Übersicht Objekttypen für den "BACnet_ObjectType" : AI,AO,AV,BI,BO, BV, CAL,CMD,DEV,FILE,GROUP,LOOP,MI,MO,NC,PROG,SCHED,MV,TLOG
- Anmerkung zu NOLINK: Der Kommentar "NOLINK" dient zur Markierung von BACnet-Properties die nicht mit einer Variable verknüpft werden sollen, falls die Variable mehrfach annotiert ist. Im Falle ausgewählter BACnet-Properties ist "NOLINK" automatisch gesetzt, da eine Verknüpfung nicht sinnvoll ist. Implizit mit "NOLINK" annotiert sind die Properties: ObjectIdentifier, ObjectType sowie alle Remote*-Properties der Client-spezifischen Konfiguration.
- Nachträglich wurde in TwinCAT BACnet/IP die Möglichkeit eingeführt die Property *ObjectIdentifier* als Prozessdatum zu aktivieren. Da diese Property automatisch als NOLINK behandelt wird, wurde aus Kompatibilitätsgründen die Option LINK eingeführt, welche für *ObjectIdentifier* NOLINK überschreibt. Damit kann auch diese Property via SPS-Automapping als Prozessdatum aktiviert und verknüpft werden.
- Beispiel hierarchische FBs: siehe Beispiel "SPS-Automapping [▶ 122]"

2.10 I/O-Automapping

TwinCAT BACnet/IP unterstützt eine automatisierte Abbildung der modularen IO-Systeme K-Bus und EtherCAT auf BACnet. Dabei wird automatisiert für jeden I/O-Kanal ein entsprechendes BACnet-Objekt der Typen BinaryInput, BinaryOutput, AnalogInput bzw. AnalogOutput angelegt, Prozessdaten entsprechend durch ein Device2Device-Mapping verknüpft und ggf. auch der Status der I/O-Systeme auf die BACnet-Property Reliability und StatusFlags übertragen.

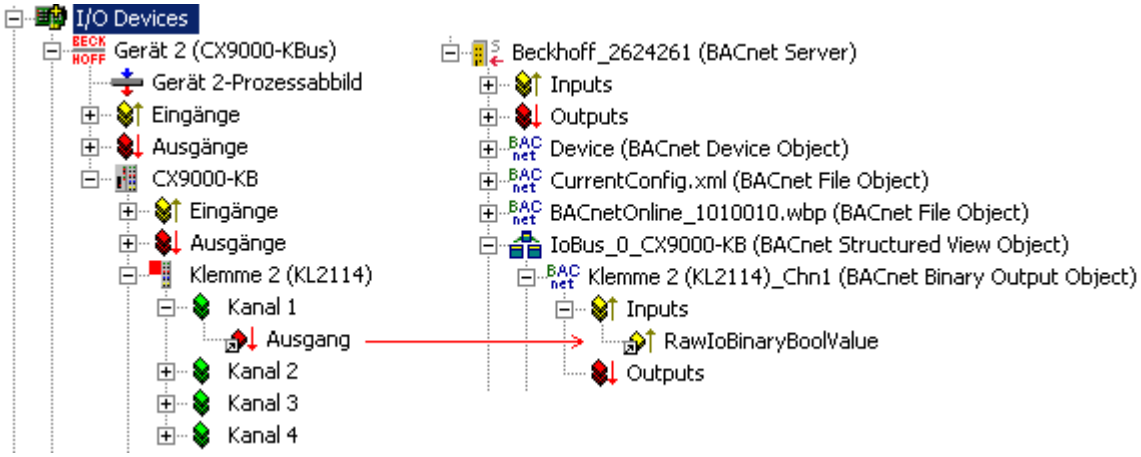
Bei der Abbildung von I/O-Systemen werden I/O-Busse unterschieden, die jeweils eine Anzahl von I/O-Modulen zusammenfassen und jeweils einen I/O-Strang abstrahieren. Über die Zuordnung einer IoBusNr werden BACnet-I/O-Objekten einem I/O-Bus zugeordnet über den u.a. der Status abgebildet wird. Wird z.B. ein K-Bus-Strang verknüpft, kann über das Prozessdatum BusState erkannt werden, ob der K-Bus operationsbereit ist und die Property Reliability aller BACnet-Objekte mit der jeweiligen IoBusNr auf den Wert NO_FAULT_DETECTED - andernfalls auf NO_SENSOR bzw. NO_OUTPUT angepasst werden. Mit Hilfe der I/O-Busse kann mit TwinCAT BACnet/IP auch zusätzlich zu einem K-Bus z.B. ein BK9000 oder weitere EtherCAT-Stränge mit einem BACnet-Controller verknüpft werden.

Ein I/O-Bus-Automapping kann über den Reiter "Settings" eines BACnet-Servers ausgelöst werden. In dem entsprechenden Dialogfeld kann ein I/O-Bus angewählt und via Button "Map" die Verknüpfung veranlasst werden:

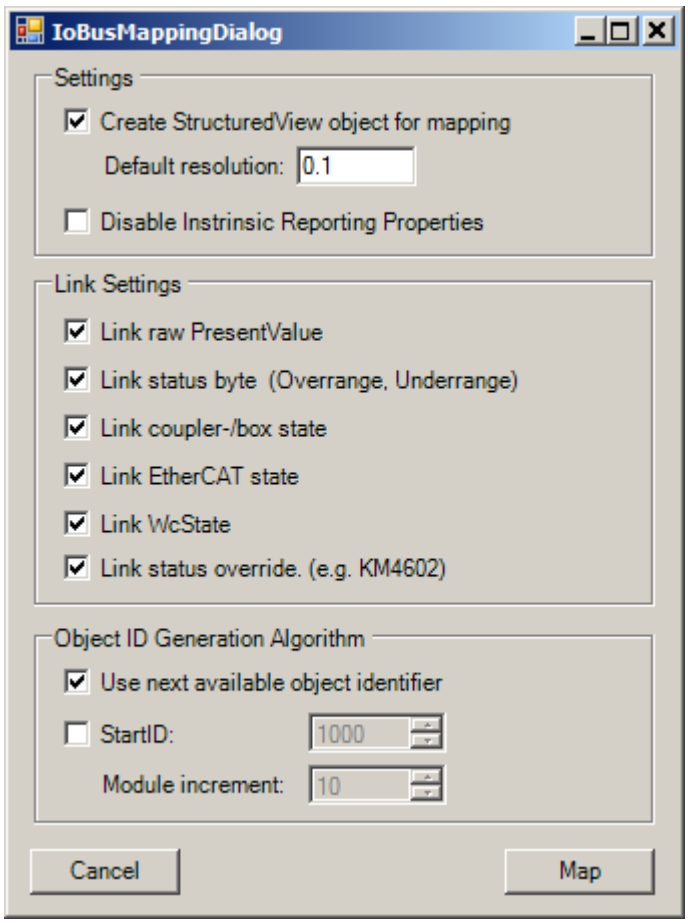


Derzeit werden folgende I/O-Bus-Typen unterstützt:

- EtherCAT
- BK1120, BK1250
- K-BUS: CX1100-BK, CX5000-BK, CX-8000-BK
- BK9000, BK9100, BK9050



Generell werden bei einem I/O-Automapping vom System-Manager alle Klemmen unterhalb eines I/O-Bus analysiert und entsprechende BACnet-Objekte erstellt und Prozessdaten verknüpft. In der Abbildung ist beispielhaft dargestellt wie eine digitale Ausgangsklemme (KL2114) an einem CX9001-KBus auf BACnet abgebildet wird. Es wurde entsprechend das BACnet-Objekt "Klemme 2 (KL2114)_Chn1" vom Typ BinaryOutput erstellt und die Ausgangsprozessdatenvariable "Ausgang" mit der BACnet-Prozessdatenvariable "RawIoBinaryValue" verknüpft. Diese Verknüpfung stellt sicher, dass der Status des digitalen Ausgangssignals immer dem PresentValue des BACnet-Objekts entspricht, sofern das Objekt nicht OutOfService ist. Dabei wird auch entsprechend die Polarität des "BinaryOutputs" betrachtet. Details hierzu können im Kapitel "Prozessdaten [▶ 43]" nachgelesen werden.



Über einen entsprechenden Dialog kann das Automapping konfiguriert und festgelegt werden, ob eine automatische Bus-Status-Überwachung erfolgen soll bzw. ob die Prozessdaten entsprechend automatisch verknüpft werden können. Zusätzlich kann der Object-ID-Vergabe-Algorithmus der erstellten BACnet-Objekte festgelegt werden. Hierbei stehen zur Auswahl:

- **"Use next available object identifier"** - Für jedes neue BACnet-Objekt wird dabei von 0 beginnend, die nächste freie Objekt-ID für den entsprechenden Objekttyp verwendet.
- Module-basierte ID-Vergabe - Ausgehend von einer Start-ID wird die Objekt-ID bei jedem neuen I/O-Module um ein "Module increment" erhöht. Damit kann realisiert werden, dass immer an Hand BACnet-Objekt-ID ermittelt werden kann, welche Klemme verknüpft wurde. Z.B. bei einer Start-ID von 1000 und einem "Module increment" von 10 ist dem 3. binären Eingangskanal der 11. Klemme ein BinaryInput-Objekt mit der ID 1113 zugeordnet wird.

Die weiteren Konfigurationsoptionen des I/O-Automapping werden im Folgenden erläutert:

- **"Create StructuredView object for mapping"** - Alle für ein I/O-Mapping erzeugten BACnet-Objekte werden unterhalb eines neu erzeugten StructuredView-Objekts angelegt, um die Übersicht innerhalb des Projekts zu erhöhen
- **"Default Resolution"** - Analog-Objekte besitzen eine BACnet-Property "Resolution", welche die Skalierung von I/O-Prozessdaten auf das PresentValue des BACnet-Objekts bzw. umgekehrt bestimmt. Der hier konfigurierte Parameter wird bei allen erzeugten Analog-Objekten eingetragen. Sollen Hardware-Werte dabei original übernommen werden, sollte eine "Resolution" von 0 eingestellt werden.
- **"Link raw PresentValue"** - Um das PresentValue der BACnet-Objekte entsprechend mit den I/O-Modulen zu verknüpfen, müssen die entsprechenden RawIoPresentValue-Properties der BACnet-Objekte aktiviert und verknüpft werden. Dies kann mit Hilfe dieser Option aktiviert werden.
- **"Link Status Byte"** - Analoge Eingangsklemmen können über ein Statusbyte anzeigen, ob der jeweilige Prozessdatenwert gültig ist oder ggf. anzeigen, ob z.B. eine Bereichsüberschreitung oder ein Kabelbruch vorliegt. Ist diese Option aktiviert, wird bei analogen Eingangsklemmen (KX3XXX, EL3XXX) der Datenstatus der Klemme mit der BACnet-I/O-Property "RawIoStatus" verknüpft. Bei EtherCAT werden die Variablen "Underrange" und "Overrange" Bit-basiert verknüpft. Sind die entsprechenden Status-Bits gesetzt wird die Reliability zur Laufzeit wie folgt gebildet:

- Ist Bit 0 gesetzt, ist Reliability UNDER_RANGE
- Sonst ist wenn Bit 1 gesetzt Reliability OVER_RANGE
- Sonst ist wenn Bit 6 gesetzt Reliability UNRELIABLE_OTHER

- **"Link coupler-/box state"** - Beim K-Bus bzw. Buskopplern (BK9000) kann über die Prozessdaten BusState/couplerState und "BoxState" der Kommunikations- und Operationszustand für alle Klemmen des Busses ermittelt werden. Die Überwachung des couplerState ist zentral über den BACnet-Server realisiert. Ist diese Option aktiviert werden vorhandene BusState/couplerState bzw. "BoxState" auf die IoBusState-Prozessdaten des BACnet-Server der jeweiligen IoBusNr verbunden. Der BusState eines K-Bus wird dabei mit couplerState verknüpft. Ist couplerState oder "BoxState" für einen I/O-Bus im BACnet-Server ungleich 0, wird die Reliability aller Objekten unter dem Server mit der entsprechenden IoBusNr auf den Wert NO_SENSOR für Eingangsobjekte bzw. NO_OUTPUT für Ausgangsobjekte gesetzt wenn die Objekte nicht OutOfService sind.
- **"Link EtherCAT state"** - Für EtherCAT steht ein ähnlicher Mechanismus wie Coupler/Box-State beim K-Bus zur Verfügung. Im Unterschied zum K-Bus kann bei EtherCAT der Zustand jeder Klemme ermittelt und so potentiell auch abgezogenen Klemmen erkannt werden. Deshalb wird die Status-Erkennung bei EtherCAT nicht zentral über den Server realisiert, sondern erfolgt in jedem Objekt durch die Verknüpfung der Prozessdaten-Property "RawIoECATState". Ist der Wert von "RawIoECATState" ungleich 8 (OP), wird entsprechend die Reliability des Objekts auf den Wert NO_SENSOR für Eingangsobjekte bzw. NO_OUTPUT für Ausgangsobjekte gesetzt, wenn die Objekte nicht OutOfService sind.
- **"Link Wc State"** - Die Überwachung des Wc-State von EtherCAT wird momentan noch nicht unterstützt.
- **"Link Status Override"** - Für Klemmen mit Handbetriebsmodus kann der jeweilige Handbetriebsstatus in BACnet über das Override-Bit in den StatusFlags abgebildet werden. Für die Klemmen KM4602 und KM2652 wird bei Aktivierung dieser Option eine entsprechende Verknüpfung mit der Prozessdaten-Property "RawIoStateOverride" generiert.

Die automatische Abbildung von EtherCAT- und K-Bus-Teilnehmern auf BACnet-Objekte ist nicht immer möglich. U.a. bei komplexen Klemmen ist in bestimmten Fällen nicht immer klar, ob für ein Prozessdatum ein BACnet-Objekt erstellt bzw. welches BACnet-Objekt erstellt werden soll. Für eine große Menge elementarer Ein-/Ausgabe Klemmen wird die Abbildung nach folgendem Algorithmus ausgeführt:

- Für alle 1er (KX-1XXX, EL-1XXX) Module wird für alle gefundenen Eingangsvariablen vom Typ BIT ein BACnet-BinaryInput-Objekt erstellt und das Prozessdatum verknüpft, wenn der Variablenname ungleich "WcState" und "InputToggle" ist.
- Für alle 2er (KX-2XXX, EL-2XXX) Module wird für alle gefundenen Ausgangsvariablen vom Typ BIT ein BACnet-BinaryOutput-Objekt erstellt und das Prozessdatum verknüpft.
- Für alle 3er (KX-3XXX, EL-3XXX) Module wird für alle gefundenen Eingangsvariablen vom Typ INT16 ein BACnet-AnalogInput-Objekt erstellt und das Prozessdatum verknüpft.
- Für alle 4er (KX-4XXX, EL-4XXX) Module wird für alle gefundenen Ausgangsvariablen vom Typ INT16 ein BACnet-AnalogOutput-Objekt erstellt und das Prozessdatum verknüpft.

Zusätzlich existiert eine Sonderbehandlung für folgende Klemmen:

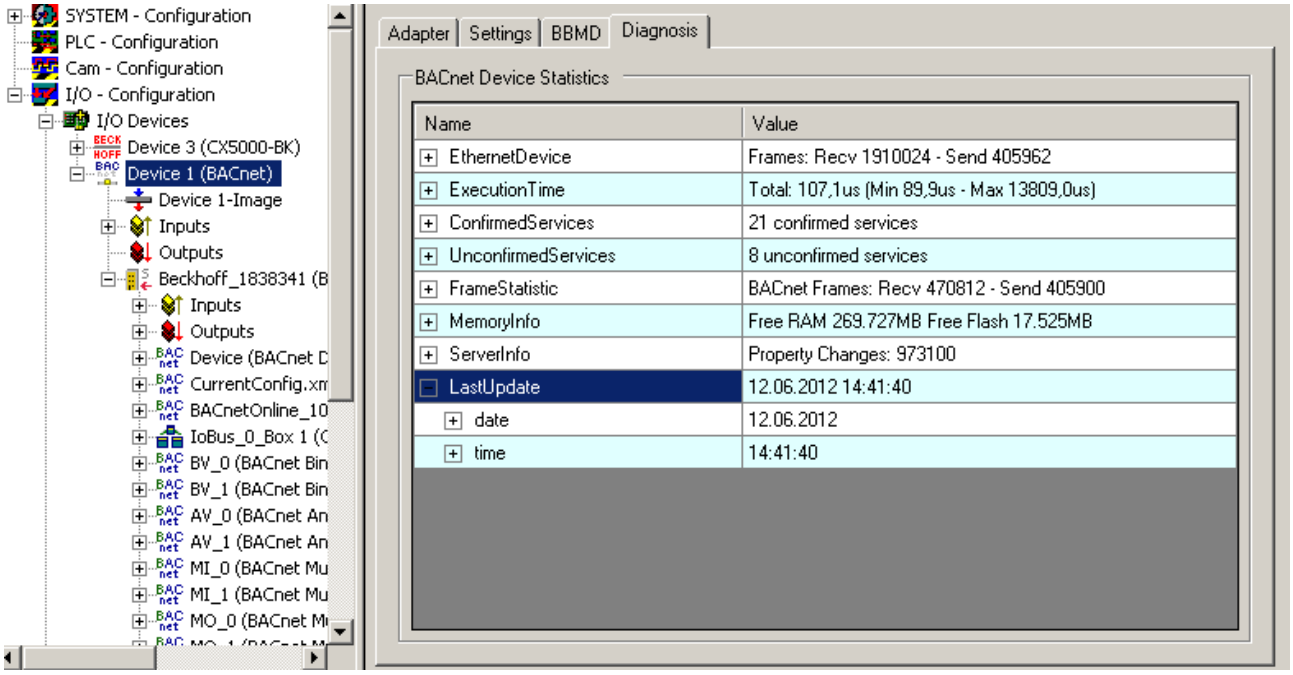
- KL1859 - Es werden jeweils 8 BinaryInput- und 8 BinaryOutput-BACnet-Objekte angelegt.

Bei einem Automapping werden automatisch die folgenden BACnet-Properties konfiguriert:

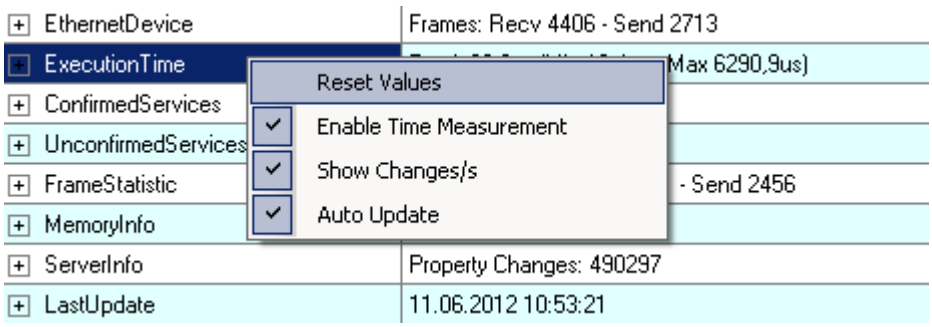
- ObjektIdentifier - wird durch den Objekt-ID-Vergabe-Algorithmus bestimmt
- ObjektName - wird aus dem Klemmennamen plus "_ChnX" für die Kanalnummer gebildet
- IoBusNr - ausgehend von 0; wird mit jedem I/O-Automapping inkrementiert
- IoModuleNr - gibt ausgehend von 1 die Nummer der verknüpften Klemme an
- IoChannelNr - gibt ausgehend von 1 den Kanal an
- Resolution - wie im Konfigurationsdialog festgelegt

2.11 Diagnose

In diesem Abschnitt soll beschrieben werden, wie BACnet-Konfigurationen zur Laufzeit analysiert werden können. Hierbei stehen zum einen als Prozessdaten der Device-Status sowie bei BACnet-Clients der Client-Status zur Verfügung (siehe Abschnitt Prozessdaten [▶ 43]). Zum anderen können im Reiter "Diagnosis" des BACnet-Device weitere Details analysiert werden. Die Diagnose im Reiter "Diagnosis" soll in diesem Abschnitt zusammen mit zahlreichen Problemlösungsstrategien vorstellt werden.



Die Diagnose des BACnet-Device ist in mehrere Kategorien unterteilt. Diese sollen in den folgenden Abschnitten im Detail vorgestellt werden. Über ein Kontextmenü können verschiedene Optionen aktiviert werden:



Mit der Option "Auto Update" kann die Diagnose-Anzeige im 1-Sekunden-Intervall aktualisiert werden. Die Kategorie "Last Update" zeigt immer den Zeitpunkt an, wann die Diagnose zuletzt ausgelesen wurden. Mit der Option "Enable Time Measurement" kann die Kategorie "ExecutionTime" aktiviert werden, über die Laufzeiten des BACnet-Stack bestimmt werden können. Die Option "Show Changes/s" aktiviert bei einigen Werten eine zusätzliche Anzeige der Veränderungen pro Sekunde. Diese wird über die letzten 10 ausgelesenen Diagnose-Informationen gebildet und funktioniert nur bei aktiviertem "Auto Update". Mit der Option "Reset Values" können alle Diagnosewerte zurückgesetzt werden.

Diagnosedaten in der SPS

Auf die in diesem Abschnitt vorgestellten Diagnosedaten kann auch von der SPS aus über eine ADS-Schnittstelle zugegriffen werden. Mit dem Funktionsbaustein FB_BACnet_GetDiagInfo steht eine einfache Zugriffsmöglichkeit bereit.

Kategorie EthernetDevice

Über die Kategorie EthernetDevice können die Anzahl der versendeten und empfangenen Nachrichten (Frames) sowie Empfangs- und Sende-Fehler ermittelt werden. Diese Informationen werden direkt aus dem Netzwerk-Treiber ausgelesen und werden dadurch beim Neuladen einer BACnet-Konfiguration nicht zurückgesetzt. Die Option "Reset Value" löscht aber auch diese Werte. In dieser Kategorie werden alle Frames gezählt, die über den Netzwerk-Treiber verarbeitet werden. Also auch nicht-BACnet-Frames.

⊟ EthernetDevice	Frames: Recv 3727 - Send 2313
⊟ ethStat	(3727;2313)
sendFrames	2313 (0.00 Frames/s)
recvFrames	3727 (0.40 Frames/s)
⊟ txRxErrCnt	(0;0)
txCnt	0 (0.00 /s)
rxCnt	0 (0.00 /s)

Der Zähler "sendFrames" gibt an wie viele Frames verwendet wurden, "recvFrames" die Anzahl empfangener Frames. "txCnt" und "rxCnt" geben die Anzahl fehlerhaft empfangener bzw. gesendeter Frames an. Sind diese Zähler erhöht kann dies z. B. auf Störungen im Netzwerk oder Überlasten (z. B. CRC-Fehler, Dropped-Frames) hindeuten.

Kategorie ExecutionTime

⊟ ExecutionTime	Total: 71,6us (Min 13,4us - Max 5898,7us)
ns100IoInCurExecutionTime	359
ns100IoInMinExecutionTime	78
ns100IoInMaxExecutionTime	53663
ns100IoOutCurExecutionTime	157
ns100IoOutMinExecutionTime	27
ns100IoOutMaxExecutionTime	2841
ns100CycleCurExecutionTime	200
ns100CycleMinExecutionTime	29
ns100CycleMaxExecutionTime	2483
ns100InputCurDistanceTime	8088
ns100InputMinDistanceTime	1912
ns100InputMaxDistanceTime	66939

Mit aktivierter Option "Enable Time Measurement" können Laufzeitinformationen von TwinCAT BACnet/IP ermittelt werden. Es wird jeweils in minimaler (Min), maximaler (Max) und aktueller (Cur)-Wert der Laufzeiten ermittelt. Alle Laufzeiten werden mit 100ns-Auflösung ermittelt. Als Laufzeiten stehen folgende Werte zur Verfügung:

- **IoIn**XXXExecutionTime : Ermittelt die Laufzeit der Funktion InputUpdate des BACnet-Device, die auch die InputUpdate-Funktionen aller BACnet-Objekte ausführt. Hier werden u.a. Prozessdaten von BACnet zur SPS/IO kopiert. Zusätzlich wird im BACnet-Server der IoBusStatus überprüft und in den BACnet-Objekten aktualisiert. Für BACnet-Remote-Objekte werden ReadProperty-Requests und CovSubscriptions für aktivierte Property-Prozessdaten aus diesem Kontext versendet.
- **IoOut**XXXExecutionTime: Ermittelt die Laufzeit der Funktion OutputUpdate des BACnet-Device, die auch die OutputUpdate-Funktionen aller BACnet-Objekte ausführt. In Objekten werden Prozessdaten von der SPS/IO nach BACnet kopiert und auf Änderung verglichen. Bei Änderung können je nach BACnet-Property EventState-Machines ausgeführt werden und potentiell COV-Meldungen und Event-Nachrichten versendet werden. Für BACnet-Remote-Objekte werden Write-Property-Requests für aktivierte Property-Prozessdaten versendet und im Falle von "WriteOnChange" Änderung relevanter Properties erkannt (Vergleich).
- **Cycle**XXXExecutionTime: Ermittelt die Laufzeit der Funktion CycleUpdate des BACnet-Device, die auch die CycleUpdate-Funktionen aller BACnet-Objekte ausführt. Im CycleUpdate-Kontext werden:
  - Device-, Client-, Server-Statemachines ausgeführt,
  - Empfangene Frames verarbeitet,

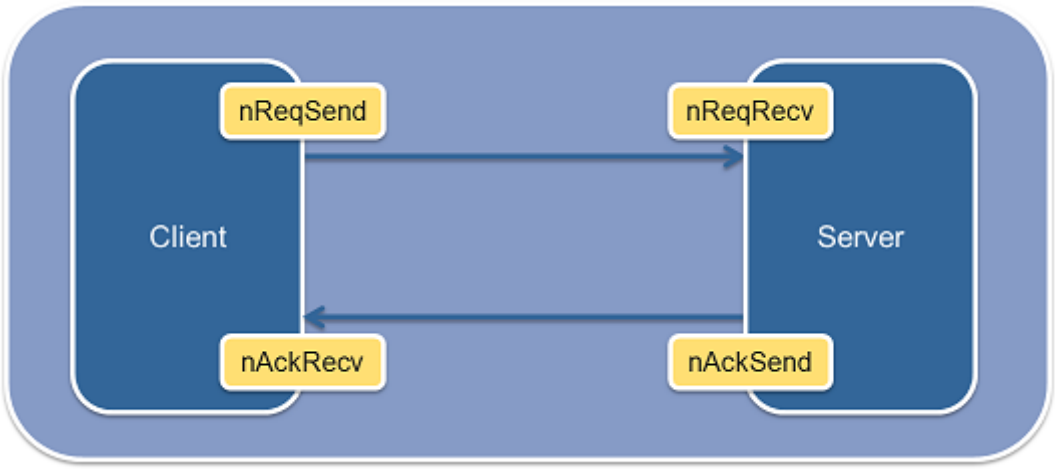
- Potentiell Objekt-EventState-Machines ausgeführt (bei Property-Änderungen),
- COV-Meldungen und Event-Nachrichten versendet,
- ADS-Requests verarbeitet,
- Segmentierte Frames verarbeitet,
- und Persistente Daten geschrieben.

- InputXXX**Distance**Time: Gibt den Abstand zwischen zwei Aufrufen der Funktion InputUpdate des BACnet-Device an. Hiermit kann im Falle von "Exceeds" der BACnet-Task ermittelt werden, in welchem maximalen Abstand der BACnet-Stack ausgeführt wird.

In der oberen Zeile der Kategorie Execution Time wird die Summe der Werte IoIn-, IoOut- und Cycle-XXXExecution-Time jeweils als minimaler, maximaler und aktueller Wert in Mikrosekunden zusammengefasst.

Kategorie ConfirmedServices

In dieser Kategorie kann eine Statistik über versendete und empfangene (bestätigte) BACnet-Nachrichten ermittelt werden. Bestätigte Dienste arbeiten in BACnet nach dem Client-Server-Prinzip. Ein Client sendet einen Request, der Server empfängt diesen Request, verarbeitet ihn und sendet eine Antwort (Acknowledge). Der Client empfängt diese Antwort und kann sie entsprechend weiterverarbeiten. Je nach Diensten kann ein TwinCAT BACnet/IP-Gerät als Client oder Server agieren. Der Ablauf bestätigter Dienste ist in der nachfolgenden Grafik dargestellt.



In der Kategorie ConfirmedServices wird für jeden unterstützen Dienst eine Liste mit der Anzahl versendeter und empfangener Requests sowie versendeter und empfangener Antworten geführt. Dabei wird jeweils aufgeschlüsselt wie viele Requests/Antworten erfolgreich bzw. mit Fehler verarbeitet wurden. Aus diesen Kombinationen ergeben sich die im unteren Screenshot dargestellten 8 Werte je BACnet-Dienst. In der obersten Zeile eines jeden Diensts wird jeweils die Summe von Requests und Antworten die erfolgreich bzw. mit Fehler ausgeführt wurden, dargestellt.

SubscribeCOVProperty	Req 2232/136 - Ack 1602/630
nReqSend	2232 (0.00 Frames/s)
nReqSendFail	136 (0.00 Frames/s)
nReqRecv	0 (0.00 Frames/s)
nReqRecvFail	0 (0.00 Frames/s)
nAckSend	0 (0.00 Frames/s)
nAckSendFail	0 (0.00 Frames/s)
nAckRecv	1602 (0.00 Frames/s)
nAckRecvFail	630 (0.00 Frames/s)

Bei der Klassifikation von Diensten werden in BACnet sogenannte A- und B-Dienste unterschieden, je nachdem welche Rolle (Client bzw. Server) ein Gerät bei einer Dienstnutzung einnimmt. Agiert ein Gerät als Client, stellt also einen Request und empfängt eine Antwort (Ack) nutzt er einen A-Dienst. Empfängt ein

Gerät einen Request, verarbeitet ihn und sendet eine Antwort (Ack) stellt er den B-Dienst bereit. Liest z.B. ein Client den Wert einer Property von einem anderen Gerät, wird der Dienst ReadProperty-A verwendet. Die Gegenstelle, der Server, implementiert in diesem Fall den Dienst ReadProperty-B. Für die Diagnose ist es wichtig zwischen A- und B-Diensten zu unterscheiden. Deshalb wird die nachfolgende Tabelle jeweils in diese Dienst-Kategorien aufgeteilt.

Bei bestätigten Diensten gehören also zu einem A-Dienst immer die Zähler ReqSend und AckRecv; zu einem B-Dienst immer ReqRecv und AckSend. In der Tabelle wird bei den möglichen Fehlern jeweils erläutert, in welchen Fällen die Fehler-Zähler inkrementiert werden (z.B. ReqSendFail>0). Ein wichtiger Fehlerindikator kann aber bei A-Diensten auch eine Abweichung von der Anzahl gesendeter Requests und empfangener Antworten (Ack) sein. Diese Fehlerart wird mit (Req<>Ack) abgekürzt.

Dienstdiagnose

Bei BACnet gilt es zu beachten, dass erhöhte Fehlerzähler in der Diagnose nicht zwangsläufig auf eine Fehlkonfiguration hindeuten. In BACnet-Netzwerken kann es durchaus zu kurzen Lastspitzen und verlorenen Nachrichten kommen. Hierfür existieren entsprechende Wiederholungsmechanismen (ApduRetries und ApduTimeout). Bei hohen Fehlerraten sollten allerdings Probleme von Grund auf behoben werden.

In der folgenden Tabelle werden relevante BACnet-Dienste und ihre Diagnose kurz beschrieben. Dabei wird darauf eingegangen, welche Funktion die Dienste erfüllen, wo die Dienste in TwinCAT BACnet/IP verwendet werden, welche möglichen Fehlerursachen diagnostiziert werden können und wie eine entsprechende Problemlösung aussehen kann.

AcknowledgeAlarm-B (ReqRecv, AckSend)	**Verwendung:** Bestätigung offener Alarme **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (falscher EventState, falscher Timestamp, Instrinsic Reporting für das Objekt ist deaktiviert) nAckSendFail: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Gegenstelle überprüfen
ConfirmedCOVNotification-A (ReqSend, AckRecv)	**Verwendung:** Gegenstelle hat Wertänderung abonniert. Dienst sendet Nachricht bei Änderung. **Mögliche Fehler:** ReqSendFail>0 : kein Framespeicher, kein Routerspeicher, keine verfügbaren Invoke-IDs Req <> Ack: Gegenstelle nicht verfügbar bzw. überlastet **Fehlerbehandlung:** Property COVIncrement bei Analog*-Objekten überprüfen/ erhöhen. BACnet-Zykluszeit verringern: bei schreibenden Prozessdaten, werden dann weniger Änderungen generiert. Konfiguration auf ständig ändernde Properties überprüfen. Prüfen ob Gegenstelle nicht-bestätigte COV-Abonnements verwenden kann.
ConfirmedCOVNotification-B (ReqRecv, AckSend)	**Verwendung:** Empfang einer Wertänderung **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (Prozess-ID nicht vorhanden, unbekanntes Objekt, unbekannte Property, Nachricht enthält nicht den erwarteten Datentyp), kein Routerspeicher **Fehlerbehandlung:** Gegenstelle überprüfen
ConfirmedEventNotification-A (ReqSend, AckRecv)	**Verwendung:**

	Nachricht bei Änderung eines Ereigniszustands (Instrinsic Reporting) **Mögliche Fehler:** ReqSendFail>0: kein Framespeicher, kein Routerspeicher, keine verfügbaren Invoke-IDs Req <> Ack: Gegenstelle nicht verfügbar bzw. überlastet **Fehlerbehandlung:** Überprüfen, ob das Versenden von Ereignisnachrichten für alle Objekte sinnvoll ist. Ggf. Instrinsic Reporting Properties deaktivieren, Ereignis-Benachrichtigung über Property EventEnable deaktivieren, oder allen Objekten, die keine Ereignisnachrichten verschicken sollen eine NotificationClass ohne Recipients zuordnen. Ggf. unbestätigte Ereignisnachrichten verwenden (issueConfirmedNotifications in NotificationClass:RecipientList.)
ConfirmedEventNotification-B (ReqRecv, AckSend) *Nicht Bestandteil der Zertifizierung*	**Verwendung:** Empfang von Ereignismeldungen in der Notification Sink **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (Prozess-ID nicht vorhanden, Empfangspuffer für Ereignisnachrichten voll), kein Routerspeicher **Fehlerbehandlung:** Gegenstelle überprüfen, Puffergröße für EventNotifications erhöhen, Löschintervall für Nachrichtenpuffer erhöhen
GetAlarmSummary-B (ReqRecv, AckSend)	**Verwendung:** Abrufen einer Zusammenfassung aller Objekte im Alarmzustand **Mögliche Fehler:** nReqRecvFail>0: kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher
GetEnrollmentSummary-B	**Verwendung:** Abrufen einer Zusammenfassung ereignisspezifischer Informationen (mit umfangreichen Filteroptionen) **Mögliche Fehler:** nReqRecvFail>0: kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher
SubscribeCOV-A (ReqSend, AckRecv)	**Verwendung:** Dienst zum Abonnieren einer Wertänderung (COV) der Property "PresentValue" und "StatusFlags" bei Analog*-, Binary*- und MultiState*-Objekten. Bei Remote-Objekten für Prozessdaten im COV-Mode. TrendLog für entfernte Objekte mit LogInterval=0. In NotificationSink konfigurierte COV-Abonnements. **Mögliche Fehler:** ReqSendFail>0: Verbindungsdaten können nicht aufgelöst werden (Client-Status nicht Complete), kein Framespeicher, kein Routerspeicher, keine verfügbaren Invoke-IDs Req <> Ack: Gegenstelle nicht verfügbar bzw. überlastet. **Fehlerbehandlung:** Optimierung der Prozessdaten im "PropertyWizzard" aktivieren, Zykluszeiten größer einstellen, weniger Properties abonnieren, Properties die nicht per COV abonniert werden können, zyklisch auslesen (Polling)
SubscribeCOV-B (ReqRecv, AckSend)	**Verwendung:** Empfang eines Abonnements einer Wertänderung (COV) der Property "PresentValue" und "StatusFlags" bei Analog-*, Binary*- und MultiState*-Objekten.

	Mögliche Fehler: nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekanntes Objekt, nicht unterstützte Property, ungültiger Array-Index), Anzahl maximal unterstützter COV-Subscriptions für Objekt erreicht, kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Anzahl unterstützter COV-Subscriptions pro Objekt in Advanced Parameter [▶ 538]n erhöhen (MAX_OBJ_SUBS_COV_ENTRIES)
AtomicReadFile-B (ReqRecv, AckSend)	**Verwendung:** Lesen einer Datei **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekanntes Objekt, nicht unterstütztes Objekt, ungültige FileStart-Position, Zugriff via Record), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher, lokaler Segmentpuffer voll, zu viele Daten für unterstützte Segmentierung der Gegenstelle **Fehlerbehandlung:** Advanced Parameter [▶ 538] MAX_RECV_BAC_SEGM_FRAMES erhöhen
AtomicWriteFile-B (ReqRecv, AckSend)	**Verwendung:** Schreiben einer Datei **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekanntes Objekt, nicht unterstütztes Objekt, ungültige FileStart-Position, Zugriff via Record), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Advanced Parameter [▶ 538] MAX_RECV_BAC_SEGM_FRAMES erhöhen
AddListElement-B (ReqRecv, AckSend)	**Verwendung:** Hinzufügen eines Listenelements **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekanntes Objekt, nicht unterstützte Property, Property ist keine Liste, Property ist schreibgeschützt), Property-Speicher überschritten, kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Property-Listenspeicher erhöhen (LIST_ALLOC_MEM_BYTES). Einige Listen werden in TwinCAT BACnet/IP als Arrays verwaltet (z.B. RecipientList, DateList). Hier MAX_PROPERTY_ARRAYELEMENTS erhöhen.
RemoveListElement-B (ReqRecv, AckSend)	**Verwendung:** Entfernen eines Listenelements **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekanntes Objekt, nicht unterstützte Property, Property ist keine Liste, Property ist schreibgeschützt, Element nicht vorhanden), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher
ServiceCreateObject-B (ReqRecv, AckSend)	**Verwendung:** Erzeugen eines dynamischen Objekts

	Mögliche Fehler: nReqRecvFail>0: Fehlerhafte Request-Parameter (nicht unterstützter Objekttyp, ungültige Property-Parameterdaten, Objektname oder ObjectIdentifier schon vorhanden, maximale Objektanzahl überschritten), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Advanced Parameter [▶ 538] MAX_OBJECTS erhöhen
ServiceDeleteObject-B (ReqRecv, AckSend)	**Verwendung:** Entladen eines dynamischen Objekts **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekanntes Objekt, statisches Objekt - nur dynamische Objekte können gelöscht werden), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher
ReadProperty-A (ReqSend, AckRecv)	**Verwendung:** Auslesen einer Property eines entfernten Gerätes **Mögliche Fehler:** ReqSendFail>0 : Verbindungsdaten können nicht aufgelöst werden (Client-Status nicht Complete), kein Framespeicher, kein Routerspeicher, keine verfügbaren Invoke-IDs Req <> Ack: Gegenstelle nicht verfügbar bzw. überlastet. **Fehlerbehandlung:** Optimierung der Prozessdaten im "Property-Wizzard" aktivieren. Zykluszeiten erhöhen. Weniger Properties per Polling abfragen: SubscribeCOV-Dienst verwenden
ReadProperty-B (ReqRecv, AckSend)	**Verwendung:** Auslesen einer lokalen Property **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekanntes Objekt, unbekannte Property, kein Lesezugriff, ungültiger Array-Index), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher, lokaler Segmentpuffer voll, zu viele Daten für unterstützte Segmentierung der Gegenstelle
ReadPropertyMultiple-B (ReqRecv, AckSend)	**Verwendung:** Auslesen mehrerer lokaler Objekte und Properties mit einem Zugriff **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekannte Objekte, unbekannte Properties, kein Lesezugriff, ungültiger Array-Index), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher, lokaler Segmentpuffer voll, zu viele Daten für unterstützte Segmentierung der Gegenstelle **Fehlerbehandlung:** Bei zu großen Daten, sollte die Gegenstelle ReadProperty verwenden. GGf. Listen- und Array-Properties per Index auslesen. Advanced Parameter [▶ 538] MAX_SEND_BAC_SEGM_FRAMES erhöhen
WriteProperty-A (ReqSend, AckRecv)	**Verwendung:** Schreiben einer Property eines entfernten Gerätes

	Mögliche Fehler: ReqSendFail>0: Verbindungsdaten können nicht aufgelöst werden (Client-Status nicht Complete), kein Framespeicher, kein Routerspeicher, keine verfügbaren Invoke-IDs Req <> Ack: Gegenstelle nicht verfügbar bzw. überlastet. **Fehlerbehandlung:** Optimierung der Prozessdaten im "PropertyWizzard" aktivieren. Zykluszeiten erhöhen. Weniger Properties abonnieren. WriteOnChange in der Prozessdaten Konfiguration aktivieren (Bei WriteOnChange wird maximal WriteProperty pro Client-Property "Cycle Time" ausgeführt.)
WriteProperty-B (ReqRecv, AckSend)	**Verwendung:** Schreiben einer lokalen Property **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekanntes Objekt, unbekannte Property, kein Schreibzugriff, ungültiger Array-Index, falscher Datentyp), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Property-Listenspeicher erhöhen (LIST_ALLOC_MEM_BYTES). Advanced Parameter [▶ 538] MAX_PROPERTY_ARRAYELEMENTS erhöhen. GGf. Schreibschutz entfernen ("Optional Properties"-Reiter). Gegenstelle untersuchen!
WritePropertyMultiple-B (ReqRecv, AckSend)	**Verwendung:** Schreiben mehrerer lokaler Objekte und Properties mit einem Zugriff **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekannte Objekte, unbekannte Properties, kein Schreibzugriff, ungültiger Array-Index, falscher Datentyp), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Property-Listenspeicher erhöhen (LIST_ALLOC_MEM_BYTES). Advanced Parameter [▶ 538] MAX_PROPERTY_ARRAYELEMENTS erhöhen. GGf. Schreibschutz entfernen ("Optional Properties"-Reiter). WriteProperty verwenden. Gegenstelle untersuchen!
DeviceCommunicationControl-B (ReqRecv, AckSend)	**Verwendung:** Kommunikationssteuerung eines Gerätes (Abschalten der Nachrichtenverarbeitung) **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (falsches Passwort), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Passwort-Konfiguration in Gegenstelle mit Passwort abgleichen (BACnet Server- "Authentification"). Encoding des Passworts prüfen.
ReinitializeDevice-B (ReqRecv, AckSend)	**Verwendung:** Initiiert einen Geräteneustart oder ändert den Gerätestatus in den Backup- oder Restore-Modus. **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (falsches Passwort, Gerät befindet sich nicht im Zustand "Operational",

	Gerät kann keinen Neustart ausführen da es sich im Zustand "Backup" oder "Restore" befindet), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Passwort-Konfiguration in Gegenstelle mit Passwort abgleichen (BACnet Server- "Authentification"). Encoding des Passworts prüfen. Einige Funktionen von ReinitializeDevice-B (Backup, Restore, Restart) funktionieren nur im RUN-Modus. Prüfen ob genügend Dateispeicher für Backup/Restore zur Verfügung steht.
ReadRange-B (ReqRecv, AckSend)	**Verwendung:** Auslesen von Listen (speziell die Property LogBuffer) **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekanntes Objekt, unbekannte Property, Property ist keine Liste, Zugriff mit dem Parameter "TimeRange" und "BySequence" auf nicht unterstützte Properties), kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher
SubscribeCOVP-A (ReqSend, AckRecv)	**Verwendung:** Dienst zum Abonnieren einer Wertänderung (COV-P) mit beliebigen Objekten und Properties. Bei Remote-Objekten für Prozessdaten im COV-Mode. Bei TrendLog für entfernte Objekte mit LogInterval=0. In NotificationSink konfigurierte COV-Abonnements. **Mögliche Fehler:** ReqSendFail>0 : Verbindungsdaten können nicht aufgelöst werden (Client-Status nicht Complete), kein Framespeicher, kein Routerspeicher, keine verfügbaren Invoke-IDs Req <> Ack: Gegenstelle nicht verfügbar bzw. überlastet. **Fehlerbehandlung:** Optimierung der Prozessdaten im "PropertyWizzard" aktivieren. Zykluszeiten erhöhen. Weniger Properties abonnieren. Properties die nicht per COV-P abonniert werden können, zyklisch auslesen (Polling).
SubscribeCOVP-B (ReqRecv, AckSend)	**Verwendung:** Empfang eines Abonnements einer Wertänderung (COV-P) mit beliebigen Objekten und Properties. **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (unbekanntes Objekt, nicht unterstützte Property, ungültiger Array-Index), Anzahl maximal unterstützter COV-Subscriptions für Objekt erreicht, kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Anzahl unterstützter COV-Subscriptions pro Objekt im Advanced Parameter [▶ 538] Menü erhöhen (MAX_OBJ_SUBS_COV_ENTRIES).
GetEventInformation-B (ReqRecv, AckSend)	**Verwendung:** Abrufen einer Zusammenfassung aller Objekte im Eventzustand **Mögliche Fehler:** nReqRecvFail>0: kein Routerspeicher nAckSendFail: kein Framespeicher, kein Routerspeicher

Kategorie UnconfirmedServices

Im Gegensatz zu den bestätigten Diensten arbeiten unbestätigte Dienste nur in eine Richtung. Ein Client sendet also einen Request und ein Server empfängt den Request. In der folgenden Tabelle sind die unbestätigten Dienste der BACnet-Diagnose im Detail beschrieben.

IAm-A (ReqSend)	**Verwendung:** Bekanntmachung des Servers im Netzwerk **Mögliche Fehler:** ReqSendFail>0: kein Framespeicher, kein Routerspeicher
IAm-B (ReqRecv)	**Verwendung:** Bekanntmachung eines Gerätes aus dem Netzwerk **Mögliche Fehler:** nReqRecvFail>0: DeviceAddressBinding Liste ist voll, kein Routerspeicher **Fehlerbehandlung:** Anzahl der unterstützen AddressBindings in den Advanced Parametern [▶ 538] erhöhen (MAX_DEVICE_BINDINGS)
IHave-A (ReqSend)	**Verwendung:** Meldet ein im Netzwerk gesuchtes Objekt **Mögliche Fehler:** ReqSendFail>0: kein Framespeicher, kein Routerspeicher
UnconfirmedCOVNotification-A (ReqSend)	**Verwendung:** Gegenstelle hat Wertänderung abonniert. Dienst sendet unbestätigte Nachricht bei Änderung. **Mögliche Fehler:** ReqSendFail>0: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Property COVIncrement bei Analog*-Objekten überprüfen/erhöhen. BACnet-Zykluszeit verringern: bei schreibenden Prozessdaten, werden dann weniger Änderungen generiert. Konfiguration auf ständig ändernde Properties überprüfen.
UnconfirmedCOVNotification-B (ReqRecv)	**Verwendung:** Unbestätigter Empfang einer Wertänderung **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (Prozess-ID nicht vorhanden, unbekanntes Objekt, unbekannte Property, Nachricht enthält nicht den erwarteten Datentyp), kein Routerspeicher
UnconfirmedEventNotification-A (ReqSend)	**Verwendung:** Unbestätigte Nachricht bei Änderung eines Ereigniszustands (Instrinsic Reporting) **Mögliche Fehler:** ReqSendFail>0: kein Framespeicher, kein Routerspeicher **Fehlerbehandlung:** Überprüfen, ob das Versenden von Ereignisnachrichten für alle Objekte sinnvoll ist. Ggf. Instrinsic Reporting Properties deaktivieren, Ereignis-Benachrichtigung über Property EventEnable deaktivieren, oder allen Objekten, die keine Ereignisnachrichten verschicken sollen eine NotificationClass ohne Recipients zuordnen.
UnconfirmedEventNotification-B (ReqRecv)	**Verwendung:** Unbestätigter Empfang von Ereignismeldungen in der Notification Sink **Mögliche Fehler:** nReqRecvFail>0: Fehlerhafte Request-Parameter (Prozess-ID nicht vorhanden, Empfangspuffer für Ereignisnachrichten voll), kein Routerspeicher
TimeSynchronisation-B (ReqRecv)	**Verwendung:** Empfang eines Zeitstempels zur Zeitsynchronisation

	Mögliche Fehler: nReqRecvFail>0: Systemzeit konnte nicht synchronisiert werden da bereits ein Synchronisation läuft oder TwinCAT sich im Free-Run State befindet, kein Routerspeicher **Fehlerbehandlung:** TwinCAT in den State "Run" schalten
WhoHas-B (ReqRecv)	**Verwendung:** Suche nach bestimmten Objekten in einem Netzwerk **Mögliche Fehler:** nReqRecvFail>0: Gerät verarbeitet keine Nachrichten da dies über den Service "DeviceCommunicationControl" abgeschaltet wurde, kein Server angelegt, kein Routerspeicher
WhoIs-A (ReqSend)	**Verwendung:** Scannen eines Netzwerkes nach Teilnehmer und Abfragen der Gerätekommunikationsparameter **Mögliche Fehler:** ReqSendFail>0: kein Framespeicher, kein Routerspeicher
WhoIs-B (ReqRecv)	**Verwendung:** Abfrage des Servers nach Gerätekommunikationsparametern **Mögliche Fehler:** nReqRecvFail>0: Gerät verarbeitet keine Nachrichten da dies über den Service "DeviceCommunicationControl" abgeschaltet wurde, kein Server angelegt, kein Routerspeicher
UTCTimeSynchronisation-A (ReqSend)	**Verwendung:** Senden eines UTC-Zeitstempels zur Zeitsynchronisation **Mögliche Fehler:** ReqSendFail>0: kein Framespeicher, kein Routerspeicher
UTCTimeSynchronisation-B (ReqRecv)	**Verwendung:** Empfang eines UTC-Zeitstempels zur Zeitsynchronisation **Mögliche Fehler:** nReqRecvFail>0: Systemzeit konnte nicht synchronisiert werden da bereits ein Synchronisation läuft oder TwinCAT sich im Free-Run State befindet, kein Routerspeicher **Fehlerbehandlung:** TwinCAT in den State "Run" schalten

Kategorie FrameStatistic

Diese Kategorie fasst Zähler von versendeten und empfangenen BACnet-Nachrichten zusammen. Zusätzlich kann festgestellt werden, wie oft während des Betriebs der physikalische Link des Netzwerkadapters verloren ging.

FrameStatistic	BACnet Frames: Recv 3704 - Send 3024
nSegmSendTimeouts	0
nFrameAllocFailCnt	143 (0.00 Frames/s)
nFrameSendCnt	3024 (2.40 Frames/s)
nFrameSendFailCnt	143 (0.00 Frames/s)
nFrameRecvCnt	3704 (2.40 Frames/s)
nFrameRecvFailCnt	0 (0.00 Frames/s)
nLinkStatusChangedCnt	1

Die einzelnen Zähler werden im Folgenden näher beschrieben und ggf. mögliche Problemindikatoren aufgezählt.

nSegmSendTimeouts	**Zweck:**

	Gibt an, wie oft es zu Zeitüberschreitungen beim Warten auf die Bestätigung eines Segments kam. **Problemindikator:** Ggf. ist die Gegenstelle überlastet oder nicht mehr erreichbar.
nFrameAllocFailCnt	**Zweck:** Gibt an, wie oft kein Speicher zum Versenden von BACnet-Nachrichten allokiert werden konnte. **Problemindikator:** Es werden potenziell zu viele Nachrichten pro Zyklus versendet. Ggf. können die Netzwerk-Adapter-Queues aktiviert werden (siehe Advanced Parameter [▶ 538]). Es sollte überprüft werden, ob eine Verbindung zum Netzwerk besteht. Ist dieser Zähler erhöht, deutet dies typischerweise auf eine lokale Überlastsituation hin.
nFrameSendCnt	**Zweck:** Gibt an, wie viele BACnet-Nachrichten erfolgreich versendet wurden.
nFrameSendFailCnt	**Zweck:** Gibt an, wie viele BACnet-Nachrichten nicht versendet werden konnten. **Problemindikator:** Es werden potentiell zu viele Nachrichten pro Zyklus versendet. Ggf. können die Netzwerk-Adapter-Queues aktiviert werden (siehe Advanced Parameter [▶ 538]). Es sollte überprüft werden, ob eine Verbindung zum Netzwerk besteht. Ist dieser Zähler erhöht, deutet dies typischerweise auf eine lokale Überlastsituation hin.
nFrameRecvCnt	**Zweck:** Gibt an wie viele BACnet-Nachrichten ohne Fehler empfangen wurden.
nFrameRecvFailCnt	**Zweck:** Gibt an, wie viele BACnet-Nachrichten mit Fehler empfangen wurden. Dieser Zähler ist die Summe der AckRecvFail- und ReqRecvFail-Zähler aller Dienste. **Problemindikator:** Eine genaue Problemanalyse sollte über die Kategorien ConfirmedServices und UnconfirmedServices erfolgen.
nLinkStatusChangedCnt	**Zweck:** Gibt an, wie oft der LinkStatus des Netzwerk-Adapters gewechselt hat. **Problemindikator:** Im Normalfall hat dieser Zähler den Wert 1 (Link bestand beim Systemstart.)

Kategorie MemoryInfo

Die Kategorie MemoryInfo fasst speicherspezifische Informationen zusammen. In der obersten Zeile werden bei WindowsCe-Geräten der verfügbare Haupt- und Flashspeicher angezeigt. Die einzelnen Zähler werden in der folgenden Tabelle erläutert.

nAmsAlloc **nAmsAllocCntMax** **nAmsAllocFailCnt**	**Zweck:** Diese Zähler repräsentieren das Allokationsverhalten des BACnet-Stack beim Router. nAmsAlloc gibt die aktuelle Anzahl an Allokation an. nAmsAllocCntMax gibt an, wie viele Allokationen maximal zu einem Zeitpunkt aktiv waren. nAmsAllocFailCnt gibt an, wie oft kein Speicher alloziert werden konnte. Diese Zähler treffen keine Aussage über die Menge an alloziertem Speicher. **Problemindikator:** Router-Speicher wird für temporäre Operationen verwendet. Z.B. für die Verarbeitung von Nachrichten und zum

	Abspeichern von EventNofitications und COV-Notifications in der NotificationSink. Ist der Zähler nAmsAlloc dauerhaft hoch, deutet dies auf eine Überlastsituation hin.
nCeAvailPhysMemBytes **nCeAvailFlashMemBytes**	**Zweck:** Diese Zähler sind nur für Windows CE implementiert. Sie geben an wie viel Hauptspeicher und Flash-Speicher zur Verfügung steht. **Problemindikator:** Steht nicht genügend Hauptspeicher (nCeAvailPhysMemBytes) zur Verfügung können ggf. keine dynamischen Objekte allokiert werden. Flash-Speicher ist für das Abspeichern persistenter Daten, sowie bei Backup und Restore von Bedeutung.
nRecvFrameFifoSize **nEmptyFrameFifoSize**	**Zweck:** Bei der Verwendung der Netzwerk-Adapter-Queues geben diese Zähler den verfügbaren Listenspeicher für empfangene (nRecvFrameFifoSize) und zu sendende Nachrichten (nEmptyFrameFifoSize) an. **Problemindikator:** Laufen die Speicher leer, kann über den Parameter Dialog der Netzwerk-Adapter-Queues die Speichergröße erhöht werden.
nRecvSegmUDPFramesListSize **nRecvSegmFramesListSize** **nSendSegmFramesListSize**	**Zweck:** Diese Zähler geben Auskunft über die aktuell zwischengespeicherten Nachrichten-Segmente. Dabei wird zwischen IP-Segmentierung (UDP) und BACnet-NPDU-Segmentierung sowie zwischen Segmenten die zum Versand vorgemerkt wird (Send) und schon empfangenen Segmenten (Recv) unterschieden. **Problemindikator:** Treten Probleme im Zusammenhang mit der Segmentierung auf, können ggf. die Advanced Parameter [▶ 538] MAX_RECV_BAC_SEGM_FRAMES und MAX_SEND_BAC_SEGM_FRAMES angepasst werden.
nClientScanListSize	**Zweck:** Gibt die Größe der Liste gefundener Geräte im Netzwerk an.
nAdsRequestListSize	**Zweck:** ADS-Requests an den BACnet-Stack werden in einer Liste zwischengespeichert und im Kontext der BACnet-Task verarbeitet. Dieser Zähler gibt die aktuelle Anzahl ausstehender ADS-Requests an.

Kategorie ServerInfo

Die Kategorie ServerInfo wird im Detail im Abschnitt Persistente Daten [▶ 60] vorgestellt.

Kategorie ClientInfo

In dieser Kategorie werden Informationen zum aktuellen Kommunikationsstatus mit jedem konfigurierten BACnet-Client angezeigt.

instanceNumber	**Zweck:** Enthält als Zuordnungsmerkmal die Geräte-ID (Instanznummer des Device-Objekt) des BACnet-Client.
ipAddr udpPort networkNumber	**Zweck:**

	Enthält Informationen zu Kommunikationsadressen. BACnet verwendet allein die Geräte-ID zur Identifikation von Geräten. Die IP-Adresse kann dynamisch zugeordnet sein. Über dieses Diagnose-Feld kann die aktuelle IP-Adresse ermittelt werden. Zusätzlich können hier der verwendetet UDP-Port sowie die Netzwerknummer (bei Kommunikation über BACnet-Router - z.B. MS/TP-Geräte) abgelesen werden.
isReachable clientState	**Zweck:** ClientSTMState spiegelt den aktuellen Zustand der Client-Statemachine wieder. ClientSTMState kann folgende Werte annehmen: Init, Connecting, WaitingForConnection, RequestDeviceInformation, Complete. Die genaue Arbeitsweise der ClientState-Machine ist in der Sektion Client-Control beschrieben. clientState gibt an, ob eine Verbindung zum entfernten BACnet-Gerät hergestellt werden konnte. clientState ist TRUE wenn die ClientSTMState den Wert "Complete" erreicht hat. **Problemindikator:** Wenn ein BACnet-Gerät nicht erreicht werden kann, verbleibt die ClientState-Maschine im Zustand "*WaitingForConnection*". Dann sollte geprüft werden, warum der BACnet-Client nicht erreichbar ist: • Stimmen die Subnetzmasken aller Geräte überein (z.B. 255.255.255.0). Dies ist eine wichtige Voraussetzung in BACnet, da die bei der Gerätesuche verwendeten Broadcast-Adressen daraus berechnet werden. • Ist ein Gateway konfiguriert; falls sich die Geräte in unterschiedlichen IP-Bereichen befinden. Ist das Gateway erreichbar?!! • Ist der Einsatz eines BBMD erforderlich (weil sich die Geräte in unterschiedlichen Netzen (LAN) befinden) • Wurde eine physikalische Verbindung hergestellt. (LinkStatus im Device Status des BACnet-Adapter; Stecken alle Kabel)
nOpenRequests **nFreeOpenRequestData** **nRequestRetries** **nRequestTimeOuts**	**Zweck:** Diese Gruppe von Diagnosewerten zeigt allgemeine Informationen zum Kommunikationsstatus konfigurierter Prozessdaten des BACnet-Client. nOpenRequests gibt an wie viele abgesendete, bis jetzt unbeantwortete Anfragen für diesen Client existieren. Dieser Wert spiegelt letztendlich die Anzahl verwendeter invokeIDs wieder. nFreeOpenRequestData gibt an, wie viele verbleibende invokeIds für den Client noch zur Verfügung stehen. nRequestRetries gibt an wie oft bei den prozessdatenspezifischen Requests (ReadProperty, WriteProperty, ReadPropertyMultiple, SubscribeCOV) Wiederholungen gesendet werden mussten. nRequestTimeOuts gibt an wie viele prozessdatenspezifischen Requests trotz Wiederholungen fehlgeschlagen sind. Die Zeitabstände zwischen Request-Wiederholungen wird aus der Property APDU-Timeout des Server-Device-Objekts abgeleitet. **Problemindikator:** nRequestTimeOuts sollte niemals einen Wert größer 0 annehmen. Dies deutet auf problematische Kommunikationsstörungen hin. Seltene Wiederholungen (nRequestRetries) können dagegen akzeptabel sein. Problemlösungen können in den folgenden 3 Tabellenreihen abgeleitet werden, die Probleme der einzelnen Kommunikationsdienste detaillierter aufschlüsseln.
nCyclicCovRequests **nCyclicCovPendingCounter** **nFailedCovRequests**	**Zweck:** Diese Werte geben Auskunft über COV spezifische Kommunikationsdienste. Diese Werte beziehen sich auf das Registrieren von COV-Meldungen (SubscribeCOV). nCyclicCovRequests gibt an wie viele Property-Prozessdaten mit dem Modus COV (Confirmed, Unconfirmed, Unsolicited) konfiguriert wurden. nCyclicCovPendingCounter gibt an wie viele offene SubscribeCOV-Requests für diesen BACnet-Client existieren. Die maximale Anzahl paralleler (offener) Requests pro Client kann über den Advanced Parameter MAX_PARALLEL_COV_SUBSCRIPTIONS festgelegt werden. nFailedCovRequests gibt an wie viele Registrierungen von COV-Meldungen fehlgeschlagen sind.

	Problemindikator: nFailedCovRequests sollte niemals einen Wert ungleich 0 aufweisen. Wenn eine große Anzahl Properties von einem entfernten Gerät abonniert werden soll, kann in manchen Fällen ReadPropertyMultiple eingesetzt werden. Es ggf. geprüft werden, ob die Gegenstelle COV unterstützt bzw. die abonnierten Properties per COV bzw. COV-P unterstützt werden.
nCyclicReadPollingRequests **nFailedReadRequests**	**Zweck:** Diese Werte geben Auskunft über polling-basierte Prozessdaten (ReadProperty, RPM-O, RPM-C). nCyclicReadPollingRequests gibt an wie viele Nachrichten für diese Art der Prozessdaten konfiguriert wurden. Achtung: dieser Wert enthält nicht die Anzahl konfigurierter Properties, sondern die Anzahl der benötigten Nachrichten um die Daten zu übertragen. Bei RPM-O und RPM-C werden mehrere Property-Werte mit einer Nachricht ausgelesen. nFailedReadRequests enthält die Anzahl fehlgeschlagener Leseversuche. **Problemindikator:** nFailedReadRequests sollte möglichst nie einen Wert größer 0 annehmen. Im Fehlerfall sollte überprüft werden, ob die Gegenstelle ggf. RPM unterstützt. Bei Überlasten sollte die Anzahl von Prozessdaten-Properties reduziert bzw. das Polling-Interval erhöht werden.
nCyclicWriteRequestsn **FailedWriteRequests**	**Zweck:** Diese Werte beziehen sich auf schreibende Prozessdaten. nCyclicWriteRequests gibt an wie viele schreibende Properties unter diesem BACnet-Client konfiguriert wurden. nFailedWriteRequests gibt an wie viele Schreibrequests (WriteProperty) trotz Wiederholungen fehlgeschlagen sind. **Problemindikator:** nFailedWriteRequests sollte niemals einen Wert ungleich 0 besitzen. Bei Überlasten sollte die "Cycle Time" erhöht werden. Diese bestimmt wie oft Änderungen geschrieben werden können. Ggf. sollte geprüft werden, ob nicht vorhandene oder schreibgeschützte BACnet-Properties beschrieben werden.

2.12 BACnet Broadcast Management Device

TwinCAT BACnet/IP unterstützt die "BACnet Broadcast Management Device" (BBMD) Funktionalität. Mit Hilfe eines BBMD können IP-Broadcast-Nachrichten (z.B. 192.168.1.255), die von bestimmten BACnet-Diensten verwendet werden, über die Grenzen lokaler Netzwerke übertragen werden. Ein Beispiel ist der *Who-Is*-Dienst bei dem über eine IP-Broadcast-Nachricht andere BACnet-Geräte gefunden werden. Sendet ein BACnet-Gerät eine *Who-Is* Nachricht, wird diese mit einer Broadcast-MAC-Adresse (FF:FF:FF:FF:FF:FF) ins lokale Netzwerk gesendet. Alle BACnet-Geräte in diesem Netzwerk empfangen die Nachricht und können entsprechend antworteten. IP-Router, die in andere IP-Subnetze vermitteln, leiten diese Nachrichten jedoch nicht weiter. Um diese Problematik zu beheben, wurde BBMD eingeführt. In einem lokalen Netzwerk kann genau ein Gerät mit BBMD-Funktionalität konfiguriert werden, der IP-Broadcast-Nachrichten anhand einer *Broadcast Distribution Table* (BDT) in entfernte IP-Subnetze weiterleitet.

Für die Weiterleitung von BBMD-Nachrichten existieren zwei Verfahren, die als *One-Hop* und *Two-Hop* bezeichnet werden. Beim *Two-Hop*-Verfahren leitet ein BBMD-Geräte Broadcast-Nachrichten an einen weiteren BBMD im entfernten IP-Subnetz weiter. Dieser verteilt die Nachrichten wiederum in seinem lokalen Netzwerk als IP-Broadcast-Nachrichten seines IP-Subnetzes. Beim *One-Hop*-Verfahren leitet der lokale BBMD die Nachricht direkt an die entfernten Geräte weiter. Hierfür müssen allerdings die vermittelnden IP-Router die Weiterleitung von IP-Broadcast-Nachrichten in entfernte IP-Subnetze unterstützen.

Gateway Konfiguration

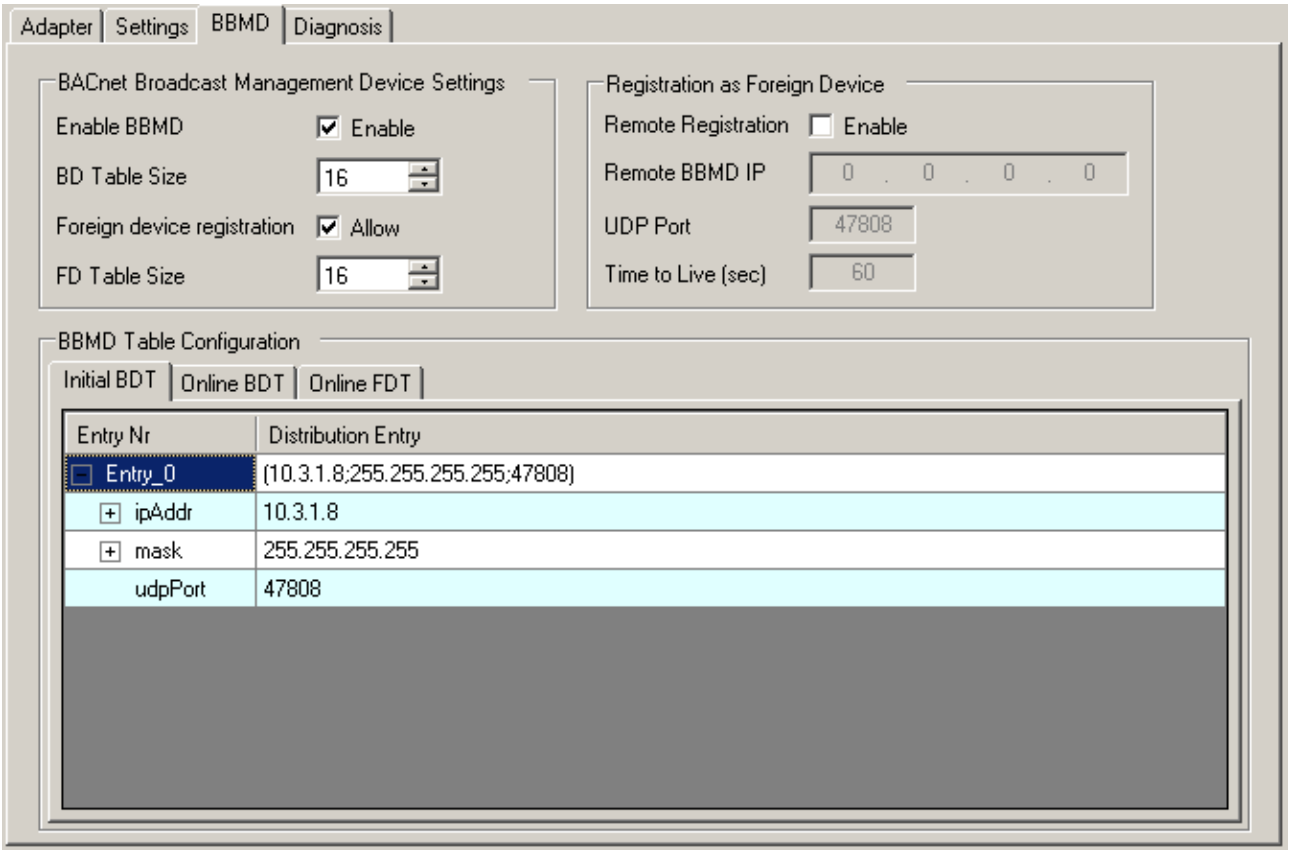
Für die BBMD Funktionalität muss ein Gateway in der IP-Konfiguration des BACnet-Device eingetragen werden, da IP-Nachrichten zwangläufig in entfernte IP-Subnetze übertragen werden müssen. Ist kein Gateway konfiguriert, wird die BBMD Funktionalität automatisch deaktiviert.

Beim "Foreign Device" Verfahren können sich entfernte Geräte, die sich z.B. über eine Internetverbindung einwählen, dynamisch in eine BACnet-Konfiguration einklinken. Hierfür existieren spezielle BACnet-Dienste, die auch von TwinCAT BACnet/IP unterstützt werden, um einen Eintrag in der *Foreign Device Table* (FDT)

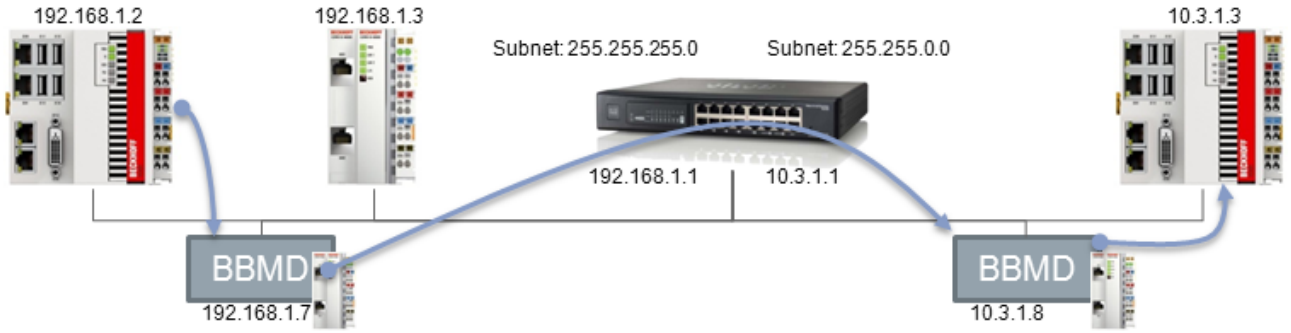
zu erzeugen. Wie auch bei der BDT leitet ein BBMD IP-Broadcast-Nachrichten an alle Einträge der FDT weiter. Im Gegensatz zur BDT haben Einträge in der FDT eine begrenzte Lebenszeit. Die Anmeldung muss also nach Ablauf einer Zeitspanne (*Time To Live*) erneuert werden.

Die Konfiguration eines BBMD erfolgt im Reiter "BBMD" des BACnet-Device. Jedes TwinCAT BACnet/IP Geräte kann als BBMD betrieben werden. Es darf aber immer nur ein BBMD pro Subnetz existieren. Mit der Option "Enable BBMD" kann die BBMD Funktionalität aktiviert werden. Zusätzlich kann die Größe der BDT und FDT konfiguriert werden. Während der Laufzeit können neue Einträge in der BDT und FDT hinzugefügt werden. Bei der Größenkonfiguration ist dies zu beachten. Einträge, die in die BDT zur Laufzeit ergänzt werden, werden als persistente Daten abgespeichert wenn diese Option aktiviert ist. Wie auch bei BACnet-Properties wird bei Einträgen der BDT zwischen Offline und Online-Daten unterschieden. Die "Initial BDT" beschreibt Einträge die beim Starten übernommen werden. Über den Reiter "Online BDT" kann der Zustand der BDT zur Laufzeit betrachtet werden. Die FDT kann im Reiter "Online FDT" eingesehen werden.

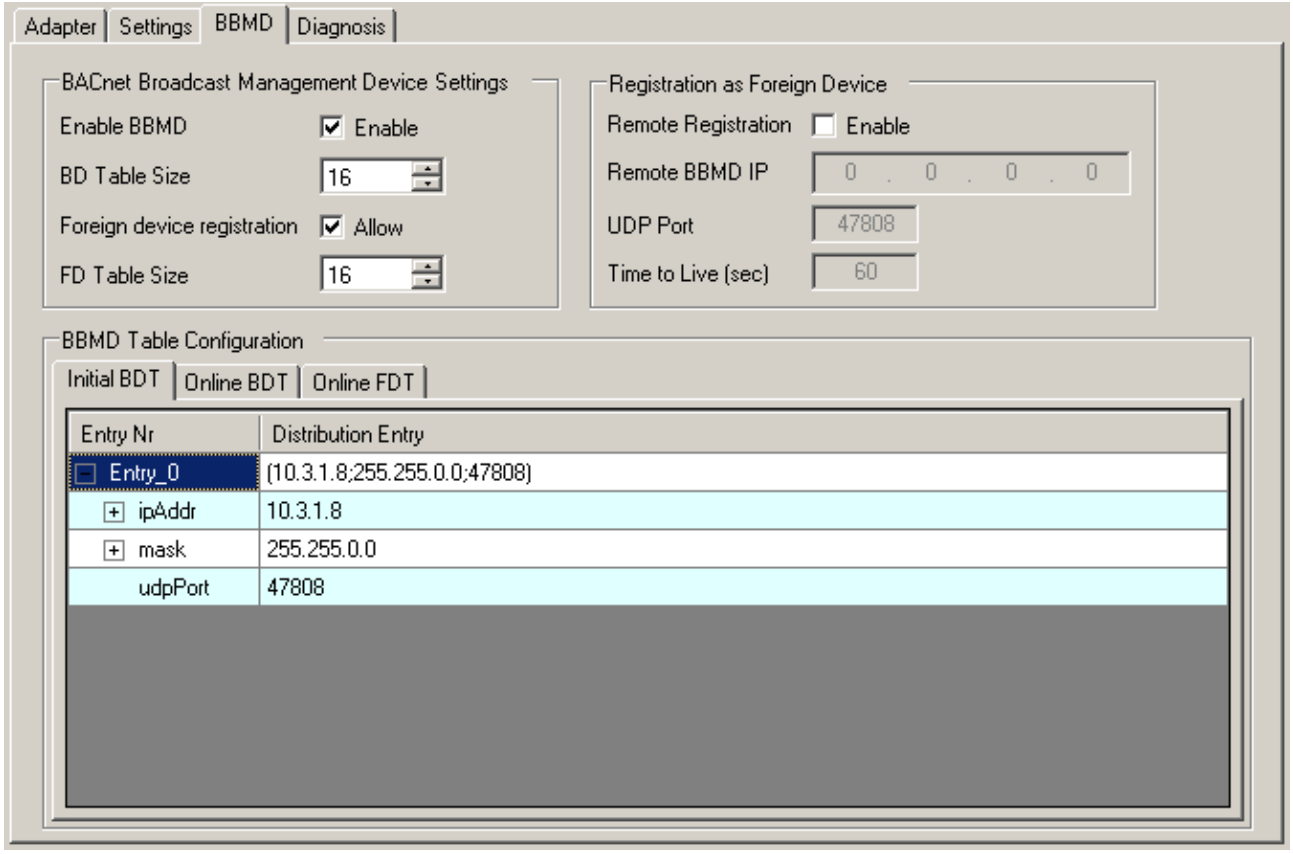
Über das Submenü "Registration as Foreign Device" kann sich ein TwinCAT BACnet/IP Geräte an einem bestehenden BACnet-Netzwerk anmelden. Dabei muss die IP-Adresse des entfernten BBMD aktiviert werden.



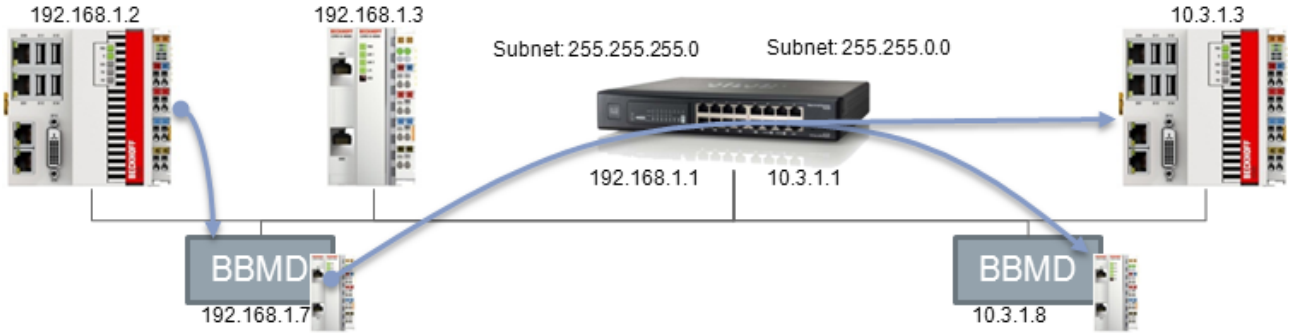
Der obige Screenshot zeigt die Konfiguration der BDT im Two-Hop-Modus. Anhand eines kleinen Beispiels soll die BBMD-Arbeitsweise kurz beschreiben werden. Im nachfolgenden Bild sind zwei IP-Subnetze über einen Router verbunden. Im Beispiel sendet das Gerät 192.168.1.2 eine IP-Broadcast-Nachricht (hier an 192.168.1.255) ins lokale Netz. Der BBMD, mit der IP Adresse 192.168.1.7, leitet die Nachricht als Forwarded-NPDU über den Router/Gateway (192.168.1.1) an den BBMD des IP-Subnetz 10.3.xxx.xxx weiter. Dies ist in diesem Fall der konfigurierte BBMD mit der IP-Adresse 10.3.1.8. Der entfernte BBMD verteilt die weitergeleitete Nachricht als IP-Broadcast (10.3.255.255) in sein lokales Netz.



Im nachfolgenden Screenshot ist die BBDM-Konfiguration im One-Hop-Modus dargestellt. Als Maske wird im Gegensatz zum Two-Hop-Verfahren die Subnetzmaske des Zielsystems eintragen, um anzudeuten, dass direkt eine Ziel-IP-Broadcast-Nachricht erstellt werden soll. Die letzten beiden Stellen der IP-Adresse sind in diesem Fall nicht relevant, da sie durch 255 ersetzt werden. Hier könnte also auch 10.3.0.0 stehen.



In diesem Fall sendet der BBMD (192.168.1.7) die Nachricht mit der MAC-Adresse des Router/Gateway an die IP-Broadcast-Adresse 10.3.255.255 des entfernten IP-Subnetz. Der entfernte BBMD leitet in diesem Modus keine Nachrichten weiter.



Über ein Kontext-Menü in der BBMD-Tabelle können Einträge in eine .txt-Datei exportiert und auch wieder importiert werden. Dies Erleichtert die Eingabe großer Tabellen.

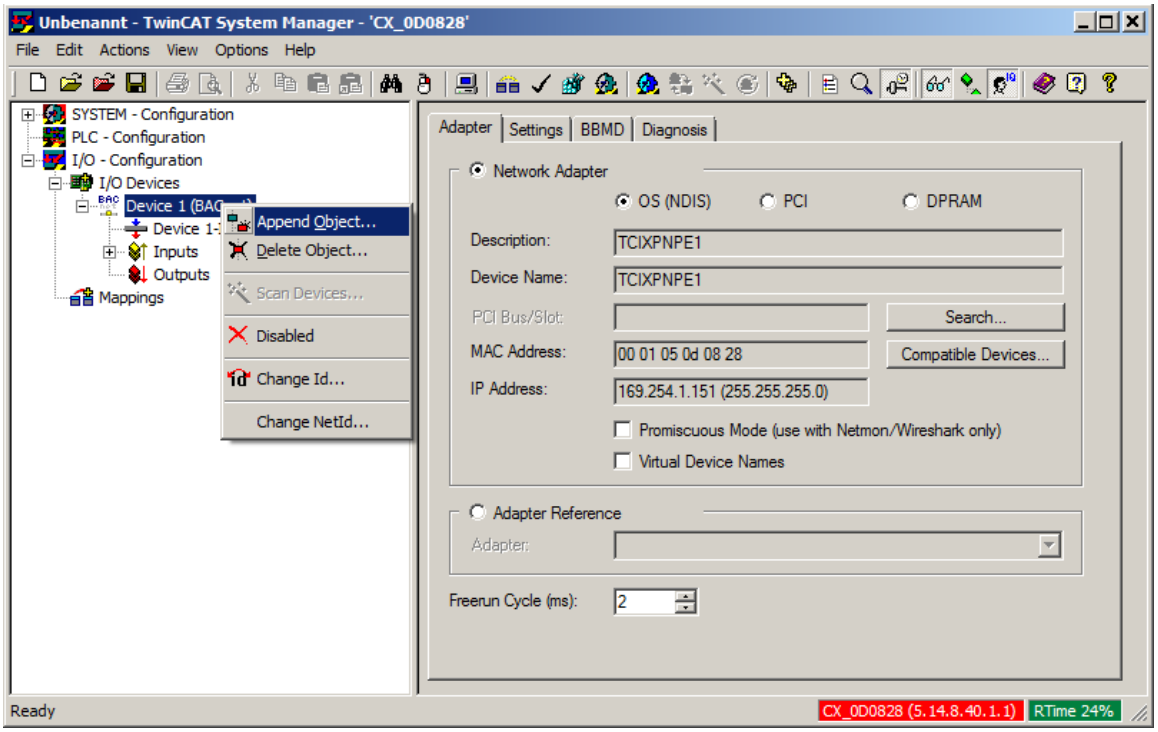
3 Anwendung

3.1 Beispiel: BACnet Adapter und Server anlegen

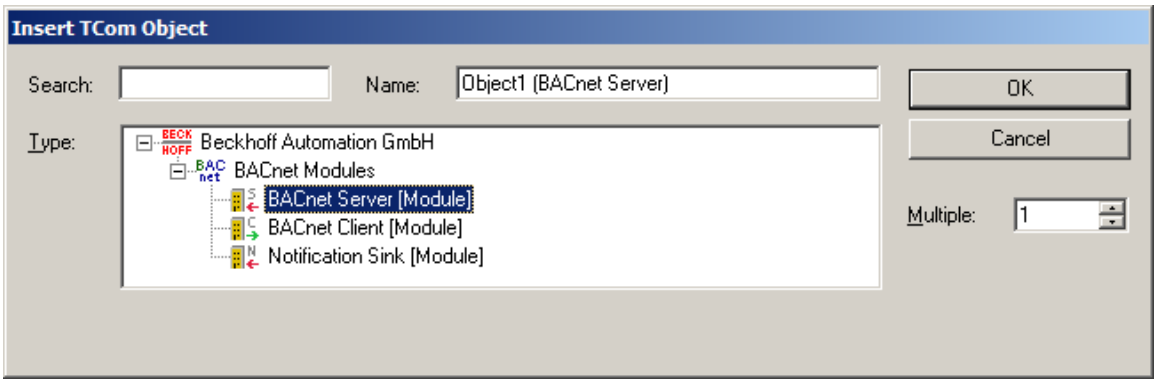
Wie im Kapitel BACnet Device, Server, Client [▶ 14] bereits beschrieben wird der BACnet-Adapter als Ethernet-Device in ein Projekt eingefügt. Im Folgenden wird schrittweise beschrieben, wie ein Server unterhalb des BACnet-Adapters angelegt wird.

1. Anlegen eines BACnet-Servers unter dem verwendeten BACnet-Adapter

   a) Klick mit der rechten Mouse-Taste auf den Adapter und "Append Object..." anwählen



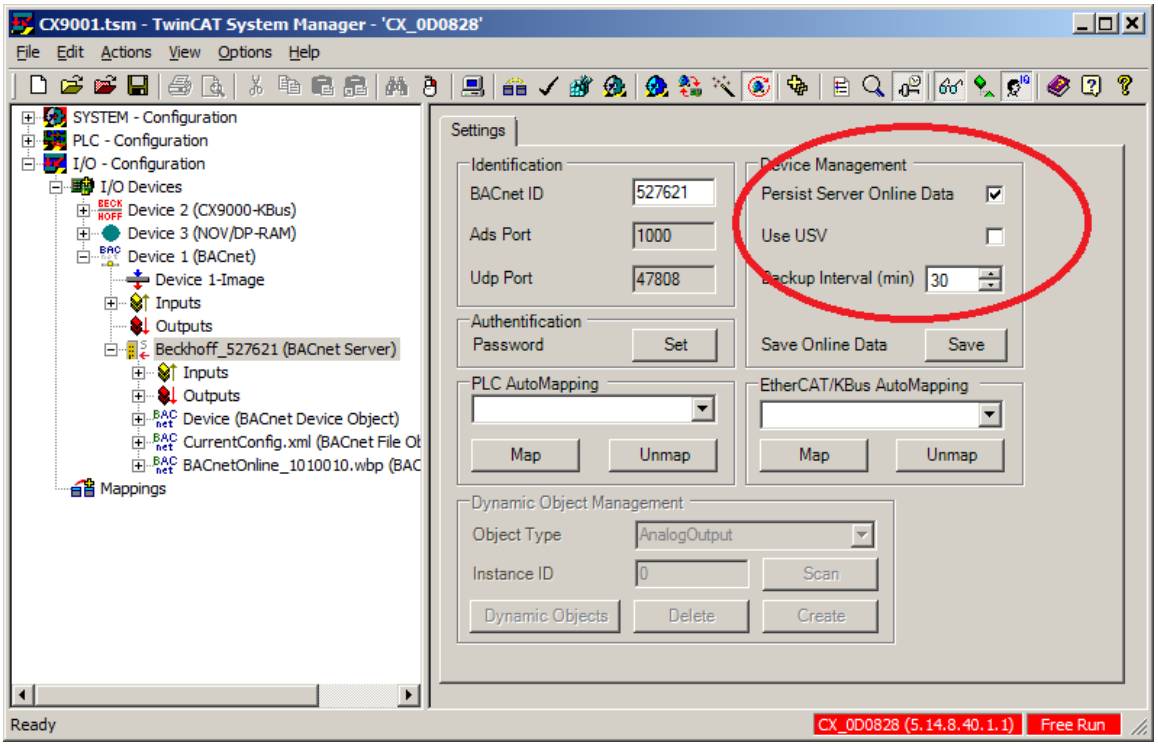
   b) Im Folgedialog das Modul "BACnet Server" anwählen und mit "OK" bestätigen



Supplement Key

Jeder projektierte BACnet-Adapter benötigt einen Lizenzschlüssel, den sogenannten "Supplement Key". Siehe Kapitel "BACnet Supplement Key einrichten" für weitere Informationen.

2. Soll der Server die Konfigurationsdaten aller untergeordneten Objekte persistent speichern, dann muss die Option Persist Server Online Data aktiviert werden. Im Folgenden ist der Reiter "Settings" des Server abgebildet unter dem die Option aktiviert werden kann (siehe auch Kapitel "Persistente Daten [▶ 60]"):

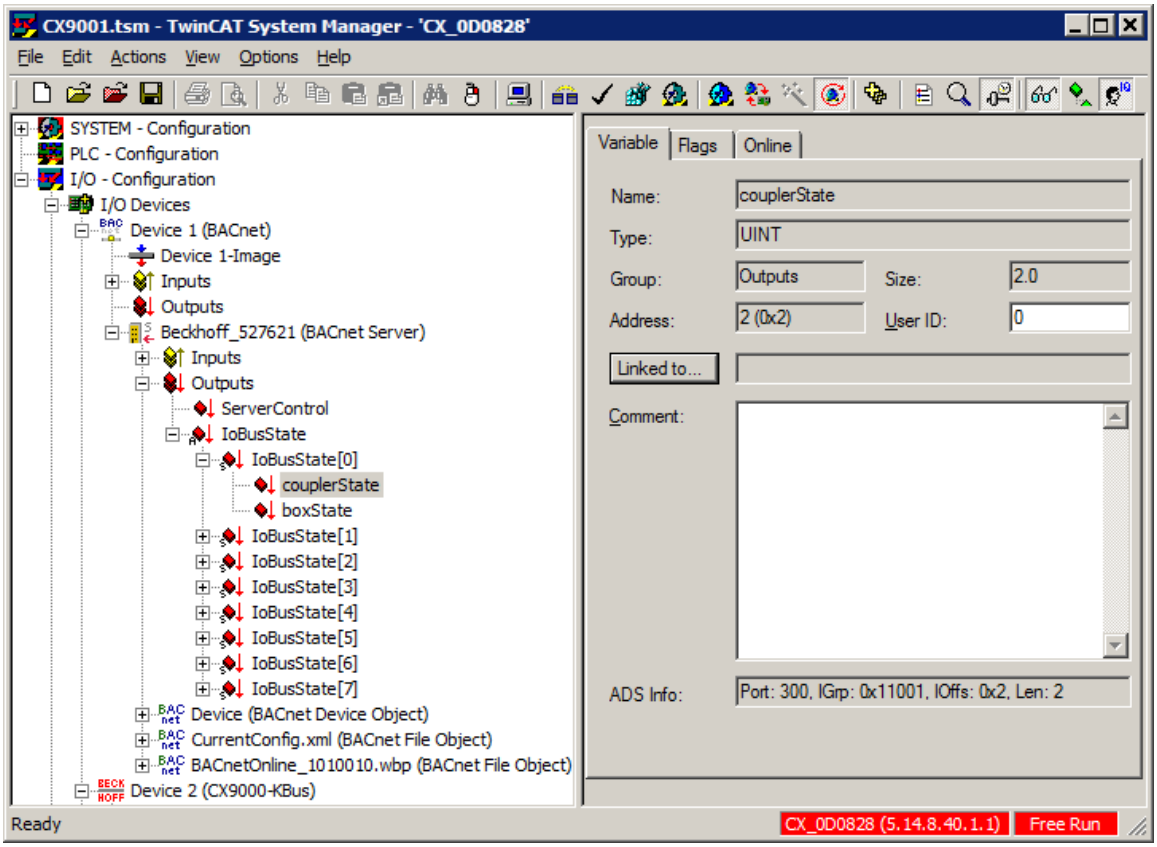


3.2 Beispiel: Manuelle Verknüpfung Hardware (Klemme), BACnet BinaryInput und SPS Programm

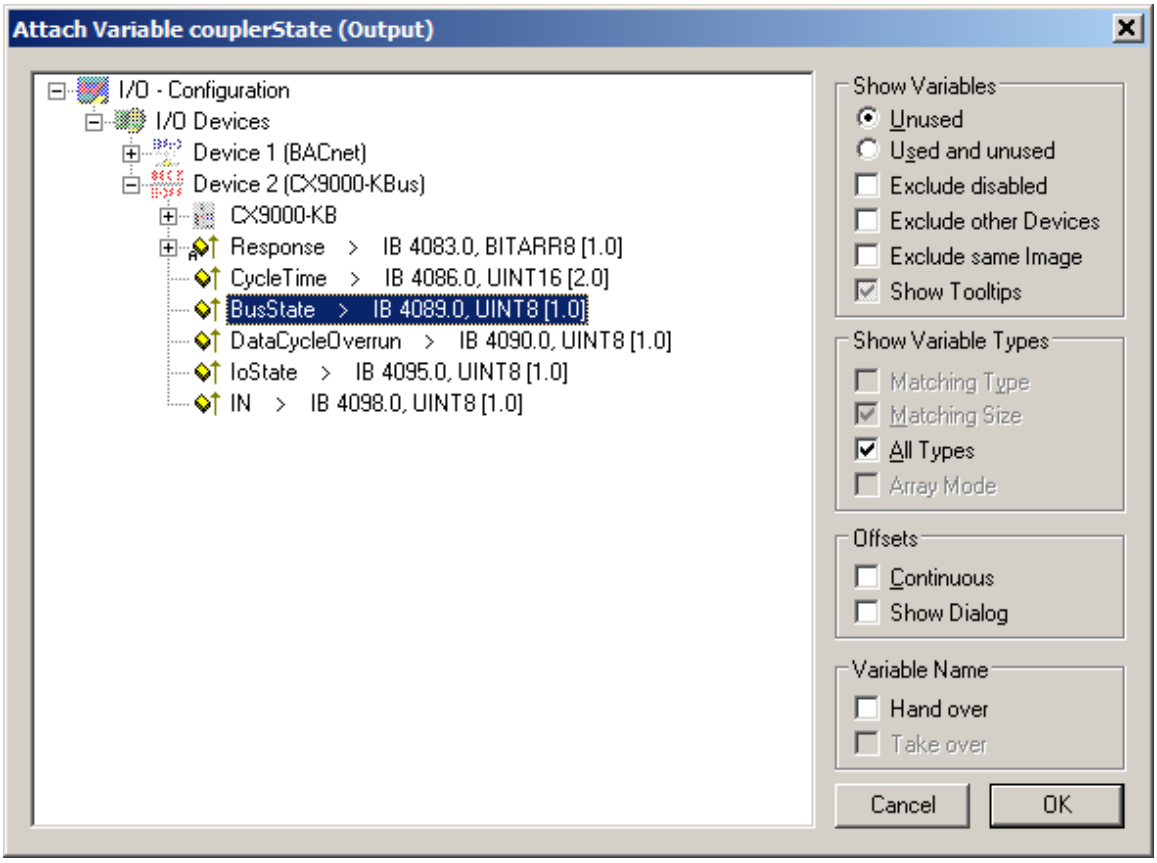
Im Folgenden wird die Verknüpfung von zwei digitalen Eingängen auf die entsprechenden Objekte und die SPS gezeigt.

1. Einen BACnet-Adapter und -Server anlegen (siehe "Beispiel: BACnet Adapter und Server anlegen [▶ 96]")
2. Den Status des Buskopplers, unter dem die Hardware-Klemme eingefügt wurde mit dem BACnet Server verbinden

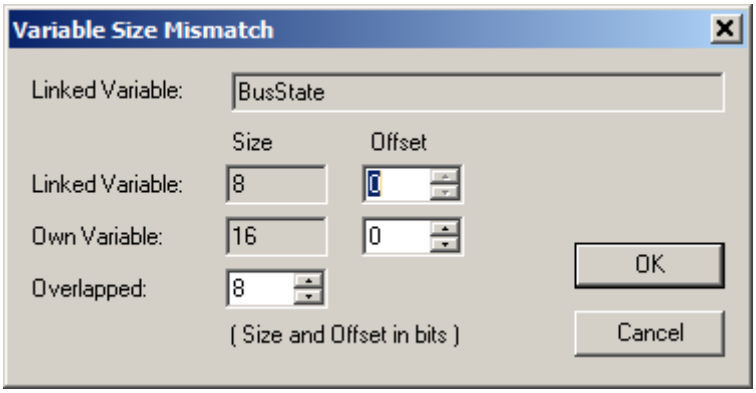
   a) Klick auf couplerState mit der rechten Maus-Taste



b) Im Folgedialog BusState des Buskopplers anwählen (ACHTUNG: Checkbox "Exclude other Devices" und "Exclude same Image" abwählen, zudem die Checkbox "All Types" anwählen) und mit "OK" bestätigen

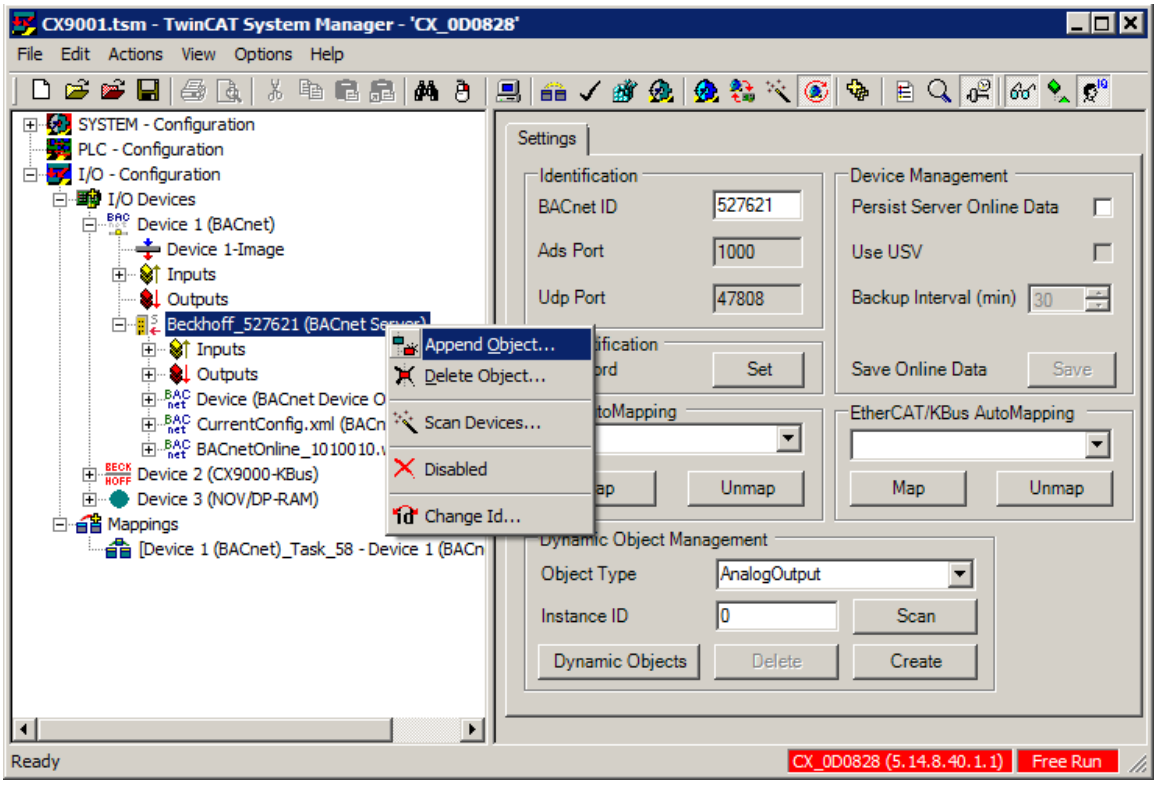


c) Den Folgedialog mit "OK" bestätigen

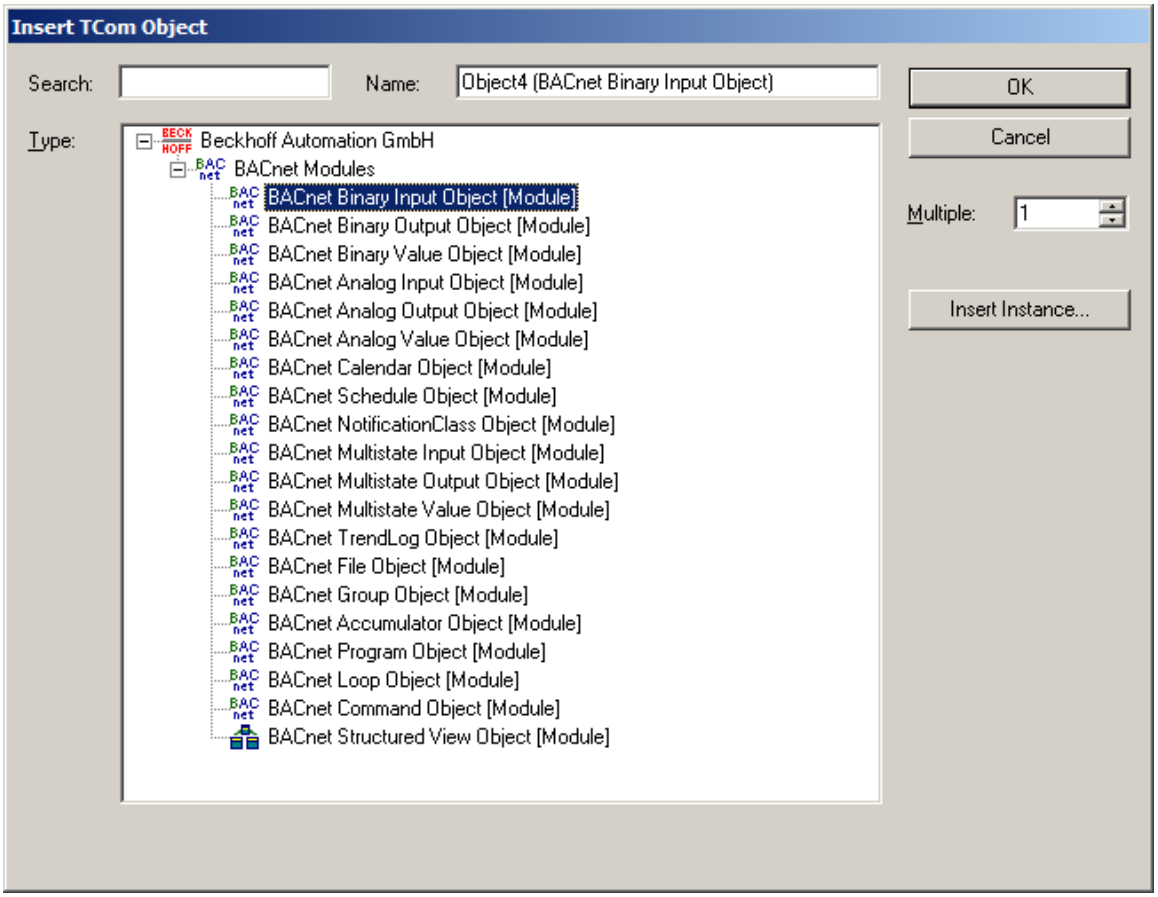


3. Nun wird das erste BinaryInput-Objekt unter dem zuvor erstellten BACnet-Server hinzugefügt

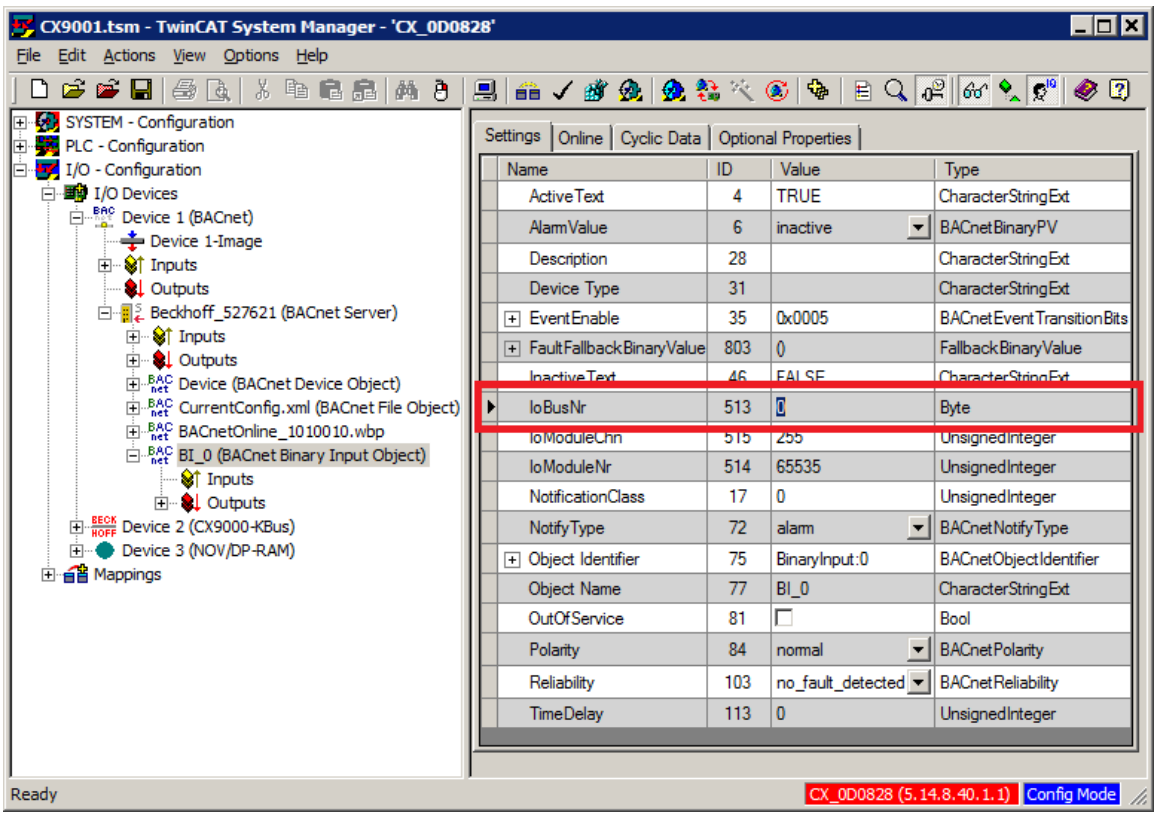
a) Klick mit der rechten Maus-Taste auf den BACnet-Server und "Append Object..." anwählen



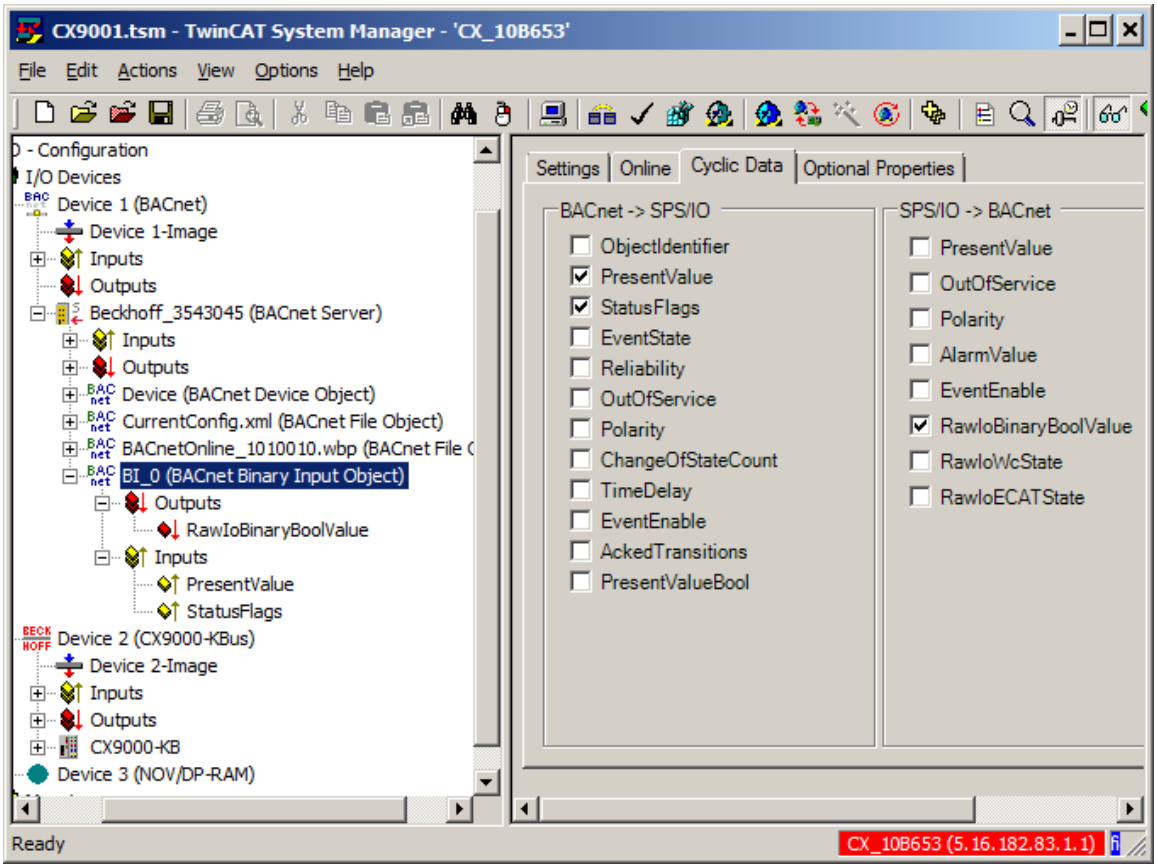
b) Im Folgedialog das Modul "BACnet Binary Input Object" anwählen und mit "OK" bestätigen



4. Für das Setzen der Property "Reliability" des BinaryInput-Objekts muss die Bus-Nummer des unter Punkt 2 verknüpften Bus-Status im BACnet-Objekt eingestellt werden (Bus-Index "0"):

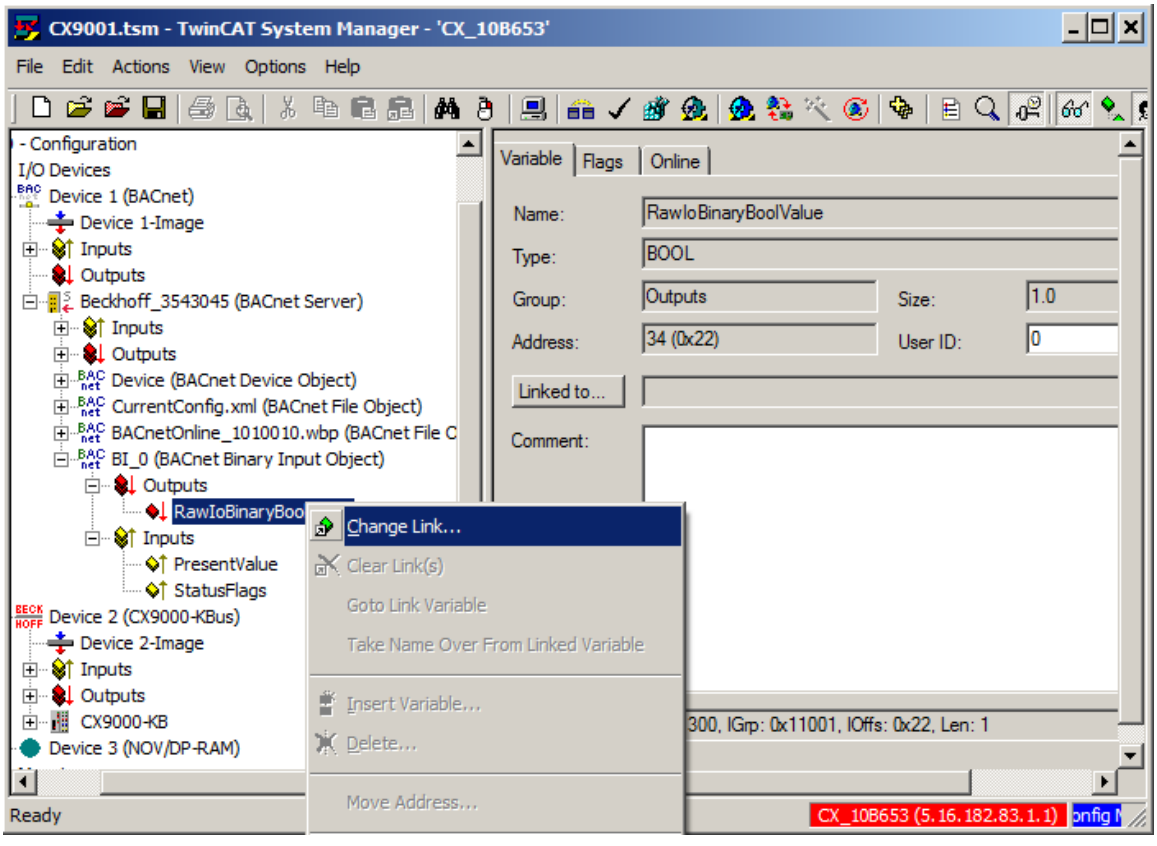


5. Für das spätere Mapping zur SPS müssen die entsprechenden Properties in das zyklische I/O-Mapping aufgenommen werden. Dazu müssen mindestens folgende Properties ausgewählt werden:

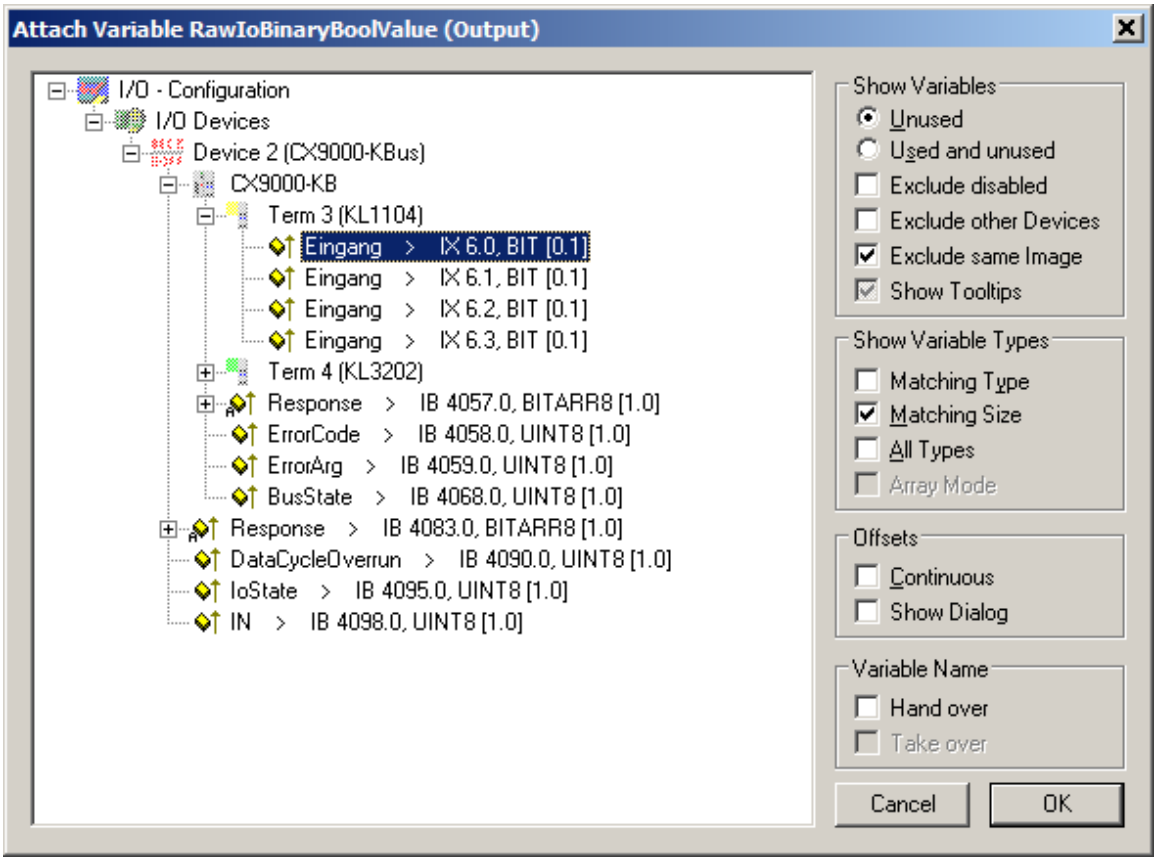


6. Anschließend wird die Verknüpfung zwischen Hardware-Klemme und BACnet-Objekt hergestellt. Dazu muss RawIoBinaryBoolValue mit dem entsprechenden Bit der Hardware-Klemme verknüpft werden:

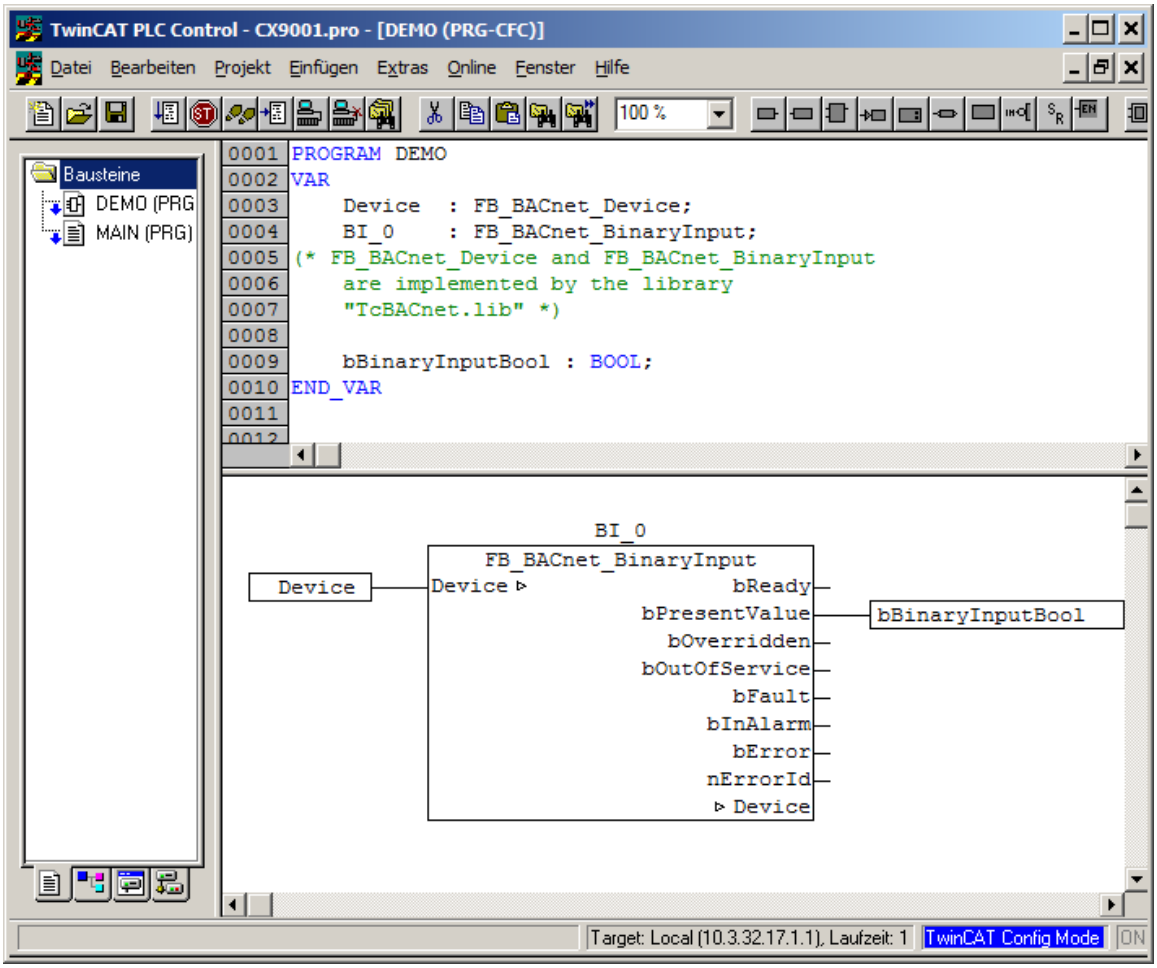
   a) Klick mit der rechten Maus-Taste auf RawIoBinaryBoolValue unter dem BACnet-Objekt "BI_0" und "Change Link..." anwählen



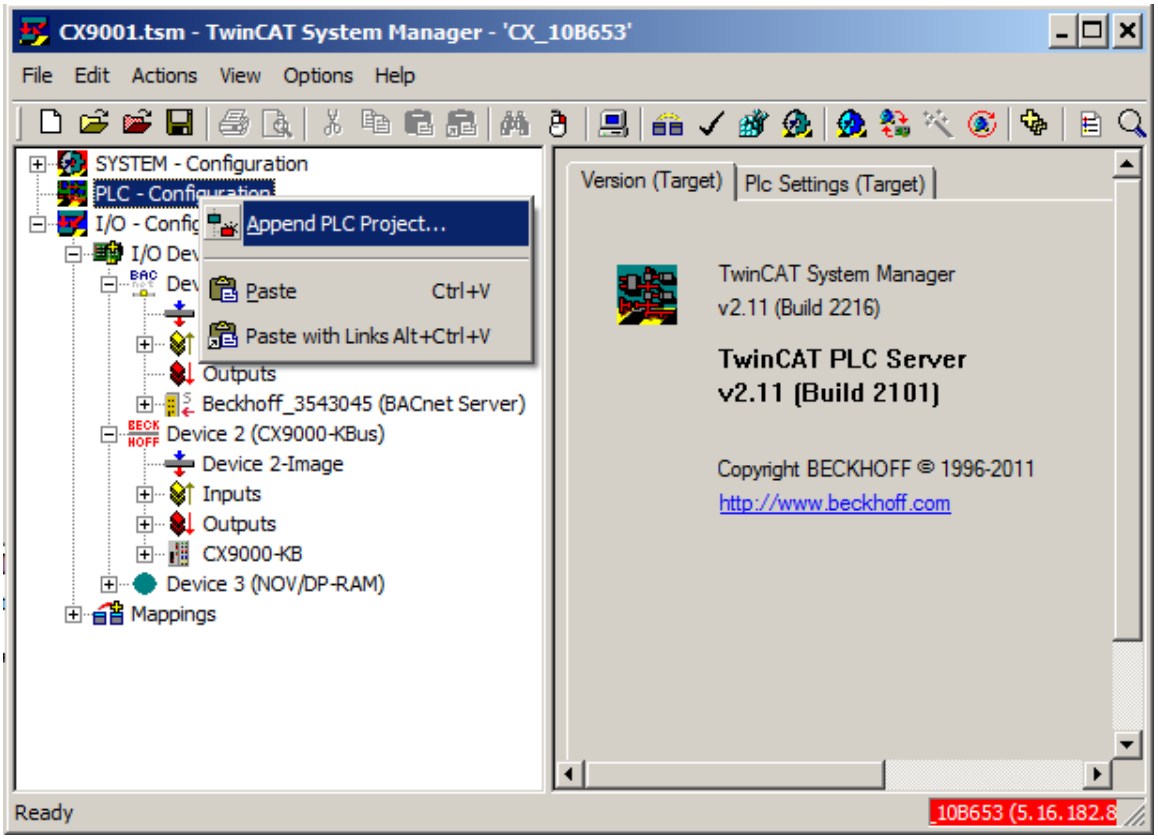
b) Im Folgedialog das Symbol "Eingang" der entsprechenden Hardware-Klemme anwählen (ACHTUNG: Checkbox "Exclude other Devices" und "Exclude same Image" abwählen, zudem die Checkbox "Matching Size" anwählen) und mit "OK" bestätigen



7. Für den Zugriff auf BACnet-Objekte aus einer SPS-Laufzeit kann die Library "TcBACnet.lib [▶ 374]" verwendet werden. Diese stellt sämtliche Objekte als Funktionsbausteine mit den entsprechenden I/O-Punkten für das Mapping im System Manager zur Verfügung. Im Folgenden wird ein SPS-Projekt mit einer Instanz der Funktion "FB_BACnet_BinaryInput [▶ 399]" mit Namen "BI_0" erstellt. Die Funktion wird wie folgt in das SPS-Programm "DEMO" eingefügt (Das Programm "DEMO" wird wiederum im Programm "MAIN" aufgerufen. Programm "MAIN" wird als Task in die Taskkonfiguration aufgenommen). Die Bausteininstanz "Device" stellt den Zugriff auf das Device-Objekt und den Server-Status bereit und muss an die Bausteininstanz "BI_0" übergeben werden:

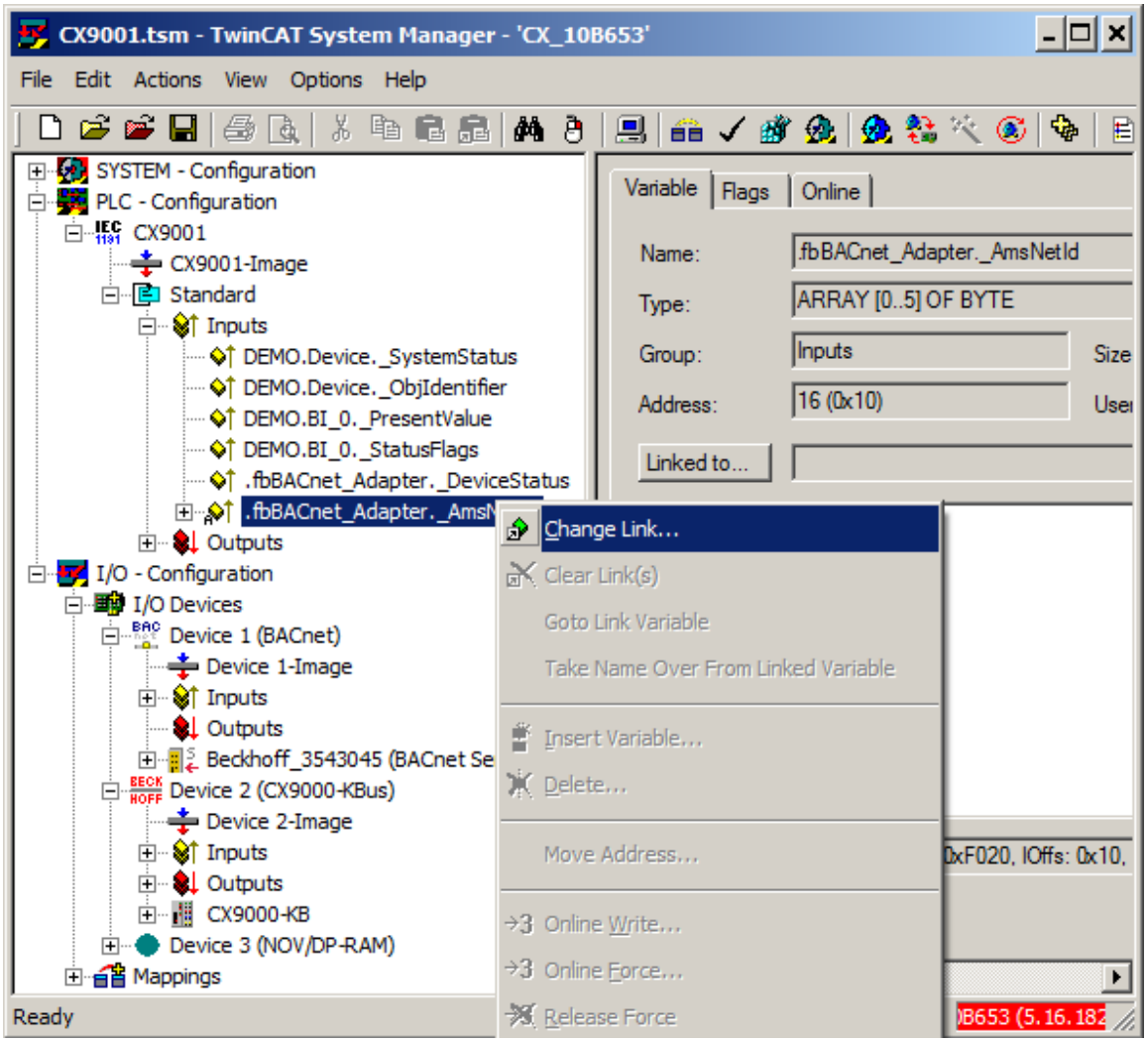


8. Nach Übersetzen des SPS-Projekts muss dieses im System Manager bereitgestellt werden. Dazu genügt ein Klick mit rechter Maus-Taste auf das Symbol "PLC - Configuration". Danach mittels "Append PLC Project..." die TPY-Datei des zuvor übersetzten SPS-Projekts hinzufügen:

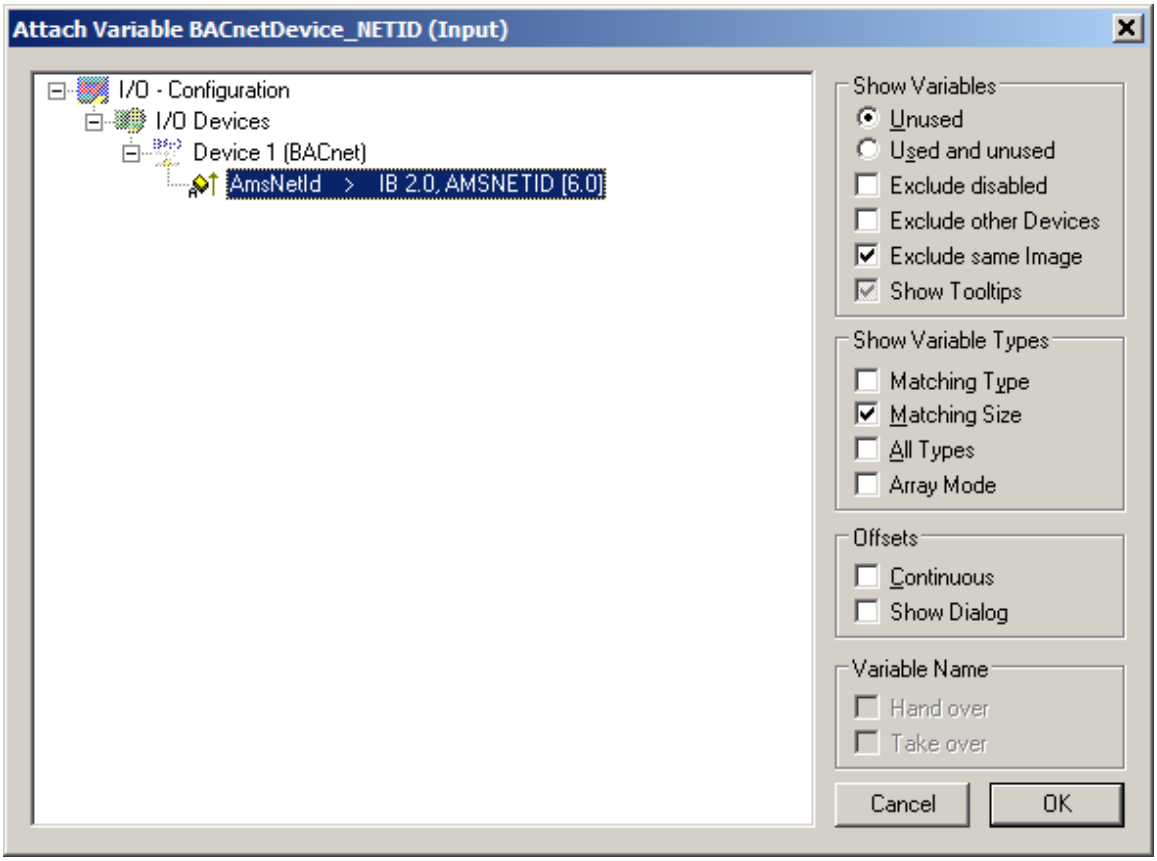


9. Nach dem Hinzufügen des SPS-Projekts stehen sämtliche I/O-Variablen des SPS-Programms im System Manager zur Verfügung. In jedem SPS-Projekt das die Library "TcBACnet.lib" einbindet, steht die globale Instanz "fbBACnet_Adapter" vom Typ FB_BACnet_Adapter [▶ 378] zur Verfügung. Diese dient dem Zugriff auf den BACnet-Netzwerkadapter, dessen AMS NetID und den Link-Status. Die I/O-Variablen unterhalb der Instanz müssen nun mit den I/O-Punkten des BACnet-Adapters verbunden werden:

   a) Klick mit der rechten Maus-Taste auf das Symbol ".fbBACnet_Adapter._AmsNetId" und "Change Link..." anwählen

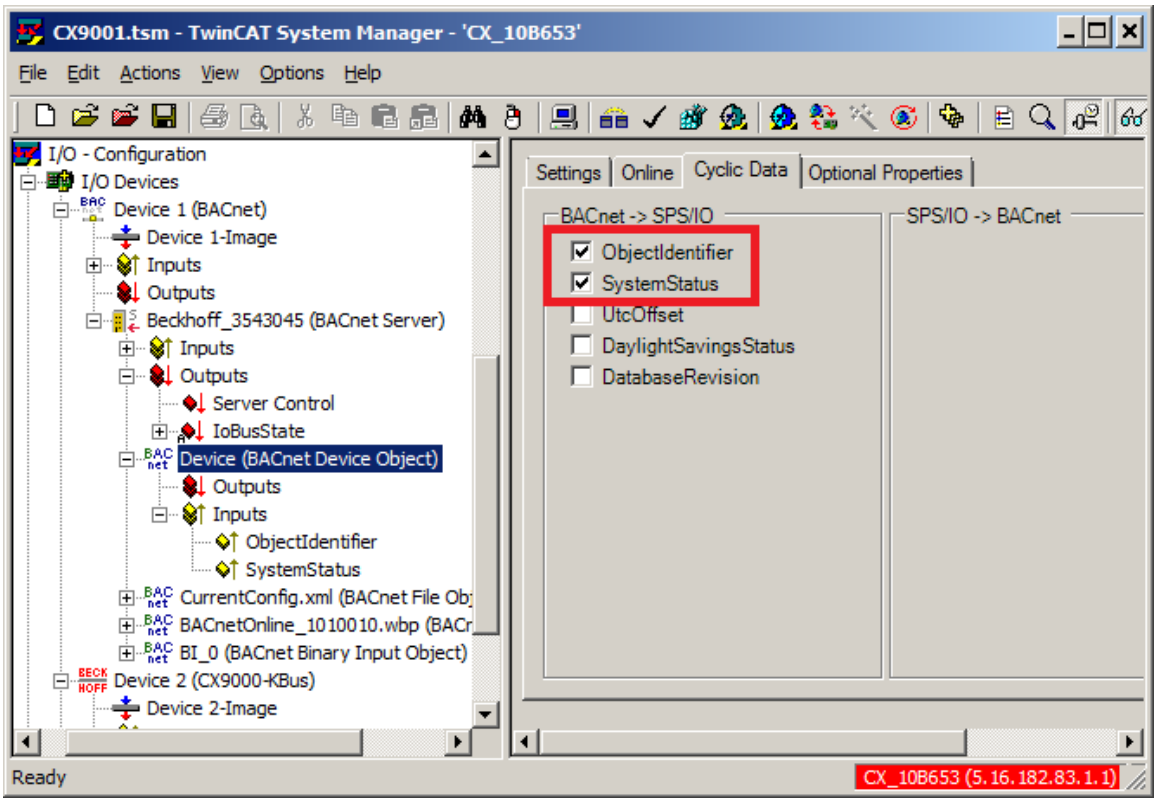


b) Im Folgedialog AmsNetId des BACnet-Device anwählen (ACHTUNG: Checkbox "Exclude other Devices" abwählen, zudem die Checkbox "Matching Size" anwählen) und mit "OK" bestätigen



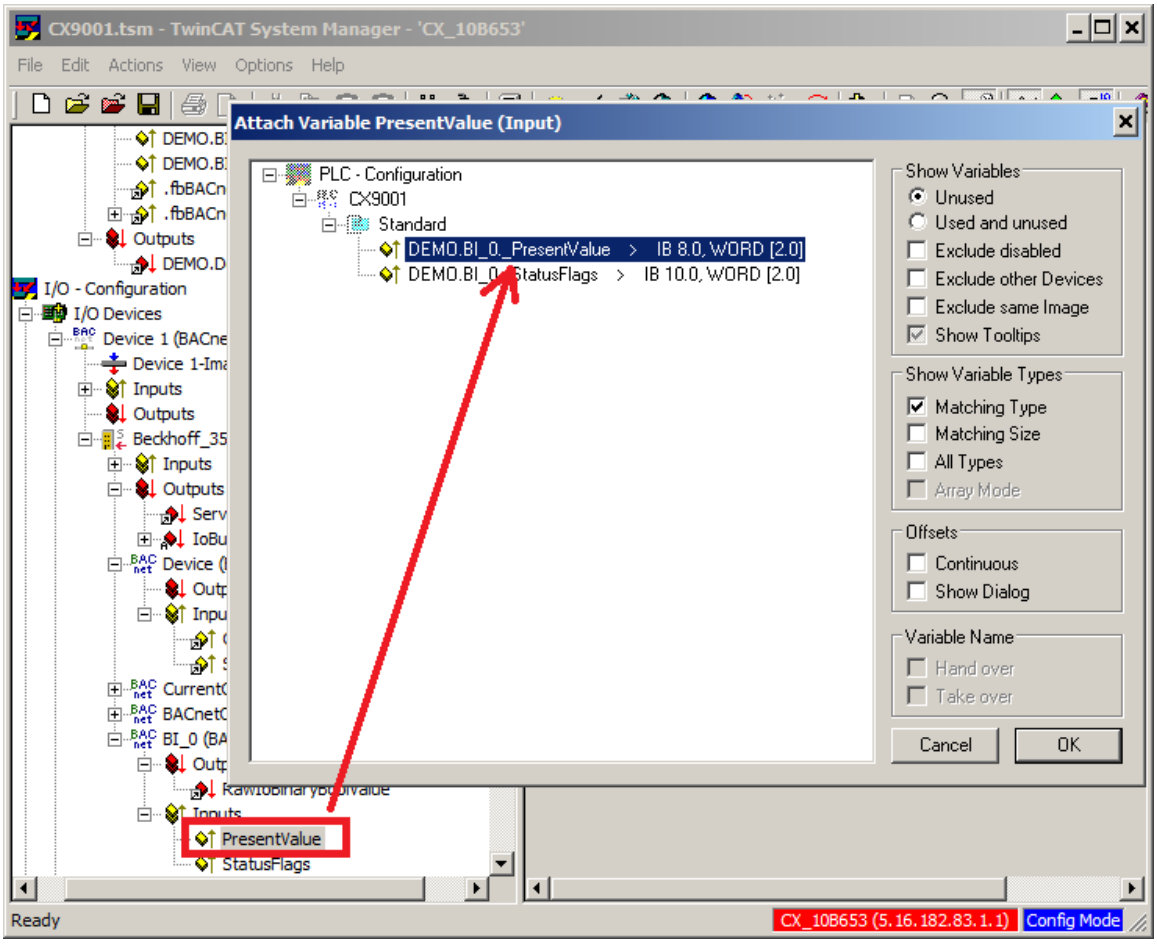
10. Schritt 9 mit "DeviceStatus" des BACnet-Adapters wiederholen
11. Die Prozessdaten-Inputs: "_SystemStatus", "_ObjIdentifier" und -Outputs: "_ServerControl" des Device Objekts müssen auf gleiche Weise verknüpft werden. Dadurch wird der Status des BACnet-Servers und des lokalen Device-Objekts in der SPS abgebildet.

    a) Prozessdaten des Device-Objekts anlegen:



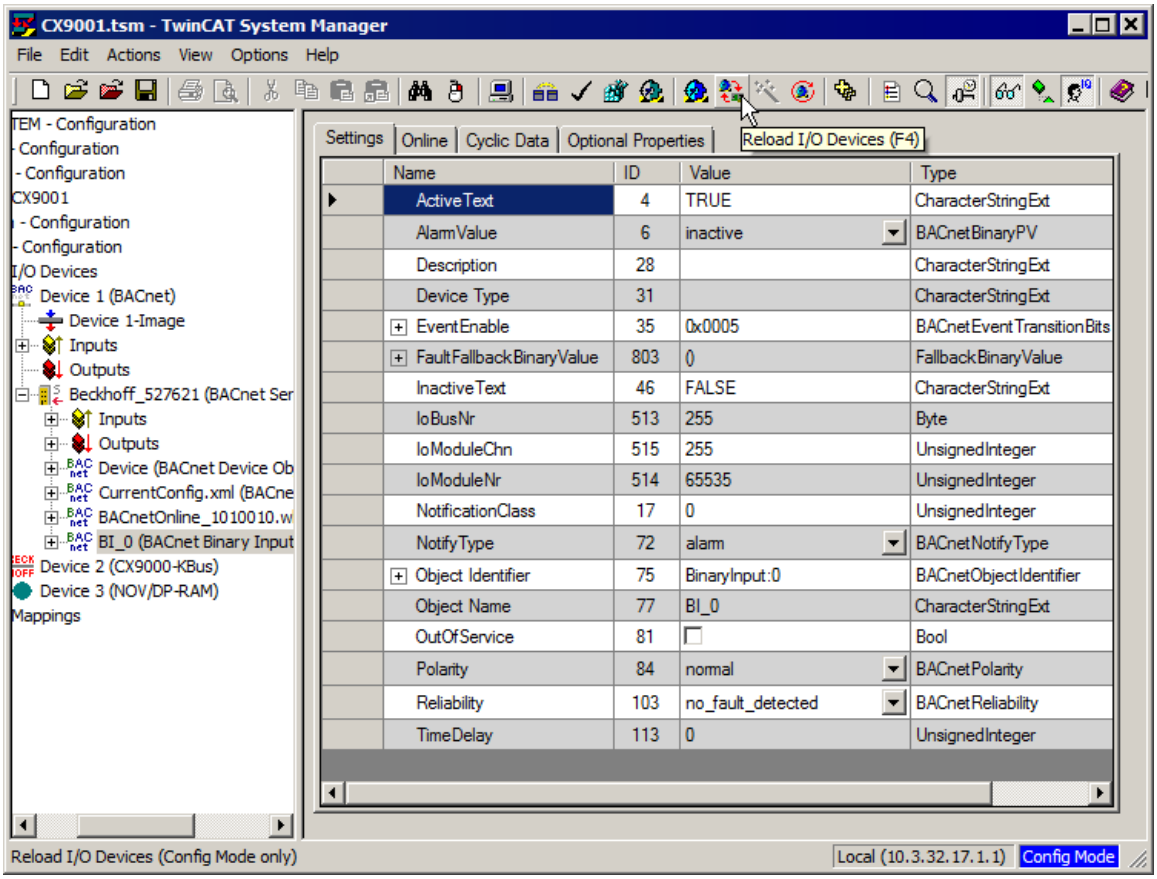
    b) Verknüpfung zwischen SPS- und I/O-Prozessdaten anlegen.

12. Nun müssen sämtliche I/O-Signale des SPS-BACnet-Bausteins "BI_0" mit dem eigentlichen BACnet-Objekt verknüpft werden. Die Verknüpfungen werden nach der gleichen Vorgehensweise wie unter Punkt 9 beschrieben durchgeführt ("_PresentValue" und "_StatusFlags"):

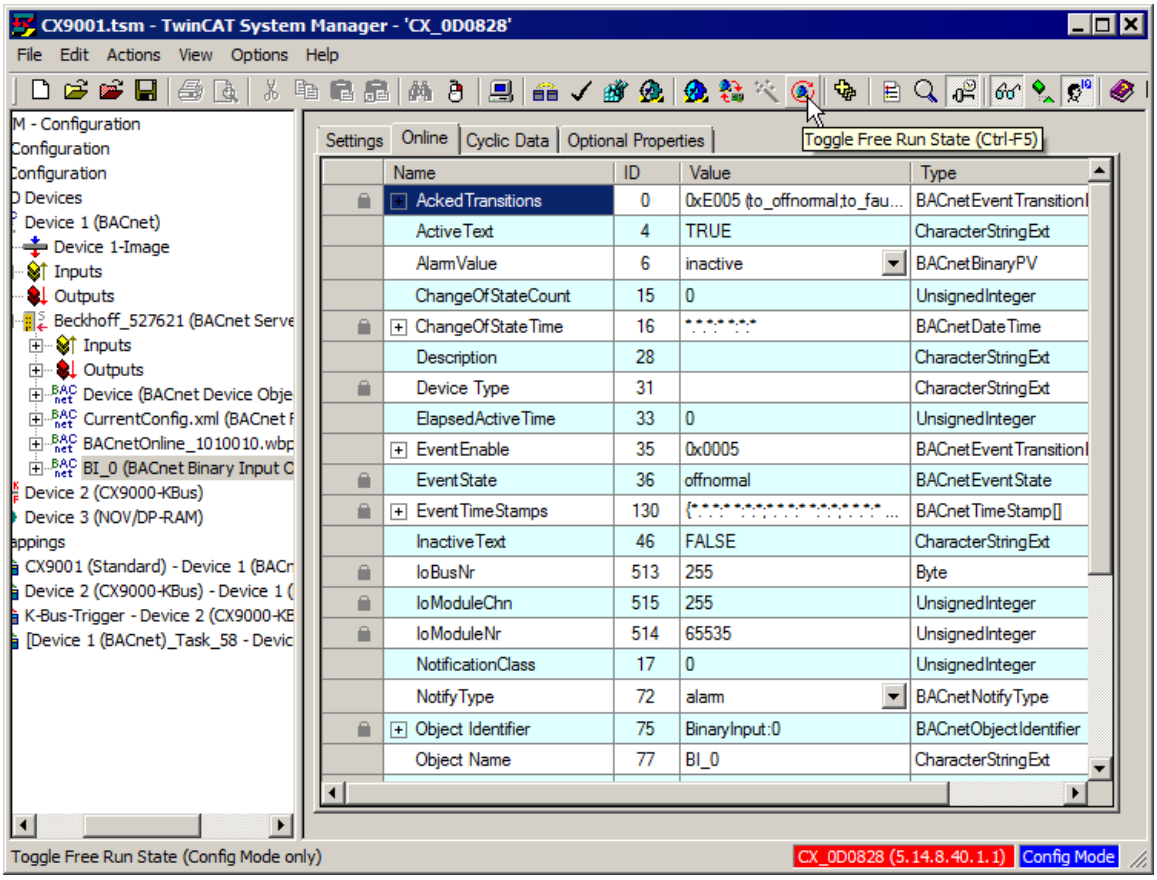


13. Zur Funktionsprüfung ohne aktive SPS-Laufzeit kann die Konfiguration des BACnet-Device und der Hardware-Klemmen im Free Run Modus ausprobiert werden:

    a) "Reload I/O Devices (F4)" aus der Tool-Bar oder F4 betätigen

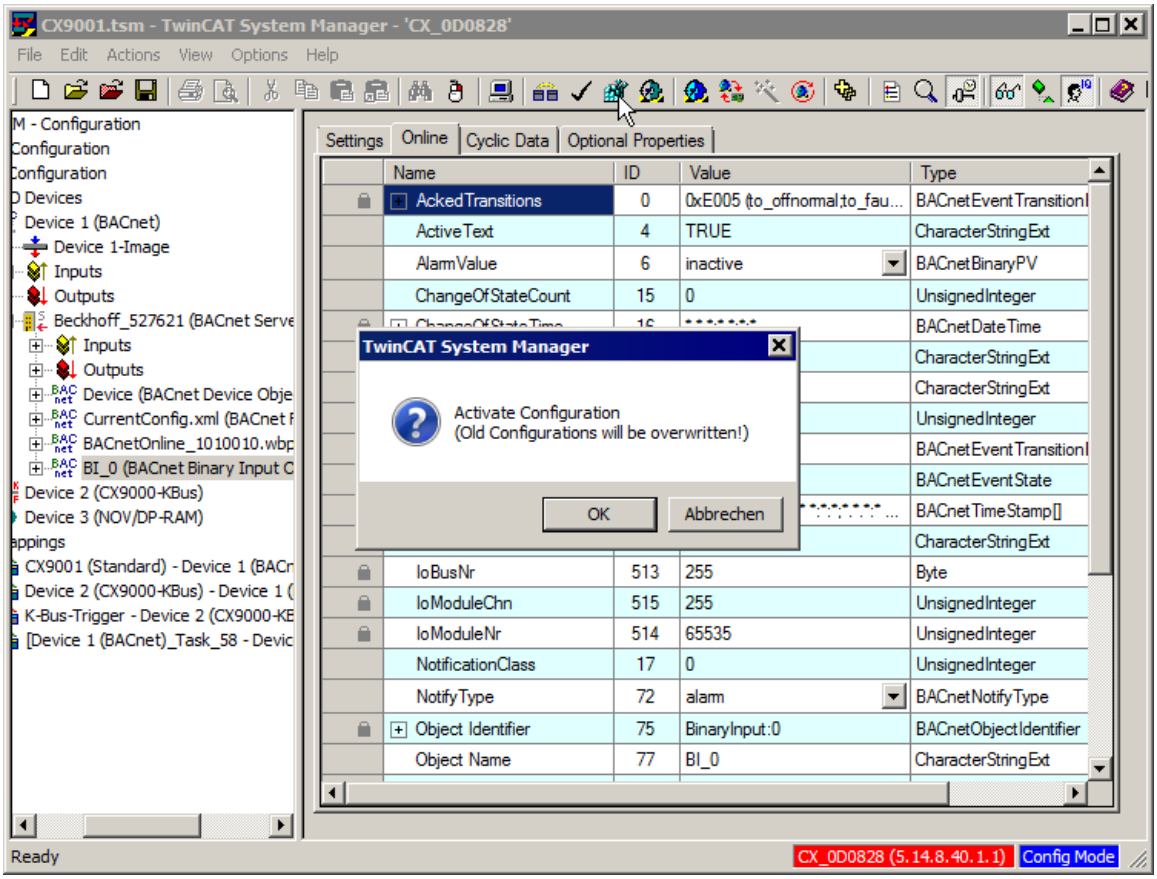


b) "Toggle Free Run State (Ctrl-F5)" aus der Tool-Bar oder CTRL+F5 betätigen

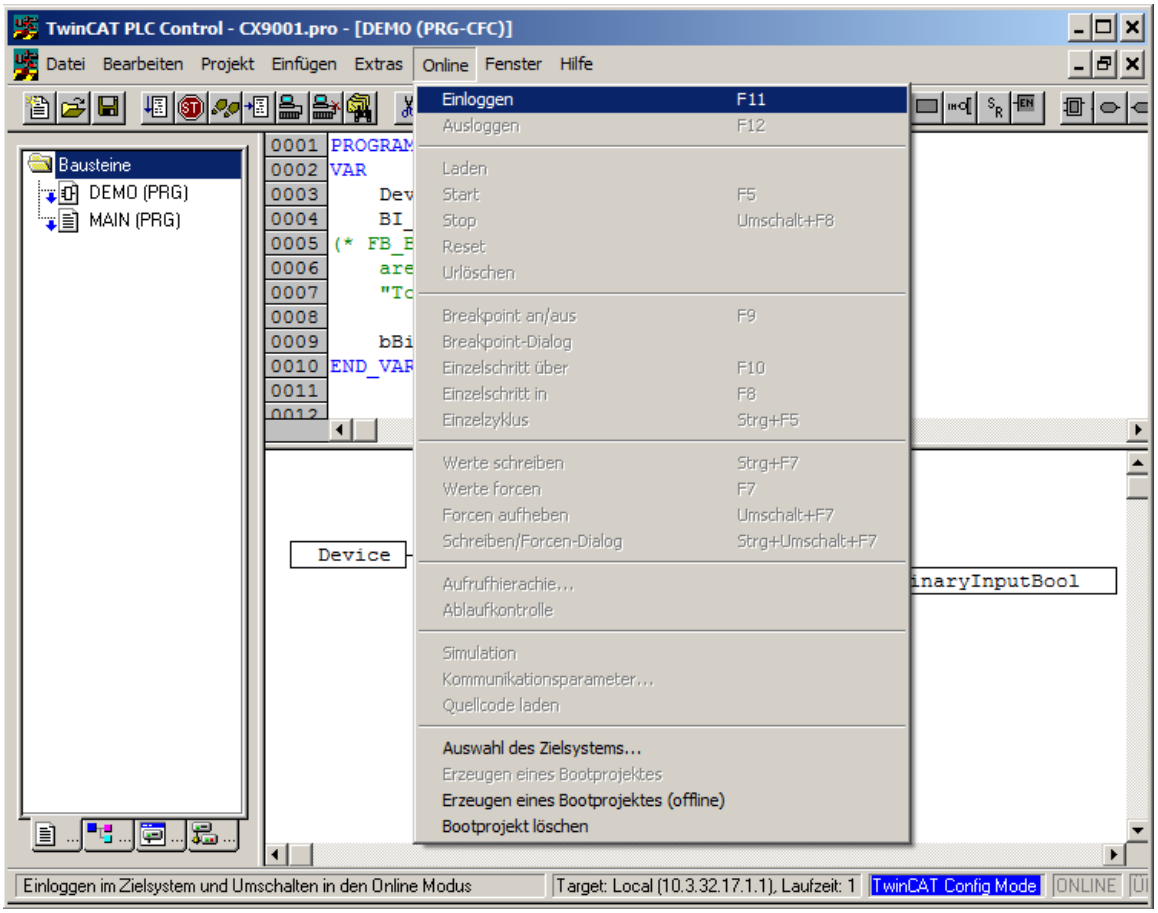


c) Anschließend sollte der I/O-Bus des Zielsystem auf RUN schalten. Der Zustand des Hardware-Eingangs kann nun im Reiter "Online" des BACnet-Objekts "BI_0" beobachtet werden. Die Property Present_Value Nr. 85 repräsentiert den logischen Zustand des Hardware-Eingangs. Mit Hilfe des Kontextmenüs des Reiters "Online" sollte vorab der Modus Auto Update aktiviert werden (siehe Kapitel "BACnet Objekte und Properties [▶ 22]").

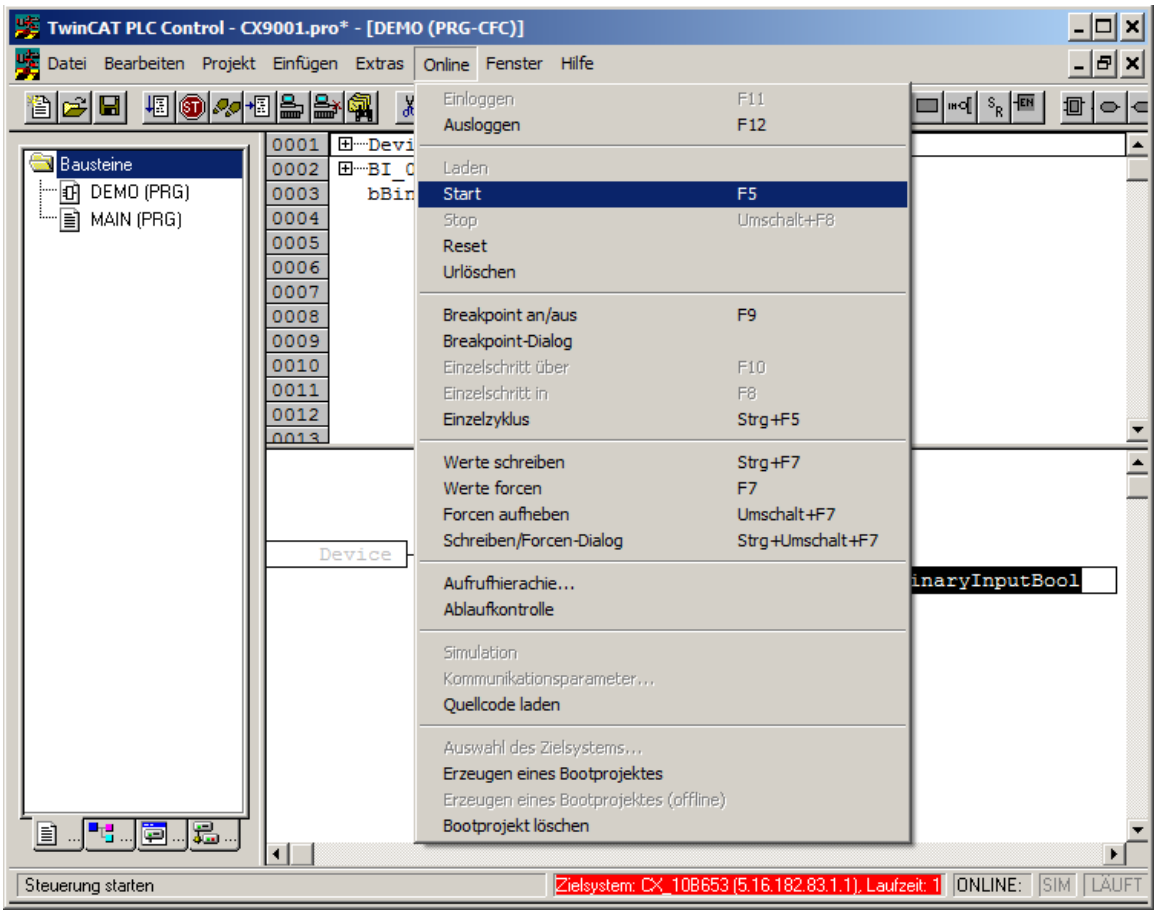
14. Nach erfolgreichem Test im "Free Run" Modus muss die Konfiguration dauerhaft aktiviert werden. Dazu wird die Konfiguration wie üblich mittels Symbols "Activate Configuration" aus der Tool-Bar oder mittels CTRL+SHIFT+F4 aktiviert (ins Zielsystem geladen):



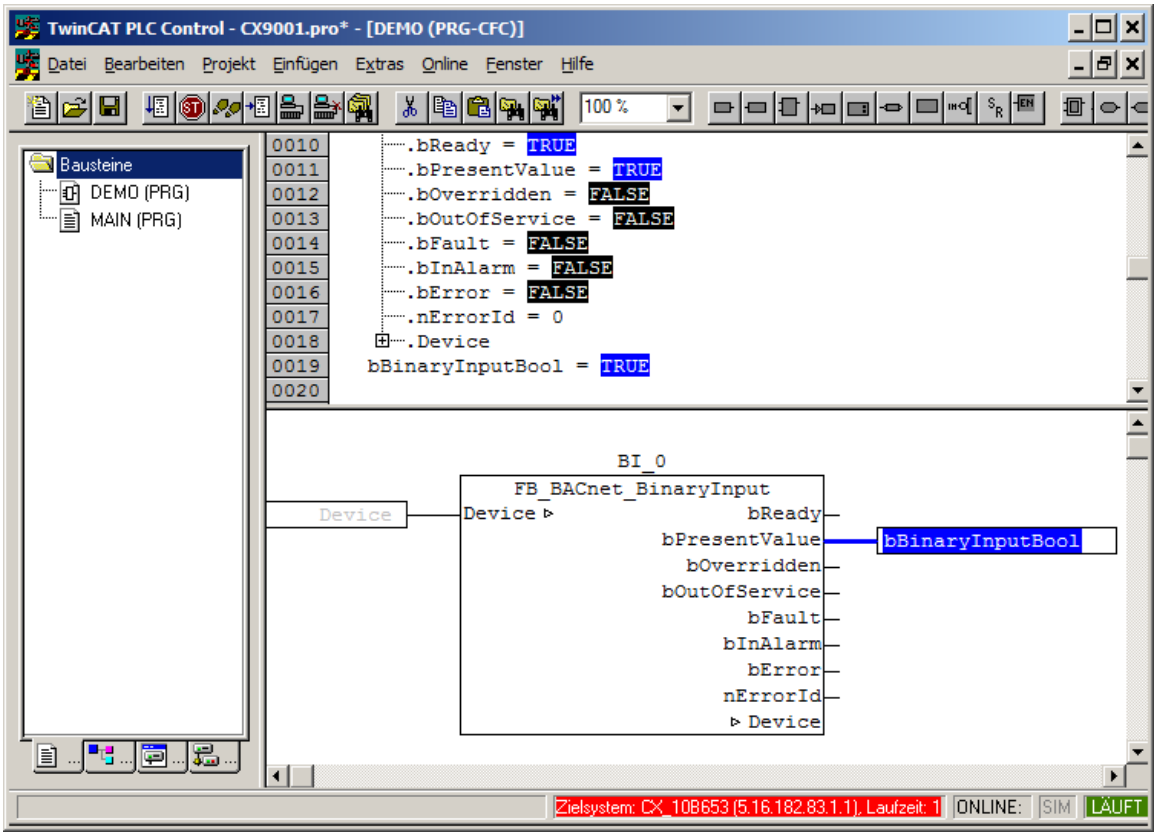
15. Anschließend erfolgt das Laden der SPS-Laufzeit:



16. und Starten in RUN Modus:



17. In der Online-Ansicht des Programms "DEMO" können die Zustände des BACnet-Objekts beobachtet werden. Der Zustand des Signals Present_Value repräsentiert den logischen Zustand des Hardware-Eingangs:



3.3 Beispiel: Verknüpfung von BinaryInput- und BinaryOutput-Objekten im SPS Programm

Im Folgenden wird beispielhaft gezeigt, wie die Verknüpfung von einem binären Eingang mit einem binären Ausgang mit Hilfe von BACnet-Objekten und eines SPS-Programms realisiert werden kann. Genutzt wird dafür das SPS-Automapping.

Die Funktion stichpunktartig erklärt:

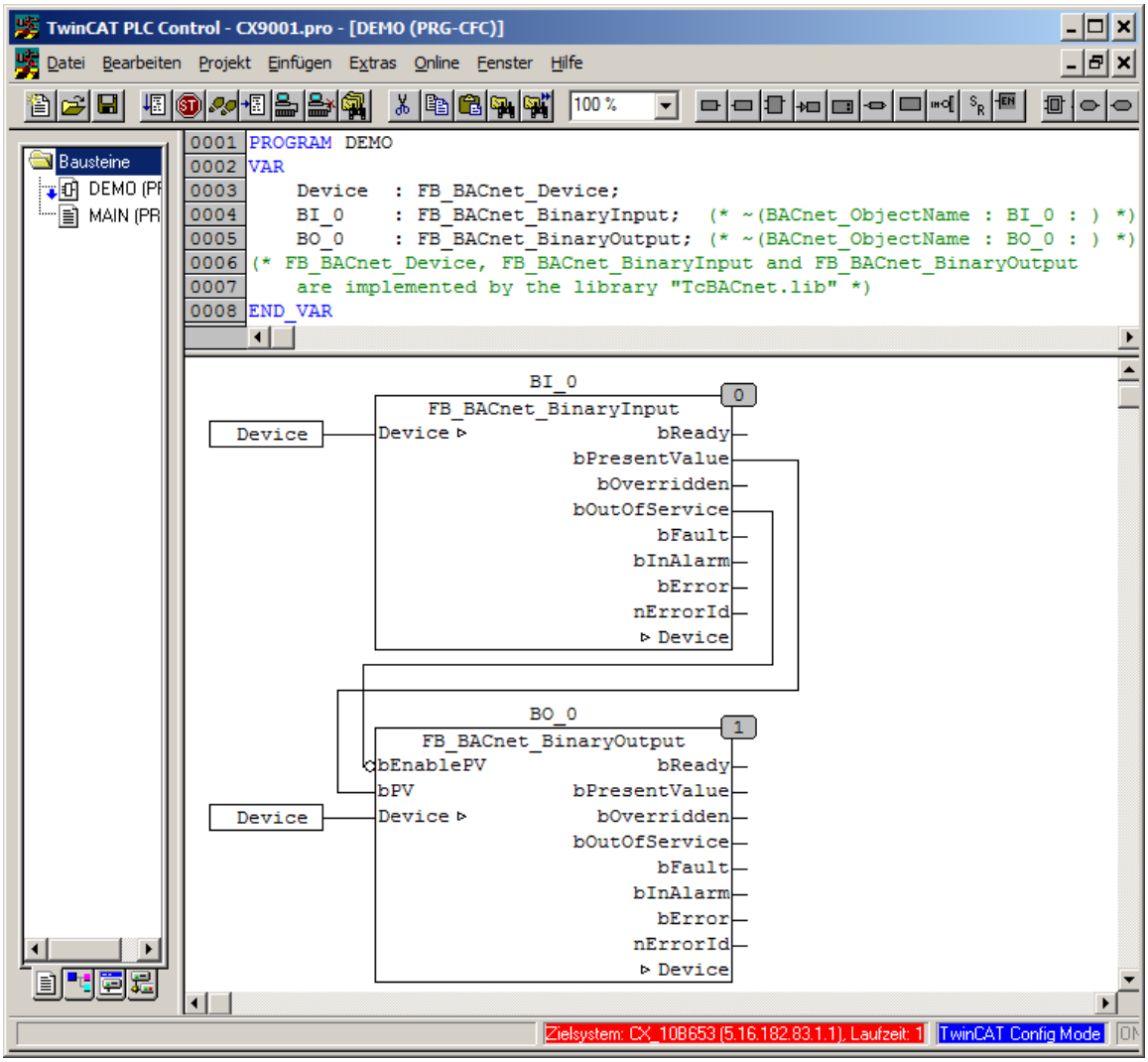
- Ein binärer Eingang, repräsentiert durch das BACnet-Objekt "BI_0", soll den binären Ausgang, repräsentiert durch das BACnet-Objekt "BO_0", mit Priorität "12" mit dem eigenen PresentValue beschreiben, wenn sich das Objekt "BI_0" nicht im Zustand OutOfService befindet
- Befindet sich das Objekt "BI_0" im Zustand OutOfService, so soll der Zugriff auf das PresentValue mit Priorität "12" auf das Objekt "BO_0" gelöscht werden

Das Beispiel https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749043083.zip kann hier: https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749043083.zipheruntergeladen werden.

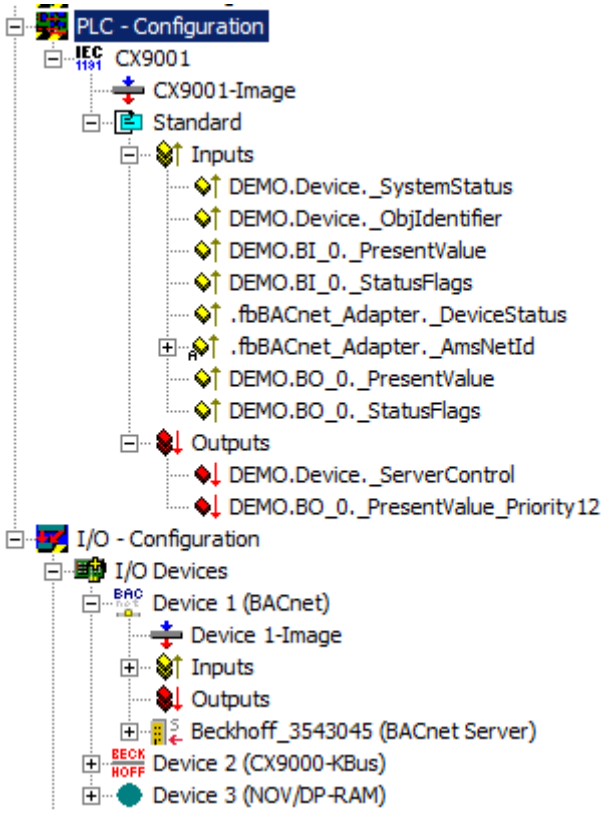
Die Vorgehensweise schrittweise mit Hilfe von Bildschirmabzügen erklärt (Anlegen von Server und Objekten siehe auch Beispiel "BACnet Adapter und Server anlegen [▶ 96]" und "Manuelle Verknüpfung Hardware (Klemme), BACnet BinaryInput und SPS Programm [▶ 97]"):

1. BACnet-Adapter und -Server anlegen (siehe Beispiel "BACnet Adapter und Server anlegen [▶ 96]")
2. Einen I/O-Bus (K-Bus, E-Bus, BK90xx) anlegen
3. I/O-Automapping [▶ 75] des Busses auf BACnet durchführen (siehe "Beispiel:I/O-Automapping [▶ 119]")

4. SPS-Projekt anlegen und Objekt-Instanzen einfügen (Die Verknüpfung zu den Eingangs- bzw. Ausgangsobjekten wird mit Hilfe der Objektnamen "BI_0" und "BO_0" im Kommentar realisiert, siehe auch "PLC Automapping [▶ 64]"):

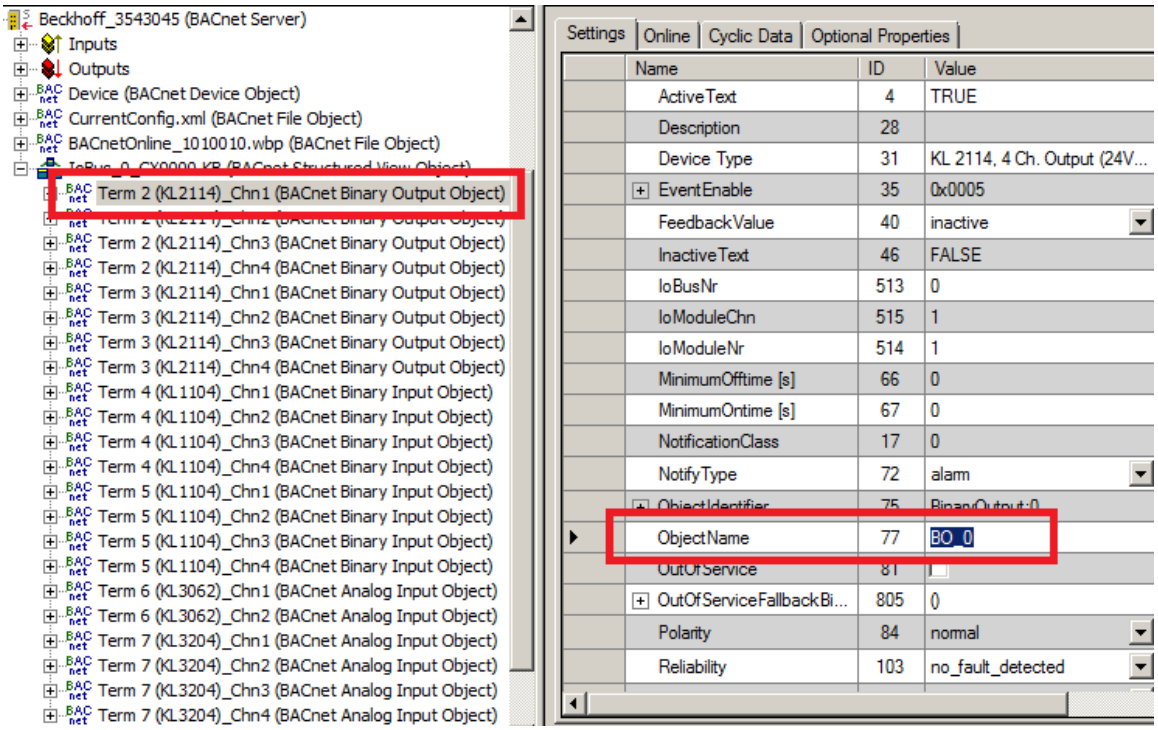


5. Das SPS-Projekt (.tpy) zur System Manager Konfiguration hinzufügen:

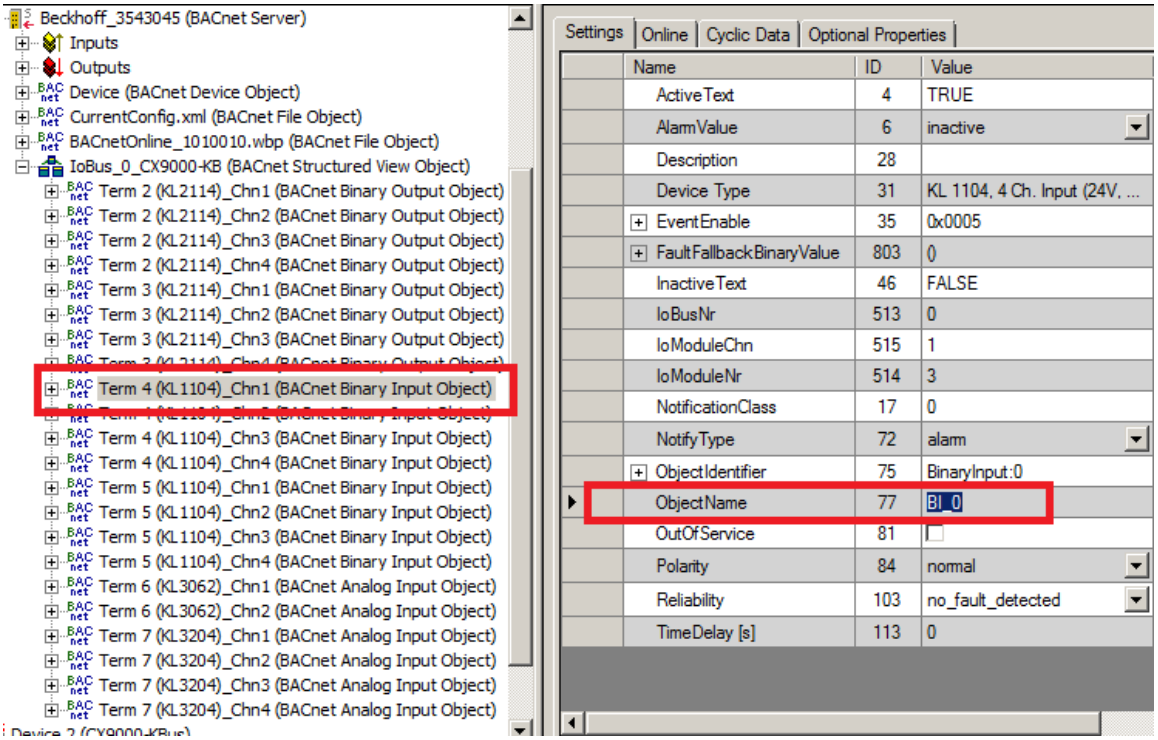


6. Wichtig: Für die richtige Verknüpfung müssen die Objektnamen korrespondierend zum SPS Kommentar vergeben werden (möglich wäre auch die Zuordnung mit Hilfe der Objekt-ID):

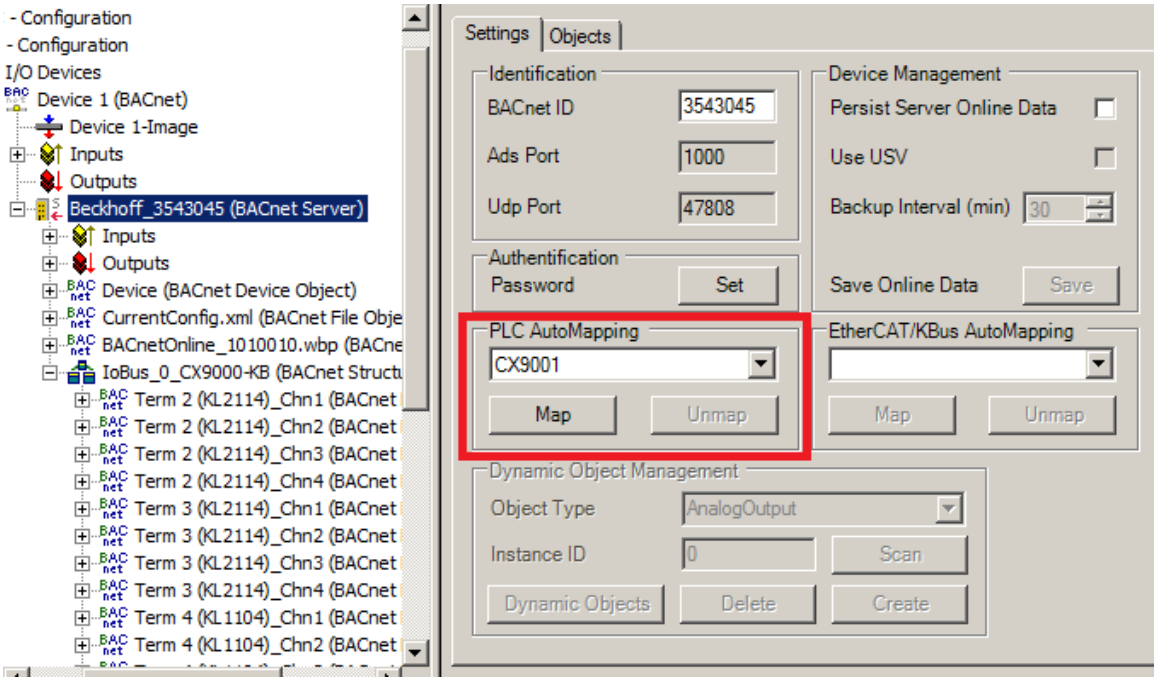
   a) Binärer Ausgang "BinaryOutput: 0" soll mit der SPS-Instanz "BO_0" verknüpft werden:



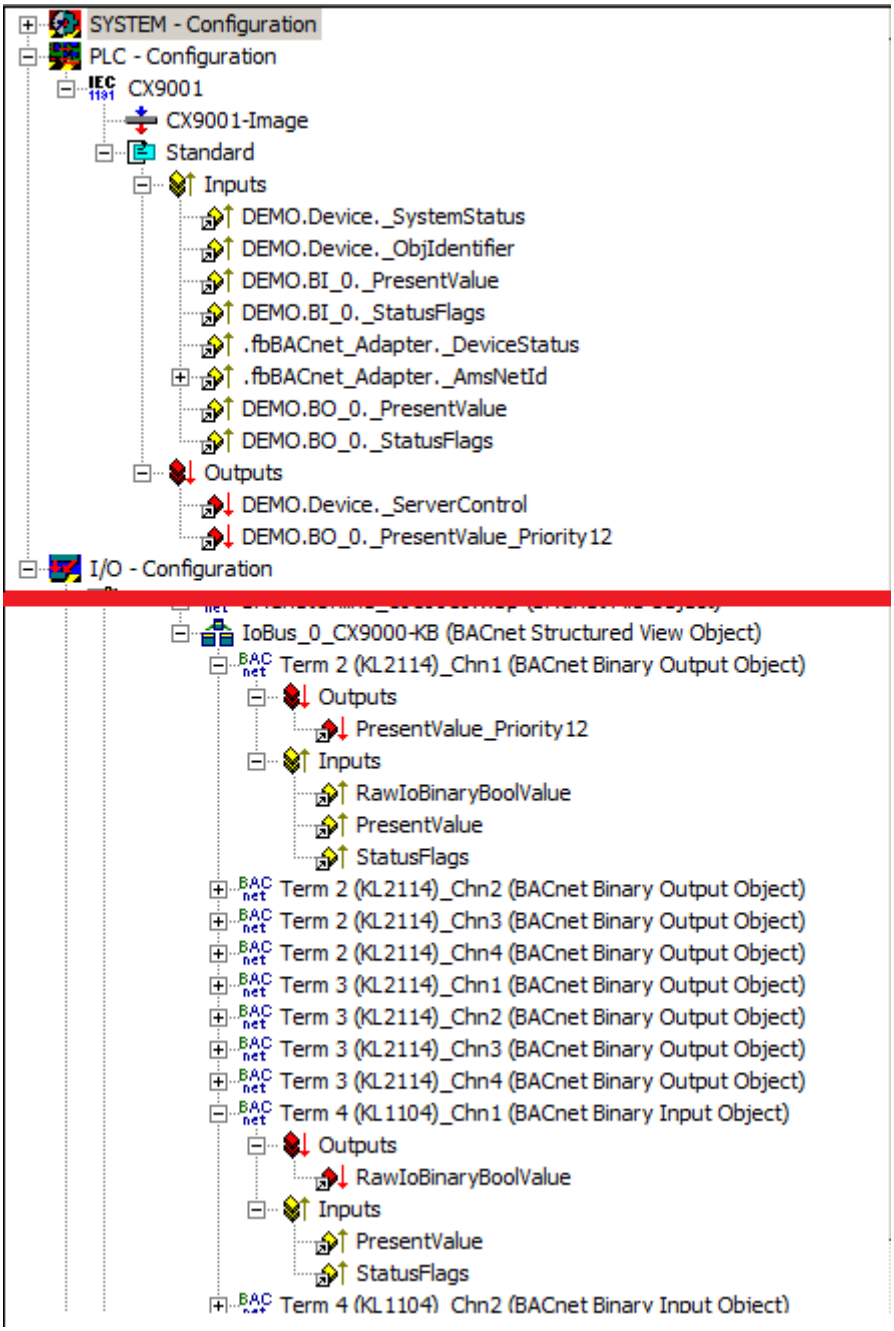
   b) Binärer Eingang "BinaryInput: 0" soll mit der SPS-Instanz "BI_0" verknüpft werden:



7. SPS-Automapping im Reiter "Settings" des BACnet-Servers ausführen:

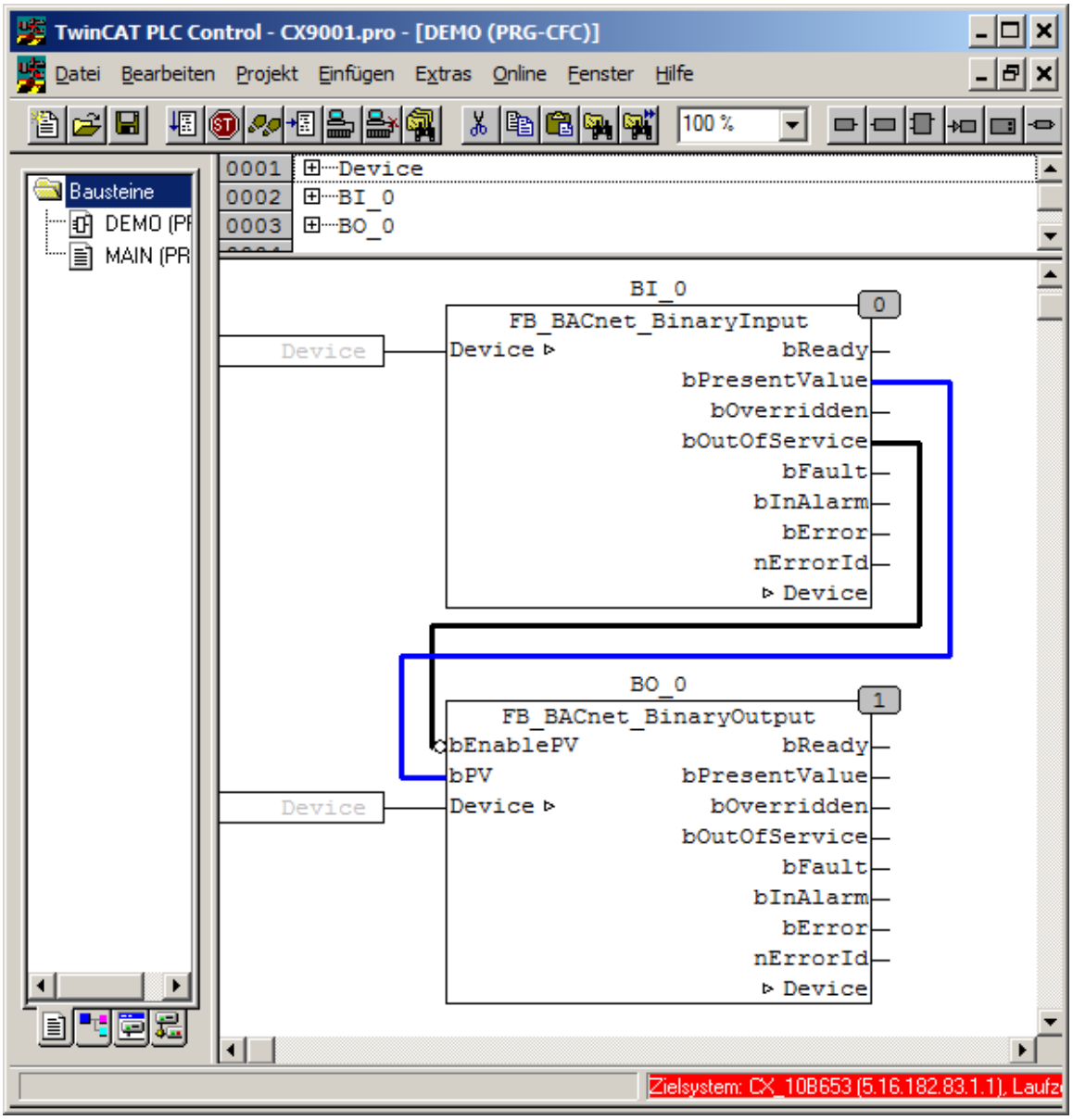


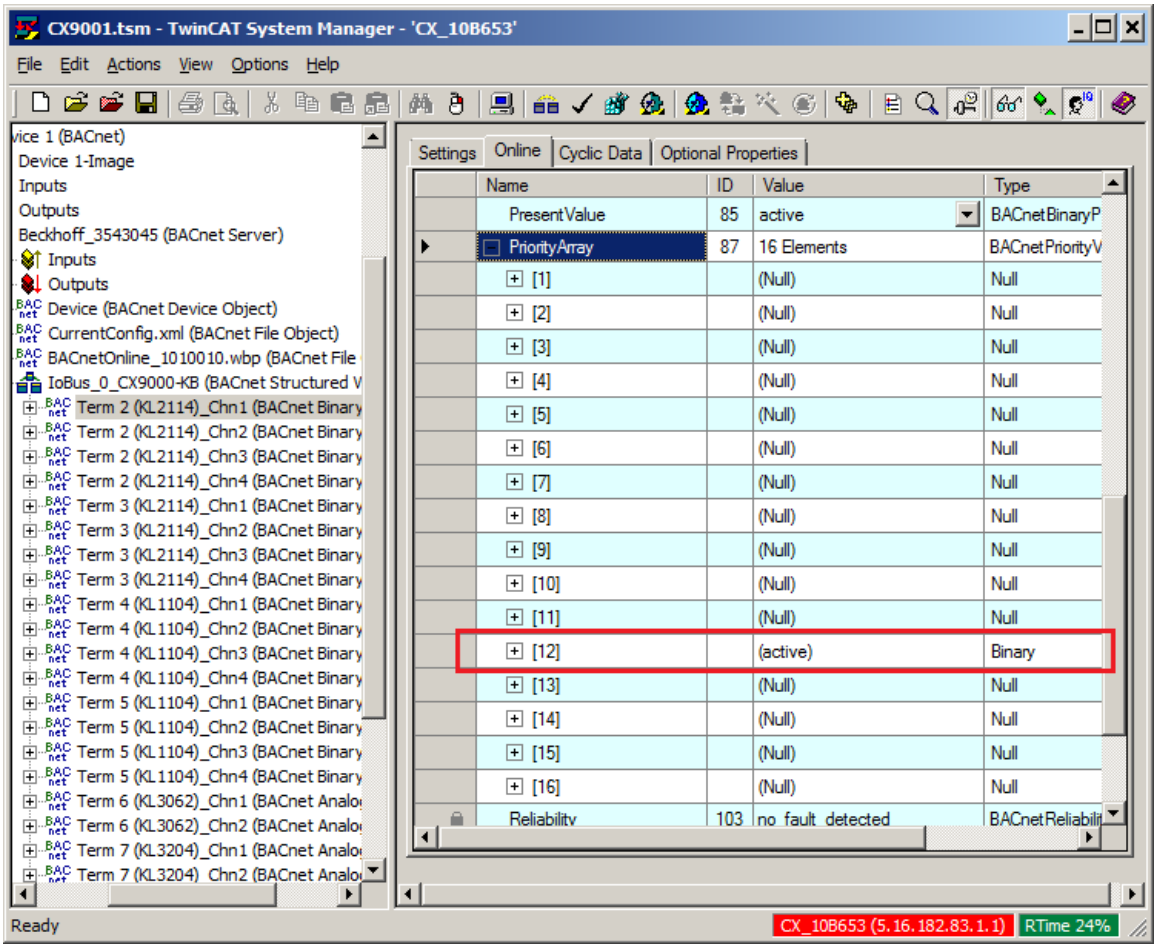
8. Anschließend sind die Prozessdaten der BACnet-Objekte und die korrespondierenden des SPS-Programms verknüpft:



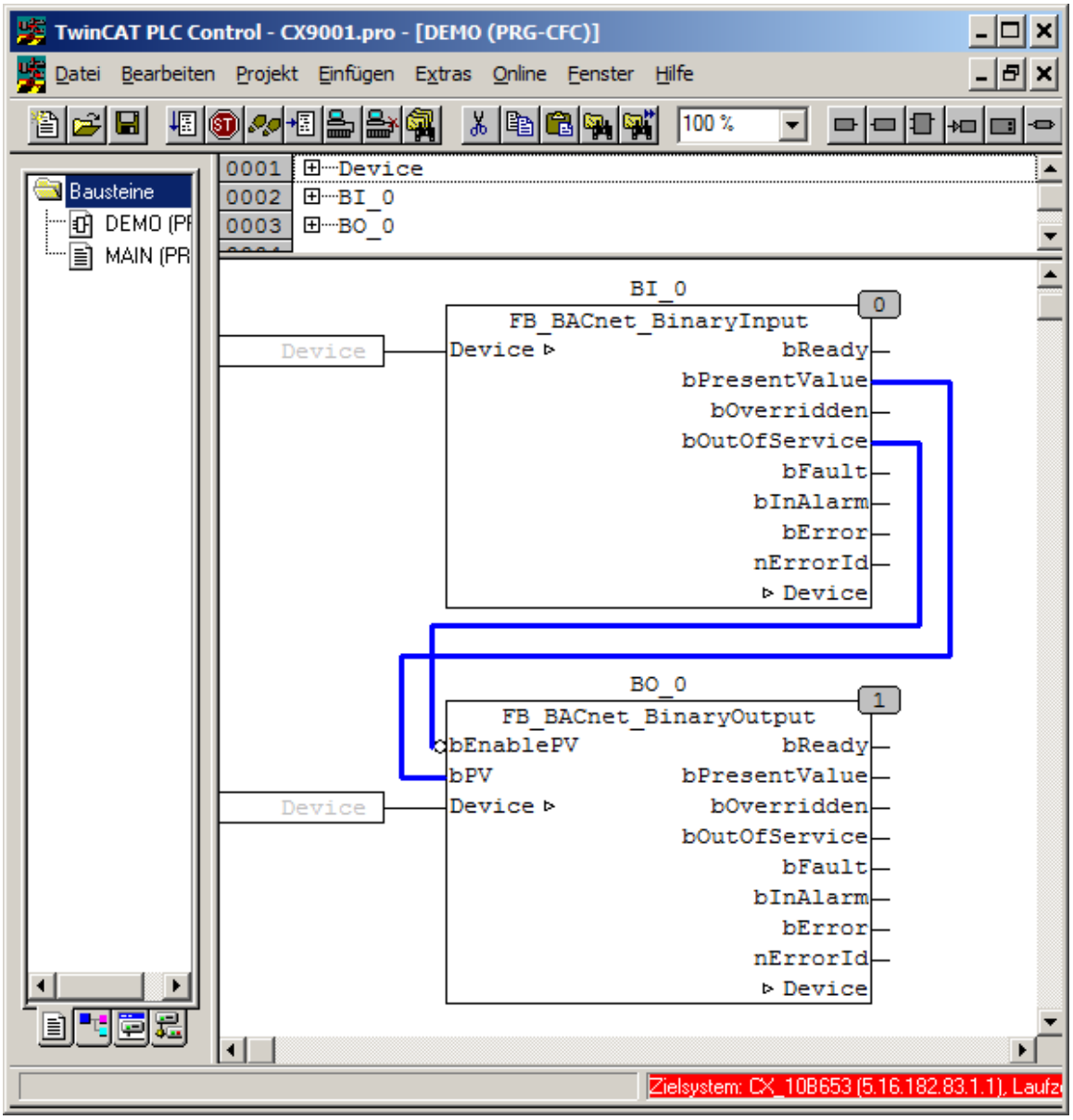
9. Die Konfiguration muss aktiviert (Ctrl+Shift-F4) und das SPS-Programm geladen werden (PLC Control über Menü "Online --> Einloggen" oder F11). Danach wird die SPS in den Run-Zustand mit F5 bzw. "Online --> Start" versetzt.
10. Im Folgenden sind die verschiedenen Zustände der SPS mit den BACnet-Properties gegenübergestellt:

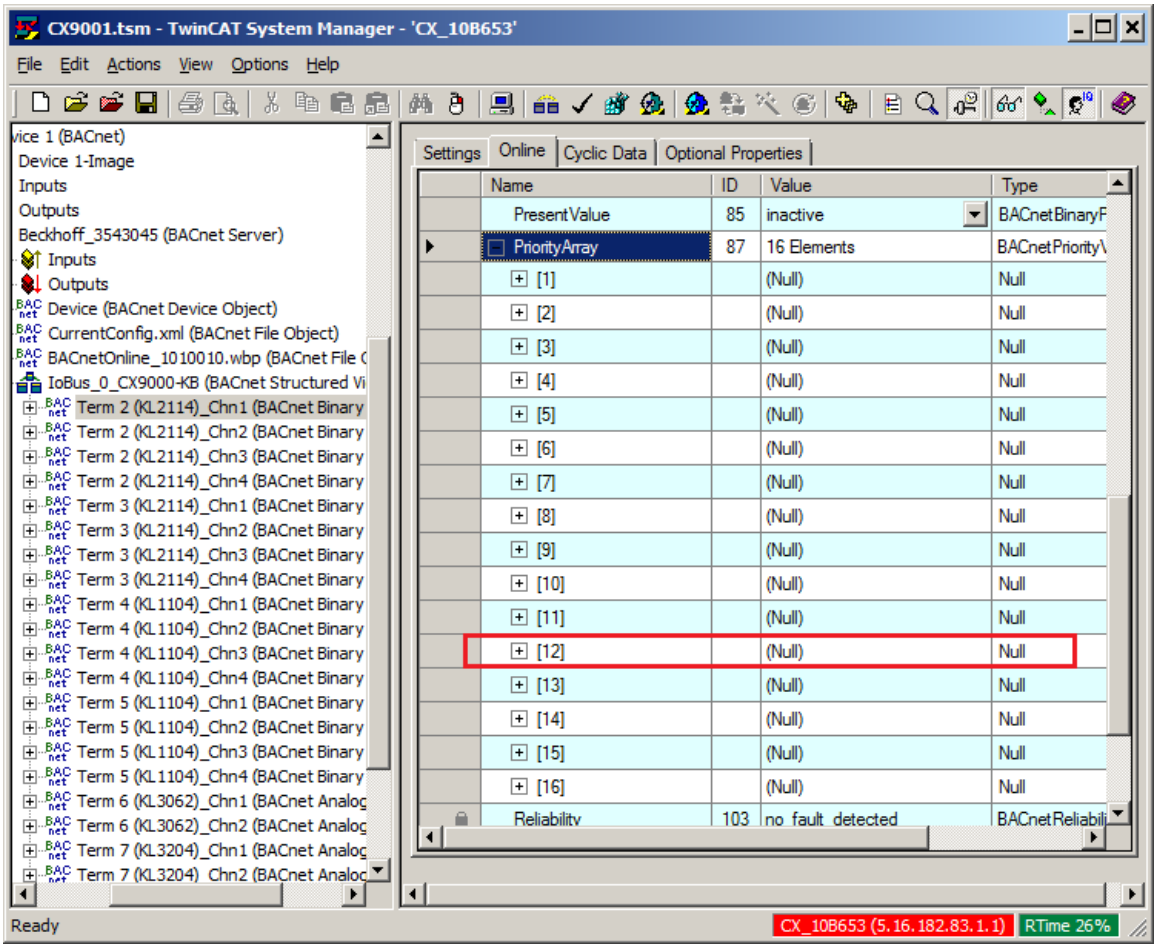
    a) **Fall 1**: "BI_0" ist nicht OutOfService und *ACTIVE* bzw. *INACTIVE* → Priorität "12" von "BO_0" ist auf *ACTIVE* bzw. *INACTIVE* gesteuert und wirkt auf die Hardware-Klemme:


TwinCAT PLC Control - CX9001.pro - [DEMO (PRG-CFC)]
Datei Bearbeiten Projekt Einfügen Extras Online Fenster Hilfe
100 %
Bausteine
DEMO (PR
MAIN (PR
0001 Device
0002 BI_0
0003 BO_0
BI_0
FB_BACnet_BinaryInput
0
Device
Device
bReady
bPresentValue
bOverridden
bOutOfService
bFault
bInAlarm
bError
nErrorId
Device
BO_0
FB_BACnet_BinaryOutput
1
bEnablePV
bPV
Device
Device
bReady
bPresentValue
bOverridden
bOutOfService
bFault
bInAlarm
bError
nErrorId
Device
Zielsystem: CX_10B653 (5.16.182.83.1.1), Laufz



b) **Fall 2**: "BI_0" ist OutOfService → Priorität "12" von "BO_0" ist gelöscht *(Null)* und der Wert aus Property *RelinquishDefault* von "BO_0" wirkt auf die Hardware-Klemme:


TwinCAT PLC Control - CX9001.pro - [DEMO (PRG-CFC)]
Datei Bearbeiten Projekt Einfügen Extras Online Fenster Hilfe
100 %
Bausteine
DEMO (PR
MAIN (PR
0001 Device
0002 BI_0
0003 BO_0
BI_0
FB_BACnet_BinaryInput
0
Device
Device
bReady
bPresentValue
bOverridden
bOutOfService
bFault
bInAlarm
bError
nErrorId
Device
BO_0
FB_BACnet_BinaryOutput
1
bEnablePV
bPV
Device
Device
bReady
bPresentValue
bOverridden
bOutOfService
bFault
bInAlarm
bError
nErrorId
Device
Zielsystem: CX_10B653 (5.16.182.83.1.1), Laufz

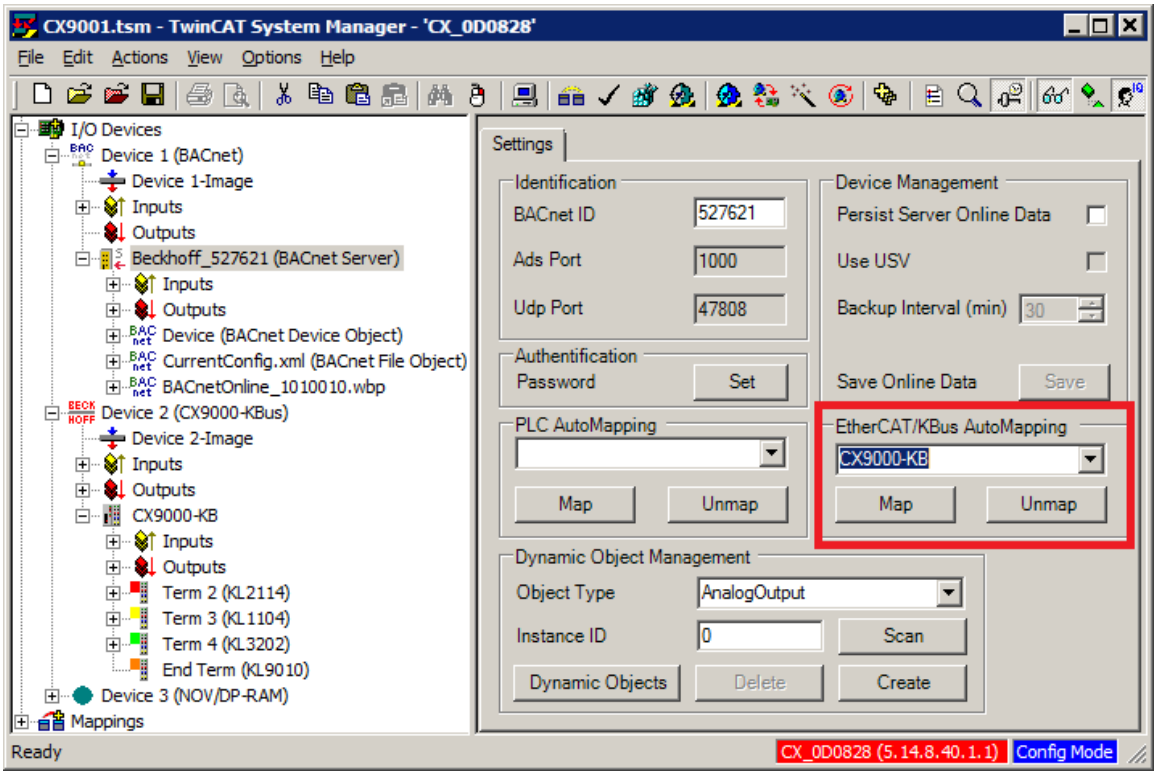


3.4 Beispiel:I/O-Automapping

In einer BACnet-Umgebung werden Daten immer mittels Objekten und deren Properties repräsentiert. Dies gilt für Speicherzustände (Variablen) genauso wie für Hardware-Ein- und -Ausgänge. Im Schluss bedeutet dies, dass sämtliche Hardware-Klemmen des I/O-Systems mittels BACnet-Objekten abgebildet werden müssen. Um die aufwendige Arbeit des Verknüpfens zwischen BACnet-Objekten und Hardware-Klemmen zu minimieren, besteht die Möglichkeit des automatischen Mappings. Im Folgenden wird der Ablauf anhand von Bildschirmabzügen erklärt. Weitere Informationen zum technischen Verständnis gibt es im Kapitel "I/O-Automapping [▶ 75]". Als Beispiel für das manuelle Verknüpfen zwischen Hardware-Klemmen und BACnet-Objekten siehe auch Kapitel "Manuelle Verknüpfung Hardware (Klemme), BACnet BinaryInput und SPS Laufzeit [▶ 97]".

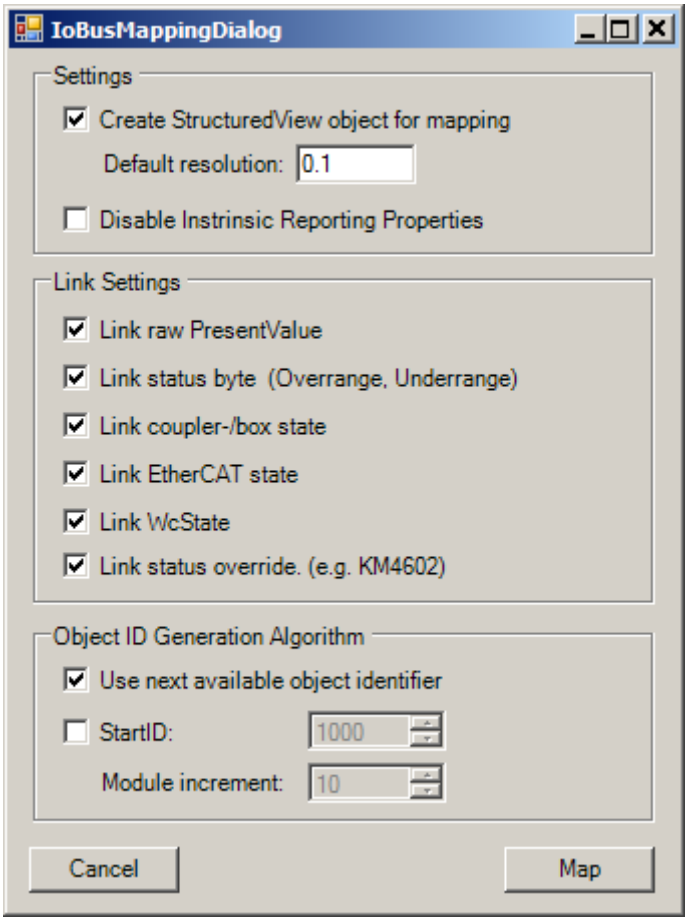
1. BACnet-Adapter und -Server anlegen (siehe Beispiel "BACnet Adapter und Server anlegen [▶ 96]")
2. Einen I/O-Bus mittels Scan-Funktion im Config Modus oder von Hand anlegen (siehe auch Kapitel "I/O-Automapping [▶ 75]")
3. EtherCAT- bzw. K-Bus-Automapping aktivieren

    a) Mittels Dropdown-Box das entsprechende Bus-System anwählen

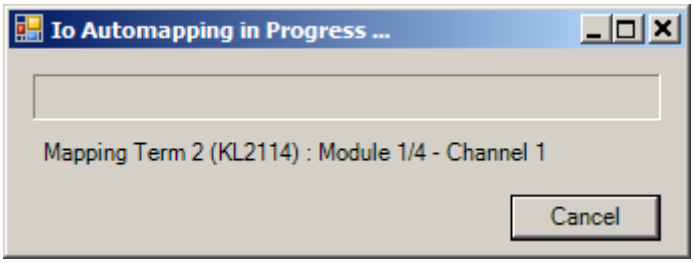


b) Button "Map" betätigen

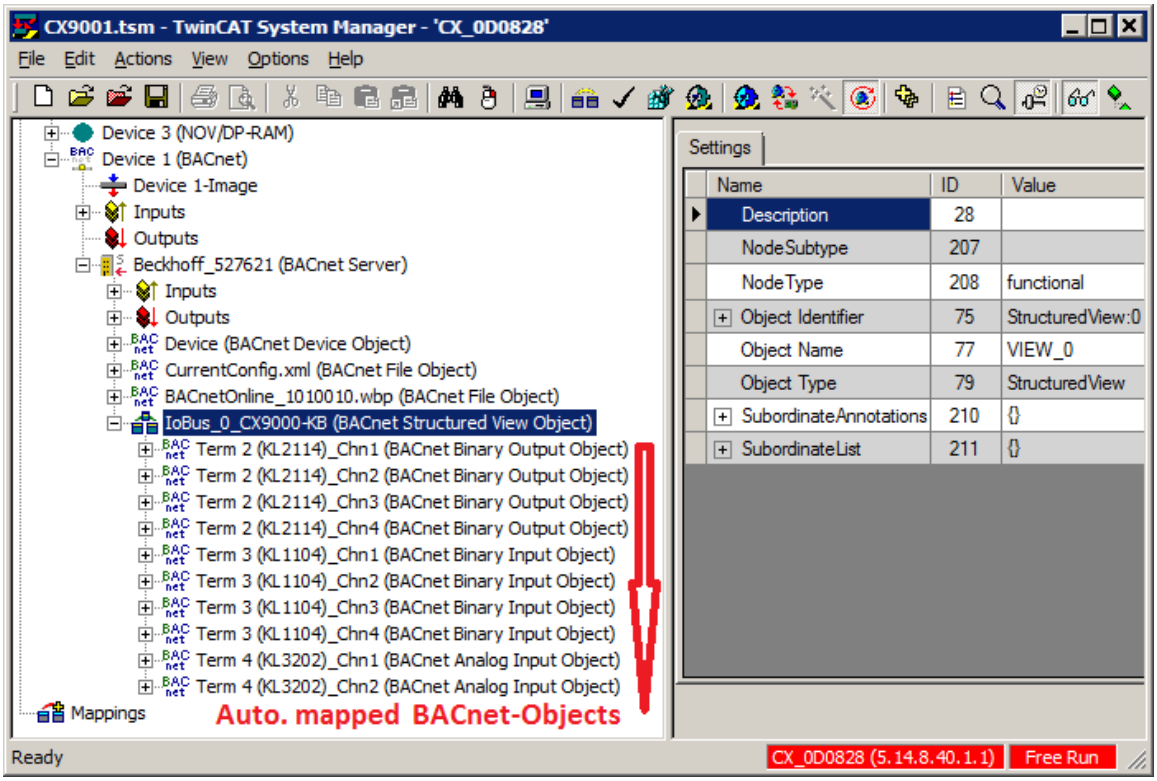
4. Im Automapping-Dialog können die Parameter für das Mapping eingestellt und anschließend mit Button "Map" bestätigt werden



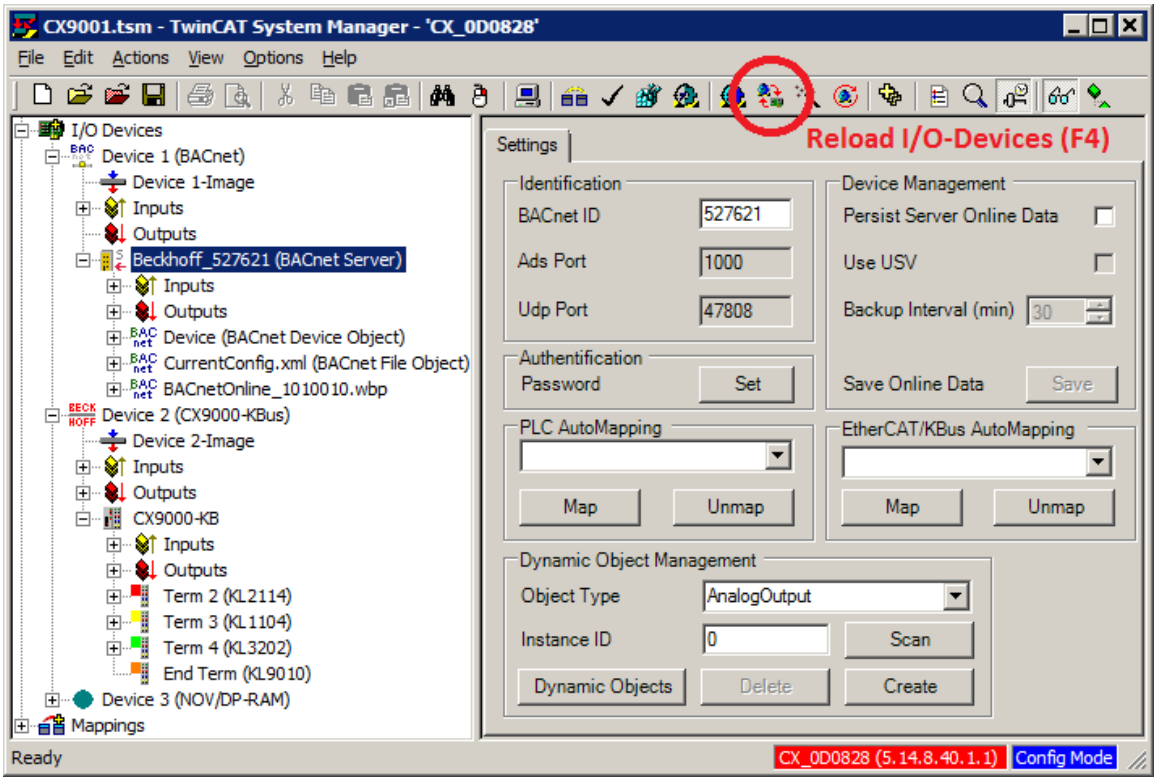
5. Der Fortschritt des Mappings wird in einem separaten Dialog angezeigt



6. Die BACnet-Objekte die die Klemmen des I/O-Buses repräsentieren sind unterhalb einer Structured View angeordnet



7. Nach dem Laden der Konfiguration mit "Reload I/O Devices" und Aktivieren des Free RUN Modus stehen die Objekte "Online" zur Verfügung



3.5 Beispiel: SPS-Automapping

In einer BACnet-Umgebung werden Daten mittels Objekten und deren Properties repräsentiert. Dies gilt für Speicherzustände (Variablen) genauso wie für SPS-Signale, die in BACnet sichtbar sein sollen. Im Schluss bedeutet dies, dass sämtliche Zustände der SPS-Laufzeit, die in BACnet sichtbar sein sollen, mittels BACnet-Objekten abgebildet werden müssen. Um die aufwendige Arbeit des Verknüpfens zwischen BACnet-Objekten und SPS-Signalen zu minimieren, besteht die Möglichkeit des automatischen Mappings. Im Folgenden wird der Ablauf anhand von Bildschirmabzügen erklärt. Weitere Informationen gibt es im Kapitel "SPS-Automapping [▶ 64]".

Das Beispiel https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749044491.zip kann hier https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749044491.zipheruntergeladen werden.

1. BACnet-Adapter und -Server anlegen (siehe "Beispiel: BACnet Adapter und Server anlegen [▶ 96]")
2. Ein SPS-Projekt mit folgenden Baustein-Instanzen anlegen

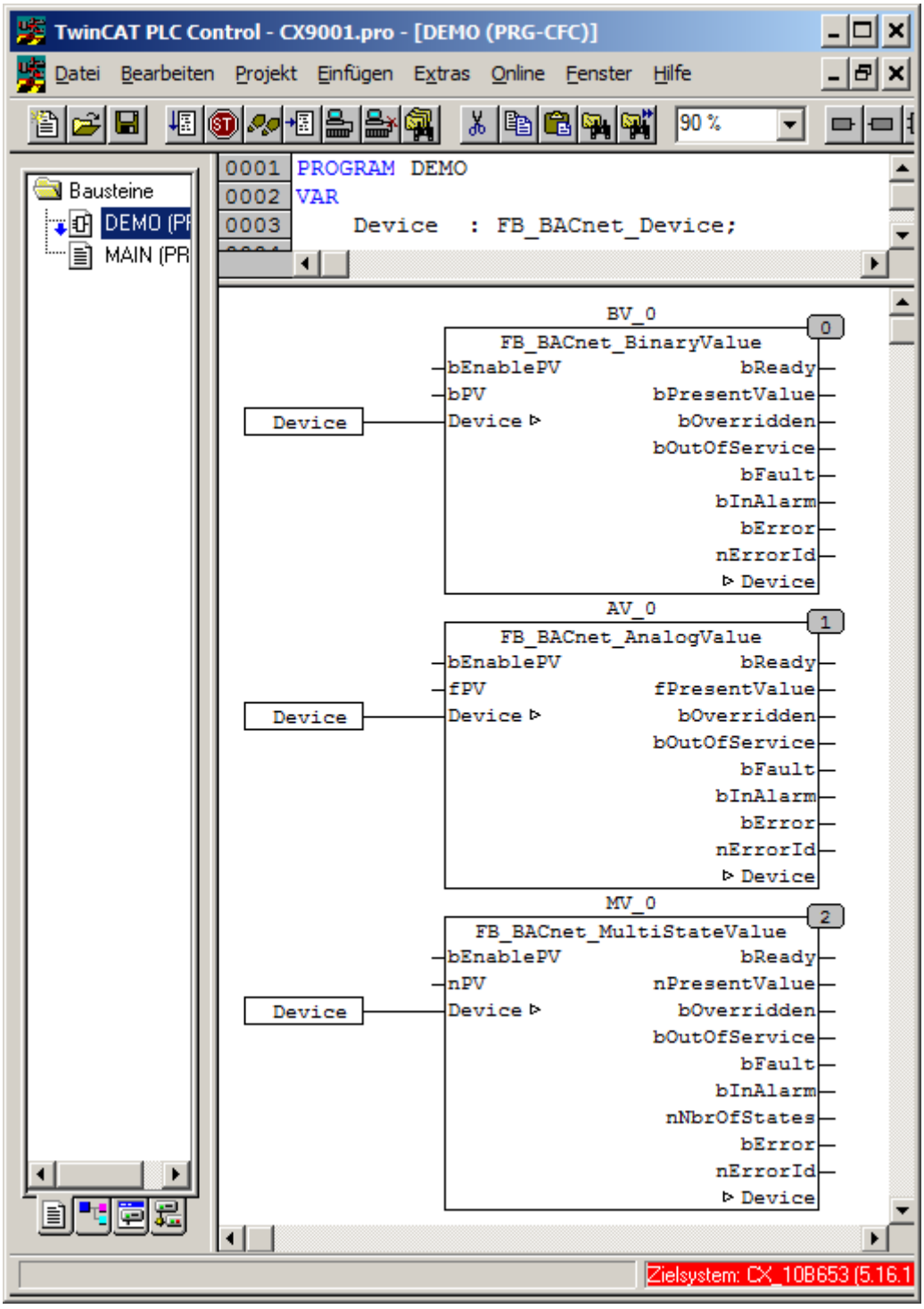
```
0001 PROGRAM DEMO
0002 VAR
0003     Device   : FB_BACnet_Device;
0004     BV_0     : FB_BACnet_BinaryValue;
0005     AV_0     : FB_BACnet_AnalogValue;
0006     MV_0     : FB_BACnet_MultiStateValue;
0007 (* FB_BACnet_Device, FB_BACnet_BinaryValue,
0008     FB_BACnet_AnalogValue and FB_BACnet_MultiStateValue
0009     are implemented by the library "TcBACnet.lib" *)
0010 END_VAR
```

Ziel ist es, die SPS-Instanzen der BACnet-Bausteine als BACnet-Objekt in der System Manager Konfiguration anzulegen, die Prozessdaten, Properties vorzukonfigurieren und die Prozessdaten zwischen SPS und Objekten zu verbinden. Die nötigen Prozessdaten-Definitionen (AT%I* / AT%Q*) sind bereits in den Library-Bausteinen der SPS-Bibliothek enthalten. Die Initialisierung der Objekt-Properties wird im Folgenden beschrieben.

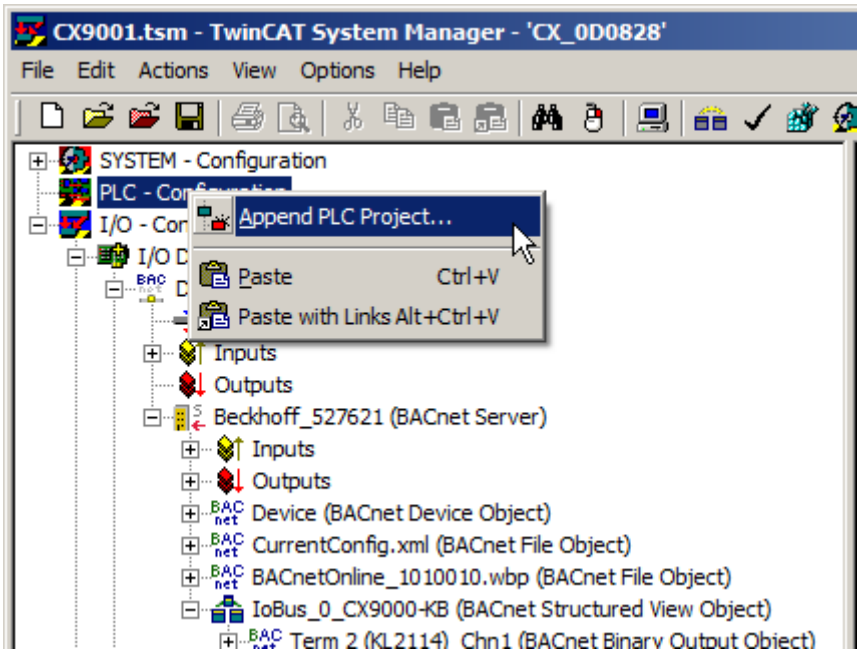
3. Automapping-Kommentare zu den SPS-Instanzen hinzufügen und die Baustein-Instanzen im SPS-Programm aufrufen

```
PROGRAM DEMO
VAR
    Device  : FB_BACnet_Device;

    BV_0    : FB_BACnet_BinaryValue;
    (* ~(BACnet_ObjectName      : DEMO.BV_0 : nolink)
        (BACnet_ObjectIdentifier: 10        : nolink)
        (BACnet_ActiveText      : An        : nolink)
        (BACnet_InactiveText    : Aus       : nolink)
        (BACnet_MinimumOntime   : 2         : nolink)
        (BACnet_MinimumOfftime  : 5         : nolink)
        (BACnet_RelinquishDefault: active   : nolink) *)

    AV_0    : FB_BACnet_AnalogValue;
    (* ~(BACnet_ObjectName      : DEMO.AV_0 : nolink)
        (BACnet_Description     : Ein Test-Objekt. : nolink)
        (BACnet_HighLimit       : 100       : nolink)
        (BACnet_LowLimit        : 0         : nolink)
        (BACnet_LimitEnable     : 0xC006    : nolink)
        (BACnet_Units           : Other_percent : nolink) *)

    MV_0    : FB_BACnet_MultiStateValue;
    (* ~(BACnet_ObjectName      : DEMO.MV_0 : nolink)
        (BACnet_NumberOfStates  : 5         : nolink)
        (BACnet_AlarmValues     : {2;4}     : nolink)
        (BACnet_FaultValues     : {1;5}     : nolink)
        (BACnet_StateText       : {eMin;wMin;OK;wMax;eMax} : nolink) *)

(* FB_BACnet_Device, FB_BACnet_BinaryValue,
    FB_BACnet_AnalogValue and FB_BACnet_MultiStateValue
    are implemented by the library "TcBACnet.lib" *)
END_VAR
```

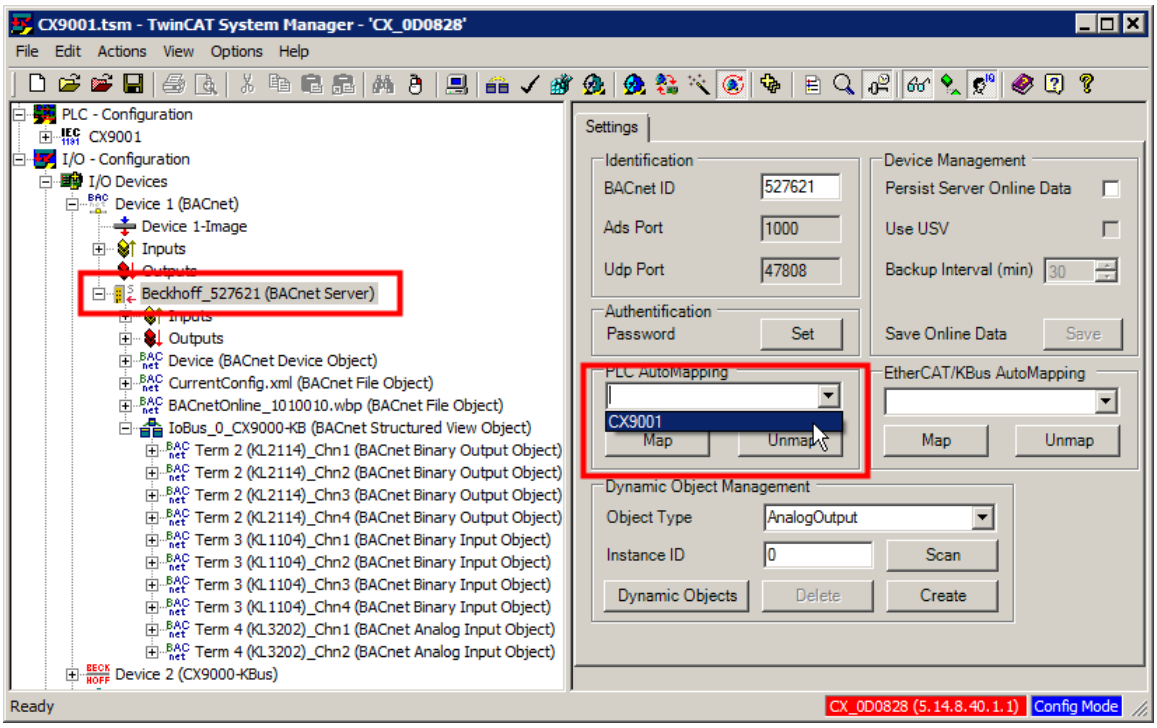
Beachten Sie:
- Das SPS-Programm "DEMO" wird im Programm "MAIN" aufgerufen; Programm "MAIN" ist wiederum als Task in die Taskkonfiguration eingetragen
- Das Testprogramm enthält die Bibliothek "TcBACnet.lib" als Referenz



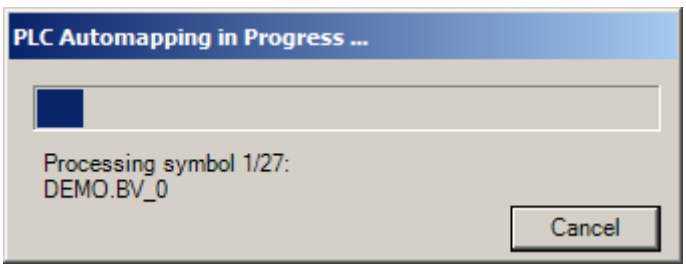
4. SPS-Projekt übersetzten (CTRL+F8)
5. SPS-Projekt zur Hardware-Konfiguration hinzufügen



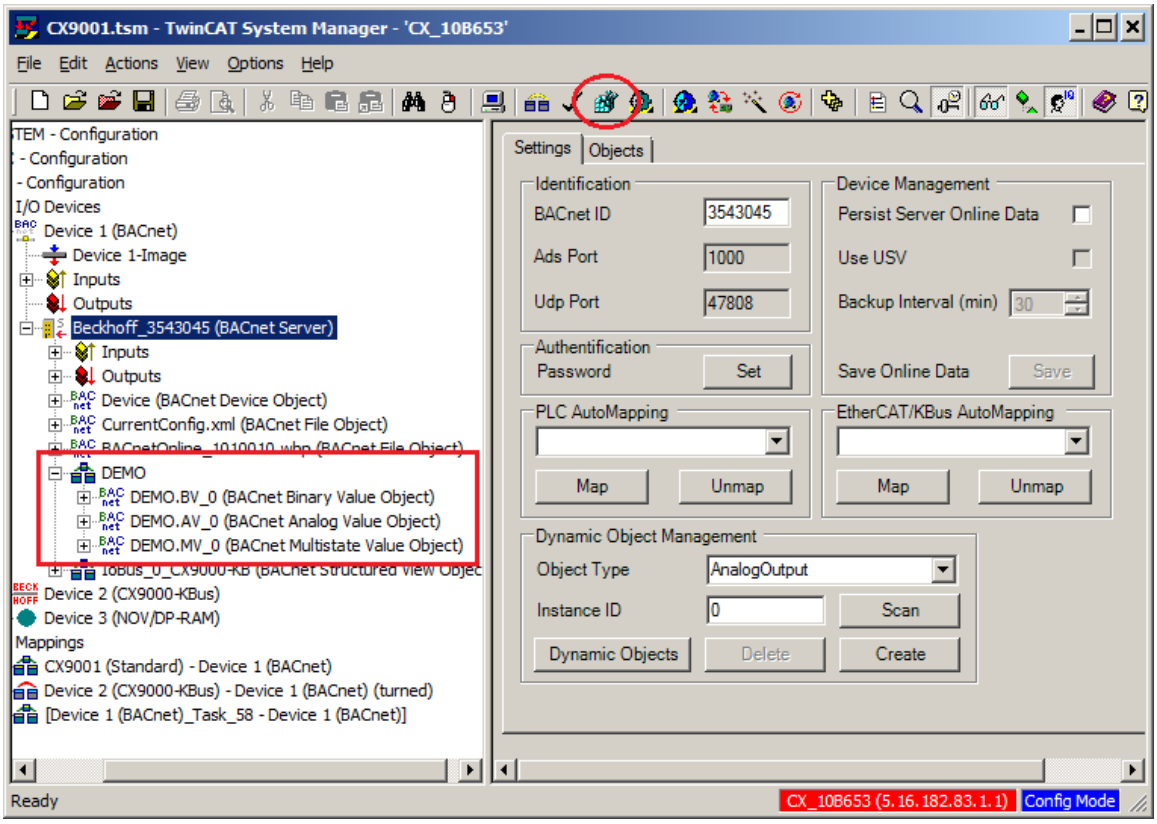
6. Im Reiter "Settings" des BACnet-Servers die SPS-Konfiguration auswählen und mit Button "Map" das Automapping durchführen



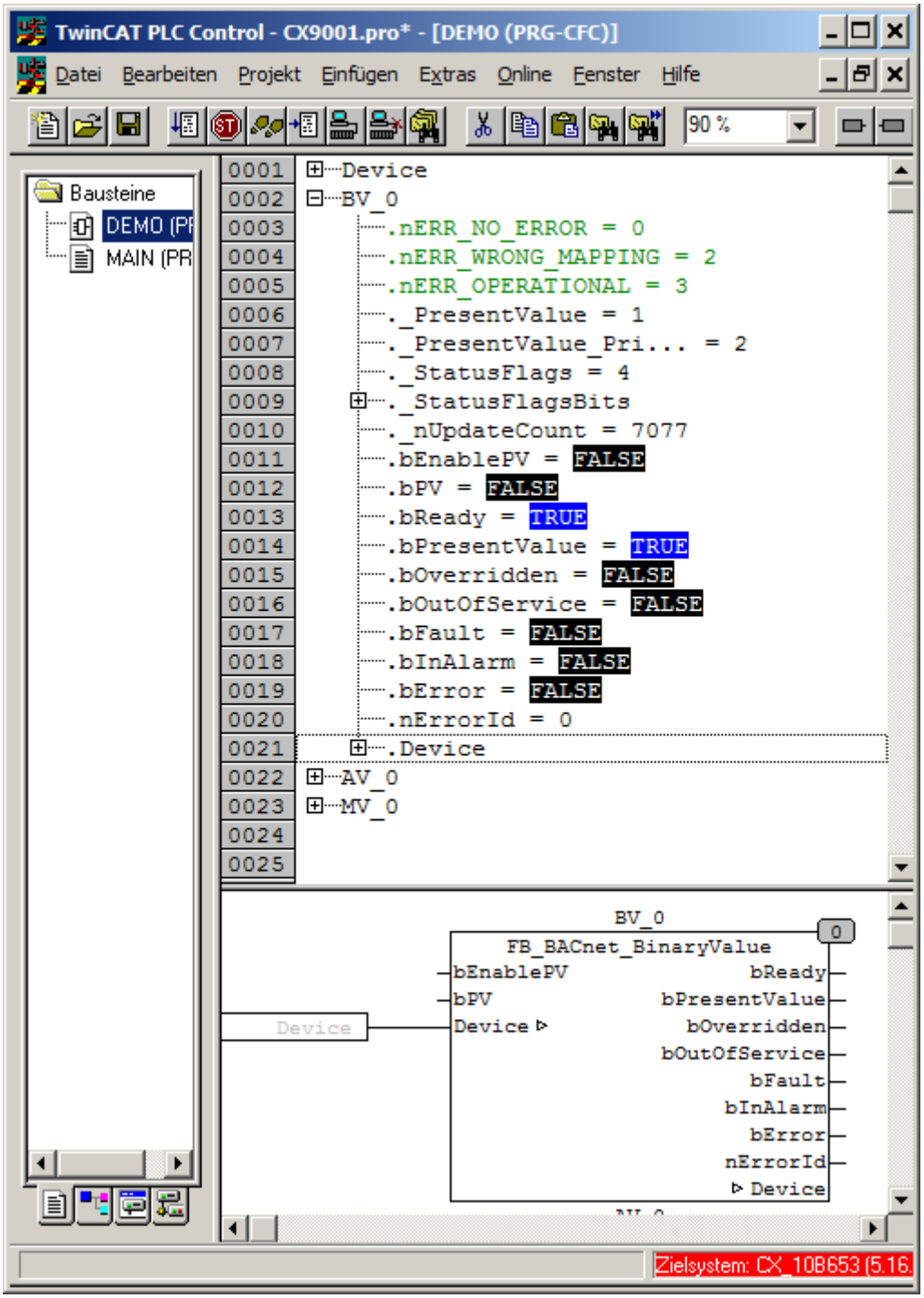
7. Warten bis das Mapping abgeschlossen ist (Dialog mit Fortschrittsbalken erscheint)



8. Die Verknüpfungen sind erstellt. Mit Button "Activate Configuration" aus der Toolbar muss die Konfiguration anschließend aktiviert werden



9. Einloggen in die SPS (F11) und Programm laden (eventuell SPS-Projekt anschließend mit F5 starten)



3.6 Beispiel: BACnet Tag-/Nacht-Schaltung

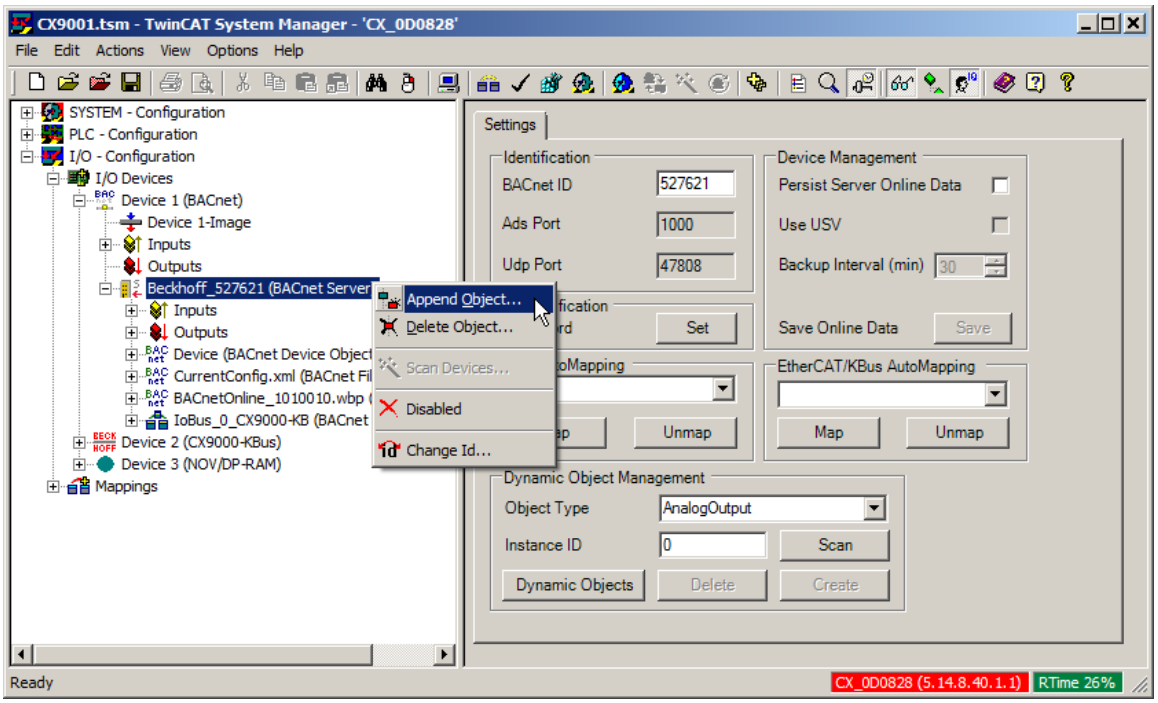
Im Folgenden wird beispielhaft gezeigt wie eine einfache Tag-/Nachtschaltung mit Hilfe eines BACnet-Schedule- und -Output-Objekts erstellt wird. Dabei geht es z.B. um eine Beleuchtungssteuerung, die nachts aktiv ist. Die Schaltzeit zum Aktivieren des binären Ausgangs wird auf 22:00 Uhr gestellt. Ab 4:00 Uhr soll der Ausgang zurückgesetzt werden.

Das Beispiel https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749045899.zip kann hier: https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749045899.zipheruntergeladen werden.

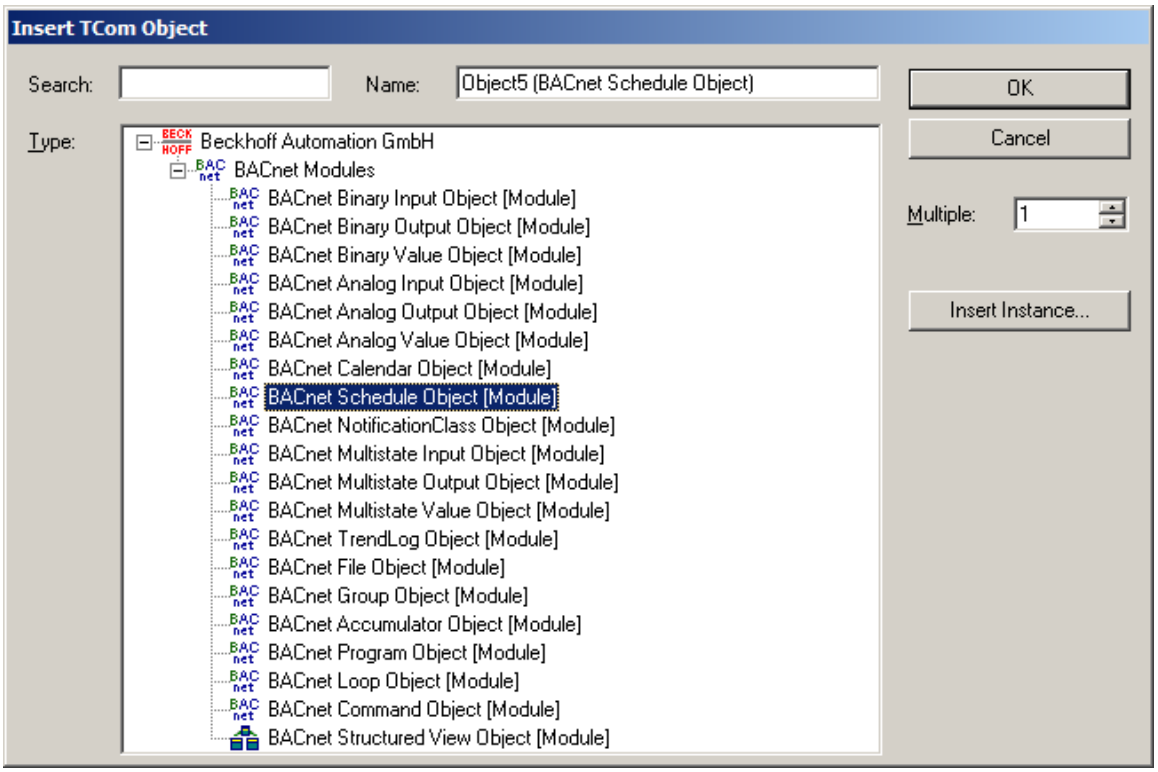
1. BACnet-Adapter und -Server anlegen (siehe: "Beispiel: BACnet Adapter und Server anlegen [▶ 96]")
2. Einen I/O-Bus mittels I/O-Automapping hinzufügen (siehe: "Beispiel: I/O-Automapping [▶ 119]")

3. Ein Schedule-Objekt unterhalb des Servers anlegen

   a) Mit Mausrechtsklick auf den Server und "Append Object..." ein neues BACnet-Objekt anlegen

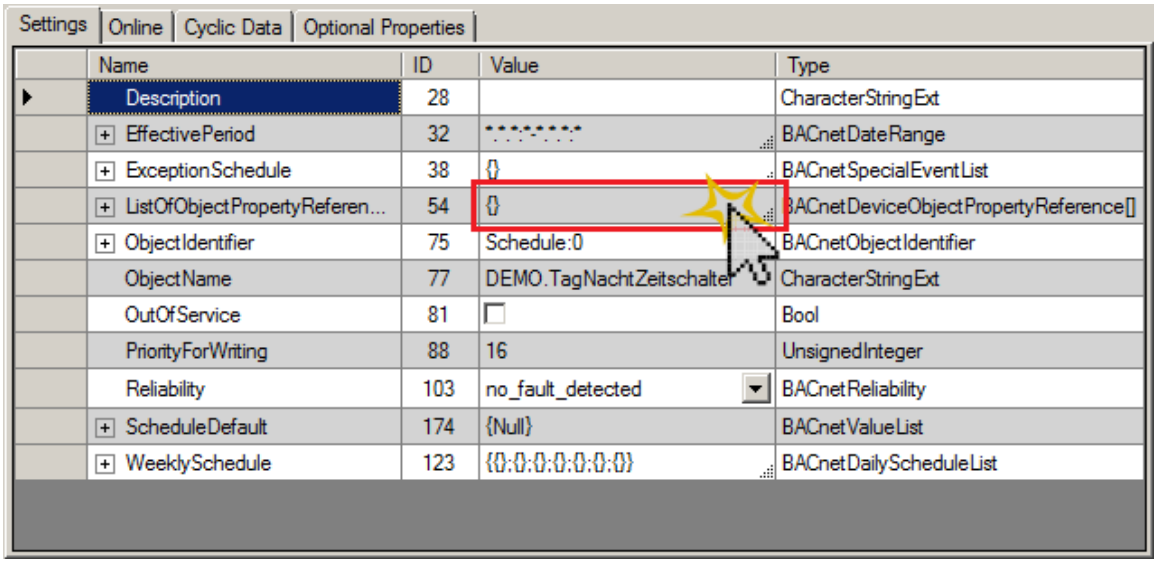


   b) Module "BACnet Schedule Object" auswählen und mit "OK" hinzufügen

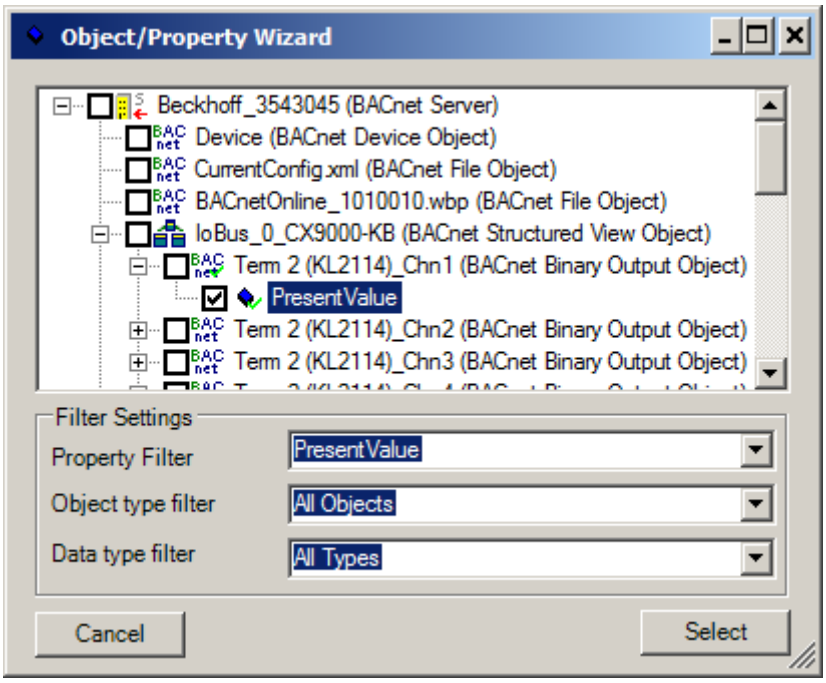


4. Als Objekt-Referenz für das Schreiben des Schedule-Werts (*ACTIVE*/*INACTIVE*) wird das Objekt vom Typ BACnetBinaryOuput, das den binären Ausgang für die Beleuchtungssteuerung repräsentiert, eingetragen:

   a) Dazu genügt ein Doppelklick auf die Property *ListOfObjectReferences*:

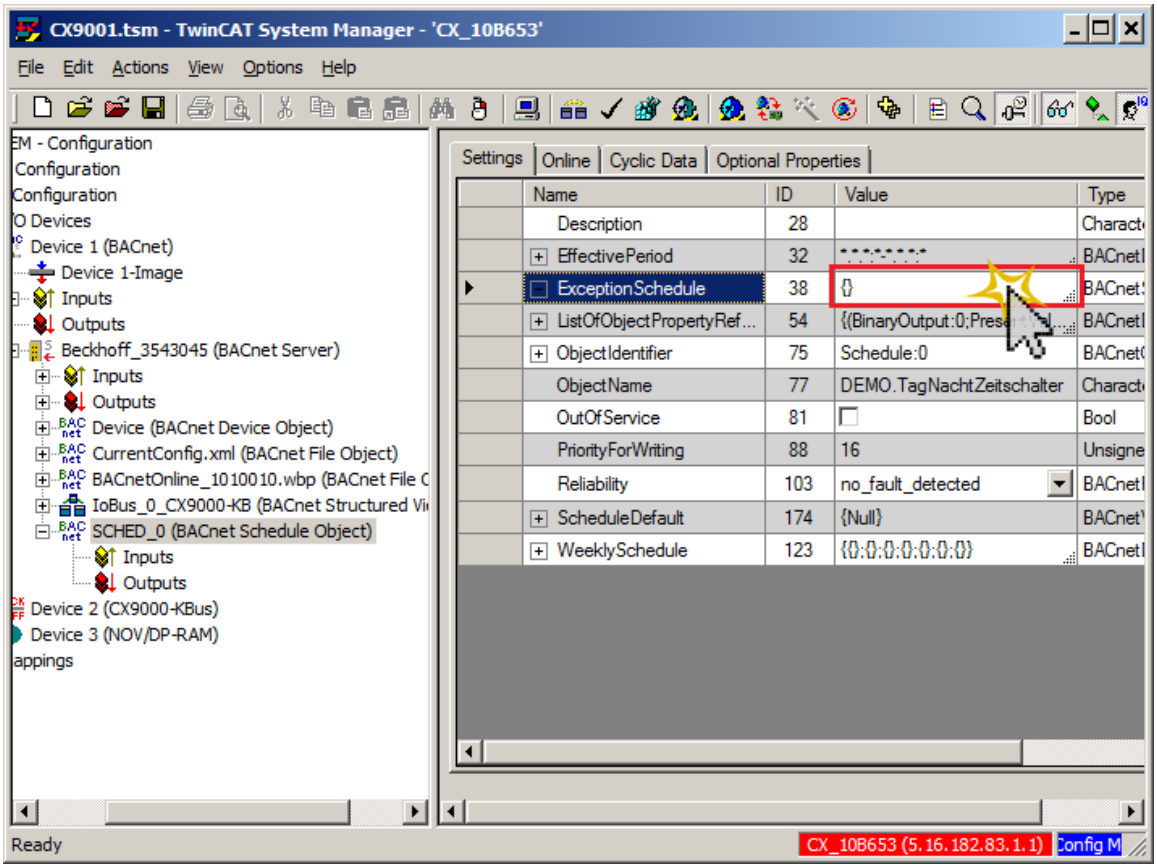


b) Dann Anwahl des *PresentValue* des gewünschten *BinaryOutput*-Objekts:

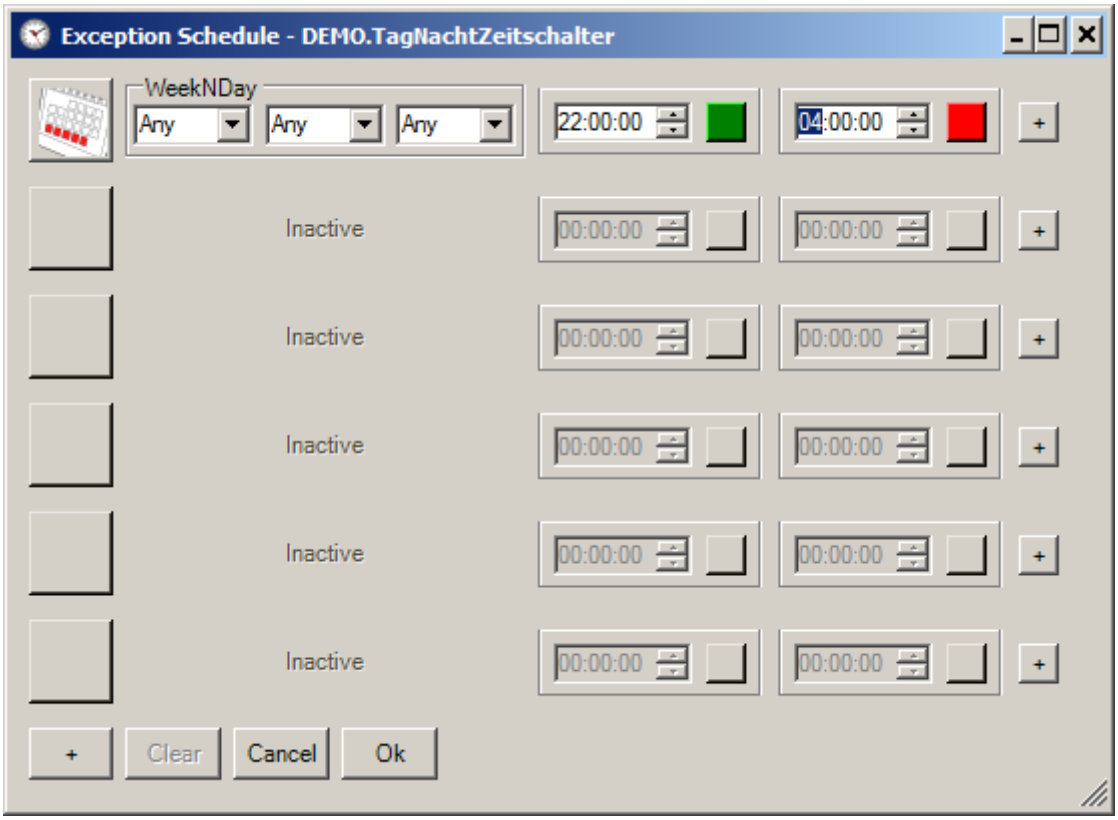


Die Auswahl der Objektreferenz ist für die weiteren Schritte wichtig, da sich dadurch der Datentyp der TimeValue-Einträge in der Property ExceptionSchedule für den folgenden Wizard ergibt. Bei Anwahl des PresentValue eines Binary*-Objekts wird der Datentyp BACnetBinaryPV andernfalls Bool hinzugefügt.

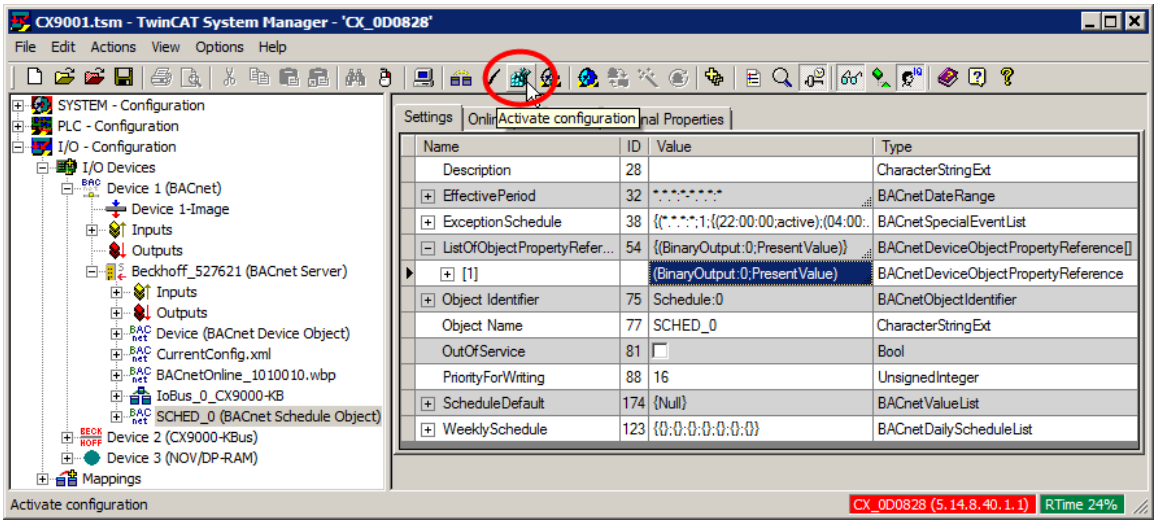
5. Doppelklick auf die Property ExceptionSchedule:



6. Mit Hilfe des Wizards können die Ein- und Ausschaltzeiten komfortabel konfiguriert werden:



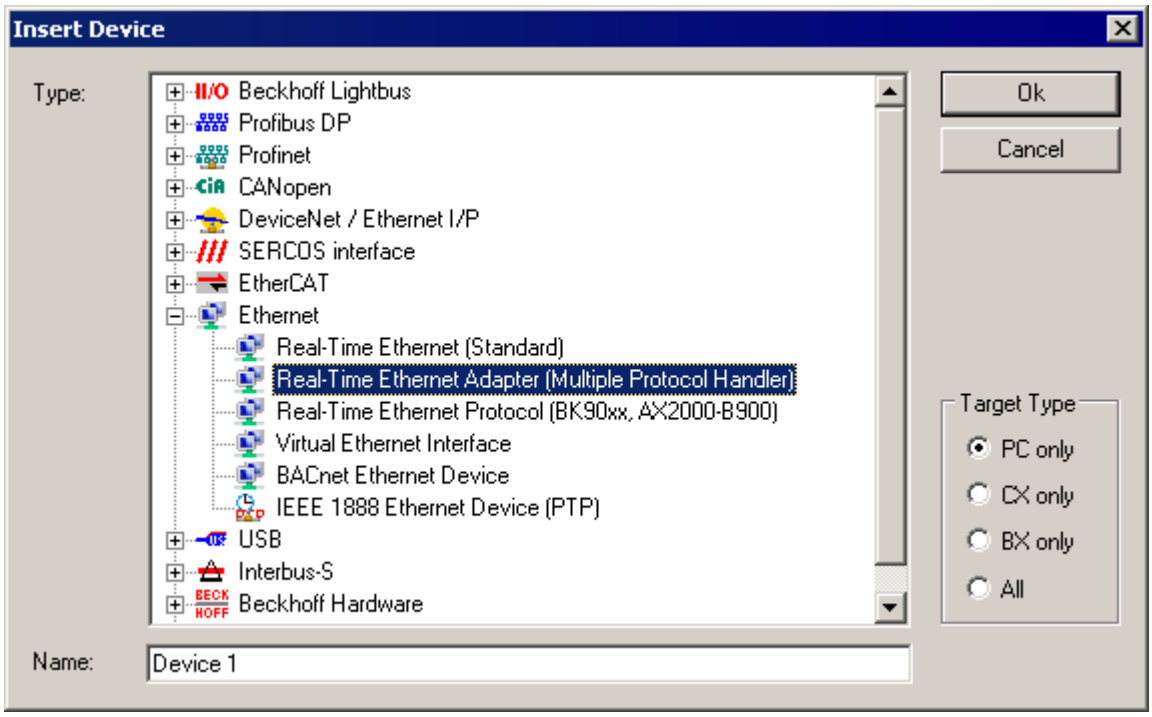
7. Nun muss die Konfiguration auf das Zielsystem, durch Klick auf "Active configuration" aus der Toolbar, geladen werden:



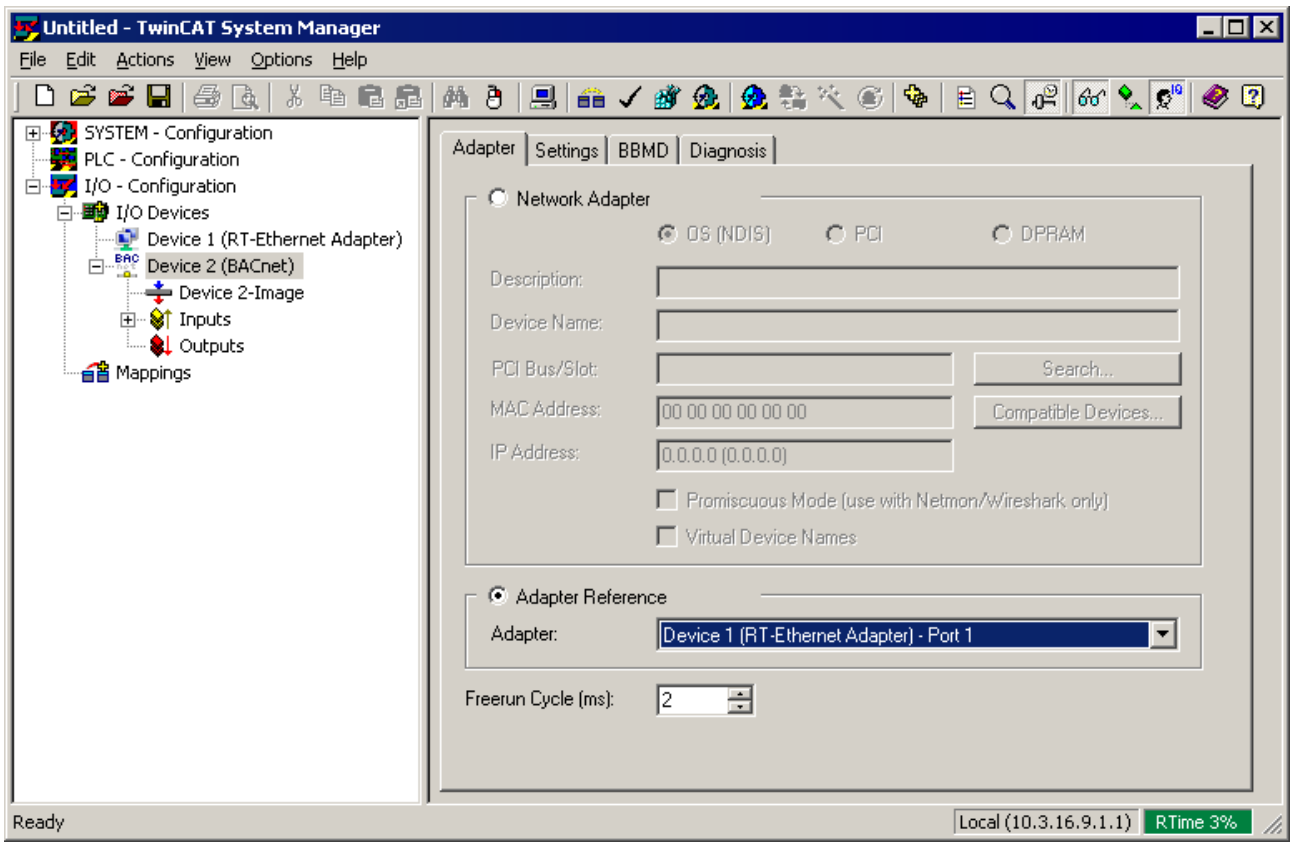
3.7 Beispiel: Betrieb von BACnet und BK90XX über eine Ethernet-Schnittstelle

Im Folgenden wird erläutert, wie ein BACnet-Device parallel mit einem bzw. mehreren BK90XX (Real-Time Ethernet) über eine Ethernet-Schnittstelle betrieben werden kann.

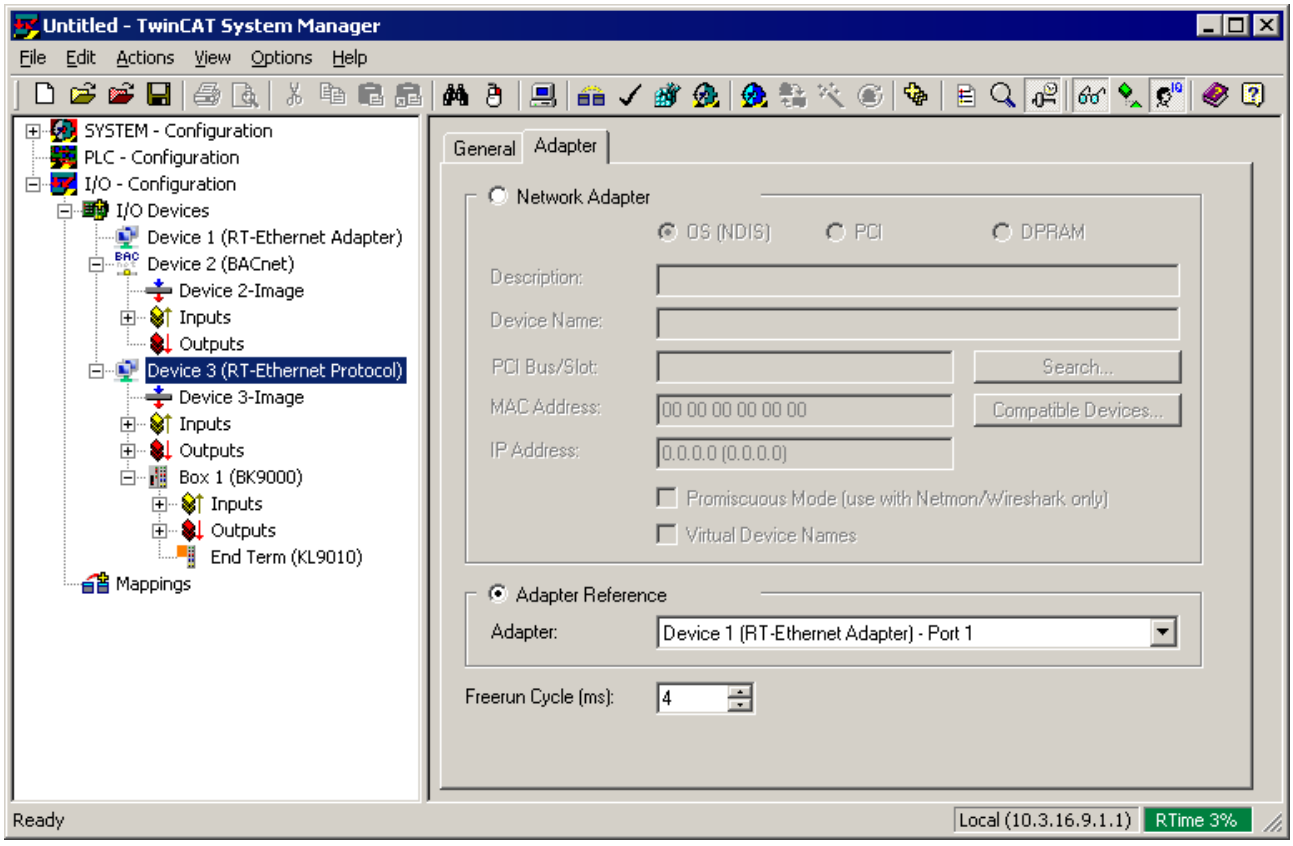
Zunächst wird ein **Real-Time Ethernet-Adapter (Multi Protocol Handler)** angelegt und mit der gewünschten Ethernet-Schnittstelle verbunden:



Anschließend wird ein BACnet-Device angelegt und als Ethernet-Adapter **Port 1** des Multi Protocol Handler verwendet:



Analog wird beim Real-time Ethernet Protocol verfahren:

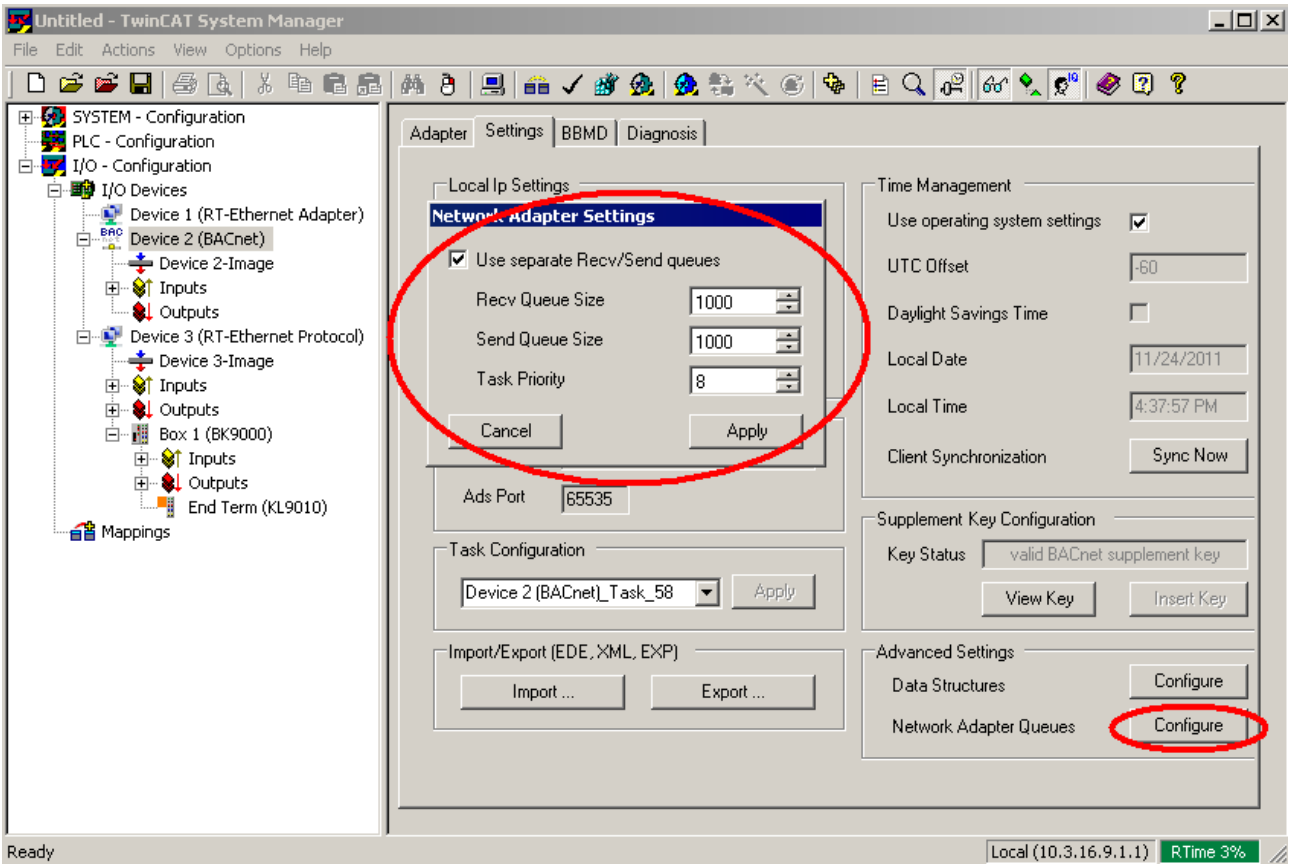


Nun können BACnet und Real-Time Ethernet parallel über die gleiche Ethernet-Schnittstelle betrieben werden. (z.B. Einscannen der Geräte usw.)

Wird das Real-Time Ethernet Protocol Device nicht via BACnet, sondern durch ein synchrones Mapping z.B. von einer SPS getrieben, müssen im BACnet-Device die Netzwerk-Queues aktiviert werden. Der Grund hierfür ist, dass BACnet-interne Funktionen nur durch die BACnet-Task aufgerufen werden dürfen. Ein

synchrones Mapping des Real-Time Ethernet Protocol Device führt dazu, das BACnet im Kontext der verknüpften Task aufgerufen wird. Die Netzwerk-Queues entkoppeln in diesem Fall BACnet-interne Funktionen von dieser Task.

Die Netzwerk-Queues können im "Settings"-Dialog des BACnet-Device aktiviert werden:



Mit Hilfe des beschriebenen Verfahrens können auch mehrere BACnet-Server (Devices) auf einer Netzwerkschnittstelle konfiguriert werden. Dabei muss für jeden BACnet-Server ein BACnet-Device angelegt und mit dem Port des Multi Protocol Handler verbunden werden. Mehrere BACnet-Server können so parallel betrieben werden - die IP-Adresse muss dann für jedes zugehörige BACnet-Device angepasst werden, da die BACnet-Server anhand der IP Adresse unterschieden werden.

3.8 Beispiel: NotificationClass und NotificationSink

Im Folgenden wird ein Beispiel zur Verwendung der *NotificationClass* in Zusammenspiel mit einer *NotificationSink* gegeben. Diese "sammelt" die Events von 3 Objekten (BV, AV, MV) und sendet diese an den lokalen Prozess 10 (*NotificationSink*) des lokalen BACnet-Device (Event-Notification mittels Intrinsic-Reporting).

Zudem soll das *PresentValue* des *AnalogValue*-Objekts bei Wertänderung von 0.1 an die *NotificationSink* gesendet werden (COV-Notification).

Das Beispiel https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749047307.zip kann hier https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749047307.zipheruntergeladen werden. Das enthaltene TSM File kann auch ohne Laden der SPS zum Testen verwendet werden.

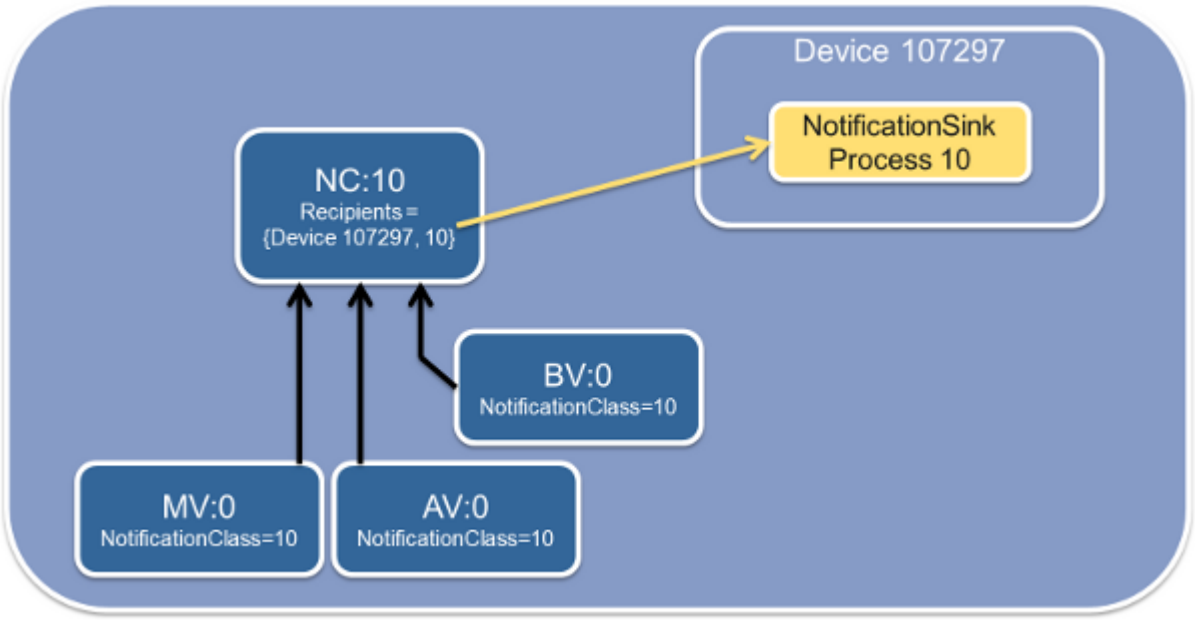


Abb. 1: Bild-1: Schematische Darstellung zur Beispielkonfiguration der NotificationClass, deren Event-generierenden Objekte (AV, MV, BV) und der NotificationSink.

Die Konfiguration der Objekte erfolgt durch die Deklaration im SPS Programm:

```
VAR
 viewDummy : BOOL; (* ~( BACnet_StructuredViewPath : \/DemoNClass : ) *)

 fbDevice : FB_BACnet_Device;

 fbBV_0 : FB_BACnet_BinaryValue;
  (* ~(BACnet_ObjectIdentifier : 0 : nolink )
  (BACnet_EventEnable : {to_offnormal;to_fault;to_normal} : nolink )
  (BACnet_ActiveText : EIN : nolink )
  (BACnet_InactiveText : AUS : nolink )
  (BACnet_NotificationClass : 10 : nolink ) *)

 fbAV_0 : FB_BACnet_AnalogValue;
  (* ~(BACnet_ObjectIdentifier : 0 : nolink )
  (BACnet_EventEnable : {to_offnormal;to_fault;to_normal} : nolink )
  (BACnet_NotificationClass : 10 : nolink ) *)

 fbMV_0 : FB_BACnet_MultiStateValue;
  (* ~(BACnet_ObjectIdentifier : 0 : nolink )
  (BACnet_EventEnable : {to_offnormal;to_fault;to_normal} : nolink )
  (BACnet_NotificationClass : 10 : nolink )
  (BACnet_NumberOfStates : 5 : nolink )
  (BACnet_StateText : ErrLow;WarnLow;Normal;WarnHigh;ErrHigh : nolink )
  (BACnet_FaultValues : 1;5 : nolink )
  (BACnet_AlarmValues : 2;4 : nolink )
  (BACnet_Description : MV DEMO Objekt. : nolink ) *)

 fbNC_10 : FB_BACnet_NotificationClass;
  (* ~(BACnet_ObjectIdentifier : 10 : nolink )
  (BACnet_Priority : {70;80;90} : nolink )
  (BACnet_AckRequired : {to_offnormal;to_fault;to_normal} : nolink )
  (BACnet_RecipientList :
      <ArrayOfBACnetDestination>
    <BACnetDestination>
      <recipient>
        <choice>device</choice>
        <BACnetObjectIdentifier>
          <objId>33661729</objId>
        </BACnetObjectIdentifier>
      </recipient>
      <validDays>65025</validDays>
      <fromTime>
        <hour>0</hour>
        <minute>0</minute>
        <second>0</second>
        <hundredths>0</hundredths>
      </fromTime>
      <toTime>
```

```
        <hour>23</hour>
        <minute>59</minute>
        <second>59</second>
        <hundredths>99</hundredths>
      </toTime>
      <processIdentifier>10</processIdentifier>
      <transitions>57349</transitions>
      <issueConfirmedNotifications>false</issueConfirmedNotifications>
    </BACnetDestination>
      </ArrayOfBACnetDestination>
  : nolink ) *)END_VAR
```

Die zum Empfang der Events eingefügte wird im System Manager konfiguriert: *NotificationSink*

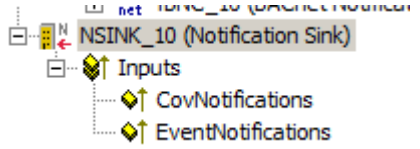


Abb. 2: Bild-2: NotificationSink in der System Manager Baumansicht

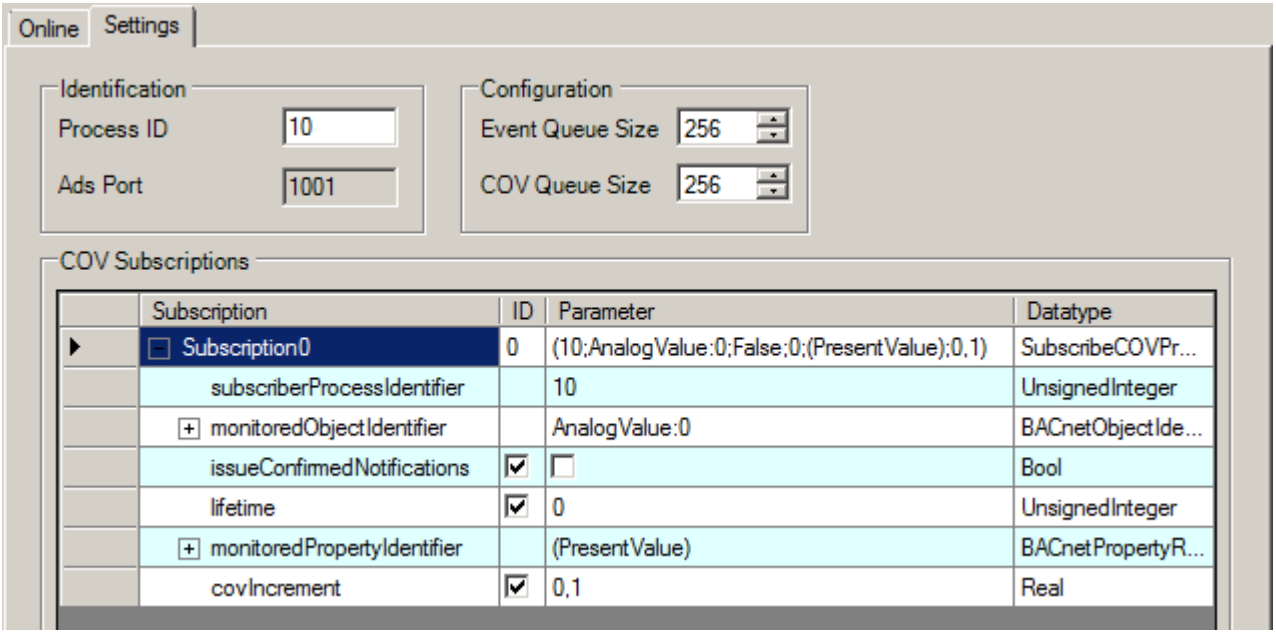


Abb. 3: Bild-3: NotificationSink Konfiguration im System Manager

Process ID: Die Prozessidentifikation wird auf 10 eingestellt und der *NotificationClass* als Empfangs-Prozess mitgeteilt (via SPS-Automapping Kommentar).

COV Subscriptions: Als Beispiel wird das *PresentValue* des *AnalogValue*-Objekts (AV:0) abonniert. Der Parameter *covIncrement* gibt die nötige Wertänderung an bei deren Erreichen eine COV-Notification versendet wird.

Folgende Einträge werden im Online-Reiter der angezeigt, nachdem die Konfiguration geladen und die 's der jeweiligen Objekte verändert wurden: *NotificationSink PresentValue*

Notifications

Nr	Time	Source	Param
COVNotification_0	12.06.2012 14:20:53	(10;Device:107297;AnalogValue:0;0;{(Pres...	COV...
COVNotification_1	12.06.2012 14:21:19	(10;Device:107297;AnalogValue:0;0;{(Pres...	COV...
COVNotification_2	12.06.2012 14:21:24	(10;Device:107297;AnalogValue:0;0;{(Pres...	COV...
COVNotification_3	12.06.2012 14:21:27	(10;Device:107297;AnalogValue:0;0;{(Pres...	COV...
COVNotification_4	12.06.2012 14:21:30	(10;Device:107297;AnalogValue:0;0;{(Pres...	COV...
EventNotification_0	12.06.2012 14:20:53	(10;Device:107297;MultiStateValue:0;12.0...	Eve...
EventNotification_1	12.06.2012 14:20:53	(10;Device:107297;BinaryValue:0;12.06.20...	Eve...
EventNotification_2	12.06.2012 14:21:11	(10;Device:107297;BinaryValue:0;12.06.20...	Eve...
EventNotification_3	12.06.2012 14:21:13	(10;Device:107297;BinaryValue:0;12.06.20...	Eve...
EventNotification_4	12.06.2012 14:21:57	(10;Device:107297;MultiStateValue:0;12.0...	Eve...
EventNotification_5	12.06.2012 14:22:03	(10;Device:107297;MultiStateValue:0;12.0...	Eve...
EventNotification_6	12.06.2012 14:22:05	(10;Device:107297;MultiStateValue:0;12.0...	Eve...
EventNotification_7	12.06.2012 14:22:08	(10;Device:107297;MultiStateValue:0;12.0...	Eve...
processIdentifier		10	Unsi...
initiatingDeviceId...		Device:107297	BAC...
eventObjectIdent...		MultiStateValue:0	BAC...
timeStamp		12.06.2012 14:22:08	d...
notificationClass		10	Unsi...
priority		80	Unsi...
eventType		change_of_state	BAC...
messageText	☑	MV DEMO Objekt.	Char...
notifyType		alarm	BAC...
ackRequired	☑	☑	Bool
fromState	☑	offnormal	BAC...
toState		fault	BAC...
eventValues	☑	((49156;11;5))	c...

Abb. 4: Bild-4: NotificationSink Online-Daten nachdem COV- und Event-Notifications empfangen wurden.

Der Text unter messageText wird aus der Property Description des Event-generierenden Objekts gebildet.

3.9 Häufig gestellte Fragen (FAQ)

- BACnet-Client liefert keine aktuellen Prozessdaten [▶ 137]
  - Wie kann das Neustarten entfernter Server erkannt werden?
  - Wie kann der Status entfernter Server überwacht werden?
  - Wie kann die maximale Anzahl Subscriptions, die der Server pro Objekt zulässt, erhöht werden?
- Beim Stopp/Herunterfahren des TwinCAT-Systems werden die Prozessdaten einiger BACnet-Objekte zurückgesetzt [▶ 139]
- Das Schreiben von Daten bei Wertänderung zu einem entfernten Server (Write On Change) soll kurzzeitig deaktiviert werden? [▶ 140]
- Wie kann das Versenden von Alarm-Events bei BinaryOutput-/-Input- bzw. -Value-Objekten unterbunden werden? [▶ 140]

- Wie kann eine NULL (NaN) in das Priority-Array der kommandierbaren Property PresentValue eines AnalogOutput-Objekts von der SPS über die Prozessdaten geschrieben werden? [▶ 140]
- Warum ist das Prozessdatum PresentValue eines Binary*-Objekts vom Typ WORD [▶ 140]
- Der K-Bus eines BACnet-Controllers fällt sporadisch, ohne ersichtlichen Grund aus. Was könnten mögliche Ursachen sein? [▶ 140]
  - Wie kann der K-Bus-Task unabhängig vom BACnet-Task getriggert werden?
- Warum hat die Property StatusFlags eines Objekts den Wert 0x0004 bzw. die Property EventEnable den Wert 0x0005 obwohl keinerlei Statusbits der Property gesetzt sind? [▶ 142]
- Ein BACnet-Client dessen Prozessdaten mit der SPS verbunden sind wird im System-Manager immer als "Offline" angezeigt. Bei einem Scan wird der Client jedoch angezeigt. Was könnte das Problem sein? [▶ 142]
  - Wie können die Eingangsprozessdaten (Read) eines Clients von COV auf Polling umgestellt werden?
- Das PLC Automapping ist wesentlich langsamer als üblich. Was könnte die Ursache sein? [▶ 143]

I. BACnet-Client liefert keine aktuellen Prozessdaten

Ein BACnet-Client liefert keine aktuellen Prozessdaten, die via COV abonniert wurden an die SPS (Prozessdaten auf Client-Seite via Mapping mit der SPS verbunden; das Objekt selbst läuft auf einem entfernten Server). Was könnten mögliche Ursachen sein?

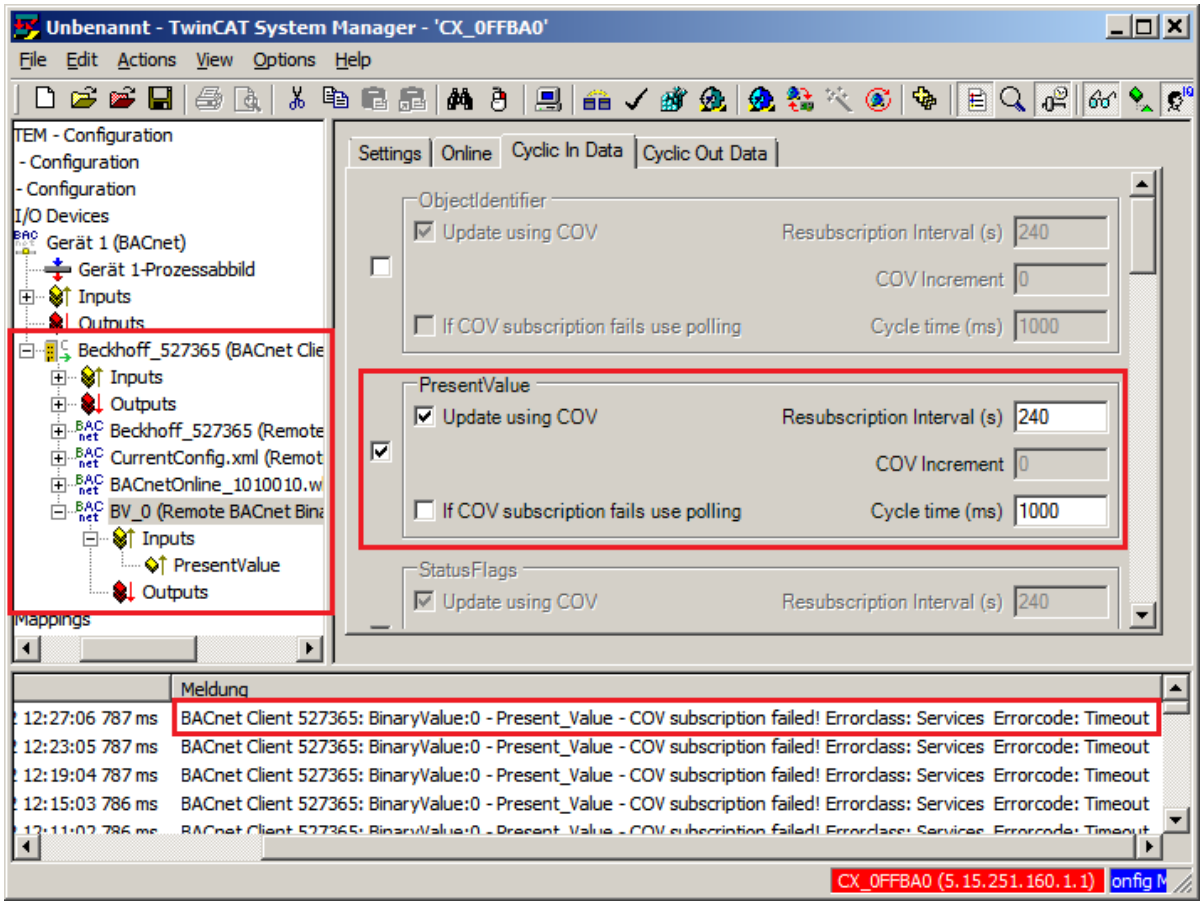
1. Der entfernte Server, auf den der Client zugreift wurde neugestartet (Reboot, TwinCAT Neustart oder Steuerspannungsausfall). Sobald der Server wieder läuft, müssen alle COV-Subscriptions vom Client neu angemeldet werden oder die Zeit die unter Resubscription Interval definiert wurde abgelaufen sein (Resubscription Interval, siehe auch Kapitel "Prozessdaten [▶ 43]" unter "Properties als Prozessdaten").
   - **Wie kann das Neustarten entfernter Server erkannt werden?**

     Unter einem Client kann das Prozessdatum SystemStatus gemapped werden. Das Mapping des SystemStatus sollte dabei auf Polling mit einer möglichst kurzen Periode (zwischen 1..5s) gestellt werden. Wird nun der entfernte Server abgeschaltet spricht zuerst das Timeout der Polling-Anfrage an (Server nicht erreichbar, Client-Statusbit 1 = TRUE → Quittierung mittels ClientControl). Beginnt der entfernte Server mit dem Neustart nimmt der SystemStatus den Zustand "4" (non-operational) an. Wechselt der SystemStatus von "4" auf "0" ist der entfernte Server für Anfragen bereit, so dass COV-Subscriptions neu angemeldet werden können. Das Anmelden der COV-Subscriptions kann mittels ClientControl Bit 3 gesteuert werden.Für weitere Details, siehe Kapitel "Prozessdaten" unter Abschnitt "BACnet Client".
2. Die Netzwerkverbindung zwischen Client und Server ist unterbrochen.
   - **Wie kann der Status entfernter Server überwacht werden?**

     Siehe Punkt 1. Zudem kann mittels Device Status des lokalen BACnet-Adapters der "linked" Zustand des Netzwerkadapters überwacht werden (siehe Kapitel "Prozessdaten" unter Abschnitt "BACnet Device").
3. Der entfernte Server unterstützt kein COV von Properties (COV-P).

   Siehe auch Punkt IX [▶ 142].

4. Der entfernte Server unterstützt COV-P, jedoch ist die maximale Anzahl von Subscriptions auf das entsprechende Objekt serverseitig überschritten. Im Loggerfenster des System Managers erscheint auf Clientseite folgende Meldung:

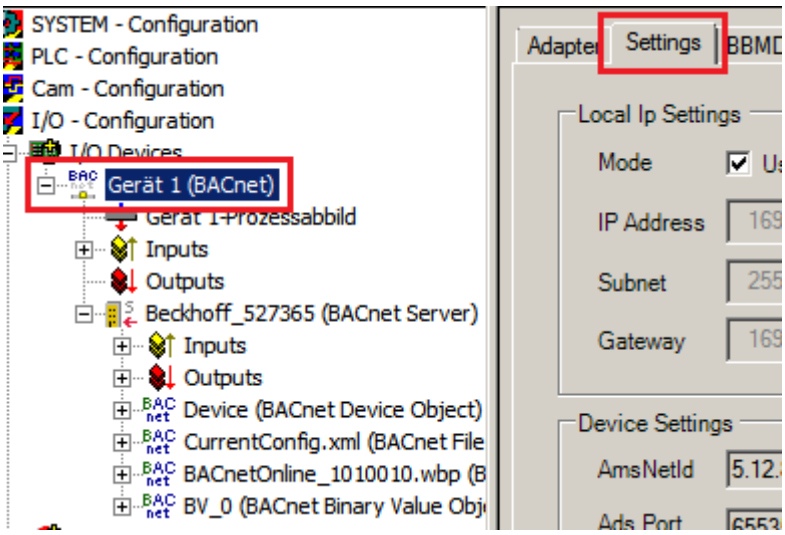


Beispiel für COV-P Fehlermeldung

- **Wie kann die maximale Anzahl Subscriptions, die der Server pro Objekt zulässt, erhöht werden?**

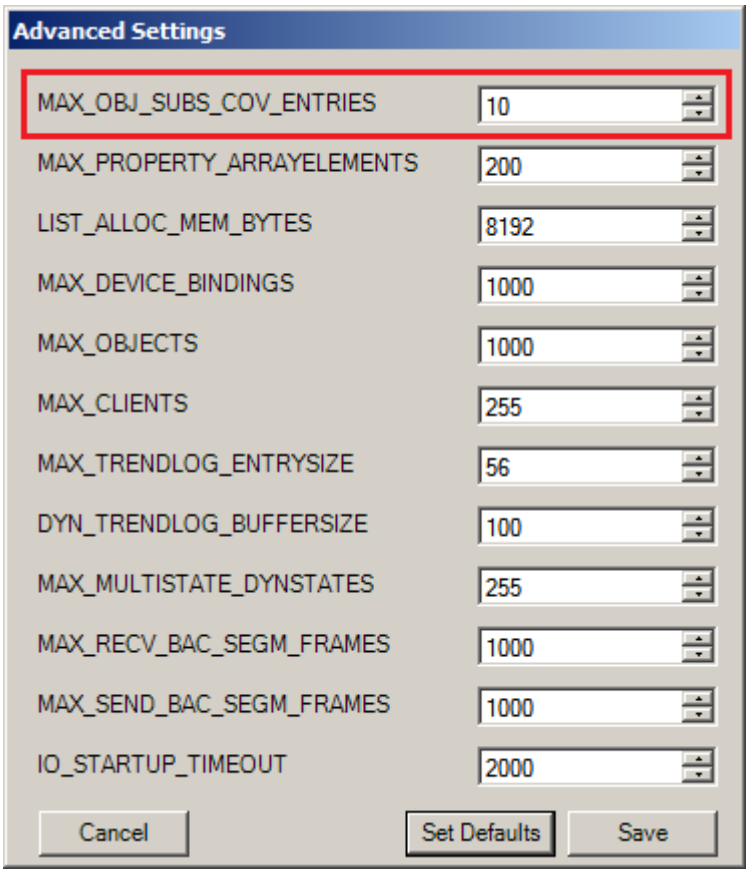
  Auf Seite des Servers muss folgender Parameter angepasst werden, um die maximal Anzahl von Subscriptions pro Objekt zu erhöhen (siehe Erweiterte Einstellungen unter Speicherspezfische Parameter [▶ 539]):

  a. BACnet-Adapter des entsprechenden Server anwählen



b. Button "Configure" unter "Advanced Settings - Data Structures" betätigen

c. Parameter "MAX_OBJ_SUBS_COV_ENTRIES" entsprechend erhöhen



d. Button "Save" zum Übernehmen betätigen / Abbruch mit Button "Cancel"

e. Geänderte Einstellung mit Symbol "Activate configuration" bzw. im Modus "CONFIG" mit Symbol "Reload I/O devices (F4)" aus der Toolbar aktivieren.

II. Beim Stopp/Herunterfahren des TwinCAT-Systems werden die Prozessdaten einiger BACnet-Objekte zurückgesetzt

Beim Stopp/Herunterfahren des TwinCAT-Systems werden die Prozessdaten einiger BACnet-Objekte zurückgesetzt. Dies führt zu ungewollten Effekten, da einige Objektzustände Einfluss auf die Persistenten Daten der SPS nehmen. Wie kann die Gültigkeit der Objekt-Zustände in der SPS überwacht werden?

1. Die Gültigkeit der Objektzustände kann mit Hilfe des Device Status (Bit 8...15) des lokalen BACnet-Adapters überprüft werden, indem der Zustand der Device-State-Machine abgefragt (in die SPS gemapped) wird. Meldet die Device-State-Machine Complete (8), so wurden der lokale Server und dessen Objekte sowie Properties initialisiert und die Kommunikation zu entfernten Servern kann aufgenommen werden. Beim Herunterfahren des TwinCAT-Systems wird die Device-State-Machine zurückgesetzt (< 8). Wird das Zurücksetzten (< 8) in der SPS erkannt, dann sollten die Objektzustände (gemappte Properties) nicht mehr ausgewertet werden.
2. Client-Objekte (von entfernten Servern) können generell nicht auf Gültigkeit geprüft werden, da Kommunikations-Timeouts etc. die zeitnahe Überwachung nicht erlauben. Das Fehlschlagen von BACnet-Diensten kann jedoch aus dem Client-Status entnommen werden (Bit 0 und 1, siehe Kapitel "Prozessdaten [▶ 43]" unter Abschnitt "BACnet Client"). Schlagen Anfragen fehl, so werden die entsprechenden Bits im Client-Status gesetzt. Diese müssen anschließend quittiert werden. Die Überwachung des lokalen BACnet-Adapters kann mit Hilfe des Device-Status erfolgen (siehe Punkt 1.).

III. Das Schreiben von Daten bei Wertänderung zu einem entfernten Server (Write On Change) soll kurzzeitig deaktiviert werden

Das Schreiben von Daten bei Wertänderung zu einem entfernten Server (Write On Change) soll kurzzeitig deaktiviert werden (z.B. für das Nach-Triggern des PresentValue eines BinaryOutput-Objekts, ohne dass der Zustand INACTIVE gesetzt wird). Wie kann das realisiert werden?

1. Im Prozessdatum ClientControl kann mit Hilfe des Bit 2 das Schreiben von Write On Change deaktiviert werden (siehe Kapitel "Prozessdaten" unter Abschnitt "BACnet Client [▶ 45]").
2. Das Nach-Triggern des Schreibens auf ein entferntes Objekt kann alternativ mittels ADS-Kommando vorgenommen werden (siehe Kapitel "ADS-Interface [▶ 520]"). Die PLC-Library "TcBACnet.lib" bietet dazu den komfortablen Funktionsbaustein "FB_BACnetWriteProp".

IV. Wie kann das Versenden von Alarm-Events bei BinaryOutput-/-Input- bzw. -Value-Objekten unterbunden werden?

1. Das Versenden der Events kann durch Abwahl der Bits im Reiter "Settings" bzw. "Online" oder durch Schreiben der Property EventEnable Nr. 35 verhindert werden (Value = 0x0005). Damit ist jedoch lediglich das Versenden der Event-Notifications unterdrückbar. Das Status-Bit in_alarm der Property StatusFlags Nr. 111 wird weiterhin gesetzt.
2. Um das Generieren jeglicher Events des jeweiligen Binary-Objekts zu verhindern, kann die Unterstützung für intrinsic reporting (O4) deaktiviert werden (siehe Kapitel "BACnet Objekte und Properties" unter Abschnitt [▶ 22] "Konfiguration"). Damit wird ebenfalls das Status-Bit in_alarm der Property StatusFlags Nr. 111 nicht gesetzt.

V. Wie kann eine NULL (NaN) in das Priority-Array der kommandierbaren Property PresentValue eines AnalogOutput-Objekts von der SPS über die Prozessdaten geschrieben werden?

1. Das PresentValue muss mit der entsprechenden Priorität als Prozessdatum aktiviert werden
2. Das Prozessdatum muss mit der SPS verknüpft werden (Datentyp REAL)
3. Im SPS-Programm kann mit der Library-Funktion "F_BACnet_NAN [▶ 492]" aus der Library "TcBACnet.lib" der Zustand NULL (NaN) auf die Prozessdaten geschrieben werden → Die entsprechende Priorität des Priority-Array ist damit gelöscht

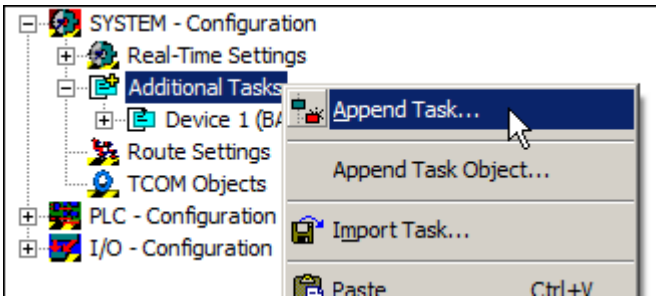
VI. Warum ist das Prozessdatum PresentValue eines Binary*-Objekts vom Typ WORD

Das PresentValue eines Binary*-Objekts wird BACnet-intern als Enumeration vom Typ BACnetBinaryPV [▶ 525] (active, inactive) geführt. Enumerationen werden generell mit dem Typ WORD gemapped. Für den einfacheren Umgang mit Binary*-Objekten wurde die herstellerspezifische Property PresentValueBool eingeführt. Diese kann mit Datentyp BOOL mit der SPS gemapped werden (siehe Kapitel "Prozessdaten [▶ 43]").

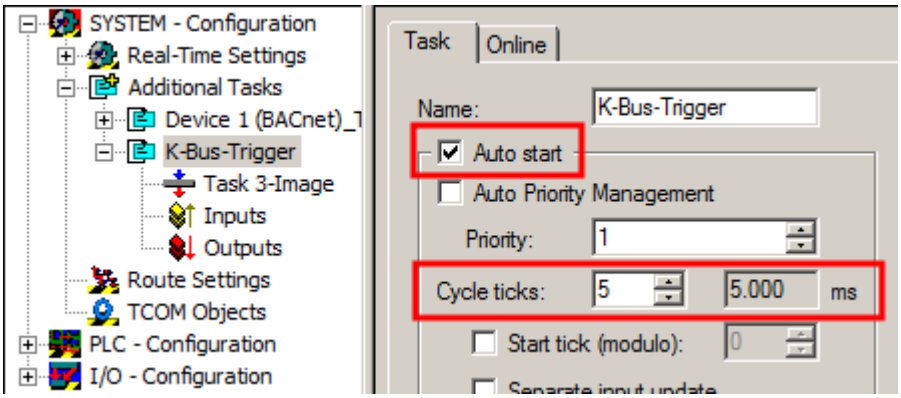
VII. Der K-Bus eines BACnet-Controllers fällt sporadisch, ohne ersichtlichen Grund aus. Was könnten mögliche Ursachen sein?

1. Der maximale K-Bus-Strom wurde überschritten. Wenn z.B. viele Relais-Ausgangsklemmen im Betrieb gleichzeitig aktiviert werden und dadurch der maximale K-Bus-Strom das zulässige Maximum übersteigt, kommt es zur K-Bus Abschaltung. Der Strombedarf der Klemmen sollte überprüft (siehe Strombedarf im Datenblatt der jeweiligen Klemme) und eine entsprechende K-Bus-Auffrischung (siehe Beschreibung zur KL9400) vorgesehen werden.
2. Wenn der K-Bus-Task durch den BACnet-Task getriggert wird (Mapping ausschließlich zwischen BACnet und K-Bus, keine SPS etc. → asynchrones Mapping), dann kann es bei starker Auslastung des BACnet-Tasks dazu kommen, dass der K-Bus-Task nicht ausreichend oft getriggert wird. Der Grund liegt in der Task-Zeit-Überschreitung des BACnet-Tasks (häufige Exceeds → siehe Exceed counter der entsprechenden Task) bei hohem BACnet-Kommunikationsaufkommen.
    - **Wie kann der K-Bus-Task unabhängig vom BACnet-Task getriggert werden?**

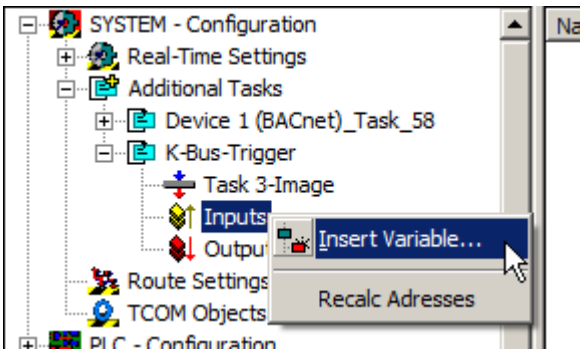
a. Einen neuen Task im TwinCAT System-Manager anlegen



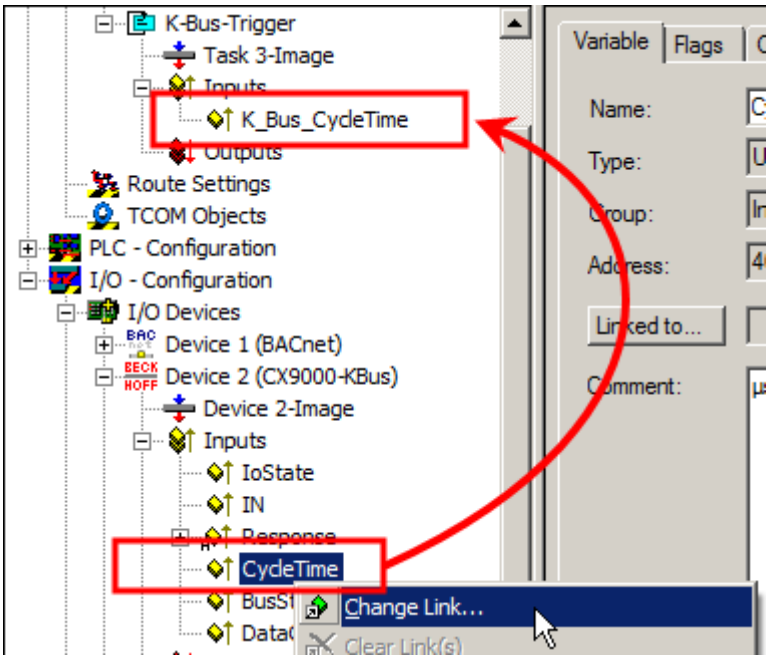
b. Taskkonfiguration vornehmen; Auto start aktivieren und die nötige Zykluszeit (Cycle ticks) einstellen



c. Eine Variable für das Mapping hinzufügen (Z.B. CycleTime vom Typ UINT)



d. K-Bus-Variable "CycleTime" und Task-Variable "K_Bus_CycleTime" verknüpfen



e. Geänderte Konfiguration mit Symbol "Activate configuration" aus der Toolbar aktivieren

VIII. Warum hat die Property StatusFlags eines Objekts den Wert 0x0004 bzw. die Property EventEnable den Wert 0x0005 obwohl keinerlei Statusbits der Property gesetzt sind?

Properties mit Bit-Flags (Untertyp BitFieldBits) werden intern als sogenannter Bit-String übertragen. Ein Bit-String besteht immer aus einem High- und Low-Byte. Im Low-Byte befindet sich die Angabe wie viel Bits des High-Bytes, von links, nicht benutzt werden. D.h. StatusFlags = 0x0004 bedeutet das die 4 höheren Bits des High-Byte nicht benutzt werden (also nur die Bits 8, 9, 10 u. 11 des WORD-Datentyps werden verwendet). Für weitere Details siehe Kapitel "Prozessdaten / Bit String [▶ 43]".

IX. Ein BACnet-Client dessen Prozessdaten mit der SPS verbunden sind wird im System-Manager immer als "Offline" angezeigt. Bei einem Scan wird der Client jedoch angezeigt. Was könnte das Problem sein?

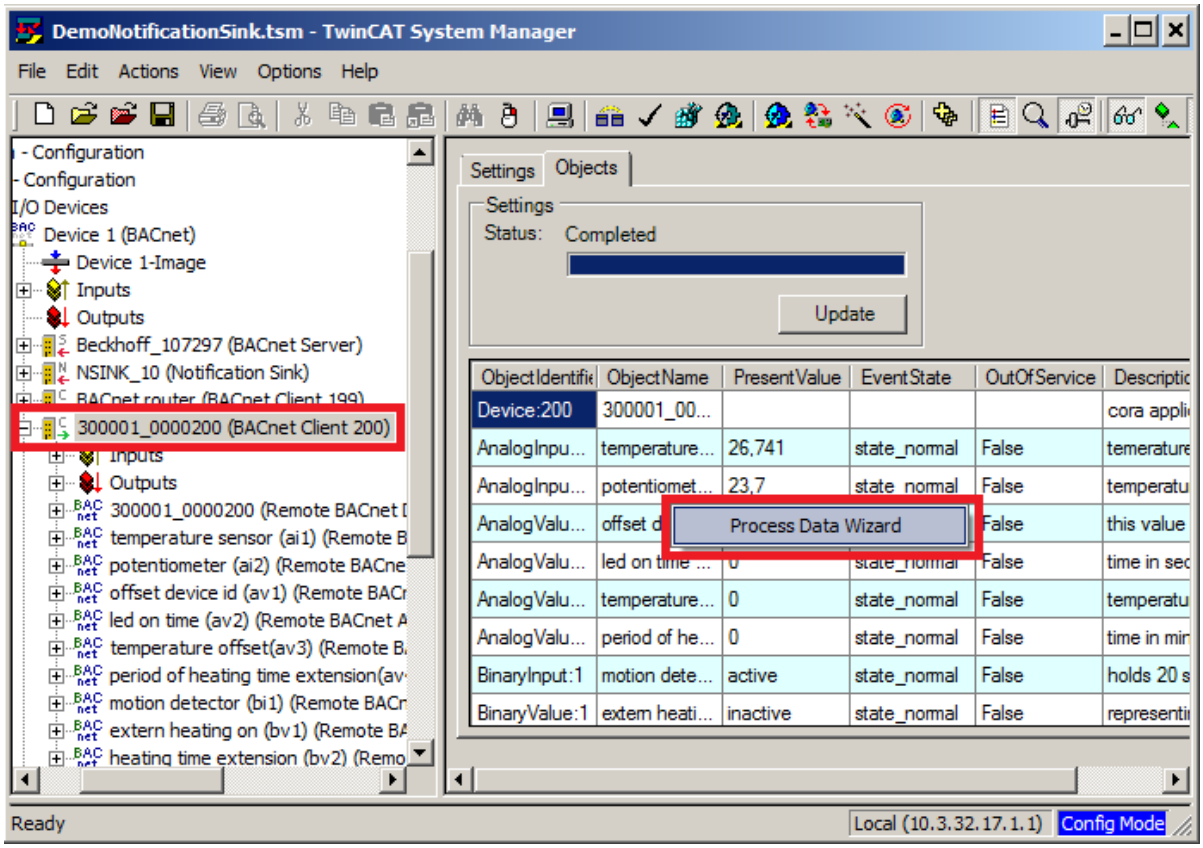
Bei Verwendung von Prozessdaten, die mit Hilfe des SPS-Automappings verknüpft wurden, werden die Properties im Standardfall mittels COV abonniert. Dies kann zu Problemen führen, wenn der Client COV nicht unterstützt (Loggermeldung wie etwa: "Error TCOM Server (10) [...] BACnet Client [...] - COV subscription failed! Reject Reason: **unrecognized service**"). Da das COV-Abonnieren fehl schlägt werden entsprechend Neu-Anfragen an den entfernten Server fortlaufend gesendet - dies führt zu Zeitüberschreitungen und schließlich zur Anzeige des Clients im System-Manager als "Offline".

Lösung: Die Eingangsprozessdaten der Properties des Clients müssen im System-Manager auf den Modus "Polling" umgestellt werden.

Wie können die Eingangsprozessdaten (Read) eines Clients von COV auf Polling umgestellt werden?

1. Entsprechenden Client anwählen
2. Reiter "Objects" öffnen

3. Property-Wizard öffnen (siehe auch BACnet Wizards [▶ 32])



4. Unter "Select Objects" den Client vollständig anwählen und...
   - a. Option "Optimize Process Data Access (Tick Modulo)" aktivieren
   - b. Option "Change Process Data In Configuration (Read)" aktivieren
   - c. Modus "Process Data Read Mode" auf "Polling" stellen
   - d. Parameter "Cycle Time (ms)" auf z.B. 10000 = 10 Sekunden Lesezyklus einstellen
     Hinweis **Dieser Parameter muss auf Grund der Applikationsanforderung optimiert und u.U. individuell für die jeweilige Property bestimmt werden**
5. Unter "Select Properties" sämtliche Einträge anwählen (mit rechter Maus-Taste und "Select All")
6. Mit der Betätigung des Buttons "Apply" werden die Änderungen übernommen.

X. Das PLC Automapping ist wesentlich langsamer als üblich. Was könnte die Ursache sein?

Ursache: Das PLC Automapping sollte nicht durchgeführt werden, wenn der System Manager auf dem Zielsystem eingeloggt ist. Durch die Verbindung zum Zielsystem kommt es zu Verzögerungen beim Anlegen von Verknüpfungen zwischen den Prozessdaten. Da beim PLC Automapping potenziell viele Prozessdaten verknüpft werden, summiert sich diese kurze Verzögerung zu einer wesentlich längeren PLC Automapping Laufzeit.

Lösung: Bevor das PLC Automapping durchgeführt wird, sollte im System Manager die lokale Laufzeit als Zielsystem gewählt werden. Nach dem PLC Automapping kann auf das eigentliche Zielsystem zurück gewechselt werden.

4 SPS Bibliothek: TcBACnetRev12.Lib

Im Folgenden wird der Funktionsumfang der SPS Bibliothek "TcBACnetRev12.lib" beschrieben. Die Bibliothek bietet die Möglichkeit auf Objekte einer BACnet-Konfiguration komfortabel aus einem SPS-Programm zuzugreifen.

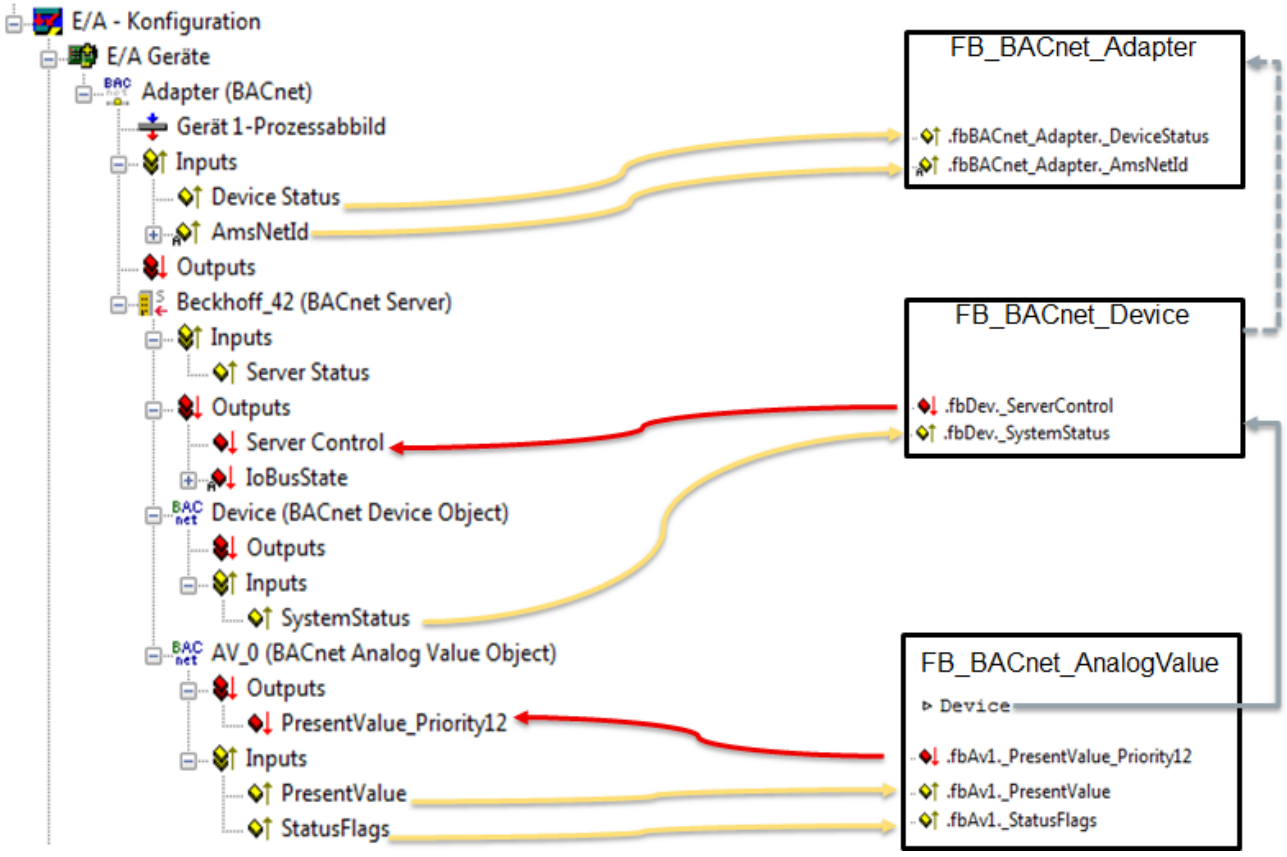
Überblick

Die SPS-Bibliothek "TcBACnetRev12.lib" ist eine Sammlung von Funktionsbausteinen zur Programmierung eines BACnet-Controllers. Für jedes BACnet-Objekt stehen Funktionsbausteine zur Verfügung, über die ausgewählte *Properties* gelesen und geschrieben werden. Generell wird zwischen Funktionsbausteinen für Server- und Client-Objekte unterschieden. Funktionsbausteine für Client-Objekte werden mit dem Präfix "Remote" bezeichnet, da sie den Zugriff auf entfernte BACnet-Objekte auf anderen Geräten ermöglichen. Die BACnet-Objekt-Funktionsbausteine realisieren eine Auswertung, ob Prozessdaten gültig sind bzw. der BACnet-Controller betriebsbereit ist (bReady).

Der Funktionsbaustein FB_BACnet_Adapter [▶ 156] repräsentiert ein BACnet-Device und damit den Zugangspunkt zum BACnet-Netzwerk über Netzwerkkarte. Über diesen Baustein kann u.a. erkannt werden, ob ein Link vorhanden ist. Um die Übersicht im SPS-Programm zu erhöhen, wird der BACnet-Adapter als globale Variable innerhalb der "TcBACnetRev12.lib" angelegt, da in der Mehrzahl der Projekte genau ein BACnet-Adapter verwendet wird.

Je nach Client- bzw. Server-Funktionalität, werden mit den Funktionsbausteinen FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243] Zustands- und Steuerinformationen des BACnet-Client bzw. Server verknüpft. Die Betriebsbereitschaft der BACnet-Objekte wird über den Client- bzw. Server-Status sowie der Property *System_Status* des jeweiligen Device bzw. Remote Device-Objekts ermittelt. Jedem BACnet-Objekt-Funktionsbaustein muss deshalb eine Referenz auf die Instanz des zugehörigen FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243] übergeben werden, um die Statusauswertung (**bReady**) zu ermöglichen.

Die nachfolgende Abbildung zeigt eine Übersicht der Funktionsbausteine der "TcBACnetRev12.lib" und den entsprechenden Verknüpfungen mit den BACnet-Modulen. Die Verknüpfung zwischen einem SPS-Programm auf Basis von "TcBACnetRev12.lib" und den BACnet-Modulen einer Konfiguration über die Funktion SPS-Automapping [▶ 64] automatisiert erstellt werden.



Kompatibilität und Neuerungen

Die SPS-Bibliothek "TcBACnetRev12.lib" wird ab TwinCAT 2.11 Build 2042 mit der TwinCAT-Basis-Installation ausgeliefert. Die Bibliothek ist in wesentlichen Zügen mit der Vorgängerversion "TcBACnet.lib" kompatibel. Es folgt ein kurzer Überblick der Neuerungen:

- BACnet-FBs namenskompatibel (außer Accumulator)
- Verbessertes Alignment der Prozessdaten
- FBs für neue (Revision12-)Objekte, konsequent _Ex für alle Objekte
- FB_BACnet_Device
  - Datentyp, Name nAmsPort geändert (mit Rev. 12 funktionslos)
  - Automatisches Auslesen der ADS-Informationen
  - Auslösen Schreiben persistenter Daten
- Loop_Drv: Interne Regelung automatisch aktiv (via BACnet-Stack)
- Loop_Ex: Autotuning (Bestimmung PID), Rampenbegrenzung (AccLimit) Ausgaben
- Neue Ads-Funktionen:
  - String Decoding nach Win1252 (Description, ObjectName,EventMessageTexts)
  - ExceptionSchedule (bool), WeeklySchedule (bool), LogBuffer (Real), RecipientList (Read,Write), ObjectList, RecpientList (Notification Class)
  - Zugriff auf NotificationSink-Funktionen (Acknowledge, Meldungsliste auslesen, Löschen)
  - Unterstützung Zeitsynchronisation
- Remote_Bausteine
  - Fehler ERR_OPERATIONAL wenn Client nicht da
  - bTriggerWOC
  - Für Schreibzugriffe müssen explizit _EX oder _WR-Varianten verwendet werden

Ältere Projekte, die auf Basis der SPS-Bibliothek "TcBACnet.Lib" erstellt wurden, können nach folgendem Schema auf die Verwendung der neuen Bibliothek umgestellt werden:

1. Löschen von "TcBACnet.Lib" aus dem SPS-Projekt
2. Hinzufügen von "TcBACnetRev12.Lib"
3. Neu Übersetzen
4. Im System Manager:
   - Neu einlesen des SPS-Projects
   - ggf. altes Mapping Rückgängig machen
   - ggf. BACnet Device in neuer Revision anlegen.
   - SPS-Automapping durchführen

Persistente Daten können bei der Umstellung auf die neue "TcBACnetRev12.lib" nicht übernommen werden.

Generell gilt, dass SPS-Projekte auf Basis der alten "TcBACnet.Lib" auch ohne Umstellung mit "BACnet Revision 12" funktionieren.

Übersicht

Im Folgenden werden die Komponenten der SPS-Bibliothek in einer Übersicht dargestellt. Über entsprechende Verknüpfungen können detailliierte Informationen aufgerufen werden.

BACnet Adapter und Notification-Sink

Bausteine	Beschreibung
FB_BACnet_Adapter [▶ 156]	Funktionsbaustein für die Anbindung des PLC Programms an einen lokalen BACnet-Adapter (Netzwerkkarte).
FB_BACnet_NotificationSink [▶ 260]	Funktionsbaustein zur Realisierung einer ADS Verbindung mit einer BACnet NotificationSink.

Lokale BACnet-Objekte (Server)

Standard Objekte

Folgende Funktionsbausteine stellen die Verbindung zwischen BACnet Objekt im TwinCAT System Manager und der Verwendung im PLC Programm her. Die Bausteine enthalten sämtliche, für das SPS-Automapping notwendingen Kommentare.

Die Standard-Funktionsbausteine sind in 2 Varianten verfügbar:

- **Minimale Prozessdaten** - es werden die nötigsten Properties verknüpft; u.a. *Present_Value* und *Status_Flags*.
- **Erweiterte Prozessdaten** - zusätzlich sind u.a. *Object_Identifier*, *Event_Flags* und *Reliability* verknüpft. Bausteine mit erweiterten Prozessdaten sind mit der Endung *_EX* gekennzeichnet.

Die Dokumentation bezieht sich auf die Variante mit erweiterten Prozessdaten. Erweiterte Prozessdaten sind in der Beschreibung als solche gekennzeichnet.

Bausteine	Symbol	BACnet Objekt	Beschreibung
FB_BACnet_Accumulator_EX [▶ 160]		Accumulator	Ein BACnet Accumulator Objekt repräsentiert einen durch Pulszählung ermittelten Messwert.
FB_BACnet_AnalogInput_EX [▶ 170]		Analog Input	Ein BACnet Analog Input Objekt repräsentiert einen analogen Eingangswert.
FB_BACnet_AnalogOutput_EX [▶ 174]		Analog Output	Ein BACnet Analog Output Objekt repräsentiert einen analogen Ausgangswert.
FB_BACnet_AnalogValue_EX [▶ 180]		Analog Value	Repräsentiert einen analogen Zustandswert innerhalb eines Programms.
FB_BACnet_Averaging_EX [▶ 183]		Averaging	Ermöglicht die Berechnung statistischer Daten innerhalb einer Steuerung.
FB_BACnet_BinaryInput_EX [▶ 184]		Binary Input	Ein Binary Input Objekt repräsentiert einen binären Eingangswert.
FB_BACnet_BinaryOutput_EX [▶ 188]		Binary Output	Ein Binary Output Objekt repräsentiert einen binaren Ausgangswert.
FB_BACnet_BinaryValue_EX [▶ 194]		Binary Value	Repräsentiert einen binären Zustandswert innerhalb eines Programms.
FB_BACnet_Calendar_EX [▶ 197]		Calendar	Ermöglicht die entkoppelte Definition von Ausnahmetagen für Zeitschaltpläne (Schedule Objekte).
FB_BACnet_Command_EX [▶ 198]		Command	Ermöglicht die Steuerung von komplexen Ablaufen über zeitlich gestaffelte Schreibbefehle auf BACnet Objekt-Properties.
FB_BACnet_Device [▶ 199]		Device	Bildet den logischen Einstiegspunkt eines BACnet-Geräts. Enthält u.a. die Liste aller BACnet-Objekte dieses Geräts.
FB_BACnet_EventEnrollment_EX [▶ 201]		Event Enrollment	Ermöglicht die Konfiguration regelbasierter Ereignismeldungen. Über das in viele Objekte Integrierte Meldesystem, können umfangreichere Regeln für das Auslösen vom Ereignismeldungen definiert werden. Ein Beispiel sind zusätzliche oder mehrfache Grenzwertpaare für ein *PresentValue*.
FB_BACnet_File_EX [▶ 202]		File	Repräsentiert Eigenschaften eines Dateiobjekts.

Bausteine	Symbol	BACnet Objekt	Beschreibung
FB_BACnet_Group_EX [▶ 203]		Group	Group-Objekte erlauben die Zusammenfassung multipler *Properties* in einem einzelnen Datenpunkt.
FB_BACnet_Loop_EX [▶ 204]		Loop	Repräsentiert die Eigenschaften eines PID-Reglers.
FB_BACnet_MultiStateInput_EX [▶ 208]		Multi State Input	Repräsentiert einen ganzzahligen/ mehrstufigen Eingangswert.
FB_BACnet_MultiStateOutput_EX [▶ 212]		Multi State Output	Repräsentiert einen ganzzahligen/ mehrstufigen Ausgangswert.
FB_BACnet_MultiStateValue_EX [▶ 218]		Multi State Value	Repräsentiert einen ganzzahligen/ mehrstufigen Zustandswert.
FB_BACnet_NotificationClass_EX [▶ 221]		Notification Class	Das Notification Class Objekt dient zur Konfiguration der Verteilung von Ereignismeldungen (EventNotifications).
FB_BACnet_Program_EX [▶ 222]		Program	Ein BACnet Program Objekt ermöglich die Veränderung der Zustände eines SPS-Programms.
FB_BACnet_PulseConverter_EX [▶ 225]		Pulse Converter	Ein Pulse Converter Objekt repräsentiert einen durch Pulszählung ermittelten Messwert.
FB_BACnet_Schedule_EX [▶ 228]		Schedule	Repräsentiert einen Zeitschaltplan mit dessen Hilfe Werte anderer BACnet-Objekte an Hand von zeitbasierten Schalteinträgen beschrieben werden.
FB_BACnet_TrendLog_EX [▶ 229]		Trend Log	Repräsentiert aufgezeichnete historische Daten, die zyklisch mit festem Intervall oder ereignisbasiert aufgezeichnet werden.

Lokale BACnet Objekte (ADS)

Der schreibende Zugriff auf die Property *PresentValue* erfolgt prioritätsbasiert via ADS. Dieser Baustein kann eingesetzt werden, wenn Prioritäten für schreibende Zugriffe vor der Programmlaufzeit nicht feststehen bzw. mehrere Prioritätsstufen gleichzeitig im Programm manipuliert werden sollen.

Bausteine	Beschreibung
FB_BACnet_AnalogOutput_ADS [▶ 175]	Ein BACnet Analog Output Objekt repräsentiert einen analogen Ausgangswert.
FB_BACnet_AnalogValue_ADS [▶ 181]	Repräsentiert einen analogen Zustandswert innerhalb eines Programms.
FB_BACnet_BinaryOutput_ADS [▶ 189]	Ein Binary Output Objekt repräsentiert einen binaren Ausgangswert.
FB_BACnet_BinaryValue_ADS [▶ 195]	Repräsentiert einen binären Zustandswert innerhalb eines Programms.
FB_BACnet_MultiStateOutput_ADS [▶ 213]	Repräsentiert einen ganzzahligen Ausgangswert
FB_BACnet_MultiStateValue_ADS [▶ 219]	Repräsentiert einen ganzzahligen Ausgangswert.

Tab. 1: Lokale BACnet Objekte (Treiber)

Bausteine	Beschreibung
FB_BACnet_Loop_DRV_EX [▶ 206]	Dieser Baustein wird mit einem TwinCAT BACnet Loop Objekt mit aktivierter interner Regelung verknüpft. Der Regelungsalgorithmus wird direkt im BACnet-Stack und nicht in der SPS implementiert. Alle anderen Loop-FBs implementieren ihren Regelungsalgorithmus innerhalb der SPS. Der Vorteil

Bausteine	Beschreibung
	dieses Bausteins in eine effizientere Ausführungszeit, da weniger BACnet-Properties über Prozessdaten verknüpft werden müssen.

Lokale BACnet Objekte (Raw Varianten)

RAW-FBs ermöglichen die Vorgabe physikalischer Mess- und Stellgrößer aus der SPS. Im Gegensatz zu anderen FBs für physikalische Mess- und Stellgrößen, welche direkt mit einem Hardware-Modul verknüpft sind. Das zugehörige RawIo*-Prozessdatum des BACnet-Objekts wird hier vom SPS-Automapping direkt mit einer SPS-Variablen verknüpft.

Bausteine	Beschreibung
FB_BACnet_Accumulator_RAW [▶ 163]	Ein BACnet Accumulator Objekt repräsentiert einen durch Pulszählung ermittelten Messwert. Dieser Messwert wir in diesem Fall in der SPS gebildet.
FB_BACnet_AnalogInput_RAW [▶ 171]	Ein BACnet Analog Input Objekt repräsentiert einen analogen Eingangswert. Dieser Funktionsbaustein kann eingesetzt werden, wenn eine Nachverarbeitung von analogen Eingangswerten in der SPS erfolgen soll, oder analoge Werte direkt in der SPS gebildet werden und diese nur lesbar sein sollen.
FB_BACnet_AnalogOutput_RAW [▶ 177]	Ein BACnet Analog Output Objekt repräsentiert einen analogen Ausgangswert. Dieser Funktionsbaustein kann u.a. eingesetzt werden, wenn ein analoges Ausgangssignal vor der Ausgabe vorverarbeitet werden soll.
FB_BACnet_BinaryInput_RAW [▶ 185]	Ein Binary Input Objekt repräsentiert einen binären Einsgangswert. Dieser Funktionsbaustein kann eingesetzt werden wenn eine Nachverarbeitung von binären Eingangswerten in der SPS erfolgen soll (z.B. Entprellung)
FB_BACnet_BinaryOutput_RAW [▶ 191]	Ein Binary Output Objekt repräsentiert einen binaren Ausgangswert. Dieser Funktionsbaustein kann u.a. eingesetzt werden, wenn ein binäres Ausgangssignal vor der Ausgabe vorverarbeitet werden soll.
FB_BACnet_MultiStateInput_RAW [▶ 209]	Ein Multistate Input Objekt repräsentiert einen ganzzahligen/ mehrstufigen Eingangswert.
FB_BACnet_MultiStateOutput_RAW [▶ 215]	Ein Multistate Output Objekt repräsentiert einen ganzzahligen/ mehrstufigen Ausgangswert.
FB_BACnet_PulseConverter_RAW [▶ 227]	Ein Pulse Converter Objekt repräsentiert einen durch Pulszählung ermittelten Messwert.

Tab. 2: Lokale BACnet Objekte (Revision 6 spezifisch)

Bausteine	Beschreibung
FB_BACnet_Accumulator_R6 [▶ 166]	Zur Revision 6 kompatibler Funktionsbausstein, der nur mit TwinCAT BACnet Revision 6 verwendet werden sollte.
FB_BACnet_Accumulator_RAW_R6 [▶ 167]	Zur Revision 6 kompatibler Funktionsbausstein, der nur mit TwinCAT BACnet Revision 6 verwendet werden sollte.

Client BACnet Objekte (Remote)

Die Funktionsbausteine für Remote-Objekte bieten den Zugriff auf BACnet-Objekte anderer Geräte. Bei den Remote-FBs werden drei Varianten unterschieden. Die Basis-FBs ohne Endung bieten den lesenden Zugriff auf die Properties *PresentValue* und *StatusFlags*. Eine umfangreichere Auswahl an Properties wird in den "_EX"-Varianten unterstützt. Wenn mit minimal lesenden Properties zusätzlich schreibende Zugriffe ausgeführt werden sollen, können die "_WR"-Varianten verwendet werden. Generell sollten die Funktionsbausteine mit wenigen Properties bevorzugt werden, um die Netzlast zu minimieren.

Für eine effiziente Einbindung entfernter BACnet-Geräte kann die Funktion .exp-Export (im BACnet Device) verwendet werden, mit der sehr einfach die SPS-Variablen-Deklaration der Remote-FBs gescannter bzw. über eine EDE-Datei eingelesene Client-Konfigurationen erstellt werden.

Bausteine	Beschreibung
FB_BACnet_RemoteAccumulator [▶ 230]	Ein BACnet Accumulator Objekt repräsentiert einen durch Pulszählung ermittelten Messwert.
FB_BACnet_RemoteAnalogInput [▶ 232]	Ein BACnet Analog Input Objekt repräsentiert einen analogen Eingangswert.
FB_BACnet_RemoteAnalogOutput [▶ 233]	Ein BACnet Analog Output Objekt repräsentiert einen analogen Ausgangswert.
FB_BACnet_RemoteAnalogValue [▶ 234]	Repräsentiert einen analogen Zustandswert innerhalb eines Programms.
FB_BACnet_RemoteAveraging [▶ 236]	Ermöglicht die Berechnung statistischer Daten innerhalb einer Steuerung.
FB_BACnet_RemoteBinaryInput [▶ 237]	Ein Binary Input Objekt repräsentiert einen binären Einsgangswert.
FB_BACnet_RemoteBinaryOutput [▶ 238]	Ein Binary Output Objekt repräsentiert einen binaren Ausgangswert.
FB_BACnet_RemoteBinaryValue [▶ 240]	Repräsentiert einen binären Zustandswert innerhalb eines Programms.
FB_BACnet_RemoteCalendar [▶ 241]	Ermöglicht die entkoppelte Definition von Ausnahmetagen für Zeitschaltpläne (Schedule Objekte).
FB_BACnet_RemoteCommand [▶ 242]	Ermöglicht die Steuerung von komplexem Abläufen über zeitlich gestaffelte Schreibbefehle auf BACnet Objekt-Properties.
FB_BACnet_RemoteDevice [▶ 243]	Bildet den logischen Einstiegspunkt eines BACnet-Geräts. Enthält u.a. die Liste aller BACnet-Objekte dieses Geräts.
FB_BACnet_RemoteEventEnrollment [▶ 246]	Ermöglicht die Konfiguration regelbasierter Ereignismeldungen. Über das in viele Objekte Integrierte Meldesystem, können umfangreichere Regeln für das Auslösen vom Ereignismeldungen definiert werden. Ein Beispiel sind zusätzliche oder mehrfache Grenzwertpaare für ein *PresentValue*.
FB_BACnet_RemoteFile [▶ 246]	Repräsentiert Eigenschaften eines Dateiobjekts.
FB_BACnet_RemoteGroup [▶ 247]	Group-Objekte erlauben die Zusammenfassung multipler *Properties* in einem einzelnen Datenpunkt.
FB_BACnet_RemoteLoop [▶ 248]	Repräsentiert die Eigenschaften eines PID-Reglers.
FB_BACnet_RemoteMultiStateInput [▶ 249]	Repräsentiert einen ganzzahligen/mehrstufigen Eingangswert.
FB_BACnet_RemoteMultiStateOutput [▶ 250]	Repräsentiert einen ganzzahligen/mehrstufigen Ausgangswert.
FB_BACnet_RemoteMultiStateValue [▶ 252]	Repräsentiert einen ganzzahligen/mehrstufigen Zustandswert.
FB_BACnet_RemoteNotificationClass [▶ 253]	Das Notification Class Objekt dient zur Konfiguration der Verteilung von Ereignismeldungen (EventNotifications).
FB_BACnet_RemoteProgram [▶ 254]	Ein BACnet Program Objekt ermöglich die Veränderung der Zustände eines SPS-Programms.
FB_BACnet_RemotePulseConverter [▶ 257]	Ein Pulse Converter Objekt repräsentiert einen durch Pulszählung ermittelten Messwert.
FB_BACnet_RemoteSchedule [▶ 258]	Repräsentiert einen Zeitschaltplan mit dessen Hilfe Werte anderer BACnet-Objekte an Hand von zeitbasierten Schalteinträgen beschrieben werden.
FB_BACnet_RemoteTrendLog [▶ 259]	Repräsentiert aufgezeichnete historische Daten, die zyklisch mit festem Intervall oder ereignisbasiert aufgezeichnet werden.

ADS Bausteine für den generischen Zugriff auf sämtliche online Properties (Rohdatenzugriff)

Funktionsbausteine für den Zugriff auf BACnet Properties über ADS. Sämtliche BACnet online Properties von Server und Client Objekten können über ADS gelesen bzw. geschrieben werden.

Bausteine	Beschreibung
FB_BACnet_ReadProp [▶ 275]	Lesezugriff auf Properties
FB_BACnet_WriteProp [▶ 278]	Schreibzugriff auf Properties

ADS Bausteine für den Zugriff auf spezifische Properties

Funktionsbausteine für den spezifischen Zugriff auf BACnet Properties mit Datentypwandlung über ADS. Sämtliche BACnet Properties von Server und Client Objekten können über ADS gelesen bzw. geschrieben werden. Aufbauend auf den Bausteinen FB_BACnet_ReadProp und -WriteProp wandeln die folgenden Bausteine, die per ADS gelesen Daten in PLC Datentypen um bzw. codieren die PLC Daten in BACnet Daten während des Schreibzugriffs. Die Größenbegrenzung der folgenden ADS Zugriffe hängt mit dem globalen ADS-Datenpuffer zusammen und liegt bei ca. 8kByte (siehe ST_BACnet_GlobalAdsBuffer [▶ 363]).

Bausteine	Beschreibung	Zugriff
FB_BACnet_ObjectNameProperty [▶ 294]	ADS Zugriff auf die Property *Object_Name* vom Typ CharacterString inklusive Decoding von UTF-8, UCS-2 und UCS-4	Lesen
FB_BACnet_DescriptionProperty [▶ 281]	ADS Zugriff auf die Property *Description* vom Typ CharacterString inklusive Decoding von UTF-8, UCS-2 und UCS-4	Lesen
FB_BACnet_EventMessageTextsProperty [▶ 283]	ADS Zugriff auf die Property *Event_Message_Texts* vom Typ CharacterStringExtList inklusive Decoding von UTF-8, UCS-2 und UCS-4	Lesen
FB_BACnet_ObjectListProperty [▶ 292]	ADS Zugriff auf die Property *Object_List* vom Typ BACnetObjectIdentifier[]	Lesen
FB_BACnet_ExceptionScheduleProperty [▶ 285]	ADS Zugriff auf die Property *Exception_Schedule* vom Typ BACnetSpecialEventList; jedoch ausschließlich für Einträge mit Datentyp *Bool*	Lesen, Schreiben
FB_BACnet_WeeklyScheduleProperty [▶ 299]	ADS Zugriff auf die Property *Weekly_Schedule* vom Typ BACnetDailyScheduleList; jedoch ausschließlich für Einträge mit Datentyp *Bool*	Lesen, Schreiben
FB_BACnet_LogBufferProperty [▶ 289]	ADS Zugriff auf die Property *Log_Buffer* vom Typ BACnetLogRecordList; jedoch ausschließlich für Einträge mit Datentyp *Real*	Lesen
FB_BACnet_RecipientListProperty [▶ 296]	ADS Zugriff auf die Property *Recipient_List* vom Typ BACnetDestination[] (z.B. NotificationClass Objekt)	Lesen, Schreiben

ADS Bausteine für den Zugriff auf Dienst- und Diagnosedaten

Bausteine	Beschreibung
FB_BACnet_GetDiagInfo [▶ 261]	ADS Zugriff auf die Diagnosedaten des BACnet Adapters
FB_BACnet_NSinkReadEvent [▶ 266]	ADS Zugriff auf die BACnet Notification Sink: Auslesen eines BACnet Events
FB_BACnet_NSinkAcknEvent [▶ 263]	ADS Zugriff auf die BACnet Notification Sink: Dienst zur Quittierung eines BACnet Events
FB_BACnet_NSinkRemoveEvent [▶ 270]	ADS Zugriff auf die BACnet Notification Sink: Löschen eines BACnet Events (ersetzt FB_BACnet_NotificationSinkDelEntry [▶ 496])
FB_BACnet_TimeSync [▶ 272]	ADS Zugriff auf den BACnet Adapter: Dienst zur Zeitsynchronisierung im BACnet Netzwerk (broadcast) oder lokal. **Der Baustein stellt die aktuelle Systemzeit zyklisch als Ausgang bereit und sollte in jedem BACnet Projekt als Uhrzeitquelle im PLC Programm verwendet werden.**

BACnet Hilfsbausteine für Datum und Uhrzeit

Bausteine	Beschreibung
F_BACnet_CheckDay [▶ 309]	Funktion zur Prüfung eines Datum-Zahlenwertes (BYTE) für den Tag des Monats auf Gültigkeit.
F_BACnet_CheckDayOfWeek [▶ 310]	Funktion zur Prüfung eines Datum-Zahlenwertes (BYTE) für den Wochentag auf Gültigkeit.
F_BACnet_CheckHour [▶ 310]	Funktion zur Prüfung eines Uhrzeit-Zahlenwertes (BYTE) für die Stunden-Angabe auf Gültigkeit.
F_BACnet_CheckHundredths [▶ 310]	Funktion zur Prüfung eines Uhrzeit-Zahlenwertes (BYTE) für die Hundertstelsekunden-Angabe auf Gültigkeit.
F_BACnet_CheckMinute [▶ 311]	Funktion zur Prüfung eines Uhrzeit-Zahlenwertes (BYTE) für die Minutenangabe auf Gültigkeit.
F_BACnet_CheckMonth [▶ 311]	Funktion zur Prüfung eines Datum-Zahlenwertes (BYTE) für den Monat auf Gültigkeit.
F_BACnet_CheckSecond [▶ 311]	Funktion zur Prüfung eines Uhrzeit-Zahlenwertes (BYTE) für die Sekundenangabe auf Gültigkeit.
F_BACnet_CheckWeekOfMonth [▶ 312]	Funktion zur Prüfung eines Datum-Zahlenwertes (BYTE) für die Woche des Monats auf Gültigkeit.
F_BACnet_CheckYear [▶ 312]	Funktion zur Prüfung eines Datum-Zahlenwertes (BYTE) für das Jahr auf Gültigkeit.
F_BACnet_DateHasPlaceholder [▶ 313]	Funktion zur Prüfung auf Platzhalter (255 → undefiniert) in einer Datumsangabe.
F_BACnet_DateMerge [▶ 313]	Funktion für das Zusammenführen von 2 Zeitstempeln.
F_BACnet_DateTime_TO_TimeStruct [▶ 308]	Funktion zur Umwandlung eines BACnet Zeitstempels in den Datentyp *TIMESTRUCT*.
F_BACnet_DateTimeString [▶ 309]	Funktion zur Zeichenkettendarstellung eines BACnet Zeitstempels.
F_BACnet_DateUnspecified [▶ 313]	Funktion zur Prüfung auf Platzhalter (*255* → undefiniert) in einer Datumsangabe.
F_BACnet_DaysInMonth [▶ 313]	Funktion zur Berechnung der Anzahl Tage im gegebenen Monat eines Jahres.
F_BACnet_Get100msDate [▶ 314]	Funktion zur Berechnung des Zeitstempels in 100ms Schritten seit 1900.
F_BACnet_Get100msTime [▶ 314]	Funktion zur Berechnung des Zeitstempels in 100ms Schritten seit 00:00:00.0 Uhr.
F_BACnet_TimeHasPlaceholder [▶ 314]	Funktion zur Prüfung auf Platzhalter (*255* → undefiniert) in einer Zeitangabe.
F_BACnet_TimeMerge [▶ 314]	Funktion für das Zusammenführen von 2 Zeitstempeln.
F_BACnet_TimeString [▶ 309]	Funktion zur Zeichenkettendarstellung eines BACnet Zeitstempels.
F_BACnet_TimeStruct_TO_DateTime [▶ 308]	Funktion zur Umwandlung eins Zeitstempels vom Datentype *TIMESTRUCT* in einen BACnet Zeitstempel.
F_BACnet_TimeUnspecified [▶ 315]	Funktion zur Prüfung auf Platzhalter (*255* → undefiniert) in einer Zeitangabe.

BACnet Hilfsbausteine für Bit-Konvertierung (Bit Strings)

Bausteine	Beschreibung
F_BACnet_EventTransitionFlags [▶ 316]	Funktion zum Decodieren der Prozessdaten der Property *Acked_Transitions* eines BACnet Objekts.
F_BACnet_GetEventTransFlagsData [▶ 316]	Funktion zum Codieren des Property-Werts der Property *Event_Enable* eines BACnet Objekts.
F_BACnet_GetLimitEnFlagsData [▶ 317]	Funktion zum Codieren des Property-Werts der Property *Limit_Enable* eines BACnet Objekts.

Bausteine	Beschreibung
F_BACnet_GetStatusFlagsData [▶ 315]	Funktion zum Codieren des Property-Werts der Property *Status_Flags* eines BACnet Objekts.
F_BACnet_LimitEnableFlags [▶ 317]	Funktion zum Decodieren des Prozessdatums der Property *Limit_Enable* eines BACnet Objekts.
F_BACnet_StatusFlags [▶ 315]	Funktion zum Decodieren des Prozessdatums der Property *Status_Flags* eines BACnet Objekts.

BACnet Hilfsbausteine für Multi-State-Objekte

Bausteine	Beschreibung
F_BACnet_MultiStatePV [▶ 318]	Funktion zur Umsetzung eines UDINT-Wertes der PLC in den Prozessdatenwert eines BACnet MultiState* Objekts Property *Present_Value*.

BACnet Hilfsbausteine für REAL-Werte

Bausteine	Beschreibung
F_BACnet_RealPV [▶ 318] (obsolet: F_BACnet_AnalogPV [▶ 324])	Funktion zur Umsetzung eines REAL-Wertes der PLC in den Prozessdatenwert eines BACnet Analog* Objekts Property *Present_Value*.
F_BACnet_IsFinite [▶ 320]	Funkktion zum Prüfen auf Endlichkeit eines REAL Werts.
F_BACnet_RealEQ [▶ 320]	Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs.
F_BACnet_RealGE [▶ 321]	Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs.
F_BACnet_RealGT [▶ 321]	Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs.
F_BACnet_RealLE [▶ 321]	Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs.
F_BACnet_RealLT [▶ 322]	Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs.
F_BACnet_RealNull [▶ 319] (obsolet: F_BACnet_NAN [▶ 492])	Funktion gibt den Gleitpunktwert "**N**ot **a**N**u**mber" zurück, der als Wert der Property *Present_Value* eines Analog* Objekt dem Wert *Null* (keine Wert) entspricht.
F_BACnet_RealNothing [▶ 319]	Funktion gibt den Gleitpunktwert "**N**ot **a**N**u**mber" zurück, der als Wert der Property *Present_Value* eines Analog* Objekt dem Wert *Nothing* (nicht-auswerten) entspricht. Prozessdaten mit dem codierten Wert *Nothing* werden vom BACnet Stack ab Revision 12 nicht verarbeitet.
F_BACnet_RealToStr [▶ 319]	Funktion zur Umwandlung einer Gleitpunktzahl in eine Zeichenkette unter Berücksichtigung des Wertebereichs.

BACnet Hilfsbausteine für String-Verarbeitung

Bausteine	Beschreibung
F_BACnet_GetObjectIdString [▶ 322]	Funktion zur Ausgabe der Objekt-ID eines BACnet Objekts als kurze Zeichenkette.
FB_BACnet_StringExtDecode [▶ 323]	Funktionsbaustein zum Decodieren von BACnet Strings.
FB_BACnet_StringExtEncode [▶ 324]	Funktionsbaustein zum Codieren von BACnet Strings.

Signalkonvertierung

Bausteine	Beschreibung
FB_BACnet_PWM [▶ 329]	Funktionsbaustein zum Erzeugen eines pulsierenden Ausgabesignals, mit definierter Ein- und Ausschaltdauer in Prozent der Periodendauer (**P**uls**w**eiten**m**odulation).

Hilfsfunktionen für BACnet BinaryPV Datentypen

Bausteine	Beschreibung
F_BinPV_AND [▶ 326]	Funktion zur logischen Verknüpfung von BACnet BinaryPV Werten.
F_BinPV_NOT [▶ 327]	Funktion zum Invertieren eines BACnet BinaryPV Werts.
F_BinPV_OR [▶ 327]	Funktion zur logischen Verknüpfung von BACnet BinaryPV Werten.
F_BinPV_XOR [▶ 328]	Funktion zur logischen Verknüpfung von BACnet BinaryPV Werten.
F_BinPV_To_Bool [▶ 326]	Funktion zum Konvertieren eines BACnet BinaryPV Werts in den Datentyp BOOL.
F_Bool_To_BinPV [▶ 325]	Funktion zur Umsetzung eines Wertes mit Datentyp BOOL der PLC in den Prozessdatenwert eines BACnet Binary* Objekts / Property *Present_Value*.

Sonstige BACnet Hilfsfunktionen

Bausteine	Beschreibung
F_BACnet_GetObjectListIndex [▶ 304]	Funktion zum Feststellen der Position einer BACnet Object-ID in einer Liste mit IDs.
F_BACnet_GetObjId [▶ 303]	Funktion zur Codierung des Objekt-Typs und der Objekt-Instance in den BACnet *Object_Identifier*.
F_BACnet_GetObjInstance [▶ 303]	Funktion zum Decodierung des BACnet *Object_Identifier* in die Objekt-Instance (Objektnummer).
F_BACnet_GetObjType [▶ 303]	Funktion zur Decodierung des BACnet *Object_Identifier* in den Objekt-Typ.
FB_BACnet_AccLimit [▶ 304]	Funktionsbaustein für die Begrenzung von Signaländerung pro Zeit (maximale Beschleunigung).
FB_BACnet_AvgValue [▶ 305]	Funktionsbaustein zur Mittelwertbildung einer Eingangsgröße **X** über **n** Werte.
FB_BACnet_PidControl [▶ 305]	PID Regelbaustein in Parallelanordnung bzw. Idealform.
FB_BACnet_PT1 [▶ 308]	Funktionsbaustein zur Nachbildung einer Verzögerung 1. Ordnung.

BACnet Konstanten

Konstanten	Beschreibung
BACnet_Globals [▶ 330]	Globale Konsten für Default-Werte, Fehlermeldungen, Datenbereichsgrenzen usw.

BACnet Diagnose-Daten

Datentypen	Beschreibung
ST_BACnet_DiagEthStatistics [▶ 360]	Beschreibt Diagnosedaten des Ethernet-Adapter wie gesendete, empfangene oder fehlerhafte Nachrichten.
ST_BACnet_DiagnosisTiming [▶ 360]	Enthält Informationen über Ausführungszeiten innerhalb des BACnet-Stack.
ST_BACnet_FrameStatistics [▶ 363]	Enthält Informationen über gesendete und empfangene BACnet-Netzwerkpakete.
ST_BACnet_Info [▶ 364]	Informationen zum Speicherverbrauch.
ST_BACnet_ServerStatistics [▶ 369]	Enthält Informationen zu persistenten Daten und eine Änderungsstatistik der Properties.
ST_BACnet_TcIoEthStatistic [▶ 370]	Beschreibt Diagnosedaten des Ethernet-Adapter.
ST_BACnet_TcIoEthTxRxErrorCount [▶ 371]	Beschreibt Diagnosedaten des Ethernet-Adapter.
ST_BACnet_UnConfirmedServiceDiag [▶ 372]	Ausführliche Daten zu BACnet-Diensten

BACnet Datentypen

Datentypen	Beschreibung
ST_BACnet_AdsConnection [▶ 358]	Struktur mit den Verbindungsinformationen eines ADS Servers des BACnet Treiber
ST_BACnet_CharacterStringExt [▶ 358]	BACnet Zeichenkette inklusive des verwendeten Encoding-Formats.
ST_BACnet_CharacterStringExtListEntry [▶ 359]	Liste von BACnet Zeichenketten.
ST_BACnet_Date [▶ 359]	Datum mit Tag, Wochentag, Monat und Jahr.
ST_BACnet_DateTime [▶ 359]	Beschreibt Datum und Uhrzeit.
ST_BACnet_Diagnosis [▶ 360]	siehe BACnet Diagnose Daten.
ST_BACnet_EventTransitionBits [▶ 361]	PLC Abbildung des BACnet Datentyps BACnetEventTransitionBits (Properties *Event_Enable* und *Acked_Transitions*).
ST_BACnet_ExceptionScheduleBool [▶ 361]	Struktur für den Datenaustausch der Property *exception_schedule* mit Hilfe des Funktionsbausteins FB_BACnet_ExceptionScheduleProperty [▶ 285]
ST_BACnet_ExceptionScheduleEntryBool [▶ 362]	Teildaten der Property ExceptionSchedule des Schedule-Objekts
ST_BACnet_LimitEnable [▶ 365]	PLC Abbildung des BACnet Datentyps BACnetLimitEnable. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Limit_Enable*.
ST_BACnet_LogBufferEntryReal [▶ 365]	PLC Abbildung des BACnet Datentyps BACnetLogRecord für *Log_Datum* vom Typ *Real*. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Log_Buffer*.
ST_BACnet_LogBufferReal [▶ 365]	Struktur für den Datenaustausch der Property *Log_Buffer* mit Hilfe des Funktionsbausteins FB_BACnet_LogBufferProperty [▶ 289]
ST_BACnet_NSinkEvent [▶ 366]	PLC Abbildung der Daten eines Event-Eintrags der BACnet Notification Sink.
ST_BACnet_ObjectIdentifierList [▶ 366]	Struktur für die Beschreibung der Objektliste eines Geräts. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 Property Object_List (Device-Objekt).
ST_BACnet_ObjectTypesSupported [▶ 367]	PLC Abbildung des BACnet Datentyps BACnetObjectTypesSupported. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Protocol_Object_Types_Supported*.
ST_BACnet_ProgramHandshakeRequests [▶ 368]	Struktur zum Betrieb des Programm-Objekts
ST_BACnet_ProgramHandshakeStates [▶ 368]	Struktur zum Betrieb des Programm-Objekts
ST_BACnet_RecipientListDevice [▶ 368]	Struktur für den Datenaustausch der Property *Recipient_List* mit Hilfe des Funktionsbausteins FB_BACnet_RecipientListProperty [▶ 296]
ST_BACnet_RecipientListDeviceEntry [▶ 368]	Teilstruktur für den Datenaustausch der Property *Recipient_List* mit Hilfe des Funktionsbausteins FB_BACnet_RecipientListProperty
ST_BACnet_ServicesSupported [▶ 369]	PLC Abbildung des BACnet Datentyps BACnetServicesSupported. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Protocol_Services_Supported.*
ST_BACnet_StatusFlags [▶ 370]	PLC Abbildung des BACnet Datentyps BACnetStatusFlags. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Status_Flags*.
ST_BACnet_Time [▶ 371]	PLC Abbildung des BACnet Datentyps Time. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime.
ST_BACnet_TimeValue [▶ 371]	PLC Abbildung des BACnet Datentyps BACnetTimeValue für Einträge vom Typ Bool oder Null. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetTimeValue.

Datentypen	Beschreibung
ST_BACnet_TimeValueBool [▶ 371]	PLC Abbildung des BACnet Datentyps BACnetTimeValue für Einträge vom Typ Bool.
ST_BACnet_TimeValueList [▶ 372]	Liste von BACnet_TimeValue Einsträgen.
ST_BACnet_Value [▶ 372]	Teilstruktur von ST_BACnet_TimeValue
ST_BACnet_WeeklyScheduleBool [▶ 372]	Struktur für den Datenaustausch der Property *Weekly_Schedule* mit Hilfe des Funktionsbausteins FB_BACnet_WeeklyScheduleProperty [▶ 299]

BACnet Enumerationen

Datentypen	Beschreibung
E_BACNETACTION [▶ 334]	PLC-Abbildung des BACnet Datentyps BACnetAction. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Action.*
E_BACNETADAPTERSTATUS [▶ 335]	Status des BACnet-Adapters.
E_BACNETBINARYPV [▶ 335]	PLC-Abbildung des BACnet Datentyps BACnetBinaryPV. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Present_Value* von Binary* Objekten.
E_BACNETDATATYPES [▶ 336]	Auflistung der möglichen BACnet Datentypen (Auszug).
E_BACNETDAYSOFWEEKBITS [▶ 338]	Bit-Belegung der Wochentage
E_BACNETDEVICESTATUS [▶ 338]	Status des BACnet Server Objekts (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet Device Objekt und Property *System_Status).*
E_BACNETEVENTSTATE [▶ 339]	PLC-Abbildung des BACnet Datentyps BACnetEventState. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Event_State*.
E_BACNETEVENTTRANSITIONBIT [▶ 340]	Bit-Belegung der Event-Transition-Flags [▶ 361] Event_Enable und *Acked_Transitions*).
E_BACNETEVENTTYPE [▶ 340]	PLC-Abbildung des BACnet Datentyps BACnetEventType. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Event_Type*.
E_BACNETFILEACCESSMETHOD [▶ 341]	PLC-Abbildung des BACnet Datentyps BACnetFileAccessMethod. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *File_Access_Method*.
E_BACNETLIFESAFETYMODE [▶ 341]	PLC-Abbildung des BACnet Datentyps BACnetLifeSafetyMode. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Mode* und *Accepted_Modes*.
E_BACNETLIFESAFETYOPERATION [▶ 342]	PLC-Abbildung des BACnet Datentyps BACnetLifeSafetyOperation. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Operation_Expected*.
E_BACNETLIMITENABLEBITS [▶ 342]	Bit-Belegung der Limit-Enable-Flags [▶ 365] (Limit_Enable).
E_BACNETLOGGINGTYPE [▶ 343]	PLC-Abbildung des BACnet Datentyps BACnetLoggingType. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Logging_Type*.
E_BACNETLOOPMODE [▶ 343]	Betriebsarten des PLC LOOP Objekts FB_BACnet_LOOP [▶ 204]
E_BACNETNOTIFYTYPE [▶ 343]	PLC-Abbildung des BACnet Datentyps BACnetNotifyType. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Notify_Type*.
E_BACNETOBJECTSUPPORTEDBITS [▶ 344]	Bit-Belegung der Object-Types-Supported-Flags Protocol_Object_Types_Supported [▶ 367]).

Datentypen	Beschreibung
E_BACNETOBJECTTYPE [▶ 344]	Auflistung der möglichen BACnet Objekte (Auszug).
E_BACNETPERSISTENTDATASTATE [▶ 345]	PLC-Abbildung des herstellerspezifischen BACnet Datentyps PersistentDataState (siehe BACnet Device Objekt
E_BACNETPIDTUNINGMODE [▶ 346]	Tuningmodi des Bausteins FB_BACnet_LOOP
E_BACNETPOLARITY [▶ 346]	PLC-Abbildung des BACnet Datentyps BACnetPolarity. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Polarity*.
E_BACNETPRIORITY [▶ 346]	Auflistung der möglichen BACnet Prioritäten einer kommandierbaren Property (z.B. *Present_Value*).
E_BACNETPROGRAMERROR [▶ 347]	PLC-Abbildung des BACnet Datentyps BACnetProgramError. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Reason_For_Halt* und Funktionsbausteinbeschreibung FB_BACnet_Program [▶ 222]
E_BACNETPROGRAMREQUEST [▶ 348]	PLC-Abbildung des BACnet Datentyps BACnetProgramRequest. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Program_Change* und Funktionsbausteinbeschreibung FB_BACnet_Program
E_BACNETPROGRAMSTATE [▶ 348]	PLC-Abbildung des BACnet Datentyps BACnetProgramState. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Program_State* und Funktionsbausteinbeschreibung FB_BACnet_Program
E_BACNETPROPERTYIDENTIFIER [▶ 348]	Auflistung der möglichen BACnet Properties (Auszug).
E_BACNETRELIABILITY [▶ 352]	Auflistung der möglichen Werte der BACnet Property *Reliability* (Auszug).
E_BACNETSEGMENTATION [▶ 352]	PLC-Abbildung des BACnet Datentyps BACnetSegmentation. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Segmentation_Supported*.
E_BACNETSILENCEDSTATE [▶ 353]	PLC-Abbildung des BACnet Datentyps BACnetSilencedState. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Silenced*.
E_BACNETSTATUSFLAGS [▶ 354]	Bit-Belegung der Status-Flags [▶ 370] (Status_Flags).
E_BACNETSTRINGENCODINGTYPES [▶ 354]	Die Enumeration enthält eine Auflistung der vom BACnet Treiber unterstützten Zeichenketten Encodings.
E_BACNETUNIT [▶ 354]	PLC-Abbildung des BACnet Datentyps BACnetEngineeringUnits. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Units*.

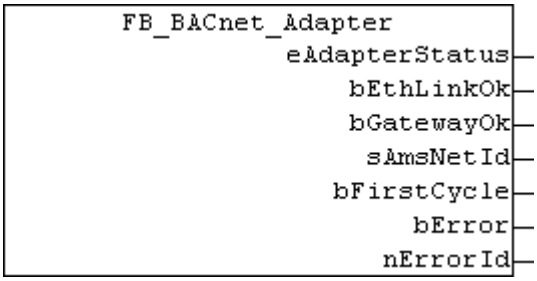
4.1 FB_BACnet_Adapter

Funktionsbaustein für die Anbindung des PLC Programms an einen lokalen BACnet-Adapter (Netzwerkkarte). Die Verknüpfung des Funktionsbausteins erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell oder mittels PLC-Automapping [▶ 64] automatisch verknüpft werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.

Achtung:

Dieser Baustein darf nicht im PLC Programm instanziiert werden! Eine Instanz des Bausteins ist bereits in den globalen Daten der PLC Library enthalten (*fbBACnet_Adapter*) und wird dort automatisch vom PLC Automapping erfasst. Die Instanziierung im PLC Programm ist nur bei Verwendung mehrer BACnet-Adapter in einem PLC Programm nötig!



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Adapter wird der Zustand des lokalen BACnet Adapters gelesen (Prozessdatum *Device Status*) und im PLC Programm ausgegeben. Zudem wird mit Hilfe des Prozessdatums *AmsNetId* die AMS-NetID des BACnet-Adapters gelesen und an **sAmsNetId** ausgegeben.

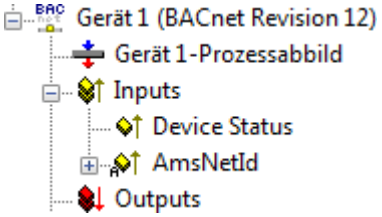


Abb. 5: Bild-1: Prozessdaten des BACnet Adapters im System Manager.

Wie bereits oben beschrieben, wird eine Instanz des Bausteins FB_BACnet_Adapter bereits durch die PLC Library als globale Instanz bereitgestellt und darf daher *nicht* explizit angelegt werden. Die globale Instanz durch das PLC-Automapping erkannt und automatisch mit dem entsprechenden BACnet-Adapter im System Manager verknüpft.

Die Bausteininstanz wird von den BACnet Funktionsbausteinen FB_BACnet_Device [▶ 199] und FB_BACnet_RemoteDevice [▶ 243] benötigt. Im Folgenden ist die Hierarchie zwischen FB_BACnet_Adapter, FB_BACnet_Device [▶ 199] und einem BACnet Objekt vom Typ AnalogValue dargestellt:

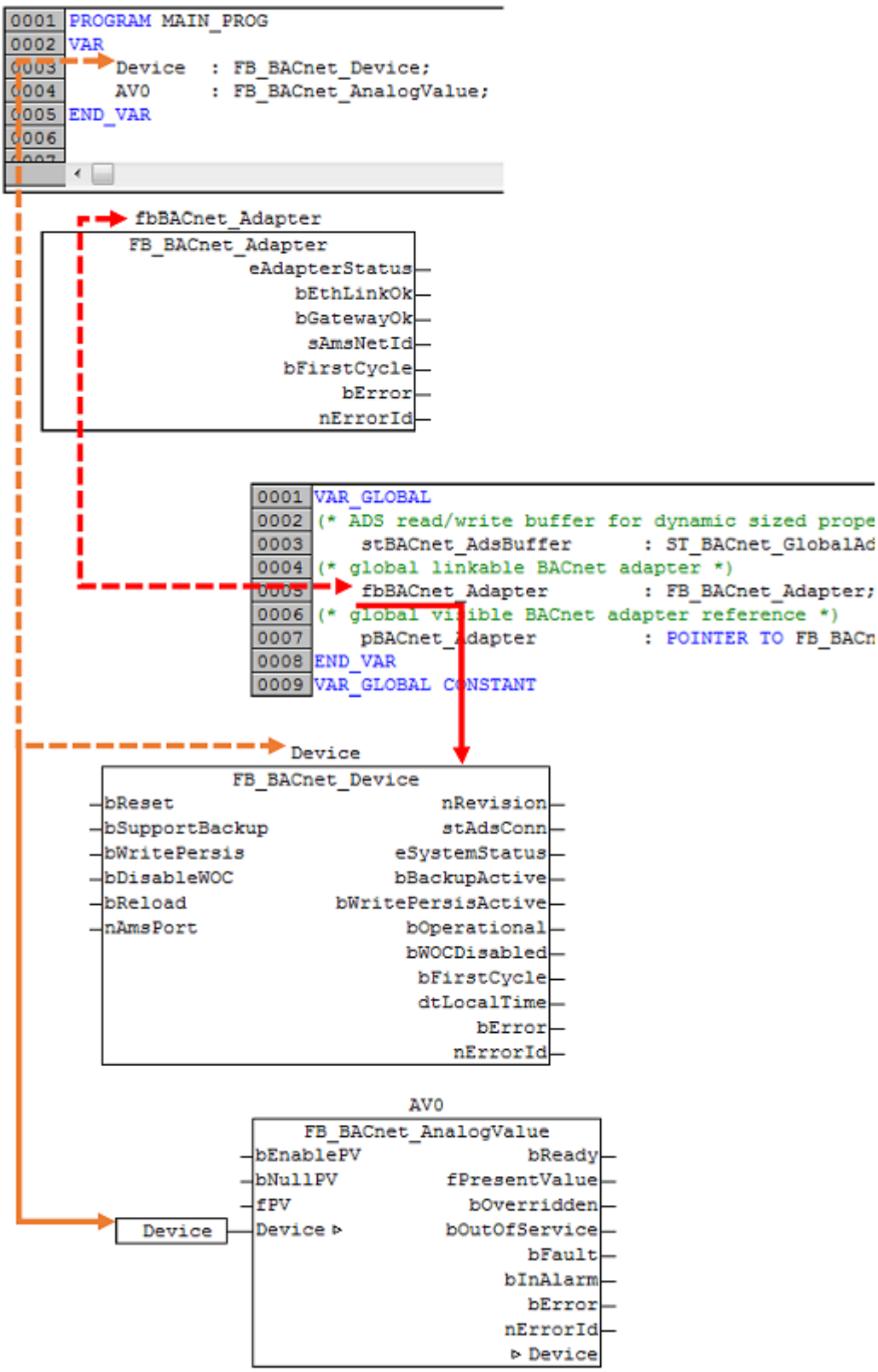


Abb. 6: Bild-2: Abhängigkeit zwischen FB_BACnet_Adapter, FB_BACnet_Device und eines BACnet-Funktionsbausteins (Bsp.: FB_BACnet_AnalogValue).

- **Rot-gestrichelt:** Funktionsbausteinaufruf FB_BACnet_Adapter aus den globalen Daten der PLC Library. Jeder Aufruf aktualisiert den globalen Pointer *pBACnet_Adapter*, den die Bausteine FB_BACnet_Device [▶ 199] und FB_BACnet_RemoteDevice [▶ 243] als Adapter-Referenz verwenden.
- **Rot:** Device Baustein FB_BACnet_Device [▶ 199] und FB_BACnet_RemoteDevice [▶ 243] intern, wird die globale Instanz des BACnet Adapters verwendet, um den Status der BACnet Verbindung zu prüfen und die AMS-NetID zu lesen. Dabei wird die Instanz über den globlen Pointer *pBACnet_Adaper* ermittelt. Der globale Pointer kann ebenfalls im PLC Programm, vor Aufruf von FB_BACnet_Device [▶ 199] und FB_BACnet_RemoteDevice [▶ 243], mit der eigenen Adapter Instanz überschrieben werden. So können beispielsweise PLC Programme mit mehreren Adaptern realisiert werden.
- **Orange-gestrichelt:** Funktionsbausteinaufruf FB_BACnet_Device [▶ 199] mit der Instanz aus dem PLC Programm. Die Instanz wird vom Anwender im PLC Programm lokal oder global angelegt und vom PLC Automapping mit dem entsprechenden Device im TwinCAT System Manager verknüpft.
- **Orange:** Übergabe der Device Instanz (FB_BACnet_Device [▶ 199]) an den Funktionsbaustein des BACnet Objekts. Der Funktionsbaustein des BACnet Objekts liest u.a. den Devicestatus aus der Instanz.

Tipp:

Die Instanz des Bausteins FB_BACnet_Adapter muss nicht im PLC Programm aufgerufen werden. Der Aufruf des Adapter wird bei Bedarf automatisch von den Bausteinen FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243] vorgenommen.

VAR_OUPUT

```
eAdapterStatus  : E_BACNETADAPTERSTATUS;
bEthLinkOk      : BOOL;
bGatewayOk      : BOOL;
sAmsNetId       : T_AmsNetId;
bFirstCycle     : BOOL;
bError          : BOOL;
nErrorId        : UINT;
```

eAdapterStatus: Aktueller Status des BACnet-Adapters; siehe: E_BACNETADAPTERSTATUS [▶ 335].

bEthLinkOk: Die lokale Ethernet-Verbindung ist aktiv (Kabel gesteckt, Adapter verbunden), wenn der Ausgang auf *TRUE* gesetzt ist.

bGatewayOk: Das konfigurierte Gateway ist erreichbar, wenn der Ausgang auf *TRUE* gesetzt ist.

sAmsNetId: Ausgabe der AMS-NetID des lokalen BACnet-Adapters (kann für den asynchronen Zugriff via ADS auf BACnet-Objekte verwendet werden).

bFirstCycle: Wird mit dem ersten Aufruf der Bausteininstanz nach SPS-Reset bzw. -Neustart für einen Zyklus gesetzt.

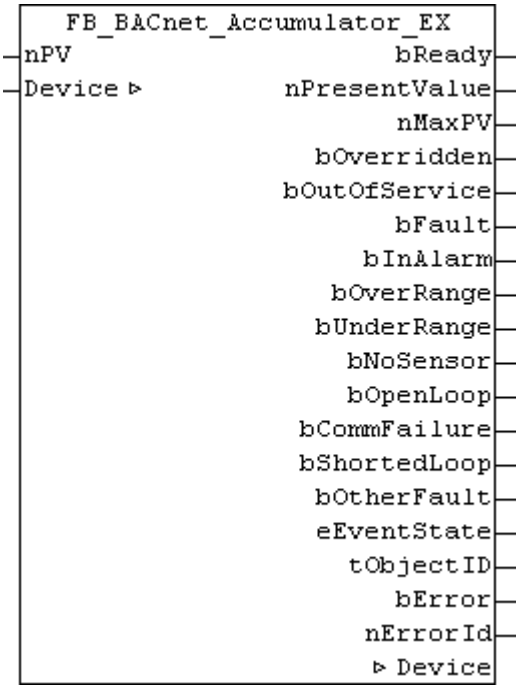
bError: Ein Fehler steht an.

nErrorId: Fehlercode, siehe BACnet_Globals [▶ 330] für eine Übersicht.

Bei fehlender Netzwerkverbindung (**bEthLinkOk** = *FALSE*) bzw. nicht erreichbarem Gateway (**bGatewayOk** = *FALSE*), werden sämtliche Remote-Objekte (BACnet Objekte unter einem Clients) gesperrt. Objekte des lokale BACnet Servers sind davon nicht betroffen.

4.2 BACnet Server Objects

4.2.1 FB_BACnet_Accumulator_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Accumulator kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Accumulator* (ACC) zugegriffen werden.

Der typische Anwendungsfall eines Accumulator-Objekt, ist die Abbildung von Zählerständen nach BACnet. Unter Beispiel [▶ 161] ist die Anbindung eines M-Bus Zählers an ein Accumulator-Objekt dargestellt.

Der Unterschied in der Basis- und *_EX*-Variante des Bausteins, besteht zum einen in der Art die Property *Present_Value* zu schreiben und zum anderen in der Anzahl der Statusausgänge. In der Basis-Variante wird mit Hilfe der Property *Value_Set* in das Objekt geschrieben. Vor der Wertübernahme in die Property *Present_Value* wird der vorherige Wert inklusive Zeitstempel gesichert. So kann anhand der Property *Value_Change_Time* z.B. erkannt werden, wann die Property *Present_Value* zuletzt aktualisiert wurde. Im Falle eines externen Zählers (Bsp. M-Bus), kann so die Aktualität des Werts überprüft werden.

VAR_INPUT

Grau dargestellte Variablen sind nicht in der Basisversion des Bausteins enthalten. Schwarz dargestellte Variablen sind nicht in der *_EX*-Variante des Bausteins enthalten.

```
nPV         : UDINT;
nValueSet   : UDINT;
```

nPV: Wert der in die Property *Present_Value* geschrieben werden soll. Das Schreiben der Prozessdaten erfolgt bei Änderung.

nValueSet: Wert der in die Property *Value_Set* geschrieben werden soll. Ein Schreibzugriff löst zudem die Aktualisierung der Property *Value_Change_Time* und *Value_Before_Change* aus. Das Schreiben der Prozessdaten erfolgt bei Änderung.

VAR_OUPUT

```
bReady          : BOOL;
nPresentValue   : UDINT;
nMaxPV          : UDINT;
```

```
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bOverRange      : BOOL;
bUnderRange     : BOOL;
bNoSensor       : BOOL;
bOpenLoop       : BOOL;
bCommFailure    : BOOL;
bShortedLoop    : BOOL;
bOtherFault     : BOOL;
eEventState     : E_BACNETEVENTSTATE;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Present_Value*).

nMaxPV: Aktueller Wert der Property *Max_Pres_Value* des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Max_Pres_Value*).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Status_Flags*.

bOverRange, bUnderRange, bNoSensor, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Event_State*.

tObjectID: Objekt ID des BACnet Objekts (Objekt Type und Objekt Instanz).

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten (BACnet Globals [▶ 330]).

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung eines M-Bus Zählers auf ein BACnet Objekt vom Typ *Accumulator* gezeigt.

```
0003 (* M-BUS *)
0004     MBUS         : FB_MBUSKL6781;
0005     MBUSCom      : ST_MBUS_Communication;
0006     MBUSRaw      : FB_MBUS_RawData;
0007
0008 (* BACnet *)
0009     Device       : FB_BACnet_Device;
0010     pAccValue    : POINTER TO UDINT;
0011     ACC          : FB_BACnet_Accumulator;
0012         (* ~(BACnet_ObjectName   : MAIN.ACC : NOLINK) *)
0013     ACCRel       AT%Q*:WORD;
0014         (* ~(BACnet_ObjectType   : ACC : NOLINK)
0015             (BACnet_ObjectName   : MAIN.ACC : OBJREF, NOLINK)
0016             (BACnet_Reliability : : LINK) *)
```

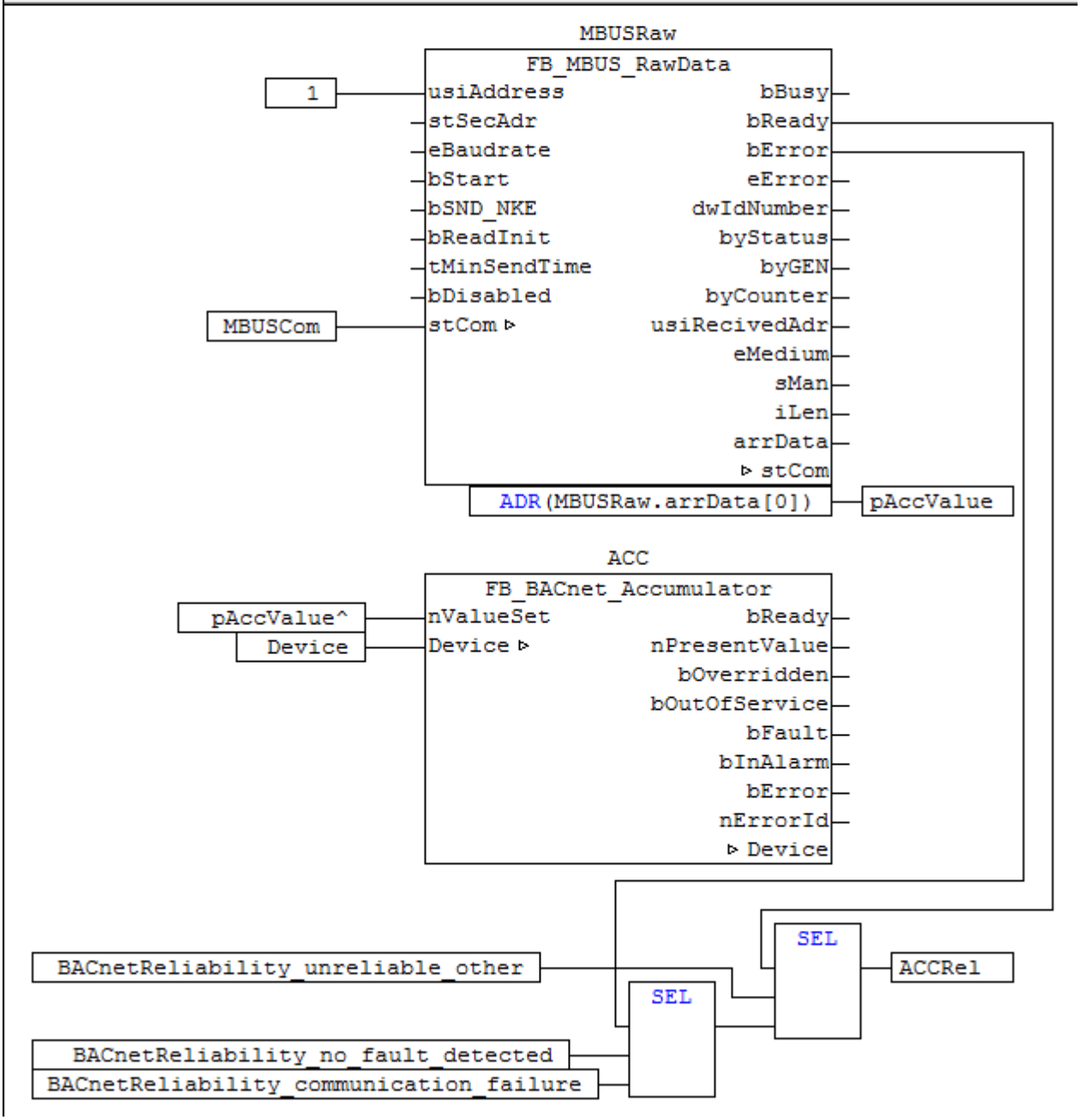
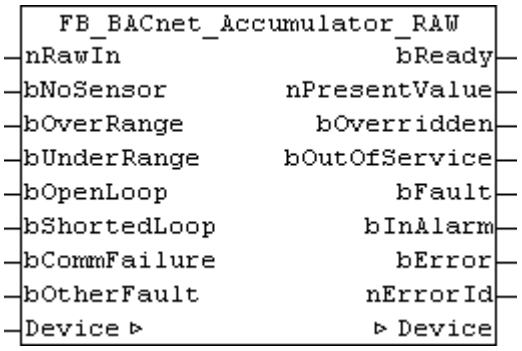


Abb. 7: Bild-1: Beispiel für die Umsetzung eines M-Bus Zählers in das BACnet Objekt Accumulator im PLC Programm

Dabei wird seitens BACnet Objekt der Wert des Eingangs nValueSet erfasst und als Property Present_Value übernommen. Als weitere Information dienen der Status bError und bReady des M-Bus Bausteins für das Setzen der Property Reliability des BACnet Objekts. Diese Property gibt an, in welchem Zustand sich der Wert des BACnet Objekts befindet. Meldet der M-Bus Baustein bReady = TRUE, dann geht das BACnet Objekt in no_fault_detected → Wert gültig.

4.2.2 FB_BACnet_Accumulator_RAW



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Accumulator_RAW kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Accumulator* zugegriffen werden.

Im Unterschied zur Standard- bzw. *_EX*-Variante des Bausteins, wird der Rohwert und der Zustand der Property *Reliablitity* durch Funktionsbaustein-Eingänge bereitgestellt und nicht durch die IO-Hardware direkt. So kann z.B. der Zustand eines Zähleingangs aus einem Sub-Bussystems in SPS-Code auf ein BACnet-Objekt abgebildet werden (Signalumsetzung von Sub-Bussystemen oder virtuellen Datenpunkten nach BACnet, siehe Beispiel [▶ 164]).

VAR_INPUT

```
nRawIn          : UINT;
bNoSensor       : BOOL;
bOverRange      : BOOL;
bUnderRange     : BOOL;
bOpenLoop       : BOOL;
bShortedLoop    : BOOL;
bCommFailure    : BOOL;
bOtherFault     : BOOL;
```

nRawIn: Rohwerteingang des Objekts im Wertebereich -32768...32767. Der Eingang wird mit dem Prozessdatum "RawIoAccumulatorUnsignedValue" des BACnet-Objekts verknüpft. Wertänderungen von **nRawIn** werden mit der Property *Prescale* zum Wert der Property *Present_Value* verrechnet (vorausgesetzt der Objektzustand ist nicht *out_of_service*).

bNoSensor, bOverRange, bUnderRange, bOpenLoop, bShortedLoop, bCommFailure, bOtherFault: *TRUE* am Eingang setzt den entsprechenden Zustand [▶ 352] der Property *Reliability*. Die Priorität fällt mit der Reihenfolge der Eingänge (**bNoSensor** höchste und **bOtherFault** niedrigste Priorität). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Reliability*.

VAR_OUPUT

```
bReady          : BOOL;
nPresentValue   : UDINT;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Present_Value*).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung eines digitalen Impulssignals auf ein BACnet Objekt vom Typ *Accumulator* gezeigt.

```
0003 (* Virtual value *)
0004     diPulseInput AT%I*:BOOL;
0005     diPulseOk    AT%I*:BOOL;
0006     rtrigPulse   : R_TRIG;
0007     nCounter     : UINT;
0008
0009 (* BACnet *)
0010     Device       : FB_BACnet_Device;
0011     ACCRawCount  : FB_BACnet_Accumulator_RAW;
```

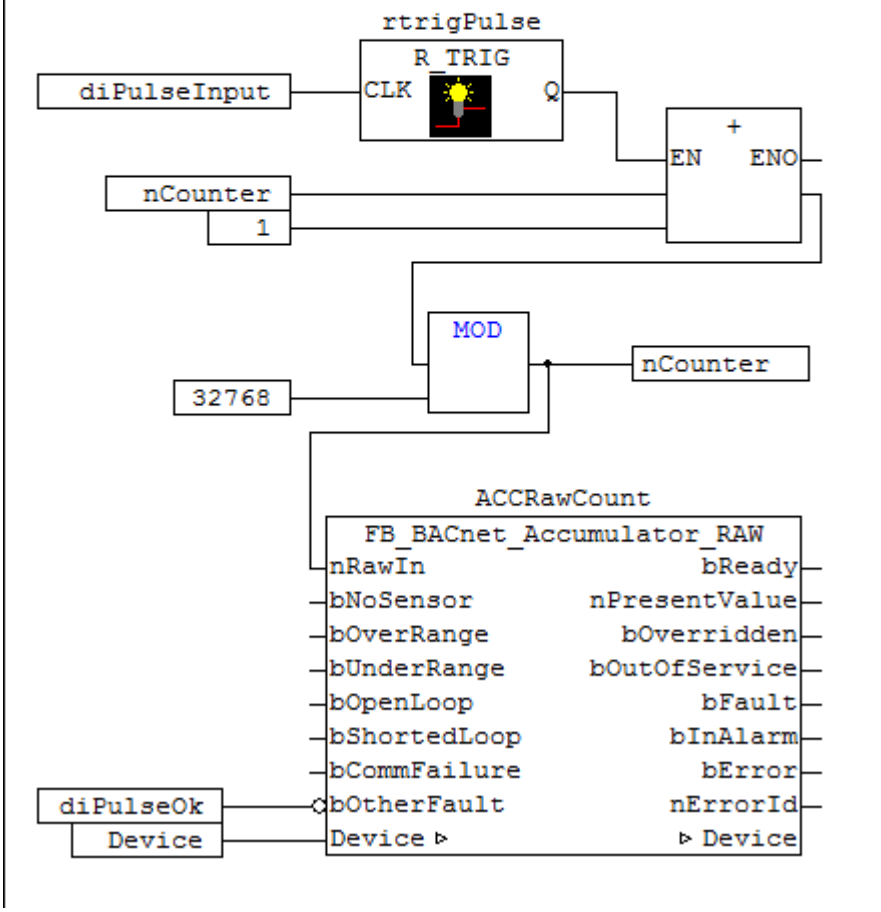
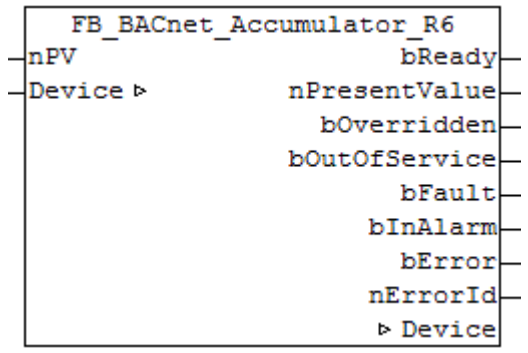


Abb. 8: Bild-1: Beispiel für die Umsetzung eines Impulssignals in das BACnet Objekt Accumulator im PLC Programm

Dabei wird seitens BACnet Objekt die Änderungen des Eingangs nRawIn erfasst. Ein Überlauf des UINT Wertes ist unkritisch, da dieser vom BACnet Objekt entsprechend behandelt wird. Gezählt wird die Diffenrenz des Eingangswerts im Modul 32768. Als weitere Information dient der Eingang diPulseOk. Dieser Eingang gibt an, dass die impulsgenerierende Seite (z.B. ein exterener Zähler) bereit ist.

4.2.3 FB_BACnet_Accumulator_R6



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Accumulator_R6 kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Accumulator* zugegriffen werden. Der schreibende Zugriff verhält sich bei dieser Variante (*_R6*) wie in der vorherigen Library Version (Revision 6): Der Wert von **nPV** wird direkt in die Property *Present_Value* übernommen.

VAR_INPUT

```
nPV         : UDINT;
```

nPV: Wert der in die Property *Present_Value* direkt geschrieben wird.

VAR_OUPUT

```
bReady          : BOOL;
nPresentValue   : UDINT;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Present_Value*).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Status_Flags*.

bError: Ein Fehler steht an.

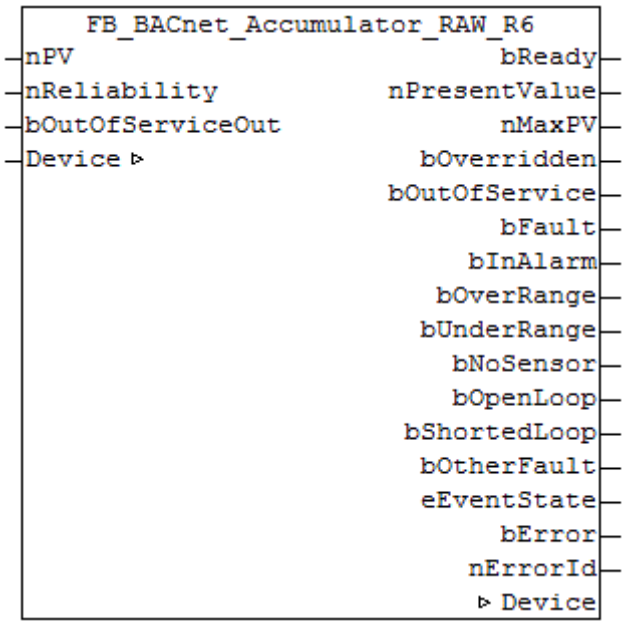
nErrorId: siehe globale Konstanten (BACnet Globals [▶ 330]).

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.4 FB_BACnet_Accumulator_RAW_R6



Anwendung

Mit Hilfe des Funktionsbausteins kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Accumulator* (ACC) zugegriffen werden.

Im Unterschied zur Standard- bzw. *_EX*-Variante des Bausteins, wird der aktuelle Zählerwert (*Present_Value*) und der Zustand der Property *Reliablitity* durch Funktionsbaustein-Eingänge bereitgestellt und nicht durch die IO-Hardware direkt. So kann z.B. der Zustand eines Zähleingangs aus einem Sub-Bussystems in SPS-Code auf ein BACnet-Objekt abgebildet werden (Signalumsetzung von Sub-Bussystemen oder virtuellen Datenpunkten nach BACnet, siehe Beispiel [▶ 168]).

VAR_INPUT

```
nPV               : UDINT;
nReliability      : WORD;
bOutOfServiceOut  : BOOL;
```

nPV: Wert der in die Property *Present_Value* direkt geschrieben wird.

nReliability: Eingang setzt den entsprechenden Zustand [▶ 352] der Property *Reliability*. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Reliability*.

bOutOfServiceOut: *TRUE* setzt den Zustand des Objekts auf *out_of_service*. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Status_Flags*.

VAR_OUPUT

```
bReady          : BOOL;
nPresentValue   : UDINT;
nMaxPV          : UDINT;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bOverRange      : BOOL;
bUnderRange     : BOOL;
bNoSensor       : BOOL;
bOpenLoop       : BOOL;
bShortedLoop    : BOOL;
bOtherFault     : BOOL;
eEventState     : E_BACNETEVENTSTATE;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Present_Value*).

nMaxPV: Aktueller Wert der Property *Max_Pres_Value* des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Max_Pres_Value*).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Status_Flags*.

bOverRange, bUnderRange, bNoSensor, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Event_State*.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device        : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung eines digitalen Impulssignals auf ein BACnet Objekt vom Typ *Accumulator* gezeigt.

```
0002 VAR PERSISTENT
0003 (* Persistent counter value *)
0004     nCounter      : UDINT;
0005 END_VAR
0006 VAR
0007 (* Virtual value *)
0008     diPulseInput AT%I*:BOOL;
0009     diPulseOk    AT%I*:BOOL;
0010     rtrigPulse   : R_TRIG;
0011 (* BACnet *)
0012     Device       : FB_BACnet_Device;
0013     ACCRawCount  : FB_BACnet_Accumulator_RAW_R6;
0014 END_VAR
```

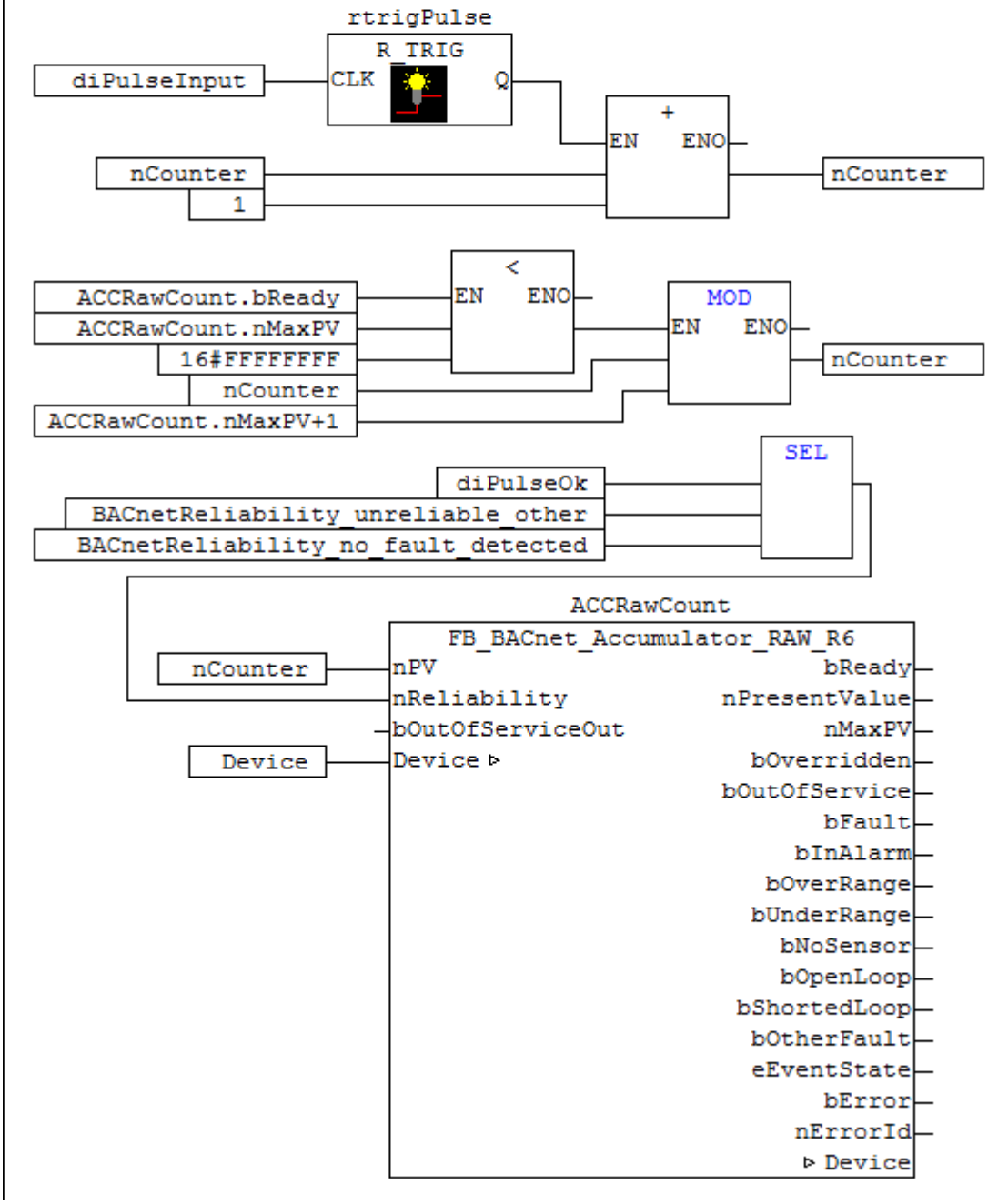
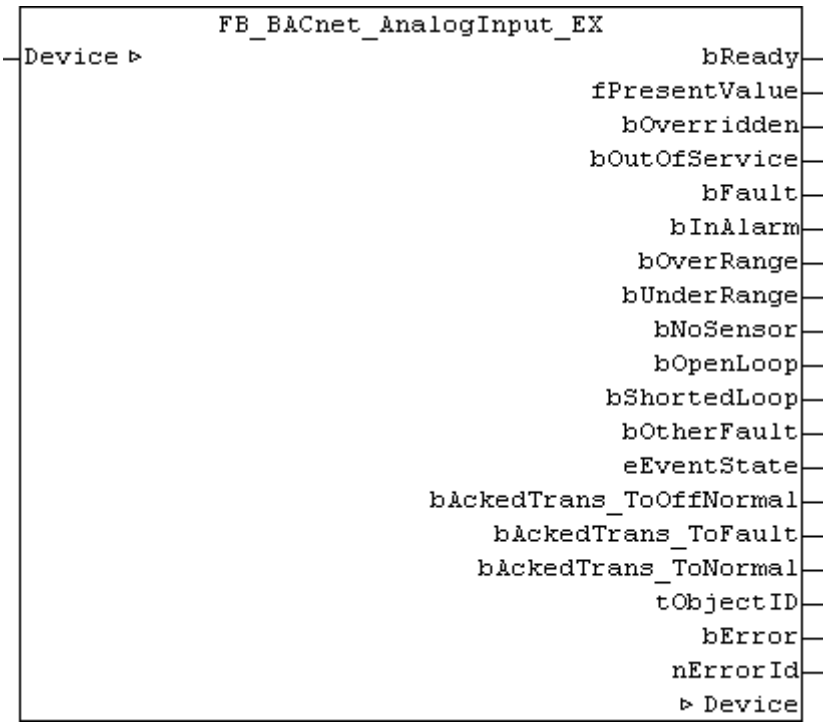


Abb. 9: Bild-1: Beispiel für die Umsetzung eines Impulssignals in das BACnet Objekt Accumulator im PLC Programm.

Dabei wird seitens BACnet Objekt der Zählwert am Eingang nPV erfasst. Ein Überlauf des UDINT Wertes erfolgt bei eingestelltem Maximalwert oder datentypbedingt. Als weitere Information dient der Eingang diPulseOk. Dieser Eingang gibt an, dass die impulsgenerierende Seite (z.B. ein exterener Zähler) bereit ist.

4.2.5 FB_BACnet_AnalogInput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_AnalogInput kann lesend auf ein BACnet-Objekt vom Typ *AnalogInput* (AI) zugegriffen werden.

VAR_OUPUT

```
bReady                  : BOOL;
fPresentValue           : REAL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bOverRange              : BOOL;
bUnderRange             : BOOL;
bNoSensor               : BOOL;
bOpenLoop               : BOOL;
bShortedLoop            : BOOL;
bOtherFault             : BOOL;
eEventState             : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
tObjectID               : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft. dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Present_Value*).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Status_Flags*.

bOverRange, bUnderRange, bNoSensor, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property Event_State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Acked_Transitions*).

tObjectID: Objekt ID des BACnet Objekts (Objekt Type und Objekt Instanz).

bError: Ein Fehler steht an.

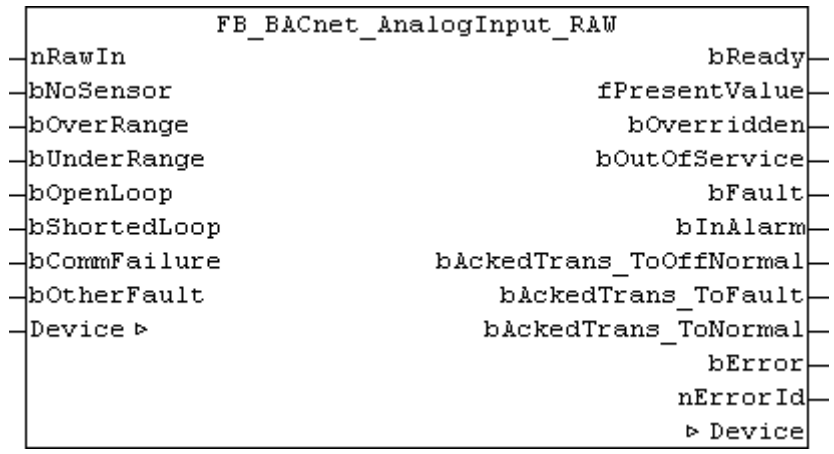
nErrorId: siehe globale Konstanten (BACnet_Globals [▶ 330]).

VAR_IN_OUT

```
Device                : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.6 FB_BACnet_AnalogInput_RAW



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_AnalogInput_RAW kann lesend und schreibend auf ein BACnet-Objekt vom Typ *AnalogInput* zugegriffen werden.

Im Unterschied zur Standard- bzw. *_EX*-Variante des Bausteins, wird der Rohwert und der Zustand der Property *Reliablitity* durch Funktionsbaustein-Eingänge bereitgestellt und nicht durch die IO-Hardware direkt. So kann z.B. der Zustand eines Analogwerts aus einem Sub-Bussystems in SPS-Code auf ein BACnet-Objekt abgebildet werden (Signalumsetzung von Sub-Bussystemen oder virtuellen Datenpunkten nach BACnet, siehe Beispiel [▶ 172]).

VAR_INPUT

```
nRawIn                : INT;
bNoSensor             : BOOL;
bOverRange            : BOOL;
bUnderRange           : BOOL;
bOpenLoop             : BOOL;
```

```
bShortedLoop          : BOOL;
bCommFailure          : BOOL;
bOtherFault           : BOOL;
```

nRawIn: Rohwerteingang des Objekts im Wertebereich -32768...32767. Der Eingang wird mit dem Prozessdatum "RawIoAnalogSignedValue" des BACnet-Objekts verknüpft. Der Wert aus **nRawIn** wird mit der Property *Resolution* zum dem Wert der Property *Present_Value* verrechnet (vorausgesetzt der Objektzustand ist nicht *out_of_service*).

bNoSensor, bOverRange, bUnderRange, bOpenLoop, bShortedLoop, bCommFailure, bOtherFault: *TRUE* am Eingang setzt den entsprechenden Zustand [▶ 352] der Property *Reliability*. Die Priorität fällt mit der Reihenfolge der Eingänge (**bNoSensor** höchste und **bOtherFault** niedrigste Priorität). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Reliability*.

VAR_OUPUT

```
bReady                    : BOOL;
fPresentValue             : REAL;
bOverridden               : BOOL;
bOutOfService             : BOOL;
bFault                    : BOOL;
bInAlarm                  : BOOL;
bAckedTrans_ToOffNormal   : BOOL;
bAckedTrans_ToFault       : BOOL;
bAckedTrans_ToNormal      : BOOL;
bError                    : BOOL;
nErrorId                  : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Present_Value*).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Status_Flags*.

bOverRange, bUnderRange, bNoSensor, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Event_State*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Acked_Transitions*).

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device                : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung eines ganzzahligen Werts aus EIB in die Property *Present_Value* eines AnalogInput-Objekts gezeigt:

```
0002 VAR
0003 (* EIB *)
0004     EIB          : KL6301;
0005     EIBAnIn      : EIB_2OCTET_SIGN_REC;
0006
0007 (* BACnet *)
0008     Device       : FB_BACnet_Device;
0009     AIRaw        : FB_BACnet_AnalogInput_RAW;
0010 END_VAR
```

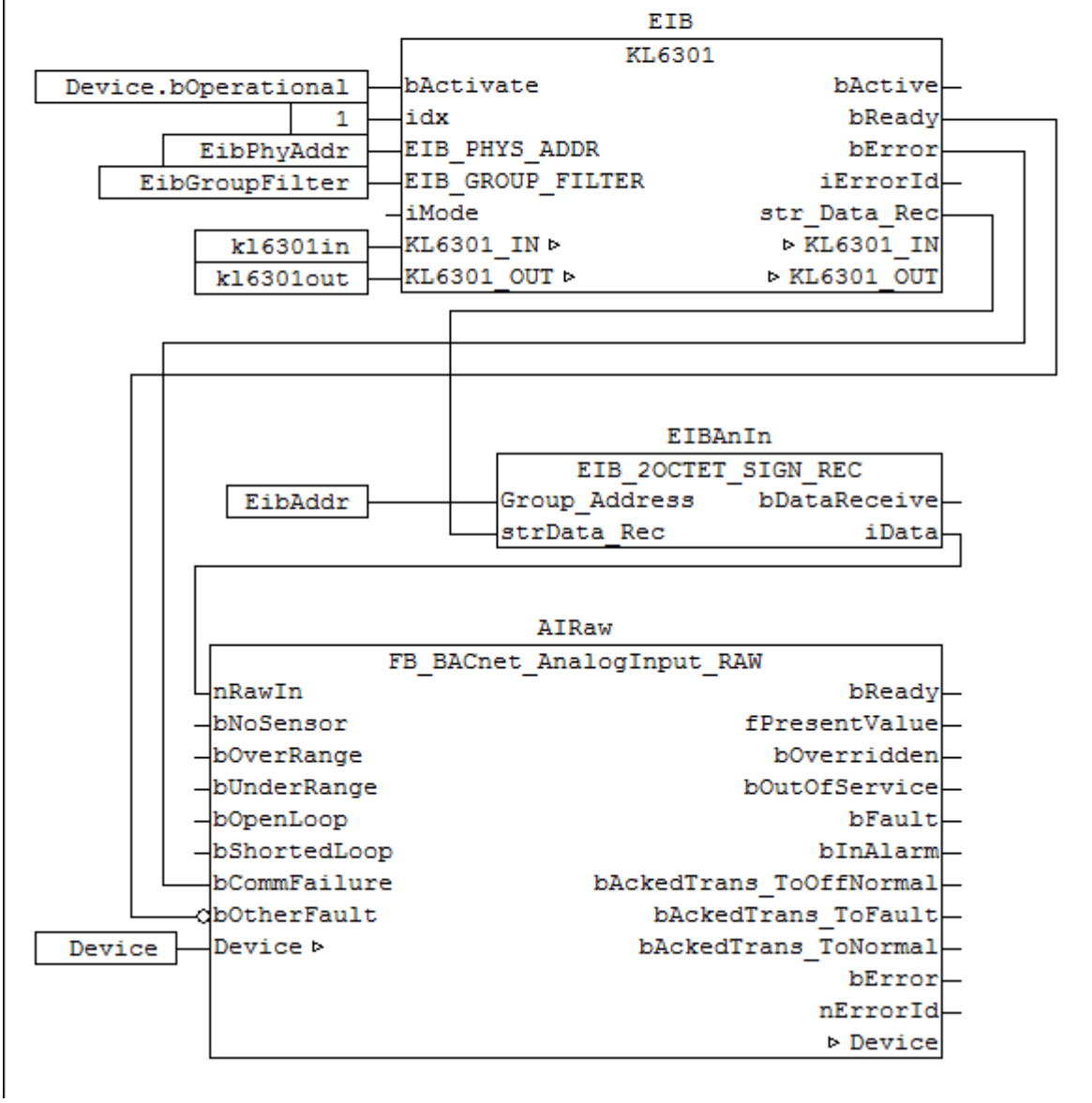
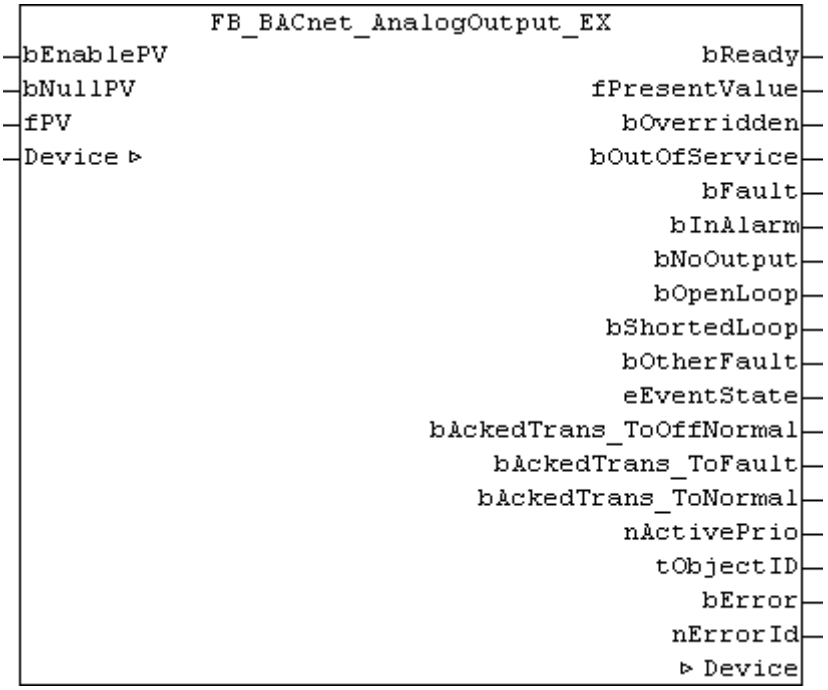


Abb. 10: Bild-1: Beispiel für die Umsetzung eines ganzzahligen Werts aus EIB in die Property Present_Value im PLC Programm.

Dieses Beispiel setzt die Bibliothek "TcEIB.lib" voraus. Siehe Dokumentation zur Klemme KL6301 für weitere Informationen zu EIB.

4.2.7 FB_BACnet_AnalogOutput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_AnalogOutput kann lesend und mit Priorität 12 auf die Property *Present_Value* schreibend auf ein BACnet-Objekt vom Typ AnalogOutput (AO) zugegriffen werden.

VAR_INPUT

```
bEnablePV   : BOOL;
bNullPV     : BOOL;
fPV         : REAL;
```

bEnablePV: *TRUE* = Schreibfreigabe des Property-Werts; *FALSE* = Eintrag im BACnet Objekt nicht verändern (**bNullPV** und **fPV** werden unwirksam).

bNullPV: *TRUE* = Eintrag der Property löschen (*NULL*); anstelle des Werts von **fPV**; *FALSE* = Wert von **fPV** als Property Wert schreiben.

fPV: Wert der in die Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** = *TRUE* und **bNullPV** = *FALSE* sind. Das Schreiben der Prozessdaten erfolgt bei Änderung.

VAR_OUPUT

```
bReady                  : BOOL;
fPresentValue           : REAL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bNoOutput               : BOOL;
bOpenLoop               : BOOL;
bShortedLoop            : BOOL;
bOtherFault             : BOOL;
eEventState             : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
nActivePrio             : UINT;
tObjectID               : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Status_Flags*.

bNoOutput, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property Event_State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Acked_Transitions*).

nActivePrio: Gibt die aktuell wirksame Priorität des Priority-Array an, die auf das *Present_Value* wirkt (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

tObjectID: Objekt ID des BACnet Objekts (Objekt Type und Objekt Instanz).

bError: Ein Fehler steht an.

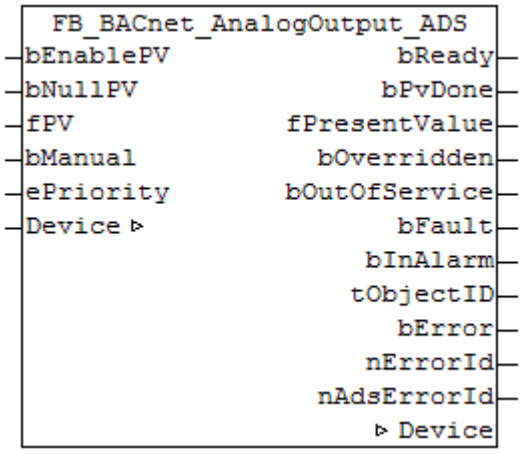
nErrorId: siehe globale Konstanten (BACnet_Globals [▶ 330]).

VAR_IN_OUT

```
Device              : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.8 FB_BACnet_AnalogOutput_ADS



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_AnalogOutput_ADS kann lesend und schreibend auf ein BACnet-Objekt vom Typ *AnalogOutput* zugegriffen werden.

Das Schrieben der Property *Present_Value* wird, im Unterschied zur Standard- und *_EX*-Variante, mittels ADS-WRITE realisiert (azyklisch).

VAR_INPUT

```
bEnablePV  : BOOL;
bNullPV    : BOOL;
fPV        : REAL;
bManual    : BOOL;
ePriority  : E_BACNETPRIORITY:=BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs **fPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus **fPV** auf die kommandierbare Property *Present_Value* mit Priorität aus **ePriority** (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Wird der Eingang auf *FALSE* gesetzt, erfolgt kein Schreibzugriff (keine Änderung) mehr.

bNullPV: Überschreibt den Eintrag der kommandierbaren Property *Present_Values* in der Priorität **ePriority** mit dem Wert *NULL*, wenn **bEnablePV** auf *TRUE* gesetzt ist. Der Eingang **fPV** wird ignoriert, so lange **bNullPV** auf *TRUE* gesetzt ist.

fPV: Wert der an die entsprechenden Stelle der Property *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** auf *TRUE* und **bNullPV** auf *FALSE* gesetzt ist. Die Priorität wird mit dem Eingang **ePriority** bestimmt (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Das Schreiben wird bei Wertänderung azyklisch ausgeführt (ADS-WRITE).

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung geschieht, ist es potenziell möglich, dass die Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:
- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE* → *TRUE*: Schreiben der Property *Present_Value* bei Flankenwechsel.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 199]).

ePriority: Vorgabe der Priorität für Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 346]).

VAR_OUPUT

```
bReady          : BOOL;
bPvDone         : BOOL;
fPresentValue   : REAL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
nAdsErrorId     : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Status_Flags*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

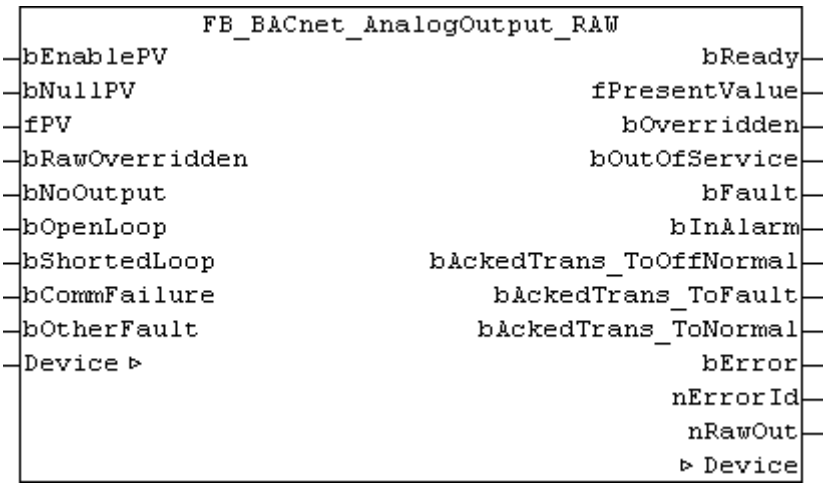
nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.9 FB_BACnet_AnalogOutput_RAW



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_AnalogOutput_RAW kann lesend und schreibend auf ein BACnet-Objekt vom Typ *AnalogOutput* zugegriffen werden.

Im Unterschied zur Standard- bzw. *_EX*-Variante des Bausteins, wird das Flag der Property *Status_Flags* / *overridden* und der Wert der Property *Reliablitity* durch Funktionsbaustein-Eingänge bereitgestellt und aus dem PLC-Programm auf das BACnet Objekt abgebildet. Die Zustand der Property *Present_Value* wird durch das PLC-Programm auf die Hardware bzw. auf das Sub-Bussystem gemappet. Dies geschieht sonst direkt durch die IO-Hardware. So kann z.B. der Ausgangswert an ein Sub-Bussystems in PLC-Code von einem BACnet-Objekt abgebildet werden (Signalumsetzung auf Sub-Bussysteme oder virtuellen Datenpunkte von BACnet, siehe Beispiel [▶ 178]).

VAR_INPUT

```
bEnablePV       : BOOL;
bNullPV         : BOOL;
fPV             : REAL;
bRawOverridden  : BOOL;
bNoOutput       : BOOL;
bOpenLoop       : BOOL;
bShortedLoop    : BOOL;
bCommFailure    : BOOL;
bOtherFault     : BOOL;
```

bEnablePV: *TRUE* = Schreibfreigabe des Property-Werts; *FALSE* = Eintrag im BACnet Objekt nicht verändern (**bNullPV** und **fPV** werden unwirksam).

bNullPV: *TRUE* = Eintrag der Property löschen (*NULL*); anstelle des Werts von **fPV**; *FALSE* = Wert von **fPV** als Property Wert schreiben.

fPV: Wert der in die Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** = *TRUE* und **bNullPV** = *FALSE* sind. Das Schreiben der Prozessdaten erfolgt bei Änderung.

bRawOverridden: *TRUE* setzt das Flag *overridden* der Property *Status_Flags*. Damit wird nach BACnet signalisiert, dass der Wert der Property *Present_Value* von der Hardware-Ausgabe entkoppelt wurde (z.B. durch eine manuelle Vorortbedienung). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Status_Flags*.

bNoOutput, bOpenLoop, bShortedLoop, bCommFailure, bOtherFault: *TRUE* am Eingang setzt den entsprechenden Zustand [▶ 352] der Property *Reliability*. Die Priorität fällt mit der Reihenfolge der Eingänge (**bNoOutput** höchste und **bOtherFault** niedrigste Priorität). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Reliability*.

VAR_OUPUT

```
bReady                  : BOOL;
fPresentValue           : REAL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
nRawOut                 : INT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Status_Flags*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Acked_Transitions*).

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet Globals [▶ 330].

nRawOut: Rohwertausgang des Objekts im Wertebereich -32768...32767. Der Ausgang wird mit dem Prozessdatum "RawIoAnalogSignedValue" des BACnet-Objekts verknüpft. Der Wert von **nRawOut** resultiert aus der Property *Present_Value* mit Verrechnung der Property *Resolution*.

VAR_IN_OUT

```
Device              : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung der Property *Present_Value* eines AnalogOutput-Objekts nach EIB gezeigt:

```
0002 VAR
0003 (* EIB *)
0004     EIB          : KL6301;
0005     EIBAnOut     : EIB_2OCTET_SIGN_SEND;
0006
0007 (* Manual override *)
0008     diManActive AT%I*:BOOL; (* key switch on cabinet *)
0009     aiManSlider AT%I*:INT; (* analog slider on cabinet *)
0010
0011 (* BACnet *)
0012     Device       : FB_BACnet_Device;
0013     AORaw        : FB_BACnet_AnalogOutput_RAW;
0014 END_VAR
```

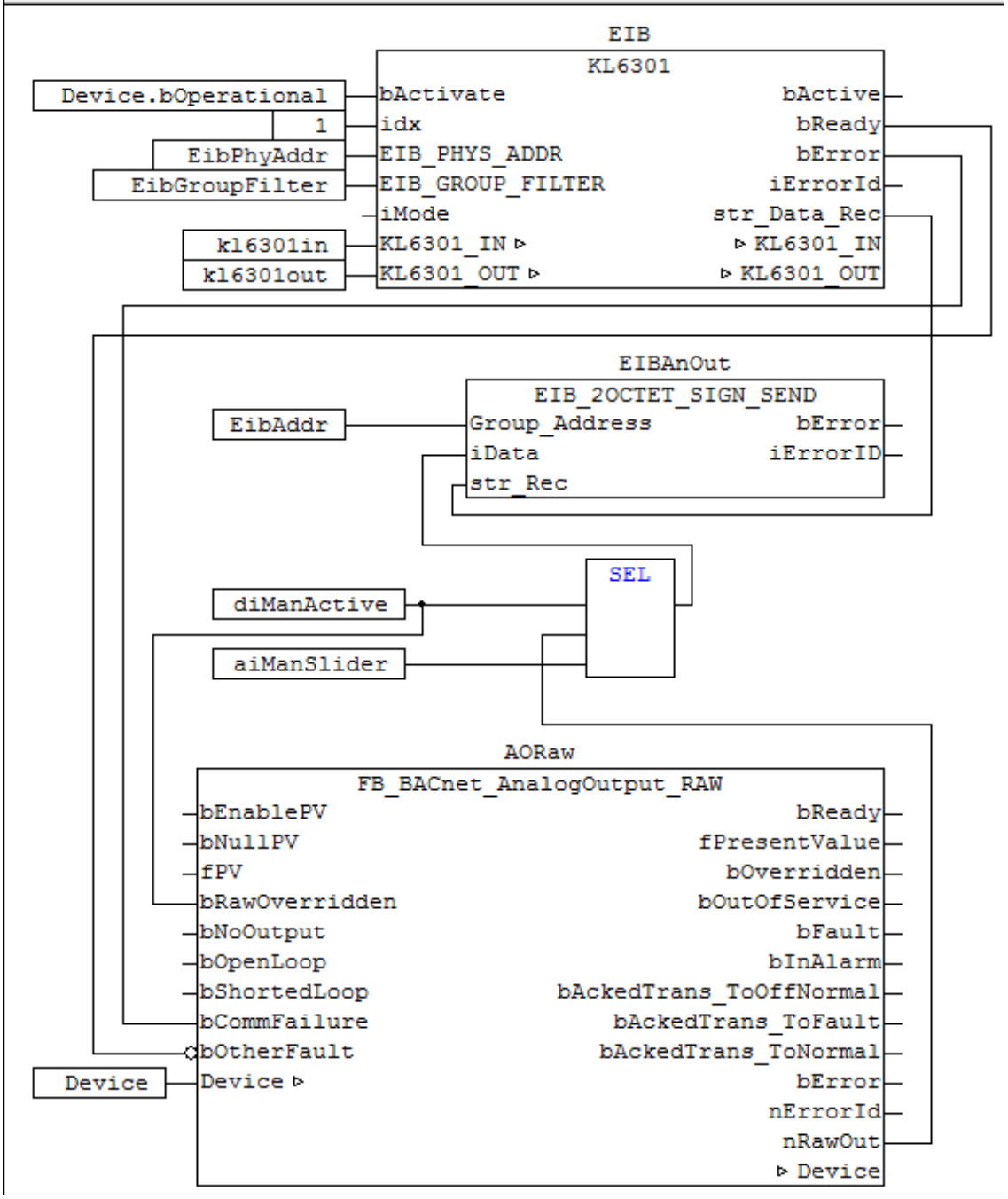
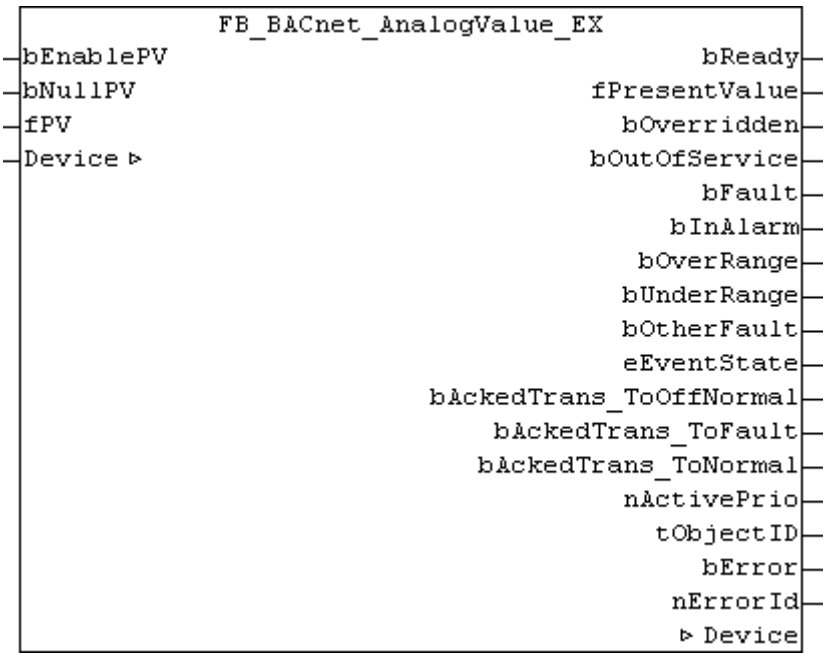


Abb. 11: Bild-1: Beispiel für die Umsetzung der Property Present_Value nach EIB im PLC Programm.

Dieses Beispiel setzt die Bibliothek "TcEIB.lib" voraus. Siehe Dokumentation zur Klemme KL6301 für weitere Informationen zu EIB.

Gezeigt wird zudem die Umsetzung einer Vorortbedienung mit Hilfe eines Vorwahlschalters (digitaler Eingang) und eines Steuerpotis (analoger Eingang).

4.2.10 FB_BACnet_AnalogValue_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_AnalogOutput kann lesend und mit Priorität 12 auf die Property *Present_Value* schreibend auf ein BACnet-Objekt vom Typ AnalogValue (AV) zugegriffen werden.

VAR_INPUT

```
bEnablePV         : BOOL;
bNullPV           : BOOL;
fPV               : REAL;
```

bEnablePV: *TRUE* = Schreibfreigabe des Property-Werts; *FALSE* = Eintrag im BACnet Objekt nicht verändern (**bNullPV** und **fPV** werden unwirksam).

bNullPV: *TRUE* = Eintrag der Property löschen (*NULL*); anstelle des Werts von **fPV**; *FALSE* = Wert von **fPV** als Property Wert schreiben.

fPV: Wert der in die Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** = *TRUE* und **bNullPV** = *FALSE* sind. Das Schreiben der Prozessdaten erfolgt bei Änderung.

VAR_OUPUT

```
bReady                  : BOOL;
fPresentValue           : REAL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bOverRange              : BOOL;
bUnderRange             : BOOL;
bOtherFault             : BOOL;
eEventState             : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
nActivePrio             : UINT;
```

```
tObjectID               : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Status_Flags*.

bOverRange, bUnderRange, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property Event_State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Acked_Transitions*).

nActivePrio: Gibt die aktuell wirksame Priorität des Priority-Arrays an, die auf das *Present_Value* wirkt (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

tObjectID: Objekt ID des BACnet Objekts (Objekt Type und Objekt Instanz).

bError: Ein Fehler steht an.

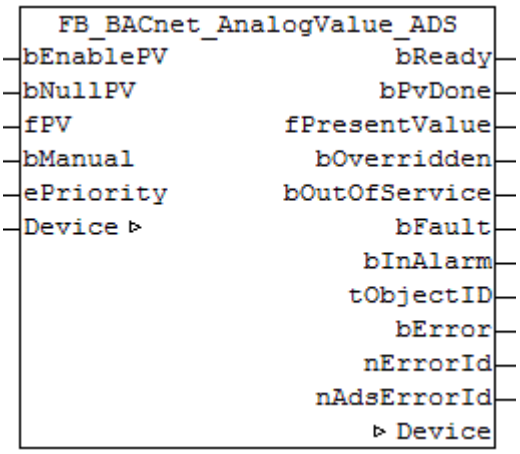
nErrorId: siehe globale Konstanten (BACnet_Globals [▶ 330]).

VAR_IN_OUT

```
Device                  : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.11 FB_BACnet_AnalogValue_ADS



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_AnalogValue_ADS kann lesend und schreibend auf ein BACnet-Objekt vom Typ *AnalogValue* zugegriffen werden.

Das Schrieben der Property *Present_Value* wird, im Unterschied zur Standard- und *_EX*-Variante, mittels ADS-WRITE realisiert (azyklisch).

VAR_INPUT

```
bEnablePV  : BOOL;
bNullPV    : BOOL;
fPV        : REAL;
bManual    : BOOL;
ePriority  : E_BACNETPRIORITY:=BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs **fPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus **fPV** auf die kommandierbare Property *Present_Value* mit Priorität aus **ePriority** (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Wird der Eingang auf *FALSE* gesetzt, erfolgt kein Schreibzugriff (keine Änderung) mehr.

bNullPV: Überschreibt den Eintrag der kommandierbaren Property *Present_Values* in der Priorität **ePriority** mit dem Wert *NULL*, wenn **bEnablePV** auf *TRUE* gesetzt ist. Der Eingang **fPV** wird ignoriert, so lange **bNullPV** auf *TRUE* gesetzt ist.

fPV: Wert der an die entsprechenden Stelle der Property *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** auf *TRUE* und **bNullPV** auf *FALSE* gesetzt ist. Die Priorität wird mit dem Eingang **ePriority** bestimmt (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Das Schreiben wird bei Wertänderung azyklisch ausgeführt (ADS-WRITE).

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung geschieht, ist es potenziell möglich, dass die Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:
- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE* → *TRUE*: Schreiben der Property *Present_Value* bei Flankenwechsel.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 199]).

ePriority: Vorgabe der Priorität für Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 346]).

VAR_OUPUT

```
bReady          : BOOL;
bPvDone         : BOOL;
fPresentValue   : REAL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
nAdsErrorId     : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogValue* und Property *Status_Flags*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

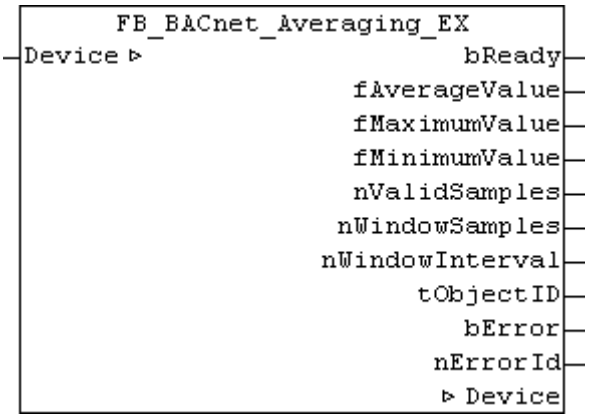
nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.12 FB_BACnet_Averaging_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Averaging kann lesend auf ein BACnet-Objekt vom Typ *Averaging* (AVG) zugegriffen werden.

VAR_OUPUT

Variablen sind nicht in der Basisversion des Bausteins enthalten.

```
bReady           : BOOL;
fAverageValue    : REAL;
fMaximumValue    : REAL;
fMinimumValue    : REAL;
nValidSamples    : UDINT;
nWindowSamples   : UDINT;
nWindowInterval  : UDINT;
tObjectID        : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;  (*siehe Info*)
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

fAverageValue: Aktueller Mittelwert (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Averaging* und Property *Average_Value*).

fMaximumValue: Größter erfasster Wert (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Averaging* und Property *Maximum_Value*).

fMinimumValue: Kleinster erfasster Wert (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Averaging* und Property *Minimum_Value*).

nValidSamples: Anzahl gültiger Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Averaging* und Property *Valid_Samples*).

nWindowSamples: Anzahl Werte die über die Zeitspanne von *Window_Intervall* erfasst werden (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Averaging* und Property *Window_Samples*).

nWindowInterval: Zeitspanne über die Werte gemittelt werden in vollen Sekunden (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Averaging* und Property *Window_Interval*).

tObjectID: Objekt ID des BACnet Objekts (Objekt Type und Objekt Instanz).

bError: Ein Fehler steht an.

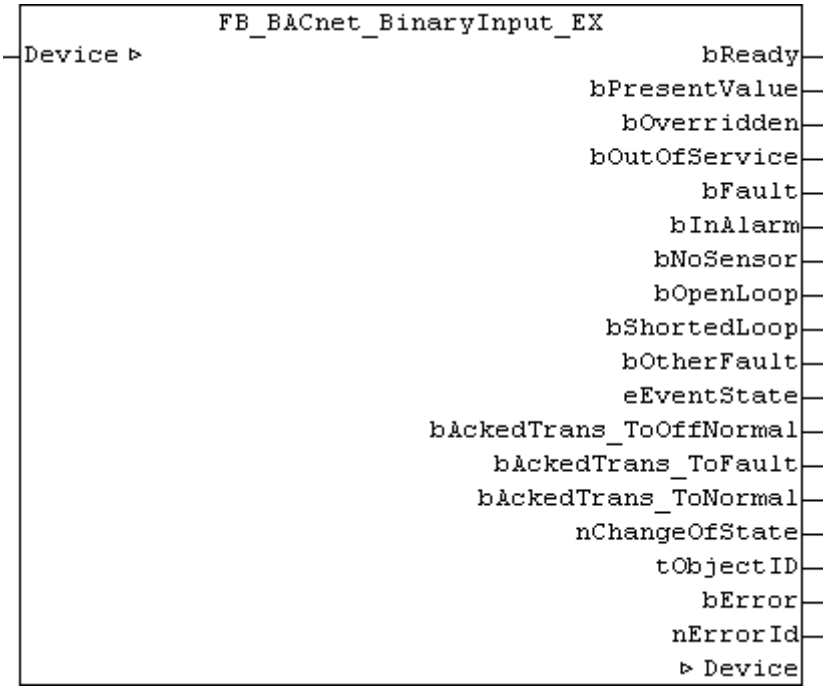
nErrorId: siehe globale Konstanten (BACnet Globals [▶ 330]).

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.13 FB_BACnet_BinaryInput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_AnalogInput kann lesend auf ein BACnet-Objekt vom Typ *BinaryInput* (BI) zugegriffen werden.

VAR_OUPUT

```
bReady                  : BOOL;
bPresentValue           : BOOL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bNoSensor               : BOOL;
bOpenLoop               : BOOL;
bShortedLoop            : BOOL;
bOtherFault             : BOOL;
eEventState             : E_BACNETEVENTSTATE;
```

```
bAckedTrans_ToOffNormal  : BOOL;
bAckedTrans_ToFault      : BOOL;
bAckedTrans_ToNormal     : BOOL;
nChangeOfState           : UDINT;
tObjectID                : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                   : BOOL;
nErrorId                 : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Status_Flags*.

bNoSensor, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property Event_State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Acked_Transitions*).

nChangeOfState: Anzahl der Zustandsänderungen (*nicht* gleichzusetzen mit Wertänderung) der Property *Present_Value* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Change_Of_State_Count*).

tObjectID: Objekt ID des BACnet Objekts (Objekt Type und Objekt Instanz).

bError: Ein Fehler steht an.

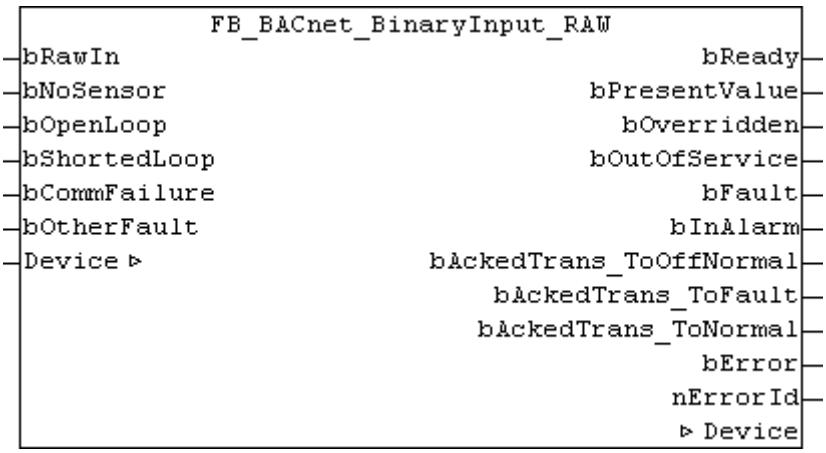
nErrorId: siehe globale Konstanten (BACnet_Globals [▶ 330]).

VAR_IN_OUT

```
Device               : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.14 FB_BACnet_BinaryInput_RAW



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_BinaryInput_RAW kann lesend und schreibend auf ein BACnet-Objekt vom Typ *BinaryInput* zugegriffen werden.

Im Unterschied zur Standard- bzw. *_EX*-Variante des Bausteins, wird der Rohwert und der Zustand der Property *Reliablitity* durch Funktionsbaustein-Eingänge bereitgestellt und nicht durch die IO-Hardware direkt. So kann z.B. der Zustand eines Binärwerts aus einem Sub-Bussystems in SPS-Code auf ein BACnet-Objekt abgebildet werden (Signalumsetzung von Sub-Bussystemen oder virtuellen Datenpunkten nach BACnet, siehe Beispiel [▶ 187]).

VAR_INPUT

```
bRawIn               : BOOL;
bNoSensor            : BOOL;
bOpenLoop            : BOOL;
bShortedLoop         : BOOL;
bCommFailure         : BOOL;
bOtherFault          : BOOL;
```

bRawIn: Rohwerteingang des Objekts. Der Eingang wird mit dem Prozessdatum "RawIoBinaryBoolValue" des BACnet-Objekts verknüpft. Der Wert aus **bRawIn** wird mit der Property *Polarity* zum dem Wert der Property *Present_Value* verrechnet (vorausgesetzt der Objektzustand ist nicht *out_of_service*).

bNoSensor, bOpenLoop, bShortedLoop, bCommFailure, bOtherFault: *TRUE* am Eingang setzt den entsprechenden Zustand [▶ 352] der Property *Reliability*. Die Priorität fällt mit der Reihenfolge der Eingänge (**bNoSensor** höchste und **bOtherFault** niedrigste Priorität). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Reliability*.

VAR_OUPUT

```
bReady                   : BOOL;
bPresentValue            : BOOL;
bOverridden              : BOOL;
bOutOfService            : BOOL;
bFault                   : BOOL;
bInAlarm                 : BOOL;
bAckedTrans_ToOffNormal  : BOOL;
bAckedTrans_ToFault      : BOOL;
bAckedTrans_ToNormal     : BOOL;
bError                   : BOOL;
nErrorId                 : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Status_Flags*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Acked_Transitions*).

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device               : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung eines Bit-Werts aus EIB in die Property *Present_Value* eines BinaryInput-Objekts gezeigt:

```
0002 VAR
0003 (* EIB *)
0004     EIB          : KL6301;
0005     EIBBinIn     : EIB_BIT_REC;
0006
0007 (* BACnet *)
0008     Device       : FB_BACnet_Device;
0009     BIRaw        : FB_BACnet_BinaryInput_RAW;
0010 END_VAR
```

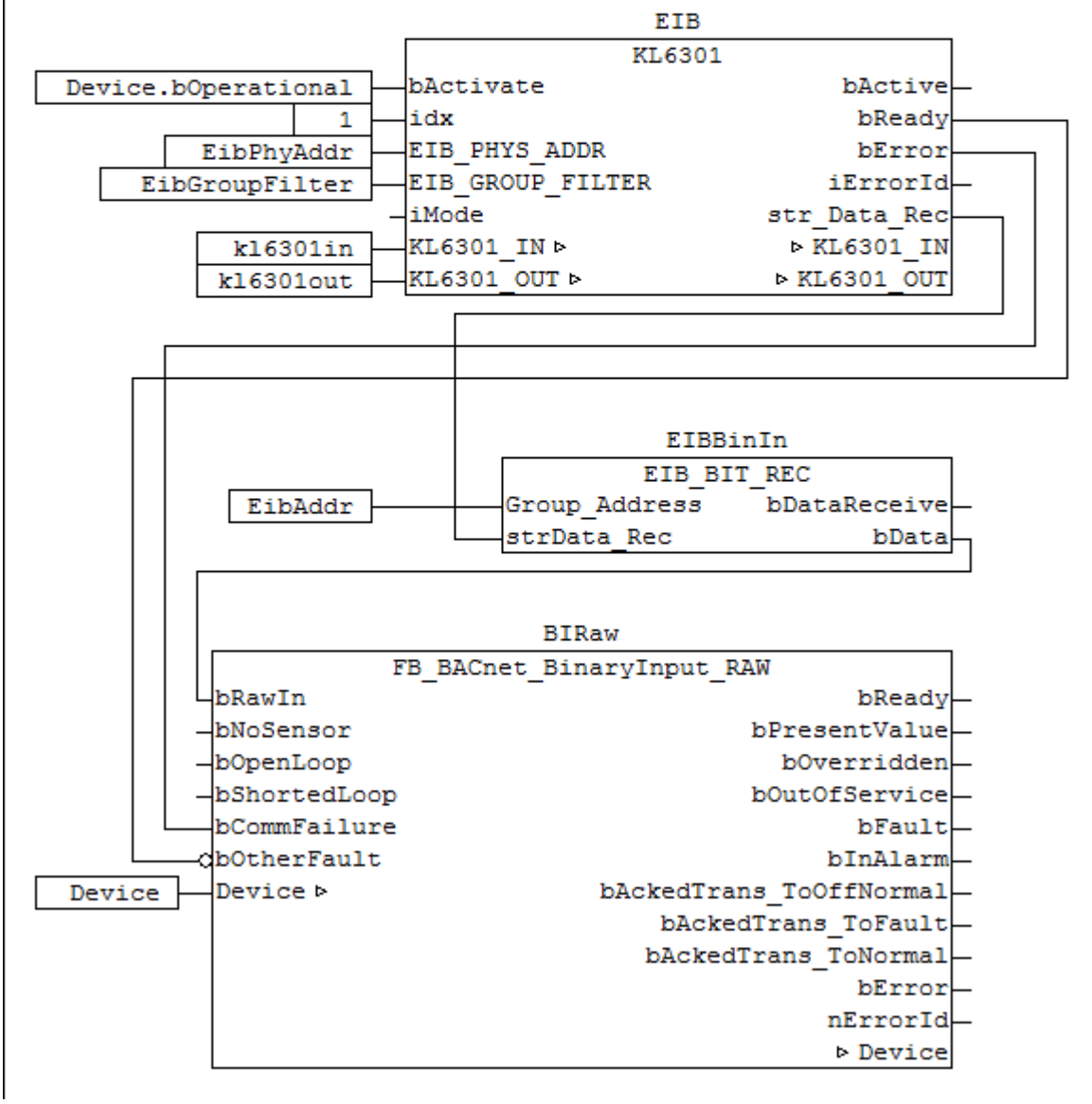
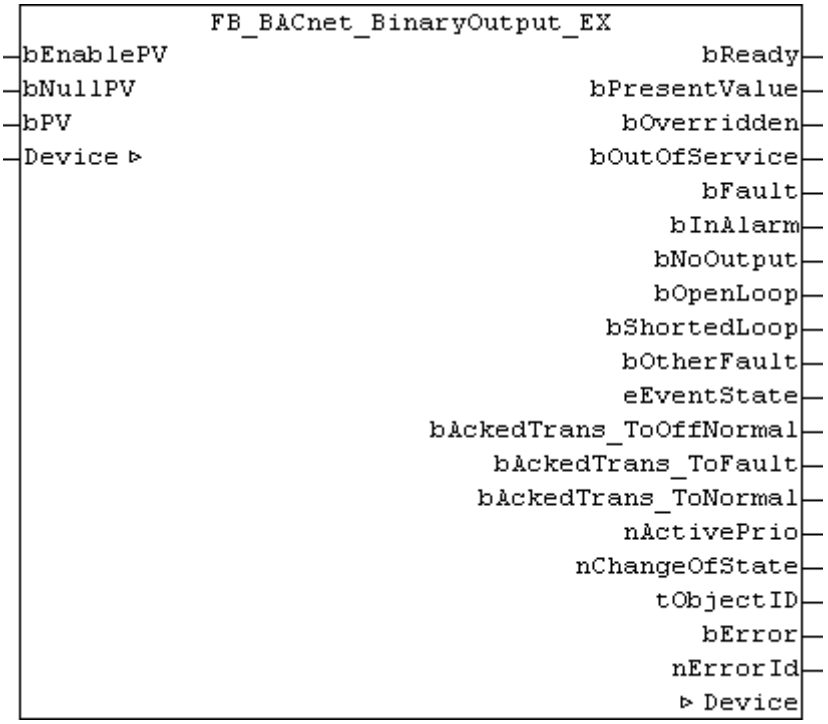


Abb. 12: Bild-1: Beispiel für die Umsetzung eines Bit-Werts aus EIB in die Property Present_Value im PLC Programm.

Dieses Beispiel setzt die Bibliothek "TcEIB.lib" voraus. Siehe Dokumentation zur Klemme KL6301 für weitere Informationen zu EIB.

4.2.15 FB_BACnet_BinaryOutput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_BinaryOutput kann lesend und mit Priorität 12 auf die Property *Present_Value* schreibend auf ein BACnet-Objekt vom Typ *BinaryOutput* (BO) zugegriffen werden.

VAR_INPUT

```
bEnablePV   : BOOL;
bNullPV     : BOOL;
bPV         : BOOL;
```

bEnablePV: *TRUE* = Schreibfreigabe des Property-Werts; *FALSE* = Eintrag im BACnet Objekt nicht verändern (**bNullPV** und **fPV** werden unwirksam).

bNullPV: *TRUE* = Eintrag der Property löschen (*NULL*); anstelle des Werts von **bPV**; *FALSE* = Wert von **bPV** als Property Wert schreiben.

b PV: Wert der in die Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** = *TRUE* und **bNullPV** = *FALSE* sind. Das Schreiben der Prozessdaten erfolgt bei Änderung.

VAR_OUPUT

```
bReady                  : BOOL;
bPresentValue           : BOOL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bNoOutput               : BOOL;
bOpenLoop               : BOOL;
bShortedLoop            : BOOL;
bOtherFault             : BOOL;
eEventState             : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
nActivePrio             : UINT;
nChangeOfState          : UDINT;
tObjectID               : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Status_Flags*.

bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property Event State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Acked_Transitions*).

nActivePrio: Gibt die aktuell wirksame Priorität des Priority-Array an, die auf das *Present_Value* wirkt (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

nChangeOfState: Anzahl der Zustandsänderungen (*nicht* gleichzusetzen mit Wertänderung) der Property *Present_Value* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Change_Of_State_Count*).

tObjectID: Objekt ID des BACnet Objekts (Objekt Type und Objekt Instanz).

bError: Ein Fehler steht an.

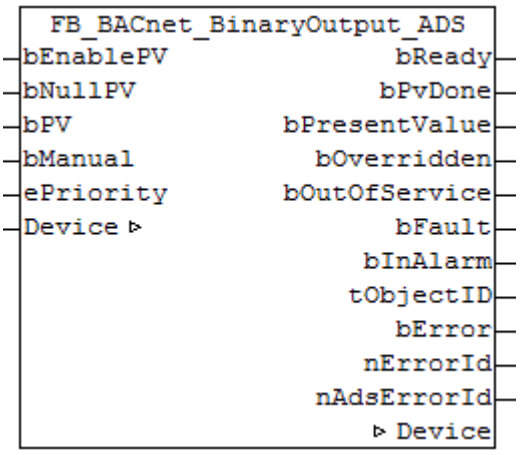
nErrorId: siehe globale Konstanten (BACnet Globals [▶ 330]).

VAR_IN_OUT

```
Device              : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.16 FB_BACnet_BinaryOutput_ADS



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_BinaryOutput_ADS kann lesend und schreibend auf ein BACnet-Objekt vom Typ *BinaryOutput* zugegriffen werden.

Das Schrieben der Property *Present_Value* wird, im Unterschied zur Standard- und *_EX*-Variante, mittels ADS-WRITE realisiert (azyklisch).

VAR_INPUT

```
bEnablePV  : BOOL;
bNullPV    : BOOL;
bPV        : BOOL;
bManual    : BOOL;
ePriority  : E_BACNETPRIORITY:=BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs **bPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus **bPV** auf die kommandierbare Property *Present_Value* mit Priorität aus **ePriority** (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Wird der Eingang auf *FALSE* gesetzt, erfolgt kein Schreibzugriff (keine Änderung) mehr.

bNullPV: Überschreibt den Eintrag der kommandierbaren Property *Present_Values* in der Priorität **ePriority** mit dem Wert *NULL*, wenn **bEnablePV** auf *TRUE* gesetzt ist. Der Eingang **bPV** wird ignoriert, so lange **bNullPV** auf *TRUE* gesetzt ist.

bPV: Wert der an die entsprechenden Stelle der Property *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** auf *TRUE* und **bNullPV** auf *FALSE* gesetzt ist. Die Priorität wird mit dem Eingang **ePriority** bestimmt (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Das Schreiben wird bei Wertänderung azyklisch ausgeführt (ADS-WRITE).

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung geschieht, ist es potenziell möglich, dass die Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:

- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE* → *TRUE*: Schreiben der Property *Present_Value* bei Flankenwechsel.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 199]).

ePriority: Vorgabe der Priorität für Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 346]).

VAR_OUPUT

```
bReady         : BOOL;
bPvDone        : BOOL;
bPresentValue  : BOOL;
bOverridden    : BOOL;
bOutOfService  : BOOL;
bFault         : BOOL;
bInAlarm       : BOOL;
tObjectID      : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError         : BOOL;
nErrorId       : UINT;
nAdsErrorId    : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Status_Flags*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

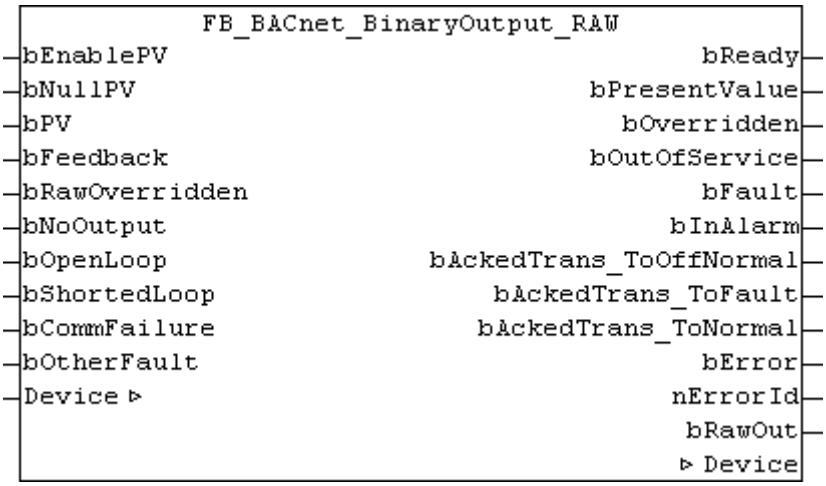
nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.17 FB_BACnet_BinaryOutput_RAW



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_BinaryOutput_RAW kann lesend und schreibend auf ein BACnet-Objekt vom Typ *BinaryOutput* zugegriffen werden.

Im Unterschied zur Standard- bzw. *_EX*-Variante des Bausteins, wird das Flag der Property *Status_Flags* / *overridden*, die Property *Feedback_Value* und der Wert der Property *Reliablitity* durch Funktionsbaustein-Eingänge bereitgestellt und aus dem PLC-Programm auf das BACnet Objekt abgebildet. Die Zustand der Property *Present_Value* wird durch das PLC-Programm auf die Hardware bzw. auf das Sub-Bussystem gemappt. Dies geschieht sonst direkt durch die IO-Hardware. So kann z.B. der Ausgangswert an ein Sub-Bussystems in PLC-Code von einem BACnet-Objekt abgebildet werden (Signalumsetzung auf Sub-Bussysteme oder virtuellen Datenpunkte von BACnet, siehe Beispiel [▶ 192]).

VAR_INPUT

```
bEnablePV       : BOOL;
bNullPV         : BOOL;
bPV             : BOOL;
bFeedback       : BOOL;
bRawOverridden  : BOOL;
bNoOutput       : BOOL;
bOpenLoop       : BOOL;
bShortedLoop    : BOOL;
bCommFailure    : BOOL;
bOtherFault     : BOOL;
```

bEnablePV: *TRUE* = Schreibfreigabe des Property-Werts; *FALSE* = Eintrag im BACnet Objekt nicht verändern (**bNullPV** und **fPV** werden unwirksam).

bNullPV: *TRUE* = Eintrag der Property löschen (*NULL*); anstelle des Werts von **bPV**; *FALSE* = Wert von **bPV** als Property Wert schreiben.

b PV: Wert der in die Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** = *TRUE* und **bNullPV** = *FALSE* sind. Das Schreiben der Prozessdaten erfolgt bei Änderung.

bFeedback: Signalrückmeldung an das BACnet Objekt. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Feedback_Value*.

bRawOverridden: *TRUE* setzt das Flag *overridden* der Property *Status_Flags*. Damit wird nach BACnet signalisiert, dass der Wert der Property *Present_Value* von der Hardware-Ausgabe entkoppelt wurde (z.B. durch eine manuelle Vorortbedienung). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Status_Flags*.

bNoOutput, bOpenLoop, bShortedLoop, bCommFailure, bOtherFault: *TRUE* am Eingang setzt den entsprechenden Zustand [▶ 352] der Property *Reliability*. Die Priorität fällt mit der Reihenfolge der Eingänge (**bNoOutput** höchste und **bOtherFault** niedrigste Priorität). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Reliability*.

VAR_OUPUT

```
bReady                  : BOOL;
bPresentValue           : BOOL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
bRawOut                 : BOOL;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Status_Flags*.

bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Reliability*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Acked_Transitions*).

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet Globals [▶ 330].

bRawOut: Rohwertausgang des Objekts. Der Ausgang wird mit dem Prozessdatum "RawIoBinaryBoolValue" des BACnet-Objekts verknüpft. Der Wert von **bRawOut** resultiert aus der Property *Present_Value* mit Verrechnung der Property *Polarity*.

VAR_IN_OUT

```
Device                : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung der Property *Present_Value* eines BinaryOutput-Objekts nach EIB gezeigt:

```
0002 VAR
0003 (* EIB *)
0004     EIB          : KL6301;
0005     EIBBinOut    : EIB_BIT_SEND;
0006
0007 (* Manual override *)
0008     diManActive AT%I*:BOOL; (* key switch on cabinet *)
0009     diManOnOff  AT%I*:BOOL; (* on/off switch on cabinet *)
0010
0011 (* BACnet *)
0012     Device       : FB_BACnet_Device;
0013     BORaw        : FB_BACnet_BinaryOutput_RAW;
0014 END_VAR
```

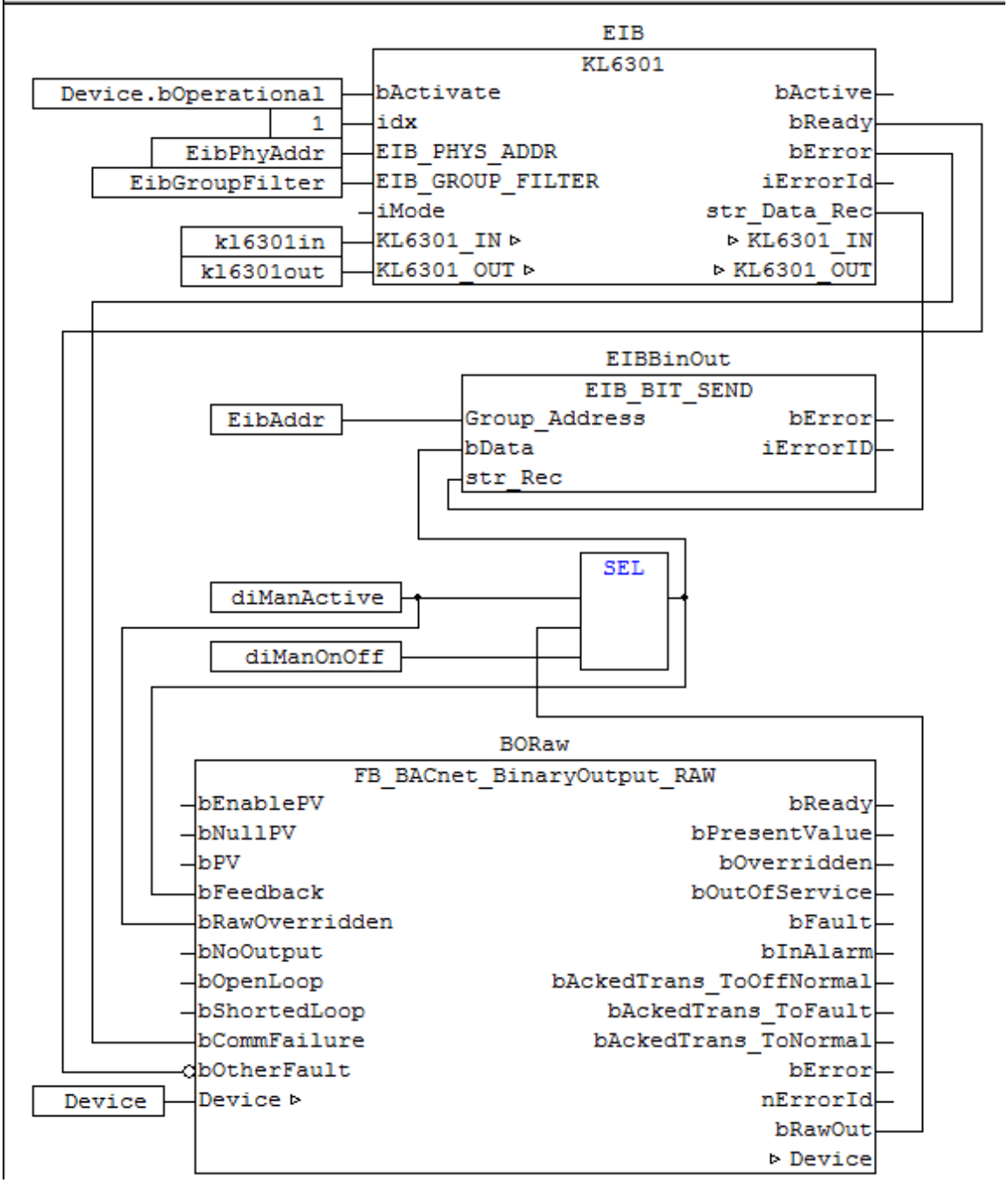
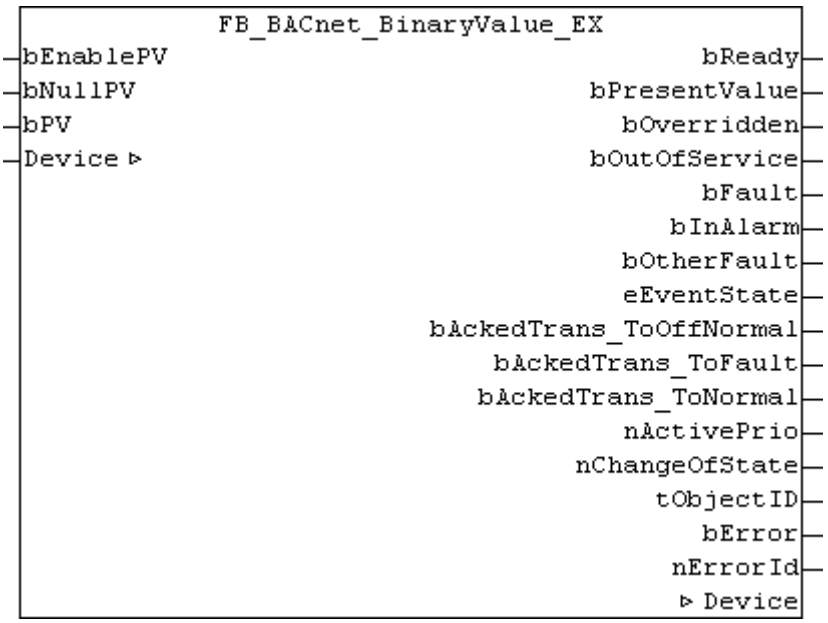


Abb. 13: Bild-1: Beispiel für die Umsetzung der Property Present_Value nach EIB im PLC Programm.

Dieses Beispiel setzt die Bibliothek "TcEIB.lib" voraus. Siehe Dokumentation zur Klemme KL6301 für weitere Informationen zu EIB.

Gezeigt wird zudem die Umsetzung einer Vorortbedienung mit Hilfe eines Vorwahlschalters (digitaler Eingang) und eines Steuerschalters (digitaler Eingang).

4.2.18 FB_BACnet_BinaryValue_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_BinaryValue kann lesend und mit Priorität 12 auf die Property *Present_Value* schreibend auf ein BACnet-Objekt vom Typ *BinaryValue* (BV) zugegriffen werden.

VAR_INPUT

```
bEnablePV       : BOOL;
bNullPV         : BOOL;
bPV             : BOOL;
```

bEnablePV: *TRUE* = Schreibfreigabe des Property-Werts; *FALSE* = Eintrag im BACnet Objekt nicht verändern (**bNullPV** und **fPV** werden unwirksam).

bNullPV: *TRUE* = Eintrag der Property löschen (*NULL*); anstelle des Werts von **bPV**; *FALSE* = Wert von **bPV** als Property Wert schreiben.

b PV: Wert der in die Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** = *TRUE* und **bNullPV** = *FALSE* sind. Das Schreiben der Prozessdaten erfolgt bei Änderung.

VAR_OUPUT

```
bReady                  : BOOL;
bPresentValue           : BOOL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bOtherFault             : BOOL;
eEventState             : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
nActivePrio             : UINT;
nChangeOfState          : UDINT;
tObjectID               : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Status_Flags*.

bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property Event State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Acked_Transitions*).

nActivePrio: Gibt die aktuell wirksame Priorität des Priority-Array an, die auf das *Present_Value* wirkt (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Present_Value*).

nChangeOfState: Anzahl der Zustandsänderungen (*nicht* gleichzusetzen mit Wertänderung) der Property *Present_Value* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Change_Of_State_Count*).

tObjectID: Objekt ID des BACnet Objekts (Objekt Typ und Objekt Instanz).

bError: Ein Fehler steht an.

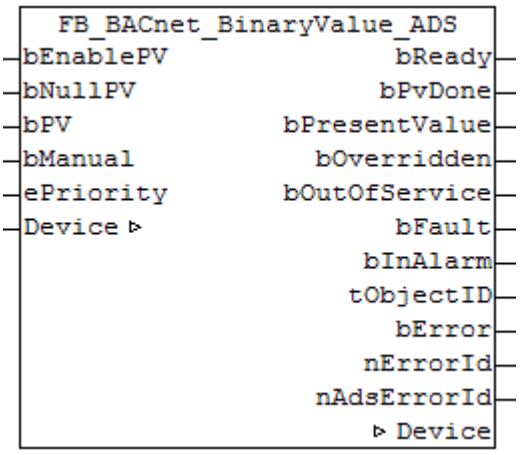
nErrorId: siehe globale Konstanten (BACnet Globals [▶ 330]).

VAR_IN_OUT

```
Device                : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.19 FB_BACnet_BinaryValue_ADS



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_BinaryValue_ADS kann lesend und schreibend auf ein BACnet-Objekt vom Typ *BinaryValue* zugegriffen werden.

Das Schrieben der Property *Present_Value* wird, im Unterschied zur Standard- und *_EX*-Variante, mittels ADS-WRITE realisiert (azyklisch).

VAR_INPUT

```
bEnablePV       : BOOL;
bNullPV         : BOOL;
bPV             : BOOL;
bManual         : BOOL;
ePriority       : E_BACNETPRIORITY:=BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs **bPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus **bPV** auf die kommandierbare Property *Present_Value* mit Priorität aus **ePriority** (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Wird der Eingang auf *FALSE* gesetzt, erfolgt kein Schreibzugriff (keine Änderung) mehr.

bNullPV: Überschreibt den Eintrag der kommandierbaren Property *Present_Values* in der Priorität **ePriority** mit dem Wert *NULL*, wenn **bEnablePV** auf *TRUE* gesetzt ist. Der Eingang **bPV** wird ignoriert, solange **bNullPV** auf *TRUE* gesetzt ist.

bPV: Wert der an die entsprechende Stelle der Property *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** auf *TRUE* und **bNullPV** auf *FALSE* gesetzt ist. Die Priorität wird mit dem Eingang **ePriority** bestimmt (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Das Schreiben wird bei Wertänderung azyklisch ausgeführt (ADS-WRITE).

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung geschieht, ist es potenziell möglich, dass die Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:
- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE* → *TRUE*: Schreiben der Property *Present_Value* bei Flankenwechsel.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 199]).

ePriority: Vorgabe der Priorität für Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 346]).

VAR_OUPUT

```
bReady          : BOOL;
bPvDone         : BOOL;
bPresentValue   : BOOL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
nAdsErrorId     : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Status_Flags*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

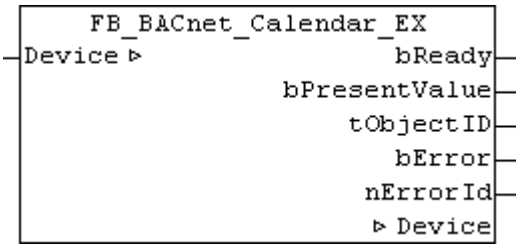
nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.20 FB_BACnet_Calendar_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Calendar kann lesend auf ein BACnet-Objekt vom Typ *Calendar* (CAL) zugegriffen werden.

VAR_OUPUT

```
bReady          : BOOL;
bPresentValue   : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Calendar* und Property *Present_Value*).

tObjectID: Objekt ID des BACnet Objekts (Objekt Type und Objekt Instanz).

bError: Ein Fehler steht an.

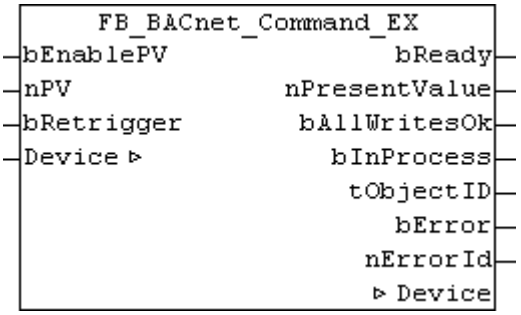
nErrorId: siehe globale Konstanten (BACnet_Globals [▶ 330]).

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.21 FB_BACnet_Command_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Command kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Command* (CMD) zugegriffen werden.

VAR_INPUT

```
bEnablePV       : BOOL;
nPV             : UDINT;
bRetrigger      : BOOL;
```

bEnablePV: Gibt den Wert des Eingangs **nPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt das Schreiben in die Property *Present_Value* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs **nPV**.
Wird **bEnablePV** auf *FALSE* gesetzt, dann werden die Prozessdaten der gemappten Property *Present_Value* auf *0* geschrieben und damit deaktiviert.

nPV: Wert der Property *Present_Value* der geschrieben werden soll. Liegt der Wert außerhalb des Wertebereicht (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *Present_Value*), dann wird das zugehörige Prozessdatum deaktiviert, d.h. das Schreiben auf die Property ist deaktiviert.
Das Schreiben eines gültigen Wertes löst die Ausführung der zugehörigen Kommando-Liste des BACnet-Objekts aus. Geschrieben wird der Wert der Property *Present_Value* immer dann, wenn eine Änderung des Prozessdatums erfolgt (d.h. Änderung des Wertes von **nPV** bei gesetztem **bEnablePV** oder Signalswechsel *FALSE* --> *TRUE* am Eingang **bRetrigger** bei gesetztem **bEnablePV**).

bRetrigger: Bei steigender Flanke an diesem Eingang wird das Schreiben und damit die Ausführung des entsprechenden Kommandos des BACnet-Objekts *Command* wiederholt. Der Signalwechsel von *FALSE* --> *TRUE* entspricht einer Änderung des Prozessdatums der Property *Present_Value* von x → *0* → x.

VAR_OUPUT

```
bReady          : BOOL;
nPresentValue   : UDINT;
bAllWritesOk    : BOOL;
bInProcess      : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *Present_Value*).

bAllWritesOk: Die zuletzt angeforderte Kommando-Liste wurde erfolgreich abgearbeitet (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *All_Writes_Successful*).

bInProcess: Die selektierte Kommando-Liste wird abgearbeitet (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *In_Process*).

tObjectID: Objekt ID des BACnet Objekts (Objekt Type und Objekt Instanz).

bError: Ein Fehler steht an.

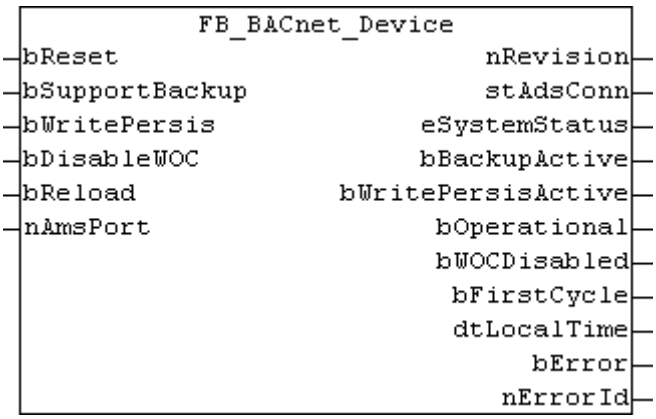
nErrorId: siehe globale Konstanten (BACnet_Globals [▶ 330]).

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.22 FB_BACnet_Device



Anwendung

Funktionsbaustein für die Anbindung des PLC Programmes an ein lokales BACnet Device Objekt (Server). Die Verknüpfung des Funktionsbausteins erfolgt mit Hilfe von Prozessdaten und ADS (AMS Port evaluieren, Objekt Liste lesen, ermitteln der Stack Revision und Backup).

Grundsätzlich ist der Funktionsbaustein abwärtskompatibel (Revision 6). Die verwendete BACnet Revision kann am Ausgang **nRevision** abgelesen werden.
Bei Verwendung von Revision 6 gibt es jedoch Einschränkungen bei einigen ADS Diensten des BACnet Stacks; z.B. Der AMS Port des BACnet Device (Server und Client) kann erst ab Revision 12 automatisch ermittelt werden. Der im TwinCAT System Manager ersichtliche AMS Port (Ads Port) muss daher, bei Verwendung von Revision 6, am Eingang **nAmsPort** übergeben werden (siehe Bild-1)!

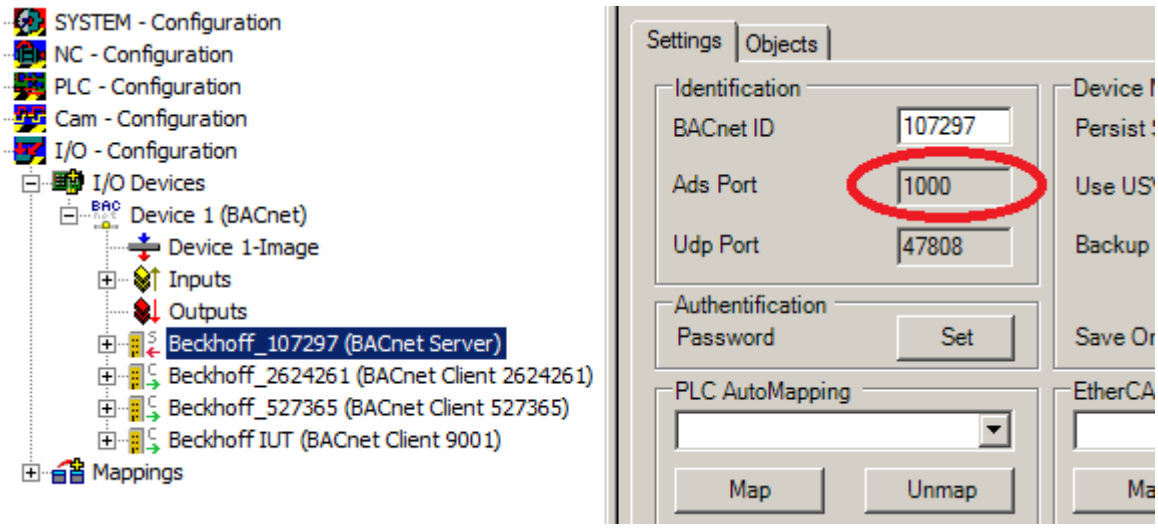


Abb. 14: Bild-1: AMS Port des BACnet Device im TwinCAT System Manager

VAR_INPUT

Eingänge sind optional und müssen bei Verwendung von Revision 12 oder höher nicht belegt werden.

```
bReset          : BOOL;
bSupportBackup  : BOOL;
bWritePersis    : BOOL;
bDisableWOC     : BOOL;
bReload         : BOOL;
nAmsPort        : T_AmsPort:=1000;  (*siehe Info*)
```

bReset: Zurücksetzen des Fehlerzustands bei Signalwechsel *FALSE* → *TRUE*.

bSupportBackup: Bei Setzen des Eingangs auf *TRUE*, wird das Sichern der persistenten Daten der PLC Laufzeit über BACnet unterstützt. Ist dieser Modus aktiviert, müssen die persistenten Dateien der SPS (standardmäßig unter: "C:\TwinCAT\Boot\") als BACnet-File Objekte angelegt und mit dem Wert "TwinCAT Configuration File" der Property *File_Type* versehen werden (damit sorgt der BACnet Stack für die Sicherung und Übertragung der Dateien an das backupende Remotesystem).
Bei Verwendung des BACnet-seitigen Backups der persistenten PLC Daten, muss keine weitere Instanz des Bausteins *FB_WritePersistentData* in der PLC Laufzeit verwendet werden (*FB_WritePersistentData* wird von FB_BACnet_Device ausgeführt) - Der Eingang **bWritePersis** löst das Schreiben der persistenten Daten unabhängig vom BACnet Backup aus.

bWritePersis: Signalswechsel *FALSE* → *TRUE* löst das Schreiben der persistenten PLC Daten unabhängig von BACnet Backup aus. Baustein-intern wird der Aufruf von *FB_WritePersistentData* ausgelöst.

bDisableWOC: Zustand TRUE am Eingang deaktiviert das Schreiben von Prozessdaten an die BACnet Objekte des entsprechenden BACnet Device. Damit werden sämtliche Property-Schreibzugriffe gesperrt.

bReload: Die ADS Verbindung wird erneuert. Nachfolgenden Bausteine, die die ADS-Verbindung nutzen, werden ebenfalls getriggert.

nAmsPort: *Nur bei Verwendung von Revision 6, sonst nicht verwenden:* AMS Port des lokalen BACnet Server. Im Standardfall ist dieser 1000 (vorausgesetzt der BACnet-Server ist das erste Element unterhalt des BACnet-Adapters im TwinCAT System Manager). Die Portnummer kann im System Manager abgelesen werden (siehe Bild-1).

VAR_OUPUT

```
nRevision           : DINT:=-1;
stAdsConn           : ST_BACnet_AdsConnection;
eSystemStatus       : E_BACNETDEVICESTATUS;
bBackupActive       : BOOL;
bWritePersisActive  : BOOL;
bOperational        : BOOL;
bWOCDisabled        : BOOL;
bFirstCycle         : BOOL;
dtLocalTime         : ST_BACnet_DateTime;
bError              : BOOL;
nErrorId            : UINT;
```

nRevision: Ausgabe der verwendeten BACnet Revisionsnummer.

stAdsConn: Struktur mit den ADS Verbindungsdaten (ab Revision 12 wird der enthaltene AMS Port automatisch ermittelt).

eSystemStatus: Status des BACnet Server Objekts (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet Device Objekt und Property *System_Status).*

bBackupActive: Rückmeldung dass ein Backup via BACnet ausgeführt wird, wenn der Ausgang auf *TRUE* gesettzt ist.

bWritePersisActive: Das Schreiben der persistenten PLC Daten wird ausgeführt. Das Schreiben der persistenten PLC Daten kann durch BACnet Backup oder durch Setzen des Eingangs **bWritePersis** ausgelöst werden.

bOperational: Device-Adapter befindet sich nicht mehr im Status *Init*. Fällt der Ausgang auf *FALSE* werden sämtliche verbundene BACnet Objekt-Funktionsbausteine im PLC Programm gesperrt.

bWOCDisabled: Rückmeldung zum Eingang **bDisableWOC**.

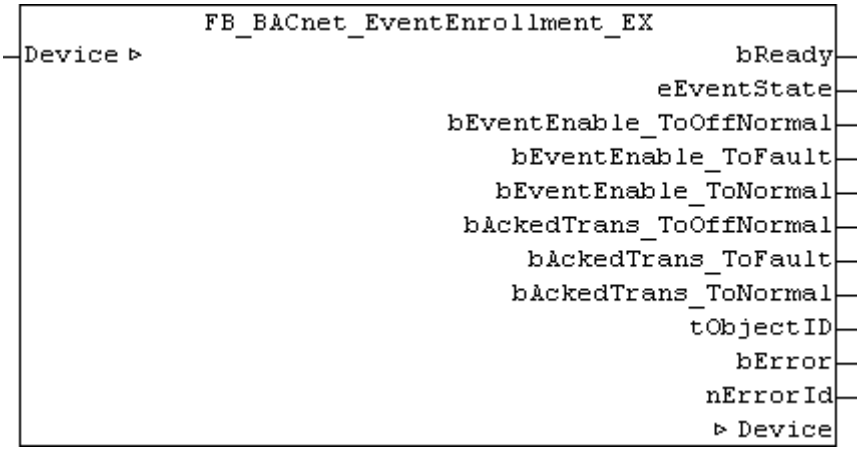
bFirstCycle: Ausgang wird beim erstmaligen Aufruf der Funktionsbausteininstanz auf *TRUE* gesetzt (PLC Start). Anschließend bleibt der Ausgang auf *FALSE*.

dtLocalTime: Aktuelle BACnet Zeit (siehe Property *Local_Date* und *Local_Time* des BACnet Device Objekts). Für die Synchronisierung der Uhrzeit zwischen Betriebssystem, BACnet Stack und BACnet Netzwerk steht der Funktionsbaustein FB_BACnet_TimeSync [▶ 272] zur Verfügung.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten (BACnet_Globals [▶ 330]).

4.2.23 FB_BACnet_EventEnrollment_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_EventEnrollment kann lesend auf ein BACnet-Objekt vom Typ *EventEnrollment* (EE) zugegriffen werden.

VAR_OUPUT

```
bReady                  : BOOL;
eEventState             : E_BACNETEVENTSTATE;
bEventEnable_ToOffNormal  : BOOL;  (*siehe Anmerkung*)
bEventEnable_ToFault      : BOOL;  (*siehe Anmerkung *)
bEventEnable_ToNormal     : BOOL;  (*siehe Anmerkung *)
bAckedTrans_ToOffNormal   : BOOL;
bAckedTrans_ToFault       : BOOL;
bAckedTrans_ToNormal      : BOOL;
tObjectID               : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Event_State*.

bEventEnable_ToOffNormal, bEventEnable_ToFault, bEventEnable_ToNormal: Flags [▶ 361] der Property *Event_Enable* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Event_Enable*).

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags [▶ 361] der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Acked_Transitions*).

tObjectID: Objekt ID des BACnet Objekts (Objekt Typ und Objekt Instanz).

bError: Ein Fehler steht an.

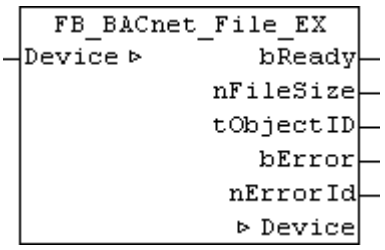
nErrorId: siehe globale Konstanten (BACnet_Globals [▶ 330]).

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.24 FB_BACnet_File_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_File kann lesend auf ein BACnet-Objekt vom Typ *File* (FILE) zugegriffen werden.

VAR_OUPUT

```
bReady      : BOOL;
nFileSize   : UDINT;
tObjectID   : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError      : BOOL;
nErrorId    : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (FileSize und ObjectID). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nFileSize: Aktuelle Dateigröße in Byte (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *File* und Property *File_Size*).

tObjectID: Objekt ID des BACnet Objekts (Objekt Typ und Objekt Instanz).

bError: Ein Fehler steht an.

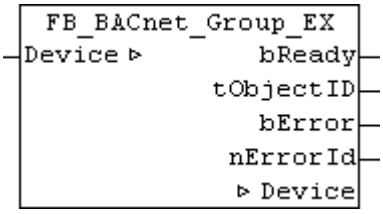
nErrorId: siehe globale Konstanten (BACnet_Globals [▶ 330]).

VAR_IN_OUT

```
Device      : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.25 FB_BACnet_Group_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Group_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Group* zugegriffen werden.

VAR_OUPUT

```
bReady     : BOOL;
tObjectID  : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError     : BOOL;
nErrorId   : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

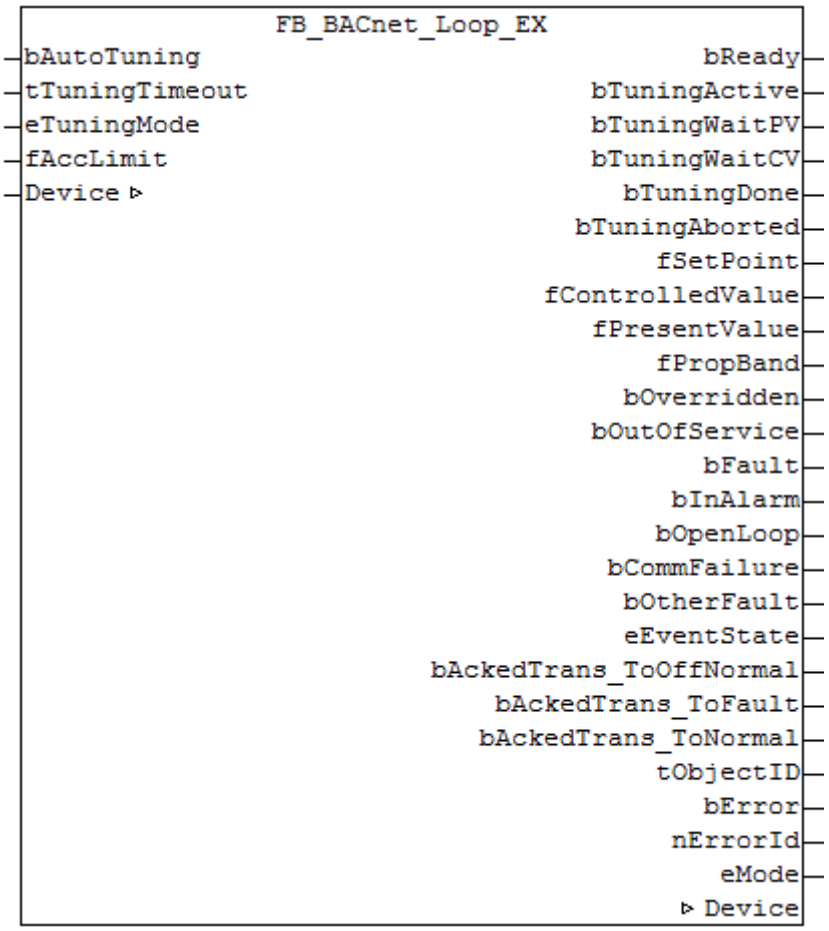
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device     : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.26 FB_BACnet_Loop_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Loop_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Loop* zugegriffen werden. Der Funktionsbaustein enthält einen Standard PID-Regler (FB_BACnet_PidControl [▶ 305]) und deckt damit ein breites Spektrum an Regelaufgaben ab.

VAR_INPUT

```
bAutoTuning      : BOOL;
tTuningTimeout   : TIME:=T#15m;
eTuningMode      : E_BACNETPIDTUNINGMODE:=BACnetPIDTuningMode_PI;
fAccLimit        : REAL:=0.0;
```

bAutoTuning: Startet internes Autotuning.

tTuningTimeout: Maximale Laufzeit des Autotunings in [ms].

eTuningMode: Vorauswahl der zu bestimmenden Regelparameter [▶ 346] (P, PI, PD, PID).

fAccLimit: Parameter zur Begrenzung der Stellgrößenbeschleunigung (Rampe) *0* = unbegrenzt oder [1/s] (positiv und negativ).

VAR_OUPUT

```
bReady                   : BOOL;
bTuningActive            : BOOL;
bTuningWaitPV            : BOOL;
bTuningWaitCV            : BOOL;
bTuningDone              : BOOL;
bTuningAborted           : BOOL;
fSetPoint                : REAL;
fControlledValue         : REAL;
fPresentValue            : REAL;
```

```
fPropBand                : REAL;
bOverridden              : BOOL;
bOutOfService            : BOOL;
bFault                   : BOOL;
bInAlarm                 : BOOL;
bOpenLoop                : BOOL;
bCommFailure             : BOOL;
bOtherFault              : BOOL;
eEventState              : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal  : BOOL;
bAckedTrans_ToFault      : BOOL;
bAckedTrans_ToNormal     : BOOL;
tObjectID                : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                   : BOOL;
nErrorId                 : UINT;;
eMode                    : E_BACNETLOOPMODE;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bTuningActive: Autotuning ist aktiv.

bTuningWaitPV: Autotuning erwartet das Erreichen des vorgegebenen Sollwertsprungs.

bTuningWaitCV: Autotuning erwartet das Erreichen der Istwertantwort.

bTuningAborted: Autotuning wurde abgebrochen (siehe **nErrorId**).

fSetPoint: Rückmeldung der Regelvorgabe (W, Sollwert).

fControlledValue: Rückmeldung der aktuellen Prozessgröße (X, Istwert).

fPresentValue: Rückmeldung der aktuellen Regelausgabe (Y, Stellwert). Achtung: *Present_Value* und *Controlled_Variable_Value* können schnell zu Verwechslungen führen (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Properties *Present_Value*, *Controlled_Variable_Value* und *Controlled_Variable_Reference*)!

fPropBand: Rückmeldung der aktuellen Regelausgabe in Prozent (-100%...+100%) in Relation zur minimalen und maximalen Regelausgabe (Properties *Minimum_Output* und *Maximum_Output*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Status_Flags*.

bOpenLoop, bCommFailure, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property Event_State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Acked_Transitions*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

eMode: Ausgabe der Reglerbetriebsart. Unterschieden wird in 2 Modi: 1. Standardform und 2. Idealform (siehe E_BACNETLOOPMODE [▶ 343]). Erkannt wird die Betriebsart automatisch anhand der eingestellten Parametereinheit für den Parameter I und/oder D als "*Time_seconds*" → Standardform. Sind diese jedoch als Faktoren "*Other_no_units* " konfiguriert, wird die Betriebsart "Idealform" erkannt. Bei der Idealform handelt es sich um eine reine Parallelschaltung der 3 Regelgrößen (P, I und D). Die Idealform ermöglicht eine praxisnahe Konfiguration des Reglers, da die Parameter ohne direkte Abhängigkeit optimiert werden können. So kann aus dem Streckenverhalten, direkt auf Regelparameter geschlossen werden.

VAR_IN_OUT

```
Device                : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

Konfiguration des Reglers

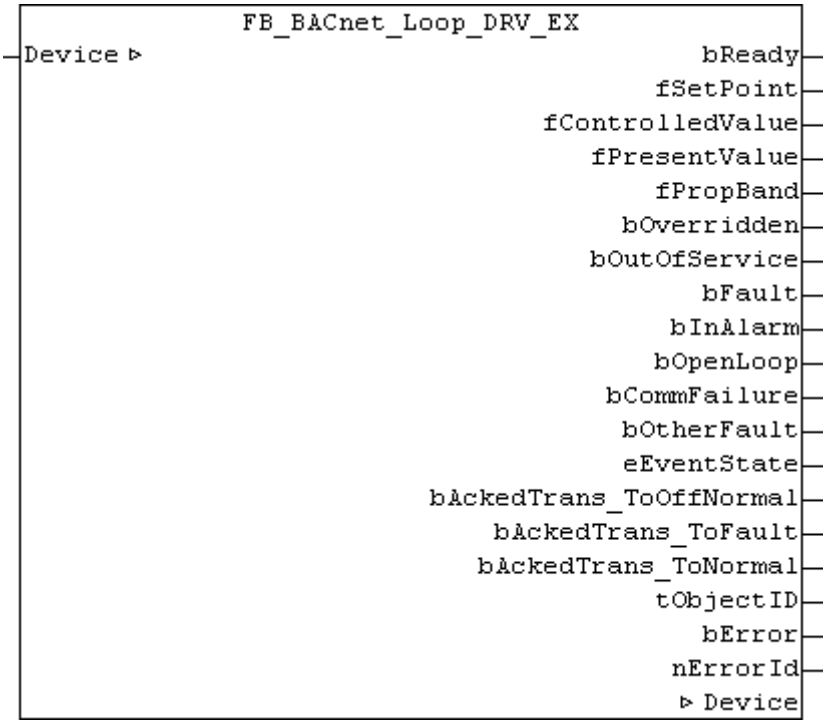
Die Konfiguration des Reglers erfolgt mit Hilfe der BACnet-Properties: *Action*, *Proportional_Constant* (P-Faktor), *Integral_Constant* (I-Faktor), *Derivative_Constant* (D-Faktor), *Bias* (Ausgabe-Offset), *Maximum_Output* (Maximale Regelausgabe) und *Minimum_Output* (Minimale Regelausgabe). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop*.

Bestimmung der Regelparameter im Modus Idealform, am Beispiel einer Temperaturregelung:

- Bestimmung des P-Faktors; Annahme: Bei einer Temperaturdifferenz von 2 Kelvin soll die Heizleistung 10% betragen → **Kp** = 5 = 10% / 2K;
- Bestimmung des I-Faktors; Annahme: Bei einer bleibenden Temperaturdifferenz von 1 Kelvin soll die Heizleistung pro Minute um 1% steigen bzw. abnehmen → **Ki** = 0.17 = 1% / (60s * 1K).

In der Regel wird jedoch die Standardform eingesetzt. Für die Bestimmung der Regelparameter in Standardform, wird üblicherweise im ersten Schritt die Sprungantwort der Strecke und für die Optimierung im zweiten Schritt das Regelverhalten selbst ausgewertet.

4.2.27 FB_BACnet_Loop_DRV_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Loop_DRV_EX kann lesend auf ein BACnet-Objekt vom Typ *Loop* zugegriffen werden. Im Unterschied zu der Standard- bzw. *_EX*-Variante des Bausteins, wird der Regelalgorithmus innerhalb des BACnet-Stacks realisiert (siehe auch FB_BACnet_Loop_EX [▶ 204]). Das *Loop* Objekt befindet sich somit auch ohne lauffähiges PLC-Programm im Regelmodus.

VAR_OUPUT

```
bReady                : BOOL;
fSetPoint             : REAL;
fControlledValue      : REAL;
```

```
fPresentValue            : REAL;
fPropBand                : REAL;
bOverridden              : BOOL;
bOutOfService            : BOOL;
bFault                   : BOOL;
bInAlarm                 : BOOL;
bOpenLoop                : BOOL;
bCommFailure             : BOOL;
bOtherFault              : BOOL;
eEventState              : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal  : BOOL;
bAckedTrans_ToFault      : BOOL;
bAckedTrans_ToNormal     : BOOL;
tObjectID                : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                   : BOOL;
nErrorId                 : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fSetPoint: Rückmeldung der Regelvorgabe (W, Sollwert).

fControlledValue: Rückmeldung der aktuellen Prozessgröße (X, Istwert).

fPresentValue: Rückmeldung der aktuellen Regelausgabe (Y, Stellwert). Achtung: *Present_Value* und *Controlled_Variable_Value* können schnell zu Verwechslungen führen (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Properties *Present_Value*, *Controlled_Variable_Value* und *Controlled_Variable_Reference*)!

fPropBand: Rückmeldung der aktuellen Regelausgabe in Prozent (-100%...+100%) in Relation zur minimalen und maximalen Regelausgabe (Properties *Minimum_Output* und *Maximum_Output*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Status_Flags*.

bOpenLoop, bCommFailure, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property Event_State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Acked_Transitions*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

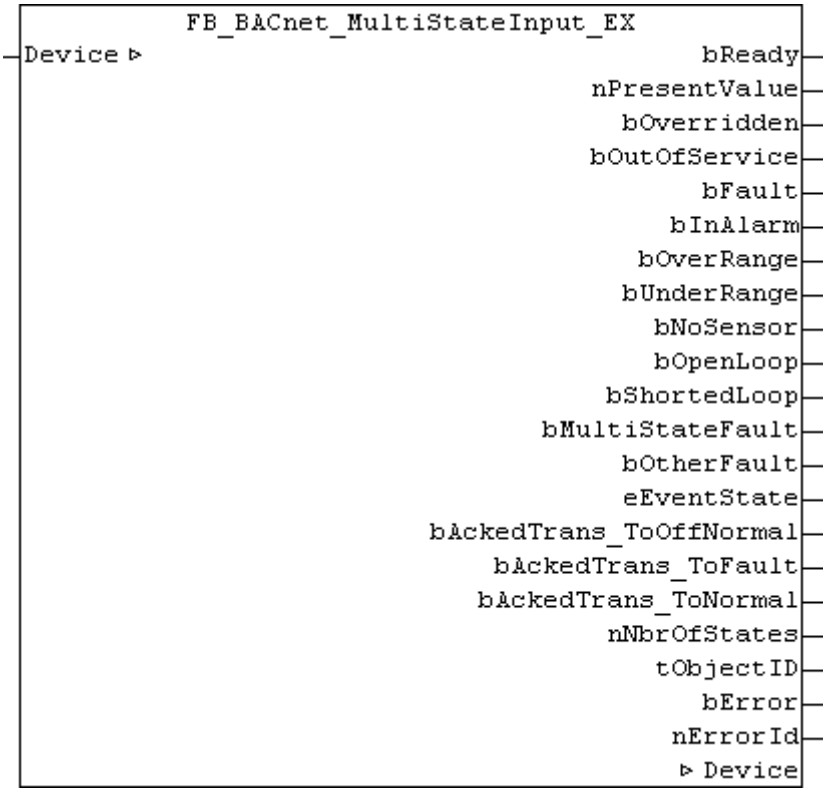
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device               : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.28 FB_BACnet_MultiStateInput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_MultiStateInput_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *MultiStateInput* zugegriffen werden.

VAR_OUPUT

```
bReady                  : BOOL;
nPresentValue           : UDINT;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bOverRange              : BOOL;
bUnderRange             : BOOL;
bNoSensor               : BOOL;
bOpenLoop               : BOOL;
bShortedLoop            : BOOL;
bMultiStateFault        : BOOL;
bOtherFault             : BOOL;
eEventState             : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
nNbrOfStates            : UDINT;
tObjectID               : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Status_Flags*.

bOverRange, bUnderRange, bNoSensor, bOpenLoop, bShortedLoop, bMultiStateFault, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Reliability*.

eEventState: E_BACNETEVENTSTATE [▶ 339], siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Event_State*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Acked_Transitions*).

nNbrOfStates: Meldet die verfügbare Anzahl von Werten, die das *Present_Value* des MultiStateInput-Objekts annehmen kann (1...*nNbrOfStates*). Ist "nNbrOfStates" gleich 0, dann gibt es keinen gültigen "State" den das Objekt annehmen kann (*Present_Value* ist 0).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

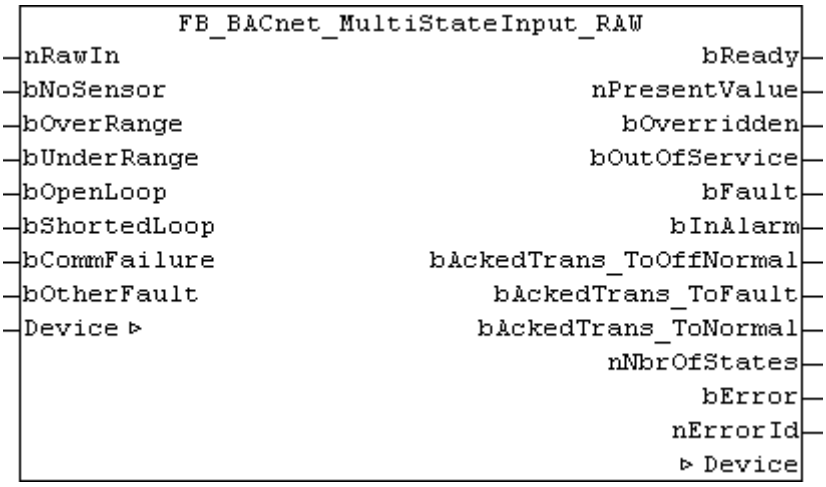
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device              : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.29 FB_BACnet_MultiStateInput_RAW



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_MultiStateInput_RAW kann lesend und schreibend auf ein BACnet-Objekt vom Typ *MultiStateInput* zugegriffen werden.

Im Unterschied zur Standard- bzw. *_EX*-Variante des Bausteins, wird der Rohwert und der Zustand der Property *Reliablitity* durch Funktionsbaustein-Eingänge bereitgestellt und nicht durch die IO-Hardware direkt. So kann z.B. ein mehrstufiger Zustand aus einem Sub-Bussystems in SPS-Code auf ein BACnet-Objekt abgebildet werden (Signalumsetzung von Sub-Bussystemen oder virtuellen Datenpunkten nach BACnet, siehe Beispiel [▶ 210]).

VAR_INPUT

```
nRawIn              : DWORD;
bNoSensor           : BOOL;
bOverRange          : BOOL;
bUnderRange         : BOOL;
```

```
bOpenLoop            : BOOL;
bShortedLoop         : BOOL;
bCommFailure         : BOOL;
bOtherFault          : BOOL;
```

nRawIn: Rohwerteingang des Objekts im Wertebereich 1 .. *Number_Of_States*. Der Eingang wird mit dem Prozessdatum "PresentValue" des BACnet-Objekts verknüpft. Der Wert aus **nRawIn** wird direkt in die Property *Present_Value* geschrieben (vorausgesetzt der Objektzustand ist nicht *out_of_service*).

bNoSensor, bOverRange, bUnderRange, bOpenLoop, bShortedLoop, bCommFailure, bOtherFault: *TRUE* am Eingang setzt den entsprechenden Zustand [▶ 352] der Property *Reliability*. Die Priorität fällt mit der Reihenfolge der Eingänge (**bNoSensor** höchste und **bOtherFault** niedrigste Priorität). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Reliability*.

VAR_OUPUT

```
bReady                  : BOOL;
nPresentValue           : UDINT;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
nNbrOfStates            : UDINT;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Status_Flags*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Acked_Transitions*).

nNbrOfStates: Meldet die verfügbare Anzahl von Werten, die das *Present_Value* des MultiStateInput-Objekts annehmen kann (1...*nNbrOfStates*). Ist "nNbrOfStates" gleich 0, dann gibt es keinen gültigen "State" den das Objekt annehmen kann (*Present_Value* ist 0).

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet Globals [▶ 330].

VAR_IN_OUT

```
Device               : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung eines ganzzahligen Werts aus EIB in die Property *Present_Value* eines MultiStateInput-Objekts gezeigt:

```
0002 VAR
0003 (* EIB *)
0004     EIB          : KL6301;
0005     EIB8BitIn    : EIB_8BIT_UNSIGN_REC;
0006
0007 (* BACnet *)
0008     Device       : FB_BACnet_Device;
0009     MIRaw        : FB_BACnet_MultiStateInput_RAW;
0010 END_VAR
```

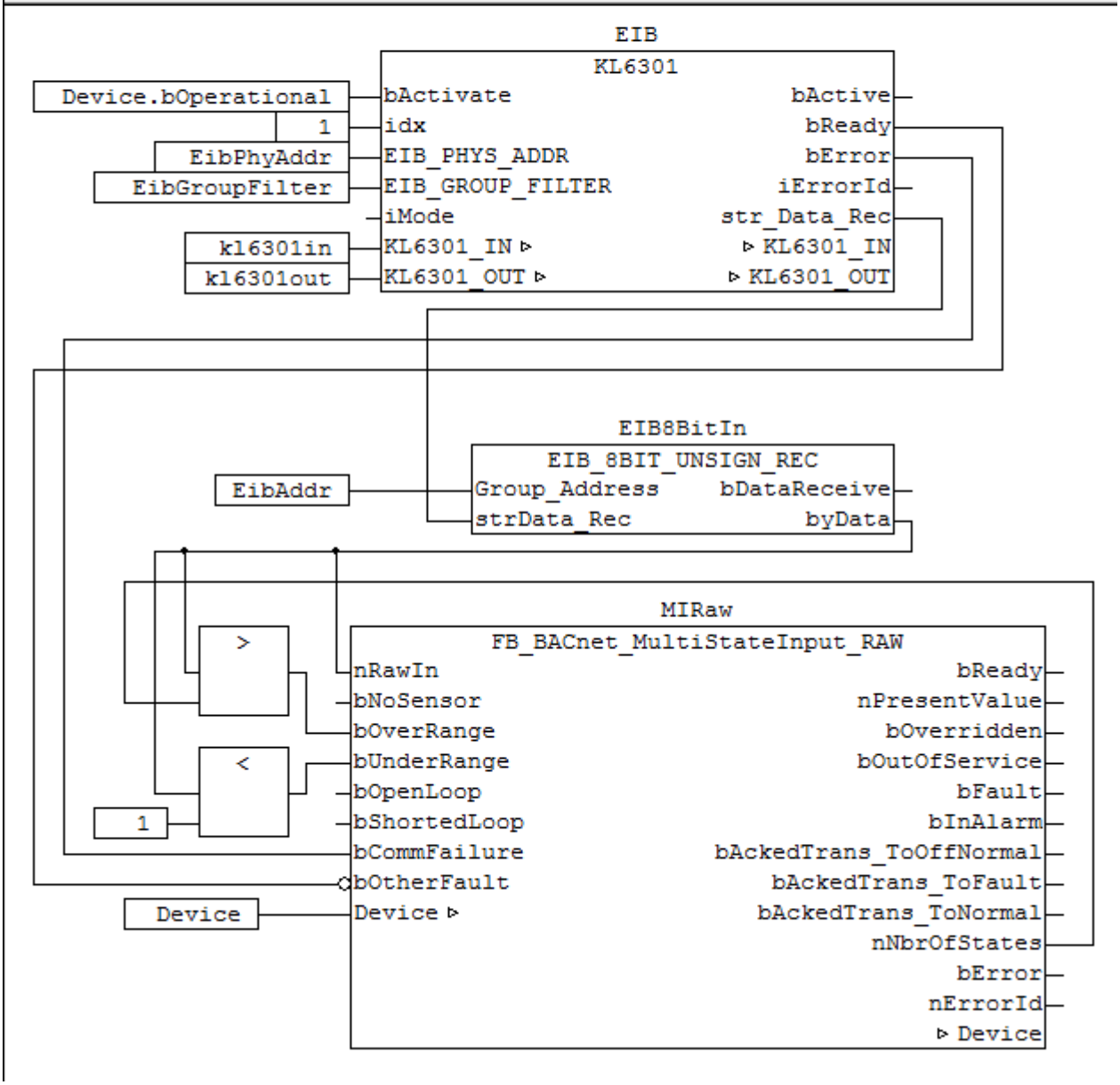
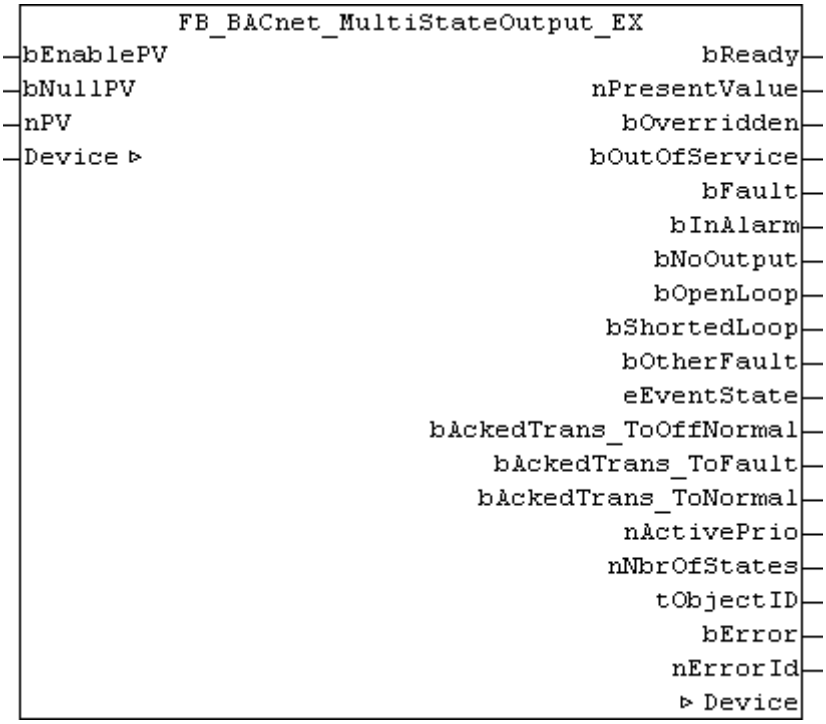


Abb. 15: Bild-1: Beispiel für die Umsetzung eines ganzzahligen Werts aus EIB in die Property Present_Value im PLC Programm.

Dieses Beispiel setzt die Bibliothek "TcEIB.lib" voraus. Siehe Dokumentation zur Klemme KL6301 für weitere Informationen zu EIB.

4.2.30 FB_BACnet_MultiStateOutput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_MultiStateOutput_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *MultiStateOutput* zugegriffen werden.

VAR_INPUT

```
bEnablePV   : BOOL;
bNullPV     : BOOL;
nPV         : UDINT;
```

bEnable: *TRUE* = Das Prozessdatum wird aktiviert; der Wert, der sich aus **bNull** bzw. **nState** ergibt, wird in das entsprechende BACnet Object geschrieben, *FALSE* = Prozessdatum wird deaktiviert

bNull: *TRUE* = Null-Schreiben des BACnet Objekts (z.B. Löschen einer Priorität), *FALSE* = Wert aus **nState** schreiben

nPV: Wert der in das BACnet Object geschrieben wird, wenn **bEnable** = *TRUE* und **bNull** = *FALSE* sind. Multi-State im Bereich [1 .. *Number_Of_States*].

VAR_OUPUT

```
bReady                  : BOOL;
nPresentValue           : UDINT;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bNoOutput               : BOOL;
bOpenLoop               : BOOL;
bShortedLoop            : BOOL;
bOtherFault             : BOOL;
eEventState             : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
nActivePrio             : UINT;
nNbrOfStates            : UDINT;
tObjectID               : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Status_Flags*.

bNoOutput, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Reliability*.

eEventState: E_BACNETEVENTSTATE [▶ 339], siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Event_State*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Acked_Transitions*).

nActivePrio: Gibt die aktuell wirksame Priorität des Priority-Array an, die auf das *Present_Value* wirkt (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Present_Value*).

nNbrOfStates: Meldet die verfügbare Anzahl von Werten, die das *Present_Value* des MultiStateOutput-Objekts annehmen kann (1...*nNbrOfStates*). Ist **nNbrOfStates** gleich 0, dann gibt es keinen gültigen "State" den das Objekt annehmen kann (*Present_Value* ist 0).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

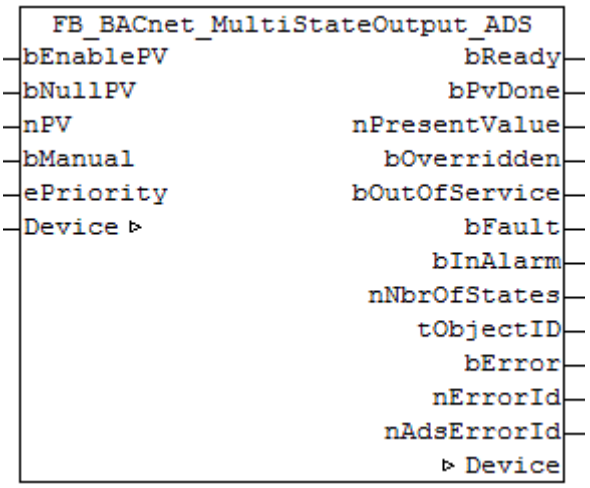
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device               : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.31 FB_BACnet_MultiStateOutput_ADS



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_MultiStateOutput_ADS kann lesend und schreibend auf ein BACnet-Objekt vom Typ *MultiStateOutput* zugegriffen werden.

Das Schrieben der Property *Present_Value* wird, im Unterschied zur Standard- und *_EX*-Variante, mittels ADS-WRITE realisiert (azyklisch).

VAR_INPUT

```
bEnablePV   : BOOL;
bNullPV     : BOOL;
nPV         : UDINT;
bManual     : BOOL;
ePriority   : E_BACNETPRIORITY:=BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs **nPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus **nPV** auf die kommandierbare Property *Present_Value* mit Priorität aus **ePriority** (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Wird der Eingang auf *FALSE* gesetzt, erfolgt kein Schreibzugriff (keine Änderung) mehr.

bNullPV: Überschreibt den Eintrag der kommandierbaren Property *Present_Values* in der Priorität **ePriority** mit dem Wert *NULL*, wenn **bEnablePV** auf *TRUE* gesetzt ist. Der Eingang **nPV** wird ignoriert, so lange **bNullPV** auf *TRUE* gesetzt ist.

nPV: Wert der an die entsprechenden Stelle der Property *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** auf *TRUE* und **bNullPV** auf *FALSE* gesetzt ist. Die Priorität wird mit dem Eingang **ePriority** bestimmt (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Das Schreiben wird bei Wertänderung azyklisch ausgeführt (ADS-WRITE).

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung geschieht, ist es potenziell möglich, dass die Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:

- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE* → *TRUE*: Schreiben der Property *Present_Value* bei Flankenwechsel.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 199]).

ePriority: Vorgabe der Priorität für Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 346]).

VAR_OUPUT

```
bReady          : BOOL;
bPvDone         : BOOL;
nPresentValue   : UDINT;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
nNbrOfStates    : UDINT;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
nAdsErrorId     : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Status_Flags*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet Globals [▶ 330].

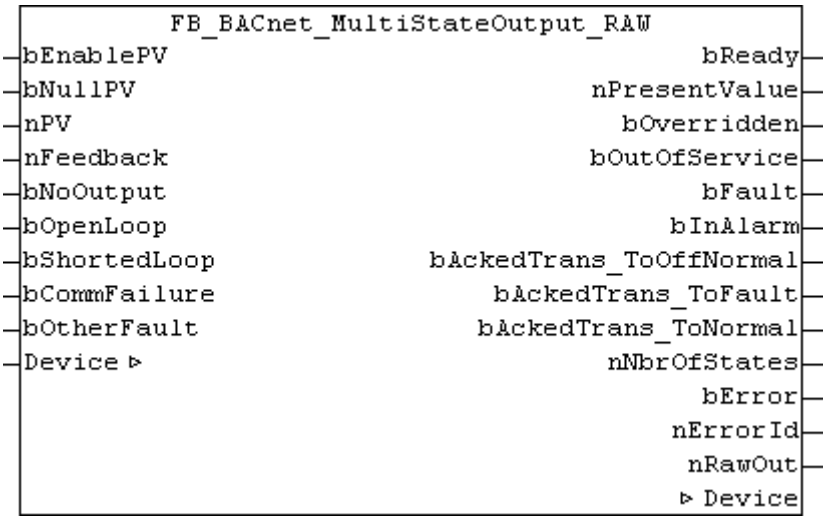
nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.32 FB_BACnet_MultiStateOutput_RAW



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_MultiStateOutput_RAW kann lesend und schreibend auf ein BACnet-Objekt vom Typ *MultiStateOutput* zugegriffen werden.

Im Unterschied zur Standard- bzw. *_EX*-Variante des Bausteins, wird das Flag der Property *Status_Flags* / *overridden*, die Property *Feedback_Value* und der Wert der Property *Reliablitity* durch Funktionsbaustein-Eingänge bereitgestellt und aus dem PLC-Programm auf das BACnet Objekt abgebildet. Der Zustand der Property *Present_Value* wird durch das PLC-Programm auf die Hardware bzw. auf das Sub-Bussystem gemappt. Dies geschieht sonst direkt durch die IO-Hardware. So kann z.B. der Ausgangswert an ein Sub-Bussystems in PLC-Code von einem BACnet-Objekt abgebildet werden (Signalumsetzung auf Sub-Bussysteme oder virtuellen Datenpunkte von BACnet, siehe Beispiel [▶ 216]).

VAR_INPUT

```
bEnablePV           : BOOL;
bNullPV             : BOOL;
nPV                 : UDINT;
nFeedback           : DWORD;
bNoOutput           : BOOL;
bOpenLoop           : BOOL;
bShortedLoop        : BOOL;
bCommFailure        : BOOL;
bOtherFault         : BOOL;
```

bEnable: *TRUE* = Das Prozessdatum wird aktiviert; der Wert, der sich aus **bNull** bzw. **nState** ergibt, wird in das entsprechende BACnet Object geschrieben, *FALSE* = Prozessdatum wird deaktiviert

bNull: *TRUE* = Null-Schreiben des BACnet Objekts (z.B. Löschen einer Priorität), *FALSE* = Wert aus **nState** schreiben

nPV: Wert der in das BACnet Object geschreiben wird, wenn **bEnable** = *TRUE* und **bNull** = *FALSE* sind. Multi-State im Bereich [1 .. *Number_Of_States*].

nFeedback: Signalrückmeldung an das BACnet Objekt. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Feedback_Value*.

bNoOutput, bOpenLoop, bShortedLoop, bCommFailure, bOtherFault: *TRUE* am Eingang setzt den entsprechenden Zustand [▶ 352] der Property *Reliability*. Die Priorität fällt mit der Reihenfolge der Eingänge (**bNoOutput** höchste und **bOtherFault** niedrigste Priorität). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Reliability*.

VAR_OUPUT

```
bReady                  : BOOL;
nPresentValue           : UDINT;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
nNbrOfStates            : UDINT;
bError                  : BOOL;
nErrorId                : UINT;
nRawOut                 : DWORD;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Status_Flags*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Acked_Transitions*).

nNbrOfStates: Meldet die verfügbare Anzahl von Werten, die das *Present_Value* des MultiStateOutput-Objekts annehmen kann (1...*nNbrOfStates*). Ist **nNbrOfStates** gleich 0, dann gibt es keinen gültigen "State" den das Objekt annehmen kann (*Present_Value* ist 0).

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet Globals [▶ 330].

nRawOut: Rohwertausgang des Objekts. Der Ausgang wird mit dem Prozessdatum "PresentValue" des BACnet-Objekts verknüpft.

VAR_IN_OUT

```
Device              : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung der Property *Present_Value* eines MultiStateOutput-Objekts nach EIB gezeigt:

```
0002 VAR
0003 (* EIB *)
0004     EIB          : KL6301;
0005     EIB8BitOut   : EIB_8BIT_UNSIGN_SEND;
0006
0007 (* BACnet *)
0008     Device       : FB_BACnet_Device;
0009     MORaw        : FB_BACnet_MultiStateOutput_RAW;
0010 END_VAR
```

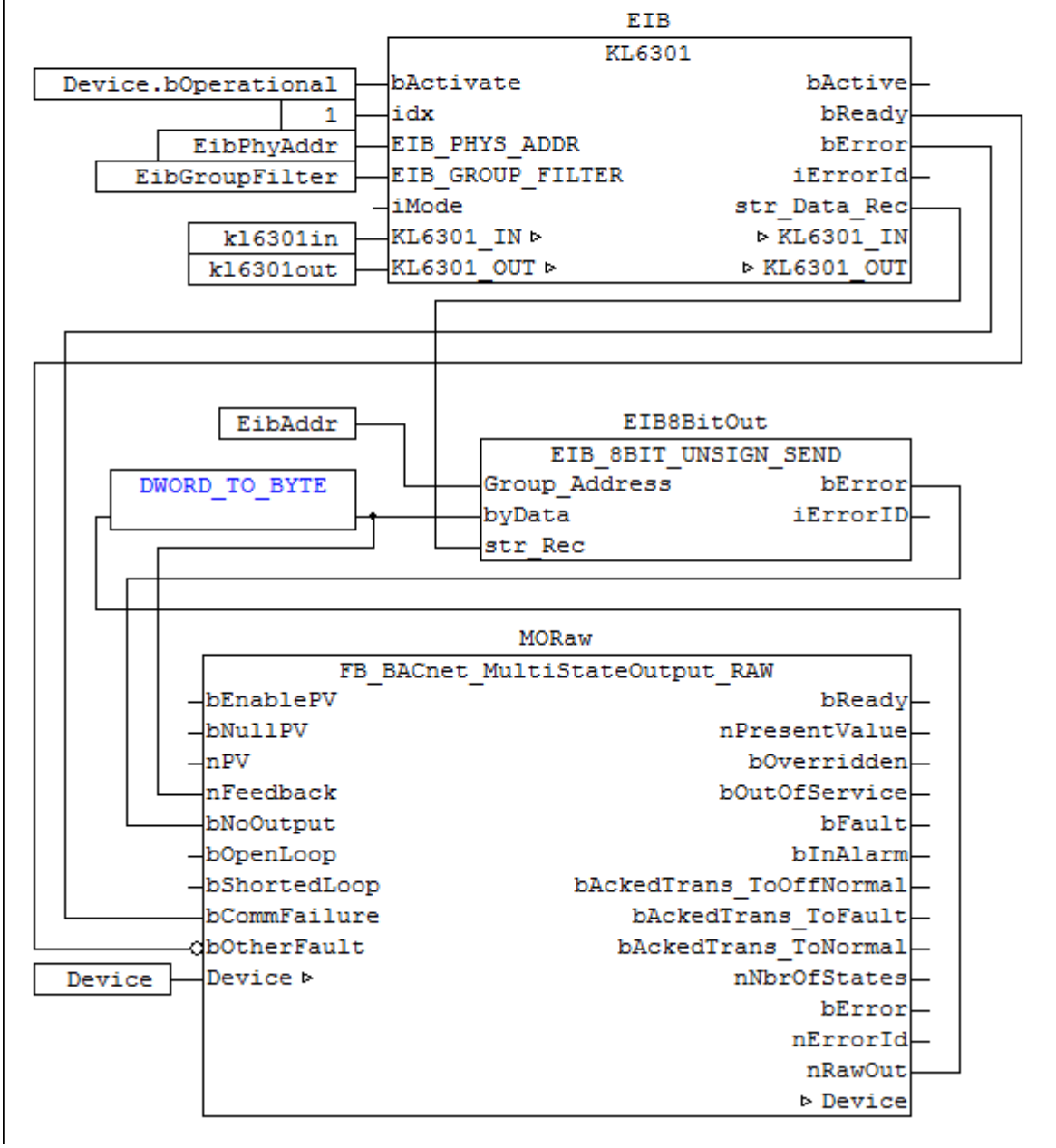
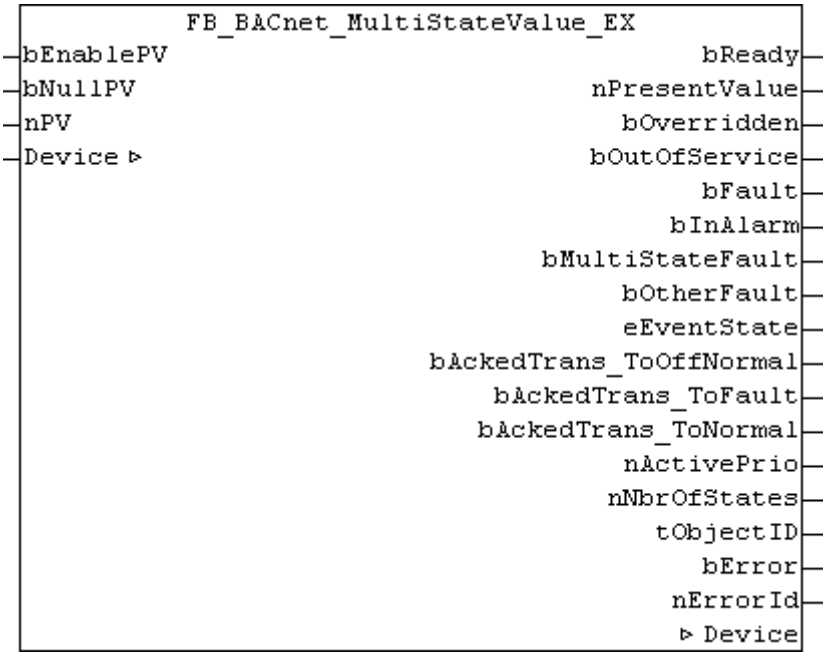


Abb. 16: Bild-1: Beispiel für die Umsetzung der Property Present_Value nach EIB im PLC Programm.

Dieses Beispiel setzt die Bibliothek "TcEIB.lib" voraus. Siehe Dokumentation zur Klemme KL6301 für weitere Informationen zu EIB.

4.2.33 FB_BACnet_MultiStateValue_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_MultiStateValue_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *MultiStateValue* zugegriffen werden.

VAR_INPUT

```
bEnablePV         : BOOL;
bNullPV           : BOOL;
nPV               : UDINT;
```

bEnable: *TRUE* = Das Prozessdatum wird aktiviert; der Wert, der sich aus **bNull** bzw. **nState** ergibt, wird in das entsprechende BACnet Object geschrieben, *FALSE* = Prozessdatum wird deaktiviert

bNull: *TRUE* = Null-Schreiben des BACnet Objekts (z.B. Löschen einer Priorität), *FALSE* = Wert aus **nState** schreiben

nState: Wert der in das BACnet Object geschrieben wird, wenn **bEnable** = *TRUE* und **bNull** = *FALSE* sind. Multi-State im Bereich [1 .. *Max_Number_Of_States*].

VAR_OUPUT

```
bReady                  : BOOL;
nPresentValue           : UDINT;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bMultiStateFault        : BOOL;
bOtherFault             : BOOL;
eEventState             : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
nActivePrio             : UINT;
nNbrOfStates            : UDINT;
tObjectID               : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Status_Flags*.

bMultiStateFault, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Reliability*.

eEventState: E_BACNETEVENTSTATE [▶ 339], siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Event_State*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Acked_Transitions*).

nNbrOfStates: Meldet die verfügbare Anzahl von Werten, die das *Present_Value* des MultiStateValue-Objekts annehmen kann (1...*nNbrOfStates*). Ist "nNbrOfStates" gleich 0, dann gibt es keinen gültigen "State" den das Objekt annehmen kann (*Present_Value* ist 0).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

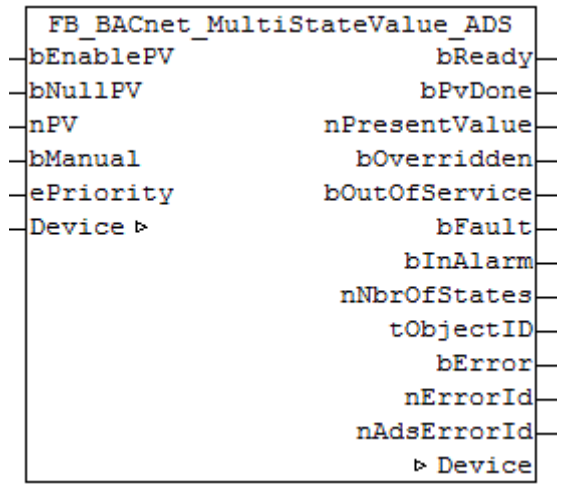
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device               : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.34 FB_BACnet_MultiStateValue_ADS



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_MultiStateValue_ADS kann lesend und schreibend auf ein BACnet-Objekt vom Typ *MultiStateValue* zugegriffen werden.

Das Schreiben der Property *Present_Value* wird, im Unterschied zur Standard- und *_EX*-Variante, mittels ADS-WRITE realisiert (azyklisch).

VAR_INPUT

```
bEnablePV  : BOOL;
bNullPV    : BOOL;
nPV        : UDINT;
bManual    : BOOL;
ePriority  : E_BACNETPRIORITY:=BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs **nPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus **nPV** auf die kommandierbare Property *Present_Value* mit Priorität aus **ePriority** (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Wird der Eingang auf *FALSE* gesetzt, erfolgt kein Schreibzugriff (keine Änderung) mehr.

bNullPV: Überschreibt den Eintrag der kommandierbaren Property *Present_Values* in der Priorität **ePriority** mit dem Wert *NULL*, wenn **bEnablePV** auf *TRUE* gesetzt ist. Der Eingang **nPV** wird ignoriert, so lange **bNullPV** auf *TRUE* gesetzt ist.

nPV: Wert der an die entsprechenden Stelle der Property *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** auf *TRUE* und **bNullPV** auf *FALSE* gesetzt ist. Die Priorität wird mit dem Eingang **ePriority** bestimmt (Default: 12, wenn der Eingang **ePriority** nicht beschaltet ist). Das Schreiben wird bei Wertänderung azyklisch ausgeführt (ADS-WRITE).

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung geschieht, ist es potenziell möglich, dass die Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:

- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE* → *TRUE*: Schreiben der Property *Present_Value* bei Flankenwechsel.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 199]).

ePriority: Vorgabe der Priorität für Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 346]).

VAR_OUPUT

```
bReady          : BOOL;
bPvDone         : BOOL;
nPresentValue   : UDINT;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
nNbrOfStates    : UDINT;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
nAdsErrorId     : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Status_Flags*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

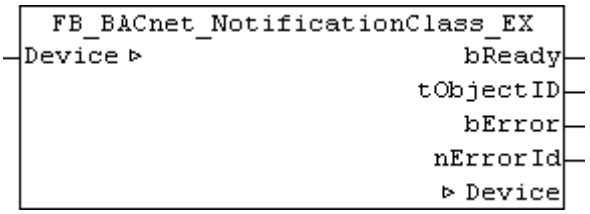
nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.35 FB_BACnet_NotificationClass_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_NotificationClass_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *NotificationClass* zugegriffen werden.

VAR_OUPUT

```
bReady      : BOOL;
tObjectID   : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError      : BOOL;
nErrorId    : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

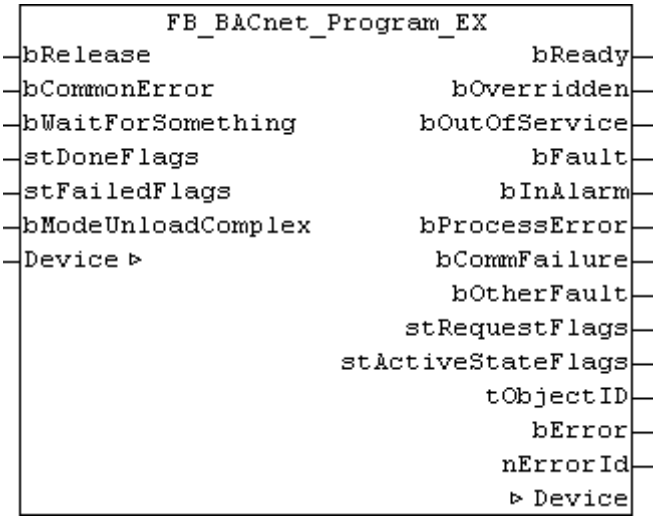
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device      : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.36 FB_BACnet_Program_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Program_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Program* zugegriffen werden.

Der Funktionsbaustein FB_BACnet_Program enthält die in der BACnet-Spezifikation beschriebenen *State-Transistions* und bildet diese auf dem FB-Interface mit Hilfe der Strukturen: *stDoneFlags*, *stFailedFlags* und *stRequestFlags* ab (siehe unter Transition-Diagram [▶ 224] Bild-1).

VAR_INPUT

```
bRelease            : BOOL;
bCommonError        : BOOL;
bWaitForSomething   : BOOL;
stDoneFlags         : ST_BACnet_ProgramHandshakeRequests;
stFailedFlags       : ST_BACnet_ProgramHandshakeRequests;
bModeUnloadComplex  : BOOL:=TRUE;
```

bRelease: Gibt die Freigabe für das Abarbeiten von Anforderungen. Wenn **bRelease** = *FALSE* gesetzt ist, dann verharrt das Programm im Zustand *HALTED*.

bCommonError: Fehler bei der Ausführung eines *Program_Change*-Requestsaus der Struktur **stRequestFlags**. Das Setzten des Eingangs bei laufender Ausführung eines *Program_Change*-Requests führt zu einer Abbruchbedingung (siehe unter Transition-Diagram [▶ 224] Bild-1).

bWaitForSomething: Befindet sich das BACnet-Objekt im Zustand *RUNNING* kann mit *TRUE* Setzen des Eingangs auf den Zustand *WAITING* umgeschaltet werden, so dass das BACnet-Objekt für weitere Requests außer *Unload* und *Halt* gesperrt wird (siehe unter Transition-Diagram [▶ 224] Bild-1).
Um in den Zustand *RUNNING* zurückzukehren muss der Eingang auf *FALSE* zurückgesetzt werden.

stDoneFlags: Die Flags innerhalb der Eingangsstruktur dienen der Quittierung von anstehenden *Program_Change*-Requests aus der Ausgangsstruktur **stRequestFlags**. Ein Signalwechsel eines der Flags von *FALSE --> TRUE* bestätigt die Ausführung des entsprechenden *Program_Change*-Requests.

stFailedFlags: Die Flags innerhalb der Eingangsstruktur dienen der Abbruchmeldung des anstehenden *Program_Change*-Requests aus der Ausgangsstruktur **stRequestFlags**. Ein Signalwechsel eines der Flags von *FALSE --> TRUE* bricht die Ausführung des entsprechenden *Program_Change*-Requests ab.

bModeUnloadComplex: Wurde der Modus "Unload Complex" mit Hilfe des Eingangs aktiviert, so wechselt der BACnet-Objekt Status von *UNLOADING* zu *IDLE* nach der Ausführung des Requests *Unload* (siehe unter Transition-Diagram [▶ 224] Bild-1). Wird der Eingang auf *FALSE* gesetzt, dann verharrt das BACnet-Objekt im Zustand *UNLOADING* nach dem der Request *Unload* ausgeführt wurde.

VAR_OUPUT

```
bReady               : BOOL;
bOverridden          : BOOL;
bOutOfService        : BOOL;
bFault               : BOOL;
bInAlarm             : BOOL;
bProcessError        : BOOL;
bCommFailure         : BOOL;
bOtherFault          : BOOL;
stRequestFlags       : ST_BACnet_ProgramHandshakeRequests;
stActiveStateFlags   : ST_BACnet_ProgramHandshakeStates;
tObjectID            : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError               : BOOL;
nErrorId             : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Program* und Property *Status_Flags*.

bProcessError, bCommFailure, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Program* und Property *Reliability*.

stRequestFlags: Enthält die Flags zu den entsprechenden *Program_Change*-Requests. Ist ein Flag auf *TRUE* gesetzt, so wird eine Statusänderung des Programm-Objekts durch das SPS-Programm angefordert. Wurde die Änderung erfolgreich ausgeführt wird dies mit einem Signalwechsel *FALSE* → *TRUE* des entsprechenden Flags der Eingangsstruktur **stDoneFlags** bestätigt. Kann der Request von dem zugehörigen SPS-Programm nicht ausgeführt werden oder gibt es einen Abbruch während der Ausführung, so muss dies mit Hilfe des entsprechenden Flags der Eingangsstruktur **stFailedFlags** gemeldet werden. Der Abbruch wird dann im Status des BACnet-Objekts sichtbar.

stActiveStateFlags: Enthält Flags die den aktuellen Zustand des BACnet-Program-Objects widerspiegeln (siehe unter Transition-Diagram [▶ 224] Bild-1).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device           : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

Transition-Diagram

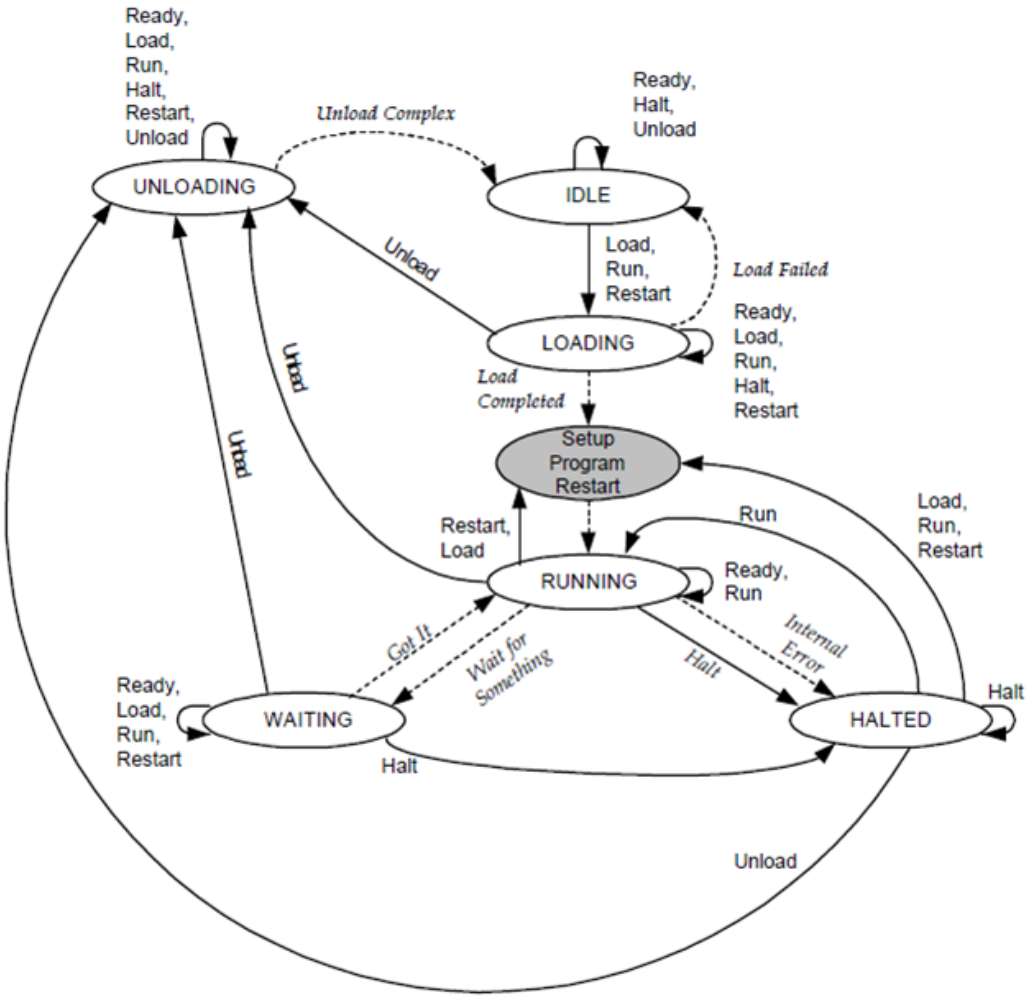
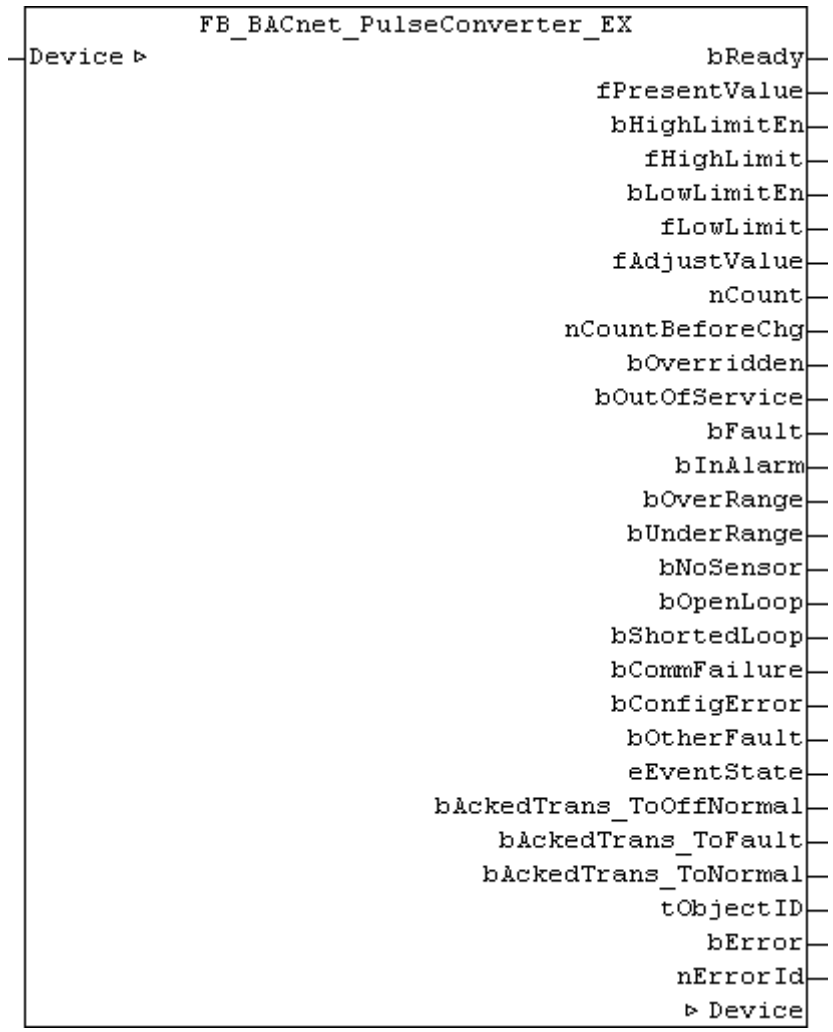


Abb. 17: Bild-1: aus BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt Program, Abbildung 12-3 "State Transitions for the program object"

4.2.37 FB_BACnet_PulseConverter_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_PulseConverter_EX kann lesend auf ein BACnet-Objekt vom Typ *PulseConverter* zugegriffen werden.

VAR_OUPUT

```
bReady                  : BOOL;
fPresentValue           : REAL;
bHighLimitEn            : BOOL;
fHighLimit              : REAL;
bLowLimitEn             : BOOL;
fLowLimit               : REAL;
fAdjustValue            : REAL;
nCount                  : UDINT;
nCountBeforeChg         : UDINT;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bOverRange              : BOOL;
bUnderRange             : BOOL;
bNoSensor               : BOOL;
bOpenLoop               : BOOL;
bShortedLoop            : BOOL;
bCommFailure            : BOOL;
bConfigError            : BOOL;
bOtherFault             : BOOL;
eEventState             : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
```

```
bAckedTrans_ToNormal     : BOOL;
tObjectID                : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError                   : BOOL;
nErrorId                 : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Present_Value*).

bHighLimitEn, fHighLimit, bLowLimitEn, fLowLimit: Status und Wert der *Present_Value* Meldegrenzen (unterer und oberer Grenzwert ab den Meldungen generiert werden). *FALSE* → Grenzwert nicht aktiv bzw. *TRUE* → Grenzwert aktiv; siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *High_Limit*, *Low_Limit* und *Limit_Enable*.

fAdjustValue: Wert der Property *Adjust_Value*. Mit Hilfe von *Adjust_Value* kann der Wert der Property *Present_Value* korrigiert werden. Dabei wird der Wert von *Adjust_Value* von *Present_Value* abgezogen. Der Wert der Propery Count wird ebenfalls, unter Berücksichtigung der Property *Scale_Factor*, korrigiert. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Adjust_Value*.

nCount: Wert der Property *Count*. *Count* repräsentiert die erfassten Eingangsimpulse bzw. -wertänderungen. Zudem kann der Wert von *Count* Korrekturen aus der Property *Adjust_Value* enthalten. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Count*.

nCountBeforeChg: Wert der Property *Count_Before_Change*. Enthält den Wert der Property *Count*, vor der Korrektur durch Property *Adjust_Value*. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Count_Before_Change*.

bOverridden, bOutOfService, **bFault**, **bInAlarm:** Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Status_Flags*.

bOverRange, bUnderRange, **bNoSensor**, **bOpenLoop**, **bShortedLoop**, **bCommFailure**, **bConfigError**, **bOtherFault:** Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Status_Flags*.

eEventState: E_BACNETEVENTSTATE [▶ 339], siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Event_State*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Acked_Transitions*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

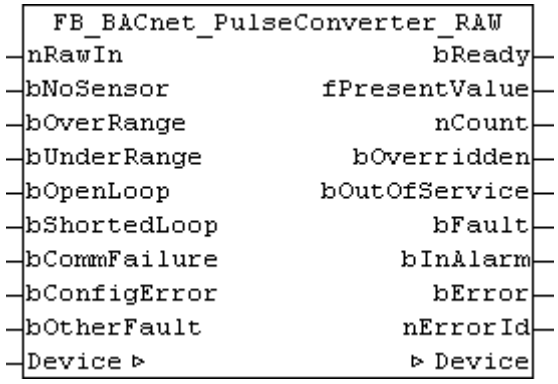
nErrorId: siehe globale Konstanten BACnet Globals [▶ 330].

VAR_IN_OUT

```
Device               : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.38 FB_BACnet_PulseConverter_RAW



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_PulseConverter_RAW kann lesend und schreibend auf ein BACnet-Objekt vom Typ *PulseConverter* zugegriffen werden.

Im Unterschied zur Standard- bzw. *_EX*-Variante des Bausteins, wird der Rohwert und der Zustand der Property *Reliablitity* durch Funktionsbaustein-Eingänge bereitgestellt und nicht durch die IO-Hardware direkt. So kann z.B. der Zustand eines Zähleingangs aus einem Sub-Bussystems in SPS-Code auf ein BACnet-Objekt abgebildet werden (Signalumsetzung von Sub-Bussystemen oder virtuellen Datenpunkten nach BACnet).

VAR_INPUT

```
nRawIn          : UINT;
bNoSensor       : BOOL;
bOverRange      : BOOL;
bUnderRange     : BOOL;
bOpenLoop       : BOOL;
bShortedLoop    : BOOL;
bCommFailure    : BOOL;
bConfigError    : BOOL;
bOtherFault     : BOOL;
```

nRawIn: Rohwerteingang des Objekts im Wertebereich -32768...32767. Der Eingang wird mit dem Prozessdatum "RawIoPulseConverterUnsignedValue" des BACnet-Objekts verknüpft. Wertänderungen von **nRawIn** werden mit der Property *Scale_Factor* zum Wert der Property *Present_Value* verrechnet (vorausgesetzt der Objektzustand ist nicht *out_of_service*).

bNoSensor, bOverRange, bUnderRange, bOpenLoop, bShortedLoop, bCommFailure, bConfigError, bOtherFault: *TRUE* am Eingang setzt den entsprechenden Zustand [▶ 352] der Property *Reliability*. Die Priorität fällt mit der Reihenfolge der Eingänge (**bNoSensor** höchste und **bOtherFault** niedrigste Priorität). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Reliability*.

VAR_OUPUT

```
bReady          : BOOL;
fPresentValue   : REAL;
nCount          : UDINT;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Present_Value*).

nCount: Wert der Property *Count*. *Count* repräsentiert die erfassten Eingangsimpulse bzw. -wertänderungen. Zudem kann der Wert von *Count* Korrekturen aus der Property *Adjust_Value* enthalten. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Count*.

bOverridden, bOutOfService, **bFault**, **bInAlarm:** Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Status_Flags*.

bError: Ein Fehler steht an.

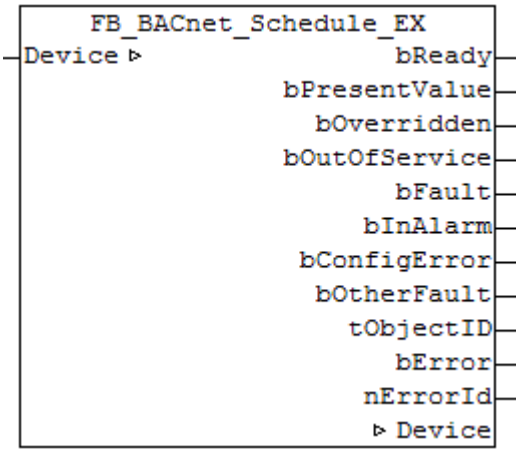
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.39 FB_BACnet_Schedule_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Schedule_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Schedule* zugegriffen werden.

VAR_OUPUT

```
bReady          : BOOL;
bPresentValue   : BOOL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bConfigError    : BOOL;
bOtherFault     : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Schedule* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Schedule* und Property *Status_Flags*.

bConfigError, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Schedule* und Property *Reliability*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

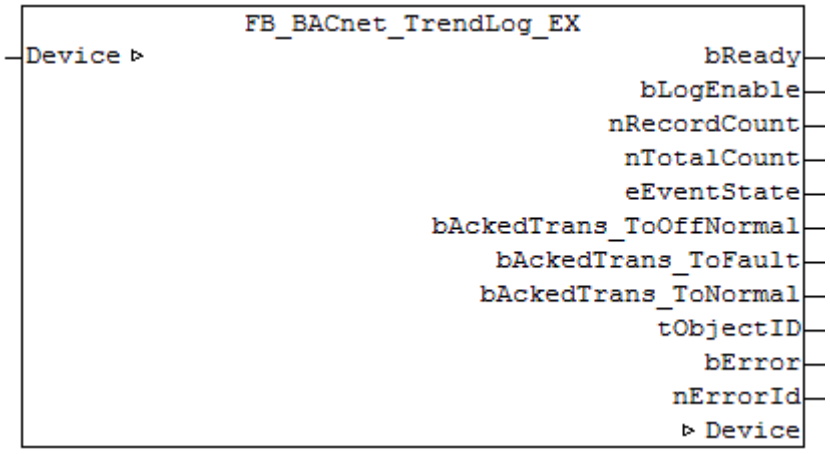
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
Device          : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.2.40 FB_BACnet_TrendLog_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_TrendLog_EX kann lesend auf ein BACnet-Objekt vom Typ *TrendLog* zugegriffen werden.

VAR_OUPUT

```
bReady                  : BOOL;
bLogEnable              : BOOL;
nRecordCount            : UDINT;
nTotalCount             : UDINT;
eEventState             : E_BACNETEVENTSTATE;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
tObjectID               : DINT:=-1;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bLogEnable: Die Aufzeichnung von Werten ist freigegeben. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Log_Enable*.

nRecordCount: Anzahl Einträge der Property *Log_Buffer*. Wird die Property *Record_Count* mit "0" beschrieben, dann wird der *Log_Buffer* zurück gesetzt. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Record_Count*.

nTotalCount: Gesamtanzahl bisheriger Einträge der Property Log_Buffer. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Total_Record_Count*.

eEventState: E_BACNETEVENTSTATE [▶ 339], siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Event_State*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Acked_Transitions*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

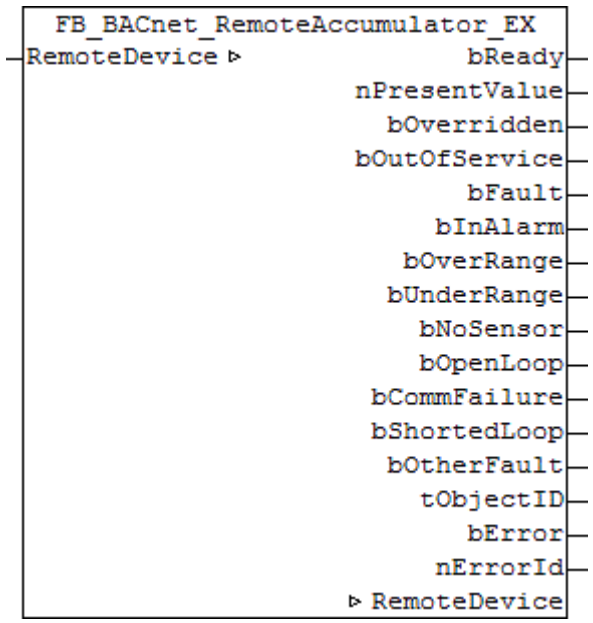
VAR_IN_OUT

```
Device               : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_Device [▶ 199] für weitere Informationen.

4.3 BACnet Client Objects

4.3.1 FB_BACnet_RemoteAccumulator_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteAccumulator_EX kann lesend auf ein BACnet-Objekt vom Typ *Accumulator* zugegriffen werden.

VAR_OUPUT

```
bReady          : BOOL;
nPresentValue   : UDINT;
bOverridden     : BOOL;
```

```
bOutOfService  : BOOL;
bFault         : BOOL;
bInAlarm       : BOOL;
bOverRange     : BOOL;
bUnderRange    : BOOL;
bNoSensor      : BOOL;
bOpenLoop      : BOOL;
bCommFailure   : BOOL;
bShortedLoop   : BOOL;
bOtherFault    : BOOL;
tObjectID      : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError         : BOOL;
nErrorId       : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Present_Value*).

nMaxPV: Aktueller Wert der Property *Max_Pres_Value* des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Max_Pres_Value*).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Status_Flags*.

bOverRange, bUnderRange, bNoSensor, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Event_State*.

tObjectID: Objekt ID des BACnet Objekts (Objekt Type und Objekt Instanz).

bError: Ein Fehler steht an.

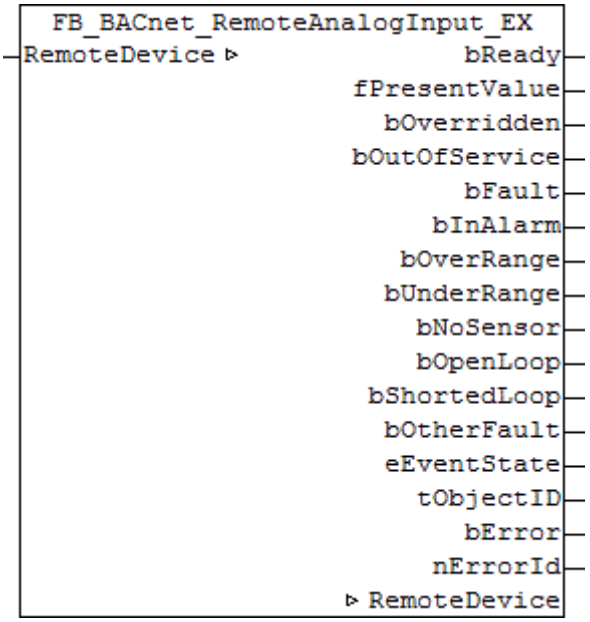
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice       : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.2 FB_BACnet_RemoteAnalogInput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteAnalogInput_EX kann lesend auf ein BACnet-Objekt vom Typ *AnalogInput* zugegriffen werden.

VAR_OUPUT

```
bReady          : BOOL;
fPresentValue   : REAL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bOverRange      : BOOL;
bUnderRange     : BOOL;
bNoSensor       : BOOL;
bOpenLoop       : BOOL;
bShortedLoop    : BOOL;
bOtherFault     : BOOL;
eEventState     : E_BACNETEVENTSTATE;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Present_Value*).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Status_Flags*.

bOverRange, bUnderRange, bNoSensor, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property Event State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Acked_Transitions*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

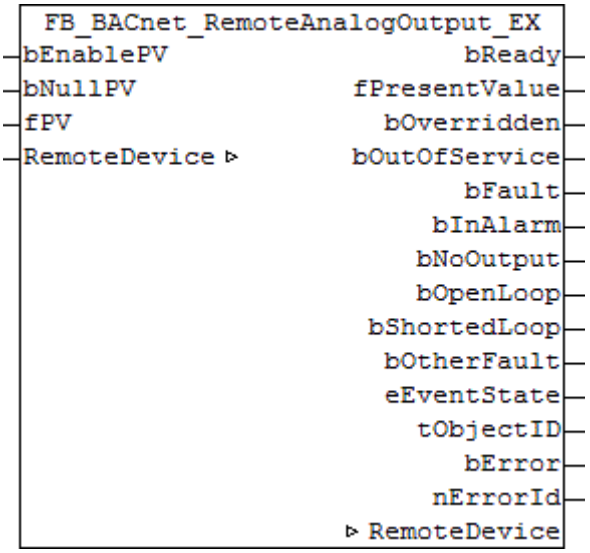
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice       : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.3 FB_BACnet_RemoteAnalogOutput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteAnalogOutput_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *AnalogOutput* zugegriffen werden.

VAR_INPUT

```
bEnablePV   : BOOL;
bNullPV     : BOOL;
fPV         : REAL;
```

bEnablePV: *TRUE* = Schreibfreigabe des Property-Werts; *FALSE* = Eintrag im BACnet Objekt nicht verändern (**bNullPV** und **fPV** werden unwirksam).

bNullPV: *TRUE* = Eintrag der Property löschen (*NULL*); anstelle des Werts von **fPV**; *FALSE* = Wert von **fPV** als Property Wert schreiben.

fPV: Wert der in die Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** = *TRUE* und **bNullPV** = *FALSE* sind. Das Schreiben der Prozessdaten erfolgt bei Änderung.

VAR_OUPUT

```
bReady          : BOOL;
fPresentValue   : REAL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bNoOutput       : BOOL;
bOpenLoop       : BOOL;
bShortedLoop    : BOOL;
bOtherFault     : BOOL;
```

```
eEventState     : E_BACNETEVENTSTATE;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Status_Flags*.

bNoOutput, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property Event_State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Acked_Transitions*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

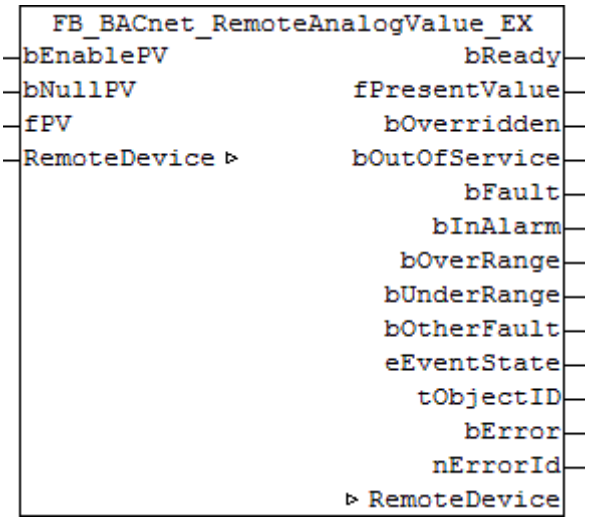
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice       : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.4 FB_BACnet_RemoteAnalogValue_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteAnalogValue_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *AnalogValue* zugegriffen werden.

VAR_INPUT

```
bEnablePV   : BOOL;
bNullPV     : BOOL;
fPV         : REAL;
```

bEnablePV: *TRUE* = Schreibfreigabe des Property-Werts; *FALSE* = Eintrag im BACnet Objekt nicht verändern (**bNullPV** und **fPV** werden unwirksam).

bNullPV: *TRUE* = Eintrag der Property löschen (*NULL*); anstelle des Werts von **fPV**; *FALSE* = Wert von **fPV** als Property Wert schreiben.

fPV: Wert der in die Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** = *TRUE* und **bNullPV** = *FALSE* sind. Das Schreiben der Prozessdaten erfolgt bei Änderung.

VAR_OUPUT

```
bReady          : BOOL;
fPresentValue   : REAL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bOverRange      : BOOL;
bUnderRange     : BOOL;
bOtherFault     : BOOL;
eEventState     : E_BACNETEVENTSTATE;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Status_Flags*.

bOverRange, bUnderRange, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property Event_State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Acked_Transitions*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

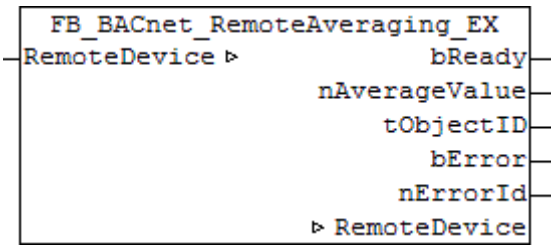
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice        : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.5 FB_BACnet_RemoteAveraging_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteAveraging_EX kann lesend auf ein BACnet-Objekt vom Typ *Averaging* zugegriffen werden.

VAR_OUPUT

```
bReady          : BOOL;
nAverageValue   : REAL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fAverageValue: Aktueller Mittelwert (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Averaging* und Property *Average_Value*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

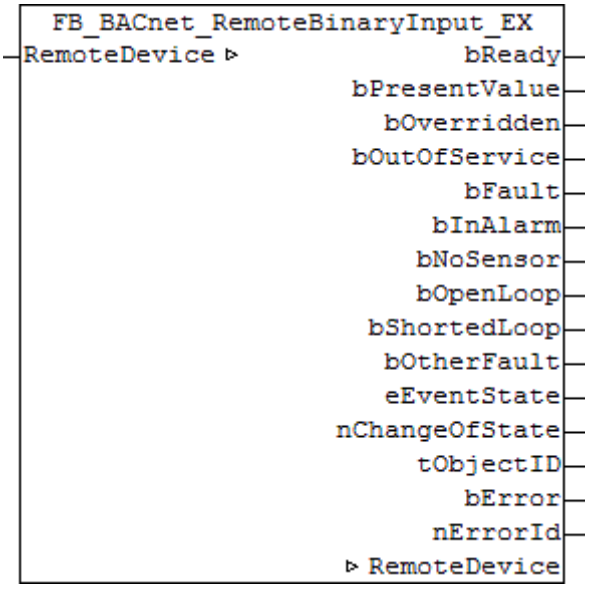
nErrorId: siehe globale Konstanten BACnet Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice        : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.6 FB_BACnet_RemoteBinaryInput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteBinaryInput_EX kann lesend auf ein BACnet-Objekt vom Typ *BinaryInput* zugegriffen werden.

VAR_OUPUT

```
bReady          : BOOL;
bPresentValue   : BOOL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bNoSensor       : BOOL;
bOpenLoop       : BOOL;
bShortedLoop    : BOOL;
bOtherFault     : BOOL;
eEventState     : E_BACNETEVENTSTATE;
nChangeOfState  : UDINT;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Status_Flags*.

bNoSensor, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property Event State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Acked_Transitions*).

nChangeOfState: Anzahl der Zustandsänderungen (*nicht* gleichzusetzen mit Wertänderung) der Property *Present_Value* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Change_Of_State_Count*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

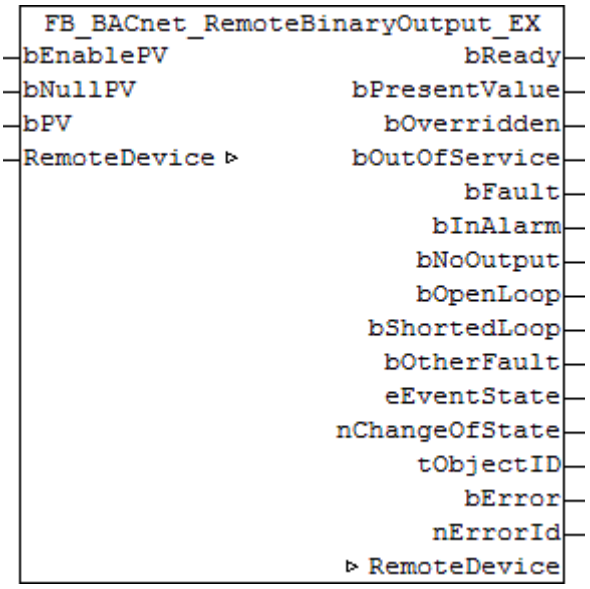
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice    : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.7 FB_BACnet_RemoteBinaryOutput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteBinaryOutput_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *BinaryOutput* zugegriffen werden.

VAR_INPUT

```
bEnablePV    : BOOL;
bNullPV      : BOOL;
bPV          : BOOL;
```

bEnablePV: *TRUE* = Schreibfreigabe des Property-Werts; *FALSE* = Eintrag im BACnet Objekt nicht verändern (**bNullPV** und **fPV** werden unwirksam).

bNullPV: *TRUE* = Eintrag der Property löschen (*NULL*); anstelle des Werts von **bPV**; *FALSE* = Wert von **bPV** als Property Wert schreiben.

b PV: Wert der in die Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** = *TRUE* und **bNullPV** = *FALSE* sind. Das Schreiben der Prozessdaten erfolgt bei Änderung.

VAR_OUPUT

```
bReady          : BOOL;
bPresentValue   : BOOL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
```

```
bFault          : BOOL;
bInAlarm        : BOOL;
bNoOutput       : BOOL;
bOpenLoop       : BOOL;
bShortedLoop    : BOOL;
bOtherFault     : BOOL;
eEventState     : E_BACNETEVENTSTATE;
nChangeOfState  : UDINT;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Status_Flags*.

bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property Event State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Acked_Transitions*).

nActivePrio: Gibt die aktuell wirksame Priorität des Priority-Array an, die auf das *Present_Value* wirkt (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

nChangeOfState: Anzahl der Zustandsänderungen (*nicht* gleichzusetzen mit Wertänderung) der Property *Present_Value* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Change_Of_State_Count*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

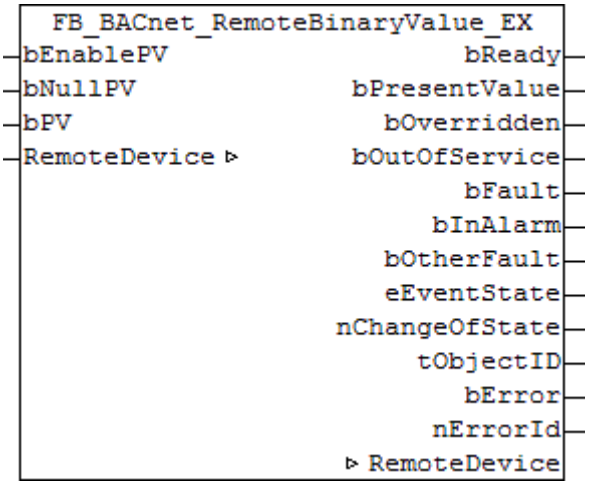
nErrorId: siehe globale Konstanten BACnet Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice    : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.8 FB_BACnet_RemoteBinaryValue_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteBinaryValue_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *BinaryValue* zugegriffen werden.

VAR_INPUT

```
bEnablePV    : BOOL;
bNullPV      : BOOL;
bPV          : BOOL;
```

bEnablePV: *TRUE* = Schreibfreigabe des Property-Werts; *FALSE* = Eintrag im BACnet Objekt nicht verändern (**bNullPV** und **fPV** werden unwirksam).

bNullPV: *TRUE* = Eintrag der Property löschen (*NULL*); anstelle des Werts von **bPV**; *FALSE* = Wert von **bPV** als Property Wert schreiben.

b PV: Wert der in die Property *Present_Value* geschrieben werden soll, wenn **bEnablePV** = *TRUE* und **bNullPV** = *FALSE* sind. Das Schreiben der Prozessdaten erfolgt bei Änderung.

VAR_OUPUT

```
bReady           : BOOL;
bPresentValue    : BOOL;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bOtherFault      : BOOL;
eEventState      : E_BACNETEVENTSTATE;
nChangeOfState   : UDINT;
tObjectID        : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Status_Flags*.

bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property Event_State [▶ 339].

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Acked_Transitions*).

nActivePrio: Gibt die aktuell wirksame Priorität des Priority-Array an, die auf das *Present_Value* wirkt (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Present_Value*).

nChangeOfState: Anzahl der Zustandsänderungen (*nicht* gleichzusetzen mit Wertänderung) der Property *Present_Value* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Change_Of_State_Count*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

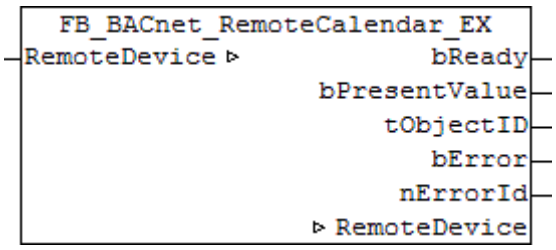
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice    : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.9 FB_BACnet_RemoteCalendar_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteCalendar_EX kann lesend auf ein BACnet-Objekt vom Typ *Calendar* zugegriffen werden.

VAR_OUPUT

```
bReady          : BOOL;
bPresentValue   : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Calendar* und Property *Present_Value*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

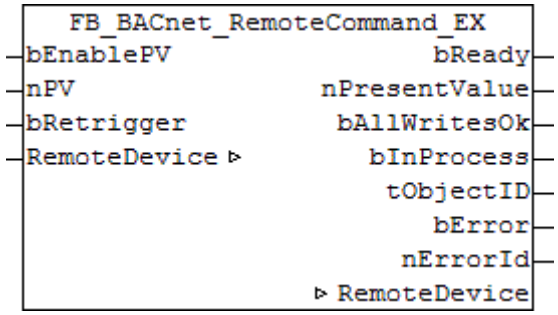
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice       : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.10 FB_BACnet_RemoteCommand_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteCommand_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Command* zugegriffen werden.

VAR_INPUT

```
bEnablePV    : BOOL;
nPV          : UDINT;
bRetrigger   : BOOL;
```

bEnablePV: Gibt den Wert des Eingangs **nPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt das Schreiben in die Property *Present_Value* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs **nPV**.
Wird **bEnablePV** auf *FALSE* gesetzt, dann werden die Prozessdaten der gemappten Property *Present_Value* auf *0* geschrieben und damit deaktiviert.

nPV: Wert der Property *Present_Value* der geschrieben werden soll. Liegt der Wert außerhalb des Wertebereicht (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *Present_Value*), dann wird das zugehörige Prozessdatum deaktiviert, d.h. das Schreiben auf die Property ist deaktiviert.
Das Schreiben eines gültigen Wertes löst die Ausführung der zugehörigen Kommando-Liste des BACnet-Objekts aus. Geschrieben wird der Wert der Property *Present_Value* immer dann, wenn eine Änderung des Prozessdatums erfolgt (d.h. Änderung des Wertes von **nPV** bei gesetztem **bEnablePV** oder Signalswechsel *FALSE* --> *TRUE* am Eingang **bRetrigger** bei gesetztem **bEnablePV**).

bRetrigger: Bei steigender Flanke an diesem Eingang wird das Schreiben und damit die Ausführung des entsprechenden Kommandos des BACnet-Objekts *Command* wiederholt. Der Signalwechsel von *FALSE* --> *TRUE* entspricht einer Änderung des Prozessdatums der Property *Present_Value* von x → *0* → x.

VAR_OUPUT

Variablen sind nicht in der Basisversion des Bausteins enthalten.

```
bReady          : BOOL;
nPresentValue   : UDINT;
bAllWritesOk    : BOOL;
bInProcess      : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *Present_Value*).

bAllWritesOk: Die zuletzt angeforderte Kommando-Liste wurde erfolgreich abgearbeitet (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *All_Writes_Successful*).

bInProcess: Die selektierte Kommando-Liste wird abgearbeitet (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *In_Process*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

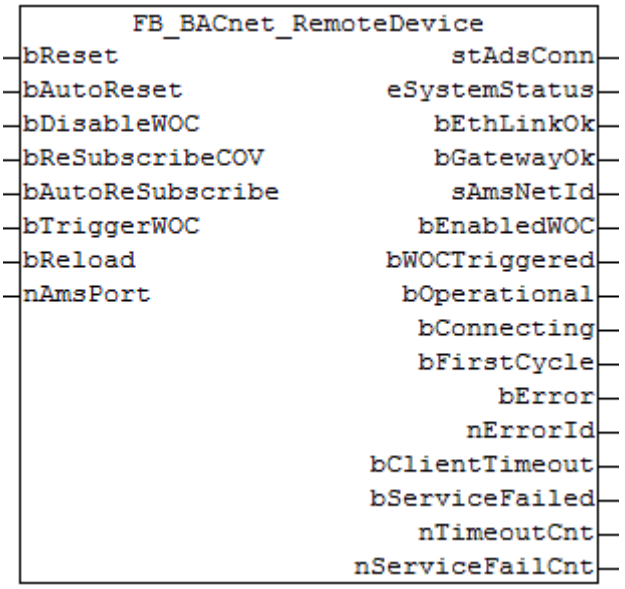
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice        : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.11 FB_BACnet_RemoteDevice



Anwendung

Funktionsbaustein für die Anbindung des PLC Programmes an ein entferntes BACnet Device Objekt (Client). Die Verknüpfung des Funktionsbausteins erfolgt mit Hilfe von Prozessdaten und ADS (AMS Port evaluieren, Objekt Liste lesen und ermitteln der Stack Revision).

Eine Instanz des Bausteins FB_BACnet_RemoteDevice ist immer dann nötig, wenn auf Objekte eines entfernten BACnet-Servers mit Hilfe eines lokalen BACnet-Clients aus der SPS zugegriffen werden soll. Mehrere Instanzen des Bausteins FB_BACnet_RemoteDevice können angelegt und mittels SPS-Automapping zu den entsprechenden Clients im TwinCAT System Manager verbunden werden.

Grundsätzlich ist der Funktionsbaustein abwärtskompatibel (Revision 6). Bei Verwendung von Revision 6 gibt es jedoch Einschränkungen bei einigen ADS Diensten des BACnet Stacks; z.B. Der AMS Port des BACnet Device (Server und Client) kann erst ab Revision 12 automatisch ermittelt werden. Der im TwinCAT System Manager ersichtliche AMS Port (Ads Port) muss daher, bei Verwendung von Revision 6, am Eingang **nAmsPort** übergeben werden (siehe Bild-1)!

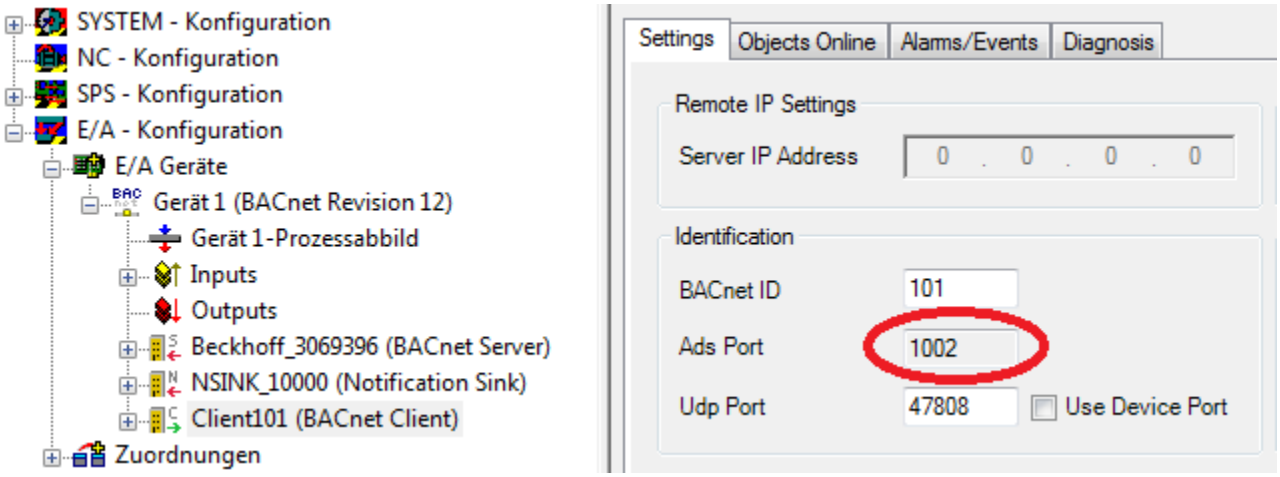


Abb. 18: Bild-1: AMS Port des BACnet Client im TwinCAT System Manager

VAR_INPUT

Variablen sind optional und müssen bei Verwendung von Revision 12 oder höher nicht belegt werden.

```
bReset            : BOOL;
bAutoReset        : BOOL;
bDisableWOC       : BOOL;
bReSubscribeCOV   : BOOL;
bAutoReSubscribe  : BOOL;
bTriggerWOC       : BOOL;
bReload           : BOOL;
nAmsPort          : UINT:=0;   (*siehe Info*)
```

bReset: Zurücksetzen des Fehlerzustands bei Signalwechsel *FALSE* --> *TRUE*.

bAutoReset: Bei Setzen des Eingangs auf *TRUE* werden Fehler mit ID 22 und 23 automatisch quittiert. Verbundene Remote-FBs werden nicht blockiert (**bReady** an den Objekt-FBs bleibt TRUE). Statt des Fehlers wird ein Fehlerzähler hoch gezählt (siehe **nTimeoutCnt** und **nServiceFailCnt**).

bDisableWOC: Bei Setzen des Eingangs auf *TRUE* wird WriteOnChange (WOC, Schreiben-bei-Änderung) für sämtliche Objekte dieses Clients unterdrückt. Das bedeutet, dass Änderungen der Prozessdaten kein automatisches Schreiben der Properties an die entfernten Objekte auslöst (gilt nur für Properties die für WriteOnChange unter dem Client konfiguriert sind).

bReSubscribeCOV: Bei Signalwechsel des Eingangs von *FALSE* --> *TRUE* wird das Neu-Abonnieren sämtlicher Properties die für COV konfiguriert sind ausgelöst. Diese Funktion betrifft unter Umständen potenziell viele Objekte und sollte daher nur bei Bedarf ausgeführt werden. Im Normalfall werden alle abonnierten Properties nach Ablauf einer Timeoutzeit automatisch erneut abonniert (Resubscription-Intervall).

bAutoReSubscribe: Bei Setzen des Eingangs auf *TRUE* wird das automatische Neu-Abonnieren sämtlicher für COV konfigurierter Properties bei Bedarf ausgelöst. Dies ist nicht mit dem Resubscription-Intervall zu verwechseln! Ein automatisches Neu-Abonnieren wird ausgelöst, wenn z.B. ein entfernter Server nicht mehr erreichbar war und die Verbindung wieder hergestellt werden konnte. Wird der Eingang auf *FALSE* gesetzt, dann wird ein erneutes Abonnieren erst nach Ablauf des Resubscription-Intervalls ausgelöst.

bTriggerWOC: ab Revision 12: Wechsel von *FALSE* → *TRUE* schreibt sämtliche Prozessdaten erneut.

bReload: Die ADS Verbindung wird erneuert. Nachfolgenden Bausteine, die die ADS-Verbindung nutzen, werden ebenfalls getriggert.

nAmsPort: *Nur bei Verwendung von Revision 6, sonst nicht verwenden:* AMS Port des lokalen BACnet Client. Im Standardfall ist dieser nicht definiert und muss im TwinCAT System Manager abgelesen werden (siehe Bild-1).

VAR_OUPUT

```
stAdsConn          : ST_BACnet_AdsConnection;
eSystemStatus      : E_BACNETDEVICESTATUS;
bEthLinkOk         : BOOL;
bGatewayOk         : BOOL;
sAmsNetId          : T_AmsNetId;
bEnabledWOC        : BOOL;
bWOCTriggered      : BOOL;
bOperational       : BOOL;
bConnecting        : BOOL;
bFirstCycle        : BOOL;
bError             : BOOL;
nErrorId           : UINT;
bClientTimeout     : BOOL;
bServiceFailed     : BOOL;
nTimeoutCnt        : UDINT;
nServiceFailCnt    : UDINT;
```

eSystemStatus: Aktueller Status des entfernten BACnet Server Objekts (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet Device Objekt und Property *System_Status*).

bEthLinkOk: Die lokale Ethernet-Verbindung ist aktiv (Kabel gesteckt, Adapter verbunden), wenn der Ausgang auf *TRUE* gesetzt ist.

bGatewayOk: Das konfigurierte Gateway ist erreichbar, wenn der Ausgang auf *TRUE* gesetzt ist.

sAmsNetId: Ausgabe der AMS-NetID des lokalen BACnet-Adapters (kann für den asynchronen Zugriff via ADS auf BACnet-Objekte verwendet werden).

bEnabledWOC: WriteOnChange (WOC, Schreiben-bei-Änderung) ist für COV konfigurierte Properties aktiviert.

bOperational: BACnet-Device Objekt des entfernten Servers meldet "Operational" (System_Status = 0), der Device-Adapter meldet den Status **bEthLinkOk**, Status **bConnecting** ist *FALSE* und das Bit 0 des Prozessdatums "Client Status" ist nicht gesetzt (Timeout-Fehler [▶ 330]). Fällt der Ausgang auf *FALSE* werden sämtliche verbundene Objekte (Bausteininstanzen) gesperrt.

bConnecting: Die Verbindung zum entfernten Server wird aufgebaut (Prozessdatum "Client Status" ist kleiner 0x04xx).

bFirstCycle: Wird mit dem ersten Aufruf der Bausteininstanz nach SPS-Reset bzw. -Neustart für einen Zyklus gesetzt.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

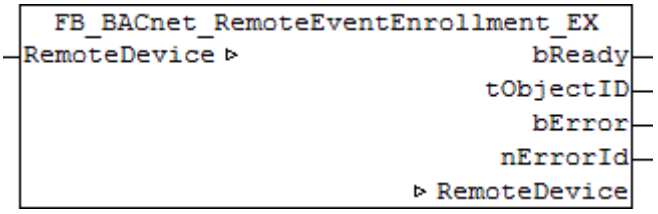
bClientTimeout: Bei gesetztem Ausgang liegt ein Timeout-Fehler [▶ 330] an. Der Ausgang wird erst zurück gesetzt, wenn 10 Sekunden lang kein Timeout Fehler mehr aufgetreten ist.

bServiceFailed: Bei gesetztem Ausgang liegt ein Service-Fehler [▶ 330] an. Der Ausgang wird erst zurück gesetzt, wenn 10 Sekunden lang kein Fehler mehr aufgetreten ist.

nTimeoutCnt: Anzahl aufgetretener Timeout-Fehler [▶ 330]. Es wird die positive Flanke des entsprechenden Status-Flag des Clients gezählt.

nServiceFailCnt: Anzahl aufgetretener Service-Fehler [▶ 330]. Es wird die positive Flanke des entsprechenden Status-Flag des Clients gezählt.

4.3.12 FB_BACnet_RemoteEventEnrollment_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteEventEnrollment_EX kann lesend auf ein BACnet-Objekt vom Typ *EventEnrollment* zugegriffen werden.

VAR_OUPUT

Variablen sind nicht in der Basisversion des Bausteins enthalten.

```
bReady          : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;  (*siehe Info*)
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

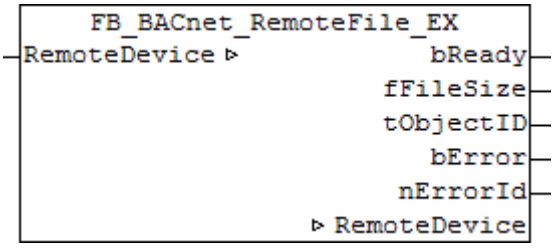
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice       : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.13 FB_BACnet_RemoteFile_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteFile_EX kann lesend auf ein BACnet-Objekt vom Typ *File* zugegriffen werden.

VAR_OUPUT

```
bReady          : BOOL;
fFileSize       : UDINT;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fFileSize: Aktuelle Dateigröße in Byte (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *File* und Property *File_Size*).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

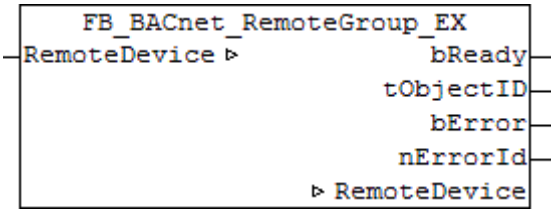
nErrorId: siehe globale Konstanten BACnet Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice        : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.14 FB_BACnet_RemoteGroup_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteGroup_EX kann lesend auf ein BACnet-Objekt vom Typ *Group* zugegriffen werden.

VAR_OUPUT

```
bReady          : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

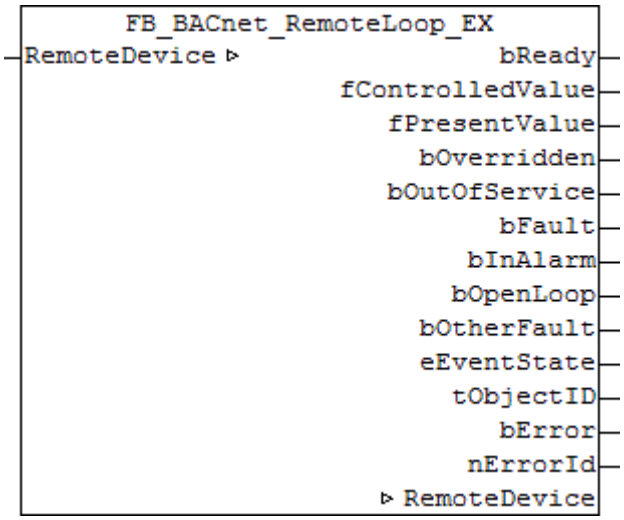
nErrorId: siehe globale Konstanten BACnet Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice        : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.15 FB_BACnet_RemoteLoop_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteLoop_EX kann lesend auf ein BACnet-Objekt vom Typ *Loop* zugegriffen werden.

VAR_OUPUT

```
bReady            : BOOL;
fControlledValue  : REAL;
fPresentValue     : REAL;
bOverridden       : BOOL;
bOutOfService     : BOOL;
bFault            : BOOL;
bInAlarm          : BOOL;
bOpenLoop         : BOOL;  (*siehe Beschreibung*)
bOtherFault       : BOOL;  (*siehe Beschreibung*)
eEventState       : E_BACNETEVENTSTATE;  (*siehe Beschreibung*)
tObjectID         : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;  (*siehe Beschreibung*)
bError            : BOOL;
nErrorId          : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fControlledValue: Rückmeldung der aktuellen Prozessgröße (X, Istwert).

fPresentValue: Rückmeldung der aktuellen Regelausgabe (Y, Stellwert). Achtung: *Present_Value* und *Controlled_Variable_Value* können schnell zu Verwechslungen führen (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Properties *Present_Value*, *Controlled_Variable_Value* und *Controlled_Variable_Reference*)!

fPropBand: Rückmeldung der aktuellen Regelausgabe in Prozent (-100%...+100%) in Relation zur minimalen und maximalen Regelausgabe (Properties *Minimum_Output* und *Maximum_Output*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Status_Flags*.

bOpenLoop, bCommFailure, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Reliability*.

eEventState: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property Event_State [▶ 339].

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

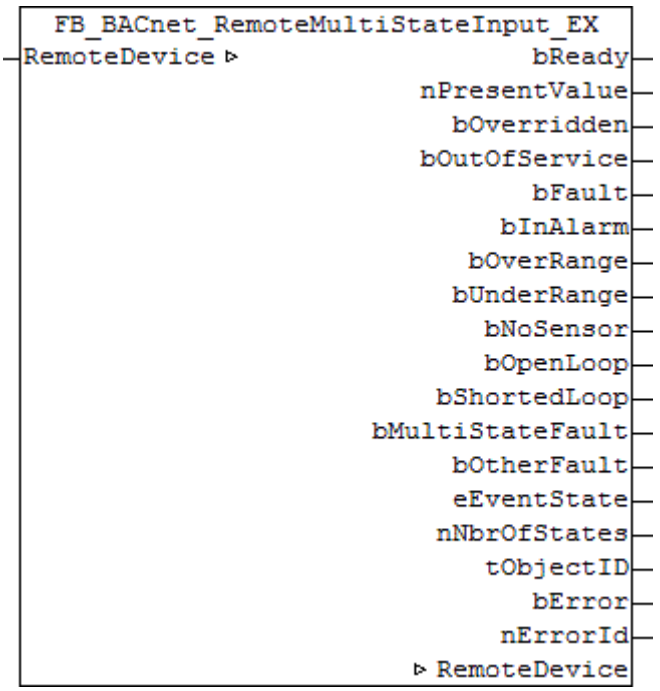
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice      : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.16 FB_BACnet_RemoteMultiStateInput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteMultiStateInput_EX kann lesend auf ein BACnet-Objekt vom Typ *MultiStateInput* zugegriffen werden.

VAR_OUPUT

```
bReady            : BOOL;
nPresentValue     : UDINT;
bOverridden       : BOOL;
bOutOfService     : BOOL;
bFault            : BOOL;
bInAlarm          : BOOL;
bOverRange        : BOOL;
bUnderRange       : BOOL;
bNoSensor         : BOOL;
bOpenLoop         : BOOL;
bShortedLoop      : BOOL;
bMultiStateFault  : BOOL;
bOtherFault       : BOOL;
eEventState       : E_BACNETEVENTSTATE;
nNbrOfStates      : UDINT;
```

```
tObjectID          : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError             : BOOL;
nErrorId           : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Status_Flags*.

bOverRange, bUnderRange, bNoSensor, bOpenLoop, bShortedLoop, bMultiStateFault, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Reliability*.

eEventState: E_BACNETEVENTSTATE [▶ 339], siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Event_State*.

nNbrOfStates: Meldet die verfügbare Anzahl von Werten, die das *Present_Value* des MultiStateInput-Objekts annehmen kann (1...*nNbrOfStates*). Ist "nNbrOfStates" gleich 0, dann gibt es keinen gültigen "State" den das Objekt annehmen kann (*Present_Value* ist 0).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

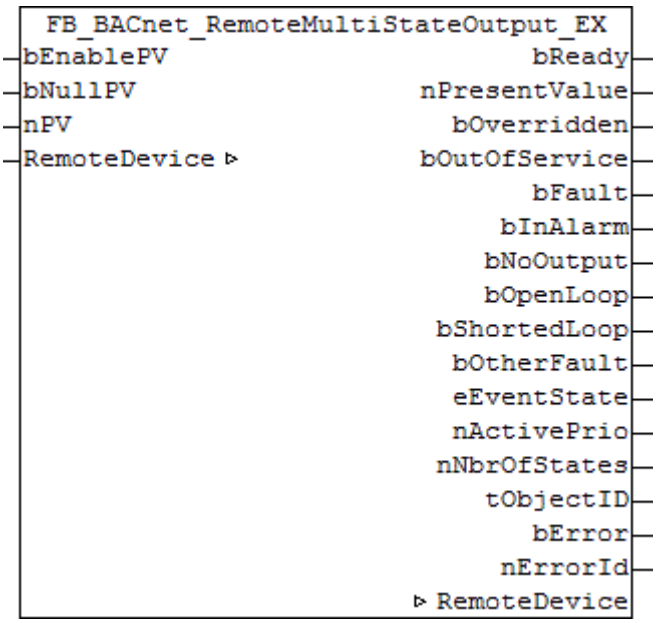
nErrorId: siehe globale Konstanten BACnet Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice       : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.17 FB_BACnet_RemoteMultiStateOutput_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteMultiStateOutput_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *MultiStateOutput* zugegriffen werden.

VAR_INPUT

```
bEnablePV       : BOOL;
bNullPV         : BOOL;
nPV             : UDINT;
```

bEnable: *TRUE* = Das Prozessdatum wird aktiviert; der Wert, der sich aus **bNull** bzw. **nState** ergibt, wird in das entsprechende BACnet Object geschrieben, *FALSE* = Prozessdatum wird deaktiviert

bNull: *TRUE* = Null-Schreiben des BACnet Objekts (z.B. Löschen einer Priorität), *FALSE* = Wert aus **nState** schreiben

nPV: Wert der in das BACnet Object geschrieben wird, wenn **bEnable** = *TRUE* und **bNull** = *FALSE* sind. Multi-State im Bereich [1 .. *Number_Of_States*].

VAR_OUPUT

```
bReady          : BOOL;
nPresentValue   : UDINT;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bNoOutput       : BOOL;
bOpenLoop       : BOOL;
bShortedLoop    : BOOL;
bOtherFault     : BOOL;
eEventState     : E_BACNETEVENTSTATE;
nActivePrio     : UINT;
nNbrOfStates    : UDINT;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Status_Flags*.

bNoOutput, bOpenLoop, bShortedLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Reliability*.

eEventState: E_BACNETEVENTSTATE [▶ 339], siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Event_State*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Acked_Transitions*).

nActivePrio: Gibt die aktuell wirksame Priorität des Priority-Array an, die auf das *Present_Value* wirkt (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Present_Value*).

nNbrOfStates: Meldet die verfügbare Anzahl von Werten, die das *Present_Value* des MultiStateOutput-Objekts annehmen kann (1...*nNbrOfStates*). Ist **nNbrOfStates** gleich 0, dann gibt es keinen gültigen "State" den das Objekt annehmen kann (*Present_Value* ist 0).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

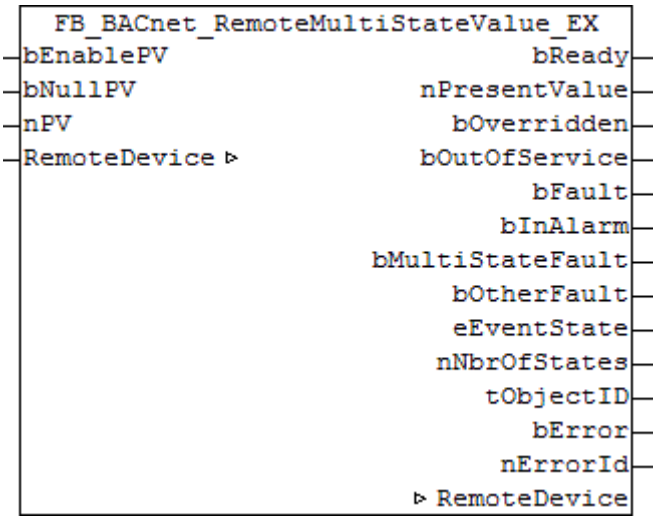
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice       : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.18 FB_BACnet_RemoteMultiStateValue_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteMultiStateValue_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *MultiStateValue* zugegriffen werden.

VAR_INPUT

```
bEnablePV      : BOOL;
bNullPV        : BOOL;
nPV            : UDINT;
```

bEnable: *TRUE* = Das Prozessdatum wird aktiviert; der Wert, der sich aus **bNull** bzw. **nState** ergibt, wird in das entsprechende BACnet Object geschrieben, *FALSE* = Prozessdatum wird deaktiviert

bNull: *TRUE* = Null-Schreiben des BACnet Objekts (z.B. Löschen einer Priorität), *FALSE* = Wert aus **nState** schreiben

nState: Wert der in das BACnet Object geschreiben wird, wenn **bEnable** = *TRUE* und **bNull** = *FALSE* sind. Multi-State im Bereich [1 .. *Max_Number_Of_States*].

VAR_OUPUT

```
bReady             : BOOL;
nPresentValue      : UDINT;
bOverridden        : BOOL;
bOutOfService      : BOOL;
bFault             : BOOL;
bInAlarm           : BOOL;
bMultiStateFault   : BOOL;
bOtherFault        : BOOL;
eEventState        : E_BACNETEVENTSTATE;
nNbrOfStates       : UDINT;
tObjectID          : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError             : BOOL;
nErrorId           : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Status_Flags*.

bMultiStateFault, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Reliability*.

eEventState: E_BACNETEVENTSTATE [▶ 339], siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Event_State*.

nNbrOfStates: Meldet die verfügbare Anzahl von Werten, die das *Present_Value* des MultiStateValue-Objekts annehmen kann (1...*nNbrOfStates*). Ist "nNbrOfStates" gleich 0, dann gibt es keinen gültigen "State" den das Objekt annehmen kann (*Present_Value* ist 0).

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

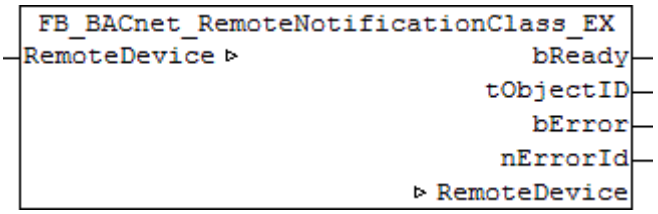
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice      : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.19 FB_BACnet_RemoteNotificationClass_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteNotificationClass_EX kann lesend auf ein BACnet-Objekt vom Typ *NotificationClass* zugegriffen werden.

VAR_OUPUT

```
bReady        : BOOL;
tObjectID     : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError        : BOOL;
nErrorId      : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

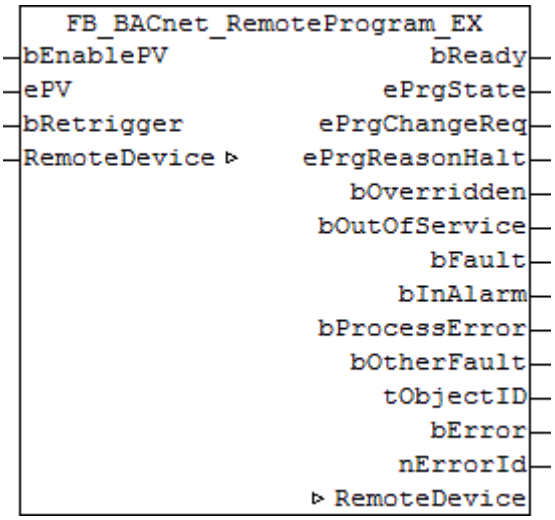
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice      : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.20 FB_BACnet_RemoteProgram_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteProgram_EX kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Program* zugegriffen werden.

VAR_INPUT

```
bEnablePV    : BOOL;
ePV          : E_BACNETPROGRAMREQUEST;
bRetrigger   : BOOL;
```

bEnablePV: Gibt den Wert des Eingangs **ePV** frei. Sobald der Eingang auf *TRUE* gesetzt wird, wird der Wert von **ePV** in die Property *Program_Change* des Objekts geschrieben. Wird der Eingang auf *FALSE* gesetzt, dann wird *0xFFFF* in das Prozessdatum der Property *Program_Change* des Objekts geschrieben. Der Wert *0xFFFF* verhindert das Schreiben an den entfernten Server und damit ebenfalls das Schreiben an das entfernte Objekt.

ePV: E_BACNETPROGRAMREQUEST [▶ 348], Anforderung an das entfernte Programm-Objekt. Wurde **bEnablePV** auf *TRUE* gesetzt, dann wird der Wert des Eingangs in die Property *Program_Change* geschrieben (siehe auch Transition-Diagram [▶ 256]).

bRetrigger: Ein Wechsel von *FALSE* → *TRUE* löst das erneute Schreiben des Eingangs **ePV** in die Property *Program_Change* aus, wenn Eingang **bEnablePV** auf *TRUE* gesetzt ist.

VAR_OUPUT

```
bReady           : BOOL;
ePrgState        : E_BACNETPROGRAMSTATE;
ePrgChangeReq    : E_BACNETPROGRAMREQUEST
ePrgReasonHalt   : E_BACNETPROGRAMERROR;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bProcessError    : BOOL;
bOtherFault      : BOOL;
```

```
tObjectID       : DINT:=-1;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

ePrgState: Rückmeldung des aktuellen Programmzustands (siehe auch Transition-Diagram [▶ 256]).

ePrgChangeReq: Rückmeldung über den Zustand der aktuellen Programmanforderung (siehe auch Transition-Diagram [▶ 256]).

ePrgReasonHalt: Fehlerrückmeldung bei Programmabbruch.

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Program* und Property *Status_Flags*.

bProcessError, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Program* und Property *Reliability*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice    : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

Transition-Diagram

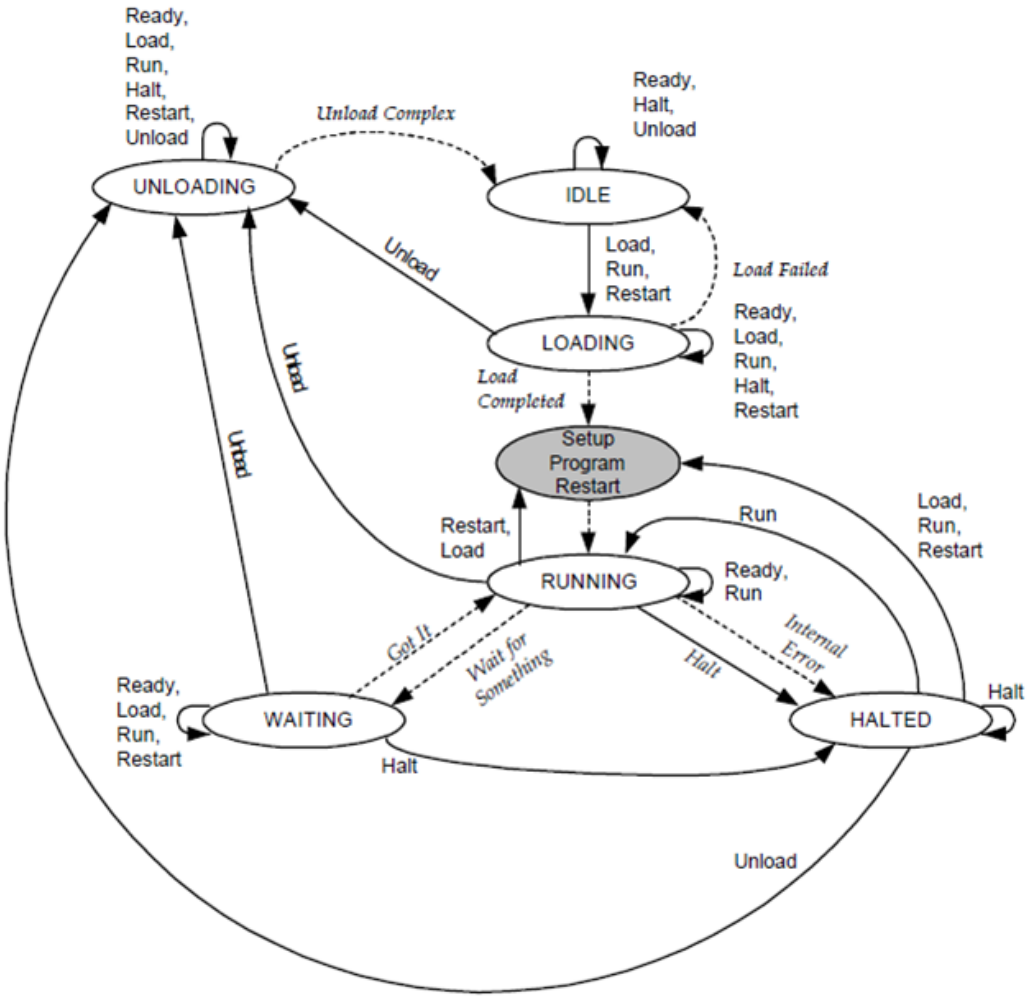
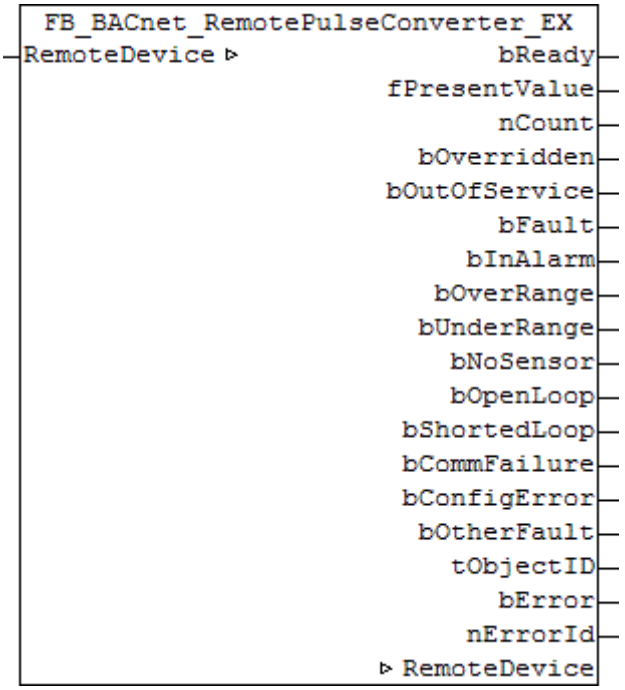


Abb. 19: Bild-1: aus BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt Program, Abbildung 12-3 "State Transitions for the program object"

4.3.21 FB_BACnet_RemotePulseConverter_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemotePulseConverter_EX kann lesend auf ein BACnet-Objekt vom Typ *PulseConverter* zugegriffen werden.

VAR_OUPUT

```
bReady          : BOOL;
fPresentValue   : REAL;
nCount          : UDINT;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bOverRange      : BOOL;
bUnderRange     : BOOL;
bNoSensor       : BOOL;
bOpenLoop       : BOOL;
bShortedLoop    : BOOL;
bCommFailure    : BOOL;
bConfigError    : BOOL;
bOtherFault     : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Present_Value*).

nCount: Wert der Property *Count*. *Count* repräsentiert die erfassten Eingangsimpulse bzw. -wertänderungen. Zudem kann der Wert von *Count* Korrekturen aus der Property *Adjust_Value* enthalten. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Count*.

bOverridden, bOutOfService, **bFault**, **bInAlarm:** Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Status_Flags*.

bOverRange, bUnderRange, **bNoSensor**, **bOpenLoop**, **bShortedLoop**, **bCommFailure**, **bConfigError**, **bOtherFault:** Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Status_Flags*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

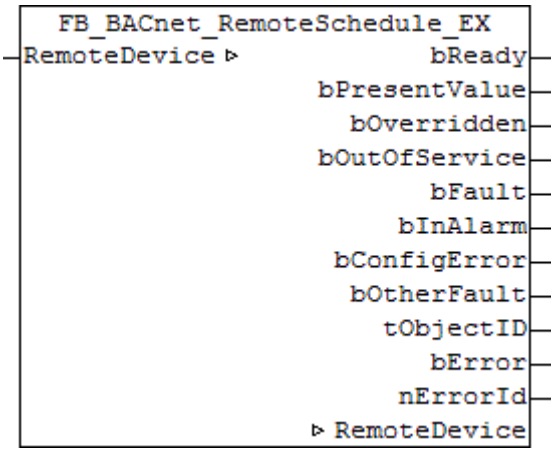
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice       : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.22 FB_BACnet_RemoteSchedule_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteSchedule_EX kann lesend auf ein BACnet-Objekt vom Typ *Schedule* zugegriffen werden.

VAR_OUPUT

```
bReady          : BOOL;
bPresentValue   : BOOL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bConfigError    : BOOL;
bOtherFault     : BOOL;
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Schedule* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Schedule* und Property *Status_Flags*.

bConfigError, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Schedule* und Property *Reliability*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

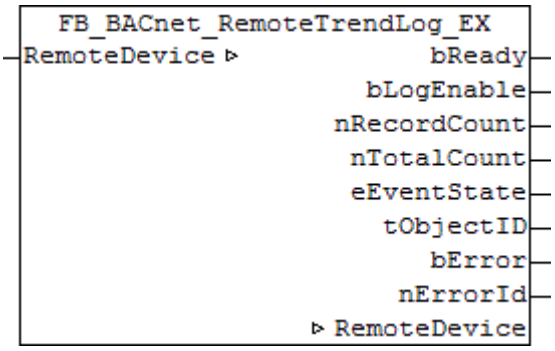
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice       : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.3.23 FB_BACnet_RemoteTrendLog_EX



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_RemoteTrendLog_EX kann lesend auf ein BACnet-Objekt vom Typ *TrendLog* zugegriffen werden.

VAR_OUPUT

```
bReady         : BOOL;
bLogEnable     : BOOL;
nRecordCount   : UDINT;
nTotalCount    : UDINT;
eEventState    : E_BACNETEVENTSTATE;
tObjectID      : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bError         : BOOL;
nErrorId       : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im TwinCAT System Manager nicht richtig verknüpft.

bLogEnable: Die Aufzeichnung von Werten ist freigegeben. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Log_Enable*.

nRecordCount: Anzahl Einträge der Property *Log_Buffer*. Wird die Property *Record_Count* mit "0" beschrieben, dann wird der *Log_Buffer* zurück gesetzt. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Record_Count*.

nTotalCount: Gesamtanzahl bisheriger Einträge der Property Log_Buffer. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Total_Record_Count*.

eEventState: E_BACNETEVENTSTATE [▶ 339], siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Event_State*.

tObjectID: Objekt ID des BACnet Objekts Objekt Type und Objekt Instanz.

bError: Ein Fehler steht an.

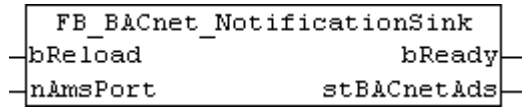
nErrorId: siehe globale Konstanten BACnet_Globals [▶ 330].

VAR_IN_OUT

```
RemoteDevice      : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Instanz des entfernten, zugehörigen BACnet-Server Bausteins (Client). Pro BACnet-Adapter sind mehrere BACnet-Clients möglich. Siehe FB_BACnet_Adapter [▶ 156] und FB_BACnet_RemoteDevice [▶ 243] für weitere Informationen.

4.4 FB_BACnet_NotificationSink



Anwendung

Funktionsbaustein zur Realisierung einer ADS Verbindung mit einer BACnet NotificationSink, die im TwinCAT System Manager angelegt wurde:

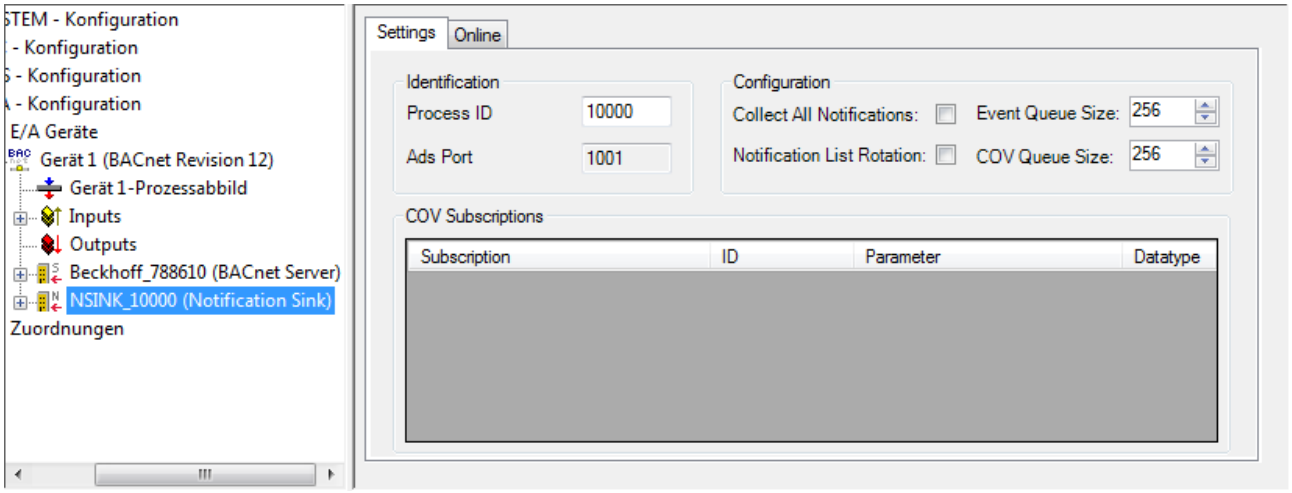


Abb. 20: Bild-1: Notification Sink im TwinCAT System Manager mit geöffnetem Settings Dialog. "Ads Port" spiegelt den AMS Port der ADS Verbindung wieder.

Folgende Bausteine stehen für den Zugriff auf die NotificationSink zur Verfügung:

Bausteine	Beschreibung
FB_BACnet_NSinkReadEvent [▶ 266]	ADS Zugriff auf die BACnet Notification Sink: Auslesen eines BACnet Events
FB_BACnet_NSinkAcknEvent [▶ 263]	ADS Zugriff auf die BACnet Notification Sink: Dienst zur Quittierung eines BACnet Events
FB_BACnet_NSinkRemoveEvent [▶ 270]	ADS Zugriff auf die BACnet Notification Sink: Löschen eines BACnet Events (ersetzt FB_BACnet_NotificationSinkDelEntry [▶ 496])

VAR_INPUT

```
bReload      : BOOL;
nAmsPort     : T_AmsPort:=0;
```

bReload: Die ADS Verbindung wird erneuert. Nachfolgenden Bausteine, die die ADS-Verbindung nutzen, werden ebenfalls getriggert.

nAmsPort: AMS Port (Ads Port) über den die NotificationSink erreichbar ist (siehe TwinCAT System Manager → Settings Dialog der entsprechenden NotificationSink; wie in Bild-1 zu sehen). Werte zwischen 1000 und 65534 sind gültig. Eine Überprüfung findet nicht statt. Ist der angegebene Port falsch oder nicht belegt, kann dies zu unerwartetem Verhalten oder ADS Fehlern in Bausteinen führen, die diese Verbindung verwenden.

VAR_OUPUT

```
bReady       : BOOL;
stBACnetAds  : ST_BACnet_AdsConnection;
```

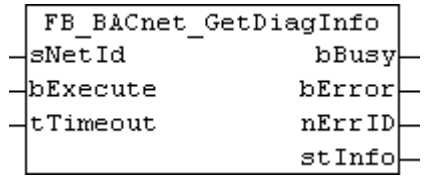
bReady: Der angegebene AMS Port liegt im gültigen Bereich (1000 ≤ **nAmsPort** < 65535).

stBACnetAds: Struktur mit den Verbindungsdaten.

4.5 BACnet ADS

4.5.1 FB_BACnet_GetDiagInfo

Funktionsbaustein für den Zugriff auf die BACnet Diagnose über ADS. Der ADS Zugriff erfolgt, im Gegensatz zu Read-/Write-Property, auf den BACnet Adapter. Der BACnet Adapter wird durch die globalen Variablen der PLC Library instanziiert und ist mit den BACnet Adapter Prozessdaten im TwinCAT System Manager verknüpft. Diagnose Daten eines entfernten BACnet Server (remote) können über dessen remote AMS NetID abgefragt werden.



Anwendung

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Eingang **sNetId** muss mit der entsprechenden AMS NetID des BACnet Adapters belegt werden. Die AMS NetID kann über die globale BACnet Adapter Instanz abgefragt werden.

Die AMS NetID entspricht nicht der lokalen AMS NetID und muss immer angegeben werden - keine Verwendung von Leerstrings möglich!

Die aktuelle AMS NetID wird vom zugehörigen FB_BACnet_Adapter [▶ 156] ausgegeben:

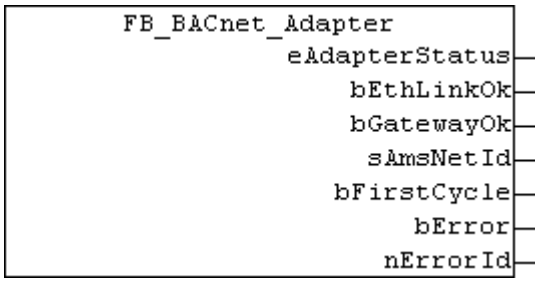


Abb. 21: **Bild-1**: FB_BACnet_Adapter [▶ 156] → **sAmsNetId** gibt die lokale AMS NetId des BACnet Adapters aus

Zudem kann die AMS NetId im TwinCAT System Manager angezeigt werden (siehe Bild-2). Der AMS Port des BACnet Adapters ist 0xFFFF (65535).

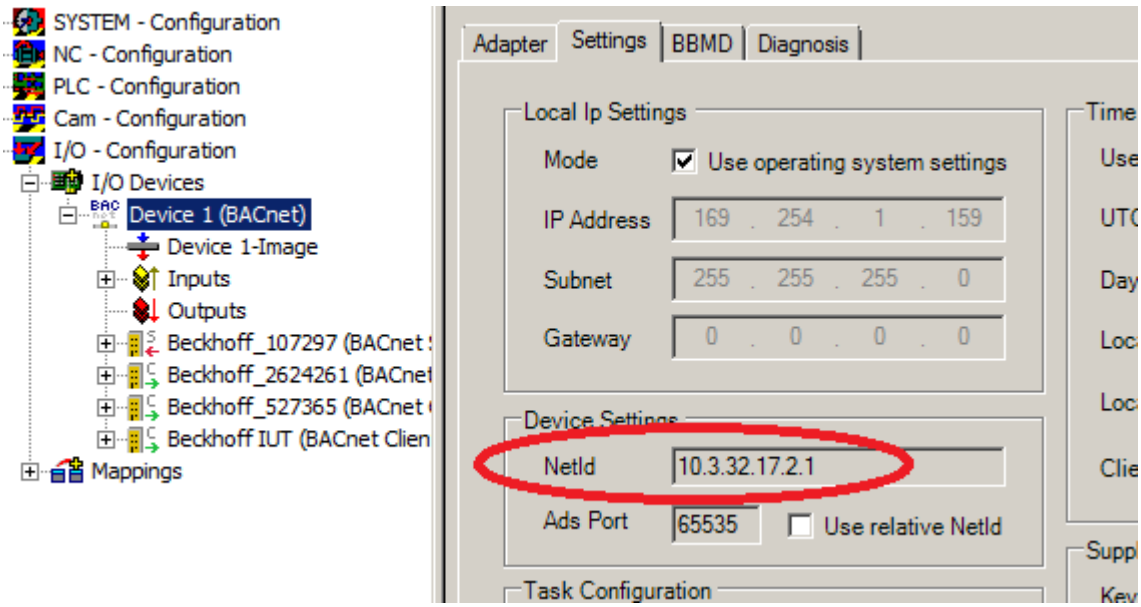


Abb. 22: **Bild-2**: AMS NetId des BACnet Adapter

VAR_INPUT

```
sNetId    : T_AmsNetId;
bExecute  : BOOL;
tTimeout  : TIME:=BACnet_ADSTimeOut;
```

sNetId: AMS NetId des BACnet Adapter.

bExecute: Steigende Flanke am Eingang startet den Lesevorgang.

tTimeout: Optionaler Eingang, Überwachungszeit für den ADS Zugriff (Default: siehe BACnet_ADSTimeOut [▶ 330]).

VAR_OUPUT

```
bBusy     : BOOL;
bError    : BOOL;
nErrID    : UDINT;
stInfo    : ST_BACnet_Diagnosis;
```

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrID: ADS Fehlercode.

stInfo: Struktur mit Diagnose-Informationen zu BACnet.

Beispiel

```
0001 PROGRAM MAIN
0002 VAR
0003     fbDiag       : FB_BACnet_GetDiagInfo;
0004     tonReadDiag : TON;
0005 END_VAR
```

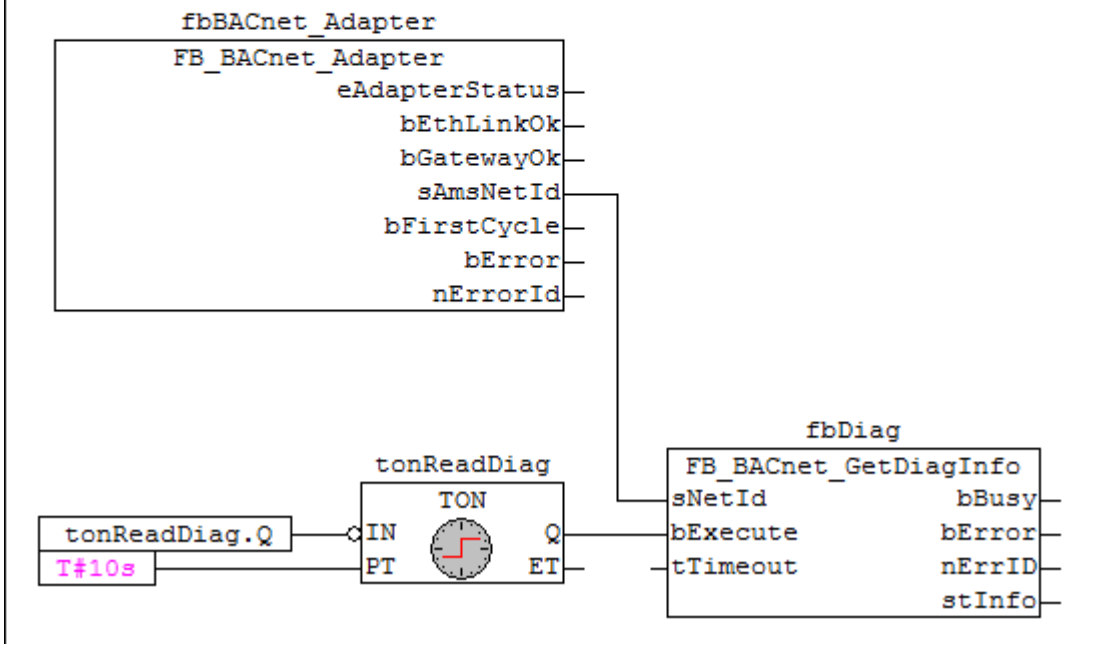
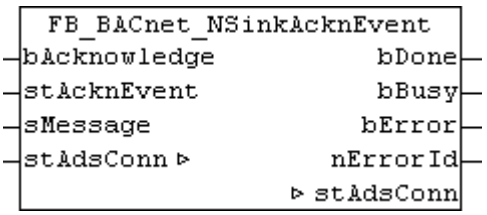


Abb. 23: **Bild-3:** Beispiel für das zyklische Lesen der Diagnose-Informationen.

Die Instanz fbBACnet_Adapter ist eine globale Variable der PLC Library vom Typ FB_BACnet_Adapter [▶ 156] und sollte nicht mehrfach instanziiert werden.

4.5.2 FB_BACnet_NSinkAcknEvent



Anwendung

Mit Hilfe des Bausteins kann ein Event aus der BACnet Notification Sink quittiert werden. Unter Beispiel wird eine mögliche Beschaltung gezeigt.

Am Eingang **stAcknEvent** wird das zu quittierende Event übergeben. Events der BACnet Notification Sink können mit Hilfe von FB_BACnet_NSinkReadEvent [▶ 266] ausgelesen werden. Dabei spielt es keine Rolle, ob das Event zum Zeitpunkt der Quittierung noch in der Notification Sink vorhanden ist oder bereits entfernt bzw. überschrieben wurde.

Bei der Quittierung eines Events wird seitens BACnet anhand der Event-Parameter eine Quittierung ausgelöst. Die Quittierung ist jedoch nur dann erfolgreich, wenn der Event-Zeitstempel der entsprechenden Transition des zu quittierenden Objekts (siehe Bild-4) mit dem Zeitstempel aus den Event-Parameter (siehe Bild-3) übereinstimmt. Die Quittierung wird mit Fehler abgebrochen, sollten die Zeitstempel nicht exakt übereinstimmen (siehe Bild-5).

Über den Eingang **sMessage** kann eine Nachricht an das Quittier-Event angefügt werden. Der Baustein erwartet ein *Windows-1252* codierten String am Eingang. Baustein intern wird daraus ein UTF-8 String codiert (siehe FB_BACnet_StringExtEncode [▶ 324]). Tritt bei der Codierung der ein Fehler auf, so wird dies am Ausgang **nErrorId** signalisiert.

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Ein-/Ausgang **stAdsConn** muss mit dem Ausgang **stBACnetAds** des entsprechenden NotificationSink-Bausteins (FB_BACnet_NotificationSink [▶ 260]) verbunden werden.

Time	Notification
12.11.2014 10:15:28	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
12.11.2014 10:15:31	Beckhoff_927609 - BI_0 : BI_0: Eingang ist in Fehler
12.11.2014 10:15:32	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
12.11.2014 10:15:34	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
12.11.2014 10:15:37	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
12.11.2014 10:15:40	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
12.11.2014 10:15:43	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
12.11.2014 10:15:46	Beckhoff_927609 - BI_0 : BI_0: Eingang ist in Fehler
12.11.2014 10:15:48	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
12.11.2014 10:15:49	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
12.11.2014 10:15:52	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
12.11.2014 10:15:55	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS

Abb. 24: Bild-1: Online Ansicht der BACnet Notification Sink.

12.11.2014 14:03:22	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
12.11.2014 14:03:36	Beckhoff_927609 - BI_0 : PLC User Reset

Abb. 25: Bild-2**:** Online-Ansicht der BACnet Notification Sink mit Quittierung als Beispiel.

```
stEvent
    .nProcessID = 10000
    .nDeviceID = 34482041
    .tObjectID = 12582912
    .stDateTime
        .stDate
            .nYear = 114
            .nMonth = 11
            .nDay = 12
            .nDayOfWeek = 3
        .stTime
            .nHour = 14
            .nMinute = 3
            .nSecond = 36
            .nHundredths = 85
    .nNC = 0
    .nPriotity = 127
    .eEventType = BACnetEventType_change_of_state
```

Abb. 26: Bild-3: Zeitstempel aus Event-Parametern in der PLC.

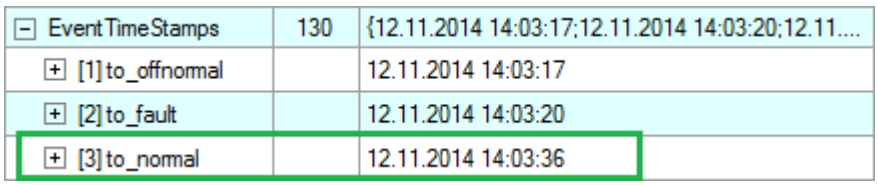


Abb. 27: Bild-4: Zeitstempel im Event-auslösenden Objekt; Property Event_Time_Stamps

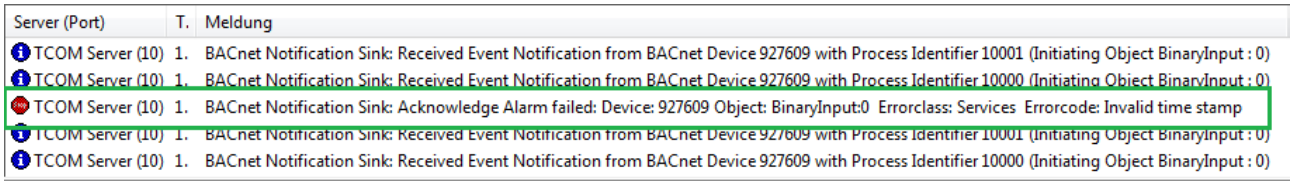


Abb. 28: Bild-5: Fehlermeldung im Logger des TwinCAT System Manager. Zeitstempel der Property Event_Time_Stamps und Zeitstempel der Event-Parameter der Quittierung stimmen nicht überein.

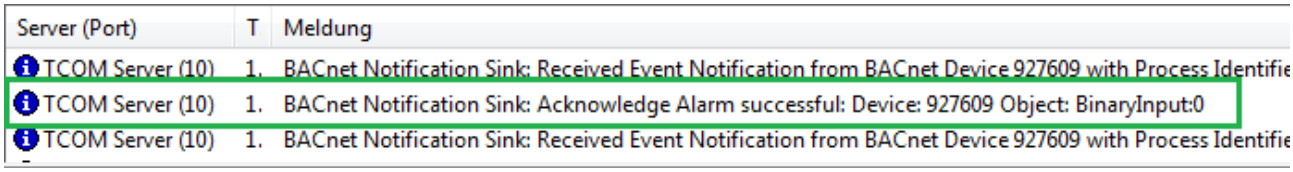


Abb. 29: Bild-6: Meldung im Logger des TwinCAT System Manager. Quittierung war erfolgreich. Im Event-generierenden Objekt (BI:0) ist das Flag der zugehörigen Transition damit zurück auf TRUE gesetzt (Property Acked_Transitions).

VAR_INPUT

```
bAcknowledge      : BOOL;
stAcknEvent       : ST_BACnet_NSinkEvent;
sMessage      : T_MaxString;
```

bAcknowledge: Flanke *FALSE → TRUE* löst das Senden der Quittierung aus.

stAcknEvent: Event-Parameter des zu quittierenden Events. Die Event-Parameter können mit Hilfe des Bausteins FB_BACnet_NSinkReadEvent [▶ 266] aus einer BACnet Notification Sink gelesen werden. Das gelesene Event muss zum Zeitpunkt der Quittierung jedoch nicht mehr zwingend in der BACnet Notification Sink gelistet sein. So können Event-Parameter in der PLC gepuffert, angezeigt und zu einem späteren Zeitpunkt quittiert werden.

sMessage: *Windows-1252* codierte Nachricht die mit der Quittierung versendet wird.

VAR_OUPUT

```
bDone         : BOOL;
bBusy         : BOOL;
bError        : BOOL;
nErrorId      : UINT;
```

bDone: Lesen der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bAcknowledge** zurückgesetzt wird. Wurde **bAcknowledge** zurück gesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrorId: Fehlercode, siehe BACnet_Globals [▶ 330] für eine Übersicht.

VAR_IN_OUT

```
stAdsConn      : ST_BACnet_AdsConnection;
```

stAdsConn: Verknüpfung mit dem Ausgang **stBACnetAds** des entsprechenden NotificationSink-Bausteins.

Beispiel: Bausteinaufruf

```
0015
0016    NSink        : FB_BACnet_NotificationSink;
0017    bAckEvent    : BOOL;
0018    stEvent      : ST_BACnet_NSinkEvent;
0019    fbAckEvent   : FB_BACnet_NSinkAcknEvent;
0020
```

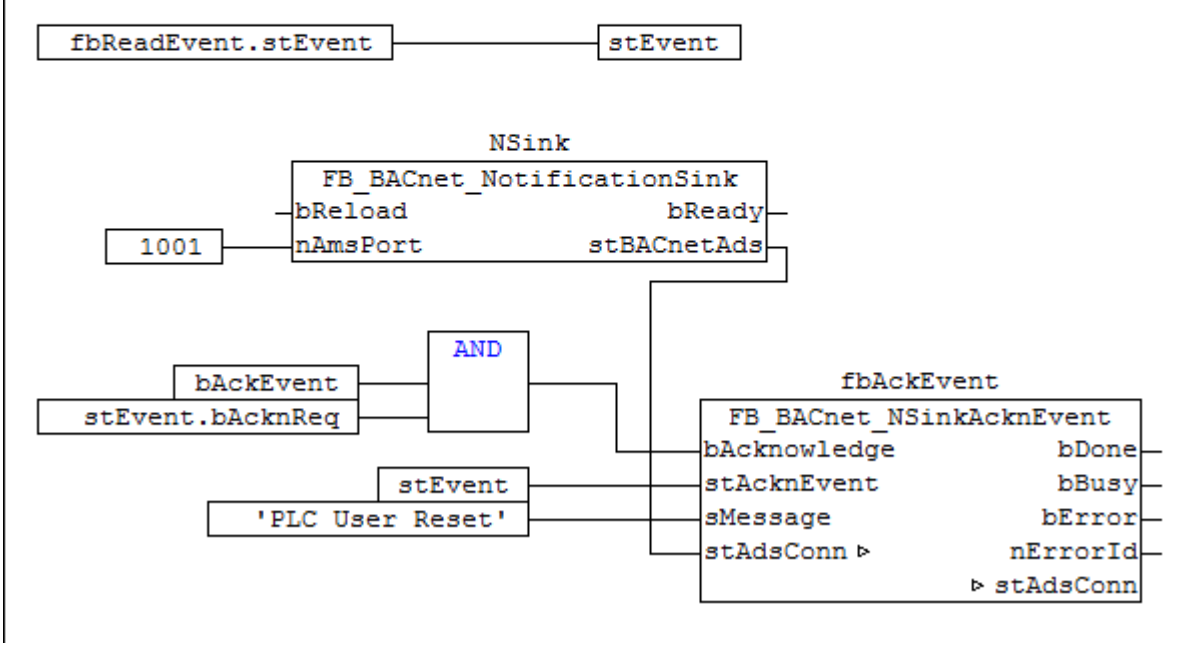
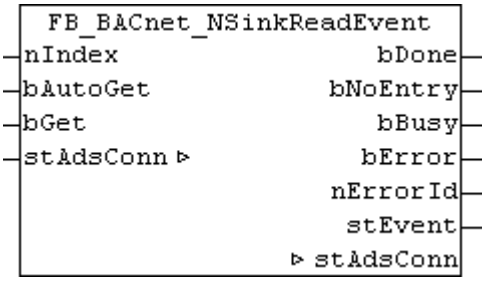


Abb. 30: Bild-7: Beispiel für die Verwendung. Quittierung eines vorab ausgelesenen Event-Eintrags

(siehe FB_BACnet_NSinkReadEvent [▶ 266]) einer BACnet Notification Sink mit AMS Port 1001.

Der AMS Port ist abhängig von der TwinCAT System Manager Konfiguration des entsprechenden BACnet Device (siehe FB_BACnet_NotificationSink [▶ 260]).

4.5.3 FB_BACnet_NSinkReadEvent



Anwendung

Mit Hilfe des Bausteins kann ein beliebiger Eintrag aus der BACnet Notification Sink ausgelesen werden. Unter Beispiel [▶ 268] wird eine mögliche Beschaltung gezeigt.

Am Eingang **nIndex** wird der zu lesende Eintrag selektiert. Die Nummerierung entspricht der Reihenfolge im Puffer der Notification Sink. Bei rotierendem Puffer beginnt die Nummierung eintreffender Events bei 0, im Falle des Pufferüberlaufs.

Baustein intern wird zudem die Decodierung des Nachrichtentextes anhand des String-Codings vorgenommen. Dabei werden folgende Encodings unterstützt: *UTF-8*, *UCS2*, *UCS4* und *ISO8859-1*. Der String **sMessage** innerhalb der Ausgabestruktur **stEvent** liegt Windows-1252 codiert vor (siehe auch FB_BACnet_StringExtDecode [▶ 323]). Tritt bei der Decodierung der ein Fehler auf, so wird dies am Ausgang **nErrorId** signalisiert.

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Ein-/Ausgang **stAdsConn** muss mit dem Ausgang **stBACnetAds** des entsprechenden NotificationSink-Bausteins (FB_BACnet_NotificationSink [▶ 260]) verbunden werden.

Time	Notification
12.11.2014 10:15:28	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
12.11.2014 10:15:31	Beckhoff_927609 - BI_0 : BI_0: Eingang ist in Fehler
12.11.2014 10:15:32	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
12.11.2014 10:15:34	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
12.11.2014 10:15:37	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
12.11.2014 10:15:40	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
12.11.2014 10:15:43	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
12.11.2014 10:15:46	Beckhoff_927609 - BI_0 : BI_0: Eingang ist in Fehler
12.11.2014 10:15:48	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
12.11.2014 10:15:49	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
12.11.2014 10:15:52	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
12.11.2014 10:15:55	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS

Abb. 31: Bild-1: Online-Ansicht der BACnet Notification Sink.

VAR_INPUT

```
nIndex      : UDINT;
bAutoGet    : BOOL:=TRUE;
bGet        : BOOL;
```

nIndex: Index des zu lesenden Event-Eintrags (0..n).

bAutoGet: *TRUE* = Lese den Eventeintrag automatisch aus, wenn die ADS Verbindung oder der Eingang **nIndex** sich geändert haben. Eine ADS Verbindungsänderung liegt vor, wenn die Verbindung nach einem Unterbruch wiederhergestellt oder die AMS NetID bzw. Port geändert wurde. Ein automatisches Auslesen geschieht nicht zyklisch und auch nicht bei Änderung des Eventeintrags selbst!

bGet: *FALSE* → *TRUE* = Eventeintrag wird unabhängig vom Eingang **bAutoGet** einmalig ausgelesen.

VAR_OUPUT

```
bDone       : BOOL;
bNoEntry    : BOOL;
bBusy       : BOOL;
bError      : BOOL;
nErrorId    : UINT;
stEvent     : ST_BACnet_NSinkEvent;
```

bDone: Lesen der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bGet** und **bAutoGet** zurückgesetzt sind oder ein erneutes Auslesen beginnt. Wurde **bGet** und **bAutoGet** zurück gesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt. **bDone** wird nur gesetzt, wenn der zu lesende Eventeintrag nicht leer ist.

bNoEntry: Lesen beendet; jedoch ohne Daten - Eventeintrag unter **nIndex** ist leer. **bNoEntry** bleibt so lange gesetzt bis ein erneutes Auslesen beginnt. Der Ausgang dient dazu das Ende der Event-Liste zu erkennen.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrorId: Fehlercode, siehe BACnet Globals [▶ 330] für eine Übersicht.

stEvent: Ausgabestruktur mit Daten eines Eventeintrags. Siehe Beispiel für eine Gegenüberstellung der PLC Daten und des Online-Eventeintrags im TwinCAT System Manager.

VAR_IN_OUT

```
stAdsConn      : ST_BACnet_AdsConnection;
```

stAdsConn: Verknüpfung mit dem Ausgang **stBACnetAds** des entsprechenden NotificationSink-Bausteins.

Beispiel 1: Gegenüberstellung PLC und System Manager

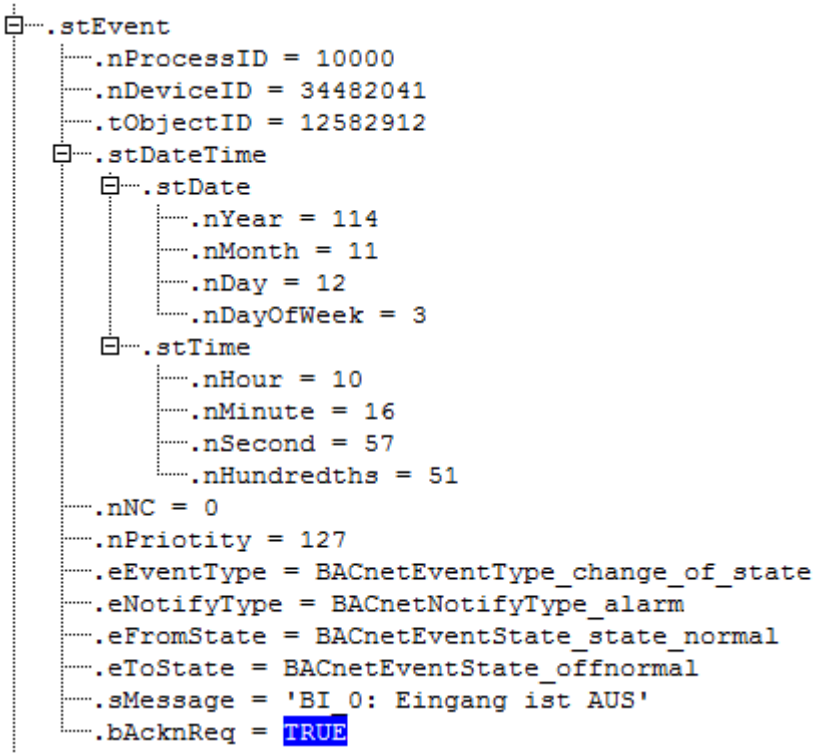


Abb. 32: Bild-2: Beispiel eines Event-Eintrag in der PLC

12.11.2014 10:16:57	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
processIdentifier	10000
initiatingDeviceIdentifier	Device:927609
eventObjectIdentifier	BinaryInput:0
timeStamp	12.11.2014 10:16:57
notificationClass	0
priority	127
eventType	change_of_state
messageText	BI_0: Eingang ist AUS
notifyType	alarm
ackRequired	☑
fromState	state_normal
toState	offnormal
eventValues	((32772;inactive))
sequenceId	1277

Abb. 33: Bild-3: Beispiel eines Event-Eintrag im TwinCAT System Manager

Beispiel 2: Bausteinaufruf

```
0001 PROGRAM MAIN
0002 VAR
0003     bReadRel     : BOOL := TRUE;
0004     NSink        : FB_BACnet_NotificationSink;
0005     fbReadEvent  : FB_BACnet_NSinkReadEvent;
0006     tonReadEvent: TON;
```

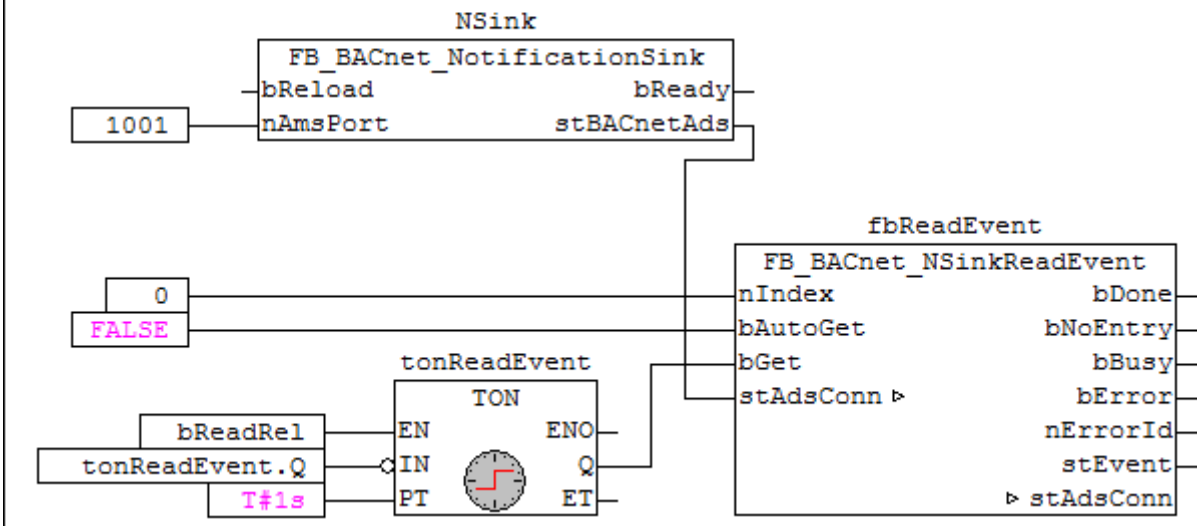
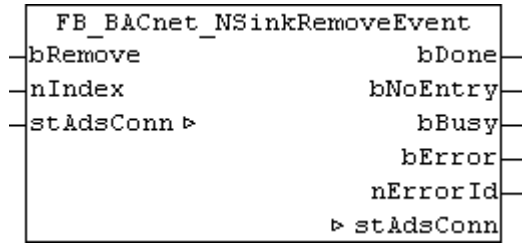


Abb. 34: Bild-4: Beispiel für die Verwendung. Zyklisches Lesen des ersten Event-Eintrags einer BACnet Notification Sink mit AMS Port 1001.

Der AMS Port ist abhängig von der TwinCAT System Manager Konfiguration des entsprechenden BACnet Device (siehe FB_BACnet_NotificationSink [▶ 260]).

4.5.4 FB_BACnet_NSinkRemoveEvent



Anwendung

Mit Hilfe des Bausteins kann ein beliebiger Eintrag aus der BACnet Notification Sink gelöscht werden. Unter Beispiel wird eine mögliche Beschaltung gezeigt.

Am Eingang **nIndex** wird der zu löschende Eintrag selektiert. Die Nummerierung entspricht der Reihenfolge im Puffer der Notification Sink. Bei rotierendem Puffer beginnt die Nummierung eintreffender Events bei 0, im Falle des Pufferüberlaufs.

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Ein-/Ausgang **stAdsConn** muss mit dem Ausgang **stBACnetAds** des entsprechenden NotificationSink-Bausteins (FB_BACnet_NotificationSink [▶ 260]) verbunden werden.

	Time	Notification
✓ ℹ	12.11.2014 10:15:28	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
✓ ⛔	12.11.2014 10:15:31	Beckhoff_927609 - BI_0 : BI_0: Eingang ist in Fehler
✓ ⚠	12.11.2014 10:15:32	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
✓ ℹ	12.11.2014 10:15:34	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
✓ ⚠	12.11.2014 10:15:37	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
✓ ℹ	12.11.2014 10:15:40	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
✓ ⚠	12.11.2014 10:15:43	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
✓ ⛔	12.11.2014 10:15:46	Beckhoff_927609 - BI_0 : BI_0: Eingang ist in Fehler
✓ ℹ	12.11.2014 10:15:48	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
✓ ⚠	12.11.2014 10:15:49	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS
✓ ℹ	12.11.2014 10:15:52	Beckhoff_927609 - BI_0 : BI_0: Eingang ist EIN
✓ ⚠	12.11.2014 10:15:55	Beckhoff_927609 - BI_0 : BI_0: Eingang ist AUS

Abb. 35: Bild-1: Online-Ansicht der BACnet Notification Sink.

VAR_INPUT

```
bRemove     : BOOL;
nIndex      : UDINT;
```

bRemove: *FALSE* → *TRUE* = Eventeintrag wird gelöscht.

nIndex: Index des zu löschenden Event-Eintrags (0..n).

VAR_OUPUT

```
bDone       : BOOL;
bNoEntry    : BOOL;
bBusy       : BOOL;
bError      : BOOL;
nErrorId    : UINT;
```

bDone: Lesen der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bRemove** zurückgesetzt wird. Wurde **bRemove** zurückgesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bNoEntry: Kein Eintrag zum Löschen vorhanden - Eventeintrag unter **nIndex** ist leer. **bNoEntry** bleibt so lange gesetzt bis ein erneutes Löschen beginnt. Der Ausgang dient dazu das Ende der Event-Liste zu erkennen.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrorId: Fehlercode, siehe BACnet_Globals [▶ 330] für eine Übersicht.

VAR_IN_OUT

```
stAdsConn       : ST_BACnet_AdsConnection;
```

stAdsConn: Verknüpfung mit dem Ausgang **stBACnetAds** des entsprechenden NotificationSink-Bausteins.

Beispiel: Bausteinaufruf

```
0015
0016    NSink        : FB_BACnet_NotificationSink;
0017    fbDelEvent   : FB_BACnet_NSinkRemoveEvent;
0018    bDelEvent    : BOOL;
0019
```

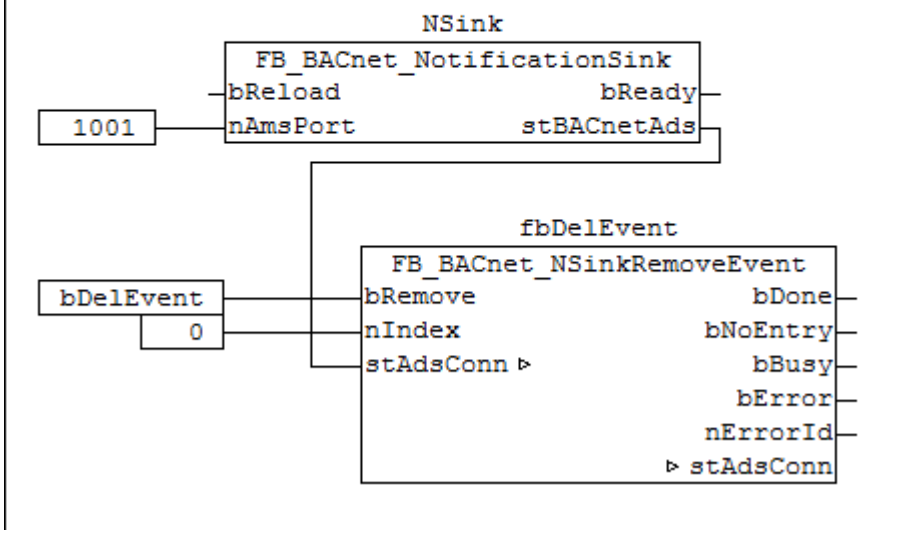
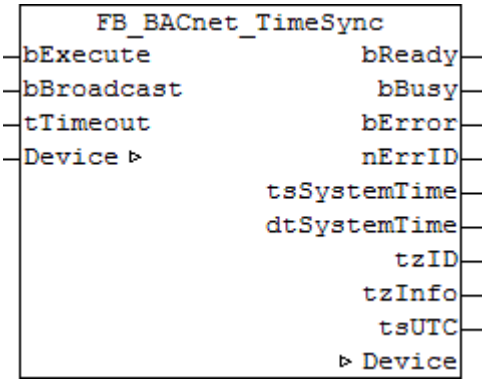


Abb. 36: Bild-7: Beispiel für die Verwendung. Löschen eines Event-Eintrags einer BACnet Notification Sink mit AMS Port 1001.

Der AMS Port ist abhängig von der TwinCAT System Manager Konfiguration des entsprechenden BACnet Device (siehe FB_BACnet_NotificationSink [▶ 260]).

4.5.5 FB_BACnet_TimeSync



Anwendung

Der Baustein wird zyklisch aufgerufen und gibt die aktuelle lokale Zeit und die aktuelle UTC Zeit aus. Wenn eine Flanke am Eingang **bExecute** angelegt wird, dann sendet der Baustein die Uhrzeit an den BACnet Stack und in das BACnet-Netzwerk als Broadcast Nachricht (Zeitmaster) - vorausgesetzt der Eingang **bBroadcast** ist gesetzt; andernfalls wird nur der lokale BACnet Stack synchronisiert.

Die Funktion enthält u.a. den Baustein FB_LocalSystemTime, um die lokale Zeit zu lesen.

Dieser Funktionsbaustein kann mit BACnet Stack Revision 6 eingesetzt werden. Jedoch wird unter Revision 6 immer ein Broadcast gesendet (Eingang **bBroadcast** ist unter Revision 6 deaktiviert).

Im Folgenden ist eine Übersicht zur Zeitsynchronisierung dargestellt:

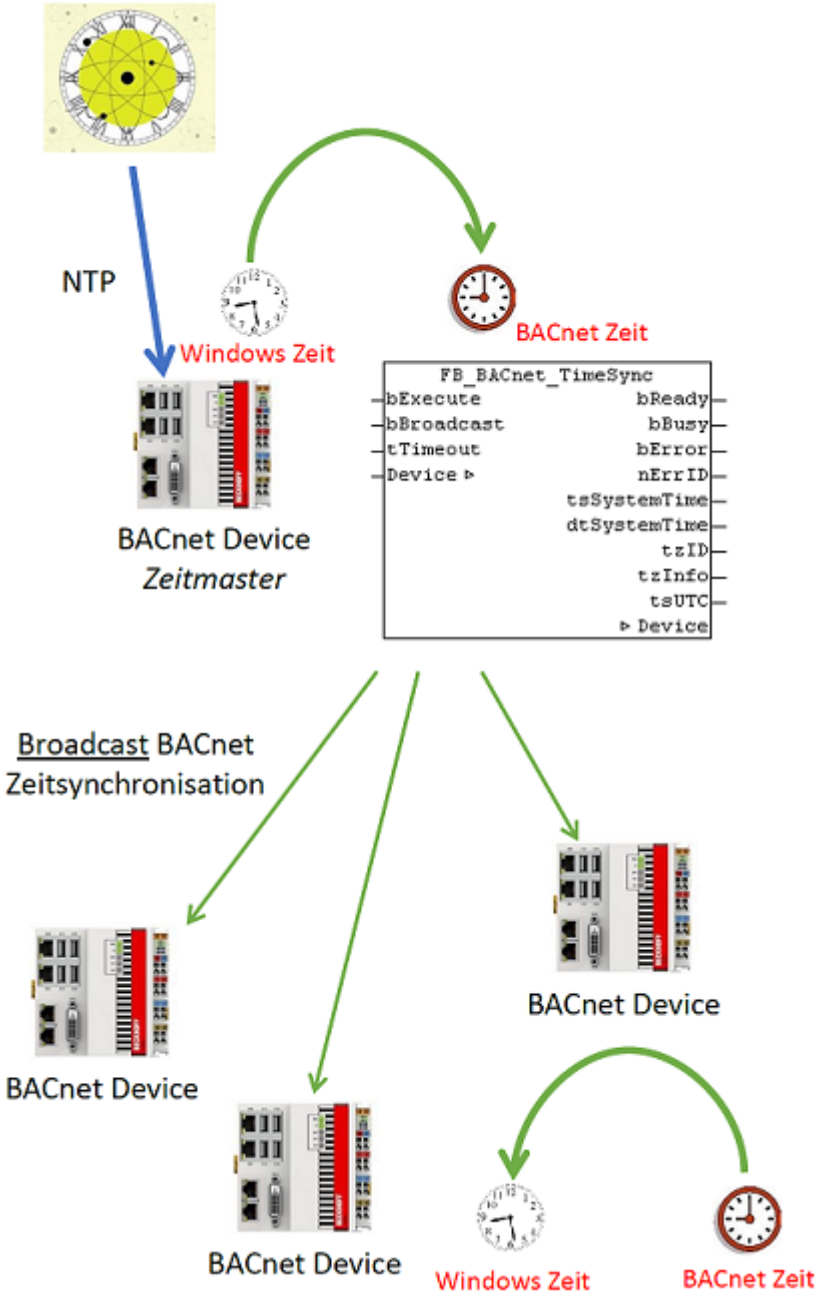


Abb. 37: Bild-1: Beispielhafter Ablauf der Zeitsynchronisierung im BACnet Netzwerk:

1. Eine externe Zeitquelle (z.B. NTP) synchronisert die Betriebssystemzeit des *Zeitmaster* Systems (NTP → Betriebssystemzeit)
2. Der Baustein FB_BACnet_TimeSync des *Zeitmaster* Systems synchronisiert bei **bExecute** = *TRUE* den BACnet Stack mit der aktuelle Betriebssystemzeit (Betriebssystemzeit → BACnet Zeit)
3. Ist der Eingang **bBroadcast** am Baustein gesetzt, erfolgt das Senden einer BACnet Broadcast Zeitsynchronisierungsnachricht ins Netzwerk
4. BACnet Devices im Netzwerk stellen die Uhrzeit entsprechend um (BACnet Zeit → Betriebssystemzeit)

Typischerweise erfolgt die Zeitsynchronisierung in größeren Abständen mehrerer Stunden bzw. täglich. Zeitpunkte zwischen 23 bis 3 Uhr sollten gemieden werden, da es u.U. zur Überschneidung mit der automatischen Sommer-/Winterzeitumschaltung der zu synchronisierenden Geräte kommen kann.

Anwendungsinformationen:

Bei der Zeitsynchronisation werden 2 Modi unterschieden:

1. Jedes BACnet-Gerät ist an eine externe Zeitquelle (z.B. NTP) angeschlossen. FB_BACnet_TimeSync muss auf jedem BACnet-Controller ausgeführt werden. **bBroadcast** hat immer den Wert *FALSE* und **bExecute** wird in regelmäßigen Abständen getriggert.
2. Ein Gerät ist an eine externe Zeitquelle (z.B. NTP) angeschlossen. Dann muss auf diesem Gerät FB_BACnet_TimeSync mit **bBroadcast** = *TRUE* ausgeführt und **bExecute** regelmäßig getriggert werden.
   Alle anderen BACnet-Geräte führen FB_BACnet_TimeSync mit **bBroadcast** = *FALSE* und **bExecute** = *FALSE* aus, werden via BACnet synchronisiert und stellen automatisch ihre lokale Betriebssystemzeit nach.

VAR_INPUT

```
bExecute      : BOOL;
bBroadcast    : BOOL:=TRUE;
tTimeout      : TIME:=BACnet_ADSTimeOut;
```

bExecute: Bei steigender Flanke beginnt die Synchronisierung.

bBroadcast: *TRUE* → Sendet zusätzlich eine Broadcast Zeitsynchronisierung in das BACnet Netzwerk; *FALSE* → nur lokal Synchronisieren (ab Revision 12 und höher)

tTimeout: Optionaler Eingang, Überwachungszeit für den ADS Zugriff (Default: siehe BACnet_ADSTimeOut [▶ 330]).

VAR_OUPUT

```
bReady        : BOOL;
bBusy         : BOOL;
bError        : BOOL;
nErrID        : UDINT;
tsSystemTime  : TIMESTRUCT;
dtSystemTime  : ST_BACnet_DateTime;
tzID          : E_TimeZoneID;
tzInfo        : ST_TimeZoneInformation;
tsUTC         : TIMESTRUCT;
```

bReady: Ausgabe der Uhrzeit (**tsSystemTime**, **dtSystemTime**, **tzID**, **tzInfo** und **tsUTC**) ist gültig

bBusy: Baustein ist beschäftigt (Zeitsynchronisierung ist aktiv)

bError: Fehler während der Abarbeitung.

nErrID: ADS Fehlercode.

tsSystemTime: Lokale Betriebssystemzeit (Windows Uhrzeit) als PLC typische Datenstruktur

dtSystemTime: Lokale Betriebssystemzeit (Windows Uhrzeit) als BACnet typische Datenstruktur

tzID: ID der aktuellen Zeitzone

tzInfo: Informationen zur aktuellen Zeitzone

tsUTC: Betriebssystemzeit als UTC Zeitstempel (GMT oder Koordinierte Weltzeit) als PLC typische Datenstruktur

VAR_IN_OUT

```
Device        : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins.

Beispiel

```
0001 PROGRAM MAIN_PROG
0002 VAR
0003     Device         : FB_BACnet_Device;
0004     fbTimeSync     : FB_BACnet_TimeSync;
0005     bTimeSyncExec  : SR;
0006 END_VAR
```

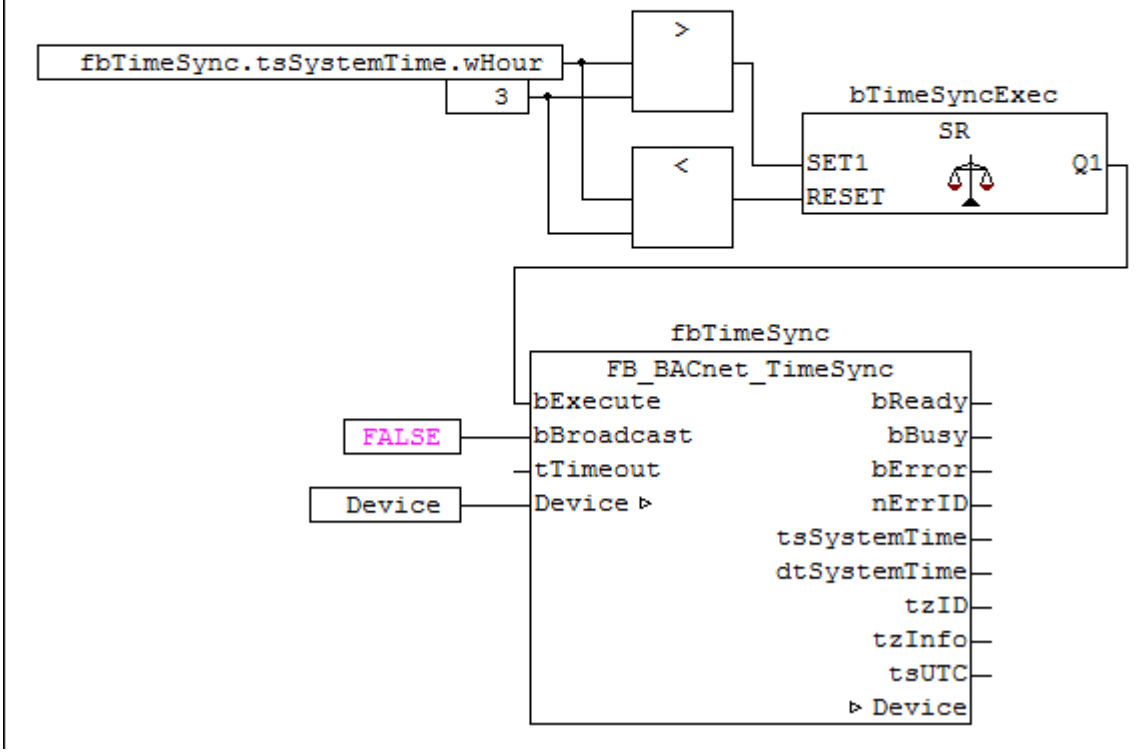
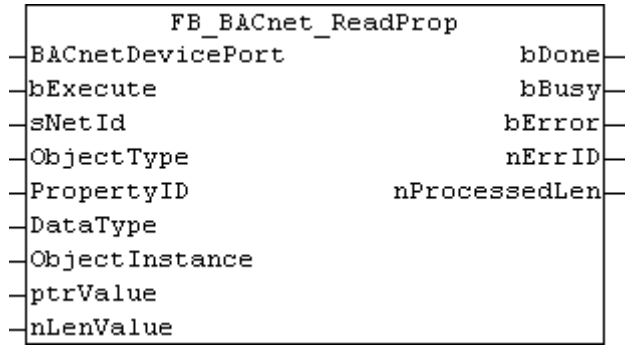


Abb. 38: Bild-2: Beispiel für das zyklische Synchronisieren der lokalen Uhrzeit (kein Broadcast). Bei Überschreiten von 3:00 Uhr wird bExecute gesetzt und bleibt aktiv bis die Uhrzeit 0:00 Uhr erreicht → tägliche Synchronisierung.

4.5.6 FB_BACnet_ReadProp

Funktionsbaustein für den Zugriff auf BACnet Properties über ADS. Sämtliche BACnet Properties, die im Online-Reiter des jeweiligen BACnet-Objekts angezeigt werden, von Server und Client Objekten können über ADS gelesen bzw. geschrieben werden. Wird auf einen Client (entfernter BACnet-Server) über ADS zugegriffen, dann werden die ADS Zugriffe in BACnet-Abfragen umgewandelt und an den entfernten Server gesendet.



Anwendung

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Eingang **BACnetDevicePort** und **sNetId** muss mit dem entsprechenden AMS Port und NetID belegt werden. Der AMS Port und NetID können für jeden BACnet-Server und -Client über den zugehörigen BACnet-Device Baustein abgefragt werden.

Die AMS NetID entspricht nicht der lokalen AMS NetID und muss immer angegeben werden - keine Verwendung von Leerstrings möglich!

Die aktuelle AMS NetID und AMS Port werden vom zugehörigen FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243] ausgegeben:

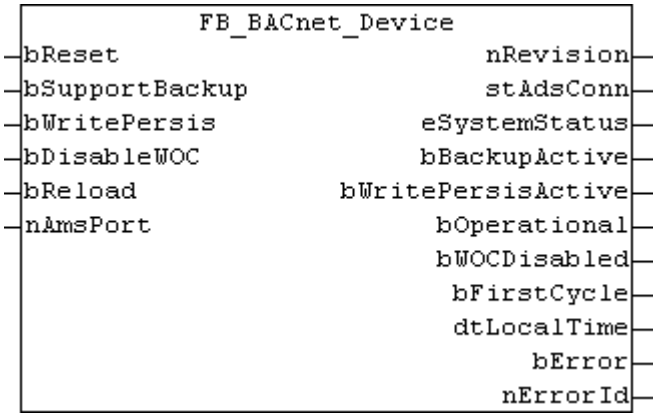


Abb. 39: Bild-1: FB_BACnet_Device → stAdsConn gibt die aktuellen ADS Verbindungsdaten aus.

AMS NetId und AMS Port in der Ausgabe-Struktur **stAdsConn** des Device Bausteins.

```
TYPE ST_BACnet_AdsConnection:
  STRUCT
    bValid      : BOOL;
    nReload     : USINT;
    sAmsNetId   : T_AmsNetId;
    nAmsPort    : T_AmsPort;
    bServer     : BOOL;
    bClient     : BOOL;
    bNSink      : BOOL;
    nDeviceId   : UDINT;
  END_STRUCT
END_TYPE
```

Zudem kann der AMS Port und die AMS NetID im TwinCAT System Manager angezeigt werden (siehe Bild-3 und -4). Je nach Konfiguration können Ports für Server und Client variieren.

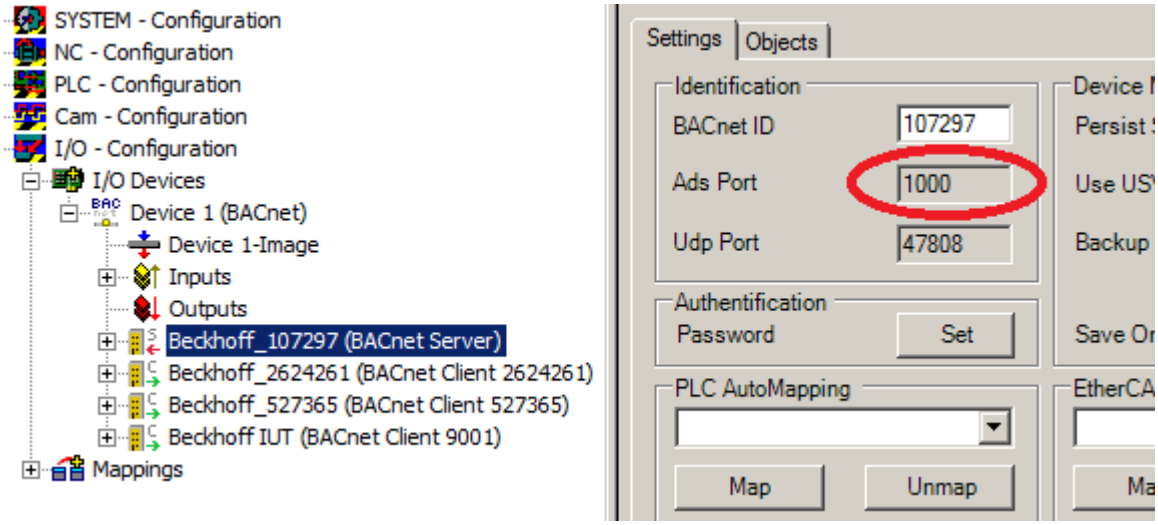


Abb. 40: Bild-3: AMS Port des lokalen BACnet Server im System Manager

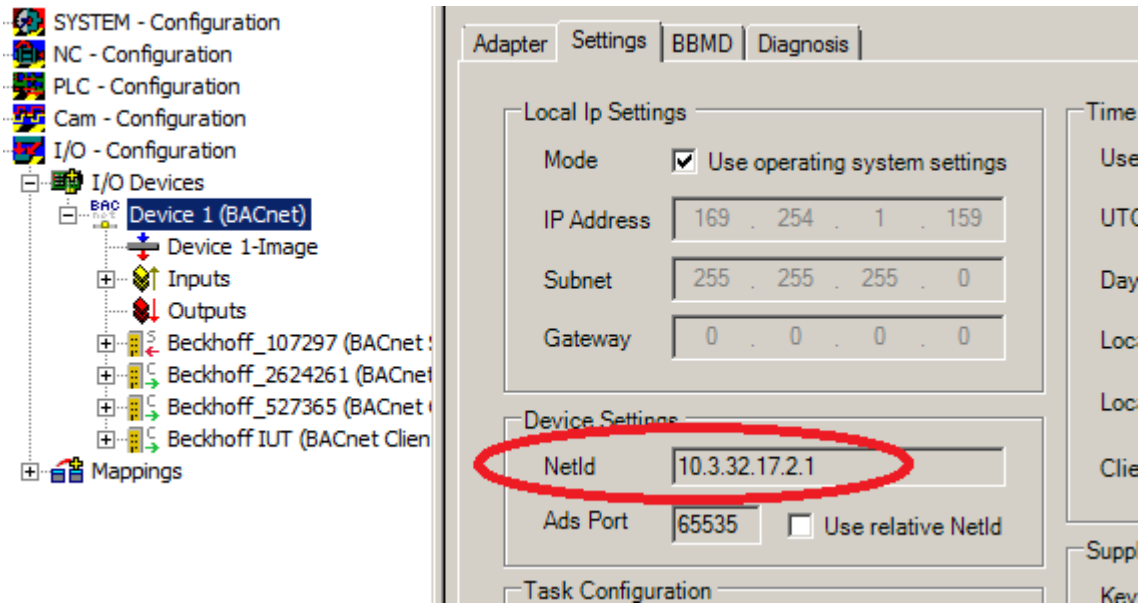


Abb. 41: Bild-4: AMS NetId des BACnet Device

VAR_INPUT

```
BACnetDevicePort  : UINT:=16#FFFF;
bExecute          : BOOL;
sNetId            : T_AmsNetId;
ObjectType        : E_BACNETOBJECTTYPE;
PropertyID        : E_BACNETPROPERTYIDENTIFIER;
DataType          : E_BACNETDATATYPES;
ObjectInstance    : DINT;
ptrValue          : POINTER TO BYTE;
nLenValue         : UDINT;
```

BACnetDevicePort: AMS Port unter dem das gewünschte Objekt angelegt wurde.

bExecute: Steigende Flanke am Eingang startet den Lesevorgang. Fallende Flanke löscht den Ausgang **bDone**.

sNetId: AMS NetId des BACnet Devices (Adapter) unter dem der Server bzw. Client konfiguriert wurde.

ObjectType: Enumeration des Objekttypes (AnalogValue, BinaryInput...).

PropertyID: Enumeration des Property-Identifiers (*Present_Value*, *Status_Flags*...).

DataType: Datentyp der Property (Bool, BinaryPV, Real, Unsigned Integer...).

ObjectInstance: Objekt-Nummer des BACnet Objekts (die unteren 22 Bit des *Object_Identifier*).

ptrValue: Zeiger auf die Variable in der die gelesenen Daten abgelegt werden sollen (zu ermitteln mit ADR()).

nLenValue: Byte-Länge der Variable in der die gelesenen Daten abgelegt werden sollen (zu ermitteln mit SIZEOF()).

VAR_OUPUT

```
bDone         : BOOL;
bBusy         : BOOL;
bError        : BOOL;
nErrID        : UDINT;
nProcessedLen : UDINT;
```

bDone: Lesen der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bExecute** zurückgesetzt wird. Wurde **bExecute** zurück gesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrID: ADS Fehlercode.

nProcessedLen: Byte-Länge der gelesenen Daten. Die Länge kann sich von der Ziellänge **nLenValue** unterscheiden (kleiner/gleich). Die Lücke zwischen der Eingangsdatenlänge **nLenValue** und der tatsächlich gelesen Datenlänge **nProcessedLen** wird im übergebenen Datenbereich mit "Nullen" aufgefüllt.

Beispiel

```
0001 PROGRAM MAIN
0002 VAR
0003     Device       : FB_BACnet_Device;
0004     fbReadProp   : FB_BACnet_ReadProp;
0005     ePvBI0       : E_BACNETBINARYPV;
0006 END_VAR
0007
```

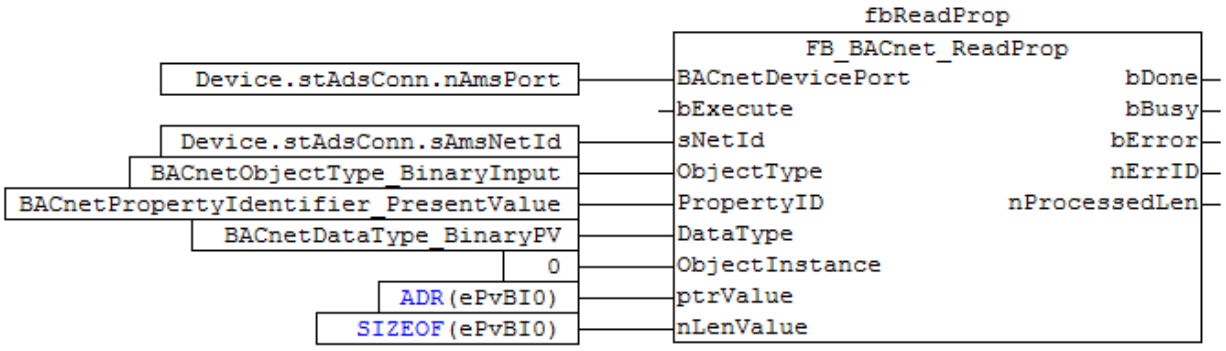
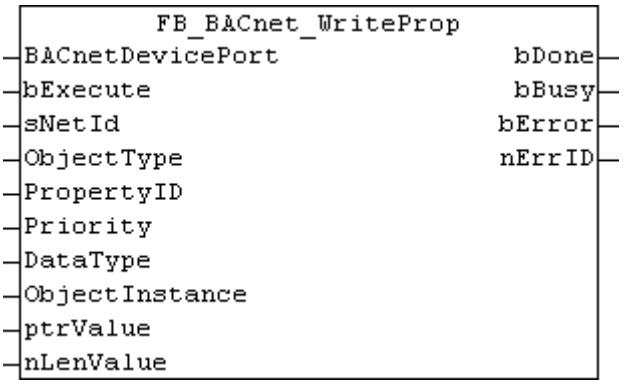


Abb. 42: Bild-5: Beispiel für die Verwendung. Lesen der Property Present_Value eines BACnet Binary Input Objekts.

4.5.7 FB_BACnet_WriteProp

Funktionsbaustein für den Zugriff auf BACnet Properties über ADS. Sämtliche BACnet Properties, die im Online-Reiter des jeweiligen BACnet-Objekts angezeigt werden, von Server und Client Objekten können über ADS gelesen bzw. geschrieben werden. Wird auf einen Client (entfernter BACnet-Server) über ADS zugegriffen, dann werden die ADS Zugriffe in BACnet-Abfragen umgewandelt und an den entfernten Server gesendet.



Anwendung

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Eingang **BACnetDevicePort** und **sNetId** muss mit dem entsprechenden AMS Port und AMS NetID belegt werden. Der AMS Port und AMS NetID können für jeden BACnet-Server und -Client über den zugehörigen BACnet-Device Baustein abgefragt werden.

Die AMS NetID entspricht nicht der lokalen AMS NetID und muss immer angegeben werden - keine Verwendung von Leerstrings möglich!

Die aktuelle AMS NetID und AMS Port werden vom zugehörigen FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243] ausgegeben:

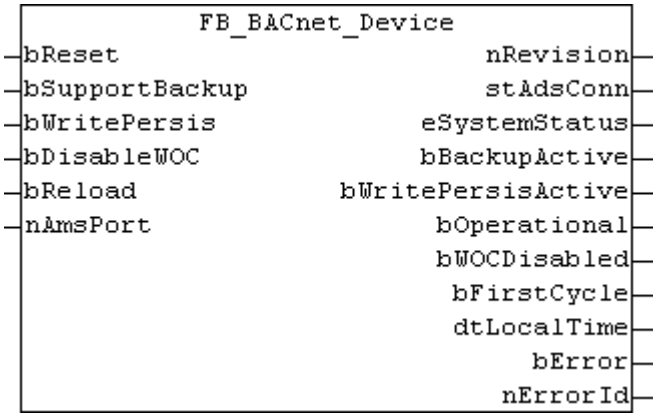


Abb. 43: Bild-1: FB_BACnet_Device → stAdsConn gibt die aktuellen ADS Verbindungsdaten aus.

AMS NetId und AMS Port in der Ausgabe-Struktur stAdsConn des Device Bausteins.

```
TYPE ST_BACnet_AdsConnection :
  STRUCT
    bValid    : BOOL;
    nReload   : USINT;
    sAmsNetId : T_AmsNetId;
    nAmsPort  : T_AmsPort;
    bServer   : BOOL;
    bClient   : BOOL;
    bNSink    : BOOL;
    nDeviceId : UDINT;
  END_STRUCT
END_TYPE
```

Zudem kann der AMS Port und die AMS NetID im TwinCAT System Manager angezeigt werden (siehe Bild-3 und -4). Je nach Konfiguration können Ports für Server und Client variieren.

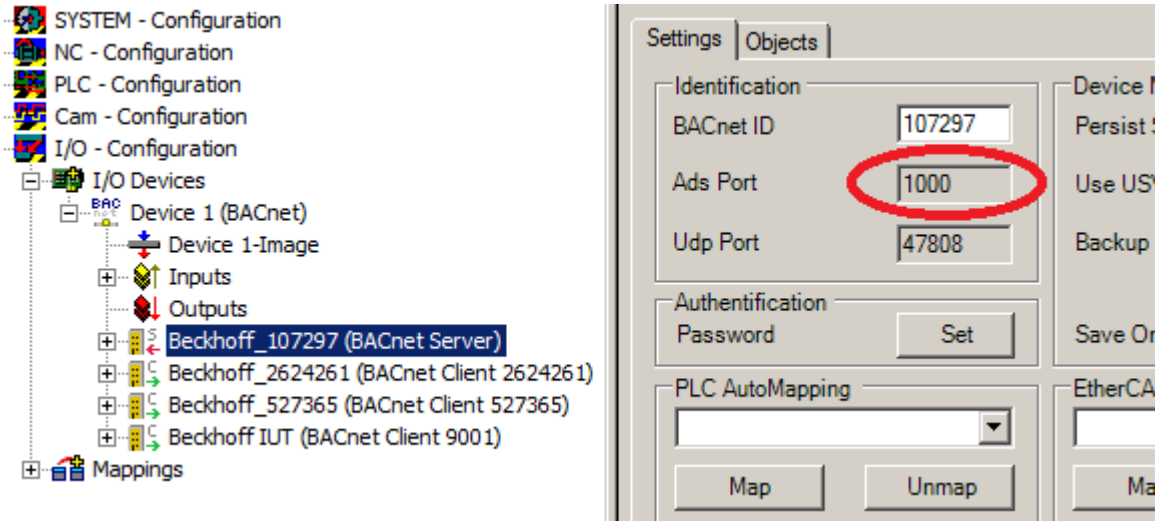


Abb. 44: Bild-3: AMS Port des lokalen BACnet Server im System Manager

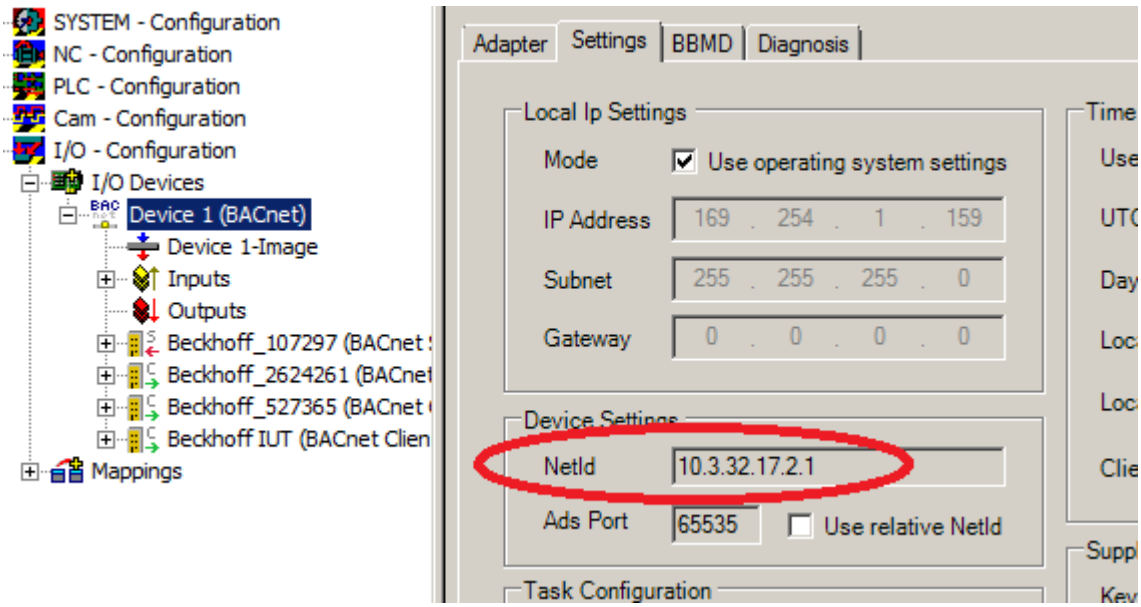


Abb. 45: Bild-4: AMS NetId des BACnet Device

VAR_INPUT

```
BACnetDevicePort   : UINT:=16#FFFF;
bExecute           : BOOL;
sNetId             : T_AmsNetId;
ObjectType         : E_BACNETOBJECTTYPE;
PropertyID         : E_BACNETPROPERTYIDENTIFIER;
Priority           : E_BACNETPRIORITY;
DataType           : E_BACNETDATATYPES;
ObjectInstance     : UDINT;
ptrValue           : POINTER TO BYTE;
nLenValue          : UDINT;
```

BACnetDevicePort: AMS Port unter dem das gewünschte Objekt angelegt wurde.

bExecute: Steigende Flanke am Eingang startet den Schreibvorgang. Fallende Flanke löscht den Ausgang **bDone**.

sNetId: AMS NetId des BACnet Devices (Adapter) unter dem der Server bzw. Client konfiguriert wurde.

ObjectType: Enumeration des Objekttypes (AnalogValue, BinaryInput...).

PropertyID: Enumeration des Property-Identifiers (*Present_Value*, *Status_Flags*...).

Priority: Enumeration der Priorität des Schreibzugriffs. Die Priorität wird bei Schreibzugriffen auf priorisierbare Properties benötigt (z.B. *Present_Value*). Ein Zugriff mit Priorität x erzeugt einen Eintrag im zugehörigen *Priority_Array* an x-ter Stelle.

DataType: Datentyp der Property (Bool, BinaryPV, Real, Unsigned Integer...).

ObjectInstance: Objekt-Nummer des BACnet Objekts (die unteren 22 Bit des *Object_Identifier*).

ptrValue: Zeiger auf die Variable in die die Daten abgelegt werden sollen (zu ermitteln mit ADR()).

nLenValue: Byte-Länge der Variable in die die Daten abgelegt werden sollen (zu ermitteln mit SIZEOF()).

VAR_OUPUT

```
bDone          : BOOL;
bBusy          : BOOL;
bError         : BOOL;
nErrID         : UDINT;
```

bDone: Schreiben der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bExecute** zurückgesetzt wird. Wurde **bExecute** zurück gesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrID: ADS Fehlercode.

Beispiel

```
0001 PROGRAM MAIN
0002 VAR
0003     Device       : FB_BACnet_Device;
0004     fbWriteProp  : FB_BACnet_WriteProp;
0005     ePvBO0       : E_BACNETBINARYPV;
0006 END_VAR
0007
```

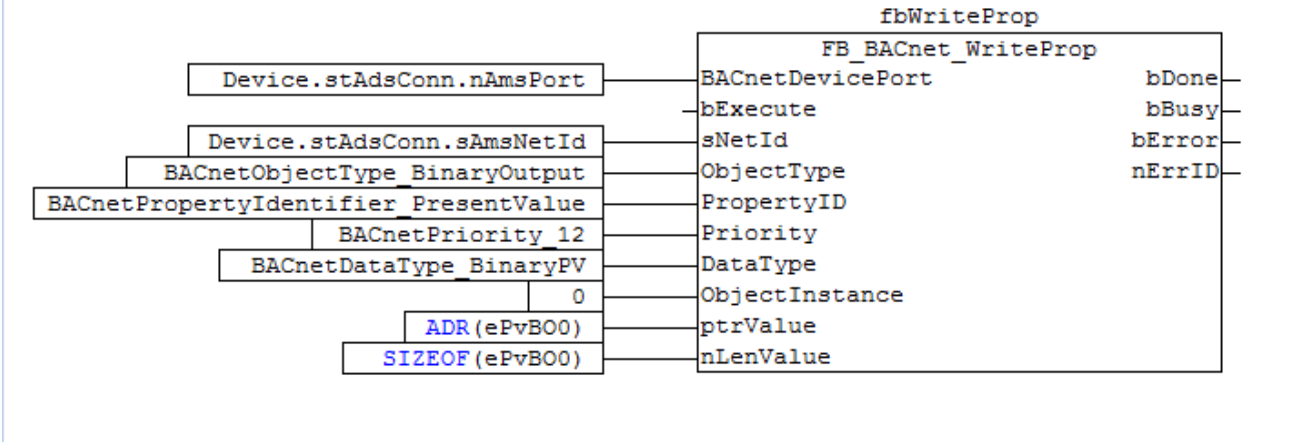
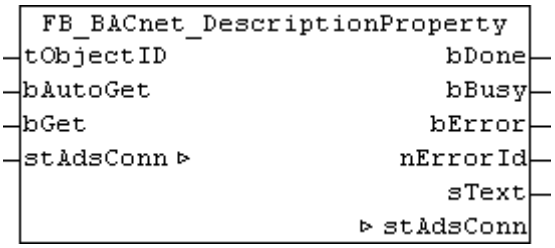


Abb. 46: Bild-5: Beispiel für die Verwendung. Schreiben der Property Present_Value eines BACnet Binary Output Objekts.

4.5.8 FB_BACnet_DescriptionProperty



Anwendung

Mit Hilfe des Bausteins kann die Property *Description* eines beliebigen BACnet Objekts ausgelesen werden. Der Eingang **tObjectID** legt dabei das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz), auf das zugegriffen werden soll. Unter Beispiel [▶ 283] wird eine mögliche Beschaltung gezeigt.

Baustein intern wird zudem die Decodierung der Property anhand des String-Codings vorgenommen. Dabei werden folgende Property Encodings unterstützt: *UTF-8*, *UCS2*, *UCS4* und *ISO8859-1*. Der Ausgabe-String **sText** liegt Windows-1252 codiert vor (siehe auch FB_BACnet_StringExtDecode [▶ 323]). Tritt bei der Decodierung der Property ein Fehler auf, so wird dies am Ausgang **nErrorId** signalisiert.

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Ein-/Ausgang **stAdsConn** muss mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins (FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243]) verbunden werden.

VAR_INPUT

```
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bAutoGet        : BOOL:=TRUE;
bGet            : BOOL;
```

tObjectID: Legt das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz) auf das zugegriffen werden soll.

bAutoGet: *TRUE* = Lese die Property automatisch aus, wenn die ADS Verbindung oder der *Object_Identifier* sich geändert haben. Eine ADS Verbindungsänderung liegt vor, wenn die Verbindung nach einem Unterbruch wiederhergestellt oder die AMS NetID bzw. Port geändert wurde. Ein automatisches Auslesen geschieht nicht zyklisch und auch nicht bei Änderung der Property selbst!

bGet: *FALSE* → *TRUE* = *Property* wird unabhängig vom Eingang **bAutoGet** einmalig ausgelesen.

VAR_OUPUT

```
bDone       : BOOL;
bBusy       : BOOL;
bError      : BOOL;
nErrorId    : UINT;
sText       : T_MaxString;
```

bDone: Lesen der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bGet** und **bAutoGet** zurückgesetzt sind oder ein erneutes Auslesen beginnt. Wurde **bGet** und **bAutoGet** zurückgesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrorId: Fehlercode, siehe BACnet_Globals [▶ 330] für eine Übersicht.

sText: *Description* als Windows-1252 codierter String.

VAR_IN_OUT

```
stAdsConn       : ST_BACnet_AdsConnection;
```

stAdsConn: Verknüpfung mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins.

Beispiel

```
0001 PROGRAM MAIN
0002 VAR
0003     Device       : FB_BACnet_Device;
0004     fbDesc       : FB_BACnet_DescriptionProperty;
0005     tonPropRead : TON;
0006 END_VAR
```

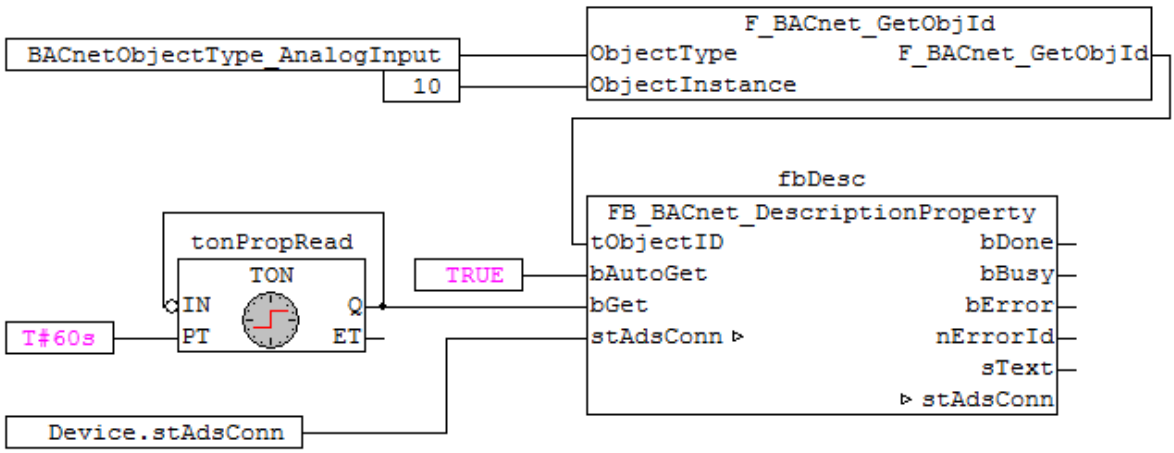
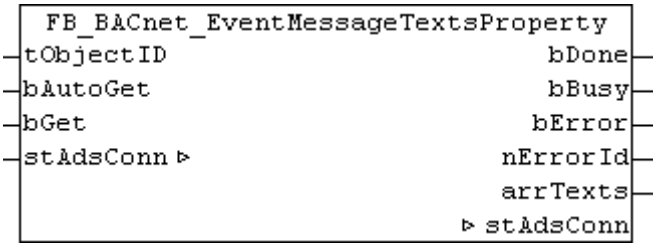


Abb. 47: Bild-1: Beispiel für die Verwendung. Zyklisches Lesen der Property Description eines BACnet Analog Input Objekts (AnalogInput: 10).

4.5.9 FB_BACnet_EventMessageTextsProperty



Anwendung

Mit Hilfe des Bausteins kann die Property *Event_Message_Texts* eines beliebigen BACnet Objekts ausgelesen werden. Der Eingang **tObjectID** legt dabei das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz), auf das zugegriffen werden soll. Unter Beispiel [▶ 285] wird eine mögliche Beschaltung gezeigt.

Baustein intern wird zudem die Decodierung der Property anhand des String-Codings vorgenommen. Dabei werden folgende Property Encoding unterstützt: *UTF-8*, *UCS2*, *UCS4* und *ISO8859-1*. Die Ausgabe-Strings im Array **arrTexts** liegen Windows-1252 codiert vor (siehe auch FB_BACnet_StringExtDecode [▶ 323]). Tritt bei der Decodierung der Property ein Fehler auf, so wird dies am Ausgang **nErrorId** signalisiert.

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Ein-/Ausgang **stAdsConn** muss mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins (FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243]) verbunden werden.

VAR_INPUT

```
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bAutoGet        : BOOL:=TRUE;
bGet            : BOOL;
```

tObjectID: Legt das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz) auf das zugegriffen werden soll.

bAutoGet: *TRUE* = Lese die Property automatisch aus, wenn die ADS Verbindung oder der *Object_Identifier* sich geändert haben. Eine ADS Verbindungsänderung liegt vor, wenn die Verbindung nach einem Unterbruch wiederhergestellt oder die AMS NetID bzw. Port geändert wurde. Ein automatisches Auslesen geschieht nicht zyklisch und auch nicht bei Änderung der Property selbst!

bGet: *FALSE* → *TRUE* = *Property* wird unabhängig vom Eingang **bAutoGet** einmalig ausgelesen.

VAR_OUPUT

```
bDone       : BOOL;
bBusy       : BOOL;
bError      : BOOL;
nErrorId    : UINT;
arrTexts    : ARRAY [0..2] OF T_MaxString;
```

bDone: Lesen der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bGet** und **bAutoGet** zurückgesetzt sind oder ein erneutes Auslesen beginnt. Wurde **bGet** und **bAutoGet** zurückgesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrorId: Fehlercode, siehe BACnet Globals [▶ 330] für eine Übersicht.

arrTexts: Meldetexte als Windows-1252 codierte Strings.

VAR_IN_OUT

```
stAdsConn       : ST_BACnet_AdsConnection;
```

stAdsConn: Verknüpfung mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins.

Beispiel

```
0001 PROGRAM MAIN
0002 VAR
0003     Device       : FB_BACnet_Device;
0004     fbMsgTexts   : FB_BACnet_EventMessageTextsProperty;
0005     tonPropRead : TON;
0006 END_VAR
```

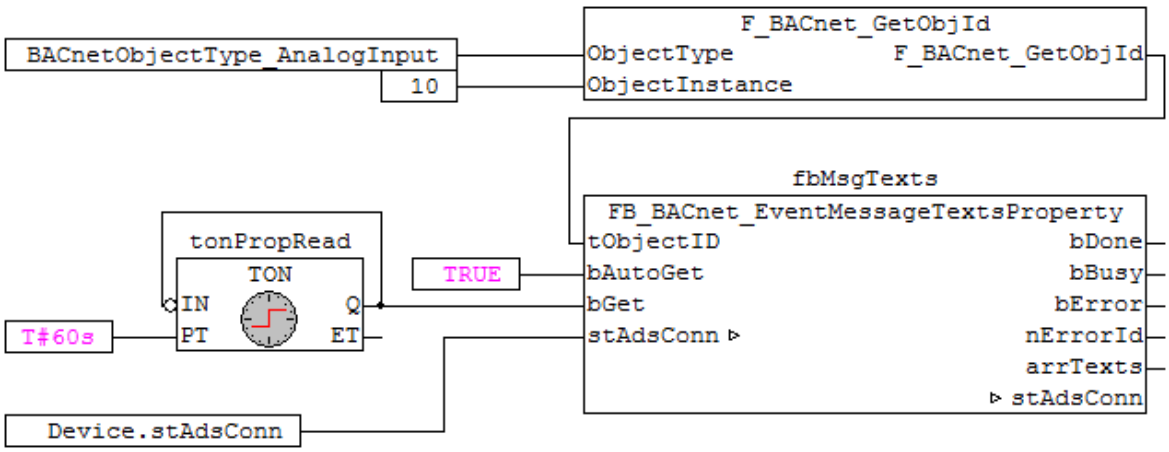
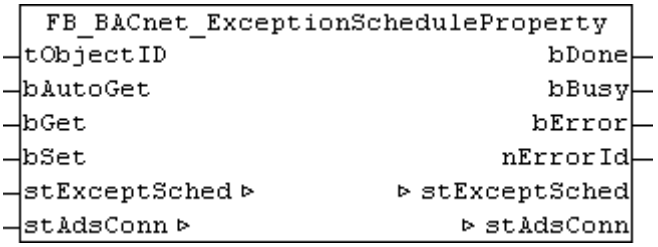


Abb. 48: Bild-1: Beispiel für die Verwendung. Zyklisches Lesen der Property Event_Message_Texts eines BACnet Analog Input Objekts (AnalogInput: 10).

4.5.10 FB_BACnet_ExceptionScheduleProperty



Anwendung

Mit Hilfe des Bausteins kann die Property *Exception_Schedule* eines beliebigen BACnet Objekts (typischerweise BACnet Schedule Objekt) ausgelesen werden. Der Eingang **tObjectID** legt dabei das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz), auf das zugegriffen werden soll. Unter Beispiel [▶ 287] wird eine mögliche Beschaltung gezeigt.

Die zu lesenden bzw. zu schreibenden Daten werden in derselben Ein-/Ausgangsdatenstrukur abgelegt (siehe auch ST_BACnet_ExceptionScheduleBool [▶ 361]).

Als Datentype für Einträge der Property *Exception_Schedule* wird seitens PLC Baustein lediglich *Bool* und *Null* unterstützt! Einträge mit anderen Datentypen werden ignoriert bzw. beim Schreiben in die Property gelöscht.

ExceptionSchedule	38	4 Elements	BACnetSpecialEventList
[1]		(Calendar:0;1;{(18:00:00;True);(06:00:00;False)})	calendarReference
calendarReference		Calendar:0	BACnetObjectIdentifier
eventPriority		1	UnsignedInteger
listOfTimeValues		{(18:00:00;True);(06:00:00;False)}	BACnetTimeValueList
[1]		(18:00:00;True)	BACnetTimeValue
time		18:00:00	BACnetTime
value		True	BoolValue
[2]		(06:00:00;False)	BACnetTimeValue
time		06:00:00	BACnetTime
value		False	BoolValue

Abb. 49: Bild-1: Beispiel Eintrag der Property Exception_Schedule mit Datentyp Bool.

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Ein-/Ausgang **stAdsConn** muss mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins (FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243]) verbunden werden.

Für die Pufferung der ADS Daten wird der ADS Puffer aus den globalen Variablen verwendet (siehe ST_BACnet_GlobalAdsBuffer [▶ 363]).

VAR_INPUT

```
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bAutoGet        : BOOL:=TRUE;
bGet            : BOOL;
bSet            : BOOL;
```

tObjectID: Legt das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz) auf das zugegriffen werden soll.

bAutoGet: *TRUE* = Lese die Property automatisch aus, wenn die ADS Verbindung oder der *Object_Identifier* sich geändert haben. Eine ADS Verbindungsänderung liegt vor, wenn die Verbindung nach einem Unterbruch wiederhergestellt oder die AMS NetID bzw. Port geändert wurde. Ein automatische Auslesen geschieht nicht zyklisch und auch nicht bei Änderung der Property selbst!

bGet: *FALSE* → *TRUE* = Property wird unabhängig vom Eingang **bAutoGet** einmalig ausgelesen.

bSet: *FALSE* --> *TRUE* = Property wird geschrieben.

VAR_OUPUT

```
bDone           : BOOL;
bBusy           : BOOL;
bError          : BOOL;
nErrorId        : UINT;
```

bDone: Lesen bzw. Schreiben der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bGet**, **bSet** und **bAutoGet** zurückgesetzt sind oder ein erneuter Lese- bzw. Schreibvorgang beginnt. Wurde **bGet**, **bSet** und **bAutoGet** zurückgesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrorId: Fehlercode, siehe BACnet_Globals [▶ 330] für eine Übersicht.

VAR_IN_OUT

```
stExceptSched       : ST_BACnet_ExceptionScheduleBool;
stAdsConn       : ST_BACnet_AdsConnection;
```

stExceptSched: Ein-/Ausgangsdatenstrukur für die Einträge der Property *Exception_Schedule* mit Datentyp *Bool* oder *Null*. Im Folgenden wird ein Eintrag in der PLC und im TwinCAT System Manager gegenüber gestellt (siehe Beispiel [▶ 287]).

stAdsConn: Verknüpfung mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins.

Beispiel 1: Gegenüberstellung PLC und System Manager

```
stExcepSched
    .bReqGet = FALSE
    .bReqSet = FALSE
    .nGet = 39
    .nSet = 0
    .bBusy = FALSE
  .arrEntries
    .arrEntries[0]
      .eEntryType = BACnetCalendarEntryType_Ref
      .ePriority = BACnetPriority_1
    + .stDate
    + .stWeekNDay
    - .stCalRef
        .tCalObjectId = 25165824
    - .arrTimeValue
        - .arrTimeValue[0]
            .bValid = TRUE
          - .stTime
              .nHour = 18
              .nMinute = 0
              .nSecond = 0
              .nHundredths = 0
            .eType = BACnetDataType_Bool
            .bValue = TRUE
        - .arrTimeValue[1]
            .bValid = TRUE
          - .stTime
              .nHour = 6
              .nMinute = 0
              .nSecond = 0
              .nHundredths = 0
            .eType = BACnetDataType_Bool
            .bValue = FALSE
```

Abb. 50: Bild-2: Beispiel eines Property Eintrag in der PLC

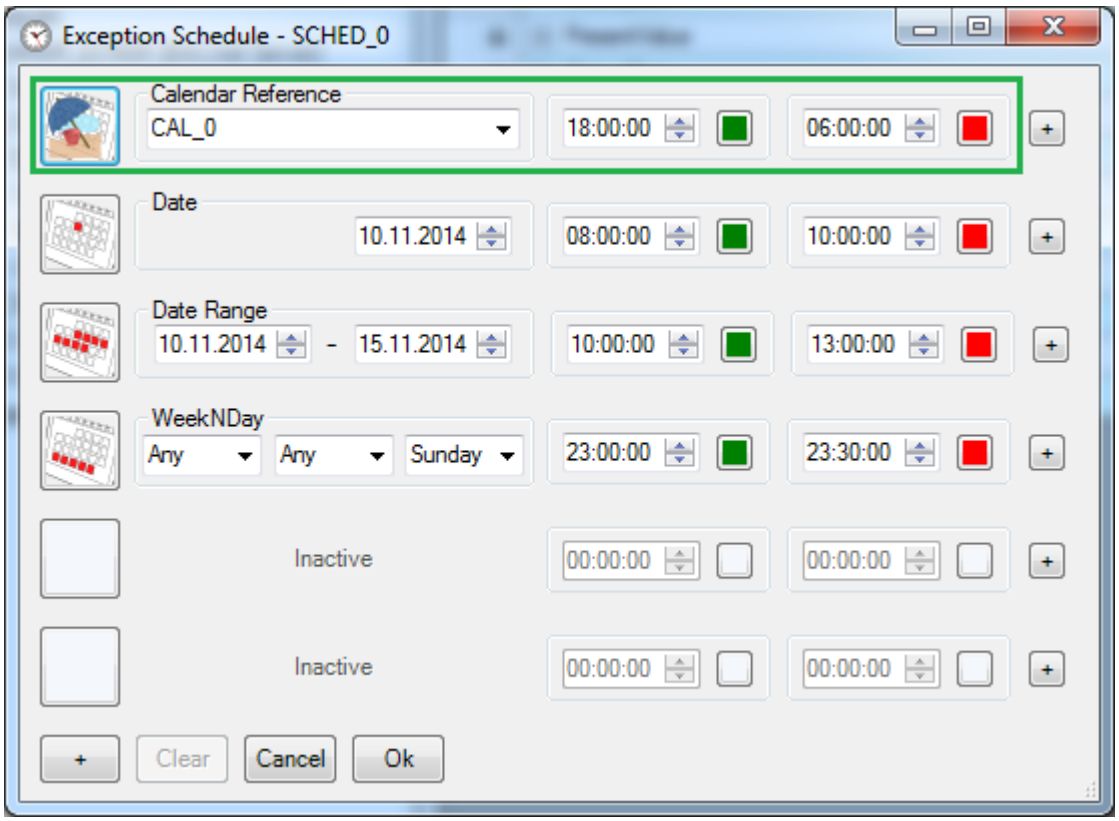


Abb. 51: Bild-3: Beispiel eines Property Eintrag im TwinCAT System Manager

Beispiel 2: Bausteinaufruf

```
0001 PROGRAM MAIN
0002 VAR
0003     Device         : FB_BACnet_Device;
0004     fbExceptSched  : FB_BACnet_ExceptionScheduleProperty;
0005     stExcepSched   : ST_BACnet_ExceptionScheduleBool;
0006
0007     bWriteScheduleData: BOOL;
0008     bReadScheduleData: BOOL;
0009 END_VAR
```

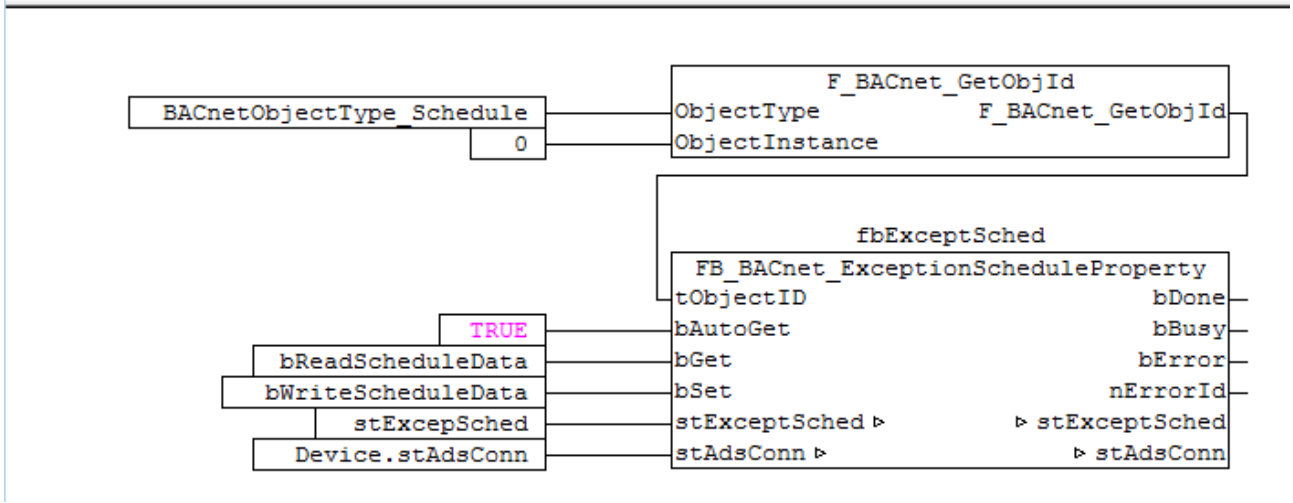
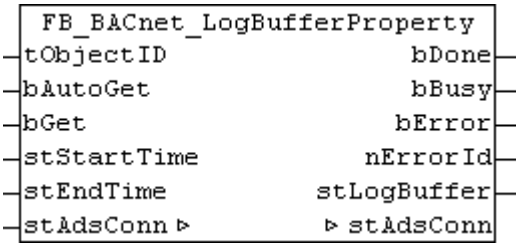


Abb. 52: Bild-4: Beispiel für die Verwendung. Lesen und Schreiben der Property Exception_Schedule eines BACnet Schedule Objekts.

4.5.11 FB_BACnet_LogBufferProperty



Anwendung

Mit Hilfe des Bausteins kann die Property *Log_Buffer* eines beliebigen BACnet Objekts (typischerweise BACnet TrendLog Objekt) ausgelesen werden. Der Eingang **tObjectID** legt dabei das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz), auf das zugegriffen werden soll. Unter Beispiel [▶ 291] wird eine mögliche Beschaltung gezeigt.

DDie zu lesenden Daten werden in der Ausgangsdatenstruktur **stLogBuffer** abgelegt (siehe auch ST_BACnet_LogBufferReal [▶ 365]).

Als Datentype für Einträge der Property *Log_Buffer* wird seitens PLC Baustein lediglich *Real* unterstützt! Einträge mit anderen Datentypen werden ignoriert.

⊟ LogBuffer	131	100 Elements	BACnetLogRecordList
⊞ [1]		10.11.2014 12:19:44: {(log_disabled)}	BACnetLogRecord
⊞ [2]		10.11.2014 12:20:51: {(log_disabled)}	BACnetLogRecord
⊞ [3]		10.11.2014 12:21:02: {(log_enabled)}	BACnetLogRecord
⊟ [4]		10.11.2014 12:21:02: {28}	BACnetLogRecord
⊞ timestamp		10.11.2014 12:21:02	BACnetDateTime
⊟ logDatum		{28}	BACnetLogDatumList
⊟ [1]		28	realValue
realValue		28	Real
⊞ statusFlags	☑	0x00000004	BACnetStatusFlags
⊞ [5]		10.11.2014 12:21:03: {28,5}	BACnetLogRecord
⊞ [6]		10.11.2014 12:21:04: {29}	BACnetLogRecord

Abb. 53: Bild-1: Beispiel Eintrag der Property Log_Buffer mit Datentyp Real.

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Ein-/Ausgang **stAdsConn** muss mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins (FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243]) verbunden werden.

Für die Pufferung der ADS Daten wird der ADS Puffer aus den globalen Variablen verwendet (siehe ST_BACnet_GlobalAdsBuffer [▶ 363]).

VAR_INPUT

```
tObjectID    : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bAutoGet     : BOOL:=TRUE;
bGet         : BOOL;
stStartTime  : ST_BACnet_DateTime := (
                   stDate := (nYear       := 16#FF,
                   nMonth      := 16#FF,
                   nDay    := 16#FF),
                   stTime := (nHour       := 16#FF,
                   nMinute     := 16#FF,
                   nSecond     := 16#FF,
                   nHundredths := 16#FF));
stEndTime    : ST_BACnet_DateTime := (
                   stDate := (nYear       := 16#FF,
```

```
                nMonth      := 16#FF,
                nDay    := 16#FF),
                stTime := (nHour       := 16#FF,
                nMinute     := 16#FF,
                nSecond     := 16#FF,
                nHundredths := 16#FF));
```

tObjectID: Legt das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz) auf das zugegriffen werden soll.

bAutoGet: *TRUE* = Lese die Property automatisch aus, wenn die ADS Verbindung oder der *Object_Identifier* sich geändert haben. Eine ADS Verbingsänderung liegt vor, wenn die Verbingung nach einem Unterbruch wiederhergestellt oder die AMS NetID bzw. Port geändert wurde. Ein automatische Auslesen geschieht nicht zyklisch und auch nicht bei Änderung der Property selbst!

bGet: *FALSE* → *TRUE* = Property wird unabhängig vom Eingang **bAutoGet** einmalig ausgelesen.

stStartTime, **:stEndTime** Nur Einträge aus dem *Log_Buffer* mit Zeitstempel innerhalb von Start- und Endzeitpunkt werden ausgegeben. Standardmäßig ist der Filter deaktivert (Platzhalter 255 → undefiniert).

VAR_OUPUT

```
bDone         : BOOL;
bBusy         : BOOL;
bError        : BOOL;
nErrorId      : UINT;
stLogBuffer   : ST_BACnet_LogBufferReal;
```

bDone: Lesen bzw. Schreiben der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bGet**, **bSet** und **bAutoGet** zurückgesetzt sind oder ein erneuter Lese- bzw. Schreibvorgang beginnt. Wurde **bGet**, **bSet** und **bAutoGet** zurückgesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrorId: Fehlercode, siehe BACnet_Globals für eine Übersicht.

stLogBuffer: Ausgangsdatenstrukur für die Einträge der Property *Log_Buffer* mit Datentyp *Real*. Im Folgenden werden zwei Einträge in der PLC und im TwinCAT System Manager gegenübergestellt (siehe Beispiel).

VAR_IN_OUT

```
stAdsConn     : ST_BACnet_AdsConnection;
```

stAdsConn: Verknüpfung mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins.

Beispiel 1: Gegenüberstellung PLC und System Manager

```
.stLogBuffer
    .nGet = 1
    .sObjectDesc = 'TLOG_0'
    .nCount = 97
  .arrEntries
      .arrEntries[0]
          .nTimeStamp = 1863042261
        .stDateTime
            .stDate
            .stTime
                .nHour = 12
                .nMinute = 21
                .nSecond = 2
                .nHundredths = 91
          .fValue = 28
      .arrEntries[1]
          .nTimeStamp = 1863042271
        .stDateTime
            .stDate
            .stTime
                .nHour = 12
                .nMinute = 21
                .nSecond = 3
                .nHundredths = 91
          .fValue = 28.5
```

Abb. 54: Bild-2: Beispiel von 2 Log-Eintrag in der PLC

LogBuffer	131	100 Elements	BACnetLogRecordList
[1]		10.11.2014 12:19:44: {(log_dis...	BACnetLogRecord
[2]		10.11.2014 12:20:51: {(log_dis...	BACnetLogRecord
[3]		10.11.2014 12:21:02: {(log_en...	BACnetLogRecord
[4]		10.11.2014 12:21:02: {28}	BACnetLogRecord
timestamp		10.11.2014 12:21:02	BACnetDateTime
date		10.11.2014	BACnetDate
time		12:21:02	BACnetTime
logDatum		{28}	BACnetLogDatumList
[1]		28	realValue
statusFlags	☑	0x00000004	BACnetStatusFlags
[5]		10.11.2014 12:21:03: {28,5}	BACnetLogRecord
timestamp		10.11.2014 12:21:03	BACnetDateTime
date		10.11.2014	BACnetDate
time		12:21:03	BACnetTime
logDatum		{28,5}	BACnetLogDatumList
[1]		28,5	realValue
statusFlags	☑	0x00000004	BACnetStatusFlags

Abb. 55: Bild-3: Beispiel von 2 Log-Eintrag im TwinCAT System Manager

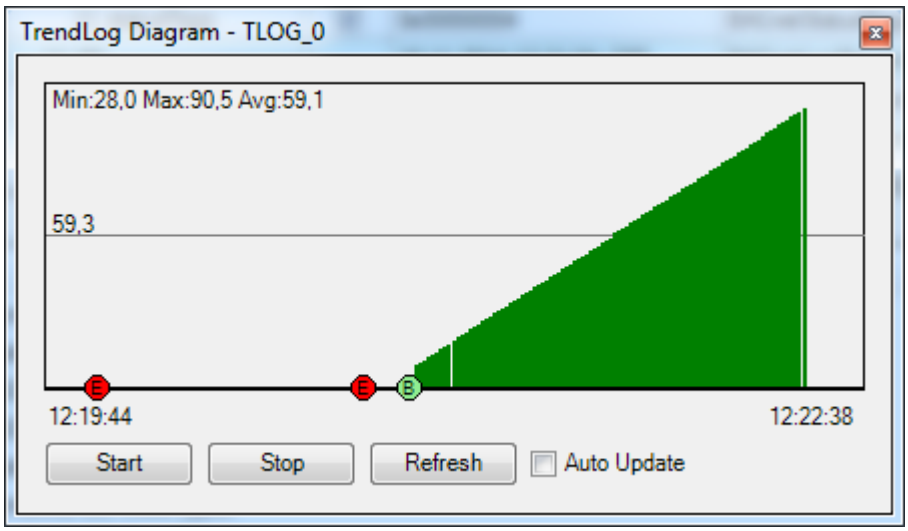


Abb. 56: Bild-4: Beispielhafte Darstellung der Property Log_Buffer im TwinCAT System Manager.

Beispiel 2: Bausteinaufruf

```
0001 PROGRAM MAIN
0002 VAR
0003     Device       : FB_BACnet_Device;
0004     fbLogBuf     : FB_BACnet_LogBufferProperty;
0005     tonReadProp  : TON;
0006 END_VAR
```

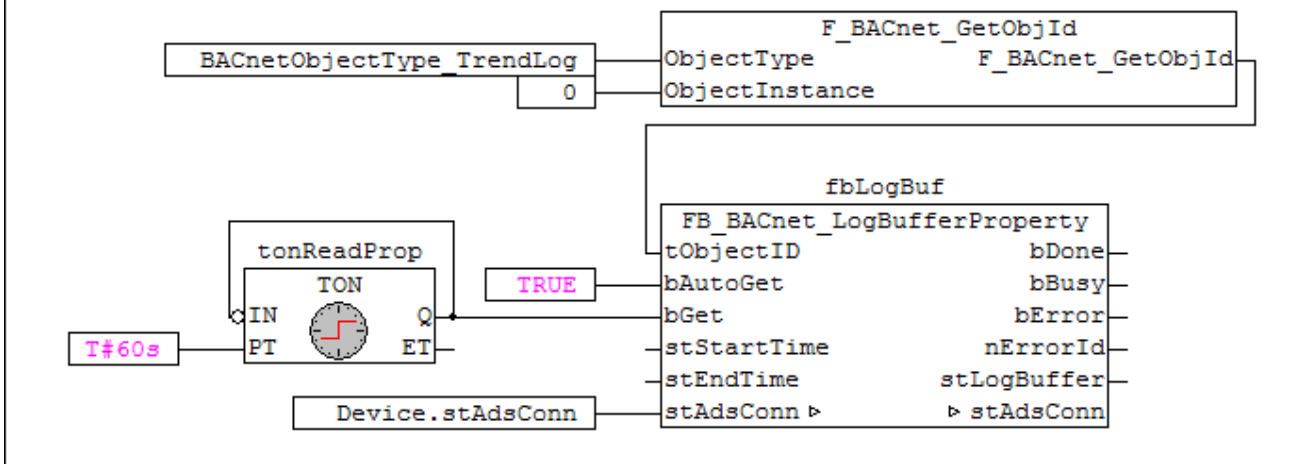
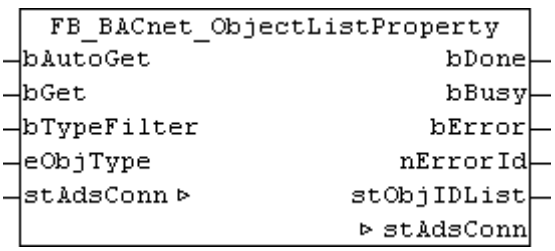


Abb. 57: Bild-5: Beispiel für die Verwendung. Zyklisches Lesen der Property Log_Buffer eines BACnet TrendLog Objekts.

4.5.12 FB_BACnet_ObjectListProperty



Anwendung

Mit Hilfe des Bausteins kann die Property *Object_List* eines beliebigen BACnet Device Objekts (lokal oder remote) ausgelesen werden. Unter Beispiel [▶ 294] wird eine mögliche Beschaltung gezeigt.

Mit Hilfe eines Objekt-Typ Filters kann die Ausgabe **stObjIDList** eingeschränkt werden.

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Ein-/Ausgang **stAdsConn** muss mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins (FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243]) verbunden werden.

Für die Pufferung der ADS Daten wird der ADS Puffer aus den globalen Variablen verwendet (siehe ST_BACnet_GlobalAdsBuffer [▶ 363]).

⊟ ObjectList	76	{Device:927609;File:0;File:1...	BACnetObjectIdentifier[]
⊞ [1]		Device:927609	BACnetObjectIdentifier
⊞ [2]		File:0	BACnetObjectIdentifier
⊞ [3]		File:1	BACnetObjectIdentifier
⊞ [4]		StructuredView:0	BACnetObjectIdentifier
⊞ [5]		Loop:0	BACnetObjectIdentifier
⊞ [6]		Schedule:0	BACnetObjectIdentifier

Abb. 58: Bild-1: Online Ansicht der Property Object_List

VAR_INPUT

```
bAutoGet     : BOOL:=TRUE;
bGet         : BOOL;
bTypeFilter  : BOOL;
eObjType     : E_BACnetObjectType;
```

tObjectID: Legt das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz) auf das zugegriffen werden soll.

bAutoGet: *TRUE* = Lese die Property automatisch aus, wenn die ADS Verbindung oder der *Object_Identifier* sich geändert haben. Eine ADS Verbindungsänderung liegt vor, wenn die Verbindung nach einem Unterbruch wiederhergestellt oder die AMS NetID bzw. Port geändert wurde. Ein automatisches Auslesen geschieht nicht zyklisch und auch nicht bei Änderung der Property selbst!

bTypeFilter: *TRUE* = Ausgabe **stObjIDList** wird auf einen BACnet Objekt Type beschränkt; *FALSE* = Ausgabe sämtlicher BACnet Objekte des BACnet Servers (oder Client).

eObjType: Filter des auszugebenden BACnet Objekt Typs. Wenn **bTypeFitler** = *FALSE*, wird der Eingang ignoriert.

VAR_OUPUT

```
bDone        : BOOL;
bBusy        : BOOL;
bError       : BOOL;
nErrorId     : UINT;
stObjIDList  : ST_BACnet_ObjectIdentifierList;
```

bDone: Lesen der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bGet** und **bAutoGet** zurückgesetzt sind oder ein erneutes Auslesen beginnt. Wurde **bGet** und **bAutoGet** zurück gesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrorId: Fehlercode, siehe BACnet_Globals für eine Übersicht.

stObjIDList: Struktur mit der Anzahl und Liste der gelesenen BACnet Objekt IDs (Objekt Type und Objekt Instanz).

VAR_IN_OUT

```
stAdsConn     : ST_BACnet_AdsConnection;
```

stAdsConn: Verknüpfung mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins.

Beispiel

```
0001 PROGRAM MAIN
0002 VAR
0003     Device       : FB_BACnet_Device;
0004     fbObjList    : FB_BACnet_ObjectListProperty;
0005     tonPropRead : TON;
0006 END_VAR
```

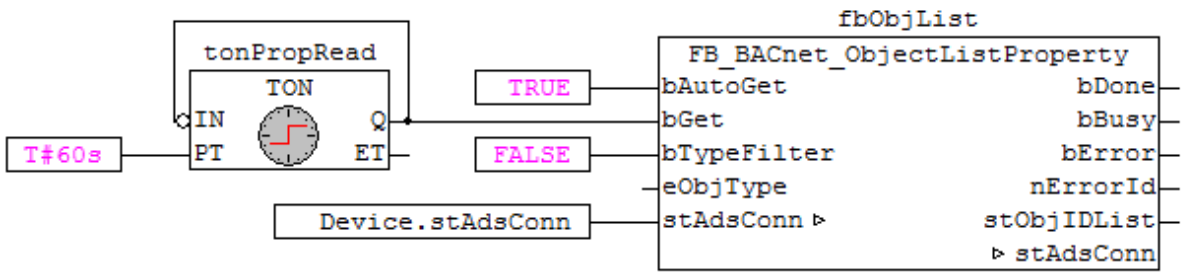
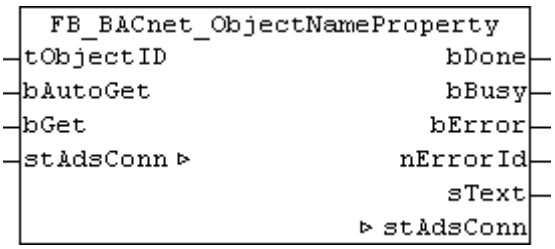


Abb. 59: Bild-2: Beispiel für die Verwendung. Zyklisches Lesen der Property Object_List eines BACnet Servers (aus dem BACnet Device Objekt).

4.5.13 FB_BACnet_ObjectNameProperty



Anwendung

Mit Hilfe des Bausteins kann die Property *Object_Name* eines beliebigen BACnet Objekts ausgelesen werden. Der Eingang **tObjectID** legt dabei das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz), auf das zugegriffen werden soll. Unter Beispiel [▶ 296] wird eine mögliche Beschaltung gezeigt.

Baustein intern wird zudem die Decodierung der Property anhand des String-Codings vorgenommen. Dabei werden folgende Property Encoding unterstützt: *UTF-8*, *UCS2*, *UCS4* und *ISO8859-1*. Der Ausgabe-String **sText** liegt Windows-1252 codiert vor (siehe auch FB_BACnet_StringExtDecode [▶ 323]). Tritt bei der Decodierung der Property ein Fehler auf, so wird dies am Ausgang **nErrorId** signalisiert.

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Ein-/Ausgang **stAdsConn** muss mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins (FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243]) verbunden werden.

VAR_INPUT

```
tObjectID  : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bAutoGet   : BOOL:=TRUE;
bGet       : BOOL;
```

tObjectID: Legt das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz) auf das zugegriffen werden soll.

bAutoGet: *TRUE* = Lese die Property automatisch aus, wenn die ADS Verbindung oder der *Object_Identifier* sich geändert haben. Eine ADS Verbindungsänderung liegt vor, wenn die Verbindung nach einem Unterbruch wiederhergestellt oder die AMS NetID bzw. Port geändert wurde. Ein automatische Auslesen geschieht nicht zyklisch und auch nicht bei Änderung der Property selbst!

bGet: *FALSE* → *TRUE* = *Property* wird unabhängig vom Eingang **bAutoGet** einmalig ausgelesen.

VAR_OUPUT

```
bDone      : BOOL;
bBusy      : BOOL;
bError     : BOOL;
nErrorId   : UINT;
sText      : T_MaxString;
```

bDone: Lesen der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bGet** und **bAutoGet** zurückgesetzt sind oder ein erneutes Auslesen beginnt. Wurde **bGet** und **bAutoGet** zurückgesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrorId: Fehlercode, siehe BACnet_Globals [▶ 330] für eine Übersicht.

sText: Objektename als Windows-1252 codierter String.

VAR_IN_OUT

```
stAdsConn      : ST_BACnet_AdsConnection;
```

stAdsConn: Verknüpfung mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins.

Beispiel

```
0001 PROGRAM MAIN
0002 VAR
0003     Device       : FB_BACnet_Device;
0004     fbObjName    : FB_BACnet_ObjectNameProperty;
0005     tonPropRead : TON;
0006 END_VAR
```

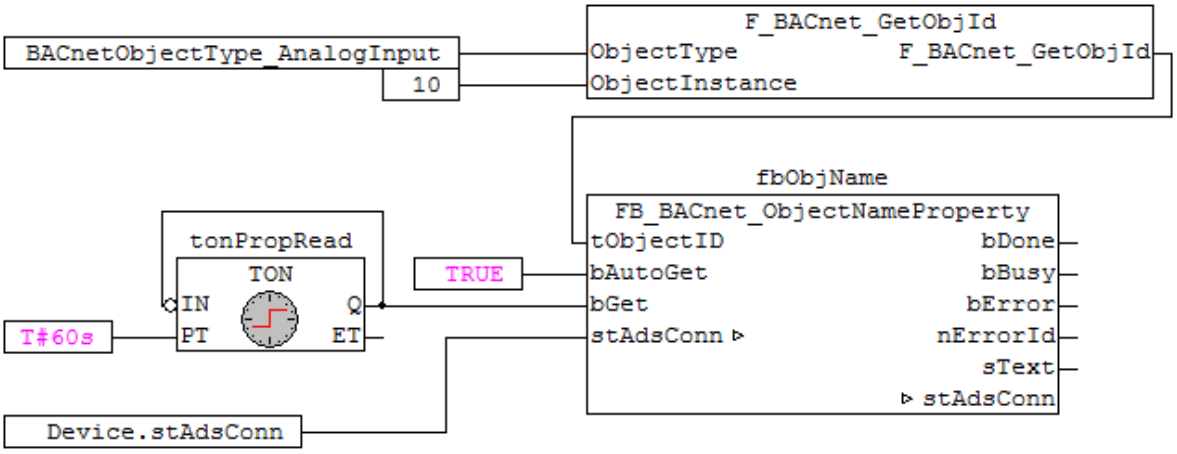
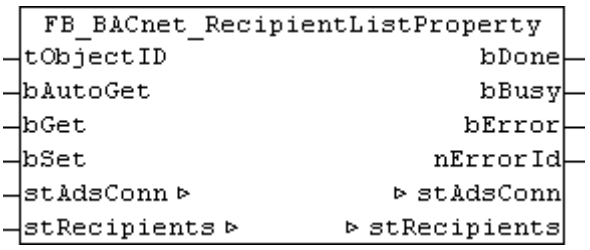


Abb. 60: Bild-1: Beispiel für die Verwendung. Zyklisches Lesen der Property Object_Name eines BACnet Analog Input Objekts (AnalogInput: 10).

4.5.14 FB_BACnet_RecipientListProperty



Anwendung

Mit Hilfe des Bausteins kann die Property *Recipient_List* eines beliebigen BACnet Objekts (typischerweise BACnet Notification Class Objekt) ausgelesen werden. Der Eingang **tObjectID** legt dabei das Objekt fest (*Object_Identifier*: Objekt Typ und Objekt Instanz), auf das zugegriffen werden soll. Unter Beispiel [▶ 298] wird eine mögliche Beschaltung gezeigt.

Die zu lesenden bzw. zu schreibenden Daten werden in derselben Ein-/Ausgangsdatenstruktur abgelegt (siehe auch ST_BACnet_RecipientListDevice [▶ 368]).

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Ein-/Ausgang **stAdsConn** muss mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins (FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243]) verbunden werden.

Für die Pufferung der ADS Daten wird der ADS Puffer aus den globalen Variablen verwendet (siehe ST_BACnet_GlobalAdsBuffer [▶ 363]).

VAR_INPUT

```
tObjectID      : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bAutoGet       : BOOL:=TRUE;
bGet           : BOOL;
bSet           : BOOL;
```

tObjectID: Legt das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz) auf das zugegriffen werden soll.

bAutoGet: *TRUE* = Lese die Property automatisch aus, wenn die ADS Verbindung oder der *Object_Identifier* sich geändert haben. Eine ADS Verbindungsänderung liegt vor, wenn die Verbindung nach einem Unterbruch wiederhergestellt oder die AMS NetID bzw. Port geändert wurde. Ein automatisches Auslesen geschieht nicht zyklisch und auch nicht bei Änderung der Property selbst!

bGet: *FALSE* → *TRUE* = Property wird unabhängig vom Eingang **bAutoGet** einmalig ausgelesen.

bSet: *FALSE* --> *TRUE* = Property wird geschrieben.

VAR_OUPUT

```
bDone          : BOOL;
bBusy          : BOOL;
bError         : BOOL;
nErrorId       : UINT;
```

bDone: Lesen bzw. Schreiben der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bGet**, **bSet** und **bAutoGet** zurückgesetzt sind oder ein erneuter Lese- bzw. Schreibvorgang beginnt. Wurde **bGet**, **bSet** und **bAutoGet** zurückgesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrorId: Fehlercode, siehe BACnet_Globals [▶ 330] für eine Übersicht.

VAR_IN_OUT

```
stAdsConn      : ST_BACnet_AdsConnection;
stRecipients   : ST_BACnet_RecipientListDevice;
```

stAdsConn: Verknüpfung mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins.

stRecipients: Ein-/Ausgangsdatenstruktur für die Einträge der Property *Recipient_List*. Im Folgenden wird ein Eintrag in der PLC und im TwinCAT System Manager gegenübergestellt (siehe Beispiel [▶ 298]).

Beispiel 1: Gegenüberstellung PLC und System Manager

```
stRecipList
    .nEntries = 1
    .arrDestList
        .arrDestList[0]
            .bMonday = TRUE
            .bTuesday = TRUE
            .bWednesday = TRUE
            .bThursday = TRUE
            .bFriday = TRUE
            .bSaturday = TRUE
            .bSunday = TRUE
            .stFromTime
                .nHour = 0
                .nMinute = 0
                .nSecond = 0
                .nHundredths = 0
            .stToTime
                .nHour = 23
                .nMinute = 59
                .nSecond = 59
                .nHundredths = 99
            .tDeviceId = 34482041
            .nProcessId = 10000
            .bToOffNormal = TRUE
            .bToFault = TRUE
            .bToNormal = TRUE
            .bIssueConfirmedNotification = FALSE
```

Abb. 61: Bild-1: Beispiel eines Property Eintrag in der PLC

⊟ RecipientList	102	{(65025;00:00:00;23:59:59;(De...	BACnetDestination[]
⊟ [1]		(65025;00:00:00;23:59:59;(De...	BACnetDestination
⊟ validDays		0x0000FE01 (Monday;Tuesday...	BACnetDaysOfWeek
Monday		☑	BitFieldBit
Tuesday		☑	BitFieldBit
Wednesday		☑	BitFieldBit
Thursday		☑	BitFieldBit
Friday		☑	BitFieldBit
Saturday		☑	BitFieldBit
Sunday		☑	BitFieldBit
⊞ fromTime		00:00:00	BACnetTime
⊞ toTime		23:59:59	BACnetTime
⊞ recipient		(Device:927609)	device
processIdentifier		10000	UnsignedInteger
⊟ transitions		0x0000E005 (to_offnormal;to_f...	BACnetEventTransitionBits
to_offnormal		☑	BitFieldBit
to_fault		☑	BitFieldBit
to_normal		☑	BitFieldBit
issueConfirmedNo...		☐	Bool

Abb. 62: Bild-2: Beispiel eines Property Eintrag im TwinCAT System Manager

Beispiel 2: Bausteinaufruf

```
0001 PROGRAM MAIN
0002 VAR
0003     Device       : FB_BACnet_Device;
0004     fbRecipList  : FB_BACnet_RecipientListProperty;
0005     stRecipList  : ST_BACnet_RecipientListDevice;
0006
0007     bReadRecipData   : BOOL;
0008     bWriteRecipData  : BOOL;
0009 END_VAR
```

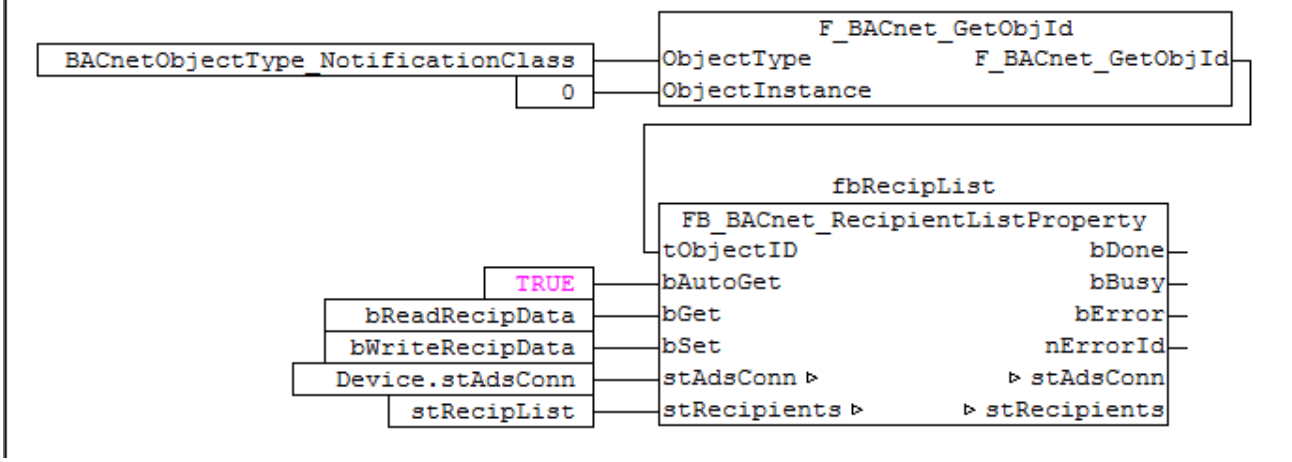
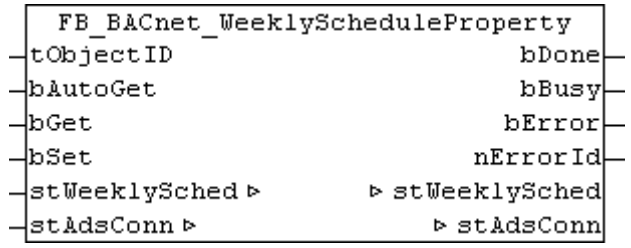


Abb. 63: Bild-3: Beispiel für die Verwendung. Lesen und Schreiben der Property Recipient_List eines BACnet Notification Class Objekts.

4.5.15 FB_BACnet_WeeklyScheduleProperty



Anwendung

Mit Hilfe des Bausteins kann die Property *Weekly_Schedule* eines beliebigen BACnet Objekts (typischerweise BACnet Schedule Objekt) ausgelesen werden. Der Eingang **tObjectID** legt dabei das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz), auf das zugegriffen werden soll. Unter Beispiel [▶ 301] wird eine mögliche Beschaltung gezeigt.

Die zu lesenden bzw. zu schreibenden Daten werden in derselben Ein-/Ausgangsdatenstruktur abgelegt (siehe auch ST_BACnet_WeeklyScheduleBool [▶ 372]).

Als Datentype für Einträge der Property *Weekly_Schedule* wird seitens PLC Baustein lediglich *Bool* und *Null* unterstützt! Einträge mit anderen Datentypen werden ignoriert bzw. beim Schreiben in die Property gelöscht.

WeeklySchedule	123	7 Elements	BACnetDailyScheduleList
[1] Monday		{(07:00:00;True);(18:00:00;False);(20:...	BACnetTimeValueList
[1]		(07:00:00;True)	BACnetTimeValue
time		07:00:00	BACnetTime
value		True	BoolValue
[2]		(18:00:00;False)	BACnetTimeValue
time		18:00:00	BACnetTime
value		False	BoolValue
[3]		(20:00:00;True)	BACnetTimeValue
[4]		(05:00:00;False)	BACnetTimeValue

Abb. 64: Bild-1: Beispiel Eintrag der Property Weekly_Schedule mit Datentyp Bool.

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Ein-/Ausgang **stAdsConn** muss mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins (FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243]) verbunden werden.

Für die Pufferung der ADS Daten wird der ADS Puffer aus den globalen Variablen verwendet (siehe ST_BACnet_GlobalAdsBuffer [▶ 363]).

VAR_INPUT

```
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
bAutoGet        : BOOL:=TRUE;
bGet            : BOOL;
bSet            : BOOL;
```

tObjectID: Legt das Objekt fest (*Object_Identifier*: Objekt Type und Objekt Instanz) auf das zugegriffen werden soll.

bAutoGet: *TRUE* = Lese die Property automatisch aus, wenn die ADS Verbindung oder der *Object_Identifier* sich geändert haben. Eine ADS Verbindungsänderung liegt vor, wenn die Verbindung nach einem Unterbruch wiederhergestellt oder die AMS NetID bzw. Port geändert wurde. Ein automatisches Auslesen geschieht nicht zyklisch und auch nicht bei Änderung der Property selbst!

bGet: *FALSE* → *TRUE* = Property wird unabhängig vom Eingang **bAutoGet** einmalig ausgelesen.

bSet: *FALSE* --> *TRUE* = Property wird geschrieben.

VAR_OUPUT

```
bDone           : BOOL;
bBusy           : BOOL;
bError          : BOOL;
nErrorId        : UINT;
```

bDone: Lesen bzw. Schreiben der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bGet**, **bSet** und **bAutoGet** zurückgesetzt sind oder ein erneuter Lese- bzw. Schreibvorgang beginnt. Wurde **bGet**, **bSet** und **bAutoGet** zurück gesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrorId: Fehlercode, siehe BACnet_Globals [▶ 330] für eine Übersicht.

VAR_IN_OUT

```
stWeeklySched      : ST_BACnet_WeeklyScheduleBool;
stAdsConn      : ST_BACnet_AdsConnection;
```

stWeeklySched : Ein-/Ausgangsdatenstruktur für die Einträge der Property *Weekly_Schedule* mit Datentyp *Bool* oder *Null*. Im Folgenden wird ein Eintrag in der PLC und im TwinCAT System Manager gegenüber gestellt (siehe Beispiel [▶ 301]).

stAdsConn: Verknüpfung mit dem Ausgang **stAdsConn** des entsprechenden Device-Bausteins.

Beispiel 1: Gegenüberstellung PLC und System Manager

```
stWeeklySched
    .bReqGet = FALSE
    .bReqSet = FALSE
    .nGet = 1
    .nSet = 0
    .bBusy = FALSE
  .arrEntries
    .arrEntries[0]
      .arrEntries[0][0]
        .bValid = TRUE
        .stTime
          .nHour = 7
          .nMinute = 0
          .nSecond = 0
          .nHundredths = 0
        .eType = BACnetDataType_Bool
        .bValue = TRUE
      .arrEntries[0][1]
        .bValid = TRUE
        .stTime
          .nHour = 18
          .nMinute = 0
          .nSecond = 0
          .nHundredths = 0
        .eType = BACnetDataType_Bool
        .bValue = FALSE
```

Abb. 65: Bild-2: Beispiel eines Property Eintrag in der PLC

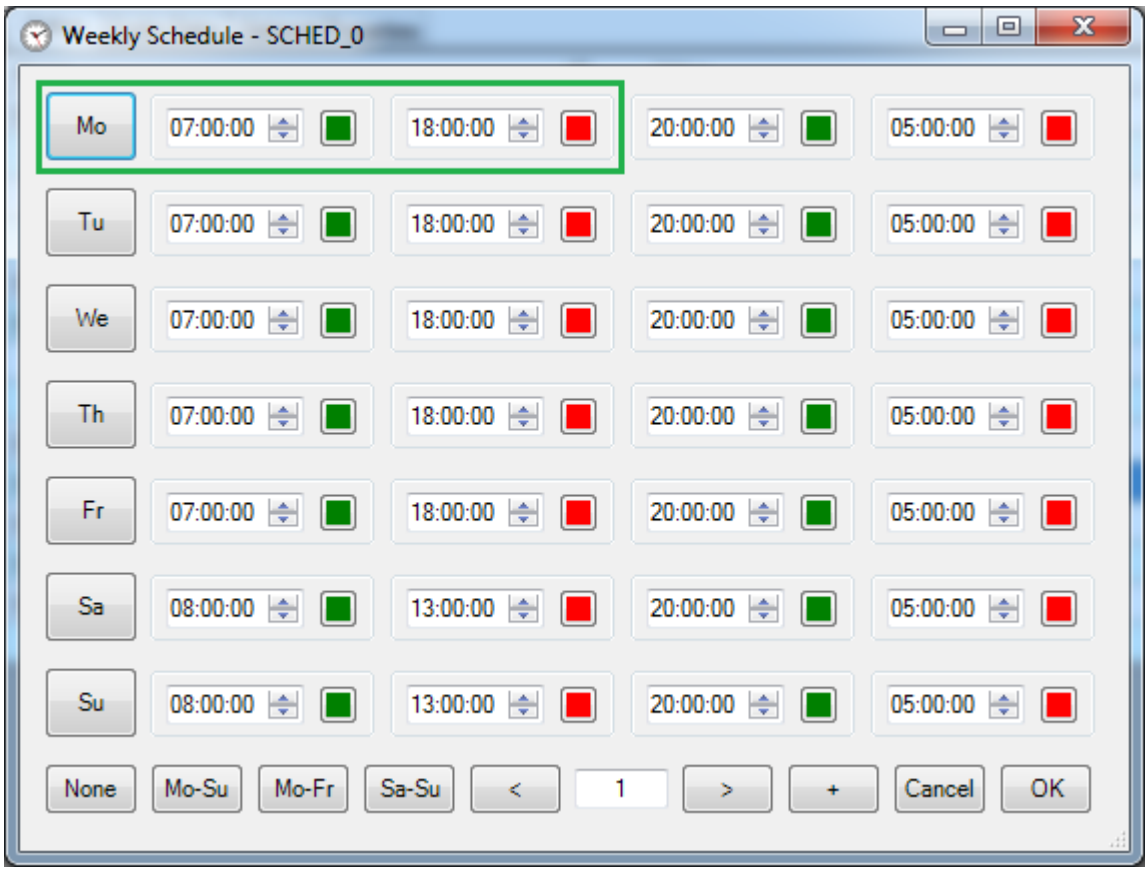


Abb. 66: Bild-3: Beispiel eines Property Eintrag im TwinCAT System Manager

Beispiel 2: Bausteinaufruf

```
0001 PROGRAM MAIN
0002 VAR
0003     Device         : FB_BACnet_Device;
0004     fbWeeklySched  : FB_BACnet_WeeklySchedulProperty;
0005     stWeeklySched  : ST_BACnet_WeeklyScheduleBool;
0006
0007     bWriteScheduleData: BOOL;
0008     bReadScheduleData: BOOL;
0009 END_VAR
0010
```

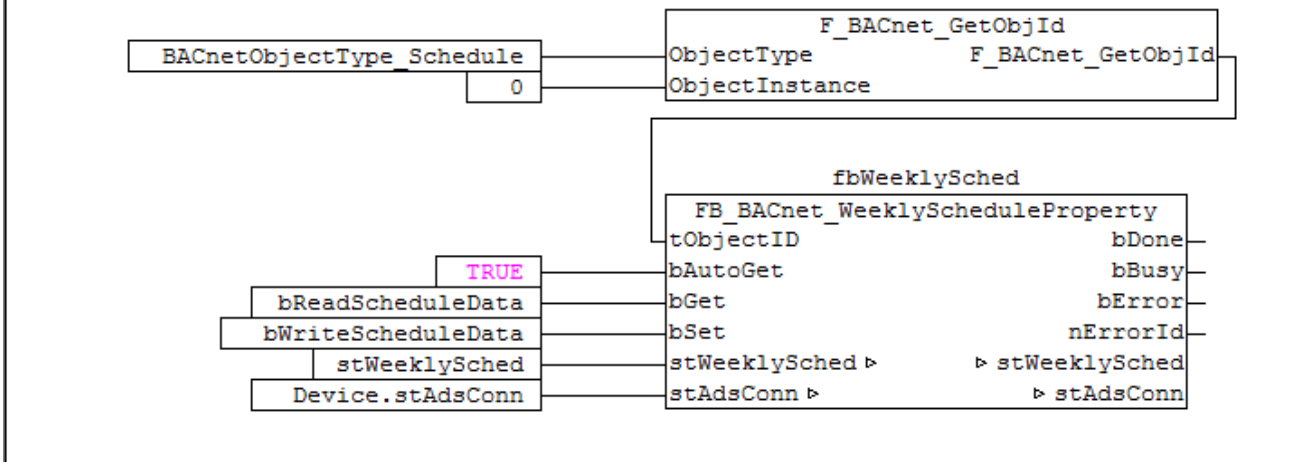
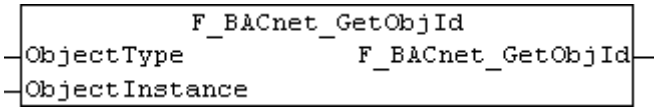


Abb. 67: Bild-4: Beispiel für die Verwendung. Lesen und Schreiben der Property Weekly_Schedule eines BACnet Schedule Objekts.

4.6 BACnet Helper

4.6.1 F_BACnet_GetObjId : DWORD



Anwendung

Funktion zur Codierung des Objekt-Typs und der Objekt-Instance in den BACnet *Object_Identifier*.

VAR_INPUT

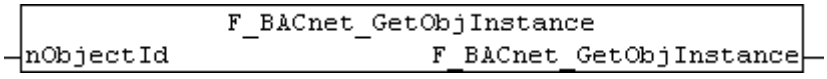
```
ObjectType      : E_BACNETOBJECTTYPE;
ObjectInstance  : UDINT;
```

ObjectType: BACnet Objekt Typ.

ObjectInstance: BACnet Objekt Nummer (Instance, 0...4'194'303).

Rückgabewert: Der resultierende BACnet *Object_Identifier*.

4.6.2 F_BACnet_GetObjInstance : UDINT



Anwendung

Funktion zu der Decodierung des BACnet *Object_Identifier* in die Objekt-Instance (Objektnummer).

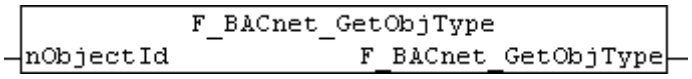
VAR_INPUT

```
nObjectId      : DWORD;
```

nObjectId: BACnet *Object_Identifier*.

Rückgabewert: Die BACnet Objekt Nummer (Instance, 0...4'194'303).

4.6.3 F_BACnet_GetObjType : E_BACNETOBJECTTYPE



Anwendung

Funktion zur Decodierung des BACnet *Object_Identifier* in den Objekt-Typ.

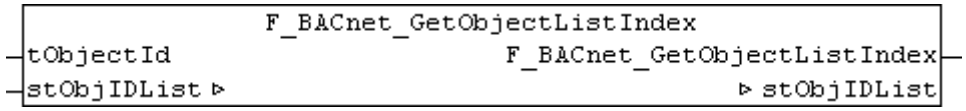
VAR_INPUT

```
nObjectId      : DWORD;
```

nObjectId: BACnet *Object_Identifier*.

Rückgabewert: Der resultierende BACnet Objekt Typ (siehe E_BACnetObjectType [▶ 344]).

4.6.4 F_BACnet_GetObjectListIndex : DINT



Anwendung

Funktion zum Feststellen der Position einer BACnet Object-ID in einer Liste mit IDs. Gibt den Index (Position) der Eingabe-Objekt-ID in einer List oder *-1* zurück. Eine BACnet Objekt-Liste liefert der Funktionsbaustein FB_BACnet_ObjectListProperty [▶ 292] für die Property *Object_List*.

VAR_INPUT

```
tObjectId    : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
```

tObjectId: Legt das Objekt fest (*Object_Identifier*: Objekt Typ und Objekt Instanz) dessen Position in der Liste ermittelt werden soll.

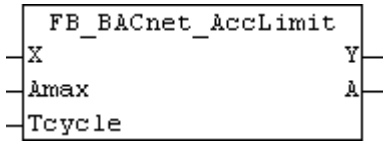
VAR_IN_OUT

```
stObjIDList      : ST_BACnet_ObjectIdentifierList;
```

stObjIDList: Liste mit BACnet Objekt IDs.

Rückgabewert: Position innerhalb der Liste (von *0* bis *Länge-1*) oder *-1*, wenn nicht vorhanden.

4.6.5 FB_BACnet_AccLimit



Anwendung

Funktionsbaustein für die Begrenzung von Signaländerung pro Zeit (maximale Beschleunigung). Die Ausgangsgröße **Y** [1] folgt der Eingangsgröße **X** [1] mit einer maximalen Beschleunigung (positiv und negativ) entsprechend des Wertes am Eingang **Amax** [1/s]. Der Eingangswert **Tcycle** gibt die aktuelle Taskzykluszeit der PLC in Sekunden an. Wird **Tcycle** auf *0* gesetzt, dann wird die aktuelle Zykluszeit intern ermittelt.

Am Ausgang **A** [1/s] wird die aktuelle Beschleunigung ausgegeben. Wird **Amax** auf *0* gesetzt, erfolgt keine Begrenzung → **Y** = **X**.

VAR_INPUT

```
X       : REAL;
Amax    : REAL;
Tcycle  : REAL;
```

X: Eingangswert [1].

Amax: Maximale Beschleunigung [1/s]; *0* = keine Begrenzung.

Tcycle: PLC Zykluszeit in Sekunden; *0* = intern ermitteln.

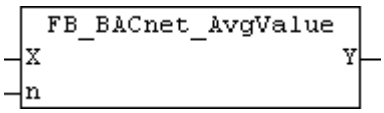
VAR_OUPUT

```
Y       : REAL;
A       : REAL;
```

Y: Begrenzter Ausgabewert [1].

A: Aktuell ermittelte Beschleunigung [1/s] des Eingangswerts **X** gegenüber Ausgabewert **Y**.

4.6.6 FB_BACnet_AvgValue



Anwendung

Funktionsbaustein zur Mittelwertbildung einer Eingangsgröße **X** über **n** Werte. Die Ausgabe erfolgt am Ausgang **Y**. Gemittelt wird bei jedem Aufruf des Bausteins. Der Mittelwert wird intern mit Hilfe eines 8-Byte Realwerts (LREAL) gebildet. Bei steigender Anzahl Mittelungen (n >>) nimmt entsprechend der Genauigkeit des resultierenden Mittelwerts ab.

VAR_INPUT

```
X       : REAL;
n       : UINT;
```

X: Zu mittelnder Eingangswert.

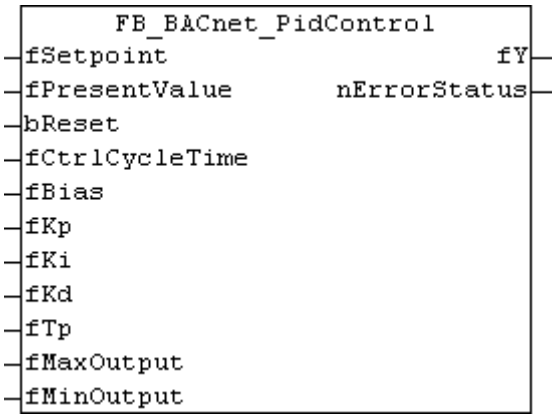
n: Anzahl Werte zu mitteln (*0* = keine Mittelwertbildung **Y** = **X**).

VAR_OUPUT

```
Y       : REAL;
```

Y: Ergebnis der Mittelwertbildung.

4.6.7 FB_BACnet_PidControl



Anwendung

PID Regelbaustein in Parallelanordnung bzw. Idealform. Der Regelbaustein dient als Basis für das BACnet Loop Objekt (FB_BACnet_Loop [▶ 204]).

Blockdiagramm

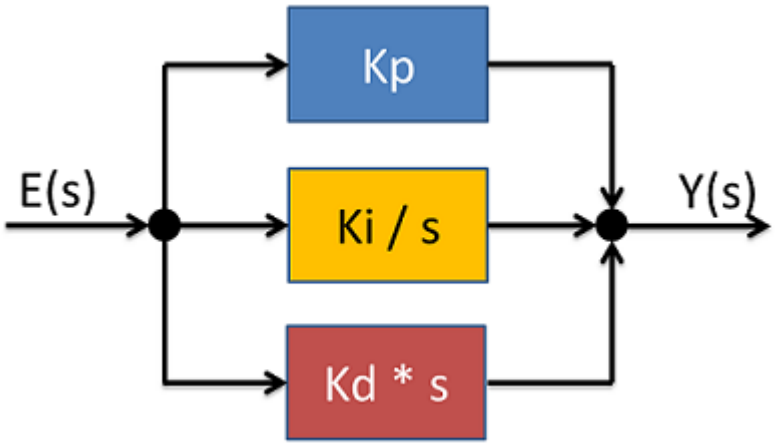


Abb. 68: Bild-1: Blockschaltbild der Übertragungsfunktion (Frequenzbereich)

Verwendung

Der Regelbaustein muss zyklisch im SPS Programm aufgerufen werden. Dabei ist zu beachten, dass die Zykluszeit im Moment des Aufruf am Eingang **fCtrlCycleTime** möglichst genau übergeben wird. Die Zykluszeit kann z.B. aus der globalen Struktur "SystemTaskInfoArr" ermittelt werden. Unter Beispiel [▶ 307] ist eine mögliche Beschaltung des Regler dargestellt.

VAR_INPUT

```
fSetpoint        : REAL;
fPresentValue    : REAL;
bReset           : BOOL;
fCtrlCycleTime   : REAL;
fBias            : REAL;
fKp              : REAL;
fKi              : REAL;
fKd              : REAL;
fTp              : REAL:=-1.0;
fMaxOutput       : REAL;
fMinOutput       : REAL;
```

fSetpoint: Sollwerteingabe (z.B. Temperaturvorgabe eines Raumes in [°C])

fPresentValue: Istwert (z.B. Raumisttemperatur in [°C])

bReset: Setzt den Regler zurück, die Ausgabe wird für die Dauer auf **fBias** gesetzt und durch **fMaxOutput** und **fMinOutput** begrenzt.

Wenn **fMinOutput** z.B. auf 5% gesetzt ist und **fBias** auf 0%, dann ist die Ausgabe **fY** für die Dauer des Reset auf 5% gestellt!

fCtrlCycleTime: Zykluszeit mit der der Regler aufgerufen wird in Sekunden [s].

fBias: Ausgabe-Offset. Dieser Wert wird mit der Ausgabe verrechnet und die Einheit muss der Einheit der Stellgröße entsprechen (z.B. Heizleistung in [%]).

fKp: Verstärkungsfaktor P. Die Regeldifferenz wird mit P multipliziert und auf die Ausgabe addiert (Ausgabe verhält sich proportional zur Eingabeabweichung).

fKi: Integralverstärkung I. Die Regeldifferenz wird unter Berücksichtigung der Zykluszeit mit I multipliziert, mit dem vorherigen Ergebnis summiert und auf die Ausgabe addiert (Hohe Regelabweichung = schnell steigende Ausgabe).

fKd: Differenzialverstärkung D. Die Regeldifferenz wird mit der vorherigen Reglerdifferenz verrechnen, das Ergebnis unter Berücksichtigung der Zykluszeit mit D multipliziert und auf die Ausgabe addiert (Starkes Reglerschwanken = starke Reaktion).

fTp: Dampfungszeit für D Anteil (PT1) [s] (wenn **fTp***0*, dann gilt: *Tp* = **fKd** / *10*).

fMaxOutput: Obere Begrenzung des Ausgabewerts **fY**. Die Einheit muss der Einheit der Stellgröße entsprechen (z.B. Heizleistung maximal 90%).

fMinOutput: Untere Begrenzung des Ausgabewerts **fY**. Die Einheit muss der Einheit der Stellgröße entsprechen (z.B. Heizleistung minimal 5%).

VAR_OUPUT

```
fY            : REAL;
nErrorStatus  : UINT;
```

fY: Stellgröße (z.B. Heizleistung in [%]).

nErrorStatus: Fehlercode, siehe BACnet Globals [▶ 330] für eine Übersicht.

Beispiel

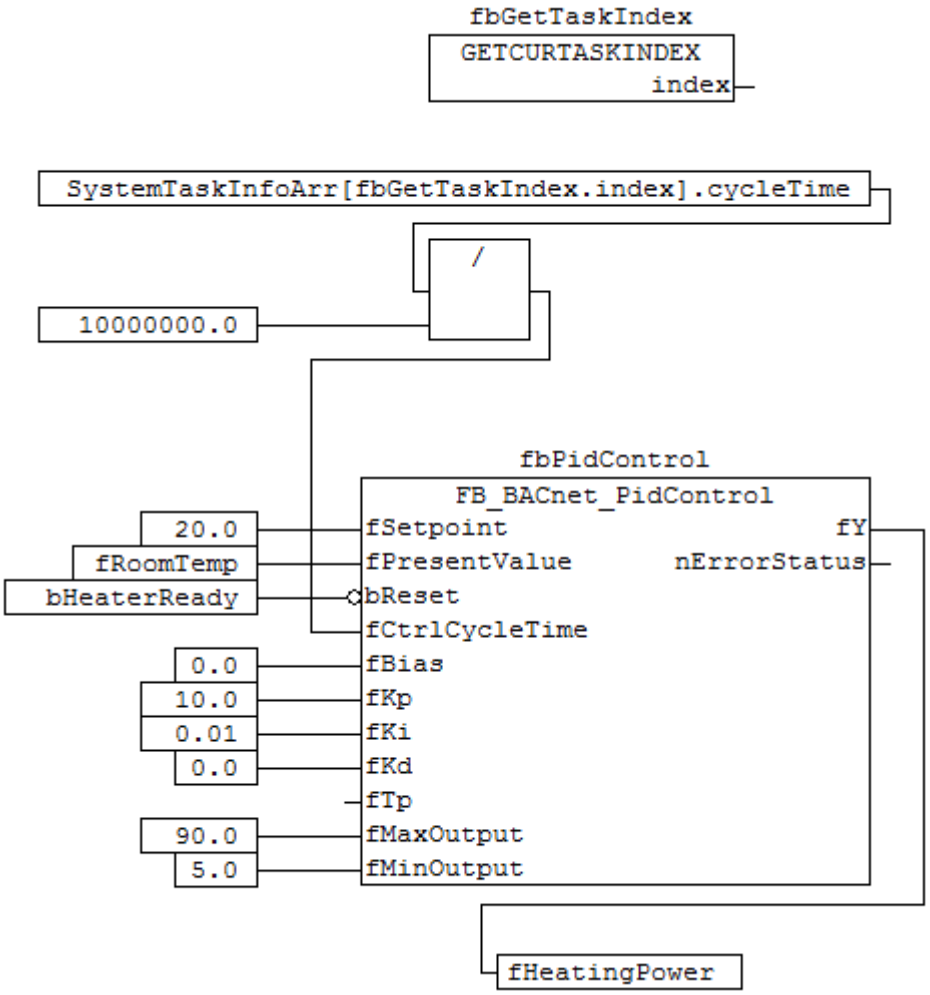


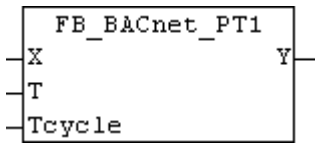
Abb. 69: Bild -2: Beispiel für die Verwendung

Anmerkung zu P = 10 --> Bei z.B. 1 K Temperaturdifferenz ergibt sich 10% Heizleistung

Anmerkung zu I = 0.01 --> Bei z.B. 1 K Temperaturdifferenz ergibt sich 0.01% Heizleistungsanstieg pro Sekunde

Anmerkung zu D = 0 --> Kein D-Anteil.

4.6.8 FB_BACnet_PT1



Anwendung

Funktionsbaustein zur Nachbildung einer Verzögerung 1. Ordnung. Durch Anreihung mehrerer PT1 Bausteine können Verzögerungen höherer Ordnung erreicht werden. Die Ausgangsgröße **Y** [1] stellt das Funktionsergebnis der Verzögerung von Eingangswert **X** [1] dar. Eingangswert **T** legt die Dämpfungszeit [s] fest (auch als τ bekannt). Wird **T** auf *0* gesetzt, erfolgt keine Verzögerung → **Y** = **X**.

Der Eingangswert **Tcycle** gibt die aktuelle Taskzykluszeit der PLC in Sekunden an. Wird **Tcycle** auf *0* gesetzt, dann wird die aktuelle Zykluszeit intern ermittelt.

VAR_INPUT

```
X       : REAL;
T       : REAL;
Tcycle  : REAL;
```

X: Eingangswert [1].

T: Dämpfungszeit [s]; *0* = keine Verzögerung.

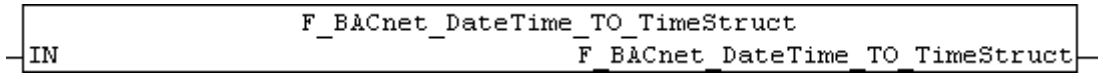
Tcycle: PLC Zykluszeit in Sekunden; *0* = intern ermitteln.

VAR_OUPUT

```
Y       : REAL;
```

Y: Verzögerter Ausgabewert [1].

4.6.9 F_BACnet_DateTime_TO_TimeStruct : TIMESTRUCT



Anwendung

Funktion zur Umwandlung eines BACnet Zeitstempels in den Datentyp *TIMESTRUCT*.

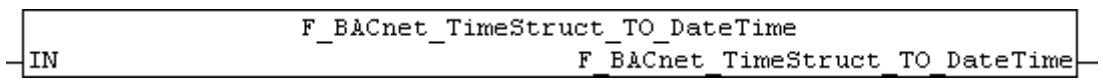
Achtung: Platzhalter (*255* → *) und Sonderwerte ("*alle geraden Monate*" oder "*letzter Tag des Monats*" etc.) dürfen nicht verwendet werden!

VAR_INPUT

```
IN      : ST_BACnet_DateTime;
```

IN: BACnet Zeitstempel als Eingangsgröße.

4.6.10 F_BACnet_TimeStruct_TO_DateTime : ST_BACnet_DateTime



Anwendung

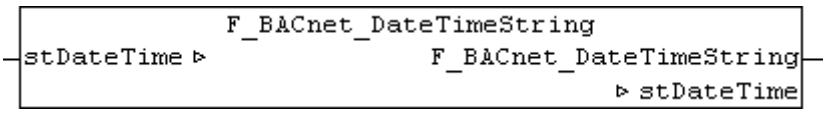
Funktion zur Umwandlung eins Zeitstempels vom Datentype *TIMESTRUCT* in einen BACnet Zeitstempel.

Achtung: Die Zeitstempelgenauigkeit verringert sich dabei von 1/1000 ms in 1/100 ms. Ausschließlich Jahreszahlen von 1900 bis 2154 werden unterstützt!

VAR_INPUT

```
IN      : TIMESTRUCT;
```

4.6.11 F_BACnet_DateTimeString : STRING(19)



Anwendung

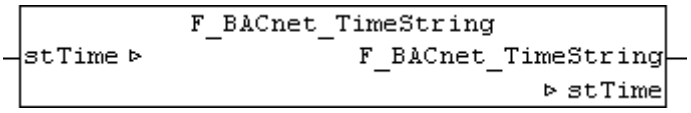
Funktion zur Zeichenkettendarstellung eines BACnet Zeitstempels. Beispiel: "*07.01.2015 11:12:20*" oder mit Platzhalter: "*07.01.**** 10:**:30*".

VAR_IN_OUT

```
stDateTime      : ST_BACnet_DateTime;
```

stDateTime: BACnet Zeitstempel zur Umwandlung in einen String.

4.6.12 F_BACnet_TimeString : STRING(12)



Anwendung

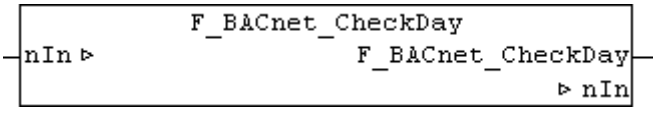
Funktion zur Zeichenkettendarstellung eines BACnet Zeitstempels. Beispiel: "*11:12:20.00*" oder mit Platzhalter: "*10:**:30.70*".

VAR_IN_OUT

```
stTime      : ST_BACnet_Time;
```

stTime: BACnet Zeitstempel zur Umwandlung in einen String.

4.6.13 F_BACnet_CheckDay : USINT



Anwendung

Funktion zur Prüfung eines Datum-Zahlenwertes (BYTE) für den Tag des Monats auf Gültigkeit. Der Rückgabewert entspricht einem gültigen Wert für den Tag des Monats oder undefiniert als Zahlencode (*). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime [▶ 359].

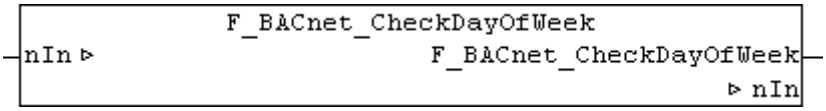
Achtung: Der Eingabe-Wert wird ebenfalls korrigiert (**nIn**).

VAR_IN_OUT

```
nIn      : BYTE;
```

nIn: Zu korrigierender Zahlenwert für die Angabe des Tages (1...31 → Tag des Monats; 32→ letzter Tag des Monats; sonst konvertiert zu 255 → undefiniert)

4.6.14 F_BACnet_CheckDayOfWeek : USINT



Anwendung

Funktion zur Prüfung eines Datum-Zahlenwertes (BYTE) für den Wochentag auf Gültigkeit. Der Rückgabewert entspricht einem gültigen Wert für den Tag des Woche oder undefiniert als Zahlencode (*). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime [▶ 359].

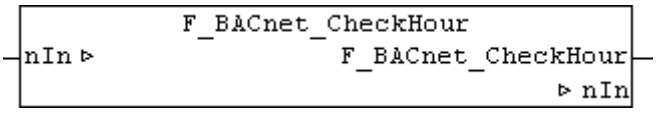
Achtung: Der Eingabe-Wert wird ebenfalls korrigiert (**nIn**).

VAR_IN_OUT

```
nIn      : BYTE;
```

nIn: Zu prüfender Zahlenwert für die Angabe des Wochentags (gültig 1...7; sonst konvertiert zu 255 → undefiniert)

4.6.15 F_BACnet_CheckHour : USINT



Anwendung

Funktion zur Prüfung eines Uhrzeit-Zahlenwertes (BYTE) für die Stunden-Angabe auf Gültigkeit. Der Rückgabewert entspricht einem gültigen Wert für die Stunden-Angabe oder undefiniert als Zahlencode (*). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime.

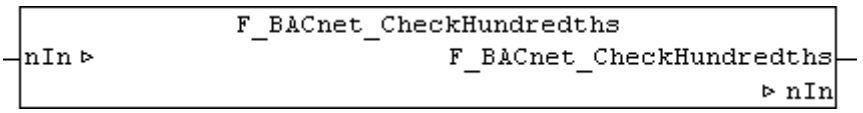
Achtung: Der Eingabe-Wert wird ebenfalls korrigiert (**nIn**).

VAR_IN_OUT

```
nIn      : BYTE;
```

nIn: Zu prüfender Zahlenwert für die Angabe der Stunde (gültig 0...23; sonst konvertiert zu 255 → undefiniert)

4.6.16 F_BACnet_CheckHundredths : USINT



Anwendung

Funktion zur Prüfung eines Uhrzeit-Zahlenwertes (BYTE) für die Hundertstelsekunden-Angabe auf Gültigkeit. Der Rückgabewert entspricht einem gültigen Wert für die Hundertstelsekunden-Angabe oder undefiniert als Zahlencode (*). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime.

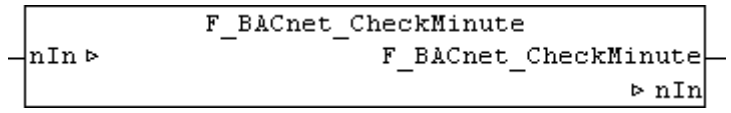
Achtung: Der Eingabe-Wert wird ebenfalls korrigiert (**nIn**).

VAR_IN_OUT

```
nIn      : BYTE;
```

nIn: Zu prüfender Zahlenwert für die Angabe der Hundertstelsekunde (gültig 0...99; sonst konvertiert zu 255 → undefiniert)

4.6.17 F_BACnet_CheckMinute : USINT



Anwendung

Funktion zur Prüfung eines Uhrzeit-Zahlenwertes (BYTE) für die Minutenangabe auf Gültigkeit. Der Rückgabewert entspricht einem gültigen Wert für die Minutenangabe oder undefiniert als Zahlencode (*). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime.

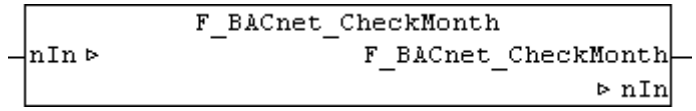
Achtung: Der Eingabe-Wert wird ebenfalls korrigiert (**nIn**).

VAR_IN_OUT

```
nIn      : BYTE;
```

nIn: Zu prüfender Zahlenwert für die Angabe der Minute (gültig 0...59; sonst konvertiert zu 255 → undefiniert)

4.6.18 F_BACnet_CheckMonth : USINT



Anwendung

Funktion zur Prüfung eines Datum-Zahlenwertes (BYTE) für den Monat auf Gültigkeit. Der Rückgabewert entspricht einem gültigen Wert für Monat oder undefiniert als Zahlencode (*). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime [▶ 359].

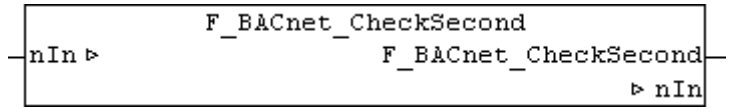
Achtung: Der Eingabe-Wert wird ebenfalls korrigiert (**nIn**).

VAR_IN_OUT

```
nIn      : BYTE;
```

nIn: Zu korrigierender Zahlenwert für die Angabe des Monats (1...12 → Monat 1=Januar; 13→ ungerade Monate; 14 → gerade Monate; sonst konvertiert zu 255 → undefiniert)

4.6.19 F_BACnet_CheckSecond : USINT



Anwendung

Funktion zur Prüfung eines Uhrzeit-Zahlenwertes (BYTE) für die Sekundenangabe auf Gültigkeit. Der Rückgabewert entspricht einem gültigen Wert für die Sekundenangabe oder undefiniert als Zahlencode (*). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime.

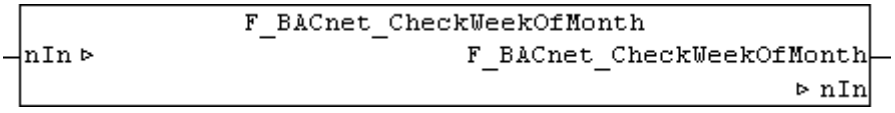
Achtung: Der Eingabe-Wert wird ebenfalls korrigiert (**nIn**).

VAR_IN_OUT

```
nIn      : BYTE;
```

nIn: Zu prüfender Zahlenwert für die Angabe der Sekunde (gültig 0...59; sonst konvertiert zu 255 → undefiniert)

4.6.20 F_BACnet_CheckWeekOfMonth : USINT



Anwendung

Funktion zur Prüfung eines Datum-Zahlenwertes (BYTE) für die Woche des Monats auf Gültigkeit. Der Rückgabewert entspricht einem gültigen Wert für die Woche des Monats oder undefiniert als Zahlencode (*). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetWeekNDay.

Achtung: Der Eingabe-Wert wird ebenfalls korrigiert (**nIn**).

VAR_IN_OUT

```
nIn      : BYTE;
```

nIn: Zu korrigierender Zahlenwert für die Angabe der Woche im Monat; mögliche Werte:

1 → Tage 1 bis 7 des Monats

2→ Tage 8 bis 14 des Monats

3 → Tage 15 bis 21 des Monats

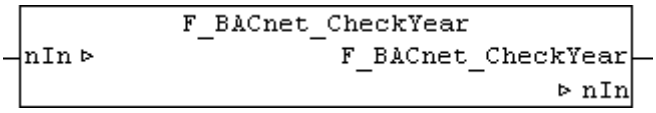
4 → Tage 22 bis 28 des Monats

5 → Tage 29 bis 31 des Monats

6 → die letzten 7 Tage des Monats

sonst konvertiert zu 255 → undefiniert

4.6.21 F_BACnet_CheckYear : USINT



Anwendung

Funktion zur Prüfung eines Datum-Zahlenwertes (BYTE) für das Jahr auf Gültigkeit. Der Rückgabewert entspricht einem gültigen Wert für ein Jahr oder undefiniert als Zahlencode (*). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime [▶ 359].

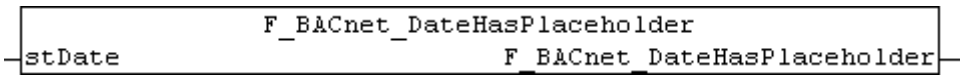
Achtung: Der Eingabe-Wert wird ebenfalls korrigiert (**nIn**).

VAR_IN_OUT

```
nIn      : BYTE;
```

nIn: Zu korrigierender Zahlenwert für die Angabe des Jahres (Jahreszahl minus 1900 → 0 bis 254 entspricht den Jahren 1900 bis 2154; 255 → undefiniert)

4.6.22 F_BACnet_DateHasPlaceholder : BOOL



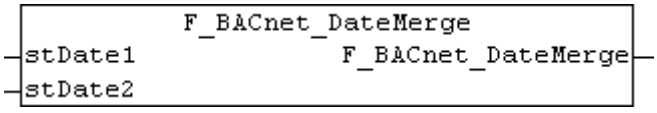
Anwendung

Funktion zur Prüfung auf Platzhalter (255 → undefiniert) in einer Datumsangabe. Rückgabewert *TRUE* → mindestens ein Platzhalter gefunden / *FALSE* → Keine Platzhalter.

VAR_INPUT

```
stDate      : ST_BACnet_Date;
```

4.6.23 F_BACnet_DateMerge : ST_BACnet_Date



Anwendung

Funktion für das Zusammenführen von 2 Zeitstempeln. Es werden jeweils die Platzhalter des einen Zeitstempels durch den Wert aus dem zweiten Zeitstempel ersetzt. Wenn in beiden Zeitstempeln kein Platzhalter vorhanden ist, dann wird der Wert des Zeitstempels aus **stDate1** bevorzugt.

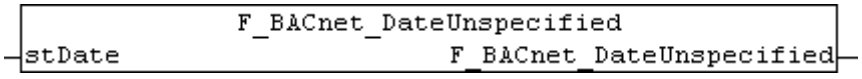
VAR_INPUT

```
stDate1      : ST_BACnet_Date;
stDate2      : ST_BACnet_Date;
```

stDate1: Zeitstempel 1 (bevorzugt, wenn kein Platzhalter vorhanden ist)

stDate2: Zeitstempel 2.

4.6.24 F_BACnet_DateUnspecified : BOOL



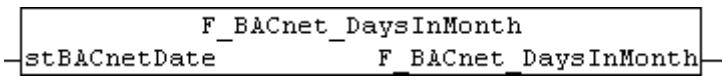
Anwendung

Funktion zur Prüfung auf Platzhalter (*255* → undefiniert) in einer Datumsangabe. Rückgabewert *TRUE* = Sämtliche Werte sind undefiniert / *FALSE* = mindestens ein Wert wurde spezifiziert.

VAR_INPUT

```
stDate      : ST_BACnet_Date;
```

4.6.25 F_BACnet_DaysInMonth : UDINT



Anwendung

Funktion zur Berechnung der Anzahl Tage im gegebenen Monat eines Jahres. Rückgabewert *16#FFFFFFFF* bedeutet Monat und/oder Jahr ist nicht spezifiziert. Schaltjahre werden berücksichtig.

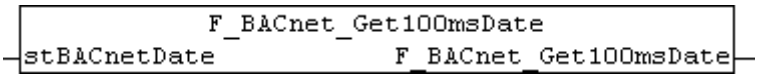
Achtung: Platzhalter (*255* → *) und Sonderwerte ("*alle geraden Monate*") dürfen nicht für Jahr- bzw. Monatsangabe verwendet werden!

VAR_INPUT

```
stBACnetDate      : ST_BACnet_Date;
```

stBACnetDate: BACnet Zeitstempel als Eingangswert. Jahr und Monat müssen gültig sein.

4.6.26 F_BACnet_Get100msDate : UDINT



Anwendung

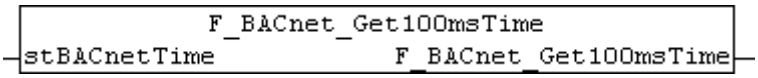
Funktion zur Berechnung des Zeitstempels in 100ms Schritten seit 1900.

VAR_INPUT

```
stBACnetDate      : ST_BACnet_Date;
```

stBACnetDate: BACnet Zeitstempel als Eingangswert.

4.6.27 F_BACnet_Get100msTime : UDINT



Anwendung

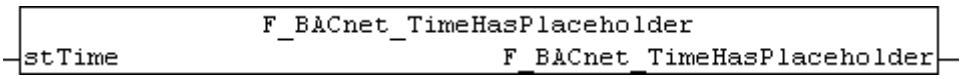
Funktion zur Berechnung des Zeitstempels in 100ms Schritten seit 00:00:00.0 Uhr.

VAR_INPUT

```
stBACnetTime      : ST_BACnet_Time;
```

stBACnetTime: BACnet Zeitstempel als Eingangswert.

4.6.28 F_BACnet_TimeHasPlaceholder : BOOL



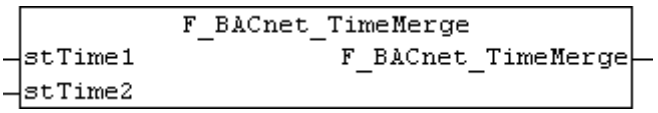
Anwendung

Funktion zur Prüfung auf Platzhalter (*255* → undefiniert) in einer Zeitangabe. Rückgabewert *TRUE* → mindestens ein Platzhalter gefunden / *FALSE* → Keine Platzhalter.

VAR_INPUT

```
stTime      : ST_BACnet_Time;
```

4.6.29 F_BACnet_TimeMerge : ST_BACnet_Time



Anwendung

Funktion für das Zusammenführen von 2 Zeitstempeln. Es werden jeweils die Platzhalter des einen Zeitstempels durch den Wert aus dem zweiten Zeitstempel ersetzt. Wenn in beiden Zeitstempeln kein Platzhalter vorhanden ist, dann wird der Wert des Zeitstempels aus **stTime1** bevorzugt.

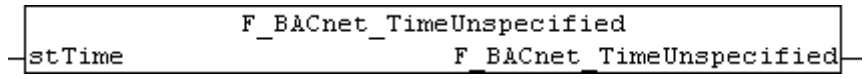
VAR_INPUT

```
stTime1      : ST_BACnet_Time;
stTime2      : ST_BACnet_Time;
```

stTime1: Zeitstempel 1 (bevorzugt, wenn kein Platzhalter vorhanden ist)

stTime2: Zeitstempel 2.

4.6.30 F_BACnet_TimeUnspecified : BOOL



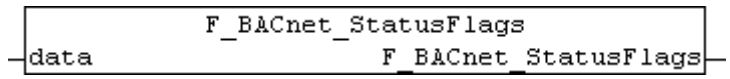
Anwendung

Funktion zur Prüfung auf Platzhalter (*255* → undefiniert) in einer Zeitangabe. Rückgabewert *TRUE* = Sämtliche Werte sind undefiniert / *FALSE* = mindestens ein Wert wurde spezifiziert.

VAR_INPUT

```
stTime       : ST_BACnet_Time;
```

4.6.31 F_BACnet_StatusFlags : ST_BACnet_StatusFlags



Anwendung

Funktion zum Decodieren des Prozessdatums der Property *Status_Flags* eines BACnet Objekts. Der Rückgabewert enthält die gesetzten Flags; siehe ST_BACnet_StatusFlags [▶ 370].

VAR_INPUT

```
data       : WORD;
```

data: Wert aus dem Prozessdatum der Property mit dem BACnet Datentyp BACnetStatusFlags.

Beispiel

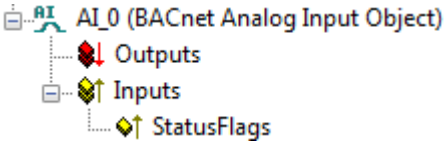
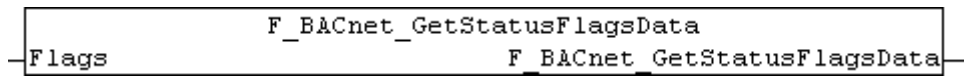


Abb. 70: Bild 1: Beispiel Prozessdatum der Property Status_Flags eines BACnet Analog Input Objekts im TwinCAT System Manager.

4.6.32 F_BACnet_GetStatusFlagsData : WORD



Anwendung

Funktion zum Codieren des Property-Werts der Property *Status_Flags* eines BACnet Objekts. Der Rückgabewert enthält den Wert für das Schreiben des Prozessdatums.

VAR_INPUT

```
Flags       : ST_BACnet_StatusFlags;
```

Beispiel

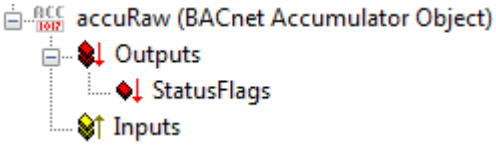
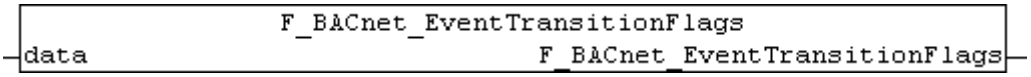


Abb. 71: Bild 1: Beispiel des Prozessdatum der Property Status_Flags eines BACnet Accumulator Objekts im TwinCAT System Manager.

4.6.33 F_BACnet_EventTransitionFlags : ST_BACnet_EventTransitionBits



Anwendung

Funktion zum Decodieren der Prozessdaten der Property *Acked_Transitions* eines BACnet Objekts. Der Rückgabewert enthält die gesetzten Flags; siehe ST_BACnet_EventTransitionBits.htm [▶ 361].

VAR_INPUT

```
data        : WORD;
```

data: Wert aus dem Prozessdatum der Property mit dem BACnet Datentyp BACnetEventTransitionBits.

Beispiel

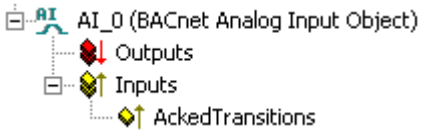
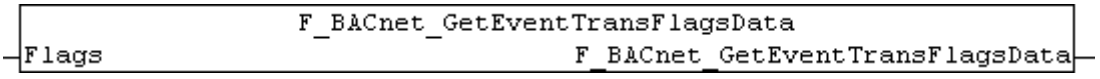


Abb. 72: Bild 1: Beispiel des Prozessdatum der Property Acked_Transitions eines BACnet Analog Input Objekts im TwinCAT System Manager.

4.6.34 F_BACnet_GetEventTransFlagsData : WORD



Anwendung

Funktion zum Codieren des Property-Werts der Property *Event_Enable* eines BACnet Objekts. Der Rückgabewert enthält den Wert für das Schreiben des Prozessdatums.

VAR_INPUT

```
Flags       : ST_BACnet_EventTransitionBits;
```

Beispiel

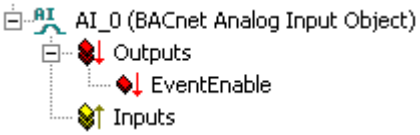
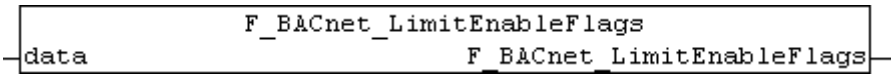


Abb. 73: Bild 1: Beispiel des Prozessdatum der Property Event_Enable eines BACnet Analog Input Objekts im TwinCAT System Manager.

4.6.35 F_BACnet_LimitEnableFlags : ST_BACnet_LimitEnable



Anwendung

Funktion zum Decodieren des Prozessdatums der Property *Limit_Enable* eines BACnet Objekts. Der Rückgabewert enthält die gesetzten Flags; siehe ST_BACnet_LimitEnable [▶ 365].

VAR_INPUT

```
data      : WORD;
```

data: Wert aus dem Prozessdatum der Property mit dem BACnet Datentyp BACnetLimitEnable.

Beispiel

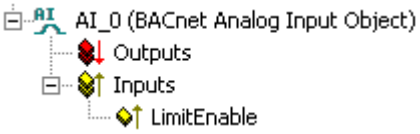
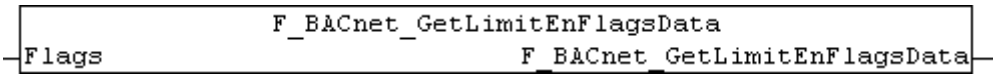


Abb. 74: Bild 1: Beispiel des Prozessdatums der Property Limit_Enable eines BACnet Analog Input Objekts im TwinCAT System Manager.

4.6.36 F_BACnet_GetLimitEnFlagsData : WORD



Anwendung

Funktion zum Codieren des Property-Werts der Property *Limit_Enable* eines BACnet Objekts. Der Rückgabewert enthält den Wert für das Schreiben des Prozessdatums.

VAR_INPUT

```
Flags      : ST_BACnet_LimitEnable;
```

Beispiel

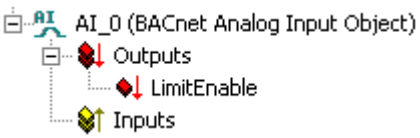
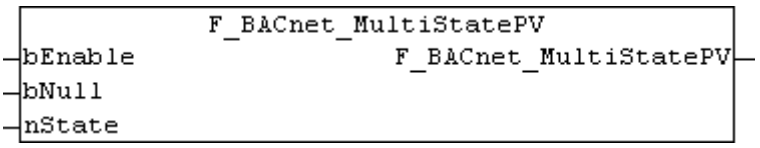


Abb. 75: Bild 1: Beispiel des Prozessdatum der Property Limit_Enable eines BACnet Analog Input Objekts im TwinCAT System Manager.

4.6.37 F_BACnet_MultiStatePV : UDINT



Anwendung

Funktion zur Umsetzung eines UDINT-Wertes der PLC in den Prozessdatenwert eines BACnet MultiState* Objekts Property *Present_Value*. Mit Hilfe dieser Funktion können z.B. BACnet MultiState* Objekte geschrieben werden, die ausschließlich über eine primitive PLC Varable (z.B.

```
nMV0 AT%Q*  : UDINT; (* ~(BACnet_ObjectType : MV : NOLINK)(BACnet_ObjectIdentifier : 0 : NOLINK)
(BACnet_PresentValue_Priority12 : : LINK) *)
```

) mit einem BACnet Objekt verknüpft sind.

VAR_INPUT

```
bEnable  : BOOL;
bNull    : BOOL;
nState   : UDINT;
```

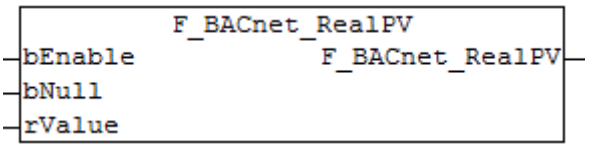
bEnable: *TRUE* = Das Prozessdatum wird aktiviert; der Wert, der sich aus **bNull** bzw. **nState** ergibt, wird in das entsprechende BACnet Object geschrieben, *FALSE* = Prozessdatum wird deaktiviert

bNull: *TRUE* = Null-Schreiben des BACnet Objekts (z.B. Löschen einer Priorität), *FALSE* = Wert aus **nState** schreiben

nState: Wert der in das BACnet Object geschrieben wird, wenn **bEnable** = *TRUE* und **bNull** = *FALSE* sind. Multi-State im Bereich [1 .. *Number_Of_States*].

Rückgabewert: Funktionsergebnis vom Typ UDINT.

4.6.38 F_BACnet_RealPV : DWORD



Anwendung

Funktion zur Umsetzung eines REAL-Wertes der PLC in den Prozessdatenwert eines BACnet Analog* Objekts Property *Present_Value*. Mit Hilfe dieser Funktion können z.B. BACnet Analog* Objekte geschrieben werden, die ausschließlich über eine primitive PLC Variable (z.B.

```
rAV0 AT%Q* : DWORD; (* ~(BACnet_ObjectType : AV : NOLINK)(BACnet_ObjectIdentifier : 0 : NOLINK)
(BACnet_PresentValue_Priority12 : : LINK) *)
```

) mit einem BACnet Objekt verknüpft sind.

Der Prozessdatentyp DWORD muss im Fall von BACnet Analog* Objekten verwendet werden, da auch Sonderformate des REAL Datentyps für die Codierung der Property *Present_Value* genutzt werden. Je nach Zielplattform würden Prozessdaten vom Typ REAL im Fall der Sonderformate von der Floatingpoint-Einheit u.U. nicht entsprechend behandelt. Das Mapping der Prozessdaten zwischen DWORD und REAL Typ ist unkritisch, da der Speicherbedarf mit jeweils 4 Byte identisch ist.

VAR_INPUT

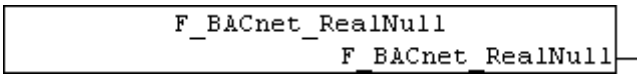
```
bEnable   : BOOL;
bNull     : BOOL;
rValue    : REAL;
```

bEnable: *TRUE* = Das Prozessdatum wird aktiviert; der Wert, der sich aus **bNull** bzw. **rValue** ergibt, wird in das entsprechende BACnet Object geschrieben, *FALSE* = Prozessdatum wird deaktiviert

bNull: *TRUE* = Null-Schreiben des BACnet Objekts (z.B. Löschen einer Priorität), *FALSE* = Wert aus **rValue** schreiben

rValue: Wert der in das BACnet Object geschrieben wird**.**

4.6.39 F_BACnet_RealNull : DWORD



Anwendung

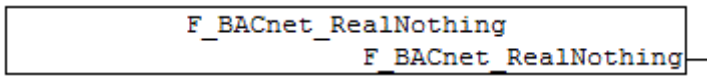
Funktion gibt den Gleitpunktwert "**N**ot **aN**umber" als DWORD codiert zurück. Damit können z.B. Prioritätseinträge im *Priority_Array* eines *Analog_Value* bzw. *Analog_Output* Objekts gelöscht werden.

Rückgabewert: "Not a number" bzw. **#QNAN** (0x7FFF0000).

4.6.40 F_BACnet_RealNothing : DWORD

Abb. 76: expand

Abb. 77: collapse

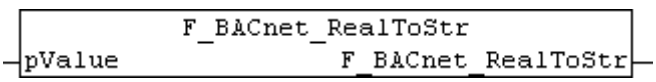


Anwendung

Funktion gibt den Gleitpunktwert "**N**ot **aN**umber" als DWORD codiert zurück. Prozessdaten der Property *Present_Value* von *AnalogValue* und -*Output* Objekten, die mit dem codierten Wert *Nothing* beschrieben sind, werden vom BACnet Stack ab Revision 12 nicht verarbeitet (*Nothing* → mache-nichts-Zustand).

Rückgabewert: "Not a number" bzw. **#QNAN** (0x7FFFFFFF).

4.6.41 F_BACnet_RealToStr : STRING



Anwendung

Funktion zur Umwandlung einer Gleitpunktzahl in eine Zeichenkette unter Berücksichtigung des Wertebereichs. Die Übergabe der REAL Wertes erfolgt mit Hilfe eines Zeigers.

VAR_INPUT

```
pValue      : POINTER TO REAL;
```

pValue: Zeiger auf die zu wandelnde Gleitpunktzahl (ADR()).

Rückgabewert: Bei endlichem Wertebereich wird die Zeichenkette der Gleitpunktzahl in dem gleichen Format ausgegeben, wie nach der Wandlung mit "REAL_TO_STRING()". Liegt der Wert im nicht-endlichen Bereich, dann wird entsprechend des Vorzeichens "#QNAN" (positive not-a-number) oder "-#QNAN" (negativ not-a-number) ausgegeben.

Beispiel

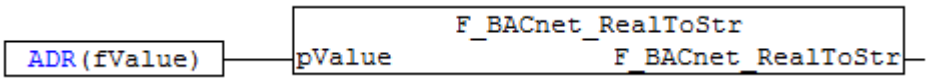
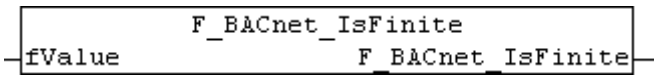


Abb. 78: Bild-1: Beispiel für die Verwendung mit Zeigerbestimmung durch "ADR()". Wobei "fValue" eine Variable vom Typ REAL darstellt.

4.6.42 F_BACnet_IsFinite : BOOL



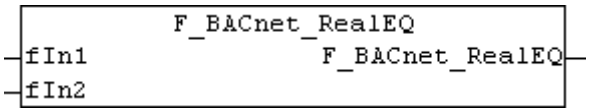
Anwendung

Funktion zur Prüfung auf Endlichkeit eines REAL Werts.

VAR_INPUT

```
fValue      : REAL;
```

4.6.43 F_BACnet_RealEQ : BOOL



Anwendung

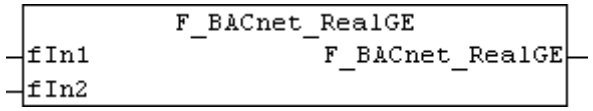
Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs. Diese Funktion sollte anstelle der Standardoperatoren ("=" bzw. "EQ") bei Werten aus BACnet verwendet werden, da sonst ein SPS Stopp ausgelöst wird, wenn die zu vergleichenden Werte nicht im endlichen Bereich liegen (z.B. Not-a-Number Werte).

VAR_INPUT

```
fIn1        : REAL;
fIn2        : REAL;
```

Rückgabewert: *TRUE* = **fIn1** und **fIn2** sind identisch oder beide liegen nicht im endlichen Wertebereich. *FALSE* = **fIn1** und **fIn2** unterscheiden sich.

4.6.44 F_BACnet_RealGE : BOOL



Anwendung

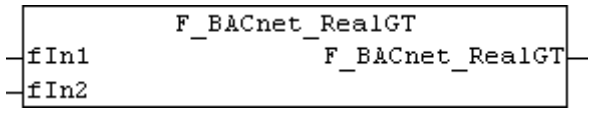
Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs. Diese Funktion sollte anstelle der Standardoperatoren (">=" bzw. "GE") bei Werten aus BACnet verwendet werden, da sonst ein SPS Stopp ausgelöst wird, wenn die zu vergleichenden Werte nicht im endlichen Bereich liegen (z.B. Not-a-Number Werte).

VAR_INPUT

```
fIn1      : REAL;
fIn2      : REAL;
```

Rückgabewert: *TRUE* = **fIn1** ist größer oder gleich **fIn2**. *FALSE* = **fIn2** ist kleiner als **fIn1** oder einer der beiden Werte liegt nicht im endlichen Wertebereich.

4.6.45 F_BACnet_RealGT : BOOL



Anwendung

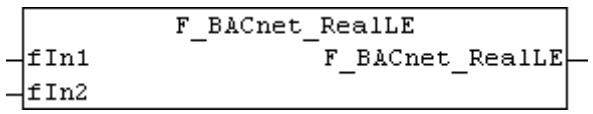
Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs. Diese Funktion sollte anstelle der Standardoperatoren (">" bzw. "GT") bei Werten aus BACnet verwendet werden, da sonst ein SPS Stopp ausgelöst wird, wenn die zu vergleichenden Werte nicht im endlichen Bereich liegen (z.B. Not-a-Number Werte).

VAR_INPUT

```
fIn1      : REAL;
fIn2      : REAL;
```

Rückgabewert: *TRUE* = **fIn1** ist größer **fIn2**. *FALSE* = **fIn2** ist kleiner oder gleich **fIn1** oder einer der beiden Werte liegt nicht im endlichen Wertebereich.

4.6.46 F_BACnet_RealLE : BOOL



Anwendung

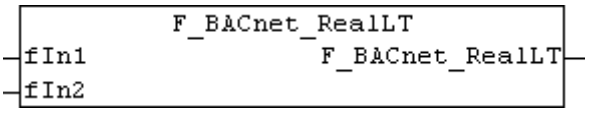
Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs. Diese Funktion sollte anstelle der Standardoperatoren ("<=" bzw. "LE") bei Werten aus BACnet verwendet werden, da sonst ein SPS Stopp ausgelöst wird, wenn die zu vergleichenden Werte nicht im endlichen Bereich liegen (z.B. Not-a-Number Werte).

VAR_INPUT

```
fIn1      : REAL;
fIn2      : REAL;
```

Rückgabewert: *TRUE* = **fIn1** ist kleiner oder gleich **fIn2**. *FALSE* = **fIn2** ist größer als **fIn1** oder einer der beiden Werte liegt nicht im endlichen Wertebereich.

4.6.47 F_BACnet_RealLT : BOOL



Anwendung

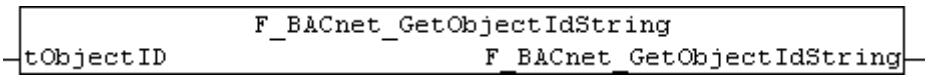
Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs. Diese Funktion sollte anstelle der Standardoperatoren ("<" bzw. "LT") bei Werten aus BACnet verwendet werden, da sonst ein SPS Stopp ausgelöst wird, wenn die zu vergleichenden Werte nicht im endlichen Bereich liegen (z.B. Not-a-Number Werte).

VAR_INPUT

```
fIn1      : REAL;
fIn2      : REAL;
```

Rückgabewert: *TRUE* = **fIn1** ist kleiner **fIn2**. *FALSE* = **fIn2** ist größer oder gleich **fIn1** oder einer der beiden Werte liegt nicht im endlichen Wertebereich.

4.6.48 F_BACnet_GetObjectIdString : STRING



Anwendung

Funktion zur Ausgabe der Objekt-ID eines BACnet Objekts als kurze Zeichenkette. Dabei wird der Objekt-Typ in Kurzform und die Instanz-Nummer des Objekts ausgegeben. Folgende Objekt-Kurzbezeichnungen werden unterstützt:

AI	AnalogInput	AO	AnalogOutput	AV	AnalogValue	BI	BinaryInput
BO	BinaryOutput	BV	BinaryValue	CAL	Calendar	CMD	Command
DEV	Device	EE	EventEnrollment	FILE	File	GROUP	Group
LOOP	Loop	MI	MultiStateInput	MO	MultiStateOutput	NC	NotificationClass
PROG	Program	SCHED	Schedule	AVG	Averaging	MV	MultiStateValue
TLOG	TrendLog	LP	LifeSafetyPoint	LZ	LifeSafetyZone	ACC	Accumulator
PC	PulseConverter						

Bei unbekannten Objekttypen wird der Platzhalter "???" anstelle der Typbezeichnung gesetzt.

VAR_INPUT

```
tObjectID       : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
```

tObjectID: Objekt-ID (*Object_Identifier*: Objekt Type und Objekt Instanz) der als Zeichenkette dargestellt werden soll.

Beispiel

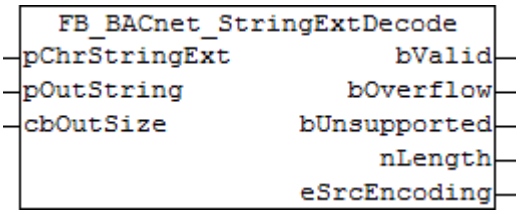
AnalogInput Objekt Nummer 0 → "AI:0"

MultiStateValue Objekt Nummer 10 → "MV:10"

Accumulator Objekt Nummer 5 → "ACC:5"

Unbekanntes Objekt Nummer 40 → "???:40"

4.6.49 FB_BACnet_StringExtDecode



Anwendung

Funktionsbaustein zum Decodieren von BACnet Strings. Unterstützt werden folgende Codings: *ASCII/ UTF-8*, *UCS2*, *UCS4* und *ISO8859-1*. Die Ausgabe erfolgt in *Windows-1252*.

VAR_INPUT

```
pChrStringExt   : POINTER TO ST_BACnet_CharacterStringExt;
pOutString      : POINTER TO T_MaxString;
cbOutSize       : UDINT;
```

pChrStringExt: BACnet Character String Datenstruktur (Property Wert) mit Header und Zeichenkette.

pOutString: Zeiger auf den Ausgabe-STRING (Windows-1252 codierte Ausgabe).

cbOutSize: Byte-Länge des Ausgabe-STRINGs (maximal beschreibbare Länge).

VAR_OUPUT

```
bValid          : BOOL;
bOverflow       : BOOL;
bUnsupported    : BOOL;
nLength         : UDINT;
eSrcEncoding    : E_BACNETSTRINGENCODINGTYPES;
```

bValid: Decodieren war erfolgreich.

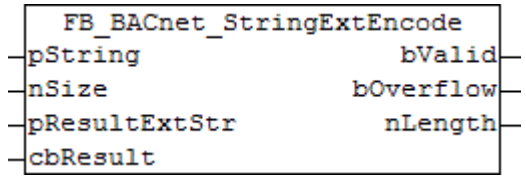
bOverflow: Eingabe-Zeichenkette ist zu lang, um auf den Ausgabe-STRING geschrieben zu werden.

bUnsupported: Die Eingabe-Zeichenkette beeinhaltet einen nicht unterstützen Zeichensatz.

nLength: Zeichen-Länge des Ausgabe-STRINGs. Die tatsächlich geschriebene Anzahl Zeichen.

eSrcEncoding: Erkannter Eingabe-Zeichensatz.

4.6.50 FB_BACnet_StringExtEncode



Anwendung

Funktionsbaustein zum Codieren von BACnet Strings. Unterstützt wird folgendes Coding: *ASCII* und *UTF-8*. Die Eingabe erfolgt in *Windows-1252*.

VAR_INPUT

```
pString         : POINTER TO T_MaxString;
nSize           : INT;
pResultExtStr   : POINTER TO ST_BACnet_CharacterStringExt;
cbResult        : UDINT;
```

pString: Zeiger auf den Eingabe-STRING (Windows-1252 codierter Text).

nSize: Zeichenlänge des Eingabe-STRINGs (LEN oder SIZEOF).

pResultExtStr: Zeiger auf die Ausgabe-Struktur (BACnet String).

cbResult: Byte-Länge der Ausgabe-Struktur (maximal beschreibare Länge in Byte, SIZEOF).

VAR_OUPUT

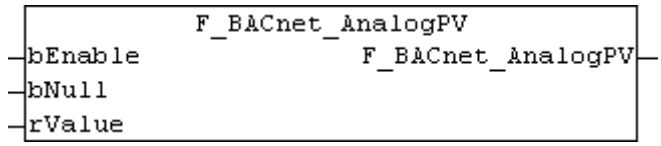
```
bValid          : BOOL;
bOverflow       : BOOL;
nLength         : UDINT;
```

bValid: Codieren war erfolgreich.

bOverflow: Eingabe-Zeichenkette ist zu lang, um auf die Ausgabe-Struktur codiert zu werden (bei der Konvertierung von *Windows-1252* [1-Byte pro Zeichen] zu *UTF-8* [1 bis 3 Byte pro Zeichen], ist der Speicherbedarf der Ausgabe u.U. wesentlich höher als der des Eingabe-STRINGs → Platzbedarf pro Zeichen variiert zwischen 1 bis 3 Byte bei *UTF-8* Coding).

nLength: Tatsächlich geschriebene Anzahl Bytes der Ausgabe-Struktur (entspricht nicht zwingend der Zeichenlänge).

4.6.51 F_BACnet_AnalogPV : REAL



Anwendung

Funktion zur Umsetzung eines REAL-Wertes der PLC in den Prozessdatenwert eines BACnet Analog* Objekts Property *Present_Value*. Mit Hilfe dieser Funktion können z.B. BACnet Analog* Objekte geschrieben werden, die ausschließlich über eine primitve PLC Varable (z.B.

```
rAV0 AT%Q* : REAL; (* ~(BACnet_ObjectType : AV : NOLINK)(BACnet_ObjectIdentifier : 0 : NOLINK)
(BACnet_PresentValue_Priority12 : : LINK) *)
```

) mit einem BACnet Objekt verknüpft sind.

VAR_INPUT

```
bEnable  : BOOL;
bNull    : BOOL;
rValue   : REAL;
```

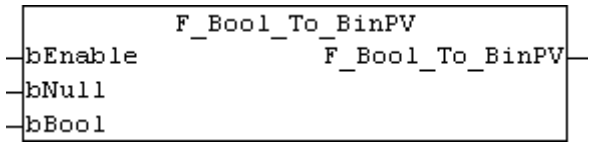
bEnable: TRUE = Das Prozessdatum wird aktiviert; der Wert, der sich aus **bNull** bzw. **rValue** ergibt, wird in das entsprechende BACnet Object geschrieben, FALSE = Prozessdatum wird deaktiviert

bNull: TRUE = Null-Schreiben des BACnet Objekts (z.B. Löschen einer Priorität), FALSE = Wert aus **rValue** schreiben

rValue: Wert der in das BACnet Object geschrieben wird, wenn **bEnable** = TRUE und **bNull** = FALSE sind

4.7 BACnet Logic

4.7.1 F_Bool_To_BinPV : E_BACNETBINARYPV



Anwendung

Funktion zur Umsetzung eines Wertes mit Datentyp BOOL der PLC in den Prozessdatenwert eines BACnet Binary* Objekts / Property *Present_Value*. Mit Hilfe dieser Funktion können z.B. BACnet Binary* Objekte geschrieben werden, die ausschließlich über eine primitve PLC Varable (z.B.

```
bBV0 AT%Q* : E_BACnetBinaryPV; (* ~(BACnet_ObjectType : BV : NOLINK)
(BACnet_ObjectIdentifier : 0 : NOLINK)(BACnet_PresentValue_Priority12 : : LINK) *)
```

) mit einem BACnet Objekt verknüpft sind.

Folgende Tabelle zeigt die Logik:

bEnable	bNull	bBool		Rückgabewert
FALSE	-	-	→	**NOTHING**
TRUE	TRUE	-	→	**NULL**
TRUE	FALSE	FALSE	→	**INACTIVE**
TRUE	FALSE	TRUE	→	**ACTIVE**

VAR_INPUT

```
bEnable  : BOOL;
bNull    : BOOL;
bBool    : BOOL;
```

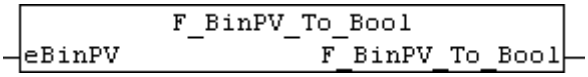
bEnable: *TRUE* = Das Prozessdatum wird aktiviert; der Wert, der sich aus **bNull** bzw. **bValue** ergibt, wird in das entsprechende BACnet Object geschrieben, *FALSE* = Prozessdatum wird deaktiviert

bNull: TRUE = Null-Schreiben des BACnet Objekts (z.B. Löschen einer Priorität), FALSE = Wert aus **bValue** schreiben

bBool: Wert der in das BACnet Object geschrieben wird, wenn **bEnable** = TRUE und **bNull** = FALSE sind

Rückgabewert: Funktionsergebnis vom Typ E_BACNETBINARYPV [▶ 335].

4.7.2 F_BinPV_To_Bool : BOOL



Anwendung

Funktion zum Konvertieren eines BACnet BinaryPV Werts in den Datentyp BOOL. Folgende Tabelle zeigt die Logik:

eBinPV		Rückgabewert
INACTIVE	→	**FALSE**
ACTIVE	→	**TRUE**
NULL	→	**FALSE**
NOTHING	→	**FALSE**

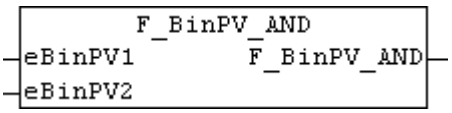
VAR_INPUT

```
eBinPV      : E_BACNETBINARYPV;
```

eBinPV: Eingangswert.

Rückgabewert: Funktionsergebnis vom Typ BOOL.

4.7.3 F_BinPV_AND : E_BACNETBINARYPV



Anwendung

Funktion zur logischen Verknüpfung von BACnet BinaryPV Werten. Folgende Tabelle zeigt die Logik:

eBinPV1	eBinPV2	Rückgabewert
INACTIVE	INACTIVE	**INACTIVE**
INACTIVE	ACTIVE	**NOTHING**
INACTIVE	NULL	**NOTHING**
INACTIVE	NOTHING	**NOTHING**
ACTIVE	INACTIVE	**NOTHING**
ACTIVE	ACTIVE	**ACTIVE**
ACTIVE	NULL	**NOTHING**
ACTIVE	NOTHING	**NOTHING**
NULL	INACTIVE	**NOTHING**
NULL	ACTIVE	**NOTHING**
NULL	NULL	**NULL**
NULL	NOTHING	**NOTHING**

eBinPV1	eBinPV2	Rückgabewert
NOTHING	INACTIVE	**NOTHING**
NOTHING	ACTIVE	**NOTHING**
NOTHING	NULL	**NOTHING**
NOTHING	NOTHING	**NOTHING**

Grundsätzlich gilt: Wenn beide Werte identisch sind, ist das Ergebnis ebenfalls identisch; sonst gibt die Funktion *NOTHING* zurück.

VAR_INPUT

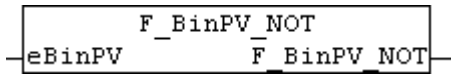
```
eBinPV1      : E_BACNETBINARYPV;
eBinPV2      : E_BACNETBINARYPV;
```

eBinPV1: Eingangswert 1.

eBinPV2: Eingangswert 2.

Rückgabewert: Funktionsergebnis vom Typ E_BACNETBINARYPV [▶ 335].

4.7.4 F_BinPV_NOT : E_BACNETBINARYPV



Anwendung

Funktion zum Invertieren eines BACnet BinaryPV Werts. Folgende Tabelle zeigt die Logik:

eBinPV	Rückgabewert
INACTIVE	**ACTIVE**
ACTIVE	**INACTIVE**
NULL	**NULL**
NOTHING	**NOTHING**

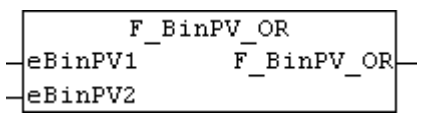
VAR_INPUT

```
eBinPV       : E_BACNETBINARYPV;
```

eBinPV: Eingangswert.

Rückgabewert: Funktionsergebnis vom Typ E_BACNETBINARYPV [▶ 335].

4.7.5 F_BinPV_OR : E_BACNETBINARYPV



Anwendung

Funktion zur logischen Verknüpfung von BACnet BinaryPV Werten. Folgende Tabelle zeigt die Logik:

eBinPV1	eBinPV2	Rückgabewert
INACTIVE	INACTIVE	**INACTIVE**
INACTIVE	ACTIVE	**ACTIVE**
INACTIVE	NULL	**INACTIVE**
INACTIVE	NOTHING	**INACTIVE**
ACTIVE	INACTIVE	**ACTIVE**
ACTIVE	ACTIVE	**ACTIVE**
ACTIVE	NULL	**ACTIVE**
ACTIVE	NOTHING	**ACTIVE**
NULL	INACTIVE	**INACTIVE**
NULL	ACTIVE	**ACTIVE**
NULL	NULL	**NULL**
NULL	NOTHING	**NULL**
NOTHING	INACTIVE	**INACTIVE**
NOTHING	ACTIVE	**ACTIVE**
NOTHING	NULL	**NULL**
NOTHING	NOTHING	**NOTHING**

Grundsätzlich gilt: Wenn einer der beiden Werte *ACTIVE* ist, dann ist das Ergebnis *ACTIVE*. Wenn einer der beiden Werte *INACTIVE* ist und der andere nicht *ACTIVE*, dann ist das Ergebnis *INACTIVE*. Wenn einer der beiden Werte *NULL* ist und der andere nicht *ACTIVE* und nicht *INACTIVE*, dann ist das Ergebnis *NULL*. Ansonsten ist das Ergebnis *NOTHING*.

VAR_INPUT

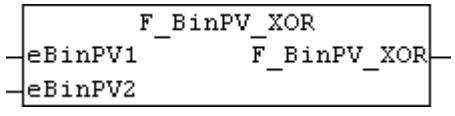
```
eBinPV1      : E_BACNETBINARYPV;
eBinPV2      : E_BACNETBINARYPV;
```

eBinPV1: Eingangswert 1.

eBinPV2: Eingangswert 2.

Rückgabewert: Funktionsergebnis vom Typ E_BACNETBINARYPV [▶ 335]

4.7.6 F_BinPV_XOR : E_BACNETBINARYPV



Anwendung

Funktion zur logischen Verknüpfung von BACnet BinaryPV Werten. Folgende Tabelle zeigt die Logik:

eBinPV1	eBinPV2	Rückgabewert
INACTIVE	INACTIVE	**NOTHING**
INACTIVE	ACTIVE	**ACTIVE**
INACTIVE	NULL	**INACTIVE**
INACTIVE	NOTHING	**INACTIVE**
ACTIVE	INACTIVE	**ACTIVE**
ACTIVE	ACTIVE	**NOTHING**
ACTIVE	NULL	**ACTIVE**
ACTIVE	NOTHING	**ACTIVE**
NULL	INACTIVE	**INACTIVE**
NULL	ACTIVE	**ACTIVE**
NULL	NULL	**NOTHING**
NULL	NOTHING	**NULL**
NOTHING	INACTIVE	**INACTIVE**
NOTHING	ACTIVE	**ACTIVE**
NOTHING	NULL	**NULL**
NOTHING	NOTHING	**NOTHING**

Grundsätzlich gilt: Wenn nur einer der beiden Werte *ACTIVE* ist, dann ist das Ergebis *ACTIVE*. Wenn nur einer der beiden Werte *INACTIVE* ist und der andere nicht *ACTIVE*, dann ist das Ergebnis *INACTIVE*. Wenn nur einer der beiden Werte *NULL* ist und der andere nicht *ACTIVE* und nicht *INACTIVE*, dann ist das Ergebnis *NULL*. Ansonsten ist das Ergebnis *NOTHING*.

VAR_INPUT

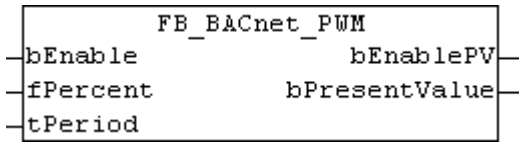
```
eBinPV1      : E_BACNETBINARYPV;
eBinPV2      : E_BACNETBINARYPV;
```

eBinPV1: Eingangswert 1.

eBinPV2: Eingangswert 2.

Rückgabewert: Funktionsergebnis vom Typ E_BACNETBINARYPV [▶ 335]

4.8 FB_BACnet_PWM



Anwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_PWM kann ein pulsierendes Ausgabesignal, mit definierter Ein- und Ausschaltdauer in Prozent der Periodendauer, erzeugt werden.

VAR_INPUT

```
bEnable         : BOOL;
fPercent        : REAL;
tPeriod         : TIME;
```

bEnable:

fPercent: 0...100%

tPeriod: PWM period time

VAR_OUPUT

```
bEnablePV       : BOOL;
bPresentValue   : BOOL;
```

bEnablePV:

bPresentValue:

4.9 BACnet Datentypen

4.9.1 BACnet_Globals

```
VAR_GLOBAL CONSTANT
  BACnet_MaxDayEntry                    := 19,
  BACnet_MaxExSchEntries                := 9,
  BACnet_MaxTimeValues                  := 9,
  BACnet_MaxObjIdList                   := 1999,
  BACnet_MaxLogBufferEntries            := 499,
  BACnet_MaxDestinationArray            := 19,

  BACnet_ERR_NO_ERROR                   := 0,
  BACnet_ERR_NO_VALID_NET_ID            := 1,
  BACnet_ERR_WRONG_MAPPING              := 2,
  BACnet_ERR_OPERATIONAL                := 3,
  BACnet_ERR_NO_ETH_LINK                := 4,
  BACnet_ERR_NO_GATEWAY                 := 5,
  BACnet_ERR_WOC_TRIGGER_UNSUP          := 6,
  BACnet_ERR_WOC_DISABLED               := 7,
  BACnet_ERR_INVALID_OBJECTID           := 8,
  BACnet_ERR_INVALID_OBJECTTYPE         := 10,
  BACnet_ERR_INVALID_CYCLETIME          := 11,
  BACnet_ERR_INVALID_PARAMETER          := 12,

  BACnet_ERR_PERSISTENT_WRITE           := 20,
  BACnet_ERR_INVALID_ADS_CONN           := 21,
  BACnet_ERR_CLIENT_TIMEOUT             := 22,
  BACnet_ERR_CLIENT_SERVICE             := 23,
  BACnet_ERR_WOC_TRIGGER_ACTIVE         := 24,
  BACnet_ERR_REVISION_READ              := 25,
  BACnet_ERR_PERSISTENT_PLC_WRITE       := 26,

  BACnet_ERR_ADS_NO_VALID_NET_ID        := 30,
  BACnet_ERR_ADS_Data                   := 31,
  BACnet_ERR_ADS_Read                   := 32,
  BACnet_ERR_ADS_Write                  := 33,
  BACnet_ERR_ADS_ObjType                := 34,
  BACnet_ERR_ADS_ObjListEmpty           := 35,
  BACnet_ERR_ADS_DeviceOP               := 36,
  BACnet_ERR_ADS_NoDevice               := 37,
  BACnet_ERR_ADS_ObjInstance            := 38,
  BACnet_ERR_ADS_AmsPort                := 39,
  BACnet_ERR_ADS_ObjectId               := 40,
  BACnet_ERR_ADS_LegacyStack            := 41,
  BACnet_ERR_ADS_ObjIdMissmatch         := 42,
  BACnet_ERR_ADS_BufferSize             := 43,
  BACnet_ERR_ADS_SchedData              := 44,
  BACnet_ERR_ADS_UnknownEventChoice     := 45,
  BACnet_ERR_ADS_UnknownEntryChoice     := 46,
  BACnet_ERR_ADS_TooManyObjects         := 47,
  BACnet_ERR_ADS_WrongEntrySize         := 48,
  BACnet_ERR_ADS_RecipientData          := 49,
  BACnet_ERR_ADS_TooManyEntries         := 50,
```

```
  BACnet_ERR_ADS_NotEnoughEntries                   := 51,
  BACnet_ERR_ADS_Encoding                           := 52,
  BACnet_ERR_ADS_NoServerClient                     := 53,
  BACnet_ERR_ADS_NoNotificationSink                 := 54,

  BACnet_ADSTimeOut                                     := t#15s,
  BACnet_PlcPersisTimeOut                               := t#2m,
  BACnet_STRING_MAXLENGTH                               := 256,

  BACnet_ADSIGRP_GET_OBJECT_LIST                        := 16#82400000,
  BACnet_ADSIGRP_CLIENT_COM_PARAMETER                   := 16#84000000,
  BACnet_ADSIGRP_DEV_R_GETADSPORTBYDEVID                := 16#80000047,
  BACnet_ADSIGRP_NTFSINK_EVENTNTF_CNT                   := 16#00000001,
  BACnet_ADSIGRP_NTFSINK_COVNTF_CNT                     := 16#00000002,
  BACnet_ADSIGRP_NTFSINK_EVENTNTF_READ                  := 16#00000003,
  BACnet_ADSIGRP_NTFSINK_COVNTF_READ                    := 16#00000004,
  BACnet_ADSIGRP_NTFSINK_EVENTNTF_DELETE                := 16#00000005,
  BACnet_ADSIGRP_NTFSINK_COVNTF_DELETE                  := 16#00000006,
  BACnet_ADSIGRP_NTFSINK_EVENTNTF_INFO                  := 16#00000007,
  BACnet_ADSIGRP_NTFSINK_COVNTF_INFO                    := 16#00000008,
  BACnet_ADSIGRP_WRITE_NTFSINK_EVENTNTF_ACK             := 16#00000009,
  BACnet_ADSIGRP_NTFSINK_W_EVENTNTF_ACK_BY_SEQUENCE     := 16#0000000A,
  BACnet_ADSIGRP_NTFSINK_W_EVENTNTF_ACK_BY_PARAMETER    := 16#0000000B,
  BACnet_ADSIGRP_NTFSINK_R_EVENTNTF_READ_BY_SEQUENCE    := 16#0000000C,
  BACnet_ADSIGRP_NTFSINK_R_COVNTF_READ_BY_SEQUENCE      := 16#0000000D,
  BACnet_ADSIGRP_NTFSINK_R_EVENTNTF_DELETE_BY_SEQUENCE  := 16#0000000E,
  BACnet_ADSIGRP_NTFSINK_R_COVNTF_DELETE_BY_SEQUENCE    := 16#0000000F
END_VAR
```

`BACnet_MaxDayEntry`: Begrenzung der maximalen Anzahl an Einträgen pro Wochentag in der Struktur ST_BACnet_WeeklyScheduleBool [▶ 372].

`BACnet_MaxExSchEntries`: Begrenzung der maximalen Anzahl an Ausnahme-Einträgen in der Struktur ST_BACnet_ExceptionScheduleBool [▶ 361].

`BACnet_MaxTimeValues`: Begrenzung der maximalen Anzahl an Zeit-Einträgen in der Struktur ST_BACnet_ExceptionScheduleEntryBool [▶ 362].

`BACnet_MaxObjIdList`: Begrenzung der maximal auslesbaren Objekt-IDs mit Hilfe des Funktionsbausteins FB_BACnet_ObjectListProperty [▶ 292].

`BACnet_MaxLogBufferEntries`: Begrenzung der maximal auslesbaren Einträge des Log-Buffers mit Hilfe des Funktionsbausteins FB_BACnet_LogBufferProperty [▶ 289].

`BACnet_MaxDestinationArray`: Begrenzung der maximal schreib-/lesbaren Einträge einer Recipient-List mit Hilfe des Funktionsbausteins FB_BACnet_RecipientListProperty [▶ 296].

`BACnet_ERR_NO_ERROR`: Fehlercode: Kein Fehler.

`BACnet_ERR_NO_VALID_NET_ID`: Fehlercode: Keine gültige AMS-NetID verfügbar (mögliche Ursache: der Funktionsbaustein FB_BACnet_Adapter [▶ 156] wurde nicht vollständig im TwinCAT System Manager verknüpft; Lösung: Mapping der Prozessdaten im TwinCAT System Manager überprüfen, eventuell neu verknüpfen bzw. erneutes PLC Automapping ausführen).

`BACnet_ERR_WRONG_MAPPING`: Fehlercode: Fehlerhaftes Prozessdatenmapping (mögliche Ursache: der entsprechende Baustein wurde nicht vollständig im TwinCAT System Manager verknüpft; Lösung: Mapping der Prozessdaten im TwinCAT System Manager überprüfen, eventuell neu verknüpfen bzw. erneutes PLC Automapping ausführen).

`BACnet_ERR_OPERATIONAL`: Fehlercode: Der verbundene BACnet Server / Client (FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243]) befindet sich nicht im Zustand "Operational" (mögliche Ursache lokal/remote: Der Baustein FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243] wird nicht zyklisch im PLC Programm aufgerufen; Lösung: Die Instanz von FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243] mindestens einmal pro PLC Programmzyklus aufrufen; mögliche Ursache remote: Die Verbindung zum entfernten Server ist unterbrochen; Maßnahmen: Die Verbindung überprüfen; die Ursache für den Unterbruch kann u.U. mit Hilfe der Diagnose [▶ 79]-Daten ermittelt werden).

BACnet_ERR_NO_ETH_LINK: Fehlercode: Der Netzwerkadapter meldet eine fehlende Netzwerkverbindung (mögliche Ursache: das Netzwerkkabel wurde entfernt bzw. die Gegenstelle ist nicht verbunden oder ohne Stromversorgung; Maßnahmen: Netzwerkstecker, -verbindung und Gegenstelle überprüfen).

BACnet_ERR_NO_GATEWAY: Fehlercode: Das konfigurierte Gateway ist nicht verfügbar (mögliche Ursachen: die Gateway-Konfiguration ist fehlerhaft; das Gateway ist durch mangelnde Netzwerkverbindung nicht erreichbar; das Gateway ist defekt/abgeschaltet; Maßnahmen: Konfiguration, Netzwerkverbindung und Erreichbarkeit des Gateways überprüfen).

BACnet_ERR_WOC_TRIGGER_UNSUP: Fehlercode: Write-On-Change Trigger wird vom verwendeten BACnet-Treiber nicht unterstützt.

BACnet_ERR_WOC_DISABLED: Fehlercode: Write-On-Change konnte nicht getriggert werden, da Write-On-Change deaktiviert ist.

BACnet_ERR_INVALID_OBJECTID: Fehlercode: Ungültige Objekt-ID erkannt (mögliche Ursachen: die Property object_identifier wurde nicht richtig im TwinCAT System Manager verknüpft; der verwendete Objekttyp passt nicht zum Objekt-Typ, der in die Objekt-ID codiert ist; Maßnahmen: Mapping im TwinCAT System Manager überprüfen/erneuern bzw. PLC Automapping erneut ausführen; Typ des verknüpften Objekts im TwinCAT System Manager und verwendeten Funktionsbausteintyp im PLC-Programm vergleichen).

BACnet_ERR_INVALID_OBJECTTYPE: Fehlercode: siehe **BACnet_ERR_INVALID_OBJECTID**.

BACnet_ERR_INVALID_CYCLETIME: Fehlercode: Ungültige PLC-Zykluszeit erkannt. Die Zykluszeit darf nicht kleiner/gleich Null sein.

BACnet_ERR_INVALID_PARAMETER: Fehlercode: Die übergebenen Parameter sind nicht korrekt.

BACnet_ERR_PERSISTENT_WRITE: Fehlercode: Persistenten Daten des BACnet-Treibers konnten nicht geschrieben werden (mögliche Ursache könnte mangelnder Speicherplatz oder zu geringer Router-Speicher sein).

BACnet_ERR_INVALID_ADS_CONN: Fehlercode: Die ADS-Verbindung zum Server/Client konnte nicht hergestellt werden bzw. ist fehlerhaft konfiguriert.

BACnet_ERR_CLIENT_TIMEOUT: Fehlercode: Die Verbindung zu einem entfernten Server (Client) wurde unterbrochen (siehe Diagnose [▶ 79]-Daten).

BACnet_ERR_CLIENT_SERVICE: Fehlercode: Ein Dienst eines entfernten Servers (Client) konnte nicht ausgeführt werden (mögliche Ursachen: ein unter einem Client konfiguriertes Objekt existiert auf dem entfernten Server nicht; ein entfernter Server ungestützt z.B. COV-P nicht, dies wurde jedoch unter dem Client konfiguriert; siehe Diagnose [▶ 79]-Daten).

BACnet_ERR_WOC_TRIGGER_ACTIVE: Fehlercode: WOC-Trigger ist aktiv, während am Eingang eine weitere Flanke erkannt wurde.

BACnet_ERR_REVISION_READ: Fehlercode: Die Revisionsnummer des BACnet-Treibers konnte nicht ausgelesen werden.

BACnet_ERR_PERSISTENT_PLC_WRITE: Fehlercode: Persistenten Daten der PLC konnten nicht geschrieben werden (mögliche Ursache könnte mangelnder Speicherplatz oder zu geringer Router-Speicher sein).

BACnet_ERR_ADS_NO_VALID_NET_ID: Fehlercode: Keine gültige AMS-NetID. Siehe auch **BACnet_ERR_NO_VALID_NET_ID**.

BACnet_ERR_ADS_Data: Fehlercode: Die empfangene Datenlänge entspricht nicht der zu erwartenden Datenlänge.

BACnet_ERR_ADS_Read: Fehlercode: Am internen Baustein ADSREAD liegt ein Fehler vor (siehe Fehlerausgang der unterlagerten ADSREAD- bzw. FW_AdsRead-Instanz für eine Fehlerbeschreibung).

BACnet_ERR_ADS_Write: Fehlercode: Am internen Baustein ADSWRITE liegt ein Fehler vor (siehe Fehlerausgang der unterlagerten ADSWRITE- bzw. FW_AdsWrite-Instanz für eine Fehlerbeschreibung).

`BACnet_ERR_ADS_ObjType`: Fehlercode: keine Verwendung.

`BACnet_ERR_ADS_ObjListEmpty`: Fehlercode: Die empfangene Objektliste enthält keinen Eintrag. Mindestens ein Objekt pro BACnet-Server wird jedoch erwartet (Device Objekt).

`BACnet_ERR_ADS_DeviceOP`: Fehlercode: Die ADS Verbindung zum Server/Client kann nicht aufgebaut werden, da das zugehörige Device-Objekt nicht "Operational" meldet (siehe FB_BACnet_Device [▶ 199] bzw. FB_BACnet_RemoteDevice [▶ 243] Status).

`BACnet_ERR_ADS_NoDevice`: Fehlercode: Beim Auslesen der Objekt-Liste des Device-Objekts wurde kein Device-Objekt gefunden.

`BACnet_ERR_ADS_ObjInstance`: Fehlercode: Beim Auslesen der Objekt-Liste des Device-Objekts wurde eine fehlerhaft codierte Objektinstanz erkannt.

`BACnet_ERR_ADS_AmsPort`: Fehlercode: Der konfigurierte AMS Port liegt nicht im gültigen Bereich (gültig: >= 1000).

`BACnet_ERR_ADS_ObjectId`: Fehlercode: siehe BACnet_ERR_INVALID_OBJECTID [▶ 330].

`BACnet_ERR_ADS_LegacyStack`: Fehlercode: Der verwendetet Dienst wird von dem verwendeten BACnet-Treiber nicht unterstützt (BACnet-Treiber der Revision 6 wurde erkannt).

`BACnet_ERR_ADS_ObjIdMissmatch`: Fehlercode: Die Objektinstanz der Objektliste des Device Objekts stimmt nicht mit der verknüpften Device-ID überein (mögliche Ursache: Fehlerhaftes Mapping im TwinCAT System Manager; Lösung: Mapping des Device-Objekts und des BACnet Adapters im TwinCAT System Manager überprüfen; Eventuell das PLC Automapping erneut ausführen).

`BACnet_ERR_ADS_BufferSize`: Fehlercode: Die zu lesenden Daten übersteigen die Größe des globalen ADS Puffer in der PLC (Standard: ca. 8 kByte). Die Größe des globalen ADS Puffer in der PLC kann in der Struktur ST_BACnet_GlobalAdsBuffer [▶ 363] angepasst werden.

`BACnet_ERR_ADS_SchedData`: Fehlercode: Fehlerhaft codierte Daten beim Auslesen der Property weekly_schedule erkannt.

`BACnet_ERR_UnknownEventChoice`: Fehlercode: Unbekannter Event-Typ beim Auslesen der Property exception_schedule erkannt (bekannt: Kalender-Referenz und Kalender-Eintrag).

`BACnet_ERR_UnknownEntryChoice`: Fehlercode: Unbekannter Entry-Typ beim Auslesen der Property exception_schedule erkannt (bekannt: Datum, Datumsbereich und Wochentag).

`BACnet_ERR_ADS_TooManyObjects`: Fehlercode: Zu viele Objekte in der Objektliste. Die Objektliste konnte nur teilweise ausgelesen werden.

`BACnet_ERR_ADS_WrongEntrySize`: Fehlercode: Der empfangene Eintrag (Text oder Daten) sind länger/kürzer als erwartet.

`BACnet_ERR_ADS_RecipientData`: Fehlercode: Die zu schreibenden Daten der Property recipient_list sind fehlerhaft (mögliche Ursache: die Empfänger-Objekt-ID entspricht nicht dem Typ Device-Objekt).

`BACnet_ERR_ADS_TooManyEntries`: Fehlercode: Die gelesenen Einträge übersteigen die maximal zu erwartenden Einträge (mögliche Ursache: beim Auslesen der Property event_message_texts wurden mehr als 3 Einträge gelesen - maximal 3 sind jedoch nur zulässig).

`BACnet_ERR_ADS_NotEnoughEntries`: Fehlercode: Die gelesenen Einträge erreichen nicht die minimal zu erwartenden Einträge (mögliche Ursache: beim Auslesen der Property event_message_texts wurden weniger als 3 Einträge gelesen - minimal 3 müssen vorhanden sein).

`BACnet_ERR_ADS_Encoding`: Fehlercode: Das verwendete String-Encoding wird nicht unterstützt bzw. es wurden Fehler im Encoding festgestellt.

`BACnet_ERR_ADS_NoServerClient`: Fehlercode: Die verwendete ADS-Verbindung stellt weder einen lokalen Server noch einen Client dar. Client oder Server wird jedoch erwartet (siehe ST_BACnet_AdsConnection [▶ 358]).

`BACnet_ERR_ADS_NoNotificationSink`: Fehlercode: Die verwendete ADS-Verbindung stellt keine Notification-Sink dar. Eine Notification-Sink wird jedoch erwartet (siehe ST_BACnet_AdsConnection [▶ 358]).

`BACnet_ADSTimeOut`: Zeitkonstante: Maximale Wartezeit bei einer ADS-Verbindung in Millisekunden (`TIME`, standard `T#15s`).

`BACnet_PlcPersisTimeOut`: Zeitkonstante: Maximale Wartezeit beim Schreiben der persistenten Daten der PLC in Millisekunden (`TIME`, standard `T#2m`).

`BACnet_STRING_MAXLENGTH`: Zeichenlänge: Maximale Länge einer BACnet-Zeichenkette in Byte (INT, standard 256).

`BACnet_ADSIGRP_GET_OBJECT_LIST`: Index-Group Konstante.

`BACnet_ADSIGRP_CLIENT_COM_PARAMETER`: Index-Group Konstante.

`BACnet_ADSIGRP_DEV_R_GETADSPORTBYDEVID`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_EVENTNTF_CNT`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_COVNTF_CNT`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_EVENTNTF_READ`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_COVNTF_READ`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_EVENTNTF_DELETE`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_COVNTF_DELETE`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_EVENTNTF_INFO`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_COVNTF_INFO`: Index-Group Konstante.

`BACnet_ADSIGRP_WRITE_NTFSINK_EVENTNTF_ACK`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_W_EVENTNTF_ACK_BY_SEQUENCE`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_W_EVENTNTF_ACK_BY_PARAMETER`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_R_EVENTNTF_READ_BY_SEQUENCE`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_R_COVNTF_READ_BY_SEQUENCE`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_R_EVENTNTF_DELETE_BY_SEQUENCE`: Index-Group Konstante.

`BACnet_ADSIGRP_NTFSINK_R_COVNTF_DELETE_BY_SEQUENCE`: Index-Group Konstante.

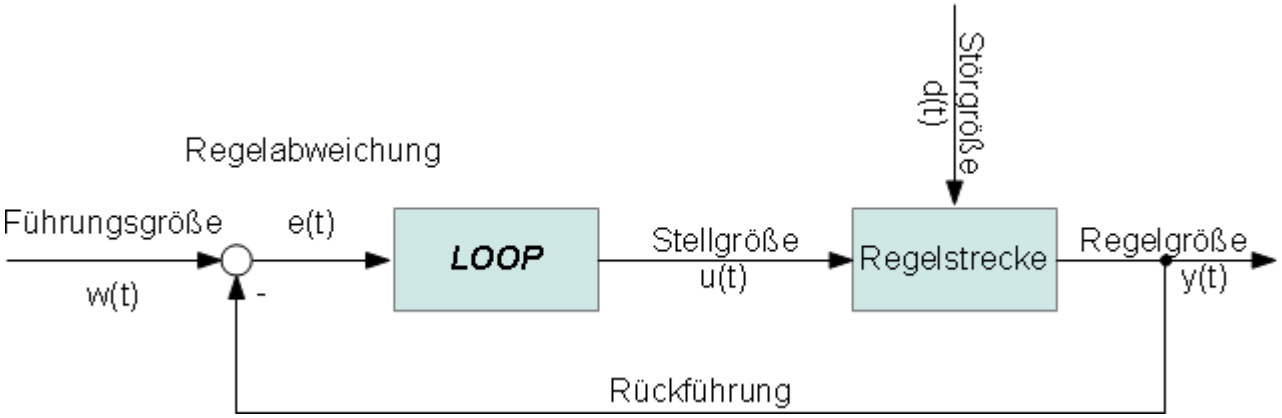
4.9.2 E_BACNETACTION

PLC-Abbildung des BACnet-Datentyps BACnetAction. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Action*.

```
TYPE E_BACNETACTION :
(
  BACnetAction_direct   := 0,
  BACnetAction_reverse  := 1,
)
END_TYPE
```

`BACnetAction_direct`: Die Reglerausgabe wirkt direkt (positiver Eingangsfehler e ergibt eine steigende Ausgabe [e = w - y])

`BACnetAction_reverse`: Die Reglerausgabe wirkt invers (der Eingangsfehler e wird invertiert [e = y - w])



4.9.3 E_BACNETADAPTERSTATUS

Status des BACnet-Adapters.

```
TYPE E_BACNETADAPTERSTATUS :
(
  BACnetAdapterStatus_Init            := 16#0,
  BACnetAdapterStatus_CheckIpAddr     := 16#1,
  BACnetAdapterStatus_CheckParam      := 16#2,
  BACnetAdapterStatus_GetGatewayMAC   := 16#3,
  BACnetAdapterStatus_WaitGatewayMAC  := 16#4,
  BACnetAdapterStatus_Complete        := 16#8,
)
END_TYPE
```

`BACnetAdapterStatus_Init`: Initialisierung hat begonnen.

`BACnetAdapterStatus_CheckIpAddr`: Prüfung der IP-Adresse.

`BACnetAdapterStatus_CheckParam`: Prüfung der Parameter.

`BACnetAdapterStatus_GetGatewayMAC`: MAC-Adresse des Gateways überprüfen.

`BACnetAdapterStatus_WaitGatewayMAC`: Warte auf die Ermittlung der MAC-Adresse des Gateways (Gateway erreichbar?).

`BACnetAdapterStatus_Complete`: Initialisierung beendet und bereit.

4.9.4 E_BACNETBINARYPV

PLC-Abbildung des BACnet-Datentyps `BACnetBinaryPV`. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Present_Value* von Binary* Objekten. Für die interne Verarbeitung der kommandierbaren Ausführung der Property vom Typ `BACnetBinaryPV`, wurden 2 weitere Zustände für die Property eingeführt (NULL und Nothing). Diese zusätzlichen Zustände sind nicht nach BACnet sichtbar und werden lediglich in der Verarbeitung der zyklischen Prozessdaten benötigt.

```
TYPE E_BACNETBINARYPV :
(
  BACnetBinaryPV_inactive  := 0,
  BACnetBinaryPV_active    := 1,
  BACnetBinaryPV_NULL      := 2,
  BACnetBinaryPV_Nothing   := 16#FFFF
)
END_TYPE
```

`BACnetBinaryPV_inactive`: Wert der Property *Present_Value* ist INACTIVE (FALSE)

`BACnetBinaryPV_active`: Wert der Property *Present_Value* ist ACTIVE (TRUE)

`BACnetBinaryPV_NULL`: Wert der Property *Present_Value* wird NULL geschrieben (kein Wert; bei kommandierbaren Properties wird der Eintrag in der zugehörigen Property *Priority_Array* gelöscht)

`BACnetBinaryPV_Nothing`: Wert der Property *Present_Value* bleibt unverändert (keine Aktion; Prozessdatum wird nicht ausgewertet)

4.9.5 E_BACNETDATATYPES

Auflistung der möglichen BACnet Datentypen (Auszug).

```
TYPE E_BACNETDATATYPES :
(
  BACnetDataType_Null                                := 0,
  BACnetDataType_Bool                                := 1,
  BACnetDataType_UnsignedInteger                     := 2,
  BACnetDataType_SignedInteger                       := 3,
  BACnetDataType_Real                                := 4,
  BACnetDataType_Double                              := 5,
  BACnetDataType_OctetString                         := 6,
  BACnetDataType_CharacterStringExt                  := 7,
  BACnetDataType_BitString                           := 8,
  BACnetDataType_Enumerated                          := 9,
  BACnetDataType_Date                                := 10,
  BACnetDataType_Time                                := 11,
  BACnetDataType_ObjectIdentifier                    := 12,
  BACnetDataType_DateTime                            := 14,
  BACnetDataType_ArrayIndex                          := 15,
  BACnetDataType_CalendarEntry                       := 16,
  BACnetDataType_ObjectType                          := 17,
  BACnetDataType_EventTransitionBits                 := 18,
  BACnetDataType_ActionList                          := 19,
  BACnetDataType_StatusFlags                         := 20,
  BACnetDataType_EventState                          := 21,
  BACnetDataType_Reliability                         := 22,
  BACnetDataType_Polarity                            := 23,
  BACnetDataType_EngineeringUnits                    := 25,
  BACnetDataType_PriorityValue                       := 26,
  BACnetDataType_LimitEnable                         := 27,
  BACnetDataType_NotifyType                          := 29,
  BACnetDataType_TimeStamp                           := 30,
  BACnetDataType_DeviceObjectPropertyReference       := 31,
  BACnetDataType_DeviceStatus                        := 32,
  BACnetDataType_ServicesSupported                   := 33,
  BACnetDataType_ObjectTypesSupported                := 34,
  BACnetDataType_Segmentation                        := 35,
  BACnetDataType_VTClass                             := 36,
  BACnetDataType_VTSession                           := 37,
  BACnetDataType_SessionKey                          := 38,
  BACnetDataType_Recipient                           := 39,
  BACnetDataType_AddressBinding                      := 40,
  BACnetDataType_COVSubscription                     := 41,
  BACnetDataType_EventType                           := 42,
  BACnetDataType_EventParameter                      := 43,
  BACnetDataType_FileAccessMethod                    := 44,
  BACnetDataType_ReadAccessSpecification             := 45,
  BACnetDataType_ReadAccessResult                    := 46,
  BACnetDataType_ObjectPropertyReference             := 47,
  BACnetDataType_SetpointReference                   := 48,
  BACnetDataType_Destination                         := 49,
  BACnetDataType_ProgramState                        := 50,
  BACnetDataType_ProgramRequest                      := 51,
  BACnetDataType_ProgramError                        := 52,
  BACnetDataType_DateRange                           := 53,
  BACnetDataType_DailySchedule                       := 54,
  BACnetDataType_SpecialEvent                        := 55,
  BACnetDataType_ClientCOV                           := 56,
  BACnetDataType_LogRecord                           := 57,
  BACnetDataType_LifeSafetyState                     := 58,
  BACnetDataType_LifeSafetyMode                      := 59,
  BACnetDataType_SilencedState                       := 60,
  BACnetDataType_LifeSafetyOperation                 := 61,
  BACnetDataType_BinaryPV                            := 62,
  BACnetDataType_DaysOfWeek                          := 63,
  BACnetDataType_WeekNDay                            := 64,
  BACnetDataType_TimeValue                           := 65,
  BACnetDataType_Value                               := 66,
  BACnetDataType_Address                             := 67,
  BACnetDataType_PropertyIdentifier                  := 68,
  BACnetDataType_EnumerationValueType                := 69,
  BACnetDataType_BitFieldBit                         := 70,
  BACnetDataType_LogDatum                            := 71,
```

```
BACnetDataType_AnyType                                := 72,
BACnetDataType_PresentValue                           := 73,
BACnetDataType_ContextTag                             := 74,
BACnetDataType_ValueChoice                            := 75,
BACnetDataType_LogStatus                              := 76,
BACnetDataType_RecipientProcess                       := 77,
BACnetDataType_SubscribeCOVPropertyRequest            := 78,
BACnetDataType_PropertyReference                      := 79,
BACnetDataType_PropertyResult                         := 80,
BACnetDataType_DeviceObjectPropertyValue              := 81,
BACnetDataType_COVNotificationRequest                 := 82,
BACnetDataType_EventNotificationRequest               := 83,
BACnetDataType_NotificationParameters                 := 84,
BACnetDataType_Change_of_Bitstring                    := 85,
BACnetDataType_Change_of_State                        := 86,
BACnetDataType_Change_of_Value                        := 87,
BACnetDataType_Command_failure                        := 88,
BACnetDataType_Floating_limit                         := 89,
BACnetDataType_Out_of_Range                           := 90,
BACnetDataType_PropertyValue                          := 91,
BACnetDataType_Complex_Tag                            := 92,
BACnetDataType_Change_of_life_safety                  := 93,
BACnetDataType_Extended                               := 94,
BACnetDataType_Buffer_ready                           := 95,
BACnetDataType_Unsigned_range                         := 96,
BACnetDataType_PropertyResultValue                    := 97,
BACnetDataType_ServiceCOVPropSubscription             := 98,
BACnetDataType_ServiceCOVSubscription                 := 99,
BACnetDataType_COVNotification                        := 100,
BACnetDataType_AcknowledgeAlarmRequest                := 101,
BACnetDataType_VTOpenRequest                          := 102,
BACnetDataType_Prescale                               := 103,
BACnetDataType_Scale                                  := 104,
BACnetDataType_Action                                 := 105,
BACnetDataType_Error                                  := 106,
BACnetDataType_ErrorType                              := 107,
BACnetDataType_ErrorClass                             := 108,
BACnetDataType_ErrorCode                              := 109,
BACnetDataType_RejectReason                           := 110,
BACnetDataType_AbortReason                            := 111,
BACnetDataType_BDTEntry                               := 112,
BACnetDataType_IpEntry                                := 113,
BACnetDataType_MaskEntry                              := 114,
BACnetDataType_EthernetAddress                        := 115,
BACnetDataType_LonTalkAddress                         := 116,
BACnetDataType_LonTalkNeuronId                        := 117,
BACnetDataType_NodeType                               := 118,
BACnetDataType_DeviceObjectReference                  := 119,
BACnetDataType_ActionCommand                          := 120,
BACnetDataType_GetEventInformation                    := 121,
BACnetDataType_GetEnrollmentSummaryCriteria           := 122,
BACnetDataType_GetEnrollmentSummaryCriteriaResponse   := 123,
BACnetDataType_DeviceCommunicationControl             := 124,
BACnetDataType_Int16                                  := 125,
BACnetDataType_UInt16                                 := 126,
BACnetDataType_UInt8                                  := 127,
BACnetDataType_TypeInt8                               := 128,
BACnetDataType_Byte                                   := 129,
BACnetDataType_FDTEntry                               := 130,
BACnetDataType_Address_3                              := 131,
BACnetDataType_Address_4                              := 132,
BACnetDataType_Address_5                              := 133,
BACnetDataType_PersistentDataState                    := 134,
BACnetDataType_FallbackRealValue                      := 135,
BACnetDataType_FallbackBinaryValue                    := 136,
BACnetDataType_RealNull                               := 137,
BACnetDataType_EnumeratedNull                         := 138,
BACnetDataType_UnsignedIntegerNull                    := 139,
BACnetDataType_Unknown                                := 140,
BACnetDataType_DiagnosisPerformance                   := 200,
BACnetDataType_DiagnosisEthStatistics                 := 201,
BACnetDataType_FrameStatistics                        := 202,
BACnetDataType_Info                                   := 203,
BACnetDataType_Diagnosis                              := 204,
BACnetDataType_TcIoEthStatistic                       := 205,
BACnetDataType_TcIoEthTxRxErrorCount                  := 206,
BACnetDataType_TcIoEthPortStatistic                   := 207,
BACnetDataType_ServerStatistics                       := 208,
BACnetDataType_FrameServicesDiag                      := 209,
BACnetDataType_UnsignedIntegerArr                     := 258,
```

```
  BACnetDataType_CharacterStringExtArr                := 263,
  BACnetDataType_ObjectIdentifierArr                  := 268,
  BACnetDataType_CalendarEntryArr                     := 272,
  BACnetDataType_ActionListArr                        := 275,
  BACnetDataType_PriorityValueArr                     := 282,
  BACnetDataType_TimeStampArr                         := 286,
  BACnetDataType_DeviceObjectPropertyReferenceArr     := 287,
  BACnetDataType_VTClassArr                           := 292,
  BACnetDataType_VTSessionArr                         := 293,
  BACnetDataType_SessionKeyArr                        := 294,
  BACnetDataType_RecipientArr                         := 295,
  BACnetDataType_AddressBindingArr                    := 296,
  BACnetDataType_COVSubscriptionArr                   := 297,
  BACnetDataType_ReadAccessSpecificationArr           := 301,
  BACnetDataType_ReadAccessResultArr                  := 302,
  BACnetDataType_DestinationArr                       := 305,
  BACnetDataType_SpecialEventArr                      := 311,
  BACnetDataType_LogRecordArr                         := 313,
  BACnetDataType_LifeSafetyStateArr                   := 314,
  BACnetDataType_TimeValueArr                         := 321,
  BACnetDataType_UnsignedIntegerList                  := 514,
  BACnetDataType_CharacterStringExtList               := 519,
  BACnetDataType_ReadAccessSpecificationList          := 557,
  BACnetDataType_ReadAccessResultList                 := 558,
  BACnetDataType_SpecialEventList                     := 567,
  BACnetDataType_LogRecordList                        := 569,
  BACnetDataType_TimeValueList                        := 577,
  BACnetDataType_ValueList                            := 578,
  BACnetDataType_LogDatumList                         := 583,
  BACnetDataType_PropertyReferenceList                := 591,
  BACnetDataType_PropertyResultList                   := 592,
  BACnetDataType_COVNotificationRequestList           := 594,
  BACnetDataType_EventNotificationRequestList         := 595,
  BACnetDataType_PropertyValueList                    := 603,
  BACnetDataType_DailyScheduleList                    := 1089,
)
END_TYPE
```

4.9.6 E_BACNETDAYSOFWEEKBITS

Bit-Belegung der Wochentage

```
TYPE E_BACNETDAYSOFWEEKBITS :
(
  BACnetDaysOfWeekBits_Monday     := 0,
  BACnetDaysOfWeekBits_Tuesday    := 1,
  BACnetDaysOfWeekBits_Wednesday  := 2,
  BACnetDaysOfWeekBits_Thursday   := 3,
  BACnetDaysOfWeekBits_Friday     := 4,
  BACnetDaysOfWeekBits_Saturday   := 5,
  BACnetDaysOfWeekBits_Sunday     := 6,
)
END_TYPE
```

`BACnetDaysOfWeekBits_Monday`: Montag.

`BACnetDaysOfWeekBits_Tuesday`: Dienstag.

`BACnetDaysOfWeekBits_Wednesday`: Mittwoch.

`BACnetDaysOfWeekBits_Thursday`: Donnerstag.

`BACnetDaysOfWeekBits_Friday`: Freitag.

`BACnetDaysOfWeekBits_Saturday`: Samstag.

`BACnetDaysOfWeekBits_Sunday`: Sonntag.

4.9.7 E_BACNETDEVICESTATUS

Status des BACnet Server Objekts (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet Device Objekt und Property *System_Status).*

```
TYPE E_BACNETDEVICESTATUS :
(
  BACnetDeviceStatus_Operational          := 0,
  BACnetDeviceStatus_OperationalReadOnly  := 1,
  BACnetDeviceStatus_DownloadRequired     := 2,
  BACnetDeviceStatus_DownloadInProgress   := 3,
  BACnetDeviceStatus_NonOperational       := 4,
  BACnetDeviceStatus_BackupInProgress     := 5,
)
END_TYPE
```

`BACnetDeviceStatus_Operational`: Betriebsbereit.

`BACnetDeviceStatus_OperationalReadOnly`: Nur-Lese-Zugriff auf Properties.

`BACnetDeviceStatus_DownloadRequired`: Konfiguration-Laden erforderlich.

`BACnetDeviceStatus_DownloadInProgress`: Konfiguration-Laden wird ausgeführt.

`BACnetDeviceStatus_NonOperational`: Nicht-Betriebsbereit.

`BACnetDeviceStatus_BackupInProgress`: Datensicherung wird ausgeführt.

4.9.8 E_BACNETEVENTSTATE

PLC-Abbildung des BACnet-Datentyps BACnetEventState. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Event_State*.

```
TYPE E_BACNETEVENTSTATE :
(
  BACnetEventState_state_normal       := 0,
  BACnetEventState_fault              := 1,
  BACnetEventState_offnormal          := 2,
  BACnetEventState_high_limit         := 3,
  BACnetEventState_low_limit          := 4,
  BACnetEventState_life_safety_alarm  := 5
)
END_TYPE
```

`BACnetEventState_state_normal`: Objektzustand normal (Property *Reliability* gleich `NO_FAULT_DETECTED`).

`BACnetEventState_fault`: Objektzustand in Fehler (Property *Reliability* ungleich `NO_FAULT_DETECTED`).

`BACnetEventState_offnormal`: Objektzustand aus-normal (Property *Reliability* gleich `NO_FAULT_DETECTED`).

`BACnetEventState_high_limit`: Objektzustand oberer Grenzwert erreicht/überschritten (Property *High_Limit*).

`BACnetEventState_low_limit`: Objektzustand unterer Grenzwert erreicht/unterschritten (Property *Low_Limit*).

`BACnetEventState_life_safety_alarm`: Objektzustand life-safety-alarm.

4.9.9 E_BACNETEVENTSTATE

PLC-Abbildung des BACnet-Datentyps BACnetEventState. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Event_State*.

```
TYPE E_BACNETEVENTSTATE :
(
  BACnetEventState_state_normal       := 0,
  BACnetEventState_fault              := 1,
  BACnetEventState_offnormal          := 2,
  BACnetEventState_high_limit         := 3,
  BACnetEventState_low_limit          := 4,
  BACnetEventState_life_safety_alarm  := 5
)
END_TYPE
```

`BACnetEventState_state_normal`: Objektzustand normal (Property *Reliability* gleich NO_FAULT_DETECTED).

`BACnetEventState_fault`: Objektzustand in Fehler (Property *Reliability* ungleich NO_FAULT_DETECTED).

`BACnetEventState_offnormal`: Objektzustand aus-normal (Property *Reliability* gleich NO_FAULT_DETECTED).

`BACnetEventState_high_limit`: Objektzustand oberer Grenzwert erreicht/überschritten (Property *High_Limit*).

`BACnetEventState_low_limit`: Objektzustand unterer Grenzwert erreicht/unterschritten (Property *Low_Limit*).

`BACnetEventState_life_safety_alarm`: Objektzustand life-safety-alarm.

4.9.10 E_BACNETEVENTTRANSITIONBIT

Bit-Belegung der Event-Transition-Flags [▶ 361] (Properties *Event_Enable* und *Acked_Transitions*).

```
TYPE E_BACNETEVENTTRANSITIONBIT :
(
  BACnetEventTransitionBit_to_offnormal  := 0,
  BACnetEventTransitionBit_to_fault      := 1,
  BACnetEventTransitionBit_to_normal     := 2
)
END_TYPE
```

4.9.11 E_BACNETEVENTTYPE

PLC-Abbildung des BACnet-Datentyps BACnetEventType. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Event_Type*.

```
TYPE E_BACNETEVENTTYPE :
(
  BACnetEventType_change_of_bitstring        := 0,
  BACnetEventType_change_of_state            := 1,
  BACnetEventType_change_of_value            := 2,
  BACnetEventType_command_failure            := 3,
  BACnetEventType_floating_limit             := 4,
  BACnetEventType_out_of_range               := 5,
  BACnetEventType_complex_event_type         := 6,
  BACnetEventType_deprecated7                := 7,
  BACnetEventType_change_of_life_safety      := 8,
  BACnetEventType_enumeration                := 9,
  BACnetEventType_buffer_ready               := 10,
  BACnetEventType_unsigned_range             := 11,
  BACnetEventType_reserved12                 := 12,
  BACnetEventType_access_event               := 13,
  BACnetEventType_double_out_of_range        := 14,
  BACnetEventType_signed_out_of_range        := 15,
  BACnetEventType_unsigned_out_of_range      := 16,
  BACnetEventType_change_of_characterstring  := 17,
  BACnetEventType_change_of_state_flags      := 18,
)
END_TYPE
```

`BACnetEventType_change_of_bitstring`: CHANGE_OF_BITSTRING

`BACnetEventType_change_of_state`: CHANGE_OF_STATE **(COS)**

`BACnetEventType_change_of_value`: CHANGE_OF_VALUE **(COV)**

`BACnetEventType_command_failure`: COMMAND_FAILURE

`BACnetEventType_floating_limit`: FLOATING_LIMIT

`BACnetEventType_out_of_range`: OUT_OF_RANGE

`BACnetEventType_complex_event_type`: COMPLEX_EVENT_TYPE

`BACnetEventType_deprecated7`: *nicht mehr verwendet*

`BACnetEventType_change_of_life_safety`: CHANGE_OF_LIFE_SAFETY

`BACnetEventType_enumeration`: ENUMERATION

`BACnetEventType_buffer_ready`: BUFFER_READY

`BACnetEventType_unsigned_range`: UNSIGNED_RANGE

`BACnetEventType_reserved12`: *reserviert für zukünftige Verwendung*

`BACnetEventType_access_event`: ACCESS_EVENT

`BACnetEventType_double_out_of_range`: DOUBLE_OUT_OF_RANGE

`BACnetEventType_signed_out_of_range`: SIGNED_OUT_OF_RANGE

`BACnetEventType_unsigned_out_of_range`: UNSIGNED_OUT_OF_RANGE

`BACnetEventType_change_of_characterstring`: CHANGE_OF_CHARACTERSTRING

`BACnetEventType_change_of_state_flags`: CHANGE_OF_STATE_FLAGS

4.9.12 E_BACNETFILEACCESSMETHOD

PLC-Abbildung des BACnet-Datentyps `BACnetFileAccessMethod`. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *File_Access_Method*.

```
TYPE E_BACNETFILEACCESSMETHOD :
(
  BACnetFileAccessMethod_record_access  := 0,
  BACnetFileAccessMethod_stream_access  := 1,
)
END_TYPE
```

`BACnetFileAccessMethod_record_access`: Dateizugriff mittels fester Datenblöcke.

`BACnetFileAccessMethod_stream_access`: Dateizugriff mittels Data-Stream (Standard Methode).

4.9.13 E_BACNETLIFESAFETYMODE

PLC-Abbildung des BACnet-Datentyps `BACnetLifeSafetyMode`. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Mode* und *Accepted_Modes*.

```
TYPE E_BACNETLIFESAFETYMODE :
(
  BACnetLifeSafetyMode_off                          := 0,
  BACnetLifeSafetyMode_on                           := 1,
  BACnetLifeSafetyMode_test                         := 2,
  BACnetLifeSafetyMode_manned                       := 3,
  BACnetLifeSafetyMode_unmanned                     := 4,
  BACnetLifeSafetyMode_armed                        := 5,
  BACnetLifeSafetyMode_disarmed                     := 6,
  BACnetLifeSafetyMode_prearmed                     := 7,
  BACnetLifeSafetyMode_slow                         := 8,
  BACnetLifeSafetyMode_fast                         := 9,
  BACnetLifeSafetyMode_disconnected                 := 10,
  BACnetLifeSafetyMode_enabled                      := 11,
  BACnetLifeSafetyMode_disabled                     := 12,
  BACnetLifeSafetyMode_automatic_release_disabled   := 13,
  BACnetLifeSafetyMode_mode_default                 := 14,
)
END_TYPE
```

`BACnetLifeSafetyMode_off`: OFF

`BACnetLifeSafetyMode_on`: ON

`BACnetLifeSafetyMode_test`: TEST

`BACnetLifeSafetyMode_manned`: MANNED

`BACnetLifeSafetyMode_unmanned`: UNMANNED

`BACnetLifeSafetyMode_armed`: ARMED

`BACnetLifeSafetyMode_disarmed`: DISARMED

`BACnetLifeSafetyMode_prearmed`: PREARMED

`BACnetLifeSafetyMode_slow`: SLOW

`BACnetLifeSafetyMode_fast`: FAST

`BACnetLifeSafetyMode_disconnected`: DISCONNECTED

`BACnetLifeSafetyMode_enabled`: ENABLED

`BACnetLifeSafetyMode_disabled`: DISABLED

`BACnetLifeSafetyMode_automatic_release_disabled`: AUTOMATIC_RELEASE_DISABLED

`BACnetLifeSafetyMode_mode_default`: MODE_DEFAULT

4.9.14 E_BACNETLIFESAFETYOPERATION

PLC-Abbildung des BACnet-Datentyps BACnetLifeSafetyOperation. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Operation_Expected*.

```
TYPE E_BACNETLIFESAFETYOPERATION :
(
  BACnetLifeSafetyOperation_none               := 0,
  BACnetLifeSafetyOperation_silence            := 1,
  BACnetLifeSafetyOperation_silence_audible    := 2,
  BACnetLifeSafetyOperation_silence_visual     := 3,
  BACnetLifeSafetyOperation_reset              := 4,
  BACnetLifeSafetyOperation_reset_alarm        := 5,
  BACnetLifeSafetyOperation_reset_fault        := 6,
  BACnetLifeSafetyOperation_unsilence          := 7,
  BACnetLifeSafetyOperation_unsilence_audible  := 8,
  BACnetLifeSafetyOperation_unsilence_visual   := 9,
)
END_TYPE
```

`BACnetLifeSafetyOperation_none`: NONE

`BACnetLifeSafetyOperation_silence`: SILENCE

`BACnetLifeSafetyOperation_silence_audible`: SILENCE_AUDIBLE

`BACnetLifeSafetyOperation_silence_visual`: SILENCE_VISUAL

`BACnetLifeSafetyOperation_reset`: RESET

`BACnetLifeSafetyOperation_reset_alarm`: RESET_ALARM

`BACnetLifeSafetyOperation_reset_fault`: RESET_FAULT

`BACnetLifeSafetyOperation_unsilence`: UNSILENCE

`BACnetLifeSafetyOperation_unsilence_audible`: UNSILENCE_AUDIBLE

`BACnetLifeSafetyOperation_unsilence_visual`: UNSILENCE_VISUAL

4.9.15 E_BACNETLIMITENABLEBITS

Bit-Belegung der Limit-Enable-Flags [▶ 365] (Property *Limit_Enable*).

```
TYPE E_BACNETLIMITENABLEBITS :
(
  BACnetLimitDisableBits_  := 0,
  BACnetLimitEnableBits_   := 1
)
END_TYPE
```

4.9.16 E_BACNETLOGGINGTYPE

PLC-Abbildung des BACnet-Datentyps BACnetLoggingType. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Logging_Type*.

```
TYPE E_BACNETLOGGINGTYPE :
(
  BACnetLoggingType_cov,
  BACnetLoggingType_polled,
  BACnetLoggingType_triggered
)
END_TYPE
```

`BACnetLoggingType_cov: COV`: Change on value: Wertänderungen werden geloggt (siehe Property *Client_Cov_Increment* und *Cov_Resubscription_Interval*).

`BACnetLoggingType_polled`: POLLED: Werte werden mit festem Abstand geloggt (siehe Property *Log_Interval*, *Interval_Offset* und *Align_Intervals*).

`BACnetLoggingType_triggered`: TRIGGERED: Werte werden mit Hilfe einer Trigger-Property geloggt (siehe online Property *Trigger*)

4.9.17 E_BACNETLOOPMODE

Betriebsarten des PLC-LOOP-Objekts FB_BACnet_LOOP [▶ 204].

```
TYPE E_BACNETLOOPMODE :
(
  BACnetLoopMode_None,
  BACnetLoopMode_Ideal,
  BACnetLoopMode_Standard
)
END_TYPE
```

`BACnetLoopMode_None`: None: Ungültige LOOP-Parameter. Die Betriebsart kann nicht eindeutig ermittelt werden.

`BACnetLoopMode_Ideal`: Ideal: Der Regler wird in Ideal-Form betrieben: P, I und D Anteil wirken parallel; die Parameter für P, I und D sind Faktoren und haben die Einheit [1] *Other_no_units* (Kp, Ki und Kd).

`BACnetLoopMode_Standard`: Standard: Der Regler wird in Standard-Form betrieben: P und I bzw. P und D Anteil wirken in Reihe; der Parameter für P ist ein Faktor und hat die Einheit [1] *Other_no_units* (Kp), die Parameter für I und D sind zeitbasiert und haben die Einheit [s] *Time_seconds* (Tn und Tv).

4.9.18 E_BACNETNOTIFYTYPE

PLC-Abbildung des BACnet-Datentyps BACnetNotifyType. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Notify_Type*.

```
TYPE E_BACNETNOTIFYTYPE :
(
  BACnetNotifyType_alarm             := 0,
  BACnetNotifyType_notify_event      := 1,
  BACnetNotifyType_ack_notification  := 2
)
END_TYPE
```

`BACnetNotifyType_alarm`: ALARM

`BACnetNotifyType_notify_event`: NOTIFY_EVENT

`BACnetNotifyType_ack_notification`: ACK_NOTIFICATION

4.9.19 E_BACNETOBJECTSUPPORTEDBITS

Bit-Belegung der Object-Types-Supported-Flags [▶ 367] (Property *Protocol_Object_Types_Supported*).

```
TYPE E_BACNETOBJECTSUPPORTEDBITS :
(
  BACnetObjectSupportedBits_AnalogInput           := 0,
  BACnetObjectSupportedBits_AnalogOutput          := 1,
  BACnetObjectSupportedBits_AnalogValue           := 2,
  BACnetObjectSupportedBits_BinaryInput           := 3,
  BACnetObjectSupportedBits_BinaryOutput          := 4,
  BACnetObjectSupportedBits_BinaryValue           := 5,
  BACnetObjectSupportedBits_Calendar              := 6,
  BACnetObjectSupportedBits_Command               := 7,
  BACnetObjectSupportedBits_Device                := 8,
  BACnetObjectSupportedBits_EventEnrollment       := 9,
  BACnetObjectSupportedBits_File                  := 10,
  BACnetObjectSupportedBits_Group                 := 11,
  BACnetObjectSupportedBits_Loop                  := 12,
  BACnetObjectSupportedBits_MultiStateInput       := 13,
  BACnetObjectSupportedBits_MultiStateOutput      := 14,
  BACnetObjectSupportedBits_NotificationClass     := 15,
  BACnetObjectSupportedBits_Program               := 16,
  BACnetObjectSupportedBits_Schedule              := 17,
  BACnetObjectSupportedBits_Averaging             := 18,
  BACnetObjectSupportedBits_MultiStateValue       := 19,
  BACnetObjectSupportedBits_TrendLog              := 20,
  BACnetObjectSupportedBits_LifeSafetyPoint       := 21,
  BACnetObjectSupportedBits_LifeSafetyZone        := 22,
  BACnetObjectSupportedBits_Accumulator           := 23,
  BACnetObjectSupportedBits_PulseConverter        := 24,
  BACnetObjectSupportedBits_EventLog              := 25,
  BACnetObjectSupportedBits_GlobalGroup           := 26,
  BACnetObjectSupportedBits_TrendLogMultiple      := 27,
  BACnetObjectSupportedBits_LoadControl           := 28,
  BACnetObjectSupportedBits_StructuredView        := 29,
  BACnetObjectSupportedBits_AccessDoor            := 30,
  BACnetObjectSupportedBits_unassigend31          := 31,
  BACnetObjectSupportedBits_AccessCredential      := 32,
  BACnetObjectSupportedBits_AccessPoint           := 33,
  BACnetObjectSupportedBits_AccessRights          := 34,
  BACnetObjectSupportedBits_AccessUser            := 35,
  BACnetObjectSupportedBits_AccessZone            := 36,
  BACnetObjectSupportedBits_CredentialDataInput   := 37,
  BACnetObjectSupportedBits_NetworkSecurity       := 38,
  BACnetObjectSupportedBits_BitStringValue        := 39,
  BACnetObjectSupportedBits_CharacterStringValue  := 40,
  BACnetObjectSupportedBits_DatePatternValue      := 41,
  BACnetObjectSupportedBits_DateValue             := 42,
  BACnetObjectSupportedBits_DateTimePatternValue  := 43,
  BACnetObjectSupportedBits_DateTimeValue         := 44,
  BACnetObjectSupportedBits_IntegerValue          := 45,
  BACnetObjectSupportedBits_LargeAnalogValue      := 46,
  BACnetObjectSupportedBits_OctetStringValue      := 47,
  BACnetObjectSupportedBits_PositiveIntegerValue  := 48,
  BACnetObjectSupportedBits_TimePatternValue      := 49,
  BACnetObjectSupportedBits_TimeValue             := 50,
  BACnetObjectSupportedBits_NetworkPort           := 51,
  BACnetObjectSupportedBits_AlertEnrollment       := 52,
  BACnetObjectSupportedBits_Channel               := 53,
)
END_TYPE
```

4.9.20 E_BACNETOBJECTTYPE

Auflistung der möglichen BACnet-Objekte (Auszug).

```
TYPE E_BACNETOBJECTTYPE :
(
  BACnetObjectType_AnalogInput       := 0,
  BACnetObjectType_AnalogOutput      := 1,
  BACnetObjectType_AnalogValue       := 2,
  BACnetObjectType_BinaryInput       := 3,
  BACnetObjectType_BinaryOutput      := 4,
  BACnetObjectType_BinaryValue       := 5,
  BACnetObjectType_Calendar          := 6,
  BACnetObjectType_Command           := 7,
```

```
  BACnetObjectType_Device            := 8,
  BACnetObjectType_EventEnrollment   := 9,
  BACnetObjectType_File              := 10,
  BACnetObjectType_Group             := 11,
  BACnetObjectType_Loop              := 12,
  BACnetObjectType_MultiStateInput   := 13,
  BACnetObjectType_MultiStateOutput  := 14,
  BACnetObjectType_NotificationClass := 15,
  BACnetObjectType_Program           := 16,
  BACnetObjectType_Schedule          := 17,
  BACnetObjectType_Averaging         := 18,
  BACnetObjectType_MultiStateValue   := 19,
  BACnetObjectType_TrendLog          := 20,
  BACnetObjectType_LifeSafetyPoint   := 21,
  BACnetObjectType_LifeSafetyZone    := 22,
  BACnetObjectType_Accumulator       := 23,
  BACnetObjectType_PulseConverter    := 24,
  BACnetObjectType_EventLog          := 25,
  BACnetObjectType_GlobalGroup       := 26,
  BACnetObjectType_TrendLogMultiple  := 27,
  BACnetObjectType_LoadControl       := 28,
  BACnetObjectType_StructuredView    := 29,
  BACnetObjectType_AccessDoor        := 30,
  BACnetObjectType_unassigend31      := 31,
  BACnetObjectType_AccessCredential  := 32,
  BACnetObjectType_AccessPoint       := 33,
  BACnetObjectType_AccessRights      := 34,
  BACnetObjectType_AccessUser        := 35,
  BACnetObjectType_AccessZone        := 36,
  BACnetObjectType_CredentialDataInput   := 37,
  BACnetObjectType_NetworkSecurity       := 38,
  BACnetObjectType_BitStringValue    := 39,
  BACnetObjectType_CharacterStringValue  := 40,
  BACnetObjectType_DatePatternValue      := 41,
  BACnetObjectType_DateValue         := 42,
  BACnetObjectType_DateTimePatternValue  := 43,
  BACnetObjectType_DateTimeValue     := 44,
  BACnetObjectType_IntegerValue      := 45,
  BACnetObjectType_LargeAnalogValue      := 46,
  BACnetObjectType_OctetStringValue      := 47,
  BACnetObjectType_PositiveIntegerValue  := 48,
  BACnetObjectType_TimePatternValue      := 49,
  BACnetObjectType_TimeValue         := 50,
  BACnetObjectType_NetworkPort       := 51,
  BACnetObjectType_AlertEnrollment       := 52,
  BACnetObjectType_Channel           := 53
  BACnetObjectType_ZCount,
)
END_TYPE
```

4.9.21 E_BACNETPERSISTENTDATASTATE

PLC-Abbildung des herstellerspezifischen BACnet-Datentyps PersistentDataState (siehe BACnet Device Objekt [▶ 199]).

```
TYPE E_BACNETPERSISTENTDATASTATE :
(
  BACnetPersistentData_Disabled,
  BACnetPersistentData_Initialize,
  BACnetPersistentData_NoChange,
  BACnetPersistentData_Ready,
  BACnetPersistentData_Write,
  BACnetPersistentData_Writing
)
END_TYPE
```

`BACnetPersistentData_Disabled`: Persistente Daten sind deaktiviert.

`BACnetPersistentData_Initialize`: Persistente Daten werden initialisiert (geladen).

`BACnetPersistentData_NoChange`: Keine Änderung an online Daten erkannt.

`BACnetPersistentData_Ready`: Persistente Daten können geschrieben werden (Änderungen an online Daten erkannt).

`BACnetPersistentData_Write`: Anforderung persistente Daten zu schreiben: Wird die Property *Persistent_Data* auf diesen Wert gesetzt, dann setzt dies die interne Anforderung zum Schreiben der persistenten Daten im BACnet-Treiber.

`BACnetPersistentData_Writing`: Persistente Daten werden geschrieben.

4.9.22 E_BACNETPIDTUNINGMODE

Tuningmodi des Bausteins FB_BACnet_LOOP [▶ 204].

```
TYPE E_BACNETPIDTUNINGMODE :
(
  BACnetPIDTuningMode_P,
  BACnetPIDTuningMode_PI,
  BACnetPIDTuningMode_PD,
  BACnetPIDTuningMode_PID
)
END_TYPE
```

`BACnetPIDTuningMode_P`: Regler auf P-Verhalten optimieren (I und D deaktiviert; typischerweise für Beleuchtungsregelung).

`BACnetPIDTuningMode_PI`: Regler auf PI-Verhalten optimieren (D deaktiviert; typischerweise für Temperaturregelung).

`BACnetPIDTuningMode_PD`: Regler auf PD-Verhalten optimieren (I deaktiviert).

`BACnetPIDTuningMode_PID`: Regler auf PID-Verhalten optimieren.

4.9.23 E_BACNETPOLARITY

PLC-Abbildung des BACnet-Datentyps BACnetPolarity. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Polarity*.

```
TYPE E_BACNETPOLARITY :
(
  BACnetPolarity_normal   := 0,
  BACnetPolarity_reverse  := 1
)
END_TYPE
```

`BACnetPolarity_normal`: Normal: z.B. BACnet Binary Input physisch aktiv (24V Pegel/TRUE/ACTIVE) → Property Present_Value = ACTIVE (wenn Property Reliability = NO_FAULT_DETECTED und Out_Of_Service = FALSE).

`BACnetPolarity_reverse`: Reverse: z.B. BACnet Binary Input physisch aktiv (24V Pegel/TRUE/ACTIVE) → Property Present_Value = INACTIVE (wenn Property Reliability = NO_FAULT_DETECTED und Out_Of_Service = FALSE).

Achtung: Die Property *Polarity* hat keine Auswirkungen auf die Bewertung der Property *Feedback_Value* (z.B. BACnet Binary Output).

4.9.24 E_BACNETPRIORITY

Auflistung der möglichen BACnet Prioritäten einer kommandierbaren Property (z.B. *Present_Value*).

```
TYPE E_BACNETPRIORITY :
(
  BACnetPriority_None  := 0,
  BACnetPriority_1     := 1,
  BACnetPriority_2     := 2,
  BACnetPriority_3     := 3,
  BACnetPriority_4     := 4,
  BACnetPriority_5     := 5,
  BACnetPriority_6     := 6,
  BACnetPriority_7     := 7,
  BACnetPriority_8     := 8,
```

```
  BACnetPriority_9     := 9,
  BACnetPriority_10    := 10,
  BACnetPriority_11    := 11,
  BACnetPriority_12    := 12,
  BACnetPriority_13    := 13,
  BACnetPriority_14    := 14,
  BACnetPriority_15    := 15,
  BACnetPriority_16    := 16
)
END_TYPE
```

`BACnetPriority_None`: Keine Priorität.

`BACnetPriority_1`: Priorität 1: Manual-Life Safety

`BACnetPriority_2`: Priorität 2: Automatic-Life Safety

`BACnetPriority_3`: Priorität 3: Frei

`BACnetPriority_4`: Priorität 4: Frei

`BACnetPriority_5`: Priorität 5: Critical Equipment Control

`BACnetPriority_6`: Priorität 6: Minimum On/Off

`BACnetPriority_7`: Priorität 7: Frei

`BACnetPriority_8`: Priorität 8: Manual Operator (z.B. vor Ort Bedienung)

`BACnetPriority_9`: Priorität 9: Frei

`BACnetPriority_10`: Priorität 10: Frei

`BACnetPriority_11`: Priorität 11: Frei

`BACnetPriority_12`: Priorität 12: PLC-Bausteine (Standard bei Automapping von Bibliothek-FBs)

`BACnetPriority_13`: Priorität 13: Frei

`BACnetPriority_14`: Priorität 14: Frei

`BACnetPriority_15`: Priorität 15: Frei

`BACnetPriority_16`: Priorität 16: Frei

4.9.25 E_BACNETPROGRAMERROR

PLC-Abbildung des BACnet-Datentyps BACnetProgramError. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Reason_For_Halt* und Funktionsbausteinbeschreibung FB_BACnet_Program [▶ 222].

```
TYPE E_BACNETPROGRAMERROR :
(
  BACnetProgramError_program_normal      := 0,
  BACnetProgramError_load_failed         := 1,
  BACnetProgramError_progerror_internal  := 2,
  BACnetProgramError_program             := 3,
  BACnetProgramError_other               := 4
)
END_TYPE
```

`BACnetProgramError_program_normal`: PROGRAM_NORMAL

`BACnetProgramError_load_failed`: LOAD_FAILED

`BACnetProgramError_progerror_internal`: PROGERROR_INTERNAL

`BACnetProgramError_program`: PROGRAM

`BACnetProgramError_other`: OTHER

4.9.26 E_BACNETPROGRAMREQUEST

PLC-Abbildung des BACnet-Datentyps BACnetProgramRequest. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Program_Change* und Funktionsbausteinbeschreibung FB_BACnet_Program [▶ 222].

```
TYPE E_BACNETPROGRAMREQUEST :
(
  BACnetProgramRequest_ready    := 0,
  BACnetProgramRequest_load     := 1,
  BACnetProgramRequest_run      := 2,
  BACnetProgramRequest_halt     := 3,
  BACnetProgramRequest_restart  := 4,
  BACnetProgramRequest_unload   := 5
)
END_TYPE
```

`BACnetProgramRequest_ready`: READY

`BACnetProgramRequest_load`: LOAD

`BACnetProgramRequest_run`: RUN

`BACnetProgramRequest_halt`: HALT

`BACnetProgramRequest_restart`: RESTART

`BACnetProgramRequest_unload`: UNLOAD

4.9.27 E_BACNETPROGRAMSTATE

PLC-Abbildung des BACnet-Datentyps BACnetProgramState. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Program_State* und Funktionsbausteinbeschreibung FB_BACnet_Program [▶ 222].

```
TYPE E_BACNETPROGRAMSTATE :
(
  BACnetProgramState_idle       := 0,
  BACnetProgramState_loading    := 1,
  BACnetProgramState_running    := 2,
  BACnetProgramState_waiting    := 3,
  BACnetProgramState_halted     := 4,
  BACnetProgramState_unloading  := 5
)
END_TYPE
```

`BACnetProgramState_idle`: IDLE

`BACnetProgramState_loading`: LOADING

`BACnetProgramState_running`: RUNNING

`BACnetProgramState_waiting`: WAITING

`BACnetProgramState_halted`: HALTED

`BACnetProgramState_unloading`: UNLOADING

4.9.28 E_BACNETPROPERTYIDENTIFIER

Auflistung der möglichen BACnet-Properties (Auszug).

```
TYPE E_BACNETPROPERTYIDENTIFIER :
(
  BACnetPropertyIdentifier_AckedTransitions                 := 0,
  BACnetPropertyIdentifier_AckRequired                      := 1,
  BACnetPropertyIdentifier_Action                           := 2,
  BACnetPropertyIdentifier_ActionText                       := 3,
  BACnetPropertyIdentifier_ActiveText                       := 4,
  BACnetPropertyIdentifier_ActiveVTSessions                 := 5,
  BACnetPropertyIdentifier_AlarmValue                       := 6,
  BACnetPropertyIdentifier_AlarmValues                      := 7,
  BACnetPropertyIdentifier_All                              := 8,
  BACnetPropertyIdentifier_AllWritesSuccessful              := 9,
```

```
BACnetPropertyIdentifier_ApduSegmentTimeout                := 10,
BACnetPropertyIdentifier_ApduTimeout                       := 11,
BACnetPropertyIdentifier_ApplicationSoftwareVersion        := 12,
BACnetPropertyIdentifier_Archive                           := 13,
BACnetPropertyIdentifier_Bias                              := 14,
BACnetPropertyIdentifier_ChangeOfStateCount                := 15,
BACnetPropertyIdentifier_ChangeOfStateTime                 := 16,
BACnetPropertyIdentifier_NotificationClass                 := 17,
BACnetPropertyIdentifier_Reserved18                        := 18,
BACnetPropertyIdentifier_ControlledVariableReference       := 19,
BACnetPropertyIdentifier_ControlledVariableUnits           := 20,
BACnetPropertyIdentifier_ControlledVariableValue           := 21,
BACnetPropertyIdentifier_CovIncrement                      := 22,
BACnetPropertyIdentifier_DateList                          := 23,
BACnetPropertyIdentifier_DaylightSavingsStatus             := 24,
BACnetPropertyIdentifier_Deadband                          := 25,
BACnetPropertyIdentifier_DerivativeConstant                := 26,
BACnetPropertyIdentifier_DerivativeConstantUnits           := 27,
BACnetPropertyIdentifier_Description                       := 28,
BACnetPropertyIdentifier_DescriptionOfHalt                 := 29,
BACnetPropertyIdentifier_DeviceAddressBinding              := 30,
BACnetPropertyIdentifier_DeviceType                        := 31,
BACnetPropertyIdentifier_EffectivePeriod                   := 32,
BACnetPropertyIdentifier_ElapsedActiveTime                 := 33,
BACnetPropertyIdentifier_ErrorLimit                        := 34,
BACnetPropertyIdentifier_EventEnable                       := 35,
BACnetPropertyIdentifier_EventState                        := 36,
BACnetPropertyIdentifier_EventType                         := 37,
BACnetPropertyIdentifier_ExceptionSchedule                 := 38,
BACnetPropertyIdentifier_FaultValues                       := 39,
BACnetPropertyIdentifier_FeedbackValue                     := 40,
BACnetPropertyIdentifier_FileAccessMethod                  := 41,
BACnetPropertyIdentifier_FileSize                          := 42,
BACnetPropertyIdentifier_FileType                          := 43,
BACnetPropertyIdentifier_FirmwareRevision                  := 44,
BACnetPropertyIdentifier_HighLimit                         := 45,
BACnetPropertyIdentifier_InactiveText                      := 46,
BACnetPropertyIdentifier_InProcess                         := 47,
BACnetPropertyIdentifier_InstanceOf                        := 48,
BACnetPropertyIdentifier_IntegralConstant                  := 49,
BACnetPropertyIdentifier_IntegralConstantUnits             := 50,
BACnetPropertyIdentifier_IssueConfirmedNotifications       := 51,
BACnetPropertyIdentifier_LimitEnable                       := 52,
BACnetPropertyIdentifier_ListOfGroupMembers                := 53,
BACnetPropertyIdentifier_ListOfObjectPropertyReferences    := 54,
BACnetPropertyIdentifier_ListOfSessionKeys                 := 55,
BACnetPropertyIdentifier_LocalDate                         := 56,
BACnetPropertyIdentifier_LocalTime                         := 57,
BACnetPropertyIdentifier_Location                          := 58,
BACnetPropertyIdentifier_LowLimit                          := 59,
BACnetPropertyIdentifier_ManipulatedVariableReference      := 60,
BACnetPropertyIdentifier_MaximumOutput                     := 61,
BACnetPropertyIdentifier_MaxApduLengthAccepted             := 62,
BACnetPropertyIdentifier_MaxInfoFrames                     := 63,
BACnetPropertyIdentifier_MaxMaster                         := 64,
BACnetPropertyIdentifier_MaxPresValue                      := 65,
BACnetPropertyIdentifier_MinimumOffTime                    := 66,
BACnetPropertyIdentifier_MinimumOnTime                     := 67,
BACnetPropertyIdentifier_MinimumOutput                     := 68,
BACnetPropertyIdentifier_MinPresValue                      := 69,
BACnetPropertyIdentifier_ModelName                         := 70,
BACnetPropertyIdentifier_ModificationDate                  := 71,
BACnetPropertyIdentifier_NotifyType                        := 72,
BACnetPropertyIdentifier_NumberOfAPDURetries               := 73,
BACnetPropertyIdentifier_NumberOfStates                    := 74,
BACnetPropertyIdentifier_ObjectIdentifier                  := 75,
BACnetPropertyIdentifier_ObjectList                        := 76,
BACnetPropertyIdentifier_ObjectName                        := 77,
BACnetPropertyIdentifier_ObjectPropertyReference           := 78,
BACnetPropertyIdentifier_ObjectType                        := 79,
BACnetPropertyIdentifier_Optional                          := 80,
BACnetPropertyIdentifier_OutOfService                      := 81,
BACnetPropertyIdentifier_OutputUnits                       := 82,
BACnetPropertyIdentifier_EventParameters                   := 83,
BACnetPropertyIdentifier_Polarity                          := 84,
BACnetPropertyIdentifier_PresentValue                      := 85,
BACnetPropertyIdentifier_Priority                          := 86,
BACnetPropertyIdentifier_PriorityArray                     := 87,
BACnetPropertyIdentifier_PriorityForWriting                := 88,
BACnetPropertyIdentifier_ProcessIdentifier                 := 89,
```

```
BACnetPropertyIdentifier_ProgramChange                  := 90,
BACnetPropertyIdentifier_ProgramLocation                := 91,
BACnetPropertyIdentifier_ProgramState                   := 92,
BACnetPropertyIdentifier_ProportionalConstant           := 93,
BACnetPropertyIdentifier_ProportionalConstantUnits      := 94,
BACnetPropertyIdentifier_ProtocolConformanceClass       := 95,
BACnetPropertyIdentifier_ProtocolObjectTypesSupported   := 96,
BACnetPropertyIdentifier_ProtocolServicesSupported      := 97,
BACnetPropertyIdentifier_ProtocolVersion                := 98,
BACnetPropertyIdentifier_ReadOnly                       := 99,
BACnetPropertyIdentifier_ReasonForHalt                  := 100,
BACnetPropertyIdentifier_Recipient                      := 101,
BACnetPropertyIdentifier_RecipientList                  := 102,
BACnetPropertyIdentifier_Reliability                    := 103,
BACnetPropertyIdentifier_RelinquishDefault              := 104,
BACnetPropertyIdentifier_Required                       := 105,
BACnetPropertyIdentifier_Resolution                     := 106,
BACnetPropertyIdentifier_SegmentationSupported          := 107,
BACnetPropertyIdentifier_Setpoint                       := 108,
BACnetPropertyIdentifier_SetpointReference              := 109,
BACnetPropertyIdentifier_StateText                      := 110,
BACnetPropertyIdentifier_StatusFlags                    := 111,
BACnetPropertyIdentifier_SystemStatus                   := 112,
BACnetPropertyIdentifier_TimeDelay                      := 113,
BACnetPropertyIdentifier_TimeOfActiveTimeReset          := 114,
BACnetPropertyIdentifier_TimeOfStateCountReset          := 115,
BACnetPropertyIdentifier_TimeSynchronizationRecipients  := 116,
BACnetPropertyIdentifier_Units                          := 117,
BACnetPropertyIdentifier_UpdateInterval                 := 118,
BACnetPropertyIdentifier_UtcOffset                      := 119,
BACnetPropertyIdentifier_VendorIdentifier               := 120,
BACnetPropertyIdentifier_VendorName                     := 121,
BACnetPropertyIdentifier_VtClassesSupported             := 122,
BACnetPropertyIdentifier_WeeklySchedule                 := 123,
BACnetPropertyIdentifier_AttemptedSamples               := 124,
BACnetPropertyIdentifier_AverageValue                   := 125,
BACnetPropertyIdentifier_BufferSize                     := 126,
BACnetPropertyIdentifier_ClientCovIncrement             := 127,
BACnetPropertyIdentifier_CovResubscriptionInterval      := 128,
BACnetPropertyIdentifier_CurrentNotifyTime              := 129,
BACnetPropertyIdentifier_EventTimeStamps                := 130,
BACnetPropertyIdentifier_LogBuffer                      := 131,
BACnetPropertyIdentifier_LogDeviceObjectProperty        := 132,
BACnetPropertyIdentifier_Enable                         := 133,
BACnetPropertyIdentifier_LogInterval                    := 134,
BACnetPropertyIdentifier_MaximumValue                   := 135,
BACnetPropertyIdentifier_MinimumValue                   := 136,
BACnetPropertyIdentifier_NotificationThreshold          := 137,
BACnetPropertyIdentifier_PreviousNotifyTime             := 138,
BACnetPropertyIdentifier_ProtocolRevision               := 139,
BACnetPropertyIdentifier_RecordsSinceNotification       := 140,
BACnetPropertyIdentifier_RecordCount                    := 141,
BACnetPropertyIdentifier_StartTime                      := 142,
BACnetPropertyIdentifier_StopTime                       := 143,
BACnetPropertyIdentifier_StopWhenFull                   := 144,
BACnetPropertyIdentifier_TotalRecordCount               := 145,
BACnetPropertyIdentifier_ValidSamples                   := 146,
BACnetPropertyIdentifier_WindowInterval                 := 147,
BACnetPropertyIdentifier_WindowSamples                  := 148,
BACnetPropertyIdentifier_MaximumValueTimestamp          := 149,
BACnetPropertyIdentifier_MinimumValueTimestamp          := 150,
BACnetPropertyIdentifier_VarianceValue                  := 151,
BACnetPropertyIdentifier_ActiveCovSubscriptions         := 152,
BACnetPropertyIdentifier_BackupFailureTimeout           := 153,
BACnetPropertyIdentifier_ConfigurationFiles             := 154,
BACnetPropertyIdentifier_DatabaseRevision               := 155,
BACnetPropertyIdentifier_DirectReading                  := 156,
BACnetPropertyIdentifier_LastRestoreTime                := 157,
BACnetPropertyIdentifier_MaintenanceRequired            := 158,
BACnetPropertyIdentifier_MemberOf                       := 159,
BACnetPropertyIdentifier_Mode                           := 160,
BACnetPropertyIdentifier_OperationExpected              := 161,
BACnetPropertyIdentifier_Setting                        := 162,
BACnetPropertyIdentifier_Silenced                       := 163,
BACnetPropertyIdentifier_TrackingValue                  := 164,
BACnetPropertyIdentifier_ZoneMembers                    := 165,
BACnetPropertyIdentifier_LifeSafetyAlarmValues          := 166,
BACnetPropertyIdentifier_MaxSegmentsAccepted            := 167,
BACnetPropertyIdentifier_ProfileName                    := 168,
BACnetPropertyIdentifier_AutoSlaveDiscovery             := 169,
```

```
BACnetPropertyIdentifier_ManualSlaveAddressBinding          := 170,
BACnetPropertyIdentifier_SlaveAddressBinding                := 171,
BACnetPropertyIdentifier_SlaveProxyEnable                   := 172,
BACnetPropertyIdentifier_LastNotifyRecord                   := 173,
BACnetPropertyIdentifier_ScheduleDefault                    := 174,
BACnetPropertyIdentifier_AcceptedModes                      := 175,
BACnetPropertyIdentifier_AdjustValue                        := 176,
BACnetPropertyIdentifier_Count                              := 177,
BACnetPropertyIdentifier_CountBeforeChange                  := 178,
BACnetPropertyIdentifier_CountChangeTime                    := 179,
BACnetPropertyIdentifier_CovPeriod                          := 180,
BACnetPropertyIdentifier_InputReference                     := 181,
BACnetPropertyIdentifier_LimitMonitoringInverval            := 182,
BACnetPropertyIdentifier_LoggingObject                      := 183,
BACnetPropertyIdentifier_LoggingRecord                      := 184,
BACnetPropertyIdentifier_Prescale                           := 185,
BACnetPropertyIdentifier_PulseRate                          := 186,
BACnetPropertyIdentifier_Scale                              := 187,
BACnetPropertyIdentifier_ScaleFactor                        := 188,
BACnetPropertyIdentifier_UpdateTime                         := 189,
BACnetPropertyIdentifier_ValueBeforeChange                  := 190,
BACnetPropertyIdentifier_ValueSet                           := 191,
BACnetPropertyIdentifier_ValueChangeTime                    := 192,
BACnetPropertyIdentifier_AlignIntervals                     := 193,
BACnetPropertyIdentifier_GroupMemberNames                   := 194,
BACnetPropertyIdentifier_IntervalOffset                     := 195,
BACnetPropertyIdentifier_LastRestartReason                  := 196,
BACnetPropertyIdentifier_LoggingType                        := 197,
BACnetPropertyIdentifier_MemberStatusFlags                  := 198,
BACnetPropertyIdentifier_NotificationPeriod                 := 199,
BACnetPropertyIdentifier_PreviousNotifyRecord               := 200,
BACnetPropertyIdentifier_RequestedUpdateInterval            := 201,
BACnetPropertyIdentifier_RestartNotificationRecipients      := 202,
BACnetPropertyIdentifier_TimeOfDeviceRestart                := 203,
BACnetPropertyIdentifier_TimeSynchronizationInterval        := 204,
BACnetPropertyIdentifier_Trigger                            := 205,
BACnetPropertyIdentifier_UTCTimeSynchronizationRecipients   := 206,
BACnetPropertyIdentifier_NodeSubtype                        := 207,
BACnetPropertyIdentifier_NodeType                           := 208,
BACnetPropertyIdentifier_StructuredObjectList               := 209,
BACnetPropertyIdentifier_SubordinateAnnotations             := 210,
BACnetPropertyIdentifier_SubordinateList                    := 211,
BACnetPropertyIdentifier_ActualShedLevel                    := 212,
BACnetPropertyIdentifier_DutyWindow                         := 213,
BACnetPropertyIdentifier_ExpectedShedLevel                  := 214,
BACnetPropertyIdentifier_FullDutyBaseline                   := 215,
BACnetPropertyIdentifier_NodeSubtype216                     := 216,
BACnetPropertyIdentifier_NodeType217                        := 217,
BACnetPropertyIdentifier_RequestedShedLevel                 := 218,
BACnetPropertyIdentifier_ShedDuration                       := 219,
BACnetPropertyIdentifier_ShedLevelDescriptions              := 220,
BACnetPropertyIdentifier_ShedLevels                         := 221,
BACnetPropertyIdentifier_StateDescription                   := 222,
BACnetPropertyIdentifier_DoorAlarmState                     := 226,
BACnetPropertyIdentifier_DoorExtendedPulseTime              := 227,
BACnetPropertyIdentifier_DoorMembers                        := 228,
BACnetPropertyIdentifier_DoorOpenTooLongTime                := 229,
BACnetPropertyIdentifier_DoorPulseTime                      := 230,
BACnetPropertyIdentifier_DoorStatus                         := 231,
BACnetPropertyIdentifier_DoorUnlockDelayTime                := 232,
BACnetPropertyIdentifier_LockStatus                         := 233,
BACnetPropertyIdentifier_MaskedAlarmValues                  := 234,
BACnetPropertyIdentifier_SecuredStatus                      := 235,
BACnetPropertyIdentifier_BackupAndRestoreState              := 338,
BACnetPropertyIdentifier_BackupPreparationTime              := 339,
BACnetPropertyIdentifier_RestoreCompletionTime              := 340,
BACnetPropertyIdentifier_RestorePreparationTime             := 341,
BACnetPropertyIdentifier_EventMessageTexts                  := 351,
BACnetPropertyIdentifier_EventMessageTextsConfig            := 352,
BACnetPropertyIdentifier_ScaleOffset                        := 512,
BACnetPropertyIdentifier_IoBusNr                            := 513,
BACnetPropertyIdentifier_IoModuleNr                         := 514,
BACnetPropertyIdentifier_IoModuleChn                        := 515,
BACnetPropertyIdentifier_PersistentData                     := 516,
BACnetPropertyIdentifier_CurrentlyActivePriority            := 517,
BACnetPropertyIdentifier_LastConfirmedServiceAccess         := 518,
BACnetPropertyIdentifier_DataRequestMode                    := 519,
BACnetPropertyIdentifier_DataRequestCycle                   := 520,
```

```
  BACnetPropertyIdentifier_FileName                          := 521
)
END_TYPE
```

4.9.29 E_BACNETRELIABILITY

Auflistung der möglichen Werte der BACnet-Property *Reliability* (Auszug).

```
TYPE E_BACNETRELIABILITY :
(
  BACnetReliability_no_fault_detected      := 0,
  BACnetReliability_no_sensor              := 1,
  BACnetReliability_over_range             := 2,
  BACnetReliability_under_range            := 3,
  BACnetReliability_open_loop              := 4,
  BACnetReliability_shorted_loop           := 5,
  BACnetReliability_no_output              := 6,
  BACnetReliability_unreliable_other       := 7,
  BACnetReliability_process_error          := 8,
  BACnetReliability_multi_state_fault      := 9,
  BACnetReliability_configuration_error    := 10,
  BACnetReliability_reserved11             := 11,
  BACnetReliability_communication_failure  := 12,
  BACnetReliability_member_fault           := 13
)
END_TYPE
```

`BACnetReliability_no_fault_detected`: NO_FAULT_DETECTED

`BACnetReliability_no_sensor`: NO_SENSOR

`BACnetReliability_over_range`: OVER_RANGE

`BACnetReliability_under_range`: UNDER_RANGE

`BACnetReliability_open_loop`: OPEN_LOOP

`BACnetReliability_shorted_loop`: SHORTED_LOOP

`BACnetReliability_no_output`: NO_OUTPUT

`BACnetReliability_unreliable_other`: UNRELIABLE_OTHER

`BACnetReliability_process_error`: PROCESS_ERROR

`BACnetReliability_multi_state_fault`: MULTI_STATE_FAULT

`BACnetReliability_configuration_error`: CONFIGURATION_ERROR

`BACnetReliability_reserved11`: *Reserviert für zukünftige Verwendung.*

`BACnetReliability_communication_failure`: COMMUNICATION_FAILURE

`BACnetReliability_member_fault`: MEMBER_FAULT

4.9.30 E_BACNETSEGMENTATION

PLC-Abbildung des BACnet-Datentyps BACnetSegmentation. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Segmentation_Supported*.

```
TYPE E_BACNETSEGMENTATION :
(
  BACnetSegmentation_both      := 0,
  BACnetSegmentation_transmit  := 1,
  BACnetSegmentation_receive   := 2,
  BACnetSegmentation_no        := 3
)
END_TYPE
```

`BACnetSegmentation_both`: BOTH

`BACnetSegmentation_transmit`: TRANSMIT

`BACnetSegmentation_receive`: RECEIVE

`BACnetSegmentation_no`: NO

4.9.31 E_BACNETSERVICESSUPPORTEDBITS

Bit-Belegung der Protocol-Services-Supported-Flags [▶ 369] (Property *Protocol_Services_Supported*).

```
TYPE E_BACNETSERVICESSUPPORTEDBITS :
(
  BACnetServicesSupportedBits_AcknowledgeAlarm              := 0,
  BACnetServicesSupportedBits_ConfirmedCOVNotification      := 1,
  BACnetServicesSupportedBits_ConfirmedEventNotification    := 2,
  BACnetServicesSupportedBits_GetAlarmSummary               := 3,
  BACnetServicesSupportedBits_GetEnrollmentSummary          := 4,
  BACnetServicesSupportedBits_SubscribeCOV                  := 5,
  BACnetServicesSupportedBits_AtomicReadFile                := 6,
  BACnetServicesSupportedBits_AtomicWriteFile               := 7,
  BACnetServicesSupportedBits_AddListElement                := 8,
  BACnetServicesSupportedBits_RemoveListElement             := 9,
  BACnetServicesSupportedBits_CreateObject                  := 10,
  BACnetServicesSupportedBits_DeleteObject                  := 11,
  BACnetServicesSupportedBits_ReadProperty                  := 12,
  BACnetServicesSupportedBits_ReadPropertyConditional       := 13,
  BACnetServicesSupportedBits_ReadPropertyMultiple          := 14,
  BACnetServicesSupportedBits_WriteProperty                 := 15,
  BACnetServicesSupportedBits_WritePropertyMultiple         := 16,
  BACnetServicesSupportedBits_DeviceCommunicationControl    := 17,
  BACnetServicesSupportedBits_ConfirmedPrivateTransfer      := 18,
  BACnetServicesSupportedBits_ConfirmedTextMessage          := 19,
  BACnetServicesSupportedBits_ReinitializeDevice            := 20,
  BACnetServicesSupportedBits_VtOpen                        := 21,
  BACnetServicesSupportedBits_VtClose                       := 22,
  BACnetServicesSupportedBits_VtData                        := 23,
  BACnetServicesSupportedBits_Authenticate                  := 24,
  BACnetServicesSupportedBits_RequestKey                    := 25,
  BACnetServicesSupportedBits_IAm                           := 26,
  BACnetServicesSupportedBits_IHave                         := 27,
  BACnetServicesSupportedBits_UnconfirmedCOVNotification    := 28,
  BACnetServicesSupportedBits_UnconfirmedEventNotification  := 29,
  BACnetServicesSupportedBits_UnconfirmedPrivateTransfer    := 30,
  BACnetServicesSupportedBits_UnconfirmedTextMessage        := 31,
  BACnetServicesSupportedBits_TimeSynchronization           := 32,
  BACnetServicesSupportedBits_WhoHas                        := 33,
  BACnetServicesSupportedBits_WhoIs                         := 34,
  BACnetServicesSupportedBits_ReadRange                     := 35,
  BACnetServicesSupportedBits_UtcTimeSynchronization        := 36,
  BACnetServicesSupportedBits_LifeSafetyOperation           := 37,
  BACnetServicesSupportedBits_SubscribeCOVProperty          := 38,
  BACnetServicesSupportedBits_GetEventInformation           := 39
)
END_TYPE
```

4.9.32 E_BACNETSILENCEDSTATE

PLC-Abbildung des BACnet-Datentyps BACnetSilencedState. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Silenced*.

```
TYPE E_BACNETSILENCEDSTATE :
(
  BACnetSilencedState_unsilenced        := 0,
  BACnetSilencedState_audible_silenced  := 1,
  BACnetSilencedState_visible_silenced  := 2,
  BACnetSilencedState_all_silenced      := 3
)
END_TYPE
```

`BACnetSilencedState_unsilenced`: UNSILENCED

`BACnetSilencedState_audible_silenced`: AUDIBLE_SILENCED

`BACnetSilencedState_visible_silenced`: VISIBLE_SILENCED

`BACnetSilencedState_all_silenced`: ALL_SILENCED

4.9.33 E_BACNETSTATUSFLAGS

Bit-Belegung der Status-Flags [▶ 370] (Property *Status_Flags*).

```
TYPE E_BACNETSTATUSFLAGS :
(
  BACnetStatusFlags_in_alarm        := 0,
  BACnetStatusFlags_fault           := 1,
  BACnetStatusFlags_overridden      := 2,
  BACnetStatusFlags_out_of_service  := 3
)
END_TYPE
```

`BACnetStatusFlags_in_alarm`: IN_ALARM

`BACnetStatusFlags_fault`: FAULT

`BACnetStatusFlags_overridden`: OVERRIDDEN

`BACnetStatusFlags_out_of_service`: OUT_OF_SERVICE

4.9.34 E_BACNETSTRINGENCODINGTYPES

Die Enumeration enthält eine Auflistung der vom BACnet-Treiber unterstützten Zeichenketten Encodings. Unter TwinCAT BACnet/IP ist der Standard Zeichensatz ANSI/UTF-8. Vor allem bei Fremdsystemen (Clients) kann es jedoch zur Durchmischung von Zeichensätzen kommen. Daher werden folgende Zeichensätze in der PLC mit dem Baustein FB_BACnet_StringExtDecode [▶ 323] decodiert: ANSI/UTF-8, UCS-2, UCS-4 und ISO8859-1.

Das Decoding sollte generell auf jeden gelesen String angewendet werden. Wird ein durch die PLC nicht unterstützter Zeichensatz erkannt, gibt der Decoding-Baustein einen Fehlercode aus. Die decodierte Zeichenkette wird in der Kodierung Windows-1252 (auch CP 1252) in der PLC gespeichert (1 Byte Character nach ISO 8859-1 und Ergänzungen).

```
TYPE E_BACNETSTRINGENCODINGTYPES :
(
  BACnetStringEncodingTypes_AnsiUtf8  := 0,
  BACnetStringEncodingTypes_DBCS      := 1,
  BACnetStringEncodingTypes_JIS       := 2,
  BACnetStringEncodingTypes_UCS4      := 3,
  BACnetStringEncodingTypes_UCS2      := 4,
  BACnetStringEncodingTypes_ISO8859   := 5,
)
END_TYPE
```

`BACnetStringEncodingTypes_AnsiUtf8`: ANSI X3.4 und UTF-8 Encoding nach ISO 10646 (mit Hilfe des PLC-Bausteins FB_BACnet_StringExtDecode [▶ 323] decodierbar zu Windows-1252)

`BACnetStringEncodingTypes_DBCS`: IBM™/Microsoft™ DBCS: Doppel-Byte Code mit spezifischer (u.U. proprietärer) Codepage

`BACnetStringEncodingTypes_JIS`: JIS C 6226 (japanische Schrift)

`BACnetStringEncodingTypes_UCS4`: UCS-4 nach ISO 10646: 4 Byte Unicode (mit Hilfe des PLC-Bausteins FB_BACnet_StringExtDecode [▶ 323] decodierbar zu Windows-1252)

`BACnetStringEncodingTypes_UCS2`: UCS-2 nach ISO 10646: 2 Byte Unicode (mit Hilfe des PLC-Bausteins FB_BACnet_StringExtDecode [▶ 323] decodierbar zu Windows-1252)

`BACnetStringEncodingTypes_ISO8859`: Latin-1 nach ISO 8859-1 (0x00 bis 0x7F entspricht US-ASCII; 0xA0 bis 0xFF ISO 8859-1 spezifisch). Die druckbaren Zeichen sind kompatibel zu Windows-1252.

4.9.35 E_BACNETUNIT

PLC-Abbildung des BACnet-Datentyps BACnetEngineeringUnits. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Units*.

```
TYPE E_BACNETUNIT :
(
  BACnetUnit_Area_square_meters                              := 0,
  BACnetUnit_Area_square_feet                                := 1,
  BACnetUnit_Electrical_milliamperes                         := 2,
  BACnetUnit_Electrical_amperes                              := 3,
  BACnetUnit_Electrical_ohms                                 := 4,
  BACnetUnit_Electrical_volts                                := 5,
  BACnetUnit_Electrical_kilovolts                            := 6,
  BACnetUnit_Electrical_megavolts                            := 7,
  BACnetUnit_Electrical_volt_amperes                         := 8,
  BACnetUnit_Electrical_kilovolt_amperes                     := 9,
  BACnetUnit_Electrical_megavolt_amperes                     := 10,
  BACnetUnit_Electrical_volt_amperes_reactive                := 11,
  BACnetUnit_Electrical_kilovolt_amperes_reactive            := 12,
  BACnetUnit_Electrical_megavolt_amperes_reactive            := 13,
  BACnetUnit_Electrical_degrees_phase                        := 14,
  BACnetUnit_Electrical_power_factor                         := 15,
  BACnetUnit_Energy_joules                                   := 16,
  BACnetUnit_Energy_kilojoules                               := 17,
  BACnetUnit_Energy_watt_hours                               := 18,
  BACnetUnit_Energy_kilowatt_hours                           := 19,
  BACnetUnit_Energy_btus                                     := 20,
  BACnetUnit_Energy_therms                                   := 21,
  BACnetUnit_Energy_ton_hours                                := 22,
  BACnetUnit_Enthalpy_joules_per_kilogram_dry_air            := 23,
  BACnetUnit_Enthalpy_btus_per_pound_dry_air                 := 24,
  BACnetUnit_Frequency_cycles_per_hour                       := 25,
  BACnetUnit_Frequency_cycles_per_minute                     := 26,
  BACnetUnit_Frequency_hertz                                 := 27,
  BACnetUnit_Humidity_grams_of_water_per_kilogram_dry_air    := 28,
  BACnetUnit_Humidity_percent_relative_humidity              := 29,
  BACnetUnit_Length_millimeters                              := 30,
  BACnetUnit_Length_meters                                   := 31,
  BACnetUnit_Length_inches                                   := 32,
  BACnetUnit_Length_feet                                     := 33,
  BACnetUnit_Light_watts_per_square_foot                     := 34,
  BACnetUnit_Light_watts_per_square_meter                    := 35,
  BACnetUnit_Light_lumens                                    := 36,
  BACnetUnit_Light_luxes                                     := 37,
  BACnetUnit_Light_foot_candles                              := 38,
  BACnetUnit_Mass_kilograms                                  := 39,
  BACnetUnit_Mass_pounds_mass                                := 40,
  BACnetUnit_Mass_tons                                       := 41,
  BACnetUnit_MassFlow_kilograms_per_second                   := 42,
  BACnetUnit_MassFlow_kilograms_per_minute                   := 43,
  BACnetUnit_MassFlow_kilograms_per_hour                     := 44,
  BACnetUnit_MassFlow_pounds_mass_per_minute                 := 45,
  BACnetUnit_MassFlow_pounds_mass_per_hour                   := 46,
  BACnetUnit_Power_watts                                     := 47,
  BACnetUnit_Power_kilowatts                                 := 48,
  BACnetUnit_Power_megawatts                                 := 49,
  BACnetUnit_Power_btus_per_hour                             := 50,
  BACnetUnit_Power_horsepower                                := 51,
  BACnetUnit_Power_tons_refrigeration                        := 52,
  BACnetUnit_Pressure_pascals                                := 53,
  BACnetUnit_Pressure_kilopascals                            := 54,
  BACnetUnit_Pressure_bars                                   := 55,
  BACnetUnit_Pressure_pounds_force_per_square_inch           := 56,
  BACnetUnit_Pressure_centimeters_of_water                   := 57,
  BACnetUnit_Pressure_inches_of_water                        := 58,
  BACnetUnit_Pressure_millimeters_of_mercury                 := 59,
  BACnetUnit_Pressure_centimeters_of_mercury                 := 60,
  BACnetUnit_Pressure_inches_of_mercury                      := 61,
  BACnetUnit_Temperature_degrees_Celsius                     := 62,
  BACnetUnit_Temperature_degrees_Kelvin                      := 63,
  BACnetUnit_Temperature_degrees_Fahrenheit                  := 64,
  BACnetUnit_Temperature_degree_days_Celsius                 := 65,
  BACnetUnit_Temperature_degree_days_Fahrenheit              := 66,
  BACnetUnit_Time_years                                      := 67,
  BACnetUnit_Time_months                                     := 68,
  BACnetUnit_Time_weeks                                      := 69,
  BACnetUnit_Time_days                                       := 70,
  BACnetUnit_Time_hours                                      := 71,
  BACnetUnit_Time_minutes                                    := 72,
  BACnetUnit_Time_seconds                                    := 73,
  BACnetUnit_Velocity_meters_per_second                      := 74,
  BACnetUnit_Velocity_kilometers_per_hour                    := 75,
  BACnetUnit_Velocity_feet_per_second                        := 76,
  BACnetUnit_Velocity_feet_per_minute                        := 77,
```

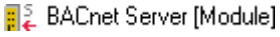
```
BACnetUnit_Velocity_miles_per_hour                      := 78,
BACnetUnit_Volume_cubic_feet                            := 79,
BACnetUnit_Volume_cubic_meters                          := 80,
BACnetUnit_Volume_imperial_gallons                      := 81,
BACnetUnit_Volume_liters                                := 82,
BACnetUnit_Volume_us_gallons                            := 83,
BACnetUnit_VolumetricFlow_cubic_feet_per_minute         := 84,
BACnetUnit_VolumetricFlow_cubic_meters_per_second       := 85,
BACnetUnit_VolumetricFlow_imperial_gallons_per_minute   := 86,
BACnetUnit_VolumetricFlow_liters_per_second             := 87,
BACnetUnit_VolumetricFlow_liters_per_minute             := 88,
BACnetUnit_VolumetricFlow_us_gallons_per_minute         := 89,
BACnetUnit_Other_degrees_angular                        := 90,
BACnetUnit_Other_degrees_Celsius_per_hour               := 91,
BACnetUnit_Other_degrees_Celsius_per_minute             := 92,
BACnetUnit_Other_degrees_Fahrenheit_per_hour            := 93,
BACnetUnit_Other_degrees_Fahrenheit_per_minute          := 94,
BACnetUnit_Other_no_units                               := 95,
BACnetUnit_Other_parts_per_million                      := 96,
BACnetUnit_Other_parts_per_billion                      := 97,
BACnetUnit_Other_percent                                := 98,
BACnetUnit_Other_percent_per_second                     := 99,
BACnetUnit_Other_per_minute                             := 100,
BACnetUnit_Other_per_second                             := 101,
BACnetUnit_Other_psi_per_degree_Fahrenheit              := 102,
BACnetUnit_Other_radians                                := 103,
BACnetUnit_Other_revolutions_per_minute                 := 104,
BACnetUnit_Currency_currency1                           := 105,
BACnetUnit_Currency_currency2                           := 106,
BACnetUnit_Currency_currency3                           := 107,
BACnetUnit_Currency_currency4                           := 108,
BACnetUnit_Currency_currency5                           := 109,
BACnetUnit_Currency_currency6                           := 110,
BACnetUnit_Currency_currency7                           := 111,
BACnetUnit_Currency_currency8                           := 112,
BACnetUnit_Currency_currency9                           := 113,
BACnetUnit_Currency_currency10                          := 114,
BACnetUnit_Area_square_inches                           := 115,
BACnetUnit_Area_square_centimeters                      := 116,
BACnetUnit_Enthalpy_btus_per_pound                      := 117,
BACnetUnit_Length_centimeters                           := 118,
BACnetUnit_MassFlow_pounds_mass_per_second              := 119,
BACnetUnit_Temperature_delta_degrees_Fahrenheit         := 120,
BACnetUnit_Temperature_delta_degrees_Kelvin             := 121,
BACnetUnit_Electrical_kilohms                           := 122,
BACnetUnit_Electrical_megohms                           := 123,
BACnetUnit_Electrical_millivolts                        := 124,
BACnetUnit_Energy_kilojoules_per_kilogram               := 125,
BACnetUnit_Energy_megajoules                            := 126,
BACnetUnit_Entropy_joules_per_degree_Kelvin             := 127,
BACnetUnit_Entropy_joules_per_kilogram_degree_Kelvin    := 128,
BACnetUnit_Frequency_kilohertz                          := 129,
BACnetUnit_Frequency_megahertz                          := 130,
BACnetUnit_Frequency_per_hour                           := 131,
BACnetUnit_Power_milliwatts                             := 132,
BACnetUnit_Pressure_hectopascals                        := 133,
BACnetUnit_Pressure_millibars                           := 134,
BACnetUnit_VolumetricFlow_cubic_meters_per_hour         := 135,
BACnetUnit_VolumetricFlow_liters_per_hour               := 136,
BACnetUnit_Other_kilowatt_hours_per_square_meter        := 137,
BACnetUnit_Other_kilowatt_hours_per_square_foot         := 138,
BACnetUnit_Other_megajoules_per_square_meter            := 139,
BACnetUnit_Other_megajoules_per_square_foot             := 140,
BACnetUnit_Other_watts_per_square_meter_degree_kelvin   := 141,
BACnetUnit_VolumetricFlow_cubic_feet_per_second         := 142,
BACnetUnit_Other_percent_obscuration_per_foot           := 143,
BACnetUnit_Other_percent_obscuration_per_meter          := 144,
BACnetUnit_Electrical_milliohms                         := 145,
BACnetUnit_Energy_megawatt_hours                        := 146,
BACnetUnit_Energy_kilo_btus                             := 147,
BACnetUnit_Energy_mega_btus                             := 148,
BACnetUnit_Enthalpy_kilojoules_per_kilogram_dry_air     := 149,
BACnetUnit_Enthalpy_megajoules_per_kilogram_dry_air     := 150,
BACnetUnit_Entropy_kilojoules_per_degree_Kelvin         := 151,
BACnetUnit_Entropy_megajoules_per_degree_Kelvin         := 152,
BACnetUnit_Force_newton                                 := 153,
BACnetUnit_MassFlow_grams_per_second                    := 154,
BACnetUnit_MassFlow_grams_per_minute                    := 155,
BACnetUnit_MassFlow_tons_per_hour                       := 156,
BACnetUnit_Power_kilo_btus_per_hour                     := 157,
```

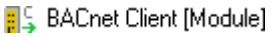
```
BACnetUnit_Time_hundredths_seconds                      := 158,
BACnetUnit_Time_milliseconds                            := 159,
BACnetUnit_Torque_newton_meters                         := 160,
BACnetUnit_Velocity_millimeters_per_second              := 161,
BACnetUnit_Velocity_millimeters_per_minute              := 162,
BACnetUnit_Velocity_meters_per_minute                   := 163,
BACnetUnit_Velocity_meters_per_hour                     := 164,
BACnetUnit_VolumetricFlow_cubic_meters_per_minute       := 165,
BACnetUnit_Acceleration_meters_per_second_per_second    := 166,
BACnetUnit_Electrical_amperes_per_meter                 := 167,
BACnetUnit_Electrical_amperes_per_square_meter          := 168,
BACnetUnit_Electrical_ampere_square_meters              := 169,
BACnetUnit_Electrical_farads                            := 170,
BACnetUnit_Electrical_henrys                            := 171,
BACnetUnit_Electrical_ohm_meters                        := 172,
BACnetUnit_Electrical_siemens                           := 173,
BACnetUnit_Electrical_siemens_per_meter                 := 174,
BACnetUnit_Electrical_teslas                            := 175,
BACnetUnit_Electrical_volts_per_degree_Kelvin           := 176,
BACnetUnit_Electrical_volts_per_meter                   := 177,
BACnetUnit_Electrical_webers                            := 178,
BACnetUnit_Light_candelas                               := 179,
BACnetUnit_Light_candelas_per_square_meter              := 180,
BACnetUnit_Temperature_degrees_Kelvin_per_hour          := 181,
BACnetUnit_Temperature_degrees_Kelvin_per_minute        := 182,
BACnetUnit_Other_joule_seconds                          := 183,
BACnetUnit_Other_radians_per_second                     := 184,
BACnetUnit_Other_square_meters_per_Newton               := 185,
BACnetUnit_Other_kilograms_per_cubic_meter              := 186,
BACnetUnit_Other_newton_seconds                         := 187,
BACnetUnit_Other_newtons_per_meter                      := 188,
BACnetUnit_Other_watts_per_meter_per_degree_Kelvin      := 189,
BACnetUnit_micro_siemens                                := 190,
BACnetUnit_cubic_feet_per_hour                          := 191,
BACnetUnit_us_gallons_per_hour                          := 192,
BACnetUnit_kilometers                                   := 193,
BACnetUnit_micrometers                                  := 194,
BACnetUnit_grams                                        := 195,
BACnetUnit_milligrams                                   := 196,
BACnetUnit_milliliters                                  := 197,
BACnetUnit_milliliters_per_second                       := 198,
BACnetUnit_decibels                                     := 199,
BACnetUnit_decibels_millivolt                           := 200,
BACnetUnit_decibels_volt                                := 201,
BACnetUnit_millisiemens                                 := 202,
BACnetUnit_watt_hours_reactive                          := 203,
BACnetUnit_kilowatt_hours_reactive                      := 204,
BACnetUnit_megawatt_hours_reactive                      := 205,
BACnetUnit_millimeters_of_water                         := 206,
BACnetUnit_per_mille                                    := 207,
BACnetUnit_grams_per_gram                               := 208,
BACnetUnit_kilograms_per_kilogram                       := 209,
BACnetUnit_grams_per_kilogram                           := 210,
BACnetUnit_milligrams_per_gram                          := 211,
BACnetUnit_milligrams_per_kilogram                      := 212,
BACnetUnit_grams_per_milliliter                         := 213,
BACnetUnit_grams_per_liter                              := 214,
BACnetUnit_milligrams_per_liter                         := 215,
BACnetUnit_micrograms_per_liter                         := 216,
BACnetUnit_grams_per_cubic_meter                        := 217,
BACnetUnit_milligrams_per_cubic_meter                   := 218,
BACnetUnit_micrograms_per_cubic_meter                   := 219,
BACnetUnit_nanograms_per_cubic_meter                    := 220,
BACnetUnit_grams_per_cubic_centimeter                   := 221,
BACnetUnit_becquerels                                   := 222,
BACnetUnit_kilobecquerels                               := 223,
BACnetUnit_megabecquerels                               := 224,
BACnetUnit_gray                                         := 225,
BACnetUnit_milligray                                    := 226,
BACnetUnit_microgray                                    := 227,
BACnetUnit_sieverts                                     := 228,
BACnetUnit_millisieverts                                := 229,
BACnetUnit_microsieverts                                := 230,
BACnetUnit_microsieverts_per_hour                       := 231,
BACnetUnit_decibels_a                                   := 232,
BACnetUnit_nephelometric_turbidity_unit                 := 233,
BACnetUnit_pH                                           := 234,
BACnetUnit_grams_per_square_meter                       := 235,
```

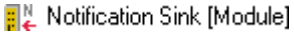
```
  BACnetUnit_minutes_per_degree_kelvin                    := 236
)
END_TYPE
```

4.9.36 ST_BACnet_AdsConnection

Struktur mit den Verbindungsinformationen eines ADS Servers des BACnet-Treiber. Unterstützt werden 3 Arten von ADS-Servern unter BACnet:

Server → ein BACnet Server 

Client → ein BACnet Client der einen entfernten BACnet Server repräsentiert 

Notification-Sink → eine BACnet Nachrichten "Senke" (Empfang und Abonnieren von BACnet Nachrichten)



```
TYPE ST_BACnet_AdsConnection :
STRUCT
  bValid     : BOOL;
  nReload    : USINT;
  sAmsNetId  : T_AmsNetId;
  nAmsPort   : T_AmsPort:=1000;
  bServer    : BOOL;
  bClient    : BOOL;
  bNSink     : BOOL;
  nDeviceId  : UDINT;
END_STRUCT
END_TYPE
```

bValid: Enthaltene Verbindungsdaten sind gültig

`nReload`: Trigger für das automatische Nachladen von Daten. Bei Änderung von **nReload** wurde die Verbindung zum ADS Server neu aufgebaut bzw. wurden Verbindungsdaten aktualisiert. Funktionsbausteine die *ST_BACnet_AdsConnection* auswerten, laden die Daten (z.B. einer Property) erneut aus dem entsprechenden ADS Server, wenn der Eingang `bAutoReload` an diesem Baustein auf *TRUE* gesetzt ist.

`sAmsNetId`: AMS-NetID des BACnet-Adapters.

`nAmsPort`: AMS-Port des BACnet-Server, BACnet-Client oder BACnet-Notification-Sink (typischerweise ≥ 1000).

`bServer`: ADS-Server vom Typ BACnet-Server

`bClient`: ADS-Server vom Typ BACnet-Client

`bNSink`: ADS-Server vom Typ BACnet-Notification-Sink

`nDeviceId`: BACnet-ID des BACnet-Server bzw. BACnet-Instance-Number des BACnet-Device Objekts (nur bei BACnet-Server oder BACnet-Client)

4.9.37 ST_BACnet_CharacterStringExt

Rohdatenstruktur der Property mit BACnet Datentyp CharacterString. Die Bausteine FB_BACnet_StringExtDecode [▶ 323] und FB_BACnet_StringExtEncode [▶ 324] decodieren und codieren Daten dieser Struktur in/von in der PLC anzeig- und bearbeitbare Strings mit dem Encoding *Windows-1252* bzw. *CP1252* (Westeuropäischer Windows-Zeichensatz).

```
TYPE ST_BACnet_CharacterStringExt :
STRUCT
  cookie      : BYTE;
  encoding    : BYTE;
  strLen      : UINT;
```

```
  stringData  : ARRAY[1..BACnet_STRING_MAXLENGTH] OF BYTE;
END_STRUCT
END_TYPE
```

`cookie`: 0 und **strLen** > 0 → kein ANSI/UTF-8 String-Encoding; sonst: Ab Adresse von **cookie** wird ein ANSI/UTF-8 String erwartet.

`encoding`: Wenn kein ANSI/UTF-8 String, dann ist hier das String-Encoding enthalten (siehe E_BACNETSTRINGENCODINGTYPES [▶ 354])

`strLen`: Wenn kein ANSI/UTF-8 String, dann ist die Byte-Länge der Daten in **stringData** enthalten

`stringData`: Wenn kein ANSI/UTF-8 String: String-Daten

4.9.38 ST_BACnet_CharacterStringExtListEntry

Rohdaten eines Eintrags der Property mit dem BACnet-Type List of CharacterString.

```
TYPE ST_BACnet_CharacterStringExtListEntry :
STRUCT
  nEntrySize  : DWORD;
  hdrInfo     : DWORD;
  extString   : ST_BACnet_CharacterStringExt;
END_STRUCT
END_TYPE
```

`nEntrySize`: Byte-Größe der Daten

`hdrInfo`: Info Flags

4.9.39 ST_BACnet_Date

PLC-Abbildung des BACnet-Datentyps Date. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime.

```
TYPE ST_BACnet_Date :
STRUCT
  nYear       : BYTE:=16#FF;
  nMonth      : BYTE:=16#FF;
  nDay        : BYTE:=16#FF;
  nDayOfWeek  : BYTE:=16#FF;
END_STRUCT
END_TYPE
```

`nYear`: Jahr minus 1900; 255 steht für undefiniert (bzw. jedes Jahr)

`nMonth`: Mögliche Werte für Monat(e): 1 bis 14; 1 → Januar usw.; 13 → alle ungeraden Monate; 14 → alle geraden Monate; 255 steht für undefiniert (bzw. jeder Monate)

`nDay`: Mögliche Werte für Tag(e) des Monats: 1 bis 32; 1 → erster Tag usw.; 32 → letzter Tag des Monats; 255 steht für undefiniert (bzw. jeder Tag)

`nDayOfWeek`: Mögliche Werte für Tag(e) der Woche: 1 bis 7; 1 → Montag usw.; 7 → Sonntag; 255 steht für undefiniert (bzw. jeder Tag)

4.9.40 ST_BACnet_DateTime

PLC-Abbildung des BACnet-Datentyps BACnetDateTime. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime.

```
TYPE ST_BACnet_DateTime :
STRUCT
  stDate  : ST_BACnet_Date;
  stTime  : ST_BACnet_Time;
END_STRUCT
END_TYPE
```

Achtung:

Werden BACnetDateTime Werte für das Schalten in z.B. Zeitschaltplänen verwendet, so sollte folgender, beispielhafte Zeitschaltpunkt vermieden werden:

Mo. 0:00.00 - ACTIVE → Mo. 23:59.59 - INACTIVE

Grund: Das Umschalten auf den Zustand INACTIVE wird unter Umständen nicht erkannt. Der Zeitschaltpunkt liegt unmittelbar vor der Tagesumschaltung. Ändert der Tag in diesem Beispiel von Montag auf Dienstag bevor die Uhrzeitänderung von 23:59.58 auf 23:59.59 erkannt wird, so wird der Eintrag für Montag 23:59.59 auf INACTIVE nicht mehr ausgeführt. Dieser Effekt kann bei besonders stark ausgelasteten Controllern auftreten oder bei der Verwendung von moderaten BACnet Zykluszeiten von >> 100ms.

Um Schaltzeiträume zu setzen, die einen kompletten Tag (24h) umfassen sollen, sind alternativ folgende Zeitschaltpunkte zu verwenden:

Mo. 0:00.00 - ACTIVE → **Di.** 0:00.00 - INACTIVE

oder (z.B. 5 Minuten vor 0 Uhr):

Mo. 0:00.00 - ACTIVE → Mo. **23:55.00** - INACTIVE

4.9.41 ST_BACnet_DiagEthStatistics

Struktur mit Informationen zum Ethernet-Adapter.

```
TYPE ST_BACnet_DiagEthStatistics :
STRUCT
  ethStat     : ST_BACnet_TcIoEthStatistic;
  txRxErrCnt  : ST_BACnet_TcIoEthTxRxErrorCount;
END_STRUCT
END_TYPE
```

4.9.42 ST_BACnet_Diagnosis

Struktur mit Informationen aus dem BACnet Geräte Adapter. Enthalten sind Zeitverhalten, Statistiken für Server und Client und Diagnoseinformationen zum Netzwerkverkehr.

```
TYPE ST_BACnet_Diagnosis :
STRUCT
  ethDevice         : ST_BACnet_DiagEthStatistics;
  execTiming        : ST_BACnet_DiagnosisTiming;
  frameStatistics   : ST_BACnet_FrameStatistics;
  nRes1             : DWORD;
  info              : ST_BACnet_Info;
  nRes2             : DWORD;
  serverInfo        : ST_BACnet_ServerStatistics;
  lastUpdate        : ST_BACnet_DateTime;
  nRes3             : DWORD;
END_STRUCT
END_TYPE
```

lastUpdate: Zeitstempel der Diagnosedaten (Zeitpunkt, an dem die Daten gelesen wurden).

4.9.43 ST_BACnet_DiagnosisTiming

Struktur mit Informationen zu aktuellen Ausführungszeiten des BACnet-Treibers.

```
TYPE ST_BACnet_DiagnosisTiming :
STRUCT
  msIoInCurExecutionTime    : DWORD;
  msIoInMinExecutionTime    : DWORD;
  msIoInMaxExecutionTime    : DWORD;
  msIoOutCurExecutionTime   : DWORD;
  msIoOutMinExecutionTime   : DWORD;
  msIoOutMaxExecutionTime   : DWORD;
  msCycleCurExecutionTime   : DWORD;
  msCycleMinExecutionTime   : DWORD;
  msCycleMaxExecutionTime   : DWORD;
  msInputCurDistanceTime    : DWORD;
  msInputMinDistanceTime    : DWORD;
```

```
  msInputMaxDistanceTime   : DWORD;
END_STRUCT
END_TYPE
```

msIoInCurExecutionTime: Aktuelle Verarbeitungsdauer der Eingangsdaten im BACnet-Task (ms).

`msIoInMinExecutionTime`: Minimale Verarbeitungsdauer der Eingangsdaten im BACnet-Task (ms).

`msIoInMaxExecutionTime`: Maximale Verarbeitungsdauer der Eingangsdaten im BACnet-Task (ms).

`msIoOutCurExecutionTime`: Aktuelle Verarbeitungsdauer der Ausgangsdaten im BACnet-Task (ms).

`msIoOutMinExecutionTime`: Minimale Verarbeitungsdauer der Ausgangsdaten im BACnet-Task (ms).

`msIoOutMaxExecutionTime`: Maximale Verarbeitungsdauer der Ausgangsdaten im BACnet-Task (ms).

`msCycleCurExecutionTime`: Aktuelle Verarbeitungsdauer der BACnet-Task (ms).

`msCycleMinExecutionTime`: Minimale Verarbeitungsdauer der BACnet-Task (ms).

`msCycleMaxExecutionTime`: Maximale Verarbeitungsdauer der BACnet-Task (ms).

`msInputCurDistanceTime`: Aktuelle Zeitdauer zwischen dem Beginn der Eingangsdatenverarbeitung (ms).

`msInputMinDistanceTime`: Minimale Zeitdauer zwischen dem Beginn der Eingangsdatenverarbeitung (ms).

`msInputMaxDistanceTime`: Maximale Zeitdauer zwischen dem Beginn der Eingangsdatenverarbeitung (ms).

4.9.44 ST_BACnet_EventTransitionBits

PLC-Abbildung des BACnet-Datentyps BACnetEventTransitionBits (Properties *Event_Enable* und *Acked_Transitions*).

```
TYPE ST_BACnet_EventTransitionBits :
STRUCT
  to_offnormal  : BOOL;
  to_fault      : BOOL;
  to_normal     : BOOL;
END_STRUCT
END_TYPE
```

`to_offnormal`: Übergang von X nach OFF_NORMAL.

`to_fault`: Übergang von X nach FAULT.

`to_normal`: Übergang von X nach NORMAL.

4.9.45 ST_BACnet_ExceptionScheduleBool

Struktur für den Datenaustausch der Property *exception_schedule* mit Hilfe des Funktionsbausteins FB_BACnet_ExceptionScheduleProperty [▶ 285].

```
TYPE ST_BACnet_ExceptionScheduleBool :
STRUCT
  bReqGet     : BOOL;
  bReqSet     : BOOL;
  nGet        : USINT;
  nSet        : USINT;
  bBusy       : BOOL;
  arrEntries  : ARRAY[0..BACnet_MaxExSchEntries] OF ST_BACnet_ExceptionScheduleEntryBool;
END_STRUCT
END_TYPE
```

`bReqGet`: Anforderung Property-Daten lesen *FALSE* → *TRUE* an den zugehörigen Property-Funktionsbaustein (siehe FB_BACnet_ExceptionScheduleProperty [▶ 285]). Der Flankenwechsel *TRUE* → *FALSE* signalisiert die Übernahme der Anforderung vom zugehörigen Property-Funktionsbaustein. Aktuelle Daten stehen demnächst zur Verfügung. Das erfolgreiche Laden der Property-Daten wird mittels **nGet** signalisiert.

`bReqSet`: Anforderung Property-Daten schreiben *FALSE* → *TRUE* an den zugehörigen Property-Funktionsbaustein (siehe FB_BACnet_ExceptionScheduleProperty [▶ 285]). Der Flankenwechsel *TRUE* → *FALSE* signalisiert die Übernahme der Anforderung vom zugehörigen Property-Funktionsbaustein. Aktuelle Daten stehen demnächst zur Verfügung. Das erfolgreiche Schreiben der Property-Daten wird mittels **nSet** signalisiert.

`nGet`: Zähler wird um 1 erhöht, wenn Property-Daten erfolgreich gelesen wurden. Der Zähler ist umlaufend.

`nSet`: Zähler wird um 1 erhöht, wenn Property-Daten erfolgreich geschrieben wurden. Der Zähler ist umlaufend.

`bBusy`: Zugehöriger Property-Funktionsbaustein lädt bzw. schreibt Daten aus bzw. in die Property. Die Daten sind ungültig, während **bBusy** = *TRUE* gesetzt ist.

`arrEntries`: Array mit Daten der Property *Exception_Schedule*. Es stehen 0 bis BACnet_MaxExSchEntries Einträge zur Verfügung. Die Einträge werden als BACnetTimeValue codiert und beschränken sich auf den Datentyp Bool bzw. Null. Einträge mit anderen Datentypen werden ignoriert. Zudem werden jedoch sämtliche Typen von Kalendereinträgen unterstützt.

4.9.46 ST_BACnet_ExceptionScheduleEntryBool

```
TYPE ST_BACnet_ExceptionScheduleEntryBool :
STRUCT
  eEntryType    : E_BACnetCalendarEntryType;
  ePriority     : E_BACnetPriority;
  stDate        : ST_BACnet_ExceptionScheduleCalDate;
  stWeekNDay    : ST_BACnet_ExceptionScheduleCalWnD;
  stCalRef      : ST_BACnet_ExceptionScheduleCalRef;
  arrTimeValue  : ARRAY[0..BACnet_MaxTimeValues] OF ST_BACnet_TimeValueBool;
END_STRUCT
END_TYPE
```

`eEntryType`: Typ des Kalender-Eintrags. Mögliche Werte:

- `BACnetCalendarEntryType_None` Eintrag leer.
- `BACnetCalendarEntryType_Ref` Ausnahme Zeitschaltdaten werden aus einem Kalender-Objekt gelesen (Verweis befindet sich in **stCalRef**).
- `BACnetCalendarEntryType_Date` Ausnahme Zeitschaltdatum wird aus **stDate.stStart** gelesen.
- `BACnetCalendarEntryType_DateRange` Ausnahme Zeitschaltdatumszeitraum wird aus **stDate.stStart** und **stDate.stEnd** gelesen.
- `BACnetCalendarEntryType_WeekNDay` Ausnahme Zeitschalttag wird aus **stWeekNDay** gelesen.

ePriority: Angabe der Priorität mit der der Zeitschaltwert in eine kommandierbare Property geschrieben werden soll.

`stDate`: Datum bzw. Datumszeitraum:

```
TYPE ST_BACnet_ExceptionScheduleCalDate :
STRUCT
  stStart : ST_BACnet_Date;
  stEnd   : ST_BACnet_Date;
END_STRUCT
END_TYPE
```

stWeekNDay: Woche und Tag Zeitraum.

stCalRef: Zeiträume werden aus einem BACnet Objekt vom Typ *Calendar* gelesen.

arrTimeValue: Es stehen pro Ausnahmezeitpunkt 0 bis BACnet_MaxTimeValues Einträge zur Verfügung. Die Einträge werden als BACnetTimeValue codiert und beschränken sich auf den Datentyp Bool bzw. Null. Einträge mit anderen Datentypen werden ignoriert.

4.9.47 ST_BACnet_FrameStatistics

Struktur mit Informationen zu BACnet Frames.

```
TYPE ST_BACnet_FrameStatistics :
STRUCT
  confirmed              : ARRAY[0..29] OF ST_BACnet_ConfirmedServiceDiag;
  unConfirmed            : ARRAY[0..9] OF ST_BACnet_UnConfirmedServiceDiag;
  nSegmSendTimeouts      : DWORD;
  nFrameAllocFailCnt     : DWORD;
  nFrameSendCnt          : DWORD;
  nFrameSendFailCnt      : DWORD;
  nFrameRecvCnt          : DWORD;
  nFrameRecvFailCnt      : DWORD;
  nLinkStatusChangedCnt  : DWORD;
END_STRUCT
END_TYPE
```

`nSegmSendTimeouts`: Zeitüberschreitung beim Empfang der Bestätigung von segmentierten Paketen (Gegenstelle quittiert eine segmentierte Antwort nicht, Segmented-ACK)

`nFrameAllocFailCnt`: Anzahl verlorener Frames wegen Problemen bei der Speicherreservierung (kein freier Speicher).

`nFrameSendCnt`: Anzahl gesendeter Frames.

`nFrameSendFailCnt`: Anzahl fehlgeschlagener Frames.

`nFrameRecvCnt`: Anzahl empfangener Frames.

`nFrameRecvFailCnt`: Anzahl fehlerhaft empfangener Frames.

`nLinkStatusChangedCnt`: Häufigkeit der Netzwerk-Statusänderung (Kabel gezogen/Link lost, Kabel gesteckt/Link OK).

4.9.48 ST_BACnet_GlobalAdsBuffer

Datenstruktur des globalen PLC ADS Puffer. Dieser Puffer wird von folgenden Funktionsbausteinen für das Puffern des ADS-Datenstroms für das Lesen und Schreiben von komplexen Properties verwendet:

- FB_BACnet_EventMessageTextsProperty [▶ 283]
- FB_BACnet_ExceptionScheduleProperty [▶ 285]
- FB_BACnet_LogBufferProperty [▶ 289]
- FB_BACnet_ObjectListProperty [▶ 292]
- FB_BACnet_RecipientListProperty [▶ 296]
- FB_BACnet_WeeklyScheduleProperty [▶ 299]

Die Puffergröße beträgt 8KByte. Funktionsbausteine, die den Puffer verwenden, sorgen mit Hilfe der Funktion *TestAndSet* aus der gleichnamigen Bibliothek für das Sperren gegen gleichzeitigen Zugriff. Das bedeutet jedoch, dass mehrere Lese- bzw. Schreibzugriffe mit den oben genannten Funktionsbausteinen nicht parallel, sondern sequenziell abgearbeitet werden. Die Zugriffkontrolle wird funktionsbausteinintern gehandhabt.

```
TYPE ST_BACnet_GlobalAdsBuffer :
STRUCT
  bTestSet : BOOL;
  iLocked  : DINT;
```

```
  arrData  : ARRAY[0..8191] OF BYTE;
END_STRUCT
END_TYPE
```

`bTestSet`: Sperr-Bit der Funktion TestAndSet.

`iLocked`: Aktuell gesperrte Datengröße im Puffer.

`arrData`: Datenpuffer (8192 Byte).

4.9.49 ST_BACnet_Info

Struktur mit allgemeinen Informationen zur BACnet-Task.

```
TYPE ST_BACnet_Info :
STRUCT
  nAmsAllocCnt                : DINT;
  nAmsAllocCntMax             : DWORD;
  nAmsAllocFailCnt            : DWORD;
  nAvailPhysMemBytes          : DWORD;
  nAvailFlashMemBytes         : ARRAY[0..1] OF DWORD;
  nRecvFrameFifoSize          : DWORD;
  nEmptyFrameFifoSize         : DWORD;
  nRecvSegmUDPFramesListSize  : DWORD;
  nRecvSegmFramesListSize     : DWORD;
  nSendSegmFramesListSize     : DWORD;
  nClientScanListSize         : DWORD;
  nAmsRecvMissCount           : DWORD;
END_STRUCT
END_TYPE
```

`nAmsAllocCnt`: Anzahl reservierter Router-Speicher Blöcke (1 Block = 1024 Byte). Die Anzahl sollte nicht kontinuierlich steigen (Maximal 2000 Blöcke=2MByte sind per Default verfügbar). Andernfalls den Router-Speicher im System Manager erhöhen (siehe Bild-1).

`nAmsAllocCntMax`: Spitzenwert der gleichzeitig reservierten Router-Speicher Blöcke.

`nAmsAllocFailCnt`: Anzahl fehlgeschlagener Router-Speicher Reservierungen (Reservierung schlägt fehl, wenn keine freien Blöcke mehr verfügbar).

`nAvailPhysMemBytes`: Freier, physikalischer Speicher des Betriebssystems in Byte.

`nAvailFlashMemBytes`: Freier Festplattenspeicher des Betriebssystems in Byte.

`nRecvFrameFifoSize`: Anzahl der aktuell zwischengespeicherten Frames in der receive Frame Queue (Receive Queue Size unter "Network Adapter Settings")

`nEmptyFrameFifoSize`: Anzahl des noch freien Speichers für empfangene Frames um diese Zwischenspeichern zu können (Frame wird kopiert und dann in RecvFrameFifo kopiert)

`nRecvSegmUDPFramesListSize`: Anzahl der aktuell empfangenen und zwischengespeicherten IP-Segmente (Name ist falsch, nicht UDP sondern IP)

`nRecvSegmFramesListSize`: Anzahl der aktuell empfangenen und zwischengespeicherten BACnet Frame Segmente

`nSendSegmFramesListSize`: Anzahl der aktuell BACnet-Frame Segmente die vor dem Senden zwischengespeichert sind

`nClientScanListSize`: Anzahl der aktuell bekannten BACnet-Geräte im Netzwerk (DeviceAddressBinding im Device-Objekt wird daraus gebildet)

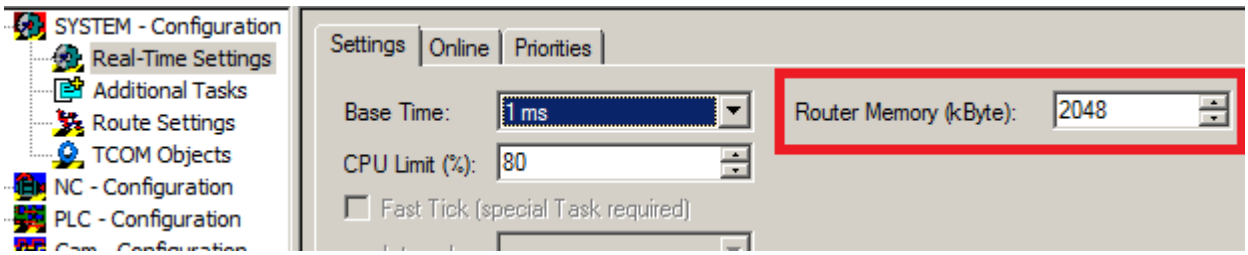


Abb. 79: Bild-1: AMS Router-Speicher Einstellung

4.9.50 ST_BACnet_LimitEnable

PLC-Abbildung des BACnet-Datentyps BACnetLimitEnable. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Limit_Enable*.

```
TYPE ST_BACnet_LimitEnable :
STRUCT
  lowLimitEnable   : BOOL;
  highLimitEnable  : BOOL;
END_STRUCT
END_TYPE
```

`lowLimitEnable`: Unterer Grenzwertüberwachung aktiv (siehe Property *Low_Limit*)

`highLimitEnable`: Obere Grenzwertüberwachung aktiv (siehe Property *High_Limit*)

Bei Grenzwertverletzung wird die Property *Event_State* entsprechend gesetzt (siehe E_BACNETEVENTSTATE [▶ 339]).

4.9.51 ST_BACnet_LogBufferEntryReal

PLC-Abbildung des BACnet-Datentyps BACnetLogRecord für *Log_Datum* vom Typ *Real*. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Log_Buffer*.

```
TYPE ST_BACnet_LogBufferEntryReal :
STRUCT
  nTimeStamp   : DWORD;
  stDateTime   : ST_BACnet_DateTime;
  fValue       : REAL;
END_STRUCT
END_TYPE
```

`nTimeStamp`: Zeitstempel in 100 ms seit dem Jahr 1900

`fValue`: Wert des Log-Eintrags

4.9.52 ST_BACnet_LogBufferReal

Struktur für den Datenaustausch der Property *Log_Buffer* mit Hilfe des Funktionsbausteins FB_BACnet_LogBufferProperty.

```
TYPE ST_BACnet_LogBufferReal :
STRUCT
  nGet          : USINT;
  sObjectDesc   : T_MAXSTRING;
  nCount        : UDINT;
  arrEntries    : ARRAY[0..BACnet_MaxLogBufferEntries] OF ST_BACnet_LogBufferEntryReal;
END_STRUCT
END_TYPE
```

`nGet`: Zähler wird um 1 erhöht, wenn Property-Daten erfolgreich gelesen wurden. Der Zähler ist umlaufend.

`sObjectDesc`: Wert der Property *Description* des zugehörigen BACnet Objekts. Ist die Property *Description* leer, wird der Inhalt der Property *Object_Name* gelesen und eingetragen.

`nCount`: Anzahl Einträge die gelesen wurden. 0 bis **nCount**-1 in **arrEntries** enthalten gültige Werte.

`arrEntries`: Array mit Daten der Property *Log_Buffer*. Es stehen 0 bis **nCount**-1 Einträge zur Verfügung. Einträge beschränken sich auf den Typ Real der Property *Log_Buffer*.

4.9.53 ST_BACnet_NSinkEvent

PLC-Abbildung der Daten eines Event-Eintrags der BACnet-Notification-Sink.

```
TYPE ST_BACnet_NSinkEvent :
STRUCT
  nProcessID    : UDINT;
  nDeviceID     : UDINT;
  tObjectID     : T_BACnet_ObjectIdentifier;
  stDateTime    : ST_BACnet_DateTime;
  nNC           : UDINT;
  nPriotity     : UDINT;
  eEventType    : E_BACNETEVENTTYPE;
  eNotifyType   : E_BACNETNOTIFYTYPE;
  eFromState    : E_BACNETEVENTSTATE;
  eToState      : E_BACNETEVENTSTATE;
  sMessage      : T_MaxString;
  bAcknReq      : BOOL;
END_STRUCT
END_TYPE
```

`nProcessID`: Identifikationsnummer des Empfänger-Prozesses des Events.

`nDeviceID`: BACnet ID des BACnet Servers, der das Event gesendet hat.

`tObjectID`: Objekt-ID des Event-generierenden Objekts.

`stDateTime`: Zeitpunkt des Events.

`nNC`: BACnet Instance des zugehörigen BACnet Notification Class Objekts.

`nPriotity`: Event-Priorität.

`eEventType`: Event-Typ.

`eNotifyType`: Meldungs-Typ.

`eFromState`: Objekt Zustand vor dem Event.

`eToState`: Objekt Zustand nach dem Event.

`sMessage`: Meldungstext (*Windows-1252* codiert).

`bAcknReq`: *TRUE* → Meldung muss quittiert werden (Acknowledge).

4.9.54 ST_BACnet_ObjectIdentifierList

```
TYPE ST_BACnet_ObjectIdentifierList :
STRUCT
  nEntries    : UDINT;
  arrObjList  : ARRAY[0..BACnet_MaxObjIdList] OF T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
END_STRUCT
END_TYPE
```

`nEntries`: Anzahl gültiger Einträge in **arrObjList** (Index **0** ... **nEntries** - 1 enthalten gültige Werte).

`arrObjList`: Liste mit BACnet-Objekt-IDs (Objekttyp und Objektinstanz).

4.9.55 ST_BACnet_ObjectTypesSupported

PLC-Abbildung des BACnet-Datentyps BACnetObjectTypesSupported. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Protocol_Object_Types_Supported*.

```
TYPE ST_BACnet_ObjectTypesSupported :
STRUCT
  SuppAnalogInput        : BOOL;
  SuppAnalogOutput       : BOOL;
  SuppAnalogValue        : BOOL;
  SuppBinaryInput        : BOOL;
  SuppBinaryOutput       : BOOL;
  SuppBinaryValue        : BOOL;
  SuppCalendar           : BOOL;
  SuppCommand            : BOOL;
  SuppDevice             : BOOL;
  SuppEventEnrollment    : BOOL;
  SuppFile               : BOOL;
  SuppGroup              : BOOL;
  SuppLoop               : BOOL;
  SuppMultiStateInput    : BOOL;
  SuppMultiStateOutput   : BOOL;
  SuppNotificationClass  : BOOL;
  SuppProgram            : BOOL;
  SuppSchedule           : BOOL;
  SuppAveraging          : BOOL;
  SuppMultiStateValue    : BOOL;
  SuppTrendLog           : BOOL;
  SuppLifeSafetyPoint    : BOOL;
  SuppLifeSafetyZone     : BOOL;
END_STRUCT
END_TYPE
```

`SuppAnalogInput`: FB_BACnet_AnalogInput [▶ 170]

`SuppAnalogOutput`: **FB_BACnet_AnalogOutput**

`SuppAnalogValue`: FB_BACnet_AnalogValue

`SuppBinaryInput`: **FB_BACnet_BinaryInput**

`SuppBinaryOutput`: **FB_BACnet_BinaryOutput**

`SuppBinaryValue`: **FB_BACnet_BinaryValue**

`SuppCalendar`: **FB_BACnet_Calendar**

`SuppCommand`: **FB_BACnet_Command**

`SuppDevice`: **FB_BACnet_Device**

`SuppEventEnrollment`: **FB_BACnet_EventEnrollment**

`SuppFile`: **FB_BACnet_File**

`SuppGroup`: **FB_BACnet_Group**

`SuppLoop`: **FB_BACnet_Loop**

`SuppMultiStateInput`: **FB_BACnet_MultiStateInput**

`SuppMultiStateOutput`: **FB_BACnet_MultiStateOutput**

`SuppNotificationClass`: **FB_BACnet_NotificationClass**

`SuppProgram`: **FB_BACnet_Program**

`SuppSchedule`: **FB_BACnet_Schedule**

`SuppAveraging`: **FB_BACnet_Averaging**

`SuppMultiStateValue`: **FB_BACnet_MultiStateValue**

`SuppTrendLog`: **FB_BACnet_TrendLog**

`SuppLifeSafetyPoint`: **SuppLifeSafetyZone:**

4.9.56 ST_BACnet_ProgramHandshakeRequests

```
TYPE ST_BACnet_ProgramHandshakeRequests :
STRUCT
  bLoad    : BOOL;
  bSetup   : BOOL;
  bRun     : BOOL;
  bHalt    : BOOL;
  bUnload  : BOOL;
END_STRUCT
END_TYPE
```

4.9.57 ST_BACnet_ProgramHandshakeStates

```
TYPE ST_BACnet_ProgramHandshakeStates :
STRUCT
  bIdle       : BOOL;
  bLoading    : BOOL;
  bRunning    : BOOL;
  bWaiting    : BOOL;
  bHalted     : BOOL;
  bUnloading  : BOOL;
END_STRUCT
END_TYPE
```

4.9.58 ST_BACnet_RecipientListDevice

Struktur für den Datenaustausch der Property *Recipient_List* mit Hilfe des Funktionsbausteins FB_BACnet_RecipientListProperty.

```
TYPE ST_BACnet_RecipientListDevice :
STRUCT
  nEntries     : INT;
  arrDestList  : ARRAY[0..BACnet_MaxDestinationArray] OF ST_BACnet_RecipientListDeviceEntry;
END_STRUCT
END_TYPE
```

`nEntries`: Anzahl Einträge, die gelesen wurden bzw. geschrieben werden sollen. 0 bis **nEntries**-1 in **arrDestList** enthalten gültige Werte.

`arrDestList`: Array mit Daten der Property *Recipient_List*. Es stehen 0 bis **nCount**-1 Einträge zur Verfügung.

4.9.59 ST_BACnet_RecipientListDeviceEntry

```
TYPE ST_BACnet_RecipientListDeviceEntry :
STRUCT
  bMonday              : BOOL:=TRUE;
  bTuesday             : BOOL:=TRUE;
  bWednesday           : BOOL:=TRUE;
  bThursday            : BOOL:=TRUE;
  bFriday              : BOOL:=TRUE;
  bSaturday            : BOOL:=TRUE;
  bSunday              : BOOL:=TRUE;
  stFromTime           : ST_BACnet_Time := (nHour       := 0,
                           nMinute     := 0,
                           nSecond     := 0,
                           nHundredths := 0);
  stToTime             : ST_BACnet_Time := (nHour       := 23,
                           nMinute     := 59,
                           nSecond     := 59,
                           nHundredths := 99);
  tDeviceId            : T_BACnet_ObjectIdentifier:=16#FFFFFFFF;
  nProcessId           : UDINT;
  bToOffNormal         : BOOL:=TRUE;
  bToFault             : BOOL:=TRUE;
```

```
  bToNormal             : BOOL:=TRUE;
  bIssueConfirmedNotification  : BOOL;
END_STRUCT
END_TYPE
```

`bMonday`, `bTuesday`, `bWednesday`, `bThursday`, `bFriday`, `bSaturday` und `bSunday`: Flags für die Wochentage: *TRUE* des jeweiligen Wochentags gibt den Empfänger an dem entsprechenden Tag frei; *FALSE* = keine Weiterleitung an den Empfänger.

`stFromTime` und `stToTime`: Bereich von-bis in dem die Weiterleitung von Nachrichten an dem Empfänger freigegeben ist (Z.B. Alarm für Bewegungsmelder oder Fensterkontakte nur Nachts aktiveren).

`tDeviceId`: BACnet-ID des Empfänger BACnet-Servers.

`nProcessId`: BACnet-Process-ID des Empfängers.

`bToOffNormal`: Transition nach *OFF-NORMAL* werden übermittelt.

`bToFault`: Transition nach *FAULT* werden übermittelt.

`bToNormal`: Transitionen nach *NORMAL* werden übermittelt.

`bIssueConfirmedNotification`: Die Nachricht soll als zu-quittieren markiert werden.

4.9.60 ST_BACnet_ServerStatistics

```
TYPE ST_BACnet_ServerStatistics :
STRUCT
  nOpenRequests         : DWORD;
  nRequestRetries       : DWORD;
  nRequestTimeOuts      : DWORD;
  nPropChanges          : DWORD;
  nPersistWrites        : DWORD;
  persistChangeSetId    : DWORD;
  secpersistNextWrite   : DWORD;
  persistLastWrite      : ST_BACnet_DateTime;
END_STRUCT
END_TYPE
```

4.9.61 ST_BACnet_ServicesSupported

PLC-Abbildung des BACnet-Datentyps BACnetServicesSupported. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Protocol_Services_Supported*.

```
TYPE ST_BACnet_ServicesSupported :
STRUCT
  SuppAcknowledgeAlarm              : BOOL;
  SuppConfirmedCOVNotification      : BOOL;
  SuppConfirmedEventNotification    : BOOL;
  SuppGetAlarmSummary               : BOOL;
  SuppGetEnrollmentSummary          : BOOL;
  SuppSubscribeCOV                  : BOOL;
  SuppAtomicReadFile                : BOOL;
  SuppAtomicWriteFile               : BOOL;
  SuppAddListElement                : BOOL;
  SuppRemoveListElement             : BOOL;
  SuppCreateObject                  : BOOL;
  SuppDeleteObject                  : BOOL;
  SuppReadProperty                  : BOOL;
  SuppReadPropertyConditional       : BOOL;
  SuppReadPropertyMultiple          : BOOL;
  SuppWriteProperty                 : BOOL;
  SuppWritePropertyMultiple         : BOOL;
  SuppDeviceCommunicationControl    : BOOL;
  SuppConfirmedPrivateTransfer      : BOOL;
  SuppConfirmedTextMessage          : BOOL;
  SuppReinitializeDevice            : BOOL;
  SuppVtOpen                        : BOOL;
  SuppVtClose                       : BOOL;
  SuppVtData                        : BOOL;
  SuppAuthenticate                  : BOOL;
```

```
  SuppRequestKey                    : BOOL;
  SuppIAm                           : BOOL;
  SuppIHave                         : BOOL;
  SuppUnconfirmedCOVNotification    : BOOL;
  SuppUnconfirmedEventNotification  : BOOL;
  SuppUnconfirmedPrivateTransfer    : BOOL;
  SuppUnconfirmedTextMessage        : BOOL;
  SuppTimeSynchronization           : BOOL;
  SuppWhoHas                        : BOOL;
  SuppWhoIs                         : BOOL;
  SuppReadRange                     : BOOL;
  SuppUtcTimeSynchronization        : BOOL;
  SuppLifeSafetyOperation           : BOOL;
  SuppSubscribeCOVProperty          : BOOL;
  SuppGetEventInformation           : BOOL;
END_STRUCT
END_TYPE
```

4.9.62 ST_BACnet_StatusFlags

PLC-Abbildung des BACnet-Datentyps BACnetStatusFlags. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zur Property *Status_Flags*.

```
TYPE ST_BACnet_StatusFlags :
STRUCT
  in_alarm        : BOOL;
  fault           : BOOL;
  overridden      : BOOL;
  out_of_service  : BOOL;
END_STRUCT
END_TYPE
```

`in_alarm`: *FALSE*, wenn Property *Event_State* = *NORMAL*, sonst *TRUE*.

`fault`: *TRUE*, wenn Property *Reliablility* ≠ *NO_FAULT_DETECTED*, sonst *FALSE*.

`overridden`: *TRUE*, wenn mit Hilfe des Prozessdatum RawIoStateOverride = *TRUE* bzw. RawIoStateOverrideInverted = *FALSE* gemeldet wird, dass die Ausgabe des jeweiligen Objekts überschrieben wird (typischerweise bei Handbedienmodulen).

`out_of_service`: *TRUE*, wenn die Property *Out_Of_Service* = *TRUE*, sonst *FALSE*.

4.9.63 ST_BACnet_TcIoEthStatistic

Struktur mit Informationen zum Senden und Empfangen von Netzwerk Frames aus dem Netzwerktreiber.

```
TYPE ST_BACnet_TcIoEthStatistic :
STRUCT
  dcTimeStamp     : ARRAY[0..1] OF DWORD;
  sendFrames      : DWORD;
  recvFrames      : DWORD;
  sendTimeRTime   : DWORD;
  recvTimeRTime   : DWORD;
  sendTimeNdis    : DWORD;
  recvTimeNdis    : DWORD;
END_STRUCT
END_TYPE
```

`dcTimeStamp`: Echtzeit-Zeitstempel (DC time).

`sendFrames`: Gesendete Ethernet Frames des BACnet-Adapters.

`recvFrames`: Verarbeitete Ethernet Frames des BACnet-Adapters.

`sendTimeRTime`: nicht-BACnet spezifische Information aus dem Netzwerktreiber (interne Diagnose).

`recvTimeRTime`: nicht-BACnet spezifische Information aus dem Netzwerktreiber (interne Diagnose).

`sendTimeNdis`: nicht-BACnet spezifische Information aus dem Netzwerktreiber (interne Diagnose).

`recvTimeNdis`: nicht-BACnet spezifische Information aus dem Netzwerktreiber (interne Diagnose).

4.9.64 ST_BACnet_TcIoEthTxRxErrorCount

Struktur mit Informationen zu fehlerhaften Netzwerk Frames aus dem Netzwerktreiber.

```
TYPE ST_BACnet_TcIoEthTxRxErrorCount :
STRUCT
  txCnt  : DWORD;
  rxCnt  : DWORD;
END_STRUCT
END_TYPE
```

`txCnt`: Anzahl fehlerhaft gesendeter Frames (verlorene Sendepakete).

`rxCnt`: Anzahl fehlerhaft empfangener Frames (verworfene Empfangspakete).

4.9.65 ST_BACnet_Time

PLC-Abbildung des BACnet-Datentyps Time. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetDateTime.

```
TYPE ST_BACnet_Time :
STRUCT
  nHour        : BYTE:=16#FF;
  nMinute      : BYTE:=16#FF;
  nSecond      : BYTE:=16#FF;
  nHundredths  : BYTE:=16#FF;
END_STRUCT
END_TYPE
```

`nHour`**:** Mögliche Werte für Stunde: 0 bis 23; 255 entspricht undefiniert (bzw. jede Stunde)

`nMinute`**:** Mögliche Werte für Minute: 0 bis 59; 255 entspricht undefiniert (bzw. jede Minute)

`nSecond`**:** Mögliche Werte für Sekunde: 0 bis 59; 255 entspricht undefiniert (bzw. jede Sekunde)

`nHundredths`**:** Mögliche Werte für Hundertstelsekunde: 0 bis 99; 255 entspricht undefiniert (bzw. jede Hundertstelsekunde)

4.9.66 ST_BACnet_TimeValue

```
TYPE ST_BACnet_TimeValue :
STRUCT
   time            : BACnetTime;
  reserved         : DWORD;
  value            : BACnetValue;
  BACnetTimeValue  : *PBACnetTimeValue;
  stTime           : ST_BACnet_Time;
  nRes             : DWORD;
  stValue          : ST_BACnet_Value;
END_STRUCT
END_TYPE
```

4.9.67 ST_BACnet_TimeValueBool

PLC-Abbildung des BACnet-Datentyps BACnetTimeValue für Einträge vom Typ Bool oder Null. Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum Datentyp BACnetTimeValue.

```
TYPE ST_BACnet_TimeValueBool :
STRUCT
  bValid  : BOOL;
  stTime  : ST_BACnet_Time;
  eType   : E_BACnetDataTypes;
  bValue  : BOOL;
END_STRUCT
END_TYPE
```

`bValid`: Wert ist gültig.

`eType`: Mögliche Werte sind BACnetDataType_Null oder BACnetDataType_Bool.

`bValue`: Wert *FALSE* oder *TRUE*, wenn **eType** = BACnetDataType_Bool.

4.9.68 ST_BACnet_TimeValueList

```
TYPE ST_BACnet_TimeValueList :
STRUCT
  nEntrySize           : DWORD;
  reserved             : DWORD;
  listEntry            : BACnetTimeValue;
  BACnetTimeValueList  : *PBACnetTimeValueList;
  nEntrySize           : DWORD;
  nRes                 : DWORD;
  stListEntry          : ST_BACnet_TimeValue;
END_STRUCT
END_TYPE
```

4.9.69 ST_BACnet_UnConfirmedServiceDiag

Struktur mit Informationen zu unbestätigten BACnet Diensten (unconfirmed).

```
TYPE ST_BACnet_UnConfirmedServiceDiag :
STRUCT
  nReqSend      : DWORD;
  nReqSendFail  : DWORD;
  nReqRecv      : DWORD;
  nReqRecvFail  : DWORD;
END_STRUCT
END_TYPE
```

`nReqSend`: Anzahl an gesendeten Anfragen (unconfirmed requests).

`nReqSendFail`: Anzahl an fehlgeschlagenen Anfragen.

`nReqRecv`: Anzahl an empfangenen Anfragen (unconfirmed requests).

`nReqRecvFail`: Anzahl an Anfragen die nicht verarbeitet werden konnten (kein freier Speicher, fehlerhafte Codierung etc.).

4.9.70 ST_BACnet_Value

```
TYPE ST_BACnet_Value :
STRUCT
  choice                : BACnetValueChoice;
  length                : DWORD;
  boolValue             : BOOL;
  realValue             : FLOAT;
  doubleValue           : DOUBLE;
  signedValue           : INT;
  unsignedValue         : UINT;
  pOctedStringValue     : BYTE*;
  pCharStringValue      : CHAR*;
  objectIdentifierValue : BACnetObjectIdentifier;
  enumerationValue      : EnumerationValueType;
  BACnetValue           : *PBACnetValue;
  eValueChoice          : E_BACnetDataTypes;
  nRes1                 : WORD;
  cbData                : DWORD;
  nData                 : BYTE;
END_STRUCT
END_TYPE
```

4.9.71 ST_BACnet_WeeklyScheduleBool

Struktur für den Datenaustausch der Property *Weekly_Schedule* mit Hilfe des Funktionsbausteins FB_BACnet_WeeklyScheduleProperty.

```
TYPE ST_BACnet_WeeklyScheduleBool :
STRUCT
  bReqGet     : BOOL;
  bReqSet     : BOOL;
  nGet        : USINT;
  nSet        : USINT;
  bBusy       : BOOL;
  arrEntries  : ARRAY[0..6] OF ARRAY[0..BACnet_MaxDayEntry] OF ST_BACnet_TimeValueBool;
END_STRUCT
END_TYPE
```

bReqGet: Anforderung Property-Daten lesen *FALSE* → *TRUE* an den zugehörigen Property-Funktionsbaustein (siehe FB_BACnet_WeeklyScheduleProperty). Der Flankenwechsel *TRUE* → *FALSE* signalisiert die Übernahme der Anforderung vom zugehörigen Property-Funktionsbaustein. Aktuelle Daten stehen demnächst zur Verfügung. Das erfolgreiche Laden der Property-Daten wird mittels **nGet** signalisiert.

bReqSet: Anforderung Property-Daten schreiben *FALSE* → *TRUE* an den zugehörigen Property-Funktionsbaustein (siehe FB_BACnet_WeeklyScheduleProperty). Der Flankenwechsel *TRUE* → *FALSE* signalisiert die Übernahme der Anforderung vom zugehörigen Property-Funktionsbaustein. Aktuelle Daten stehen demnächst zur Verfügung. Das erfolgreiche Schreiben der Property-Daten wird mittels **nSet** signalisiert.

nGet: Zähler wird um 1 erhöht, wenn Property-Daten erfolgreich gelesen wurden. Der Zähler ist umlaufend.

nSet: Zähler wird um 1 erhöht, wenn Property-Daten erfolgreich geschrieben wurden. Der Zähler ist umlaufend.

bBusy: Zugehöriger Property-Funktionsbaustein lädt bzw. schreibt Daten aus bzw. in die Property. Die Daten sind ungültig, während **bBusy** = *TRUE* gesetzt ist.

arrEntries: Array mit Daten der Property *Weekly_Schedule*. Eintrag 0 → Montag ... 6 → Sonntag; Pro Wochentag stehen 0 bis BACnet_MaxDayEntry Einträge zur Verfügung. Die Einträge werden als BACnetTimeValue codiert und beschränken sich auf den Datentyp Bool bzw. Null. Einträge mit anderen Datentypen werden ignoriert.

5 SPS Bibliothek: TcBACnet.Lib

Im Folgenden wird der Funktionsumfang der SPS Bibliothek "TcBACnet.lib" beschrieben. Die Bibliothek bietet die Möglichkeit auf Objekte einer BACnet-Konfiguration komfortabel aus einem SPS-Programm zuzugreifen.

Der Einsatz der Bibliothek ist **nicht** zwingend.

Der Zugriff auf die Daten (Properties) der BACnet-Objekte kann auf 2 Arten erfolgen:

1. Die Funktionsbausteine bzw. selbst deklarierten Variablen werden mit Hilfe des "synchronen" Prozessdatenmappings verknüpft (Verknüpfung der SPS mit der Hardware-Konfiguration im System Manager). Dafür enthalten die Funktionsbausteine der SPS die Zuweisung als Eingangs- bzw. Ausgangsdaten ("bValue AT%I* : BOOL" bzw. "bValue AT%Q* : BOOL"). Der Vorteil liegt im zyklischen Update der Daten. Sämtliche Werte werden synchron zum SPS-Zyklus geschrieben bzw. gelesen.
2. Objekt-Daten werden asynchron mit Hilfe von ADS gelesen bzw. geschrieben. Dabei werden die zu schreibenden/lesenden Daten mit Hilfe von ADS-Bausteinen an die entsprechende Adresse (NetId, Port, Index-Group u. -Offset --> Objekt-Type, -Instance und Property-ID) gesendet (FB_BACnet_WriteProp [▶ 504]) bzw. von dieser gelesen (FB_BACnet_ReadProp [▶ 501]). Der Vorteil dieser Methode besteht in der Möglichkeit Daten nur dann zu Schreiben, wenn tatsächlich eine Änderung vorliegt. Bei häufigen Änderungen kann es jedoch zu Verzögerungen kommen, die nicht vorhersehbar sind. Die maximale Verzögerung liegt bei 5 Sekunden ("tBACnet_ADSTimeOut : TIME := t#5s"). Nach Ablauf der Zeit meldet der Baustein einen Fehlercode zurück.

Die Bibliothek (https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749057547.zip) kann hier https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749057547.zipheruntergeladen werden.

Installation: Der Inhalt des ZIP-Archives muss in den TwinCAT Ordner (default: "C:\TwinCAT") unter "\Plc\Lib" kopiert werden.

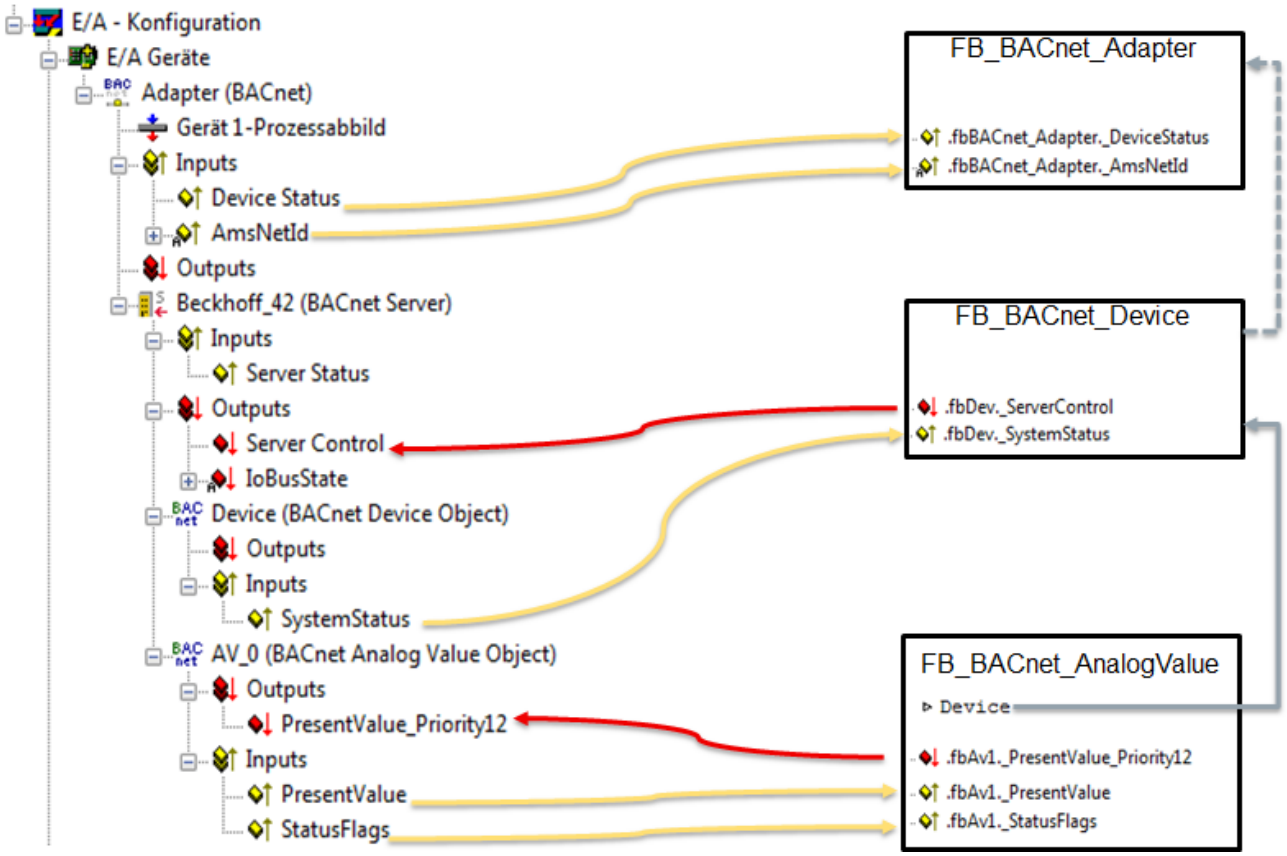
Übersicht

Die SPS-Bibliothek TcBACnet-Lib ist eine Sammlung von Funktionsbausteinen zur Programmierung eines BACnet-Controllers. Für jedes BACnet-Objekt stehen Funktionsbausteine zur Verfügung, über die ausgewählte Properties gelesen und geschrieben werden. Generell wird zwischen Funktionsbausteinen für Server- und Client-Objekte unterschieden. Funktionsbausteine für Client-Objekte werden mit dem Präfix "Remote" bezeichnet, da sie den Zugriff auf entfernte BACnet-Objekte auf anderen Geräten ermöglichen. Die BACnet-Objekt-Funktionsbausteine realisieren eine Auswertung, ob Prozessdaten gültig sind bzw. der BACnet-Controller betriebsbereit ist (bReady).

Der Funktionsbaustein FB_BACnet_Adapter [▶ 378] repräsentiert ein BACnet-Device und damit den Zugangspunkt zum BACnet-Netzwerk über Netzwerkkarte. Über diesen Baustein kann u.a. erkannt werden, ob ein Link vorhanden ist. Um die Übersicht im SPS-Programm zu erhöhen, wird der BACnet-Adapter als globale Variable innerhalb der TcBACnet-Lib angelegt, da in der Mehrzahl der Projekte genau ein BACnet-Adapter verwendet wird.

Je nach Client- bzw. Server-Funktionalität, werden mit den Funktionsbausteinen FB_BACnet_Device [▶ 419] bzw. FB_BACnet_RemoteDevice [▶ 465] Zustands- und Steuerinformationen des BACnet-Client bzw. Server verknüpft. Die Betriebsbereitschaft der BACnet-Objekte wird über den Client- bzw. Server-Status sowie der Property *System_Status* des jeweiligen Device bzw. Remote Device-Objekts ermittelt. Jedem BACnet-Objekt-Funktionsbaustein muss deshalb eine Referenz auf die Instanz des zugehörigen FB_BACnet_Device [▶ 419] bzw. FB_BACnet-RemoteDevice [▶ 465] übergeben werden, um die Statusauswertung (**bReady**) zu ermöglichen.

Die nachfolgende Abbildung zeigt eine Übersicht der Funktionsbausteine der TcBACnet-Lib und den entsprechenden Verknüpfungen mit den BACnet-Modulen. Die Verknüpfung zwischen einem SPS-Programm auf Basis von TcBACnet-Lib und den BACnet-Modulen einer Konfiguration über die Funktion SPS-Automapping [▶ 64] automatisiert erstellt werden.



Bausteinbezeichnung

Die Bausteinbeschreibungen beziehen sich auf die Bausteinversionen mit maximalen Prozessdaten (Erweiterung "_EX"). Die Standardbausteine enthalten einen minimal nötigen Satz an Prozessdaten, um Performanceverluste bei großen Konfigurationen zu vermeiden.Zudem gibt es Funktionsbausteinvarianten für Objekte mit kommandierbarer Property Present_Value die den Schreibzugriff mit Hilfe von ADS realisieren (Erweiterung "_ADS"). Diese Bausteinvariante schreibt den Wert der Property Present_Value mit entsprechender Priorität nur bei Änderungen in das Objekt via ADS (WOC). So kann auf Objekte des lokalen Server mehrfach geschrieben werden (z.B. Schreiben gleicher Priorität).Funktionsbausteinvarianten mit der Erweiterung "_RAW" bieten den Zugriff auf den Rohwert des entsprechenden Objekts. Bei Verwendung dieser Bausteinvariante wird die Property Present_Value nicht auf die Klemmenhardware sondern im PLC Programm abgebildet. So lassen sich BACnet-Werte z.B. von Subbussystemen umsetzen. Ein Beispiel befindet sich in der Bausteinbeschreibung.

5.1 Beispiel: NotificationClass und NotificationSink

Im Folgenden wird ein Beispiel zur Verwendung der *NotificationClass* in Zusammenspiel mit einer *NotificationSink* gegeben. Diese "sammelt" die Events von 3 Objekten (BV, AV, MV) und sendet diese an den lokalen Prozess 10 (*NotificationSink*) des lokalen BACnet-Device (Event-Notification mittels Intrinsic-Reporting).

Zudem soll das *PresentValue* des *AnalogValue*-Objekts bei Wertänderung von 0.1 an die *NotificationSink* gesendet werden (COV-Notification).

Das Beispiel https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749047307.zip kann hier https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749047307.zipheruntergeladen werden. Das enthaltene TSM File kann auch ohne Laden der SPS zum Testen verwendet werden.

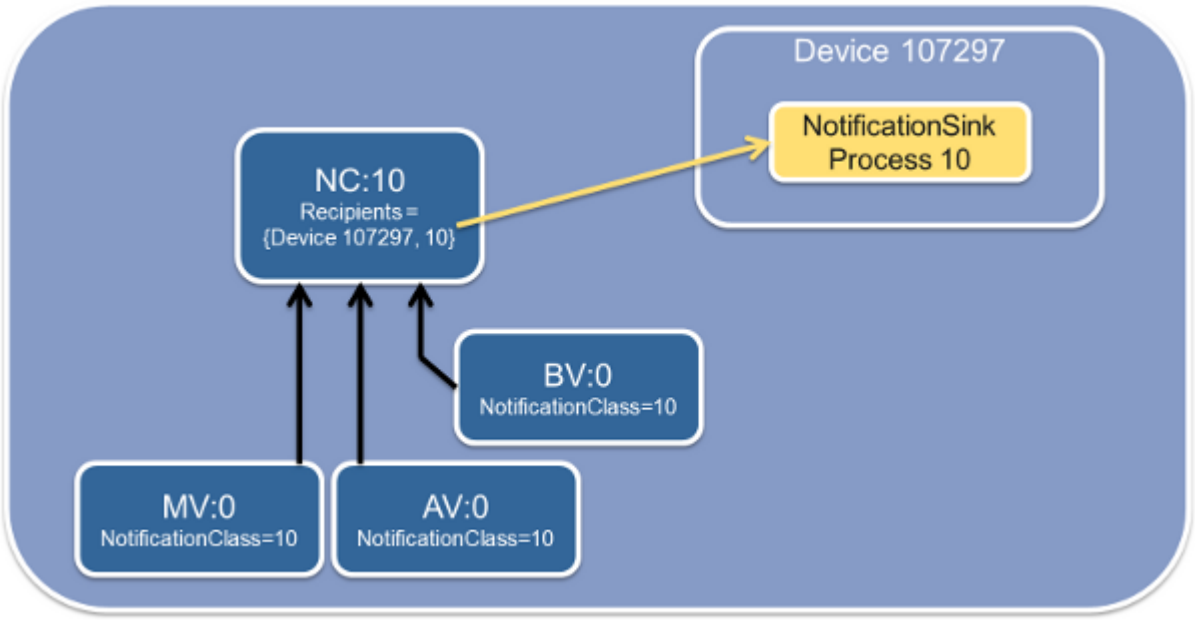


Abb. 80: Bild-1: Schematische Darstellung zur Beispielkonfiguration der NotificationClass, deren Event-generierenden Objekte (AV, MV, BV) und der NotificationSink.

Die Konfiguration der Objekte erfolgt durch die Deklaration im SPS Programm:

```
VAR
 viewDummy : BOOL; (* ~( BACnet_StructuredViewPath : \/DemoNClass : ) *)

 fbDevice : FB_BACnet_Device;

 fbBV_0 : FB_BACnet_BinaryValue;
  (* ~(BACnet_ObjectIdentifier : 0 : nolink )
  (BACnet_EventEnable : {to_offnormal;to_fault;to_normal} : nolink )
  (BACnet_ActiveText : EIN : nolink )
  (BACnet_InactiveText : AUS : nolink )
  (BACnet_NotificationClass : 10 : nolink ) *)

 fbAV_0 : FB_BACnet_AnalogValue;
  (* ~(BACnet_ObjectIdentifier : 0 : nolink )
  (BACnet_EventEnable : {to_offnormal;to_fault;to_normal} : nolink )
  (BACnet_NotificationClass : 10 : nolink ) *)

 fbMV_0 : FB_BACnet_MultiStateValue;
  (* ~(BACnet_ObjectIdentifier : 0 : nolink )
  (BACnet_EventEnable : {to_offnormal;to_fault;to_normal} : nolink )
  (BACnet_NotificationClass : 10 : nolink )
  (BACnet_NumberOfStates : 5 : nolink )
  (BACnet_StateText : ErrLow;WarnLow;Normal;WarnHigh;ErrHigh : nolink )
  (BACnet_FaultValues : 1;5 : nolink )
  (BACnet_AlarmValues : 2;4 : nolink )
  (BACnet_Description : MV DEMO Objekt. : nolink ) *)

 fbNC_10 : FB_BACnet_NotificationClass;
  (* ~(BACnet_ObjectIdentifier : 10 : nolink )
  (BACnet_Priority : 70;80;90 : nolink )
  (BACnet_AckRequired : {to_offnormal;to_fault;to_normal} : nolink )
  (BACnet_RecipientList :
      <ArrayOfBACnetDestination>
    <BACnetDestination>
      <recipient>
        <choice>device</choice>
        <BACnetObjectIdentifier>
          <objId>33661729</objId>
        </BACnetObjectIdentifier>
      </recipient>
      <validDays>65025</validDays>
      <fromTime>
        <hour>0</hour>
        <minute>0</minute>
        <second>0</second>
        <hundredths>0</hundredths>
      </fromTime>
      <toTime>
```

```
        <hour>23</hour>
        <minute>59</minute>
        <second>59</second>
        <hundredths>99</hundredths>
      </toTime>
      <processIdentifier>10</processIdentifier>
      <transitions>57349</transitions>
      <issueConfirmedNotifications>false</issueConfirmedNotifications>
    </BACnetDestination>
      </ArrayOfBACnetDestination>
  : nolink ) *)END_VAR
```

Die zum Empfang der Events eingefügte wird im System Manager konfiguriert: *NotificationSink*

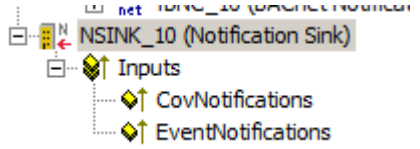


Abb. 81: Bild-2: NotificationSink in der System Manager Baumansicht

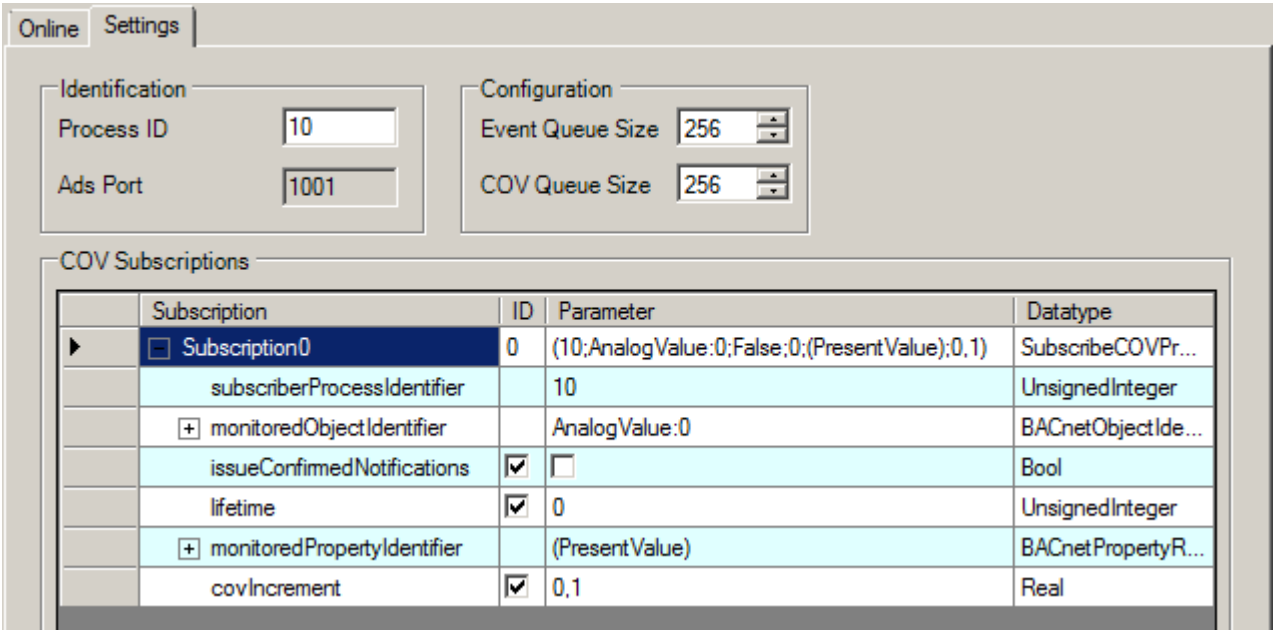


Abb. 82: Bild-3: NotificationSink Konfiguration im System Manager

Process ID: Die Prozessidentifikation wird auf 10 eingestellt und der *NotificationClass* als Empfangs-Prozess mitgeteilt (via SPS-Automapping Kommentar).

COV Subscriptions: Als Beispiel wird das *PresentValue* des *AnalogValue*-Objekts (AV:0) abonniert. Der Parameter *covIncrement* gibt die nötige Wertänderung an bei deren Erreichen eine COV-Notification versendet wird.

Folgende Einträge werden im Online-Reiter der angezeigt, nachdem die Konfiguration geladen und die 's der jeweiligen Objekte verändert wurden: *NotificationSink PresentValue*

Notifications

Nr	Time	Source	Param
COVNotification_0	12.06.2012 14:20:53	(10;Device:107297;AnalogValue:0;0;{(Pres...	COV...
COVNotification_1	12.06.2012 14:21:19	(10;Device:107297;AnalogValue:0;0;{(Pres...	COV...
COVNotification_2	12.06.2012 14:21:24	(10;Device:107297;AnalogValue:0;0;{(Pres...	COV...
COVNotification_3	12.06.2012 14:21:27	(10;Device:107297;AnalogValue:0;0;{(Pres...	COV...
COVNotification_4	12.06.2012 14:21:30	(10;Device:107297;AnalogValue:0;0;{(Pres...	COV...
EventNotification_0	12.06.2012 14:20:53	(10;Device:107297;MultiStateValue:0;12.0...	Eve...
EventNotification_1	12.06.2012 14:20:53	(10;Device:107297;BinaryValue:0;12.06.20...	Eve...
EventNotification_2	12.06.2012 14:21:11	(10;Device:107297;BinaryValue:0;12.06.20...	Eve...
EventNotification_3	12.06.2012 14:21:13	(10;Device:107297;BinaryValue:0;12.06.20...	Eve...
EventNotification_4	12.06.2012 14:21:57	(10;Device:107297;MultiStateValue:0;12.0...	Eve...
EventNotification_5	12.06.2012 14:22:03	(10;Device:107297;MultiStateValue:0;12.0...	Eve...
EventNotification_6	12.06.2012 14:22:05	(10;Device:107297;MultiStateValue:0;12.0...	Eve...
EventNotification_7	12.06.2012 14:22:08	(10;Device:107297;MultiStateValue:0;12.0...	Eve...
processIdentifier		10	Unsi...
initiatingDeviceId...		Device:107297	BAC...
eventObjectIdent...		MultiStateValue:0	BAC...
timeStamp		12.06.2012 14:22:08	d...
notificationClass		10	Unsi...
priority		80	Unsi...
eventType		change_of_state	BAC...
messageText	☑	MV DEMO Objekt.	Char...
notifyType		alarm	BAC...
ackRequired	☑	☑	Bool
fromState	☑	offnormal	BAC...
toState		fault	BAC...
eventValues	☑	((49156;11;5))	c...

Abb. 83: Bild-4: NotificationSink Online-Daten nachdem COV- und Event-Notifications empfangen wurden.

Der Text unter messageText wird aus der Property Description des Event-generierenden Objekts gebildet.

5.2 FB_BACnet_Adapter

Der folgende Funktionsbaustein wird für die Anbindung des SPS-Programmes an einen lokalen BACnet-Adapter (Netzwerkkarte) verwendet. Die Verknüpfung des Funktionsbausteins erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell oder mittels PLC-Automapping [▶ 64] automatisch verknüpft werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.

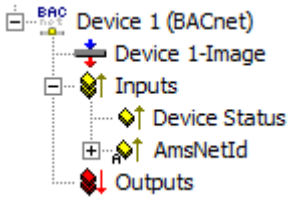


Abb. 84: Bild-1: Prozessdaten des BACnet Adapters im System Manager.

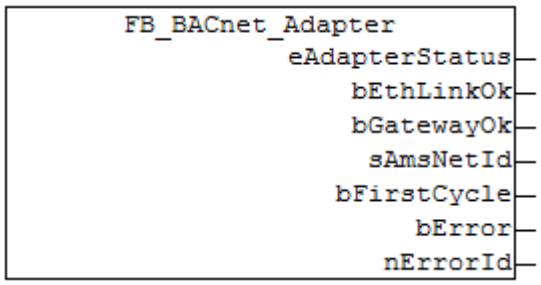


Abb. 85: Bild-2: Funktionsbaustein des BACnet Adapters im SPS-Programm.

Verwendung

Mit Hilfe des Funktionsbausteins FB_BACnet_Adapter wird der Zustand des lokalen BACnet Adapters gelesen (Prozessdatum "Device Status") und im SPS-Programm ausgegeben. Zudem wird mit Hilfe des Prozessdatums "AmsNetId" die AMS-NetID des BACnet-Adapters gelesen (ARRAY OF BYTE --> String) und als Zeichenkette ausgegeben.

Eine Instanz des Bausteins FB_BACnet_Adapter wird bereits durch die SPS-Bibliothek als globale Instanz bereitgestellt und muss daher *nicht* explizit angelegt werden. Die globale Instanz wird bei Verwendung des SPS-Automappings erkannt und automatisch mit dem BACnet-Adapter im System Manager verknüpft. Die Bausteininstanz wird von den Bausteinen FB_BACnet_Device und FB_BACnet_RemoteDevice benötigt. Im Folgenden ist die Hierarchie zwischen FB_BACnet_Adapter, FB_BACnet_Device bzw. FB_BACnet_RemoteDevice und einem Objekt vom Typ AnalogValue dargestellt:

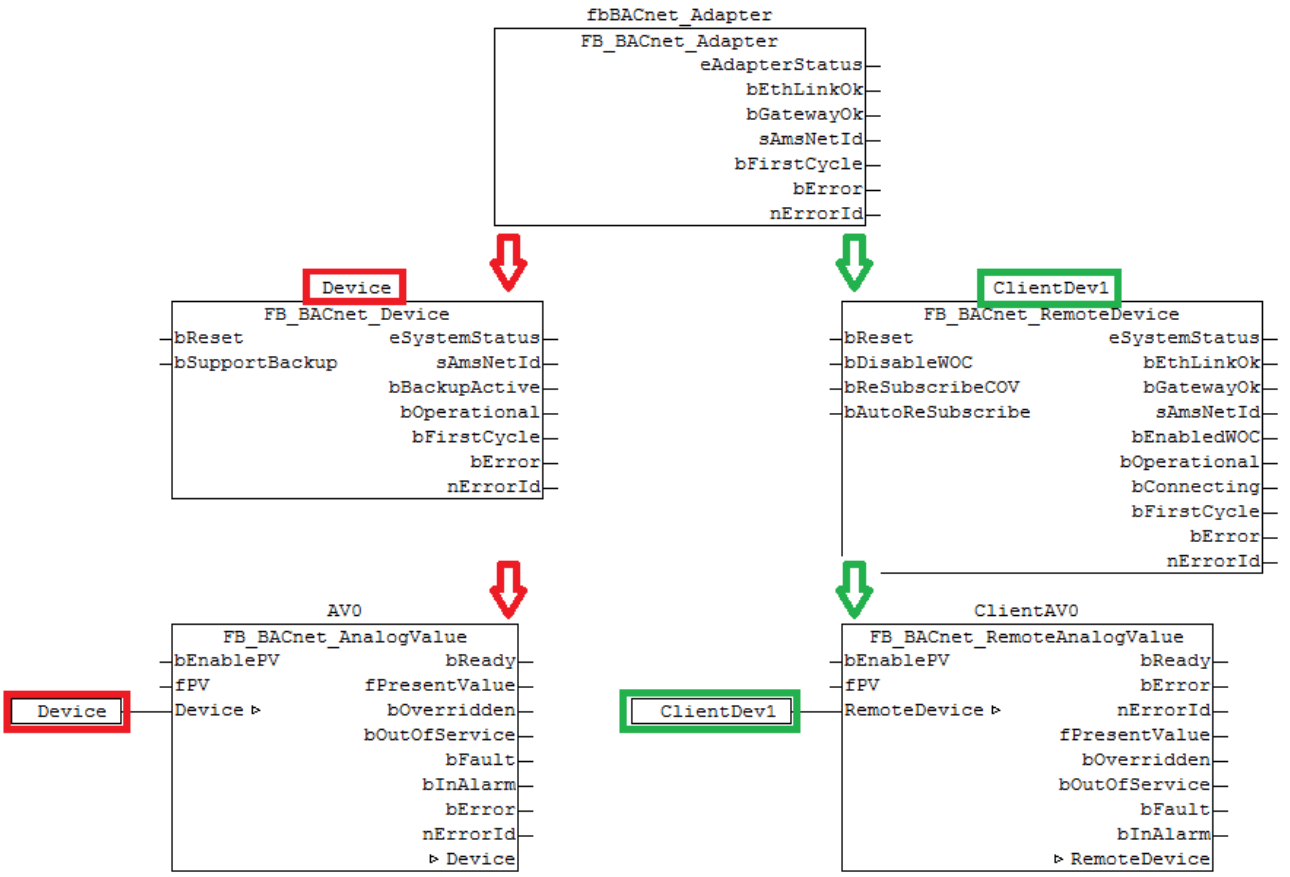


Abb. 86: Bild-3: Beispiel zur Verknüpfung der FB-Instanzen in der SPS.

Für Informationen zur Verwendung der Bausteine FB_BACnet_Device und -_RemoteDevice siehe FB_BACnet_Device bzw. FB_BACnet_RemoteDevice. Die Instanz des Bausteins FB_BACnet_Adapter muss jedoch nicht zwingend aufgerufen werden. Der Aufruf des Adapter-Bausteins wird bei Bedarf automatisch von den Instanzen der Bausteine FB_BACnet_Device bzw. FB_BACnet_RemoteDevice vorgenommen.

VAR_OUTPUT

```
eAdapterStatus    : E_BACnetAdapterStatus;
bEthLinkOk        : BOOL;
bGatewayOk        : BOOL;
sAmsNetId         : T_AmsNetId;
bFirstCycle       : BOOL;
bError            : BOOL;
nErrorId          : UINT;
```

eAdapterStatus: Aktueller Status des BACnet-Adapters. Folgende Werte sind möglich:
0: BACnetAdapterStatus_Init (Initialisierung hat begonnen)
1: BACnetAdapterStatus_CheckIpAddr (Prüfung der IP-Adresse)
2: BACnetAdapterStatus_CheckParam (Prüfung der Parameter)
3: BACnetAdapterStatus_GetGatewayMAC (MAC des Gateways ermitteln)
4: BACnetAdapterStatus_WaitGatewayMAC (Auf MAC Adresse des Gateways warten)
5: BACnetAdapterStatus_Complete (Fertig initialisiert und bereit)

bEthLinkOk: Die lokale Ethernet-Verbindung ist aktiv (Kabel gesteckt, Adapter verbunden), wenn der Ausgang auf *TRUE* gesetzt ist.

bGatewayOk: Das konfigurierte Gateway ist erreichbar, wenn der Ausgang auf *TRUE* gesetzt ist.

sAmsNetId: Ausgabe der AMS-NetID des lokalen BACnet-Adapters (kann für den asynchronen Zugriff via ADS auf BACnet-Objekte verwendet werden).

bFirstCycle: Wird mit dem ersten Aufruf der Bausteininstanz nach SPS-Reset bzw. -Neustart für einen Zyklus gesetzt.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer (0 = kein Fehler; 1 = keine gültige AMS-NetID). Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Adapter*.**nERR_xxx**).

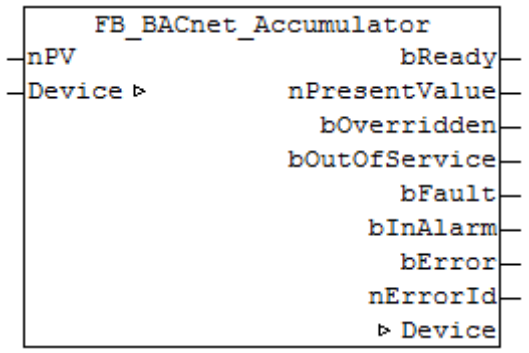
Bei fehlender Netzwerkverbindung (**bEthLinkOk** = *FALSE*) bzw. nicht erreichbarem Gateway (**bGatewayOk** = *FALSE*) werden alle verbundenen Remote-Objekte (Clients) gesperrt. Lokale Objekte sind davon nicht betroffen.

5.3 BACnet Server Objects

5.3.1 FB_BACnet_Accumulator

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_Accumulator" kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Accumulator* (ACC) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

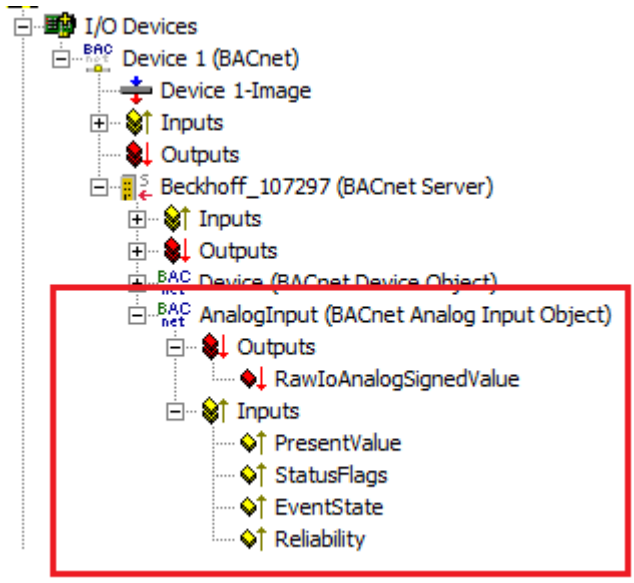


Abb. 87: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
nPV          : UDINT;
```

nPV: Wert der in die Property *Present_Value* geschrieben werden soll. Das Schreiben der Prozessdaten erfolgt immer zyklisch.

VAR_OUTPUT

```
bReady           : BOOL;
nPresentValue    : UDINT;
nMaxPV           : UDINT;  (*siehe Beschreibung*)
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device [▶ 419]" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Present_Value*).

bOverriden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device [▶ 419])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

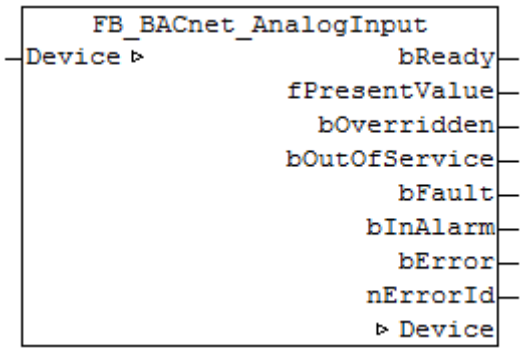
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.2 FB_BACnet_AnalogInput

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_AnalogInput" kann lesend auf ein BACnet-Objekt vom Typ AnalogInput (AI) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

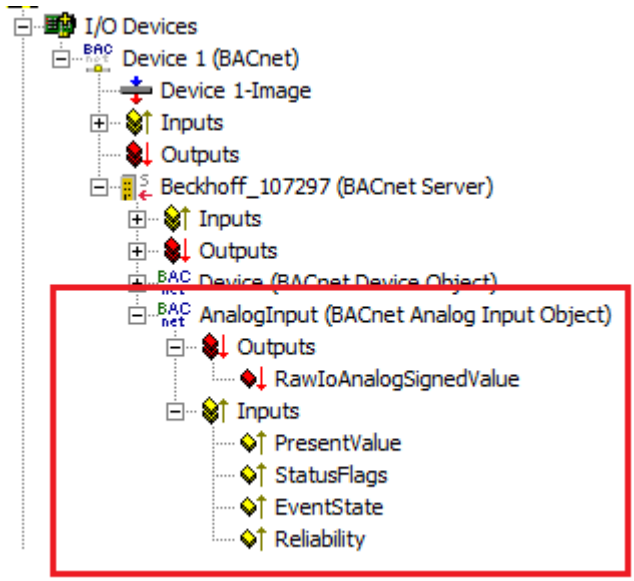


Abb. 88: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_OUTPUT

```
bReady                  : BOOL;
fPresentValue           : REAL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
```

```
bInAlarm                : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Present_Value*).

bOverriden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

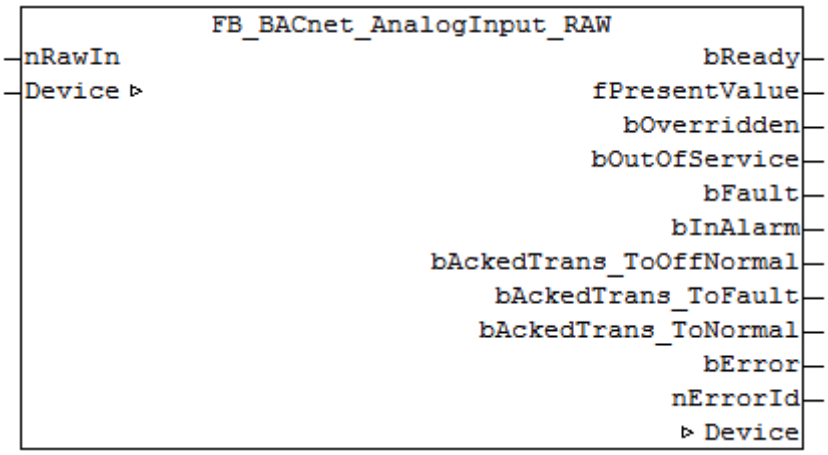
Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

5.3.3 FB_BACnet_AnalogInput_RAW

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.

Im Gegensatz zu der Standardvariante des Bausteins wird der Rohwert durch einen Funktionsbaustein-Eingang bereitgestellt und nicht durch die Klemmen-Hardware. So können z.B. Analogeingangsinformationen die im SPS-Code generiert werden als AnalogInput-Objekt nach BACnet abgebildet werden (Signalumsetzung von Subbussystemen oder virtuellen Datenpunkten etc.).



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_AnalogInput_RAW" kann lesend auf ein BACnet-Objekt vom Typ AnalogInput (AI) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

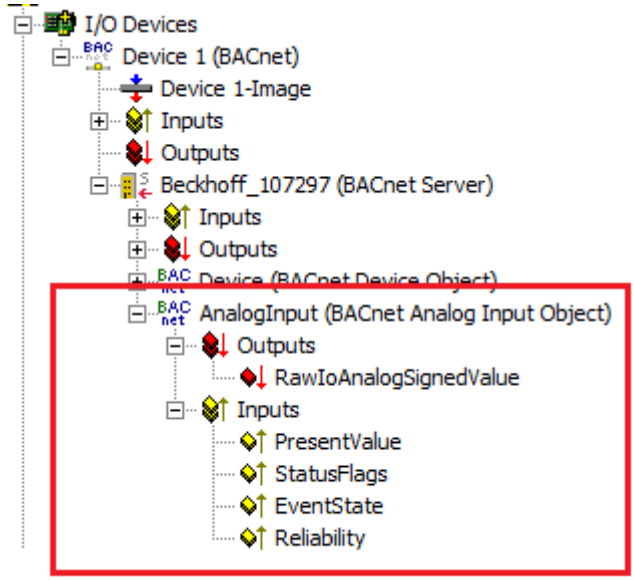


Abb. 89: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
nRawIn       : INT; (* ~(BACnet_RawIoAnalogSignedValue : : ) *)
```

nRawIn: Rohwerteingang des Objekts im Wertebereich -32768...32767. Der Eingang wird mit dem Prozessdatum "RawIoAnalogSignedValue" des BACnet-Objekts verknüpft. Der Wert aus "nRawIn" wird mit der Property *Resolution* zum dem Wert der Property *Present_Value* verrechnet (vorausgesetzt der Objektzustand ist nicht *out_of_service*).

VAR_OUTPUT

```
bReady                  : BOOL;
fPresentValue           : REAL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Present_Value*).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Status_Flags*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Acked_Transitions*).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung eines ganzzahligen Werts aus EIB in die Property *Present_Value* eines AnalogInput-Objekts gezeigt:

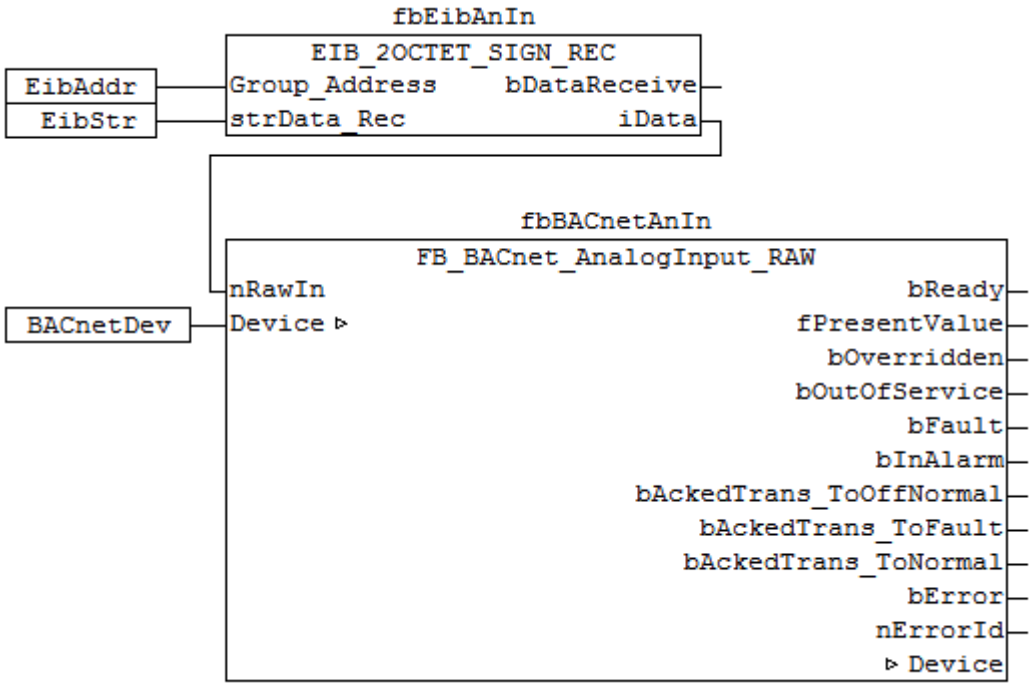
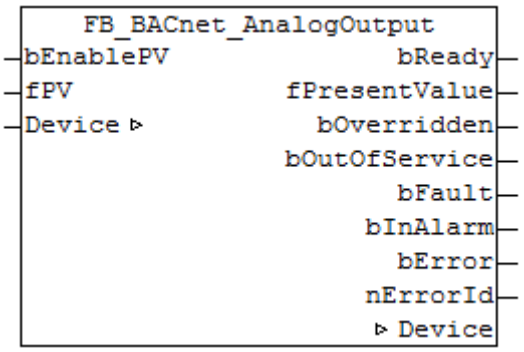


Abb. 90: Bild-2: Beispiel für die Umsetzung eines ganzzahligen Werts aus EIB in die Property Present_Value im PLC Programm. Dieses Beispiel setzt die Bibliothek "TcKL6301.lib" voraus. Siehe Dokumentation zur Klemme KL6301 für weitere Informationen zu EIB.

5.3.4 FB_BACnet_AnalogOutput

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_AnalogOutput" kann lesend und schreibend auf ein BACnet-Objekt vom Typ *AnalogOutput* (AO) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

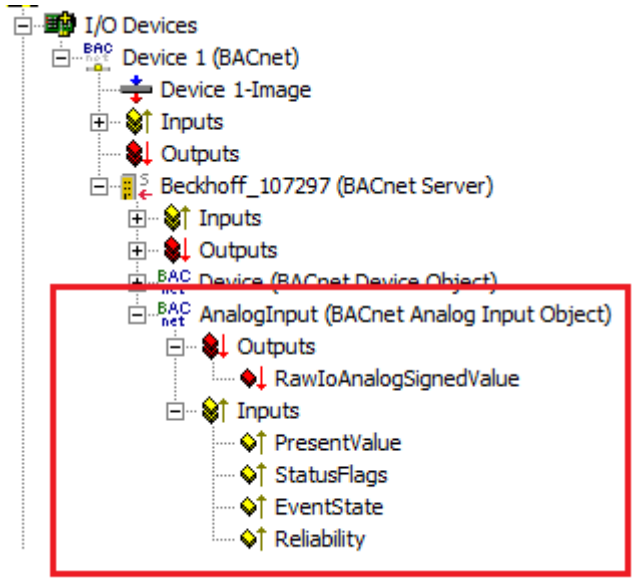


Abb. 91: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV     : BOOL;
fPV           : REAL;
```

bEnablePV: Gibt den Wert des Eingangs "fPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs "fPV". Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).

fPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und 3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben der Prozessdaten erfolgt nach Freigabe immer zyklisch.

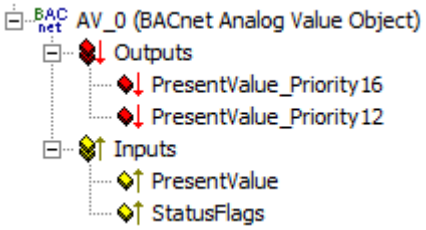


Abb. 92: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

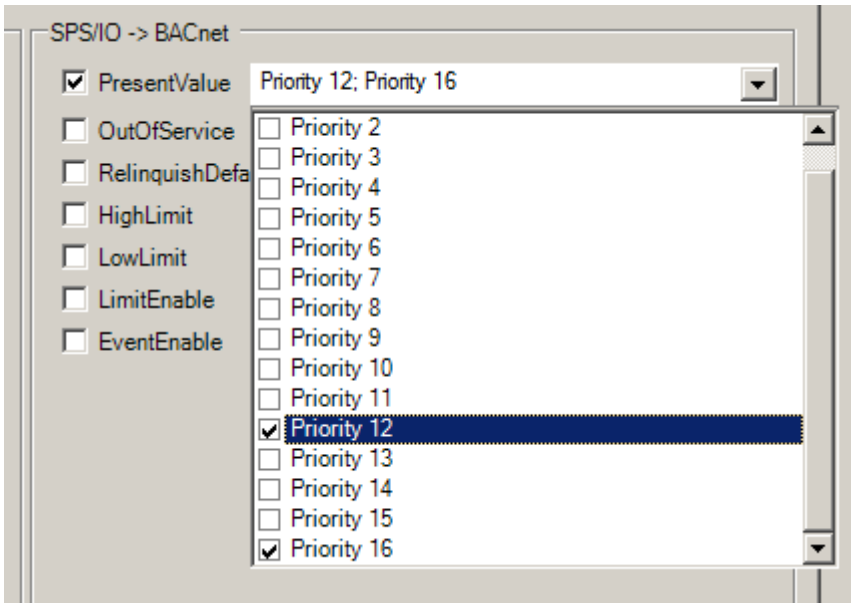


Abb. 93: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 und 16 werden als mappbare Prozessdaten angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady                  : BOOL;
fPresentValue           : REAL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

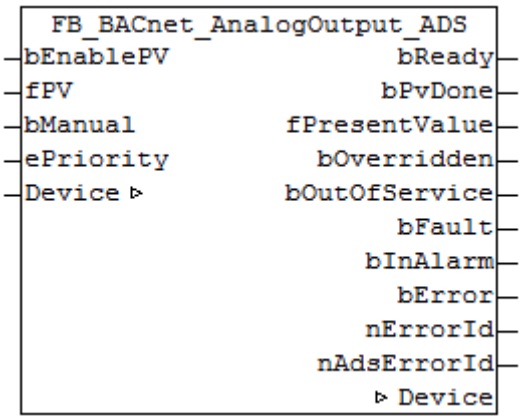
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

5.3.5 FB_BACnet_AnalogOutput_ADS

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_AnalogOutput_ADS" kann lesend und asynchron schreibend (per ADS) auf ein BACnet-Objekt vom Typ *AnalogOutput* (AO) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

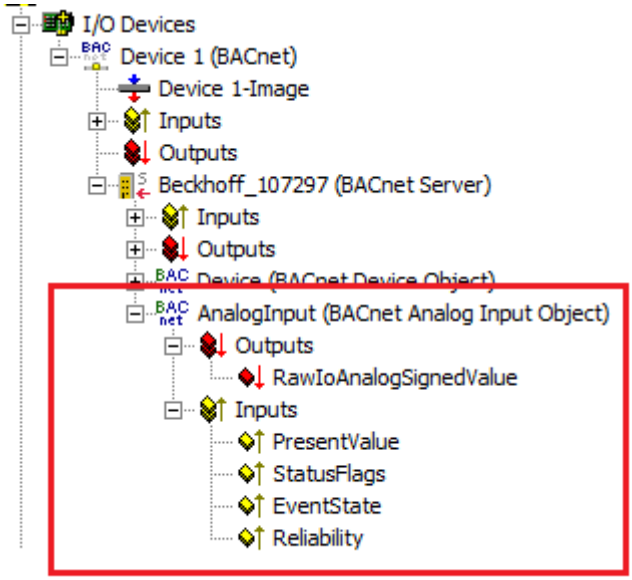


Abb. 94: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV           : BOOL;
fPV                 : REAL; (* decimal number for priority x, when input bEnablePV is TRUE *)
bManual             : BOOL; (* FALSE --> TRUE: write present_value immediately,
                FALSE: only write present_value on-change,
                TRUE: write present_value on-change and when device turns operational *)
ePriority           : E_BACNETPRIORITY := BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs "fPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus "fPV" auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist).
Wird der Eingang auf *FALSE* gesetzt erfolgt ein Schreibzugriff mit dem Wert *NULL* auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (NULL Schreiben).

fPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn "bEnablePV" auf *TRUE* gesetzt ist. Die Priorität wird mit dem Eingang "ePriority" bestimmt (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben auf das Objekt wird bei Wertänderung via ADS ausgeführt.

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung geschieht, ist es potenziell möglich dass das Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:

- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE* --> *TRUE*: löst das Schreiben der Property *Present_Value* unmittelbar aus.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 419]).

ePriority: Vorgabe der Priorität der Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 515]).

VAR_OUTPUT

```
bReady              : BOOL;
bPvDone             : BOOL; (* present_value written over ADS *)
fPresentValue       : REAL;
bOverridden         : BOOL;
bOutOfService       : BOOL;
bFault              : BOOL;
bInAlarm            : BOOL;
bError              : BOOL;
nErrorId            : UINT;
nAdsErrorId         : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
4 = der Objekttyp des BACnet Objekts passt nicht zum Funktionsbaustein (Der Objekttyp wird über die

Prozessdaten ausgelesen --> Verknüpfung im System Manager prüfen!)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

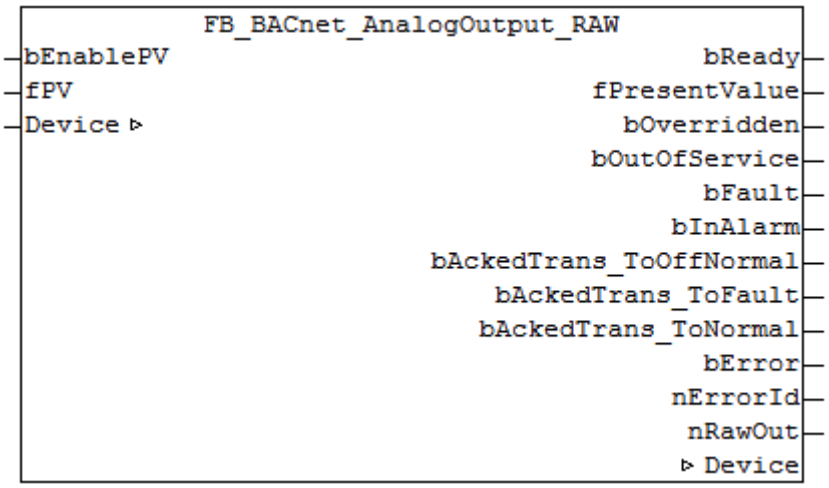
Device: Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

5.3.6 FB_BACnet_AnalogOutput_RAW

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.

Im Gegensatz zu der Standardvariante des Bausteins wird der Rohwert durch einen Funktionsbaustein-Ausgang im SPS-Programm bereitgestellt und nicht auf die Klemmen-Hardware abgebildet. So können z.B. Analogausgangsinformationen im SPS-Code weiterverarbeitet werden (Signalumsetzung in Subbussysteme oder virtueller Ausgangsdatenpunkt etc.).



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_AnalogOutput_RAW" kann lesend und schreibend auf ein BACnet-Objekt vom Typ *AnalogOutput* (AO) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

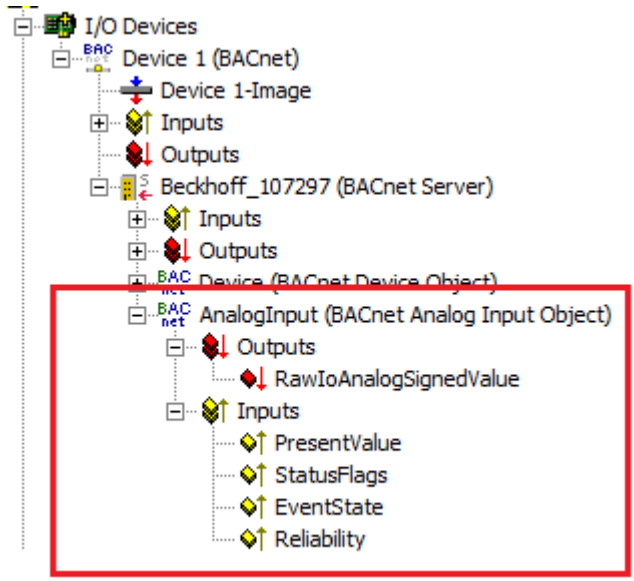


Abb. 95: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV    : BOOL;
fPV          : REAL;
```

bEnablePV: Gibt den Wert des Eingangs "fPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs "fPV". Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).

fPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und 3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben der Prozessdaten erfolgt nach Freigabe immer zyklisch.

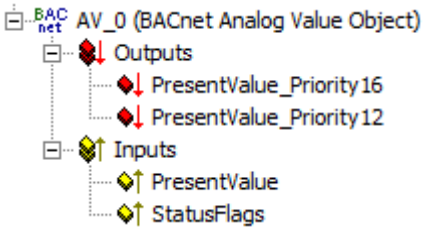


Abb. 96: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

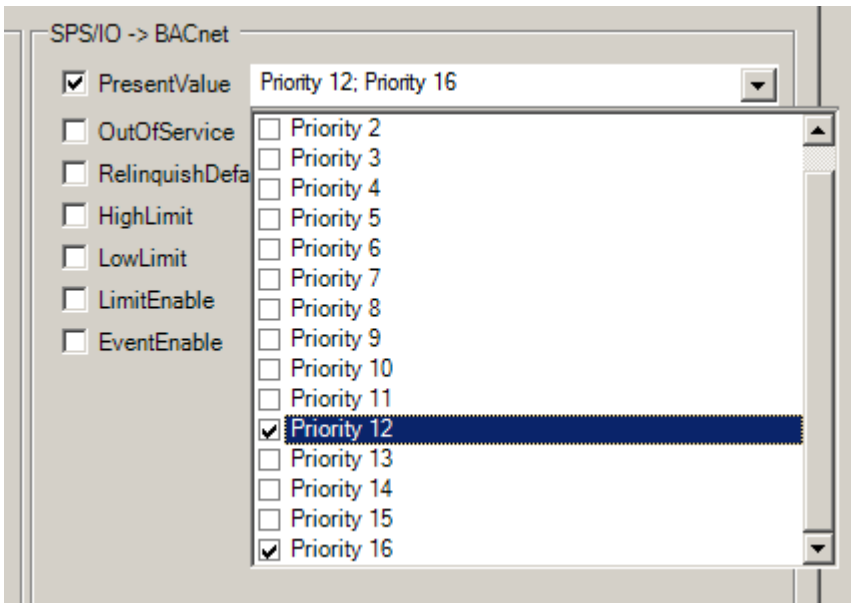


Abb. 97: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 und 16 werden als mappbare Prozessdaten angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady                  : BOOL;
fPresentValue           : REAL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
nRawOut                 : INT; (* ~(BACnet_RawIoAnalogSignedValue : : ) *)
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Status_Flags*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Acked_Transitions*).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

nRawOut: Rohwertausgang des Objekts im Wertebereich -32768...32767. Der Ausgang wird mit dem Prozessdatum "RawIoAnalogSignedValue" des BACnet-Objekts verknüpft. Der Wert der Property *Present_Value* wird mit der Property *Resolution* zu "nRawOut" verrechnet (vorausgesetzt der Objektzustand ist nicht *out_of_service*).

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung der Property *Present_Value* eines AnalogOutput-Objekts nach EIB gezeigt:

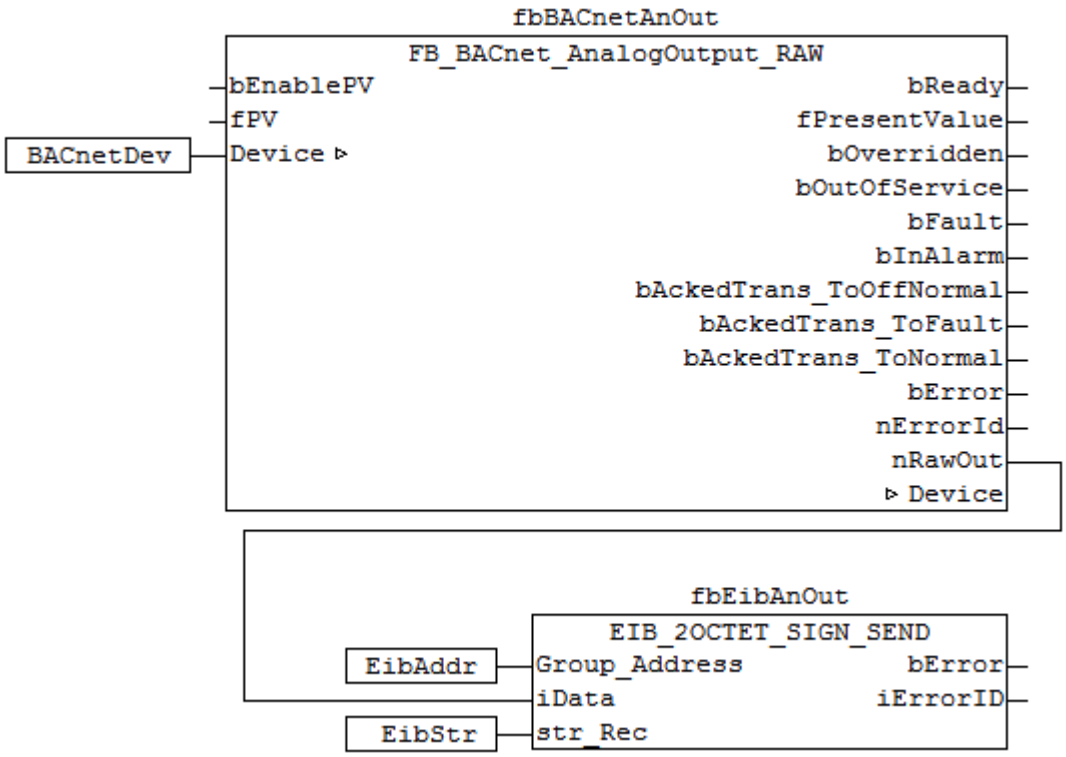
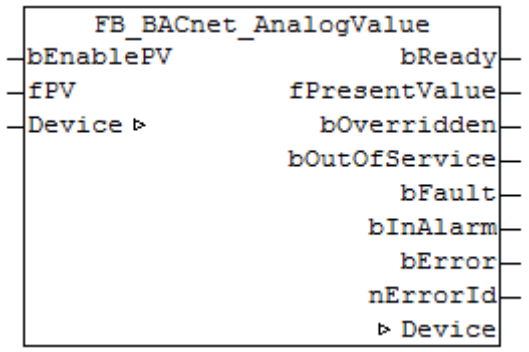


Abb. 98: Bild-4: Beispiel für die Umsetzung der Property Present_Value nach EIB im PLC Programm. Dieses Beispiel setzt die Bibliothek "TcKL6301.lib" voraus. Siehe Dokumentation zur Klemme KL6301 für weitere Informationen zu EIB.

5.3.7 FB_BACnet_AnalogValue

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_AnalogValue" kann lesend und schreibend auf ein BACnet-Objekt vom Typ *AnalogValue* (AV) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

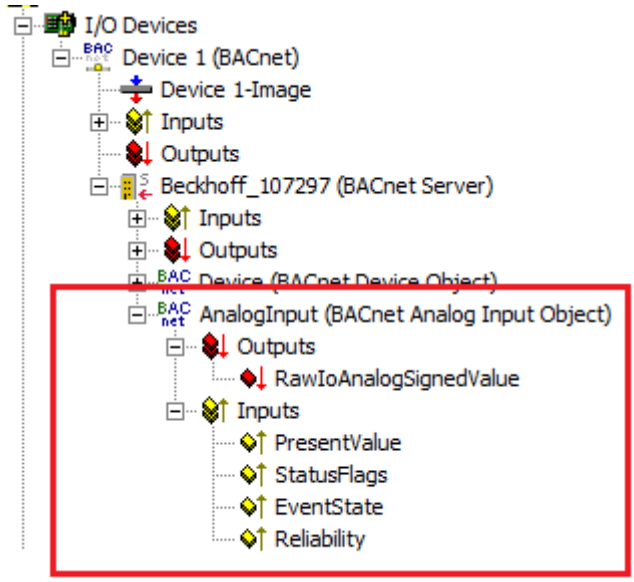


Abb. 99: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV     : BOOL;
fPV           : REAL;
```

bEnablePV: Gibt den Wert des Eingangs "fPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs "fPV". Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).
Wird bEnablePV auf *FALSE* gesetzt, dann wird der zugehörige Eintrag aus dem *Priority_Array* wieder entfernt (NULL Schreiben).

fPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und 3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs **bEnablePV** freigegeben. Das Schreiben der Prozessdaten erfolgt nach Freigabe immer zyklisch.

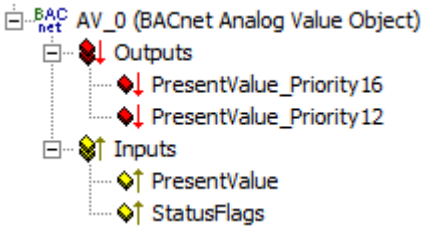


Abb. 100: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

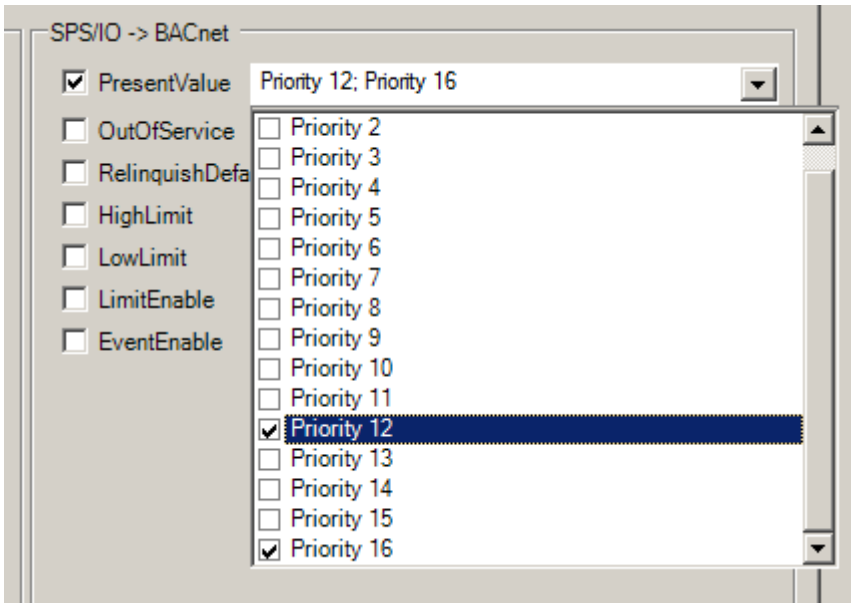


Abb. 101: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 und 16 werden als mappbare Prozessdaten angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady                  : BOOL;
fPresentValue           : REAL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogValue* und Property *Present_Value*).

bOverriden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogValue* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

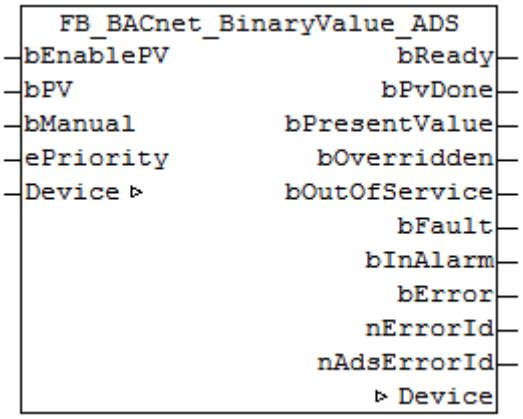
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

5.3.8 FB_BACnet_BinaryValue_ADS

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_BinaryValue_ADS" kann lesend und asynchron schreibend (per ADS) auf ein BACnet-Objekt vom Typ *BinaryValue* (BV) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

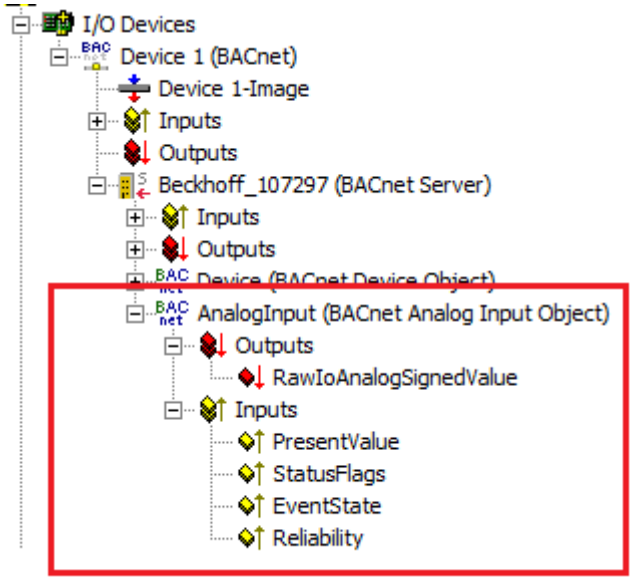


Abb. 102: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV           : BOOL;
bPV                 : BOOL; (* present value bool type *)
bManual             : BOOL; (* FALSE --> TRUE: write present_value immediately,
                FALSE: only write present_value on-change,
                TRUE: write present_value on-change and when device turns operational *)
ePriority           : E_BACNETPRIORITY := BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs "bPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus "bPV" auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist).
Wird der Eingang auf *FALSE* gesetzt erfolgt ein Schreibzugriff mit dem Wert *NULL* auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (NULL Schreiben).

bPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn "bEnablePV" auf *TRUE* gesetzt ist. Die Priorität wird mit dem Eingang "ePriority" bestimmt (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben auf das Objekt wird bei Wertänderung via ADS ausgeführt.

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung geschieht, ist es potenziell möglich dass das Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:

- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE* --> *TRUE*: löst das Schreiben der Property *Present_Value* unmittelbar aus.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 419]).

ePriority: Vorgabe der Priorität der Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 515]).

VAR_OUTPUT

```
bReady              : BOOL;
bPvDone             : BOOL; (* present_value written over ADS *)
bPresentValue       : BOOL;
bOverridden         : BOOL;
bOutOfService       : BOOL;
bFault              : BOOL;
bInAlarm            : BOOL;
bError              : BOOL;
nErrorId            : UINT;
nAdsErrorId         : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
4 = der Objekttyp des BACnet Objekts passt nicht zum Funktionsbaustein (Der Objekttyp wird über die

Prozessdaten ausgelesen --> Verknüpfung im System Manager prüfen!)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

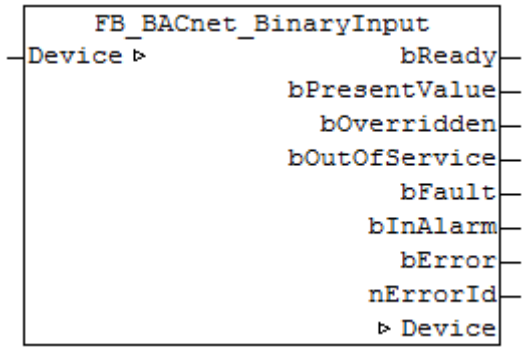
```
Device       : FB_BACnet_Device;
```

Device: Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.9 FB_BACnet_BinaryInput

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_BinaryInput" kann lesend auf ein BACnet-Objekt vom Typ *BinaryInput* (BI) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

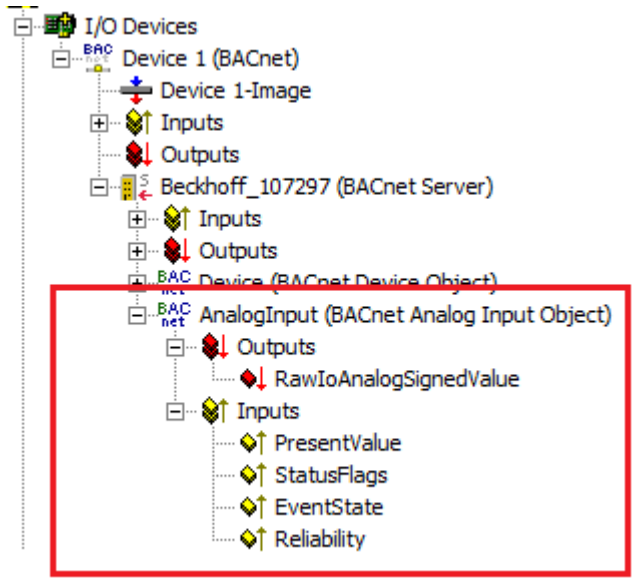


Abb. 103: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_OUTPUT

```
bReady                  : BOOL;
bPresentValue           : BOOL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

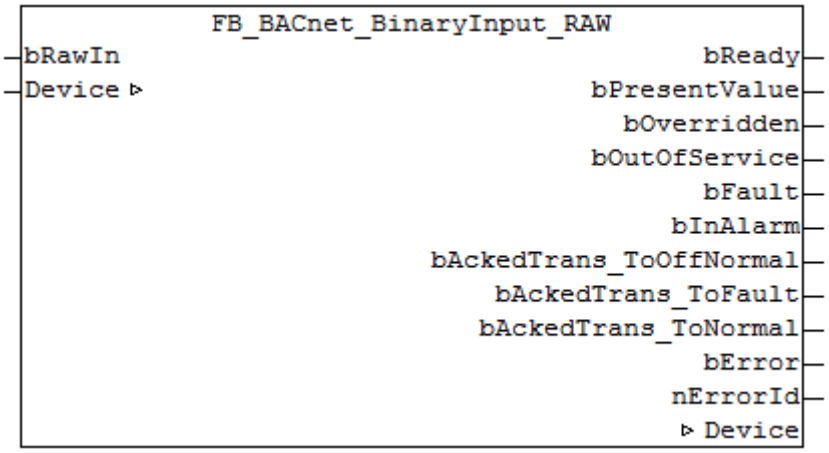
Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

5.3.10 FB_BACnet_BinaryInput_RAW

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.

Im Gegensatz zu der Standardvariante des Bausteins wird der Rohwert durch einen Funktionsbaustein-Eingang bereitgestellt und nicht durch die Klemmen-Hardware. So können z.B. Digitaleingangsinformationen die im SPS-Code generiert werden als BinaryInput-Objekt nach BACnet abgebildet werden (Signalumsetzung von Subbussystemen oder virtuellen Datenpunkten etc.).



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_BinaryInput_RAW" kann lesend auf ein BACnet-Objekt vom Typ *BinaryInput* (BI) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

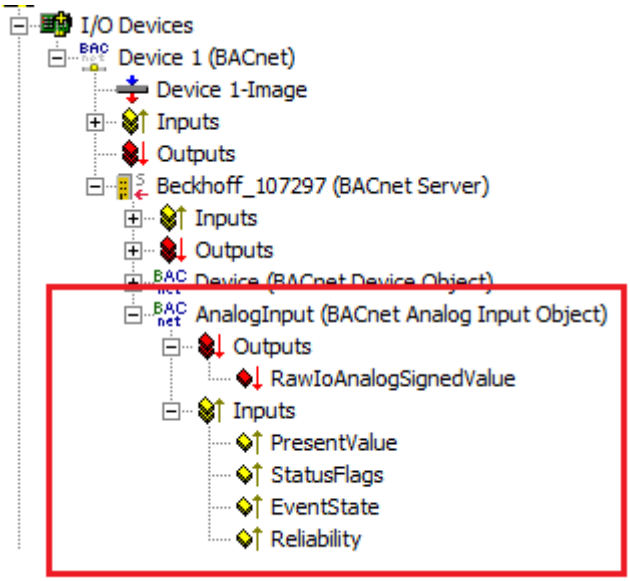


Abb. 104: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bRawIn       : BOOL; (* ~(BACnet_RawIoBinaryBoolValue : : ) *)
```

bRawIn: Rohwerteingang des Objekts. Der Eingang wird mit dem Prozessdatum "RawIoBinaryBoolValue" des BACnet-Objekts verknüpft. Der Wert aus "bRawIn" wird auf die Property *Present_Value* abgebildet (vorausgesetzt der Objektzustand ist nicht *out_of_service*).

VAR_OUTPUT

```
bReady                  : BOOL;
bPresentValue           : BOOL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Status_Flags*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Acked_Transitions*).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung eines Bit-Werts aus EIB in die Property *Present_Value* eines BinaryInput-Objekts gezeigt:

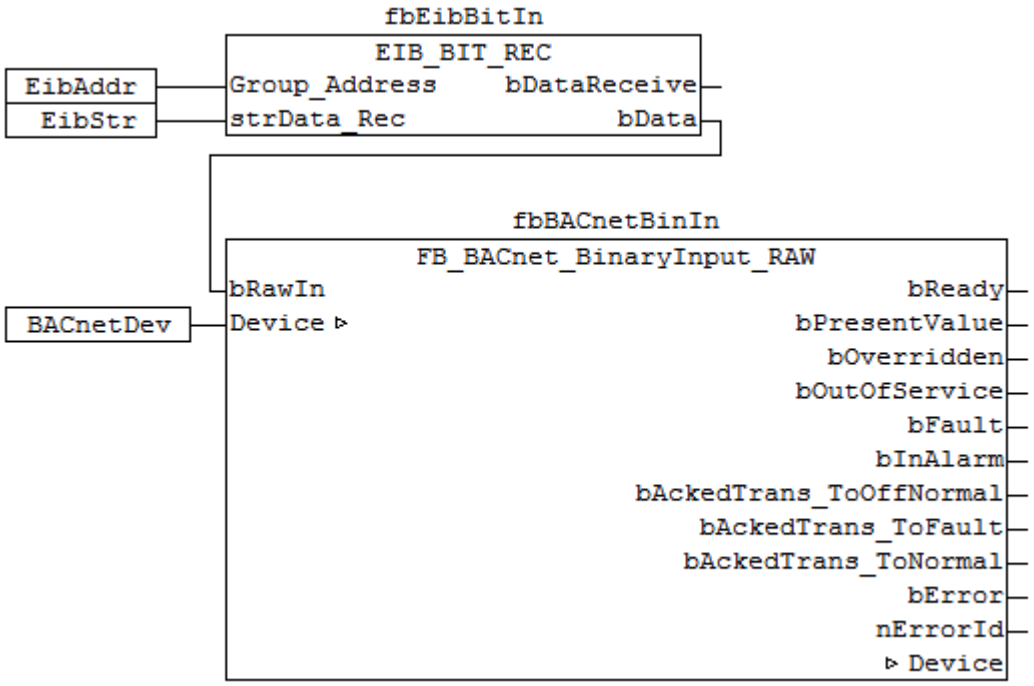


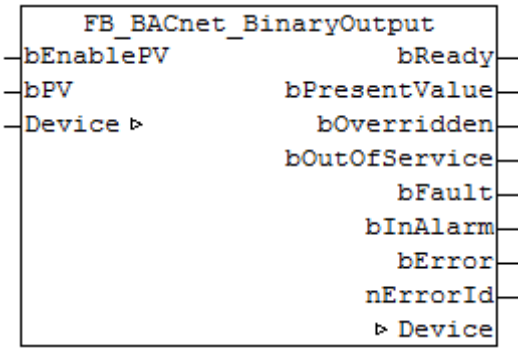
Abb. 105: Bild-2: Beispiel für die Umsetzung eines Bit-Werts aus EIB in die Property Present_Value im PLC Programm.

Dieses Beispiel setzt die Bibliothek "TcKL6301.lib" voraus. Siehe Dokumentation zur Klemme KL6301 für weitere Informationen zu EIB.

5.3.11 FB_BACnet_BinaryOutput

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_BinaryOutput" kann lesend und schreibend auf ein BACnet-Objekt vom Typ *BinaryOutput* (BO) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

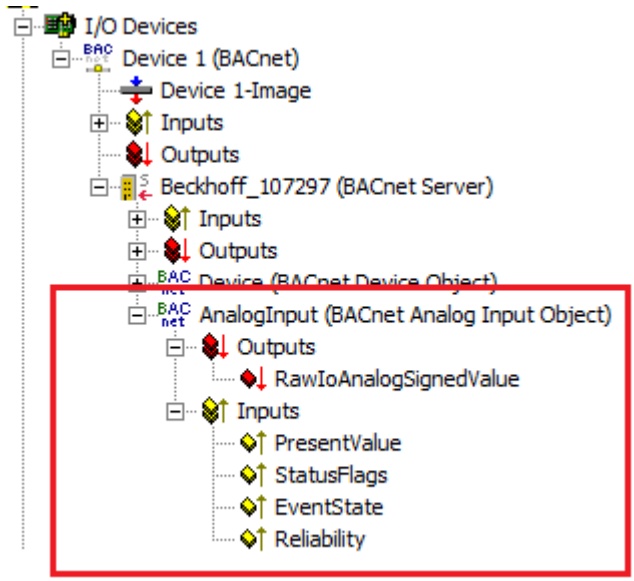


Abb. 106: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV    : BOOL;
bPV          : BOOL;
```

bEnablePV: Gibt den Wert des Eingangs "bPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs "bPV". Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).
Wird bEnablePV auf *FALSE* gesetzt, dann wird der zugehörige Eintrag aus dem *Priority_Array* wieder entfernt (NULL Schreiben).

bPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und 3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben der Prozessdaten erfolgt nach Freigabe immer zyklisch.

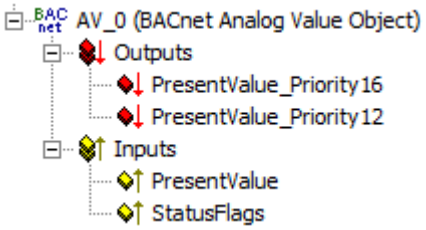


Abb. 107: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

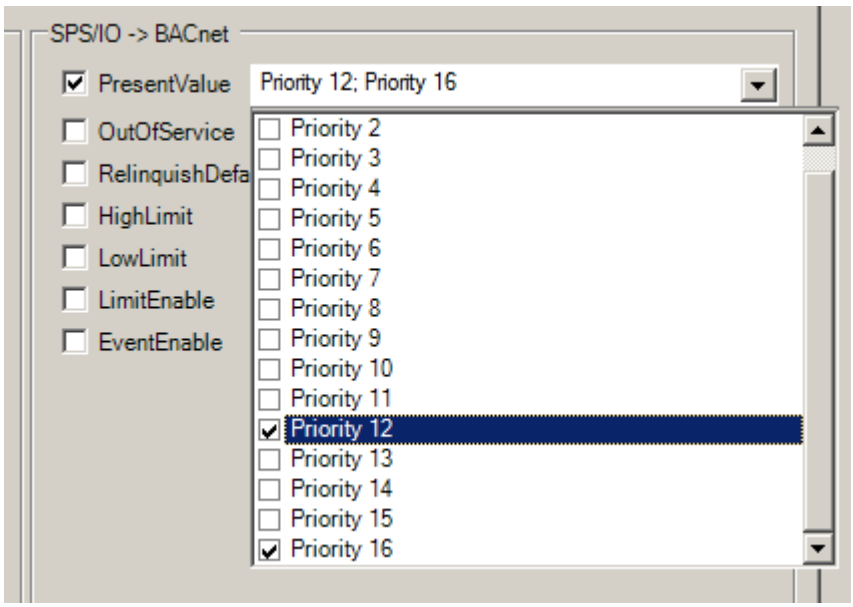


Abb. 108: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 und 16 werden als mappbare Prozessdaten angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady                  : BOOL;
bPresentValue           : BOOL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

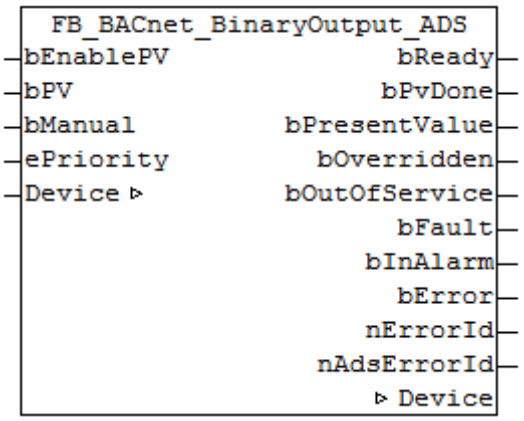
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

5.3.12 FB_BACnet_BinaryOutput_ADS

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_BinaryOutput_ADS" kann lesend und asynchron schreibend (per ADS) auf ein BACnet-Objekt vom Typ *BinaryOutput* (BO) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

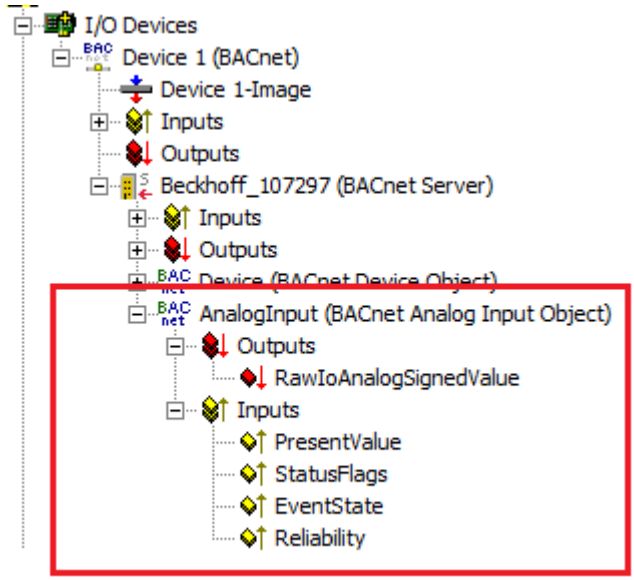


Abb. 109: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV           : BOOL;
bPV                 : BOOL; (* present value bool type *)
bManual             : BOOL; (* FALSE --> TRUE: write present_value immediately,
                FALSE: only write present_value on-change,
                TRUE: write present_value on-change and when device turns operational *)
ePriority           : E_BACNETPRIORITY := BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs "bPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus "bPV" auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist).
Wird der Eingang auf *FALSE* gesetzt erfolgt ein Schreibzugriff mit dem Wert *NULL* auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (NULL Schreiben).

bPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn "bEnablePV" auf *TRUE* gesetzt ist. Die Priorität wird mit dem Eingang "ePriority" bestimmt (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben auf das Objekt wird bei Wertänderung via ADS ausgeführt.

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung geschieht, ist es potenziell möglich dass das Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:

- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE* --> *TRUE*: löst das Schreiben der Property *Present_Value* unmittelbar aus.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 419]).

ePriority: Vorgabe der Priorität der Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 515]).

VAR_OUTPUT

```
bReady           : BOOL;
bPvDone          : BOOL; (* present_value written over ADS *)
bPresentValue    : BOOL;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bError           : BOOL;
nErrorId         : UINT;
nAdsErrorId      : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
4 = der Objekttyp des BACnet Objekts passt nicht zum Funktionsbaustein (Der Objekttyp wird über die Prozessdaten ausgelesen --> Verknüpfung im System Manager prüfen!)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

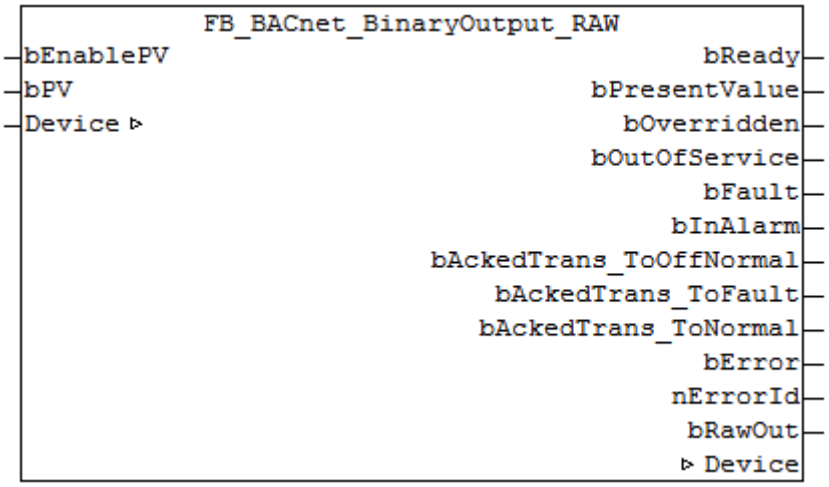
Device: Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.13 FB_BACnet_BinaryOutput_RAW

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.

Im Gegensatz zu der Standardvariante des Bausteins wird der Rohwert durch einen Funktionsbaustein-Ausgang im SPS-Programm bereitgestellt und nicht auf die Klemmen-Hardware abgebildet. So können z.B. Digitalausgangsinformationen im SPS-Code weiterverarbeitet werden (Signalumsetzung in Subbussysteme oder virtueller Ausgangsdatenpunkt etc.).



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_BinaryOutput_RAW" kann lesend und schreibend auf ein BACnet-Objekt vom Typ *BinaryOutput* (BO) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

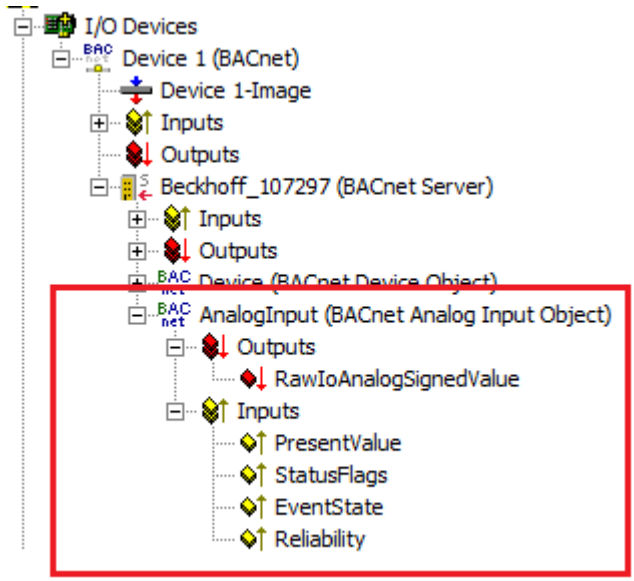


Abb. 110: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV    : BOOL;
bPV          : BOOL;
```

bEnablePV: Gibt den Wert des Eingangs "bPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs "bPV". Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).
Wird bEnablePV auf *FALSE* gesetzt, dann wird der zugehörige Eintrag aus dem *Priority_Array* wieder entfernt (NULL Schreiben).

bPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und 3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben der Prozessdaten erfolgt nach Freigabe immer zyklisch.

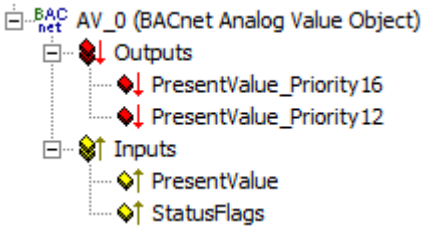


Abb. 111: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

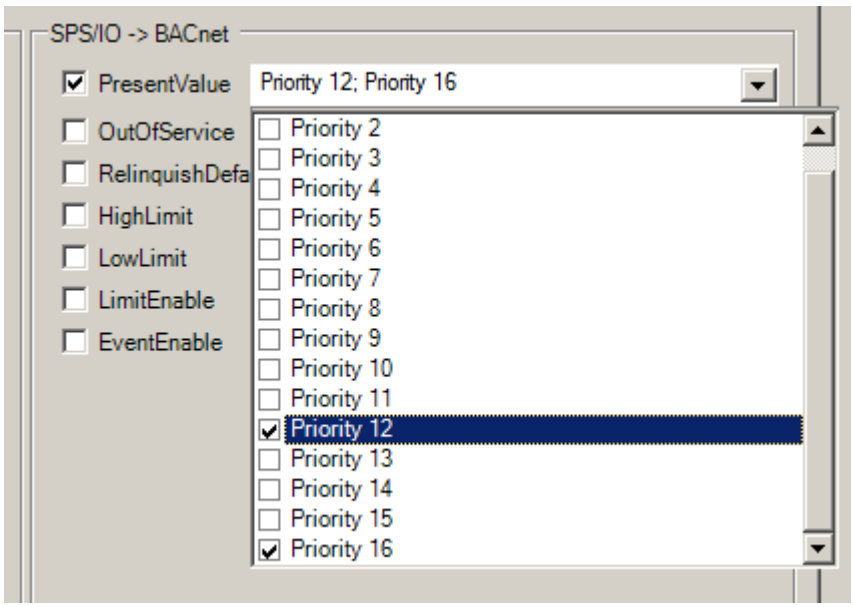


Abb. 112: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 und 16 werden als mappbare Prozessdaten angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady                  : BOOL;
bPresentValue           : BOOL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bAckedTrans_ToOffNormal : BOOL;
bAckedTrans_ToFault     : BOOL;
bAckedTrans_ToNormal    : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
bRawOut                 : BOOL; (* ~(BACnet_RawIoBinaryBoolValue : : ) *)
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Status_Flags*.

bAckedTrans_ToOffNormal, bAckedTrans_ToFault, bAckedTrans_ToNormal: Flags der Property *Acked_Transitions* (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Acked_Transitions*).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

nRawOut: Rohwertausgang des Objekts. Der Ausgang wird mit dem Prozessdatum "RawIoBinaryBoolValue" des BACnet-Objekts verknüpft. Der Wert der Property *Present_Value* wird auf "bRawOut" abgebildet (vorausgesetzt der Objektzustand ist nicht *out_of_service*).

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

Beispiel

Im folgenden Beispiel wird die Umsetzung der Property *Present_Value* eines BinaryOutput-Objekts nach EIB gezeigt:

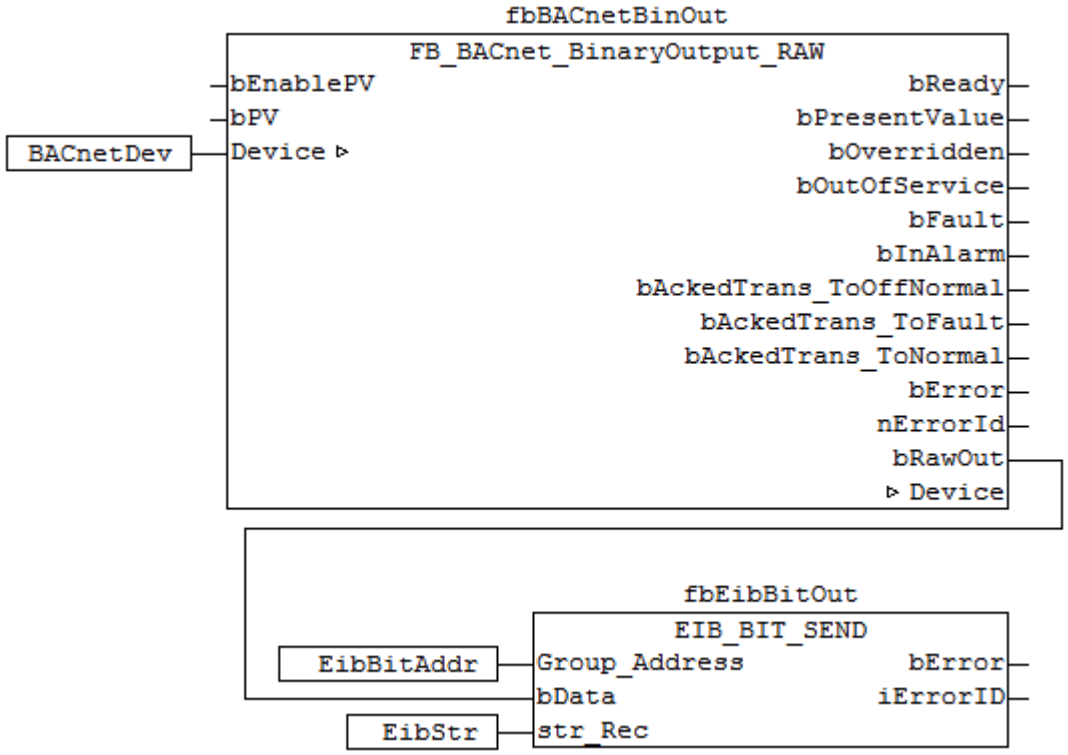
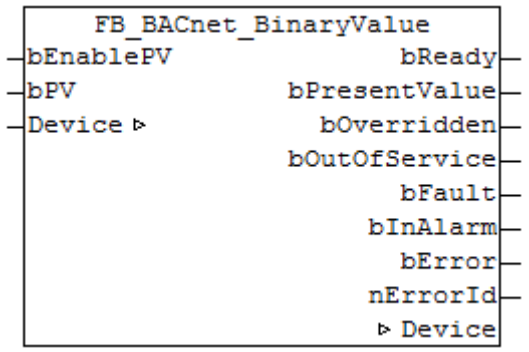


Abb. 113: Bild-4: Beispiel für die Umsetzung der Property Present_Value nach EIB im PLC Programm. Dieses Beispiel setzt die Bibliothek "TcKL6301.lib" voraus. Siehe Dokumentation zur Klemme KL6301 für weitere Informationen zu EIB.

5.3.14 FB_BACnet_BinaryValue

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_BinaryValue" kann lesend und schreibend auf ein BACnet-Objekt vom Typ *BinaryValue* (BV) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

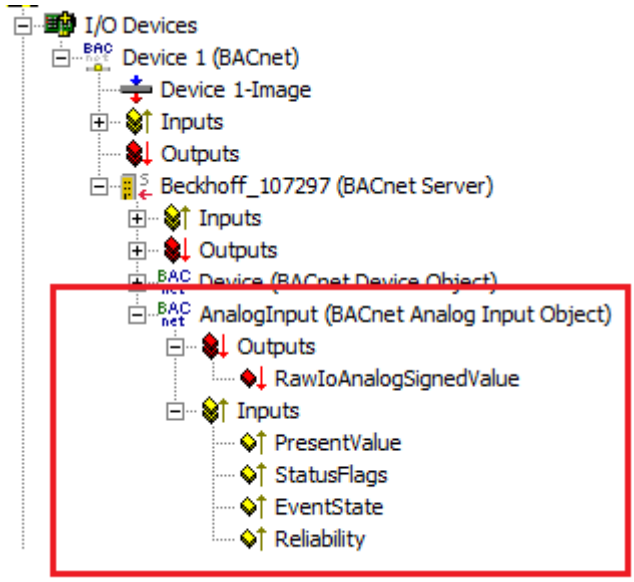


Abb. 114: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV     : BOOL;
bPV           : BOOL;
```

bEnablePV: Gibt den Wert des Eingangs "bPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs "bPV". Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).
Wird bEnablePV auf *FALSE* gesetzt, dann wird der zugehörige Eintrag aus dem *Priority_Array* wieder entfernt (NULL Schreiben).

bPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und 3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben der Prozessdaten erfolgt nach Freigabe immer zyklisch.

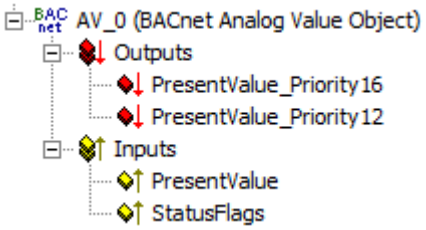


Abb. 115: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

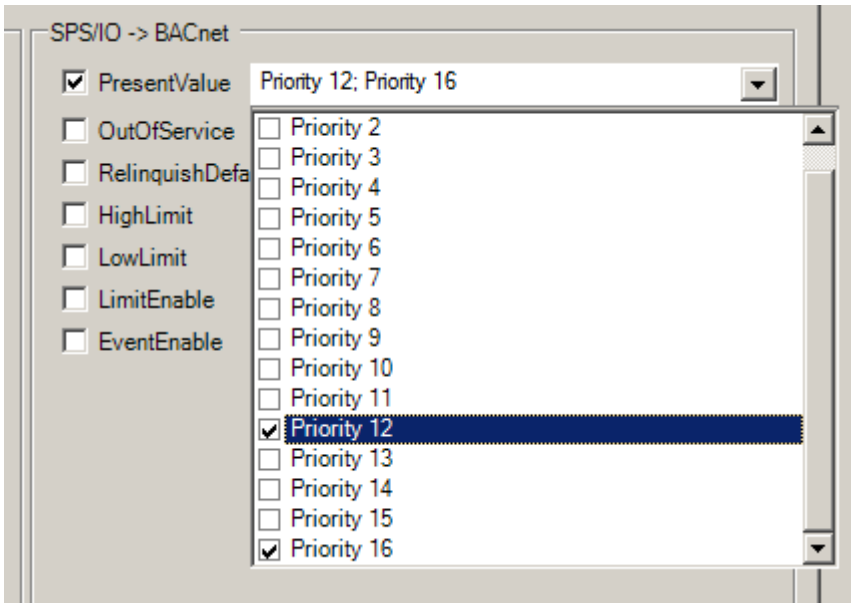


Abb. 116: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 und 16 werden als mappbare Prozessdaten angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady                  : BOOL;
bPresentValue           : BOOL;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

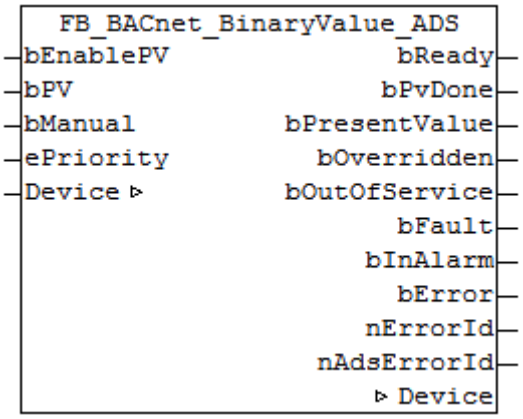
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

5.3.15 FB_BACnet_BinaryValue_ADS

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_BinaryValue_ADS" kann lesend und asynchron schreibend (per ADS) auf ein BACnet-Objekt vom Typ *BinaryValue* (BV) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

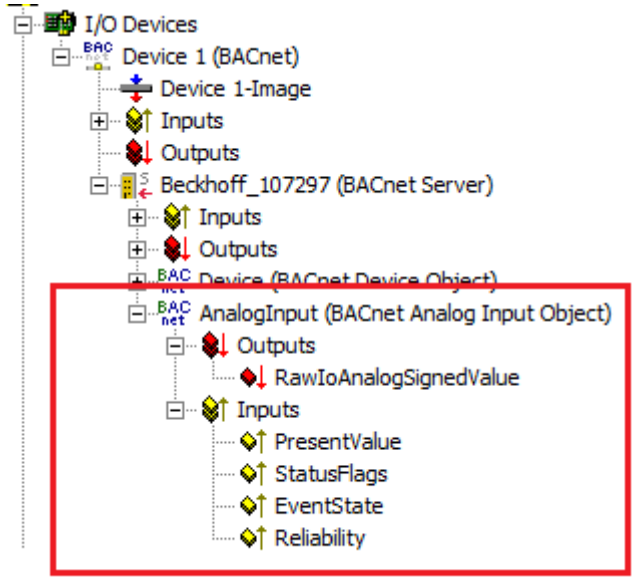


Abb. 117: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV            : BOOL;
bPV                  : BOOL; (* present value bool type *)
bManual              : BOOL; (* FALSE --> TRUE: write present_value immediately,
                FALSE: only write present_value on-change,
                TRUE: write present_value on-change and when device turns operational *)
ePriority            : E_BACNETPRIORITY := BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs "bPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus "bPV" auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist).
Wird der Eingang auf *FALSE* gesetzt erfolgt ein Schreibzugriff mit dem Wert *NULL* auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (NULL Schreiben).

bPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn "bEnablePV" auf *TRUE* gesetzt ist. Die Priorität wird mit dem Eingang "ePriority" bestimmt (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben auf das Objekt wird bei Wertänderung via ADS ausgeführt.

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung geschieht, ist es potenziell möglich dass das Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:

- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE* --> *TRUE*: löst das Schreiben der Property *Present_Value* unmittelbar aus.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 419]).

ePriority: Vorgabe der Priorität der Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 515]).

VAR_OUTPUT

```
bReady           : BOOL;
bPvDone          : BOOL; (* present_value written over ADS *)
bPresentValue    : BOOL;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bError           : BOOL;
nErrorId         : UINT;
nAdsErrorId      : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
4 = der Objekttyp des BACnet Objekts passt nicht zum Funktionsbaustein (Der Objekttyp wird über die

Prozessdaten ausgelesen --> Verknüpfung im System Manager prüfen!)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

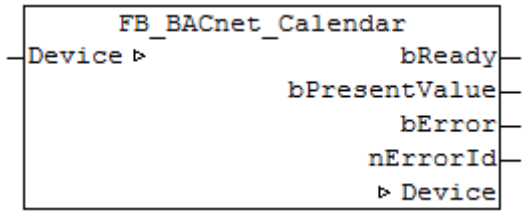
```
Device       : FB_BACnet_Device;
```

Device: Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.16 FB_BACnet_Calendar

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_Calendar" kann lesend auf ein BACnet-Objekt vom Typ *Calendar* (CAL) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

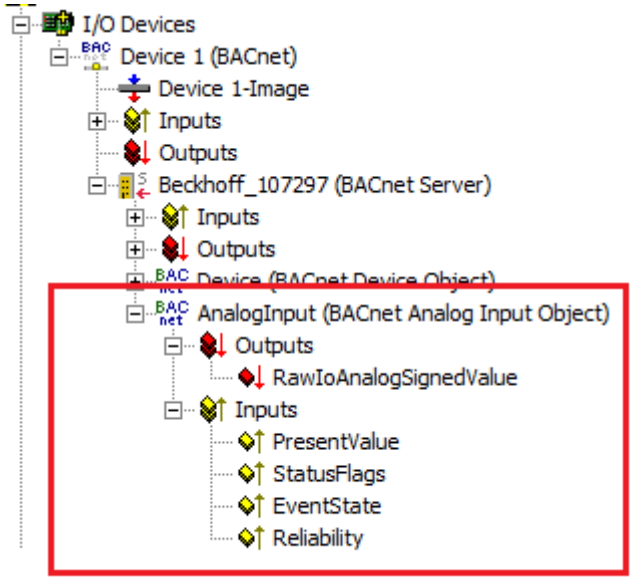


Abb. 118: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_OUTPUT

```
bReady           : BOOL;
bPresentValue    : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational".

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Calendar* und Property *Present_Value*).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

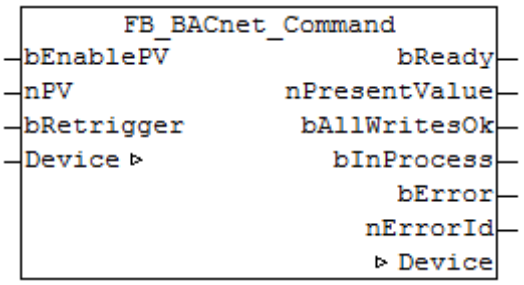
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.17 FB_BACnet_Command

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_Command" kann lesend und schreibend auf ein BACnet-Objekt vom Typ *Command* (CMD) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

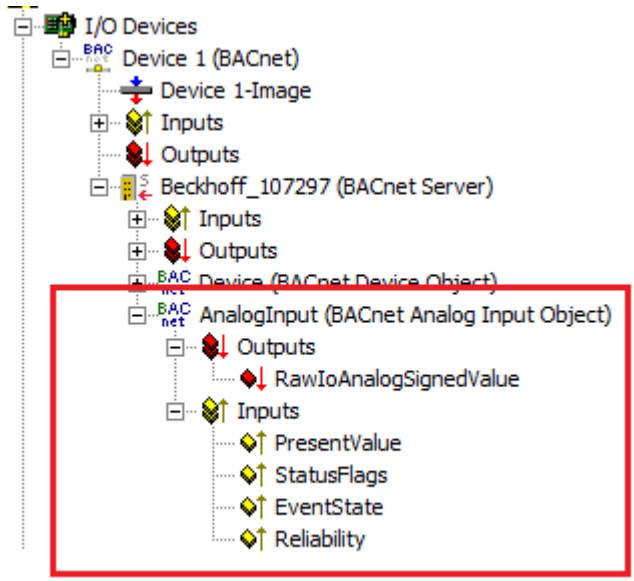


Abb. 119: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV    : BOOL;
nPV          : UDINT;
bRetrigger   : BOOL;
```

bEnablePV: Gibt den Wert des Eingangs **nPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt das Schreiben in die Property *Present_Value* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs **nPV**.
Wird **bEnablePV** auf *FALSE* gesetzt, dann werden die Prozessdaten des gemappten Property *Present_Value* auf *16#FFFFFFFF* geschrieben und damit deaktiviert.

nPV: Wert der Property *Present_Value* der geschrieben werden soll. Liegt der Wert außerhalb des Wertebereicht (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *Present_Value*), dann wird das zugehörige Prozessdatum deaktiviert, d.h. das Schreiben auf die Property ist deaktivert und andere BACnet-Dienste können schreibend auf die Property *Present_Value* zugreifen.
Das Schreiben eines gültigen Wertes löst die Ausführung der zugehörigen Kommando-Liste des BACnet-Objekts aus. Geschrieben wird der Wert der Property Present_Value immer dann, wenn eine Änderung des Prozessdatums erfolgt (d.h. Änderung des Wertes von **nPV** bei gesetztem **bEnablePV** oder Signalswechsel *FALSE* --> *TRUE* am Eingang **bRetrigger** bei gesetztem **bEnablePV**).

bRetrigger: Bei steigender Flanke an diesem Eingang wird das Schreiben und damit die Ausführung des entsprechenden Kommandos des BACnet-Objekts *Command* wiederholt. Der Signalwechsel von *FALSE* --> *TRUE* entspricht einer Änderung des Prozessdatums der Property *Present_Value* von x --> 0 --> x.

VAR_OUTPUT

```
bReady           : BOOL;
nPresentValue    : UDINT;
bAllWritesOk     : BOOL;
bInProcess       : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, AllWritesOk, InProcess). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational".

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *Present_Value*).

bAllWritesOk: Die zuletzt angeforderte Kommando-Liste wurde erfolgreich abgearbeitet (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *All_Writes_Successful*).

bInProcess: Die selektierte Kommando-Liste wird abgearbeitet (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *In_Process*).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.18 FB_BACnet_Device

Der folgende Funktionsbaustein wird für die Anbindung des SPS-Programmes an ein lokales BACnet-Device Objekt (Server) verwendet. Die Verknüpfung des Funktionsbausteins erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell oder mittels PLC-Automapping [▶ 64] automatisch verknüpft werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten. Zu den Prozessdaten des BACnet-Device Objekts sind ebenfalls die nötigen Prozessdaten für die Anbindung des BACnet Servers enthalten.

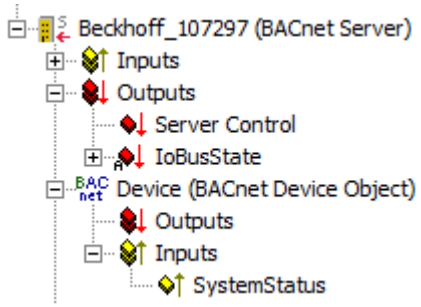


Abb. 120: Bild-1: Prozessdaten des BACnet Device Objekts und -Server im System Manager.

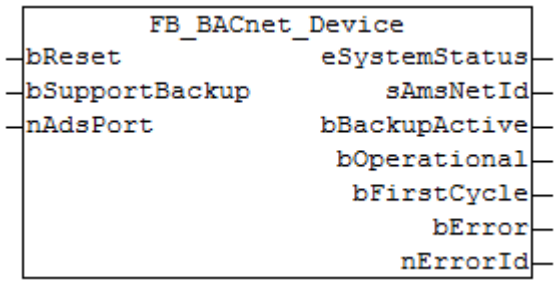


Abb. 121: Bild-2: Funktionsbaustein des BACnet Device Objekts und -Server im SPS-Programm.

Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_Device" wird der Zustand des lokalen BACnet Device Objects gelesen (System_Status) und im SPS-Programm ausgegeben. Zudem wird mit Hilfe des Prozessdatums "Server Control" der lokale BACnet Server gesteuert (Aktivierung und Handhabung des SPS-Backups).

Für die Verwendung des Bausteins "FB_BACnet_Device" wird zudem eine globale Instanz des Bausteins FB_BACnet_Adapter [▶ 378] pro SPS-Projekt benötigt. Der Baustein FB_BACnet_Adapter [▶ 378] stellt die Verbindung zwischen SPS und BACnet-Adapter im System Manager her. Diese globale Instanz wird bereits

von der SPS-Bibliothek bereit gestellt. Im Folgenden ist die Hierarchie zwischen FB_BACnet_Adapter [▶ 378], "FB_BACnet_Device" bzw. FB_BACnet_RemoteDevice [▶ 465] und einem Objekt vom Typ *AnalogValue* (AV) dargestellt:

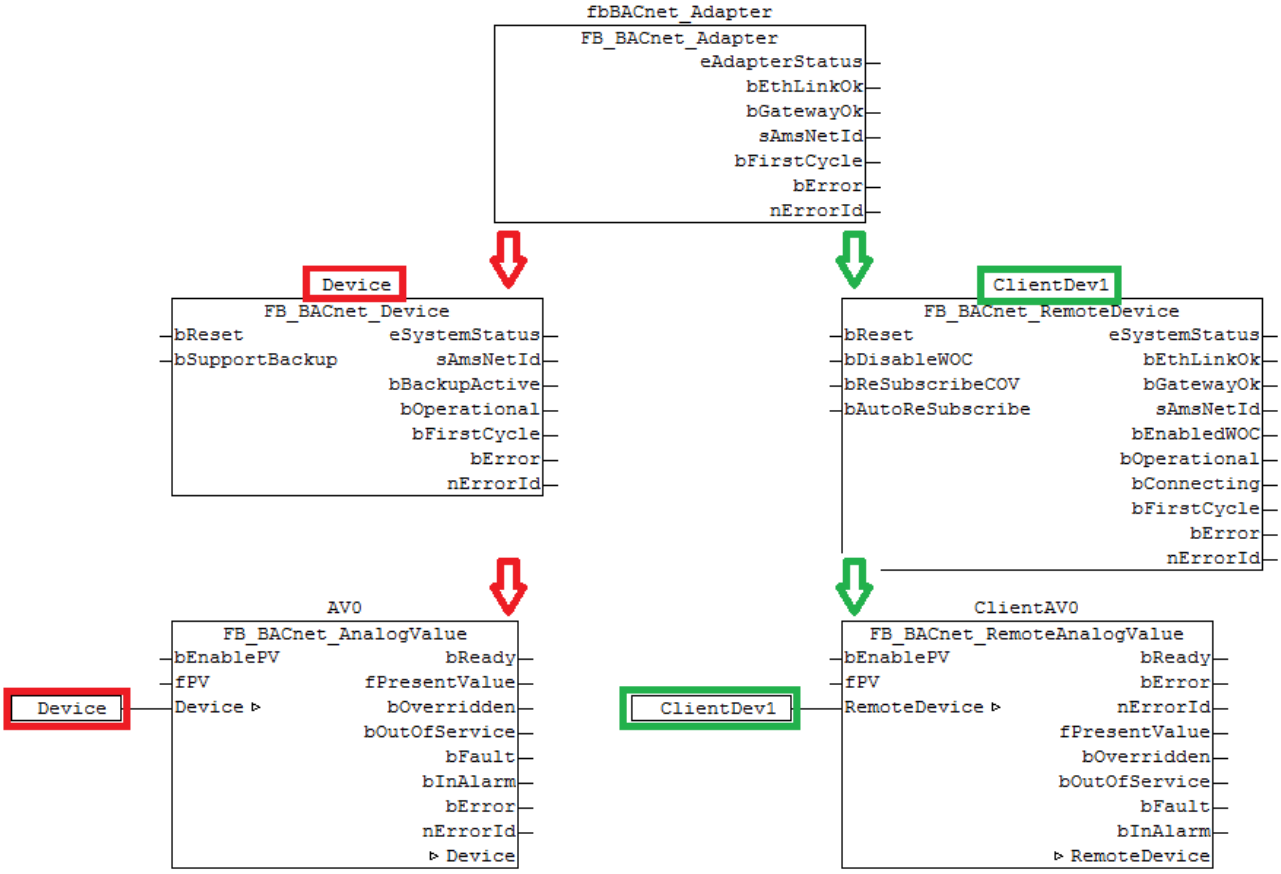


Abb. 122: Bild-3: Beispiel zur Verknüpfung der FB-Instanzen in der SPS.

Für Informationen zur Verwendung der Bausteine FB_BACnet_Adapter [▶ 378] und -_RemoteDevice [▶ 465] siehe FB_BACnet_Adapter [▶ 378] bzw. FB_BACnet_RemoteDevice [▶ 465].

VAR_INPUT

```
bReset           : BOOL;
bSupportBackup   : BOOL;
nAdsPort         : UINT := 1000; (* Optional setting: Ads port of local BACnet-
Server, default = 1000 *)
```

bReset: Zurücksetzen des Fehlerzustands bei Signalwechsel *FALSE --> TRUE*.

bSupportBackup: Bei Setzen des Eingangs auf *TRUE* wird das Sichern der persistenten Daten der SPS-Laufzeit über BACnet unterstützt. Ist dieser Modus aktiviert müssen die persistenten Dateien der SPS (standardmäßig unter: "C:\TwinCAT\Boot\") als BACnet-File Objekte angelegt und mit dem Wert "TwinCAT Configuration File" der Property *File_Type* versehen werden (damit sorgt der BACnet-Stack für die Sicherung der Dateien).
Bei Verwendung des BACnet-seitigen Backups der persistenten SPS-Daten sollte keine weitere Instanz des Bausteins *FB_WritePersistentData* in der SPS-Laufzeit verwendet werden (*FB_WritePersistentData* wird von *FB_BACnet_Device* ausgeführt)!

nAdsPort: ADS Port des lokalen BACnet-Servers. Im Standardfall ist dieser 1000 (vorausgesetzt der BACnet-Server ist das erste Element unterhalt des BACnet-Adapters im System Manager). Die Portnummer kann im System Manager abgelesen werden (siehe Bild-4):

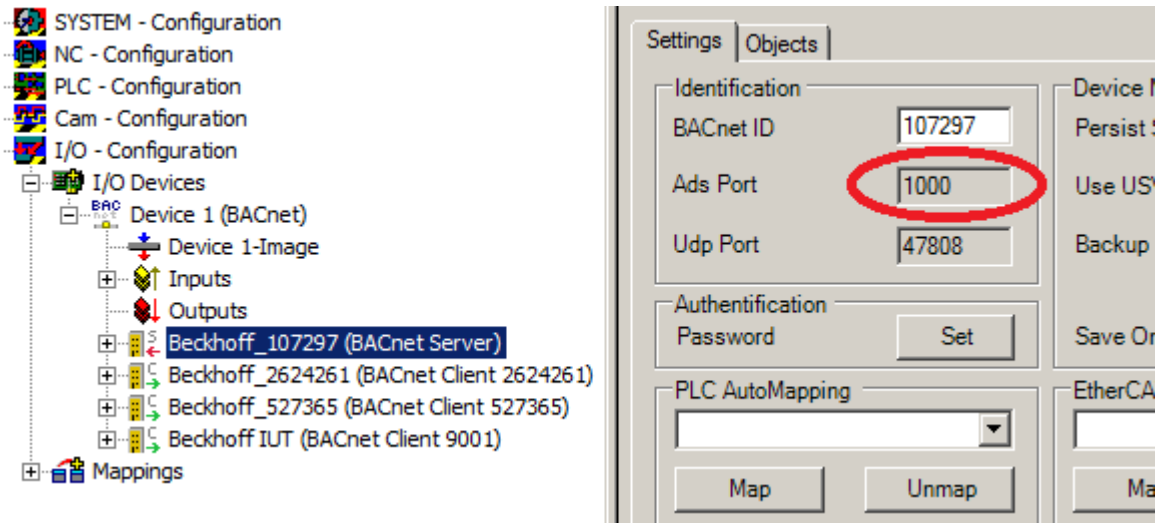


Abb. 123: Bild-4: ADS Port im System Manager

VAR_OUTPUT

```
eSystemStatus    : E_BACnetDeviceStatus;
sAmsNetId        : T_AmsNetId;
bBackupActive    : BOOL;
bOperational     : BOOL;
bFirstCycle      : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

eSystemStatus: Aktueller Status des BACnet Server Objekts (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet Device Objekt und Property *System_Status*).
0: BACnetDeviceStatus_Operational (Betriebsbereit)
1: BACnetDeviceStatus_OperationalReadOnly (Nur-Lese-Zugriff auf Properties)
2: BACnetDeviceStatus_DownloadRequired (Konfiguration-Laden erforderlich)
3: BACnetDeviceStatus_DownloadInProgress (Konfiguration-Laden wird ausgeführt)
4: BACnetDeviceStatus_NonOperational (Nicht-Betriebsbereit)
5: BACnetDeviceStatus_BackupInProgress (Datensicherung wird ausgeführt)

sAmsNetId: Ausgabe der AMS-NetID des lokalen BACnet-Adapters (kann für den asynchronen Zugriff via ADS auf BACnet-Objekte verwendet werden).

bBackupActive: Rückmeldung dass ein Backup via BACnet ausgeführt wird, wenn der Ausgang auf *TRUE* gestzt ist.

bOperational: BACnet-Device Objekt (des lokalen Servers) meldet "Operational" und der Device-Adapter meldet den Status "GetGatewayMAC" (0x03xx) oder "WaitGatewayMAC" (0x04xx) oder "Complete" (0x08xx). Fällt der Ausgang auf *FALSE* werden sämtliche verbundene Objekte (Bausteininstanzen) gesperrt.

bFirstCycle: Wird mit dem ersten Aufruf der Bausteininstanz nach SPS-Reset bzw. -Neustart für einen Zyklus gesetzt.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = keine gültige AMS-NetID
20 = Fehler beim Schreiben der persistenten Daten
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Device*.**nERR_xxx**). Alle Fehler mit Fehlernummer größer oder gleich 20 müssen mittels **bReset** quittiert werden.

Eine Instanz des Bausteins FB_BACnet_Device ist immer dann nötig, wenn auf Objekte eines lokalen BACnet-Servers aus der SPS zugegriffen werden soll. Mehrere Instanzen des Bausteins FB_BACnet_Device können prinzipiell angelegt werden, führen jedoch zu einer Fehlermeldung bei Verwendung des SPS-Automappings im System Manager. Der Grund dafür liegt in der Architektur von BACnet. Unter einem BACnet-Adapter (Netzwerk-Device) kann immer nur ein BACnet-Server angelegt

werden. Wird jedoch auf das SPS-Automapping verzichtet, können x-beliebige BACnet-Adapter und -Server in einem SPS-Projekt verwendet werden. Die Verknüpfung der Instanzen wird dann von Hand vorgenommen.

Der Baustein FB_BACnet_Device greift intern auf die globale Variable "pBACnet_Adapter" zu. Diese enthält eine Referenz zur globalen Instanz des Bausteins FB_BACnet_Adapter. Sollen mehrere Instanzen des Bausteins FB_BACnet_Device angelegt und von Hand im System Manager mit den entsprechenden BACnet-Servern verknüpft werden, so müssen auch mehrere Instanzen des Bausteins FB_BACnet_Adapter angelegt werden. Vor dem Aufruf der Instanz des BACnet-Server Bausteins muss zwingend vorab die zugehörige Instanz des BACnet-Adapters Bausteins erfolgen. Im Folgenden ein CFC-Beispiel (Wichtig ist die Aufrufreihenfolge der Bausteine zu beachten!):

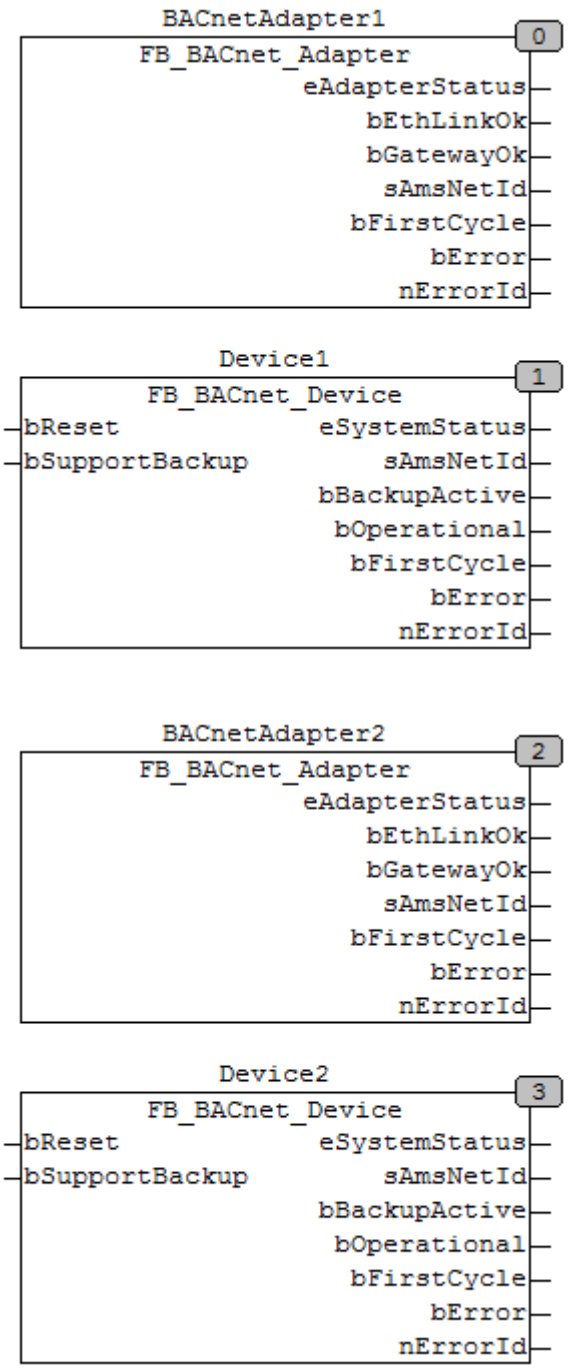


Abb. 124: Bild-5: Beispiel für das Verwenden von mehreren Adapter- und Server-Instanzen in einem SPS-Projekt.

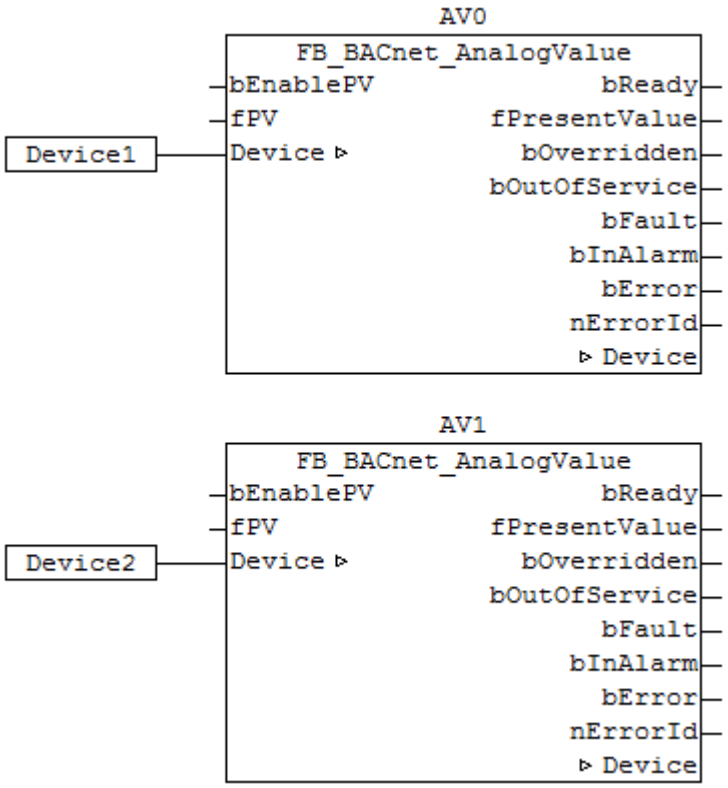


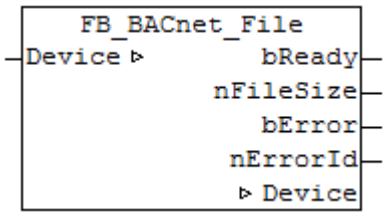
Abb. 125: Bild-6: Beispiel für den Aufruf von Bausteinen vom Typ FB_BACnet_AnalogValue mit unterschiedlichen lokalen BACnet-Servern in einem SPS-Projekt.

Die Verwendung von mehreren lokalen BACnet-Servern (-Adapter) pro SPS-Projekt ist ein Spezialfall und wird nicht empfohlen. Sinnvoll wäre (z.B. bei Verwendung mehrerer Netzwerkkarten und damit getrennte BACnet-Netzwerke) die Verwendung mehrerer SPS-Projekte und damit mehrere SPS-Laufzeiten. Der Datenaustausch zwischen den einzelnen Laufzeiten kann via Prozessdatenmapping oder ADS-Kommunikation realisiert werden. Somit wäre auch die Verwendung des SPS-Automappings für BACnet-Objekte möglich.

5.3.19 FB_BACnet_File

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_File" kann lesend auf ein BACnet-Objekt vom Typ *File* (FILE) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

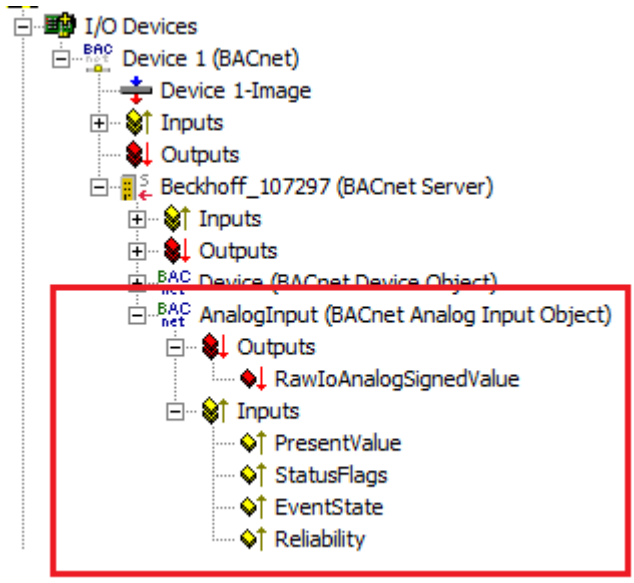


Abb. 126: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_OUTPUT

```
bReady          : BOOL;
bPresentValue   : BOOL;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational".

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *File* und Property *Present_Value*).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

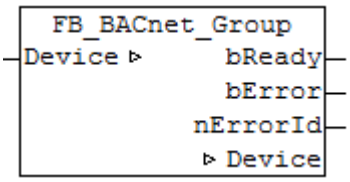
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.20 FB_BACnet_Group

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Der Funktionsbausteins "FB_BACnet_Group" dient als Platzhalter für zukünftige Funktionalitäten.

VAR_OUTPUT

```
bReady       : BOOL;
bError       : BOOL;
nErrorId     : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device [▶ 419] nicht "Operational".

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device [▶ 419])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

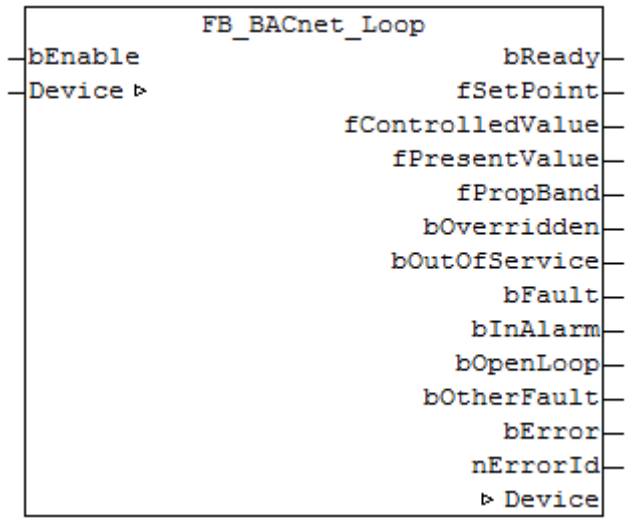
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.21 FB_BACnet_Loop

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_Loop" wird die Objekt-Funktionalität für den BACnet-Stack zur Verfügung gestellt. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt. Das BACnet-Objekt auf Seite BACnet-Stack (System Manager) dient als Interface zur BACnet-Welt. Die Regelung und damit Funktionalität des BACnet-Objekts wird durch die SPS-Instanz des Bausteins "FB_BACnet_Loop" bereit gestellt.

Diese Herangehensweise hat den Vorteil, dass damit bereits bestehende, kundenspezifische Regler aus einem SPS-Projekt leicht mit der BACnet-Welt verknüpft werden können. Lediglich die Prozessdaten des BACnet-LOOP-Objekts müssen bereitgestellt und verarbeitet werden.

Der Funktionsbaustein "FB_BACnet_Loop" aus der TcBACnet-Lib enthält einen Standard PID-Regler (FB_BACnet_PidControl [▶ 499]) und deckt damit ein breites Spektrum an Regelaufgaben ab.

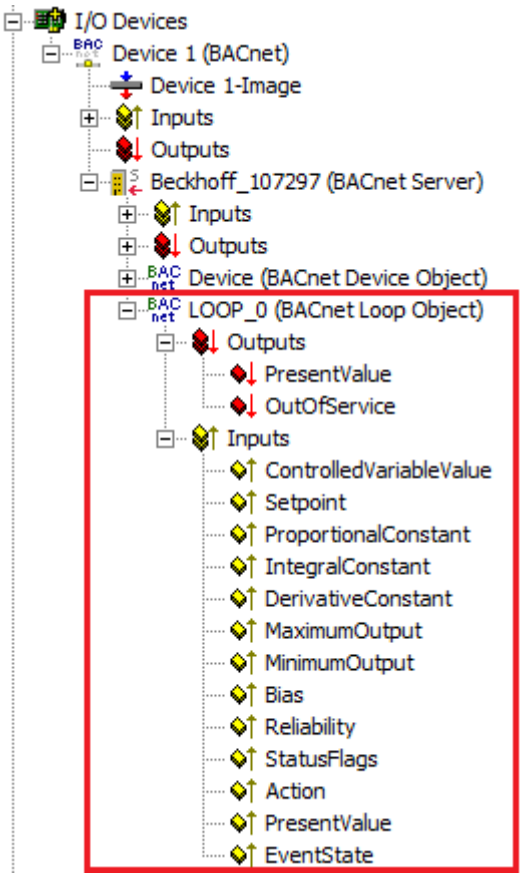


Abb. 127: Bild 1: Beispiel eines BACnet-LOOP-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnable      : BOOL;
```

bEnable: Gibt die Regelung frei. *FALSE* am Eingang setzt den Zustand des BACnet-Objekts auf *Out_Of_Service* (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Out_Of_Service*).

VAR_OUTPUT

```
bReady           : BOOL;
fSetPoint        : REAL;
fControlledValue : REAL;
fPresentValue    : REAL;
fPropBand        : REAL;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bOpenLoop        : BOOL;
bOtherFault      : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

fSetPoint: Rückmeldung der Regelvorgabe (W, Sollwert).

fControlledValue: Rückmeldung der aktuellen Prozessgröße (X, Istwert).

fPresentValue: Rückmeldung der aktuellen Regelausgabe (Y, Stellwert). Achtung: *Present_Value* und *Controlled_Variable_Value* können schnell zu Verwechslungen führen (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Properties *Present_Value*, *Controlled_Variable_Value* und *Controlled_Variable_Reference*)!

fPropBand: Rückmeldung der aktuellen Regelausgabe in Prozent (-100%...+100%) in Relation zur minimalen und maximalen Regelausgabe (Properties *Minimum_Output* und *Maximum_Output*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Status_Flags*.

bOpenLoop, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Reliability*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device [▶ 419])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

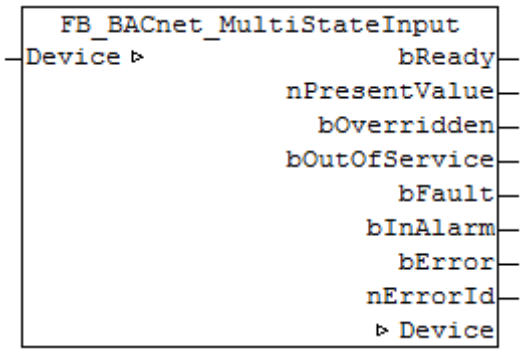
Konfiguration des Reglers

Die Konfiguration des Reglers erfolgt mit Hilfe der BACnet-Properties: *Action*, *Proportional_Constant* (P-Faktor), *Integral_Constant* (I-Faktor), *Derivative_Constant* (D-Faktor), Bias (Ausgabe-Offset), *Maximum_Output* (Maximale Regelausgabe) und *Minimum_Output* (Minimale Regelausgabe). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop*.

5.3.22 FB_BACnet_MultiStateInput

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_MultiStateInput" kann lesend auf ein BACnet-Objekt vom Typ *MultiStateInput* (MI) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

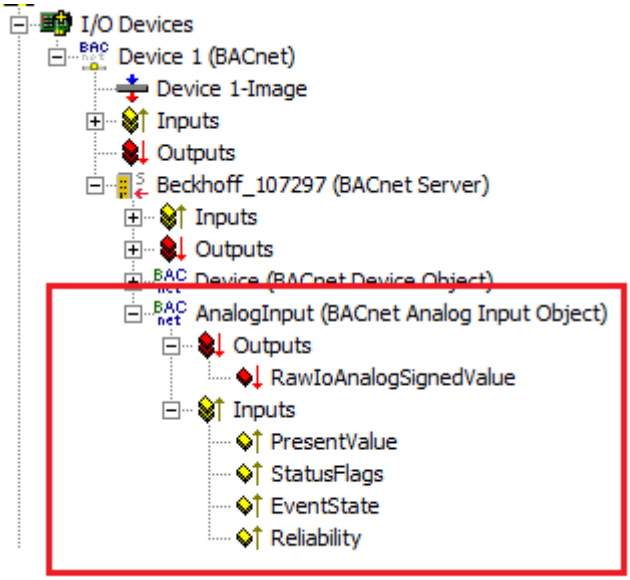


Abb. 128: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_OUTPUT

```
bReady                  : BOOL;
nPresentValue           : UDINT;
bOverridden             : BOOL;
```

```
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

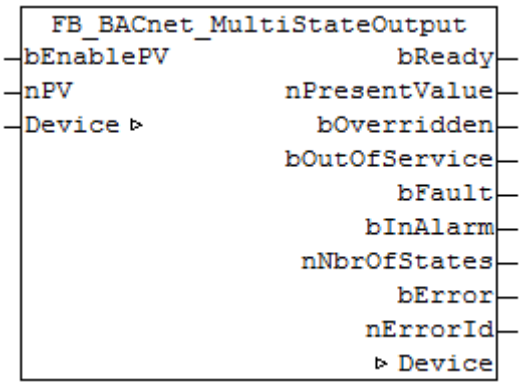
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

5.3.23 FB_BACnet_MultiStateOutput

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_MultiStateOutput" kann lesend und schreibend auf ein BACnet-Objekt vom Typ *MultiStateOutput* (MO) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

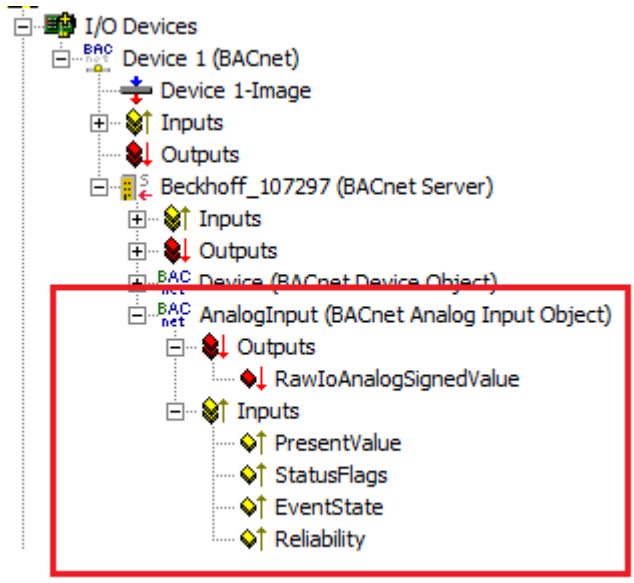


Abb. 129: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV    : BOOL;
nPV          : UDINT;
```

bEnablePV: Gibt den Wert des Eingangs **nPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs **nPV**. Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).
Wird bEnablePV auf FALSE gesetzt, dann wird der zugehörige Eintrag aus dem *Priority_Array* wieder entfernt (NULL Schreiben).

nPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und -3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs **bEnablePV** freigegeben. Das Schreiben der Prozessdaten erfolgt nach Freigabe immer zyklisch.

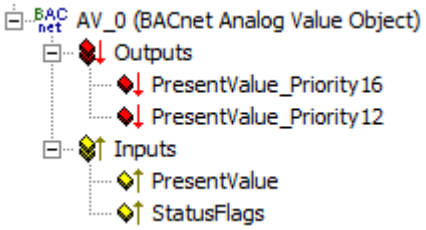


Abb. 130: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

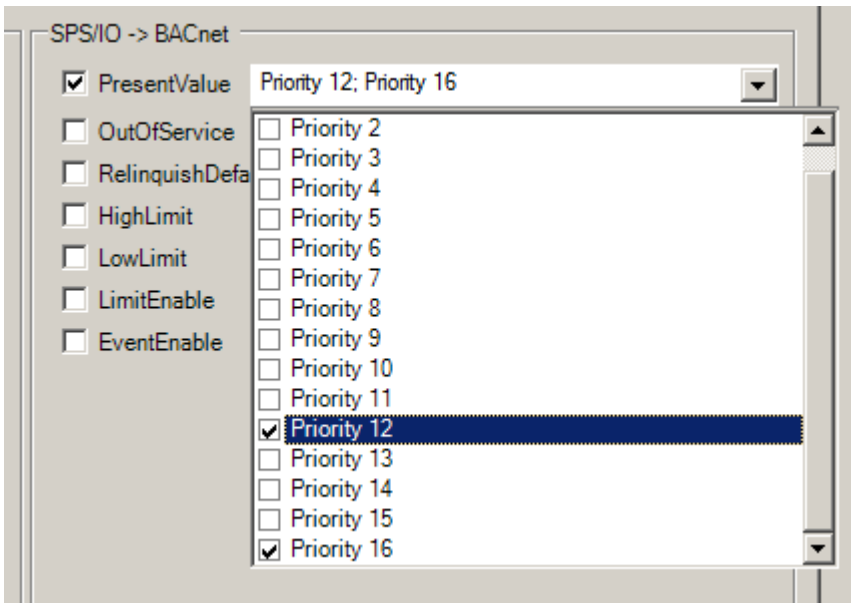


Abb. 131: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 und 16 werden als mappbare Prozessdaten angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady                  : BOOL;
nPresentValue           : UDINT;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

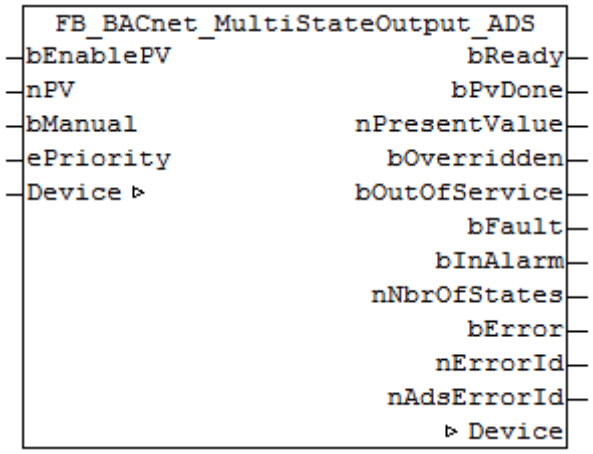
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

5.3.24 FB_BACnet_MultiStateOutput_ADS

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_MultiStateOutput_ADS" kann lesend und asynchron schreibend (per ADS) auf ein BACnet-Objekt vom Typ *MultiStateOutput* (MO) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

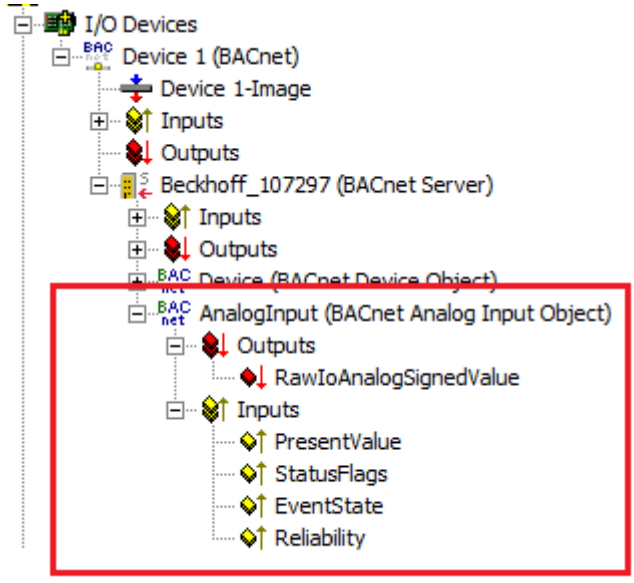


Abb. 132: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV           : BOOL;
nPV                 : UDINT; (* number for priority x, when input bEnablePV is TRUE *)
bManual             : BOOL; (* FALSE --> TRUE: write present_value immediately,
                FALSE: only write present_value on-change,
                TRUE: write present_value on-change and when device turns operational *)
ePriority           : E_BACNETPRIORITY := BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs "nPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus "nPV" auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist).
Wird der Eingang auf *FALSE* gesetzt erfolgt ein Schreibzugriff mit dem Wert *NULL* auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (NULL Schreiben).

nPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn "bEnablePV" auf *TRUE* gesetzt ist. Die Priorität wird mit dem Eingang "ePriority" bestimmt (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben auf das Objekt wird bei Wertänderung via ADS ausgeführt.

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung durchgeführt wird, ist es potenziell möglich dass das Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:

- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE --> TRUE*: löst das Schreiben der Property *Present_Value* unmittelbar aus.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 419]).

ePriority: Vorgabe der Priorität der Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 515]).

VAR_OUTPUT

```
bReady           : BOOL;
bPvDone          : BOOL; (* present_value written over ADS *)
nPresentValue    : UDINT;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
nNbrOfStates     : UDINT;
bError           : BOOL;
nErrorId         : UINT;
nAdsErrorId      : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Status_Flags*.

nNbrOfStates: Anzahl Zustände die die Property *Present_Value* annehmen kann. Widerspiegelt den Wert der Property *Number_Of_States*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
4 = der Objekttyp des BACnet Objekts passt nicht zum Funktionsbaustein (Der Objekttyp wird über die Prozessdaten ausgelesen --> Verknüpfung im System Manager prüfen!)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

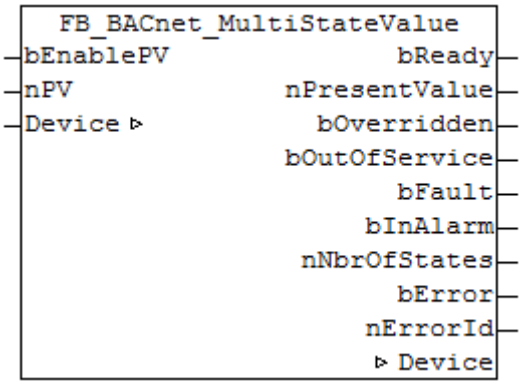
```
Device       : FB_BACnet_Device;
```

Device: Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.25 FB_BACnet_MultiStateValue

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_MultiStateValue" kann lesend und schreibend auf ein BACnet-Objekt vom Typ *MultiStateValue* (MV) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

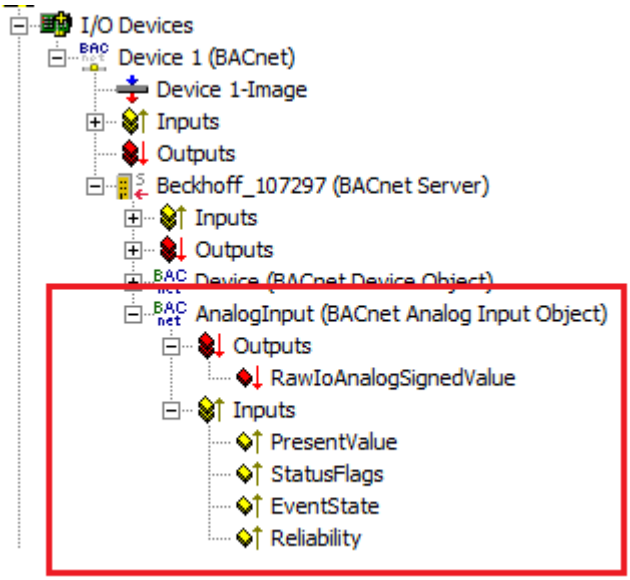


Abb. 133: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV    : BOOL;
nPV          : UDINT;
```

bEnablePV: Gibt den Wert des Eingangs **nPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs **nPV**. Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager). Wird bEnablePV auf *FALSE* gesetzt, dann wird der zugehörige Eintrag aus dem *Priority_Array* wieder entfernt (NULL Schreiben).

nPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und -3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs **bEnablePV** freigegeben. Das Schreiben der Prozessdaten erfolgt nach Freigabe immer zyklisch.

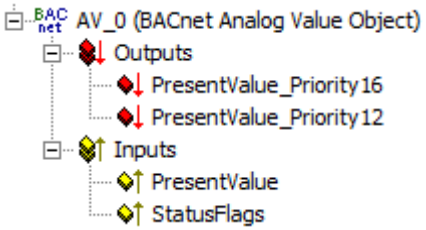


Abb. 134: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

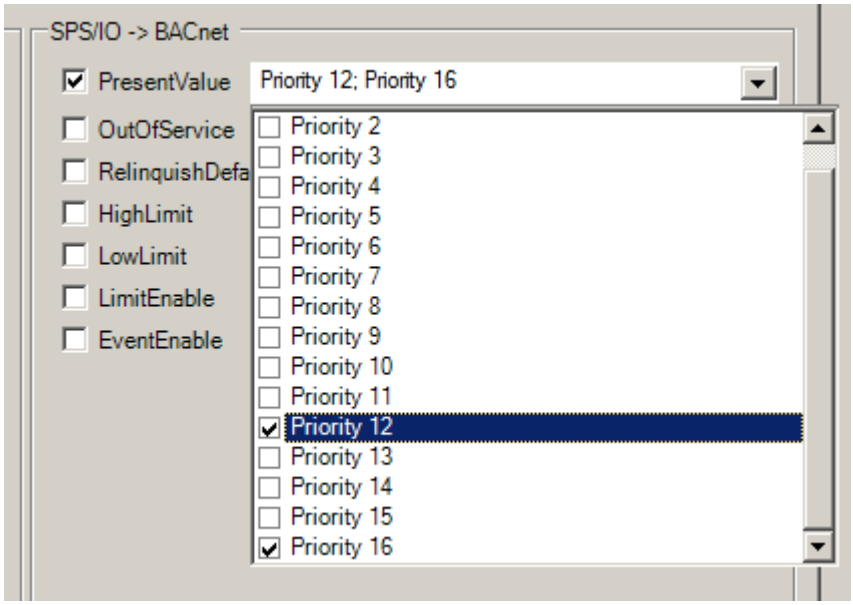


Abb. 135: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 und 16 werden als mappbare Prozessdaten angelegt (siehe Ergebnis in Bild 2).

VAR_OUTPUT

```
bReady                  : BOOL;
nPresentValue           : UDINT;
bOverridden             : BOOL;
bOutOfService           : BOOL;
bFault                  : BOOL;
bInAlarm                : BOOL;
nNbrOfStates            : UDINT;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Status_Flags*.

nNbrOfStates: Meldet die verfügbare Anzahl von Werten, die das *Present_Value* des MultiStateValue-Objekts annehmen kann (1...*nNbrOfStates*). Ist "nNbrOfStates" gleich 0, dann gibt es keinen gültigen "State" den das Objekt annehmen kann (*Present_Value* ist 0).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

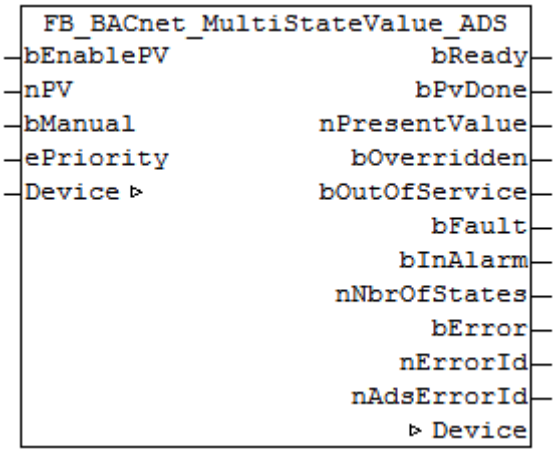
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

5.3.26 FB_BACnet_MultiStateValue_ADS

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_MultiStateValue_ADS" kann lesend und asynchron schreibend (per ADS) auf ein BACnet-Objekt vom Typ *MultiStateValue* (MV) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

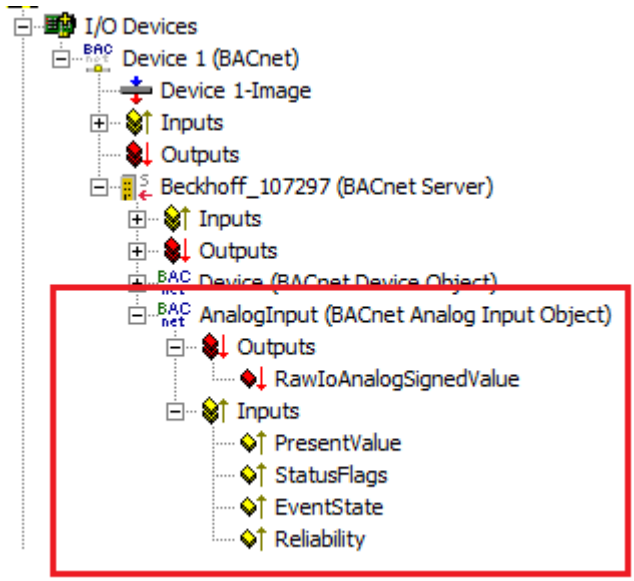


Abb. 136: Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bEnablePV            : BOOL;
nPV                  : UDINT; (* number for priority x, when input bEnablePV is TRUE *)
bManual              : BOOL; (* FALSE --> TRUE: write present_value immediately,
                FALSE: only write present_value on-change,
                TRUE: write present_value on-change and when device turns operational *)
ePriority            : E_BACNETPRIORITY := BACNETPRIORITY_12;
```

bEnablePV: Gibt den Wert des Eingangs "nPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird erfolgt ein Schreibzugriff mit dem Wert aus "nPV" auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist).
Wird der Eingang auf *FALSE* gesetzt erfolgt ein Schreibzugriff mit dem Wert *NULL* auf das kommandierbare Property *Present_Value* mit Priorität aus "ePriority" (NULL Schreiben).

nPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll, wenn "bEnablePV" auf *TRUE* gesetzt ist. Die Priorität wird mit dem Eingang "ePriority" bestimmt (Default: 12, wenn der Eingang "ePriority" nicht beschaltet ist). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben auf das Objekt wird bei Wertänderung via ADS ausgeführt.

Da das Schreiben der Property *Present_Value* azyklisch und nur bei Wertänderung durchgeführt wird, ist es potenziell möglich dass das Property *Present_Value* mit gleicher Priorität auch von anderen BACnet-Geräten überschrieben wird. Bei gleicher Priorität "gewinnt" immer der letzte Schreibzugriff.

bManual: Folgende Auswertung werden für den Eingang unterschieden:
- *FALSE*: Schreiben bei Wertänderung
- Flankenwechsel *FALSE* --> *TRUE*: löst das Schreiben der Property *Present_Value* unmittelbar aus.
- *TRUE*: Schreiben bei Wertänderung *und* wenn der zugehörige BACnet-Server in den Zustand "Operational" wechselt (siehe FB_BACnet_Device [▶ 419]).

ePriority: Vorgabe der Priorität der Schreibzugriffe auf die Property *Present_Value* (Default: 12, siehe auch E_BACNETPRIORITY [▶ 515]).

VAR_OUTPUT

```
bReady               : BOOL;
bPvDone              : BOOL; (* present_value written over ADS *)
nPresentValue        : UDINT;
bOverridden          : BOOL;
bOutOfService        : BOOL;
bFault               : BOOL;
bInAlarm             : BOOL;
nNbrOfStates         : UDINT;
```

```
bError          : BOOL;
nErrorId        : UINT;
nAdsErrorId     : UDINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPvDone: Positive Flanke bei erfolgreichem Schreiben der Property *Present_Value* über ADS.

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Status_Flags*.

nNbrOfStates: Anzahl Zustände die die Property *Present_Value* annehmen kann. Widerspiegelt den Wert der Property *Number_Of_States*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
4 = der Objekttyp des BACnet Objekts passt nicht zum Funktionsbaustein (Der Objekttyp wird über die Prozessdaten ausgelesen --> Verknüpfung im System Manager prüfen!)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

nAdsErrorId: ADS Fehlercode.

VAR_IN_OUT

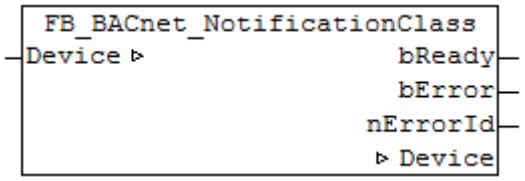
```
Device       : FB_BACnet_Device;
```

Device: Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.27 FB_BACnet_NotificationClass

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Der Funktionsbausteins "FB_BACnet_NotificationClass" dient als Platzhalter für zukünftige Funktionalitäten.

VAR_OUTPUT

```
bReady       : BOOL;
bError       : BOOL;
nErrorId     : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational".

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

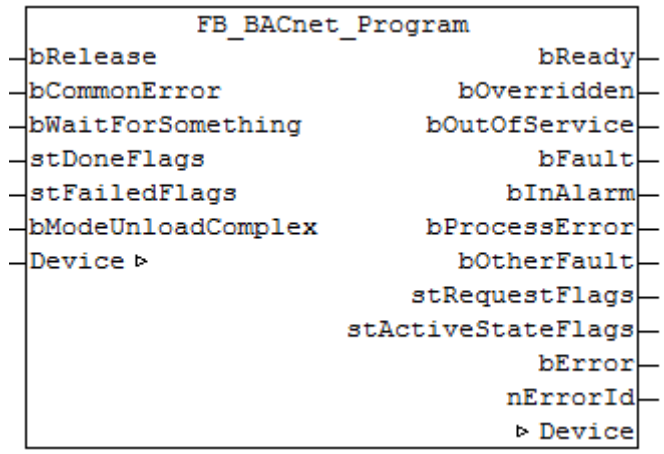
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.28 FB_BACnet_Program

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_Program" wird die Objekt-Funktionalität für den BACnet-Stack zur Verfügung gestellt. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt. Das BACnet-Objekt auf Seite BACnet-Stack dient als Interface zur BACnet-Welt. Die Abarbeitung der Funktionalität des BACnet-Objekts wird durch die SPS-Instanz des Bausteins "FB_BACnet_Program" bereitgestellt.

Diese Herangehensweise hat den Vorteil, dass damit bereits bestehende, kundenspezifische SPS-Funktionen leicht mit der BACnet-Welt verknüpft werden können. In SPS-Code implementierte Abläufe können so mit Hilfe des Handshake-Interfaces (*stDoneFlags*, *stFailedFlags* und *stRequestFlags*) des FB "FB_BACnet_Program" getriggert werden.

Der Funktionsbaustein "FB_BACnet_Program" enthält die in der BACnet-Spezifikation beschrieben *State-Transistions* und bildet diese auf dem FB-Interface mit Hilfe der Strukturen: *stDoneFlags*, *stFailedFlags* und *stRequestFlags* ab (siehe unter Transition-Diagram [▶ 442] Bild-2).

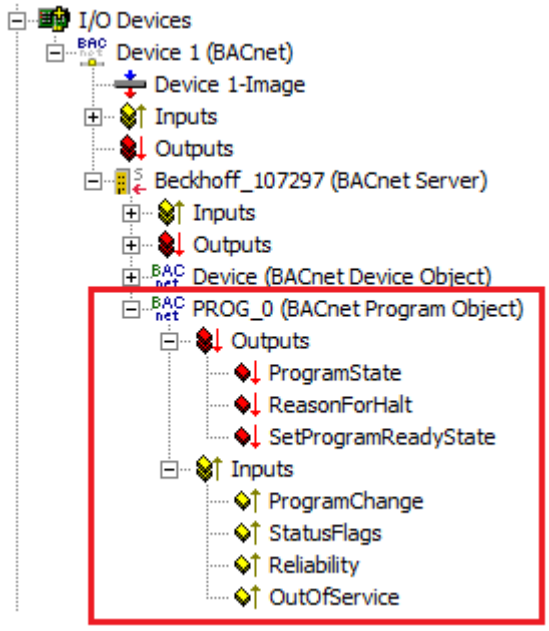


Abb. 137: Bild-1: Beispiel eines BACnet-Program-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_INPUT

```
bRelease           : BOOL;
bCommonError       : BOOL;
bWaitForSomething  : BOOL;
stDoneFlags        : ST_BACnet_ProgramHandshakeRequests;
stFailedFlags      : ST_BACnet_ProgramHandshakeRequests;
bModeUnloadComplex : BOOL := TRUE;
```

bRelease: Gibt die Freigabe für das Abarbeiten von Anforderungen. Wenn **bRelease** = *FALSE* gesetzt ist, dann verharrt des Programm im Zustand *HALTED*.

bCommonError: Fehler bei der Ausführung eines *Program_Change*-Requestsaus der Struktur **stRequestFlags**. Das Setzten des Eingangs bei laufender Ausführung eines *Program_Change*-Requests führt zu einer Abbruchbedingung (siehe unter Transition-Diagram Bild-2).

bWaitForSomething: Befindet sich das BACnet-Objekt im Zustand *RUNNING* kann mit *TRUE* Setzen des Eingangs auf den Zustand *WAITING* umgeschaltet werden, so dass das BACnet-Objekt für weitere Requests außer *Unload* und *Halt* gesperrt wird (siehe unter Transition-Diagram Bild-2).
Um in den Zustand *RUNNING* zurück zu kehren muss der Eingang auf *FALSE* zurück gesetzt werden.

stDoneFlags: Die Flags innerhalb der Eingangsstruktur dienen der Quittierung von anstehenden *Program_Change*-Requests aus der Ausgangsstruktur **stRequestFlags**. Ein Signalwechsel eines der Flags von *FALSE --> TRUE* bestätigt die Ausführung des entsprechenden *Program_Change*-Requests.

stFailedFlags: Die Flags innerhalb der Eingangsstruktur dienen der Abbruchmeldung des anstehenden *Program_Change*-Requests aus der Ausgangsstruktur **stRequestFlags**. Ein Signalwechsel eines der Flags von *FALSE --> TRUE* bricht die Ausführung des entsprechenden *Program_Change*-Requests ab.

bModeUnloadComplex: Wurde der Modus "Unload Complex" mit Hilfe des Eingangs aktiviert, so wechselt der BACnet-Objekt Status von *UNLOADING* zu *IDLE* nach der Ausführung des Requests *Unload* (siehe unter Transition-Diagram Bild-2). Wird der Eingang auf *FALSE* gesetzt, dann verharrt das BACnet-Objekt im Zustand *UNLOADING* nach dem der Request *Unload* ausgeführt wurde.

VAR_OUTPUT

```
bReady             : BOOL;
bOverridden        : BOOL;
bOutOfService      : BOOL;
bFault             : BOOL;
```

```
bInAlarm           : BOOL;
bProcessError      : BOOL;
bOtherFault        : BOOL;
stRequestFlags     : ST_BACnet_ProgramHandshakeRequests;
stActiveStateFlags : ST_BACnet_ProgramHandshakeStates;
bError             : BOOL;
nErrorId           : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (ActiveStateFlags, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Program* und Property *Status_Flags*.

bProcessError, bOtherFault: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Program* und Property *Reliability*.

stRequestFlags: Enthält die Flags zu den entsprechenden *Program_Change*-Requests. Ist ein Flag auf *TRUE* gesetzt, so wird eine Statusänderung des Programm-Objekts durch das SPS-Programm angefordert. Wurde die Änderung erfolgreich ausgeführt wird dies mit einem Signalwechsel *FALSE* --> *TRUE* des entsprechenden Flags der Eingangsstruktur **stDoneFlags** bestätigt. Kann der Request von dem zugehörigen SPS-Programm nicht ausgeführt werden oder gibt es einen Abbruch während der Ausführung, so muss dies mit Hilfe des entsprechenden Flags der Eingangsstruktur **stFailedFlags** gemeldet werden. Der Abbruch wird dann im Status des BACnet-Objekts sichtbar.

stActiveStateFlags: Enthält Flags die den aktuellen Zustand des BACnet-Program-Objects widerspiegeln (siehe unter Transition-Diagram Bild-2).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

Transition-Diagram

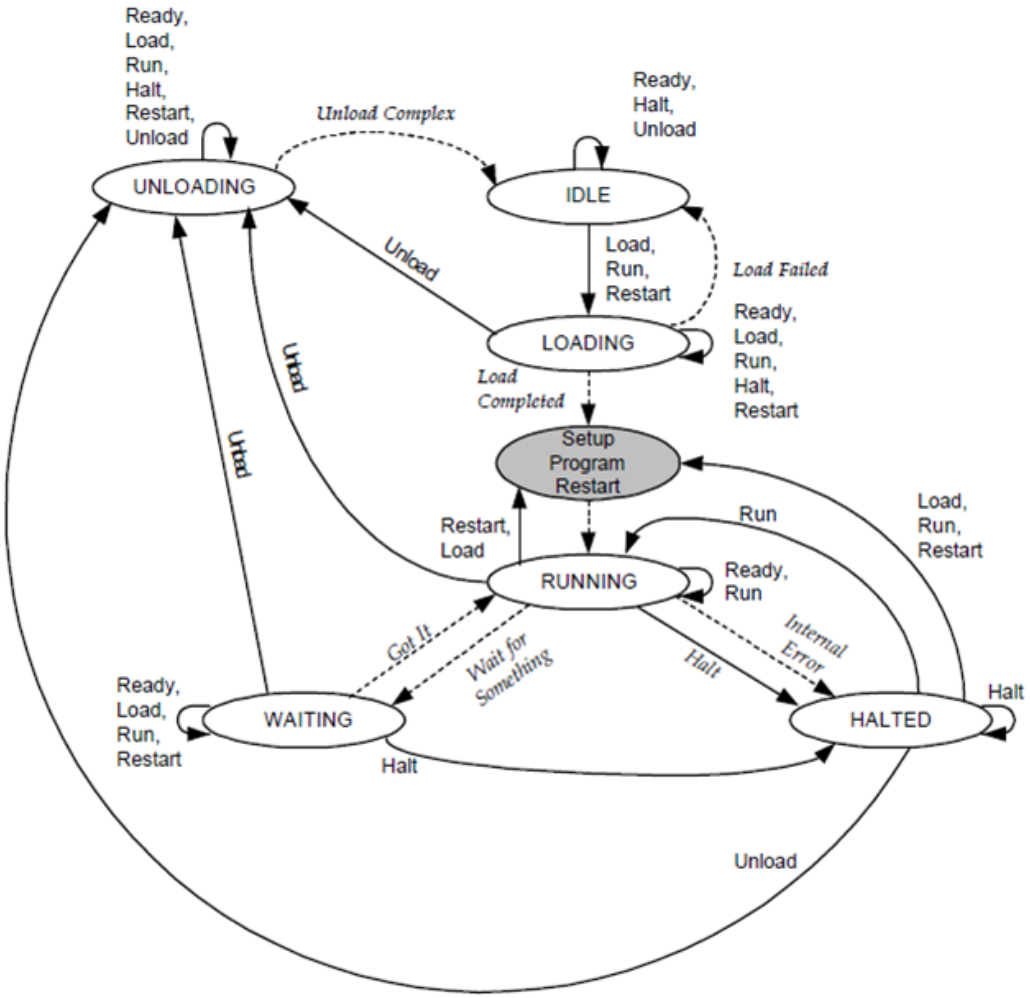
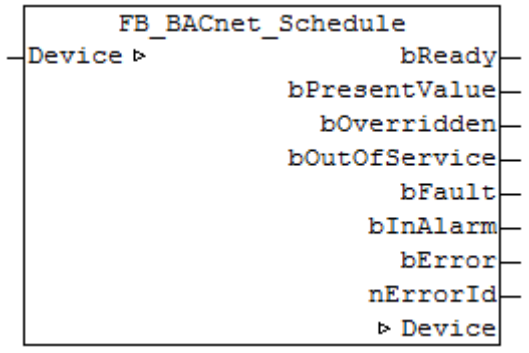


Bild-2: aus BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt Program, Abbildung 12-3 "State Transitions for the program object"

5.3.29 FB_BACnet_Schedule

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_Schedule" kann lesend auf ein BACnet-Objekt vom Typ *Schedule* (SCHED) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

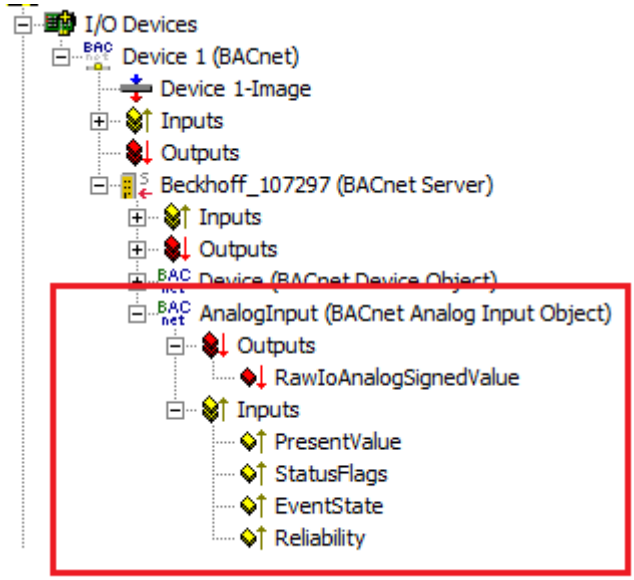


Bild-1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_OUTPUT

```
bReady           : BOOL;
bPresentValue    : BOOL;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_Device" nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Schedule* und Property *Present_Value*). Der Zustand wird in 3 Stufen (Enumeration) angegeben (0 = INACTIVE, 1 = ACTIVE und 2 = NULL).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Schedule* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device [▶ 419])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

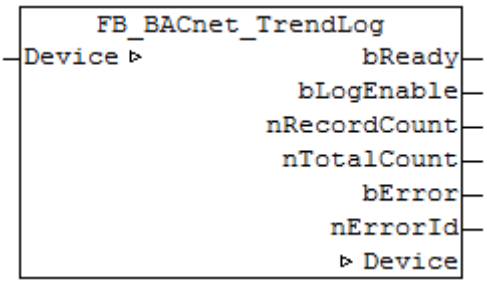
```
Device       : FB_BACnet_Device;
```

Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_Device [▶ 419] für weitere Informationen.

5.3.30 FB_BACnet_TrendLog

Der folgende Funktionsbaustein wird für die Verbindung von einem BACnet-Objekt des lokalen BACnet-Servers verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_TrendLog" kann lesend auf ein BACnet-Objekt vom Typ *TrendLog* (TREND) zugegriffen werden. Das BACnet-Objekt wurde dazu unter einem lokalen BACnet-Server angelegt.

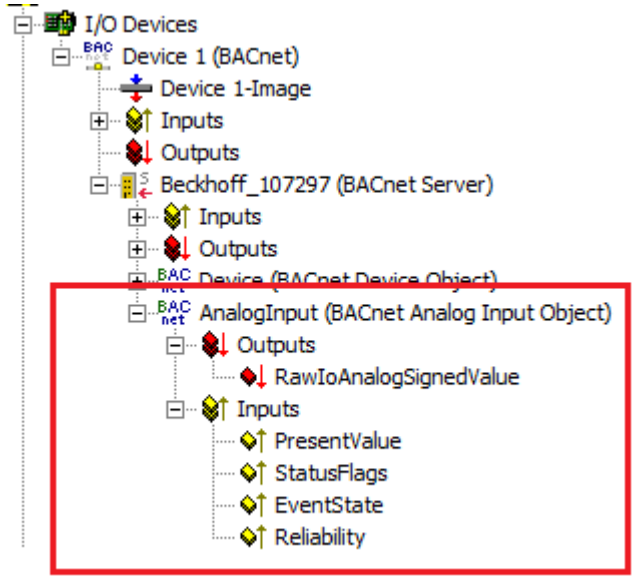


Abb. 138: Bild 1: Beispiel eines BACnet-Objekts unterhalb eines lokalen BACnet-Servers.

VAR_OUTPUT

```
bReady                  : BOOL;
bLogEnable              : BOOL;
nRecordCount            : UDINT;
nTotalCount             : UDINT;
bError                  : BOOL;
nErrorId                : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (LogEnable, RecordCount...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_Device nicht "Operational" oder die Baustein-Instanz wurde im System Manager nicht richtig verknüpft.

bLogEnable: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Log_Enable*.

nRecordCount: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Record_Count*.

nTotalCount: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Total_Record_Count*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Server ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_Device)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

```
Device       : FB_BACnet_Device;
```

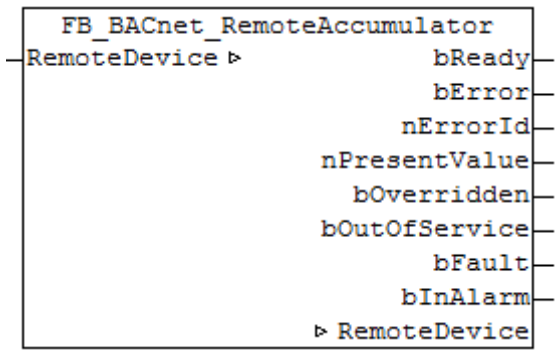
Device: Angabe der Instanz des lokalen, zugehörigen BACnet-Server Bausteins. Pro BACnet-Adapter ist ein BACnet-Server möglich. Siehe FB_BACnet_Adapter und FB_BACnet_Device für weitere Informationen.

5.4 BACnet Client Objects

5.4.1 FB_BACnet_RemoteAccumulator

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteAccumulator" kann lesend auf ein entferntes BACnet-Objekt vom Typ *Accumulator* (AC) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

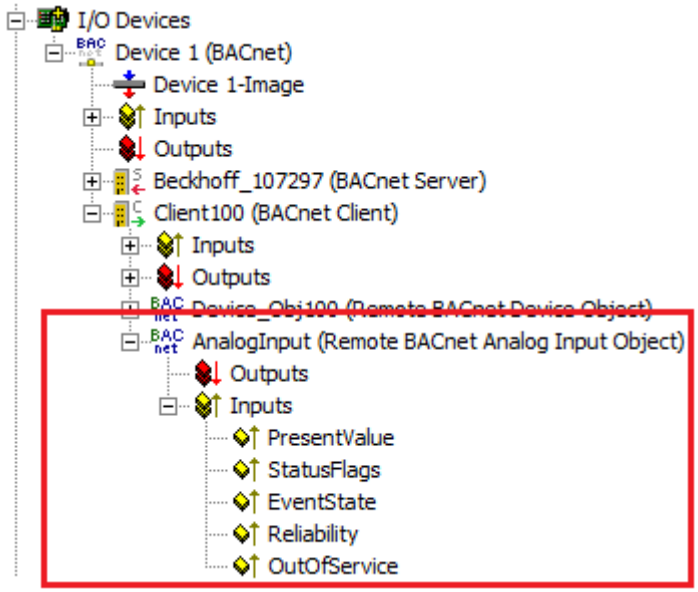


Abb. 139: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_OUTPUT

```
bReady           : BOOL;
nPresentValue    : UDINT;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Present_Value*).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Accumulator* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

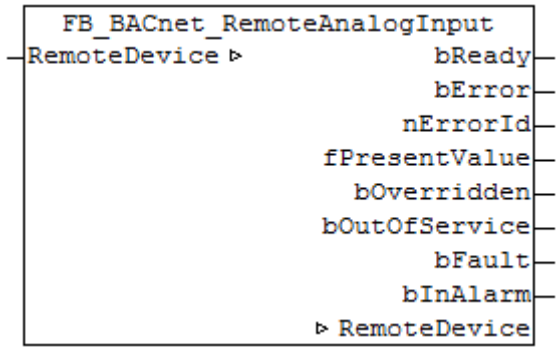
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

5.4.2 FB_BACnet_RemoteAnalogInput

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteAnalogInput" kann lesend auf ein entferntes BACnet-Objekt vom Typ *AnalogInput* (AI) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

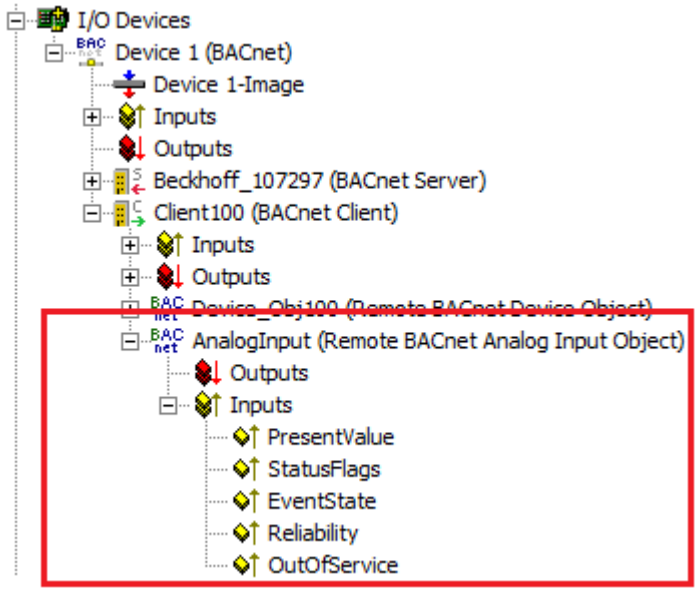


Abb. 140: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_OUTPUT

```
bReady       : BOOL;
fPresentValue    : REAL;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Present_Value*).

bOverriden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogInput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_???*.**nERR_xxx**).

VAR_IN_OUT

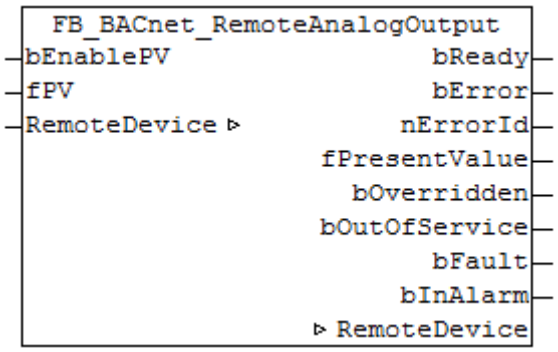
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter und FB_BACnet_RemoteDevice für weitere Informationen.

5.4.3 FB_BACnet_RemoteAnalogOutput

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteAnalogOutput" kann lesend und schreibend auf ein entferntes BACnet-Objekt vom Typ *AnalogOutput* (AO) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

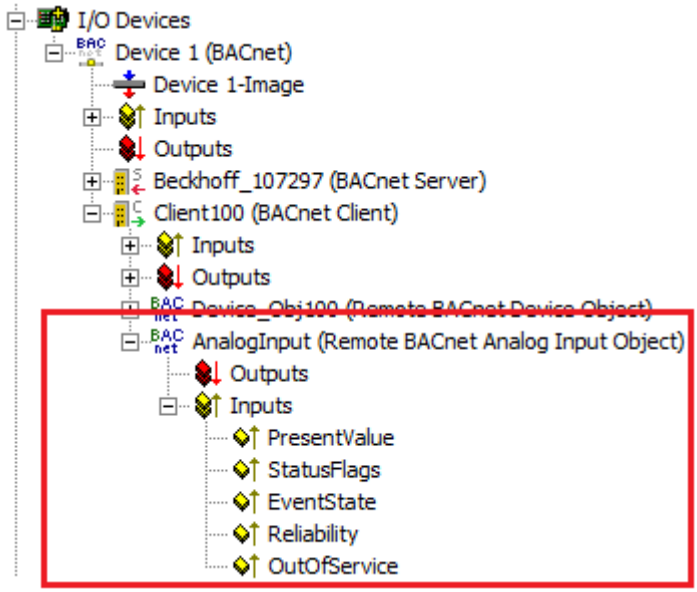


Abb. 141: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_INPUT

```
bEnablePV    : BOOL;
fPV          : REAL;
```

bEnablePV: Gibt den Wert des Eingangs "fPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs "fPV". Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).

fPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und 3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben der Prozessdaten an den lokalen Client erfolgt immer zyklisch. Nach Freigabe mit Hilfe von bEnablePV erfolgt das Versenden der Daten via BACnet an den entfernten Server entweder zyklisch (Periodic Write) oder bei Änderung (Write on Change - empfohlen).

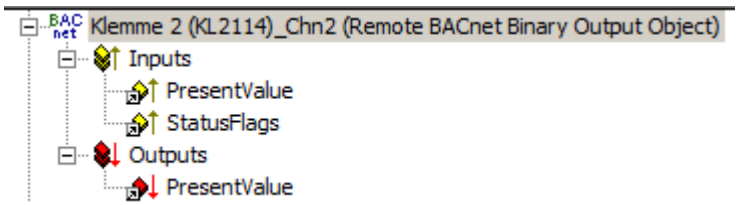


Abb. 142: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

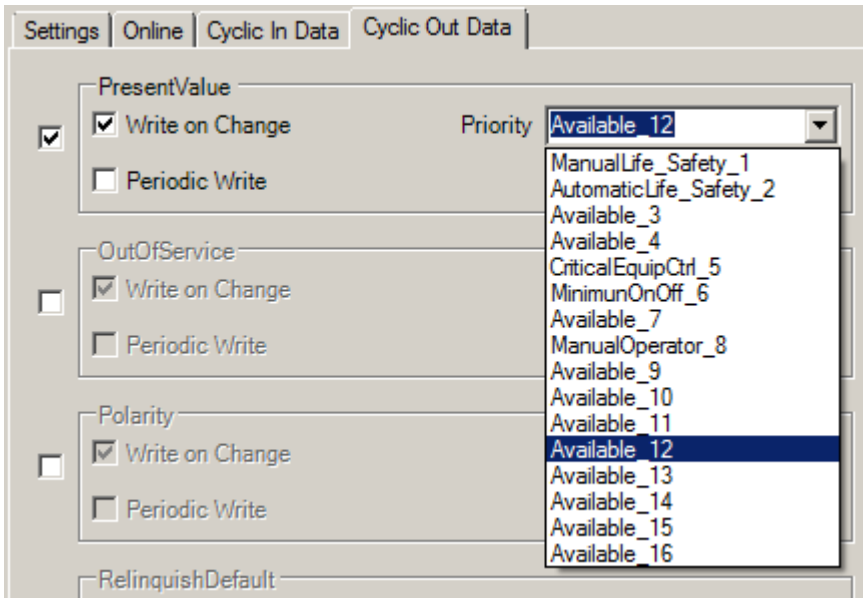


Abb. 143: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 wird als mappbares Prozessdatum angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady          : BOOL;
fPresentValue   : REAL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_RemoteDevice nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogOutput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des

FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

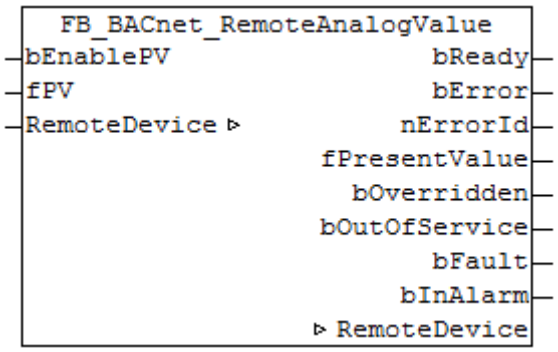
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter und FB_BACnet_RemoteDevice für weitere Informationen.

5.4.4 FB_BACnet_RemoteAnalogValue

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteAnalogValue" kann lesend und schreibend auf ein entferntes BACnet-Objekt vom Typ *AnalogValue* (AV) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

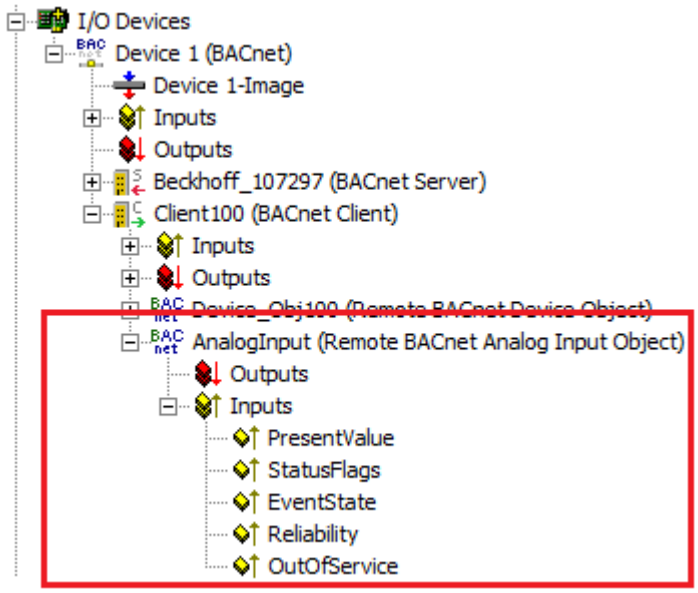


Abb. 144: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_INPUT

```
bEnablePV    : BOOL;
fPV          : REAL;
```

bEnablePV: Gibt den Wert des Eingangs **fPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs **fPV**. Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).
Wird bEnablePV auf *FALSE* gesetzt, dann wird der zugehörige Eintrag aus dem *Priority_Array* wieder entfernt (NULL Schreiben).

fPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und 3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs **bEnablePV** freigegeben. Das Schreiben der Prozessdaten an den lokalen Client erfolgt immer zyklisch. Nach Freigabe mit Hilfe von bEnablePV erfolgt das Versenden der Daten via BACnet an den entfernten Server entweder zyklisch (Periodic Write) oder bei Änderung (Write on Change - empfohlen).

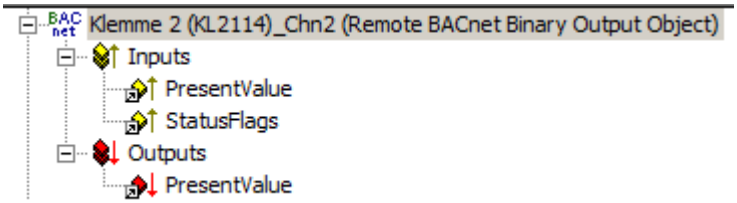


Abb. 145: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

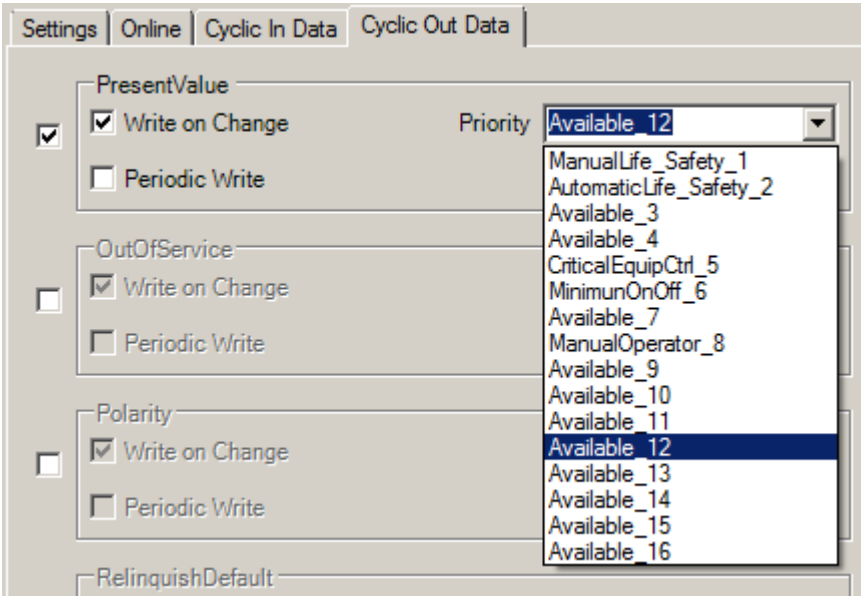


Abb. 146: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 wird als mappbares Prozessdatum angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady          : BOOL;
fPresentValue   : REAL;
bOverridden     : BOOL;
bOutOfService   : BOOL;
bFault          : BOOL;
bInAlarm        : BOOL;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

fPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogValue* und Property *Present_Value*).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *AnalogValue* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

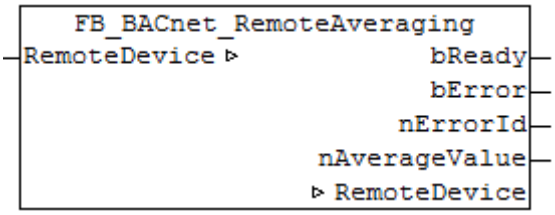
```
RemoteDevice    : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter und FB_BACnet_RemoteDevice für weitere Informationen.

5.4.5 FB_BACnet_RemoteAveraging

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteAveraging" kann lesend auf ein entferntes BACnet-Objekt vom Typ *Averaging* (AVG) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

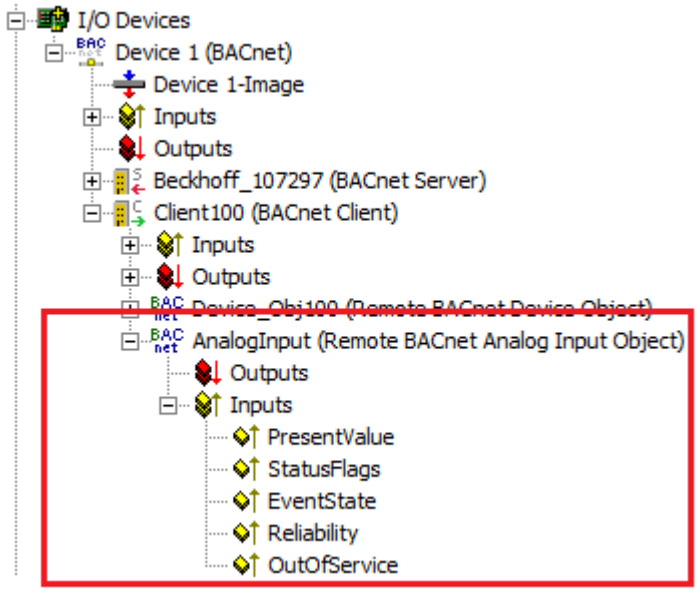


Abb. 147: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_OUTPUT

```
bReady           : BOOL;
nAverageValue    : UDINT;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
```

```
bInAlarm          : BOOL;
bError            : BOOL;
nErrorId          : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

nAverageValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Averaging*.

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Averaging* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

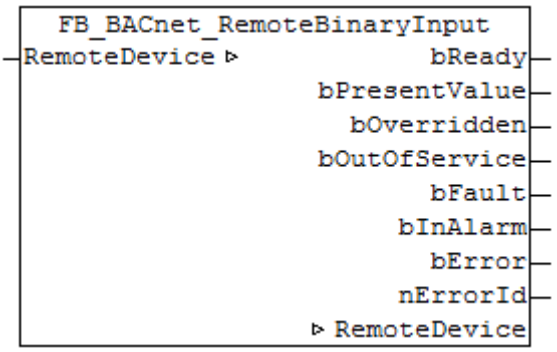
```
RemoteDevice      : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

5.4.6 FB_BACnet_RemoteBinaryInput

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteBinaryInput" kann lesend auf ein entferntes BACnet-Objekt vom Typ *BinaryInput* (BI) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

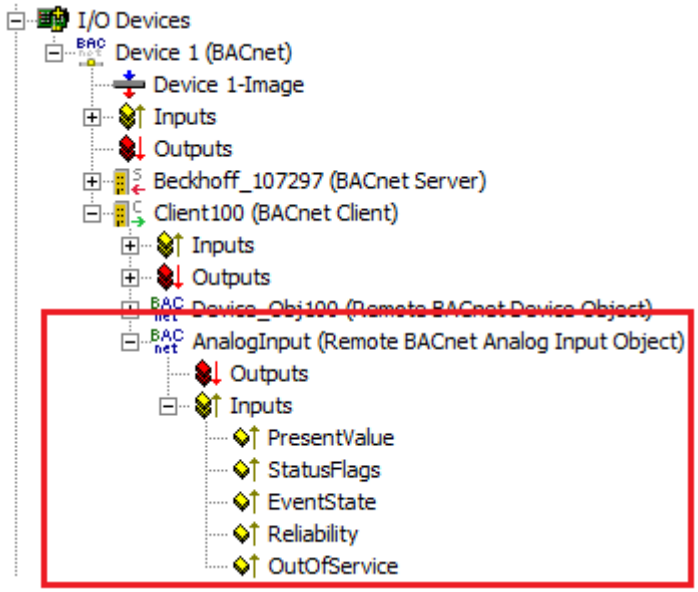


Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_OUTPUT

```
bReady           : BOOL;
bPresentValue    : BOOL;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryInput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

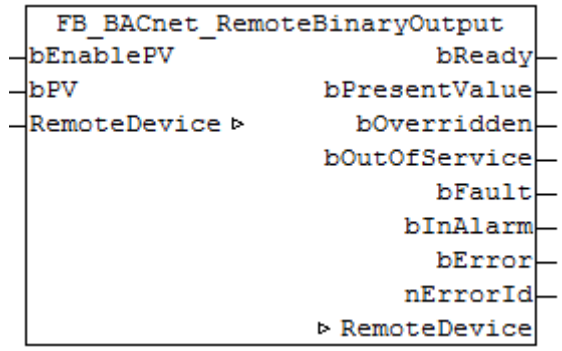
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter und FB_BACnet_RemoteDevice für weitere Informationen.

5.4.7 FB_BACnet_RemoteBinaryOutput

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteBinaryOutput" kann lesend und schreibend auf ein entferntes BACnet-Objekt vom Typ *BinaryOutput* (BO) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

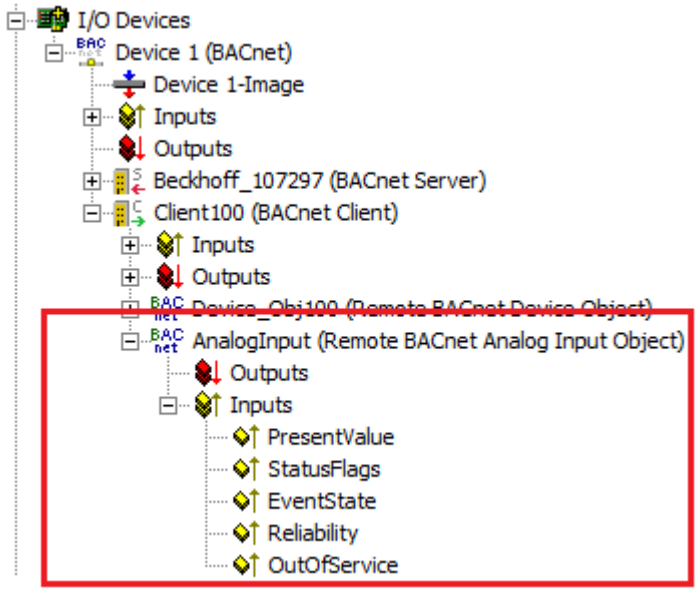


Abb. 148: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_INPUT

```
bEnablePV    : BOOL;
bPV          : E_BACnetBinaryPV;
```

bEnablePV: Gibt den Wert des Eingangs "bPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs "bPV". Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).
Wird bEnablePV auf *FALSE* gesetzt, dann wird der zugehörige Eintrag aus dem *Priority_Array* wieder entfernt (NULL Schreiben).

bPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und 3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben der Prozessdaten an den lokalen Client erfolgt immer zyklisch. Nach Freigabe mit Hilfe von bEnablePV erfolgt das Versenden der Daten via BACnet an den entfernten Server entweder zyklisch (Periodic Write) oder bei Änderung (Write on Change - empfohlen).

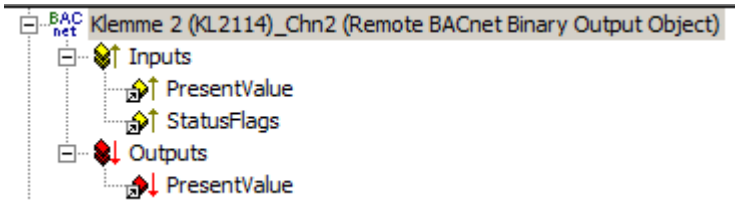


Abb. 149: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

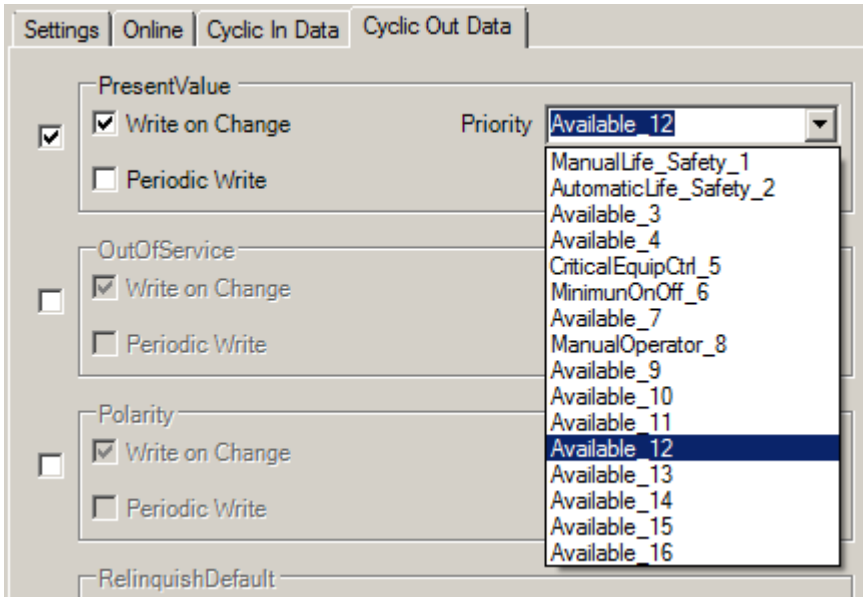


Abb. 150: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 wird als mappbares Prozessdatum angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady           : BOOL;
bPresentValue    : BOOL;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

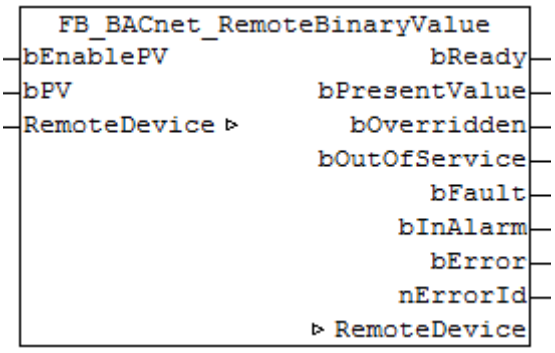
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter und FB_BACnet_RemoteDevice für weitere Informationen.

5.4.8 FB_BACnet_RemoteBinaryValue

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteBinaryValue" kann lesend und schreibend auf ein entferntes BACnet-Objekt vom Typ *BinaryValue* (BV) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

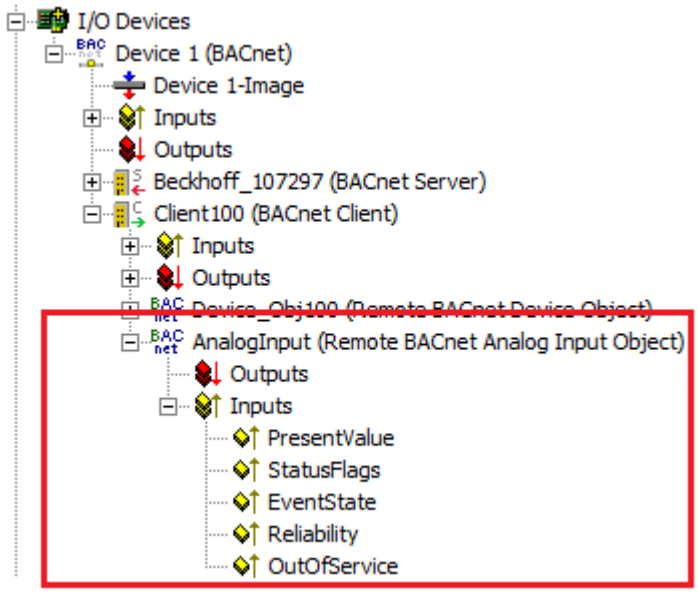


Abb. 151: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_INPUT

```
bEnablePV    : BOOL;
bPV          : E_BACnetBinaryPV;
```

bEnablePV: Gibt den Wert des Eingangs "bPV" frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs "bPV". Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).
Wird bEnablePV auf *FALSE* gesetzt, dann wird der zugehörige Eintrag aus dem *Priority_Array* wieder entfernt (NULL Schreiben).

bPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und 3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs "bEnablePV" freigegeben. Das Schreiben der Prozessdaten an den lokalen Client erfolgt immer zyklisch. Nach Freigabe mit Hilfe von bEnablePV erfolgt das Versenden der Daten via BACnet an den entfernten Server entweder zyklisch (Periodic Write) oder bei Änderung (Write on Change - empfohlen).

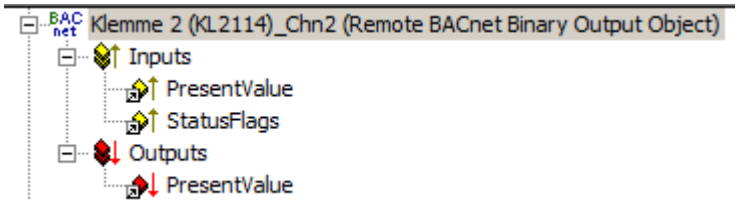


Abb. 152: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value

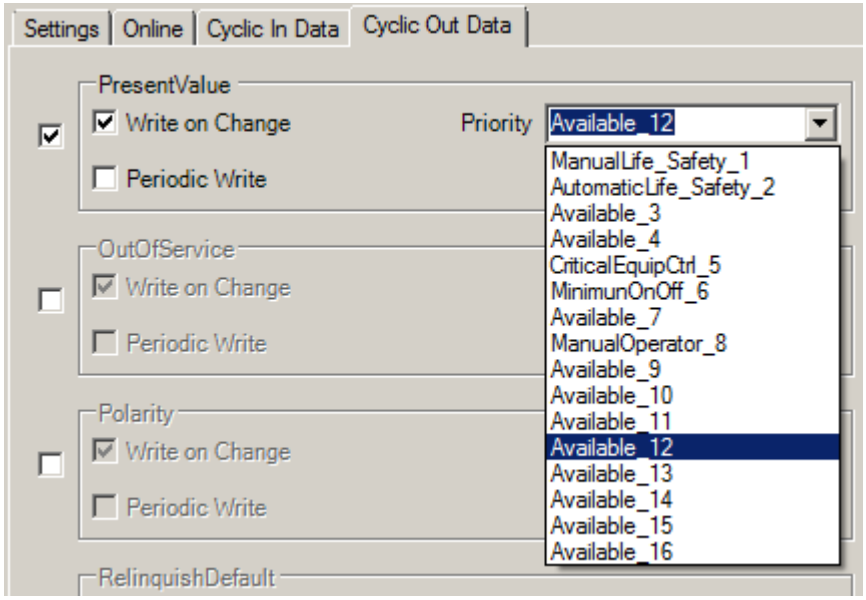


Abb. 153: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 wird als mappbares Prozessdatum angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady           : BOOL;
bPresentValue    : BOOL;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *BinaryValue* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

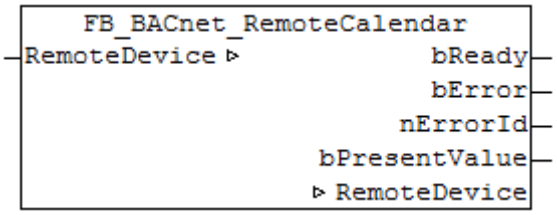
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter und FB_BACnet_RemoteDevice für weitere Informationen.

5.4.9 FB_BACnet_RemoteCalendar

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteCalendar" kann lesend auf ein entferntes BACnet-Objekt vom Typ *Calendar* (CAL) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

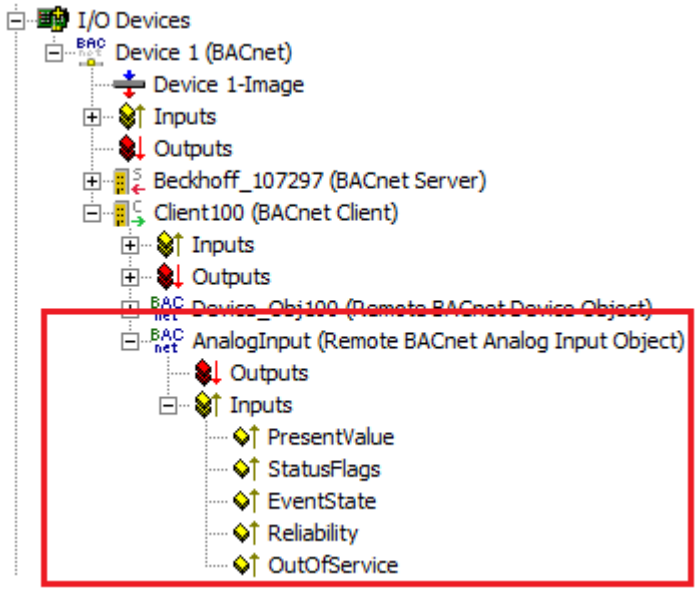


Abb. 154: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_OUTPUT

```
bReady         : BOOL;
bPresentValue  : BOOL;
bError         : BOOL;
nErrorId       : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational".

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Calendar* und Property *Present_Value*).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice [▶ 465])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

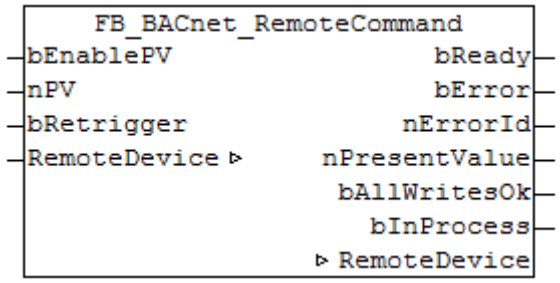
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

5.4.10 FB_BACnet_RemoteCommand

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteCommand" kann lesend und schreibend auf ein entferntes BACnet-Objekt vom Typ *Command* (CMD) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

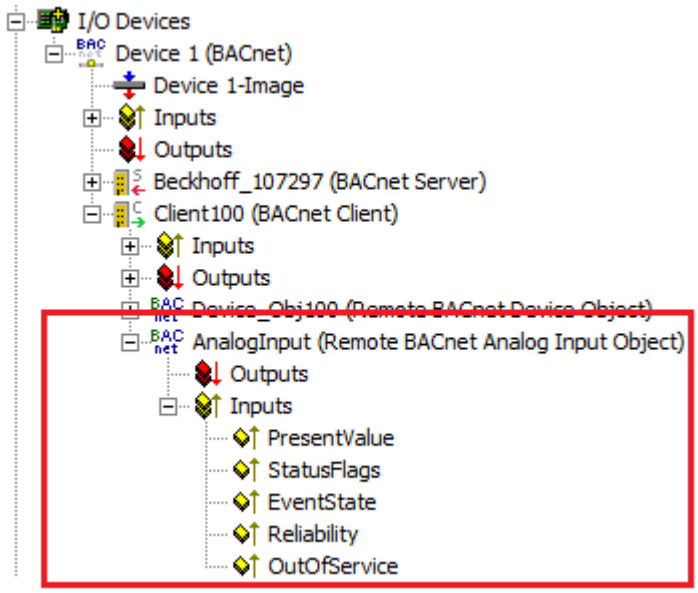


Abb. 155: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_INPUT

```
bEnablePV    : BOOL;
nPV          : UDINT;
bRetrigger   : BOOL;
```

bEnablePV: Gibt den Wert des Eingangs **nPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt das Schreiben in die Property *Present_Value* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs **nPV**.
Wird **bEnablePV** auf *FALSE* gesetzt, dann werden die Prozessdaten des gemappten Property *Present_Value* auf *16#FFFFFFFF* geschrieben und damit deaktiviert.

nPV: Wert der Property *Present_Value* der geschrieben werden soll. Liegt der Wert außerhalb des Wertebereicht (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *Present_Value*), dann wird das zugehörige Prozessdatum deaktiviert, d.h. das Schreiben auf die Property ist deaktivert und andere BACnet-Dienste können schreibend auf die Property *Present_Value* zugreifen.
Das Schreiben eines gültigen Wertes löst die Ausführung der zugehörigen Kommando-Liste des BACnet-Objekts aus. Geschrieben wird der Wert der Property Present_Value immer dann, wenn eine Änderung des Prozessdatums erfolgt (d.h. Änderung des Wertes von **nPV** bei gesetztem **bEnablePV** oder Signalwechsel *FALSE* --> *TRUE* am Eingang **bRetrigger** bei gesetztem **bEnablePV**).

bRetrigger: Bei steigender Flanke an diesem Eingang wird das Schreiben und damit die Ausführung des entsprechenden Kommandos des BACnet-Objekts *Command* wiederholt. Der Signalwechsel von *FALSE* --> *TRUE* entspricht einer Änderung des Prozessdatums der Property *Present_Value* von x --> 0 --> x.

VAR_OUTPUT

```
bReady           : BOOL;
nPresentValue    : UDINT;
bAllWritesOk     : BOOL;
bInProcess       : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, AllWritesOk, InProcess). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational".

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *Present_Value*).

bAllWritesOk: Die zuletzt angeforderte Kommando-Liste wurde erfolgreich abgearbeitet (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *All_Writes_Successful*).

bInProcess: Die selektierte Kommando-Liste wird abgearbeitet (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Command* und Property *In_Process*).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice [▶ 465])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

5.4.11 FB_BACnet_RemoteDevice

Der folgende Funktionsbaustein wird für die Anbindung des SPS-Programmes an ein entferntes BACnet-Device Objekt (lokaler Client) verwendet. Die Verknüpfung des Funktionsbausteins erfolgt mit Hilfe von Prozessdaten.

Die Prozessdaten können manuell oder mittels PLC-Automapping [▶ 64] automatisch verknüpft werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten. Zu den Prozessdaten des BACnet-Device Objekts sind ebenfalls die nötigen Prozessdaten für die Anbindung des BACnet Clients enthalten.

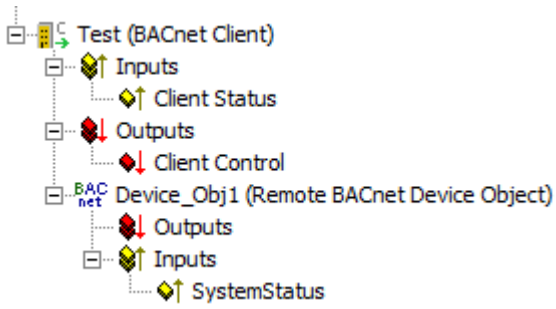


Abb. 156: Bild-1: Prozessdaten des entfernten BACnet Device Objekts und -Client im System Manager.

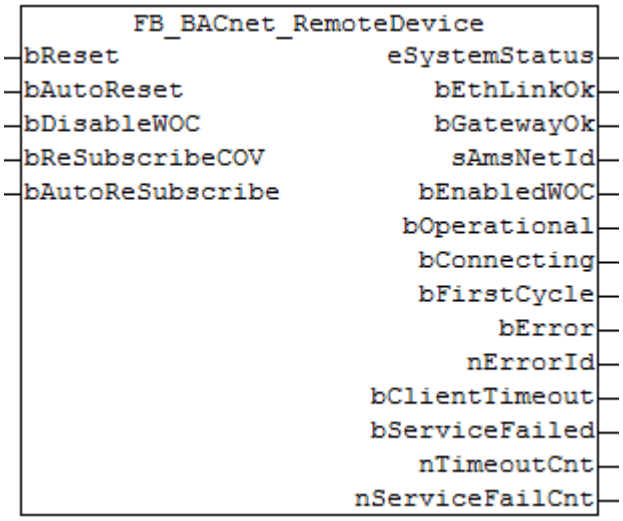


Abb. 157: Bild-2: Funktionsbaustein des entfernten BACnet Device Objekts und -Client im SPS-Programm.

Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteDevice" wird der Zustand des entfernten BACnet Device Objects gelesen (*System_Status*) und im SPS-Programm ausgegeben. Zudem wird mit Hilfe des Prozessdatums "Client Control" und "Client Status" der lokale BACnet Client gesteuert (WriteOnChange sperren/freigeben, Neu-Abonnieren von Properties, automatisches Abonnieren von Properties nach Verbindungsfehlern und Fehlerquittierung).

Für die Verwendung des Bausteins "FB_BACnet_RemoteDevice" wird zudem eine globale Instanz des Bausteins FB_BACnet_Adapter [▶ 378] pro SPS-Projekt benötigt. Der Baustein FB_BACnet_Adapter [▶ 378] stellt die Verbindung zwischen SPS und BACnet-Adapter im System Manager her. Diese globale Instanz wird bereits von der SPS-Bibliothek bereit gestellt. Im Folgenden ist die Hierarchie zwischen FB_BACnet_Adapter [▶ 378], FB_BACnet_Device [▶ 419] bzw. "FB_BACnet_RemoteDevice" und einem Objekt vom Typ *AnalogValue* dargestellt:

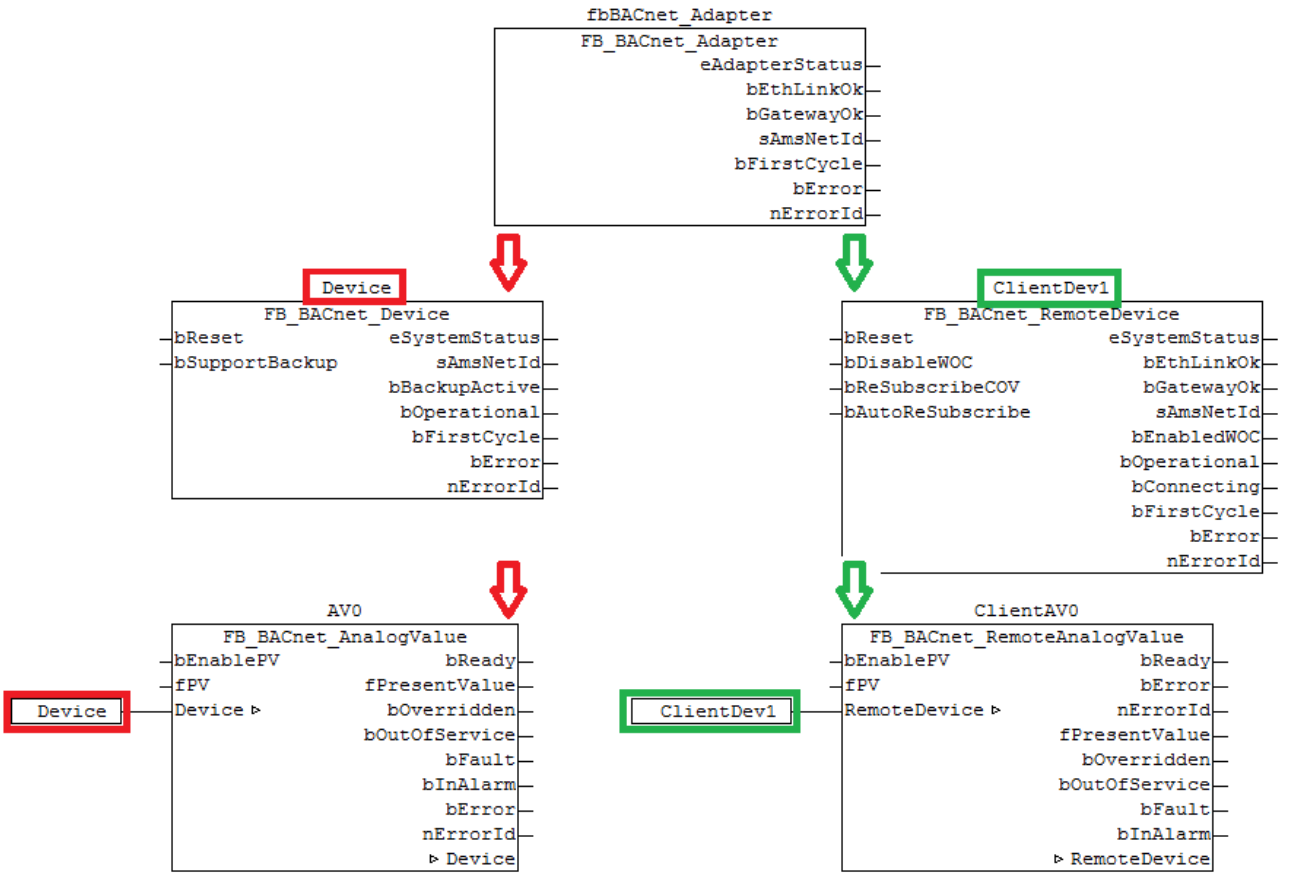


Abb. 158: Bild-3: Beispiel zur Verknüpfung der FB-Instanzen in der SPS.

Informationen zur Verwendung: siehe auch Bausteine FB_BACnet_Adapter [▶ 378] bzw. FB_BACnet_Device [▶ 419].

VAR_INPUT

```
bReset           : BOOL;
bAutoReset       : BOOL;
bDisableWOC      : BOOL;
bReSubscribeCOV  : BOOL;
bAutoReSubscribe : BOOL;
```

bReset: Zurücksetzen des Fehlerzustands bei Signalwechsel *FALSE --> TRUE*.

bAutoReset: Bei Setzen des Eingangs auf *TRUE* werden Fehler mit ID 20 und 21 automatisch quittiert. Verbundene Remote-FBs werden nicht blockiert (**bReady** an den Objekt-FBs bleibt TRUE). Statt des Fehlers wird ein Fehlerzähler hoch gezählt (siehe **nTimeoutCnt** und **nServiceFailCnt**).

bDisableWOC: Bei Setzen des Eingangs auf *TRUE* wird WriteOnChange (WOC, Schreiben-bei-Änderung) für sämtliche Objekte dieses Clients unterdrückt. Das bedeutet, dass Änderungen der Prozessdaten kein automatisches Schreiben der Properties an die entfernten Objekte auslöst (gilt nur für Properties die für WriteOnChange unter dem Client konfiguriert sind).

bReSubscribeCOV: Bei Signalwechsel des Eingangs von *FALSE --> TRUE* wird das Neu-Abonnieren sämtlicher Properties die für COV konfiguriert sind ausgelöst. Diese Funktion betrifft unter Umständen potentiell viele Objekte und sollte daher nur bei Bedarf ausgeführt werden. Im Normalfall werden alle abonnierten Properties nach Ablauf einer Timeoutzeit automatisch erneut abonniert (Resubscription-Intervall).

bAutoReSubscribe: Bei Setzen des Eingangs auf *TRUE* wird das automatische Neu-Abonnieren sämtlicher für COV konfigurierter Properties bei Bedarf ausgelöst. Dies ist nicht mit dem Resubscription-Intervall zu verwechseln! Ein automatisches Neu-Abonnieren wird ausgelöst, wenn z.B. ein entfernter Server nicht mehr erreichbar war und die Verbindung wieder hergestellt werden konnte. Wird der Eingang auf *FALSE* gesetzt, dann wird ein erneutes Abonnieren erst nach Ablauf des Resubscription-Intervalls ausgelöst.

VAR_OUTPUT

```
eSystemStatus    : E_BACnetDeviceStatus;
bEthLinkOk       : BOOL;
bGatewayOk       : BOOL;
sAmsNetId        : T_AmsNetId;
bEnabledWOC      : BOOL;
bOperational     : BOOL;
bConnecting      : BOOL;
bFirstCycle      : BOOL;
bError           : BOOL;
nErrorId         : UINT;
bClientTimeout   : BOOL;
bServiceFailed   : BOOL;
nTimeoutCnt      : UDINT;
nServiceFailCnt  : UDINT;
```

eSystemStatus: Aktueller Status des entfernten BACnet Server Objekts (siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet Device Objekt und Property *System_Status*).
0: BACnetDeviceStatus_Operational (Betriebsbereit)
1: BACnetDeviceStatus_OperationalReadOnly (Nur-Lese-Zugriff auf Properties)
2: BACnetDeviceStatus_DownloadRequired (Konfiguration-Laden erforderlich)
3: BACnetDeviceStatus_DownloadInProgress (Konfiguration-Laden wird ausgeführt)
4: BACnetDeviceStatus_NonOperational (Nicht-Betriebsbereit)
5: BACnetDeviceStatus_BackupInProgress (Datensicherung wird ausgeführt)

bEthLinkOk: Die lokale Ethernet-Verbindung ist aktiv (Kabel gesteckt, Adapter verbunden), wenn der Ausgang auf *TRUE* gesetzt ist.

bGatewayOk: Das konfigurierte Gateway ist erreichbar, wenn der Ausgang auf *TRUE* gesetzt ist.

sAmsNetId: Ausgabe der AMS-NetID des lokalen BACnet-Adapters (kann für den asynchronen Zugriff via ADS auf BACnet-Objekte verwendet werden).

bEnabledWOC: WriteOnChange (WOC, Schreiben-bei-Änderung) ist für COV konfigurierte Properties aktiviert.

bOperational: BACnet-Device Objekt des entfernten Servers meldet "Operational" (System_Status = 0), der Device-Adapter meldet den Status **bEthLinkOk**, Status **bConnecting** ist *FALSE* und das Bit 0 des Prozessdatums "Client Status" ist nicht gesetzt (nERR_CLIENT_TIMEOUT). Fällt der Ausgang auf *FALSE* werden sämtliche verbundene Objekte (Bausteininstanzen) gesperrt.

bConnecting: Die Verbindung zum entfernten Server wird aufgebaut (Prozessdatum "Client Status" ist kleiner 0x04xx).

bFirstCycle: Wird mit dem ersten Aufruf der Bausteininstanz nach SPS-Reset bzw. -Neustart für einen Zyklus gesetzt.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
1 = keine gültige AMS-NetID
3 = keine Netzwerkverbindung (bEthLinkOk = *FALSE*)
4 = Gateway nicht erreichbar (bGatewayOk = *FALSE*)
20 = Timeout bei Kommunikation zum entfernten Server
21 = Fehler bei der Ausführung eines Dienstes zum entfernten Server (Subscribe-COV, Property-Write etc.)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_RemoteDevice*.**nERR_xxx**). Alle Fehler mit Fehlernummer größer oder gleich 20 müssen mittels **bReset** quittiert werden.

bClientTimeout: Bei gesetztem Ausgang liegt ein Timeout-Fehler (ID 20) an. Der Ausgang wird erst zurück gesetzt, wenn 10 Sekunden lang kein Timeout Fehler mehr aufgetreten ist.

bServiceFailed: Bei gesetztem Ausgang liegt ein Service-Fehler (ID 21) an. Der Ausgang wird erst zurück gesetzt, wenn 10 Sekunden lang kein Fehler mehr aufgetreten ist.

nTimeoutCnt: Anzahl aufgetretener Timeout-Fehler (ID 20). Es wird die positive Flanke des entsprechenden Status-Flag des Clients gezählt.

nServiceFailCnt: Anzahl aufgetretener Service-Fehler (ID 21). Es wird die positive Flanke des entsprechenden Status-Flag des Clients gezählt.

Eine Instanz des Bausteins "FB_BACnet_RemoteDevice" ist immer dann nötig, wenn auf Objekte eines entfernten BACnet-Servers mit Hilfe eines lokalen BACnet-Clients aus der SPS zugegriffen werden soll. Mehrere Instanzen des Bausteins "FB_BACnet_RemoteDevice" können problemlos angelegt und mittels SPS-Automapping zu den entsprechenden Clients im System Manager verbunden werden. Alle Instanzen des Bausteins "FB_BACnet_RemoteDevice" verwenden die gleiche, globale Instanz des Bausteins FB_BACnet_Adapter.

Sollen mehrere Instanzen des Bausteins FB_BACnet_Adapter verwendet werden, dann muss vor dem Aufruf der Instanz des BACnet-Client Bausteins zwingend die zugehörige Instanz des BACnet-Adapters Bausteins erfolgen. Im Folgenden ein CFC-Beispiel (Wichtig ist die Aufrufreihenfolge der Bausteine zu beachten!):

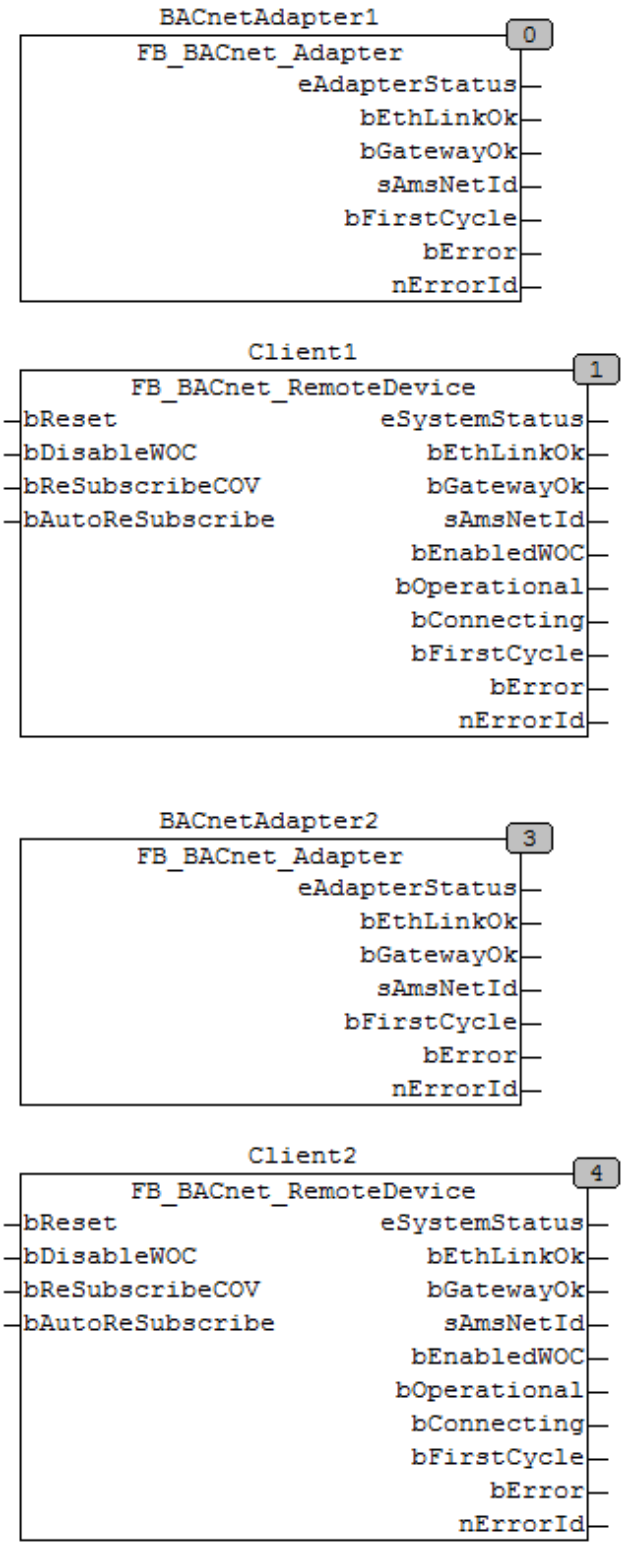


Abb. 159: Bild-4: Beispiel für das Verwenden von mehreren Adapter- und Client-Instanzen in einem SPS-Projekt.

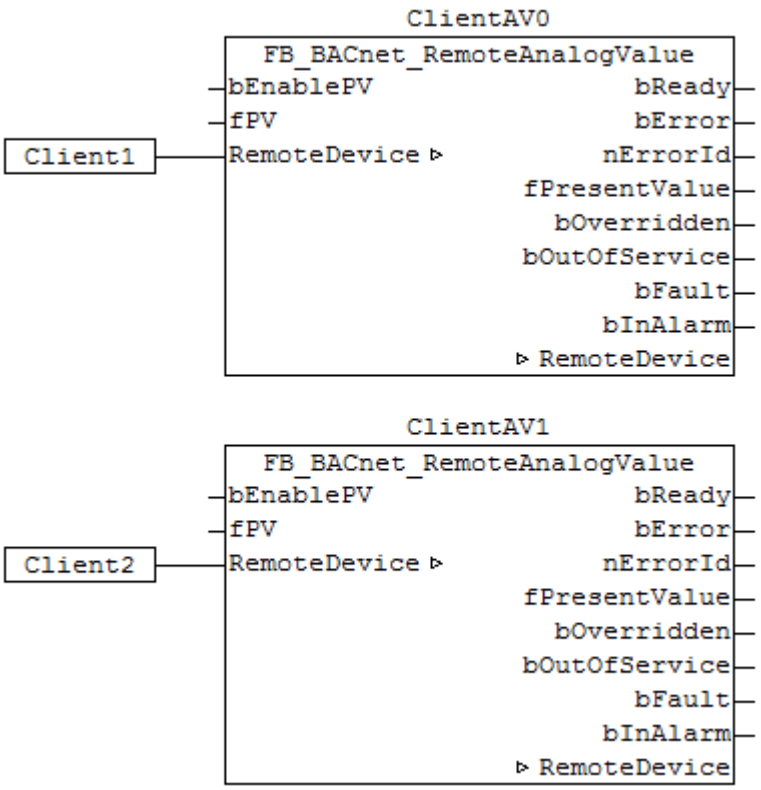


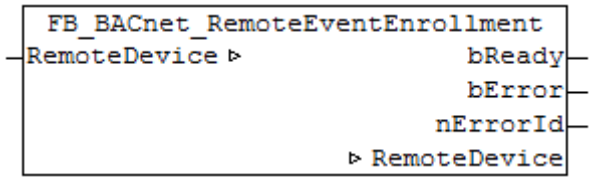
Abb. 160: Bild-5: Beispiel für den Aufruf von Bausteinen vom Typ FB_BACnet_RemoteAnalogValue mit unterschiedlichen entfernten BACnet-Servern in einem SPS-Projekt.

Die Verwendung von mehreren BACnet-Adaptern pro SPS-Projekt ist ein Spezialfall und wird nicht empfohlen. Sinnvoll wäre (z.B. bei Verwendung mehrerer Netzwerkkarten und damit getrennte BACnet-Netzwerke) die Verwendung mehrerer SPS-Projekte und damit mehrere SPS-Laufzeiten. Der Datenaustausch zwischen den einzelnen Laufzeiten kann via Prozessdatenmapping oder ADS-Kommunikation realisiert werden. Somit wäre auch die Verwendung des SPS-Automappings für BACnet-Objekte möglich.

5.4.12 FB_BACnet_RemoteEventEnrollment

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Der Funktionsbausteins "FB_BACnet_RemoteEventEnrollment" dient als Platzhalter für zukünftige Funktionalitäten.

VAR_OUTPUT

```
bReady       : BOOL;
bError       : BOOL;
nErrorId     : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_RemoteDevice [▶ 465] nicht "Operational".

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice [▶ 465])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

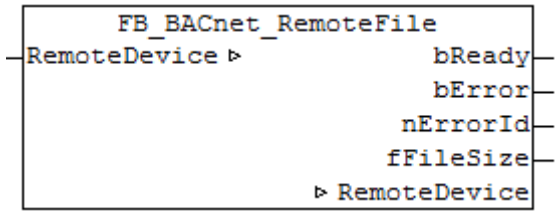
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

5.4.13 FB_BACnet_RemoteFile

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteFile" kann lesend auf ein entferntes BACnet-Objekt vom Typ *File* (FILE) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

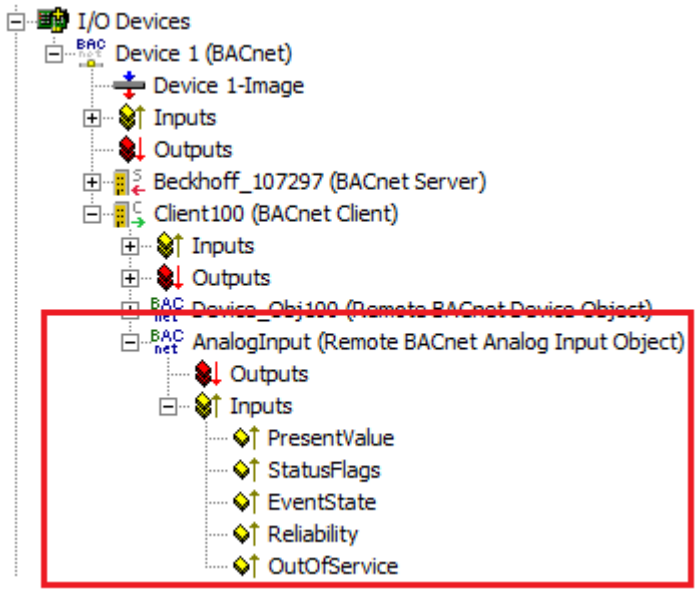


Abb. 161: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_OUTPUT

```
bReady          : BOOL;
bPresentValue   : BOOL;
bError          : BOOL;
nErrorId        : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational".

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *File* und Property *Present_Value*).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice [▶ 465])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

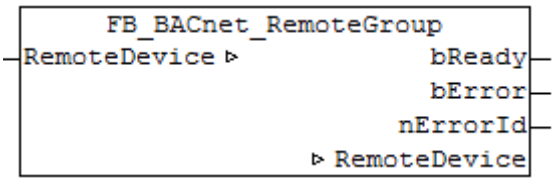
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

5.4.14 FB_BACnet_RemoteGroup

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Der Funktionsbausteins "FB_BACnet_RemoteGroup" dient als Platzhalter für zukünftige Funktionalitäten.

VAR_OUTPUT

```
bReady       : BOOL;
bError       : BOOL;
nErrorId     : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_RemoteDevice [▶ 465] nicht "Operational".

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice [▶ 465])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

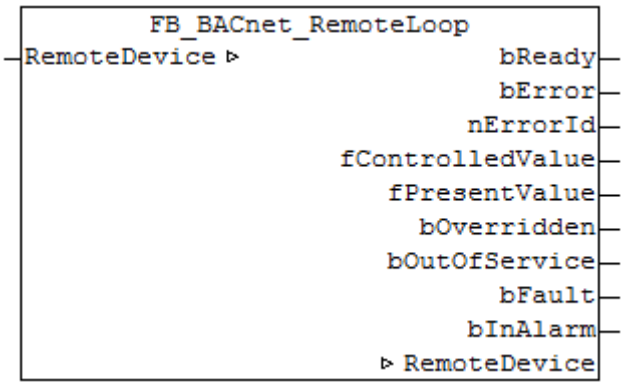
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

5.4.15 FB_BACnet_RemoteLoop

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteLoop" kann lesend auf ein entferntes BACnet-Objekt vom Typ *Loop* (LOOP) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

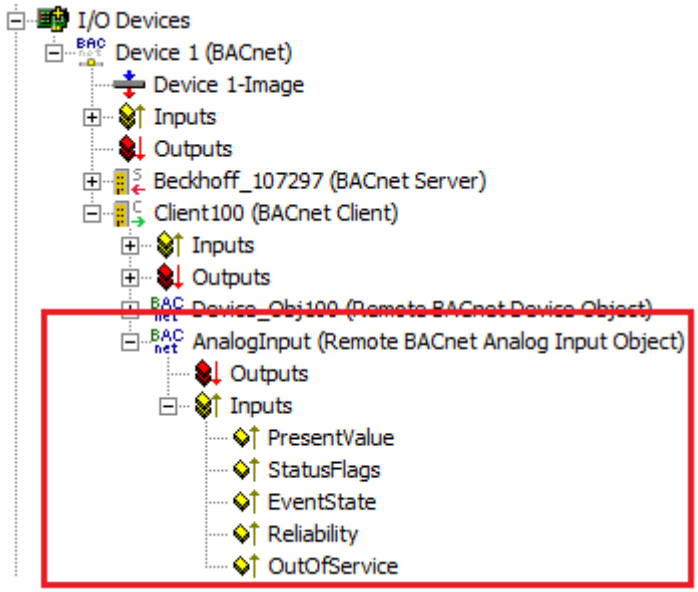


Abb. 162: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_OUTPUT

Variablen sind nur in der erweiterten Version des Bausteins enthalten. Die sogenannten extended Versionen der Bausteine enden mit "_EX". Diese bieten wesentlich mehr Prozessdaten, was jedoch in Projekten mit sehr vielen Objekten zu Leistungseinbußen führen kann.

```
bReady           : BOOL;
fControlledValue : REAL;
fPresentValue    : REAL;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

fControlledValue: Rückmeldung der aktuellen Prozessgröße (X, Istwert).

fPresentValue: Rückmeldung der aktuellen Regelausgabe (Y, Stellwert). Achtung: *Present_Value* und *Controlled_Variable_Value* können schnell zu Verwechslungen führen (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Properties *Present_Value*, *Controlled_Variable_Value* und *Controlled_Variable_Reference*)!

fPropBand: Rückmeldung der aktuellen Regelausgabe in Prozent (-100%...+100%) in Relation zur minimalen und maximalen Regelausgabe (Properties *Minimum_Output* und *Maximum_Output*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

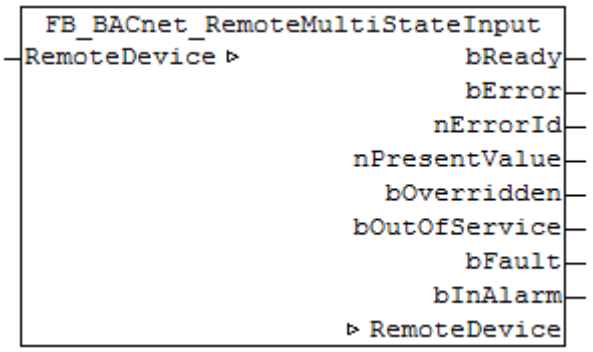
Konfiguration des Reglers

Die Konfiguration des Reglers erfolgt mit Hilfe der BACnet-Properties: *Action*, *Proportional_Constant* (P-Faktor), *Integral_Constant* (I-Faktor), *Derivative_Constant* (D-Faktor), Bias (Ausgabe-Offset), *Maximum_Output* (Maximale Regelausgabe) und *Minimum_Output* (Minimale Regelausgabe). Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Loop*.

5.4.16 FB_BACnet_RemoteMultiStateInput

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteMultiStateInput" kann lesend auf ein entferntes BACnet-Objekt vom Typ *MultiStateInput* (MI) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

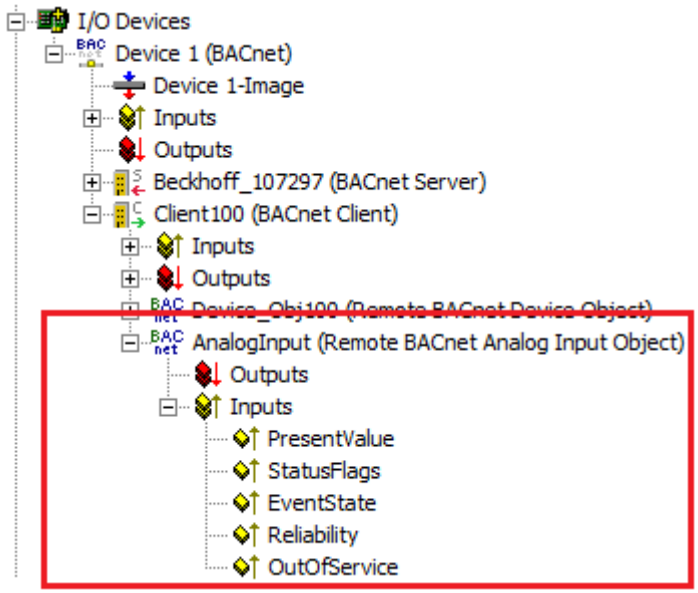


Abb. 163: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_OUTPUT

```
bReady           : BOOL;
nPresentValue    : UDINT;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateInput* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

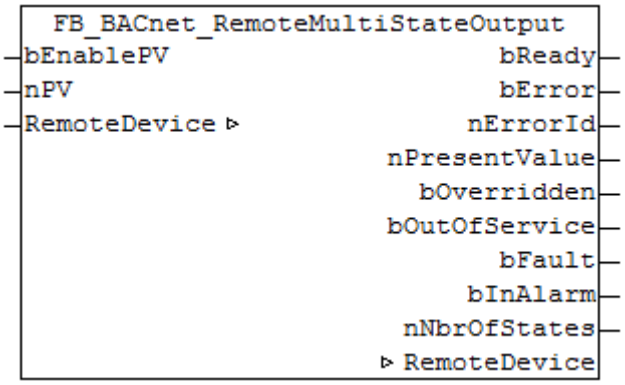
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter und FB_BACnet_RemoteDevice für weitere Informationen.

5.4.17 FB_BACnet_RemoteMultiStateOutput

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteMultiStateOutput" kann lesend und schreibend auf ein entferntes BACnet-Objekt vom Typ *MultiStateOutput* (MO) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

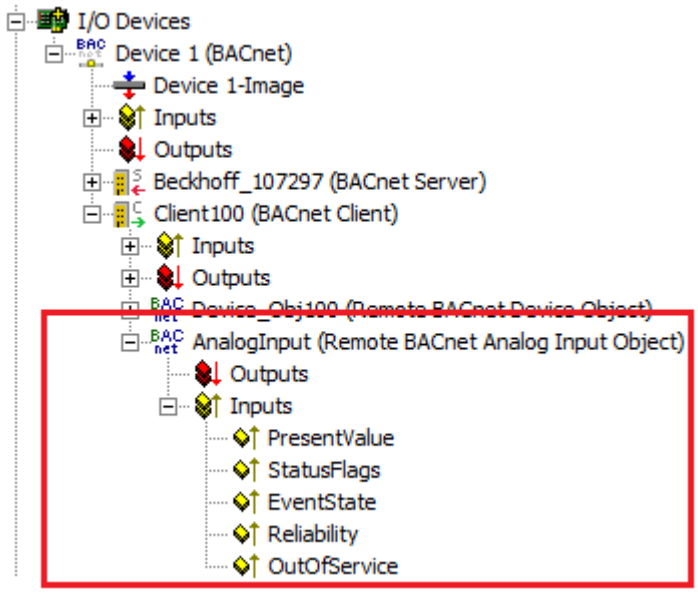


Abb. 164: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_INPUT

```
bEnablePV    : BOOL;
nPV          : UDINT;
```

bEnablePV: Gibt den Wert des Eingangs **nPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs **nPV**. Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager).
Wird bEnablePV auf FALSE gesetzt, dann wird der zugehörige Eintrag aus dem *Priority_Array* wieder entfernt (NULL Schreiben).

nPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und -3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs **bEnablePV** freigegeben. Das Schreiben der Prozessdaten an den lokalen Client erfolgt immer zyklisch. Nach Freigabe mit Hilfe von bEnablePV erfolgt das Versenden der Daten via BACnet an den entfernten Server entweder zyklisch (Periodic Write) oder bei Änderung (Write on Change - empfohlen).

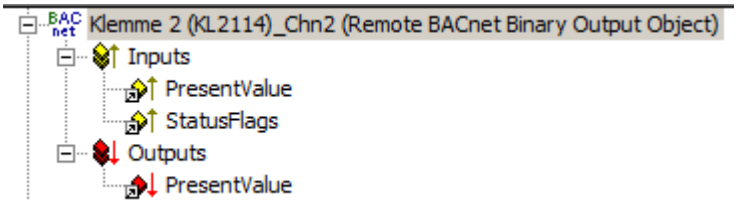


Abb. 165: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

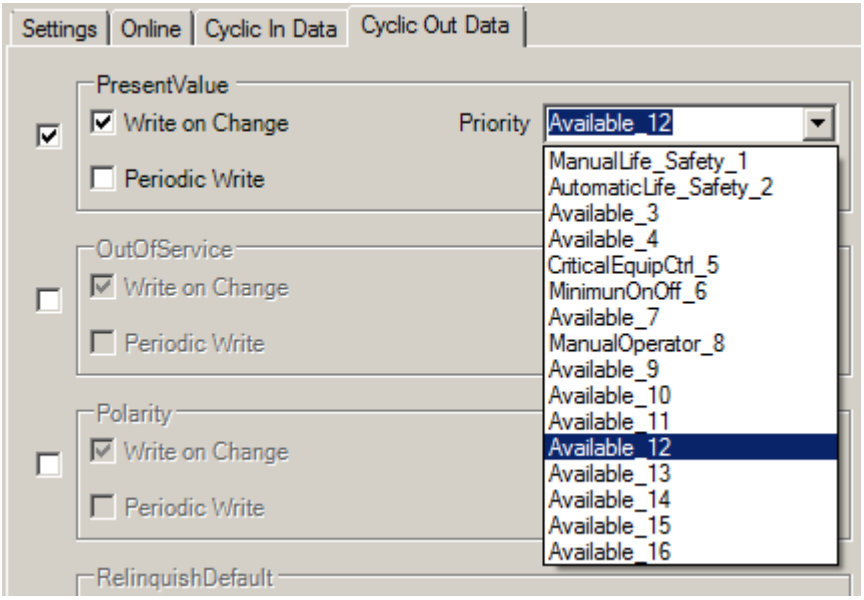


Abb. 166: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 wird als mappbares Prozessdatum angelegt (siehe Ergebnis in Bild-2).

VAR_OUTPUT

```
bReady           : BOOL;
nPresentValue    : UDINT;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
nNbrOfStates     : UDINT;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_RemoteDevice nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateOutput* und Property *Status_Flags*.

nNbrOfStates: Meldet die verfügbare Anzahl von Werten, die das *Present_Value* des MultiStateOutput-Objekts annehmen kann (1...*nNbrOfStates*). Ist **nNbrOfStates** gleich 0, dann gibt es keinen gültigen "State" den das Objekt annehmen kann (*Present_Value* ist 0).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

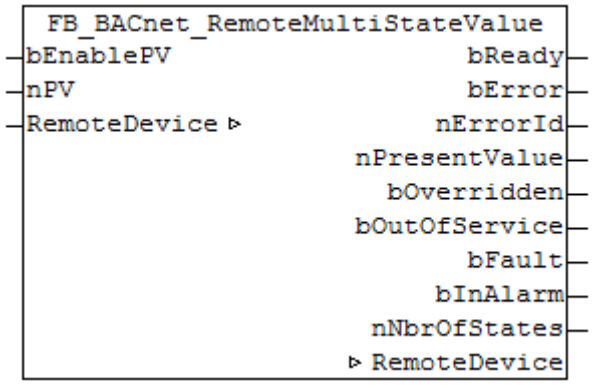
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter und FB_BACnet_RemoteDevice für weitere Informationen.

5.4.18 FB_BACnet_RemoteMultiStateValue

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteMultiStateValue" kann lesend und schreibend auf ein entferntes BACnet-Objekt vom Typ *MultiStateValue* (MV) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

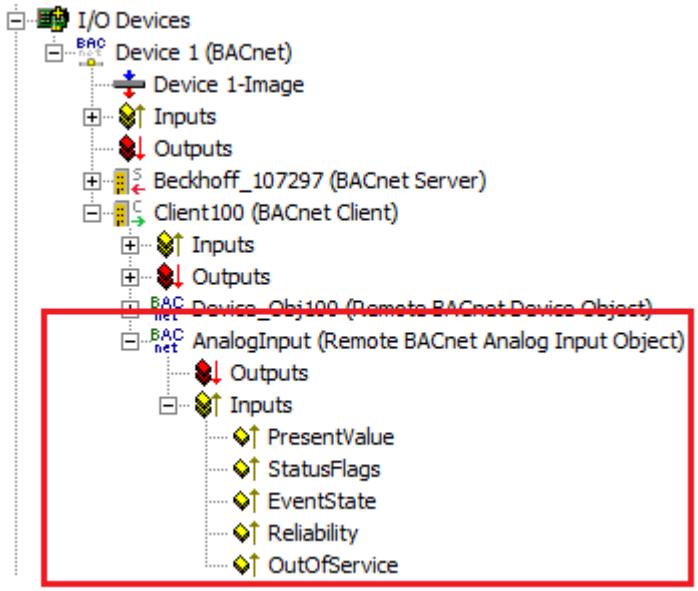


Abb. 167: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_INPUT

```
bEnablePV    : BOOL;
nPV          : UDINT;
```

bEnablePV: Gibt den Wert des Eingangs **nPV** frei. Sobald der Eingang auf *TRUE* gesetzt wird, erfolgt ein Eintrag im *Priority_Array* des zugehörigen BACnet-Objekts mit dem Wert des Eingangs **nPV**. Dies entspricht einem Schreibzugriff auf das kommandierbare Property *Present_Value* mit eingestellter Priorität (Default: 12 bei Verwendung des PLC-Automappings bzw. 16 bei manueller Verknüpfung im System Manager). Wird bEnablePV auf *FALSE* gesetzt, dann wird der zugehörige Eintrag aus dem *Priority_Array* wieder entfernt (NULL Schreiben).

nPV: Wert der in das *Priority_Array* der kommandierbaren Property *Present_Value* geschrieben werden soll. Die Priorität ergibt sich aus dem Mapping und Konfiguration der Prozessdaten des BACnet-Objekts im System Manager (siehe auch Bild-2 und -3). Das Schreiben wird mit *TRUE*-Setzen des Eingangs **bEnablePV** freigegeben. Das Schreiben der Prozessdaten an den lokalen Client erfolgt immer zyklisch. Nach Freigabe mit Hilfe von bEnablePV erfolgt das Versenden der Daten via BACnet an den entfernten Server entweder zyklisch (Periodic Write) oder bei Änderung (Write on Change - empfohlen).

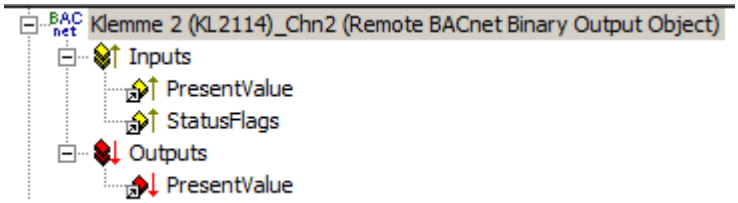


Abb. 168: Bild-2: Beispiel eines BACnet-Objekts mit Prozessdaten für das Schreiben der kommandierbaren Property Present_Value.

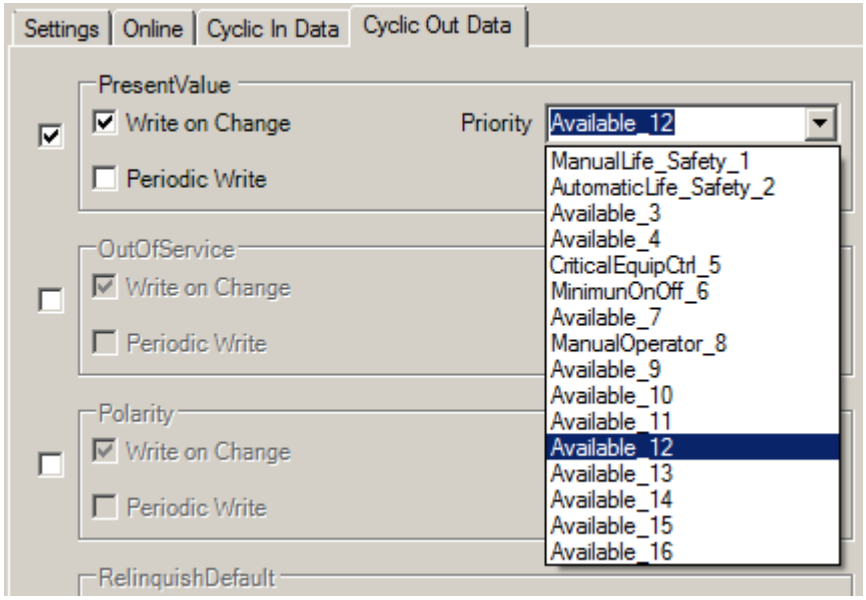


Abb. 169: Bild-3: Beispiel der Prozessdatenkonfiguration eines BACnet-Objektes im System Manager. Priorität 12 wird als mappbares Prozessdatum angelegt (siehe Ergebnis in Bild 2).

VAR_OUTPUT

```
bReady           : BOOL;
nPresentValue    : UDINT;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
nNbrOfStates     : UDINT;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

nPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Present_Value*).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *MultiStateValue* und Property *Status_Flags*.

nNbrOfStates: Meldet die verfügbare Anzahl von Werten, die das *Present_Value* des MultiStateValue-Objekts annehmen kann (1...*nNbrOfStates*). Ist "nNbrOfStates" gleich 0, dann gibt es keinen gültigen "State" den das Objekt annehmen kann (*Present_Value* ist 0).

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

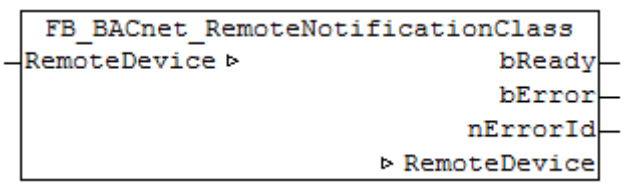
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter und FB_BACnet_RemoteDevice für weitere Informationen.

5.4.19 FB_BACnet_RemoteNotificationClass

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Der Funktionsbausteins "FB_BACnet_RemoteNotificationClass" dient als Platzhalter für zukünftige Funktionalitäten.

VAR_OUTPUT

```
bReady       : BOOL;
bError       : BOOL;
nErrorId     : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_RemoteDevice [▶ 465] nicht "Operational".

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice [▶ 465])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

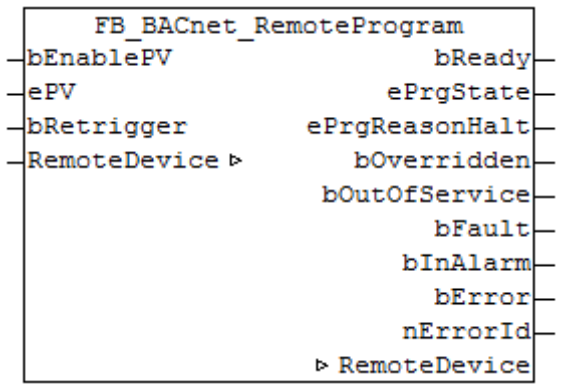
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

5.4.20 FB_BACnet_RemoteProgram

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteProgram" kann lesend und schreibend auf ein entferntes BACnet-Objekt vom Typ *Program* (PROG) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

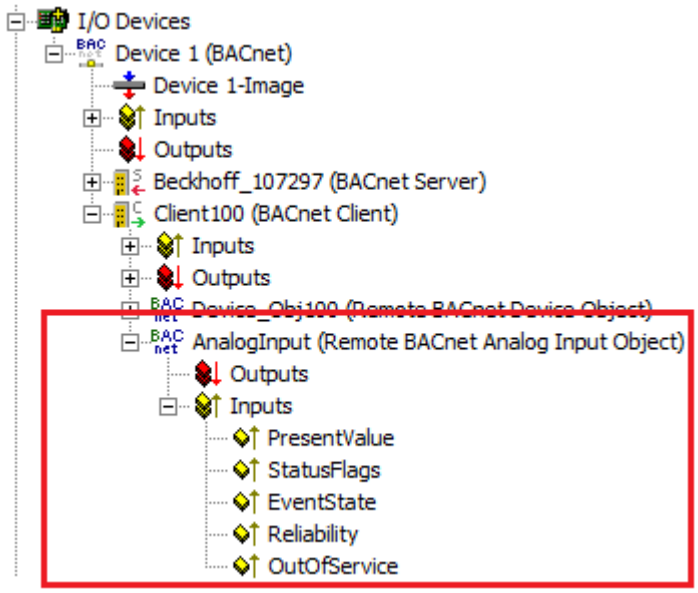


Abb. 170: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_INPUT

```
bEnablePV       : BOOL;
ePV             : E_BACnetProgramRequest;
bRetrigger      : BOOL; (* execute selected request on raising edge again *)
```

bEnablePV: Gibt den Wert des Eingangs **ePV** frei. Sobald der Eingang auf *TRUE* gesetzt wird, wird der Wert von **ePV** in die Property *Program_Change* des Objekts geschrieben. Wird der Eingang auf *FALSE* gesetzt, dann wird *0xFFFF* in das Prozessdatum der Property *Program_Change* des Objekts geschrieben. Der Wert *0xFFFF* verhindert das Schreiben an den entfernten Server und damit ebenfalls das Schreiben an das entfernte Objekt.

ePV: Anforderung an das entfernte Programm-Objekt. Wurde **bEnablePV** auf *TRUE* gesetzt, dann wird der Wert des Eingangs in die Property *Program_Change* geschrieben (siehe auch Transition-Diagram).

bRetrigger: Ein Wechsel von *FALSE* --> *TRUE* löst das erneute Schreiben des Eingangs **ePV** in die Property *Program_Change* aus, wenn Eingang **bEnablePV** auf *TRUE* gesetzt ist.

VAR_OUTPUT

```
bReady              : BOOL;
ePrgState           : E_BACnetProgramState;
ePrgChangeReq       : E_BACnetProgramRequest;
ePrgReasonHalt      : E_BACnetProgramError;
bOverridden         : BOOL;
bOutOfService       : BOOL;
bFault              : BOOL;
bInAlarm            : BOOL;
bError              : BOOL;
nErrorId            : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (ProgramState, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_RemoteDevice nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

ePrgState: Rückmeldung des aktuellen Programmzustands (siehe auch Transition-Diagram).

ePrgChangeReq: Rückmeldung über den Zustand der aktuellen Programmanforderung (siehe auch Transition-Diagram).

ePrgReasonHalt: Fehlerrückmeldung bei Programmabbruch.

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Program* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

Transition-Diagram

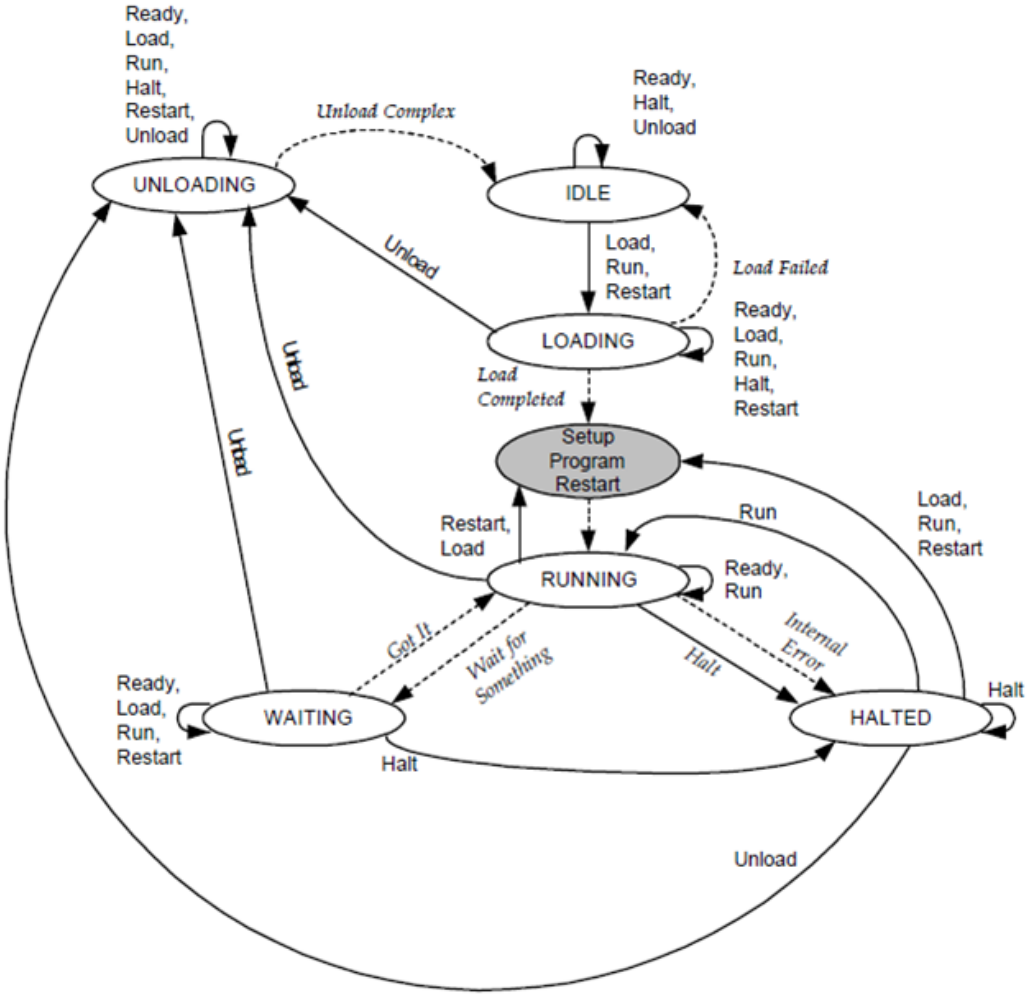
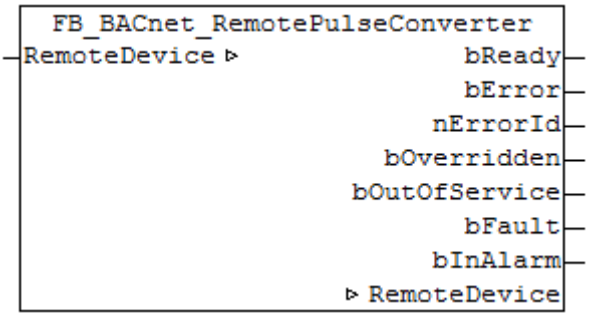


Abb. 171: Bild-2: aus BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt Program, Abbildung 12-3 "State Transitions for the program object"

5.4.21 FB_BACnet_RemotePulseConverter

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemotePulseConverter" kann lesend auf ein entferntes BACnet-Objekt vom Typ *PulseConverter* (PC) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

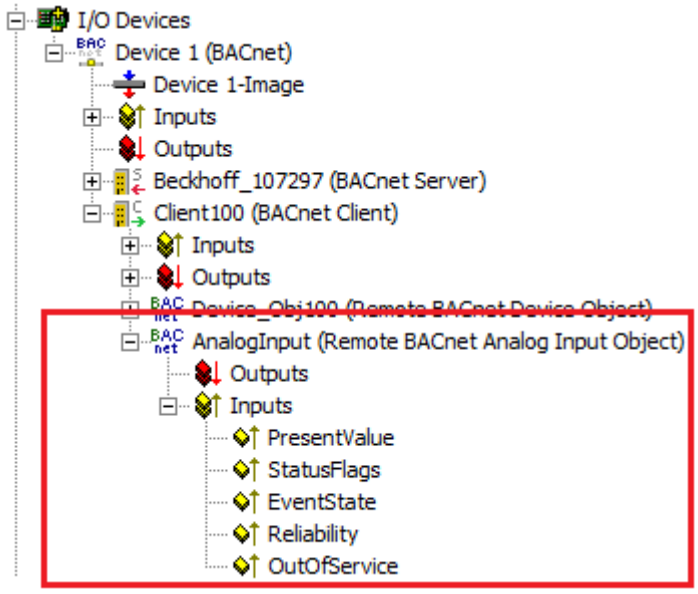


Abb. 172: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_OUTPUT

```
bReady       : BOOL;
bOverridden  : BOOL;
bOutOfService: BOOL;
bFault       : BOOL;
bInAlarm     : BOOL;
bError       : BOOL;
nErrorId     : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang FALSE, dann meldet der zugehörige Funktionsbaustein FB_BACnet_RemoteDevice [▶ 465] nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

bOverride, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *PulseConverter* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice [▶ 465])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

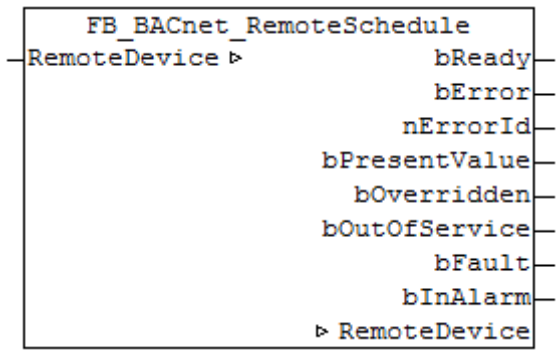
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

5.4.22 FB_BACnet_RemoteSchedule

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteSchedule" kann lesend auf ein entferntes BACnet-Objekt vom Typ *Schedule* (SCHED) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

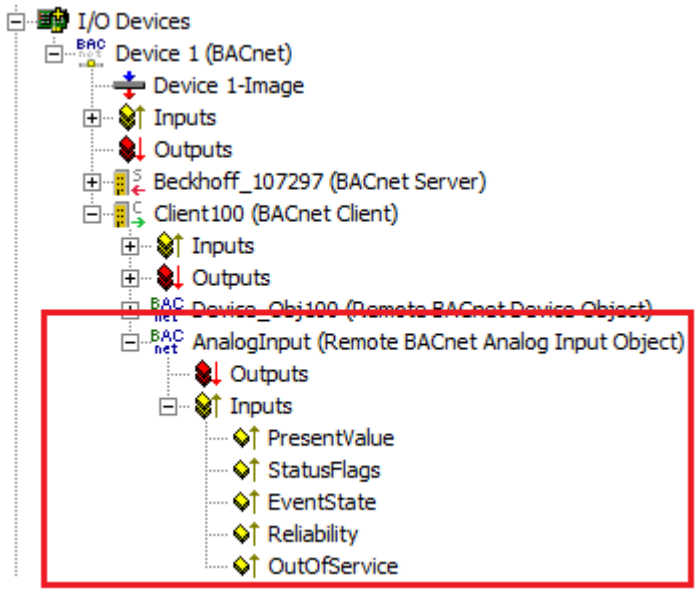


Abb. 173: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_OUTPUT

```
bReady           : BOOL;
bPresentValue    : BOOL;
bOverridden      : BOOL;
bOutOfService    : BOOL;
bFault           : BOOL;
bInAlarm         : BOOL;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (PresentValue, Overridden ...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein "FB_BACnet_RemoteDevice" nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

bPresentValue: Aktueller Wert des BACnet-Objekts (siehe auch BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Schedule* und Property *Present_Value*). Der Zustand wird in 3 Stufen (Enumeration) angegeben (0 = INACTIVE, 1 = ACTIVE und 2 = NULL).

bOverridden, bOutOfService, bFault, bInAlarm: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *Schedule* und Property *Status_Flags*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice [▶ 465])
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

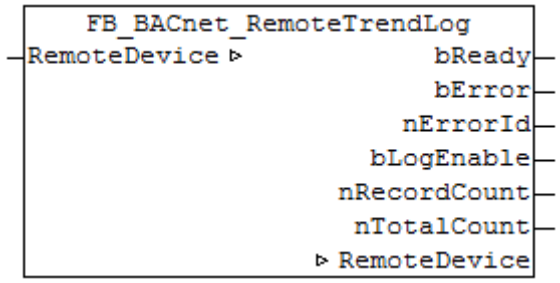
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter [▶ 378] und FB_BACnet_RemoteDevice [▶ 465] für weitere Informationen.

5.4.23 FB_BACnet_RemoteTrendLog

Der folgende Funktionsbaustein wird für die Verbindung von einem entfernten BACnet-Objekt eines lokalen BACnet-Clients verwendet. Die Verknüpfung des Funktionsbausteins zu dem entsprechenden BACnet-Objekt erfolgt mit Hilfe von Prozessdaten. Der Datenaustausch mit dem entfernten BACnet-Server erfolgt via BACnet mit Hilfe von WOC (Write-On-Change) und COV (Change-On-Value) oder via Polling (nicht empfohlen).

Die Prozessdaten können manuell in dem BACnet-Objekt erstellt und von Hand verknüpft werden oder mittels PLC-Automapping [▶ 64] automatisch erzeugt werden. Die für das PLC-Automapping [▶ 64] nötigen Kommentare ((* ~ (BACnet... | ??? | ???) *)) sind bereits in der Deklaration des Funktionsbausteins enthalten.



Verwendung

Mit Hilfe des Funktionsbausteins "FB_BACnet_RemoteTrendLog" kann lesend auf ein entferntes BACnet-Objekt vom Typ *TrendLog* (TREND) zugegriffen werden. Das entfernte BACnet-Objekt wurde dazu unter einem lokalen BACnet-Client hinzugefügt.

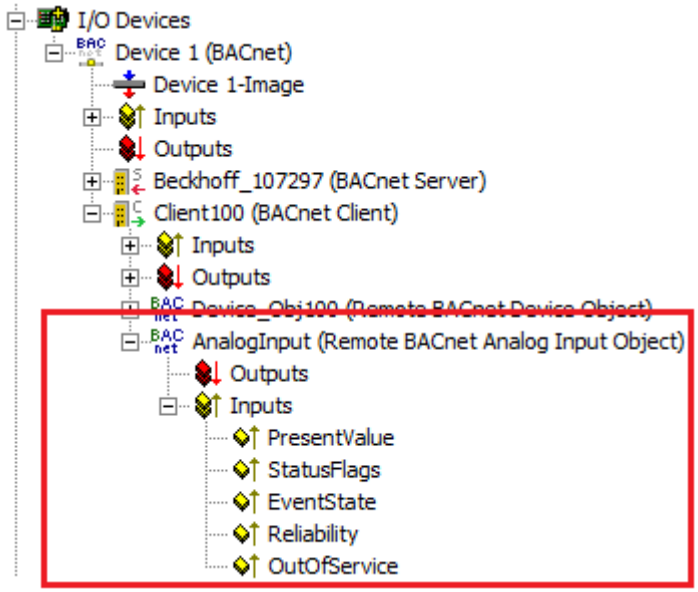


Abb. 174: Bild-1: Beispiel eines entfernten BACnet-Objekts unterhalb eines lokalen BACnet-Clients.

VAR_OUTPUT

```
bReady           : BOOL;
bLogEnable       : BOOL;
nRecordCount     : UDINT;
```

```
nTotalCount      : UDINT;
bError           : BOOL;
nErrorId         : UINT;
```

bReady: Meldung der allgemeinen Bereitschaft. Ist dieser Ausgang gesetzt, so sind die übrigen Status-Ausgänge gültig (LogEnable, RecordCount...). Ist der Ausgang *FALSE*, dann meldet der zugehörige Funktionsbaustein FB_BACnet_RemoteDevice nicht "Operational", die Baustein-Instanz wurde im System Manager nicht richtig verknüpft oder der entfernte Server ist nicht erreichbar (Gateway nicht erreichbar, kein Ethernet-Link).

bLogEnable: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Log_Enable*.

nRecordCount: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Record_Count*.

nTotalCount: Siehe BACnet-Spezifikation DIN EN ISO 16484-5 zum BACnet-Objekt *TrendLog* und Property *Total_Record_Count*.

bError: Ein Fehler steht an.

nErrorId: Fehlernummer
0 = kein Fehler
1 = Funktionsbaustein des zugehörigen Clients (RemoteDevice) wird gar nicht oder zu unregelmäßig im SPS-Programm aufgerufen.
2 = fehlerhaftes Prozessdatenmapping erkannt (Mapping im System Manager prüfen; evtl. SPS-Projekt komplett übersetzen und erneut laden)
3 = der zugehörige BACnet-Client ist nicht bereit (**bOperational** = *FALSE* an Instanz des FB_BACnet_RemoteDevice)
Die Fehlernummern können als Baustein-Konstanten über die FB-Instanz abgefragt werden (*FB_BACnet_Remote???*.**nERR_xxx**).

VAR_IN_OUT

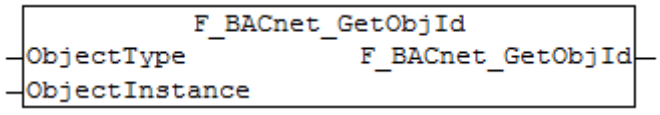
```
RemoteDevice     : FB_BACnet_RemoteDevice;
```

RemoteDevice: Angabe der Baustein-Instanz des zugehörigen entfernten BACnet-Device Objekts (remote). Das entfernte BACnet-Device Objekt eines entfernten BACnet-Servers ist unter einem lokalen BACnet-Client hinzugefügt. Lokaler Client und entfernter Server sind via BACnet verbunden. Pro BACnet-Adapter sind beliebig viele Clients möglich. Siehe FB_BACnet_Adapter und FB_BACnet_RemoteDevice für weitere Informationen.

5.5 BACnet Helper

5.5.1 F_BACnet_GetObjId

Funktion zur Codierung des Objekt-Typs und der Objekt-Instance in den BACnet *Object_Identifier*.



VAR_INPUT

```
ObjectType       : E_BACnetObjectType;
ObjectInstance   : UDINT;
```

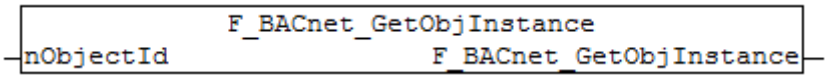
ObjectType: BACnet Objekt Typ.

ObjectInstance: BACnet Objekt Nummer (Instance, 0...4'194'303).

Rückgabewert ist der resultierende BACnet *Object_Identifier*.

5.5.2 F_BACnet_GetObjInstance

Funktion zum Decodierung des BACnet *Object_Identifier* in die Objekt-Instance (Objektnummer).



VAR_INPUT

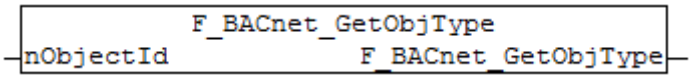
```
nObjectId    : DWORD;
```

nObjectId: BACnet *Object_Identifier*.

Rückgabewert ist die BACnet Objekt Nummer (Instance, 0...4'194'303).

5.5.3 F_BACnet_GetObjType

Funktion zur Decodierung des BACnet *Object_Identifier* in den Objekt-Typ.



VAR_INPUT

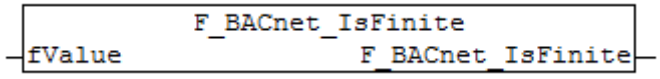
```
nObjectId    : DWORD;
```

nObjectId: BACnet *Object_Identifier*.

Rückgabewert ist der resultierende BACnet Objekt Typ (siehe E_BACnetObjectType [▶ 515]).

5.5.4 F_BACnet_IsFinite

Funktion zur Prüfung einer Gleitpunktzahl auf endlichen Wertebereich.



VAR_INPUT

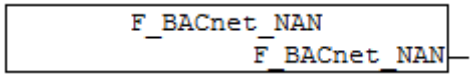
```
fValue       : REAL;
```

fValue: Zu prüfende Gleitpunktzahl.

Rückgabewert: TRUE = Wert liegt im endlichen Wertebereich, FALSE = Zahl unendlich positiv oder unendlich negativ.

5.5.5 F_BACnet_NAN

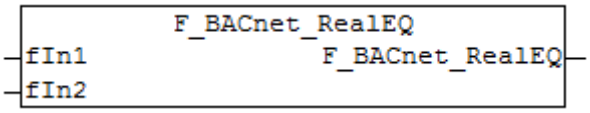
Funktion gibt den Gleitpunktwert "**N**ot **AN**umber" zurück. Damit können z.B. Prioritätseinträge im *Priority_Array* eines *Analog_Value* bzw. *Analog_Output* Objekts gelöscht werden.



Rückgabewert: "Not a number" bzw. **#QNAN** (0x7FFF0000).

5.5.6 F_BACnet_RealEQ

Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs. Diese Funktion sollte anstelle der Standardoperatoren ("=" bzw. "EQ") bei Werten aus BACnet verwendet werden, da sonst ein SPS Stopp ausgelöst wird, wenn die zu vergleichenden Werte nicht im endlichen Bereich liegen.



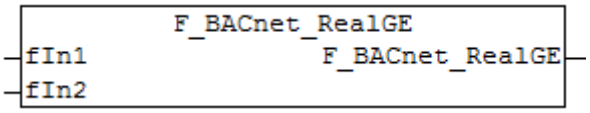
VAR_INPUT

```
fIn1         : REAL;
fIn2         : REAL;
```

Rückgabewert: TRUE = Wert 1 und Wert 2 sind identisch oder beide liegen nicht im endlichen Wertebereich. FALSE = Wert 1 und Wert 2 unterscheiden sich.

5.5.7 F_BACnet_RealGE

Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs. Diese Funktion sollte anstelle der Standardoperatoren (">=" bzw. "GE") bei Werten aus BACnet verwendet werden, da sonst ein SPS Stopp ausgelöst wird, wenn die zu vergleichenden Werte nicht im endlichen Bereich liegen.



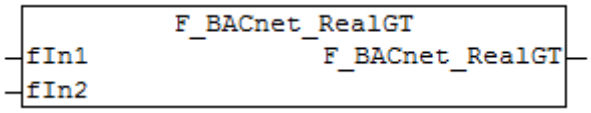
VAR_INPUT

```
fIn1         : REAL;
fIn2         : REAL;
```

Rückgabewert: TRUE = Wert 1 ist größer oder gleich Wert 2. FALSE = Wert 2 ist kleiner als Wert1 oder einer der beiden Werte liegt nicht im endlichen Wertebereich.

5.5.8 F_BACnet_RealGT

Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs. Diese Funktion sollte anstelle der Standardoperatoren (">" bzw. "GT") bei Werten aus BACnet verwendet werden, da sonst ein SPS Stopp ausgelöst wird, wenn die zu vergleichenden Werte nicht im endlichen Bereich liegen.



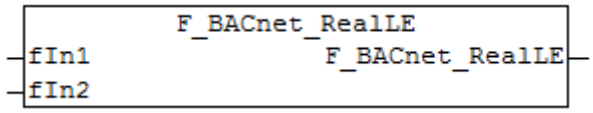
VAR_INPUT

```
fIn1         : REAL;
fIn2         : REAL;
```

Rückgabewert: TRUE = Wert 1 ist größer Wert 2. FALSE = Wert 2 ist kleiner oder gleich Wert1 oder einer der beiden Werte liegt nicht im endlichen Wertebereich.

5.5.9 F_BACnet_RealLE

Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs. Diese Funktion sollte anstelle der Standardoperatoren ("<=" bzw. "LE") bei Werten aus BACnet verwendet werden, da sonst ein SPS Stopp ausgelöst wird, wenn die zu vergleichenden Werte nicht im endlichen Bereich liegen.



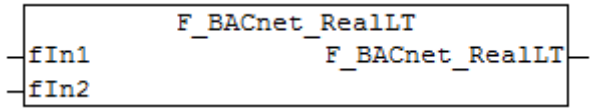
VAR_INPUT

```
fIn1         : REAL;
fIn2         : REAL;
```

Rückgabewert: TRUE = Wert 1 ist kleiner oder gleich Wert 2. FALSE = Wert 2 ist größer als Wert1 oder einer der beiden Werte liegt nicht im endlichen Wertebereich.

5.5.10 F_BACnet_RealLT

Funktion zum Vergleich von zwei Gleitpunktwerten unter Berücksichtigung des Wertebereichs. Diese Funktion sollte anstelle der Standardoperatoren ("<" bzw. "LT") bei Werten aus BACnet verwendet werden, da sonst ein SPS Stopp ausgelöst wird, wenn die zu vergleichenden Werte nicht im endlichen Bereich liegen.



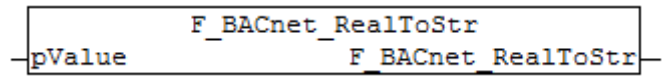
VAR_INPUT

```
fIn1         : REAL;
fIn2         : REAL;
```

Rückgabewert: TRUE = Wert 1 ist kleiner Wert 2. FALSE = Wert 2 ist größer oder gleich Wert1 oder einer der beiden Werte liegt nicht im endlichen Wertebereich.

5.5.11 F_BACnet_RealToStr

Funktion zur Umwandlung einer Gleitpunktzahl in eine Zeichenkette unter Berücksichtigung des Wertebereichs.



VAR_INPUT

```
pValue       : POINTER TO REAL;
```

pValue: Zeiger auf die zu wandelnde Gleitpunktzahl ("ADR()").

Rückgabewert: Bei endlichem Wertebereich wird die Zeichenkette der Gleitpunktzahl in dem gleichen Format ausgegeben, wie nach der Wandlung mit "REAL_TO_STRING()". Liegt der Wert im nicht-endlichen Bereich, dann wird entsprechend des Vorzeichens "#QNAN" (positive not-a-number) oder "-#QNAN" (negativ not-a-number) ausgegeben.

Beispiel

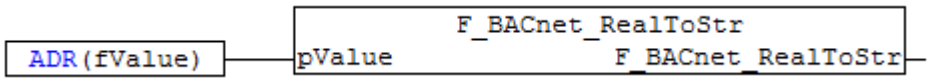
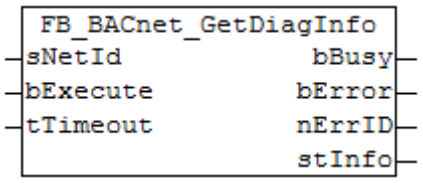


Abb. 175: Bild-1: Beispiel für die Verwendung mit Zeigerbestimmung durch "ADR()". Wobei "fValue" eine Variable vom Typ REAL darstellt.

5.5.12 FB_BACnet_GetDiagInfo

Funktionsbaustein für den Zugriff auf die BACnet Diagnose über ADS.



Verwendung

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen.

Die nötige NetId (AMS NetId) des BACnet Device kann im System Manager abgelesen werden (siehe Bild-1: Die NetId entspricht nicht der lokalen NetId und muss immer angegeben werden - keine Verwendung von Leerstrings möglich!). Eine andere Möglichkeit besteht darin, die NetId aus dem Device Baustein bzw. Adapter Baustein in der SPS abzufragen (siehe Bild-2).

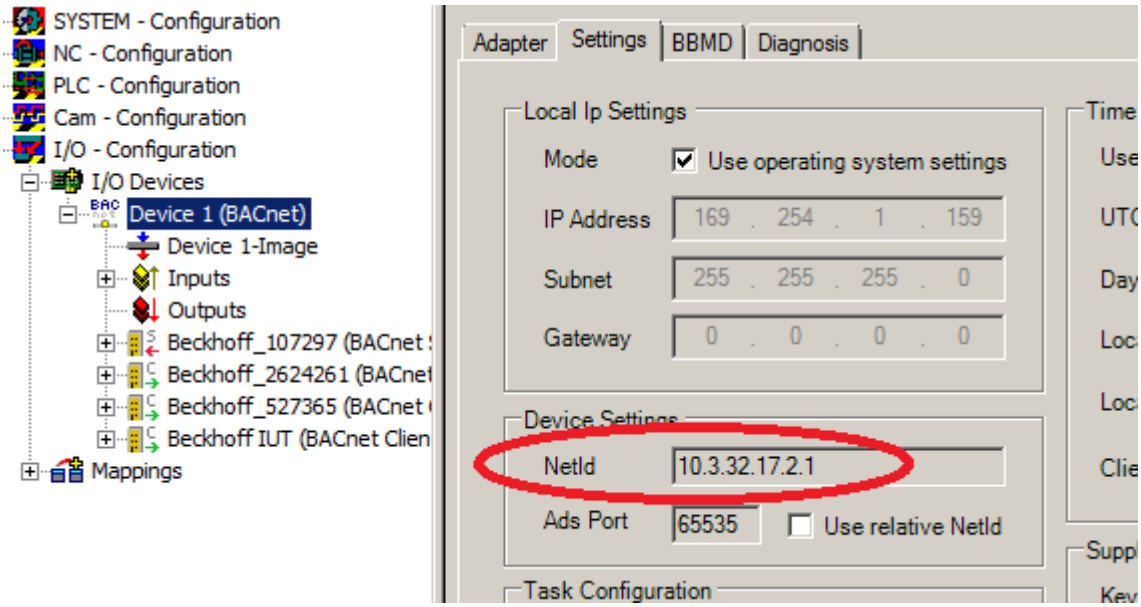


Abb. 176: Bild-1: AMS NetId des BACnet Device

Für Informationen zur Funktionsweise des Adapter Bausteins siehe FB_BACnet_Adapter [▶ 378].

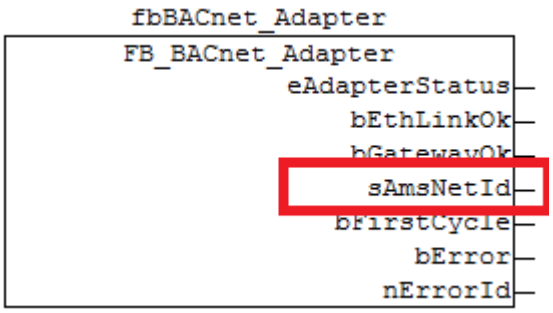


Abb. 177: Bild-2: AMS NetId des BACnet Device

VAR_INPUT

```
sNetId       : T_AmsNetId;
bExecute     : BOOL;
tTimeout     : TIME := tBACnet_ADSTimeOut;
```

sNetId: AMS NetId des BACnet Devices (Adapter) unter dem der Server bzw. Client konfiguriert wurde.

bExecute: Steigende Flanke am Eingang startet den Lesevorgang.

tTimeout: Optionaler Eingang, Überwachungszeit für den ADS Zugriff (Default: tBACnet_ADSTimeOut [▶ 519]).

VAR_OUTPUT

```
bBusy        : BOOL;
bError       : BOOL;
nErrID       : UDINT;
stInfo       : ST_BACnet_Diagnosis;
```

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrID: ADS Fehlercode.

stInfo: Struktur mit Diagnose-Informationen zu BACnet.

Beispiel

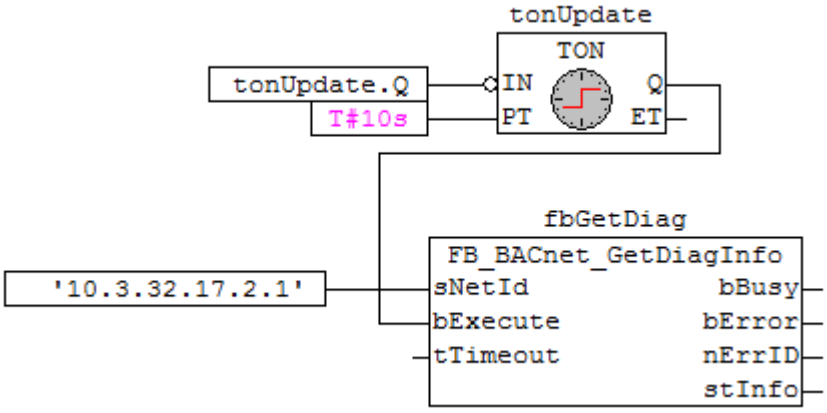
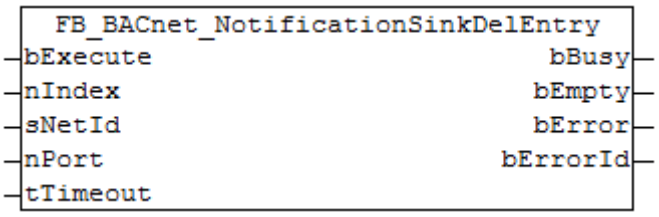


Abb. 178: Bild-3: Beispiel für das zyklische Lesen der Diagnose-Informationen

5.5.13 FB_BACnet_NotificationSinkDelEntry

Funktionsbaustein zum Löschen von Einträge der "Notification Sink" über ADS.



Verwendung

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Eingang **nPort** wird mit dem ADS Port aus der System Manager Konfiguration belegt (siehe Bild-1). Je nach Konfiguration können Ports variieren.

Mit Hilfe des Bausteins können Einträge an beliebiger Stelle der "Notification Sink" gelöscht werden. Wurden sämtliche Einträge entfernt, bzw. ist der angegebene Index größer als Einträge in der Liste vorhanden sind, dann meldet der Baustein **bEmpty** zurück.

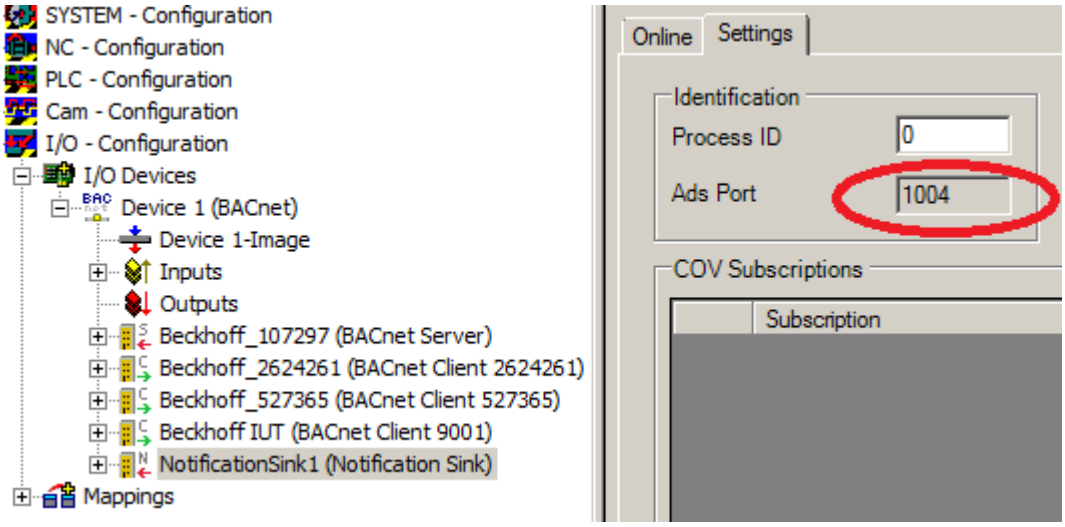


Abb. 179: Bild-1: ADS Port im System Manager

Die nötige NetId (AMS NetId) des BACnet Device kann im System Manager abgelesen werden (siehe Bild-2).

Die NetId entspricht nicht der lokalen NetId und muss immer angegeben werden - keine Verwendung von Leerstrings möglich!). Eine andere Möglichkeit besteht darin, die NetId aus dem Device Baustein bzw. Adapter Baustein in der SPS abzufragen (siehe Bild-3).

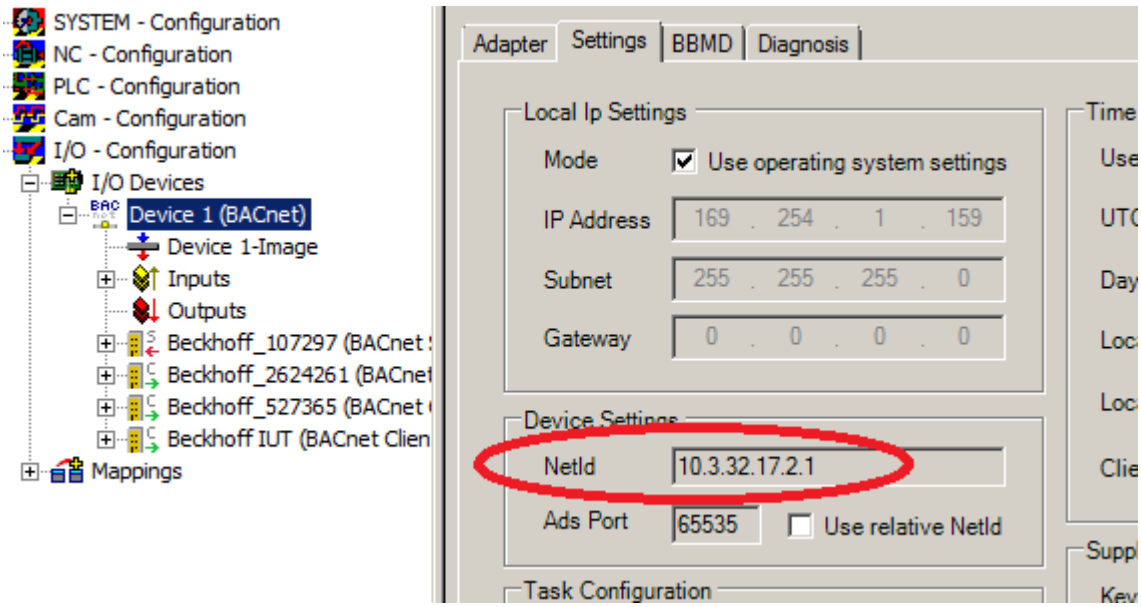


Abb. 180: Bild-2: AMS NetId des BACnet Device

Für Informationen zur Funktionsweise des Adapter Bausteins siehe FB_BACnet_Adapter [▶ 378].

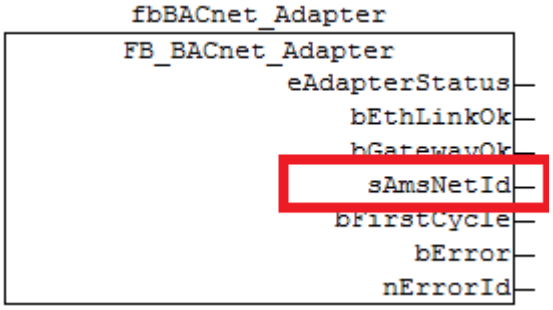


Abb. 181: Bild-3: AMS NetId des BACnet Device

VAR_INPUT

```
bExecute     : BOOL;
nIndex       : UDINT;
sNetId       : T_AmsNetId;
nPort        : T_AmsPort;
tTimeout     : TIME := tBACnet_ADSTimeOut;
```

bExecute: Steigende Flanke am Eingang startet den Lesevorgang.

nIndex: Index des zu löschenden Eintrags (0 ... n-1).

sNetId: AMS NetId des BACnet Devices (Adapter) unter dem der Server bzw. Client konfiguriert wurde.

nPort: ADS Port unter dem die "Notification Sink" angelegt wurde (siehe auch unter Verwendung [▶ 497]).

tTimeout: Optionaler Eingang, Überwachungszeit für den ADS Zugriff (Default: tBACnet_ADSTimeOut [▶ 519]).

VAR_OUTPUT

```
bBusy        : BOOL;
bEmpty       : BOOL;
bError       : BOOL;
nErrorId     : UDINT;
```

bBusy: Der Baustein ist beschäftigt.

bEmpty: Ist gesetzt, wenn versucht wird einen nicht vorhandenen Eintrag zu löschen.

Wird der letzte Eintrag (Index 0) gelöscht, dann erfolgt die Meldung **bEmpty** erst nach einem erneuten Versuch einen Eintrag (Index 0) aus der bereits leeren Liste zu löschen!

bError: Fehler während der Abarbeitung.

nErrorId: ADS Fehlercode.

Beispiel

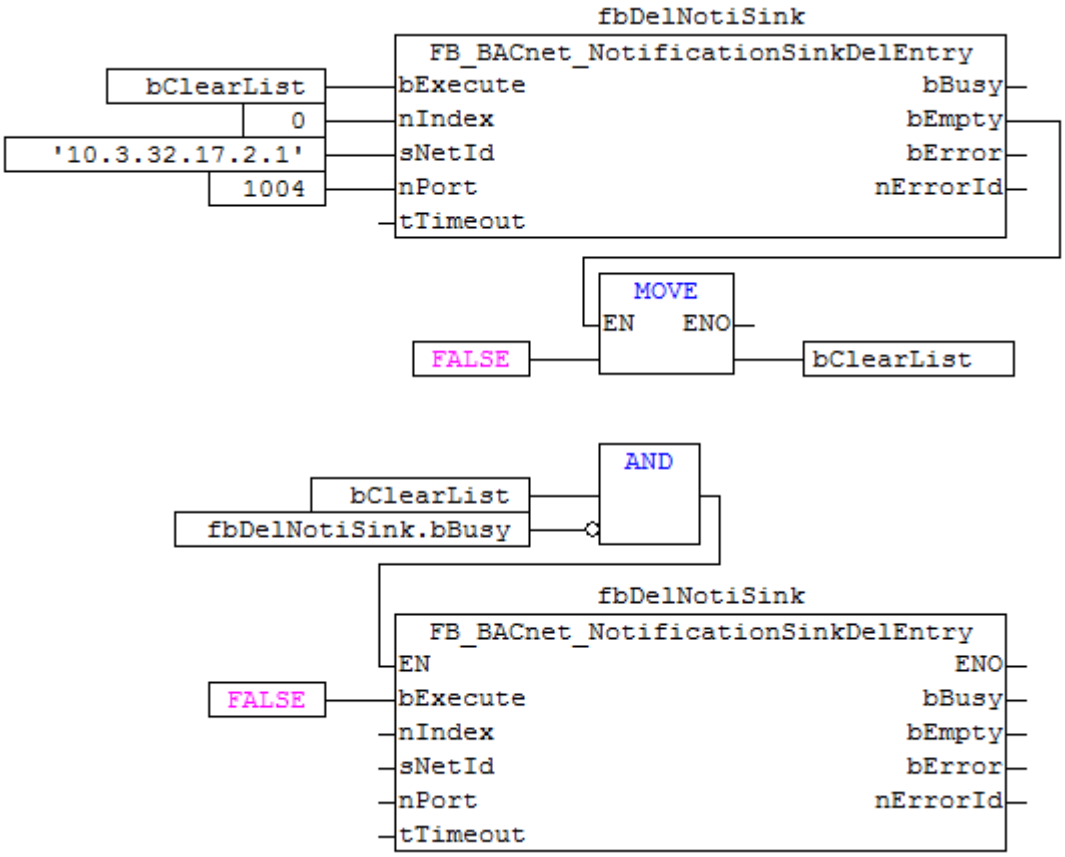
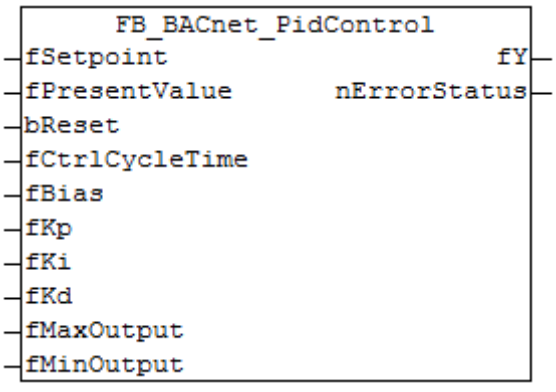


Abb. 182: Bild-4: Beispiel für das Löschen der kompletten "Notification Sink" unter Port 1004.

Wird das Bit "bClearList" gesetzt, dann wird das Löschen des Eintrags "0" so oft wiederholt bis der Baustein **bEmpty** zurück meldet. Anschließend wird das Bit "**bClearList**" selbständig zurück gesetzt.

5.5.14 FB_BACnet_PidControl

PID Regelbaustein in Parallelanordnung (Kp wirkt nicht auf I- und D-Anteil). Der Regelbaustein dient als Basis für das Objekt "LOOP" (FB_BACnet_Loop [▶ 425]).



Blockdiagramm

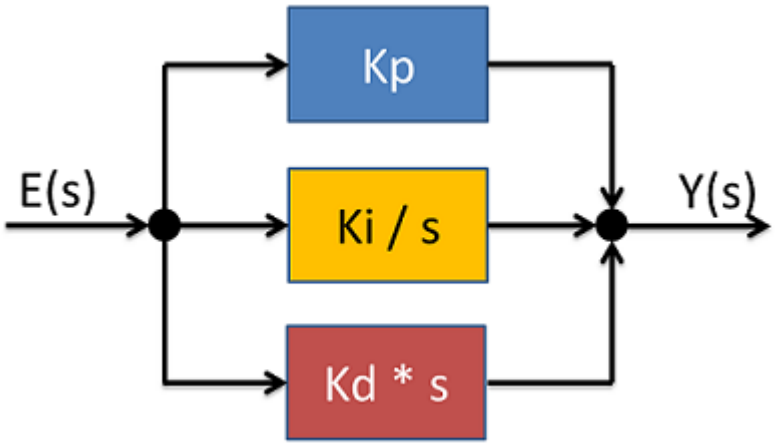


Abb. 183: Bild-1: Blockschaltbild der Übertragungsfunktion (Frequenzbereich).

Verwendung

Der Regelbaustein muss zyklisch im SPS Programm aufgerufen werden. Dabei ist zu beachten, dass die Zykluszeit des Aufruf am Eingang **fCtrlCycleTime** möglichst genau übergeben wird. Die Zykluszeit kann z.B. aus der globalen Struktur "SystemTaskInfoArr" ermittelt werden. Unter Beispiel [▶ 501] ist eine mögliche Beschaltung des Regler dargestellt.

VAR_INPUT

```
fSetpoint         : REAL; (* setpoint value *)
fPresentValue     : REAL; (* actual value *)
bReset            : BOOL; (* reset flag *)
fCtrlCycleTime    : REAL; (* controller cycle time in seconds [s] *)
fBias             : REAL; (* bias also active when reset is set to 1 *)
fKp               : REAL; (* proportional gain Kp (P) *)
fKi               : REAL; (* integral gain Ki (I) *)
fKd               : REAL; (* derivative gain Kd (D) *)
fMaxOutput        : REAL;
fMinOutput        : REAL;
```

fSetpoint: Sollwerteingabe (z.B. Temperaturvorgabe eines Raumes in [°C])

fPresentValue: Istwert (z.B. Raumisttemperatur in [°C])

bReset: Setzt den Regler zurück, die Ausgabe wird für die Dauer auf **fBias** gesetzt und durch **fMaxOutput** und **fMinOutput** begrenzt.

Wenn **fMinOutput** z.B. auf 5% gesetzt ist und **fBias** auf 0%, dann ist die Ausgabe **fY** für die Dauer des Reset auf 5% gestellt!

fCtrlCycleTime: Zykluszeit mit der der Regler aufgerufen wird in Sekunden [s].

fBias: Ausgabe-Offset. Dieser Wert wird mit der Ausgabe verrechnet und die Einheit muss der Einheit der Stellgröße entsprechen (z.B. Heizleistung in [%]).

fKp: Verstärkungsfaktor P. Die Regeldifferenz wird mit P multipliziert und auf die Ausgabe addiert (Ausgabe verhält sich proportional zur Eingabeabweichung).

fKi: Integralverstärkung I. Die Regeldifferenz wird unter Berücksichtigung der Zykluszeit mit I multipliziert, mit dem vorherigen Ergebnis summiert und auf die Ausgabe addiert (Hohe Regelabweichung = schnell steigende Ausgabe).

fKd: Differenzialverstärkung D. Die Regeldifferenz wird mit der vorherigen Reglerdifferenz verrechnen, das Ergebnis unter Berücksichtigung der Zykluszeit mit D multipliziert und auf die Ausgabe addiert (Starkes Reglerschwanken = starke Reaktion).

fMaxOutput: Obere Begrenzung des Ausgabewerts **fY**. Die Einheit muss der Einheit der Stellgröße entsprechen (z.B. Heizleistung maximal 90%).

fMinOutput: Untere Begrenzung des Ausgabewerts **fY**. Die Einheit muss der Einheit der Stellgröße entsprechen (z.B. Heizleistung minimal 5%).

VAR_OUTPUT

```
fY            : REAL; (* controller output command *)
nErrorStatus  : UINT; (* controller error output (0: no error; >0:error) *)
```

fY: Stellgröße (z.B. Heizleistung in [%]).

nErrorStatus: Fehlerstatus (*0 = kein Fehler*, 1 = ungültige Parameter, 2 = ungültige Zykluszeit). Fehlermeldungen müssen nicht quittiert werden.

Beispiel

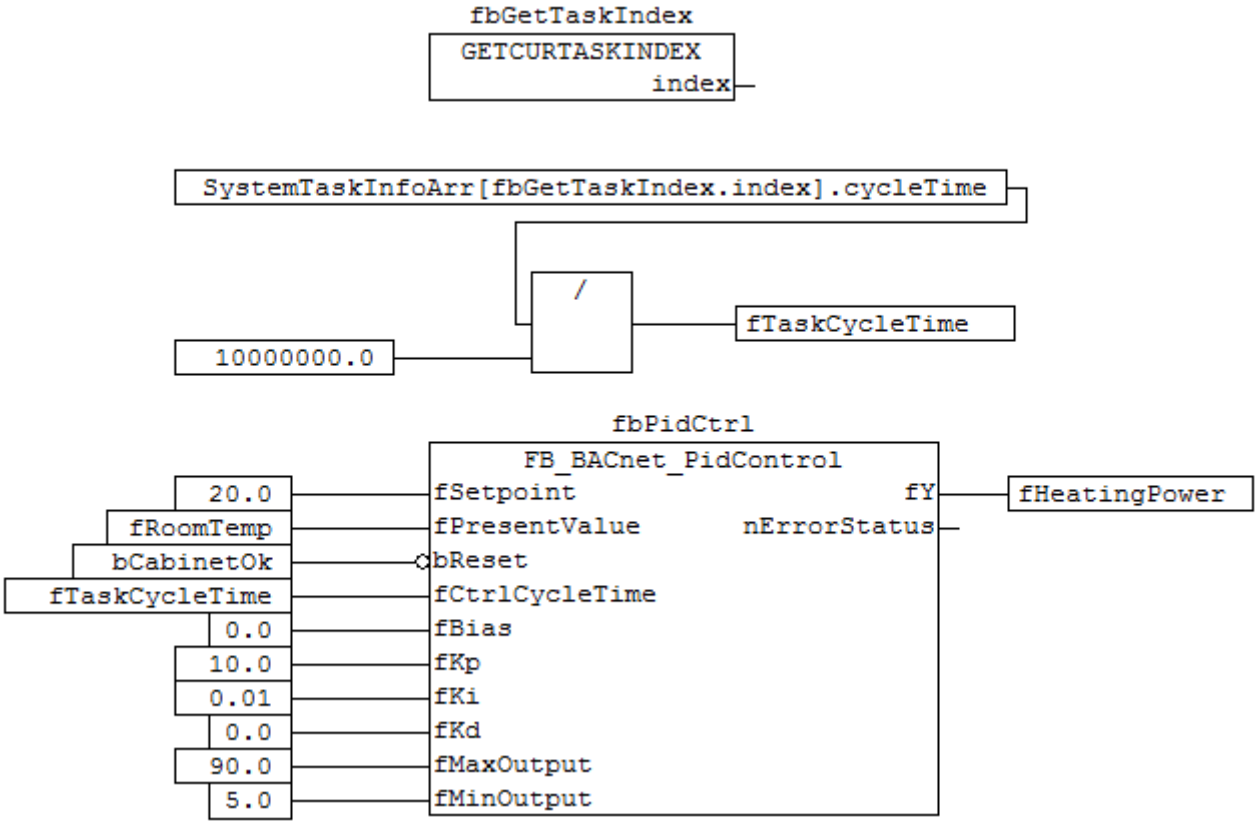


Abb. 184: Bild-4: Beispiel für die Verwendung

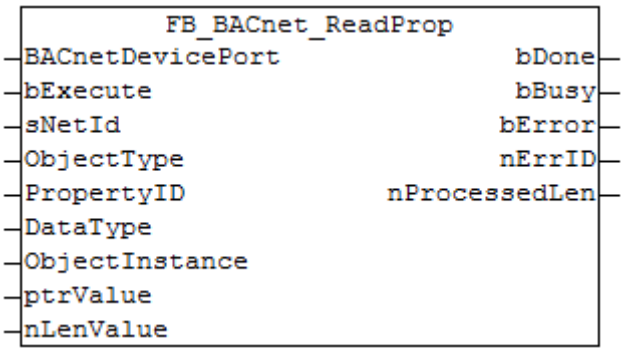
Anmerkung zu P = 10 --> Bei z.B. 1 K Temperaturdifferenz ergibt sich 10% Heizleistung

Anmerkung zu I = 0.01 --> Bei z.B. 1 K Temperaturdifferenz ergibt sich 0.01% Heizleistungsanstieg pro Sekunde

Anmerkung zu D = 0 --> Kein D-Anteil.

5.5.15 FB_BACnet_ReadProp

Funktionsbaustein für den Zugriff auf BACnet Properties über ADS. Sämtliche BACnet Properties von Server und Client Objekten können über ADS gelesen bzw. geschrieben werden. Wird auf einen Client über ADS zugegriffen, dann werden die ADS Zugriffe in "Read_Property" bzw. "Read_Range" BACnet-Abfragen umgewandelt und an den entfernten Server gesendet.



Verwendung

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Eingang **BACnetDevicePort** wird mit dem ADS Port aus der System Manager Konfiguration belegt (siehe Bild-1). Je nach Konfiguration können Ports für Server und Client variieren.

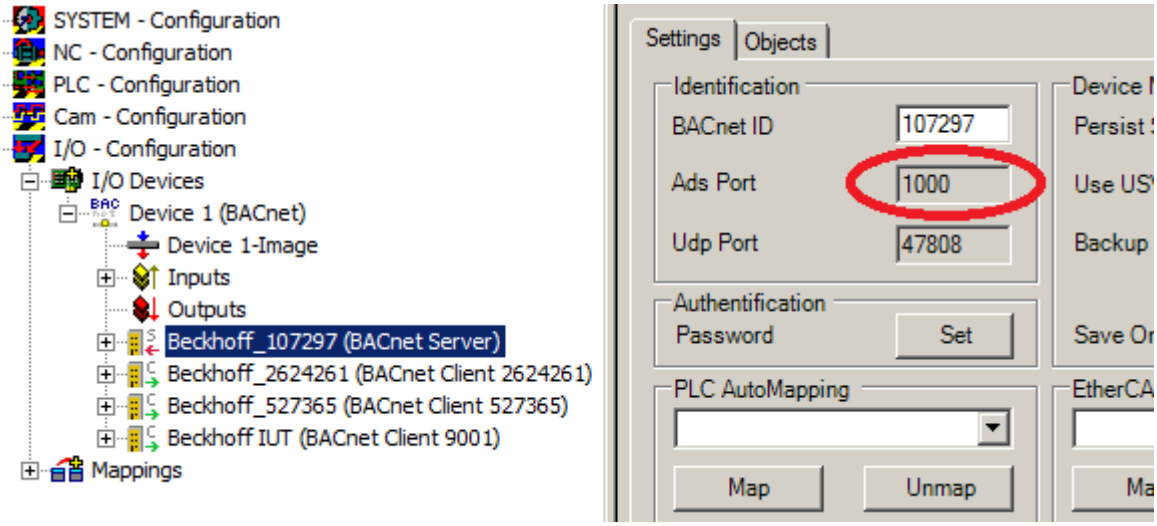


Abb. 185: Bild-1: ADS Port im System Manager

Die NetId (AMS NetId) des BACnet Device kann ebenfalls im System Manager abgelesen werden (siehe Bild-2).

Die NetId entspricht nicht der lokalen NetId und muss immer angegeben werden - keine Verwendung von Leerstrings möglich!). Eine andere Möglichkeit besteht darin, die NetId aus dem Device Baustein bzw. Adapter Baustein in der SPS abzufragen (siehe Bild-3).

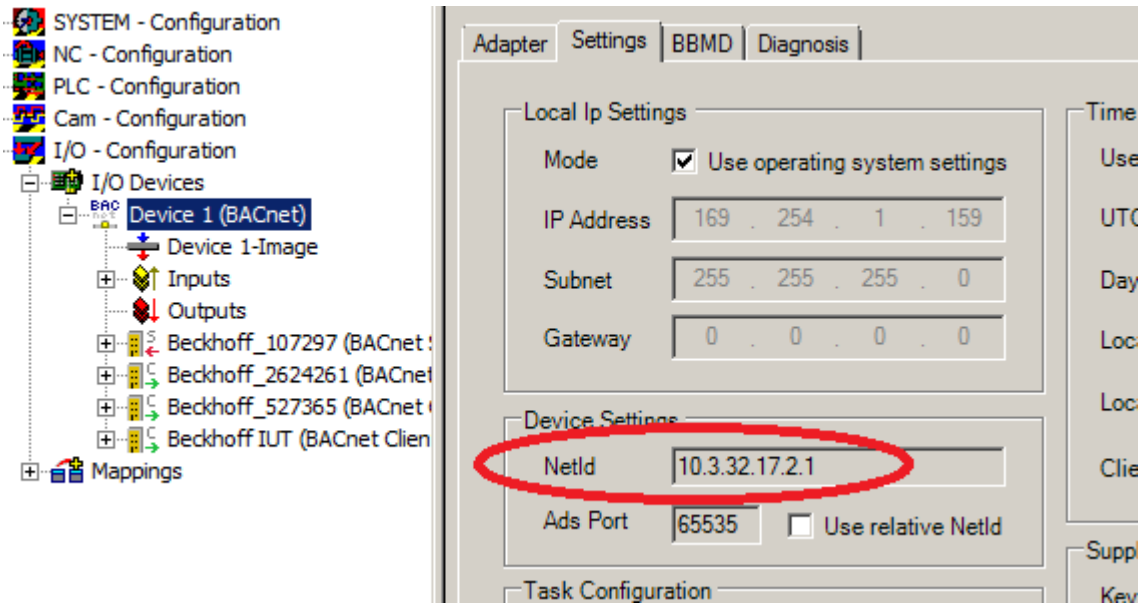


Abb. 186: Bild-2: AMS NetId des BACnet Device

Für Informationen zur Funktionsweise des Adapter Bausteins siehe FB_BACnet_Adapter [▶ 378].

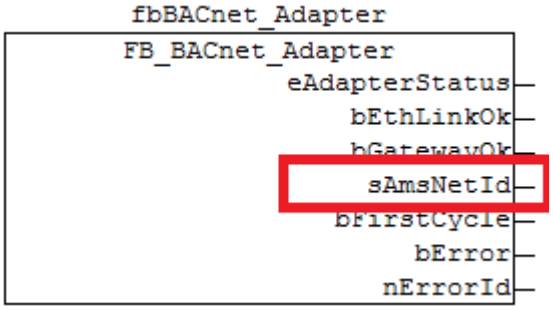


Abb. 187: Bild-3: AMS NetId des BACnet Device

VAR_INPUT

```
BACnetDevicePort : UINT := 16#FFFF;
bExecute         : BOOL;
sNetId           : T_AmsNetId;
ObjectType       : E_BACnetObjectType;
PropertyID       : E_BACnetPropertyIdentifier;
DataType         : E_BACnetDataTypes;
ObjectInstance   : DINT;
ptrValue         : POINTER TO BYTE;
nLenValue        : UDINT; (* byte size of value (SIZEOF) *)
```

BACnetDevicePort: ADS Port unter dem das gewünschte Objekt angelegt wurde (siehe auch unter Verwendung).

bExecute: Steigende Flanke am Eingang startet den Lesevorgang. Fallende Flanke löscht den Ausgang **bDone**.

sNetId: AMS NetId des BACnet Devices (Adapter) unter dem der Server bzw. Client konfiguriert wurde.

ObjectType: Enumeration des Objekttypes (AnalogValue, BinaryInput...).

PropertyID: Enumeration des Property-Identifiers (Present_Value, Status_Flags...).

ObjectInstance: Objekt-Nummer des BACnet Objekts (die unteren 22 Bit des Object_Identifier).

ptrValue: Zeiger auf die Variable in der die gelesenen Daten abgelegt werden sollen (zu ermitteln mit ADR()).

nLenValue: Byte-Länge der Variable in der die gelesenen Daten abgelegt werden sollen (zu ermitteln mit SIZEOF()).

VAR_OUTPUT

```
bDone          : BOOL;
bBusy          : BOOL;
bError         : BOOL;
nErrID         : UDINT;
nProcessedLen  : UDINT; (* the received data length (can be different to given length) *)
```

bDone: Lesen der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bExecute** zurückgesetzt wird. Wurde **bExecute** zurück gesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrID: ADS Fehlercode.

nProcessedLen: Byte-Länge der gelesenen Daten. Die Länge kann sich von der Ziellänge **nLenValue** unterscheiden (kleiner/gleich).

Beispiel

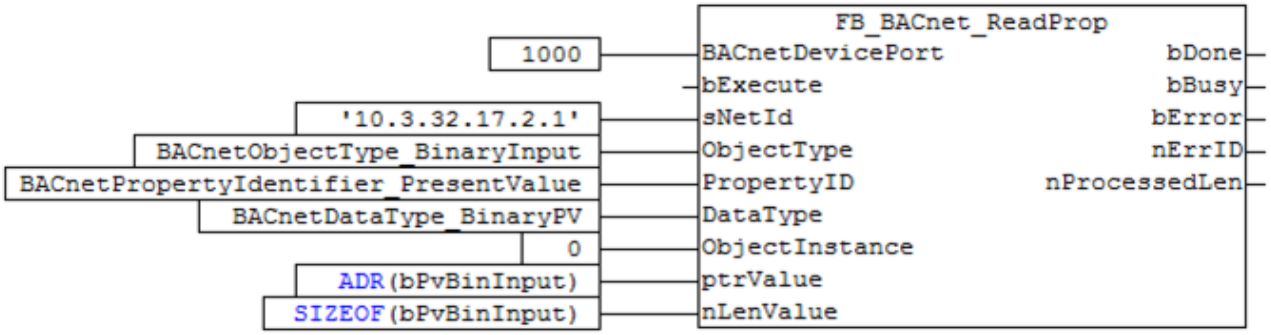
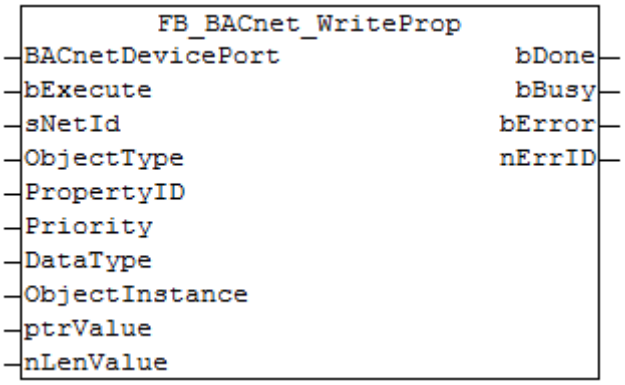


Abb. 188: Bild-4: Beispiel für die Verwendung

5.5.16 FB_BACnet_WriteProp

Funktionsbaustein für den Zugriff auf BACnet Properties über ADS. Sämtliche BACnet Properties von Server und Client Objekten können über ADS gelesen bzw. geschrieben werden. Wird auf einen Client über ADS zugegriffen, dann werden die ADS Zugriffe in "Write_Property" BACnet-Abfragen umgewandelt und an den entfernten Server gesendet.



Verwendung

Die Bausteininstanz wird im SPS Programm angelegt und zyklisch aufgerufen. Der Eingang **BACnetDevicePort** wird mit dem ADS Port aus der System Manager Konfiguration belegt (siehe Bild-1). Je nach Konfiguration können Ports für Server und Client variieren.

Durch Verwendung des WriteProp-Bausteins für den Zugriff z.B. auf das Present_Value kann WOC (Write On Change) auf lokale BACnet-Server-Objekte realisiert werden. Im Gegensatz zum Prozessdatenmapping kann die Priorität des Schreibzugriffs online angepasst werden. Zudem kommt es beim Prozessdatenmapping auf lokale Server-Objekte zum zyklischen Schreiben der Daten, so dass entfernte Clients nicht mit gleicher Priorität wie die lokale SPS zugreifen können.

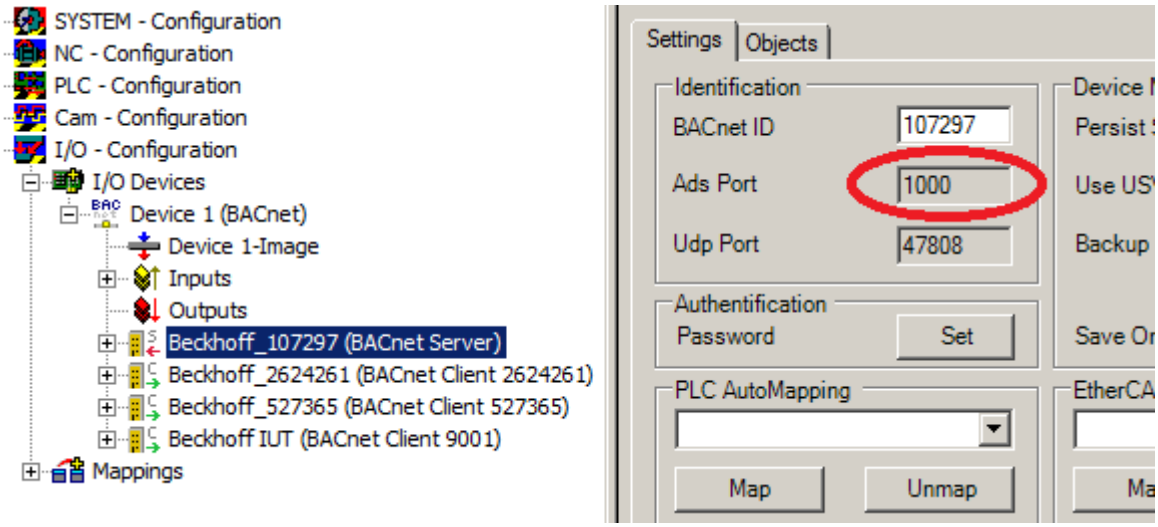


Abb. 189: Bild-1: ADS Port im System Manager

Die NetId (AMS NetId) des BACnet Device kann ebenfalls im System Manager abgelesen werden (siehe Bild-2).

Die NetId entspricht nicht der lokalen NetId und muss immer angegeben werden - keine Verwendung von Leerstrings möglich!). Eine andere Möglichkeit besteht darin, die NetId aus dem Device Baustein bzw. Adapter Baustein in der SPS abzufragen (siehe Bild-3).

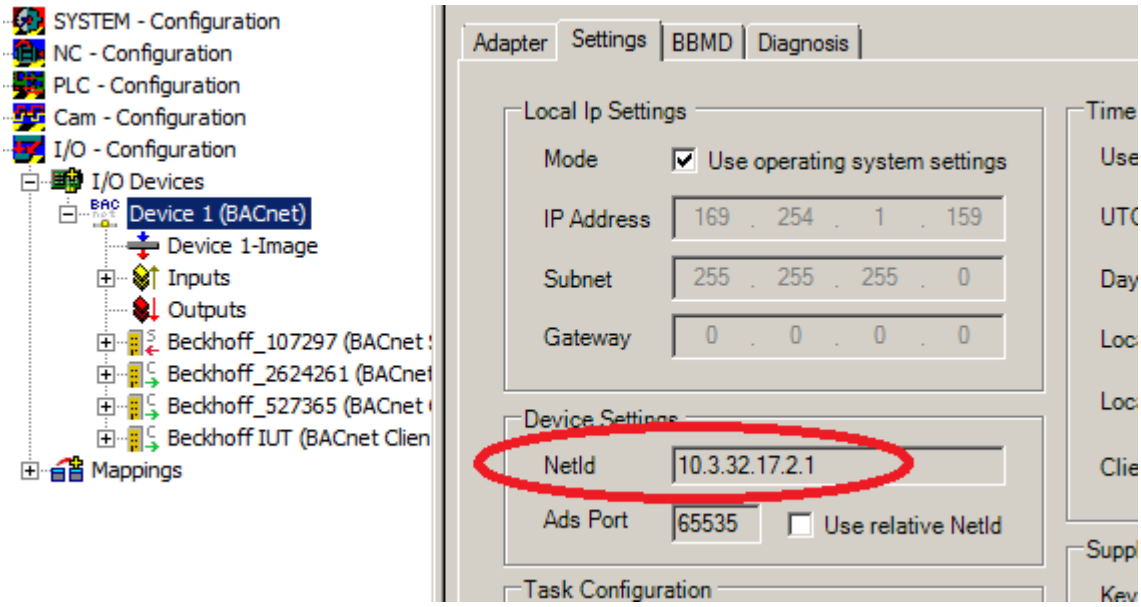


Abb. 190: Bild-2: AMS NetId des BACnet Device

Für Informationen zur Funktionsweise des Adapter Bausteins siehe FB_BACnet_Adapter [▶ 378].

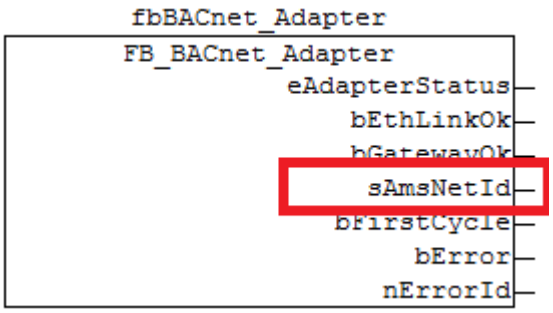


Abb. 191: Bild-3: AMS NetId des BACnet Device

VAR_INPUT

```
BACnetDevicePort : UINT := 16#FFFF;
bExecute         : BOOL;
sNetId           : T_AmsNetId;
ObjectType       : E_BACnetObjectType;
PropertyID       : E_BACnetPropertyIdentifier;
Priority         : E_BACnetPriority;
DataType         : E_BACnetDataTypes;
ObjectInstance   : UDINT;
ptrValue         : POINTER TO BYTE;
nLenValue        : UDINT; (* byte size of value (SIZEOF) *)
```

BACnetDevicePort: ADS Port unter dem das gewünschte Objekt angelegt wurde (siehe auch unter Verwendung).

bExecute: Steigende Flanke am Eingang startet den Lesevorgang. Fallende Flanke löscht den Ausgang **bDone**.

sNetId: AMS NetId des BACnet Devices (Adapter) unter dem der Server bzw. Client konfiguriert wurde.

ObjectType: Enumeration des Objekttypes (AnalogValue, BinaryInput...).

PropertyID: Enumeration des Property-Identifiers (Present_Value, Status_Flags...).

Priority: Enumeration der Priorität des Schreibzugriffs. Die Priorität wird bei Schreibzugriffen auf priorisierbare Properties benötigt (z.B. Present_Value). Ein Zugriff mit Priorität x erzeugt dann einen Eintrag im zugehörigen Priority_Array an x-ter Stelle.

ObjectInstance: Objekt-Nummer des BACnet Objekts (die unteren 22 Bit des Object_Identifier).

ptrValue: Zeiger auf die Variable aus der die zu schreibenden Daten entnommen werden (zu ermitteln mit ADR()).

nLenValue: Byte-Länge der Variable aus der die zu schreibenden Daten entnommen werden (zu ermitteln mit SIZEOF()).

VAR_OUTPUT

```
bDone        : BOOL;
bBusy        : BOOL;
bError       : BOOL;
nErrID       : UDINT;
```

bDone: Schreiben der Daten erfolgreich beendet. **bDone** bleibt so lange gesetzt bis **bExecute** zurückgesetzt wird. Wurde **bExecute** zurückgesetzt bevor **bDone** aktiv ist, dann wird **bDone** für einen Zyklus gesetzt.

bBusy: Der Baustein ist beschäftigt.

bError: Fehler während der Abarbeitung.

nErrID: ADS Fehlercode.

Beispiel

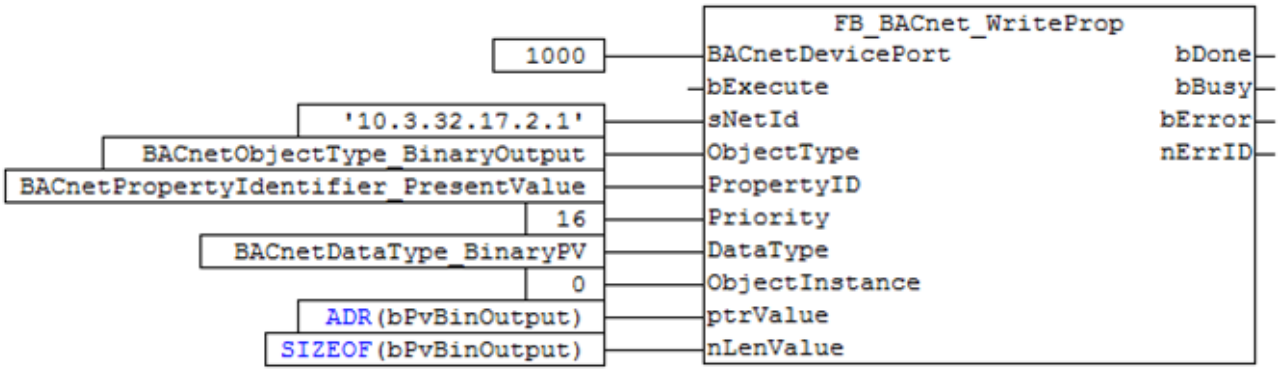


Abb. 192: Bild-4: Beispiel für die Verwendung

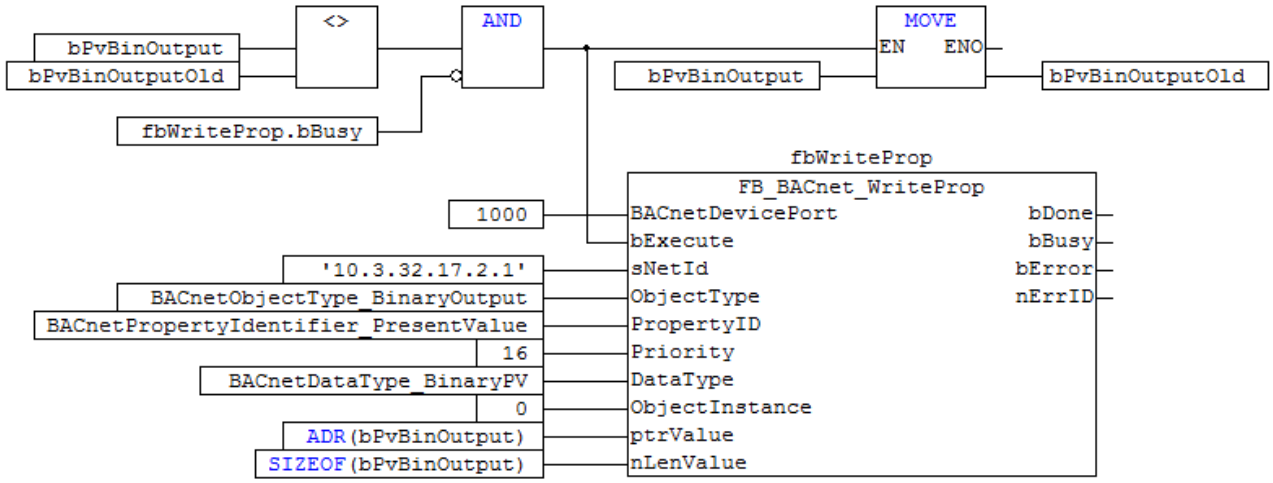


Abb. 193: Bild-5: Beispiel für das Schreiben bei Wertänderung (WOC)

5.6 BACnet Datatypes

5.6.1 ST_BACnet_Date

Struktur mit Datum Informationen.

```
nYear        : BYTE; (* year minus 1900 X'FF' = unspecified *)
nMonth       : BYTE; (* (1.. 12) 1 = January X'FF' = unspecified *)
nDay         : BYTE; (* day OF month (1..31), X'FF' = unspecified *)
nDayOfWeek   : BYTE; (* day OF week (1..7) 1 = Monday -- 7 = Sunday -- X'FF' = unspecified *)
```

nYear: Jahreszahl minus 1900 (0 ... 254 = 1900 ... 2154, 0xFF = nicht spezifiziert).

nMonth: Monatszahl (1...12, 1 = Januar, 0xFF = nicht spezifiziert).

nDay: Tag des Monats (1...31, 0xFF = nicht spezifiziert).

nDayOfWeek: Wochentag (1...7, 1 = Montag, 7 = Sonntag, 0xFF = nicht spezifiziert).

5.6.2 ST_BACnet_DateTime

Struktur mit Datum/Uhrzeit Informationen.

```
stDate : ST_BACnet_Date;
stTime : ST_BACnet_Time;
```

5.6.3 ST_BACnet_ConfirmedServiceDiag

Struktur mit Informationen zu bestätigten BACnet Diensten (confirmed).

```
nReqSend        : DWORD; (* ULONG *)
nReqSendFail    : DWORD; (* ULONG *)
nReqRecv        : DWORD; (* ULONG *)
nReqRecvFail    : DWORD; (* ULONG *)
nAckSend        : DWORD; (* ULONG *)
nAckSendFail    : DWORD; (* ULONG *)
nAckRecv        : DWORD; (* ULONG *)
nAckRecvFail    : DWORD; (* ULONG *)
```

nReqSend: Anzahl an gesendeten zu bestätigenden Anfragen (confirmed requests) inkl. Neuversuch (retries).

nReqSendFail: Anzahl an fehlgeschlagenen zu bestätigenden Anfragen (keine Bestätigung empfangen - acknowledge).

nReqRecv: Anzahl an empfangenen zu bestätigenden Anfragen (confirmed requests).

nReqRecvFail: Anzahl an zu bestätigenden Anfragen die nicht verarbeitet werden konnten (kein freier Speicher, fehlerhafte Codierung etc.).

nAckSend: Anzahl an gesendeten Bestätigungen (acknowledge).

nAckSendFail: Anzahl an fehlgeschlagenen Bestätigungen (acknowledge).

nAckRecv: Anzahl an empfangenen Bestätigungen (acknowledge).

nAckRecvFail: Anzahl an fehlerhaft empfangenen Bestätigungen (kein freier Speicher, fehlerhafte Codierung etc.).

5.6.4 ST_BACnet_DiagEthStatistics

Struktur mit Informationen zum Ethernet-Adapter.

```
ethStat    : ST_BACnet_TcIoEthStatistic;
txRxErrCnt : ST_BACnet_TcIoEthTxRxErrorCount;
```

5.6.5 ST_BACnet_DiagnosisTiming

Struktur mit Informationen zu fehlerhaften Netzwerk Frames aus dem Netzwerktreiber.

```
msIoInCurExecutionTime    : DWORD; (* ULONG *)
msIoInMinExecutionTime    : DWORD; (* ULONG *)
msIoInMaxExecutionTime    : DWORD; (* ULONG *)
msIoOutCurExecutionTime   : DWORD; (* ULONG *)
msIoOutMinExecutionTime   : DWORD; (* ULONG *)
msIoOutMaxExecutionTime   : DWORD; (* ULONG *)
msCycleCurExecutionTime   : DWORD; (* ULONG *)
msCycleMinExecutionTime   : DWORD; (* ULONG *)
msCycleMaxExecutionTime   : DWORD; (* ULONG *)
msInputCurDistanceTime    : DWORD; (* ULONG *)
msInputMinDistanceTime    : DWORD; (* ULONG *)
msInputMaxDistanceTime    : DWORD; (* ULONG *)
```

msIoInCurExecutionTime: Aktuelle Verarbeitungsdauer der Eingangsdaten im BACnet-Task (ms).

msIoInMinExecutionTime: Minimale Verarbeitungsdauer der Eingangsdaten im BACnet-Task (ms).

msIoInMaxExecutionTime: Maximale Verarbeitungsdauer der Eingangsdaten im BACnet-Task (ms).

msIoOutCurExecutionTime: Aktuelle Verarbeitungsdauer der Ausgangsdaten im BACnet-Task (ms).

msIoOutMinExecutionTime: Minimale Verarbeitungsdauer der Ausgangsdaten im BACnet-Task (ms).

msIoOutMaxExecutionTime: Maximale Verarbeitungsdauer der Ausgangsdaten im BACnet-Task (ms).

msCycleCurExecutionTime: Aktuelle Verarbeitungsdauer der BACnet-Task (ms).

msCycleMinExecutionTime: Minimale Verarbeitungsdauer der BACnet-Task (ms).

msCycleMaxExecutionTime: Maximale Verarbeitungsdauer der BACnet-Task (ms).

msInputCurDistanceTime: Aktuelle Zeitdauer zwischen dem Beginn der Eingangsdatenverarbeitung (ms).

msInputMinDistanceTime: Minimale Zeitdauer zwischen dem Beginn der Eingangsdatenverarbeitung (ms).

msInputMaxDistanceTime: Maximale Zeitdauer zwischen dem Beginn der Eingangsdatenverarbeitung (ms).

5.6.6 ST_BACnet_DiagnosisTiming

Struktur mit Informationen zu BACnet Frames.

```
confirmed            : ARRAY[0..29] OF ST_BACnet_ConfirmedServiceDiag; (* [SERVICE_CONFIRMED_MAX + 1
]; 29 + 1 *)
unConfirmed          : ARRAY[0..9] OF ST_BACnet_UnConfirmedServiceDiag; (* [SERVICE_UNCONFIRMED_MAX
+ 1]; 9 *)
nSegmSendTimeouts    : DWORD; (* ULONG *)
nFrameAllocFailCnt   : DWORD; (* ULONG *)
nFrameSendCnt        : DWORD; (* ULONG *)
nFrameSendFailCnt    : DWORD; (* ULONG *)
nFrameRecvCnt        : DWORD; (* ULONG *)
nFrameRecvFailCnt    : DWORD; (* ULONG *)
nLinkStatusChangedCnt: DWORD; (* ULONG *)
```

nSegmSendTimeouts: Zeitüberschreitung beim Empfang der Bestätigung von segmentierten Paketen (Gegenstelle quittiert eine segmentierte Antwort nicht, Segmented-ACK)

nFrameAllocFailCnt: Anzahl verlorener Frames wegen Problemen bei der Speicherreservierung (kein freier Speicher).

nFrameSendCnt: Anzahl gesendeter Frames.

nFrameSendFailCnt: Anzahl fehlgeschlagener Frames.

nFrameRecvCnt: Anzahl empfangener Frames.

nFrameRecvFailCnt: Anzahl fehlerhaft empfangener Frames.

nLinkStatusChangedCnt: Häufigkeit der Netzwerk-Statusänderung (Kabel gezogen/Link lost, Kabel gesteckt/Link OK).

5.6.7 ST_BACnet_Info

Struktur mit allgemeinen Informationen zur BACnet-Task.

```
nAmsAllocCnt: DINT; (* LONG *)
nAmsAllocCntMax: DWORD; (* ULONG *)
nAmsAllocFailCnt: DWORD; (* ULONG *)
nAvailPhysMemBytes: DWORD; (* ULONG *)
nAvailFlashMemBytes: ARRAY[0..1] OF DWORD; (* ULONGLONG *)
nRecvFrameFifoSize: DWORD; (* ULONG *)
nEmptyFrameFifoSize: DWORD; (* ULONG *)
nRecvSegmUDPFramesListSize: DWORD; (* ULONG *)
nRecvSegmFramesListSize: DWORD; (* ULONG *)
nSendSegmFramesListSize: DWORD; (* ULONG *)
nClientScanListSize: DWORD; (* ULONG *)
nAdsRequestListSize: DWORD; (* ULONG *)
```

nAmsAllocCnt: Anzahl reservierter Router-Speicher Blöcke (1 Block = 1024 Byte). Die Anzahl sollte nicht kontinuierlich steigen (Maximal 2000 Blöcke=2MByte sind per Default verfügbar). Andernfalls den Router-Speicher im System Manager erhöhen (siehe Bild-1).

nAmsAllocCntMax: Spitzenwert der gleichzeitig reservierten Router-Speicher Blöcke.

nAmsAllocFailCnt: Anzahl fehlgeschlagener Router-Speicher Reservierungen (Reservierung schlägt fehl, wenn keine freien Blöcke mehr verfügbar).

nAvailPhysMemBytes: Freier, physikalischer Speicher des Betriebssystems in Byte.

nAvailFlashMemBytes: Freier Festplattenspeicher des Betriebssystems in Byte.

nRecvFrameFifoSize: Anzahl der aktuell zwischengespeicherten Frames in der receive Frame Queue (Receive Queue Size unter "Network Adapter Settings")

nEmptyFrameFifoSize: Anzahl des noch freien Speichers für empfangene Frames um diese Zwischenspeichern zu können (Frame wird kopiert und dann in RecvFrameFifo kopiert)

nRecvSegmUDPFramesListSize: Anzahl der aktuell empfangenen und zwischengespeicherten IP Segmente (Name ist falsch, nicht UDP sondern IP)

nRecvSegmFramesListSize: Anzahl der aktuell empfangenen und zwischengespeicherten BACnet Frame Segmente

nSendSegmFramesListSize: Anzahl der aktuell BACnet Frame Segmente die vor dem Senden zwischengespeichert sind

nClientScanListSize: Anzahl der aktuell bekannten BACnet Geräte im Netzwerk (DeviceAddressBinding im Device Objekt wird daraus gebildet)

nAdsRequestListSize: Anzahl der aktuell offenen internen Ads Requests die beim Lesen/Schreiben von Dateien (Stack greift per Ads auf lokale Dateien zu)

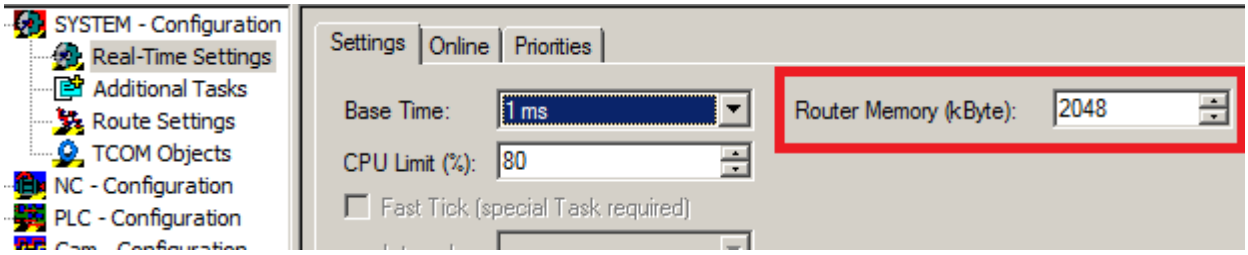


Abb. 194: Bild-1: AMS Router-Speicher Einstellung

5.6.8 ST_BACnet_ServerStatistics

Struktur mit Informationen zum BACnet-Server.

```
nOpenRequests       : DWORD; (* ULONG *)
nRequestRetries     : DWORD; (* ULONG *)
nRequestTimeOuts    : DWORD; (* ULONG *)
nPropChanges    : DWORD; (* ULONG *)
nPersistWrites      : DWORD; (* ULONG *)
persistChangeSetId  : DWORD; (* ULONG *)
secpersistNextWrite : DWORD; (* ULONG *)
persistLastWrite    : ST_BACnet_DateTime;
```

nOpenRequests: Anzahl offener Anfragen.

nRequestRetries: Anzahl wiederholter Anfragen.

nRequestTimeOuts: Anzahl fehlgeschlagener Anfragen.

nPropChanges: Anzahl Wertänderung von Properties (Present_Value, Status_Flags... etc.).

nPersistWrites: Anzahl Schreibzyklen der persistenten Daten (Häufigkeit seit dem Start des Systems).

persistChangeSetId: Absolute Anzahl Schreibzyklen seit dem Laden des Systems (Neustart sicher).

secpersistNextWrite: Sekunden bis zur nächsten Sicherung der persistenten Daten.

persistLastWrite: Zeitpunkt der letzten Sicherung der persistenten Daten.

5.6.9 ST_BACnet_TcIoEthStatistic

Struktur mit Informationen zum Senden und Empfangen von Netzwerk Frames aus dem Netzwerktreiber.

```
dcTimeStamp     : ARRAY[0..1] OF DWORD; (* ULONGLONG *)
sendFrames      : DWORD; (* ULONG *)
recvFrames      : DWORD; (* ULONG *)
sendTimeRTime   : DWORD; (* ULONG *)
```

```
recvTimeRTime  : DWORD; (* ULONG *)
sendTimeNdis   : DWORD; (* ULONG *)
recvTimeNdis   : DWORD; (* ULONG *)
```

dcTimeStamp: Echtzeit-Zeitstempel (DC time).

sendFrames: Gesendete Ethernet Frames des BACnet-Adapters.

recvFrames: Verarbeitete Ethernet Frames des BACnet-Adapters.

sendTimeRTime: nicht-BACnet spezifische Information aus dem Netzwerktreiber (interne Diagnose).

recvTimeRTime: nicht-BACnet spezifische Information aus dem Netzwerktreiber (interne Diagnose).

sendTimeNdis: nicht-BACnet spezifische Information aus dem Netzwerktreiber (interne Diagnose).

recvTimeNdis: nicht-BACnet spezifische Information aus dem Netzwerktreiber (interne Diagnose).

5.6.10 ST_BACnet_TcIoEthTxRxErrorCount

Struktur mit Informationen zu fehlerhaften Netzwerk Frames aus dem Netzwerktreiber.

```
txCnt    : DWORD; (* ULONG *)
rxCnt    : DWORD; (* ULONG *)
```

txCnt: Anzahl fehlerhaft gesendeter Frames (verlorene Sendepakete).

rxCnt: Anzahl fehlerhaft empfangener Frames (verworfene Empfangspakete).

5.6.11 ST_BACnet_UnConfirmedServiceDiag

Struktur mit Informationen zu unbestätigten BACnet Diensten (unconfirmed).

```
nReqSend        : DWORD; (* ULONG *)
nReqSendFail    : DWORD; (* ULONG *)
nReqRecv        : DWORD; (* ULONG *)
nReqRecvFail    : DWORD; (* ULONG *)
```

nReqSend: Anzahl an gesendeten Anfragen (unconfirmed requests).

nReqSendFail: Anzahl an fehlgeschlagenen Anfragen.

nReqRecv: Anzahl an empfangenen Anfragen (unconfirmed requests).

nReqRecvFail: Anzahl an Anfragen die nicht verarbeitet werden konnten (kein freier Speicher, fehlerhafte Codierung etc.).

5.6.12 ST_BACnet_ProgramHandshakeRequests

Struktur mit Programm-Request-Flags:

```
bLoad   : BOOL;
bSetup  : BOOL;
bRun    : BOOL;
bHalt   : BOOL;
bUnload : BOOL;
```

bLoad: Fordere das Laden eines Programms an.

bSetup: Fordere das Einrichten eines Programms an.

bRun: Fordere das Starten eines Programms an.

bHalt: Fordere das Stoppen eines Programms an.

bUnload: Fordere das Stoppen und Entladen des Programms an.

5.6.13 ST_BACnet_ProgramHandshakeStates

Struktur mit Programm-Handshake-Flags:

```
bIdle      : BOOL;
bLoading   : BOOL;
bRunning   : BOOL;
bWaiting   : BOOL;
bHalted    : BOOL;
bUnloading : BOOL;
```

bIdle: Programm ist entladen.

bLoading: Programm wird geladen.

bRunning: Programm wurde gestartet.

bWaiting: Programm wartet auf eine externe Bedingung.

bHalted: Programm wurde angehalten (gestoppt).

bUnloading: Programm wird entladen.

5.6.14 ST_BACnet_Time

Struktur mit Uhrzeitinformationen.

```
nHour       : BYTE; (* hour (0..23), (X'FF') = unspecified *)
nMinute     : BYTE; (* minute (0..59), (X'FF') = unspecified *)
nSecond     : BYTE; (* (0..59), (X'FF') = unspecified *)
nHundredths : BYTE; (* hundredths (0..99), (X'FF') = unspecified *)
```

nHour: Stunde (0..23, 0xFF = nicht spezifiziert).

nMinute: Minute (0..59, 0xFF = nicht spezifiziert).

nSecond: Sekunde (0..59, 0xFF = nicht spezifiziert).

nHundredths: Hundertstel Sekunden (0..99, 0xFF = nicht spezifiziert).

5.6.15 ST_BACnet_Diagnosis

Struktur mit Informationen aus dem BACnet Geräte Adapter. Enthalten sind Zeitverhalten, Statistiken für Server und Client und Diagnoseinformationen zum Netzwerkverkehr.

```
ethDevice         : ST_BACnet_DiagEthStatistics;
execTiming        : ST_BACnet_DiagnosisTiming;
frameStatistics   : ST_BACnet_FrameStatistics;
nRes1             : DWORD;
info              : ST_BACnet_Info;
nRes2             : DWORD;
serverInfo        : ST_BACnet_ServerStatistics;
lastUpdate        : ST_BACnet_DateTime;
nRes3             : DWORD;
```

lastUpdate: Zeitstempel der Diagnosedaten (Zeitpunkt, an dem die Daten gelesen wurden).

5.7 BACnet Enumerations

5.7.1 E_BACnetAdapterStatus

```
BACnetAdapterStatus_Init := 16#0,
BACnetAdapterStatus_CheckIpAddr := 16#1,
BACnetAdapterStatus_CheckParam := 16#2,
BACnetAdapterStatus_GetGatewayMAC := 16#3,
BACnetAdapterStatus_WaitGatewayMAC := 16#4,
BACnetAdapterStatus_Complete := 16#8
```

5.7.2 E_BACnetBinaryPV

```
BACnetBinaryPV_inactive :=0,
BACnetBinaryPV_active :=1,
BACnetBinaryPV_NULL := 2
```

5.7.3 E_BACnetDataTypes

```
BACnetDataType_Null :=0,
BACnetDataType_Bool :=1,
BACnetDataType_UnsignedInteger :=2,
BACnetDataType_SignedInteger :=3,
BACnetDataType_Real :=4,
BACnetDataType_Double :=5,
BACnetDataType_OctetString :=6,
BACnetDataType_CharacterStringExt :=7,
BACnetDataType_BitString :=8,
BACnetDataType_Enumerated :=9,
BACnetDataType_Date :=10,
BACnetDataType_Time :=11,
BACnetDataType_ObjectIdentifier :=12,
BACnetDataType_DateTime :=14,
BACnetDataType_ArrayIndex :=15,
BACnetDataType_CalendarEntry :=16,
BACnetDataType_ObjectType :=17,
BACnetDataType_EventTransitionBits :=18,
BACnetDataType_ActionList :=19,
BACnetDataType_StatusFlags :=20,
BACnetDataType_EventState :=21,
BACnetDataType_Reliability :=22,
BACnetDataType_Polarity :=23,
BACnetDataType_EngineeringUnits :=25,
BACnetDataType_PriorityValue :=26,
BACnetDataType_LimitEnable :=27,
BACnetDataType_NotifyType :=29,
BACnetDataType_TimeStamp :=30,
BACnetDataType_DeviceObjectPropertyReference :=31,
BACnetDataType_DeviceStatus :=32,
BACnetDataType_ServicesSupported :=33,
BACnetDataType_ObjectTypesSupported :=34,
BACnetDataType_Segmentation :=35,
BACnetDataType_VTClass :=36,
BACnetDataType_VTSession :=37,
BACnetDataType_SessionKey :=38,
BACnetDataType_Recipient :=39,
BACnetDataType_AddressBinding :=40,
BACnetDataType_COVSubscription :=41,
BACnetDataType_EventType :=42,
BACnetDataType_EventParameter :=43,
BACnetDataType_FileAccessMethod :=44,
BACnetDataType_ReadAccessSpecification :=45,
BACnetDataType_ReadAccessResult :=46,
BACnetDataType_ObjectPropertyReference :=47,
BACnetDataType_SetpointReference :=48,
BACnetDataType_Destination :=49,
BACnetDataType_ProgramState :=50,
BACnetDataType_ProgramRequest :=51,
BACnetDataType_ProgramError :=52,
BACnetDataType_DateRange :=53,
BACnetDataType_DailySchedule :=54,
BACnetDataType_SpecialEvent :=55,
BACnetDataType_ClientCOV :=56,
BACnetDataType_LogRecord :=57,
BACnetDataType_LifeSafetyState :=58,
BACnetDataType_LifeSafetyMode :=59,
BACnetDataType_SilencedState :=60,
BACnetDataType_LifeSafetyOperation :=61,
BACnetDataType_BinaryPV :=62,
BACnetDataType_DaysOfWeek :=63,
BACnetDataType_WeekNDay :=64,
BACnetDataType_TimeValue :=65,
BACnetDataType_Value :=66,
BACnetDataType_Address :=67,
BACnetDataType_PropertyIdentifier :=68,
BACnetDataType_EnumerationValueType :=69,
BACnetDataType_BitFieldBit :=70,
BACnetDataType_LogDatum :=71,
BACnetDataType_AnyType :=72,
```

```
BACnetDataType_PresentValue :=73,
BACnetDataType_ContextTag :=74,
BACnetDataType_ValueChoice :=75,
BACnetDataType_LogStatus :=76,
BACnetDataType_RecipientProcess :=77,
BACnetDataType_SubscribeCOVPropertyRequest :=78,
BACnetDataType_PropertyReference :=79,
BACnetDataType_PropertyResult :=80,
BACnetDataType_DeviceObjectPropertyValue :=81,
BACnetDataType_COVNotificationRequest :=82,
BACnetDataType_EventNotificationRequest :=83,
BACnetDataType_NotificationParameters :=84,
BACnetDataType_Change_of_Bitstring :=85,
BACnetDataType_Change_of_State :=86,
BACnetDataType_Change_of_Value :=87,
BACnetDataType_Command_failure :=88,
BACnetDataType_Floating_limit :=89,
BACnetDataType_Out_of_Range :=90,
BACnetDataType_PropertyValue :=91,
BACnetDataType_Complex_Tag :=92,
BACnetDataType_Change_of_life_safety :=93,
BACnetDataType_Extended :=94,
BACnetDataType_Buffer_ready :=95,
BACnetDataType_Unsigned_range :=96,
BACnetDataType_PropertyResultValue :=97,
BACnetDataType_ServiceCOVPropSubscription :=98,
BACnetDataType_ServiceCOVSubscription :=99,
BACnetDataType_COVNotification :=100,
BACnetDataType_AcknowledgeAlarmRequest :=101,
BACnetDataType_VTOpenRequest :=102,
BACnetDataType_Prescale :=103,
BACnetDataType_Scale :=104,
BACnetDataType_Action :=105,
BACnetDataType_Error :=106,
BACnetDataType_ErrorType :=107,
BACnetDataType_ErrorClass :=108,
BACnetDataType_ErrorCode :=109,
BACnetDataType_RejectReason :=110,
BACnetDataType_AbortReason :=111,
BACnetDataType_BDTEntry :=112,
BACnetDataType_IpEntry :=113,
BACnetDataType_MaskEntry :=114,
BACnetDataType_UnsignedIntegerArr :=258,
BACnetDataType_CharacterStringExtArr :=263,
BACnetDataType_ObjectIdentifierArr :=268,
BACnetDataType_CalendarEntryArr :=272,
BACnetDataType_ActionListArr :=275,
BACnetDataType_PriorityValueArr :=282,
BACnetDataType_TimeStampArr :=286,
BACnetDataType_DeviceObjectPropertyReferenceArr :=287,
BACnetDataType_VTClassArr :=292,
BACnetDataType_VTSessionArr :=293,
BACnetDataType_SessionKeyArr :=294,
BACnetDataType_RecipientArr :=295,
BACnetDataType_AddressBindingArr :=296,
BACnetDataType_COVSubscriptionArr :=297,
BACnetDataType_ReadAccessSpecificationArr :=301,
BACnetDataType_ReadAccessResultArr :=302,
BACnetDataType_DestinationArr :=305,
BACnetDataType_SpecialEventArr :=311,
BACnetDataType_LogRecordArr :=313,
BACnetDataType_LifeSafetyStateArr :=314,
BACnetDataType_TimeValueArr :=321,
BACnetDataType_UnsignedIntegerList :=514,
BACnetDataType_CharacterStringExtList :=519,
BACnetDataType_ReadAccessSpecificationList :=557,
BACnetDataType_ReadAccessResultList :=558,
BACnetDataType_SpecialEventList :=567,
BACnetDataType_LogRecordList :=569,
BACnetDataType_TimeValueList :=577,
BACnetDataType_ValueList :=578,
BACnetDataType_LogDatumList :=583,
BACnetDataType_PropertyReferenceList :=591,
BACnetDataType_PropertyResultList :=592,
BACnetDataType_COVNotificationRequestList :=594,
BACnetDataType_EventNotificationRequestList :=595,
BACnetDataType_PropertyValueList :=603,
BACnetDataType_DailyScheduleList :=1089
```

5.7.4 E_BACnetDeviceStatus

```
BACnetDeviceStatus_Operational :=0,
BACnetDeviceStatus_OperationalReadOnly :=1,
BACnetDeviceStatus_DownloadRequired :=2,
BACnetDeviceStatus_DownloadInProgress :=3,
BACnetDeviceStatus_NonOperational :=4,
BACnetDeviceStatus_BackupInProgress :=5
```

5.7.5 E_BACnetEventState

```
BACnetEventState_state_normal :=0,
BACnetEventState_fault :=1,
BACnetEventState_offnormal :=2,
BACnetEventState_high_limit :=3,
BACnetEventState_low_limit :=4,
BACnetEventState_life_safety_alarm :=5
```

5.7.6 E_BACnetObjectType

```
BACnetObjectType_AnalogInput :=0,
BACnetObjectType_AnalogOutput :=1,
BACnetObjectType_AnalogValue :=2,
BACnetObjectType_BinaryInput :=3,
BACnetObjectType_BinaryOutput :=4,
BACnetObjectType_BinaryValue :=5,
BACnetObjectType_Calendar :=6,
BACnetObjectType_Command :=7,
BACnetObjectType_Device :=8,
BACnetObjectType_EventEnrollment :=9,
BACnetObjectType_File :=10,
BACnetObjectType_Group :=11,
BACnetObjectType_Loop :=12,
BACnetObjectType_MultiStateInput :=13,
BACnetObjectType_MultiStateOutput :=14,
BACnetObjectType_NotificationClass :=15,
BACnetObjectType_Program :=16,
BACnetObjectType_Schedule :=17,
BACnetObjectType_Averaging :=18,
BACnetObjectType_MultiStateValue :=19,
BACnetObjectType_TrendLog :=20,
BACnetObjectType_LifeSafetyPoint :=21,
BACnetObjectType_LifeSafetyZone :=22,
BACnetObjectType_Accumulator :=23,
BACnetObjectType_PulseConverter :=24
```

5.7.7 E_BACnetPriority

```
BACnetPriority_None :=0,
BACnetPriority_1    :=1,
BACnetPriority_2    :=2,
BACnetPriority_3    :=3,
BACnetPriority_4    :=4,
BACnetPriority_5    :=5,
BACnetPriority_6    :=6,
BACnetPriority_7    :=7,
BACnetPriority_8    :=8,
BACnetPriority_9    :=9,
BACnetPriority_10   :=10,
BACnetPriority_11   :=11,
BACnetPriority_12   :=12,
BACnetPriority_13   :=13,
BACnetPriority_14   :=14,
BACnetPriority_15   :=15,
BACnetPriority_16   :=16
```

5.7.8 E_BACnetProgramError

Enumeration mit Fehlerrückmeldungen des Programm-Objekts (Program-Error):

```
BACnetProgramError_program_normal :=0,
BACnetProgramError_load_failed :=1,
BACnetProgramError_progerror_internal :=2,
BACnetProgramError_program :=3,
BACnetProgramError_other :=4
```

BACnetProgramError_program_normal: Kein Fehler.

BACnetProgramError_load_failed: Laden des Programmes fehlgeschlagen.

BACnetProgramError_progerror_internal: Ein interner Programmfehler ist aufgetreten.

BACnetProgramError_program: Fehler bei der Programmausführung.

BACnetProgramError_other: Unbekannter Fehler.

5.7.9 E_BACnetProgramRequest

Enumeration mit Anforderungen an das Programm-Objekt (Program-Requests):

```
BACnetProgramRequest_ready :=0,
BACnetProgramRequest_load :=1,
BACnetProgramRequest_run :=2,
BACnetProgramRequest_halt :=3,
BACnetProgramRequest_restart :=4,
BACnetProgramRequest_unload :=5
```

BACnetProgramRequest_ready: Das Programm ist bereit für eine Anforderung bzw. hat die letzte Anforderung ausgeführt/abgebrochen.

BACnetProgramRequest_load: Fordere das Laden des Programms an.

BACnetProgramRequest_run: Fordere das Starten des Programmes an.

BACnetProgramRequest_halt: Fordere das Stoppen eines Programmes an.

BACnetProgramRequest_restart: Fordere das Neustarten des Programmes an.

BACnetProgramRequest_unload: Fordere das Stoppen und Entladen des Programmes an.

5.7.10 E_BACnetProgramState

Enumeration mit dem Status des Programm-Objekts (Program-State):

```
BACnetProgramState_idle :=0,
BACnetProgramState_loading :=1,
BACnetProgramState_running :=2,
BACnetProgramState_waiting :=3,
BACnetProgramState_halted :=4,
BACnetProgramState_unloading :=5
```

BACnetProgramState_idle: Programm ist entladen.

BACnetProgramState_loading: Programm wird geladen.

BACnetProgramState_running: Programm wurde gestartet.

BACnetProgramState_waiting: Programm wartet auf eine externe Bedingung.

BACnetProgramState_halted: Programm wurde angehalten (gestoppt).

BACnetProgramState_unloading: Programm wird entladen.

5.7.11 E_BACnetPropertyIdentifier

```
BACnetPropertyIdentifier_AckedTransitions :=0,
BACnetPropertyIdentifier_AckRequired :=1,
BACnetPropertyIdentifier_Action :=2,
BACnetPropertyIdentifier_ActionText :=3,
BACnetPropertyIdentifier_ActiveText :=4,
BACnetPropertyIdentifier_ActiveVtSessions :=5,
BACnetPropertyIdentifier_AlarmValue :=6,
```

```
BACnetPropertyIdentifier_AlarmValues :=7,
BACnetPropertyIdentifier_All :=8,
BACnetPropertyIdentifier_AllWritesSuccessful :=9,
BACnetPropertyIdentifier_ApduSegmentTimeout :=10,
BACnetPropertyIdentifier_ApduTimeout :=11,
BACnetPropertyIdentifier_ApplicationSoftwareVersion :=12,
BACnetPropertyIdentifier_Archive :=13,
BACnetPropertyIdentifier_Bias :=14,
BACnetPropertyIdentifier_ChangeOfStateCount :=15,
BACnetPropertyIdentifier_ChangeOfStateTime :=16,
BACnetPropertyIdentifier_NotificationClass :=17,
BACnetPropertyIdentifier_ControlledVariableReference :=19,
BACnetPropertyIdentifier_ControlledVariableUnits :=20,
BACnetPropertyIdentifier_ControlledVariableValue :=21,
BACnetPropertyIdentifier_CovIncrement :=22,
BACnetPropertyIdentifier_Datelist :=23,
BACnetPropertyIdentifier_DaylightSavingsStatus :=24,
BACnetPropertyIdentifier_Deadband :=25,
BACnetPropertyIdentifier_DerivativeConstant :=26,
BACnetPropertyIdentifier_DerivativeConstantUnits :=27,
BACnetPropertyIdentifier_Description :=28,
BACnetPropertyIdentifier_DescriptionOfHalt :=29,
BACnetPropertyIdentifier_DeviceAddressBinding :=30,
BACnetPropertyIdentifier_DeviceType :=31,
BACnetPropertyIdentifier_EffectivePeriod :=32,
BACnetPropertyIdentifier_ElapsedActiveTime :=33,
BACnetPropertyIdentifier_ErrorLimit :=34,
BACnetPropertyIdentifier_EventEnable :=35,
BACnetPropertyIdentifier_EventState :=36,
BACnetPropertyIdentifier_EventType :=37,
BACnetPropertyIdentifier_ExceptionSchedule :=38,
BACnetPropertyIdentifier_FaultValues :=39,
BACnetPropertyIdentifier_FeedbackValue :=40,
BACnetPropertyIdentifier_FileAccessMethod :=41,
BACnetPropertyIdentifier_FileSize :=42,
BACnetPropertyIdentifier_FileType :=43,
BACnetPropertyIdentifier_FirmwareRevision :=44,
BACnetPropertyIdentifier_HighLimit :=45,
BACnetPropertyIdentifier_InactiveText :=46,
BACnetPropertyIdentifier_InProcess :=47,
BACnetPropertyIdentifier_InstanceOf :=48,
BACnetPropertyIdentifier_IntegralConstant :=49,
BACnetPropertyIdentifier_IntegralConstantUnits :=50,
BACnetPropertyIdentifier_IssueConfirmedNotifications :=51,
BACnetPropertyIdentifier_LimitEnable :=52,
BACnetPropertyIdentifier_ListOfGroupMembers :=53,
BACnetPropertyIdentifier_ListOfObjectPropertyReferences :=54,
BACnetPropertyIdentifier_ListOfSessionKeys :=55,
BACnetPropertyIdentifier_LocalDate :=56,
BACnetPropertyIdentifier_LocalTime :=57,
BACnetPropertyIdentifier_Location :=58,
BACnetPropertyIdentifier_LowLimit :=59,
BACnetPropertyIdentifier_ManipulatedVariableReference :=60,
BACnetPropertyIdentifier_MaximumOutput :=61,
BACnetPropertyIdentifier_MaxApduLengthAccepted :=62,
BACnetPropertyIdentifier_MaxInfoFrames :=63,
BACnetPropertyIdentifier_MaxMaster :=64,
BACnetPropertyIdentifier_MaxPresValue :=65,
BACnetPropertyIdentifier_MinimumOfftime :=66,
BACnetPropertyIdentifier_MinimumOntime :=67,
BACnetPropertyIdentifier_MinimumOutput :=68,
BACnetPropertyIdentifier_MinPresValue :=69,
BACnetPropertyIdentifier_ModelName :=70,
BACnetPropertyIdentifier_ModificationDate :=71,
BACnetPropertyIdentifier_NotifyType :=72,
BACnetPropertyIdentifier_NumberOfAPDURetries :=73,
BACnetPropertyIdentifier_NumberOfStates :=74,
BACnetPropertyIdentifier_ObjectIdentifier :=75,
BACnetPropertyIdentifier_ObjectList :=76,
BACnetPropertyIdentifier_ObjectName :=77,
BACnetPropertyIdentifier_ObjectReference :=78,
BACnetPropertyIdentifier_ObjectType :=79,
BACnetPropertyIdentifier_Optional :=80,
BACnetPropertyIdentifier_OutOfService :=81,
BACnetPropertyIdentifier_OutputUnits :=82,
BACnetPropertyIdentifier_EventParameters :=83,
BACnetPropertyIdentifier_Polarity :=84,
BACnetPropertyIdentifier_PresentValue :=85,
BACnetPropertyIdentifier_Priority :=86,
BACnetPropertyIdentifier_PriorityArray :=87,
```

```
BACnetPropertyIdentifier_PriorityForWriting :=88,
BACnetPropertyIdentifier_ProcessIdentifier :=89,
BACnetPropertyIdentifier_ProgramChange :=90,
BACnetPropertyIdentifier_ProgramLocation :=91,
BACnetPropertyIdentifier_ProgramState :=92,
BACnetPropertyIdentifier_ProportionalConstant :=93,
BACnetPropertyIdentifier_ProportionalConstantUnits :=94,
BACnetPropertyIdentifier_ProtocolObjectTypesSupported :=96,
BACnetPropertyIdentifier_ProtocolServicesSupported :=97,
BACnetPropertyIdentifier_ProtocolVersion :=98,
BACnetPropertyIdentifier_ReadOnly :=99,
BACnetPropertyIdentifier_ReasonForHalt :=100,
BACnetPropertyIdentifier_Recipient :=101,
BACnetPropertyIdentifier_RecipientList :=102,
BACnetPropertyIdentifier_Reliability :=103,
BACnetPropertyIdentifier_RelinquishDefault :=104,
BACnetPropertyIdentifier_Required :=105,
BACnetPropertyIdentifier_Resolution :=106,
BACnetPropertyIdentifier_SegmentationSupported :=107,
BACnetPropertyIdentifier_Setpoint :=108,
BACnetPropertyIdentifier_SetpointReference :=109,
BACnetPropertyIdentifier_StateText :=110,
BACnetPropertyIdentifier_StatusFlags :=111,
BACnetPropertyIdentifier_SystemStatus :=112,
BACnetPropertyIdentifier_TimeDelay :=113,
BACnetPropertyIdentifier_TimeOfActiveTimeReset :=114,
BACnetPropertyIdentifier_TimeOfStateCountReset :=115,
BACnetPropertyIdentifier_TimeSynchronizationRecipients :=116,
BACnetPropertyIdentifier_Units :=117,
BACnetPropertyIdentifier_UpdateInterval :=118,
BACnetPropertyIdentifier_UtcOffset :=119,
BACnetPropertyIdentifier_VendorIdentifier :=120,
BACnetPropertyIdentifier_VendorName :=121,
BACnetPropertyIdentifier_VtClassesSupported :=122,
BACnetPropertyIdentifier_WeeklySchedule :=123,
BACnetPropertyIdentifier_AttemptedSamples :=124,
BACnetPropertyIdentifier_AverageValue :=125,
BACnetPropertyIdentifier_BufferSize :=126,
BACnetPropertyIdentifier_ClientCovIncrement :=127,
BACnetPropertyIdentifier_CovResubscriptionInterval :=128,
BACnetPropertyIdentifier_EventTimeStamps :=130,
BACnetPropertyIdentifier_LogBuffer :=131,
BACnetPropertyIdentifier_LogDeviceObjectProperty :=132,
BACnetPropertyIdentifier_LogEnable :=133,
BACnetPropertyIdentifier_LogInterval :=134,
BACnetPropertyIdentifier_MaximumValue :=135,
BACnetPropertyIdentifier_MinimumValue :=136,
BACnetPropertyIdentifier_NotificationThreshold :=137,
BACnetPropertyIdentifier_ProtocolRevision :=139,
BACnetPropertyIdentifier_RecordsSinceNotification :=140,
BACnetPropertyIdentifier_RecordCount :=141,
BACnetPropertyIdentifier_StartTime :=142,
BACnetPropertyIdentifier_StopTime :=143,
BACnetPropertyIdentifier_StopWhenFull :=144,
BACnetPropertyIdentifier_TotalRecordCount :=145,
BACnetPropertyIdentifier_ValidSamples :=146,
BACnetPropertyIdentifier_WindowInterval :=147,
BACnetPropertyIdentifier_WindowSamples :=148,
BACnetPropertyIdentifier_MaximumValueTimestamp :=149,
BACnetPropertyIdentifier_MinimumValueTimestamp :=150,
BACnetPropertyIdentifier_VarianceValue :=151,
BACnetPropertyIdentifier_ActiveCovSubscriptions :=152,
BACnetPropertyIdentifier_BackupFailureTimeout :=153,
BACnetPropertyIdentifier_ConfigurationFiles :=154,
BACnetPropertyIdentifier_DatabaseRevision :=155,
BACnetPropertyIdentifier_DirectReading :=156,
BACnetPropertyIdentifier_LastRestoreTime :=157,
BACnetPropertyIdentifier_MaintenanceRequired :=158,
BACnetPropertyIdentifier_MemberOf :=159,
BACnetPropertyIdentifier_Mode :=160,
BACnetPropertyIdentifier_OperationExpected :=161,
BACnetPropertyIdentifier_Setting :=162,
BACnetPropertyIdentifier_Silenced :=163,
BACnetPropertyIdentifier_TrackingValue :=164,
BACnetPropertyIdentifier_ZoneMembers :=165,
BACnetPropertyIdentifier_LifeSafetyAlarmValues :=166,
BACnetPropertyIdentifier_MaxSegmentsAccepted :=167,
BACnetPropertyIdentifier_ProfileName :=168,
BACnetPropertyIdentifier_InternalUse :=169,
BACnetPropertyIdentifier_LastNotifyRecord :=173,
```

```
BACnetPropertyIdentifier_ScheduleDefault :=174,
BACnetPropertyIdentifier_AcceptedModes :=175,
BACnetPropertyIdentifier_AdjustValue :=176,
BACnetPropertyIdentifier_Count :=177,
BACnetPropertyIdentifier_CountBeforeChange :=178,
BACnetPropertyIdentifier_CovPeriod :=180,
BACnetPropertyIdentifier_InputReference :=181,
BACnetPropertyIdentifier_LimitMonitoringInverval :=182,
BACnetPropertyIdentifier_LoggingObject :=183,
BACnetPropertyIdentifier_LoggingRecord :=184,
BACnetPropertyIdentifier_Prescale :=185,
BACnetPropertyIdentifier_PulseRate :=186,
BACnetPropertyIdentifier_Scale :=187,
BACnetPropertyIdentifier_ScaleFactor :=188,
BACnetPropertyIdentifier_UpdateTime :=189,
BACnetPropertyIdentifier_ValueBeforeChange :=190,
BACnetPropertyIdentifier_ValueSet :=191,
BACnetPropertyIdentifier_ValueChangeTime :=192
```

5.8 BACnet_Globals

```
VAR_GLOBAL
(* global linkable BACnet adapter *)
    fbBACnet_Adapter : FB_BACnet_Adapter;
(* global visible BACnet adapter reference *)
    pBACnet_Adapter : POINTER TO FB_BACnet_Adapter;
END_VAR
VAR_GLOBAL CONSTANT
    tBACnet_ADSTimeOut : TIME := t#5s;
END_VAR
```

6 Anhang

6.1 ADS-Interface

Sämtliche azyklischen Daten werden mit ADS-Read, ADS-Write, ADS-ReadWrite oder ADS-Write-Control an ein BACnet-Device, eine Notification Sink, einen BACnet-Server bzw. einen BACnet-Client übertragen. Ein BACnet-Device hat eine eigene Net-ID, die über den Reiter "Settings" angezeigt wird und unterstützt die folgenden Ports:

Port	Beschreibung
0xFFFF	damit wird das BACnet-Device selbst adressiert, d.h. Daten die nicht spezifisch für einen BACnet-Server, eine Notification Sink oder einen BACnet-Client sind
1000 - 8192	damit wird ein BACnet-Server, eine Notification Sink oder BACnet-Client adressiert, wobei z.B. der erste angelegte BACnet-Server/Client über den Port 1000 erreicht werden kann, der 2. BACnet-Client über Port 1001. Das Löschen eines BACnet-Server/Client führt nicht zum Verschieben der Ports, die Zuordnung der ADS-Ports kann also nicht zwangsläufig an der Reihenfolge im System-Manager abgeleitet werden. Der jeweilige Port kann über den Reiter "Settings" eines BACnet-Server/Client eingesehen werden.

IndexGroup/IndexOffset bei lokaler Adressierung des BACnet-Device (Port 0xFFFF)

ADS-Read

Es folgt eine Übersicht der von einem BACnet-Device unterstützten IndexGroup/IndexOffset bei ADS-Read.

IndexGroup (Lo-Word)	IndexOffset	Beschreibung
0x80000006	immer 0	Starten eines Scans nach anderen BACnet-Geräten im Netzwerk.
0x80000008	immer 0	Stoppen eines Scans nach anderen BACnet-Geräten im Netzwerk. Vorbereitung auf das Auslesen der Ergebnisliste.
0x80000007	0 : Anzahl der Einträge in der Ergebnisliste (Anzahl der Clients*2) 2*n : IP-Adresse von Gerät n 2*n+1: Device-ID von Gerät n 0xffffffff	Ein Scan-Vorgang mit Hilfe des BACnet-Device erfolgt nach dem Schema: 1. Starten des Scans 2. Abwarten eines beliebigen Timeouts 3. Stoppen des Scans 4. Lesen der Scan-Ergebnisliste Das Auslesen der Scan-Ergebnisse erfolgt über diese IndexGruppe. Über den IndexOffset 0 kann die Anzahl der Einträge in der Ergebnisliste ermittelt werden. Diese entspricht der doppelten Anzahl der gefundenen BACnet-Geräte im Netzwerk. Gerade IndexOffset > 0 adressieren die IP-Adresse eines Geräts; ungerade IndexOffset adressieren die Device-ID des Geräts. Vor dem Auslesen der Ergebnisliste muss der Scan-Vorgang gestoppt werden. Werden z.B. in einem Netzwerk 2 Geräte gefunden, ergibt die Abfrage von Index 0 eine 4. Über IndexOffset 1 die Device-ID von Gerät 1, IndexOffset die IP-Adresse von Gerät 1, IndexOffset 3 die Device-ID von Gerät 2 und IndexOffset 4 die IP-Adresse von Gerät 2. Auslesen der im Server vorhandenen Objekte. Es wird ein Array vom Typ BACnetScanResult mit der Anzahl der gefundenen Clients zurückgegeben: enum BACnetAddressChoice { Ethernet,Arcnet,MsTp,LonTalk,LonTalkNeuronId, Address_3,Address_4,Address_5,Broadcast } struct BACnetAddress { BACnetAddressChoice choice; BYTE layerAddress[7]; // MS/TP...LonTalk BYTE reserved; USHORT network_number; }

IndexGroup (Lo-Word)	IndexOffset	Beschreibung
		structBACnetScanResult { BACnetObjectIdentifier deviceObjIdentifier; BACnetAddress netAddress; ULONG IPadress; WORD UdpPort; WORD vendorId; BYTE ethernetMacAddr[6]; }
0x8000000A	immer 0	Starten eines Backup. (Auslösen des BACnet-Dienstes Backup Start)
0x8000001A	immer 0	Beenden eines Backup. (Auslösen des BACnet-Dienstes Backup End) Zwischen Backup-Start und -End können mit Hilfe der Property ConfigurationFiles des Device-Objekts und des ADS-File-Dienstes die erzeugten Backup-Dateien gesichert werden.
0x8000002A	immer 0	Starten eines Restore. (Auslösen des BACnet-Dienstes Restore Start)
0x8000003A	immer 0	Beenden eines Restore (Auslösen des BACnet-Dienstes Restore End)
0x8000004A	immer 0	Abbruch eines Restore (Auslösen des BACnet-Dienstes Restore Abort)
0x8000005A	immer 0	Neustart des TwinCAT-Rechners (Auslösen des BACnet-Dienstes Cold Start)
0x8000006A	immer 0	Restart von TwinCAT (Auslösen des BACnet-Dienstes Warm Start)
0x8000000C	immer 0	Auslesen des Supplement-Key Status: • 0x0000000 - Supplement-Key ist nicht gültig • 0x0000001 - Supplement-Key ist gültig
0x8000000D	immer 0	Auslesen der aktuellen Foreign Device Table (FDT) als Array vom Typ FDTEntry: struct FDTEntry { ULONG ipAddr; USHORT udpPort; USHORT nTtlSecs; USHORT nPurgeSecs; // seconds until entry is deleted USHORT reserved; // alignment }
0x8000000E	immer 0	Auslesen der aktuellen Broadcast Distribution Table (BDT) als Array vom Typ BDTEntry: struct BDTEntry { ULONG ipAddr; ULONG mask; USHORT udpPort; USHORT reserved; // alignment }
0x8000000F	immer 0	Auslesen der BACnet-Diagnose der Form BACnetDiagnosis. (Siehe Reiter "Diagnosis" vom BACnet-Device)
0x8000001F	immer 0	Auslesen der Advanced Parameter in der Form Advanced Parameters struct AdvancedParameters { ULONG ap_MAX_SUBSCRIBE_COV_ENTRIES; ULONG ap_MAX_PROPERTY_ELEMENTS; ULONG ap_LIST_ALLOC_MEM_BYTES; ULONG ap_MAX_DEVICE_BINDINGS; ULONG ap_HASHTABLE_MAX_OBJECTS; ULONG ap_HASHTABLE_MAX_CLIENTS; ULONG ap_MAX_TASK_LIST_ENTRIES; ULONG ap_MAX_TRENDLOG_ENTRYSIZE; ULONG ap_DYN_TRENDLOG_BUFFERSIZE; ULONG ap_MAX_MULTISTATE_DYNSTATES; ULONG ap_BACNET_MAX_RECV_BAC_SEGM_FRAMES; ULONG ap_BACNET_MAX_SEND_BAC_SEGM_FRAMES; ULONG ap_IO_STARTUP_TIMEOUT; }

ADS-Write

Es folgt eine Übersicht der von einem BACnet-Device unterstützten IndexGroup/IndexOffset bei ADS-Write.

IndexGroup	IndexOffset	Beschreibung
0x8000002D	immer 0	Auslösen einer globalen Broadcast UTC-Synchronisation. Als Daten muss ein gültiger BACnetDateTime-Zeitstempel übergeben werden.

IndexGroup	IndexOffset	Beschreibung
		struct BACnetTime { BYTE hour; BYTE minute; BYTE second; BYTE hundredths; } structBACnetDate { BYTE year; BYTE month; BYTE day; BYTE dayOfWeek; } structBACnetDateTime { BACnetDate date; BACnetTime time; }
0x8000000F	immer 0	BACnet Diagnose Steuerung: Übergebene Datengröße 4: • 0x00000000 : Zeitmessung deaktivieren • 0x00000001 : Zeitmessung aktivieren Datengröße ungleich 4: Zurücksetzen der BACnet Diagnose
0x8000000E	immer 0	Schreiben der Broadcast Distribution Table (BDT) als Array der Form BDTEntry: struct BDTEntry { ULONG ipAddr; ULONG mask; USHORT udpPort; USHORT reserved; // alignment }

IndexGroup/IndexOffset bei Adressierung eines BACnet-Server (Port 1000-8191)

ADS-Read

IndexGroup	IndexOffset	Beschreibung
0x82400000	0 Anzahl der Listeneinträge > 0 und < 0xffffffff	Auslesen der im Server vorhandenen Objekte. Die auf einem Server angelegten Objekte können zusätzlich durch das Auslesen der Property ObjectList des Device-Objekts ermittelt werden. Index 0: Anzahl der Einträge Index n*2: Objekt-Typ für Eintrag n Index n*2+1: Objekt-Instanz für Eintrag n
0x83400000	immer 0	Auslesen der Liste der dynamisch erzeugten Objekte. Pro Objekt enthält die Liste einen 4 Byte Eintrag, der die jeweilige Objekt-ID enthält.

ADS-Write

IndexGroup	IndexOffset	Beschreibung
0x80800000	immer 0	Objekt dynamisch erzeugen. Folgende Daten müssen beim Write-Request übergeben werden, um die Objekterzeugung zu parametrieren: • 4 Byte Objekttyp • 4 Byte Objekt-ID (Objekttyp der Objekt-ID muss dem Objekttyp übereinstimmen) • 4 Byte Flags (0x00000001 entferntes Objekt (bei Client) , 0x00000000 lokales Objekt (bei Server)) • 8 Byte 0x00 für eine leere Liste von initialen Parametern; bzw. eine Liste mit initialen Parametern mit dem Datentyp ReadPropertyMultipleResult
0x81000000	immer 0	Dynamisch erzeugtes Objekt löschen. Folgende Daten müssen beim Write-Request übergeben werden: • 4 Byte Objekttyp • 4 Byte Objekt-ID (Objekttyp der Objekt-ID muss mit dem Objekttyp übereinstimmen)

IndexGroup	IndexOffset	Beschreibung
		• 4 Byte Flags (0x00000001 entferntes Objekt (bei Client) , 0x00000000 lokales Objekt (bei Server)) • 8 Byte 0x00
0x80C00000	immer 0	Abspeichern der persistenten Daten auslösen. Nur gültig für BACnet-Server wenn Persist Online Data aktiv ist.

ADS-ReadWrite

IndexGroup	IndexOffset	Beschreibung
0x82C00000	Objekt-ID • Bit 0 - 21 : Objektinstanz • Bit 22-31 : Objekttyp	Auslesen der Property-Liste eines Objekts. Mit diesem Dienst werden alle Properties eines Objekts ähnlich des BACnet-Dienstes ReadPropertyMultiple zusammenfasst. Beim Aufruf des ADS-ReadWrite-Kommandos muss eine Liste mit den gewünschten Properties übergeben werden. Momentan wird nur das Auslösen der kompletten Liste unterstützt. Hierfür müssen 4 Bytes mit dem Wert 0x00000008 übergeben werden. Als Ergebnis wird eine Liste vom Typ ReadPropertyMultipleResult übergeben. Die Liste enthält jeweils einen Listen-Header, eine Property-ID gefolgt von den Daten der Property.
0x8B800000	immer 0	Abfrage der Objekt-ID mit dem Namen eines Objekts. Es muss der Name des gesuchten Objekts als ASCII-String geschrieben werden. Als Rückgabe wird die zugehörige Objekt-ID (4 Byte) zurückgegeben. Wird ein Objekt nicht gefunden wird der ADS-Fehler NOT_FOUND zurückgegeben.

IndexGroup bei Adressierung eines BACnet-Server/Client (Port 1000-8191) - Property-Zugriff

Property-spezifische ADS-Dienste können mit einer IndexGroup/IndexOffset mit folgendem Schema verwendet werden:

- Der IndexOffset enthält die Objekt-ID des gewünschten Objekts
- Bit 0-21 der IndexGroup enthält die Property-ID der gewünschten BACnet-Property
- Bit 22-30 der IndexGroup enthält den Gruppentyp, der die jeweilige Funktionalität beschreibt
- Bit 31 der IndexGroup ist immer 1 und aktiviert den Gruppentyp-Modus für die Nutzung der Property-spezifischen Dienste

ADS-Read

IndexGroup	IndexOffset	Beschreibung
0x80000000 + Property-ID Gruppentyp (Bit 22-30): 0	Objekt-ID • Bit 0 - 21 : Objektinstanz • Bit 22-31 : Objekttyp	Auslesen der aktuellen Daten einer Property. Beispiel: Die Property PresentValue des Objekts BinaryInput:42 kann via: IndexGroup: 0x80000055 und IndexOffset: 0x00C0002A ausgelesen werden.
0x80400000 + Property-ID Gruppentyp (Bit 22-30): 1	Objekt-ID • Bit 0 - 21 : Objektinstanz • Bit 22-31 : Objekttyp	Auslesen des BACnet-Datentyps einer Property. Die Liste der Datentyp-Codierung findet sich im Anhang.
0x80800000 + Property-ID Gruppentyp (Bit 22-30): 2	Objekt-ID • Bit 0 - 21 : Objektinstanz • Bit 22-31 : Objekttyp	Auslesen der Datenlänge des aktuellen Property-Wertes.

IndexGroup	IndexOffset	Beschreibung
0x80C00000 + Property-ID Gruppentyp (Bit 22-30): 3	Objekt-ID • Bit 0 - 21 : Objektinstanz • Bit 22-31 : Objekttyp	Auslesen der Schreibschutzeigenschaft einer Property. Die ausgelesenen Daten werden wie folgt interpretiert: • 0x00 Property-Wert kann beschrieben werden • 0x01 Property-Wert ist schreibgeschützt
0x81400000 + Property-ID Gruppentyp (Bit 22-30): 5	Objekt-ID • Bit 0 - 21 : Objektinstanz • Bit 22-31 : Objekttyp	Auslesen der Anzahl der Elemente von Listen- und Arraytypen.

AdsWrite

IndexGroup	IndexOffset	Beschreibung
0x80000000 + Property-ID	Objekt-ID • Bit 0 - 21 : Objektinstanz • Bit 22-31 : Objekttyp	Schreiben eines Property-Wertes. Über die Bits 22-30 der IndexGroup kann auch ein priorisierter Schreibzugriff erfolgen. Priorität 1 (höchste Priorität in BACnet) kann über den Wert 0x010 in den Bits 22-30 definiert werden. Priorität 16 (die niedrigste Priorität in BACnet) mit dem Wert 0x100.
0x84000000 + Property-ID	Objekt-ID • Bit 0 - 21 : Objektinstanz • Bit 22-31 : Objekttyp	Schreibzugriff mit Priorität 1.
...		
0xC0000000 + Property-ID	Objekt-ID • Bit 0 - 21 : Objektinstanz • Bit 22-31 : Objekttyp	Schreibzugriff mit Priorität 16.

IndexGroup bei Adressierung einer Notification Sink (Port 1000-8191)

Über die ADS-Schnittstelle der Notification Sink können u.a. empfangene Event- und COV-Notifications (Meldungen) ausgelesen werden. Bei der Verarbeitung der einzelnen Meldungen werden zwei verschiedene Adressierungsverfahren verwendet. Zum einen kann die Position der Meldung im Puffer verwendet werden (per Nummer). Bei rotierendem Puffer kann es sinnvoll sein die Adressierung der Sequenz-ID zu verwenden. Hierbei wird jeder Meldung beim Empfang eine eindeutige 32-Bit-Nummer zugeordnet. Gehen sehr viele Nachrichten ein, kann so erkannt werden, ob Meldungen verloren gegangen sind. Beim Löschen kann über die Sequenz-ID sichergestellt werden, dass keine durch Rotation verschobene Meldung "aus Versehen" gelöscht wird. Die Sequenz-ID ist beim Auslesen der EventNotifications an Byte-Offset 312 zu finden. Bei COV-Notifications an Byte-Offset 24.

ADS-Read

IndexGroup	IndexOffset	Beschreibung
0x00000001	immer 0	Auslesen der aktuellen Anzahl von Event-Nofitications im Event-Notification-Puffer
0x00000002	immer 0	Auslesen der aktuellen Anzahl von COV-Nofitications im COV-Notification-Puffer
0x00000003	Nummer der Event-Notification (0 .. Anzahl-1)	Auslesen einer Event-Notification. Die Struktur von Event-Notifications kann u.a. im Online-Dialog der Notification Sink ermittelt werden.

IndexGroup	IndexOffset	Beschreibung
0x00000004	Nummer der COV-Notification (0 .. Anzahl-1)	Auslesen einer COV-Notification. Die Struktur von COV-Notifications kann u.a. im Online-Dialog der Notification Sink ermittelt werden.
0x00000005	Nummer der Event-Notification (0 .. Anzahl-1)	Löschen einer Event-Notification
0x00000006	Nummer der COV-Notification (0 .. Anzahl-1)	Löschen einer COV-Notification
0x00000007	Nummer der Event-Notification (0 .. Anzahl-1)	Auslesen des Empfangszeitpunkts einer Event-Notification
0x00000008	Nummer der COV-Notification (0 .. Anzahl-1)	Auslesen des Empfangszeitpunkts einer COV-Notification
0x0000000C	Sequenz-Nr. der Event-Notification	Auslesen einer Event-Notification über eine Sequenz-Nummer
0x0000000D	Sequenz-Nr. der COV-Notification	Auslesen einer COV-Notification über eine Sequenz-Nummer
0x0000000E	Sequenz-Nr. der Event-Notification	Löschen einer Event-Notification über eine Sequenz-Nummer
0x0000000F	Sequenz-Nr. der COV-Notification	Löschen einer COV-Notification über eine Sequenz-Nummer

AdsWrite

IndexGroup	IndexOffset	Beschreibung
0x0000000A	Sequenz-Nr. der zu bestätigenden Event-Notification	Bestätigen einer EventNotification (AcknowledgeAlarm Service). Es wird eine AcknowledgeAlarm-Meldung für die mit der übergebenen Sequenz-Nr. übergegebenen EventNotification gesendet.
0x0000000B	EventNotification-Daten	Bestätigen einer EventNotification (AcknowledgeAlarm Service). Als Daten wird die empfangene EventNotification übergeben, die zuvor ausgelesen wurde.

ADS-Fehlercodes

Der 32-Bit-ADS-Fehlercode besteht immer aus einem allgemeinen ADS-Fehlercode (Lo-Word, siehe ADS-Dokumentation). Die entsprechende Meldung wird auch im Logger des TwinCAT System Managers textuell angezeigt.

6.2 Datentypen

Im Automation Interface und via ADS werden komplexe Properties als Byte-Array kodiert. In diesem Abschnitt soll erläutert werden, wie die BACnet-Datenkonzepte innerhalb von TwinCAT BACnet/IP auf C-Datenstrukturen abgebildet und bei der Konfiguration via "Automation Interface" bzw. ADS verwendet werden können. Im BACnet-Standard werden in Kapitel 21 ("FORMAL DESCRIPTION OF APPLICATION PROTOCOL DATA UNITS") die BACnet-Datenstrukturen spezifiziert.

Bis auf wenige Ausnahmen erfolgt die Abbildung von BACnet nach C der BACnet-Spezifikation. Ausnahmen bilden dynamische Daten, die entsprechend auf Listen bzw. AnyTypes abgebildet werden. Generell kann der Typ einer Property mit Hilfe des TwinCAT System-Managers ermittelt werden. In der Spalte "Type" im Reiter "Settings" und Reiter "Online" ist der Datentyp jeder Property inkl. der Struktur (durch Aufklappen der Property-Daten) zu erkennen. Datentypnamen die in der Spalte "Type" auf "[]" enden sind Arrays,

Datentypnamen die auf "List" enden sind Listen. Choice-basierte Datentypen sind an der Auswahl-Combobox in der Spalte "Type" zu erkennen, Datentypen mit optionalen Felder an einer CheckBox im ID-Feld.

Primitive Datentypen

0 = Null	Wird abgebildet durch eine Datenlänge von 0.
1 = Boolean	Wird abgebildet auf ULONG (4 Byte) mit den Werten 0x00000001 für TRUE und 0x00000000 für FALSE
2 = Unsigned Integer	Wird abgebildet auf ULONG (4 Byte). Eine optimierte Variante kann verwendet werden, wobei Ganzzahlwerte bis 255 als 1 Byte und Ganzzahlwerte bis 65535 als 2 Byte (USHORT) dargestellt werden.
3 = Signed Integer	Wird abgebildet auf LONG (4 Byte)
4 = Real	Wird abgebildet auf FLOAT (4 Byte)
5 = Double	Wird momentan nicht verwendet
6 = Octet String	Wird abgebildet auf BYTE[]
7 = Character String	Wird abgebildet auf CHAR[]. Also 0 terminierte ISO 8859-1 Zeichenketten. Momentan werden in TwinCAT BACnet/IP Zeichenkette der Länge 256 unterstützt.
8 = Bit String	Wird abgebildet auf BYTE[]. Im höherwertigsten (most significant) Byte steht die Anzahl der im nachfolgenden Byte nicht verwendeten Bits (0..7). Der erste boole'sche Wert steht im höherwertigsten Bit des nachfolgenden Byte. Bis auf wenige Ausnahmen haben Bit Strings eine Länge von 2 Bytes.
9 = Enumerated	Wird abgebildet auf C-Enumerationen (4 Byte) - von 0 beginnend.

10 = Date	Wird abgebildet auf:
	`struct BACnetDate`
	`{`
	`  BYTE year; // year minus 1900 X'FF' = unspecified`
	`  BYTE month; // (1.. 12) 1 = January X'FF' = unspecified`
	`  BYTE day; // day of month (1..31), X'FF' = unspecified`
	`  BYTE dayOfWeek; // day of week (1..7) 1 = Monday -- 7 = Sunday -- X'FF' = unspecified`
	`}`

11 = Time	Wird abgebildet auf:
	`struct BACnetTime`
	`{`
	`  BYTE hour; // hour (0..23), (X'FF') = unspecified`
	`  BYTE minute; // minute (0..59), (X'FF') = unspecified`
	`  BYTE second; // (0..59), (X'FF') = unspecified`
	`  BYTE hundredths; // hundredths (0..99), (X'FF') = unspecified`
	`}`

12 = BACnetObjectIdentifier	Wird abgebildet auf einen 4-Byte ObjektIdentifier mit 10Bit für den Objekttyp und einer 22 Bit Objektinstanz
	`struct BACnetObjectIdentifier`
	`{`
	`  DWORD objInst : 22;`
	`  DWORD objType : 10;`
	`}`

Strukturen

BACnet-Sequenzen ohne optionale Felder werden auf Strukturen abgebildet. Im Folgenden ist ein Beispiel dargestellt.

```
BACnetPrescale ::=
SEQUENCE
{
  multiplier [0] Unsigned,
  moduloDivide [1] Unsigned
}
```

```
struct BACnetPrescale
{
  ULONG multiplier;
  ULONG moduloDivide;
}
```

Optionale Felder

BACnet-Sequenzen mit optionalen Feldern werden auf Strukturen mit einem zusätzlichen Feld *validFields* abgebildet. Das zusätzliche Feld gibt an welche Felder in der Struktur aktiv sind. Bit 0 von *validFields* gibt an ob das nächstfolgende Feld (im Beispiel *objectIdentifier*) aktiv ist. Bei nicht optionalen Feldern ist/muss das entsprechende Bit immer gesetzt/sein.

```
BACnetObjectPropertyReference ::= SEQUENCE
{
  objectIdentifier [0] BACnetObjectIdentifier,
  propertyIdentifier [1]
BACnetPropertyIdentifier,
  propertyArrayIndex [2] Unsigned OPTIONAL
}
```

```
struct BACnetObjectPropertyReference
{
  ULONG validFields;
  BACnetObjectIdentifier objectIdentifier; // [0]
  BACnetPropertyIdentifier propertyIdentifier; //
[1]
  ULONG propertyArrayIndex; // [2] Unsigned
OPTIONAL
}
```

Choice

Eine BACnet-Choice wird auf eine Struktur mit dem Feld *choiceField* und einer Union abgebildet. Das Feld *choiceField* selektiert dabei welches Feld der union aktiv ist. Die Länge der Daten wird dabei durch das größte Feld der union bestimmt. Das Feld *choiceField* ist eine Enumeration (4 Byte).

```
BACnetRecipient ::=
CHOICE
{
  device [0] BACnetObjectIdentifier,
  address [1] BACnetAddress
}
```

```
enum BACnetRecipient_Choice
{
  device,
  address };struct _BACnetRecipient
{
  BACnetRecipient_Choice choiceField; union
  {
    BACnetObjectIdentifier device;
    BACnetAddress address;
  };
}
```

Arrays

BACnet-Arrays mit fixen Datengrößen werden auf C-Arrays abgebildet. Einige BACnet-Arrays können Daten variabler Länge enthalten (Listen oder AnyTypes). In diesem Fall werden BACnet-Arrays auf Listen abgebildet, die im folgenden Abschnitt beschrieben werden.

Listen

BACnet-Listen und einige BACnet-Arrays werden auf eine spezielle Listenstruktur abgebildet. Jedes Listenelement besitzt einen Header (ListEntryHdr) dem direkt die Daten des Elements folgen. In obersten Bit von *nEntrySize* (*nEntrySize* & 0x80000000) ist kodiert, ob ein weiteres Element folgt (Bit 31 = 1), oder ob es sich um das letzte Listenelement handelt (Bit 31 = 0). In den unteren 31 Bit von *nEntrySize* ist die Größe der nachfolgenden Daten als Anzahl von Bytes kodiert. Der Header enthält ein weiteres intern verwendetes *reserved* Feld dem die Daten folgen. Das nachfolgende Listenelement beginnt an einer 4-Byte-alignten den Daten folgenden Grenze.

```
struct
_ListEntryHdr
{
    DWORD nEntrySize;
    DWORD reserved;
}
```

Enthält z.B. eine Liste ein erstes Element der Länge 1 mit dem Wert 1, ist die Liste kodiert: 08 00 00 01 00 00 00 00 01 dann folgen 3 Alignment-Bytes 00 00 00. Ist das nächste Listenelement das letzte Element der Liste mit 1 Byte Daten mit dem Wert 3: 00 00 00 01 00 00 00 00 03. Die Datenlänge der Beispielliste wäre 24 Byte, da dem letzten Element von 3 Alignment-Bytes folgen.

Eine leere Liste ist kodiert als (4 Byte): 00 00 00 00.

AnyTypes

BACnet Daten vom Typ *ABSTRACT-SYNTAX.&Type* werden durch die Struktur AnyType abgebildet. Das Feld *nSize* enthält die Größe der nachfolgenden Daten inkl. dem Feld dataType (4 Byte). Die eigentlichen Daten haben also die Größe nSize-4. Das Feld dataType gibt den Datentyp des Datenelements an. Eine Liste der Datentypen ist im Folgenden dargestellt.

```
struct
AnyType
{
    ULONG nSize;
    BACnetDataTypes dataType;    //... Data
}
```

Im Folgenden sind die verwendeten Datentyp-Kodierungen für AnyTypes aufgelistet:

```
#define ARRAYTYPE 0x100
#define LISTTYPE 0x200
enum BACnetDataTypes
{
// Primitive Data
DataTypeNull = 0,
DataTypeBool = 1,
DataTypeCharacterStringExt = 2,
DataTypeUnsignedInteger = 3,
DataTypeSignedInteger = 4,
DataTypeReal = 5,
DataTypeDouble = 6,
DataTypeOctetString = 7,
DataTypeEnumerated = 8,
DataTypeBACnetObjectIdentifier = 9,
DataTypeAddressBinding = 10,
DataTypeBitString = 11,
DataTypeDate = 12,
DataTypeTime = 13,
// Complex Data
DataTypeDateTime = 14,
DataTypeArrayIndex = 15,
DataTypeBACnetCalendarEntry = 16,
DataTypeBACnetObjectType = 17,
DataTypeBACnetEventTransitionBits = 18,
DataTypeBACnetStatusFlags = 20,
DataTypeBACnetEventState = 21,
DataTypeBACnetReliability = 22,
DataTypeBACnetPolarity = 23,
DataTypeBACnetDateTime = 24,
DataTypeBACnetEngineeringUnits = 25,
DataTypeBACnetPriorityValue = 26,
DataTypeBACnetLimitEnable = 27,
DataTypeBACnetNotifyType = 29,
DataTypeBACnetTimeStamp = 30,
DataTypeBACnetDeviceObjectPropertyReference = 31,
DataTypeBACnetDeviceStatus = 32,
DataTypeBACnetServicesSupported = 33,
DataTypeBACnetObjectTypesSupported = 34,
DataTypeBACnetSegmentation = 35,
DataTypeBACnetVTClass = 36,
DataTypeBACnetVTSession = 37,
DataTypeBACnetSessionKey = 38,
DataTypeBACnetRecipient = 39,
DataTypeBACnetAddressBinding = 40,
DataTypeBACnetCOVSubscription = 41,
DataTypeBACnetEventType = 42,
DataTypeBACnetEventParameter = 43,
DataTypeBACnetFileAccessMethod = 44,
DataTypeReadAccessSpecification = 45,
DataTypeReadAccessResult = 46,
DataTypeBACnetObjectPropertyReference = 47,
DataTypeBACnetSetpointReference = 48,
DataTypeBACnetDestination = 49,
DataTypeBACnetProgramState = 50,
DataTypeBACnetProgramRequest = 51,
DataTypeBACnetProgramError = 52,
DataTypeBACnetDateRange = 53,
DataTypeBACnetDailySchedule = 54,
DataTypeBACnetSpecialEvent = 55,
DataTypeBACnetClientCOV = 56,
DataTypeBACnetLogRecord = 57,
DataTypeBACnetLifeSafetyState = 58,
```

```
DataTypeBACnetLifeSafetyMode = 59,
DataTypeBACnetSilencedState = 60,
DataTypeBACnetLifeSafetyOperation = 61,
DataTypeBACnetBinaryPV = 62,
DataTypeBACnetDaysOfWeek = 63,
DataTypeBACnetWeekNDay = 64,
DataTypeBACnetTimeValue = 65,
DataTypeBACnetValue = 66,
DataTypeBACnetAddress = 67,
DataTypeBACnetPropertyIdentifier = 68,
DataTypeEnumerationValueType = 69,
DataTypeBitFieldBit = 70,
DataTypeBACnetLogDatum = 71,
DataTypeAnyType = 72,
DataTypePresentValue = 73,
DataTypeContextTag = 74,
DataTypeValueChoice = 75,
DataTypeBACnetLogStatus = 76,
DataTypeBACnetRecipientProcess = 77,
DataTypeSubscribeCOVPropertyRequest = 78,
DataTypeBACnetPropertyReference = 79,
DataTypeBACnetPropertyResult = 80,
DataTypeBACnetDeviceObjectPropertyValue = 81,
DataTypeCOVNotificationRequest = 82,
DataTypeEventNotificationRequest = 83,
DataTypeBACnetNotificationParameters = 84,
DataTypeChange_of_Bitstring = 85,
DataTypeChange_of_State=86,
DataTypeChange_of_Value=87,
DataTypeCommand_failure=88,
DataTypeFloating_limit=89,
DataTypeOut_of_Range =90,
DataTypeBACnetPropertyValue=91,
DataTypeComplex_Tag=92,
DataTypeChange_of_life_safety = 93,
DataTypeExtended=94,
DataTypeBuffer_ready =95,
DataTypeUnsigned_range = 96,
DataTypeBACnetPropertyResultValue = 97,
DataTypeServiceCOVPropSubscription = 98,
DataTypeServiceCOVSubscription = 99,
DataTypeCOVNotification = 100,
DataTypeAcknowledgeAlarmRequest = 101,
DataTypeBACnetVTOpenRequest = 102,
DataTypeBACnetPrescale = 103,
DataTypeBACnetScale = 104,
DataTypeBACnetAction = 105,
DataTypeBACnetError = 106,
DataTypeBacnetErrorType = 107,
DataTypeErrorClass = 108,
DataTypeErrorCode = 109,
DataTypeBACnetRejectReason = 110,
DataTypeBACnetAbortReason = 111,
DataTypeBDTEntry = 112,
DataTypeIpEntry = 113,
DataTypeMaskEntry = 114,
DataTypeBACnetEthernetAddress = 115,
DataTypeBACnetLonTalkAddress = 116,
DataTypeBACnetLonTalkNeuronId = 117,
DataTypeBACnetNodeType = 118,
DataTypeBACnetDeviceObjectReference = 119,
DataTypeBACnetActionCommand = 120,
DataTypeGetEventInformation = 121,
DataTypeGetEnrollmentSummaryCriteria = 122,
DataTypeGetEnrollmentSummaryCriteriaResponse = 123,
DataTypeDeviceCommunicationControl = 124,
DataTypeInt16 = 125,
DataTypeUInt16 = 126,
DataTypeUInt8 = 127,
DataTypeInt8 = 128,
DataTypeByte = 129,
DataTypeFDTEntry = 130,
DataTypeBACnetAddress_3 = 131,
DataTypeBACnetAddress_4 = 132,
DataTypeBACnetAddress_5 = 133,
DataTypePersistentDataState = 134,
DataTypeFallbackRealValue = 135,
DataTypeFallbackBinaryValue = 136,
DataTypeRealNull = 137, // PresentValue in AnalogObj
DataTypeEnumeratedNull = 138, // PresentValue in BinaryObj
```

```
DataTypeUnsignedIntegerNull = 139, // PresentValue in Multistate
DataTypeUnknown = 140,
DataTypeBACnetDiagnosisPerformance = 200,
DataTypeBACnetDiagnosisEthStatistics = 201,
DataTypeBACnetFrameStatistics = 202,
DataTypeBACnetInfo = 203,
DataTypeBACnetDiagnosis = 204,
DataTypeTcIoEthStatistic = 205,
DataTypeTcIoEthTxRxErrorCount = 206,
DataTypeTcIoEthPortStatistic = 207,
DataTypeBACnetServerStatistics = 208,
DataTypeBACnetFrameServicesDiag = 209,
// Arraytypen
DataTypeBACnetCalendarEntryArr = ARRAYTYPE + DataTypeBACnetCalendarEntry,
DataTypeBACnetObjectIdentifierArr = ARRAYTYPE + DataTypeBACnetObjectIdentifier,
DataTypeCharacterStringExtArr = ARRAYTYPE + DataTypeCharacterStringExt,
DataTypeBACnetPriorityValueArr = ARRAYTYPE + DataTypeBACnetPriorityValue,
DataTypeBACnetTimeStampArr = ARRAYTYPE + DataTypeBACnetTimeStamp,
DataTypeBACnetVTClassArr = ARRAYTYPE + DataTypeBACnetVTClass,
DataTypeBACnetVTSessionArr = ARRAYTYPE + DataTypeBACnetVTSession,
DataTypeBACnetSessionKeyArr = ARRAYTYPE + DataTypeBACnetSessionKey,
DataTypeBACnetRecipientArr = ARRAYTYPE + DataTypeBACnetRecipient,
DataTypeBACnetAddressBindingArr = ARRAYTYPE + DataTypeBACnetAddressBinding,
DataTypeBACnetCOVSubscriptionArr = ARRAYTYPE + DataTypeBACnetCOVSubscription,
DataTypeUnsignedIntegerArr = ARRAYTYPE + DataTypeUnsignedInteger,
DataTypeBACnetDestinationArr = ARRAYTYPE + DataTypeBACnetDestination,
DataTypeBACnetDeviceObjectPropertyReferenceArr = ARRAYTYPE + DataTypeBACnetDeviceObjectPropertyRefer
ence,
DataTypeBACnetLifeSafetyStateArr = ARRAYTYPE + DataTypeBACnetLifeSafetyState,
DataTypeEnumeratedArr = ARRAYTYPE + DataTypeEnumerated,
DataTypeReadAccessSpecificationArr = ARRAYTYPE + DataTypeReadAccessSpecification,
DataTypeReadAccessResultArr = ARRAYTYPE + DataTypeReadAccessResult,
DataTypeBACnetSpecialEventArr = ARRAYTYPE + DataTypeBACnetSpecialEvent,
DataTypeBACnetLogRecordArr = ARRAYTYPE + DataTypeBACnetLogRecord,
DataTypeBACnetTimeValueArr = ARRAYTYPE + DataTypeBACnetTimeValue,
DataTypeBACnetDeviceObjectReferenceArr = ARRAYTYPE + DataTypeBACnetDeviceObjectReference,
// Listen
DataTypeBACnetTimeValueList = LISTTYPE + DataTypeBACnetTimeValue,
DataTypeBACnetSpecialEventList = LISTTYPE + DataTypeBACnetSpecialEvent,
DataTypeBACnetDailyScheduleList = LISTTYPE + DataTypeBACnetTimeValueList,
DataTypeCharacterStringExtList = LISTTYPE + DataTypeCharacterStringExt,
DataTypeBACnetLogRecordList = LISTTYPE + DataTypeBACnetLogRecord,
DataTypeUnsignedIntegerList = LISTTYPE + DataTypeUnsignedInteger,
DataTypeReadAccessResultList = LISTTYPE + DataTypeReadAccessResult,
DataTypeReadAccessSpecificationList = LISTTYPE + DataTypeReadAccessSpecification,
DataTypeBACnetValueList = LISTTYPE + DataTypeBACnetValue,
DataTypeBACnetLogDatumList = LISTTYPE + DataTypeBACnetLogDatum,
DataTypeBACnetPropertyResultList = LISTTYPE + DataTypeBACnetPropertyResult,
DataTypeBACnetPropertyReferenceList = LISTTYPE + DataTypeBACnetPropertyReference,
DataTypeCOVNotificationRequestList = LISTTYPE + DataTypeCOVNotificationRequest,
DataTypeEventNotificationRequestList = LISTTYPE + DataTypeEventNotificationRequest,
DataTypeBACnetPropertyValueList = LISTTYPE + DataTypeBACnetPropertyValue,
DataTypeBACnetActionList = LISTTYPE + DataTypeBACnetActionCommand,
DataTypeBACnetActionListList = LISTTYPE + DataTypeBACnetActionList,
};
```

6.3 Automation Interface

Der TwinCAT System Manager bietet eine XML-basierte Schnittstelle zur programmatischen Konfiguration u.a. von TwinCAT BACnet/IP. Zusätzlich können auch programmatisch alle BACnet-Elemente in einer TwinCAT-Konfiguration erzeugt werden. Generell wird hier auf die allgemeine Dokumentation des TwinCAT Automation Interface verwiesen. In diesem Abschnitt wird erläutert, welche Parameter für BACnet-spezifische Komponenten verwendet werden können.

Erzeugen von BACnet Client-, Server, Objekten

Für das Erzeugen von TwinCAT-Konfigurationselementen bietet der System Manager die Methode CreateChild. Als Parameter kann ein TreeItem-Name, ein Sub-Typ (bei BACnet immer 48), ein Einfügepositions-String und eine Sub-Typ-spezifische Information.

```
serverItem.CreateChild("Bi_Demo", 48, "", 0x03060203);
```

Der letzte Parameter wird auch als Class-ID bezeichnet und identifiziert das Element das angelegt werden soll. Es können ein BACnet-Device (CID_TcIoBACnetBACnetDevice), ein BACnet-Server (CID_TcIoBACnetServer) und ein BACnet-Client (CID_TcIoBACnetClient) angelegt werden. BACnet-Server-Objekte können mit der Class-ID CID_TcIoBACnetObjBase addiert mit dem Objekttyp angelegt werden. Ein BinaryInput-Objekt hat damit die Class-ID 0x03060203. BACnet-Client-Objekte mit der Basis Class-ID CID_TcIoBACnetObjRemoteBase.

```
ULONG CID_TcIoBACnetBACnetDevice   = 0x03060001;
ULONG CID_TcIoBACnetServer     = 0x03060110;
ULONG CID_TcIoBACnetClient     = 0x03060120;
ULONG CID_TcIoBACnetObjBase    = 0x03060200;
ULONG CID_TcIoBACnetObjRemoteBase  = 0x03060300;
ULONG CID_TcIoBACnetDynamicObject  = 0x030602FE;
```

Property-Konfiguration

Mit Hilfe der Methoden ProduceXml und ConsumeXml erfolgt die Konfiguration der einzelnen TwinCAT-Komponenten. Über das System Manager-Menü "Action" -> "Export XML Description ..." kann die XML-Beschreibung für Testzwecke exportiert und betrachtet werden.

```
<Parameter>
  <Name>TcIoBACnetPropCfg_PropTimeDelay</Name>
  <BaseType GUID="{18071995-0000-0000-0000-000000000007}">DWORD</BaseType>
  <PTCID>#x03060171</PTCID>
</Parameter>
...
<Value>
  <Name>TcIoBACnetPropCfg_PropTimeDelay</Name>
  <Value>0</Value>
</Value>
```

Die Abbildung zeigt einen Ausschnitt der mit ProduceXml erzeugten XML-Beschreibung eines BACnet-Objekts. In der Parameter-Sektion ist der Aufbau und Typ eines Parameters beschrieben. In der Value-Sektion wird der eigentliche Wert festgelegt. Die Parameter-ID setzt sich bei BACnet-Properties aus 0x03060100 addiert mit der jeweiligen Property-ID zusammen. Der folgende C#-Quellcode-Ausschnitt zeigt wie der Wert der Property TimeDelay wie im "Settings"-Reiter angezeigt, ausgelesen werden kann.

```
XmlDocument doc = new System.Xml.XmlDocument();
s = objItem.ProduceXml(true);
doc.LoadXml(s);
XmlNodeList list = doc.SelectNodes("//Value[./Name='" + "TcIoBACnetPropCfg_PropTimeDelay"+ "']/
Value");
UInt32 timeDelay = Convert.ToUInt32(list[0].InnerText);
```

Beim Schreiben der Property-Daten empfiehlt es sich zunächst via ProcuceXml die Xml-Beschreibung auszulesen, dann den Parameterwert zu editieren und abschließend via ConsumeXml die Parameterdaten zu schreiben.

```
XmlDocument doc = new System.Xml.XmlDocument();
s = objItem.ProduceXml(true);
doc.LoadXml(s);
XmlNodeList list = doc.SelectNodes("//Value[./Name='" + "TcIoBACnetPropCfg_PropTimeDelay"+ "']/
Value");
list[0].InnerText = 10;
objItem.ConsumeXml(doc.OuterXml);
```

Je nach Datentyp der Property kann es vorkommen, dass neben dem Parameterwert auch die Parameter-Sektion angepasst werden muss. Bei STRING-Datentypen muss beim Datentyp STRING(XX) die Länge der Zeichenkette eingetragen werden.

```
<Parameter>
  <Name>TcIoBACnetPropCfg_PropObjectName</Name>
  <BaseType GUID="{18071995-0000-0000-0000-000000010006}">STRING(5)</BaseType>
  <PTCID>#x0306014d</PTCID>
</Parameter>
...
<Value>
  <Name>TcIoBACnetPropCfg_PropObjectName</Name>
  <String>CAL_0</String>
</Value>
```

Bei Properties mit komplexen Datentypen werden die Konfigurationsdaten als Byte-Array gespeichert. Die genaue Kodierung der Daten wird im Abschnitt "Datentypen" beschrieben. Um komplexe Parameterdaten zu verändern muss entsprechend in der Parametersektion die *BitSize* und die *Elements* der *ArrayInfo* angepasst werden, welche die Anzahl der Bytes angibt. In der Value-Sektion sind die eigentlich Daten im HEXBIN-Format kodiert (pro Byte 2 Hexadezimalwerte).

```
<Parameter>
  <Name HideSubItems="1">TcIoBACnetPropCfg_PropDatelist</Name>
  <BitSize>96</BitSize>
  <BaseType GUID="{18071995-0000-0000-0000-000000000001}">BYTE</BaseType>
  <ArrayInfo>
    <LBound>0</LBound>
    <Elements>12</Elements>
  </ArrayInfo>
  <PTCID>#x03060117</PTCID>
</Parameter>
...
<Value>
  <Name>TcIoBACnetPropCfg_PropDatelist</Name>
  <Data>00000000ffffffff00000000</Data>
</Value>
```

Prozessdatenkonfiguration

Auch die Konfiguration der Prozessdaten kann über das Automation Interface innerhalb der XML-Beschreibung angepasst werden. Jede Property die als Prozessdatenvariable aktiviert wurde ist mit einem Eintrag in der Sektion DataArea vermerkt. In TwinCAT BACnet/IP existieren 2 DataAreas je eine für Eingangs- und Ausgangsprozessdaten. Im Beispiel ist die Konfiguration der DataArea mit 2 Eingangsprozessdaten gezeigt. Pro aktivierter Property existiert ein Symbol. Im Symbol ist jeweils der BitOffset vermerkt, der die Position der Property-Daten im Prozessdaten-Image bestimmt. Wichtig ist, dass die Properties jeweils an einer alignten Adresse beginnen (WORD-Symbol an durch 2 teilbaren Adressen, DWORD-Symbole an durch 4 teilbaren Adressen). Die ByteSize der DataArea gibt jeweils die Größe der Symbole an.

```
<DataAreas>
  <DataArea>
    <AreaNo AreaType="InputSrc" ChildArea="1">0</AreaNo>
    <Name>Inputs</Name>
    <ContextId>1</ContextId>
    <ByteSize>4</ByteSize>
    <Symbol>
      <Name>PresentValue</Name>
      <BaseType GUID="{18071995-0000-0000-0000-000000000004}">WORD</BaseType>
      <BitSize>16</BitSize>
      <BitOffs>0</BitOffs>
    </Symbol>
    <Symbol>
      <Name>StatusFlags</Name>
      <BaseType GUID="{18071995-0000-0000-0000-000000000004}">WORD</BaseType>
      <BitSize>16</BitSize>
      <BitOffs>16</BitOffs>
    </Symbol>
  </DataArea>
  <DataArea>
    <AreaNo AreaType="OutputDst" ChildArea="1">1</AreaNo>
    <Name>Outputs</Name>
    <ContextId>1</ContextId>
    <ByteSize>0</ByteSize>
  </DataArea>
</DataAreas>
```

Zusätzlich zur Konfiguration der DataArea-Symbole muss in den Parametern TcIoBACnetPdInCfg und TcIoBACnetPdOutCfg dem BACnet-Stack mitgeteilt werden, welche Property-Daten als Prozessdaten behandelt werden sollen. Beim Verändern der Parameter TcIoBACnetPdInCfg und TcIoBACnetPdOutCfg muss wieder jeweils die Datenlänge in der Parameter-Sektion angepasst werden. Die Struktur der beiden Parameter wird im Folgenden erläutert.

```
<Value>
  <Name>TcIoBACnetPdInCfg</Name>
  <Data>0000000055000001000000006f000000</Data>
</Value>
```

BACnet Server

Pro aktivierter Prozessdaten-Property müssen die Parameter TcIoBACnetPdInCfg und TcIoBACnetPdOutCfg jeweils ein Element der Struktur ProcessDataServerCfg enthalten. In der Struktur muss festgelegt werden an welches Bit-Offset die Daten kopiert werden sollen, um welche Property es sich handelt (*propId*) und bei Out-Prozessdaten mit welcher Priorität die Property-Daten ins PriorityArray kopiert werden sollen.

```
struct ProcessDataServerCfg
{
    ULONG bitOffs;
    USHORT propId;
    USHORT priority;
}
```

BACnet Client

Auch bei Client-Objekten existieren neben der Symbolkonfiguration zusätzliche Parameter, welche die Konfiguration der Property-Prozessdaten festlegen. Es wird bei Client-Objekten zwischen Ein- und Ausgangsprozessdaten unterschieden. Die Struktur-Felder entsprechen den Konfigurationsdialogen im System Manager.

```
struct ProcessDataClientInCfg
{
    USHORT propId;
    USHORT useCov      : 1;
    USHORT usePolling  : 1;
    USHORT reserved    : 6;
    USHORT tickModulo  : 8;
    ULONG timeDataCOV;
    ULONG timeDataPoll;
    ULONG bitOffs;
    ULONG bitLen;
    floatcovIncrement;
}
```

```
struct ProcessDataClientOutCfg
{
    USHORT propId;
    USHORT useWriteOnChange : 1;
    USHORT usePeriodicWrite : 1;
    USHORT usePriority      : 1;
    USHORT reserved     : 5;
    USHORT tickModulo       : 8;
    ULONG priority;
    ULONG cycleTime;
    ULONG bitOffs;
    ULONG bitLen;
}
```

6.4 Automation Interface und das Microsoft .NET Framework

Die in diesem Kapitel vorgestellte Funktionalität unterliegt nicht dem allgemeinen Support durch Beckhoff. Die Benutzung erfolgt auf eigene Gefahr. Für die Fehlerfreiheit der vorgestellten Funktionen wird nicht garantiert. Rückmeldungen auf eventuelle Fehler werden gerne entgegengenommen.

Für die Nutzung des Automation Interface aus dem Microsoft .NET Framework steht die Bibliothek BACnetExtensionCtrl.dll zur Verfügung, die mit TwinCAT ausgeliefert wird. In der Bibliothek sind Wrapper-Funktionen implementiert, die das programmatische Erstellen einer TwinCAT BACnet/IP Konfiguration u.a. mit C# erleichtern. In diesem Kapitel soll anhand ausgewählter Beispiele, die Verwendung der Bibliothek erläutert werden. Dabei werden nicht alle Varianten der implementierten Funktionen eingegangen. Ein vollständige Übersicht aller Funktionen erschließt sich durch die Verwendung von IntelliSense des Visual Studio. Alle BACnet-spezifischen Funktionen sind im Namensraum TwinCAT.BACnet angesiedelt.

Um die Bibliothek verwenden zu können, müssen einem .NET-Projekt Referenzen auf folgende Dateien hinzugefügt werden:

- C:\TwinCAT\Io\AddIns\BACnetExtensionCtrl.dll
- C:\TwinCAT\Io\AddIns\TCatSysManagerLib.dll

Für die folgenden Beispiele wird von dieser Namespace-Deklaration ausgegangen:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using TCatSysManagerLib;
using TwinCAT.BACnet.Extension;
using TwinCAT.BACnet.ExtensionHelper;
using TwinCAT.BACnet.BACnetDefinitions;
using System.Xml;
```

Zunächst muss eine Referenz auf den TwinCAT System Manager akquiriert werden.

```
ITcSysManager ipSysMan = (ITcSysManager)Activator.CreateInstance(Type.GetTypeFromProgID("TCatSysMana
ger.TcSysManager"));
```

BACnet-Device Konfiguration

Im Namensraum TwinCAT.BACnet.Extension kann mit Hilfe der Klasse BACnetExtension ein BACnetDevice erstellt werden (CreateBACnetDevice). Die Konfiguration von Elementen erfolgt im Automation Interface durch die Funktionen ProduceXml und ConsumeXml. Mit Hilfe von ProduceXml können Parameter gelesen und mit Hilfe von ConsumtXml geschrieben werden. Um Default-Parameter zu erhalten, sollte generell eine Konfiguration ausgelesen werden bevor Parameter verändert werden. Dies kann mit der Wrapper-Funktion BACnetExtension.GetXml erfolgen. Nachfolgend können mit Hilfe weiterer Wrapper-Funktionen Daten im XML-Dokument manipuliert werden. Abschließend können veränderte Parameter mit der Funktion XmlHelper.ConsumeAll gespeichert werden. Die Funktion XmlHelper.ConsumeParameters kann verwendet werden, wenn nur Parameter und keine Prozessdaten verändert wurden. Die Verarbeitung von ConsumeParameters ist dabei effizienter. Generell ist zu empfehlen die ConsumeXml-Funktionen der BACnet-Bibliothek zu verwenden, da hier Optimierungen implementiert sind, die bei großen Projekten eine schnellere Generierung ermöglichen.

Die meisten Funktionen von der BACnetExtensionCtrl.dll unterstützen 2 Varianten. Eine Variante verwendet ein ITcSmTreeItem als Parameter. Hier wird intern ProduceXml und ConsumeXml aufgerufen. In der zweiten Variante wird ein XmlDocument übergeben und erwartet, das Lesen und Schreiben der Daten extern realisiert werden. Bei komplexen Operationen auf einem TreeItem sollten die Funktionen mit XmlDocument verwendet werden, da ConsumeXml eine potenziell rechenaufwendige Operation ist und nicht mehrfach aufgerufen werden sollte.

Im folgenden Beispiel wird ein BACnet-Device erstellt. Der Modus für relative AmsNetIds wird aktiviert, der bei automatisch generierten Projekten empfohlen wird, da so automatisch eine gültige AmsNetId auf dem Zielsystem berechnet wird. Dabei werden die ersten vier Stellen der System-Net-Id verwendet. Mit Hilfe der Funktion SetNetworkAdapterVirtualName kann die Netzwerk-Adapter-Konfiguration über einen virtuellen Gerätenamen erstellt werden. Dies ist auf Windows CE-basierten Systemen sinnvoll, da so die generierte Konfiguration auf beliebigen CXxxx-Geräten eingesetzt werden kann, ohne manuell einen Netzwerkadapter bei der Inbetriebnahme auswählen zu müssen. Die IP-Adresse wird in der Standardeinstellung automatisch vom Betriebssystem bezogen: SetUseOsIpSettingsMode. Zusätzlich wird die automatische Sommer-/Winterzeitanpassung durch das Betriebssystem aktiviert (ActivateOsTimeMode).

```
ITcSmTreeItemipDevice = BACnetExtension.CreateBACnetDevice(ipSysMan);
XmlDocument doc = BACnetExtension.GetXml(ipDevice);
BACnetDeviceConfiguration.EnableRelativeAmsNetIdMode(doc);
BACnetDeviceConfiguration.SetNetworkAdapterVirtualName(doc, "TCIXPNPE1");
// use ip given by operating system
BACnetDeviceConfiguration.SetUseOsIpSettingsMode(doc, true);
// set daylightsavings status to automatic ( is also default )
BACnetDeviceConfiguration.ActivateOsTimeMode(doc);
// Commit all changes made to BACnet-Device configuration
XmlHelper.ConsumeAll(ipDevice, doc);
```

BACnet-Server Konfiguration

Mit Hilfe der Funktion BACnetDeviceConfiguration.CreateBACnetServer kann ein BACnet Server erstellt werden. Dabei muss ein ITcSmTreeItem auf ein BACnet Device sowie eine BACnet-ID als Parameter übergeben werden. Im Beispiel ist erläutert, wie ein Passwort konfiguriert, sowie persistente Daten aktiviert werden können.

```
ITcSmTreeItem ipServer = BACnetDeviceConfiguration.CreateBACnetServer(ipDevice, 123);
doc = BACnetExtension.GetXml(ipServer);
BACnetServerConfiguration.SetPassword(doc, newBACnetCharacterString("secure"));
// enable persistent data
BACnetServerConfiguration.SetPersistentDataMode(doc, true);
BACnetServerConfiguration.SetPersistentDataUsvMode(doc, false);
```

```
BACnetServerConfiguration.SetPersistentDataInterval(doc, 30 * 60); // set persistent data interval t
o 30 min.
// Commit all changes made to BACnet-Server configuration
XmlHelper.ConsumeAll(ipServer, doc);
```

BACnet-Objekt Konfiguration

Objekterzeugung und Property-Zugriff

Mit Hilfe der Funktion CreateObject können neue BACnet-Objekte erstellt werden. Es wird ein Modus mit automatischer ID-Generierung unterstützt sowie die explizite Konfiguration mit einer Objekt-ID. Mit Hilfe der Funktion WriteProperty können Property-Werte manipuliert werden. Dabei wird der Typ der Daten in <> hinter der Funktionen angeben. Die Ableitung der Property-Datentypen ist im allgemeinen Automation Interface Kapitel beschrieben. Um die Datentypen von Properties zu ermitteln kann auch die Funktion ReadProperty ohne <> verwendet werden. Parameter können so ausgelesen und der Typ des zurückgegebenen .NET Objekts ausgeben werden.

Im Beispiel werden einige Properties konfiguriert. Bei Properties vom Datentyp string wird automatisch eine entsprechende UTF8-Kodierung erstellt. Das heißt die in .NET verwendeten UCS2-Zeichenketten werden in das BACnet-spezifische UTF8-Format umgewandelt.

```
// automatic mode -> obj-
id and objname are assigned automaticallyITcSmTreeItem biObj = BACnetServerConfiguration.CreateObjec
t(ipServer, BACnetObjectType.BinaryInput);
// manual obj-id configuration
ITcSmTreeItem boObj = BACnetServerConfiguration.CreateObject(ipServer, BACnetObjectType.BinaryOutput
, 17);
doc = BACnetExtension.GetXml(boObj);
BACnetObjectConfiguration.WriteProperty<string>(doc, BACnetPropertyIdentifier.Description, "компании
 Beckhoff"); // UTF-8 Support
BACnetObjectConfiguration.WriteProperty<string>(doc, BACnetPropertyIdentifier.ObjectName, "bo_12_x")
;
BACnetObjectConfiguration.WriteProperty<bool>(doc, BACnetPropertyIdentifier.OutOfService, true);
BACnetObjectConfiguration.WriteProperty<Byte[]>(doc, BACnetPropertyIdentifier.EventEnable, newByte[]
 { 0x05, 0xE0 });
XmlHelper.ConsumeAll(boObj, doc);
```

Im folgenden Beispiel werden 10 MultistateValue-Objekte erzeugt und jeweils mit einer wachsenden Anzahl von Zuständen (NumberOfStates) konfiguriert.

```
// create 10 multistate value objects with increasing number of states
for (uint i = 0; i < 10; i++)
{
    ITcSmTreeItem mvObj = BACnetServerConfiguration.CreateObject(ipServer, BACnetObjectType.MultiSta
teValue);
    doc = BACnetExtension.GetXml(mvObj);
    CharacterStringExtList stateText = newCharacterStringExtList();
    for( int j=0; j<i+2; j++ )
    stateText.AddCharacterString("MV_State_" + j.ToString());
    BACnetObjectConfiguration.WriteProperty<CharacterStringExtList>(doc, BACnetPropertyIdentifier.St
ateText, stateText);
    BACnetObjectConfiguration.WriteProperty<string>(doc, BACnetPropertyIdentifier.Description, "Objec
ct with " + i.ToString() +" states");
    BACnetObjectConfiguration.WriteProperty<UInt32>(doc, BACnetPropertyIdentifier.NumberOfStates, i
+ 2);
    XmlHelper.ConsumeAll(mvObj, doc);
}
```

Auch komplexe BACnet-Properties können mit Hilfe der Bibliothek konfiguriert werden. Im folgenden Beispiel wird die Datelist-Property eines Calendar-Objekts mit jeweils einem DateRange-, Date- und WeekNDay-Eintrag konfiguriert.Die Bibliotheksfunktion WriteProperty wandelt dabei die .NET Klassen bzw. Objekten in serielle Datenströme (Byte-Array) um und speichert sie im HexBin-Format im XmlDocument.

```
ITcSmTreeItem calObj = BACnetServerConfiguration.CreateObject(ipServer, BACnetObjectType.Calendar);
doc = BACnetExtension.GetXml(calObj);
BACnetCalendarEntry[] calEntries = newBACnetCalendarEntry[3];
calEntries[0] = newBACnetCalendarEntry(newBACnetDate(112, 1, 1, 255)); // Add date: 1.Jan.2012
calEntries[1] = newBACnetCalendarEntry(newBACnetDateRange(newBACnetDate(255, 1,1,255), newBACnetDate
(255, 7,19,155))); // DateRange 1.7.-19.7
calEntries[2] = newBACnetCalendarEntry(newBACnetWeekNDay(255, 255, 7)); // every sunday
BACnetObjectConfiguration.WriteProperty<BACnetCalendarEntry[]>(doc, BACnetPropertyIdentifier.Datelis
t, calEntries);
XmlHelper.ConsumeAll(calObj, doc);
```

Im folgenden Beispiel wird ein Schedule-Objekt mit einer Ausnahmeliste konfiguriert, die aktiv wird, wenn das zuvor erstellte Calendar-Objekt aktiv ist. In der Ausnahmeliste wird zur Demonstration ein Blinken mit 10 Hz eines BinaryOutput-Objekt mit der Objekt-ID 10 konfiguriert.

```
// remember calendar object for schedule configuration
BACnetObjectIdentifier calObjId = BACnetObjectConfiguration.ReadProperty<BACnetObjectIdentifier>(doc
, BACnetPropertyIdentifier.ObjectIdentifier);
ITcSmTreeItem schedObj = BACnetServerConfiguration.CreateObject(ipServer, BACnetObjectType.Schedule)
;
doc = BACnetExtension.GetXml(schedObj);

// blink with 10 Hz
BACnetSpecialEvent specialEvent = newBACnetSpecialEvent(calObjId, 1, newBACnetTimeValueList());
    specialEvent.AddBACnetTimeValue(
    newBACnetTimeValue(newBACnetTime(255, 255, 255, 0),
    newBACnetValue(BACnetBinaryPV.active)));
    specialEvent.AddBACnetTimeValue(
    newBACnetTimeValue(newBACnetTime(255, 255, 255, 50),
    newBACnetValue(BACnetBinaryPV.inactive)));

BACnetObjectConfiguration.WriteProperty<BACnetSpecialEventList>(
    doc,
    BACnetPropertyIdentifier.ExceptionSchedule,
    newBACnetSpecialEventList(newBACnetSpecialEvent[] { specialEvent }));
// set object/property reference to BO:0
BACnetObjectConfiguration.WriteProperty<BACnetDeviceObjectPropertyReference[]>(
    doc,
    BACnetPropertyIdentifier.ListOfObjectPropertyRefs,
    newBACnetDeviceObjectPropertyReference[]{newBACnetDeviceObjectPropertyReference(
    newBACnetObjectIdentifier(BACnetObjectType.BinaryOutput, 17), BACnetPropertyIdentifier.PresentVa
lue)});
    XmlHelper.ConsumeAll(schedObj, doc);
```

Prozessdatenkonfiguration

Auch Prozessdaten von BACnet-Objekten können mit Hilfe der Bibliothek aktiviert und mit der Automation Interface Funktionen LinkVariables verknüpft werden. Im Beispiel wird jeweils ein BinaryInput- und BinaryOutput-Objekt erstellt und die PresentValue-Properties über eine Device2Device-Mapping verknüpft. Mit Hilfe der Funktion XmlHelper.GetBaseTypeBitSize kann dabei die Bitlänge der Property-Prozessdaten ermittelt werden, die als Parameter der Funktion LinkVariables benötigt wird.

```
// create to objects to be linked together ( BO is on whenever BI is on )
ITcSmTreeItem pdBiObj = BACnetServerConfiguration.CreateObject(ipServer, BACnetObjectType.BinaryInpu
t);
ITcSmTreeItem pdBoObj = BACnetServerConfiguration.CreateObject(ipServer, BACnetObjectType.BinaryOutp
ut);
// add property PresentValue as input process data
doc = BACnetExtension.GetXml(pdBiObj);
string biVarName = BACnetObjectConfiguration.AddInputSymbolChecked(doc, BACnetPropertyIdentifier.Pre
sentValue, BACnetObjectType.BinaryInput, false, null);
XmlHelper.ConsumeAll(pdBiObj, doc);
// add property PresentValue as output process data with priority 10
doc = BACnetExtension.GetXml(pdBoObj);
string boVarName = BACnetObjectConfiguration.AddOutputSymbolChecked(doc, BACnetPropertyIdentifier.Pr
esentValue_Priority10, BACnetObjectType.BinaryOutput, false, null);
XmlHelper.ConsumeAll(pdBoObj, doc);
// get Property description to get process data size of present value properties
PropEntryDesc eDesc = PropertyDescriptions.GetPropDesc(BACnetPropertyIdentifier.PresentValue);
uint bitLen = XmlHelper.GetBaseTypeBitSize(eDesc.GetProcessDataTmcType(BACnetObjectType.BinaryInput)
);
// link the objects via process data
ipSysMan.LinkVariables(pdBiObj.PathName + "^Inputs^" + biVarName, pdBoObj.PathName + "^Outputs^" + b
oVarName, 0, 0, (int)bitLen);
```

Optionale Properties

Auch optionale Properties können mit Hilfe der Bibliotheksfunktionen EnableProperty und DisableProperty aktiviert und deaktiviert werden. Zusätzlich demonstriert das Beispiel wie über alle Properties eines Objekttyps iteriert werden kann. DisableProperty unterstützt dabei das Deaktivieren zusammenhängender Properties. Wird eine Property deaktiviert, welche die Existenz einer zweiten Property voraussetzt, wird auch diese automatisch deaktiviert. Sollen z.B. alle Properties des Instrinsic Reporting deaktiviert werden, reicht das Deaktivieren der Property NotifyType aus.

```
ITcSmTreeItem bvObj = BACnetServerConfiguration.CreateObject(ipServer, BACnetObjectType.BinaryValue)
;
doc = BACnetExtension.GetXml(bvObj);
```

```
// disable all optional properties
foreach (BACnetPropertyIdentifier prop inPropertyDescriptions.GetSupportedActiveOnlineProps(doc, BAC
netObjectType.BinaryValue, true))
{
    eDesc = PropertyDescriptions.GetPropDesc(prop);
    if (PropertyDescriptions.IsOptProp(BACnetObjectType.BinaryValue, prop))
    PropertyDescriptions.DisableProperty(doc, BACnetObjectType.BinaryValue, prop);
}
XmlHelper.ConsumeAll(bvObj, doc);
```

Schreibschutzkonfiguration

Mit Hilfe der Funktion EnablePropertyWriteProtection kann der Schreibschutz einer Property aktiviert werden.

```
ITcSmTreeItem bv2Obj = BACnetServerConfiguration.CreateObject(ipServer, BACnetObjectType.BinaryValue
);
doc = BACnetExtension.GetXml(bv2Obj);
// enable write protection for all writable properties
foreach (BACnetPropertyIdentifier prop inPropertyDescriptions.GetSupportedActiveOnlineProps(doc, BAC
netObjectType.BinaryValue, true))
{
    eDesc = PropertyDescriptions.GetPropDesc(prop);
    if (PropertyDescriptions.IsWritable(BACnetObjectType.BinaryValue, prop))
    PropertyDescriptions.EnablePropertyWriteProtection(doc, BACnetObjectType.BinaryValue, prop);
}
XmlHelper.ConsumeAll(bv2Obj, doc);
```

Mit Hilfe der Funktion ObjectDescriptions.SupportedOfflineProps kann auch über alle Properties iteriert werden, die im Settings-Dialog der Objekte konfiguriert werden können. Im Gegensatz iteriert GetSupportedActiveOnlineProps über alle Properties eines Objekts die zur Laufzeit existieren.

BACnet-Client Konfiguration

Auch für die Konfiguration von entfernten BACnet-Geräten stehen Funktionen zur Verfügung. CreateBACnetClient erstellt einen BACnet-Client mit angegebener BACnet-ID. Mit Hilfe der Funktion CreateRemoteObject können BACnet-Objekte erstellt werden. Auch Prozessdaten von Client-Objekten können aktiviert und mit Hilfe der Automation Interface Funktion LinkVariables verknüpft werden. Dabei muss den Funktionen AddInputSymbol und AddOutputSymbol eine RemotePdCfg übergeben werden, welche den Modus der Prozessdatenbehandlung konfiguriert. RemotePdCfg fasst Parameter für lesende und schreibende Property-Zugriffe zusammen, das heißt die Remote-Konfiguration kann einmal erstellt und dann für das Anlegen von Input- und Output-Symbolen verwendet werden. Die Modi entsprechen dabei den im System Manager einstellbaren Optionen (Polling, COV, WriteOnChange usw.). Durch die Parameter RemotePdReadTickModulo und RemotePdWriteTickModulo kann eingestellt werden in welchen Zyklus Client-Interaktionen stattfinden sollen. Es empfiehlt sich bei vielen Client-Objekten die Interaktionen über mehrere Zyklen zu verteilen.

Funktionen mit dem Suffix Checked überprüfen vor dem Aktivieren einer Prozessdatenvariable, ob diese schon aktiviert wurde. Ggf. wird sie deaktiviert bevor sie neu aktiviert wird.

```
ITcSmTreeItem ipClient = BACnetDeviceConfiguration.CreateBACnetClient(ipDevice, 42);
// create BACnet remote object
ITcSmTreeItem remAV = BACnetClientConfiguration.CreateRemoteObject(ipClient, BACnetObjectType.Analog
Value, 17);
doc = BACnetExtension.GetXml(remAV);
// add input symbol - read data from remote device
BACnetObjectConfiguration.AddInputSymbolChecked(
    doc,
    BACnetPropertyIdentifier.PresentValue,
    BACnetObjectType.AnalogValue,
    true,
    newBACnetObjectConfiguration.RemotePdCfg(BACnetObjectConfiguration.RemotePdCfg.RemotePdReadMode.
COV_POLL, 1000, 1.0f, 240));
// add output symbol - with explicit remote configuration
// write property present value on change with priority 10
BACnetObjectConfiguration.RemotePdCfg remoteCfg = newBACnetObjectConfiguration.RemotePdCfg();
remoteCfg.WriteMode = BACnetObjectConfiguration.RemotePdCfg.RemotePdWriteMode.ONCHANGE;
remoteCfg.UsePriority = true;
remoteCfg.priority = 10;
remoteCfg.RemotePdWriteTickModulo = 2;
remoteCfg.resubsInterval = 240;
remoteCfg.cycleTimeWrite = 2000;
BACnetObjectConfiguration.AddOutputSymbolChecked(
    doc,
```

```
    BACnetPropertyIdentifier.PresentValue,
    BACnetObjectType.AnalogValue,    true,
    remoteCfg);
XmlHelper.ConsumeAll(remAV, doc);
```

Notification Sink Konfiguration

Auch eine Notification Sink kann mit Hilfe der BACnetExtensionCtrl-Bibliothek erstellt werden. Im folgenden Beispiel wird eine Notification Sink mit der Prozess-ID 100 erstellt. Anschließend wird ein NotificationClass-Objekt erstellt und zum Senden lokaler EventNotifications an die erstellte Notification Sink konfiguriert.

```
ITcSmTreeItem ipSink = BACnetDeviceConfiguration.CreateNotificationSink(ipDevice);
doc = BACnetExtension.GetXml(ipSink);
BACnetNotificationSinkConfiguration.SetProcessId(doc, 100);
XmlHelper.ConsumeAll(ipSink, doc);
ITcSmTreeItem ipNotifyObj = BACnetServerConfiguration.CreateObject(ipServer, BACnetObjectType.Notifi
cationClass);
doc = BACnetExtension.GetXml(ipNotifyObj);
BACnetDestination[] destArr = newBACnetDestination[1];
destArr[0] = newBACnetDestination();
destArr[0].processIdentifier = 100;
destArr[0].recipient = newBACnetRecipient(newBACnetObjectIdentifier(BACnetObjectType.Device, 123));
BACnetObjectConfiguration.WriteProperty<BACnetDestination[]>(doc, BACnetPropertyIdentifier.Recipient
List, destArr);
XmlHelper.ConsumeAll(ipNotifyObj, doc);
```

Abschließend kann die erstellte Konfiguration abgespeichert werden. Das Abspeichern erfolgt mit Hilfe der Automation Interface Funktionen SaveConfiguration.

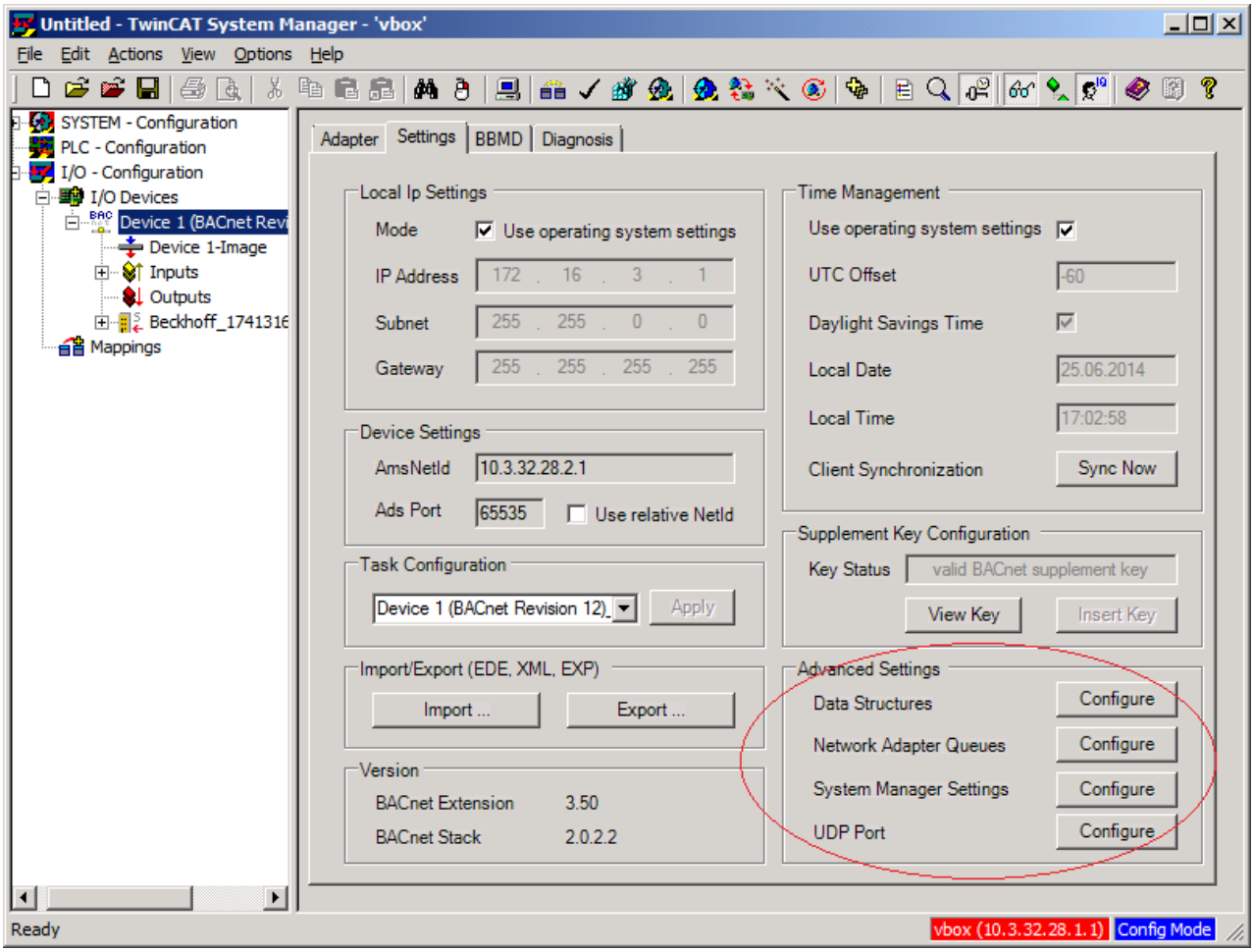
```
ipSysMan.SaveConfiguration("C:\\BACnetExample.tsm");
```

Alle in diesem Kapitel vorgestellten Quellcodeausschnitte finden sich als ausführbares Visual Studio Projekt in folgender .zip-Datei. Das Projekt kann dabei auch als Startpunkt für ein eigenes Projekt verwendet werden.

https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749058955.zip https://infosys.beckhoff.com/content/1031/tcbacnet/Resources/12749058955.zip

6.5 Erweiterte Einstellungen

Im "Settings"-Reiter des BACnet-Device können erweiterte Einstellungen (*Advanced Settings*) vorgenommen werden, die im Folgenden erläutert werden sollen.



Speicherspezfische Parameter

Über die erweiterten Einstellungen können datenstrukturspezifische Parameter (Data Structures) angepasst werden. Im Wesentlichen beeinflussen diese Parameter den Speicherverbrauch des BACnet-Stack.

Parameter	Default-Wert (Min.-Max.)	Beschreibung
MAX_OBJ_SUBS_COV_ENTRIES	10 (1 - 255)	Die COV-Subscriptions werden in TwinCAT BACnet/IP pro Objekt verwaltet. Per Default sind 10 Subscriptions pro BACnet-Objekt möglich. Mit Hilfe dieses Parameters kann die maximal mögliche Anzahl von COV-Subscriptions pro Objekt festgelegt werden. Aus Performancegründen kann es auch Sinn machen, den Wert dieses Parameters zu verringern. Damit beim Auftreten vieler Änderungen im System nicht zu viele COV-Notifications erzeugt werden.
MAX_PROPERTY_ARRAYELEMENTS	200 (10 - 10000)	Bei BACnet-Properties mit Array-Datentypen in TwinCAT BACnet/IP (Datentyp endet auf []) wird beim Start Speicher für eine zuvor definierte Anzahl von Elementen angelegt damit während des Betriebs dynamisch Elemente hinzugefügt werden können. Per Default können maximal 200 Elemente in einer Array-Property gespeichert werden. Werden schon im Settings-Dialog mehr als 200 Elemente in eine Array-Property konfiguriert, wird bei Start automatisch mehr Speicher angelegt, um alle Elemente aufnehmen zu können.

Parameter	Default-Wert (Min.-Max.)	Beschreibung
LIST_ALLOC_MEM_BYTES	8192 (500 - 4294967295)	Bei Properties mit Listen-Datentyp in TwinCAT BACnet/IP (Datentyp endet auf List) kann der pro Element benötigte Speicher variieren. Um auch bei diesen Properties dynamisch Elemente hinzufügen und verändern zu können, wird beim Start ein festgelegter Speicher angelegt. Per Default werden 8 kByte angelegt. Über diesen Parameter kann der Speicher pro Listen-Property definiert werden. Wie auch bei Array-Properties gilt, wenn im Settings-Dialog die Property mit mehr benötigtem Speicher konfiguriert wird, wird auch entsprechend mehr Speicher angelegt.
MAX_DEVICE_BINDINGS	1000 (10 - 10000)	Jedes Gerät im BACnet-Netzwerk (das ein I-Am Paket sendet) wird in der DeviceBindings-Tabelle verwaltet. Dort werden alle für die Kommunikation mit einem Gerät benötigten Parameter gespeichert. Die Größe dieser Tabelle kann über diesen Parameter festgelegt werden. Die Tabelle wird bei jedem Neustart und beim Auslösen eines Scan-Vorgangs (z.B. im System Manager) gelöscht.
MAX_OBJECTS	1000 (10 - 4294967295)	BACnet-Objekte pro BACnet-Client und BACnet-Server werden jeweils in einer Hash-Tabelle verwaltet. Dieser Parameter gibt die Größe dieser Tabellen an. Sollen in einer Konfiguration BACnet-Server und BACnet-Clients mit mehr als 1000 Objekten angelegt werden, muss dieser Parameter erhöht werden. Die Anzahl der Objekte bezieht sich auf die Summe aller unter BACnet-Server und BACnet-Clients projektierten Objekte und muss entsprechend gewählt werden. Beim Hinzufügen von BACnet-Objekten im System Manager sowie beim Einscannen von BACnet-Clients wird dieser Parameter automatisch in 100er Schritten inkrementiert. Soll zur Laufzeit das Anlegen vieler dynamischer Objekte unterstützt werden, sollte dieser Parameter manuell erhöht werden.
MAX_CLIENTS	255 (10 - 10000)	Die im System Manager angelegten Clients werden in einer Liste verwaltet. Soll mit mehr als 255 (statisch im System Manager konfigurierten) Clients gearbeitet werden, muss dieser Parameterwert erhöht werden.
MAX_TRENDLOG_ENTRYSIZE	56 (56 - 10000)	Auch bei TrendLog-Objekten wird der Speicher für den LogBuffer beim Start angelegt. Mit dem Parameter BufferSize kann die Anzahl der Einträge als BACnet-Property festgelegt werden. Der für einen Eintrag benötigte Speicher hängt von der geloggten Property ab. Die minimale Größe von 56 Byte ist ausreichend für FehlerLog-Einträge und für das Loggen einfacher Datentypen (REAL, Integer ...). Erscheint im LogBuffer der Eintrag: **"Resources:No_space_to_write_property"**, reicht der verfügbare Speicher nicht aus und der Wert dieses Parameters muss erhöht werden.
DYN_TRENDLOG_BUFFERSIZE	100 (10 - 10000)	Da der Speicher des TrendLog-LogBuffers beim Erzeugen des BACnet-Objekts angelegt wird, muss für dynamisch erzeugte TrendLog-Objekte

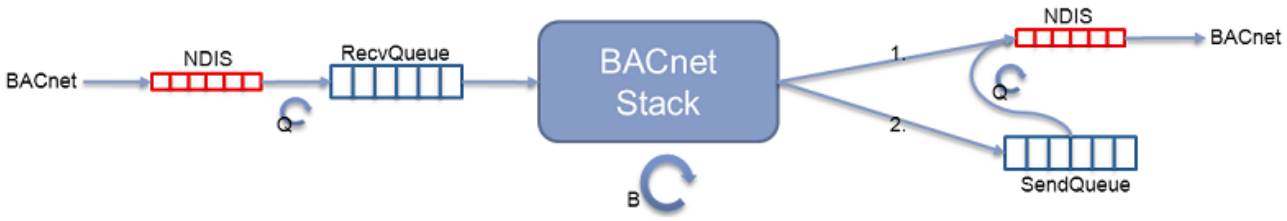
Parameter	Default-Wert (Min.-Max.)	Beschreibung
		bekannt sein, welchen Wert die BACnet-Property BufferSize besitzen soll. Dies wird über diesen Parameter festgelegt.
MAX_MULTISTATE_DYNSTATES	255 (2 - 10000)	Bei MultiState-Objekten muss für die BACnet Property StateText beim Start Speicher angelegt werden. Hierfür muss die maximal mögliche Anzahl der Zustände (NumberOfStates) bekannt sein, da die BACnet-Property NumberOfStates während der Laufzeit verändert werden kann. Dieser Parameter legt fest, wie der Maximalwert der Property NumberOfStates sein kann und damit wie viel Speicher beim Start angelegt werden muss.
MAX_RECV_BAC_SEGM_FRAMES	1000 (100 - 10000)	Segmentierte Frames müssen in ihren Bestandteilen zwischengespeichert werden, um nach dem Empfang aller Bestandteile zur ursprünglichen Nachricht vereinigt werden zu können. Je nach der angestrebten parallelen Verarbeitung segmentierter Nachrichten, muss dieser Parameter entsprechend eingestellt werden.
MAX_SEND_BAC_SEGM_FRAMES	1000 (100 - 10000)	Auch beim Senden müssen Nachrichten potenziell segmentiert werden. Zum Versenden vorbereitete Frames werden in der Tabelle gespeichert. Die Größe dieser Tabelle ist über diesen Parameter einstellbar.
IO_STARTUP_TIMEOUT	2000 (0 - 10000)	Die Initialisierung eines angeschlossenen Feldbusses (K-BUS, EtherCAT) kann potenziell länger dauern, als das Starten des BACnet-Servers. Dies kann dazu führen, dass über die automatische Feldbusüberwachung (Reliability, StatusFlags) alle IO-BACnet-Objekte in einen Fehlerzustand setzt, was zum einen Trendlog-Einträge mit gesetztem Fault-Bit in den StatusFlags erzeugt, aber auch potenziell Notifications versendet werden. Deshalb kann die StateMachine des BACnet Server um eine IO-Start-Zeit verzögert werden, um die erfolgreiche Initialisierung des verwendeten Feldbusses beim Start abzuwarten.
MAX_PARALLEL_COV_SUBS	250 (1 - 255)	Bei bei Verwendung von Client-Prozessdaten im Modus COV kann es bei langsamen Gegenstellen zu Kommunikationsausfällen kommen bzw. COV-Anmeldungen nicht vollständig ausgeführt werden. Mit diesem Parameter kann die maximale Anzahl gleichzeitiger COV-Subscriptions-Requests pro BACnet-Client festgelegt werden.
AVERAGING_MAX_BUFFERSIZE	1000 (1 - 10000)	Legt die Größe des internen Puffers beim Averaging fest. Dieser Puffer speichert Daten zur Berechnung statistischer Daten (Mittelwert etc.). Sollen mehr als 1000 Samples als Basis für die Berechnung dienen, muss dieser Parameter erhöht werden.
MAX_SENDFRAMES_PERCYCLE	4294967295 (1 - 4294967295)	Um temporäre Spitzenlasten an gesendeten BACnet-Telegrammen zu begrenzen, kann die Anzahl zu sendender Frames pro Zyklus eingeschränkt werden. Bei einer Zykluszeit von 50 ms und einem Wert von 1 werden dann maximal 20 Frames/sec. versendet. Dieser Mechanismus

Parameter	Default-Wert (Min.-Max.)	Beschreibung
		kann dazu dienen ein Fluten des BACnet-Netzes in bestimmten Fällen zu verhindern. Generell gilt aber, dass besser die Ursache vieler gesendeter Frames gefunden und beseitigt werden sollte. Z.B. durch den Einsatz von Filtern für AnalogInput-Objekte.
PROPOSED_WINDOW_SIZE	16 (1 - 127)	Bei der Übertragung großer Datenmengen wird in BACnet Segmentierung eingesetzt. Die maximale Anzahl offener unbestätigt gesendeter Nachrichten wird bei beim Verbindungsaufbau "ausgehandelt". Dieser Parameter bestimmt, welche "Fenstergröße" (also der Zahl unbestätigter Nachrichten) von diesem Gerät vorgeschlagen werden. Beim CX8091 muss dieser Wert auf 4 reduziert werden, da ansonsten Dateizugriffe auf Grund kleiner Netzspeicher nicht funktionieren.
FREERUN_CYCLETIME_MODULO	10 (1 - 100)	Im Freerun werden die internen Zustandsmachinen der BACnet-Objekte mit einer Zykluszeit von 2 ms aufgerufen. Bei großen BACnet-Konfigurationen kann dies zu einer Überlastung des Systems führen. Dieser Parameter legt fest, alle wie viel Zyklen BACnet-Objekte aufgerufen werden. Im Default-Fall (10) werden die BACnet-Objekte also nur alle 20 ms verarbeitet.
ARCHIVE_BUFFER_SIZE	65535 (65535 - 16777216)	Beim Speichern persistenter Daten wird ein Zwischenspeicher verwendet. Dieser Puffer sollte mindestens so groß wie die größte Property im System sein. Im typischen Fall handelt es sich dabei um die Property LogBuffer eines TrendLog-Objektes. Bei einer Größe von 40 Byte pro Eintrag (normaler Wert; kein Fehler) wird der Default-Wert 65535 dieses Parameters ab 1638 Einträgen überschritten. In diesem Fall sollte dieser Parameter erhöht werden. Bei einem LogBuffer von 3000 Einträgen sollte der Wert also auf 120.000 Byte eingestellt werden (40*3000)

Netzwerk-Adapter-Queues

Kleine Veränderungen können in BACnet leicht zum Versenden vieler Nachrichten führen. Z.B. weil viele COV-Subscriptions auf eine bestimmte Property existieren, oder viele *Notification-Recipients* in einer NotificationClass eingetragen wurden. Auf Grund dieser asynchronen Natur wird BACnet in einer niederprioren Task ausgeführt, um Verzögerungen der SPS bzw. des IO-Subsystems auszuschließen. Bei großen Konfigurationen sollte die Zykluszeit dieser Task im Bereich 40ms - 100ms konfiguriert werden. Diese Bearbeitung "im Hintergrund" führt, dazu das Netzwerkadapterfunktionen seltener ausgeführt werden. Beim Empfangen von Nachrichten können hierdurch Treiber-interne Puffer überlaufen und damit Nachrichten verloren gehen. Auch beim Senden von BACnet-Nachrichten kann es zu Überlast-Situationen der Netzwerk-Hardware kommen, wenn innerhalb eines Zyklus zu viele Nachrichten versendet, werden müssen.

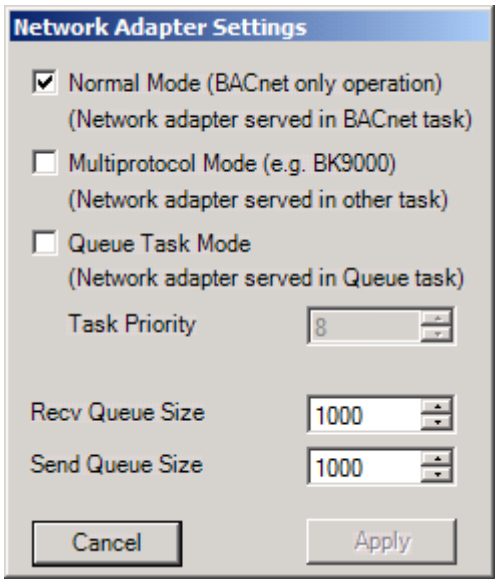
Für die Verarbeitung der Netzwerknachrichten stehen deshalb in BACnet verschiedene Modi zur Verfügung, um die Funktionalität des Netzwerkadapters zu optimieren. Um Überlasten sowohl beim Senden als auch Empfangen abzufangen verwendet BACnet einen Sende- und einen Empfangspuffer deren Größe separat festgelegt werden kann. In der folgenden Abbildung ist die Arbeitsweise dieser "Queues" illustriert. Auf linke Seite werden BACnet-Nachrichten z.B. über den NDIS-Treiber empfangen. Eine Task (Q) kopiert diese in die Recv-Queue. Diese Nachrichten werden dann im Kontext der BACnet-Task (B) verarbeitet und potenziell Nachrichten versendet. Wenn genügend Ressourcen im Netzwerkadapter zur Verfügung stehen wird die Nachricht direkt versendet (1.); ansonsten werden Nachrichten in der Send-Queue zwischengespeichert und im Kontext der Task Q abgesendet.



BACnet unterstützt 3 Netzwerk-Adapter-Modi, die sich im Wesentlichen dadurch unterscheiden im Kontext welcher Task (im Bild Q) die Funktionen zum Empfangen und Senden des Netzwerk-adapters aufgerufen werden:

- Im **Normal Modus** werden netzwerk-adapter-spezifische Funktionen im Kontext der BACnet-Task selbst ausgeführt. Zum Zyklusbeginn werden auch hier alle Nachrichten in den Empfangspuffer kopiert und im weiteren Verlauf von dort verarbeitet.
- Im **Multiprotocol Modus** werden netzwerk-adapter-spezifische Funktionen im Kontext einer anderen Task ausgeführt. Dieser Modus muss verwendet werden, wenn parallel zu BACnet weitere ethernet-basierte Treiber (z.B. Real-Time Ethernet) über die gleiche Netzwerkschnittstelle betrieben wird.
- Im **Queue-Task Modus** werden netzwerk-adapter-spezifische Funktionen im Kontext einer speziellen Queue-Task ausgeführt. Dieser Modus findet z.B. Anwendung, wenn die BACnet-Task mit hohen Zykluszeiten betrieben wird. Die hoch-priore Queue-Task mit niedriger Zykluszeit kopiert die Nachrichten vom Netzwerktreiber in die BACnet-interne Recv-Queue bzw. versendet Nachrichten der Send-Queue. Hierdurch werden die Hardwareressourcen schneller bedient, aufwendigen Berechnung aber trotzdem nieder-prior ausgeführt.

Über den Dialog "Network Adapter Settings" können netzwerk-adapter-spezifische Einstellungen vorgenommen werden. Die Größe der Sende- und Empfangsqueue kann separat festgelegt werden. Zusätzlich kann die Priorität der Queue-Task kann festgelegt werden und natürlich der Modus für den Netzwerk-Adapter-Betrieb ausgewählt werden.



Die einzelnen Modi sollten nach den folgenden Empfehlungen gewählt werden:

- Normal Mode
  - Kurze BACnet-Zykluszeiten (<20 ms)
  - Leistungsstarkes Gerät (CX50XX)
- Multiprotocol Mode
  - Wenn Real-Time-Ethernet verwendet wird!
  - Triggerzyklus sollte nicht zu lang sein (<10 ms)
- Queue Task Mode
  - Leistungsschwache Geräte (CX9010, CX8091, CX9020)
  - Immer bei CCAT (CX8091)

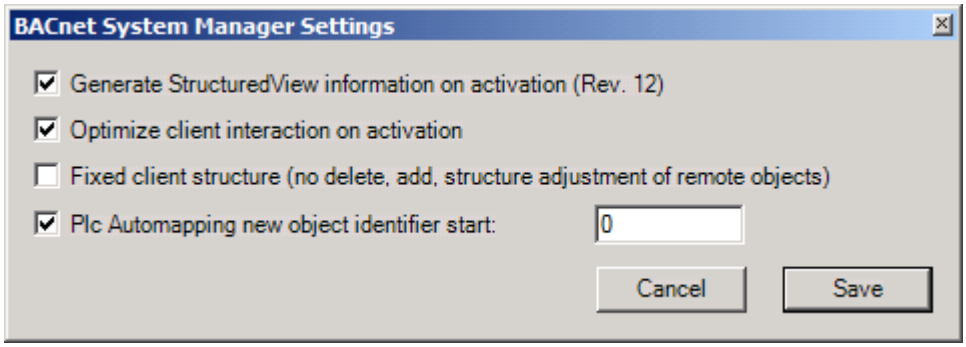
- Lange BACnet-Zykluszeit (>20 ms)
- Queue-Task am besten mit 1 ms (kostet ca. 5-10% Performance)

Typische Einstellung CX8091

BACnet-Task: 50ms – 100ms

SystemManager Settings

Im Dialog "BACnet System Manager Settings" können BACnet-spezifische Einstellung für die Konfigurations-Software - den TwinCAT SystemManager vorgenommen werden.



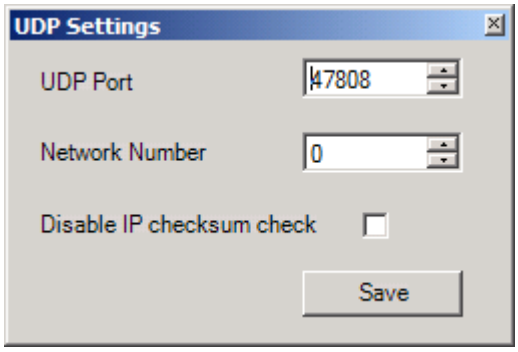
Über den Punkt **"Generate ..."** kann festgelegt werden, ob beim Aktivieren einer BACnet-Konfiguration StructuredView-spezifische Property-Inhalte automatisch erzeugt werden. Hierdurch ist es möglich die hierarchische Darstellung im SystemManager auch im BACnet-Netzwerk anzuzeigen. Diese Punkt ist per Default aktiviert, dass heißt die StructuredView-Informationen werden erzeugt. StructuredView ermöglichen prinzipiell die Darstellung mehrerer Sichtweisen (z.B. auf Aggregatebene, Ortsbezogen oder kaufmännisch). Sollen in einem Projekt spezifischen Vorgaben für StructuredView-Objekte implementiert werden, die über die reine Ordnung von BACnet-Objekten hinausgehen, kann dieser Punkt deaktiviert und StructuredView-Objekte manuell konfiguriert werden.

Über den Punkt **"Optimize ..."** kann festgelegt werden, ob die Interaktions-Modulo-Faktoren für die Interaktion mit entfernten BACnet-Geräten automatisch optimiert werden soll. Diese Funktion ist per Default aktiviert.

Über den Punkt **"Fixed ..."** kann festgelegt werden, dass die Objekt-Struktur unter einem BACnet-Client beim Scannen nicht verändert wird. Wenn nur wenige Objekte in die Querkommunikation integriert sind, kann es sinnvoll über das PLC-Automapping nur die verwendeten BACnet-Objekte anzufügen. Wenn dieser Punkt aktiviert wird, wird beim Scannen ein Abgleich der Objekt-Identifier anhand des BACnet-Objektnamen durchgeführt. Wenn hier Änderungen festgestellt werden, können neue ObjectIdentifier übernommen werden. Sinnvoll ist diese Funktion, auch wenn zum Projektierungszeitpunkt die ObjectIdentifier noch nicht feststehen, sondern nur die Objektnamen.

Über den Punkt **"Plc ..."** kann festgelegt werden, ab welcher Instanznummer beim PLC-Automapping automatisch generierte Objekt-Identifier beginnen. Diese Funktionalität wird für die Funktion Remap benötigt, um Kollisionen mit veralteten persistenten Daten (.wbp) zu vermeiden.

UDP-Port Settings



Über den Dialog "UDP Settings" kann der von BACnet verwendete UDP-Port festgelegt werden.

Zusätzlich kann hier die BACnet-Netzwerk-Nummer eingestellt werden. In reinen BACnet/IP ist die Konfiguration der Netzwerknummer in dem meisten Fällen nicht notwendig. Wenn zu Zwecken der zusätzlichen Aussplittung eines BACnet/IP-Netzwerkes die Netzwerk-Nummer auf einen Wert ungleich 0 festgelegt wird, dann wird im BACnet-Stack folgendes Verhalten umgesetzt:

- Netzwerknummer wird im I-Am-Frame versendet
- Es werden nur BACnet-Frames ohne Netzwerknummer bzw. mit der konfigurierten Netzwerknummer akzeptiert
- Optional wird die konfigurierte Netzwerknummer als Source-Adresse übertragen

Über den Punkt "*Disable IP checksum check*" kann die Überprüfung der IP-Checksumme deaktiviert werden. Diese Funktion sollte nur aktiviert werden, wenn unbedingt mit Geräten kommuniziert werden muss, die keine korrekte Checksummenberechnung umsetzen.

6.6 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

Mehr Informationen:
www.beckhoff.de/ts8020

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com