Dokumentation zur Lynus HTTP/S Bibliothek für die Beckhoff Steuerungen

Inhalt

Version	Datum	Bearbeiter/Ersteller	Äderung/Ergänzung	Firma
1.0.0.0	25.01.2022	Kai Ebensperger	Dokument erstellt	Lynus AG

1. Systemvoraussetzungen

Die Lynus Bibliothek funktioniert auf allen Beckhoff Steuerungen welche TwinCat 3.1 4022.0 oder höher installiert haben. Für diese Bibliothek werden Beckhoff Funktionen verwendet welche standartmäßig bei der Installation von TwinCat 3.1 dabei sind. Zusätzlich dazu brauch es noch die TC3 TF6760 Function (HTTP/Rest).

Diese Bibliothek funktioniert nur in Verbindung mit der Lynus Communicator Bibliothek und einer aktiven Verbindung zu einem erzeugten Projekt im Lynus Dashboard. Gibt es keine aktive Verbindung zu einem Cloud Projekt, stoppen die Funktionalitäten dieser Bibliothek automatisch nach 7 Tagen betrieb.

Diese Bibliothek verlangt die Installation der Lynus Standard Bibliothek.

2. Einführung

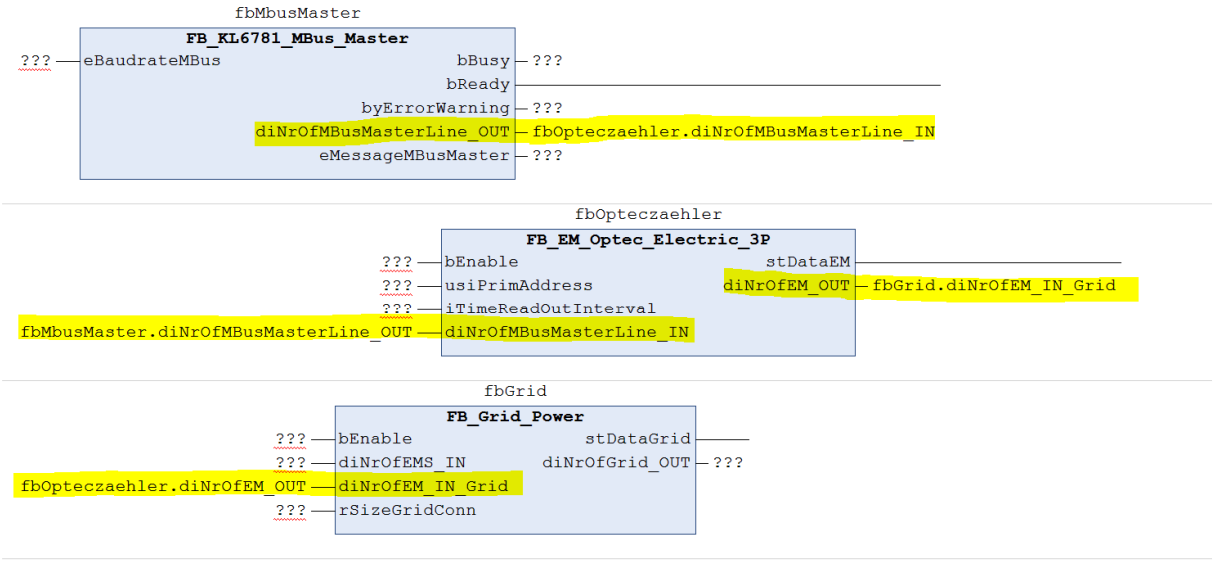
In der Lynus HTTP Bibliothek befinden sich Funktionsblöcke und Datentypen welche vor allem für die Kommunikation zu einem übergeordneten Regelsystem oder GLT dienen. Man findet hier ausschliesslich Funktionsblöcke die die spezifische Schnittstelle HTTP verwenden. Unter anderem findet man hier Funktionsblöcke zu Gebäudeleitsystemen oder externen Clouddiensten jeglicher Art.

3. Funktionsweise

Die meisten Funktionsblöcke sind so aufgebaut, dass Sie die Daten wie oben genannt über verschiedene Schnittstellen aufnehmen und dann intern auf eine eigene Schnittstelle legen. Diese Daten werden dann intern weitergegeben und stehen somit anderen Funktionsblöcken zur Verfügung. Somit hat man z.B. die Möglichkeit die Energiedaten einer klassischen M-Bus Messung an einen spezifischen «Netzanschluss» oder «PV-Anlage» Funktionsblock weiterzugeben und diese dann auch als solche zu spezifizieren. Somit ist man extrem Flexibel was den Datenaustausch unter den Funktionsblöcken betrifft, hat aber gleichzeitig schon fix fertige Funktionsblöcke die Plug&Play verwendet werden können. Diese Übergabe der Daten kann über eine einfache Nummer welche das System selber berechnet erledigt werden.

3.1 Beispiel

Hier wird das ganze anhand einer Energiemessung eines M-Bus Elektrozählers erklärt, welcher im realen Umfeld die Leistung am Netzanschluss misst. Die Bezeichnungen unterschieden sich voneinander im Text, beinhalten aber immer einmal die Bezeichnung _IN oder _OUT. _IN bedeutet dass hier die Nummer übergeben kann, von welchem Funktionsbaustein dieser hier die Daten empfangen soll. _OUT bedeutet dass dieser Funktionsbaustein die daten versendet und anderen am _IN Eingang bereitstellt.

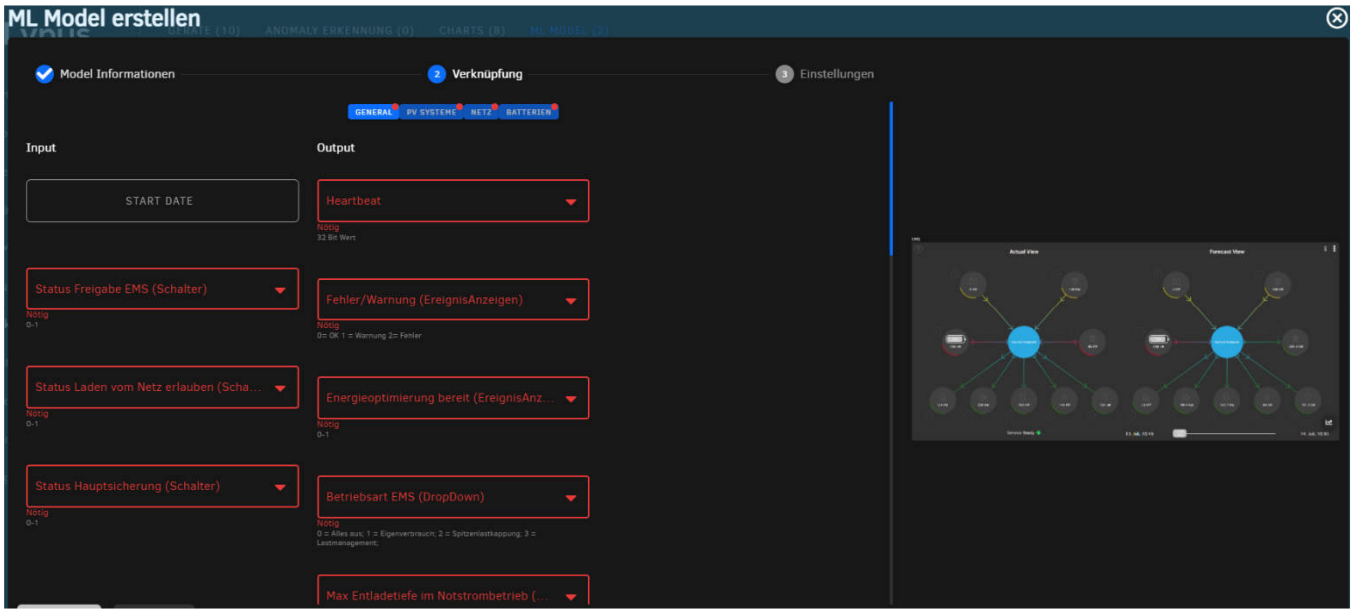


Hier sieht man recht gut, wie der M-Bus Masterbaustein, welcher die Kommunikation zur Beckhoff M-Bus Klemme übernimmt über den Ausgang diNrOfMBusMasterLine_OUT die Daten weitergibt an den M-Bus Zähler von Optec über den Eingang diNrOfMBusMasterLine_IN. Dieser Baustein gibt die Daten dann über den Ausgang diNrOfEM_OUT an den Eingang diNrOfEM_IN_Grid des Netanschluss Funktionsblockes. **Diese Ein und Ausgänge müssen bei allen Funktionsblöcken richtig zugewiesen werden, um eine Störungsfreie Funktion zu gewährleisten.**

3.2 Verknüpfungen im Dashboard

Bestimmte Geräte im Lynus Dashboard benötigen bestimmte Verknüpfungen zu Variablen damit diese auch ordnungsgemäss funktionieren. Um es dem Kunden einfacher zu machen haben bereits alle Funktionsblöcke aus dieser Beschreibung die zu einem spezifischen Device im Dashboard gemappt werden können alle nötigen Variablen dazu. Diese sind durch das klassische Lynus Kommentar markiert im Funktionsbaustein. Diese werden dann automatisch in das Projekt hochgeschickt sobald die SPS über den Communicator mit dem Backend verbunden ist. Dazu muss allerdings mit dem TMC File des jeweiligen Projektes gearbeitet werden, um die Variablen direkt aus den Funktionsblöcken verwenden zu können. Siehe dazu mehr in der Beschreibung für den Lynus Communicator.

Nötiges Variablenmapping im Dashboard



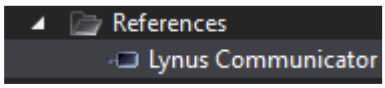
Variablen im Funktionsblock markiert über das Kommentar

FUNCTION_BLOCK **FB_EMS**

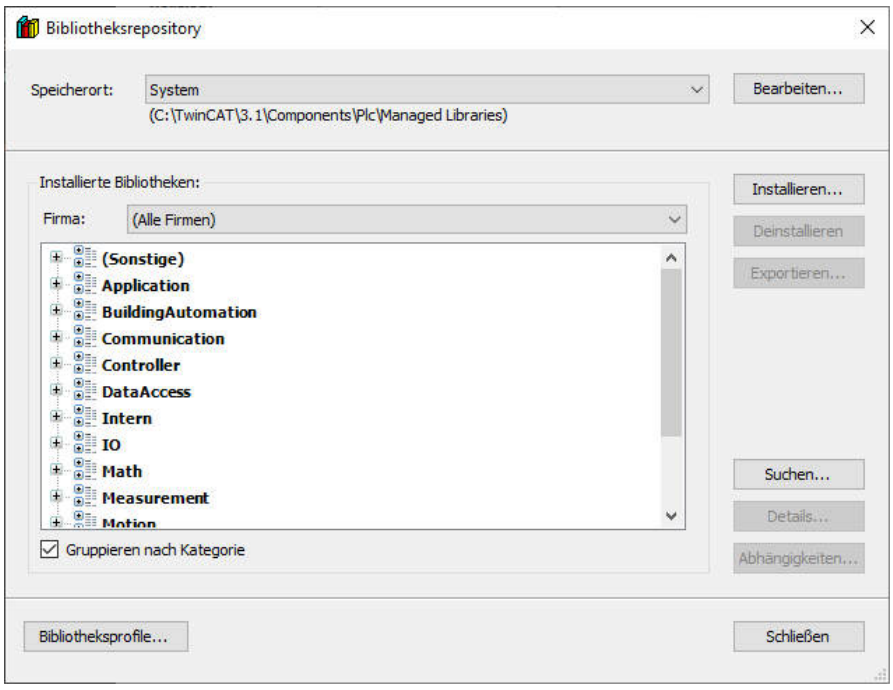
Name	Datentyp	Geerbt von	Adresse	Initialwert	Kommentar
bEnable	BOOL				{#lynus.ag#()} //Enable and dissable this function
diNrOfEM_IN_ECS	DINT				Number of the electric meter for incoming power data. (Its for the complete consumption of all charging stations)
stSetupEMS	ST_Setup_EMS			STRUCT (uiUpdateTime := 15)	{#lynus.ag#()} //Setup EMS
eOperationMode	E_OperatingMode_EMS				{#lynus.ag#()} //Operation Mode EMS
eSpeedModeBackendController	E_SpeedModeBackendController_EMS				Speed for the Controller what discharge the Battery when the commands come from the Backend
bActivateUP	BOOL				{#lynus.ag#

4.Installation der Bibliothek

Nachdem die Bibliothek über den erstellten Account heruntergeladen wurde, im SPS Projekt rechtsklick auf References und dann klick auf Bibliotheksrepository = >



Danach auf installieren klicken = >



Die Lynus Bibliothek im abgespeicherten Pfad auswählen und dann auf Öffnen klicken. Nach erfolgreicher Installation erscheint die Bibliothek im Ordner „Lynus AG“ = >

5. Verwendete Datentypen

Anbei findet man die Beschreibung der Datentypen welche ausschliesslich in dieser Bibliothek verwendet werden. Alle anderen Datentypen findet man in der Lynus Standards Bibliothek. **Achtung :** Es ist zu beachten, dass je nach verwendetem Funktionsblock oder dessen Konfiguration, nicht immer alle Variablen der Datentypen mit Werten belegt sind.

5.1 ST_ListOfVariables_Auhofcenter

Diese Struktur enthält gängige Ausgabe und Eingabe Werte zum Gebäudeleitsystem der HLK im Einkaufszentrum Auhofcenter in Wien. Dies ist eine spezifische Struktur genau für dieses System.

Name	Beschreibung
bIsString	Diese Variable gibt an ob es sich beim Datenpunkt um einen Datentyp String handelt oder ob es ein Numerischer Wert ist.
sVariableName	Namen des Datenpunktes auf dem GLT welcher gelesen oder beschrieben werden soll.
sVariableValue	Wert des Datenpunktes auf dem GLT

5.2 ST_ListOfValues_SmartMe_ElectricMeter

Diese Struktur enthält gängige Ausgabewerte für einen Elektrozähler welcher über das SmartME Cloud System ausgelesen werden soll. Dies ist eine spezifische Struktur genau für dieses System.

Name	Beschreibung
sID	Enthält die ID vom Gerät aus dem SmartMe Cloud Projekt.
sName	Enthält den Namen vom Gerät wie es im SmartME Cloud Projekt benannt wurde.
sActivePowerUnit	Enthält die Einheit für die Wirkleistung die vom Elektrozähler übermittelt werden.
sCounterReadingUnit	Enthält die Einheit der Zählerstände die vom Elektrozähler übermittelt werden.
sValueDate	Datum und Uhrzeit wann der Elektrozähler das letzte mal Daten an die SmartME Clous gesendet hat.
bNotAllDataIncludes	Diese Variable ist gesetzt wenn der Elektrozähler nicht alle Daten die in dieser Struktur enthalten sind liefert.
bDigitalOutput1	Zeigt den Status des 1 Digitalen Ausganges vom Elektrozähler an.
bDigitalOutput2	Zeigt den Status des 2 Digitalen Ausganges vom Elektrozähler an.

udiActiveTariff	Zeigt an aktuellen aktiven Tariff vom Elektrozähler an.
uliSerialNr	Hier wir die Seriennummer vom Elektrozähler angezeigt.
lrPowerFactor	Leistungsfaktor welcher am Elektrozähler gemessen wird.
lrVoltageL1	Spannung an L1 in V
lrVoltageL2	Spannung an L2 in V
lrVoltageL3	Spannung an L3 in V
lrCurrentL1	Strom an L1 in A
lrCurrentL2	Strom an L2 in A
lrCurrentL3	Strom an L3 in A
lrActivePower	Momentane totale Wirkleistung in sActivePowerUnit die am Elektrozähler gemessen wird.
lrActivePowerL1	Momentane Wirkleistung in sActivePowerUnit an L1 die am Elektrozähler gemessen wird.
lrActivePowerL2	Momentane Wirkleistung in sActivePowerUnit an L2 die am Elektrozähler gemessen wird.
lrActivePowerL3	Momentane Wirkleistung in sActivePowerUnit an L3 die am Elektrozähler gemessen wird.
lrCounterReading	Siehe SmartMe Dokumentation
lrCounterReadingT1	Siehe SmartMe Dokumentation
lrCounterReadingT2	Siehe SmartMe Dokumentation
lrCounterReadingT3	Siehe SmartMe Dokumentation
lrCounterReadingT4	Siehe SmartMe Dokumentation
lrCounterReadingImp	Totaler Zählerstand für Import in sCounterReadingUnit
lrCounterReadingExp	Totaler Zählerstand für Export in sCounterReadingUnit

5.3 ST_Values_Energy_Surstoffi

Diese Struktur enthält gängige Ausgabewerte für die Grafan Schnittstelle welche in der Surstoffi in Zug verwendet wird. Dies ist eine spezifische Struktur genau für dieses System.

Name	Beschreibung
sTimestamp	Zeitstempel wann sich der Wert das letzte Mal geändert hat.
sID	ID vom Datenpunkt welcher empfangen wurde.
lrValue	Enthält den aktuellen Wert des Datenpunktes.

sCounterReadingUnit	Enthält die Einheit der Zählerstände die vom Elektrozähler übermittelt werden.
sValueDate	Datum und Uhrzeit wann der Elektrozähler das letzte mal Daten an die SmartME Clous gesendet hat.

5.4 ST_Values_Checked_Surstoffi	
Diese Struktur enthält gängige Ausgabewerte für die Iosys Schnittstelle von B&R welche in der Surstoffi in Zug verwendet wird. Dies ist eine spezifische Struktur genau für dieses System.	
Name	**Beschreibung**
sTimestamp	Zeitstempel wann sich der Wert das letzte Mal geändert hat.
sID	ID vom Datenpunkt welcher empfangen wurde.
lrValue	Enthält den aktuellen Wert des Datenpunktes.

5.5 ST_Zaptec_ECS_State	
Diese Struktur enthält gängige Ausgabewerte für die Zaptec Schnittstelle. Damit können Werte der Zaptec Ladestationen dargestellt werden. Dies ist eine spezifische Struktur genau für dieses System.	
Name	**Beschreibung**
bIsOnline	Zeigt an ob die Ladestation Online ist und mit der Cloud verbunden ist.
bIsEnabled	Zeigt an ob die Ladestation freigegeben und Aktiv ist.
rAllocatedCurrent	Zeigt den zugewiesenen Strom in A für eine Ladestation Installation an. Eine Installation kann aus mehrere Ladestationen bestehen.
rMaxCurrent	Zeigt den Maximalen Ladestrom in A an pro Ladestation.
rMinCurrent	Zeigt den Minimalen Ladestrom in A an pro Ladestation.
rCurrentL1	Aktueller Strom in A auf L1
rCurrentL2	Aktueller Strom in A auf L2
rCurrentL3	Aktueller Strom in A auf L3
rPower	Aktuelle Leistung der Ladestation in kW
rEnergySession	Abgegebenen Energie in kWh des aktuellen bzw. letzten Ladevorganges.
diOperationMode	Aktueller Modus der Ladestation
diWarnings	Warnungsstatus der Ladestation

sChargerID	Ladestations ID der Antwort

5.5 ST_Zaptec_Input_Charger

Diese Struktur enthält gängige Eingabewerte für die Zaptec Schnittstelle. Damit können Werte an eine Zaptec Ladestationen gesendet werden. Dies ist eine spezifische Struktur genau für dieses System.

Name	Beschreibung
rMinCurrentCharger	Minimaler Ladestrom in A der an der Ladestation zur Verfügung stehen darf.
rMaxCurrentCharger	Maximaler Ladestrom in A der an der Ladestation zur Verfügung stehen darf.
sChargerID	ID der Ladestation, an welche die Werte geschickt werden sollen. Findet man in der Cloudumgebung von Zaptec.

5.6 ST_Zaptec_Input_Installation

Diese Struktur enthält gängige Eingabewerte für die Zaptec Schnittstelle. Damit können Werte an eine Installation aus Zaptec Ladestationen gesendet werden. Dies ist eine spezifische Struktur genau für dieses System.

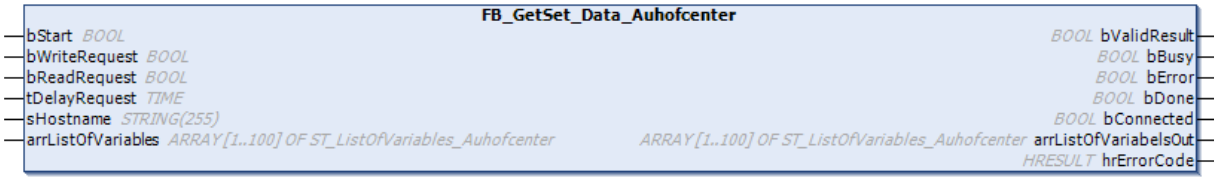
Name	Beschreibung
rAvailableCurrentInstallation	Strom der für die Installation der Ladestationen zur Verfügung steht. Die Ladestationen teilen sich den Strom dann selbstständig untereinander auf.
sInstallationID	ID der Installation der Ladestationen, an welche der Wert geschickt werden sollen. Findet man in der Cloudumgebung von Zaptec.

6. Funktionsblöcke

Anbei findet man eine Beschreibung zu allen Funktionsblöcken welche sich in der Lynus HTTP Bibliothek befinden. Hier findet man Funktionen welche über die HTTP Schnittstelle kommunizieren.

6.1 FB_GetSet_Data_Auhofcenter

Mit diesem Funktionsblock ist es möglich eine Kommunikation mit dem bestehenden GLT System im Auhofcenter in Wien herzustellen. Somit ist es möglich Werte von Datenpunkten aus diesem GLT auszulesen oder auch zu beschreiben. Die Datenpunkte sind vorgängig mit dem Techniker des GLT zu definieren. Es können bis zu 100 Datenpunkte gleichzeitig gelesen oder geschrieben werden.



Eingänge

Name	Typ	Beschreibung
bStart	BOOL	Mit diesem Eingang kann der Funktionsblock aktiviert oder deaktiviert werden. Beim setzen wird eine Verbindung mit dem Server aufgebaut.
bWriteRequest	BOOL	Ist dieser Eingang gesetzt wird an die Datenpunkten aus dem Eingangsarray ein Write Request geschickt. Das heisst diese Datenpunkte werden mit den Werten aus dem Eingangsarray beschrieben.
bReadRequest	BOOL	Ist dieser Eingang gesetzt wird an die Datenpunkten aus dem Eingangsarray ein Read Request geschickt. Das heisst diese Datenpunkte

		werden mit den Werten aus der GLT gelesen.
tDelayRequest	TIME	Zeit die vergehen soll zwischen den unterschiedlichen Requests an der Server.
sHostname	STRING(255)	URL vom Server der GLT
arrListOfVariables	ARRAY[1..100] OF ST_ListOfVariables_Auhofcenter	Array mit den Datenpunkten welche gelesen oder beschrieben werden sollen.

Ausgänge

Name	Typ	Beschreibung
bValidResult	BOOL	Wird True wenn eine gültige Antwort vom Server nach einem Request empfangen wurde.
bBusy	BOOL	Zeigt an dass der Baustein gerade Aktiv ist.
bError	BOOL	Zeigt an dass der Baustein einen Fehler hat.
bDone	BOOL	Zeigt an dass der Baustein den Request und die Auswertung der Daten fertig abgearbeitet hat.
bConnected	BOOL	Zeigt an dass der Baustein mit dem Server verbunden ist.
arrKistOfVariablesOut	ARRAY[1..100] OF ST_ListOfVariables_Auhofcenter	Gibt die aktuellen Werte der Datenpunkte zurück, die über das Eingangsarray gelesen oder

		geschrieben worden sind.
hrErrorCode	HRESULT	Error Code. Sie Beckhoff Dokumentation. Link

6.2 FB_Get_Data_SmartMeApi_ElectricMeter

Mit diesem Funktionsblock ist es möglich eine Kommunikation mit einem SmartMe Cloud Projekt herzustellen. Somit ist es möglich Werte von Elektrozählern aus einem SmartMe Projekt auszulesen. Wichtig ist, dass er Benutzer Leserechte hat auf diese Projekte. Je nach Lizenzmodell bei SmartME für das jeweilige Projekt können die Daten dann schneller oder langsamer ausgelesen werden. Es können maximal 50 Zähler aus einem Projekt ausgelesen werden. Wichtig ist, dass die Mailadresse bei Username komplett Klein geschrieben werden muss.

Eingänge

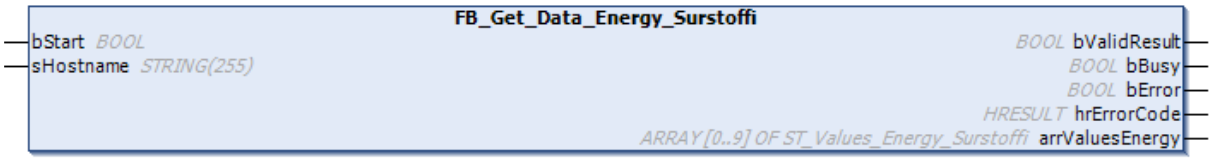
Name	Typ	Beschreibung
bStart	BOOL	Mit diesem Eingang kann der Funktionsblock aktiviert oder deaktiviert werden. Beim setzen wird eine Verbindung mit dem Server aufgebaut.
sUsername	STRING(50)	Benutzername des SmartME Projektes. Meistens ist das die Mailadresse. Der selbe der verwendet wird beim Login über die SmartME Webseite.
sPassword	STRING(50)	Passwort des SmartME Projektes. Das selbe das verwendet wird beim Login über die SmartME Webseite.
tDelayRequest	TIME	Zeit die vergehen soll zwischen den unterschiedlichen Requests an der Server.

Ausgänge

Name	Typ	Beschreibung
bValidResult	BOOL	Wird True wenn eine gültige Antwort vom Server nach einem Request empfangen wurde.
bBusy	BOOL	Zeigt an dass der Baustein gerade Aktiv ist.
bError	BOOL	Zeigt an dass der Baustein einen Fehler hat.
hrErrorCode	HRESULT	Error Code. Sie Beckhoff Dokumentation. Link
arrListOfDevices	ARRAY[1..50] OF ST_ListOFValues_SmartMe_ElectricMeter	Array mit der Ausgangsstruktur mit allen relevanten Daten der SmartMe Elektrozähler aus dem jeweiligen Projekt.

6.3 FB_Get_Data_Energy_Surstoffi

Mit diesem Funktionsblock ist es möglich eine Kommunikation mit dem Grafana Server in der Surstoffi in Zug aufzubauen. Auf diesem Server laufen in der Surstoffi viele elektrische Messdaten zusammen. Mit diesem Funktionsbaustein können die Netzdaten und PV Daten der Gebäude S16, S18, S20 und S22 ausgelesen werden. Die Daten dafür müssen zuerst durch eine Techniker auf der Grafana Seite freigegeben werden.



Eingänge

Name	Typ	Beschreibung
bStart	BOOL	Mit diesem Eingang kann der Funktionsblock aktiviert oder deaktiviert werden. Beim setzen wird eine Verbindung mit dem Server aufgebaut.
sHostname	STRING(255)	URL vom Grafana Server.

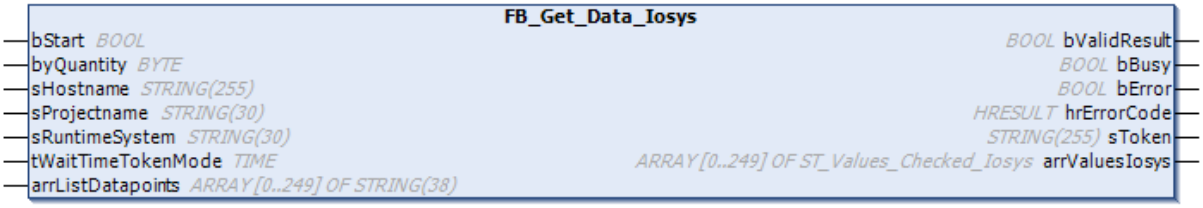
Ausgänge

Name	Typ	Beschreibung
bValidResult	BOOL	Wird True wenn eine gültige Antwort vom Server nach einem Request empfangen wurde.
bBusy	BOOL	Zeigt an dass der Baustein gerade Aktiv ist.
bError	BOOL	Zeigt an dass der Baustein einen Fehler hat.
hrErrorCode	HRESULT	Error Code. Sie Beckhoff Dokumentation. Link
arrValuesEnergy	ARRAY[0..9] OF ST_Values_Energy_Surstoffi	Array mit der Ausgangsstruktur mit allen relevanten Daten der Netz und PV Daten der

		Gebäude S16 bis S22.

6.4 FB_Get_Data_Iosys

Mit diesem Funktionsblock ist es möglich eine Kommunikation über die Iosys Schnittstelle von B&R herzustellen und auf dessen APROL Server zuzugreifen. Somit können alle Daten die auf diesem Server gespeichert sind abgefragt werden. Wichtig zu wissen ist, dass die Datenpunkte in dem Eingangsarray sich bei aktivem Baustein nicht ändern dürfen. Dies hat den Grund weil der Baustein im Hintergrund mit einem Token arbeitet, den er beim starten des Bausteines generiert. Sollten Änderungen vorgenommen werden müssen, muss der Baustein zuerst deaktiviert werden. Mit dieser Funktion können bis zu 250 Datenpunkte gleichzeitig abgefragt werden.



Eingänge

Name	Typ	Beschreibung
bStart	BOOL	Mit diesem Eingang kann der Funktionsblock aktiviert oder deaktiviert werden. Beim setzen wird eine Verbindung mit dem Server aufgebaut.
byQuantity	BYTE	Anzahl an Datenpunkte die vom Aprol Server abgefragt werden sollen. Muss mit den belegten Positionen im Eingangs Array übereinstimmen.
sHostname	STRING(255)	URL vom APROL Server.
sProjectname	STRING(30)	Projektnamen welches auf dem Aprol Server oben läuft. Ist auf B&R Seite deklariert.
sRuntimeSystem	STRING(30)	Runtime System auf welchem das Projekt läuft. Ist auf B&R Seite deklariert.
tWaitTimeTokenMode	TIME	Abfragezeit der Daten am Server nachdem der

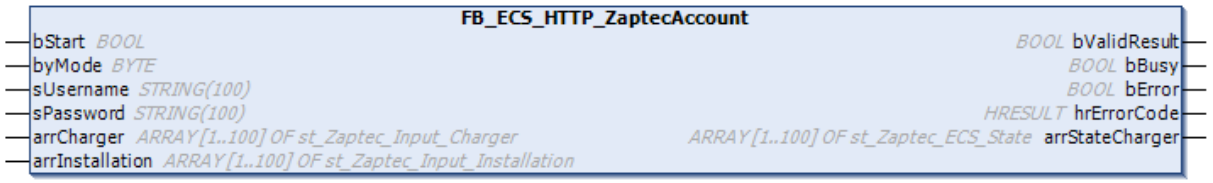
		Token generiert wurde für alle Daten aus dem Eingangsarray.
arrListDatapoints	ARRAY[0..249] OF STRING(38)	Eingangsarray in welchem die Namen der Datenpunkte eingetragen werden müssen. Diese Namen werden auf B&R Seite deklariert.

Ausgänge

Name	Typ	Beschreibung
bValidResult	BOOL	Wird True wenn eine gültige Antwort vom Server nach einem Request empfangen wurde.
bBusy	BOOL	Zeigt an dass der Baustein gerade Aktiv ist.
bError	BOOL	Zeigt an dass der Baustein einen Fehler hat.
hrErrorCode	HRESULT	Error Code. Sie Beckhoff Dokumentation. Link
sToken	STRING(255)	Token welcher bei der ersten Abfrage vom Server bezogen wurde. Mit diesem Token werden alle weiteren Anfragen am Server ausgeführt.
arrValueslosys	ARRAY[0..9] OF ST_Values_Checked_losys	Array mit der Ausgangsstruktur mit allen Werten der Datenpunkte aus dem Eingangsarray.

6.5 FB_ECS_HTTP_ZaptecAccount

Mit diesem Funktionsblock ist es möglich eine Kommunikation zu den Zaptec Ladestationen herzustellen. Dies geschieht direkt über das Zaptec Cloud Projekt. Somit müssen zuerst alle Ladestationen in der Zaptec Cloud angelegt und richtig konfiguriert werden, sollte dies nicht schon vorgängig erledigt worden sein. Ist dies erledigt, können die Ladestationen über diesen Funktionsbaustein in Ihrer Leistung geregelt werden. Dazu müssen die Ladestationen aber noch über Fremdzugriff gesteuert werden können. Diese Erlaubnis ist ebenfalls im Projekt in der Zaptec Cloud zu erledigen. Der Funktionsbaustein bietet 2 Möglichkeiten um die Ladestationen zu regeln. Mann kann jede Ladestation einzeln über den Maximal Strom regeln. Mit einzukalkulieren ist dort aber, dass jede Ladestation immer nur einzeln angesprochen werden kann. Zusätzlich dauert es pro Ladestation ca. 1 Sekunde bis der Wert übergeben ist und alle Vorgänge abgeschlossen sind vom Request an den Server. Erst dann können Daten an die nächste Ladestation geschickt werden. Somit ist diese Methode bei Zeitkritischen Regelungen und vielen Ladestationen nicht empfehlenswert. Alternativ zu dieser Methode kann auch eine Ansteuerung und Regelung der Ladestationen auf Installations Basis ausgeführt werden. Hierzu werden in der Cloud mehrere Ladestationen zu einer Installation zusammengeführt. Die Leistungsvorgabe, also ein Gesamtstrom der dann an diese Installation geschickt wird, teilen sich die Ladestationen untereinander dann selber auf. Somit können dann mit einem Request zum Server mehrere Ladestationen gleichzeitig geregelt werden. Mit diesem Funktionsbaustein können maximal 100 einzelne Ladestationen oder 100 Installationen aus Ladestationen betrieben werden. Um diesen Funktionsbaustein über das Lynus EMS anzusteuern müssen ein oder mehrere «Dummy» Funktionsblöcke für Ladestationen verwendet werden. Diese können dann die Leistungsvorgabe ihrer Ausgänge an das Eingangsarray diese Funktionsblocks übergeben.



Eingänge

Name	Typ	Beschreibung
bStart	BOOL	Mit diesem Eingang kann der Funktionsblock aktiviert oder deaktiviert werden. Beim setzen wird eine Verbindung mit dem Server aufgebaut.
byMode	BYTE	Arbeitsmodus des Funktionsbausteines. 0 = Sende Leistungsvorgabe an

		jede Ladestation einzeln. 1 = Sende eine Leistungsvorgabe an eine Installation von Ladestationen.
sUsername	STRING(50)	Benutzername des Zaptec Projektes. Meistens ist das die Mailadresse. Der selbe der verwendet wird beim Login über die Zaptec Webseite.
sPassword	STRING(50)	Passwort des Zaptec Projektes. Das selbe das verwendet wird beim Login über die Zaptec Webseite.
arrCharger	ARRAY[1..100] OF ST_Zaptec_Input_Charger	Eingangsarray für Zugriffs und Leistungsdaten für bis zu 100 Ladestationen.
arrInstallation	ARRAY[1..100] OF ST_Zaptec_Input_Installation	Eingangsarray für Zugriffs und Leistungsdaten für bis zu 100 Installationen welche aus mehreren Ladestationen bestehen können.

Ausgänge

Name	Typ	Beschreibung
bValidResult	BOOL	Wird True wenn eine gültige Antwort vom Server nach einem Request empfangen wurde.
bBusy	BOOL	Zeigt an dass der Baustein gerade Aktiv ist.
bError	BOOL	Zeigt an dass der Baustein einen Fehler hat.
hrErrorCode	HRESULT	Error Code. Sie Beckhoff Dokumentation. Link

arrStateCharger	ARRAY[1..100] OF ST_Zaptec_ECS_State	Array mit der Ausgangsstruktur mit Werten der einzelnen Ladestationen.