Lynus ModbusTCP Helper

Beschreibung der Kommunikation zwischen einer Beckhoff SPS (Edge Device) und anderen Steuerungen

Inhalt

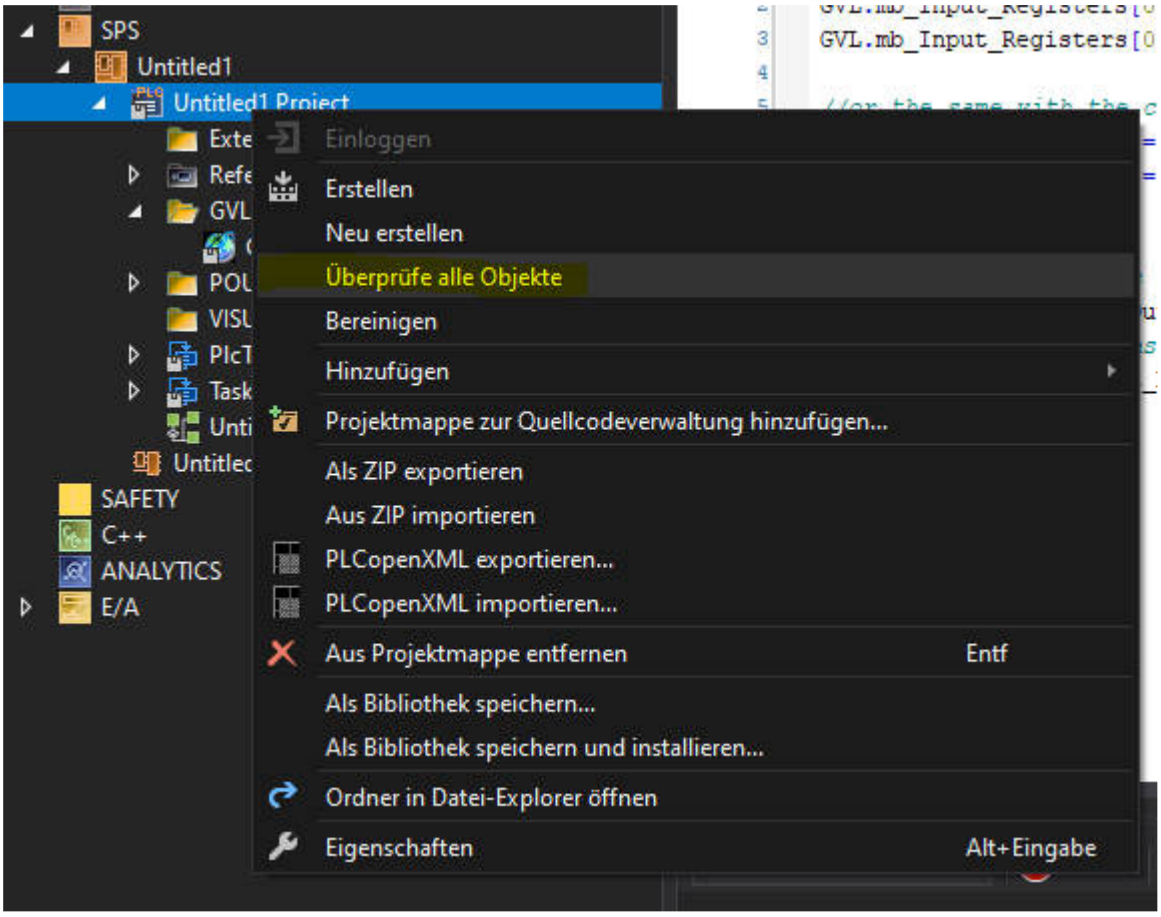
Version	Datum	Bearbeiter/Ersteller	Änderung/Ergänzung	Firma
0.9	23.06.2020	Kai Ebensperger	Dokument erstellt	Lynus AG

1. Systemvoraussetzungen

Die Lynus Bibliothek funktioniert auf allen Beckhoff Steuerungen welche TwinCat 3.1 4022.0 oder höher installiert haben. Zusätzlich zur Verwendung der Bibliothek muss eine globale Variablenliste erstellt werden mit den Einträgen für die Modbus TCP Register (siehe Kapitel 4. Verwendung). Zusätzlich muss auf der Beckhoff SPS die TwinCat 3 Modbus TCP Funktion TF6250 installiert sein. Minimale Programmierkenntnisse der IEC 61131 sollten vorhanden sein. Twincat 3 wird vollständig in Microsoft Visual Studio integriert. Es empfiehlt sich die „Community Edition“ zu verwenden, da diese Kostenlos ist und in verschiedenen Sprachen zur Verfügung steht. Sollte dies nicht gewünscht sein, kann natürlich auch das Visual Studio verwendet werden, welches bei der Installation von Twincat 3 direkt mitinstalliert wird. Diese Bibliothek ist als Zusatz zur „Lynus Communicator“ Bibliothek anzusehen und ist lediglich als Erweiterung für diese gedacht. Um mit der Cloud kommunizieren zu können siehe bitte separates Dokument „Lynus Beckhoff SPS Communicator“.

Als weiteres muss folgendes erfüllt sein =>

- Installation Twincat 3 auf einem Entwicklungsrechner (https://infosys.beckhoff.com/content/1031/tc3_installation/18014404672187787.html?id=669649889506832157)
- Erstellen eines Neuen TwinCat 3 Projektes (https://infosys.beckhoff.com/content/1031/tc3_c/81064793403366923-1.html?id=4615842966533058323)
- Erstellen eines Neuen TwinCat 3 SPS Projektes (https://infosys.beckhoff.com/content/1031/tc3_plc_intro/2526124939.html?id=8945077908446532140)
- Lokale Variablen anlegen für Cloud Kommunikation und Globale Variablen für Modbus TCP kommunikation(https://infosys.beckhoff.com/content/1031/tc3_plc_intro/2526546571.html?id=587565583649663674) (siehe auch Kapitel 4. Verwendung)
- Übergabe der Daten zwischen Globalen und Lokalen Variablen für den Modbus TCP Datenaustausch zwischen Beckhoff SPS und Fremdsteuerung (siehe auch Kapitel 4. Verwendung).
- Beckhoff SPS Zielsystem auswählen (https://infosys.beckhoff.com/content/1031/cx51x0_hw/2241767691.html?id=3375236967558668276)
- Beckhoff SPS Projekt überprüfen und dann auf SPS aktivieren (https://infosys.beckhoff.com/content/1031/tc3_c/81064793403374091-1.html?id=2479372716407997111)

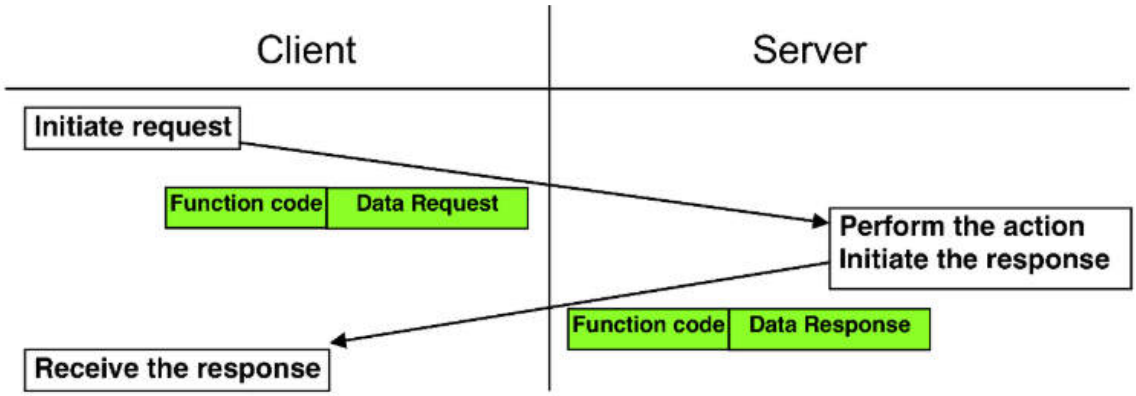


2. Einführung

Diese Bibliothek wurde dafür entwickelt um über eine Beckhoff SPS (dient als Edge Device) und einer Steuerung anderen Herstellers deren Daten in die Ventus Cloud zu bringen. Als Schnittstelle zwischen der Beckhoff SPS und der Steuerung des anderen Herstellers wurde Modbus TCP gewählt. Modbus TCP ist ein gängiges und standardisiertes Protokoll das fast jede Steuerung unterstützt. Da Modbus TCP vom Datentyp „WORD“ orientiert ist, wurden in dieser Bibliothek hauptsächlich Konvertierungsfunktionalitäten bereitgestellt um es dem Benutzer so einfach wie möglich zu machen, aus dem Datentyp „WORD“ jeden anderen gängigen Datentypen den die Beckhoff SPS unterstützt zu konvertieren. Somit sollte es einfach sein, Daten zwischen den beiden Steuerungen auszutauschen. Die Beckhoff SPS bildet in dieser Konstellation den Modbus TCP Server. Die Steuerung des anderen Herstellers bildet den Modbus TCP Client. Somit hat der Client die Möglichkeit Daten über Modbus TCP an die Beckhoff SPS zu schicken bzw. Daten von der Beckhoff SPS über Modbus TCP abzufragen.

Wie die Daten von der Beckhoff SPS in die Cloud kommen und wieder zurück entnehmen Sie bitte der Dokumentation „Lynus Beckhoff SPS Communicator“.

Die Bibliothek selber und alle dazugehörigen Informationen können nach dem Erstellen des Lynus Accounts heruntergeladen werden.



2.1 Kommunikationsdetails

Modbus TCP funktioniert über eine IP Verbindung. Jeder Client und jeder Server braucht im IP Netzwerk somit eine eindeutige IP Adresse. Der Standard Port bei Modbus TCP ist 502. Dieser Port muss je nach Firewall und deren Konfiguration als „Neue“ eingehende oder ausgehende Regel definiert werden und zugelassen werden. Ansonsten kann es sein, dass die Verbindung nicht funktioniert. Diese Einstellung muss über die Windows Netzwerkeinstellungen gemacht werden. Bei der Beckhoff SPS empfiehlt es sich eine statische IP Adresse zu verwenden, damit diese für den Client ständig über die selbe IP erreichbar bleibt.

2.2 Modbus Register

Die Register welche über die Beckhoff SPS angesprochen werden können, starten bei 32768 und enden bei 33268. Somit stehen insgesamt 501 Input und 501 Output Register und 501 Input und 501 Output Coils zur Verfügung. Register sind immer WORD orientiert. Coils sind einem digitalen Signal gleichzustellen, also können diese nur den Wert 0 oder 1 annehmen. In den Globalen Variablen der Beckhoff SPS gibt es 4 Arrays dazu. Jedes Array hat einen Range von 501 Einträgen. Somit entspricht z.B. Register 32768 Array Eintrag 0, Register 32769 Array Eintrag 1 usw.

2.3 Modbus Funktionen

Als Modbus Funktionen werden folgende unterstützt =>

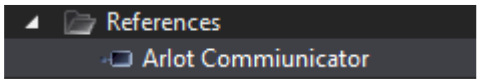
- FC1 (mb_Output_Coils in den GVL) (RW)
- FC2 (mb_Input_Coils in den GVL) (R)
- FC3 (mb_Output_Registers in den GVL) (RW)
- FC4 (mb_Input_Registers in den GVL) (R)
- FC5 (mb_Output_Coils in den GVL) (RW)
- FC6 (mb_Output_Registers in den GVL) (RW)
- FC15 (mb_Output_Coils in den GVL) (RW)
- FC16 (mb_Output_Registers in den GVL) (RW)

2.4 Empfehlung Beckhoff Hardware

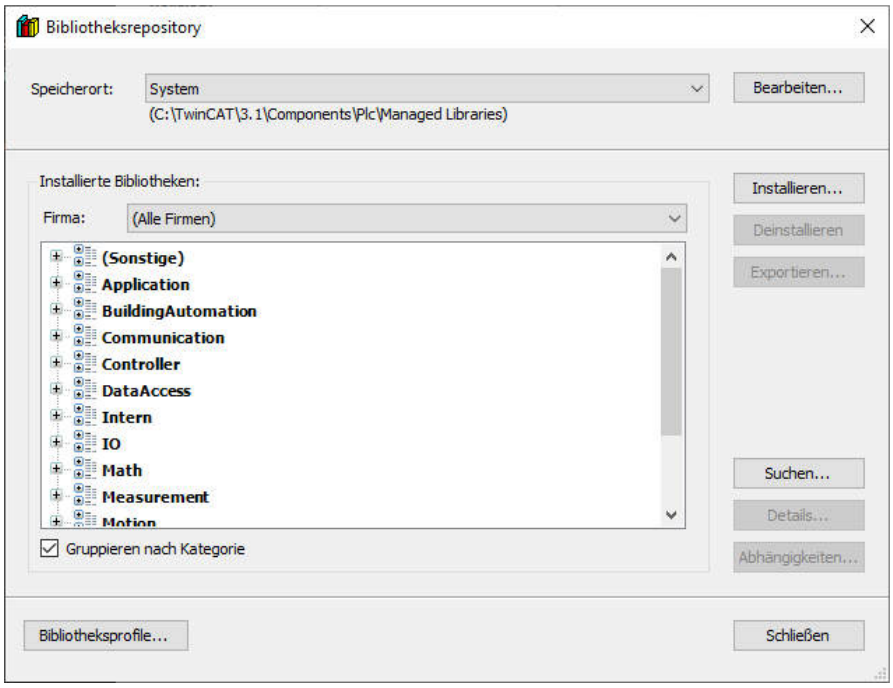
Da die Beckhoff SPS in dieser Konstellation keine aktiven Steuerungen oder hochkomplexen Code ausführen muss, sondern als reines Edge Device dient, kann auf eine hohe Performance je nach Projekt verzichtet werden. Bei Platzproblemen im Schaltschrank empfiehlt sich die Verwendung eines Ultra kleinen C6015-0010 IPC’s. Ansonsten kann auch auf die etwas günstigere Embedded Serie zurückgegriffen werden und ein CX8190 oder CX9020 verwendet werden.

3. 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 =>

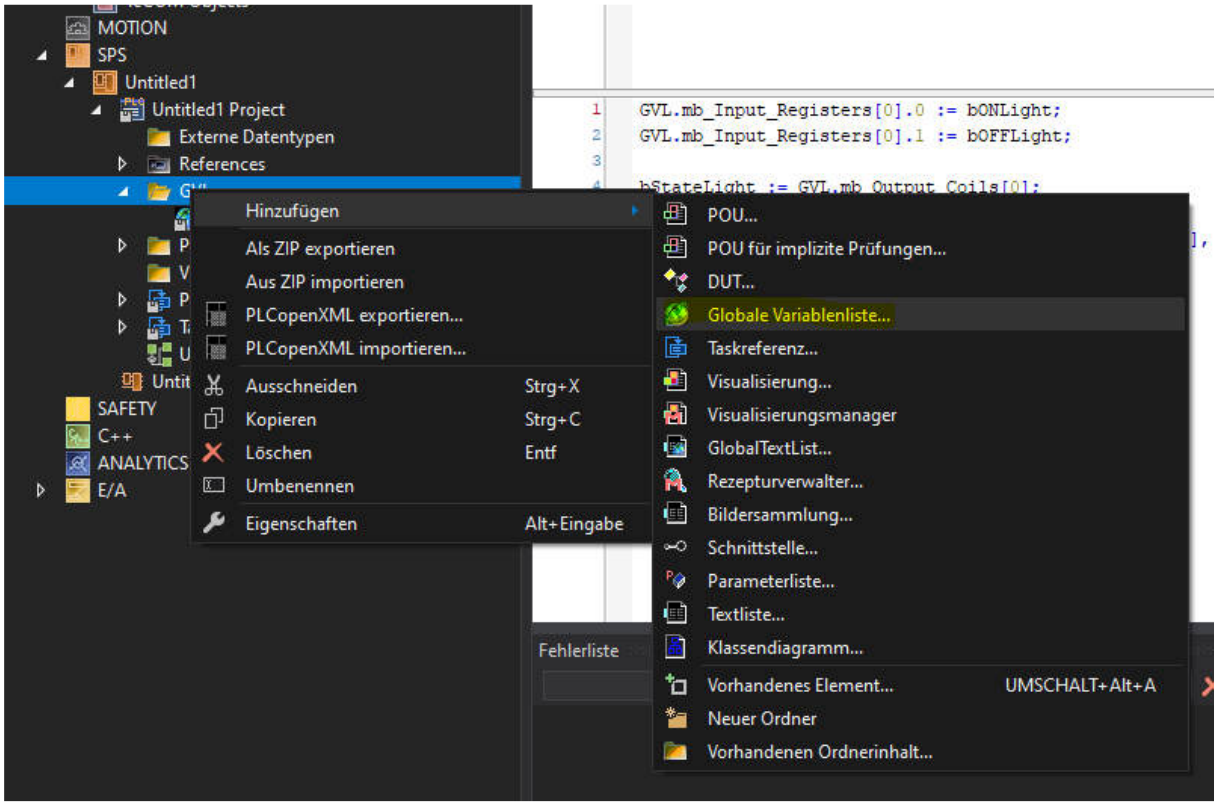


Die Lynus Bibliothek im abgespeicherten Pfad auswählen und dann auf Öffnen klicken. Nach erfolgreicher Installation erscheint die Bibliothek im Ordner „Kapa Informatik AG“ =>

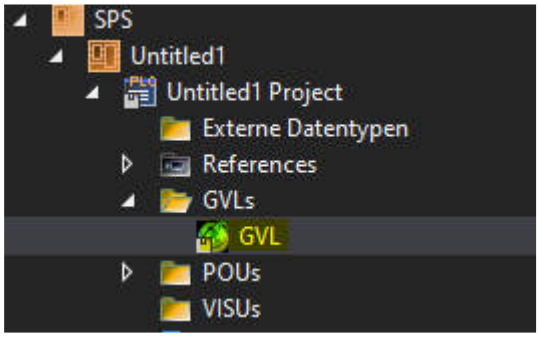
4. Verwendung

4.1 Globale Variablenliste

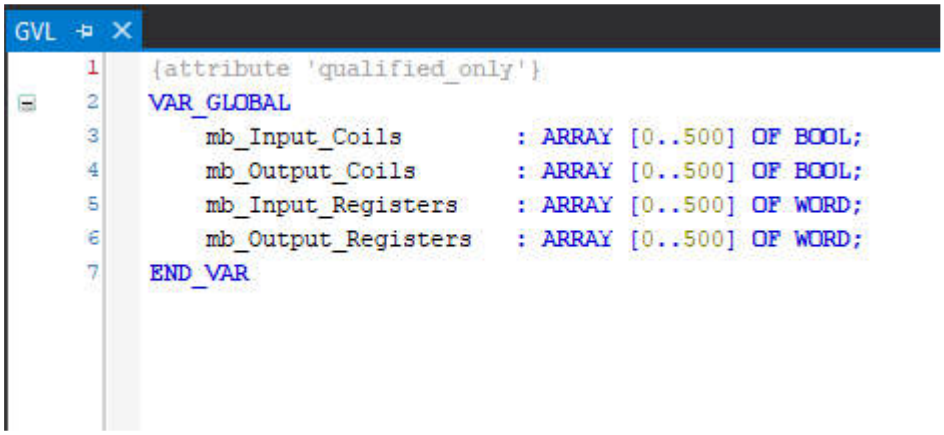
Um die Daten, die über Modbus TCP übertragen werden in der Beckhoff SPS verwenden zu können, muss zuerst eine Globale Variablenliste erstellt werden. Dazu im linken Reiter auf den leeren Ordner GVL's klicken und mit Klick auf die rechte Maustaste unter „Hinzufügen" eine Globale Variablenliste hinzufügen.



Die Globale Variablenliste muss als «GVL» gekennzeichnet und auch so erstellt werden.



Die Variablen müssen in der globalen Variablenliste zwingend wie folgt heissen =>



ACHTUNG : Beim Erstellen des Lynus Accounts kann eine Datei heruntergeladen werden, in welcher die Globalen Variablen abgespeichert sind und so direkt in die GVL Liste kopiert werden können.

4.2 XML

Defaultmässig können bei der Modbus TCP Übertragung nur 255 Datenpunkte je Modbus Funktion übertragen werden. Dies wurde angepasst auf 500 Datenpunkte. Um dies zu aktivieren muss das angepasste XML File durch das Standartfile auf der Beckhoff SPS ersetzt werden.

ACHTUNG : Beim Erstellen des Lynus Accounts kann eine Datei heruntergeladen werden in welcher die 2 XML Typen abgespeichert sind. Diese können auf der SPS wie folgt ersetzt werden:

Unter Windows welches höher/neuer ist als Windows CE, kann man das XML über ein Tool von Beckhoff ersetzten. Unter Windows CE kann das jeweilige File auf der SPS einfach durch das neue ersetzt werden. Danach muss die SPS jedoch neu gestartet werden. Wichtig ist dass das File nicht umbenannt wird, bzw. immer so heissen muss wie das Standard File auf der SPS.

https://infosys.beckhoff.de/content/1031/tf6250_tc3_modbus_tcp/18014398702226955.html?id=2747905489100401501

4.3 Lokale Variablen erstellen

Beim Erstellen eines neuen Twincat 3 SPS Projektes wird automatisch der Programmaufruf «MAIN» erstellt. Dieser Programmaufruf ist der Standarttask mit 10MS zugeteilt. In diesem Programmaufruf können Variablen, Funktionen usw. deklariert werden und dann im SPS Teil verwendet werden. Es können natürlich auch neue Programme erstellt werden und diese derselben Task, oder auch einer neuen Task zugewiesen werden. Diese lokale Variablen dienen später dazu, dessen Werte in die Cloud zu kriegen. Sie sind sozusagen das Ebenbild der Daten auf der Fremdsteuerung die via Modbus TCP zur Beckhoff SPS übertragen werden, bzw. von dort über die Fremdsteuerung auch wieder ausgelesen werden können.

MAIN GVL

```
1  PROGRAM MAIN
2  VAR
3      bONLight          : BOOL;          //{#ventus.ag#()}
4      bOFFLight         : BOOL;          //{#ventus.ag#()}
5      bStateLight       : BOOL;          //{#ventus.ag#()}
6      wValue            : WORD;          //{#ventus.ag#()}
7
```

4.4 Variablen markieren

Wie Sie die Variablen markieren welche in die Cloud geschickt werden sollen, entnehmen Sie bitte der Beschreibung „Lynus Beckhoff SPS Communicator“.

4.5 Array Zugriff auf die GVL

Da die globalen Variablen alle in einem Array abgespeichert werden, kann auf diese wie folgt in TwinCat 3 zugegriffen werden um diese an die lokalen Variablen zu übergeben oder umgekehrt. Bei den „WORD“ Registern kann auch bitweise zugegriffen werden. Dafür wird hinter dem ARRAY Index welcher in der [ArrayIndex] Klammer steht ein Punkt mit dem gewünschten nachfolgendem Bitzugriff gemacht. Bitzugriff bei einem „WORD“ von 0 bis 15.

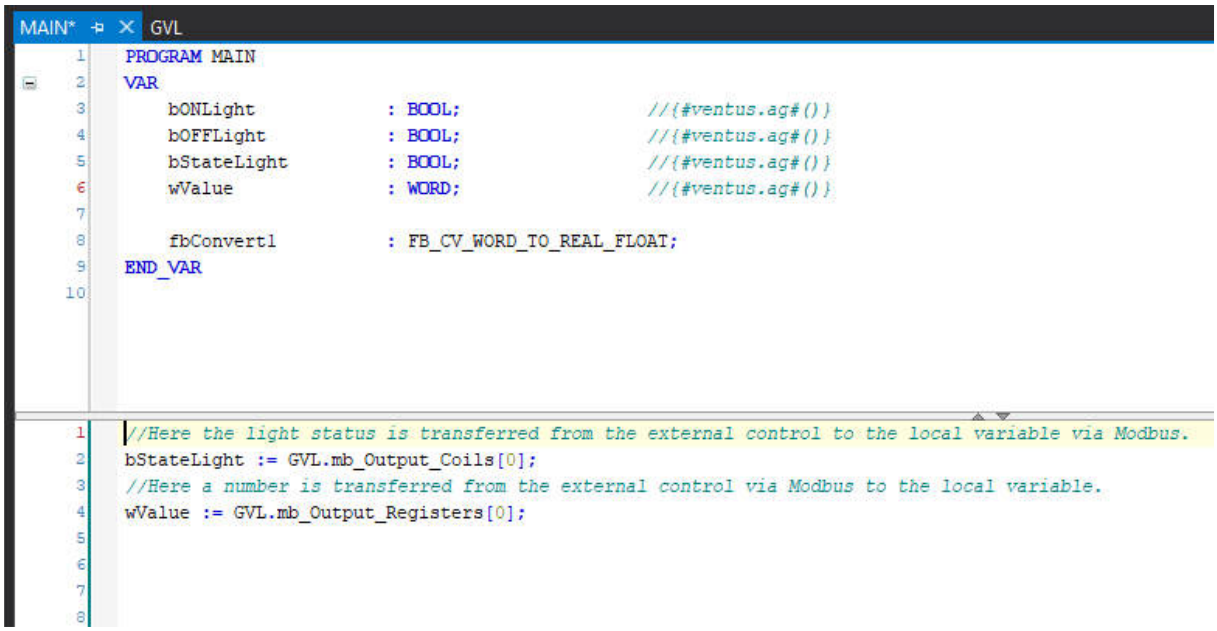
MAIN* GVL

```
1  PROGRAM MAIN
2  VAR
3      bONLight          : BOOL;          //{#ventus.ag#()}
4      bOFFLight         : BOOL;          //{#ventus.ag#()}
5      bStateLight       : BOOL;          //{#ventus.ag#()}
6      wValue            : WORD;          //{#ventus.ag#()}
7
8      fbConvert1        : FB_CV_WORD_TO_REAL_FLOAT;
9  END_VAR
10
```

```
1  //On and off command for the lamp to the external control
2  GVL.mb_Input_Registers[0].0 := bONLight;
3  GVL.mb_Input_Registers[0].1 := bOFFLight;
4
5  //or the same with the coil register
6  GVL.mb_Input_Coils[0] := bONLight;
7  GVL.mb_Input_Coils[1] := bOFFLight;
8
9
```

4.6 Datenaustausch zwischen lokalen und globalen Variablen Richtung Beckhoff

In welcher Art die Daten (zyklisch, eventbasiert) vom Client zur Beckhoff SPS übertragen werden, ist jedem Programmierer selber überlassen. Es empfiehlt sich jedoch eine eventbasierte Lösung um Ressourcen zu schonen. Nachdem die Daten in den Modbus Registern der Beckhoff SPS angekommen sind, müssen diese noch an die lokalen Variablen übergeben werden. Dies hat den Grund, damit nicht alle zum Teil noch unbenutzten Modbus Register in die Cloud geschickt werden und somit unnötiger Datentraffic verursacht wird. Die Übergabe von Globalen Variablen zu den lokalen Variablen findet im Code Teil statt. Bei lokalen Variablen vom Typ BOOL oder WORD können die Werte direkt übergeben werden. Für alle anderen Datentypen stehen Konvertierungsfunktionen zur Verfügung. (Siehe Kapitel „5. Funktionsblöcke“).



4.7 Datenaustausch zwischen lokalen und globalen Variablen Richtung Fremdsteuerung

In welcher Art die Daten (Zyklisch, eventbasiert) vom der Beckhoff SPS vom Client abgefragt werden, ist jedem Programmierer selber überlassen. Es empfiehlt sich jedoch eine zyklische Auslesung, da via Modbus auch mehrere Register/Coils gleichzeitig abgefragt werden können über den Client. Somit hat man innerhalb von wenigen Anfragen an den Server alle nötigen oder gewünschten Werte übertragen. Auch hier müssen die lokalen Variablen, welche Daten von der Cloud empfangen, zuerst an die Globalen Variablen übergeben werden um diese dann via Modbus TCP dem Client für die Abfrage zur Verfügung zu stellen.

MAIN* GVL

```
PROGRAM MAIN
VAR
    bONLight            : BOOL;                 //{#ventus.ag#()}
    bOFFLight           : BOOL;                 //{#ventus.ag#()}
    bStateLight         : BOOL;                 //{#ventus.ag#()}
    wValue              : WORD;                 //{#ventus.ag#()}

    fbConvert1          : FB_CV_WORD_TO_REAL_FLOAT;
END_VAR
```

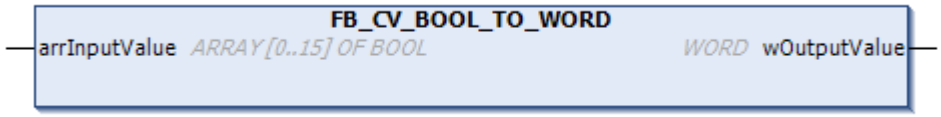
```
//On and off command for the lamp to the external control
GVL.mb_Input_Registers[0].0 := bONLight;
GVL.mb_Input_Registers[0].1 := bOFFLight;

//or the same with the coil register
GVL.mb_Input_Coils[0] := bONLight;
GVL.mb_Input_Coils[1] := bOFFLight;
```

5. Funktionsblöcke

Da die Übertragung über Modbus TCP «WORD» orientiert ist, wurden in der Bibliothek hauptsächlich Konvertierungsfunktionen implementiert um aus dem Datentyp «WORD» jeden anderen Datentypen konvertieren zu können. Die Konvertierung funktioniert immer in beide Richtungen. Dies kann z.B. der Fall sein, wenn der Bereich eines WORD's nicht mehr ausreicht um den gewünschten Wert zu übertragen.

5.1 BOOL_TO_WORD

Mit diesem Funktionsblock kann ein Eingangs Array mit 16 einzelnen Bits auf das Ausgabe WORD konvertiert werden. So entspricht z.B. Bit 0 = TRUE den Wert 1, Bit 1 = TRUE den Wert 2 oder Bit 0 und Bit 1 = TRUE den Wert 3…..



Eingänge

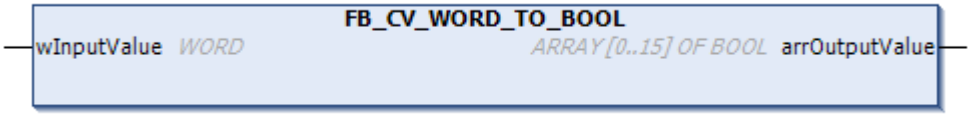
Name	Typ	Beschreibung
arrInputValue	ARRAY[0..15] OF BOOL	Eingangsarray mit den 16 einzelnen Bits für das Ausgabe WORD.

Ausgänge

Name	Typ	Beschreibung
wOutputValue	WORD	Konvertierter Ausgangswert

5.2 WORD_TO_BOOL

Mit diesem Funktionsblock kann ein Eingangswert vom Typ WORD in seine 16 einzelne Bits konvertiert werden. So entspricht z.B. der Wert 1 am Eingang Bit 0 am Ausgang oder der Wert 3 Bit 0 und Bit 1…..



Eingänge

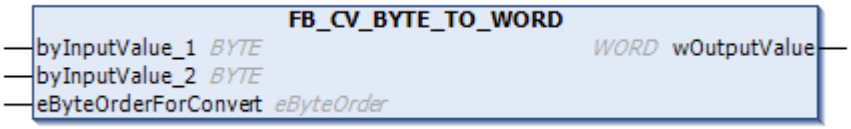
Name	Typ	Beschreibung
wInputValue	WORD	Eingangswert der in den Ausgangsdatentyp konvertiert werden soll.

Ausgänge

Name	Typ	Beschreibung
arrOutputValue	ARRAY[0..15] OF BOOL	Konvertierter Ausgangswert

5.3 BYTE_TO_WORD

Mit diesem Funktionsblock können 2 Eingangswerte vom Typ BYTE auf das Ausgabe WORD konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

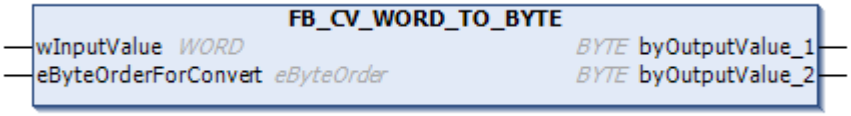
Name	Typ	Beschreibung
byInputValue_1	BYTE	Eingangswert 1 der in den Ausgangsdatentyp konvertiert werden soll.
byInputValue_2	BYTE	Eingangswert 2 der in den Ausgangsdatentyp konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
wOutputValue	WORD	Konvertierter Ausgangswert

5.4 WORD_TO_BYTE

Mit diesem Funktionsblock kann ein Eingangswert vom Typ WORD in 2 einzelne Bytes zerlegt werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

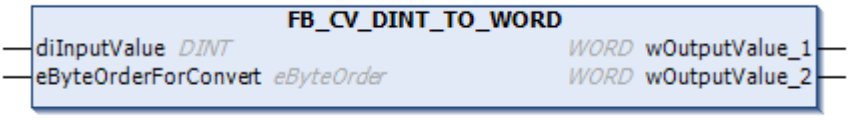
Name	Typ	Beschreibung
wInputValue	WORD	Eingangswert der in den Ausgangsdatentyp konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
byOutputValue_1	BYTE	1 Konvertierter Ausgangswert
byOutputValue_2	BYTE	2 Konvertierter Ausgangswert

5.5 DINT_TO_WORD

Mit diesem Funktionsblock kann der Vorzeichenbehaftete Datentyp DINT in 2 nicht Vorzeichenbehaftete Ausgangs WORD konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

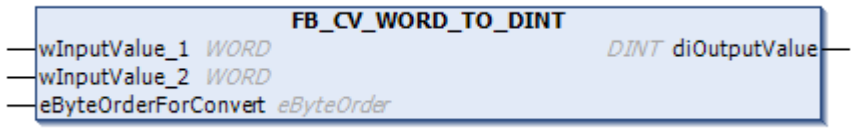
Name	Typ	Beschreibung
diInputValue	DINT	Eingangswert der in den Ausgangsdatentyp konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
wOutputValue_1	WORD	1 Konvertierter Ausgangswert
wOutputValue_2	WORD	2 Konvertierter Ausgangswert

5.6 WORD_TO_DINT

Mit diesem Funktionsblock können 2 nicht Vorzeichenbehaftete WORD in 1 Vorzeichenbehafteten Wert vom Datentypen DINT konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

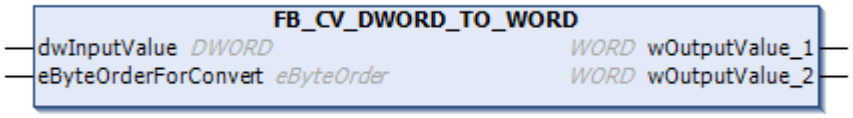
Name	Typ	Beschreibung
wInputValue_1	WORD	Eingangswert 1 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_2	WORD	Eingangswert 2 der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
diOutputValue	DINT	Konvertierter Ausgangswert

5.7 DWORD_TO_WORD

Mit diesem Funktionsblock kann ein Wert vom Datentyp WORD in 2 WORD Datentypen konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

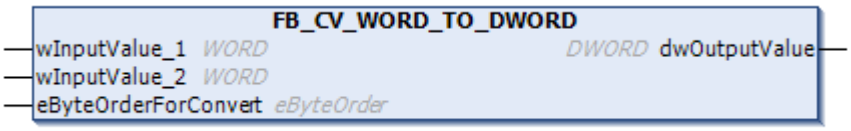
Name	Typ	Beschreibung
dwInputValue	DWORD	Eingangswert der in den Ausgangsdatentyp konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
wOutputValue_1	WORD	1 Konvertierter Ausgangswert
wOutputValue_2	WORD	2 Konvertierter Ausgangswert

5.8 WORD_TO_DWORD

Mit diesem Funktionsblock können 2 Werte vom Datentypen WORD in 1 DWORD Datentypen konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

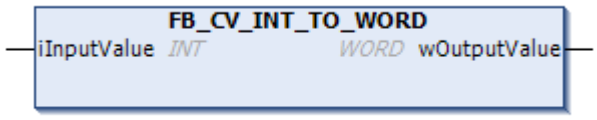
Name	Typ	Beschreibung
wInputValue_1	WORD	Eingangswert 1 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_2	WORD	Eingangswert 2 der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
dwOutputValue	DWORD	Konvertierter Ausgangswert

5.9 INT_TO_WORD

Mit diesem Funktionsblock kann ein Vorzeichenbehafteter Wert vom Datentypen INT in einen nicht Vorzeichenbehafteten Wert vom Datentyp WORD konvertiert werden.



Eingänge

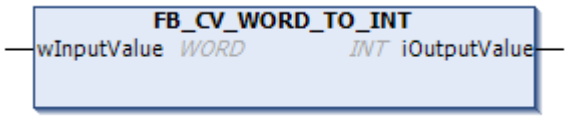
Name	Typ	Beschreibung
iInputValue	INT	Eingangswert der in den Ausgangsdatentypen konvertiert werden soll.

Ausgänge

Name	Typ	Beschreibung
wOutputValue	WORD	Konvertierter Ausgangswert

5.10 WORD_TO_INT

Mit diesem Funktionsblock kann ein nicht vorzeichenbehafteter Wert vom Datentypen WORD in einen Vorzeichenbehaftetet Wert vom Datentypen INT konvertiert werden.



Eingänge

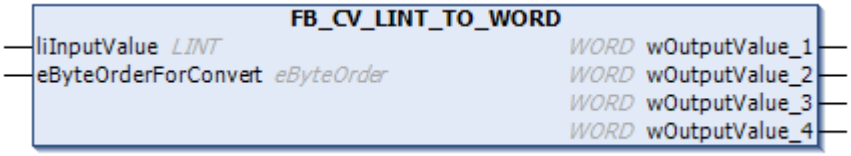
Name	Typ	Beschreibung
wInputValue	WORD	Eingangswert der in den Ausgangsdatentypen konvertiert werden soll.

Ausgänge

Name	Typ	Beschreibung
iOutputValue	INT	Konvertierter Ausgangswert

5.11 LINT_TO_WORD

Mit diesem Funktionsblock kann vorzeichenbehafteter Wert vom Datentyp LINT in 4 nicht vorzeichenbehaftete Werte vom Datentypen WORD konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

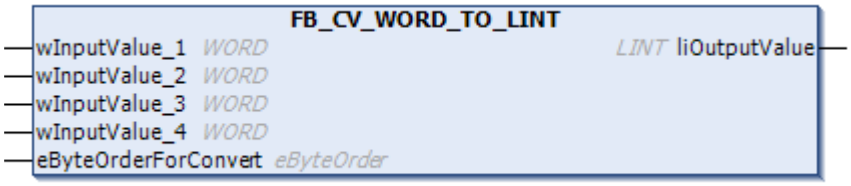
Name	Typ	Beschreibung
liInputValue	LINT	Eingangswert der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
wOutputValue_1	WORD	1 Konvertierter Ausgangswert
wOutputValue_2	WORD	2 Konvertierter Ausgangswert
wOutputValue_3	WORD	3 Konvertierter Ausgangswert
wOutputValue_4	WORD	4 Konvertierter Ausgangswert

5.12 WORD_TO_LINT

Mit diesem Funktionsbaustein können 4 nicht vorzeichenbehaftete Werte vom Datentyp WORD in 1 vorzeichenbehafteten Wert vom Datentypen LINT konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

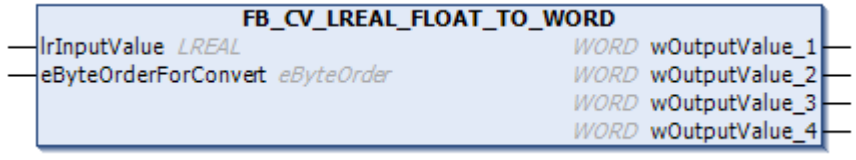
Name	Typ	Beschreibung
wInputValue_1	WORD	Eingangswert 1 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_2	WORD	Eingangswert 2 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_3	WORD	Eingangswert 3 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_4	WORD	Eingangswert 4 der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
liOutputValue	LINT	Konvertierter Ausgangswert

5.13 LREAL_FLOAT_TO_WORD

Mit diesem Funktionsblock ist es möglich einen LREAL-FLOAT Wert in 4 nicht vorzeichenbehaftete Werte vom Datentypen WORD zu konvertieren. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

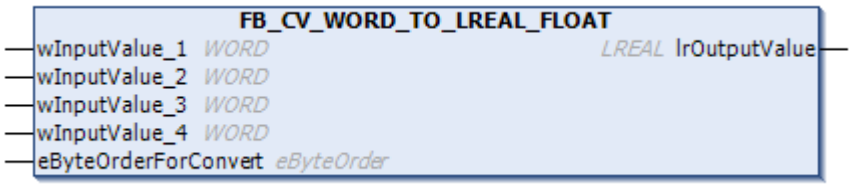
Name	Typ	Beschreibung
lrInputValue	LREAL	Eingangswert der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
wOutputValue_1	WORD	1 Konvertierter Ausgangswert
wOutputValue_2	WORD	2 Konvertierter Ausgangswert
wOutputValue_3	WORD	3 Konvertierter Ausgangswert
wOutputValue_4	WORD	4 Konvertierter Ausgangswert

5.14 WORD_TO_LREAL_FLOAT

Mit diesem Funktionsblock ist es möglich 4 nicht vorzeichenbehaftete Werte vom Datentyp WORD in 1 vorzeichenbehafteten Wert vom Datentyp LREAL-FLOAT zu konvertieren. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

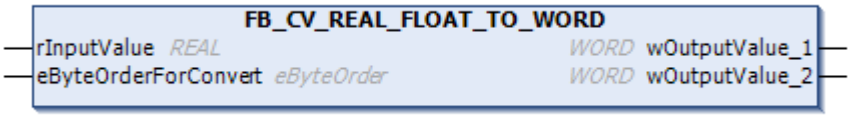
Name	Typ	Beschreibung
wInputValue_1	WORD	Eingangswert 1 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_2	WORD	Eingangswert 2 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_3	WORD	Eingangswert 3 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_4	WORD	Eingangswert 4 der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
lrOutputValue	LREAL	Konvertierter Ausgangswert

5.15 REAL_FLOAT_TO_WORD

Mit diesem Funktionsbaustein kann ein vorzeichenbehafteter Wert vom Datentyp REAL-FLOAT in 2 nicht vorzeichenbehaftete Werte vom Typ WORD konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

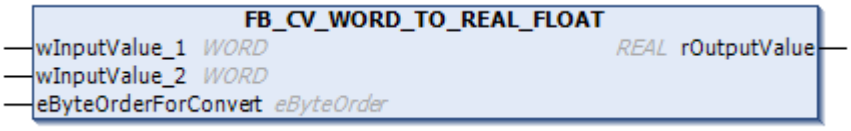
Name	Typ	Beschreibung
rInputValue	REAL	Eingangswert der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
wOutputValue_1	WORD	1 Konvertierter Ausgangswert
wOutputValue_2	WORD	2 Konvertierter Ausgangswert

5.16 WORD_TO_REAL_FLOAT

Mit diesem Funktionsbaustein können 2 nicht vorzeichenbehaftete Werte vom Datentyp WORD in 1 vorzeichenbehafteten Wert vom Typ REAL-FLOAT konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

Name	Typ	Beschreibung
wInputValue_1	WORD	Eingangswert 1 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_2	WORD	Eingangswert 2 der in den Ausgangsdatentypen konvertiert werden soll.

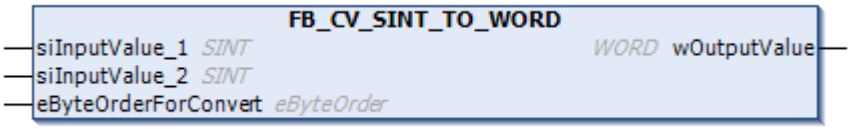
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
rOutputValue	REAL	Konvertierter Ausgangswert

5.17 SINT_TO_WORD

Mit diesem Funktionsbaustein ist es möglich 2 vorzeichenbehaftete Werte vom Datentyp SINT in 1 nicht vorzeichenbehafteten Wert vom Typ WORD zu konvertieren. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

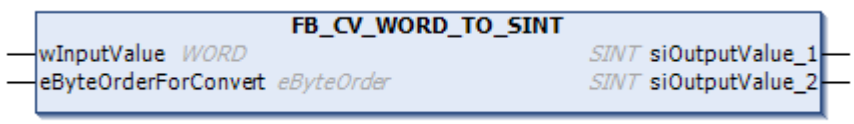
Name	Typ	Beschreibung
siInputValue_1	SINT	Eingangswert 1 der in den Ausgangsdatentypen konvertiert werden soll.
siInputValue_2	SINT	Eingangswert 2 der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
wOutputValue	WORD	Konvertierter Ausgangswert

5.18 WORD_TO_SINT

Mit diesem Funktionsbaustein ist es möglich 1 nicht vorzeichenbehafteten Wert vom Datentyp WORD in 2 vorzeichenbehaftete Werte vom Typ SINT zu konvertieren. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

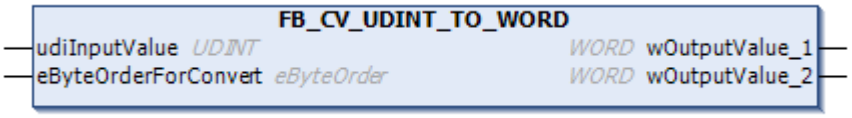
Name	Typ	Beschreibung
wInputValue	WORD	Eingangswert der in den Ausgangsdatentyp konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
siOutputValue_1	SINT	1 Konvertierter Ausgangswert
siOutputValue_2	SINT	2 Konvertierter Ausgangswert

5.19 UDINT_TO_WORD

Mit diesem Funktionsbaustein kann 1 Wert vom Datentyp UDINT in 2 Werte vom Datentyp WORD konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

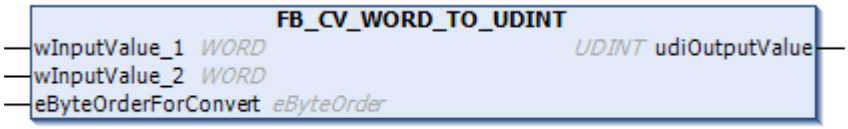
Name	Typ	Beschreibung
udiInputValue	UDINT	Eingangswert der in den Ausgangsdatentyp konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
wOutputValue_1	WORD	1 Konvertierter Ausgangswert
wOutputValue_2	WORD	2 Konvertierter Ausgangswert

5.20 WORD_TO_UDINT

Mit diesem Funktionsbaustein können 2 Werte vom Datentyp WORD in 1 Wert vom Datentyp UDINT konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

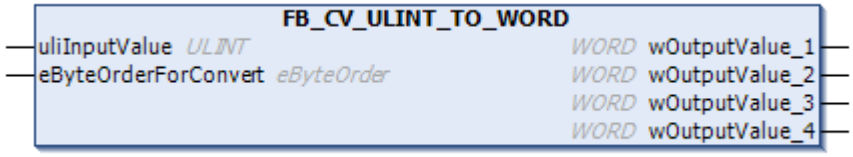
Name	Typ	Beschreibung
wInputValue_1	WORD	Eingangswert 1 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_2	WORD	Eingangswert 2 der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
udiOutputValue	UDINT	Konvertierter Ausgangswert

5.21 ULINT_TO_WORD

Mit diesem Funktionsbaustein kann 1 Werte vom Datentyp ULINT in 4 Werte vom Datentyp WORD konvertiert werden. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

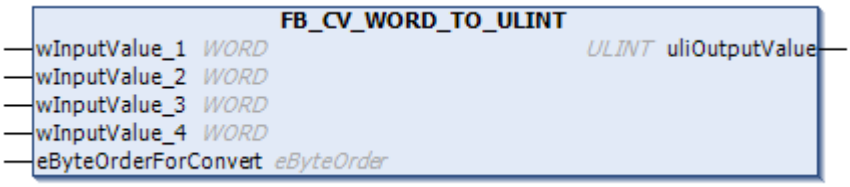
Name	Typ	Beschreibung
uliInputValue	ULINT	Eingangswert der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
wOutputValue_1	WORD	1 Konvertierter Ausgangswert
wOutputValue_2	WORD	2 Konvertierter Ausgangswert
wOutputValue_3	WORD	3 Konvertierter Ausgangswert
wOutputValue_4	WORD	4 Konvertierter Ausgangswert

5.22 WORD_TO_ULINT

Mit diesem Funktionsbaustein ist es möglich 4 Werte vom Datentypen WORD in 1 Wert vom Datentypen ULINT zu konvertieren. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

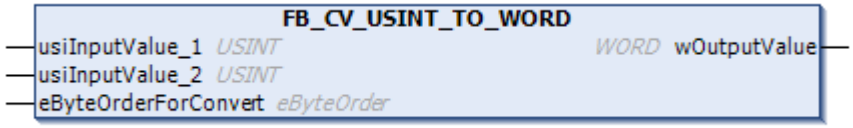
Name	Typ	Beschreibung
wInputValue_1	WORD	Eingangswert 1 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_2	WORD	Eingangswert 2 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_3	WORD	Eingangswert 3 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_4	WORD	Eingangswert 4 der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
uliOutputValue	ULINT	Konvertierter Ausgangswert

5.23 USINT_TO_WORD

Mit diesem Funktionsbaustein ist es möglich 2 Werte vom Datentypen USINT in 1 Wert vom Datentypen WORD zu konvertieren. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

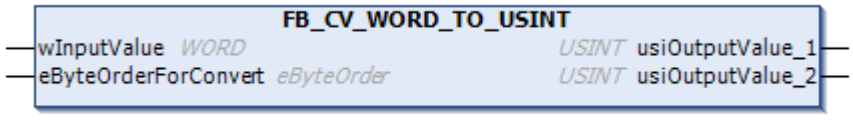
Name	Typ	Beschreibung
usiInputValue_1	USINT	Eingangswert 1 der in den Ausgangsdatentyp konvertiert werden soll.
usiInputValue_2	USINT	Eingangswert 2 der in den Ausgangsdatentyp konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
wOutputValue	WORD	Konvertierter Ausgangswert

5.24 WORD_TO_USINT

Mit diesem Funktionsbaustein ist es möglich 1 Wert vom Datentypen WORD in 2 Werte vom Datentypen SINT zu konvertieren. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

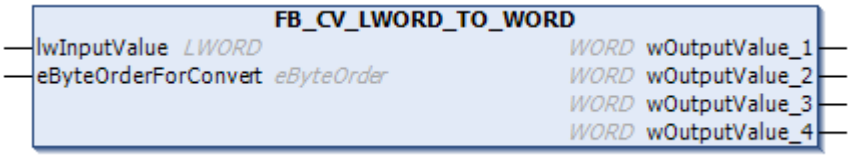
Name	Typ	Beschreibung
wInputValue	WORD	Eingangswert der in den Ausgangsdatentyp konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
usiOutputValue_1	USINT	1 Konvertierter Ausgangswert
usiOutputValue_2	USINT	2 Konvertierter Ausgangswert

5.25 LWORD_TO_WORD

Mit diesem Funktionsbaustein ist es möglich 1 Wert vom Datentyp LWORD in 4 Werte vom Datentyp WORD zu konvertieren. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

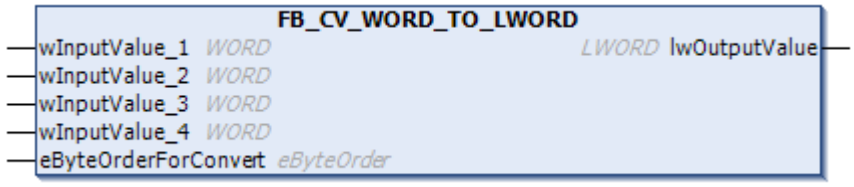
Name	Typ	Beschreibung
lwInputValue	LWORD	Eingangswert der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
wOutputValue_1	WORD	1 Konvertierter Ausgangswert
wOutputValue_2	WORD	2 Konvertierter Ausgangswert
wOutputValue_3	WORD	3 Konvertierter Ausgangswert
wOutputValue_4	WORD	4 Konvertierter Ausgangswert

5.26 WORD_TO_LWORD

Mit diesem Funktionsbaustein ist es möglich 4 Werte vom Datentypen WORD in 1 Wert vom Datentypen LWORD zu konvertieren. Zusätzlich kann ausgewählt werden in welcher Reihenfolge die Bytes übertragen werden sollen.



Eingänge

Name	Typ	Beschreibung
wInputValue_1	WORD	Eingangswert 1 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_2	WORD	Eingangswert 2 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_3	WORD	Eingangswert 3 der in den Ausgangsdatentypen konvertiert werden soll.
wInputValue_4	WORD	Eingangswert 4 der in den Ausgangsdatentypen konvertiert werden soll.
eByteOrderForConvert	ENUM	Auswahl in welcher Reihenfolge die Bytes übertragen werden sollen.

Ausgänge

Name	Typ	Beschreibung
lwOutputValue	LWORD	Konvertierter Ausgangswert

6. Verwendete Datentypen

Anbei findet man eine kurze Beschreibung von den verwendeten Datentypen der Bibliothek.

eByteOrder	
Name	Beschreibung
eBigEndian	Die Bytes werden im BigEndian Format übertragen
eLittleEndian	Die Bytes werden im LittleEndian Format übertragen.

7. Beispiele

Anbei finden sich Beispiele wie mit den Konvertierungen der Bibliothek gearbeitet werden kann.

7.1 Beispiel 1

In diesem Beispiel wird aufgezeigt wie ein empfangener Wert vom Type WORD der von der Fremdsteuerung an Register 32768 geschickt wurde, in 2 Bytes zerlegt wird und diese dann in die Cloud geschickt werden. Die Konvertierung passiert mit der dazu passenden Konvertierungsfunktion.

Deklarationsteil:

```
    fbConvert           : FB_CV_WORD_TO_BYTE;
    byByte1             : BYTE;                 //{#ventus.ag#()}
    byByte2             : BYTE;                 //{#ventus.ag#()}
END_VAR
```

Code Teil:

```
//Example to convert 1 WORD in 2 Bytes
fbConvert(wInputValue:= GVL.mb_Output_Registers[0], eByteOrderForConvert:= eByteOrder.eBigEndian,
byOutputValue_1=> byByte1, byOutputValue_2=> byByte2);
```

7.2 Beispiel 2

In diesem Beispiel wird aufgezeigt wie 2 empfangene Werte aus der Cloud vom Typ BYTE in einen Datentypen WORD konvertiert werden und auf Register 32768 gelegt werden um von der Fremdsteuerung via Modbus TCP ausgelesen zu werden.

Deklarationsteil:

```
    fbConvert           : FB_CV_BYTE_TO_WORD;
    byByte1             : BYTE;                 //{#ventus.ag#()}
    byByte2             : BYTE;                 //{#ventus.ag#()}
END_VAR
```

Code Teil:

```
//Example to convert 2 BYTES in 1 WORD
fbConvert(byInputValue_1:= byByte1, byInputValue_2:= byByte2,
eByteOrderForConvert:= eByteOrder.eLittleEndian, wOutputValue=> GVL.mb_Input_Registers[0]);
```

7.3 Beispiel 3

In diesem Beispiel wird aufgezeigt wie eine LREAL FLOAT Zahl in 4 Werte vom Datentypen WORD konvertiert wird um sie via Modbus TCP zum Fremdsystem zu übertragen. Der Wert wird ab Register 32768 abgespeichert.

Deklarationsteil:

```
7
8       fbConvert           : FB_CV_LREAL_FLOAT_TO_WORD;
9       lrValue             : LREAL;                    //{#ventus.ag#()}
10  END_VAR
11
```

Code Teil:

```
16  //Example to convert 1 LREAL to 4 WORD
17  fbConvert(
18      lrInputValue:= lrValue,
19      eByteOrderForConvert:= eByteOrder.eBigEndian,
20      wOutputValue_1=> GVL.mb_Input_Registers[0],
21      wOutputValue_2=> GVL.mb_Input_Registers[1],
22      wOutputValue_3=> GVL.mb_Input_Registers[2],
23      wOutputValue_4=> GVL.mb_Input_Registers[3]);
24
25
```