

Verbindung mit VAEM-V via Modbus/TCP in CODESYS

Kurzbeschreibung zur Erstellung einer Ethernet-Verbindung in CODESYS zwischen einer Festo Steuerung (CPX- oder CECC) und einem Ventil-Ansteuermodul VAEM-V-S8EPRS2

VAEM-V

TitelVerbindung mit VAEM-V via Ethernet in CODESYS
Version 1.10
Dokumentennummer 100334
Original de
AutorFesto
Letztes Speicherdatum 30.03.2021

Urheberrechtshinweis

Diese Unterlagen sind geistiges Eigentum der Festo SE & Co. KG, der auch das ausschließliche Urheberrecht daran zusteht. Eine inhaltliche Änderung, die Vervielfältigung oder der Nachdruck dieser Unterlagen sowie deren Weitergabe an Dritte ist nur mit der ausdrücklichen Erlaubnis der Festo SE & Co. KG gestattet.

Festo SE & Co. KG behält sich das Recht vor, dieses Dokument vollständig oder teilweise zu ändern. Alle Marken- und Produktnamen sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Titelhälter.

Rechtliche Hinweise

Hardware, Software, Betriebssysteme und Treiber dürfen nur für die beschriebenen Einsatzfälle und nur in Verbindung mit den von Festo SE & Co. KG empfohlenen Komponenten verwendet werden.

Festo SE & Co. KG lehnt jede Haftung für Schäden ab, die durch die Anwendung von allenfalls falschen bzw. unzureichenden Informationen oder aufgrund fehlender Informationen in diesen Unterlagen entstehen.

Defekte, die durch unsachgemäße Behandlung von Geräten und Baugruppen entstehen, sind von der Gewährleistung ausgeschlossen.

Sicherheitsrelevante Funktionen, im Sinne von Personen- und Maschinenschutz, dürfen mit Angaben und Informationen aus diesem Dokument nicht realisiert werden.

Für Folgeschäden, die durch einen Ausfall oder eine Funktionsstörung entstehen, wird dann jede Haftung abgelehnt. Im Übrigen gelten die Regelungen bzgl. Haftung aus den Liefer-, Zahlungs- und Softwarenutzungsbedingungen der Festo SE & Co. KG, welche Sie unter www.festo.com finden, welche wir Ihnen aber auch auf Anforderung gerne zukommen lassen.

Alle in diesem Dokument angegebenen Daten sind keine zugesicherten Eigenschaften, insbesondere nicht für Funktionalität, Zustand oder Qualität im rechtlichen Sinn.

Die Informationen dieses Dokuments gelten nur als einfache Hinweise für die Umsetzung einer ganz bestimmten, hypothetischen Anwendung, keinesfalls als Ersatz für die Bedienungsanleitung der jeweiligen Hersteller sowie der Konstruktion und Prüfung jeweils eigenen Anwendung durch den Benutzer.

Die jeweiligen Bedienungsanleitungen der Festo Produkte sind unter www.festo.com/sp zu finden.

Der Benutzer dieses Dokuments (Funktion und Anwendung) muss selbst sicherstellen, dass jede Funktion die hier beschrieben ist, auch in seiner Applikation ordnungsgemäß funktioniert. Der Benutzer bleibt auch durch das Studium dieses Dokuments sowie der Nutzung der darin genannten Angaben weiterhin allein verantwortlich für die eigene Anwendung.

Inhaltsverzeichnis

1	Verwendete Bauteile/Software	5
2	Verwendete Dokumentationen	6
3	Systembeispiele	7
3.1	CPX-CEC-M1 und VAEM-V	7
3.2	CECC-S und VAEM-V	7
4	Konfiguration der IP-Adresse von VAEM-V	8
5	Konfiguration einer Verbindung via Modbus/TCP in CODESYS	9
5.1	Ethernet-Verbindung hinzufügen	9
5.2	Mastergerät konfigurieren.....	10
5.3	Slave Gerät konfigurieren.....	11
6	Lesen und Schreiben eines Funktionsobjekts	14
6.1	Globale Kommunikationsdaten	14
6.2	Statuswort auslesen.....	15
6.3	Wert in das Kontrollwort schreiben, um zu starten.....	16
7	Anhang.....	18
7.1	Beispielprojekte	18
7.2	Verbindung mit 2 VAEM-V	18

1 Verwendete Bauteile/Software

Typ/Name	Version Software/Firmware	Teilenummer
VAEM-V-S8EPRS2	1.2.3.0	8088772
VAEM GUI	1.2.1.0	8087449
CECC-S	2.3.8.1.9245	574416
CPX-CEC-M1-V3	3.0.8.0.17190	3472765
CODESYS	V3.5SP7 Patch 2 pbF	

Tabelle 1.1: Verwendete Bauteile/Software

Link zum Festo Support Portal

<https://www.festo.com/SupportPortal/>

2 Verwendete Dokumentationen

Typ/Name	Version/Datum	Teilnummer
VAEM-V Handbuch	2020-	8127480
CECC-S Handbuch	2016-03a	8036061
CPX-CEC-M1-V3 Handbuch	2014-11c	8042601

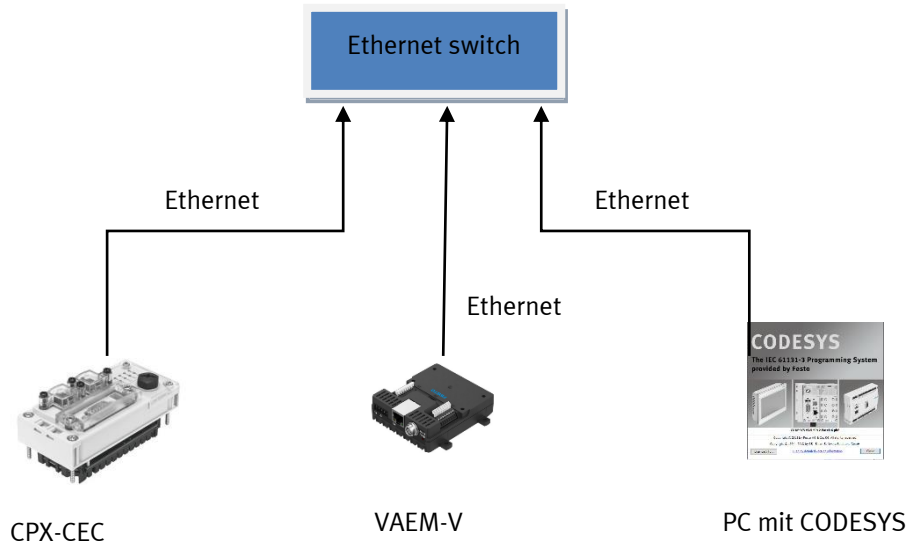
Tabelle 2.1: Verwendete Dokumentationen

Link zum Festo Support Portal

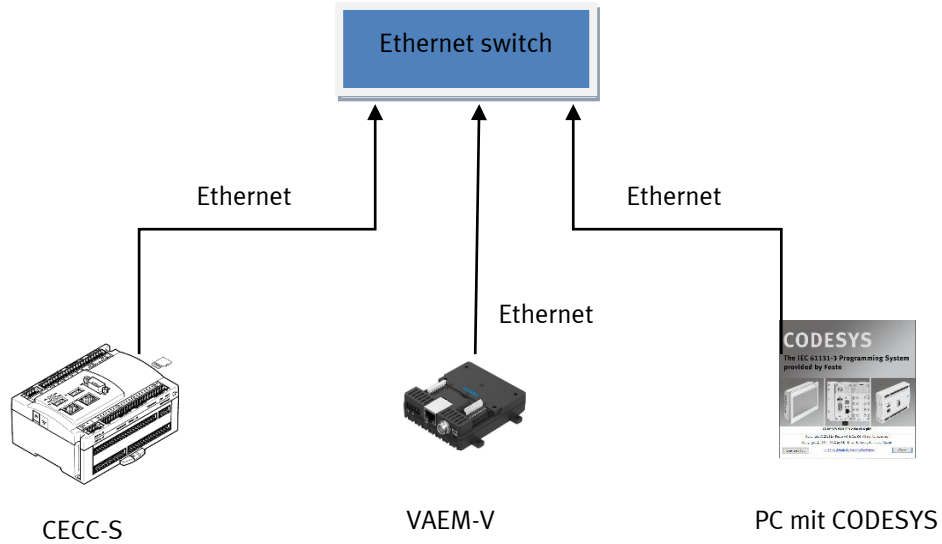
<https://www.festo.com/SupportPortal/>

3 Systembeispiele

3.1 CPX-CEC-M1 und VAEM-V



3.2 CECC-S und VAEM-V



4 Konfiguration der IP-Adresse von VAEM-V

Mit der VAEM GUI, die aus dem Support Portal heruntergeladen werden kann, kann das Ventil-Ansteuermodul VAEM-V in Betrieb genommen und die Ethernet-Verbindungseinstellung konfiguriert werden. (z.B. die Standard-IP-Adresse 192.168.178.1).

The screenshot displays the 'Festo VAEM GUI' window. The top navigation bar includes tabs for 'Verbindung', 'Steuerung', 'Analyse', 'Statuswort', 'Systeminfo', and 'Ethernet', with the 'Ethernet' tab currently selected. The Festo logo is positioned in the top right corner. The main content area is titled 'Ethernet Verbindungseigenschaften'. It shows the following configuration details:

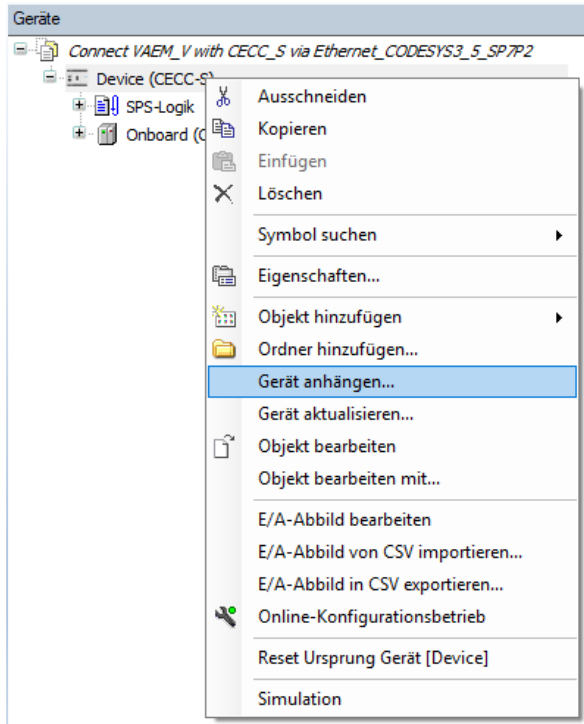
- MAC Adresse: 00:0E:F0:0D:00:37
- IP Adresse: 192.168.178.1
- Port: 502
- Timeout (ms): 1000

Below these details, there are input fields for 'Neue IP Adresse:', 'Port:', and 'Timeout (ms):', each containing the same values as above. A button labeled 'IP setzen' is located below the input fields. At the bottom of the window, a status bar indicates 'Status: verbunden' on the left, a blue 'IP' button in the center, and 'Statuswort: 0000001100010000' on the right.

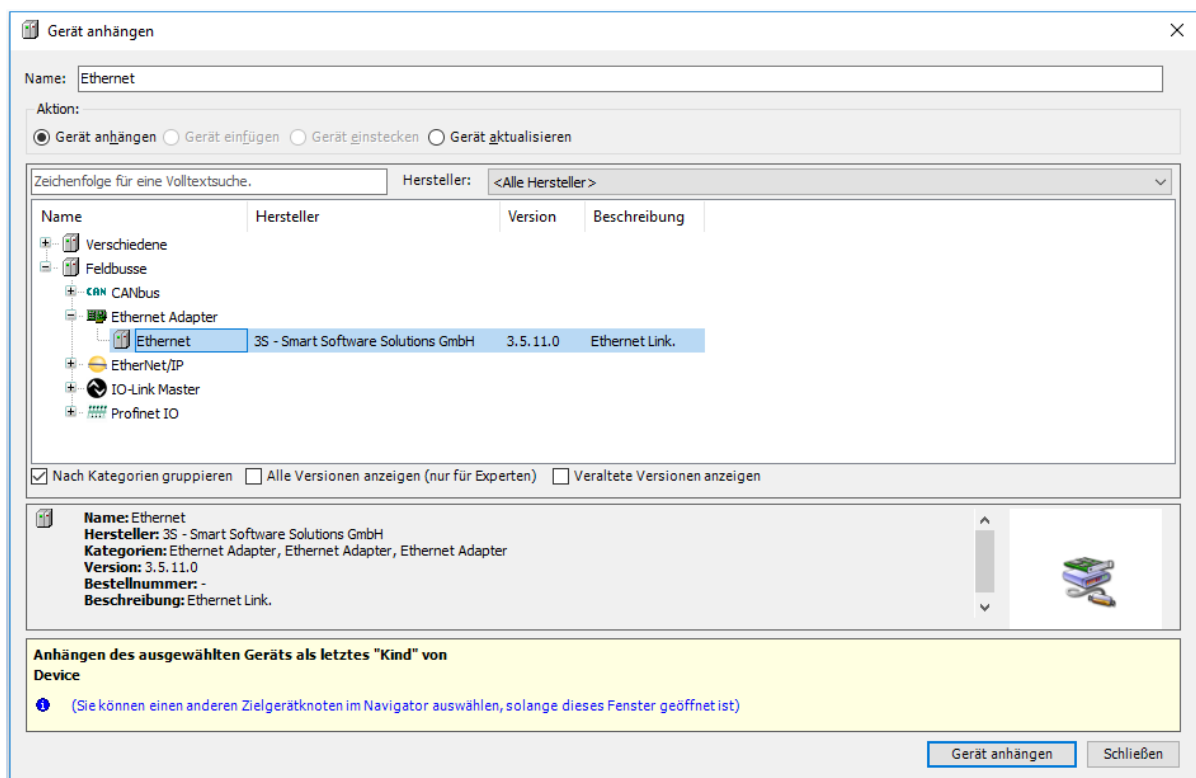
5 Konfiguration einer Verbindung via Modbus/TCP in CODESYS

5.1 Ethernet-Verbindung hinzufügen

1. Rechtsklick auf das Device, dann mit „Gerät anhängen“ die Schnittstellenauswahl öffnen.

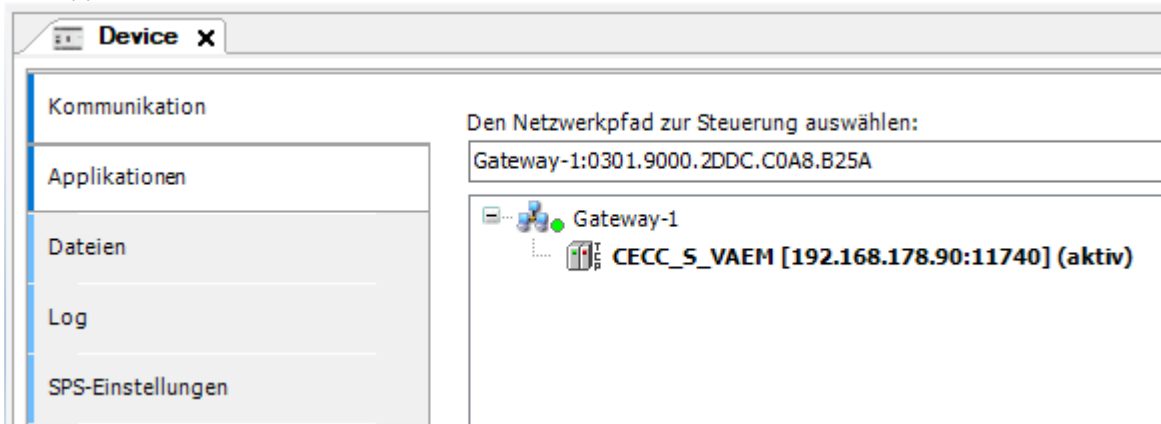


2. Ethernet-Schnittstelle auswählen, anhängen und die Auswahl schließen.

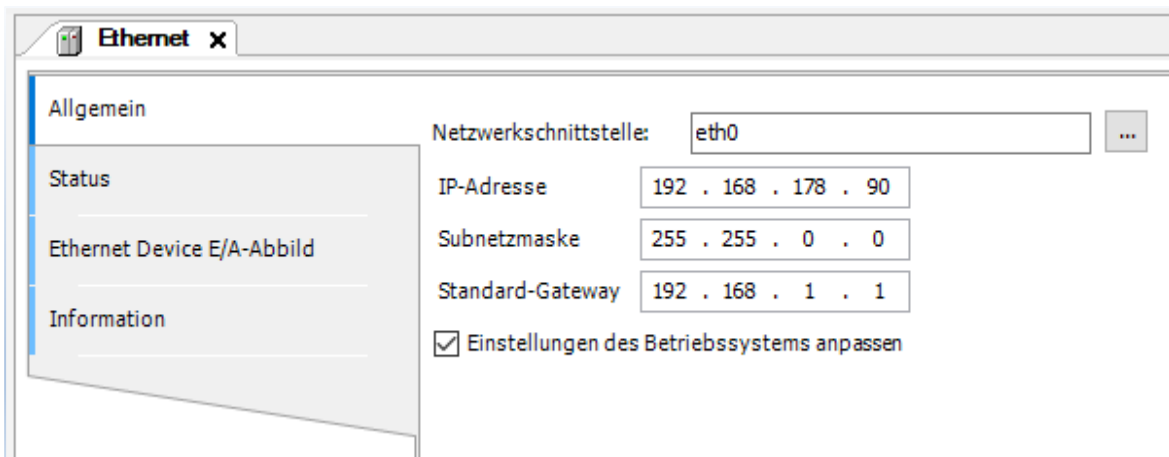


5.2 Mastergerät konfigurieren

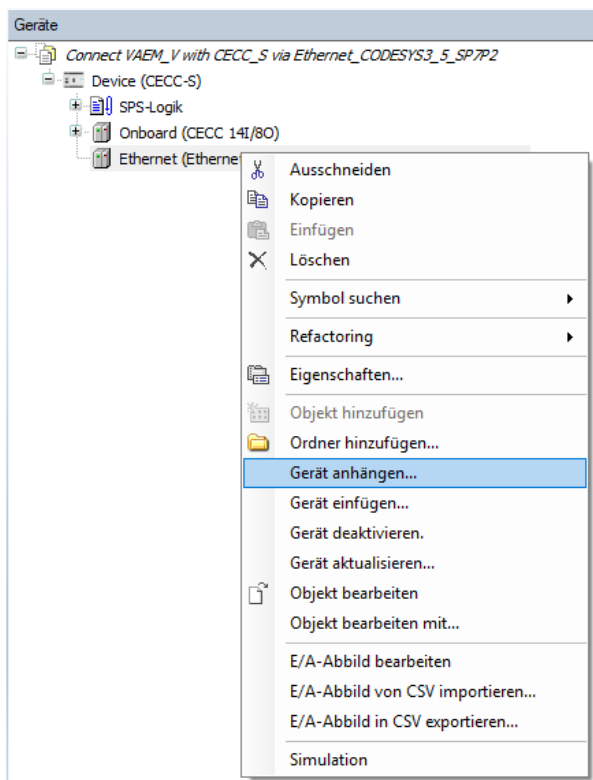
1. Doppelklick auf „Device“ um die Steuerung aktiv zu setzen.



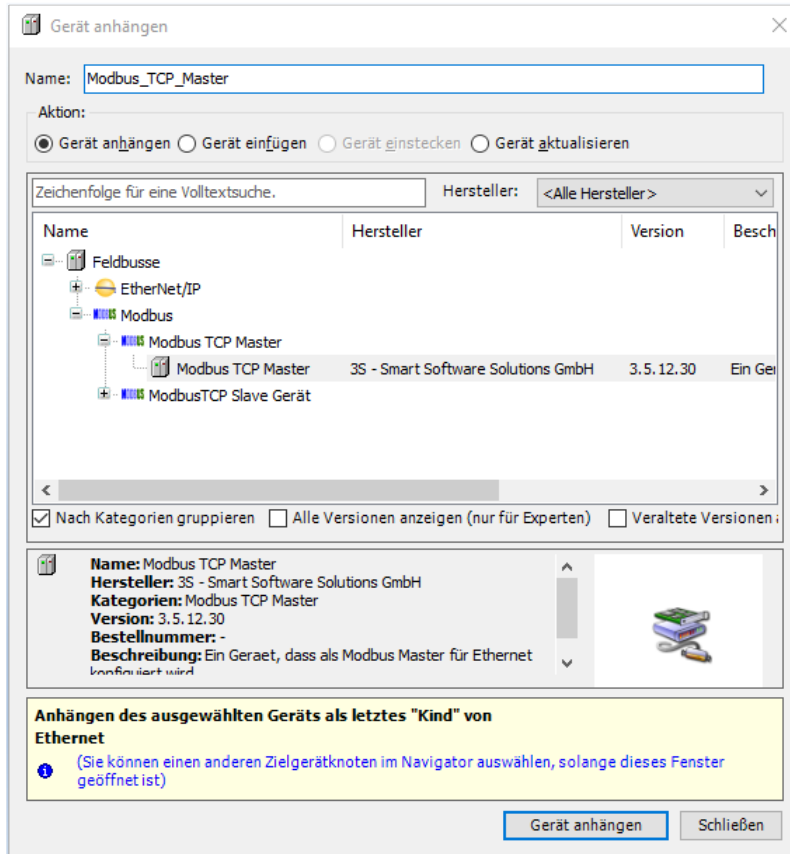
2. Doppelklick auf „Ethernet“ um die Einstellung zu öffnen und dann die Netzwerkschnittstelle selektieren.



3. Rechtsklick auf „Ethernet“, dann mit „Gerät anhängen“ die Auswahl öffnen.

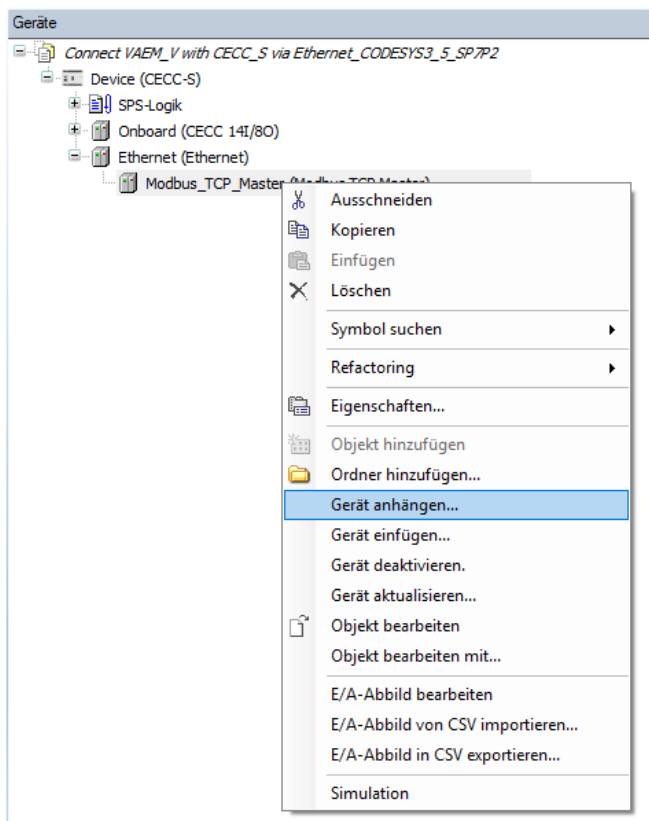


3. Modbus TCP Master auswählen, anhängen und die Auswahl schließen.

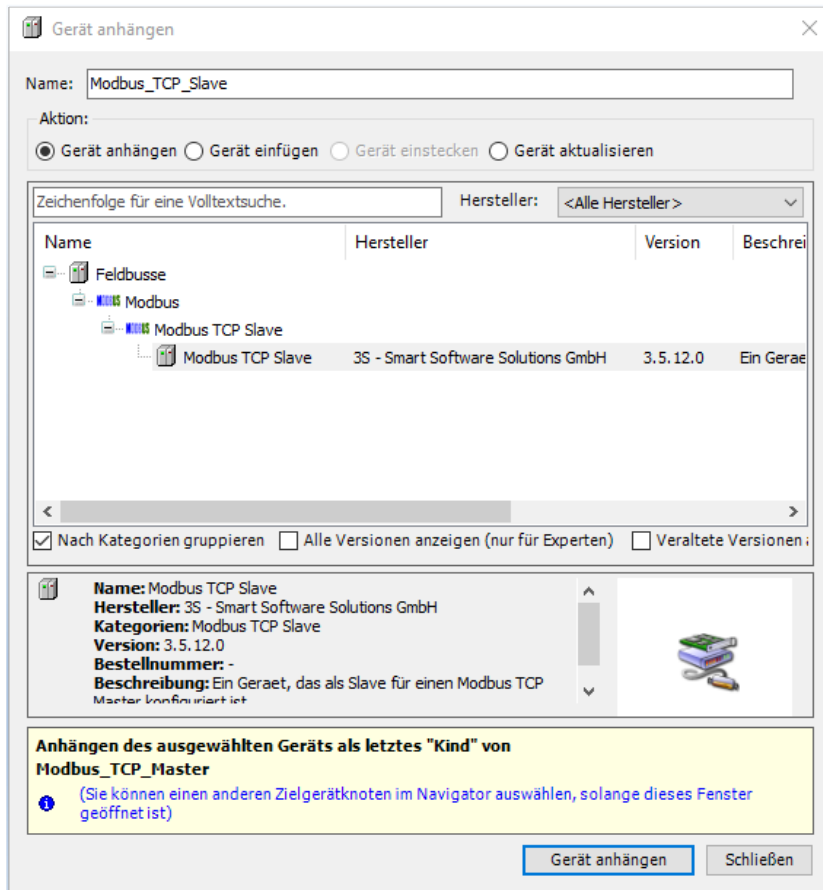


5.3 Slave Gerät konfigurieren

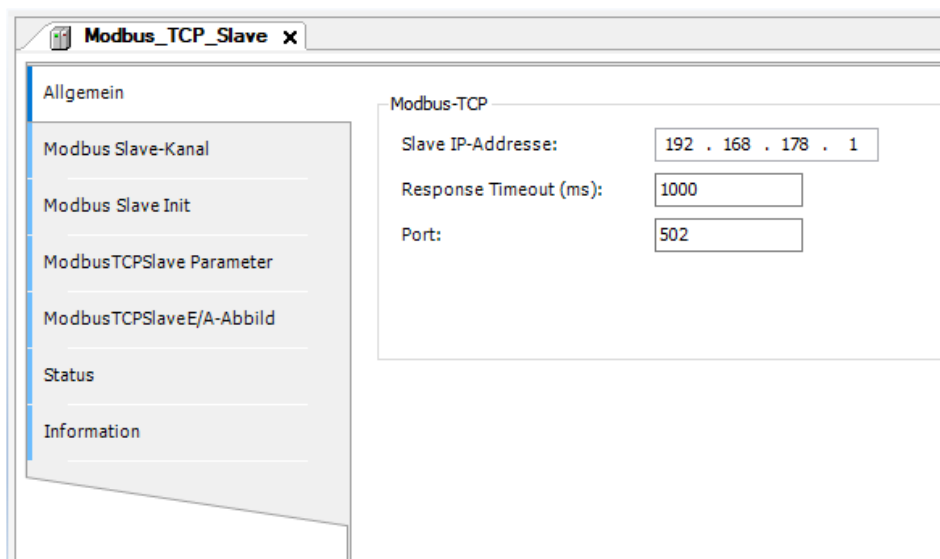
1. Rechtsklick auf „Modbus_TCP_Master“, dann mit „Gerät anhängen“ die Auswahl zu öffnen.



2. Modbus TCP Slave auswählen, anhängen und die Auswahl schließen.



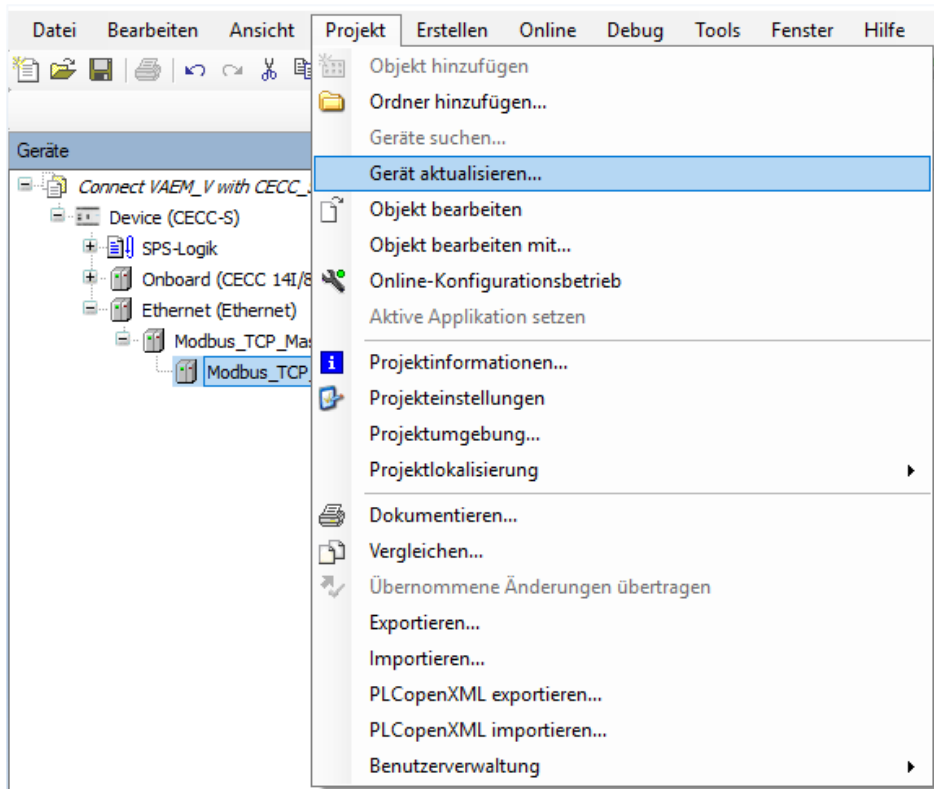
4. Doppelklick auf „Modbus_TCP_Slave“ um die Einstellung zu öffnen und die IP-Adresse des Ventil-Ansteuermoduls zu konfigurieren.



5. Klick auf „Modbus Slave Kanal“ und dann auf „Kanal hinzufügen“ um einen Modbus-Kanal zu erstellen und zu konfigurieren (z.B. hat die Datenaktualisierung eine Zykluszeit von 100ms).

The screenshot shows the 'ModbusChannel' configuration window. It has a 'Kanal' section with fields for 'Name' (Channel 0), 'Zugriffstyp' (Read/Write Multiple Registers (Funktionscode 23)), 'Trigger' (Zyklisch), 'Zykluszeit (ms)' (100), and 'Kommentar'. Below this are sections for 'READ Register' and 'WRITE Register', each with 'Offset' (0) and 'Länge' (7) fields, and a 'Fehlerbehandlung' dropdown set to 'Letzen Wert beibehalten'. At the bottom are 'OK' and 'Abbrechen' buttons.

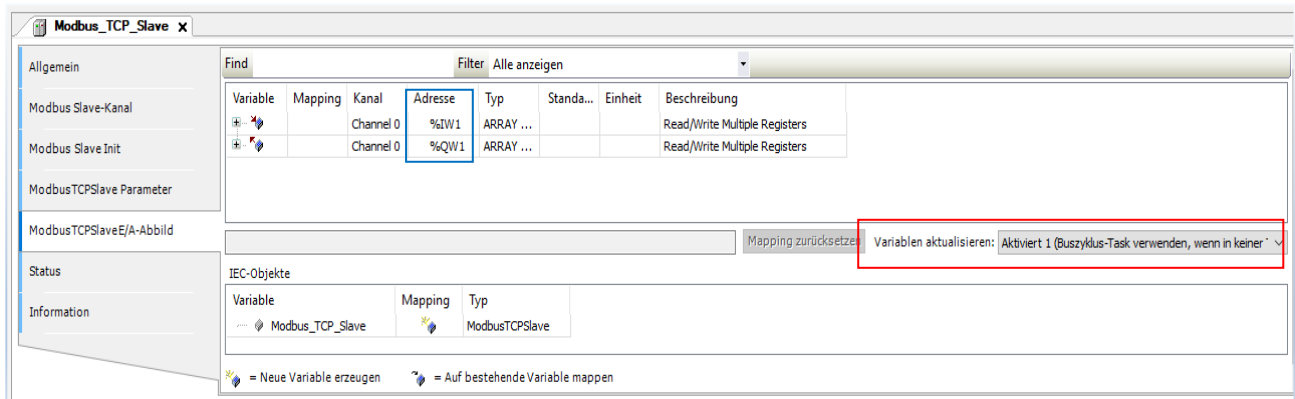
6. „Modbus_TCP_Slave“ im Fenster „Geräte“ selektieren und das Menü „Projekt“ öffnen. Dann Klick auf „Gerät aktualisieren“.



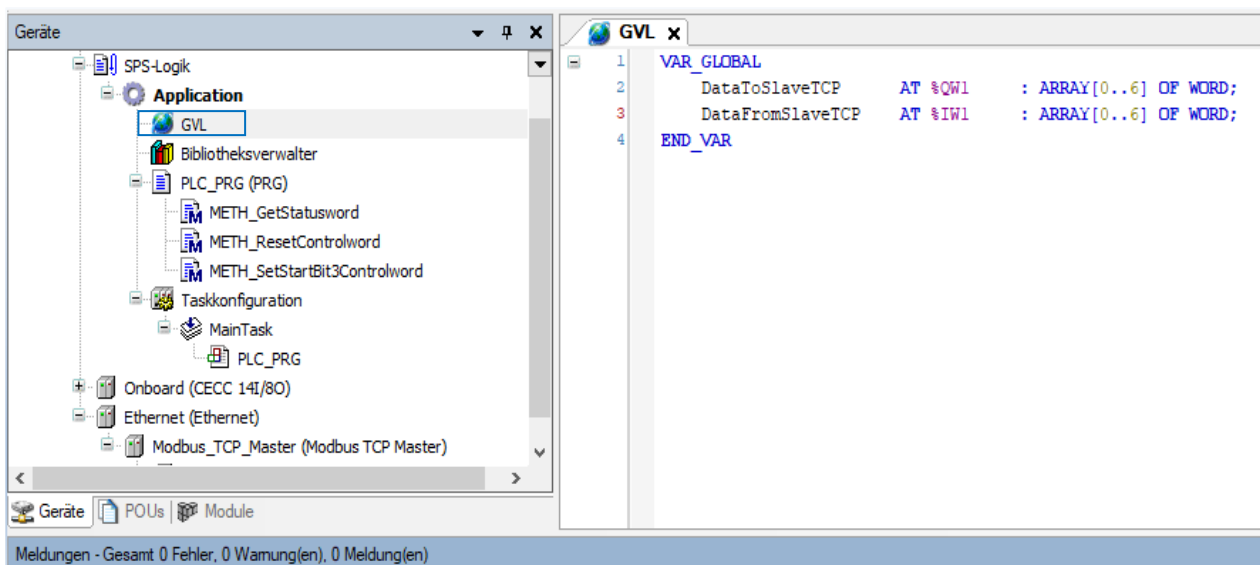
6 Lesen und Schreiben eines Funktionsobjekts

6.1 Globale Kommunikationsdaten

- Auf der Seite „Modbus_TCP_Slave“ befindet sich der Reiter „ModbusTCPSlaveE/A-Abbild“. Hier sind die Adressen für die Kommunikation mit dem Ventil-Ansteuermodul VAEM hinterlegt. Für die ausgehenden Daten zum Ventil-Ansteuermodul VAEM wäre das im Beispiel die Adresse %QW1 und für die vom Ventil-Ansteuermodul VAEM zurückgesendeten Daten die Adresse %IW1.



- „Variablen aktualisieren“ durch die Auswahl „Aktivieren 1(Buszyklus-Task verwenden , wenn in keiner Task)“.
- Eine globale Variablenliste erstellen.



6.2 Statuswort auslesen

```

1  PLC_PRG.METH_GetStatusword x
2  METHOD METH_GetStatusword : UINT
3  VAR_INPUT
4  END_VAR

1  (* Command To slave*)
2  //Byte 1 :Function-> 0:Read object,1:Write object
3  DataToSlaveTCP[0] := UINT_TO_WORD(SHL(16#00,8));
4  //Byte 2: Type of object -> 1: UINT08, 2: UINT16, 3: UINT32, 4: UINT64
5  DataToSlaveTCP[0] := DataToSlaveTCP[0] + 16#02;
6  //Byte 3: Index of object (Most significant byte)
7  //Byte 4: Index of object (Least significant byte)
8  DataToSlaveTCP[1] := 16#0002;
9  //Byte 5: Subindex of Object
10 DataToSlaveTCP[2] := UINT_TO_WORD(SHL(16#00,8));
11 //Byte 6: Reserved for response error
12 DataToSlaveTCP[2] := DataToSlaveTCP[2] + 16#00;
13 //Byte 7: Byte value of object (Most significant byte at using of a value with data type UINT64)
14 //Byte 8,9,10: Byte value of object (at using of a value with data type UINT64)
15 //Byte 11,12: Byte value of object (at using of a value with data type UINT64, UINT32)
16 //Byte 13: Byte value of object (at using of a value with data type UINT64, UINT32 and UINT16)
17 //Byte 14: Byte value of object (Least significant byte at using of a value with data type UINT64, UINT32 and UINT16, UINT08)
18 DataToSlaveTCP[3] := 16#0000;
19 DataToSlaveTCP[4] := 16#0000;
20 DataToSlaveTCP[5] := 16#0000;
21 DataToSlaveTCP[6] := 16#0000;
22
23 (* Answer from slave*)
24 //BYTE 1 :FUNCTION-> 0:Read object,1:Write object
25 //DataFromSlaveTCP[0];
26 //Byte 2: Type of object -> 1: UINT08, 2: UINT16, 3: UINT32, 4: UINT64
27 //DataFromSlaveTCP[0];
28 //Byte 3: Index of object (Most significant byte)
29 //Byte 4: Index of object (Least significant byte)
30 //DataFromSlaveTCP[1];
31 //Byte 5: Subindex of Object
32 //DataFromSlaveTCP[2];
33 //Byte 6: Response error
34 //DataFromSlaveTCP[2];
35 //Byte 7: Byte value of object (Most significant byte at using of a value with data type UINT64)
36 //Byte 8,9,10: Byte value of object (at using of a value with data type UINT64)
37 //Byte 11,12: Byte value of object (at using of a value with data type UINT64, UINT32)
38 //Byte 13: Byte value of object (at using of a value with data type UINT64, UINT32 and UINT16)
39 //Byte 14: Byte value of object (Least significant byte at using of a value with data type UINT64, UINT32 and UINT16, UINT08)
40 METH_GetStatusword := WORD_TO_UINT( DataFromSlaveTCP[6]);

```

6.3 Wert in das Kontrollwort schreiben, um zu starten

- Startbit 3 von Kontrollwort auf den Wert 1 setzen

```

1  (* ***** Set bit 1 of controlword ***** *)
2  (* Command To slave*)
3  //Byte 1 :Function-> 0:Read object,1:Write object
4  DataToSlaveTCP[0] := UINT_TO_WORD(SHL(16#01,8));
5  //Byte 2: Type of object -> 1: UINT08, 2: UINT16, 3: UINT32, 4: UINT64
6  DataToSlaveTCP[0] := DataToSlaveTCP[0] + 16#02;
7  //Byte 3: Index of object (Most significant byte)
8  //Byte 4: Index of object (Least significant byte)
9  DataToSlaveTCP[1] := 16#0001;
10 //Byte 5: Subindex of Object
11 DataToSlaveTCP[2] := UINT_TO_WORD(SHL(16#00,8));
12 //Byte 6: Reserved for response error
13 DataToSlaveTCP[2] := DataToSlaveTCP[2] + 16#00;
14 //Byte 7: Byte value of object (Most significant byte at using of a value with data type UINT64)
15 //Byte 8,9,10: Byte value of object (at using of a value with data type UINT64)
16 //Byte 11,12: Byte value of object (at using of a value with data type UINT64, UINT32)
17 //Byte 13: Byte value of object (at using of a value with data type UINT64, UINT32 and UINT16)
18 //Byte 14: Byte value of object (Least significant byte at using of a value with data type UINT64, UINT32 and UINT16, UINT08)
19 DataToSlaveTCP[3] := 16#0000;
20 DataToSlaveTCP[4] := 16#0000;
21 DataToSlaveTCP[5] := 16#0000;
22 DataToSlaveTCP[6] := 16#0001;
23
24 (* Answer from slave*)
25 //BYTE 1 :FUNCTION-> 0:Read object,1:Write object
26 //DataFromSlaveTCP[0];
27 //Byte 2: Type of object -> 1: UINT08, 2: UINT16, 3: UINT32, 4: UINT64
28 //DataFromSlaveTCP[0];
29 //Byte 3: Index of object (Most significant byte)
30 //Byte 4: Index of object (Least significant byte)
31 //DataFromSlaveTCP[1];
32 //Byte 5: Subindex of Object
33 //DataFromSlaveTCP[2];
34 //Byte 6: Response error
35 //DataFromSlaveTCP[2];
36 //Byte 7: Byte value of object (Most significant byte at using of a value with data type UINT64)
37 //Byte 8,9,10: Byte value of object (at using of a value with data type UINT64)
38 //Byte 11,12: Byte value of object (at using of a value with data type UINT64, UINT32)
39 //Byte 13: Byte value of object (at using of a value with data type UINT64, UINT32 and UINT16)
40 //Byte 14: Byte value of object (Least significant byte at using of a value with data type UINT64, UINT32 and UINT16, UINT08)
41 // DataFromSlaveTCP[6];

```

- Kontrollwort zurücksetzen

```

1  (* ***** Reset controlword ***** *)
2  (* Command To slave*)
3  //Byte 1 :Function-> 0:Read object,1:Write object
4  DataToSlaveTCP[0] := UINT_TO_WORD(SHL(16#01,8));
5  //Byte 2: Type of object -> 1: UINT08, 2: UINT16, 3: UINT32, 4: UINT64
6  DataToSlaveTCP[0] := DataToSlaveTCP[0] + 16#02;
7  //Byte 3: Index of object (Most significant byte)
8  //Byte 4: Index of object (Least significant byte)
9  DataToSlaveTCP[1] := 16#0001;
10 //Byte 5: Subindex of Object
11 DataToSlaveTCP[2] := UINT_TO_WORD(SHL(16#00,8));
12 //Byte 6: Reserved for response error
13 DataToSlaveTCP[2] := DataToSlaveTCP[2] + 16#00;
14 //Byte 7: Byte value of object (Most significant byte at using of a value with data type UINT64)
15 //Byte 8,9,10: Byte value of object (at using of a value with data type UINT64)
16 //Byte 11,12: Byte value of object (at using of a value with data type UINT64, UINT32)
17 //Byte 13: Byte value of object (at using of a value with data type UINT64, UINT32 and UINT16)
18 //Byte 14: Byte value of object (Least significant byte at using of a value with data type UINT64, UINT32 and UINT16, UINT08)
19 DataToSlaveTCP[3] := 16#0000;
20 DataToSlaveTCP[4] := 16#0000;
21 DataToSlaveTCP[5] := 16#0000;
22 DataToSlaveTCP[6] := 16#0000;
23
24 (* Answer from slave*)
25 //BYTE 1 :FUNCTION-> 0:Read object,1:Write object
26 //DataFromSlaveTCP[0];
27 //Byte 2: Type of object -> 1: UINT08, 2: UINT16, 3: UINT32, 4: UINT64
28 //DataFromSlaveTCP[0];
29 //Byte 3: Index of object (Most significant byte)
30 //Byte 4: Index of object (Least significant byte)
31 //DataFromSlaveTCP[1];
32 //Byte 5: Subindex of Object
33 //DataFromSlaveTCP[2];
34 //Byte 6: Response error
35 //DataFromSlaveTCP[2];
36 //Byte 7: Byte value of object (Most significant byte at using of a value with data type UINT64)
37 //Byte 8,9,10: Byte value of object (at using of a value with data type UINT64)
38 //Byte 11,12: Byte value of object (at using of a value with data type UINT64, UINT32)
39 //Byte 13: Byte value of object (at using of a value with data type UINT64, UINT32 and UINT16)
40 //Byte 14: Byte value of object (Least significant byte at using of a value with data type UINT64, UINT32 and UINT16, UINT08)
41 // DataFromSlaveTCP[6];

```

7 Anhang

7.1 Beispielprojekte

In der zur Application Note gehörenden ZIP-Datei gibt es 2 Projektarchive mit Beispielen für eine Ethernet-Verbindung zwischen einer CECC-S und einem Ventil-Ansteuermodul VAEM-V sowie zwischen einer CPX-CEC-M1 und einem Ventil-Ansteuermodul VAEM.

- „Connect VAEM_V with CECC_S via Modbus TCP_CODESYS3_5_SP7P2.projectarchive“
- „Connect VAEM_V with CPX_CEC_M1 via Modbus TCP_CODESYS3_5_SP7P2.projectarchive“

7.2 Verbindung mit 2 VAEM-V

Zur Verbindung mit einem zweiten Ventil-Ansteuermodul VAEM-V müssen die Schritte im Kapitel 5.3 „Slave konfigurieren“ erneut durchgeführt werden.

