

Connection with VAEM-V via Modbus/TCP in CODESYS

Brief description of how to create an Ethernet connection in CODESYS between a Festo controller (CPX- or CECC) and a valve control module VAEM-V-S8EPRS2

VAEM-V

TitleConnection with VAEM-V via Ethernet in CODESYS
Version 1.10
Document number 100334
Original de
AuthorFesto
Last saved 17.05.2021

Copyright notice

This documentation is the intellectual property of Festo SE & Co. KG, who also holds the exclusive copyright. Any modification of the content, duplication or reprinting of this documentation as well as distribution to third parties can only be made with the express consent of Festo SE & Co. KG.

Festo SE & Co KG reserves the right to make modifications to this document in whole or in part. All brand and product names are trademarks or registered trademarks of their respective owners.

Legal information

Hardware, software, operating systems and drivers may only be used for the applications described and only in conjunction with the components recommended by Festo SE & Co. KG.

Festo SE & Co. KG does not accept any liability for damages arising from the use of any incorrect or incomplete information contained in this documentation or any information missing therefrom.

Defects resulting from the improper handling of devices and modules are excluded from the warranty.

The data and information specified in this document should not be used for the implementation of safety functions related to the protection of personnel and machinery.

No liability is accepted for claims for consequential damages arising from failure or malfunction. Otherwise, the clauses concerning liability included in the terms and conditions of delivery, payment and use of software of Festo SE & Co. KG shall apply, which can be found at www.festo.com or provided upon request.

None of the information contained in this document is a guarantee of product features in the legal sense, in particular with regard to functionality, condition or quality.

The information in this document serves only as basic information for the implementation of a specific, hypothetical application and is in no way intended as a substitute for the operating instructions of the respective manufacturers and the design and testing of the respective application by the end user.

The respective operating instructions for products from Festo can be found at www.festo.com/sp.

End users of this document (function and application) must themselves verify that all functions described herein also work correctly in their own applications. Even after examining this document and while using the specifications contained herein, end users nevertheless remain solely responsible for their own applications.

Table of contents

1	Components/software used	5
2	Documentation used	6
3	System examples.....	7
3.1	CPX-CEC-M1 and VAEM-V	7
3.2	CECC-S and VAEM-V	7
4	Configuration of the IP address of VAEM-V.....	8
5	Configuration of a connection via Modbus/TCP in CODESYS.....	9
5.1	Adding an Ethernet connection	9
5.2	Configuring the master device.....	10
5.3	Configuring the slave device	11
6	Reading and writing a function object.....	14
6.1	Global communication data.....	14
6.2	Reading the status word.....	15
6.3	Write the value to the control word to start	16
7	Appendix.....	18
7.1	Sample project	18
7.2	Connection with 2 VAEM-V units	18

1 Components/software used

Type/name	Software/firmware version	Part number
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	

Table 1.1: Components/software used

Link to Festo Support Portal

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

2 Documentation used

Type/name	Version/date	Part number
VAEM-V manual	2020-	8127480
CECC-S manual	2016-03a	8036061
CPX-CEC-M1-V3 manual	2014-11c	8042601

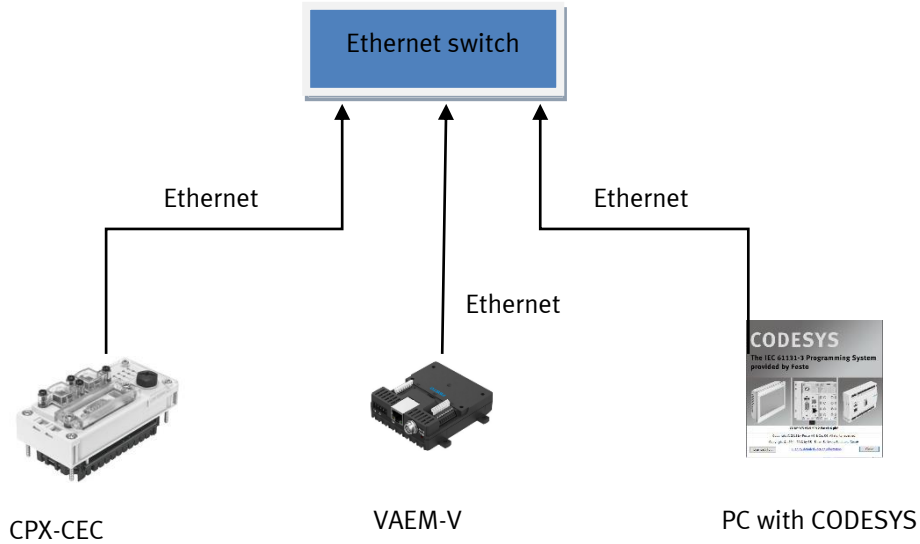
Table 2.1: Documentation used

Link to Festo Support Portal

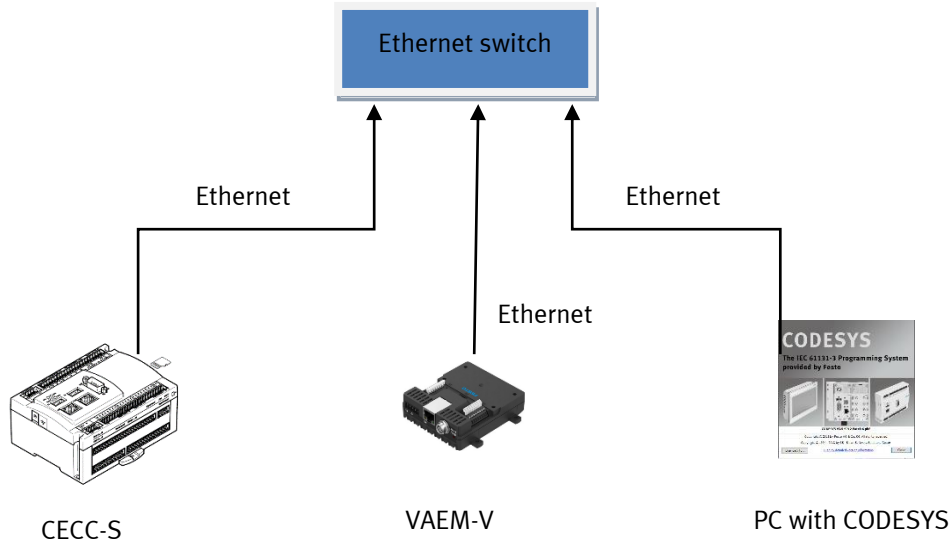
<https://www.festo.com/SupportPortal/>

3 System examples

3.1 CPX-CEC-M1 and VAEM-V

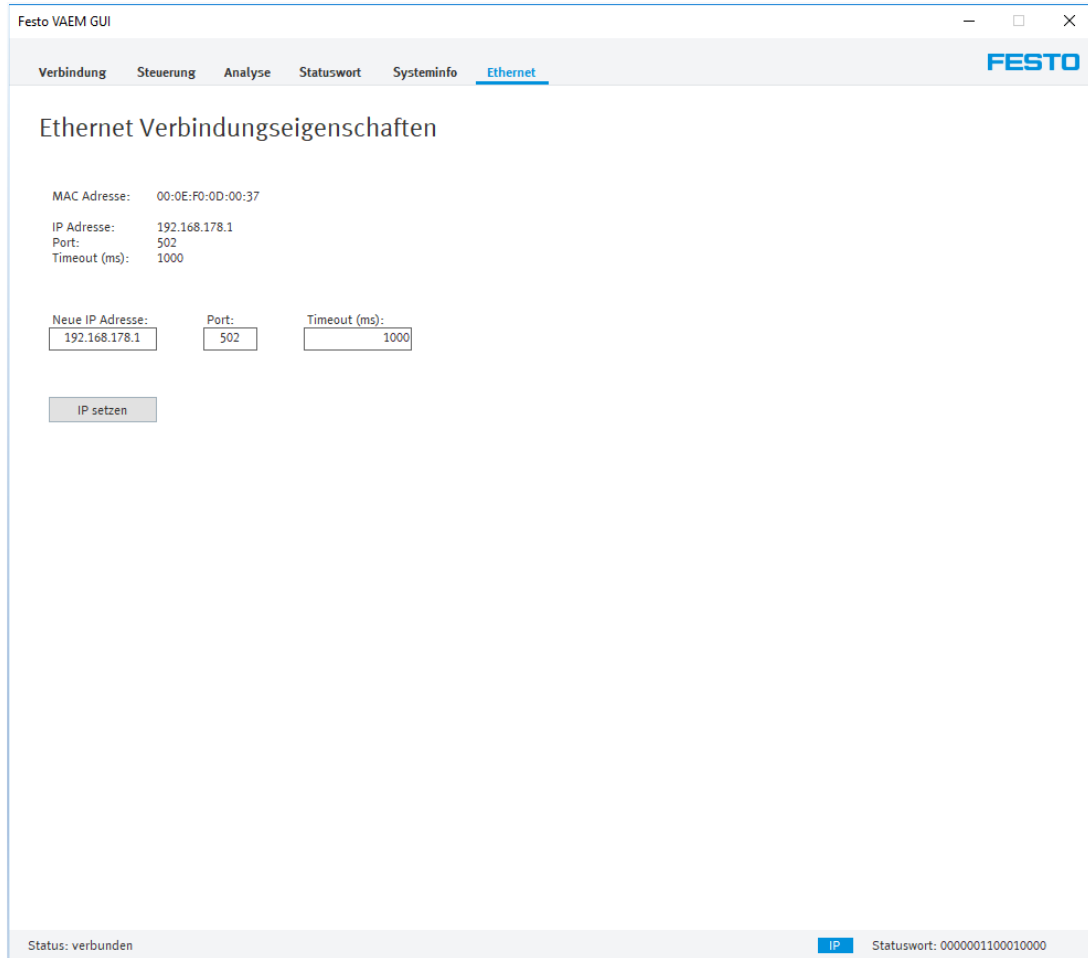


3.2 CECC-S and VAEM-V



4 Configuration of the IP address of VAEM-V

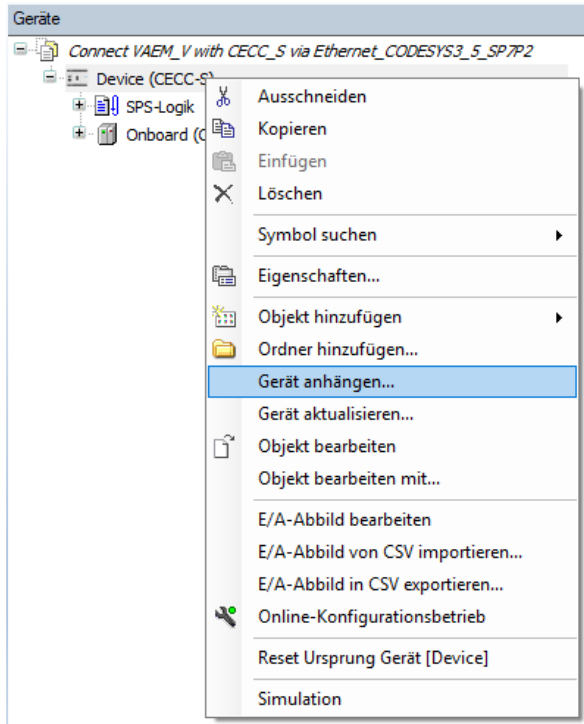
The VAEM GUI, which can be downloaded from the Support Portal, can be used to commission the VAEM-V valve control module and configure the Ethernet connection setting. (e.g., the standard IP address 192.168.178.1).



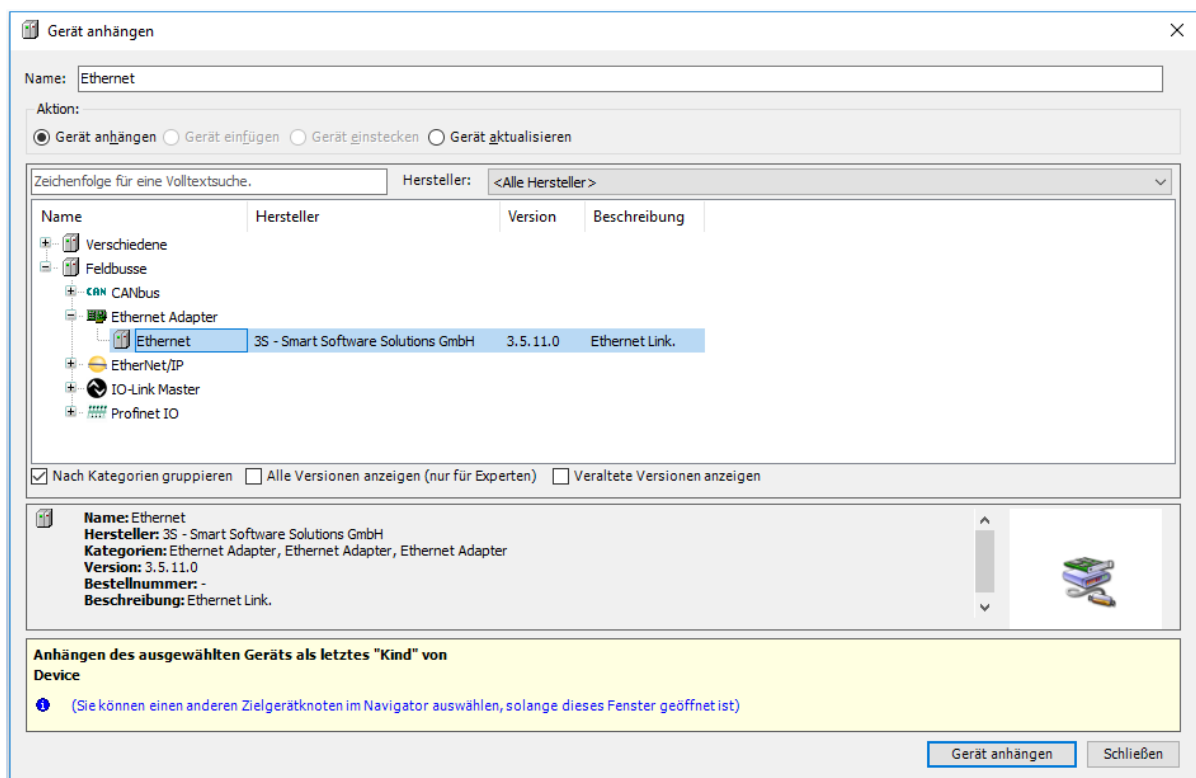
5 Configuration of a connection via Modbus/TCP in CODESYS

5.1 Adding an Ethernet connection

1. Right-click the device, and then open the interface selection with “Add device”.

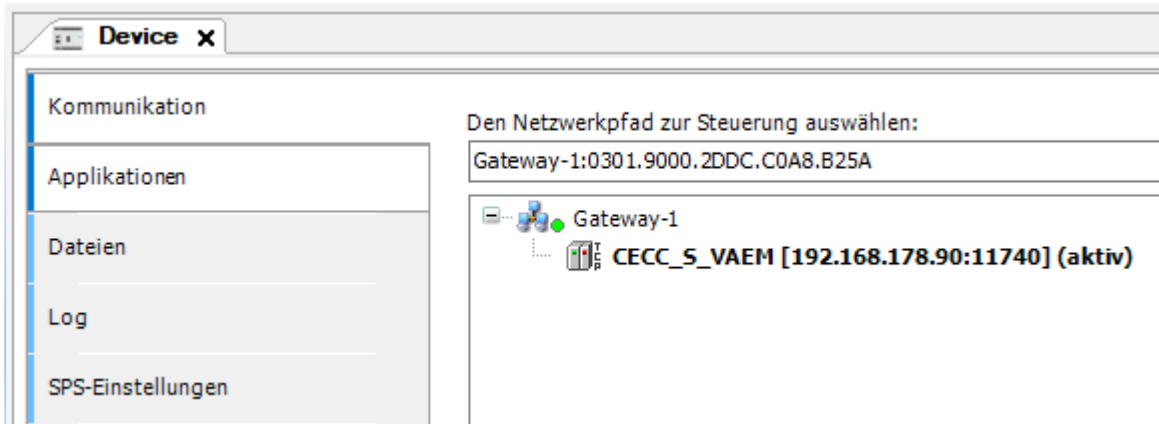


2. Select the Ethernet interface, attach it and close the selection.

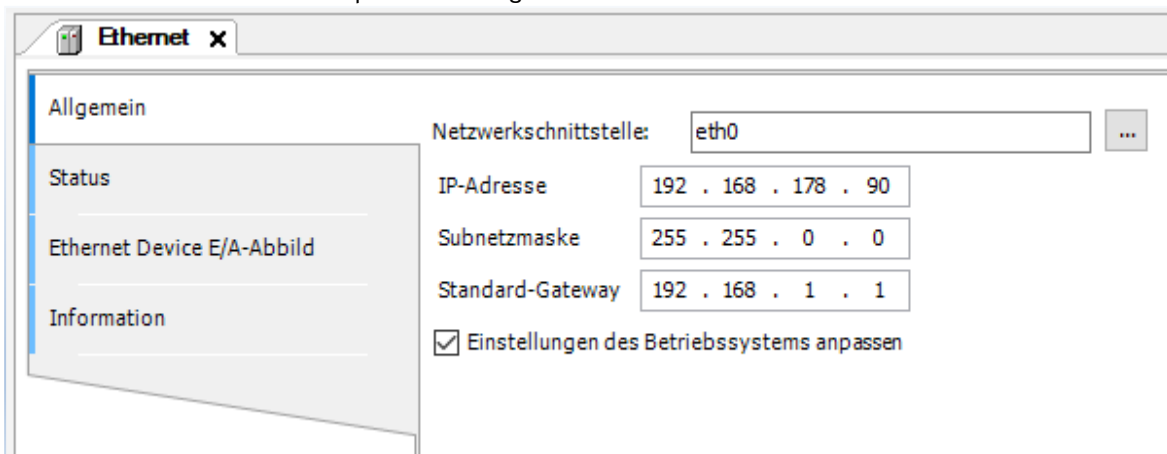


5.2 Configuring the master device

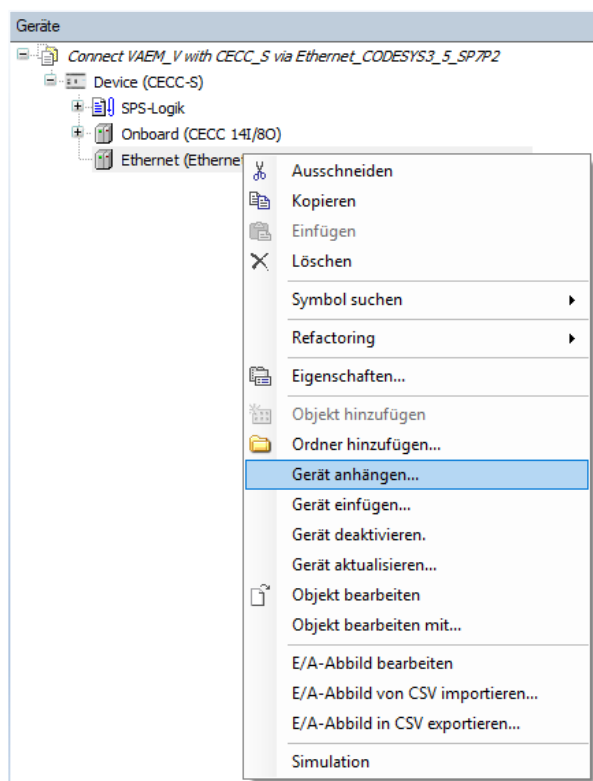
1. Double-click “Device” to activate the controller.



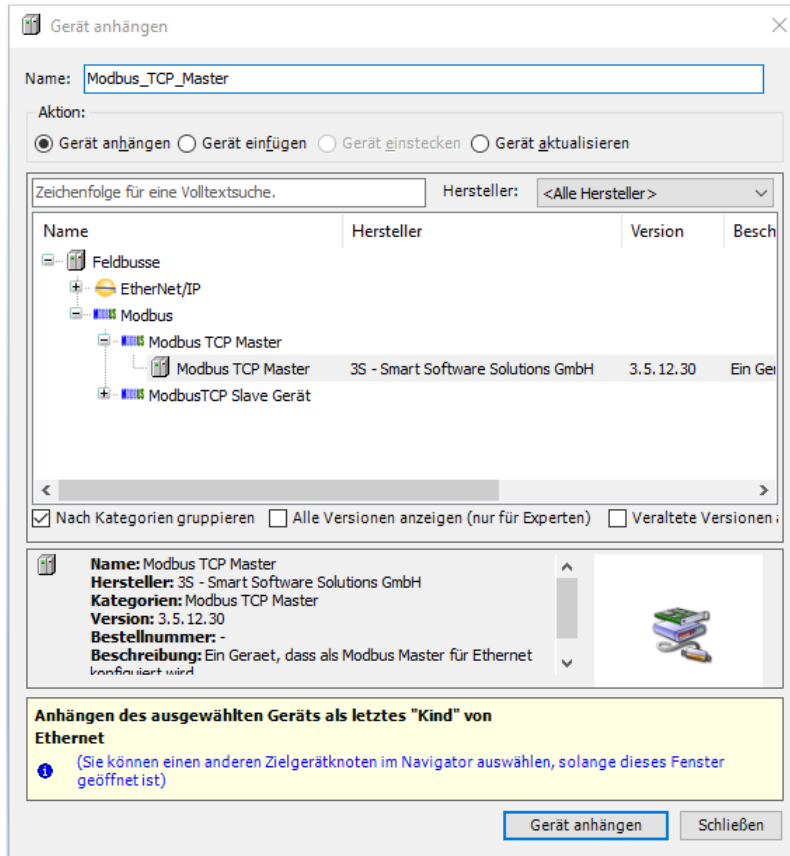
2. Double-click “Ethernet” to open the setting and then select the network interface.



3. Right-click “Ethernet”, and then open the selection with “Attach device”.

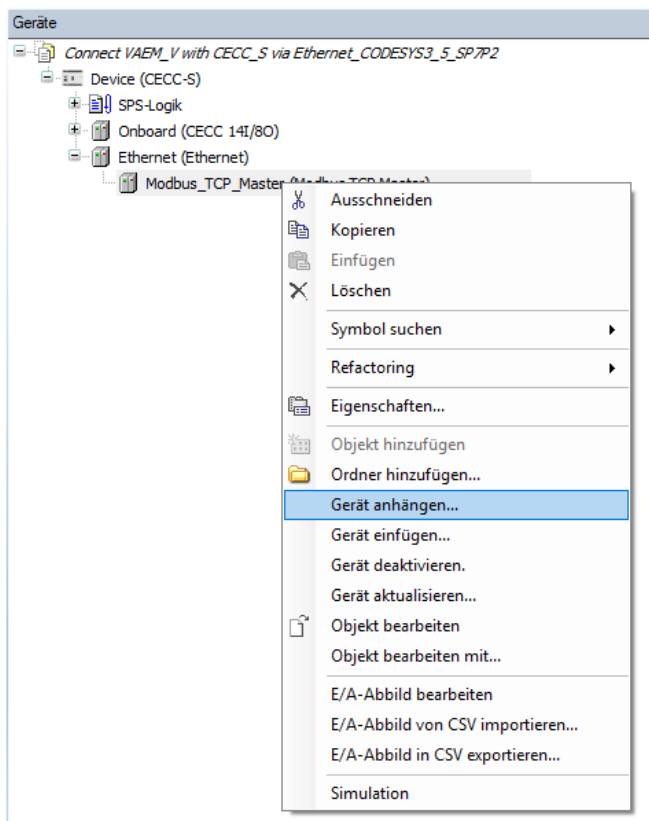


3. Select the Modbus TCP master, attach it and close the selection.

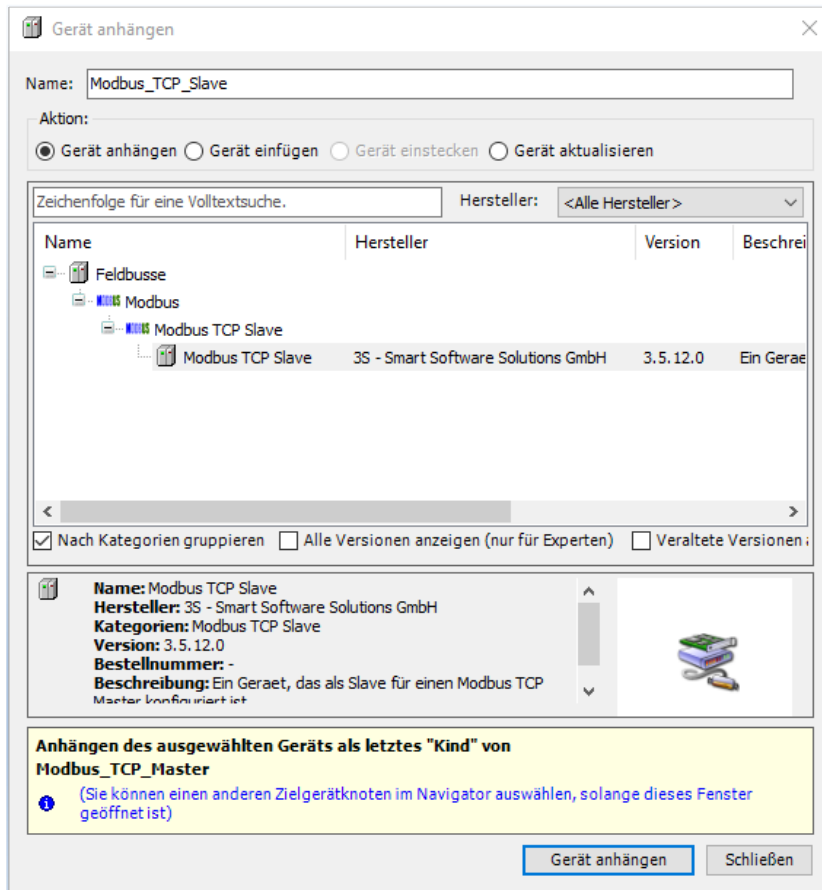


5.3 Configuring the slave device

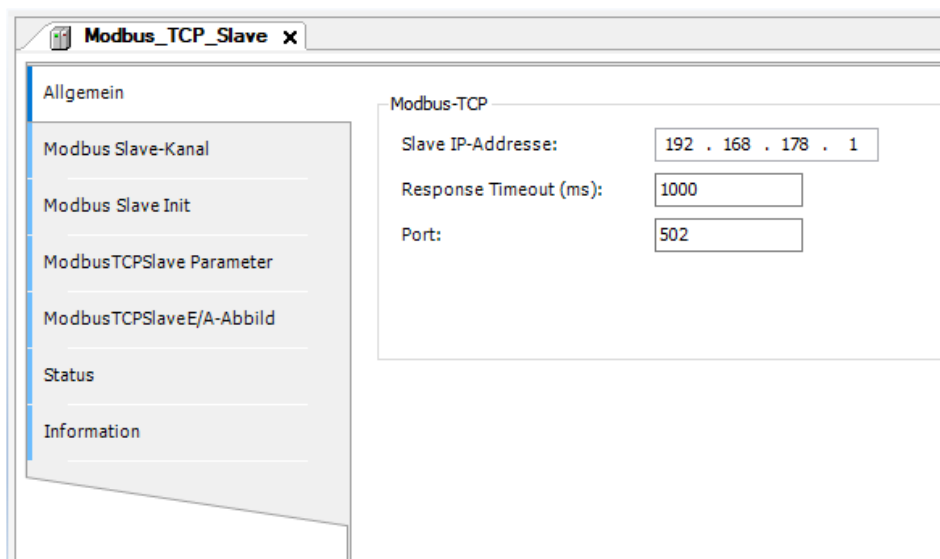
1. Right-click "Modbus_TCP_Master", and then use "Attach device" to open the selection.



2. Select the Modbus TCP slave, attach it and close the selection.



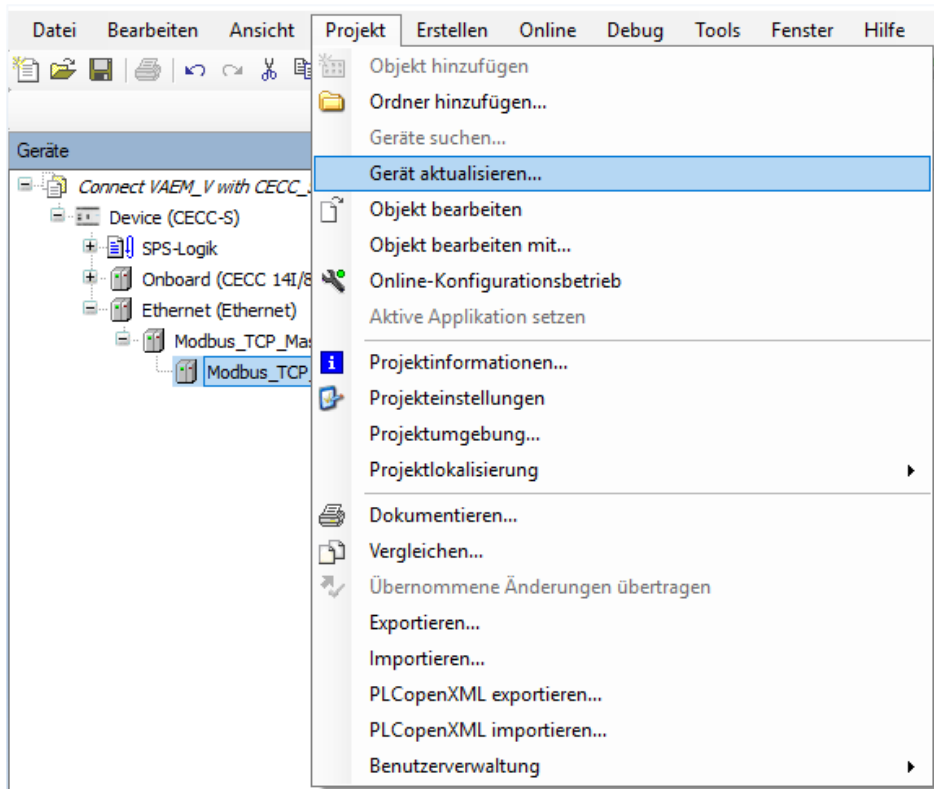
4. Double-click “Modbus_TCP_Slave” to open the setting and configure the IP address of the valve control module.



- Click “Modbus Slave Channel” and then “Add channel” to create and configure a Modbus channel (e.g., the data update has a cycle time of 100 ms).

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 is the 'READ Register' section with 'Offset' (0), 'Länge' (7), and 'Fehlerbehandlung' (Letzen Wert beibehalten). The 'WRITE Register' section has 'Offset' (0) and 'Länge' (7). At the bottom are 'OK' and 'Abbrechen' buttons.

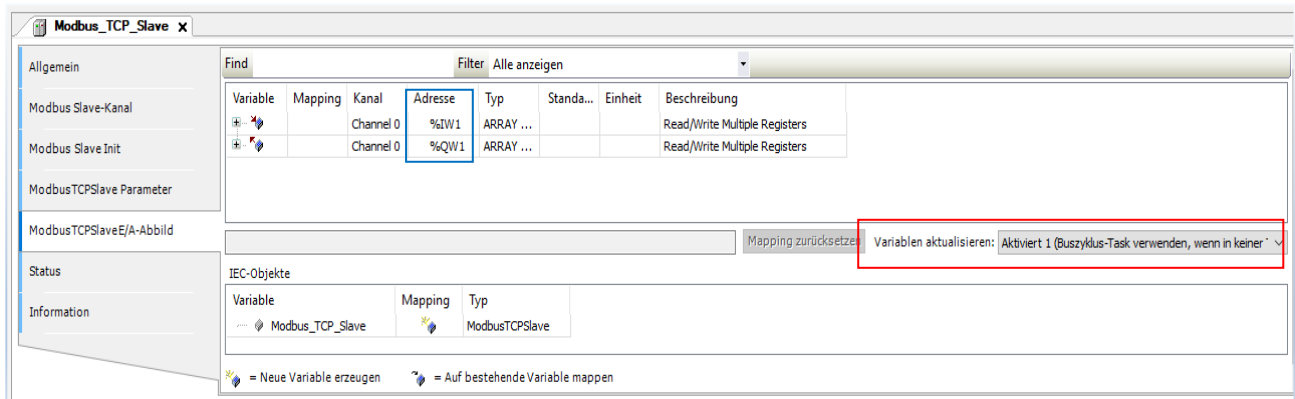
- Select “Modbus_TCP_Slave” in the “Devices” window and open the “Project” menu. Then click “Update device”.



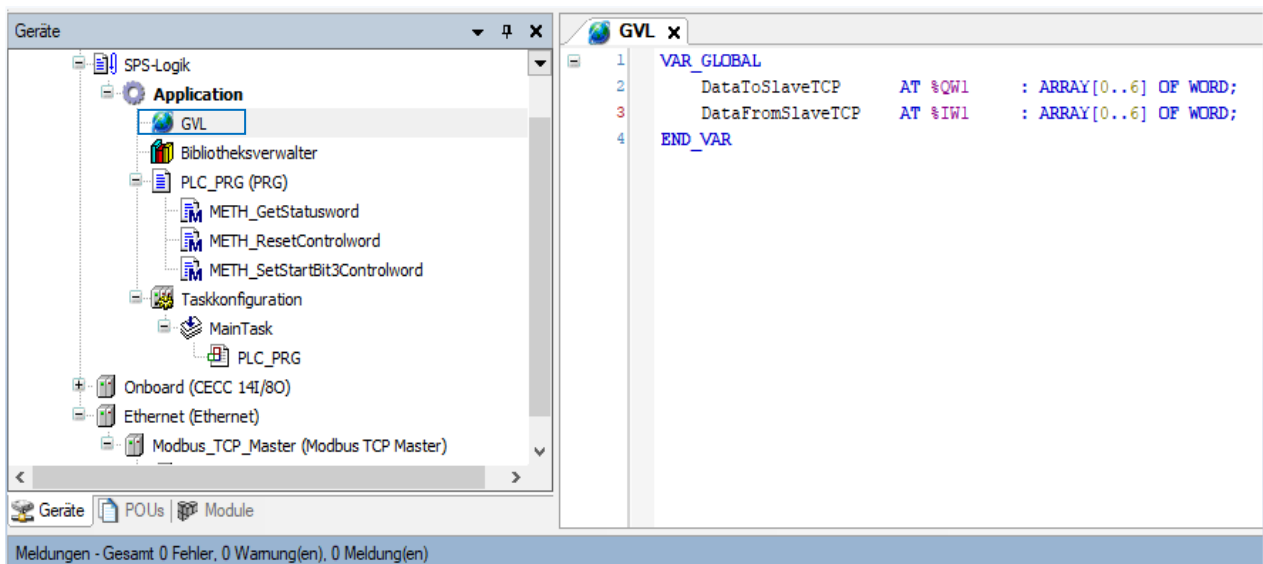
6 Reading and writing a function object

6.1 Global communication data

- The “ModbusTCPSlave I/O-Map” tab is located on the “Modbus_TCP_Slave” screen. The addresses for communication with the valve control module VAEM are stored here. For the outgoing data to the valve control module VAEM, this would be address %QW1 in the example, and address %IW1 for the data returned by the valve control module VAEM.



- “Update variables” by selecting “Enable 1 (use bus cycle task if not in task)”.
- Create a global variable list.



6.2 Reading the status word

```

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 Write the value to the control word to start

- Set start bit 3 of the control word to a value of 1

```

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];

```

- Reset the control word

```

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 Appendix

7.1 Sample project

The ZIP file belonging to the application note contains 2 project archives with examples of an Ethernet connection between a CECC-S and a valve control module VAEM-V and between a CPX-CEC-M1 and a valve control module 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 Connection with 2 VAEM-V units

To connect to a second valve control module VAEM-V, the steps in chapter 5.3 “Configuring the slave” must be carried out again.

