Industrial
Automation

GETTING STARTED

VT250-57x - HMI PLC with Modbus TCP

Sense it! Connect it! Bus it! Solve it!

Edition 08/2010

Subject to alterations without notice

Warning!
Before commencing the installation

- Disconnect the power supply of the device.
- Ensure that devices cannot be accidentally restarted.
- Verify isolation from the supply.
- Earth and short circuit.
- Cover or enclose neighboring units that are live.
- Follow the engineering instructions of the device concerned.
- Only suitably qualified personnel in accordance with EN 50 110-1/-2 (VDE 0 105 Part 100) may work on this device/system.
- Before installation and before touching the device ensure that you are free of electrostatic charge.
- The functional earth (FE) must be connected to the protective earth (PE) or to the potential equalization. The system installer is responsible for implementing this connection.
- Connecting cables and signal lines should be installed so that inductive or capacitive interference do not impair the automation functions.
- Install automation devices and related operating elements in such a way that they are well protected against unintentional operation.
- Suitable safety hardware and software measures should be implemented for the I/O interface so that a line or wire breakage on the signal side does not result in undefined states in the automation devices.
- Ensure a reliable electrical isolation of the low voltage for the 24 volt supply. Only use power supply units complying with IEC 60 364-4-41 (VDE 0 100 Part 410) or HD 384.4.41 S2.
- Deviations of the mains voltage from the rated value must not exceed the tolerance limits given in the specifications, otherwise this may cause malfunction and dangerous operation.
- Emergency stop devices complying with IEC/EN 60 204-1 must be effective in all operating modes of the automation devices. Unlatching the emergency-stop devices must not cause restart.
- Devices that are designed for mounting in housings or control cabinets must only be operated and controlled after they have been installed with the housing closed. Desktop or portable units must only be operated and controlled in enclosed housings.
- Measures should be taken to ensure the proper restart of programs interrupted after a voltage dip or failure. This should not cause dangerous operating states even for a short time. If necessary, emergency-stop devices should be implemented.
- Wherever faults in the automation system may cause damage to persons or property, external measures must be implemented to ensure a safe operating state in the event of a fault or malfunction (for example, by means of separate limit switches, mechanical interlocks etc.).
- The electrical installation must be carried out in accordance with the relevant regulations (e. g. with regard to cable cross sections, fuses, PE).
- All work relating to transport, installation, commissioning and maintenance must only be carried out by qualified personnel. (IEC 60 364 and HD 384 and national work safety regulations).
- All shrouds and doors must be kept closed during operation.

Table of contents

About this manual

About this manual

General

This Getting Started describes the use of the VT250 with Modbus TCP-master function. In CoDeSys 3.4 Patch 2, the Modbus TCP-Master from 3S is used.

Additional documentation

- D301189 "Getting Started - VT250-57P HMI PLC with PROFIBUS-DP"
- D301191 "VT250 - Hardware-description"
- D301195 "Getting Started - VT250-57P HMI PLC with CANopen"
- D301218 "Getting Started - VT250-57x Visualization of Step®-projects"
- D301226 "Getting Started - VT250-57x - QViS with Modbus TCP-driver"

Description of symbols used

Warning

This sign can be found next to all notes that indicate a source of hazards. This can refer to danger to personnel or damage to the system (hardware and software) and to the facility.

This sign means for the operator: work with extreme caution.

Attention

This sign can be found next to all notes that indicate a potential source of hazards.

This can refer to possible danger to personnel and damages to the system (hardware and software) and to the facility.

Note

This sign can be found next to all general notes that supply important information about one or more operating steps.

These specific notes are intended to make operation easier and avoid unnecessary work due to incorrect operation.

Overview

⚠ **Attention**

Please read this section carefully. Safety aspects cannot be left to chance when dealing with electrical equipment.

This manual includes all information necessary for the prescribed use of TURCK HMIs. It has been specially conceived for personnel with the necessary qualifications.

Prescribed use

Appropriate transport, storage, deployment and mounting as well as careful operating and thorough maintenance guarantee the trouble-free and safe operation of these devices.

⚠ **Warning**

The devices described in this manual must be used only in applications prescribed in this manual or in the respective technical descriptions, and only with certified components and devices from third party manufacturers.

Notes concerning planning /installation of this product

⚠ **Warning**

All respective safety measures and accident protection guidelines must be considered carefully and without exception.

1 Getting Started

Used hard-/ software

Hardware

- VT250-57P, firmware VT250-57P-L7-DPM
- BL67-GW-EN (IP-address 192.168.1.12)
  with:
  - Slot 1: BL67-4DI-PD
  - Slot 2: BL67-2AI-I with analog sensor Bi5-M18-Li2
  - Slot 3: BL67-8DO-0.5A-P

Software

- CoDeSys 3.4 Patch2

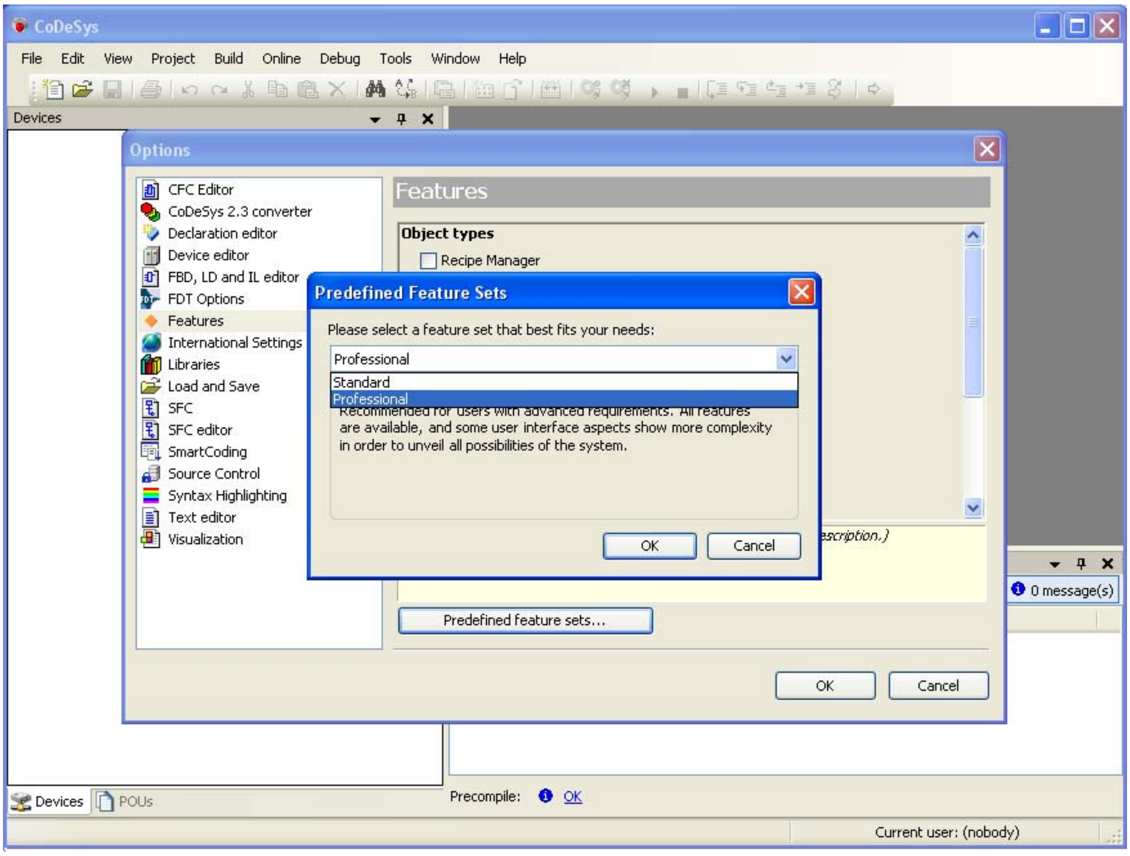
Network configuration and programming in CoDeSys

Open CoDeSys via "Start → All Programs → 3S CoDeSys → CoDeSys → CoDeSys V 3.4".

Predefined Feature Sets

In this example, CoDeSys is run with the "Professional feature set" not with the "Standard feature set". This setting has influence on different CoDeSys functions and can be changed via "Tools → Options..." in the "Features" under "Predefined feature sets...". For further information concerning this topic, please read the CoDeSys online help.

Figure 1: Predefined feature sets



Predefined Feature Sets

Please select a feature set that best fits your needs:

Professional

Recommended for users with advanced requirements. All features are available, and some user interface aspects show more complexity in order to unveil all possibilities of the system.

Creating a new project

1 Create a new CoDeSys-project using the "File → New project" command.

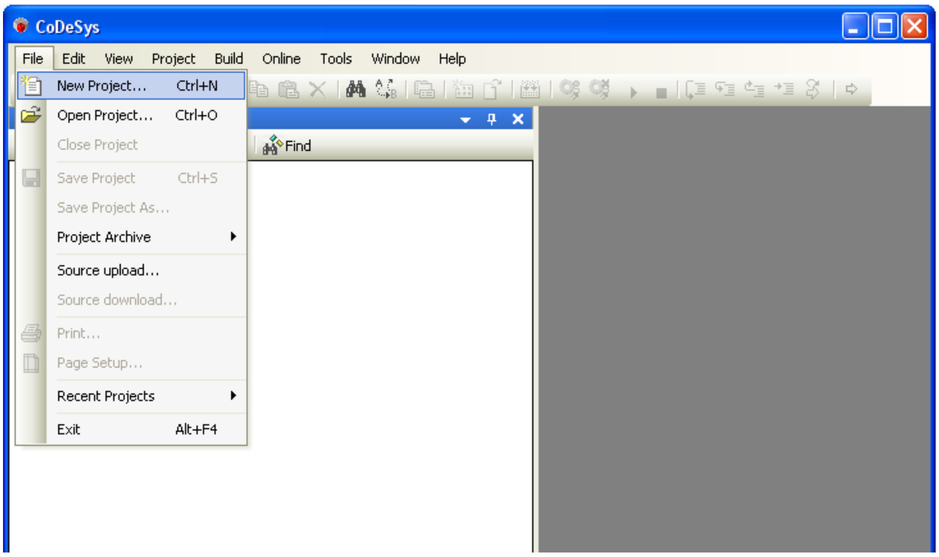
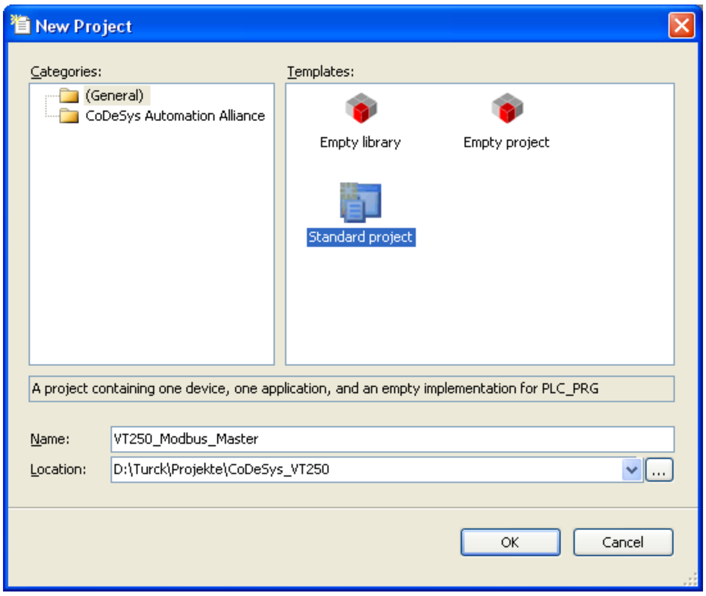


Figure 2: New project

2 In the opening dialog box "New Project", activate the option "Standard project", assign a project name and define the storage location for the project.



A project containing one device, one application, and an empty implementation for PLC_PRG

Name: VT250_Modbus_Master

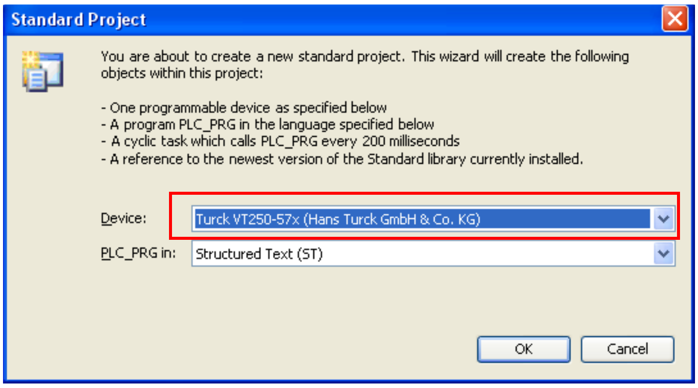
Location: D:\Turck\Projekte\CoDeSys_VT250

Figure 3: Standard project

3 In the new dialog box "Standard project" the used TURCK VT250-57x (Hans Turck GmbH & Co. KG) is selected as "Device".

4 Please define your preferred programming language as well. In this example, Structured Text is used.

Figure 4: Selection of the VT250-57x



Standard Project

You are about to create a new standard project. This wizard will create the following objects within this project:

- One programmable device as specified below
- A program PLC_PRG in the language specified below
- A cyclic task which calls PLC_PRG every 200 milliseconds
- A reference to the newest version of the Standard library currently installed.

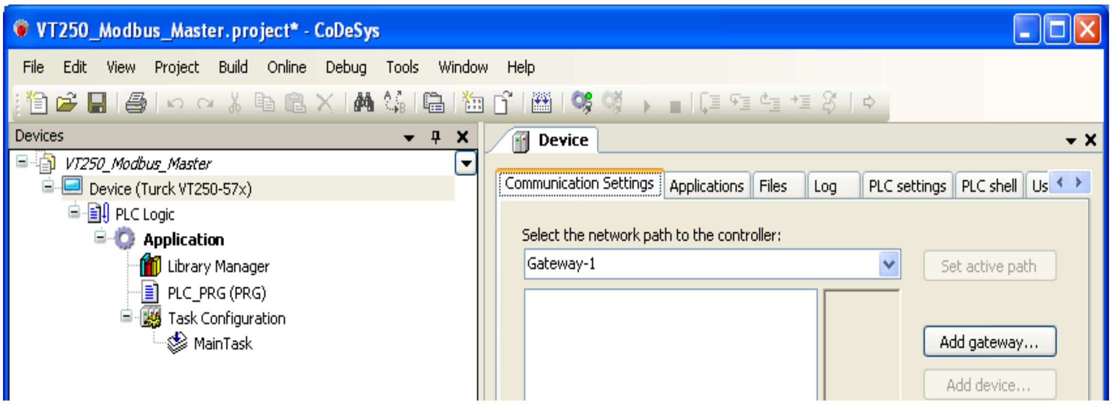
Device: Turck VT250-57x (Hans Turck GmbH & Co. KG)

PLC_PRG in: Structured Text (ST)

5 The new project is created.

6 In CoDeSys, the project tree is build up as follows:

Figure 5: Project tree



Note

If the window "devices" should not be displayed, it can be activated via "View → Devices".

Defining the communication settings

Double-clicking the "Device VT250-57x" opens the corresponding editors. The communication path (Gateway) to the HMI is defined in the "Communication Settings" tab.

Note

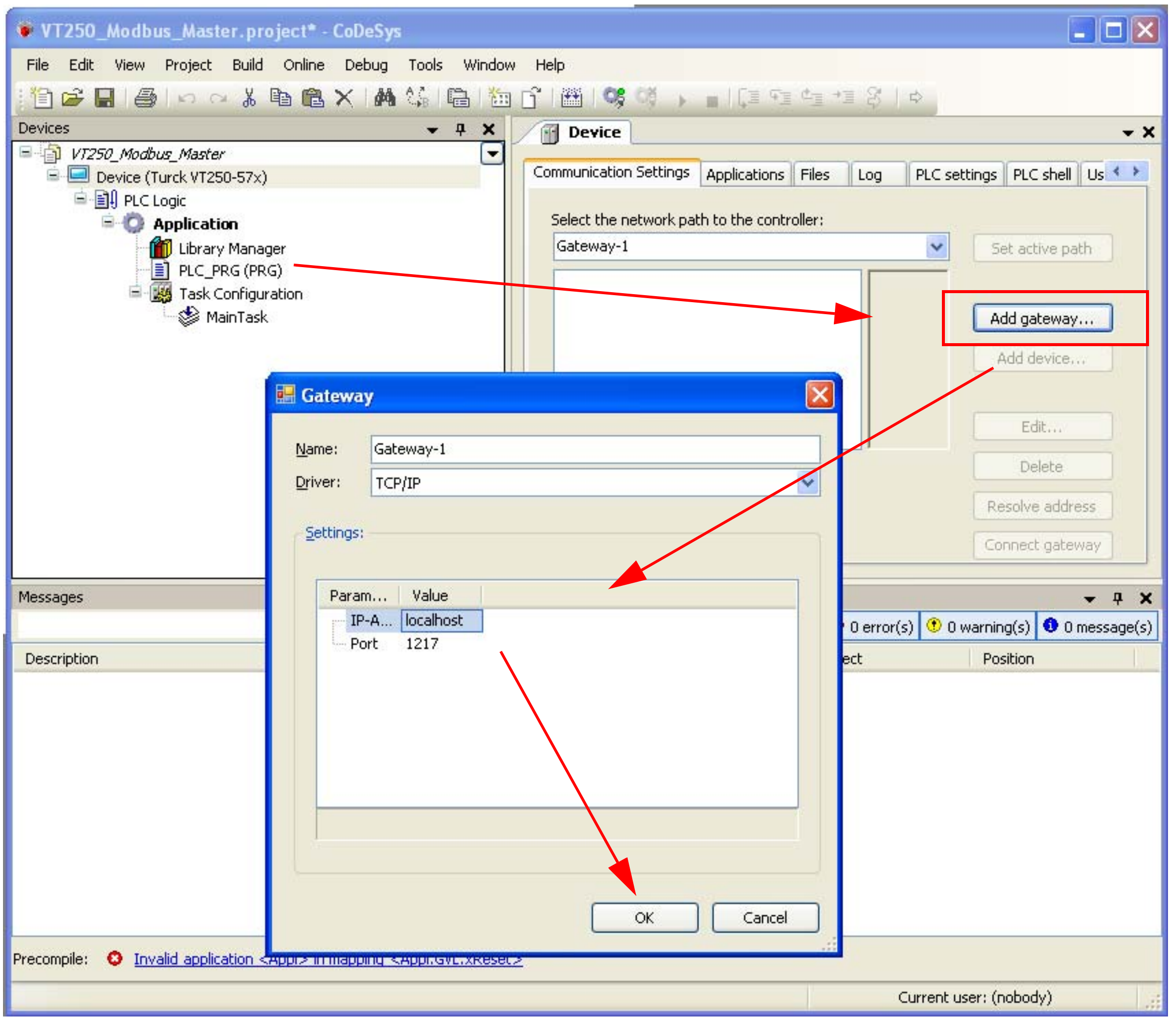
Please observe, that the VT250-57x and your PC on which CoDeSys is running, are nodes of the same Ethernet network! If this is not the case, a communication between the two is not possible.

Gateway definition

1 Use the "Add gateway"-button to open the dialog box "Gateway" and, where necessary, assign a new gateway name.

2 Keep the setting "localhost" or define an IP-address for the gateway instead.
When using the setting "localhost", the CoDeSys communication gateway of the PC, on which this CoDeSys-installation is running, is used as programming interface.

Figure 6: Communication settings



Select the network path to the controller: Gateway-1

Gateway

Name: Gateway-1

Driver: TCP/IP

Settings:

Param...	Value
IP-A...	localhost
Port	1217

0 error(s) 0 warning(s) 0 message(s)

Precompile: Invalid application <Appl> in mapping <Appl.GVL.xReset>

Current user: (nobody)

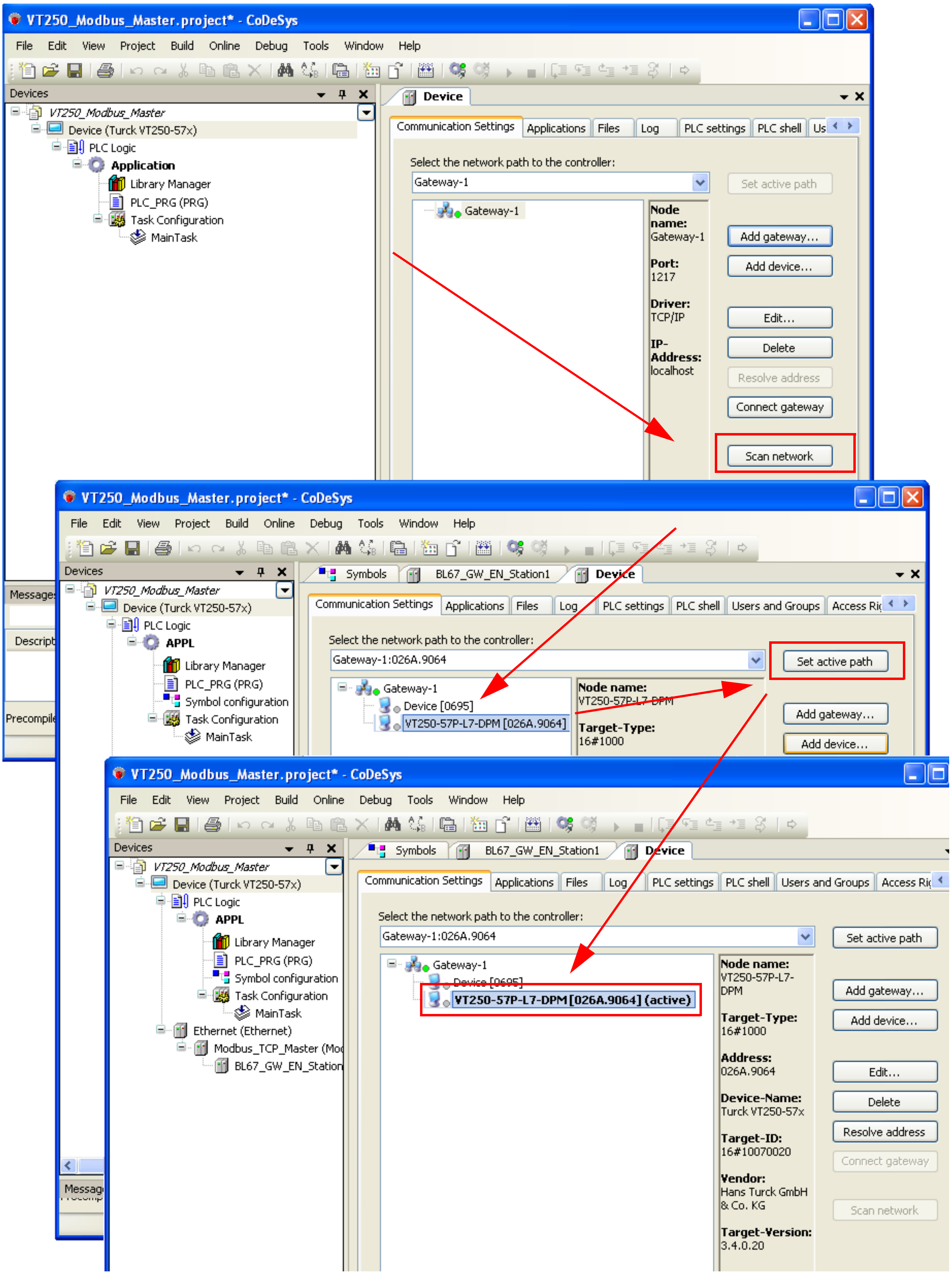
Setting the communication path

1 Mark the gateway and scan the network via the respective button. The connected VT250-57x, in this example the VT250-57P-L7-DPM, is found.

2 Mark the VT250-57x and set the active path using the "Set active path" button.

Figure 7:

Setting the communication path



VT250_Modbus_Master.project* - CoDeSys

Select the network path to the controller: Gateway-1

Node name: Gateway-1

Port: 1217

Driver: TCP/IP

IP-Address: localhost

Select the network path to the controller: Gateway-1:026A.9064

Node name: VT250-57P-L7-DPM

Target-Type: 16#1000

Address: 026A.9064

Device-Name: Turck VT250-57x

Target-ID: 16#10070020

Vendor: Hans Turck GmbH & Co. KG

Target-Version: 3.4.0.20

Renaming the application

The operating system rcX uses the 8.3-filename convention.

Therefore, it has to be observed, that the name of the stored application file also corresponds to this convention.

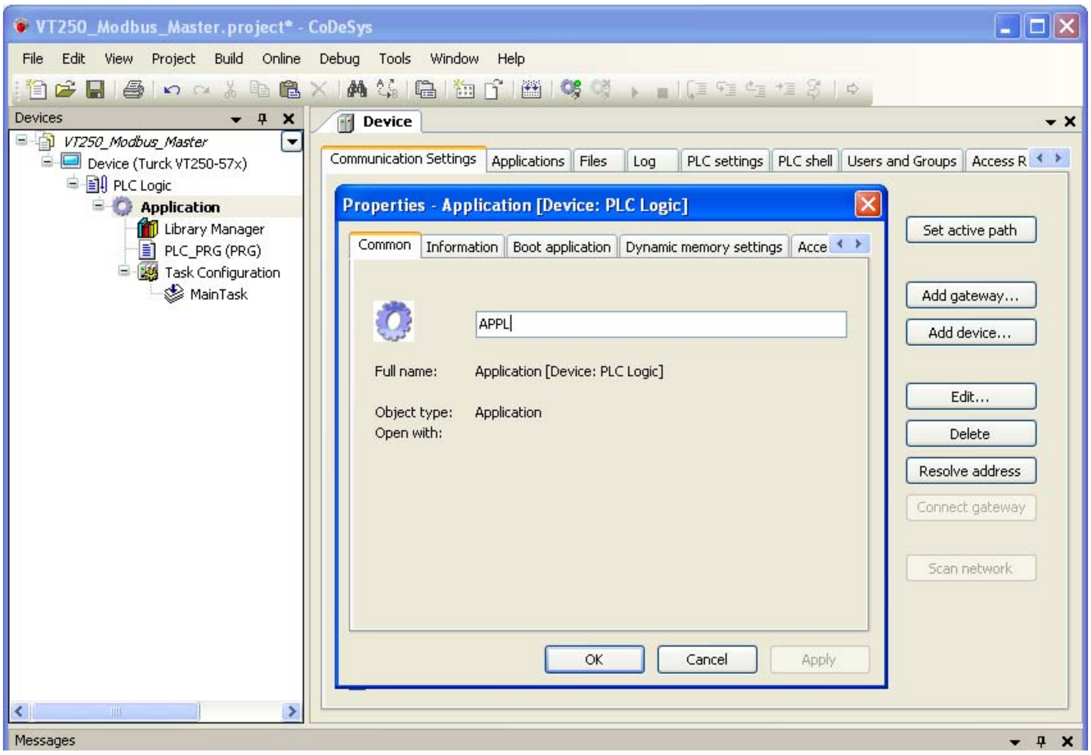
Note

Using an application name, which does not correspond to the 8.3-naming convention, may cause data loss and communication problems between PC and VT250-57×!

1 Open the context menu by right-clicking the "Application" and select "Properties".

2 In the "Common"-tab, define an application name with a maximum of 7 characters and confirm it with OK. The following warning can be confirmed either.

Figure 8: Renaming the application



Properties - Application [Device: PLC Logic]

APPL

Full name: Application [Device: PLC Logic]

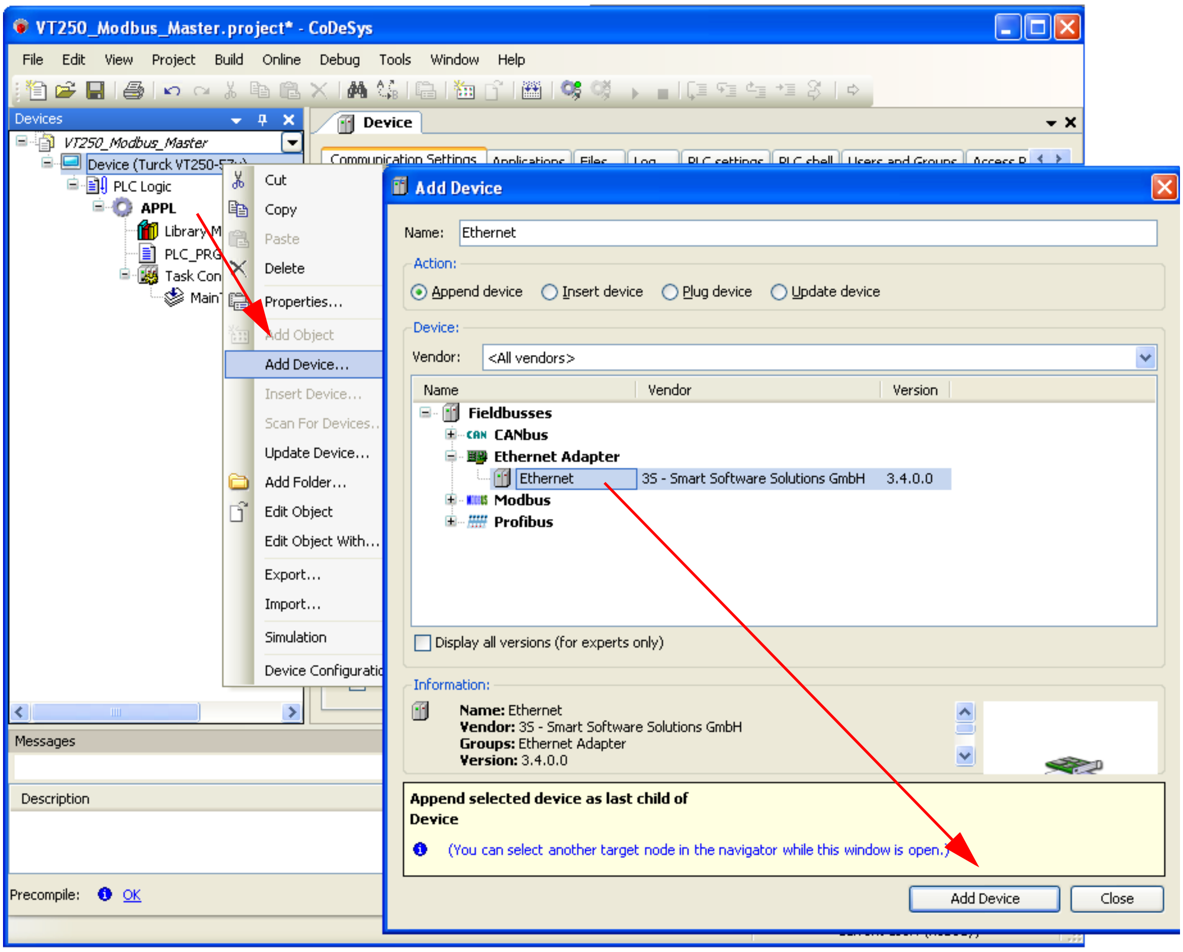
Object type: Application

Open with:

Adding the Ethernet Adapter

Open again the context menu by right-clicking the Device Turck VT250-57x entry. In the dialog "Add Device", select the 3S Ethernet Adapter under "Fieldbusses → Ethernet Adapter" and add it to the project tree.

Figure 9: Adding the Ethernet Adapter as device



Name: Ethernet

Action: ● Append device ○ Insert device ○ Plug device ○ Update device

Device:

Vendor: <All vendors>

Name	Vendor	Version
Fieldbusses		
CANbus		
Ethernet Adapter		
Ethernet	3S - Smart Software Solutions GmbH	3.4.0.0
Modbus		
Profibus		

□ Display all versions (for experts only)

Information:

Name: Ethernet
Vendor: 3S - Smart Software Solutions GmbH
Groups: Ethernet Adapter
Version: 3.4.0.0

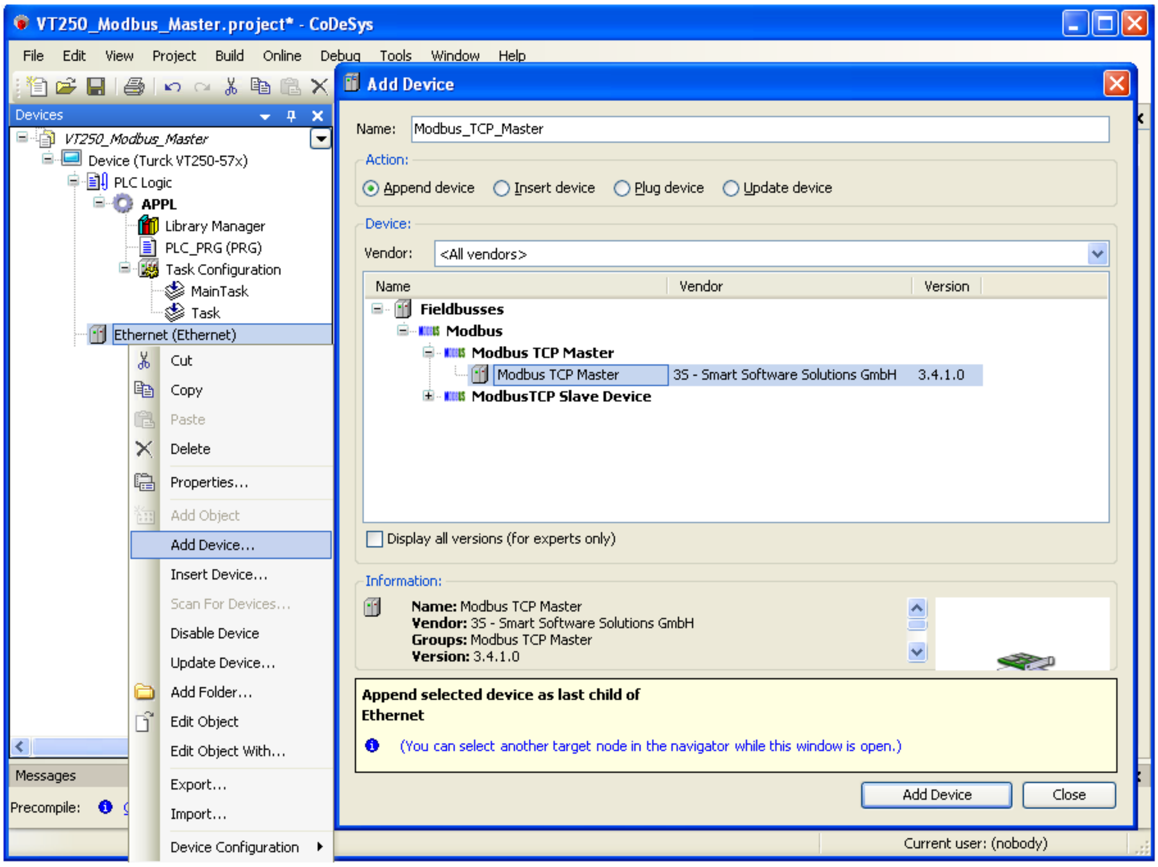
Append selected device as last child of
Device

(You can select another target node in the navigator while this window is open.)

Adding the Modbus Master

A right-click on the Ethernet Adapter opens the context menu. Select "Add Device" and add the Modbus TCP Master to the project tree.

Figure 10: Adding the Modbus Master



Name: Modbus_TCP_Master

Action:

● Append device ○ Insert device ○ Plug device ○ Update device

Device:

Vendor: <All vendors>

Name	Vendor	Version
Fieldbusses		
Modbus		
Modbus TCP Master		
Modbus TCP Master	3S - Smart Software Solutions GmbH	3.4.1.0
ModbusTCP Slave Device		

□ Display all versions (for experts only)

Information:

Name: Modbus TCP Master
Vendor: 3S - Smart Software Solutions GmbH
Groups: Modbus TCP Master
Version: 3.4.1.0

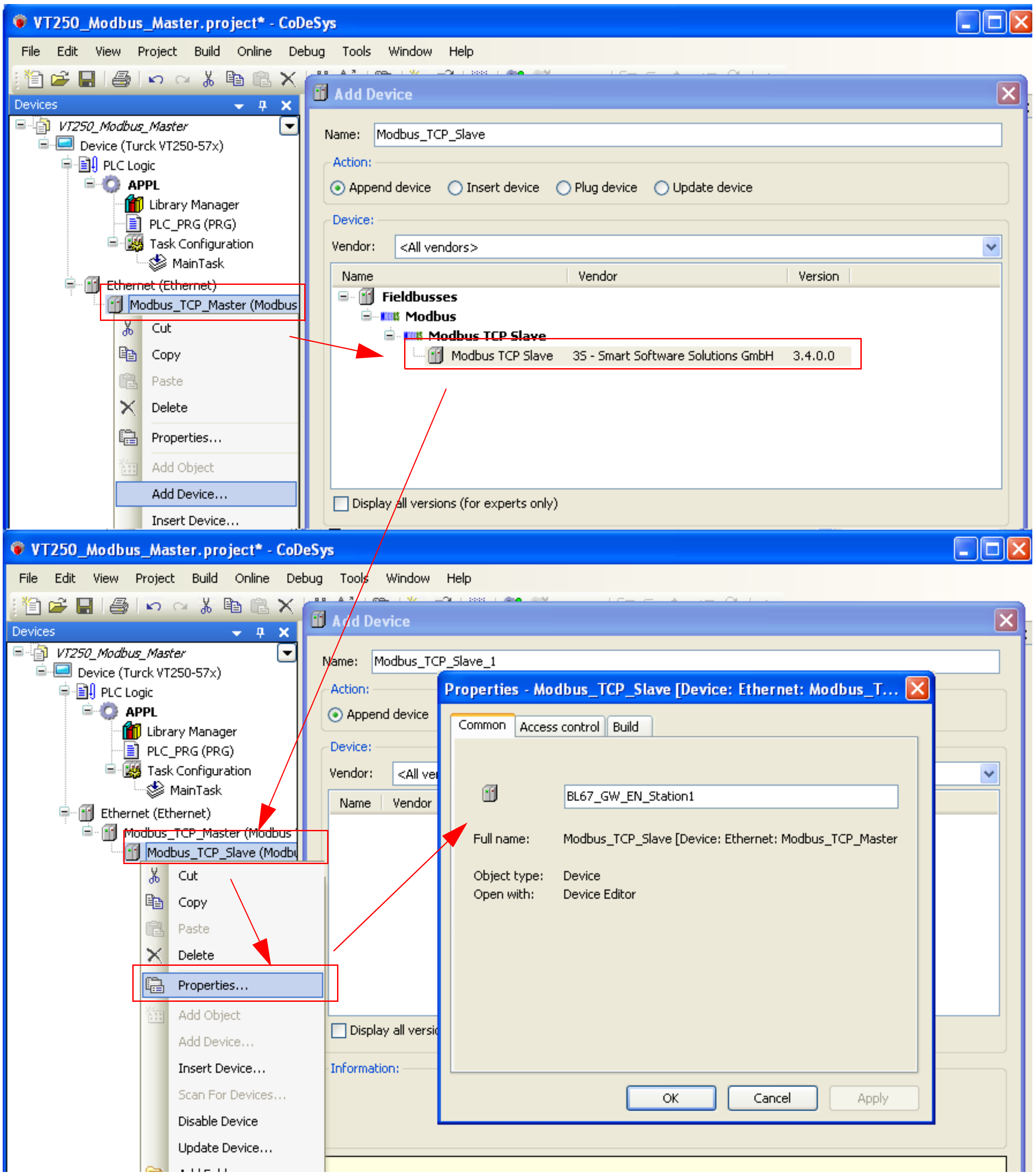
Append selected device as last child of Ethernet

(You can select another target node in the navigator while this window is open.)

Adding a Modbus TCP slave

1 Now, add the Modbus TCP slaves to the project and rename them if necessary.

Figure 11: Selecting a slave



VT250_Modbus_Master.project* - CoDeSys

Add Device

Name: Modbus_TCP_Slave

Action: ● Append device ○ Insert device ○ Plug device ○ Update device

Device: Vendor: <All vendors>

Name	Vendor	Version
Fieldbusses		
Modbus		
Modbus TCP Slave		
Modbus TCP Slave	3S - Smart Software Solutions GmbH	3.4.0.0

□ Display all versions (for experts only)

Add Device

Name: Modbus_TCP_Slave_1

Action: ● Append device

Properties - Modbus_TCP_Slave [Device: Ethernet: Modbus_T...

BL67_GW_EN_Station1

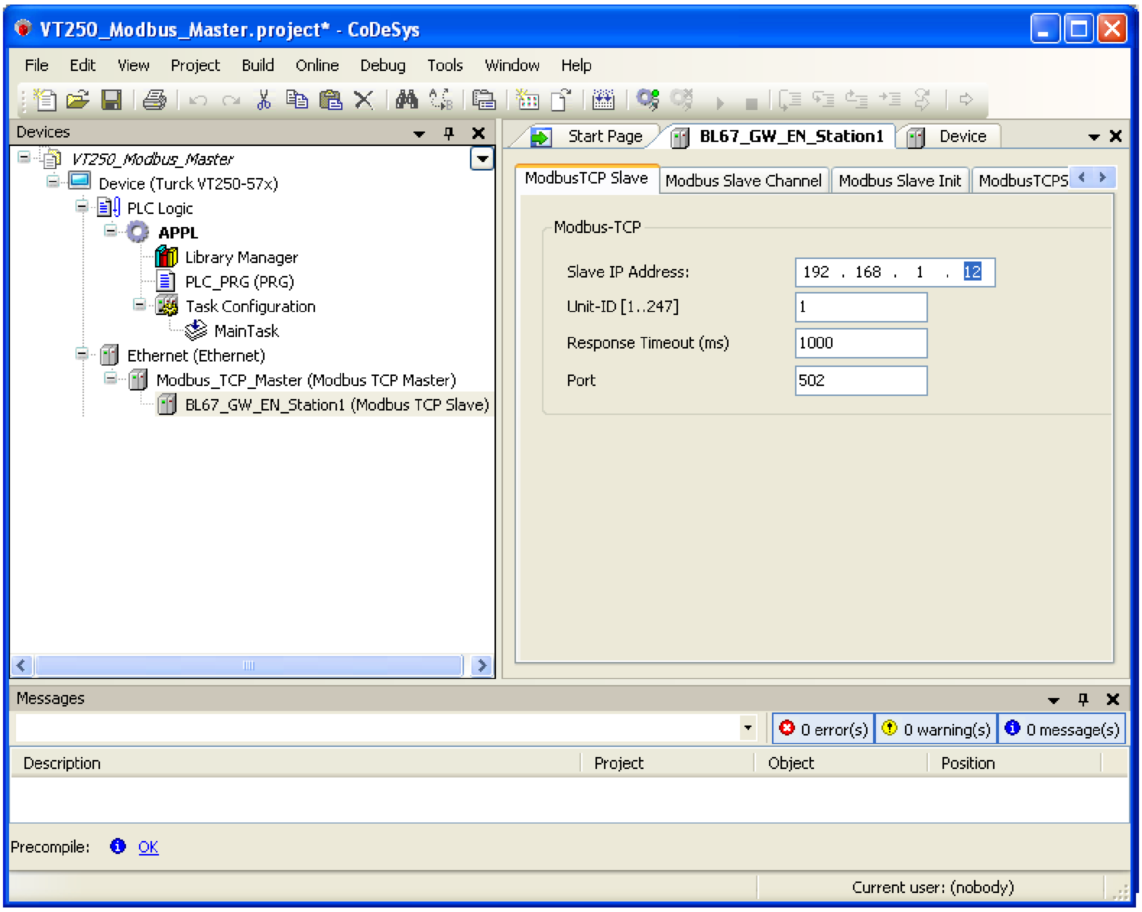
Full name: Modbus_TCP_Slave [Device: Ethernet: Modbus_TCP_Master

Object type: Device

Open with: Device Editor

2 Again, a double-click onto the slave in the project tree opens the respective editors. Enter the node's IP address in the register-tab "Modbus TCP Slave" (in this example: address **192.168.1.12**). All other settings can be kept.

Figure 12: Setting the IP address at the slave



VT250_Modbus_Master.project* - CoDeSys

Devices:
- VT250_Modbus_Master
  - Device (Turck VT250-57x)
    - PLC Logic
      - APPL
        - Library Manager
        - PLC_PRG (PRG)
        - Task Configuration
          - MainTask
    - Ethernet (Ethernet)
      - Modbus_TCP_Master (Modbus TCP Master)
        - BL67_GW_EN_Station1 (Modbus TCP Slave)

Tabs: ModbusTCP Slave | Modbus Slave Channel | Modbus Slave Init | ModbusTCPS

Modbus-TCP

Slave IP Address:	192 . 168 . 1 . 12
Unit-ID [1..247]	1
Response Timeout (ms)	1000
Port	502

Messages: 0 error(s) 0 warning(s) 0 message(s)

Precompile: OK

Current user: (nobody)

0

Programming (example program)

The programming is done in the program PLC_PRG . This example is programmed in ST as defined under "Creating a new project".

Small example program

1 The counter counts

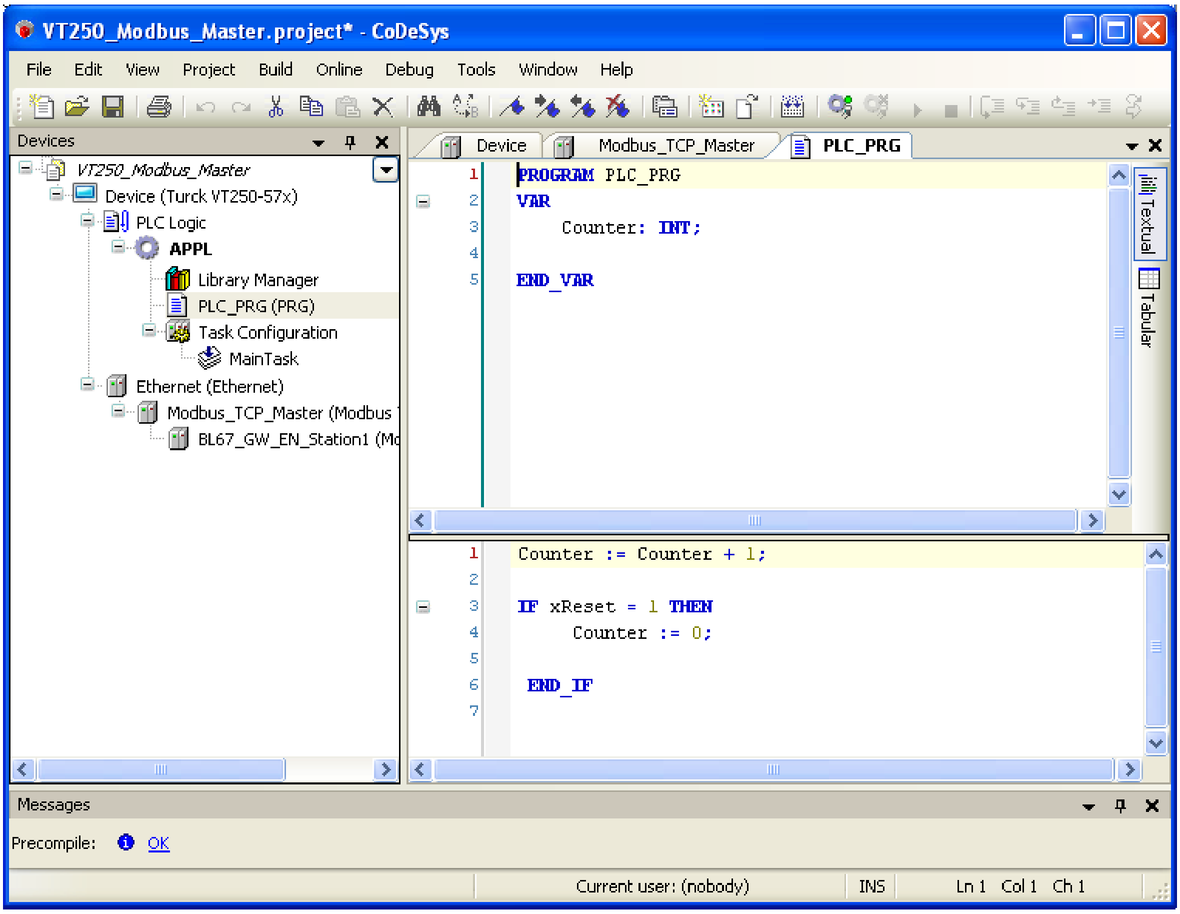
2 Counter-reset via setting the variable "xReset" (BOOL) to "1".
"xReset" has been defined in the global variables (see also page page 1-14)

3 The analog value of the sensor at channel 1of the analog input module (slot 2 of the BL67 station) is read out.

Note

The status of process values is only shown in the process image if a program refers to them or if the function "Always update variables" in the "MobusTCPSlave I/O Mapping" (see Reading out the process data, page 1-28) is enabled.

Figure 13: Example program



```
PROGRAM PLC_PRG
VAR
    Counter: INT;

END_VAR
```

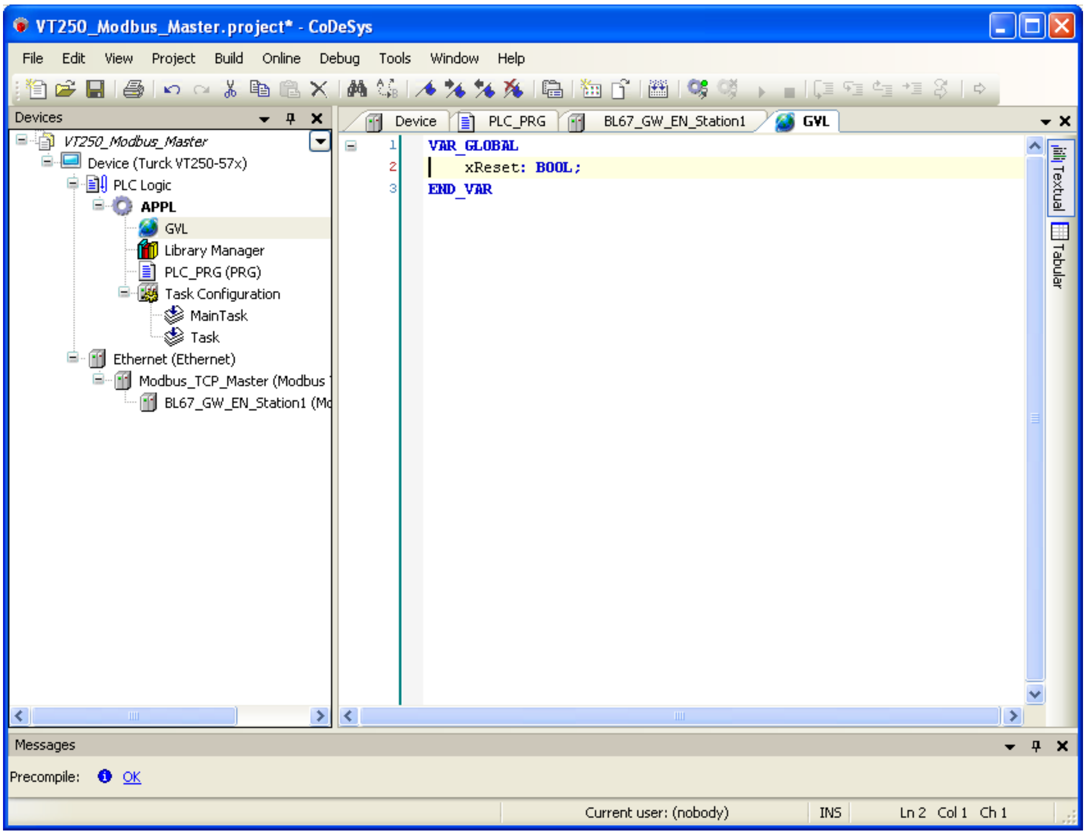
```
Counter := Counter + 1;

IF xReset = 1 THEN
    Counter := 0;

 END_IF

```

Precompile: OK

CoDeSys: global variables

Global variables are defined either in the Global Variable List (see page 1-15) or directly in the I/O Mappings of the single I/O modules.

Figure 14: Example for the definition of a global variable



```
VAR_GLOBAL
    xReset: BOOL;
END_VAR
```

Precompile: OK

Global variable list

The creation of a "Global Variable List" is possible, too:
right-click to "APPL" → Add object → "Global Variable List".

Define the global variables. The global variables are also automatically exported when building the project if they have been chosen for export in the symbol configuration. (see also Figure 26: Symbol configuration, select variables, page 1-26).

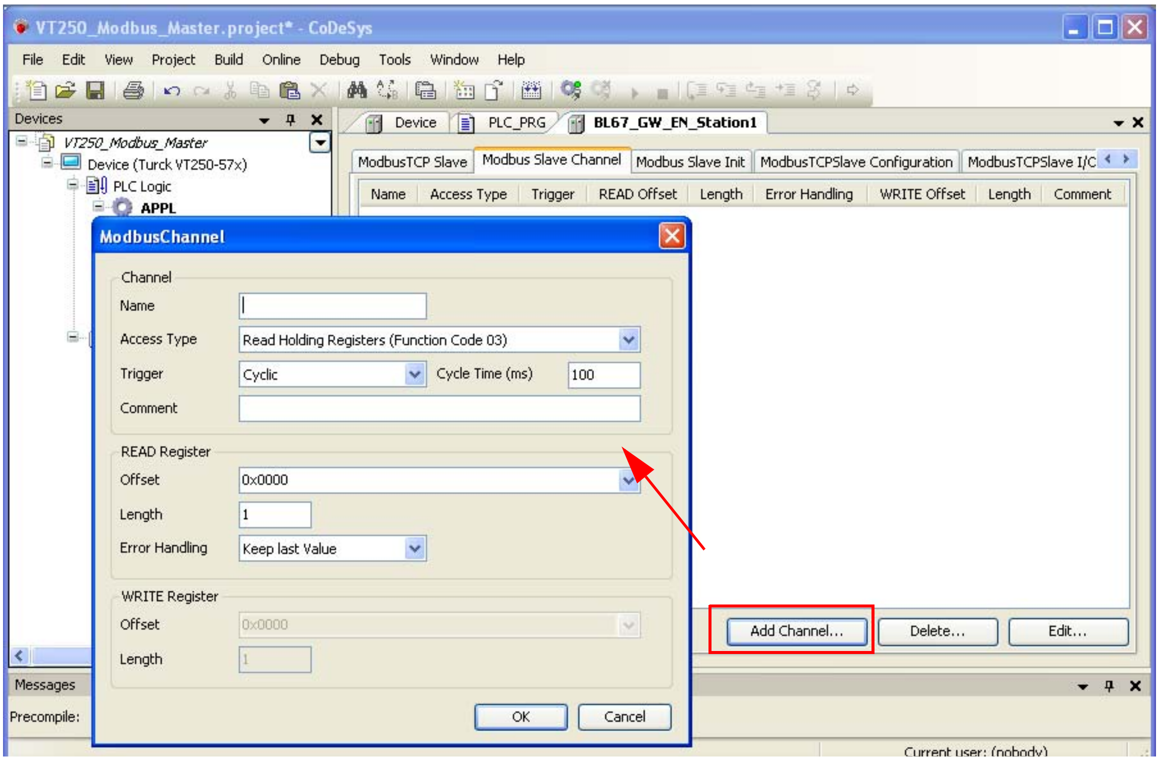
Modbus channels

The communication between Modbus TCP master and Modbus slaves is realized through defined Modbus channels.

These channels are set in the register-tab "Modbus Slave Channel" using the "Add Channel..." button.

Correspondent to the set channels, the slave's process data can then be monitored under "ModbusTCPSlave I/O Mapping" (see Reading out the process data, page 1-28).

Figure 15: Setting the Modbus channels, examples



ModbusChannel

Channel
Name:
Access Type: Read Holding Registers (Function Code 03)
Trigger: Cyclic
Cycle Time (ms): 100
Comment:

READ Register
Offset: 0x0000
Length: 1
Error Handling: Keep last Value

WRITE Register
Offset: 0x0000
Length: 1

The Modbus communication channels are defined by:

- "Access Type":
  Modbus function code, which defines the access method (bit- or word-wise, read or write).
- "READ Register" or "WRITE Register" →"Offset":
  Specification of the start address for the Modbus Slave's register that has to be read or written. These specifications are part of the slave's Modbus documentation!

Modbus data mapping

The mapping for the input and output data of a BLxx-Modbus station depends on it's configuration.

The TURCK-Software "I/O-ASSISTANT" offers the feature of creating a Modbus report for each BLxx-Modbus station, which shows the mapping for the respective station.

Example station (see also page 1-2):

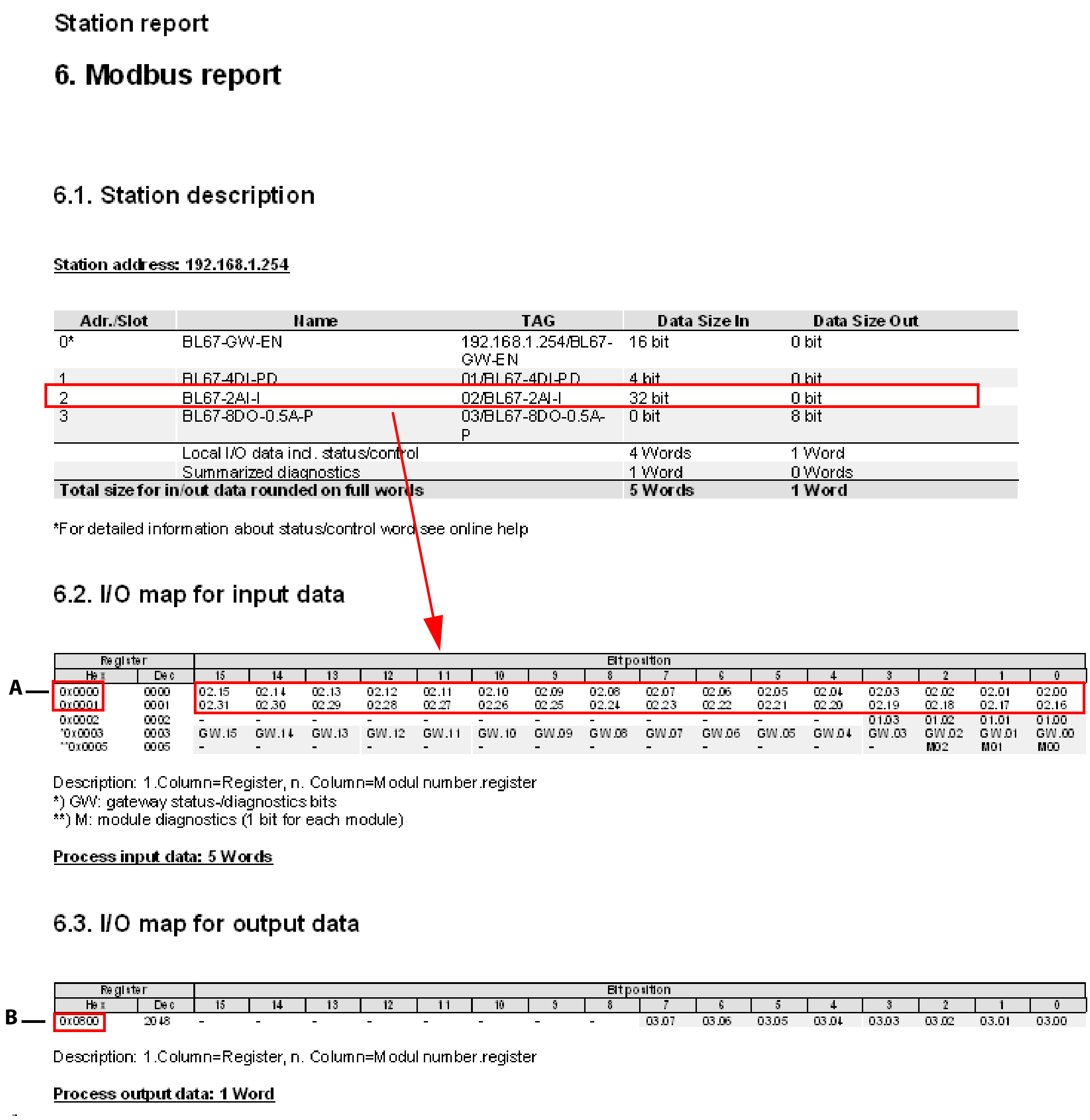
BL67-GW-EN (IP address 192.168.1.12) with:

- Slot 1: BL67-4DI-PD
- Slot 2: BL67-2AI-I with analog sensor Bi5-M18-Li2
- Slot 3: BL67-8DO-0.5A-P

Modbus mapping (I/O-ASSISTANT)

Figure 16: Modbus report in the I/O-ASSISTANT

A Register, read, analog value, slot 2
B Register write, 8DO, %QW0, slot 3



Station report

6. Modbus report

6.1. Station description

Station address: 192.168.1.254

Adr./Slot	Name	TAG	Data Size In	Data Size Out
0*	BL67-GW-EN	192.168.1.254/BL67-GW-EN	16 bit	0 bit
1	BL67-4DI-PD	01/BL67-4DI-PD	4 bit	0 bit
2	BL67-2AI-I	02/BL67-2AI-I	32 bit	0 bit
3	BL67-8DO-0.5A-P	03/BL67-8DO-0.5A-P	0 bit	8 bit
	Local I/O data incl. status/control		4 Words	1 Word
	Summarized diagnostics		1 Word	0 Words
Total size for in/out data rounded on full words			**5 Words**	**1 Word**

*For detailed information about status/control word see online help

6.2. I/O map for input data

Register		Bit position															
Hex	Dec	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	0000	02.15	02.14	02.13	02.12	02.11	02.10	02.09	02.08	02.07	02.06	02.05	02.04	02.03	02.02	02.01	02.00
0x0001	0001	02.31	02.30	02.29	02.28	02.27	02.26	02.25	02.24	02.23	02.22	02.21	02.20	02.19	02.18	02.17	02.16
0x0002	0002	-	-	-	-	-	-	-	-	-	-	-	-	01.03	01.02	01.01	01.00
*0x0003	0003	GW.15	GW.14	GW.13	GW.12	GW.11	GW.10	GW.09	GW.08	GW.07	GW.06	GW.05	GW.04	GW.03	GW.02	GW.01	GW.00
**0x0005	0005	-	-	-	-	-	-	-	-	-	-	-	-	-	M02	M01	M00

Description: 1.Column=Register, n. Column=Modul number.register
*) GW: gateway status-/diagnostics bits
**) M: module diagnostics (1 bit for each module)

Process input data: 5 Words

6.3. I/O map for output data

Register		Bit position															
Hex	Dec	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0800	2048	-	-	-	-	-	-	-	-	03.07	03.06	03.05	03.04	03.03	03.02	03.01	03.00

Description: 1.Column=Register, n. Column=Modul number.register

Process output data: 1 Word

Note

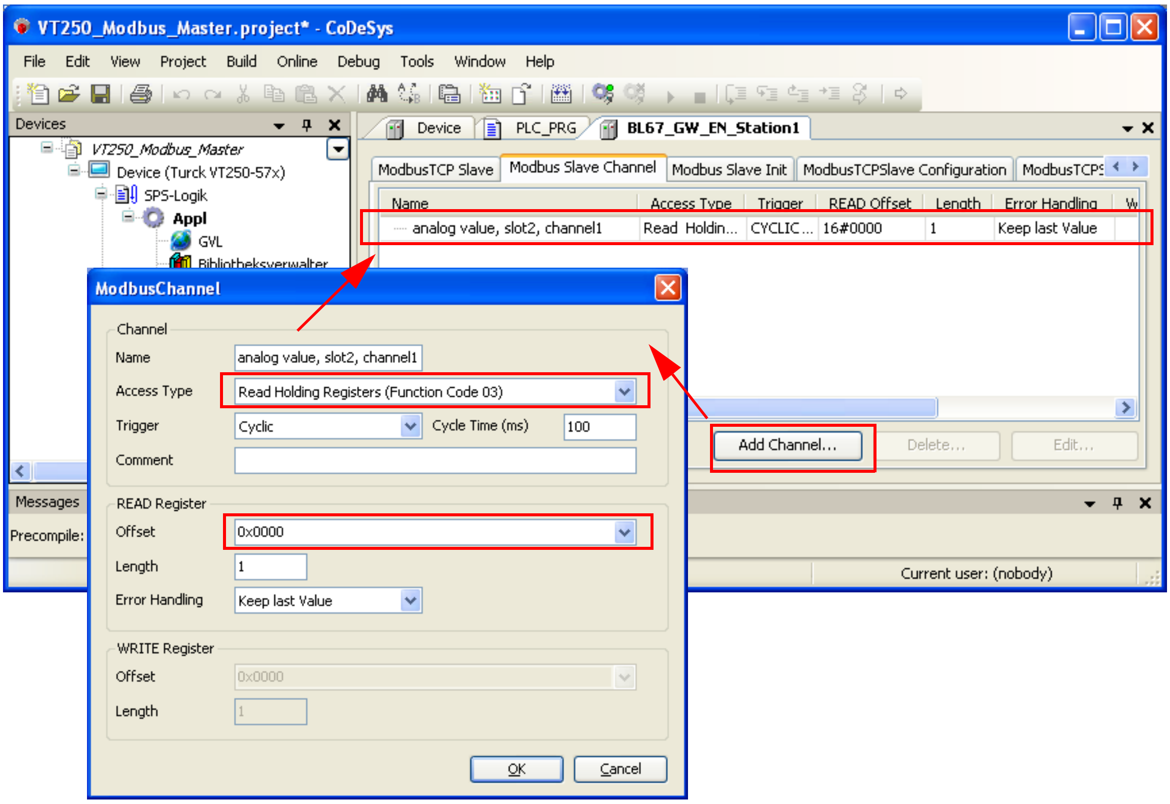
For more detailed information about the Modbus registers of the BL67-GW-EN, please read the respective manual (D300815, chapter 4).

Setting the Modbus channels, (examples)

1 Read: Analog value, sensor at slot 2 (see also page 1-2).

- Access Type:
  Read Holding Registers (function code **03**)
- Read Register, Offset:
  0x0000 (see above Modbus report in the I/O-ASSISTANT, page 1-16):
  The *packed* process input data of the modules in a BL67-station can be found in registers 0x0000 to 0x01FF, whereas the data of the intelligent modules (analog and technology modules) are mapped first, followed by the data of the digital modules in a station.
  As the BL67-2AI-I at slot 2 is the first analog input module in the example station, register 0x0000 has to be read.

Figure 17: Modbus channel, read analog value, FC03



Name	Access Type	Trigger	READ Offset	Length	Error Handling	W
analog value, slot2, channel1	Read Holdin...	CYCLIC ...	16#0000	1	Keep last Value	

ModbusChannel

Channel
Name: analog value, slot2, channel1
Access Type: Read Holding Registers (Function Code 03)
Trigger: Cyclic
Cycle Time (ms): 100
Comment:

READ Register
Offset: 0x0000
Length: 1
Error Handling: Keep last Value

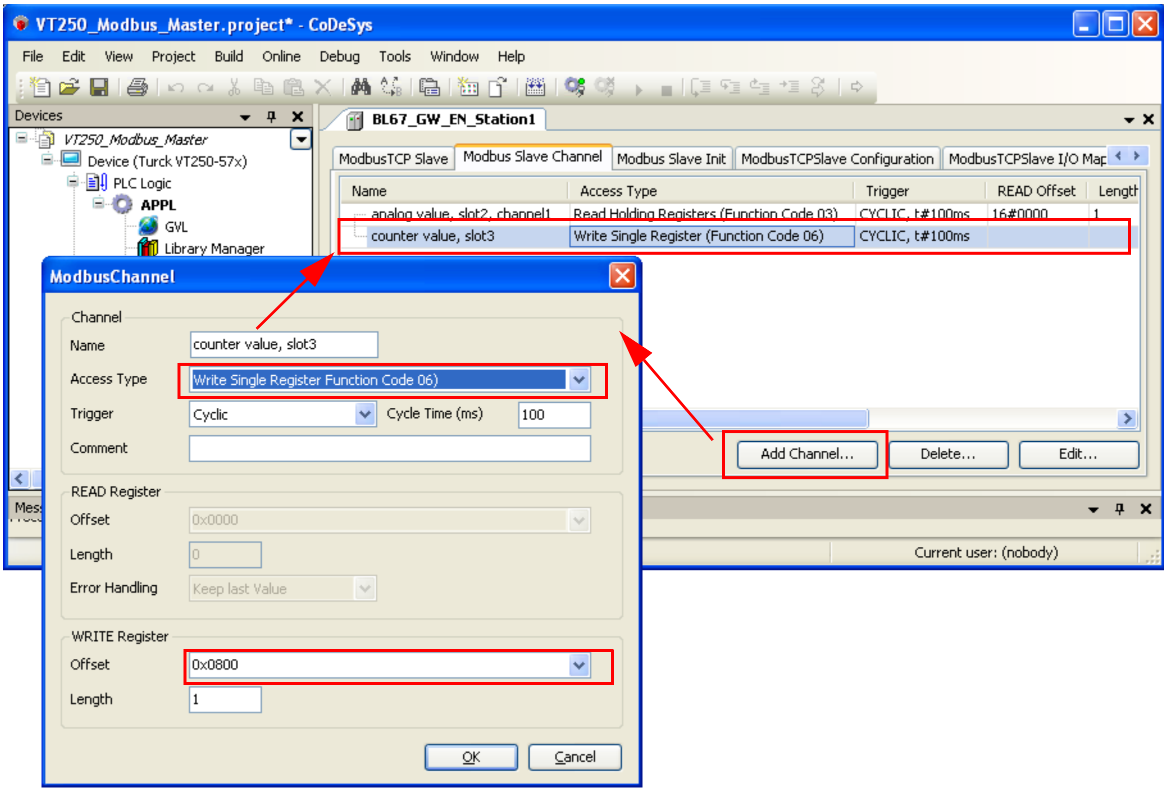
WRITE Register
Offset: 0x0000
Length: 1

2 Writing of **%QW0** and mapping of the counter value (VAR "Counter", see PLC_PRG, page 1-13) to the output byte of the digital output module (BL67-8DO-0.5A-P, Slot 3, %QW0).

2.1 Write: %QW0

- Access Type:
  Write Single Registers (function code **06**)
- Write Register, Offset:
  0x0800 (see above Modbus report in the I/O-ASSISTANT, page 1-16):
  The *packed* process output data of the modules in a BL67-station can be found in registers 0x0800 to 0x09FF, whereas the data of the intelligent modules (analog and technology modules) are mapped first, followed by the data of the digital modules in a station.
  As the BL67-8DO-0,5A-P at slot 3 is the first and only output module in the example station, register 0x0800 has to be written.

Figure 18: Modbus channel, write counter value, FC03



Name	Access Type	Trigger	READ Offset	Length
analog value, slot2, channel1	Read Holding Registers (Function Code 03)	CYCLIC, t#100ms	16#0000	1
counter value, slot3	Write Single Register (Function Code 06)	CYCLIC, t#100ms		

Channel
Name: counter value, slot3
Access Type: Write Single Register Function Code 06)
Trigger: Cyclic
Cycle Time (ms): 100
Comment:

READ Register
Offset: 0x0000
Length: 0
Error Handling: Keep last Value

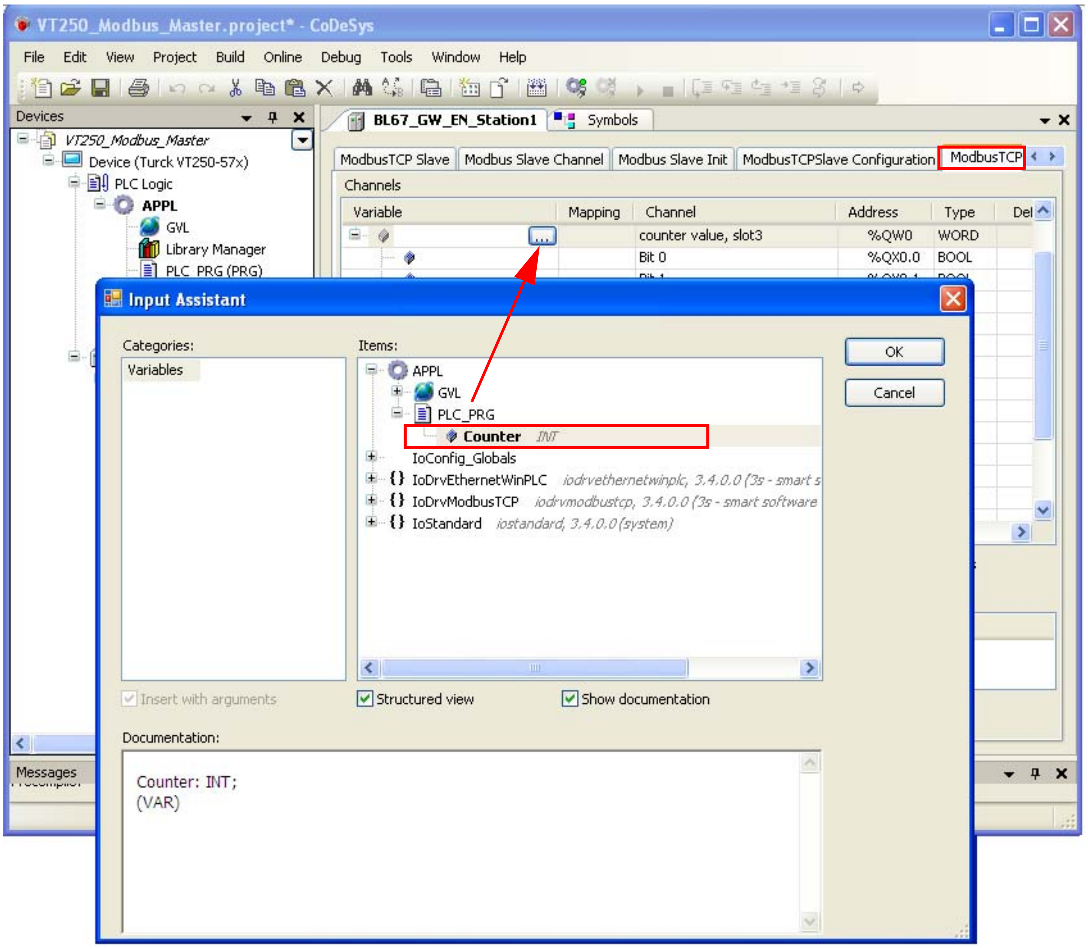
WRITE Register
Offset: 0x0800
Length: 1

2.2 Mapping: counter value to %QW0

– The mapping of the counter value (VAR "Counter") to the module's output register is done in the "Modbus TCP Slave I/O Mapping".
Double click the field "variable" in the respective line. Use the "..."-button to open the dialog box "Input Assistant".

– Here, select the variable to be mapped. "Counter" can be found under "PLC_PRG", as it has been defined there see Programming (example program).

Figure 19: Mapping of the counter value to %QW0



Variable	Mapping	Channel	Address	Type	De
...		counter value, slot3	%QW0	WORD	
		Bit 0	%QX0.0	BOOL	

```
APPL
  GVL
  PLC_PRG
    Counter   INT
IoConfig_Globals
{} IoDrvEthernetWinPLC   iodrvethernetwinplc, 3.4.0.0 (3s - smart s
{} IoDrvModbusTCP   iodrvmodbustcp, 3.4.0.0 (3s - smart software
{} IoStandard   iostandard, 3.4.0.0 (system)
```

☑ Insert with arguments ☑ Structured view ☑ Show documentation

```
Counter: INT;
(VAR)
```

– Confirm with "OK". The counter value is now mirrored to %QW0 of the output module at slot 3 in the example station and displayed.

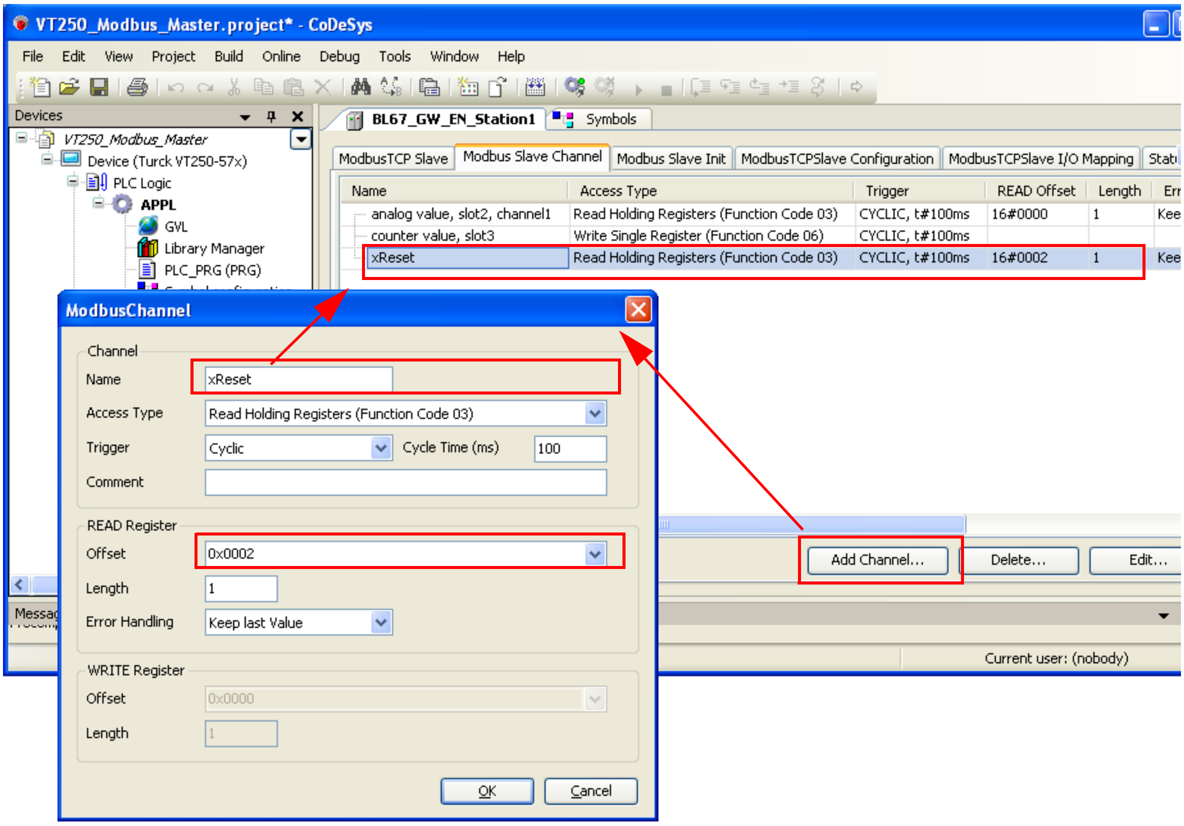
3 Read: **Bit 0** at BL67-4DI-PD → resetting the counter (with "xReset" = 1)

3.1 Read: %IW2

- Access Type:
  Read Holding Registers (function code **03**)
- Read Register, Offset:
  0x0002 (see above Modbus report in the I/O-ASSISTANT, page 1-16):
  The *packed* process input data of the modules in a BL67-station can be found in registers 0x0000 to 0x01FF, whereas the data of the intelligent modules (analog and technology modules) are mapped first, followed by the data of the digital modules in a station.

  As the BL67-4DI-PD at slot 1 is the first **digital** input module in the example station, it's input data are mapped after those of the analog module (slot 2) and are thus read from register 0x0002 (%IW2).

Figure 20: Modbus channel, read "xReset", FC03



Name	Access Type	Trigger	READ Offset	Length	Err
analog value, slot2, channel1	Read Holding Registers (Function Code 03)	CYCLIC, t#100ms	16#0000	1	Kee
counter value, slot3	Write Single Register (Function Code 06)	CYCLIC, t#100ms			
xReset	Read Holding Registers (Function Code 03)	CYCLIC, t#100ms	16#0002	1	Kee

ModbusChannel

Channel
Name: xReset
Access Type: Read Holding Registers (Function Code 03)
Trigger: Cyclic
Cycle Time (ms): 100
Comment:

READ Register
Offset: 0x0002
Length: 1
Error Handling: Keep last Value

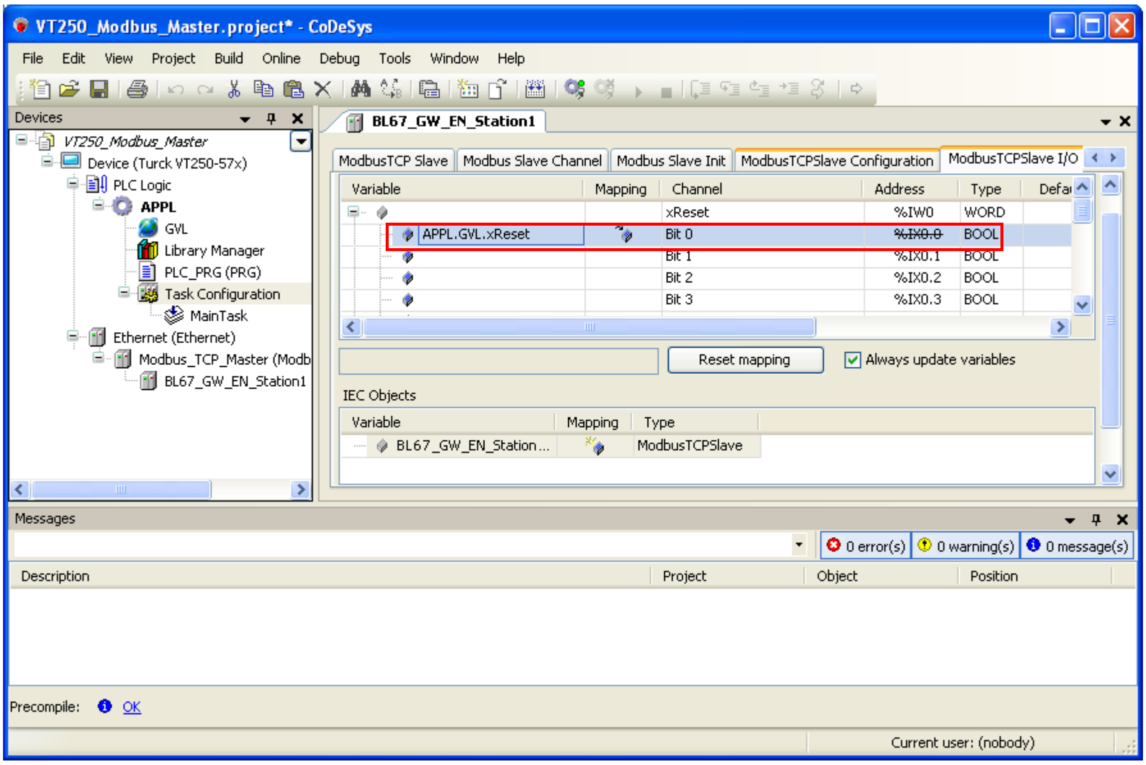
WRITE Register
Offset: 0x0000
Length: 1

OK Cancel

3.2 Mapping: "xReset" (global variable) to %IX4.0 in %IW2

- "xReset" is mapped to the first bit in %IW2 of BL67-4DI-PD. This is done in the "ModbusTCPSlave I/O Mapping".
- Double click the field "variable" in the respective line. Use the "..."-button to open the dialog box "Input Assistant".
- Here, select the variable to be mapped. "Counter" can be found under in the global variables, as it has been defined there, see CoDeSys: global variables.

Figure 21: Mapping of "xReset" to bit %IX4.0



Variable	Mapping	Channel	Address	Type	Defau
		xReset	%IW0	WORD	
APPL.GVL.xReset		Bit 0	~~%IX0.0~~	BOOL	
		Bit 1	%IX0.1	BOOL	
		Bit 2	%IX0.2	BOOL	
		Bit 3	%IX0.3	BOOL	

Variable	Mapping	Type
BL67_GW_EN_Station...		ModbusTCPSlave

Description	Project	Object	Position

Precompile: OK

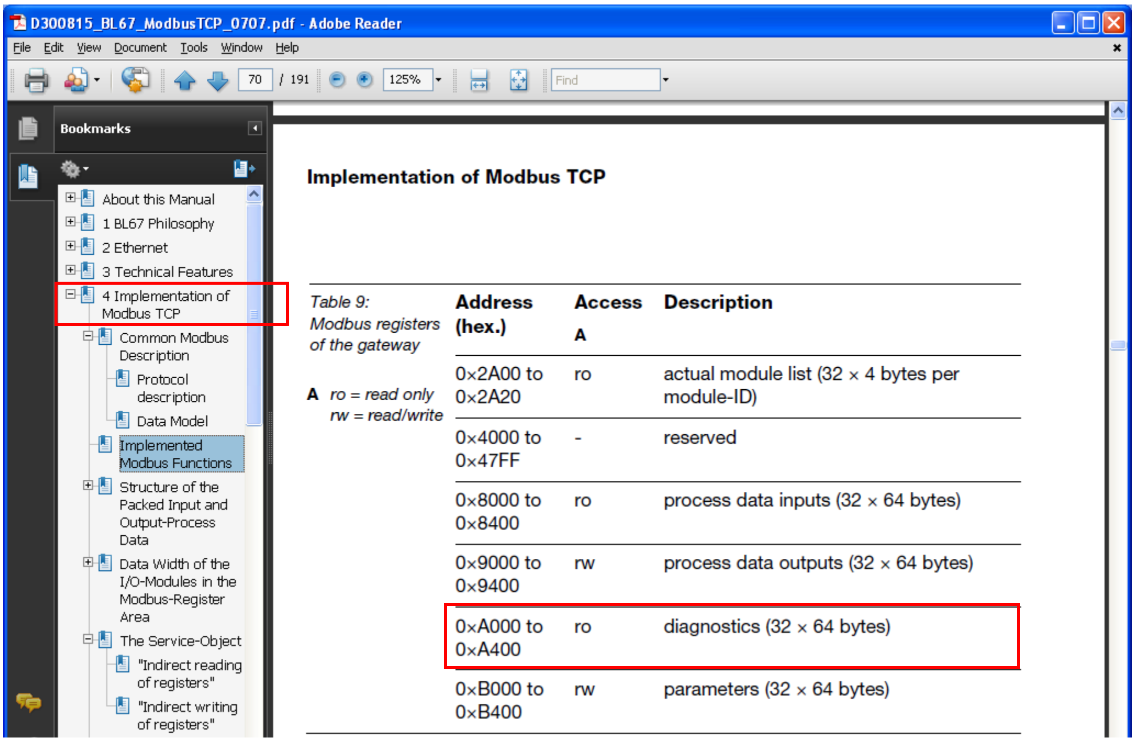
Current user: (nobody)

- Confirm with "OK". A "1" at bit %IX4.0 will now reset the counter to zero.

4 Read: **Diagnosis** of the analog module at **slot2**

- Access Type:
  Read Holding Registers (function code **03**)
- Read Register, Offset:
  0xA020 (see D300815, chapter 4, Modbus registers)
  The diagnosis information of the I/O modules can either be read as a *group diagnosis* starting with register 0x1018 or as *individual diagnosis* starting with register 0xA000 (64 byte = 32 register per module).

Figure 22: D300815, excerpt Modbus registers, diagnosis


D300815_BL67_ModbusTCP_0707.pdf - Adobe Reader
File Edit View Document Tools Window Help
70 / 191
125%
Find
Bookmarks
About this Manual
1 BL67 Philosophy
2 Ethernet
3 Technical Features
4 Implementation of Modbus TCP
Common Modbus Description
Protocol description
Data Model
Implemented Modbus Functions
Structure of the Packed Input and Output-Process Data
Data Width of the I/O-Modules in the Modbus-Register Area
The Service-Object
"Indirect reading of registers"
"Indirect writing of registers"

Implementation of Modbus TCP

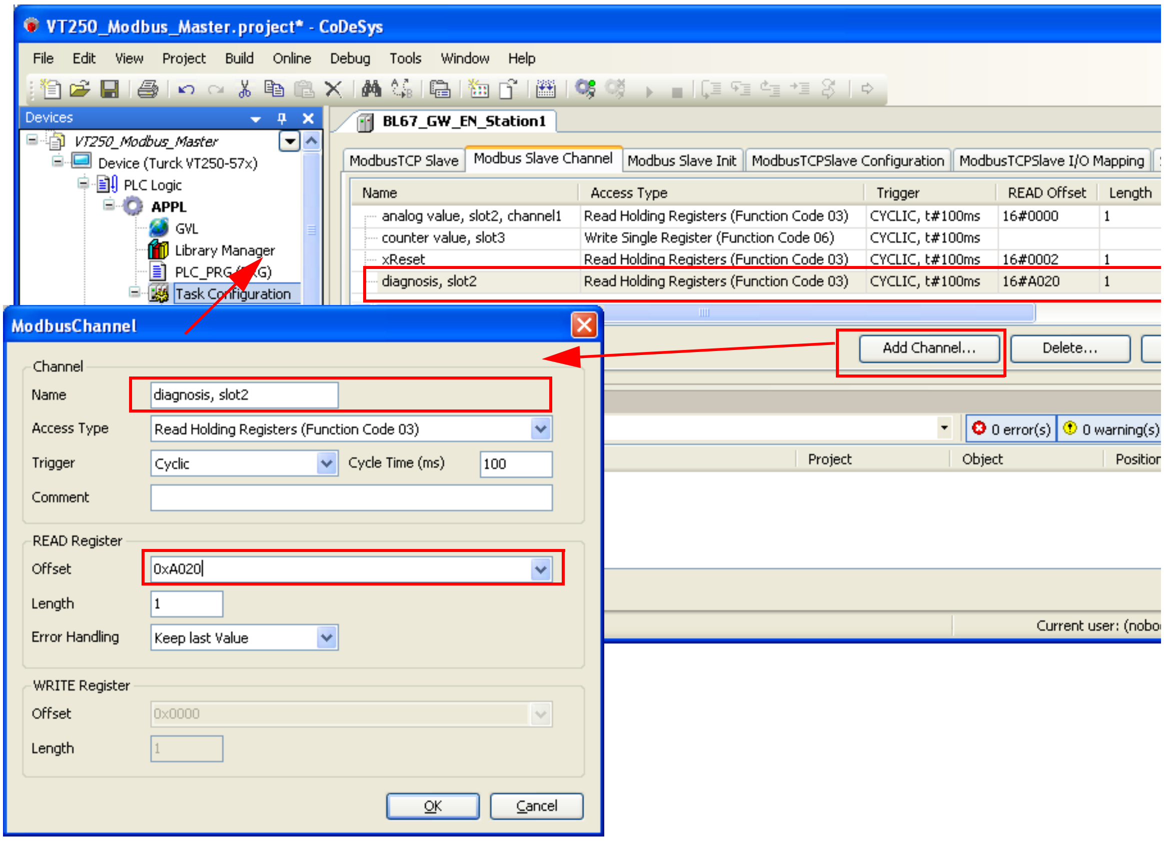
Table 9: Modbus registers of the gateway

Address (hex.)	Access A	Description
0×2A00 to 0×2A20	ro	actual module list (32 × 4 bytes per module-ID)
0×4000 to 0×47FF	-	reserved
0×8000 to 0×8400	ro	process data inputs (32 × 64 bytes)
0×9000 to 0×9400	rw	process data outputs (32 × 64 bytes)
0×A000 to 0×A400	ro	diagnostics (32 × 64 bytes)
0×B000 to 0×B400	rw	parameters (32 × 64 bytes)

A *ro = read only*
rw = read/write

– The diagnostic data of the analog module at slot 2 in the station will thus be read from register 0xA020.

Figure 23: Setting the Modbus channel for reading the diagnostic data



Name	Access Type	Trigger	READ Offset	Length
analog value, slot2, channel1	Read Holding Registers (Function Code 03)	CYCLIC, t#100ms	16#0000	1
counter value, slot3	Write Single Register (Function Code 06)	CYCLIC, t#100ms		
xReset	Read Holding Registers (Function Code 03)	CYCLIC, t#100ms	16#0002	1
diagnosis, slot2	Read Holding Registers (Function Code 03)	CYCLIC, t#100ms	16#A020	1

Channel
Name: diagnosis, slot2
Access Type: Read Holding Registers (Function Code 03)
Trigger: Cyclic
Cycle Time (ms): 100
Comment:

READ Register
Offset: 0xA020
Length: 1
Error Handling: Keep last Value

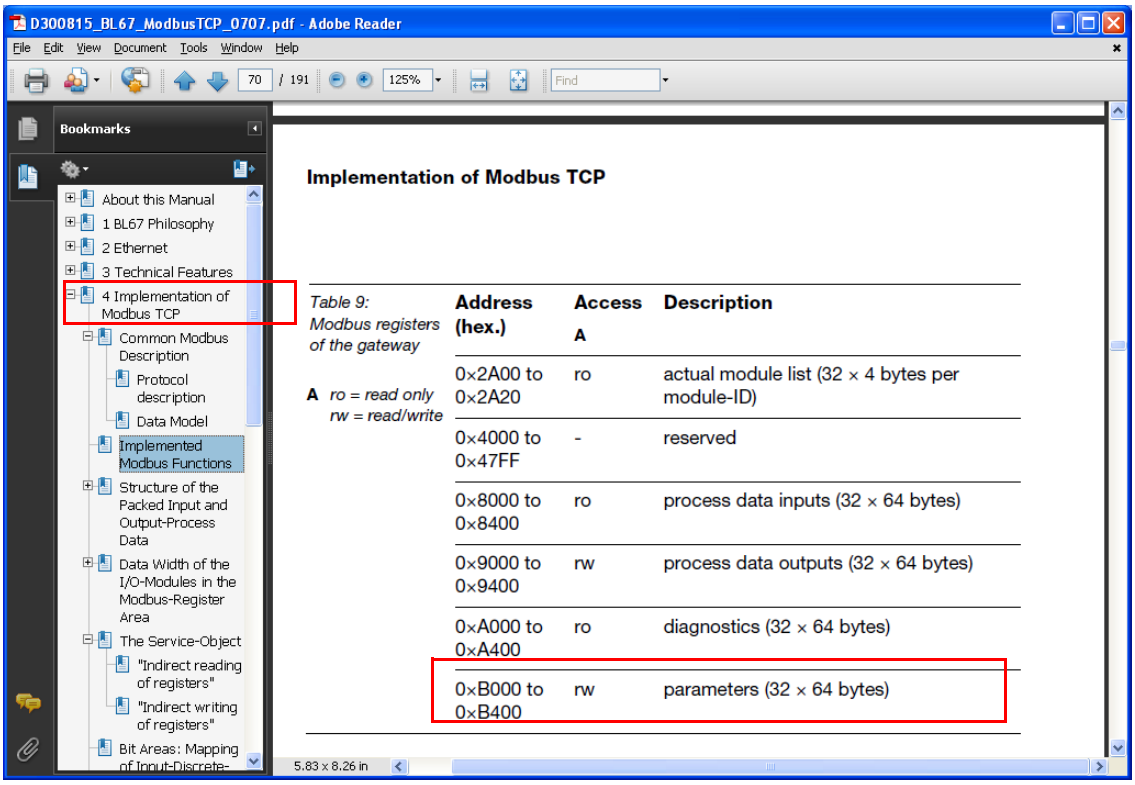
WRITE Register
Offset: 0x0000
Length: 1

5 Write: **parameters** of the analog module at **slot2**
Intention→ deactivating the channel diagnosis
Writing parameters is normally done once during the program start and is thus not built as a "normal" Modbus channel under "ModbusSlaveChannel", but as an Initialization channel under "**Modbus Slave Init**" (see Figure 25: Setting the initialization channel for the parameterization).

– Access Type:
Write Single Register (function code **06**)

– Write Register, Offset:
0xB020 (see D300815, chapter 4, Modbus registers)
The parameters for the I/O modules are written starting with register 0xB000 (64 byte = 32 registers per module).

Figure 24: D300815, Modbus registers, parameters


D300815_BL67_ModbusTCP_0707.pdf - Adobe Reader
File Edit View Document Tools Window Help
70 / 191
125%
Find
Bookmarks
About this Manual
1 BL67 Philosophy
2 Ethernet
3 Technical Features
4 Implementation of Modbus TCP
Common Modbus Description
Protocol description
Data Model
Implemented Modbus Functions
Structure of the Packed Input and Output-Process Data
Data Width of the I/O-Modules in the Modbus-Register Area
The Service-Object
"Indirect reading of registers"
"Indirect writing of registers"
Bit Areas: Mapping of Input-Discrete-
5.83 x 8.26 in

Implementation of Modbus TCP

Table 9: Modbus registers of the gateway

Address (hex.)	Access A	Description
0x2A00 to 0x2A20	ro	actual module list (32 × 4 bytes per module-ID)
0x4000 to 0x47FF	-	reserved
0x8000 to 0x8400	ro	process data inputs (32 × 64 bytes)
0x9000 to 0x9400	rw	process data outputs (32 × 64 bytes)
0xA000 to 0xA400	ro	diagnostics (32 × 64 bytes)
0xB000 to 0xB400	rw	parameters (32 × 64 bytes)

A *ro = read only*
rw = read/write

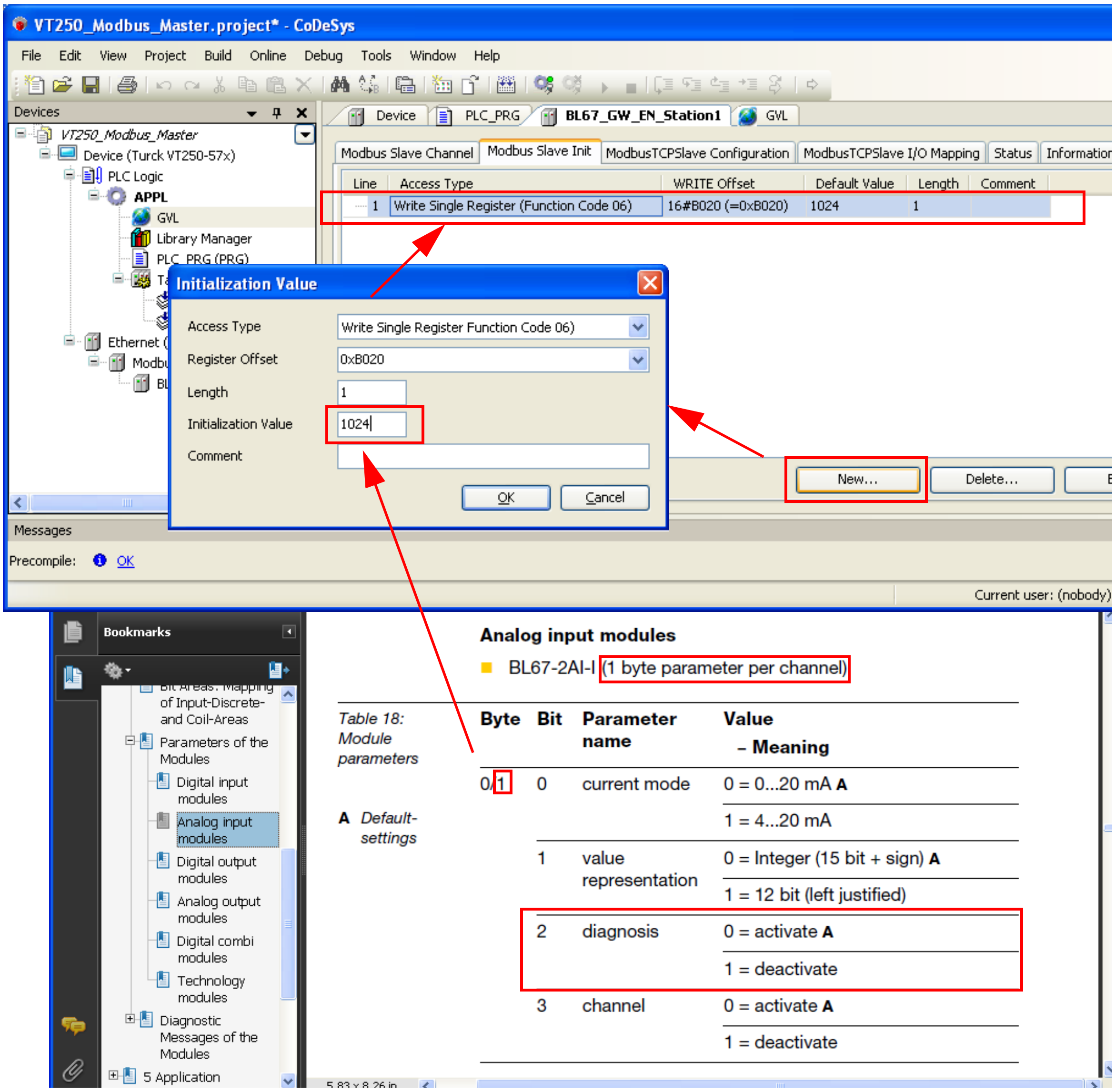
Parameterization of the module

The parameters of the module at slot 2 in the station are located in registers 0xB020 ff..

Each channel of the module has one byte parameter data. The parameter data for channel 1 are located in byte 1 of the parameter word (register).

In this case, the parameterization should be a deactivation of the diagnostic messages for channel 2 of the analog module (byte1, bit 2 = 1, see below).

A 2^{10} = **1024** will be written, which results from module's the parameter byte assignment (see below).



Line	Access Type	WRITE Offset	Default Value	Length	Comment
1	Write Single Register (Function Code 06)	16#B020 (=0xB020)	1024	1	

Initialization Value

- Access Type: Write Single Register Function Code 06)
- Register Offset: 0xB020
- Length: 1
- Initialization Value: 1024
- Comment:

OK Cancel

Analog input modules

- BL67-2AI-I (1 byte parameter per channel)

Table 18: Module parameters

A Default-settings

Byte	Bit	Parameter name	Value – Meaning
0/1	0	current mode	0 = 0...20 mA **A**
			1 = 4...20 mA
	1	value representation	0 = Integer (15 bit + sign) **A**
			1 = 12 bit (left justified)
	2	diagnosis	0 = activate **A**
			1 = deactivate
	3	channel	0 = activate **A**
			1 = deactivate

Figure 25: Setting the initialization channel for the parameterization

Table 1: *Parameter register of the analog module*

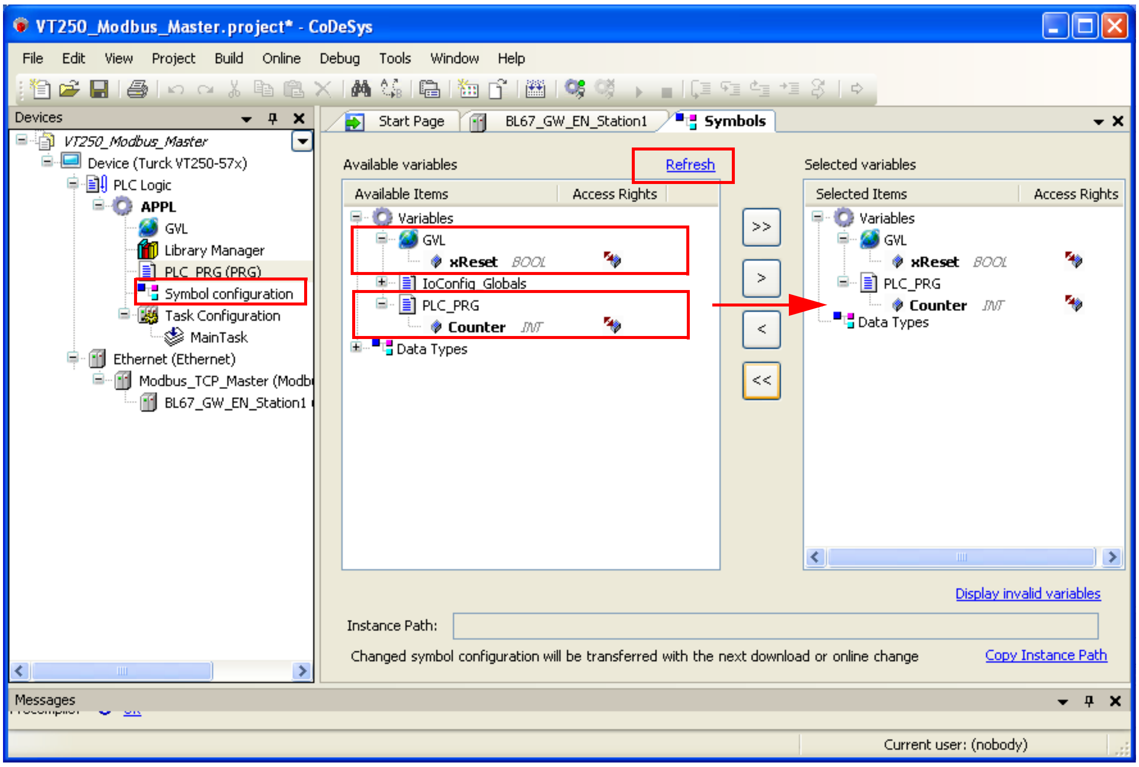
	Parameter byte 1 (channel 1)								Parameter byte 0 (channel 0)							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	$\mathbf{2^{10}}$	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Export of variables

The variables defined in the program have to be exported, for example in order to be used within an visualization.

1. Therefore a symbol configuration has to be added to the project. This is done via right-click on "APPL" and "Add object".
2. The symbol configuration shows all available variables (left side of the window) and allows the definition of all variables to be exported (right side of the window).
3. Export **only** the variables which are necessary for your visualization and add them to the window "Selected variable" by using the arrow keys.
   If any variable is missing, please "Refresh" the variable list.

Figure 26: Symbol configuration, select variables



Changed symbol configuration will be transferred with the next download or online change

4. This symbol configuration is automatically stored to the project folder as an *.xml-file when downloading the project.

Note

In case of an offline project without download, the simulation of the project ("Online → Simulation") is necessary in order to build a *.xml-file containing the exported variables.

Loading the project into the VT250-57×

1 Building the project:
Building the project is done via "Build → Rebuild".

2 Log-in:
The connection to the device is established via "Online→ Login". The application is downloaded.

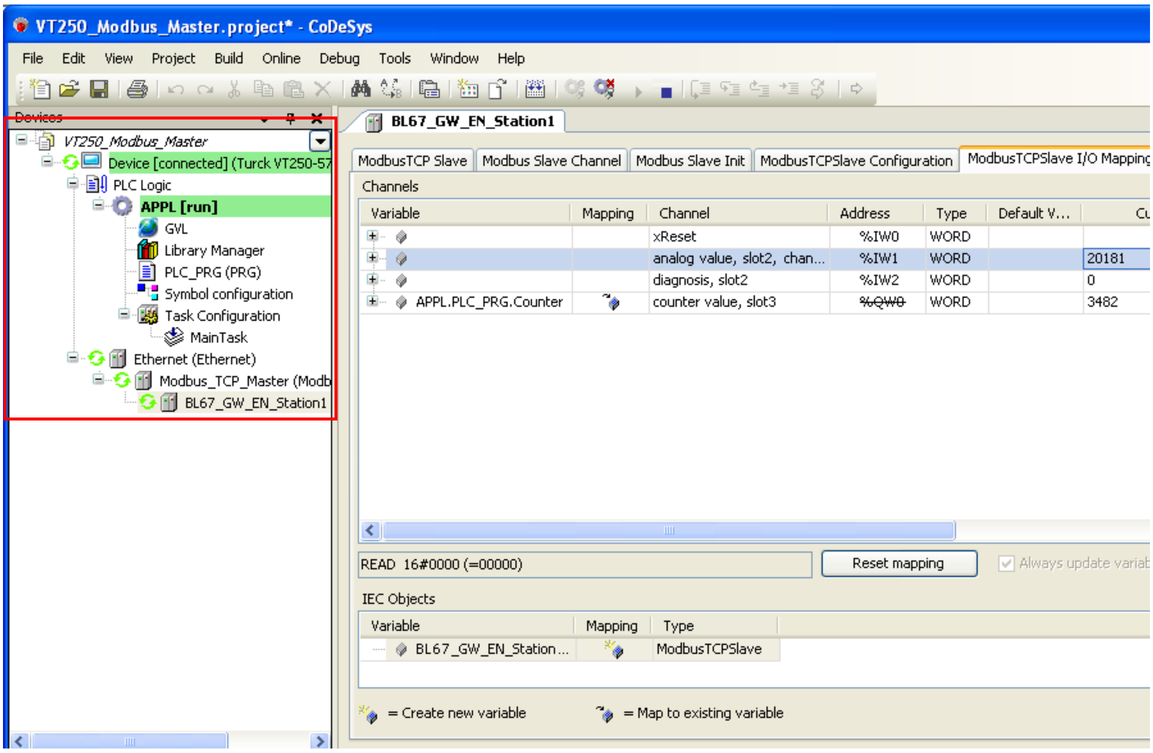
3 Create boot application:
A boot project is created via "Online → Create boot application" and is then directly written into the device.

Note
A boot application can only be created in online-mode.

4 Starting the program:
Start the program using "Online → Start".

5 The program is running (see PLC_PRG in project tree).

Figure 27: Program running, network OK



Variable	Mapping	Channel	Address	Type	Default V...	Cu
		xReset	%IW0	WORD		
		analog value, slot2, chan...	%IW1	WORD		20181
		diagnosis, slot2	%IW2	WORD		0
APPL.PLC_PRG.Counter		counter value, slot3	~~%QW0~~	WORD		3482

READ 16#0000 (=00000)

Variable	Mapping	Type
BL67_GW_EN_Station...		ModbusTCPSlave

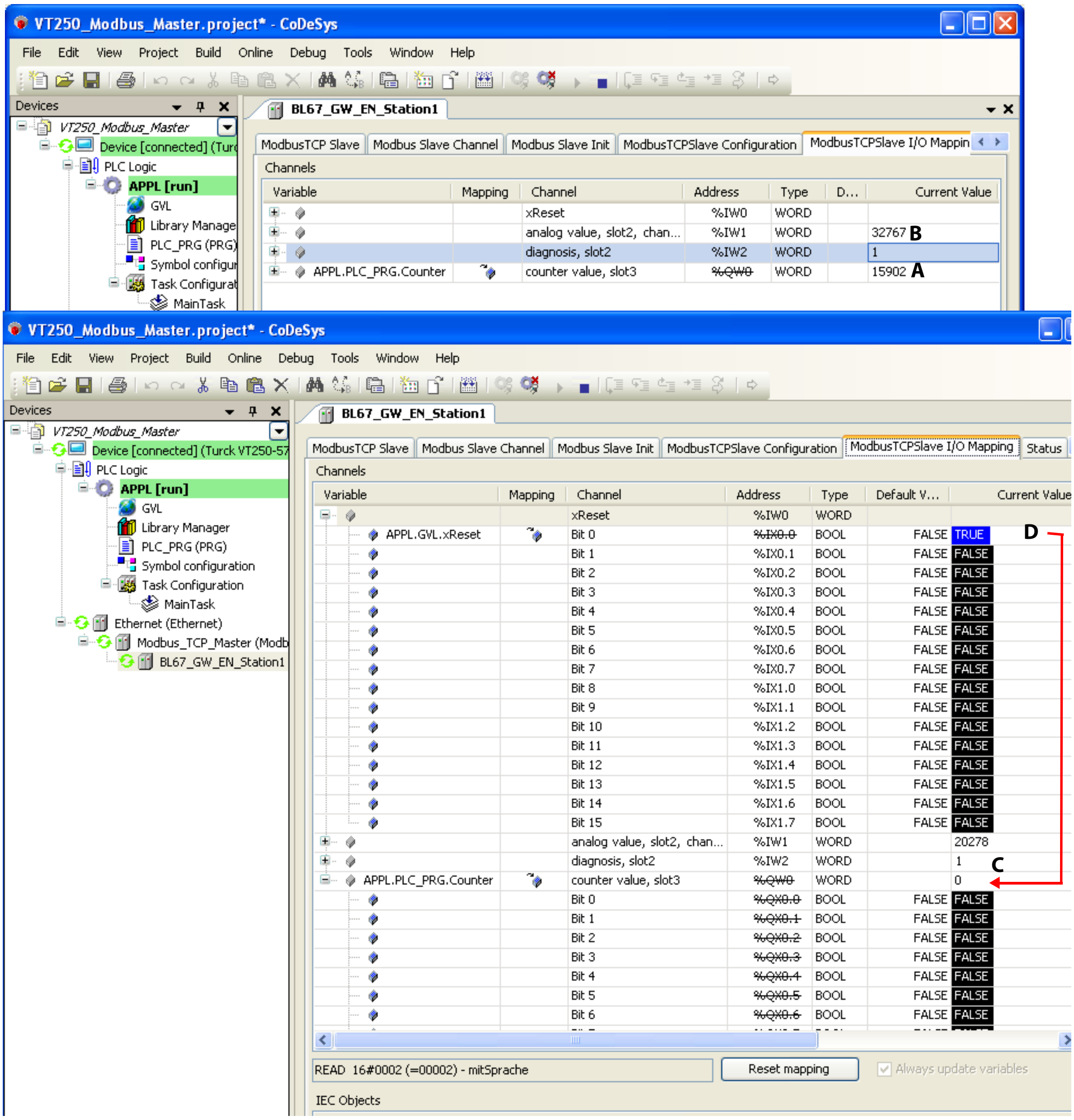
6 CoDeSys 3 also shows that the configuration of the network is correct

Reading out the process data

The station's process data are shown in the register tab "ModbusTCPSlave I/O Mapping".

Note

In order assure a regular updating of the process data, activate the function "Always update variables".



Variable	Mapping	Channel	Address	Type	D...	Current Value
		xReset	%IW0	WORD		
		analog value, slot2, chan...	%IW1	WORD		32767 B
		diagnosis, slot2	%IW2	WORD		1
APPL.PLC_PRG.Counter		counter value, slot3	~~%QW0~~	WORD		15902 A

Variable	Mapping	Channel	Address	Type	Default V...	Current Value
		xReset	%IW0	WORD		
APPL.GVL.xReset		Bit 0	~~%IX0.0~~	BOOL	FALSE	TRUE
		Bit 1	%IX0.1	BOOL	FALSE	FALSE
		Bit 2	%IX0.2	BOOL	FALSE	FALSE
		Bit 3	%IX0.3	BOOL	FALSE	FALSE
		Bit 4	%IX0.4	BOOL	FALSE	FALSE
		Bit 5	%IX0.5	BOOL	FALSE	FALSE
		Bit 6	%IX0.6	BOOL	FALSE	FALSE
		Bit 7	%IX0.7	BOOL	FALSE	FALSE
		Bit 8	%IX1.0	BOOL	FALSE	FALSE
		Bit 9	%IX1.1	BOOL	FALSE	FALSE
		Bit 10	%IX1.2	BOOL	FALSE	FALSE
		Bit 11	%IX1.3	BOOL	FALSE	FALSE
		Bit 12	%IX1.4	BOOL	FALSE	FALSE
		Bit 13	%IX1.5	BOOL	FALSE	FALSE
		Bit 14	%IX1.6	BOOL	FALSE	FALSE
		Bit 15	%IX1.7	BOOL	FALSE	FALSE
		analog value, slot2, chan...	%IW1	WORD		20278
		diagnosis, slot2	%IW2	WORD		1
APPL.PLC_PRG.Counter		counter value, slot3	~~%QW0~~	WORD		0
		Bit 0	~~%QX0.0~~	BOOL	FALSE	FALSE
		Bit 1	~~%QX0.1~~	BOOL	FALSE	FALSE
		Bit 2	~~%QX0.2~~	BOOL	FALSE	FALSE
		Bit 3	~~%QX0.3~~	BOOL	FALSE	FALSE
		Bit 4	~~%QX0.4~~	BOOL	FALSE	FALSE
		Bit 5	~~%QX0.5~~	BOOL	FALSE	FALSE
		Bit 6	~~%QX0.6~~	BOOL	FALSE	FALSE

READ 16#0002 (=00002) - mitSprache

Reset mapping

☑ Always update variables

IEC Objects

Figure 28: Modbus TCP Slave I/O Mapping

A Counter value

B Analog value

C Diagnostic message at analog module, evaluation (see below page 1-29)

D "xReset" = 1, reset of the counter to "0"

Interpretation of the diagnostic message at slot 2 (%IW1)

According to the definition of the Modbus communication channel, %IW1 contains the diagnostic data for the analog module at slot 2 of the example station (see Setting the Modbus channels, (examples), example 4, page 1-22).

The diagnostic message is to be interpreted as follows:

%IW 2, "actual value" = 1

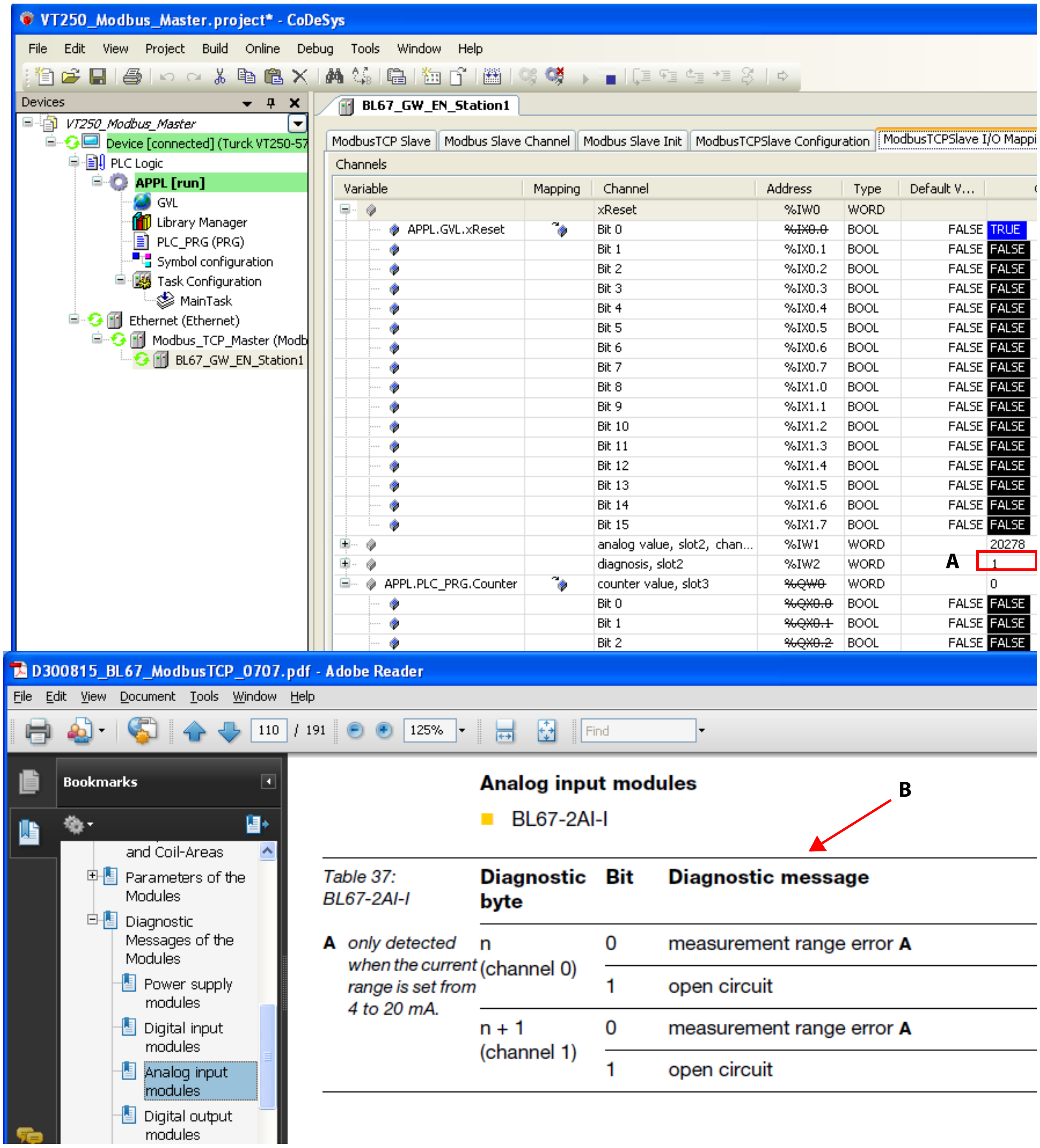
→ byte 0, bit 0 = 1

→ diagnostic message: "Measurement range error" at channel 0

Figure 29: Diagnostic message at analog module

A Diagnostic byte

B Description of the diagnostic byte of the module in D300815, chapter 4



Variable	Mapping	Channel	Address	Type	Default V...	
		xReset	%IW0	WORD		
APPL.GVL.xReset		Bit 0	~~%IX0.0~~	BOOL	FALSE	TRUE
		Bit 1	%IX0.1	BOOL	FALSE	FALSE
		Bit 2	%IX0.2	BOOL	FALSE	FALSE
		Bit 3	%IX0.3	BOOL	FALSE	FALSE
		Bit 4	%IX0.4	BOOL	FALSE	FALSE
		Bit 5	%IX0.5	BOOL	FALSE	FALSE
		Bit 6	%IX0.6	BOOL	FALSE	FALSE
		Bit 7	%IX0.7	BOOL	FALSE	FALSE
		Bit 8	%IX1.0	BOOL	FALSE	FALSE
		Bit 9	%IX1.1	BOOL	FALSE	FALSE
		Bit 10	%IX1.2	BOOL	FALSE	FALSE
		Bit 11	%IX1.3	BOOL	FALSE	FALSE
		Bit 12	%IX1.4	BOOL	FALSE	FALSE
		Bit 13	%IX1.5	BOOL	FALSE	FALSE
		Bit 14	%IX1.6	BOOL	FALSE	FALSE
		Bit 15	%IX1.7	BOOL	FALSE	FALSE
		analog value, slot2, chan...	%IW1	WORD		20278
		diagnosis, slot2	%IW2	WORD		1
APPL.PLC_PRG.Counter		counter value, slot3	~~%QW0~~	WORD		0
		Bit 0	~~%QX0.0~~	BOOL	FALSE	FALSE
		Bit 1	~~%QX0.1~~	BOOL	FALSE	FALSE
		Bit 2	~~%QX0.2~~	BOOL	FALSE	FALSE

Analog input modules

- BL67-2AI-I

Table 37: BL67-2AI-I

Diagnostic byte	Bit	Diagnostic message
n (channel 0)	0	measurement range error **A**
	1	open circuit
n + 1 (channel 1)	0	measurement range error **A**
	1	open circuit

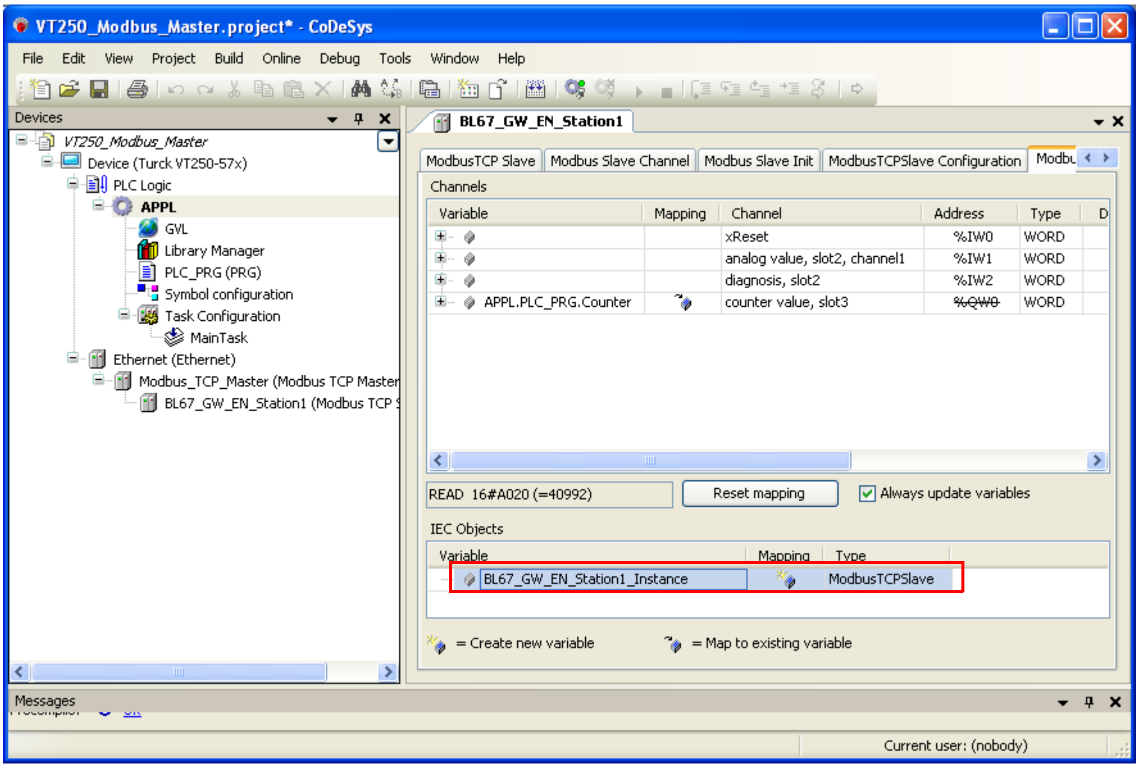
A *only detected when the current range is set from 4 to 20 mA.*

Modbus error handling in CoDeSys

In case of a slave failure, an error bit is set in the Modbus master ("xError").

After the elimination of the failure, the slave is only automatically restarted by the master, if a confirmation ("xConfirmError") of this error bit is executed in the master.

"xError" as well as "xConfirmError" are part of the ModbusTCPSlave-instance. They are automatically assigned to the instance as soon as a slave is added to the Modbus master in the CoDeSys project tree.



Variable	Mapping	Channel	Address	Type
		xReset	%IW0	WORD
		analog value, slot2, channel1	%IW1	WORD
		diagnosis, slot2	%IW2	WORD
APPL.PLC_PRG.Counter		counter value, slot3	~~%QW0~~	WORD

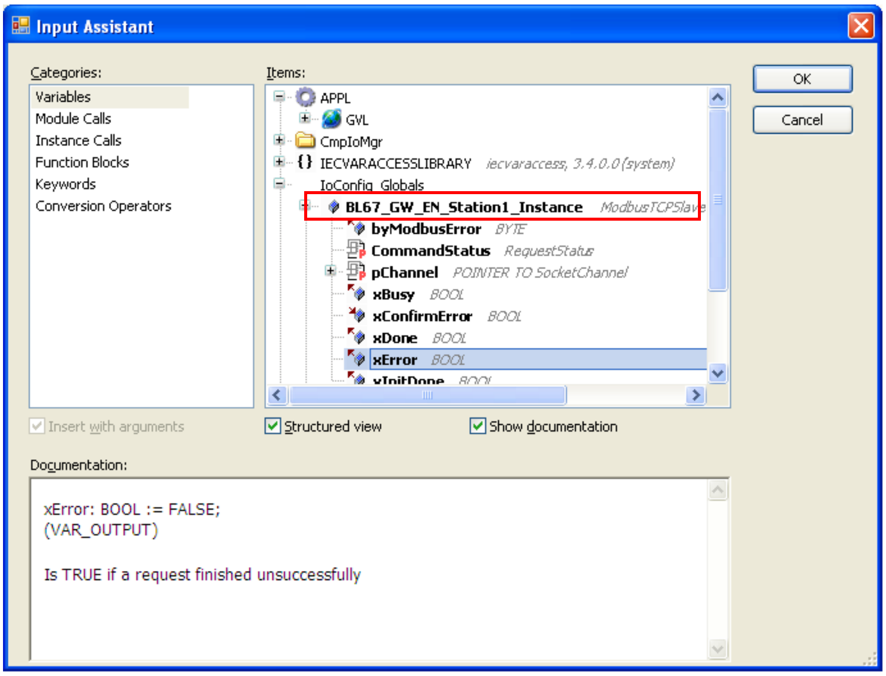
READ 16#A020 (=40992) Reset mapping ☑ Always update variables

IEC Objects

Variable	Mapping	Type
BL67_GW_EN_Station1_Instance		ModbusTCPSlave

= Create new variable = Map to existing variable

Figure 30: Slave instance



Categories:

- Variables
- Module Calls
- Instance Calls
- Function Blocks
- Keywords
- Conversion Operators

Items:

```
APPL
  GVL
CmpIoMgr
{} IECVARACCESSLIBRARY   iecvaraccess, 3.4.0.0 (system)
IoConfig_Globals
  BL67_GW_EN_Station1_Instance   ModbusTCPSlave
    byModbusError   BYTE
    CommandStatus   RequestStatus
    pChannel   POINTER TO SocketChannel
    xBusy   BOOL
    xConfirmError   BOOL
    xDone   BOOL
    xError   BOOL
    xInitDone   BOOL
```

☑ Insert with arguments ☑ Structured view ☑ Show documentation

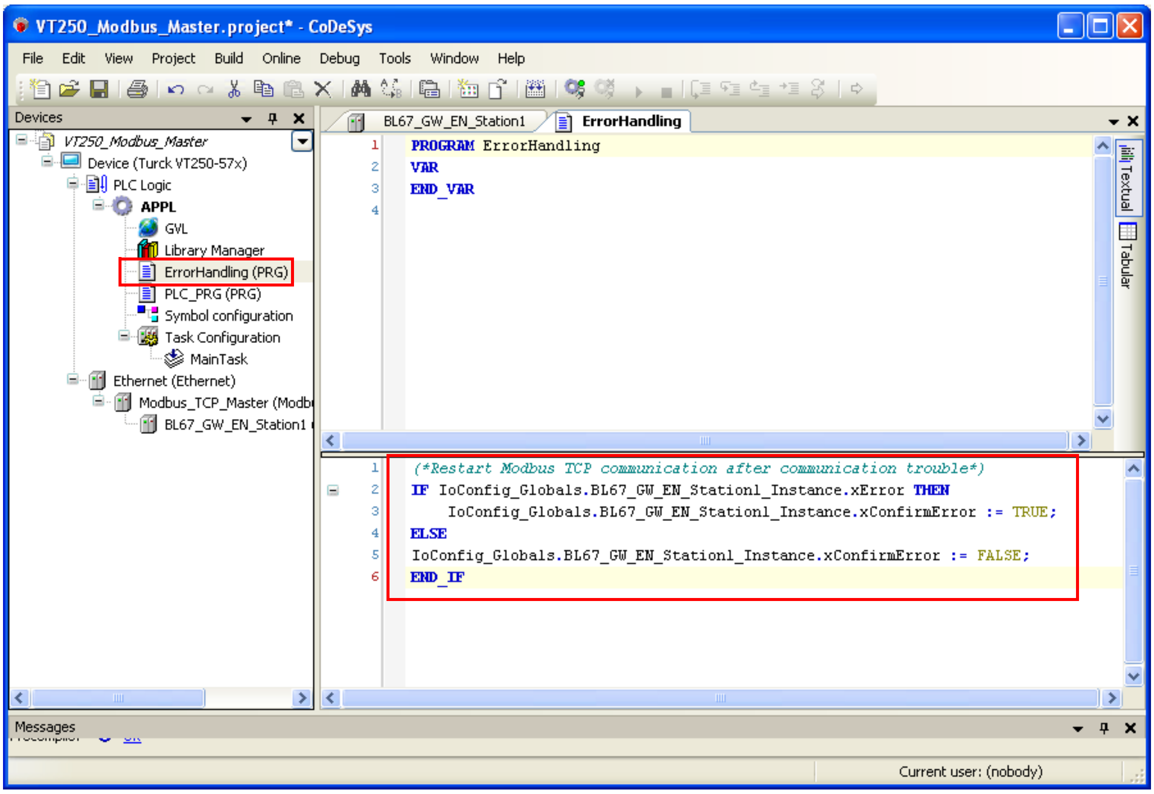
Documentation:

xError: BOOL := FALSE;
(VAR_OUTPUT)

Is TRUE if a request finished unsuccessfully

Figure 31: Variables of the slave instance

In this example, the error handling is done in a separate program (POU = Program Organization Unit) "ErrorHandling" (see below).



```
PROGRAM ErrorHandling
VAR
END_VAR
```

```
(*Restart Modbus TCP communication after communication trouble*)
IF IoConfig_Globals.BL67_GW_EN_Station1_Instance.xError THEN
    IoConfig_Globals.BL67_GW_EN_Station1_Instance.xConfirmError := TRUE;
ELSE
IoConfig_Globals.BL67_GW_EN_Station1_Instance.xConfirmError := FALSE;
END_IF
```

Figure 32: ErrorHandling

2 Index

TURCK

Industrial
Automation

Hans Turck GmbH & Co. KG
45472 Mülheim an der Ruhr
Germany
Witzlebenstraße 7
Tel. +49 (0) 208 4952-0
Fax +49 (0) 208 4952-264
E-Mail more@turck.com
Internet www.turck.com