Industrial Ethernet Training 02

Example application of the Serial Converter and TCP/RTU Gateway

Abstract:

The Modbus converters allow to interface a Modbus device with one or more generic serial devices, such as a sensor, printer or a machine. The serial messages and commands are completely configurable from Modbus side and the converter is bi-directional, meaning that it is possible to write and read. This application note provides an example how to use the converter to connect a Modbus RTU sensor to a CODESYS soft PLC running a Modbus TCP master.

Hardware reference

No.	Component name	Article No.	Hardware / Firmware version
1	IE-Training Kit-01	2881730000	1.1.2 (Build 125086)
2			
3			

IE-Training Kit Content

No.	Component name	Article No.	Hardware / Firmware version
1	IE-SR-4TX	2751270000	1.4.7
2	IE-SW-AL08M-8TX	2682280000	1.08
3	IE-SW-AL05M-5TX	2682250000	1.14
4	IE-CS-MBGW-2TX-1COM	2682600000	3.11

Software reference

No.	Software name	Article No.	Software version
1	CODESYS		V3.5 SP16
2			
3			

File reference

No.	Name	Description	Version
1			
2			

Contact
Weidmüller Interface GmbH & Co. KG
Klingenbergstraße 26
32758 Detmold, Germany
www.weidmueller.com

For any further support please contact your local sales representative:
https://www.weidmueller.com/countries

Content

1 Warning and Disclaimer

Warning
Controls may fail in unsafe operating conditions, causing uncontrolled operation of the controlled devices. Such hazardous events can result in death and / or serious injury and / or property damage. Therefore, there must be safety equipment provided / electrical safety design or other redundant safety features that are independent from the automation system.

Disclaimer
This Application Note / Quick Start Guide / Example Program does not relieve you of the obligation to handle it safely during use, installation, operation and maintenance. Each user is responsible for the correct operation of his control system. By using this Application Note / Quick Start Guide / Example Program prepared by Weidmüller, you accept that Weidmüller cannot be held liable for any damage to property and / or personal injury that may occur because of the use.

Note
The given descriptions and examples do not represent any customer-specific solutions, they are simply intended to help for typical tasks. The user is responsible for the proper operation of the described products. Application notes / Quick Start Guides / Example Programs are not binding and do not claim to be complete in terms of configuration as well as any contingencies. By using this Application Note / Quick Start Guide / Example Program, you acknowledge that we cannot be held liable for any damages beyond the described liability regime. We reserve the right to make changes to this application note / quick start guide / example at any time without notice. In case of discrepancies between the proposals Application Notes / Quick Start Guides / Program Examples and other Weidmüller publications, like manuals, such contents have always more priority to the examples. We assume no liability for the information contained in this document. Our liability, for whatever legal reason, for damages caused using the examples, instructions, programs, project planning and performance data, etc. described in this Application Note / Quick Start Guide / Example is excluded.

Security notes
In order to protect equipment, systems, machines and networks against cyber threats, it is necessary to implement (and maintain) a complete state-of-the-art industrial security concept. The customer is responsible for preventing unauthorized access to his equipment, systems, machines and networks. Systems, machines and components should only be connected to the corporate network or the Internet if necessary and appropriate safeguards (such as firewalls and network segmentation) have been taken.

2 Prerequisites for doing

You need to have the following hard- and software and documentation

- Industrial Ethernet Training Kit
- Application Note Industrial Ethernet Training 01 “Setting up default configuration of IE Training Kit” for applying default IP address configuration
- CODESYS Version 3.5 SP16 or higher, download here: Download (codesys.com)

3 Modbus Protocol

For reading sensors and communication between devices, the Modbus protocol is used. It is a standard protocol for communication between industrial devices using the request-response method. There is a master, most of the times our computer or a PLC, that requests information from the slave devices.
These slave devices for example can be Industrial machines or sensors that then respond with the requested information. Furthermore, they only provide information when they are asked for it by the master. There are different types of Modbus protocols, with the most common versions being Remote Terminal Unit (RTU) or Modbus Transmission Control Protocol (TCP). Modbus data is being transmitted in binary form over serial lines in the RTU protocol and over Ethernet in the TCP protocol.
Modbus packets have specific formats including the device address, a function code, registers as well as the data and a checksum, as illustrated by the figure below.

Figure 1: Modbus RTU packet

4 Configuring Modbus Gateway

To continue, make sure that you are either directly connected to the Modbus Gateway on the Training Kit or that you are connected to the router or switch via LAN, which then must be connected to the Modbus Gateway via Ethernet cable as well.
The Modbus Gateway acts like a translator for Modbus RTU, which is the sensor's communication protocol, and Modbus TCP, which is the protocol for Ethernet-based communication in the network. This means, that the sensor is sending its data based on the Modbus RTU protocol to the Modbus Gateway, which at this point is a not readable format. Then, the Modbus Gateway translates the incoming data to Modbus TCP protocol. Therefore, the data can be sent with Ethernet to the Modbus Master. In this case, the Master is the computer which is displaying the data.
As per the default settings, the Modbus Gateway is not configured in a way to communicate with Modbus RTU and translating it to Modbus TCP, which is why a certain configuration is needed beforehand.

1. Log in to the Modbus Gateway web interface via your browser with the IP we configured beforehand, "*192.168.1.50*" and the credentials. Make sure your computer is in the same network as the Modbus Gateway to be able to connect to the web interface.

Figure 2: Accessing web interface

2. Navigate to "*Serial Port Settings*" and then to "*Serial Configuration*". The following table explains the settings found in this menu option.

Setting	Explanation
Port Alias	Option for giving the Port with the traffic an individual name
Interface	The cable your are using to connect the sensor to the Gateway
Baud Rate	The sensor's transmission frequency, can be found in manual
Data Bits	The amount of allowed bits to send data, optimally 1 byte (8bits)
Stop Bits	Allow to identify a data packet and perpare for the next byte of data
Parity	Parity refers to a control bit which is used to recognize transmission errors
Flow Control	Controls the maximum flow of data traffic to allow for a synchronized data transmission, in case one device is slower than the other

Table 1: Serial configuration settings

For the sensor on the Training Kit, these are the following configuration settings for the Serial Port. Change them as depicted below and hit "*Apply*" to apply the configuration. For any other sensor, refer to the manual and the explanation table to adjust the settings accordingly.

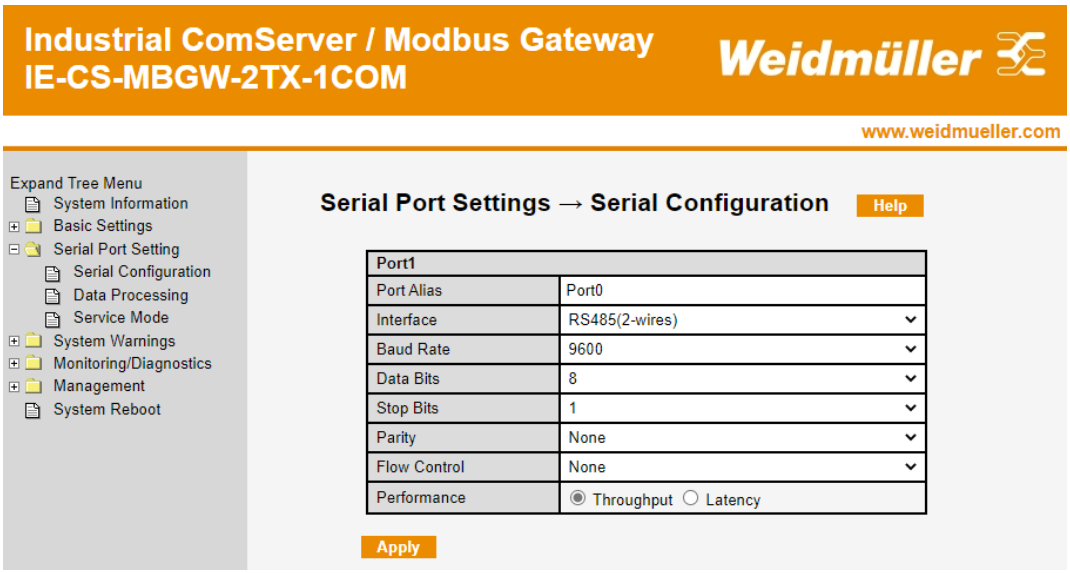


Figure 3: Serial Port configuration

3. To further configure the Modbus Gateway, navigate again to "*Serial Port Settings*" and then to "*Service Mode*". Set the "*Service Mode*" with the drop-down menu to "*Modbus: TCP Master to Serial Slave Gateway*" to read the Modbus RTU sensor. Furthermore, select "*Modbus RTU*" as the "*Serial Protocol".* All the remaining settings can stay in their default state for now, but they can also be tweaked to our liking, for example the timeout to close the connection when there is no traffic coming for a certain time period.

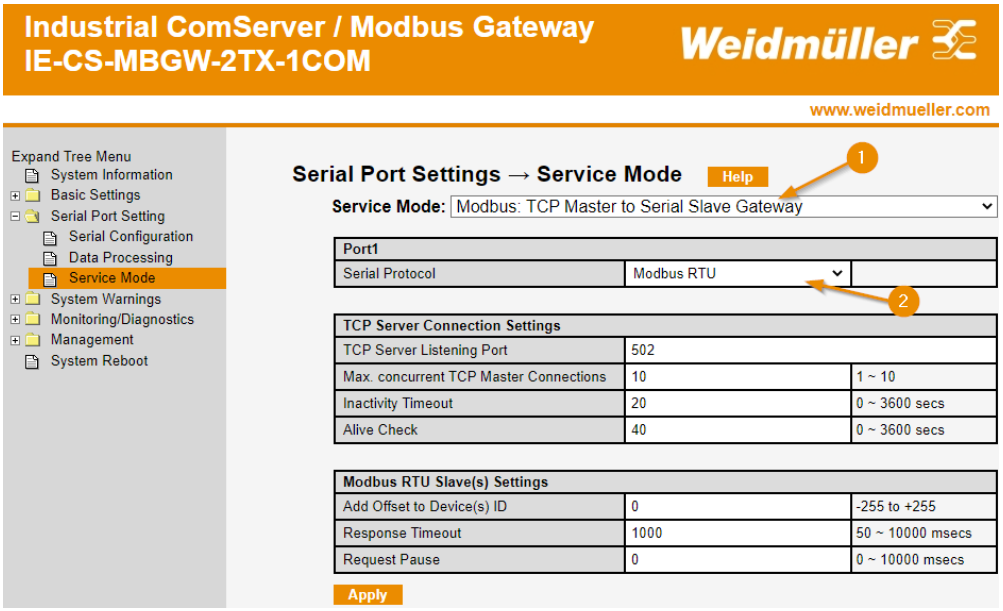


Figure 4: Service mode configuration

4.1 Troubleshooting with Modbus Gateway

In case there are connection issues or any problems with the Modbus traffic, Weidmueller's Modbus Gateway provides monitoring tools such as the Modbus Traffic Monitor.
With the Traffic monitor, one can follow all occurring traffic between the Modbus Gateway and the Slave device such as the sensor on the Training Kit. This method provides an early examination of the occurring problems because we can analyze if there is a connection between the devices or not.

1. To access the monitoring tool, go to the section "*Monitoring/Diagnostics*" and click on "*Modbus Traffic Monitor".* The following window opens with no logs written yet.

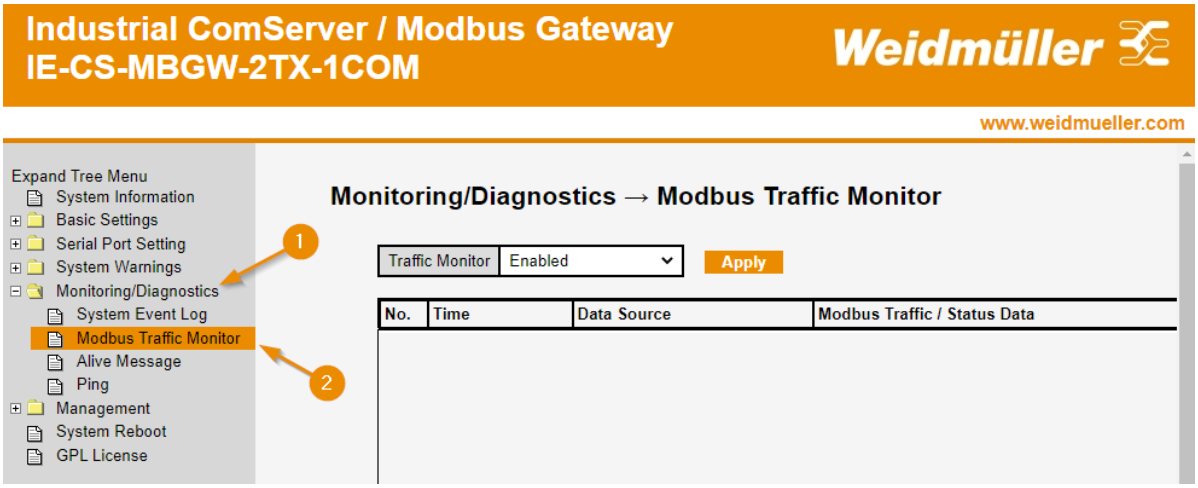


Figure 5: Modbus Traffic Monitor

2. The following screenshot shows example traffic between the sensor and our CODESYS application. We can see that a successful communication between these two includes four traffic messages. These state the ports and IPs the packets travelled in and out from and further also include the requests and data in Modbus RTU format.
In case this pattern is not recognizable or there is no traffic occurring, there might be a problem with the Modbus connection.

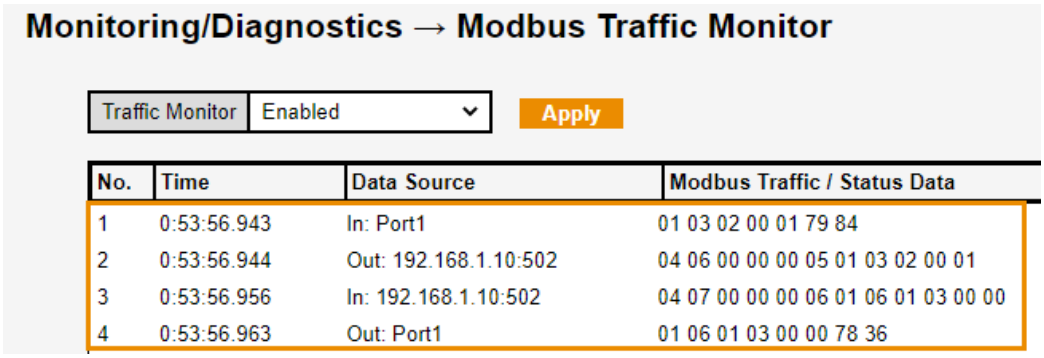


Figure 6: Modbus Traffic

5 Reading sensor data with CODESYS

Accessing certain data from sensors operating on the Modbus RTU protocol can be done by using the development environment CODESYS. Since our Training Kit does not provide a PLC for the program to run on, using CODESYS' soft PLC is very useful meaning that we can use our computer to access our RTU sensor. Knowing how to access our Modbus RTU sensors via CODESYS is an important skillset for reading and writing with our industrial sensors in our production hall for example.
Furthermore, a properly implemented, working configuration for one Modbus RTU sensor means that we can connect any other Modbus RTU sensor to our CODESYS configuration as well. Hence, we can read our sensor's data with one working program which we can also copy and utilize in our PLCs or other computers to analyze the data.

5.1 Setting up CODESYS Project

1. After installing the CODESYS software (Link can be found in chapter 2 Prerequisites for doing), please start the application. We are now on the starting page, where we must click on "*New Project*" under "*Basic Operations*" to start a new project.

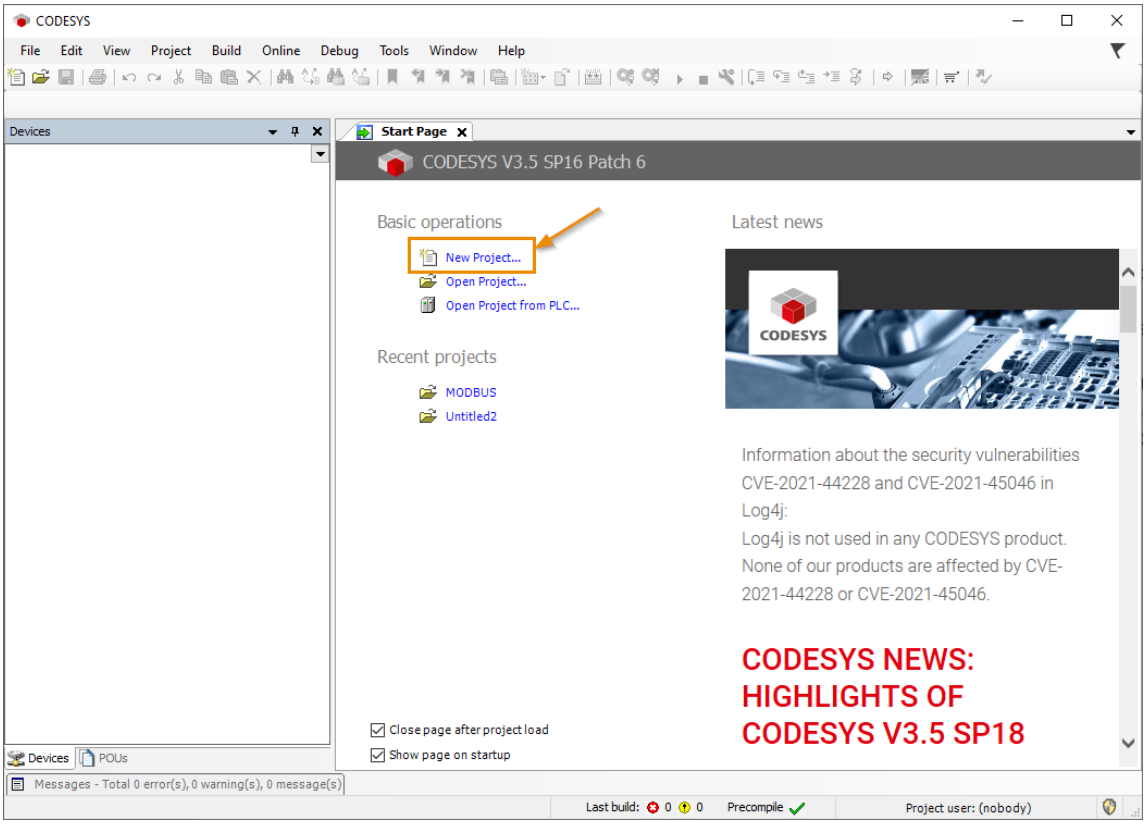


Figure 7: Starting new project

2. A window appears asking to name the project and decide which template to use. We want to use the "*standard project*" template and we can name it as we wish, we chose sensor.

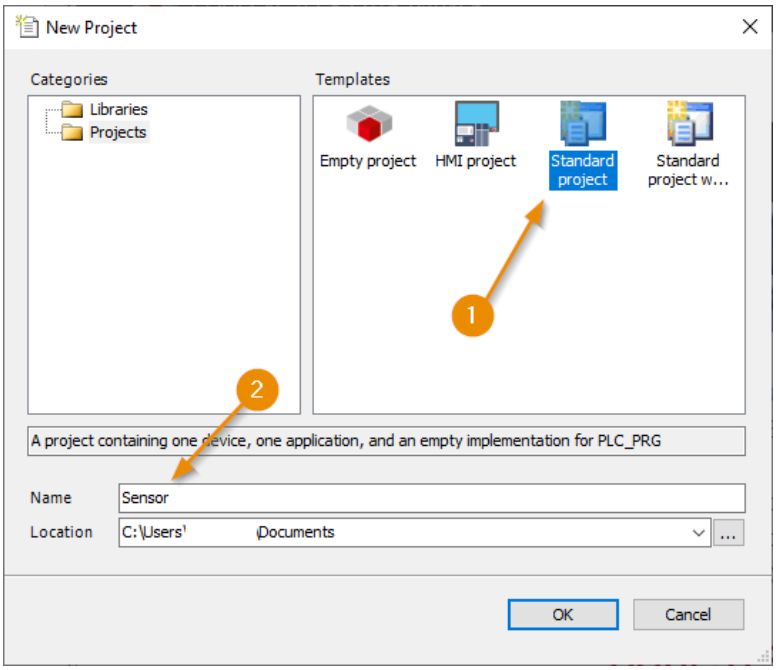


Figure 8: Naming project and selecting template

3. Next, we need to further configure our project. We are now asked to select a (virtual) programmable device and a PLC programming language. Now, it is important to either select, based on your operating system, the 32-bit or the 64-bit version. In our case, we need the x64 version as a device. You can select the appropriate version in the "*Device*" drop-down menu. As of the PLC language, we simply select "*Structured Text*" since we are not going to program our Modbus RTU sensor.

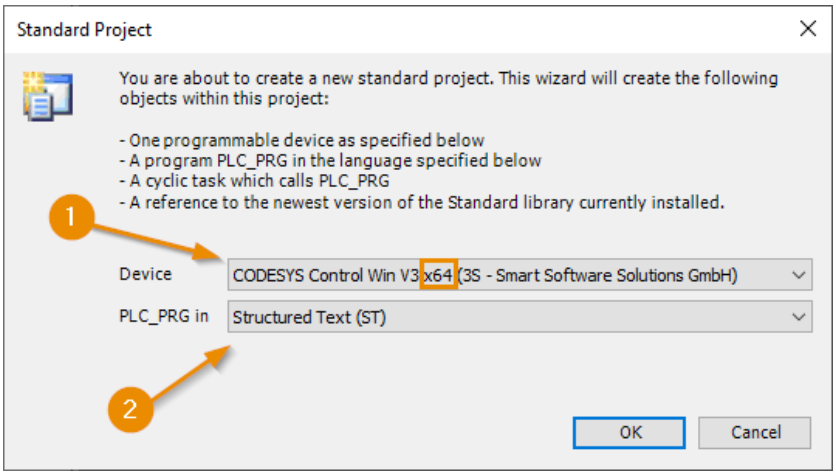


Figure 9: Selecting device and PLC

5.2 Setting up Modbus Device chain

1. We have now successfully set the basis for our CODESYS sensor reading project. Our next step is to add our devices to the project. This is done by right-clicking on “*Device*” in the tree on the left side and then clicking on “*Add Device…*”.

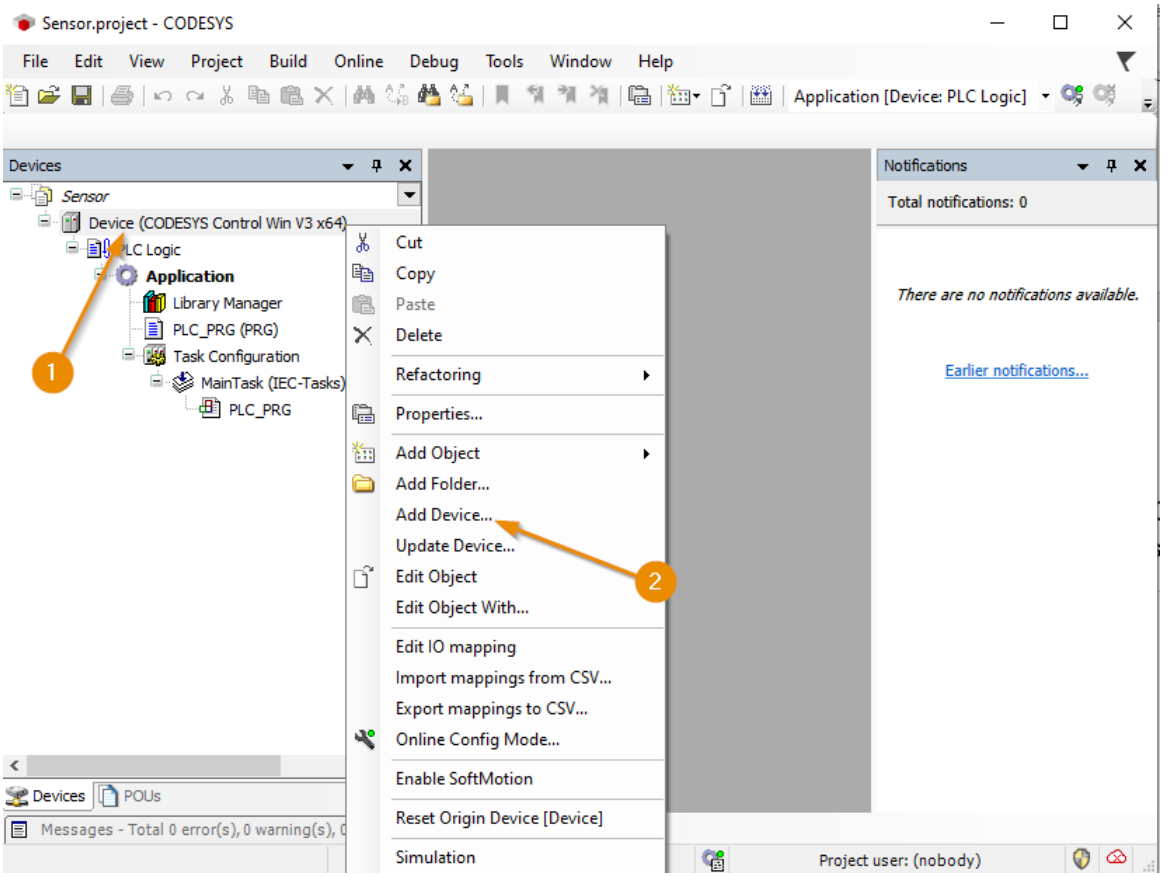


Figure 10: Adding new device

2. Since we are using our Ethernet cable to connect via LAN to our Modbus Gateway, we want to add an Ethernet device that is able to recognize and connect to other LAN devices. In fact, we click on "*Ethernet adapter*", then select "*Ethernet*" and lastly click on "*Add Device*".

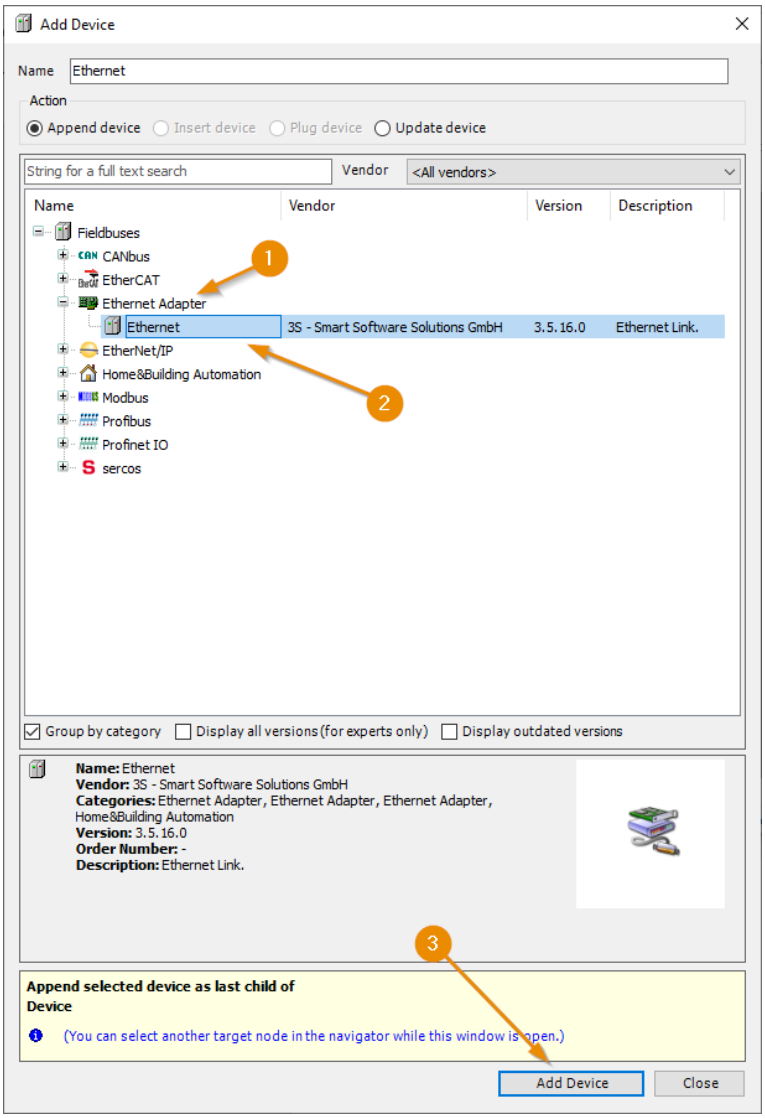


Figure 11: Appending Ethernet device

3. Next, we have to add our Modbus device. Of course, our Modbus device, in this case our Modbus Master that asks the sensor for information, is appended to the Ethernet device since we have our computer and Modbus gateway connected via LAN to each other. Hence, we repeat the previous method of adding a new device.
   But this time, we want to right-click on our Ethernet device since we want to append our Modbus Master to our Ethernet interface. After clicking on new device, we click on the menu “*Modbus*”. Now, we select the “*Modbus TCP Master*” menu and lastly click on the “*Modbus TCP Master*” and *Add device*.

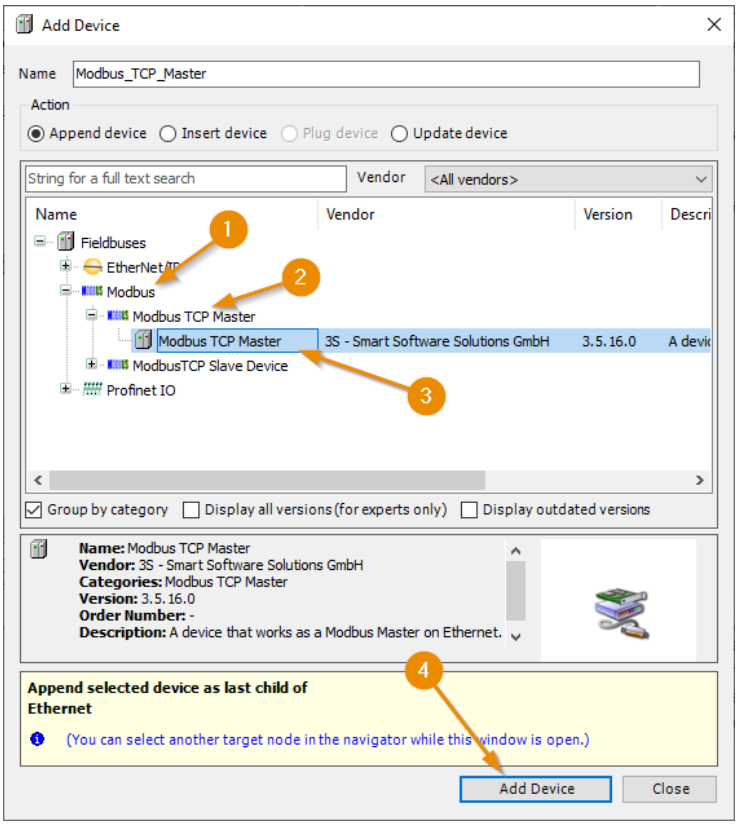


Figure 12: Adding Modbus master

4. Adding the Modbus Slave device is our next step. Hence, we once again repeat the previously mentioned steps for adding a new device. Note, that we must append our slave to our Modbus Master. Therefore, we right-click on our Modbus Master and click on "*Add Device…*", where we then append our Modbus Slave device to the Master.

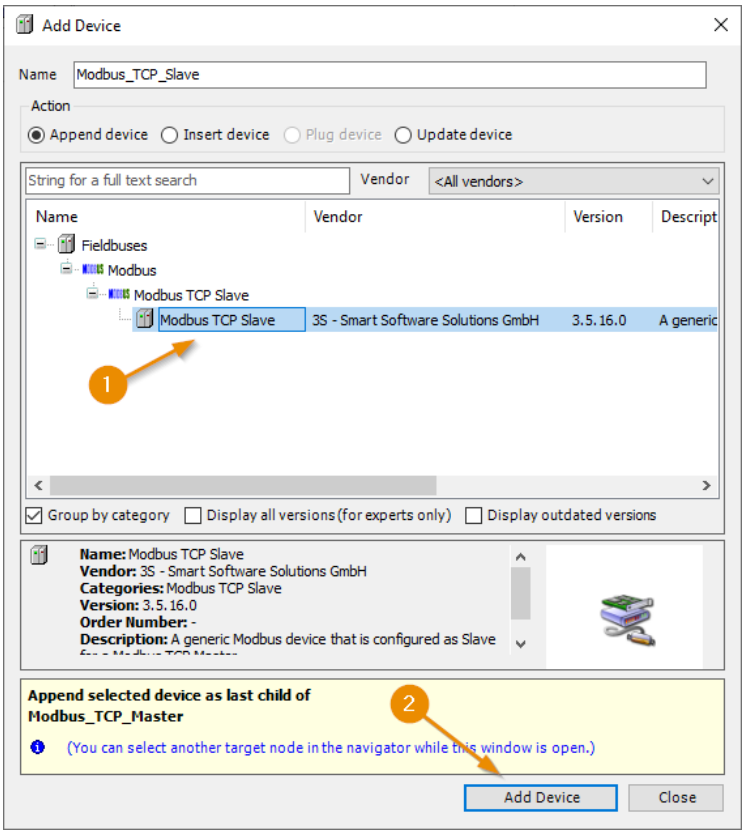


Figure 13: Adding Modbus Slave

As of now, we have successfully added our Modbus Master, which in our case is the computer, and the Modbus Slave device, which will be our Modbus Gateway. Lastly, we are going to append our Modbus sensor to our Slave to complete our device setup.

5. As mentioned before, our last step is adding the sensor to the device chain. This is done the same way we added all the other devices. Thus, we right-click on our Modbus Slave device and append the Modbus Slave COM-Port device.

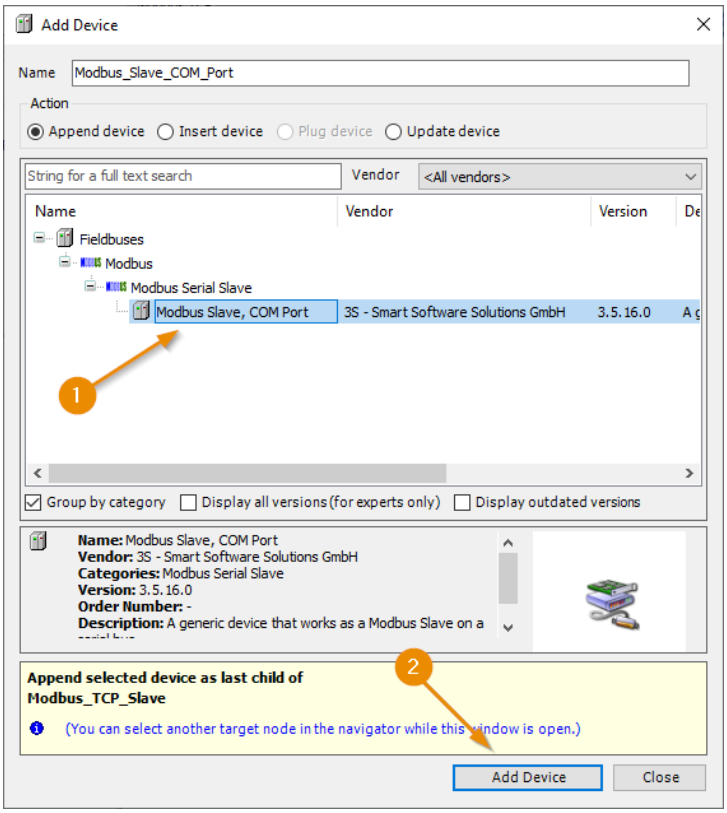


Figure 14: Adding Modbus sensor

5.3 Starting up virtual Gateway and Control

For our CODESYS application to work and to recognize our devices in the network, we must activate a gateway, which is given from CODESYS, and a soft PLC (virtual programmable logic controller) on which our application has to run on.

1. Firstly, we locate the small *Arrow Up* icon in our task bar and click on it.

Figure 15: Task bar arrow icon

2. Now, we want to locate the Gateway and Control icon from CODESYS, the icons are seen below.

Figure 16: CODESYS gateway and control icon

3. Lastly, right-click on both of these and start the PLC first by clicking "*Start PLC*", then do the same for the Gateway by clicking on "*Start Gateway*".

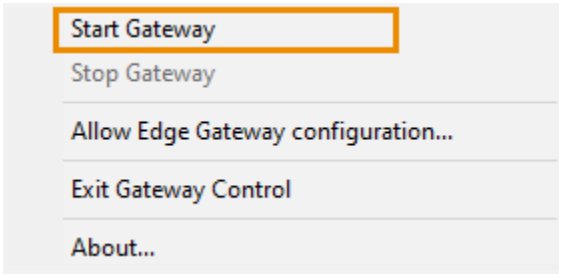


Figure 18: Starting up Gateway and PLC

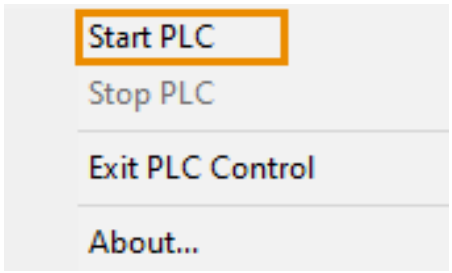


Figure 17: Starting up PLC

5.4 Configuring the devices

We have now successfully set up our project and started all the required applications needed for our project to run. As of now, our devices are not configured properly which we are going to do now.

1. To start, we must configure our *Device* which in this case is our PLC. Select Gateway 1 in the drop-down menu for the Gateway and then next to the Gateway we have to choose our computer in the drop-down menu. After doing so, press *Enter* and we should have two green circles signalizing our Gateway is configured properly.

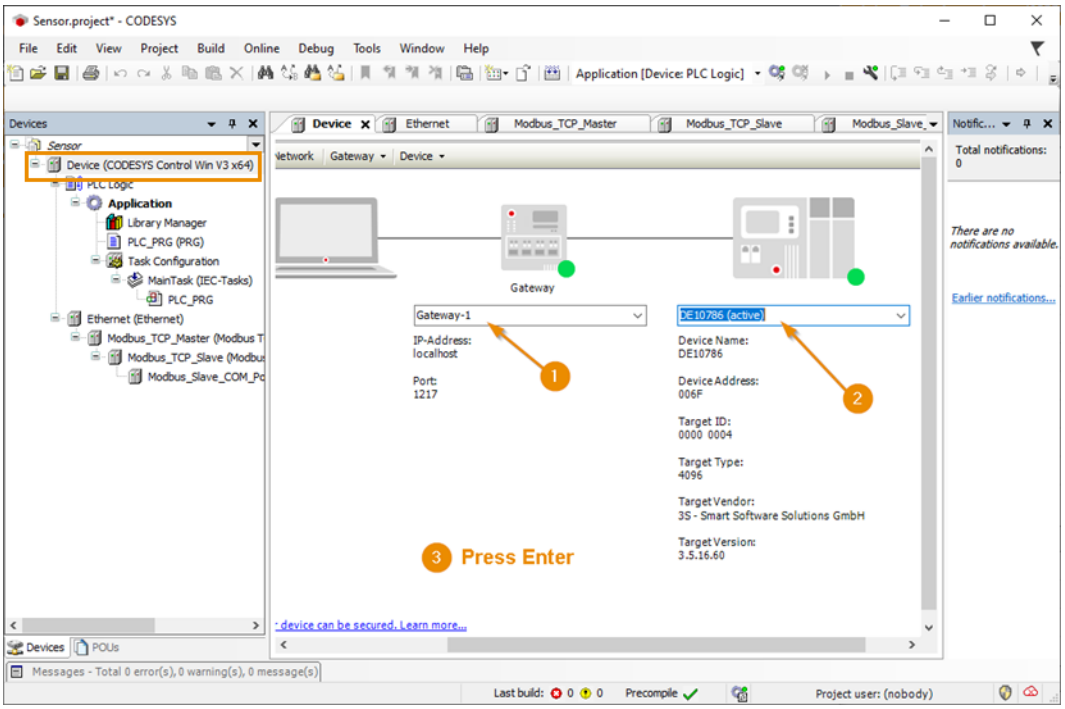


Figure 19: Device configuration

Furthermore, configure the *PLC Settings* of our *Device*. On the left-hand side we can find the option "*PLC Settings*" in the menu tree. Here, check the "*Update IO while in stop*" checkbox and select "*Enabled 1*" in the drop-down menu "*Always update variables*" so we can see the sensor's measured values in real time.

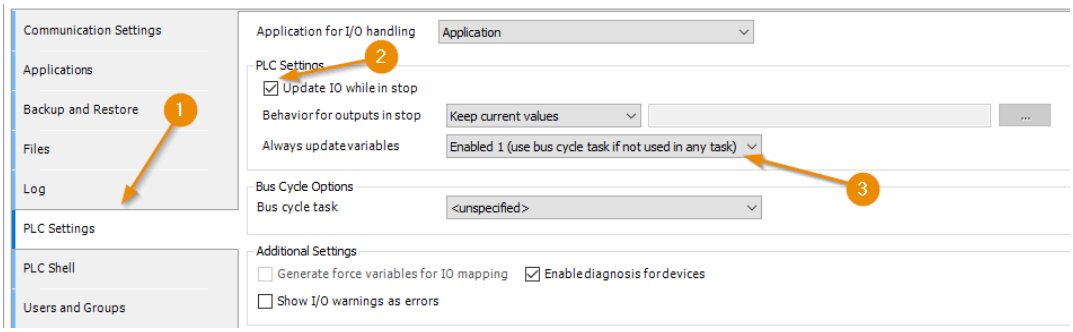


Figure 20: Configuring PLC settings

1. For the rest of the device topology, we are starting off with our sensor which is the *Modbus Slave COM Port*. It must have a device address and a response timeout configured. In this case, there is not much to configure as the sensor's slave address is per default 1 but your Slave address might be different per default, please refer to your device's manual for further information. Furthermore, our response timeout is set to 1000ms or 1 second before the device disconnects automatically when not transmitting data.

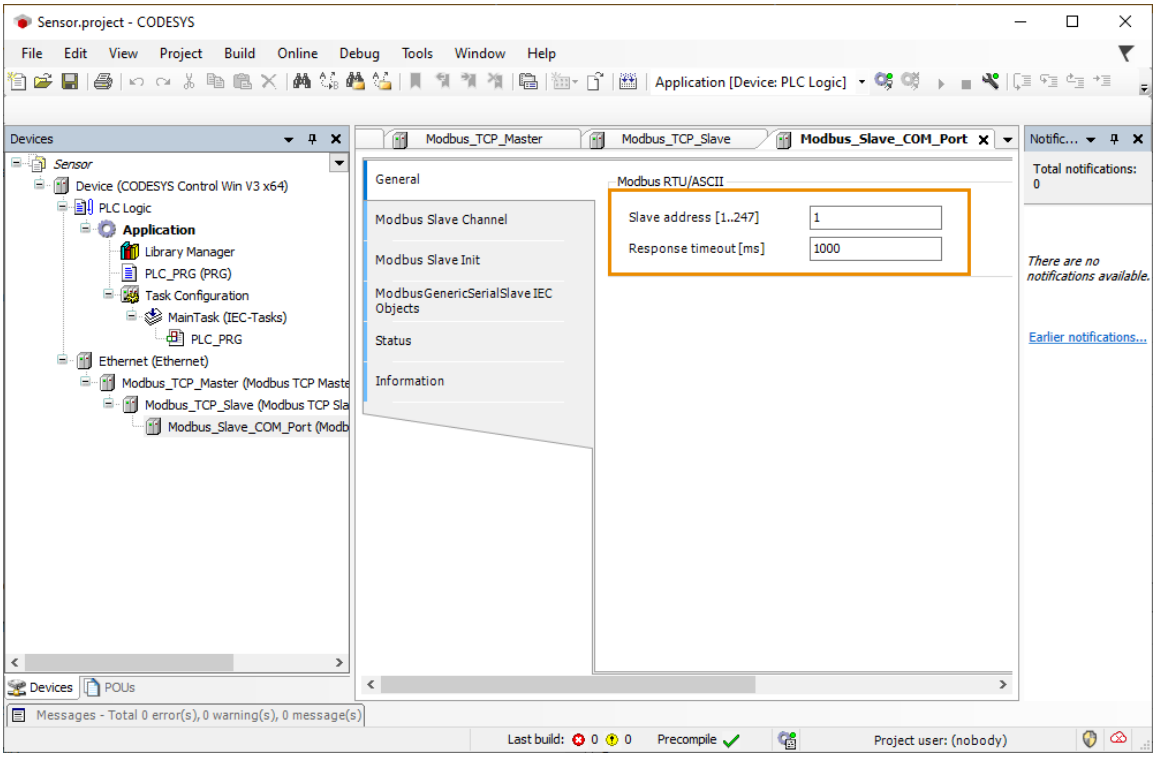


Figure 21: Configuring Slave address and response timeout

2. Next, we go one step higher in the device topology to the *Modbus TCP Slave*. This is our Modbus Gateway that transforms the data so we can read it with our application. Here, we insert our Gateway's IP address, which in our case is "*192.168.1.50*" and the according port which is "*502"*.

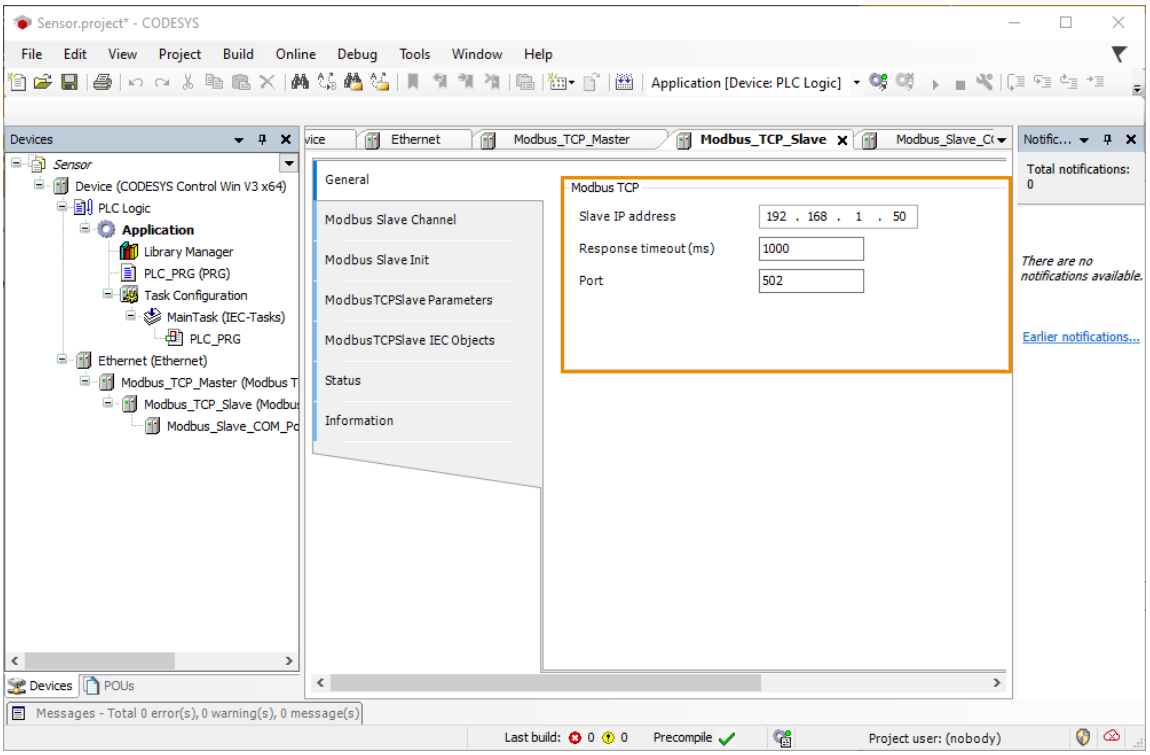


Figure 22: Adding gateway IP address

3. Our Modbus Master, that is our computer, does not need any further configuration which is why we skip it in our topology and go to our Ethernet device. Since we are using LAN, we must click on the button *"Browse…"* next to the *"Network interface*" field. The menu shows all kinds of possible network interfaces from the computer. Thus, we select the Ethernet interface which is connected to our Modbus Gateway's network and press *OK*.

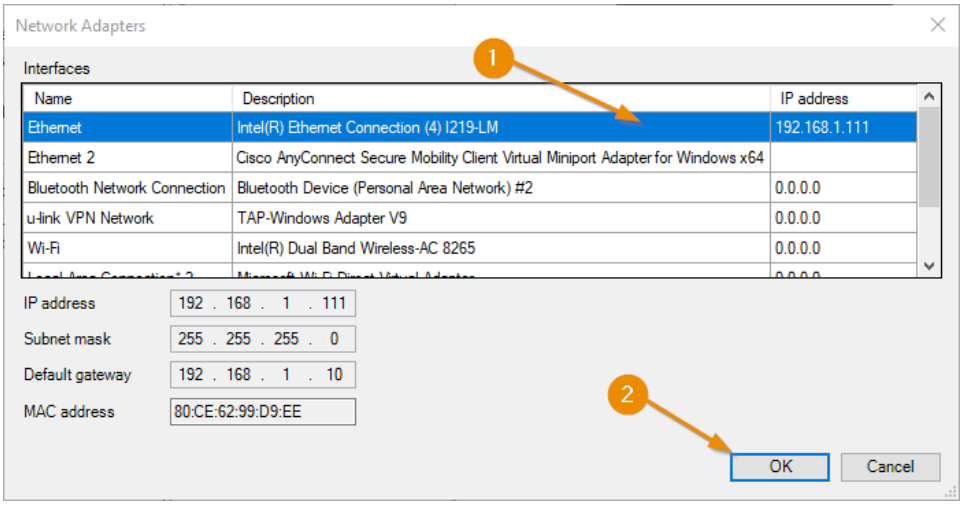


Figure 23: Selecting network interface

The settings are applied automatically afterwards. If the Gateway is 0.0.0.0, change it to the router's IP address (*192.168.1.10*) so the data will be redirected through the router to your computer.

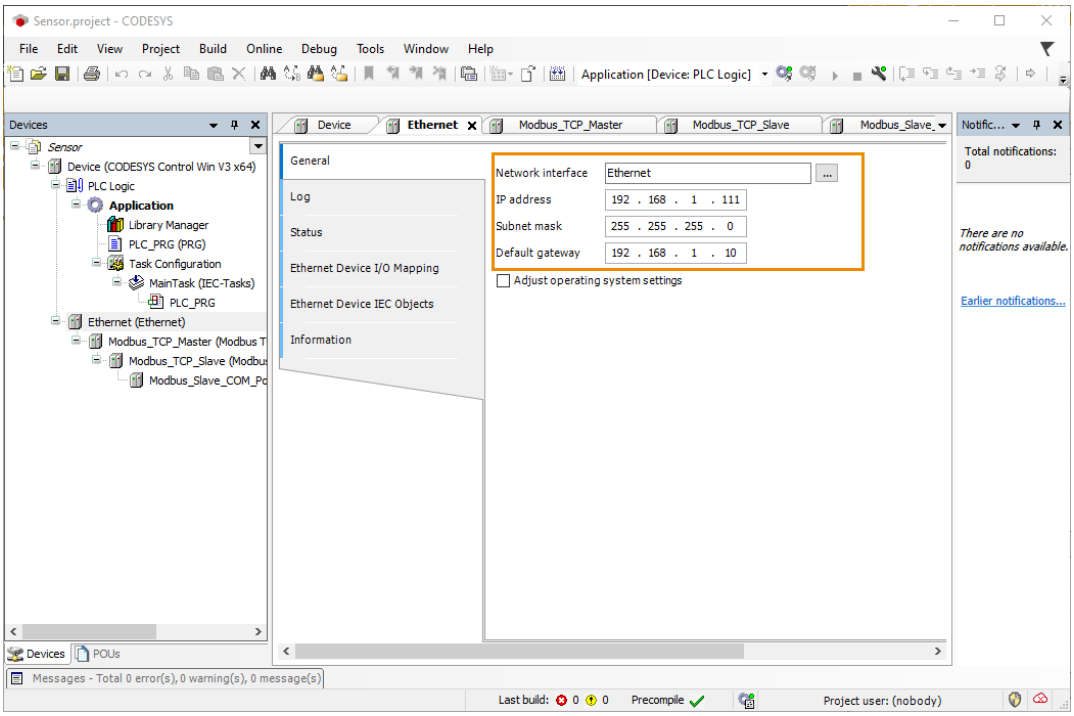


Figure 24: Ethernet settings

4. We may now log in to our device. This is done by either pressing *Alt + F8* or clicking on the small gearwheel up top in CODESYS. After starting, we might get a message to compile our current changes. Click on *Yes* to download the latest changes.

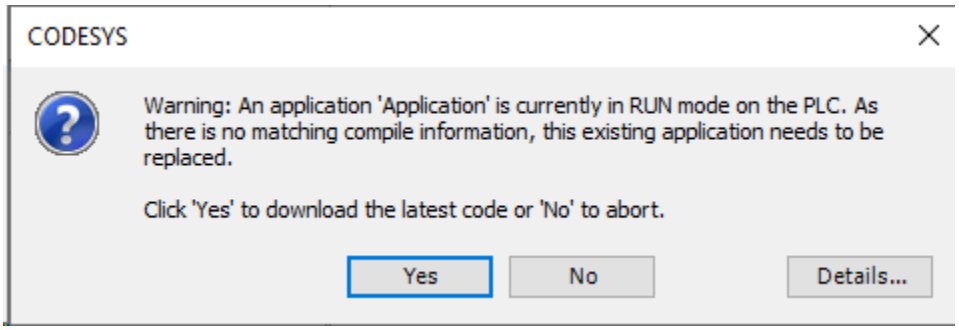


Figure 25: Accepting current changes

Our application is now up and running. As of now, no sensor data is transmitted even when we are pressing either *F5* or the "*Start*" button at the top to start our application, since we have not configured a query yet, but this will be our next step.

5.5 Configuring Slave Channel for reading sensor data

We have now finished our setup and configurations, meaning that we are ready to create a channel to read the sensor's data. Keep in mind that we have to log out before being able to create a Slave Channel or to edit our settings.

1. For that, we want to navigate to our Modbus COM Port in the device topology tree and then select the "*Modbus Slave Channel*".

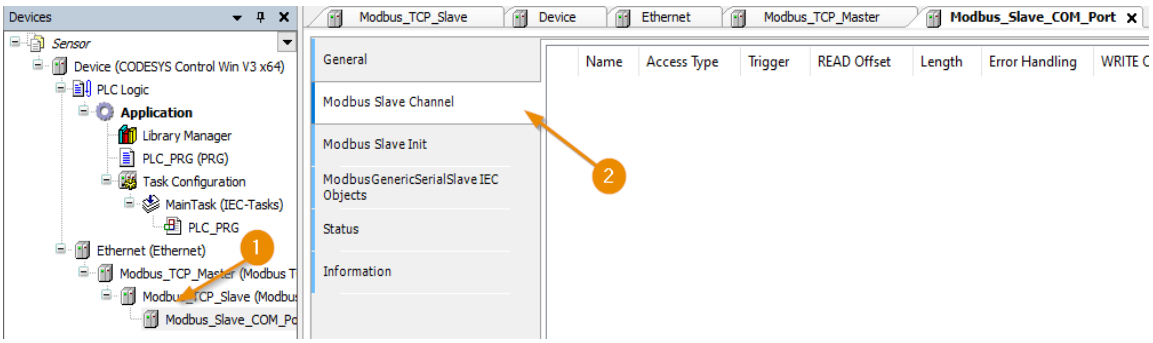


Figure 26: Modbus Slave Channel menu

In this menu, we then click on "*Add Channel…*" in the bottom right corner.

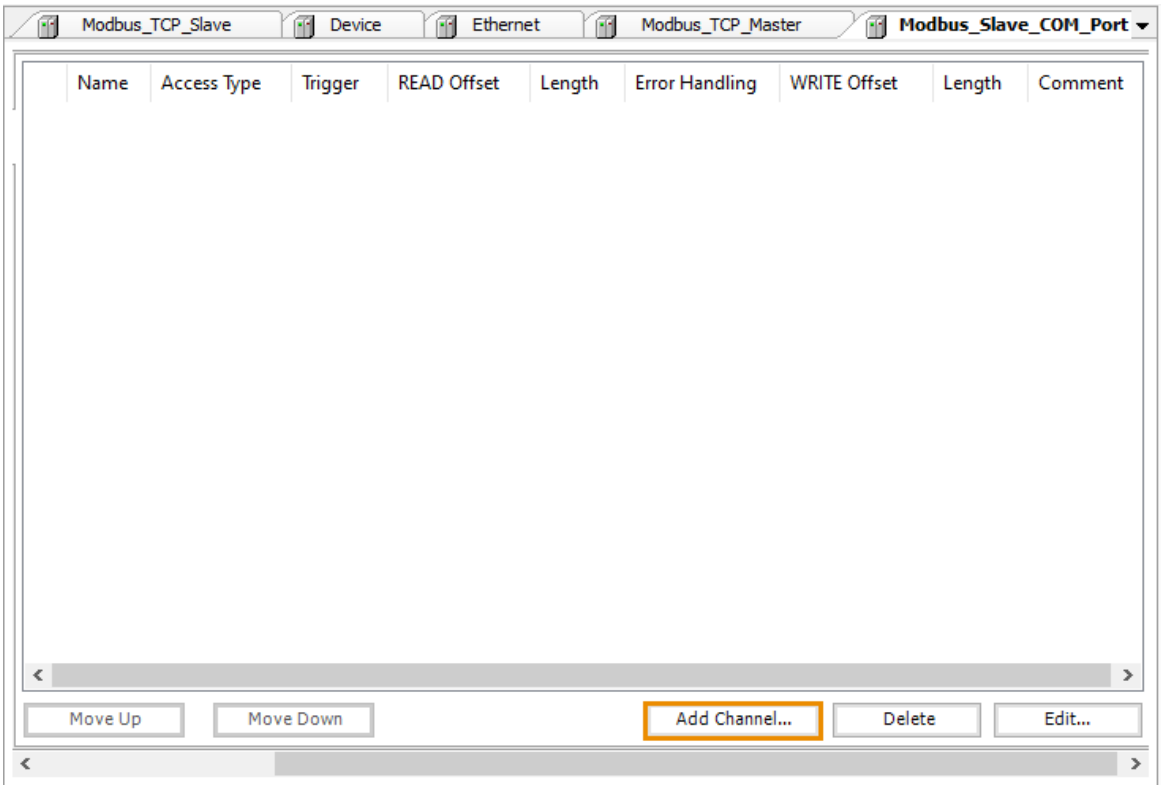


Figure 27: Adding new slave channel

2. Our Training Kit sensor is capable of measuring temperature and humidity. For this example, we are going to access the temperature data and Slave address.
   The table below explains the settings we can configure when creating a Slave Channel.

Setting	Explanation
Name	Giving the Channel a name
Access type	The function code you want to use in order to access a certain dataset of your sensor specific function code can be found in the sensor's manual
Trigger and Cycle time	The trigger for the channel to ask for the data, can be either cyclic with a certain cycle time or a rising edge which is defined in the variables
Comment	Comment your Slave Channel for better understanding
Offset	The specific register or offset you are addressing to get the data from the sensor you want, offsets can be found in the sensor's manual as well
Length	The length of the message sent in bytes
Error Handling	When there is an error in your application, you can choose what value to display, which can be either the last value or set to zero

Table 2: Settings explanation for configuring Slave Channel

3. To understand how and which registers to address, refer to the sensor's manual. The sensor on the Training Kit provides the following table with the registers and its contents.

Modbus Protocol			
Function Code			
Command Register		Funciotn	
0x03		Read keep register	
0x04		Read input register	
0x06		Write a single keep register	
0x10		Write more keep registers	
Register Type	Register Address	Register Contents	Bytes
Input Register	0x0001	Temperature	2
	0x0002	Humidity	2
Keep Register	0x0101	Device Address	2
	0x0102	Baud Rate: 0:9600 1:14400 2:19200	2
	0x0103	Temperature Correction -10℃~10℃	2
	0x0104	Humidity Correction -10%RH~10%RH	2

Figure 28: Register table for Training Kit sensor

As we can see, it states the function codes to work with the sensor and the corresponding registers to address. For example, reading an input register like the temperature, requires function code 0x04 and addressing the register 0x0001 whereas reading the Slave Address requires function code 0x03 and the register 0x0101.
The difference between input and keep registers being, that the input registers are meant to evaluate ingoing data, like temperature, humidity or any other measurements the sensor is capable of doing, and the keep registers act as a storage for the sensor, where it saves constant values like its Slave Address or the Baud Rate (Frequency of transmission).

4. Here is an example configuration of a reading Slave Channel for the sensor on the Training Kit. The Slave Channel shows the temperature using function code 4 to read the registers. We have a cyclic query every tenth of a second for our first offset, which has the temperature in it. In case of an error, we want to see the last measured value in our data.

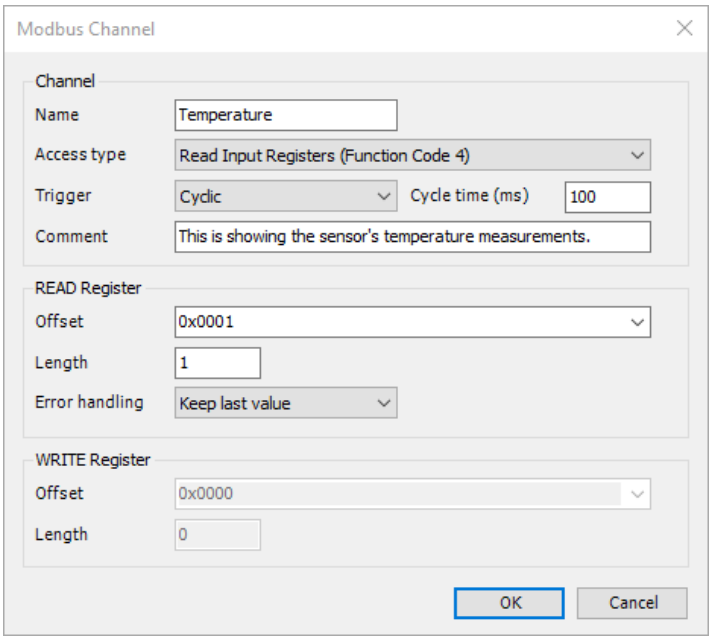


Figure 29: Example Slave Channel for temperature

Also, there is another example setting for reading the Slave Address of the sensor addressing the corresponding offset in our sensor and using function code 3 as the Slave Address is saved in a Holding Register and not Input Register like the temperature.

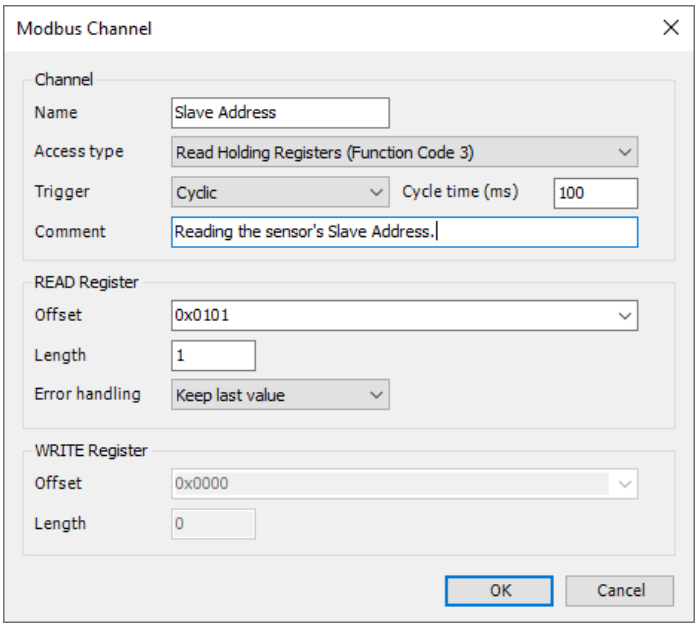


Figure 30: Reading Slave Address

5.6 Reading sensor data

We have now finished every step for reading any sensor data from our Modbus RTU sensor using CODESYS. Lastly, we will look at the option where we can see the data coming in.

1. In order to do that, we want to navigate to our *Modbus COM Port* and then click on "*Modbus Generic Serial Slave I/O Mapping*" on the left-hand side in the menu options. To finally see our sensor's data, we click on the "+" symbol of our configured channel to unfold its values which are located in the column "*Current Value*". If we want to, we can unfold it even further to see the bits in detail.

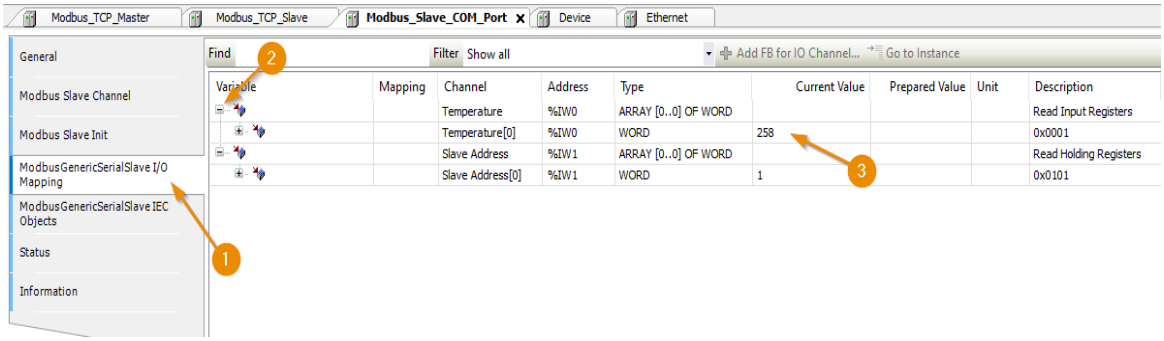


Figure 31: Reading sensor data

We can see our sensor measuring 25.8 degrees Celsius, which is a bit warmer than usual temperatures indoor and its Slave Address is 1 suggesting that the default Slave address has not been changed.

5.7 Configuring Slave Channel for writing on sensor

As mentioned earlier, the Modbus RTU protocol also allows the Master to write values into the Slave, in case the Slave is capable of accepting new values. The sensor on the Training Kit, according to Figure 28, keeps registers which can be written on. For example, we can set a temperature correction value up to 10 degrees or a humidity correction value up to 10%.
These corrections simply add the written value into the measured value. For instance, the sensor measures 20 degrees and 40% humidity. A correction value of 5 (in CODESYS depicted as 50) would result in the temperature showing as 25 degrees Celsius and a humidity of 45%.

1. Follow the steps as explained before in: Configuring Slave Channel for reading sensor data to create a new channel.

2. Below, we can see the channel configuration for writing on the sensor. To write on the register, function code 6 is used. Furthermore, we are addressing the offset "*0x0103*" which is the temperature correction for our sensor (see: Figure 28: Register table for Training Kit sensor).

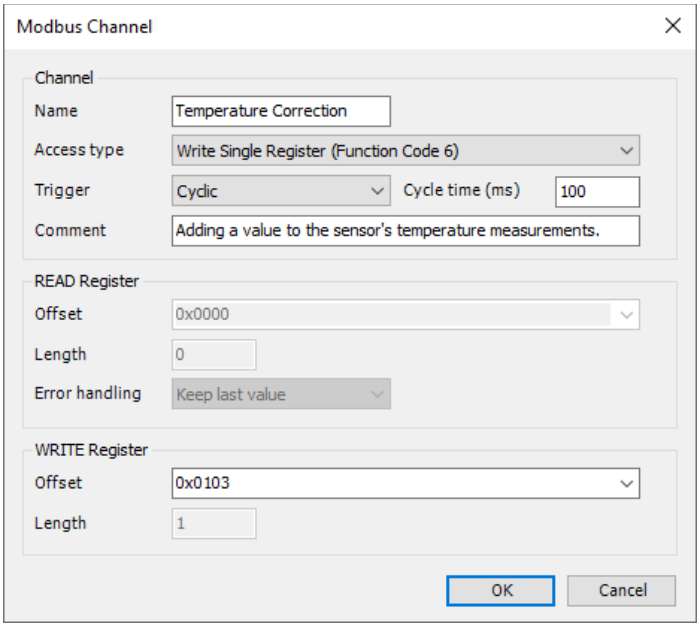
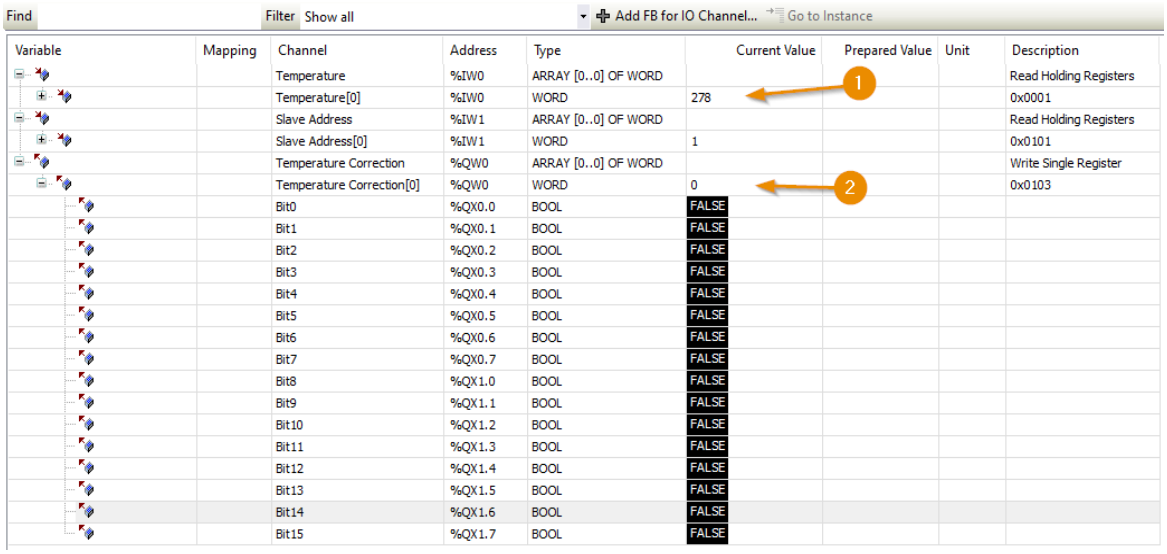


Figure 32: Slave Channel for writing on sensor's temperature correction

5.8 Writing on sensor

Of course, we want to write values to see how they affect the sensor's input measurements and to possibly correct any false measurements (e.g.: a temperature sensor is next to a machine running hot, which affects the measurements and does not depict the temperature correctly).

1. First, we want to navigate to the Modbus I/O Mapping again. This is done the same way as in chapter 5.6 Reading sensor data.
   Here, we can see that the sensor is currently measuring 27.8 degrees Celsius and that the temperature correction is 0, with the bits set to "*FALSE*".



Variable	Mapping	Channel	Address	Type	Current Value	Prepared Value	Unit	Description
		Temperature	%IW0	ARRAY [0..0] OF WORD				Read Holding Registers
		Temperature[0]	%IW0	WORD	278			0x0001
		Slave Address	%IW1	ARRAY [0..0] OF WORD				Read Holding Registers
		Slave Address[0]	%IW1	WORD	1			0x0101
		Temperature Correction	%QW0	ARRAY [0..0] OF WORD				Write Single Register
		Temperature Correction[0]	%QW0	WORD	0			0x0103
		Bit0	%QX0.0	BOOL	FALSE			
		Bit1	%QX0.1	BOOL	FALSE			
		Bit2	%QX0.2	BOOL	FALSE			
		Bit3	%QX0.3	BOOL	FALSE			
		Bit4	%QX0.4	BOOL	FALSE			
		Bit5	%QX0.5	BOOL	FALSE			
		Bit6	%QX0.6	BOOL	FALSE			
		Bit7	%QX0.7	BOOL	FALSE			
		Bit8	%QX1.0	BOOL	FALSE			
		Bit9	%QX1.1	BOOL	FALSE			
		Bit10	%QX1.2	BOOL	FALSE			
		Bit11	%QX1.3	BOOL	FALSE			
		Bit12	%QX1.4	BOOL	FALSE			
		Bit13	%QX1.5	BOOL	FALSE			
		Bit14	%QX1.6	BOOL	FALSE			
		Bit15	%QX1.7	BOOL	FALSE			

Figure 33: Temperature before writing on sensor

2. To add a temperature correction, we can add a value between 0 to 10 degrees (which correlates to 0-100 in CODESYS as explained beforehand). In our example, we are writing a value of 7 degrees (correlates to 70) into the sensor, meaning that the temperature should be at around 34.8 afterwards.
   To do this, double-click into the field next to "*Current value*", which has the 0 in it, called "*Prepared Value*". Enter the value you want to add, as mentioned, we will put in 70.

Variable	Mapping	Channel	Address	Type	Current Value	Prepared Value
		Temperature	%IW0	ARRAY [0..0] OF WORD		
		Temperature[0]	%IW0	WORD	278	
		Slave Address	%IW1	ARRAY [0..0] OF WORD		
		Slave Address[0]	%IW1	WORD	1	
		Temperature Correction	%QW0	ARRAY [0..0] OF WORD		
		Temperature Correction[0]	%QW0	WORD	0	70
		Bit0	%QX0.0	BOOL	FALSE	
		Bit1	%QX0.1	BOOL	FALSE	
		Bit2	%QX0.2	BOOL	FALSE	
		Bit3	%QX0.3	BOOL	FALSE	
		Bit4	%QX0.4	BOOL	FALSE	
		Bit5	%QX0.5	BOOL	FALSE	
		Bit6	%QX0.6	BOOL	FALSE	
		Bit7	%QX0.7	BOOL	FALSE	
		Bit8	%QX1.0	BOOL	FALSE	
		Bit9	%QX1.1	BOOL	FALSE	
		Bit10	%QX1.2	BOOL	FALSE	
		Bit11	%QX1.3	BOOL	FALSE	
		Bit12	%QX1.4	BOOL	FALSE	
		Bit13	%QX1.5	BOOL	FALSE	
		Bit14	%QX1.6	BOOL	FALSE	
		Bit15	%QX1.7	BOOL	FALSE	

Figure 34: Inserting new temperature correction

3. To fully apply the prepared value, use the hotkeys "*CTRL + F7*", which will write the prepared value into the current value field. But we can also go to "*Debug*" up top and click on "*Write Values*" to write the prepared value into the current value.

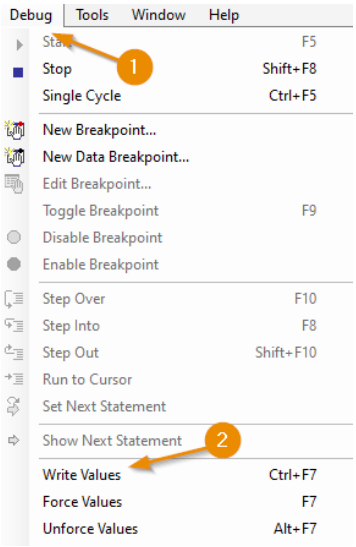
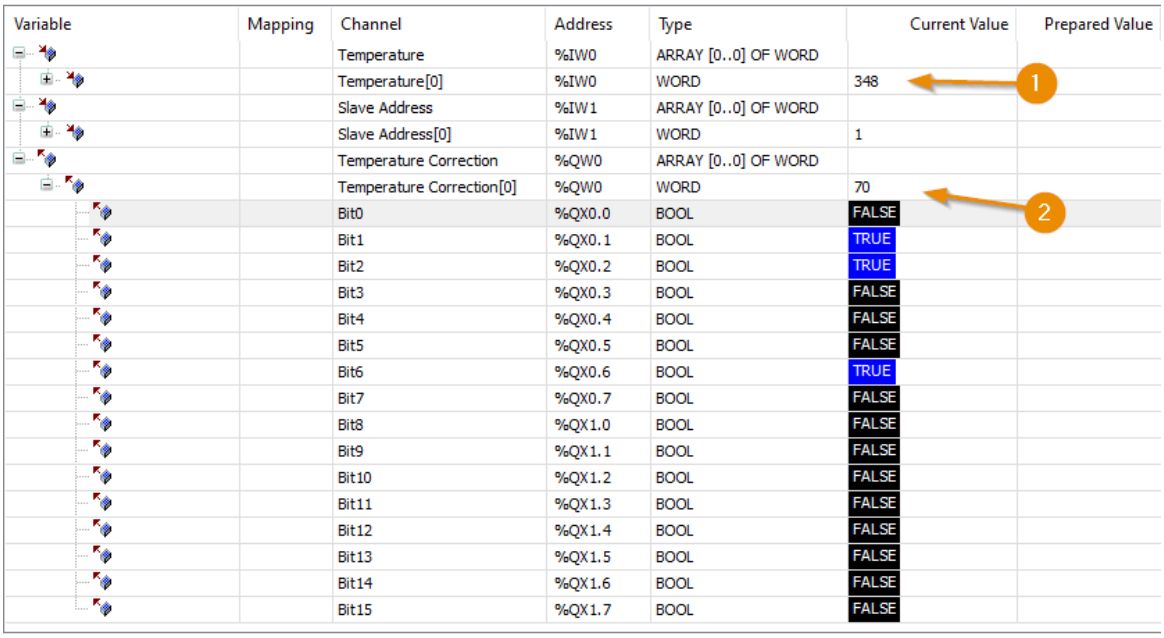


Figure 35: Writing value into the current value

4. After applying the prepared value, the temperature measures now 34.8 as pre-calculated. One can also see the corresponding bits to represent the number 70 are set to "*TRUE"*.



Variable	Mapping	Channel	Address	Type	Current Value	Prepared Value
		Temperature	%IW0	ARRAY [0..0] OF WORD		
		Temperature[0]	%IW0	WORD	348	
		Slave Address	%IW1	ARRAY [0..0] OF WORD		
		Slave Address[0]	%IW1	WORD	1	
		Temperature Correction	%QW0	ARRAY [0..0] OF WORD		
		Temperature Correction[0]	%QW0	WORD	70	
		Bit0	%QX0.0	BOOL	FALSE	
		Bit1	%QX0.1	BOOL	TRUE	
		Bit2	%QX0.2	BOOL	TRUE	
		Bit3	%QX0.3	BOOL	FALSE	
		Bit4	%QX0.4	BOOL	FALSE	
		Bit5	%QX0.5	BOOL	FALSE	
		Bit6	%QX0.6	BOOL	TRUE	
		Bit7	%QX0.7	BOOL	FALSE	
		Bit8	%QX1.0	BOOL	FALSE	
		Bit9	%QX1.1	BOOL	FALSE	
		Bit10	%QX1.2	BOOL	FALSE	
		Bit11	%QX1.3	BOOL	FALSE	
		Bit12	%QX1.4	BOOL	FALSE	
		Bit13	%QX1.5	BOOL	FALSE	
		Bit14	%QX1.6	BOOL	FALSE	
		Bit15	%QX1.7	BOOL	FALSE	

Figure 36: Temperature after writing the values

6 Results

We are now able to read and write any Modbus RTU sensor data with CODESYS using our Modbus Gateway to transform the data into a readable format. This is helpful as we can use this application on any Modbus RTU sensor for reading and writing its data and we can use this single application to read and write on multiple sensors at once if we have more than one connected in our field bus. Furthermore, we now understand the Modbus protocol and can differentiate between the sensor's registers like the input and keep registers. Besides, we can set up a CODESYS project accordingly.