

Simplified Motion Series Control with Festo PLC through Modbus using Festo CPX-AP IO-Link Master

This Application Note explains about Simplified Motion Series actuator units (EPCE, EPCS-BS, EGSS, ELGS-BS, ELGS-TB, ELGE, ERMS) IN/OUT Control, Read/Write Parameters & Read Diagnostic Data. Function blocks are tested and released for Festo CPX-AP IO Link Master. Use with other IO-Link Masters not tested and on own responsibility.

This library is intended for common use of CPX-AP-I IO Link Master and CPX-AP-A IO Link Master.

EPCE
EPCS-BS
EGSS
ELGS-BS
ELGS-TB
ELGE
ERMS

Title Simplified Motion Series Control With Festo PLC by Modbus using Festo CPX-AP IO-Link Master
Version 1.40
Document no. 100350
Originalen
AuthorFesto

Last saved 25.09.2023

Copyright Notice

This documentation is the intellectual property of Festo SE & Co. KG, which also has 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 Notice

Hardware, software, operating systems and drivers may only be used for the applications described and only in conjunction with 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 relating to the protection of personnel and machinery.

No liability is accepted for claims for damages arising from a failure or functional defect. In other respects, the regulations with regard to liability from the terms and conditions of delivery, payment and use of software of Festo SE & Co. KG, which can be found at www.festo.com and can be supplied on request, shall apply.

All data contained in this document do not represent guaranteed specifications, particularly with regard to functionality, condition or quality, in the legal sense.

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 user.

The operating instructions for Festo products can be found at www.festo.com.

Users of this document (application note) must verify that all functions described here also work correctly in the application. By reading this document and adhering to the specifications contained therein, users are also solely responsible for their own application.

Table of contents

1	Introduction	5
1.1	Hardware & Software Requirements	7
1.2	Hardware Overview	7
1.3	Topology Overview of Simplified Motion Series with CPX-AP-I-EP	8
1.4	Topology Overview of Simplified Motion Series with CPX-AP-A-EP	9
1.5	Festo CPX-E-CEC-M1-EP PLC IP Address Configuration	10
1.6	Festo CPX-AP-I-4IO-M12 IO-Link Master Port Configuration for Activation	12
2	Software Configuration.....	16
2.1	Ethernet Module Configuration in Festo CODESYS	16
2.2	Adding Modbus Slave channel	19
2.3	MasterTCPSlave I/O Mapping	23
2.3.1	Method – 1: Link the I/O address using Global variable	23
2.3.2	Method – 2: Link the I/O address using Program variable	24
2.4	Import “Festo_SMS_ModbusTCP_Library” into CODESYS Runtime Project.....	26
2.5	Add “Festo_SMS_ModbusTCP” Library in CODESYS Library Manager.....	28
3	Festo SMS Modbus Library Function Blocks	30
3.1	Introduction to “SMS_Festo_Advanced” Function Block.....	30
3.1.1	SMS_Festo_Advanced block IO interface Tag Details	30
3.1.2	SMS_Festo_Advanced block Control Tag Details	31
3.1.3	SMS_Festo_Advanced block Monitor Tag Details	33
3.2	Introduction to “SMS_Festo_Basic” Function Block.....	35
3.2.1	SMS_Festo_Basic block IO interface Tag Details	35
3.2.2	SMS_Festo_Basic block Control Tag Details	36
3.2.3	SMS_Festo_Basic block Monitor Tag Details	38
3.3	Introduction to “SMS_Festo_Intermediate” Function Block	39
3.3.1	SMS_Festo_Intermediate block IO interface Tag Details	40
3.3.2	SMS_Festo_Intermediate block Control Tag Details	40
3.3.3	SMS_Festo_Intermediate block Monitor Tag Details	41
3.4	Introduction to “SMS_Festo_IO” Function Block.....	42
3.4.1	SMS_Festo_IO block IO interface Tag Details	43
3.4.2	SMS_Festo_IO block Control Tag Details	43
3.4.3	SMS_Festo_IO block Monitor Tag Details	43
3.5	Introduction to “SMS_Festo_Parameter” Function Block.....	45
3.5.1	SMS_Festo_Parameter block IO interface Tag Details	45
3.5.2	SMS_Festo_Parameter block Control Tag Details	45
3.5.3	SMS_Festo_Parameter block Monitor Tag Details	48
3.5.4	Diagnosis code details of Simplified Motion Series	49

4	Configuring Channel Input and Output in CODESYS	51
4.1	Configuring the RefInput & RefOutput Interface Tags.....	51
5	Configuration of Festo_SMS_Modbus Function blocks in CODESYS	55
5.1	Configuring the SMS_Festo_Advanced Function Block Input & Output Interface Tags.....	55
5.2	Configuring the SMS_Festo_Basic Function Block Input & Output Interface Tags.....	61
5.3	Configuring the SMS_Festo_Intermediate Function Block Input & Output Interface Tags	67
5.4	Configuring the SMS_Festo_IO Function Block Input & Output Interface Tags.....	73
5.5	Configuring the SMS_Festo_Parameter Function Block Input & Output Interface Tags.....	78
6	Important Operating Details of SMS	84
6.1	Control Tag Operation details.....	84
6.2	Position Value Operation.....	84
7	Sample Code	85
7.1	Initial Setup	85
7.2	Actuator unit Homing with stroke detection	87
7.3	Actuator unit Move to Inwards	89
7.4	Actuator unit Move to Outwards.....	90
7.5	Actuator unit Move to Intermediate.....	91
7.6	Sample Code Description for Move The Axis to Multiple Intermediate positions	92
8	Technical appendix.....	94
8.1	Find Modbus Address Details for Simplified Motion Series actuator unit Connected Port	94

1 Introduction

This application note demonstrate the controlling of Simplified Motion Series like EPCE, EPCS-BS, EGSS, ELGS-BS, ELGS-TB, ELGE, ERMS with Festo PLC through Festo CPX-AP IO-Link Master. The Library has five function block variants based on its function.

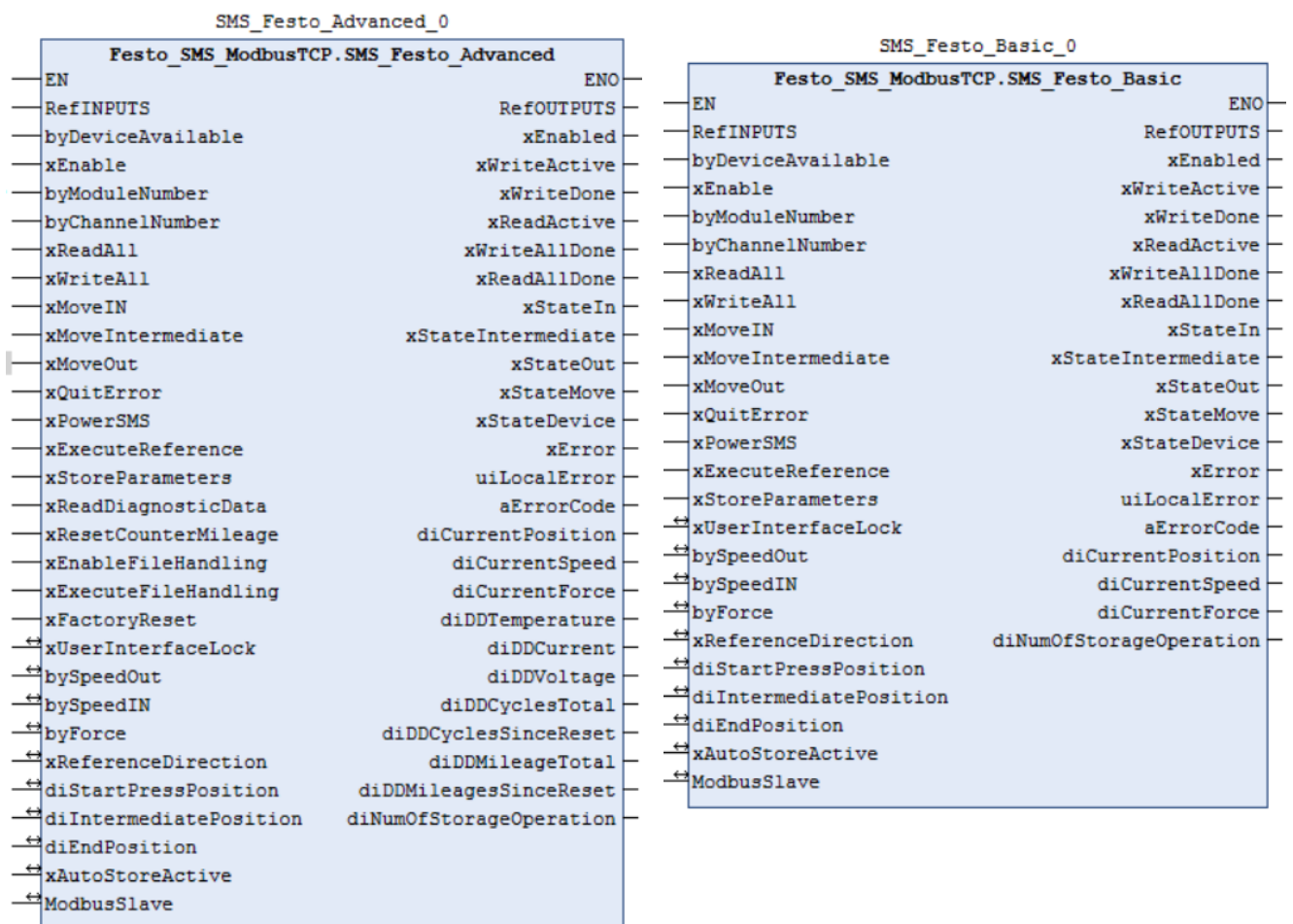
The “**SMS_Festo_Advance**” includes to access all advanced parameters and it is capable to control the motion system like IN & OUT function, Error reset, Enable power stage of drive, Parameters read & write, Also to read the diagnostic data from the drive unit.

The “**SMS_Festo_Basic**” has access to few mandatory parameters and process data parameters.

The “**SMS_Festo_Intermediate**” has basic function to control the actuator IN & Out movements and also access the Intermediate function Parameters.

The “**SMS_Festo_IO**” has only basic function to control the actuator IN & Out movements.

The “**SMS_Festo_Parameter**” has only the acyclic parameters read and write function.



**Note**

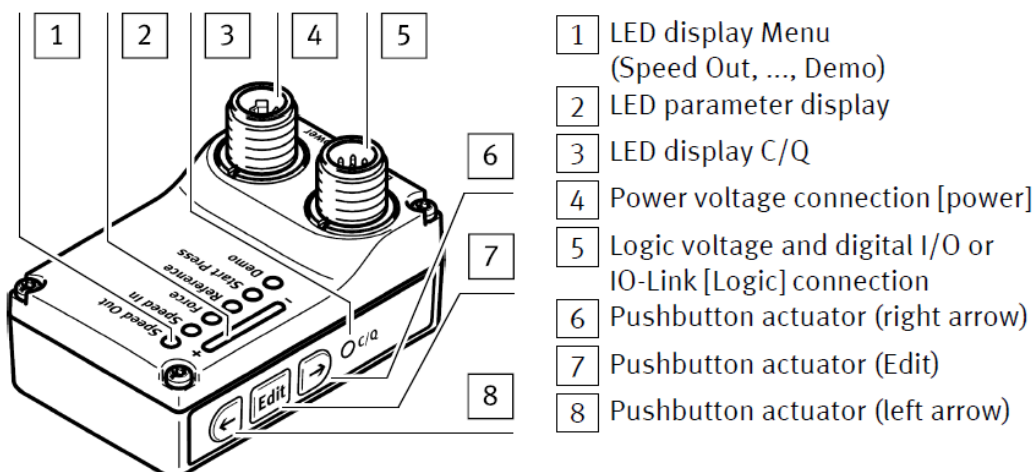
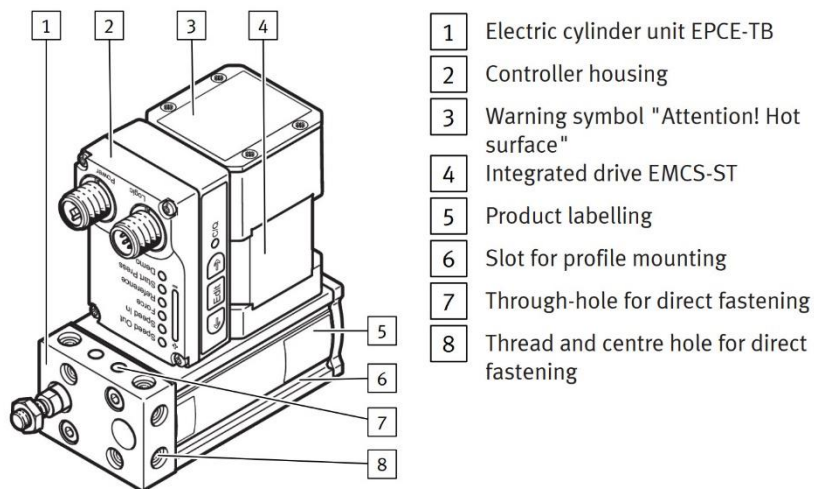
- This application note shows how to use function block with Festo CPX-AP IO-Link Master. The function blocks are tested and released for this IO-Link Master.
- Compatibility with other IO-Link Masters is not tested and released. Use of function blocks with different IO-Link masters is on own responsibility.
- This Application note is common for both CPX-AP-I and CPX-AP-A IO Link Masters. So it is mentioned as **“CPX-AP IO Link Masters”**.

1.1 Hardware & Software Requirements

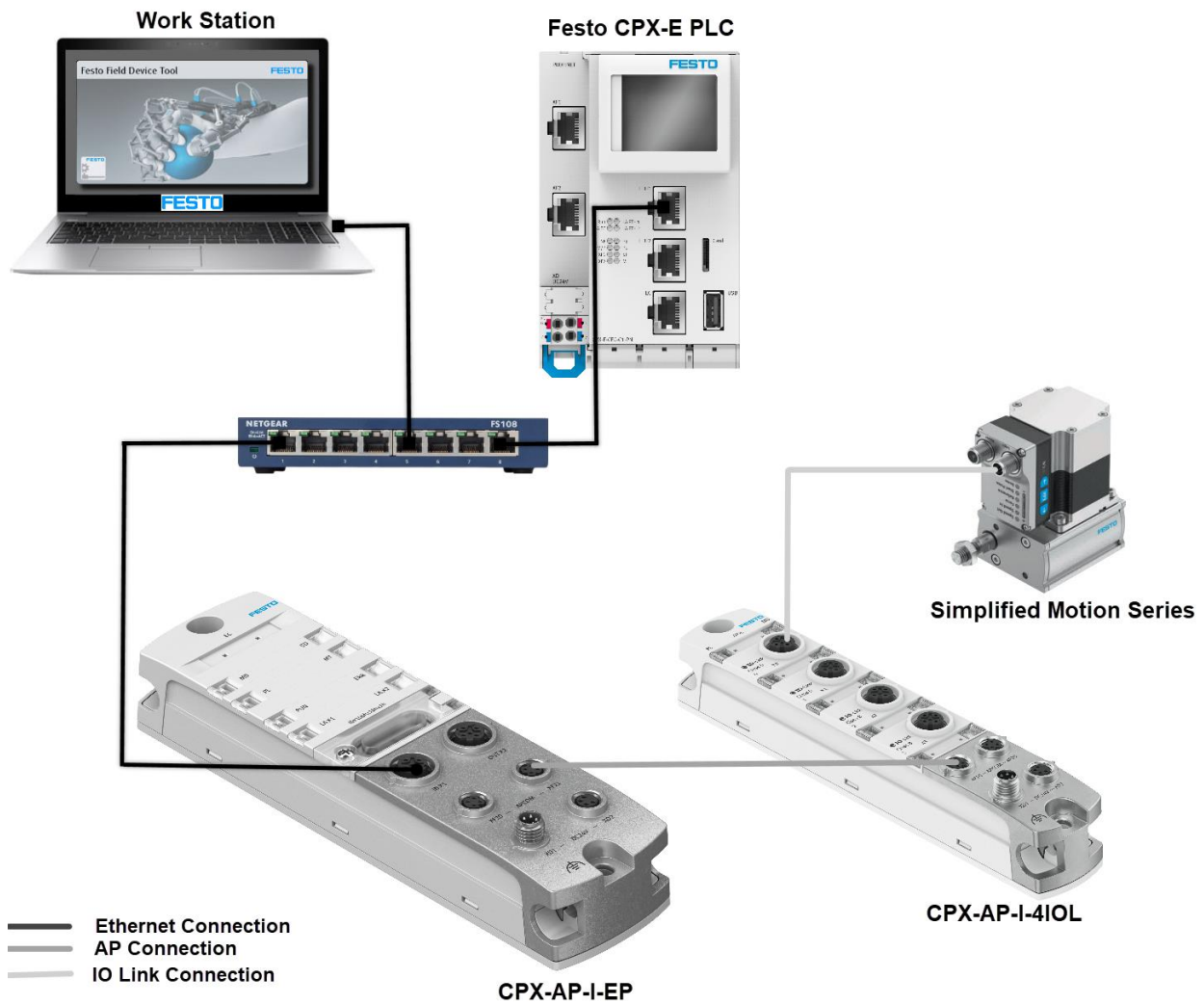
Device Name	Part Number	Version Software/Firmware	Device Type
Any Simplified Motion Series (EMCS-ST, EGSS-BS, ELGS-TB/-BS, ELGS-TB, ERMS, EPCS-BS & EPCE-TB)	-	Latest	Hardware
Festo IO-Link Master	CPX-AP-I-4IOL-M12	Latest	Hardware
Festo CPX-AP-I Head Module	CPX-AP-I-EC-M12	Latest	Hardware
CPX Module Connecting Cable	NEBC-D8G4-ES-0.3-N-S-D8G4-ET	-	Hardware
CPX Module Power Cable	NEBL-M8G4-E-5-N-LE4	-	Hardware
Festo PLC CPX-E-CEC	-	Version Free	Hardware
Festo SMS V2.2 Library	-	SMS V2.2	Software
Festo CODESYS	-	V3.5 SP15 Patch 4 and above	Software
Festo Automation Suite	-	Latest	Software
Internet Explorer	-	Latest	Software

Table 1.1: 1 Components/Software used

1.2 Hardware Overview



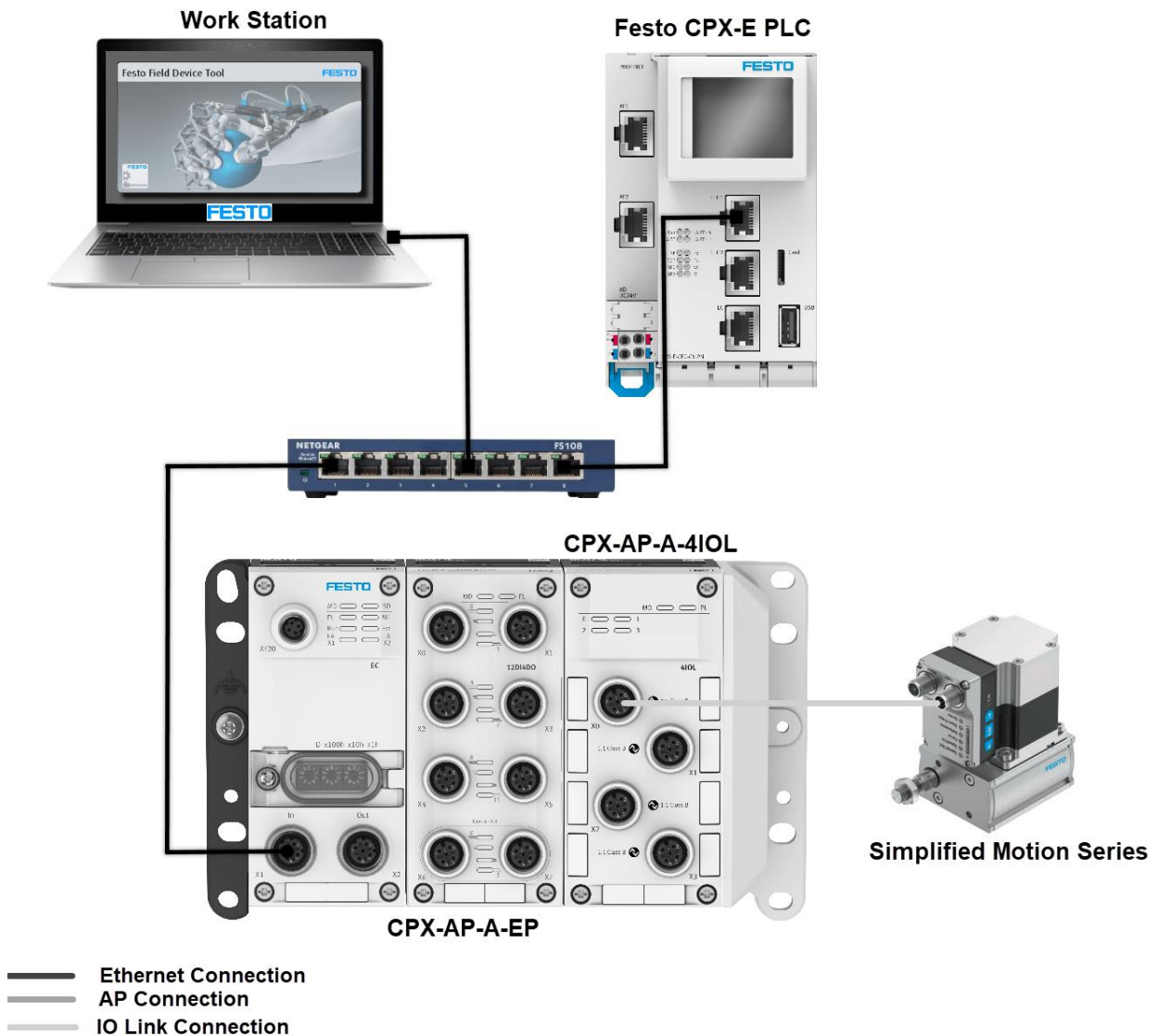
1.3 Topology Overview of Simplified Motion Series with CPX-AP-I-EP



Note

- Topology overview may change depend on user CPX-AP-I expansion modules. The above structure shown for demonstration.

1.4 Topology Overview of Simplified Motion Series with CPX-AP-A-EP



Note

- Topology overview may change depend on user CPX-AP-A expansion modules. The above structure shown for demonstration.

1.5 Festo CPX-E-CEC-M1-EP PLC IP Address Configuration

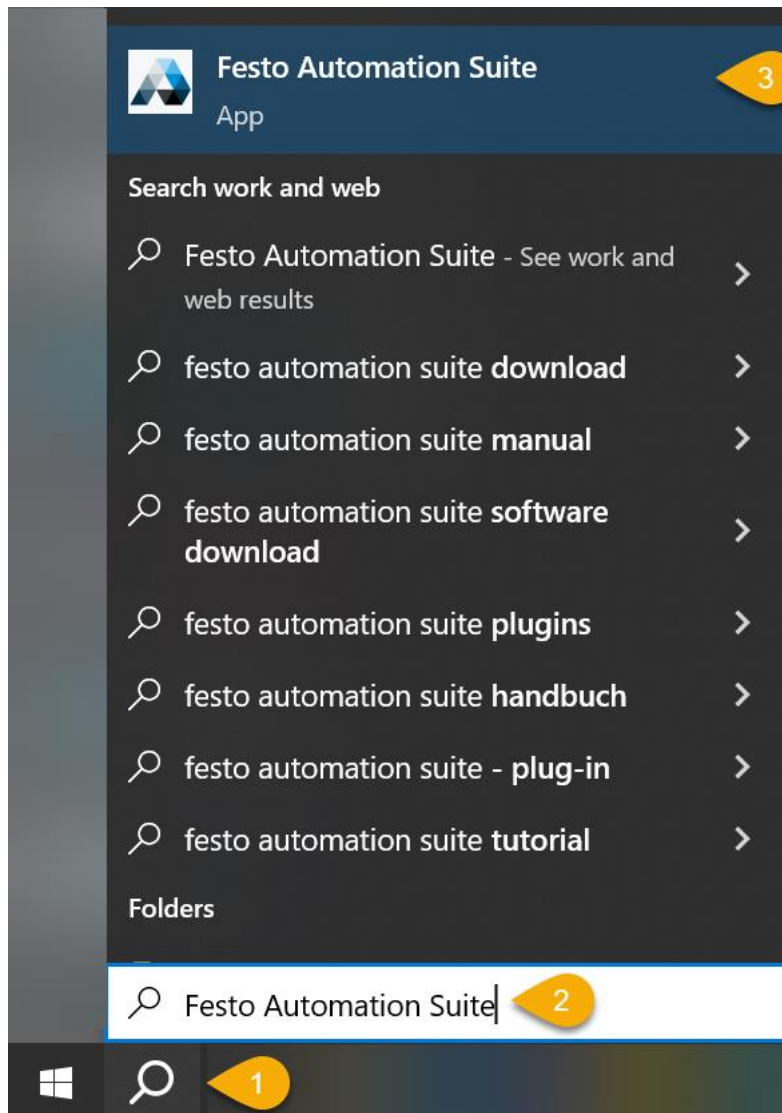


Note

The IP address of the PLC can be configured in two methods. One is using the rotary switch for 192.168.1.xxx series & another one is using the Festo field device tool for any IP series.

This chapter explains how to configure the IP address of the CPX-E-CEC-M1-EP PLC using the Festo Field Device Tool.

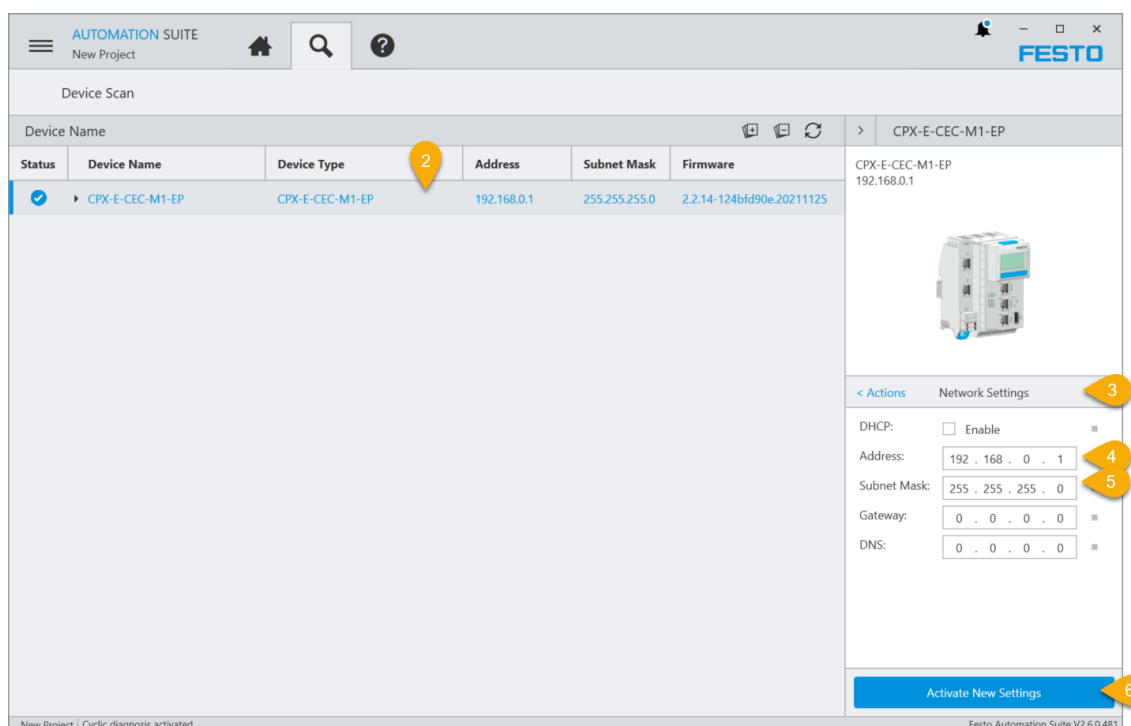
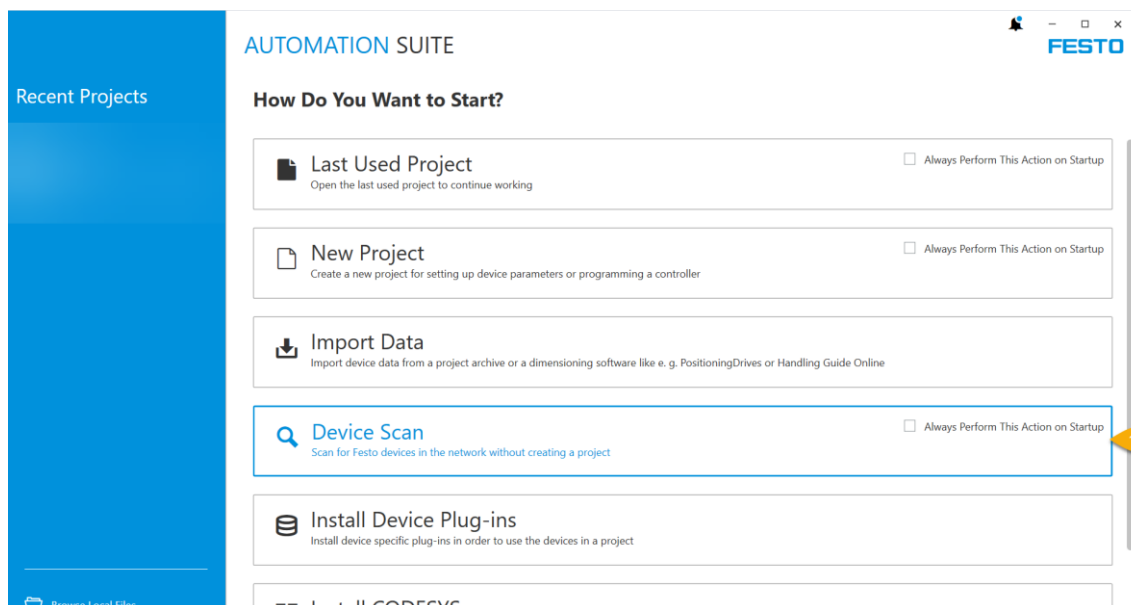
Step – 1



1. Click the **“Search”** button on Taskbar.
2. Enter **“Festo Automation Suite”** in search box or user can navigate to **Start > Festo Software > Festo Automation Suite**
3. Click the software to open it.

Step – 2**Note**

- Make sure the PLC and work station communication cables are connected as shown in [Chapter – 1.3.](#)



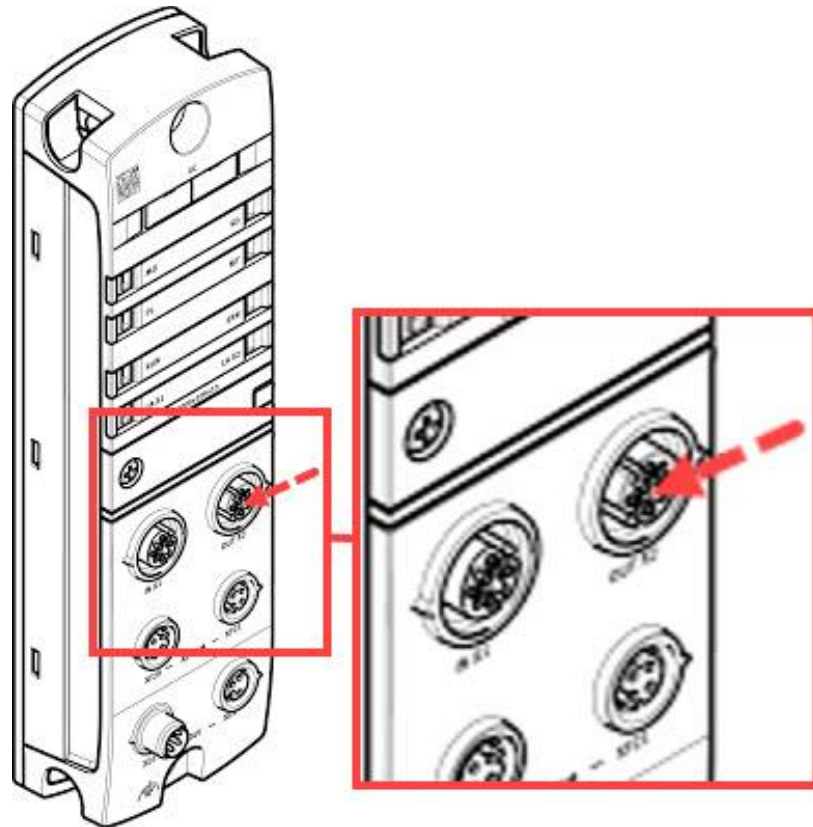
- Select the **“Device Scan”**.
- Select the CPX-E-CEC-M1-EP PLC from the connected device list in Festo Automation Suite.
- Select **“Network Settings”** option from the right side menu.
- Enter the **“IP address”** of the PLC.
- Enter the **“Subnet Mask”** of the PLC.
- Click **“Activate New Settings”** to confirm the configuration.

**Note**

- After the new IP address configuration, power restart the PLC once to activate the new static IP address.

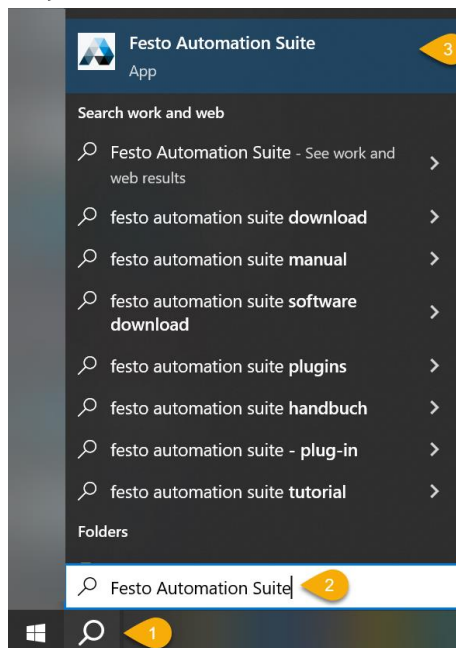
1.6 Festo CPX-AP-I-4IO-M12 IO-Link Master Port Configuration for Activation

Step – 1



Connect the communication cable in X2 port. Make sure the communication cable is directly connected with workstation & avoid ethernet hub's.

Step – 2



1. Click the **“Search”** button on Taskbar.
2. Enter **“Festo Automation Suite”** in search box or user can navigate to **Start > Festo Software > Festo Automation Suite**
3. Click the software to open it.

Step – 3

AUTOMATION SUITE

How Do You Want to Start?

- Last Used Project** (Always Perform This Action on Startup)
- New Project** (Always Perform This Action on Startup)
- Import Data**
- Device Scan** (Always Perform This Action on Startup) 1
- Install Device Plug-ins**
- Install CODESYS**

Device Scan

Status	Device Name	Device Type	Address	Subnet Mask	Firmware
i	ap_i_ec	AP-I-EC	169.254.29.16	255.255.0.0	1.6.3-8b3954211.20230508

Network Settings 3

DHCP: ☐ Enable

Address: 192 . 168 . 0 . 2 4

Subnet Mask: 255 . 255 . 255 . 0 5

Gateway: 0 . 0 . 0 . 0

DNS: 0 . 0 . 0 . 0

Activate New Settings 6

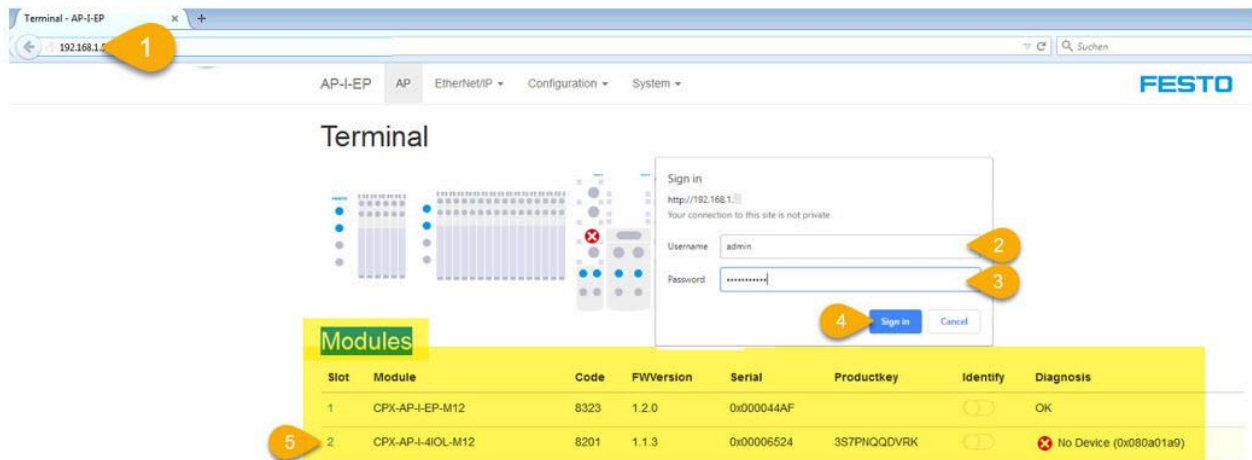
1. Select the **“Device Scan”**.
2. Select the CPX-AP-I-EP from the connected device list in Festo Automation Suite.
3. Select **“Network Settings”** option from the right side menu.
4. Enter the **“IP address”** of the Module.
5. Enter the **“Subnet Mask”** of the Module.
6. Click **“Activate New Settings”** to confirm the configuration.

**Note**

- After the new IP address configuration don't power restart the CPX-AP-I module to avoid the new dynamic IP address.

Step – 4

The web server configuration shows connection details of connected modules, Description of module, Firmware details, Serial number & Product key. As per this application note we have connected two number of modules.



1. Open internet browser and type the IP address of the CPX-AP-I-EP module.
2. Enter the “**Username**” as admin in Sign-in pop-up window.
3. Enter the CPX-AP-I-EP product key as password. Product key is available in module name plate.
4. Press “**Sign in**” button on Sign in pop-up window.
5. Select the CPX-AP-I-4IOL-M12 IO-Link module where the SMS control unit is connected.

Step – 5

User has to do the below configurations for the IO-Link port where SMS control unit is connect. Assign the parameter as like below green highlighted to activate an IO-Link port.

2	CPX-AP-I-4IOL-M12	8206	1.5.6	0x000DDA3	3S7PP8J63CB		OK
---	-------------------	------	-------	-----------	-------------	--	----

AP Id/Instance	Parameter	Startup	Value
20022:0	Setup monitoring load supply (PL) 24 V DC	yes	Load supply monitoring active, diagnosis suppressed in case of switch-off
20049:0	Nominal Cycle Time (Port 0)	yes	as fast as possible
20049:1	Nominal Cycle Time (Port 1)	yes	as fast as possible
20049:2	Nominal Cycle Time (Port 2)	yes	as fast as possible
20049:3	Nominal Cycle Time (Port 3)	yes	as fast as possible
20050:0	Enable diagnosis of IO-Link device lost (Port 0)	yes	☒
20050:1	Enable diagnosis of IO-Link device lost (Port 1)	yes	☒
20050:2	Enable diagnosis of IO-Link device lost (Port 2)	yes	☒
20050:3	Enable diagnosis of IO-Link device lost (Port 3)	yes	☒
20071:0	Port Mode (Port 0)	yes	IO_LINK_AUTOSTART
20071:1	Port Mode (Port 1)	yes	DEACTIVATED
20071:2	Port Mode (Port 2)	yes	DEACTIVATED
20071:3	Port Mode (Port 3)	yes	DEACTIVATED
20072:0	Validation & Backup (Port 0)	yes	Type compatible Device V1.1, Backup + Restore
20072:1	Validation & Backup (Port 1)	yes	No Device check

Once port activation completed successfully feedback parameters update the port status as highlighted in yellow and pink highlighted parameters for inactive port feedback values.

35	20074:0	Port status information (Port 0)	OPERATE
36	20074:1	Port status information (Port 1)	DEACTIVATED
39	20075:0	Revision ID (Port 0)	17
40	20075:1	Revision ID (Port 1)	0
43	20076:0	Port transmission rate (Port 0)	COM2
44	20076:1	Port transmission rate (Port 1)	NOT_DETECTED
47	20077:0	Actual cycle time in 100 us (Port 0)	28
48	20077:1	Actual cycle time in 100 us (Port 1)	0
51	20078:0	Actual VendorID (Port 0)	310
52	20078:1	Actual VendorID (Port 1)	0
55	20079:0	Actual DeviceID (Port 0)	403
56	20079:1	Actual DeviceID (Port 1)	0
59	20108:0	InputDataLength (Port 0)	2
60	20108:1	InputDataLength (Port 1)	0

Select the variant as per the IO-Link device Inputs and outputs

20108:3	InputDataLength (Port 3)		CPX-AP-I-4IOL-M12 Variant 8
20109:0	OutputDataLength (Port 0)		CPX-AP-I-4IOL-M12 Variant 8 OE
20109:1	OutputDataLength (Port 1)		CPX-AP-I-4IOL-M12 Variant 2
20109:2	OutputDataLength (Port 2)		CPX-AP-I-4IOL-M12 Variant 2 OE
20109:3	OutputDataLength (Port 3)		CPX-AP-I-4IOL-M12 Variant 4
			CPX-AP-I-4IOL-M12 Variant 4 OE
			CPX-AP-I-4IOL-M12 Variant 16
			CPX-AP-I-4IOL-M12 Variant 16 OE
			CPX-AP-I-4IOL-M12 Variant 32
			CPX-AP-I-4IOL-M12 Variant 32 OE
20090:0	Variant selection	yes	CPX-AP-I-4IOL-M12 Variant 2



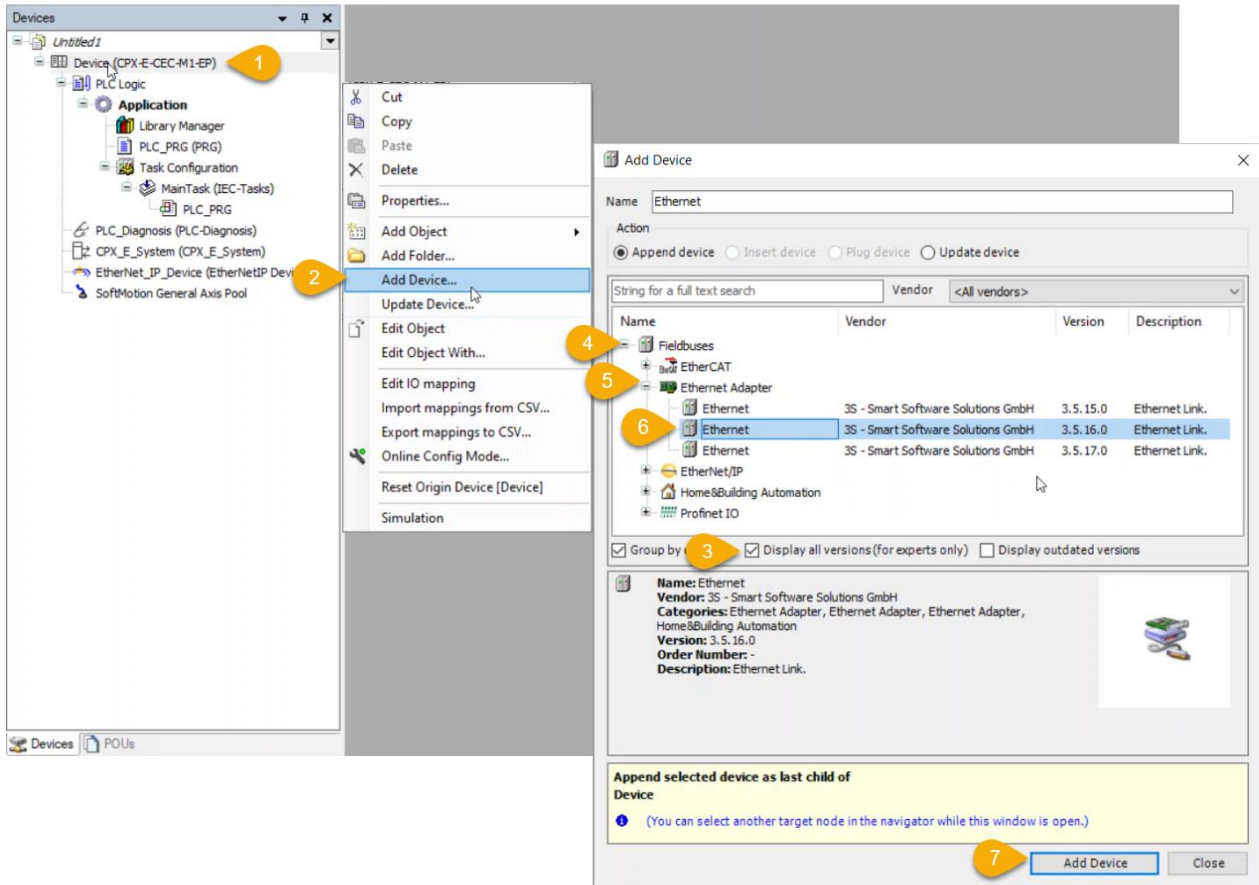
Note

- CPX-AP-I-4IOL- M12 Variant 2 has been selected since Simplified motion Series requires 2 Byte inputs and outputs.
- Variants ending with OE will not allow the host controller to control module outputs.

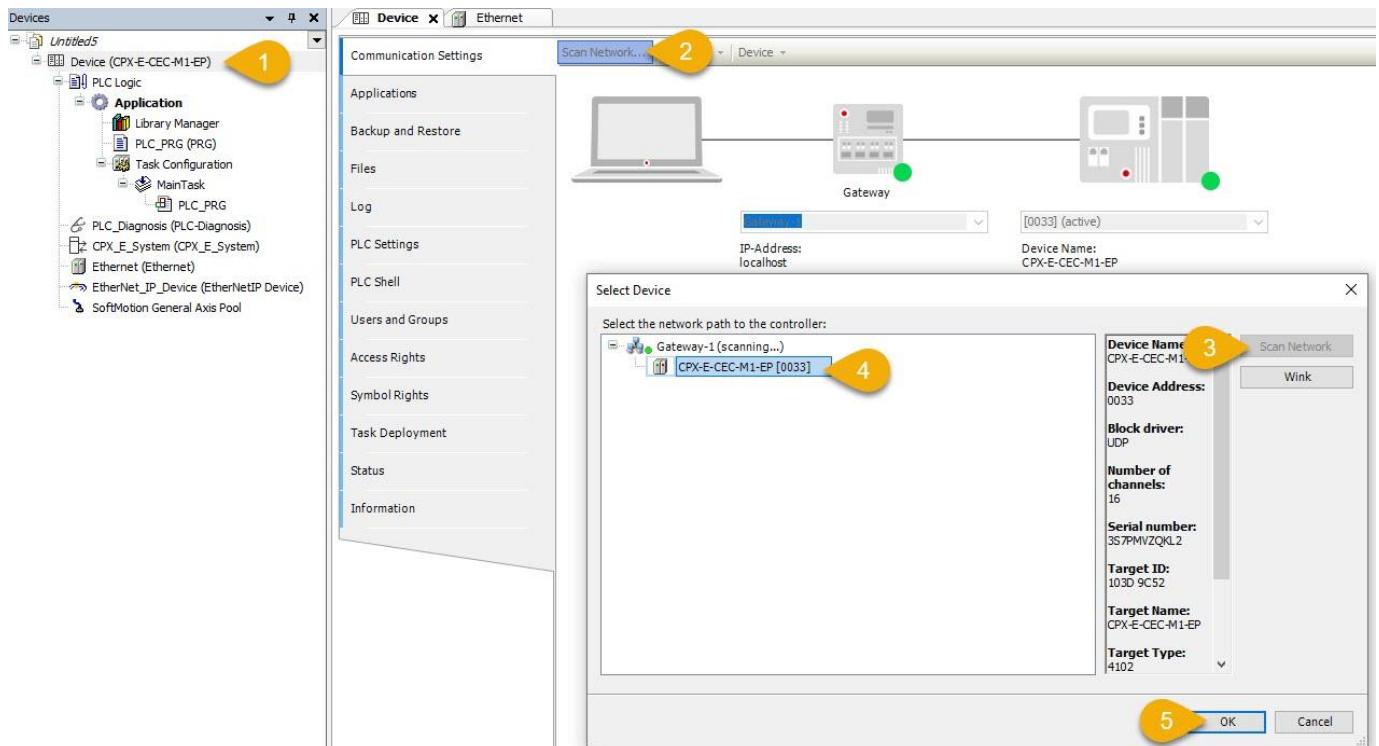
2 Software Configuration

2.1 Ethernet Module Configuration in Festo CODESYS

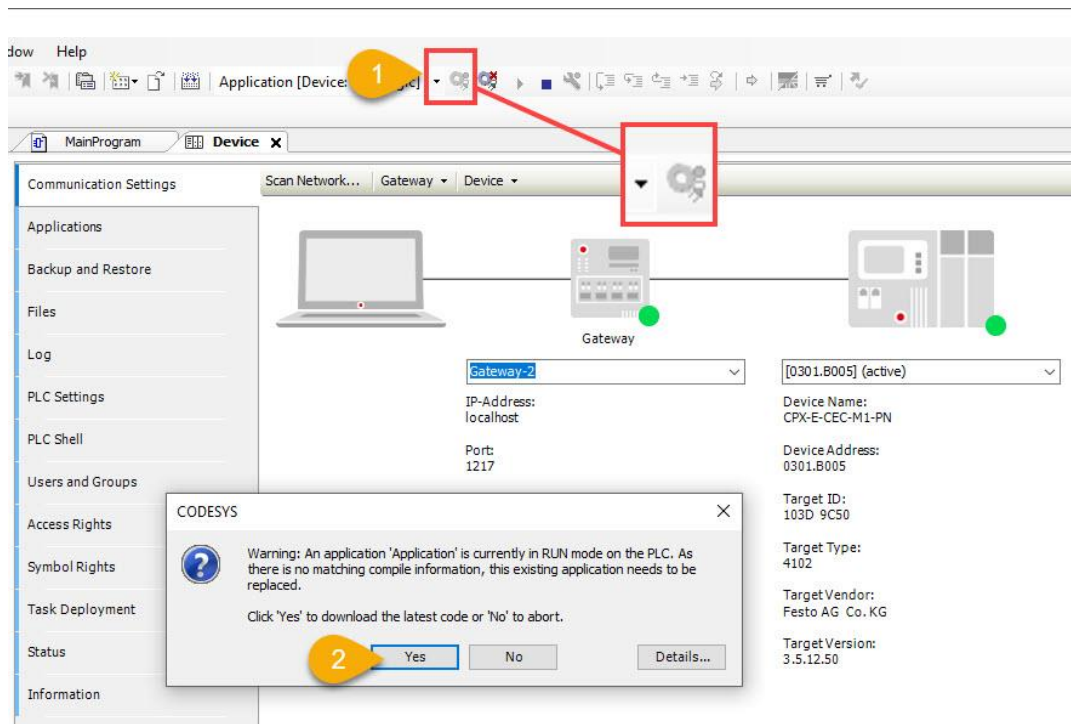
Step – 1



1. Select the **“Device”** folder with PLC model name & right click.
2. Select **“Add Device”** from the right click menu.
3. Select **“Display all versions(for experts only)”**
4. Explore the **“Fieldbuses”** folder from Add Device window.
5. Explore the **“Ethernet Adapter”**
6. Select **“Ethernet”** module from Ethernet Adapter. (subject to user's preferred version)
7. Click **“Add Device”** button from add device window.

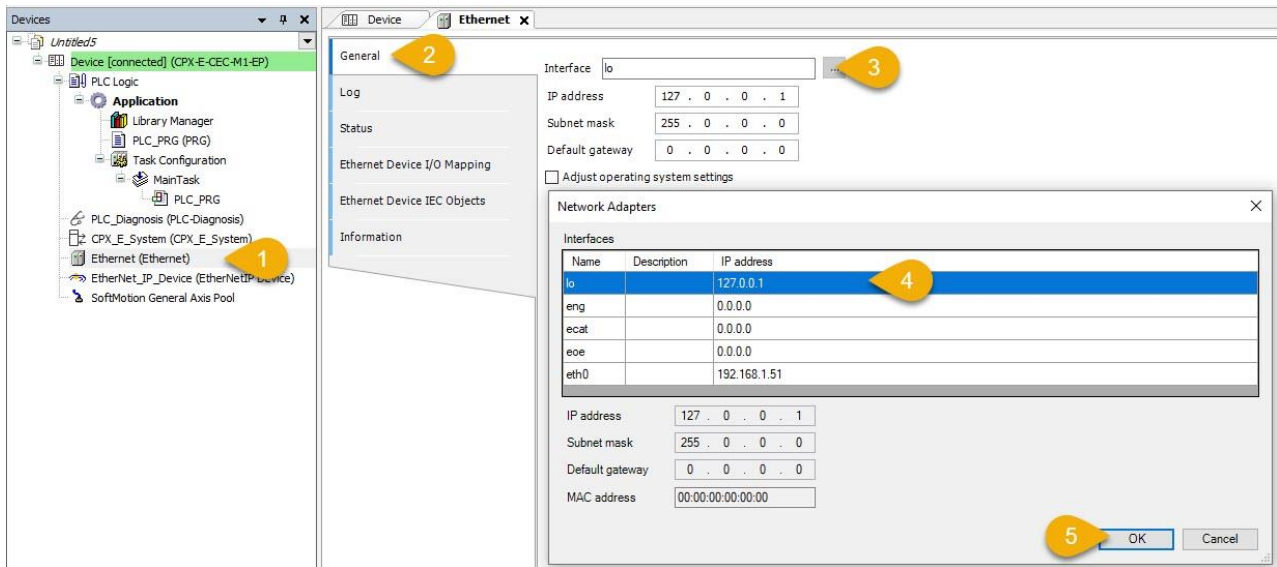
Step – 2

1. Double click the **“Device (CPX-E-CEC-M1-EP)”** from the device list.
2. Select the **“Scan Network”** from the Communication setting tab.
3. Click **“Scan Network”** from the select device window.
4. Select the active PLC from the scan list.
5. Click **“OK”** to confirm the selected PLC path.

Step – 3

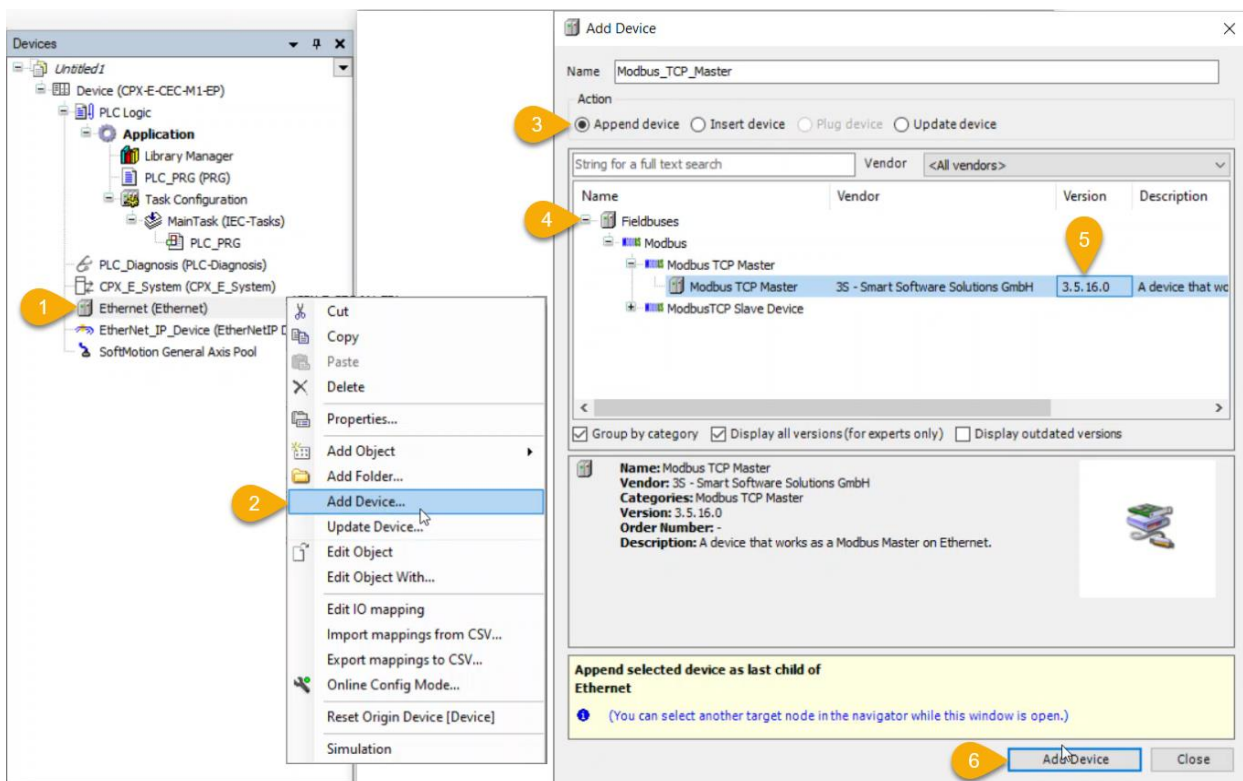
1. Click the **Log IN** button from tool bar or user can select the Log IN through short cut key Alt+F8.
2. Click **“Yes”** option on pop-up window.

Step – 4



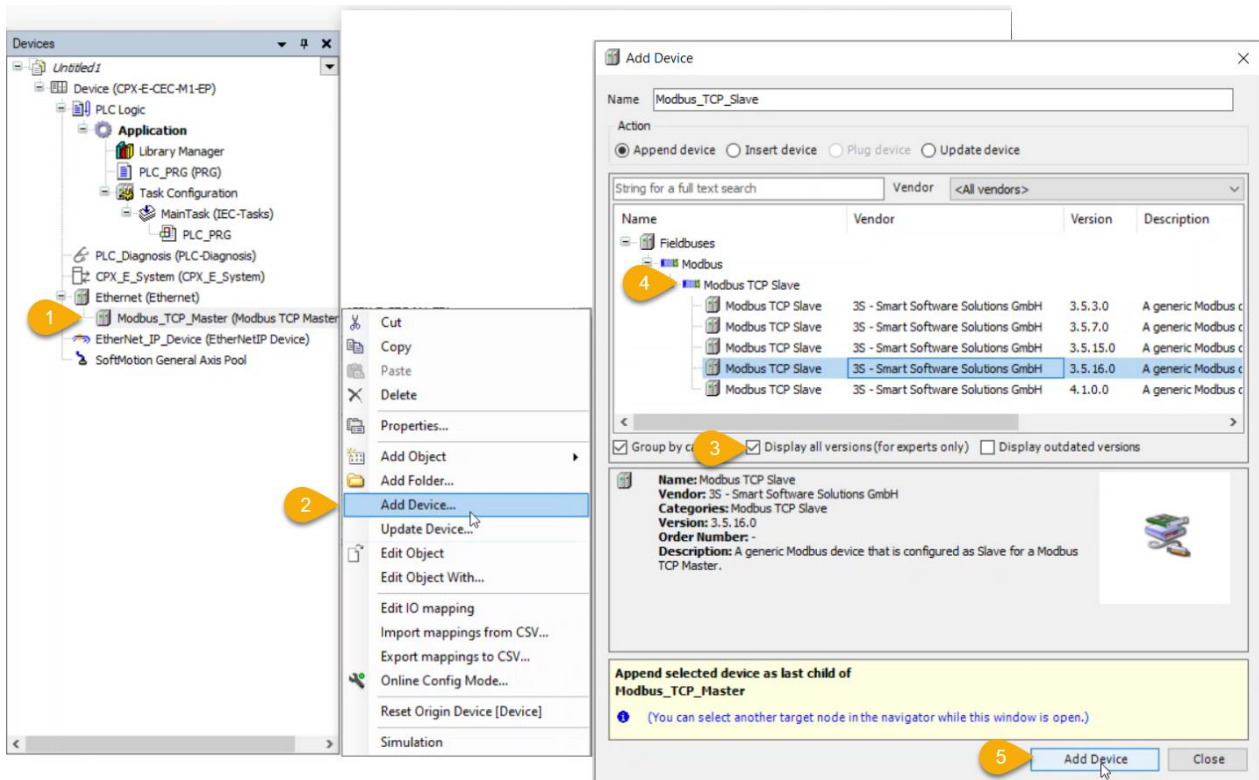
1. Double click the **“Ethernet”** module created in step – 1.
2. Click **“General”** in Ethernet window.
3. Select browse button of Interface option.
4. Select **“lo”** (Local Network Adapter) from network adapters window.
5. Click **“OK”** button to confirm the selection.

Step – 5



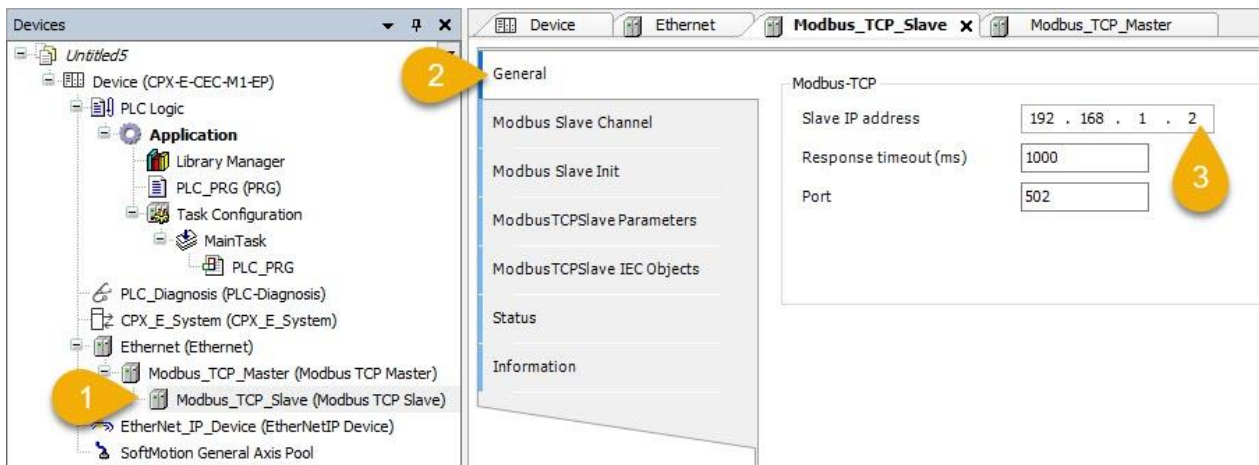
1. Right click the **“Ethernet”** module.
2. Select the **“Add Device”** option from right click menu.
3. Select the **“Append device”** from add device window.
4. Explore the **“Fieldbuses”** folder & explore the Modbus folder.
5. Select **“Modbus TCP Master”** module as shown in above picture.
6. Click **“Add Device”** button from add device window.

Step – 6



1. Select **"Modbus_TCP_Master"** module & right click which created in step – 5
2. Select **"Add Device"** from right click menu.
3. Select **"Display all versions(for experts only)"**
4. Explore the **"Modbus TCP Slave"** from fieldbuses list and select the "Modbus TCP Slave" module based on the Modbus TCP Master version.
5. Click **"Add Device"** to confirm the configuration.

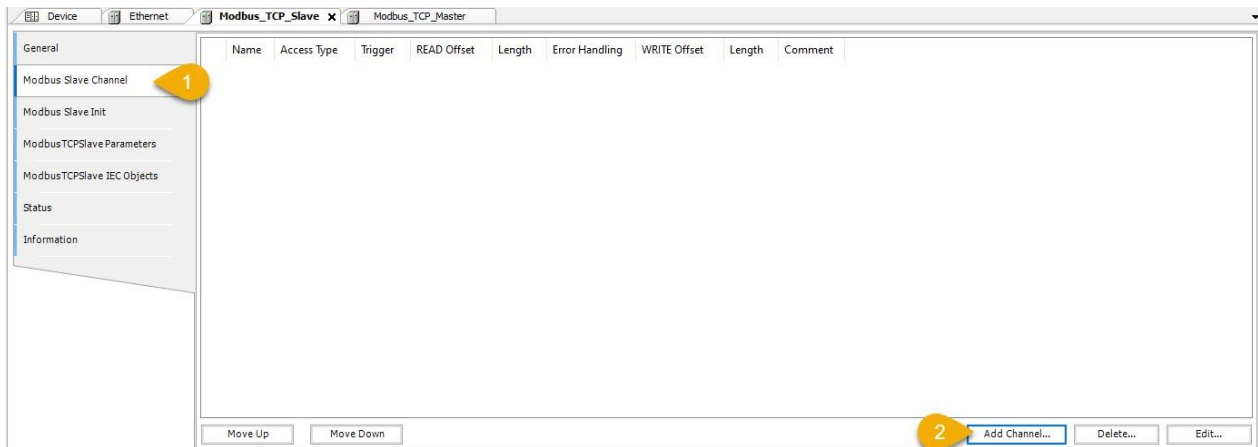
Step – 7



1. Double click the **"Modbus_TCP_Slave"** module which created in step – 6.
2. Select **"General"** tab.
3. Enter the CPX-AP-I-EP IP address in Slave IP address field.

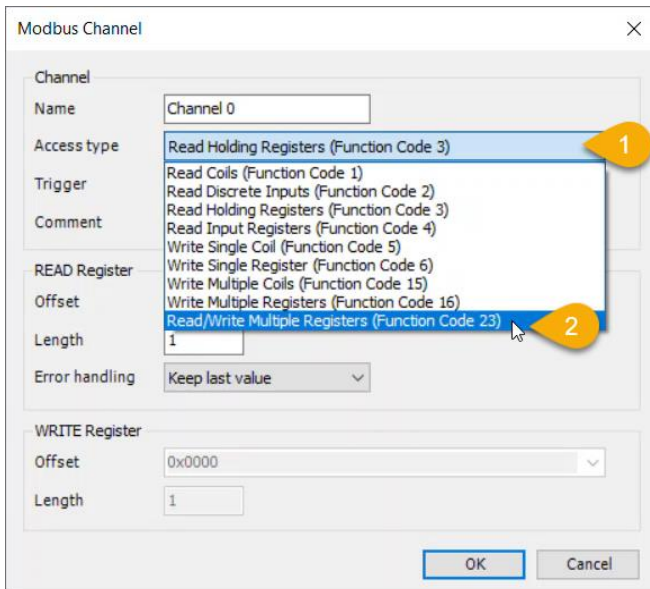
2.2 Adding Modbus Slave channel

Step – 1



1. Select “**Modbus Slave Channel**” from Modbus TCP Slave configuration window.
2. Click “**Add Channel**” button to add the new connection channel.

Step – 2



1. Click “**Access type**” drop-down menu.
2. Select “**Function Code 23**” for read/write multiple registers value

Step – 3

Register	Offset (bit)	Bit length	Module	Channel	Datatype	Name
Outputs						
0	0	16	2	0	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 0
1	0	16	2	1	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 1
2	0	16	2	2	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 2
3	0	16	2	3	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 3
Inputs						
5000	0	16	2	0	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 0
5001	0	16	2	1	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 1
5002	0	16	2	2	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 2
5003	0	16	2	3	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 3
5004	0	8	2	4	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 4 - PQI
5004	8	8	2	5	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 5 - PQI
5005	0	8	2	6	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 6 - PQI
5005	8	8	2	7	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 7 - PQI

SMS Connected in Port Number - 0

Enter the Length in Word Size

1. Enter the Modbus input register address & length.
2. Enter the Modbus output register address & length.
3. Click **“OK”** to confirm the Modbus channel configuration.

Step – 4

1. Follow Step – 1 and Click **“Access type”** drop-down menu.
2. Select **“Function Code 3”** for read registers value.

Step – 5

Register	Offset (bit)	Bit length	Module	Channel	Datatype	Name
Outputs						
0	0	16	2	0	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 0
1	0	16	2	1	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 1
2	0	16	2	2	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 2
3	0	16	2	3	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 3
Inputs						
5000	0	16	2	0	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 0
5001	0	16	2	1	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 1
5002	0	16	2	2	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 2
5003	0	16	2	3	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 3
5004	0	8	2	4	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 4 - PQI
5004	8	8	2	5	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 5 - PQI
5005	0	8	2	6	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 6 - PQI
5005	8	8	2	7	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 7 - PQI

SMS Connected in Port Number - 0 & PQI in Port Number 4

Channel

Name

Channel 1

Access type

Read Holding Registers (Function Code 3)

Trigger

Cyclic

Cycle time (ms)

100

Comment

READ Register

Offset

5004

Length

2

Error handling

Keep last value

WRITE Register

Offset

0x0000

Length

0

OK

Cancel

Enter the Length in Word Size

1.

Enter the Modbus input register address & length.
2.

Click “OK” to confirm the Modbus channel configuration.



Note

•

Refer [Appendix – 9.1](#) to Know about Modbus address details.

Once Modbus channel configuration completed successfully, Channel look like below image.

Modbus_TCP_Slave x										
General										
Modbus Slave Channel	0	Channel 0	Read/Write Multiple Registers (Function Code 23)	Cyclic, t#100ms	16#1388	4	Keep last value	16#0000	4	
Modbus Slave Init	1	Channel 1	Read Holding Registers (Function Code 03)	Cyclic, t#100ms	16#138C	2	Keep last value			

2.3 MasterTCPSlave I/O Mapping

Either use Method 1 or Method 2 for I/O Mapping

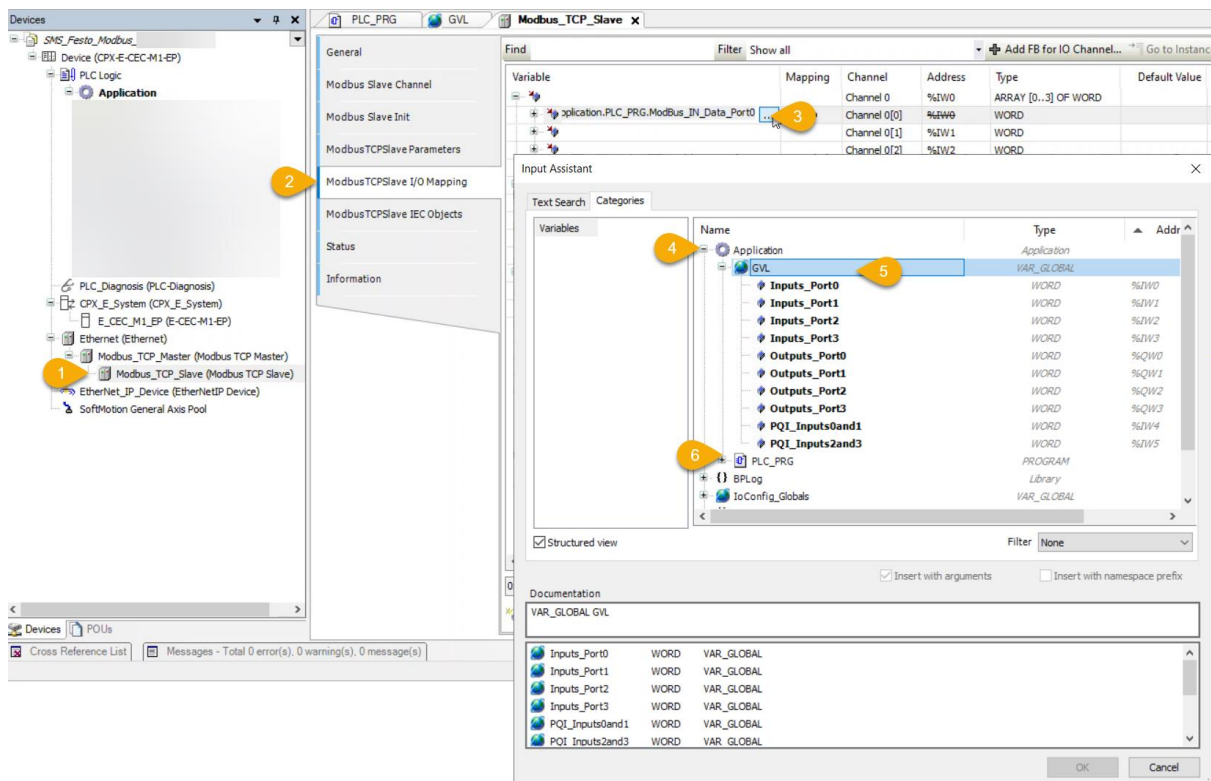
2.3.1 Method – 1: Link the I/O address using Global variable

```

1 {attribute 'qualified_only'}
2 VAR_GLOBAL
3   Inputs_Port0 AT %IW0 : WORD;
4   Inputs_Port1 AT %IW1 : WORD;
5   Inputs_Port2 AT %IW2 : WORD;
6   Inputs_Port3 AT %IW3 : WORD;
7   PQI_Inputs0and1 AT %IW4 : WORD;
8   PQI_Inputs2and3 AT %IW5 : WORD;
9   Outputs_Port0 AT %QW0 : WORD;
10  Outputs_Port1 AT %QW1 : WORD;
11  Outputs_Port2 AT %QW2 : WORD;
12  Outputs_Port3 AT %QW3 : WORD;
13 END_VAR

```

Create the Inputs, PQI_Inputs and Outputs tag in the GVL and assign the device I/O address to the tags.



1. Double click the **“Modbus_TCP_Slave”** module.
2. Select the **“ModbusTCPSlave I/O Mapping”** Tab.
3. Click **“Browse”** button in Input’s channel mapping.
4. Under **“Categories”** window from Input Assistant window Explore **“Application”** to find the GVL and PLC_PRG.
5. Select the variable to map the Channel’s Input and Output address via GVL(Global Variable) and select the Inputs_Port0 tag to map the Modbus input data in ModbusTCPSlave I/O Mapping.
6. Select the variable to map the Channel’s Input and Output address via PLC_PRG

2.3.2 Method – 2: Link the I/O address using Program variable

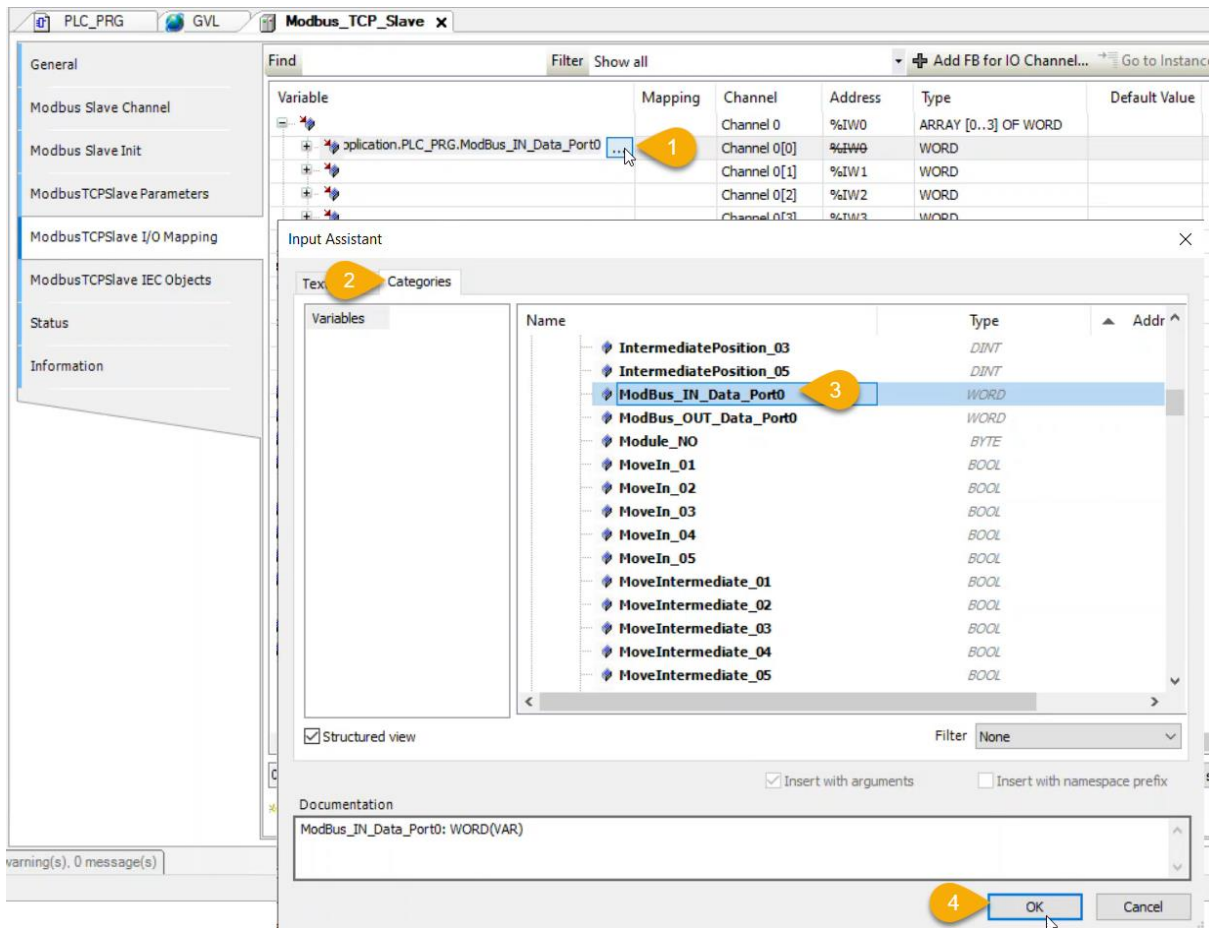
```

1  PROGRAM PLC_PRG
2  VAR
3      CPXAP_IOLink_Parameter_0: CPXAPModbusLib.CPXAP_IOLink_Parameter;
4      Module_NO: BYTE:= 2;    // IO-Link Master Module Number | FOR Read Module Value Starts FROM One
5      Ch_NO: BYTE:= 0;        // SMS Connected Channel Number in IO-Link Master | FOR First Port Value Starts FROM Zero
6      SMS_Inputs: Festo_SMS_ModbusTCP.UDI_WORD_TO_SINT;
7      SMS_Command: BYTE;
8      ModBus_IN_Data_Port0: WORD;
9      ModBus_OUT_Data_Port0: WORD;
10     ModBus_IN_Data_Port1: WORD;
11     ModBus_OUT_Data_Port1: WORD;
12     ModBus_IN_Data_Port2: WORD;
13     ModBus_OUT_Data_Port2: WORD;
14     ModBus_IN_Data_Port3: WORD;
15     ModBus_OUT_Data_Port3: WORD;
16
17 END_VAR

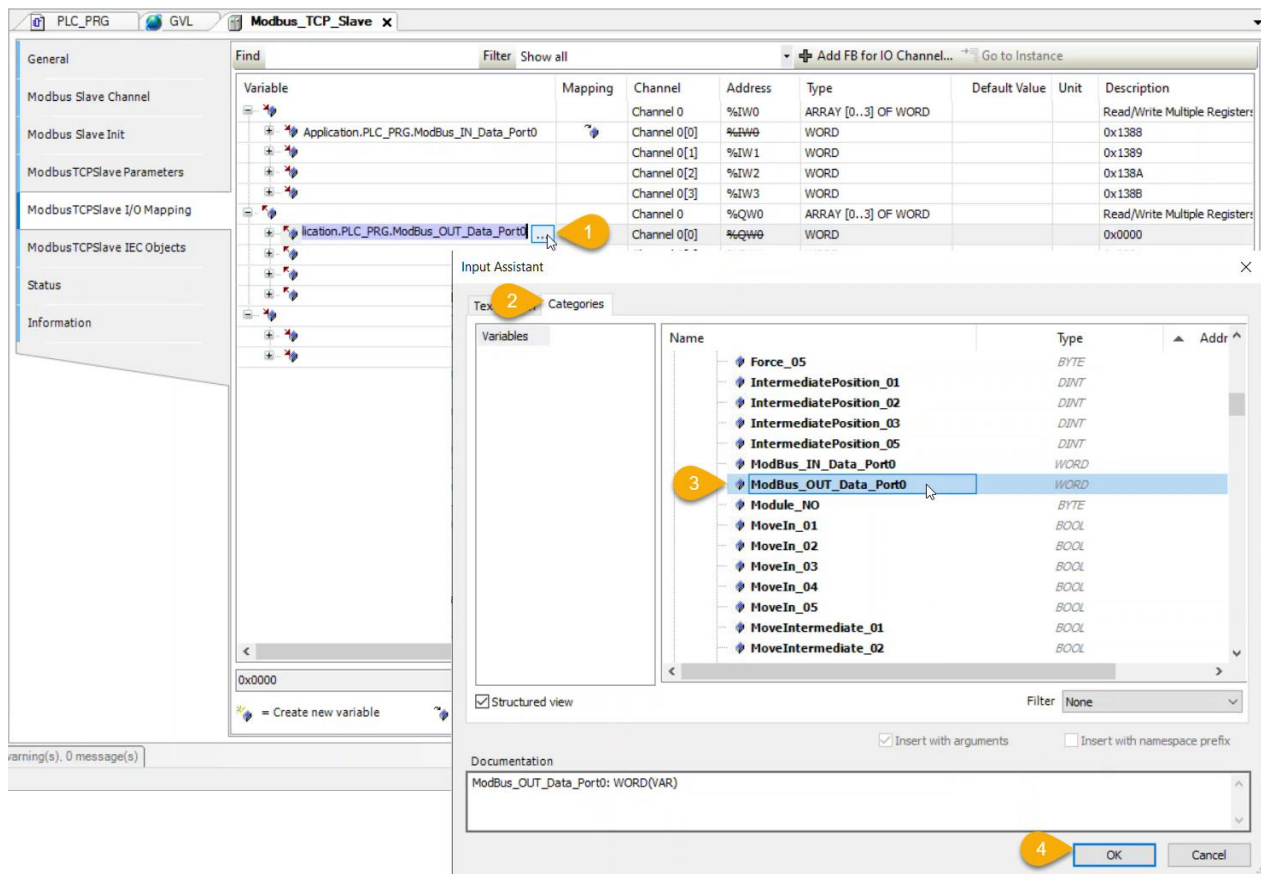
```

Create the variable tag in the Program and assign the device I/O address with variable tags.

Step – 1



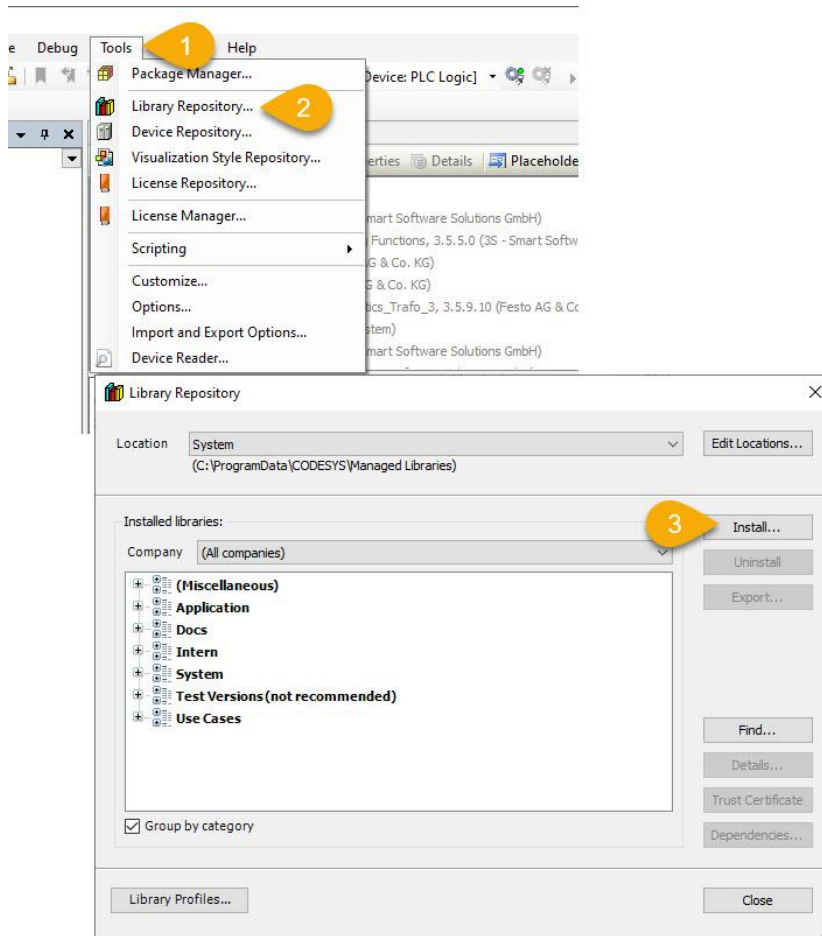
1. Click **"Browse"** button in Input's channel mapping.
2. Select **"Categories"** window from Input Assistant window.
Explore **"Application"** to find the GVL and PLC_PRG.
3. Select the variable to map the Channel's Input and Output address via PLC_PRG
Select the Modbus_IN_Data_Port0 tag to map the Modbus input data in ModbusTCP Slave I/O Mapping.
4. Click **"OK"** button to confirm the data mapping variables

Step – 2

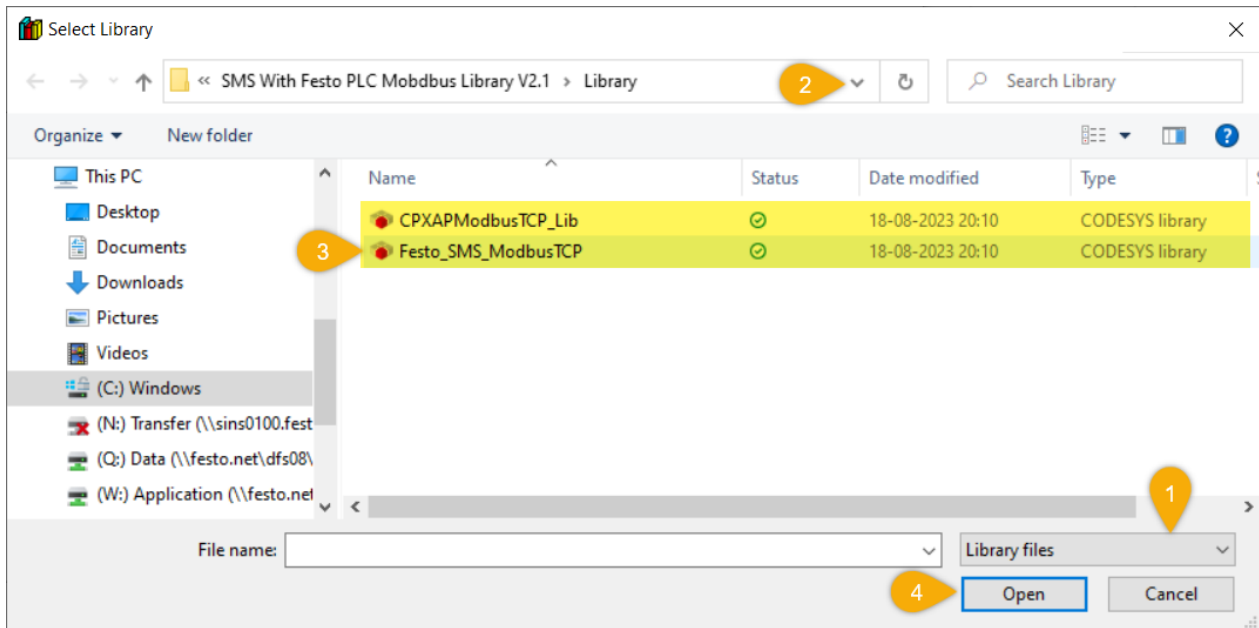
1. Click **"Browse"** button in Input's channel mapping.
2. Select **"Categories"** window from Input Assistant window.
Explore **"Application"** to find the GVL and PLC_PRG.
3. Select the variable to map the Channel's Input and Output address via PLC_PRG
Select the Modbus_OUT_Data_Port0 tag to map the Modbus input data in ModbusTCPSlave I/O Mapping.
4. Click **"OK"** button to confirm the data mapping variables

2.4 Import “Festo_SMS_ModbusTCP_Library” into CODESYS Runtime Project

Step – 1

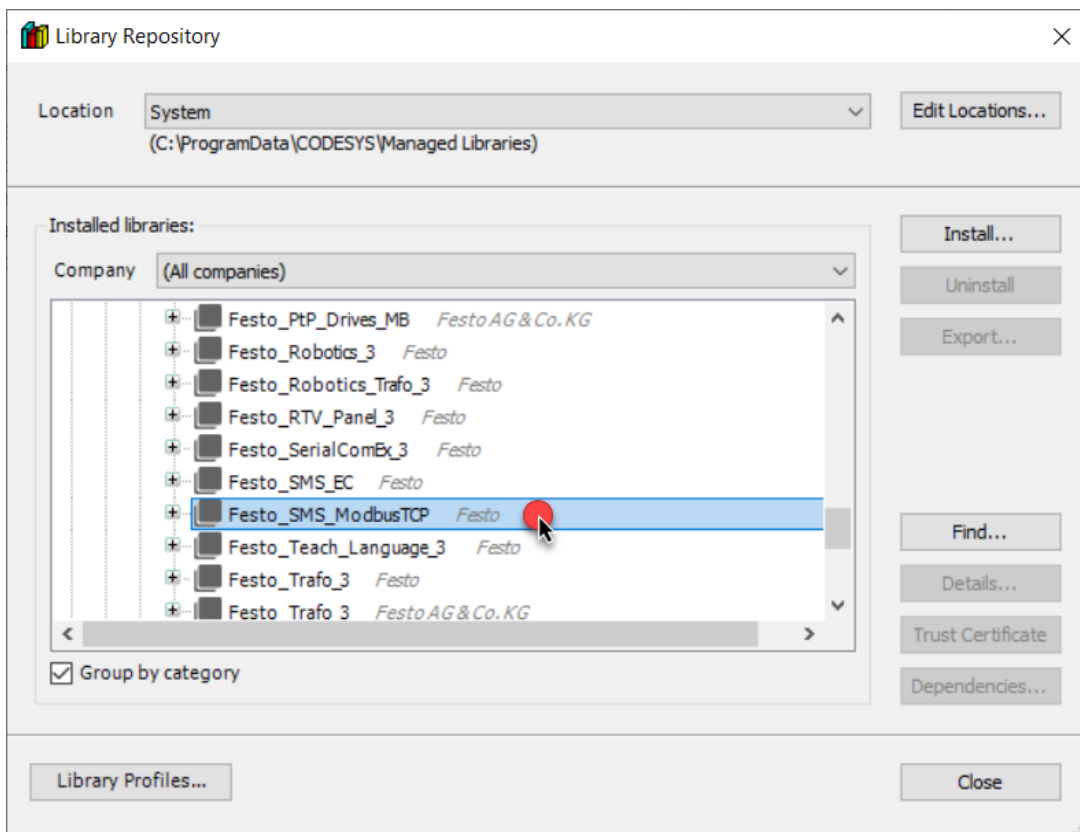


1. Open the CODESYS runtime project and select “**Tools**” option from the tool bar.
2. Select “**Library Repository**” from the Tools menu.
3. Click “**Install**” button from Library Repository window.

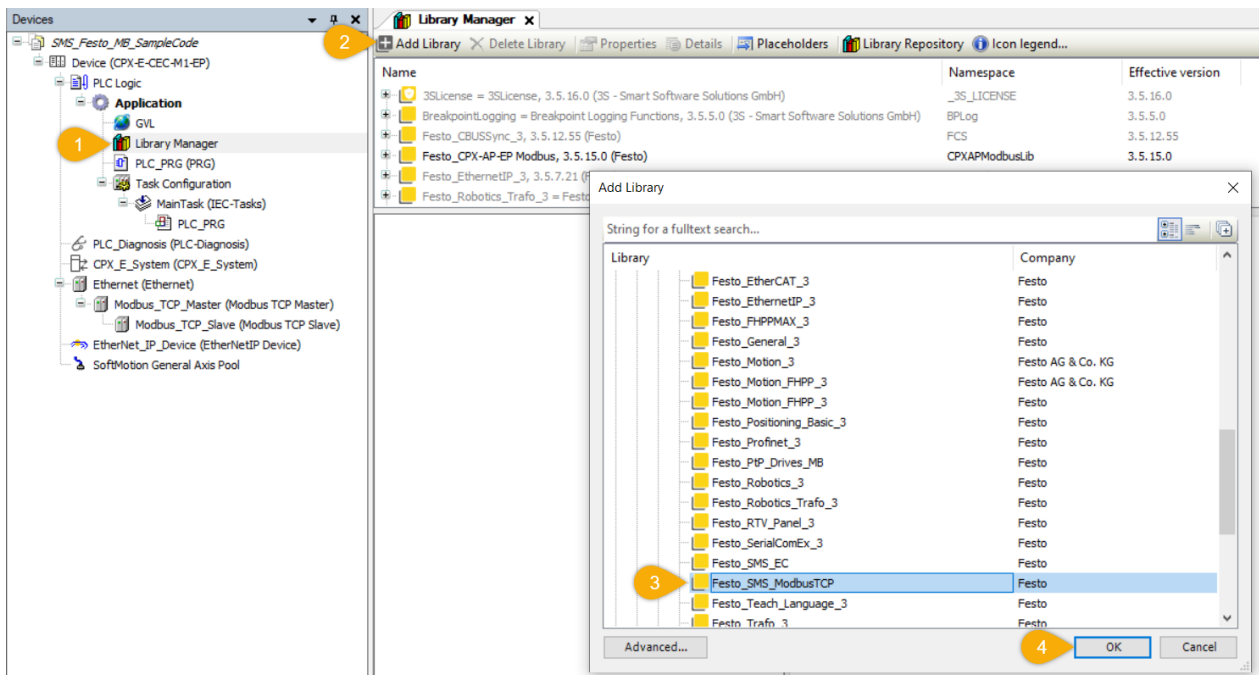
Step – 2

1. Select **“Library files”** from file type drop-down menu.
2. Navigate to the SMS library saved folder.
3. Select **“Festo_SMS_ModbusTCP.library”** file.
4. Click **“Open”** button.

Once the “Festo_SMS_ModbusTCP” library installed successfully then the library elements will be available in Installed libraries list. (**Application > Festo AG. Co. KG > Common > Festo_SMS_ModbusTCP**)

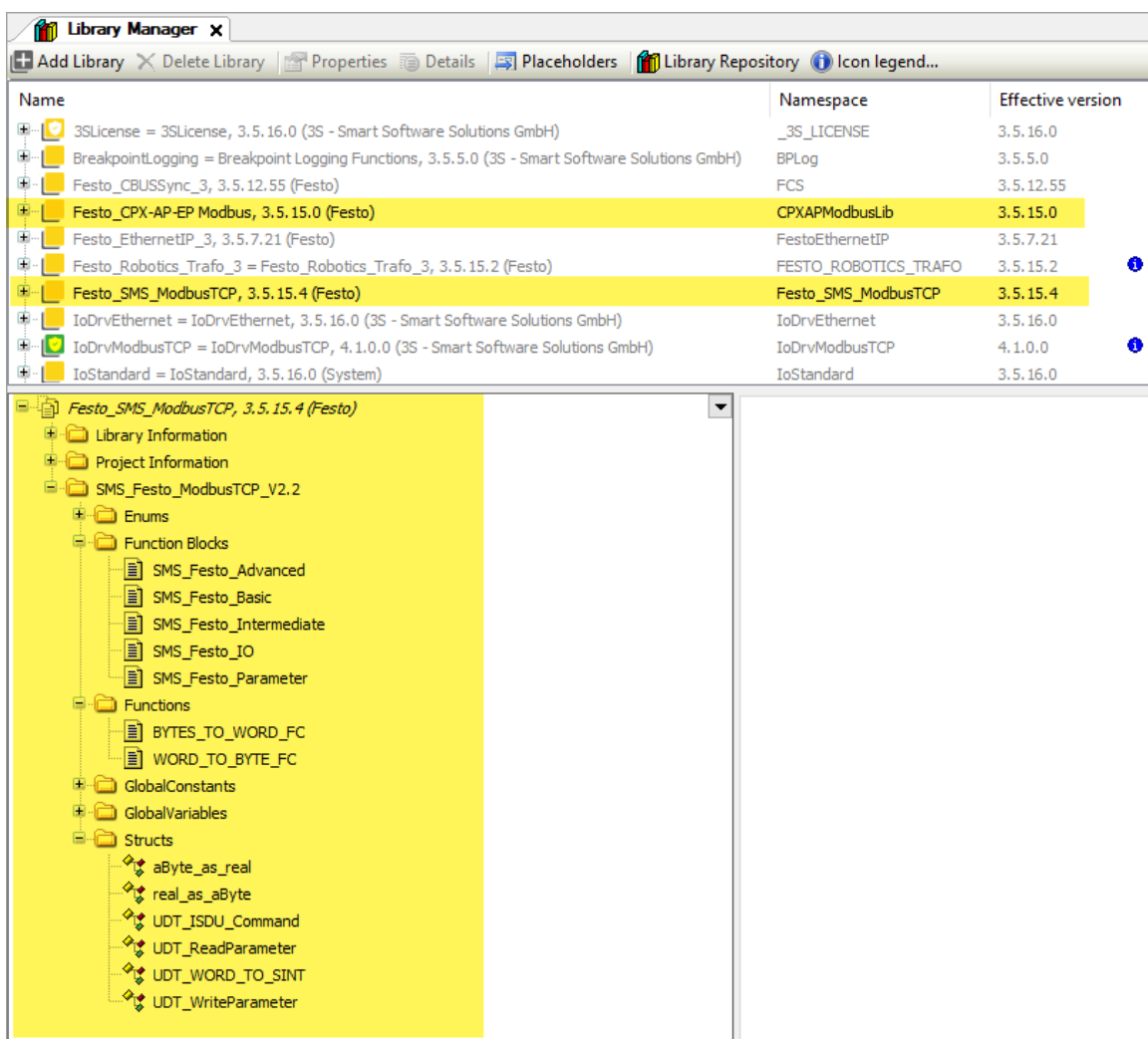


2.5 Add “Festo_SMS_ModbusTCP” Library in CODESYS Library Manager



1. Double click the “**Library Manager**” from device menu.
2. Click “**Add Library**” button in Library Manager window.
3. Browse for Festo_SMS_ModbusTCP library in installed library list. (**Application > Festo > Common > Festo_SMS_ModbusTCP**)
4. Click “**OK**” button to add the file.

Once the library added successfully the library elements will be available like below yellow highlight.



The Library sub folder Function Blocks and Structs has following Blocks and Supporting files.

Function Blocks

- SMS_Festo_Advance block is used to access all advanced parameters.
- SMS_Festo_Basic block is used to access the basic parameters
- SMS_Festo_Intermediate block is used to operate the basic function like IN and OUT movements and also access the Intermediate function Parameters.
- SMS_Festo_IO block is used to operate the basic function like IN and OUT movements.
- SMS_Festo_Parameter block is used to access the acyclic parameter function.

Supporting files

- aByte_as_real
- real_as_aByte
- UDT_ISDU_Common
- UDT_ReadParameter
- UDT_WORD_TO_SINT
- UDT_WriteParameter

The above supporting files are User data type files which are used in Advanced block, Basic block, Intermediate block and Parameter block.



Note

- Incase if you are getting “Festo_CPX-AP-EP_Modbus” **library missing error** then import “CPXAP_ModbusTCP_Lib” library attached in project directory, Repeat the [Chapter 3.4](#) and [Chapter 3.5](#)

3 Festo SMS Modbus Library Function Blocks

3.1 Introduction to “SMS_Festo_Advanced” Function Block

SMS_Festo_Advanced function block is used to control the all system function like IN & OUT function, Error reset, Enable power stage of drive, Execute reference, Enable and Execute file handling, Execute factory reset, Parameters read & write, Also to read the diagnostic data from the drive unit.



3.1.1 SMS_Festo_Advanced block IO interface Tag Details

Tag Name	Data Type	Function Description
RefINPUTS	BYTE	Feedback data from actuator unit. (Data Read from actuator unit to Controller)
RefOUTPUTS	BYTE	Command data to actuator unit. (Data Write from Controller to actuator unit)
byDeviceAvailable	BYTE	Port qualifier data from actuator. (Data Read from actuator unit to Controller)

3.1.2 SMS_Festo_Advanced block Control Tag Details

Tag Name	Data Type	Function Description
xEnable	BOOL	True - Enable function block
byModuleNumber	BYTE	Simplified Motion Series Connected Module Number From Head Module Value Start From 1
byChannelNumber	USINT	Simplified Motion Series Connected Channel Number From First Port Value Start From 0
xReadAll	BOOL	Read All Setpoint Parameter
xWriteAll	BOOL	Write All Setpoint Parameter
xMoveIN	BOOL	Control command for executing the motion "Mov _{In} " to the inner (reference) end position "Lim _{In} (Ref)" TRUE - Actuator unit moves to inner end position "Lim _{In} (Ref)" FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveOUT and MoveIntermediate.
xMoveIntermediate	BOOL	Control command for executing the movement "Mov _{Imp} " to the intermediate position "Pos _{Imp} " TRUE - Actuator unit moves to Intermediate Position "Pos _{Imp} " FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveIN and MoveOUT.
xMoveOut	BOOL	Control command for executing the motion "Mov _{Out} " to the outer end position "Lim _{Out} " TRUE - Actuator unit moves to outer end position "Lim _{Out} " FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveIN and MoveIntermediate.
xQuitError	BOOL	Control command for acknowledging errors TRUE - Start error acknowledgement FALSE - Do not start error acknowledgement
xPowerSMS	BOOL	Control command for switching on the power stage. TRUE - Enable power stage of actuator unit FALSE - Disable power stage of actuator unit This variable has no general feedback if the power stage is enabled or disabled. The variable does not remain constantly TRUE or FALSE, but rather works on rising edge principle. Due to "simplified" functionality of actuator unit, the device tries everything to re-power itself. E.g. if setting xMoveIn, xMoveOut, or xMoveIntermediate to TRUE, then the axis will automatically be enabled.
xExecuteReference	BOOL	Command for executing the homing with end position detection (detection of the mechanical end positions) TRUE - Start homing FALSE - Do not start homing Note: Take effect only when falling to raising edge when AOI is enable state.

Tag Name	Data Type	Function Description
xStoreParameters	BOOL	Control command for single and conscious permanent saving of parameters in the flash memory TRUE - Manually stores the last downloaded parameters in the flash memory FALSE - Parameters are temporarily saved in RAM. Note: Take effect only when falling to raising edge when AOI is enable state.
xReadDiagnosticData	BOOL	TRUE - Update value of outputs: DDTemperature DDCurrent DDVoltage DDCyclesTotal DDCyclesSinceReset DDMileageTotal DDMileageSinceReset
xResetCounterMileage	BOOL	TRUE - Resets values of parameters DDCyclesSinceReset
xEnableFileHandling	BOOL	Control command for activating file handling, e.g. firmware update TRUE - File handling activated FALSE - File handling deactivated (default)
xExecuteFileHandling	BOOL	Control command for executing file handling TRUE - Execute file handling, e.g. firmware update FALSE - File handling deactivated (default)
xFactoryReset	BOOL	Reset the device to the factory Settings TRUE – Enable FactoryReset FALSE – Do not execute Factory Reset
xUserInterfaceLock	BOOL	TRUE - Access to display and operating components (HMI) on control unit locked FALSE - Access to display and operating components (HMI) on control unit enabled
bySpeedOut	BYTE	Speed for movement “Mov _{Out} ” towards inner outer end position “Lim _{Out} ” 1 - 10% (default) ... 10 - 100%
bySpeedIN	BYTE	Speed for movement “Mov _{In} ” towards inner (reference) end position “Lim _{In} (Ref)” 1 - 10% (default) ... 10 - 100%
byForce	BYTE	Force for force controlled movement “Mov _{Out} ” from position “Pos _{StartPress} ” towards outer end position “Lim _{Out} ” 1 - 10% (default) ... 10 - 100%

Tag Name	Data Type	Function Description
xReferenceDirection	BOOL	Position of the reference end position "Ref" after homing has been carried out Linear drive systems: TRUE: Facing away from motor FALSE: Facing motor Rotary drive systems (ERMS: view of the rotating plate) TRUE: Right FALSE: Left
diStartPressPosition	DINT	Stroke or angle of rotation distance from the "Start Press" position to the reference end position "Ref". Defines point on stroke, actuator starts force controlled movement. Linear actuator unit: -unit: [mm]-gradient: 0.01 Rotative actuator unit: -unit: [°]-gradient: 0.1
diIntermediatePosition	DINT	Stroke or rotation angle distance of the intermediate position "Pos _{imp} " to the reference end position "Ref". Defines intermediate point on stroke, actuator movement stops and waits for next motion command. Linear actuator unit: -unit: [mm]-gradient: 0.01 Rotative actuator unit: -unit: [°]-gradient: 0.1
diEndPosition	DINT	Stroke or rotation angle distance of the "Lim _{out} " end position to the reference end position "Ref" Defines end point on stroke, actuator movement stops and waits for next motion command. Linear actuator unit: -unit: [mm] -gradient: 0.01 Rotative actuator unit: -unit: [°] -gradient: 0.1
xAutoStoreActive	BOOL	Control command for activating automatic and permanent saving of parameters in the flash memory TRUE - Automatic saving activated (default) FALSE - Automatic saving deactivated Note: When changing parameters frequently, make sure to deactivate auto store to avoid damages to the flash memory by exceeding 100.000 write cycles.

3.1.3 SMS_Festo_Advanced block Monitor Tag Details

Tag Name	Data Type	Function Description
xEnabled	BOOL	TRUE – function block enabled.
xWriteActive	BOOL	TRUE - Executing parameter write function.
xWriteDone	BOOL	TRUE - Last parameter wrote successfully.
xWriteAllDone	BOOL	TRUE - All parameter value over written successfully
xReadActive	BOOL	TRUE - Parameter read function is active.
xReadAllDone	BOOL	TRUE – All parameter value feedback updated successfully
xError	BOOL	TRUE - Actuator unit in error state.
xStateln	BOOL	Status at inner (reference) end position "Lim _{in} (Ref)" TRUE - Inner (reference) end position "Lim _{in} (Ref)" reached FALSE – Inner (reference) end position "Lim _{in} (Ref)" not reached

Tag Name	Data Type	Function Description
xStateIntermediate	BOOL	Status at intermediate position “Pos _{Imp} ” TRUE - Intermediate position “Pos _{Imp} ” reached FALSE – Intermediate position “Pos _{Imp} ” not reached
xStateOut	BOOL	Status at outer end position “Lim _{Out} ” TRUE - Outer end position "Lim _{Out} " reached FALSE - Outer end position "Lim _{Out} " not reached
xStateMove	BOOL	Status of the actuator unit TRUE - Actuator unit in moving condition FALSE - Actuator unit in standstill condition
xStateDevice	BOOL	TRUE – Device Ready for Operation. FALSE – Device not Ready.
diCurrentPosition	DINT	Actuator unit current position feedback Linear actuator unit: - unit: [mm] - gradient: 0.01 Rotative actuator unit: - unit: [°] - gradient: 0.1
diCurrentSpeed	DINT	Actuator unit current speed feedback Linear actuator unit: - unit [mm/s] -gradient: 0.01 Rotative actuator unit: - unit [rpm] -gradient: 0.1
diCurrentForce	DINT	Actuator unit current feed force feedback Linear actuator unit: - unit [N] -gradient: 0.01 Rotative actuator unit: - unit [Nm] -gradient: 0.1
diDDTemperature	DINT	Actuator unit current temperature feedback Unit: [°C].
diDDCurrent	DINT	Actuator unit present value of current Unit: [A].
diDDVoltage	DINT	Actuator unit present voltage feedback Unit: [V].
diDDCyclesTotal	DINT	Number (dec) of completed movement cycles of the actuator unit since delivery
diDDCyclesSinceReset	DINT	Number (dec) of completed movement cycles of the actuator unit since last reset command
diDDMileageTotal	DINT	Mileage of the actuator unit since delivery Linear actuator unit: unit [km], 0.000001. Rotative actuator unit: unit [r], 0.001.
diDDMileageSinceReset	DINT	Mileage of the actuator unit since last reset command Linear actuator unit: unit [km], 0.000001. Rotative actuator unit: unit [r], 0.001.
diNumOfStorageOperation	DINT	Total number of permanent storage processes in the flash memory since delivery
aErrorCode	Array of WORD[16]	Error code of actuator unit Refer Chapter-3.5.4 for detailed description of Error code

Tag Name	Data Type	Function Description
uiLocalError	UINT	Function block Local Error 1001 – bySpeedIN Parameter limit out of range error. 1002 – bySpeedOut Parameter limit out of range error. 1003 – byForce Parameter limit out of range error. 1004 – Read_Parameter Error. 1005 – Write_Parameter Error. 1006 – “Position Value” Error. Check If Start Press Position value is Higher than the End Position Value. 1007 – “Position Value” Error. Check If Intermediate Position value is Higher than the End Position Value. 1008 – “PQI” Error. Check the device connection.

3.2 Introduction to “SMS_Festo_Basic” Function Block

SMS_Festo_Basic function block is used to control some basic function like IN & OUT function, Error reset, Enable power stage of drive, Execute reference, Parameters read & write.



3.2.1 SMS_Festo_Basic block IO interface Tag Details

Tag Name	Data Type	Function Description
RefINPUTS	BYTE	Feedback data from actuator unit. (Data Read from actuator unit to Controller)
RefOUTPUTS	BYTE	Command data to actuator unit. (Data Write from Controller to actuator unit)
byDeviceAvailable	WORD	Port qualifier data from actuator. (Data Read from actuator unit to Controller)

3.2.2 SMS_Festo_Basic block Control Tag Details

Tag Name	Data Type	Function Description
xEnable	BOOL	True - Enable function block
byModuleNumber	BYTE	Simplified Motion Series Connected Module Number From Head Module Value Start From 1
byChannelNumber	USINT	Simplified Motion Series Connected Channel Number From First Port Value Start From 0
xReadAll	BOOL	Read All Setpoint Parameter
xWriteAll	BOOL	Write All Setpoint Parameter
xMoveIN	BOOL	Control command for executing the motion "Mov _{In} " to the inner (reference) end position "Lim _{In} (Ref)" TRUE - Actuator unit moves to inner end position "Lim _{In} (Ref)" FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveOUT and MoveIntermediate.
xMoveIntermediate	BOOL	Control command for executing the movement "Mov _{Imp} " to the in- termediate position "Pos _{Imp} " TRUE - Actuator unit moves to Intermediate Position "Pos _{Imp} " FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveIN and MoveOUT.
xMoveOut	BOOL	Control command for executing the motion "Mov _{Out} " to the outer end position "Lim _{Out} " TRUE - Actuator unit moves to outer end position "Lim _{Out} " FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveIN and MoveIntermediate.
xQuitError	BOOL	Control command for acknowledging errors TRUE - Start error acknowledgement FALSE - Do not start error acknowledgement
xPowerSMS	BOOL	Control command for switching on the power stage. TRUE - Enable power stage of actuator unit FALSE - Disable power stage of actuator unit This variable has no general feedback if the power stage is ena- bled or disabled. The variable does not remain constantly TRUE or FALSE , but ra- ther works on rising edge principle. Due to "simplified" functionality of actuator unit, the device tries everything to re-power itself. E.g. if setting xMoveIn, xMoveOut, or xMoveIntermediate to TRUE, then the axis will automatically be enabled.

Tag Name	Data Type	Function Description
xExecuteReference	BOOL	Command for executing the homing with end position detection (detection of the mechanical end positions) TRUE - Start homing FALSE - Do not start homing Note: Take effect only when falling to raising edge when AOI is enable state.
xStoreParameters	BOOL	Control command for single and conscious permanent saving of parameters in the flash memory TRUE - Manually stores the last downloaded parameters in the flash memory FALSE - Parameters are temporarily saved in RAM. Note: Take effect only when falling to raising edge when AOI is enable state.
xUserInterfaceLock	BOOL	TRUE - Access to display and operating components (HMI) on control unit locked FALSE - Access to display and operating components (HMI) on control unit enabled
bySpeedOut	BYTE	Speed for movement "Mov _{Out} " towards inner outer end position "Lim _{Out} " 1 - 10% (default) ... 10 - 100%
bySpeedIN	BYTE	Speed for movement "Mov _{In} " towards inner (reference) end position "Lim _{In} (Ref)" 1 - 10% (default) ... 10 - 100%
byForce	BYTE	Force for force controlled movement "Mov _{Out} " from position "Pos _{StartPress} " towards outer end position "Lim _{Out} " 1 - 10% (default) ... 10 - 100%
xReferenceDirection	BOOL	Position of the reference end position "Ref" after homing has been carried out Linear drive systems: TRUE: Facing away from motor FALSE: Facing motor Rotary drive systems (ERMS: view of the rotating plate) TRUE: Right FALSE: Left
diStartPressPosition	DINT	Stroke or angle of rotation distance from the "Start Press" position to the reference end position "Ref". Defines point on stroke, actuator starts force controlled movement. Linear actuator unit: -unit: [mm]-gradient: 0.01 Rotative actuator unit: -unit: [°]-gradient: 0.1

Tag Name	Data Type	Function Description
diIntermediatePosition	DINT	Stroke or rotation angle distance of the intermediate position "Pos _{Imp} " to the reference end position "Ref". Defines intermediate point on stroke, actuator movement stops and waits for next motion command. Linear actuator unit: -unit: [mm]-gradient: 0.01 Rotative actuator unit: -unit: [°]-gradient: 0.1
diEndPosition	DINT	Stroke or rotation angle distance of the "Lim _{Out} " end position to the reference end position "Ref" Defines end point on stroke, actuator movement stops and waits for next motion command. Linear actuator unit: -unit: [mm] -gradient: 0.01 Rotative actuator unit: -unit: [°] -gradient: 0.1
xAutoStoreActive	BOOL	Control command for activating automatic and permanent saving of parameters in the flash memory TRUE - Automatic saving activated (default) FALSE - Automatic saving deactivated Note: When changing parameters frequently, make sure to deactivate auto store to avoid damages to the flash memory by exceeding 100.000 write cycles.

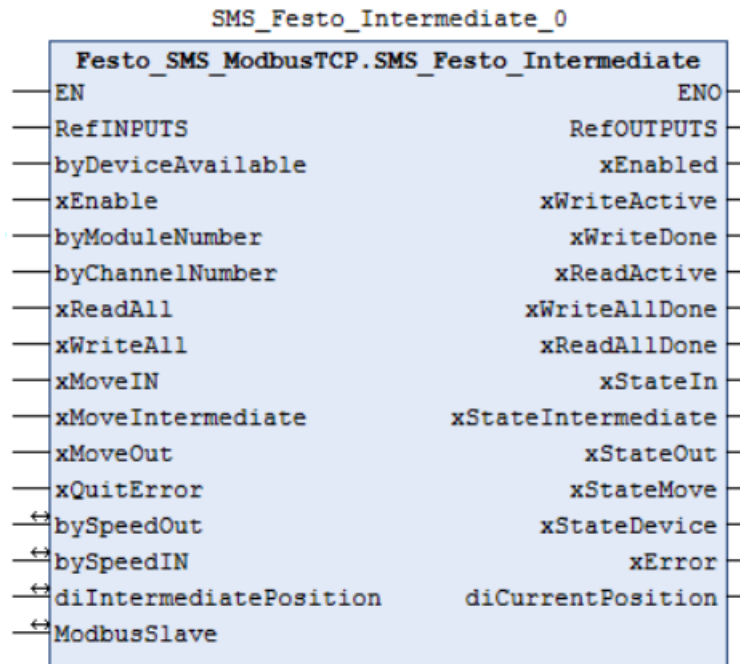
3.2.3 SMS_Festo_Basic block Monitor Tag Details

Tag Name	Data Type	Function Description
xEnabled	BOOL	TRUE – function block enabled.
xWriteActive	BOOL	TRUE - Executing parameter write function.
xWriteDone	BOOL	TRUE - Last parameter wrote successfully.
xWriteAllDone	BOOL	TRUE - All parameter value over written successfully
xReadActive	BOOL	TRUE - Parameter read function is active.
xReadAllDone	BOOL	TRUE – All parameter value feedback updated successfully
xError	BOOL	TRUE - Actuator unit in error state.
xStateIn	BOOL	Status at inner (reference) end position "Lim _{In} (Ref)" TRUE - Inner (reference) end position "Lim _{In} (Ref)" reached FALSE – Inner (reference) end position "Lim _{In} (Ref)" not reached
xStateIntermediate	BOOL	Status at intermediate position "Pos _{Imp} " TRUE - Intermediate position "Pos _{Imp} " reached FALSE – Intermediate position "Pos _{Imp} " not reached
xStateOut	BOOL	Status at outer end position "Lim _{Out} " TRUE - Outer end position "Lim _{Out} " reached FALSE - Outer end position "Lim _{Out} " not reached
xStateMove	BOOL	Status of the actuator unit TRUE - Actuator unit in moving condition FALSE - Actuator unit in standstill condition
xStateDevice	BOOL	TRUE – Device Ready for Operation. FALSE – Device not Ready.

Tag Name	Data Type	Function Description
diCurrentPosition	DINT	Actuator unit current position feedback Linear actuator unit: - unit: [mm] - gradient: 0.01 Rotative actuator unit: - unit: [°] - gradient: 0.1
diCurrentSpeed	DINT	Actuator unit current speed feedback Linear actuator unit: - unit [mm/s] -gradient: 0.01 Rotative actuator unit: - unit [rpm] -gradient: 0.1
diCurrentForce	DINT	Actuator unit current feed force feedback Linear actuator unit: - unit [N] -gradient: 0.01 Rotative actuator unit: - unit [Nm] -gradient: 0.1
diNumOfStorageOperation	DINT	Total number of permanent storage processes in the flash memory since delivery
aErrorCode	Array of WORD[16]	Error code of actuator unit Refer Chapter-3.5.4 for detailed description of Error code
uiLocalError	UINT	Function block Local Error 1001 – bySpeedIN Parameter limit out of range error. 1002 – bySpeedOut Parameter limit out of range error. 1003 – byForce Parameter limit out of range error. 1004 – Read_Parameter Error. 1005 - Write_Parameter Error. 1006 - “Position Value” Error. Check If Start Press Position value is Higher than the End Position Value. 1007 - “Position Value” Error. Check If Intermediate Position value is Higher than the End Position Value. 1008 – “PQI” Error. Check the device connection.

3.3 Introduction to “SMS_Festo_Intermediate” Function Block

SMS_Festo_Intermediate function block is used to control the intermediate position and some basic function like IN & OUT function, Error reset and intermediate position related Parameters read & write.



3.3.1 SMS_Festo_Intermediate block IO interface Tag Details

Tag Name	Data Type	Function Description
RefINPUTS	BYTE	Feedback data from actuator unit. (Data Read from actuator unit to Controller)
RefOUTPUTS	BYTE	Command data to actuator unit. (Data Write from Controller to actuator unit)
byDeviceAvailable	WORD	Port qualifier data from actuator. (Data Read from actuator unit to Controller)

3.3.2 SMS_Festo_Intermediate block Control Tag Details

Tag Name	Data Type	Function Description
xEnable	BOOL	True - Enable function block
byModuleNumber	BYTE	Simplified Motion Series Connected Module Number From Head Module Value Start From 1
byChannelNumber	USINT	Simplified Motion Series Connected Channel Number From First Port Value Start From 0
xReadAll	BOOL	Read All Setpoint Parameter
xWriteAll	BOOL	Write All Setpoint Parameter
xMoveIN	BOOL	Control command for executing the motion "Mov _{in} " to the inner (reference) end position "Lim _{in} (Ref)" TRUE - Actuator unit moves to inner end position "Lim _{in} (Ref)" FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveOUT and MoveIntermediate.
xMoveIntermediate	BOOL	Control command for executing the movement "Mov _{imp} " to the in- termediate position "Pos _{imp} " TRUE - Actuator unit moves to Intermediate Position "Pos _{imp} " FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveIN and MoveOUT.

Tag Name	Data Type	Function Description
xMoveOut	BOOL	Control command for executing the motion "Mov _{Out} " to the outer end position "Lim _{Out} " TRUE - Actuator unit moves to outer end position "Lim _{Out} " FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveIn and MoveIntermediate.
xQuitError	BOOL	Control command for acknowledging errors TRUE - Start error acknowledgement FALSE - Do not start error acknowledgement
bySpeedOut	BYTE	Speed for movement "Mov _{Out} " towards inner outer end position "Lim _{Out} " 1 - 10% (default) ... 10 - 100%
bySpeedIN	BYTE	Speed for movement "Mov _{In} " towards inner (reference) end position "Lim _{In} (Ref)" 1 - 10% (default) ... 10 - 100%
diIntermediatePosition	DINT	Stroke or rotation angle distance of the intermediate position "Pos _{Imp} " to the reference end position "Ref". Defines intermediate point on stroke, actuator movement stops and waits for next motion command. Linear actuator unit: -unit: [mm]-gradient: 0.01 Rotative actuator unit: -unit: [°]-gradient: 0.1

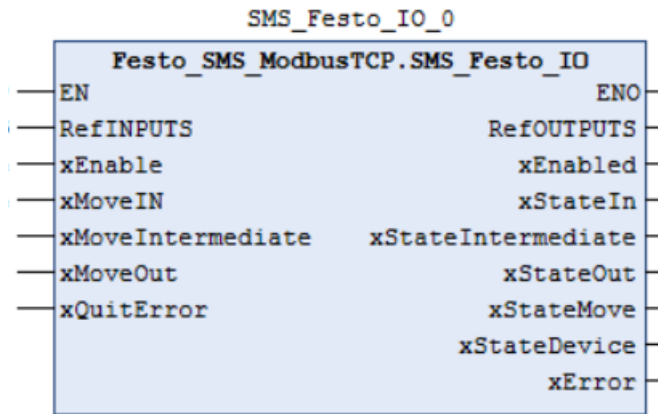
3.3.3 SMS_Festo_ Intermediate block Monitor Tag Details

Tag Name	Data Type	Function Description
xEnabled	BOOL	TRUE – function block enabled.
xWriteActive	BOOL	TRUE - Executing parameter write function.
xWriteDone	BOOL	TRUE - Last parameter wrote successfully.
xWriteAllDone	BOOL	TRUE - All parameter value over written successfully
xReadActive	BOOL	TRUE - Parameter read function is active.
xReadAllDone	BOOL	TRUE – All parameter value feedback updated successfully
xError	BOOL	TRUE - Actuator unit in error state.
xStateIn	BOOL	Status at inner (reference) end position "Lim _{In} (Ref)" TRUE - Inner (reference) end position "Lim _{In} (Ref)" reached FALSE – Inner (reference) end position "Lim _{In} (Ref)" not reached
xStateIntermediate	BOOL	Status at intermediate position "Pos _{Imp} " TRUE - Intermediate position "Pos _{Imp} " reached FALSE – Intermediate position "Pos _{Imp} " not reached
xStateOut	BOOL	Status at outer end position "Lim _{Out} " TRUE - Outer end position "Lim _{Out} " reached FALSE - Outer end position "Lim _{Out} " not reached

Tag Name	Data Type	Function Description
xStateMove	BOOL	Status of the actuator unit TRUE - Actuator unit in moving condition FALSE - Actuator unit in standstill condition
xStateDevice	BOOL	TRUE – Device Ready for Operation. FALSE – Device not Ready.
diCurrentPosition	DINT	Actuator unit current position feedback Linear actuator unit: - unit: [mm] - gradient: 0.01 Rotative actuator unit: - unit: [°] - gradient: 0.1

3.4 Introduction to “SMS_Festo_IO” Function Block

SMS_Festo_IO function block is used to control only the motion function like IN & OUT function and Error reset.



3.4.1 SMS_Festo_IO block IO interface Tag Details

Tag Name	Data Type	Function Description
RefINPUTS	BYTE	Feedback data from actuator unit. (Data Read from actuator unit to Controller)
RefOUTPUTS	BYTE	Command data to actuator unit. (Data Write from Controller to actuator unit)

3.4.2 SMS_Festo_IO block Control Tag Details

Tag Name	Data Type	Function Description
xEnable	BOOL	True - Enable function block
xMoveIN	BOOL	Control command for executing the motion "Mov _{In} " to the inner (reference) end position "Lim _{In} (Ref)" TRUE - Actuator unit moves to inner end position "Lim _{In} (Ref)" FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveOUT and MoveIntermediate.
xMoveIntermediate	BOOL	Control command for executing the movement "Mov _{Imp} " to the intermediate position "Pos _{Imp} " TRUE - Actuator unit moves to Intermediate Position "Pos _{Imp} " FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveIN and MoveOUT.
xMoveOut	BOOL	Control command for executing the motion "Mov _{Out} " to the outer end position "Lim _{Out} " TRUE - Actuator unit moves to outer end position "Lim _{Out} " FALSE - Actuator unit stops Note: The control command only works in combination with the bits MoveIN and MoveIntermediate.
xQuitError	BOOL	Control command for acknowledging errors TRUE - Start error acknowledgement FALSE - Do not start error acknowledgement

3.4.3 SMS_Festo_IO block Monitor Tag Details

Tag Name	Data Type	Function Description
xEnabled	BOOL	TRUE – function block enabled.
xError	BOOL	TRUE - Actuator unit in error state.
xStateIn	BOOL	Status at inner (reference) end position "LimIn(Ref)" TRUE - Inner (reference) end position "LimIn(Ref)" reached FALSE – Inner (reference) end position "LimIn(Ref)" not reached
xStateIntermediate	BOOL	Status at intermediate position "PosImp" TRUE - Intermediate position "PosImp" reached FALSE – Intermediate position "PosImp" not reached
xStateOut	BOOL	Status at outer end position "LimOut" TRUE - Outer end position "LimOut" reached FALSE - Outer end position "LimOut" not reached
xStateMove	BOOL	Status of the actuator unit TRUE - Actuator unit in moving condition FALSE - Actuator unit in standstill condition
xStateDevice	BOOL	TRUE – Device Ready for Operation. FALSE – Device not Ready.

3.5 Introduction to “SMS_Festo_Parameter” Function Block

SMS_Festo_Parameter function block is used to control only the acyclic parameter like Enable power stage of drive, Execute reference, Enable and Execute file handling, Execute factory reset, Parameters read & write, Also to read the diagnostic data from the drive unit.



3.5.1 SMS_Festo_Parameter block IO interface Tag Details

Tag Name	Data Type	Function Description
RefINPUTS	BYTE	Feedback data from actuator unit. (Data Read from actuator unit to Controller)
byDeviceAvailable	WORD	Port qualifier data from actuator. (Data Read from actuator unit to Controller)

3.5.2 SMS_Festo_Parameter block Control Tag Details

Tag Name	Data Type	Function Description
xEnable	BOOL	True - Enable function block
byModuleNumber	BYTE	Simplified Motion Series Connected Module Number From Head Module Value Start From 1

Tag Name	Data Type	Function Description
byChannelNumber	USINT	Simplified Motion Series Connected Channel Number From First Port Value Start From 0
xReadAll	BOOL	Read All Setpoint Parameter
xWriteAll	BOOL	Write All Setpoint Parameter
xQuitError	BOOL	Control command for acknowledging errors TRUE - Start error acknowledgement FALSE - Do not start error acknowledgement
xPowerSMS	BOOL	Control command for switching on the power stage. TRUE - Enable power stage of actuator unit FALSE - Disable power stage of actuator unit This variable has no general feedback if the power stage is enabled or disabled. The variable does not remain constantly TRUE or FALSE , but rather works on rising edge principle. Due to “simplified” functionality of actuator unit, the device tries everything to re-power itself. E.g. if setting xMoveIn, xMoveOut, or xMoveIntermediate to TRUE, then the axis will automatically be enabled.
xExecuteReference	BOOL	Command for executing the homing with end position detection (detection of the mechanical end positions) TRUE - Start homing FALSE - Do not start homing Note: Take effect only when falling to raising edge when AOI is enable state.
xStoreParameters	BOOL	Control command for single and conscious permanent saving of parameters in the flash memory TRUE - Manually stores the last downloaded parameters in the flash memory FALSE - Parameters are temporarily saved in RAM. Note: Take effect only when falling to raising edge when AOI is enable state.
xReadDiagnosticData	BOOL	TRUE - Update value of outputs: DDTemperature DDCurrent DDVoltage DDCyclesTotal DDCyclesSinceReset DDMileageTotal DDMileageSinceReset
xResetCounterMileage	BOOL	TRUE - Resets values of parameters DDCyclesSinceReset
xEnableFileHandling	BOOL	Control command for activating file handling, e.g. firmware update TRUE - File handling activated FALSE - File handling deactivated (default)
xExecuteFileHandling	BOOL	Control command for executing file handling TRUE - Execute file handling, e.g. firmware update FALSE - File handling deactivated (default)
xFactoryReset	BOOL	Reset the device to the factory Settings TRUE – Enable FactoryReset FALSE – Do not execute Factory Reset

Tag Name	Data Type	Function Description
xUserInterfaceLock	BOOL	TRUE - Access to display and operating components (HMI) on control unit locked FALSE - Access to display and operating components (HMI) on control unit enabled
bySpeedOut	BYTE	Speed for movement "Mov _{Out} " towards inner outer end position "Lim _{Out} " 1 - 10% (default) ... 10 - 100%
bySpeedIN	BYTE	Speed for movement "Mov _{In} " towards inner (reference) end position "Lim _{In} (Ref)" 1 - 10% (default) ... 10 - 100%
byForce	BYTE	Force for force controlled movement "Mov _{Out} " from position "Pos _{StartPress} " towards outer end position "Lim _{Out} " 1 - 10% (default) ... 10 - 100%
xReferenceDirection	BOOL	Position of the reference end position "Ref" after homing has been carried out Linear drive systems: TRUE : Facing away from motor FALSE : Facing motor Rotary drive systems (ERMS: view of the rotating plate) TRUE : Right FALSE : Left
diStratPressPosition	DINT	Stroke or angle of rotation distance from the "Start Press" position to the reference end position "Ref". Defines point on stroke, actuator starts force controlled movement. Linear actuator unit :-unit: [mm]-gradient: 0.01 Rotative actuator unit :-unit: [°]-gradient: 0.1
diIntermediatePosition	DINT	Stroke or rotation angle distance of the intermediate position "Pos _{Imp} " to the reference end position "Ref". Defines intermediate point on stroke, actuator movement stops and waits for next motion command. Linear actuator unit :-unit: [mm]-gradient: 0.01 Rotative actuator unit :-unit: [°]-gradient: 0.1
diEndPosition	DINT	Stroke or rotation angle distance of the "Lim _{Out} " end position to the reference end position "Ref" Defines end point on stroke, actuator movement stops and waits for next motion command. Linear actuator unit :-unit: [mm] -gradient: 0.01 Rotative actuator unit :-unit: [°] -gradient: 0.1

Tag Name	Data Type	Function Description
xAutoStoreActive	BOOL	Control command for activating automatic and permanent saving of parameters in the flash memory TRUE - Automatic saving activated (default) FALSE - Automatic saving deactivated Note: When changing parameters frequently, make sure to deactivate auto store to avoid damages to the flash memory by exceeding 100.000 write cycles.

3.5.3 SMS_Festo_Parameter block Monitor Tag Details

Tag Name	Data Type	Function Description
xEnabled	BOOL	TRUE – function block enabled.
xWriteActive	BOOL	TRUE - Executing parameter write function.
xWriteDone	BOOL	TRUE - Last parameter wrote successfully.
xWriteAllDone	BOOL	TRUE - All parameter value over written successfully
xReadActive	BOOL	TRUE - Parameter read function is active.
xReadAllDone	BOOL	TRUE – All parameter value feedback updated successfully
xError	BOOL	TRUE - Actuator unit in error state.
diCurrentPosition	DINT	Actuator unit current position feedback Linear actuator unit: - unit: [mm] - gradient: 0.01 Rotative actuator unit: - unit: [°] - gradient: 0.1
diCurrentSpeed	DINT	Actuator unit current speed feedback Linear actuator unit: - unit [mm/s] -gradient: 0.01 Rotative actuator unit: - unit [rpm] -gradient: 0.1
diCurrentForce	DINT	Actuator unit current feed force feedback Linear actuator unit: - unit [N] -gradient: 0.01 Rotative actuator unit: - unit [Nm] -gradient: 0.1
diDDTemperature	DINT	Actuator unit current temperature feedback Unit: [°C].
diDDCurrent	DINT	Actuator unit present value of current Unit: [A].
diDDVoltage	DINT	Actuator unit present voltage feedback Unit: [V].
diDDCyclesTotal	DINT	Number (dec) of completed movement cycles of the actuator unit since delivery
diDDCyclesSinceReset	DINT	Number (dec) of completed movement cycles of the actuator unit since last reset command
diDDMileageTotal	DINT	Mileage of the actuator unit since delivery Linear actuator unit: unit [km], 0.000001. Rotative actuator unit: unit [r], 0.001.
diDDMileageSinceReset	DINT	Mileage of the actuator unit since last reset command Linear actuator unit: unit [km], 0.000001. Rotative actuator unit: unit [r], 0.001.
diNumOfStorageOperation	DINT	Total number of permanent storage processes in the flash memory since delivery

Tag Name	Data Type	Function Description
aErrorCode	Array of WORD[16]	Error code of actuator unit Refer Chapter-3.5.4 for detailed description of Error code
uiLocalError	UINT	Function block Local Error 1001 – bySpeedIN Parameter limit out of range error. 1002 – bySpeedOut Parameter limit out of range error. 1003 – byForce Parameter limit out of range error. 1004 – Read_Parameter Error. 1005 - Write_Parameter Error. 1006 - “Position Value” Error. Check If Start Press Position value is Higher than the End Position Value. 1007 - “Position Value” Error. Check If Intermediate Position value is Higher than the End Position Value. 1008 – “PQI” Error. Check the device connection.

3.5.4 Diagnosis code details of Simplified Motion Series

Event code	Event type	Mode	Device status	Event name
0x1000	Error	Event appears (disappears)	Failure	General malfunction – unknown error
0x1801	Error	Event appears (disappears)	Failure	Supply voltage error – Power supply voltage too high
0x1802	Error	Event appears (disappears)	Failure	Supply voltage error – Power supply voltage too low or not connected
0x1803	Error	Event appears (disappears)	Failure	Supply voltage error – Logic supply voltage too high
0x1804	Error	Event appears (disappears)	Failure	Supply voltage error – Logic supply voltage too low
0x1805	Error	Event appears (disappears)	Failure	Power consumption error – I2t exceeded
0x1806	Error	Event appears (disappears)	Failure	Voltage error – DC link voltage too low
0x1810	Error	Event appears (disappears)	Failure	Firmware download error – Firmware package not valid (e.g. checksum error)
0x1811	Error	Event appears (disappears)	Failure	Firmware download error – Firmware package is not compatible to device and/or hardware
0x1812	Error	Event appears (disappears)	Failure	Firmware download error – Unknown slot
0x1813	Error	Event appears (disappears)	Failure	Firmware download error – Empty slot

Event code	Event type	Mode	Device status	Event name
0x1814	Error	Event appears (disappears)	Failure	Firmware download error – Firmware update not possible in this state (e.g. power stage is active)
0x1815	Error	Event appears (disappears)	Failure	Firmware download error – Package currently in use (e.g. by file transfer)
0x1816	Error	Event appears (disappears)	Failure	Firmware download error – Error initiating firmware update
0x1817	Error	Event appears (disappears)	Failure	Firmware download error – Error resuming firmware update (e.g. write error in file system)
0x1820	Error	Event appears (disappears)	Failure	Parameter package download error – Empty slot
0x1821	Error	Event appears (disappears)	Failure	Parameter package download error – Parameter set not readable (e.g. checksum error)
0x1822	Error	Event appears (disappears)	Failure	Parameter package download error – Parameter set incompatible
0x1823	Error	Event appears (disappears)	Failure	Parameter package download error – Parameter not found
0x4000	Error	Event appears (disappears)	Failure	Temperature fault – Overload
0x4210	Warning	Event appears (disappears)	Outside the specification	Device temperature overrun – Clear source of heat
0x4220	Warning	Event appears (disappears)	Outside the specification	Device temperature underrun – Insulate Device
0x8CA0	Notification	Simple message	Device is operating properly	Application information – Device not referenced
0x8CA1	Notification	Simple message	Device is operating properly	Application information – Reference running
0x8CA2	Warning	Event appears (disappears)	Outside the specification	Application warning – Start Press position out of range
0x8CA3	Warning	Event appears (disappears)	Outside the specification	Application warning – End position out of range
0x8CA4	Warning	Event appears (disappears)	Outside the specification	Power consumption warning – I2t close to be exceeded

4 Configuring Channel Input and Output in CODESYS

4.1 Configuring the RefInput & RefOutput Interface Tags of Function Blocks



Note

- Function Block require only 1st byte of Process Data IN & Process Data Out
- Because Modbus reads Process Data as WORD, it must be divided to BYTE and assigned to FB.

Process Data IN

Bit	15	...	4	3	2	1	0
Process data	not used		ProcessDataVariable (PDV)				
Data content			State "Inter-mediate"	State "De-vice"	State "Move"	State "Out"	State "In"
Index			0x28 (40)				
Sub-Index			5	4	3	2	1
Data type			BooleanT				

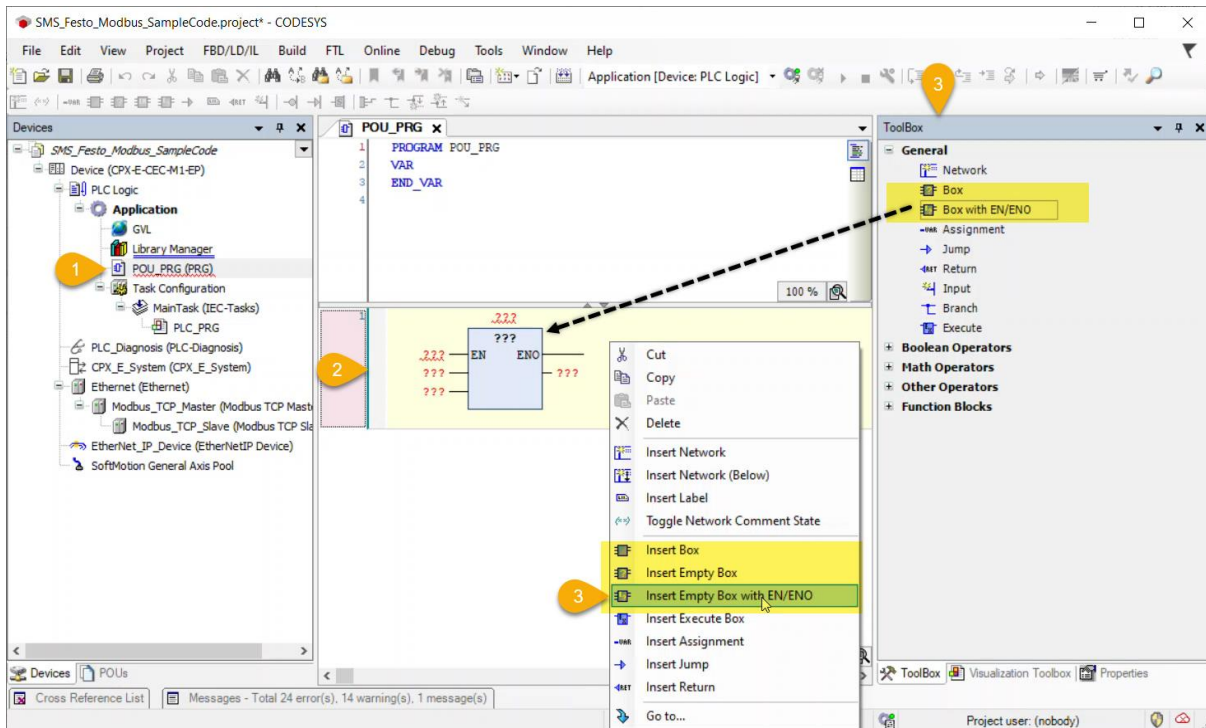
Mapping the process data IN

Process Data OUT

Bit	15	...	4	3	2	1	0
Process data	not used		ProcessDataVariable (PDV)				
Data content			Move "Inter-mediate"	not used	Quit Error	Move "Out"	Move "In"
Index			0x29 (41)				
Sub-Index			5	...	3	2	1
Data type			BooleanT				

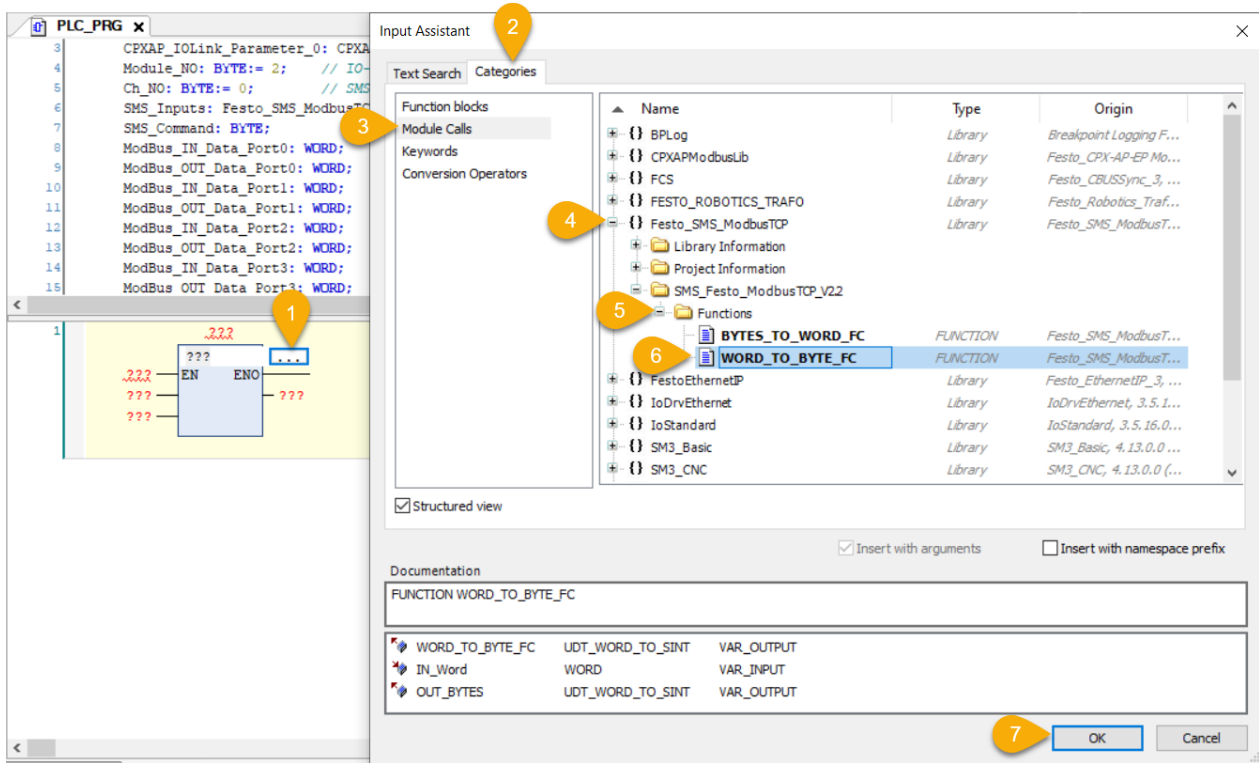
Mapping the process data OUT

Step – 1



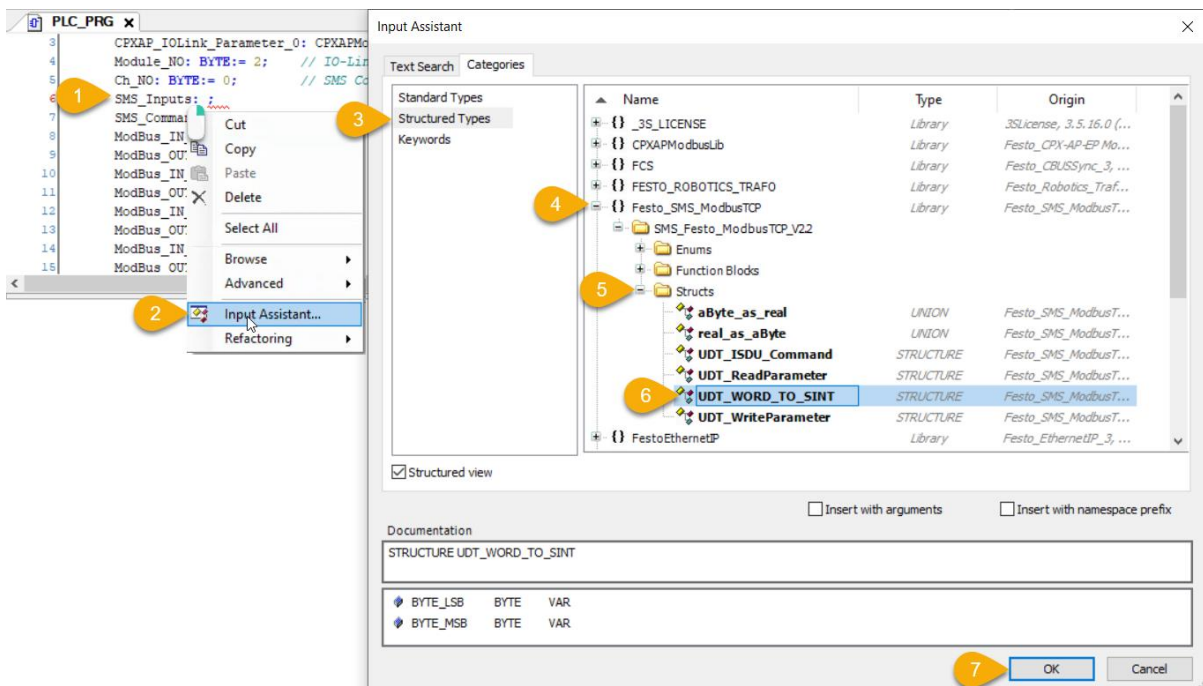
1. Double click on **POU_PRG** from Application.
2. Right click on the **Network** from Main window.
3. Click on **Insert Empty Box** to open the Input assistant window.
or
Drag & Drop the "Box with EN/EO" block from toolbox to program area.

Step – 2



1. Click on the Box, Input Assistant window pops-up on screen.
2. Click on **Categories** tab from Input Assistant window.
3. Click on **Module Calls** from Categories tab.
4. Expand the **Festo_SMS_ModbusTCP** library and then expand the **SMS_Festo_ModbusTCP_V2.2** folder
5. Expand the **Functions** folder to view the Functions.
6. Select **WORD_TO_BYTE_FC**
7. Click on **OK** button to call the function block.

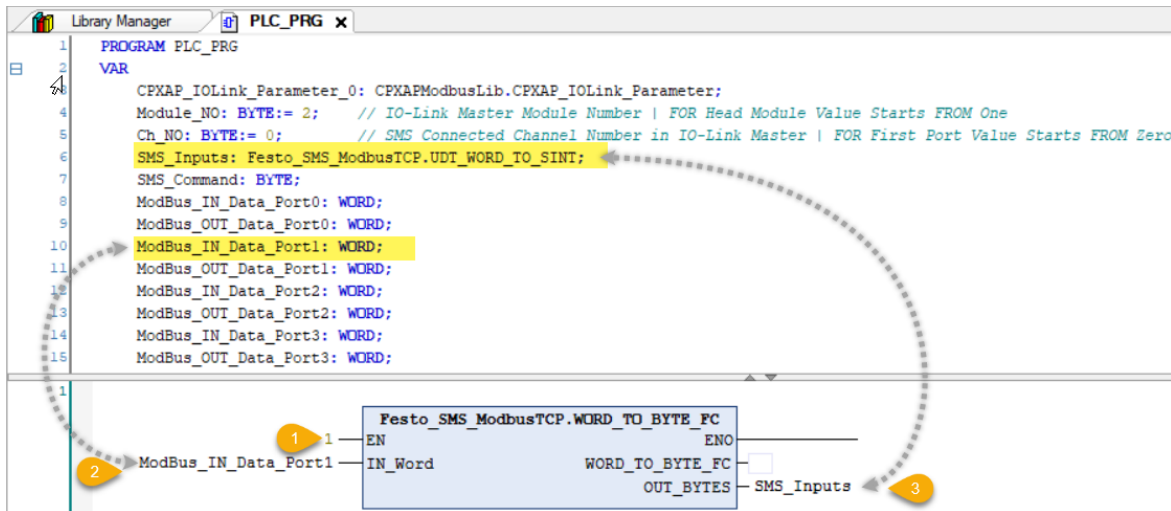
Step – 3



- Create the Tag in Local Program as given above image.

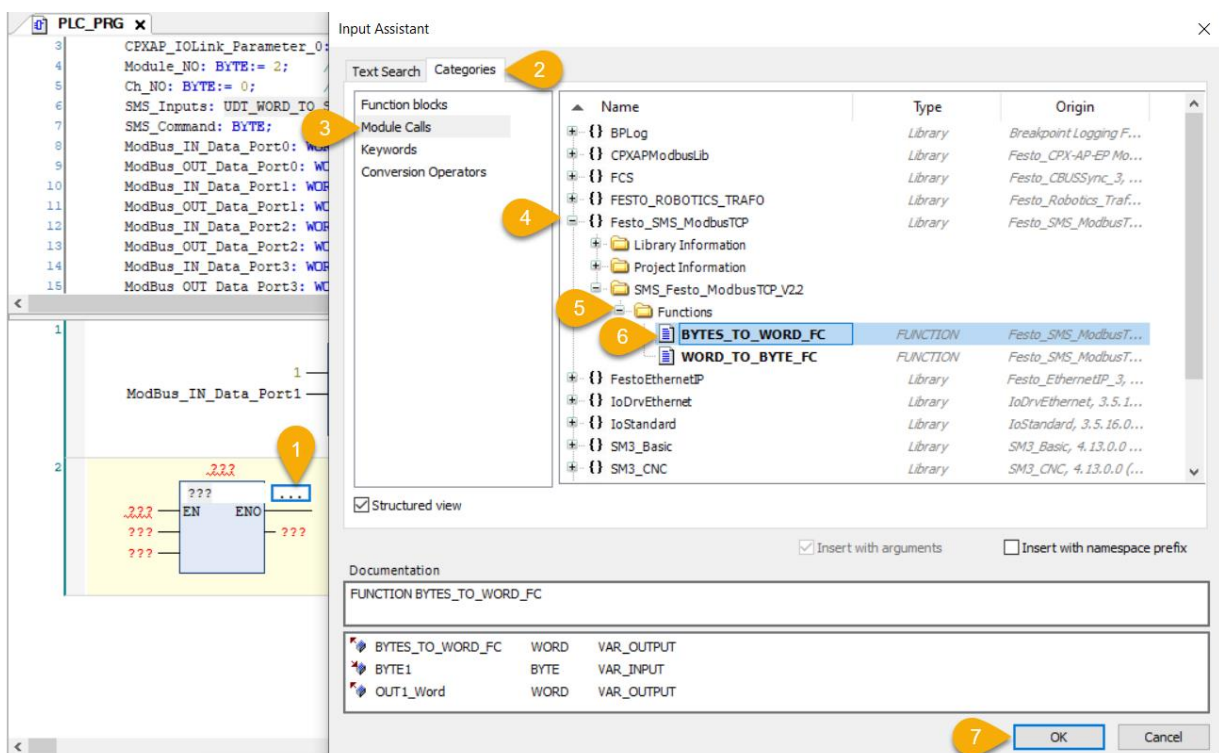
1. Enter the name as “SMS_Inputs” and add colon and then right click after the colon to define Tag type.
2. Select Input Assistant and window will open.
3. Click on **Structured Types** from Categories tab.
4. Expand the **Festo_SMS_ModbusTCP** library and expand the **SMS_Festo_ModbusTCP_V2.2** folder
5. Expand the **Structs** folder to view the Structs.
6. Select **UDT_WORD_TO_SINT**
7. Click on **OK** button to call the function block.

Step – 4

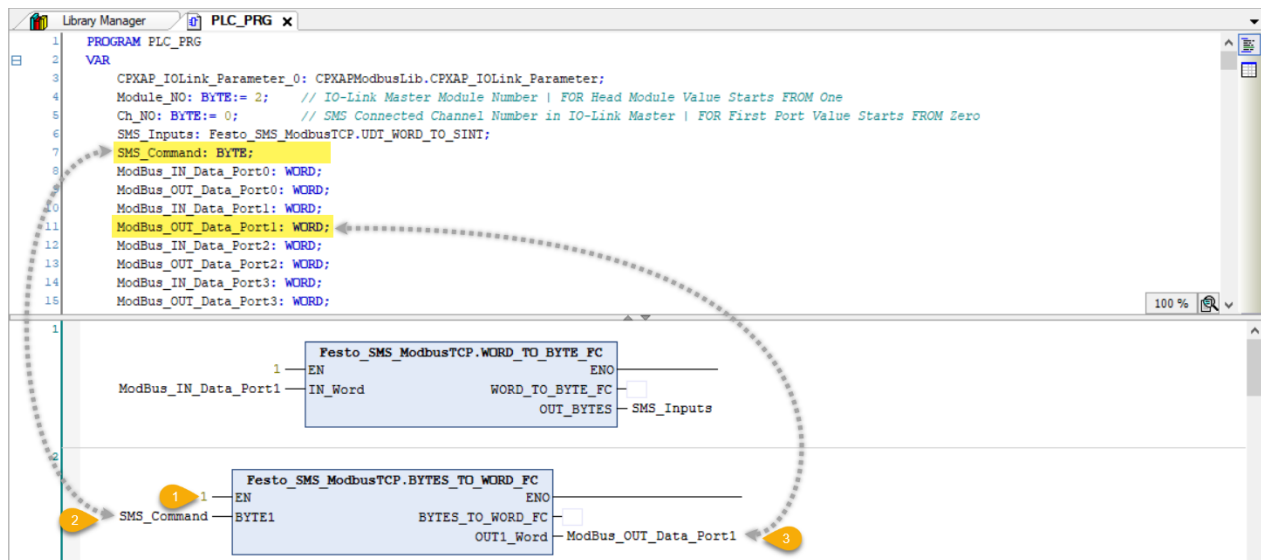


1. Always keep the FC block Enabled by assigning “1” to EN.
2. Enter ModBus_IN_Data_Port1 in IN_Word to assign the local variable input tag. (The port that SMS is linked to must be chosen by the user).
3. Enter SMS_Inputs in OUT_BYTES to assign the local variable input tag.

Step – 5



1. Click on the Box, Input Assistant window pops-up on screen.
2. Click on **Categories** tab from Input Assistant window.
3. Click on **Function blocks** from Categories tab.
4. Expand the **Festo_SMS_ModbusTCP** library and expand the **SMS_Festo_ModbusTCP_V2.2** folder
5. Expand the **Functions** folder to view the Functions.
6. Select **BYTE_TO_WORD_FC**
7. Click on **OK** button to call the function block.



1. Always keep the FC block Enabled by assigning “1” to EN.
2. Enter SMS_Command in BYTE1 to assign the local variable input tag.
3. Enter ModBus_OUT_Data_Port1 in OUT1_Word to assign the local variable input tag. (The port that SMS is linked to must be chosen by the user).



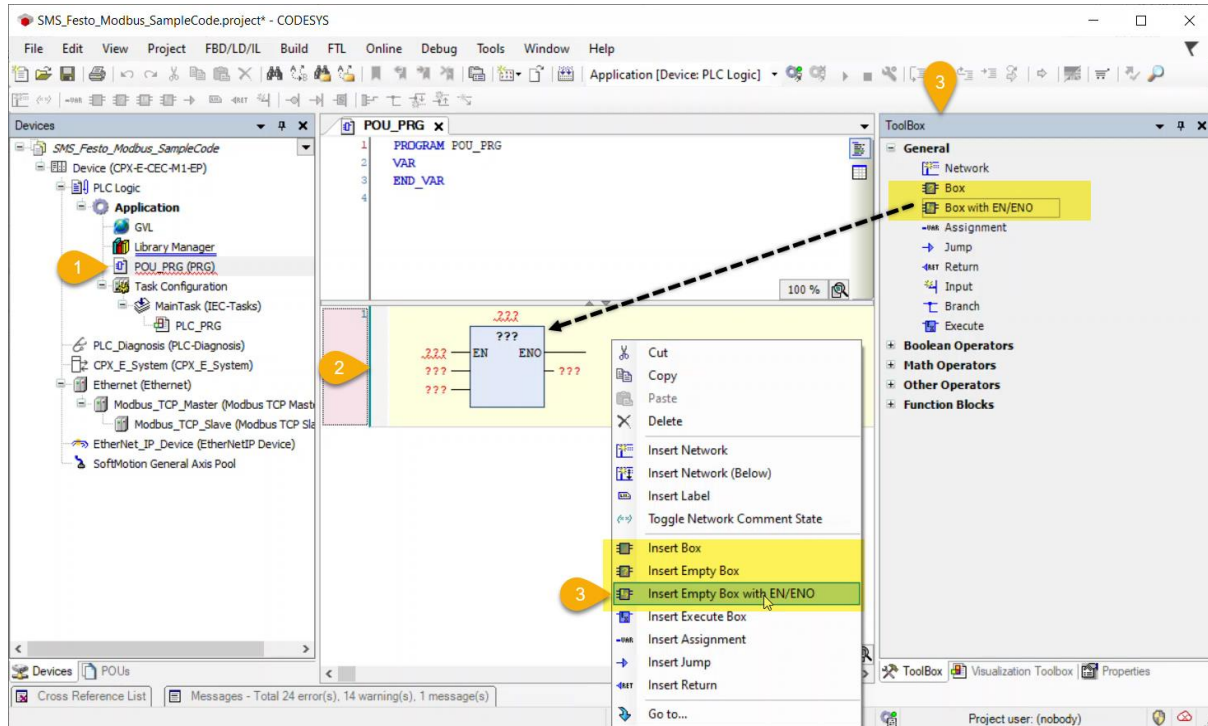
Note

- Refer [Chapter 3.3](#) to use Mapped I/O tags

5 Configuration of Festo_SMS_Modbus Function blocks in CODESYS

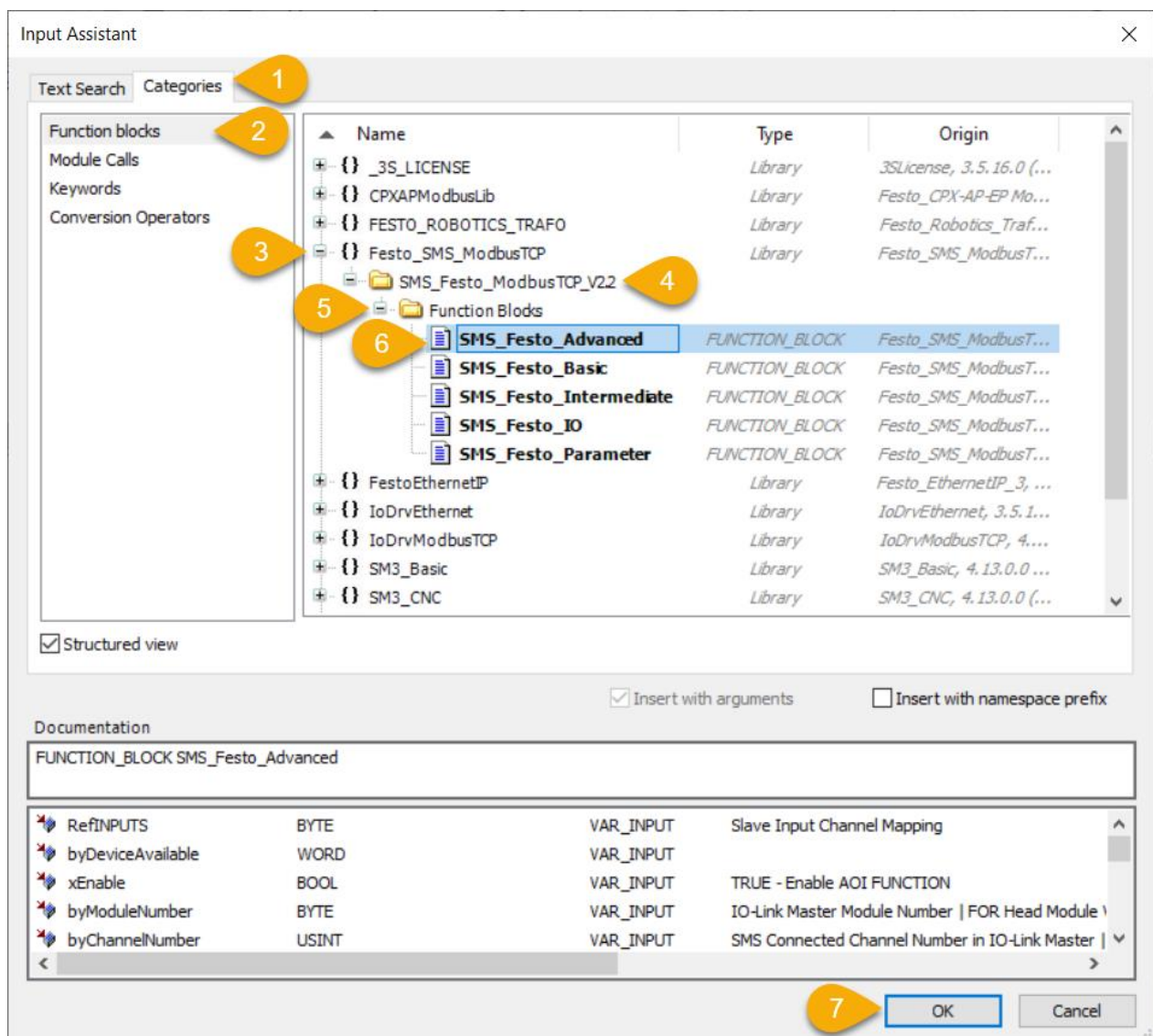
5.1 Configuring the SMS_Festo_Advanced Function Block Input & Output Interface Tags

Step – 1



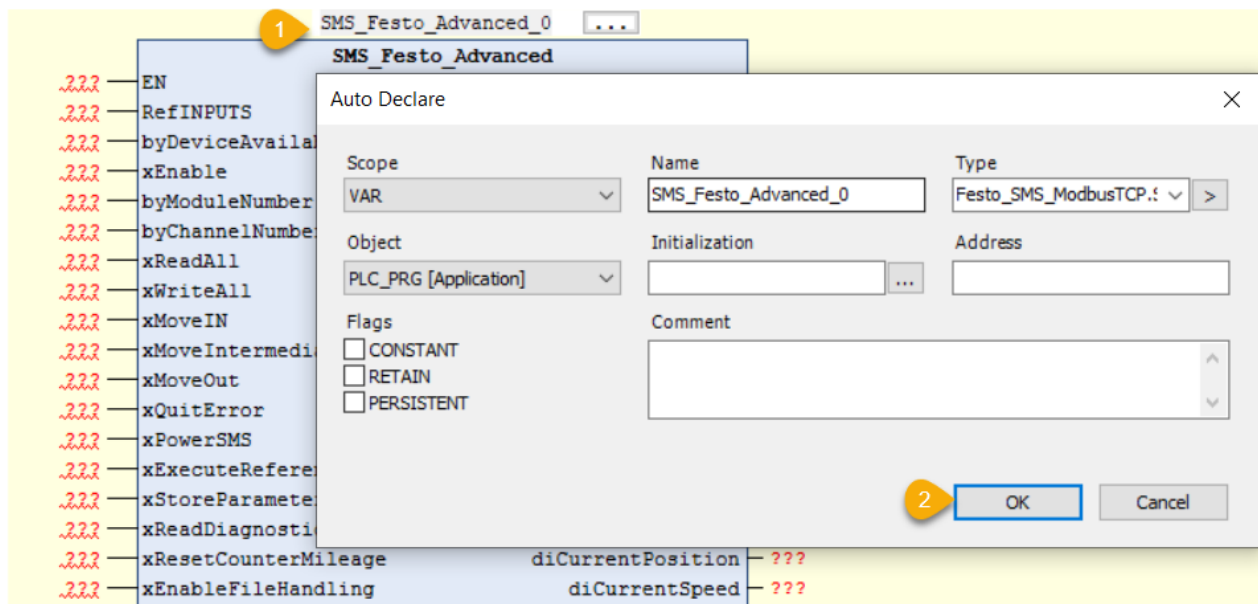
1. Double click on **POU_PRG** from Application.
2. Right click on the **Network** from Main window.
3. Click on **Insert Empty Box** to open the Input assistant window.
or
Drag & Drop the “Box with EN/EO” block from toolbox to program area.

The opened Input Assistant window will be displayed as show below step-2.

Step – 2

1. Click on **Categories** tab from Input Assistant window.
2. Click on **Function blocks** from Categories tab.
3. Expand the **Festo_SMS_ModbusTCP** library.
4. Expand the **SMS_Festo_ModbusTCP_V2.2** folder
5. Expand the **Function Blocks** folder to view the Function blocks.
6. Select **SMS_Festo_Advanced** block
7. Click on **OK** button to call the function block.

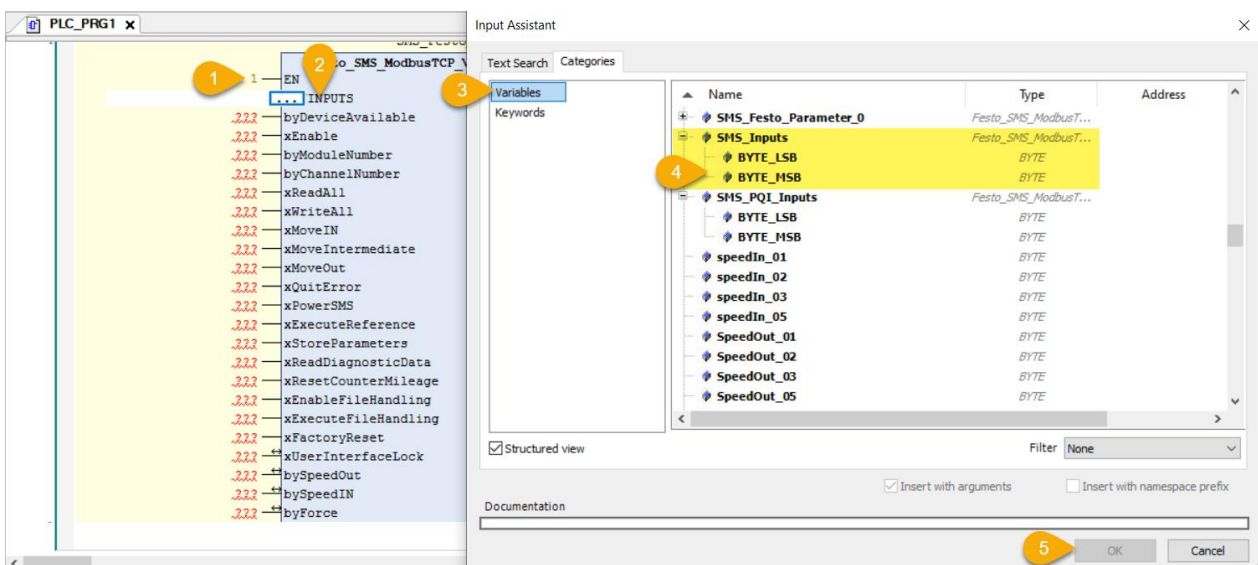
The called function block shown in below Step-3

Step – 3

1. Enter the name of the FB instance & press enter key for Auto declare the tag name.
2. Click **“OK”** button from the auto declare window.

Step – 4**Note**

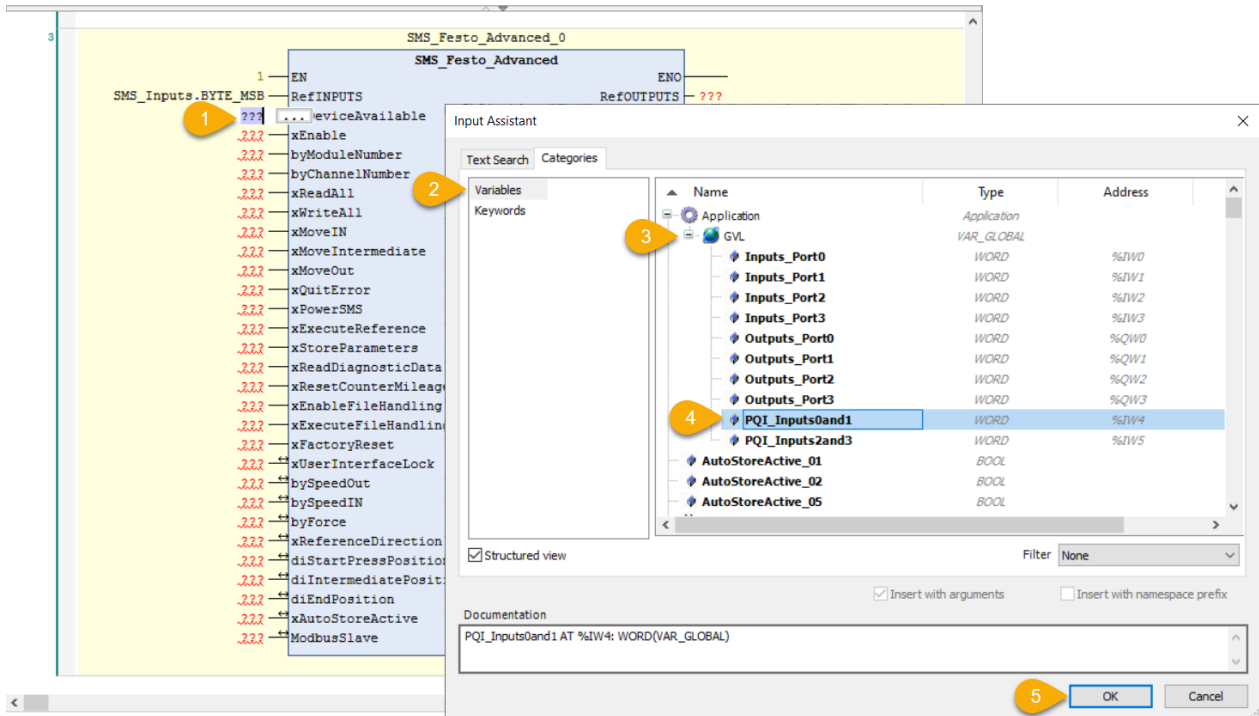
- Function Block require only 1st byte of Process Data IN & Process Data Out.
- Because Modbus reads Process Data as WORD, it must be divided to BYTE and assigned to FB.

Process Data IN

1. Always enable the function block by assigning value 1 in **“EN”** input.
2. Click the **“Browse”** button of the FB Input Tag.
3. Select **“Variables”** from categories list.
4. Expand **“SMS_Inputs”** from the list shown in window. Select **“SMS_Inputs.BYTE_MSB”**. (Refer the above Note)
5. Click **“OK”** to confirm the selection.

Step – 5

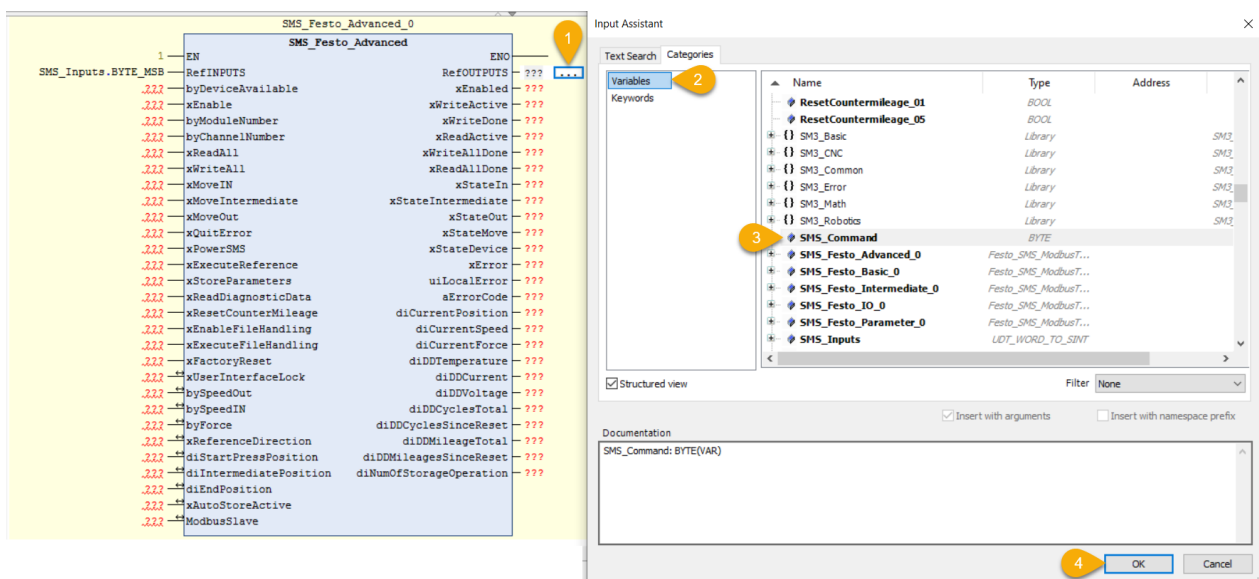
PQI – Process Data IN



1. Click the **“Browse”** button of the FB Input Tag.
2. Select **“Variables”** from categories list.
3. Expand **“Application”** --> **“GVL”**
4. Select **“PQI_Inputs0and1”** from the list shown in window. (The port that SMS is linked to must be chosen by the user, Example: PQI of Port 0 and Port 1 will be available in one word).
Refer – [Chapter 3.2 Step-5](#) or [Technical Appendix 9.1](#)
5. Click **“OK”** to confirm the selection.

Step – 6

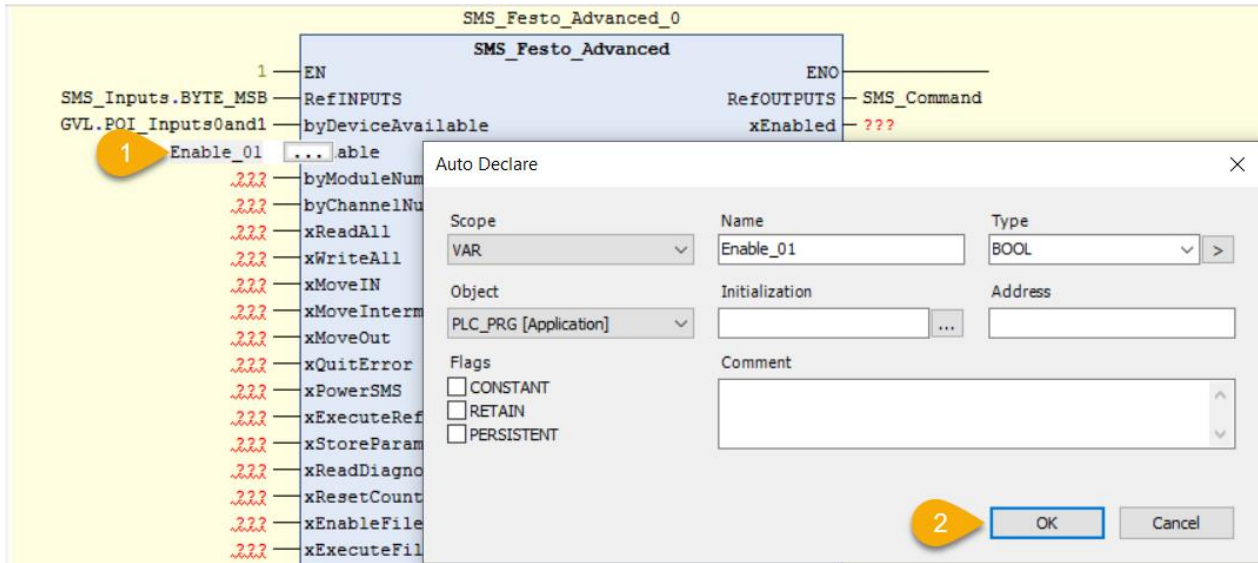
Process Data Out



1. Click the **“Browse”** button of the FB Input Tag.
2. Select **“Variables”** from categories list.
3. Select **“SMS_Command”** from list.

- Click “OK” to confirm the selection.

Step – 7



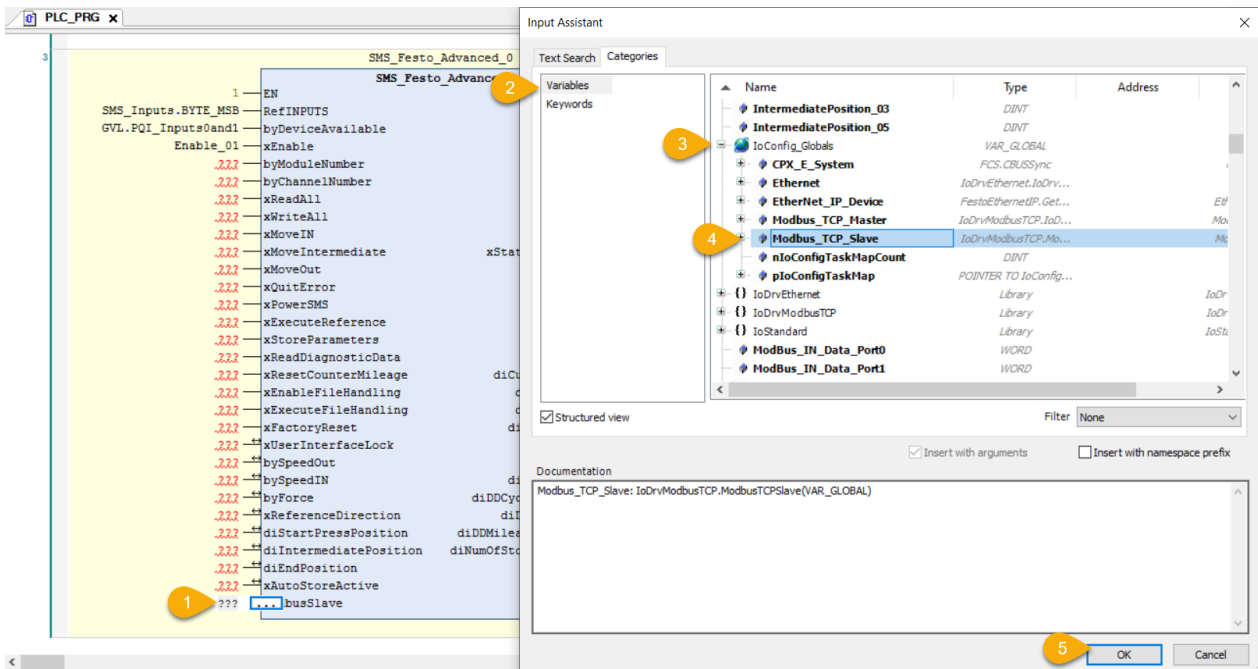
- Enter the tag name for “xEnabled” input tag & press enter to open Auto Declare window.
- Click “OK” to confirm the entered tag name.



Note

- Repeat Step – 7 for different input & output tags in function block.

Step – 8



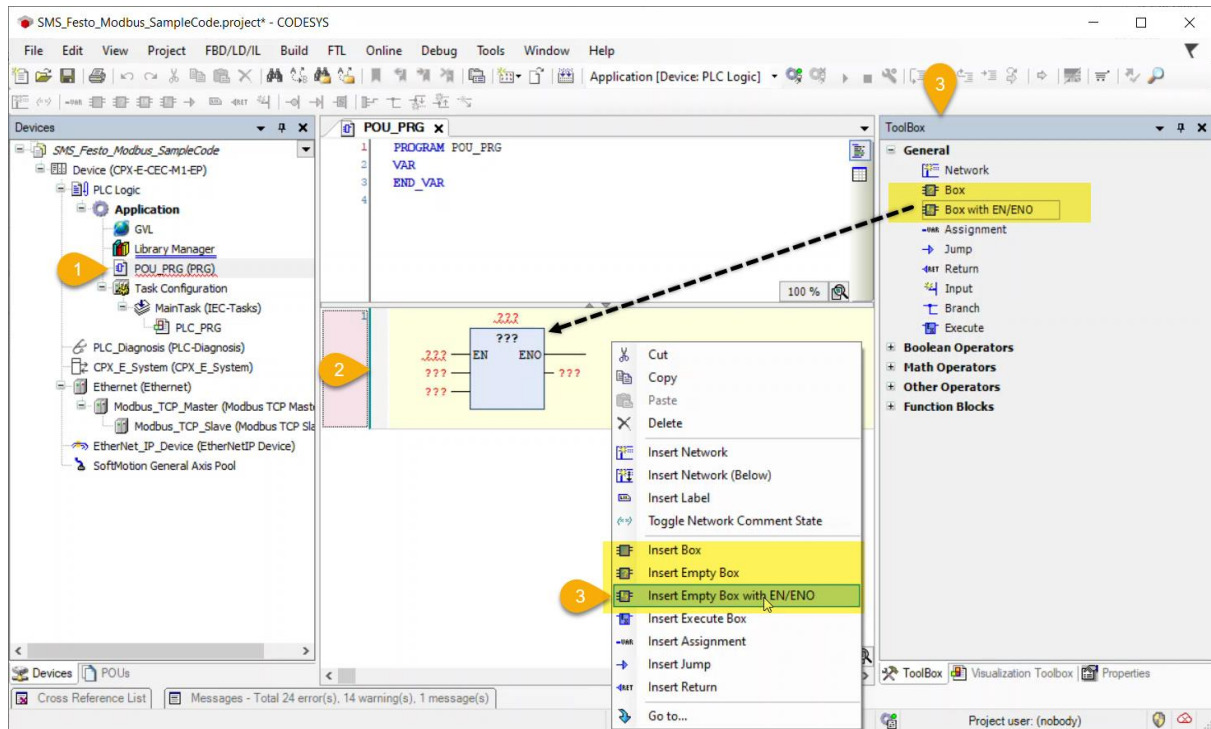
- Click the “Browse” button of the FB Input Tag.
- Select “Variables” from categories list.
- Expand “IoConfig_Globals” from the list shown in window.
- Select “Modbus_TCP_Slave” from IoConfig_Globals list.
- Click “OK” to confirm the selection.

Once all steps completed successfully the **SMS_Festo_Advanced** Function block will be look like below.



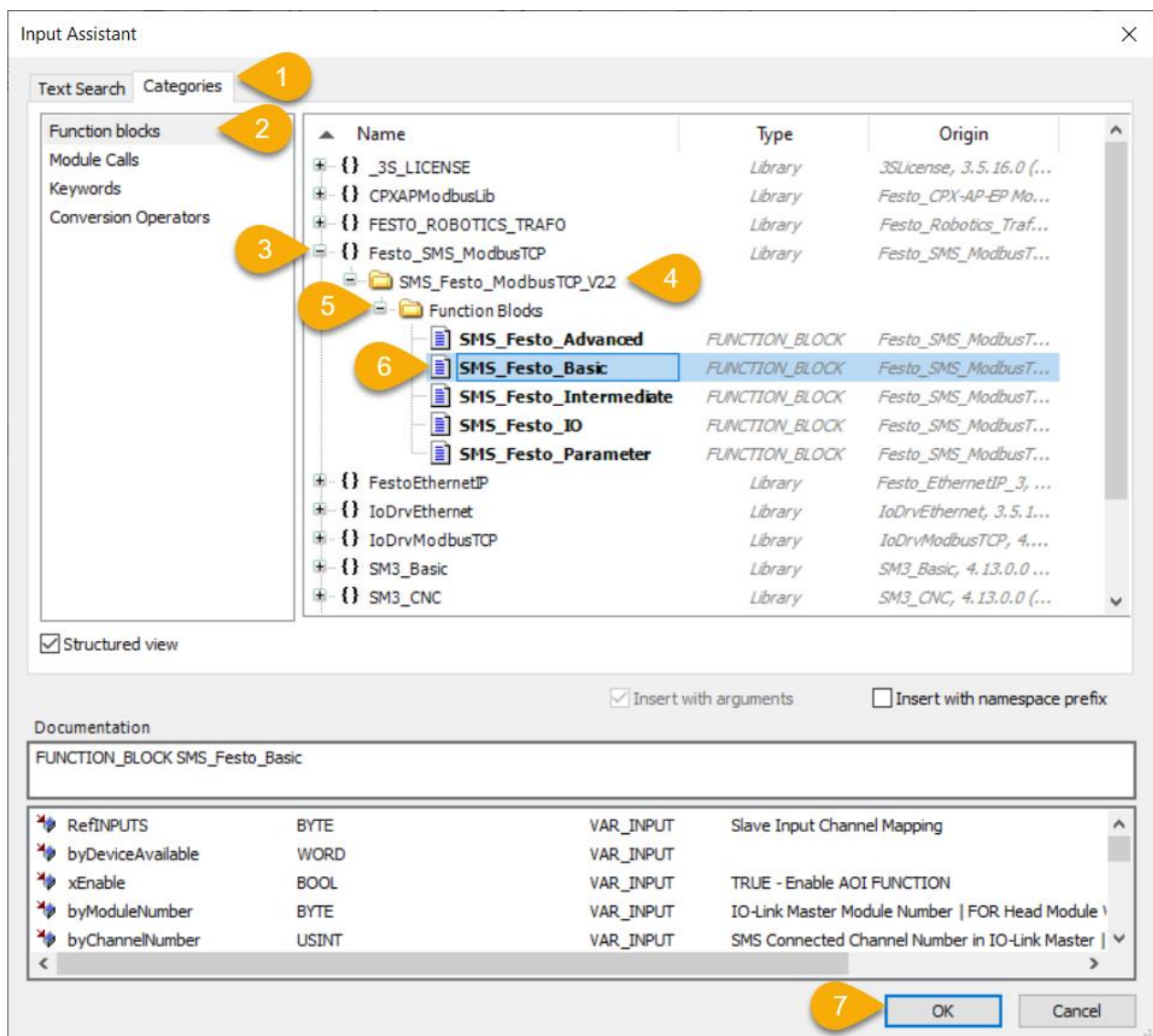
5.2 Configuring the SMS_Festo_Basic Function Block Input & Output Interface Tags

Step – 1



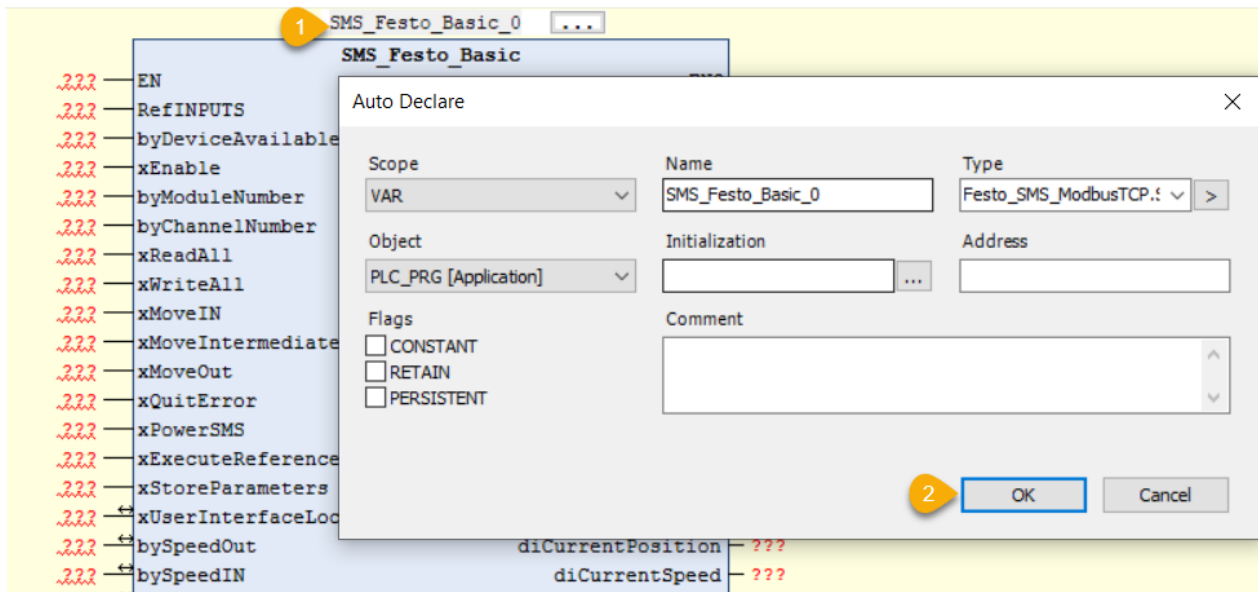
1. Double click on **POU_PRG** from Application.
2. Right click on the **Network** from Main window.
3. Click on **Insert Empty Box** to open the Input assistant window.
or
Drag & Drop the “Box with EN/EO” block from toolbox to program area.

The opened Input Assistant window will be displayed as show below step-2.

Step – 2

1. Click on **Categories** tab from Input Assistant window.
2. Click on **Function blocks** from Categories tab.
3. Expand the **Festo_SMS_ModbusTCP** library and expand the **SMS_Festo_ModbusTCP_V2.2** folder
4. Expand the **Function Blocks** folder to view the Function blocks.
5. Select **SMS_Festo_Basic** block
6. Click on **OK** button to call the function block.

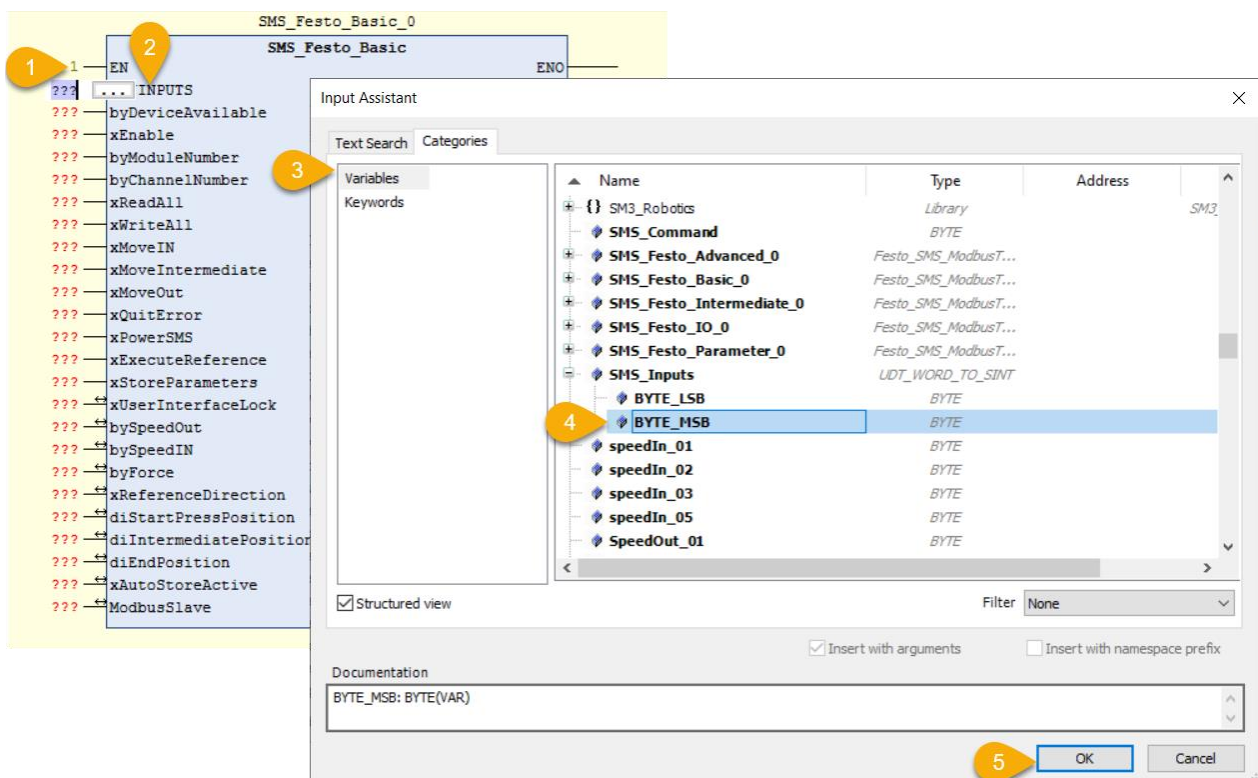
The called function block shown in below Step-3

Step – 3

1. Enter the name of the FB instance & press enter key for Auto declare the tag name.
2. Click “OK” button from the auto declare window.

Step – 4**Note**

- Function Block require only 1st byte of Process Data IN & Process Data Out.
- Because Modbus reads Process Data as WORD, it must be divided to BYTE and assigned to FB.

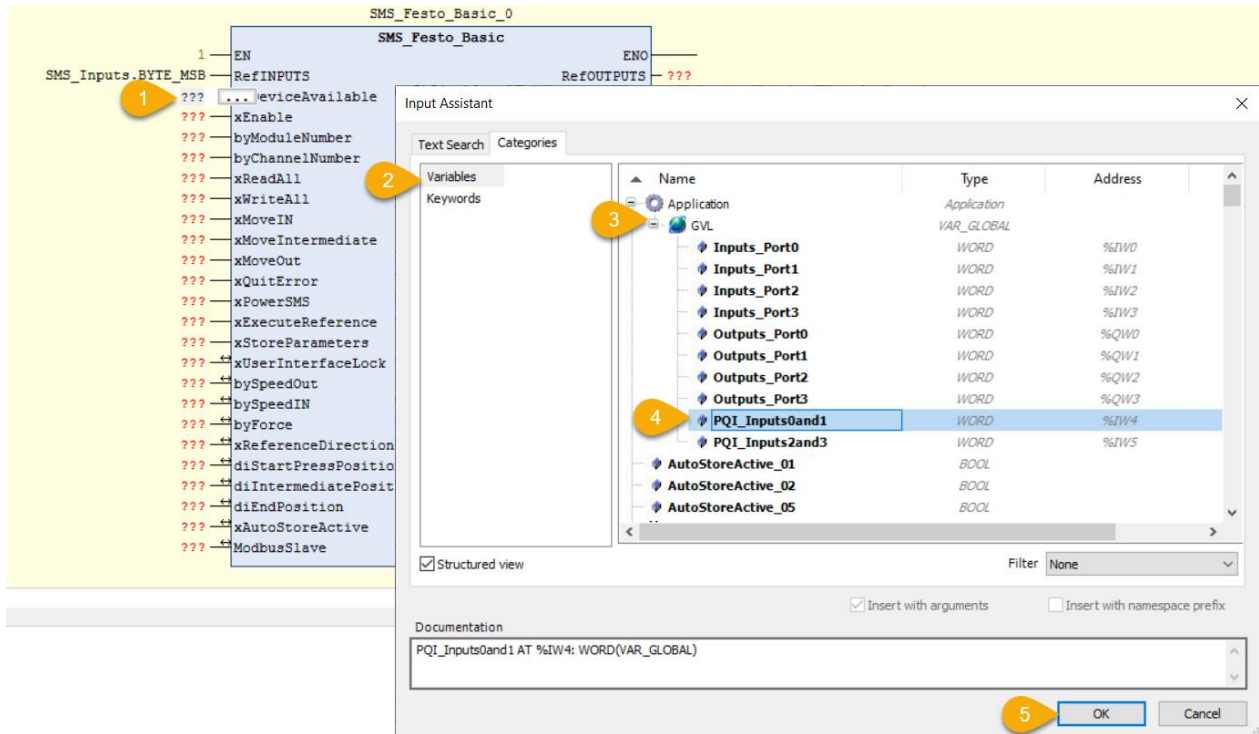
Process Data IN

1. Always enable the function block by assigning value 1 in “EN” input.
2. Click the “Browse” button of the FB Input Tag.

3. Select **"Variables"** from categories list.
4. Expand **"SMS_Inputs"** from the list shown in window.
Select **"SMS_Inputs.BYTE_MSB"**. (Refer the above Note)
5. Click **"OK"** to confirm the selection.

Step – 5

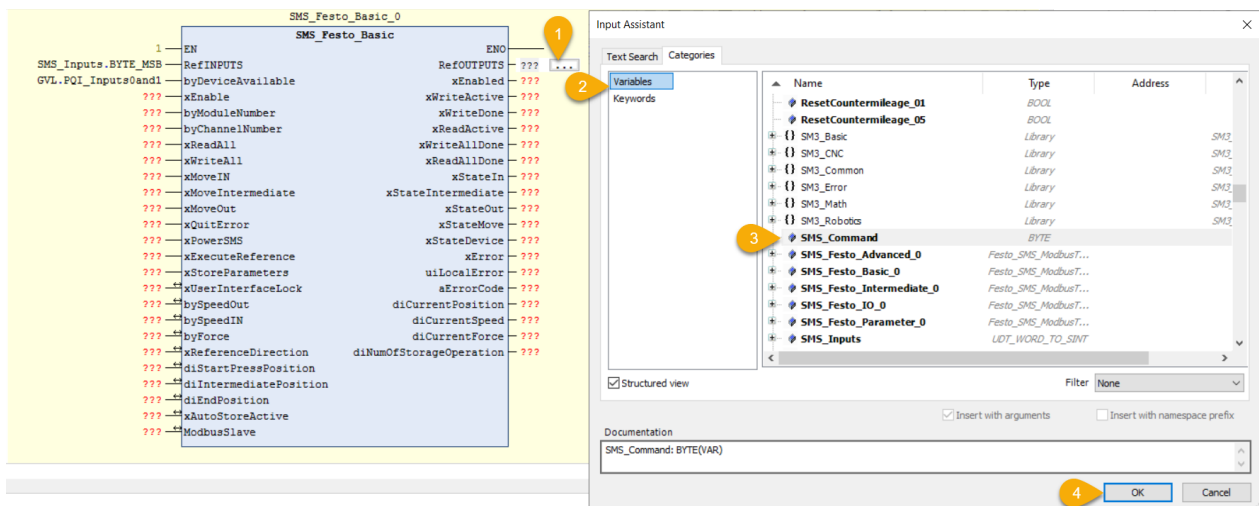
PQI – Process Data IN



1. Click the **"Browse"** button of the FB Input Tag.
2. Select **"Variables"** from categories list.
3. Expand **"Application"** --> **"GVL"**
4. Select **"PQI_Inputs0and1"** from the list shown in window. (The port that SMS is linked to must be chosen by the user, Example: PQI of Port 0 and Port 1 will be available in one word).
Refer – [Chapter 3.2 Step-5](#) or [Technical Appendix 9.1](#)
5. Click **"OK"** to confirm the selection.

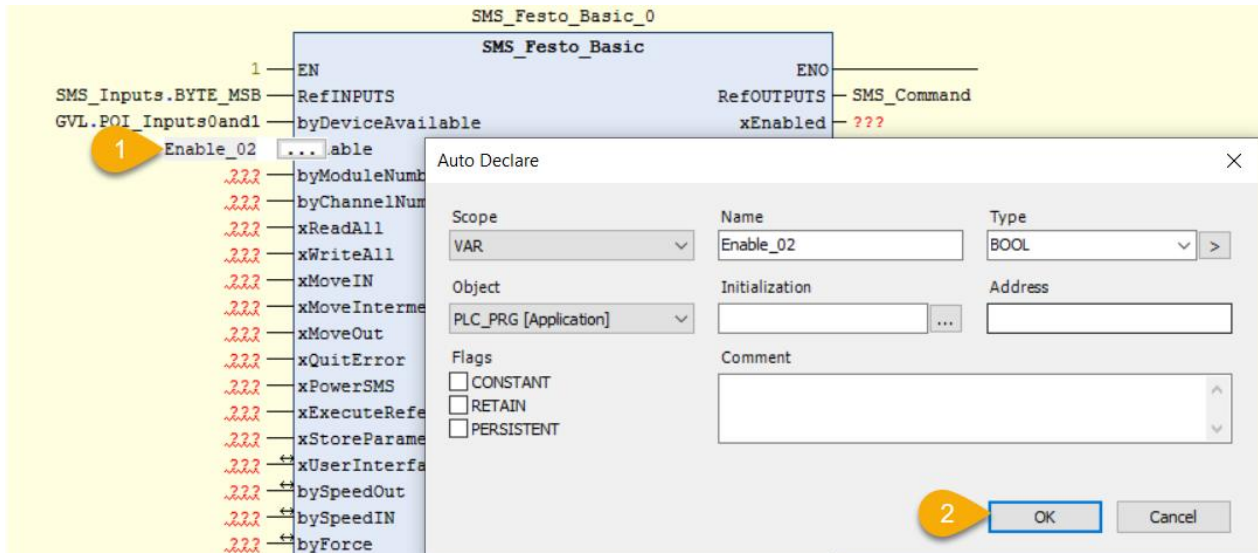
Step – 6

Process Data Out



1. Click the “**Browse**” button of the FB Input Tag.
2. Select “**Variables**” from categories list.
3. Select “**SMS_Command**” from list.
4. Click “**OK**” to confirm the selection.

Step – 7



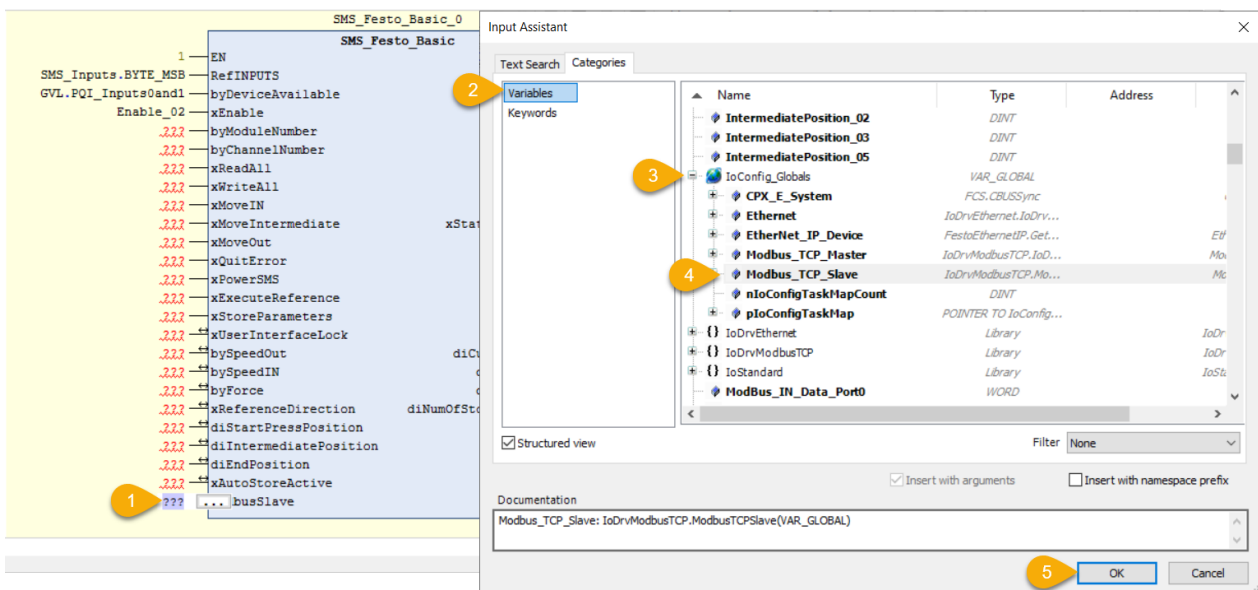
1. Enter the tag name for “**xEnable**” input tag & press enter to open Auto Declare window.
2. Click “**OK**” to confirm the entered tag name.



Note

- Repeat Step – 7 for different input & output tags in function block.

Step – 8



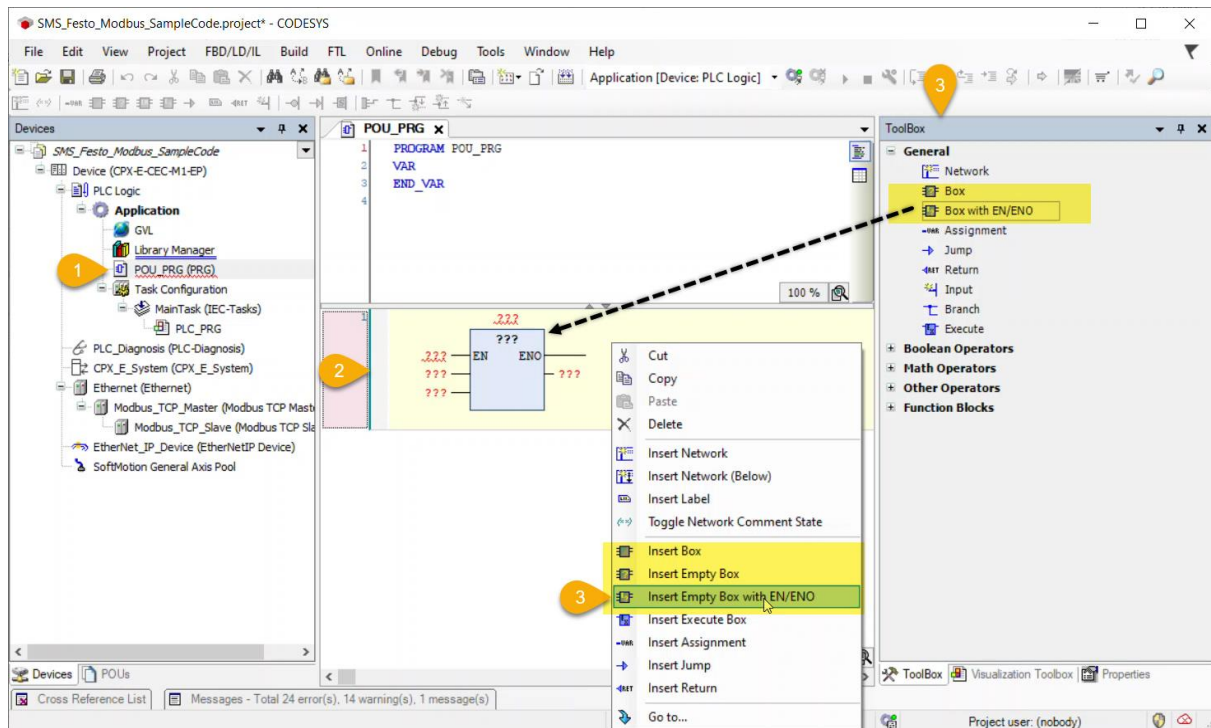
1. Click the “**Browse**” button of the FB Input Tag.
2. Select “**Variables**” from categories list.
3. Expand “**IoConfig_Globals**” from the list shown in window.
4. Select “**Modbus_TCP_Slave**” from IoConfig_Globals list.
5. Click “**OK**” to confirm the selection.

Once all steps completed successfully the **SMS_Festo_Basic** Function block will be look like below.



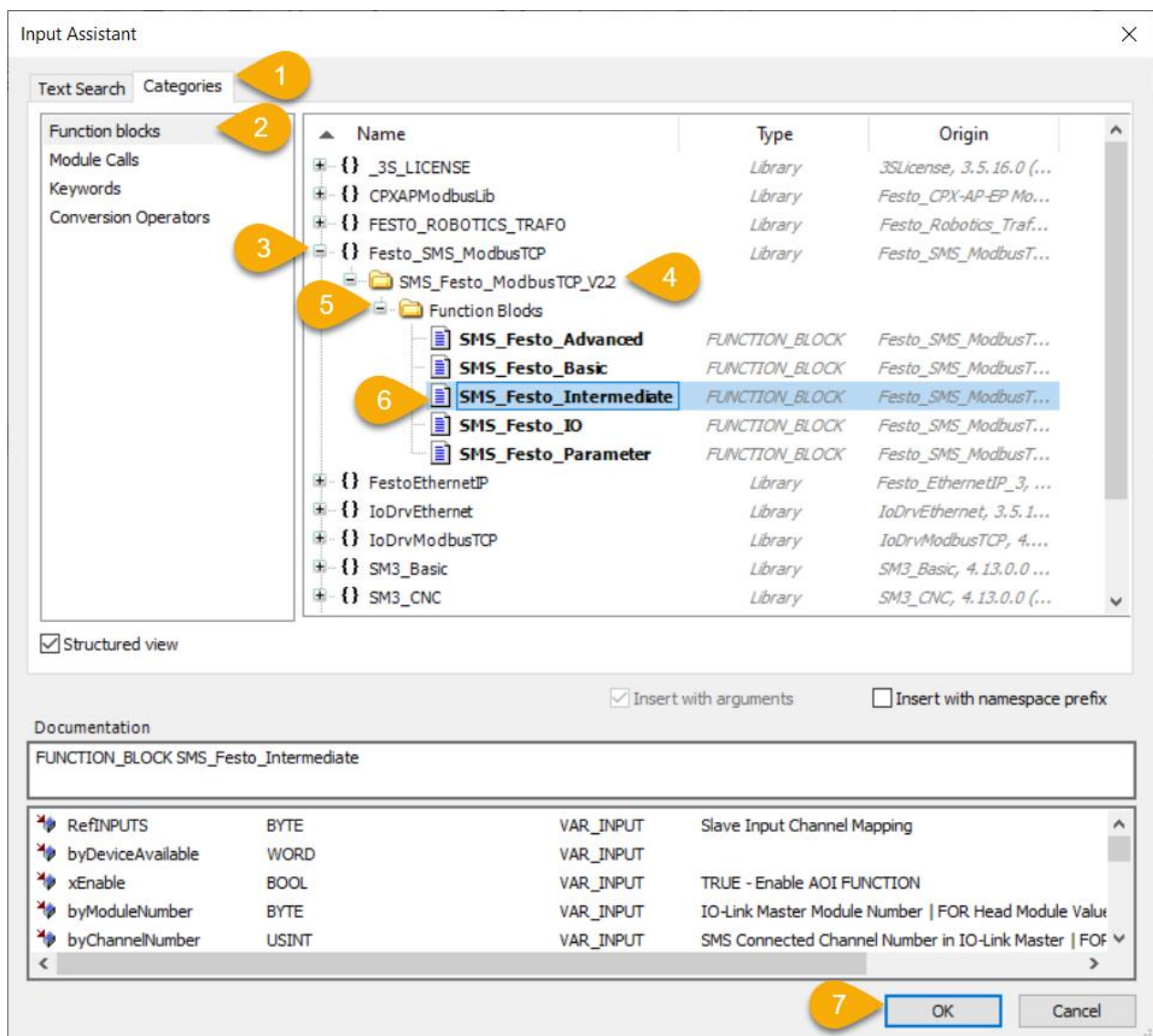
5.3 Configuring the SMS_Festo_Intermediate Function Block Input & Output Interface Tags

Step – 1



1. Double click on **POU_PRG** from Application.
2. Right click on the **Network** from Main window.
3. Click on **Insert Empty Box** to open the Input assistant window.
or
Drag & Drop the “Box with EN/EO” block from toolbox to program area.

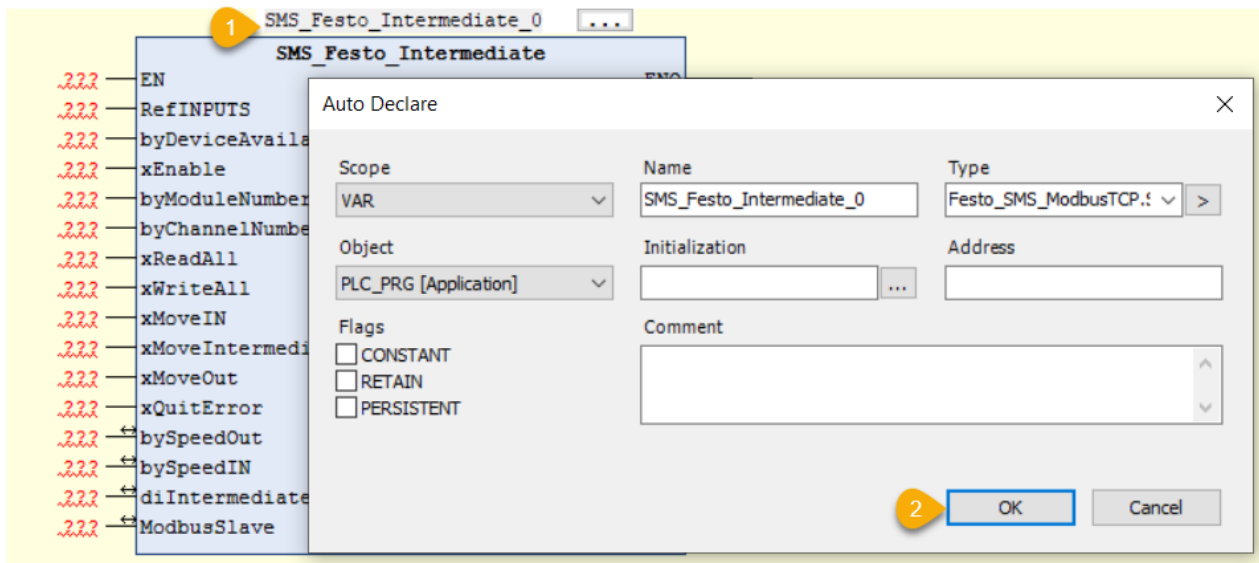
The opened Input Assistant window will be displayed as show below step-2.

Step – 2

1. Click on **Categories** tab from Input Assistant window.
2. Click on **Function blocks** from Categories tab.
3. Expand the **Festo_SMS_ModbusTCP** library and expand the **SMS_Festo_ModbusTCP_V2.2** folder
4. Expand the **Function Blocks** folder to view the Function blocks.
5. Select **SMS_Festo_Intermediate** block
6. Click on **OK** button to call the function block.

The called function block shown in below Step-3

Step – 3



1. Enter the name of the FB instance & press enter key for Auto declare the tag name.
2. Click **“OK”** button from the auto declare window.

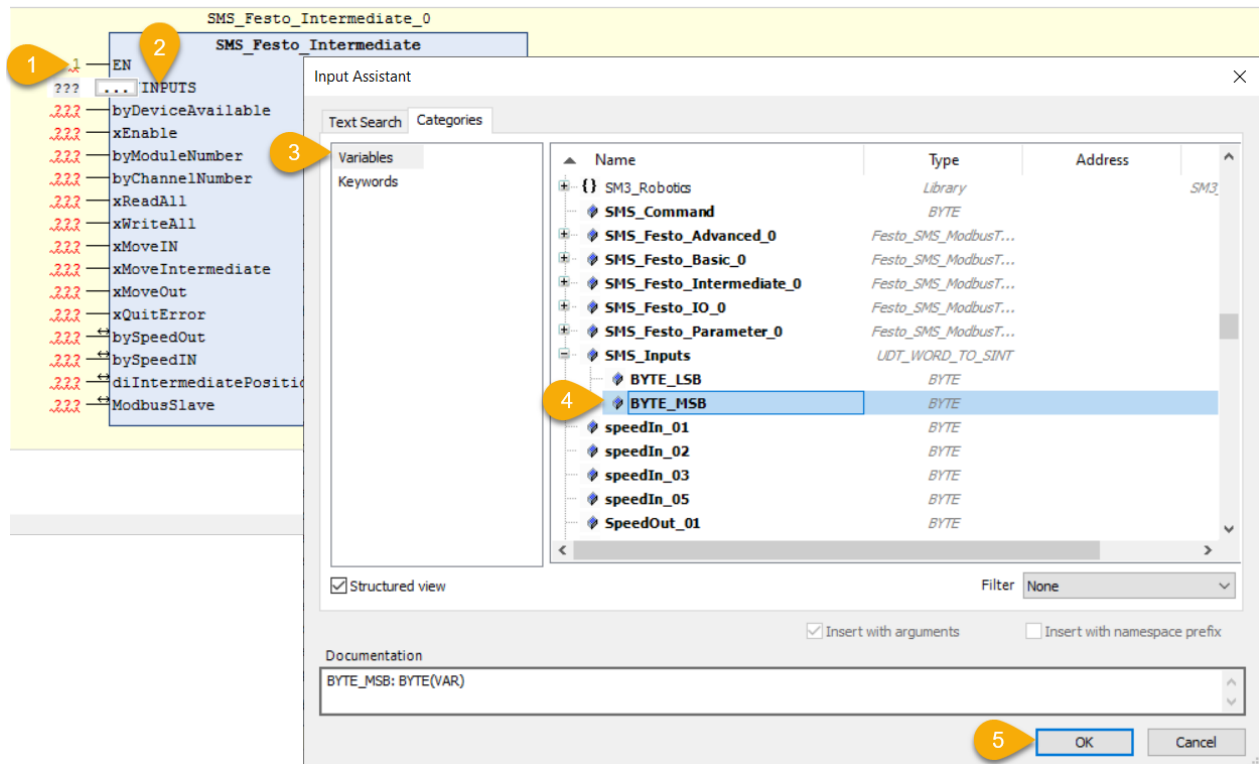
Step – 4



Note

- Function Block require only 1st byte of Process Data IN & Process Data Out.
- Because Modbus reads Process Data as WORD, it must be divided to BYTE and assigned to FB.

Process Data IN

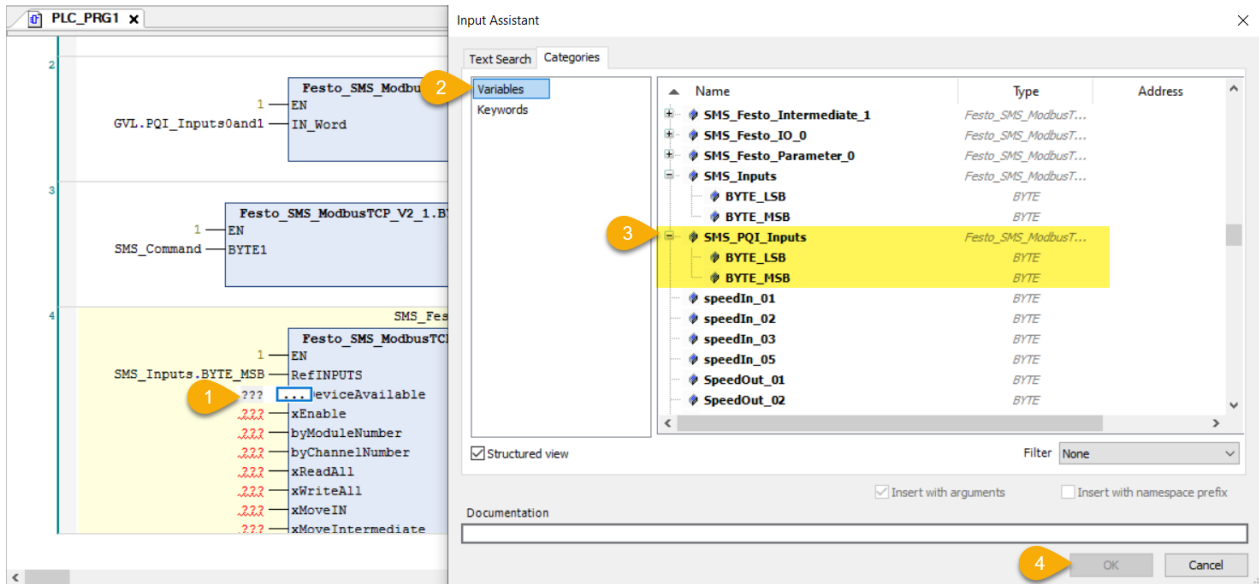


1. Always enable the function block by assigning value 1 in **“EN”** input.
2. Click the **“Browse”** button of the FB Input Tag.
3. Select **“Variables”** from categories list.

4. Expand “**SMS_Inputs**” from the list shown in window.
Select “**SMS_Inputs.BYTE_MSB**”. (Refer the above Note)
5. Click “**OK**” to confirm the selection.

Step – 5

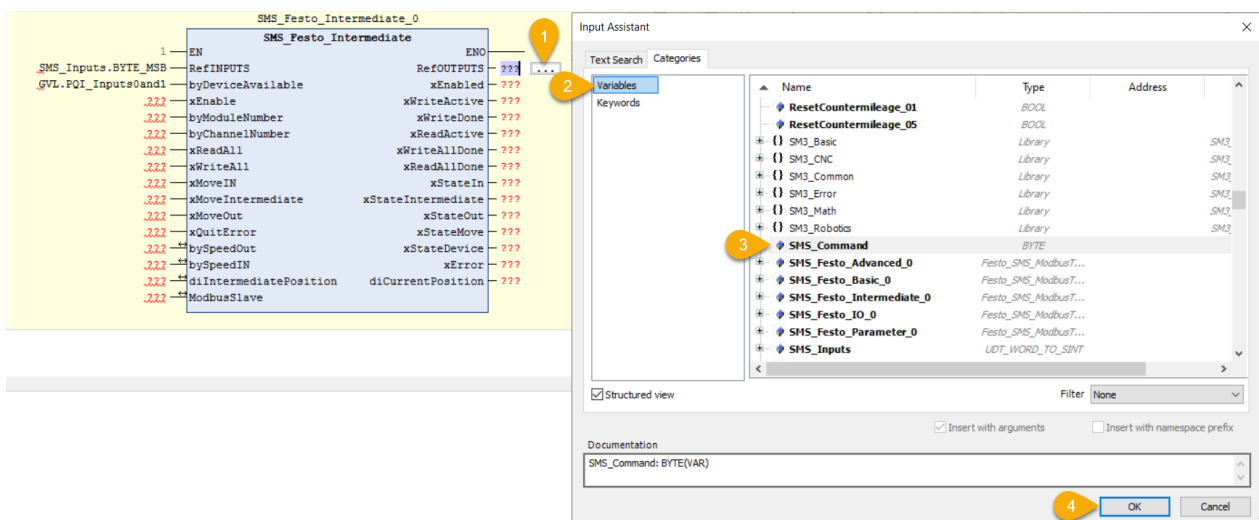
PQI – Process Data IN



1. Click the “**Browse**” button of the FB Input Tag.
2. Select “**Variables**” from categories list.
3. Expand “**Application**” --> “**GVL**”
4. Select “**PQI_Inputs0and1**” from the list shown in window. (The port that SMS is linked to must be chosen by the user, Example: PQI of Port 0 and Port 1 will be available in one word).
Refer – [Chapter 3.2 Step-5](#) or [Technical Appendix 9.1](#)
5. Click “**OK**” to confirm the selection.

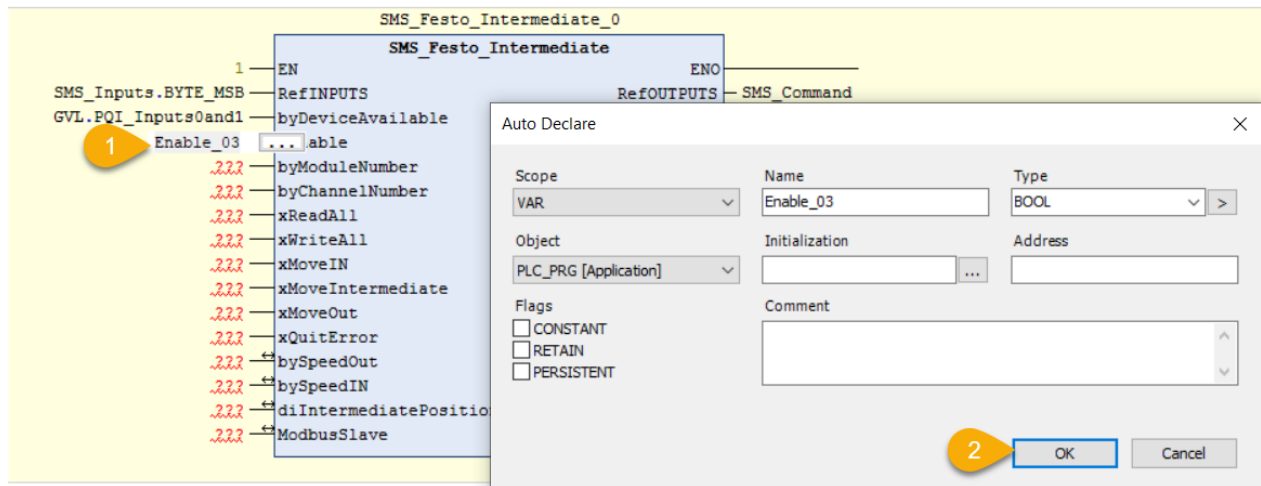
Step – 6

Process Data Out



1. Click the “**Browse**” button of the FB Input Tag.
2. Select “**Variables**” from categories list.
3. Select “**SMS_Command**” from list.
4. Click “**OK**” to confirm the selection.

Step – 7



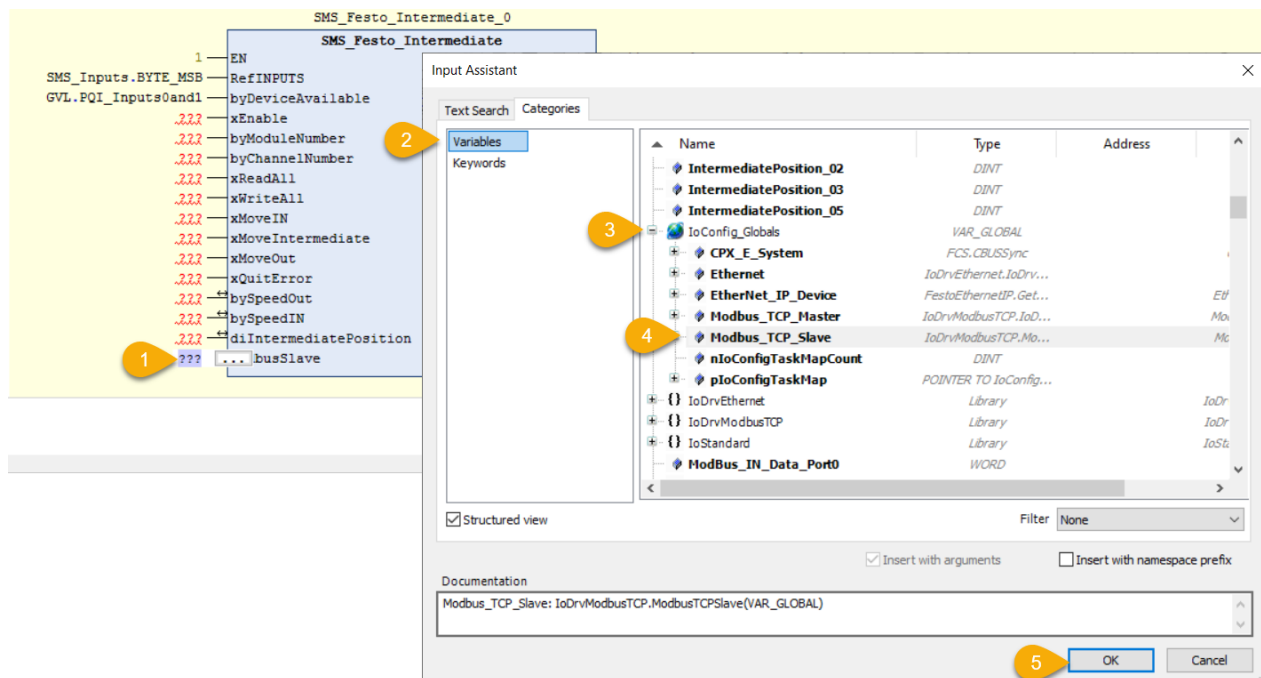
1. Enter the tag name for “**xEnable**” input tag & press enter to open Auto Declare window.
2. Click “**OK**” to confirm the entered tag name.



Note

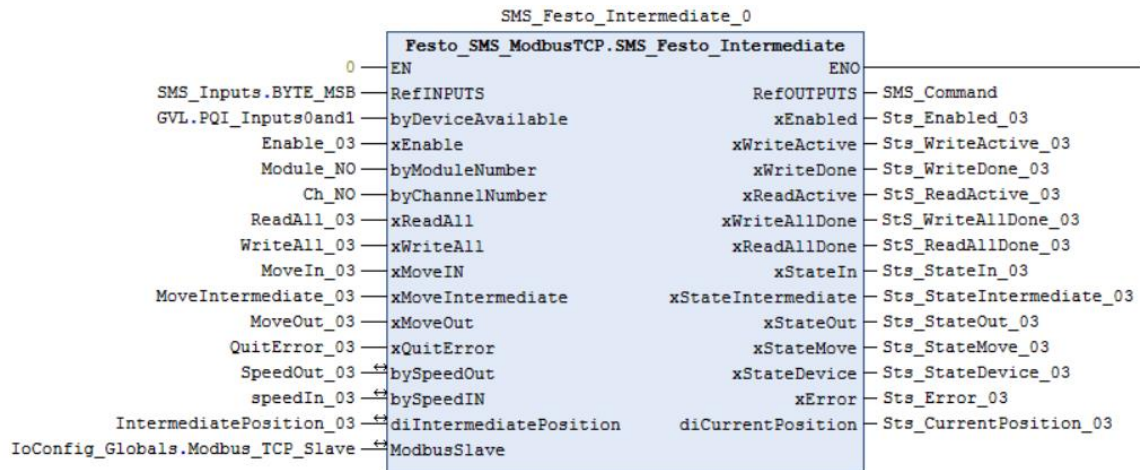
- Repeat Step – 7 for different input & output tags in function block.

Step – 8



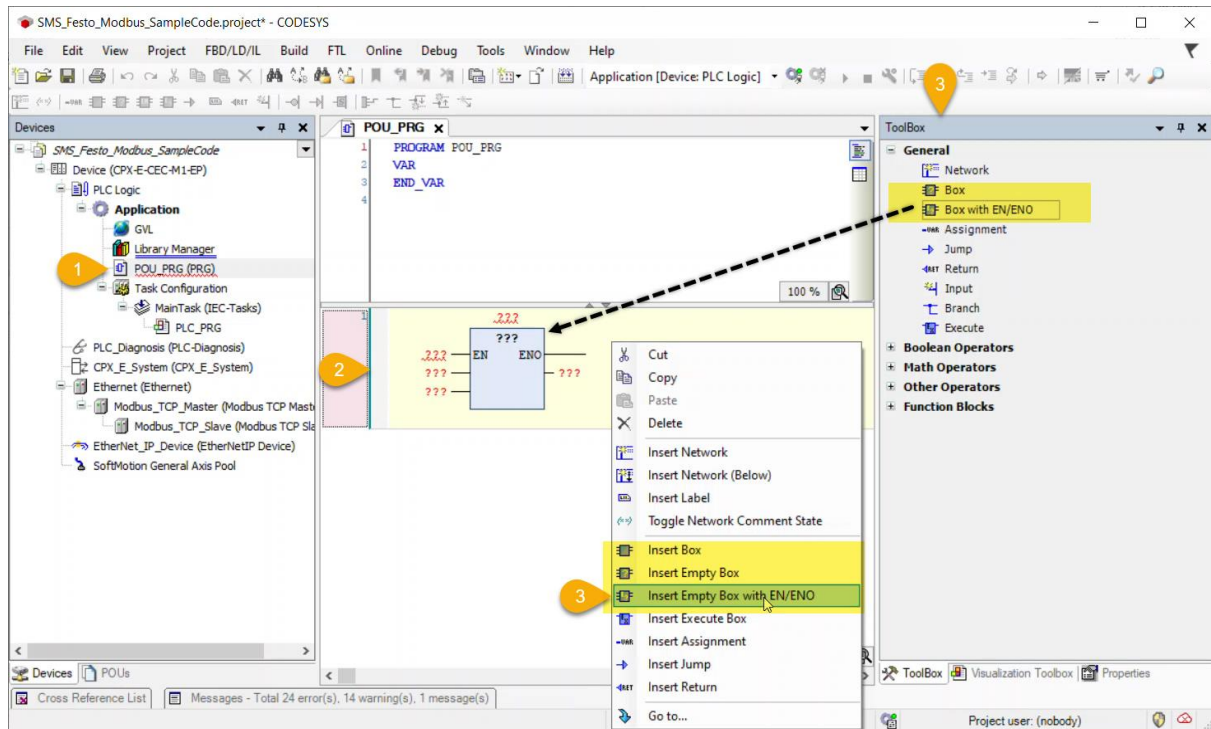
1. Click the “**Browse**” button of the FB Input Tag.
2. Select “**Variables**” from categories list.
3. Expand “**IoConfig_Globals**” from the list shown in window.
4. Select “**Modbus_TCP_Slave**” from IoConfig_Globals list.
5. Click “**OK**” to confirm the selection.

Once all steps completed successfully the **SMS_Festo_Intermediate** Function block will be look like below.



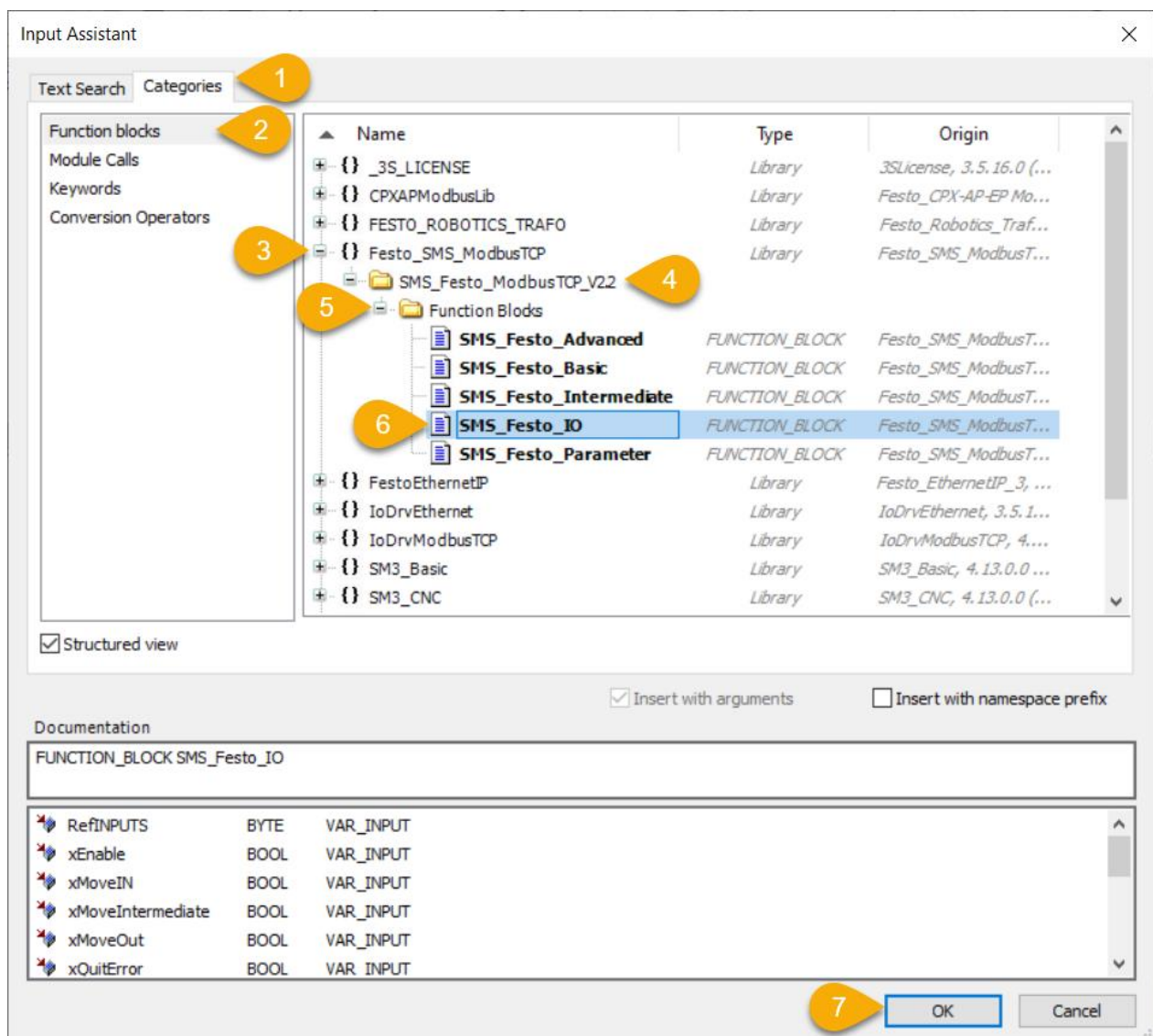
5.4 Configuring the SMS_Festo_IO Function Block Input & Output Interface Tags

Step – 1



1. Double click on **POU_PRG** from Application.
2. Right click on the **Network** from Main window.
3. Click on **Insert Empty Box** to open the Input assistant window.
or
Drag & Drop the “Box with EN/EO” block from toolbox to program area.

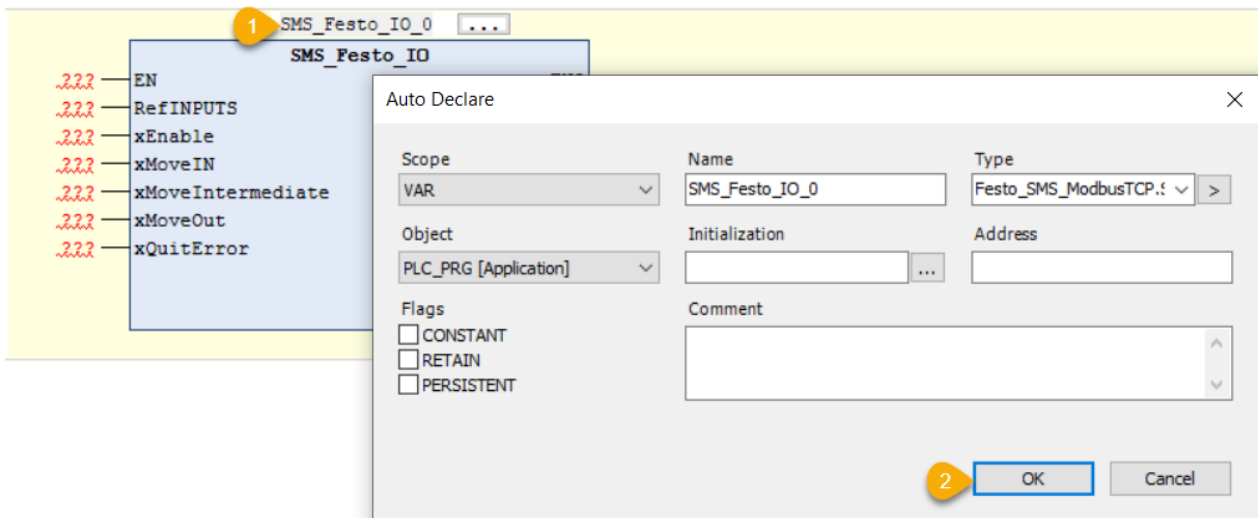
The opened Input Assistant window will be displayed as show below step-2.

Step – 2

1. Click on **Categories** tab from Input Assistant window.
2. Click on **Function blocks** from Categories tab.
3. Expand the **Festo_SMS_ModbusTCP** library and expand the **SMS_Festo_ModbusTCP_V2.2** folder
4. Expand the **Function Blocks** folder to view the Function blocks.
5. Select **SMS_Festo_IO** block
6. Click on **OK** button to call the function block.

The called function block shown in below Step-3

Step – 3



1. Enter the name of the FB instance & press enter key for Auto declare the tag name.
2. Click **“OK”** button from the auto declare window.

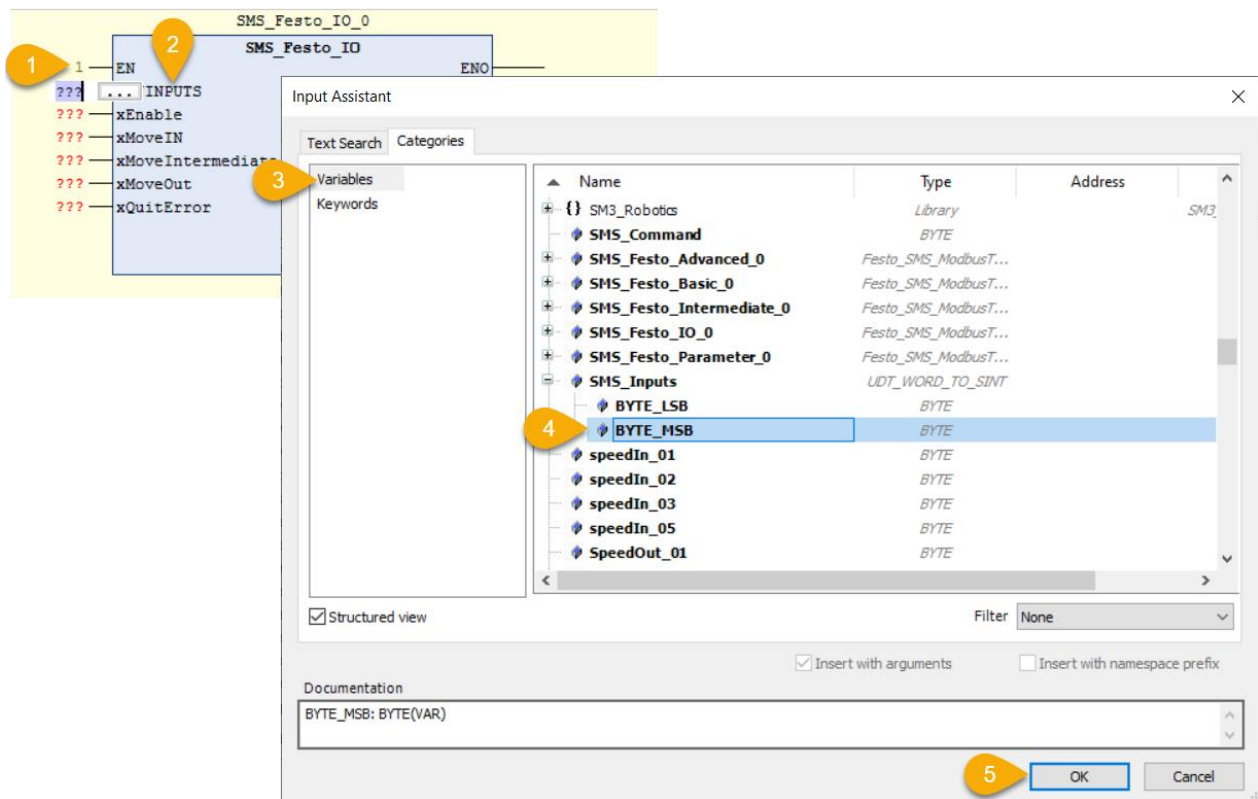
Step – 4



Note

- Function Block require only 1st byte of Process Data IN & Process Data Out.
- Because Modbus reads Process Data as WORD, it must be divided to BYTE and assigned to FB.

Process Data IN

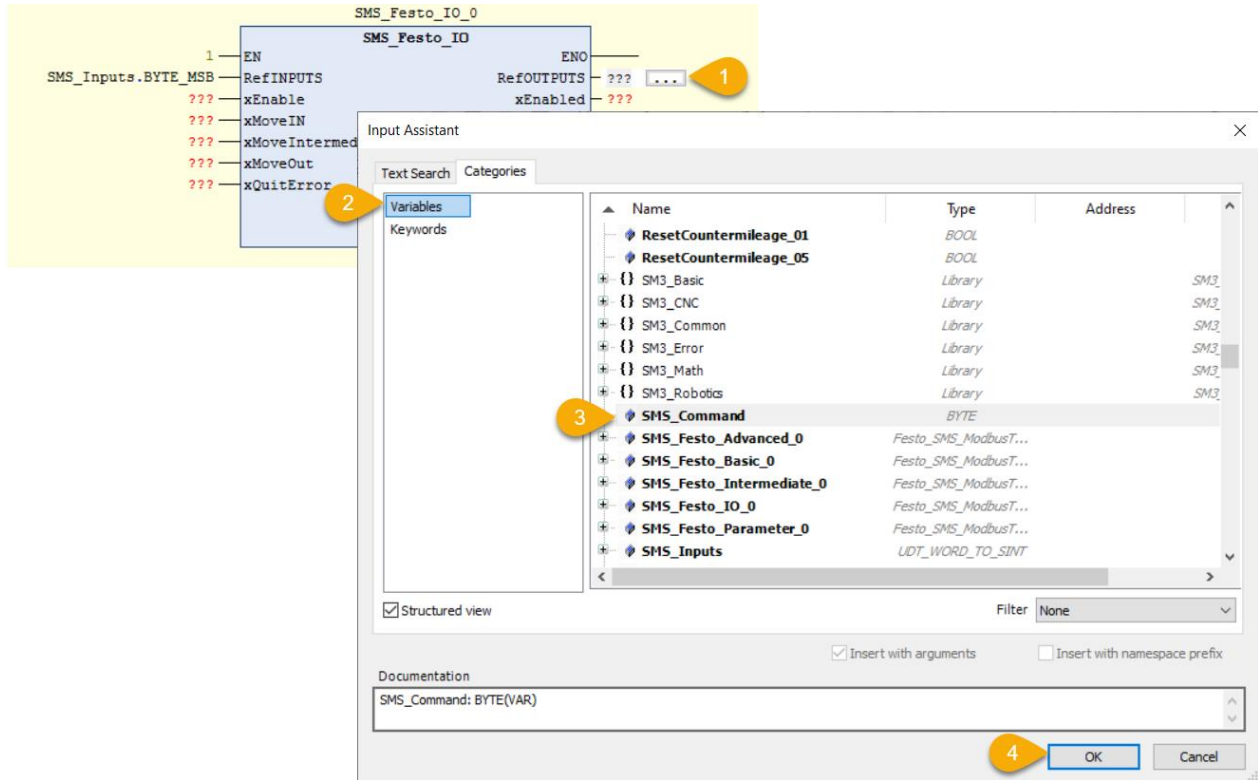


1. Always enable the function block by assigning value 1 in **“EN”** input.
2. Click the **“Browse”** button of the FB Input Tag.
3. Select **“Variables”** from categories list.

4. Expand “**SMS_Inputs**” from the list shown in window.
Select “**SMS_Inputs.BYTE_MSB**”. (Refer the above Note)
5. Click “**OK**” to confirm the selection.

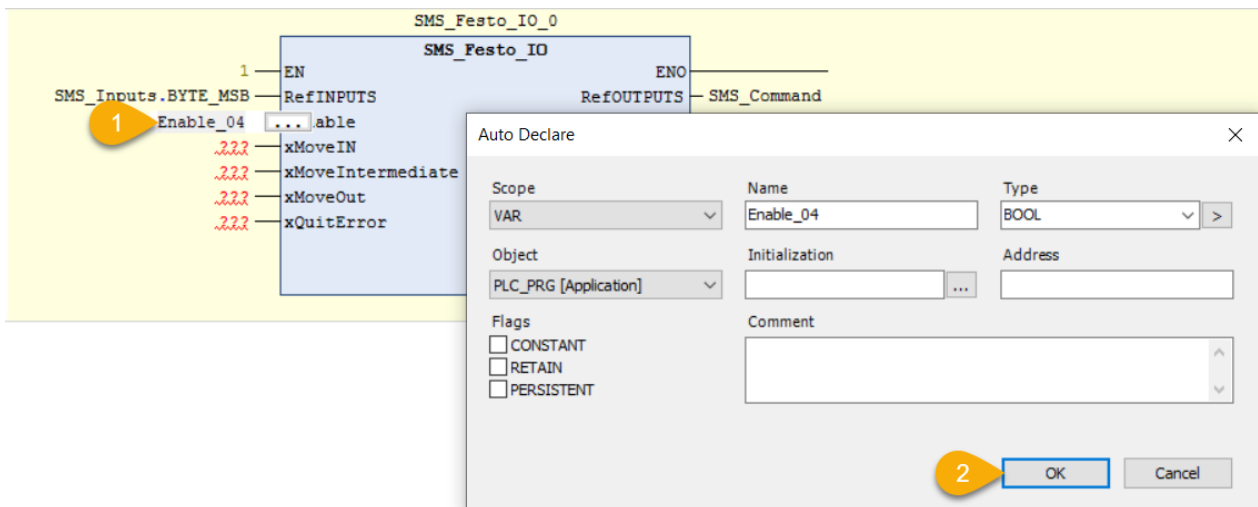
Step – 5

Process Data Out



1. Click the “**Browse**” button of the FB Input Tag.
2. Select “**Variables**” from categories list.
3. Select “**SMS_Command**” from list.
4. Click “**OK**” to confirm the selection.

Step – 6



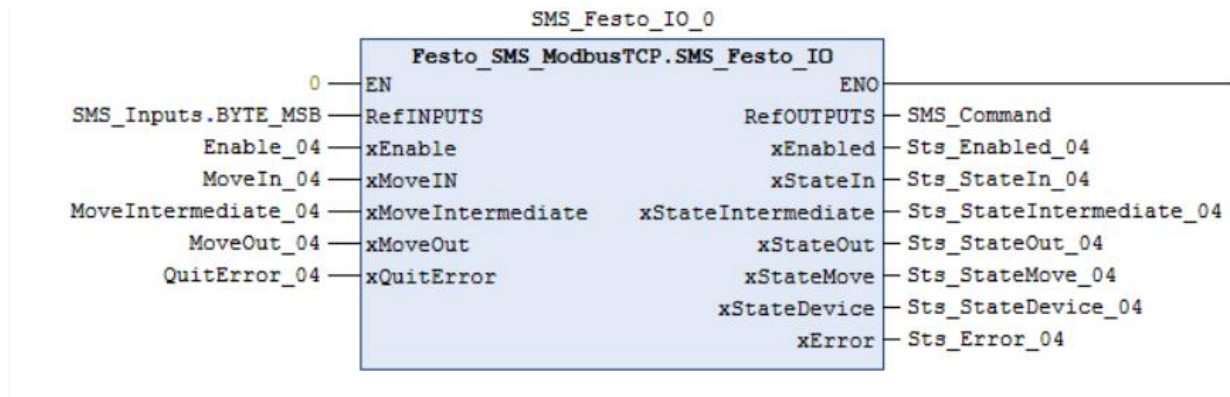
1. Enter the tag name for “**xEnable**” input tag & press enter to open Auto Declare window.
2. Click “**OK**” to confirm the entered tag name.



Note

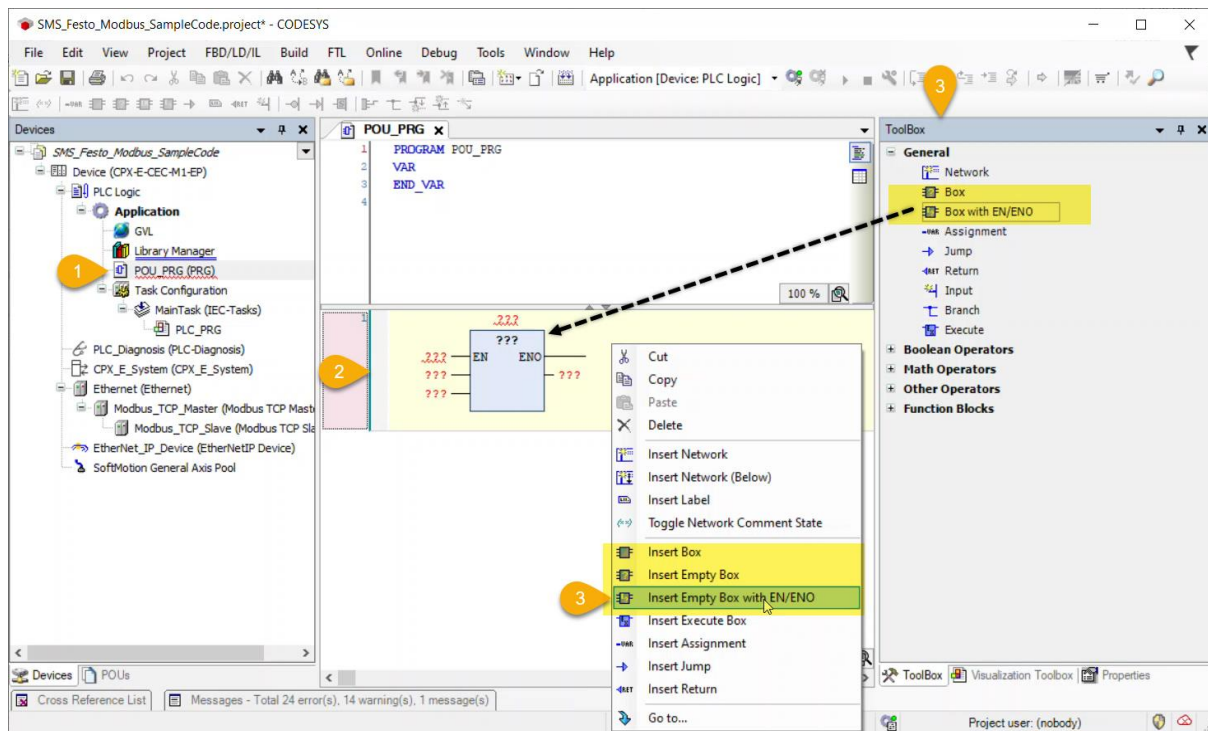
- Repeat Step – 6 for different input & output tags in function block.

Once all steps completed successfully the **SMS_Festo_IO** Function block will be look like below.



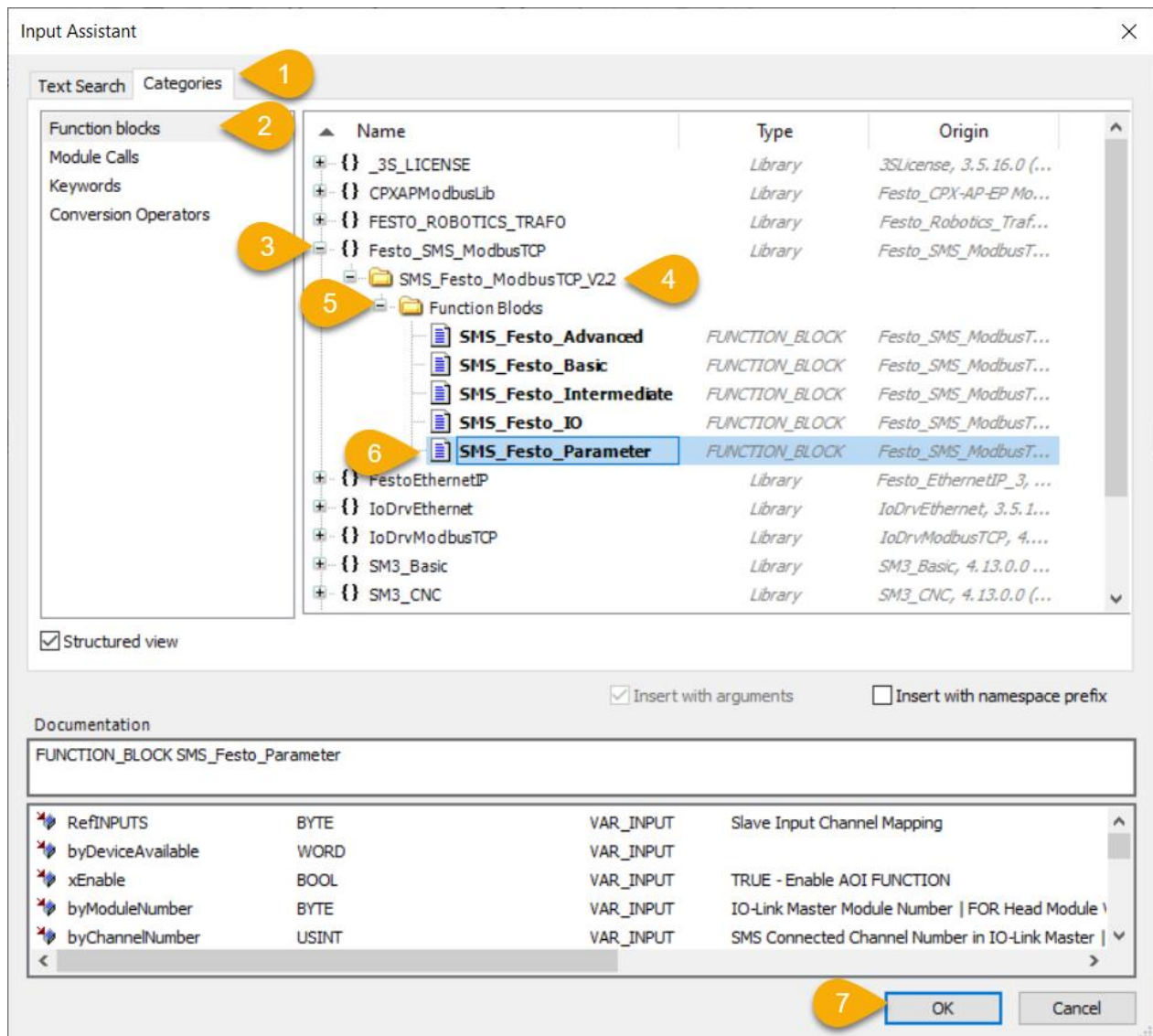
5.5 Configuring the SMS_Festo_Parameter Function Block Input & Output Interface Tags

Step – 1



1. Double click on **POU_PRG** from Application.
2. Right click on the **Network** from Main window.
3. Click on **Insert Empty Box** to open the Input assistant window.
or
Drag & Drop the “Box with EN/EO” block from toolbox to program area.

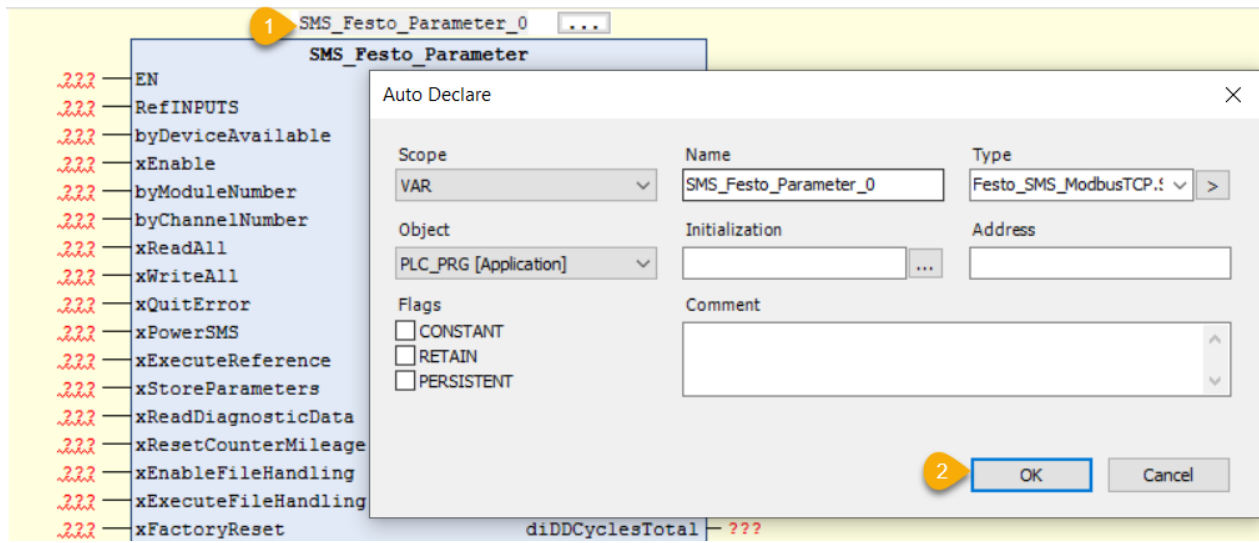
The opened Input Assistant window will be displayed as show below step-2.

Step – 2

1. Click on **Categories** tab from Input Assistant window.
2. Click on **Function blocks** from Categories tab.
3. Expand the **Festo_SMS_ModbusTCP** library and expand the **SMS_Festo_ModbusTCP_V2.2** folder
4. Expand the **Function Blocks** folder to view the Function blocks.
5. Select **SMS_Festo_Paarameter** block
6. Click on **OK** button to call the function block.

The called function block shown in below Step-3

Step – 3



1. Enter the name of the FB instance & press enter key for Auto declare the tag name.
2. Click **“OK”** button from the auto declare window.

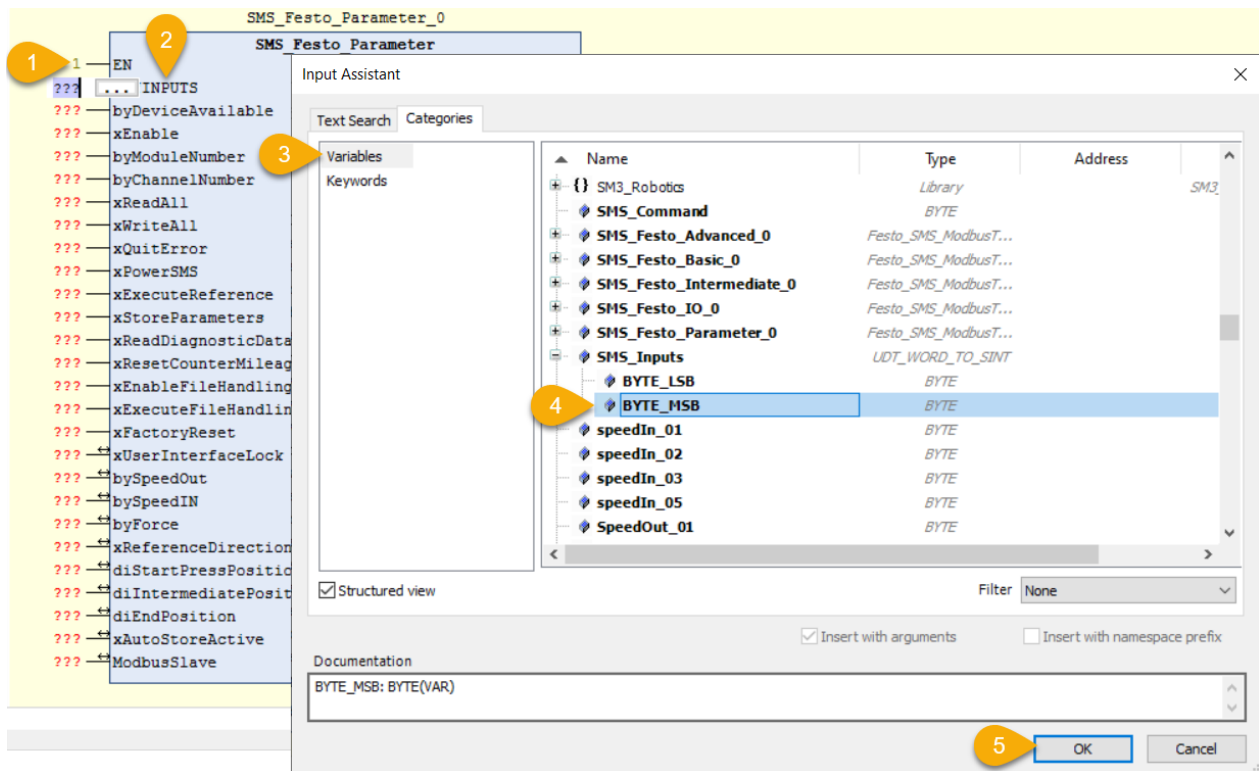
Step – 4



Note

- Function Block require only 1st byte of Process Data IN & Process Data Out.
- Because Modbus reads Process Data as WORD, it must be divided to BYTE and assigned to FB.

Process Data IN

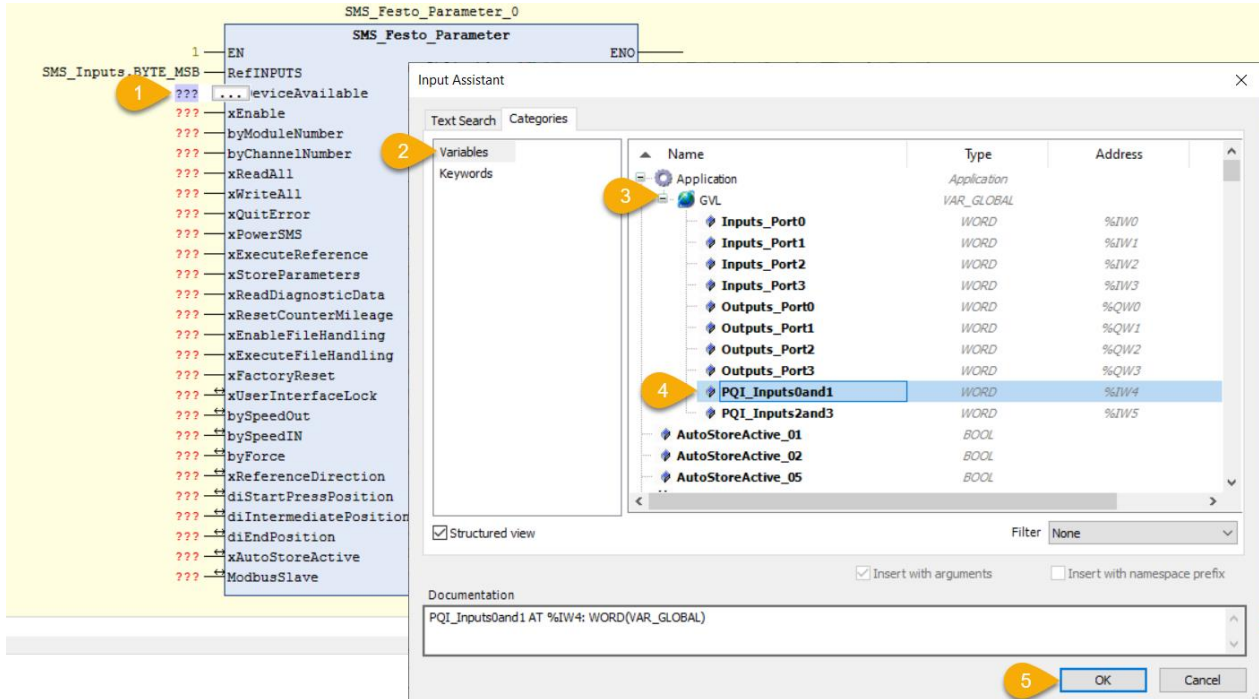


1. Always enable the function block by assigning value 1 in **“EN”** input.
2. Click the **“Browse”** button of the FB Input Tag.
3. Select **“Variables”** from categories list.

4. Expand “**SMS_Inputs**” from the list shown in window.
Select “**SMS_Inputs.BYTE_MSB**”. (Refer the above Note)
5. Click “**OK**” to confirm the selection.

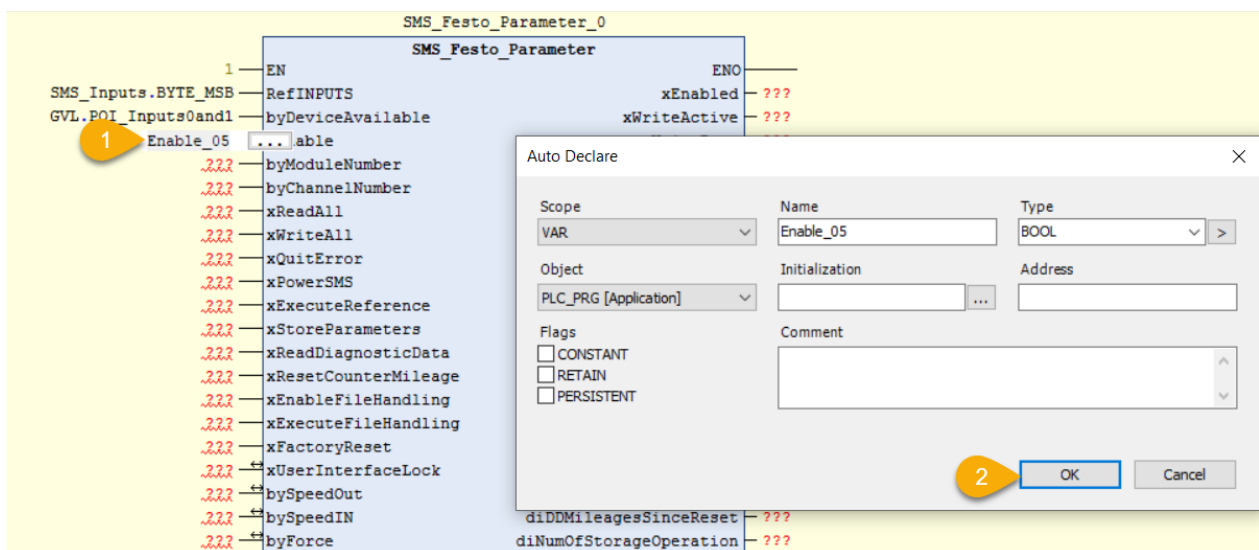
Step – 5

PQI – Process Data IN



1. Click the “**Browse**” button of the FB Input Tag.
2. Select “**Variables**” from categories list.
3. Expand “**Application**” --> “**GVL**”
4. Select “**PQI_Inputs0and1**” from the list shown in window. (The port that SMS is linked to must be chosen by the user, Example: PQI of Port 0 and Port 1 will be available in one word).
Refer – [Chapter 3.2 Step-5](#) or [Technical Appendix 9.1](#)
5. Click “**OK**” to confirm the selection.

Step – 6



1. Enter the tag name for “**xEnable**” input tag & press enter to open Auto Declare window.

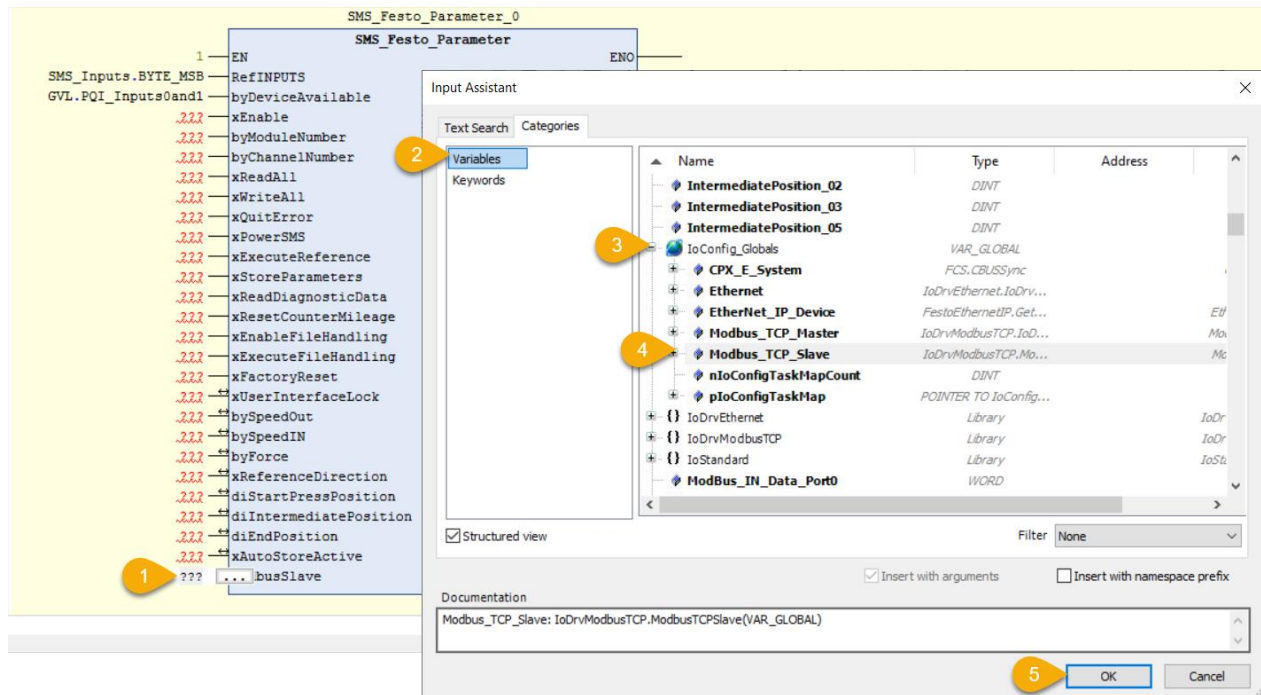
- Click **“OK”** to confirm the entered tag name.



Note

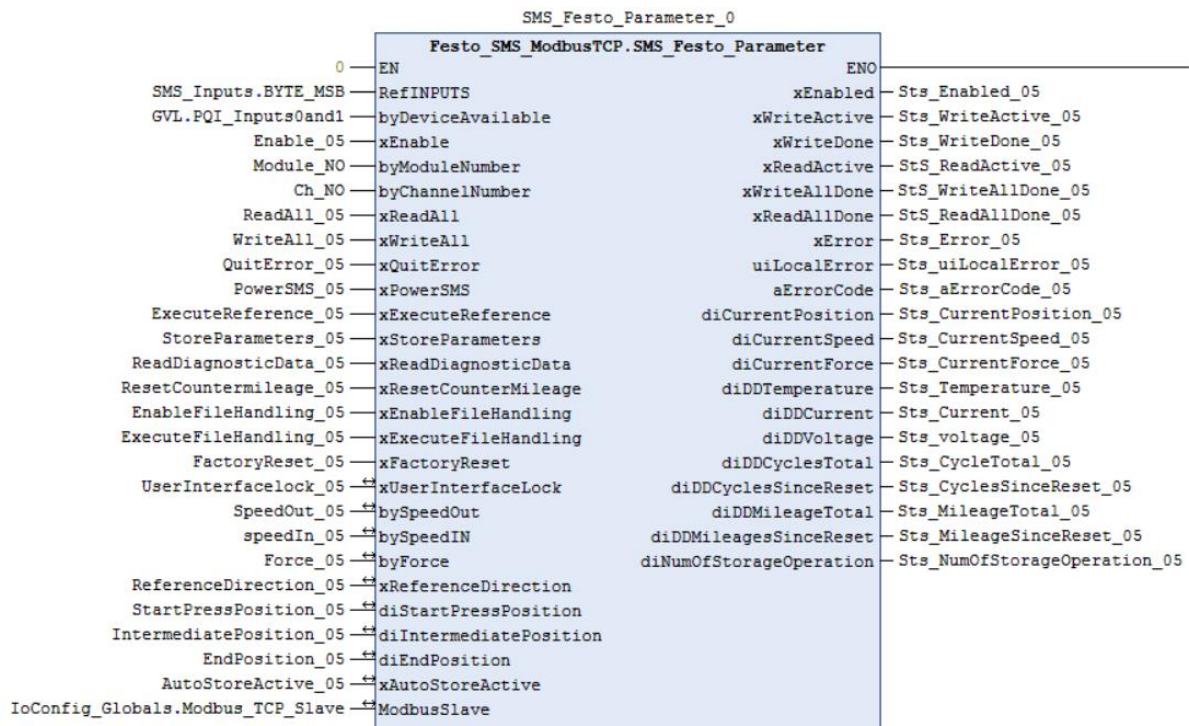
- Repeat Step – 6 for different input & output tags in function block.

Step – 7



- Click the **“Browse”** button of the FB Input Tag.
- Select **“Variables”** from categories list.
- Expand **“IoConfig_Globals”** from the list shown in window.
- Select **“Modbus_TCP_Slave”** from IoConfig_Globals list.
- Click **“OK”** to confirm the selection.

Once all steps completed successfully the **SMS_Festo_Parameter** Function block will be look like below.



Note

- SMS_Festo_Parameter block is used to control the acyclic parameter functions. User can not able to control the process data parameter functions in this block . If user wants to control the process data functions with this block, they can use the SMS_Festo_IO block along with SMS_Festo_Parameter block.



Note

- User has to call one function block at a time in program, except SMS_Festo_IO and SMS_Festo_Parameter block can call simultaneously.
- User has to call the function block depends on the application.
- If the user application need only I/O related operation, then user can call the **SMS_Festo_IO** block.
- If the user application need only I/O related operation and Intermediate function parameter, then user can call the **SMS_Festo_Intermediate** block.
- If the user application needs only I/O and basic parameter function, then user can call the **SMS_Festo_Basic** block.
- If the user application need I/O and All parameter function, then user can call the **SMS_Festo_Advanced** block.
- If the user application need only acyclic parameter function, then user can call the **SMS_Festo_Parameter** block.

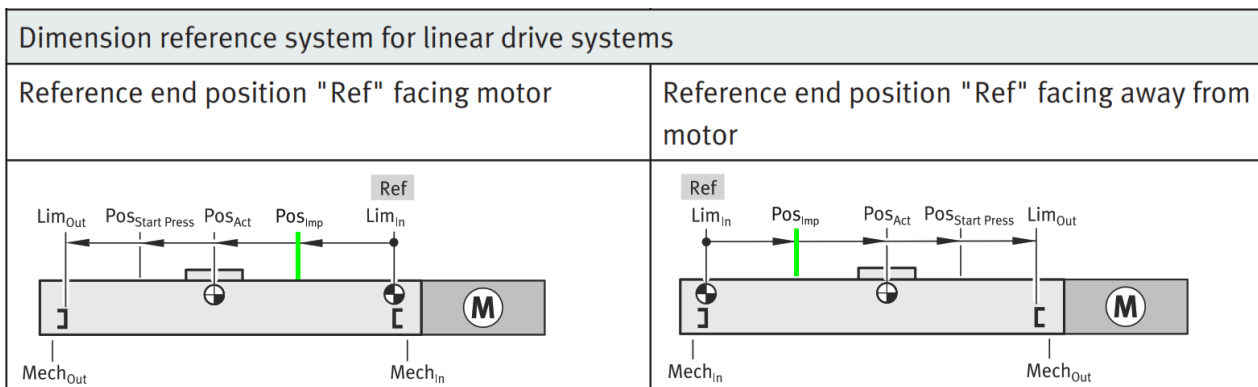
6 Important Operating Details of SMS

6.1 Control Tag Operation details

S.NO	IN_MoveIN	In_MoveIntermediate	In_MoveOut	Output
1.	0	0	0	0
2.	0	0	1	1
3.	0	1	0	1
4.	0	1	1	0
5.	1	0	0	1
6.	1	0	1	0
7.	1	1	0	0
8.	1	1	1	0

Refer the above given truth table for operating the Control tag's to generate the desired Output.

6.2 Position Value Operation



1. User Providing Input value of **diEndPosition** should be always higher than the **diStartPressPosition** and **diIntermediatePosition** value or else Device will lead the operation to Error.

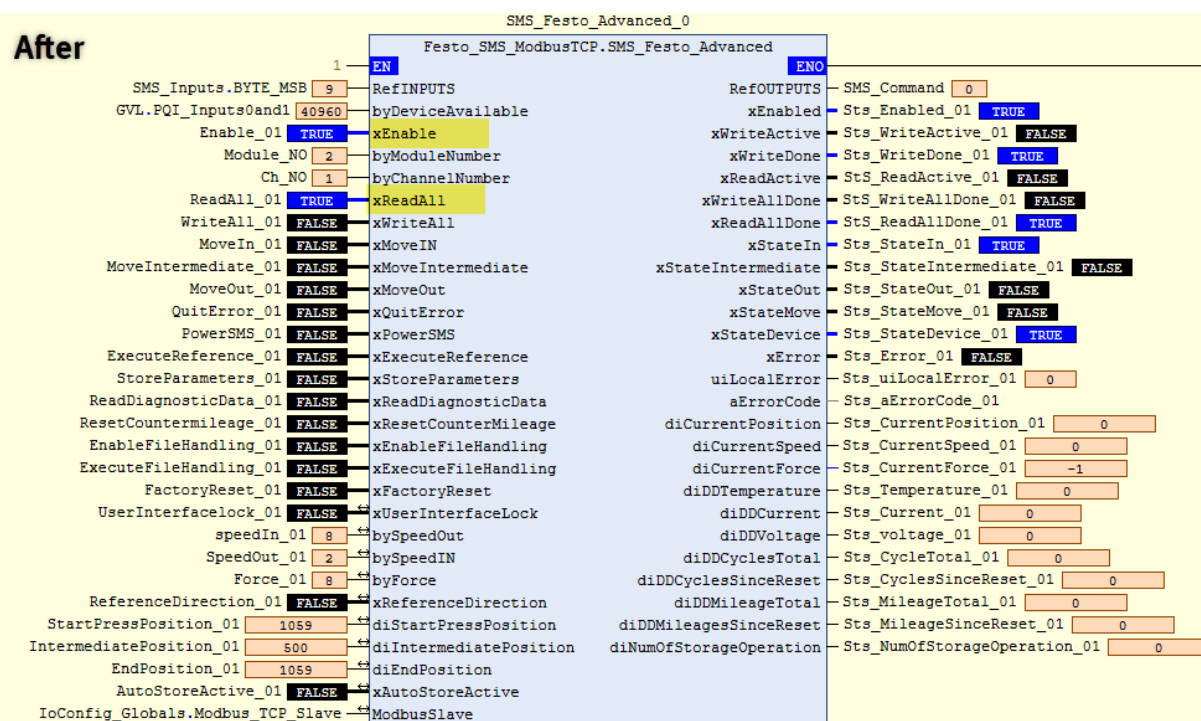
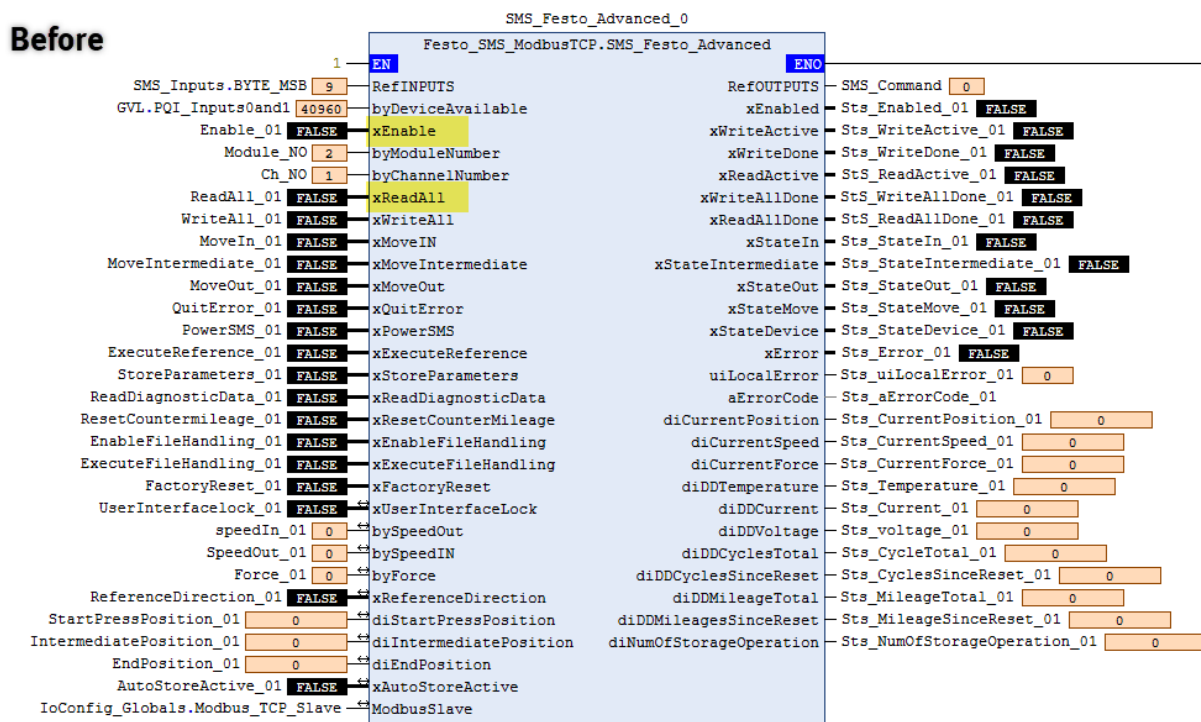


Warning

2. If User Provided Input value of **diStartPressPosition** is higher than the **diEndPosition**, then the SMS Firmware Write the **diStartPressPosition** value automatically to the **diEndPosition** Value.

7 Sample Code

7.1 Initial Setup



1. Enter the actuator unit connected module number in “**bymoduleNumber**” input tag. Head module number = 1.
2. Enter the actuator unit connected IO-Link master Port number in “**byChannelNumber**” input tag.
3. Toggle the “**xEnable**” control tag from false to true. If function block enabled “**xEnabled**” tag will update the status by turning into true.

4. Toggle the **“xReadAll”** control tag from false to true. To Read the Present Data from the actuator unit to Synchronize the function block with the actuator unit.
5. Wait until **“xReadAllDone”** tag will update the status by turning into true. Once the **“xReadAllDone”** tag is true, All the Control Tag data is updated with the Present Data of actuator unit.
6. Toggle the **“xReadAll”** control tag from true to false.
7. Enter the user preferred values for the control tags **bySpeedOut**, **bySpeedIN**, **byForce**, **diStartPress-Position**, **diIntermediatePosition** and **diEndPosition** values.
8. Once All entered parameters values writing are successful then the monitor tag **“xWriteDone”** will be in true state.

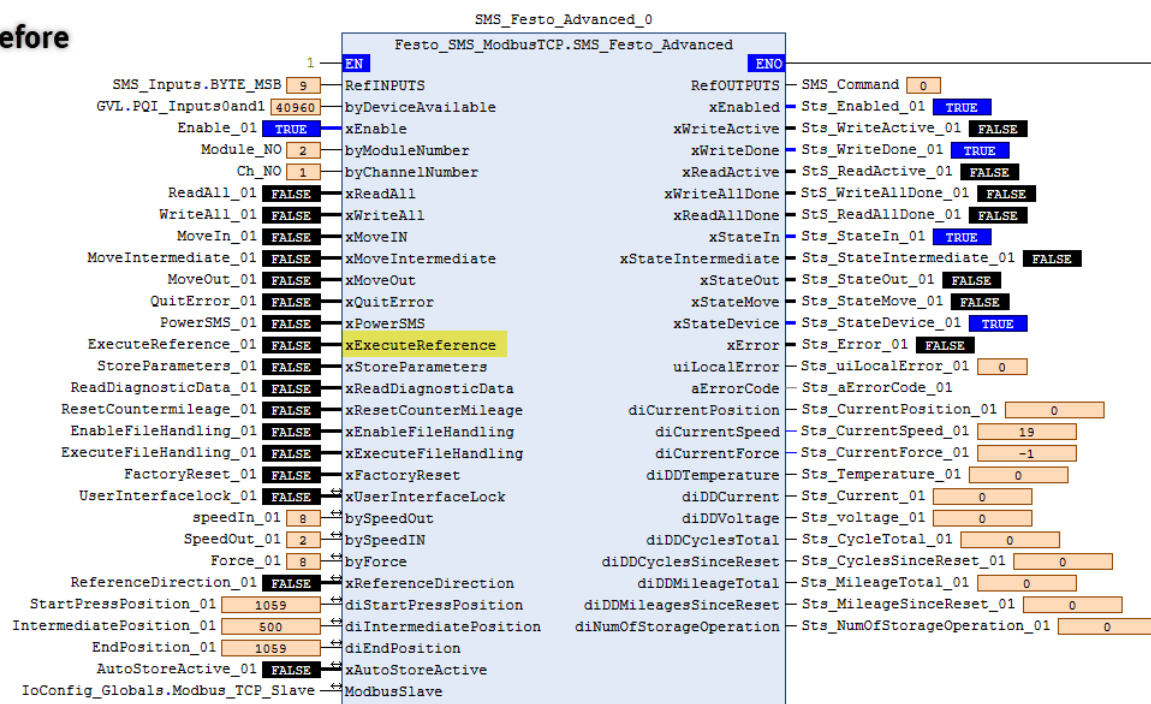


Note

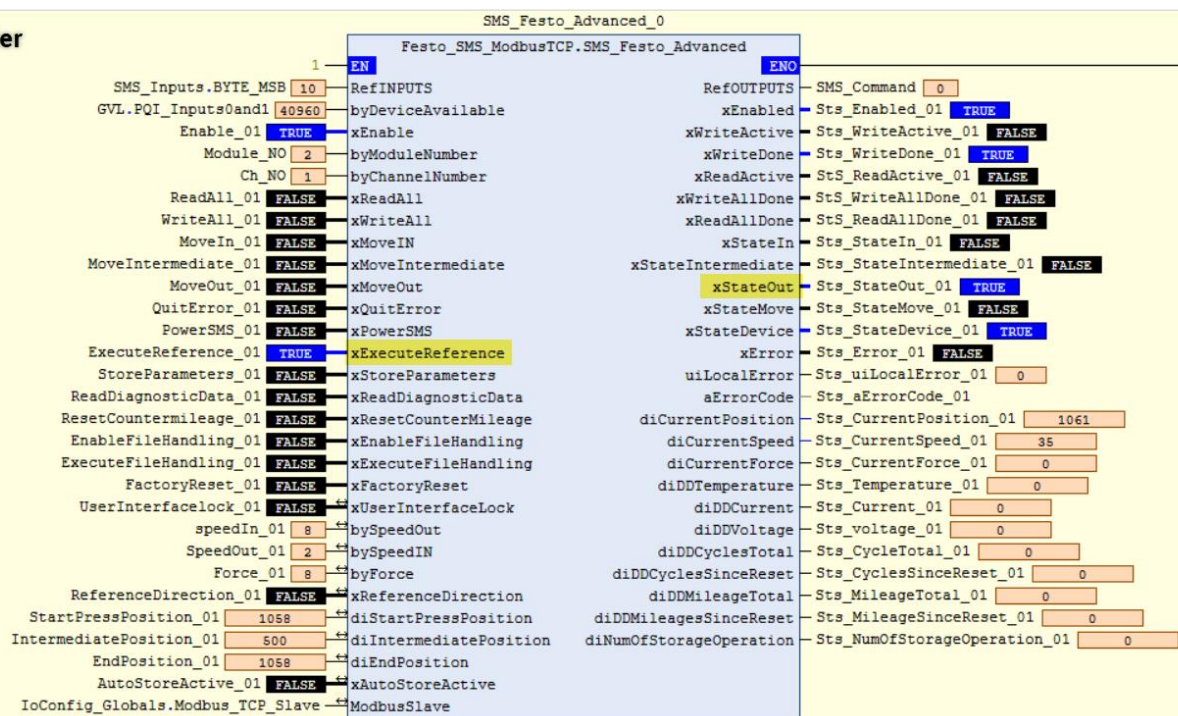
- After enabling the function block user have to give a reset command to clear the local error 1001. Reset command can give through input tag **“xQuitError”** by toggling false to true.

7.2 Actuator unit Homing with stroke detection

Before



After



1. Toggle the **"xExecuteReference"** control tag from false to true.
2. At this stage actuator unit starts reference movement move towards selected reference end position **"Pos_{Ref}"** defined in **"xReferenceDirection"**, automatically followed by movement towards end position **"Lim_{Out}"**.
3. So initially status tag **"xStateMove"** will be ON and then **"xStateOut"**.
4. Wait Until **"xReadActive"** tag Status turn to True.
5. Once the Homing is done **diStartPressPosition**, **diIntermediatePosition** and **diEndPosition** values will change automatically to the maximum limit of Position out values which was shown in the above After image.
6. To view the Position values after homing Toggle **"xReadAll"** control tag from false to true. Wait Until **"xReadAllDone"** tag status turn to True. Value will be updated accordingly with the actuator unit.

Remark:

Actuator unit Homing with stroke detection is just needed once during commissioning for teaching the maximum stroke length of the application, or after changing the stroke length e.g. by external mechanical end stops.

This method is not needed to be executed after every power off.

Homing to existing reference end position after power off is done automatically with the first motion command sent to the device.

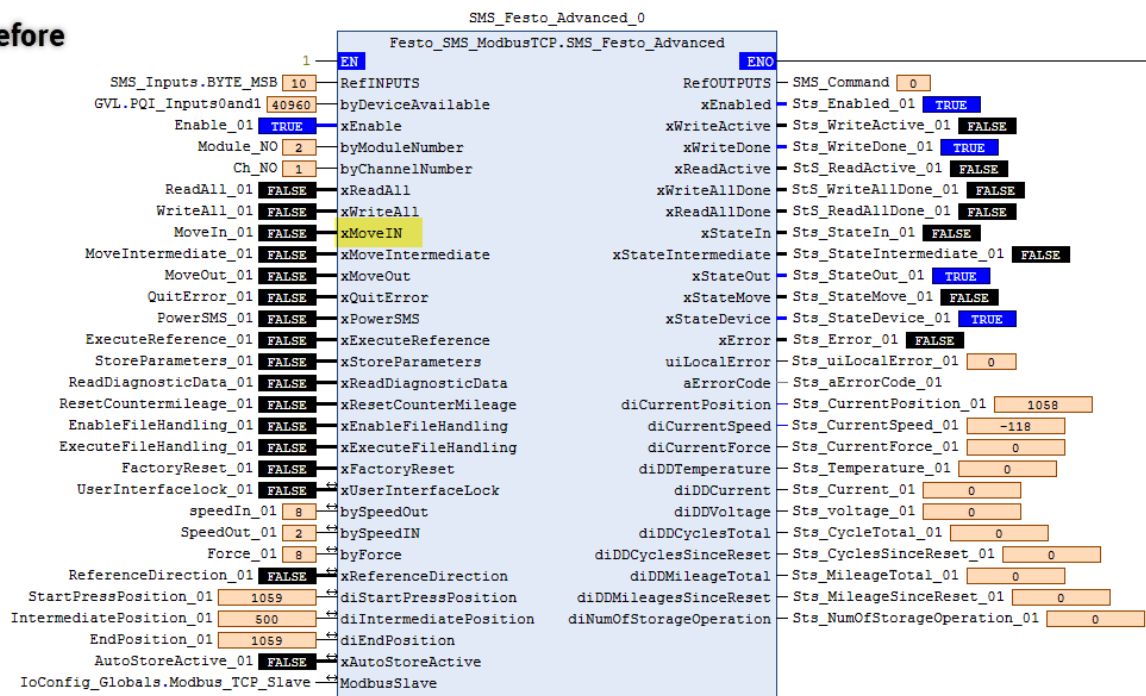


Note

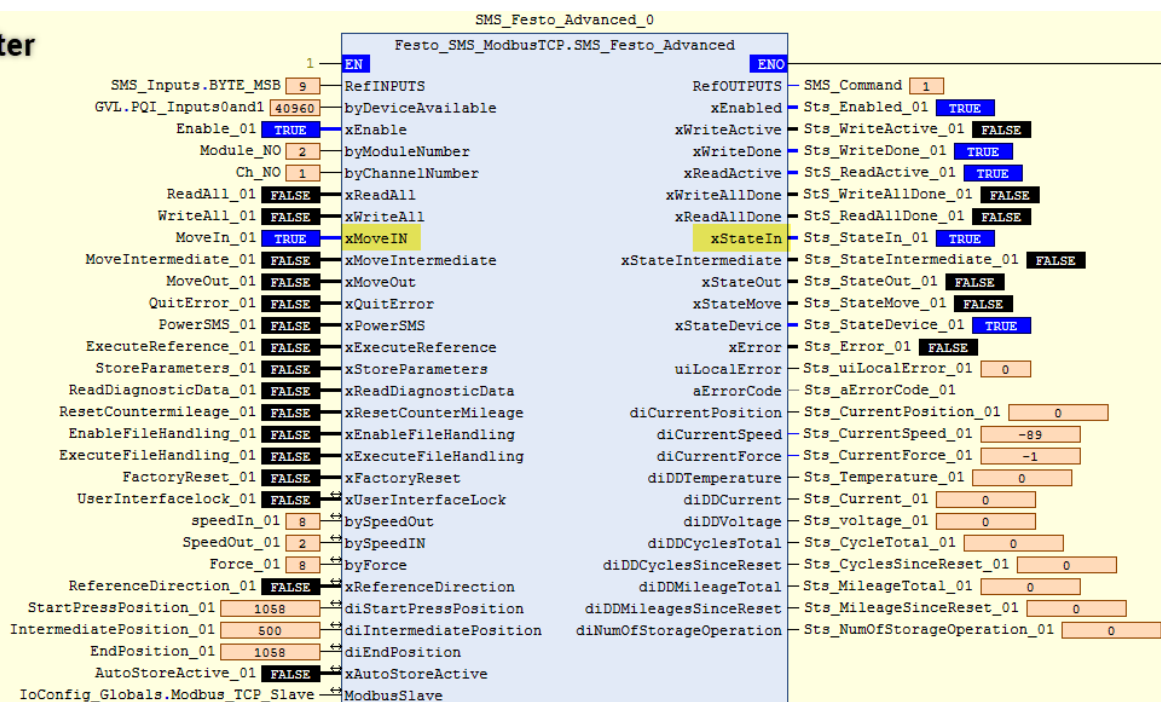
- If user want to change the homing direction then user can change direction by toggling the input tag **xReferenceDirection** from False to True before the reference command.

7.3 Actuator unit Move to Inwards

Before



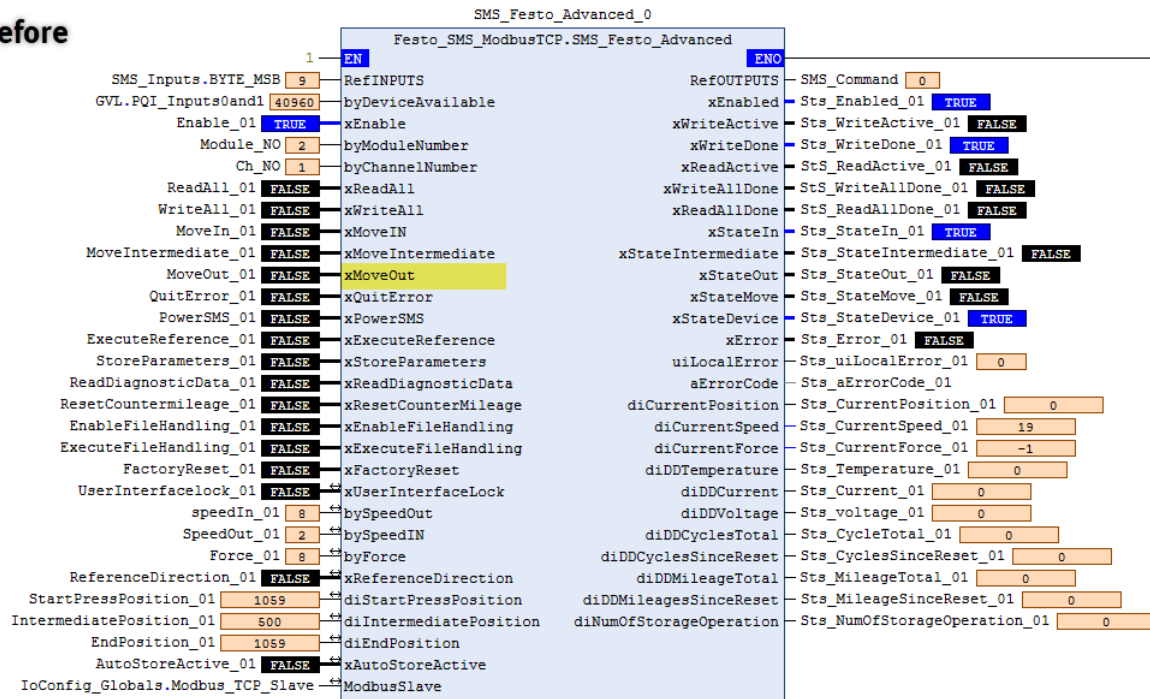
After



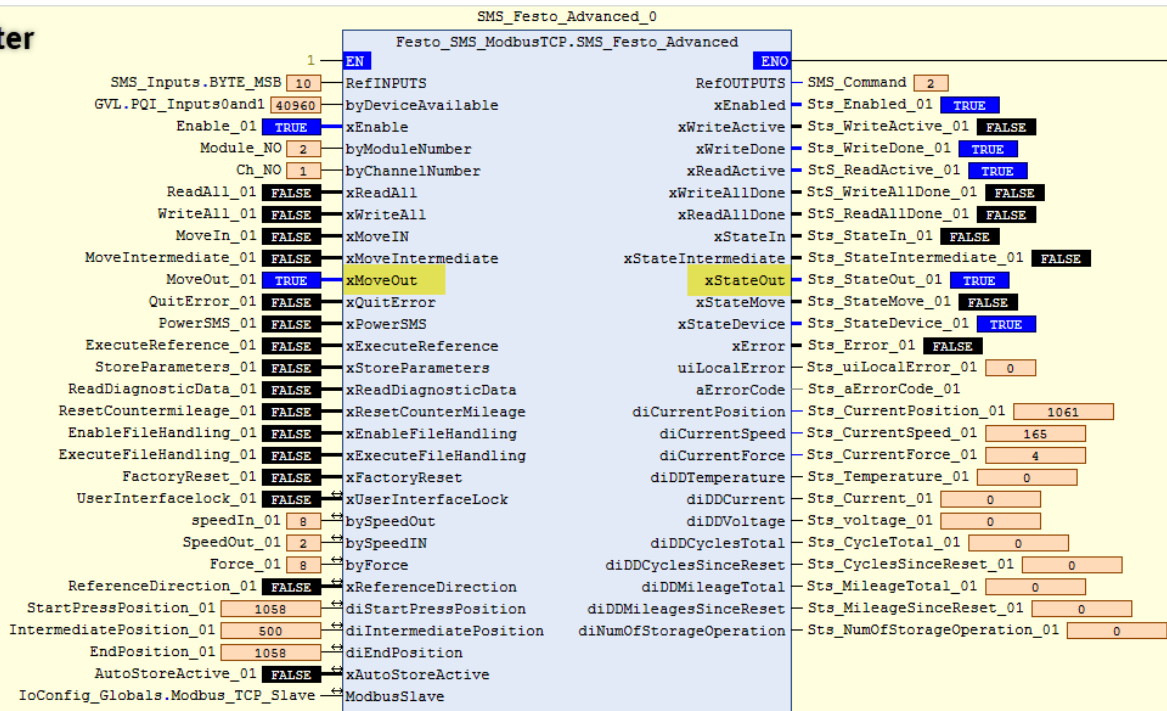
- Enter the required Position value in input tag **diStartPressPosition**, **diIntermediatePosition** and **diEndPosition**.
- Toggle control tag **xMoveIN** from false to true to move the actuator unit towards IN direction. Once actuator unit reached IN position status tag **xStatusIn** will be true and tag **diCurrentPosition** will be show the IN position.

7.4 Actuator unit Move to Outwards

Before



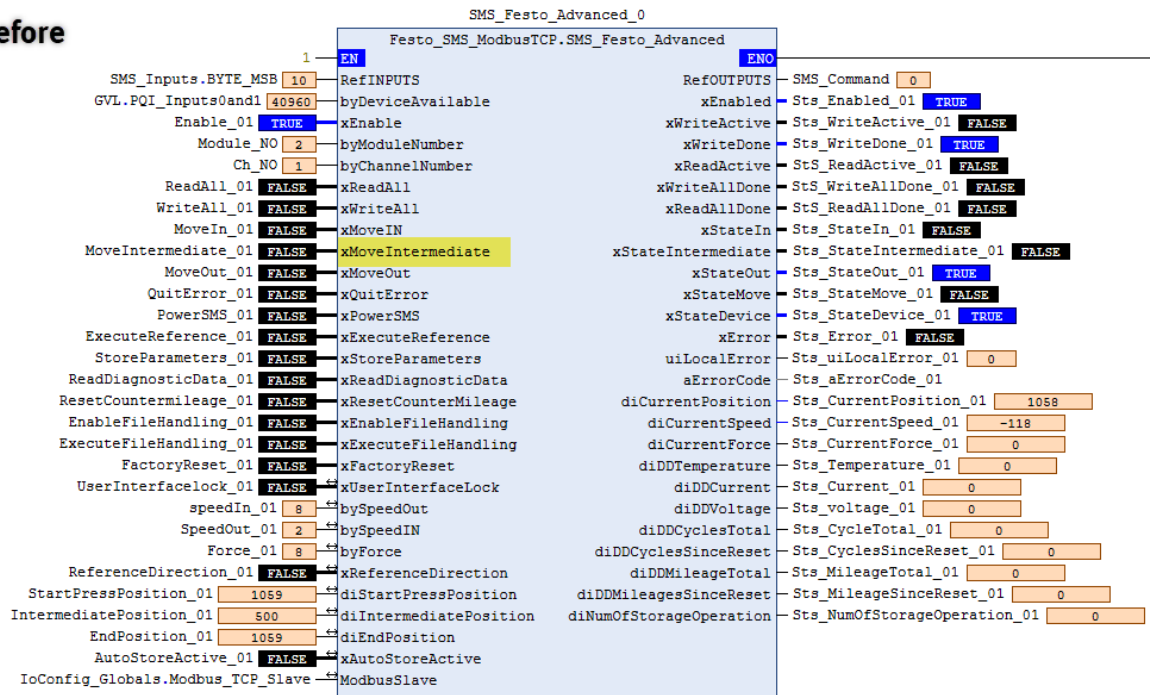
After



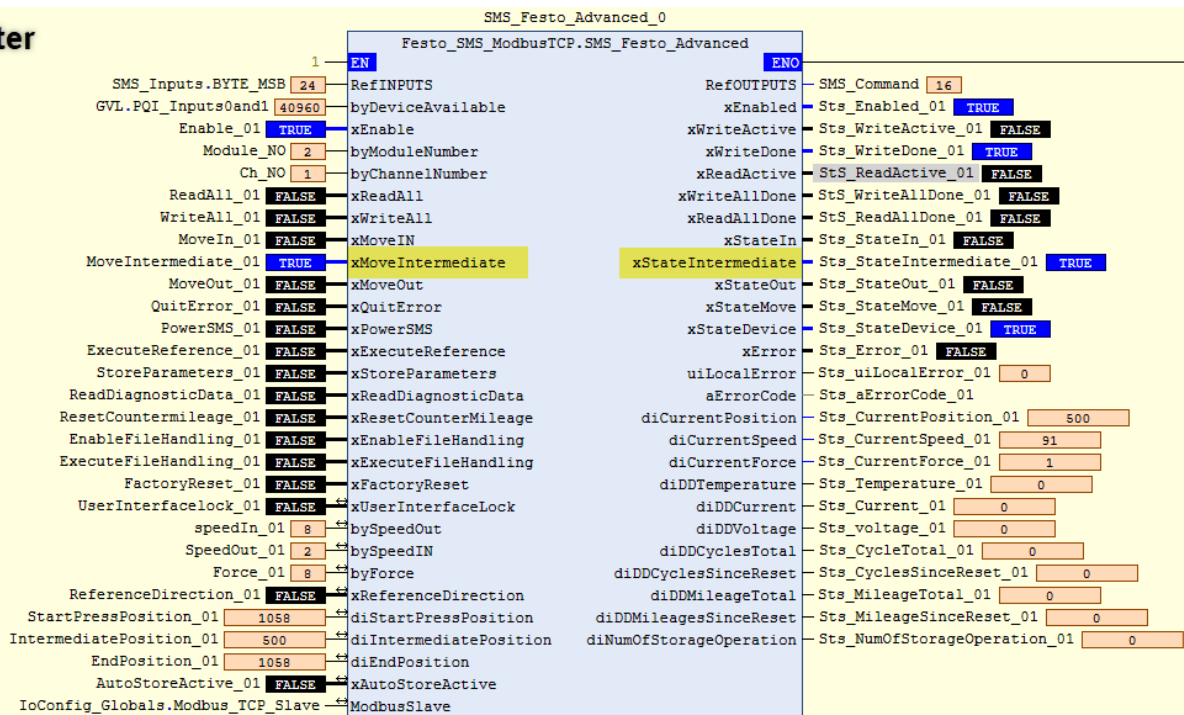
- Toggle control tag **xMoveOut** from false to true to move the actuator unit towards Out direction. Once actuator unit reached Out position status tag **xStateOut** will be true and tag **diCurrentPosition** will be show the Out position.

7.5 Actuator unit Move to Intermediate

Before



After



- Toggle control tag **xMoveIntermediate** from false to true to move the actuator unit towards intermediate position. Once actuator unit reached intermediate position status tag **xStateIntermediate** will be true and tag **diCurrentPosition** will be show the Intermediate position.

7.6 Sample Code Description for Move The Axis to Multiple Intermediate positions

Block Enable

1. Setup Inputs
 - xEnable := FALSE;
2. Feedback Outputs
 - xEnabled is FALSE;
 - xError is FALSE;
3. Sequence
 - Set xEnable := TRUE;
 - Wait until xEnabled is TRUE

Move out Position

1. Setup Inputs
 - xMoveIN := FALSE;
 - xMoveIntermediate := FALSE;
 - xMoveOut := FALSE;
2. Feedback Outputs
 - xEnabled is TRUE;
 - xError is FALSE;
 - XStateout is FALSE;
 - XStateMove is FALSE;
3. Sequence
 - Set xMoveOut := TRUE;
 - Wait until xStateOut is TRUE AND xStateMove is FALSE;
 - Set xMoveOut := FALSE;

Move First Intermediate Position

1. Setup Inputs
 - xMoveIN := FALSE;
 - xMoveIntermediate := FALSE;
 - xMoveOut := FALSE;
2. Feedback Outputs
 - xEnabled is TRUE;
 - xError is FALSE;
 - XState Intermediate is FALSE;
 - XStateMove is FALSE;
3. Sequence
 - Set diIntermediatePosition := 750;
 - Wait until xWriteActive is TRUE;
 - Wait until xWriteDone is TRUE;
 - Set xMoveIntermediate := TRUE;
 - Wait until xStateMove is TRUE;
 - Wait until xStateIntermediate is TRUE AND xStateMove is FALSE;
 - Set xMoveIntermediate := FALSE;

Move Second Intermediate Position

1. Setup Inputs
 - xMoveIN := FALSE;
 - xMoveIntermediate := FALSE;
 - xMoveOut := FALSE;
2. Feedback Outputs
 - xEnabled is TRUE;
 - xError is FALSE;
 - XStateMove is FALSE;
3. Sequence
 - Set diIntermediatePosition := 500;
 - Wait until xWriteActive is TRUE;
 - Wait until xWriteDone is TRUE;
 - Set xMoveIntermediate := TRUE;
 - Wait until xStateMove is TRUE;
 - Wait until xStateIntermediate is TRUE AND xStateMove is FALSE;
 - Set xMoveIntermediate := FALSE;

Move Third Intermediate Position

1. Setup Inputs
 - xMoveIN := FALSE;
 - xMoveIntermediate := FALSE;
 - xMoveOut := FALSE;
2. Feedback Outputs
 - xEnabled is TRUE;
 - xError is FALSE;
 - XStateMove is FALSE;
3. Sequence
 - Set diIntermediatePosition := 250;
 - Wait until xWriteActive is TRUE;
 - Wait until xWriteDone is TRUE;
 - Set xMoveIntermediate := TRUE;
 - Wait until xStateMove is TRUE;
 - Wait until xStateIntermediate is TRUE AND xStateMove is FALSE;
 - Set xMoveIntermediate := FALSE;

Move In Position

1. Setup Inputs
 - xMoveIN := FALSE;
 - xMoveIntermediate := FALSE;
 - xMoveOut := FALSE;
2. Feedback Outputs
 - xEnabled is TRUE;
 - xError is FALSE;
 - XStateIn is FALSE;
 - XStateMove is FALSE;
3. Sequence
 - Set xMoveIn := TRUE;
 - Wait until xStateIn is TRUE AND xStateMove is FALSE;
 - Set xMoveIn := FALSE;

8 Technical appendix

8.1 Find Modbus Address Details for Simplified Motion Series actuator unit Connected Port

As per this application note Simplified Motion Series actuator unit module is connected in port – 0 of the IO-Link master module in slot position – 2. Below is the procedure to find the Modbus registers details.

The screenshot shows the Festo Modbus TCP configuration interface. The 'Modules' table lists two modules: CPX-AP-I-EP-M12 (Slot 1) and CPX-AP-I-4IOL-M12 (Slot 2). The 'Holding Register View' is displayed for Module 2, showing a table of registers. The 'Outputs' section lists registers 0 to 3, and the 'Inputs' section lists registers 5000 to 5005. The register 5004 is highlighted in yellow, indicating it is the input address for Port 0.

Register	Offset (bit)	Bit length	Module	Channel	Datatype	Name
Outputs						
0	0	16	2	0	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 0
1	0	16	2	1	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 1
2	0	16	2	2	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 2
3	0	16	2	3	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 3
Inputs						
5000	0	16	2	0	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 0
5001	0	16	2	1	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 1
5002	0	16	2	2	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 2
5003	0	16	2	3	USINT[2]	Module 2 - CPX-AP-I-4IOL-M12 - Port 3
5004	0	8	2	4	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 4 - PQI
5004	8	8	2	5	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 5 - PQI
5005	0	8	2	6	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 6 - PQI
5005	8	8	2	7	USINT	Module 2 - CPX-AP-I-4IOL-M12 - Port 7 - PQI

1. Open a web browser and enter the IP address of the CPX-AP-I-EP module.
2. Click **“Modbus TCP”** from tool bar.
3. Select **“Holding Register View”** from drop-down list.

As highlighted in yellow colour are the input & output address related to the slot number – 2 & port – 0.

PQI inputs for Ports 0 and 1 and Ports 2 and 3 are both available in Register 5004 and Register 5005, respectively. More specifically, Port 0 input information are accessible in Byte_LSB of 5004 and Port 1 input details are available in Byte_MSB of 5004.