

VTEM Motion Terminal commissioning with Mitsubishi R-Series PLC by Modbus Simple CPU Communication

This Application Note gives detailed procedure to integrate VTEM Motion Terminal with Mitsubishi R-series PLC through Modbus Simple CPU communication Network. Also describe Library and Function block to control the VTEM Motion Terminal from Festo.

VTEM

Title VTEM Motion Terminal Comm. with Mitsubishi R-Series PLC by Modbus Simple CPU communication.
Version 1.10
Document no. 100471
Originalen
AuthorFesto

Last saved 25.05.2023

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	Important Information	6
1.1	Safety Instructions	6
1.2	Marking Special Information	6
1.2.1	Pictograms	6
1.2.2	Text Symbols and Markers	6
1.2.3	Further Conventions.....	6
2	Introduction	7
2.1	Introduction of VTEM Library	7
2.2	Software's and Hardware's Used	8
2.2.1	PLC Firmware up-dation:.....	8
2.3	Topology	9
3	VTEM Motion Terminal Module Over View.....	10
3.1	VTEM Module Over View	10
3.2	CPX - FB36 Bus node Over View.....	10
3.3	CPX - FB36 Hardware Configuration	11
3.3.1	Arrangement of the DIL switches	11
3.3.2	Procedure to Set DIL switches	11
3.3.3	Setting operating mode and network protocol	11
4	Software Configuration.....	13
4.1	IP Address Configuration of VTEM Parametrizing Port (Using FFT).....	13
4.2	Establish VTEM Module Connection	14
4.2.1	IP Address Configuration of VTEM (using FMT)	15
5	Starting a Project.....	16
5.1	Create New Project File & Configurations of R Series PLC in Gx Works 3	16
	Open MELSOFT GX Works3 development software and start a new project by clicking	16
	1. [Project] => [New] 16	
	Or Select preferred programming language, select your settings or use the ones in the example image.	
	16	
	5. Confirm your selection with the button "OK".	16
5.2	Ethernet port parameters	18
5.2.1	Procedure to set IP address of PLC	18
5.3	Simple CPU Communication Settings	19
5.3.1	Procedure to set IP address for RJ71EN71	19
5.3.2	Procedure to use Simple CPU Communication	19
5.3.3	Simple CPU Communication settings	20
5.4	Adding Library to the project	21
5.4.1	Import Festo_VTEM_DevCon_Rseries_LIB (Library) in FB/Fun under program	21
6	Elements in Festo_VTEM_DevCon_Rseries_LIB.....	25
6.1	General information on Motion Terminal VTEM	25
6.1.1	Documentation for Motion Terminal VTEM	25
6.2	Communication protocol of the Motion Terminal VTEM	26
6.2.1	Motion Terminal VTEM Function and Parameterization.....	26

6.3	Connecting I/O Data to the Function Blocks	28
7	Festo_VTEM_DevCon_Rseries_LIB's Function Blocks	29
7.1	FB_ValveControl – Control of a Valve by Operating a Motion App	29
7.1.1	Inputs and Outputs	29
7.1.2	Response to Valve Mode Set	30
7.2	FB_Upload_Single_Parameter – Reading Single Value from the Motion Terminal	31
7.2.1	Inputs and Outputs	31
7.3	FB_Download_Single_Parameter – Transfer Single Parameters to the Motion Terminal.....	32
7.3.1	Inputs and Outputs	32
7.3.2	Response to Transfer Mode	33
7.4	FB_Save_Configuration_Persistent – Save Parameterization Persistent	33
7.4.1	Inputs and Outputs	33
7.4.2	Response to Save Persistent.....	34
7.5	FB_ReadMalfunctionList – Read Out Malfunction List	34
7.5.1	Inputs and Outputs	34
7.5.2	Response to Read Malfunction List.....	35
7.5.3	stMalfunction	36
7.5.4	stTimeStamp.....	36
8	Configuration of VTEM Function blocks in Gx Works3	37
8.1	Configuring Program	37
8.2	Adding FB in Program and Assigning Inputs and Outputs Tags	37
8.3	Add “FB_ValveControl” in PLC Program.....	41
8.4	Add “FB_Upload_Single_Parameter” in PLC Program	41
8.5	Add “FB_Download_Single_Parameter” in PLC Program.....	41
8.6	Add “FB_Save_Configuration” in PLC Program.....	42
8.7	Add “FB_ReadMalfunctionList” in PLC Program	42
9	PLC Configuration Transfer to Controller	44
9.1	Downloading Mode	44
9.2	Monitor Mode.....	47
9.2.1	Checking Connection:	49
10	Sample Code For VTEM Function Block.....	51
10.1	Sample Code for Mapping Modbus Simple CPU Communication I/O Data to Internal I/O Array Tag	51
10.2	Sample Code for Control the VTEM Motion Terminal Valve Using “FB_ValveControl” FB	52
10.3	Sample Code for Upload VTEM Motion Terminal Valve Parameter Using “FB_Upload_Single_Parameter” FB	53
10.4	Sample Code for Download VTEM Motion Terminal Valve Parameter Using “FB_Download_Single_Parameter” FB	54
10.5	Sample Code for Save the VTEM Motion Terminal Valve Parameter Using “FB_Save_Configuration” FB ..	55
10.6	Sample Code for Read the Malfunction list of the VTEM Motion Terminal Valve Using “FB_ReadMalfunctionList” FB	55

1 Important Information

1.1 Safety Instructions

When commissioning and programming pneumatic systems, you must observe the safety regulations in the descriptions and operating instructions for the components used. The user must ensure that nobody has access to the sphere of influence of the connected actuators. Access to the potential danger area must be prevented by suitable measures such as barriers and warning signs.

1.2 Marking Special Information

1.2.1 Pictograms

The following pictograms mark passages in the text containing special information:



Caution

Type and source of danger to equipment.

Result if disregarded.

- Action to avoid danger.



Information

Recommendations, tips and references to other sources of information



Warning

- Instruction



Note

Text.

- Instruction.

1.2.2 Text Symbols and Markers

1. ---> Figures denote activities that must be carried out in the order specified.
- ---> Bullet points denote activities that can be carried out in any order.
 - ---> The dash heads indicate general lists.

1.2.3 Further Conventions

"FBs"	In some places the abbreviation "FB" is used for "function block" (also "FBs")
"OK"	Names of windows, dialogs and buttons such as "Message Window", "Dearchive Project", "OK" as well as designations are shown in inverted commas.
CTRL	Names of keys on the PC keyboard are represented in the text with uppercase letters (e.g. ENTER KEY, CTRL, C, F1 etc.).
CTRL+C	For some functions, two keys must be pressed at the same time. Example: The CTRL key together with the "C" key produces the CTRL+C character.
	Where it says "click" or "double-click", this always refers to the left mouse button. If the righthand mouse button is to be used, this will be explicitly mentioned.

2 Introduction

The Festo_VTEM_DevCon_Rseries_LIB library is for Mitsubishi GXWorks3 with R-Series PLC.

The function blocks of the library support the user in the parameterisation and operation of Motion Apps on the Motion Terminal VTEM from Festo. Here the Motion Terminal VTEM can be controlled and parameterised via Mitsubishi R-Series PLC through CPX-FB36 fieldbus node (EtherNet/IP Modbus TCP).



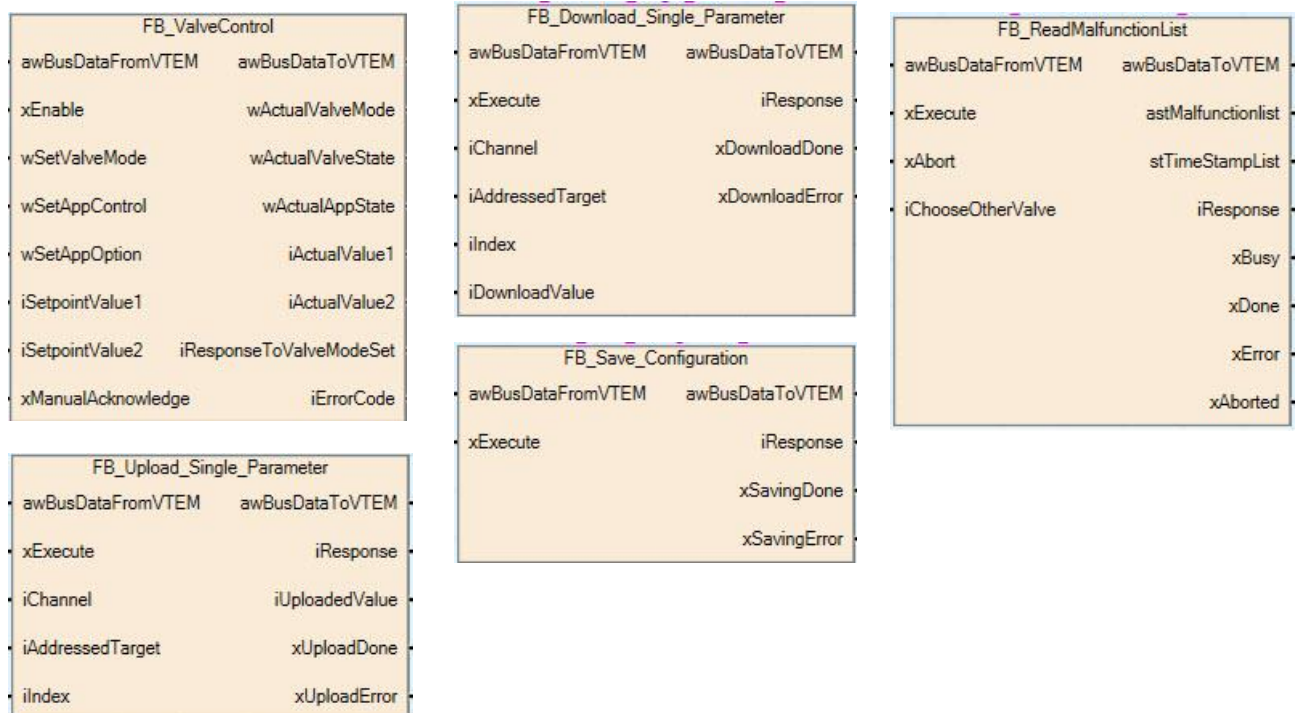
Information

The function blocks represent a simplified operation of the protocol which is described in the user documentation of the Motion Terminal VTEM. All functions of the function blocks described here can also be implemented with the corresponding effort in the user program directly via the cyclic process data of the Motion Terminal.

Detailed information about the function, operation and parameterisation is available in the user documentation of the Motion Terminal VTEM (→ [Documentation for Motion Terminal VTEM](#)).

2.1 Introduction of VTEM Library

This library contains five Function blocks they are used to Control, Download, Upload, Save a parameters to VTEM motion terminal & diagnose the error details of the Motion terminal. All five function blocks are shown below & they are named as **“FB_ValveControl”**, **“FB_Upload_Single_Parameter”**, **“FB_Download_Single_Parameter”**, **“FB_Save_Configuration”**, & **“FB_ReadMalfunctionList”**.



2.2 Software's and Hardware's Used

Type/Name	Software Version	Remarks
Festo CPX-FMT	V4.21.210	Software
GxWorks3	V1.090U & above	Software
R..-ENCPU (Mitsubishi PLC)	Firmware V52 & above	Hardware
Festo VTEM (Motion Terminal)	4.24 & above	Hardware
Festo CPX-FB36 (EtherNet/IP Mod-bus TCP)	Rev 22	Hardware
Ethernet Cable	Version Free	Hardware



Warning

- **“Simple CPU communication”** is possible only in **“R..EN” CPU (“EN” is mandatory, only Port 1 of RJ71EN71 supports Simple CPU communication)**. And also for **PLC Firm ware version 52** and above.
- **Simple CPU communication (Modbus TCP Compatible)** is possible in **GX Works3 version 1.090U** and above.

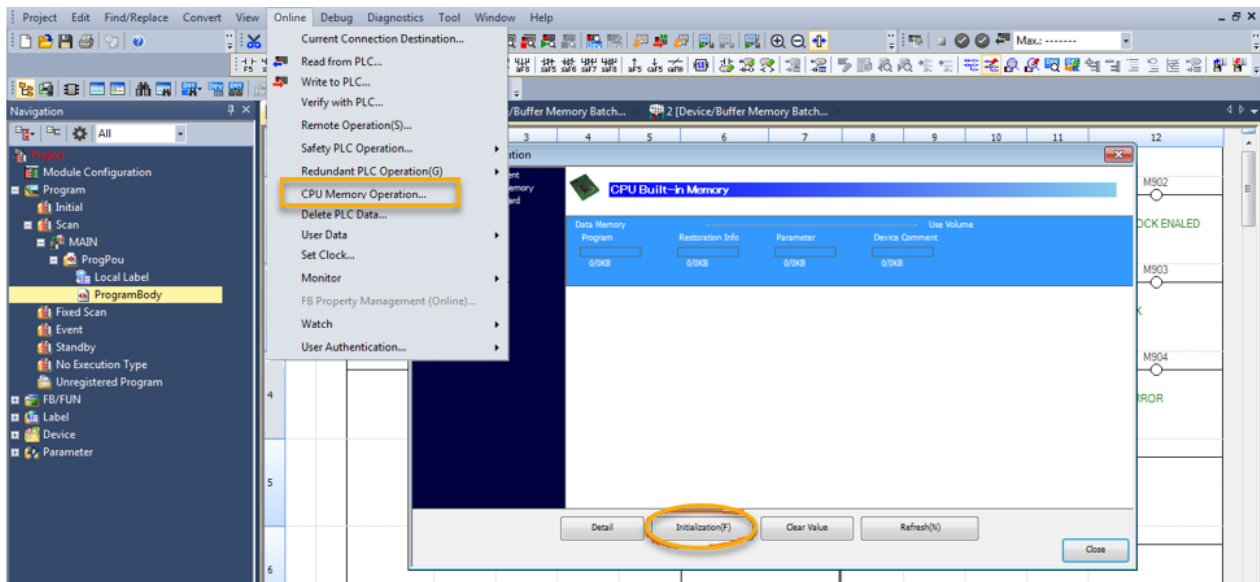
2.2.1 PLC Firmware up-dation:

- Initial Checks: (PLC year of Manufacture, PLC Firm ware, Software- GX Works3 - 1.090 (Version) and above.
- Check Firm ware version of PLC- should be 52 & above (GX Works 3- In Diagnostics, System monitor- Firm ware version data will be available). If it less than 52, update Firm ware using SD Card.

Firm ware Update Procedure- In empty SD Card, put Firm ware file.

Take Program Backup of PLC Programs, including intelligent module if available.

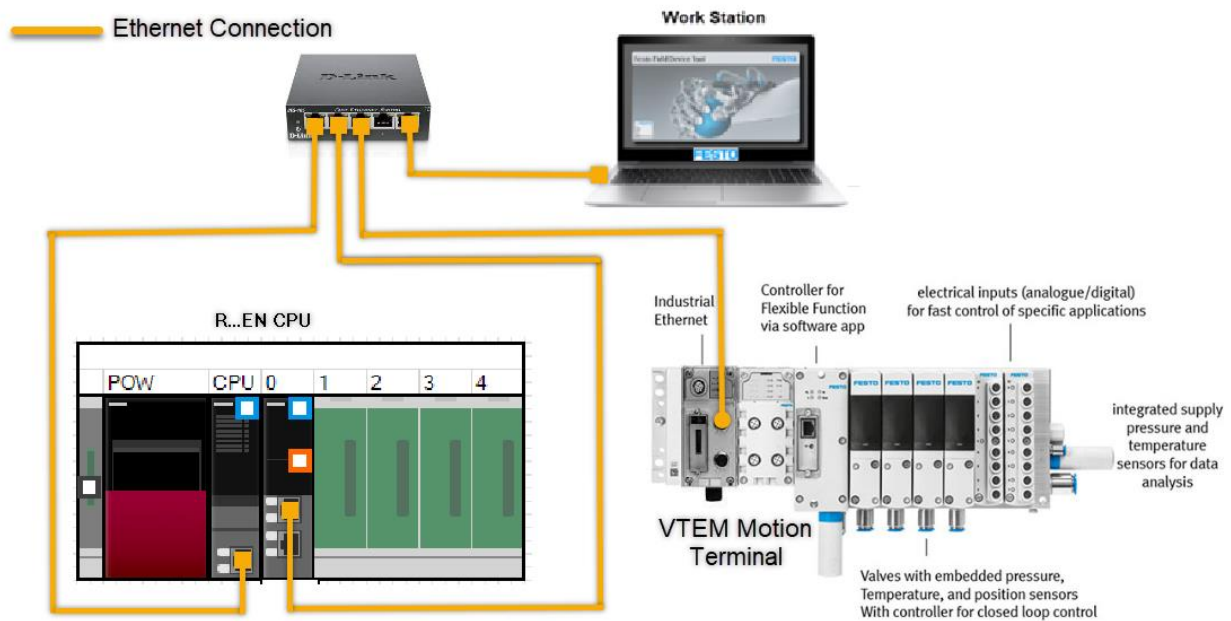
Initialize PLC- Online- CPU Memory operation-Initialize.



- Make PLC stop mode, switch off PLC , insert memory card, switch ON PLC, Card LED in PLC will get ON (denotes updating), Error LED and P. RUN LED blinks (denotes completion of Update).
- Off PLC, remove Memory card, make PLC Run mode, and switch ON PLC, cross check Firmware.

2.3 Topology

When we want to communicate between Mitsubishi R PLC via Modbus protocol the following system architecture must be followed.



Note

- Above network configuration is an example network for this application note but depends on users network configuration above architecture may change.



Warning

- “Simple CPU communication” is possible only in “R..EN” CPU (“EN” is mandatory, only Port 1 of RJ71EN71 supports Simple CPU communication).
- Simple CPU communication (Modbus TCP Compatible) is possible in GX Works3 version 1.090U and above.

3 VTEM Motion Terminal Module overview

3.1 VTEM Module overview

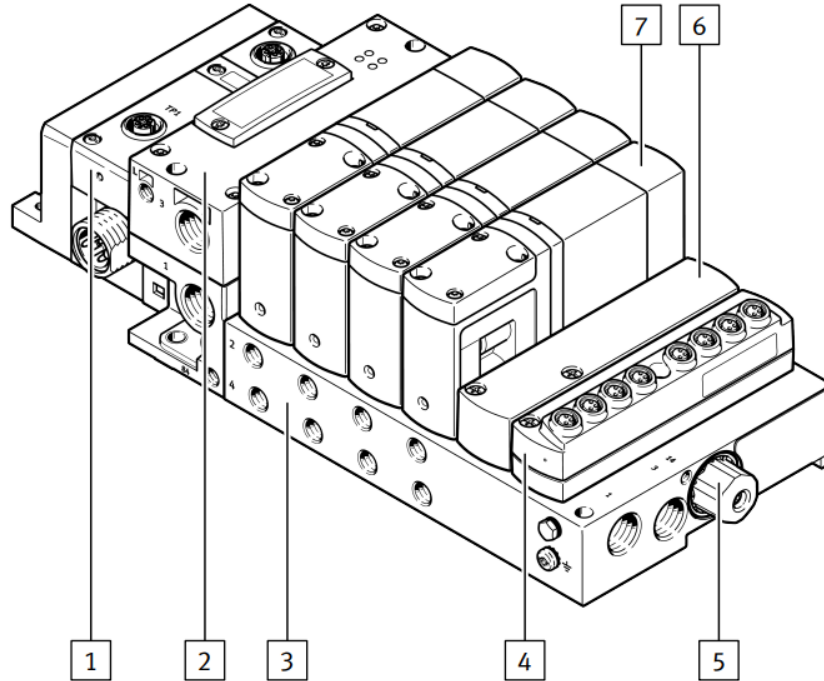
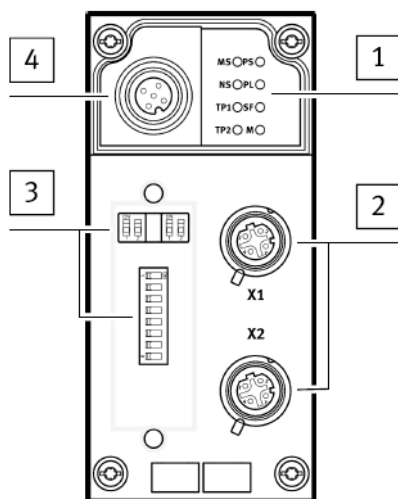


Fig. 6: Product design

- | | |
|---|--|
| 1 CPX terminal electronics side | 5 Pilot pressure regulator |
| 2 Controller Motion Terminal CTMM-S1-C | 6 Cover plate VABB-P11-27-T (optional) |
| 3 Manifold rail | 7 Valve slice VEVM-S1-27-... |
| 4 Input module CTMM-S1-A/D-... (optional) | |

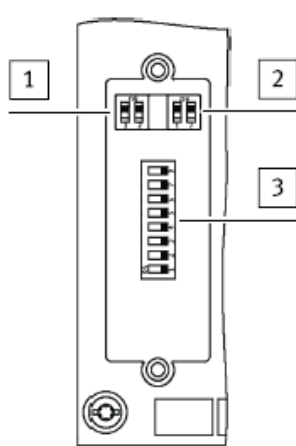
3.2 CPX - FB36 Bus node overview



- | |
|---|
| 1 Network-specific and CPX-specific LED displays |
| 2 Network connections [X1], [X2]
(2 x socket, M12, D-coded, 4-pin) |
| 3 DIL switch |
| 4 Service interface for operator unit CPX-MMI or Festo
Maintenance Tool CPX-FMT
(socket, M12, A-coded, 5-pin) |

3.3 CPX - FB36 Hardware Configuration

3.3.1 Arrangement of the DIL switches



- 1 DIL switch 1:
Operating mode and network protocol
- 2 DIL switch 2:
Diagnostic mode or data field size
(depending on the set operating mode)
- 3 DIL switch 3:
IP addressing

3.3.2 Procedure to Set DIL switches

1. Switch off power supply.
2. Remove the DIL switch cover
3. Make the settings for the DIL switches → 3.3.3 Setting operating mode and network protocol ...
- 3.3.4 Setting IP addressing.
4. Replace the cover of the DIL switches

3.3.3 Setting operating mode and network protocol

Step – 1 DIL Switch Group – 1.1

DIL switch 1.1	Mode of operation
 DIL 1.1: OFF (factory setting)	Remote I/O All functions of the CPX terminal are controlled directly by the higher-order controller (SPS). A control block integrated into the CPX terminal (e.g. CPX-CEC or CPX-FEC) works as a passive function module without controller.
 DIL 1.1: ON	Remote Controller A control block integrated into the CPX terminal (e.g. CPX-CEC or CPX-FEC) controls the I/O controller. This operating mode is only useful if a control block is integrated into the CPX terminal.

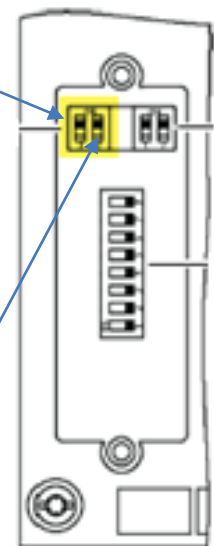
Leave DIL 1.1 should be in factory setting (OFF) to give the control to PLC.

Step – 2 DIL Switch Group – 1.2

DIL switch 1.2	Network protocol
 DIL 1.2: OFF (factory setting)	EtherNet/IP The CPX terminal uses the EtherNet/IP network protocol.
 DIL 1.2: ON	Modbus TCP The CPX terminal uses the EtherNet/IP network protocol.

Leave DIL 1.2 should be in setting (ON) to give the network protocol - Modbus TCP.

Depend on diagnostic mode DIL 1.1 & DIL 1.2 need to be configure. Above table shows configuration details. In our case we are using VTEM in Modbus TCP.



Step – 3 DIL Switch Group -2

Leave both DIL 2.1 & DIL 2.2 should be in OFF state (Factory setting).

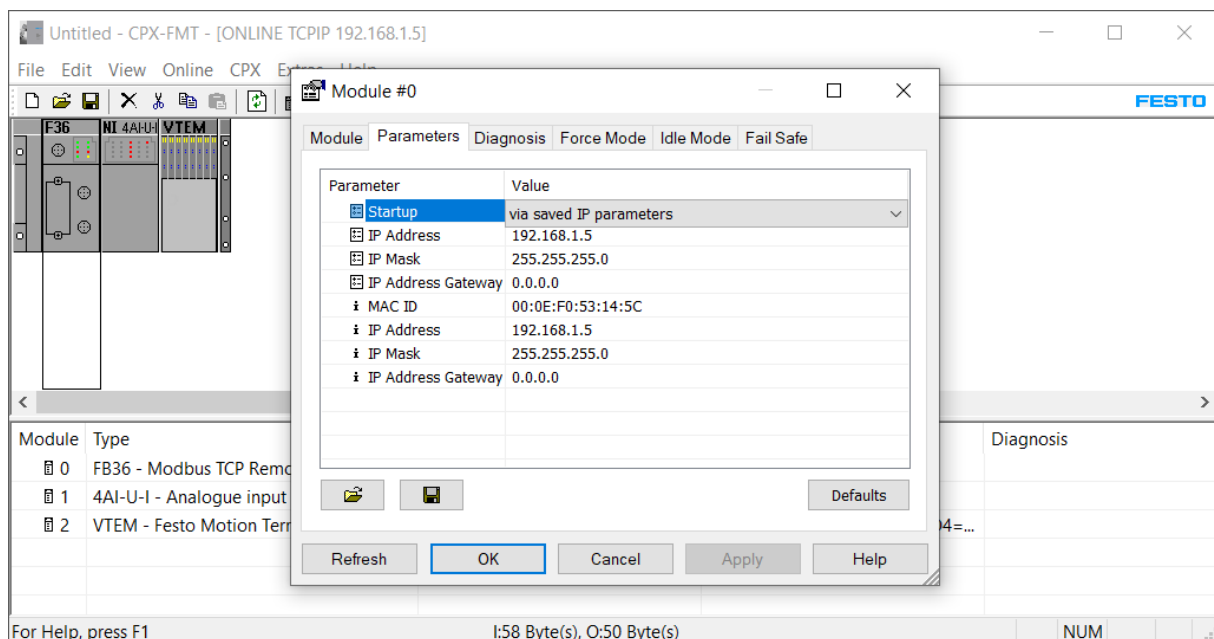
Step – 3 DIL Switch Group - 3

DIL switch3			IP addressing
	DIL 3.8:	$2^7 = 128$	The type of addressing or the host ID of the IP address of the bus node is set via DIL switch elements 3.1 ... 3.8. Possible settings: 0 = Dynamic addressing via DHCP/BOOTP 1 ... 254 = Permissible address range 255 = Reset all IP parameters to factory setting Factory setting: 0
	DIL 3.7:	$2^6 = 64$	
	DIL 3.6:	$2^5 = 32$	
	DIL 3.5:	$2^4 = 16$	
	DIL 3.4:	$2^3 = 8$	
	DIL 3.3:	$2^2 = 4$	
	DIL 3.2:	$2^1 = 2$	
	DIL 3.1:	$2^0 = 1$	

DIL switch group – 3 used to configure the IP address of VTEM module. We will configure the IP address of VTEM module using FMT tool so ensure all the DIL switches from DIL 3.1 to DIL 3.8 should be in OFF condition to avoid IP address configuration through DIL switch.

Example with IP address: 192.168.001.005		Example with IP address: 192.168.001.038	
	$2^0 + 2^2 =$		$2^1 + 2^2 + 2^5 =$
	$1 + 4 =$		$2 + 4 + 32 =$
	5		38

Once all the setting complete user can verify the setting using CPX Maintenance Tool as shown below



1. Open the CPX Maintenance Tool
2. Connect the CPX Maintenance Tool to device.
3. Double Click the FB36 Module to view the settings.

4 Software Configuration

This chapter explain in detail about software configuration of VTEM Motion Terminal & Mitsubishi PLC.

4.1 IP Address Configuration of VTEM Parametrizing Port (Using FFT)

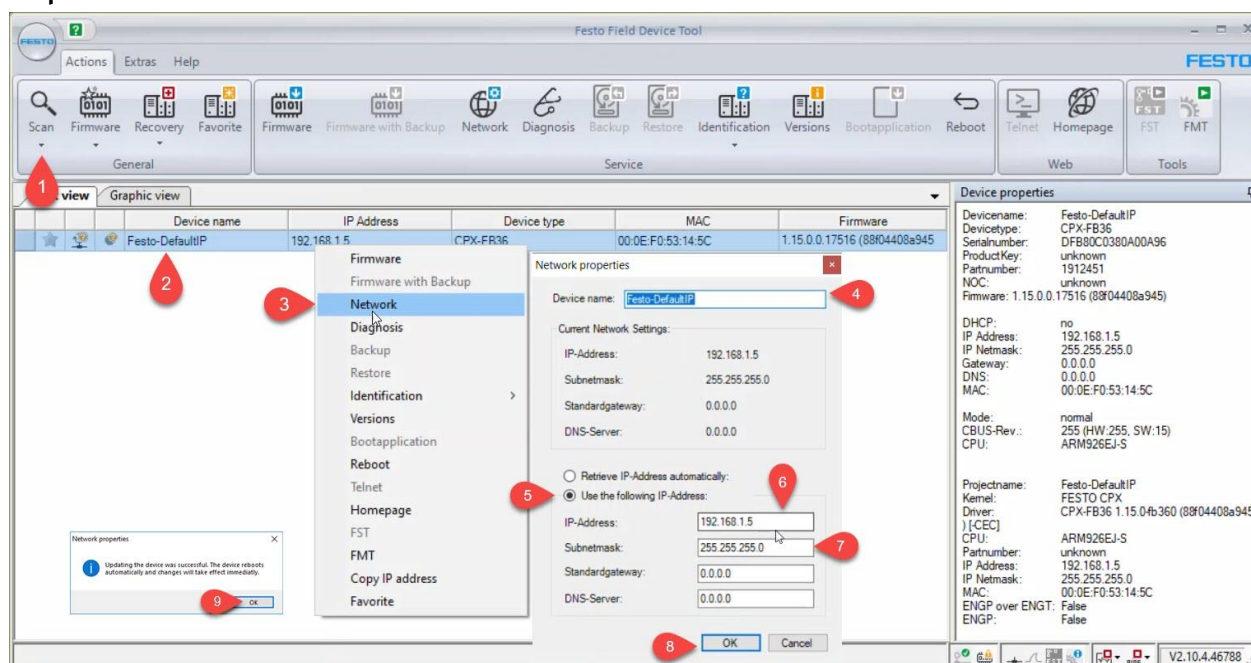
This chapter describes about IP address configuration of VTEM parametrizing port using Festo Field Device Tool.



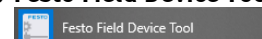
Note

- Before starting VTEM, IP address configuration process make sure the hardware connection between PC & VTEM is done as per communication architecture shown in [chapter 2.3](#). In the [chapter 2.3](#) Orange color connection is for parametrization and after parametrization to establish connection with PLC.
- User can download the Festo Field Device Tool from Festo support portal website using below link.
https://www.festo.com/net/en-gb_gb/SupportPortal/Default.aspx?tab=0&q=8004365

Step – 1



Open the Festo Field Device Tool from following path **Windows Start > Festo Software > Festo Field Device Tool**



1. Select **“Scan Button”** to update the connected device list.
2. Select the correct drive from device list based on identifiers like Device name & MAC address of devices.
3. Right click the selected device & click the **“Network”** option from right click menu.
4. Enter the **“New name”** for the drive. It’s an optional point by default drive name will be device type name.
5. Select **“Use following IP-Address”** option for enable static IP mode. By default the IP address mode will in Dynamic mode.
6. Enter the **new IP address** for the drive. Make sure the drive IP series must meet series of PLC IP address.
7. Enter the **subnet mask** value to define capacity of the network range. Above network configuration done for 255 devices.
8. Click **“OK”** to confirm configuration.
9. Click **“OK”** button from pop-up window to confirm the reboot the drive.

4.2 Establish VTEM Module Connection



Note

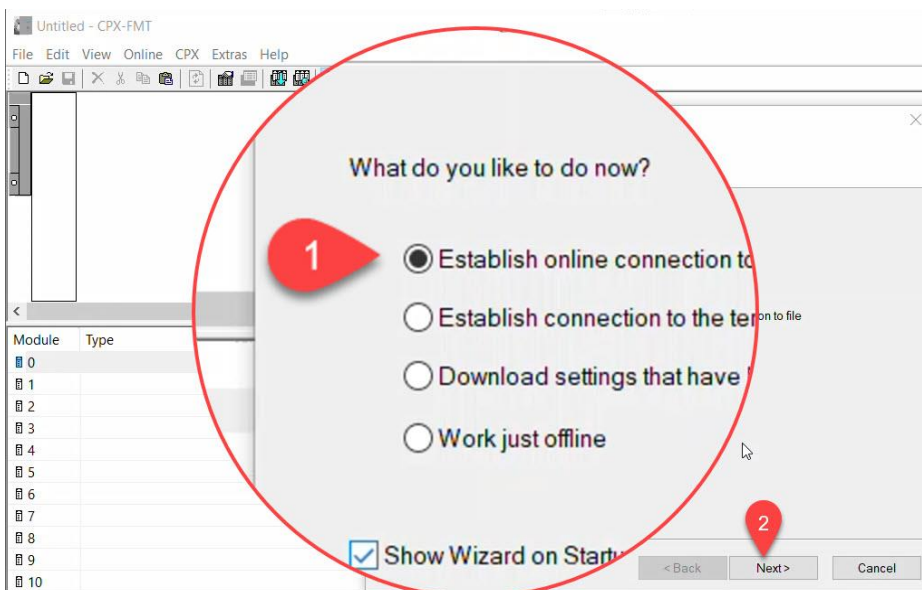
- Before starting VTEM, IP address configuration process make sure the hardware connection between PC & VTEM is done as per communication architecture shown in [chapter 2.3](#). In the [chapter 2.3](#) orange color connection is for parametrization and after parametrization to establish connection with PLC.
- User can download the CPX Festo Maintenance Tool form Festo support portal website using below link.
https://www.festo.com/net/en-id_id/SupportPortal/default.aspx?q=cpx&tab=4&s=t#result

Step – 1



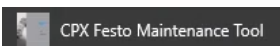
Note

- Ensure the USB serial communication cable or EtherNet Cable has been established between work station & VTEM module as shown in Architecture.

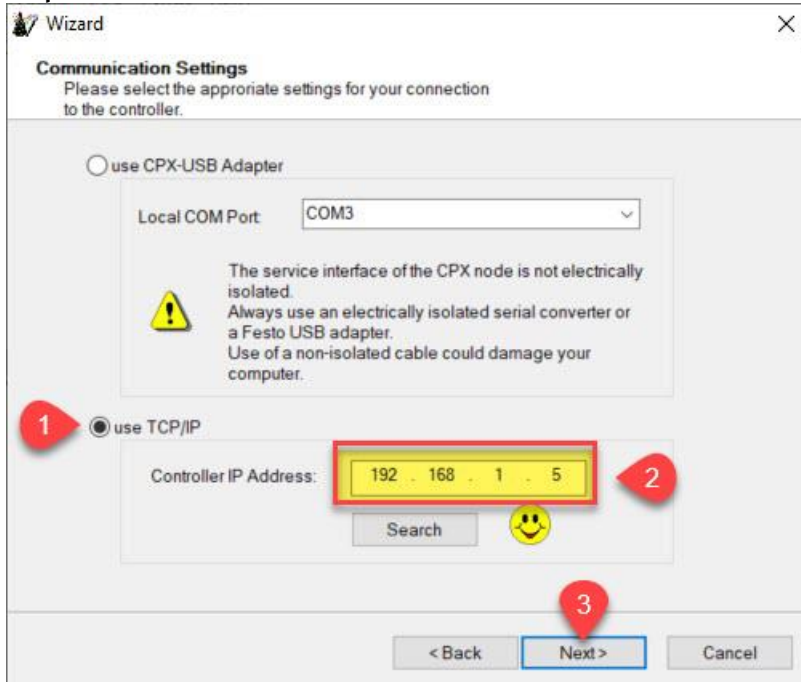


Open the CPX Festo Maintenance Tool from following path

Windows Start > Festo Software > CPX Festo Maintenance Tool

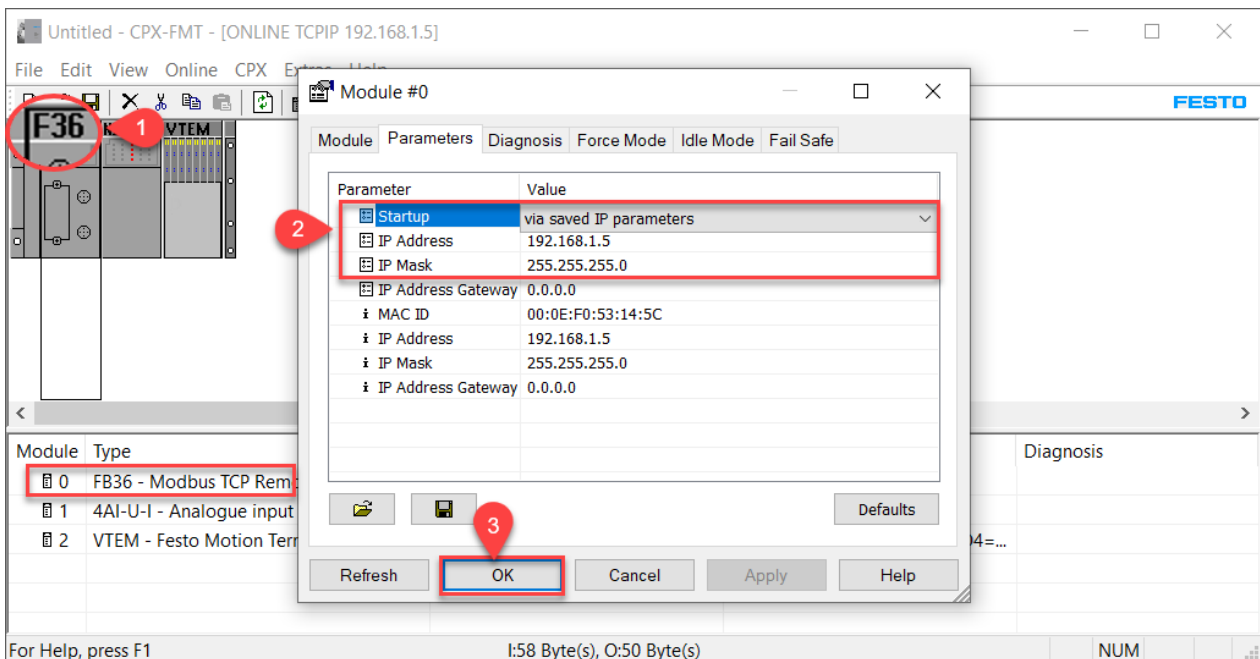


- Once FMT software opens Welcome POP-UP window will appear automatically. From there select “**Establish online connection**” option. If welcome pop-up not appear got to tool bar Extras > Preference then jump to Step - 2.
- Select “Next” button.

Step – 2**Note**

- If USB serial communication cable is connected between work station & VTEM module then select **“Use CPX-USB Adapter”**

1. Select **“use TCP/IP”** option from preference window.
2. If Ethernet cable is used then use TCP/IP option from preference window and enter PLC TCP/IP address of VTEM-FB36 to establish connection.
3. Click **“Next”** Button.

4.2.1 IP Address Configuration of VTEM (using FMT)

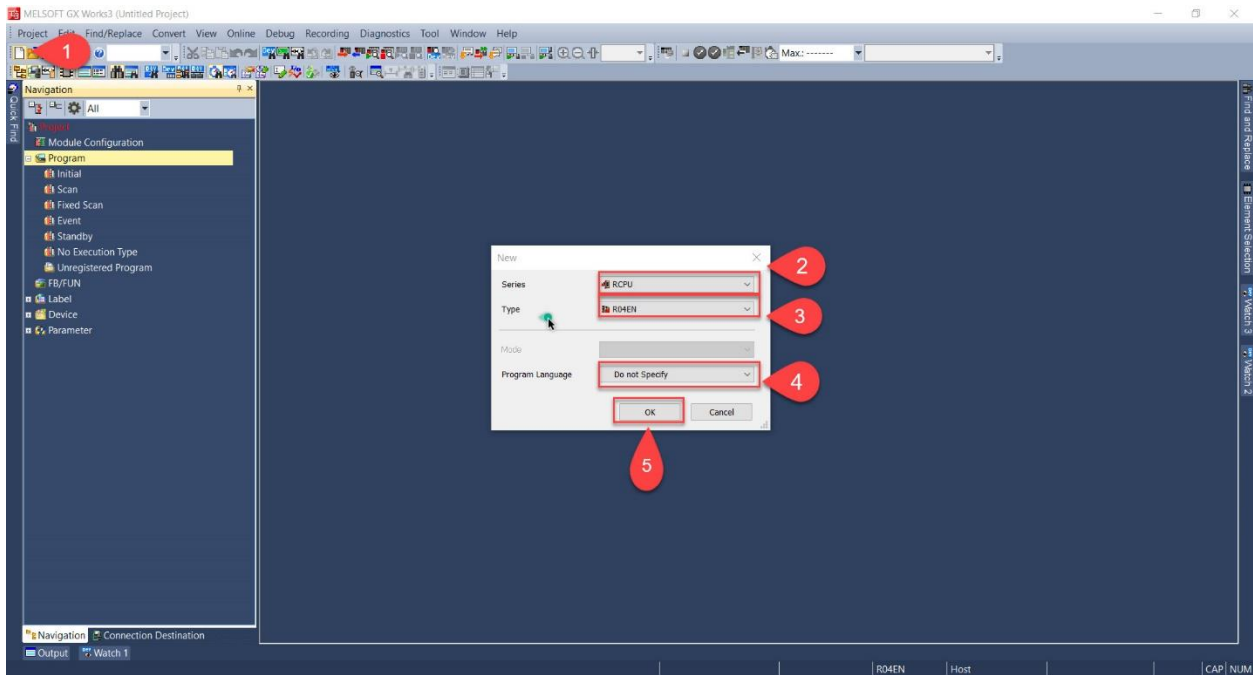
1. Double click on **“FB36 Module”**.
2. Note the **“EtherNet/IP address”** of VTEM module.
3. Click **“ok”** Button.

5 Starting a Project

5.1 Create New Project File & Configurations of R Series PLC in Gx Works 3

This chapter will also explain how to create step by step to create project, procedure to establish communication settings, software configurations like PLC IP address setting, Modbus Simple CPU Communication configuration and also adding Library in Gx Works3 programming tool.

Step – 1



Open MELSOFT **GX Works3** development software and start a new project by clicking

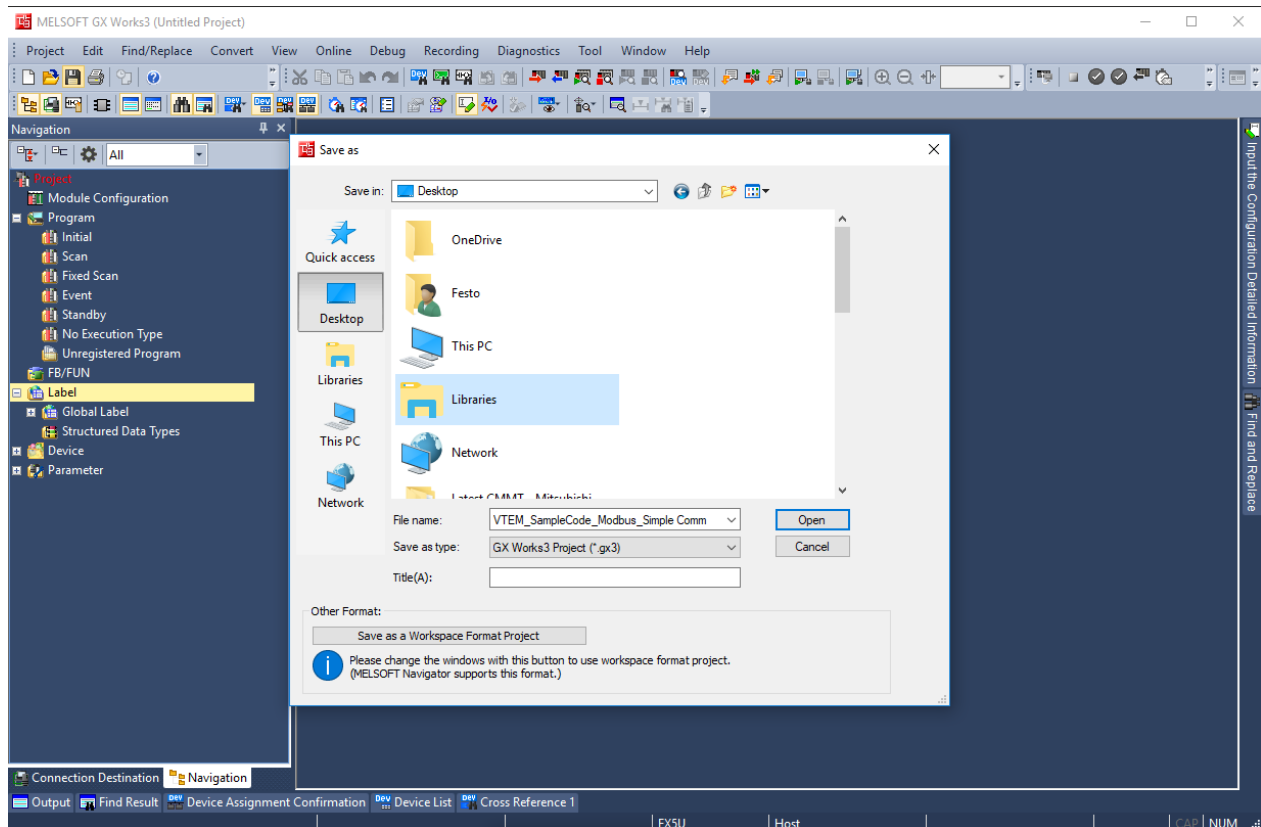
1.  [Project] => [New]
Select **"New Project"** to create a new project.
2. Select **"RCPU"** from **"Series"** drop down option.
3. Select your CPU model from **"Type"** drop down option.
4. Select **"Do not Specify"** as programming language from **"Programming Language"** drop down option.
Or Select preferred programming language, select your settings or use the ones in the example image.
5. Confirm your selection with the button **"OK"**.



Note

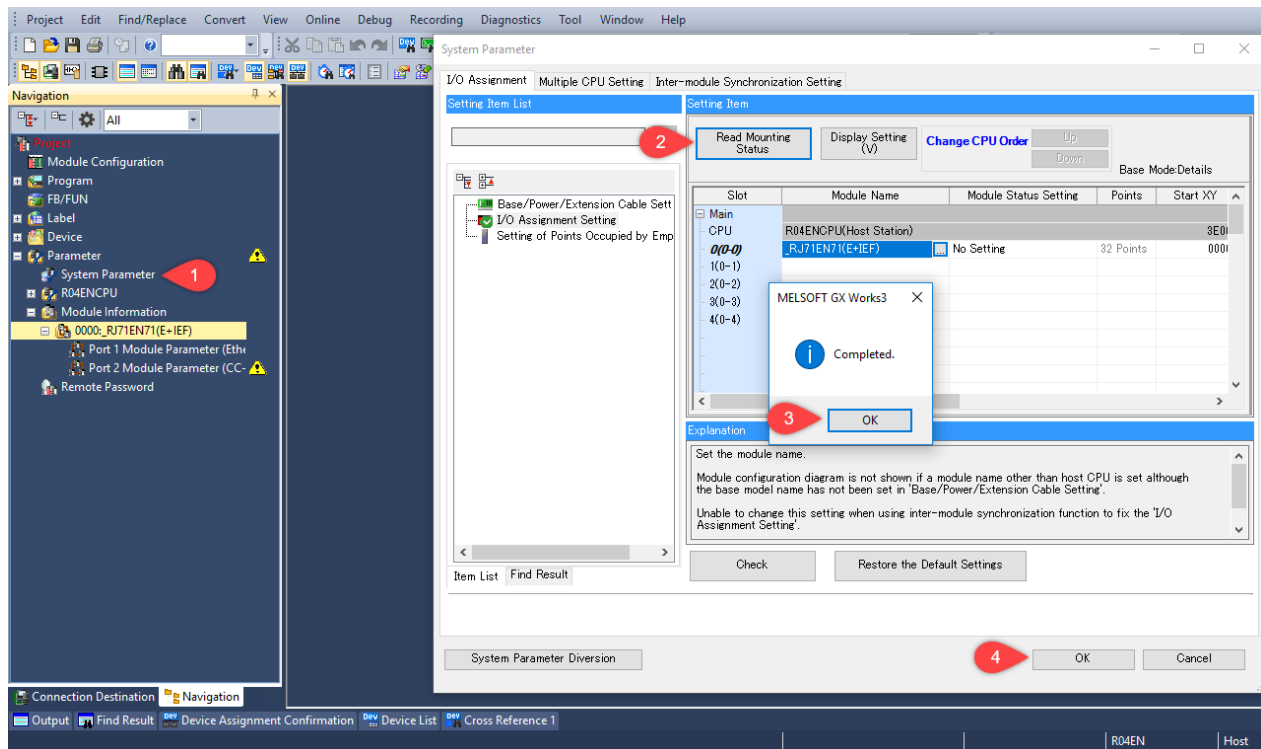
- Users can also create **"New Project"** by clicking shortcut key **CTRL+N** or **ICON** .
- CPU model and Programming language can be select as per user requirement.
- In this App note as **example** we have considered **R04EN** CPU.

Step – 2



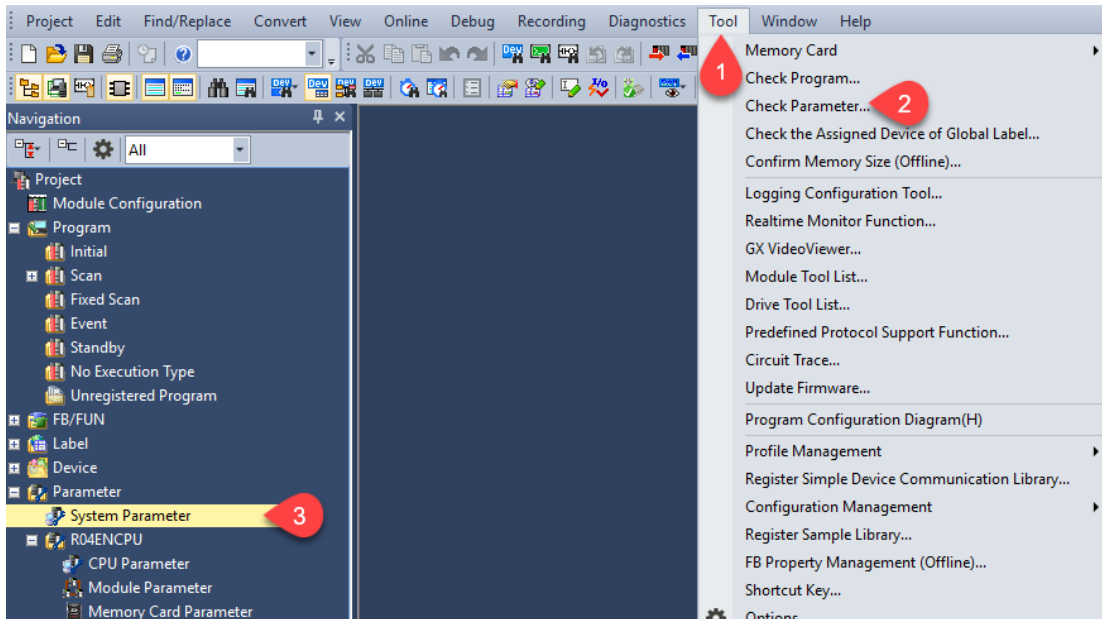
Save the project as desired.

Step – 3



1. Click on “System Parameter” under Parameter tree.
2. Click on **“Read Mounting Status”** to get the Hardware Module configuration.
3. Click on **“OK”** in Completed pop up.
4. Press **“OK”** to complete the process.

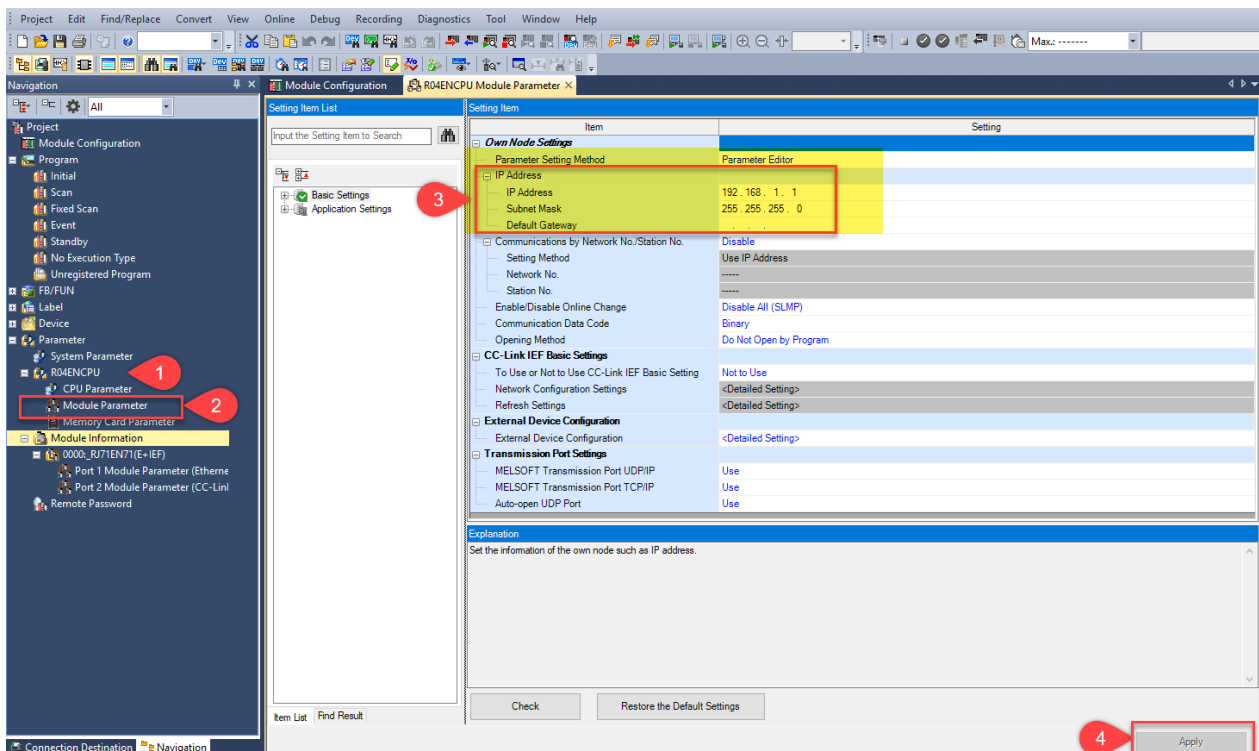
Step – 4



1. Click on **Tool**.
2. Click **Check Parameter**.
3. Check whether Warning symbol cleared.

5.2 Ethernet port parameters

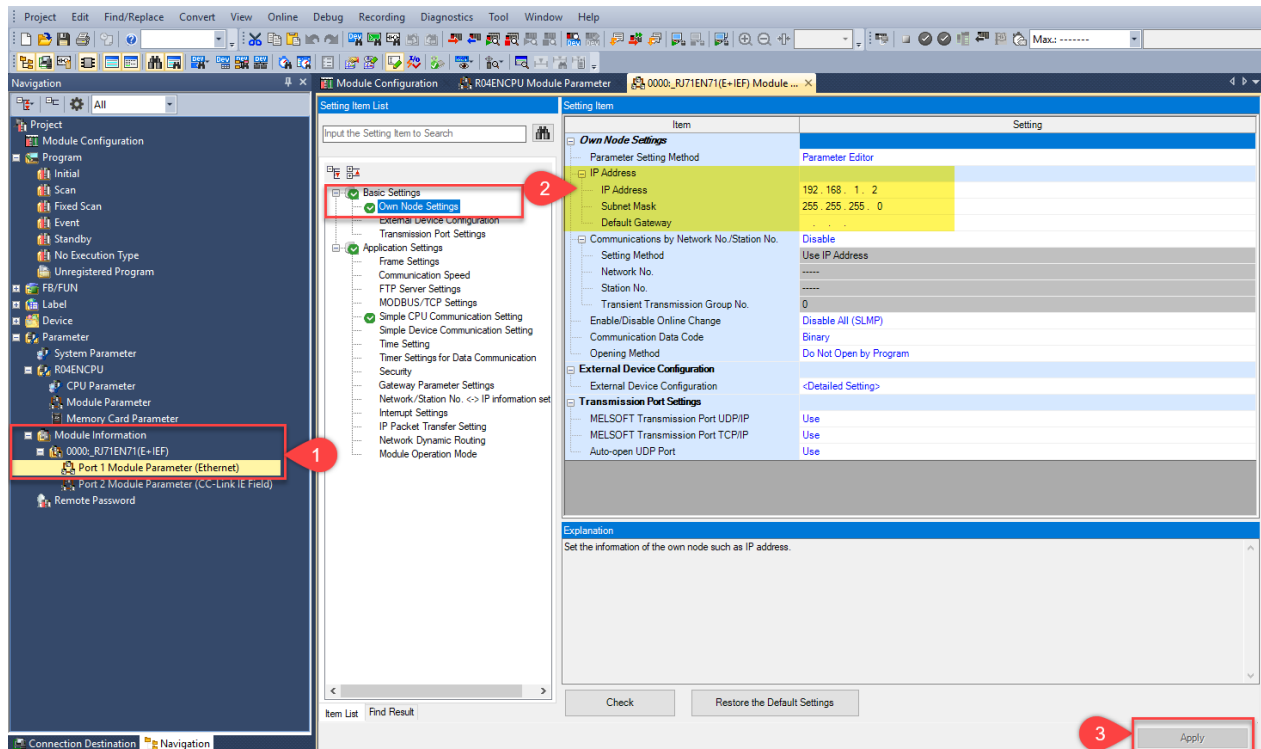
5.2.1 Procedure to set IP address of PLC



1. Expand **"R04ENCPU"** from **"Parameter"** option.
2. Double click **"Module Parameter"** option.
3. Enter the **"IP Address & Subnet Mask"** of the PLC as per requirement. as example shown in above image.
4. Click **"Apply"** button to confirm the configuration.

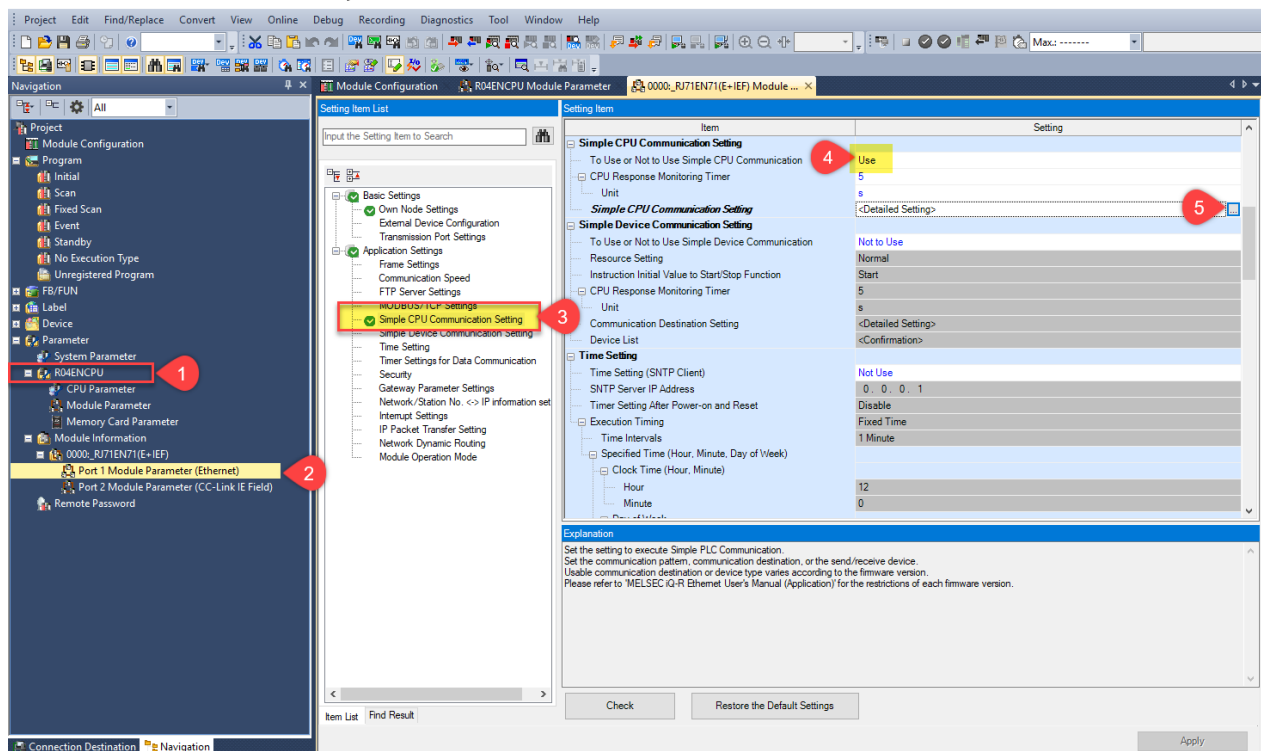
5.3 Simple CPU Communication Settings

5.3.1 Procedure to set IP address for RJ71EN71



1. Under Module information expand RJ71EN71(E+IEF) and click on **“Port1 Module Parameter (Ethernet)”**.
2. Enter the **“IP Address & Subnet Mask”** as example shown in above image.
3. Click **“Apply”** button to confirm the configuration.

5.3.2 Procedure to use Simple CPU Communication



1. Click on **“Parameter”**, Expand **“Module Information”**.
2. Double click on **“Port 1 Module Parameter (Ethernet)”** of **RJ71EN71(E+IEF)** under Module information.

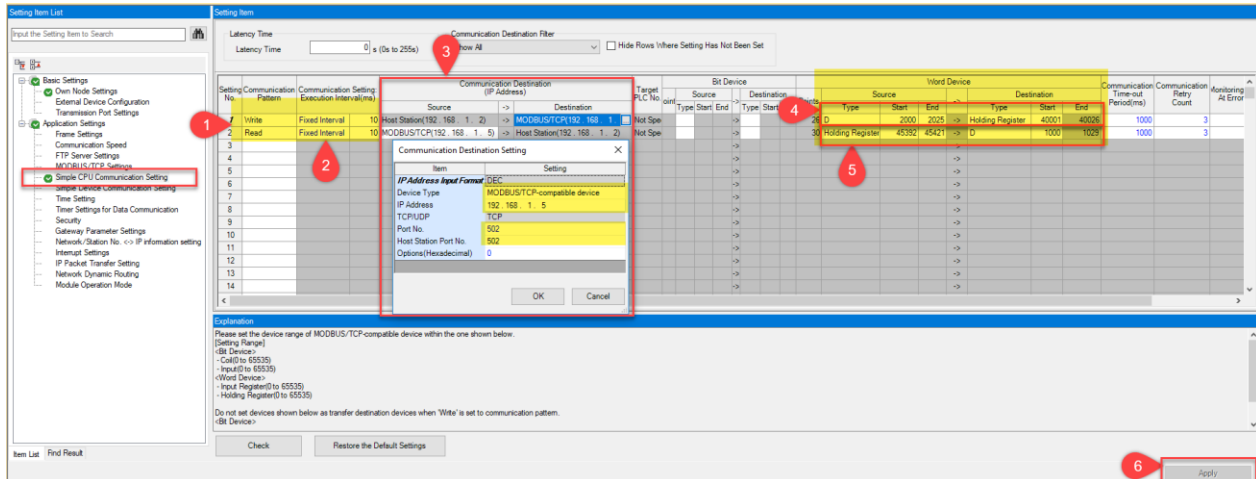
3. Expand **"Application Settings"** and click on **"Simple Communication Setting"**.
4. Select **"Use"** in To Use or Not to Use Simple CPU Communication.
5. Click on **"Detail Settings"** in Simple CPU Communication setting.



Warning

"Simple CPU communication" is possible only in **"R..EN"** CPU (**"EN"** is mandatory, only **Port 1** of **RJ71EN71** supports Simple CPU communication).

5.3.3 Simple CPU Communication settings



1. Select **"Write/Read"** in Communication Pattern.
2. Set **"Fixed Interval"** in Communication Setting & **"10"** as Execution Interval(ms).
3. Click on **"Please Set"** in Source & Destination column of Communication Destination (IP Address).
In Device Type select **"MODBUS TCP Compatible Device"**.
In **"IP Address"** set **CPX-FB36 Module IP Address**.
In **"Host Station Port No"** Set **502**.
Click **OK**.
4. Under **"Write"** setting – Ref: Fig. FMT image taken for to note Input Bytes and Output Bytes.
-> Word Device -> Source Type -> select Register type (**D/W/R/SW/SD**) and set user defined **Start Address** and **End Address**.
-> Word Device -> Destination Type -> select **"Holding Register"** and set Modbus **Start Address – 40001**.
5. Under **"Read"** setting – Ref: Fig. FMT image taken for to note Input Bytes and Output Bytes.
-> Word Device -> Source Type -> select **"Holding Register"** and set Modbus **Start Address – 45392** and **End Address – 45421**. – Ref: Fig: Modbus Address for Reference, taken from CPX-FB36 Manual.
-> Word Device -> Destination Type -> select Register type (**D/W/R/SW/SD**) and set Modbus **Start Address – 1000**. – Ref: Fig: Modbus Address for Reference, taken from CPX-FB36 Manual.
6. Click **"Apply"** to save setting.



Note: Max Write Points for each setting is 246 bytes & Read points 250 bytes. Next bytes can be read by assigning in next setting number. This limitation is same in FB as well.

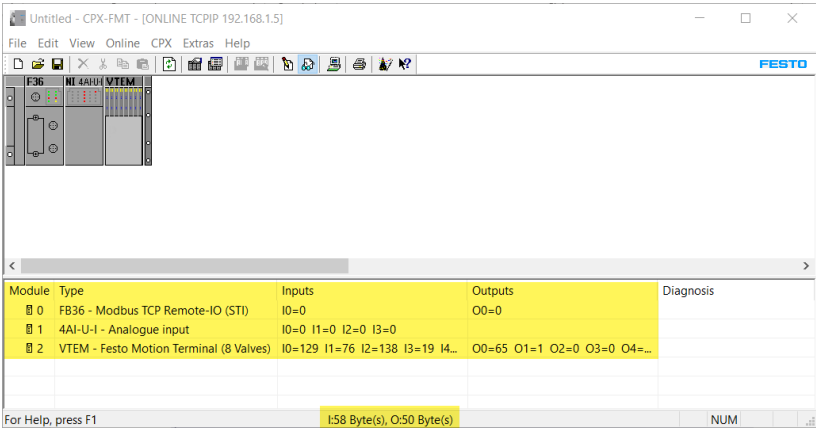


Fig. FMT image taken for to note Input Bytes and Output Bytes.

- Based on the above image Input size will be 29 Word & Output size will be 25 Word

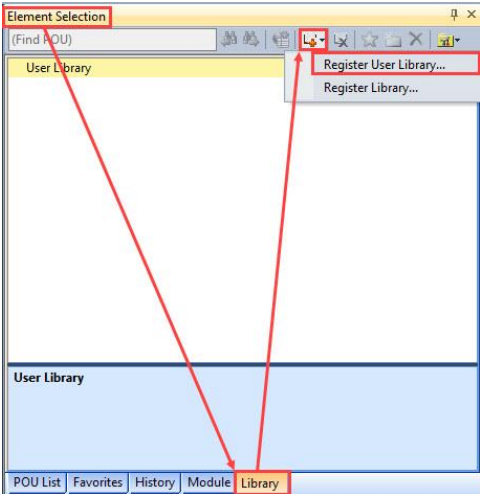
Modbus command	Function code	Modbus address	Meaning	Remote-I/O 16-bit access	Group
read 4x registers	3	45357 ... 45391 45392 ... 45647 45648 ... 45655 45656 ... 46055 46100	CPX status information Process data inputs Diagnostic memory parameters Diagnostic memory data Modbus connection timeout	read read read read read	O B C C O
write 4x registers	6, 16	40001 ... 40256 40257 ... 40264 46100	Process data outputs Diagnostic memory parameters Modbus connection timeout	write write write	D I O
read/write 4x registers	23	45357 ... 45391 45392 ... 45647 45648 ... 45655 45656 ... 46055 40001 ... 40256 40257 ... 40264	CPX status information Process data inputs Diagnostic memory parameters Diagnostic memory data Process data outputs Diagnostic memory parameters	read read read read write write	O B C C D I
read device identification	43	Objects	Objects ID0, 1, 2, 3, 4, 5	read	F

Tab. 114 Overview of the Modbus function codes for the bus node CPX-FB36 in operating mode Remote I/O

Fig: Modbus Address for Reference, taken from CPX-FB36 Manual.

5.4 Adding Library to the project

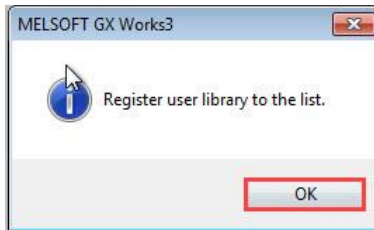
5.4.1 Import Festo_VTEM_DevCon_Rseries_LIB (Library) in FB/Fun under program Step –1



- In **Element Selection** Select **Library**.
- Select on **“Register to Library List”**
- Click on **“Register User Library”**

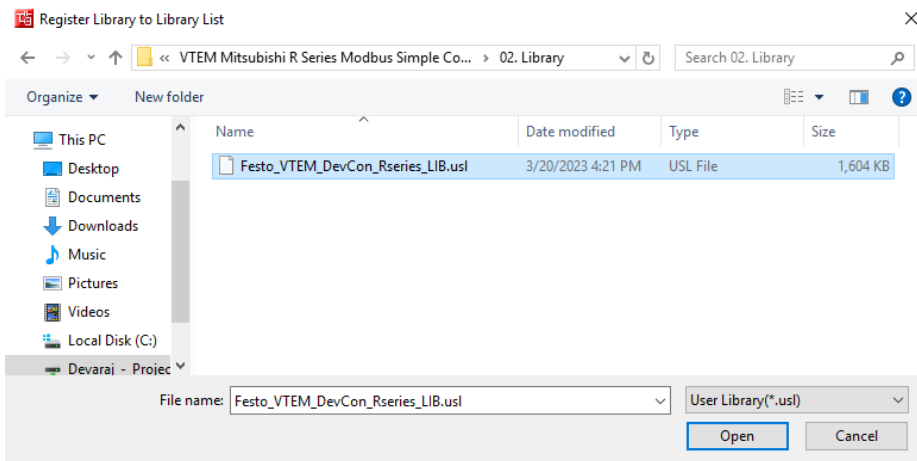
Starting a Project

Step – 2



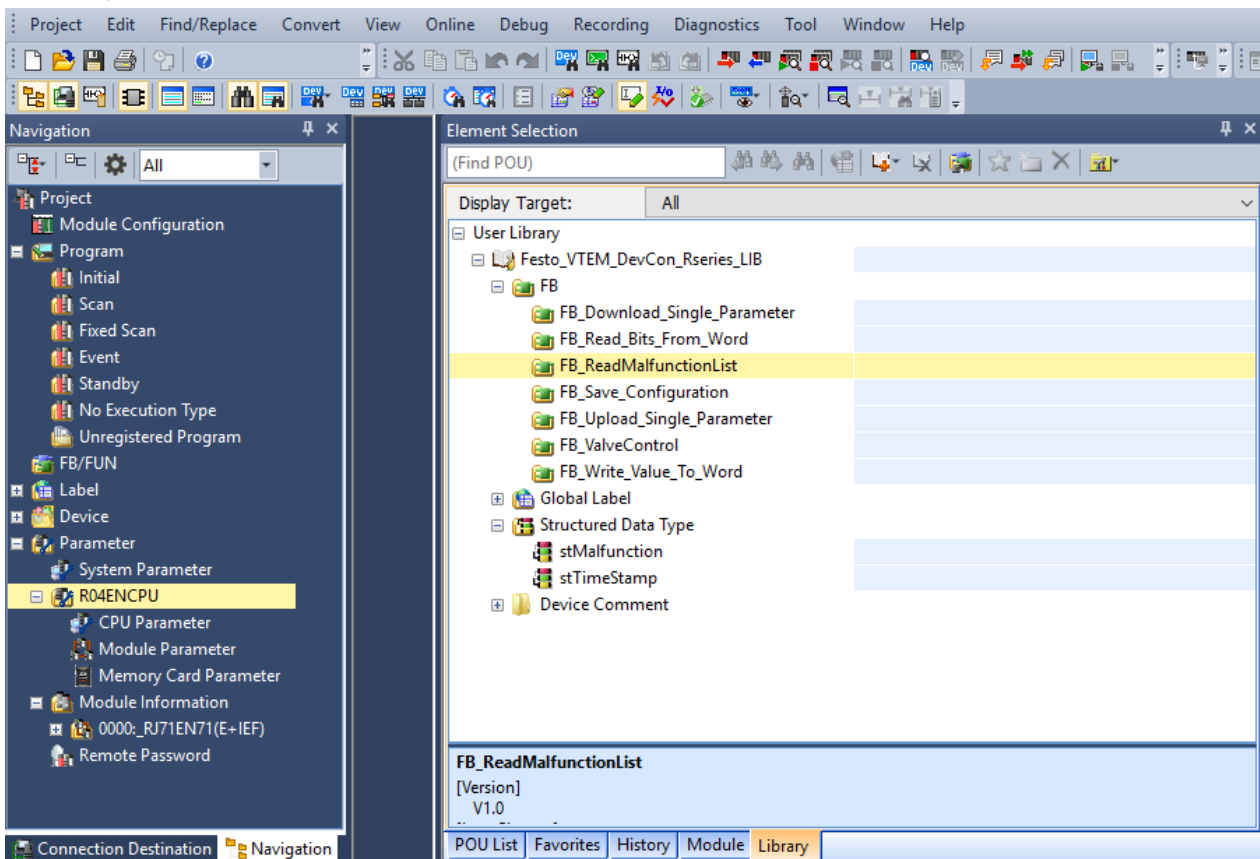
Click **“OK”** button in import configuration window.

Step – 3



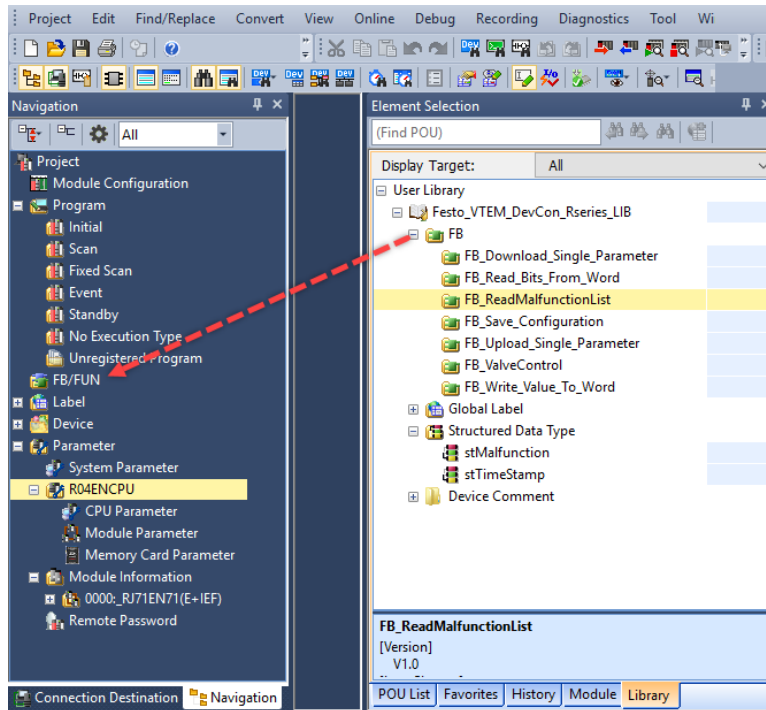
1. Select **“Festo_VTEM_DevCon_Rseries_LIB.usl”**.
2. Click **“Open”**

After successful import of Festo_VTEM_DevCon_Rseries_LIB.usl will be displayed in User Library folder as shown in below picture.

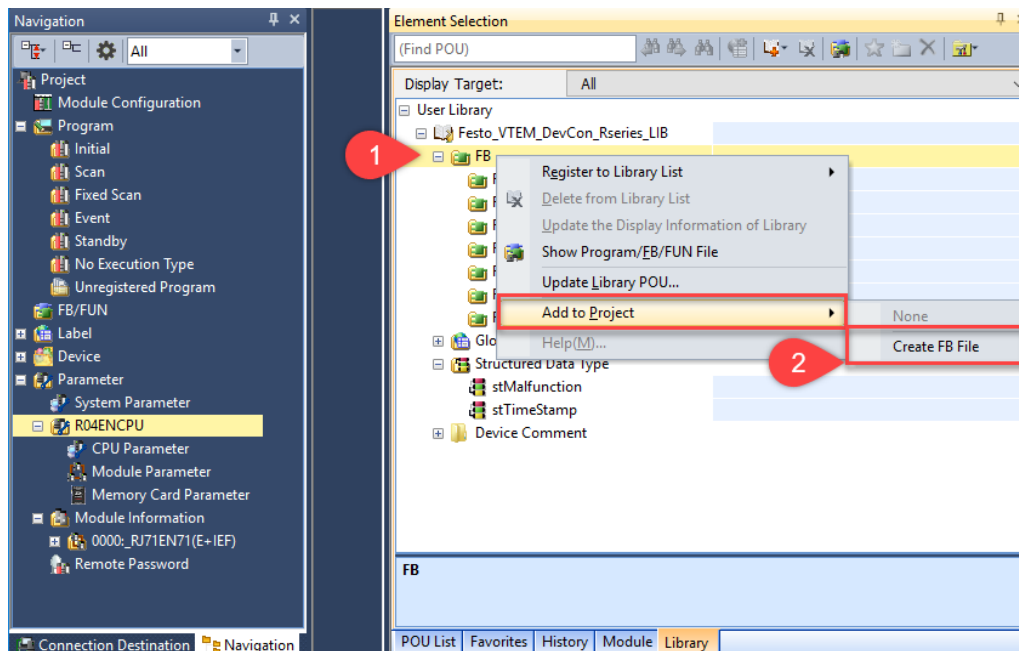


**Note**

- All user defined libraries will be under User Library Folder.

Step – 5 Add function block to Program.**Method 1:-**

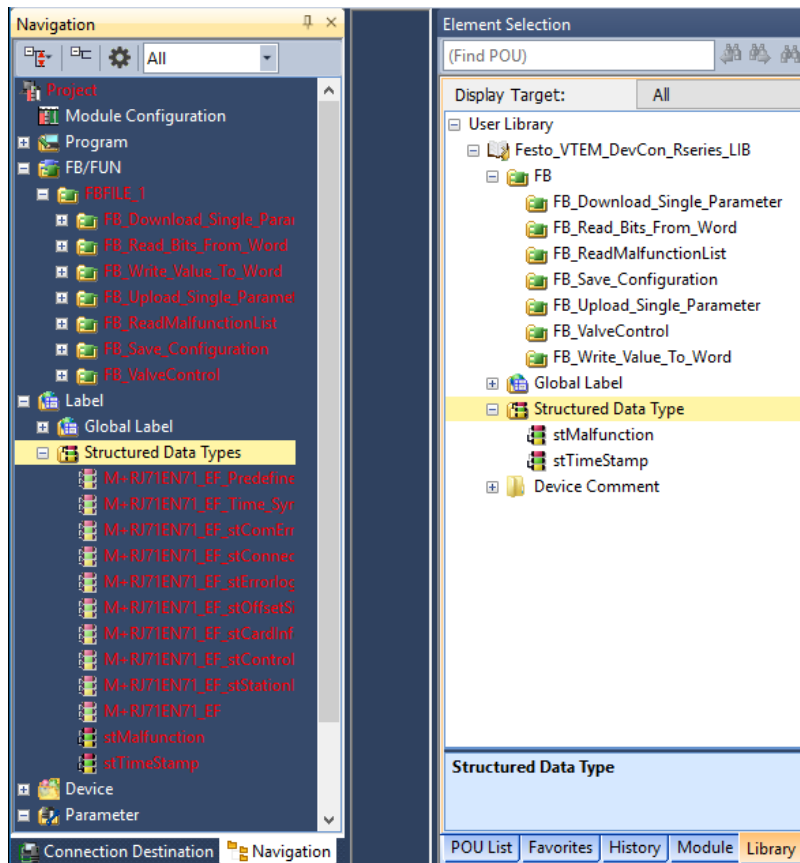
Drag and drop the **All FB's** from User Library folder to the **FB/FUN** as shown in below image.

Method 2:-

1. Right Click on **FB** under [Festo_VTEM_DevCon_Rseries_LIB] as shown in below image .
2. Go to **Add to Project** => Select **"Create FB File"**.

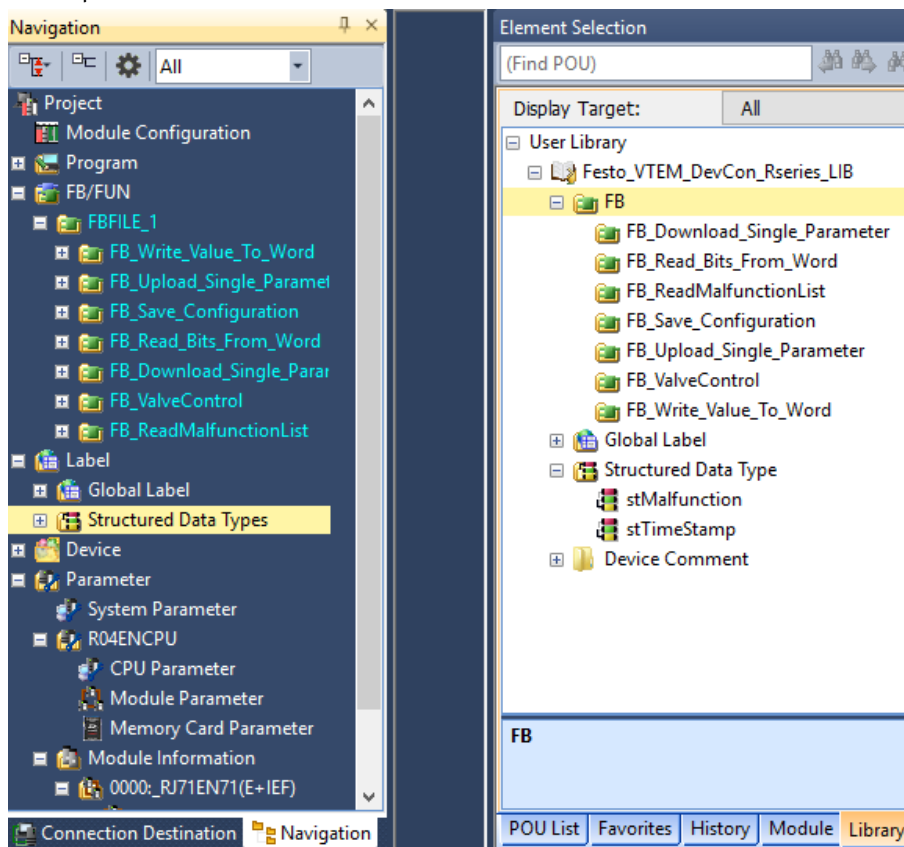
After successful addition of Festo_VTEM_DevCon_Rseries_LIB will be displayed in FB/FUN folder as shown in below picture.

Starting a Project



1. Click on the “**Rebuild All**” icon in the top menu bar.
2. Click “**Yes**” to rebuild the project.

After successful install of Festo_VTEM_DevCon_Rseries_Lib will be displayed in User Library folder as shown in below picture.



6 Elements in Festo_VTEM_DevCon_Rseries_LIB

The function blocks in “Festo_VTEM_DevCon_Rseries_Lib” library are described in detail in the following chapters.

Function Blocks

1. **FB_ValveControl** block is used to Control the Single valve in the motion terminal.
2. **FB_Upload_Single_Parameter** block is used to read the motion terminal valve parameter.
3. **FB_Download_Single_Parameter** block is used to write the motion terminal valve parameter.
4. **FB_Save_Configuration** block is used to save the motion terminal valve parameter.
5. **FB_ReadMalfunctionList** block is used to read the malfunction list of the VTEM motion terminal.
6. **FB_Read_Bits_From_Word** block is used to read the bits data from the word.
7. **FB_Write_Value_To_Word** block is used to write the value to particular bits of the word.

Structured Data type files

1. stMalfunction
2. stTimeStamp

The above Structured data types are used in FB_ReadMalfunctionList block.

6.1 General information on Motion Terminal VTEM

For commissioning and handling of the Motion Terminal, a safe understanding of the function of the product is absolutely essential.

The following additional descriptions are required for a complete understanding:

6.1.1 Documentation for Motion Terminal VTEM

Documentation	Contents
Quick Start Guide	Examples of the description of the WebConfig interface
“Instructions for using Motion Terminal VTEM” (VTEM)	Instructions on safe use and important notes on handling, installation, commissioning and maintenance of the Motion Terminal VTEM.
“Description of Motion Terminal VTEM, function and parameterisation” (VTEM-BES-...)	Detailed description of the Motion Terminal VTEM as well as its function and parameterisation
“Motion Terminal VTEM description, Motion App ...” (VTEM-MA-... / GAMM-A...-...)	Detailed description of the Motion Apps for the Festo Motion Terminal VTEM
“CPX system description” (CPX-SYS-...)	Information on the complete system of the CPX terminal



All available documents for the product → https://www.festo.com/net/en-in_in/SupportPortal/default.aspx?cat=5675&q=8047502&tab=3&s=t#result

Always observe the information and safety instructions in the documentation!

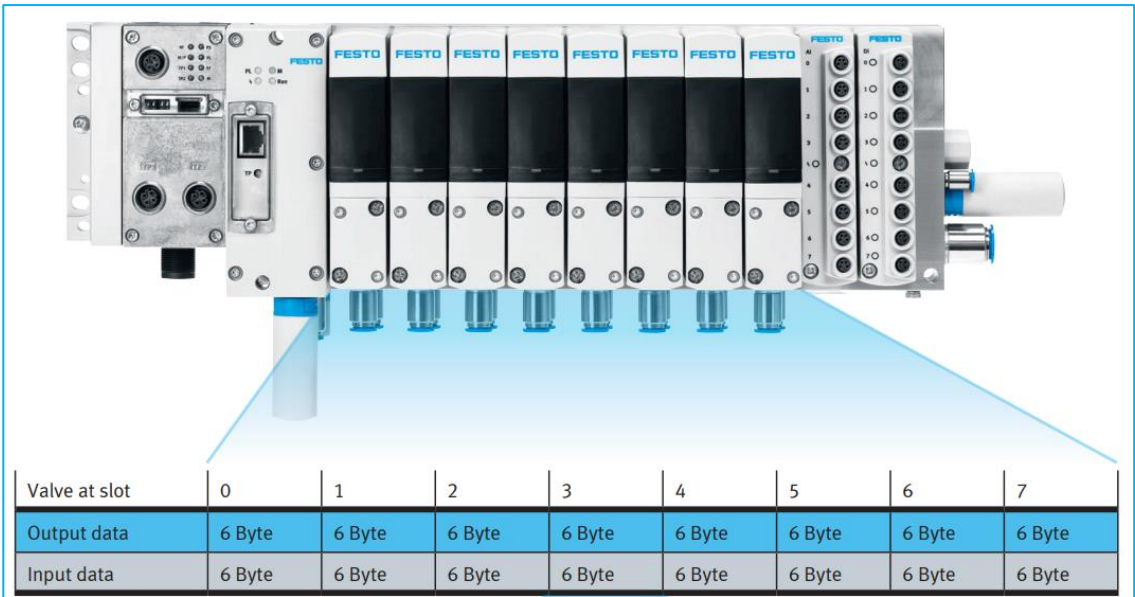


Note

- Motion Terminal VTEM description quick reference manual available with this application note folder.

6.2 Communication protocol of the Motion Terminal VTEM

The communication between the higher-order controller (PLC) and the Motion Terminal controller is based on input and output data of in each case 8 × 6 bytes from the CPX terminal. Each of the maximum 8 valve slots on a Motion Terminal is assigned 6 bytes of input data (Process Data Input, PDI) and 6 bytes of output data (Process Data Output, PDO), regardless of the actual number of valves present.



i The exact structure of the input and output data is described in detail in “Motion Terminal VTEM function and parameterization”.
The information in this description is assumed as being known.

6.2.1 Motion Terminal VTEM Function and Parameterization

Here the input and output Controlling data byte wise Details given bellow.

Output Data

Output data, valve at slot 2				Output data, valve at slot 3						Output data, valve at slot 4					
Byte 3	Byte 2	Byte 1	Byte 0	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Byte 5	Byte 4	Byte 3	Byte 2		
Setpoint Value 1		Command		Setpoint Value 2		Setpoint Value 1		Command		Setpoint Value 2		Setpoint Value 1			
Byte 1								Byte 0							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
App Option								App Control		Valve Mode					

The operating mode of the valve is specified via the Valve Mode command:

Valve Mode	Meaning	IF	THEN
0	reserved	–	–
1 ... 59	Start / run Motion App 1 ... 59	– Required parametrisation has been completed. – Valve is inactive and configurable (Actual Valve Mode = 61, Valve State = 1 (configurable)). – Setpoint / control values are valid.	Requested Motion App is started
		– Valve is inactive and configurable (Actual Valve Mode = 61, Valve State = 1 (configurable)). – Setpoint / control values are invalid.	See „Invalid output data“ on page 17
		– Another Motion App or the Transfer Mode is still running	Running Motion App keeps running, visible via the actual Valve Mode
		– Startup process of the device has been completed, at least one inactive error but no active error is present (Valve Mode = 61, Valve State = 0 (not ready))	Motion App can not be started because of an inactive error that needs to be acknowledged first (send Valve Mode „62“)
		– Startup process of the device has been completed, at least one active error is present – (Valve Mode = 61, – Valve State = 3 (failure))	Motion App cannot be started because of an active error that needs to be fixed first. After the error has been fixed, Valve State should change to 0 (not ready).
60	Start Teach-in run	see Valve Mode 1 ... 59	
61	End running Motion App / Teach-in run	– Motion App / Teach-in run is running	Motion App is stopped (Current Valve Mode = 61, Valve State = 1 (configurable))
		– Transfer Mode is running	Transfer Mode keeps running
62	Acknowledge inactive errors	– Valve State = 0 (not ready) because of an inactive error	Valve State changes to „configurable“, malfunction list is cleaned
63	Transfer Mode	– Valve is inactive (Actual Valve Mode = 61)	Transfer Mode command is executed

Input Data

Input data, valve at slot 2				Input data, valve at slot 3						Input data, valve at slot 4			
Byte 3	Byte 2	Byte 1	Byte 0	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Byte 5	Byte 4	Byte 3	Byte 2
Actual Value 1		Status		Actual Value 2		Actual Value 1		Status		Actual Value 2		Actual Value 1	

Byte 1								Byte 0							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
App State								Valve State		Actual Valve Mode					

The current operating mode and state of the valve can be seen in the Actual Valve Mode and Valve State section of the input data:

Actual Valve Mode	Meaning
0	reserved
1 ... 59	Running Motion App 1 ... 59
60	Executing Teach-in run
61	Valve is inactive or performs self-calibration
63	Transfer Mode is active

Valve State	Meaning
0	not ready Startup process for Motion Terminal is not completed or an error that has been detected and eliminated is yet to be acknowledged.
1	configurable The valve is inactive. A Motion App can be run or a switch to transfer mode can be made.
2	running A Motion App is currently being executed.
3	failure An error has been detected but not yet eliminated. Starting / Running a Motion App is not possible.

6.3 Connecting I/O Data to the Function Blocks

In the program, which use the function blocks of this library, the 6 bytes of input data and 6 bytes of output data must be maintained per valve as ARRAY [0..2] OF INT.

It is also recommended to create a separate instance of the function blocks used for each valve.



The specified start address for the variables of the input and output data must match the valve with which communication is to take place (→ [Communication protocol of the Motion Terminal VTEM](#)).

Modules of the CPX terminal must be taken into account.

The examples below refer to a Motion Terminal with a Mitsubishi Modbus Simple CPU communication as master in the CPX terminal. Here data manipulation in Mitsubishi PLCs are either Bit device or Word device whereas Festo VTEM Motion Terminals handles in Byte device. Hence Mitsubishi PLC automatically combines 2 bytes into word device in Word format (valve at slot 0: input data word 4 ... 6, output data word 0...2).



Note

- The VTEM Motion Terminal used to this sample code consists CPX-FB36 module which consume first 4 data word (D1000 ... D1003). Refer the below picture.
- The VTEM Motion Terminal used to this sample code consists 4 channel Analog Input module which consume first 4 data word (D1004 ... D1007), VTEM input data word starts from D1008 and the output starts from D2002. Refer the below picture.

Device Name	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	Current Value	
D1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Data from CPX-Fb36 module
D1001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D1002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D1003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D1004	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Data from Analog module
D1005	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D1006	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D1007	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D1008	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	61	Data from Valve 0
D1009	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	75	
D1010	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	143	
D1011	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	61	
D1012	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	234	Data from Valve 1
D1013	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	281	
D1014	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	61	
D1015	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	6	
D1016	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	1	157	Data from Valve 2
D1017	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	61	
D1018	0	0	0	0	0	0	0	1	0	1	1	0	1	0	1	1	363	
D1019	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	6	

Device Name	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	Current Value	
D2000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Data from CPX-Fb36 module
D2001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D2002	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	61	
D2003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D2004	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Data to Valve 0
D2005	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	69	
D2006	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1000	
D2007	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1000	
D2008	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Data to Valve 1
D2009	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D2010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D2011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D2012	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Data to Valve 2
D2013	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
D2014	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The input data and output data of a valve are transferred to the function blocks as ARRAY [0..2] OF WORD.

7 Festo_VTEM_DevCon_Rseries_LIB's Function Blocks

The function blocks in Festo_VTEM_DevCon_Rseries_LIB library are described in detail in the following chapters.

7.1 FB_ValveControl – Control of a Valve by Operating a Motion App

Function block FB_ValveControl serves to operate and control the Motion Apps. For this the function block interprets the values at the inputs and generates the in-output data that is sent to the Motion Terminal.

In parallel, the function block interprets the input data of a valve received from the Motion Terminal, classifies the information in it and issues it to the appropriate outputs.

This means that the status of the function block outputs takes place at least one cycle after the change of the input values.

FB_ValveControl	
awBusDataFromVTEM	awBusDataToVTEM
xEnable	wActualValveMode
wSetValveMode	wActualValveState
wSetAppControl	wActualAppState
wSetAppOption	iActualValue1
iSetpointValue1	iActualValue2
iSetpointValue2	iResponseToValveModeSet
xManualAcknowledge	iErrorCode

7.1.1 Inputs and Outputs

The following table contains the variables of the outputs and inputs of the function block.

Argument Name	Argument Type	Data Type / Unit Type	Description
awBusDataFromVTEM	INPUT	Word [Unsigned]/Bit String [16-bit](0..2)	The 6 Bytes of input data received from the Motion Terminal must be applied to this input.
awBusDataToVTEM	OUTPUT	Word [Unsigned]/Bit String [16-bit](0..2)	The 6 Bytes of output data that must be sent to the Motion Terminal are available at this output.
xEnable	INPUT	BOOL	Input must be TRUE to be able to use the function block. At a change to FALSE, the function block stops any operation running on the valve (Valve Mode = “valve inactive”). The output iResponseToValveModeSet shows the value 1051 after the function block was stopped successfully.
wSetValveMode	INPUT	Word [Unsigned]/Bit String [16-bit]	Input for setting the valve mode. Permissible values are: 1 ... 59 (ID Motion App) 60 (Teach-in run) 61 (End Motion App) 62 (Acknowledge inactive error), alternatively use input “xManualAcknowledge”
wSetAppControl	INPUT	Word [Unsigned]/Bit String [16-bit]	Input for setting the Motion App control. The significance is specific to the Motion App.

Argument Name	Argument Type	Data Type / Unit Type	Description
wSetAppOption	INPUT	Word [Unsigned]/Bit String [16-bit]	Input for setting the Motion App setting. The significance is specific to the Motion App.
iSetpointValue1	INPUT	Word [signed]	Input for setpoint value 1: The significance is specific to the Motion App.
iSetpointValue2	INPUT	Word [signed]	Input for setpoint value 2: The meaning is specific to the Motion App.
xManualAcknowledge	INPUT	BOOL	Inactive errors in valve status “not ready” can be acknowledged with a rising edge at this input (corresponds to specification “bySetValveMode := 62” --> Valve status becomes “configurable”).
wActualValveMode	OUTPUT	Word [Unsigned]/Bit String [16-bit]	Returned mode of the valve
wActualValveState	OUTPUT	Word [Unsigned]/Bit String [16-bit]	Returned status of the valve
wActualAppState	OUTPUT	Word [Unsigned]/Bit String [16-bit]	Returned status of the Motion App
iActualValue1	OUTPUT	Word [signed]	Returned actual value 1
iActualValue2	OUTPUT	Word [signed]	Returned actual value 2
iResponseToValveModeSet	OUTPUT	Word [signed]	Response to the sent ValveMode (Input bySetValveMode). Meaning: ➔ Response to Valve Mode Set
iErrorCode	OUTPUT	Word [signed]	In case of an error, the error code of the last error is at this output.

7.1.2 Response to Valve Mode Set

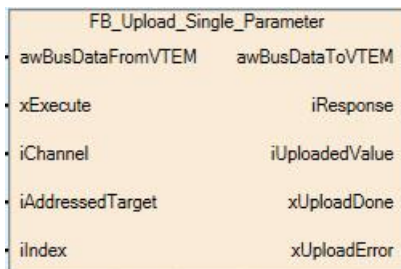
The argument “iResponseToValveModeSet” serves for the output of feedbacks to the ValveMode sent (Value at input “SetValveMode”). A part of the stored return values correspond to the value of output “AppState” in case of faulty output data (awBusDataToVTEM). There are also function-block-specific return values that cannot be imaged via the output “AppState” (e.g. via the status of the acknowledge function).

Response to Valve Mode Set No.	Response to Valve Mode Set Description	Response to Valve Mode Set No.	Response to Valve Mode Set Description
0	reserved	12	no_license_for_this_MA
1	invalid_SetValveMode	13	all_licenses_in_use
2	invalid_SetAppControl	14	invalid_license_file
3	invalid_SetAppOption	15	no_valve_detected
4	invalid_SetPointValue1	16	valve_self_calibration_running
5	invalid_SetPointValue2	1020	no_TransferMode_with_this_FB
6	invalid_configuration	1031	preparing_for_start
10	access_denied_WebConfig	1032	starting
11	access_denied_saving_parameterisation	1033	MA_running

Response to Valve Mode Set No.	Response to Valve Mode Set Description
1034	TransferMode_running
1035	stopping
1036	all_MAs_stopped
1037	acknowledge_needed
1038	failure
1040	preparing_for_acknowledge
1041	acknowledging
1042	all_errors_acknowledged
1043	failure_still_active
1050	no_communication
1051	FB_not_enabled

7.2 FB_Upload_Single_Parameter – Reading Single Value from the Motion Terminal

The function block transmits a value via the transfer mode from the Motion Terminal to the PLC.



7.2.1 Inputs and Outputs

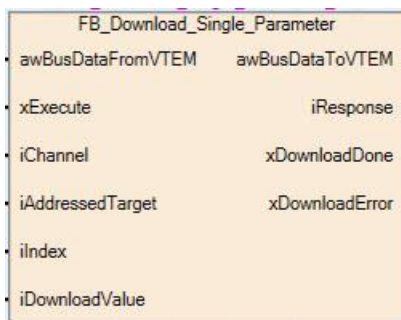
The following table contains the variables of the outputs and inputs of the function block.

Argument Name	Argument Type	Data Type / Unit Type	Description
awBusDataFromVTEM	INPUT	Word [Unsigned]/Bit String [16-bit](0..2)	The 6 Bytes of input data received from the Motion Terminal must be applied to this input.
awBusDataToVTEM	OUTPUT	Word [Unsigned]/Bit String [16-bit](0..2)	The 6 Bytes of output data that must be sent to the Motion Terminal are available at this output.
xExecute	INPUT	BOOL	Rising edge at the input starts the transmission. If the “TRUE” signal is removed before transmission is completed (xUploadDone = TRUE), the transfer mode is terminated.
iChannel	INPUT	Word [signed]	Transfer mode channel
iAddressedTarget	INPUT	Word [signed]	Channel-specific address of the target from which the value to be transmitted is to be read.
iIndex	INPUT	Word [signed]	Index for which the stored value is to be read out.
iResponse	OUTPUT	Word [signed]	Response to the status of the transmission. Meaning:→ Response to Transfer Mode

Argument Name	Argument Type	Data Type / Unit Type	Description
iUploadedValue	OUTPUT	Word [signed]	Read out value.
xUploadDone	OUTPUT	BOOL	Output becomes TRUE when the transfer could be concluded without an error.
xUploadError	OUTPUT	BOOL	Output becomes TRUE when errors occurred during the transfer.

7.3 FB_Download_Single_Parameter – Transfer Single Parameters to the Motion Terminal

The function block Instruction transmits a value via the transfer mode from the PLC to the Motion Terminal. It can only be used for writable values.



7.3.1 Inputs and Outputs

The following table contains the variables of the outputs and inputs of the function block.

Argument Name	Argument Type	Data Type / Unit Type	Description
awBusDataFromVTEM	INPUT	Word [Unsigned]/Bit String [16-bit](0..2)	The 6 Bytes of input data received from the Motion Terminal must be applied to this input.
awBusDataToVTEM	OUTPUT	Word [Unsigned]/Bit String [16-bit](0..2)	The 6 Bytes of output data that must be sent to the Motion Terminal are available at this output.
xExecute	INPUT	BOOL	Rising edge at the input starts the transmission. If the “TRUE” signal is removed before transmission is completed (xDownloadDone = TRUE), the transfer mode is terminated.
iChannel	INPUT	Word [signed]	Transfer mode channel
iAddressedTarget	INPUT	Word [signed]	Channel-specific address of the target into which the value to be transmitted is to be written.
iIndex	INPUT	Word [signed]	Index of the parameter that is to be set on the transmitted value.
iDownloadValue	INPUT	Word [signed]	Value on which the parameter is to be set.
iResponse	OUTPUT	Word [signed]	Response to the status of the transmission. Meaning:➔ Response to Transfer Mode
xDownloadDone	OUTPUT	BOOL	Output becomes TRUE when the transfer could be concluded without an error.
xDownloadError	OUTPUT	BOOL	Output becomes TRUE when errors occurred during the transfer.

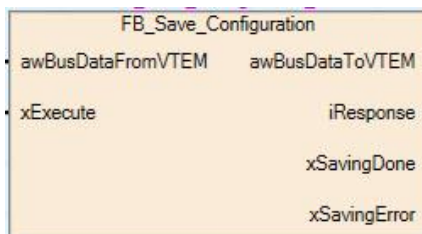
7.3.2 Response to Transfer Mode

The argument “iResponse” serves for the output of feedbacks to the current status of the transfer.

Response to Transfer Mode Set No.	Response to Transfer Mode Set Description	Response to Transfer Mode Set No.	Response to Transfer Mode Set Description
0	reserved	6	invalid_Combination
1	invalid_Channel	7	access_denied_DataInUse
2	invalid_TransferControl	8	no_data_empty_slot
3	invalid_AddressedTarget	9	invalid_ParameterSet
4	invalid_Index	10	access_denied_WebConfig
5	invalid_TransferValue	11	access_denied_saving
15	no_valve_detected	26	Valve_Mode_is_not_61_inactive
20	reading_transfermode	27	invalid_firmware_version
21	ready_for_new_transfer	28	communication_failed
22	preparing_transfer	29	invalid_module_number
23	transferring	30	resetting
24	transfer_successful	31	aborting
25	FB_needs_rising_edge_xExecute		

7.4 FB_Save_Configuration_Persistent – Save Parameterization Persistent

The function block serves for the simple call-up of function “Save configuration persistent”. The data for the valve currently being parameterised are saved permanently on the controller of the Motion Terminal.



7.4.1 Inputs and Outputs

The following table contains the variables of the outputs and inputs of the function block.

Argument Name	Argument Type	Data Type / Unit Type	Description
awBusDataFromVTEM	INPUT	Word [Unsigned]/Bit String [16-bit](0..2)	The 6 Bytes of input data received from the Motion Terminal must be applied to this input.
awBusDataToVTEM	OUTPUT	Word [Unsigned]/Bit String [16-bit](0..2)	The 6 Bytes of output data that must be sent to the Motion Terminal are available at this output.
xExecute	INPUT	BOOL	Rising edge at the input starts the transmission. If the “TRUE” signal is removed before transmission is completed (xSavingDone = TRUE), the transfer mode is terminated.
iResponse	OUTPUT	Word [signed]	Response to the status of saving process. Meaning: ➔ Response to Save Persistent
xSavingDone	OUTPUT	BOOL	The output becomes “TRUE” when the saving process is completed.

Argument Name	Argument Type	Data Type / Unit Type	Description
xSavingError	OUTPUT	BOOL	The output becomes “TRUE” when an error occurred during the saving process.

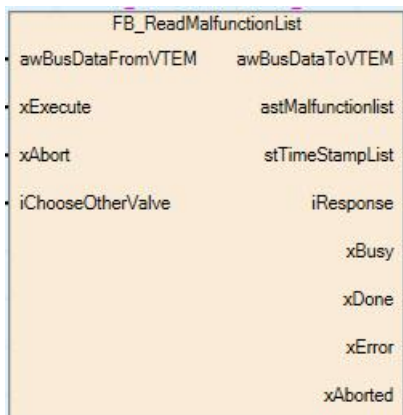
7.4.2 Response to Save Persistent

The argument “iResponse” serves for the output of feedbacks to the current status of the saving process.

Response to Save Persistent Set No.	Response to Save Persistent Description	Response to Save Persistent No.	Response to Save Persistent Description
0	reserved	11	access_denied_saving
1	saving_in_progress	15	no_valve_detected
2	saving_successful	20	preparing_to_save
3	saving_not_possible	21	ready_to_save
4	saving_failed	25	FB_needs_rising_edge_xExecute
10	access_denied_WebConfig		

7.5 FB_ReadMalfunctionList – Read Out Malfunction List

The function block “FB_ReadMalfunctionList” automatically reads the contents of the malfunction list of a valve and passes it to the output “astMalfunctionlist”.



The output “astMalfunctionlist” passes an array of 40 elements of the type “stMalfunction”. Each element thereby contains a complete entry of the malfunction list consisting of “Code”, “Subcode” and “Classification”. Via the input “uiChooseOtherValve” it can be defined from which valve the malfunction list should be read. This way the process data of any other slot can be used for this purpose.

7.5.1 Inputs and Outputs

The following table contains the variables of the outputs and inputs of the function block.

Argument Name	Argument Type	Data Type / Unit Type	Description
awBusDataFromVTEM	INPUT	Word [Unsigned]/Bit String [16-bit](0..2)	The 6 Bytes of input data received from the Motion Terminal must be applied to this input.
awBusDataToVTEM	OUTPUT	Word [Unsigned]/Bit String [16-bit](0..2)	The 6 Bytes of output data that must be sent to the Motion Terminal are available at this output.

Argument Name	Argument Type	Data Type / Unit Type	Description
xExecute	INPUT	BOOL	TRUE signal at the input starts the transmission. If the TRUE signal is removed, the transfer mode is terminated and the function block is reset as soon as the transfer process has been completed successfully (iResponse = "4" or "5") or with an error (xError = TRUE).
xAbsort	INPUT	BOOL	TRUE signal at the input terminates the process in the function block (iResponse = "7"). The transfer mode is terminated if necessary and the valve is set to the "inactive" state (valve mode = 61).
iChooseOtherValve	INPUT	Word [signed]	Select the valve whose malfunction list is to be read. 0: Read the malfunction list of the valve whose process data the function block uses. 100 ... 107: Read the malfunction list of the valve in slot 0 (100) ... 7 (107).
astMalfunctionlist	OUTPUT	stMalfunction[41]	Content of the malfunction list(→ The content of the array is complete as soon as xDone = TRUE. The output is reset as soon as xAbort = FALSE and xEnable = FALSE.
stTimeStampList	OUTPUT	stTimeStamp	The output shows the current time stamp of the VTEM motion terminal.
iResponse	OUTPUT	Word [signed]	Response to the status of the transmission. Meaning:→ Response to Read Malfunction List
xBusy	OUTPUT	BOOL	Output is TRUE as long as the function block executes an action.
xDone	OUTPUT	BOOL	Output is TRUE when the transfer could be concluded without an error (even if the malfunction list is empty).
xError	OUTPUT	BOOL	Output is TRUE if an error occurred during the transmission process.
xAborted	OUTPUT	BOOL	Output is TRUE if the transmission was aborted by the input "xAbsort". Output is reset if xExecute and xAbsort are FALSE at the same time.

7.5.2 Response to Read Malfunction List

The argument "iResponse" serves for the output of feedbacks to the current status of the Malfunction list.

Response to Read Malfunction List Set No.	Response to Read Malfunction List	Response to Read Malfunction List Set No.	Response to Read Malfunction List
0	idle	102	invalid_SetAppControl
1	setting_valve_to_inactive	103	invalid_SetAppOption
2	getting_malfunction_count	104	invalid_SetPointValue1
3	transferring	105	invalid_SetPointValue2
4	done	106	invalid_configuration
5	no_entries_in_malfunction_list	110	access_denied_WebConfig
6	error_while_transmitting_data_to_vtem	111	access_denied_saving_parameterisation
7	aborting	112	no_license_for_this_MA

Response to Read Malfunction List Set No.	Response to Read Malfunction List	Response to Read Malfunction List Set No.	Response to Read Malfunction List
8	resetting	113	all_licenses_in_use
10	FB_needs_rising_edge_xExecute	114	invalid_license_file
100	reserved	115	no_valve_detected
101	invalid_SetValveMode	116	valve_self_calibration_running
201	invalid_Channel	205	invalid_TransferValue
202	invalid_TransferControl	206	no_valve_detected_at_selected_slot
203	invalid_iChooseOtherValve	207	access_denied_DataInUse
204	invalid_Index	208	no_data_empty_slot

7.5.3 stMalfunction

The structure data contains all data required to Read Malfunction of VTEM Motion Terminal.

	Label Name	Data Type	English(Display Target)
1	iMalfunctionCode	Word [Signed]	Malfunction Code
2	iMalfunctionSubcode	Word [Signed]	Malfunction Subcode
3	iMalfunctionClassification	Word [Signed]	Malfunction Classification
4	iHours	Word [Signed]	Time stamp (hours) of the malfunction message
5	iMinutes	Word [Signed]	Time stamp (minutes) of the malfunction message
6	iSeconds	Word [Signed]	Time stamp (seconds) of the malfunction message
+	iMilliseconds	Word [Signed]	Time stamp (milliseconds) of the malfunction
8			

The above Structured data types are used in FB_ReadMalfunctionList block.

7.5.4 stTimeStamp

The structure data contains all data required to Read Time Stamp when Malfunction occurs in VTEM Motion Terminal.

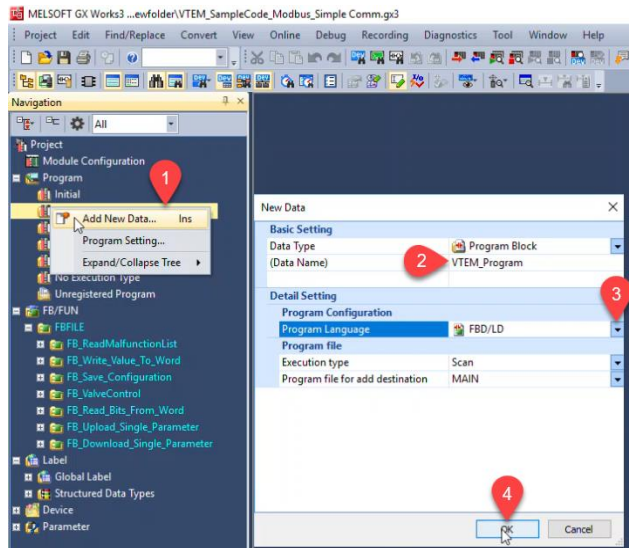
	Label Name	Data Type	English(Display Target)
1	iHours	Word [Signed]	Time stamp (hours) of the malfunction message
2	iMinutes	Word [Signed]	Time stamp (minutes) of the malfunction message
3	iSeconds	Word [Signed]	Time stamp (seconds) of the malfunction message
+	iMilliseconds	Word [Signed]	Time stamp (milliseconds) of the malfunction
5			

The above Structured data types are used in FB_ReadMalfunctionList block.

8 Configuration of VTEM Function blocks in Gx Works3

8.1 Configuring Program

This chapter describes about adding “Festo_VTEM_DevCon_Rseries_LIB” library in program & assigning input, output tags.

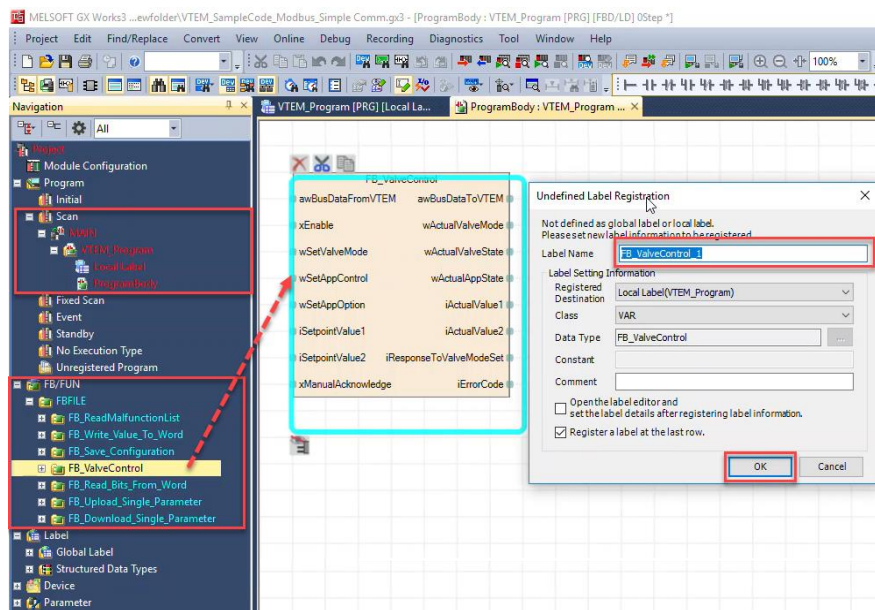


1. [Program] => [Scan]
 - a. Right-Click on “Scan” in the “Project” side bar.
 - b. Left-Click on “Add new Data.”
 - c. A new window “New Data” will be opened.
2. [Data Name] > [User Defined] or use the ones in the example image.
3. [Program Language] > [FBD/LD]
4. Confirm your selection with the button “OK”.

8.2 Adding FB in Program and Assigning Inputs and Outputs Tags

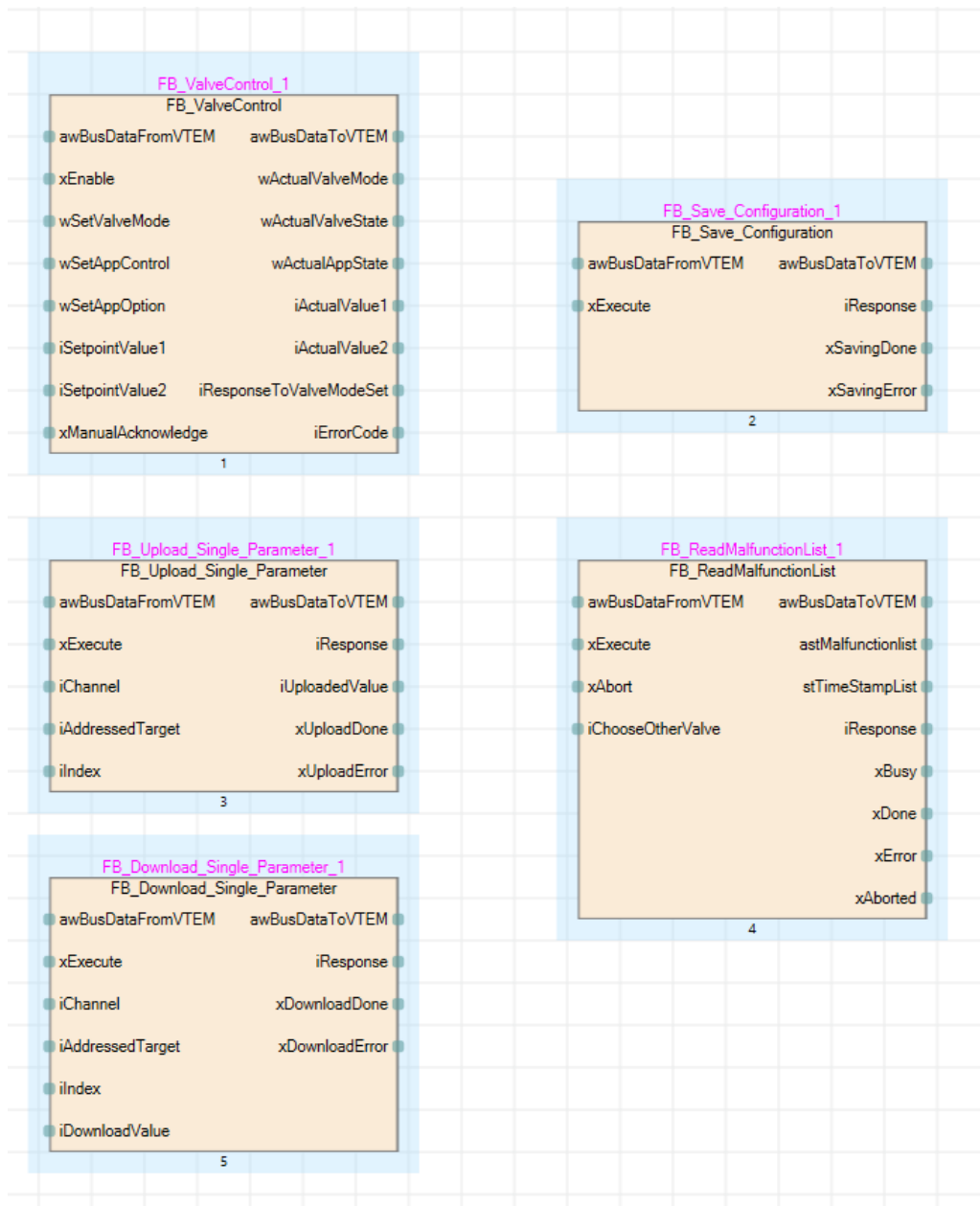
This chapter describes about adding FB in program & assigning input, output tags.

Step – 1



1. Insert the following code inside the implementation window. Drag and drop the FB_ValveControl, FB_Upload_Single_Parameter, FB_Download_Single_Parameter, FB_Save_Configuration, & FB_ReadMalfunctionList from FB/FUN to the empty ladder rung as shown in above image.
2. Confirm your selection with the button “OK”.

After successful addition of the FB_ValveControl, FB_Upload_Single_Parameter, FB_Download_Single_Parameter, FB_Save_Configuration, & FB_ReadMalfunctionList will be displayed in Function ladder rung (program folder) as shown in below picture.



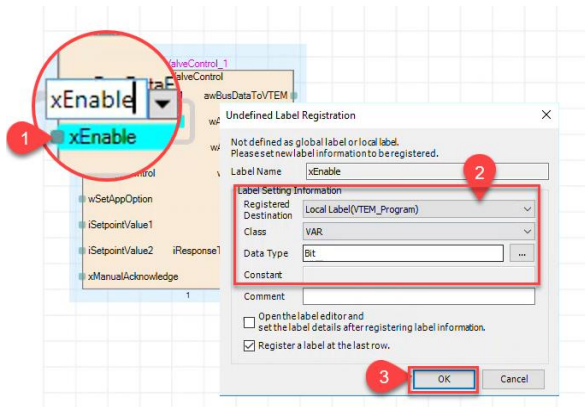
Step – 2



Note

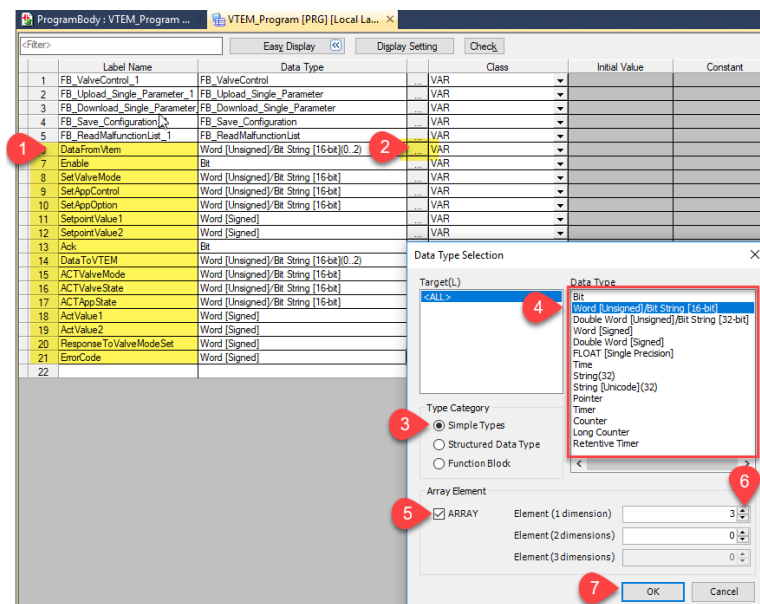
- User have to created correct tag with Data Type and Class which show in [[7.1: FB_ValveControl - Inputs & Outputs Table](#)], [[7.2: FB_Upload_Single_Parameter - Input & Outputs Table](#)], [[7.3: FB_Download_Single_Parameter - Inputs & Outputs Table](#)], [[7.4: FB_Save_Configuration - Inputs & Outputs Table](#)] [[7.5: FB_ReadMalfunctionList - Inputs & Outputs Table](#)].
- Insert the following code inside the declaration window. As described in **Step – 2**. Follow Method 1 or Method 2 given below.

Method - 1



1. Select the **“Variable”** from the **“FBD/LD”** menu buttons, as shown in below image.
1. Click /Select the interface tag. Enter the **“Tag Name”** for input **“xEnable”** and press **“Enter”** button to open **“Undefined Label Registration”** window.
2. Under Label Setting Information (This will be auto declared,)
 - a. Registered Destination => Local Label(C2M_Program)
 - b. Class => Var
 - c. Data Type => Bit (Refer [Chapter 7.1.1](#) Inputs and Outputs)
3. click **“OK”** button to declare the variable.
4. Repeat the previous step for rest of the interface tag for all function block.

Method-2



Double-Click on **“Local Label”** of the program **“VTEM_Program”** in the **“Program”** side bar. A new window **“VTEM_Program Local Label Setting”** will be opened as shown in above image.

1. Write under [Label Name] => [Used defined name] or [As given in [Chapter 7.1.1](#) Table 7.1: FB_ValveControl- Inputs & Outputs Table].
2. Click on the highlighted box, then Data Type pop-up window will open as shown in above image.
3. Select **Simple Type** in **Type Category**.
4. Select the **Data Type** for the Label. [Ref [Chapter 7.1.1](#) 7.1: FB_ValveControl - Inputs & Outputs Table].
5. In case if the Data is Array, then select Array.
6. Enter the dimension under Element (1 dimension).
7. Confirm your selection with the button **“OK”**.

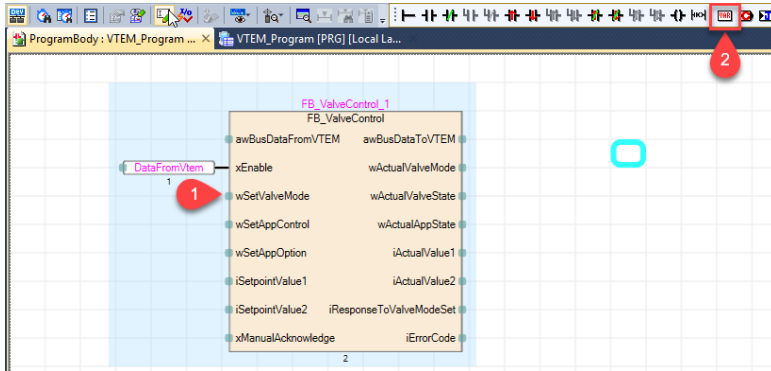


Note

- Repeat the **steps 2** follow either Method 1 or Method 2 the same procedure for rest of the Tag that's need for AOI's interface.
- Fill In and Out Tags of the Function Blocks.

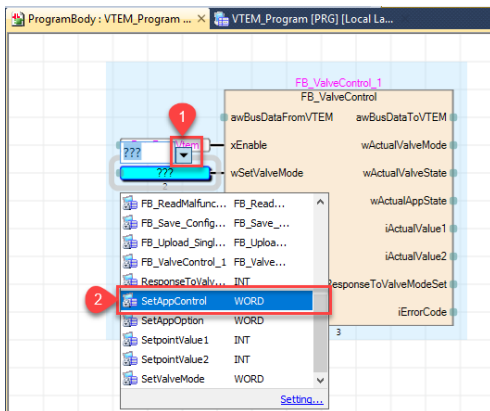
After Step – 2 follow the below steps to Map the correct Communication interface tag.

Step – 3 Variables to be filled in the function block.



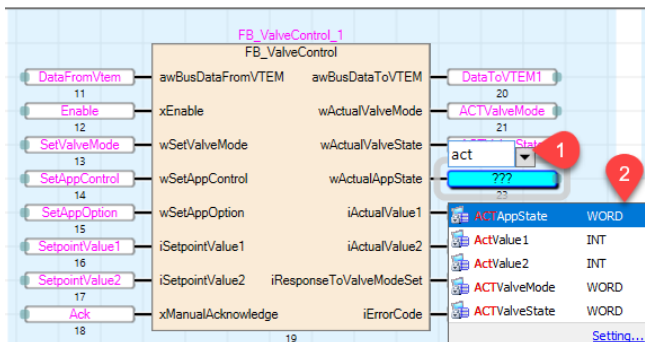
1. Click/Select the interface tag.
2. Select the **“Variable”** from the **“FBD/LD”** menu buttons, as shown in below image.
3. Repeat the previous step for rest of the interface tag on communication module’s function block.

Step – 4



1. Double click on the Variable, to add the tag.
2. Click on drop/down arrow button or the highted box, a selection window opens. As shown in below image.

Or

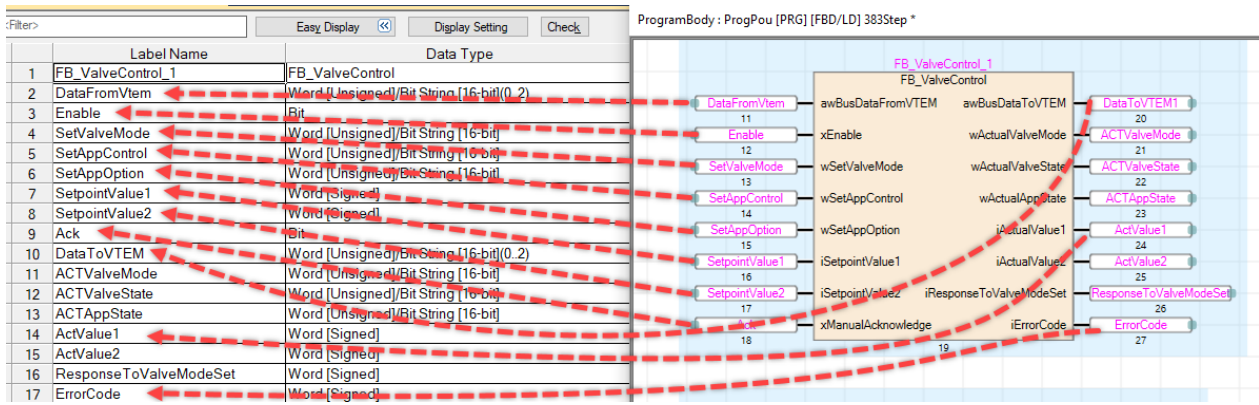


1. Double click on the Variable, to add the tag.
2. Write the correct tag name and left click mouse button or press enter.

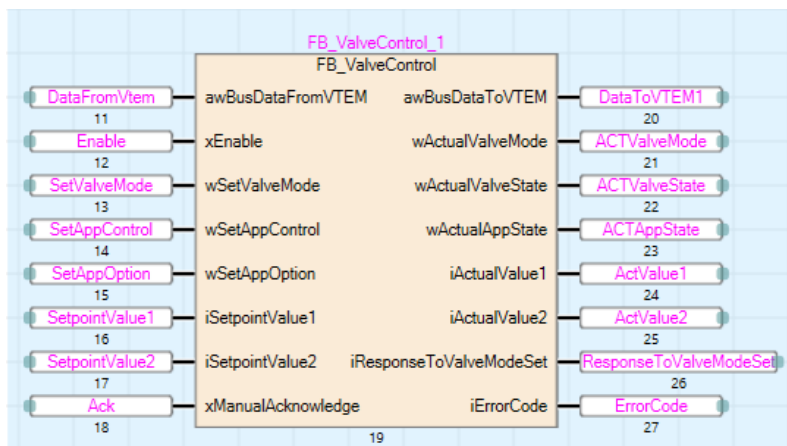


Note

User have to map the correct communication module interface tag to FB module which are created.



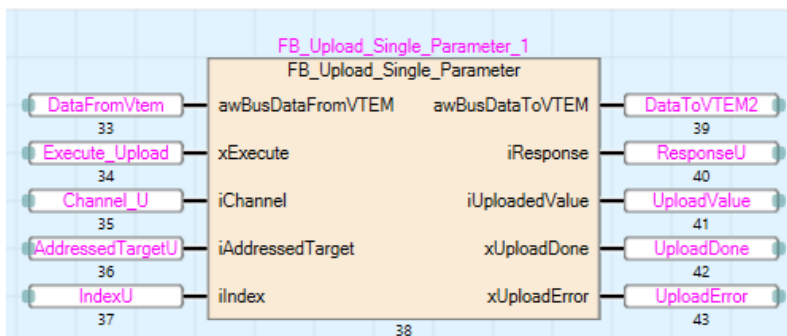
8.3 Add “FB_ValveControl” in PLC Program



Configure the “FB_ValveControl” FB interface tags as shown above. Refer [chapter 8.2 - Step – 2](#) to know how to create external tags and to map the tags.

8.4 Add “FB_Upload_Single_Parameter” in PLC Program

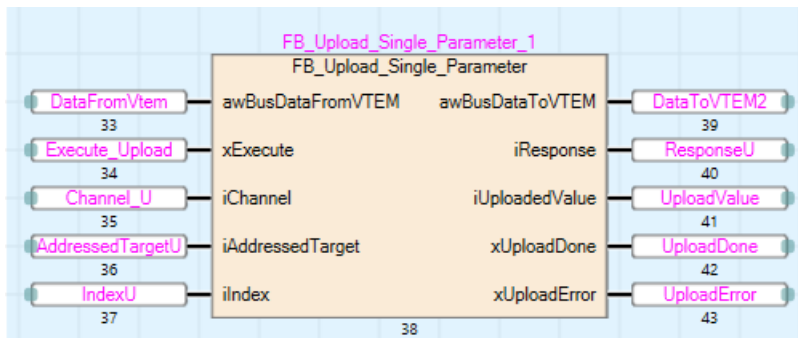
This chapter describes about adding “FB_Upload_Single_Parameter” FB in program & assigning input, output tags.



Configure the “FB_Upload_Single_Parameter” FB interface tags as shown above. Refer [chapter 8.2 - Step – 2](#) to know how to create external tags and to map the tags.

8.5 Add “FB_Download_Single_Parameter” in PLC Program

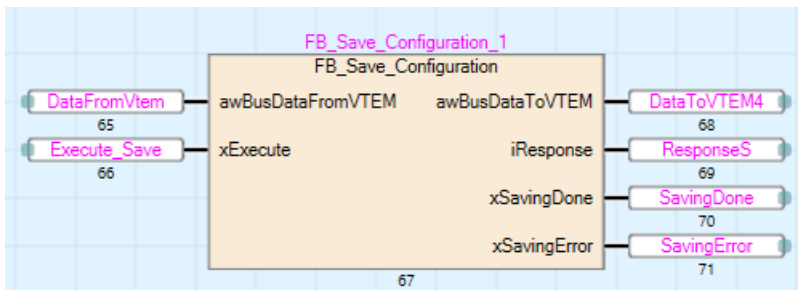
This chapter describes about adding “FB_Download_Single_Parameter” FB in program & assigning input, output tags.



Configure the “FB_Download_Single_Parameter” FB interface tags as shown above. Refer [chapter 8.2 - Step – 2](#) to know how to create external tags and to map the tags.

8.6 Add “FB_Save_Configuration” in PLC Program

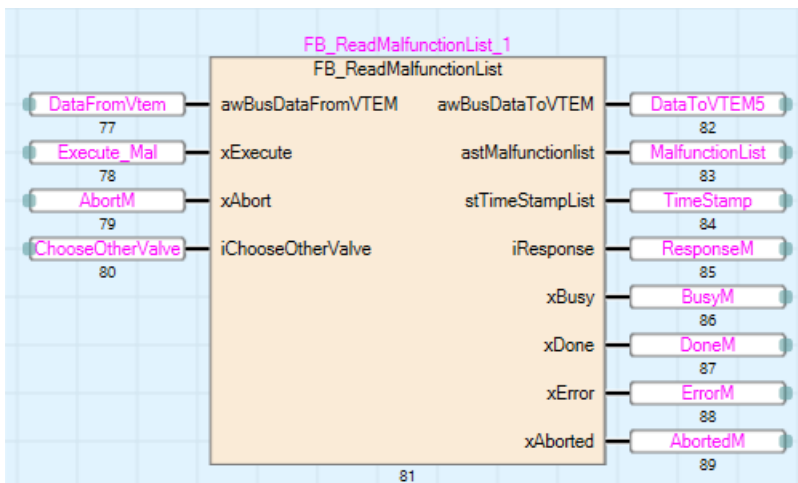
This chapter describes about adding “FB_Save_Configuration” FB in program & assigning input, output tags.



Configure the “FB_Save_Configuration” FB interface tags as shown above. Refer [chapter 8.2 - Step – 2](#) to know how to create external tags and to map the tags.

8.7 Add “FB_ReadMalfunctionList” in PLC Program

This chapter describes about adding “FB_ReadMalfunctionList” FB in program & assigning input, output tags.



Configure the “FB_ReadMalfunctionList” FB interface tags as shown above. Refer [chapter 8.2 - Step – 2](#) to know how to create external tags and to map the tags.



Note

Unsafe status of the system at parallel call-up of function blocks

- The simultaneous access of several function blocks to the output data of a valve (awBusDataToVTEM) can lead to the unforeseeable behaviour of the Motion Terminal, as well as the connected periphery.
- Ensure that never more than one function block instance can gain access to the output data of a valve at the same time (awBusDataToVTEM).



It is recommended to use a separate instance of the function blocks for each valve.



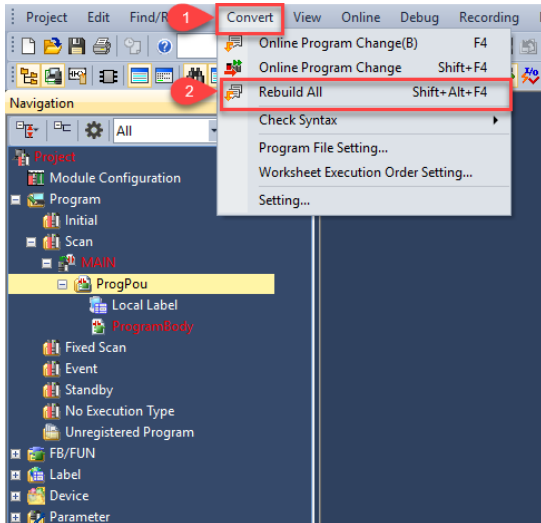
Note

- The Common pointers are used in all function block. So user can use the all five function block for one or two valves depends on the PLC common pointer capacity.
- If user wants to use all five blocks for 8 valves, then they have to go with higher end PLC.

9 PLC Configuration Transfer to Controller

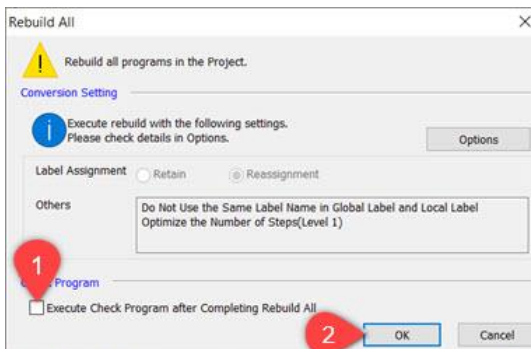
9.1 Downloading Mode

Step – 1



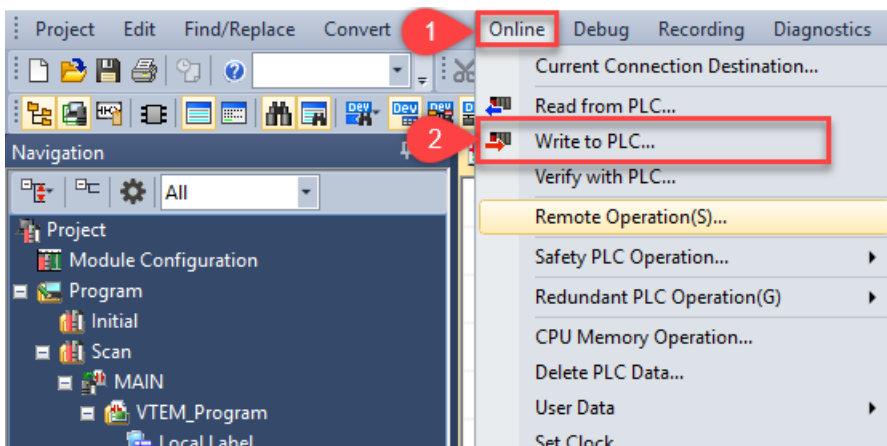
1. Select **“Convert”** option from menu bar.
2. Click **“Rebuild All”** option from compile menu.

Step – 2



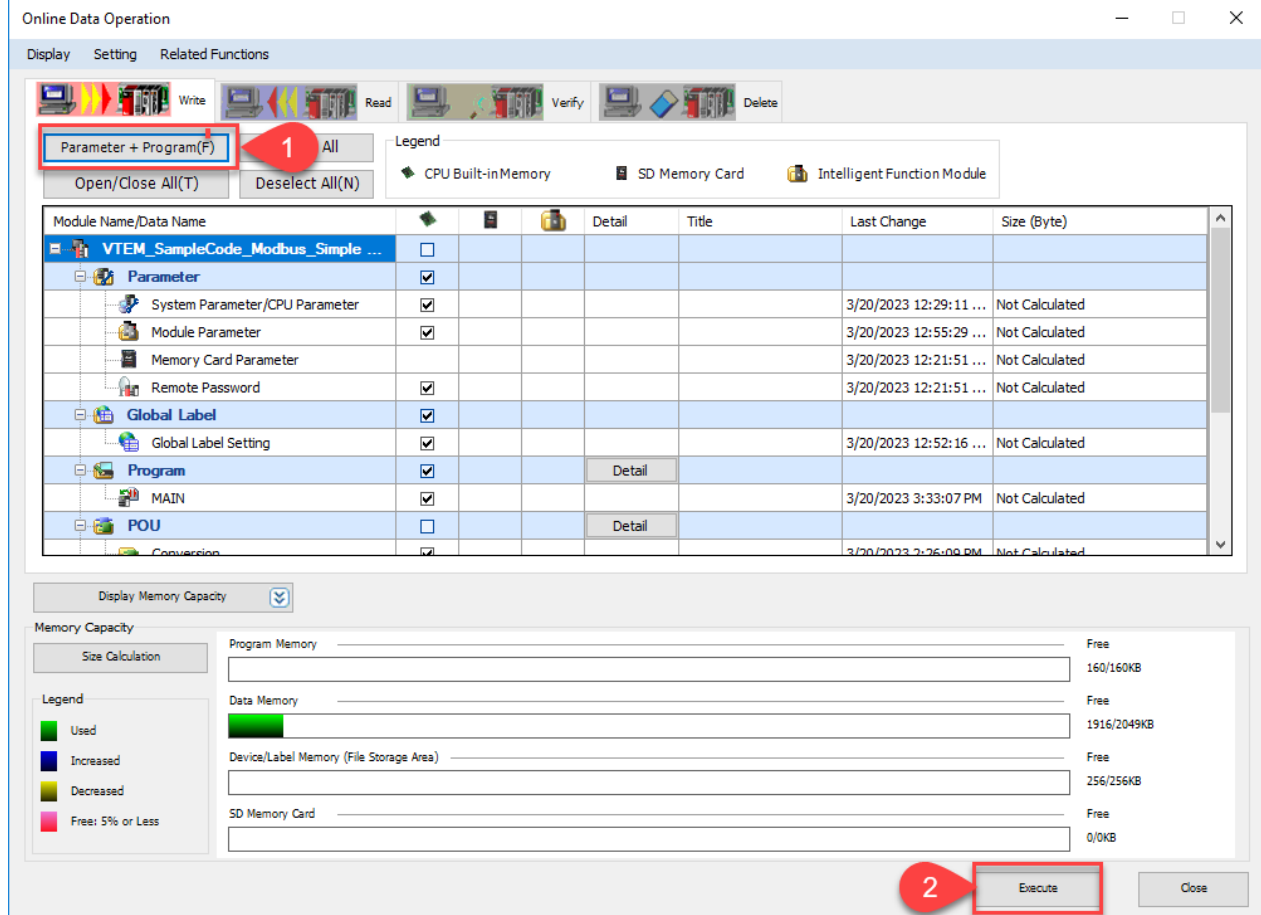
1. Check the box **“Execute Check Program after Completing Rebuild All”** option from **“Rebuild All”** window.
2. Click **“OK”** button.

Step – 3



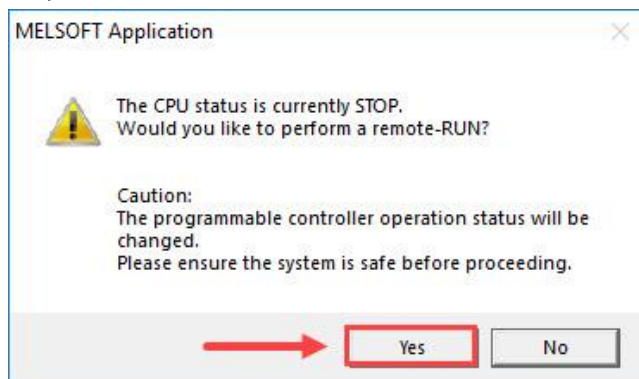
1. Go to **“Online”** option from menu bar.
2. Select **“Write to PLC...”** option from **“Online”** menu. Or user can also perform this process by clicking  this icon from toolbar.

Step – 4



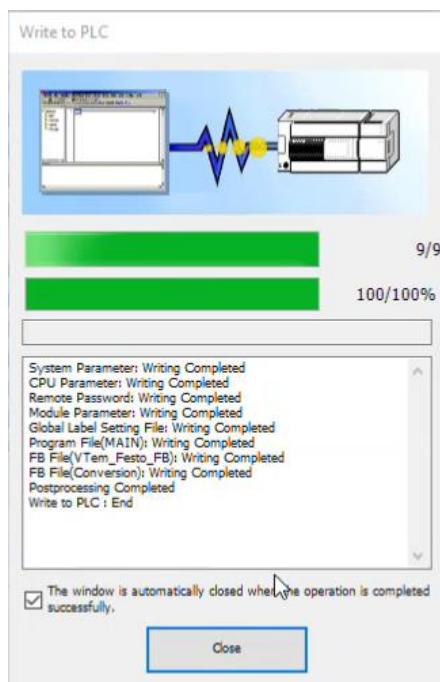
1. Click on **“Select All”** option from **“Online Data Operation”** window to download all program and parameters.
2. Click on **“Execute”** button.

Step – 5



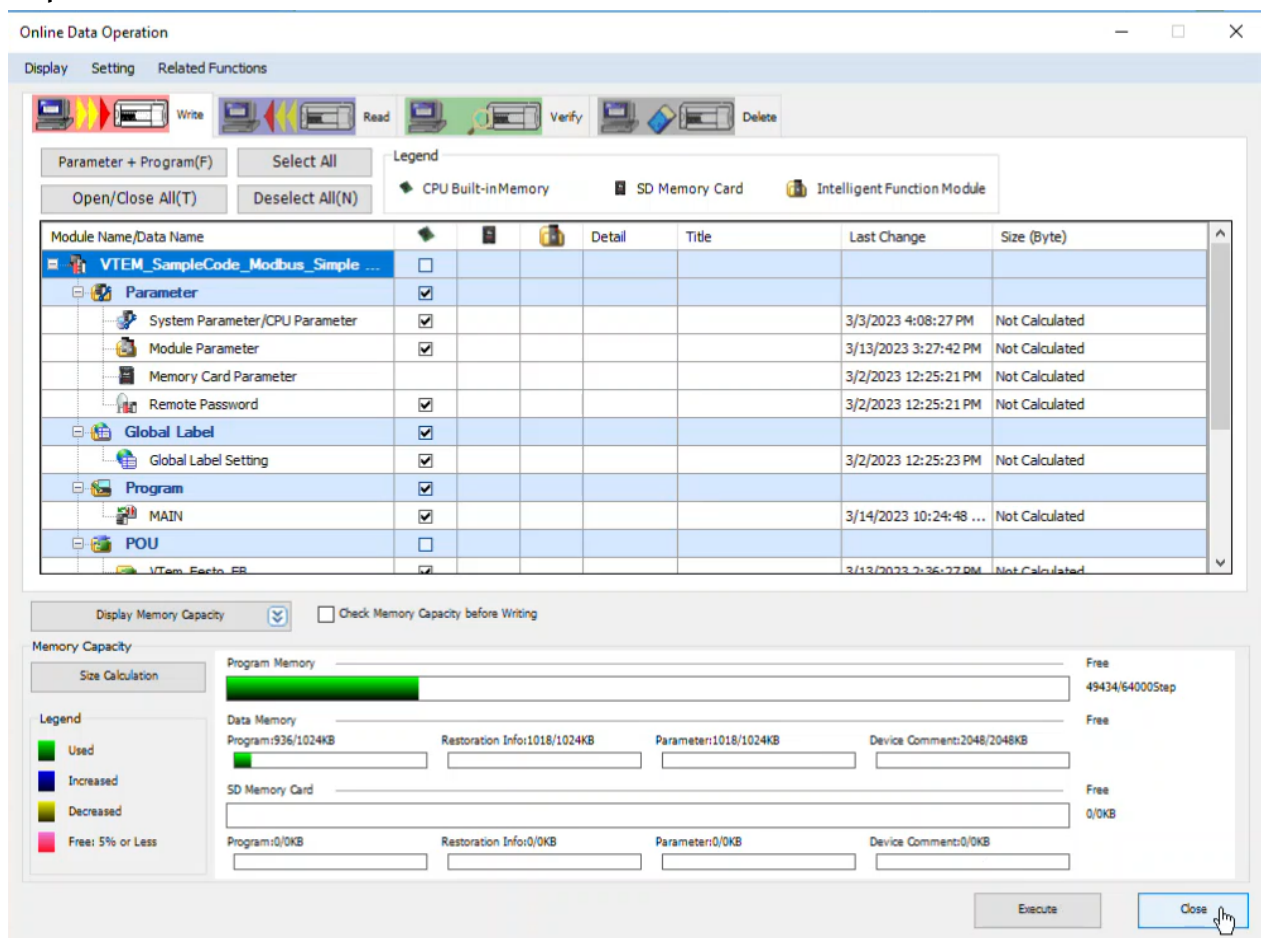
1. Click on **Yes** to Download the Project.

Step – 6



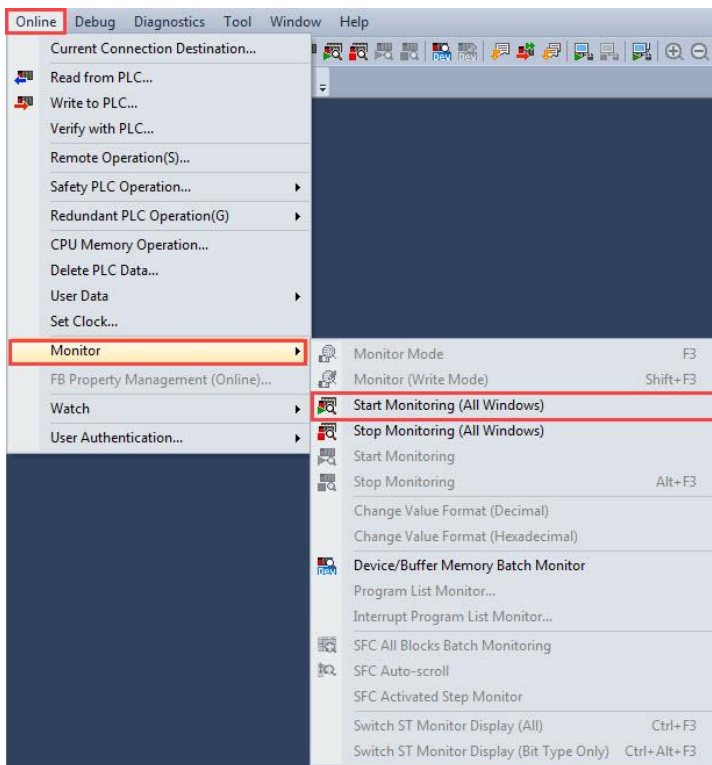
- Write to PLC in progress.

Step – 7



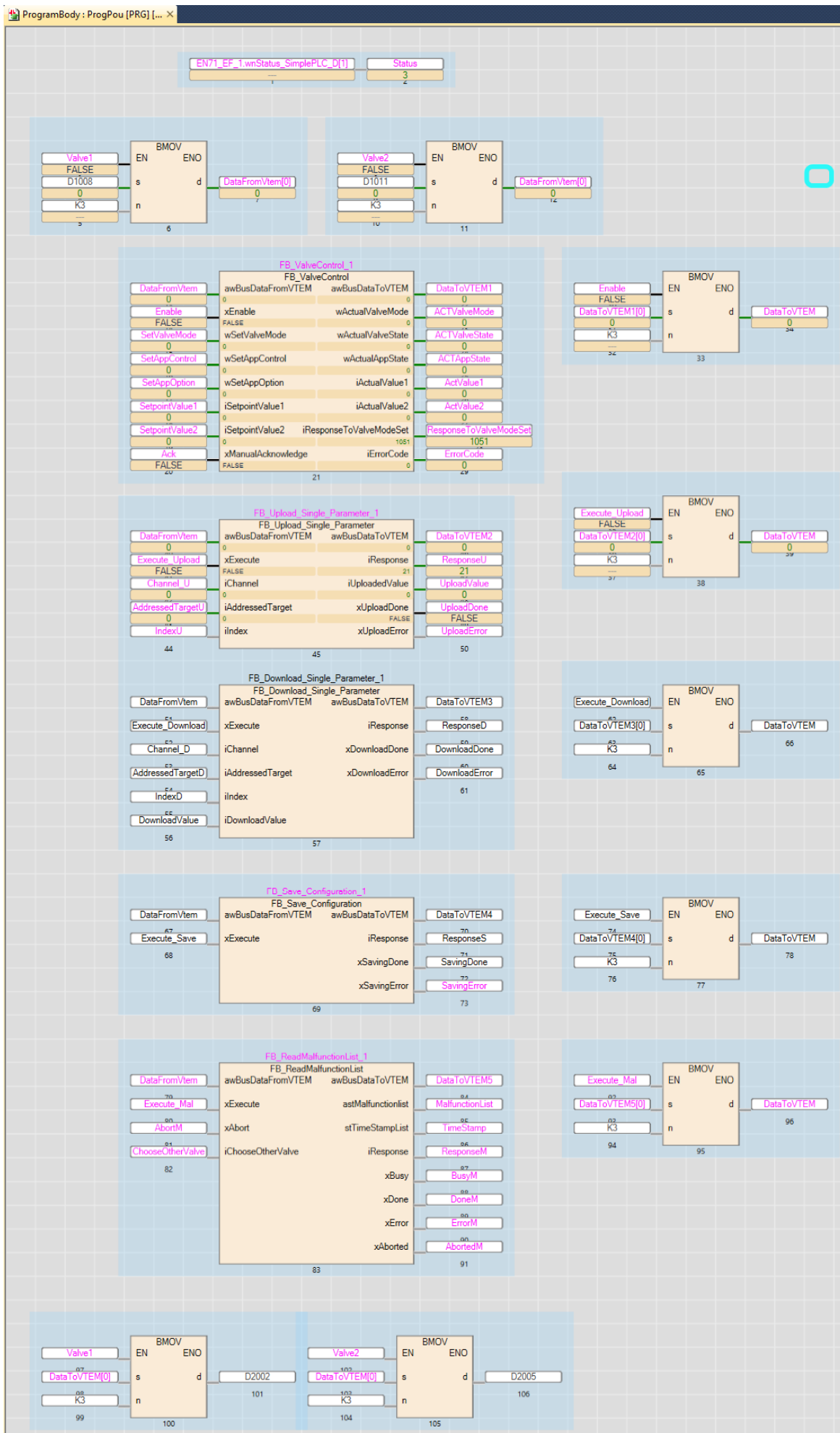
- “Write to PLC” window will close automatically and then click “Close” button to close the window.

9.2 Monitor Mode.



1. Go to **“Online”** option from menu bar.
2. Select **“Monitor”** option from **“Online”** menu.
3. Click **“Start Monitoring (All Windows)”** option from **“Monitor”** option. Or user can also perform this process by  clicking this icon from toolbar.

In monitoring mode “FB_ValveControl”, “FB_Upload_Single_Parameter”, “FB_Download_Single_Parameter”, “FB_Save_Configuration”, & “FB_ReadMalfunctionList”. Function Blocks is monitored as shown in below image.



9.2.1 Checking Connection:

Checking the simple CPU communication status

The simple CPU communication status can be checked with the buffer memory or diagnostic functions.

Checking with the buffer memory

The simple CPU communication status can be checked with the storage status of the corresponding setting number in the following buffer memory areas.

Item	Address		Remarks
Request to start communication at request	UnlG721896 to UnlG721899 UnlG1247300 to UnlG1247327		• Setting No.1: UnlG721896.0 ⋮ • Setting No.64: UnlG721899.F • Setting No.65: UnlG1247300.0 ⋮ • Setting No.512: UnlG1247327.F
Ready	UnlG721912 to UnlG721915 UnlG1247412 to UnlG1247439		• Setting No.1: UnlG721912.0 ⋮ • Setting No.64: UnlG721915.F • Setting No.65: UnlG1247412.0 ⋮ • Setting No.512: UnlG1247439.F
Simple CPU communication status	0H: Unset	UnlG721936 to UnlG721999 UnlG1247460 to UnlG1247907	• Setting No.1: UnlG721936 ⋮ • Setting No.64: UnlG721999 • Setting No.65: UnlG1247460 ⋮ • Setting No.512: UnlG1247907
	1H: Preparing		
	2H: Waiting for the request		
	3H: Communicating		
	4H: Communication stop		
	5H: Retry being executed		
	6H: Monitoring at error		
	AH: Communications impossible		
Simple CPU communication error code		UnlG722000 to UnlG722063 UnlG1247908 to UnlG1248355	• Setting No.1: UnlG722000 ⋮ • Setting No.64: UnlG722063 • Setting No.65: UnlG1247908 ⋮ • Setting No.512: UnlG1248355

Fig: Simple Communication Status Device No, taken from Simple CPU Communication Manual.

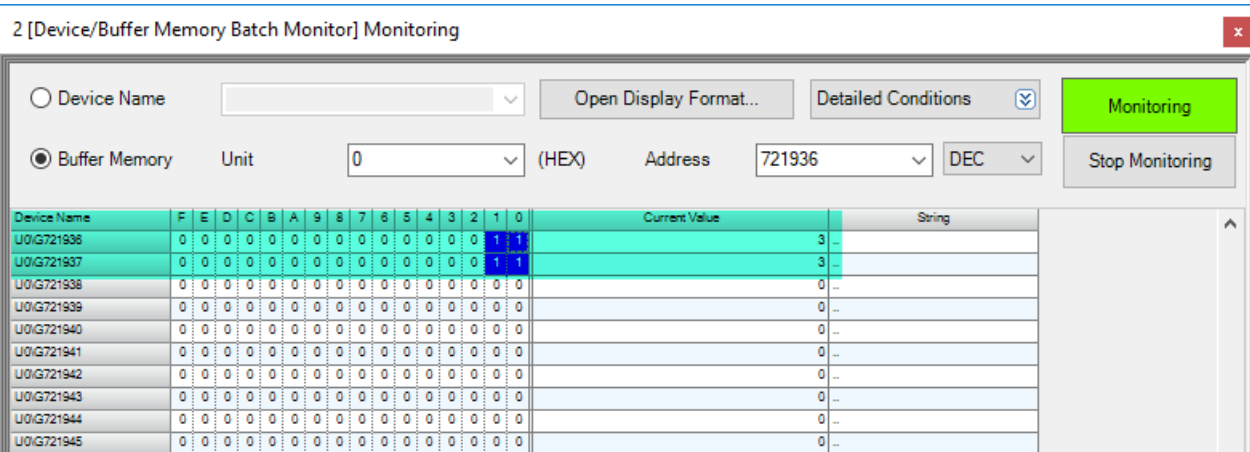
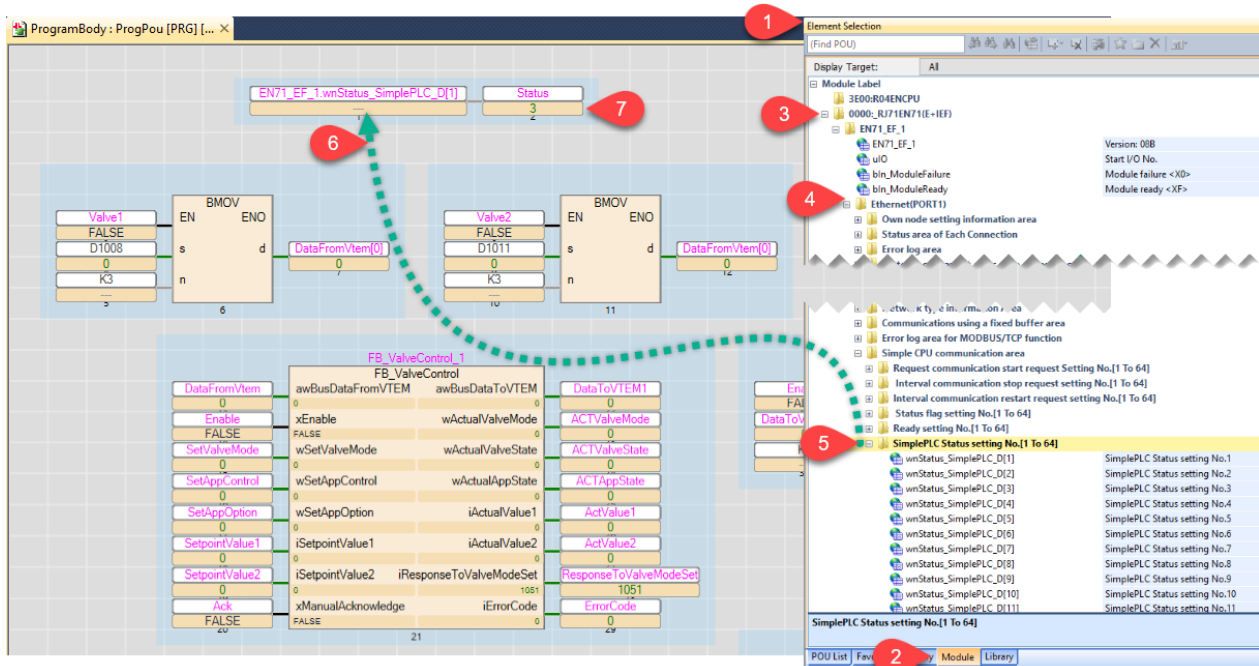


Fig: Device Memory
If Communication is ready then value will be 3, else value will be less than or greater than 3.



1. Under Element Selection
2. Select Module
3. Expand RJ71EN71(E+IEF)
4. Expand Ethernet (Port 1)
5. Expand Simple PLC Status setting No.[1 To 64]
Referring simple CPU Communication **Setting No.**, select Simple PLC communication status **(unStatus_SimplePLC[_])** from “Simple PLC communication for each setting No. requested” tree in **Module** page of **Element Selection**.
6. Drag & drop Simple PLC communication status **(unStatus_SimplePLC[_])**, and link it to Variable Local tag (Status – Word [Signed]).
7. It communication is established successfully then value will be displayed 3.



Note

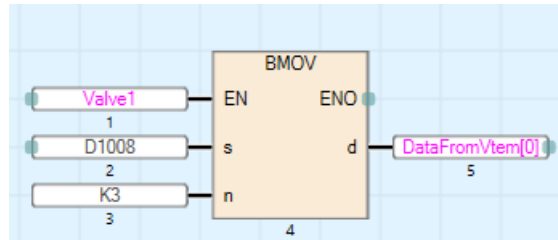
- Point 5. - In Simple CPU Communication, if multiple Settings are assigned to read/write data with CPX Head module, any of those Setting No. can be used for Simple CPU Communication status.

10 Sample Code For VTEM Function Block

This sample code is used to control, read or write, save parameters and diagnostic data of VTEM Motion Terminal module connected to Mitsubishi PLC through Modbus Simple CPU Communication network using “**Festo_VTEM_DevCon_Rseries_Lib**” library FBs.

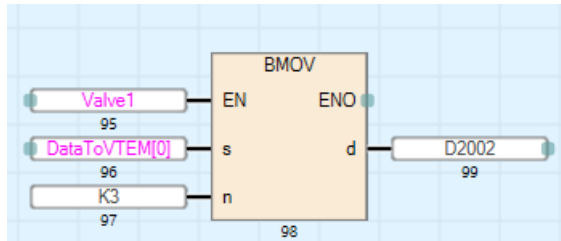
10.1 Sample Code for Mapping Modbus Simple CPU Communication I/O Data to Internal I/O Array Tag

Input Mapping



1. Call the “**BMOV**” Function block to move the input data from Modbus Simple CPU Communication input data to Internal array tag.
2. Assign the enable tag to “**EN**” to enable the function block.
3. Assign the Starting address of the Modbus Simple CPU Communication input Data of specific valve in the “**S**” input.
4. Assign the number of data to be moved in the “**n**” input.
5. Assign the starting address of the internal array tag in the “**d**” output.

Output Mapping



1. Call the “**BMOV**” Function block to move the Internal array tag data to Modbus Simple CPU Communication output data address.
2. Assign the enable tag to “**EN**” to enable the function block.
3. Assign the Starting address of the Internal array tag in the “**S**” input.
4. Assign the number of data to be moved in the “**n**” input.
5. Assign the starting address of the Modbus Simple CPU Communication output tag in the “**d**” output.

10.2 Sample Code for Control the VTEM Motion Terminal Valve Using “FB_ValveControl” FB

The sample code is used to control the specific valve using FB_ValveControl function block.



Motion App 02
Proportional directional control valve

Control

App Option (Byte 1): Valve type, meaning of actual values
nc = normally closed

Value	Valve type (Bit 0...3)	Actual value 1 (Bit 5)	Actual value 2 (Bit 6)
14	4/3 G (nc) (14)	valve position (0)	valve position (0)
15	2x3/3 G (nc) (15)	valve position (0)	valve position (0)
46	4/3 G (nc) (14)	pressure (1)	valve position (0)
47	2x3/3 G (nc) (15)	pressure (1)	valve position (0)
78	4/3 G (nc) (14)	valve position (0)	pressure (1)
79	2x3/3 G (nc) (15)	valve position (0)	pressure (1)
110	4/3 G (nc) (14)	pressure (1)	pressure (1)
111	2x3/3 G (nc) (15)	pressure (1)	pressure (1)

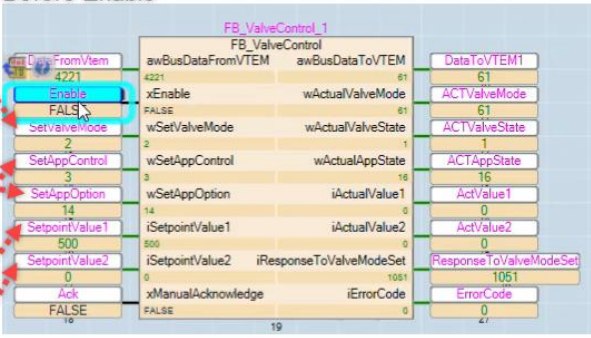
App Control (Byte 0, Bits 6 ... 7): Outlet port activation
A = Active, I = Inactive

Valve type	App Control value			
	00	01	02	03
	(2)	(4)	(2)	(4)
4/3 G (nc)	I	I	I	A
2 x 3/3 G (nc)	I	I	A	A

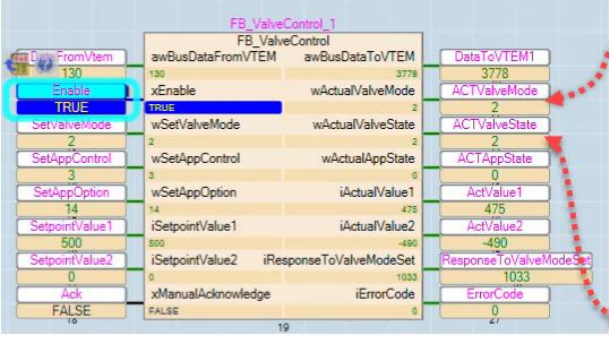
Setpoint Value 1 (Byte 2 ... 3): Setpoint valve position
Val. type 4/3 G: outlet port (2) and (4)
-10000 (14, 32) ... +10000 (12, 34) × 0.01 %
Val. type 2x3/3 G: outlet port (2)
-10000 (32) ... +10000 (12) × 0.01 %

Setpoint Value 2 (Byte 4 ... 5): Setpoint valve position
Val. type 4/3 G: not used
Val. type 2x3/3 G: outlet port (4)
-10000 (34) ... +10000 (12) × 0.01 %

Before Enable



After Enabled



Actual Valve Mode	Meaning
0	reserved
1 ... 59	Running Motion App 1 ... 59
60	Executing Teach-in run
61	Valve is inactive or performs self-calibration
63	Transfer Mode is active

Valve State	Meaning
0	not ready Startup process for Motion Terminal is not completed or an error that has been detected and eliminated is yet to be acknowledged.
1	configurable The valve is inactive. A Motion App can be run or a switch to transfer mode can be made.
2	running A Motion App is currently being executed.
3	failure An error has been detected but not yet eliminated. Starting / Running a Motion App is not possible.

1. Enter the Motion App number in the “wSetValveMode” input tag.
2. Enter the App control value in the “wSetAppControl” input tag.
3. Enter the App option value in the “wSetAppOption” input tag.
4. Enter the setpoint value1 in the “iSetpointValue1” input tag.
5. Enter the setpoint value2 in the “iSetpointValue2” input tag.
6. Once all the parameters entered in the input tags then toggle the “xEnable” tag to proceed the valve control.
7. If Error shows in the “iErrorCode” monitor tag then user has to execute the Acknowledge function by using “xManualAcknowledge” control tag. Then user has to enable the block once again.
8. Once the function block is enabled then user will get the feedback from the valve like Actual valve mode, Actual valve state, Actual App state, Actual value1 and Actual value2 in the appropriate tags.
9. And user will get the response to valve mode set value in “iResponseToValveModeSet” monitor tag in Integer format. (Refer the [Chapter 7.1.2](#)).
10. If error occurs in the block user can get the error code data in “iErrorCode” monitor tag in Integer format.
11. And Read data will be stored in the “awReadData” array tag as shown in above picture. Data value is separated in to word of array. For an example if user want to read DINT value then it has to split into 2 word.

10.3 Sample Code for Upload VTEM Motion Terminal Valve Parameter Using “FB_Upload_Single_Parameter” FB

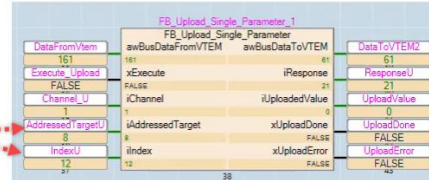
This sample code is used to Upload the valve parameter of the VTEM Motion terminal using the Upload single parameter function block.



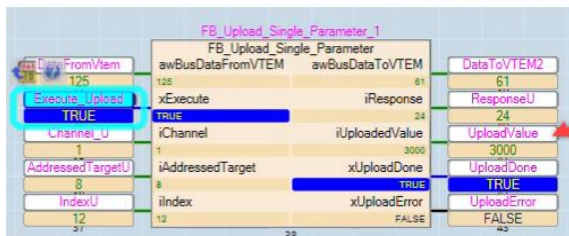
Motion App 08
Selectable pressure level (ECO)

Type	Index	Parameter	Range	Unit
System	12	Tubing length at (2)	0 ... 20000	1 mm
	13	Tubing length at (4)	0 ... 20000	1 mm
	14	Tubing inner diameter at (2)	200 ... 1100	0.01 mm
	15	Tubing inner diameter at (4)	200 ... 1100	0.01 mm
	20	Drive type (→ List of supported drives at Support Portal)	—	—
	21	Drive stroke	10 ... 5000	1 mm
60		Installation position of drive	−18000 ... +18000	0.01 °

Before Execute



After Execute



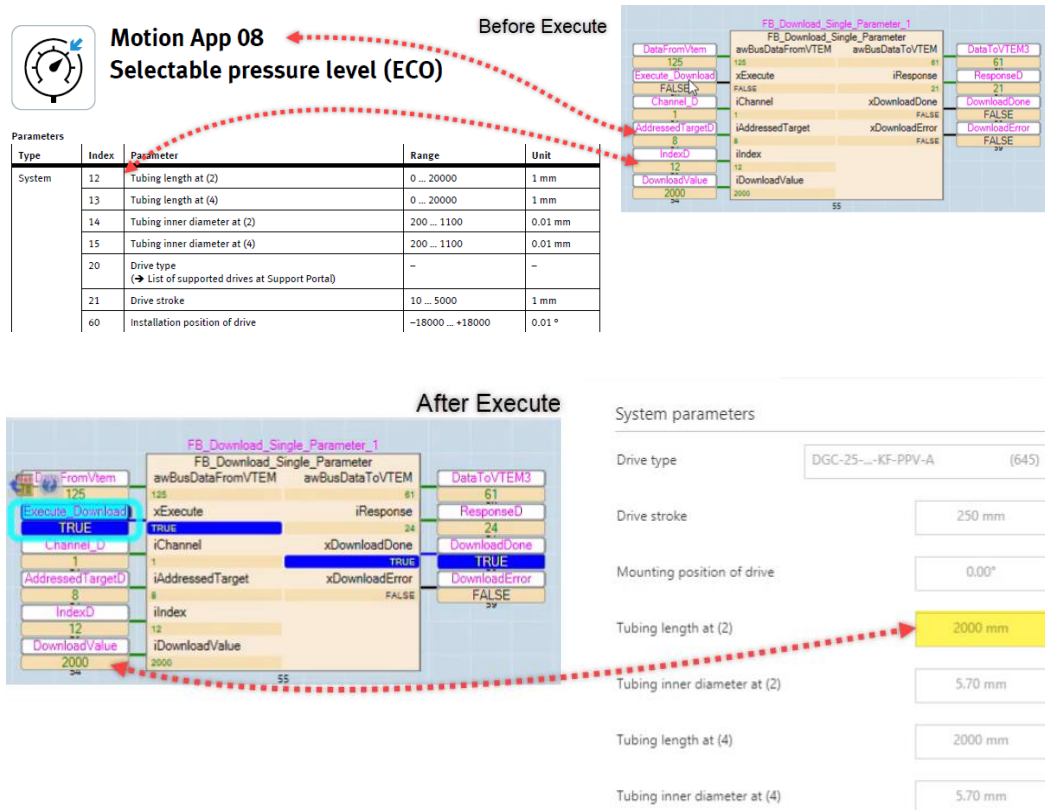
System parameters

Drive type	DGC-25-...-KF-PPV-A (645)
Drive stroke	250 mm
Mounting position of drive	0.00°
Tubing length at (2)	3000 mm
Tubing inner diameter at (2)	5.70 mm
Tubing length at (4)	2000 mm
Tubing inner diameter at (4)	5.70 mm

1. Enter the channel number nothing but the Program SET number in the “**iChannel**” input tag.
2. Enter the motion app number in the “**iAddressedTarget**” input tag.
3. Enter the index number in the “**iIndex**” input tag.
4. Once all the parameters entered in the input tags then toggle the “**xExecute**” tag to proceed upload process.
5. Once parameters value upload are successful then the monitor tag “**xUploadDone**” will be in true state.
6. And user will get the Upload data in the “**iUploadedValue**” monitor tag in integer format.
7. And user will get the response of upload process in “**iResponse**” monitor tag in Integer format.(Refer the [Chapter 7.3.2](#)).
8. If error occurs in the block user can get the error feedback in “**xUploadError**” monitor tag.

10.4 Sample Code for Download VTEM Motion Terminal Valve Parameter Using “FB_Download_Single_Parameter” FB

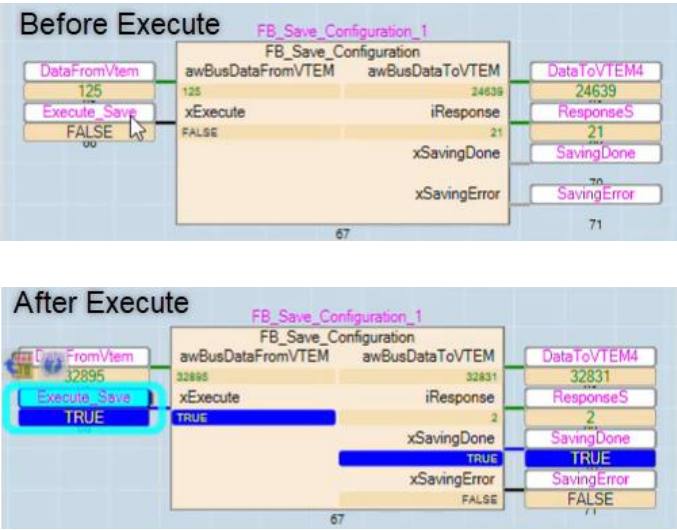
This sample code is used to Download the valve parameter of the VTEM Motion terminal using the Download single parameter function block.



1. Enter the channel number nothing but the Program SET number in the “**iChannel**” input tag.
2. Enter the motion app number in the “**iAddressedTarget**” input tag.
3. Enter the index number in the “**iIndex**” input tag.
4. Enter the parameter data value in the “**iDownloadValue**” input tag.
5. Once all the parameters entered in the input tags then toggle the “**xExecute**” tag to proceed Download process.
6. Once parameters value download are successful then the monitor tag “**xDownloadDone**” will be in true state.
7. And user will get the response of download process in “**iResponse**” monitor tag in Integer format.(Refer the [Chapter 7.3.2](#)).
8. If error occurs in the block user can get the error feedback in “**xDownloadError**” monitor tag.

10.5 Sample Code for Save the VTEM Motion Terminal Valve Parameter Using “FB_Save_Configuration” FB

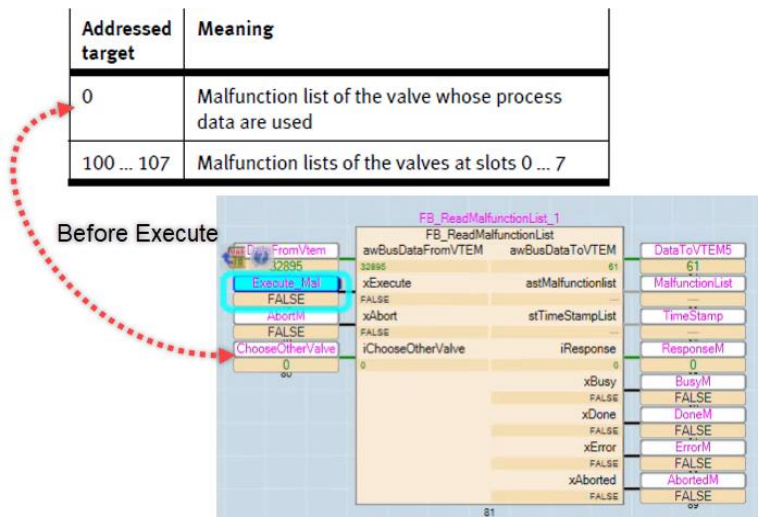
This sample code is used to Save the valve parameter of the VTEM Motion terminal using the Save configuration function block.

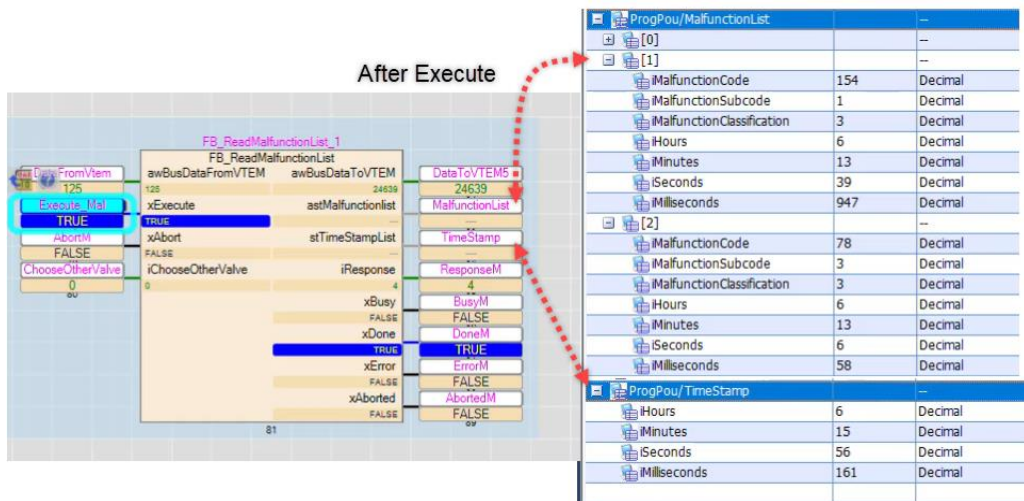


- 1. Toggle the “**xExecute**” tag to proceed save process.
- 2. Once parameters value save are successful then the monitor tag “**xSavingDone**” will be in true state.
- 3. And user will get the response of save process in “**iResponse**” monitor tag in Integer format.(Refer the [Chapter 7.4.2](#)).
- 4. If error occurs in the block user can get the error feedback in “**xSavingError**” monitor tag.

10.6 Sample Code for Read the Malfunction list of the VTEM Motion Terminal Valve Using “FB_ReadMalfunctionList” FB

This sample code is used to read the malfunction list of the VTEM Motion terminal using the Read Malfunction list function block.





1. Enter the Choose other valve nothing but the process data or specific valve selection in the “**iChooseOtherValve**” input tag.
2. Toggle the “**xExecute**” control tag from false to true to read the malfunction list.
3. If Error shows in the “**xError**” monitor tag then user has to execute the abort function by using “**xAbort**” control tag.
4. Once abort done then the monitor tag “**xAborted**” will be in true state. Then user has to execute the block once again.
5. Once malfunction list read are successful then the monitor tag “**xDone**” will be in true state.
6. And user will get the Malfunction list in the “**stMalfunctionList**” structure data type tag as shown in above picture.
7. And user will get the current time stamp in the “**stTimeStamp**” structure data type tag as shown in above picture.
8. And user will get the response of download process in “**iResponse**” monitor tag in Integer format.(Refer the [Chapter 7.5.2](#)).