Application Note

FESTO

FPosCR Integration with Siemens PLC through Profinet

This Application Note describes how to use the host function blocks of the FPosCR from Festo & how to integrate them in Siemens TIA Portal programming environment using Profinet communication.

CPX-E-CEC-M1-PN

100758

Title ..... FPosCR Integration with Siemens PLC through Profinet
Version 1.20
Document no. 100758
Original ..... en
Author ..... Festo

Last saved 19.11.2026

General information:

This document is intended for qualified, trained and instructed professionals. The data provided in this document has no guaranteed specifications, in particular with regard to functionality, condition or quality in the legal sense. The information in this document is intended only as simple indications for the implementation of a specific, hypothetical application, and in no way as a substitute for the operating instructions of the respective manufacturers or the design and testing of the respective application by the user. The respective operating instructions for Festo products can be found under www.festo.com. The user of this document must ensure that each function described herein works properly in his application. The user remains solely responsible for his or her own use of this document, even by studying this document and using the information mentioned therein. This also applies to any software (in source code and/or object code) that is made available to the user as an appendix to this document.

Rights of use of the software:

If software is made available to the user as an appendix to this document, the user is granted a simple and unlimited right of use hereto. This right also includes the processing and distribution of the software to third parties in edited or unedited form. The User is not permitted to use the name "Festo" to endorse or promote products derived from the Software without express prior written permission.

Software is provided to the user "as is". Festo does not assume any warranty or guarantee with regard to software. Festo's liability for damages of any kind arising from the use of the software is limited to intent and gross negligence.

Legal Notices:

©Festo SE & Co. KG, all rights reserved. A change in content and form is only permitted with the express written consent of Festo SE & Co. KG. Festo grants the user the right to reproduce this document in a form that remains unchanged in terms of content and form and to pass it on to third parties.

© (Festo SE & CO. KG, D-73726 Esslingen, 2026)
Internet: http://www.festo.com

Table of contents

1 Introduction 7

1.1 Software Used 8

1.2 Hardware Used 8

1.3 Topology Overview 9

2 Software Configuration 10

2.1 Creating a new project and Adding controller in TIA Portal 10

2.2 Siemens S7-1500 PLC IP Address Configuration 12

2.3 Adding the GSD File of CPX-E-CEC-M1-PN to TIA Portal 14

2.4 Adding CPX-E-CEC-PN module to Network View 16

2.5 CPX-E-CEC-PN IP Address Configuration 19

2.6 CPX-E-CEC-PN IO Module Configuration 21

2.7 Assigning Profinet Name to CPX-E-CEC-PN module 23

2.8 Finding Hardware ID of CPX-E-CEC-PN Module 25

2.9 Adding FPosCR_Host_Client_PN Library to TIA Portal 27

2.9.1 Import the FPosCR_Host_Client_PN Library into Global Libraries 27

2.9.2 Adding Function blocks and supporting files to program 30

3 FPosCR Host PN Library Function Block Description 31

3.1 Introduction to FPosCR_Host_Client_Basic Function Block 31

3.1.1 FPosCR_Host_Client_Basic Block Input Variable Details 31

3.1.2 FPosCR_Host_Client_Basic Block Output Variable Details 32

3.2 Introduction to FPosCR_Host_Client_Extended Function Block 32

3.2.1 FPosCR_Host_Client_Extended Block In-Out Variables 33

3.2.2 FPosCR_Host_Client_Extended Block Input Variables 33

3.2.3 FPosCR_Host_Client_Extended Block Output Variables 33

3.3 Structure 34

3.3.1 HOST_CONTROL_DWORD 34

3.3.2 HOST_STATUS_DWORD 34

3.3.3 DYNAMIC_CLIENT 35

3.3.4 FPosCR_Ref 36

3.3.5 MC_COOR_SYS_3P_CLIENT 36

3.3.6 MC_COOR_POS _CLIENT 36

3.3.7 RING_BUFFER_PAR 37

3.3.8 DATE_AND_TIME 37

3.3.9 Error Description 37

4 Configuration of FPosCR Host Client Function Block in TIA Portal 43

4.1 Configuring the Function Block Input & Output Interface Tags 43

5 Sample Code Description for FPosCR Host Client Block 46

5.1 Initialize the function block 46

5.2 Control and monitor the FPosCR Server using FPosCR_Host_Client_Basic Block 47

5.2.1 Take control 47
5.2.2 Error Monitor and Acknowledge 48
5.2.3 Power On/Off 48
5.2.4 Homing the system 48
5.2.5 xStop_Abort 48
5.2.6 Jogging 49
5.2.7 Halt 49
5.2.8 Acknowledge All M-Functions 49
5.2.9 Enable Path Correction 50
5.2.10 Acknowledge Error 50
5.2.11 Acknowledge actual M-Function 50
5.2.12 Enable CNC StepMode / Next Step CNC 51
5.2.13 Start User Prg 51
5.2.14 Start CNC 52
5.2.15 Actual Position 52
5.2.16 Actual Coordinate System 52
5.2.17 Actual Movement ID 53
5.2.18 Actual Path Velocity 53
5.2.19 Actual Path Acceleration and Deceleration 53
5.2.20 Last MC Movement ID 53
5.2.21 Actual Coordinate System 53

5.3 Control and monitor the FPosCR Server using FPosCR_Host_Client_Extended Block 54

5.3.1 Lin 54
5.3.2 LinRel 55
5.3.3 PTP 55
5.3.4 PTP_Rel 56
5.3.5 Circ 56
5.3.6 Circ_rel 56
5.3.7 MoveAxisCartRel 57
5.3.8 HomingSingleAxis 57
5.3.9 ForceGroupHomed 58
5.3.10 JogSteppingParam 58
5.3.11 Dyn 58
5.3.12 Wait 58
5.3.13 Blending Distance 59
5.3.14 Blending Vel 59
5.3.15 BufferMode 59
5.3.16 ChangeRefSys 60
5.3.17 SetRefSys 60
5.3.18 SetRefSys3P 60
5.3.19 DeleteConfig 61
5.3.20 LoadConfig 61

5.3.21 ReadErrMessage 61
5.3.22 SetSysTime 62
5.3.23 ReadTime 62
5.3.24 BrakeControl 62
5.3.25 SetPathCorrectionParam 63
5.3.26 CNCFile 63
5.3.27 Braketest 63
5.3.28 ReadParam 64
5.3.29 SetParam 66
5.3.30 SaveParamSet 67
5.3.31 UpdateConfig 68
5.3.32 iParamListID Values 68
5.3.33 iParamID and iParamSubID 68

1 Introduction

This application note on the FPosCR Host Client software library helps in connecting a Cartesian handling system or Cartesian robot automated with the FPosCR software library to a higher-level industrial controller (e.g., PLC).

The application note describes the main components and contents of the FPosCR Host Client library by way of example. It briefly describes the basic functional structure and the basic possibilities as well as the mode of operation of the library. It describes individual functional elements, supported application technologies (Profinet) and mechanisms of action used.

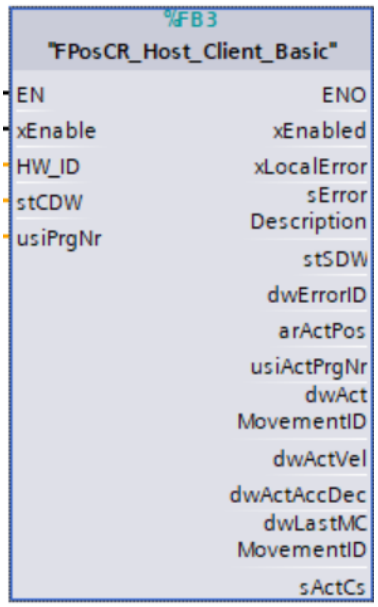
Specific chapters go on to describe a typical procedure and key aspects of setting up, configuring, and parameterizing a communication interface or connection implemented with FPosCR Host. Other chapters provide selected insights into the variety of the library's range of functions and into individual usage and setting options relating to the communication software itself. In other places, further down, various functionalities and access options are shown, how effectively and flexibly a bound Cartesian handling or an integrated Cartesian robot can be accessed – for queries as well as for settings and control actions (read and write).

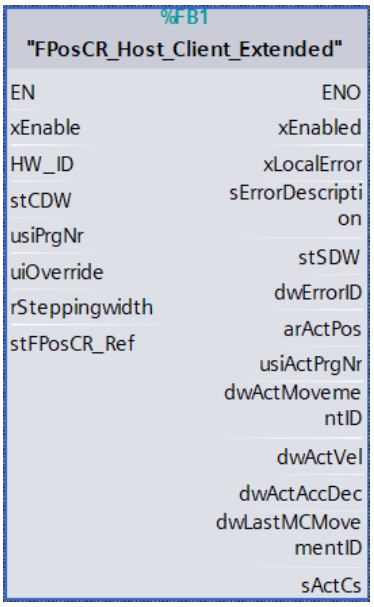
This application note considers the client side of the communication connection.

The library has two function blocks called **FPosCR_Host_Client_Basic** and **FPosCR_Host_Client_Extended.**

The **FPosCR_Host_Client_Basic** allows user to control and monitor the basic functionalities of FPosCR software.

The **FPosCR_Host_Client_Extended** allows users to control and monitor the basic and extended functionalities of FPosCR software. It also includes some callable functions that user can call it based on their user application.





Note

- This function block is intended for both S7-1200 and S7-1500 PLCs and has been evaluated on both PLC models.

1.1 Software Used

Software Name	Version / Firmware
Siemens TIA Portal	V14
CODESYS	V3.5.18 Patch 4
CODESYS SoftMotion (Target Package)	4.16.0.0
Library (FPosCR_Host_Client_PN)	V1.1

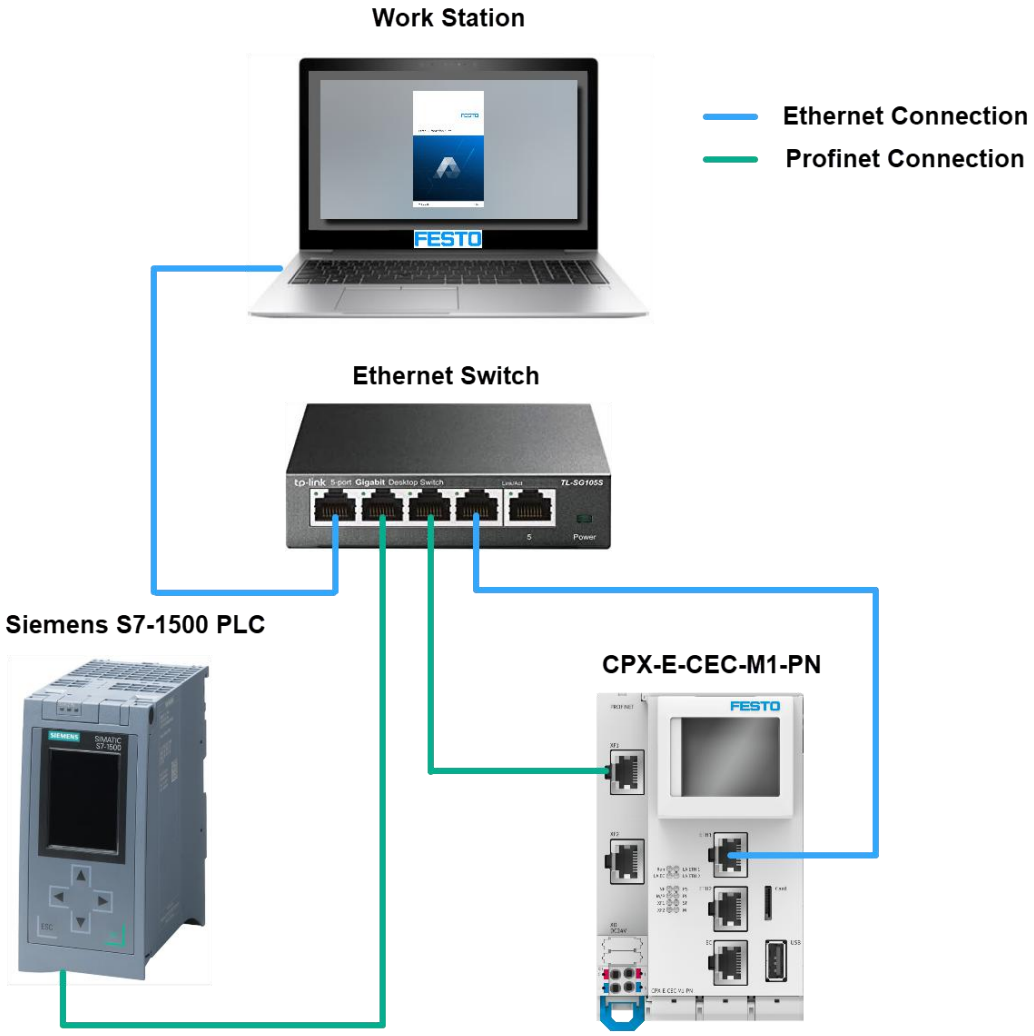
Table 1.1: Software used

1.2 Hardware Used

Device Name	Part Number	Version / Firmware
CPX-E-CEC-M1-PN	4252743	V3.3.8
S7-1500	-	V2.9
Ethernet Cable	-	Version Free

Table 1.2: Hardware used

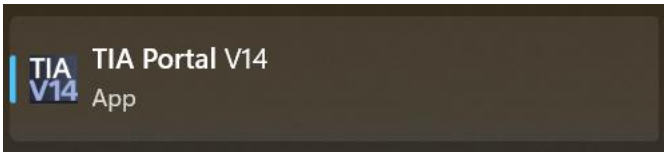
1.3 Topology Overview

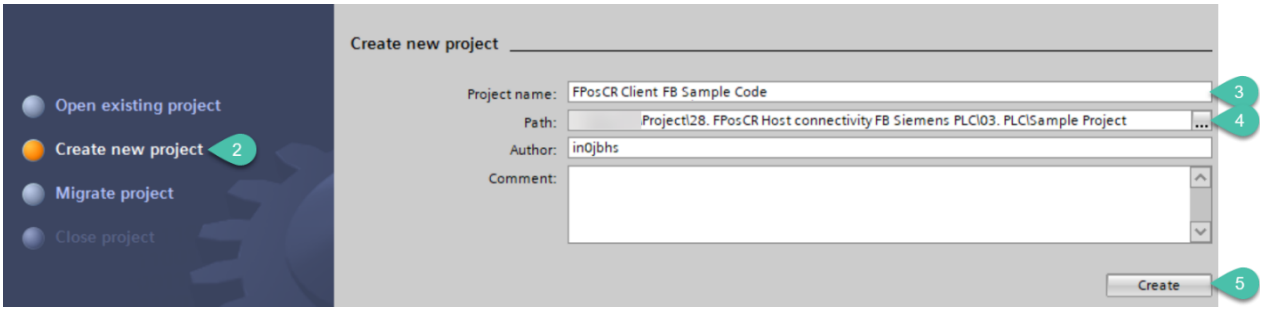


2 Software Configuration

2.1 Creating a new project and Adding controller in TIA Portal

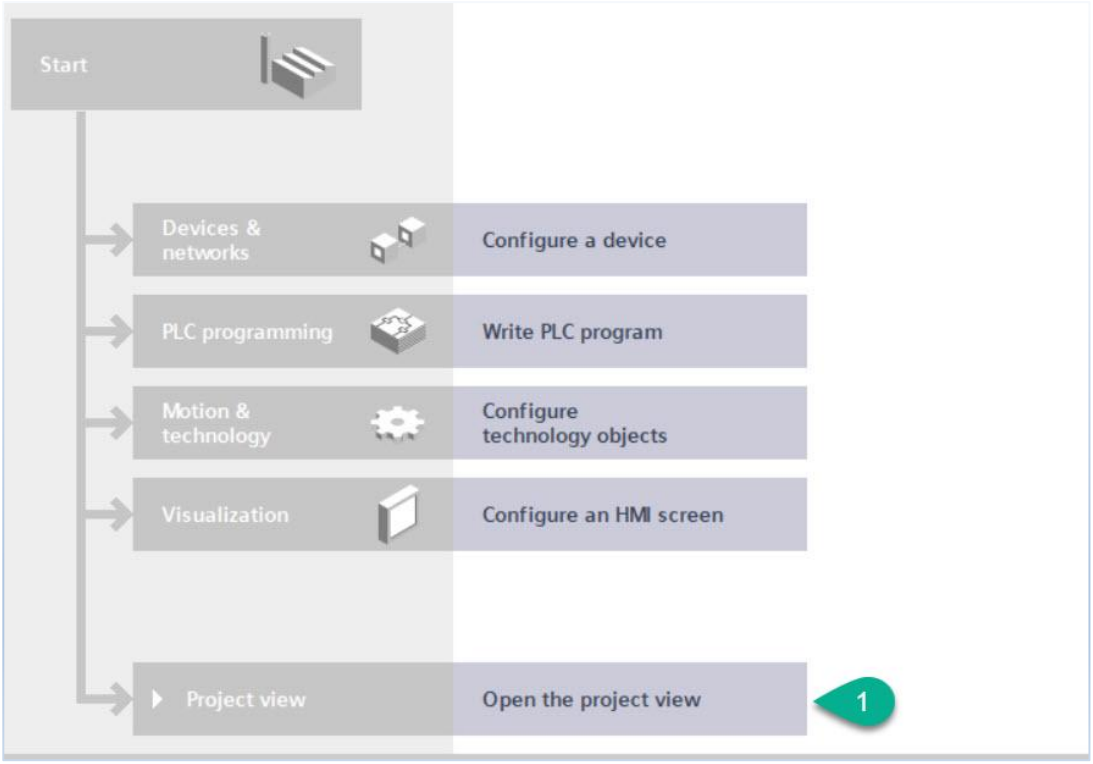
Step 1

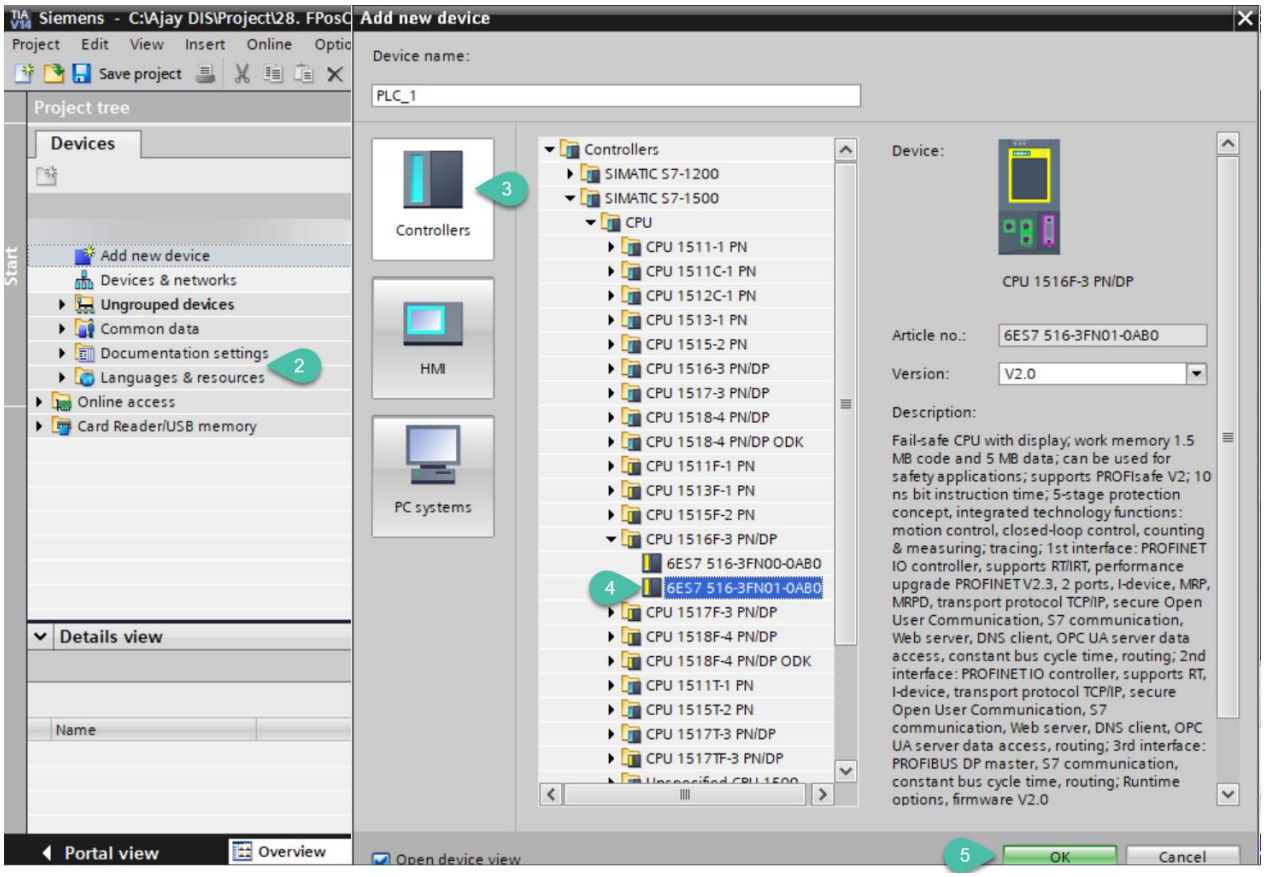




1. Open the TIA Portal V14 Software.
2. Click on **Create a new project** to create the project.
3. Type the Project name in the **Project name** field.
4. Select the **path** where the project must be saved.
5. Click on **Create to Create** a project.

Step 2

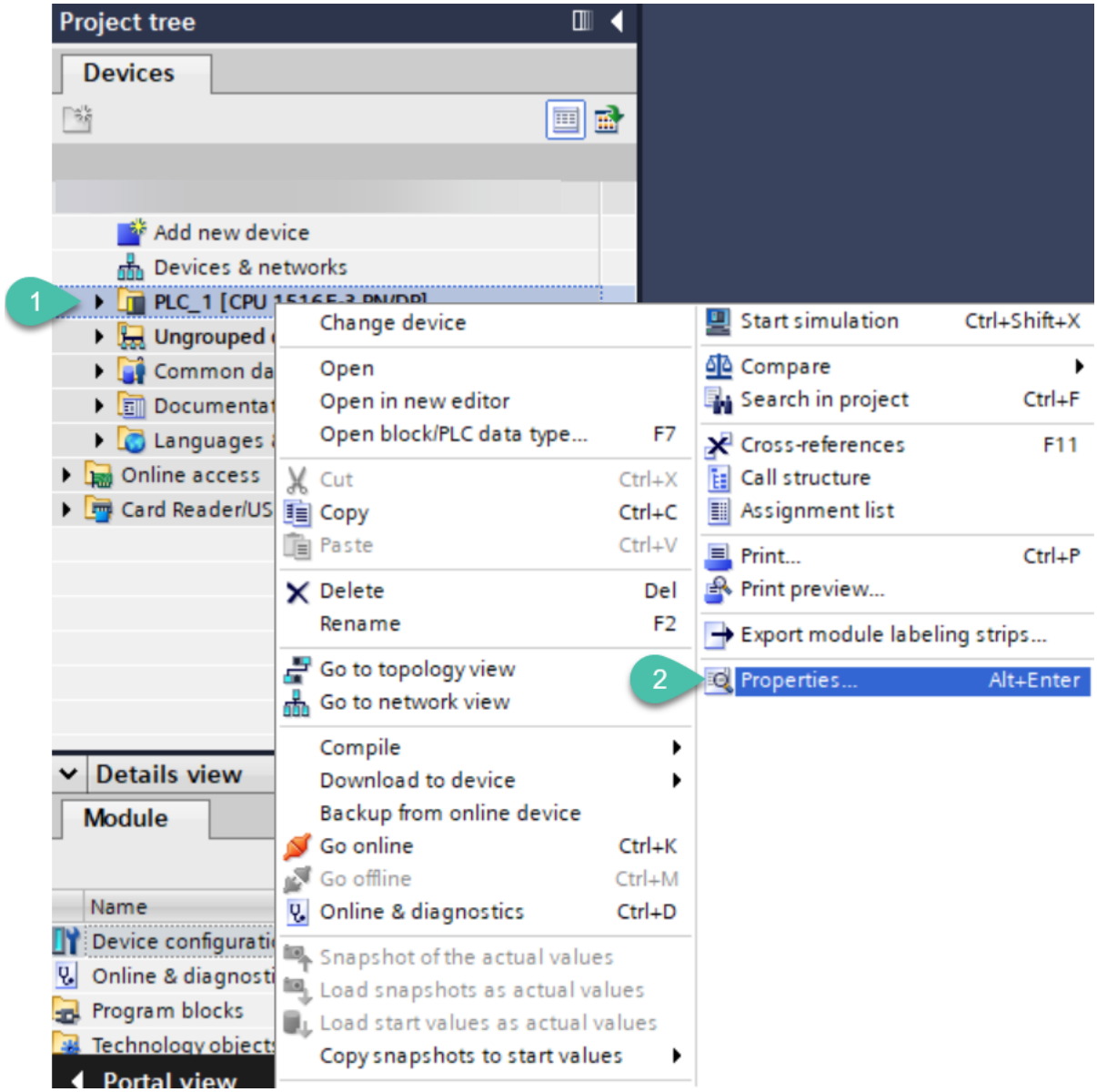




1. Click on **Project view** to Open the project view.
2. Double click on **Add New Device** to add a new PLC to the project.
3. Click on **Controllers;** the list of available Controllers will be displayed.
4. Select the required PLC in the list.
5. Click **OK** to add the selected PLC to the Project.

2.2 Siemens S7-1500 PLC IP Address Configuration

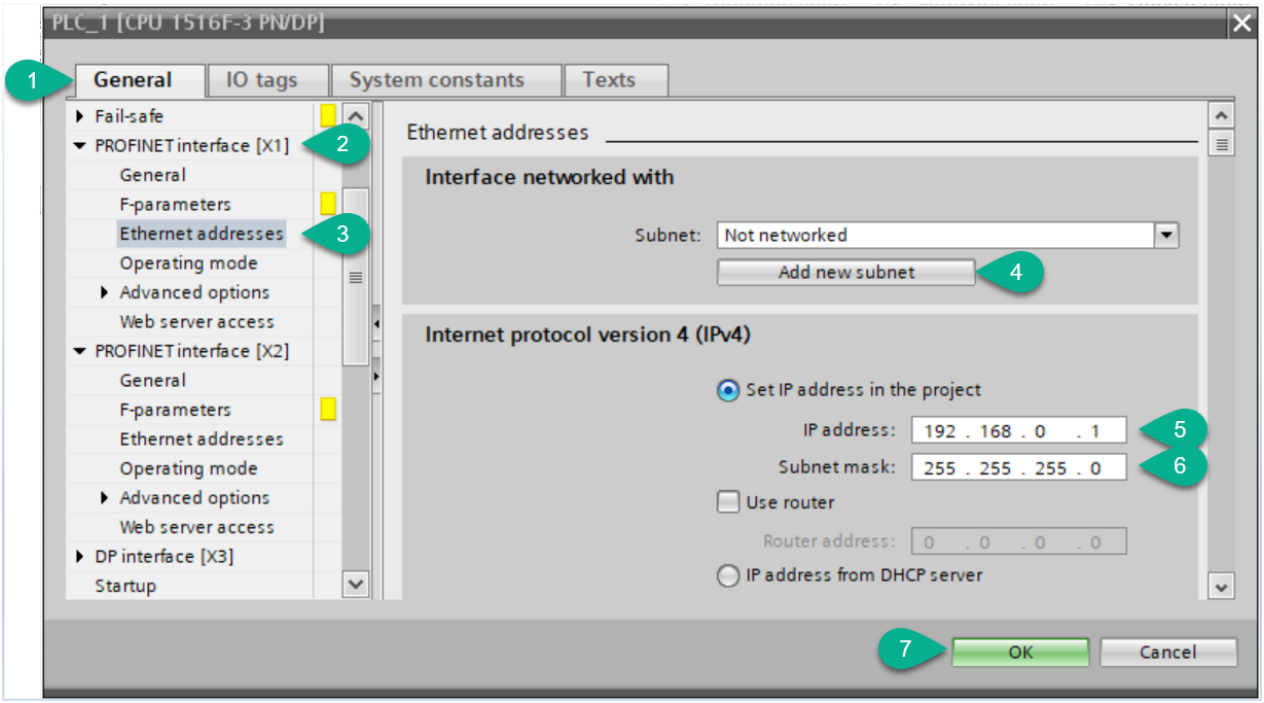
Step 1



1. Right on **PLC_1** from the Project tree.
2. Click on **Properties** from the right click menu.

The properties tab will be displayed as shown below Step 2.

Step 2



1. Select **General** from Properties tab.
2. Click on **the PROFINET interface [X1]** to open the drop down.
3. Click on **Ethernet addresses** from the PROFINET interface dropdown to change the IP address.
4. Click on **Add new subnet** to add new interface network.
5. Enter the required IP address in the **IP address** field.
6. Enter the Subnet mask in the **Subnet mask** field.
7. Click **OK** to confirm the configuration and close the Properties tab.

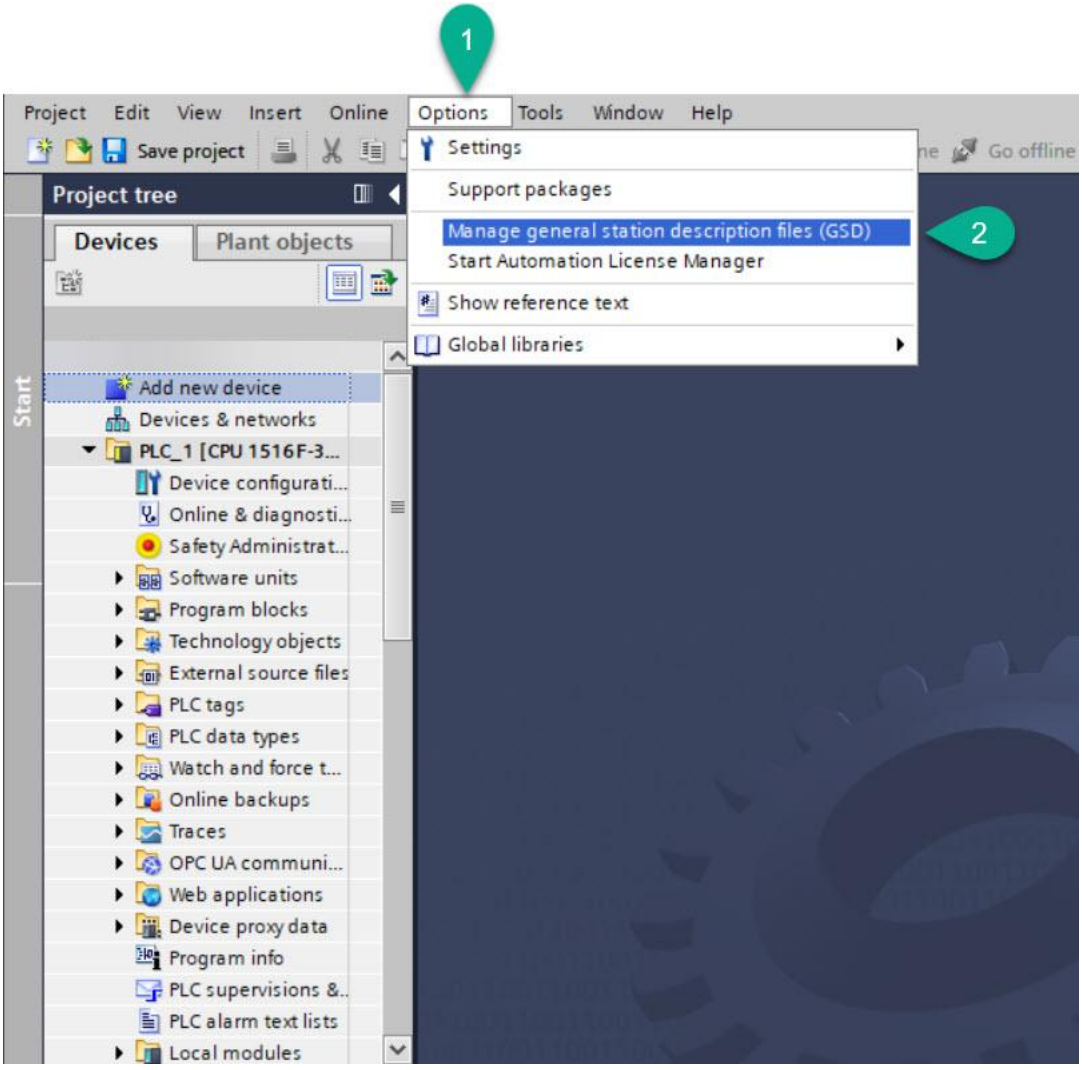
2.3 Adding the GSD File of CPX-E-CEC-M1-PN to TIA Portal

Step 1

Download the GSDML File for **CPX-E-CEC-M1-PN** module from the Festo Support Portal. And save the GSDML file in an appropriate folder in your PC.

Step 2

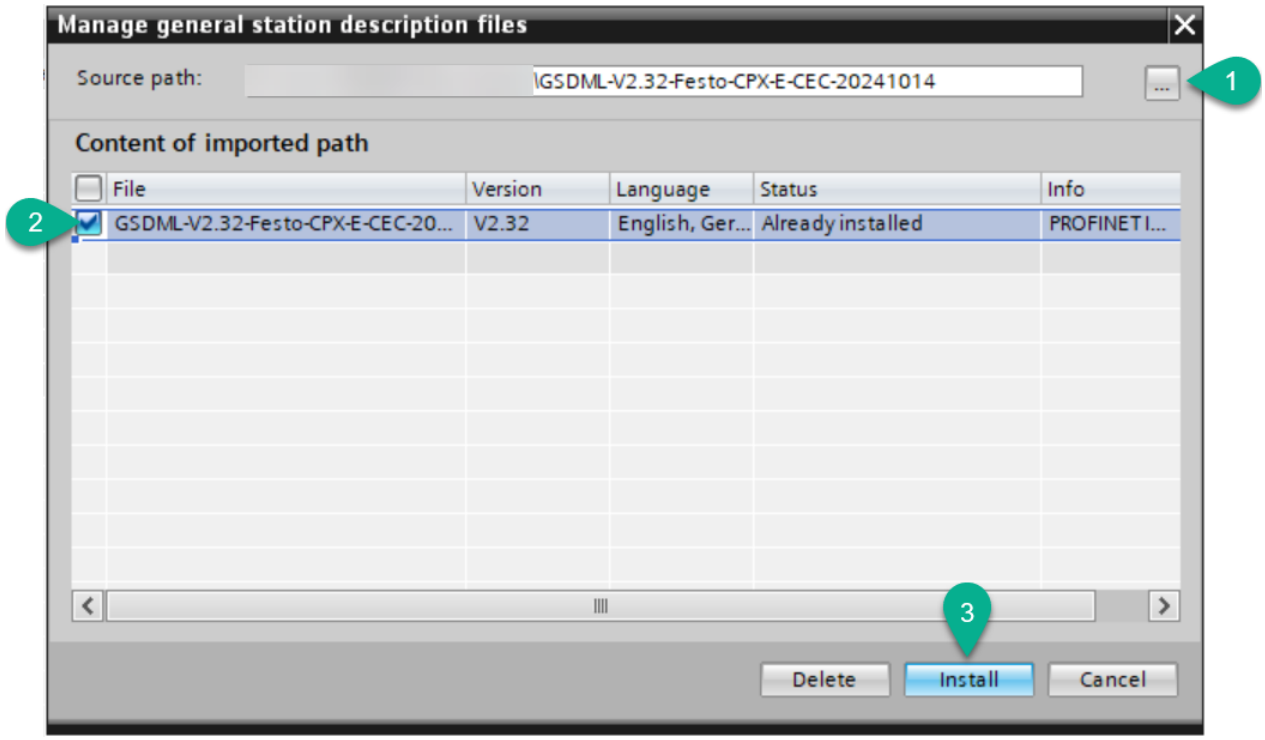
The downloaded GSDML file of **CPX-E-CEC-M1-PN** module must be added to the TIA portal.



1. Click on **the Options** menu from the top menu bar.
2. Click on **Manage General Station description files (GSD)** from the Options menu.

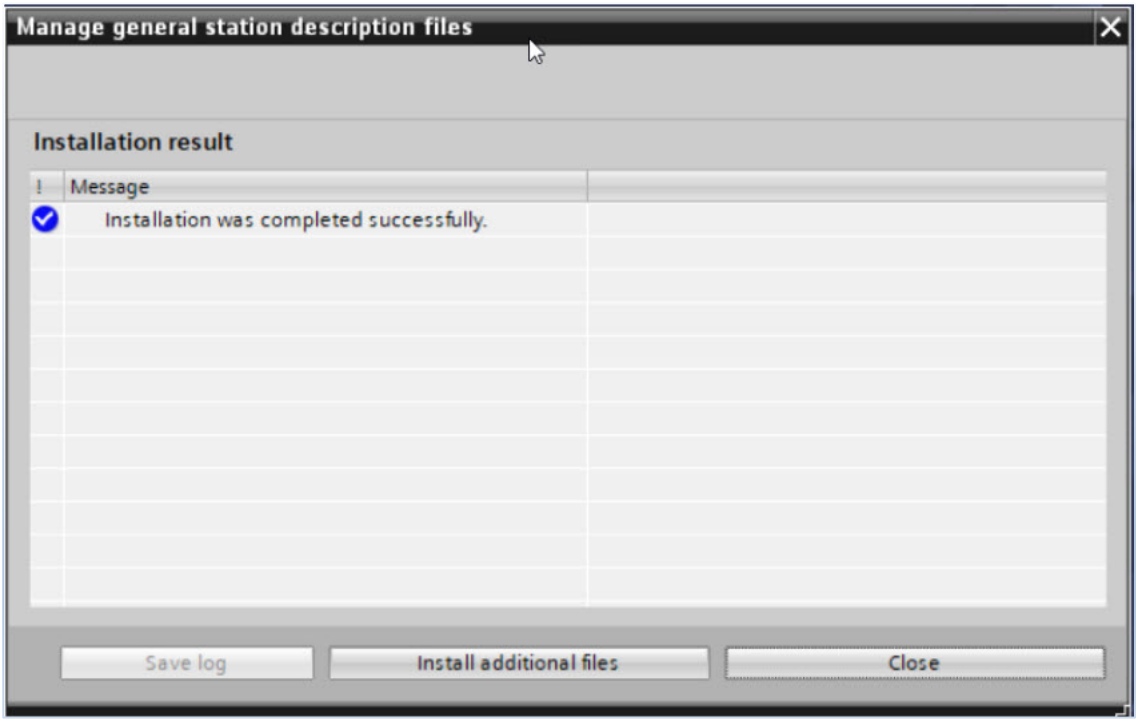
The Manage general station description files tab will be displayed as shown below Step 3.

Step 3



1. Select the Path from the **Source path** field, where the GSDML file is saved.
2. Select the GSDML file from the file list.
3. Click on **Install** to add GSD to the project.

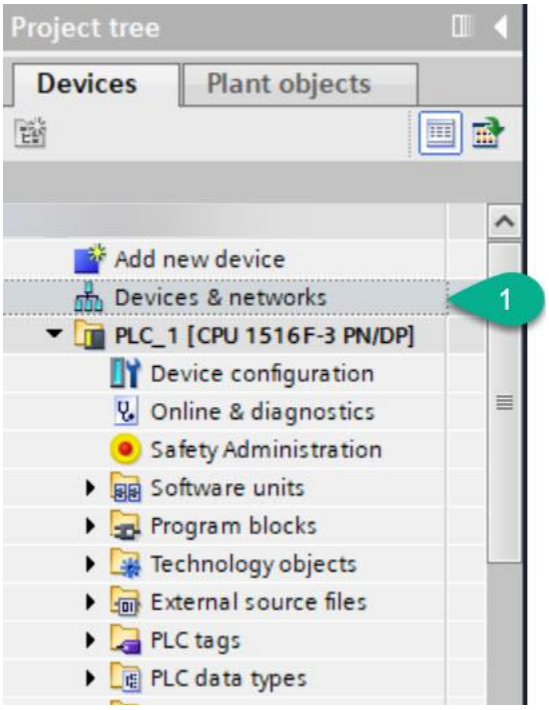
Once the installation is completed, the installation wizard will be shown as below.



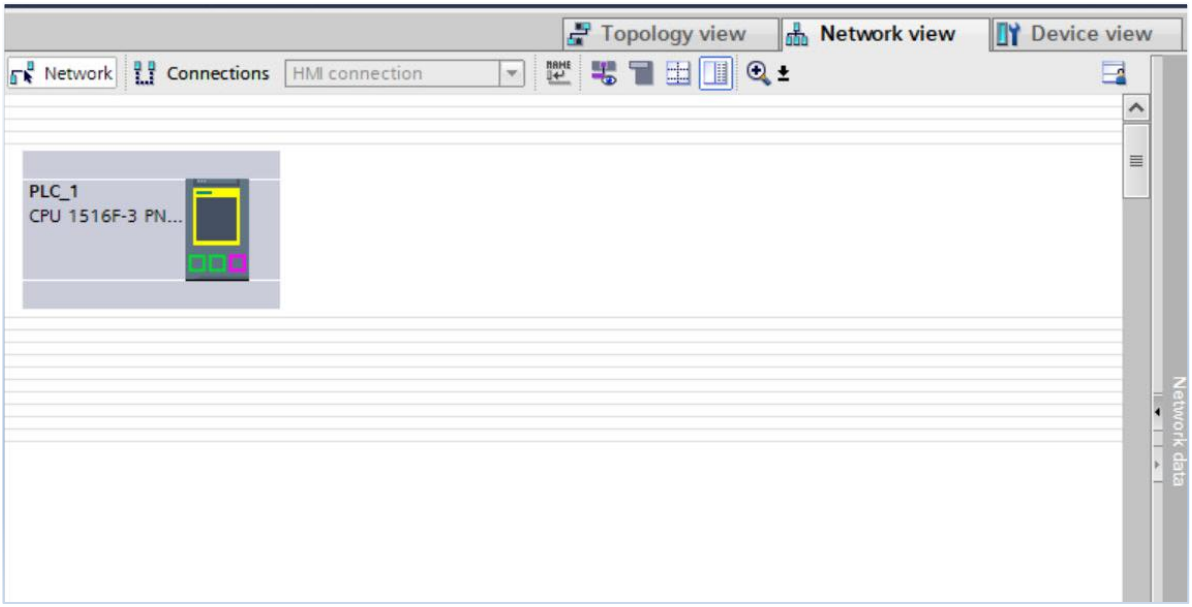
2.4 Adding CPX-E-CEC-PN module to Network View

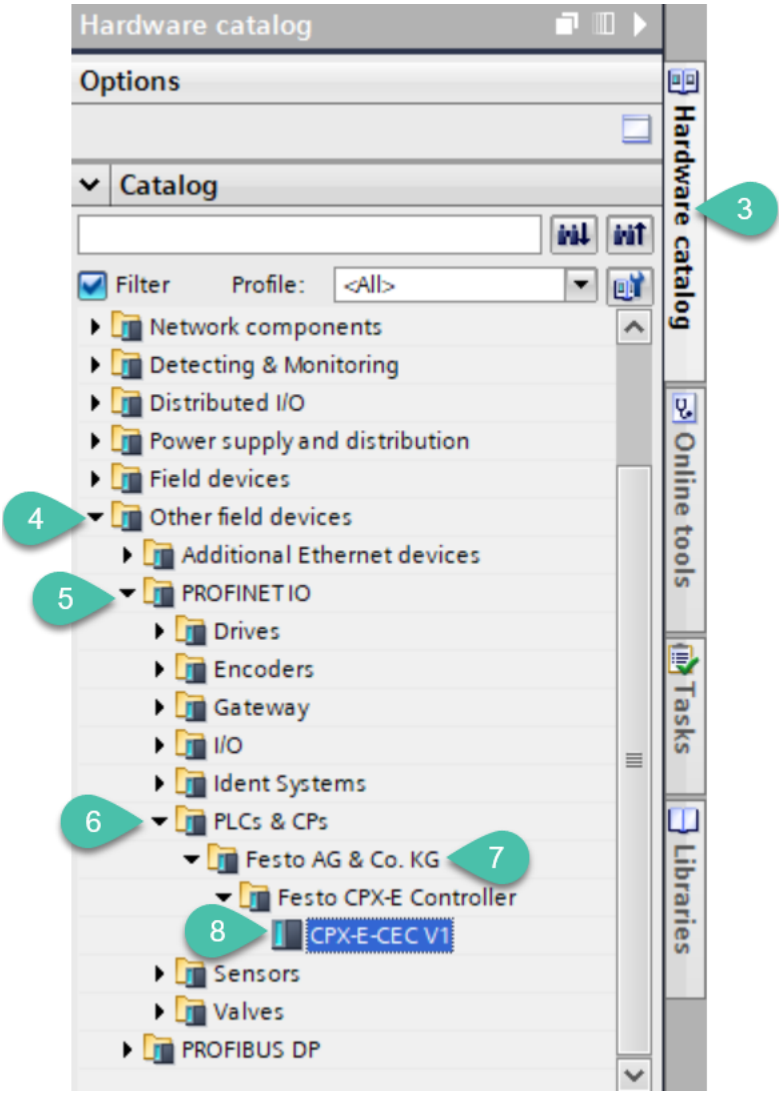
Step 1

1. Double click on **Devices and Networks** to go to network view.



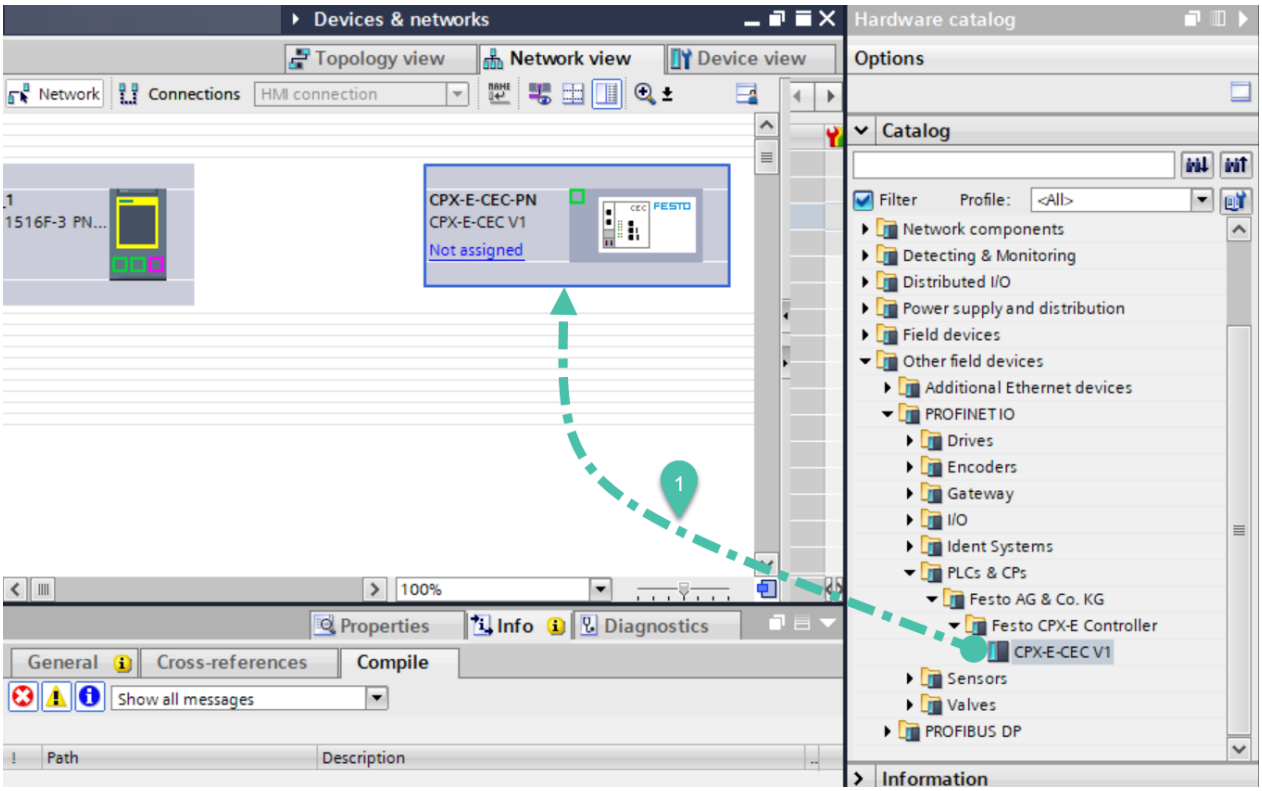
2. The network view will be shown below.

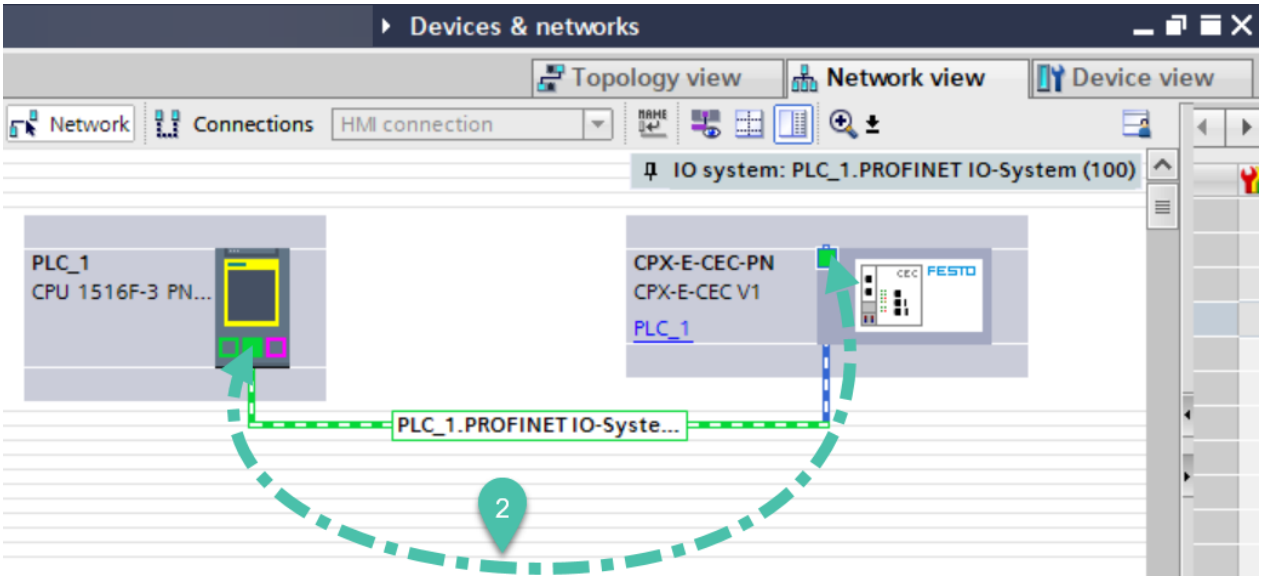




3. Go to **Hardware Catalog** as shown below.
4. Select **Other field devices** from the Hardware catalog menu.
5. Select **PROFINET IO** from the other field devices drop down menu.
6. Select **PLCs & CPs** from the PROFINET IO drop down menu.
7. Select **Festo AG & Co.KG** from the PLCs & CPs drop down menu.
8. **The CPX-E-CEC V1** module can be viewed under the Festo CPX-E Controller folder.

Step 2

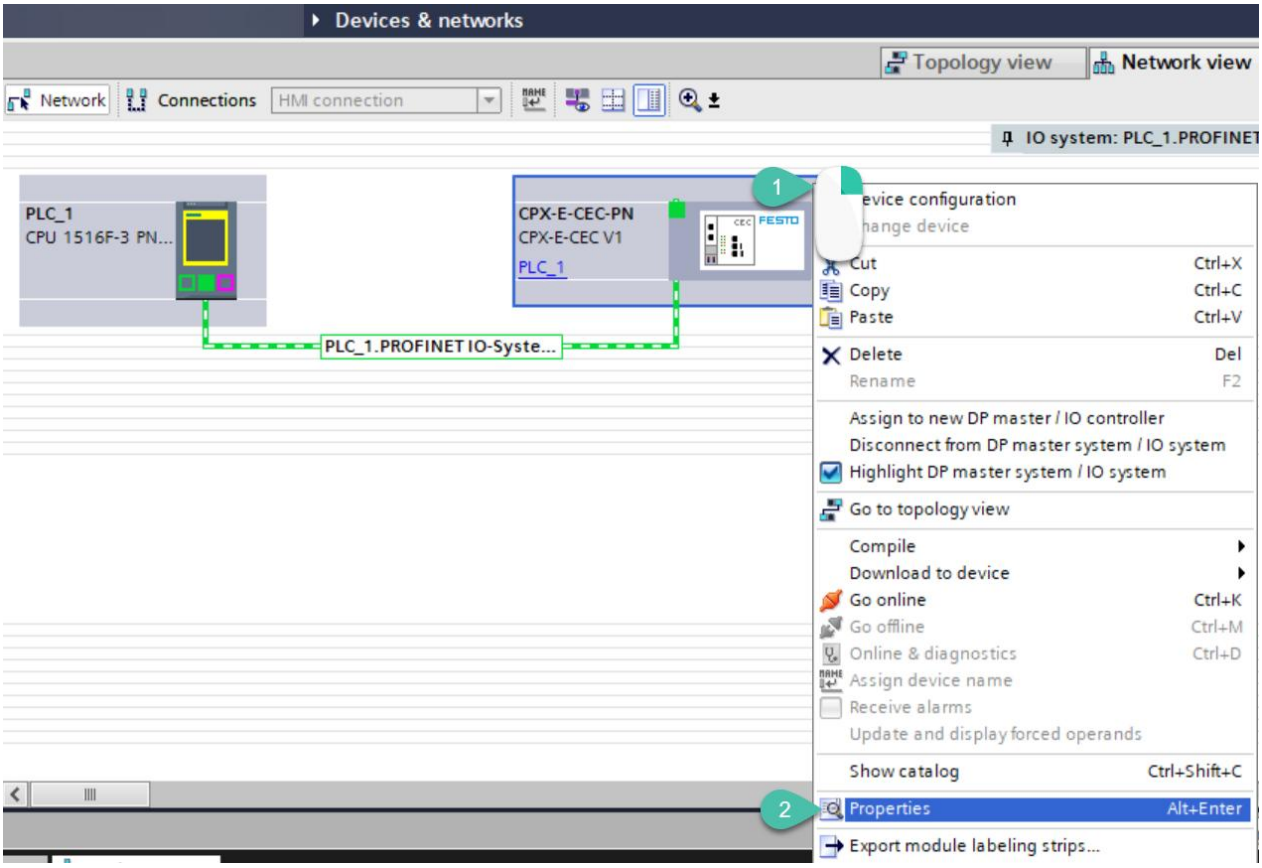




1. Drag and drop **the CPX-E-CEC V1** module to network view.
2. Connect the S7-1500 PLC to the CPX-E-CEC-PN module. In the example, the CPX-E-CEC-PN module is connected to **the PROFINET interface[X1]** of S7-1500 PLC.

2.5 CPX-E-CEC-PN IP Address Configuration

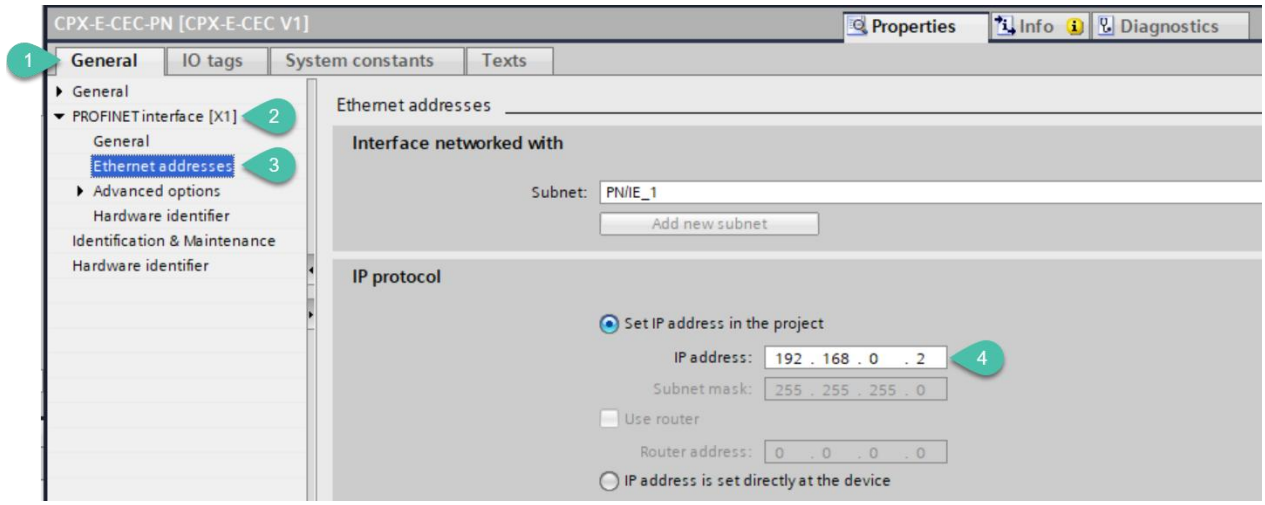
Step 1



1. Right on **AP-I-PN** from the Network view.
2. Click on **Properties** from the right click menu.

The properties will be viewed as shown below Step 2.

Step 2



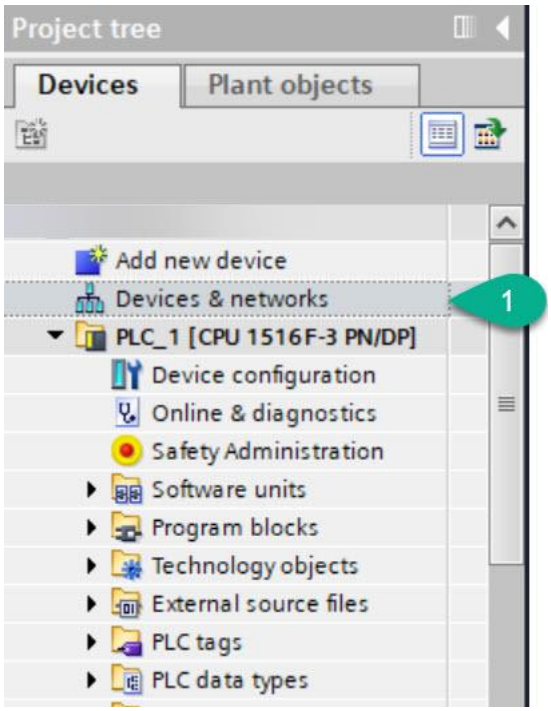
1. Select **General** from Properties tab.
2. Click on **the PROFINET interface [X1]** to open the drop down.
3. Click on **Ethernet addresses** from the PROFINET interface dropdown to change the IP address.
4. Enter the required IP address in the **IP address** field.

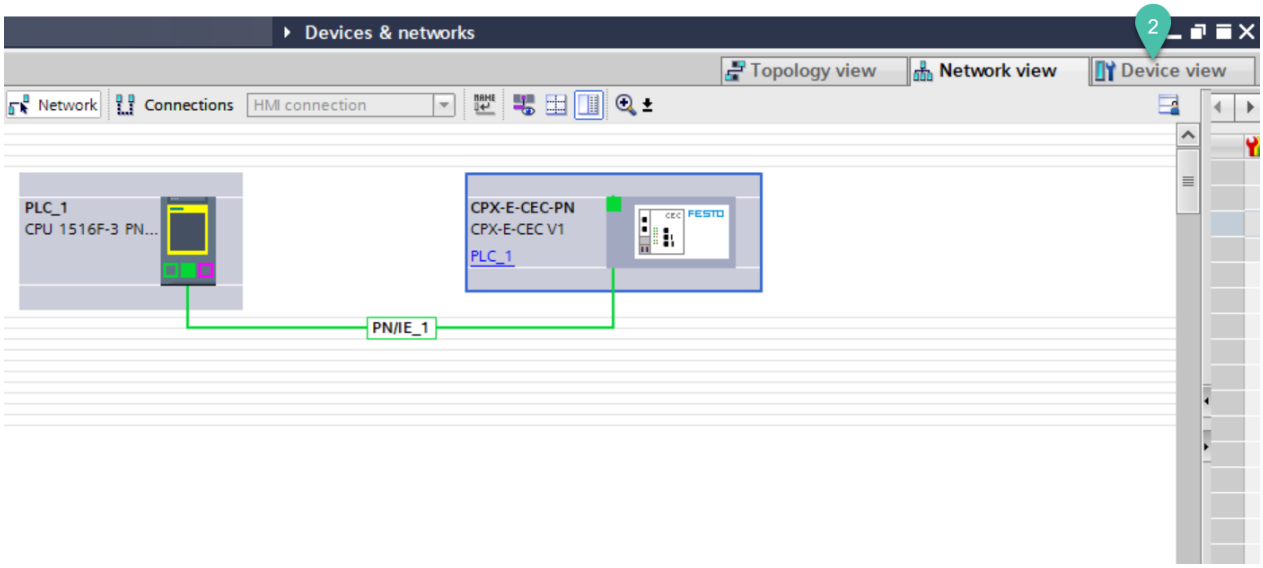
Note

- The IP address of the Profinet interface of the CPX-E-CEC-PN module, S7-1500 PLC, must be in the same range.

2.6 CPX-E-CEC-PN IO Module Configuration

Step 1

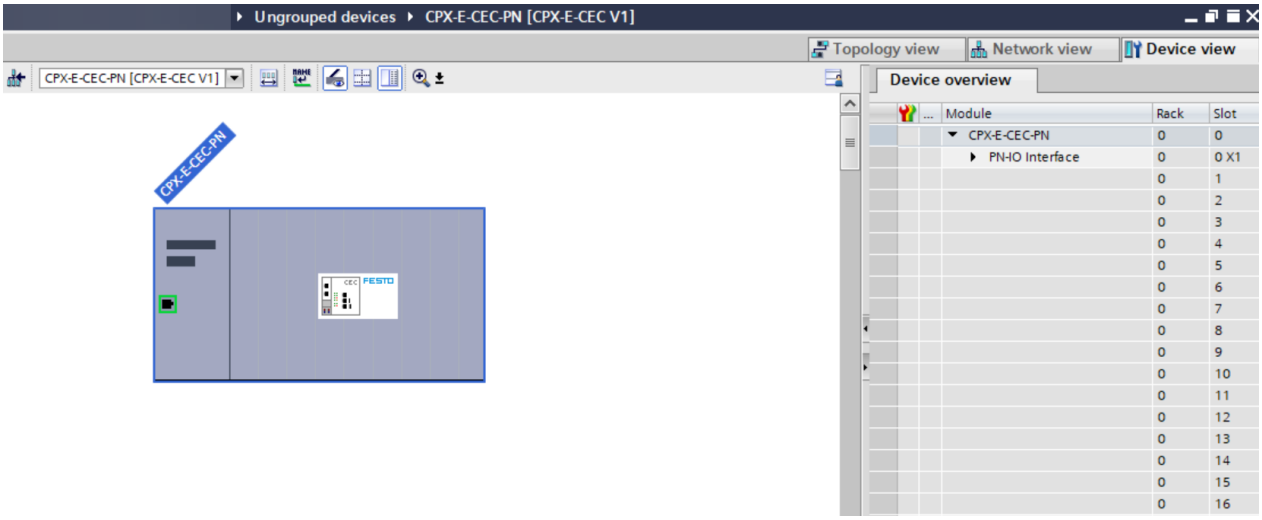


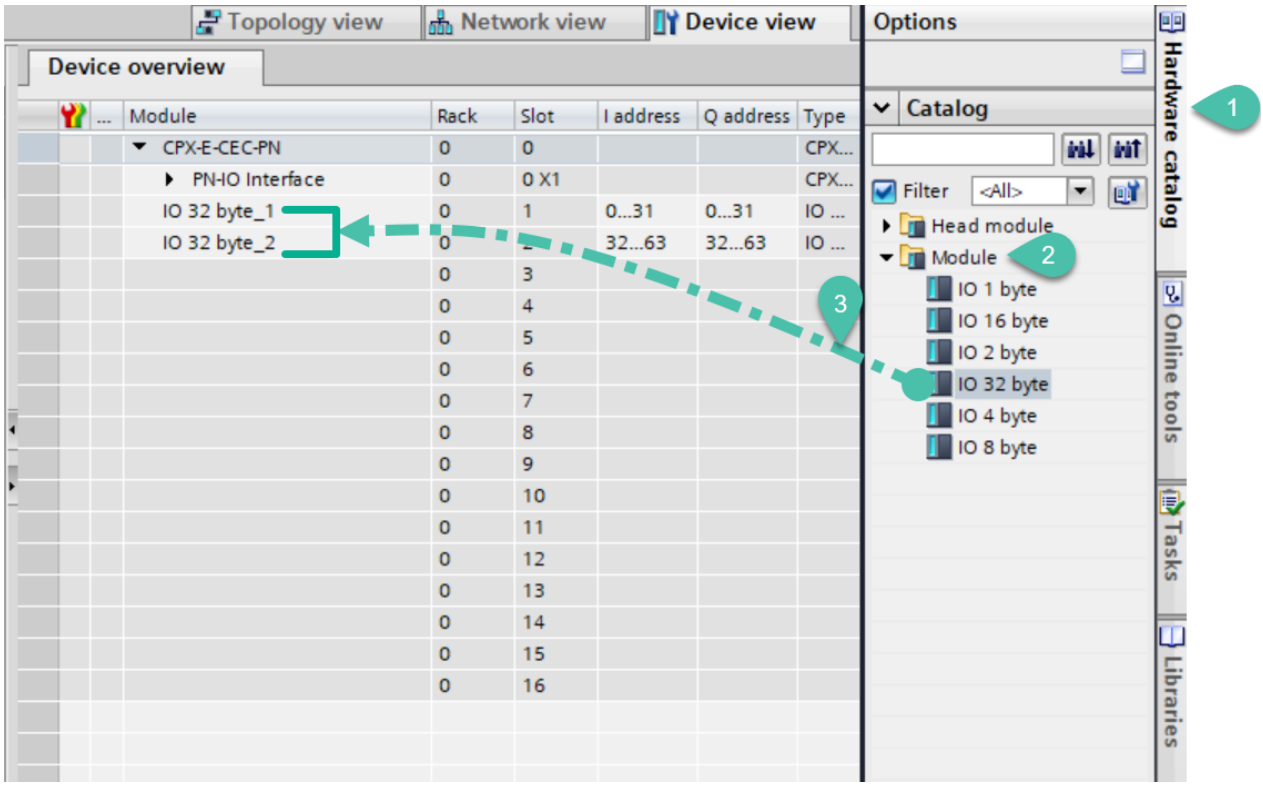


1. Double on **Devices and Networks** to go to network view.
2. Click on **Device view**.

The Device Configuration tab will be displayed as shown below Step 2.

Step 2

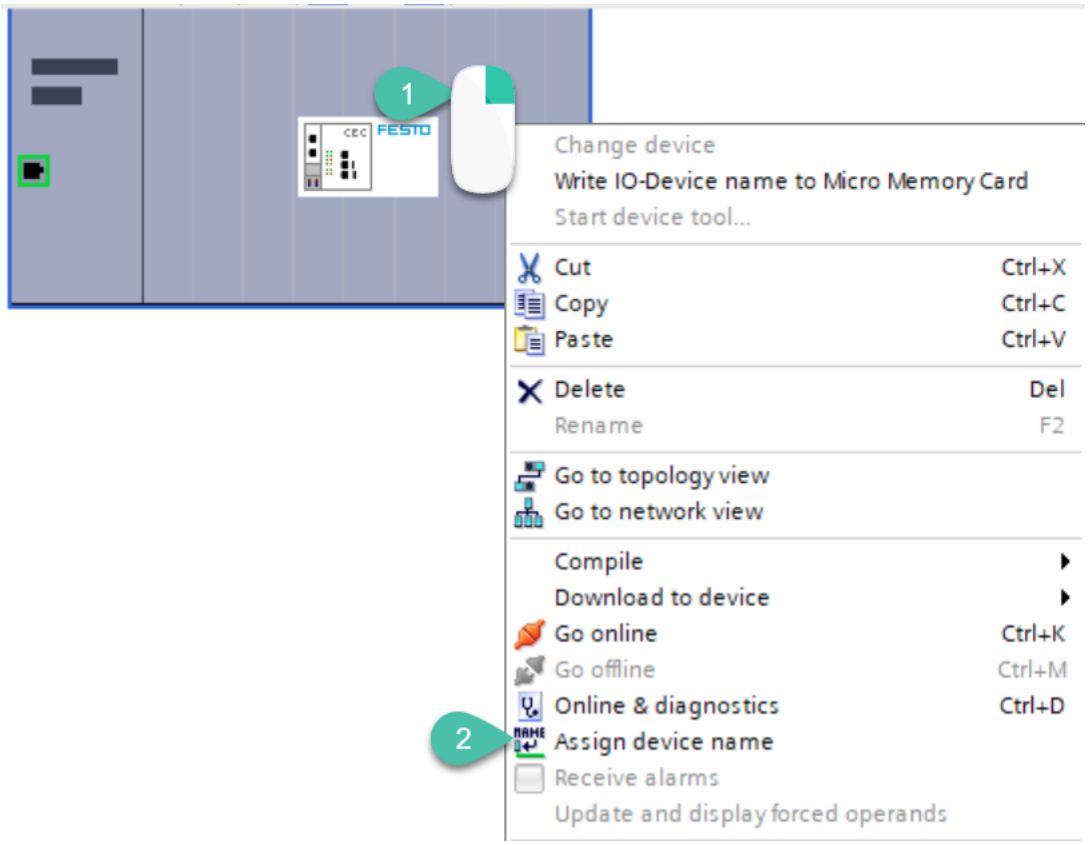




1. Click on **Hardware Catalog** from the Device configuration tab.
2. Expand **Module** to find all IO modules.
3. Drag and drop the IO 32-byte module in slot 1 and slot 2.

2.7 Assigning Profinet Name to CPX-E-CEC-PN module

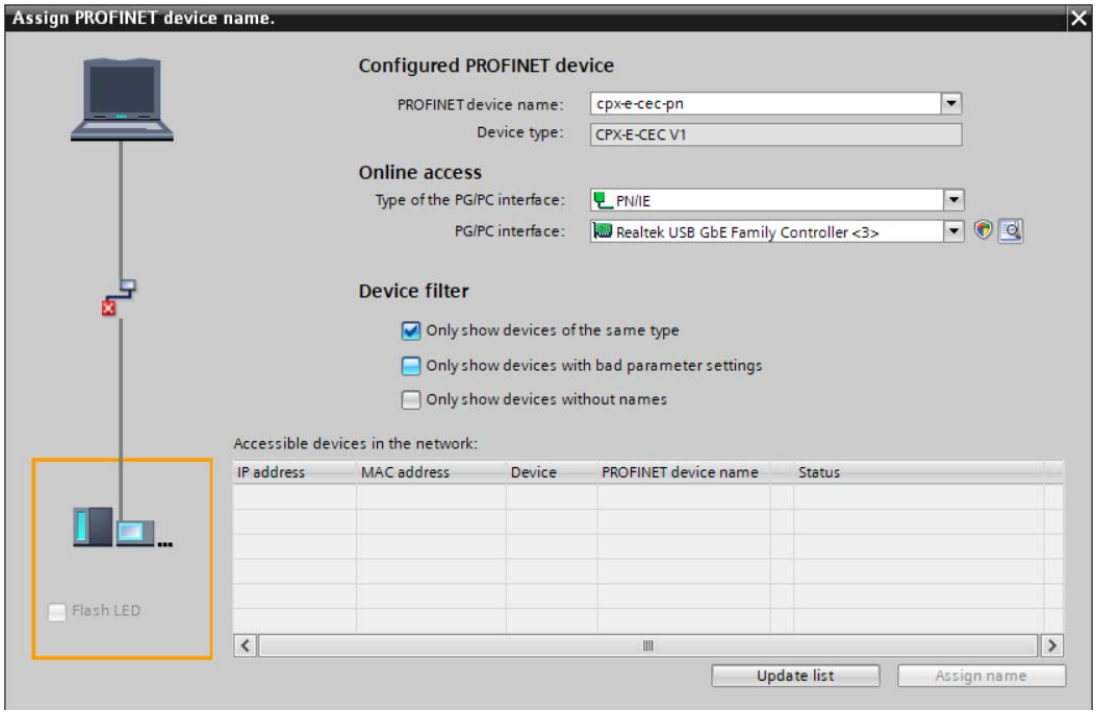
Step 1

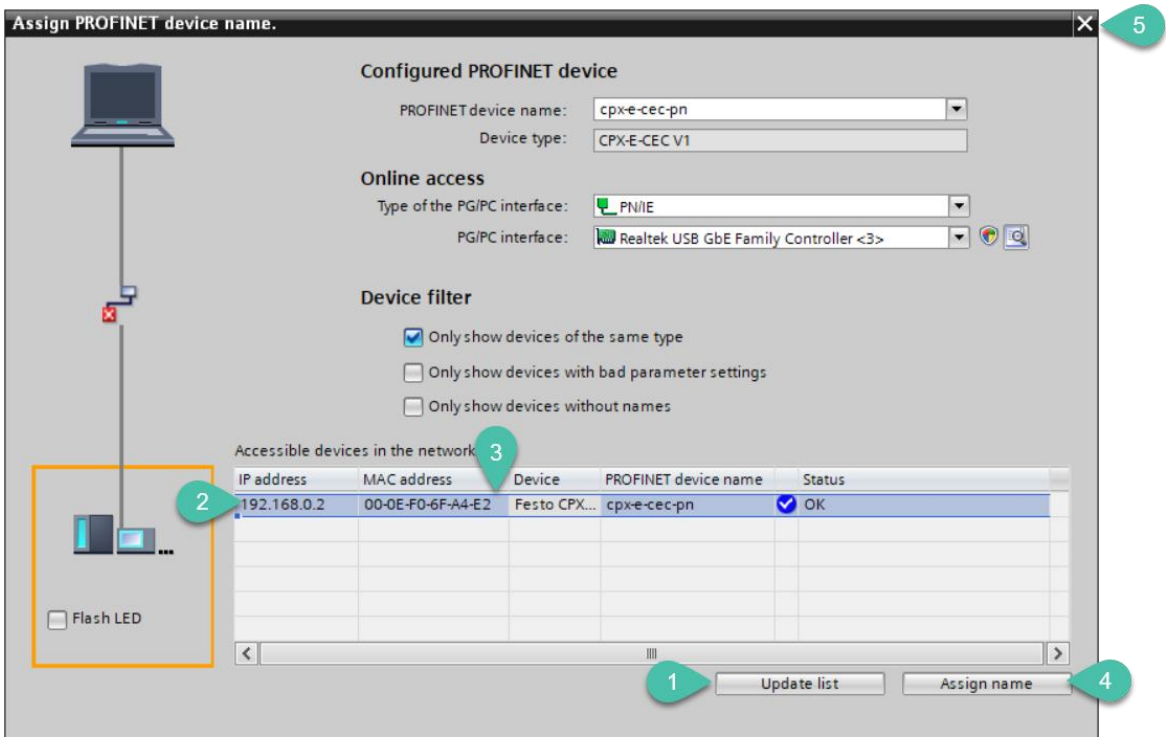


1. Right on **CPX-E-CEC-PN** from the Network view.
2. Click on **the Assign device name** from the right click menu.

The Assign device name tab will be displayed as shown below Step 2.

Step 2



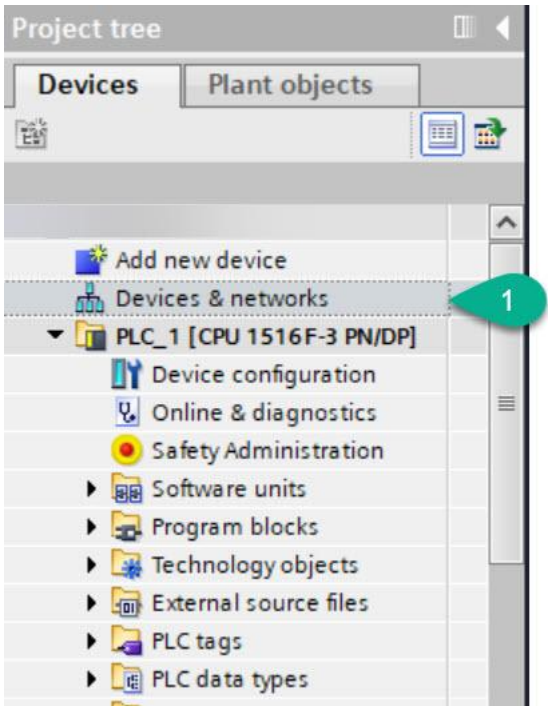


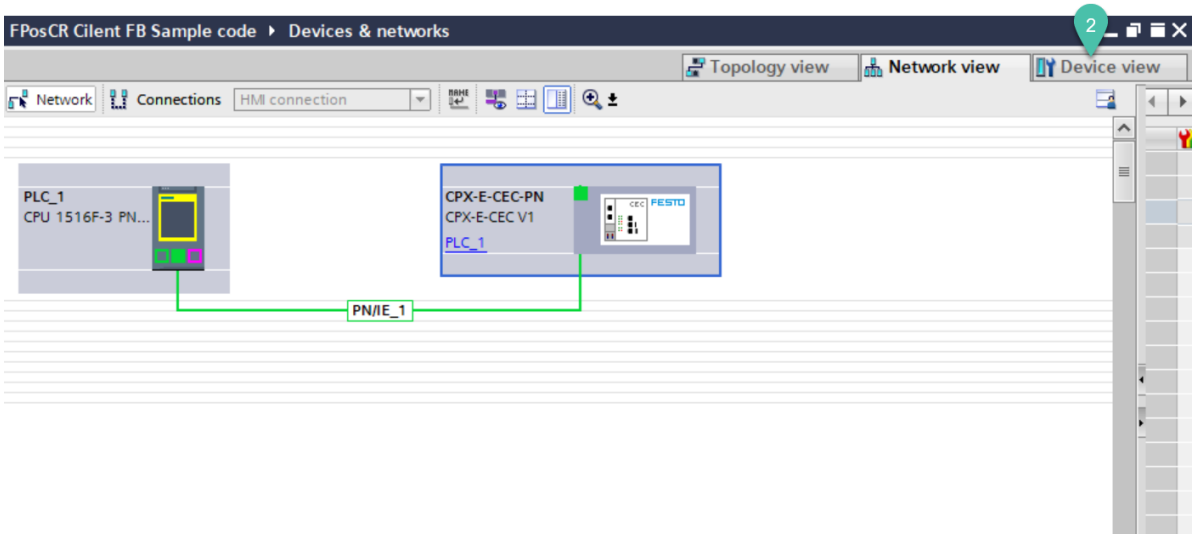
1. Click on **the Update List** to scan the accessible devices in the network.
2. Once the scan is finished, the CPX-E-CEC-PN module will appear in the field.
3. Select the CPX-E-CEC-PN device from an accessible device column.
4. Click on **the Assign name** to assign the device name.
5. Click on **Close** to Finish the Assigning device name.

After completing the steps above, the project should be saved, compiled, and downloaded to the PLC.

2.8 Finding Hardware ID of CPX-E-CEC-PN Module

Step 1

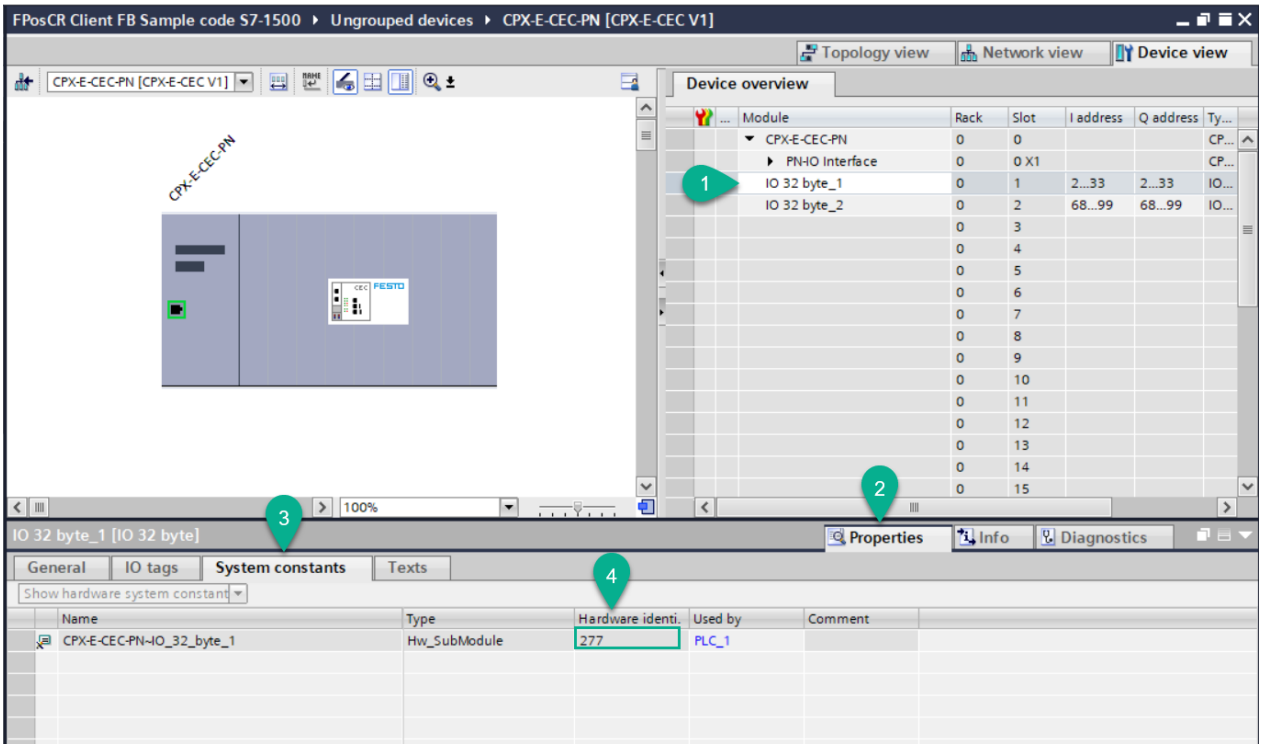




1. Double on **Devices and Networks** to go to network view.
2. Click on **Device view**.

The Device Configuration tab will be displayed as shown below Step 2.

Step 2



1. Click on **the IO 32 byte_1** device from **the Device overview** tab.
2. Click on **Properties** tab to view the **IO 32 byte_1** properties.
3. Click on **System constants** tab to find the Hardware id.
4. Find the **HW_ID** of **the IO 32 byte_1** module from the Hardware identifier column.

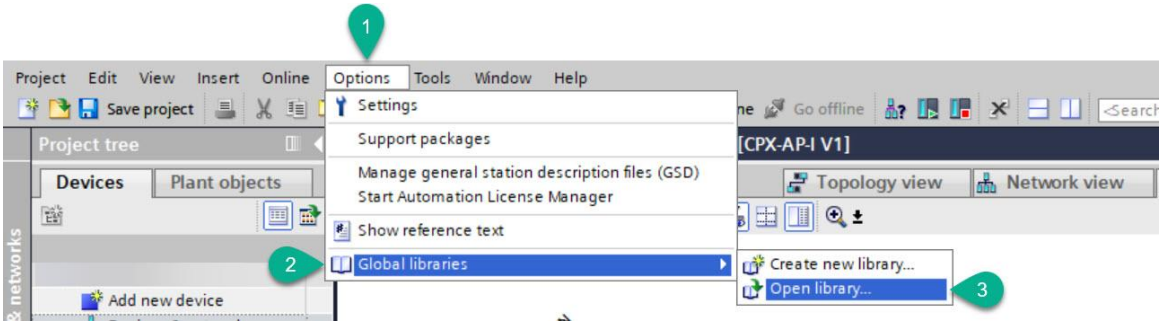
2.9 Adding FPosCR_Host_Client_PN Library to TIA Portal

2.9.1 Import the FPosCR_Host_Client_PN Library into Global Libraries

Step 1

1. Download the **FPosCR_Host_Client_PN** library File from the Festo Support Portal.
2. Save the Downloaded file in an appropriate folder on your PC.
3. Unzip the library file in an appropriate folder.

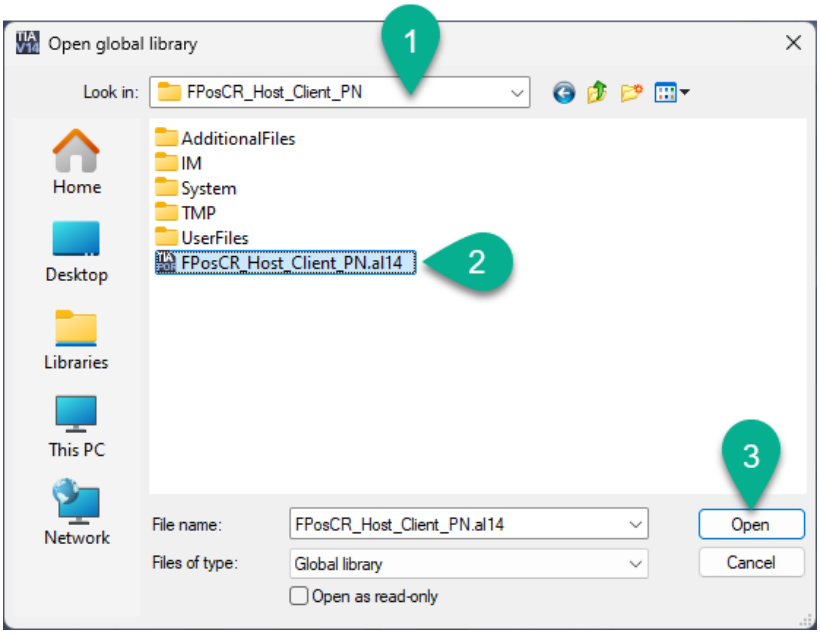
Step 2



1. Click on **the Options** menu from the top menu bar.
2. Expand the **Global libraries to** find the library options.
3. Click on **Open library** to open the library file.

The open global library window shown below Step 3

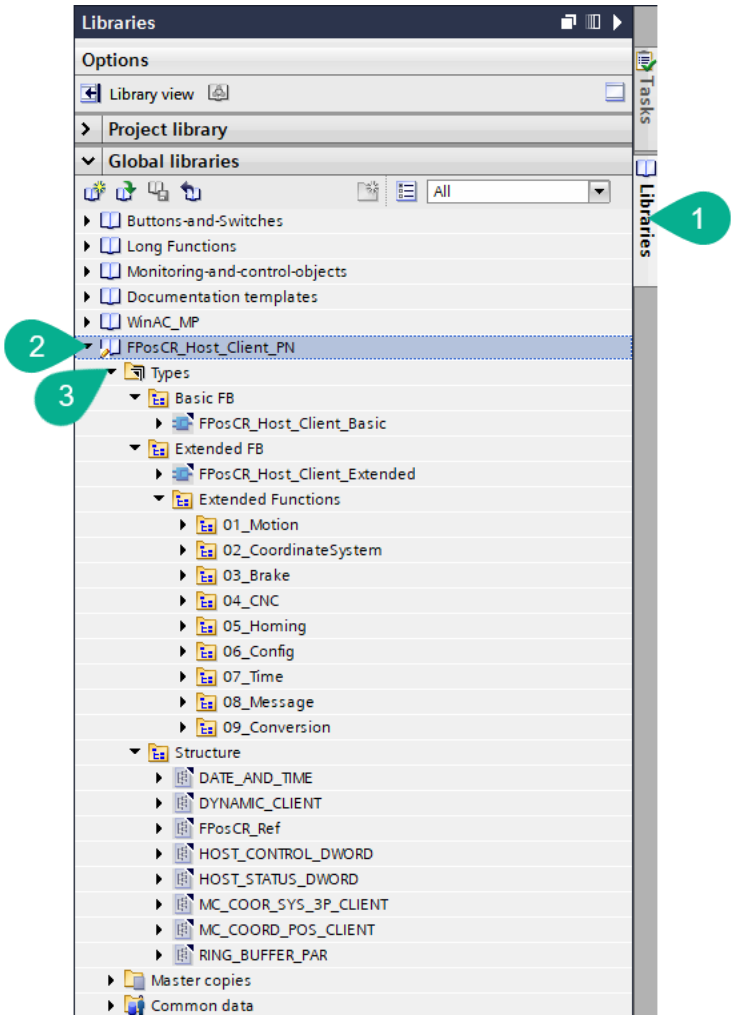
Step 3



1. Choose the path where the library is saved in your system.
2. Select the Library file from the Open Global Library window.
3. Click on **Open** to import the library to Global libraries.

The opened Library shown in below Step 4.

Step 4



1. Click on **Libraries** to view the open library.
2. Expand **FPosCR_Host_Client_PN** to find all Library folders.
3. Expand **the Types** folder to find the library content.

The library has following Blocks and Supporting files.

Function Blocks

1. FPosCR_Host_Client_Basic block
2. FPosCR_Host_Client_Extended block

Function

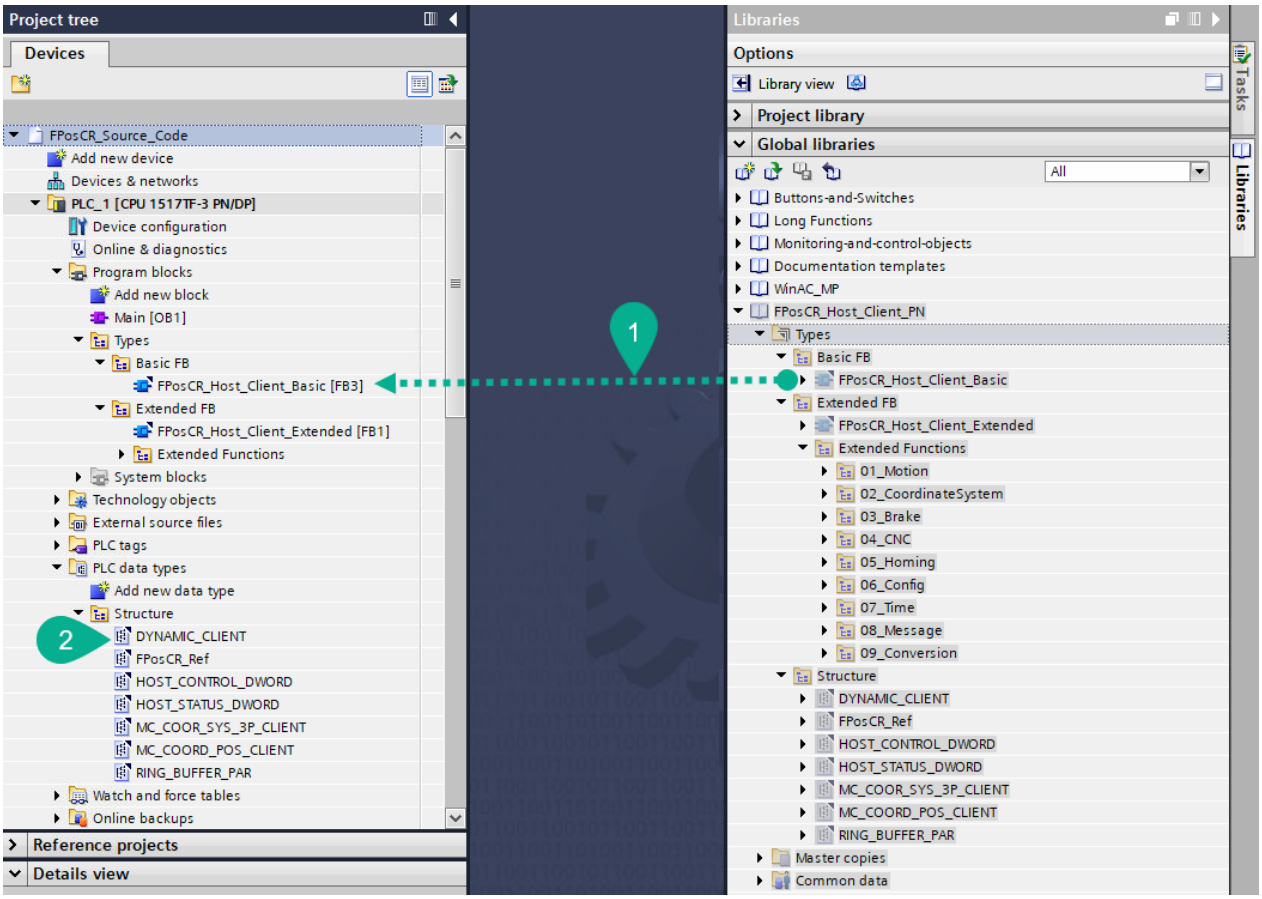
1. Lin
2. LinRel
3. PTP
4. PTP_Rel
5. Circ
6. Circ_rel
7. MoveAxisCartRel
8. BlendingDistance

9. BlendingVel
10. BufferMode
11. Wait
12. Dyn
13. JogSteppingParam
14. ChangeRefSys
15. SetRefSys
16. SetRefSys3P
17. BrakeControl
18. Braketest
19. CNCFile
20. SetPathCorrectionParam
21. ForceGrpHomed
22. HomingSingleAxis
23. DeleteConfig
24. LoadConfig
25. ReadParam
26. SaveParamSet
27. SetParam
28. UpdateConfig
29. ReadTime
30. SetSysTime
31. ReadErrMessage
32. Dword_To_DTL

Structure

- HOST_CONTROL_DWORD
- HOST_STATUS_DWORD
- FPosCR_Ref
- DYNAMIC_CLIENT
- MC_COOR_SYS_3P_CLIENT
- MC_COORD_POS_CLIENT
- RING_BUFFER_PAR
- DATE_AND_TIME

2.9.2 Adding Function blocks and supporting files to program



1. Drag and drop the function block from the FPosCR_Host_Client_PN folder to the Program Blocks in the project.
2. The Supporting Blocks and User data type files come along with the function block.

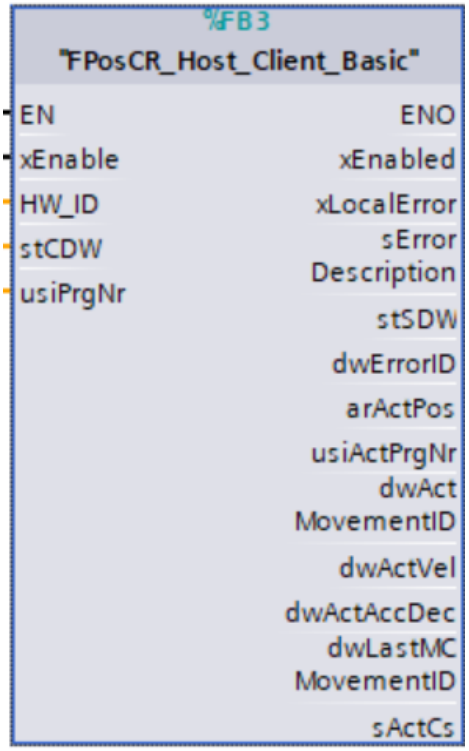
Note

- User need not drag and drop the user data type files, because while drag and drop the function block, the supporting blocks and user data type files automatically come and sit on the corresponding folders.

3 FPosCR Host PN Library Function Block Description

3.1 Introduction to FPosCR_Host_Client_Basic Function Block

The **FPosCR_Host_Client_Basic** function block is used to control and monitor FPosCR function such as Power on/off, Homing, Jogging, stop axis, Start CNC, Start Prg, Monitor actual position, Actual coordinate system, and Actual path velocity etc.



3.1.1 FPosCR_Host_Client_Basic Block Input Variable Details

Input Variable	Data Type	Description
xEnable	BOOL	**TRUE** = Enable block **FALSE** = Disable block
HW_ID	UINT	Hardware identifier of the CPX-E-CEC-PN IO Module.
stCDW	STRUCT of the type **HOST_CONTROL_DWORD**	Control Dword to the Server. Refer Chapter 3.3.1
usiPrgNr	USINT	User Program number.

Table 3.1: FPosCR_Host_Client_Basic Block Input Variable descriptions

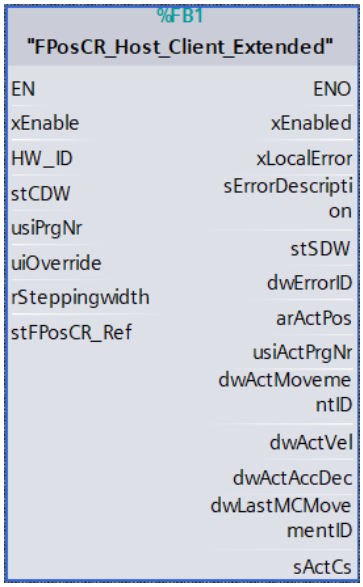
3.1.2 FPosCR_Host_Client_Basic Block Output Variable Details

Output Variable	Data Type	Description
xEnabled	BOOL	Functional Block Enabled.
xLocalError	BOOL	Profinet connection Status.
sErrorDescription	BOOL	Function was executed successfully.
stSDW	STRUCT of the type **HOST_STATUS_DWORD**	Status Dword from the Server. Refer Chapter 3.3.2
dwErrorID	DWORD	Actual Error ID from the server. Refer Chapter 3.3.9
arActPos	ARRAY [1...6] OF REAL	Actual position of X, Y, Z, A, B, C axis.
dwActMovementID	DWORD	Actual movement ID from the server.
dwActVel	DWORD	Actual velocity from the server.
dwActAccDec	DWORD	Actual acceleration and Deceleration from server.
dwLastMCMovementID	DWORD	Last MC Movement ID from server.
sActCs	STRING	Actual Coordinate System from server.

Table 3.2: FPosCR_Host_Client_Basic Block Output Variable descriptions

3.2 Introduction to FPosCR_Host_Client_Extended Function Block

The **FPosCR_Host_Client_Extended** function block inherits all the functionality from **FPosCR_Host_Client_Basic** and includes additional function calls for more advanced features. These function calls can be executed similarly to those found in the FPosCR library, enabling user to interact with a server and manage robotic or CNC operations.



3.2.1 FPosCR_Host_Client_Extended Block In-Out Variables

In-Out Variable	Data Type	Description
stFPosCR_Ref	STRUCT of the type **FPosCR_Ref**	Data structure to share FC data with extended block Refer Chapter 3.3.4

Table 3.3: FPosCR_Host_Client_Extended Block In-Out Variables descriptions

3.2.2 FPosCR_Host_Client_Extended Block Input Variables

Input Variable	Data Type	Description
xEnable	BOOL	**TRUE** = Enable block **FALSE** = Disable block
HW_ID	UINT	Hardware identifier of the CPX-E-CEC-PN IO Module.
stCDW	STRUCT of the type **HOST_CONTROL_DWORD**	Control Dword to the Server. Refer Chapter 3.3.1
usiPrgNr	USINT	User Program number
uiOverride	UINT	Override the velocity in %
rSteppingwidth	REAL	Set Jog step width in mm

Table 3.4: FPosCR_Host_Client_Extended Block Input Variables descriptions

3.2.3 FPosCR_Host_Client_Extended Block Output Variables

Output Variable	Data Type	Description
xEnabled	BOOL	Functional Block Enabled
xLocalError	BOOL	Profinet connection Status
sErrorDescription	BOOL	Function executed successfully
stSDW	STRUCT of the type **HOST_STATUS_DWORD**	Status Dword from the Server. Refer Chapter 3.3.2
dwErrorID	DWORD	Actual Error ID from the server. Refer Chapter 3.3.9
arActPos	ARRAY[1..6] OF REAL	Actual position of X, Y, Z, AA, B, C axis
dwActMovementID	DWORD	Actual movement ID from server
dwActVel	DWORD	Actual velocity from server
dwActAccDec	DWORD	Actual acceleration and Deceleration from server
dwLastMCMovementID	DWORD	Last MC Movement ID from server
sActCs	STRING	Actual Coordinate System from server

Table 3.5: FPosCR_Host_Client_Extended Block Output Variables descriptions

3.3 Structure

3.3.1 HOST_CONTROL_DWORD

Structure Variable	Data Type	Description
xPowerGroup	BOOL	Power group enable
xHomingSystem	BOOL	Home the system
xAckError	BOOL	Acknowledge error and reset drive
xStop_Abort	BOOL	Stop or abort actual movement
xEnableStepping	BOOL	Enable stepping mode
xJogStepPos	BOOL	Jog in positive direction
xJogStepNeg	BOOL	Jog in negative direction
xJogIgnoreSwLimits	BOOL	Ignore SW limits for jogging
xHalt	BOOL	CNC halt
xAckM	BOOL	CNC acknowledge actual M function
xAckAllM	BOOL	CNC acknowledge all M functions
xEnablePathCor	BOOL	CNC enable path correction
xEnableCNCStepmode	BOOL	CNC enable step mode
xNextStep	BOOL	CNC set next step
xBit14	BOOL	Reserved
xTakeControl	BOOL	Take the control
xStartPrg	BOOL	Start the host robotic program
xStartCnc	BOOL	Start the CNC Program
xJog_Axis_X	BOOL	Jog X axis
xJog_Axis_Y	BOOL	Jog Y axis
xJog_Axis_Z	BOOL	Jog Z axis
xJog_Axis_A	BOOL	Jog A axis
xJog_Axis_B	BOOL	Jog B axis
xJog_Axis_C	BOOL	Jog C axis
xKine_Instance1	BOOL	MultiKinematic select number of instance Bit1
xKine_Instance2	BOOL	MultiKinematic select number of instance Bit2
xKine_Instance3	BOOL	MultiKinematic select number of instance Bit3
xBit27	BOOL	Reserved
xBit28	BOOL	Reserved
xBit29	BOOL	Reserved
xBit30	BOOL	Reserved
xLifeBit	BOOL	Life bit for handshaking with server

Table 3.6: HOST_CONTROL_DWORD descriptions

3.3.2 HOST_STATUS_DWORD

Structure Variable	Data Type	Description
xGroupEnabled	BOOL	Power Group enabled
xSystemHomed	BOOL	System homed
xError	BOOL	Error
xSystemReady	BOOL	System ready to enable
xMotionComplete	BOOL	LastMotionDone
xIsMoving	BOOL	CNC group is moving
xBusy	BOOL	Busy
xBit07	BOOL	Reserved
xHalted	BOOL	CNC halt is active
x_HActive	BOOL	H function is active
xM_Active	BOOL	M function is active
xPathCorrActive	BOOL	Path correction is active
xStepModeActive	BOOL	Step mode is active
xBit13	BOOL	Reserved
xBit14	BOOL	Reserved
xHostHasControl	BOOL	Control taken via host
xPrgBusy	BOOL	Program busy: predefined sequences in codesys
xPrgDone	BOOL	Program done: predefined sequences in codesys
xJog_Axis_X	BOOL	Jog-axis X active
xJog_Axis_Y	BOOL	Jog-axis Y active
xJog_Axis_Z	BOOL	Jog-axis Z active
xJog_Axis_A	BOOL	Jog-axis A active
xJog_Axis_B	BOOL	Jog-axis B active
xJog_Axis_C	BOOL	Jog-axis C active
xAct_Kine_Instance1	BOOL	Selected Mult Kinematic instance Bit1
xAct_Kine_Instance2	BOOL	Selected Mult Kinematic instance Bit2
xAct_Kine_Instance3	BOOL	Selected Mult Kinematic instance Bit3
xAct_Cs1	BOOL	Actual coordinate system Bit1
xAct_Cs2	BOOL	Actual coordinate system Bit2
xAct_Cs3	BOOL	Actual coordinate system Bit3
xBit30	BOOL	Reserved
xLifeBit	BOOL	Life bit for handshaking with client

Table 3.7: HOST_STATUS_DWORD descriptions

3.3.3 DYNAMIC_CLIENT

Structure Variable	Data Type
rVel	REAL
rAcc	REAL
rDec	REAL
rJerk	REAL

Table 3.8: DYNAMIC_CLIENT data types

3.3.4 FPosCR_Ref

Structure Variable	Data Type
stSDW	STRUCT of the type **HOST_STATUS_DWORD**
wActualMessageID	WORD
FillPosition	INT
SendPosition	INT
arRingBuffer	ARRAY [0..100] OF **RING_BUFFER_PAR**
newRingBufferElement	STRUCT of the type **RING_BUFFER_PAR**
ClearRingBufferElement	STRUCT of the type **RING_BUFFER_PAR**
inMessageID	WORD
outMessageID	WORD
BufferFull	BOOL
Step	INT
adwUserParameters	ARRAY [0..100] OF DWORD
intMessageID	WORD
arbStatusParameter	ARRAY [0..15] OF BYTE
xInit	BOOL

Table 3.9: FPosCR_Ref data types

3.3.5 MC_COOR_SYS_3P_CLIENT

Structure Variable	Data Type
stOrigin	STRUCT of the type **MC_COOR_POS _CLIENT**
stPointOnX	STRUCT of the type **MC_COOR_POS _CLIENT**
stPointOnXY	STRUCT of the type **MC_COOR_POS _CLIENT**

Table 3.10: MC_COOR_SYS_3P_CLIENT data types

3.3.6 MC_COOR_POS _CLIENT

Structure Variable	Data Type
X	REAL
Y	REAL
Z	REAL
A	REAL
B	REAL
C	REAL

Table 3.11: MC_COOR_POS _CLIENT data types

3.3.7 RING_BUFFER_PAR

Structure Variable	Data Type
FC	BYTE
wMessageID	WORD
arwParameters	ARRAY [0..13] OF WORD
iParameterIdx	INT

Table 3.12: RING_BUFFER_PAR data types

3.3.8 DATE_AND_TIME

Structure Variable	Data Type
YEAR	UINT
MONTH	USINT
DAY	USINT
HOUR	USINT
MINUTE	USINT
SECOND	USINT

Table 3.13: DATE_AND_TIME data types

3.3.9 Error Description

Error Code	Description
16#0000_0000	No Error
16#0000_0001	Time Out Error
16#0000_0002	invalid command
16#0000_0003	Host Connection Error
16#0001_0002	System Power-Up Error
16#0001_0003	System Enable Error
16#0001_0004	System not enabled
16#0001_0007	Drive not Boot up correctly, no connection
16#0001_0008	Drive not ready to power up
16#0001_0009	AxisRef Drive interface not initialized
16#0001_000A	Invalid internal pointer
16#0001_000B	Drive not Available
16#0001_000C	Invalid Case State
16#0001_000D	Command not Supported from Drive
(* Config Error *)	
16#0002_0001	Invalid configuration parameters
16#0002_0002	Invalid Coordinate System
16#0002_0003	Software Limit exceeded
16#0002_0004	HW Configuration was changed, Update of Configuration is needed
16#0002_0005	Delta Kinematic Trafo Error
16#0002_0006	new XY Axis of PCS not perpendicular
16#0002_0007	Selected Kinematic is not supported
16#0002_0008	Read Kinematic Transformation Error
(* JSON CONFIG ERROR *)	
16#0002_1001	Invalid value type in configuration file
16#0002_1002	Config File not available, save current Config to File

Error Code	Description
(* CNC ERROR *)	
16#0003_0001	Invalid Cnc Program
16#0003_0002	Path PreProcess Error
16#0003_0003	Read CNC File Error
16#0003_0004	CNC Out of Software Limits, ADD1 wAxisOutOfLimits ADD2 iSourceLine_No of CNC
16#0003_0005	Interpolator Error
16#0003_0006	Control by Pos Error
16#0003_0007	Invalid Jerk Value
16#0003_0008	Max Dynamic of the Drive exceeded, old Error: ERR_AxisNotSynchronal
16#0003_0009	Missing Position Information in CNC Outque. All existing Axis should have a position Information in the CNC.
16#0003_000A	Error not Acknowledged
16#0003_000B	PathCorrection not available for CNC Outque
16#0003_000C	Invalid or missing CNC Dynamic Parameters
16#0003_000D	Pre-Acknowledge M-Function Error
(* CNC File Edit *)	
16#0003_1001	CNC File Limit, File not fit into Internal Buffer
16#0003_1002	CNC File Write Error
16#0003_1003	CNC File open / read Error
(* ROBOTIC MOTION ERROR*)	
16#0004_0001	Position outside of Software Limits
16#0004_0002	Halt Error
16#0004_0003	Stop Error
16#0004_0004	Wait Time Error
16#0004_0005	Continue Error
16#0004_0006	Interrupt Error
16#0004_0007	SaveContinueData Error
16#0004_0008	Resume After Error not possible
16#0004_0009	Invalid Dynamic
16#0004_000A	Invalid Override
16#0004_000B	Robot Command not accepted

Error Code	Description
16#0004_000D	Get/Poll Robotic Trigger Error
16#0004_000E	Prepare Distance Trigger Error
16#0004_000F	Prepare Relative Pos Trigger Error
16#0004_0010	Prepare Plane Trigger Error
16#0004_100A	AxisGroup Error
16#0004_100B	Jogging Error
16#0004_100C	Linear Absolute Error
16#0004_100D	Linear Relative Error
16#0004_100E	Point to Point Absolute Error
16#0004_100F	Point to Point Relative Error
16#0004_1010	Circular Absolute Error
16#0004_1011	Circular Relative Error
16#0004_2001	Tune CP Kernel Error
(* Homing Error*)	
16#0005_0001	System not referenced
16#0005_0002	Invalid homing order
16#0005_0003	Homing Error
16#0005_0004	Homing time out Error
16#0005_0005	EMCX Homing Sensor Configuration Error
16#0005_0006	Invalid motor orientation for H-/T-Gantry
16#0005_0007	Homing aborted
16#0005_0008	Homing method not supported
16#0005_0009	Safe Homing Offset not successful
16#0005_000A	Homing of another axis has already started
16#0005_000B	Motor Controller not supported
16#0005_000C	Homing Method and homing position not compatible
(* CAN_COE Read Write Error *)	
16#0006_0001	Invalid Can Network ID

Error Code	Description
16#0006_0002	Can read SDO Error
16#0006_0003	Can write SDO Error
16#0006_0004	Can over EtherCAT read SDO Error
16#0006_0005	Can over EtherCAT write SDO Error
(* Brake Error *)	
16#0007_0001	Close Brake Error
16#0007_0002	Open Brake Error
16#0007_0003	Brake input Error
16#0007_0004	Brake test Error
16#0007_0005	Save brake test result Error
16#0007_0006	Invalid brake test Unit Error
16#0007_0007	Axis is virtual
(* File Access *)	
16#0008_0001	File read Error
16#0008_0002	File write Error
16#0008_0003	File OpenRead Error
16#0008_0004	File CloseRead Error
16#0008_0005	File OpenWrite Error
16#0008_0006	File CloseWrite Error
16#0008_0007	Delete File Error
16#0008_0008	Invalid Pointer or Size Error
16#0008_0009	File Flush Error
16#0008_000A	Memory Allocation Error
16#0008_000B	Memory Reallocation Error
16#0008_000C	Memory Move Error
16#0008_000D	Memory Copy Error
(* Drives Error *)	
16#000A_0001	Drives Error
(* Host *)	
16#000C_0001	Invalid jog axis

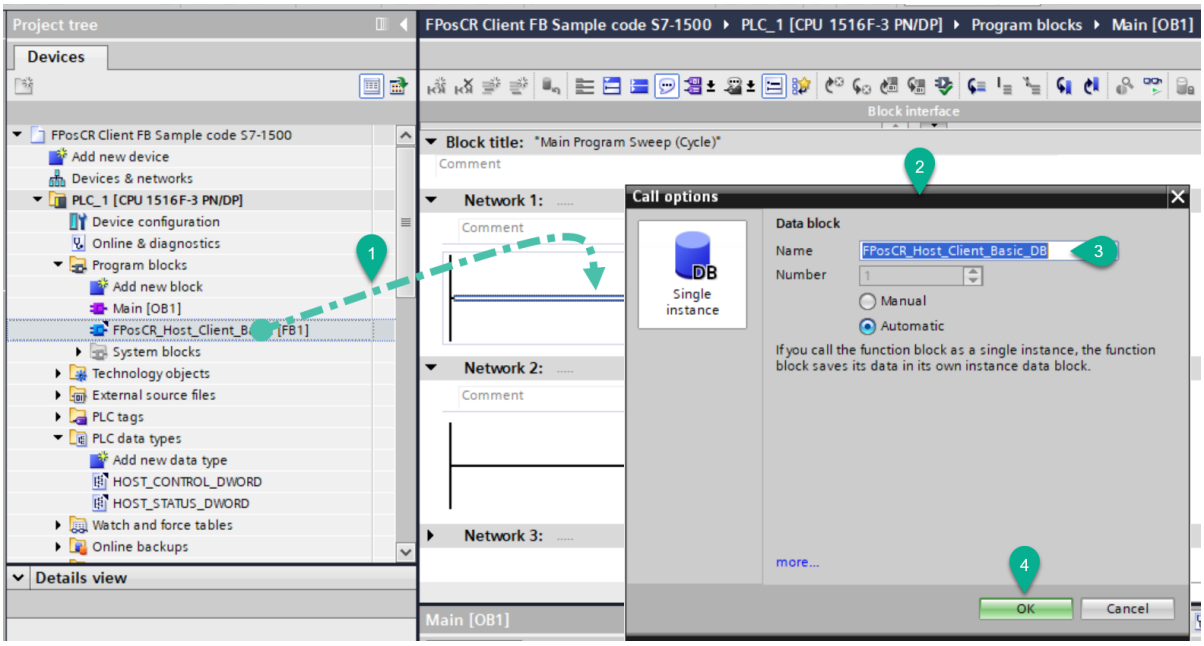
Error Code	Description
(*Warning*)	
16#A000_0001	FPosCR running in Demo Mode
16#A000_0002	Warning Buffer Mode was reset to default
16#A000_0003	Multiple Call of the FB_ROBOT_MOTION within one Cycle.
(*Other Error *)	
16#FFFF_FFFF	undefined Error

Table 3.14: Error Description descriptions

4 Configuration of FPosCR Host Client Function Block in TIA Portal

4.1 Configuring the Function Block Input & Output Interface Tags

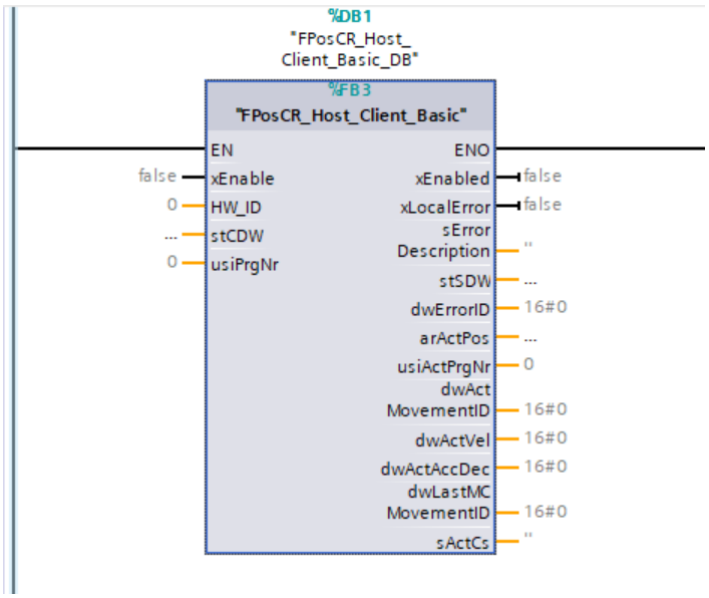
Step 1



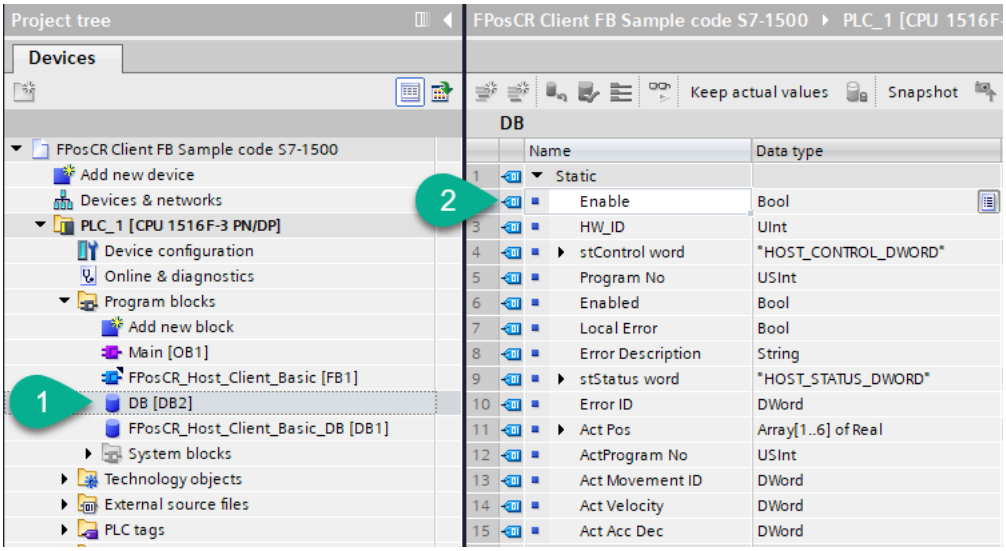
1. Drag & drop the **FPosCR_Host_Client_Basic** block from Program blocks to program area.
2. The **Call options** tab will open.
3. Enter the name of the FB instance on the Name field.
4. Click **OK** to confirm the FB name and close the Call options tab.

The called function block shown in below Step 2

Step 2

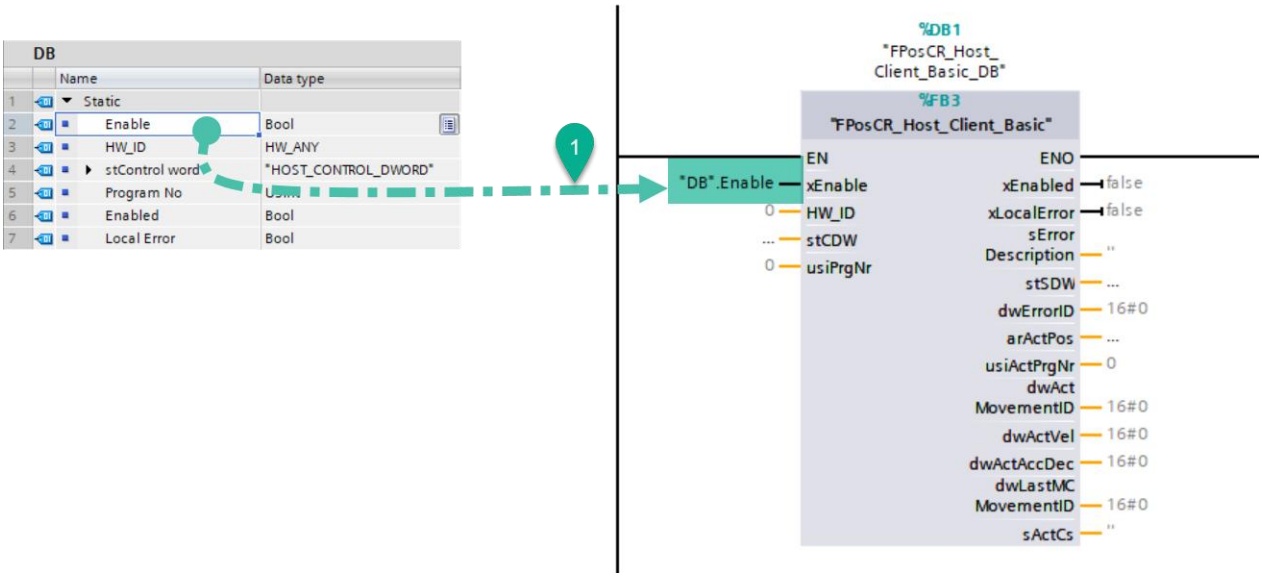


Step 3



1. Create a **Data block** to store the tags for FPosCR_Host_Client_Basic function block.
2. Create all required tags inside the Data blocks and assign the required data type for the tags.

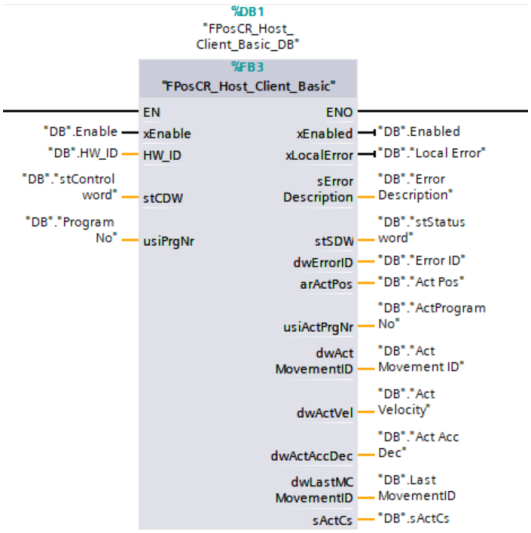
Step 4



1. Assign the created tags to FPosCR_Client_Basic block input and output interface.

Step 5

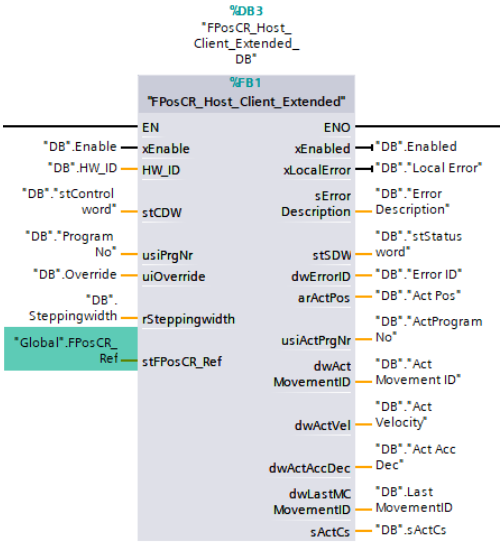
Once IO tags assignment is completed, **FPosCR_Client_Basic** looks as below.



Note

- Follow the same procedures explained above for the assignment of FPosCR_Host_Client_Extended block tags.
- User has to create a variable (FPosCR_Ref) and assign the stFPosCR_Ref PLC data type as variable data type and link it to the FB.

Once IO tags assignment completes. **FPosCR_Host_Client_Extended** block looks as below.



Note

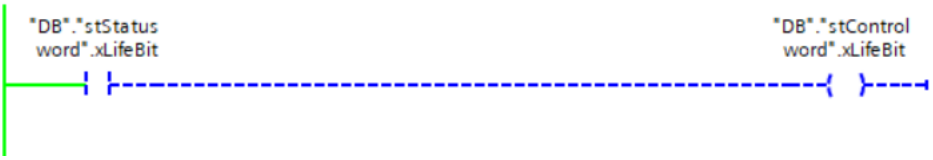
- Users must call one function block at a time in the program.
- If the user application needs only basic functionality of FPosCR, then the user can call the **FPosCR_Host_Client_Basic** block.
- If the user application needs both basic & Extended functionalities of FPosCR, then the user can call the **FPosCR_Host_Client_Extended** block, and Functions.

5 Sample Code Description for FPosCR Host Client Block

5.1 Initialize the function block

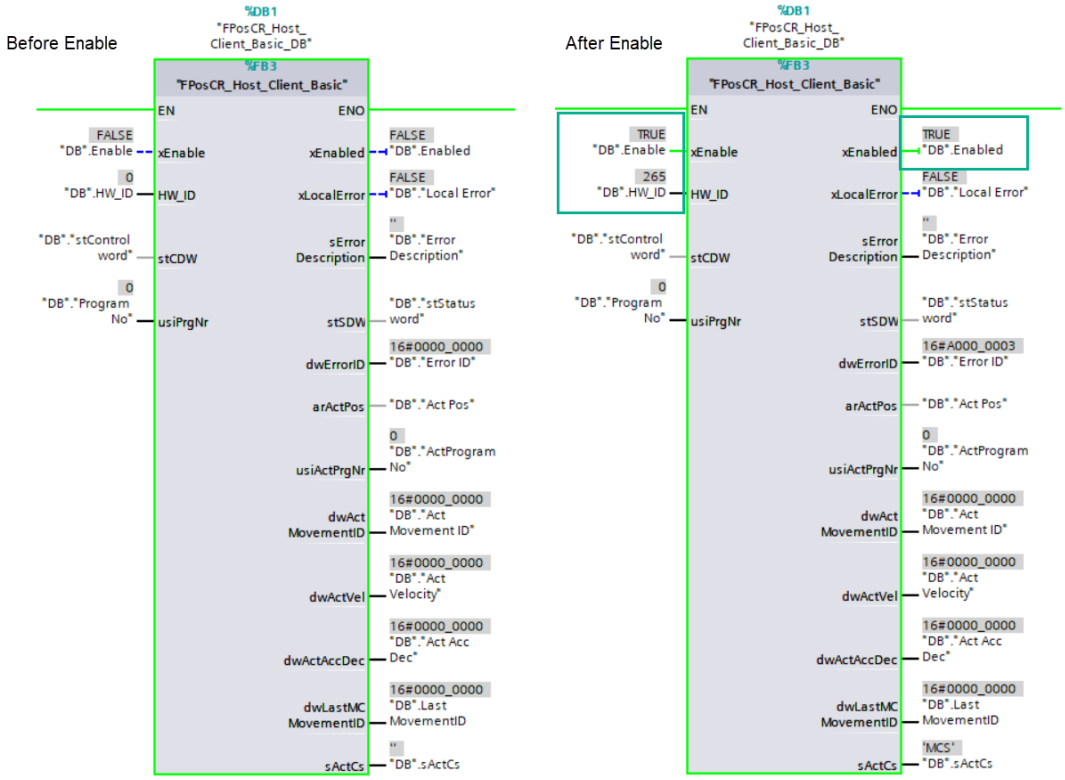
Step 1

> **Note**
> - Before initializing the function block, the user must activate the Life bit to establish a handshake with the server.



- Create the logic to activate the Life bit as shown above.
- Connect the **xLifeBit** status variable input to **xLifeBit** control variable output.

Step 2



1. Enter the FPosCR Server IO module Hardware ID in the **"HW_ID"** input tag.
2. Toggle the **"xEnable"** control tag from false to True to enable the block.
3. Once the block has been successfully enabled, then the monitor tag **"xEnabled"** will be in True state.
4. After enabling the block, users can control and monitor the FPosCR server.

5.2 Control and monitor the FPosCR Server using FPosCR_Host_Client_Basic Block

Users can control and monitor the FPosCR Server using **stCDW** and **stSDW** structure variables.

Name	Data type	...	Monitor...
stControl word	"HOST_CONTROL_DWOR...		
xPowerGroup	Bool	fal:	FALSE
xHomingSystem	Bool	fal:	FALSE
xAckError	Bool	fal:	FALSE
xStop_Abort	Bool	fal:	FALSE
xEnableStepping	Bool	fal:	FALSE
xJogStepPos	Bool	fal:	FALSE
xJogStepNeg	Bool	fal:	FALSE
xJogIgnoreSwLimits	Bool	fal:	FALSE
xHalt	Bool	fal:	FALSE
xAckM	Bool	fal:	FALSE
xAckAllM	Bool	fal:	FALSE
xEnablePathCor	Bool	fal:	FALSE
xEnableCNCStepm...	Bool	fal:	FALSE
xNextStep	Bool	fal:	FALSE
xBit14	Bool	fal:	FALSE
xTakeControl	Bool	fal:	FALSE
xStartPrg	Bool	fal:	FALSE
xStartCnc	Bool	fal:	FALSE
Jog_Axis_X	Bool	fal:	FALSE
Jog_Axis_Y	Bool	fal:	FALSE
Jog_Axis_Z	Bool	fal:	FALSE
Jog_Axis_A	Bool	fal:	FALSE
Jog_Axis_B	Bool	fal:	FALSE
Jog_Axis_C	Bool	fal:	FALSE
Kine_Instance1	Bool	fal:	FALSE
Kine_Instance2	Bool	fal:	FALSE
Kine_Instance3	Bool	fal:	FALSE
xBit27	Bool	fal:	FALSE
xBit28	Bool	fal:	FALSE
xBit29	Bool	fal:	FALSE
xBit30	Bool	fal:	FALSE
xLifeBit	Bool	fal:	TRUE

Name	Data type	...	Monitor...
stStatus word	"HOST_STATUS_DWORD"		
xGroupEnabled	Bool	fal:	FALSE
xSystemHomed	Bool	fal:	FALSE
xError	Bool	fal:	TRUE
xSystemReady	Bool	fal:	FALSE
xMotionComplete	Bool	fal:	FALSE
xIsMoving	Bool	fal:	FALSE
xBusy	Bool	fal:	FALSE
xBit07	Bool	fal:	FALSE
xHalted	Bool	fal:	FALSE
x_HActive	Bool	fal:	FALSE
xM_Active	Bool	fal:	FALSE
xPathCorrActive	Bool	fal:	FALSE
xStepModeActive	Bool	fal:	FALSE
xBit13	Bool	fal:	FALSE
xBit14	Bool	fal:	FALSE
xHostHasControl	Bool	fal:	FALSE
xPrgBusy	Bool	fal:	FALSE
xPrgDone	Bool	fal:	FALSE
Jog_Axis_X	Bool	fal:	FALSE
Jog_Axis_Y	Bool	fal:	FALSE
Jog_Axis_Z	Bool	fal:	FALSE
Jog_Axis_A	Bool	fal:	FALSE
Jog_Axis_B	Bool	fal:	FALSE
Jog_Axis_C	Bool	fal:	FALSE
Act_Kine_Instance1	Bool	fal:	FALSE
Act_Kine_Instance2	Bool	fal:	FALSE
Act_Kine_Instance3	Bool	fal:	FALSE
Act_Cs1	Bool	fal:	TRUE
Act_Cs2	Bool	fal:	FALSE
Act_Cs3	Bool	fal:	FALSE
xBit30	Bool	fal:	FALSE
xLifeBit	Bool	fal:	TRUE

5.2.1 Take control

The **xTakeControl** signal in the Control word is used to request control authority. The **xHostHasControl** signal in the Status word indicates that control authority has been successfully acquired. Only then can commands be sent and received.

1. Take Control:
    1. Set the input " **stCDW.xTakeControl** " to **True**.
    2. Check if the " **stSDW.xHostHasControl** " is **True**.
2. Release Control:
    1. Set the input " **stCDW.xTakeControl** " to **False**.
    2. Check if the output " **stSDW.xHostHasControl** " is **False**.



Note

- Users can directly access the FB variables in the program using the instance name, such as "FPosCR_Host_Client_Basic_DB".stCDW.xTakeControl or "FPosCR_Host_Client_Basic_DB".stSDW.xHostHasControl. Alternatively, if the user assigns the DB variables to the FB, they can reference the DB variables as follows: "DB"."stControl word".xTakeControl or "DB"."stStatus word".xHostHasControl.

5.2.2 Error Monitor and Acknowledge

If there is any error in the server side that can be monitor using status word and acknowledge the error using the control word:

1. Error Monitor:
    1. Check if output **"stSDW. xError "** is **True.**
    2. Check the Error code in output **"diErrorID".**
2. Error acknowledge:
    1. Set input **"stCDW.xAckError"** to **True.**
    2. Check if output **"stSDW. xError"** is **False.**

5.2.3 Power On/Off

Power on/off can be executed using the control word:

1. Precondition:
    1. Output **"stSDW. xSystemReady"** is **True.**
    2. Output **"stSDW. xError"** is **False.**
2. Power On:
    1. Set input **"stCDW.xPowerGroup"** to **True.**
    2. Check if output **"stSDW. xGroupEnabled"** is **True.**
3. Power Off:
    1. Set input **"stCDW.xPowerGroup"** to **False.**
    2. Check if output **"stSDW. xGroupEnabled"** is **False.**

5.2.4 Homing the system

Homing the System Using the Control Word:

1. Precondition:
    1. Output **"stSDW. xSystemReady"** is **True.**
    2. Output **"stSDW. xError "** is **False**.
    3. Output **"stSDW. xGroupEnabled"** is **True.**
2. Home the system:
    1. Set input **"stCDW.xHomingSystem"** to **True.**
    2. Check if output **"stSDW. xSystemHomed"** is **True.**

5.2.5 xStop_Abort

xStop_Abort can be executed with the control word.

Stop movement command:

Falling edge: No action is performed.

Rising edge: Stops the movement.

1. if output **"stSDW. xIsMoving"** is **True.**
2. Set input **"stCDW.xStop_Abort"** to **True.**
3. Check if the output **"stSDW.xIsMoving"** is **False.**
4. Check if output **"stSDW. xMotionComplete"** is **True.**

5.2.6 Jogging

1. Precondition:
    1. Output **"stSDW. xSystemReady"** is **True.**
    2. Output **"stSDW. xError"** is **False.**
    3. Output **"stSDW. xGroupEnabled"** is **True.**
    4. Output **"stSDW. xSystemHomed"** is **True.**
    5. Set input **"stCDW.Jog_Axis_X"** to **True** (for axis X, or the corresponding axis).
    6. Ensure output **"stSDW.Jog_Axis_X"** is **True** (for axis X, or the corresponding axis).
2. Jog the axis:
    1. Set input **"stCDW.xJogStepPos"** or **"stCDW.xJogStepNeg"** to **True.**
    2. Check if output **"stSDW. xGroupEnabled"** is **True.**
    3. During the axis movement: The output **"stSDW.xIsMoving"** will be **True.**
    4. Jogging can be stopped: Set input **"stCDW.xJogStepPos"** or "**stCDW.xJogStepPNeg**" to **False** to stop the movement.



Note

- Jogging is possible even if the axis is outside the software limits or the gantry has not been homed if the input **stCDW.xJogIgnoreSwLimits** is set to **True**.

5.2.7 Halt

1. Precondition:
    1. Output **“stSDW. xSystemReady "** is **True.**
    2. Output **“stSDW. xError "** is **False** .
    3. Output **“stSDW. xGroupEnabled "** is **True.**
    4. Output **“stSDW. xSystemHomed"** is **True.**
    5. CNC is running
2. Halt:
    1. Set input **“stCDW.xHalt"** to **True.**
    2. Check if CNC is stopped or paused with the current dynamics.
    3. Check if output **"stSDW.xHalted"** is **True.**

5.2.8 Acknowledge All M-Functions

1. Precondition:
    1. Output **“stSDW. xSystemReady "** is **True.**
    2. Output **“stSDW. xError "** is **False**.
    3. Output **“stSDW. xGroupEnabled "** is **True.**
    4. Output **“stSDW. xSystemHomed"** is **True.**
    5. The CNC is executed without error, and **stSDW.xMotionComplete** is **True.**
    6. **stSDW.xM_Active** is set to **True.**

2. Acknowledge all M-Function:
    1. Set input **“stCDW. xAckM"** to **True.**
    2. Check if the CNC with M-Function will continue or skip all M-Functions of the CNC.
    3. Check if **stSDW.xM_Active** is set to **False.**

5.2.9 Enable Path Correction

1. Precondition:
    1. Output **“stSDW. xSystemReady"** is **True.**
    2. Output **“stSDW. xError"** is **False**.
    3. Output **“stSDW. xGroupEnabled"** is **True.**
    4. Output **“stSDW. xSystemHomed"** is **True.**
    5. The PathCorrection parameter should be set.
2. Enable Path Correction:
    1. Set input **“stCDW. xEnablePathCor"** to **True.**
    2. Check if CNC is shifted with the given parameters from the Path Correction parameter.

➔ **Note**

- With the Path Correction command, a CNC can be fixed using an XY and an angle offset. These parameters can be configured in the FPosCR visualization commissioning CNC tab under **“Server”.**

5.2.10 Acknowledge Error

1. Precondition:
    1. Output **“stSDW. xSystemReady"** is **True.**
2. Acknowledge Error:
    1. Output **“stSDW.xError"** is **True.**
    2. Set input **“stCDW.xAckError"** to **True.**
    3. Check if **"stSDW.xError"** is set to **False.**

5.2.11 Acknowledge actual M-Function

1. Precondition:
    1. Output **“stSDW. xSystemReady "** is **True.**
    2. Output **“stSDW. xError "** is **False**.
    3. Output **“stSDW. xGroupEnabled "** is **True.**
    4. Output **“stSDW. xSystemHomed"** is **True.**
    5. The CNC is executed without error, and **stSDW.xMotionComplete** is **True.**
    6. **stSDW.xM_Active** is set to **True.**

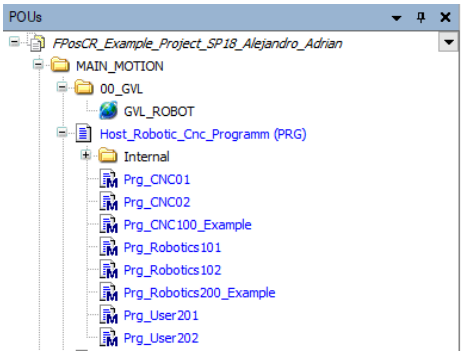
2. Acknowledge M-Function:
    1. Set input **“stCDW. xAckM"** to **True.**
    2. Check if all M-Functions in CNC can be acknowledged.
    3. Check if **stSDW.xM_Active** is set to **False.**

5.2.12 Enable CNC StepMode / Next Step CNC

1. Precondition:
    1. Output **“stSDW. xSystemReady"** is **True.**
    2. Output **“stSDW. xError"** is **False.**
    3. Output **“stSDW. xGroupEnabled"** is **True.**
    4. Output **“stSDW. xSystemHomed"** is **True.**
    5. CNC is executed without error.
2. Enable CNC Stepmode / Next Step CNC:
    1. Set input **"stCDW. xEnableCNCStepmode"** to **True.**
    2. Check if **"stSDW. xStepModeActive"** is set to **True.**
    3. Set input **"xStartCNC"** to **True.**
    3. Set input **"stCDW. xNextStep**" to **True.**
    4. CNC will stop at the next segment transition when the velocity is equal to 0.
        - Each rising edge of **xNextStep** moves the CNC and stops at the next segment transition.
    5. Set input **“stCDW. xEnableCNCStepmode"** to **False.**
        - To deactivate the function
    6. Check if **stSDW. xStepModeActive** is set to **False.**

5.2.13 Start User Prg

The **Host_Robotic_CNC_Program** consists of several methods that can be used by the user:



1. Precondition:
    1. Output **“stSDW. xSystemReady"** is **True.**
    2. Output **“stSDW. xError"** is **False.**
    3. Output **“stSDW. xGroupEnabled"** is **True.**
    4. Output **“stSDW. xSystemHomed"** is **True.**
2. Start User Prg:

    Example of how to call program number 100: **"Prg_CNC100_Example"**

    1. Set the input **"usiPrgNr"** to 100.
    2. Check if **“usiActPrgNr”** is set to 100.
    3. Set input **“stCDW. xStartPrg"** to **True.**
    4. Check if **“stSDW.xPrgBusy”** and **“stSDW.xIsMoving”** are set to **True.**
    5. If the motion is complete, **stSDW.xPrgDone** is set to **True**.
    6. In case of an error, **stSDW.xError** will be set to **True.**

5.2.14 Start CNC

1. Precondition:

    1. Output **“stSDW. xSystemReady "** is **True.**
    2. Output **“stSDW. xError "** is **False.**
    3. Output **“stSDW. xGroupEnabled "** is **True.**
    4. Output **“stSDW. xSystemHomed"** is **True.**

2. Start CNC:

    1. Set input **“usiPrgNr"** to the desired CNC program number.
    2. Check if **“usiActPrgNr”** is shown as set program number.
    3. Set input **“stCDW.xStartCnc**" to **True.**
    4. Check if **“stSDW.xPrgBusy”** and **“stSDW.xIsMoving”** are set to **True.**
    5. If the motion is complete, **stSDW.xPrgDone** is set to **True**.
    6. In case of an error **stSDW.xError** will be set to **True.**

5.2.15 Actual Position

1. Precondition:

    1. Output **“stSDW. xSystemReady "** is **True.**

2. Read Actual position

    arActPos is an array of six positions.

    Example for actual X-position: **arActPos[1].**

5.2.16 Actual Coordinate System

1. Precondition:

    1. Output **“stSDW. xSystemReady "** is **True.**

2. Read the Actual Coordinate System:

The actual coordinate system (CS) can be read from the status word using the following variables:

“stSDW.Act_Cs1, stSDW.Act_C2 and stSDW.Act_C3”

Coordinate System	stSDW.Act_Cs3	stSDW.Act_Cs2	stSDW.Act_Cs1
ACS	0	0	0
MCS	0	0	1
WCS	0	1	0
PCS_1	0	1	1
PCS_2	1	0	0
TCS	1	0	1

Table 5.1: Coordinate System

5.2.17 Actual Movement ID

1. Precondition:

    1. Output **“stSDW. xSystemReady "** is **True.**

2. To read the actual movement ID, you can access it through the output of the **FPosCR_Host_Client_Basic FB** as follows: **dwActMovementID.**

5.2.18 Actual Path Velocity

1. Precondition:

    1. Output **“stSDW. xSystemReady "** is **True.**

2. To read the actual velocity value, you can access it through the output of the **FPosCR_Host_Client_Basic FB** as follows: **dwActVel.**

5.2.19 Actual Path Acceleration and Deceleration

1. Precondition:

    1. Output **“stSDW. xSystemReady "** is **True.**

2. To read the actual acceleration/deceleration value, you can access it through the output of the FPosCR_Host_Client_Basic FB as follows: **dwActAccDec**.

5.2.20 Last MC Movement ID

1. Precondition:

    1. Output **“stSDW. xSystemReady "** is **True.**

2. To read the Last MC Movement ID, you can access it through the output of the FPosCR_Host_Client_Basic FB as follows: **dwLastMCMovementID**.

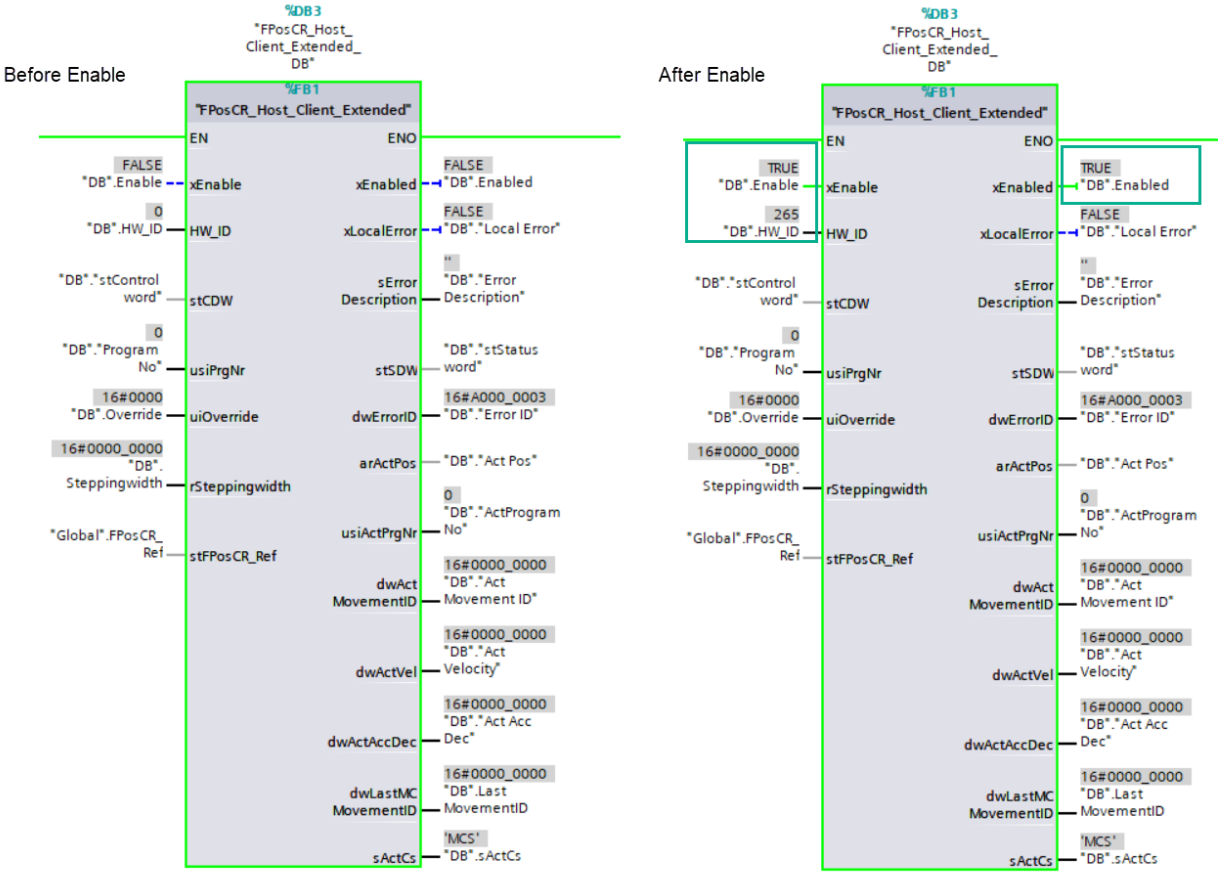
5.2.21 Actual Coordinate System

1. Precondition:

    1. Output **“stSDW. xSystemReady "** is **True.**

2. To read the Actual coordinate system, you can access it through the output of the FPosCR_Host_Client_Basic FB as follows: **sActCs**.

5.3 Control and monitor the FPosCR Server using FPosCR_Host_Client_Extended Block



1. Ensure that the FPosCR_Ref structure variable connected to **"stFPosCR_Ref"**.
2. Enter the FPosCR Server IO module Hardware ID in the "HW**_ID**" input tag.
3. Toggle the **"xEnable"** control tag from false to True to enable the block.
4. Once the block has been successfully enabled, then the monitor tag **"xEnabled"** will be in True state.
5. After enabling the block, users can control and monitor the FPosCR server.

The following function call are supported:

5.3.1 Lin

The Lin command can be executed using the existing Lin function call.

The Lin Function Call Requires Two Input Parameters:

Tag Declaration

stPosition : MC_COORD_POS_CLIENT; // Absolute target position

iBufferIndexForMoveID : INT; // Buffer index where the movement ID will be saved

Precondition:

1. The output stSDW.xSystemReady must be **True**.
2. The output stSDW.xError must be **FALSE**.
3. The output stSDW.xGroupEnabled must be **True**.

Lin Command:

1. **Declare the target position:**
   stPosition:= MC_COORD_POS_CLIENT;
2. **Call the Lin command:**

"Lin"(stPosition:="Global".stPosition,
iBufferIndexForMoveID:=1,
stFPosCR_Ref:="Global". FPosCR_Ref);

Note

- Users should use the same FPosCR_Ref variable which is already created in the Global DB and linked to FPosCR_Host_Client_Extended FB.

Note

If iBufferIndexForMoveID is equal to 0, the movement ID will not be saved.

- The movement ID is saved in Global.FPosCRref.adwUserParameters[1]
- The actual movement ID can be checked with dwActMovementID.

5.3.2 LinRel

The LinRel command can be executed using the existing LinRel function call, similar to the Lin command (Refer Chapter 5.3.1).

The LinRel Function Call Requires Two Input Parameters:

stPosition : MC_COORD_POS_CLIENT; // relative target position
iBufferIndexForMoveID : INT; // Buffer index where the movement ID will be saved

Example of how to call the LinRel command:

"LinRel"(stPosition:="Global".stPosition,
iBufferIndexForMoveID:=2,
stFPosCR_Ref:="Global". FPosCR_Ref);

Note

If iBufferIndexForMoveID is equal to 0, the movement ID will not be saved.

5.3.3 PTP

The PTP command can be executed using the existing PTP function call, similar to the Lin command (Refer Chapter 5.3.1).

The PTP Function Call Requires Two Input Parameters:

stPosition : MC_COORD_POS_CLIENT;// Absolute target position
iBufferIndexForMoveID : INT; // Buffer index where the movement ID will be saved

Example of how to call the PTP command:

```
"PTP"( stPosition:="Global".stPosition,
       iBufferIndexForMoveID:=3,
       stFPosCR_Ref:="Global". FPosCR_Ref);
```

Note

If iBufferIndexForMoveID is equal to 0, the movement ID will not be saved.

5.3.4 PTP_Rel

The PTP_Rel command can be executed using the existing PTP_Rel function call, similar to the Lin command (Refer Chapter 5.3.1).

The PTP_Rel Function Call Requires Two Input Parameters:

stPosition : MC_COORD_POS_CLIENT;// Relative target position

iBufferIndexForMoveID : INT; // Buffer index where the movement ID will be saved

Example of how to call the PTP_Rel command:

```
"PTP_Rel"( stPosition:="Global".stPosition,
       iBufferIndexForMoveID:=3, stFPosCR_Ref:="Global". FPosCR_Ref);
```

Note

If iBufferIndexForMoveID is equal to 0, the movement ID will not be saved.

5.3.5 Circ

The Circ command can be executed using the existing Circ function call, similar to the Lin command (Refer Chapter 5.3.1).

The Circ Function Call Requires Three Input Parameters:

stPosition_Border : MC_COORD_POS_CLIENT;// Absolute target position for the border

stPosition_Target : MC_COORD_POS_CLIENT;// Absolute target position for the target

iBufferIndexForMoveID : INT;// Buffer index where the movement ID will be saved

Example of how to call the Circ command:

```
Circ(stPosition_Border, stPosition_Target, iBufferIndexForMoveID := 5, stFPosCR_Ref := "Global".FPosCR_Ref);
```

Note

If iBufferIndexForMoveID is equal to 0, the movement ID will not be saved.

5.3.6 Circ_rel

The Circ_rel command can be executed using the existing Circ_rel function call, similar to the Lin command (Refer Chapter 5.3.1).

The Circ_rel Function Call requires three input parameters:

stPosition_Border : MC_COORD_POS_CLIENT;// relative target position for the border

stPosition_Target : MC_COORD_POS_CLIENT;// relative target position for the target

iBufferIndexForMoveID : INT;// Buffer index where the movement ID will be saved

Example of how to call the Circ_rel command:

```
Circ_rel(stPosition_Border, stPosition_Target, iBufferIndexForMoveID := 5, stFPosCR_Ref :=
"Global".FPosCR_Ref);
```



Note

If iBufferIndexForMoveID is equal to 0, the movement ID will not be saved.

5.3.7 MoveAxisCartRel

The MoveAxisCartRel command can be executed using the existing MoveAxisCartRel function call, similar to the Lin command (Refer Chapter 5.3.1).

The MoveAxisCartRel Function Call Requires Three Input Parameters:

```
usiAXIS                 : USINT; // Axis 1..6
rPosition               : REAL; // Relative position of the single axis
iBufferIndexForMoveID   : INT; // Buffer index where the movement ID will be saved
```

Example of how to call the MoveAxisCartRel command:

```
"MoveAxisCartRel"( iBufferIndexForMoveID := 7,
                   usiAXIS := 3,
                   rPosition := 3.0,
                   stFPosCR_Ref := "Global".FPosCR_Ref);
```

5.3.8 HomingSingleAxis

The HomingSingleAxis command can be executed using the existing HomingSingleAxis function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

The HomingSingleAxis Function Call Requires Three Input Parameters:

```
iAXIS                   : INT; // Select the Axis 1..6 for homing
iHomingMethod           : INT; // Method for homing the axis
rHomingPosition         : REAL; // Position for the homing action
```

Example of how to call the HomingSingleAxis command:

```
IF "HomingSingleAxis"(iAXIS := 1, iHomingMethod := -18, rHomingPosition := 10.0, stFPosCR_Ref :=
"Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

5.3.9 ForceGroupHomed

The ForceGroupHomed command can be executed using the existing ForceGroupHomed function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

Example of how to call the ForceGrpHomed command:

```
IF "ForceGrpHomed"("Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

5.3.10 JogSteppingParam

The JogSteppingParam command can be executed using the existing JogSteppingParam function call, like the Lin command (Refer Chapter 5.3.1). This function call sets the dynamics to be used during jogging or stepping.

The JogSteppingParam Function Call Requires One Input Parameter:

```
stJogDynamic                : DYNAMIC_CLIENT; // Structure for jogging dynamics parameters
```

Example of how to call the JogSteppingParam command:

```
"JogSteppingParam"(stJogDynamic := "Global".JogDynamic,
                   stFPosCR_Ref := "Global".FPosCR_Ref);
```

5.3.11 Dyn

The Dyn command can be executed using the existing Dyn function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

The Dyn Function Call Requires One Input Parameter:

```
stDynamic : DYNAMIC_CLIENT; // Structure for dynamics parameters
```

Example of how to call the Dyn command:

```
IF "Dyn"(stDynamic := "Global". Dynamic, stFPosCR_Ref := "Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

5.3.12 Wait

The Wait command can be executed using the existing Wait function call, similar to the Lin command (Refer Chapter 5.3.1). This function call sets the wait time to be used during robotic motion.

The Wait Function Call Requires Two Input Parameters:

```
rWaitTime                   : REAL; // Wait time in seconds [s]
iBufferIndexForMoveID       : INT; // Buffer index where the movement ID will be saved
```

Example of how to call the Wait command

```
"Wait"(rWaitTime := 10.0,
        iBufferIndexForMoveID := 8,
        stFPosCR_Ref := "Global".FPosCR_Ref);
```

5.3.13 Blending Distance

The BlendingDistance command can be executed using the existing BlendingDistance function call, similar to the Lin command (Refer Chapter 5.3.1).

The BlendingDistance Function Call Requires One Input Parameter:

rRadius : REAL; // Radius for blending distance

Example of how to call the BlendingDistance command:

```
"BlendingDistance"(rRadius:= 100.0,
                   stFPosCR_Ref:="Global".FPosCR_Ref);
```

5.3.14 Blending Vel

The BlendingVel command can be executed using the existing BlendingVel function call, similar to the Lin command (Refer Chapter 5.3.1).

The BlendingVel Function Call Requires One Input Parameter:

rVelFactor : REAL; // Velocity factor for blending

Example of how to call the BlendingVel command:

```
"BlendingVel"(rVelFactor:=50.0,
              stFPosCR_Ref:="Global".FPosCR_Ref);
```

5.3.15 BufferMode

The BufferMode command can be executed using the existing BufferMode function call, similar to the Lin command (Refer Chapter 5.3.1).

The BufferMode Function Call Requires One Input Parameter:

iBufferMode : INT; // Specifies the buffer mode
"0:Aborting,1:Buffered,2:BlendingLow,3:BlendingPrevious,4:BlendingNext,5:BlendingHigh"

Example of how to call the BufferMode command:

```
"BufferMode"(iBufferMode:=1,
              stFPosCR_Ref:="Global".FPosCR_Ref);
```

5.3.16 ChangeRefSys

The ChangeRefSys command can be executed using the existing ChangeRefSys function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

The ChangeRefSys Function Call Requires One Input Parameter:

```
iCoordSystem                : INT; // PCS_1, PCS_2, or MCS
```

Example of how to call the ChangeRefSys command:

```
IF "ChangeRefSys"(iCoordSystem := 1, stFPosCR_Ref := "Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

5.3.17 SetRefSys

The SetRefSys command can be executed using the existing SetRefSys function call, similar to the Lin command (Refer Chapter 5.3.1).

The SetRefSys Function Call Requires Two Input Parameters:

```
iCoordSystem                : INT; // PCS_1, PCS_2, or MCS
stCoordRef                  : MC_COORD_POS_CLIENT; // Parameter for the CS
```

Example of how to call the SetRefSys command:

```
"SetRefSys"(iCoordSystem := 1,
        stCoordRef := "Global".CoordSys,
        stFPosCR_Ref := "Global".FPosCR_Ref);
```

5.3.18 SetRefSys3P

The SetRefSys3P command can be executed using the existing SetRefSys3P function call, similar to the Lin command (Refer Chapter 5.3.1).

The SetRefSys3P Function Call Requires Two Input Parameters:

```
iCoordSystem                : INT; // PCS_1, PCS_2, or MCS
stCoorSys3P                 : COOR_SYS_3P_CLIENT; // Parameters for the new CS
```

Example of how to call the SetRefSys3P command:

```
"SetRefSys3P"(iCoordSystem := 3,
        stCoordSys3P := "Global".CoordSys3P,
        stFPosCR_Ref := "Global". FPosCR_Ref);
```

5.3.19 DeleteConfig

The DeleteConfig command can be executed using the existing DeleteConfig function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

Example of how to call the DeleteConfig command:

```
IF "DeleteConfig"("Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

5.3.20 LoadConfig

The LoadConfig command can be executed using the existing LoadConfig function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

Example of how to call the LoadConfig command:

```
IF "LoadConfig"("Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

5.3.21 ReadErrMessage

The ReadErrMessage command can be executed using the existing ReadErrMessage function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

The ReadErrMessage Function Call Requires One Input Parameter:

```
iBufferID          : INT; // Buffer index to read the error message
```

Example of how to call the ReadErrMessage command:

```
IF "ReadErrMessage"(iBufferID := BufferID, iOutBufferID => "Global". OutBufferID, dwOutMessageID =>
"Global". MessageID, iAdditional01 => "Global". Additional01, iAdditional02 => "Global". Additional02,
stFPosCR_Ref := "Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;f
```

Message System Overview:

The FPosCR provides a messaging system for error handling that includes a buffer for 16 elements. The iBufferID parameter allows reading the message from the corresponding buffer index.

Outputs of the ReadErrMessage Function Call:

- iOutBufferID: A mirror of iBufferID
- dwOutMessageID: Retrieves the MessageID from ERROR_FPosCR
- iAdditional01: Additional information 1
- iAdditional02: Additional information 2

5.3.22 SetSysTime

The SetSysTime command can be executed using the existing SetSysTime function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

The SetSysTime Function Call Requires One Input Parameter:

```
dtTime                  : DATE_AND_TIME; // Example: dt#1972-03-29-14:26:45
```

Example of how to set the PLC system time:

Call the SetSysTime Function Call:

```
IF "SetSysTime"(dtTime := "Global". Time, stFPosCR_Ref := "Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

5.3.23 ReadTime

The ReadTime command can be executed using the existing ReadTime function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

The ReadTime Function Call Requires One Input Parameter:

```
IBufferIndexForReadedTime       : INT; // Buffer index where the read PLC time will be saved.
```

Example how to call the plc real time clock:

Call the ReadTime command:

```
IF "ReadTime"(iBufferIndexForReadedTime := 1, stFPosCR_Ref := "Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

The read time can be accessed as follows:

```
// Tag Declaration
"Global"."ReadTime" := "Global".FPosCRRef.adwUserParameters[1];
    Dword_To_DTL(Dword:="Global".ReadTime,DTL=>"Global".DateandTime);
```

5.3.24 BrakeControl

The BrakeControl command can be executed using the existing BrakeControl function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

The BrakeControl Function Call Requires Two Inputs Parameter:

```
usiAxis         : USINT; // Axis 1..6
iSetBrake       : INT;  // 1: SMC_BRAKE_OPEN,2: SMC_BRAKE_CLOSE;
```

Example how to open the break for the Axis 2:

Call the BrakeControl command:

```
IF "BrakeControl"(iSetBrake := 1, usiAxis := 1, stFPosCR_Ref := "Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

5.3.25 SetPathCorrectionParam

The SetPathCorrectionParam command can be executed using the existing SetPathCorrectionParam function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

The SetPathCorrectionParam Function Call Requires Three Inputs Parameters:

```
rPhi            : REAL; // Rotation Angle Phi
stStatic_Rot_Pos    : MC_COORD_POS_CLIENT; // Rotation Position of the CNC, a position in the CNC
                                           where the whole CNC should be rotated around the
                                           reference Position is stStatic_Rot_Pos
stDynamic_Rot_Pos   : MC_COORD_POS_CLIENT; // New Position to shift the Rotation Pos, an absolute posi-
                                           tion where the rotated CNC should be shifted.
```

Example how to call the SetPathCorrectionParam command:

Call the SetPathCorrectionParam command:

```
IF "SetPathCorrectionParam"(rPhi := "Global".Phi, stStatic_Rot_Pos := "Global".stRot, stDynamic_Rot_Pos :=
"Global".DyRot, stFPosCR_Ref := "Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

5.3.26 CNCFile

The CNCFile command can be executed using the existing CNCFile function call, similar to the Lin command (Refer Chapter 5.3.1).

The CNCFile Function Call Requires One Input Parameter:

```
sCncFilePath        : STRING; // refers to the location of a CNC file
```

Example of how to call the CNCFile command:

```
“CNCFile”(sCncFilePath:= '_cnc\Example.cnc',stFPosCR_Ref:= "Global".FPosCR_Ref);
```

5.3.27 Braketest

The Braketest command can be executed using the existing Braketest function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

The Braketest Function Call Requires One Input Parameter:

```
iAXIS           :INT ; // Type of robot axis (e.g., AXIS_X)
```

Example of how to call the Braketest command:

```
IF "Braketest"(iAXIS := 1, stFPosCR_Ref := "Global".FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

5.3.28 ReadParam

The ReadParam command can be executed using the existing ReadParam function call, similar to the Lin command (Refer Chapter 5.3.1).

ReadParam function call requires three input parameters. As output, the input parameters will be mirrored, and the read value will be returned.

The function call returns **True** if the reading process was successful.

```
iParamListID : INT;
iParamID : INT;
iParamSubID : INT;
```

Example of how to call the ReadParam command:

```
IF "ReadParam"(iParamListID :=3, iParamID :=0, iParamSubID := 4, iMirrorParamListID =>
"Global".MirrorListID, iMirrorParamID => "Global".MirrorParamID, iMirrorParamSubID =>
"Global".MirrorParamSubID, dwReturnValue => "Global".Data, stFPosCR_Ref := "Global". FPosCR_Ref) THEN
        ;// Do something
END_IF;
```

iParamListID

iParamListID indicates the source of the parameter list. The following parameter lists are supported:

- Configuration parameters from a JSON file
- FPosCR Position list for robotics
- FPosCR Dynamic list for robotics
- User list

The values of the parameter list ID “iParamListID” are to find Refer Chapter 5.3.32

- **Json Configuration parameter:**
  The parameter Index and Parameter Sub index, are needed to read a specific parameter and the description is to find here 0
- **Position list (FPosCR Position list for robotics):**
  The FPosCR library provides a position list. The position list is an array of 255 elements of the structure MC_COORD_REF with fields "X, Y, Z, A, B, and C."

iParamSubID	Description
1	X
2	Y
3	Z
4	A
5	B
6	C

Table 5.2: iParamSubID descriptions

The iParamID indicates which array element needs to be read, and the iParamSubID indicates which axis point from this element needs to be read.
For example, if iParamID = 2 and iParamSubID = 3, the position to be read is FPOsCR.g_stPosList[2].c.Z.

- **Dynamic list (FPosCR Dynamic list for robotics):**
  The FPosCR library provides a dynamic list. The dynamic list is an array of 255 elements of the structure DYNAMIC with fields: lrVel, lrAcc, lrDec, and lrJerk.
  The iParamID indicates which array element needs to be read, and the iParamSubID indicates which dynamic quantity from this element needs to be read.

iParamSubID	Description
1	lrVel (Velocity)
2	lrAcc (Acceleration)
3	lrDec (Deceleration)
4	lrJerk(Jerk)

Table 5.3: iParamSubID dynamic list

For example, if iParamID = 2 and iParamSubID = 3, the position to be read is FPOsCR. g_stDynList [2].lrDec.

- **User list:**
  The FPOsCR library provides a user list. The user list is an array of 255 elements of DWORD.
  The iParamID indicates which array element needs to be read, and the iParamSubID is not used.
  For example, if iParamID = 2, the variable to be read is g_dwUserList[2] is part of **FPosCR** Server.

5.3.29 SetParam

The SetParam command can be executed using the existing SetParam function call, similar to the Lin command (Refer Chapter 5.3.1).

SetParam function call requires four input parameters. As output, the input parameters will be mirrored.

The function call returns **True**, if setting of the parameter was successful.

```
iParamListID        : INT;
iParamID            : INT;
iParamSubID         : INT;
dwSetValue          : DWORD;
```

Example of how to call the SetParam command:

```
IF "SetParam"(iParamListID :=1, iParamID :=22, iParamSubID := 1, dwSetValue := "Global".SetValue,
iMirrorParamListID => "Global".MirrorListID, iMirrorParamID => "Global".MirrorParamID, iMirrorParamSubID
=> "Global".MirrorParamSubID, stFPosCR_Ref := "Global". FPosCR_Ref) THEN
                ;// Do something
END_IF;
```

iParamListID

iParamListID indicates the source of the parameter list. The following parameter lists are supported:

- **Configuration parameters** from a JSON file
- **FPosCR Position list** for robotics
- **FPosCR Dynamic list** for robotics
- **User list**

The values of the parameter list ID **iParamListID** are to find Refer Chapter 5.3.32

- **Json Configuration parameter:**

  The parameter Index and Parameter Sub index, are needed to read a specific parameter and the description is to find Refer Chapter 5.3.33

- **Position list (FPosCR Position list for robotics):**
  The FPOsCR library provides a position list. The position list is an array of 255 elements of the structure MC_COORD_REF with fields "lrX, lrY, lrZ, lrA, lrB, and lrC".

iParamSubID	Description
1	lrX
2	lrY
3	lrZ
4	lrA
5	lrB
6	lrC

Table 5.4: iParamSubID descriptions

The iParamID indicates which array element needs to be read, and the iParamSubID indicates which axis point from this element needs to be read.
For example, if **iParamID = 2** and **iParamSubID = 3**, the position to be read is **FPOsCR.g_stPosList[2].c.z**.

- **Dynamic list (FPosCR Dynamic list for robotics):**
  The FPOsCR library provides a dynamic list. The dynamic list is an array of 255 elements of the structure DYNAMIC with fields: **lrVel, lrAcc, lrDec,** and **lrJerk**.
  The iParamID indicates which array element needs to be read, and the **iParamSubID** indicates which dynamic quantity from this element needs to be read.

iParamSubID	Description
1	lrVel (Velocity)
2	lrAcc (Acceleration)
3	lrDec (Deceleration)
4	lrJerk(Jerk)

Table 5.5: iParamSubID dynamic list

For example, if **iParamID = 2** and **iParamSubID = 3**, the position to be read is **FPOsCR. g_stDynList [2].lrDec**.

- **User list:**
  The FPOsCR library provides a user list. The user list is an array of 255 elements of DWORD.
  The **iParamID** indicates which array element needs to be read, and the **iParamSubID** is not used.
  For example, if **iParamID = 2**, the variable to be read is **FPOsCR.g_dwUserList[2].**

5.3.30 SaveParamSet

The SaveParamSet command can be executed using the existing SaveParamSet function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

Example of how to call the SaveParamSet command:

```
iParamListID                : INT;
```

Example of how to call the SaveParamSet command:

```
IF "SaveParamSet"(iParamListID := 1, stFPosCR_Ref := "Global".FPosCR_Ref) THEN
                ;// Do something
END_IF;
```

iParamListID

iParamListID indicates the source of the parameter list. The following parameter lists are supported:

- **Configuration parameters** from a JSON file
- **FPosCR Position list** for robotics
- **FPosCR Dynamic list** for robotics
- **User list**

The values of the parameter list ID “iParamListID” are to Refer Chapter 5.3.32.

For example, if iParamID = 2, the dynamic list FPOsCR. g_stDynList will be saved.

5.3.31 UpdateConfig

The UpdateConfig command can be executed using the existing UpdateConfig function call, similar to the Lin command (Refer Chapter 5.3.1).

The function call returns **True** if the function call execution was successful.

Example of how to call the UpdateConfig command:

```
"UpdateConfig"("Global".FPosCR_Ref);
```

5.3.32 iParamListID Values

Parameter List ID	Source Parameter
1	Json Configuration parameter
2	Position list
3	Dynamic list
4	User list

Table 5.6: iParamListID Values parameters

5.3.33 iParamID and iParamSubID

Description	Parameter ID	SubID	Data Type
stDynamic.arstMaxAxisDyn[1]	1	1:lrVel 2:lrAcc 3:lrDec 4:lrJerk	REAL
stDynamic.arstMaxAxisDyn[2]	2	1:lrVel 2:lrAcc 3:lrDec 4:lrJerk	REAL
stDynamic.arstMaxAxisDyn[3]	3	1:lrVel 2:lrAcc 3:lrDec 4:lrJerk	REAL
stDynamic.arstMaxAxisDyn[4]	4	1:lrVel 2:lrAcc 3:lrDec 4:lrJerk	REAL
stDynamic.arstMaxAxisDyn[5]	5	1:lrVel 2:lrAcc 3:lrDec 4:lrJerk	REAL
stDynamic.arstMaxAxisDyn[6]	6	1:lrVel 2:lrAcc 3:lrDec 4:lrJerk	REAL
stDynamic.lrQuickStopDec	7	N/A	REAL
stDynamic.lrQuickStopJerk	8	N/A	REAL
stHoming.tHomingTimeOut	9	N/A	DINT
stHoming.arHomingMethod[1]	10	N/A	DINT
stHoming.arHomingMethod[2]	11	N/A	DINT
stHoming.arHomingMethod[3]	12	N/A	DINT
stHoming.arHomingMethod[4]	13	N/A	DINT
stHoming.arHomingMethod[5]	14	N/A	DINT
stHoming.arHomingMethod[6]	15	N/A	DINT
stHoming.arrHomingPosition[1]	16	N/A	REAL
stHoming.arrHomingPosition[2]	17	N/A	REAL
stHoming.arrHomingPosition[3]	18	N/A	REAL
stHoming.arrHomingPosition[4]	19	N/A	REAL

Description	Parameter ID	SubID	Data Type
stHoming.arrHomingPosition[5]	20	N/A	REAL
stHoming.arrHomingPosition[6]	21	N/A	REAL
stHoming.stParkPos	22	1:X 2:Y 3:Z 4:A 5:B 6:C	REAL
stHoming.stParkDynamic	23	1:lrVel 2:lrAcc 3:lrDec 4:lrJerk	REAL
stHoming.xMoveToParkPos	24	N/A	DINT
stHoming.arEmcxHomingSensor[1].xEnableHomingSensor	25	N/A	DINT
stHoming.arEmcxHomingSensor[2].xEnableHomingSensor	26	N/A	DINT
stHoming.arEmcxHomingSensor[3].xEnableHomingSensor	27	N/A	DINT
stHoming.arEmcxHomingSensor[4].xEnableHomingSensor	28	N/A	DINT
stHoming.arEmcxHomingSensor[5].xEnableHomingSensor	29	N/A	DINT
stHoming.arEmcxHomingSensor[6].xEnableHomingSensor	30	N/A	DINT
stHoming.arEmcxHomingSensor[1].SensorTypeAtDI3	31	N/A	BYTE
stHoming.arEmcxHomingSensor[2].SensorTypeAtDI3	32	N/A	BYTE
stHoming.arEmcxHomingSensor[3].SensorTypeAtDI3	33	N/A	BYTE
stHoming.arEmcxHomingSensor[4].SensorTypeAtDI3	34	N/A	BYTE
stHoming.arEmcxHomingSensor[5].SensorTypeAtDI3	35	N/A	BYTE
stHoming.arEmcxHomingSensor[6].SensorTypeAtDI3	36	N/A	BYTE
stHoming.arEmcxHomingSensor[1].SensorTypeAtDI4	37	N/A	BYTE

Description	Parameter ID	SubID	Data Type
stHoming.arEmcxHomingSensor[2].SensorTypeAtDI4	38	N/A	BYTE
stHoming.arEmcxHomingSensor[3].SensorTypeAtDI4	39	N/A	BYTE
stHoming.arEmcxHomingSensor[4].SensorTypeAtDI4	40	N/A	BYTE
stHoming.arEmcxHomingSensor[5].SensorTypeAtDI4	41	N/A	BYTE
stHoming.arEmcxHomingSensor[6].SensorTypeAtDI4	42	N/A	BYTE
stHW_Config.eH_GantryBeltLock	43	N/A	DINT
stHW_Config.arstSWLimit[1].dlower	44	N/A	REAL
stHW_Config.arstSWLimit[2].dlower	45	N/A	REAL
stHW_Config.arstSWLimit[3].dlower	46	N/A	REAL
stHW_Config.arstSWLimit[4].dlower	47	N/A	REAL
stHW_Config.arstSWLimit[5].dlower	48	N/A	REAL
stHW_Config.arstSWLimit[6].dlower	49	N/A	REAL
stHW_Config.arstSWLimit[1].dUpper	50	N/A	REAL
stHW_Config.arstSWLimit[2].dUpper	51	N/A	REAL
stHW_Config.arstSWLimit[3].dUpper	52	N/A	REAL
stHW_Config.arstSWLimit[4].dUpper	53	N/A	REAL
stHW_Config.arstSWLimit[5].dUpper	54	N/A	REAL
stHW_Config.arstSWLimit[6].dUpper	55	N/A	REAL
stHW_Config.arstSWLimit[1].xUnlimited	56	N/A	DINT
stHW_Config.arstSWLimit[2].xUnlimited	57	N/A	DINT
stHW_Config.arstSWLimit[3].xUnlimited	58	N/A	DINT
stHW_Config.arstSWLimit[4].xUnlimited	59	N/A	DINT
stHW_Config.arstSWLimit[5].xUnlimited	60	N/A	DINT
stHW_Config.arstSWLimit[6].xUnlimited	61	N/A	DINT
stHW_Config.eMotorOrientation	62	N/A	DINT
	63	N/A	

Description	Parameter ID	SubID	Data Type
stCs_Config.stMCS.stOrigin	64	1: lrX 2: lrY 3: lrZ 4: lrA 5: lrB 6: lrC	REAL
stCs_Config.stMCS.stPointOnX	65	1: lrX 2: lrY 3: lrZ 4: lrA 5: lrB 6: lrC	REAL
stCs_Config.stMCS.stPointOnXY	66	1: lrX 2: lrY 3: lrZ 4: lrA 5: lrB 6: lrC	REAL
stCs_Config.stPCS1.stOrigin	67	1: lrX 2: lrY 3: lrZ 4: lrA 5: lrB 6: lrC	REAL
stCs_Config.stPCS1.stPointOnX	68	1: lrX 2: lrY 3: lrZ 4: lrA 5: lrB 6: lrC	REAL
stCs_Config.stPCS1.stPointOnXY	69	1: lrX 2: lrY 3: lrZ 4: lrA 5: lrB 6: lrC	REAL
stCs_Config.stPCS2.stOrigin	70	1: lrX 2: lrY 3: lrZ 4: lrA 5: lrB 6: lrC	REAL

Description	Parameter ID	SubID	Data Type
stCs_Config.stPCS2.stPointOnX	71	1: lrX 2: lrY 3: lrZ 4: lrA 5: lrB 6: lrC	REAL
stCs_Config.stPCS2.stPointOnXY	72	1: lrX 2: lrY 3: lrZ 4: lrA 5: lrB 6: lrC	REAL
stCNC.stDefaultDynCNC	73	1:lrVel 2:lrAcc 3:lrDec 4:lrJerk	REAL
stCNC.stDefaultDynFFCNCG0	74	1:lrVel 2:lrAcc 3:lrDec 4:lrJerk	REAL
stCNC.lrPathAngleTolerance	75	N/A	REAL
stCNC.lrMaxJerk	76	N/A	REAL
stCNC.stGapDyn	77	1:lrVel 2:lrAcc 3:lrDec 4:lrJerk	REAL
stCNC.eIpoVelMode	78	N/A	DINT
stCNC.xUseUvwAxes	79	N/A	DINT
stCnc.lrXY_Compensation	80	N/A	REAL
stCNC.stAddAxesConfig.xAbcAxesAsOrientationAxes	81	N/A	DINT
stCNC.stAddAxesConfig.unToolLength.ar0_2	82	1:ar0_2[0] 2:ar0_2[1] 3:ar0_2[2]	REAL
stCNC.stAddAxesConfig.eOriConv	83	N/A	DINT

Table 5.7: iParamID and iParamSubID data types