Application Note

YJKP with Keyence KV-X PLC through EtherNet/IP

This Application Note describes how to use and integrate the Festo's host function blocks for the YJKP-Servo Press Kit into Keyence KV STUDIO projects with KV-X PLCs over an EtherNet/IP network.

YJKP

100849

Title ..... YJKP with Keyence KV-X PLC through EtherNet/IP
Version 1.10
Document no. 100849
Original ..... en
Author ..... Festo

Last saved 01.09.2026

General information:

This document is intended for qualified, trained and instructed professionals. The data provided in this document are 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 Hardware Configuration 10

2.1 Servo Press Kit YJKP Overview 10

2.1.1 Device Overview 10

2.1.2 Product Overview 11

2.2 Servo Press Kit YJKP System Configuration 12

2.2.1 Assigning a Static IP Address 12

2.2.2 Configuring the YJKP System Settings 14

3 Software Configuration 18

3.1 Creating a New Project and Adding Controller in KV STUDIO 18

3.2 Controller Configuration and Setup 19

3.2.1 Built-in EtherNet/IP Port Configuration 19

3.2.2 PLC Communication Setup 20

3.3 YJKP Device Configuration 23

3.3.1 Adding YJKP Device to EtherNet/IP Network 23

3.3.2 Adapter Connection Settings 25

3.3.3 Creating Variables Used for YJKP Device Connection 27

3.4 Adding Festo_YJKP_Client_EIP Library into KV STUDIO Project 32

4 Festo_YJKP_Client_EIP Library Function Blocks Description 35

4.1 FB_ControlChannel 35

4.1.1 FB_ControlChannel Block In-Out Variables 36

4.1.2 FB_ControlChannel Block Input Variables 36

4.1.3 FB_ControlChannel Block Output Variables 39

4.2 FB_ParameterChannel 42

4.2.1 FB_ParameterChannel Block In-Out Variables 42

4.2.2 FB_ParameterChannel Block Input Variables 42

4.2.3 FB_ParameterChannel Block Output Variables 43

4.3 FB_Byte_To_Udint 44

4.3.1 FB_Byte_To_Udint Block Input Variables 44

4.3.2 FB_Byte_To_Uint Block Output Variables 44

4.4 FB_Byte_To_Uint 44

4.4.1 FB_Byte_To_Uint Block Input Variables 44

4.4.2 FB_Byte_To_Uint Block Output Variables 44

4.5 FB_ Udint_To_Byte 45
4.5.1 FB_Udint_To_Byte Block Input Variables 45
4.5.2 FB_ Udint_To_Byte Block Output Variables 45
4.6 FB_Uint_To_Byte 45
4.6.1 FB_Uint_To_Byte Block Input Variables 46
4.6.2 FB_Uint_To_Byte Block Output Variables 46
4.7 Union 46
4.7.1 Udint_To_Array 46
4.8 Enumerations 46
4.8.1 NOK Reason 47
4.8.2 Acknowledge 47
4.8.3 Data Type 47
4.8.4 Error Table 48
5 Configuration of YJKP Client Function Blocks in KV STUDIO Software 48
6 Sample Code for Festo_YJKP_Client_EIP Library Function Block 56
6.1 Sample Code for FB_ControlChannel Block 56
6.1.1 Enabling the YJKP System 56
6.1.2 Sample Code for Homing the YJKP System 57
6.1.3 Sample Code for Start Press Process 58
6.1.4 Sample Code for Jog Movement 61
6.1.5 Sample Code for Absolute Movement 62
6.1.6 Sample Code for Relative Movement 63
6.2 Sample Code for FB_ParameterChannel 64
6.2.1 Write Single Variable 64
6.2.2 Read Single Variable 66
6.2.3 Write Multiple Variables 68
6.2.4 Read Multiple Variables 71
7 Procedure to Operate YJKP Client Function Blocks 73
7.1 Procedure of Press Control Functions through FB_ControlChannel 73
7.1.1 Load Hardware Configuration 73
7.1.2 Activate Logging (For all logging functions the same) 73
7.1.3 System Reset 73
7.1.4 Reset Statistic 74
7.1.5 Enable System 74
7.1.6 Disable System 74
7.1.7 Tare Force Sensor 75
7.1.8 Add Offset to the Force Sensor 75
7.1.9 Homing 75
7.1.10 Jog 76
7.1.11 Move Absolute 76
7.1.12 Move Relative 77
7.1.13 Load Program 78
7.1.14 Start Press Process 78

7.1.15 Record Reference Curve 79
7.1.16 Step Mode 79
7.1.17 Quit System Error 80
7.1.18 Function Block Error Reaction 80
7.2 Procedure of YJKP Parameter Read / Write through FB_ParameterChannel 81
7.2.1 Write Variable (Single) 81
7.2.2 Read Variable (Single) 81
7.2.3 Write Variable (Multi) 81
7.2.4 Read Variable (Multi) 82
7.2.5 Function Block Error Reaction 82

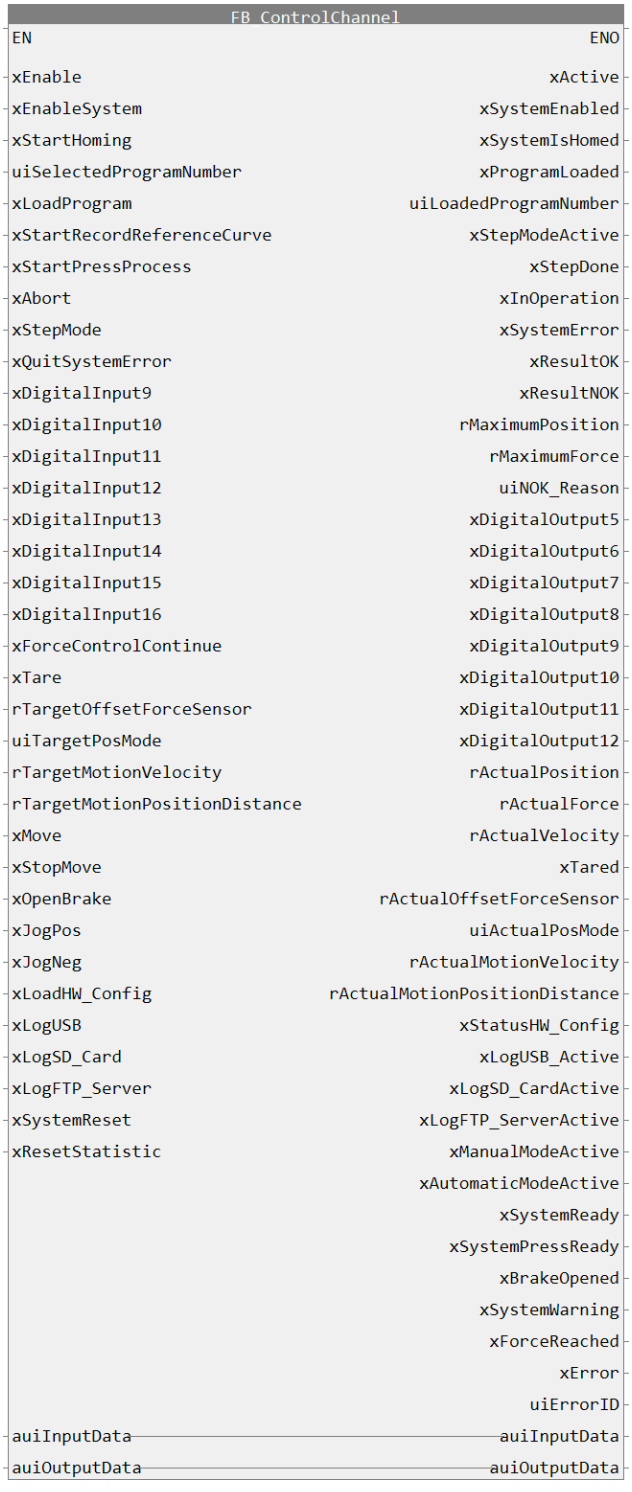
1 Introduction

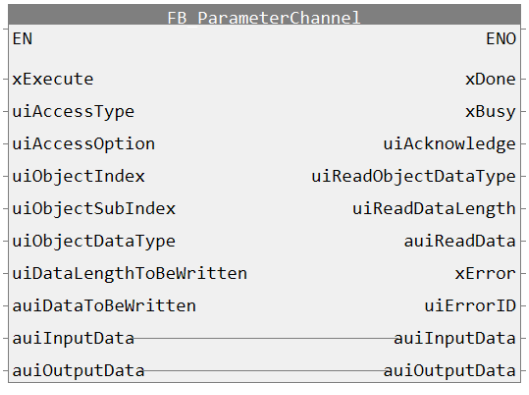
This application note describes the communication between the YJKP Servo Press Kit and a Keyence KV-X Series PLC via EtherNet/IP.

The **Festo_YJKP_Client_EIP** library contains two function blocks for controlling and parameterizing the YJKP Servo Press Kit.

The **FB_ControlChannel** block provides press control functions for operating the Press System.

The **FB_ParameterChannel** block provides read and write access to the specified Press System objects.





1.1 Software Used

Software Name	Software Version
Keyence KV STUDIO	V12.41
Festo Automation Suite	V2.12.0.436
Application Software YJKP	V2.3.0
Library (Festo_YJKP_Client_EIP)	V12.41.10

Table 1.1: Software Used

1.2 Hardware Used

Device Name	Device Type	Version / Firmware
Keyence PLC	KV-X520	V1.602
Servo Press Kit	8077950	EVO 4.1
Firmware Controller	CECC-X	V3.8.14
Firmware Motor Controller	CMMT-AS	V36.12.0
RJ45 Ethernet Cable	-	Version Free

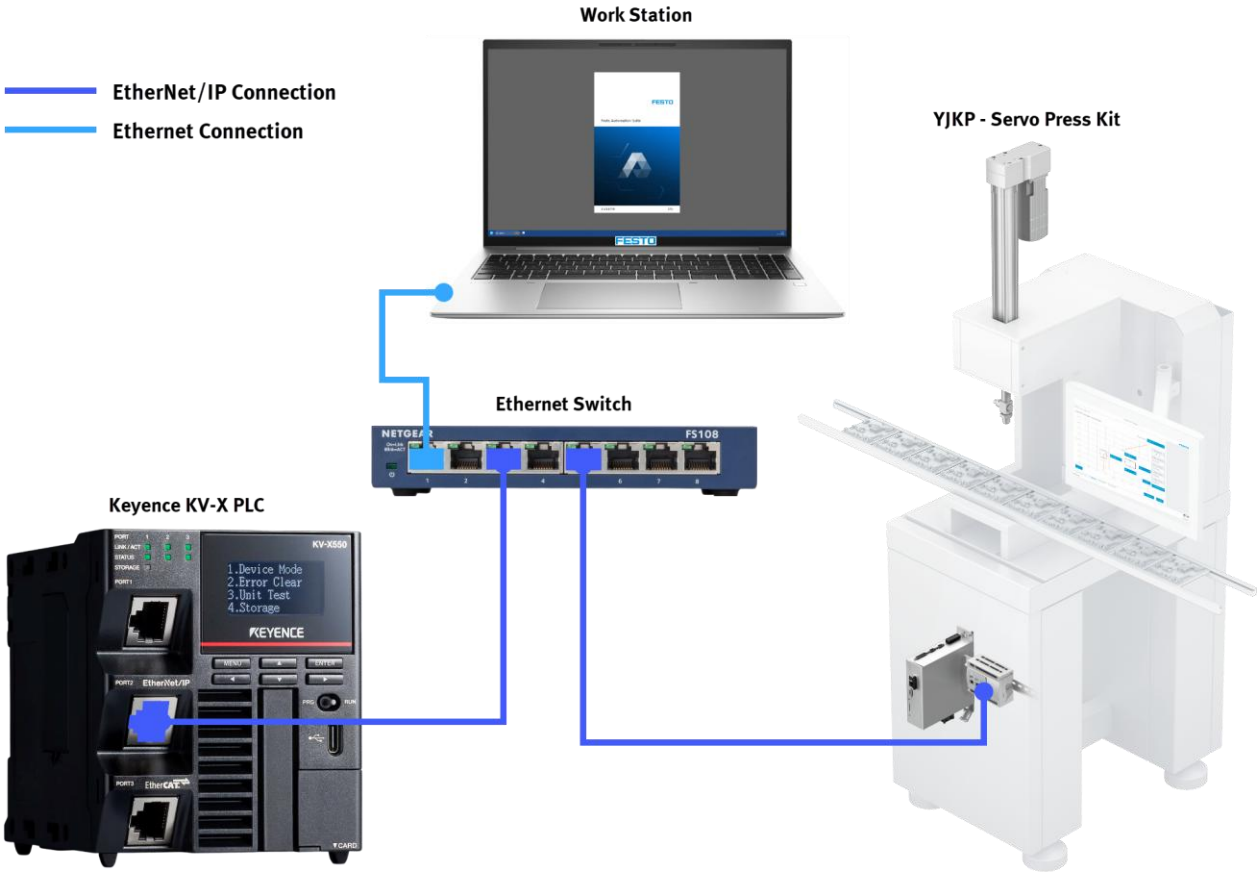
Table 1.2: Hardware Used

1.3 Topology Overview

Note

This application note entirely based on the following Topology Overview configuration as described.

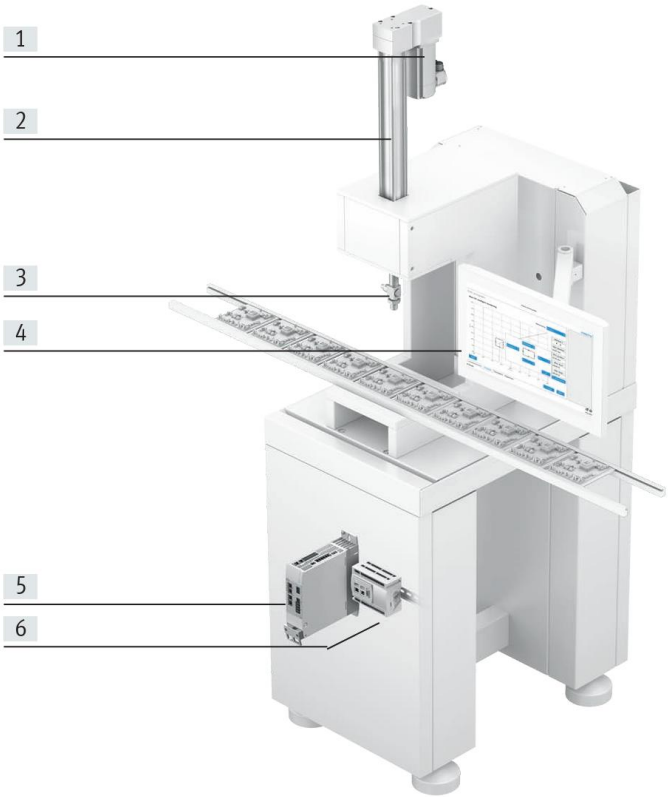
- The following Topology Overviews are only for an example to this application note. It may vary depending on actual user network configuration.



2 Hardware Configuration

2.1 Servo Press Kit YJKP Overview

2.1.1 Device Overview



Individual components:
[1] Servo motor
[2] Electric cylinder
[3] Force sensor (incl. inspection record)
[4] Software package
[5] Motor controller
[6] Controller (incl. micro SD memory card)

Motor cable included in the scope of delivery.

Note

- The Servo Press Kit YJKP - device overview above illustrates a sample device configuration. However, the device configuration may vary depending on the specific application requirements.

2.1.2 Product Overview

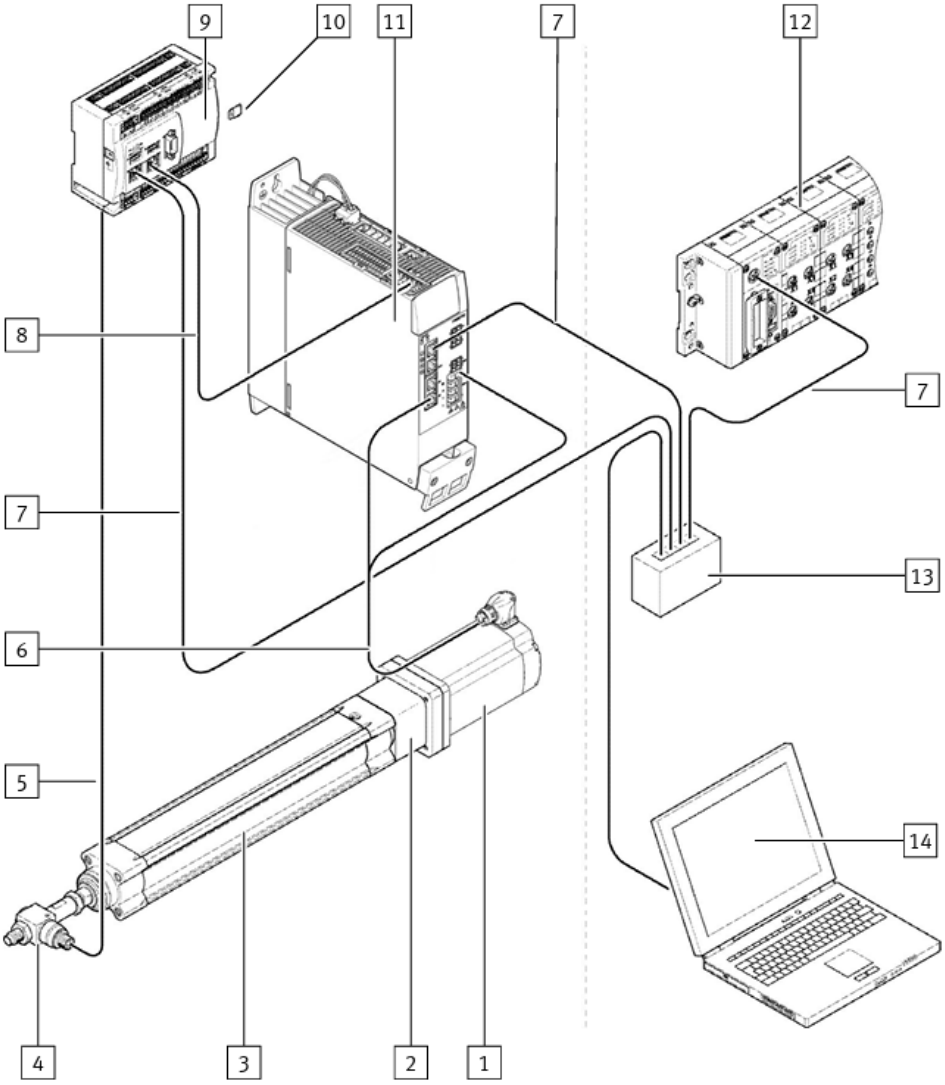


Fig. 1: System overview

1	Servo motor EMMT-AS
2	Axial kit EAMM-A/parallel kit EAMM-U
3	Electric cylinder ESBF-BS
4	Sensor SKDA-...-AB
5	Connecting cable NEBS-M12G5-ES-...-LE5
6	Motor cable NEBM-M...-EH-...-R3LEG14
7	LAN cable (not in scope of delivery)
8	LAN cable
9	Controller CECC-X-M1-YS
10	Memory card
11	Motor controller CMMT-AS
12	Higher-order controller (not in scope of delivery)
13	Switch (not in scope of delivery)
14	Laptop (not in scope of delivery)

Note

- The Servo Press Kit YJKP - product overview above illustrates a sample product configuration. However, the product configuration may vary depending on the specific application requirements.

2.2 Servo Press Kit YJKP System Configuration

This chapter describes the system configurations for the YJKP Servo Press Kit.

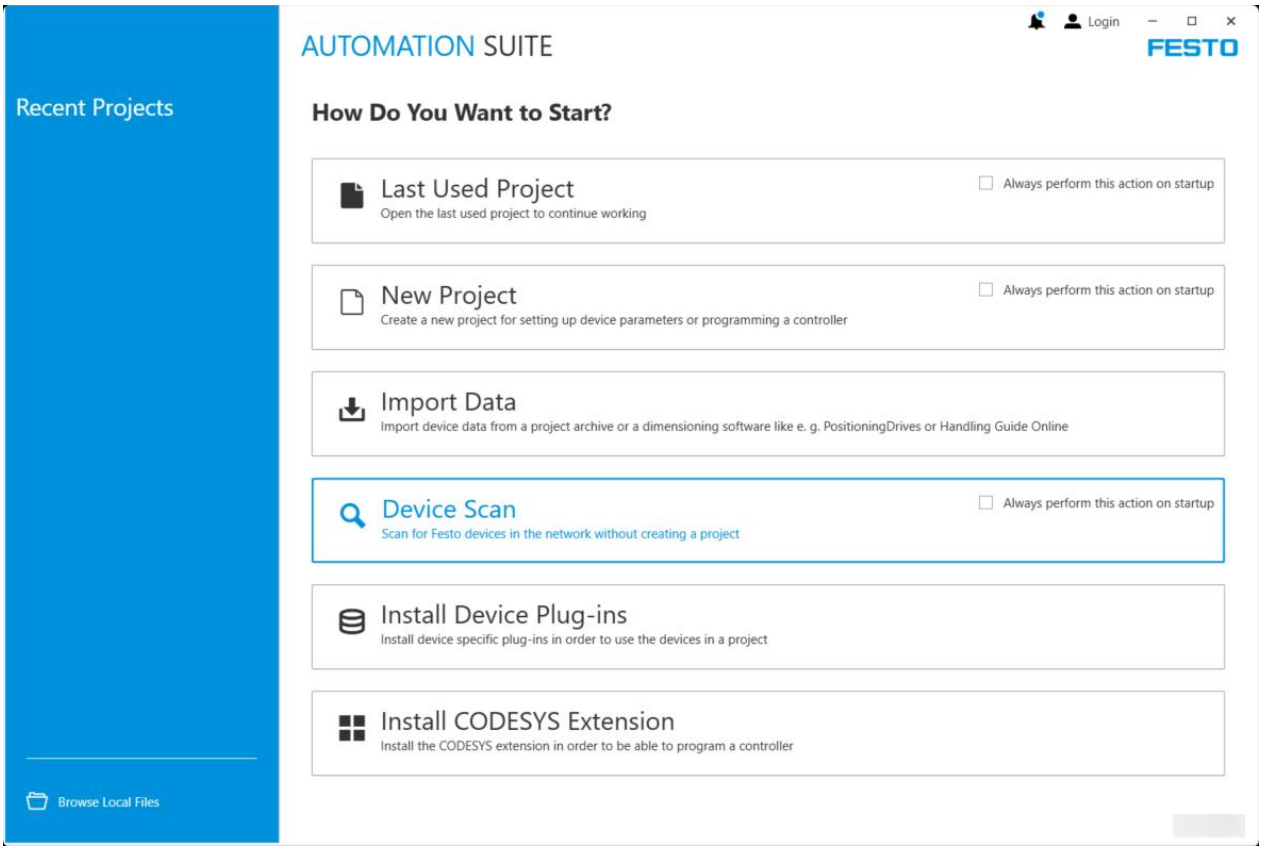
2.2.1 Assigning a Static IP Address

This section explains how to assign a static IP address to the firmware controller (CECC-X) of the YJKP Servo Press Kit using Festo Automation Suite.

Step 1

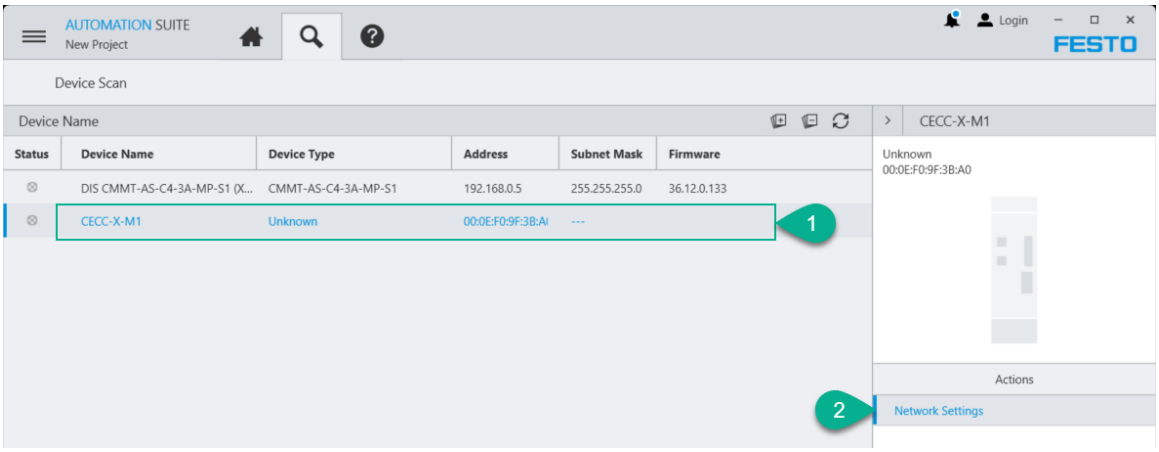
Open the **Festo Automation Suite** software (**Windows Start >> Festo Software >> Festo Automation Suite**).

Step 2



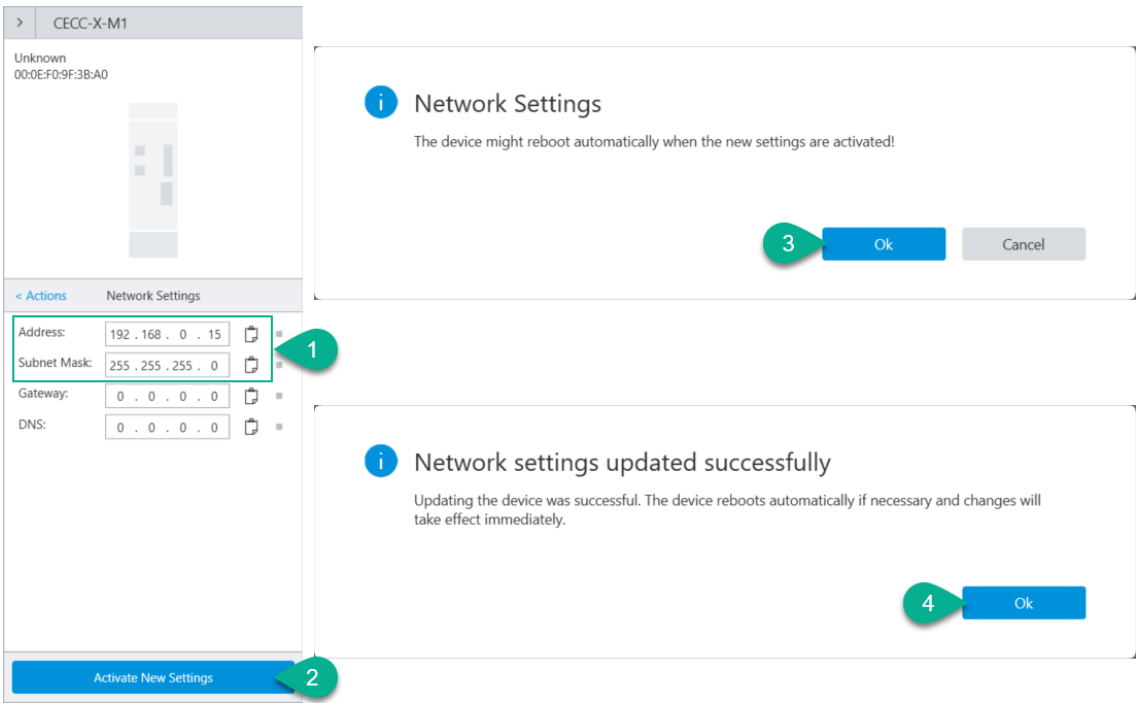
Click the **Device Scan** button on the start screen.

Step 3



1. Select the **CECC-X1-M1** controller from the list view in Festo Automation Suite.
2. Click **Network Settings** in the Action window.

Step 4



1. Enter the IP Address and Subnet Mask values for the firmware controller (CECC-X) as required.
2. Click the **Active New Settings** button to assign the entered values to the controller.
3. Click the **OK** button to reboot the device.
4. Click the **OK** button to close the confirmation window.

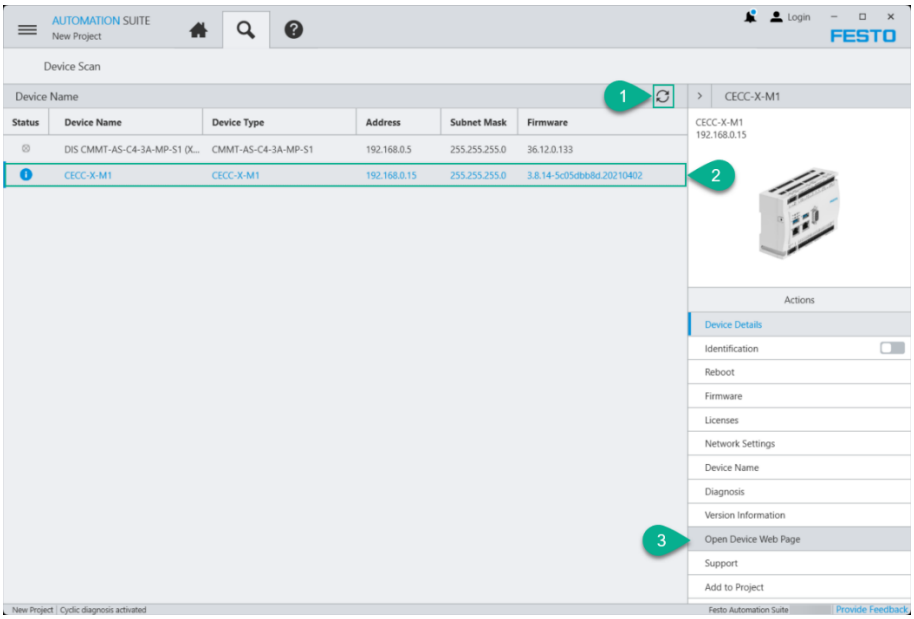
Note

- Ensure that the firmware controller (CECC-X) IP address is on the same subnet as the PLC and workstation IP addresses. The network configuration described above supports up to 254 devices.

2.2.2 Configuring the YJKP System Settings

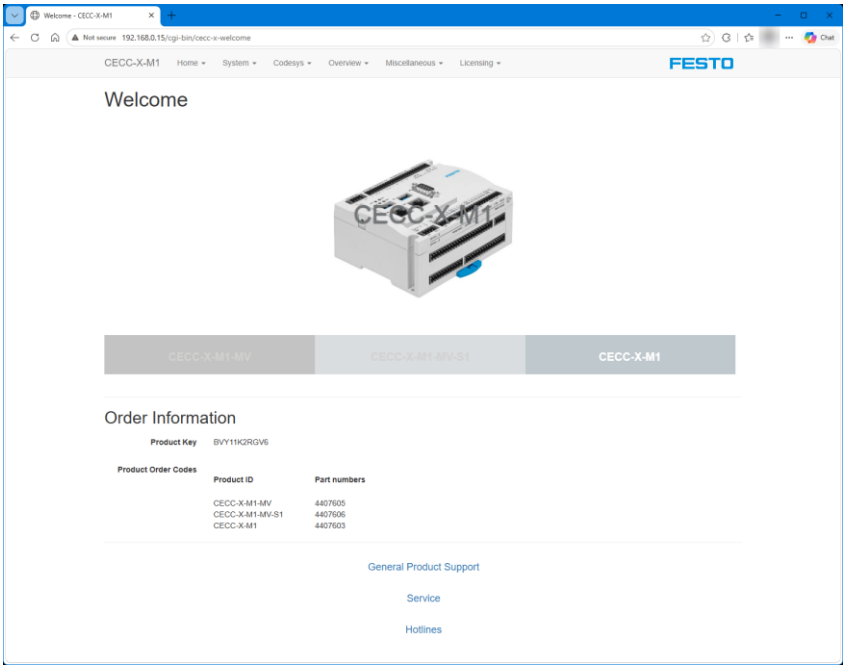
This section explains how to configure the YJKP Servo Press Kit system settings through the web server.

Step 1



1. Select the **Scan** button to update the list of connected devices.
2. Select the **CECC-X-M1** controller in the List View of Festo Automation Suite.
3. Click **Open Device Web Page** in the Action window.

Step 2

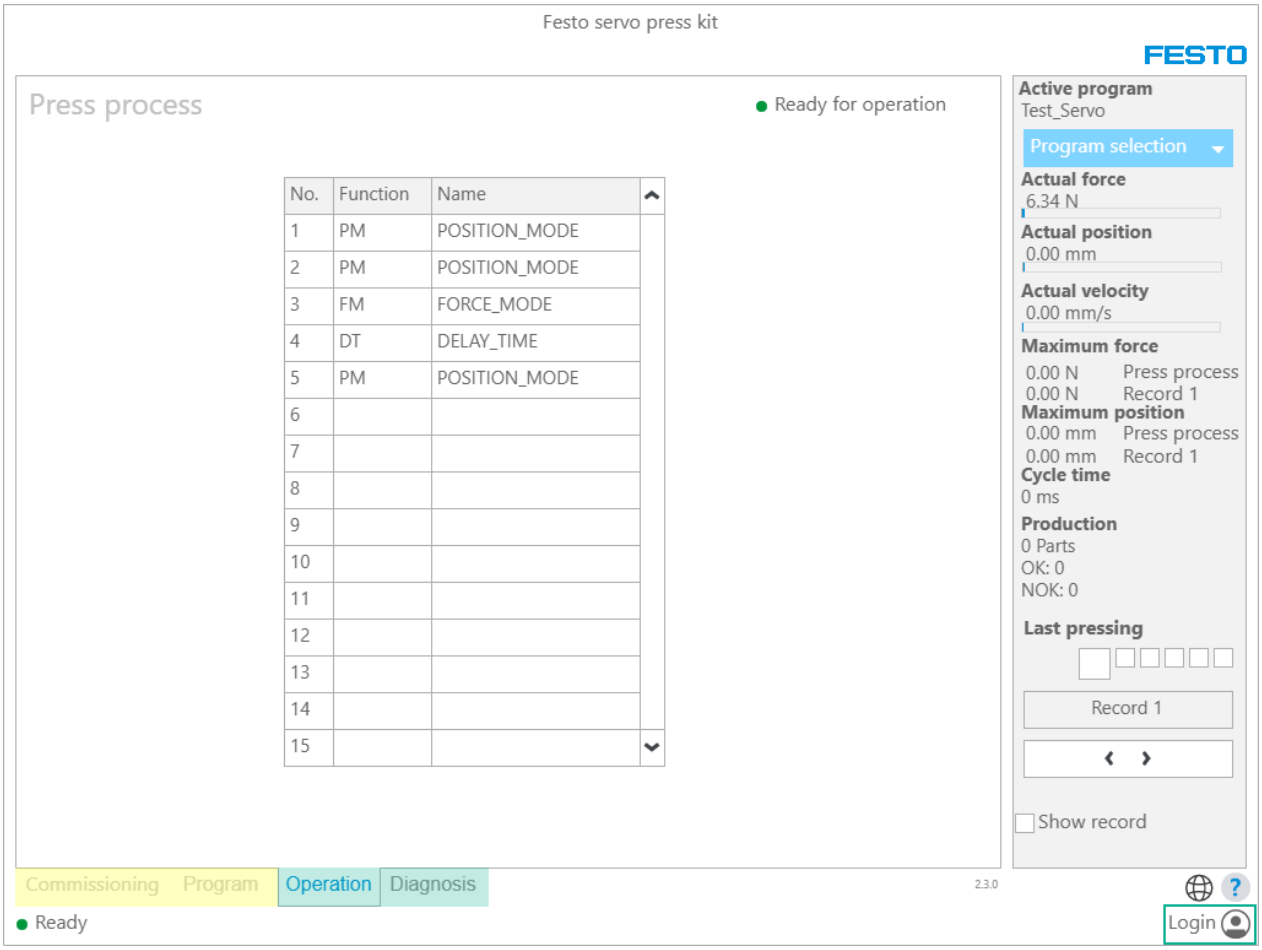


The CECC-X-M1 controller's web server opens, as shown above. The order details are displayed on the web server page.

Step 3

1. Select the **Codesys** tab on the Home page of the web server.
2. Select **webvisu-servo_press_kit** from the drop-down list.

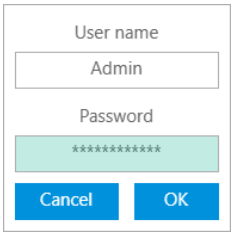
Step 4



The servo-press kit visualization opens with four tabs:

- Commissioning and Program: Inactive (highlighted in yellow)
- Operation and Diagnosis: Active (highlighted in green)

Click **Login** to activate the **Commissioning** and **Program** tabs. A pop-up window is displayed, prompting you to enter a password to log in.

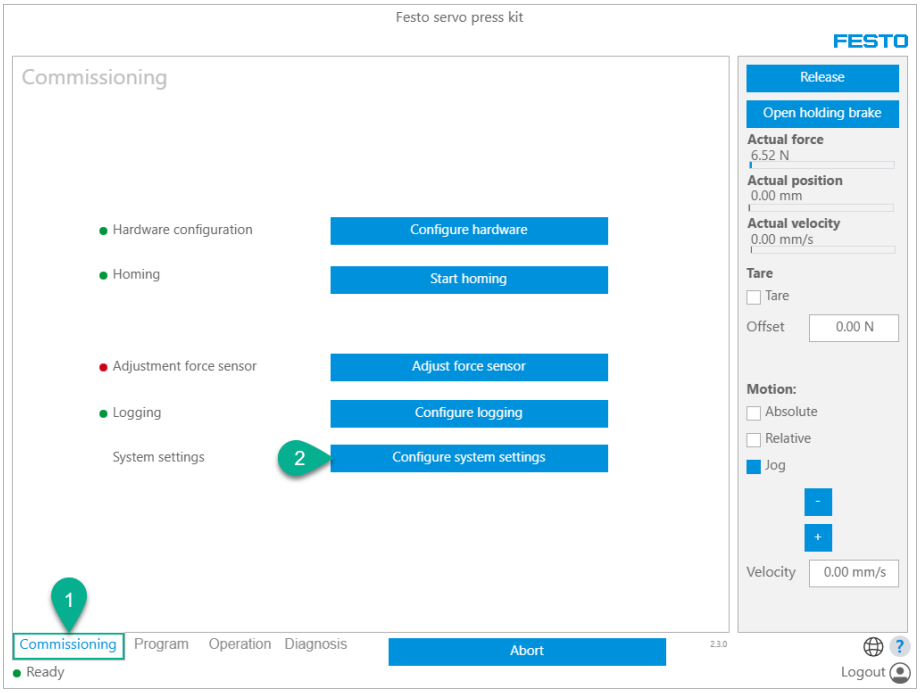


Default Credentials

User Name : Admin

Password : ServoPressKit

Step 5

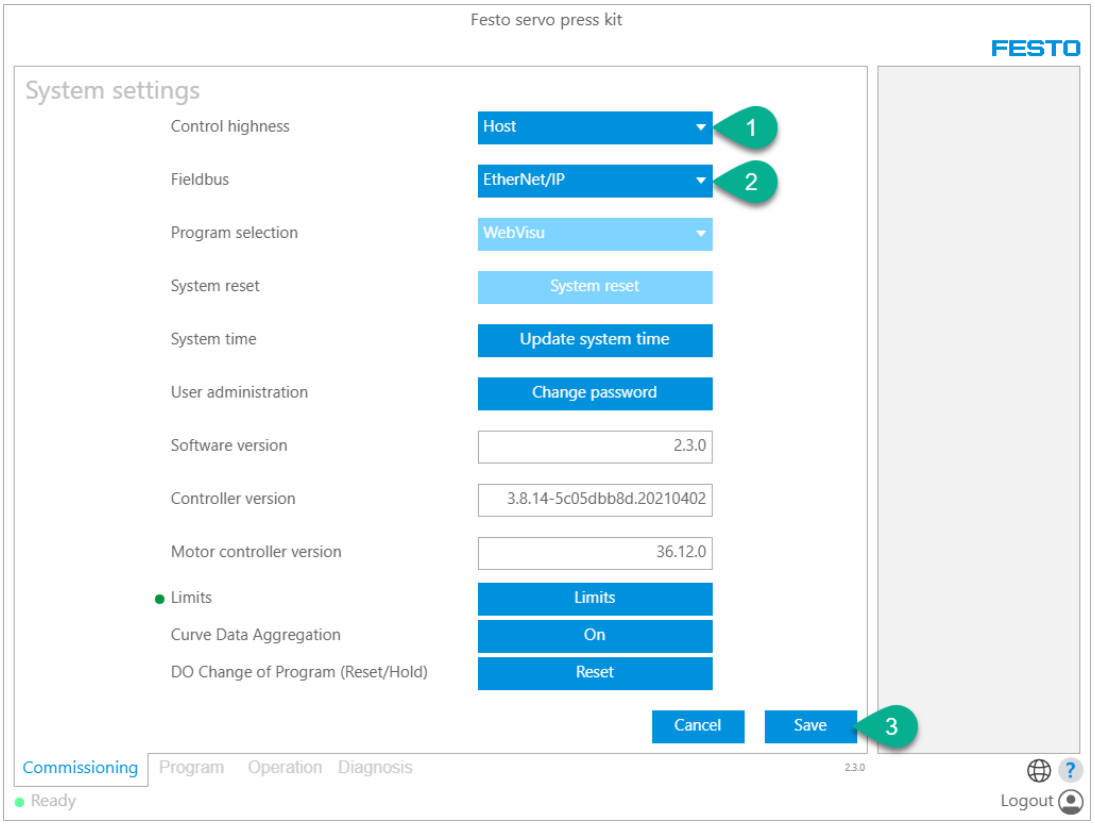


1. Go to the **Commissioning** tab.
2. Click the **Configure System Settings** button.

Note

- To configure the YJKP Servo Press Kit's hardware configuration according to the user's requirements, refer to the application note **YJKP – Hardware Configuration and System Settings**, available on the **Festo Support Portal**.

Step 6



1. Set **Control Highness** for the YJKP Servo Press Kit to **Host** to control the YJKP system using the host PLC.
2. Set **Fieldbus** for the YJKP Servo Press Kit to **EtherNet/IP** to establish communication between the host PLC and the YJKP system.
3. Click the **Save** button to save the changes to the YJKP Servo Press Kit.

Note

- If the YJKP Servo Press Kit has a single-turn encoder, you must perform the homing procedure again after restarting the system.
- For more information about the single-turn encoder, refer to the application note **YJKP – Hardware Configuration and System Settings**, available on the **Festo Support Portal**.

3 Software Configuration

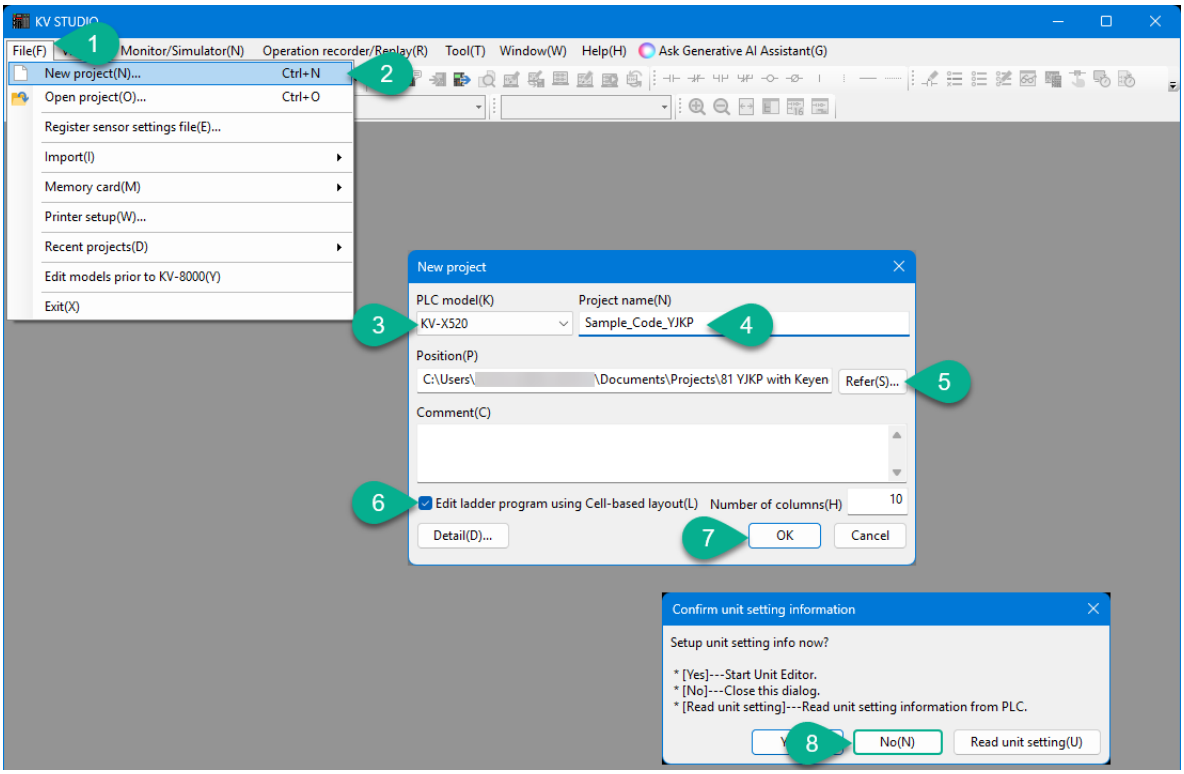
This chapter describes Keyence KV STUDIO software configuration tasks, including creating a new project, adding the controller, configuring YJKP device on the EtherNet/IP network, and adding the **Festo_YJKP_Client_EIP** library to the project.

3.1 Creating a New Project and Adding Controller in KV STUDIO

Step 1

Open the **KV STUDIO Ver.12G** software (**Windows Start >> KEYENCE KV STUDIO Ver.12G >> KV STUDIO Ver.12G**).

Step 2

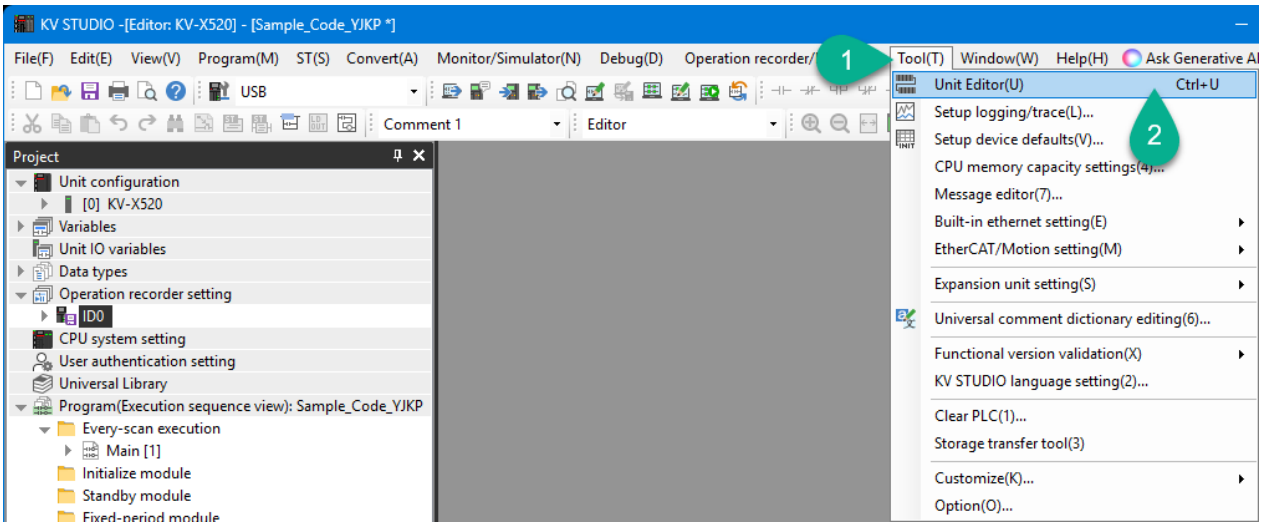


1. Click **File(F)** on the toolbar.
2. Select **New project(N)...** from the **File** menu.
3. Select **KV-X520** as the PLC model from the **PLC model(K)** drop-down list. Select the appropriate PLC model according to user requirements.
4. Enter a name for the project in the **Project name(N**) field.
5. Click the **Refer(S)...** button, then select the folder in which to save the project.
6. If required, select the **Edit ladder program using Cell-based layout(L)** checkbox.
7. Click **OK** button to create the project.
8. Click **No** button to close the **Confirm unit setting information** window.

3.2 Controller Configuration and Setup

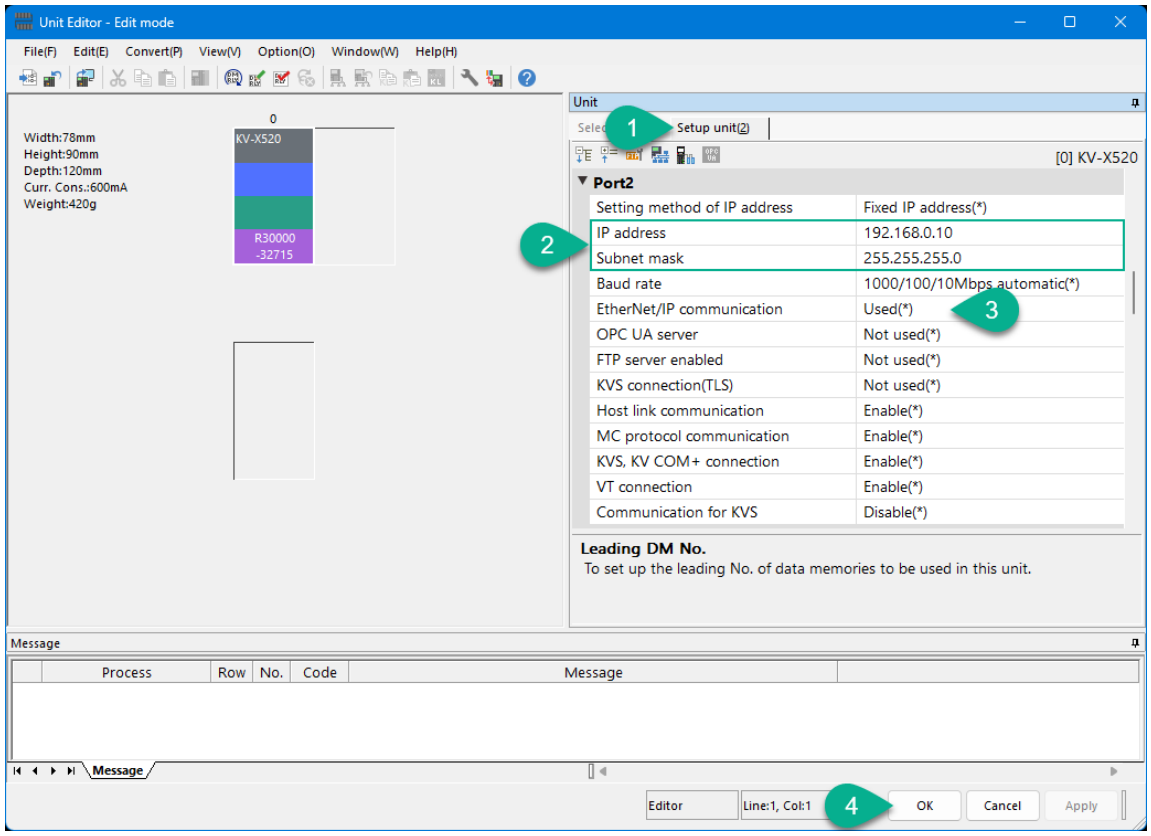
3.2.1 Built-in EtherNet/IP Port Configuration

Step 1



1. Click **Tool (T)** on the toolbar.
2. Select **Unit Editor (U)...** from the **Tool** menu.

Step 2



1. Click the **Setup Unit(2)** tab in the **Unit Editor** window.
2. Enter the required **IP address** and **Subnet mask** values for the PLC in the **Port2** settings. For this example, **192.168.0.10** as the **IP address** and **255.255.255.0** as the **Subnet mask** has been used.
3. Ensure that **EtherNet/IP communication** is selected as **Used** in the **Settings** field.
4. Click **OK** button to apply the changes and close the window.

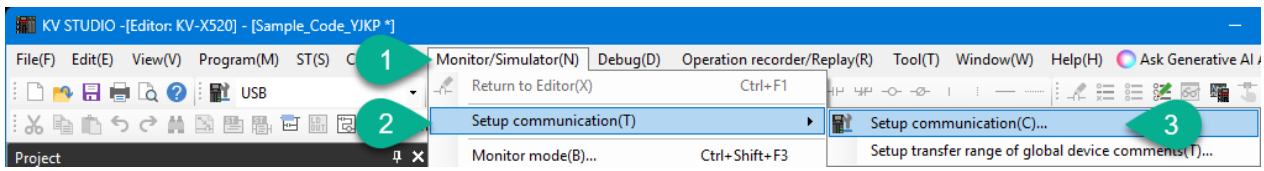
Note

Port configuration selection.

- The Keyence KV-X520 PLC has **two Ethernet ports** that are used as programming interfaces. In this example, the PLC is connected to the workstation through **Ethernet port 2**. Refer Chapter 1.3.

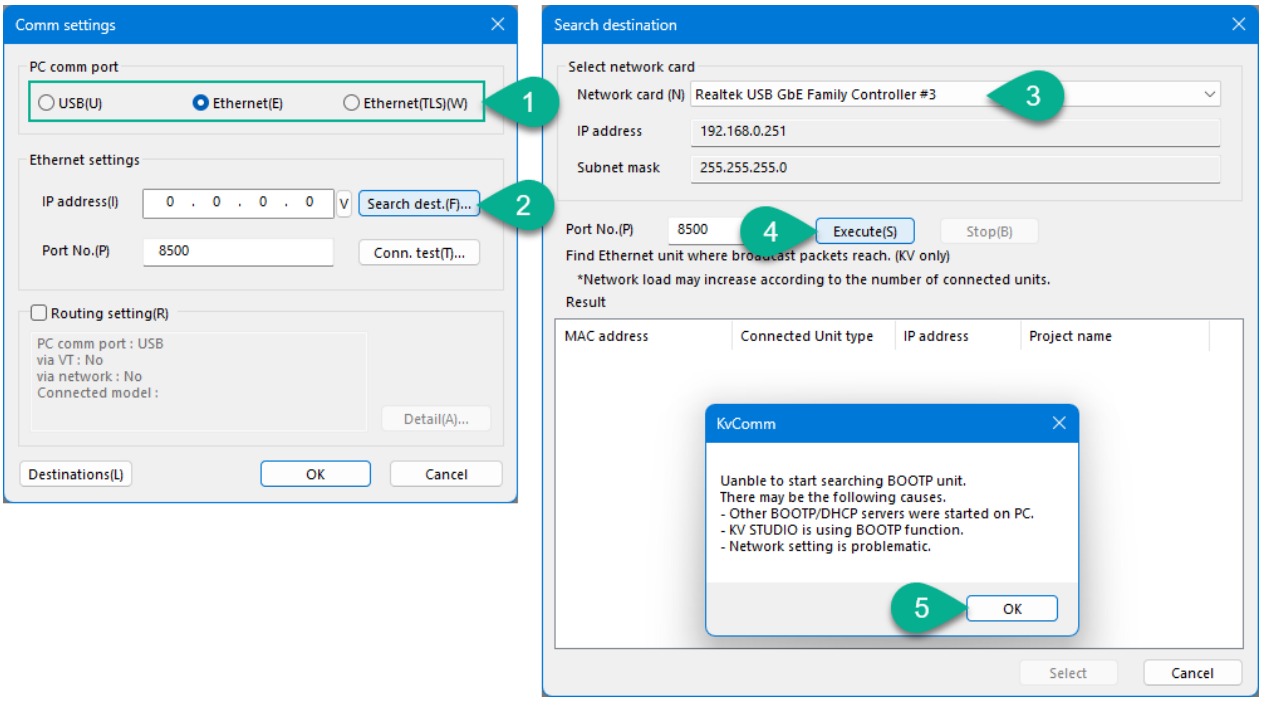
3.2.2 PLC Communication Setup

Step 1



1. Click **Monitor/Simulator(N)** on the toolbar.
2. Select **Setup communication(T)** option.
3. Click **Setup communication(C)...** to open the communication setup.

Step 2



1. Select the required **Connection Type** under **PC comm port** settings in the **Comm settings** window. For this application note, **Ethernet(E)** is used as the connection type.
2. Click **Search dest.(F)...** under **Ethernet settings** option.
3. Select the required **Network Card** from the **Network card (N)** drop-down list in the **Search destination** window.
4. Click **Execute(S)** to search for the connected controller.
5. Click **OK** button to close the **KVComm** dialog box.

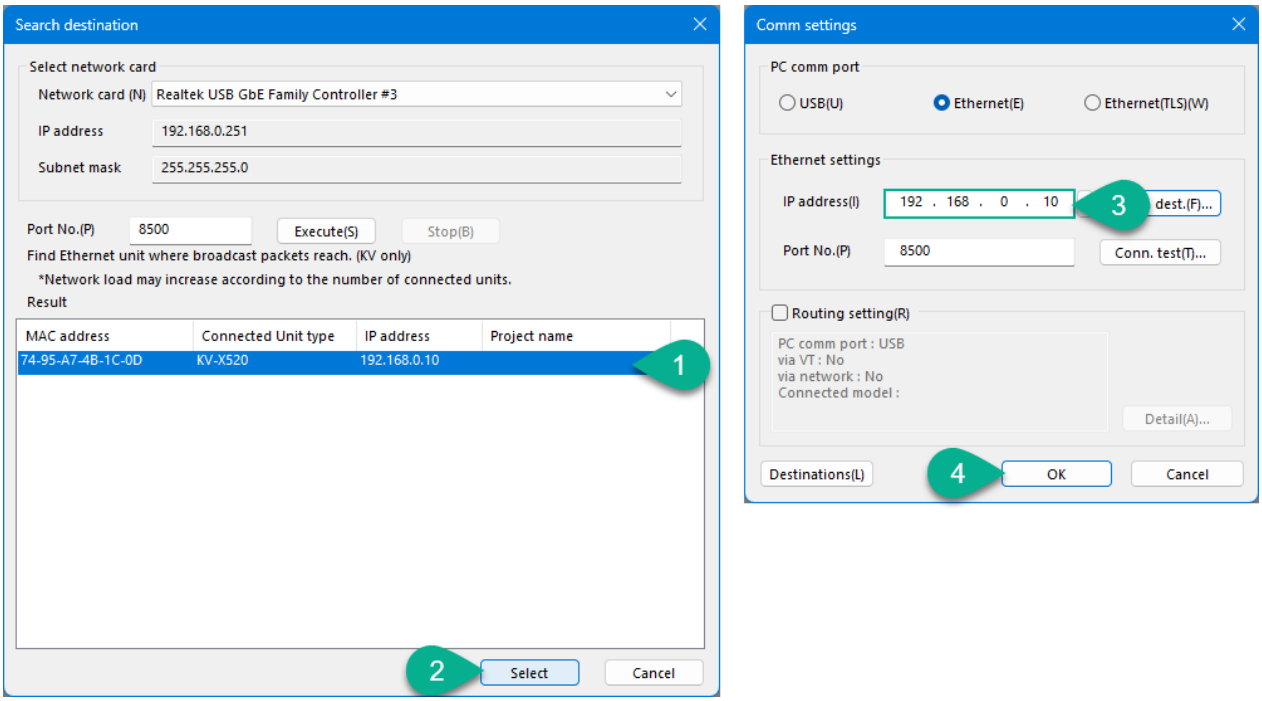
Note

The previous step applies only when using an existing controller.

If you are not using an existing controller, follow the step below.

- Factory-new controllers have a default IP address. If you are using a factory-new controller or do not know the controller's IP address, select **USB** as the connection type during initial setup.

Step 3

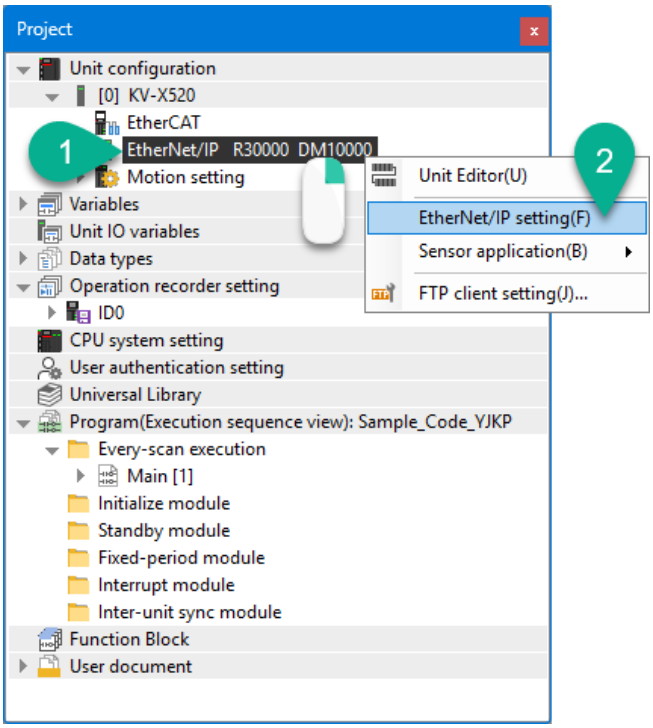


1. Once the connected controller is detected, it appears in the **Results** area.
2. Select the detected controller, then click **Select** to apply the settings.
3. The selected controller's **IP address** is displayed in the **IP address(I)** field under **Ethernet settings** as shown.
4. Click **OK** button to update the connection settings and **Close** the **Comm settings** window.

3.3 YJKP Device Configuration

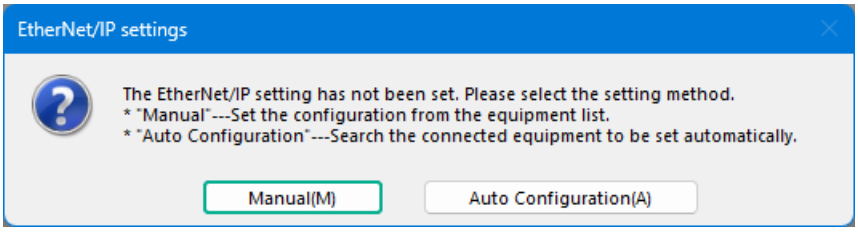
3.3.1 Adding YJKP Device to EtherNet/IP Network

Step 1



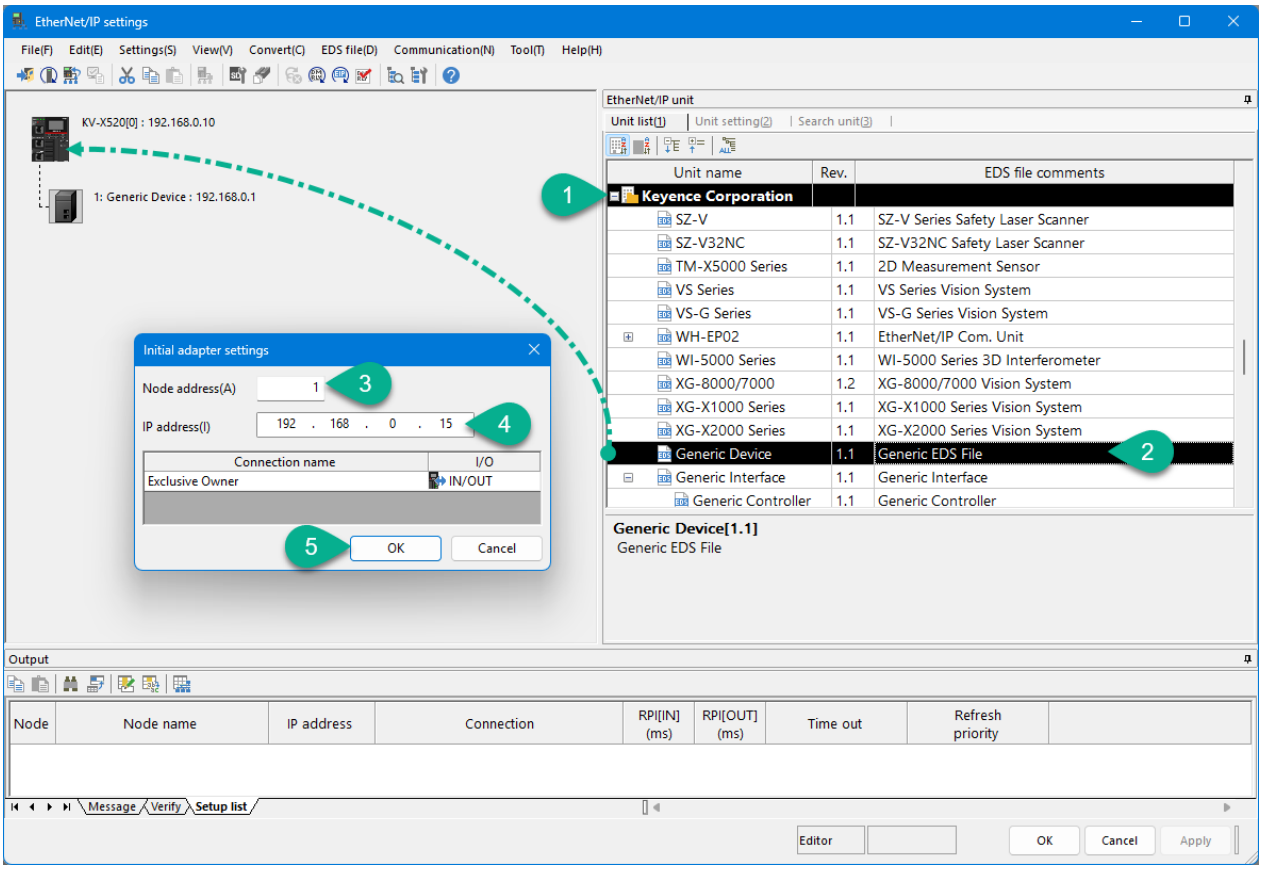
1. Right-click **EtherNet/IP** settings of the **PLC** under **Unit configuration** in the **Project** workspace.
2. Select **EtherNet/IP settings(F)** from the list.

Step 2



Click **Manual** to configure the equipment list in the **EtherNet/IP settings** window.

Step 3

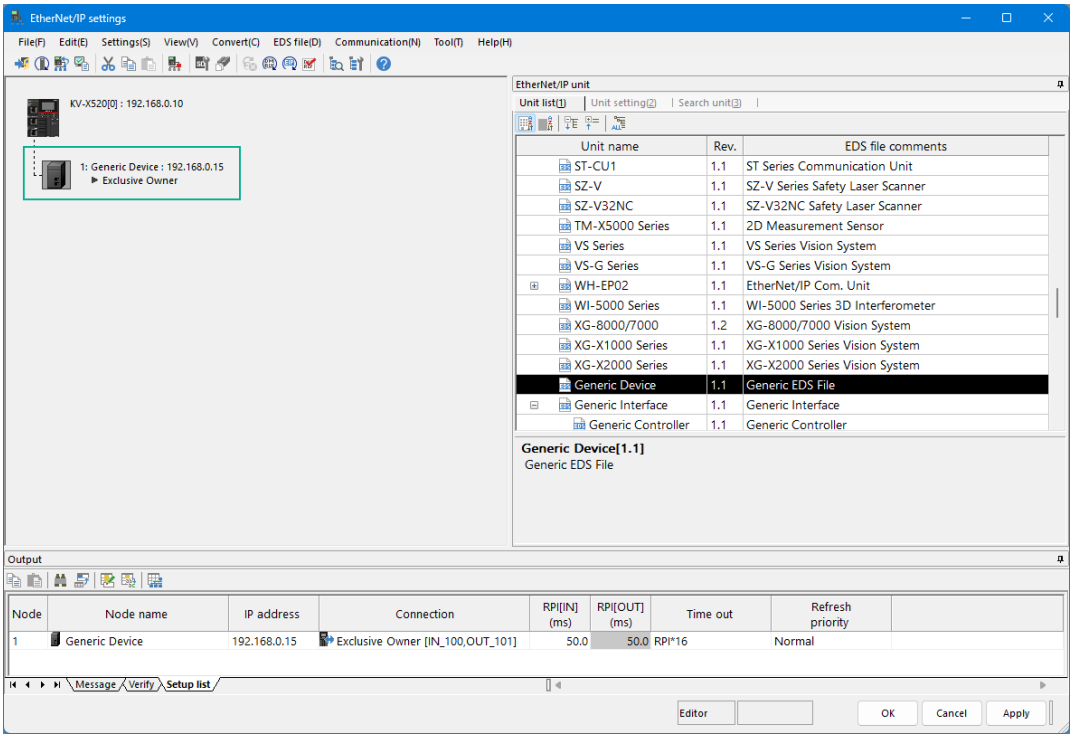


1. Select the **Generic Device EDS** file from the repository under **Keyence Corporation** in the **EtherNet/IP Unit** list.
2. Drag and drop the **Generic Device EDS** file into the project.
3. Enter the required **Node address** in the **Initial adapter settings** window. The **Node address** is the device number on the network.
4. Enter the required **IP address** in the **Initial adapter settings** window. Refer Chapter 3.1 for details about the IP address.
5. Click **OK** button to add the module to the network.

Note

- Because the **YJKP** does not have an **EDS** file of its own, user must use **Generic Device** to configure it.
- The IP addresses of the **YJKP device**, the **KV-X520 PLC**, and the **PC** must be on the same subnet.

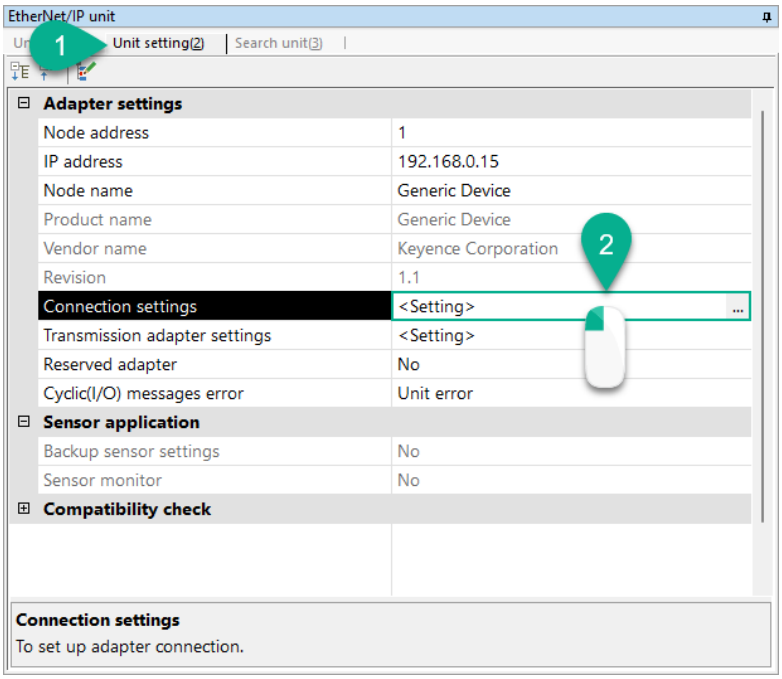
Step 4



The added **Generic Device** appears as shown.

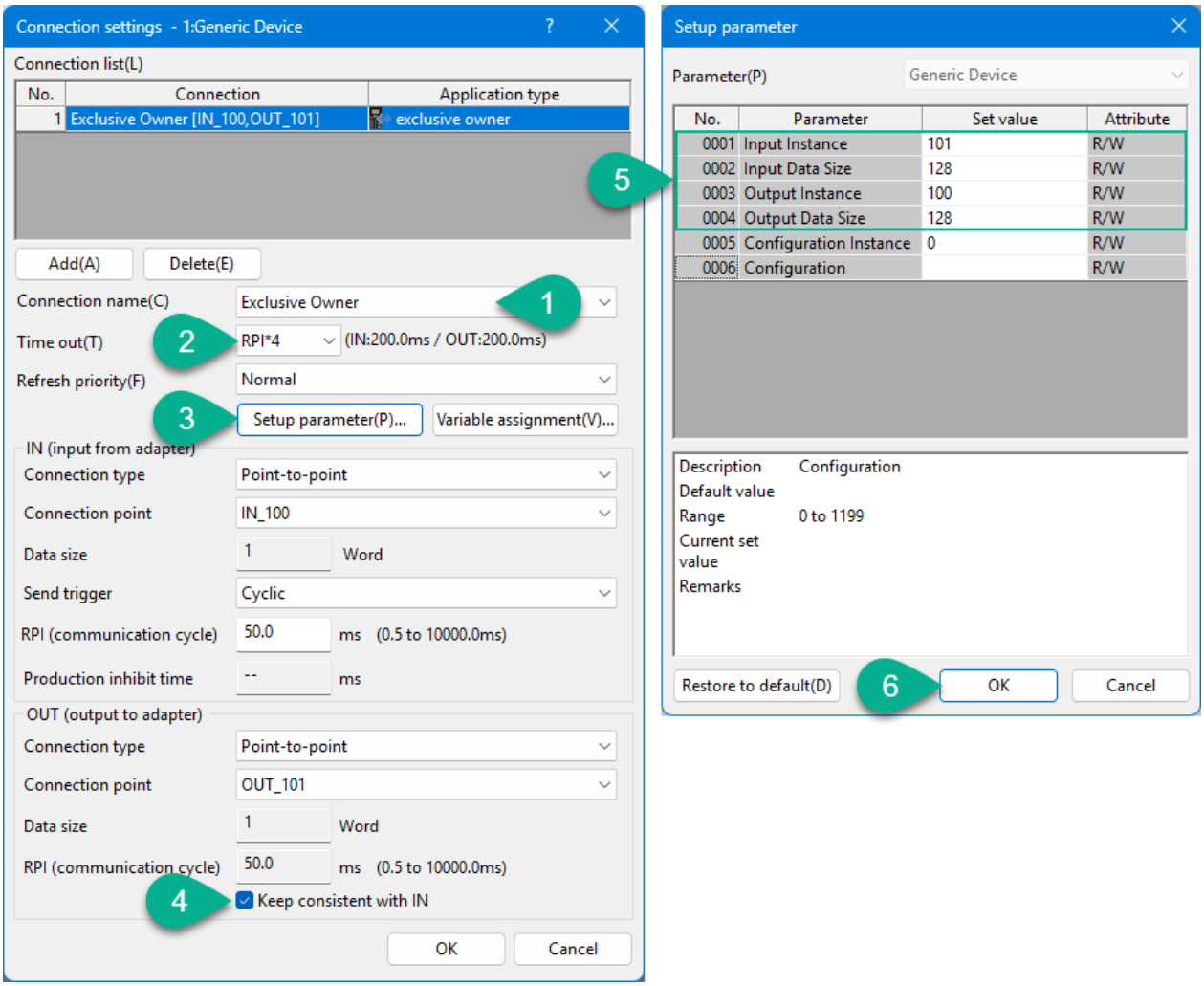
3.3.2 Adapter Connection Settings

Step 1



1. Select the added **Generic Device**, then go to the **Unit setting(2)** tab in the **EtherNet/IP unit** window.
2. Double-click **Connection settings** in the **Adapter settings** category.

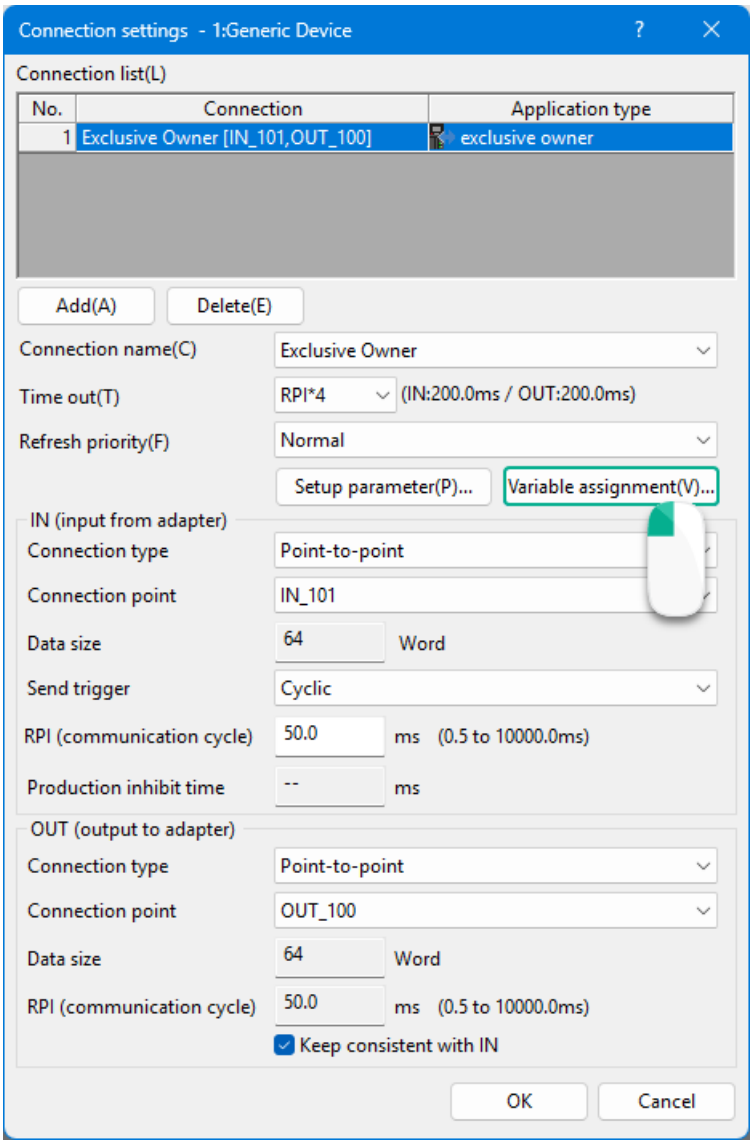
Step 2



1. Select **Exclusive Owner** from the **Connection name(C)** drop-down list.
2. Select the required **Time out(T)** value.
3. Click **Setup parameter(P)...** to configure the input and output sizes.
4. Set the input and output instances and data sizes as follows:
    - **Input Instance:** 101
    - **Input Data Size:** 128 bytes
    - **Output Instance:** 100
    - **Output Data Size:** 128 bytes
5. Click **OK** button in the **Setup parameter** window.
6. Select the **Keep consistent with IN** checkbox.

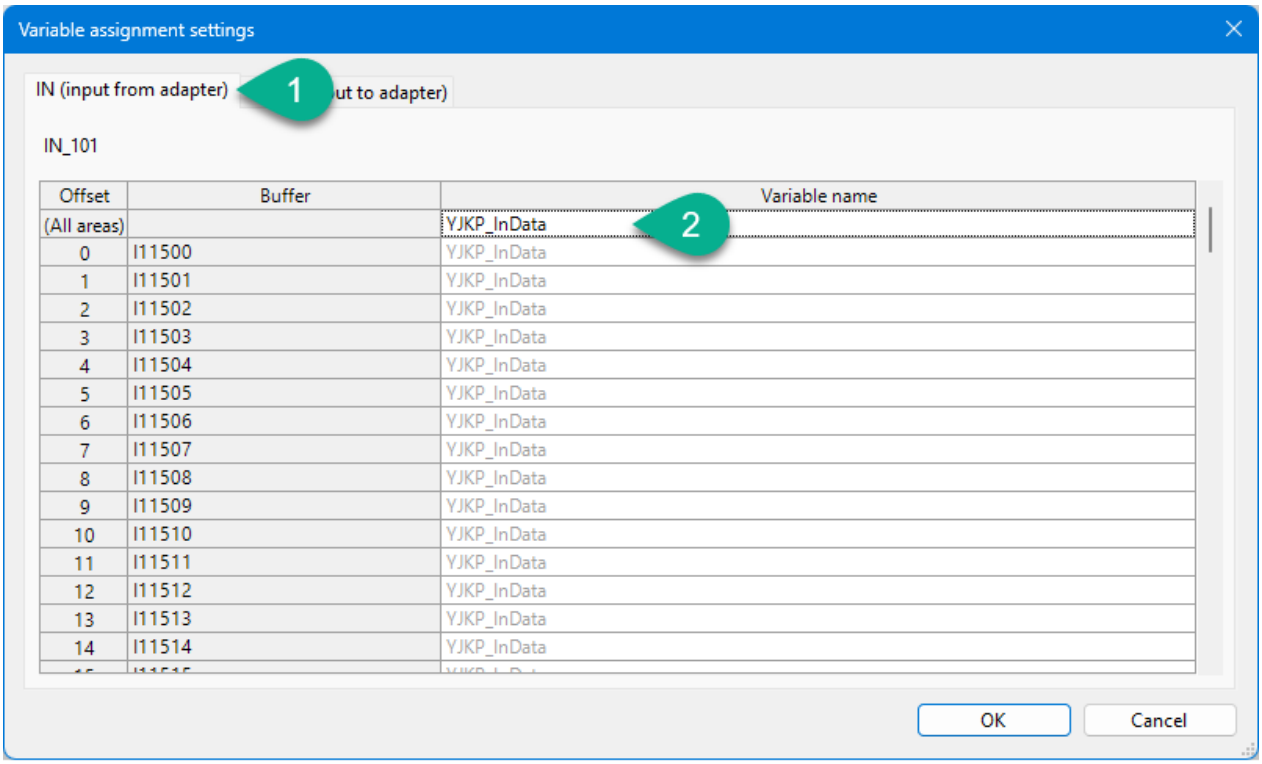
3.3.3 Creating Variables Used for YJKP Device Connection

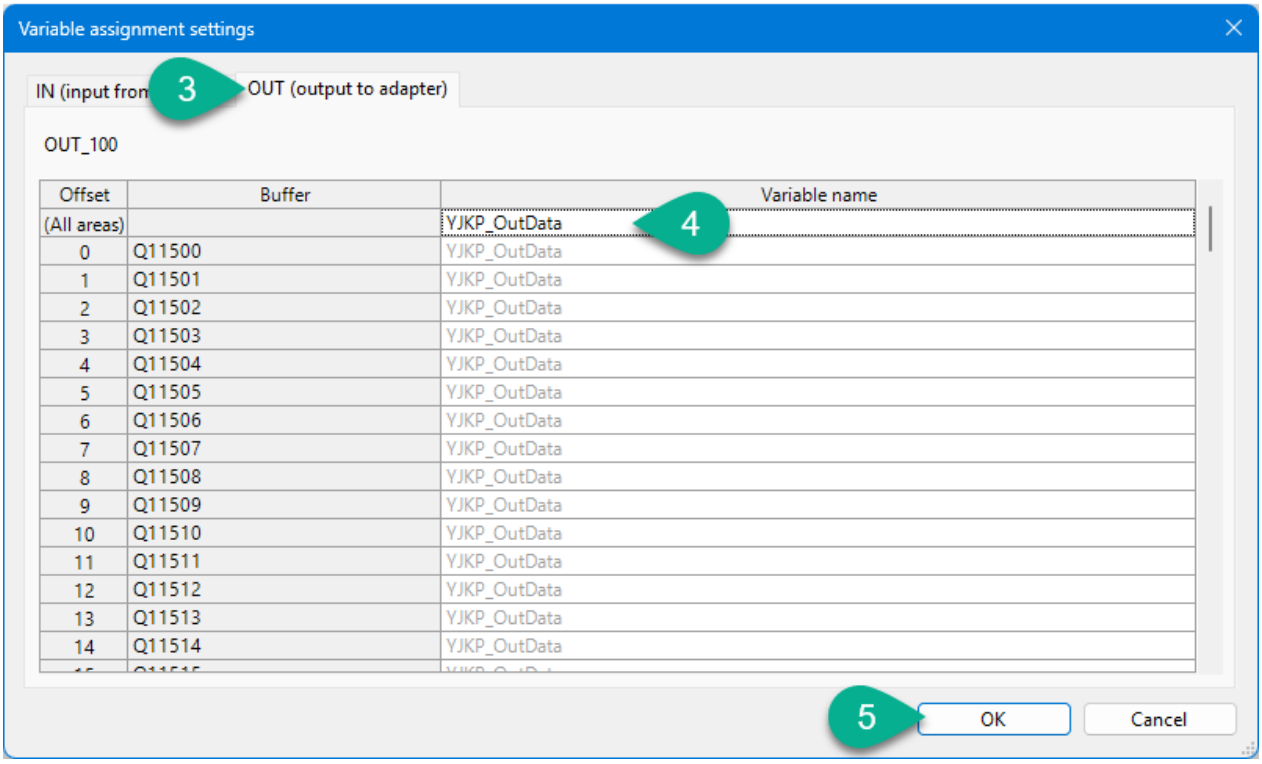
Step 1



Once the **Setup parameter** settings are complete, click the **Variable assignment(V)...** button to configure the **Variables** for the YJKP device.

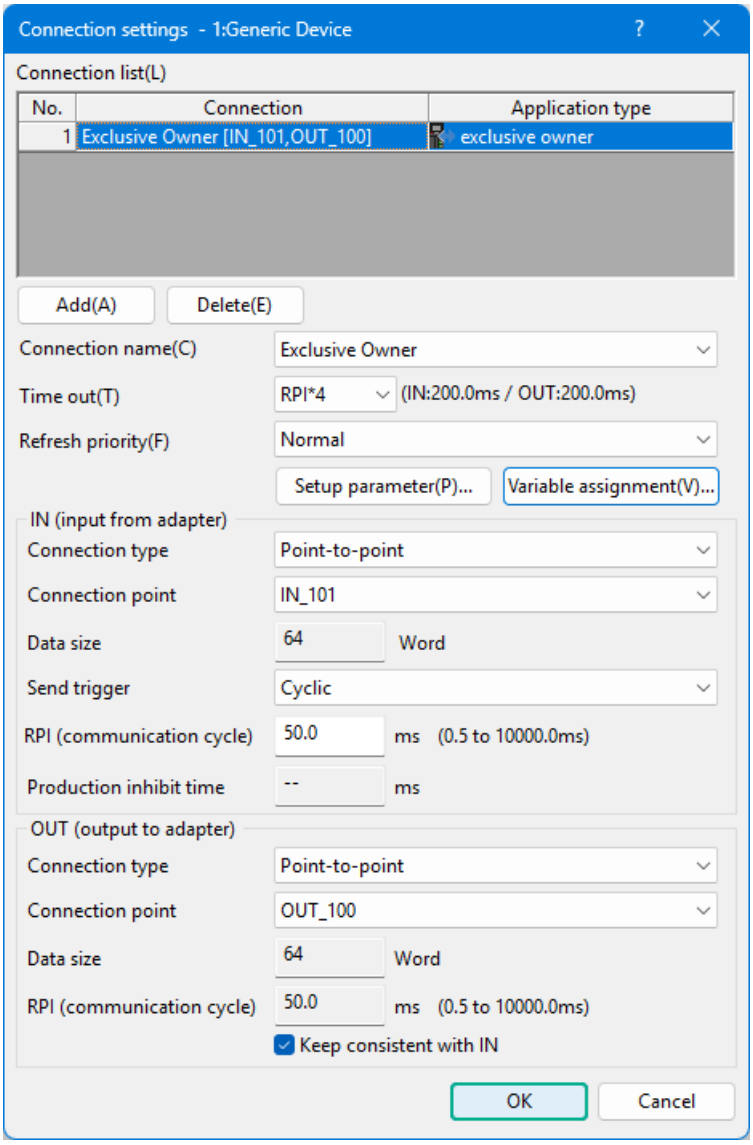
Step 2





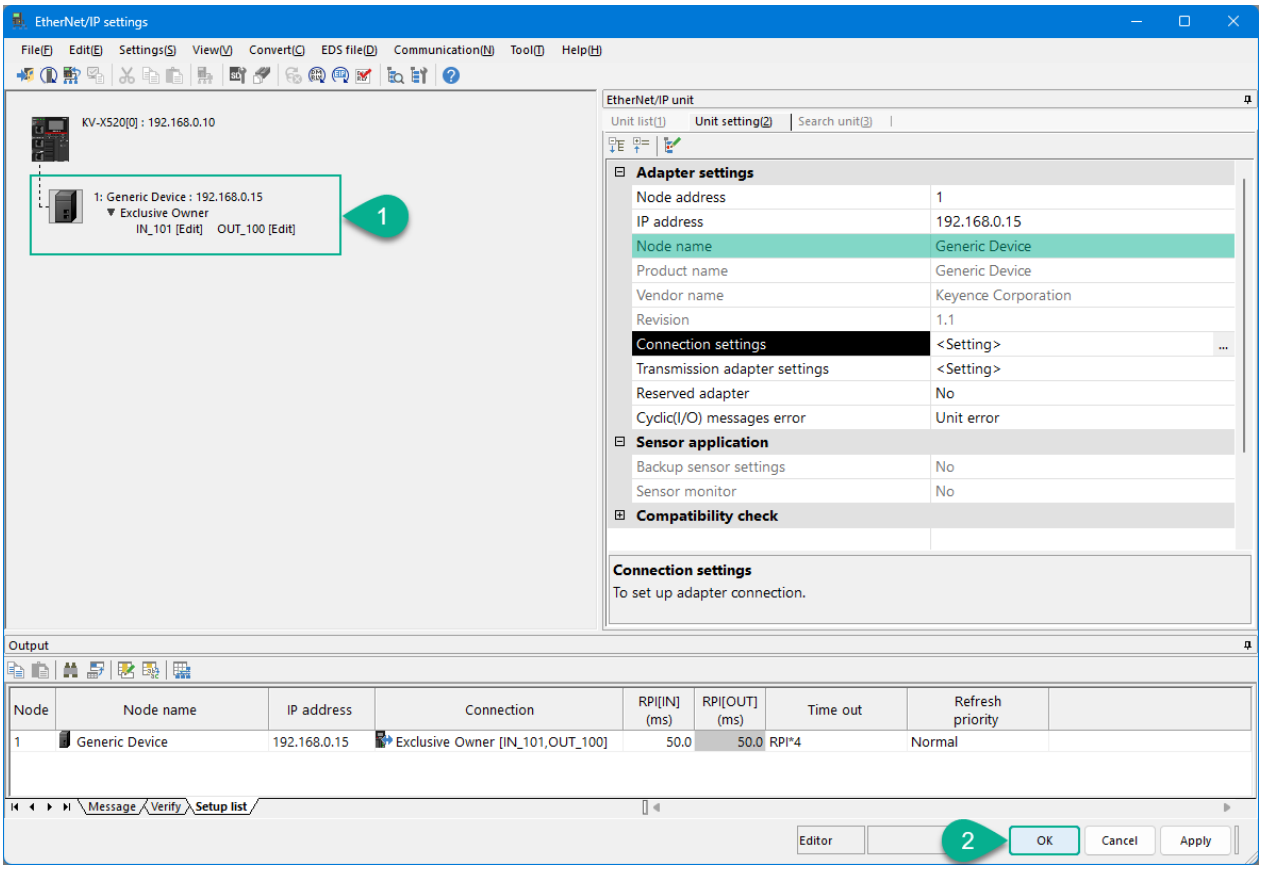
1. Go to the **IN (input from adapter)** tab in the **Variable assignment settings** window.
2. Enter the required **Input Tag** name in the **Variable name** column of the **All areas** row, as shown.
3. Go to the **OUT (output to adapter)** tab in the **Variable assignment settings** window.
4. Enter the required **Output Tag** name in the **Variable name** column of the **All areas** row, as shown.
5. Click the **OK** button to close **the Variable assignment settings** window.

Step 3



Click the **OK** button in the **Connection settings** window to apply the changes.

Step 4

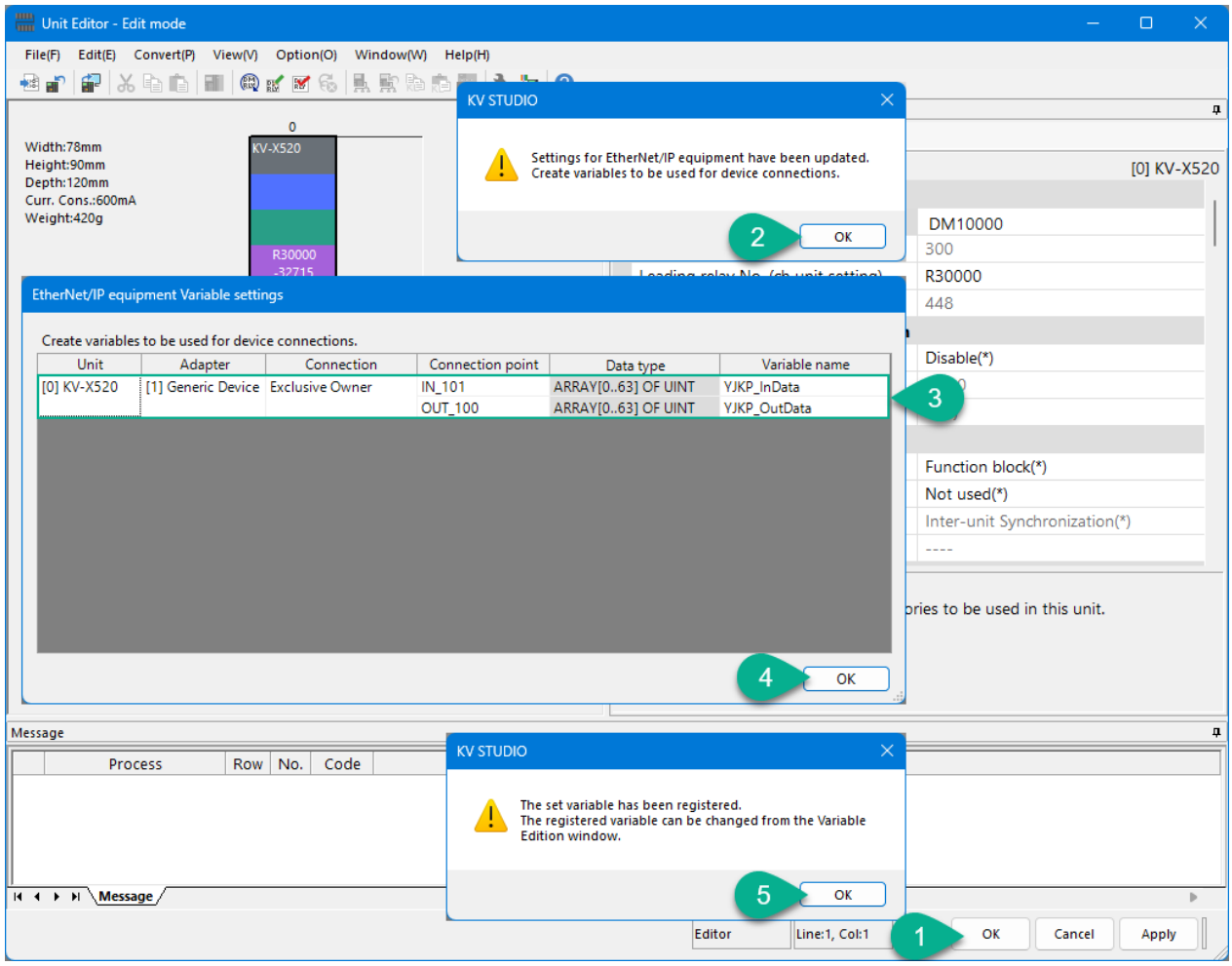


1. The **Connection settings** of the **YJKP Device** have been updated as shown.
2. Click **OK** button to **Apply** and **Save** the changes and close the **EtherNet/IP settings** window.

Note

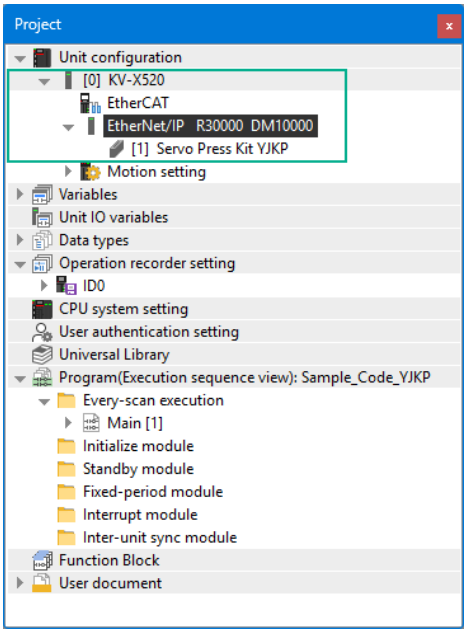
- The user can change the **Node Name** of the adapter device **(YJKP)** in the **Node name** option under the **Adapter settings** category, highlighted by the green box, on the **Unit setting(2)** tab when multiple **Generic Devices** are used.

Step 5



1. Click the **OK** button to **Apply** and **Save** the changes and **Close** the **Unit Editor** window.
2. Click the **OK** button in the pop-up window to create the connection device variables for the YJKP device, as shown.
3. Confirm that the **Unit, Adapter, Connection, Connection point, Data type,** and **Variable name** match the settings configured in the previous steps. If necessary, modify them in the **EtherNet/IP equipment Variable settings** window.
4. Click the **OK** button to confirm the details and create the YJKP device connection variables.
5. Click the **OK** button in the pop-up window to acknowledge the registered YJKP device variables.

Step 6



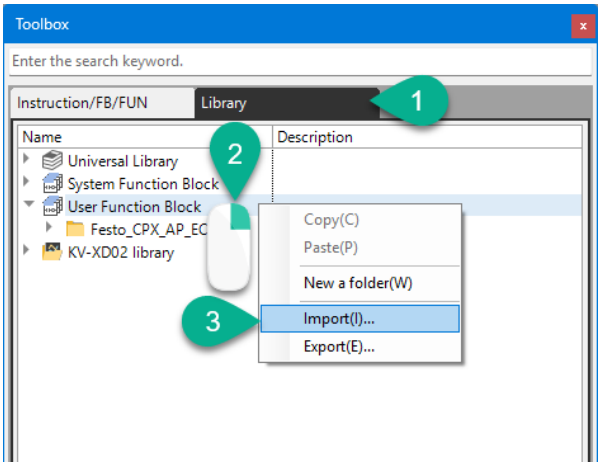
The added **Generic Device** for the YJKP (**Servo Press Kit YJKP**) appears under **Unit Configuration >> PLC (KV-X520) >> EtherNet/IP settings** in the **Project Workspace**, as shown.

3.4 Adding Festo_YJKP_Client_EIP Library into KV STUDIO Project

Step 1

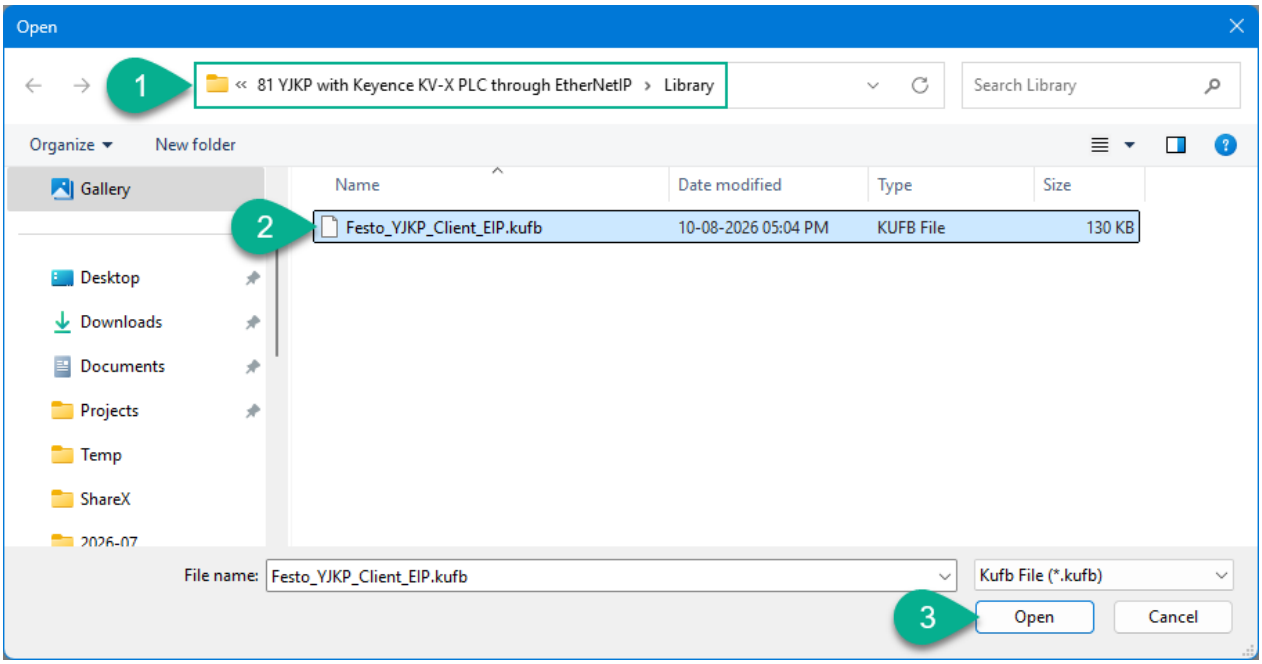
Download the **Festo_YJKP_Client_EIP** library from the **Festo Support Portal** and save it in an appropriate folder on the workstation.

Step 2



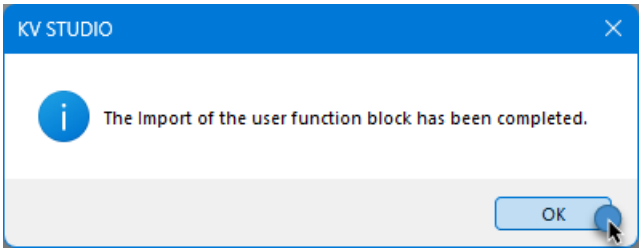
1. Select the **Library** tab in the **Toolbox** window.
2. Right-click the **User Function Block** option in the **Library** tab.
3. Click **Import** to import the library into the project.

Step 3



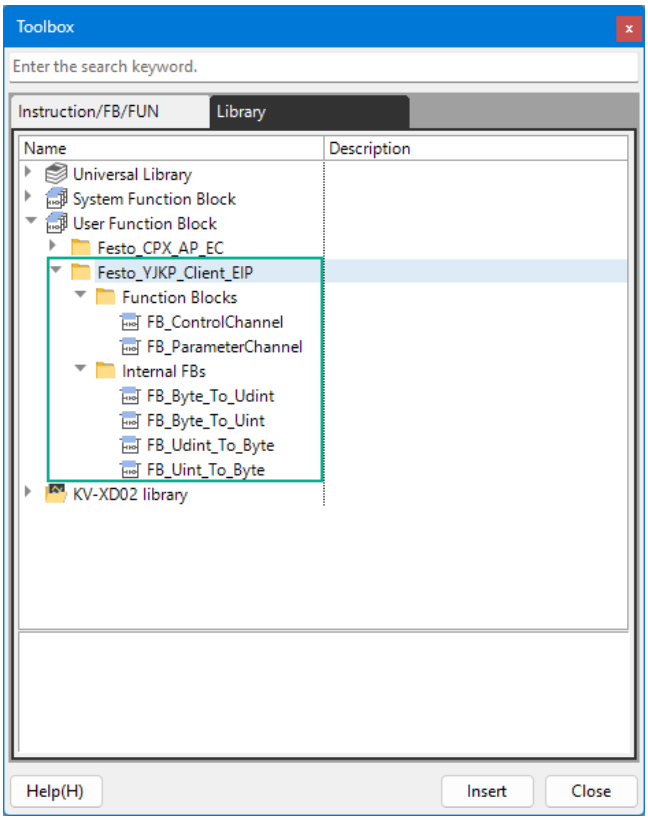
1. Locate the downloaded **Festo_YJKP_Client_EIP** library.
2. Select the **Library (Festo_YJKP_Client_EIP.kufb)** file from its location.
3. Click the **Open** button to install the selected library file.

Step 4



Click the **OK** button to complete the library import.

Step 5



The imported **Festo_YJKP_Client_EIP** library is available as highlighted.

The library contains the following function blocks and internal fb's :

Function Blocks

- **FB_ControlChannel** block provides press control functions for operating the Press System.
- **FB_ParameterChannel** block provides read and write access to the specified Press System objects.

Internal FBs

- **FB_Byte_To_Udint** block used to combine four bytes of data into a single UDINT value.
- **FB_Byte_To_Uint** block used to combine two bytes of data into a single UINT value.
- **FB_Udint_To_Byte** block used to split the single UDINT value into a four bytes of data.
- **FB_Uint_To_Byte** block used to split the single UINT data into a two bytes of data.

Union

- **Udint_To_Array** union data type used to convert a single UDINT value into an array of UINT values and vice versa.

The above internal FBs and union are user-defined data types that are used in the YJKP Client library function blocks.

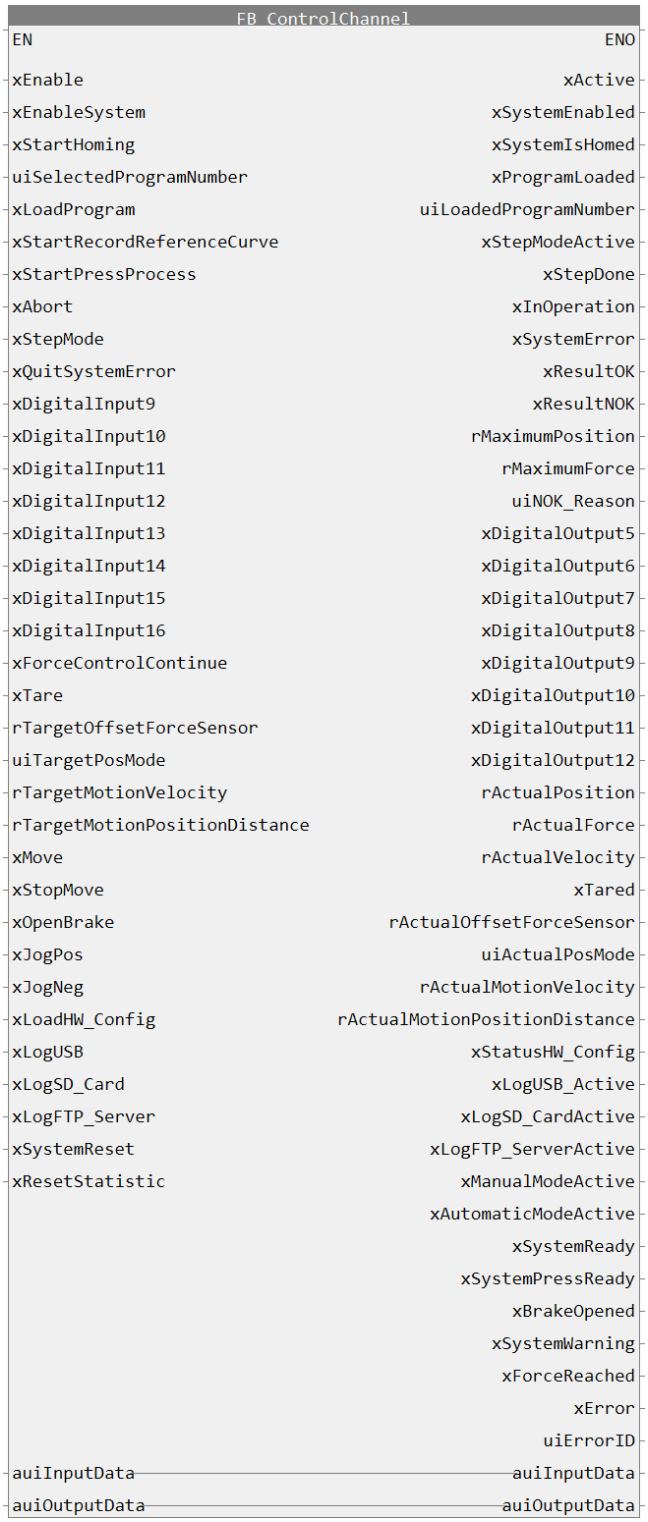
4 Festo_YJKP_Client_EIP Library Function Blocks Description

4.1 FB_ControlChannel

This function block provides press control functions for the Festo Press Control System.

Preparation:

- General:
  - Map the starting address of the input data to the function block input **auiInputData**.
  - Map the starting address of the output data to the function block input **auiOutputData**.



4.1.1 FB_ControlChannel Block In-Out Variables

In-Out Variables	Data Type	Description
auiInputData	ARRAY [0..63] OF UINT	Array of YJKP Input Data.
auiOutputData	ARRAY [0..63] OF UINT	Array of YJKP Output Data.

Table 4.1: FB_ControlChannel Block In-Out Variables descriptions

4.1.2 FB_ControlChannel Block Input Variables

Input Variables	Data Type	Description
xEnable	BOOL	Activate press control functions for the Festo system. **FALSE:** Action stopped, resets outputs. **TRUE:** Action running.
xEnableSystem	BOOL	Enable system command. **Falling edge:** Disable system. **Rising edge:** Enable system.
xStartHoming	BOOL	Homing command. **Falling edge:** No action. **Rising edge:** Start homing movement.
uiSelectedProgramNumber	UINT	Selected program number.
xLoadProgram	BOOL	Load program command. **Falling edge:** No action. **Rising edge:** Load selected program number.
xStartRecordReferenceCurve	BOOL	Start record reference curve command. **Falling edge:** No action. **Rising edge:** Start record reference curve.
xStartPressProcess	BOOL	Start press process command. **Falling edge:** No action. **Rising edge:** Start press process.
xAbort	BOOL	Abort command. **FALSE:** Abort press process or homing. **TRUE:** No action.
xStepMode	BOOL	Step mode command. **FALSE:** Deactivate step mode. **TRUE:** Activate step mode.
xQuitSystemError	BOOL	Quit system error command. **Falling edge:** No action. **Rising edge:** Quit upcoming system error.

Input Variables	Data Type	Description
xDigitalInput9	BOOL	Virtual digital input 9 for step enabling conditions in the sequencer of the press process.
xDigitalInput10	BOOL	Virtual digital input 10 for step enabling conditions in the sequencer of the press process.
xDigitalInput11	BOOL	Virtual digital input 11 for step enabling conditions in the sequencer of the press process.
xDigitalInput12	BOOL	Virtual digital input 12 for step enabling conditions in the sequencer of the press process.
xDigitalInput13	BOOL	Virtual digital input 13 for step enabling conditions in the sequencer of the press process.
xDigitalInput14	BOOL	Virtual digital input 14 for step enabling conditions in the sequencer of the press process.
xDigitalInput15	BOOL	Virtual digital input 15 for step enabling conditions in the sequencer of the press process.
xDigitalInput16	BOOL	Virtual digital input 16 for step enabling conditions in the sequencer of the press process.
xForceControlContinue	BOOL	Continue force control command. **Falling edge:** No action. **Rising edge:** Stop force control and move to next step without NOK evaluation although force control step is not finished.
xTare	BOOL	Tare of the force sensor command. **Falling edge:** Reset tare of the force sensor. **Rising edge:** Tare of the force sensor.
rTargetOffsetForceSensor	REAL	Target offset of the force sensor. **Unit [N]**.
uiTargetPosMode	UINT	Target position mode. **0x00 =** Jog. **0x01 =** Move absolute. **0x02 =** Move relative.
rTargetMotionVelocity	REAL	Target motion velocity for target position mode (Jog, Move, absolute/relative). **Unit [mm/s]**.
rTargetMotionPositionDistance	REAL	Target motion position/distance for target position mode (Move, absolute/relative). **Unit [mm]**.

Input Variables	Data Type	Description
xMove	BOOL	Move command. **Falling edge:** No action. **Rising edge:** Start movement depending on target position mode (Move absolute/relative), target motion velocity and target motion position/distance.
xStopMove	BOOL	Stop move command. **Falling edge:** No action. **Rising edge:** Stop movement depending on target position mode (Move absolute/relative).
xOpenBrake	BOOL	Open holding brake. **Falling edge:** Close holding brake. **Rising edge:** Open holding brake. This bit is ineffective when enable system command is true.
xJogPos	BOOL	Jog positive command. **Falling edge:** Stop movement depending on target position mode (**Jog**). **Rising edge:** Start movement in positive direction depending on target position mode (**Jog**) and target motion velocity.
xJogNeg	BOOL	Jog negative command. **Falling edge:** Stop movement depending on target position mode (**Jog**). **Rising edge:** Start movement in negative direction depending on target position mode (**Jog**) and target motion velocity.
xLoadHW_Config	BOOL	Load hardware configuration command. **Falling edge:** No action. **Rising edge:** Load hardware configuration (Configured hardware in window **Hardware configuration**).
xLogUSB	BOOL	Log USB command. **FALSE:** Deactivate log on USB-Stick. **TRUE:** Activate log on USB-Stick.
xLogSD_Card	BOOL	Log SD-Card command. **FALSE:** Deactivate log on SD-Card. **TRUE:** Activate log on SD-Card.
xLogFTP_Server	BOOL	Log FTP-Server command. **FALSE:** Deactivate log on FTP-Server. **TRUE:** Activate log on FTP-Server.

Input Variables	Data Type	Description
xSystemReset	BOOL	System reset command. **Falling edge:** No action. **Rising edge:** System reset.
xResetStatistic	BOOL	Reset statistic command. **Falling edge:** No action. **Rising edge:** Reset statistic (Reset statistic in window **Process diagnosis**).

Table 4.2: FB_ControlChannel Block Input Variables descriptions

4.1.3 FB_ControlChannel Block Output Variables

Output Variables	Data Type	Description
xActive	BOOL	Function block is active. **FALSE:** Function block is inactive. **TRUE:** Function block is active.
xSystemEnabled	BOOL	System is enabled **FALSE:** System is disabled. **TRUE:** System is enabled.
xSystemIsHomed	BOOL	System is homed **FALSE:** System is not homed. **TRUE:** System is homed.
xProgramLoaded	BOOL	Program is loaded. **Falling edge:** No action. **Rising edge:** Program is loaded.
uiLoadedProgramNumber	UINT	Loaded program number.
xStepModeActive	BOOL	Step mode is active. **FALSE:** Step mode is inactive. **TRUE:** Step mode is active.
xStepDone	BOOL	Step is done. **Falling edge:** No action. **Rising edge:** Step is done.
xInOperation	BOOL	System is in operation. **FALSE:** System is not in operation. **TRUE:** System is in operation.
xSystemError	BOOL	System error. **FALSE:** No error. **TRUE:** Error occurred.
xResultOK	BOOL	OK result of the press process.

Output Variables	Data Type	Description
xResultNOK	BOOL	NOK result of the press process. For further information see NOK reason.
rMaximumPosition	REAL	Maximum position of the press process. **Unit [mm].**
rMaximumForce	REAL	Maximum force of the press process. **Unit [N].**
uiNOK_Reason	UINT	NOK reason of the press process. **Digits 1...3** contain the step number of the press process being carried out (example: 010 = step 10). The fourth digit contains the actual reason why the press process was terminated. Refer Chapter 4.3.1.
xDigitalOutput5	BOOL	Virtual digital output 5 in the sequencer of the press process.
xDigitalOutput6	BOOL	Virtual digital output 6 in the sequencer of the press process.
xDigitalOutput7	BOOL	Virtual digital output 7 in the sequencer of the press process.
xDigitalOutput8	BOOL	Virtual digital output 8 in the sequencer of the press process.
xDigitalOutput9	BOOL	Virtual digital output 9 in the sequencer of the press process.
xDigitalOutput10	BOOL	Virtual digital output 10 in the sequencer of the press process.
xDigitalOutput11	BOOL	Virtual digital output 11 in the sequencer of the press process.
xDigitalOutput12	BOOL	Virtual digital output 12 in the sequencer of the press process.
rActualPosition	REAL	Actual position. **Unit [mm].**
rActualForce	REAL	Actual force. **Unit [N].**
rActualVelocity	REAL	Actual velocity. **Unit [mm/s].**
xTared	BOOL	Status of tare of the force sensor. **FALSE:** Force sensor not tared. **TRUE:** Force sensor tared.
rActualOffsetForceSensor	REAL	Actual offset of the force sensor. **Unit [N].**
uiActualPosMode	UINT	Actual position mode. **0x00 =** Jog. **0x01 =** Move absolute. **0x02 =** Move relative.
rActualMotionVelocity	REAL	Actual motion velocity for actual position mode (Jog, Move, absolute/relative). **Unit [mm/s].**

Output Variables	Data Type	Description
rActualMotionPositionDistance	REAL	Actual motion position/distance for actual position mode (Move, absolute/relative). **Unit [mm]**.
xStatusHW_Config	BOOL	Status of the hardware configuration. **FALSE:** Hardware configuration invalid. **TRUE:** Hardware configuration valid.
xLogUSB_Active	BOOL	Log on USB is active. **FALSE:** Log on USB-Stick is inactive. **TRUE:** Log on USB-Stick is active.
xLogSD_CardActive	BOOL	Log on SD-Card is active. **FALSE:** Log on SD-Card is inactive. **TRUE:** Log on SD-Card is active.
xLogFTP_ServerActive	BOOL	Log on FTP-Server is active. **FALSE:** Log on FTP-Server is inactive. **TRUE:** Log on FTP-Server is active.
xManualModeActive	BOOL	Manual mode active. **FALSE:** Manual mode is inactive. **TRUE:** Manual mode is active.
xAutomaticModeActive	BOOL	Automatic mode active. **FALSE:** Automatic mode is inactive. **TRUE:** Automatic mode is active.
xSystemReady	BOOL	System is ready. **FALSE:** System is not ready. **TRUE:** System is ready.
xSystemPressReady	BOOL	System servo press is ready. **FALSE:** System servo press is not ready. **TRUE:** System servo press is ready.
xBrakeOpened	BOOL	Holding brake is opened. **FALSE:** Holding brake is closed. **TRUE:** Holding brake is opened.
xSystemWarning	BOOL	System warning. **FALSE:** No warning. **TRUE:** Warning occurred.
xForceReached	BOOL	Force reached in force control. **FALSE:** Force not reached in force control. **TRUE:** Force reached in force control.

Output Variables	Data Type	Description
xError	BOOL	Error occurred during action. **FALSE:** No error. **TRUE:** Error occurred function block aborts action.
uiErrorID	UINT	Contains detailed error information in case of an error. Refer Chapter 4.3.4.

Table 4.3: FB_ControlChannel Block Output Variables descriptions

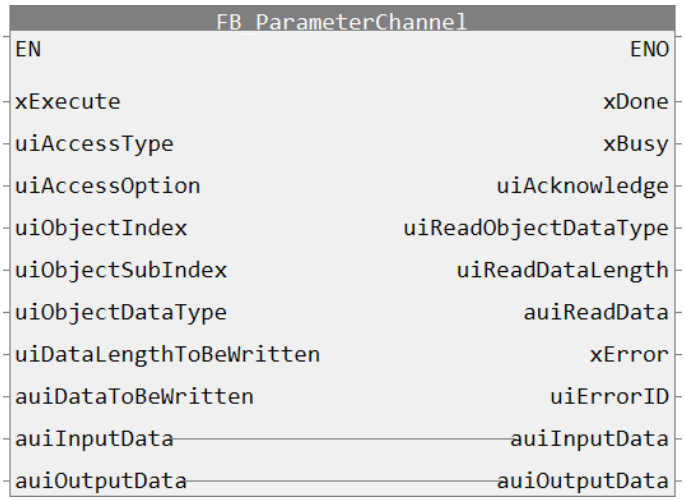
4.2 FB_ParameterChannel

This function block provides read/write access to the specified objects of the Festo Press Control System.

For detailed information of the available objects use the application note **YJKP – Object directory**.

Preparation:

- General:
  - Map the starting address of the input data to the function block input **auiInputData**.
  - Map the starting address of the output data to the function block input **auiOutputData**.



4.2.1 FB_ParameterChannel Block In-Out Variables

In-Out Variables	Data Type	Description
auiInputData	ARRAY [0..63] OF UINT	Array of YJKP Input Data.
auiOutputData	ARRAY [0..63] OF UINT	Array of YJKP Output Data.

Table 4.4: FB_ParameterChannel Block In-Out Variables descriptions

4.2.2 FB_ParameterChannel Block Input Variables

Input Variables	Data Type	Description
xExecute	BOOL	Start action for read/write object. **Falling edge:** Resets outputs. **Rising edge:** Action start.
uiAccessType	UINT	Access type. **0x00 =** Read object. **0x01 =** Write object.
uiAccessOption	UINT	Access option. **0x00 =** Single read/write object. **0x01 =** Multi read/write object.
uiObjectIndex	UINT	Object index.
uiObjectSubIndex	UINT	Object sub index.
uiObjectDataType	UINT	Object data type. Refer Chapter 4.3.3.
uiDataLengthToBeWritten	UINT	Data length of the value to be written.
auiDataToBeWritten	ARRAY [0..37] OF UINT	Data bytes of the value to be written.

Table 4.5: FB_ParameterChannel Block Input Variables descriptions

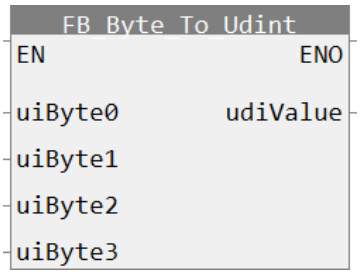
4.2.3 FB_ParameterChannel Block Output Variables

Output Variables	Data Type	Description
xDone	BOOL	Function block is done. **FALSE:** Function block is not done. **TRUE:** Function block is done.
xBusy	BOOL	Function block is busy. **FALSE:** Function block is not busy. **TRUE:** Function block is busy.
uiAcknowledge	UINT	Acknowledge. Refer Chapter 4.3.2.
uiReadObjectDataType	UINT	Read object data type. Refer Chapter 4.3.3.
uiReadDataLength	UINT	Data length of the read value.
auiReadData	ARRAY [0..37] OF UINT	Data bytes of the read value.
xError	BOOL	Error occurred during action. **FALSE:** No error. **TRUE:** Error occurred function block aborts action.
uiErrorID	UINT	Contains detailed error information in case of an error. Refer Chapter 4.3.4.

Table 4.6: FB_ParameterChannel Block Output Variables descriptions

4.3 FB_Byte_To_Udint

This block used to combine four bytes of data into a single UDINT value.



4.3.1 FB_Byte_To_Udint Block Input Variables

Input Variables	Data Type	Description
uiByte0	UINT	Low Word - Low Byte.
uiByte1	UINT	Low Word - High Byte.
uiByte2	UINT	High Word - Low Byte.
uiByte3	UINT	High Word - High Byte.

Table 4.7: FB_Byte_To_Udint Block Input Variables descriptions

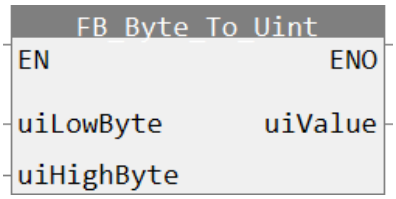
4.3.2 FB_Byte_To_Uint Block Output Variables

Output Variables	Data Type	Description
udiValue	UDINT	32 Bit UDINT Data Value.

Table 4.8: FB_Byte_To_Uint Block Output Variables description

4.4 FB_Byte_To_Uint

This block used to combine two bytes of data into a single UINT value.



4.4.1 FB_Byte_To_Uint Block Input Variables

Input Variables	Data Type	Description
uiLowByte	UINT	Low Byte of 16 Bit UINT Data.
uiHighByte	UINT	High Byte of 16 Bit UINT Data.

Table 4.9: FB_Byte_To_Uint Block Input Variables descriptions

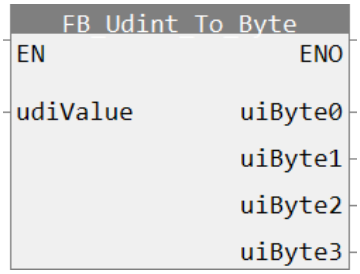
4.4.2 FB_Byte_To_Uint Block Output Variables

Output Variables	Data Type	Description
uiValue	UINT	16 Bit UINT Data Value.

Table 4.10: FB_Byte_To_Uint Block Output Variables description

4.5 FB_ Udint_To_Byte

This block used to split the single UDINT value into a four bytes of data.



4.5.1 FB_Udint_To_Byte Block Input Variables

Input Variables	Data Type	Description
udiValue	UDINT	32 Bit UDINT Data Value.

Table 4.11: FB_Udint_To_Byte Block Input Variables description

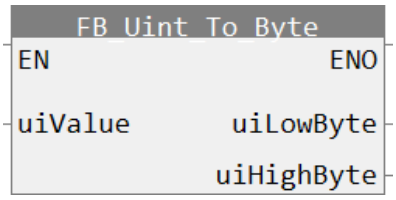
4.5.2 FB_ Udint_To_Byte Block Output Variables

Output Variables	Data Type	Description
uiByte0	UINT	Low Word - Low Byte.
uiByte1	UINT	Low Word - High Byte.
uiByte2	UINT	High Word - Low Byte.
uiByte3	UINT	High Word - High Byte.

Table 4.12: FB_ Udint_To_Byte Block Output Variables descriptions

4.6 FB_Uint_To_Byte

This block used to split the single UINT data into a two bytes of data.



4.6.1 FB_Uint_To_Byte Block Input Variables

Input Variables	Data Type	Description
uiValue	UINT	16 Bit UINT Data Value.

Table 4.13: FB_Uint_To_Byte Block Input Variables description

4.6.2 FB_Uint_To_Byte Block Output Variables

Output Variables	Data Type	Description
uiLowByte	UINT	Low Byte of 16 Bit UINT Data.
uiHighByte	UINT	High Byte of 16 Bit UINT Data.

Table 4.14: FB_Uint_To_Byte Block Output Variables descriptions

4.7 Union

4.7.1 Udint_To_Array

Union Variables	Data Type	Description
udiDataUdint	UDINT	32 Bit UDINT Data.
auiDataUint	ARRAY[0..1] OF UINT	UDINT Data as Array of UINT.

Table 4.15: Udint_To_Array descriptions

4.8 Enumerations

4.8.1 NOK Reason

NOK Reason 4th Digit Value	Description
16#00	Press process completed OK.
16#01	Termination by the **Position mode** step.
16#02	Termination by the **Force mode** step.
16#03	Termination by the **Digital signal mode** step.
16#04	Not applicable.
16#05	Not OK evaluation using **Windowing**.
16#06	Not OK evaluation using **Threshold**.
16#07	Not OK evaluation using **Envelope**
16#08	Termination of the press process.
16#09	Termination by the **Force control** step.
16#0A	Termination by the **Advanced force mode** step.
16#0B	Press process was terminated through error.

Table 4.16: NOK Reason descriptions

4.8.2 Acknowledge

Acknowledge Value	Meaning	Description
16#00	OK_OK	OK.
16#01	SERVICE_IS_NOT_SUPPORTED	Service is not supported | Check service ID in the request data.
16#02	DATA_LENGTH_INVALID	Data length invalid | Check data length in the request data.
16#03	OBJECT_INDEX_INVALID	Object index invalid | Check index and sub index of the object in the request data.
16#04	ACCESS_TYPE_INVALID	Access type invalid | Check access type of the object in the request data.
16#05	DATA_TYPE_INVALID	Data type invalid | Check data type of the object in the request data.
16#06	VALUE_INVALID	Value invalid | Check range of values of the object in the request data.
16#07	READ_TYPE_INVALID	Read type invalid | Check read type in the request data.
16#08	WRITE_TYPE_INVALID	Write type invalid | Check write type in the request data.
16#09	VALUES_INVALID	Values invalid | Check range of values of the object in the request data.

Table 4.17: Acknowledge descriptions

4.8.3 Data Type

Index Value	Data Type	Bytes	Description
16#00	UNKNOWN	-	Unknown data type.
16#01	TYPE_BOOL	1	8 bits Boolean (0 (FALSE) … 1 (TRUE)).
16#02	TYPE_SINT	1	8-bit signed short integer (-128 … 127).
16#03	TYPE_USINT	1	8-bit unsigned short integer (0 … 255).
16#04	TYPE_INT	2	16-bit signed integer (-32768 … 32767).
16#05	TYPE_UINT	2	16-bit unsigned integer (0 … 65535).
16#06	TYPE_DINT	4	32-bit signed long integer (-2147483648 … 2147483647).
16#07	TYPE_UDINT	4	32-bit unsigned long integer (0 … 4294967295).
16#08	TYPE_BYTE	1	8-bit unsigned short integer (0 … 255).
16#09	TYPE_WORD	2	16-bit unsigned integer (0 … 65535).
16#0A	TYPE_DWORD	4	32-bit unsigned long integer (0 … 4294967295).
16#0B	TYPE_REAL	4	32-bit float (1.400e-45 … 3.403e+38).

Table 4.18: Data Type descriptions

4.8.4 Error Table

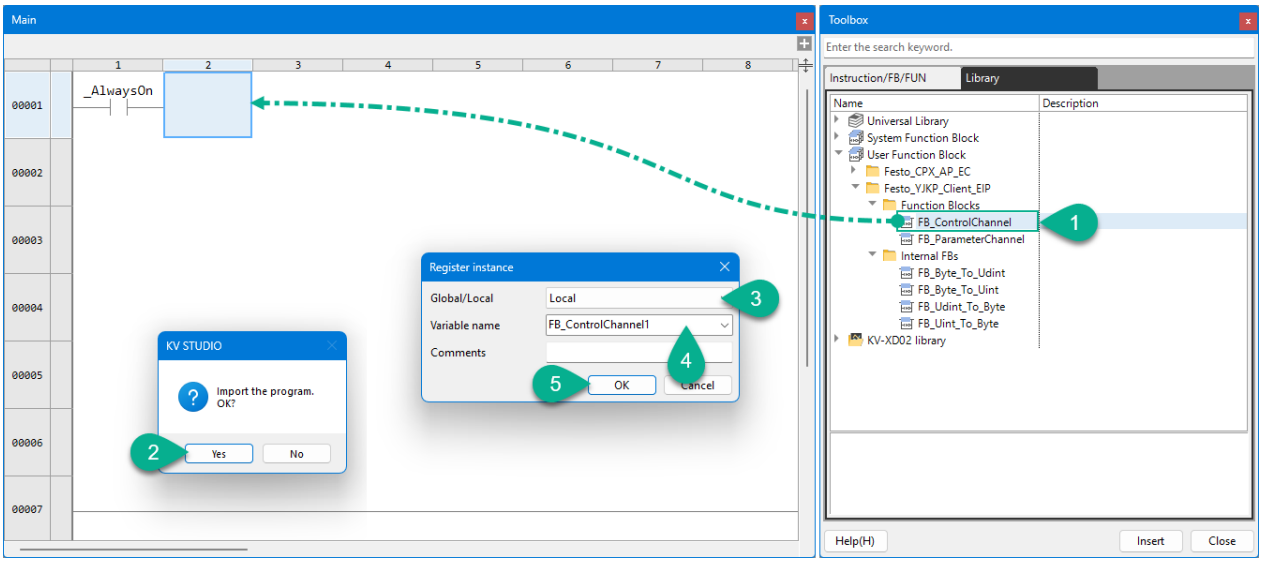
Error ID	Description
0x0000	No error / No error information available.
0x0001	Invalid state of the main state machine.
0x0002	Invalid state of the in-operation state machine.
0x0003	Invalid input data.
0x0004	Invalid output data.
0x0005	Requested resource is not available (e.g. file not found) / connection timeout.
0x0006	Drive fault / hardware fault.
0xFFFF	Other fault.

Table 4.19: Error Table descriptions

5 Configuration of YJKP Client Function Blocks in KV STUDIO Software

This chapter describes how to add the **Festo_YJKP_Client_EIP** library function blocks to the program and assign input and output tags.

Step 1



1. Drag and drop the **FB_ControlChannel** function block from the **Function Blocks** folder under the **Festo_YJKP_Client_EIP** library.
2. Click the **Yes** button to import the function block into the project.
3. Select the required **Variable Scope** from the **Global/Local** drop-down list in the **Register instance** window.
4. Enter the required function block instance name in the **Variable** name field.
5. Click the **OK** button to register the variable.

Step 2

The added function block will be shown with **Instance Name** as shown above.

Step 3

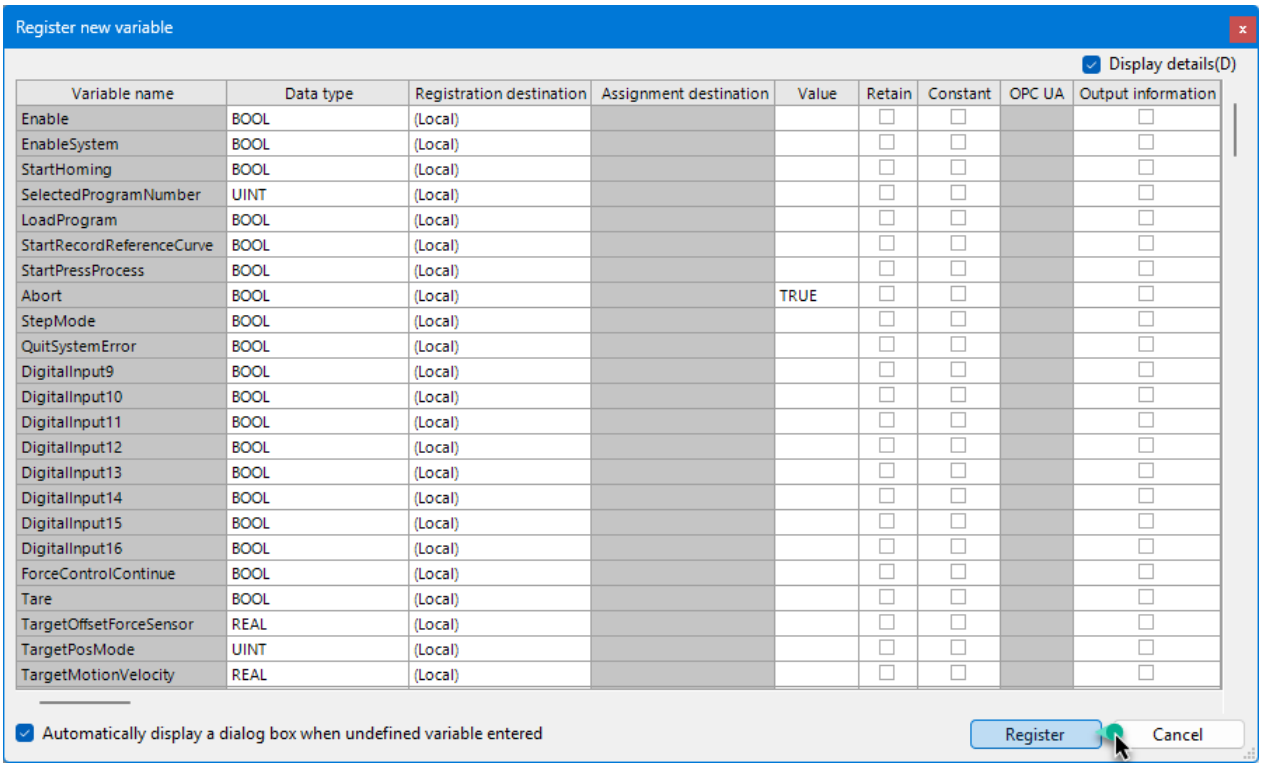
Enter the required names for function block interface tags as shown above.

Refer Chapter 3.3.3 for the function block I/O interface tags assignment.

Step 4

Select and right click the function block, click the **Register new variable(N)...** option to register all function block interface variables.

Step 5



Variable name	Data type	Registration destination	Assignment destination	Value	Retain	Constant	OPC UA	Output information
Enable	BOOL	(Local)			☐	☐		☐
EnableSystem	BOOL	(Local)			☐	☐		☐
StartHoming	BOOL	(Local)			☐	☐		☐
SelectedProgramNumber	UINT	(Local)			☐	☐		☐
LoadProgram	BOOL	(Local)			☐	☐		☐
StartRecordReferenceCurve	BOOL	(Local)			☐	☐		☐
StartPressProcess	BOOL	(Local)			☐	☐		☐
Abort	BOOL	(Local)		TRUE	☐	☐		☐
StepMode	BOOL	(Local)			☐	☐		☐
QuitSystemError	BOOL	(Local)			☐	☐		☐
DigitalInput9	BOOL	(Local)			☐	☐		☐
DigitalInput10	BOOL	(Local)			☐	☐		☐
DigitalInput11	BOOL	(Local)			☐	☐		☐
DigitalInput12	BOOL	(Local)			☐	☐		☐
DigitalInput13	BOOL	(Local)			☐	☐		☐
DigitalInput14	BOOL	(Local)			☐	☐		☐
DigitalInput15	BOOL	(Local)			☐	☐		☐
DigitalInput16	BOOL	(Local)			☐	☐		☐
ForceControlContinue	BOOL	(Local)			☐	☐		☐
Tare	BOOL	(Local)			☐	☐		☐
TargetOffsetForceSensor	REAL	(Local)			☐	☐		☐
TargetPosMode	UINT	(Local)			☐	☐		☐
TargetMotionVelocity	REAL	(Local)			☐	☐		☐

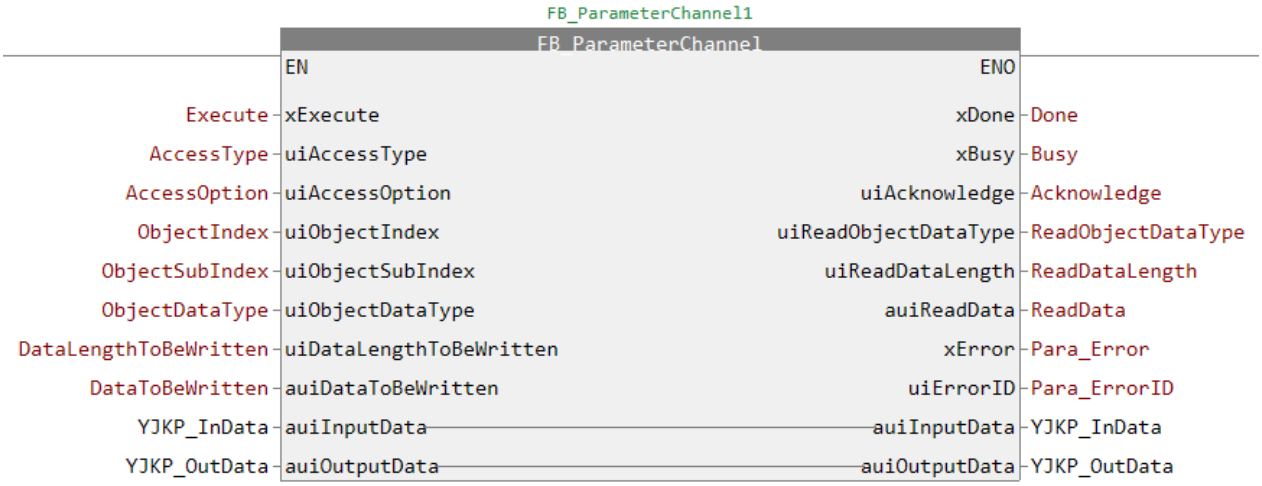
Click the **Register** button to register all the created variables.

Step 6

Once the function block interface variables have been registered successfully, the **FB_ControlChannel** function block appears as shown above.

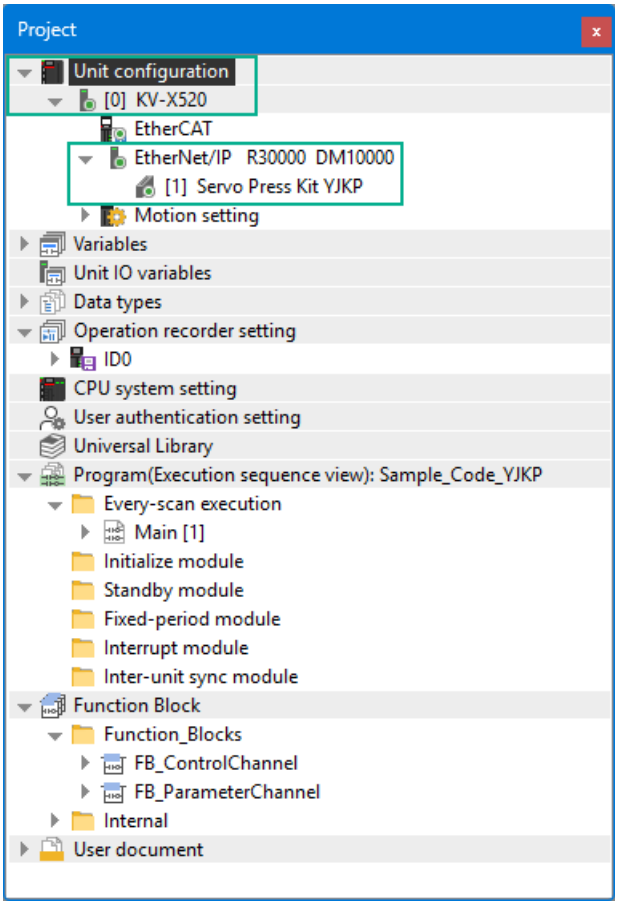
Follow the steps described above to configure all the function blocks in the **Festo_YJKP_Client_EIP** library.

Once all the steps are complete, the **FB_ParameterChannel** function block appears as shown below.



Once all configurations are complete, **build and download** the project to the controller.

After the project has been downloaded successfully, communication between the **Keyence KV-X PLC** and the **YJKP Servo Press Kit** device is established, as highlighted by the green box. The Project Workspace window appears as shown.



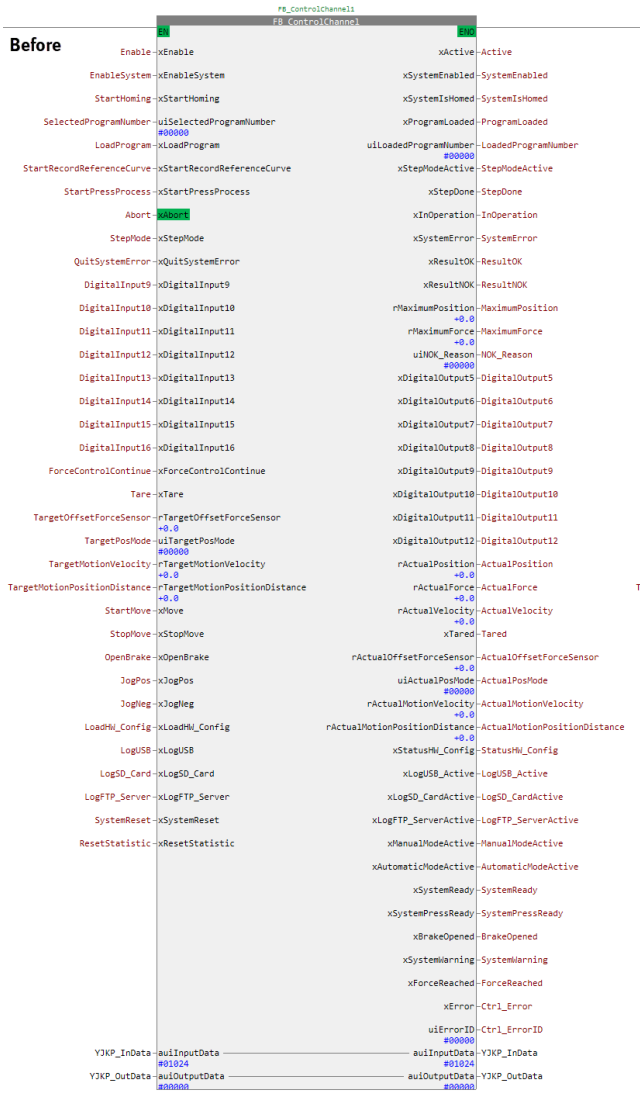
6 Sample Code for Festo_YJKP_Client_EIP Library Function Block

This chapter demonstrates how to enable the YJKP system, home the servo press, run the selected program in automatic mode, control the servo press in manual mode (jog, absolute, and relative movement), and read and write parameters using the function blocks from the **Festo_YJKP_Client_EIP** library..

6.1 Sample Code for FB_ControlChannel Block

6.1.1 Enabling the YJKP System

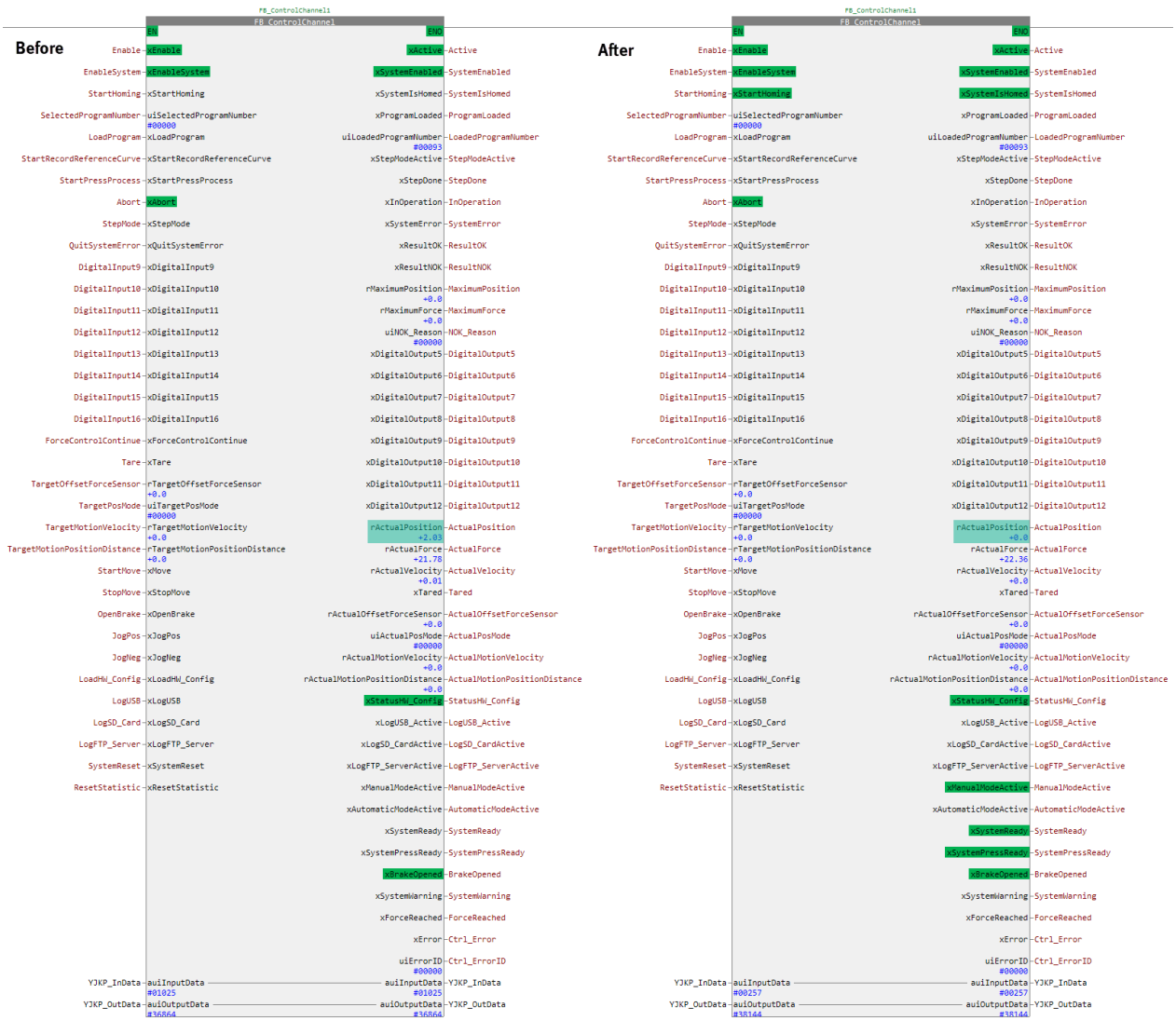
Before



After

1. Toggle the **xAbort** control tag to **TRUE**, and it should remain ON till end of the control process.
2. Toggle the **xEnable** control tag to **TRUE**, to enable the function block.
3. Once there is no error, **xActive** monitor tag will turns **TRUE**.
4. Toggle the **xEnableSystem** control tag to **TRUE**, to enable the YJKP system.
5. Once the YJKP system enabled successfully, **xSystemEnabled** and **xBrakeOpened** monitor tags will turns **TRUE**.

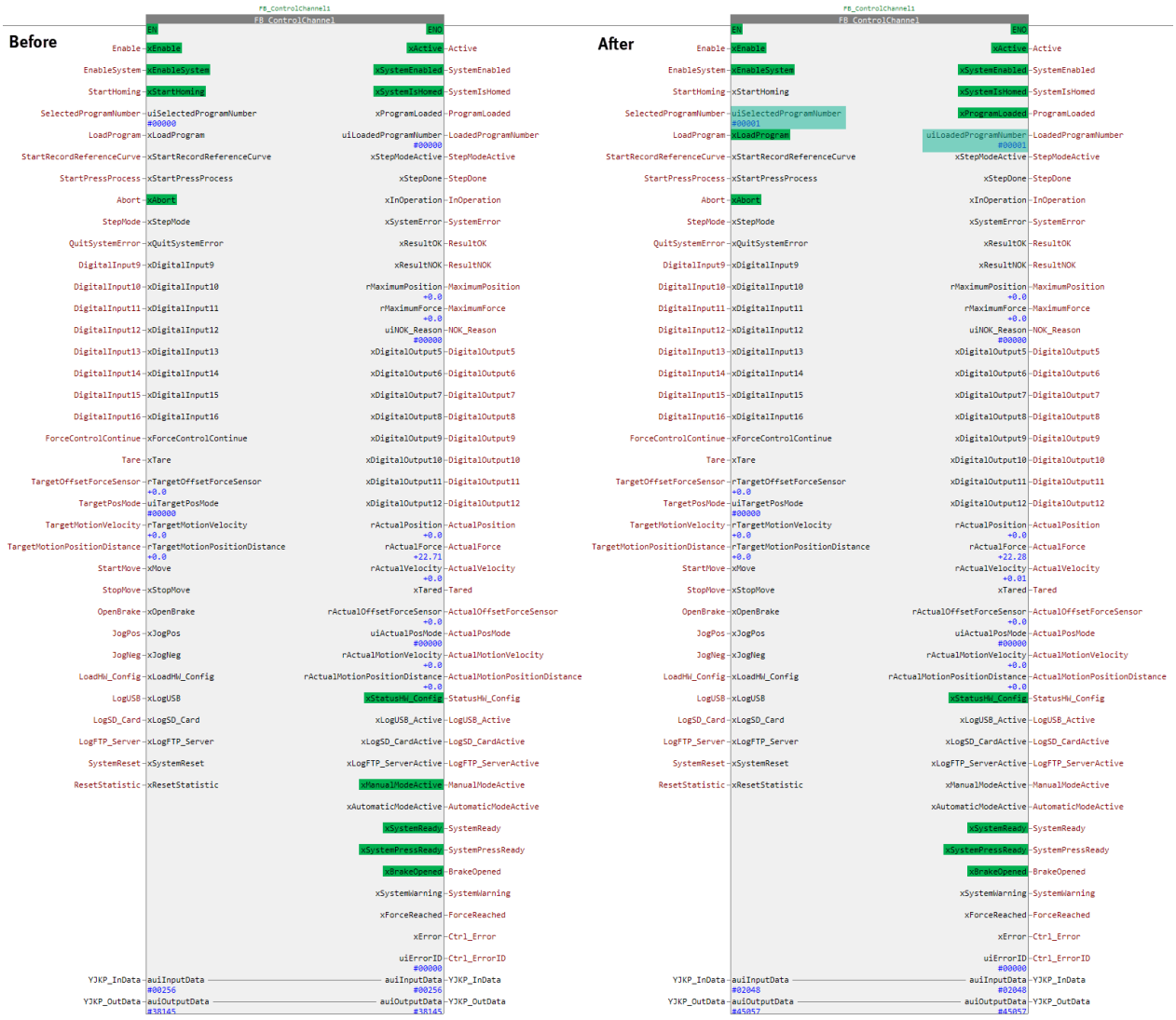
6.1.2 Sample Code for Homing the YJKP System



1. Toggle the **xStartHoming** control tag to **TRUE,** to home the YJKP press system.
2. Once homing is successful, **xSystemIsHomed** monitor tag will turns **TRUE**.
3. The **rActualPosition** monitor tag will be updated to the current position after homing (**0.0 mm**).
4. Once the YJKP system homed successfully, **xSystemReady** and **xSystemPressReady** monitor tags will turns **TRUE**.

6.1.3 Sample Code for Start Press Process

Step 1

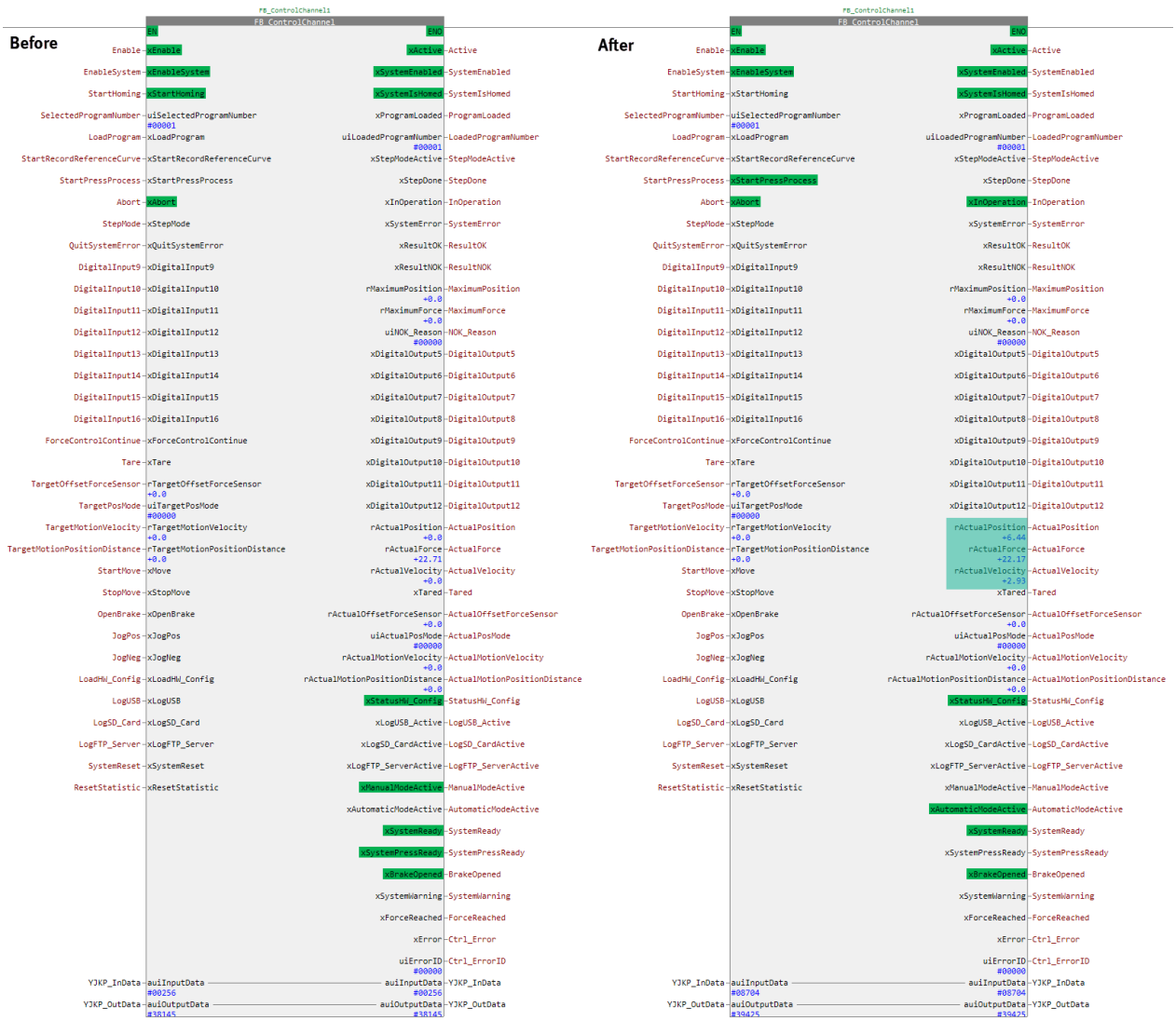


1. Enter the required Program Number defined in the YJKP Servo Press System. For this sample code, **Program Number 1** is entered.
2. Toggle the **xLoadProgram** control tag to **TRUE**, to load the selected program to the YJKP press system.
3. Once program loading is successful, **xProgramLoaded** monitor tag will turns **TRUE**.
4. The **uiLoadedProgramNumber** monitor tag will be updated to the value **1**.

Note

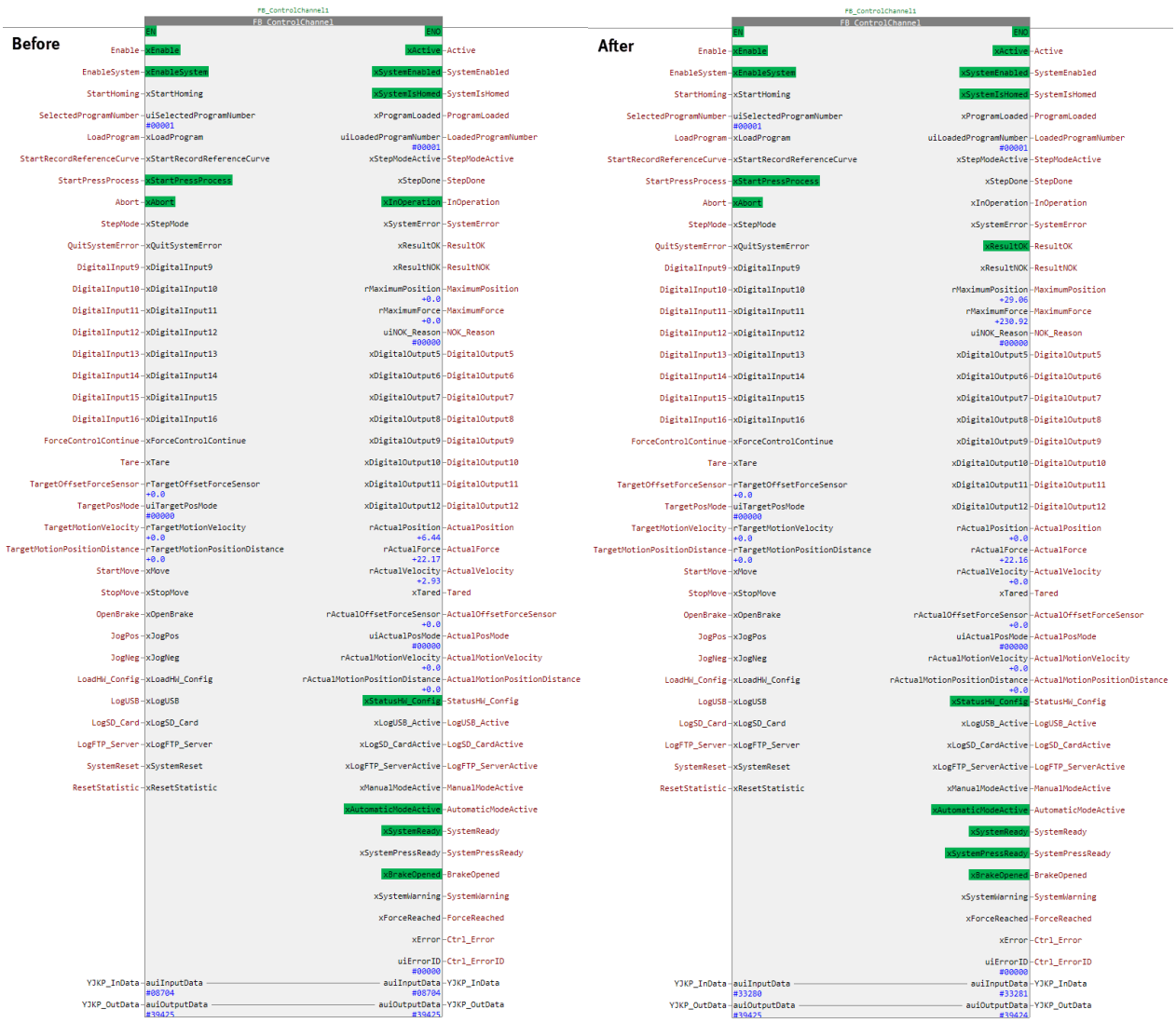
- Refer **YJKP – Creating program and descriptions of sequencer functions**, application note from the **Festo Support Portal** for defining the program in the YJKP Servo Press System.

Step 2



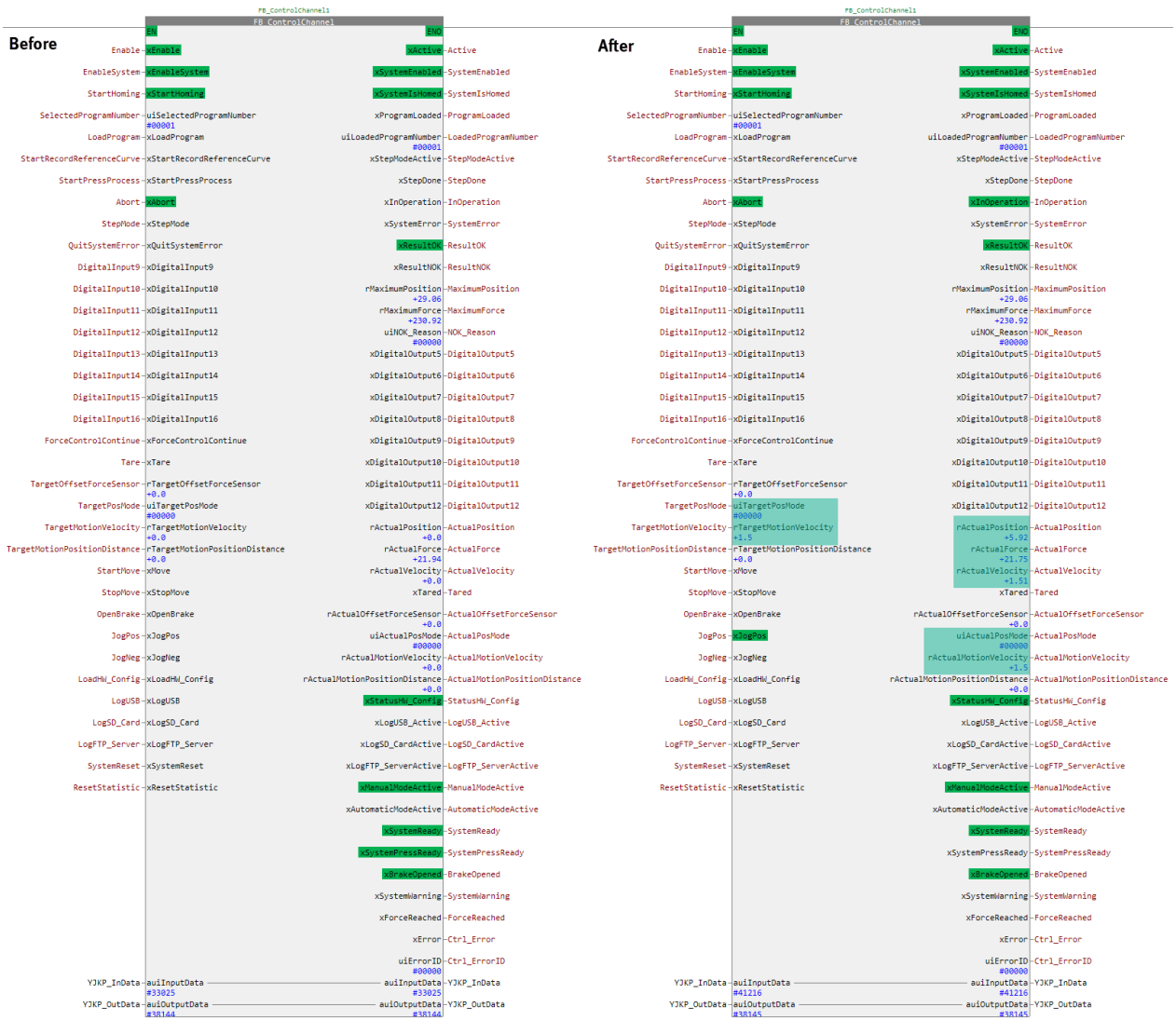
1. Toggle the **xStartPressProcess** control tag to **TRUE**, to run the selected program.
2. Once program started running, **xInOperation** and **xAutomaticModeActive** monitor tags will turns **TRUE**.
3. The **rActualPosition**, **rActualForce** and **rActualVelocity** monitor tags will be updated to the current **Position**, **Force** and **Velocity** values of the YJKP system.

Step 3



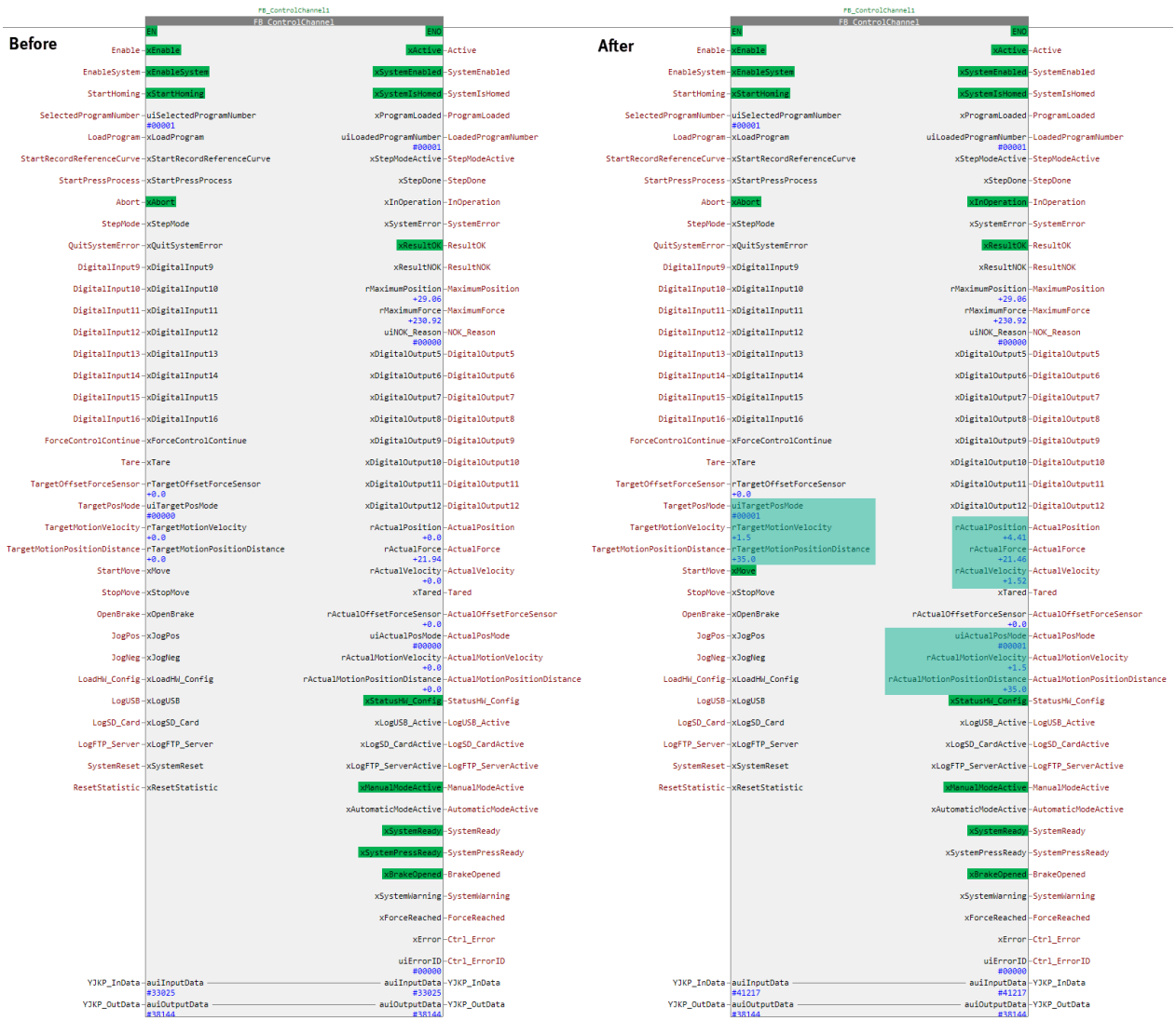
1. Once the Press Process is completed, **xInOperation** monitor tag will turns **FALSE**.
2. If the Press Process is successful, **xResultOK** monitor tag will turns **TRUE**.

6.1.4 Sample Code for Jog Movement



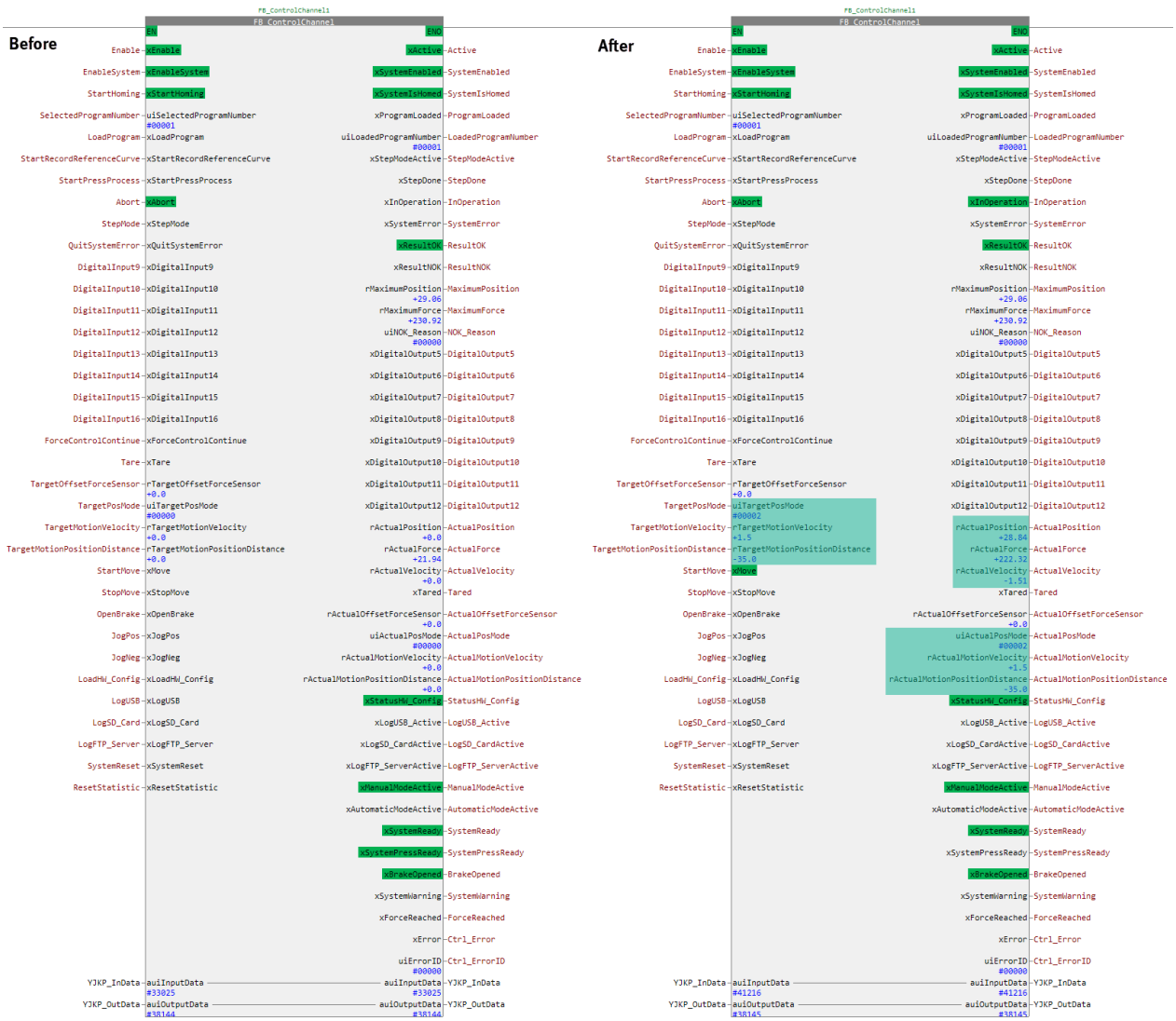
1. Enter the **uiTragetPosMode** control tag value as **0** (**Jog Mode**).
2. Enter the required **Target Velocity** value in the **rTargetMotionVelocity** control tag.
3. Toggle the **xJogPos** or **xJogNeg** control tags, to perform the **Jog** movement.
4. When jogging is running, the **xInOperation** and **xManualModeActive** monitor tags will turns **TRUE**.
5. The **uiActualPosMode** and **rActualMotionVelocity** monitor tags will be updated as the same as respective control tags.
6. The **rActualPosition**, **rActualForce** and **rActualVelocity** monitor tags will be updated to the current **Position**, **Force** and **Velocity** values of the YJKP system.

6.1.5 Sample Code for Absolute Movement



1. Enter the **uiTragetPosMode** control tag value as **1** (**Absolute Mode**).
2. Enter the required **Target Velocity** value in the **rTargetMotionVelocity** control tag.
3. Enter the required **Target Position** value in the **rTargetMotionPositionDistance** control tag.
4. Toggle the **xMove** control tags, to perform the **Absolute** movement.
5. When YJKP is running in the Absolute movement, the **xInOperation** and **xManualModeActive** monitor tags will turns **TRUE**.
6. The **uiActualPosMode**, **rActualMotionVelocity** and **rActualMotionPositionDistance** monitor tags will be updated as the same as respective control tags.
7. The **rActualPosition**, **rActualForce** and **rActualVelocity** monitor tags will be updated to the current **Position**, **Force** and **Velocity** values of the YJKP system.

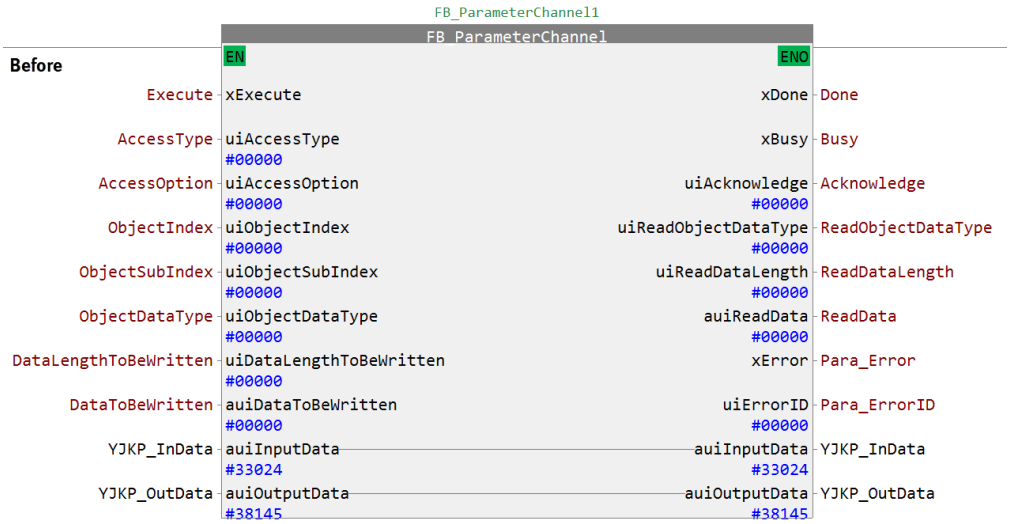
6.1.6 Sample Code for Relative Movement

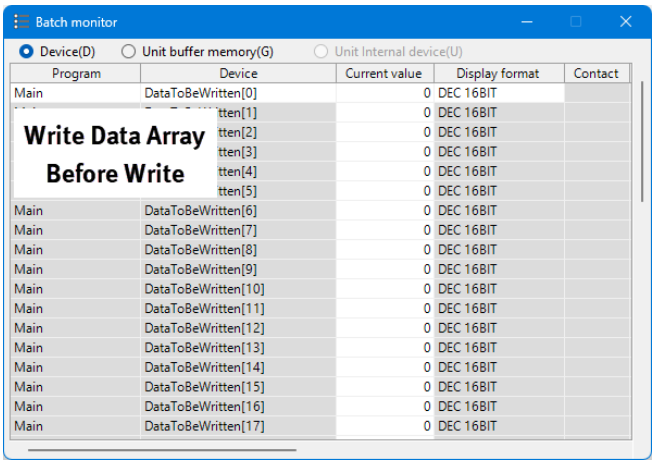


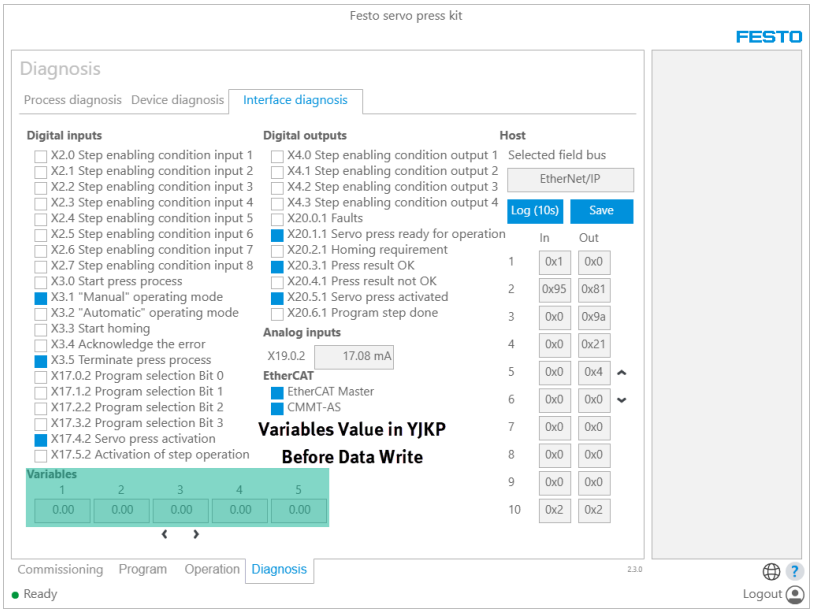
1. Enter the **uiTragetPosMode** control tag value as **2** (**Relative Mode**).
2. Enter the required **Target Velocity** value in the **rTargetMotionVelocity** control tag.
3. Enter the required **Target Position** value in the **rTargetMotionPositionDistance** control tag.
4. Toggle the **xMove** control tags, to perform the **Relative** movement.
5. When YJKP is running in the Relative movement, the **xInOperation** and **xManualModeActive** monitor tags will turns **TRUE**.
6. The **uiActualPosMode**, **rActualMotionVelocity** and **rActualMotionPositionDistance** monitor tags will be updated as the same as respective control tags.
7. The **rActualPosition**, **rActualForce** and **rActualVelocity** monitor tags will be updated to the current **Position**, **Force** and **Velocity** values of the YJKP system.

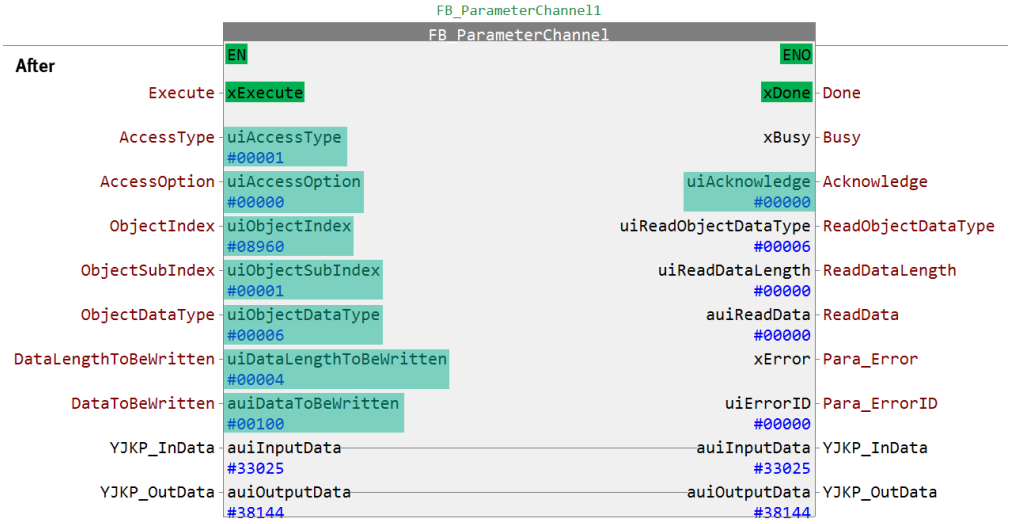
6.2 Sample Code for FB_ParameterChannel

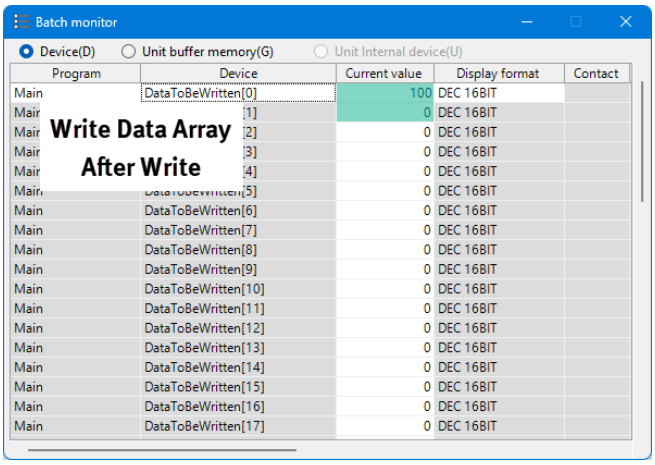
6.2.1 Write Single Variable

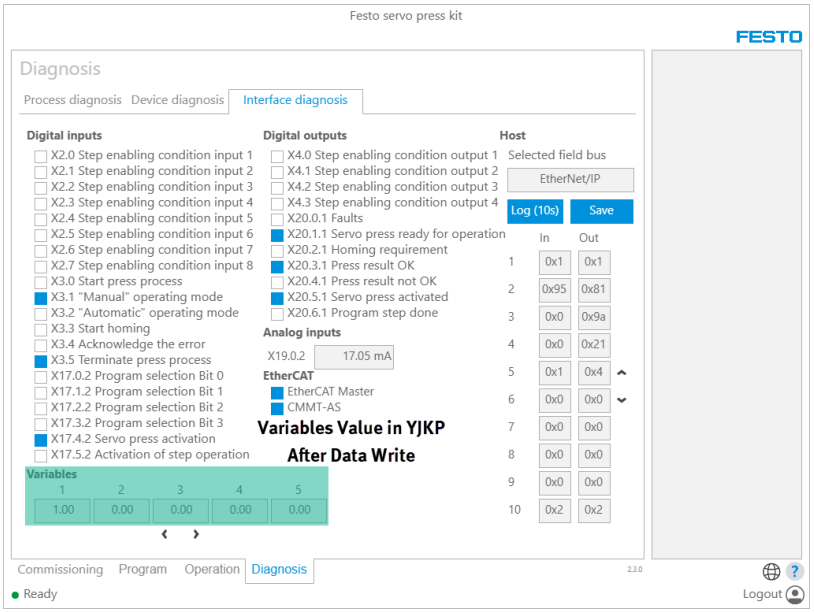












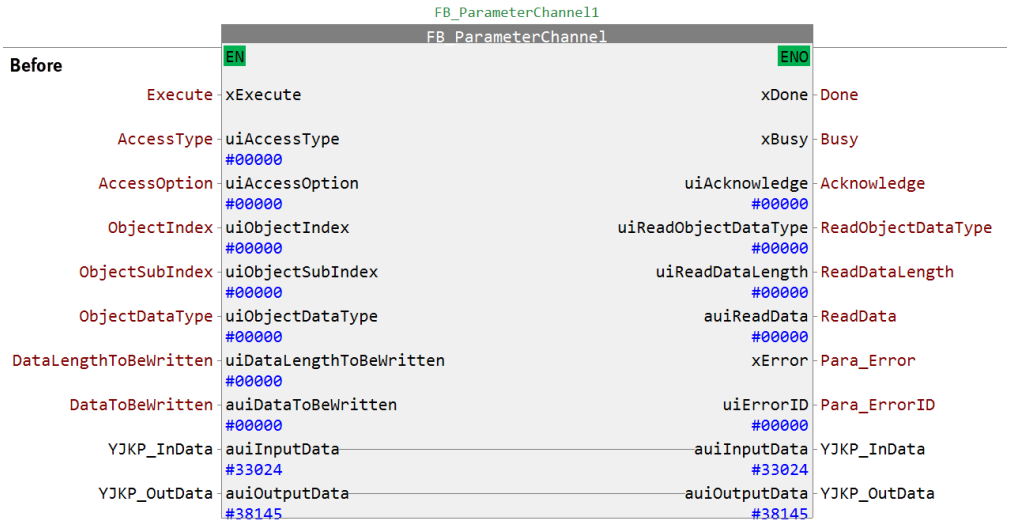
1. Enter the **uiAccessType** control tag value as **0x01** (**Write**) to enable the write function of the block.
2. Enter the **uiAccessOption** control tag value as **0x00** (**Single**) for writing single parameter.
3. Enter the required **Parameter Index** value in the **uiObjectIndex** control tag. For this application **0x2300** (**HEX**) or **8690** (**DEC**) (**Variables 1 – 10**) is used as the parameter index.
4. Enter the required **Parameter Subindex** value in the **uiObjectSubIndex** control tag. For this application **0x01** (**Variables 1**) is used as the parameter subindex.
5. Enter the **Parameter Data Type** value in the **uiObjectDataType** control tag. For this application **DINT** data type is used to write. So the value **6** has been set to the control tag. Refer Chapter 4.8.3.

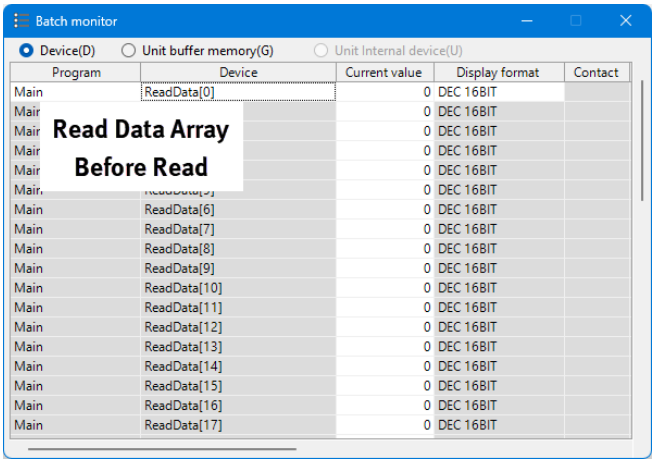
6. Enter the **Data Length** of the parameter to be written in the **uiDataLengthToBeWritten** control tag. For this application **DINT** data type is to be written. So the value **4** has been set to the control tag.
7. Enter the **Parameter Value** to be written in the **auiDataToBeWritten** control tag array as shown. In this example write data value is 100 (1.00).
8. Toggle the **xExecute** control tag, to **TRUE** to execute the write function.
9. Once the parameter written successfully, **xDone** monitor tag will turns **TRUE** and **uiAcknowledge** monitor tag will be **0**.

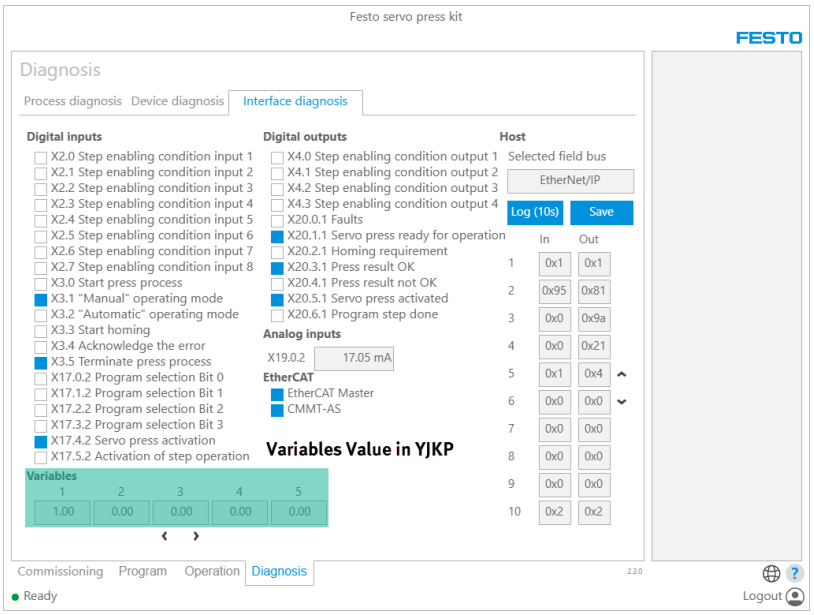
Note

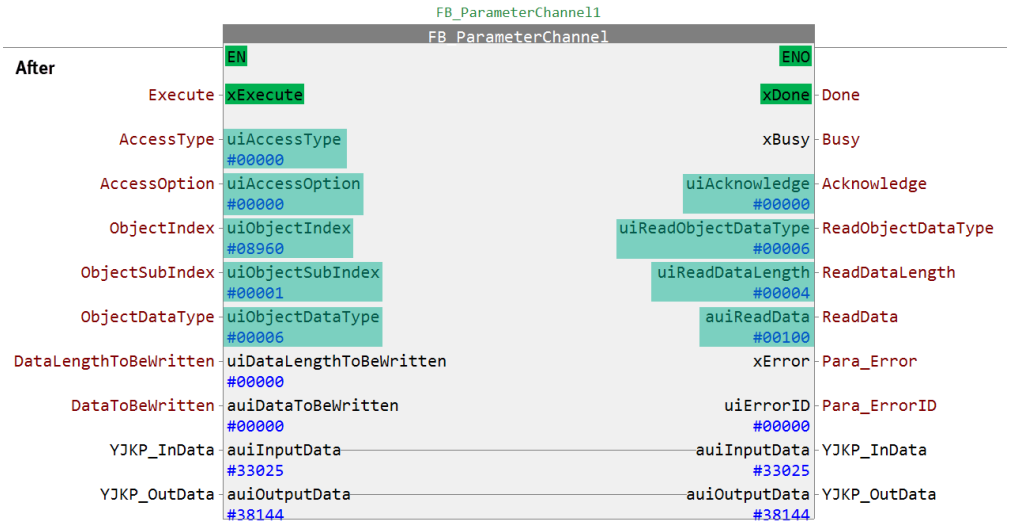
- The **FB_ParameterChannel** block provides the capability to read and write the parameters starting from index **0x2010** provided in the application note **YJKP - Object directory**.
- Data value is separated in to words (UINT) of array. For an example if user want to write a DINT value then the value has to split into 2 words (UINT).

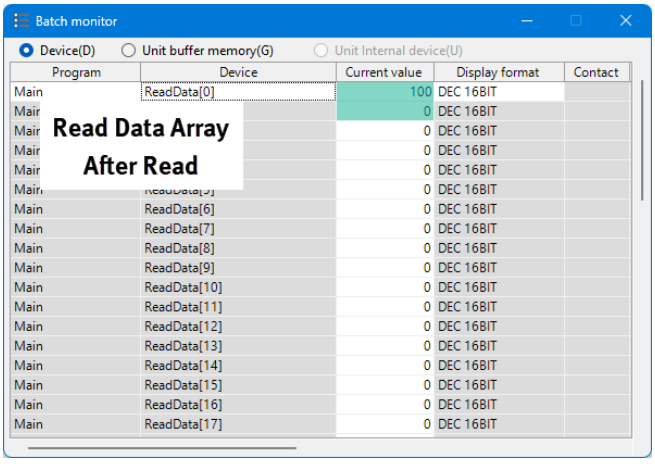
6.2.2 Read Single Variable











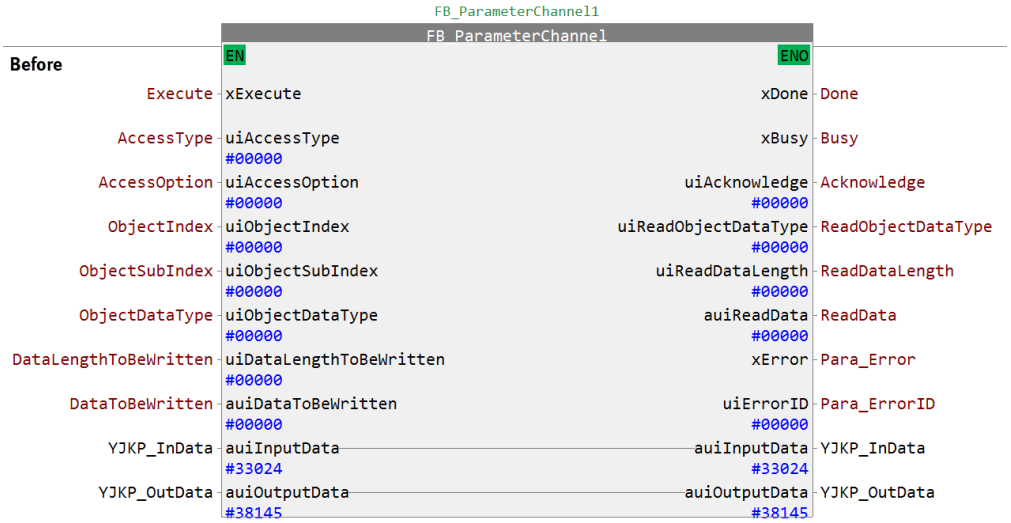
1. Enter the **uiAccessType** control tag value as **0x00** (**Read**) to enable the read function of the block.
2. Enter the **uiAccessOption** control tag value as **0x00** (**Single**) for reading single parameter.
3. Enter the required **Parameter Index** value in the **uiObjectIndex** control tag. For this application **0x2300** (**HEX**) or **8690** (**DEC**) (**Variables 1 – 10**) is used as the parameter index.
4. Enter the required **Parameter Subindex** value in the **uiObjectSubIndex** control tag. For this application **0x01** (**Variables 1**) is used as the parameter subindex.

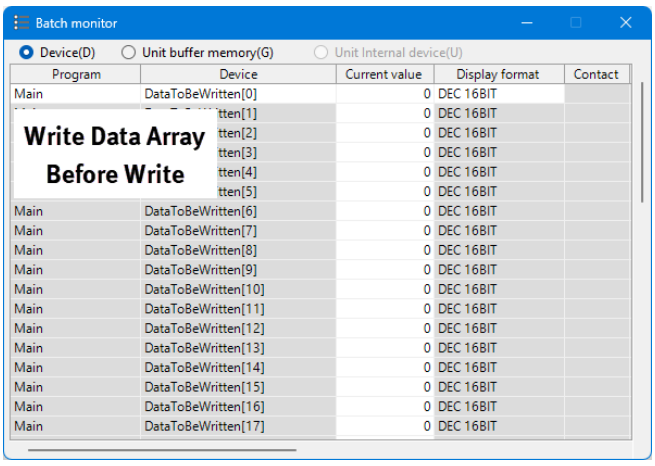
5. Enter the **Parameter Data Type** value in the **uiObjectDataType** control tag. For this application **DINT** data type is used to read. So the value **6** has been set to the control tag. Refer Chapter 4.8.3.
6. Toggle the **xExecute** control tag, to **TRUE** to execute the read function.
7. Once the parameter written successfully, **xDone** monitor tag will turns **TRUE** and **uiAcknowledge** monitor tag will be **0**.
8. The **uiReadObjectDataType** and **uiReadDataLength** monitor tags will be updated to the respective data type read and its length respectively. In this application note **DINT** data type is read so the values are **6** and **4** will be updated to the respective tags.
9. The **Parameter Value** will be update in **auiReadData** monitor tag array as shown.

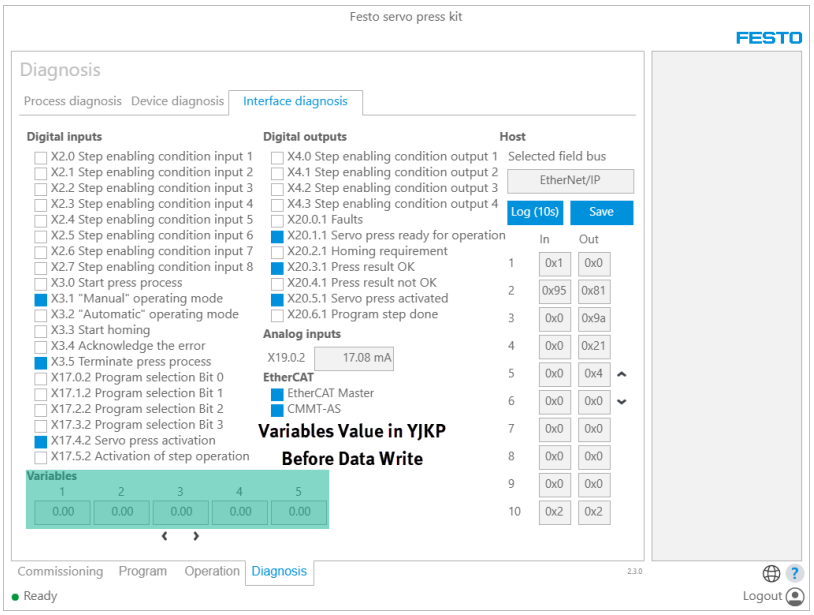
Note

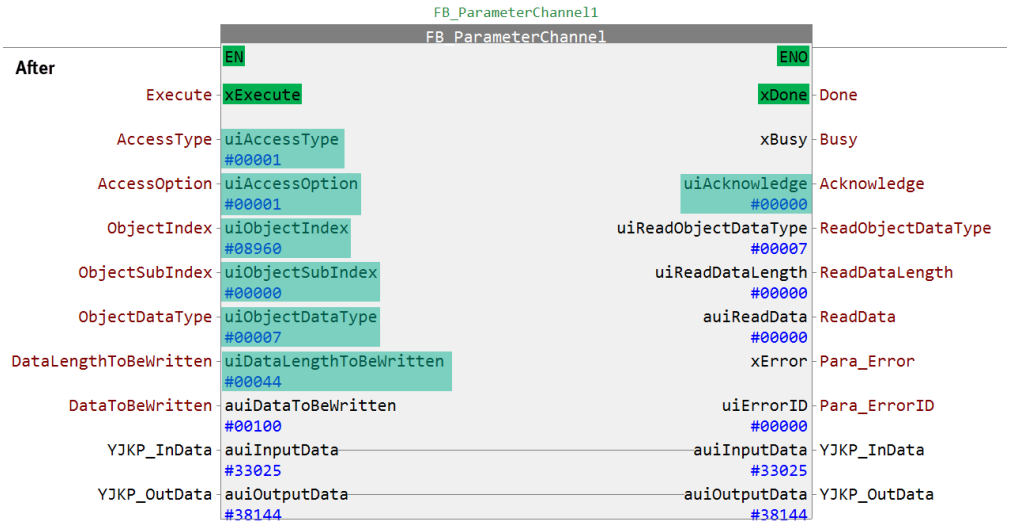
- The **FB_ParameterChannel** block provides the capability to read and write the parameters starting from index **0x2010** provided in the application note **YJKP - Object directory**.
- Data value is separated in to words (UINT) of array. For an example if user want to read a DINT value then the value has to split into 2 words (UINT).

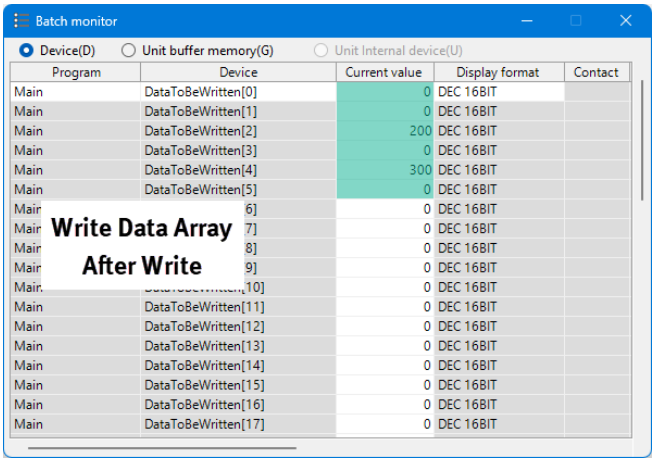
6.2.3 Write Multiple Variables

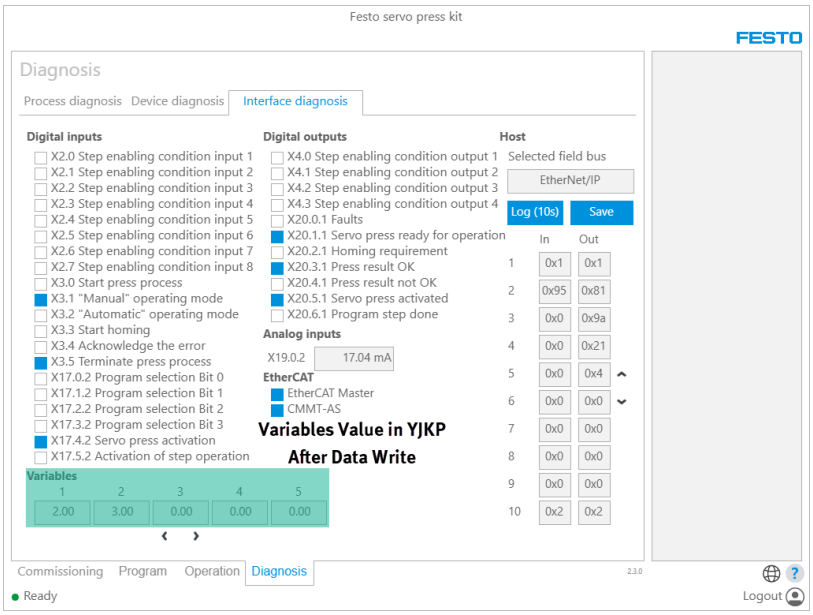





Festo servo press kit
FESTO
Diagnosis
Process diagnosis
Device diagnosis
Interface diagnosis
Digital inputs
X2.0 Step enabling condition input 1
X2.1 Step enabling condition input 2
X2.2 Step enabling condition input 3
X2.3 Step enabling condition input 4
X2.4 Step enabling condition input 5
X2.5 Step enabling condition input 6
X2.6 Step enabling condition input 7
X2.7 Step enabling condition input 8
X3.0 Start press process
X3.1 "Manual" operating mode
X3.2 "Automatic" operating mode
X3.3 Start homing
X3.4 Acknowledge the error
X3.5 Terminate press process
X17.0.2 Program selection Bit 0
X17.1.2 Program selection Bit 1
X17.2.2 Program selection Bit 2
X17.3.2 Program selection Bit 3
X17.4.2 Servo press activation
X17.5.2 Activation of step operation
Digital outputs
X4.0 Step enabling condition output 1
X4.1 Step enabling condition output 2
X4.2 Step enabling condition output 3
X4.3 Step enabling condition output 4
X20.0.1 Faults
X20.1.1 Servo press ready for operation
X20.2.1 Homing requirement
X20.3.1 Press result OK
X20.4.1 Press result not OK
X20.5.1 Servo press activated
X20.6.1 Program step done
Analog inputs
X19.0.2
17.08 mA
EtherCAT
EtherCAT Master
CMMT-AS
Variables Value in YJKP
Before Data Write
Variables
Host
Selected field bus
EtherNet/IP
Log (10s)
Save
In
Out
Commissioning
Program
Operation
Diagnosis
Ready
Logout


FB_ParameterChannel1
FB_ParameterChannel
After
EN
ENO
Execute
xExecute
xDone
Done
AccessType
uiAccessType
#00001
xBusy
Busy
AccessOption
uiAccessOption
#00001
uiAcknowledge
#00000
Acknowledge
ObjectIndex
uiObjectIndex
#08960
uiReadObjectDataType
#00007
ReadObjectDataType
ObjectSubIndex
uiObjectSubIndex
#00000
uiReadDataLength
#00000
ReadDataLength
ObjectDataType
uiObjectDataType
#00007
auiReadData
#00000
ReadData
DataLengthToBeWritten
uiDataLengthToBeWritten
#00044
xError
Para_Error
DataToBeWritten
auiDataToBeWritten
#00100
uiErrorID
#00000
Para_ErrorID
YJKP_InData
auiInputData
#33025
YJKP_OutData
auiOutputData
#38144


Batch monitor
Device(D)
Unit buffer memory(G)
Unit Internal device(U)
Program
Device
Current value
Display format
Contact
Main DataToBeWritten[0] 0 DEC 16BIT
Main DataToBeWritten[1] 0 DEC 16BIT
Main DataToBeWritten[2] 200 DEC 16BIT
Main DataToBeWritten[3] 0 DEC 16BIT
Main DataToBeWritten[4] 300 DEC 16BIT
Main DataToBeWritten[5] 0 DEC 16BIT
Write Data Array
After Write
Main DataToBeWritten[11] 0 DEC 16BIT
Main DataToBeWritten[12] 0 DEC 16BIT
Main DataToBeWritten[13] 0 DEC 16BIT
Main DataToBeWritten[14] 0 DEC 16BIT
Main DataToBeWritten[15] 0 DEC 16BIT
Main DataToBeWritten[16] 0 DEC 16BIT
Main DataToBeWritten[17] 0 DEC 16BIT



1. Enter the **uiAccessType** control tag value as **0x01** (**Write**) to enable the write function of the block.
2. Enter the **uiAccessOption** control tag value as **0x01** (**Multi**) for writing multiple parameters.
3. Enter the required **Parameter Index** value in the **uiObjectIndex** control tag. For this application **0x2300** (**HEX**) or **8690** (**DEC**) (**Variables 1 – 10**) is used as the parameter index.
4. Enter the required **Parameter Subindex** value in the **uiObjectSubIndex** control tag. For this application **0x00** (**Variables 1 – 10**) is used as the parameter subindex.
5. Enter the **Parameter Data Type** value in the **uiObjectDataType** control tag. For this application **UDINT** data type is used to write. So the value **7** has been set to the control tag. Refer Chapter 4.8.3.
6. Enter the **Data Length** of the parameter to be written in the **uiDataLengthToBeWritten** control tag. For this application **UDINT** data type is to be written for whole parameter subindexes **1 to 10**. So the value **44** has been set to the control tag.
7. Enter the **Parameter Value** to be written in the **auiDataToBeWritten** control tag array as shown starting from **auiDataToBeWritten[2]**.
8. Toggle the **xExecute** control tag, to **TRUE** to execute the write function.
9. Once the parameter written successfully, **xDone** monitor tag will turns **TRUE** and **uiAcknowledge** monitor tag will be **0**.

Note

- The **FB_ParameterChannel** block provides the capability to read and write the parameters starting from index **0x2010** provided in the application note **YJKP - Object directory**.
- Data value is separated in to words (UINT) of array. For an example if user want to write a DINT value then the value has to split into 2 words (UINT).

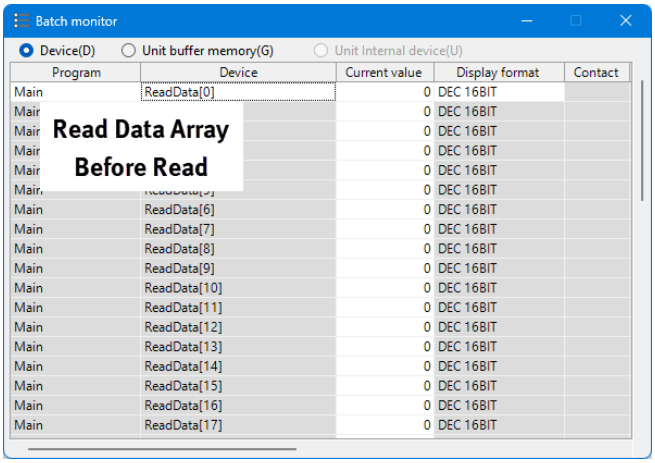
6.2.4 Read Multiple Variables

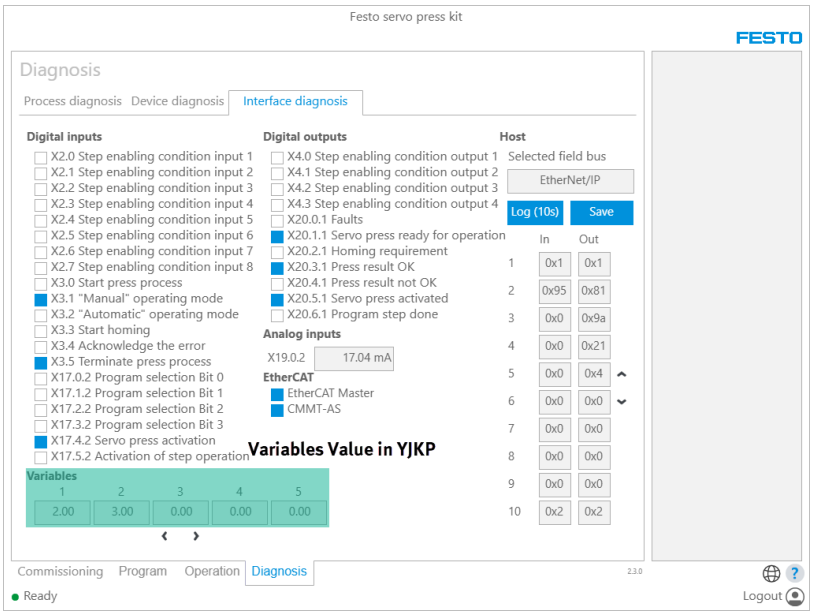
Before

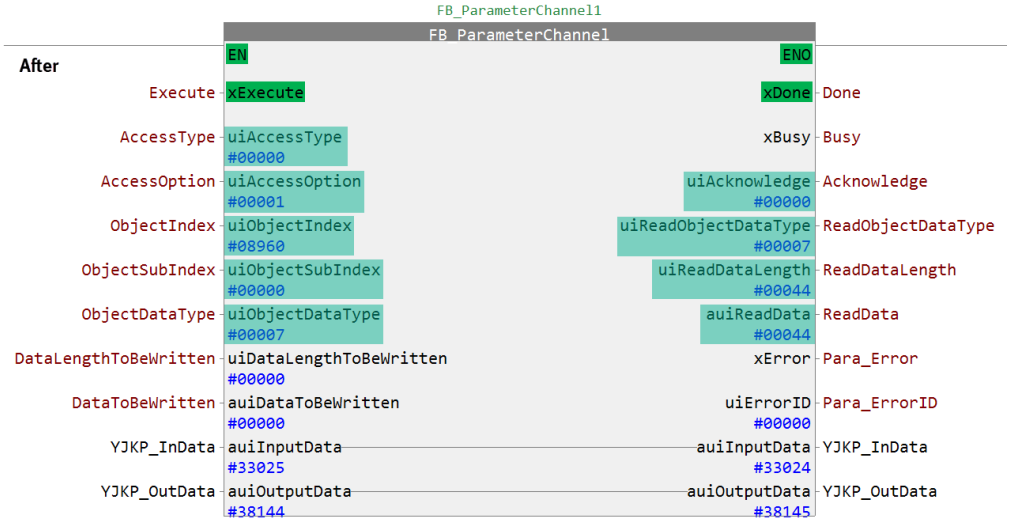
FB_ParameterChannel1

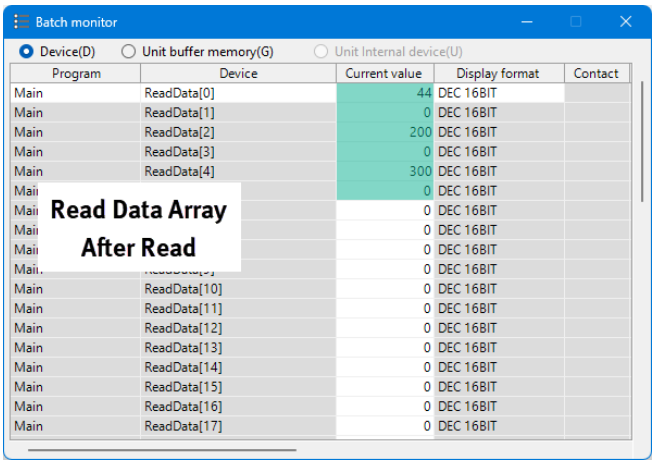
FB_ParameterChannel

	EN	ENO	
Execute	xExecute	xDone	Done
AccessType	uiAccessType #00000	xBusy	Busy
AccessOption	uiAccessOption #00000	uiAcknowledge #00000	Acknowledge
ObjectIndex	uiObjectIndex #00000	uiReadObjectDataType #00000	ReadObjectDataType
ObjectSubIndex	uiObjectSubIndex #00000	uiReadDataLength #00000	ReadDataLength
ObjectDataType	uiObjectDataType #00000	auiReadData #00000	ReadData
DataLengthToBeWritten	uiDataLengthToBeWritten #00000	xError	Para_Error
DataToBeWritten	auiDataToBeWritten #00000	uiErrorID #00000	Para_ErrorID
YJKP_InData	auiInputData #33024	auiInputData #33024	YJKP_InData
YJKP_OutData	auiOutputData #38145	auiOutputData #38145	YJKP_OutData









1. Enter the **uiAccessType** control tag value as **0x00** (**Read**) to enable the read function of the block.
2. Enter the **uiAccessOption** control tag value as **0x01** (**Multi**) for reading multiple parameters.
3. Enter the required **Parameter Index** value in the **uiObjectIndex** control tag. For this application **0x2300** (**HEX**) or **8690** (**DEC**) (**Variables 1 – 10**) is used as the parameter index.
4. Enter the required **Parameter Subindex** value in the **uiObjectSubIndex** control tag. For this application **0x00** (**Variables 1 – 10**) is used as the parameter subindex.
5. Enter the **Parameter Data Type** value in the **uiObjectDataType** control tag. For this application **UDINT** data type is used to read. So the value **7** has been set to the control tag. Refer Chapter 4.8.3.
6. Toggle the **xExecute** control tag, to **TRUE** to execute the read function.
7. Once the parameter written successfully, **xDone** monitor tag will turns **TRUE** and **uiAcknowledge** monitor tag will be **0**.
8. The **uiReadObjectDataType** and **uiReadDataLength** monitor tags will be updated to the respective data type read and its length respectively. In this application note **UINT** data type is read so the values are **7** and **44** (whole parameter index) will be updated to the respective tags.
9. The **Parameter Value** will be update in **auiReadData** monitor tag array as shown. The first four bytes of the read data will be the read data length.

➔ **Note**

- The **FB_ParameterChannel** block provides the capability to read and write the parameters starting from index **0x2010** provided in the application note **YJKP - Object directory**.
- Data value is separated in to words (UINT) of array. For an example if user want to read a DINT value then the value has to split into 2 words (UINT).

7 Procedure to Operate YJKP Client Function Blocks

7.1 Procedure of Press Control Functions through FB_ControlChannel

7.1.1 Load Hardware Configuration

1. Precondition:
   - Input **xEnable** is TRUE
   - Output **xActive** is TRUE
   - Output **xSystemError** is FALSE
   - Output **xInOperation** is FALSE
   - Output **xError** is FALSE
2. You have to be in the window **Hardware configuration** in the WebVisu of the Festo system
3. Set in the WebVisu your desired configuration
4. Set input **xLoadHW_Config** to TRUE
5. During operation output **xInOperation** is TRUE
6. Condition of load hardware configuration done
   - Check if output **xActive** is TRUE
   - Check if output **xInOperation** is FALSE
   - Check if output **xSystemError** is FALSE
   - Check if output **xError** is FALSE
7. Save the loaded hardware configuration via the button **Save** in the WebVisu

7.1.2 Activate Logging (For all logging functions the same)

1. Precondition:
   - Input **xEnable** is TRUE
   - Output **xActive** is TRUE
   - Output **xSystemError** is FALSE
   - Output **xInOperation** is FALSE
   - Output **xError** is FALSE
2. Set input **xLogSD_Card** to TRUE
3. Condition of logging on the SD-Card active
   - Check if output **xActive** is TRUE
   - Check if output **xLogSD_CardActive** is TRUE

7.1.3 System Reset

1. Precondition:
   - Input **xEnable** is TRUE
   - Output **xActive** is TRUE
   - Output **xSystemError** is FALSE
   - Output **xInOperation** is FALSE
   - Output **xError** is FALSE
2. Set input **xSystemReset** to TRUE
3. A reboot of the controller of the Festo system will be performed

7.1.4 Reset Statistic

1. Precondition:
   - Input **xEnable** is TRUE
   - Output **xActive** is TRUE
   - Output **xSystemError** is FALSE
   - Output **xInOperation** is FALSE
   - Output **xError** is FALSE
2. Set input **xResetStatistic** to TRUE
3. The statistic in the window **Process diagnosis** in the WebVisu of the Festo system will be reset

7.1.5 Enable System

1. Precondition:
   - Input **xEnable** is TRUE
   - Output **xActive** is TRUE
   - Output **xInOperation** is FALSE
   - Output **xSystemError** is FALSE
   - Output **xError** is FALSE
   - Output **xSystemPressReady** is TRUE
   - Output **xBrakeOpened** is FALSE
2. Set input **xEnableSystem** to TRUE
3. During operation output **xInOperation** is TRUE
4. Condition of system enabled
   - Check if output **xActive** is TRUE
   - Check if output **xInOperation** is FALSE
   - Check if output **xSystemError** is FALSE
   - Check if output **xError** is FALSE
   - Check if output **xSystemIsEnabled** is TRUE
   - Check if output **xBrakeOpened** is TRUE

7.1.6 Disable System

1. Precondition:
   - Input **xEnable** is TRUE
   - Input **xEnableSystem** is TRUE
   - Output **xActive** is TRUE
   - Output **xInOperation** is FALSE
   - Output **xSystemEnabled** is TRUE
   - Output **xSystemError** is FALSE
   - Output **xError** is FALSE
   - Output **xSystemPressReady** is TRUE
   - Output **xBrakeOpened** is TRUE
2. Set input **xEnableSystem** to FALSE
3. During operation output **xInOperation** is TRUE
4. Condition of system disabled
   - Check if output **xActive** is TRUE
   - Check if output **xInOperation** is FALSE
   - Check if output **xSystemError** is FALSE

- Check if output **xError** is FALSE
- Check if output **xSystemIsEnabled** is FALSE
- Check if output **xBrakeOpened** is FALSE

7.1.7 Tare Force Sensor

Caution

Not used for the press process, for the press process use the function inside the sequencer to tare the force sensor!!!

1. Precondition:
   - Input **xEnable** is TRUE
   - Output **xActive** is TRUE
   - Output **xInOperation** is FALSE
   - Output **xSystemError** is FALSE
   - Output **xError** is FALSE
2. Set input **xTare** to TRUE
3. Condition of force sensor tared
   - Check if output **xActive** is TRUE
   - Check if output **xTared** is TRUE

7.1.8 Add Offset to the Force Sensor

Caution

Not used for the press process, for the press process use the function inside the sequencer to tare the force sensor!!!

1. Precondition:
   - Input **xEnable** is TRUE
   - Output **xActive** is TRUE
   - Output **xInOperation** is FALSE
   - Output **xSystemError** is FALSE
   - Output **xError** is FALSE
2. Set input **rTargetOffsetForceSensor** to your desired offset (N)
3. Condition of offset added to the force sensor
   - Check if output **xActive** is TRUE
   - Check if output **rActualOffsetForceSensor** is old value plus **rTargetOffsetForceSensor**

7.1.9 Homing

1. Precondition:
   - Input **xEnable** is TRUE
   - Output **xActive** is TRUE
   - Output **xInOperation** is FALSE
   - Output **xSystemError** is FALSE
   - Output **xError** is FALSE
   - Output **xStatusHW_Config** is TRUE

2. Set input **xAbort** to TRUE
3. Enable the system by following Chapter 6.1.5.
4. Set input **xStartHoming** to TRUE
5. During operation output **xInOperation** is TRUE
6. Homing can be aborted via set input **xAbort** to FALSE
7. Condition of homing aborted
   - Check if output **xActive** is TRUE
   - Check if output **xInOperation** is FALSE
   - Check if output **xSystemError** is FALSE
   - Check if output **xError** is FALSE
   - Check if output **xSystemIsHomed** is FALSE
4. Condition of homing done
   - Check if output **xActive** is TRUE
   - Check if output **xInOperation** is FALSE
   - Check if output **xSystemError** is FALSE
   - Check if output **xError** is FALSE
   - Check if output **xSystemIsHomed** is TRUE

7.1.10 Jog

1. Precondition:
   - Input **xEnable** is TRUE
   - Input **xEnableSystem** is TRUE
   - Output **xActive** is TRUE
   - Output **xSystemEnabled** is TRUE
   - Output **xInOperation** is FALSE
   - Output **xSystemError** is FALSE
   - Output **xError** is FALSE
   - Output **xSystemPressReady** is TRUE
   - Output **xBrakeOpened** is TRUE
2. Set input **uiTargetPosMode** to 0x00 (Jog)
3. Set input **rTargetMotionVelocity** to your desired velocity (mm/s)
4. Set input **xJogPos** or **xJogNeg** to TRUE
5. During operation output **xInOperation** is TRUE
6. Jogging can be stopped via set input **xJogPos** or **xJogNeg** to FALSE
7. Condition of jogging stopped or done
   - Check if output **xActive** is TRUE
   - Check if output **xInOperation** is FALSE
   - Check if output **xSystemError** is FALSE
   - Check if output **xError** is FALSE

7.1.11 Move Absolute

1. Precondition:
   - Input **xEnable** is TRUE
   - Input **xEnableSystem** is TRUE
   - Output **xActive** is TRUE
   - Output **xSystemEnabled** is TRUE
   - Output **xInOperation** is FALSE

- Output **xSystemError** is FALSE
- Output **xError** is FALSE
- Output **xSystemPressReady** is TRUE
- Output **xBrakeOpened** is TRUE

2. Set input **uiTargetPosMode** to 0x01 (Move absolute)
3. Set input **rTargetMotionVelocity** to your desired velocity (mm/s)
4. Set input **rTargetMotionPositionDistance** to your desired position (mm)
5. Set input **xStopMove** to FALSE
6. Set input **xMove** to TRUE
7. During operation output **xInOperation** is TRUE
8. Move absolute can be stopped via set input **xStopMove** to TRUE
9. Condition of move absolute stopped
   - Check if output **xActive** is TRUE
   - Check if output **xInOperation** is FALSE
   - Check if output **xSystemError** is FALSE
   - Check if output **xError** is FALSE
   - Check if output **rActualPosition** is unequal to input **rTargetMotionPositionDistance**
10. Condition of move absolute done
    - Check if output **xActive** is TRUE
    - Check if output **xInOperation** is FALSE
    - Check if output **xSystemError** is FALSE
    - Check if output **xError** is FALSE
    - Check if output **rActualPosition** is equal to input **rTargetMotionPositionDistance**

7.1.12 Move Relative

1. Precondition:
   - Input **xEnable** is TRUE
   - Input **xEnableSystem** is TRUE
   - Output **xActive** is TRUE
   - Output **xSystemEnabled** is TRUE
   - Output **xInOperation** is FALSE
   - Output **xSystemError** is FALSE
   - Output **xError** is FALSE
   - Output **xSystemPressReady** is TRUE
   - Output **xBrakeOpened** is TRUE
2. Set input **uiTargetPosMode** to 0x02 (Move relative)
3. Set input **rTargetMotionVelocity** to your desired velocity (mm/s)
4. Set input **rTargetMotionPositionDistance** to your desired distance (mm)
5. Set input **xStopMove** to FALSE
6. Set input **xMove** to TRUE
7. During operation output **xInOperation** is TRUE
8. Move relative can be stopped via set input **xStopMove** to TRUE
9. Condition of move relative stopped
   - Check if output **xActive** is TRUE
   - Check if output **xInOperation** is FALSE
   - Check if output **xSystemError** is FALSE
   - Check if output **xError** is FALSE

- Check if output **rActualPosition** is unequal to output **rActualPosition** plus input **rTargetMotionPositionDistance**

10. Condition of move relative done
    - Check if output **xActive** is TRUE
    - Check if output **xInOperation** is FALSE
    - Check if output **xSystemError** is FALSE
    - Check if output **xError** is FALSE
    - Check if output **rActualPosition** is equal to output **rActualPosition** plus input **rTargetMotionPositionDistance**

7.1.13 Load Program

1. Precondition:
    - Input **xEnable** is TRUE
    - Output **xActive** is TRUE
    - Output **xInOperation** is FALSE
    - Output **xSystemError** is FALSE
    - Output **xError** is FALSE
2. Set input **uiSelectedProgramNumber** to your desired program number
3. Set input **xLoadProgram** to TRUE
4. During operation output **xInOperation** is TRUE
5. Condition of program loaded
    - Check if output **xActive** is TRUE
    - Check if output **xInOperation** is FALSE
    - Check if output **xSystemError** is FALSE
    - Check if output **xError** is FALSE
    - Check if output **uiLoadedProgramNumber** is input **uiSelectedProgramNumber**

7.1.14 Start Press Process

1. Precondition:
    - Input **xEnable** is TRUE
    - Input **xEnableSystem** is TRUE
    - Output **xActive** is TRUE
    - Output **xInOperation** is FALSE
    - Output **xSystemError** is FALSE
    - Output **xError** is FALSE
    - Output **xSystemPressReady** is TRUE
    - Output **xBrakeOpened** is TRUE
2. Check if output **uiLoadedProgramNumber** is set to your desired program number
3. Set input **xAbort** to TRUE
4. Set input **xStartPressProcess** to TRUE
5. During operation output **xInOperation** is TRUE
6. Press process can be aborted via set input **xAbort** to FALSE
7. Condition of press process aborted
    - Check if output **xActive** is TRUE
    - Check if output **xInOperation** is FALSE
    - Check if output **xSystemError** is FALSE
    - Check if output **xError** is FALSE

- Check if output **xResultNOK** is TRUE

8. Condition of press process done
   - Check if output **xActive** is TRUE
   - Check if output **xInOperation** is FALSE
   - Check if output **xSystemError** is FALSE
   - Check if output **xError** is FALSE
   - Check if output **xResultOK** is TRUE or output **xResultNOK** is TRUE

7.1.15 Record Reference Curve

1. Precondition:
   - Input **xEnable** is TRUE
   - Input **xEnableSystem** is TRUE
   - Output **xActive** is TRUE
   - Output **xInOperation** is FALSE
   - Output **xSystemError** is FALSE
   - Output **xError** is FALSE
   - Output **xSystemPressReady** is TRUE
   - Output **xBrakeOpened** is TRUE
2. You have to be in the window **Step 2/4: Record / loading reference curve(s)** in the Festo system
3. Set input **xAbort** to TRUE
4. Set input **xRecordReferenceCurve** to TRUE
5. During operation output **xInOperation** is TRUE
6. Record reference curve can be aborted via set input **xAbort** to FALSE
7. Condition of record reference curve aborted or done
   - Check if output **xActive** is TRUE
   - Check if output **xInOperation** is FALSE
   - Check if output **xSystemError** is FALSE
   - Check if output **xError** is FALSE

7.1.16 Step Mode

1. Precondition:
   - Input **xEnable** is TRUE
   - Input **xEnableSystem** is TRUE
   - Output **xActive** is TRUE
   - Output **xSystemEnabled** is TRUE
   - Output **xInOperation** is FALSE
   - Output **xSystemError** is FALSE
   - Output **xError** is FALSE
   - Output **xSystemPressReady** is TRUE
   - Output **xBrakeOpened** is TRUE
2. Check if output **uiLoadedProgramNumber** is set to your desired program number
3. Set input **xAbort** to TRUE
4. Set input **xStepMode** to TRUE
5. Check if output **xStepModeActive** is TRUE
6. Set input **xStartPressProcess** to TRUE
7. During operation output **xInOperation** is TRUE
8. During step processing output **xStepDone** is FALSE

9. Press process can be aborted via set input **xAbort** to FALSE
10. Condition of step done
    - Check if output **xActive** is TRUE
    - Check if output **xStepDone** is TRUE
    - Check if output **xSystemError** is FALSE
    - Check if output **xError** is FALSE
11. Set input **xStartPressProcess** to FALSE
12. Set input **xStartPressProcess** to TRUE
13. During step processing output **xStepDone** is FALSE
14. Condition of step done
    - Check if output **xActive** is TRUE
    - Check if output **xStepDone** is TRUE
    - Check if output **xSystemError** is FALSE
    - Check if output **xError** is FALSE
15. Repeat step 11 to14 until all steps are process and condition of press process done is reached
16. Condition of press process aborted
    - Check if output **xActive** is TRUE
    - Check if output **xInOperation** is FALSE
    - Check if output **xSystemError** is FALSE
    - Check if output **xError** is FALSE
    - Check if output **xResultNOK** is TRUE
17. Condition of press process done
    - Check if output **xActive** is TRUE
    - Check if output **xInOperation** is FALSE
    - Check if output **xSystemError** is FALSE
    - Check if output **xError** is FALSE
    - Check if output **xResultOK** is TRUE or output **xResultNOK** is TRUE

7.1.17 Quit System Error

1. Precondition:
    - Input **xEnable** is TRUE
    - Output **xActive** is TRUE
    - Output **xSystemError** is TRUE
    - Output **xInOperation** is FALSE
    - Output **xError** is FALSE
2. Acknowledge the error via the input **xQuitSystemError**

7.1.18 Function Block Error Reaction

1. Precondition:
    - Input **xEnable** is TRUE
    - Output **xActive** is FALSE
    - Output **xError** is TRUE
    - Output **uiErrorID** contains detailed error information
        - For further information see error table
2. Set input **xEnable** to FALSE
3. Fix the error

7.2 Procedure of YJKP Parameter Read / Write through FB_ParameterChannel

7.2.1 Write Variable (Single)

1. Set input **uiAccessType** to 0x01 (Write)
2. Set input **uiAccessOption** to 0x00 (Single)
3. Set input **uiObjectIndex** 0x2300 (Variables 1-10)
4. Set input **uiObjectSubIndex** 0x01 (Variable 1)
5. Set input **uiObjectDataType** 0x06 (DINT)
6. Set input **uiDataLengthToBeWritten** 0x04 (4 Bytes)
7. Set data to be written to input **auiDataToBeWritten** (Data to be written starts at input **auiDataToBe-Written[0]** with a length of 4 Bytes)
8. Set input **xExecute** to True
9. Condition of write variable done
   - Check if output **xDone** is TRUE
   - Check if output **xBusy** is FALSE
   - Check if output **uiAcknowledge** is 0x00
   - Check if output **xError** is FALSE

7.2.2 Read Variable (Single)

1. Set input **uiAccessType** to 0x00 (Read)
2. Set input **uiAccessOption** to 0x00 (Single)
3. Set input **uiObjectIndex** 0x2300 (Variables 1-10)
4. Set input **uiObjectSubIndex** 0x01 (Variable 1)
5. Set input **uiObjectDataType** 0x06 (DINT)
6. Set input **xExecute** to True
7. Condition of read variable done
   - Check if output **xDone** is TRUE
   - Check if output **xBusy** is FALSE
   - Check if output **uiAcknowledge** is 0x00
   - Check if output **xError** is FALSE
8. Check if output **uiReadObjectDataType** is 0x06 (DINT)
9. Check if output **auiReadDataLength** is 0x04 (4 Bytes)
10. Read data is in output **auiReadData** (Read data starts at **auiReadData[0]** with a length of 4 Bytes)

7.2.3 Write Variable (Multi)

1. Set input **uiAccessType** to 0x01 (Write)
2. Set input **uiAccessOption** to 0x01 (Multi)
3. Set input **uiObjectIndex** 0x2300 (Variables 1-10)
4. Set input **uiObjectSubIndex** 0x00 (If user uses an object sub index unequal to zero the object data type **uiObjectDataType** must be 0x06 (DINT))
5. Set input **uiObjectDataType** 0x07 (UDINT)
6. Set input **uiDataLengthToBeWritten** 0x2C (44 Bytes)
7. Set data to be written to input **auiDataToBeWritten**
   - The first four bytes of the array are the data length and are not used
   - Variable 1 starts at **auiDataToBeWritten[2]** with a length of 4 Bytes
   - Variable 10 starts at **auiDataToBeWritten[20]** with a length of 4 bytes

8. Set input **xExecute** to True
9. Condition of write variable done
   - Check if output **xDone** is TRUE
   - Check if output **xBusy** is FALSE
   - Check if output **uiAcknowledge** is 0x00
   - Check if output **xError** is FALSE

7.2.4 Read Variable (Multi)

1. Set input **uiAccessType** to 0x00 (Read)
2. Set input **uiAccessOption** to 0x01 (Multi)
3. Set input **uiObjectIndex** 0x2300 (Variables 1-10)
4. Set input **uiObjectSubIndex** 0x00 (If user uses an object sub index unequal to zero the object data type **uiObjectDataType** and **uiReadObjectDataType** must be 0x06 (DINT))
5. Set input **uiObjectDataType** 0x07 (UDINT)
6. Set input **xExecute** to True
7. Condition of read variable done
   - Check if output **xDone** is TRUE
   - Check if output **xBusy** is FALSE
   - Check if output **uiAcknowledge** is 0x00
   - Check if output **xError** is FALSE
   - Check if output **uiReadObjectDataType** is 0x07 (UDINT)
   - Check if output **uiReadDataLength** is 0x2C (44 Bytes)
   - Read data is in output **auiReadData**
     - The first four bytes of the array are the data length
8. Variable 1 starts at **auiReadData [2]** with a length of 4 Bytes
9. Variable 10 starts at **auiReadData [20]** with a length of 4 bytes

7.2.5 Function Block Error Reaction

1. Precondition:
   - Input **xExecute** is TRUE
   - Output **xBusy** is FALSE
   - Output **xError** is TRUE
   - Output **uiErrorID** contains detailed error information
     - For further information see error table. Refer Chapter 4.8.4.
2. Set input **xExecute** to FALSE
3. Fix the error

Note

- The **FB_ParameterChannel** block provides the capability to read and write the parameters starting from index **0x2010** provided in the application note **YJKP - Object directory**.