Mech-Mind Robot Integrations

Mech-Mind

Jul 01, 2022

CONTENTS

This chapter introduces different ways of integrating your robot with Mech-Mind Software Suite.

The following section provides instructions for loading the full-control program onto different robots:

Full-Control Program

The following section provides information on the Standard Interface of Mech-Interface for different robots and PLCs:

Standard Interface

CHAPTER

ONE

FULL-CONTROL PROGRAM

This chapter provides detailed information on realizing full-control of robots by Mech-Mind.

1.1 ABB

This section introduces the full-control program for ABB robots and the procedure of setting up the communication with a robot through the program.

1.1.1 ABB Setup Instructions

This section introduces the process of loading the robot full-control program onto an ABB robot.

The process consists of 4 steps:

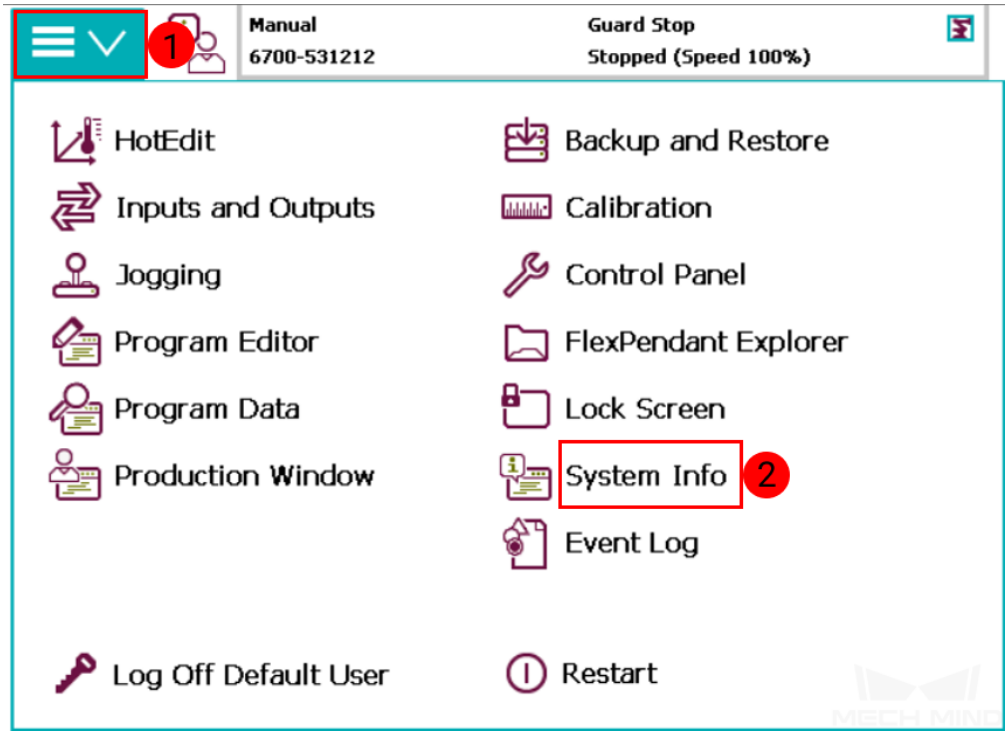
- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program Files*
- *Test Robot Connection*

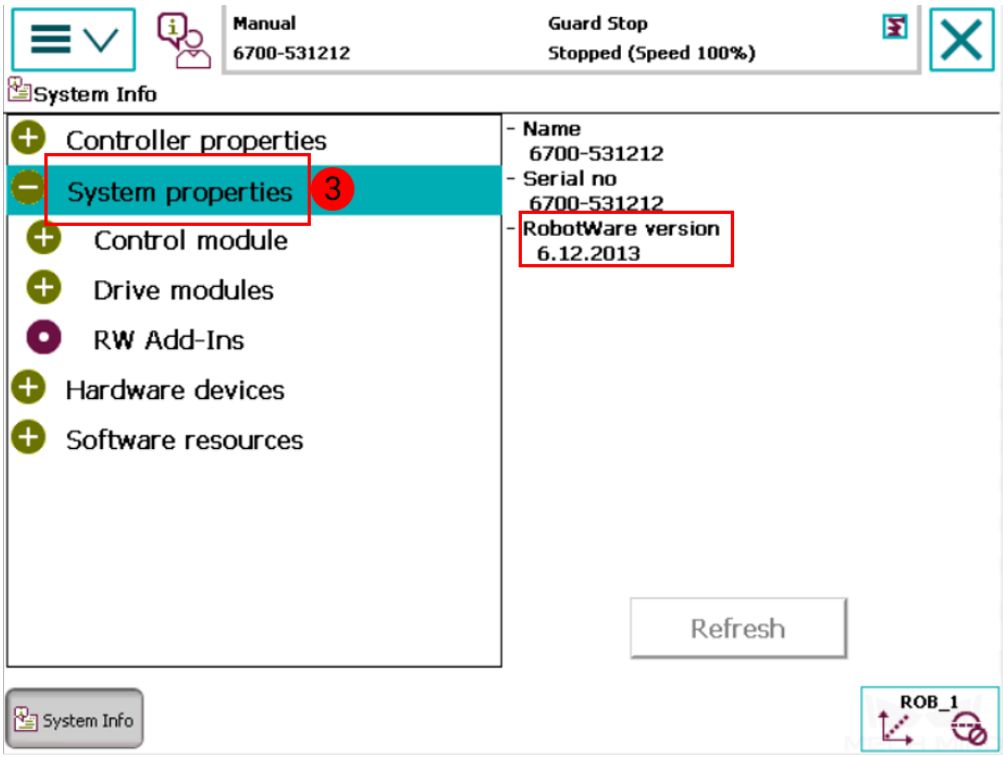
Please have a flash drive ready at hand.

Check Controller and Software Compatibility

- Controller: no requirement
- Controller system software version: RobotWare 6.02 or above
- Additional controller software options: 623-1 Multitasking and 8616-1 PCInterface
- Mech-Center: latest version recommended

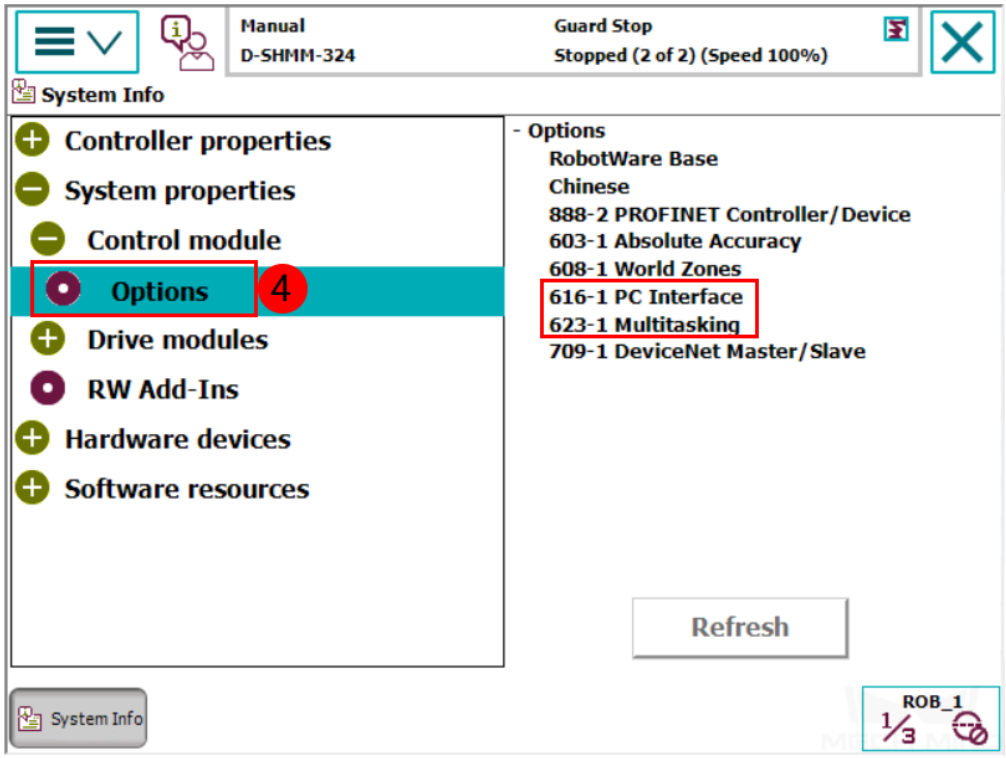
1. Tap  and then go to *System Info*→ *System properties* to check the Robotware version.





2. Go to *System properties*→ *Control Module*→ *Options* to check whether the necessary options are

installed.



Setup the Network Connection

Hardware Connection

Plug the Ethernet cable of the IPC into the X6 (WAN) port of the robot controller, as shown below.

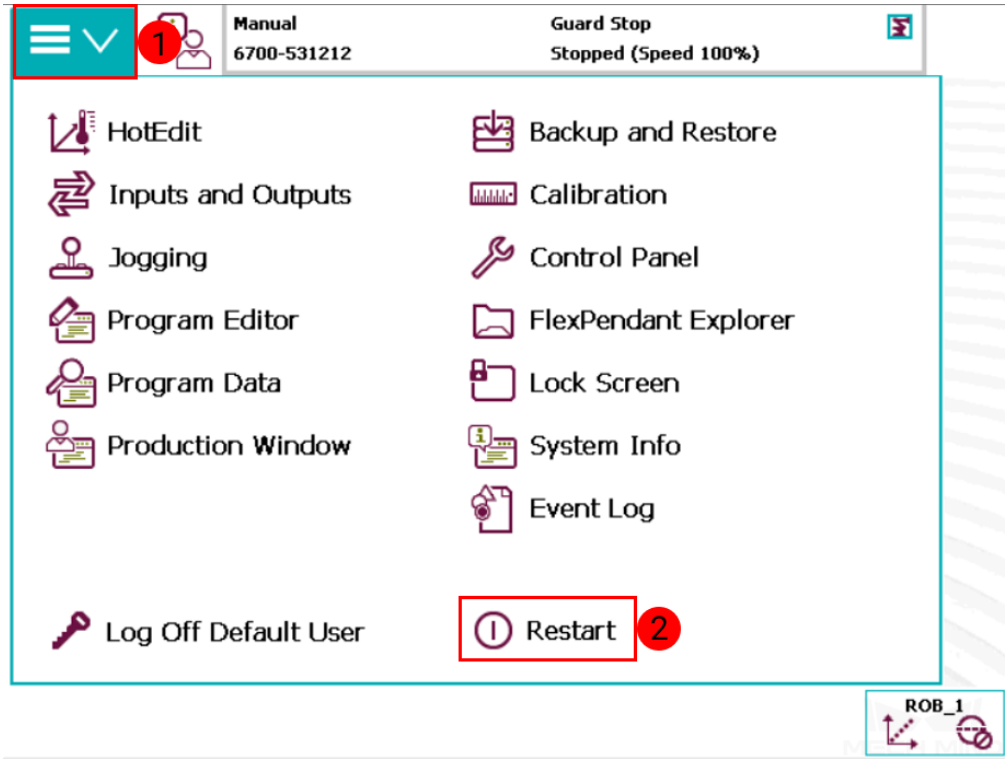
Hint: If you only need to load the full-control program via RobotStudio, you can use either LAN port or WAN port to connect the robot controller. However, in order to enable visual communication, the Ethernet cable can only be connected to the WAN port.

IP Configuration

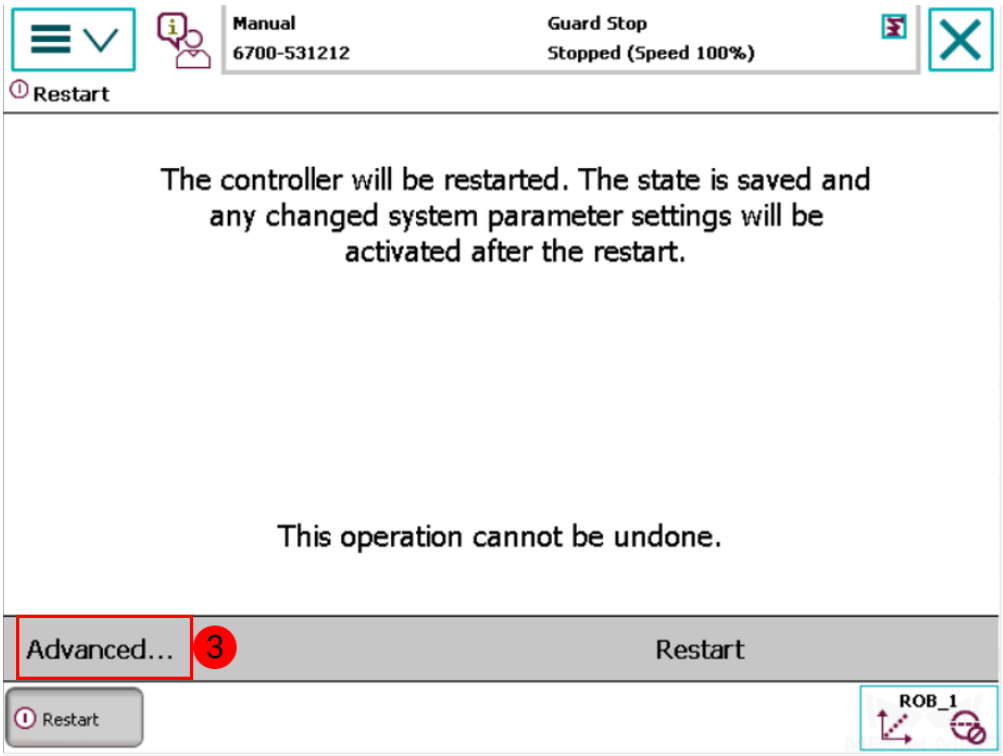
You can set the IP on the teach pendant or via RobotStudio.

- Set the IP on the teach pendant

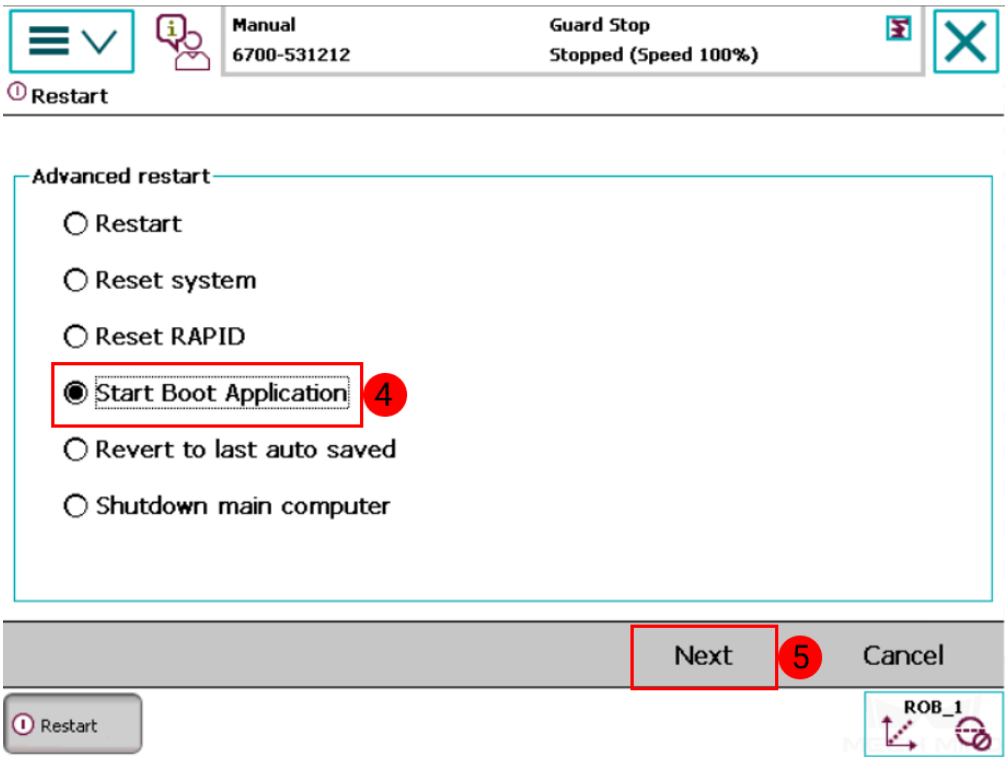
  1. Tap  and select *Restart*.



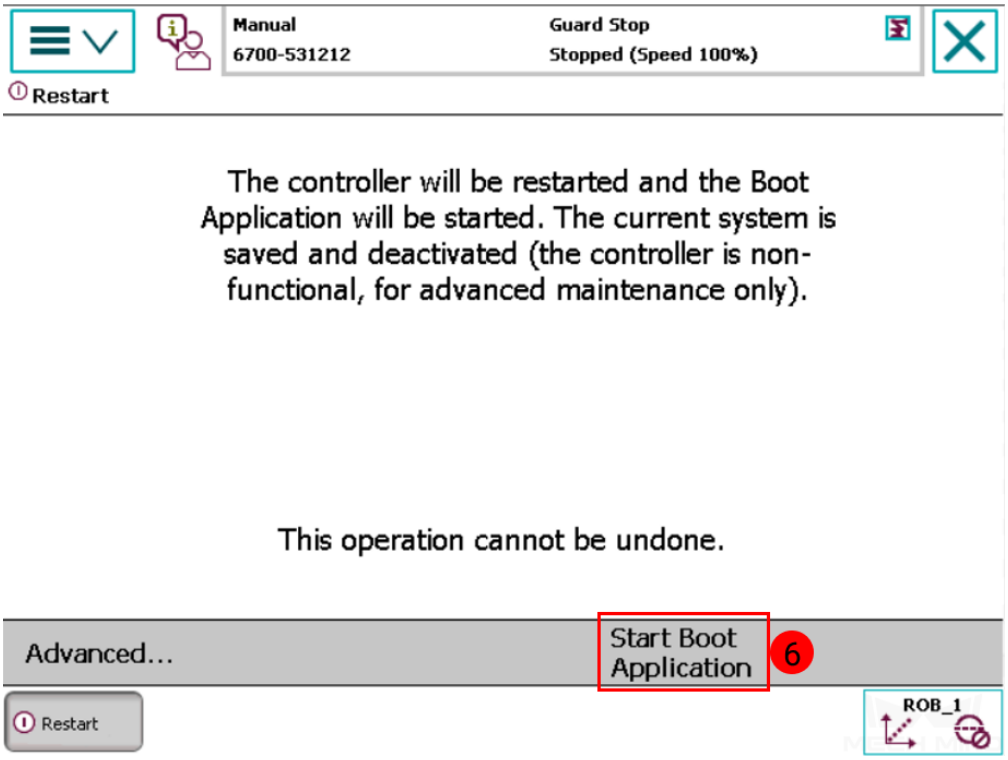
2. Select *Advanced*···



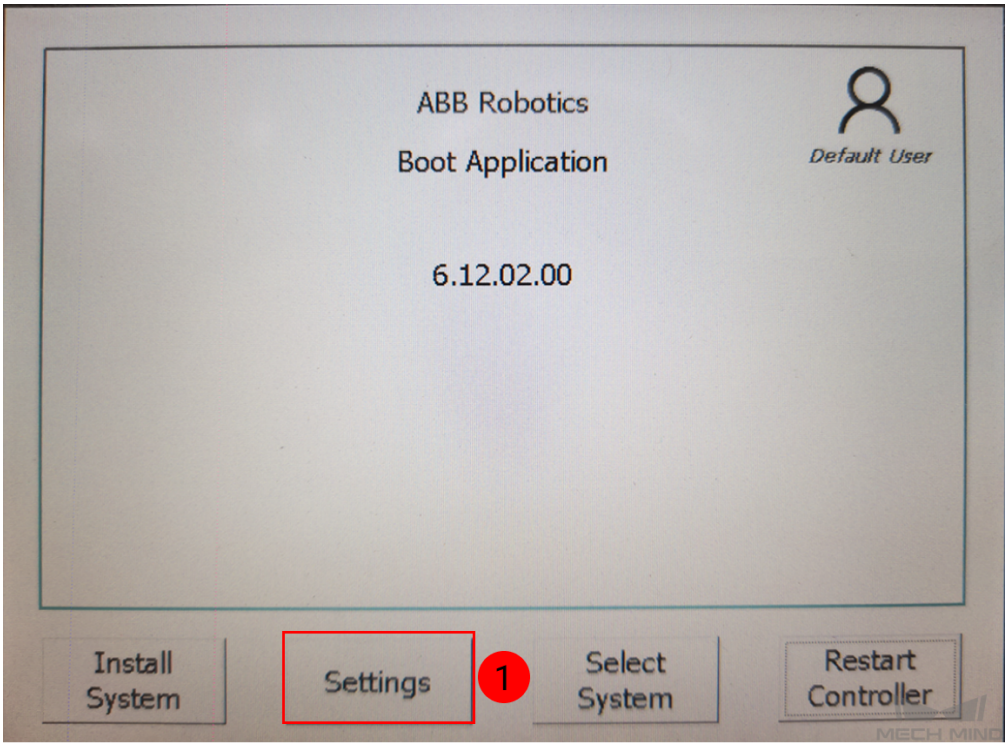
3. Select *Start Boot Application* and tap *Next*.



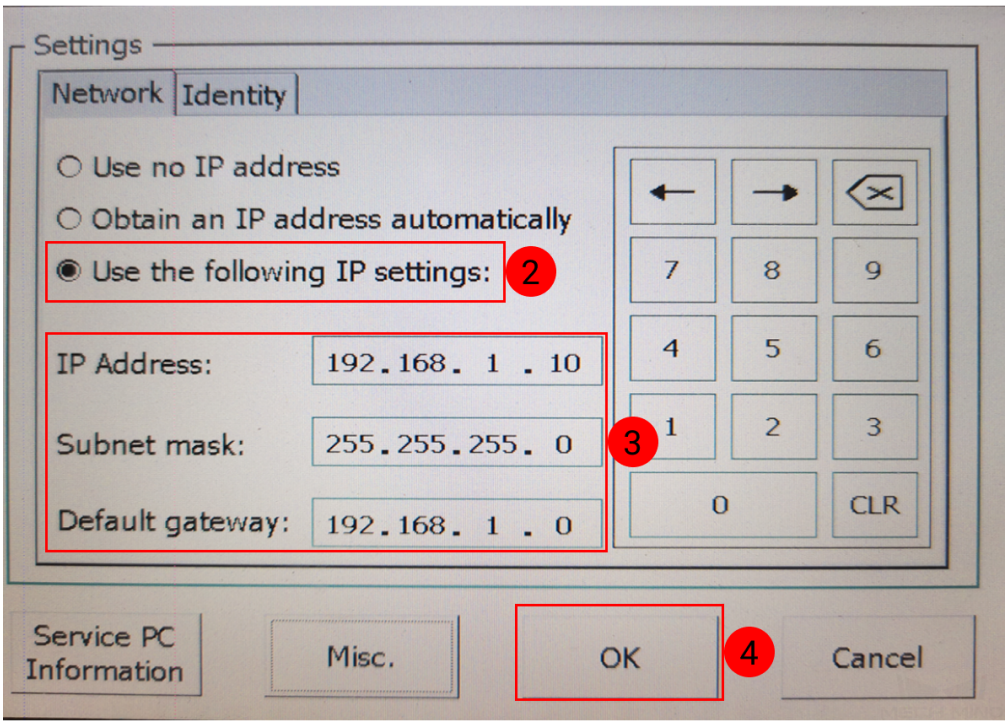
4. Select *Start Boot Application* to confirm.



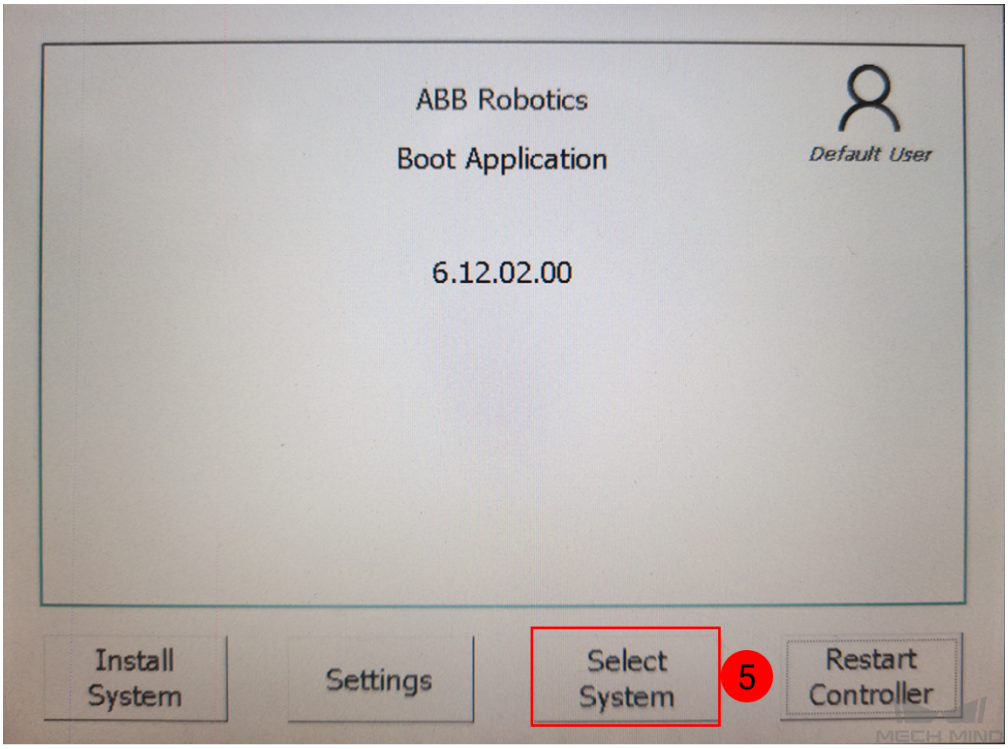
5. After restarting, you will see the interface as shown below. Tap *Settings*.



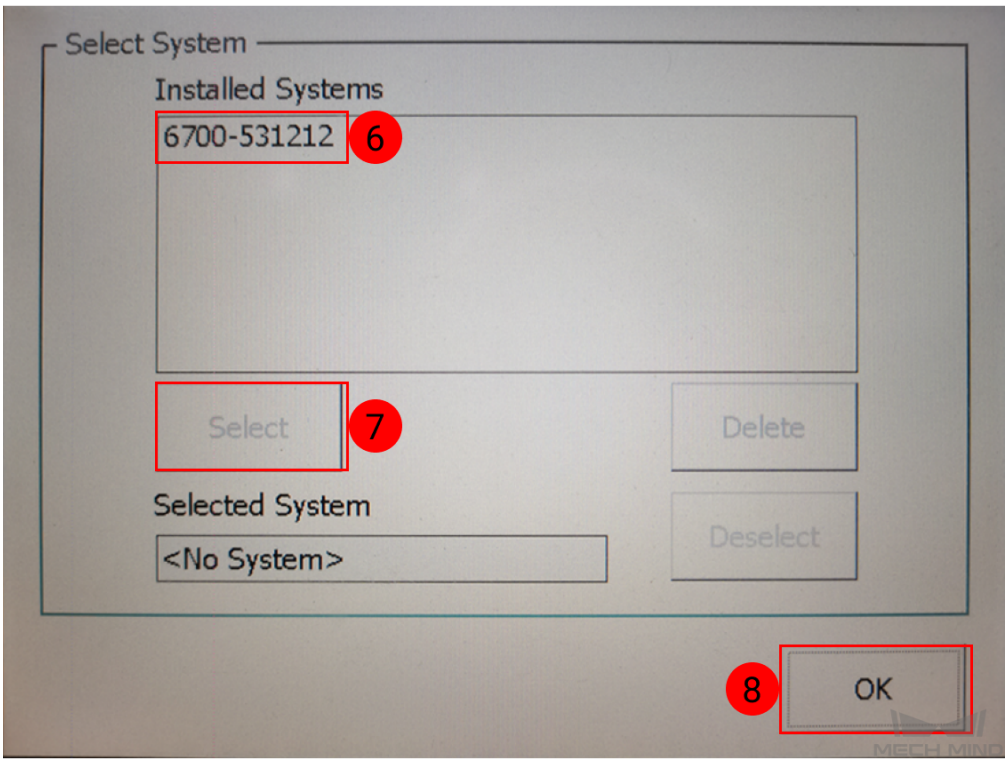
6. Select **Use the following IP settings** and configure the **IP Address**, **Subnet Mask**, and **Default gateway**. Tap *OK* after configuration.



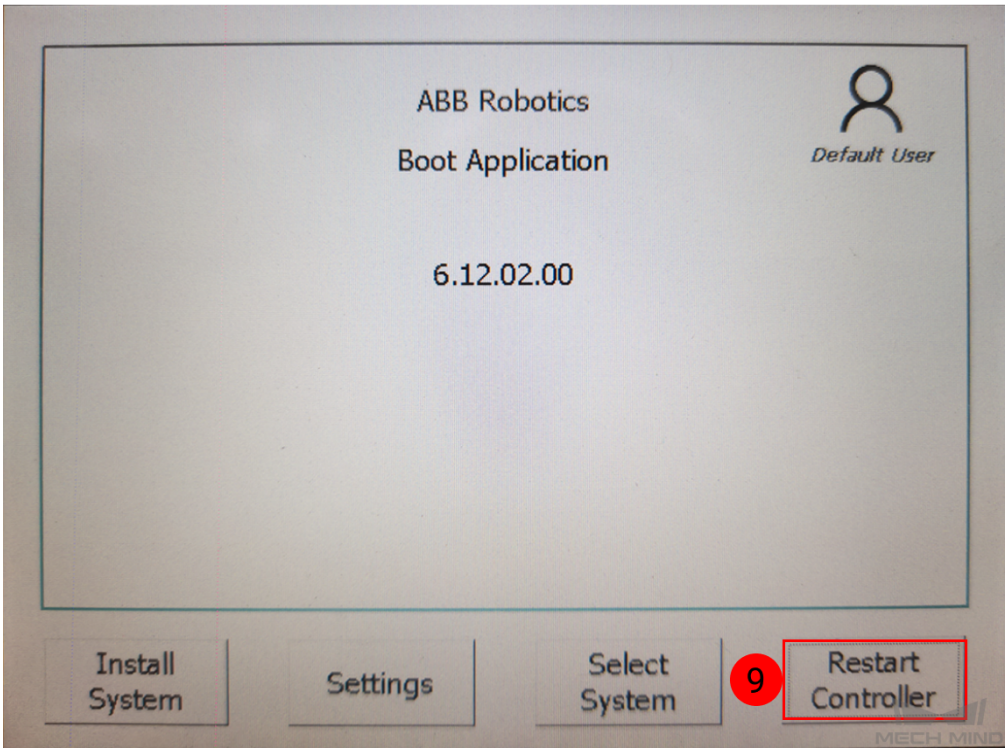
7. Tap *Select System*.



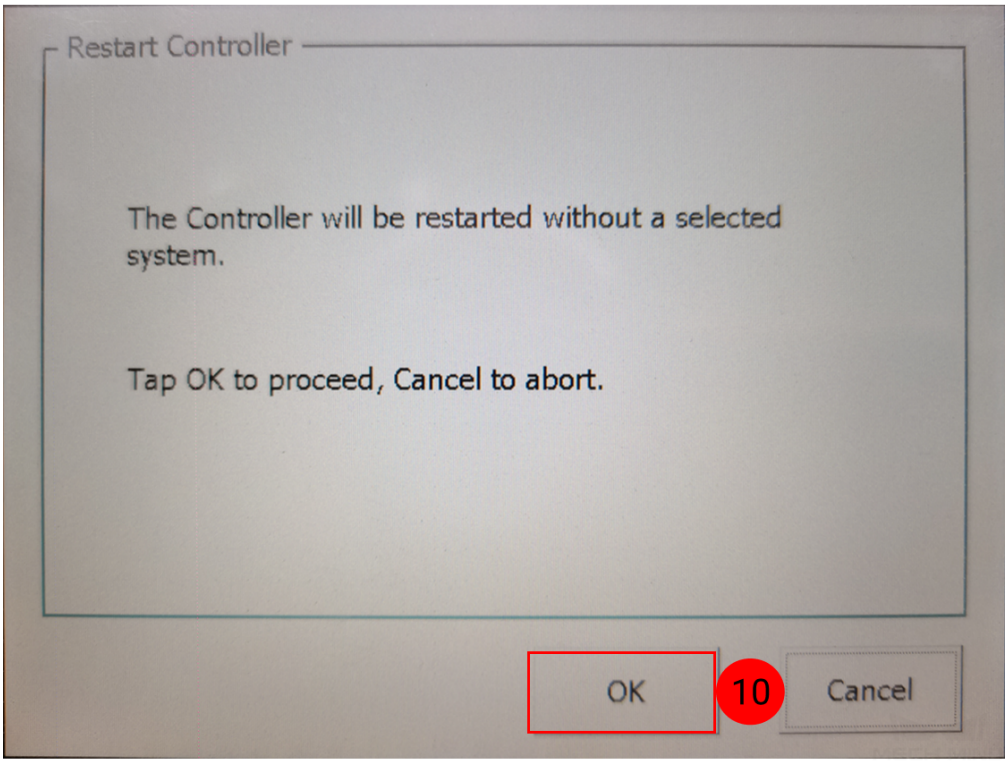
8. Select the system name in **Installed Syetems** box and then tap *Select*. Tap *OK* after configuration.



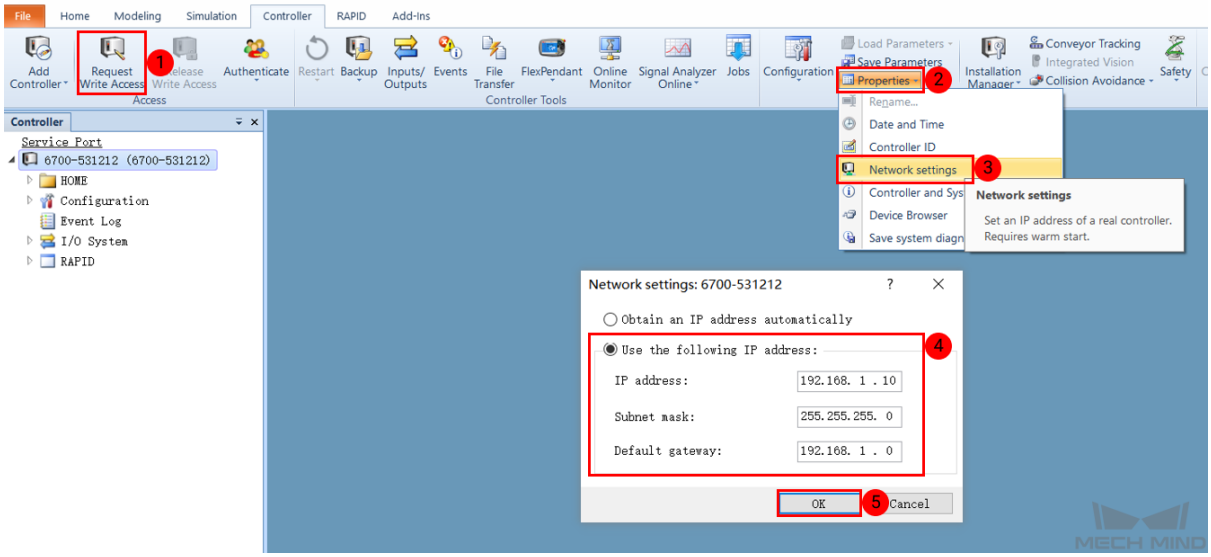
9. Select *Restart Controller*.



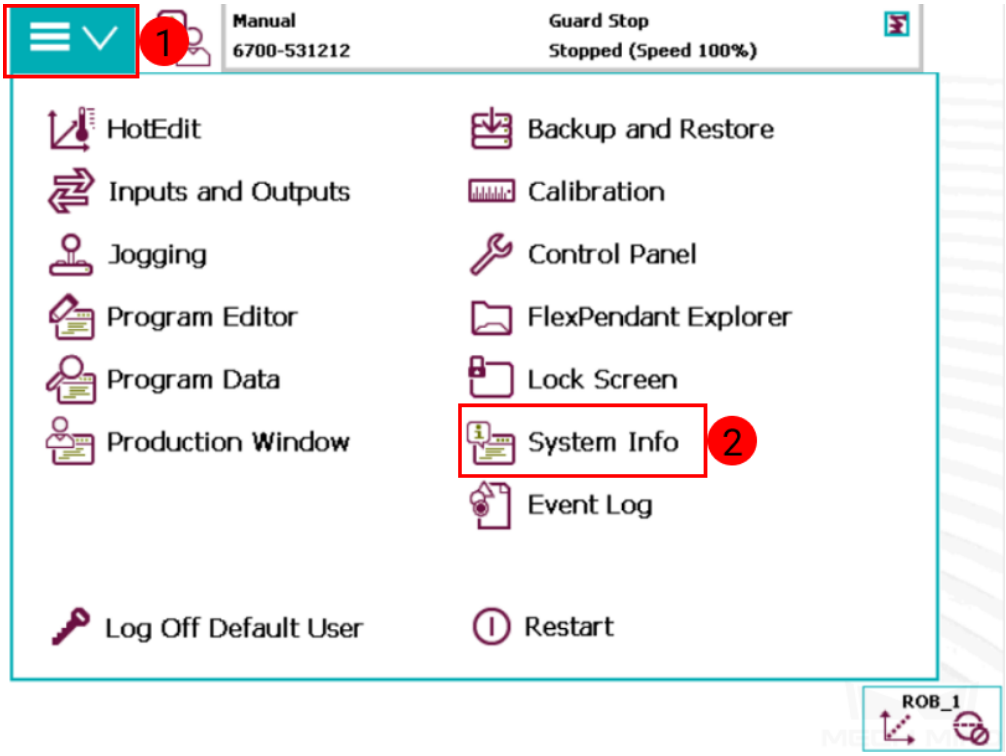
10. Tap *OK* to proceed.

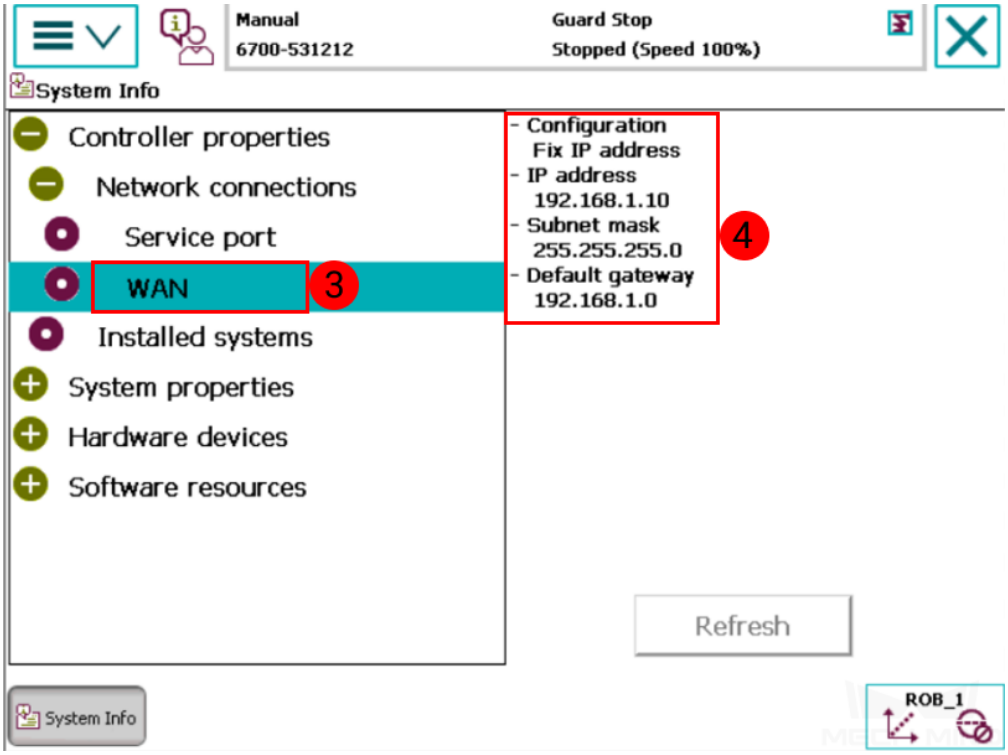


- Set the IP via RobotStudio
  1. Follow the steps as shown below to configure the robot IP, and restart the robot after configuration.



- Go to *Sytem Info*→ *Network connections*→ *WAN* to check if the IP configuration was successful after restarting.

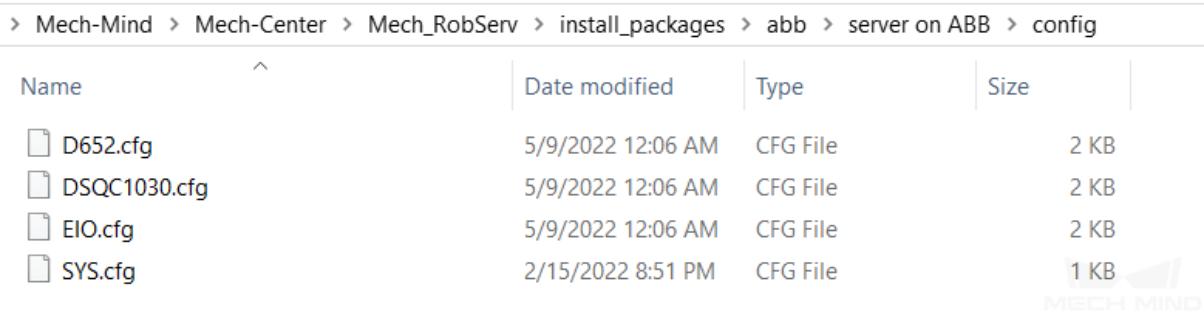




Load the Program Files

Prepare the Files

1. Copy the program files into an USB flash drive. Please locate the folder where Mech-Center is installed and the files are stored in *XXXX/Mech-Center/Mech_RobServ/install_packages/abb/server on ABB/config*.

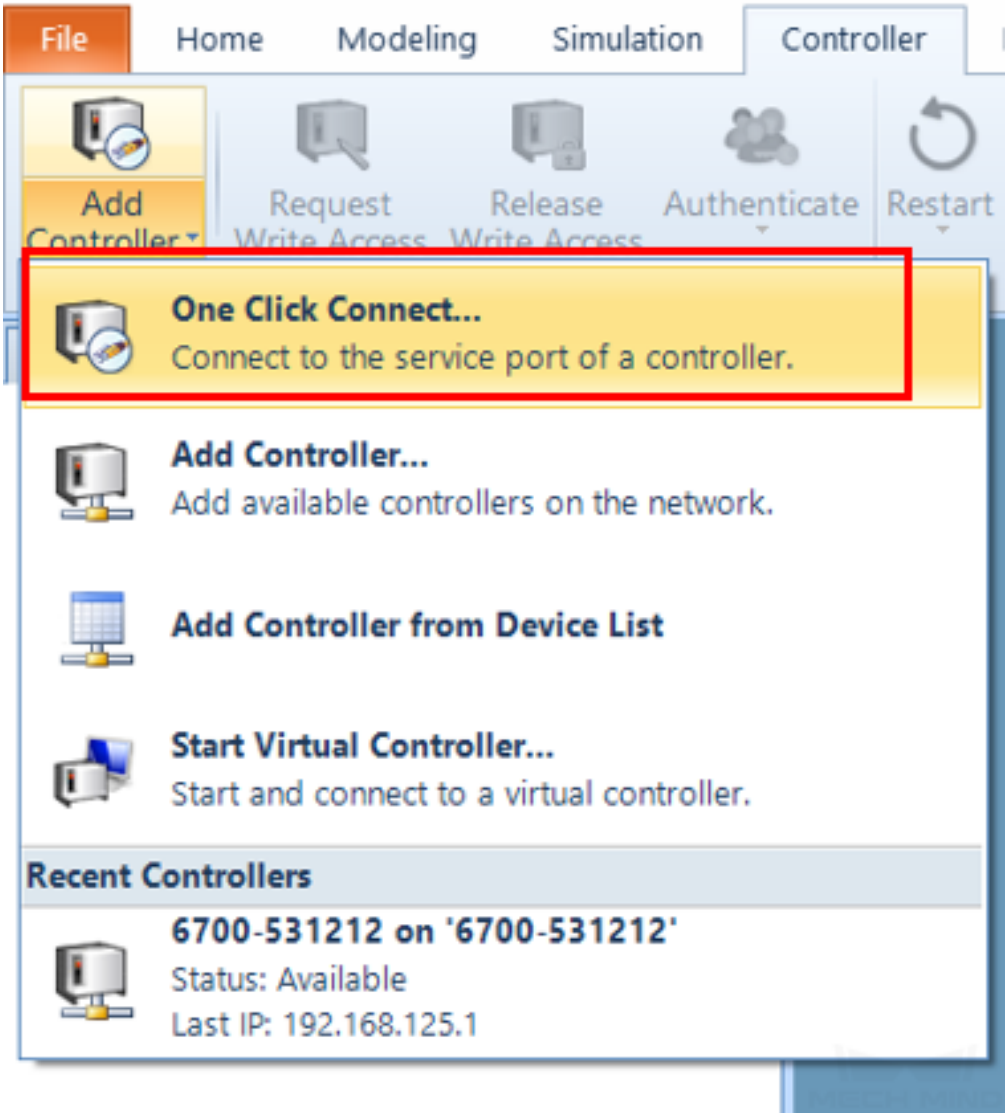


2. The config file should be compatible with the I/O Unit in use. Please choose the right config file according to the table below:

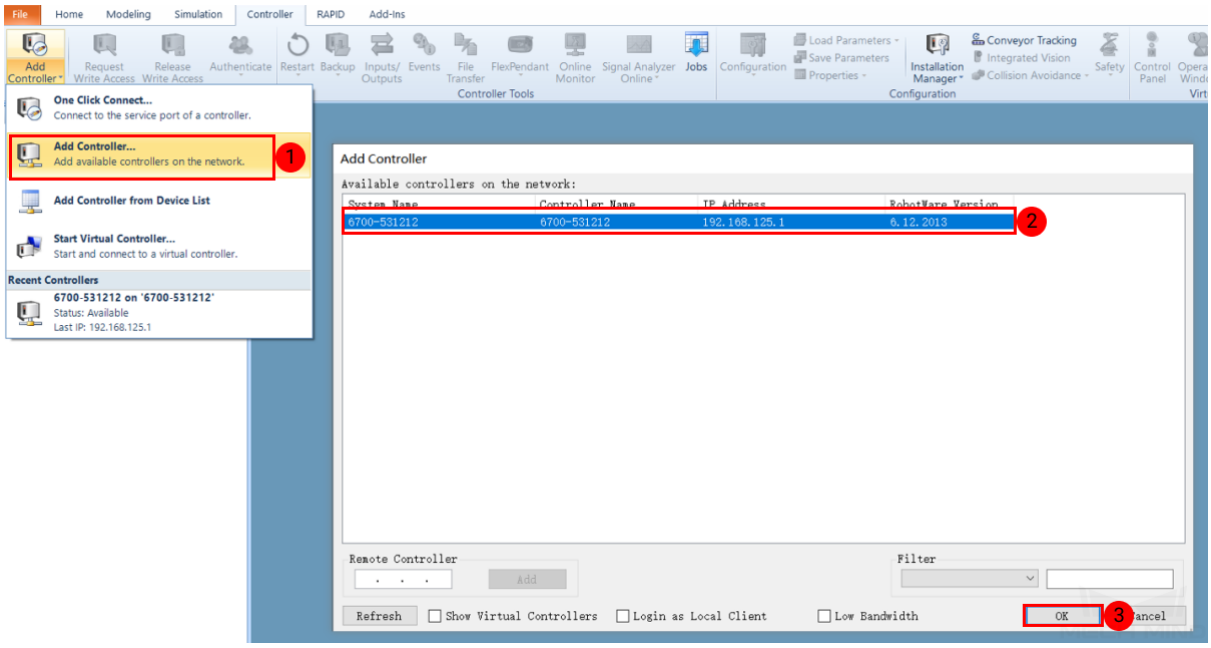
I/O Unit	config file
DSQC 652	D652.cfg, SYS.cfg
DSQC 1030	DSQC1030.cfg, SYS.cfg
Other I/O Units or the program is only used for auto-calibration	EIO.cfg, SYS.cfg

Load the Files to the Robot

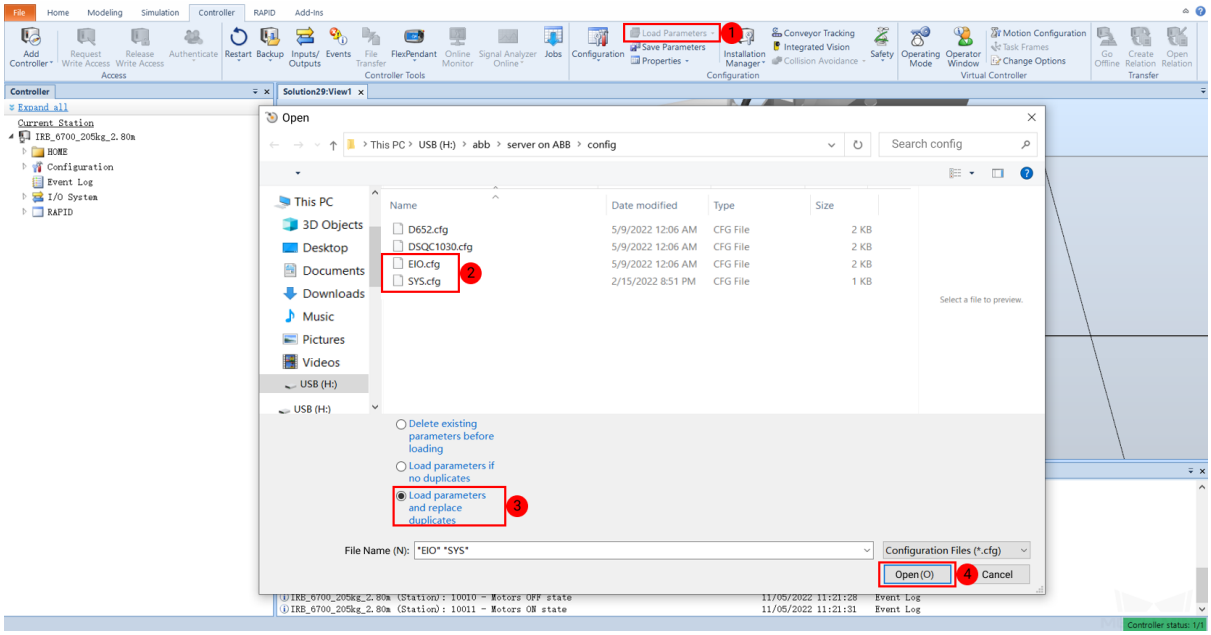
1. Open RobotStudio on the IPC and connect to the controller.
    - If the robot controller is connected via the LAN port, click on **One Click Connect ...**.

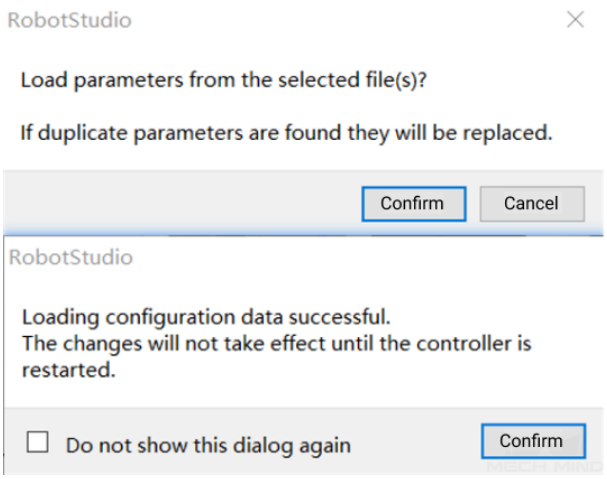


- If the robot controller is connected via the WAN port or a switch, click on *Add Controller* and then select the controller and click on *OK*.

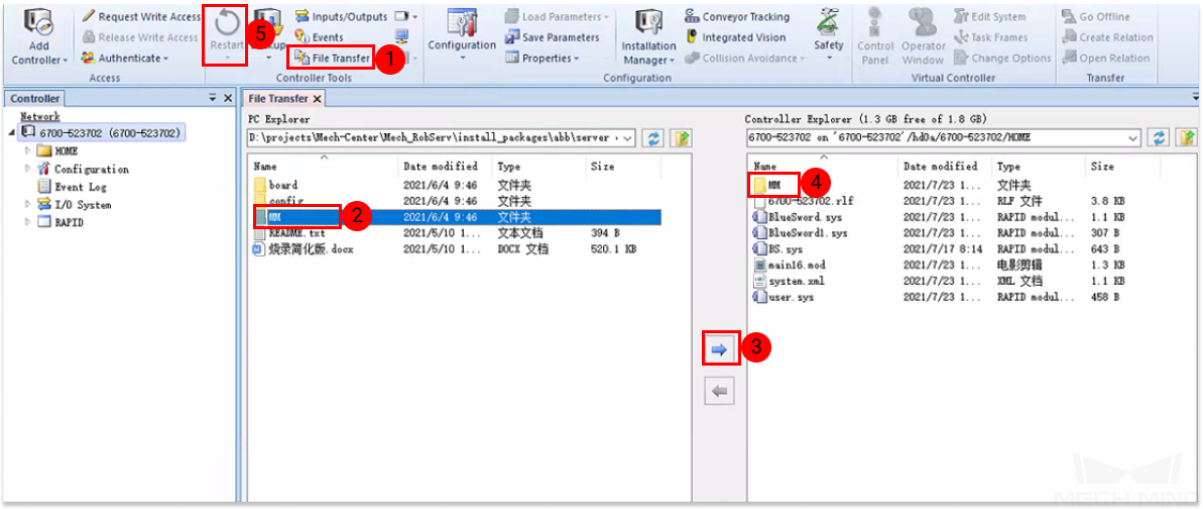


2. Import the config files as shown below. Click on *Confirm* in the pop-up windows.





3. Copy the whole **MM** folder and paste it to the **HOME** directory of the robot system, as shown below. Restart the controller to complete loading the program files.



4. Modify the safe zone threshold (in mm) in the **safe_area.mod** program according to the actual on-site work space of the robot.

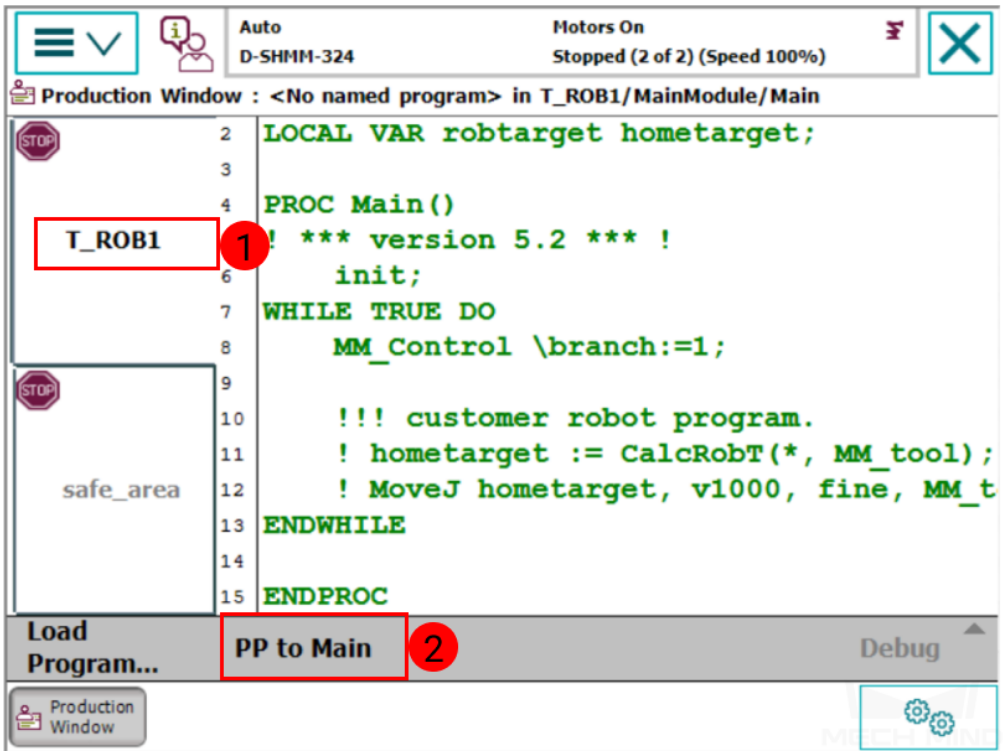
Solution29:视图1 | IRB_6700_205kg_2.80m (工作站) ×

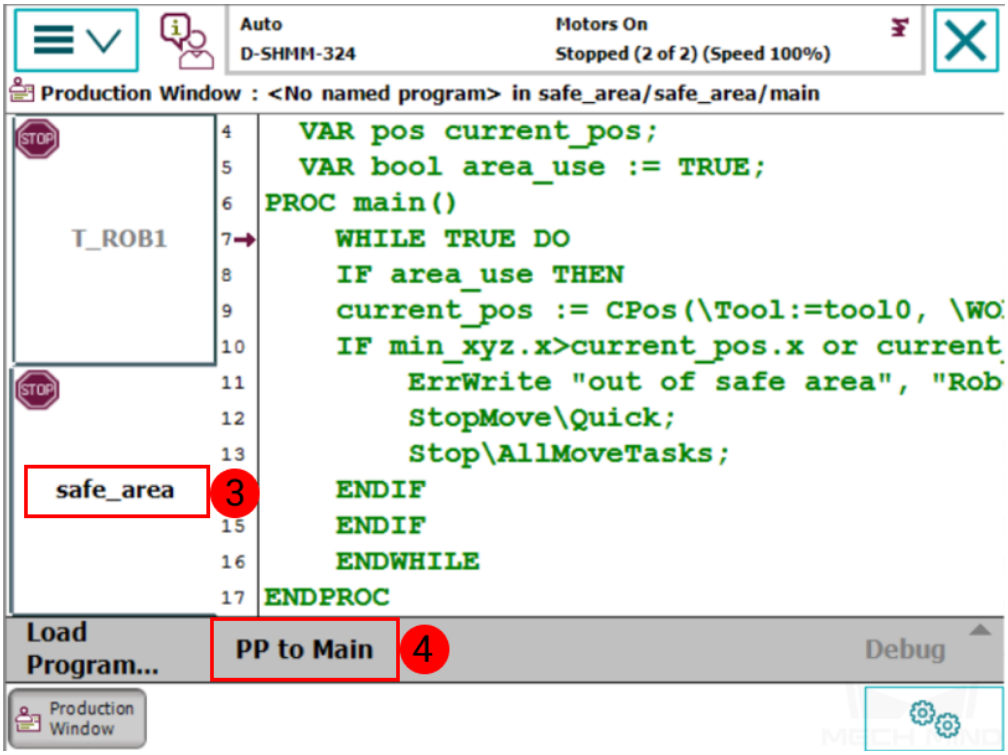
safe_area/safe_area ×

```
MODULE safe_area
  CONST pos min_xyz:=[-3000,-3000,-3000];
  CONST pos max_xyz:=[3000,3000,3000];
  VAR pos current_pos;
  VAR bool area_use := TRUE;
PROC main()
    WHILE TRUE DO
    IF area_use THEN
    current_pos := CPos(\Tool:=tool0, \WObj:=wobj0)
    IF min_xyz.x>current_pos.x or current_pos.x>max
        ErrWrite "out of safe area", "Robot Stop" \
        StopMove\Quick;
        Stop\AllMoveTasks;
    ENDIF
    ENDIF
    ENDWHILE
ENDPROC
ENDMODULE
```

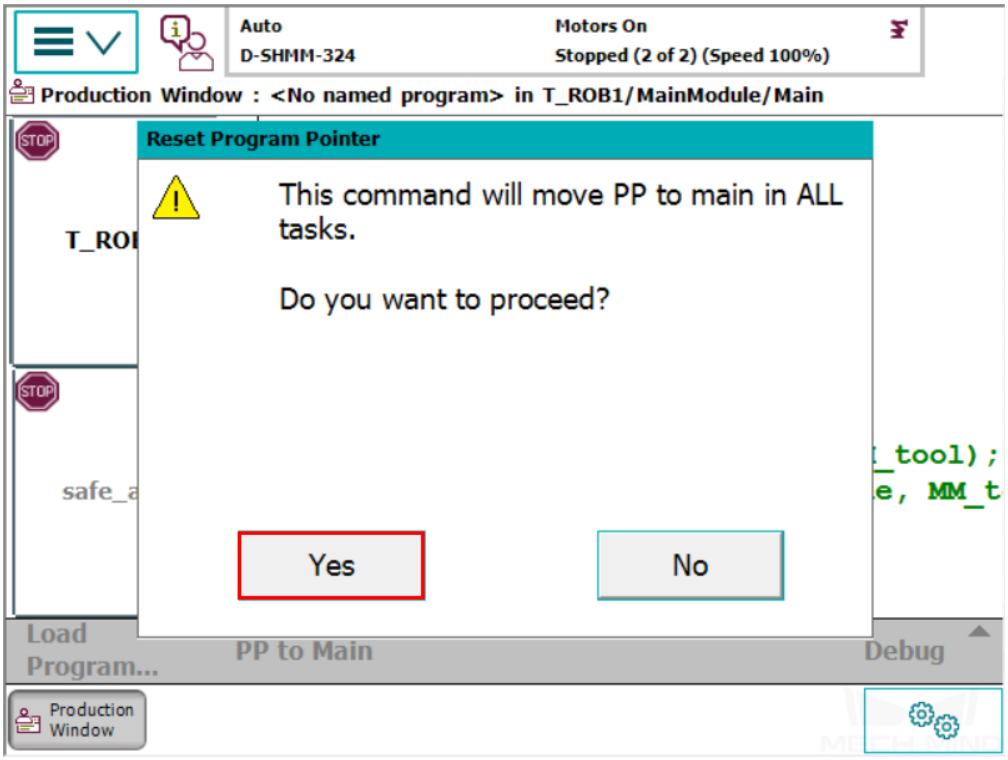
Run the program

1. Move the PP of tasks **T_ROB1** and **safe_area** to Main respectively.

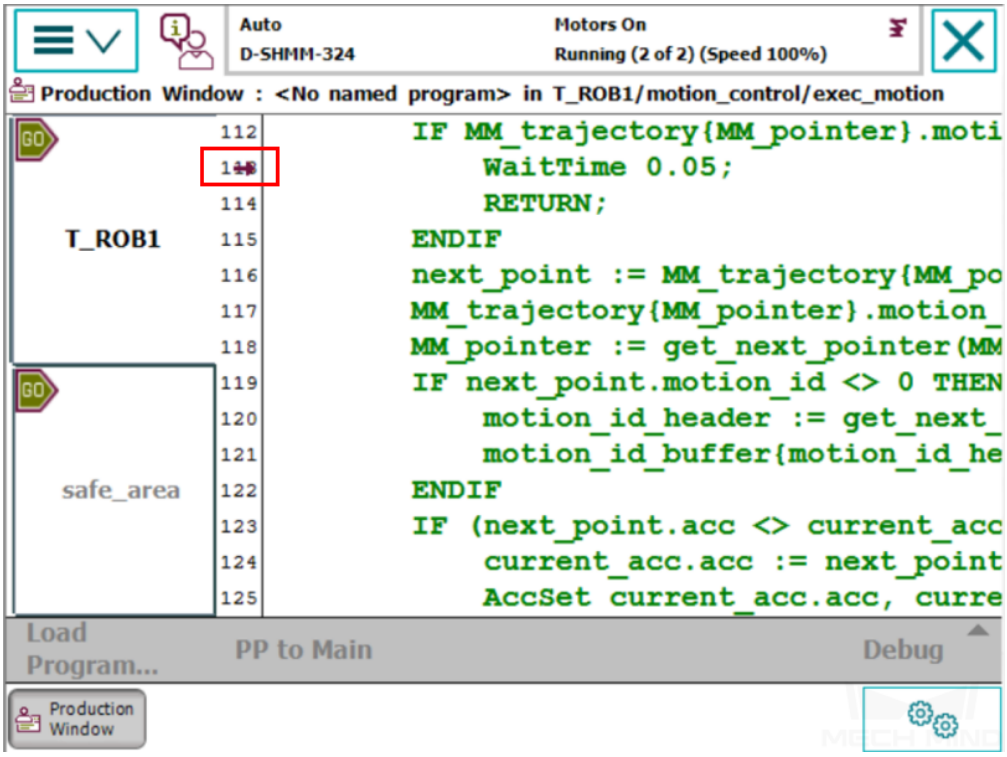




2. After selecting *PP to Main*, if a window as shown below pops up, please tap *Yes* to confirm.



3. Run the program manually or automatically. The program pointer is as shown below.



Test Robot Connection

Please refer to *Test Robot Connection* for detailed instructions.

1.1.2 ABB Program Description

Program Module

Program Module	Description
motion_server	Background program used to receive data from Mech-Center
status_server	Background program used to send data of robot pose, signal, and status
motion_control	Foreground program used to guide the robot to move
pause_control	Program used to pause
MainModule	Main program
mm	Program data used to define the full-control program

Signal

Support 64 D/I and D/O signals in maximum. Support 16 D/I and D/O signals when loading the program files by default.

D/O	go16	go16_2	go16_3	go16_4
D/I	gi16	gi16_2	gi16_3	gi16_4

1.2 YASKAWA

This section introduces the full-control program for YASKAWA robots and the procedure of setting up the communication with a robot through the program.

1.2.1 YASKAWA Setup Instructions

This section introduces the process of loading the robot full-control program onto a YASKAWA robot.

The process consists of 4 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program File*
- *Test Robot Connection*

Please have a flash drive ready at hand.

Note: The flash drive must:

- Have a storage capacity smaller than 32 GB
- Be formatted to the FAT32 file system

Check Controller and Software Compatibility

- Controller: YRC1000 (excluding YRC1000 micro) and DX200
- Controller system software version:
  - YRC1000: no requirement
  - DX200: DN2.25.00A(US/CN)-00 or above
- Option function requirements: must have the MotoPlus and Ethernet functions enabled.

Note: The following instructions are based on YRC1000 controller. Details may differ for DX200 controller.

Setup the Network Connection

Hardware Connection

Plug the Ethernet cable into:

- An Ethernet port on the IPC
- LAN2 (CN106) port on YRC1000 controller; CN104 port on DX200 controller

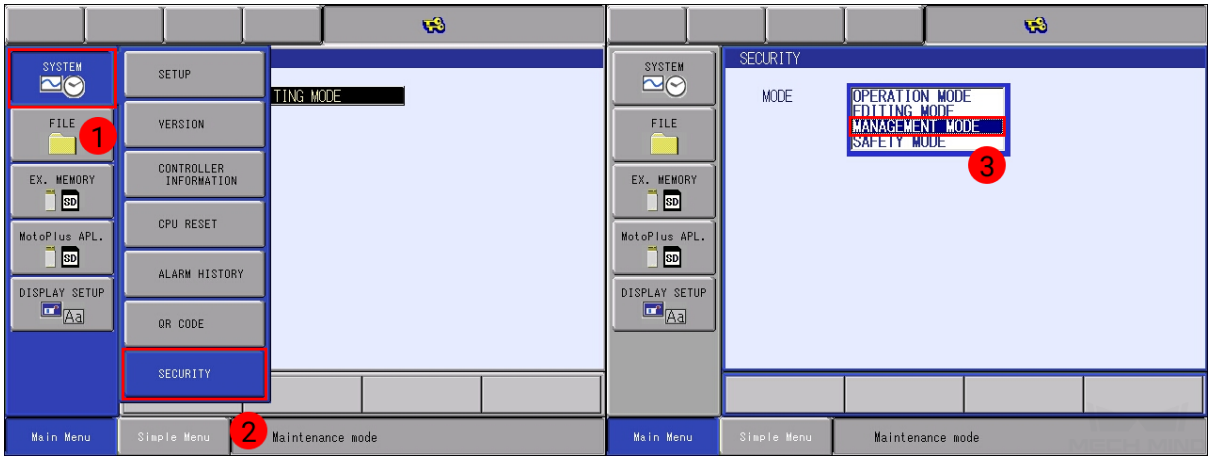
Note:

- LAN1 port on YRC1000 and CN105 port on DX200 are for connecting the teach pendant only.
- If LAN2 port is occupied, please use LAN3 (CN107) instead.

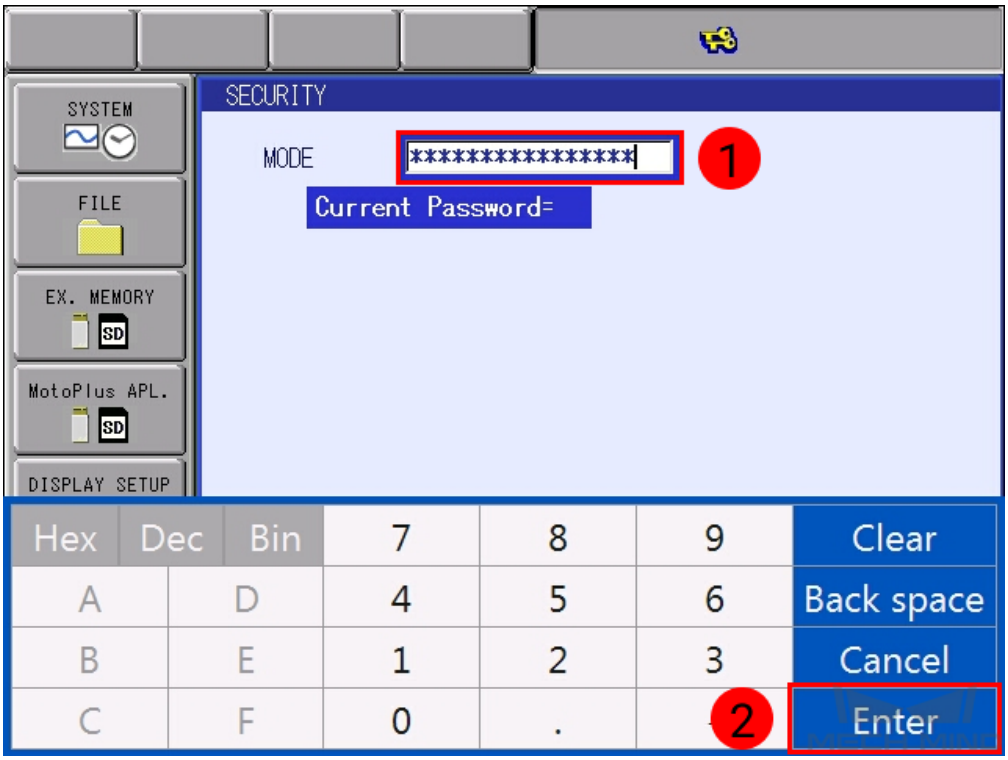
IP Configuration

To allow communication between the IPC and the robot controller, both must have an IP address in the same subnet. This means that the first three numbers of the IP addresses should be the same. For example, 192.168.100.1 and 192.168.100.2 are in the same subnet.

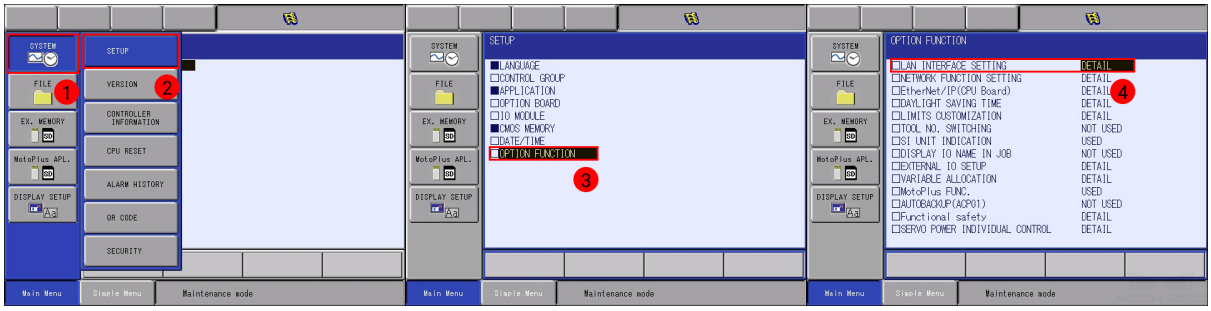
1. Press down `MAIN MENU` when powering on the controller to enter the maintenance mode.
2. Select *SYSTEM* → *SECURITY* → *MANAGEMENT MODE*.



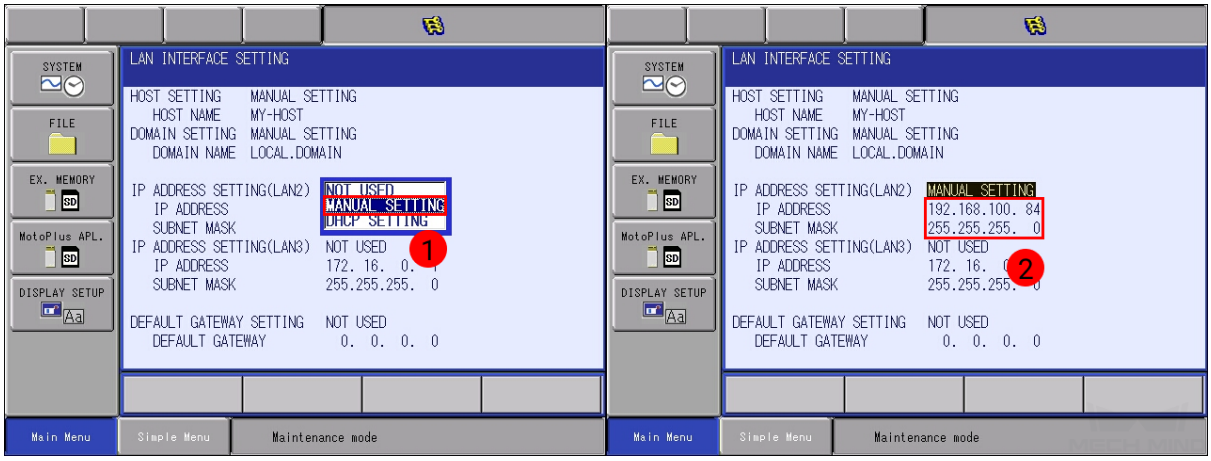
3. Enter the password (the default password is sixteen **9** ‘s), and then press on *Enter*.



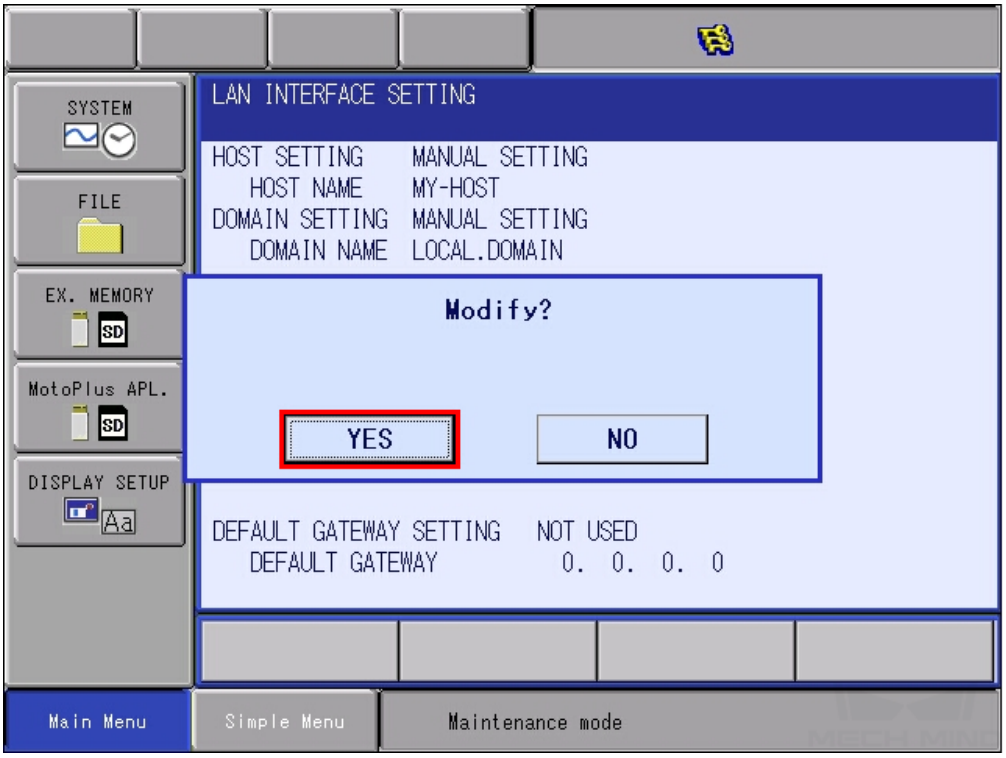
4. Select *SYSTEM* → *SETUP* → *OPTION FUNCTION* → *LAN INTERFACE SETTING*.



5. In **IP ADDRESS SETTING(LAN2)**, select **MANUAL SETTING**, and then set the **IP ADDRESS** to one in the same subnet as the IPC, and the **SUBNET MASK** to **255.255.255.0**.



6. Press the `ENTER` key, and then press on *YES* in the pop-up message.



Load the Program File

Prepare the File

The program files are stored in the installation directory of Mech-Center. The default directory is *C:/Mech-Mind/Mech-Center*.

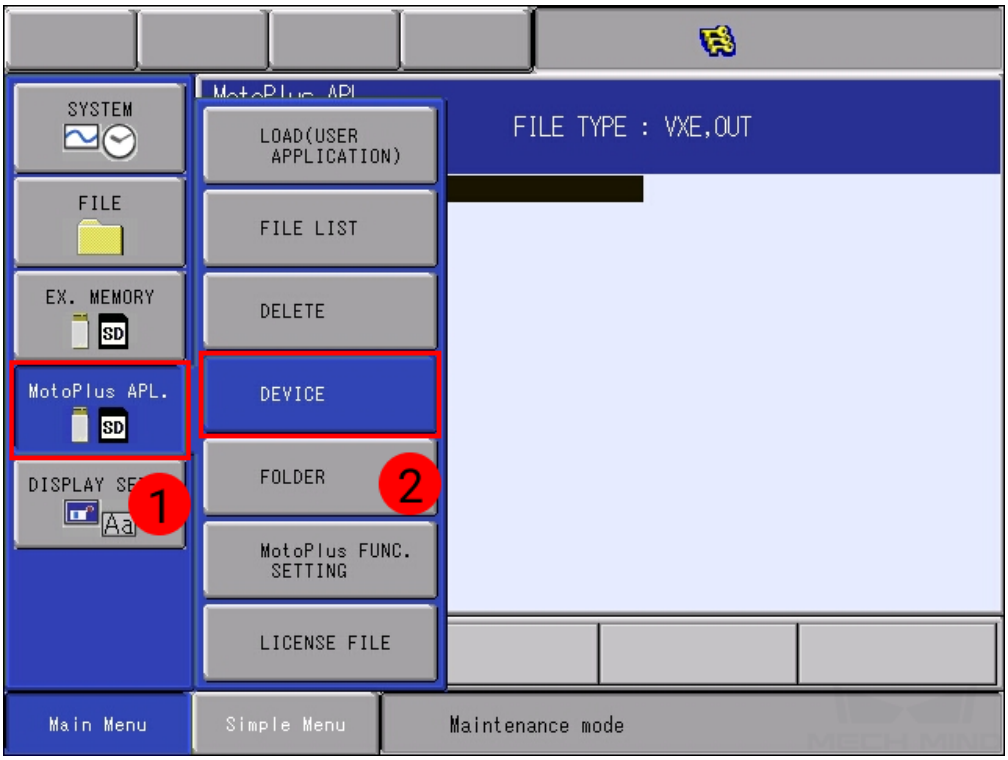
Navigate to *xxx/Mech-Center/mech_interface/yaskawa*, and copy the full-control program to your flash drive:

- If you are using a YRC1000 controller, copy **yrc1000.out**.
- If you are using a DX200 controller, copy **dx200.out**.

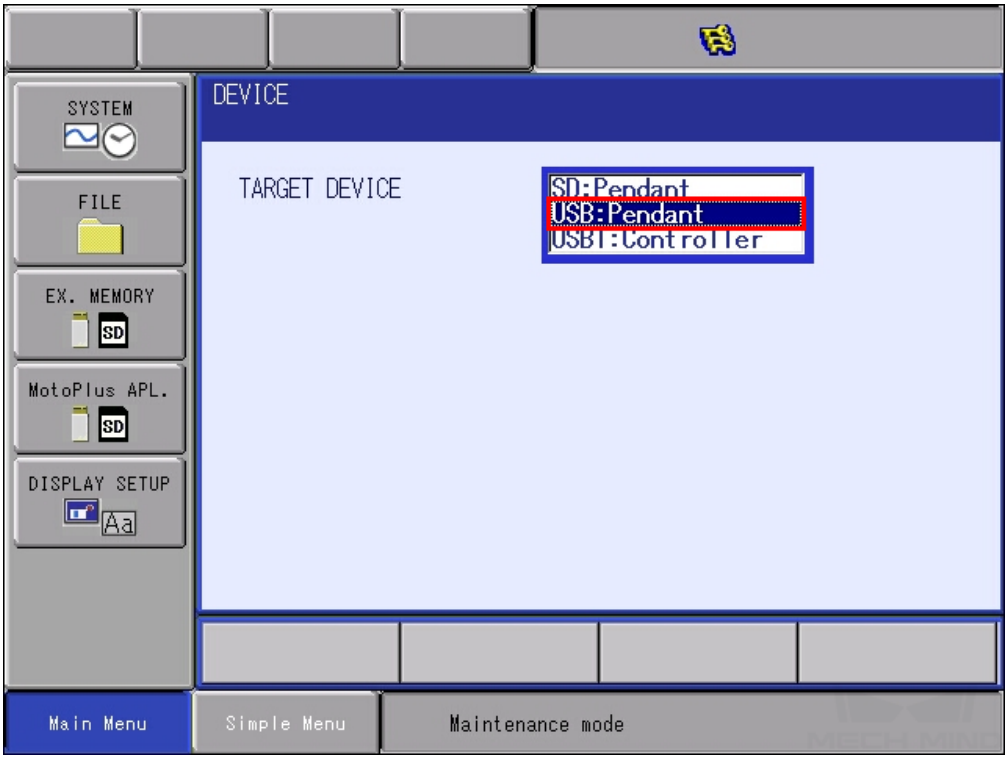
Note: Copy the file to the root directory of the flash drive. Do not put it in another folder or rename it.

Load the File to the Robot

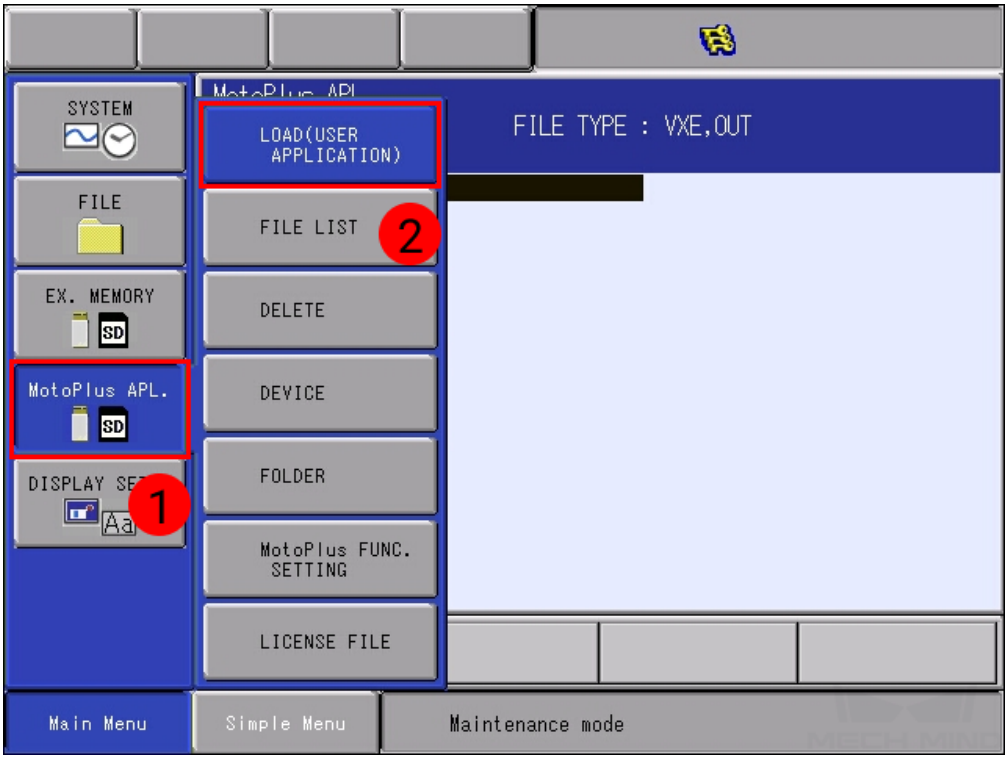
1. Insert the flash drive into the USB port on the back of the teach pendant.
2. Under maintenance mode, select *MotoPlus APL.* → *DEVICE*.



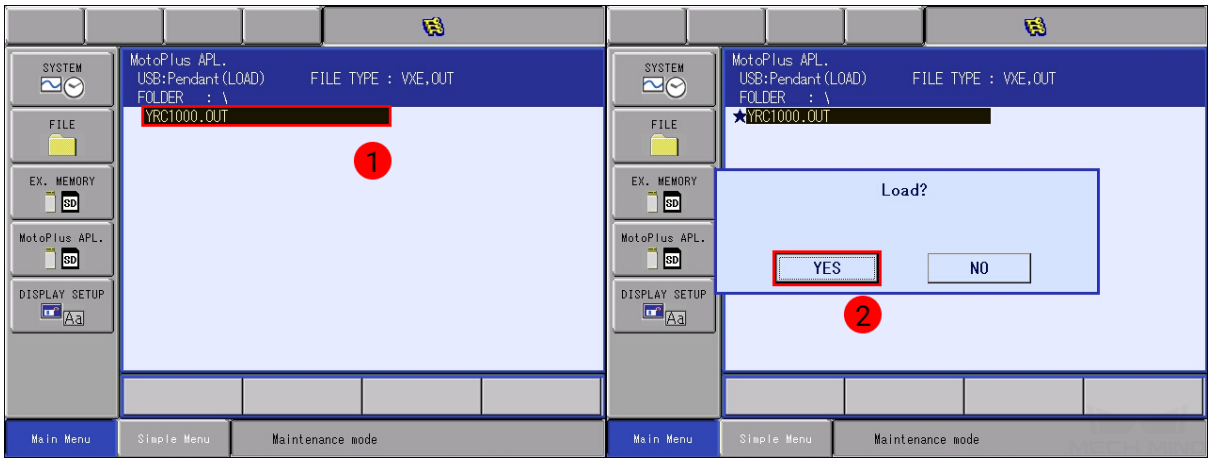
3. Select **USB:Pendant** for **TARGET DEVICE**.



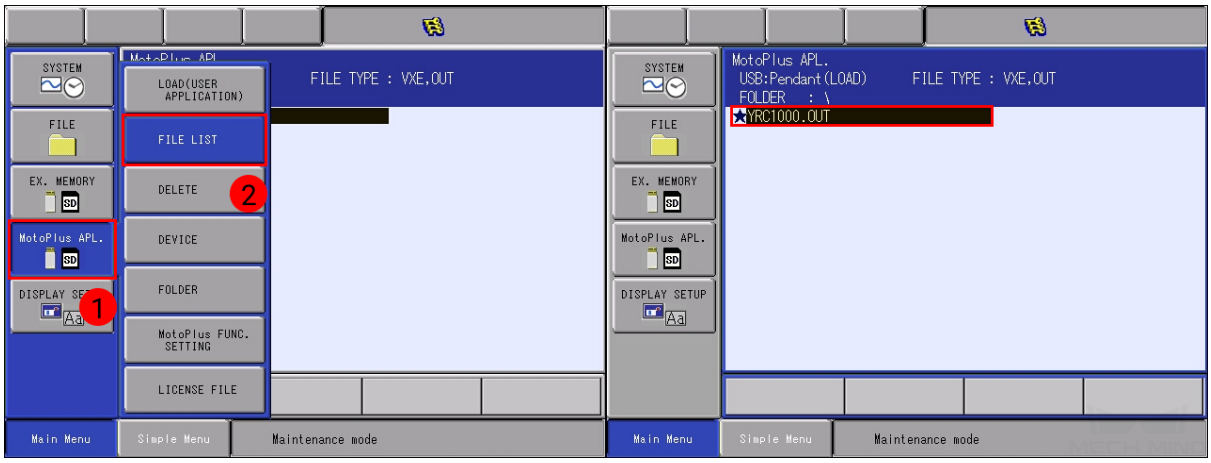
4. Select *MotoPlus APL.* → *LOAD(USER APPLICATION)*.



5. Select **YRC1000.OUT** (**DX200.OUT** for DX200 controller), and press `ENTER`. Select **YES** in the pop-up message to start loading the program.



6. After loading completes, go to *MotoPlus APL.* → *FILE LIST*, and you should see **YRC1000.OUT** (**DX200.OUT**) displayed.



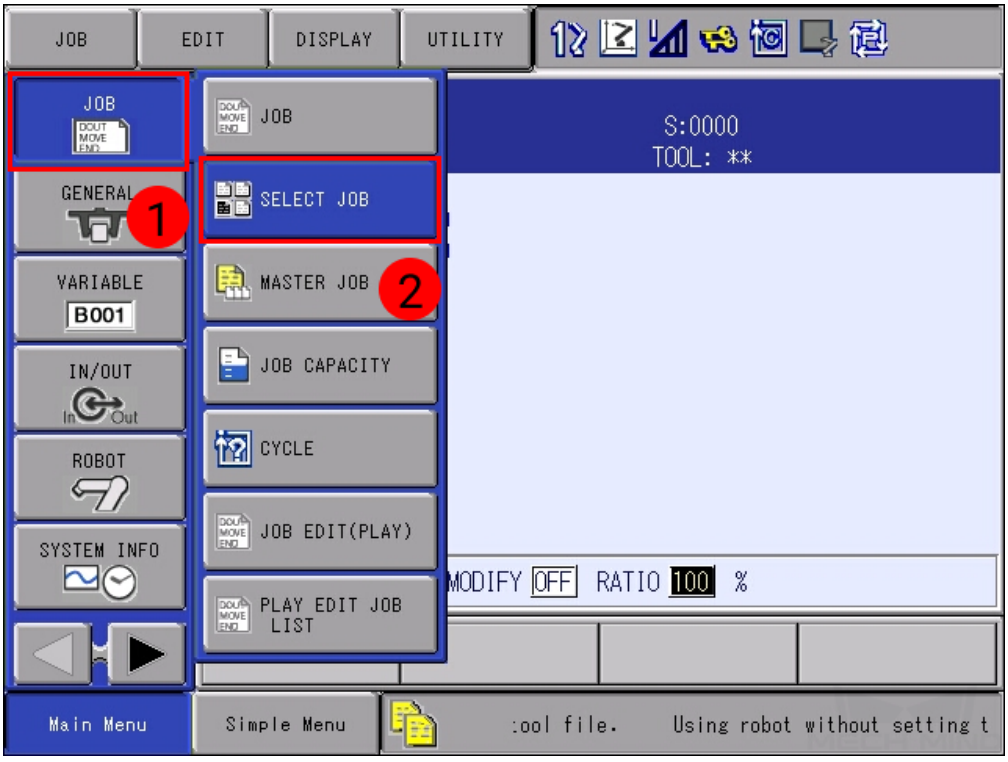
7. Restart the controller without pressing the `MAIN MENU` key; the program is now running automatically in the background. Turn the mode switch key to **PLAY**, and proceed to **Test Robot Connection**.

Test Robot Connection

Please refer to *Test Robot Connection* for detailed instructions on connecting to the robot in Mech-Center.

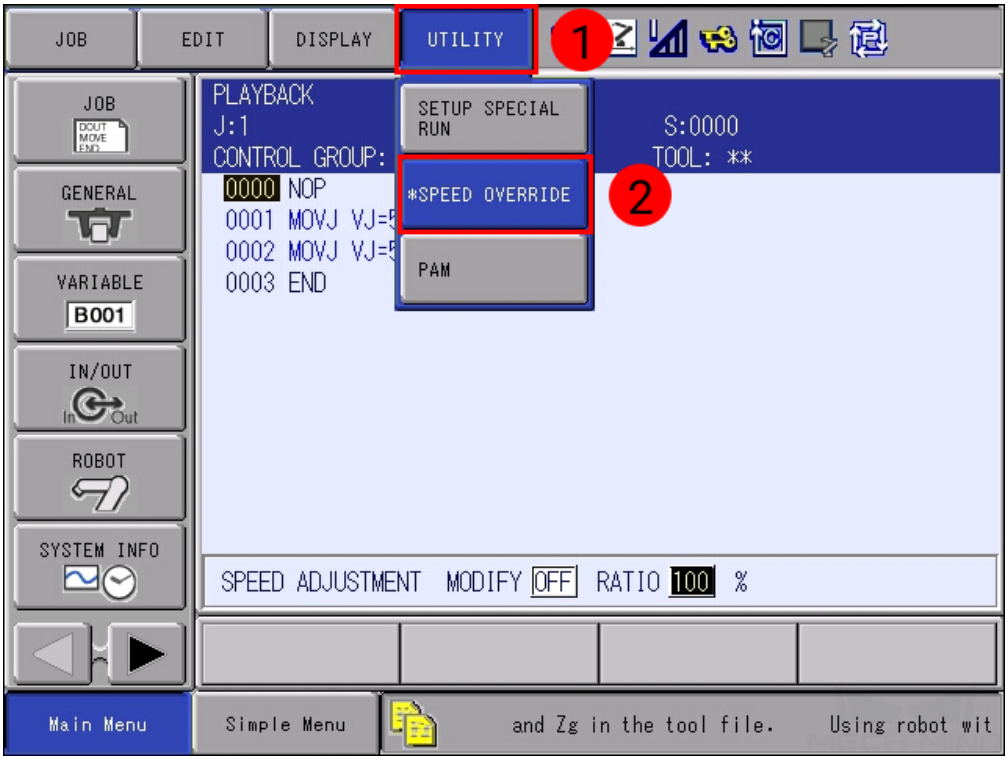
As the robot will move at 100% velocity by default, it is recommended to adjust its velocity before running the corresponding Mech-Viz project.

1. Select *JOB* → *SELECT JOB*.

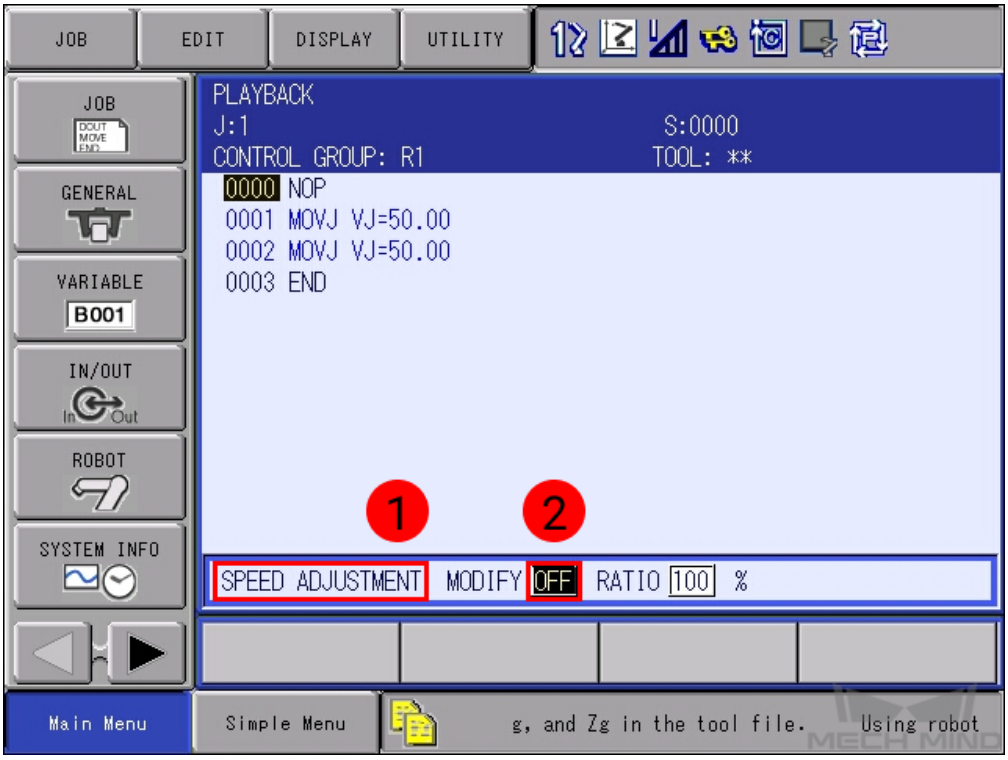


2. Select any job in the **JOB LIST**, and then press the SELECT key.

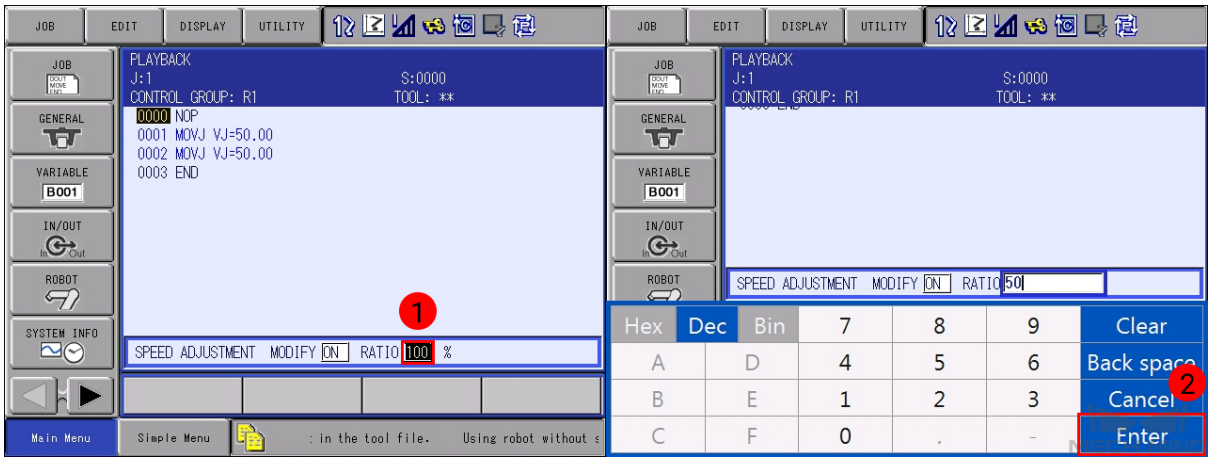
3. Select *UTILITY* → *SPEED OVERRIDE*.



4. Press on **SPEED ADJUSTMENT** and then **OFF**, press the SELECT key to switch **MODIFY** to **ON**.



5. Press on the number after RATIO, and press the SELECT key to change the speed ratio. Press on *Enter* to save the change.



1.2.2 YASKAWA Program Description

Occupied IO

Occupied IO	Signal
DI (16)	IN1–IN16
DO (16)	OT1–OT16

1.3 FANUC

This section introduces the full-control program for FANUC robots and the procedure of setting up the communication with a robot through the program.

1.3.1 FANUC Setup Instructions

This section introduces the process of loading the robot full-control program onto a FANUC robot.

The process consists of 4 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program Files*
- *Test Robot Connection*

Please have a flash drive ready at hand.

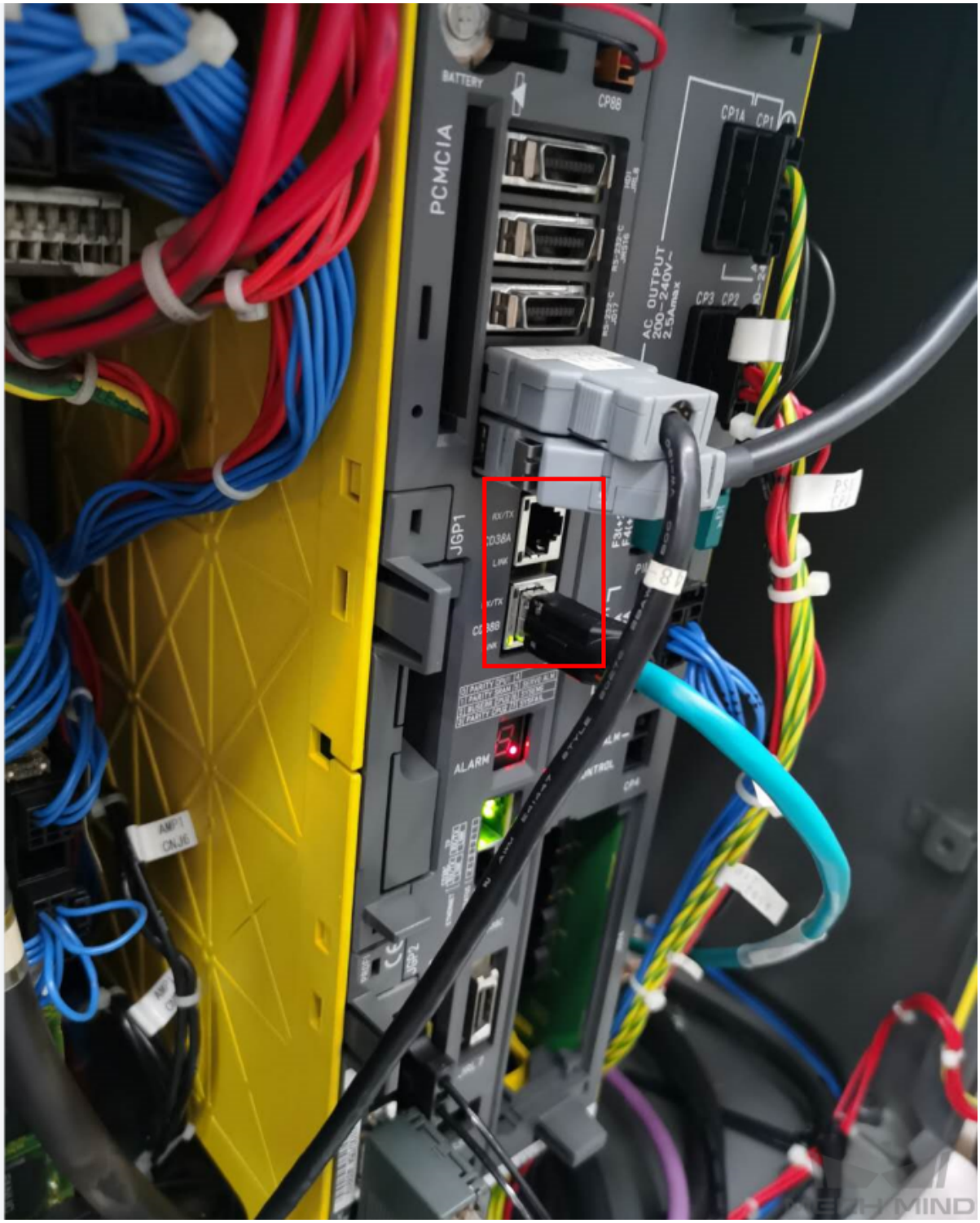
Check Controller and Software Compatibility

- Controller: no requirement
- Controller system software version: V7.5, V7.7, V8.*, and V9.*
- Additional controller software packages:
  - R651 or R632 (karel) - used to enable karel function
  - R648 (User Socket Msg)
- Mech-Center: latest version recommended

Setup the Network Connection

Hardware Connection

Plug the Ethernet cable of the IPC into the Ethernet port of robot controller as shown in the figure. You can plug the cable into either CD38A port or CD38B port. CD38A corresponds to **Port#1** in the robot IP setting, while CD38B corresponds to **Port#2**.

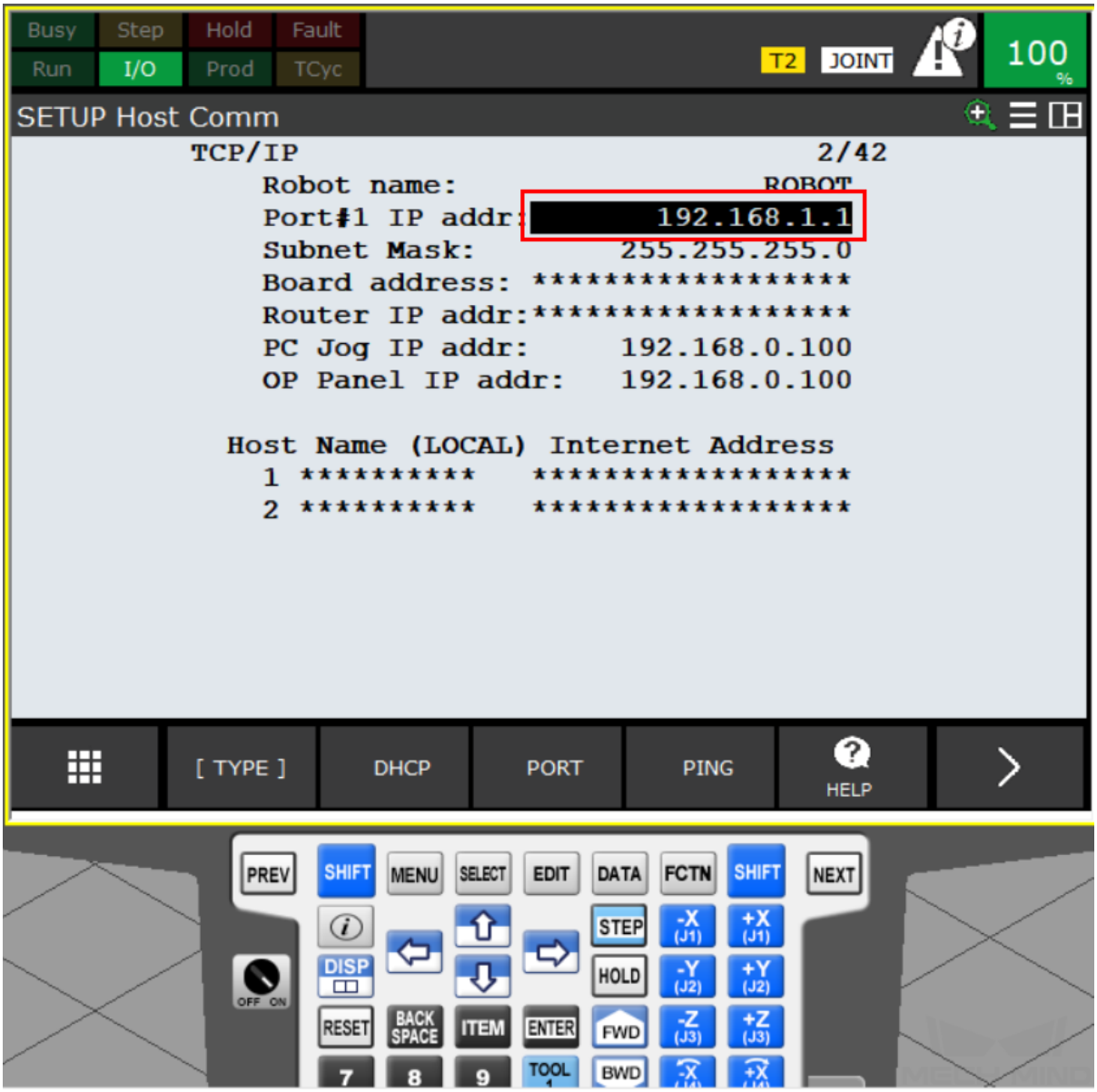

BATTERY
CP8B
PCMCIA
CP1A
CP1
AC OUTPUT
200~240V~
2.5Amax
CP3
CP2
JGP1
RX/TX
LINK
ALARM
CP4

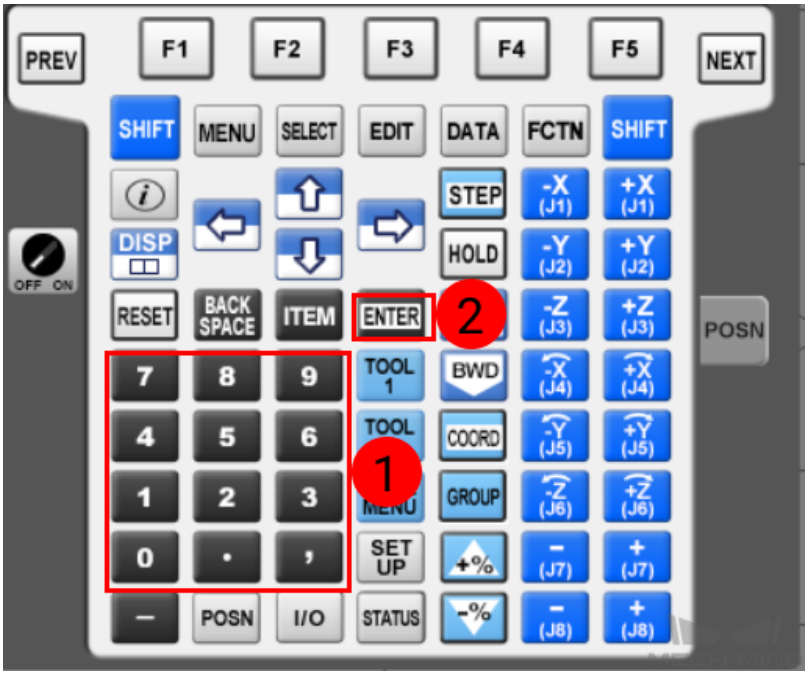
IP Configuration

1. Press on *MENU*→ *SETUP*, select **Host Comm** in the context menu, and then press `ENTER` to open the **SETUP Protocols** window.

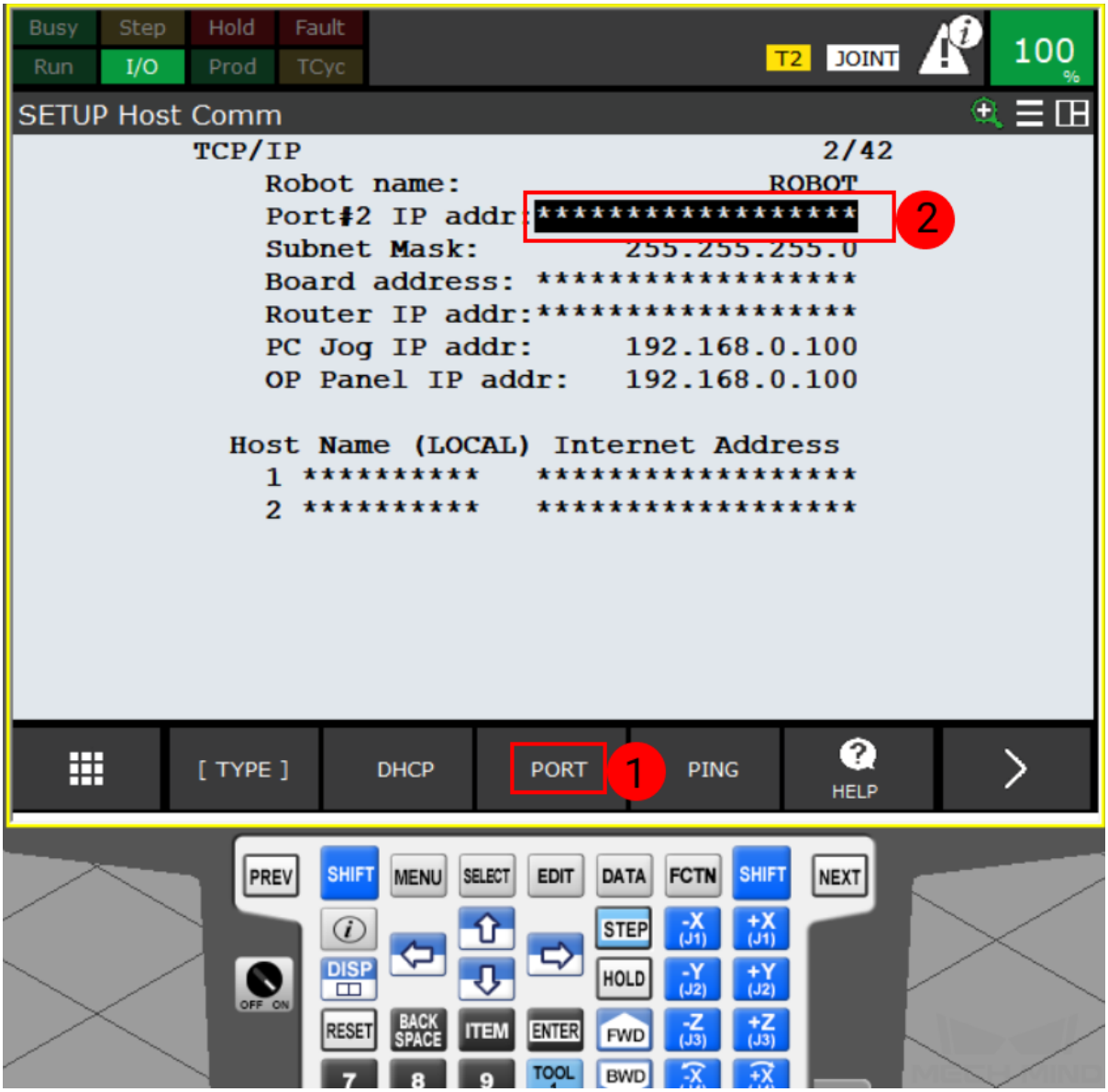
2. Select **TCP/IP** and press on *DETAIL* to open the **SETUP Host Comm** window.

3. Enter the robot IP in the **IP address** line with the keyboard of the teach pendant. The robot IP should be in the same subnet as the IPC.


Busy
Step
Hold
Fault
Run
I/O
Prod
TCyc
T2
JOINT
100
%
SETUP Host Comm
TCP/IP
2/42
Robot name:
ROBOT
Port#1 IP addr:
192.168.1.1
Subnet Mask:
255.255.255.0
Board address: **********
Router IP addr:**********
PC Jog IP addr:
192.168.0.100
OP Panel IP addr:
192.168.0.100
Host Name (LOCAL) Internet Address
1 **********
****
2 **********
****
[TYPE]
DHCP
PORT
PING
HELP
PREV
SHIFT
MENU
SELECT
EDIT
DATA
FCTN
SHIFT
NEXT
DISP
STEP
HOLD
RESET
BACK SPACE
ITEM
ENTER
FWD
BWD
TOOL
-X (J1)
+X (J1)
-Y (J2)
+Y (J2)
-Z (J3)
+Z (J3)
OFF ON
7
8
9



4. If the Ethernet cable is connected to port 2, please press *Port* to switch the port. Then you can enter the robot IP in the **IP address** line.



Load the Program Files

Prepare the Files

The program file is stored in the installation directory of Mech-Center. The default directory for Mech-Center 1.5.2 is *C:/Mech-Mind/Mech-Center*.

Navigate to *xxx/Mech-Center/Mech-RobServ/install_packages/fanuc*, and copy all the contents of this folder to your flash drive:

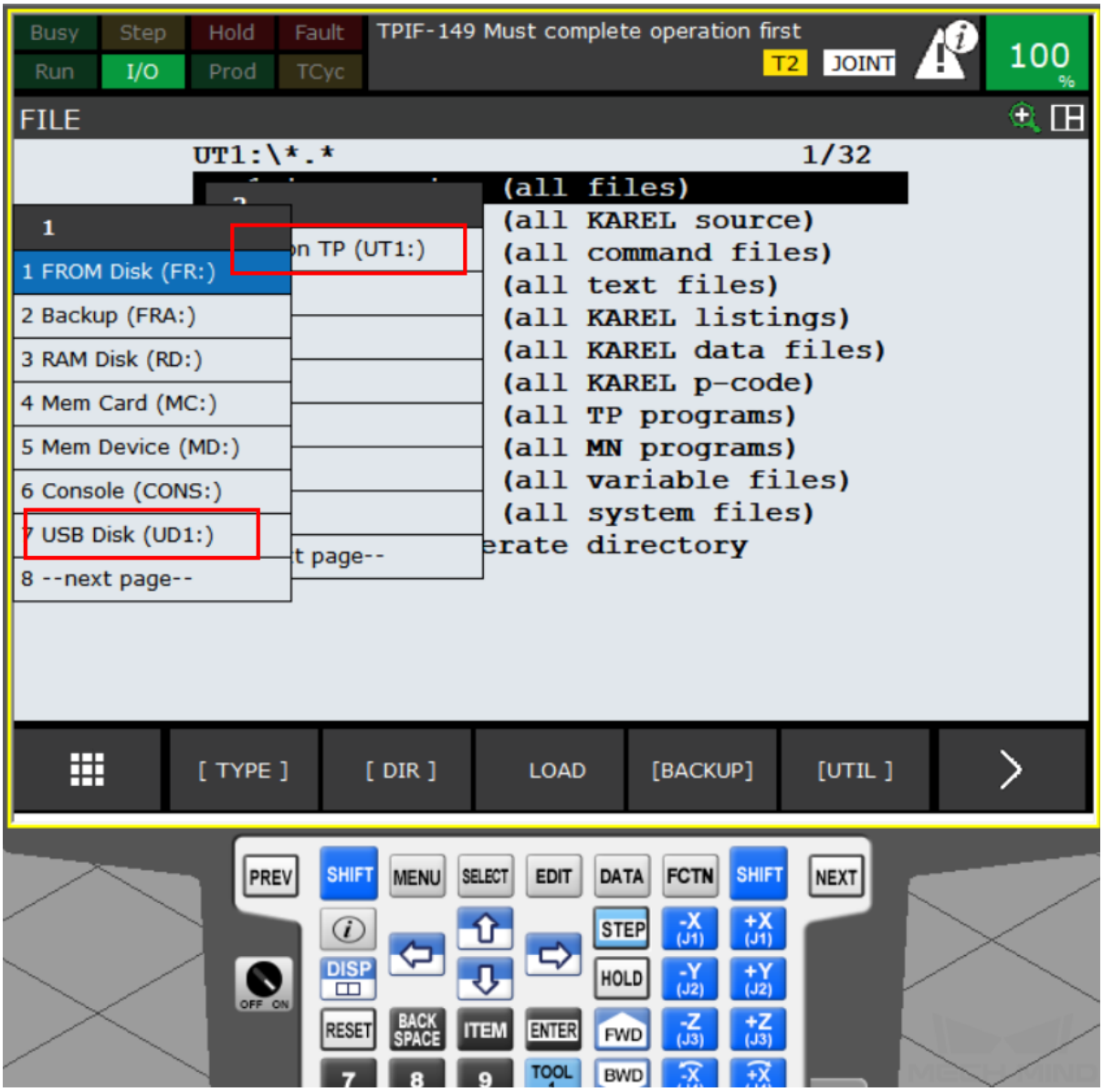
Note: The folders and files must be saved in the root directory of the USB flash drive. Do not rename them.

Load the Files to the Robot

1. After connecting the USB flash drive, press `MENU` and select *FILE* → *File*, and then press `ENTER` to open the **FILE** window.

2. Press *UTIL* and select **Set Device** in the context menu.

3. Select the USB flash drive. If your flash drive is connected to the **controller**, please select **USB Disk (UD1:)**; if your USB flash drive is connected to the **teach pendant**, please select **USB on TP (UT1:)**.



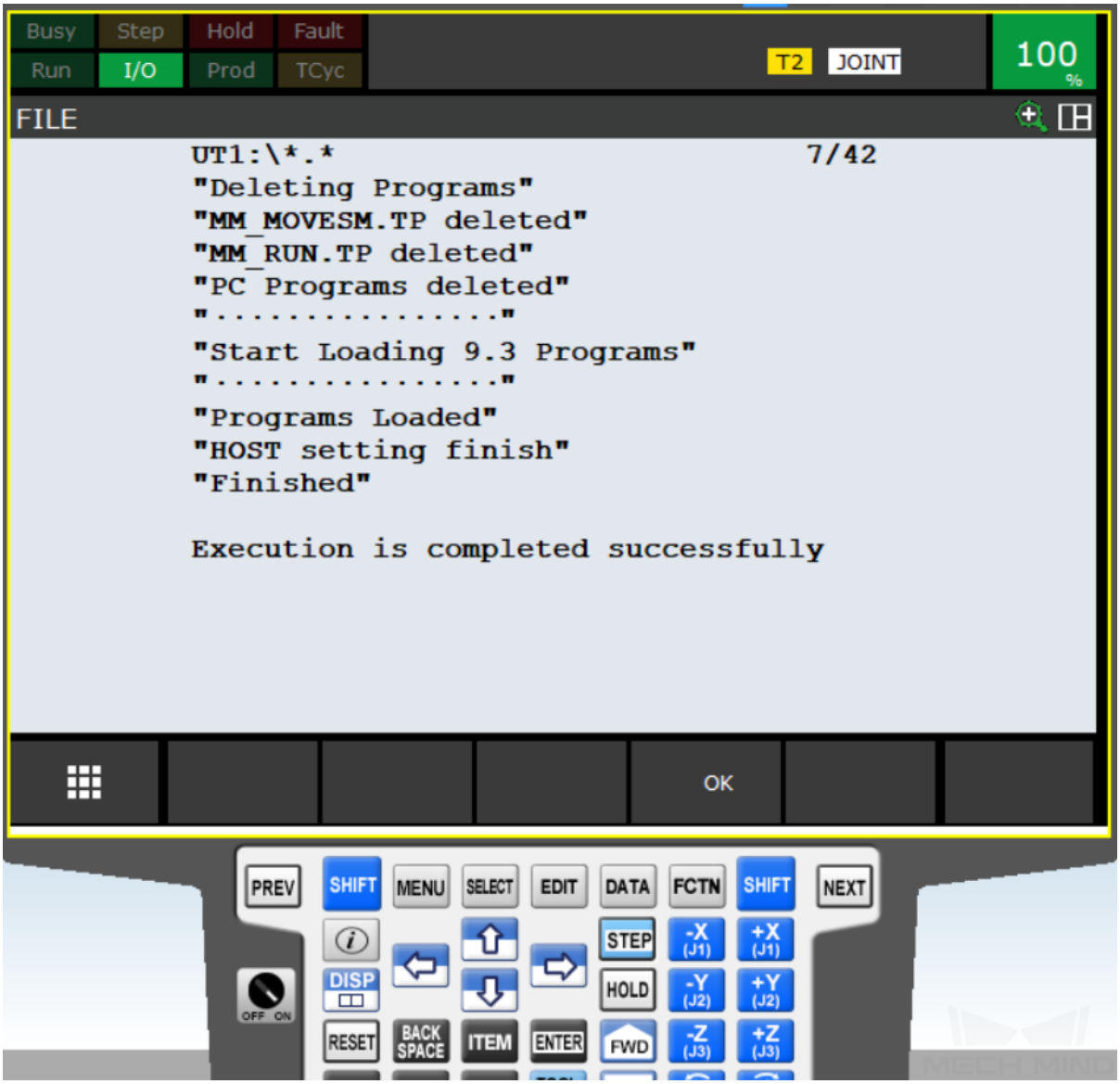
4. Select the first line **(all files)** and press `ENTER` to enter the root directory of the USB flash drive.

Hint: For the next step:

- If the USB flash drive is connected to the **robot controller**, please select **INSTALL_UD.cm**.
- If the USB flash drive is connected to the **robot teach pendant**, please select **INSTALL_UT.cm**.

1. Select the corresponding CM file and press `ENTER` key on the teach pendant. Choose *YES* to start loading the programs.

2. When the following screen is displayed, the loading and relavent configuration are completed. Press F4 to exit the program.



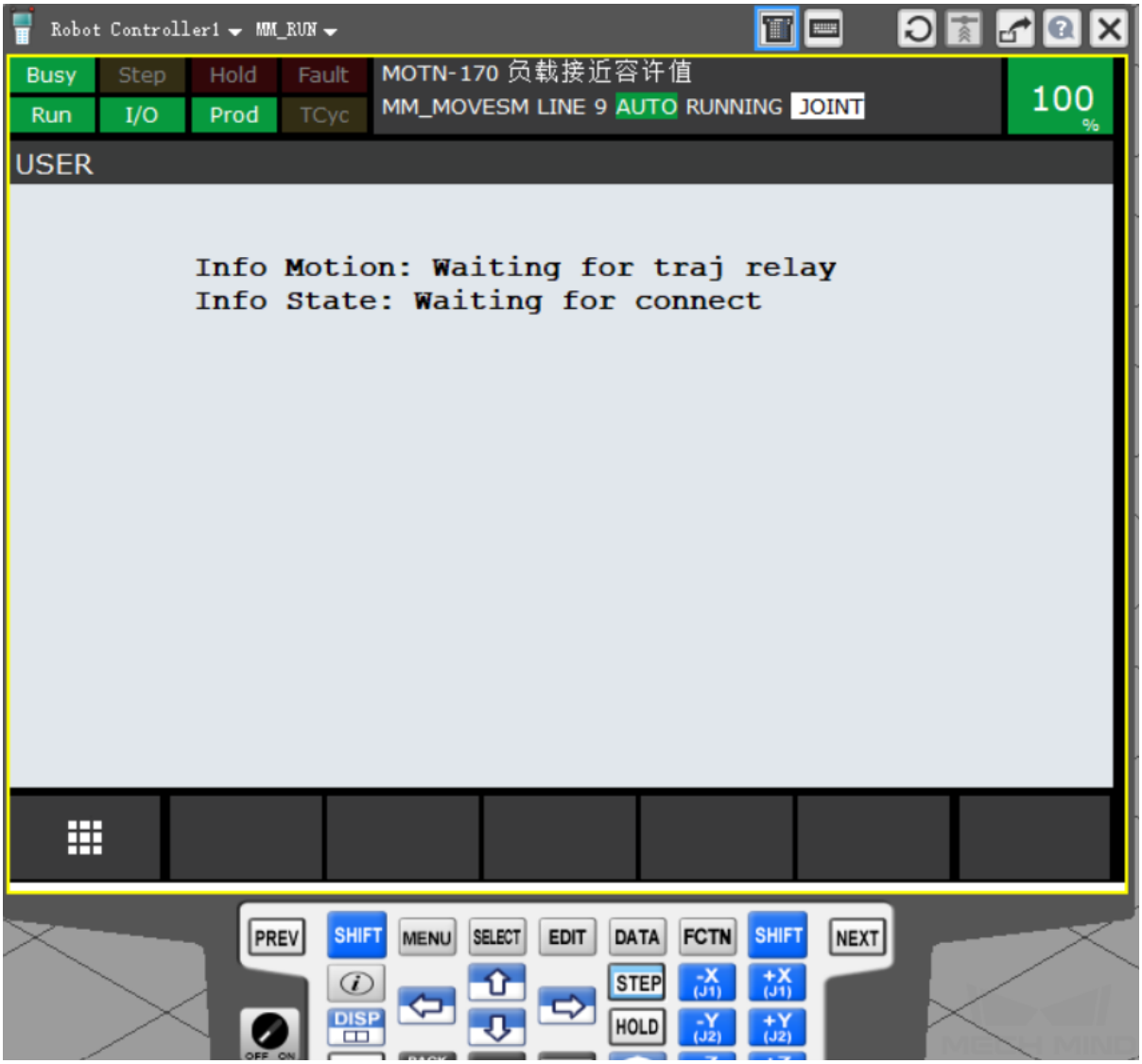
Attention: Please restart the robot after exiting the program.

Run the Program

1. Press SELECT key on the teach pendant to open the program selection window, and then select *TYPE→ TP Programs*, as shown below.

2. Select **MM_RUN** and press ENTER to open the program. Then press the green button on the controller to auto-run the program.

3. If the following screen is displayed, you can proceed to the next section.



Test Robot Connection

Please refer to *Test Robot Connection* for detailed instructions.

1.3.2 FANUC Program Description

Program Module

Program Modules	Description
mm_relay	background program used to receive Robserver data
mm_state	background program used to send data of robot pose, signal, and status
mm_movesm	foreground program used to guide the robot to move
mm_moveid	background program used to write the data received by mm_relay to the register
mm_run	automatically run the foreground and background program after running this module
mm_struct_h	defined struct data

Occupied Registers

Register		Description
MOVE_SPD_REG	180	integer register: motion speed (in %)
MOVE_CNT_REG	181	integer register: motion termination (in %)
MOVE_TYP_REG	182	indicate the move type J or L
MOVL_RG_SPD	183	move L speed
RI_C_BRANCH	184	mm control branch

Position Register		Description
MOVE_PREG	80	position register for current point

Occupied FLAGs

FLAG		Description
F_MSM_RDY	180	ready signal flag
F_MSM_DRDY	181	data ready signal flag
F_TK_CTRL	182	takes control flag
RI_MM_LOCK	183	trajectory lock flag
R_STOP_MOVE	190	stop move flag
F_STOP_CLR	191	clear data flag

1.4 KUKA

This section introduces the full-control program for KUKA robots and the procedure of setting up the communication with a robot through the program.

1.4.1 KUKA Setup Instructions

This section introduces the process of loading the robot full-control program onto a KUKA robot.

The process consists of 4 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program Files*
- *Test Robot Connection*

Please have a flash drive ready at hand.

Check Controller and Software Compatibility

Compatibility requirements are as follows:

- Controller model: KUKA KR C4
- Controller system software version: KSS 8.2 to 8.6
- Add-on software package: Ethernet KRL (V 2.2.8 or above)
- Mech-Center: latest version recommended

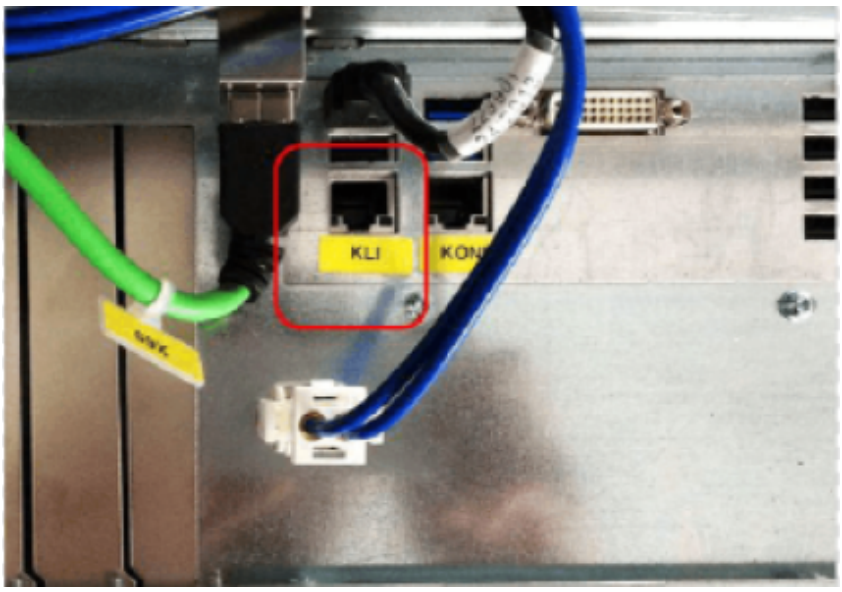
Note: All teach pendant actions in this chapter are performed on KSS 8.6. The specific steps and menu selections may differ slightly in older versions of system software.

Setup the Network Connection

Hardware Connection

Plug the Ethernet cable into:

- An Ethernet port on the IPC
- The X66 port on KR C4 compact and KLI port on other KR C4 controllers

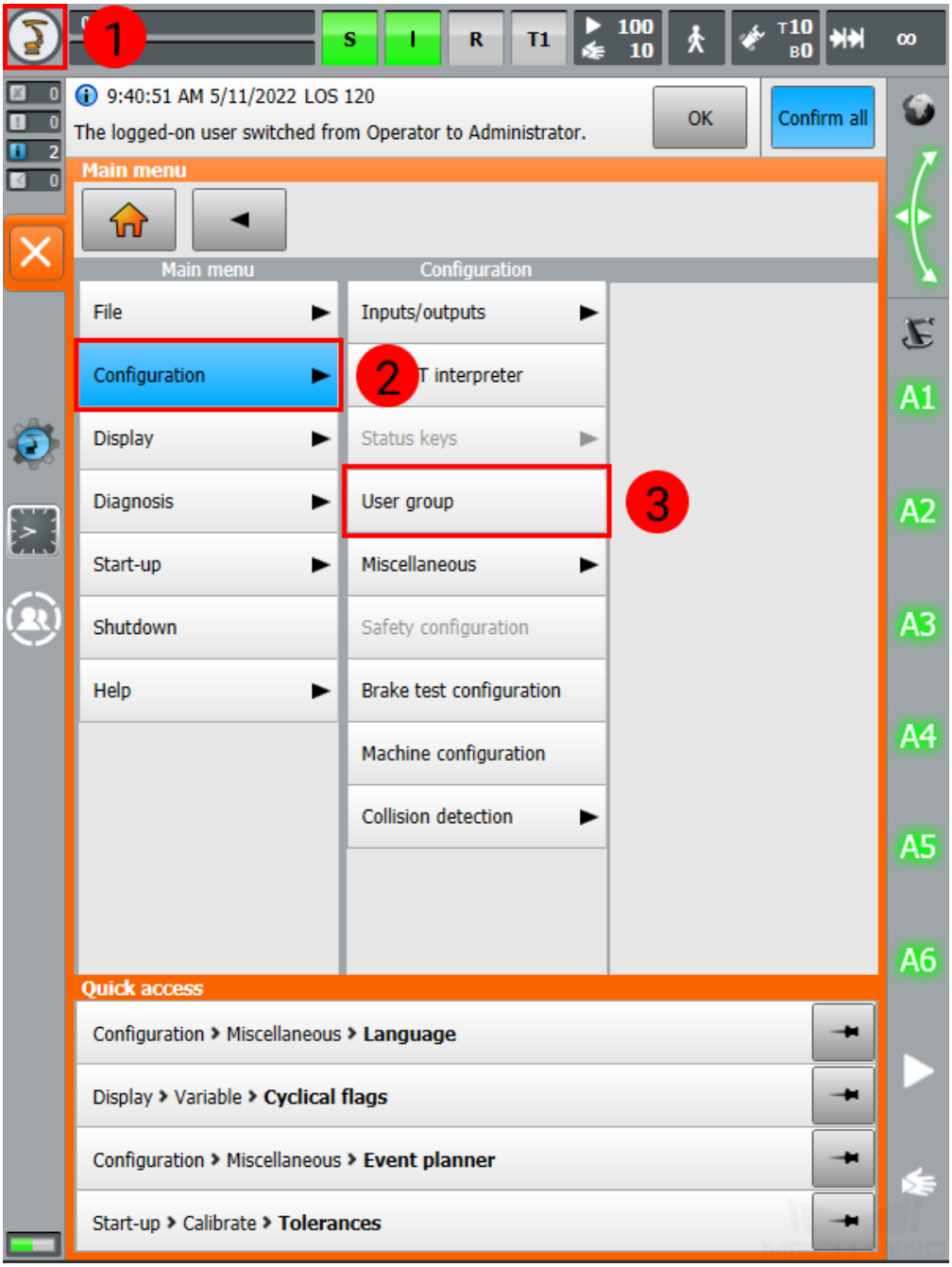


IP Configuration

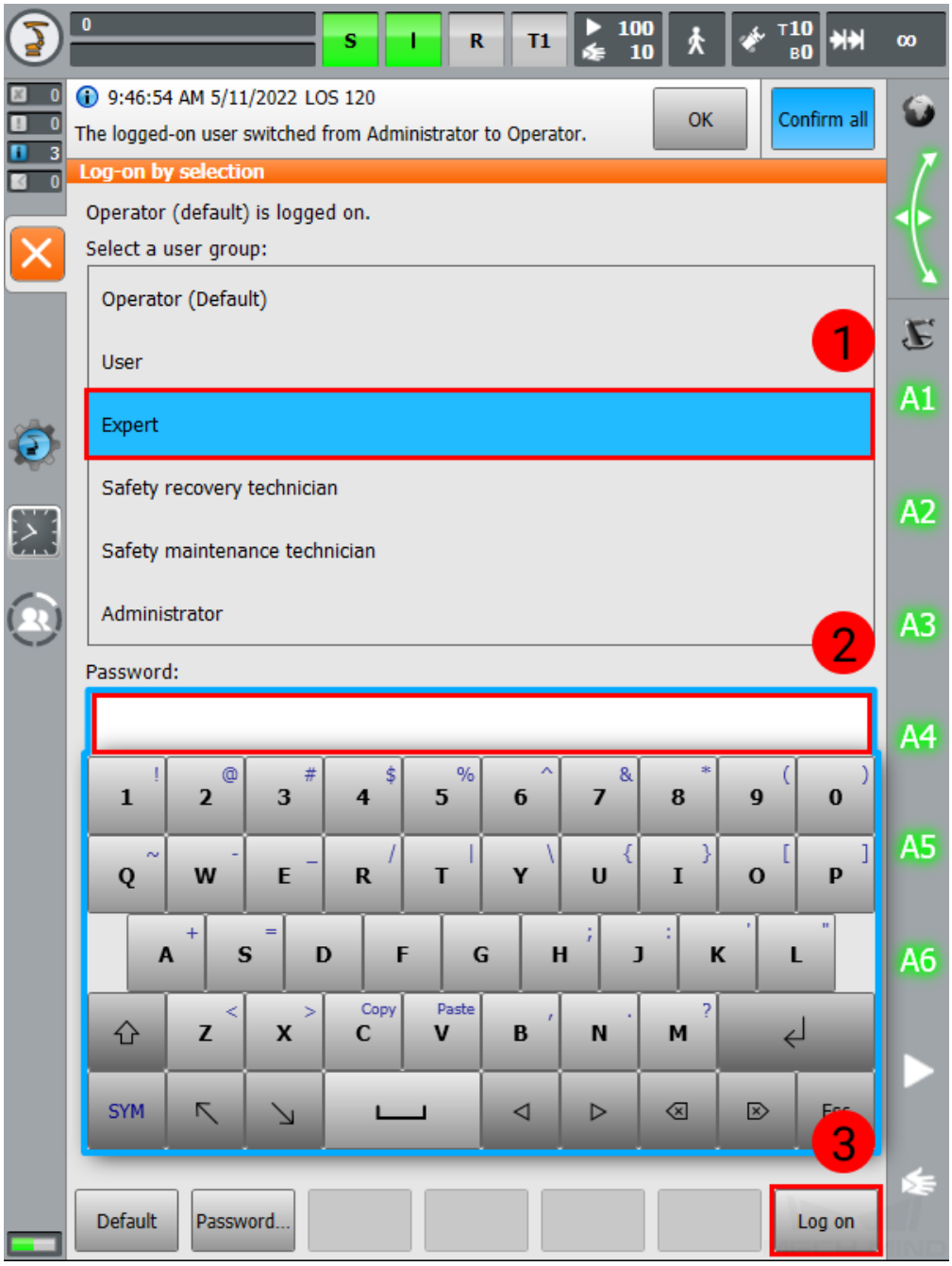
To allow communication between the IPC and the robot controller, both must have an IP address in the same subnet. This means that the first three numbers of the IP addresses should be the same. For example, 192.168.100.1 and 192.168.100.2 are in the same subnet.

1. Check the IP address of the IPC: please use the *ipconfig* command in Command Prompt or PowerShell to check the IP address.
2. Switch to expert mode:

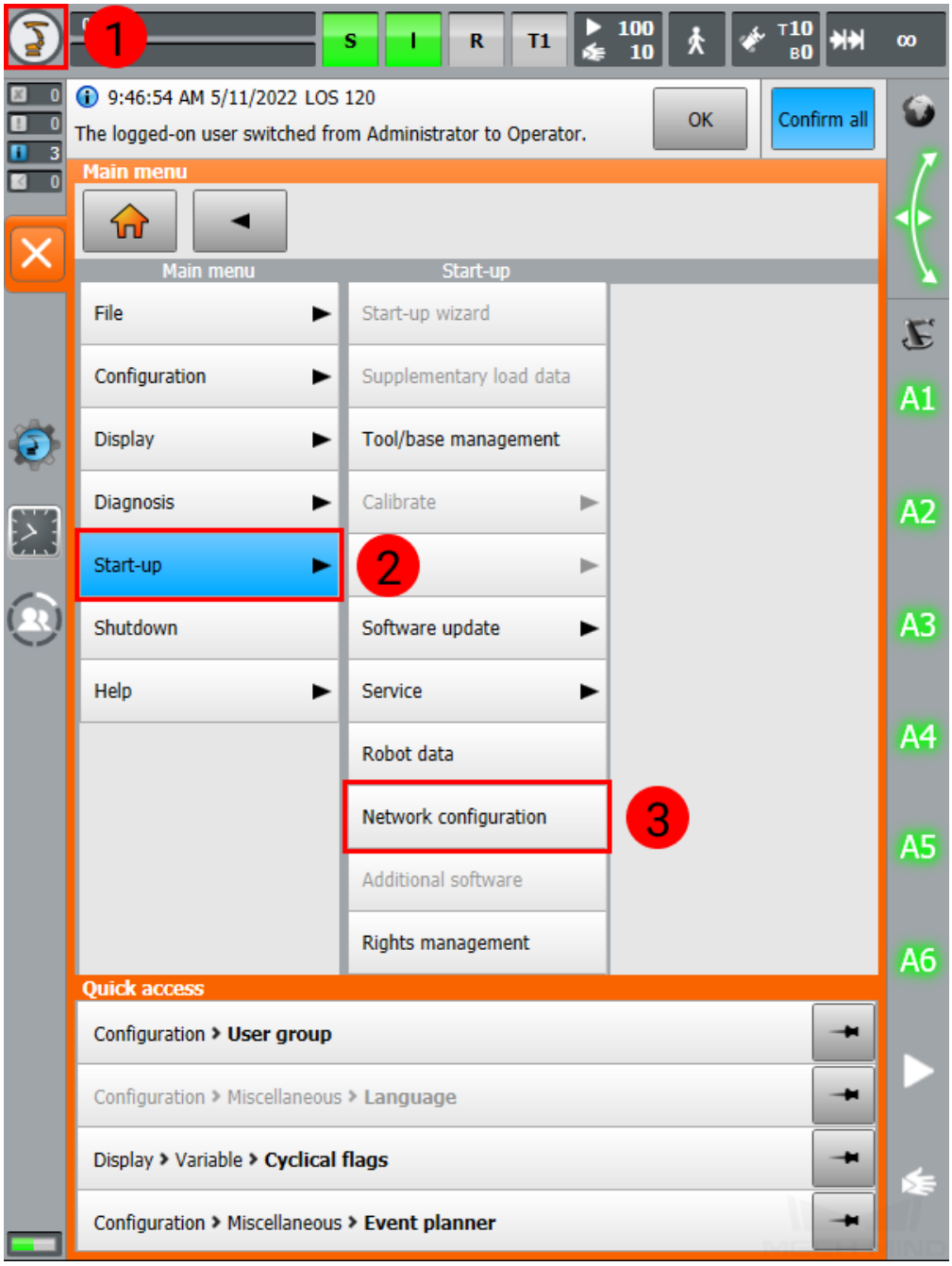
1. Press on  , and then select *Configuration* → *User group*.



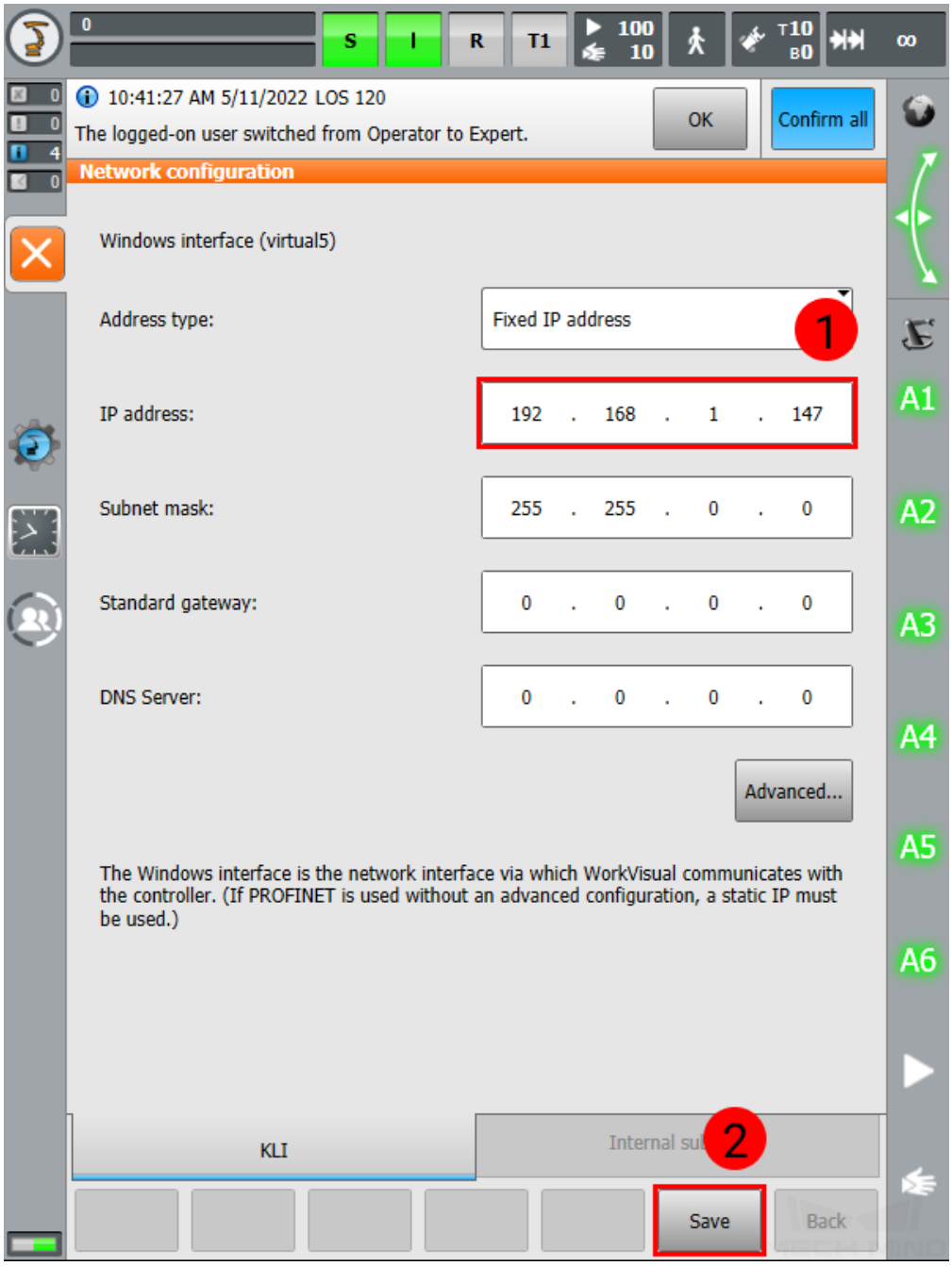
2. Select **Expert**, enter the password (the default password is **kuka**), and press on *Log on*.

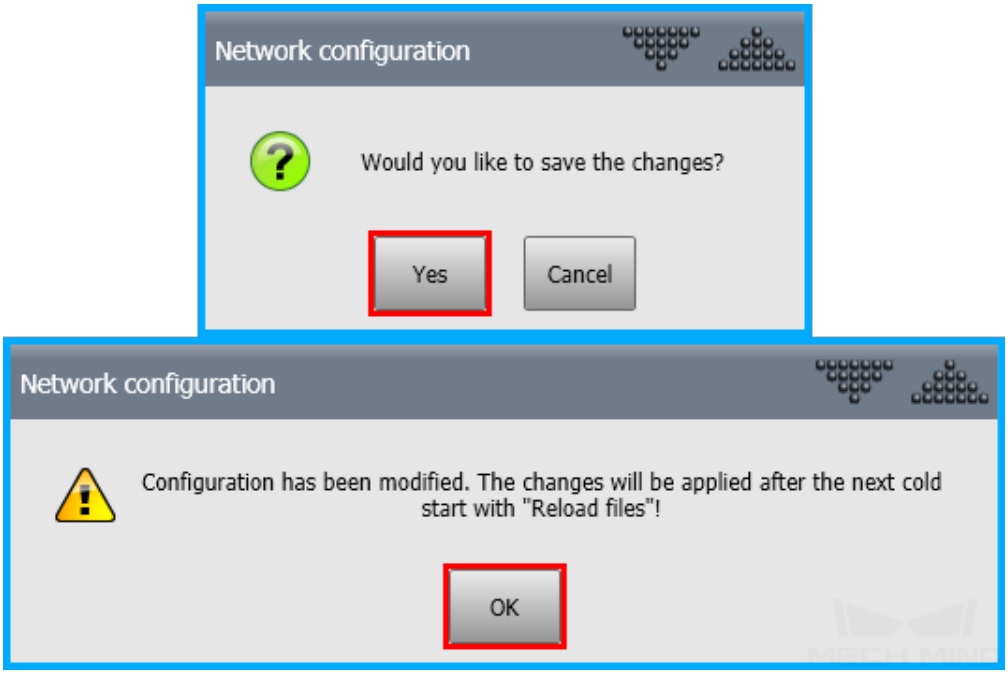


3. Press on  , and then select *Start-up* → *Network configuration*.



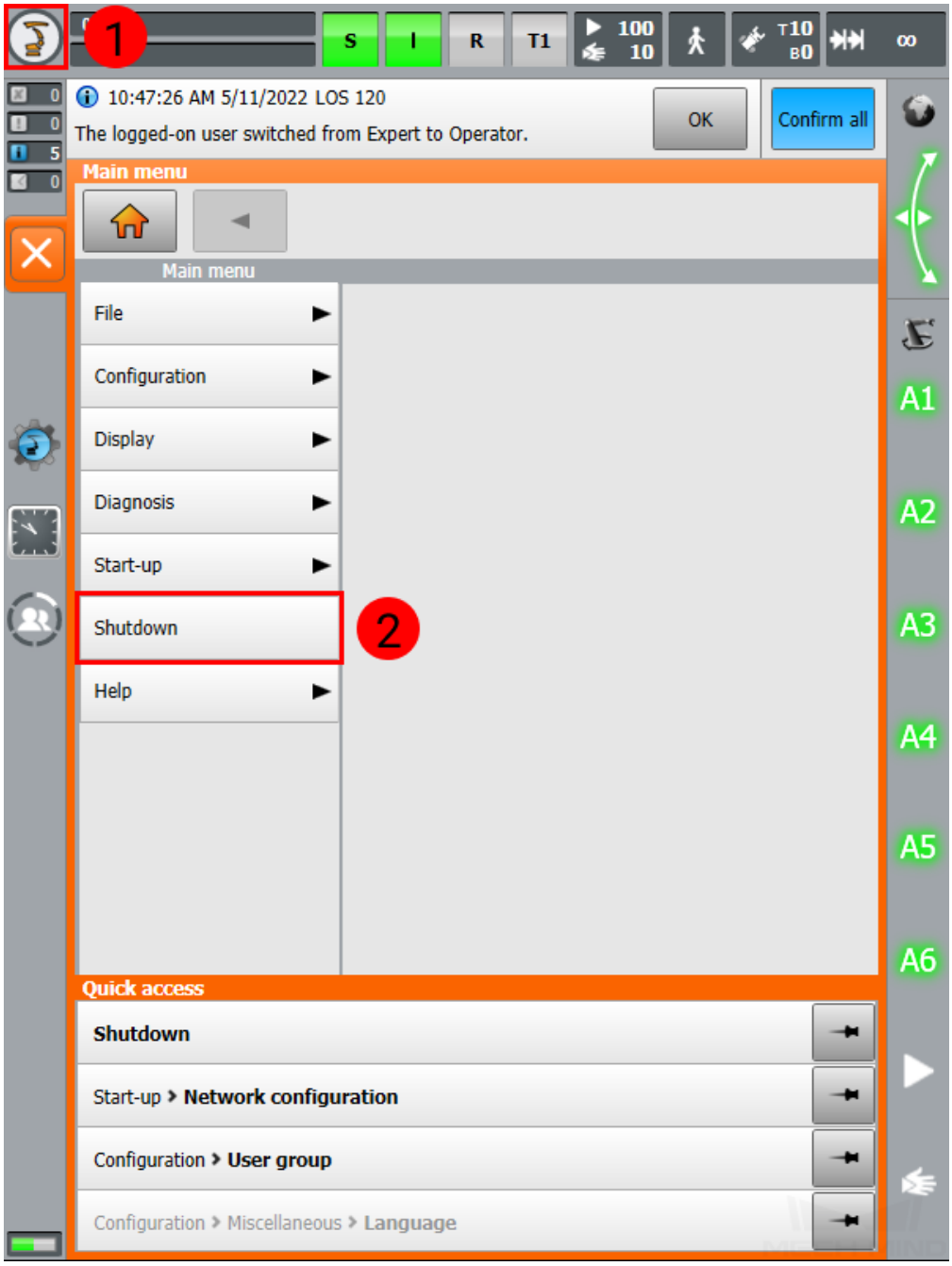
4. Input an **IP address** in the same subnet as that of the IPC, and then press on *Save*. In the next two pop-up windows, press on *Yes* and *OK*, respectively.


10:41:27 AM 5/11/2022 LOS 120
The logged-on user switched from Operator to Expert.
OK
Confirm all
Network configuration
Windows interface (virtual5)
Address type:
Fixed IP address
1
IP address:
192 . 168 . 1 . 147
Subnet mask:
255 . 255 . 0 . 0
Standard gateway:
0 . 0 . 0 . 0
DNS Server:
0 . 0 . 0 . 0
Advanced...
The Windows interface is the network interface via which WorkVisual communicates with the controller. (If PROFINET is used without an advanced configuration, a static IP must be used.)
KLI
Internal su
2
Save
Back
A1
A2
A3
A4
A5
A6

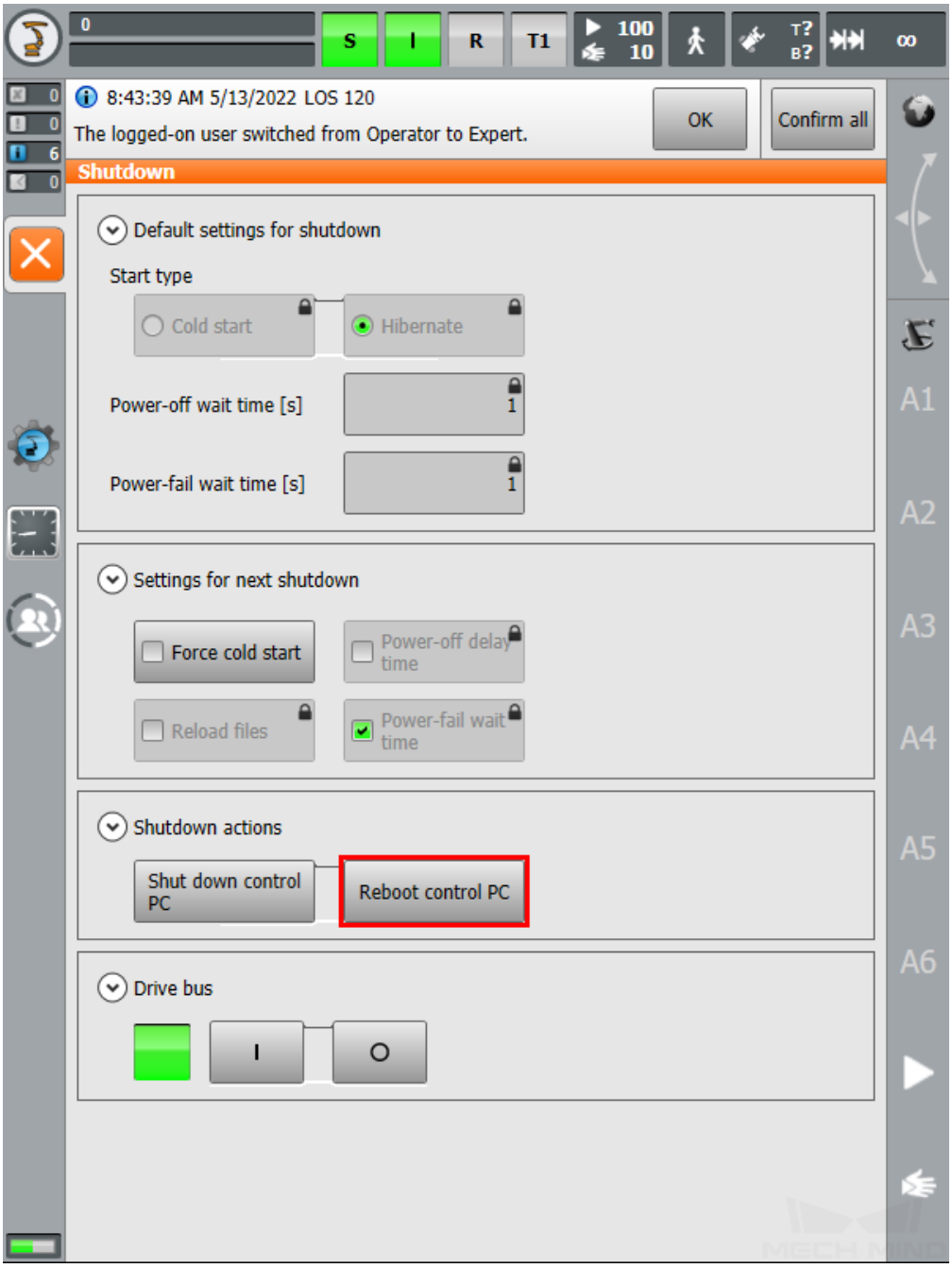


5. Restart the robot to finish setting the IP address:

   1. Press on  , and select **Shutdown**.



2. Press on *Reboot control PC*.


8:43:39 AM 5/13/2022 LOS 120
The logged-on user switched from Operator to Expert.
OK
Confirm all
Shutdown
Default settings for shutdown
Start type
Cold start
Hibernate
Power-off wait time [s]
1
Power-fail wait time [s]
1
Settings for next shutdown
Force cold start
Power-off delay time
Reload files
Power-fail wait time
Shutdown actions
Shut down control PC
Reboot control PC
Drive bus
I
O

Load the Program Files

Prepare the Files

The program files are stored in the installation directory of Mech-Center. The default directory for Mech-Center 1.5.2 is *C:/Mech-Mind/Mech-Center*.

Navigate to *xxx/Mech-Center/Mech-RobServ/install_packages/kuka/kuka_new*, and copy the following files to your flash drive.

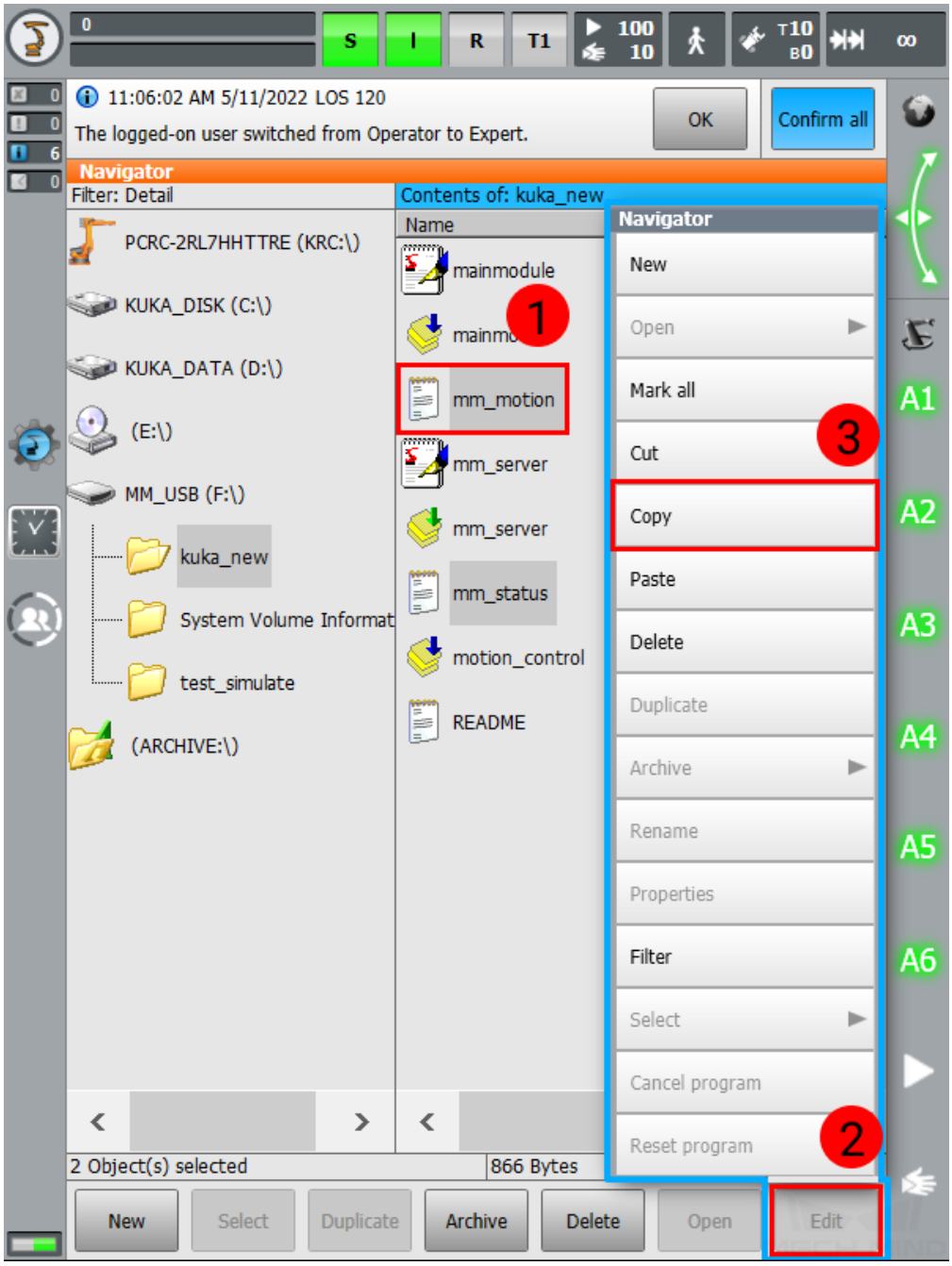
- **mm_motion.xml**
- **mm_status.xml**
- **mm_server.dat**
- **mm_server.sub**
- **motion_control.src**
- **mainmodule.src**
- **mainmodule.dat**

Load the Files to the Robot

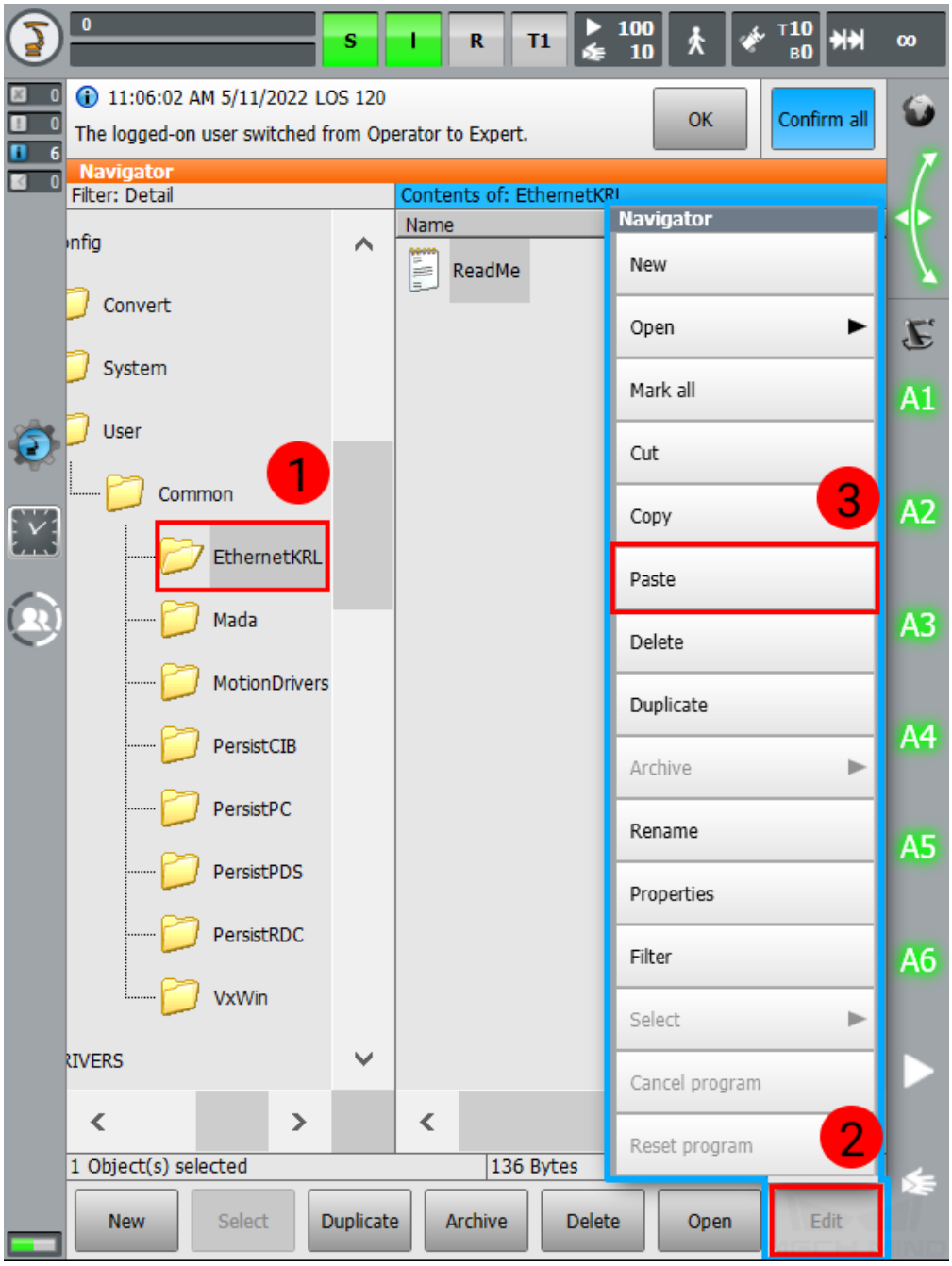
Note: Make sure you have switched to expert mode on the teach pendant. For instructions, see step 2 in **IP Configuration**.

---

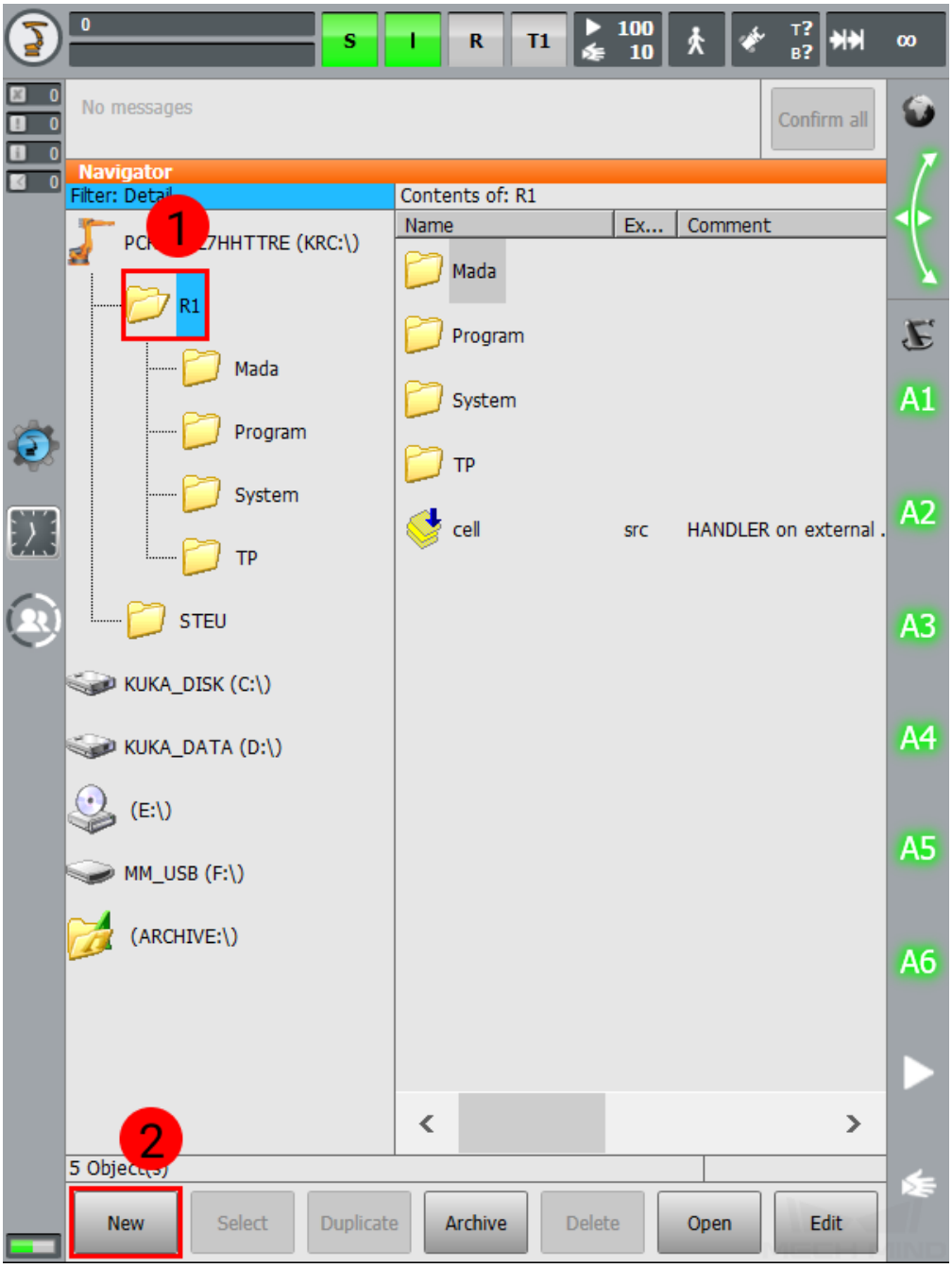
1. Plug the flash drive to the controller.
2. Select the flash drive, and locate the above files.
3. Select **mm_motion.xml**, press on *Edit*, and then select **Copy**.



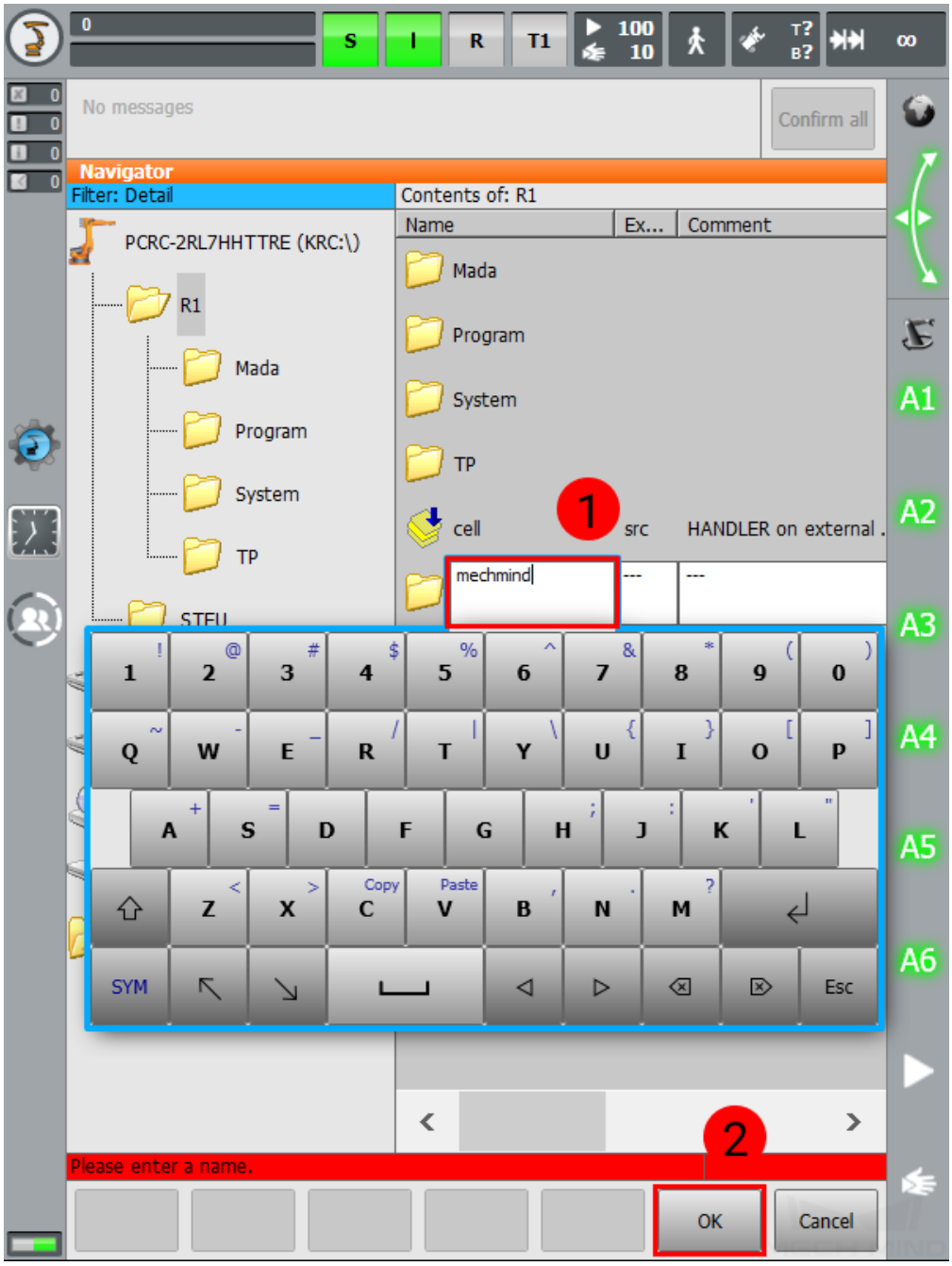
4. Navigate to *C:/KRC/ROBOTER/Config/User/Common/EthernetKRL*, press on *Edit*, and then select **Paste**.



5. Navigate back to the flash drive, and repeat steps 3 and 4 for **mm_status.xml**.
6. Navigate to *KRC:/R1*, and press on *New*.

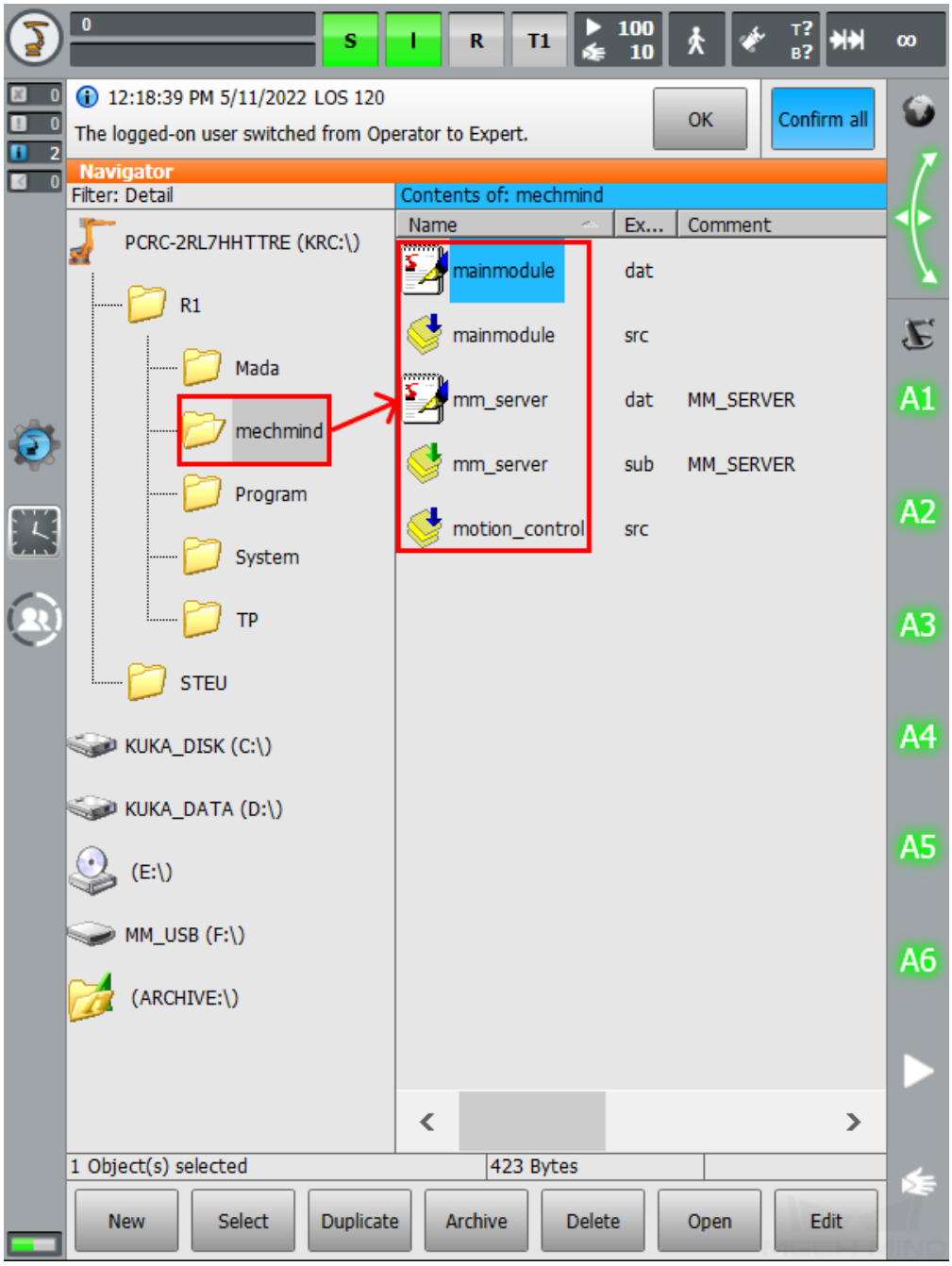


7. Input **mechmind** for the folder name, and press on *OK*.



8. Navigate back to the flash drive, and copy and paste the other 5 files to *KRL:/R1/mechmind*.

Note: Long-press and drag to select multiple adjacent files.

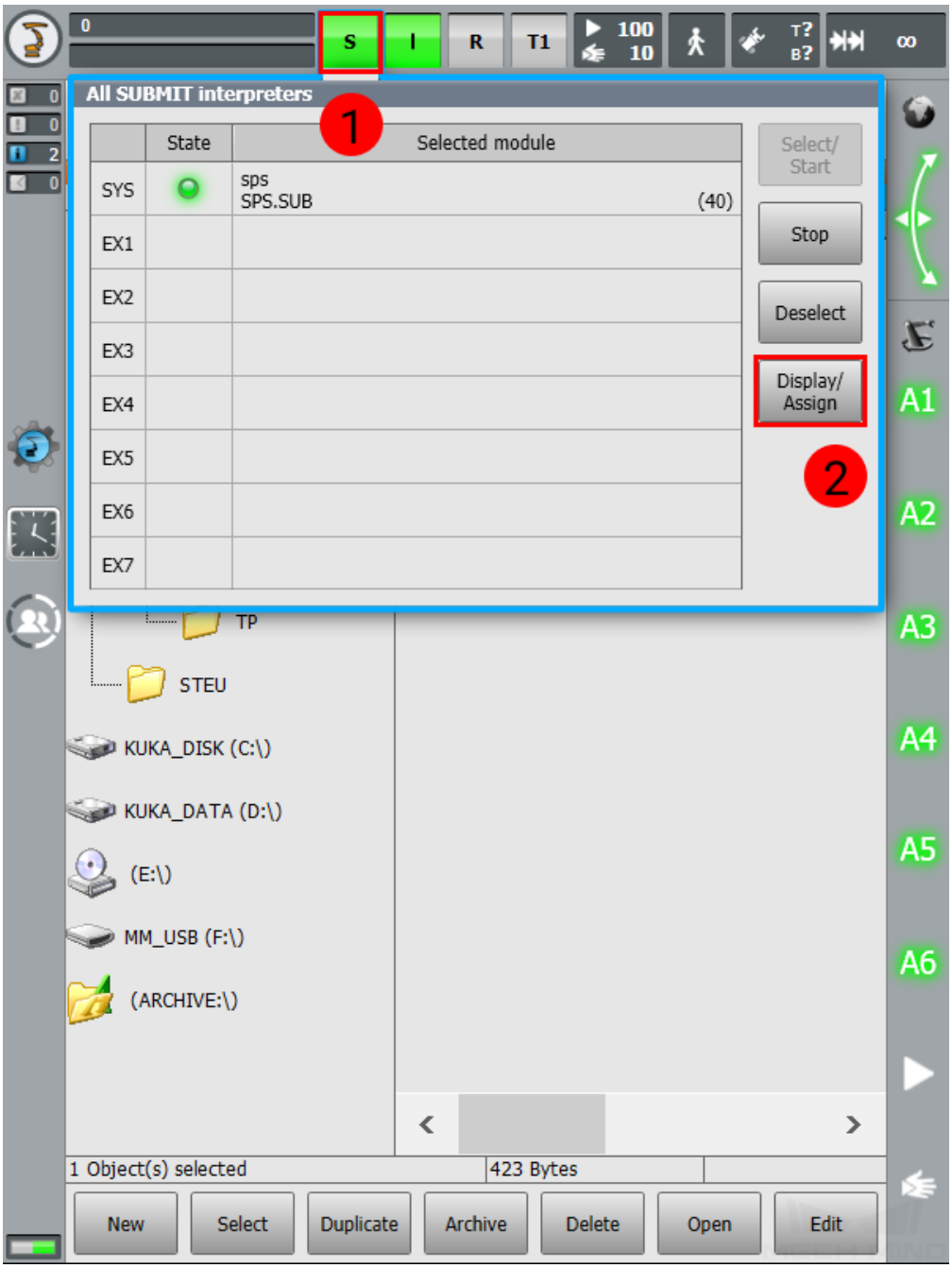

12:18:39 PM 5/11/2022 LOS 120
The logged-on user switched from Operator to Expert.
OK
Confirm all
Navigator
Filter: Detail
Contents of: mechmind
Name
Ex...
Comment
PCRC-2RL7HHTTRE (KRC:\)
R1
Mada
mechmind
Program
System
TP
STEU
KUKA_DISK (C:\)
KUKA_DATA (D:\)
(E:\)
MM_USB (F:\)
(ARCHIVE:\)
mainmodule dat
mainmodule src
mm_server dat MM_SERVER
mm_server sub MM_SERVER
motion_control src
1 Object(s) selected
423 Bytes
New
Select
Duplicate
Archive
Delete
Open
Edit

Set Autostart for Background Porgram

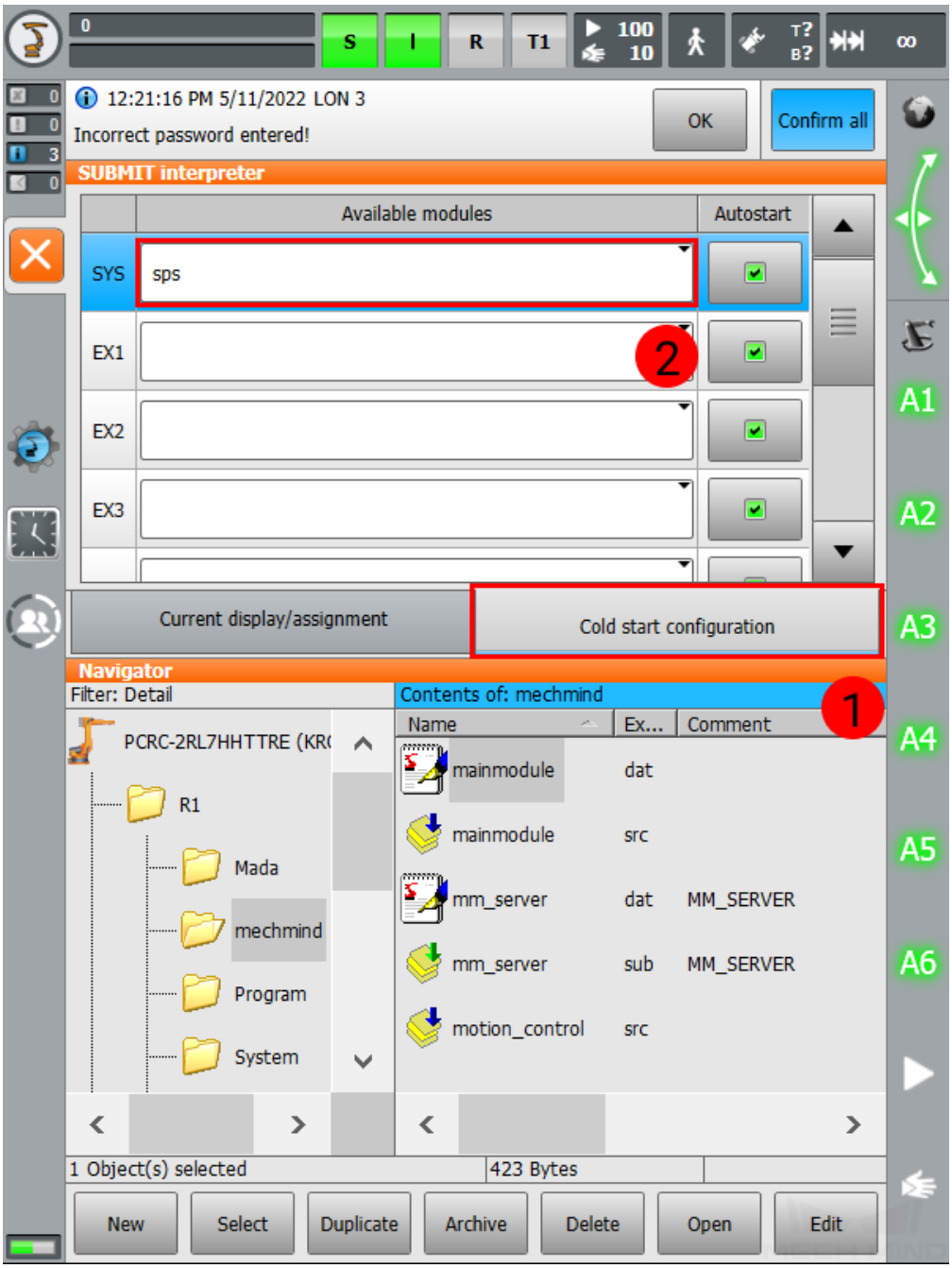
Note: Make sure you have switched to expert mode on the teach pendant. For instructions, see step 2 in **IP Configuration**.

1. Press on 

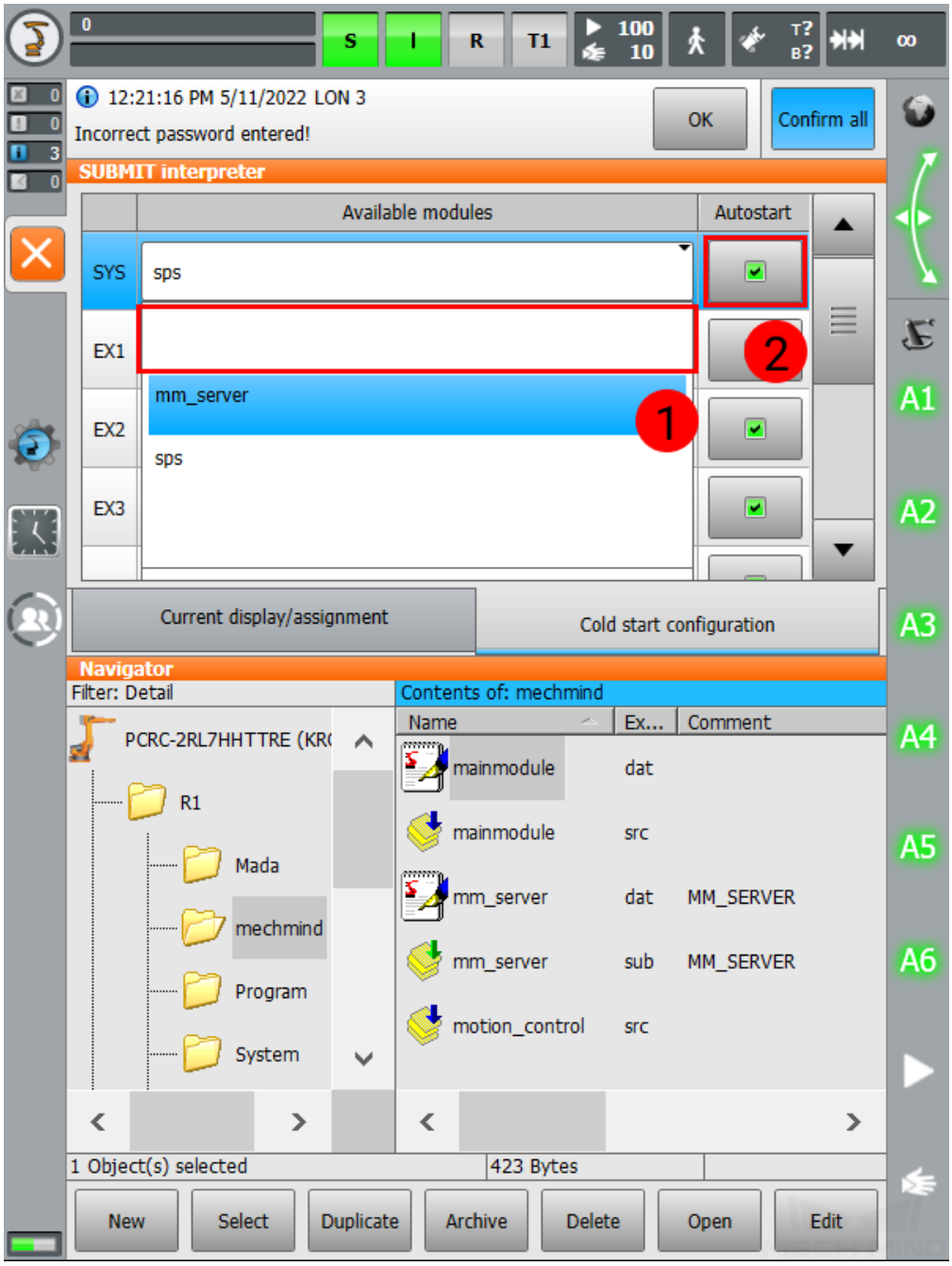
and then *Display/Assign*.



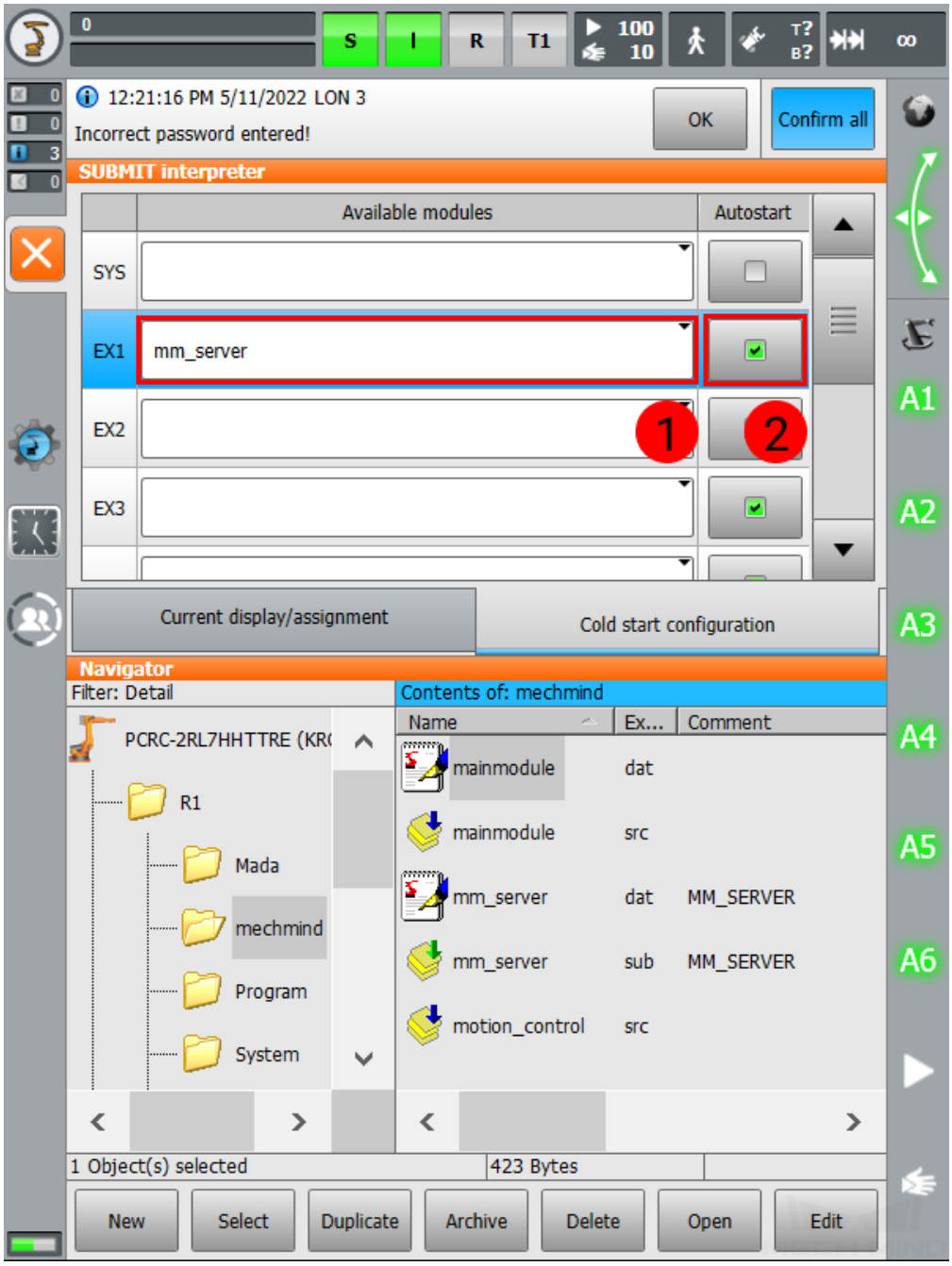
2. Press on **Cold start configuration**, and then press on the text box to the right of **SYS**.



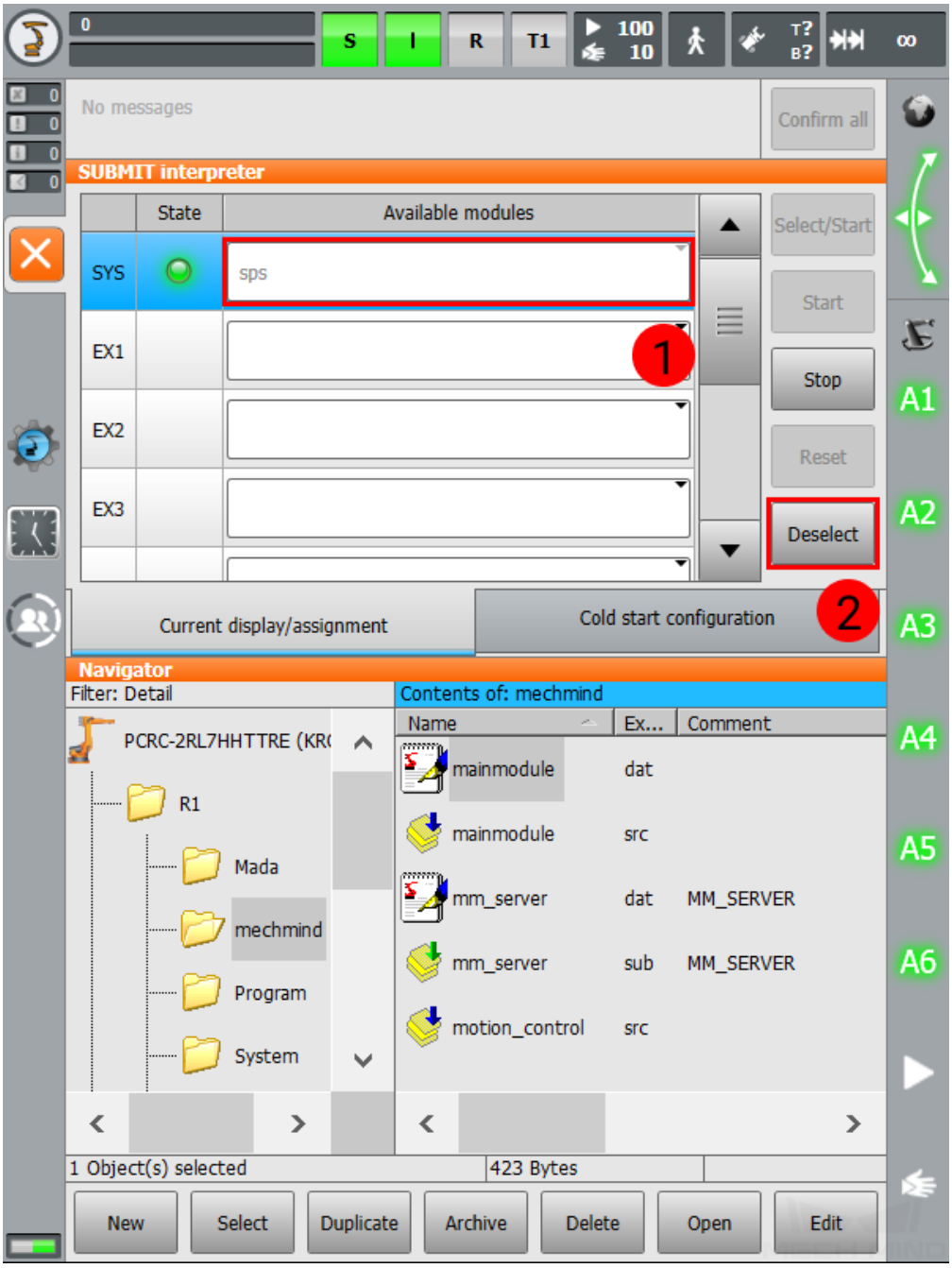
3. Select the blank in the drop-down menu, and then press on the check-box in **Autostart** to uncheck it.



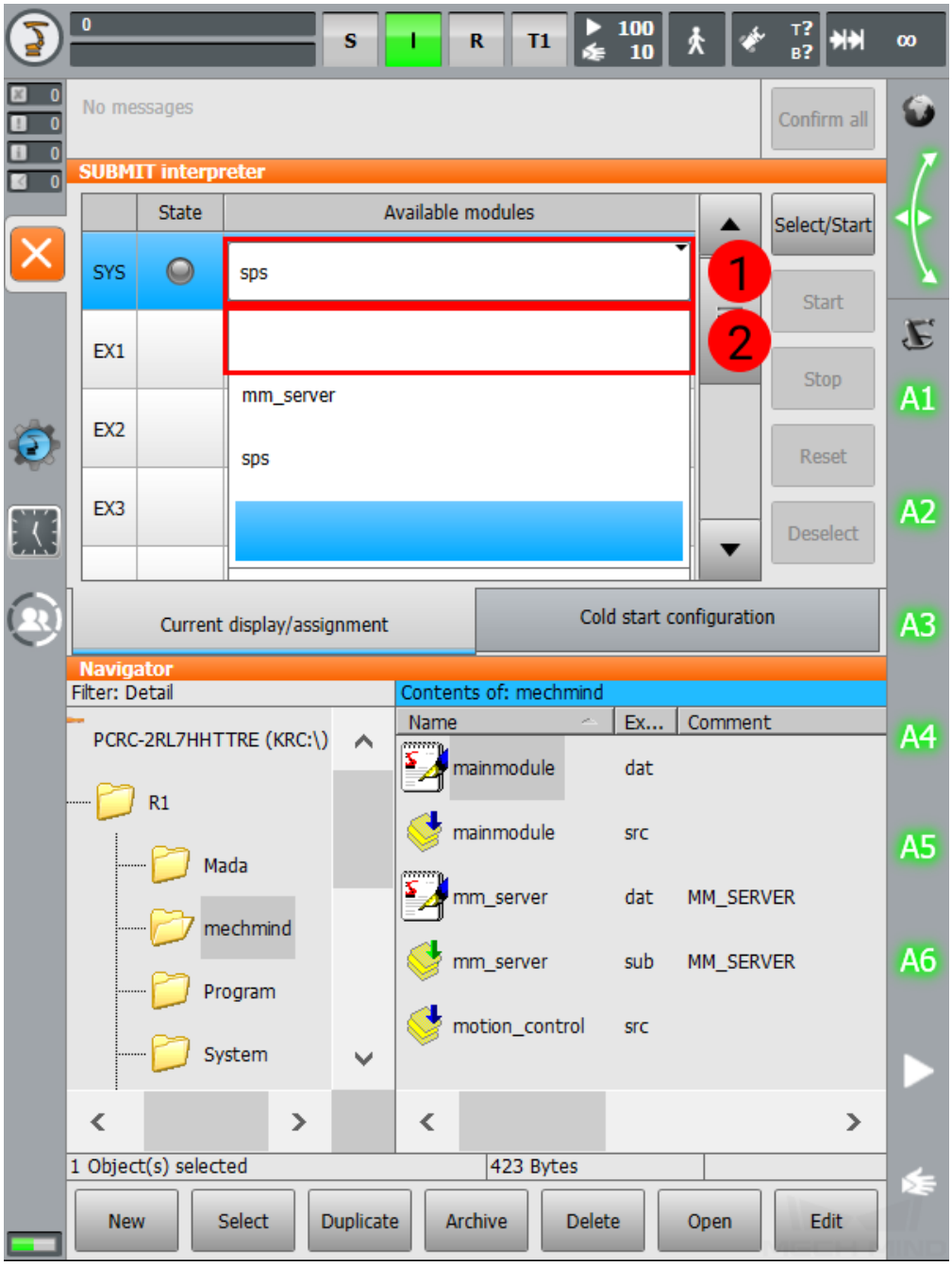
4. Similarly, select **mm_server** from the drop-down menu for **EX1**, and make sure **Autostart** is checked.



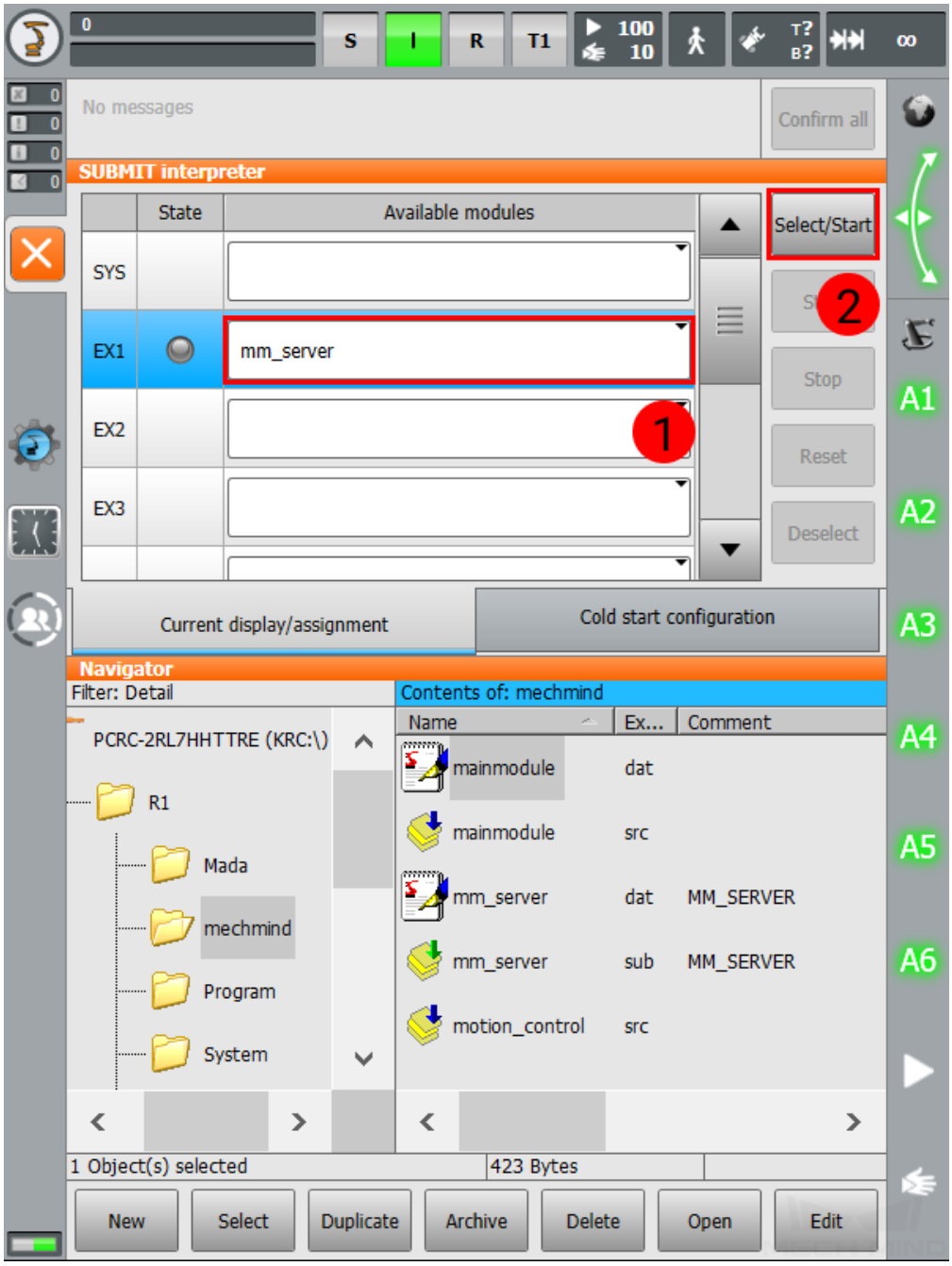
5. Press on **Current display/assignment**, press on the text box to the right of **SYS**, and then press *Deselect*.



6. Press on the text box to the right of **SYS** again, and select the blank in the drop-down menu.

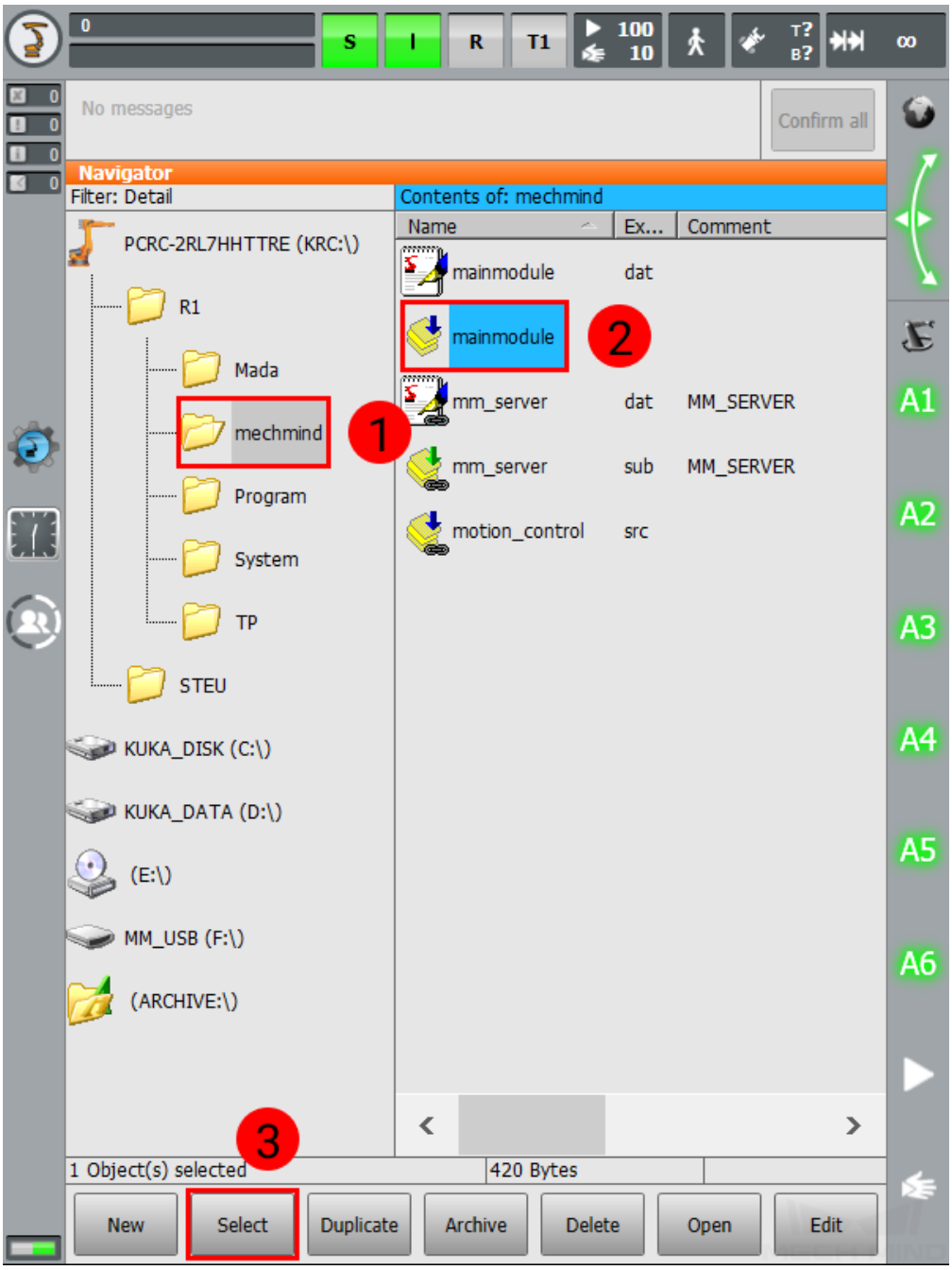


7. Similarly, select **mm_server** from the drop-down menu for **EX1**, and press on *Select/Start*.

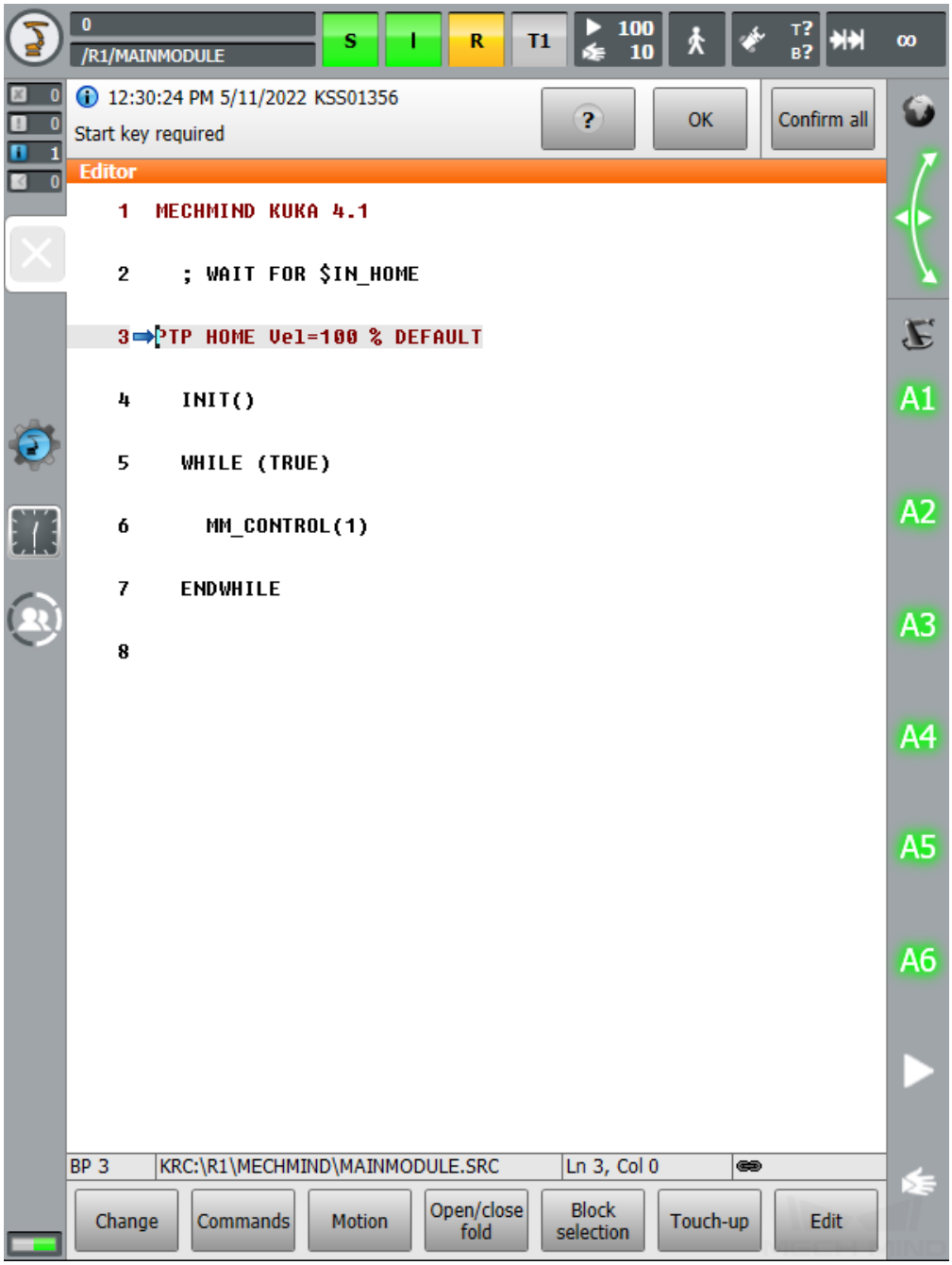

No messages
Confirm all
SUBMIT interpreter
State
Available modules
SYS
EX1
mm_server
EX2
EX3
Select/Start
Stop
Reset
Deselect
1
2
Current display/assignment
Cold start configuration
Navigator
Filter: Detail
Contents of: mechmind
PCRC-2RL7HHTTRE (KRC:\)
R1
Mada
mechmind
Program
System
Name
Ex...
Comment
mainmodule
dat
mainmodule
src
mm_server
dat
MM_SERVER
mm_server
sub
MM_SERVER
motion_control
src
1 Object(s) selected
423 Bytes
New
Select
Duplicate
Archive
Delete
Open
Edit
A1
A2
A3
A4
A5
A6

Select Foreground Program

1. Open the **mechmind** folder, select **mainmodule.src**, and then press on *Select*.



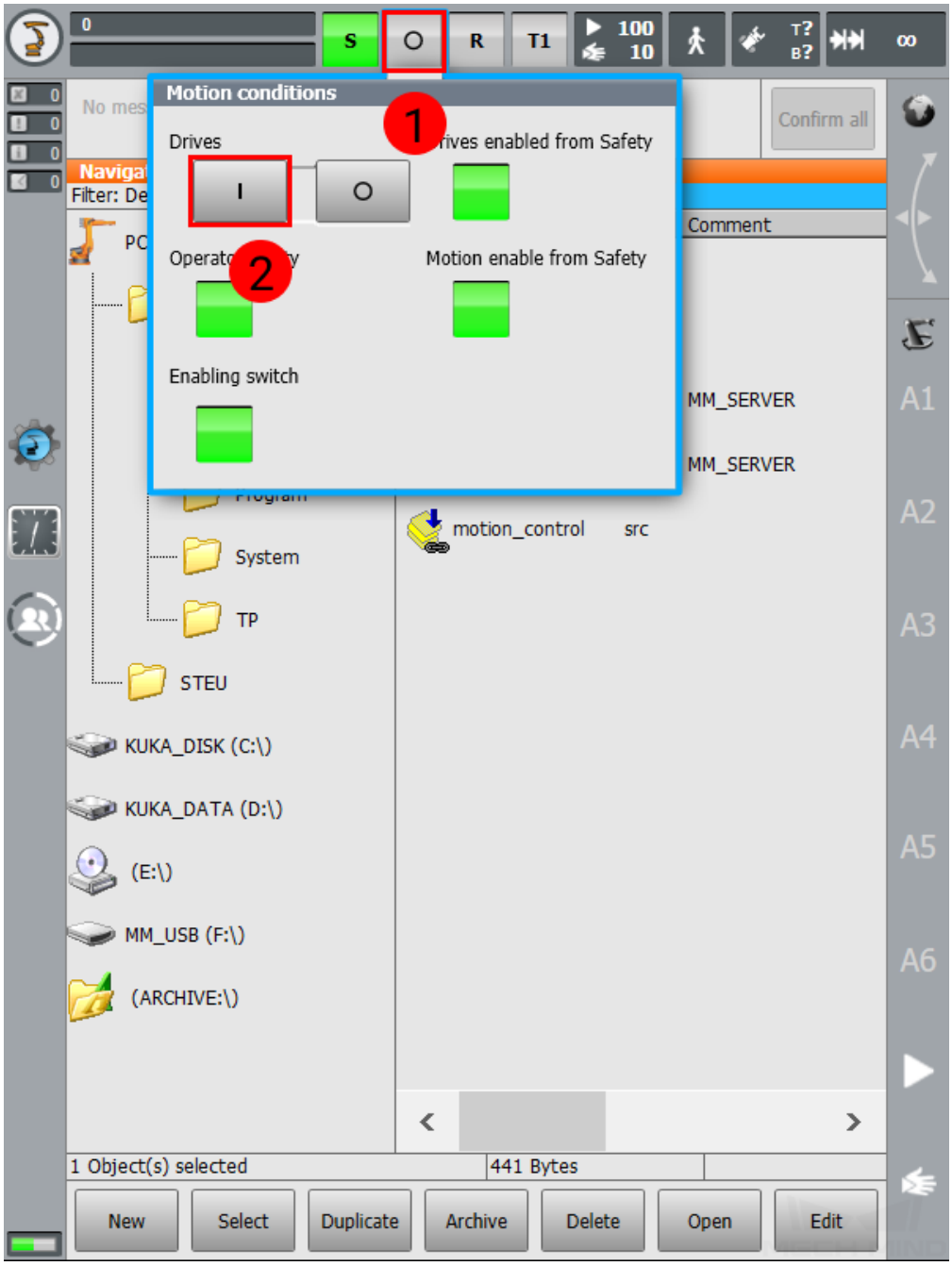
2. The following should appear on the screen.


/R1/MAINMODULE
S
I
R
T1
100
10
12:30:24 PM 5/11/2022 KSS01356
Start key required
?
OK
Confirm all
Editor
1 MECHMIND KUKA 4.1
2 ; WAIT FOR $IN_HOME
3 PTP HOME Vel=100 % DEFAULT
4 INIT()
5 WHILE (TRUE)
6 MM_CONTROL(1)
7 ENDWHILE
8
A1
A2
A3
A4
A5
A6
BP 3
KRC:\R1\MECHMIND\MAINMODULE.SRC
Ln 3, Col 0
Change
Commands
Motion
Open/close fold
Block selection
Touch-up
Edit

Run Program in AUT Mode

1. Turn the key switch to horizontal, select **T1** on the screen, and then turn the switch back to vertical.

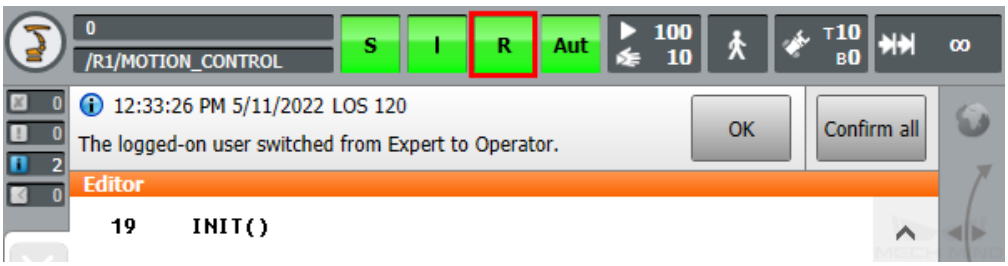
2. Check the icon to the right of  :
   - If it looks like  , then skip this step.
   - If it looks like  , then press on it and select  in the drop-down window.



3. Press on the enabling switch (either one of three) on the back of the pendant and  on the front at the same time to move the robot back to Home position. When the screen displays a message saying **Programmed path reached (BCO)**, and  turns red, release the enabling switch and  .

Note: Set an appropriate speed for the robot before moving it, and observe its motion carefully to avoid accidents.

4. Switch to **AUT** mode as described in step 10, and press on  to start running the full-control program ( should turn green).



Test Robot Connection

Please refer to *Test Robot Connection* for detailed instructions.

1.4.2 KUKA Program Description

Full-Control Programs

Program	Description
mm_server.sub	Background program for receiving data from Mech-Center and sending robot pose, signal and status data
mm_server.dat	Data file for the background program
motion_control.src	Foreground program for controlling and moving the robot
mainmodule.src	Foreground main module of the full-control program
mainmodule.dat	Data file for the main module
mm_status.xml	Configuration file for communicating robot status
mm_motion.xml	Configuration file for communicating robot motion

Internal Flags

Internal flag	Description
$FLAG[1]	Flag indicating that mm_motion has successfully connected
$FLAG[2]	Flag indicating that mm_motion has received data
$FLAG[5]	Flag indicating that mm_status has successfully connected
$FLAG[6]	Flag indicating that mm_status has received data

IOs

IO occupied	Signal
DI (16)	$IN[1]–$IN[16]
DO (16)	$OUT[1]–$OUT[16]
DI (64)	$IN[1]–$IN[64]
DO (64)	$OUT[1]–$OUT[64]

1.5 Kawasaki

This section introduces the full-control program for Kawasaki robots and the procedure of setting up the communication with a robot through the program.

1.5.1 Kawasaki Setup Instructions

This section introduces the process of loading the robot full-control program onto a Kawasaki robot.

The process consists of 4 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program File*
- *Test Robot Connection*

Please have a flash drive ready at hand.

Note: A USB 2.0 flash drive is recommended. Otherwise, the robot controller may not recognize the flash drive.

Check Controller and Software Compatibility

- Controller: no requirement
- Controller system software version: no requirement
- Additional controller software options: not required
- Mech-Center: latest version recommended

Setup the Network Connection

Hardware Connection

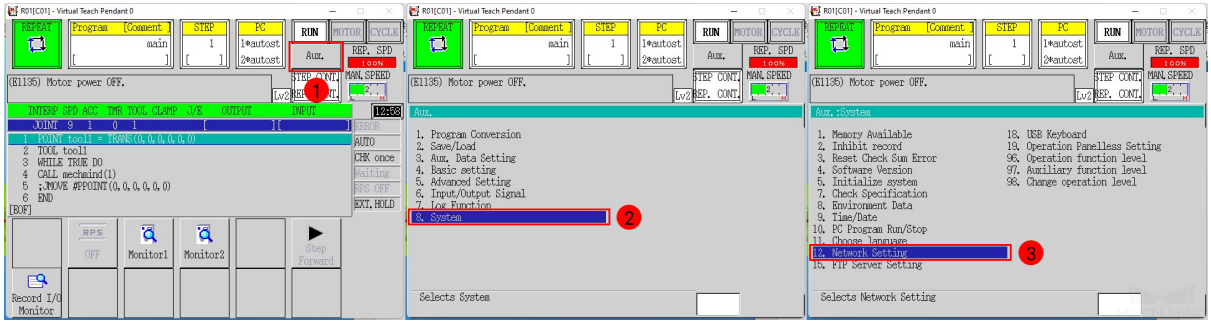
Plug the Ethernet cable into:

- An Ethernet port on the IPC
- The Ethernet port inside the accessory panel on the front of the controller

IP Configuration

To allow communication between the IPC and the robot controller, both must have an IP address in the same subnet. This means that the first three numbers of the IP addresses should be the same. For example, 192.168.100.1 and 192.168.100.2 are in the same subnet.

1. Check the IP address of the IPC: please use the *ipconfig* command in Command Prompt or PowerShell to check the IP address.
2. On the teach pendant, press on *Aux.*, and then select *8. System* → *12. Network Setting*.

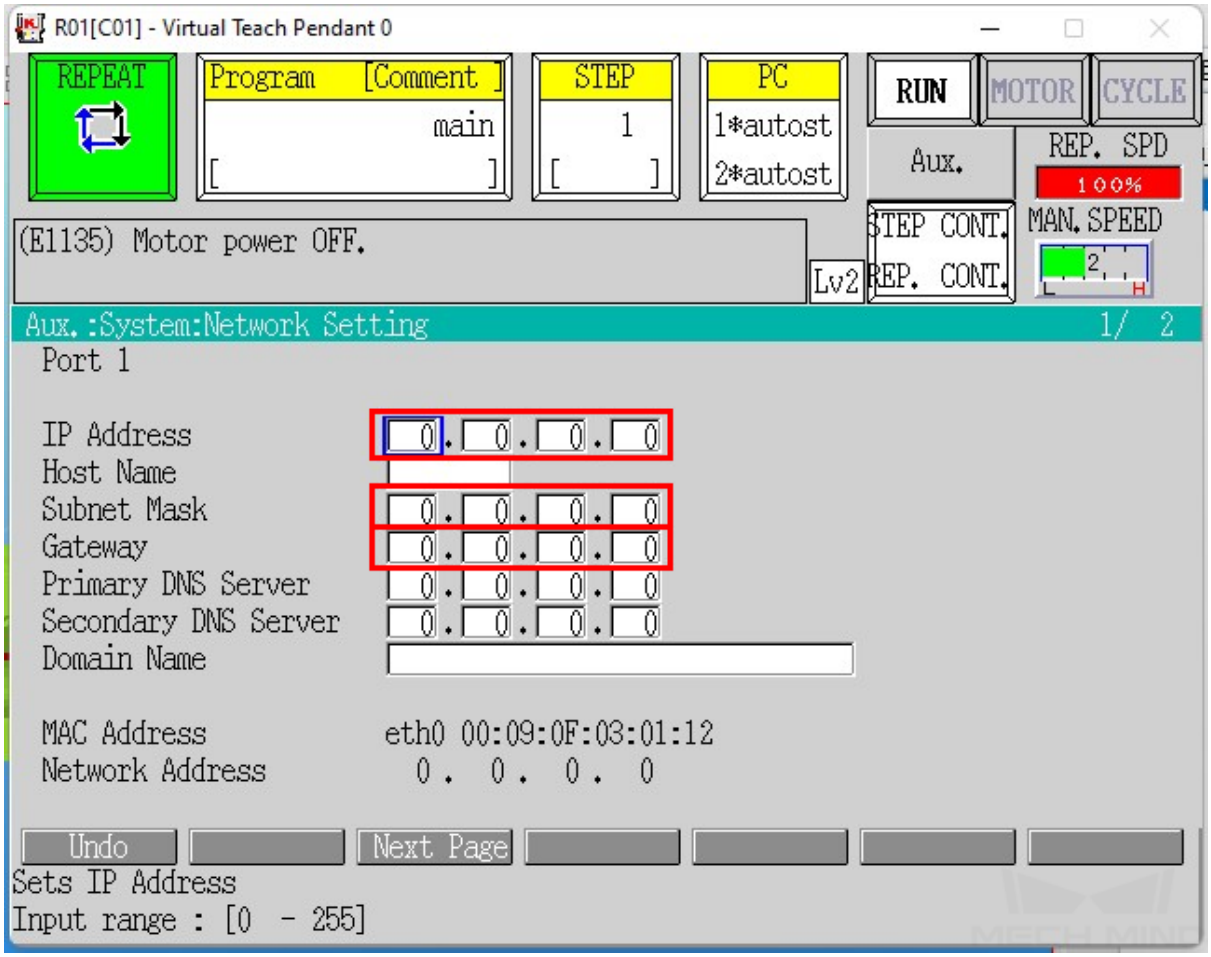


3. Set **IP Address** to one in the same subnet as that of the IPC.
4. Set **Subnet Mask** to **255.255.255.0**.

Note: If the IP address is set to either 192.168.0.1 or 192.168.1.1, please set **Subnet Mask** to **255.255.0.0** instead.

---

5. If you are using a network gateway, the gateway address should also be set. Please consult your IT support for help.



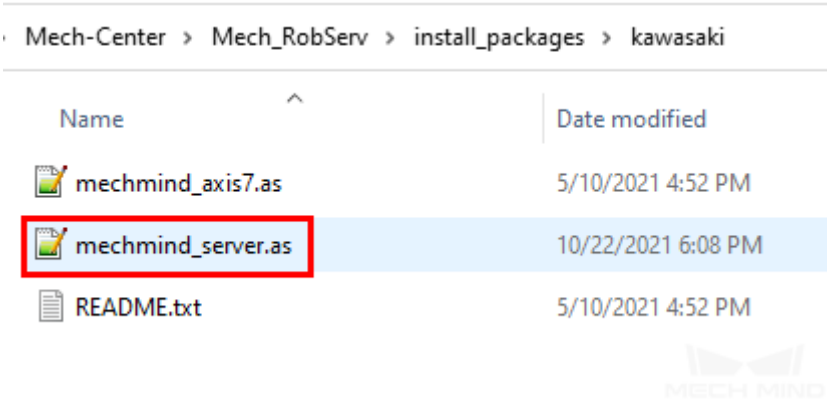
6. Press the `ENTER` key to confirm, and then restart the controller.

Load the Program File

Prepare the File

The program file is stored in the installation directory of Mech-Center. The default directory for Mech-Center 1.5.2 is *C:/Mech-Mind/Mech-Center*.

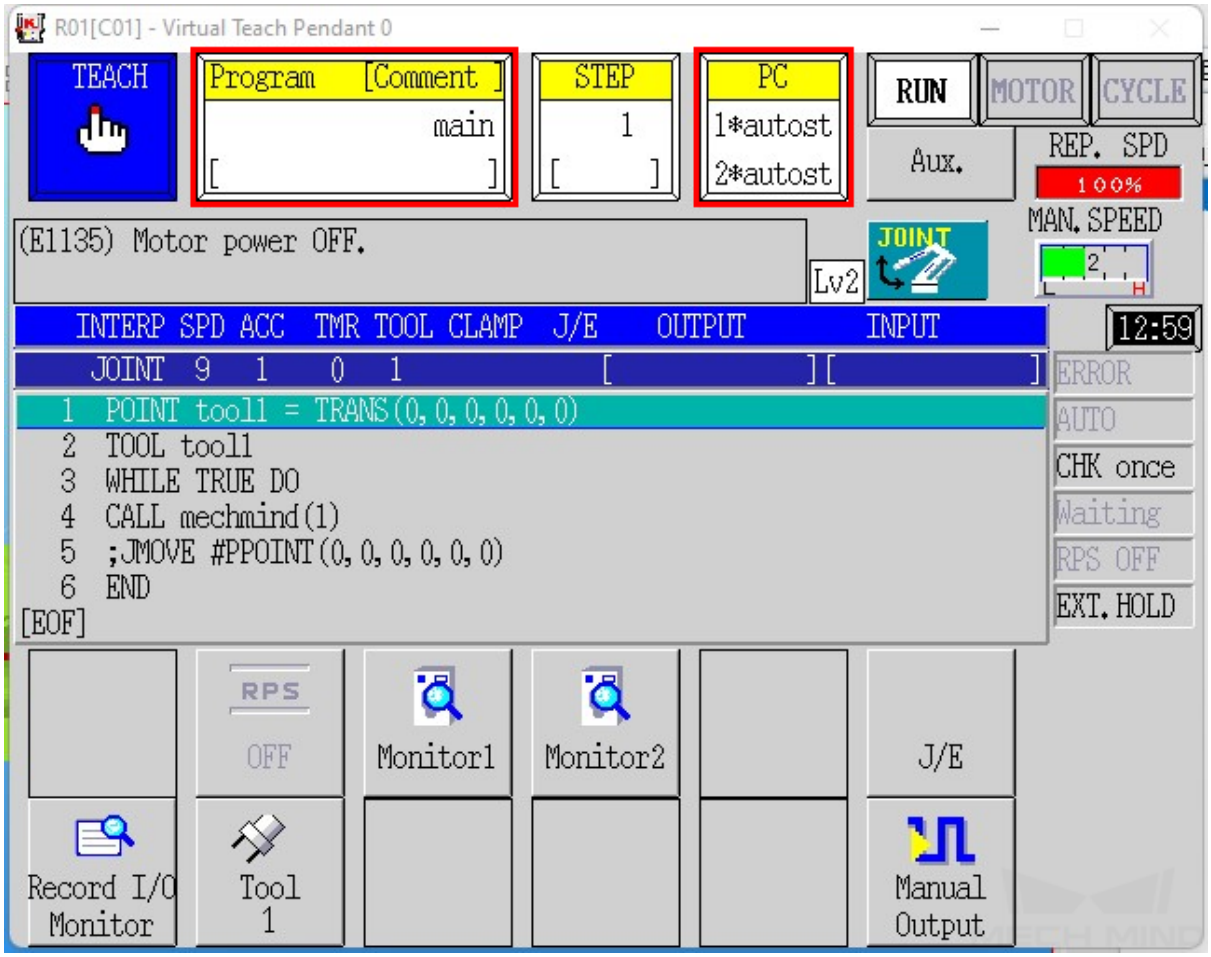
Navigate to *xxx/Mech-Center/Mech-RobServ/install_packages/kawasaki*, and copy **mech-mind_server.as** to your flash drive.



Note: Copy the file to the root directory of the flash drive. Do not put it in another folder or rename it.

Load the File to the Robot

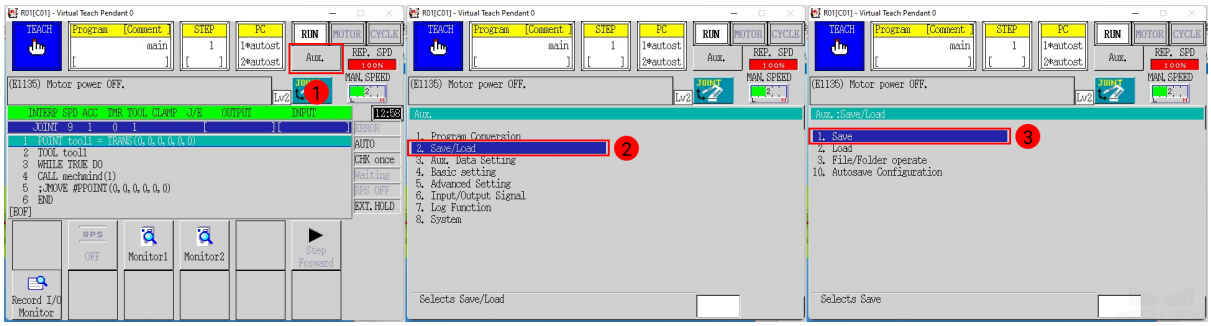
1. Insert the flash drive to the USB port inside the accessory panel on the controller.
2. Check the **Program** and **PC** areas to see if any programs are running. If so, backup and exit the programs according to the following instructions.



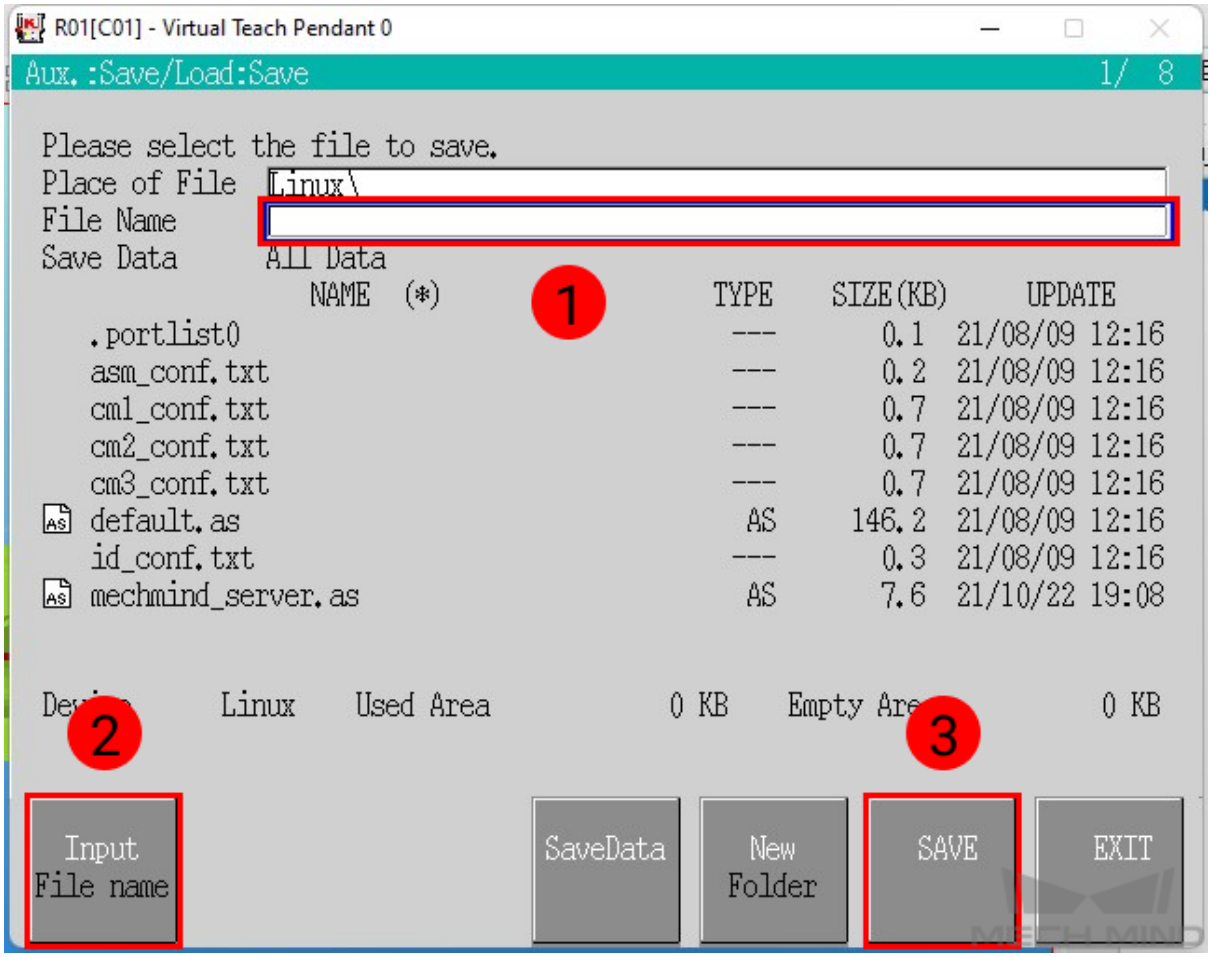
If there is a program in the **Program** area, please create backup first.

Follow the steps below to back up all files except for system logs to the flash drive.

1. Press on *Aux.*, and select *2. Save/Load* → *1. Save*.

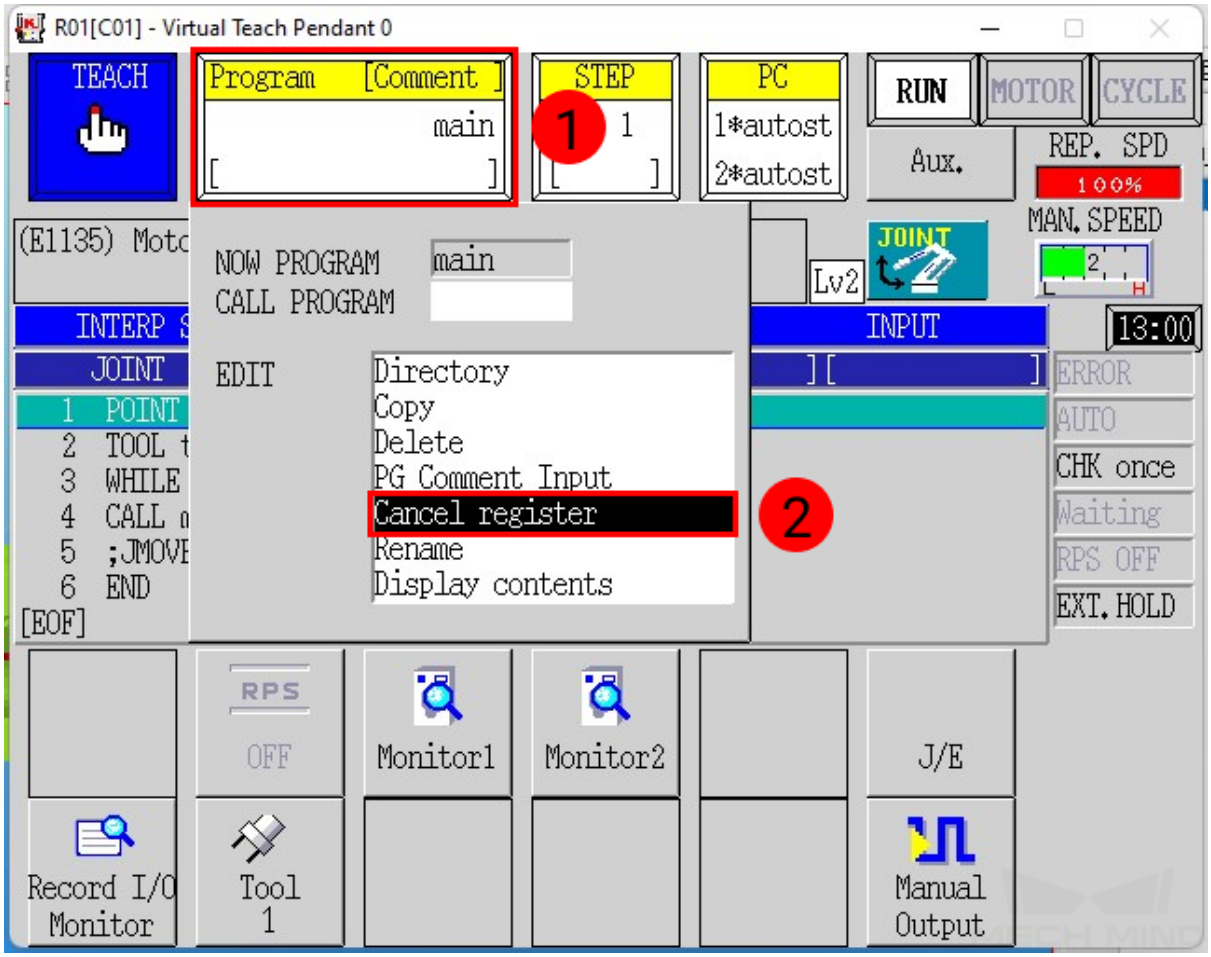


2. Press on *Input File name* to input a **File Name** for the backup file, and then press on *SAVE*.

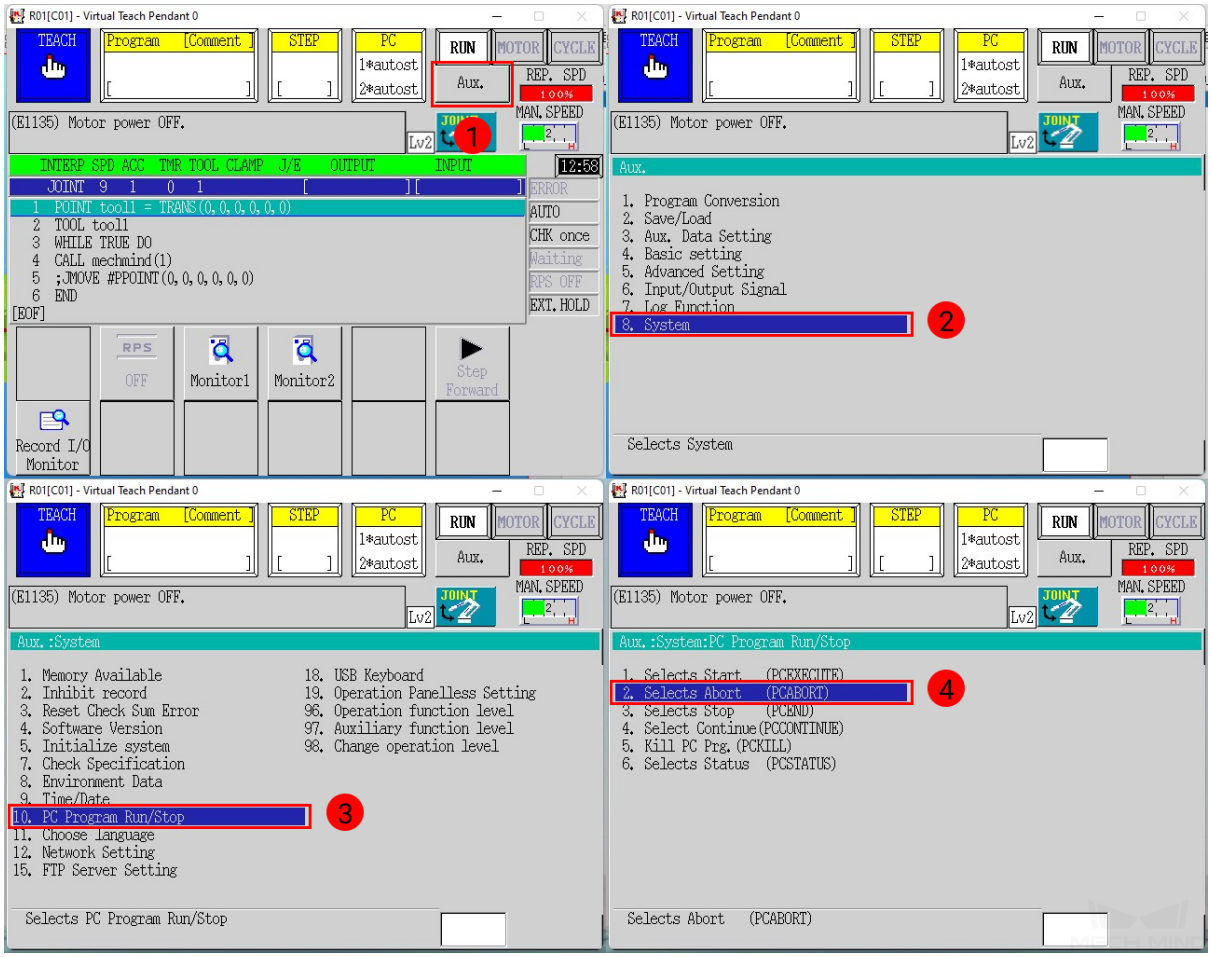


Cancel robot control program and kill PC programs.

1. Press on the **Program** area, and select **Cancel register** in the drop down menu.

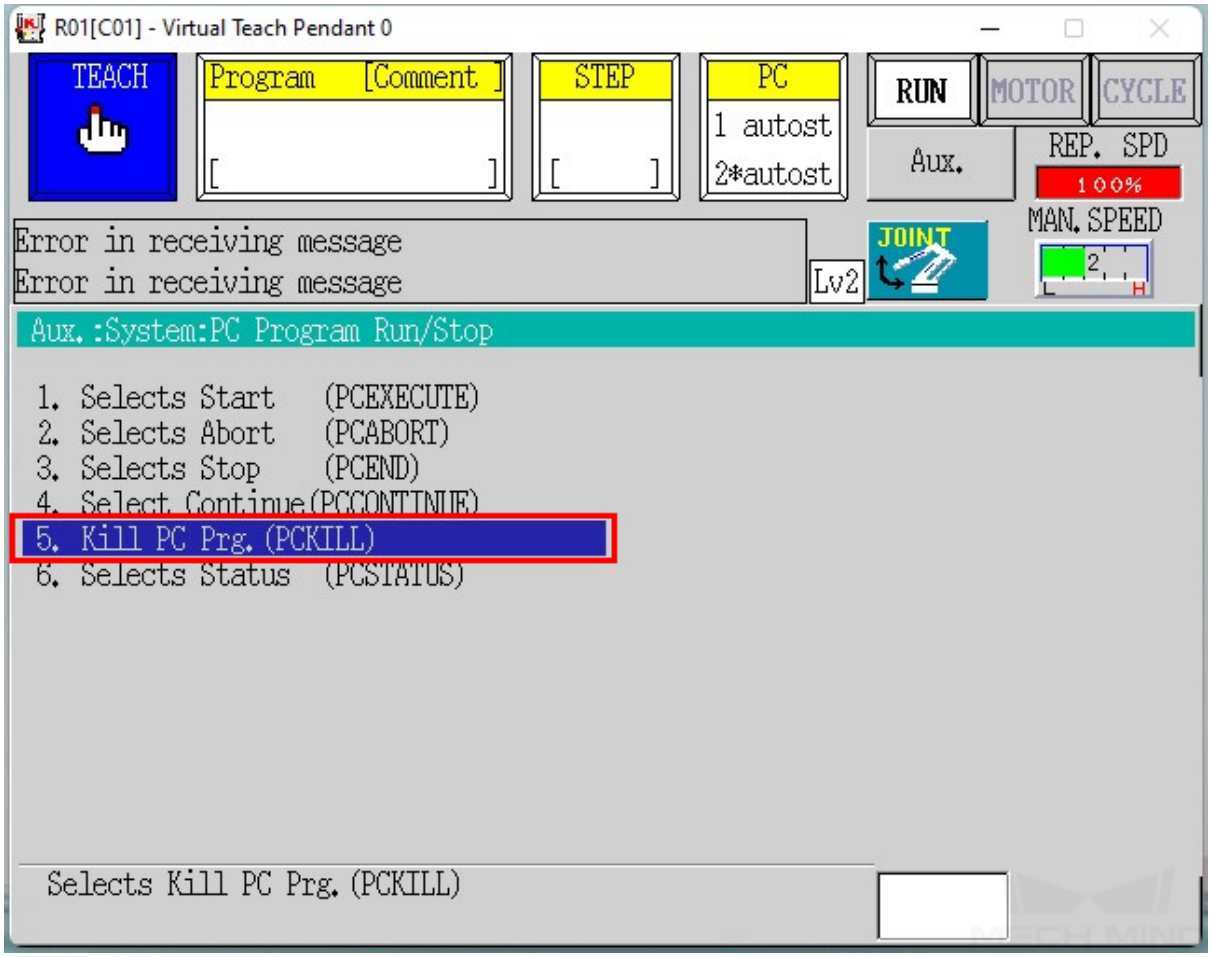


2. Press on *Aux.*, and select *8. System* → *10. PC Program Run/Stop* → *Selects Abort.*



3. Press the ENTER key to abort PC program 1. Then, press 2 to change the program number, and then press ENTER to abort PC program 2.

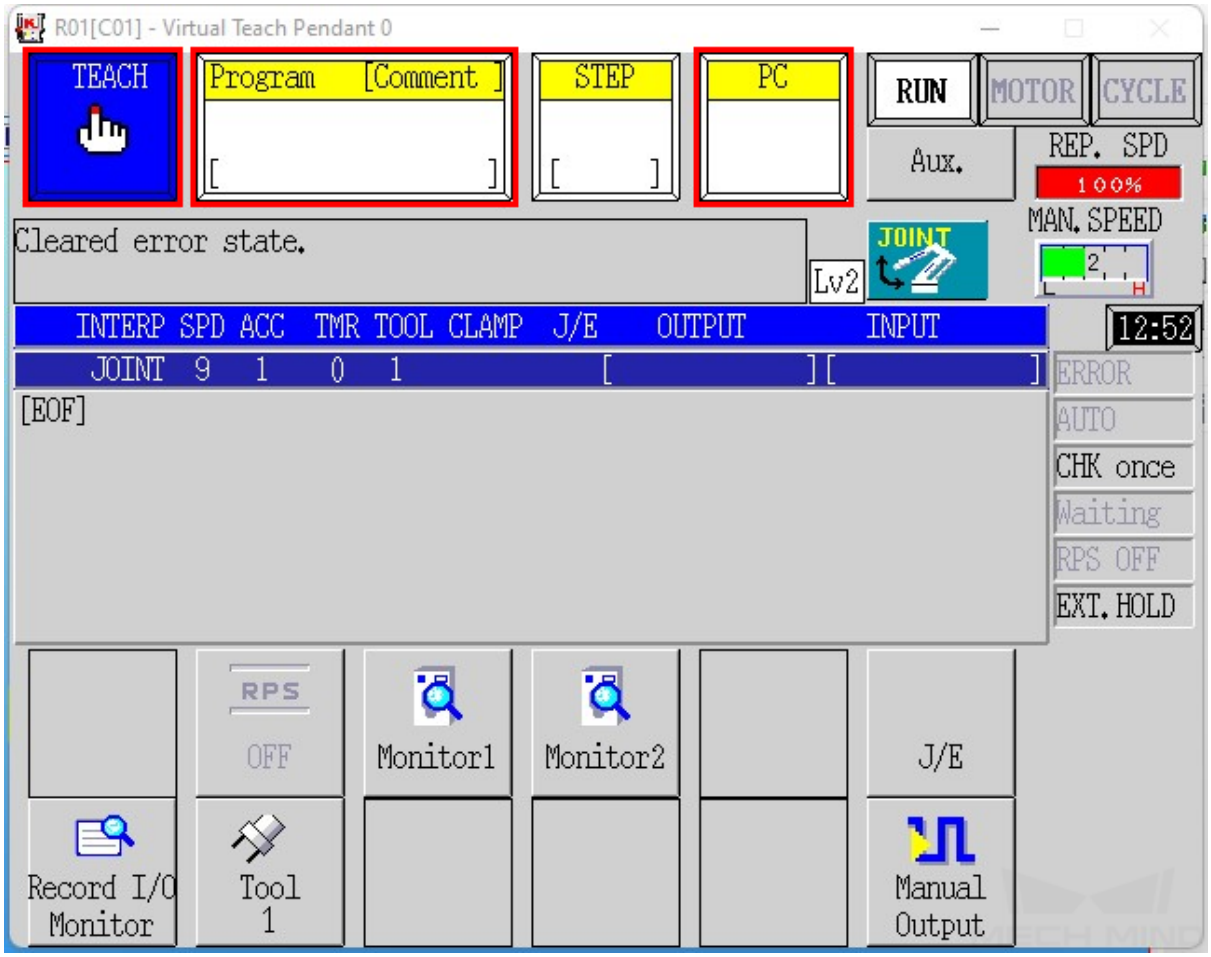
4. Press the R key to return, and select **Kill PC Prog.**.



5. Press the `ENTER` key to kill PC program 1. Then, press `2` to change the program number, and then press `ENTER` to kill PC program 2.

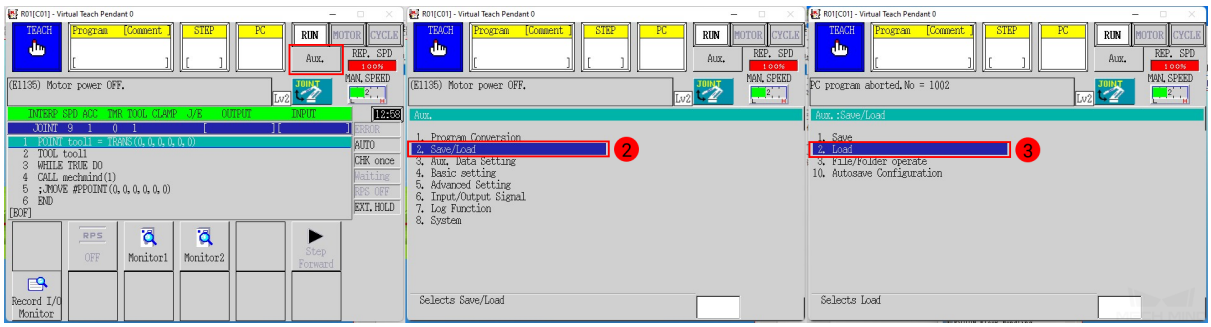
6. Press the R key to return. Check the **PC** area to see if any programs are still listed. If so, repeat steps 2 and 4 until there are no programs listed in the **PC** area.

3. Make sure that the robot is in teach mode, and make sure that the **Program** and **PC** areas have nothing listed.

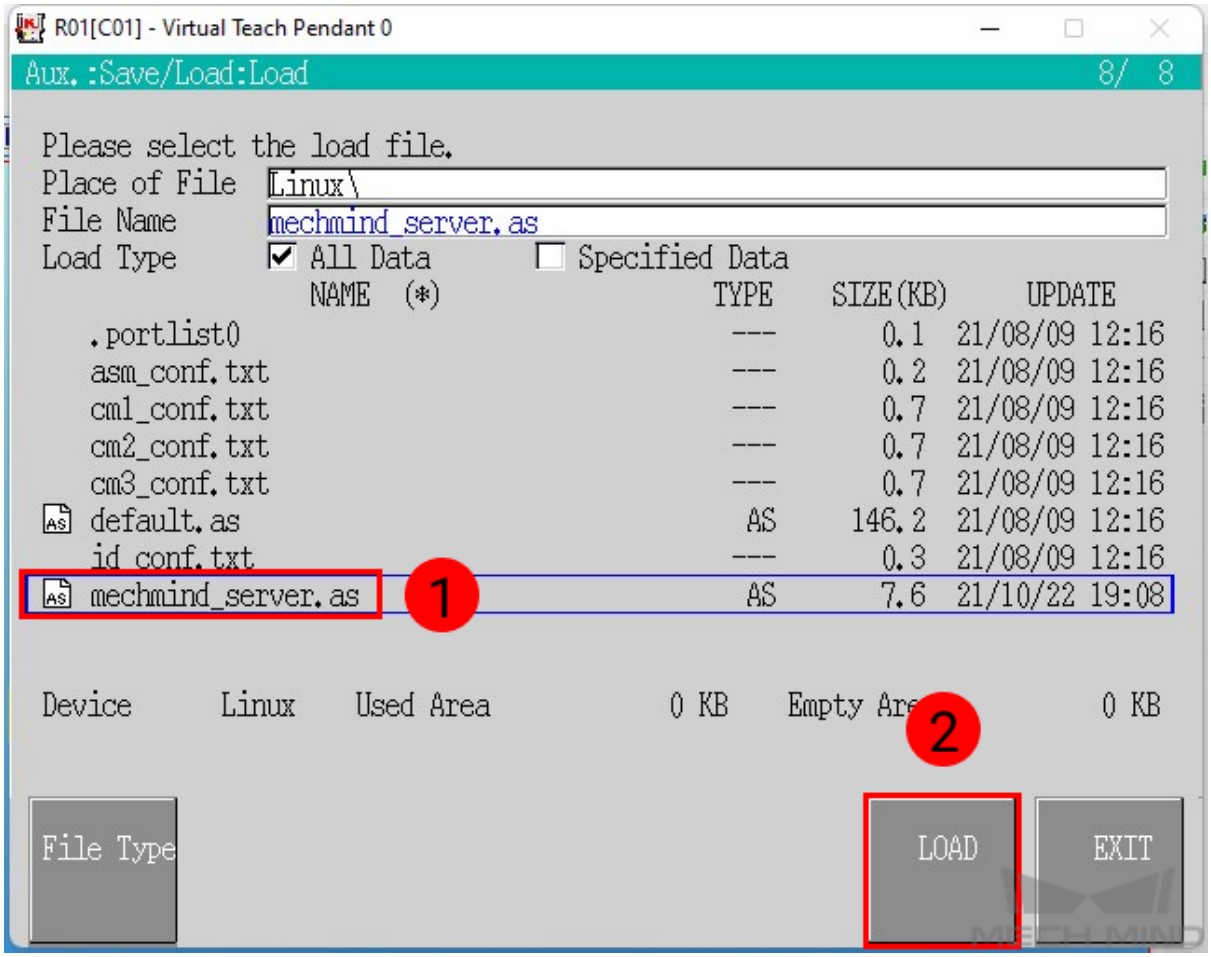


Note: To switch the robot to teach mode, turn the Teach/Repeat switch on the controller to **TEACH**, and the teach lock switch on the teach pendant to **ON**.

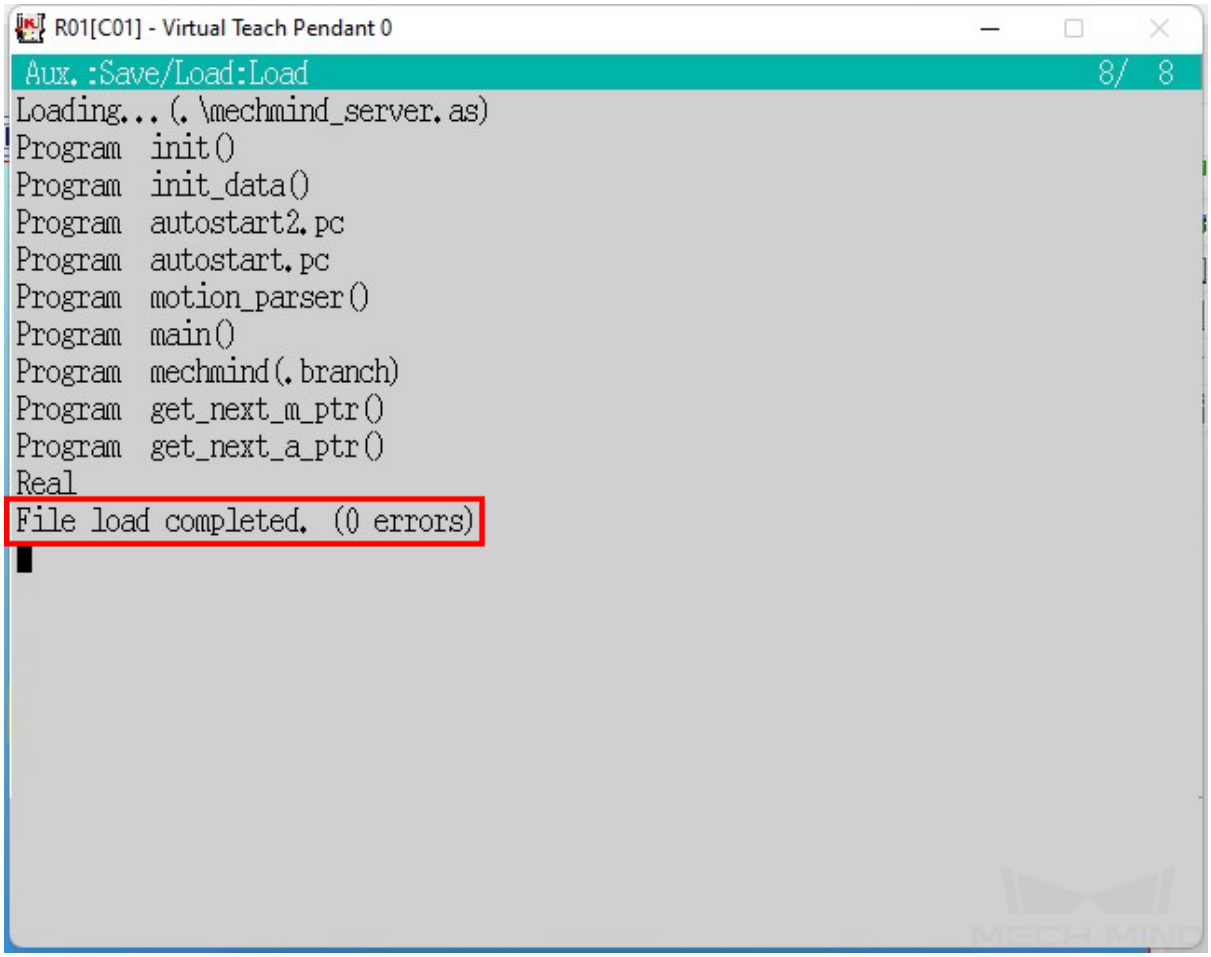
4. Press on *Aux.*, and select *2. Save/Load* → *2. Load*.



5. Press on **mechmind_server.as** in the file list twice to select it, and then press on *LOAD*.

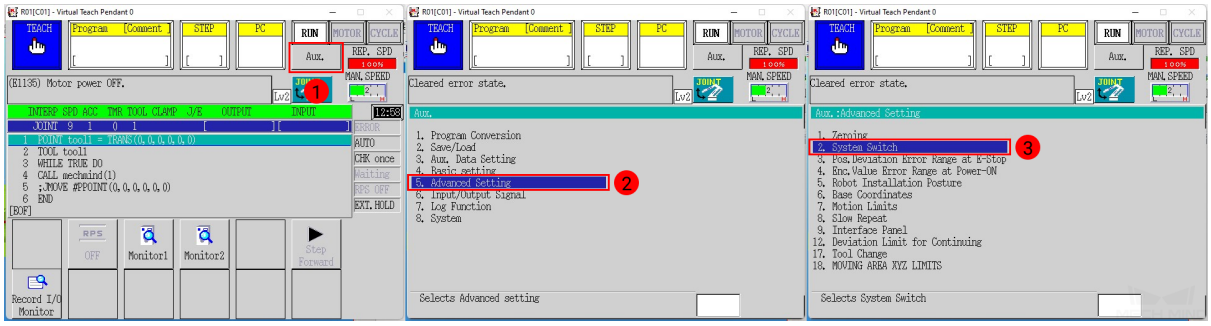


6. When loading completes, make sure no errors occurred during loading, and press on the R key to exit.

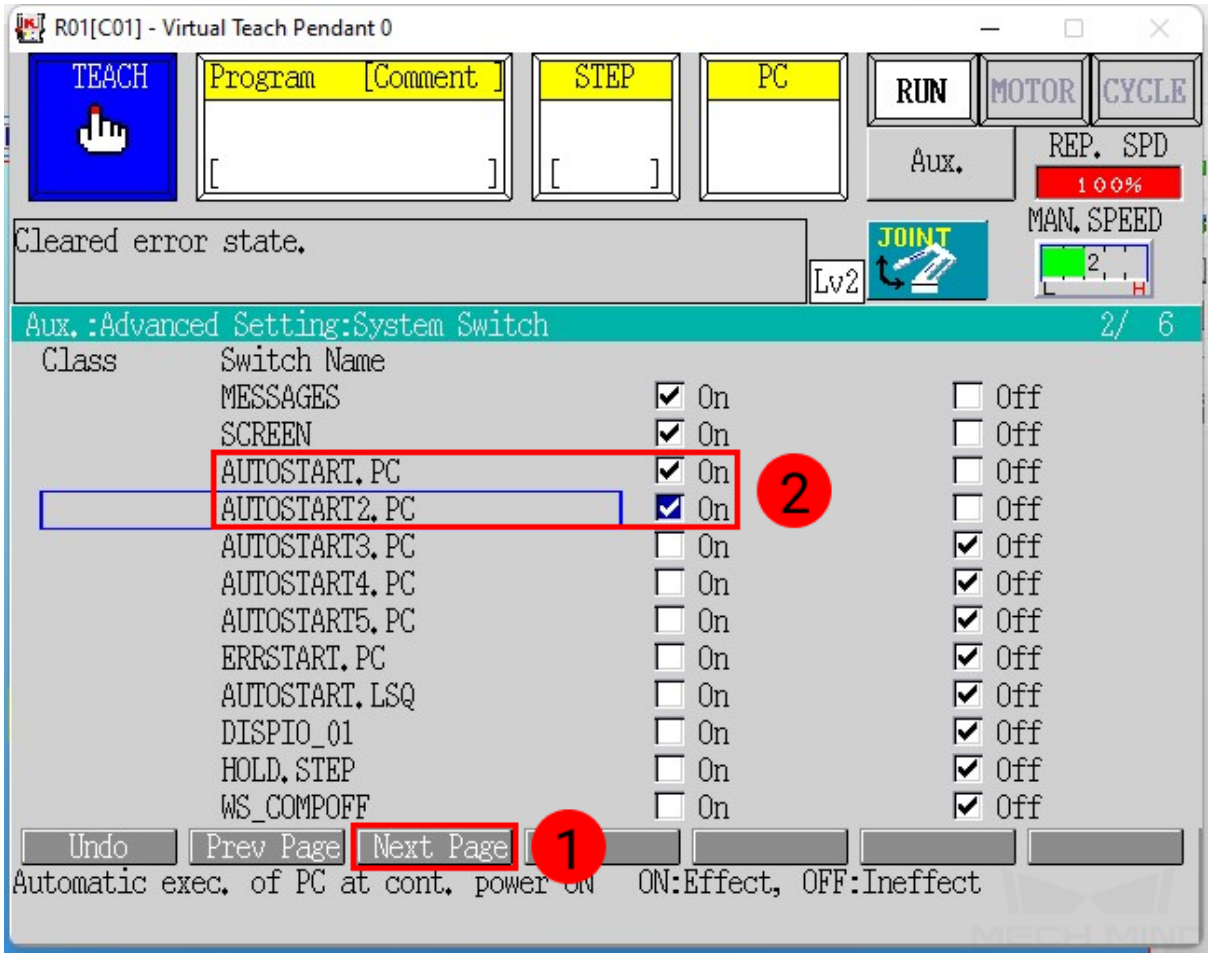


Further Configurations

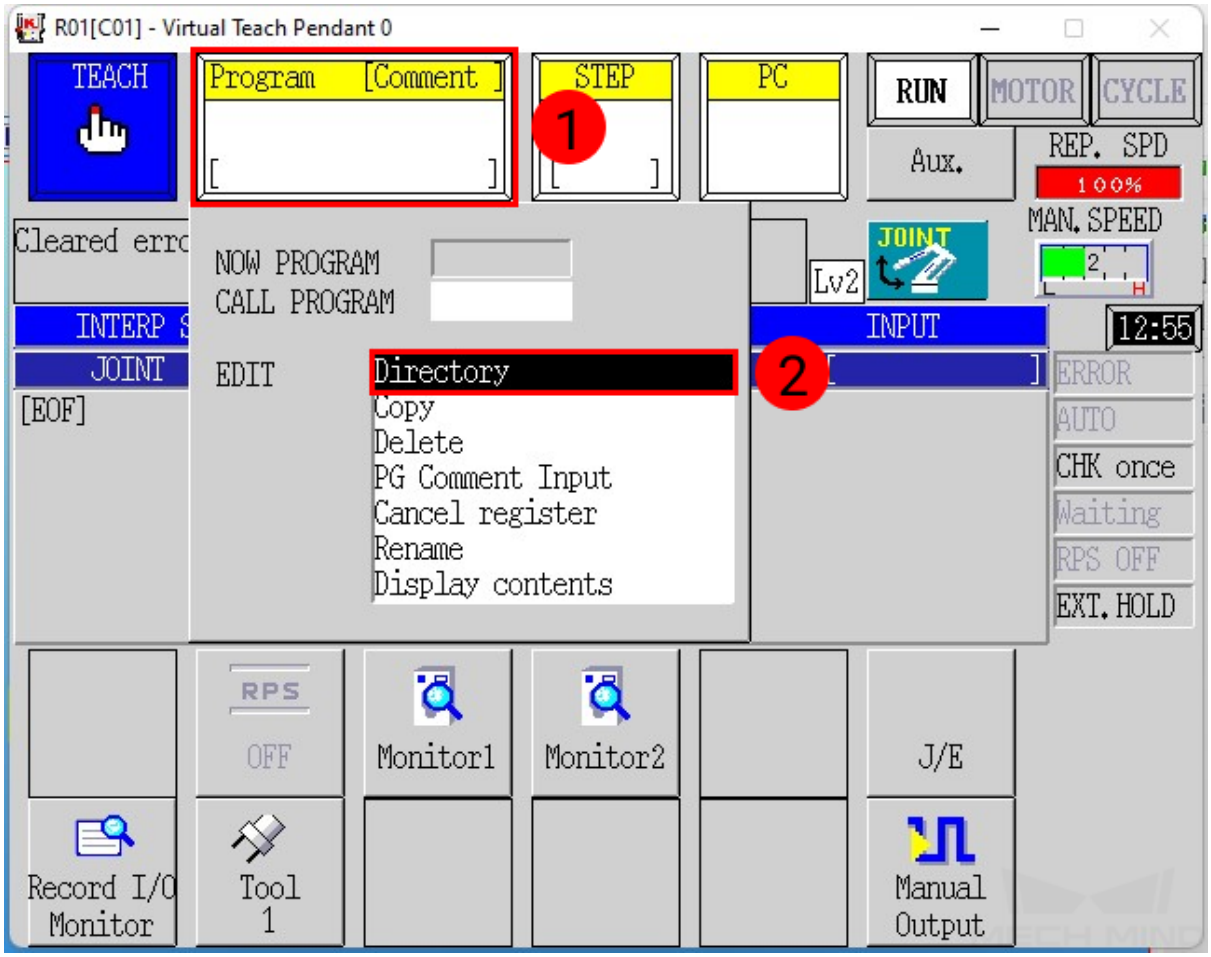
1. Press on *Aux.*, and select *5. Advanced Settings* → *2. System Switch.*



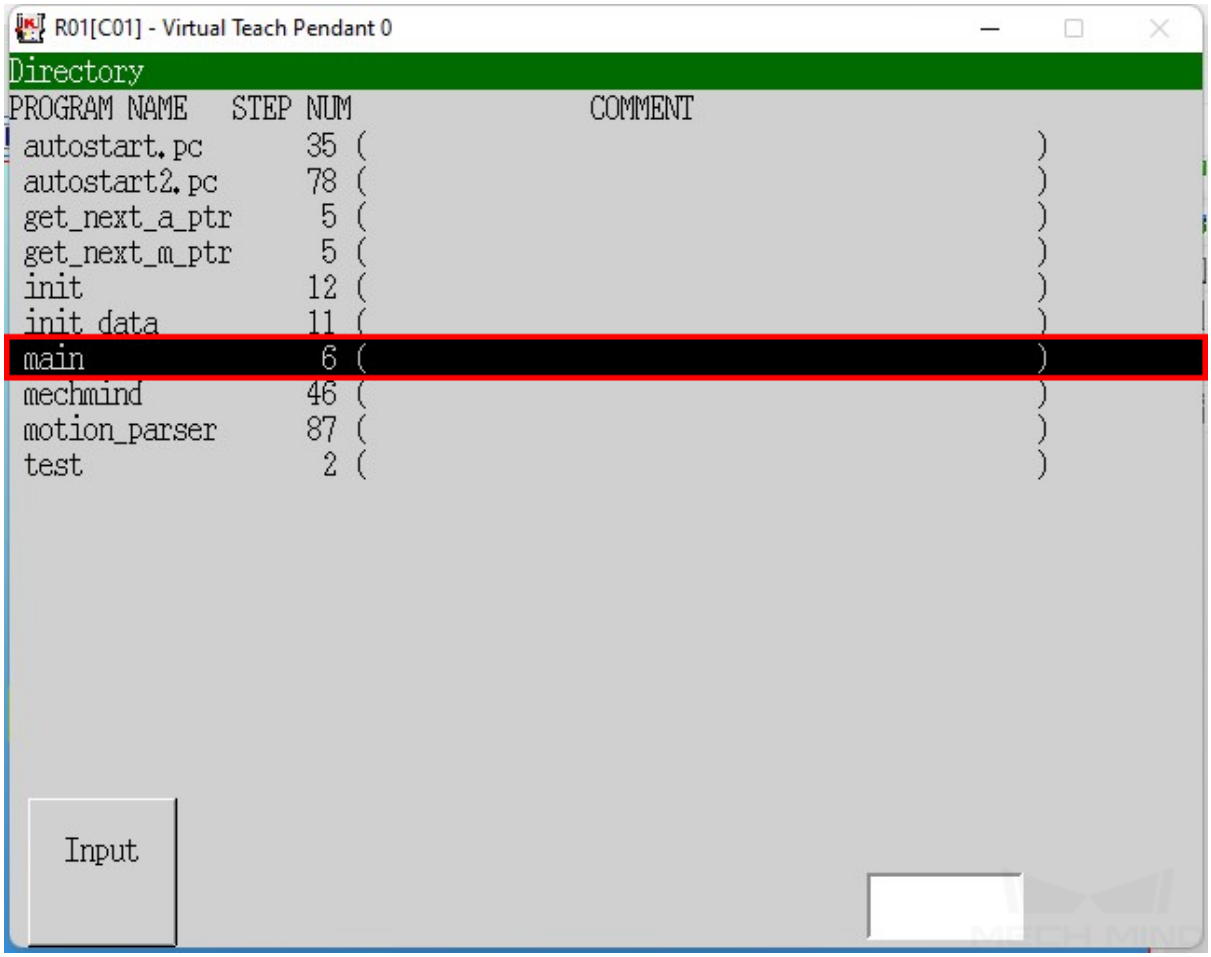
2. Press on *Next Page*, find **AUTOSTART.PC** and **AUTOSTART2.PC**, check the **On** boxes of these two programs, and press the `ENTER` key to save the changes.



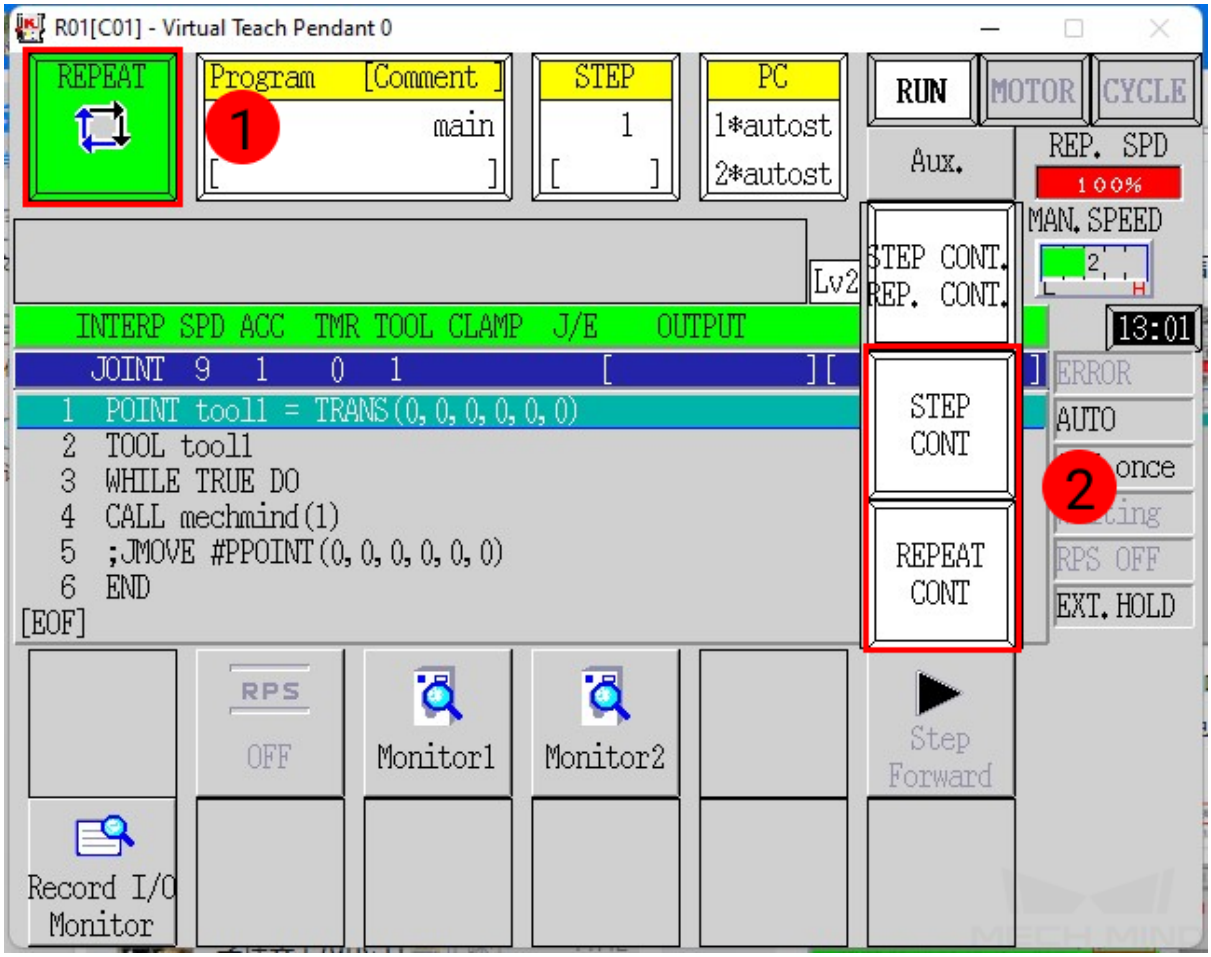
3. Press on the **Program** area, select **Directory**.



4. Select **main**, and press the `ENTER` key to save the change.

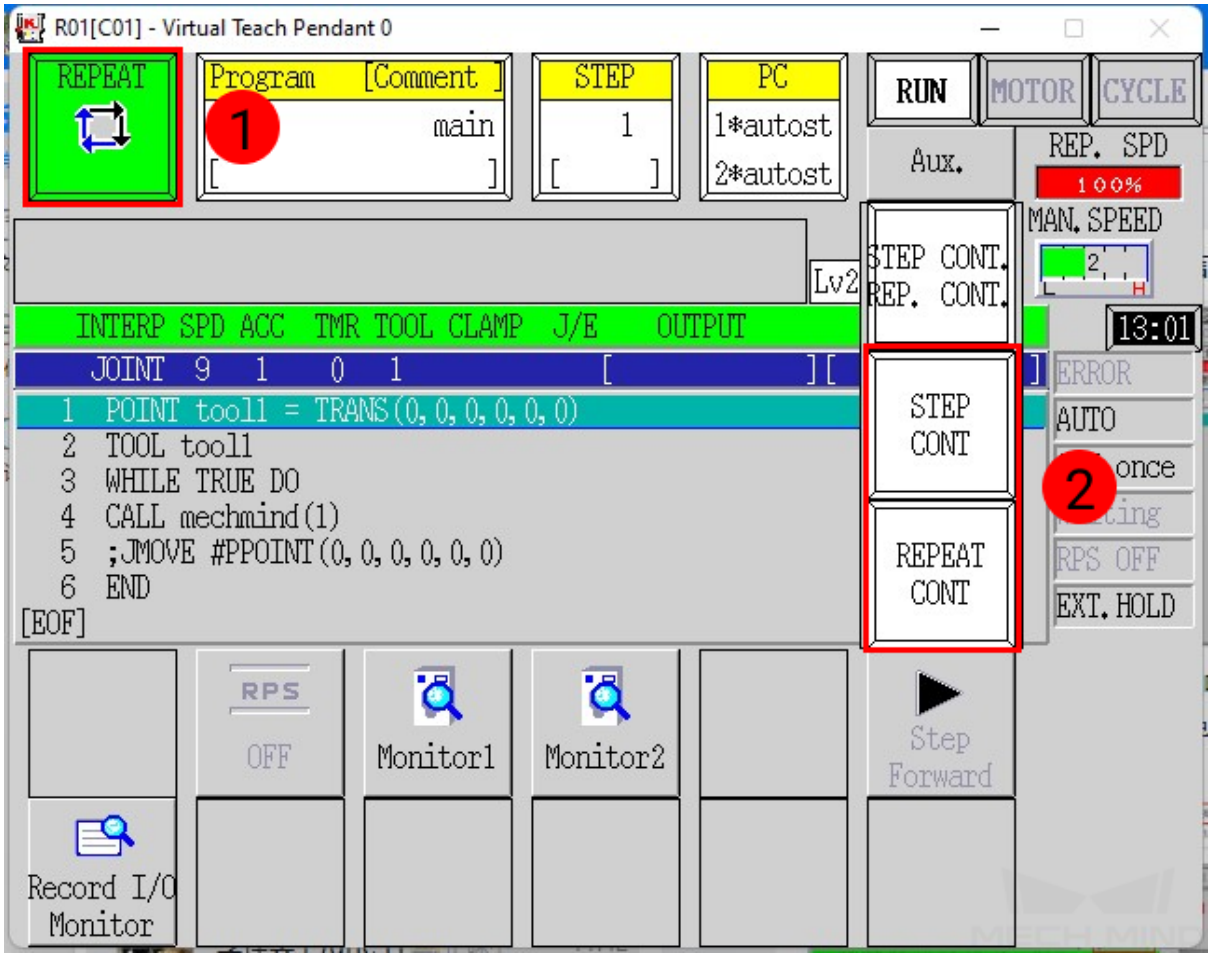


5. Switch the robot to repeat mode, press on the white button below *Aux.*, and change the drop-down options to **STEP CONT** and **REPEAT CONT**.



Note: To switch the robot to repeat mode, turn the Teach/Repeat switch on the controller to **REPEAT**, and the teach lock switch on the teach pendant to **OFF**.

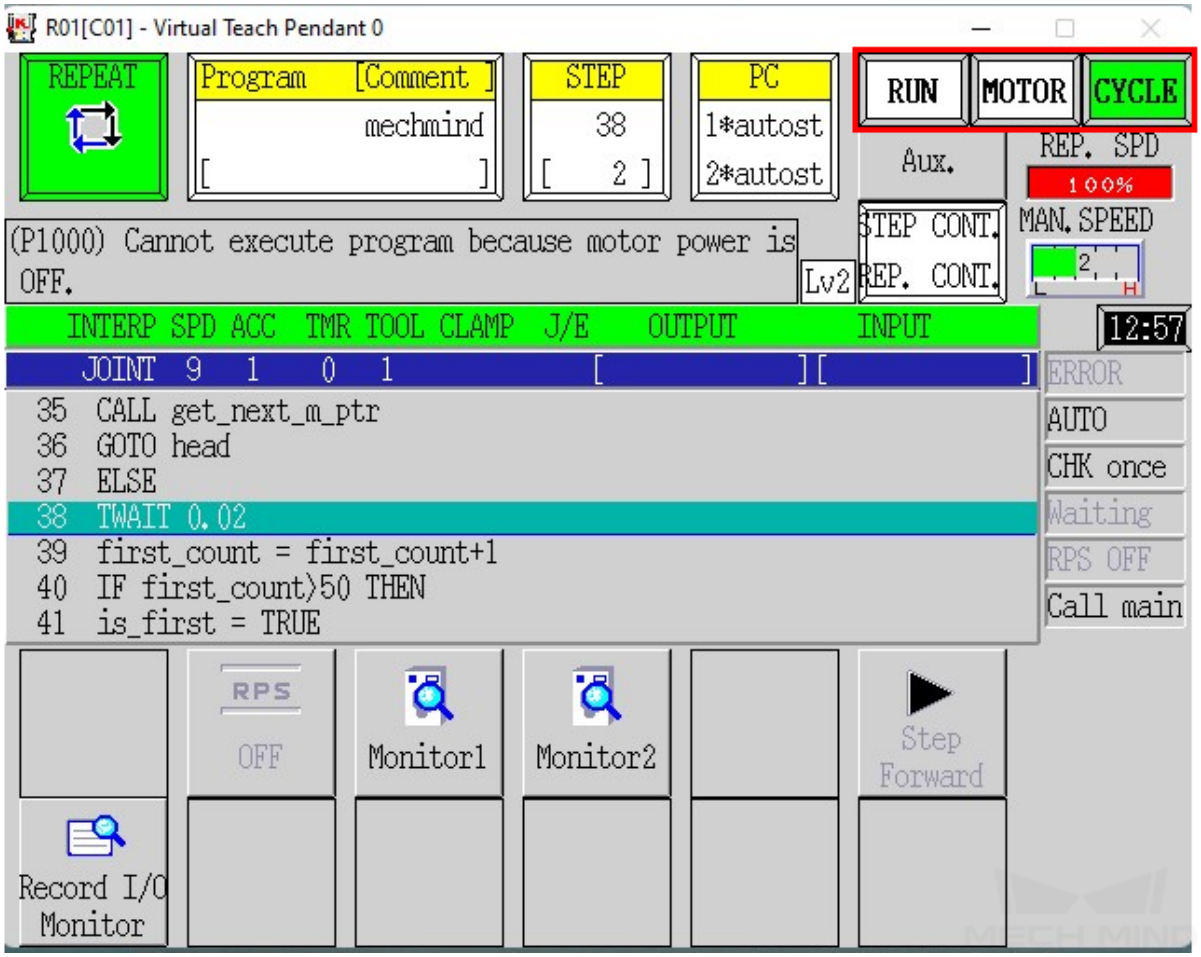
6. Restart the controller. Now the teach pendant should show the following interface, with two PC programs running (indicated by the asterisks after the program numbers) and **1** displayed in the **STEP** area.



Note: If the **STEP** isn't **1**, please press on the **STEP** area and change it to **1**.

Test Robot Connection

1. Please refer to *Test Robot Connection* for detailed instructions on connecting to the robot in Mech-Center.
2. Press on *MOTOR* while holding down the `A` key to power the motor.
3. Press on *CYCLE* while holding down the `A` key to run the program.
4. If *RUN* does not turn green, press the `RUN/HOLD` key while holding down the `A` key.



The robot is successfully connected if *MOTOR*, *CYCLE*, and *RUN* are lit green, and no error messages appear in Mech-Center or on the teach pendant.

1.5.2 Kawasaki Program Description

Full-Control Programs

Program	Description
init	Set IP and signals
init_data	Initialize variables
autostart.pc	PC program 1 for receiving commands
autostart2.pc	PC program 2 for sending robot pose and status
motion_parser	Subprogram called by PC program for decoding commands
main	Foreground main program
mechmind	The full-control program
get_next_m_ptr	Index control (motion)
get_next_a_ptr	Index control (storage)

Internal Signals

Name	Signal	Description
stop_sig	2011	Stop motion
viz_conl_sig	2012	Full-control signal

1.6 NACHI

This section introduces the process of loading the robot full-control program onto an NACHI robot.

The process consists of 4 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program Files*
- *Test Robot Connection*

Please have a flash drive ready at hand.

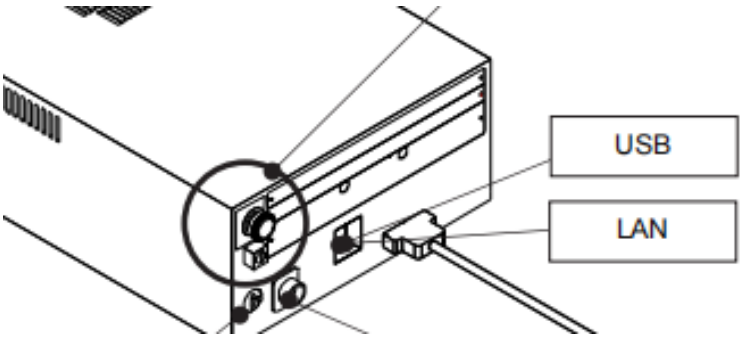
1.6.1 Check Controller and Software Compatibility

- There is no specific requirements on the version of robot controller.
- It is recommended to use Mech-Center of the lastest version.

1.6.2 Setup the Network Connection

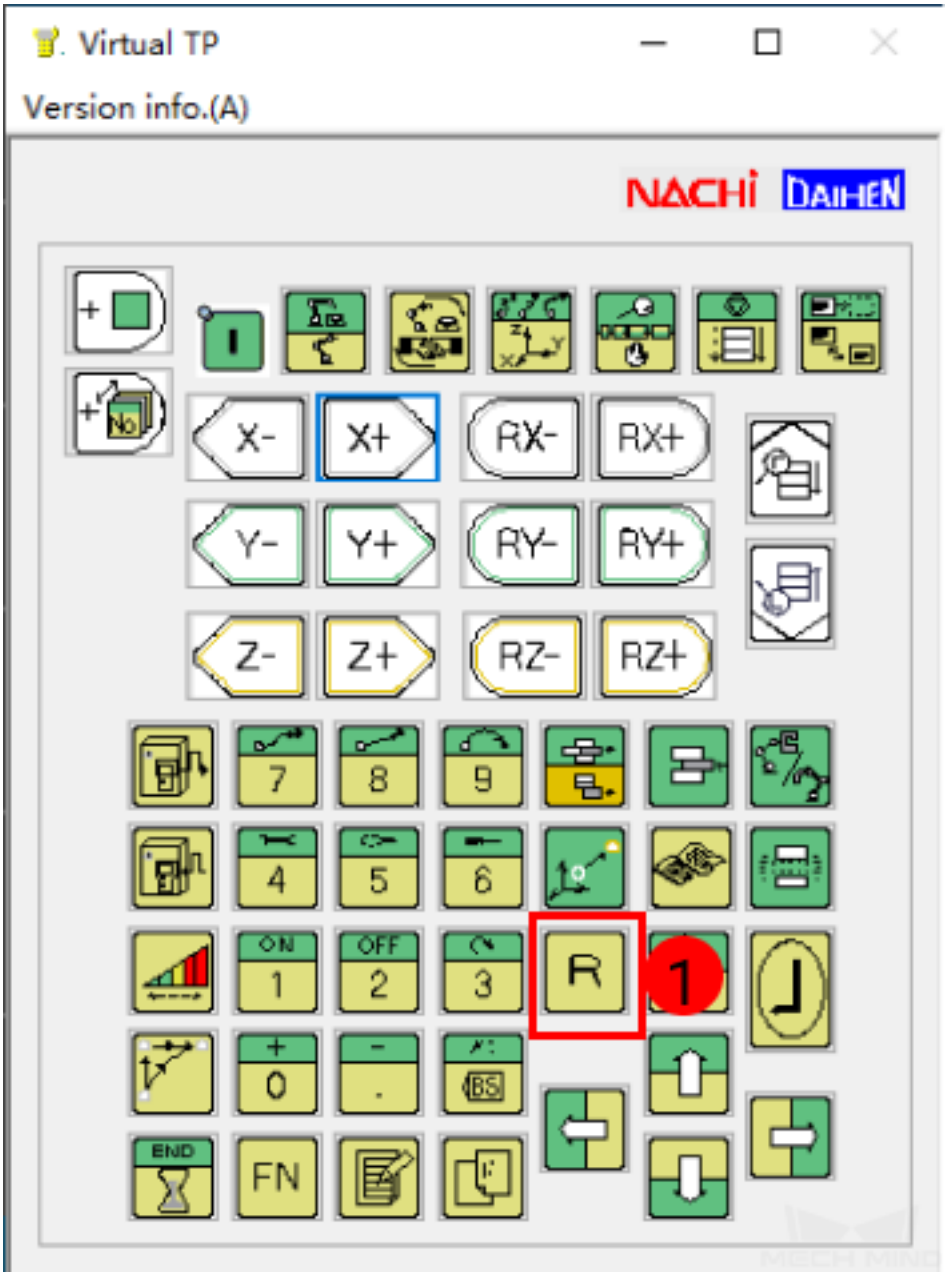
Hardware Connection

Plug the Ethernet cable into the LAN port of the robot controller to connect it with the IPC.

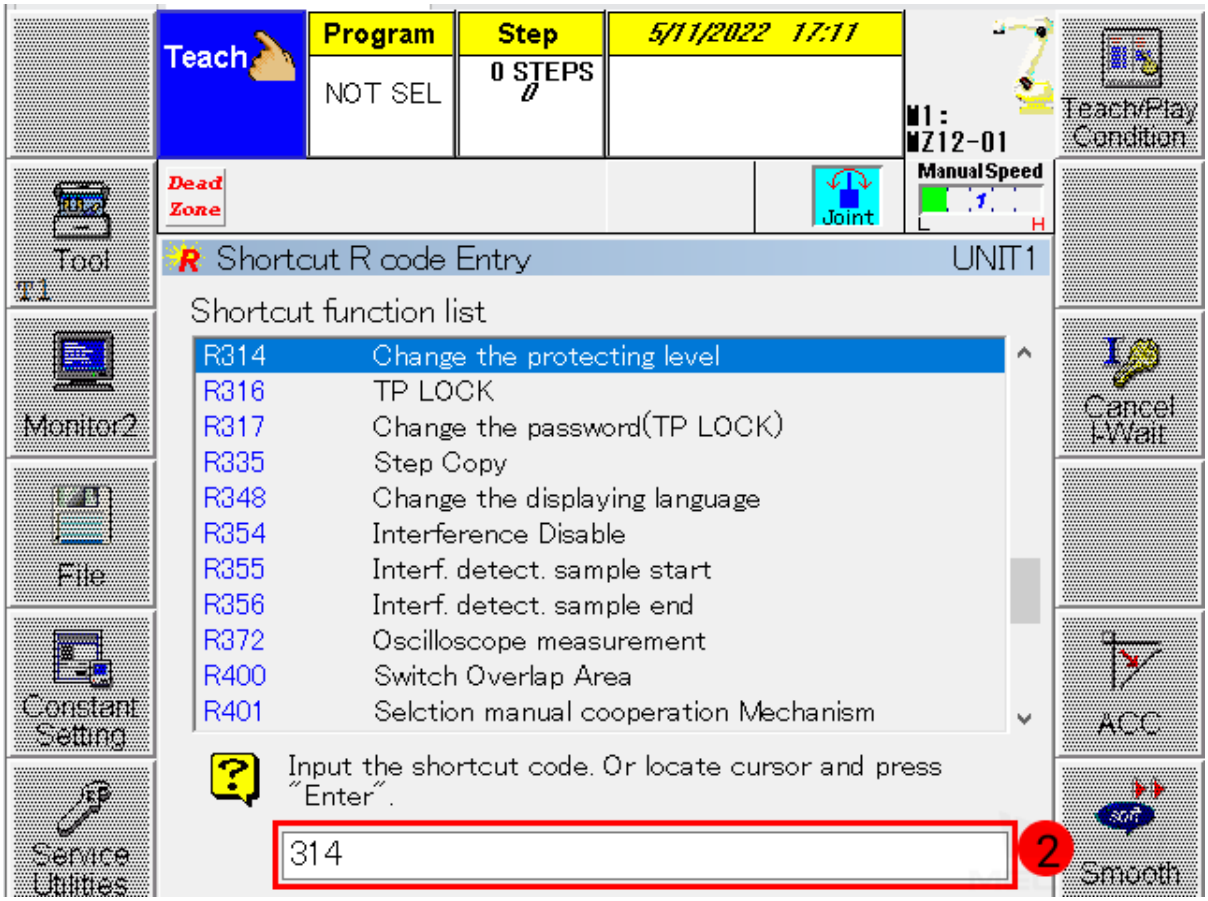


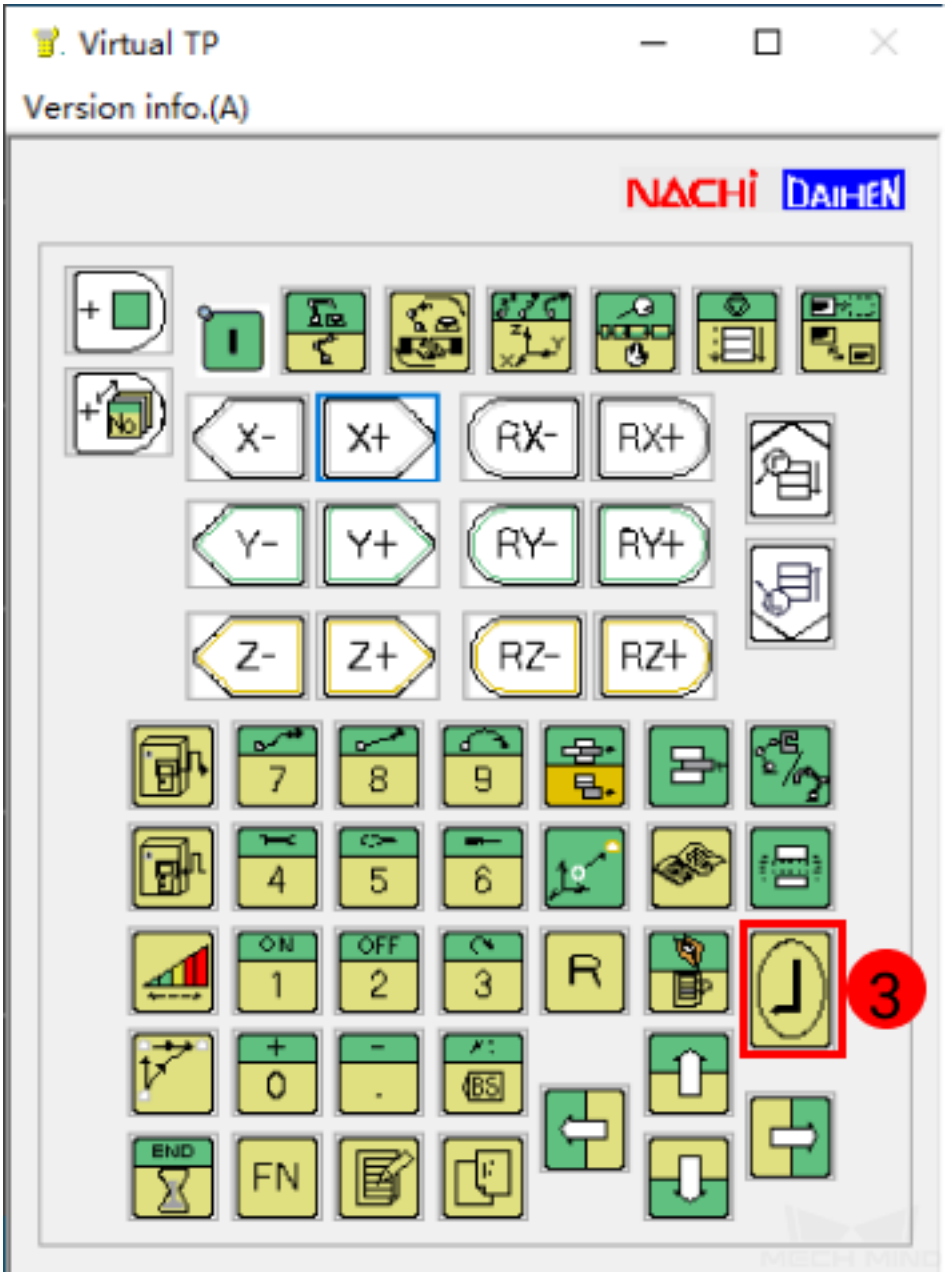
Change the Protecting Level

1. Open the teach pendant and press the  key, as shown below.

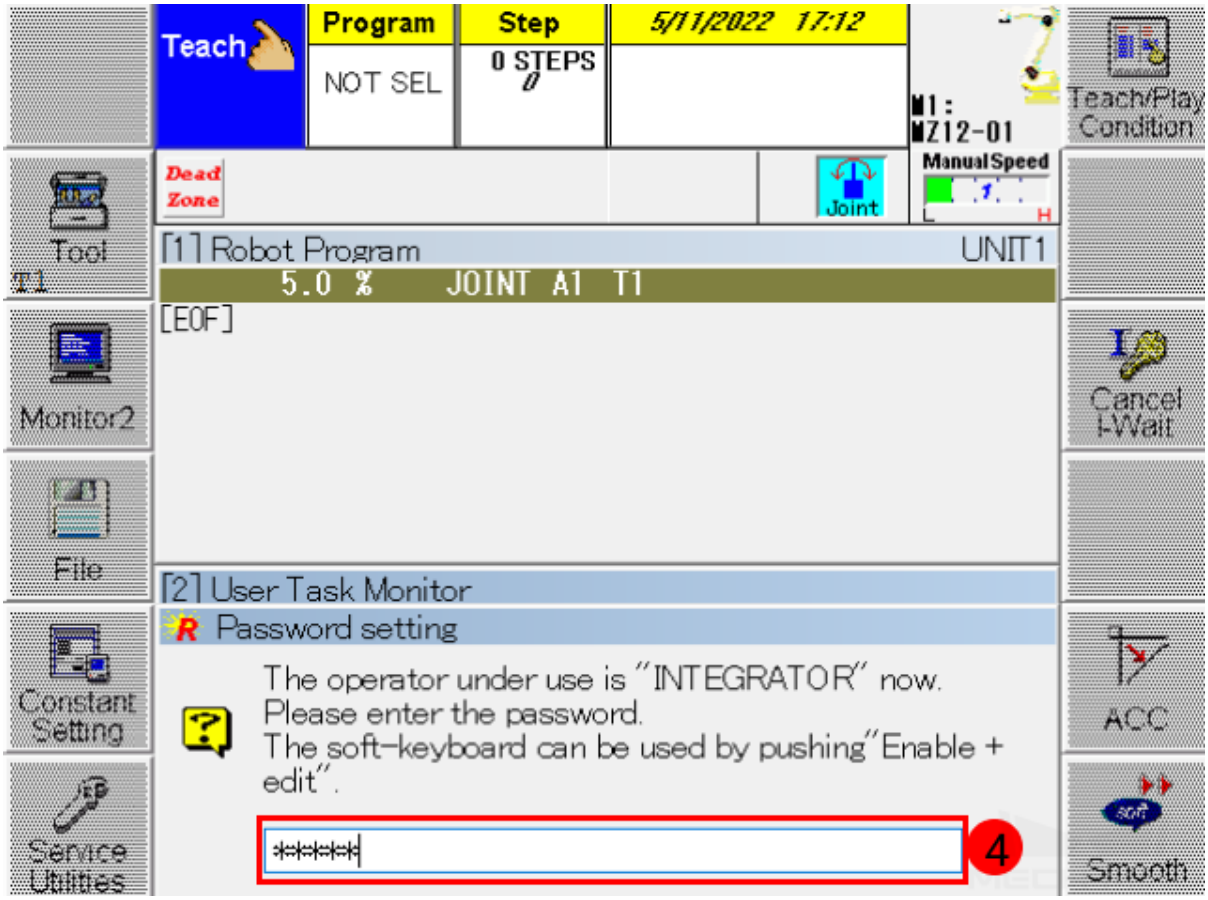


2. Enter **314** in the box as shown below, and then press  (Enter key) on the teach pendant.

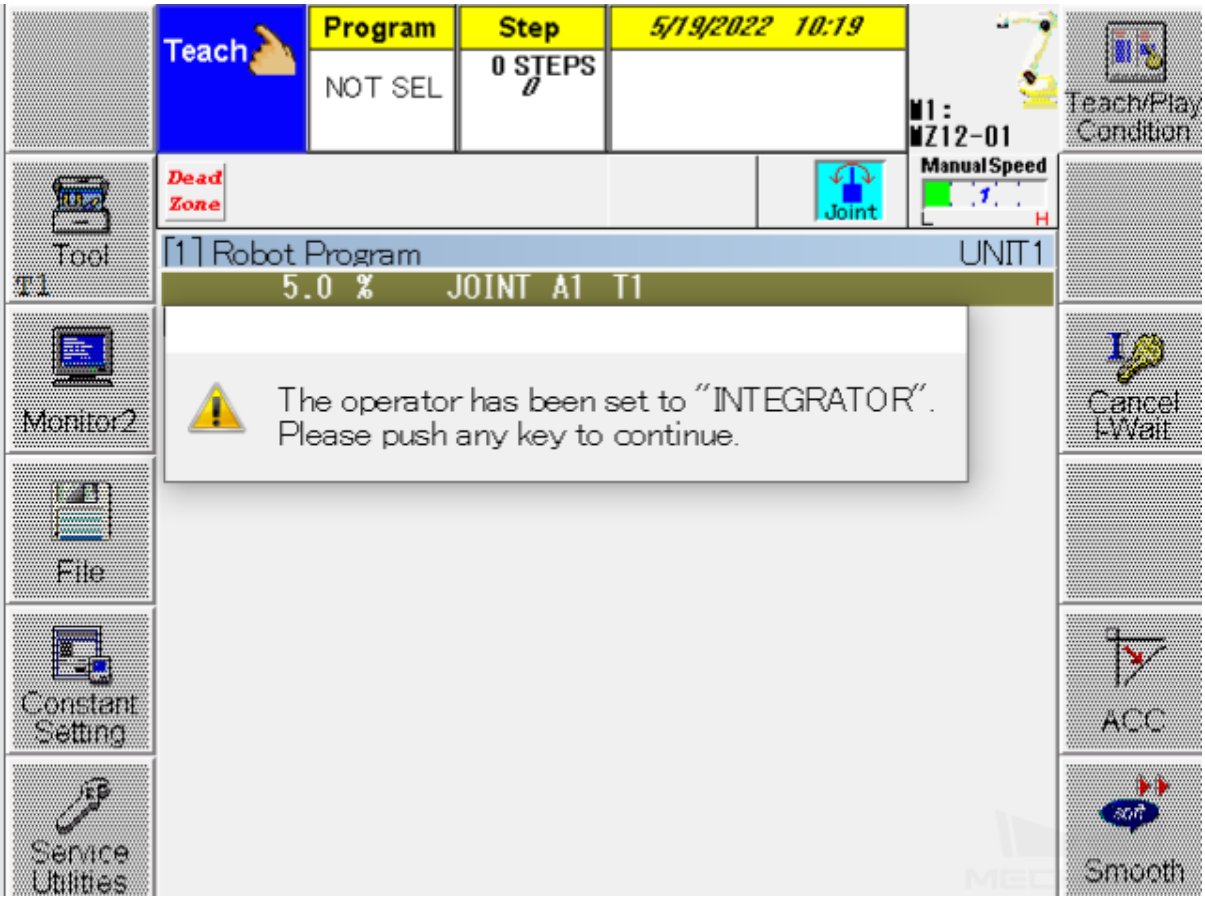

Teach
Program
NOT SEL
Step
0 STEPS
5/11/2022 17:11
M1:
MZ12-01
Teach/Play Condition
Dead Zone
Joint
Manual Speed
Tool
T1
Shortcut R code Entry
UNIT1
Shortcut function list
R314 Change the protecting level
R316 TP LOCK
R317 Change the password(TP LOCK)
R335 Step Copy
R348 Change the displaying language
R354 Interference Disable
R355 Interf. detect. sample start
R356 Interf. detect. sample end
R372 Oscilloscope measurement
R400 Switch Overlap Area
R401 Selction manual cooperation Mechanism
Monitor2
Cancel Wait
File
Constant Setting
ACC
Input the shortcut code. Or locate cursor and press "Enter".
314
2
Service Utilities
Smooth



3. Enter the default password **12345** in the box, and then press  key on the teach pendant to change into **SPECIALIST** level.

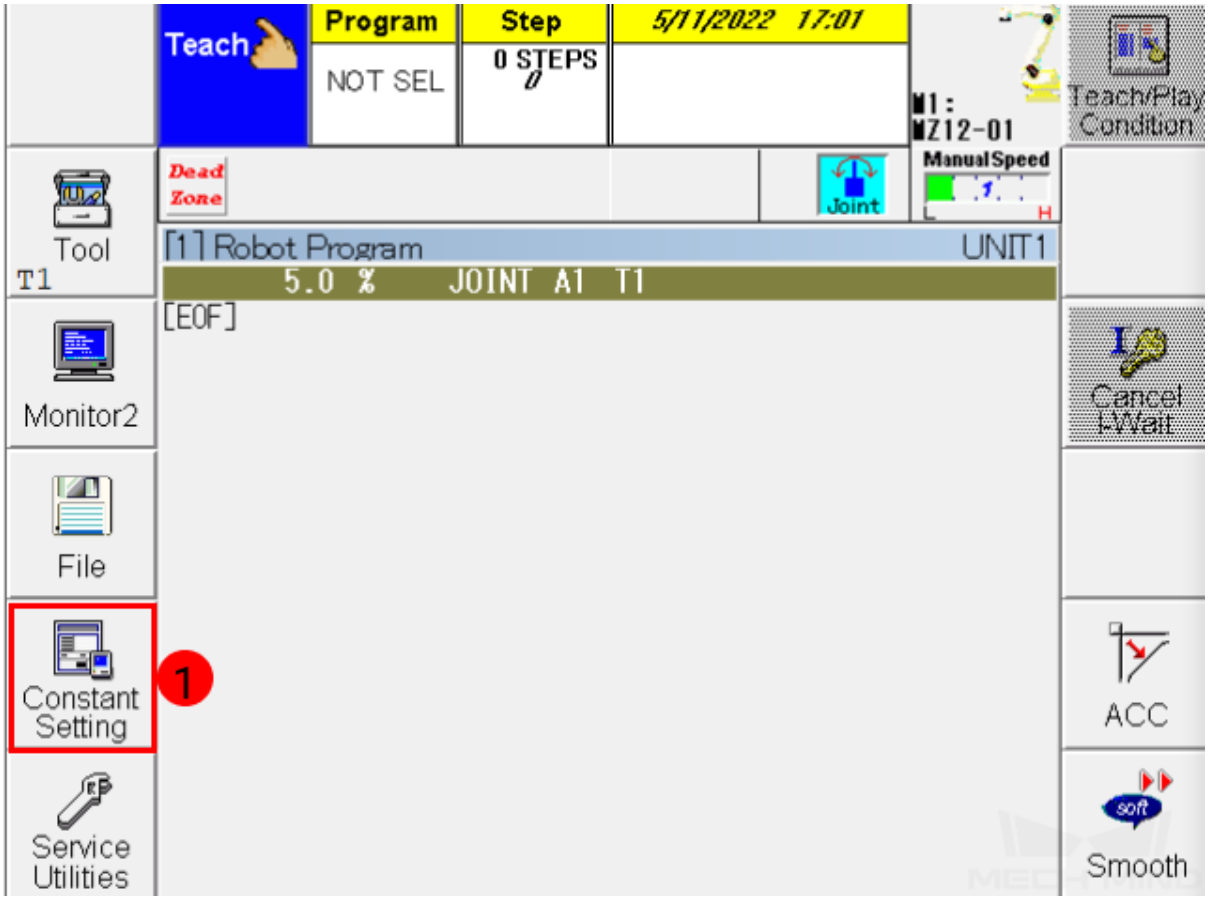


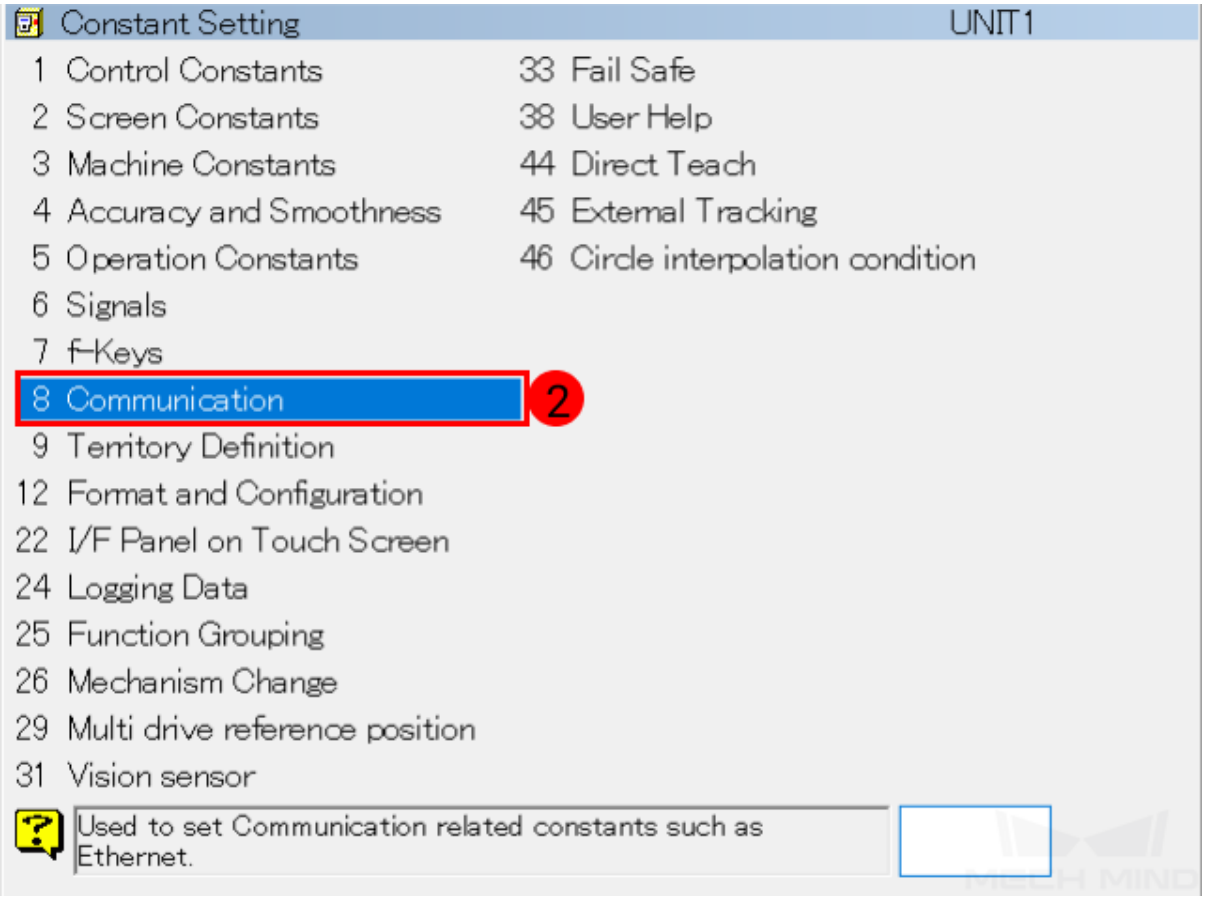
4. A pop-up window as shown below suggests that you have successfully changed the protecting level and all functions can be used from now on.



IP Configuration

1. Go to *Constant Setting*→ *8 Communication*→ *2 Ethernet* → *1 TCP/IP* to configure **TCP/IP Settings**.


Teach
Program
NOT SEL
Step
0 STEPS
5/11/2022 17:01
Teach/Play Condition
Dead Zone
Joint
Manual Speed
Tool
T1
[1] Robot Program
UNIT1
5.0 % JOINT A1 T1
[EOF]
Monitor2
Cancel I-Wait
File
Constant Setting
1
ACC
Service Utilities
Smooth



2. Enter the robot IP address in the **IP Address** box.

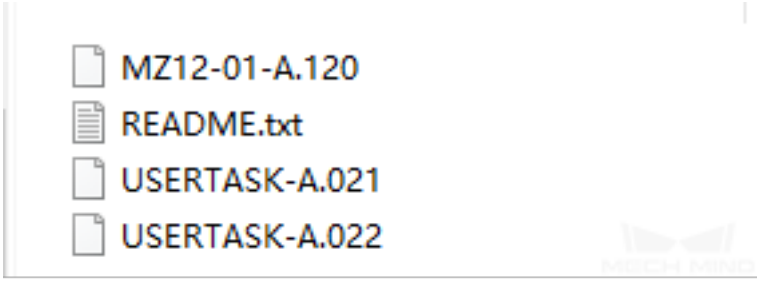
Hint: The robot IP should be in the same subnet as the IPC. If you need to set the static IP, please contact the network administrator.

1.6.3 Load the Program Files

Prepare the Files

Copy the full-control program files of NACHI robot with an USB flash drive.

The files are stored in: XXXX/Mech-Center/Mech_RobServ/install_packages/nachi

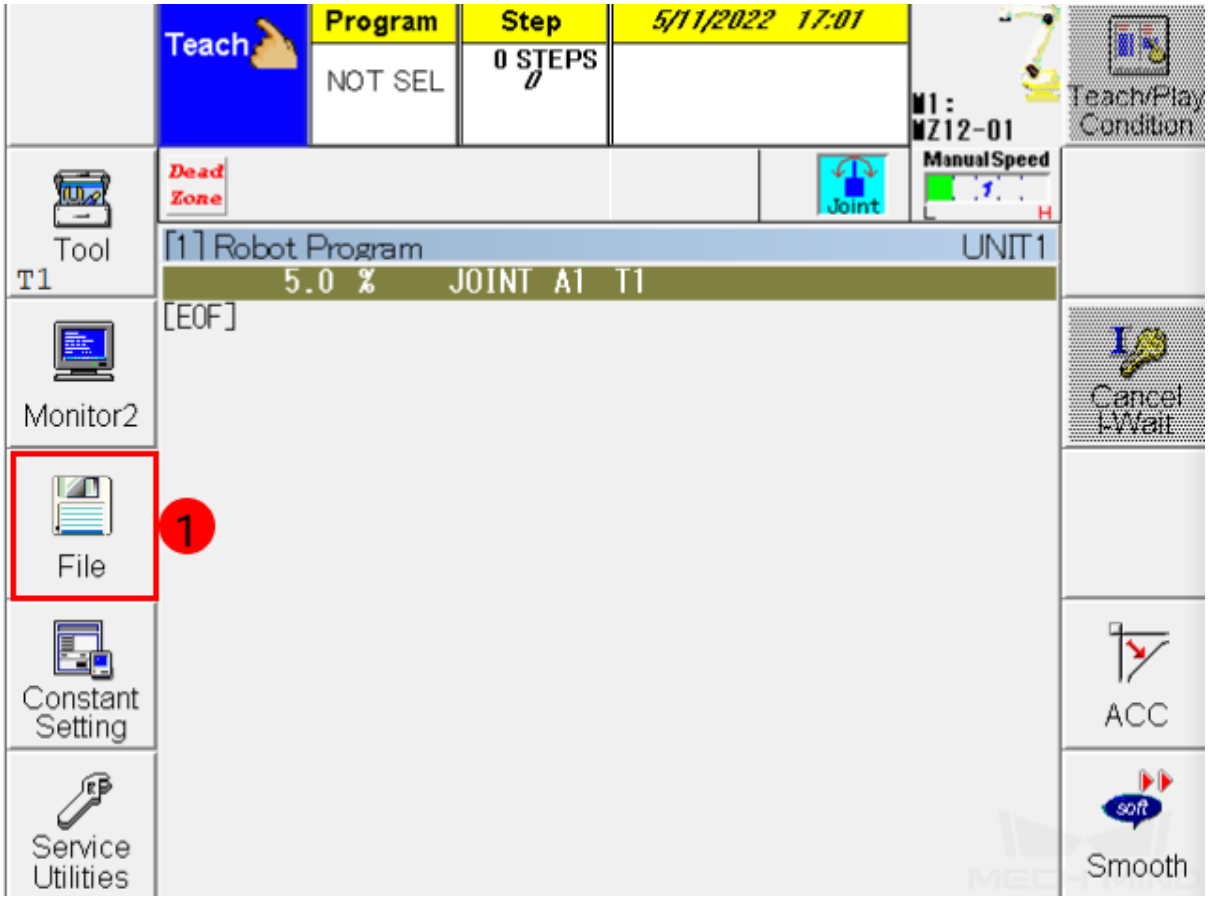


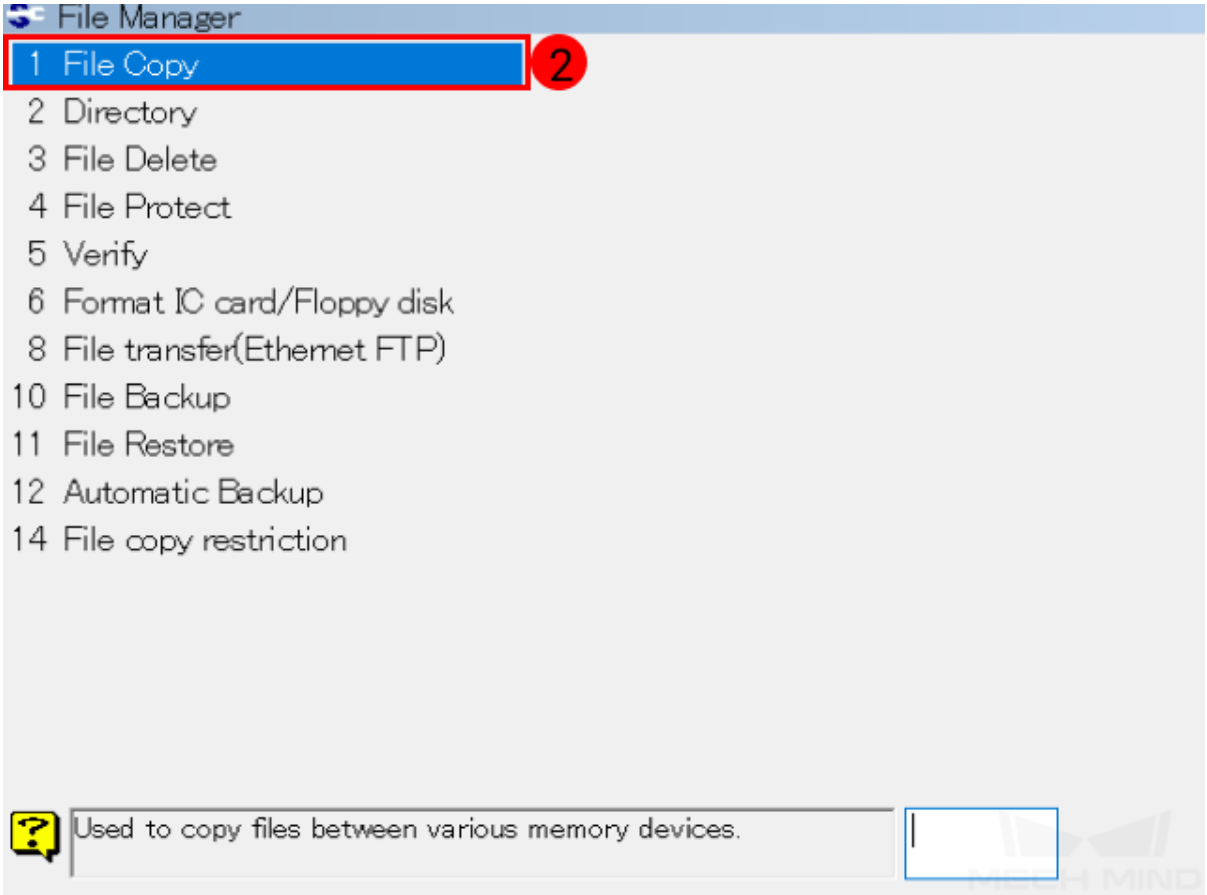
Attention: Please rename the MZ12-01-A.120 file according to the actual robot model name. For example, when loading the file to the SRA166-1 robot, rename the file as SRA166-1-A.120, and then copy and paste it into your flash drive.

Load the File to the Robot

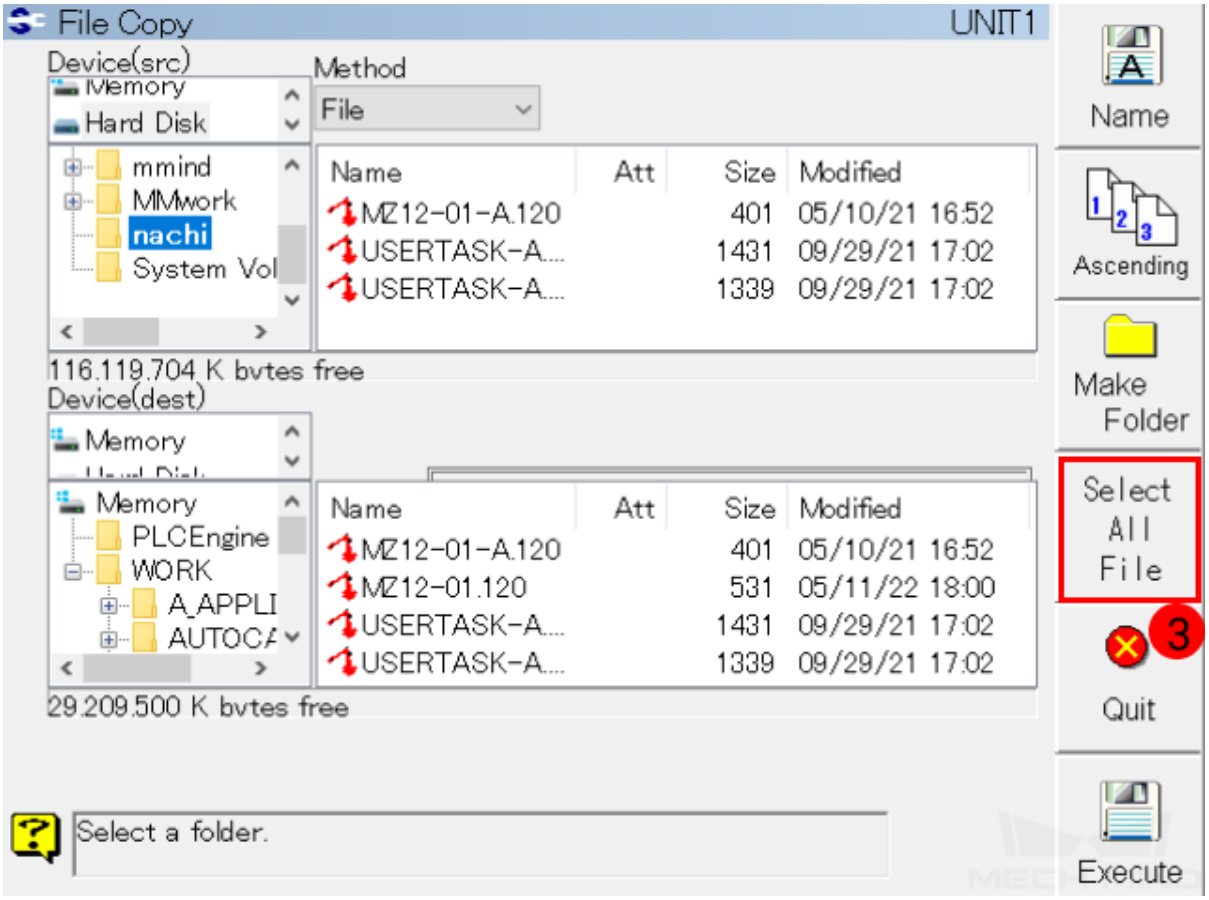
1. Plug the USB flash drive into the USB port on the back of the teach pendant, as shown below.

2. Select *File*→ *File copy* on the touch panel.

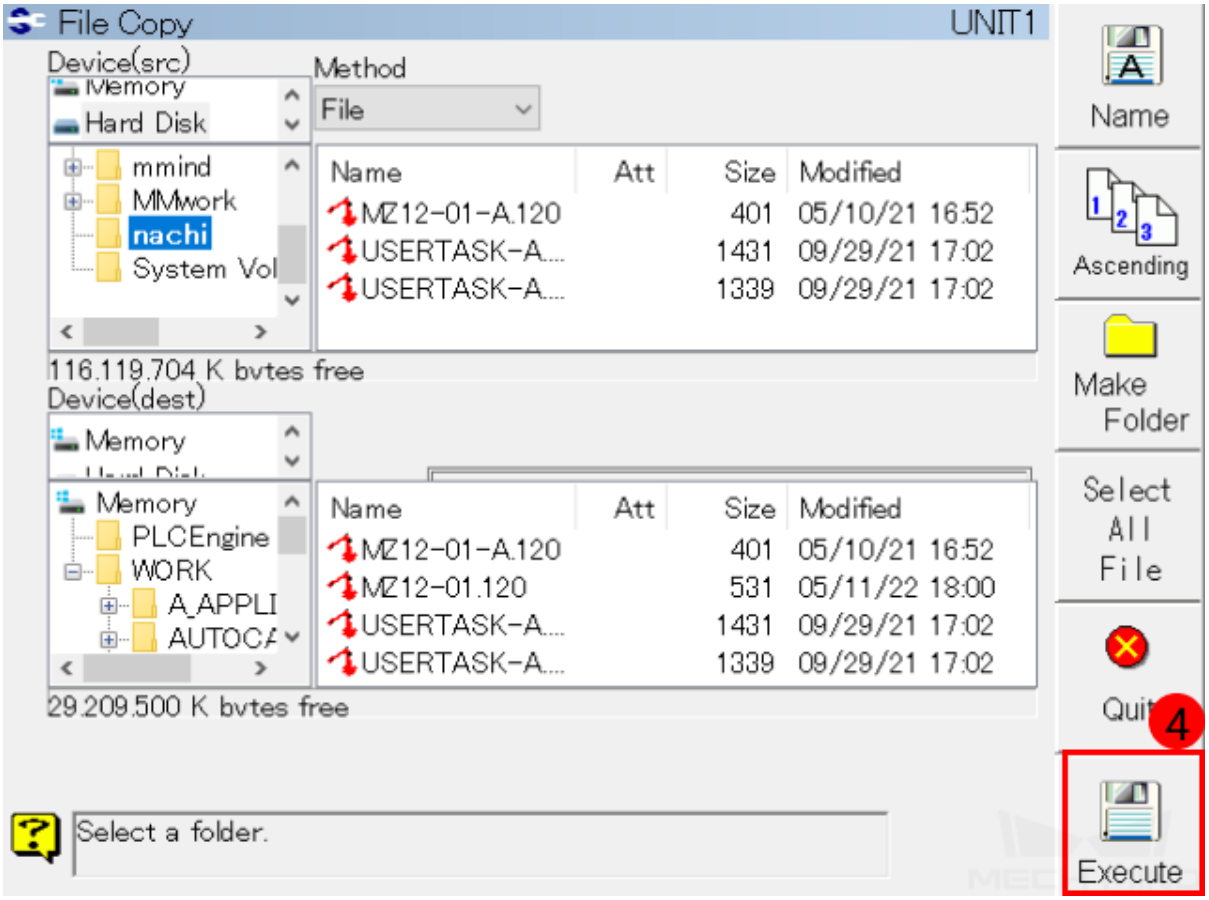

Teach
Program
NOT SEL
Step
0 STEPS
5/11/2022 17:01
M1:
MZ12-01
Teach/Play Condition
Dead Zone
Joint
Manual Speed
Tool
T1
[1] Robot Program
UNIT1
5.0 % JOINT A1 T1
[EOF]
Monitor2
Cancel I-Wait
File
1
Constant Setting
ACC
Service Utilities
Smooth



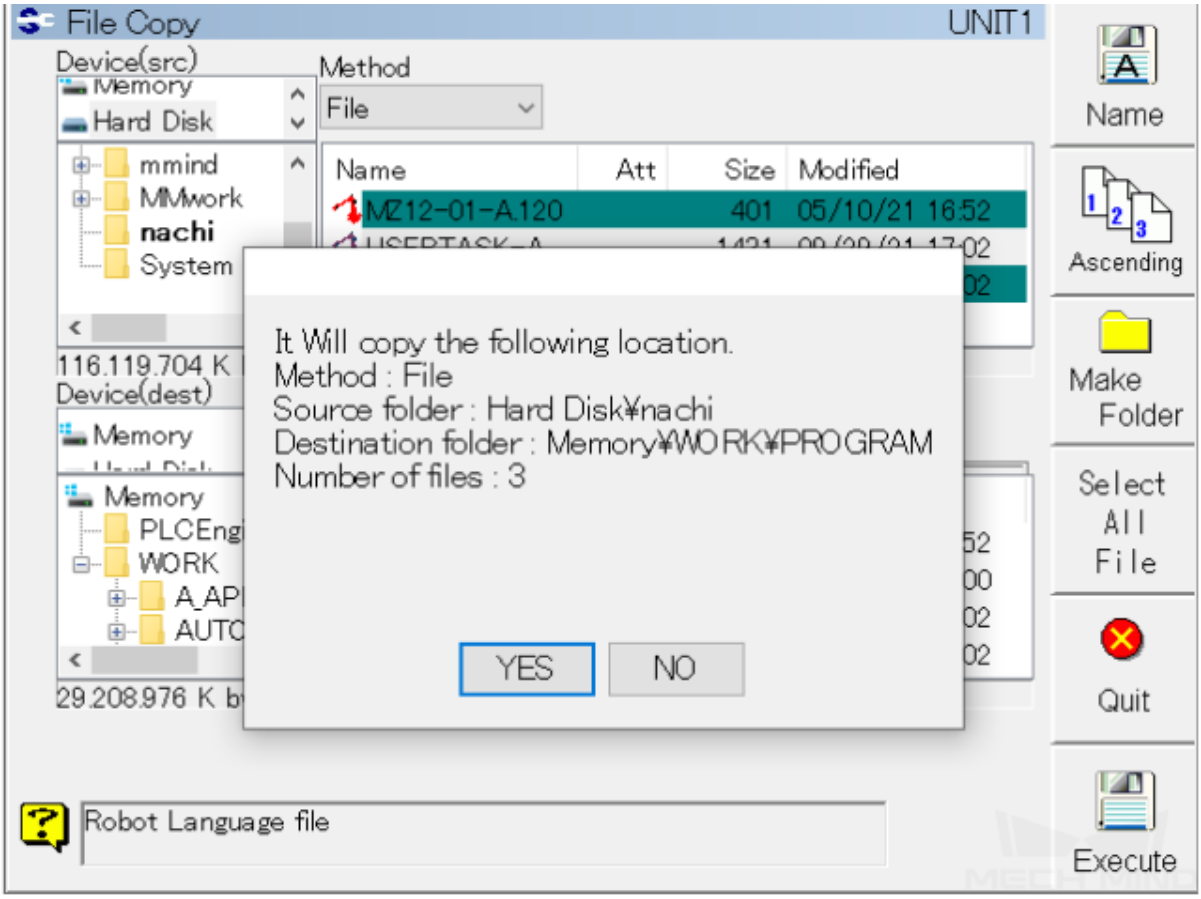
3. Device (src) is the folder of the USB flash drive. Select the folder where the program files are stored under Device (src) and the **PROGRAM** folder under Device (dest) and then press on *Select All File*

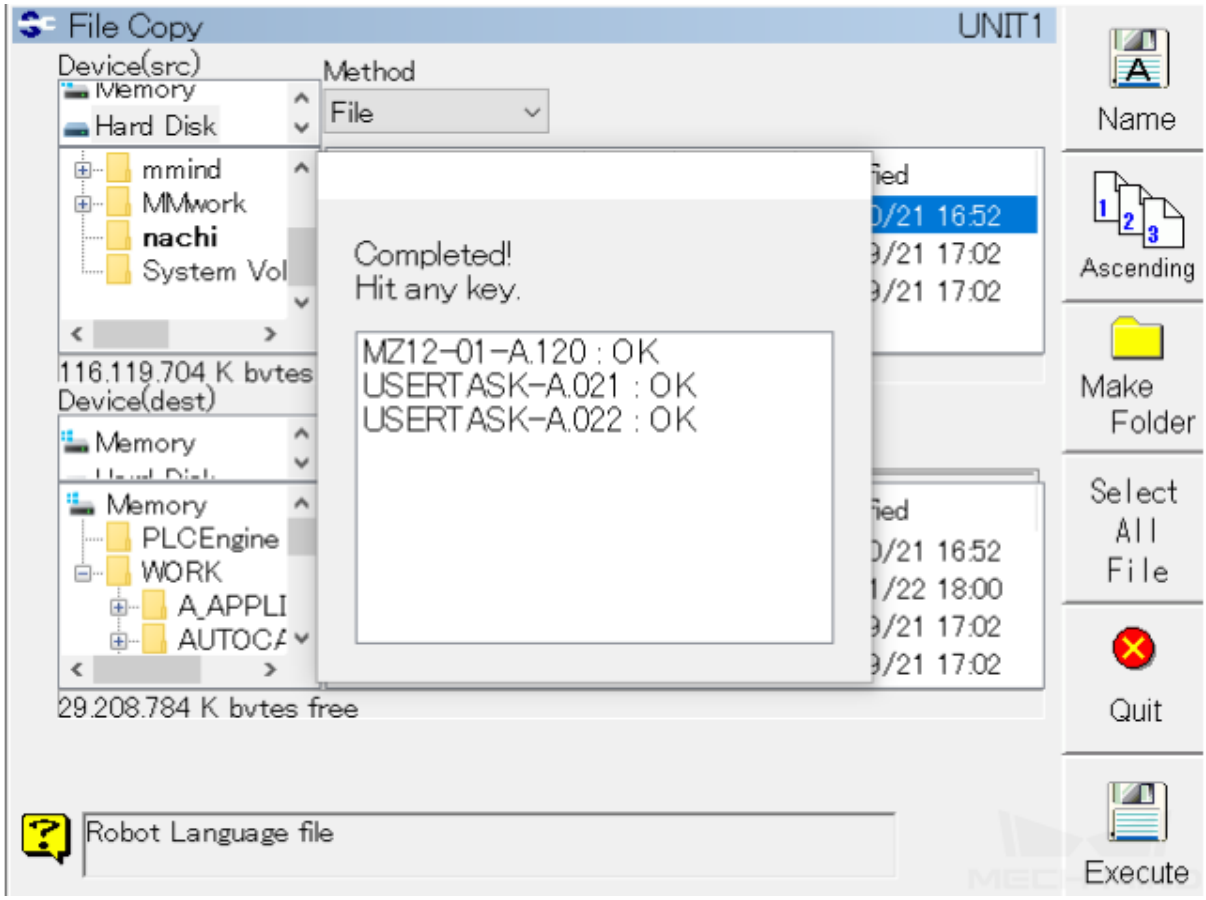


4. Press on *Execute* to import files.



5. If the following messages appear on the screen, the program files have been loaded successfully.

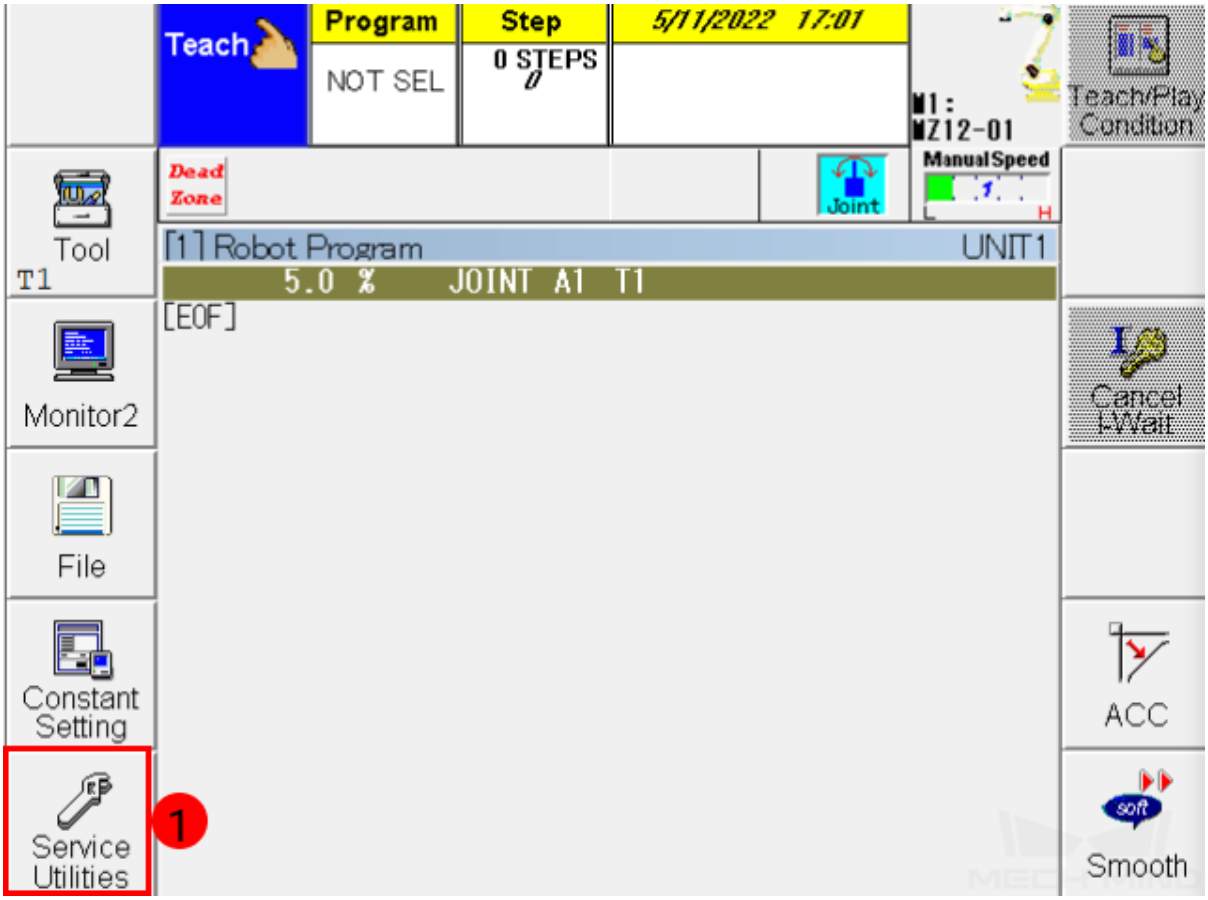

File Copy
UNIT1
Device(src)
Memory
Hard Disk
mmind
MMwork
nachi
System
Method
File
Name
Att
Size
Modified
MZ12-01-A.120
401
05/10/21 16:52
116,119,704 K
Device(dest)
Memory
Memory
PLCEngi
WORK
A_AP
AUTC
29,208,976 K b
It Will copy the following location.
Method : File
Source folder : Hard Disk¥nachi
Destination folder : Memory¥WORK¥PROGRAM
Number of files : 3
YES
NO
Robot Language file
Name
Ascending
Make Folder
Select All File
Quit
Execute

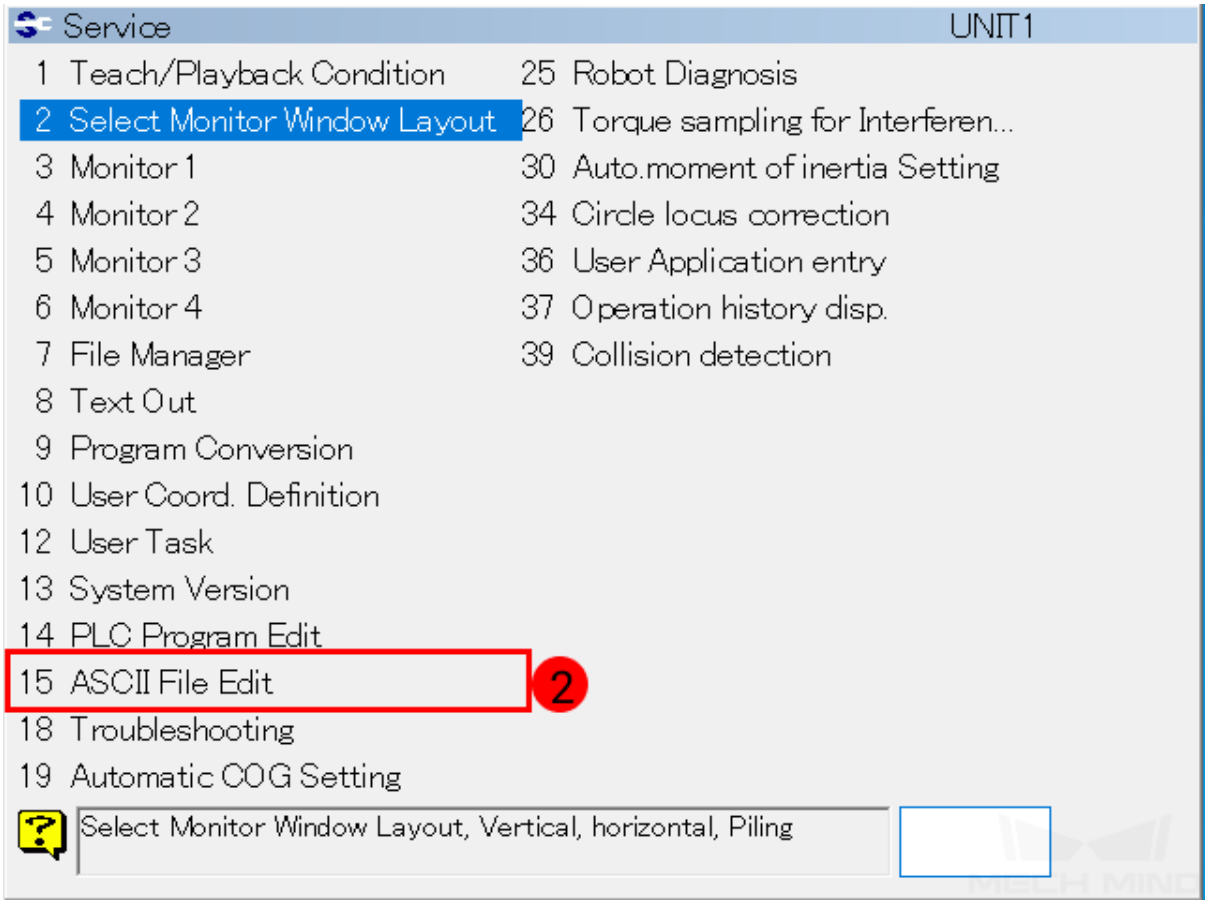


Attention: Please reboot the robot after exiting the program.

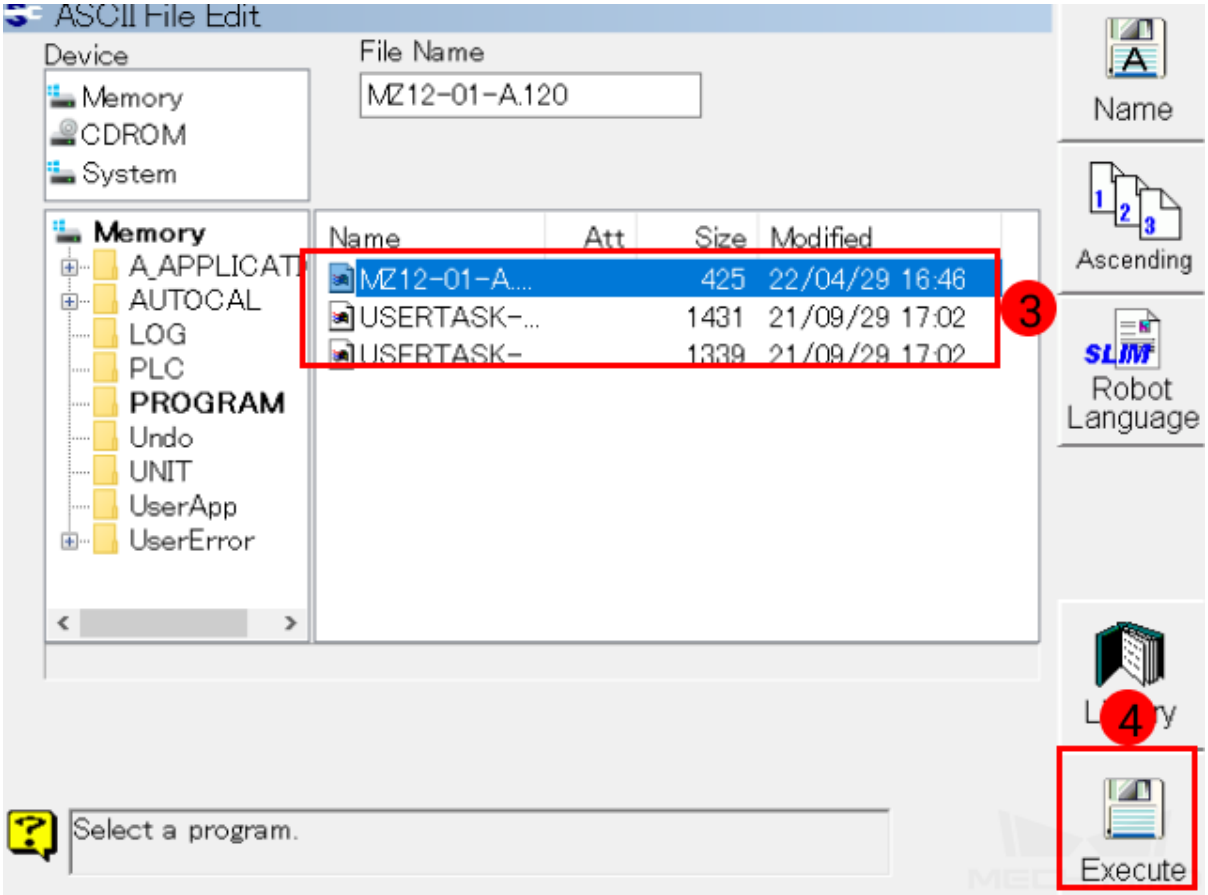
Convert the Program File

1. Go back to the main interface, press on *Service Utilities*→ *ASCII File Edit* to edit the ASCII files.

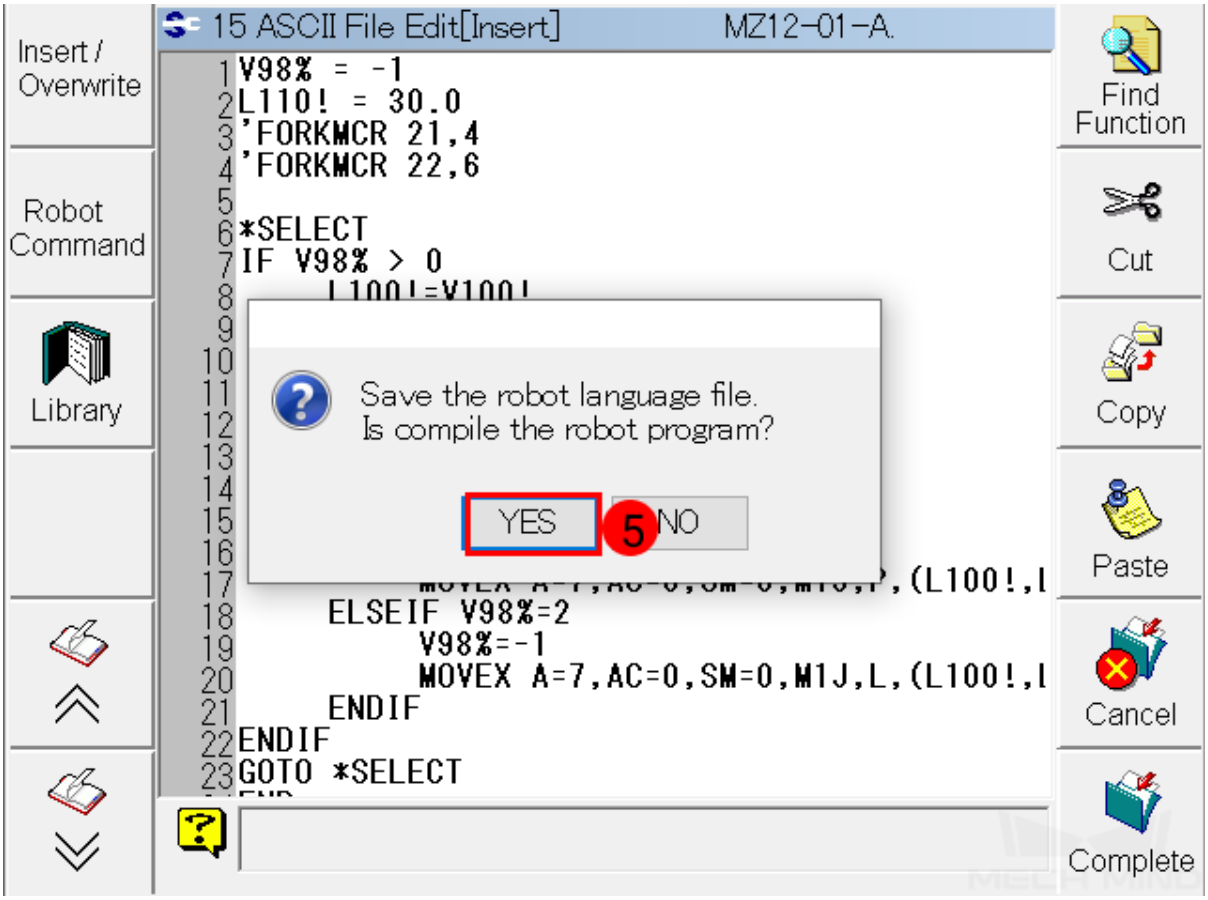

Teach
Program
NOT SEL
Step
0 STEPS
5/11/2022 17:01
M1:
MZ12-01
Teach/Play Condition
Dead Zone
Joint
Manual Speed
Tool
T1
[1] Robot Program
UNIT1
5.0 % JOINT A1 T1
[EOF]
Monitor2
Cancel I-Wait
File
Constant Setting
ACC
Service Utilities
1
Smooth



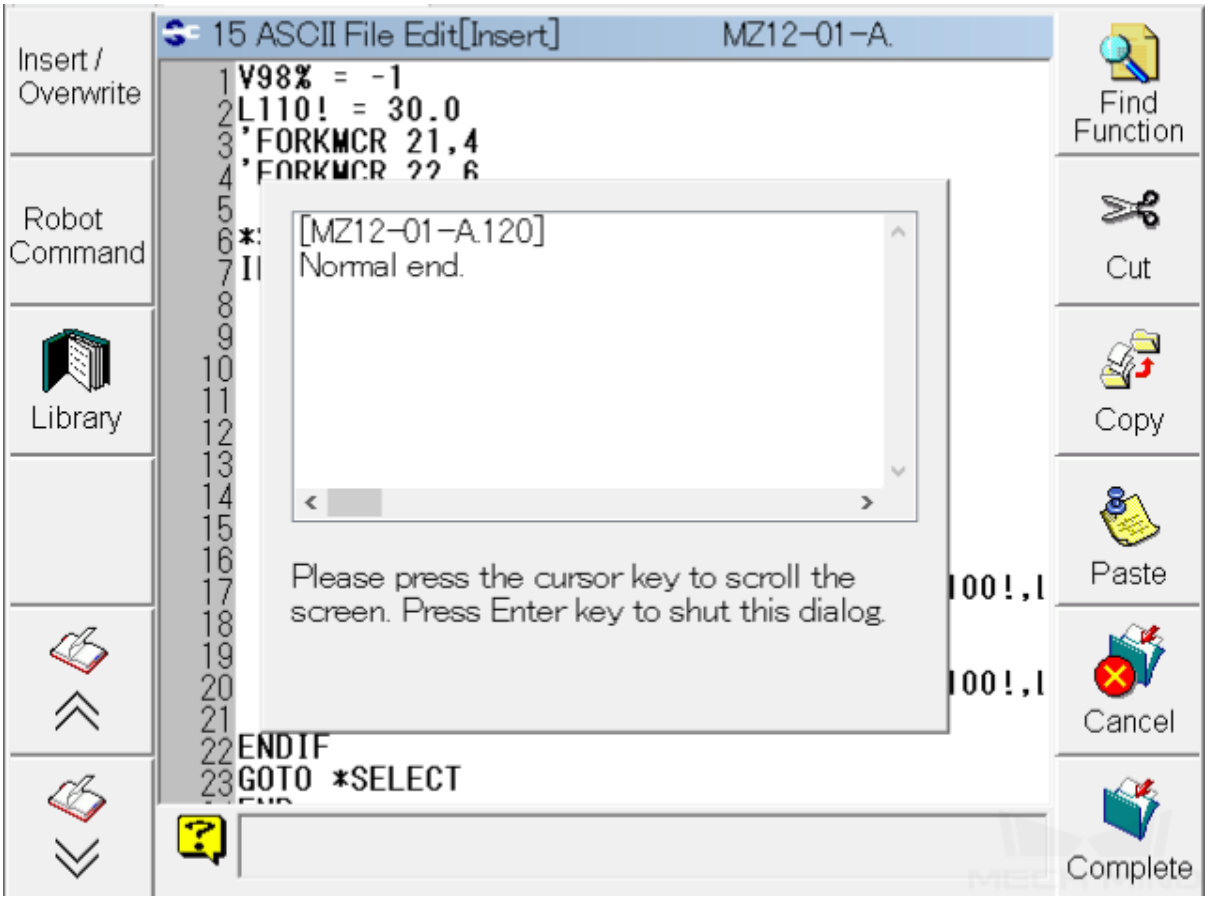
2. Select the file and press on *Execute*. Please perform the same operation on all three files in turn, and you can only start executing after the previous execution is completed.



3. After pressing on *Execute*, a window as below will pop up, and then select *Yes*.

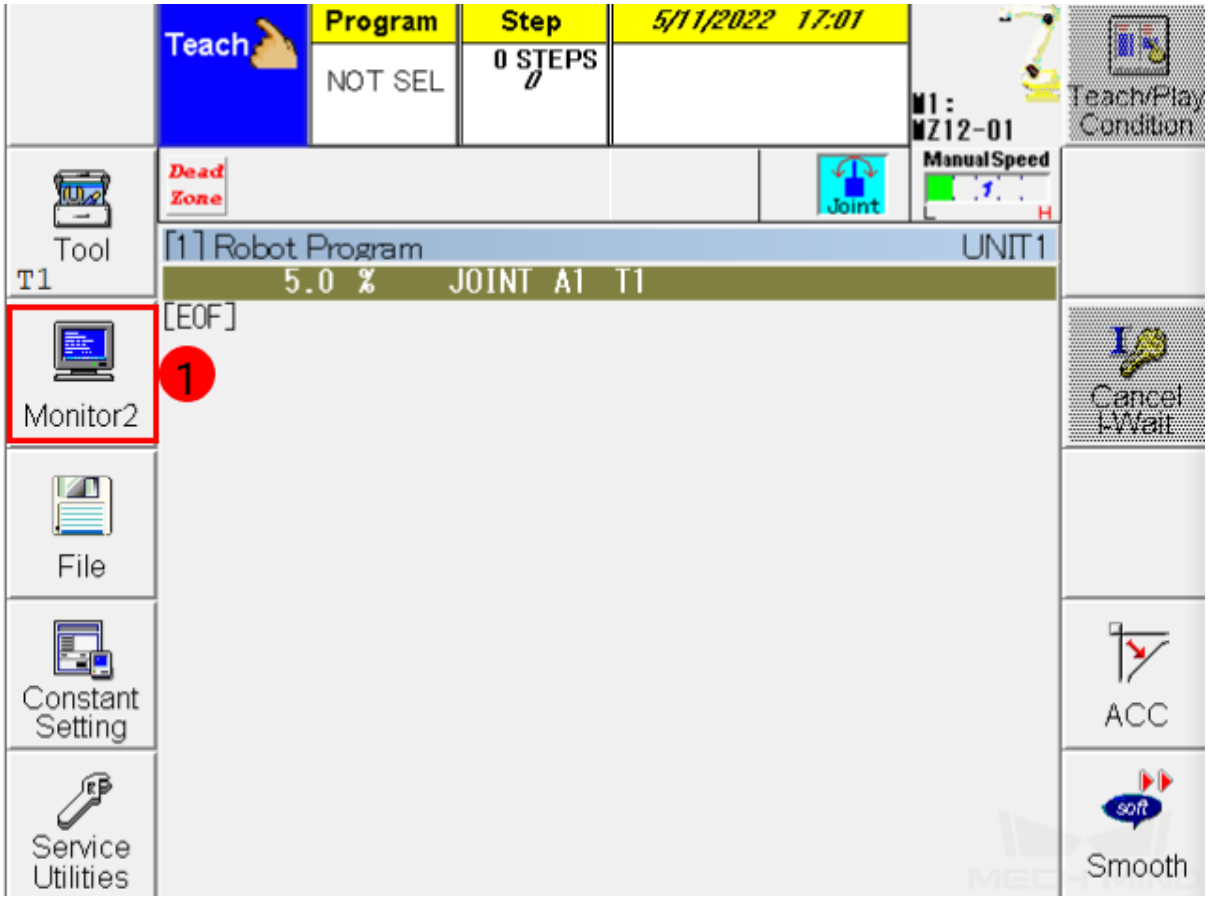


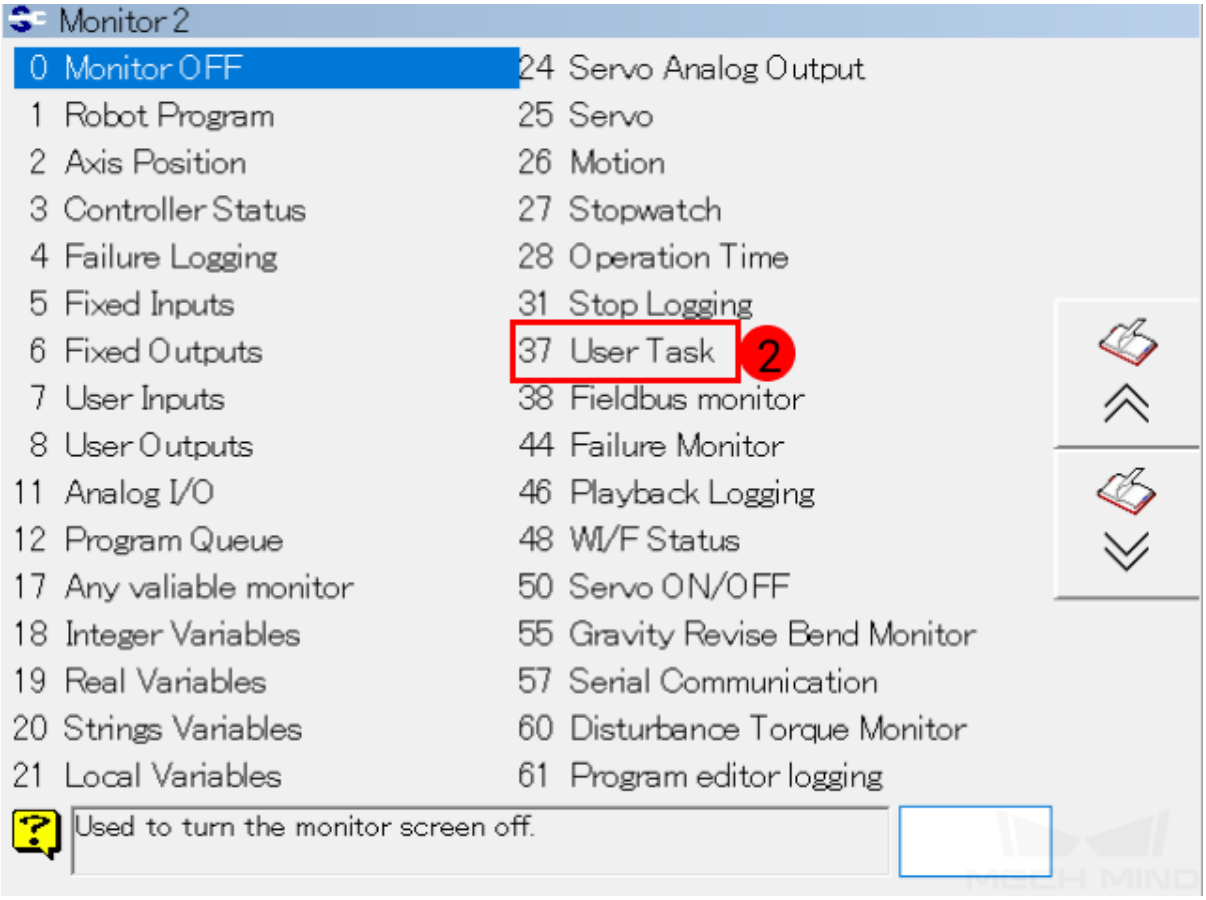
4. After converting the program file to the robot language, a message as shown below will appear.

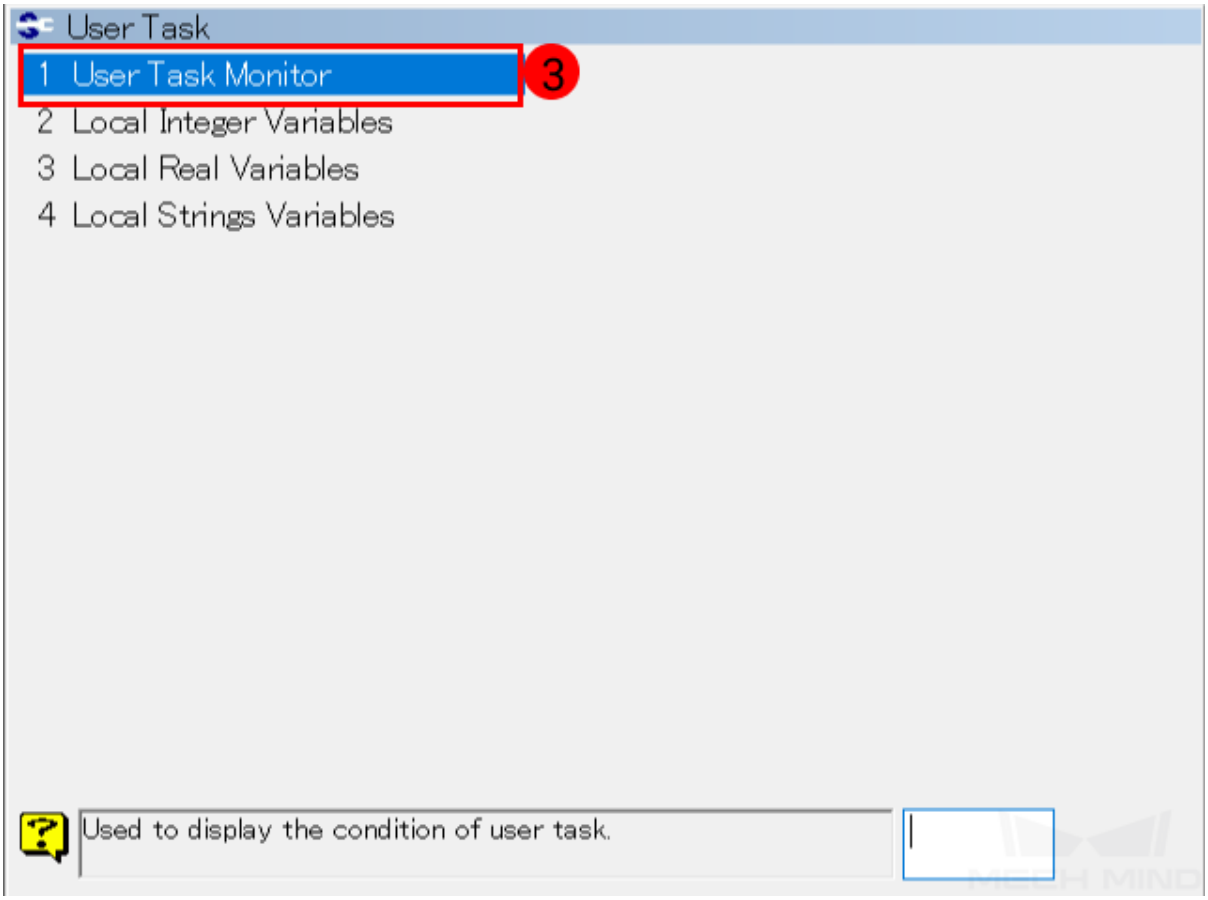


Designate the Program

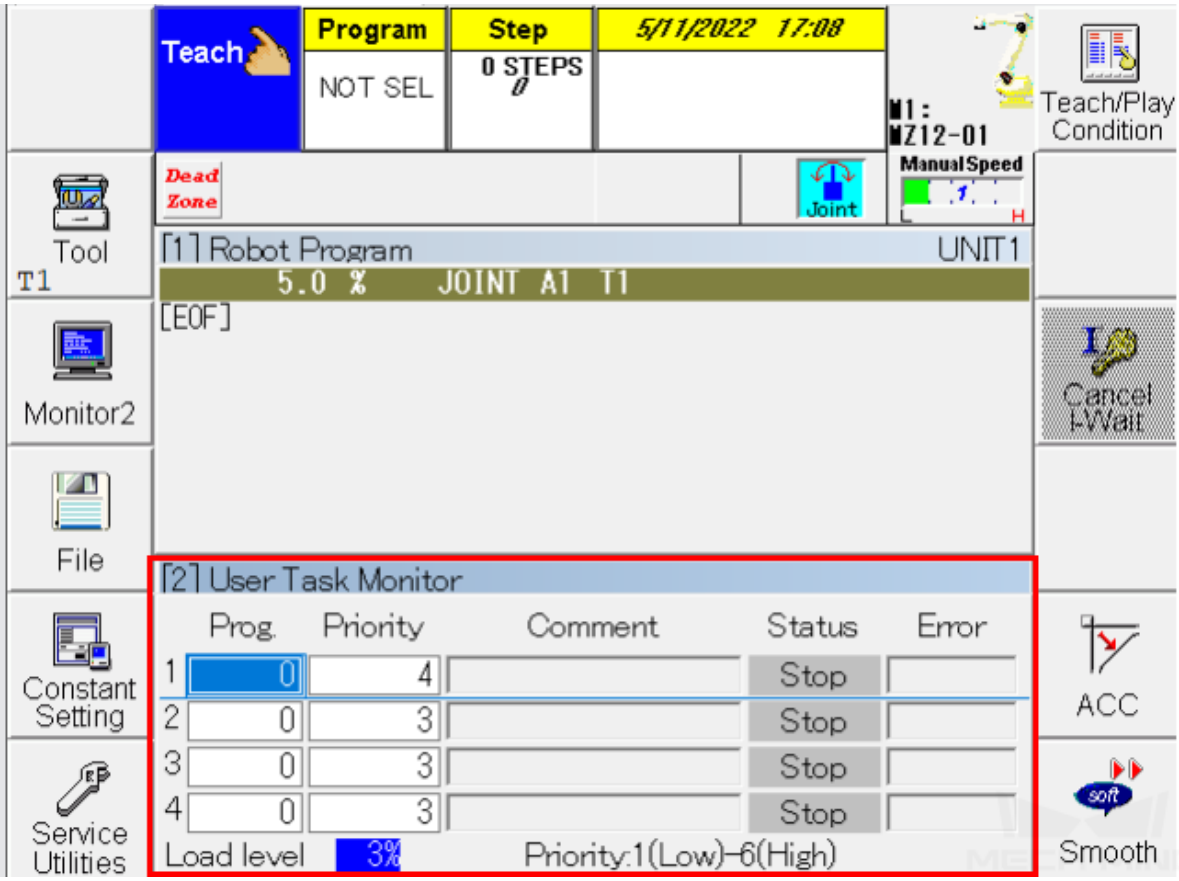
1. Return to the main interface, go to *Monitor2–> User Task–> User Task Monitor*.

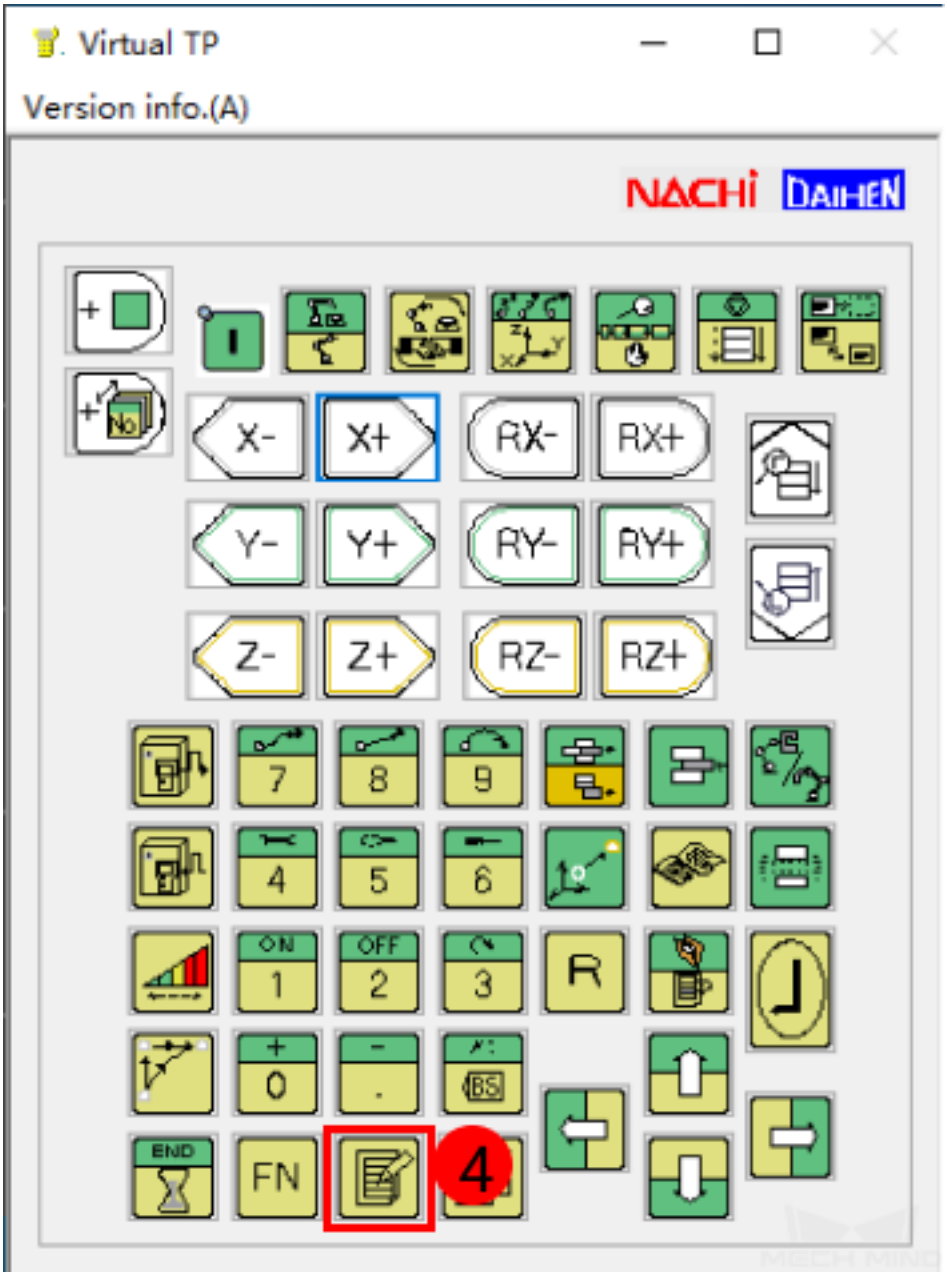

Teach
Program
NOT SEL
Step
0 STEPS
5/11/2022 17:01
M1:
MZ12-01
Teach/Play Condition
Dead Zone
Joint
Manual Speed
Tool
T1
[1] Robot Program
UNIT1
5.0 % JOINT A1 T1
[EOF]
1
Monitor2
Cancel I-Wait
File
Constant Setting
ACC
Service Utilities
Smooth

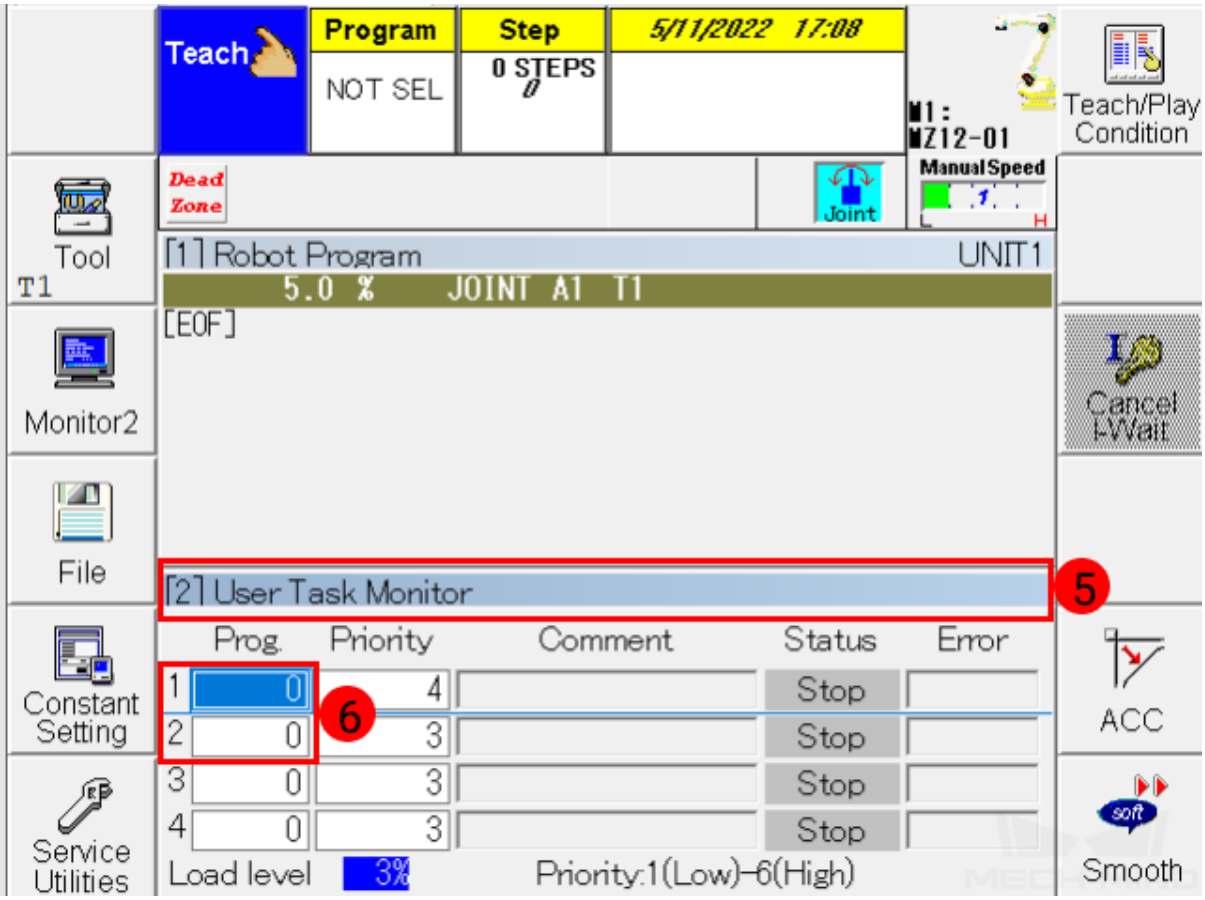

Monitor 2
0 Monitor OFF
1 Robot Program
2 Axis Position
3 Controller Status
4 Failure Logging
5 Fixed Inputs
6 Fixed Outputs
7 User Inputs
8 User Outputs
11 Analog I/O
12 Program Queue
17 Any valiable monitor
18 Integer Variables
19 Real Variables
20 Strings Variables
21 Local Variables
24 Servo Analog Output
25 Servo
26 Motion
27 Stopwatch
28 Operation Time
31 Stop Logging
37 User Task
2
38 Fieldbus monitor
44 Failure Monitor
46 Playback Logging
48 WI/F Status
50 Servo ON/OFF
55 Gravity Revise Bend Monitor
57 Serial Communication
60 Disturbance Torque Monitor
61 Program editor logging
Used to turn the monitor screen off.

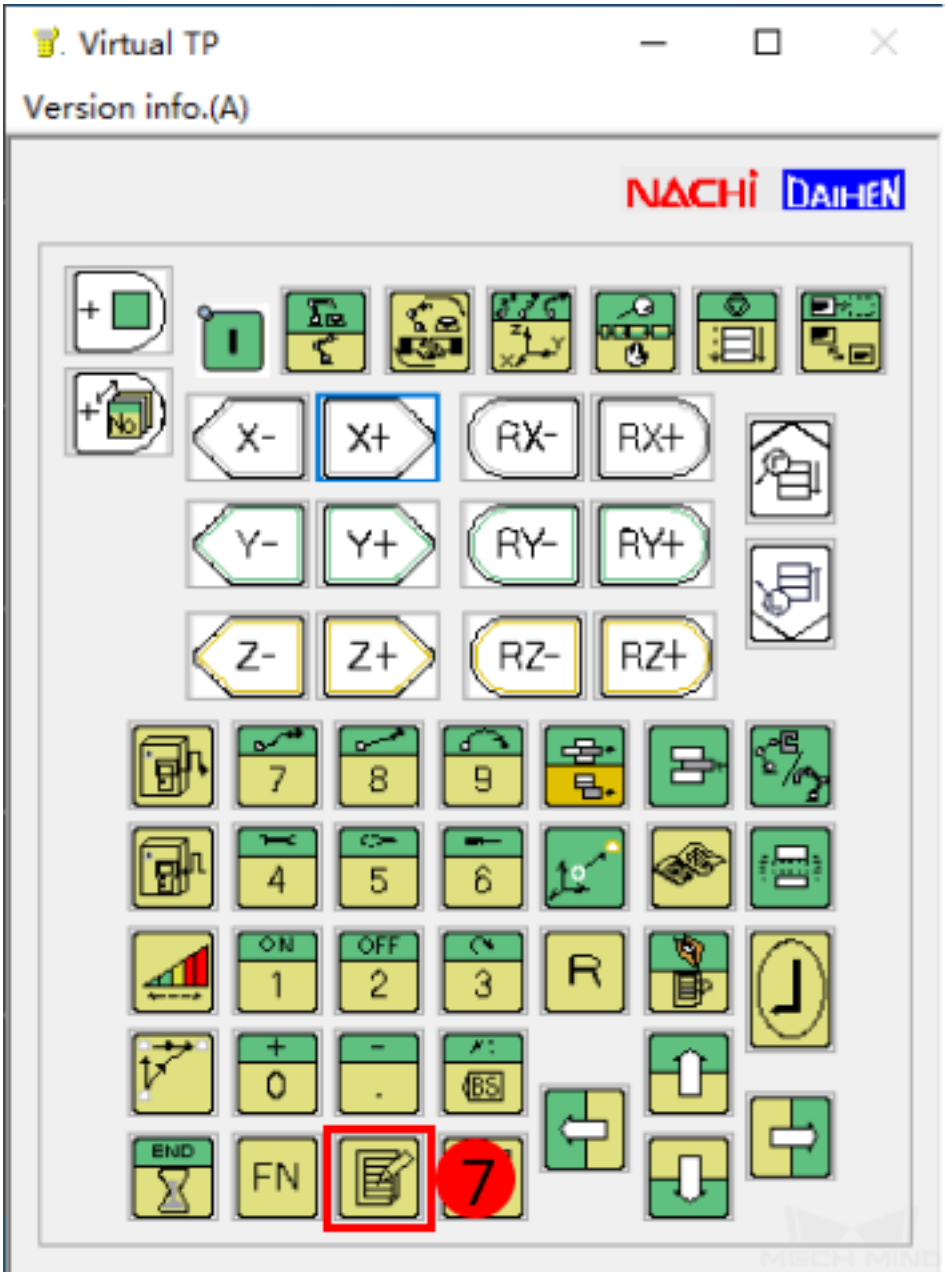


2. Now you can see the **User Task Monitor** as shown below. Press the  key on the teach pendant, and then the **User Task Monitor** ( ) line will turn orange, suggesting that it is editable now. Enter 21 in the first line in the Program column, and enter 22 in the second line. Then, press  key.


Teach
Program
NOT SEL
Step
0 STEPS
5/11/2022 17:08
M1:
MZ12-01
Manual Speed
Teach/Play Condition
Dead Zone
Joint
Tool
T1
[1] Robot Program
UNIT1
5.0 % JOINT A1 T1
[EOF]
Monitor2
Cancel I-Wait
File
[2] User Task Monitor
Prog. Priority Comment Status Error
1 0 4 Stop
2 0 3 Stop
3 0 3 Stop
4 0 3 Stop
Load level 3%
Priority:1(Low)-6(High)
Constant Setting
ACC
Service Utilities
Smooth

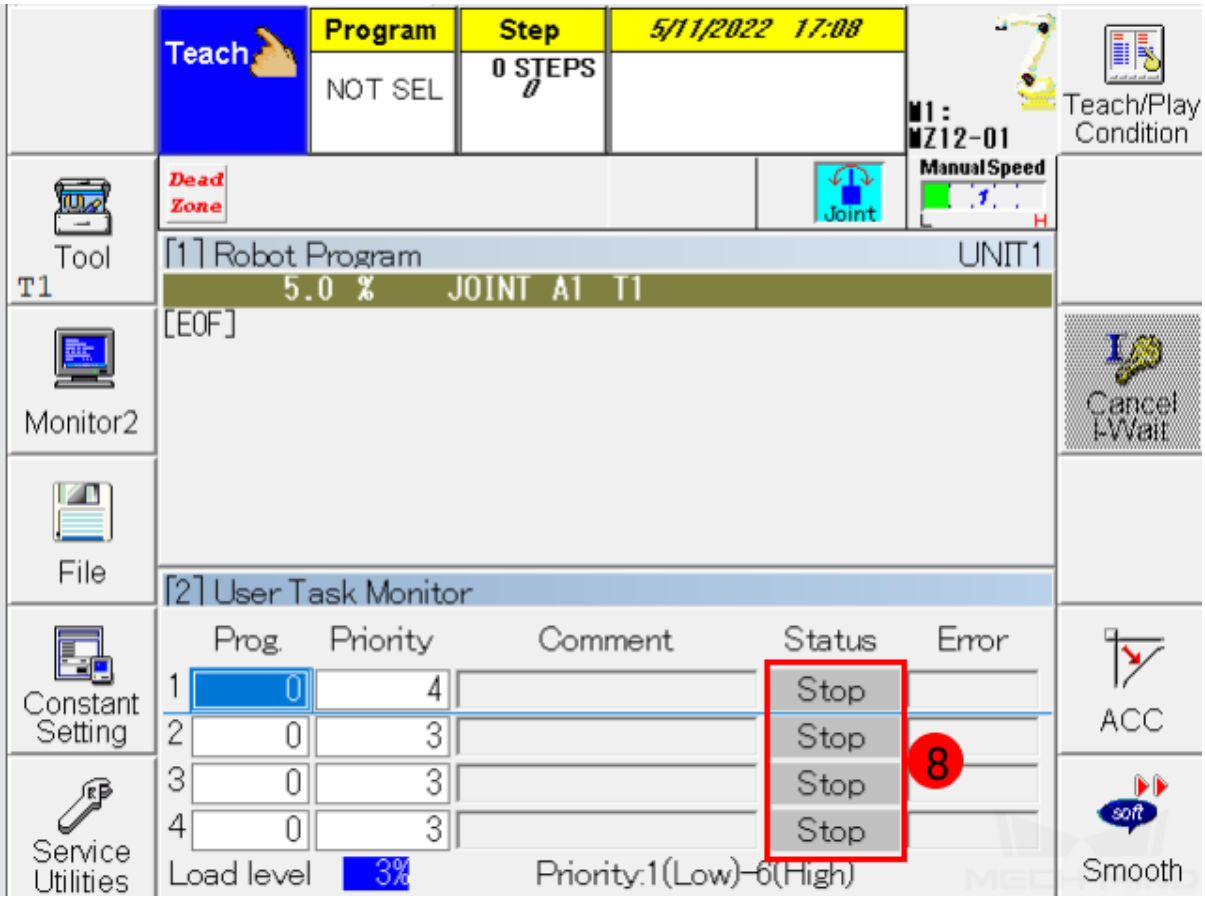

Virtual TP
Version info.(A)
NACHI
DAIHEN
X-
X+
RX-
RX+
Y-
Y+
RY-
RY+
Z-
Z+
RZ-
RZ+
7
8
9
4
5
6
ON
1
OFF
2
3
R
0
.
BS
END
FN
4

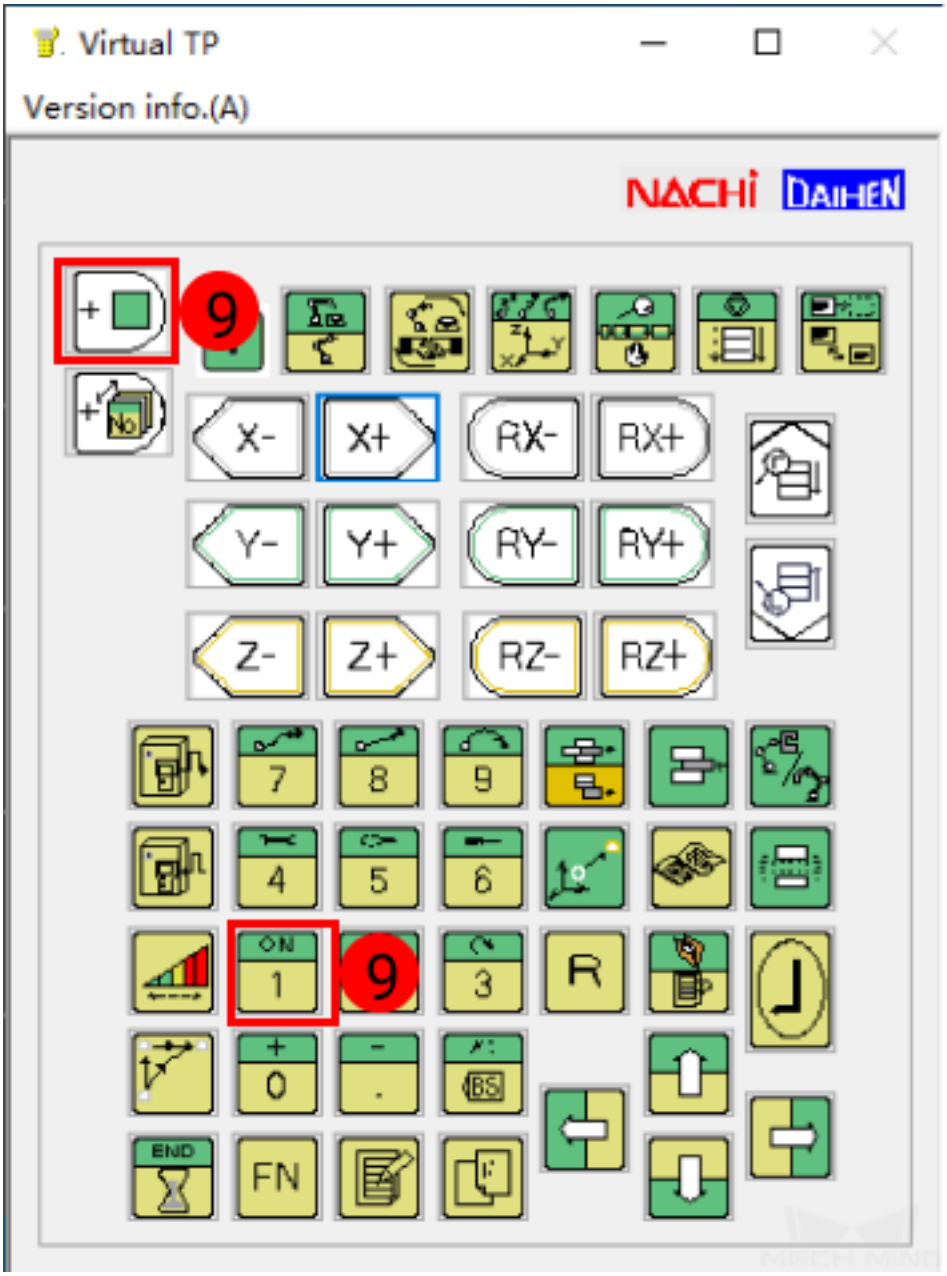

Teach
Program
NOT SEL
Step
0 STEPS
5/11/2022 17:08
M1:
MZ12-01
Teach/Play Condition
Dead Zone
Joint
Manual Speed
Tool
T1
[1] Robot Program
UNIT1
5.0 % JOINT A1 T1
[EOF]
Monitor2
Cancel I-Wait
File
[2] User Task Monitor
5
Prog.
Priority
Comment
Status
Error
1 0 4 Stop
2 0 3 Stop
3 0 3 Stop
4 0 3 Stop
6
Constant Setting
ACC
Service Utilities
Smooth
Load level 3%
Priority:1(Low)-6(High)



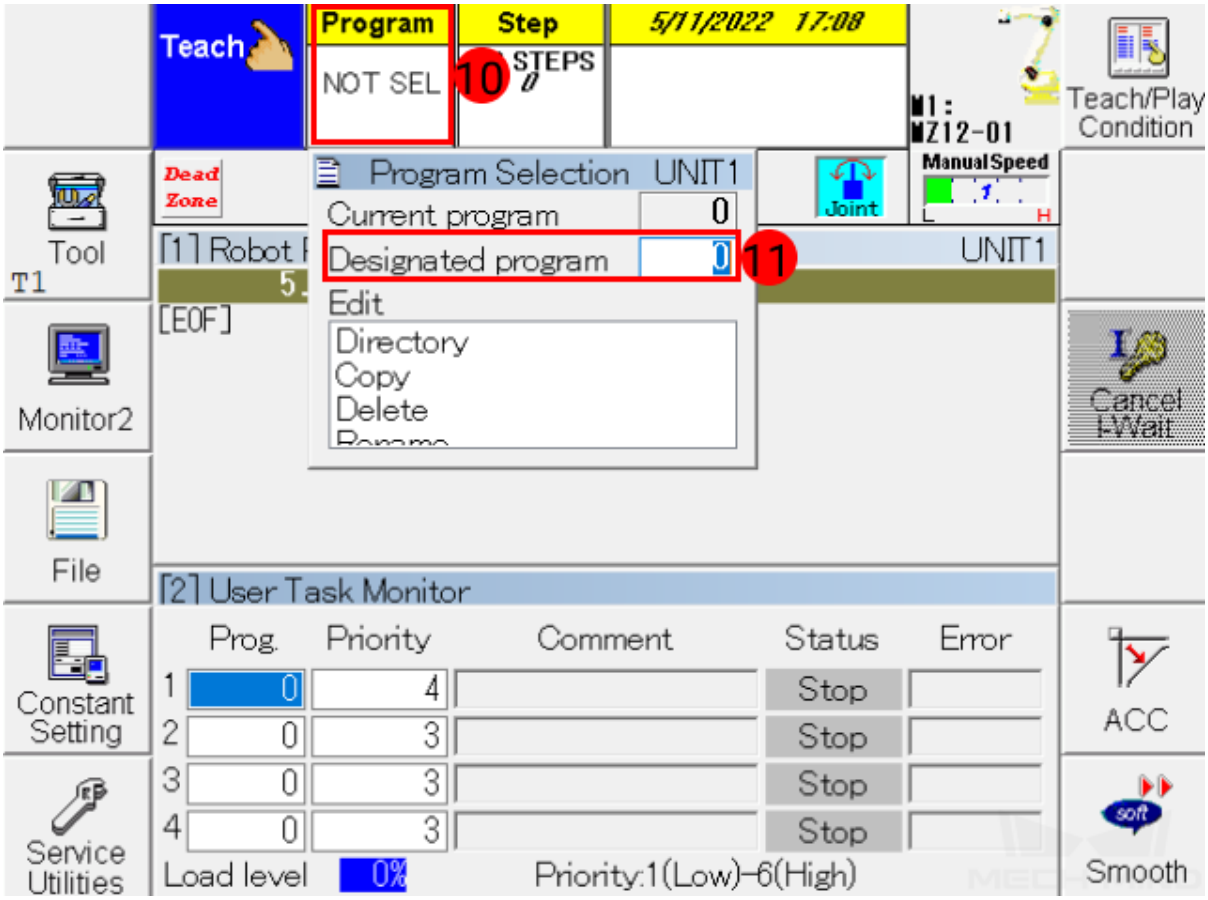
Hint: If the robot cannot move in a smooth way, please change the priority level of the program 21 from 4 to 5.

3. Select the **Status** column, and press  and  keys at the same time.


Teach
Program
NOT SEL
Step
0 STEPS
5/11/2022 17:08
M1:
MZ12-01
Teach/Play Condition
Dead Zone
Joint
Manual Speed
Tool
T1
[1] Robot Program
UNIT1
5.0 % JOINT A1 T1
[EOF]
Monitor2
Cancel I-Wait
File
[2] User Task Monitor
Prog.
Priority
Comment
Status
Error
1 0 4 Stop
2 0 3 Stop
3 0 3 Stop
4 0 3 Stop
8
Constant Setting
ACC
Service Utilities
Smooth
Load level 3%
Priority:1(Low)-6(High)

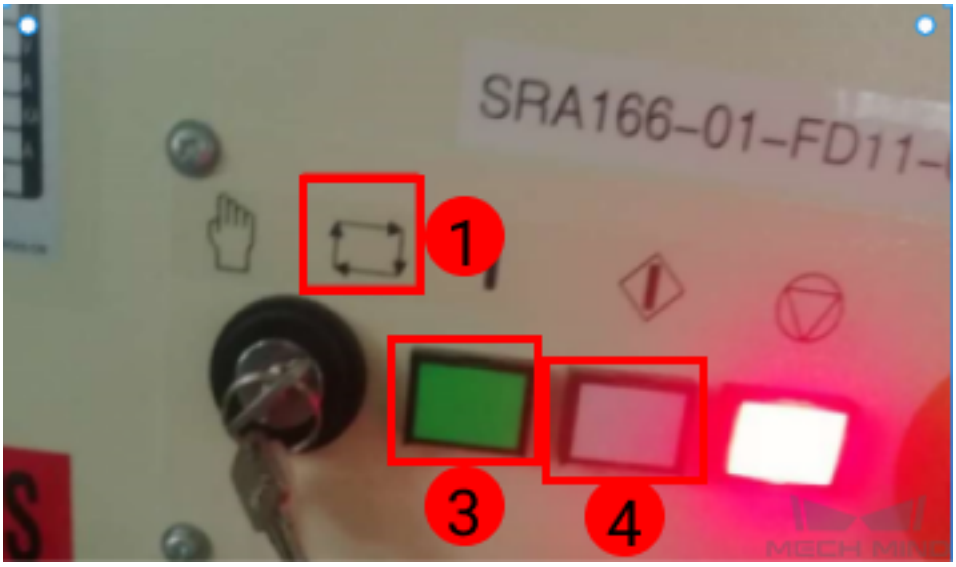


4. Go to *Program→ Designated Program*, enter **120** in the box, and then press  key. The designated program will appear in the Program panel and **User Task Monitor**.

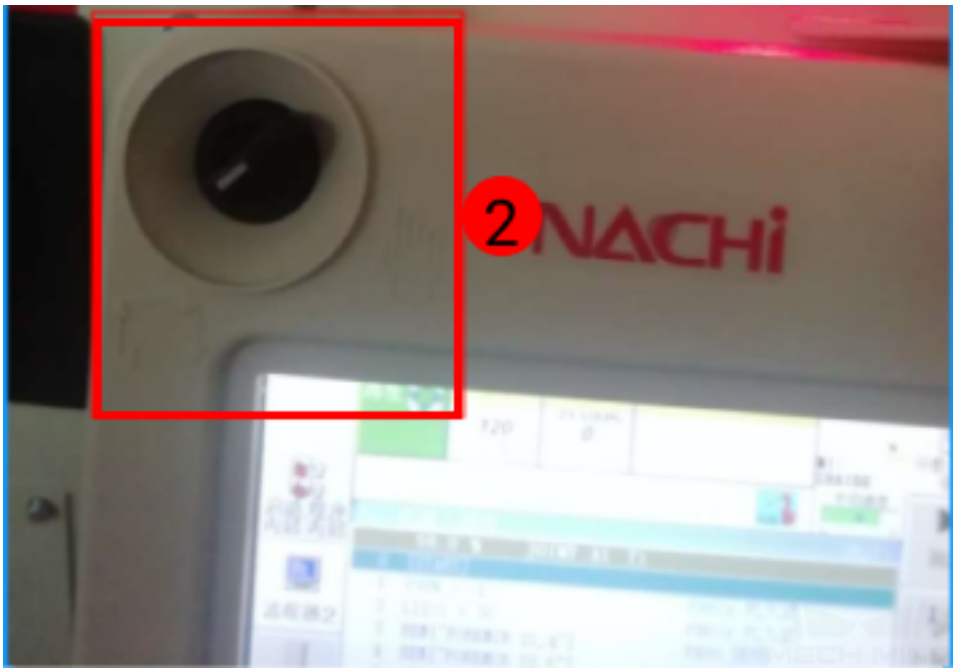


Start the robot

1. Turn the key on the controller and orientate it to the location .



2. Turn the selector switch as shown below. Press the green button and white button in turn to start the robot.



Hint: After running the full-control program successfully, you can open Mech-Center to connect the robot.

1.6.4 Test Robot Connection

Please refer to *Test Robot Connection* for detailed instructions.

1.7 AE Peitian Setup Instructions

This section introduces the process of loading the robot full-control program onto an AE Peitian robot.

The process consists of 5 steps:

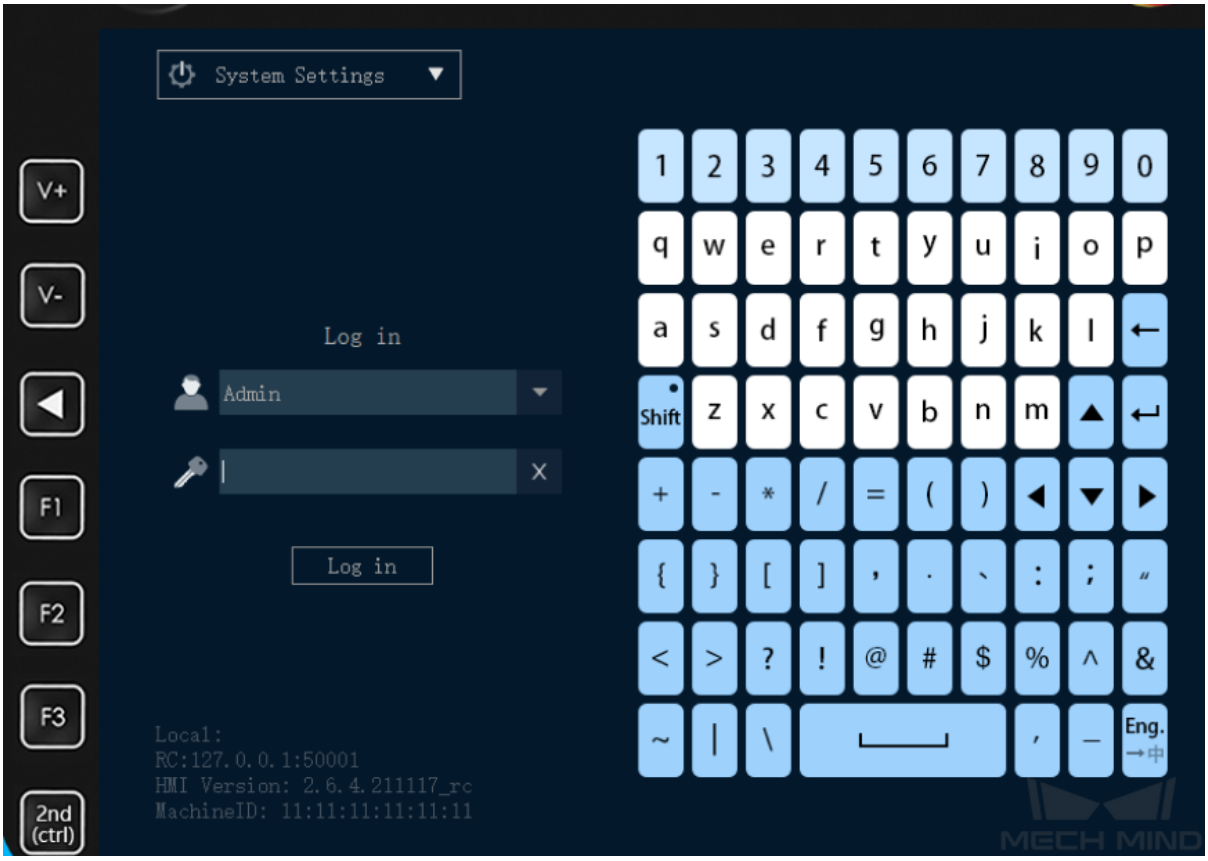
- *Log In*
- *Check IP and Controller Compatibility*
- *IP Configuration*
- *Load the Program File*
- *Test Robot Connection*

Please have a flash drive ready at hand.

1.7.1 Log In

Every time you open the teach pendant, you will need to log in with an account. The initial passwords are as shown below.

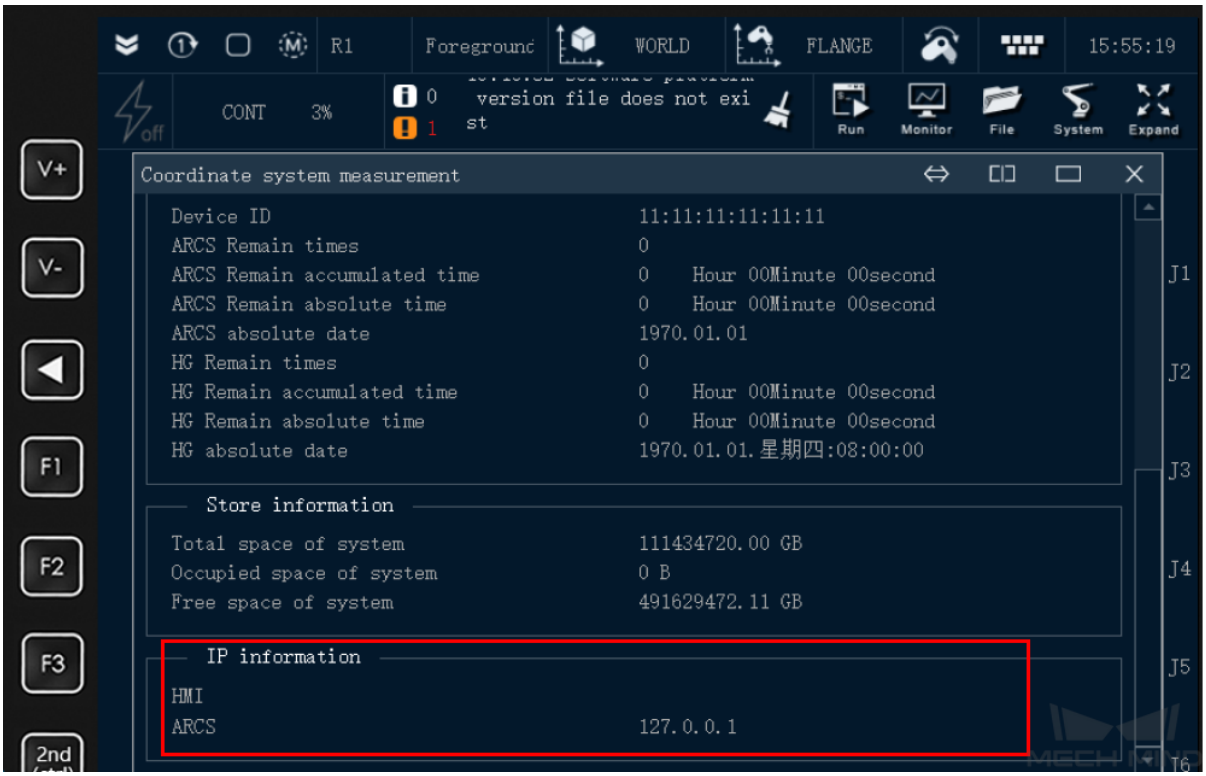
Account	Password
Teacher	PEACE
OEM	GRACE
Admin	OMNIPOTENT



1.7.2 Check IP and Controller Compatibility

1. Go to *System* → *System and Update* → *System Information*.

2. Now you can check the IP information in the window as shown below.

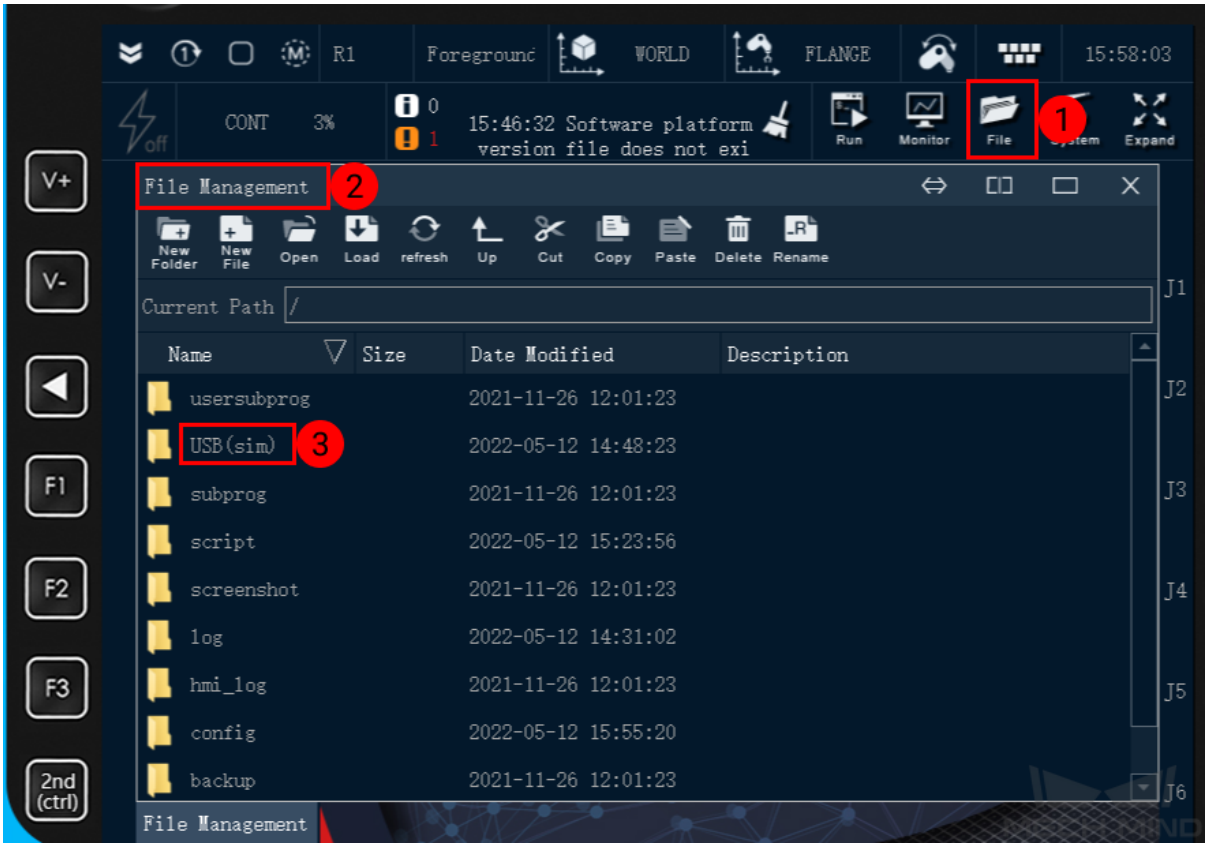


1.7.3 IP Configuration

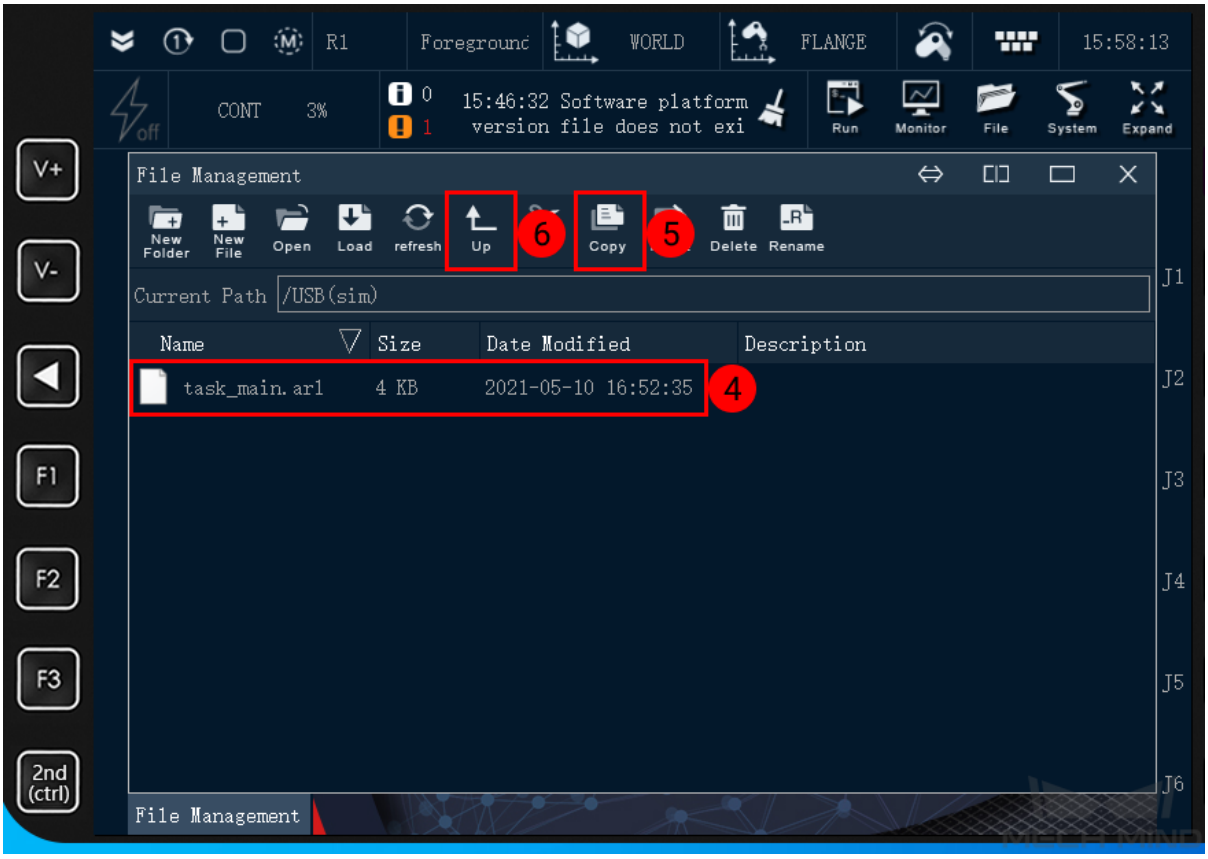
The IP address of an AE Peitian robot has been specified during programming, and you only need to select the IP adddress when loading the program files.

1.7.4 Load the Program File

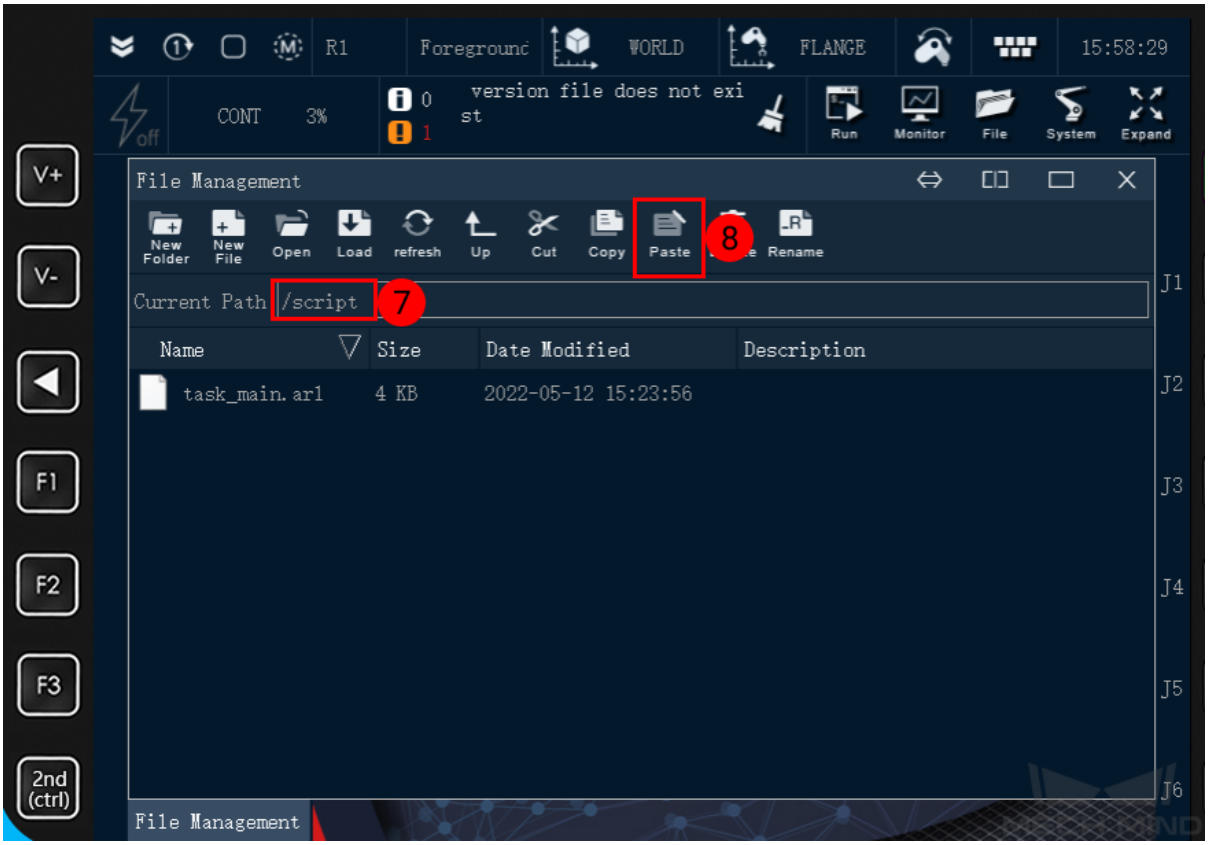
1. Go to the folder where Mech-Center is installed, and locate the full-control program file `task_main.arl`. The path of the file is: XXXX/Mech-Center/Mech_RobServ/install_packages/ae.
2. Use an USB flash drive to copy the program file `task_main.arl` and paste it into the `/script` folder of the robot system.
3. Go to *File* → *File Management*, select *USB(sim)* in the list and open the folder.



4. Select the file `task_main.arl`, and select *Copy*, and then select *Up* to go to the parent directory.

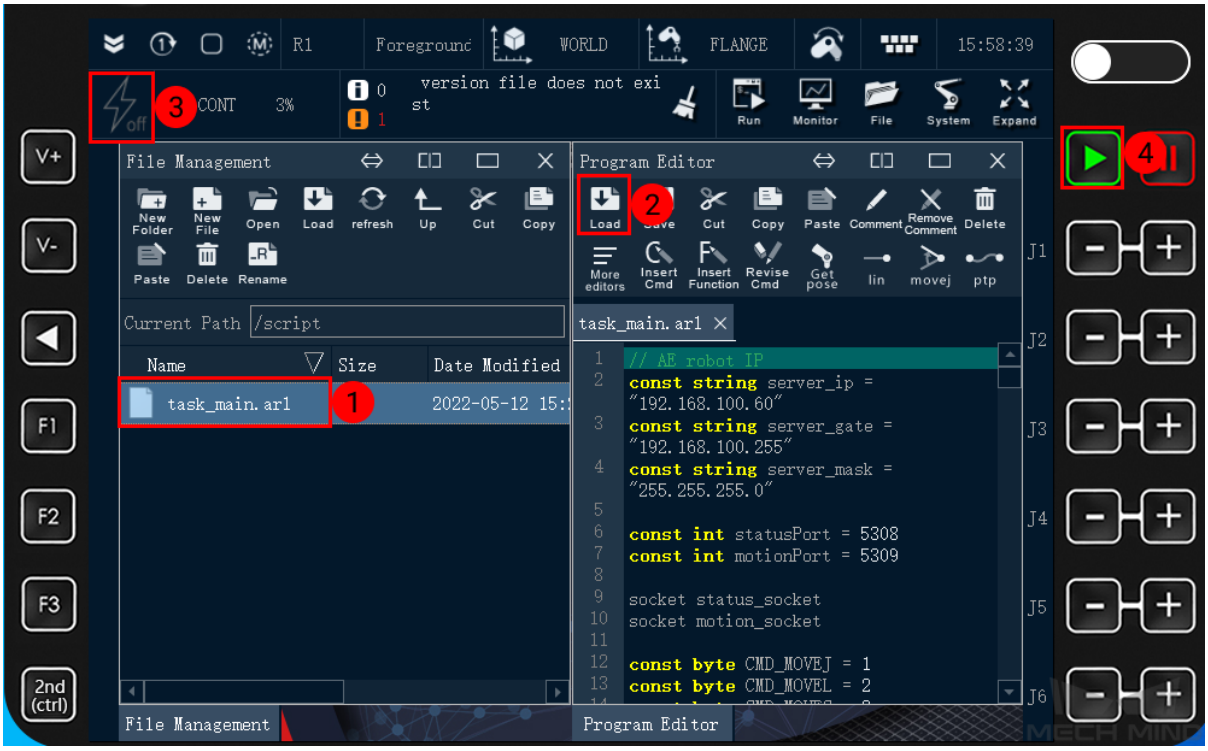


5. After opening the folder `/script`, select *Paste* to move the file `task_main.arl` into it.



1.7.5 Run the Program

1. Double click on `task_main.arl` to open the program file and then select *Load*.



2. Press on  , and then press  to run the program.

1.7.6 Test Robot Connection

Please refer to *Test Robot Connection* for detailed instructions.

1.8 HYUNDAI Setup Instructions

This section introduces the process of loading the robot full-control program onto a Hyundai robot.

The process consists of 5 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program Files*
- *Further Configuration*
- *Connect to the Robot*

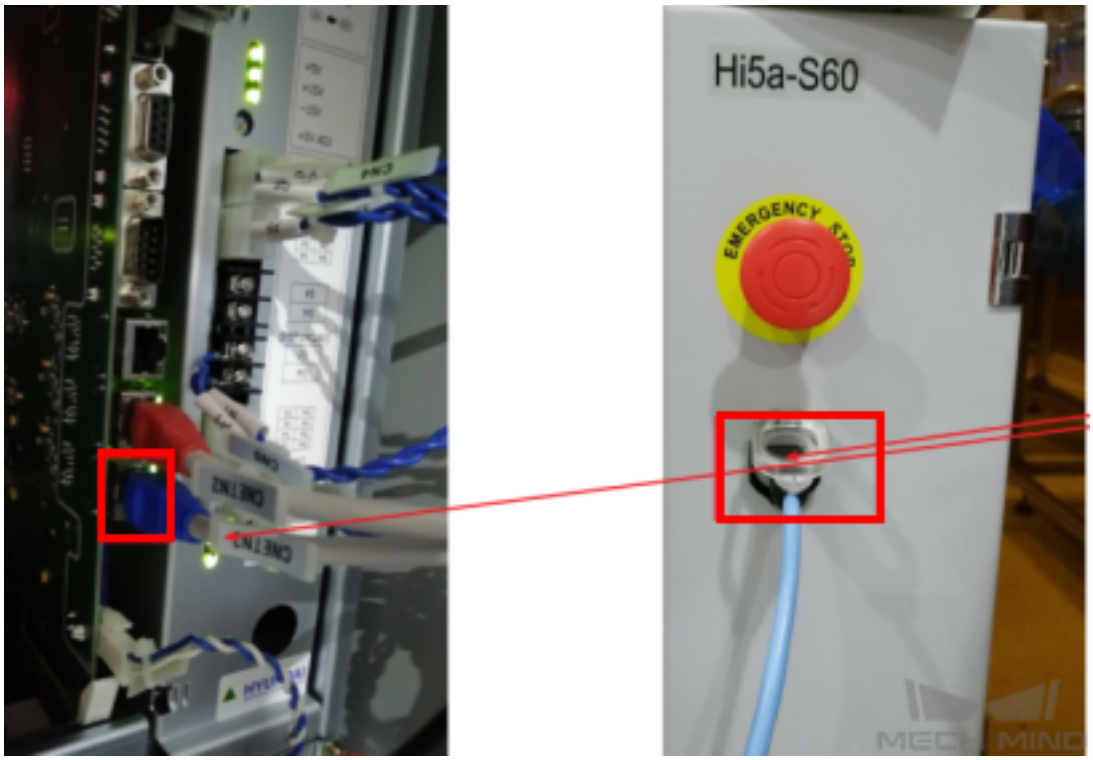
1.8.1 Check Controller and Software Compatibility

- There is no requirement on the version of robot controller.

1.8.2 Setup the Network Connection

Hardware Connection

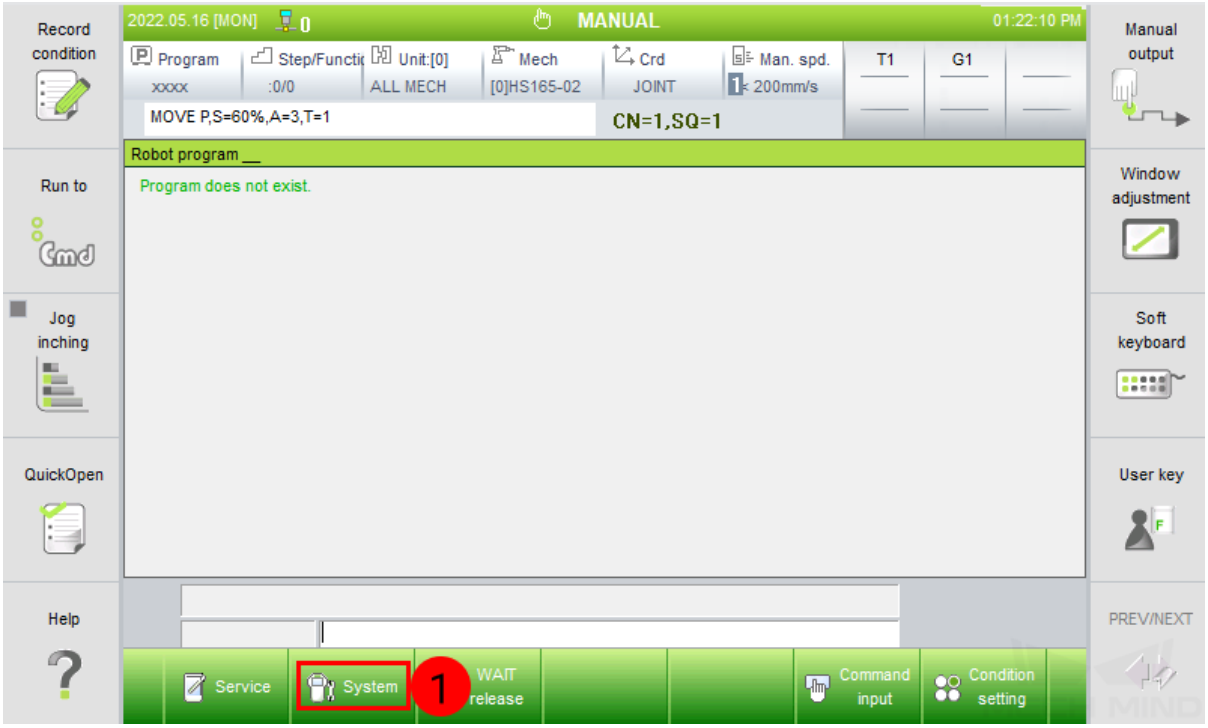
Plug the Ethernet cable of the IPC into the CNETN3 port inside the controller or the Ethernet port on the outside, as shown below.

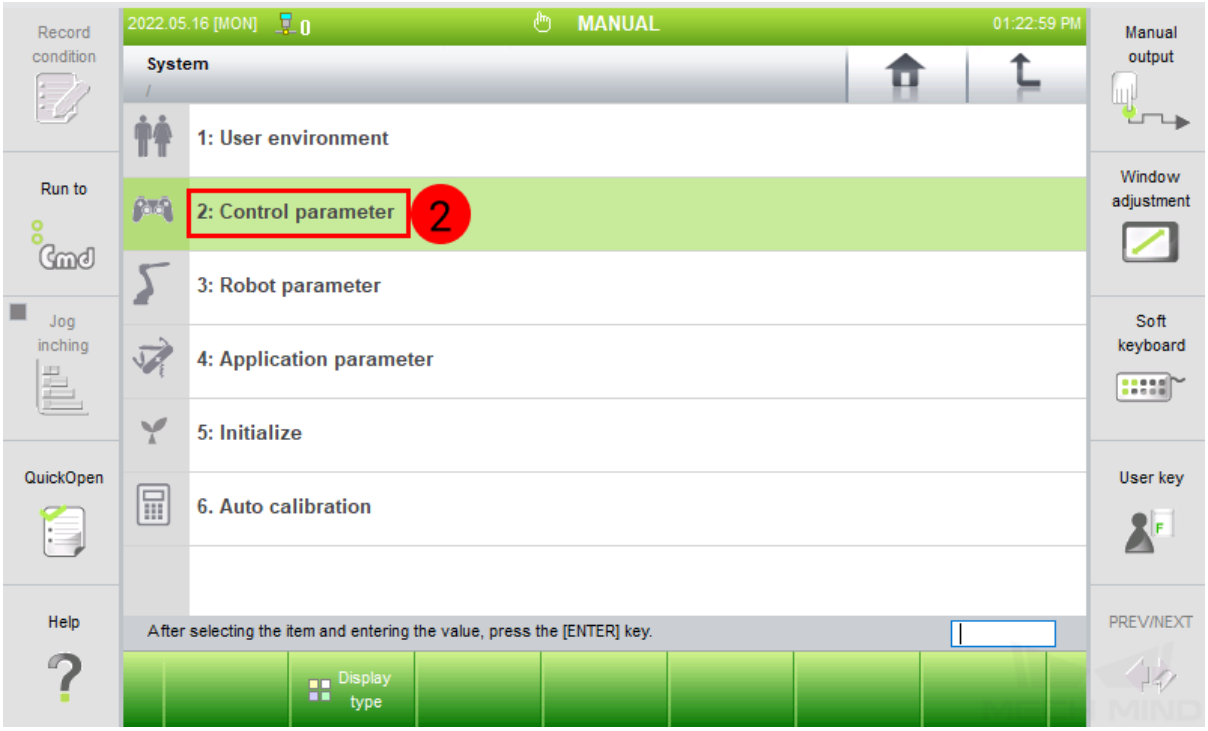


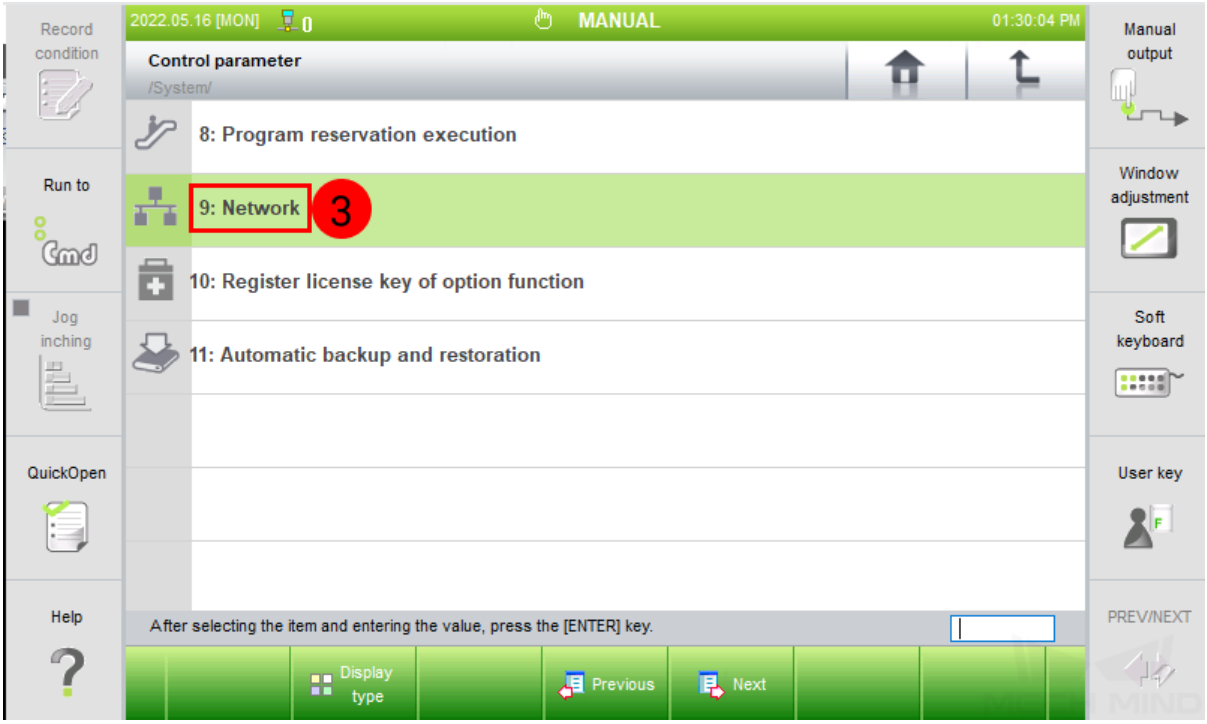
Hint: There are three port inside the controller, namely CNETN1, CNETN2, and CNETN3, which correspond to the EN0 address, TP address and EN2 User Ethernet address on the teach pendant respectively.

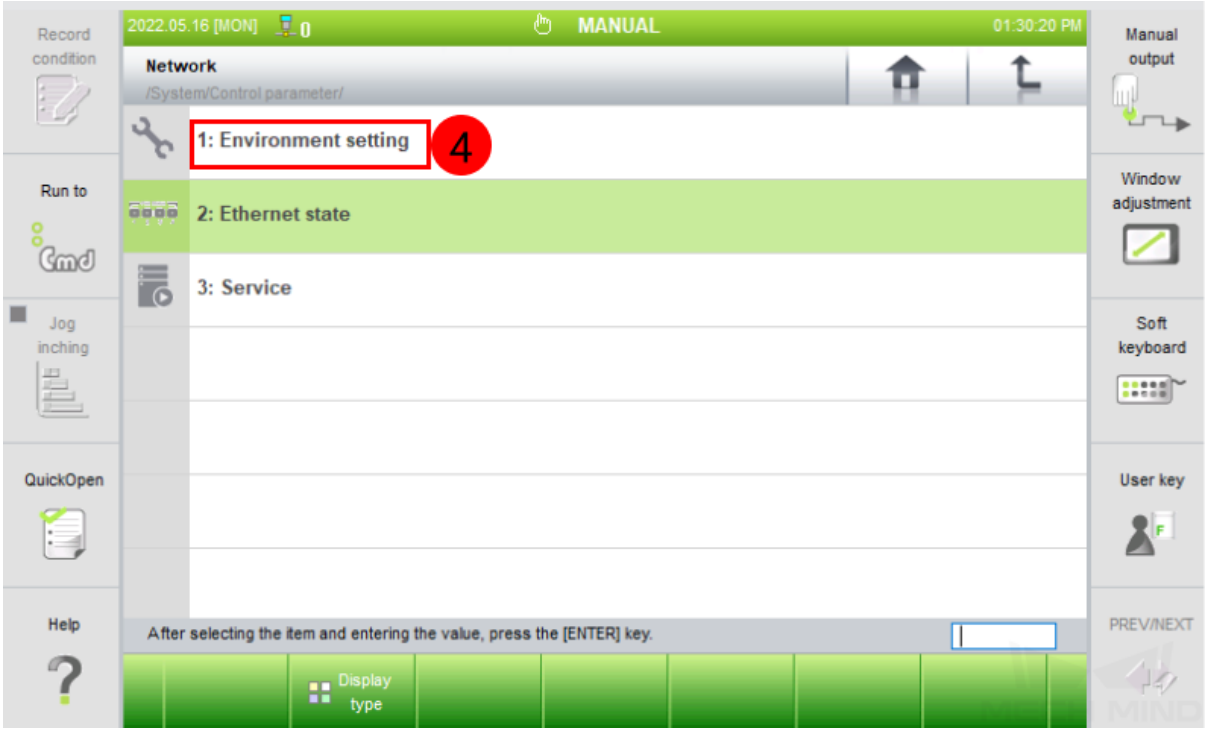
IP Configuration

1. Go to *System* → *Control parameter* → *Network* → *Environment setting*.

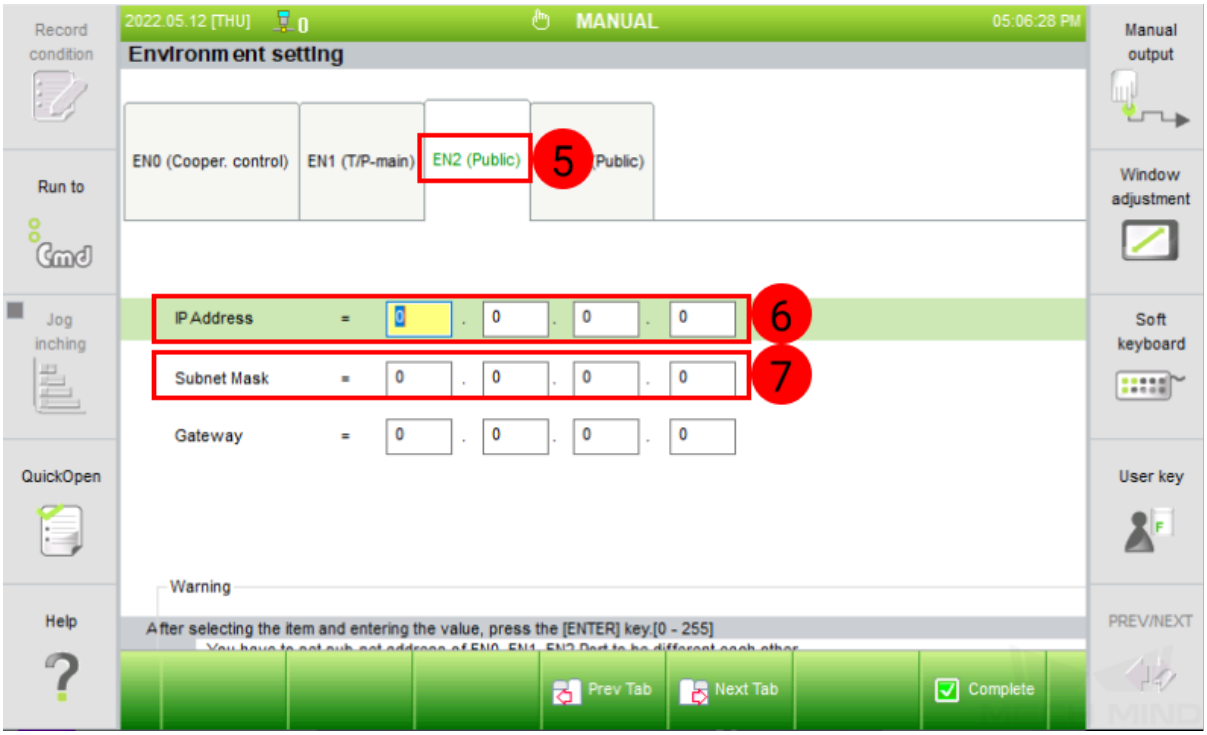








2. Select *EN2(Public)*, and set the right **IP Address**. Please also make sure that the **Sebnet Mask** is set to **255.255.255.0**.



Hint:

- The robot IP should be in the same subnet as the IPC.
- The subnet mask of the IPC is the same as that of the robot, which is **255.255.255.0**.
- Restart the robot after modifying the IP address.

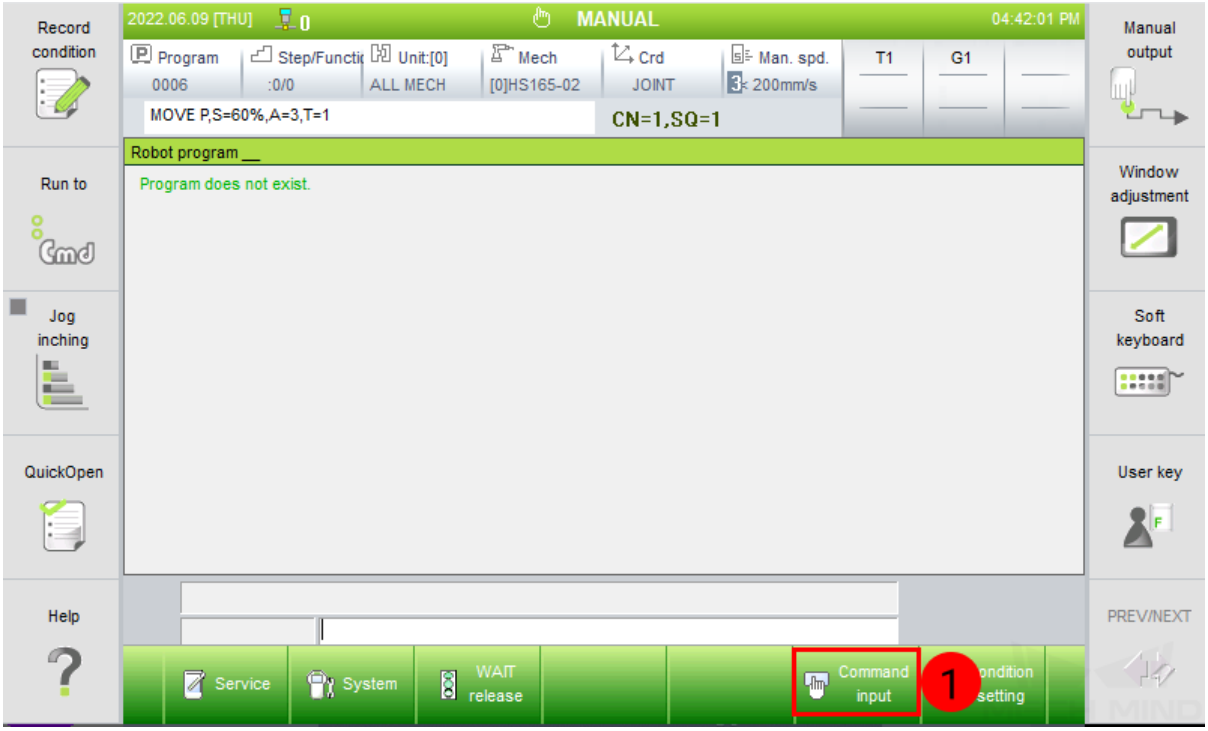
1.8.3 Load the Program Files

1. Connect the USB flash drive to the teach pendant.

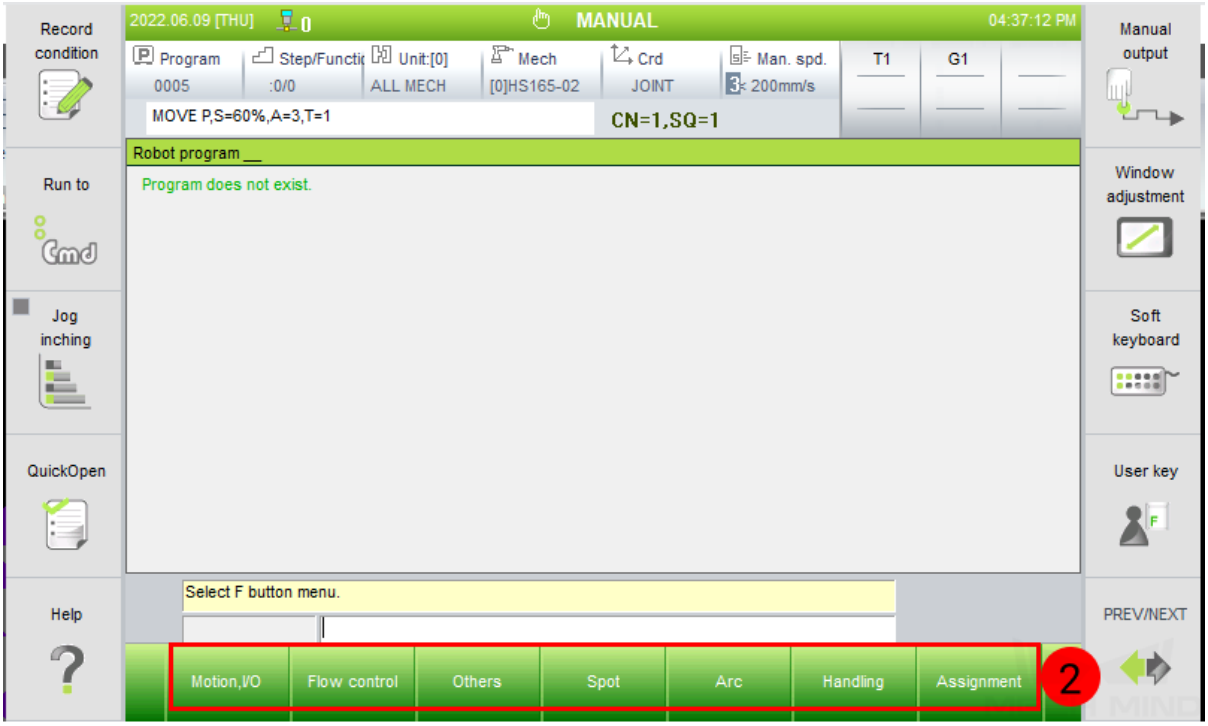
Note: To control a HYUNDAI robot, our full-control program **0101.JOB** and **0102.JOB** need to be initialized by changing the **Program File Format Version** informtion on the first line.

Hint: If you already have other available program on the robot, please directly copy and paste the program into the flash drive and skip to step 5.

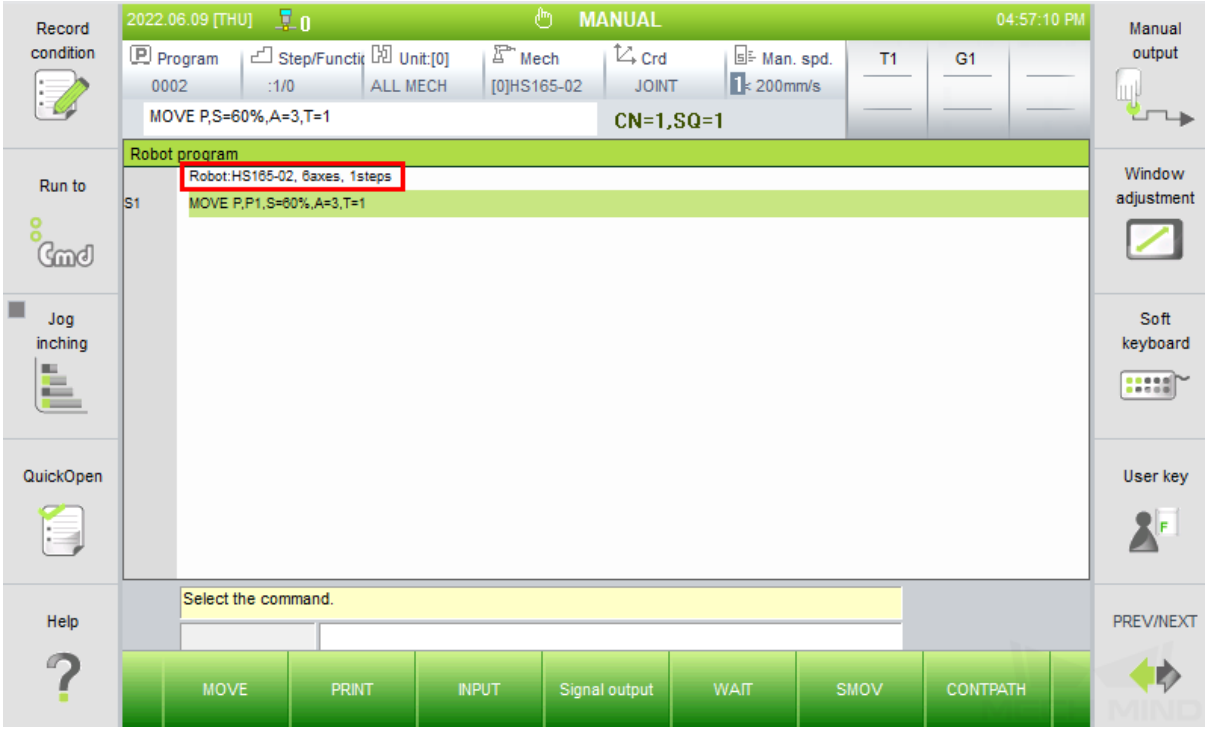
2. Select *Command input*.



3. Choose either of the command to input, as shown below.



4. Now you can see the robot version information on the top. Then save the program.



5. Select the newly created program file (or a previous program file), and select *Copy*. Then switch to the USB folder and select *Paste*.
6. Connect the flash drive to the IPC. Open the program, and then copy the first line.
7. Copy and paste the full-control program files **0101.JOB** and **0102.JOBO** into the flash drive. Open the two full-control program and paste the copied code to replace the first line in the program, and then save the changes.

```
Program File Format Version : 1.6  MechType: 370(HS220-01)  TotalAxis: 6  AuxAxis: 0
    DIM liIdx AS Integer
    DIM liVel[200] AS Integer
    DIM limotionType[200] AS Integer
    FOR liIdx=1 TO 200
    liVel[liIdx]=0
```

Hint: The full-control program files are stored in *XXX/Mech-Center/Mech_RobServ/install_packages/hyundai/Hi5a-S*.

8. Connect the flash drive to the teach pendant. Go to *System* → *File manager* → *USB*, select **0101.JOB** and **0102.JOB**, and then select *Copy*. Then switch to **T/P**, select *Paste* to load the files to the robot.

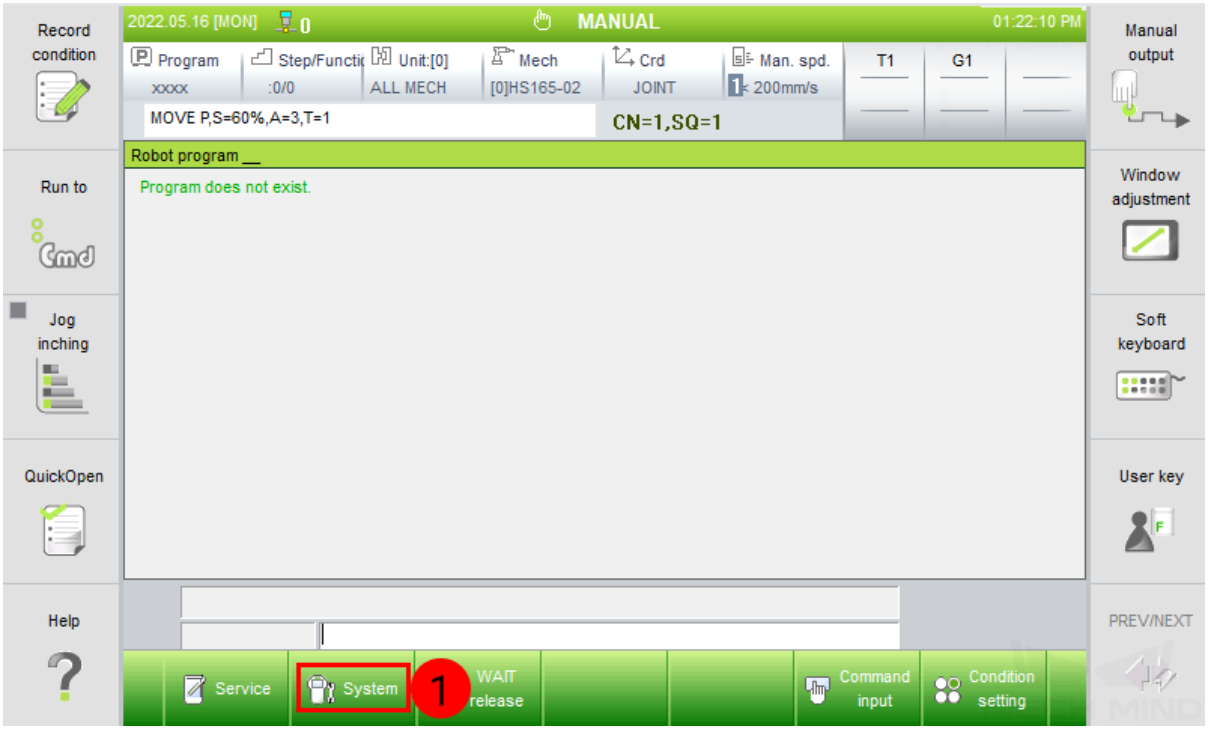
1.8.4 Further Configuration

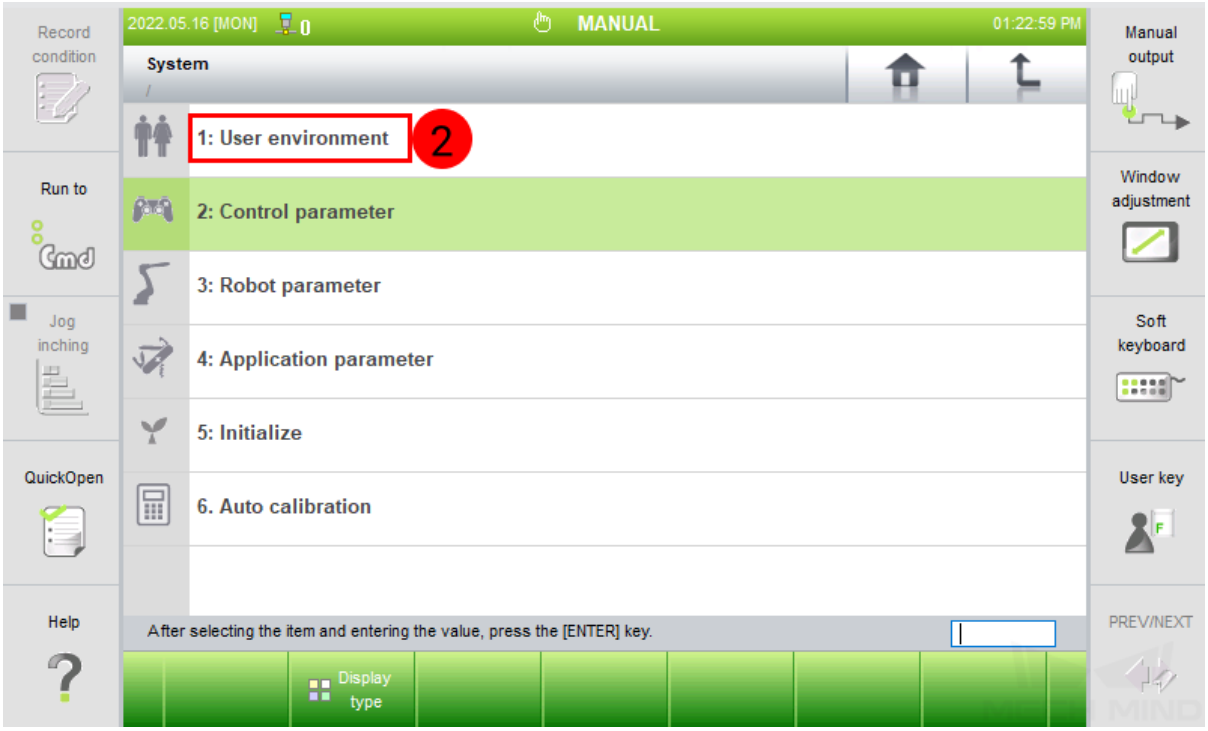
1. Change the IP address of the IPC to **192.168.0.150**.

Hint: This IP address is a default one which is specified in **0101.JOB** and **0102.JOB**. If you need to change the IP address, please modify the IP address in the program accordingly, and the new IP address should be in the same subnet as that of the robot controller.

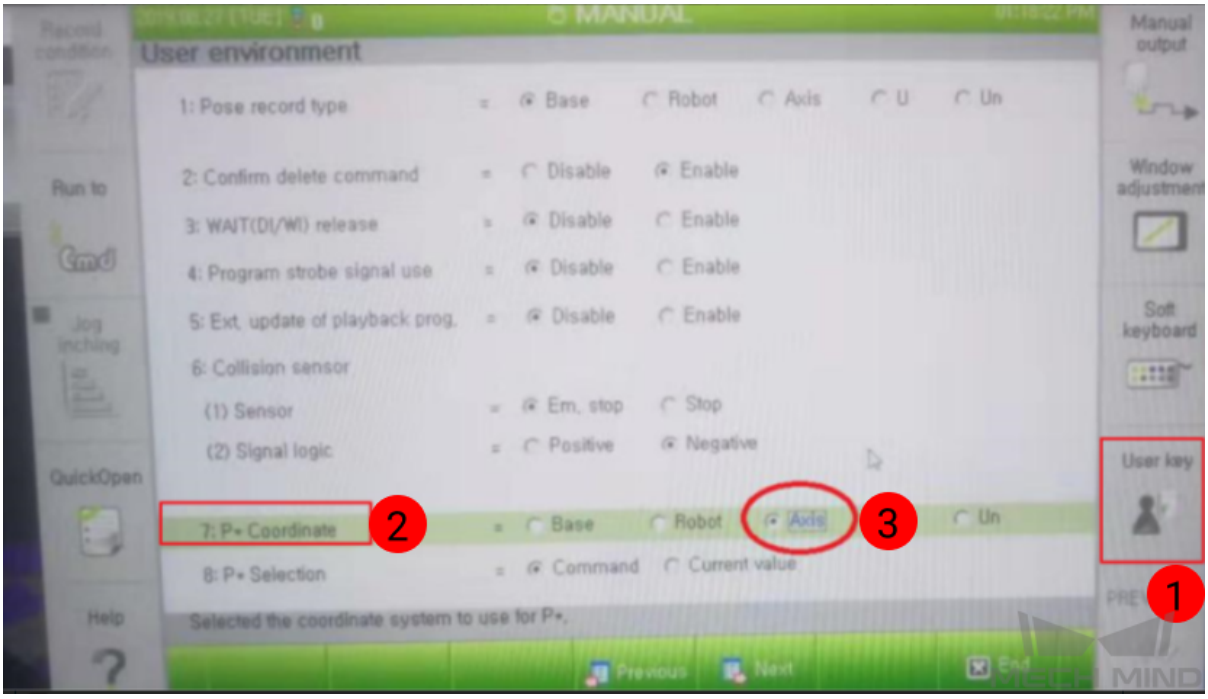
---

1. Set the management IP address of the router to **192.168.0.1**.
2. Go to *Syetem*→ *User Environment*.



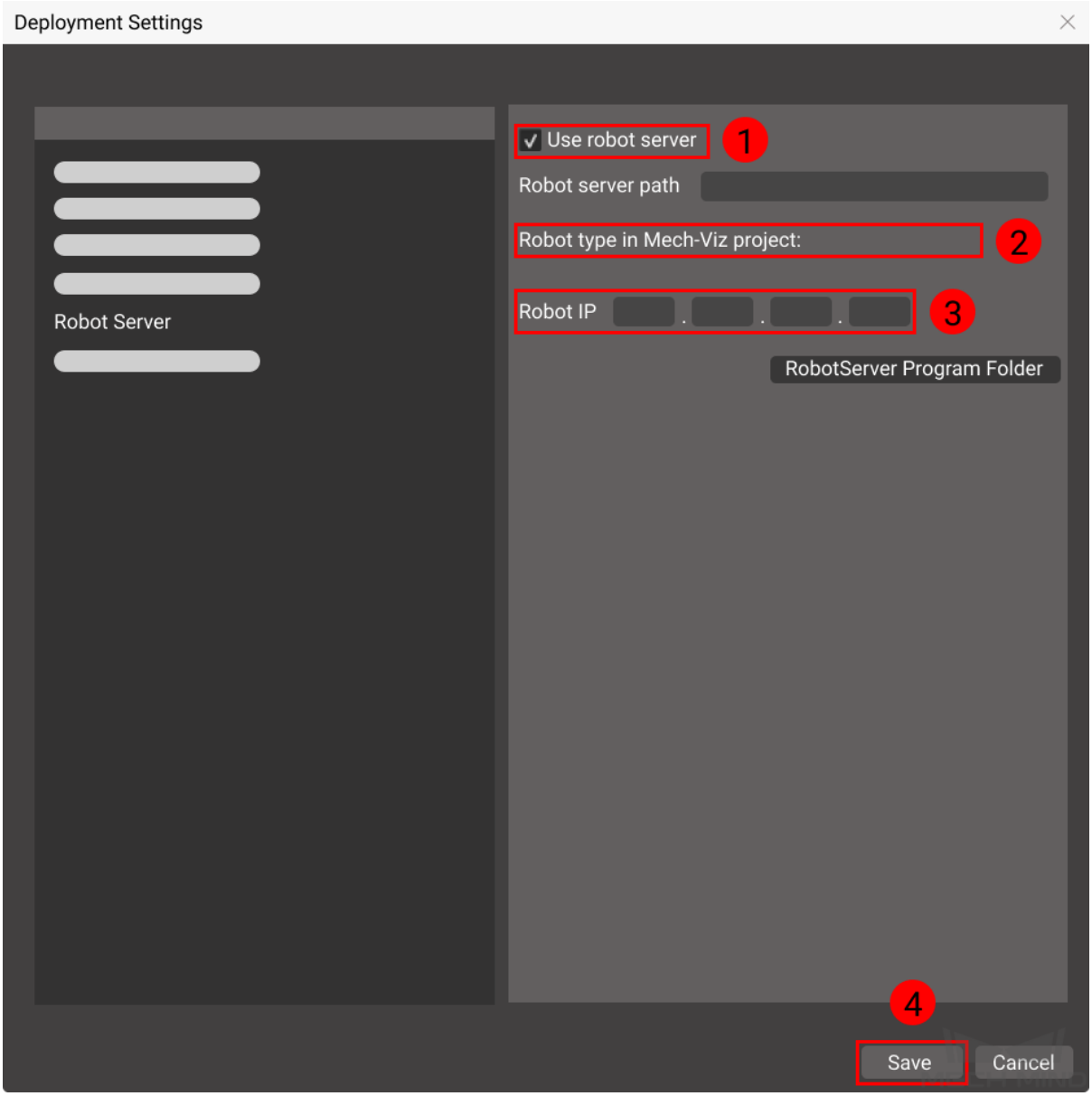


3. Select *User Key* and enter the general password **314** to request permission.
4. Change the **P* Coordinate** to *Axis*.



1.8.5 Connect to the Robot

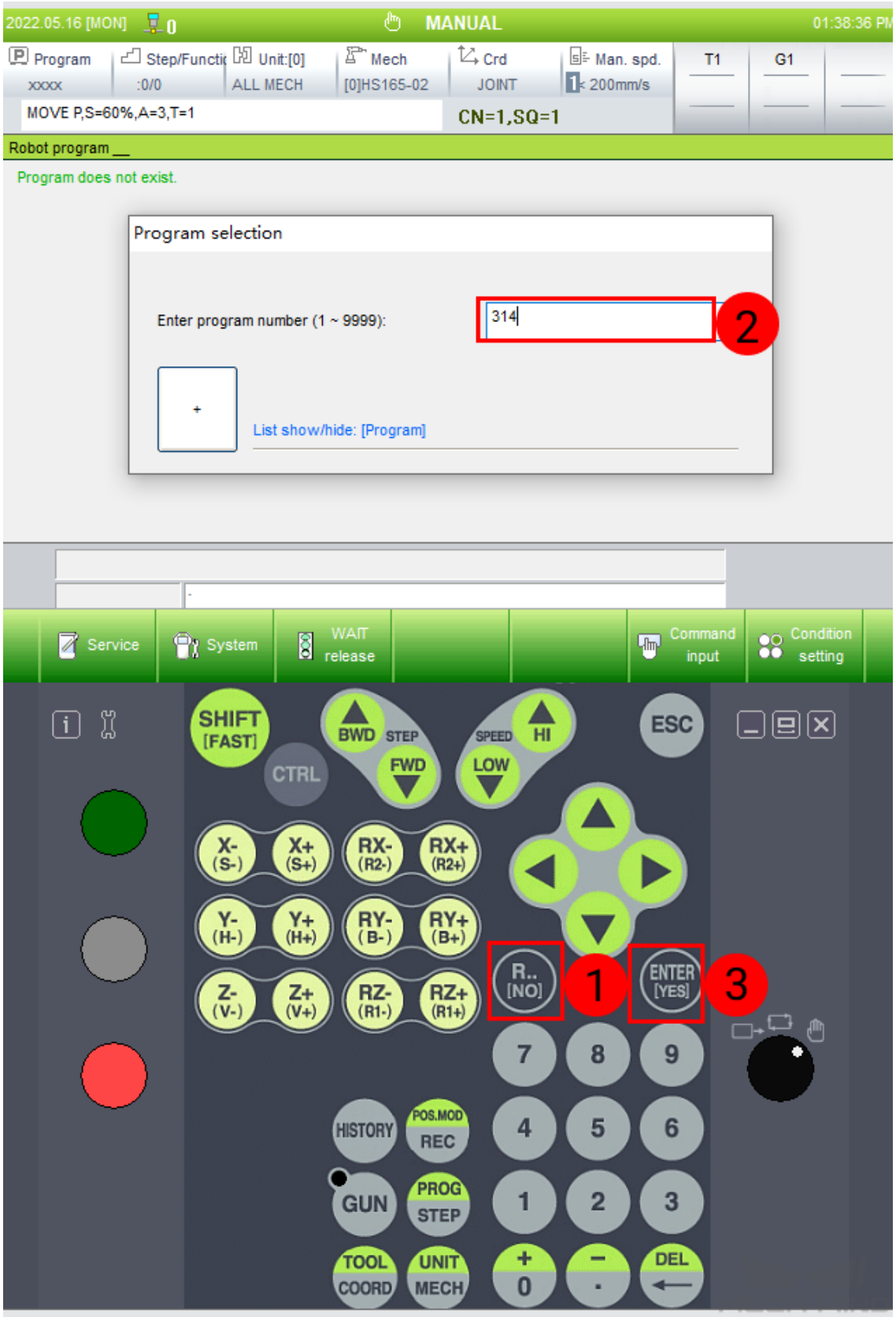
1. Open Mech-Center and click on *Deployment Settings*.
2. Go to **Robot Server**, and make sure **Use robot server** is checked.
3. Check if the robot model displayed after **Robot type in Mech-Viz project** matches the one in use.
4. Set the Robot IP address, and click on **Save**.



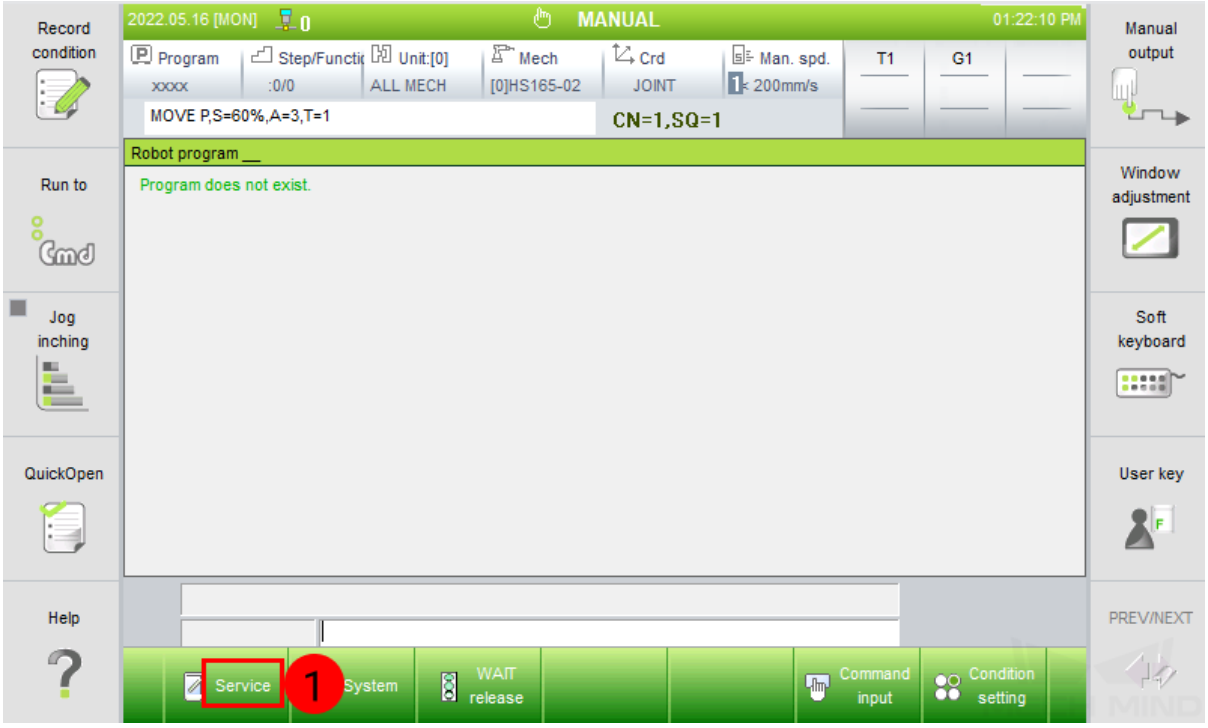
5. Click on *Connect Robot* in the Toolbar.
6. Switch the robot into AUTO mode.
7. Open **0101.JOB**, select *Program* → *Step/Function*, and enter **0** in the pop-up **Step selection** window. The way to reset the **0102.JOB** is the same.
8. Execute the program **0101.JOB**.
9. The robot is successfully connected if:
   - A message saying **Robot: server connected to the robot** shows up in the **Log** panel, and
   -  with the robot model shows up in the **Service Status** panel.

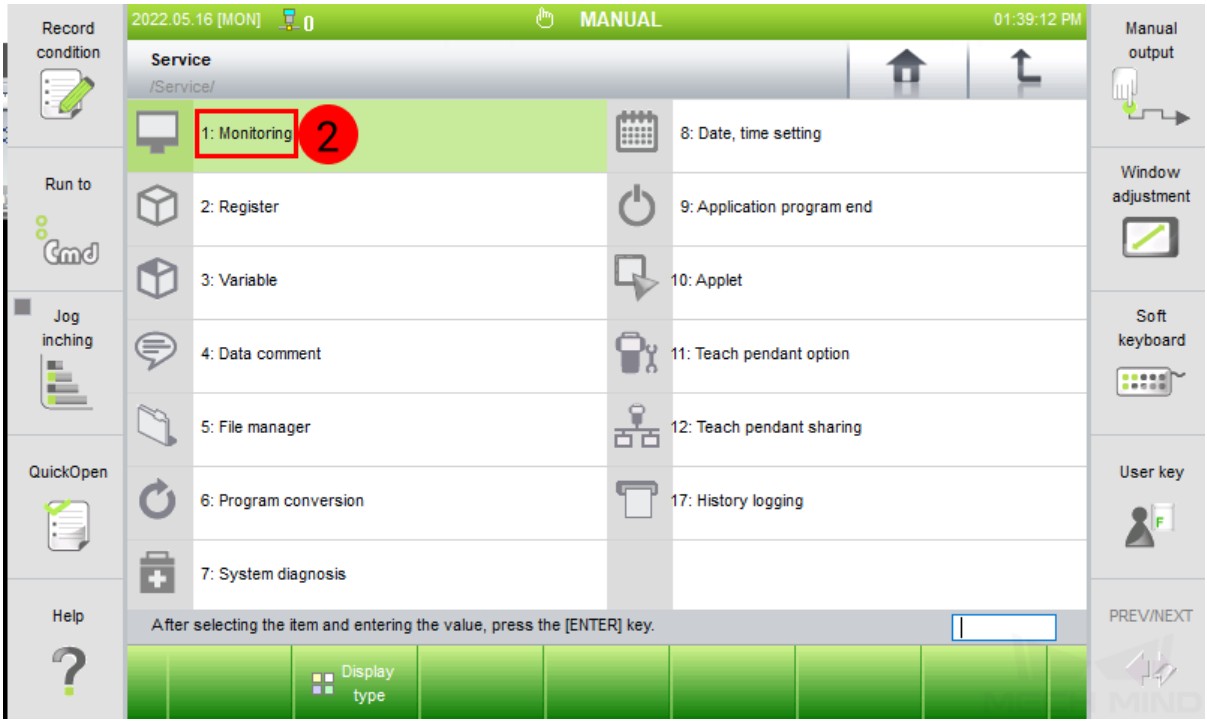
Reconnect the robot

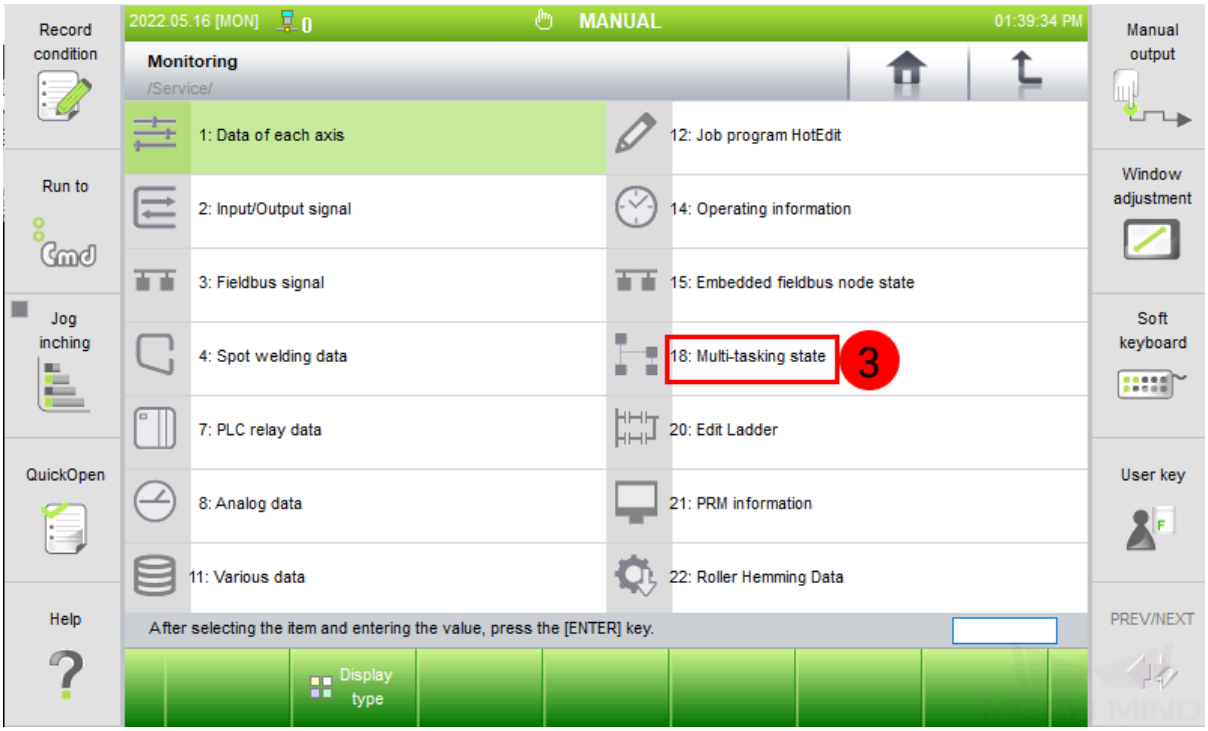
1. Under teach mode, press `R.. [NO]`, enter **314** and then press `ENTER [YES]` to request permission.


2022.05.16 [MON]
MANUAL
01:38:36 PM
Program
xxxx
Step/Functi
:0/0
Unit:[0]
ALL MECH
Mech
[0]HS165-02
Crd
JOINT
Man. spd.
200mm/s
T1
G1
MOVE P,S=60%,A=3,T=1
CN=1,SQ=1
Robot program __
Program does not exist.
Program selection
Enter program number (1 ~ 9999):
314
2
List show/hide: [Program]
Service
System
WAIT release
Command input
Condition setting
SHIFT [FAST]
CTRL
BWD
STEP
FWD
SPEED
HI
LOW
ESC
X- (S-)
X+ (S+)
RX- (R2-)
RX+ (R2+)
Y- (H-)
Y+ (H+)
RY- (B-)
RY+ (B+)
Z- (V-)
Z+ (V+)
RZ- (R1-)
RZ+ (R1+)
R.. [NO]
1
ENTER [YES]
3
7
8
9
4
5
6
1
2
3
HISTORY
POS.MOD
REC
GUN
PROG
STEP
TOOL
COORD
UNIT
MECH
+
0
−
.
DEL

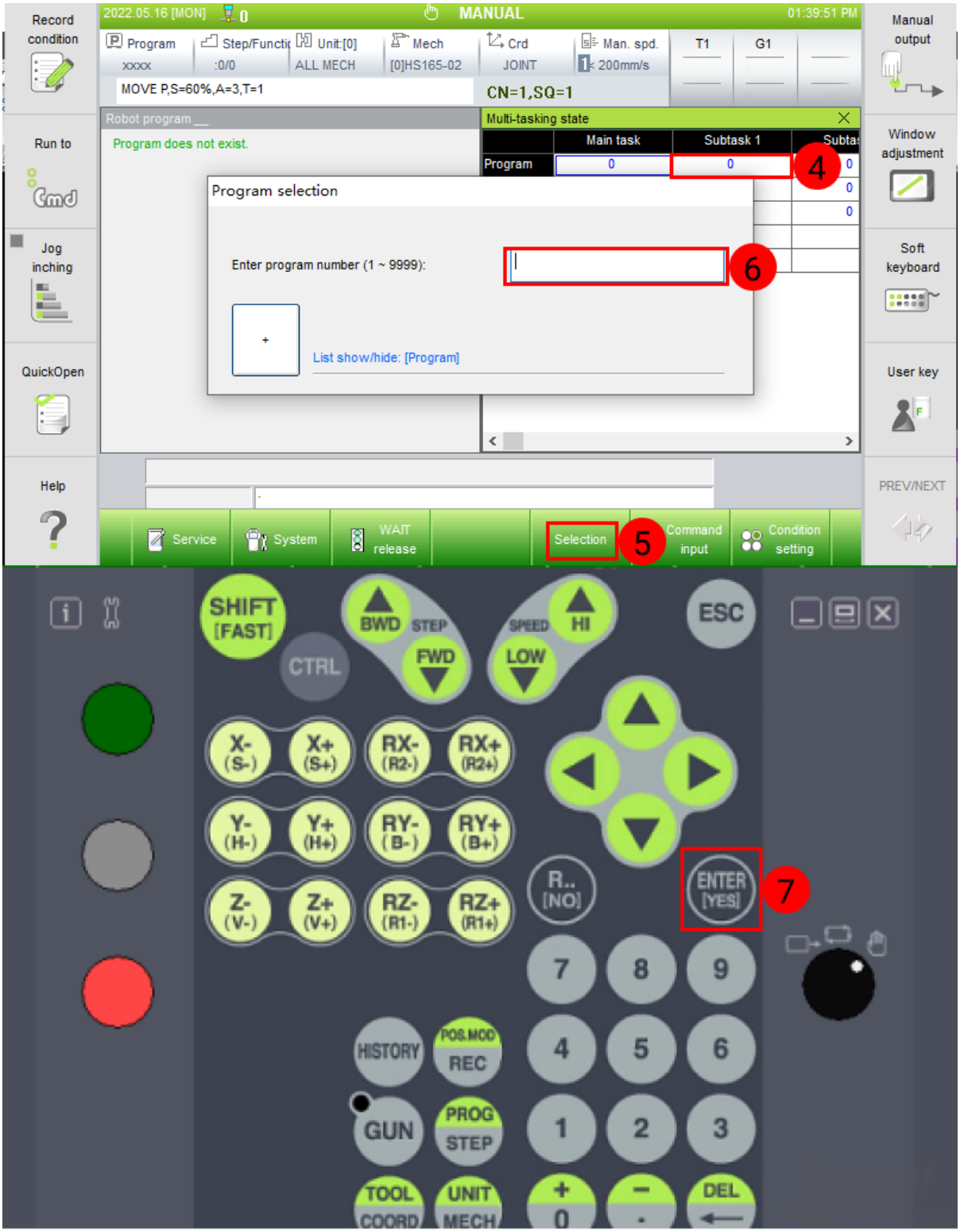
2. Each time when reconnect the robot, the background tasks should be stopped and the pstep in the foreground program should be reset to 0. Select *Service→ Monitoring → Multi-tasking_state → ENTER [YES]* to enter the multi-tasking interface.







3. Select Program (Suntask) and then select *Selection*. Enter **3** in the **Program selection** window and then press `ENTER [YES]` to complete configuration.


Record condition
2022.05.16 [MON]
MANUAL
01:39:51 PM
Manual output
Program
Step/Functi
Unit:[0]
Mech
Crd
Man. spd.
T1
G1
xxxx
:0/0
ALL MECH
[0]HS165-02
JOINT
200mm/s
MOVE P,S=60%,A=3,T=1
CN=1,SQ=1
Run to
Robot program
Program does not exist.
Multi-tasking state
Main task
Subtask 1
Window adjustment
Program
4
Program selection
Enter program number (1 ~ 9999):
6
Jog inching
Soft keyboard
List show/hide: [Program]
QuickOpen
User key
Help
PREV/NEXT
Service
System
WAIT release
Selection
5
Command input
Condition setting
SHIFT [FAST]
CTRL
BWD
STEP
FWD
SPEED
HI
LOW
ESC
X- (S-)
X+ (S+)
RX- (R2-)
RX+ (R2+)
Y- (H-)
Y+ (H+)
RY- (B-)
RY+ (B+)
Z- (V-)
Z+ (V+)
RZ- (R1-)
RZ+ (R1+)
R... [NO]
ENTER [YES]
7
HISTORY
POS.MOD REC
GUN
PROG STEP
TOOL COORD
UNIT MECH
DEL

1.9 UR Setup Instructions

This section introduces the process of setting up full control of an UR robot.

The UR robots can be controlled at script level. At the script level, the **URScript** is the programming language that controls the robot. Therefore, the IPC controls an UR robot by sending scripts to it.

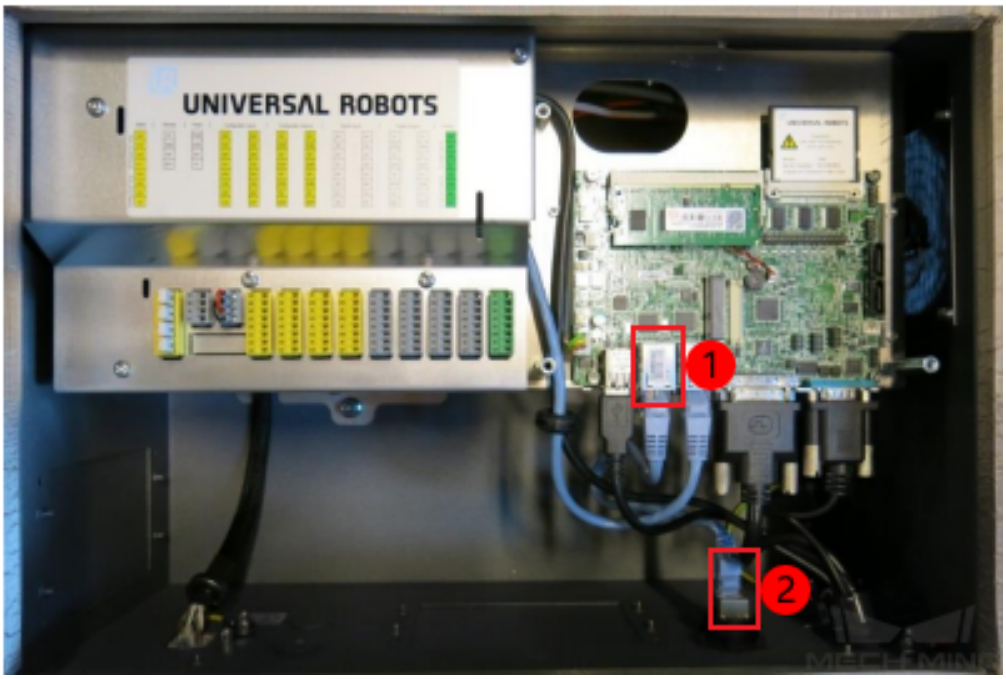
Attention:

- Once received a script, the robot will execute it. If the script has not finished executing, and the robot received another script which contains motion commands, the robot will stop immediately and execute the current script. If a script function defined by "sec" (not "def") is sent via the port 30002, and the function does not contain motion commands, the robot will not stop immediately.
- For e-Series robots such as UR5e, Mech-Viz can only guide the robot to move when they are set to remote control, or else Mech-Viz can only synchronize the robot instead of moving it.

1.9.1 Setup the Network Connection

Hardware Connection

Start the robot and plug the Ethernet cable into the port of the robot controller, as shown below.

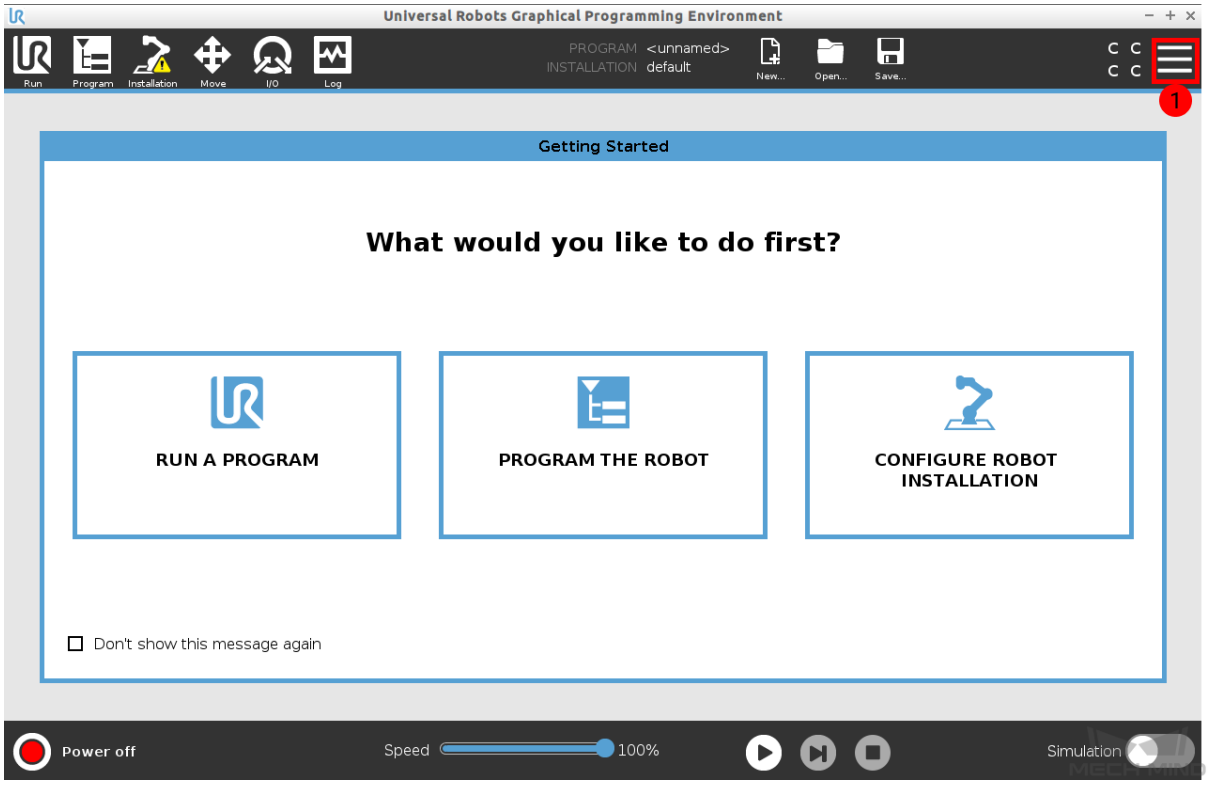


is the network port inside the controller, and is the default network port.

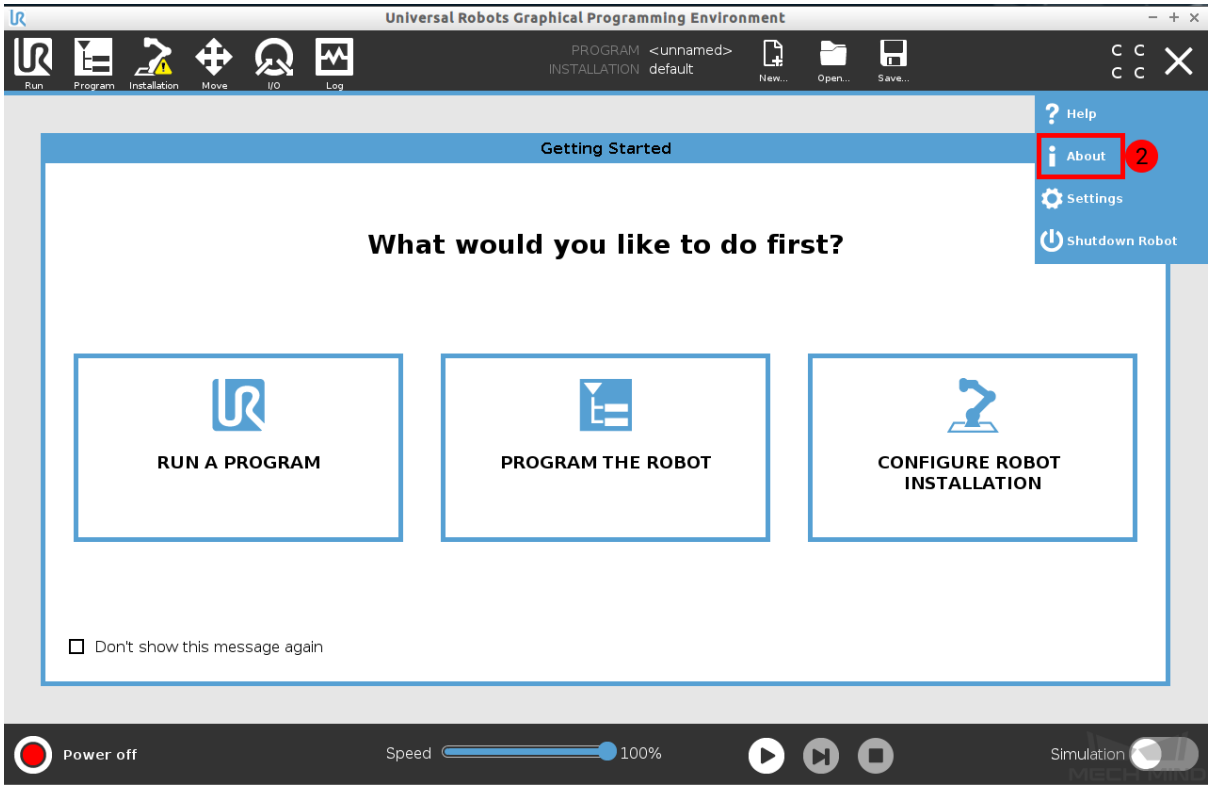
Tip: In order to make the network connection faster and more stable, it is recommended to use port .

1.9.2 Check IP and Controller Compatibility

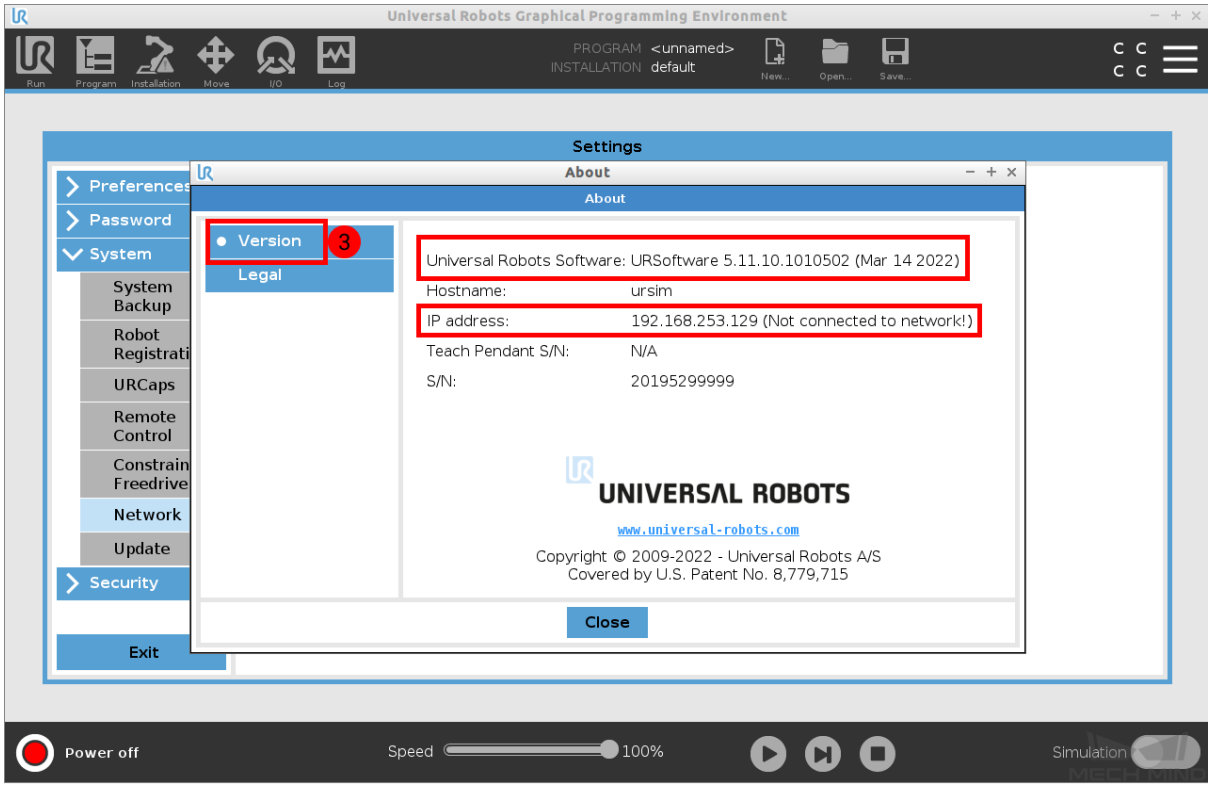
1. Press on the icon in the upper-right corner of the teach pendant.



2. Select *About*.

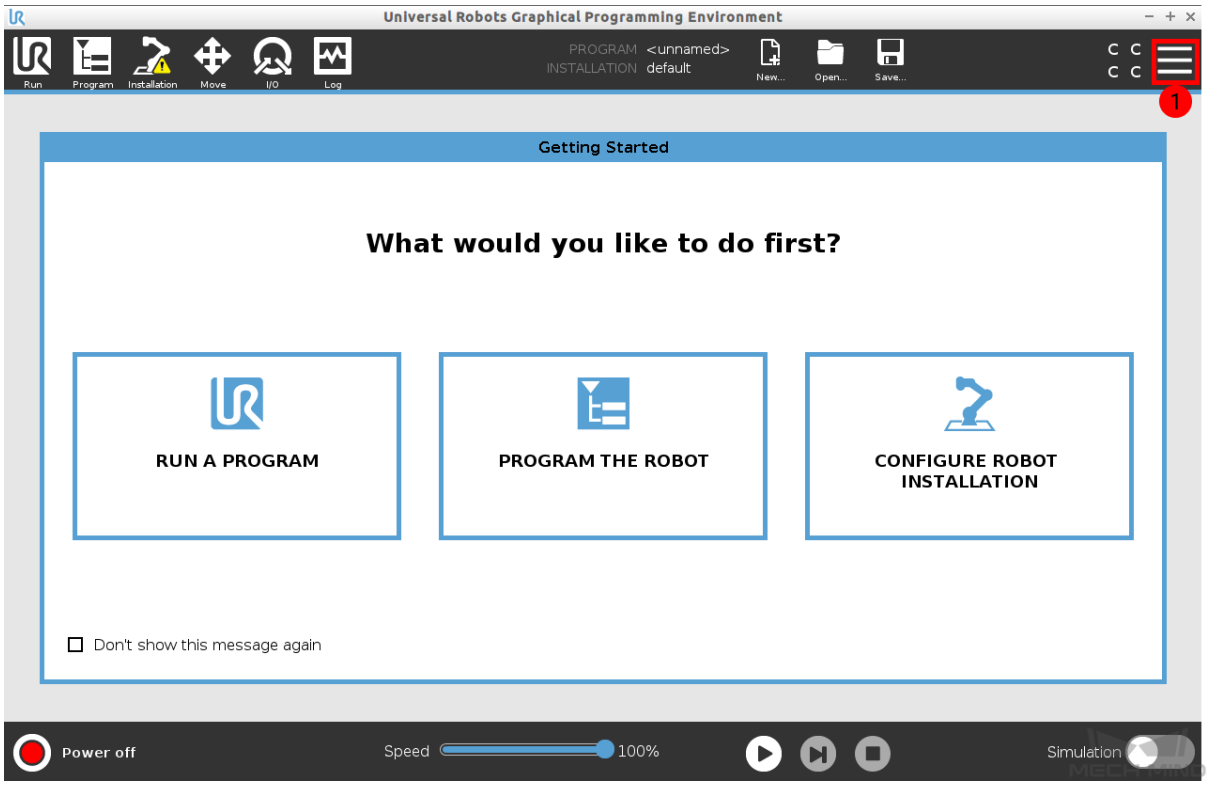


3. Press on *Version*.

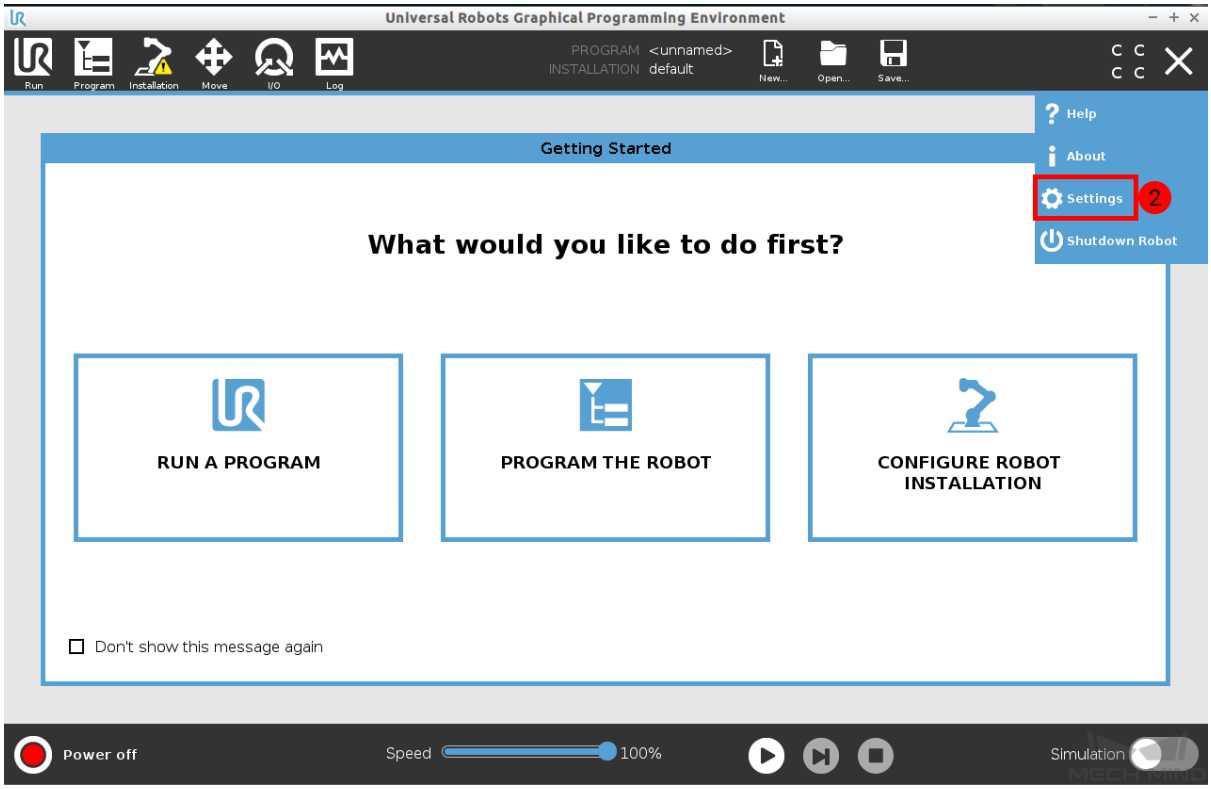


1.9.3 IP Configuration

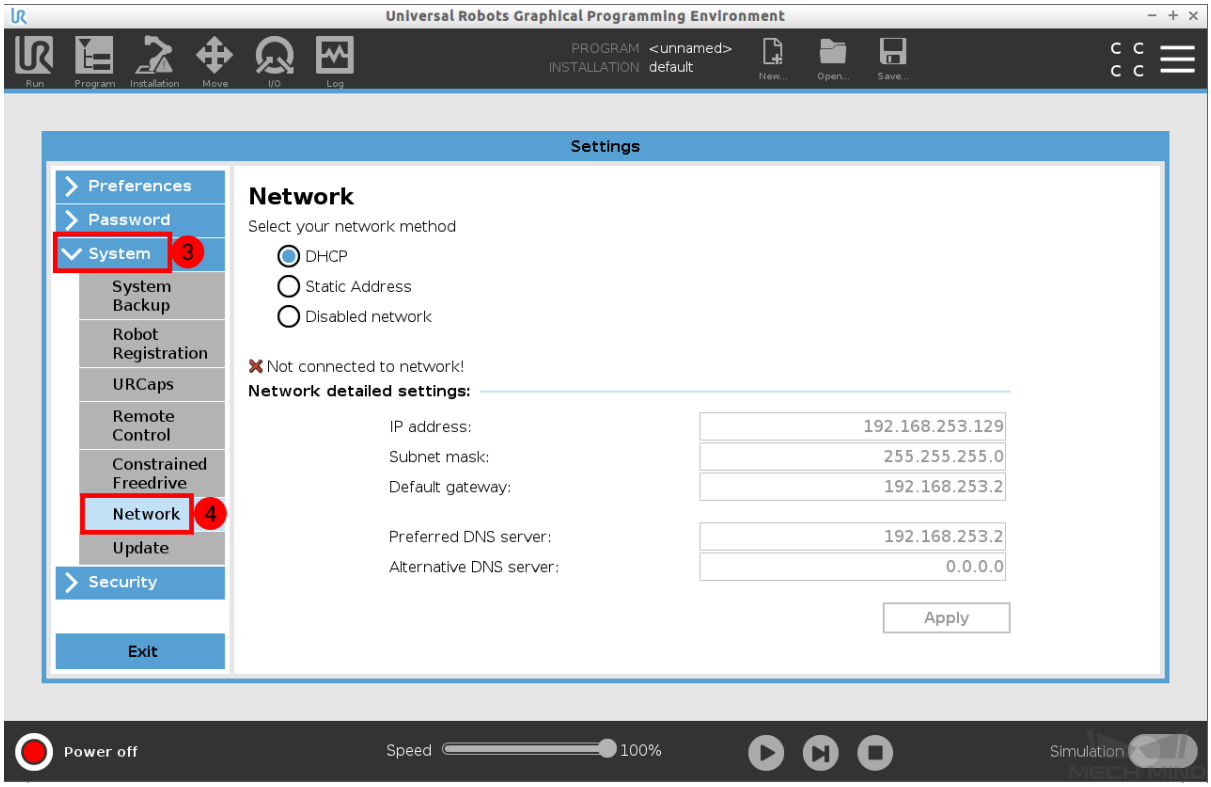
1. Press on the icon in the upper-right corner of the teach pendant.



2. Select *Settings*.



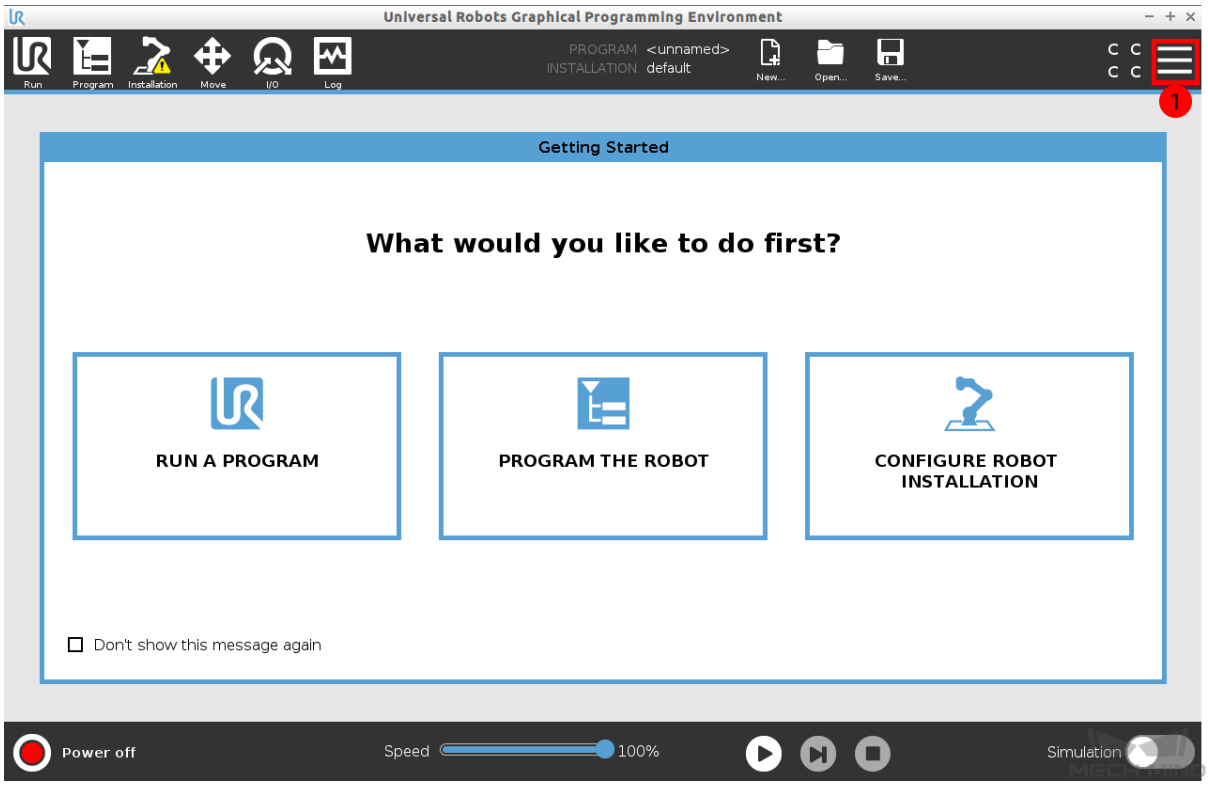
3. Select *Syetem* → *Network*.



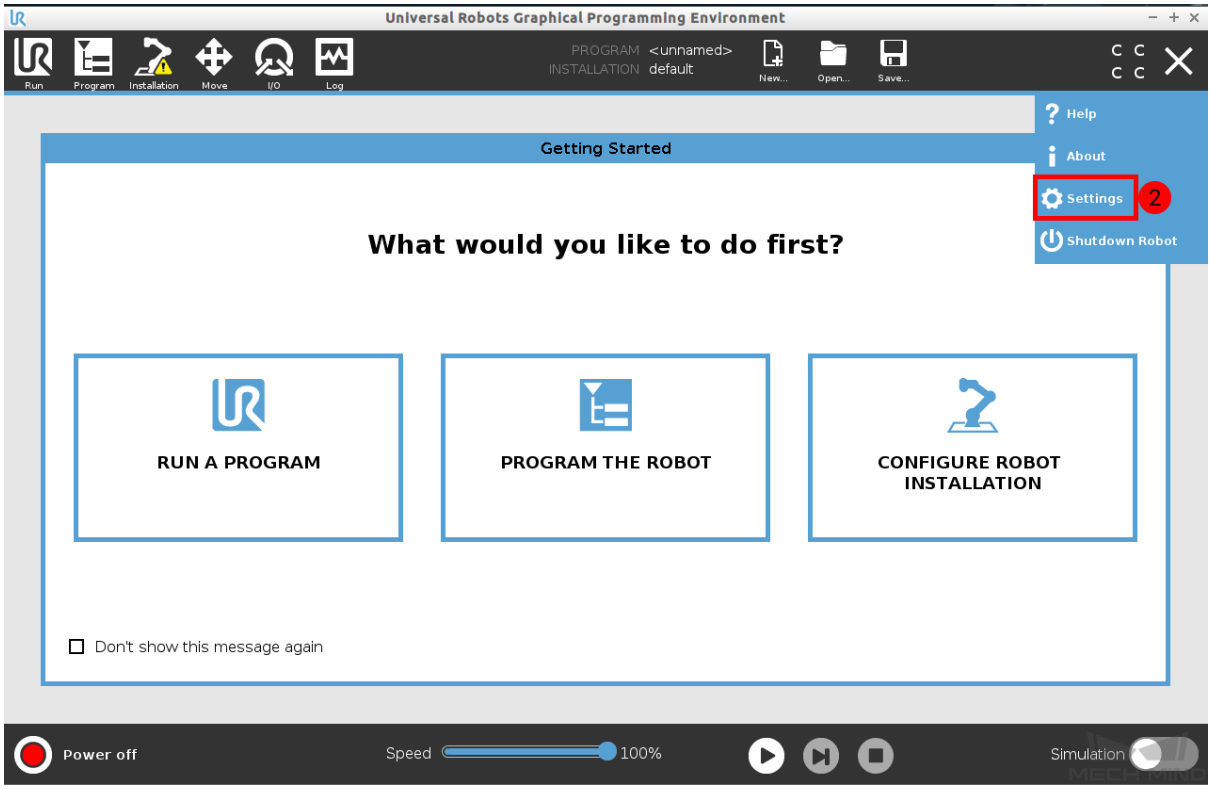
Enter the correnct IP address, which should be in the same subnet as the IPC. After configuration, press on *Apply*.

1.9.4 Start Remote Control

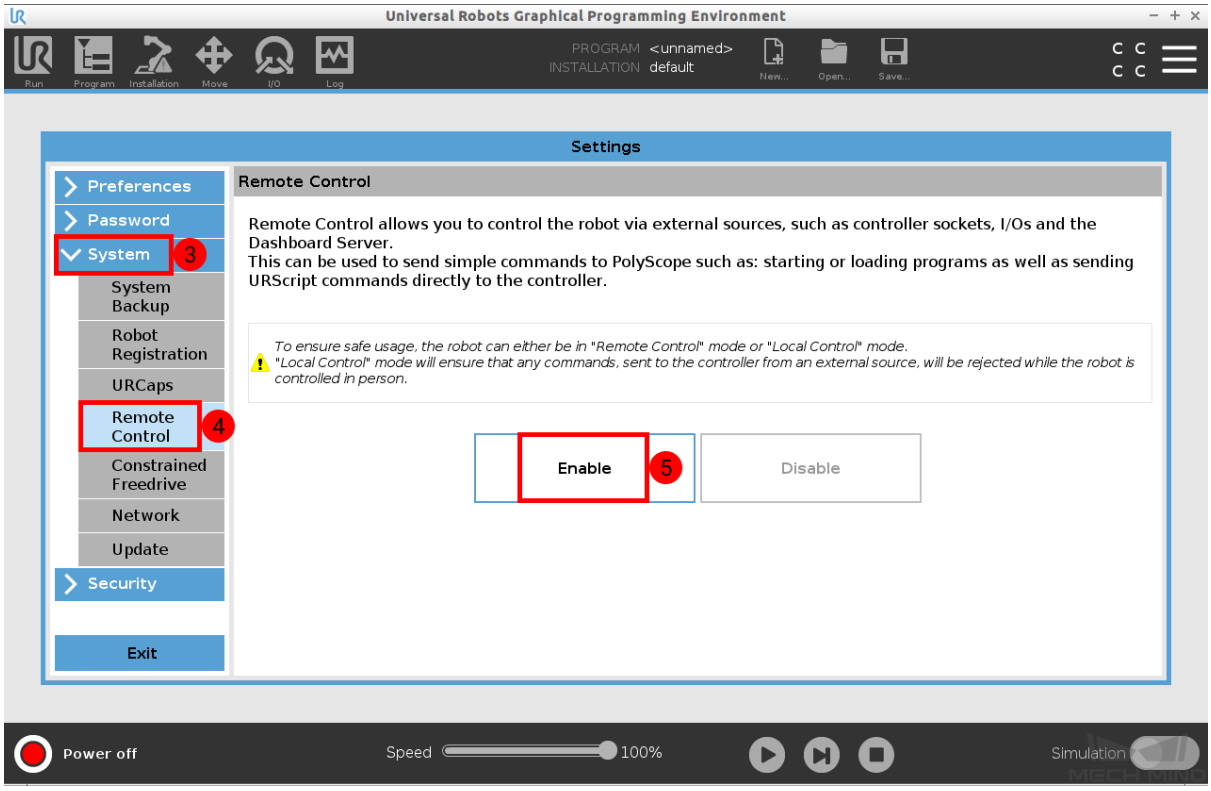
1. Press on the icon in the upper-right corner of the teach pendant.



2. Select *Settings*.



3. Select *Syetem* → *Remote Control*.



4. Press on *Enable* to enable the remote control.

1.9.5 Test Robot Connection

Please refer to *Test Robot Connection* for detailed instructions.

1.10 ROKAE Xmate 7 Collaborative Robot Setup Instructions

This section introduces the process of setting up full control of a ROKAE Xmate 7 collaborative robot.

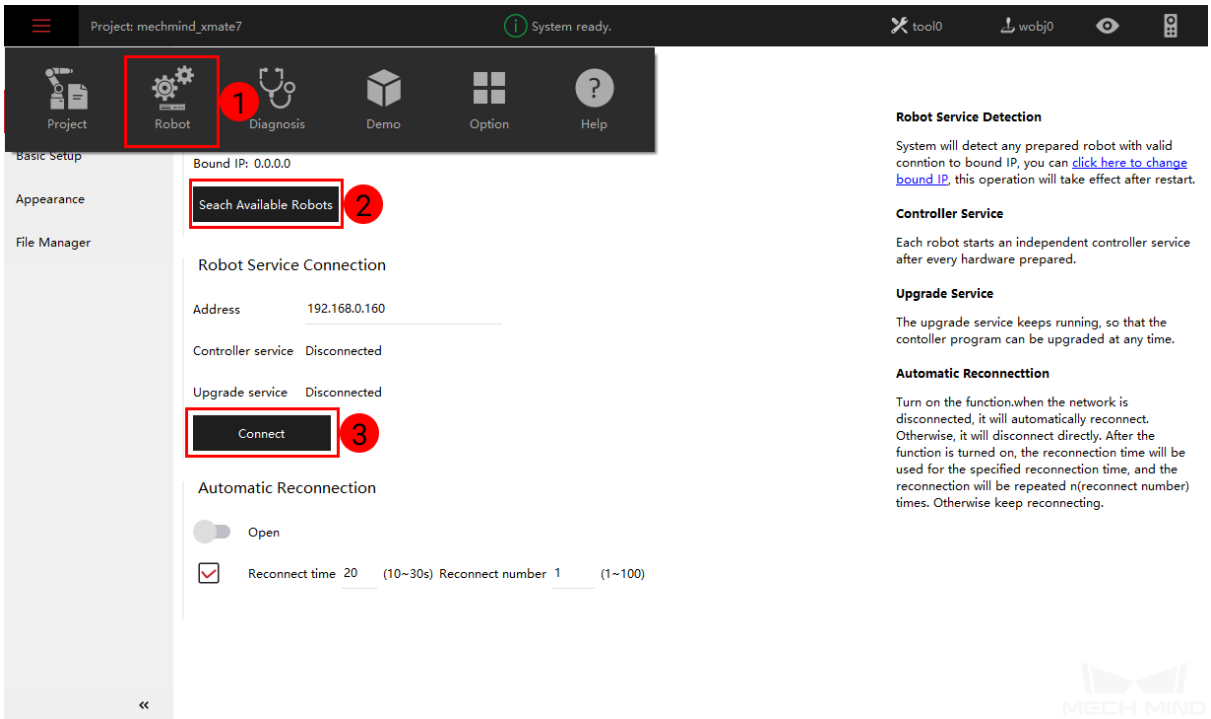
The process consists of 4 steps:

- *Upgrade Software*
- *Setup the Network Connection*
- *Load the Program Files*
- *Test Robot Connection*

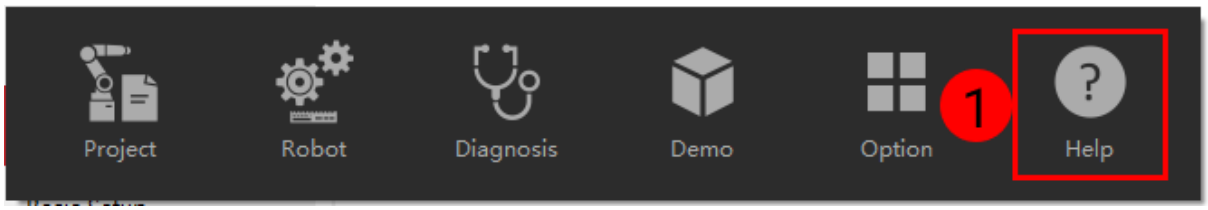
Please have a flash drive ready at hand.

1.10.1 Upgrade Software

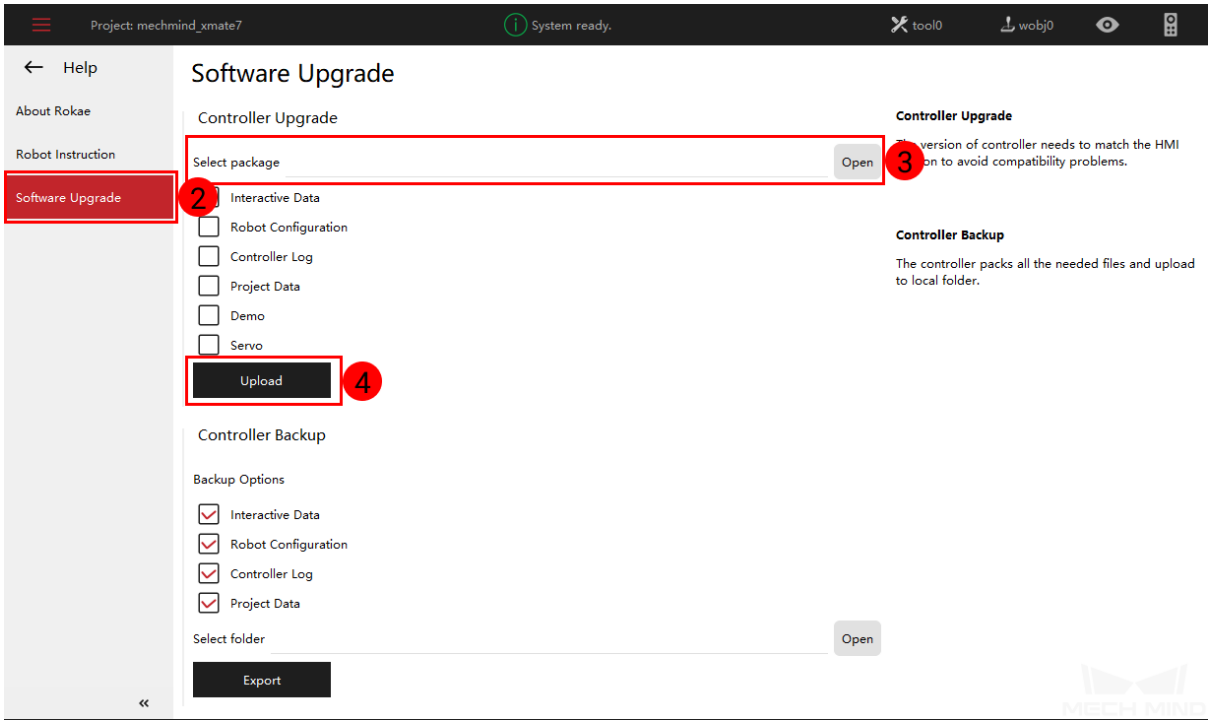
1. Start the robot and open the ROKAE Xmate7 control system software.
2. Click on *Robot* → *Search Available Robots* → *Connect* to connect the robot as shown below.



3. If an alert window pops up, showing that the current control system is not compatible with the robot model, please upgrade the system according to the instruction.
4. After upgrading the control system, you will need to upgrade the controller software manually.
   1. Please download the ROKAE upgrade package first and then copy and paste it into a USB flash drive.
   2. Select *Help*.



   3. Go to *Software Upgrade* → *Open* to select the upgrade package in the USB flash drive, and then click on *Upload*.

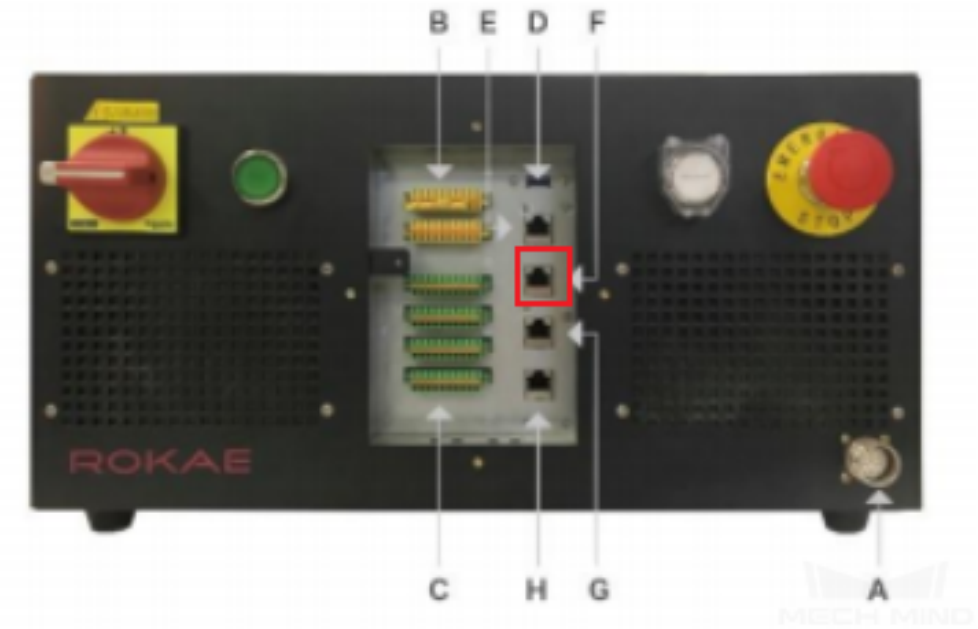


Note: The ROKAE controller of 3.6 version is compatible with Mech-Center 1.5.0 or higher. If you are using a controller whose version is lower than 3.6, please use the Mech-Center of a previous version.

1.10.2 Setup the Network Connection

Hardware Connection

Plug the Ethernet cable of the IPC into the F port as shown below.



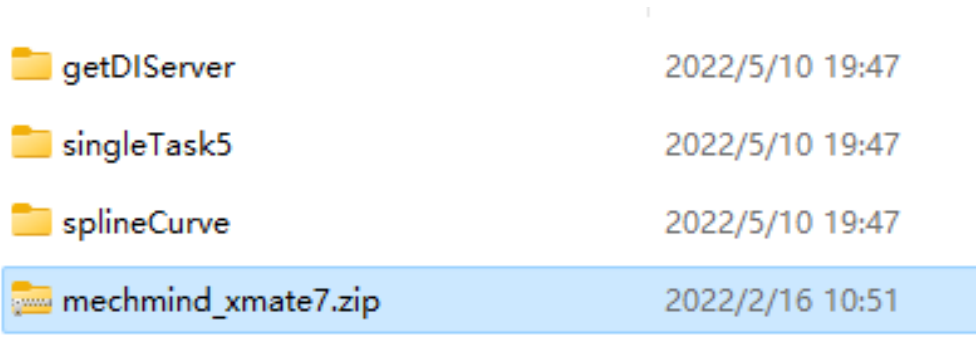
IP Configuration

The default robot IP address is 192.168.0.160, please set the IP address of the IPC to 192.168.0.222. After configuration, you can check the connection by entering the command **ping 192.168.0.160** in the Command Prompt window.

1.10.3 Load the Program Files

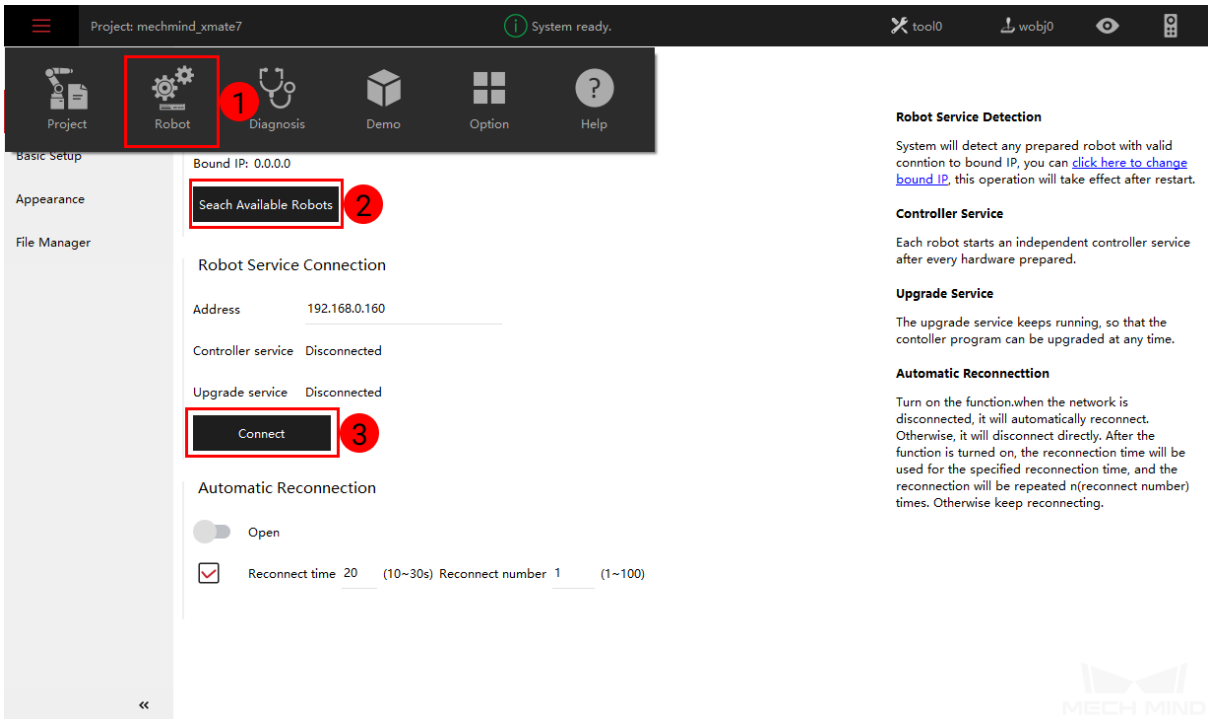
Prepare the Files

Go to the folder where Mech-Center is installed and copy the **mechmind_xmate7.zip** on the path *XXX\Mech-Center\Mech_RobServ\install_packages\rokae*, and paste it into the flash drive.



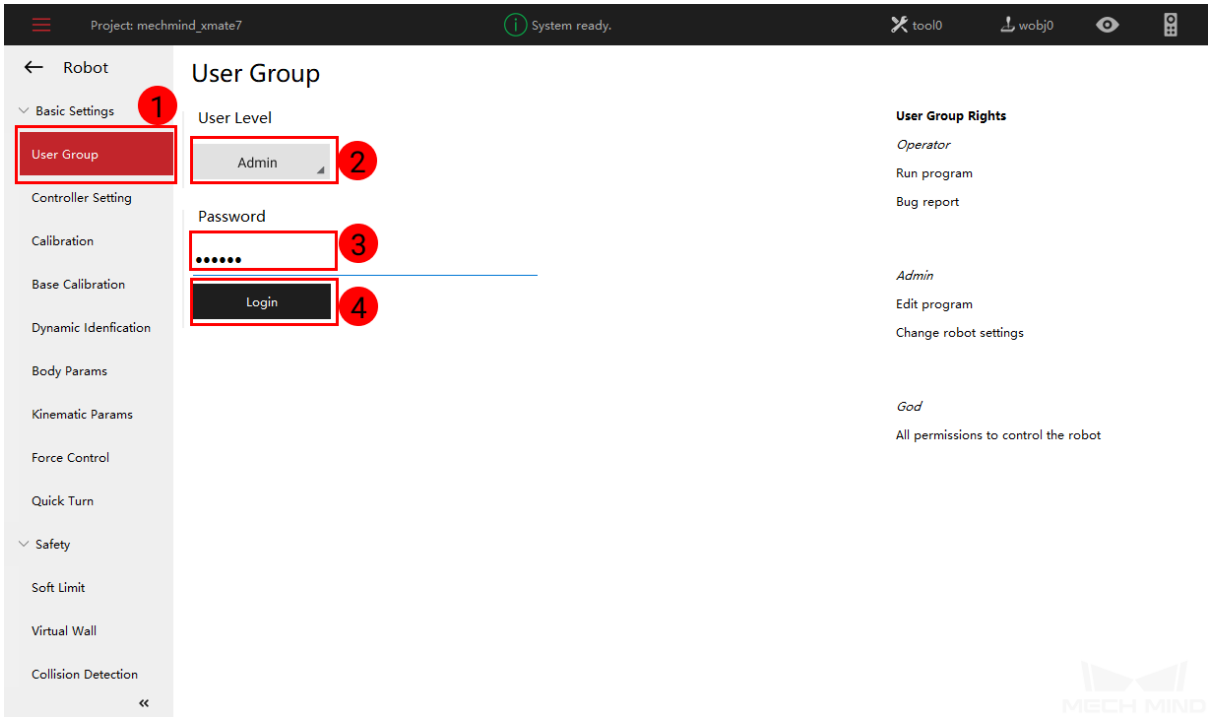
Connect the Robot

1. Click on *Robot* → *Search Available Robots* → *Connect* to connect the robot again.



Switch the Level

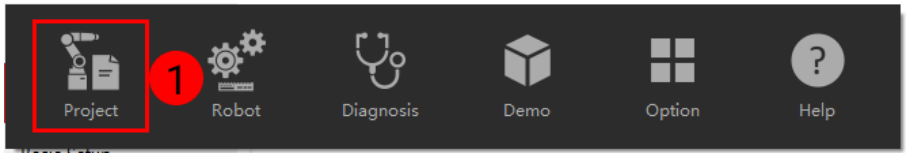
Go to *Basic Settings* → *User Group* and select **Admin** as the user level, enter the default password 123456, and then click on *Login* to finish setting.

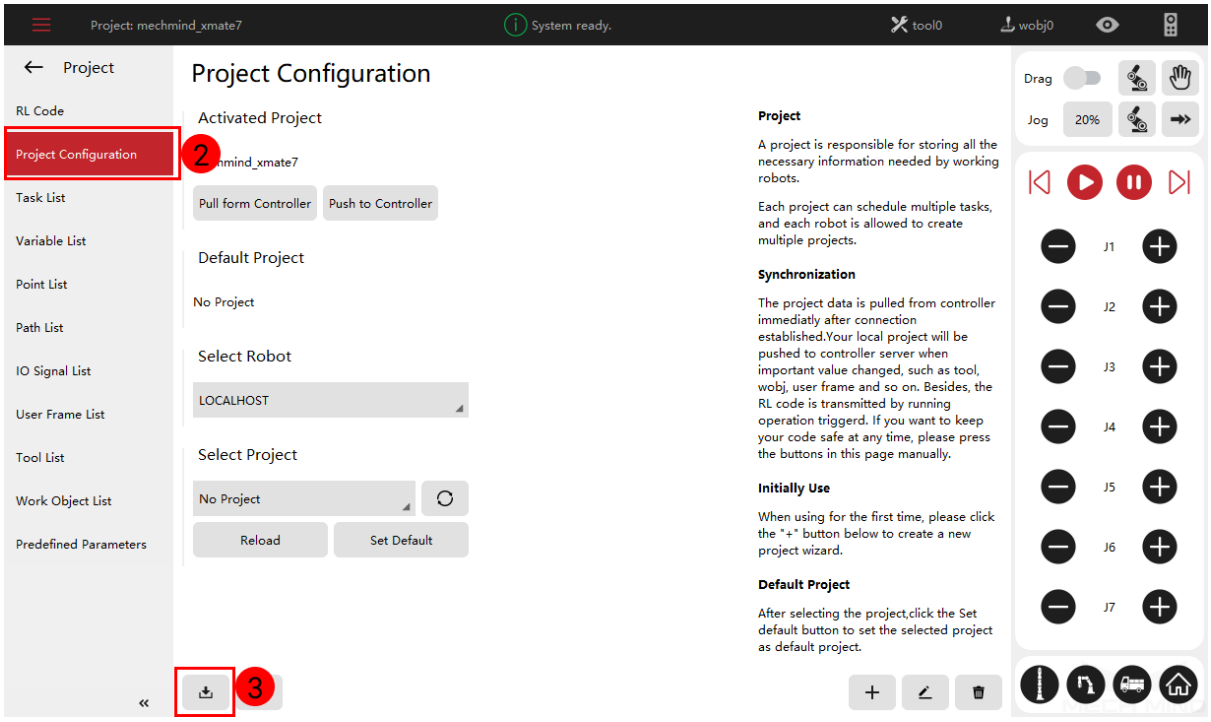


Load the Files to the Robot

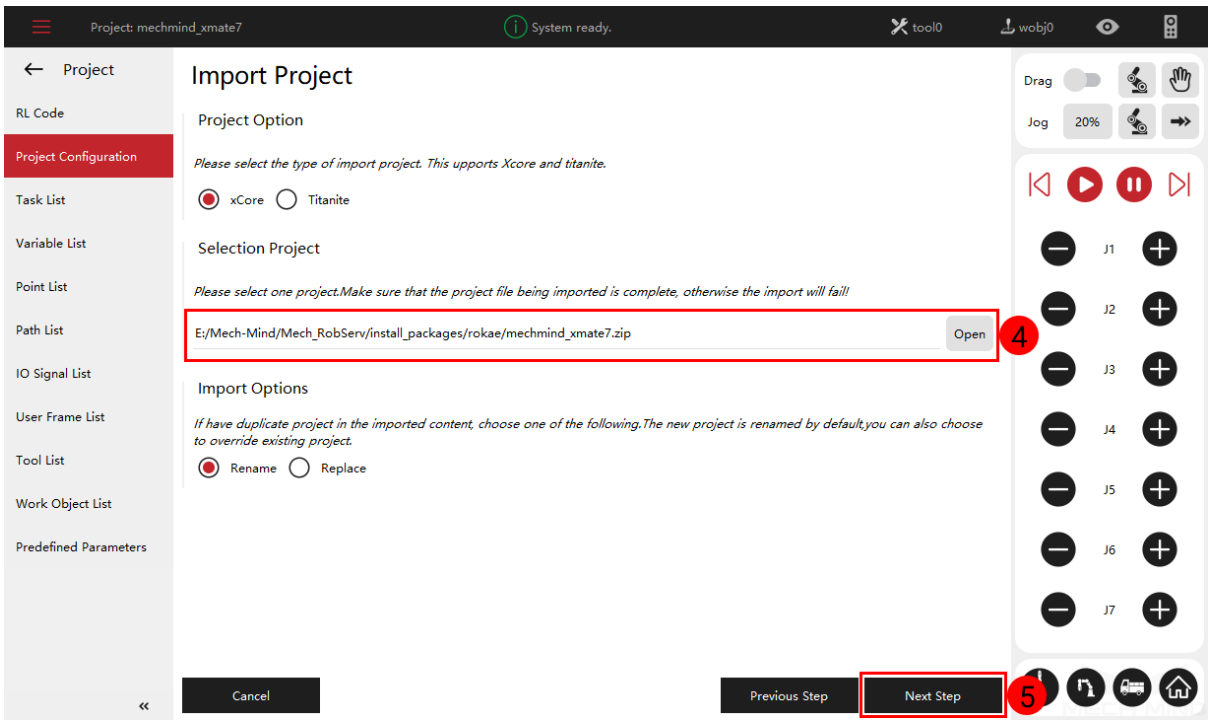
Please follow the steps below to load the full-control program to the robot.

1. Go to *Project* → *Project Configuration*.

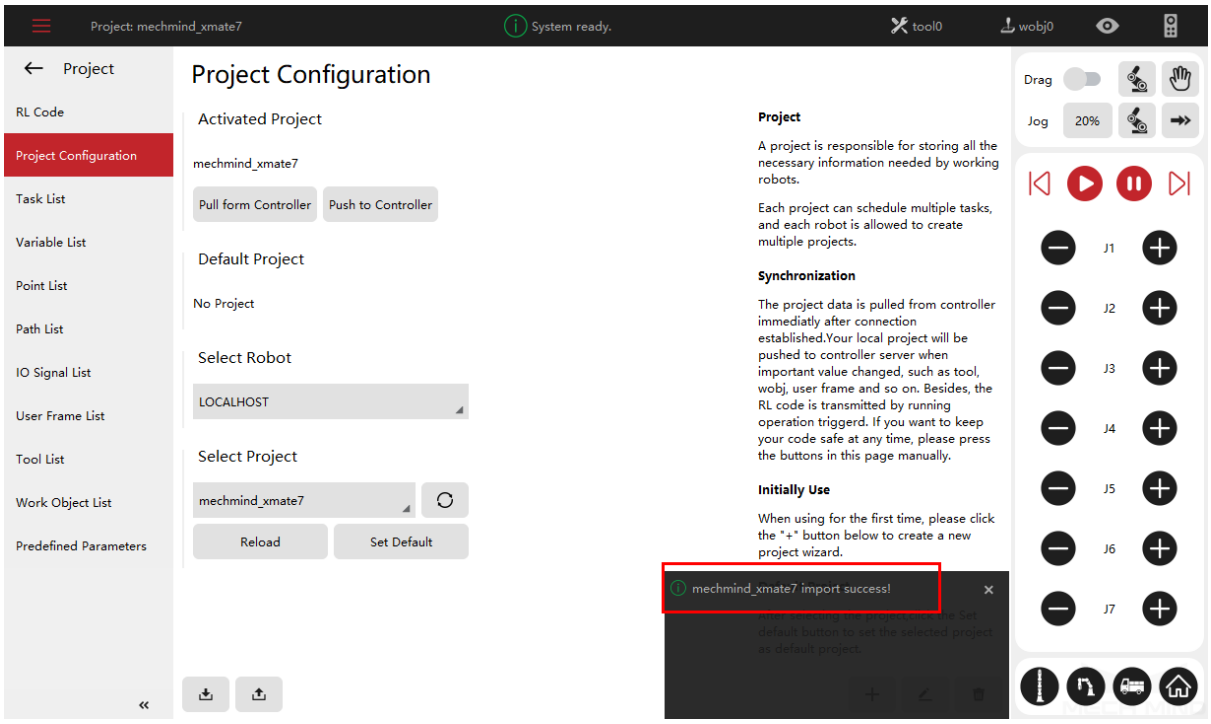




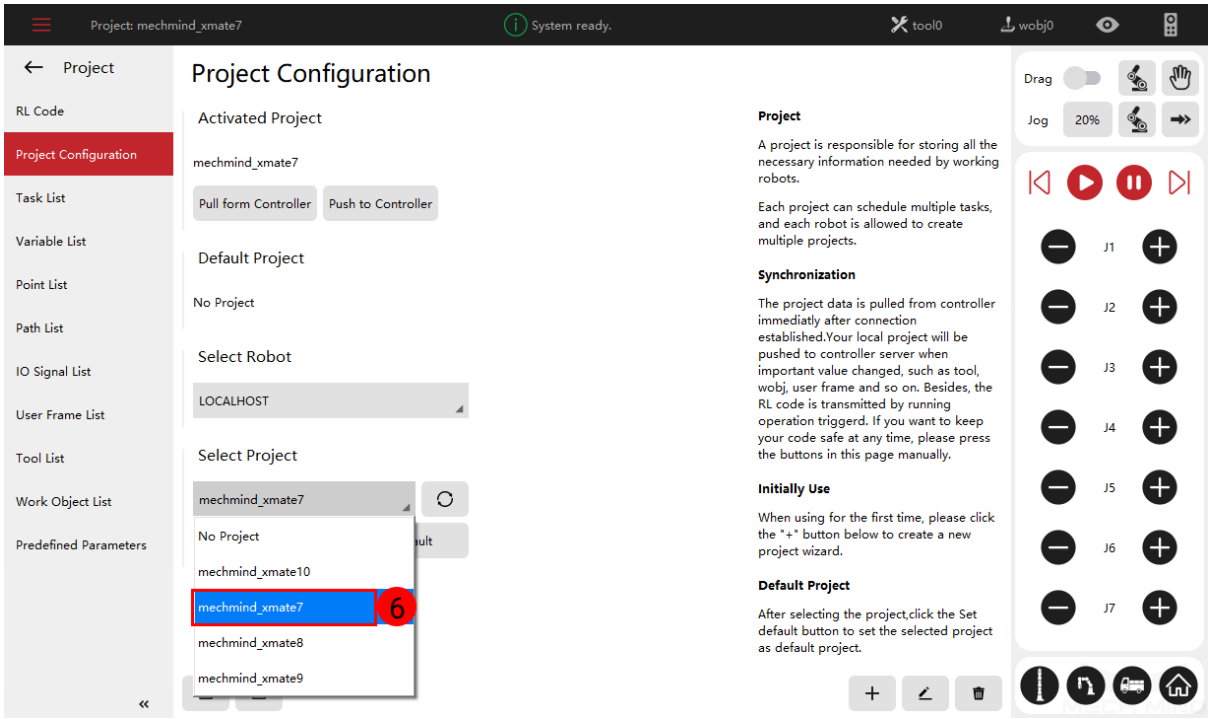
2. Open the program file to be imported, and then click on *Next Step*.



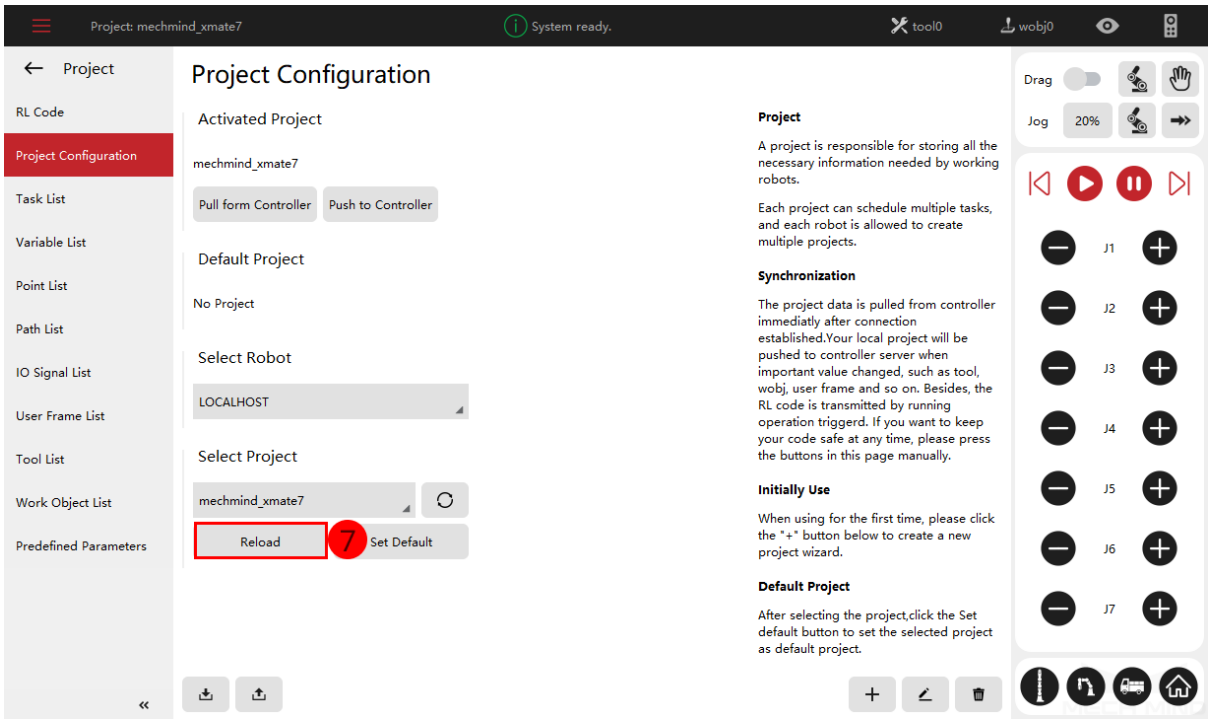
3. An "Import Success" message will appear in the lower right corner.



4. Select the program to be loaded.

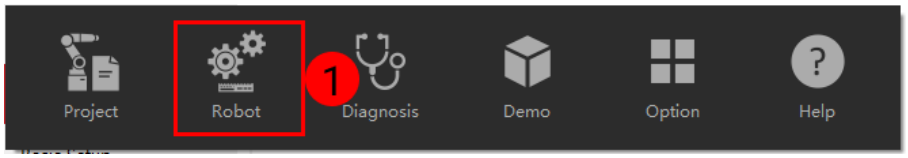


5. Click on *Reload*.

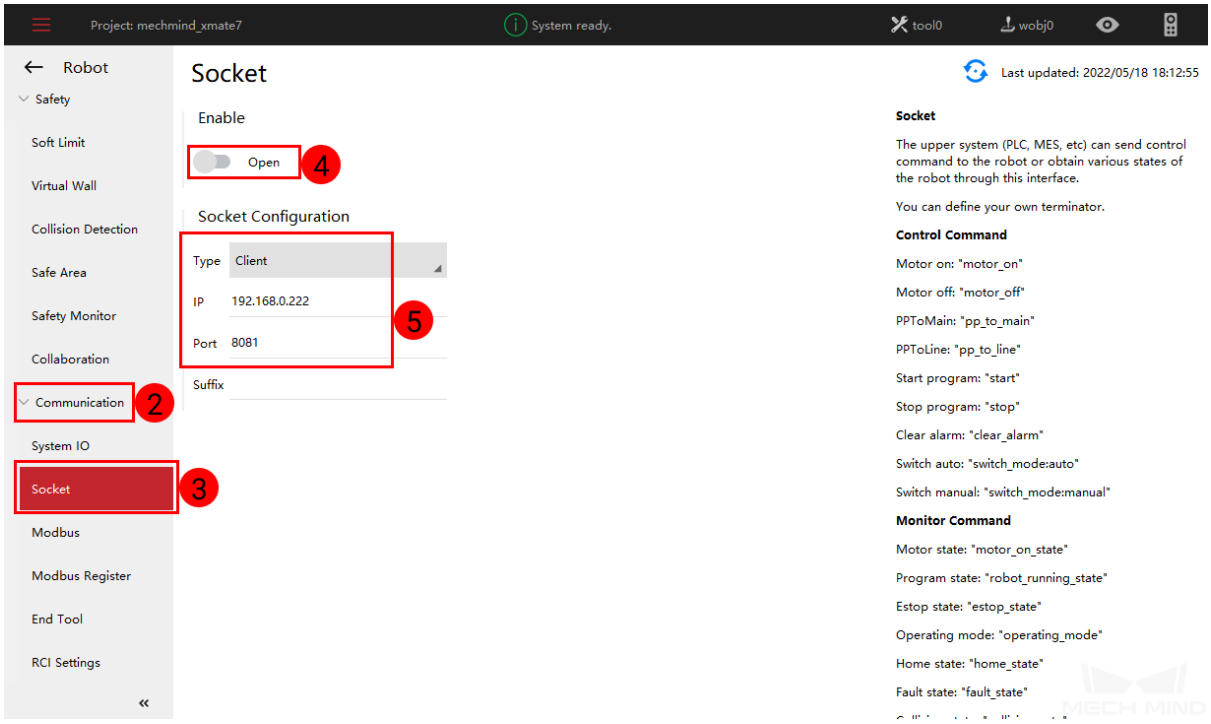


Socket Configuration

1. Click on *Robot*.

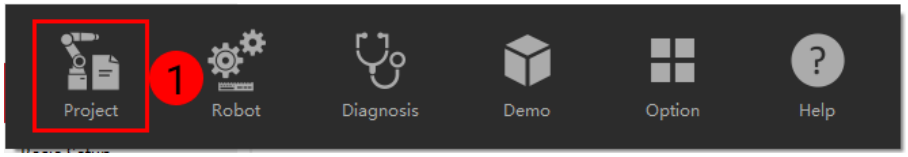


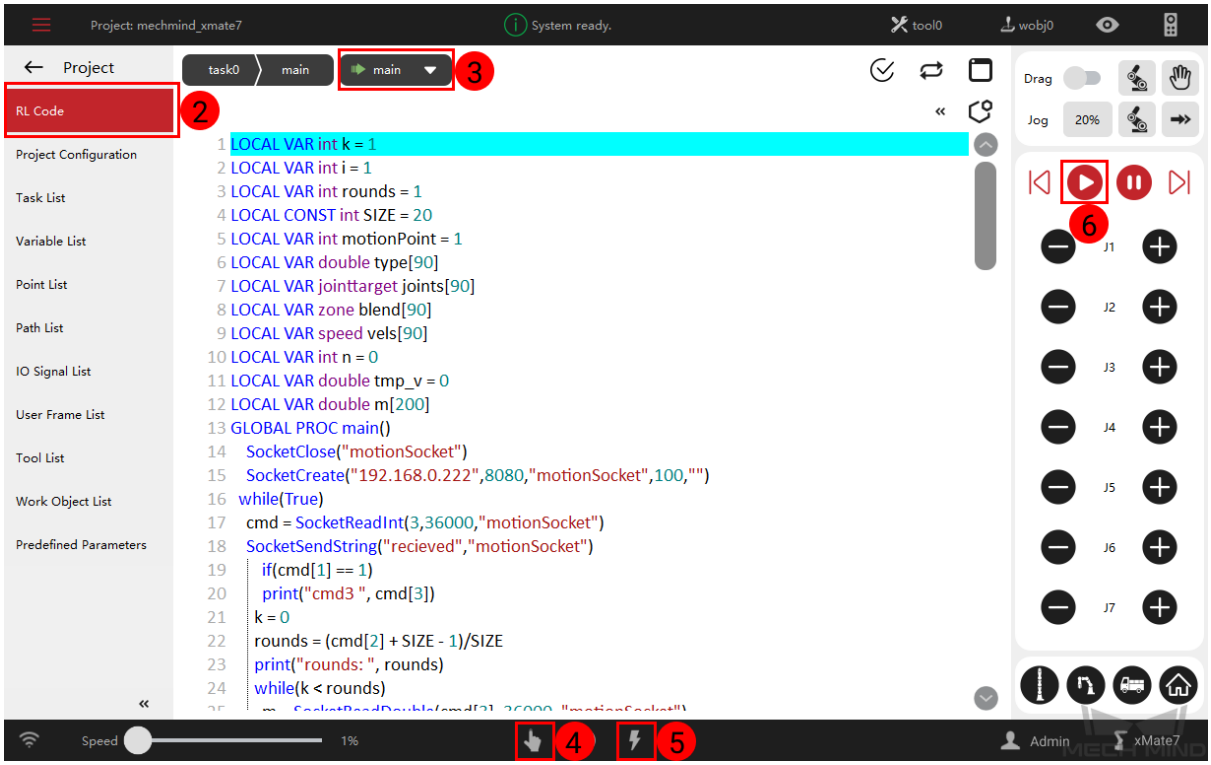
2. Go to *Communication* → *Socket* and then follow the steps as shown in the figure below to configure the socket.



1.10.4 Test Robot Connection

1. Click on *Connect Robot* in Mech-Center.
2. Run the full-control program, as shown below.





3. If a message saying **Robot: server connected to the robot** shows up in the **Log** panel, the robot is successfully connected.

1.11 AUBO Setup Instructions

This section introduces the process of setting up full control of an AUBO robot.

The process consists of 5 steps:

- *Check Controller Compatibility*
- *Setup the Network Connection*
- *SDK Compatibility*
- *Troubleshooting*
- *Test Robot Connection*

1.11.1 Check Controller Compatibility

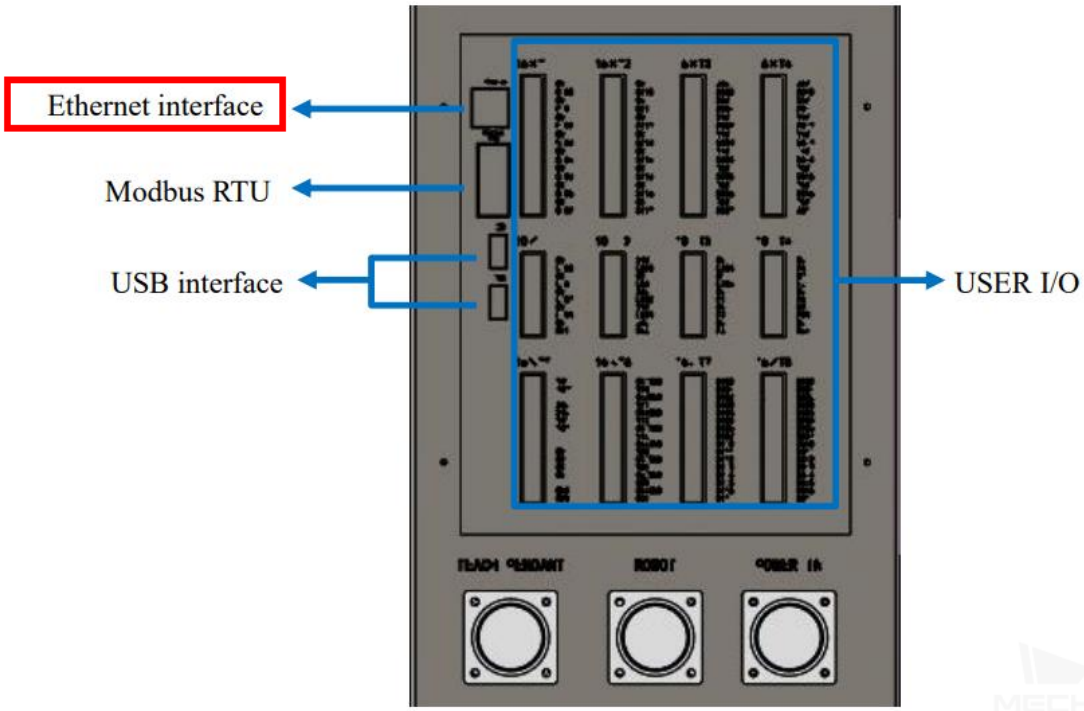
The version of the controller should be above 4.5.44.

Go to *About* → *Version* → *Server Version* to check the version of the controller.

1.11.2 Setup the Network Connection

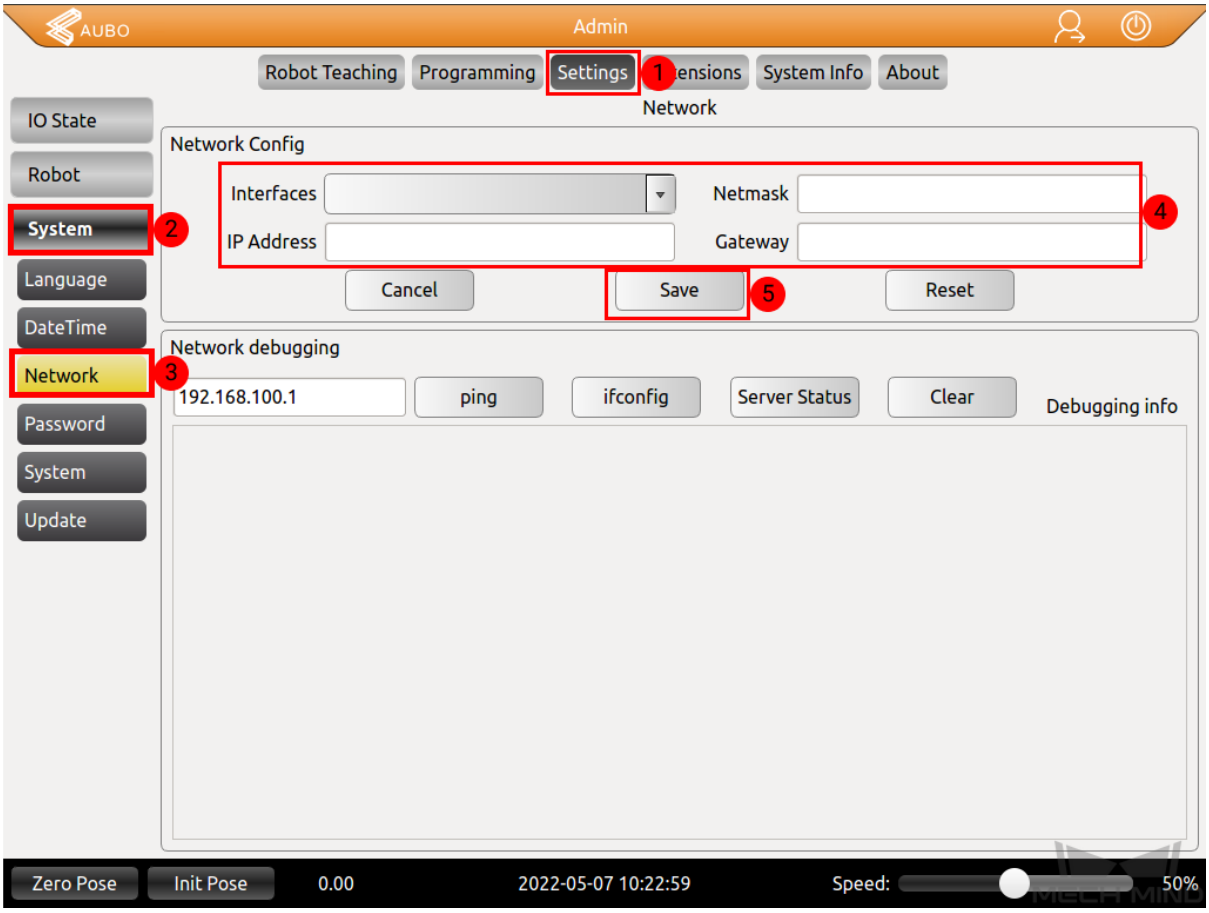
Hardware Connection

Plug the Ethernet cable into the Ethernet interface of the controller to connect the IPC and the robot controller.



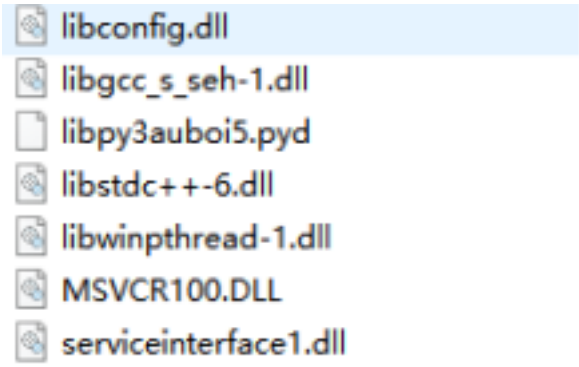
IP Configuration

Press on *Settings* → *System* → *Network* to configure the **Interfaces**, **Netmask**, **IP Address**, and **Gateway**, and then select *Save*. Please note that the robot IP should be in the same subnet as the IPC.



1.11.3 SDK Compatibility

1. Please download the AUBO Python-Dlls first.
2. Copy all the DLL files in the python-Dlls folder.
3. Locate the folder where python is installed on the IPC and paste all the DLL files into the **DLLs** folder (The default path is *C:/Program Files/Python36/DLLs*). The DLL files are as shown below.



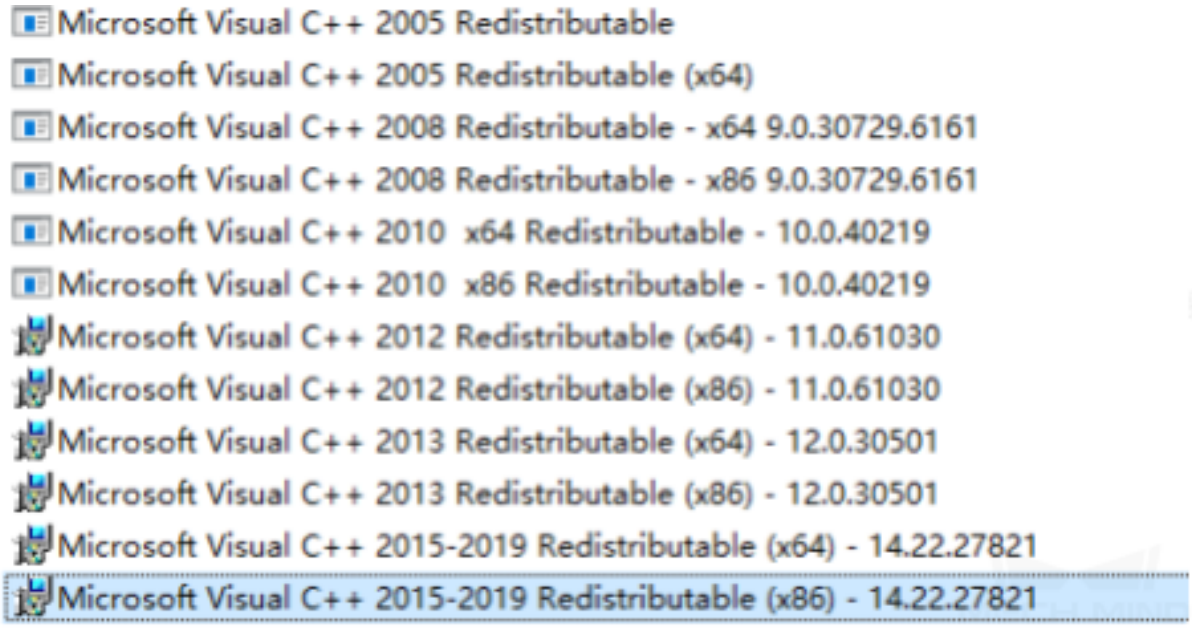
4. You do not need to load the files to the robot.

1.11.4 Test Robot Connection

Please refer to *Test Robot Connection* for detailed instructions.

1.11.5 Troubleshooting

After copying and pasting the files, if the robot cannot be connected successfully, and the error is **DLL load failed: %1 is not a valid Win32 application**, please check whether the C++ runtime library on your computer is complete. The complete C++ runtime library is as shown below.



If the library is not complete, please download the vc runtime library repair DirectX Repair V3.9 to fix the error.

1.12 JAKA Setup Instructions

This section introduces the process of setting up full control of a JAKA robot.

The process consists of 3 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Test Robot Connection*

JAKA robots are controlled through the mobile or PC application JAKA Zu APP, instead of a teach pendant.

Before proceeding, please make sure you have downloaded and installed JAKA Zu APP on the device you wish to use for controlling the robot.

Please download the application here, under *Technical Information* → *APP Software*.

1.12.1 Check Controller and Software Compatibility

- Controller version: 1.5.12_28_x86

  Note: The controller version must be 1.5.12_28_x86. If your controller is of a lower version, please upgrade it; if higher, please downgrade.

- JAKA Zu APP version: V 1.5 (for both Android and Windows)
- Mech-Center: latest version recommended

1.12.2 Setup the Network Connection

Hardware Connection

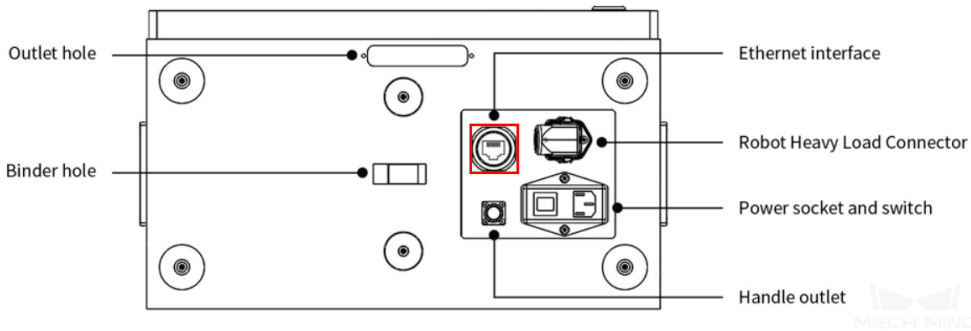
The hardware setup differs depending on what device you have installed JAKA Zu APP on.

- If you installed JAKA Zu APP on the IPC:

  Plug the Ethernet cable into:

  – An Ethernet port on the IPC

  – The Ethernet port inside the accessory panel on the front of the controller



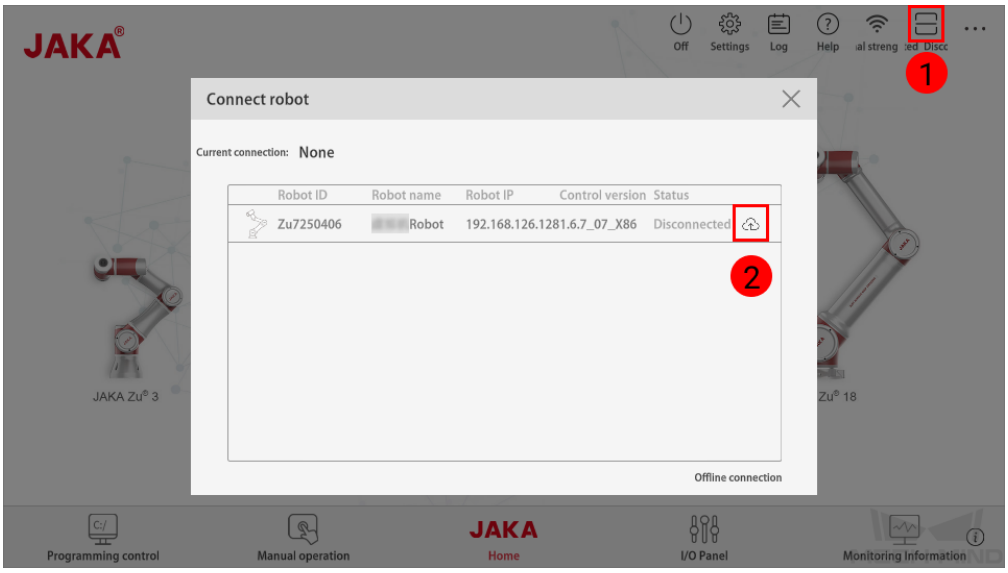
- If you installed JAKA Zu APP on a mobile device:
  1. Prepare a router with wireless function.
  2. Connect both the IPC and the controller to the router with Ethernet cables.
  3. Connect the mobile device to the router' s wireless network.

Note: If you are using multiple cameras, make sure the router has enough LAN ports for the IPC and cameras.

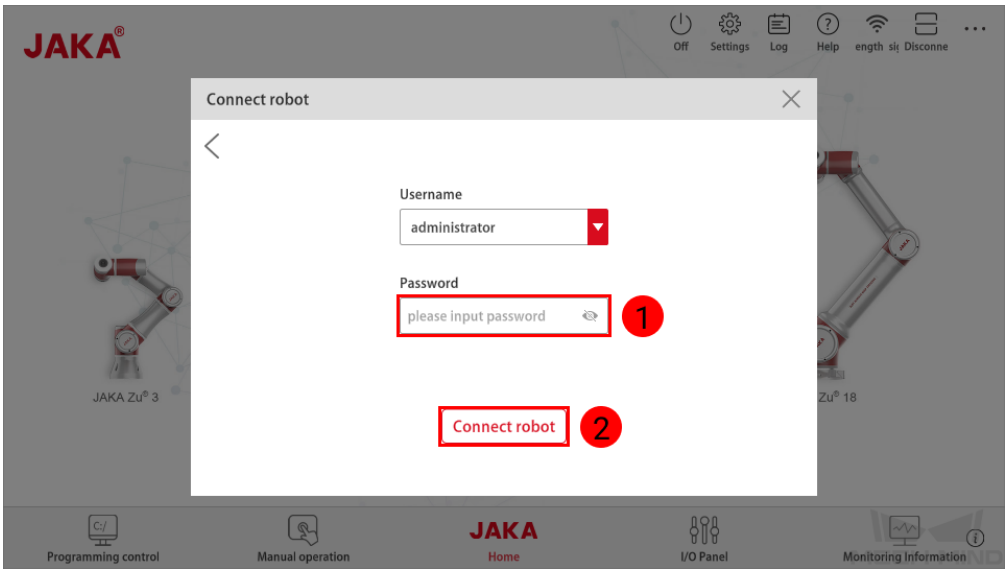
IP Configuration

To allow communication between the IPC and the robot controller, both must have an IP address in the same subnet. This means that the first three numbers of the IP addresses should be the same. For example, 192.168.100.1 and 192.168.100.2 are in the same subnet.

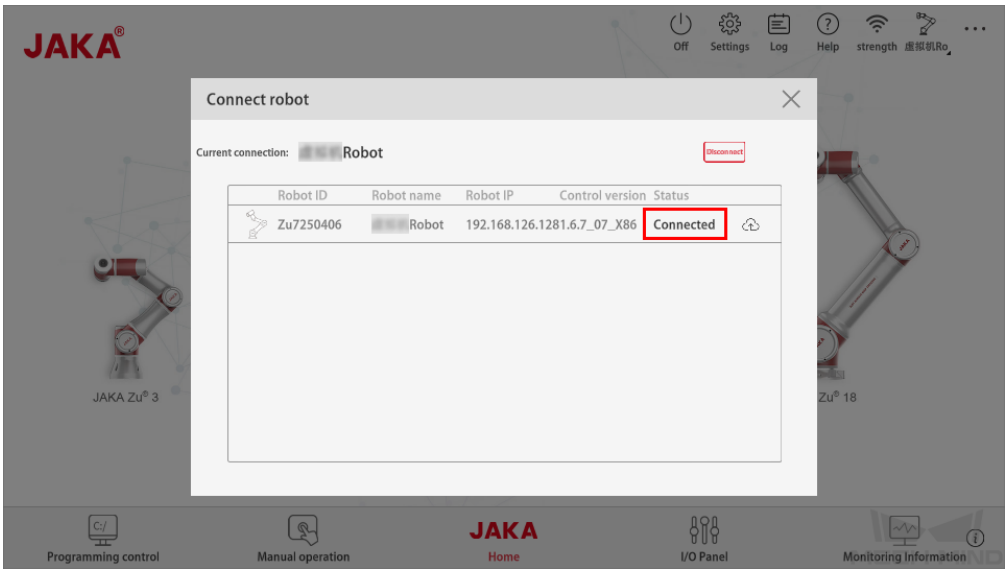
1. Check the IP address of the IPC: please use the *ipconfig* command in Command Prompt or PowerShell to check the IP address.
2. Open JAKA Zu APP, click on  in the upper right, and then click on  .



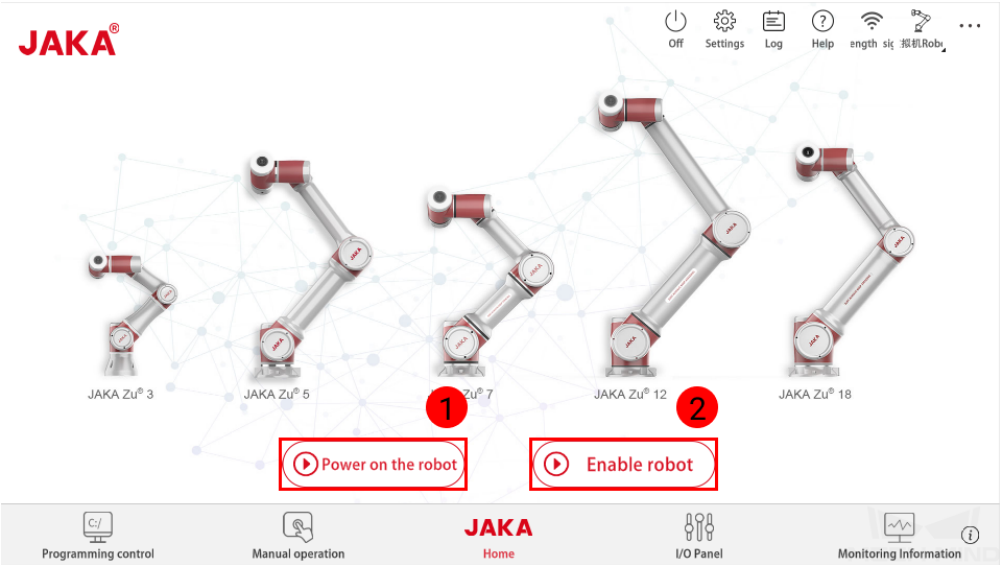
3. Enter the password for **administrator** (the default password is *jakazuadmin*), and click on *Connect robot*.



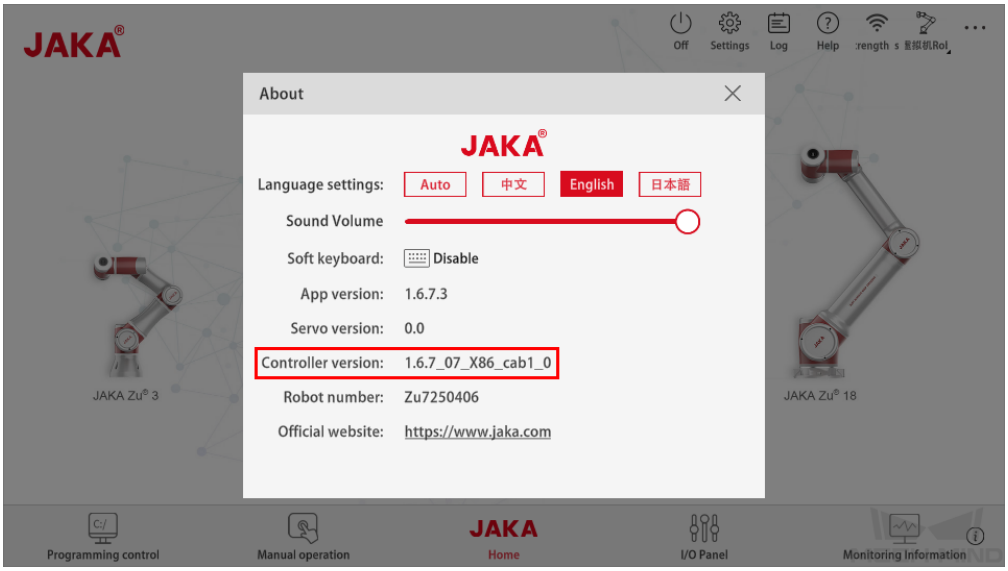
4. If successfully connected, the **Status** of the robot should change to **Connected**. Close the **Connect robot** window.



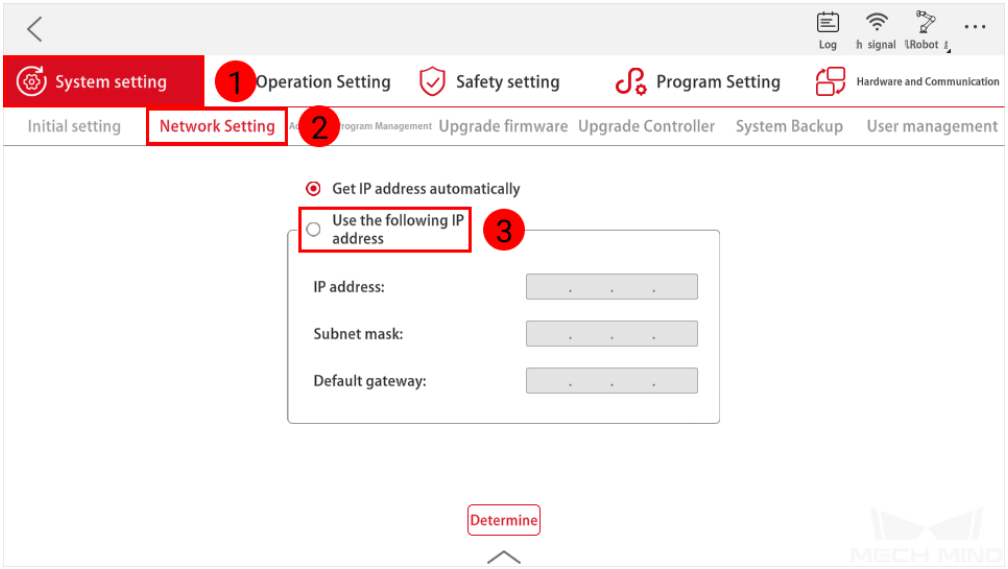
5. Click on *Power on the robot* and then *Enable robot* to power on the robot.



6. Click on  in the lower right to check the controller version. The controller version must be **1.5.12_28_x86**. If not, please upgrade or downgrade the controller to this version.



7. Close the **About** window, and click on  in the upper right. In *System setting* → *Network Setting*, select **Use the following IP address**, and set the **IP address** to one in the same subnet as the IPC, and **Subnet mask** to **255.255.255.0**.



1.12.3 Test Robot Connection

Please refer to *Test Robot Connection* for detailed instructions.

1.13 ELITE Setup Instructions

This section introduces the process of setting up full control of an ELITE robot.

1.13.1 Check Controller and Software Compatibility

- Controller software version: 2.19.2 or lower

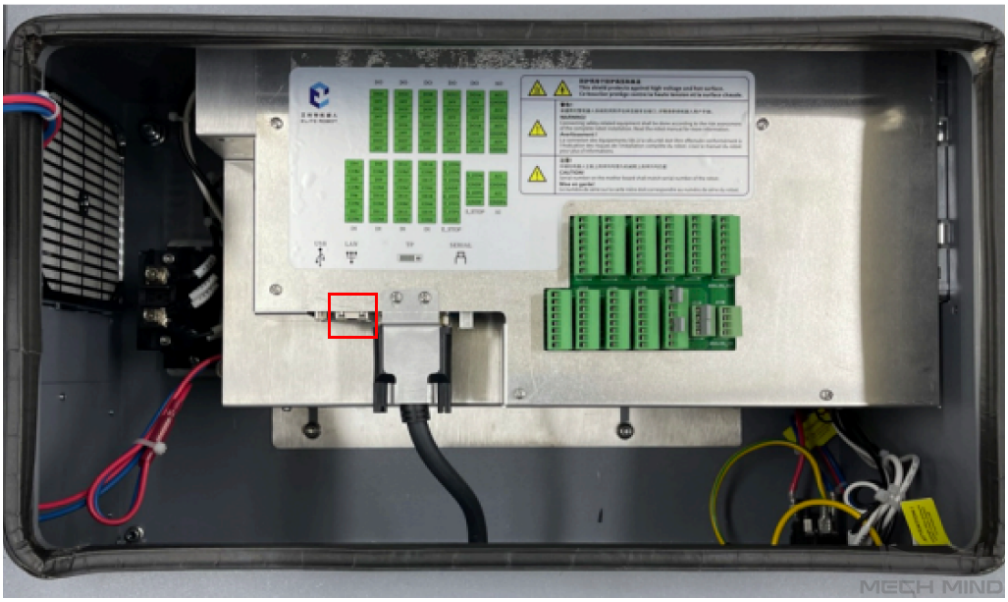
Note: See step 2 in **IP Configuration** for instructions on checking the software version.

1.13.2 Setup the Network Connection

Hardware Connection

Plug the Ethernet cable into:

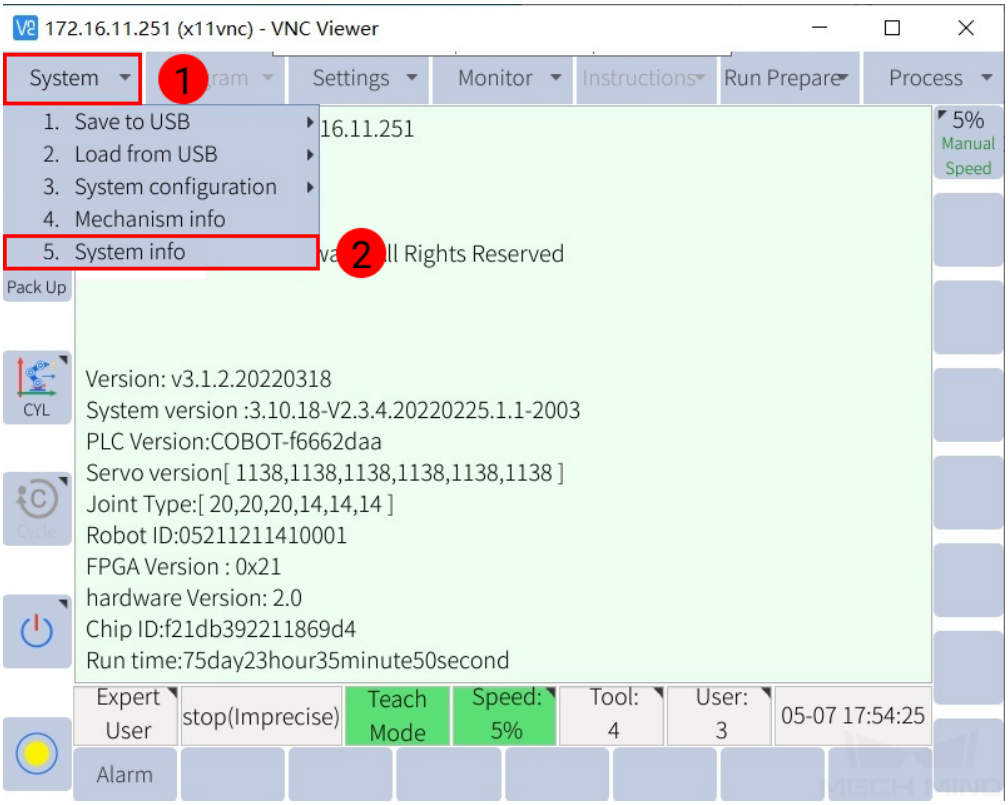
- An Ethernet port on the IPC
- The Ethernet port inside the controller

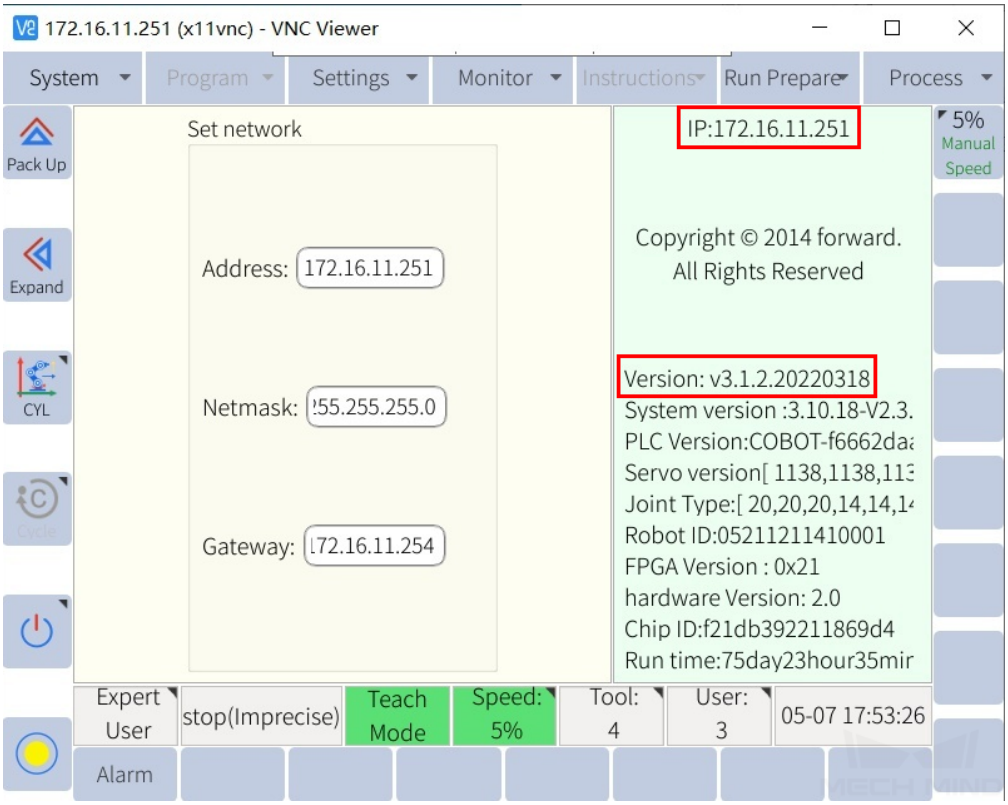


IP Configuration

To allow communication between the IPC and the robot controller, both must have an IP address in the same subnet. This means that the first three numbers of the IP addresses should be the same. For example, 192.168.100.1 and 192.168.100.2 are in the same subnet.

1. Check the IP address of the IPC: please use the *ipconfig* command in Command Prompt or PowerShell to check the IP address.
2. Press on *System* → *5. System info* to check the current IP address and software version.

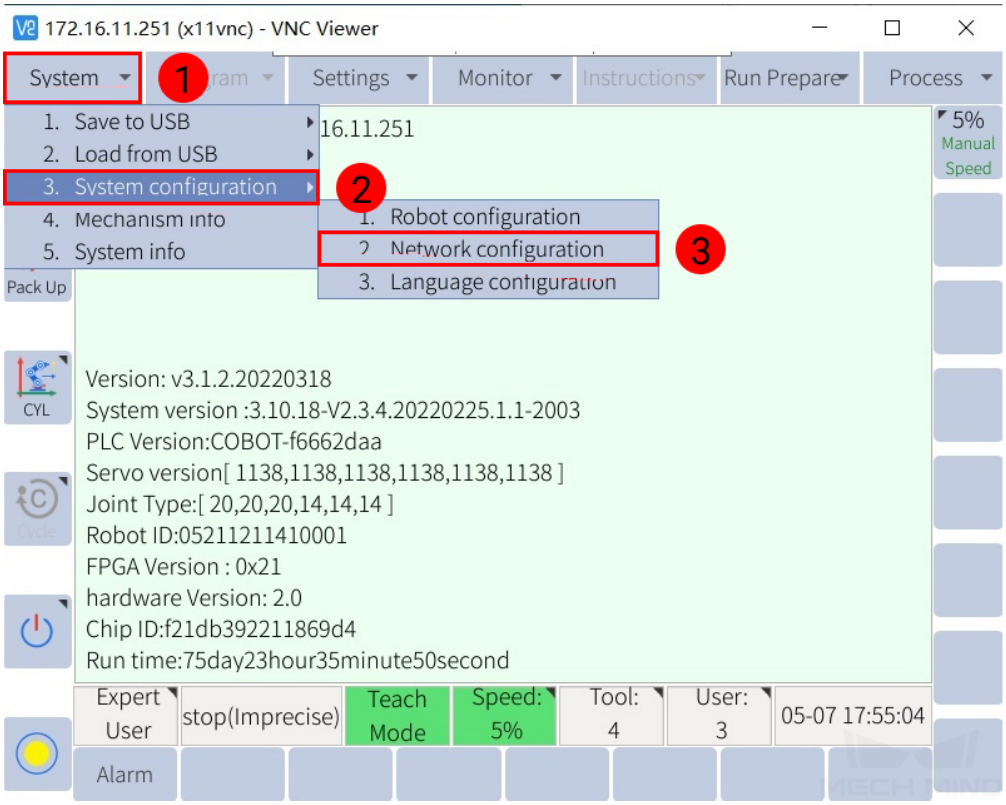




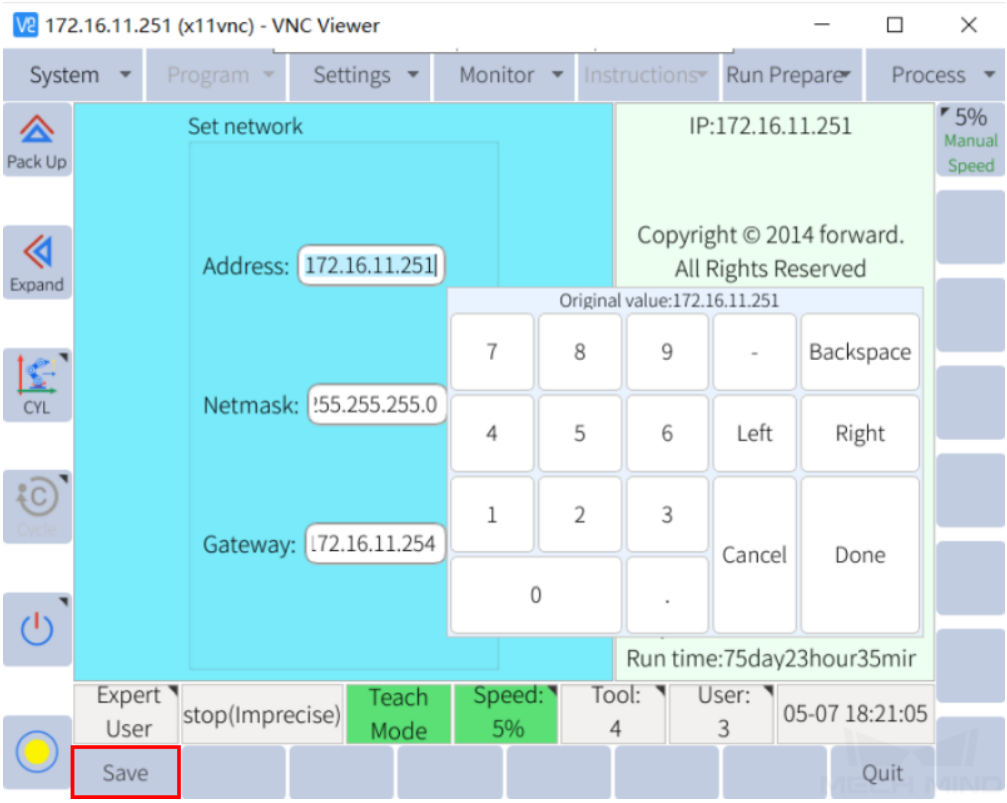
3. If the IP address isn't in the same subnet as the IPC, change it with the following steps:

   1. Turn the key to **TEACH**, and check the current user mode in the lower left. If it's not Admin mode, press and select **Admin**. Then, enter the **Password** and press **OK**.

2. Select *System* → *3. System configuration* → *Network configuration*.



3. Set the IP address to one in the same subnet as the IPC. Press *Save* to save the change.



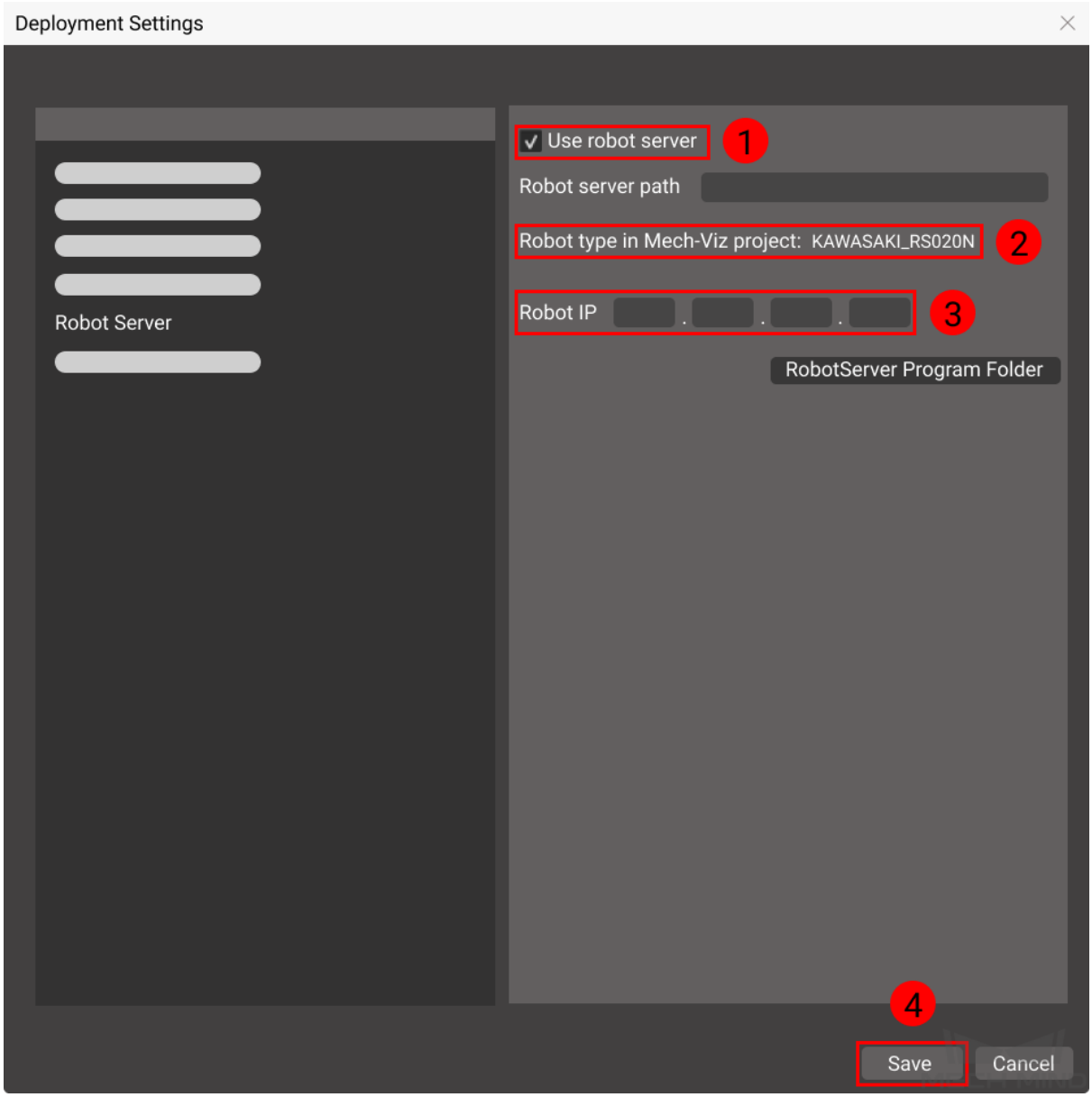
1.13.3 Test Robot Connection

1. Turn the key to **REMOTE**, press the `Servo` key in the lower right of the teach pendant, and make sure the **SERVO** indicator in the upper left lights up.
2. Connect to the robot in Mech-Center. Please refer to *Test Robot Connection* for detailed instructions.

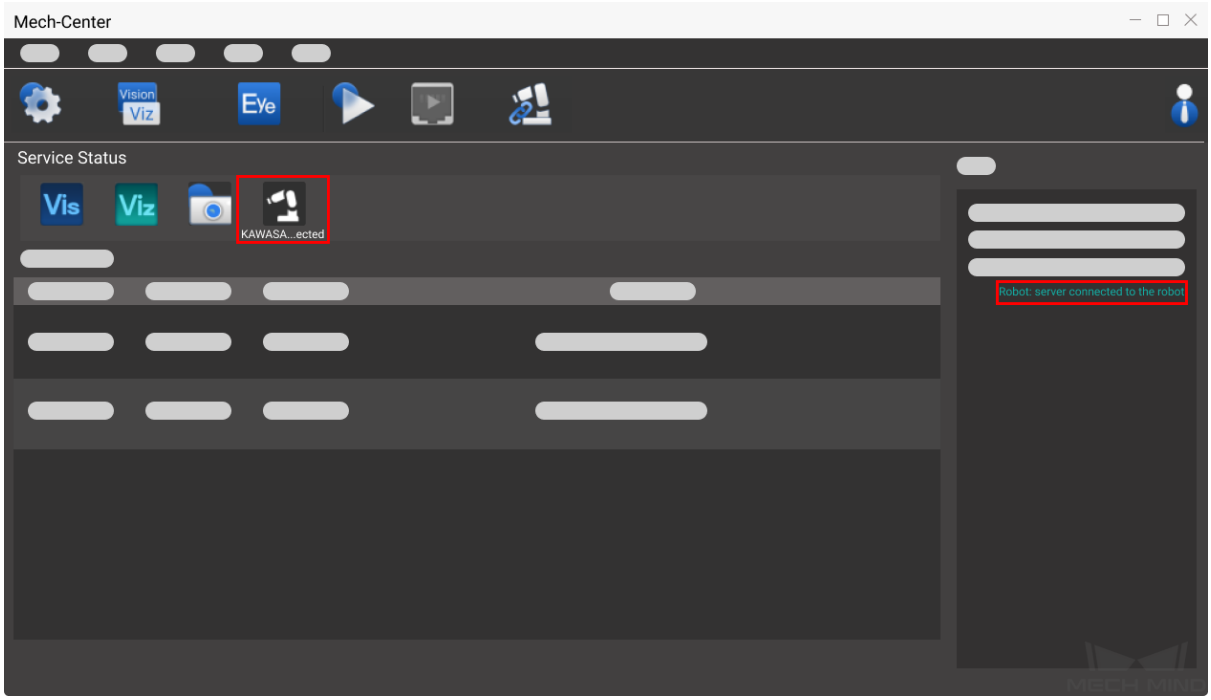
1.14 Test Robot Connection

Note: To successfully connect to the robot in Mech-Center, the corresponding Mech-Viz project must be open, and **Autoload Current Project** must be checked for the project.

1. Open Mech-Center and click on *Deployment Settings*.
2. Go to **Robot Server**, and make sure **Use robot server** is checked.
3. Check if the robot model displayed after **Robot type in Mech-Viz project** matches the one in use.
4. Set the Robot IP address, and click on **Save**.



5. Click on *Connect Robot* in the Toolbar.
6. The robot is successfully connected if:
   - A message saying **Robot: server connected to the robot** shows up in the **Log** panel, and
   -  with the robot model shows up in the **Service Status** panel.


Mech-Center
Vision
Viz
Eye
Service Status
Vis
Viz
KAWASA...ected
Robot: server connected to the robot

CHAPTER

TWO

STANDARD INTERFACE

This chapter provides detailed information on setting up communication with Mech-Mind Software Suite through Mech-Interface' s Standard Interface.

2.1 ABB

This section introduces the Standard Interface for ABB robots.

2.1.1 ABB Setup Instructions

This section introduces the process of loading the standard interface program onto an ABB robot.

The process consists of 4 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program Files*
- *Test Robot Connection*

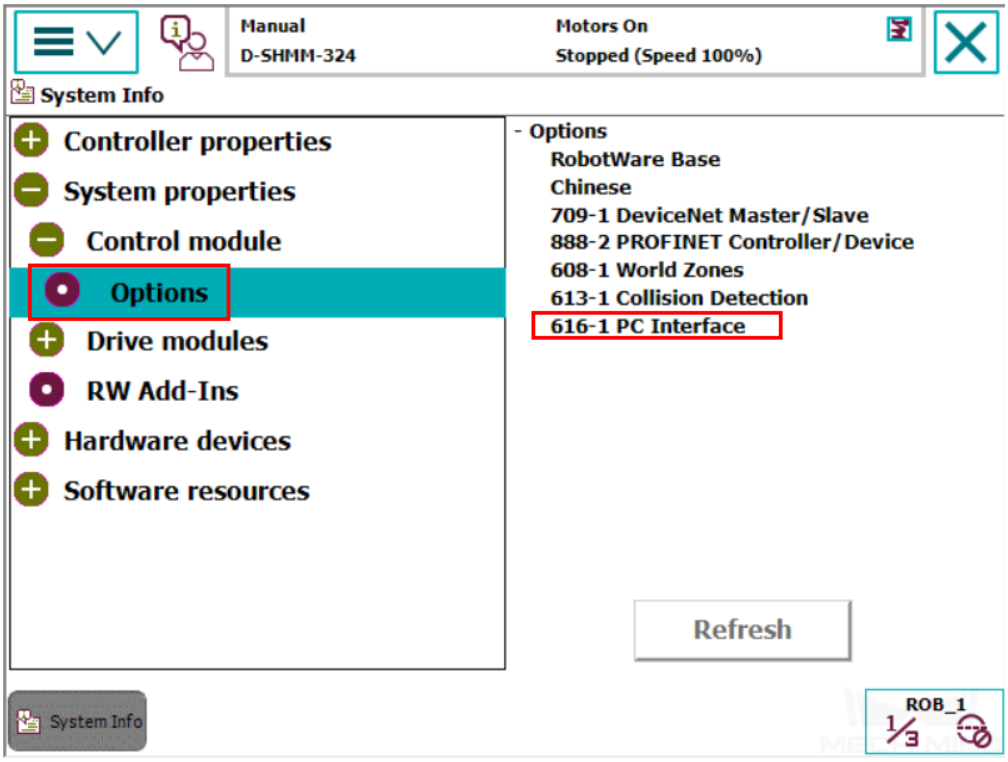
Please have a flash drive ready at hand.

Check Controller and Software Compatibility

Compatibility requirements are as follows:

- Robot: a 4-axis or 6-axis ABB robot
- RobotWare Option: 616-1 PC Interface

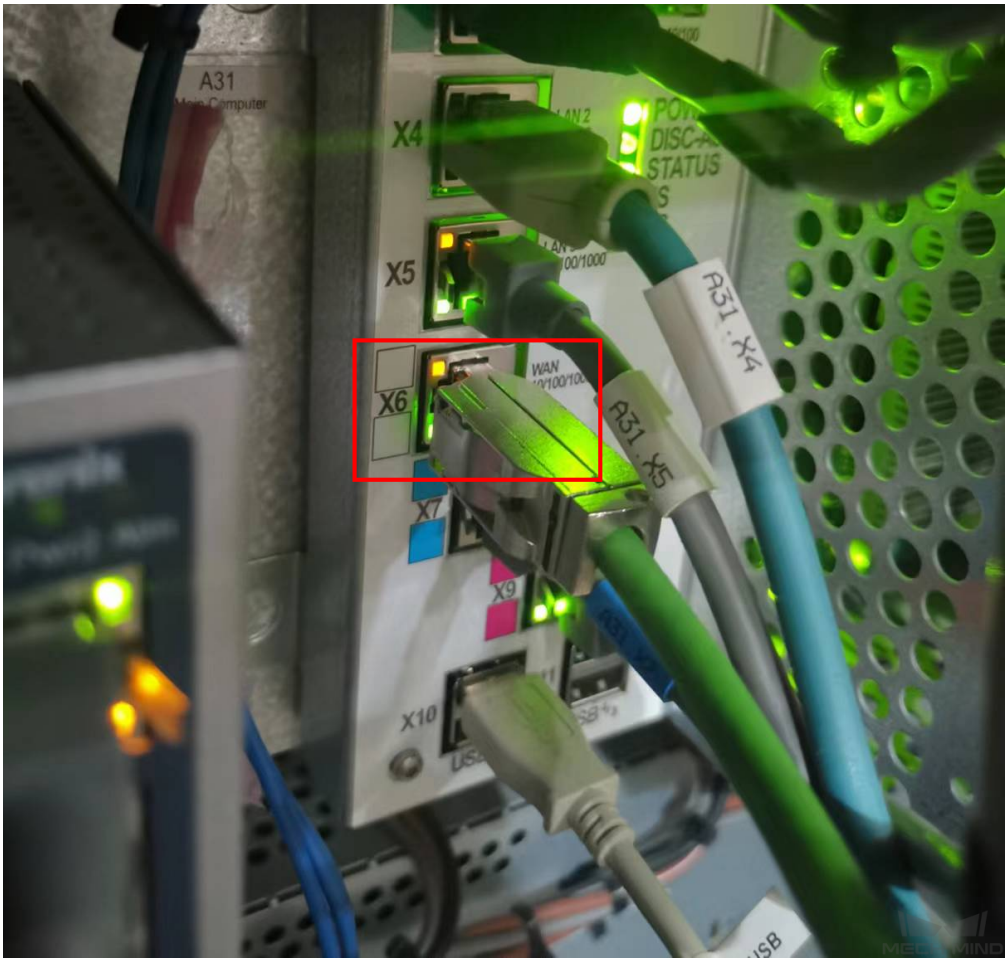
Hint: You can tap  in the upper left corner on the teach pendant and then go to *System Info* → *System properties* to check if the necessary option has been installed.



Setup the Network Connection

Hardware Connection

Plug the Ethernet cable of the IPC into the X6 (WAN) port of the robot controller, as shown below.



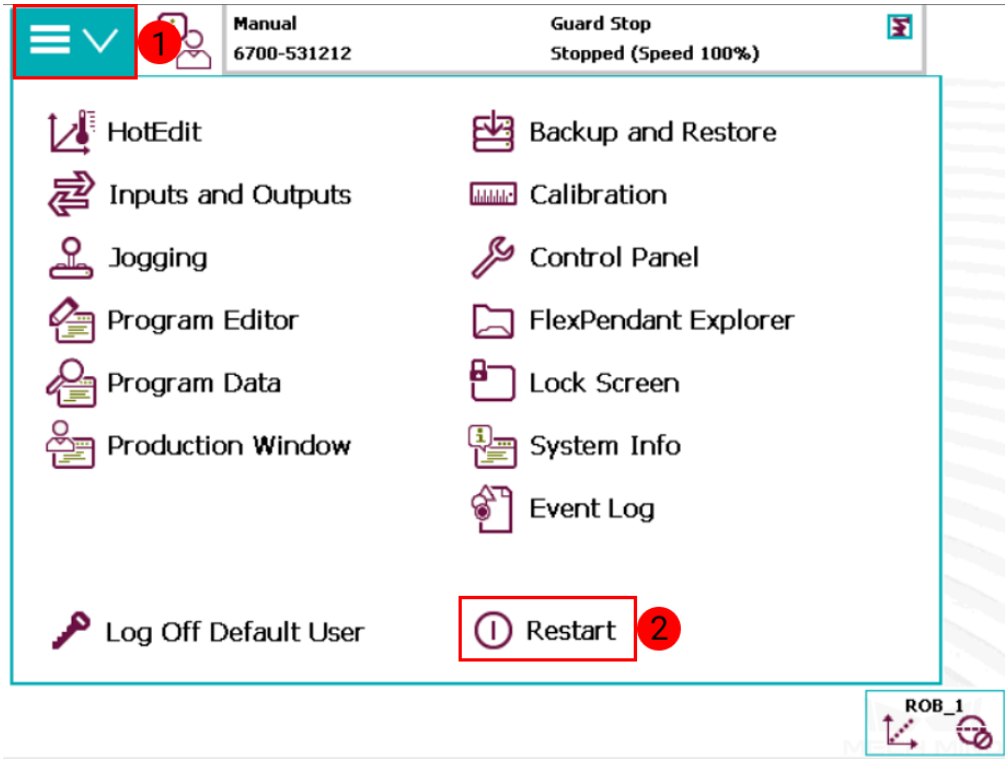
Hint: If you only need to load the program files via RobotStudio, you can use either LAN port or WAN port to connect the robot controller. However, in order to enable visual communication, the Ethernet cable can only be connected to the WAN port.

IP Configuration

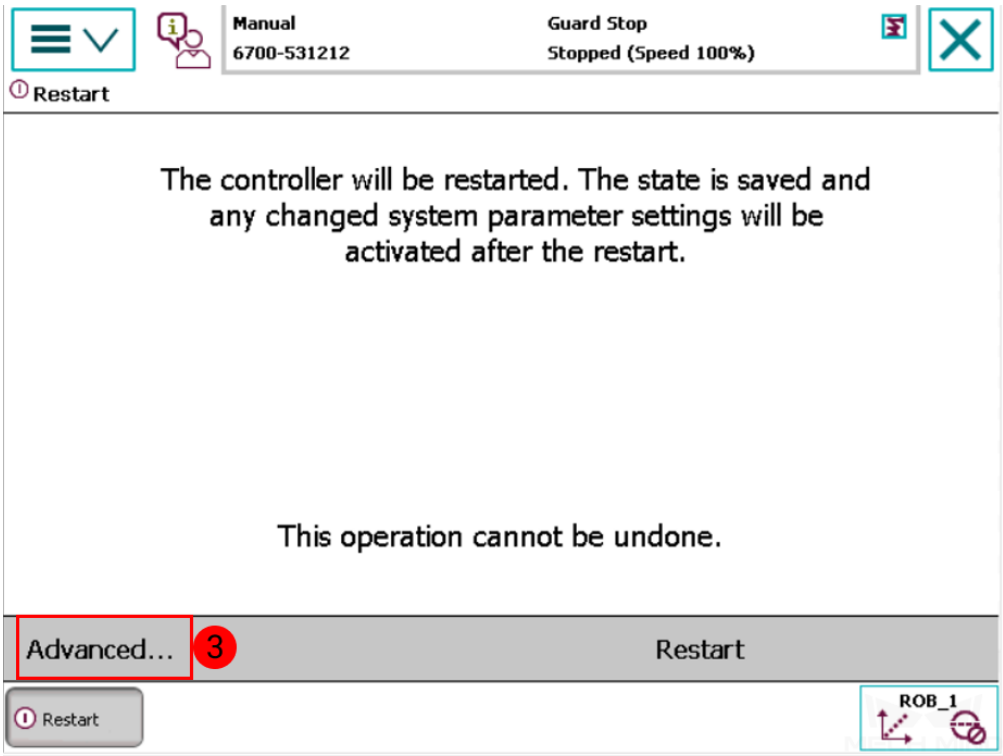
You can set the IP on the teach pendant or via RobotStudio.

- Set the IP on the teach pendant

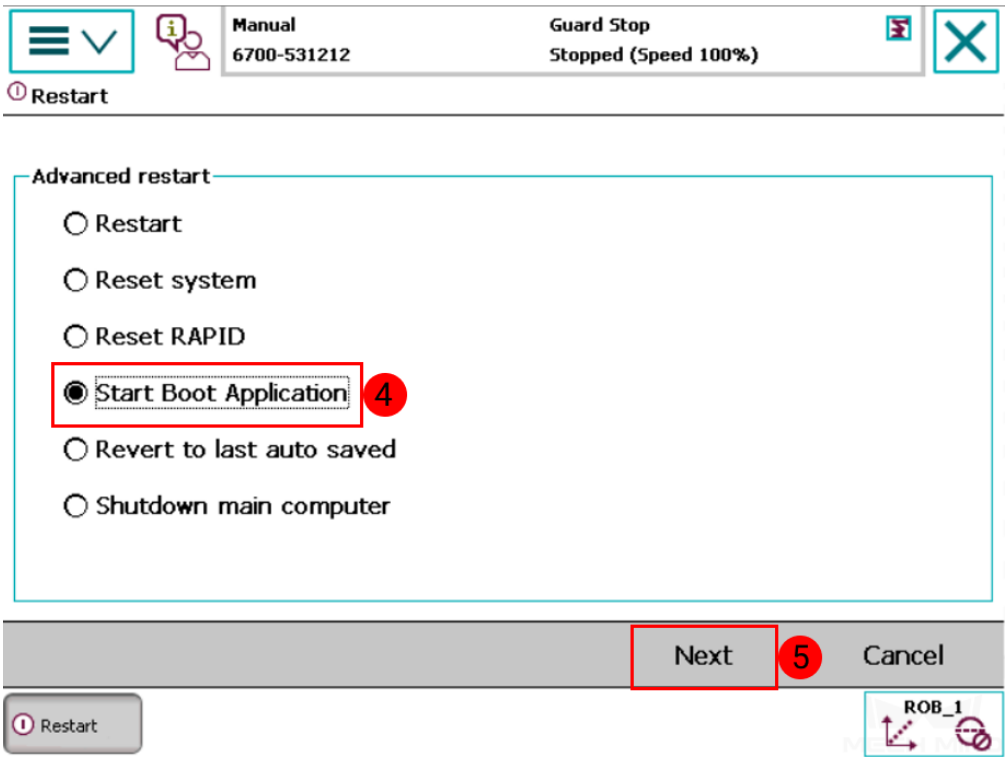
  1. Tap  and select *Restart*.



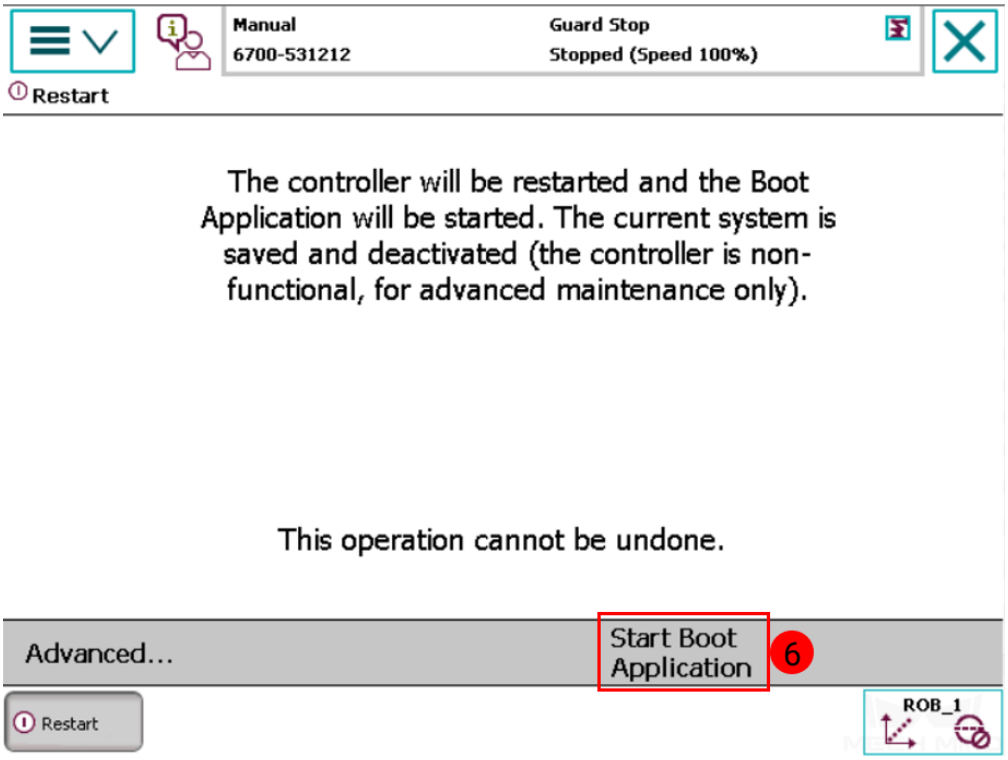
2. Select *Advanced*···



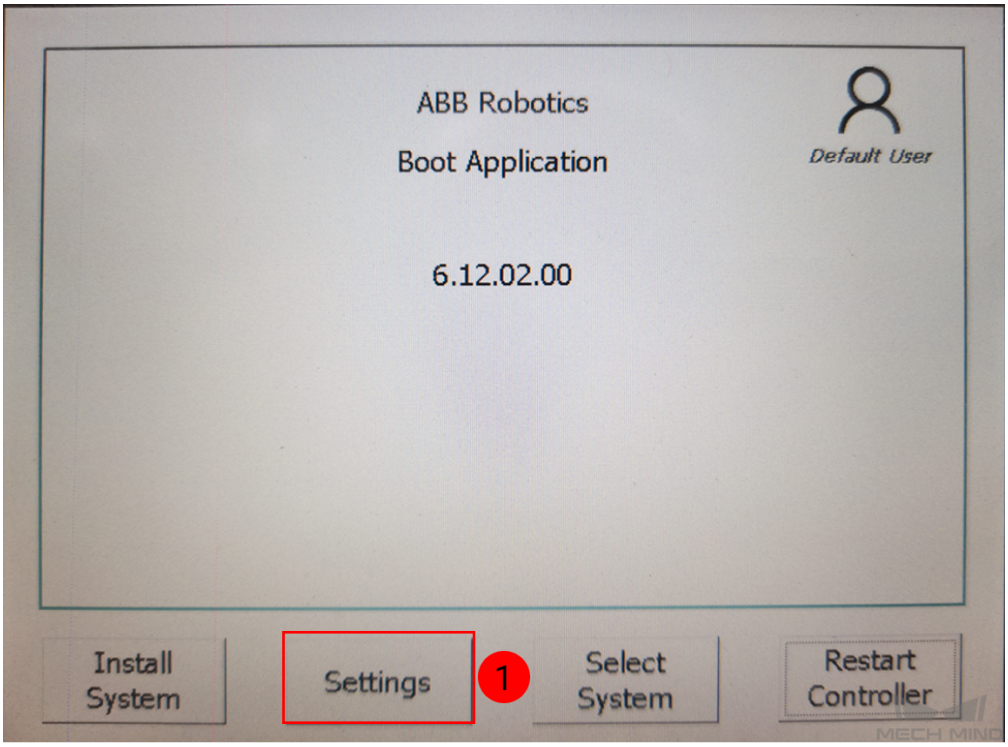
3. Select *Start Boot Application* and tap *Next*.



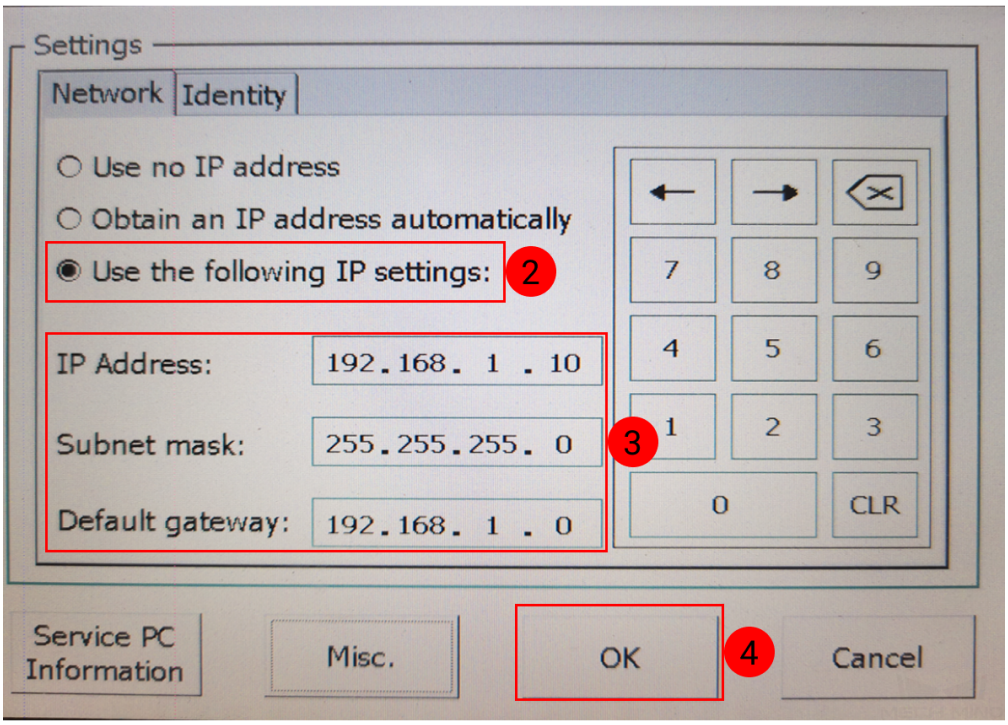
4. Select *Start Boot Application* to confirm.



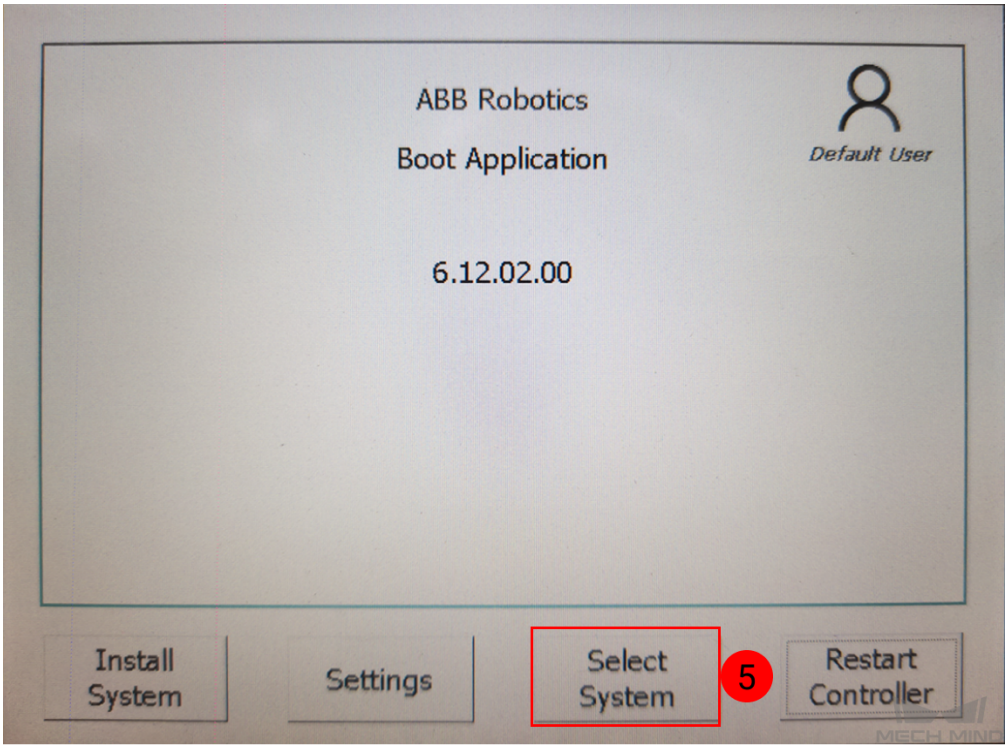
5. After restarting, you will see the interface as shown below. Tap *Settings*.



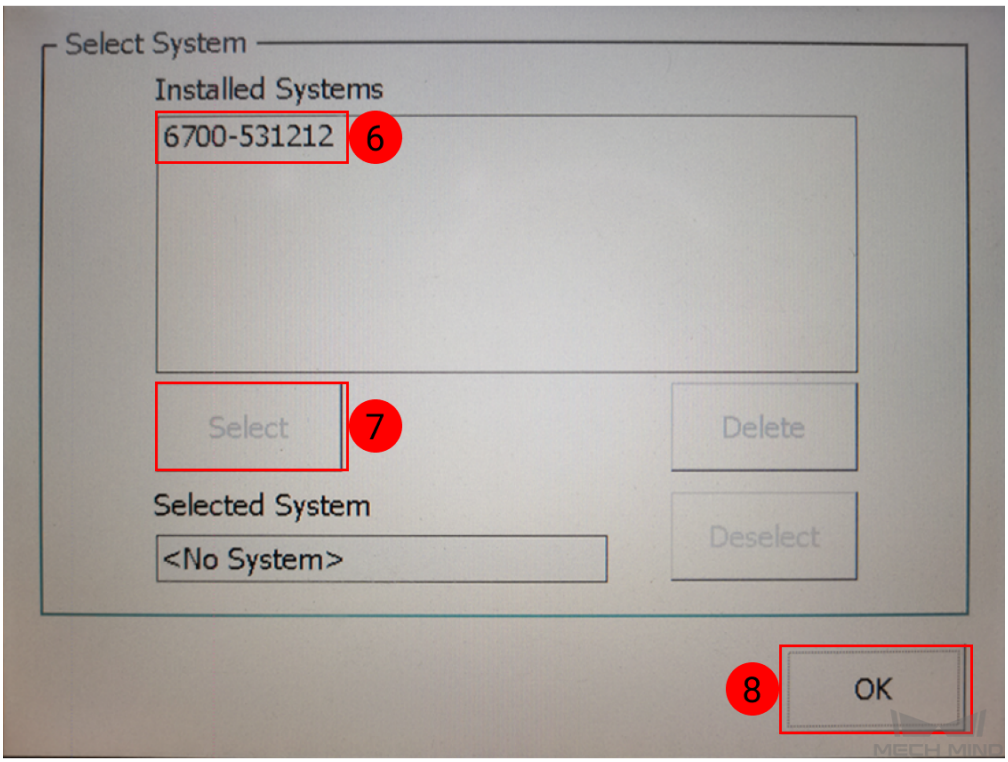
6. Select **Use the following IP settings** and configure the **IP Address**, **Subnet Mask**, and **Default gateway**. Tap *OK* after configuration.



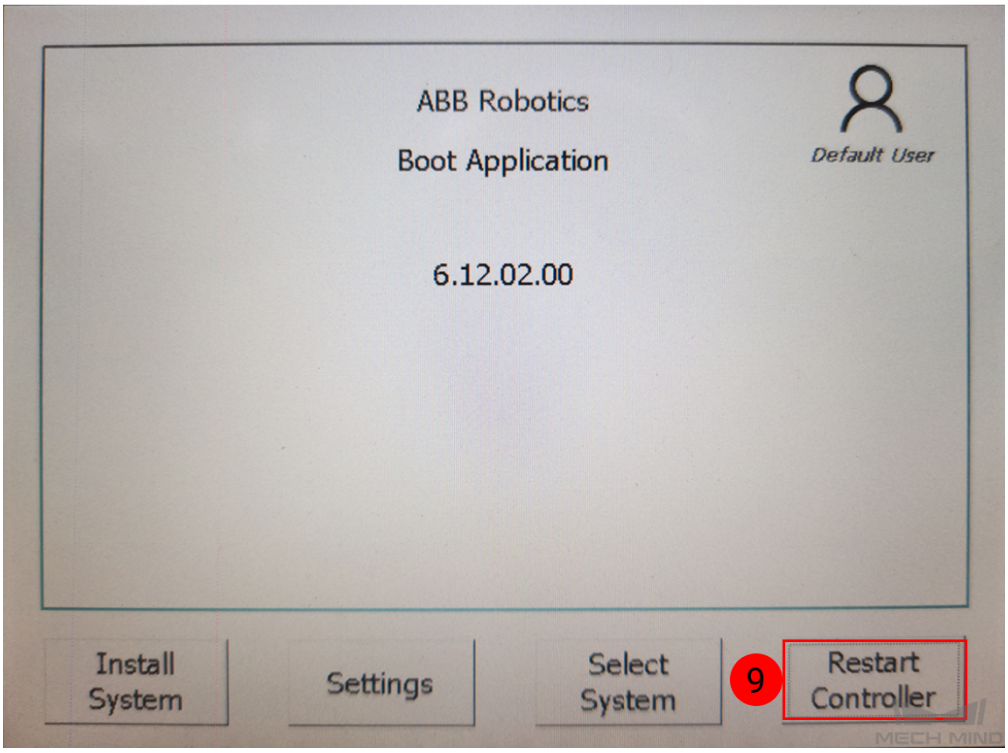
7. Tap *Select System*.



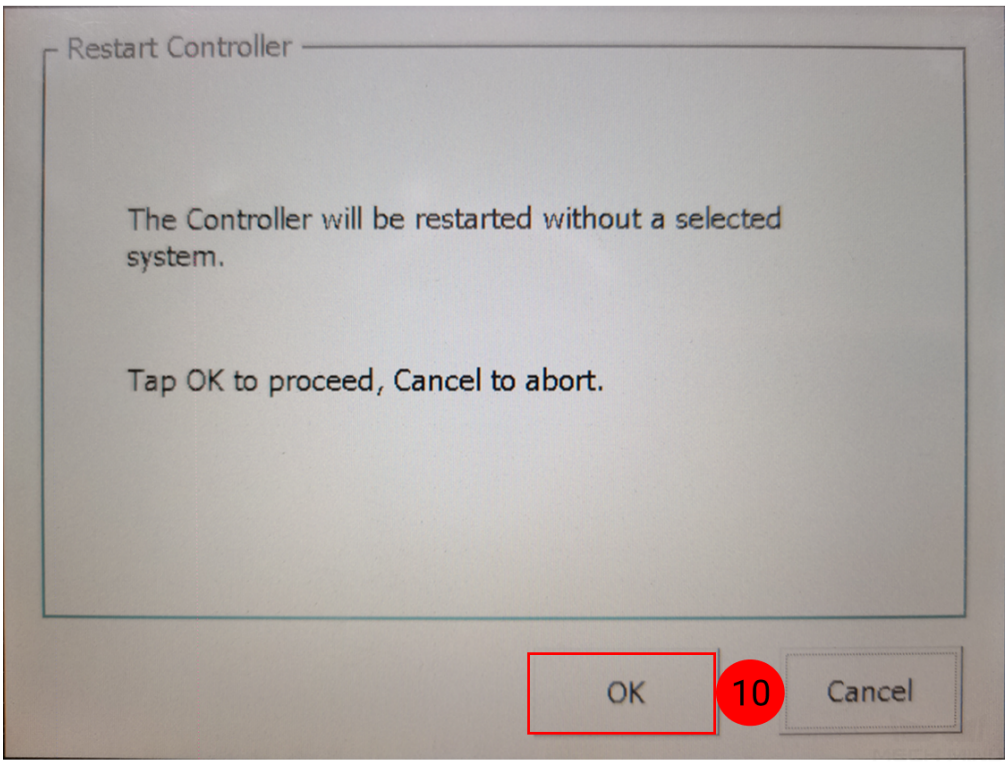
8. Select the system name in **Installed Syetems** box and then tap *Select*. Tap *OK* after configuration.



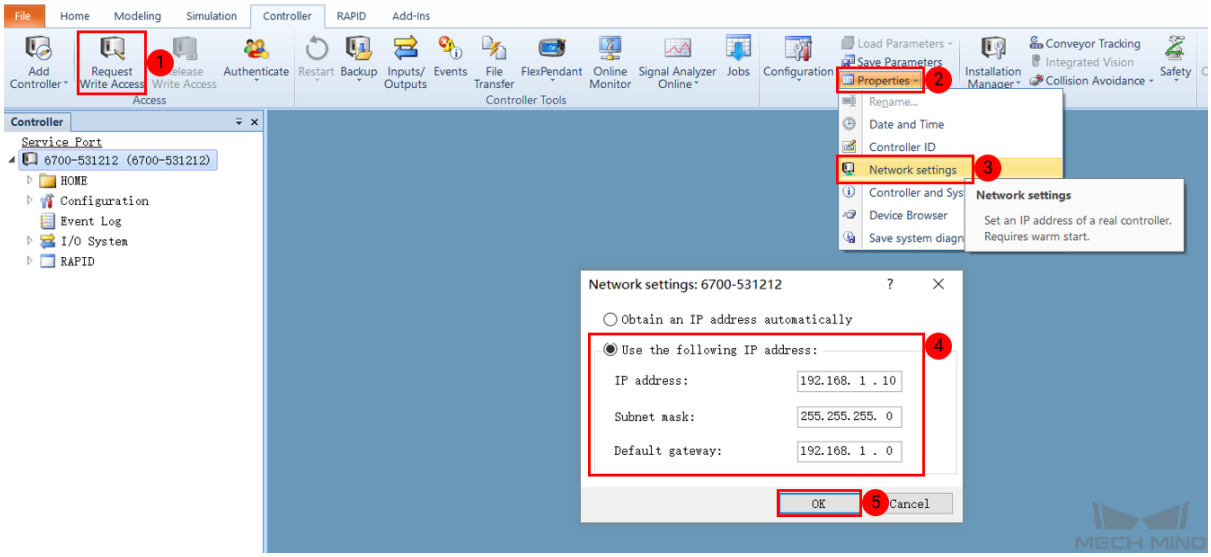
9. Select *Restart Controller*.



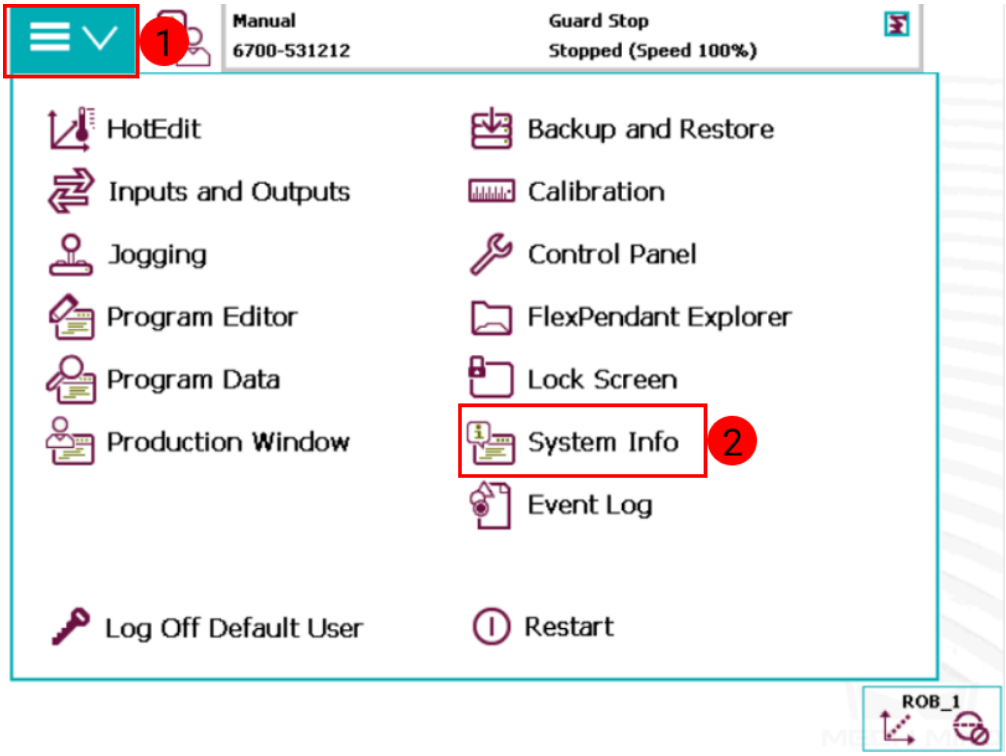
10. Tap *OK* to proceed.

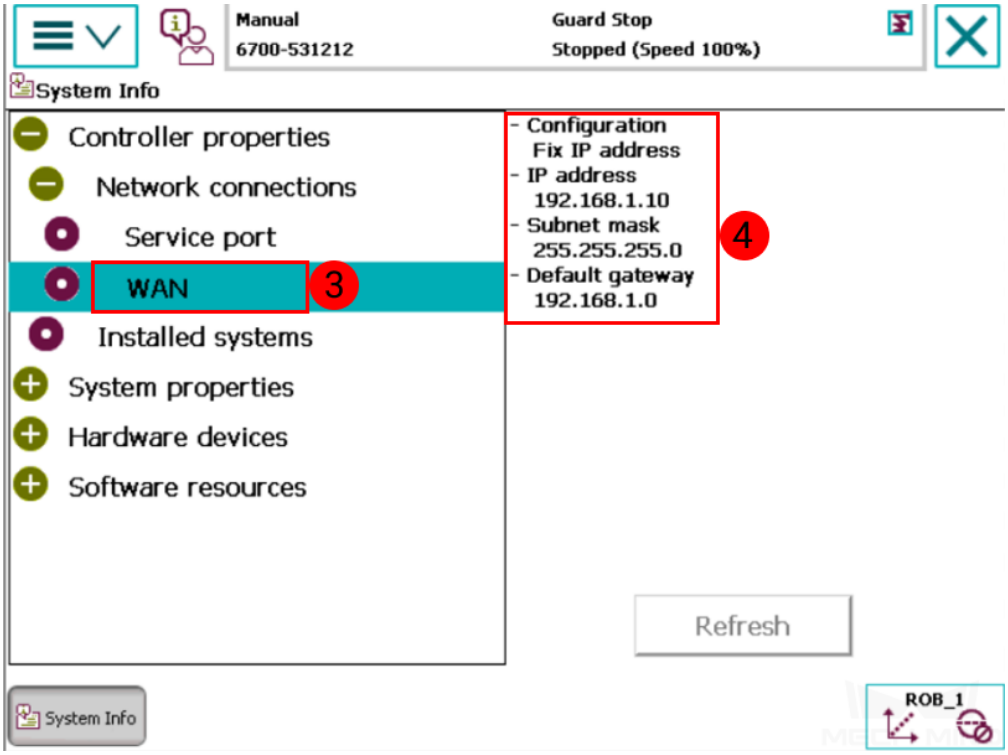


- Set the IP via RobotStudio
  1. Follow the steps as shown in the figure below to configure the robot IP, and restart the robot after configuration.



- Go to *Sytem Info*→ *Network connections*→ *WAN* to check if the IP configuration was successful after restarting.

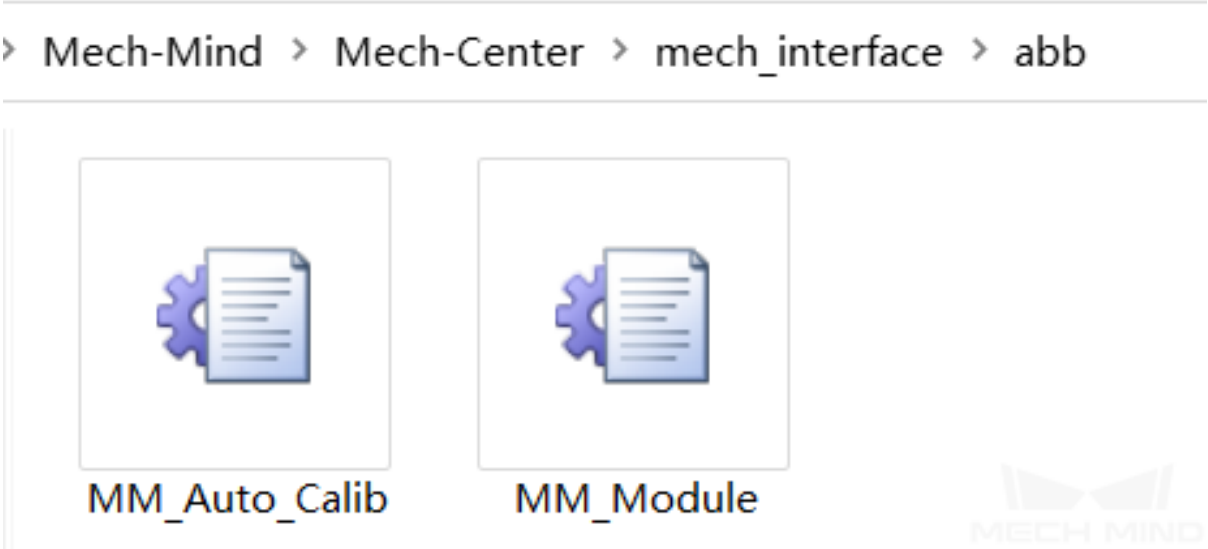




Load the Program Files

Prepare the Files

Copy **MM_Module.mod** and **MM_Auto_Calib.mod** from XXXX\Mech-Mind\Mech-Center\mech_interface\abb and paste them into the flash drive.

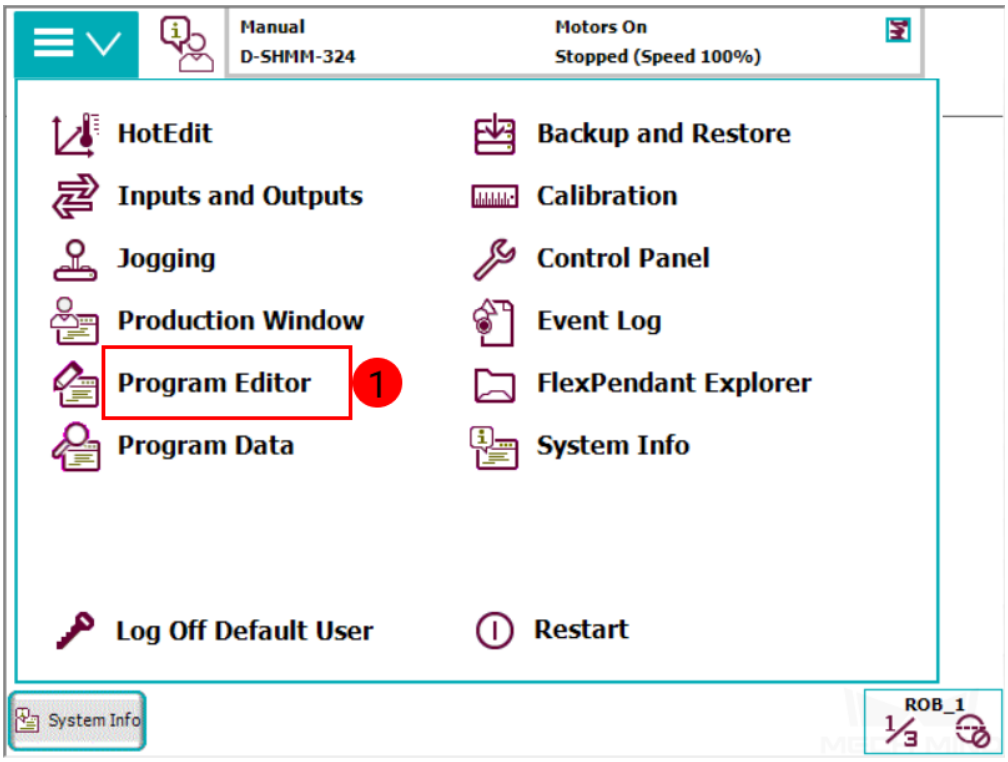


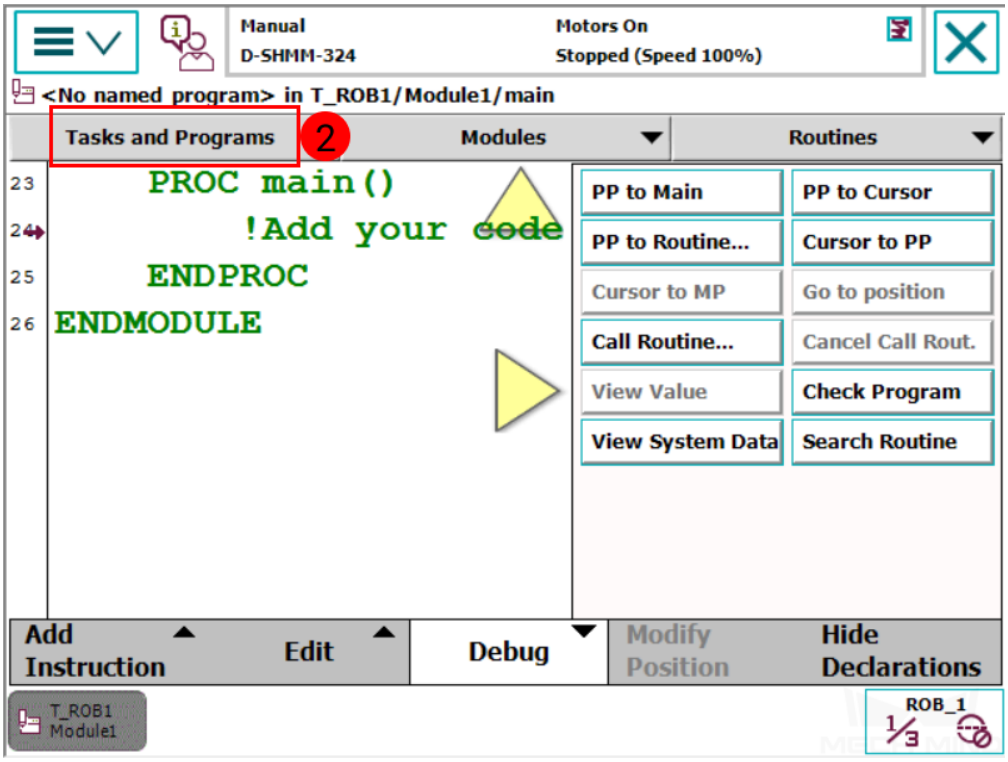
Hint: **MM_Module.mod** and **MM_Auto_Calib.mod** are program module files.

Load the Files to the Robot

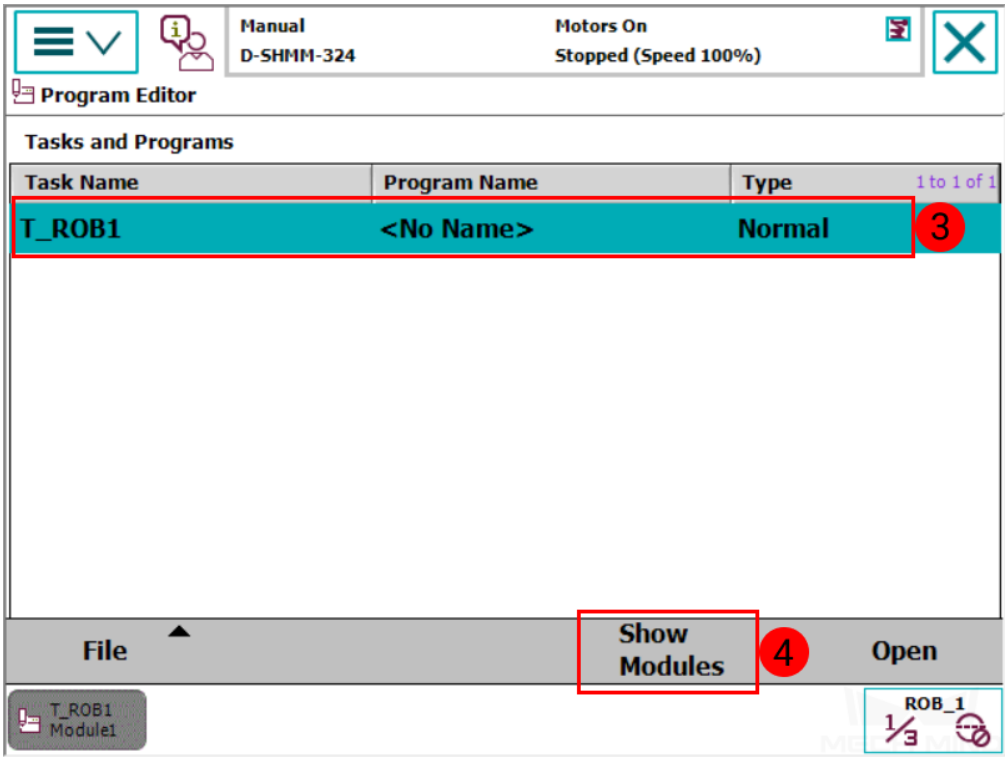
You can load the program modules (i.e., MM_Module.mod and MM_Auto_Calib.mod) with the teach pendant or via RobotStudio.

- Load the modules with the teach pendant
  1. Insert the flash drive into the USB port of the teach pendant.
  2. Go to *Program Editor* → *Tasks and Programs*.

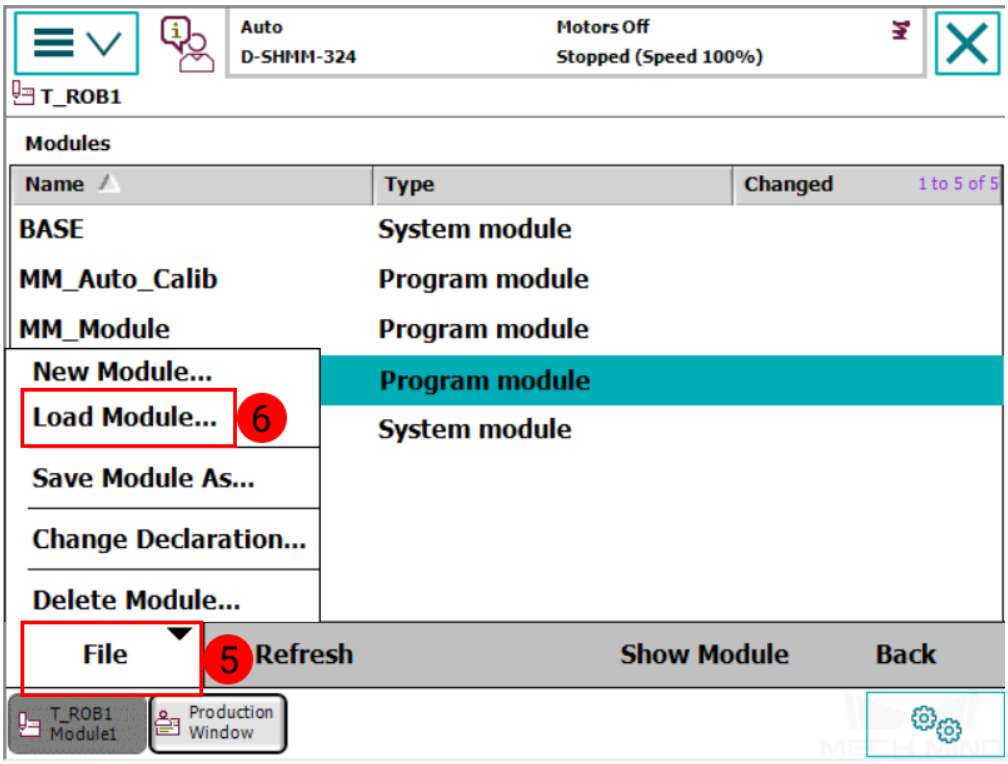




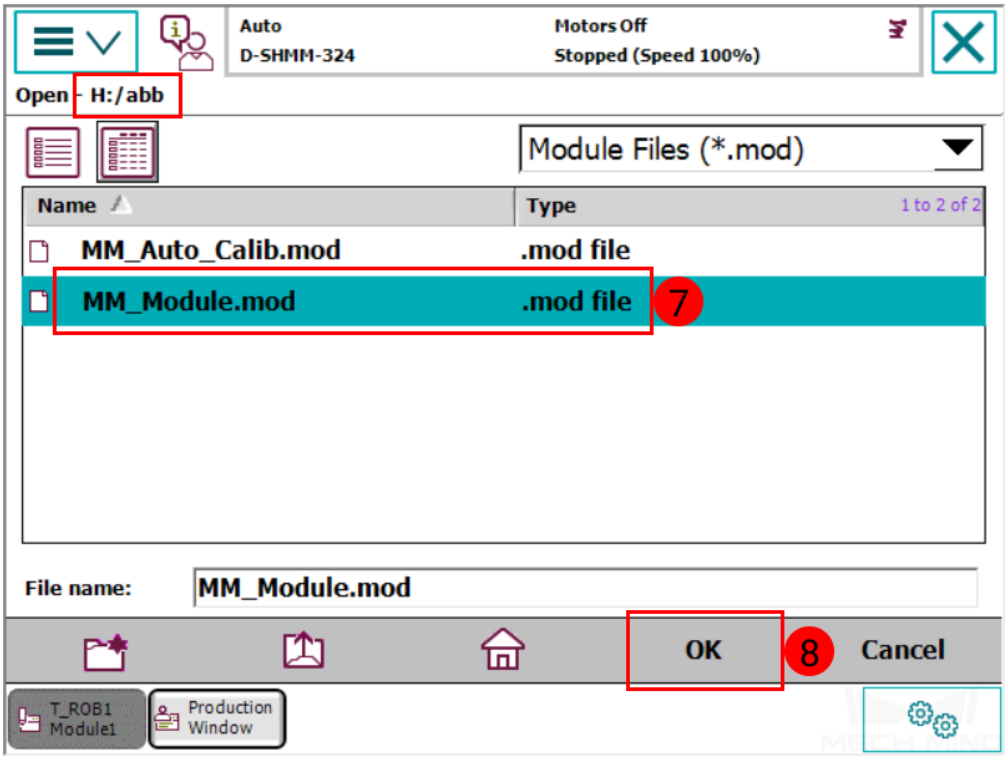
3. Select **T_ROB1** and tap *Show Modules*.



4. Select *File* → *Load Module*.

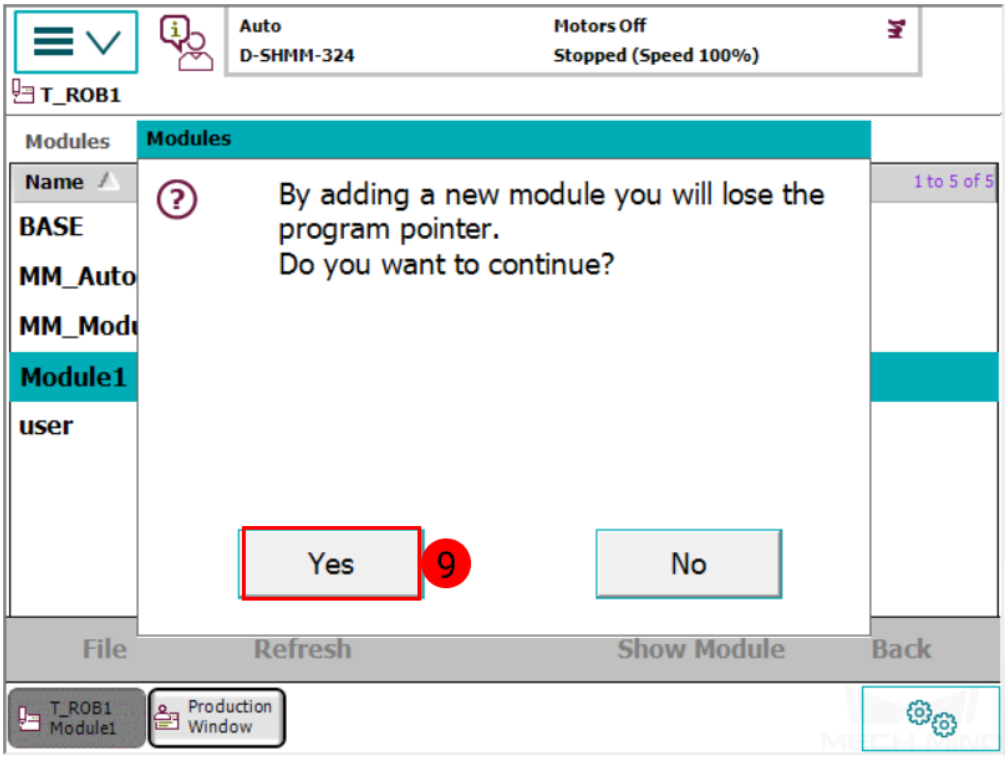


5. Select the module in the USB flash drive one at a time and tap *OK* to load the module.

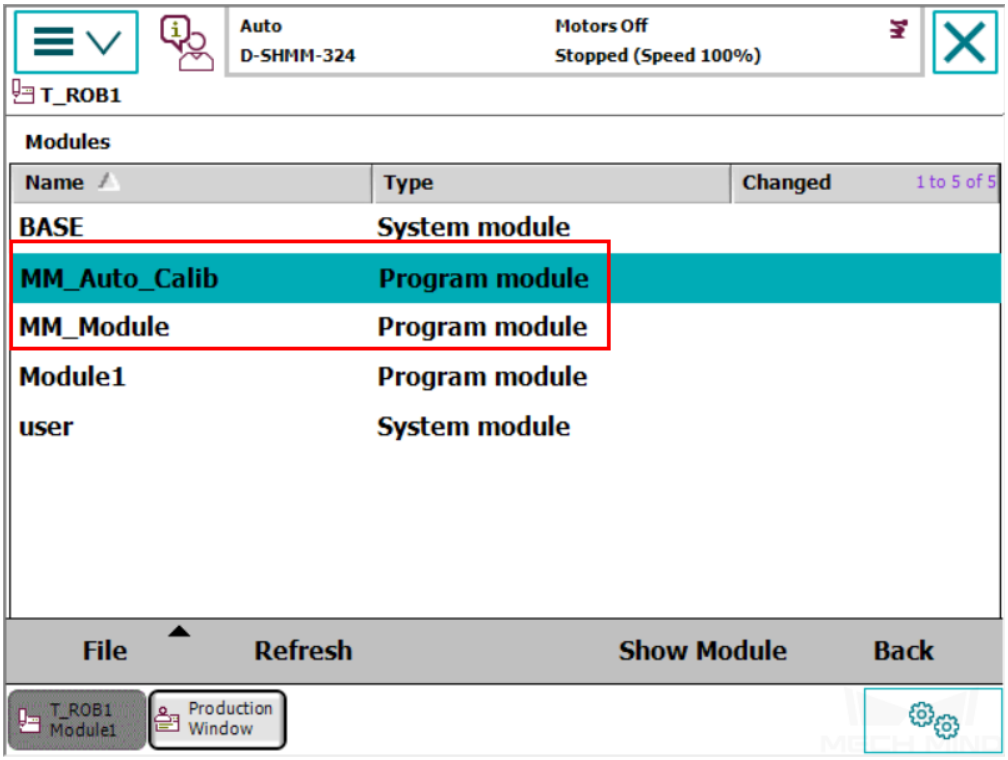


Hint: It is recommended to load the **MM_Module.mod** module before the **MM_Auto_Calib.mod** module, or else an error may occur. The instructions on loading the two modules are the same.

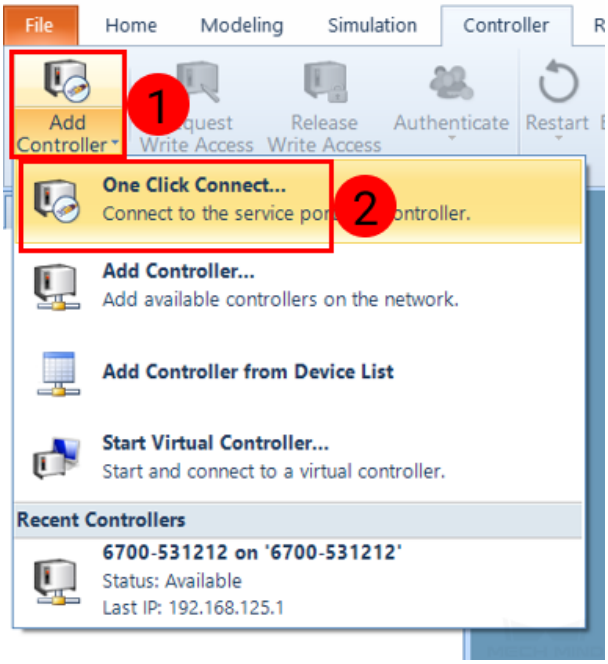
6. Select *Yes* in the pop-up window to confirm adding a new module.



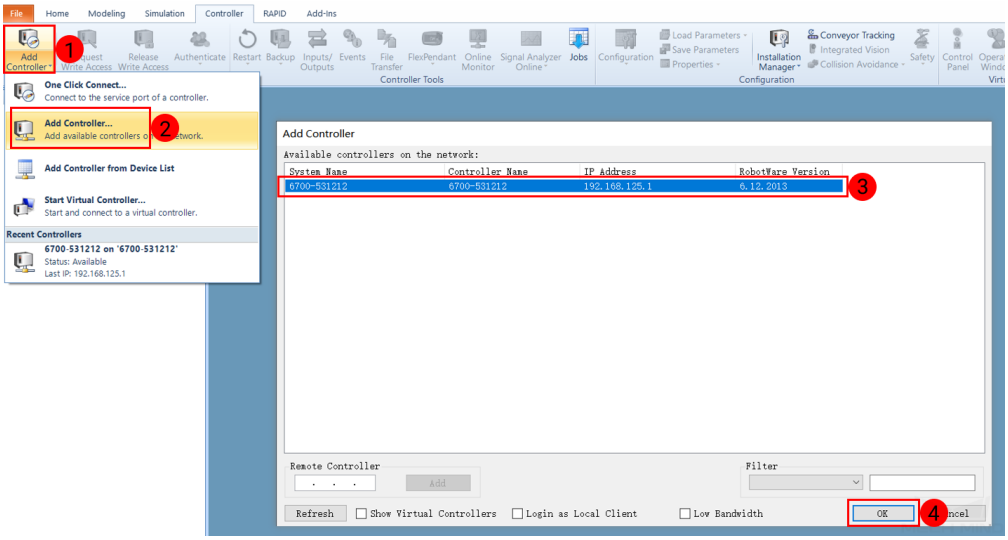
7. After loading successfully, you can see the two modules in **T_ROB1**.



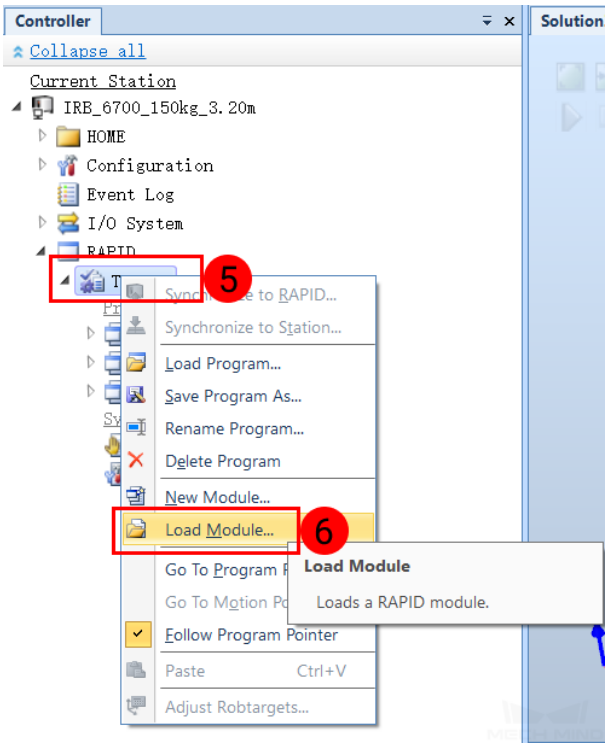
- Load the modules via RobotStudio
  1. Insert the flash drive into the USB port of the computer.
  2. Open RobotStudio and connect the robot.
     - If the robot controller is connected via the LAN port, click on **One Click Connect⋯**.

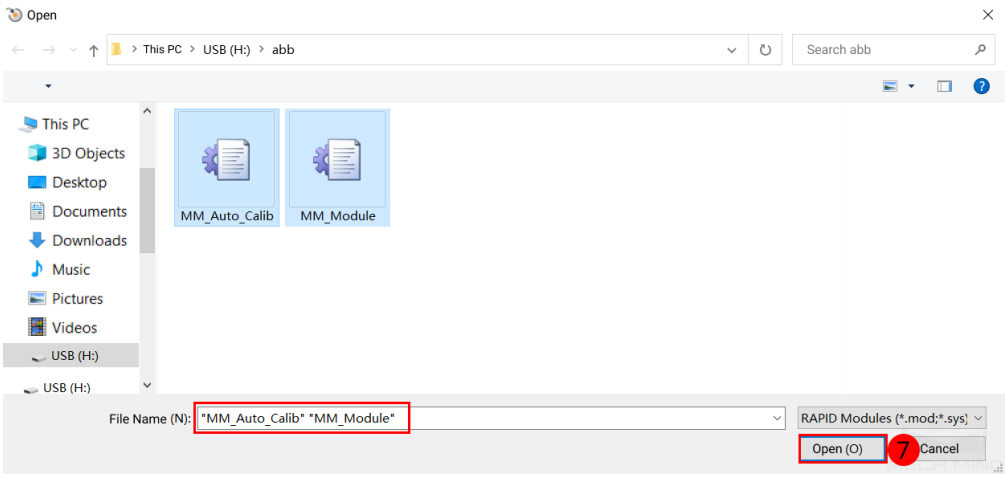


     - If the robot controller is connected via the WAN port or a switch, click on *Add Controller* and then select the controller and click on *OK*.

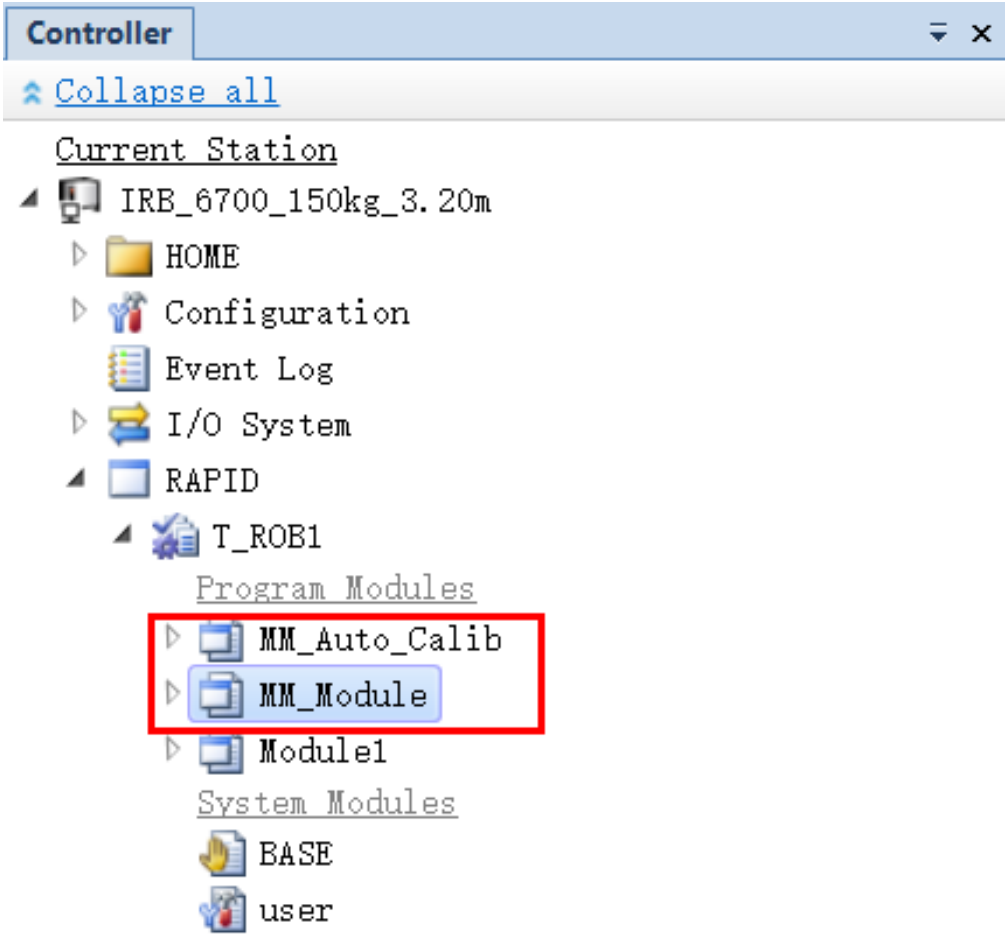


3. Click on *Controller* and right click on **T_ROB1**. Select **Load Module** in the context menu, and then open the two modules, as shown below.



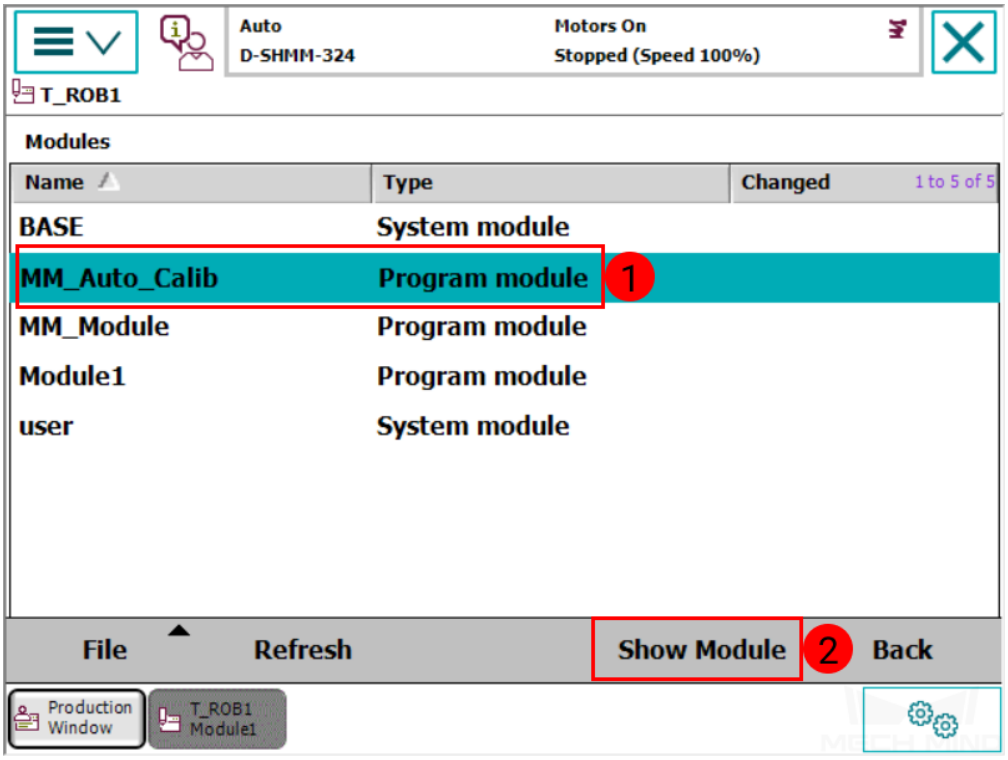


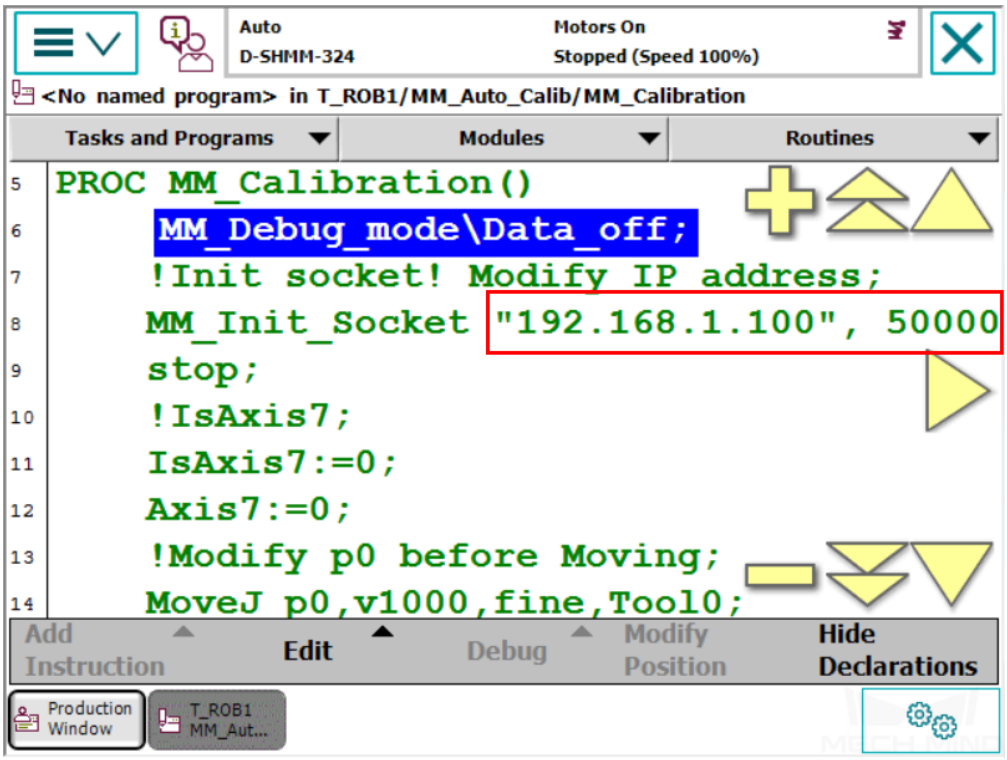
4. If **MM_Module.mod** and **MM_Auto_Calib.mod** appear in **T_ROB1**, the modules are loaded successfully.



Further Configurations

Locate and open **MM_Calibration** routine in **MM_Auto_Calib** module. Change the IP address in the program to the IP address of the IPC.

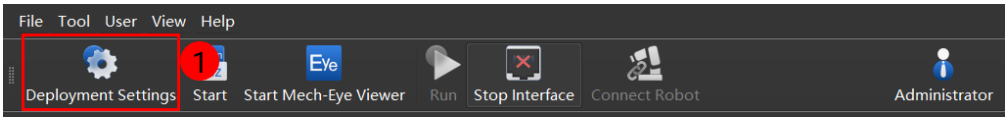

Auto
D-SHMM-324
Motors On
Stopped (Speed 100%)
T_ROB1
Modules
Name
Type
Changed
1 to 5 of 5
BASE
System module
MM_Auto_Calib
Program module
1
MM_Module
Program module
Module1
Program module
user
System module
File
Refresh
Show Module
2
Back
Production Window
T_ROB1 Module1

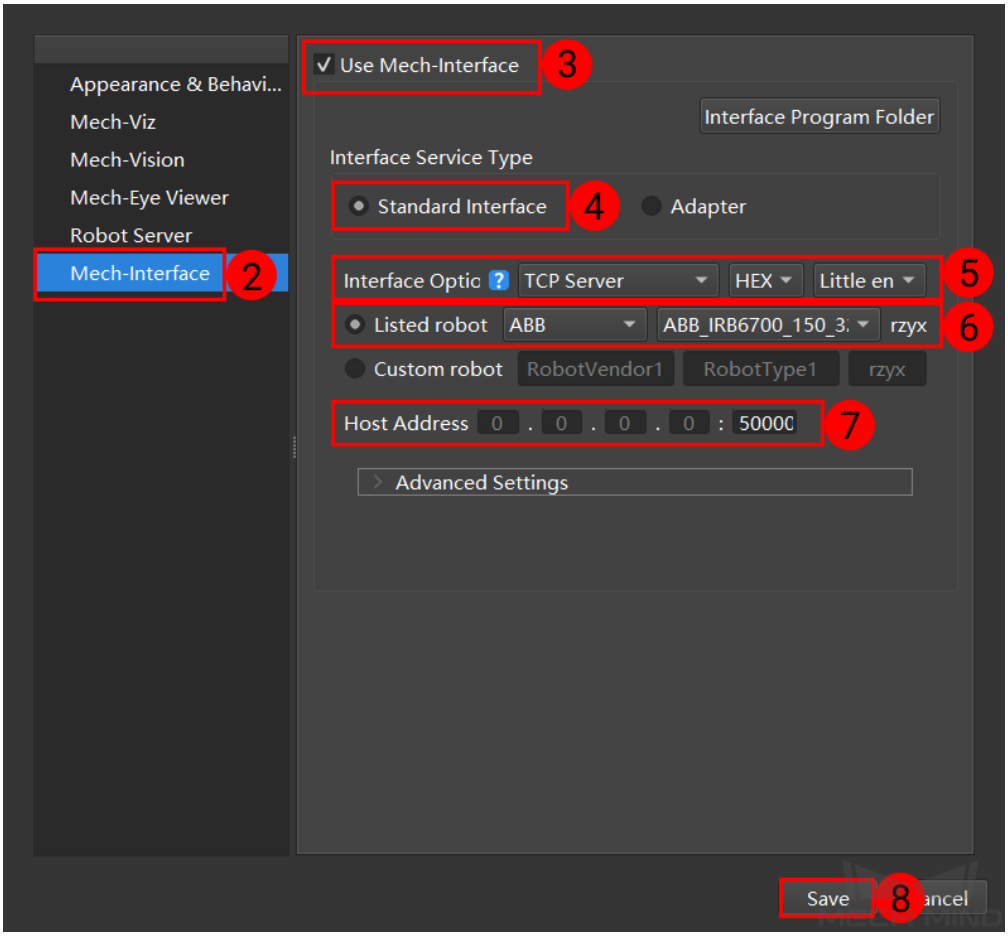

Auto
D-SHMM-324
Motors On
Stopped (Speed 100%)
<No named program> in T_ROB1/MM_Auto_Calib/MM_Calibration
Tasks and Programs
Modules
Routines
5 PROC MM_Calibration()
6 MM_Debug_mode\Data_off;
7 !Init socket! Modify IP address;
8 MM_Init_Socket "192.168.1.100", 50000
9 stop;
10 !IsAxis7;
11 IsAxis7:=0;
12 Axis7:=0;
13 !Modify p0 before Moving;
14 MoveJ p0,v1000,fine,Tool0;
Add Instruction
Edit
Debug
Modify Position
Hide Declarations
Production Window
T_ROB1 MM_Aut...

Test Robot Connection

Configuration in Mech-Center

1. Open Mech-Center and click on *Deployment Settings*.
2. Select *Mech-Interface*
3. Check **Use Mech-Interface**.
4. Select **Standard Interface**.
5. Select **TCP Server**, **HEX** and **Little endian** in Interface Option.
6. Select the robot model and click on *Save* to save the configurations.



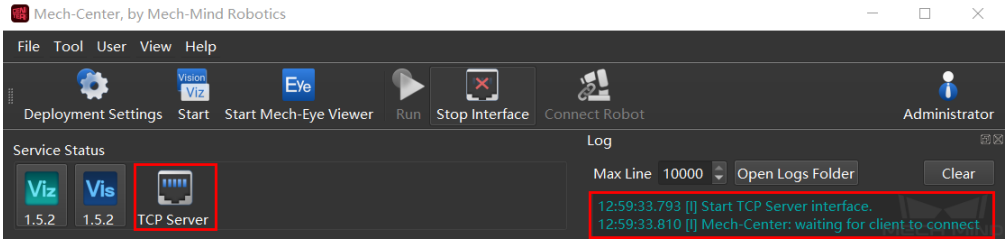


Note: The default port number is 50000. If it is modified, please modify the corre-

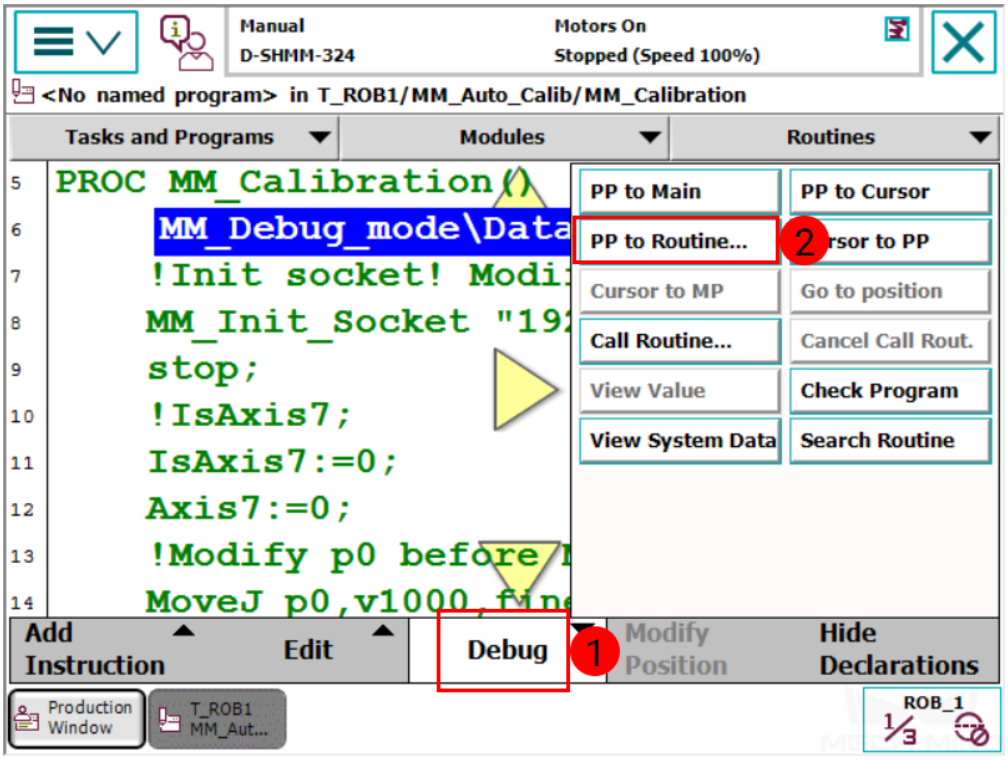
sponding code in the robot program when initializing communication.

Test the Connection

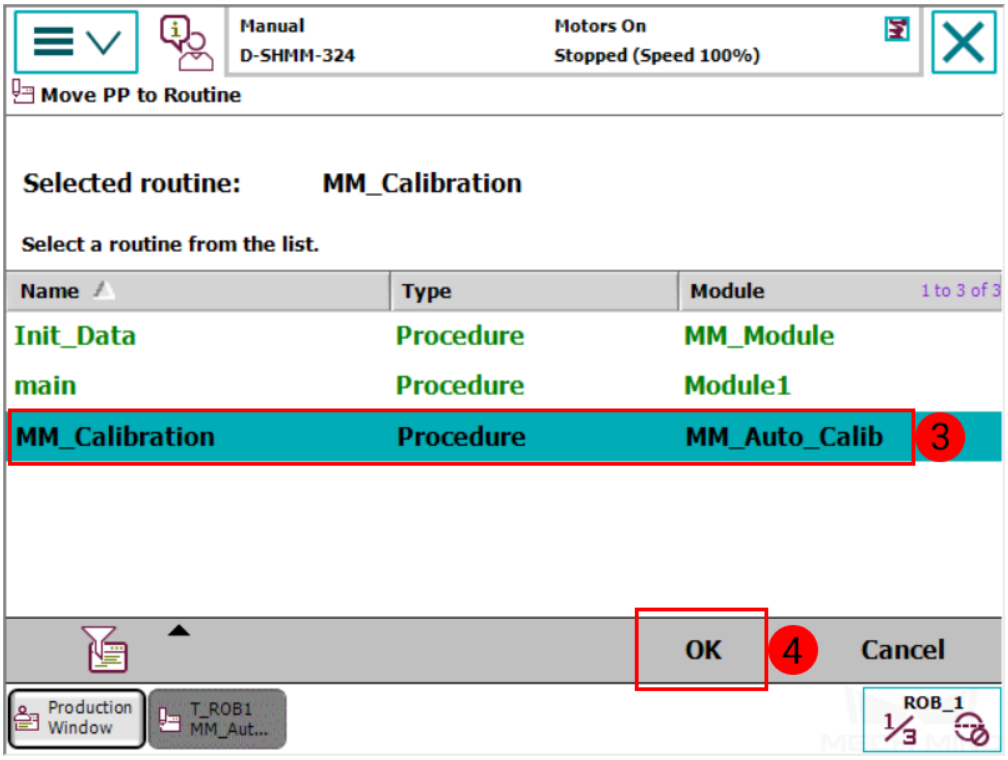
1. Start TCP Server interface in Mech-Center.



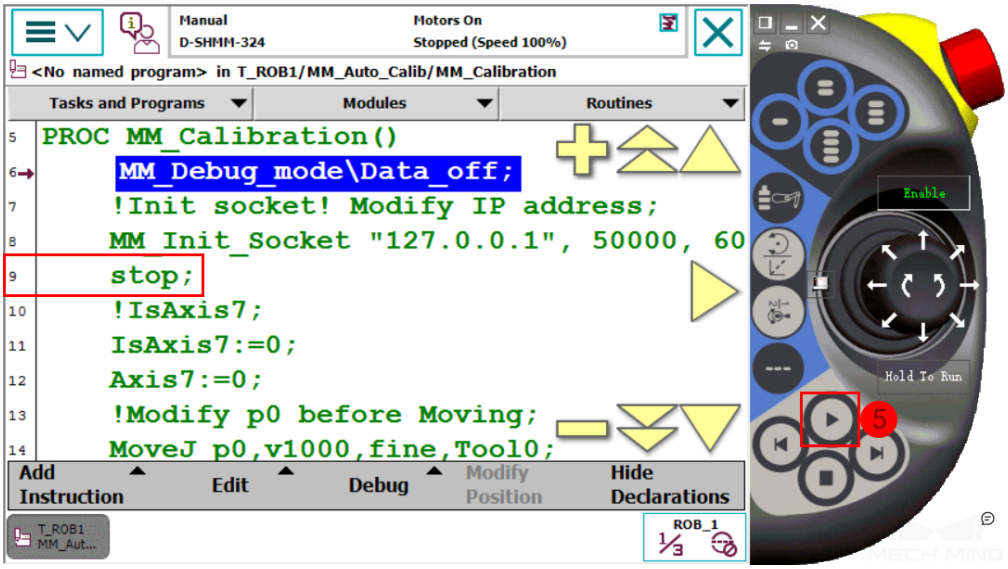
2. Tap *Debug* → *PP to Routine*.



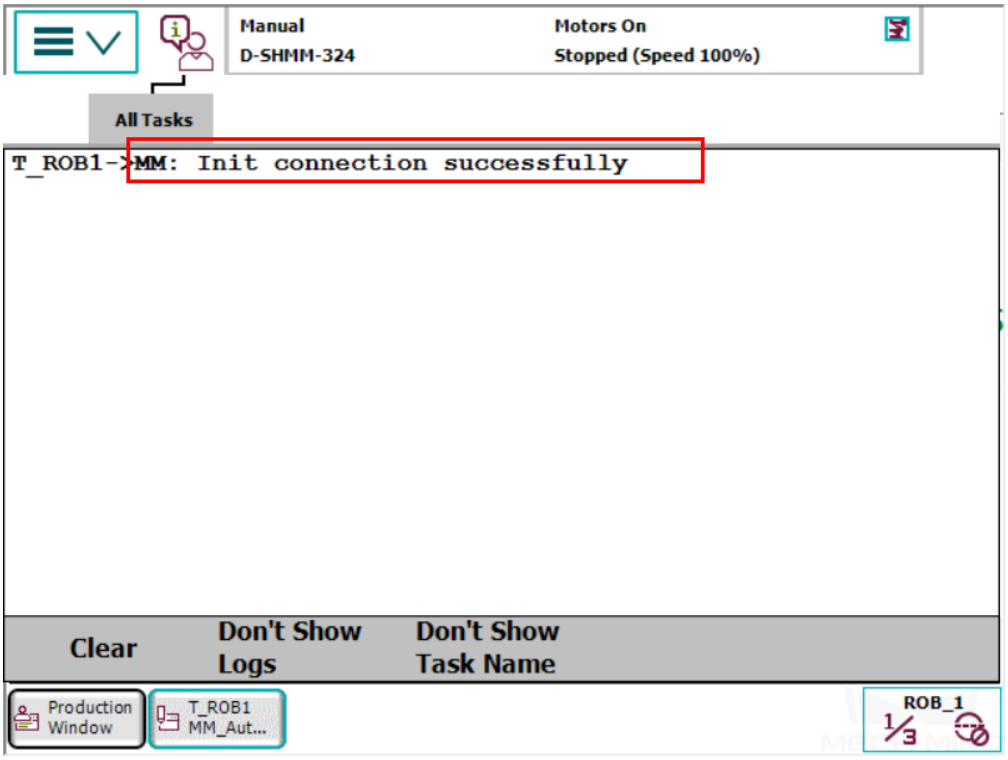
3. Select **MM_Calibration** and then tap *OK*.

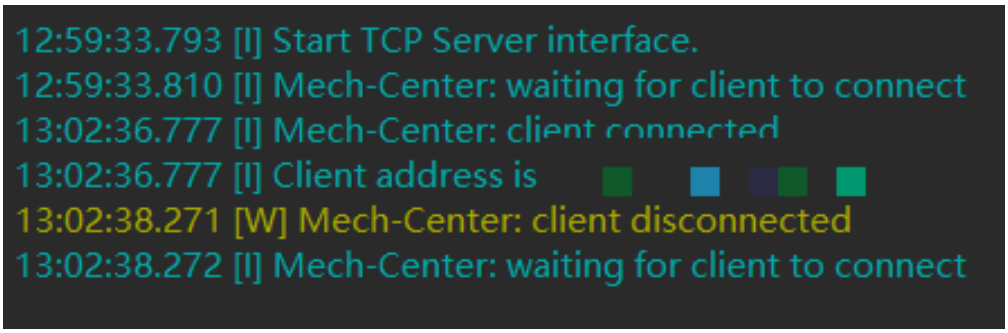


4. Press the  button on the teach pendant to execute the program until the PP moves to line 9.



5. If the messages as shown below appear respectively on the teach pendant and in the **Log** panel of Mech-Center, then the robot can be connected successfully.





2.1.2 ABB Calibration Program

This section introduces the process of calibrating the camera extrinsic parameters using the calibration program.

The process consists of 4 steps:

- *Select the Calibration Program*
- *Teach the Calibration Start Point*
- *Run the Calibration Program*
- *Start Calibration in Mech-Vision*

Before proceeding, please make sure that:

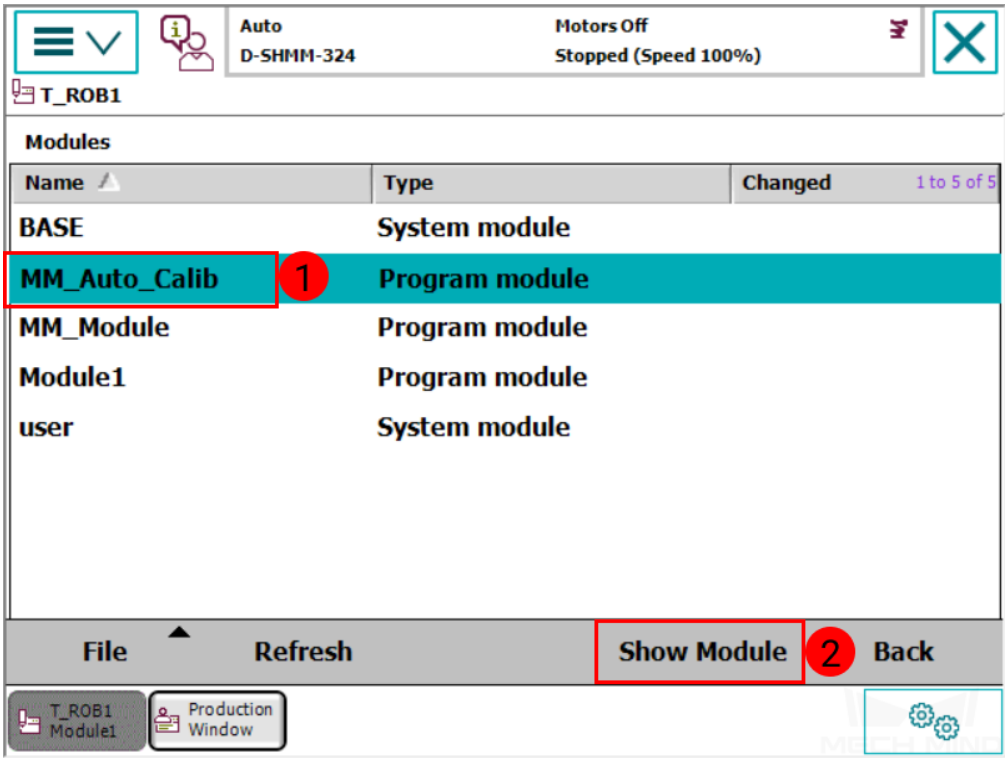
- You have *loaded the Standard Interface program* onto the robot and can establish communcation with Mech-Center.
- You are familiar with the contents in calibration_guide.

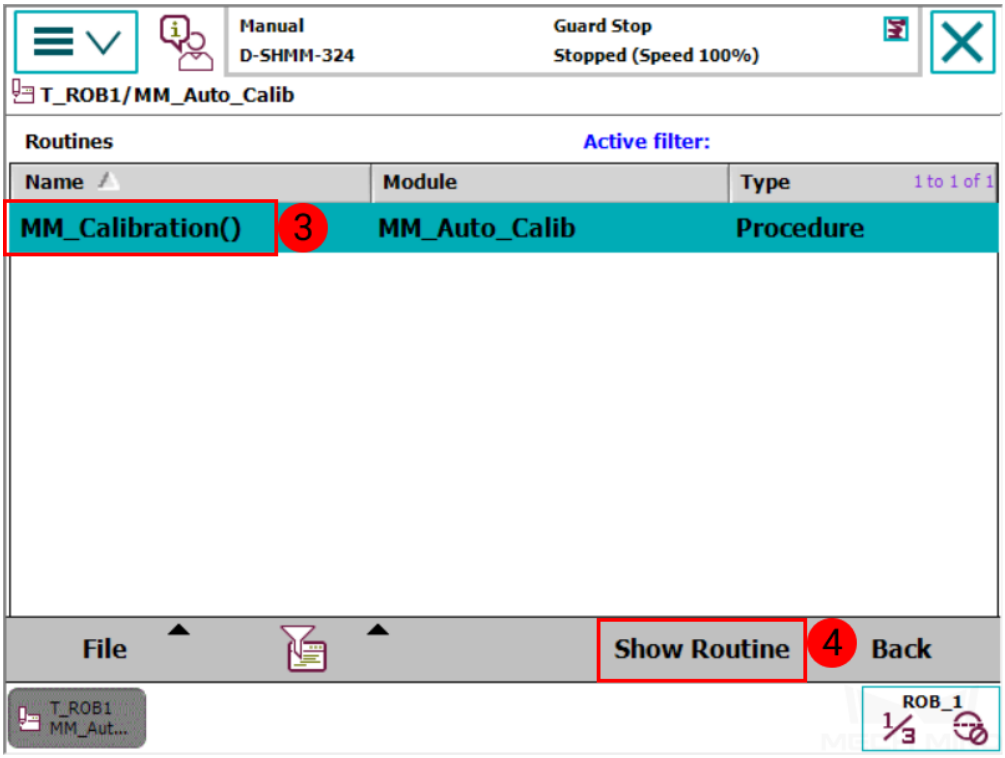
The calibration process introduced in this section is applicable to scenarios where standard interface is used to communicate and the extrinsic parameters need to be calibrated multiple times.

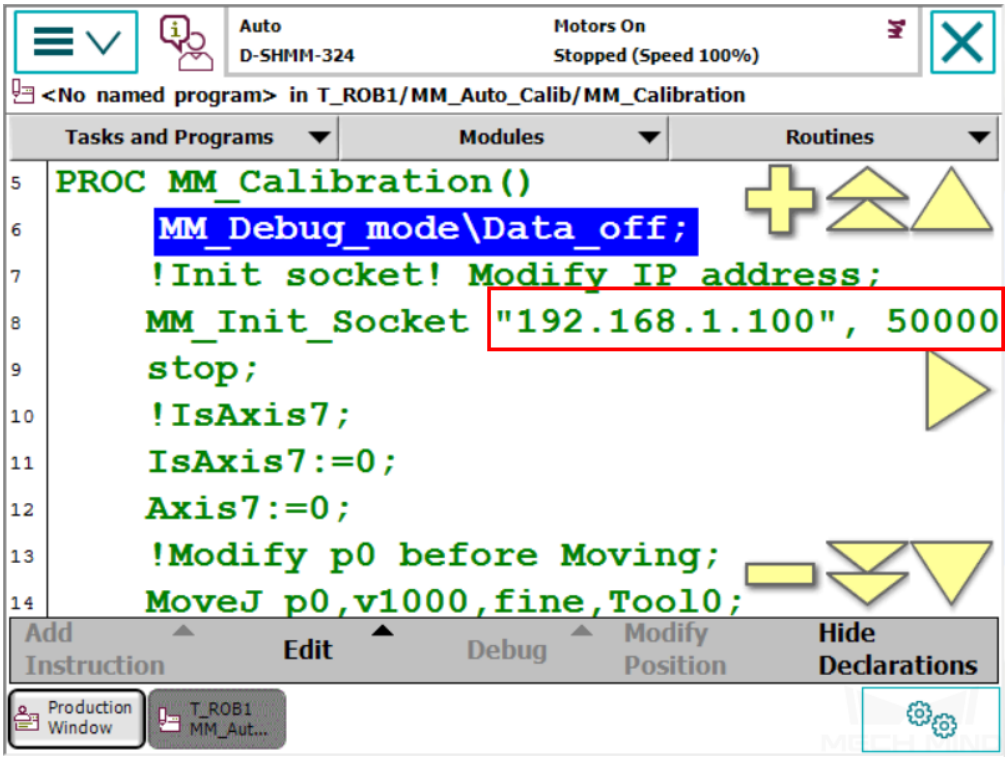
Calibration Process

Select the Calibration Program

Select and open the program module **MM_Auto_Calib** in **T_ROB1**. Please make sure that the IP address in the program is the same as that of the IPC, and the port number is set correctly.

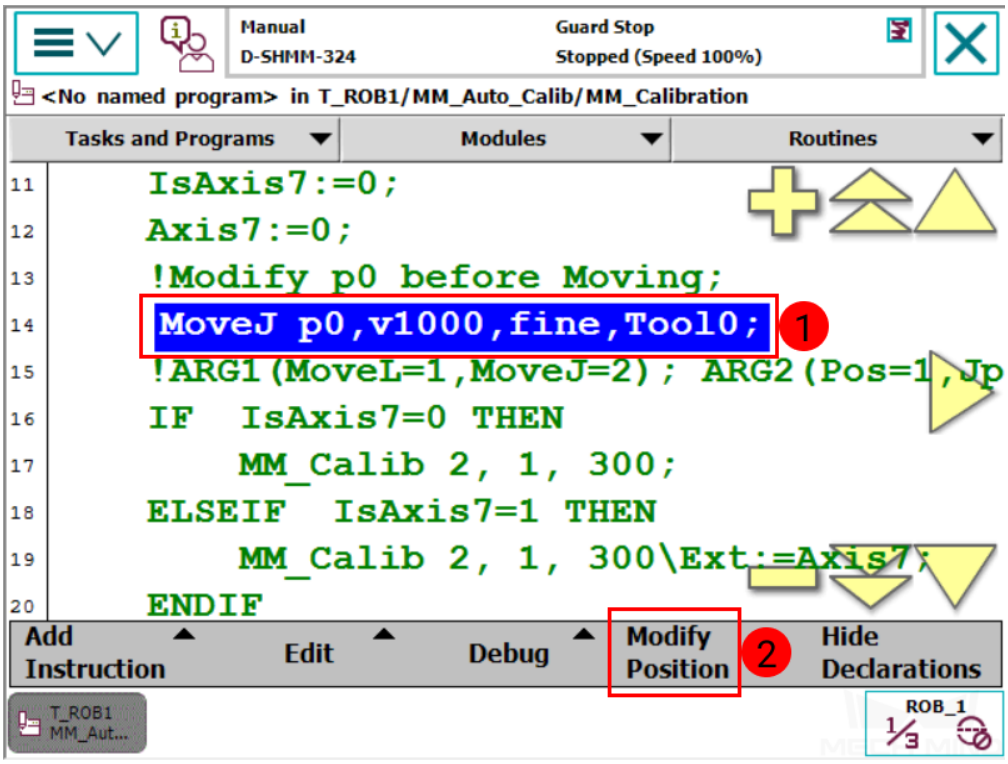


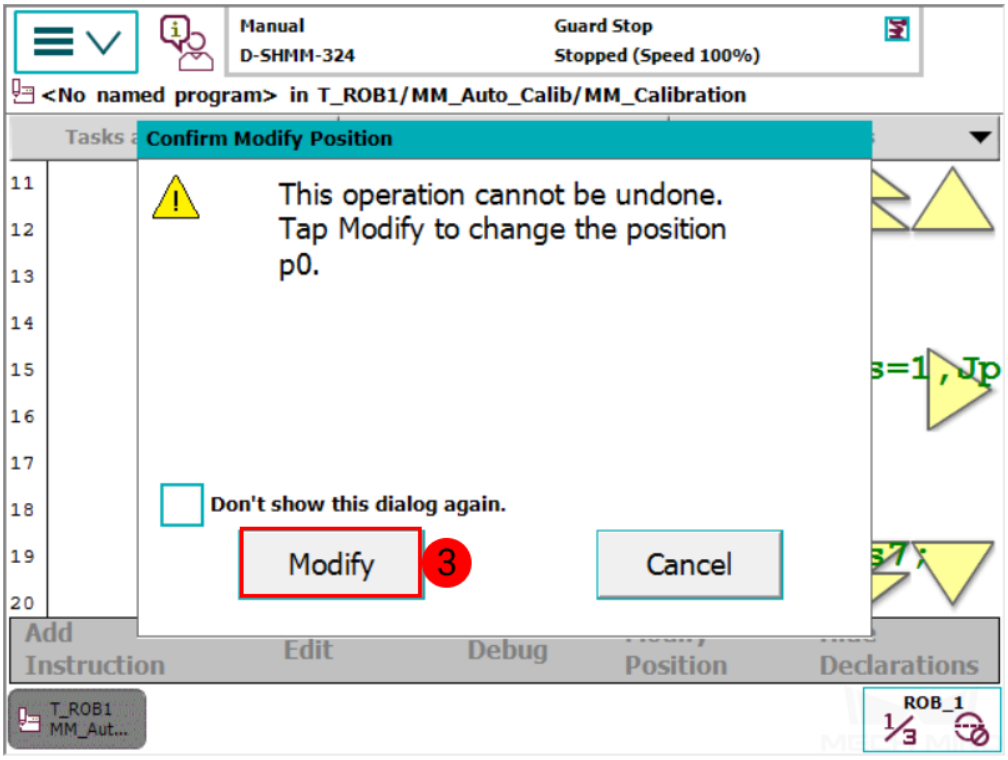

Manual
D-SHMM-324
Guard Stop
Stopped (Speed 100%)
T_ROB1/MM_Auto_Calib
Routines
Active filter:
Name
Module
Type
1 to 1 of 1
MM_Calibration()
3
MM_Auto_Calib
Procedure
File
Show Routine
4
Back
T_ROB1
MM_Aut...
ROB_1
1/3


Auto
D-SHMM-324
Motors On
Stopped (Speed 100%)
<No named program> in T_ROB1/MM_Auto_Calib/MM_Calibration
Tasks and Programs
Modules
Routines
5 PROC MM_Calibration()
6 MM_Debug_mode\Data_off;
7 !Init socket! Modify IP address;
8 MM_Init_Socket "192.168.1.100", 50000
9 stop;
10 !IsAxis7;
11 IsAxis7:=0;
12 Axis7:=0;
13 !Modify p0 before Moving;
14 MoveJ p0,v1000,fine,Tool0;
Add Instruction
Edit
Debug
Modify Position
Hide Declarations
Production Window
T_ROB1
MM_Aut...

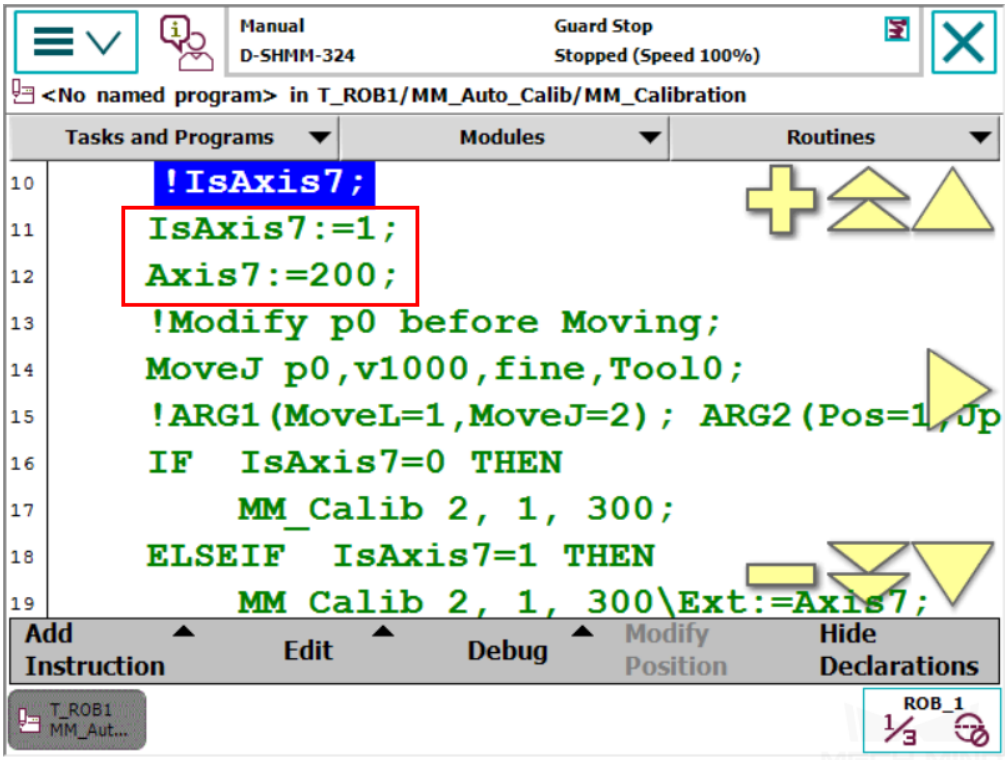
Teach the Calibration Start Point

Move the robot to the start point for the calibration, and then modify this position as **P0**. Press on *Modify* in the pop-up window to confirm.



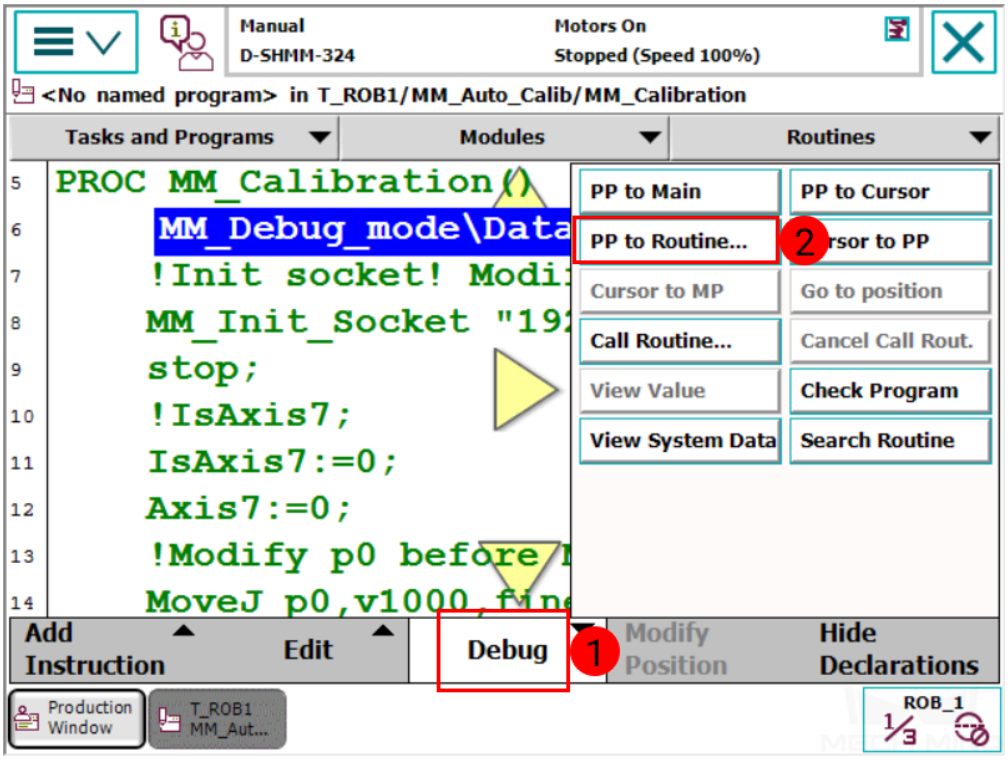


If the 6-axis robot is installed on a guide rail that is controlled by the robot, please set IsAxis7: =1, and configure the value of Axis7 according to actual situation, as shown below. If there is no guide rail on site or the guide rail is controlled by a PLC, please skip this step.

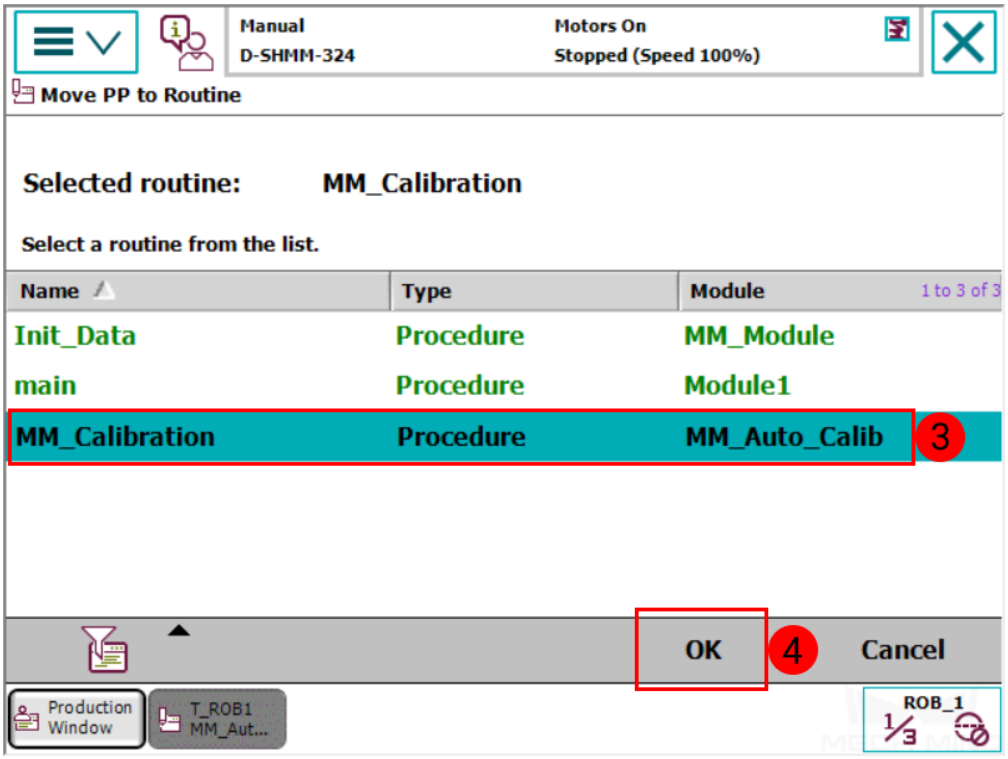


Run the Calibration Program

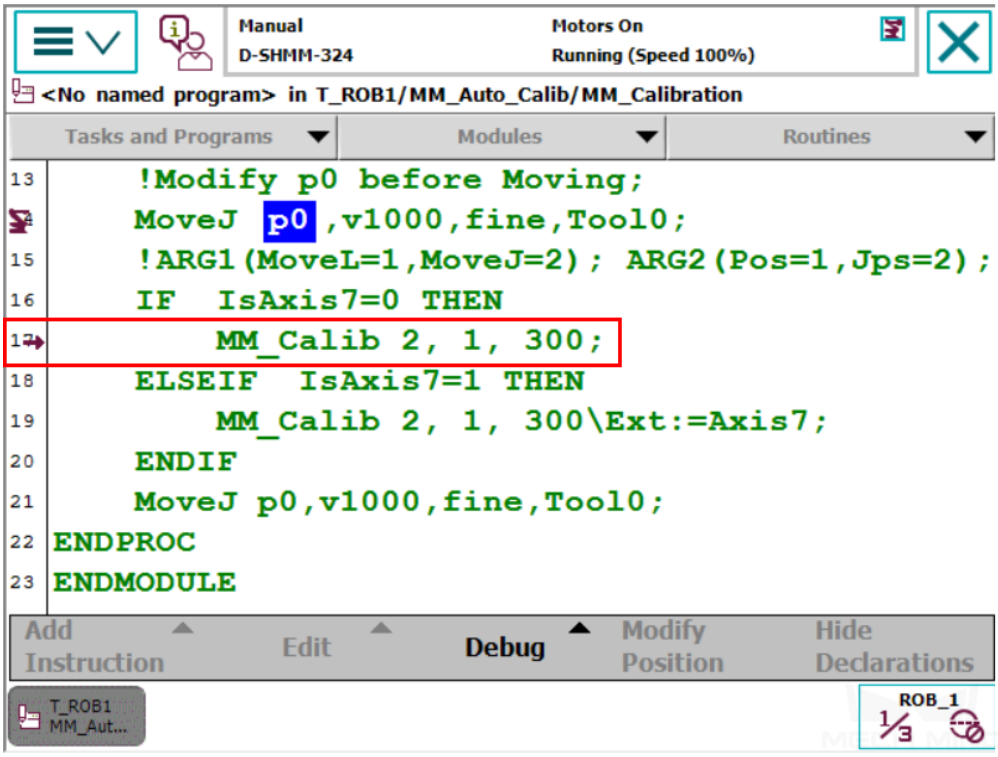
Press on *Debug* → *PP to Routine*.



1. Select **MM_Calibration**, and then press on *OK*.

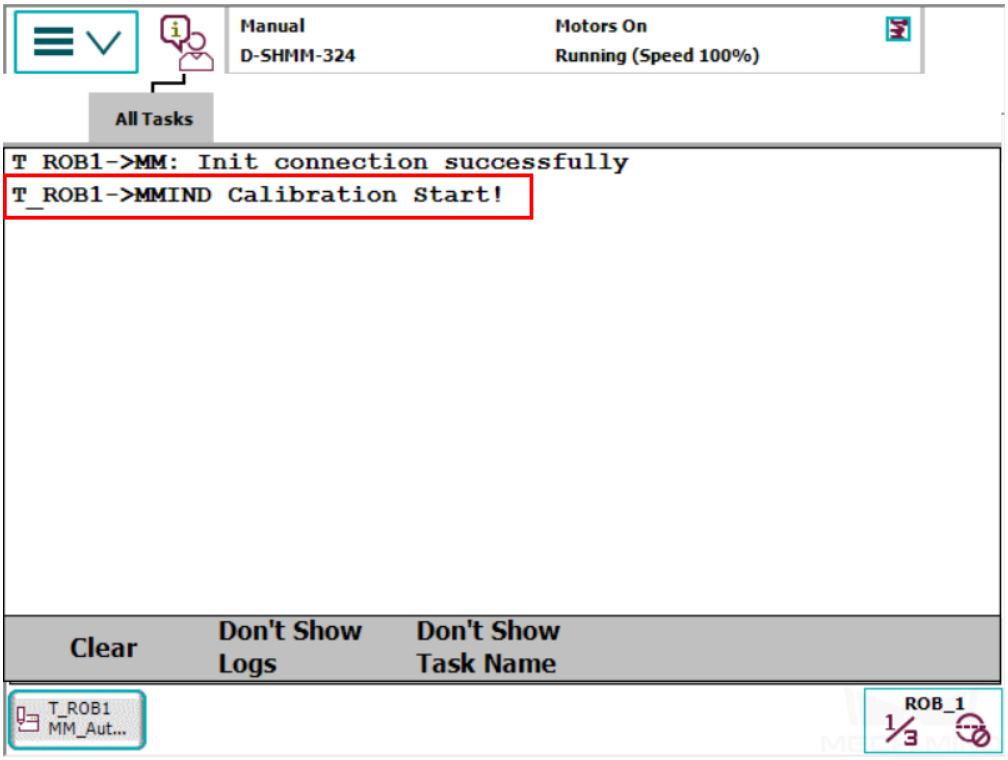


2. Press the  key on the teach pendant to run the program until the PP moves to line 17.

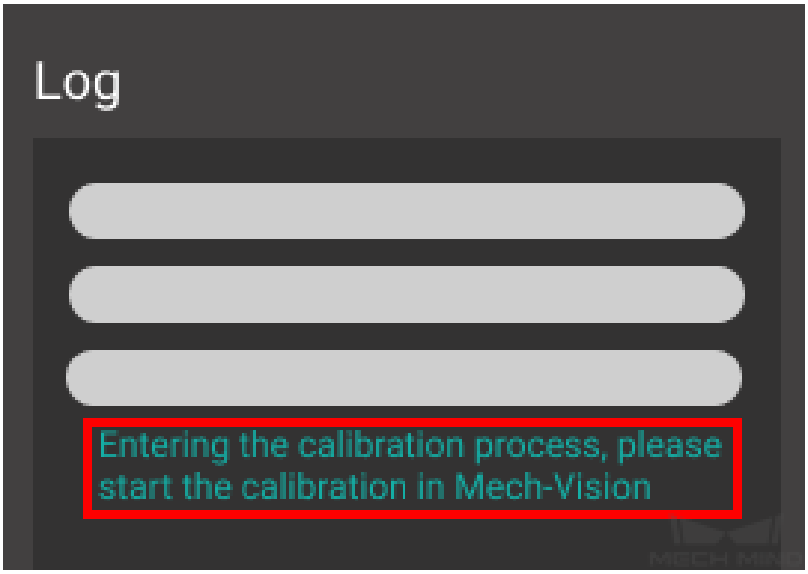


Proceed to the next section when the the following are displayed:

- On the teach pendant:

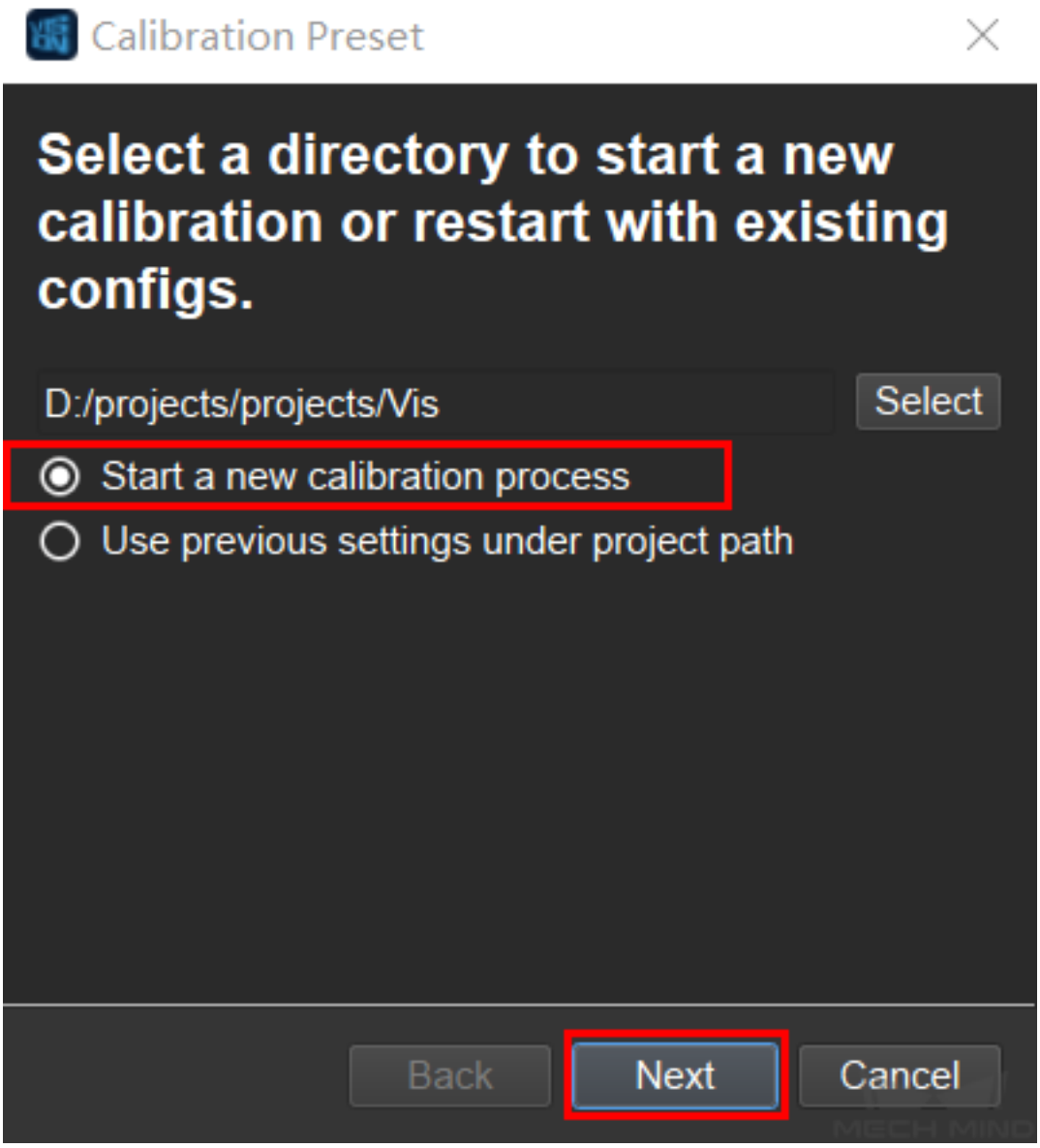


- In Mech-Center **Log** panel:



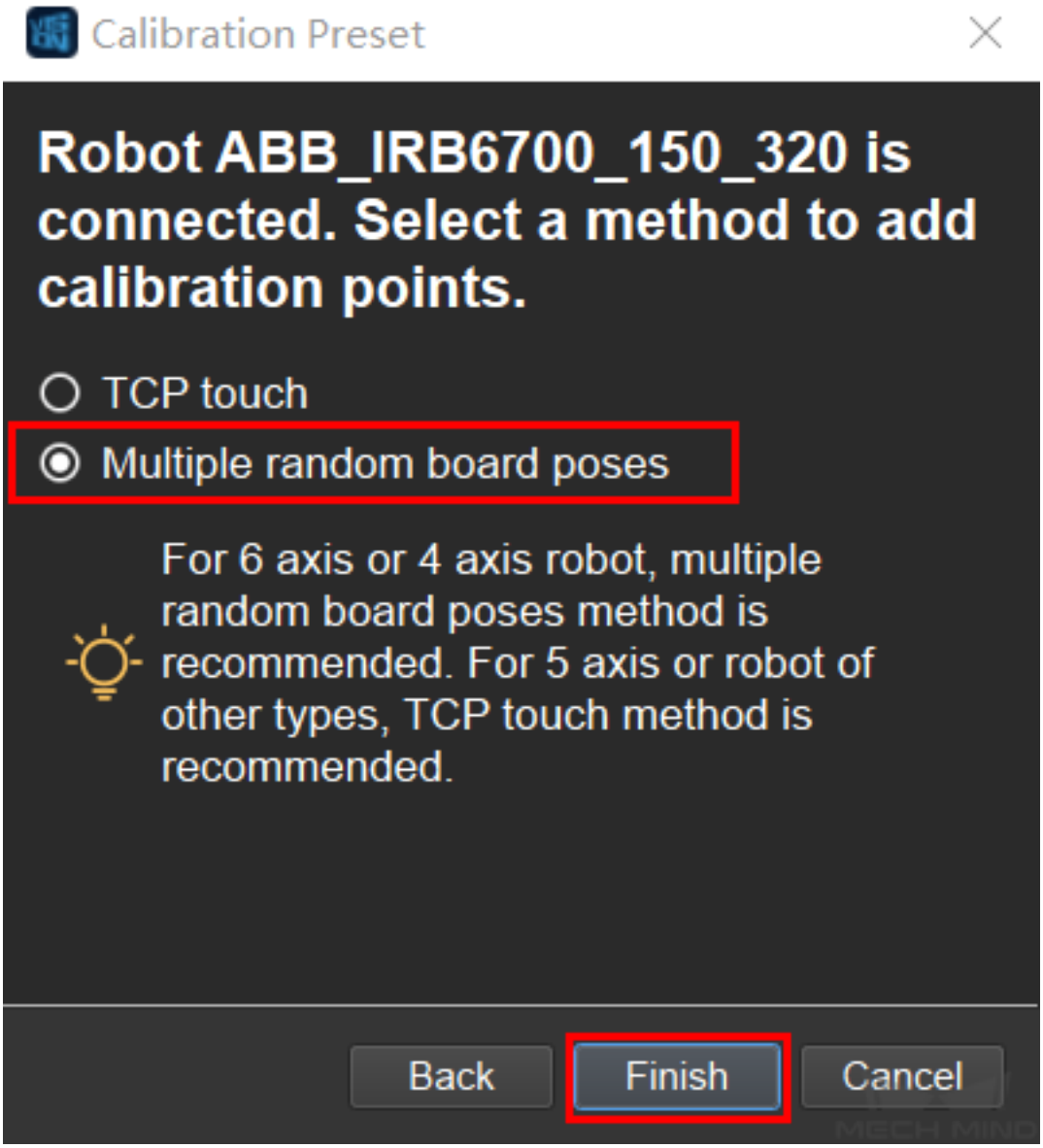
Start Calibration in Mech-Vision

1. In Mech-Vision, click on *Camera Calibration (Standard)* in the Toolbar, or select *Camera* → *Camera Calibration* → *Standard* from the Menu Bar.
2. Follow the instructions in Mech-Vision to complete the following configuration:
   1. Select **Start a new calibration process**;



   2. Select the camera mounting method;

3. Select **Multiple random board poses** for adding calibration points.



Note: If after selecting the camera mounting method, the window says **No robot is connected**, the connection between the robot and Mech-Center is not properly established. Please re-run the robot program.

3. Follow the instructions in Mech-Vision to finish the calibration.

Note: In **5 Add Marker-Images and Poses** after you click on *Move Robot along Trajectory and Add Board Images*, if the robot does not reach the next calibration point within 60 seconds, Mech-Vision will report a timeout error and stop the calibration process. In such case, please *select and run MM_Auto_Calib* on the teach pendant again, and restart the calibration process in

Mech-Vision.

2.1.3 ABB Example Program

This section introduces the example program provided with Mech-Center and the operations required to perform an actual pick-and-place task.

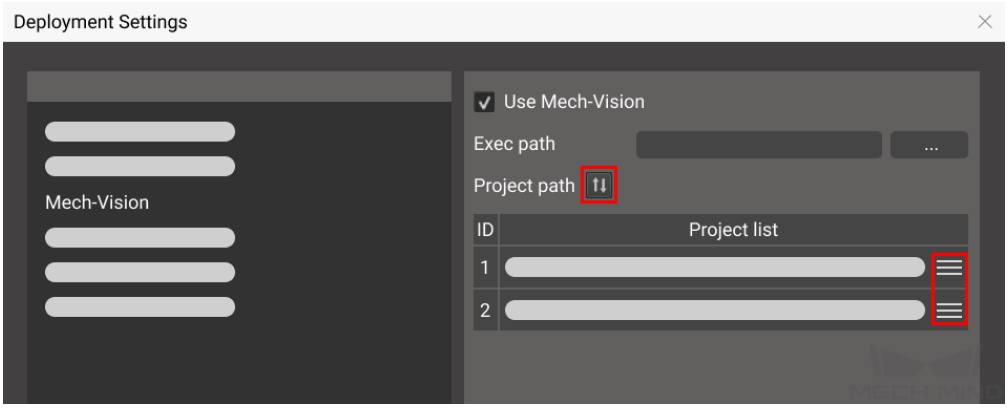
Please download the pick-and-place sample first.

Check the section corresponding to your own application setup:

- *Obtain Vision Results from Mech-Vision*
- *Obtain Planned Path from Mech-Viz*

Before running the program, please make sure that:

- You have *loaded the Standard Interface program* onto the robot and can establish communcation with Mech-Center.
- You have completed the extrinsic parameter calibration with *the calibration program* or by manually adding calibration points.
- Mech-Vision and Mech-Viz projects are created and set to autoload.
- The **Project list** in *Mech-Center* → *Deployment Settings* → *Mech-Vision* is synced by clicking on  , and the order of Mech-Vision projects have been adjusted according to actual needs.



- The TCP has been correctly specified.
- The robot speed is set to a low value, so that the operator can notice any unexpected behavior before accidents occur.

Obtain Vision Results from Mech-Vision

```
PROC Vision_Sample_1()
  !***********************************************************
  FUNCTION:Eye to Hand simple pick and place with Mech-Vision
  ! mechmind, 2022-5-1
  !***********************************************************
  AccSet 100, 100;
  VelSet 100, 5000;
  MoveAbsJ Home\NoEOffs,v3000,fine,Gripp1TCP;!move robot home position
  MoveL camera_capture,v500,fine,Gripp1TCP;
  PoseNum:=0;
  !Set ip address of IPC
  MM_Init_Socket "127.0.0.1",50000,60;
  WaitTime 0.1;
  !Set vision recipe
  MM_Switch_Model 1,1;
  !Run vision project
  MM_Start_Vis 1,1,2;
  WaitTime 1;
  MM_Get_VisData 1,LastData,PoseNum,Status;
  IF Status<>1100 THEN
      Stop;
  ENDIF
  MM_Get_Pose 1,pickpoint,label,speed1;
  MoveL RelTool(pickpoint,0,0,-100),v1000,fine,Gripp1TCP;
  MoveL pickpoint,v300,fine,Gripp1TCP;
  Stop;
  !Add object grasping logic here.
  MoveL Offs(pickpoint,0,0,100), v1000, fine, Gripp1TCP;
  MoveL waypoint1,v1000,z50,Gripp1TCP;
  MoveL RelTool(drop,0,0,-100),v1000,fine,Gripp1TCP;
  MoveL drop,v300,fine,Gripp1TCP;
  Stop;
  !Add object releasing logic here.
  MoveL Offs(drop,0,0,100), v1000, fine, Gripp1TCP;
  MoveAbsJ Home\NoEOffs,v3000,fine,Gripp1TCP;
  RETURN ;
ENDPROC
```

Program Logic

1. Move the robot to HOME position.
2. Move the robot to the image capturing pose.
3. Initialize communication with **MM_Init_Socket**.
4. If parameter recipes are used in the Mech-Vision project, the recipe to be used is set with **MM_Switch_Model**.
5. Run the Mech-Vision project with **MM_Start_Vis**.
6. Wait for 1 second. Under Eye-In-Hand, this **WaitTime** instruction is required to make sure the robot stays still until image acquisition is completed. Under Eye-To-Hand, this **WaitTime** instruction can be replaced with **MoveL** or **MoveJ**.

7. Obtain the vision results from Mech-Vision.
8. Check if the returned status code indicates any error. If an error code is returned, the program is stopped.
9. Move the robot to the picking pose and perform picking.
10. Move the robot to a waypoint between the picking pose and placing pose.
11. Move the robot to the set placing pose and perform placing.
12. Return the robot to HOME position.

The following parts should be modified according to your actual needs.

Define the TCP

The TOOL is defined as **Gripp1TCP**. Change the value of **Gripp1TCP** to the actual TCP values.

Define HOME position

Set the Home position in **Home**.

Teach the Image Capturing Pose

Record the image capturing pose in **camera_capture**.

Teach the Waypoint(s)

Waypoints are intermediate poses between the picking pose and placing pose. They are used to ensure that the robot doesn't collide with the surrounding when moving between the picking and placing poses.

You can add one or more waypoints to **waypoint**.

Teach the Placing Pose

Record the placing pose in **drop**.

Define Z-Offset from Picking/Placing Pose

Z-offset distances relative to the tool frame from the picking/placing pose are used to ensure collision doesn't occur when the robot is approching or departing the picking/placing pose.

Adjust the following commands according to your actual needs.

- **MoveJ RelTool(pickpoint,0,0,-100)**: the Z-offset when approching the picking pose is set to **100**. Robot will move to 100 mm above the picking pose.
- **MoveL Offs(pickpoint,0,0,100)**: the Z-offset when departing the picking pose is set to **100**. Robot will move to 100 mm above the picking pose.

- **MoveL RelTool(drop,0,0,-100)**: the Z-offset when approching placing pose is set to **100**. Robot will move to 100 mm above the placing pose.
- **MoveL Offs(drop,0,0,100)**: the Z-offset when departing the placing pose is set to **100**. Robot will move to 100 mm above the placing pose.

Add Object Grasping and Releasing Logics

Add logic for controlling the tool action when picking or placing the object.

Obtain Planned Path from Mech-Viz

```
PROC vision_sample_2()
  !***********************************************************
  ! FUNCTION:Eye to Hand simple pick and place with Mech-Viz
  ! mechmind, 2022-5-1
  !***********************************************************
  AccSet 100, 100;
  VelSet 100, 5000;
  MoveAbsJ Home\NoEOffs,v3000,fine,Gripp1TCP;!move robot home position
  MoveL camera_capture,v500,fine,Gripp1TCP;
  PoseNum:=0;
  !Set ip address of IPC
  MM_Init_Socket "127.0.0.1",50000,60;
  WaitTime 0.1;
  !Set vision recipe
  MM_Switch_Model 1,1;
  !Run Viz project
  MM_Start_Viz 1;
  WaitTime 0.1;
  !set branch exitport
  MM_Set_Branch 1,1;
  !get planned path
  MM_Get_VizData 2, LastData, PoseNum, VisPosNum, MM_Status;
  IF MM_Status<>2100 THEN
      Stop;
  ENDIF
  FOR i FROM 1 TO PoseNum DO
      count:=i;
      MM_Get_Pose count,P{count},label1{count},speed{count};
  ENDFOR
  !follow the planned path to pick
  FOR j FROM 1 TO PoseNum DO
      count:=j;
      MoveL p{count},v1000,fine,Gripp1TCP;
      IF count=VisPosNum THEN
          Stop;
          !add object grasping logic here
      ENDIF
  ENDFOR
  !go to drop location
  MoveL RelTool(drop,0,0,-100),v1000,z50,Gripp1TCP;
  MoveL drop,v500,fine,Gripp1TCP;!drop point
```

(continues on next page)

(continued from previous page)

```
  stop;
  !add object releasing logic here
  MoveL Offs(drop,0,0,100), v1000, fine, Gripp1TCP;
  MoveAbsJ Home\NoEOffs,v3000,fine,Gripp1TCP;
  RETURN ;
ENDPROC
```

Program Logic

1. Move the robot to HOME position.
2. Move the robot to the image capturing pose.
3. Initialize communication with **MM_Init_Socket**.
4. If parameter recipes are used in the Mech-Vision project, the recipe to be used is set with **MM_Switch_Model**.
5. Run the Mech-Viz project with **MM_Start_Viz**.
6. Obtain the planned path from Mech-Viz.
7. Check if the returned status code indicates any error. If an error code is returned, the program is stopped.
8. Store obtained target points in the planned path to **P{}** with a **FOR** loop.
9. Move the robot along the planned path with a **FOR** loop and perform picking.
10. Move the robot to a waypoint between the picking pose and placing pose.
11. Move the robot to the set placing pose and perform placing.
12. Return the robot to HOME position.

The following parts should be modified according to your actual needs.

Define the TCP

The TOOL is defined as **Gripp1TCP**. Change the value of **Gripp1TCP** to the actual TCP values.

Define HOME position

Set the Home position in **Home**.

Teach the Image Capturing Pose

Record the image capturing pose in **camera_capture**.

Teach the Waypoint(s)

Waypoints are intermediate poses between the picking pose and placing pose. They are used to ensure that the robot doesn't collide with the surrounding when moving between the picking and placing poses.

You can add one or more waypoints to **waypoint**.

Teach the Placing Pose

Record the placing pose in **drop**.

Add Object Grasping and Releasing Logics

Add logic for controlling the tool action when picking or placing the object.

Define Z-Offset from Picking/Placing Pose

Z-offset distances relative to the tool frame from the picking/placing pose are used to ensure collision doesn't occur when the robot is approching or departing the picking/placing pose.

Adjust the following commands according to your actual needs.

- **MoveL RelTool(drop,0,0,-100)**: the Z-offset when approching placing pose is set to **100**. Robot will move to 100 mm above the placing pose.
- **MoveL Offs(drop,0,0,100)**: the Z-offset when departing the placing pose is set to **100**. Robot will move to 100 mm above the placing pose.

2.1.4 ABB Standard Interface Commands

The ABB Standard Interface provides the following procedures:

- *Initialize Communication*
- *Start Mech-Vision Project*
- *Get Vision Result*
- *Start Mech-Viz Project*
- *Get Planned Path*
- *Obtain Pose*
- *Obtain Joint Positions*
- *Switch Mech-Vision Recipe*
- *Select Mech-Viz Branch*

- *Set Move Index*
- *Get Software Status*
- *Input Object Dimensions to Mech-Vision*
- *Get DO Signal List*
- *Set DO Signal List*
- *Input TCP to Mech-Viz*
- *Calibration*

When programming the ABB robot, please pay attention to the following:

- Multiple parameters should be separated by commas.
- All parameters should be defined as local variables in the program file.
- Parameters can be defined as Input or Output parameters.
- Arguments in the program: the input arguments can be constants, global variables or local variables, and the output arguments can be global variables or local variables.

This Standard Interface is over the TCP/IP protocol.

Initialize Communication

```
MM_Init_Socket "IP_Address",Server_Port,Time_Out;
```

This procedure is used to set the host IP address, port number, and wait time for TCP/IP communication.

Parameters

- Input Parameters

Name	Description
IP_Address	the IP address of the host
Server_Port	TCP/IP port number; the default port number is 50000
Time_Out	Wait time in seconds before stopping connection attempt

Example

```
MM_Init_Socket "192.168.1.20",50000,60;
```

When running the example, the host IP address should be set to 192.168.1.20, the port number should be set to 50000, and the wait time is 60 seconds.

Start Mech-Vision Project

```
MM_Start_Vis Job,Pos_Num_Need,SendPos_Type;
```

This procedure is applications that use Mech-Vision but not Mech-Viz. It runs the corresponding Mech-Vision project to acquire and process data.

Parameters

- Input Parameters

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Please go to *Deployment Settings* → *Mech-Vision* to check and adjust the number.
Pos_Num_Need	Number of poses for Mech-Vision to send, from 0 to 20, where 0 means "send all".
SendPos_Type	Set the image capturing pose for the robot to send, from 0 to 2
	0: Do not send image capturing pose (for Eye To Hand) 1: Send image capturing pose as joint positions 2: Send image capturing pose as robot flange pose

Example

```
MM_Start_Vis 1,1,1;
```

This example triggers Mech-Vision No. 1 project to run; the Mech-Vision No. 1 project is expected to send back 1 pose; and the robot will send image capturing pose as joint to Mech-Center.

Get Vision Result

```
MM_Get_VisData Job,Last_Data,Pos_Num,MM_Status;
```

This procedure is for applications that use Mech-Vision but not Mech-Viz. It obtains the vision result from the corresponding Mech-Vision project.

Parameters

- Input Parameter

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Please go to *Deployment Settings* → *Mech-Vision* to check and adjust the number.

- Output Parameters

Name	Description
Last_Data	Variable, indicating whether all vision result has been sent, 0 or 1
	0: NOT all vision result has been sent (more on the way) 1: All vision result has been sent
Pos_Num	Variable for storing the number of received poses
MM_Status	Variable for storing status code, refer to the standard_interface_status_codes

Example

```
MM_Get_VisData 1,LastData,PoseNum,MMStatus;
```

This example obtains the vision result from Mech-Vision project No.1. Whether all vision result has been sent is stored in **LastData**, the number of poses received is stored in **PoseNum**, and the status code is stored in **MMStatus**.

Start Mech-Viz Project

```
MM_Start_Viz SendPos_Type;
```

This procedure is for applications that use both Mech-Vision and Mech-Viz. It runs the corresponding Mech-Viz project (which triggers the corresponding Mech-Vision project to run), and sets the initial joint positions of the simulated robot in Mech-Viz.

Parameter

- Input Parameter

Name	Description
SendPos_Type	The initial joint positions for the simulated robot in Mech-Viz, 0 or 1
	0: Set the initial joint positions of the simulated robot to [0,0,0,0,0,0] 1: Set the initial joint positions of the simulated robot to the current joint positions of the real robot

Note: When the scene contains object models that obstruct the robot to move from [0,0,0,0,0,0] to the first target point, this parameter must be set to 1.

Example

```
MM_Start_Viz 1;
```

This example runs the corresponding Mech-Viz project, and sets the initial joint positions of the simulated robot to the current joint positions of the real robot.

Get Planned Path

```
MM_Get_VizData GetPos_Type,Last_Data,Pos_Num,VisPos_Num,MM_Status;
```

This procedure obtains the planned path from Mech-Viz.

Parameters

- Input Parameter

Name	Description
GetPos_Type	Whether Mech-Viz should send target points as joint positions or TCPs, 1 or 2
	1: Mech-Viz sends joint positions 2: Mech-Viz sends TCPs

- Output Parameters

Name	Description
Last_Data	Variable, indicating whether all target points have been sent, 0 or 1
	0: NOT all target points have been sent (more on the way) 1: All target points have been sent
Pos_Num	Variable for storing the number of received target points
VisPos_Num	Variable for storing the position of the first visual_move target point in the path
	Example path: move-1, move-2, visual_move-3, move-3, visual_move-2 In this path, the position of the first visual_move target point is 3. If the path does not contain visual_move target point, the return value is 0.
MM_Status	Variable for storing status code, refer to the standard_interface_status_codes

Example

```
MM_Get_VizData 2,Last,Pose_Num,Vis_Index,StatusCode;
```

This example obtains the planned path from Mech-Viz in the form of TCPs. Whether all target points have been sent is stored in **Last**, the number of target points received is stored in **Pose_Num**, the position of the visual_move target point is stored in **Vis_Index**, and the status code is stored in **StatusCode**.

Obtain Pose

```
MM_Get_Pose Index,P90,Label,Pose_Speed;
```

This procedure stores a pose returned by Mech-Vision or a target point (as TCP) returned by Mech-Viz in the specified variable.

Parameters

- Input Parameter

Name	Description
Index	Specify the index of the pose to be stored

- Output Parameters

Name	Description
P90	Variable for storing the specified pose
Label	Variable for storing the label corresponding to the specified pose
Pose_Speed	Variable for storing the speed corresponding to the specified pose

Example

```
MM_Get_Pose 1,P90,Label,PoseSpeed1;
```

This example stores the first received pose to **P90**, the corresponding label to **Label**, and the corresponding speed to **PoseSpeed1**.

Obtain Joint Positions

```
MM_Get_Jps Index,Jointtarget,Label,Pose_Speed;
```

This procedure stores a set of joint positions returned by Mech-Viz in the specified variable.

Note: As Mech-Vision does not output joint position data, this procedure can only be used with Mech-Viz.

Parameters

- Input Parameter

Name	Description
Index	Specify the index of the set of joint positions to be stored

- Output Parameters

Name	Description
Jointtarget	Variable for storing the specified set of joint positions
Label	Variable for storing the label corresponding to the specified set of joint positions
Pose_Speed	Variable for storing the speed corresponding to the specified set of joint positions

Example

```
MM_Get_Jps 1,jpose1,Label1,PoseSpeed1;
```

This example stores the first set of received joint positions to **jpose1**, the corresponding label to **Label1**, and the corresponding speed to **PoseSpeed1**.

Switch Mech-Vision Recipe

```
MM_Switch_Model Job,Model_Number;
```

This procedure specifies which parameter recipe of the Mech-Vision project to use. For more information on parameter recipe, please see parameter_recipe_configuration.

Note:

- This procedure must be called BEFORE **MM_Start_Vis**.

Parameters

- Input Parameters:

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center* → *Deployment Settings* → *Mech-Vision*
Model_Number	The number of a parameter recipe in the Mech-Vision project, from 1 to 99

Example

```
MM_Switch_Model 1,1;
```

This example switches the parameter recipe used to No. 1 in Mech-Vision project No. 1.

Select Mech-Viz Branch

```
MM_Set_Branch Branch_Num,Exit_Num;
```

This procedure is used to select along which branch the Mech-Viz project should proceed. Such branching is achieved by adding branch_by_service_message Task(s) to the project. This procedure specifies which out port such Task(s) should take.

Note:

- **MM_Start_Viz** must be called BEFORE this procedure.
- When the next Task to be executed in Mech-Viz is a **branch_by_service_message** Task, Mech-Viz will wait for this procedure to send the out port number it should take.

- The name of all **branch_by_service_message** Tasks in the Mech-Viz project must be changed to numbers between 1 and 99, and the names should be unique among all tasks in the project.

Parameters

- Input Parameters

Name	Description
Branch_Num	Name of the **branch_by_service_message** Task, from 1 to 99
Exit_Num	The number of the out port to take, from 1 to 99

Example

```
MM_Set_Branch 1,3;
```

This example tells Mech-Viz to take out port **3** for the **branch_by_service_message** Task named **1**.

Set Move Index

```
MM_Set_Index Skill_Num,Index_Num;
```

This procedure sets the value for the Current Index parameter of Mech-Viz Tasks. Tasks that have this parameter include move_list, move_grid, custom_pallet_pattern, and smart_pallet_pattern.

Note:

- **MM_Start_Viz** must be called BEFORE this procedure.
- The name of all Tasks with index parameters in the Mech-Viz project must be changed to numbers between 1 and 99, and the names should be unique among all tasks in the project.

Parameters

- Input Parameters

Name	Description
Skill_Num	Name of the Task, from 1 to 99
Index_Num	Value for the Current Index parameter when the Task is executed

Example

```
MM_Set_Index (2,10);
```

This example sets the Current Index value to 9 for the Task named **2**. When the Task is executed, the Current Index value will be added 1 and become 10.

Get Software Status

```
MM_Get_Status MM_Status;
```

This procedure is currently capable of checking whether Mech-Vision is ready to run projects. In the future, this procedure can be used for obtaining the execution status of Mech-Vision, Mech-Viz and Mech-Center.

Parameter

- Output Parameter

Name	Description
MM_Status	Variable for storing the status code, refer to the standard_interface_status_codes

Example

```
MM_Get_Status StatusCode;
```

This example obtains the status code and stores it in **StatusCode**.

Input Object Dimensions to Mech-Vision

```
MM_Set_BoxSize Job,Lenght,Width,Height;
```

This procedure inputs object dimensions to the Mech-Vision project.

Note:

- This procedure must be called BEFORE **MM_Start_Vis**.

Parameters

- Input Parameters

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center → Deployment Settings → Mech-Vision*
Lenght	Length of object in mm
Width	Width of object in mm
Height	Height of object in mm

Example

```
MM_Set_BoxSize(1,500,300,200)
```

This example sets the object dimensions in the read_object_dimensions Step in the Mech-Vision project No. 1 to 500*300*200 mm.

Get DO Signal List

```
MM_Get_DoList Status;
```

This procedure obtains the planned DO Signal list for controlling multiple sections of a sectioned vacuum gripper.

Note:

- **MM_Get_VizData** must be called BEFORE this procedure.
- Please deploy the Mech-Viz project based on the template project in *Mech-Center/tool/viz_project/suction_zone*, and set the suction cup configuration file in the Mech-Viz project.
- Set the current robot model as the **Received Name** in Mech-Viz.

Parameter

- Output Parameter

Name	Description
Status	Variable for storing the status code, refer to the standard_interface_status_codes

Example

```
MM_Get_DoList Status;
```

This procedure stores the status code of Mech-Viz in **Status**.

Set DO Signal List

```
MM_Set_DoList Serial,GO16;
```

This procedure sets the DoList sent by Mech-Viz to go signals. It supports 4 groups of 16-bit go signals in maximum. If you need to set multiple groups of go signal, please run this procedure for several times.

Note: **MM_Get_DoList** must be called BEFORE this procedure.

Parameters

- Input Parameters

Name	Description
Serial	Variable for storing the status code
GO16	Group output for setting DO signals

Example

```
| MM_Set_DoList 1,GO16;

| MM_Set_DoList 2,GO32;
```

The first example sets the 0-15 values in the DoList calculated by Mech-Viz to corresponding **GO16** signals, and the second example sets the 16-32 values in DoList to corresponding **GO32** signals.

Input TCP to Mech-Viz

```
MM_Set_Pos Pos;
```

This procedure inputs TCP data to the outer_move Task.

Note:

- This procedure must be called BEFORE **MM_Start_Viz**.
- Please deploy the Mech-Viz project based on the template project in *Mech-Centertoolviz_projectouter_move*, and put the **outer_move** Task at a proper position in the workflow.

Parameter

- Input Parameter

Name	Description
Pos	Variable for storing the TCP data to be sent to Mech-Viz

Example

```
MM_Set_Pos P10;
```

This example sends the TCP data stored in **P10** to the **outer_move** task in the Mech-Viz project.

Calibration

```
MM_Calib Move_Type,Pos_Jps,Wait_time,\\num Ext;
```

This procedure is used for hand-eye calibration (camera extrinsic parameter calibration). It automates the calibration process in conjunction with the **Camera Calibration** function in Mech-Vision. For detailed instructions, see *ABB Calibration Program*.

Parameters

- Input Parameters

Name	Description
Move_Type	Motion type, 1 or 2
	1: MoveL 2: MoveJ
Pos_Jps	Pose as flange pose or joint positions, 1 or 2
	1: flange pose 2: Joint positions
Wait_Time	Wait time between poses in seconds
Ext	Data of the external 7th axis in mm (Optional)

Examples

- Example 1

```
MM_Calib 2,1,300;
```

This example moves the robot in MoveJ type, receives pose data in the form of flange pose, and sets the wait time between two poses to 300 seconds. In addtion, this robot does not have an external axis installed.

- Example 1

```
MM_Calib 2,1,300,\\EXT:=Axis7;
```

This example moves the robot in MoveJ type, receives pose data in the form of flange pose, and sets the wait time between two poses to 300 seconds. Moreover, the 7th axis value of this robot is Axis7.

2.1.5 ABB Error Messages

The following errors may occur while running the Standard Interface program on the robot.

"MM:Robot Error" ," Socket Timeout,Check connetion"

The robot program didn' t receive data within the specified wait time when calling the instruction **SocketRecive**.

Troubleshooting

- Check if the sequence of calling **SocketSend** and **SocketReceive** is correct.
- Check if the Standard Interface is started in Mech-Center.
- Check if the arguments of **SocketReceive** are set correctly.

"MM:Robot Error" ," Socket Closed,Check connection"

The robot couldn' t establish communication with Mech-Center.

Troubleshooting

- Check if the hardware are properly connected.
- Check if the Standard Interface is started in Mech-Center.
- Check the IP addresses of the robot and the IPC, and if the port number is configured correctly.
- Check if the firewall is turned off on the IPC.
- Contact Mech-Mind Technical Support for further assistance.

"MM:Robot Error" ," Wrong Arguments"

When calling a Mech-Mind Standard Interface procedure, arguments provided are not correct.

Troubleshooting

Please refer to *ABB Standard Interface Commands* and input the correct arguments accordingly.

"MM:Robot Error", "CMD No. Error"

The command code returned to the robot does not match the one sent to Mech-Center.

Troubleshooting

The sequence of command sending and receiving is problematic. Please contact Mech-Mind Technical Support for further assistance.

"MM:IPC Error", "Check IPC status"

Returned status code is an error code. Please check Mech-Center' s log.

Troubleshooting

- Please refer to the standard_interface_status_codes for the specific error.
- Please contact Mech-Mind Technical Support for further assistance.

2.2 YASKAWA

This section introduces the Standard Interface for YASKAWA robots.

2.2.1 YASKAWA Setup Instructions

This section introduces the process of loading the Standard Interface program onto a YASKAWA robot.

The process consists of 4 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program Files*
- *Test Robot Connection*

Please have a flash drive ready at hand.

Note: A USB 2.0 flash drive is recommended. Otherwise, the robot controller may not recognize the flash drive.

Check Controller and Software Compatibility

- Robot: 6-axis YASKAWA robot
- Controller: YRC1000 (excluding YRC1000 micro) and DX200
- Option function requirements: must have the MotoPlus and Ethernet functions enabled.

Note: The following instructions are based on YRC1000 controller. Details may differ for DX200 controller.

Setup the Network Connection

Hardware Connection

Plug the Ethernet cable into:

- An Ethernet port on the IPC
- LAN2 (CN106) port on YRC1000 controller; CN104 port on DX200 controller

---

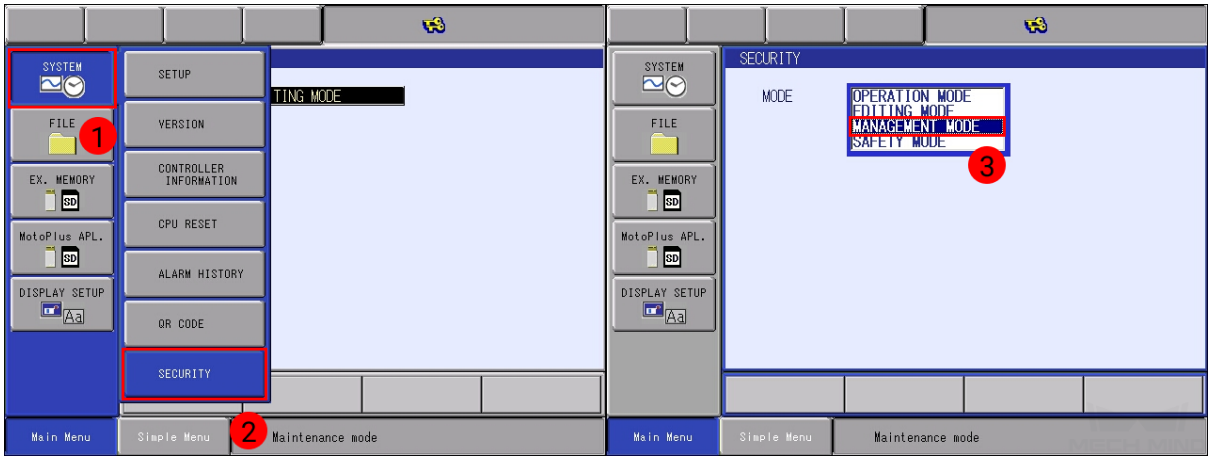
Note:

- LAN1 port on YRC1000 and CN105 port on DX200 are for connecting the teach pendant only.
- If LAN2 port is occupied, please use LAN3 (CN107) instead.

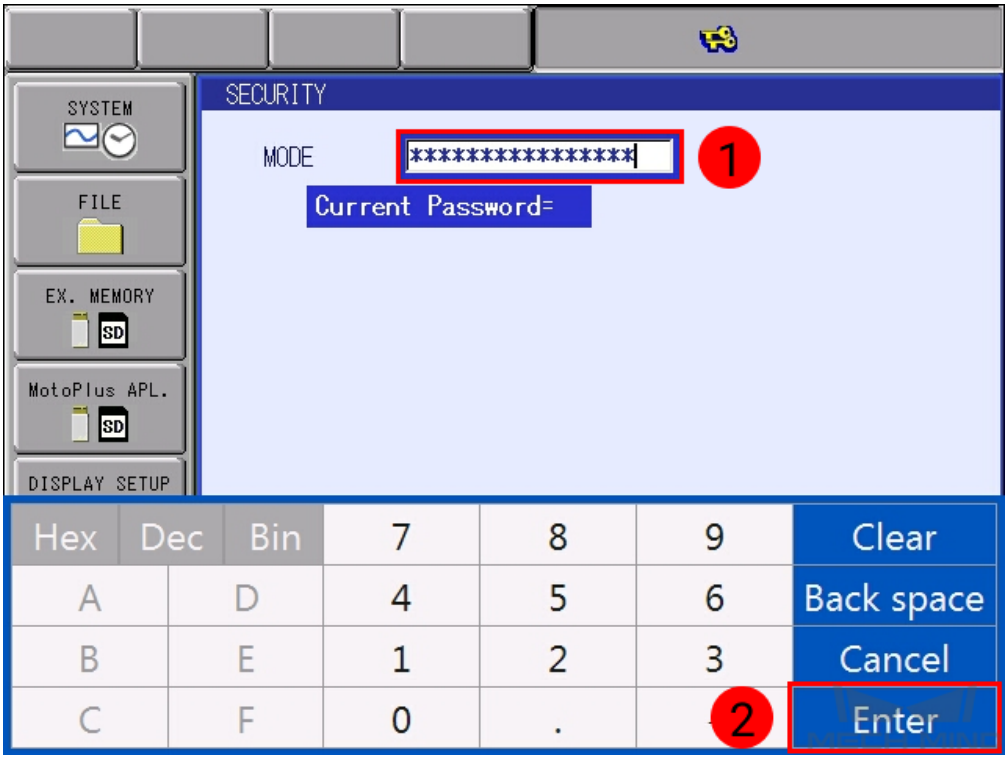
IP Configuration

To allow communication between the IPC and the robot controller, both must have an IP address in the same subnet. This means that the first three numbers of the IP addresses should be the same. For example, 192.168.100.1 and 192.168.100.2 are in the same subnet.

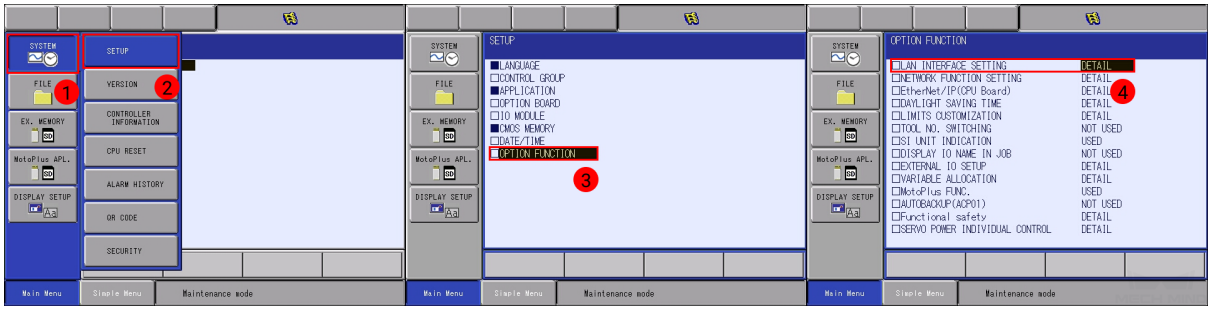
1. Press down `MAIN MENU` when powering on the controller to enter the maintenance mode.
2. Select *SYSTEM* → *SECURITY* → *MANAGEMENT MODE*.



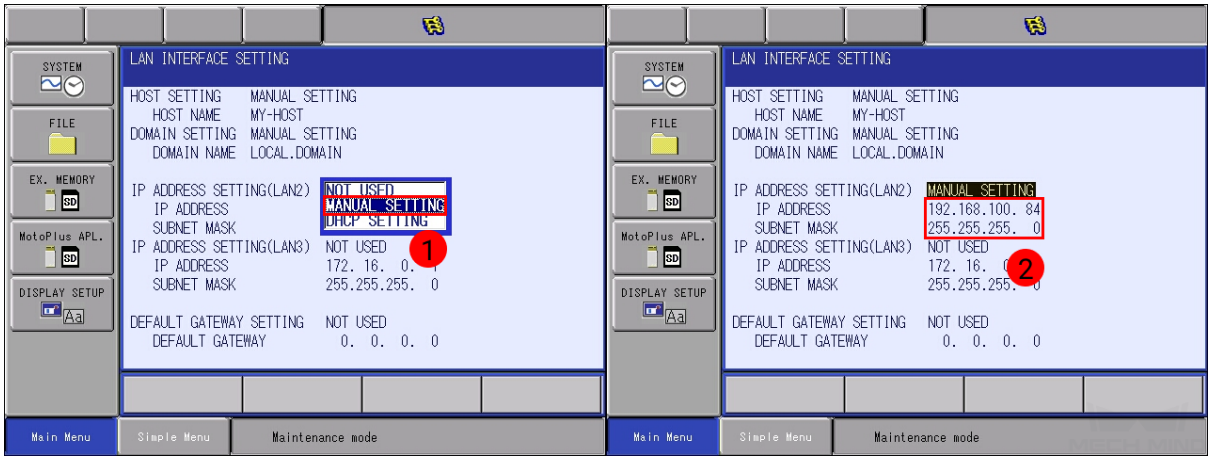
3. Enter the password (the default password is sixteen **9** ‘s), and then press on *Enter*.



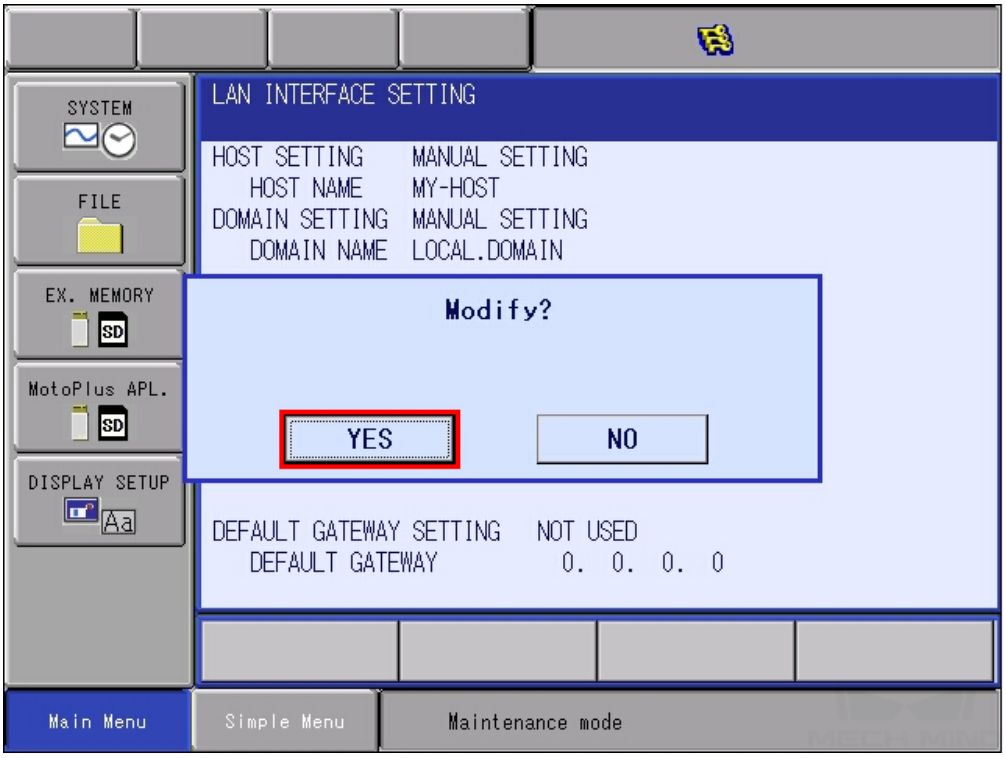
4. Select *SYSTEM* → *SETUP* → *OPTION FUNCTION* → *LAN INTERFACE SETTING*.



5. In **IP ADDRESS SETTING(LAN2)**, select **MANUAL SETTING**, and then set the **IP ADDRESS** to one in the same subnet as the IPC, and the **SUBNET MASK** to **255.255.255.0**.



6. Press the `ENTER` key, and then press on *YES* in the pop-up message.



Load the Program Files

Prepare the Files

The program files are stored in the installation directory of Mech-Center. The default directory is *C:/Mech-Mind/Mech-Center*.

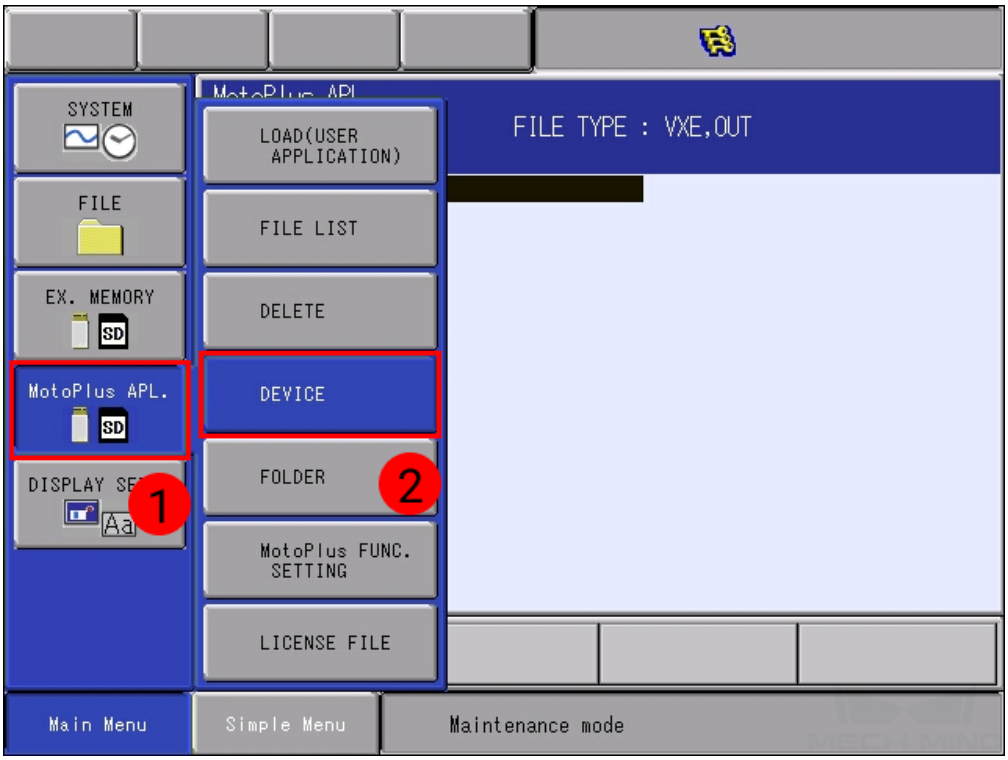
Navigate to *xxx/Mech-Center/mech_interface/yaskawa*, and copy the following files to your flash drive。

- **JBI** folder
- **yrc1000.out** if you are using a YRC1000 controller
- **dx200.out** if you are using a DX200 controller

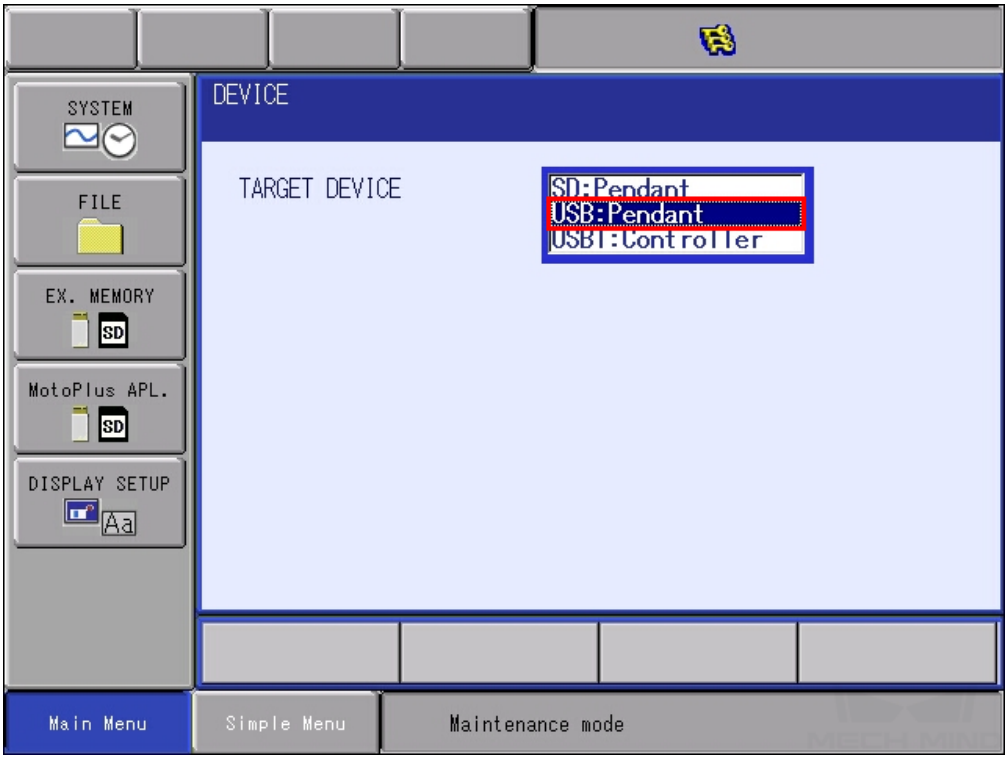
Note: Copy the file to the root directory of the flash drive. Do not put it in another folder or rename it.

Load the MotoPlus Application File to the Robot

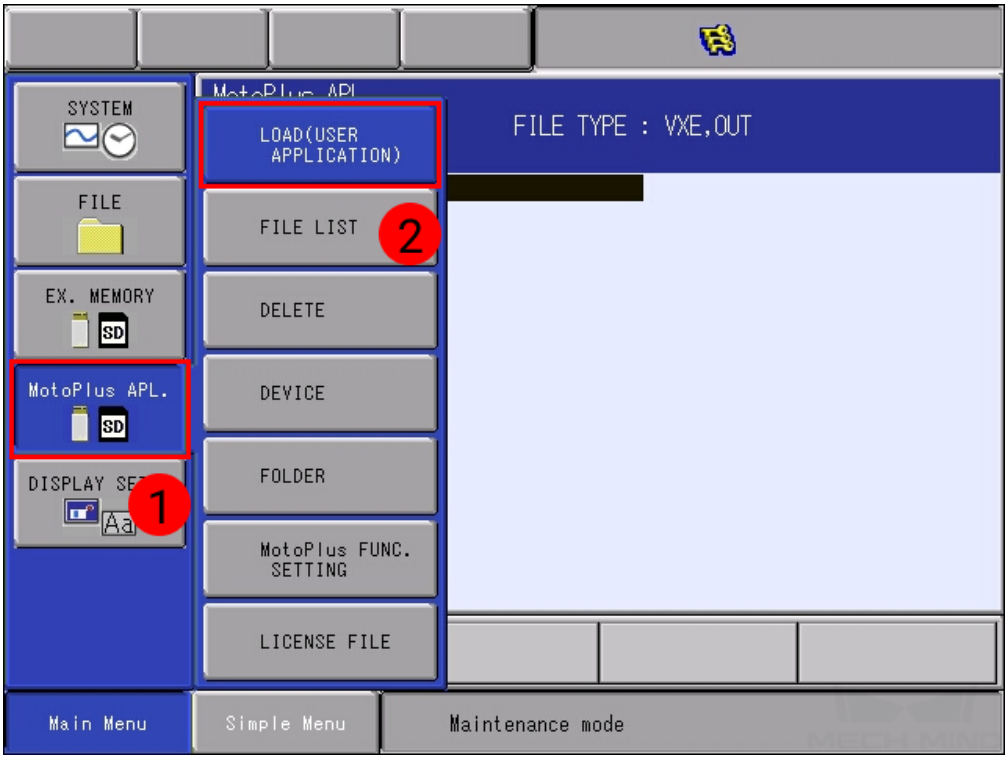
1. Insert the flash drive into the USB port on the back of the teach pendant.
2. Under maintenance mode, select *MotoPlus APL.* → *DEVICE*.



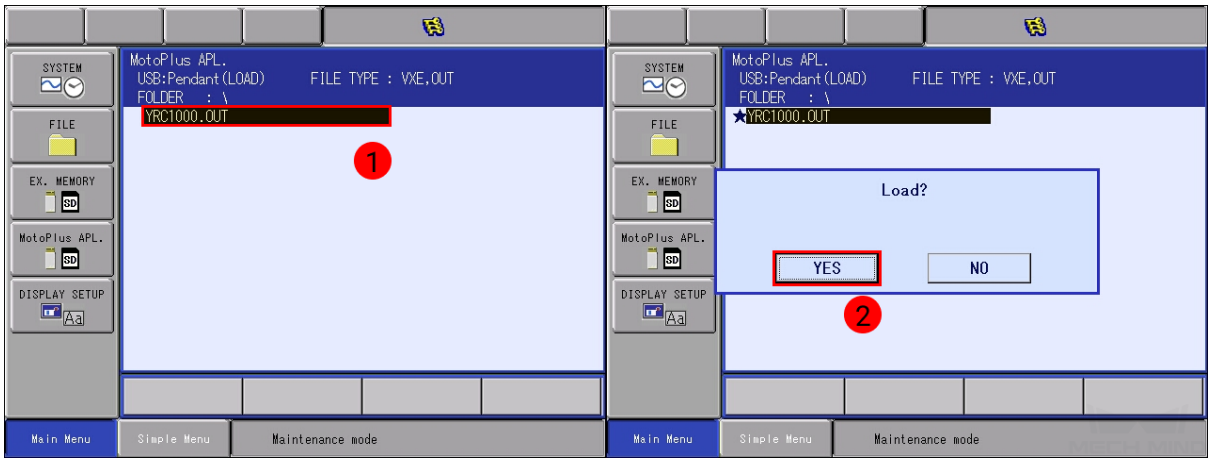
3. Select **USB:Pendant** for **TARGET DEVICE**.



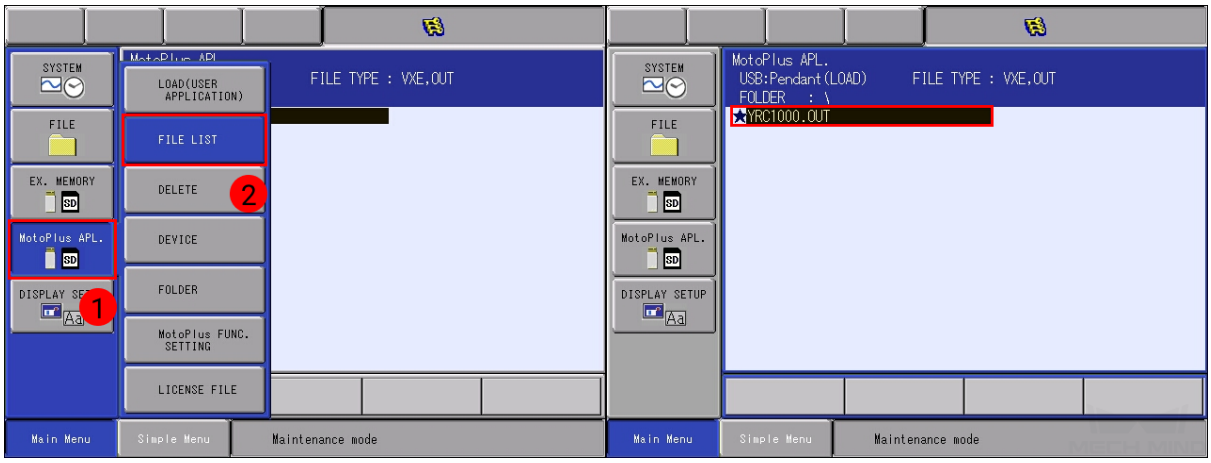
4. Select *MotoPlus APL.* → *LOAD(USER APPLICATION)*.



5. Select **YRC1000.OUT** (**DX200.OUT** for DX200 controller), and press `ENTER`. Select **YES** in the pop-up message to start loading the program.

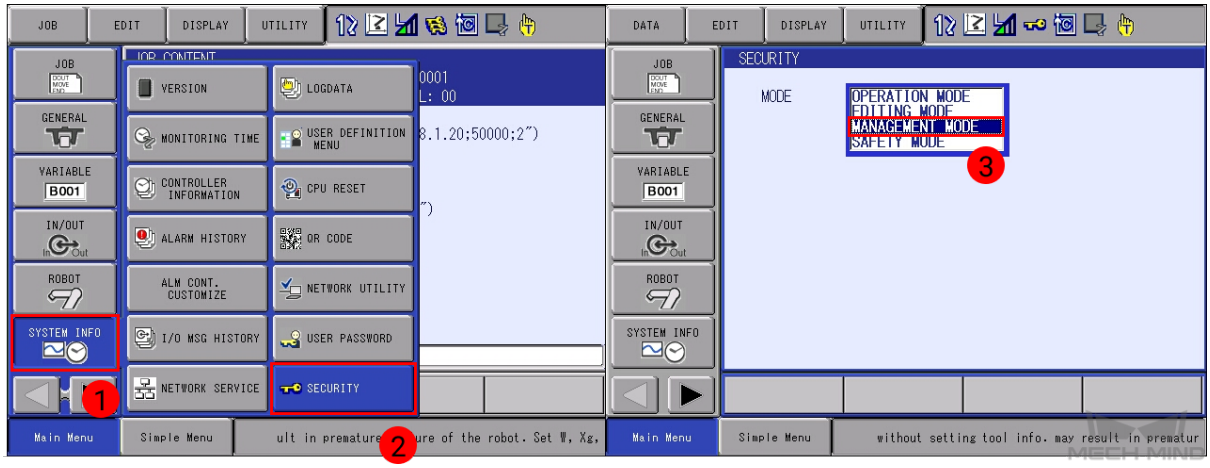


6. After loading completes, go to *MotoPlus APL.* → *FILE LIST*, and you should see **YRC1000.OUT** (**DX200.OUT**) displayed.

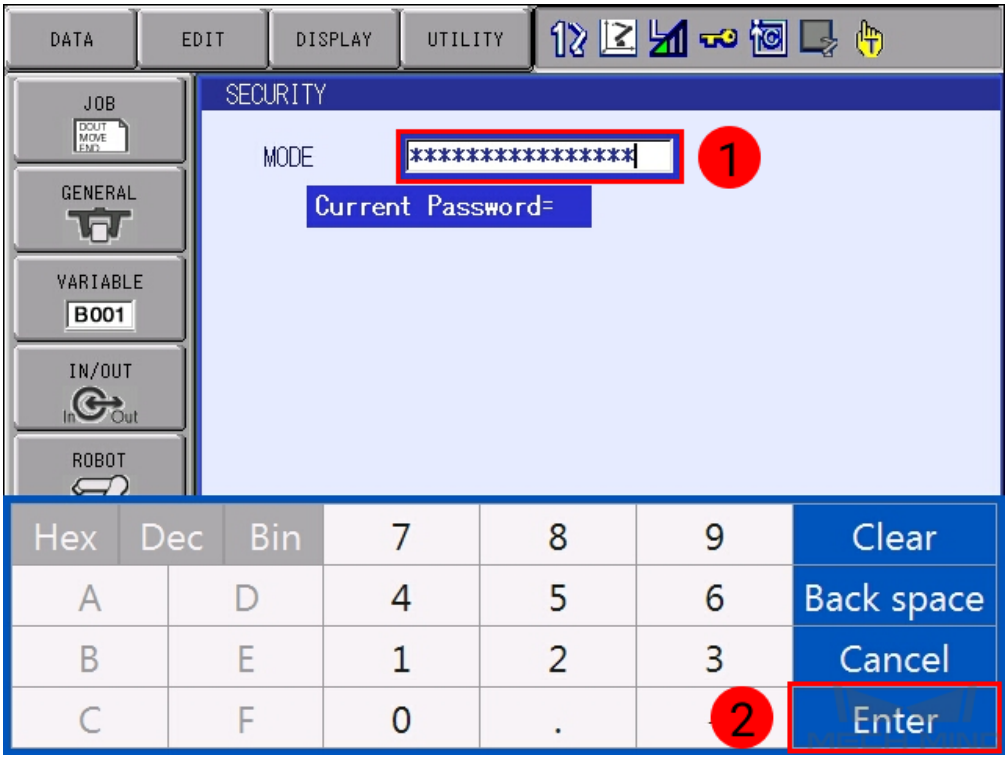


Load the Job Files to the Robot

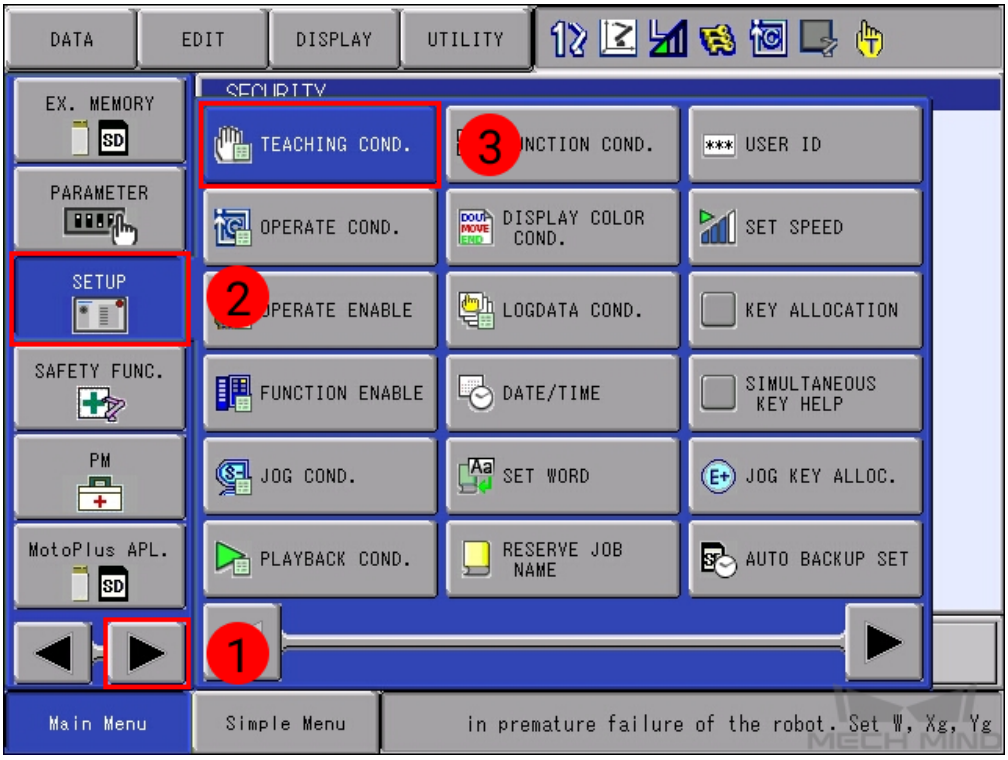
1. Restart the controller without pressing the `MAIN MENU` key, and select *SYSTEM INFO* → *SECURITY* → *MANAGEMENT MODE*.



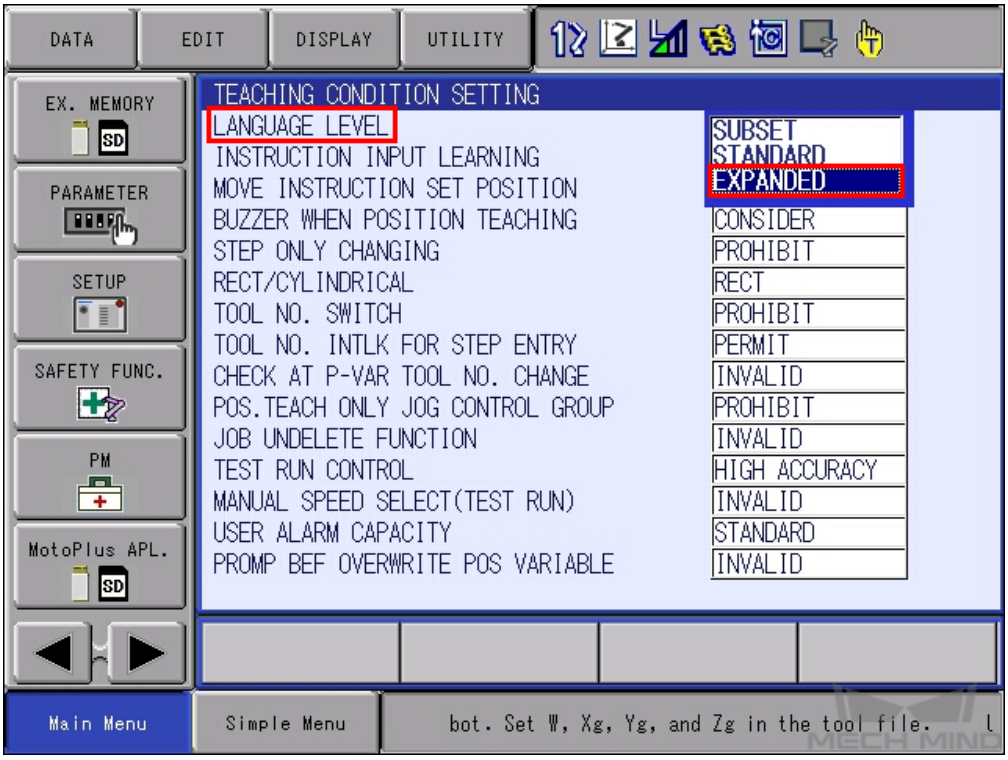
2. Enter the password (the default password is sixteen **9** ‘s), and then press on *Enter*.



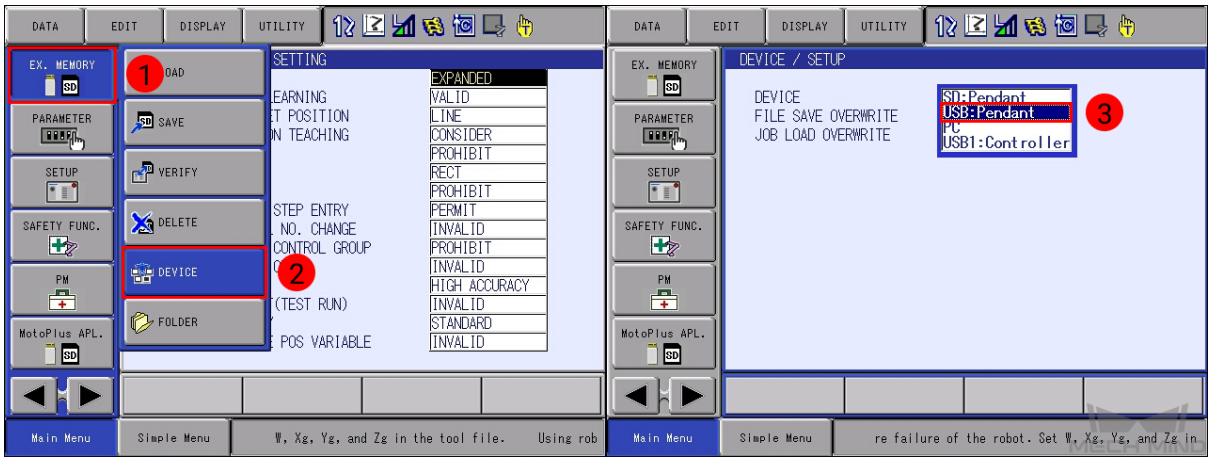
3. Press on the right arrow button, and select *SETUP* → *TEACHING COND.*



4. Set **LANGUAGE LEVEL** to **EXPANDED**.

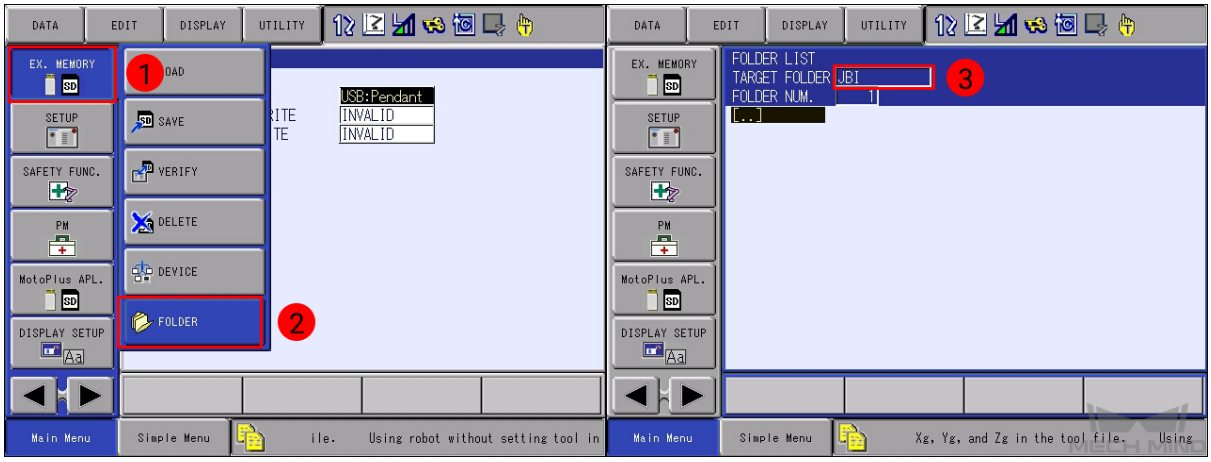


5. Select *EX. MEMORY* → *DEVICE*, and then select **USB:Pendant** for **DEVICE**.

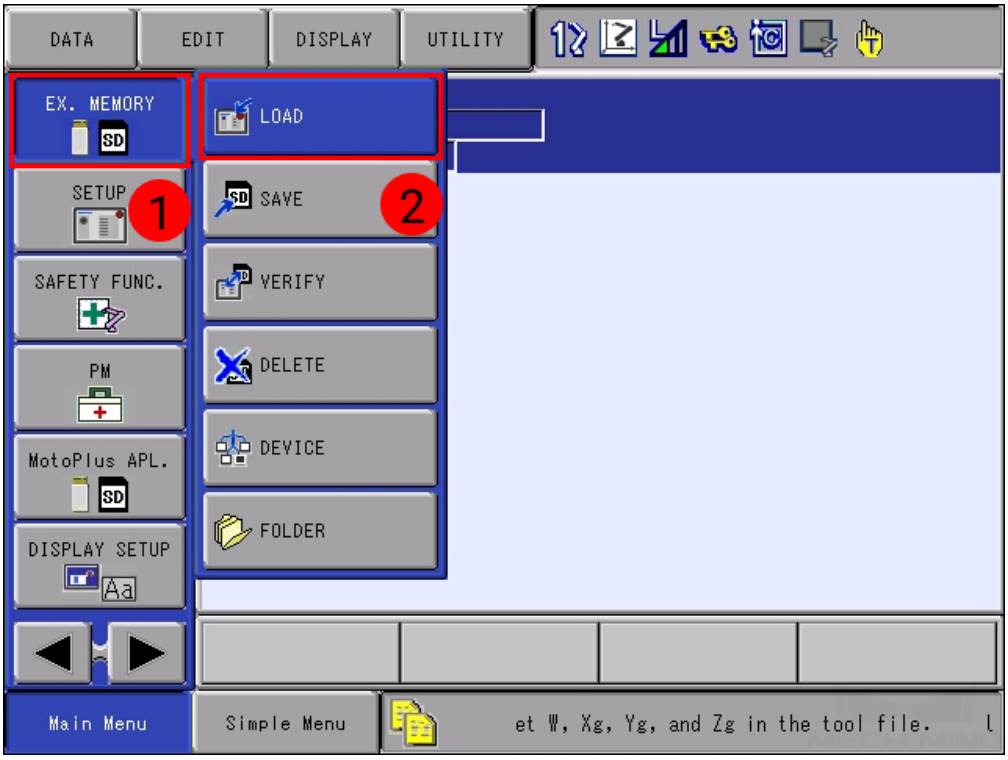


6. Select *EX. MEMORY* → *FOLDER*, and then select **JBI** from the list.

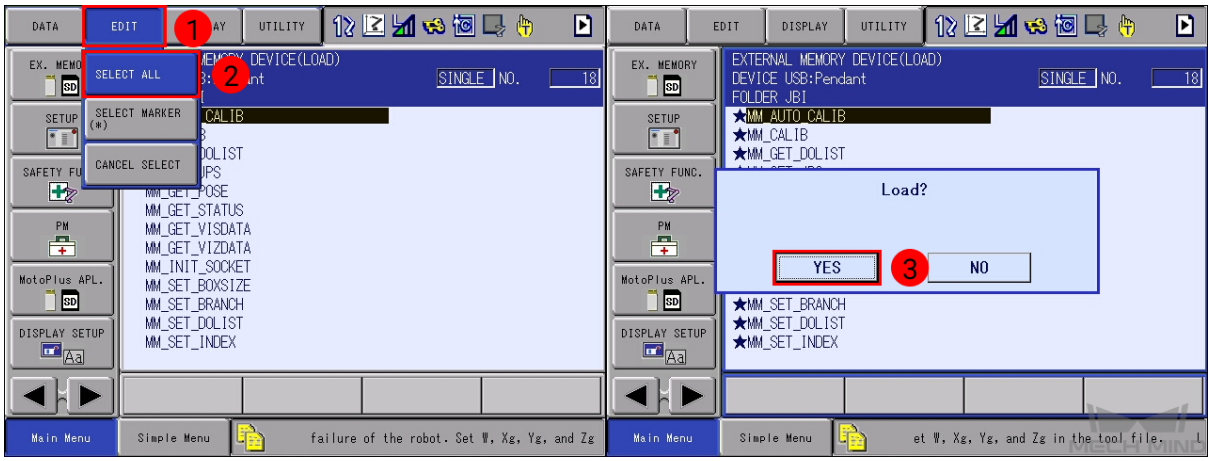
Note: Make sure you are IN the **JBI** folder (**JBI** is displayed after **TARGET FOLDER**).



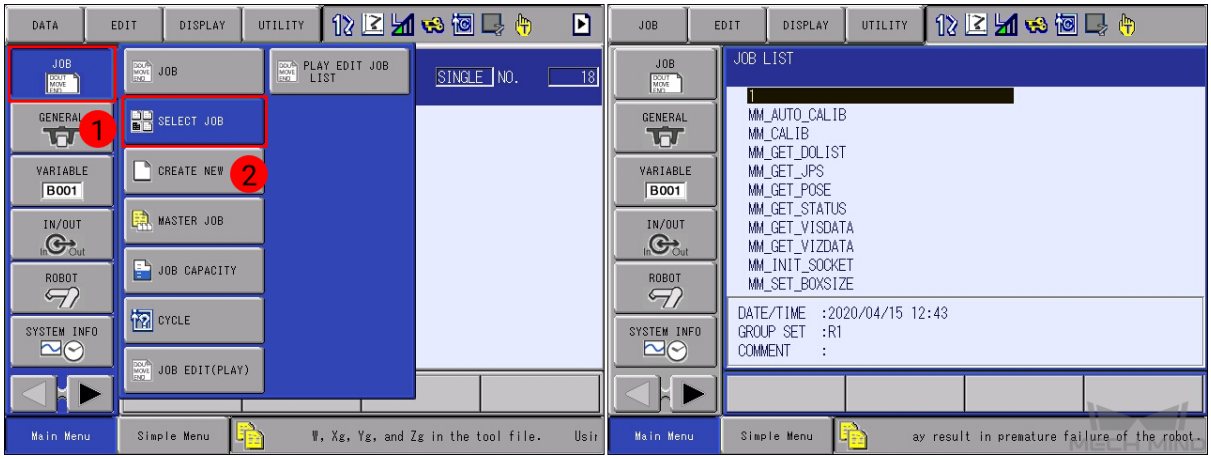
7. Select *EX. MEMORY* → *LOAD*.



8. Select *EDIT* → *SELECT ALL*, and then press ENTER. Select **YES** in the pop-up message to start loading the programs.



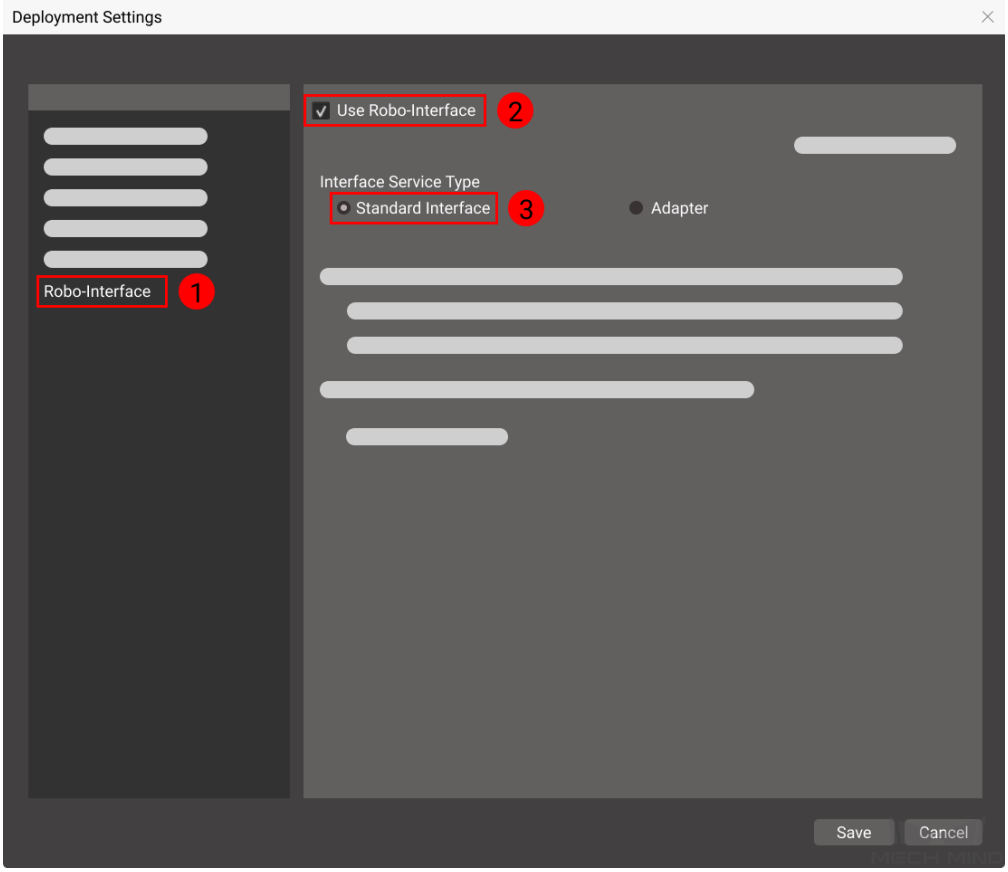
9. After loading completes, go to *JOB* → *SELECT JOB*, and you should see all the job files displayed.



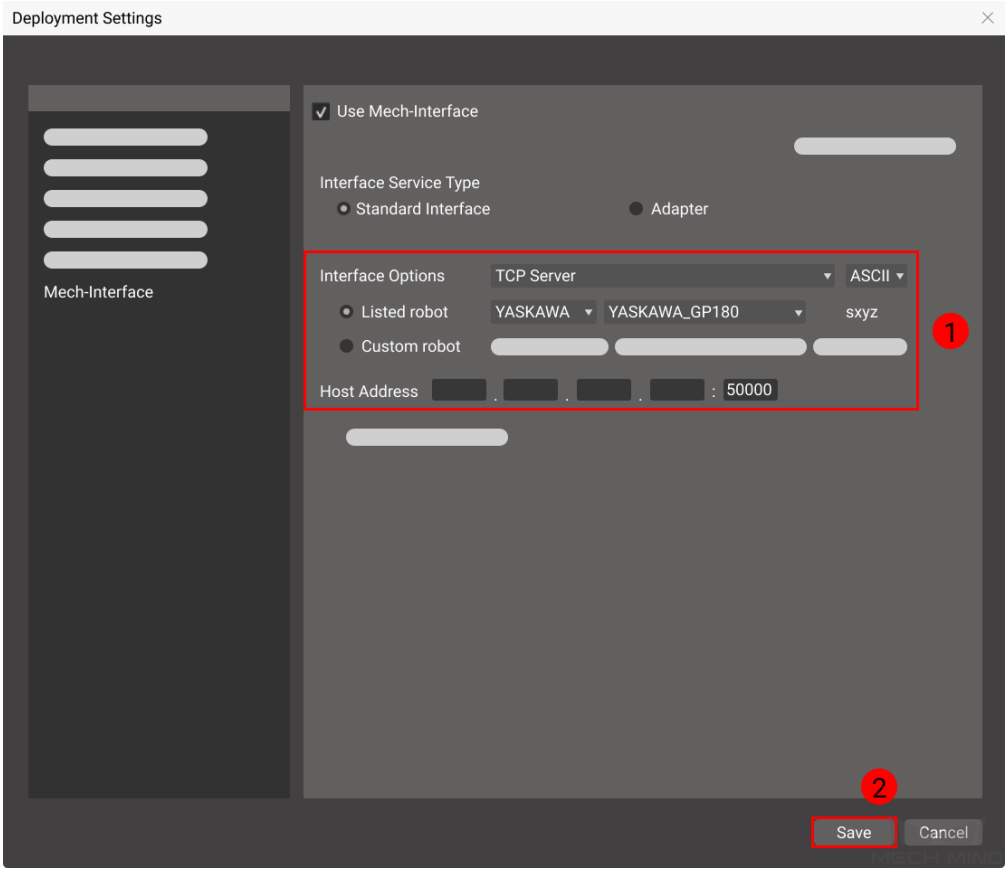
Test Robot Connection

Configure Mech-Interface in Mech-Center

1. Open Mech-Center and click on *Deployment Settings*.
2. Go to **Mech-Interface**, check **Use Mech-Interface** and select **Standard Interface**.



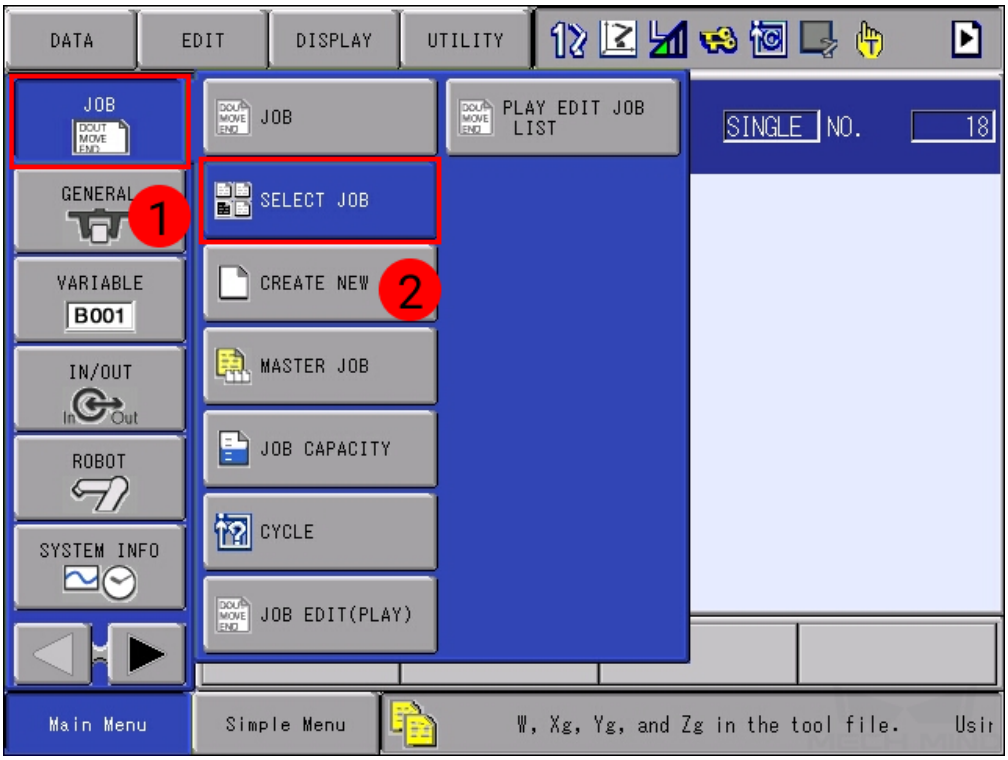
3. Set the following fields:
   - **Interface Option**: Set to **TCP Server** and **ASCII**.
   - **Listed robot**: Select the robot model you are using.
   - **Host Address**: The default port number is **50000**. If you need to change the port number, make sure to change it later on in the robot program as well.
4. Click on *Save*.



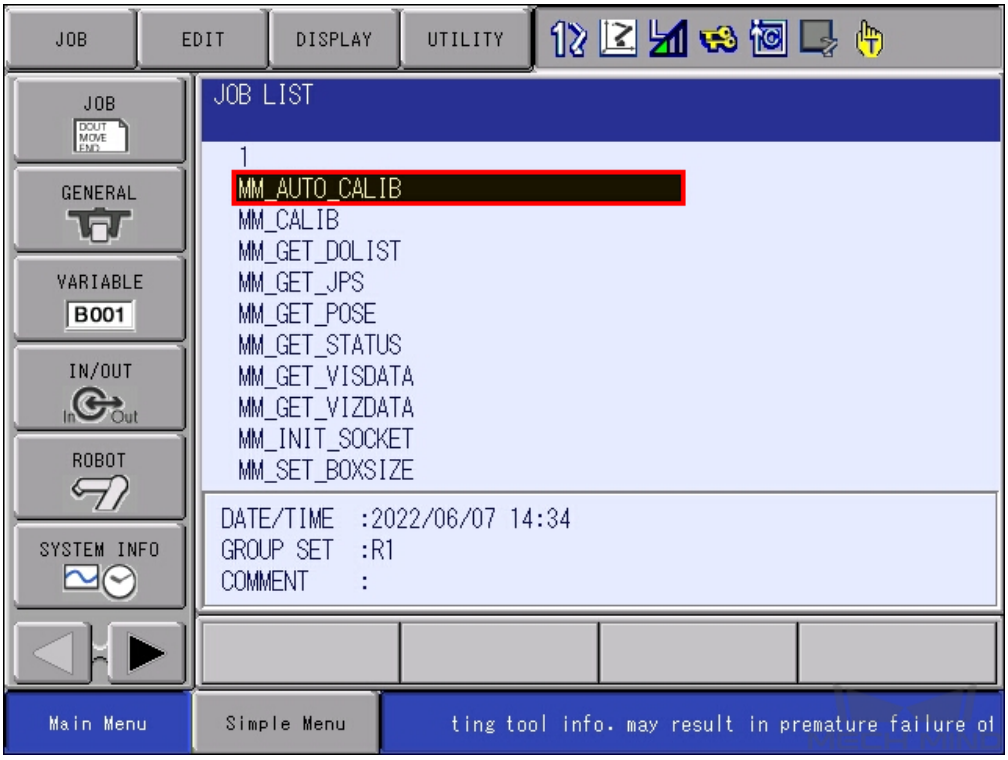
5. Click on *Start Interface* in the Toolbar.

Modify and Run Robot Program

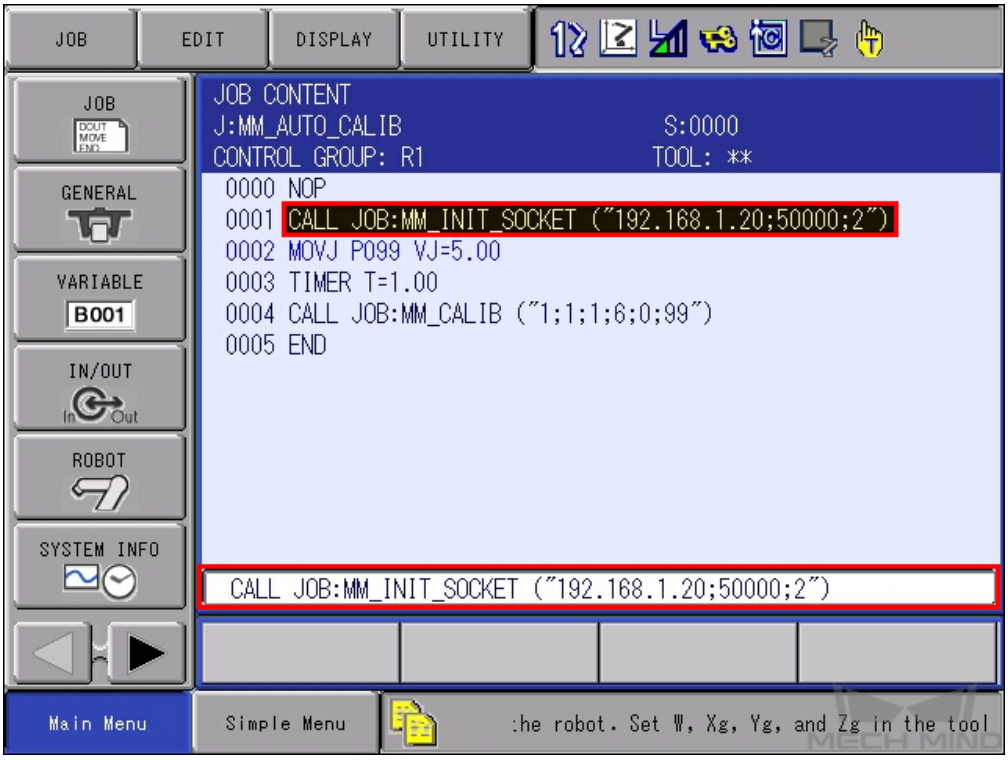
1. Select *JOB* → *SELECT JOB*.



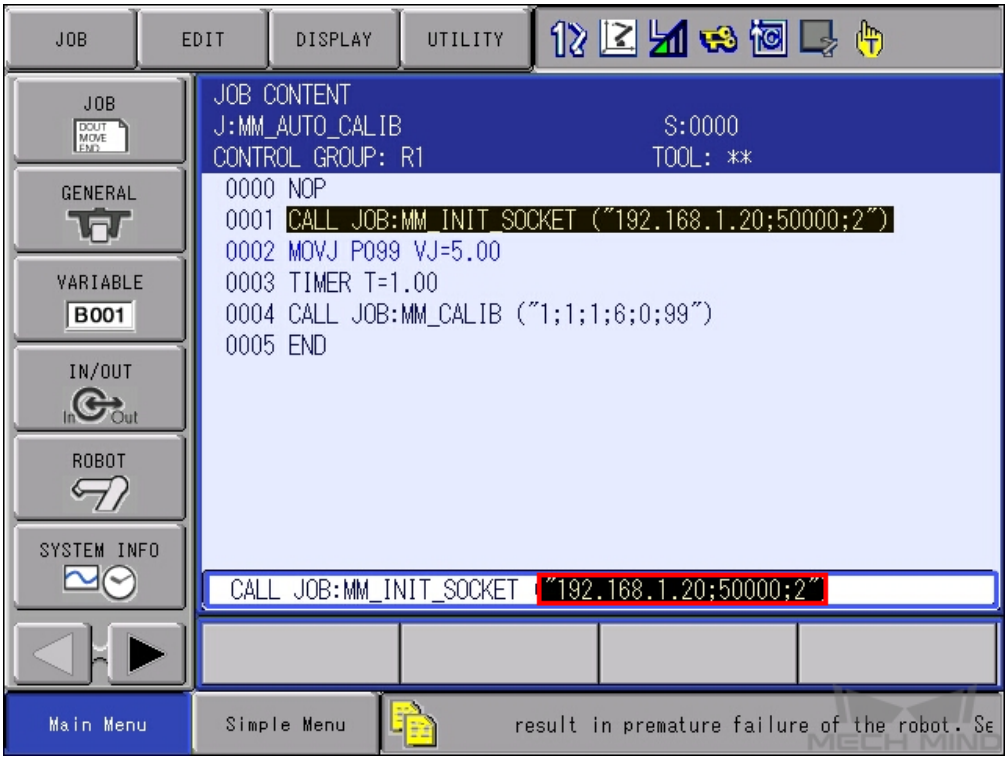
2. Select **MM_AUTO_CALIB** in the **JOB LIST**, and then press the SELECT key.



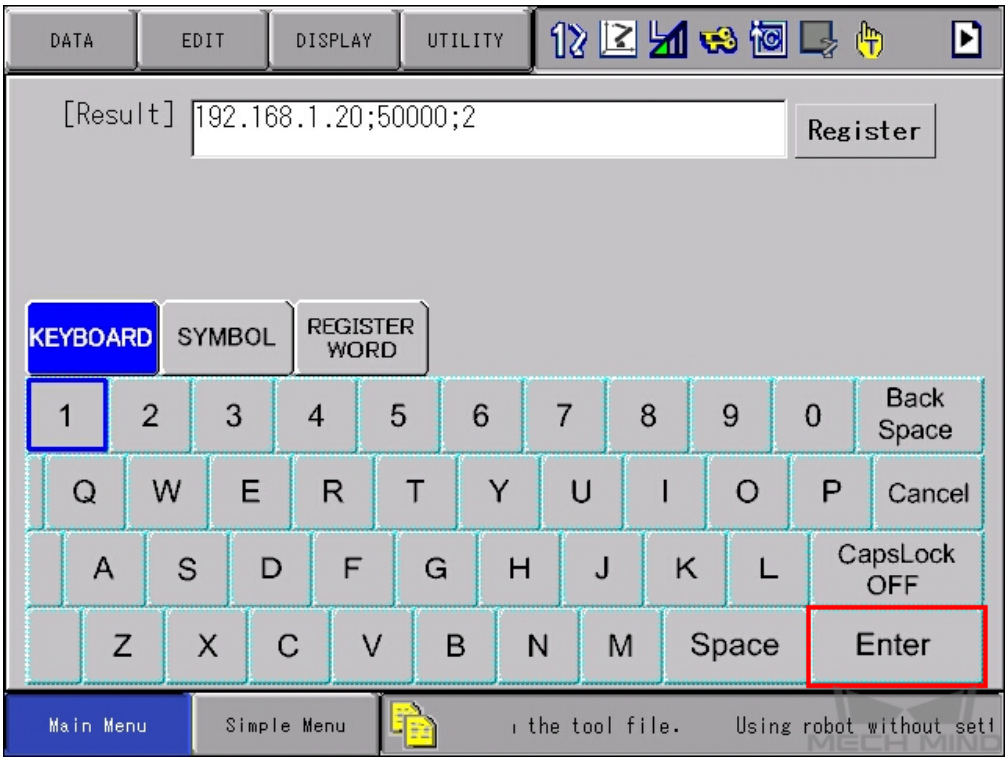
3. Change the IP address and port number in line 0001 to the actual ones of the IPC:
   1. Move the cursor to the instruction side of line 0001, and press `SELECT`. a text box will show on the bottom.



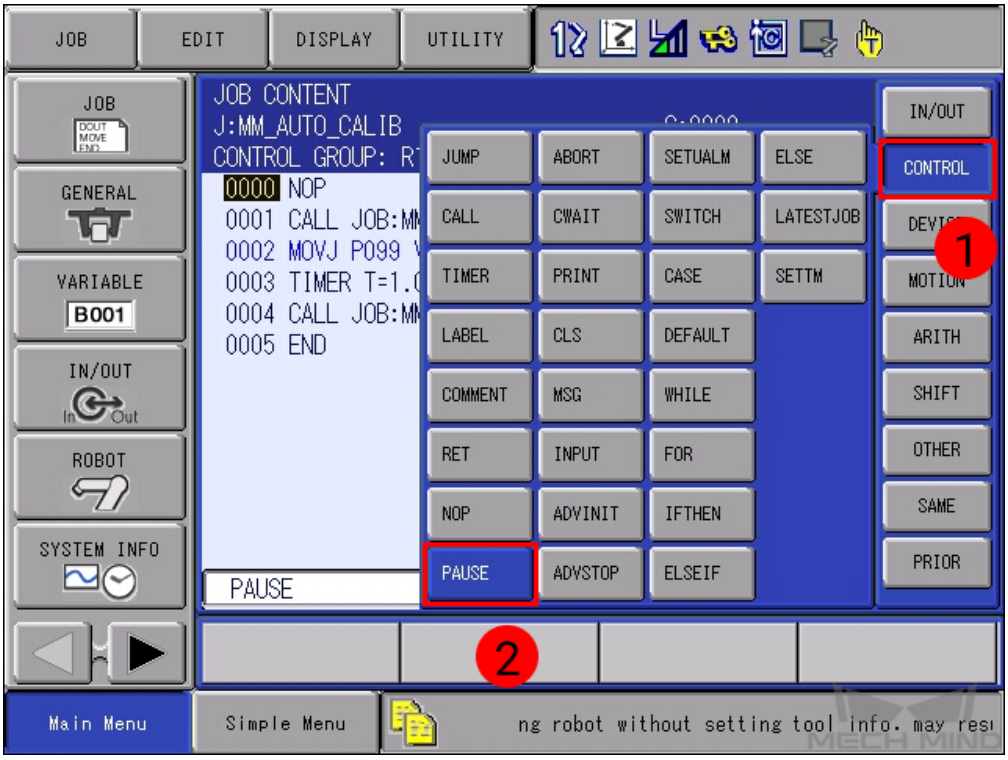
2. In the text box, move the cursor to the IP address and port number, and press ENTER.



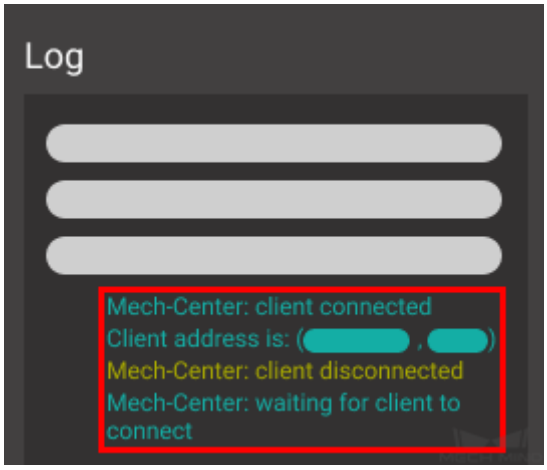
3. Change the IP address and port number, and then press on *Enter*.



4. Insert a **PAUSE** command after line 0001: make sure the cursor is on line 0001, and press `INFORM LIST`. Select *CONTROL* → *PAUSE* in the pop-up menu, and press `INSERT` and then `ENTER`.



5. Turn the mode switch to TEACH mode, press the `SERVO ON READY` key, and then hold down the enable switch on the back while moving the cursor back to line 0000.
6. Press the `INTERLOCK` key and `TEST START` key at the same time; the job will start running and should be paused after line 0001.
7. The robot can be successfully connected if Mech-Center' s **Log** panel displays the following messages:
   - **Mech-Center: client connected**
   - A message showing the **client address**
   - **Mech-Center: client disconnected**
   - **Mech-Center: waiting for client to connect**



Note: Delete **PAUSE** after testing the connection to avoid pausing the robot by mistake during calibration.

2.2.2 YASKAWA Calibration Program

This section introduces the process of calibrating the camera extrinsic parameters using the calibration program.

The process consists of 4 steps:

- *Select the Calibration Program*
- *Teach the Calibration Start Point*
- *Run the Calibration Program*
- *Start Calibration in Mech-Vision*

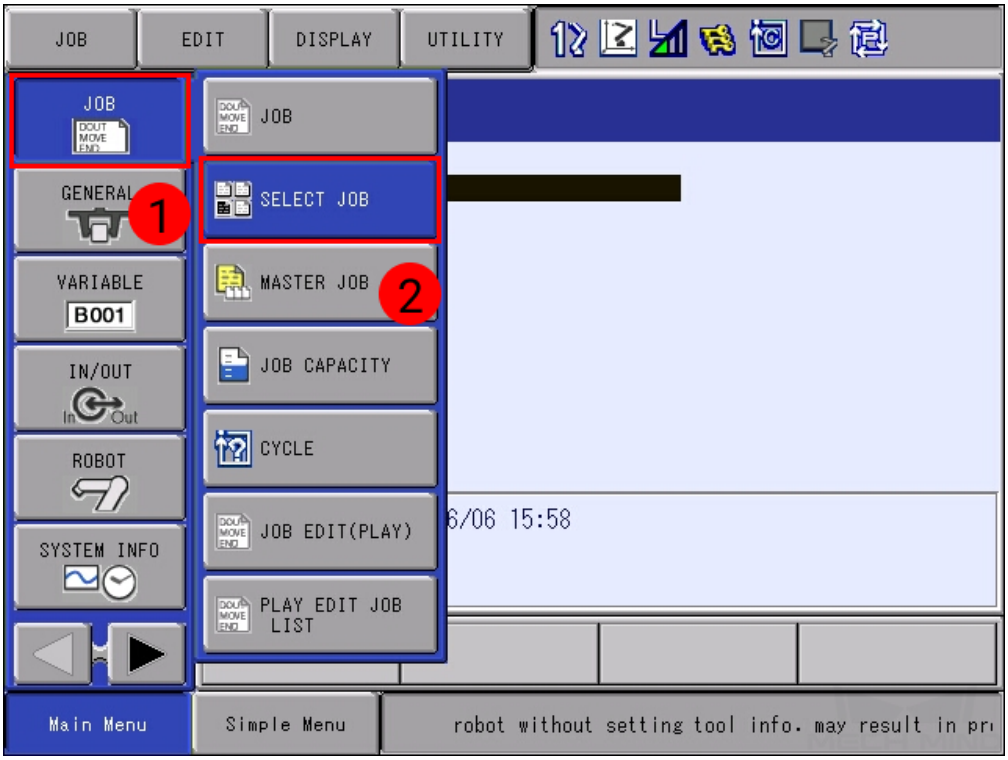
Before proceeding, please make sure that:

- You have *loaded the Standard Interface program* onto the robot and can establish communcation with Mech-Center.
- You are familiar with the contents in calibration_guide.

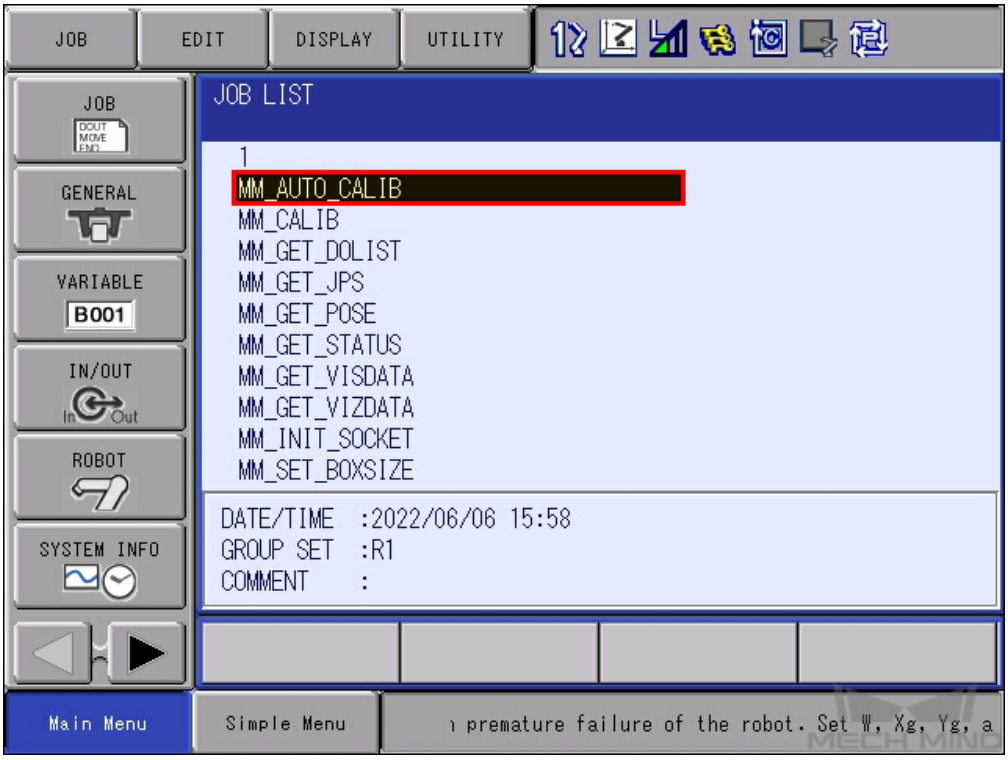
Note: This section is intended for scenarios where the communication between the robot and Mech-Center is established through Standard Interface, and calibration has to be performed frequently.

Select the Calibration Program

1. Select *JOB* → *SELECT JOB*.

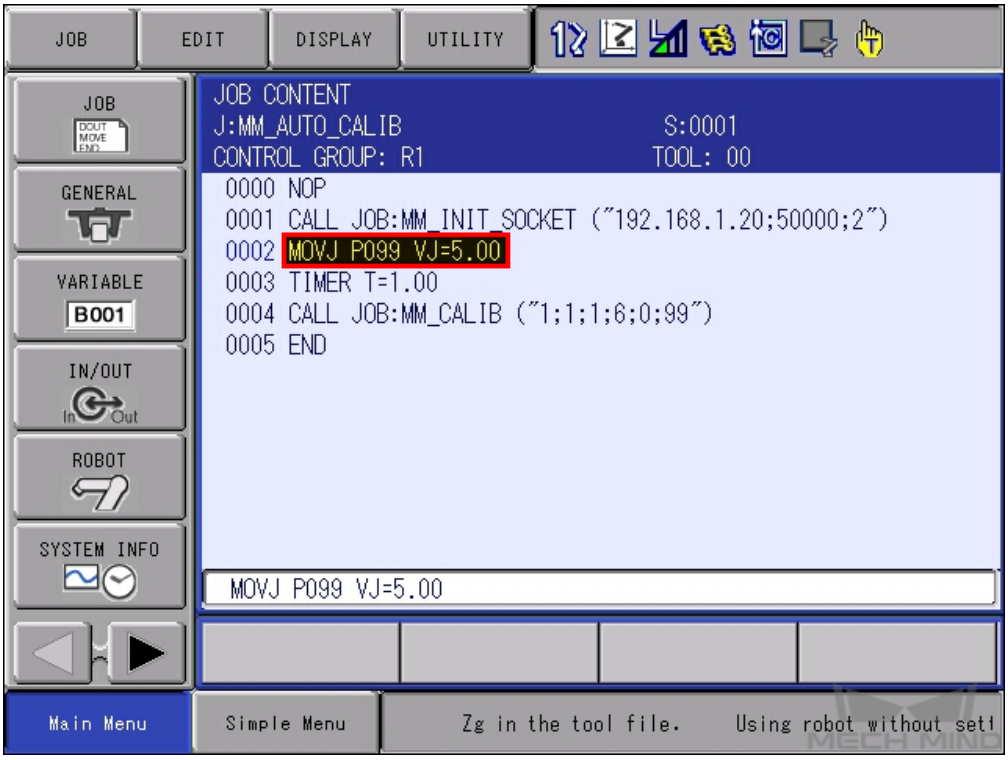


2. Select **MM_AUTO_CALIB** in the **JOB LIST**, and then press the `SELECT` key.

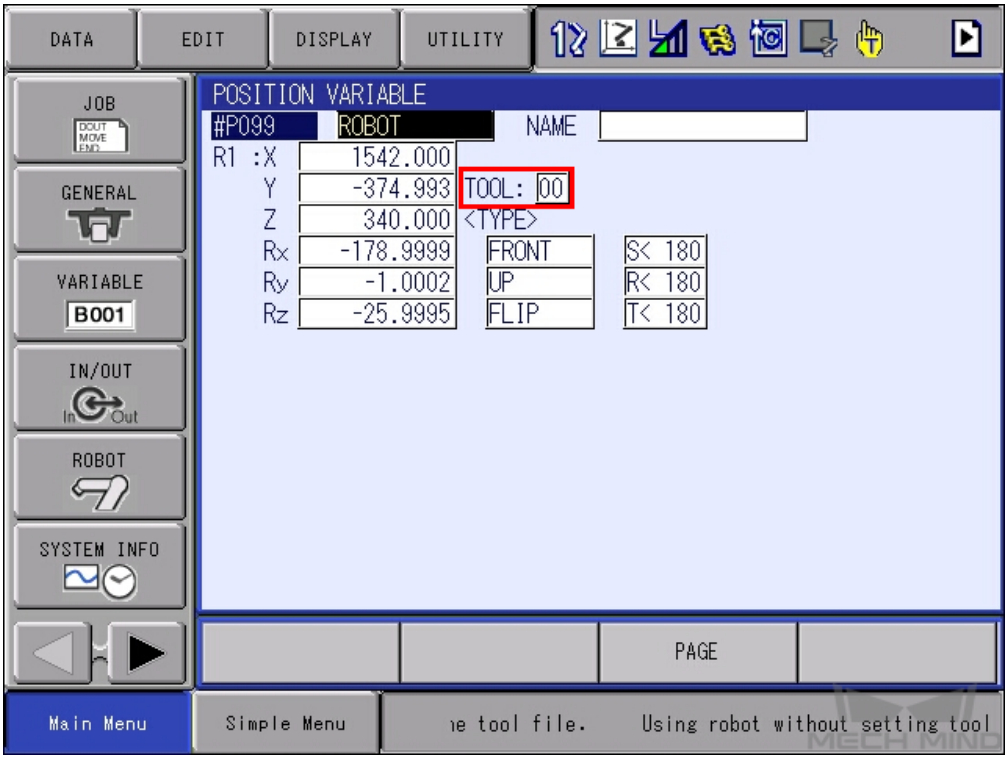


Teach the Calibration Start Point

1. Move the robot to the start point for the calibration.
2. Move the cursor to the instruction side of line 2, and press the `DIRECT OPEN` key.



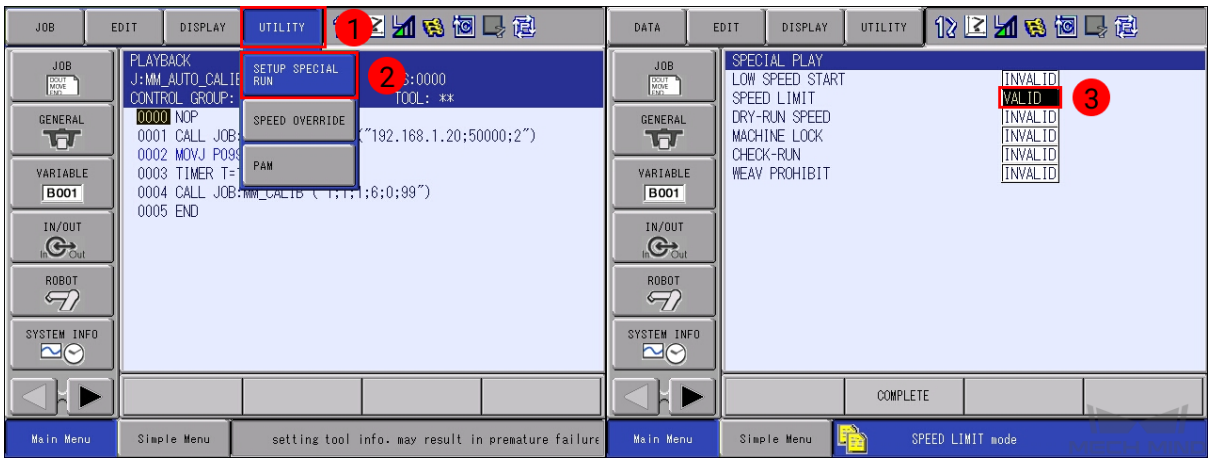
3. Change the value of **P099** to the current pose of the robot: press `SERVO ON READY` key, and then press `MODIFY` and then `ENTER` while holding down the enable switch on the back. Make sure the value for **TOOL** is **00**.



4. Press `DIRECT OPEN` again to return to the job.

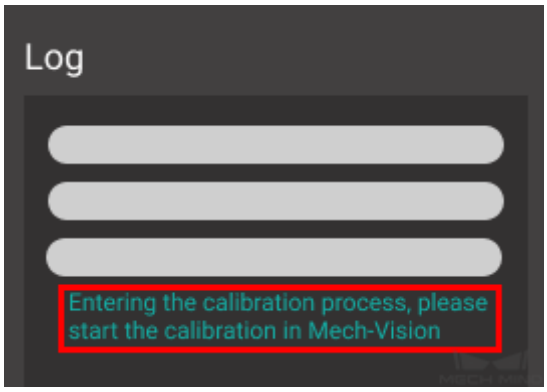
Run the Calibration Program

1. Move the cursor back to line 0000, and turn the mode switch to PLAY mode, and press the `SERVO ON READY` key.
2. To move the robot at low speed, go to *UTILITY* → *SETUP SPECIAL RUN*, and change **SPEED LIMIT** to **VALID**.



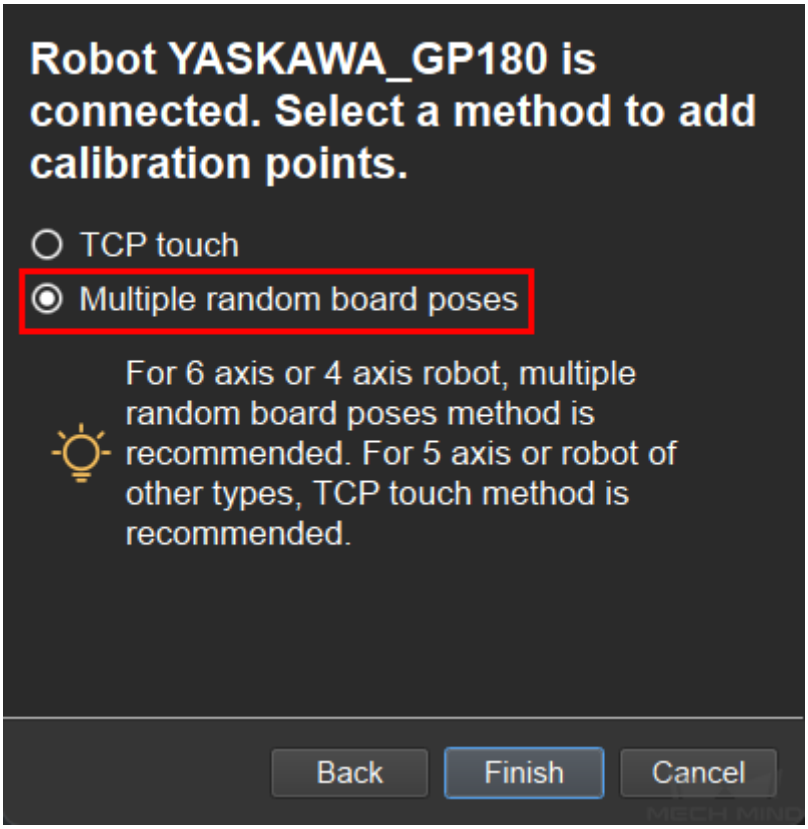
3. Press the `START` button. The program starts to run when the button lights up.
4. Proceed to the next section when the following message is displayed in Mech-Center' s **Log** panel:

   Entering the calibration process, please start the calibration in Mech-Vision

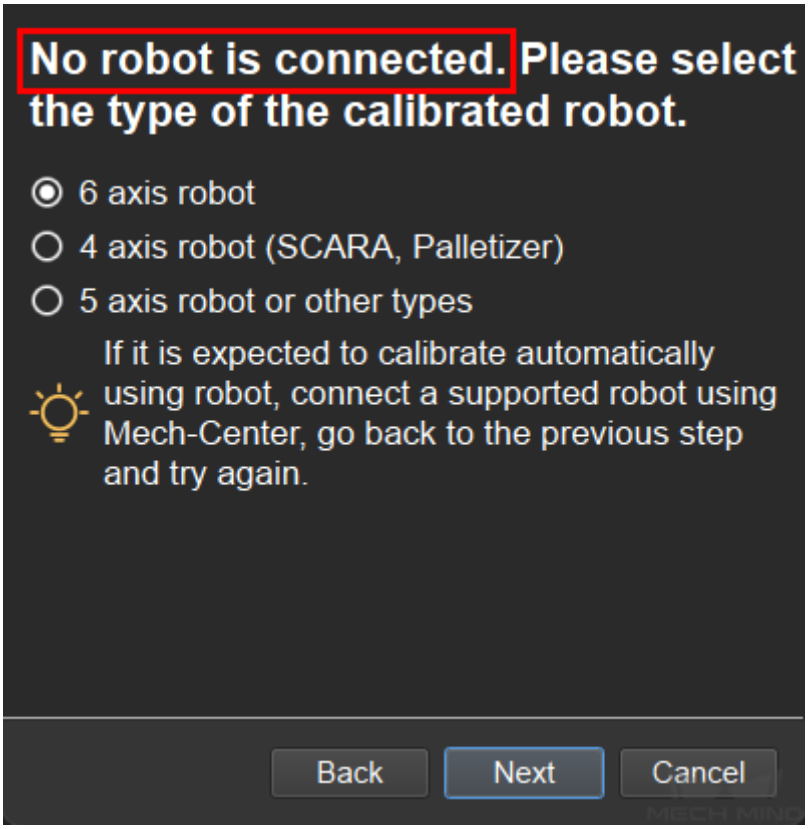


Start Calibration in Mech-Vision

1. In Mech-Vision, click on *Camera Calibration (Standard)* in the Toolbar, or select *Camera* → *Camera Calibration* → *Standard* from the Menu Bar.
2. Follow the instructions in Mech-Vision to complete the following configuration:
   1. Select **Start a new calibration process**;
   2. Select the camera mounting method;
   3. Select **Multiple random board poses** for adding calibration points.



Note: If after selecting the camera mounting method, the window says **No robot is connected**, the connection between the robot and Mech-Center is not properly established. Please re-run the robot program.



3. Follow the instructions in Mech-Vision to finish the calibration.

Note: In **5 Add Marker-Images and Poses** after you click on *Move Robot along Trajectory and Add Board Images*, if the robot does not reach the next calibration point within 60 seconds, Mech-Vision will report a timeout error and stop the calibration process. In such case, please select **calibration** in the program directory and run this program again, and restart the calibration process in Mech-Vision.

2.2.3 YASKAWA Example Program

This section introduces the example program provided with Mech-Center and the operations required to perform an actual pick-and-place task.

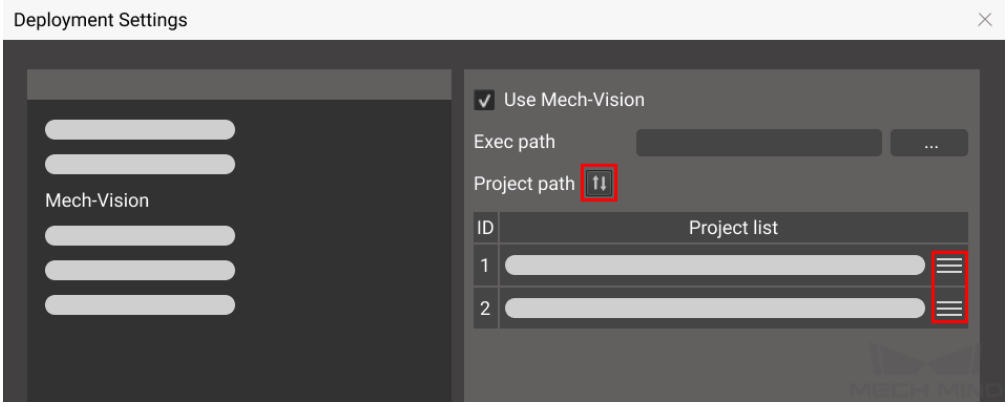
The example program **mm_sample.JBI** can be found in *Mech-Center/mech_interface/yaskawa*. It contains two parts: the first part obtains vision results from Mech-Vision; the second part obtains planned path from Mech-Viz.

Check the section corresponding to your own application setup:

- *Obtain Vision Results from Mech-Vision*
- *Obtain Planned Path from Mech-Viz*

Before running the program, please make sure that:

- You have *loaded the Standard Interface program* onto the robot and can establish communcation with Mech-Center.
- You have completed the extrinsic parameter calibration with *the calibration program* or by manually adding calibration points.
- Mech-Vision and Mech-Viz projects are created and set to autoload.
- The **Project list** in *Mech-Center* → *Deployment Settings* → *Mech-Vision* is synced by clicking on  , and the order of Mech-Vision projects have been adjusted according to actual needs.



- The TCP has been correctly specified.
- The robot speed is set to a low value, so that the operator can notice any unexpected behavior before accidents occur.

Obtain Vision Results from Mech-Vision

```
'****************************
'FUNCTION:simple pick and place
'with Mech-Vision
'mechmind, 2022-5-27
'****************************
'clear I50 to I69
CLEAR I050 20
'Initialize p variables
CNVRT PX070 PX070 RF TL#(1)
CNVRT PX071 PX071 RF TL#(1)
CNVRT PX072 PX072 RF TL#(1)
CNVRT PX073 PX073 RF TL#(1)
SUB P070 P070
SUB P071 P071
SUB P072 P072
SUB P073 P073
'move robot home position
MOVJ C00000 VJ=1.00
'camera capture position
MOVJ C00001 VJ=1.00 PL=0
'set ip address of IPC
```

(continues on next page)

(continued from previous page)

```
CALL JOB:MM_INIT_SOCKET ARGF"192.168.170.22;50000;1"
TIMER T=0.20
'set vision recipe
'MM_SET_MODEL("1,1")
'Run vision project
CALL JOB:MM_START_VIS ARGF"1;2;2"
TIMER T=1.00
'get result from Vis
CALL JOB:MM_GET_VISDATA ARGF"1;50;51;52"
TIMER T=0.20
PAUSE IF I052<>1100
'set the first pos to P071;
'set lables to I61;
'set speed to I62;
CALL JOB:MM_GET_POSE ARGF"1;71;61;62"
PAUSE IF I061<>1
'set  -200mm to Z
SETE P070 (3) -200000
MULMAT P072 P071 P070
'set  -300mm to Z
SETE P070 (3) -300000
MULMAT P073 P071 P070
'way point
MOVJ C00002 VJ=1.00
MOVL P072 V=80.0 PL=0
MOVL P071 V=80.0 PL=0
'enable girpper
DOUT OT#(1) ON
MOVL P073 V=80.0 PL=0
MOVJ C00003 VJ=1.00
'drop point
MOVJ C00004 VJ=1.00 PL=0
'release gripper
DOUT OT#(1) OFF
'way point
MOVJ C00005 VJ=1.00
'move robot home position
MOVJ C00006 VJ=1.00
```

Program Logic

1. Move the robot to HOME position.
2. Move the robot to the image capturing pose.
3. Initialize communication with **MM_INIT_SOCKET**.
4. If parameter recipes are used in the Mech-Vision project, the recipe to be used is set with **MM_SET_MODEL**.
5. Run the Mech-Vision project with **MM_START_VIS**.
6. Wait for 1 second. Under Eye-In-Hand, this **TIMER** instruction is required to make sure the robot stays still until image acquisition is completed. Under Eye-To-Hand, this **TIMER** instruction can be replaced with **MOVJ** or **MOVL**.

7. Obtain the vision results from Mech-Vision.
8. Check if the returned status code indicates any error. If an error code is returned, the program is paused.
9. Move the robot to the picking pose and perform picking.
10. Move the robot to a waypoint between the picking pose and placing pose.
11. Move the robot to the set placing pose and perform placing.
12. Return the robot to HOME position.

The following parts should be modified according to your actual needs.

Define the TCP

Change **TL#(1)** to the tool number to which the actual TCP is saved.

Note: Do not use TOOL 0 as it is used for calibration in **MM_AUTO_CALIB**.

Teach the Image Capturing Pose

Record the image capturing pose in **C00001** in line 54.

Teach the Waypoint(s)

Waypoints are intermediate poses between the picking pose and placing pose. They are used to ensure that the robot doesn' t collide with the surrounding when moving between the picking and placing poses.

Record the waypoints to **C00002** in line 79 and **C00005** in line 91. You can add more waypoints if needed.

Teach the Placing Pose

Record the placing pose in **C00004** in line 87.

Define Z-Offset from Picking/Placing Pose

Z-offset distances relative to the tool frame from the picking/placing pose are used to ensure collision doesn' t occur when the robot is approching or departing the picking/placing pose.

The **MULMAT** function is used to calculate and store the pose representing the offset from the picking pose in a new variable.

Adjust the following commands according to your actual needs.

- **SETE P070 (3) -200000** in line 73: the Z-offset when approching the picking pose is set to **200** mm. Robot will move to **P072**, which is 200 mm above the picking pose.

- **SETE P070 (3) -300000** in line 75: the Z-offset when departing the picking pose is set to **300** mm. Robot will move to **P073**, which is 300 mm above the picking pose.

Add Object Grasping and Releasing Logics

Modify **DOUT OT#(1) ON** in line 83 to the actual logic for controlling the tool action when picking the object.

Modify **DOUT OT#(1) OFF** in line 89 to the actual logic for controlling the tool action when placing the object.

Define HOME position

Teach HOME position to **C00000** in line 52.

Obtain Planned Path from Mech-Viz

```
94  '****************************
95  'FUNCTION:simple pick and place
96  'with Mech-Viz
97  'mechmind, 2022-5-27
98  '****************************
99  'clear I50 to I69
100 CLEAR I050 20
101 'Initialize p variables
102 CNVRT PX071 PX071 RF TL#(1)
103 CNVRT PX072 PX072 RF TL#(1)
104 CNVRT PX073 PX073 RF TL#(1)
105 SUB P071 P071
106 SUB P072 P072
107 SUB P073 P073
108 'move robot home position
109 MOVJ C00007 VJ=1.00
110 'camera capture position
111 MOVJ C00008 VJ=1.00 PL=0
112 'set ip address of IPC
113 CALL JOB:MM_INIT_SOCKET ARGF"192.168.170.22;50000;1"
114 TIMER T=0.20
115 'set vision recipe
116 CALL JOB:MM_SET_MODEL ARGF"1;1"
117 TIMER T=0.20
118 'Run Viz project
119 CALL JOB:MM_START_VIZ ARGF"1"
120 TIMER T=0.20
121 CALL JOB:MM_SET_BRANCH ARGF"1;1"
122 'get result from Viz
123 TIMER T=1.00
124 CALL JOB:MM_GET_VIZDATA ARGF"2;50;51;52;53"
125 TIMER T=0.20
126 PAUSE IF I053<>2100
127 'set the first pos to P071;
128 'set lables to I61;
```

(continues on next page)

(continued from previous page)

```
'set speed to I62;
CALL JOB:MM_GET_POSE ARGF"1;71;61;62"
PAUSE IF I061<>1
'set the second pos to P072;
'set lables to I63;
'set speed to I64;
CALL JOB:MM_GET_POSE ARGF"2;72;63;64"
PAUSE IF I063<>1
'set the third pos to P073;
'set lables to I65;
'set speed to I66;
CALL JOB:MM_GET_POSE ARGF"3;73;65;66"
PAUSE IF I065<>1
'way point
MOVJ C00009 VJ=1.00
MOVL P071 V=80.0 PL=0
MOVL P072 V=80.0 PL=0
'enable girpper
DOUT OT#(1) ON
MOVL P073 V=80.0 PL=0
'way point
MOVJ C00010 VJ=1.00
'drop point
MOVJ C00011 VJ=1.00 PL=0
'release gripper
DOUT OT#(1) OFF
'move robot home position
MOVJ C00012 VJ=1.00
```

Program Logic

1. Move the robot to HOME position.
2. Move the robot to the image capturing pose.
3. Initialize communication with **MM_INIT_SOCKET**.
4. If parameter recipes are used in the Mech-Vision project, the recipe to be used is set with **MM_SET_MODEL**.
5. Run the Mech-Viz project with **MM_START_VIZ**.
6. Obtain the planned path from Mech-Viz.
7. Check if the returned status code indicates any error. If an error code is returned, the program is halted.
8. Store obtained target points in the planned path to **P071**, **P072**, and **P073**.
9. Move the robot along the planned path and perform picking.
10. Move the robot to a waypoint between the picking pose and placing pose.
11. Move the robot to the set placing pose and perform placing.
12. Return the robot to HOME position.

The following parts should be modified according to your actual needs.

Define the TCP

Change **TL#(1)** to the tool number to which the actual TCP is saved.

Note: Do not use TOOL 0 as it is used for calibration in **MM_AUTO_CALIB**.

Teach the Image Capturing Pose

Record the image capturing pose in **C00001** in line 111.

Teach the Waypoint(s)

Waypoints are intermediate poses between the picking pose and placing pose. They are used to ensure that the robot doesn' t collide with the surrounding when moving between the picking and placing poses.

Record the waypoint to **C00009** in line 143. You can add more waypoints if needed.

Teach the Placing Pose

Record the placing pose in **C00011** in line 152.

Add Object Grasping and Releasing Logics

Modify **DOUT OT#(1) ON** in line 147 to the actual logic for controlling the tool action when picking the object.

Modify **DOUT OT#(1) OFF** in line 154 to the actual logic for controlling the tool action when placing the object.

Define HOME position

Teach HOME position to **C00007** in line 109.

2.2.4 YASKAWA Standard Interface Commands

The YASKAWA Standard Interface provides the following jobs:

- *Initialize Communication*
- *Start Mech-Vision Project*
- *Get Vision Result*
- *Start Mech-Viz Project*
- *Get Planned Path*
- *Obtain Pose*

- *Obtain Joint Positions*
- *Switch Mech-Vision Recipe*
- *Select Mech-Viz Branch*
- *Set Move Index*
- *Get Software Status*
- *Input Object Dimensions to Mech-Vision*
- *Get DO Signal List*
- *Input TCP to Mech-Viz*
- *Calibration*

When writing your own program, pay attention to the following:

- Multiple parameters should be separated by semi-colons.
- When calling jobs: input argument is a string by default; output argument is a string by default, each element in the string corresponding to a global variable in the background.

This Standard Interface is over the TCP/IP protocol.

Initialize Communication

```
MM_INIT_SOCKET ("IP_Address;Server_Port;Time_Out")
```

This job sets the IP address and port number of the IPC and wait time before the program stops trying to establish the communication.

Arguments

- Input arguments

Name	Description
IP_Address	IP address of the IPC
Server_Port	Port number of the IPC, the default is 50000
Time_Out	Wait time in minutes before stopping connection attempt

Example

```
CALL JOB:MM_INIT_SOCKET ("192.168.1.1;50000;5")
```

This example sets the IP address and port number of the IPC to 192.168.1.1:50000 and wait time to 5 minutes.

Start Mech-Vision Project

```
MM_START_VIS ("Job;Pos_Num_Need;SendPos_Type")
```

This job is for applications that use Mech-Vision but not Mech-Viz. It runs the corresponding Mech-Vision project to acquire and process data.

Arguments

- Input arguments

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center → Deployment Settings → Mech-Vision*
Pos_Num_Need	Number of poses for Mech-Vision to send, from 0 to 20, where 0 means "send all".
SendPos_Type	Set the image capturing pose for the robot to send, from 0 to 2
	0: Do not send image capturing pose (for Eye To Hand) 1: Send image capturing pose as joint positions 2: Send image capturing pose as robot flange pose

Example

```
CALL JOB:MM_START_VIS ("1;1;1")
```

This example runs Mech-Vision project No. 1, and asks the project to send over 1 pose, and the robot sends its joint positions when this job is called as the image capturing pose to Mech-Center.

Get Vision Result

```
MM_GET_VISDATA ("Job;Last_Data;Pose_Num;MM_Status")
```

This job is for applications that use Mech-Vision but not Mech-Viz. It obtains the vision result from the corresponding Mech-Vision project.

Arguments

- Input argument

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center → Deployment Settings → Mech-Vision*

- Output arguments

Name	Description
Last_Data	I variable, indicating whether all vision result has been sent, 0 or 1
	0: NOT all vision result has been sent (more on the way) 1: All vision result has been sent If 0, call this JOB again until all are sent
Pose_Num	I variable for storing the number of received poses
MM_Status	I variable for storing status code, refer to the standard_interface_status_codes

Example

```
CALL JOB:MM_GET_VISDATA ("1;50;51;52")
```

This example obtains the vision result from Mech-Vision project No. 1. Whether all vision result has been sent is stored in **I50**, the number of poses received is stored in **I51**, and the status code is stored in **I52**.

Start Mech-Viz Project

```
MM_START_VIZ ("SendPos_Type")
```

This job is for applications that use both Mech-Vision and Mech-Viz. It runs the corresponding Mech-Viz project (which triggers the corresponding Mech-Vision project to run), and sets the initial joint positions of the simulated robot in Mech-Viz.

Argument

- Input argument

Name	Description
SendPos_Type	The initial joint positions for the simulated robot in Mech-Viz, 0 or 1
	0: Set the initial joint positions of the simulated robot to [0,0,0,0,0,0] 1: Set the initial joint positions of the simulated robot to the current joint positions of the real robot

Note: When the scene contains object models that obstruct the robot to move from [0,0,0,0,0,0] to the first target point, this parameter must be set to 1.

Example

```
CALL JOB:MM_START_VIZ ("1")
```

This example runs the corresponding Mech-Viz project, and sets the initial joint positions of the simulated robot to the current joint positions of the real robot.

Get Planned Path

```
MM_GET_VIZDATA ("GetPos_Type;Last_Data;Pos_Num;VisPos_Num;MM_Status")
```

This job obtains the planned path from Mech-Viz.

Arguments

- Input argument

Name	Description
GetPos_Type	Whether Mech-Viz should send target points as joint positions or TCPs, 1 or 2
	1: Mech-Viz sends joint positions 2: Mech-Viz sends TCPs

- Output arguments

Name	Description
Last_Data	Invariable, indicating whether all target points have been sent, 0 or 1
	0: NOT all target points have been sent (more on the way) 1: All target points have been sent If 0, call this JOB again until all are sent
Pos_Num	Invariable for storing the number of received target points
VisPos_Num	Invariable for storing the position of the first visual_move target point in the path
	Example path: move-1, move-2, visual_move-3, move-3, visual_move-2 In this path, the position of the first visual_move target point is 3. If the path does not contain visual_move target point, the return value is 0.
MM_Status	Invariable for storing status code, refer to the standard_interface_status_codes

Example

```
CALL JOB:MM_GET_VIZDATA ("2;50;51;52;53")
```

This example obtains the planned path from Mech-Viz in the form of TCPs. Whether all target points have been sent is stored in **I50**, the number of target points received is stored in **I51**, the position of the visual_move target point is stored in **I52**, and the status code is stored in **I53**.

Obtain Pose

```
MM_GET_POSE ("Index;PosTarget;Label;Pose_Speed")
```

This job stores a pose returned by Mech-Vision or a target point (as TCP) returned by Mech-Viz in the specified variable.

Arguments

- Input argument

Name	Description
Index	Specify the index of the pose to be stored

- Output arguments

Name	Description
PosTarget	P variable for storing the specified pose
Label	I variable for storing the label corresponding to the specified pose
Pose_Speed	I variable for storing the speed corresponding to the specified pose

Example

```
CALL JOB:MM_GET_POSE ("1;60;61;62")
```

This example stores the first received pose to **P60**, the corresponding label to **I61**, and the corresponding speed to **I62**.

Obtain Joint Positions

```
MM_GET_JPS ("Index;JointTarget;Label;Pose_Speed")
```

This job stores a set of joint positions returned by Mech-Viz in the specified variable.

Note: As Mech-Vision does not output joint position data, this job can only be used with Mech-Viz.

Arguments

- Input argument

Name	Description
Index	Specify the index of the set of joint positions to be stored

- Output arguments

Name	Description
JointTarget	P variable for storing the specified set of joint positions
Label	I variable for storing the label corresponding to the specified set of joint positions
Pose_Speed	I variable for storing the speed corresponding to the specified set of joint positions

Example

```
CALL JOB:MM_GET_JPS ("1;60;61;62")
```

This example stores the first set of received joint positions to **P60**, the corresponding label to **I61**, and the corresponding speed to **I62**.

Switch Mech-Vision Recipe

```
MM_SET_MODEL ("Job;Model_Number")
```

This job specifies which parameter recipe of the Mech-Vision project to use. For more information on parameter recipe, please see parameter_recipe_configuration.

Note:

- This job must be called BEFORE **MM_START_VIS**.
- The corresponding Mech-Vision project must have parameter recipes already configured and saved.

Arguments

- Input arguments

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center* → *Deployment Settings* → *Mech-Vision*
Model_Number	The number of a parameter recipe in the Mech-Vision project, from 1 to 99

Example

```
CALL JOB:MM_SET_MODEL ("1;1")
```

This example switches the parameter recipe used to No. 1 in Mech-Vision project No. 1.

Select Mech-Viz Branch

```
MM_SET_BRANCH ("Branch_Num;Exit_Num")
```

This job is used to select along which branch the Mech-Viz project should proceed. Such branching is achieved by adding branch_by_service_message Task(s) to the project. This job specifies which out port such Task(s) should take.

Note:

- **MM_START_VIZ** must be called BEFORE this job.

- When the next Task to be executed in Mech-Viz is a **branch_by_service_message** Task, Mech-Viz will wait for this job to send the out port number it should take.
- The name of all **branch_by_service_message** Tasks in the Mech-Viz project must be changed to numbers between 1 and 99, and the names should be unique among all tasks in the project.

Arguments

- Input arguments

Name	Description
Branch_Num	Name of the **branch_by_service_message** Task, from 1 to 99
Exit_Num	The number of the out port to take, from 1 to 99

Example

```
CALL MM_SET_BRANCH ("1;3")
```

This example tells Mech-Viz to take out port 3 for the **branch_by_service_message** Task named **1**.

Set Move Index

```
MM_SET_INDEX    ("Skill_Num;Index_Num")
```

This job sets the value for the Current Index parameter of Mech-Viz Tasks. Tasks that have this parameter include move_list, move_grid, custom_pallet_pattern, and smart_pallet_pattern.

Note:

- **mm_start_viz** must be called BEFORE this job.
- The name of all Tasks with index parameters in the Mech-Viz project must be changed to numbers between 1 and 99, and the names should be unique among all tasks in the project.

Arguments

- Input arguments

Name	Description
Skill_Num	Name of the Task, from 1 to 99
Index_Num	Value for the Current Index parameter when the Task is executed

Example

```
CALL JOB:MM_SET_INDEX ("2;10")
```

This example sets the Current Index value to 9 for the Task named **2**. When the Task is executed, the Current Index value will be added 1 and become 10.

Get Software Status

```
MM_GET_STATUS ("Status")
```

This job is currently capable of checking whether Mech-Vision is ready to run projects. In the future, this job can be used for obtaining the execution status of Mech-Vision, Mech-Viz and Mech-Center.

Argument

- Output argument

Name	Description
Status	I variable for storing the status code, refer to the standard_interface_status_codes

Example

```
CALL JOB:MM_GET_STATUS ("70")
```

This example obtains the status code and stores it in **I70**.

Input Object Dimensions to Mech-Vision

```
MM_SET_BOXSIZE("Job;Length;Width;Height")
```

This job inputs object dimensions to the Mech-Vision project.

Note:

- This job must be called BEFORE **MM_START_VIS**.

Arguments

- Input arguments

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center → Deployment Settings → Mech-Vision*
Length	Length of object in mm
Width	Width of object in mm
Height	Height of object in mm

Example

```
CALL JOB:MM_SET_BOXSIZE ("1;500;300;200")
```

This example sets the object dimensions in the read_object_dimensions Step in the Mech-Vision project No. 1 to 500*300*200 mm.

Get DO Signal List

```
MM_GET_DOLIST
```

This job obtains the planned DO Signal list for controlling multiple sections of a sectioned vacuum gripper.

Note:

- **MM_GET_VIZDATA** must be called BEFORE this job.
- Please deploy the Mech-Viz project based on the template project in *Mech-Center/tool/viz_project/suction_zone*, and set the suction cup configuration file in the Mech-Viz project.

Arguments

No parameters.

Example

```
CALL JOB:MM_GET_DOLIST
```

This example obtains the DO signal list planned by Mech-Viz and writes the values in **OT1** – **OT16**.

Input TCP to Mech-Viz

```
MM_SET_POSE ("Pos")
```

This job inputs TCP data to the outer_move Task.

Note:

- This job must be called BEFORE **MM_START_VIZ**.
- Please deploy the Mech-Viz project based on the template project in *Mech-Centertoolviz_projectouter_move*, and put the **outer_move** Task at a proper position in the workflow.

Argument

- Input argument

Name	Description
Pos	P variable for storing the TCP data to be sent to Mech-Viz

Example

```
CALL JOB:MM_SET_POSE ("10")
```

This example sends the TCP data stored in **P10** to the **outer_move** task in the Mech-Viz project.

Calibration

```
MM_CALIB ("Move_Type;Pos_Jps;Wait_Time;Rnum;Ext;Pos")
```

This job is used for hand-eye calibration (camera extrinsic parameter calibration). It automates the calibration process in conjunction with the **Camera Calibration** function in Mech-Vision. For detailed instructions, see *KUKA Calibration Program*.

Arguments

- Input arguments

Name	Description
Move_Type	Motion type, 1 or 2
	1: MOVL 2: MOVJ
Pos_Jps	Pose as flange pose or joint positions, 1 or 2
	1: flange pose 2: Joint positions
Wait_Time	Wait time before stopping communication attempt in minutes
Rnum	Number of robot axes
Ext	Data of the external 7th axis in mm, optional
Pos	Start point for the calibration, P99 by default

Example

```
CALL JOB:MM_CALIB ("2;1;5;6;0;99")
```

This example moves the robot with MOVJ, receives pose data in the form of flange pose, and sets the wait time to 5 minutes. This robot has 6 axes and does not have an external axis installed. The start point for the calibration is stored in **P99**.

2.2.5 YASKAWA Error Messages

The following errors may occur while running the Standard Interface program on the robot.

MM:Robot_Internal_Error

Error occurred while the MotoPlus application attempts to call the MotoPlus API.

Example

When the MotoPlus application attempts to obtain the current pose of the robot, ret = -1.

```
ret = mpGetCartPos(&cgsData, &cartPosData);
if (ret != 0)
return INTERNAL_ERROR;
```

Troubleshooting

Please refer to *Programmer' s Manual for New Language Environment MotoPlus* and check the application.

MM:Robot_Socket_Closed

Error occurred when the MotoPlus application called the **mpSocket** and **mpConnect** functions, and the robot is disconnected from Mech-Center.

Troubleshooting

- Check if the hardware are properly connected.
- Check if the Standard Interface is started in Mech-Center.
- Check the IP addresses of the robot and the IPC, and if the port number is configured correctly.
- Check if the firewall is turned off on the IPC.
- Contact Mech-Mind Technical Support for further assistance.

MM:Robot_Argument_Error

When calling a Mech-Mind Standard Interface job, arguments provided are not sufficient.

Example

When calling **MM_START_VIS**, 3 arguments should be provided. If only 1 argument is provided, this error is reported.

- Correct:

```
CALL JOB:MM_START_VIS ("1;1;2")
```

- Incorrect:

```
CALL JOB:MM_START_VIS ("1")
```

Troubleshooting

Please refer to *YASKAWA Standard Interface Commands* and input the correct arguments accordingly.

MM:Robot_CMD_Error

- The command code sent by the job does not exist;
- The command code received by the MotoPlus Application does not match the one sent.

Troubleshooting

- Please refer to the Standard Interface Development Manual and make sure the command code sent by the job is correct.
- The sequence of command sending and receiving is problematic. Please contact Mech-Mind Technical Support for further assistance.

MM:IPC_Return_Error

Returned status code is an error code. Please check Mech-Center' s log.

Troubleshooting

- Please refer to the standard_interface_status_codes for the specific error.
- Please contact Mech-Mind Technical Support for further assistance.

2.3 FANUC

This section introduces the Standard Interface for FANUC robots.

2.3.1 FANUC Setup Instructions

This section introduces the process of loading the standard interface program onto a FANUC robot.

The process consists of 4 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program Files*
- *Test Robot Connection*

Please have a flash drive ready at hand.

Check Controller and Software Compatibility

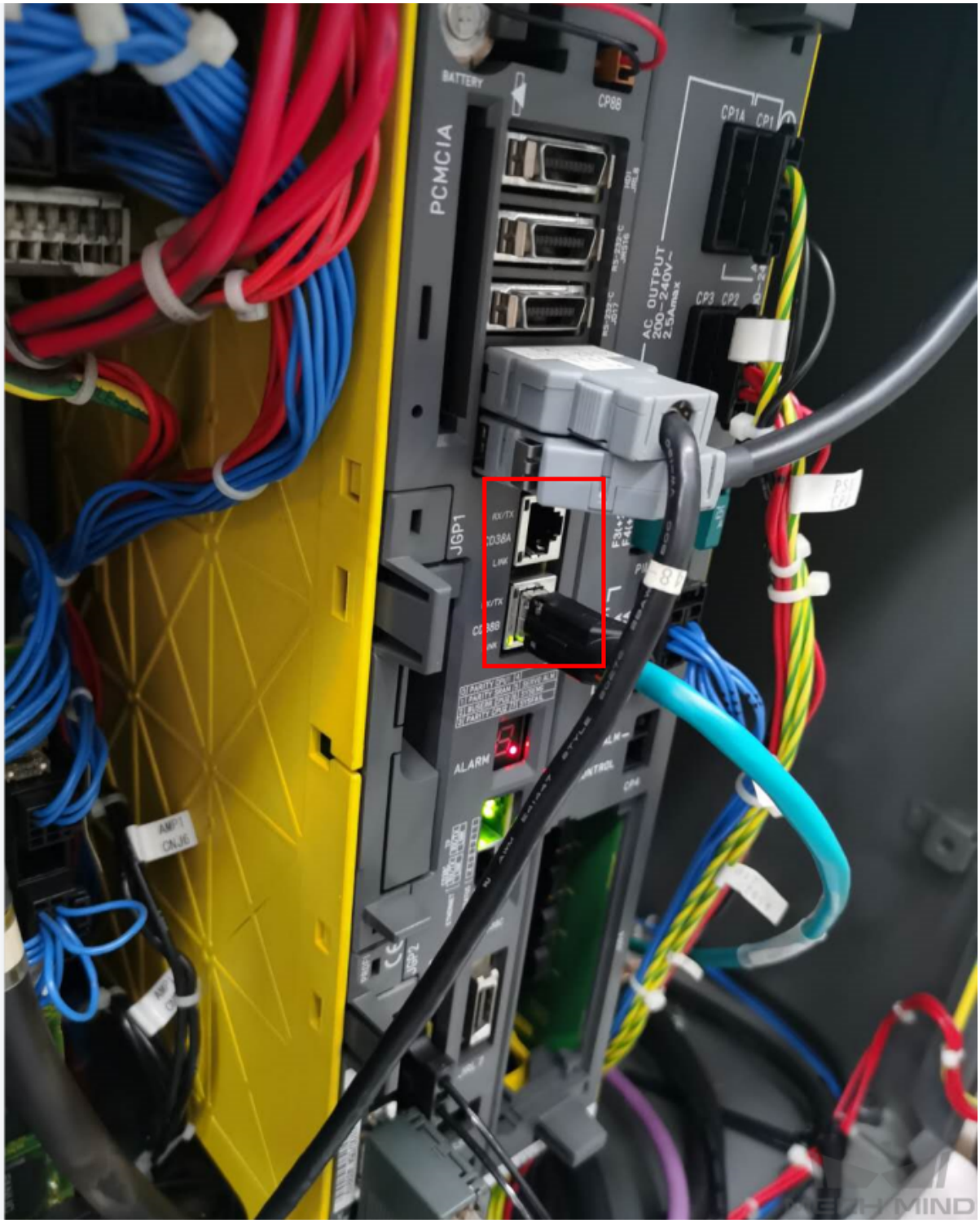
Compatibility requirements are as follows:

- Robot: a 6-axis or 7-axis FANUC robot
- Additional controller software packages:
  - R651 or R632 (karel)
  - R648 (User Socket Msg)
- Controller system software version: V7.5, V7.7, V8.*, and V9.*

Setup the Network Connection

Hardware Connection

Plug the Ethernet cable of the IPC into the Ethernet port of robot controller as shown in the figure. You can plug the cable into either CD38A port or CD38B port. CD38A corresponds to **Port#1** in the robot IP setting, while CD38B corresponds to **Port#2**.

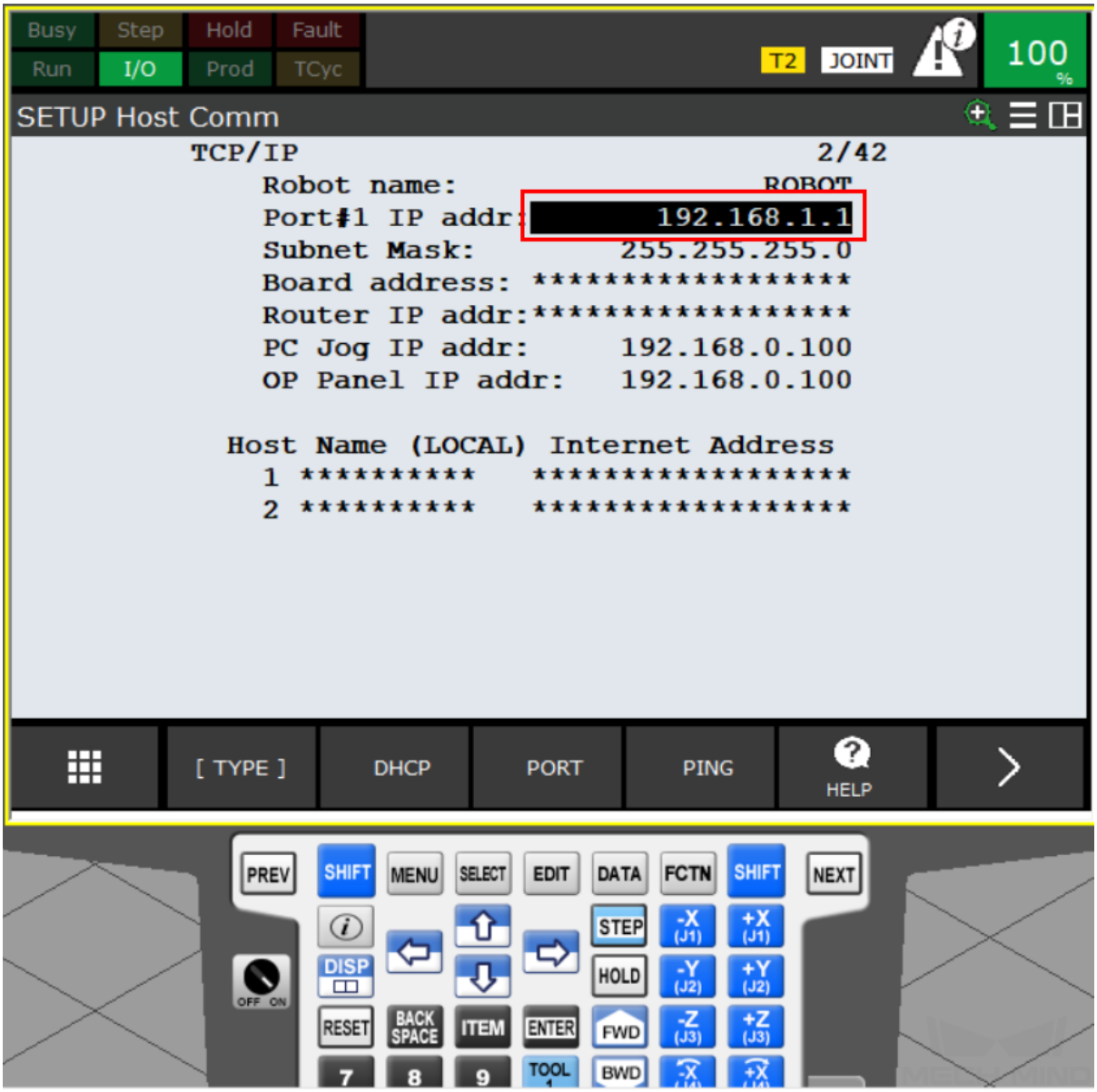

BATTERY
CP8B
PCMCIA
CP1A
CP1
AC OUTPUT
200–240V~
2.5Amax
CP3
CP2
JGP1
RX/TX
LINK
ALARM
CP4

IP Configuration

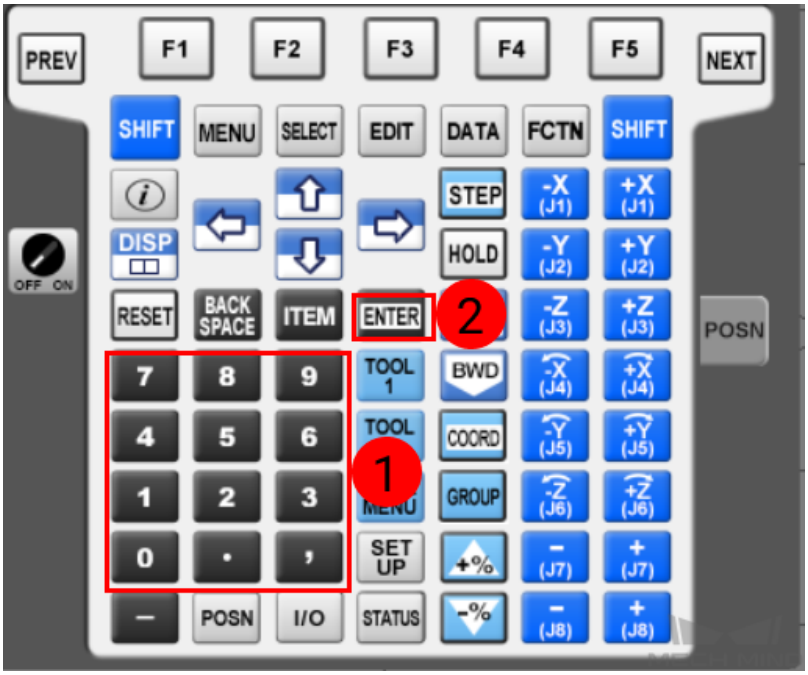
1. Press *MENU*→ *SETUP*, and then select **Host Comm** in the context menu, and then press `ENTER` to open the **SETUP Protocols** window.

2. Select **TCP/IP** and press *DETAIL* to open the **SETUP Host Comm** window.

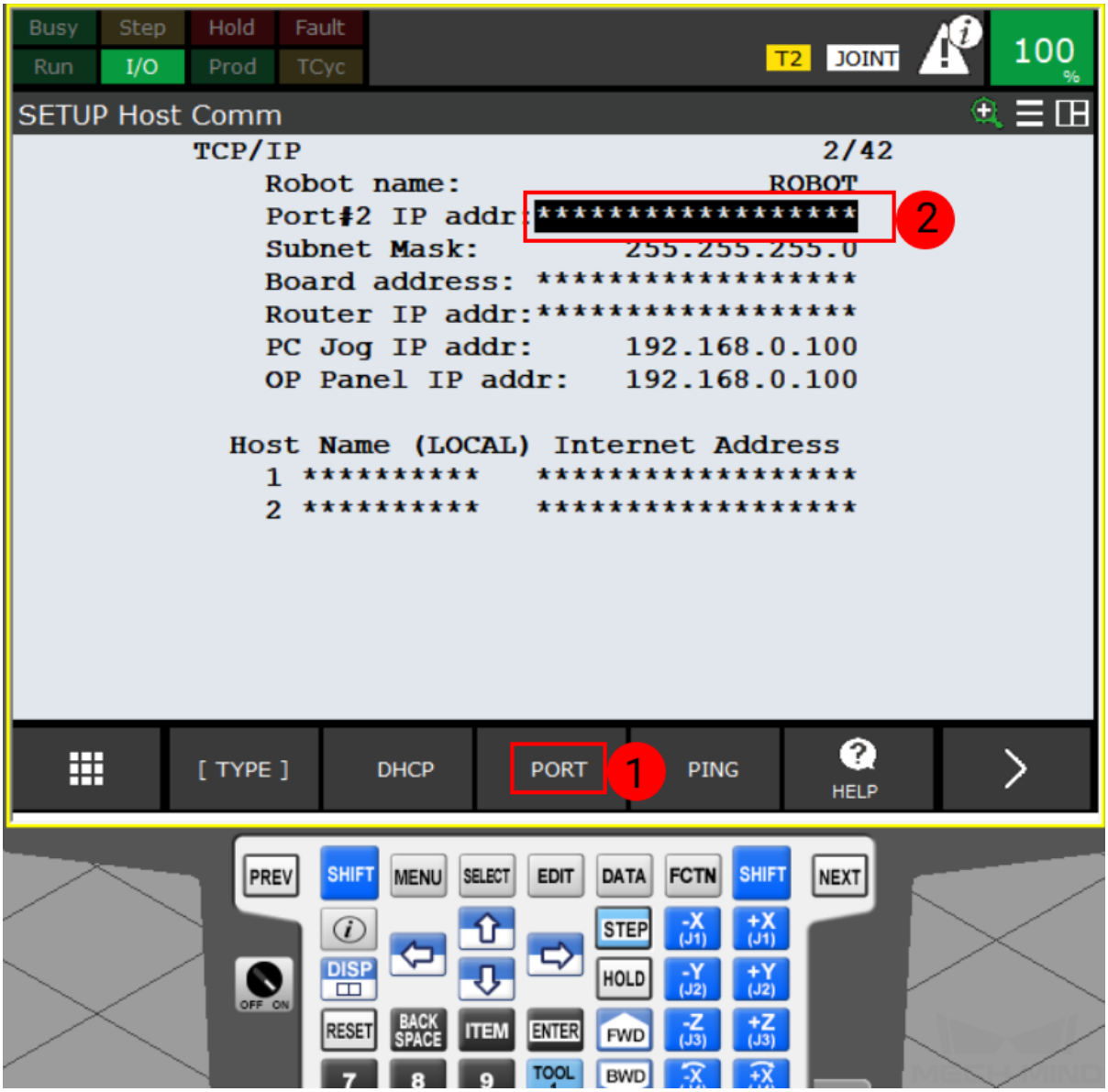
3. Enter the robot IP in the **IP address** line with the keyboard of the teach pendant. The robot IP should be in the same subnet as the IPC.


Busy
Step
Hold
Fault
Run
I/O
Prod
TCyc
T2
JOINT
100
%
SETUP Host Comm
TCP/IP
2/42
Robot name:
ROBOT
Port#1 IP addr:
192.168.1.1
Subnet Mask:
255.255.255.0
Board address: **********
Router IP addr:**********
PC Jog IP addr:
192.168.0.100
OP Panel IP addr:
192.168.0.100
Host Name (LOCAL) Internet Address
1 ********** **********
2 ********** **********
[TYPE]
DHCP
PORT
PING
HELP

PREV
SHIFT
MENU
SELECT
EDIT
DATA
FCTN
SHIFT
NEXT
DISP
STEP
HOLD
RESET
BACK SPACE
ITEM
ENTER
FWD
BWD
TOOL
-X (J1)
+X (J1)
-Y (J2)
+Y (J2)
-Z (J3)
+Z (J3)
OFF ON
7
8
9



4. If the Ethernet cable is connected to port 2, please press *Port* to switch the port. Then you can enter the robot IP in the **IP address** line.



Load the Program Files

Prepare the Files

The program files are stored in the installation directory of Mech-Center. The default directory for Mech-Center 1.5.2 is *C:/Mech-Mind/Mech-Center*.

Navigate to *xxx/Mech-Center/Mech-interface/fanuc*, and copy all the contents of this folder to your flash drive.

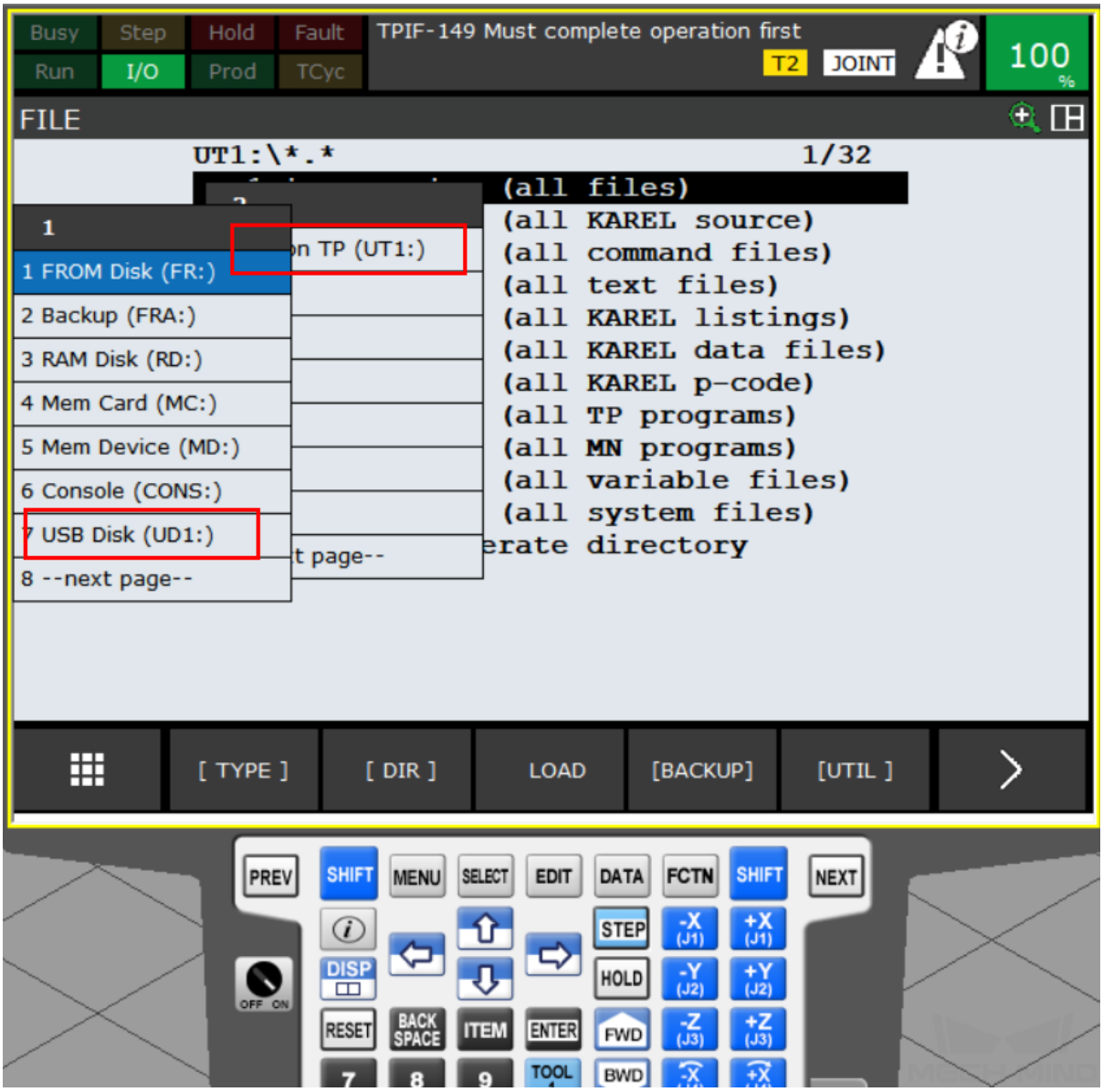
Note: The folders and files must be saved in the root directory of the USB flash drive. Do not rename them.

Load the Files to the Robot

1. After connecting the USB flash drive, press `MENU` and select *FILE* → *File*, and then press `ENTER` to open the **FILE** window.

2. Press *UTIL* and select **Set Device** in the context menu.

3. Select the USB flash drive. If your flash drive is connected to the **controller**, please select **USB Disk (UD1:)**; if your USB flash drive is connected to the **teach pendant**, please select **USB on TP (UT1:)**.



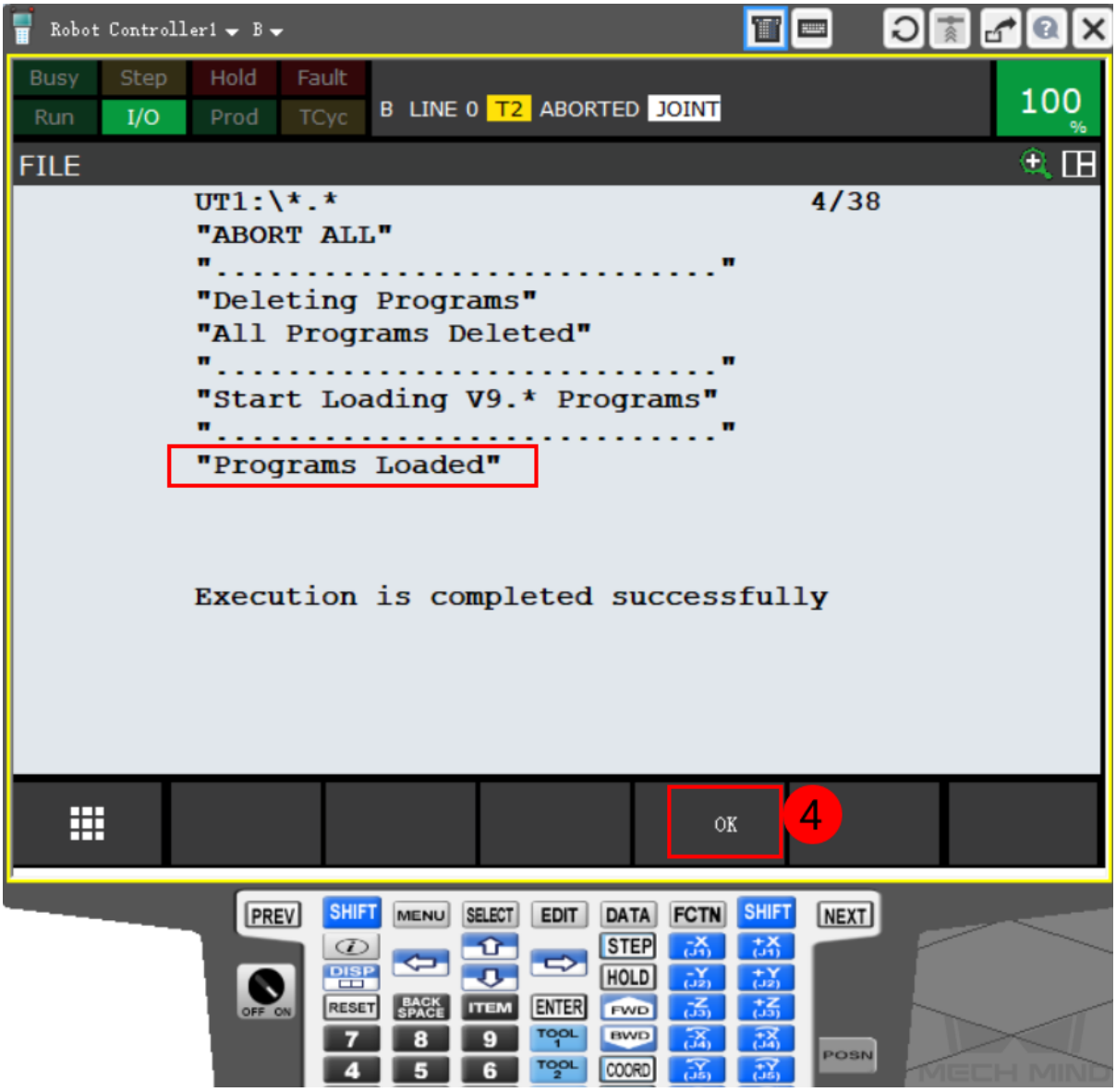
4. Select the first line **(all files)** and press `ENTER` to enter the root directory of the USB flash drive.

Hint: For the next step:

- If the USB flash drive is connected to the **robot controller**, please select **INSTALL_UD.cm**.
- If the USB flash drive is connected to the **robot teach pendant**, please select **INSTALL_UT.cm**.

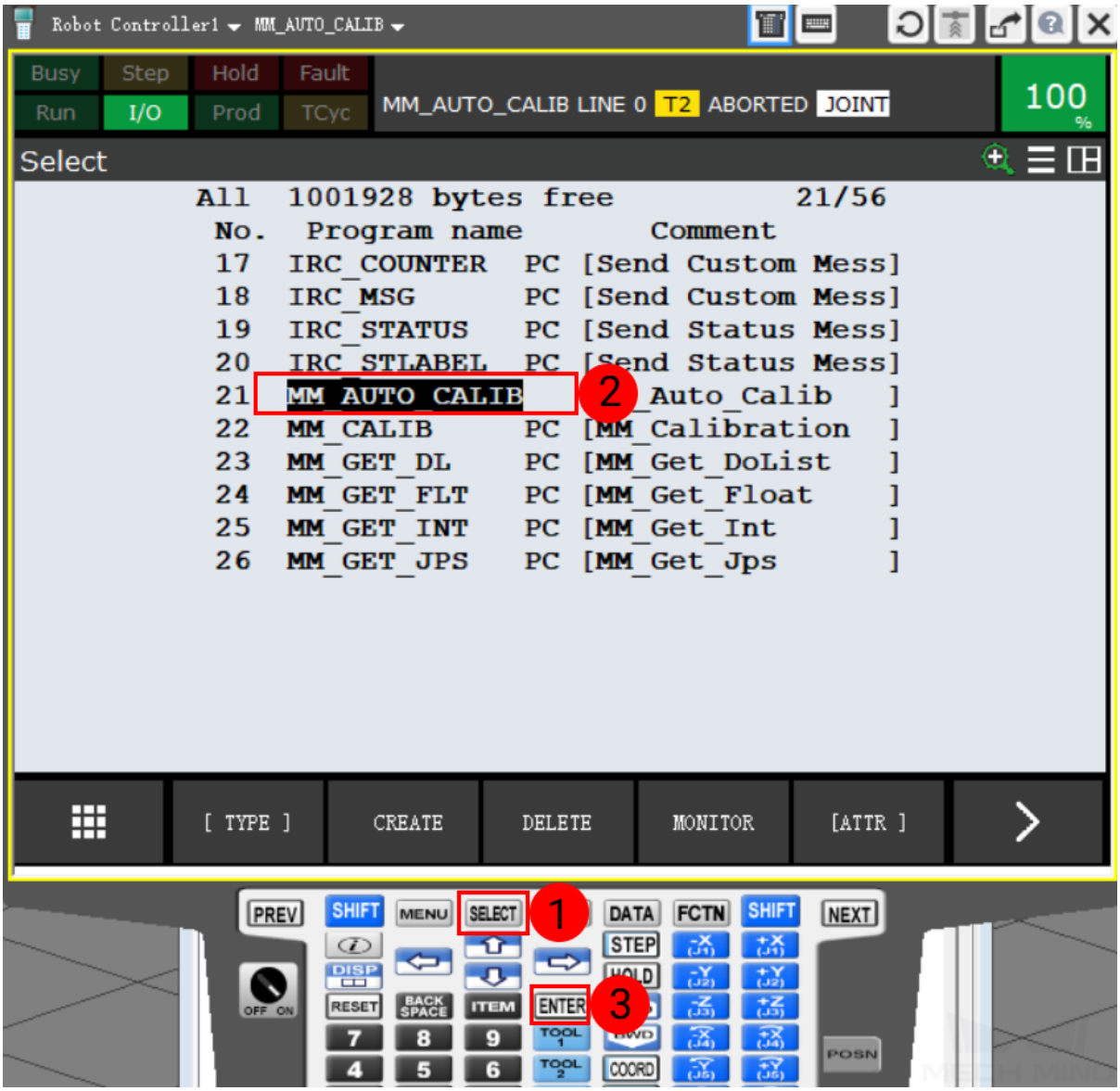
5. Select the corresponding CM file and press `ENTER` key on the teach pendant. Choose *YES* to start loading the programs.

6. When the message "Programs Loaded" is displayed, the program files have been loaded successfully. Press *OK* to exit the program.


Robot Controller1 B
Busy
Step
Hold
Fault
Run
I/O
Prod
TCyc
B LINE 0 T2 ABORTED JOINT
100 %
FILE
UT1:*.*
4/38
"ABORT ALL"
"....."
"Deleting Programs"
"All Programs Deleted"
"....."
"Start Loading V9.* Programs"
"....."
"Programs Loaded"
Execution is completed successfully
OK
4
PREV
SHIFT
MENU
SELECT
EDIT
DATA
FCTN
SHIFT
NEXT
STEP
HOLD
FWD
BWD
COORD
RESET
BACK SPACE
ITEM
ENTER
POSN
OFF ON

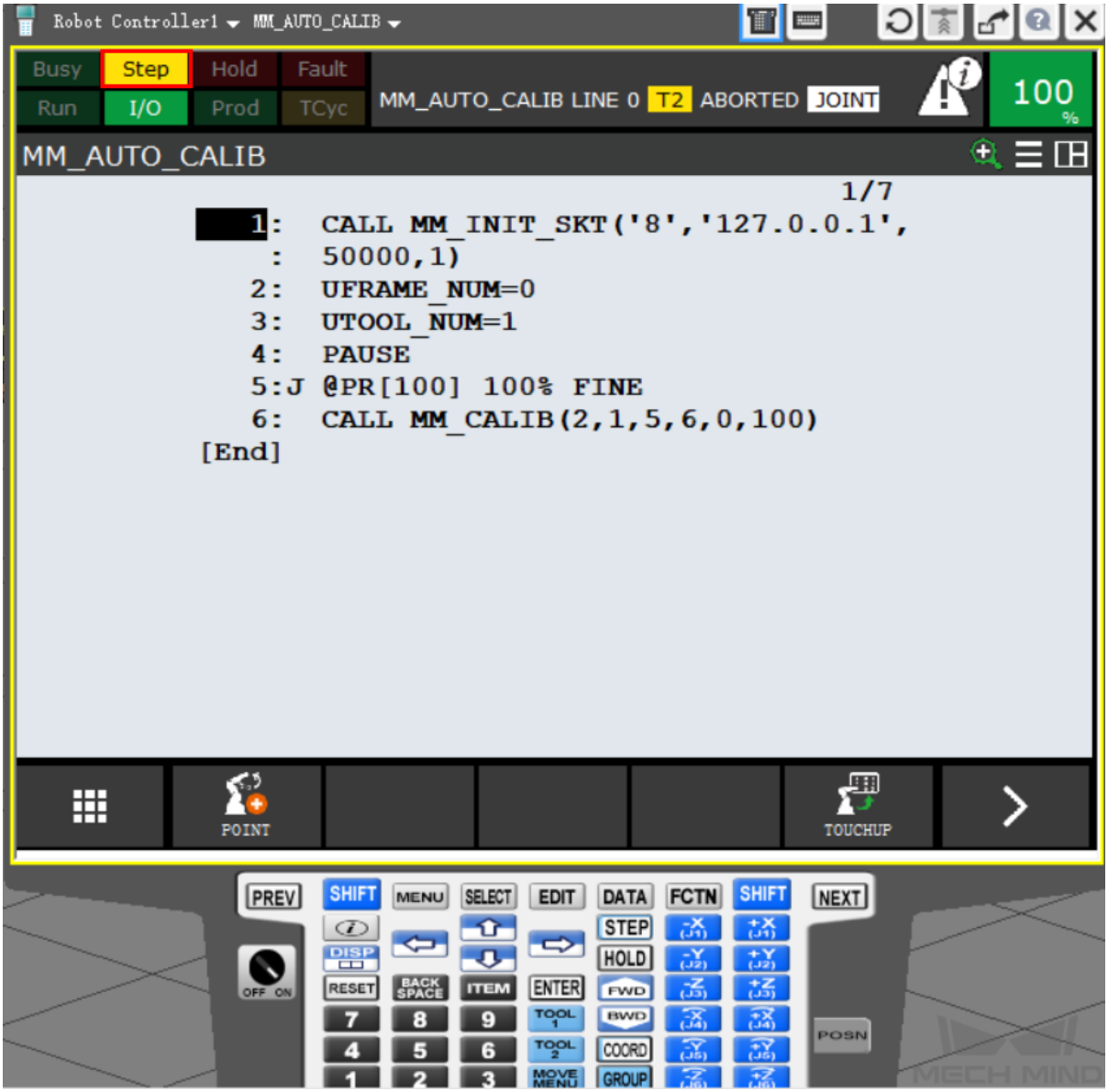
Further Configurations

1. After loading the program files, press `SELECT` key on the teach pendant to enter the program selection interface. Select **MM_AUTO_CALIB** and then press `ENTER` to open the program.



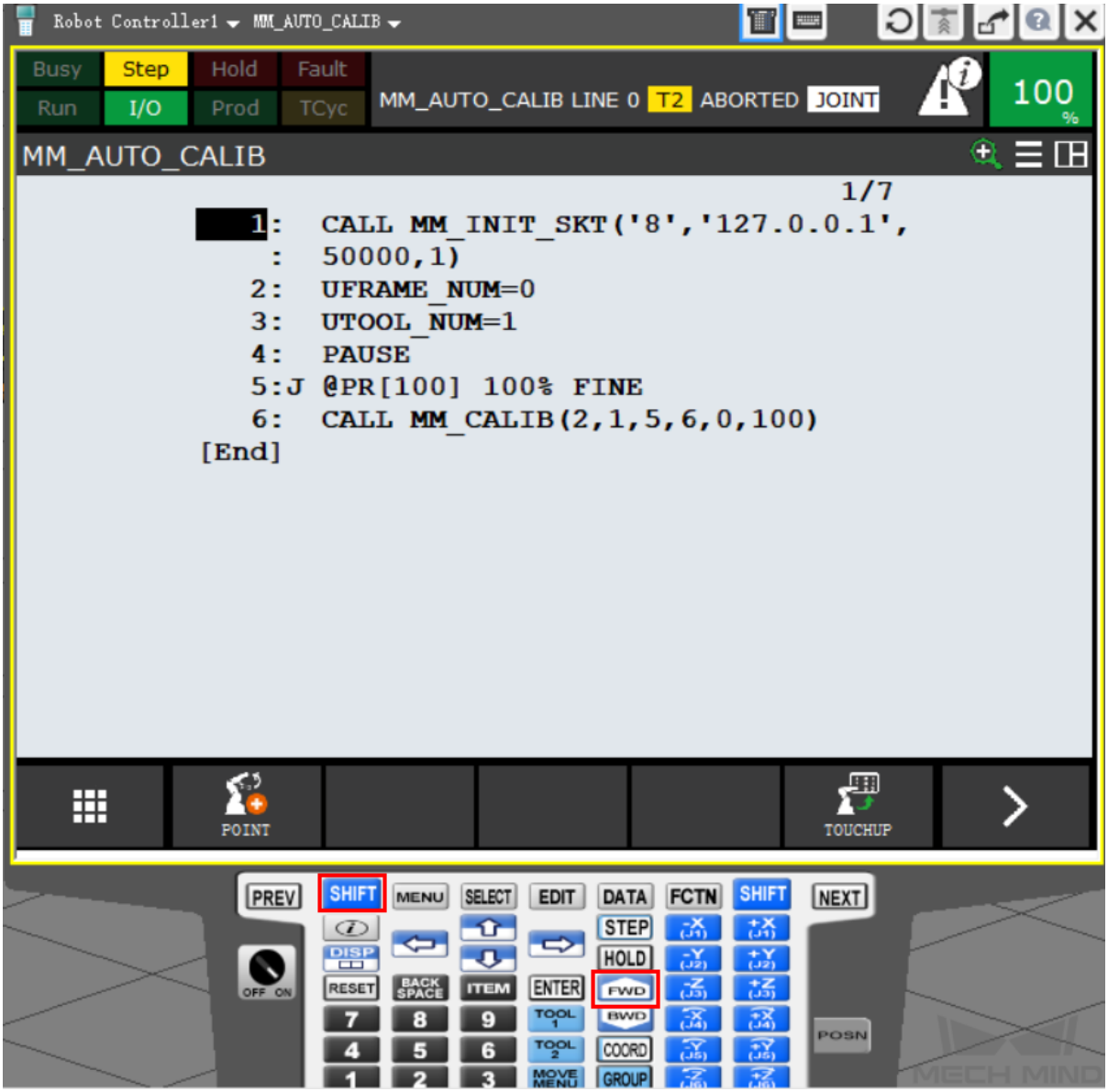
2. Configure the arguments of **MM_INIT_SKT**. There are 4 arguments in total. Please configure them according to your actual situation. Then, press `STEP` key to switch into **Step** mode.
   - Argument 1: client port number (string 1-8)
   - Argument 2: IP address of the IPC
   - Argument 3: server port number of the IPC
   - Argument 4: timeout (min)

3. Now the *Step* icon on the teach pendant turns yellow as shown below.



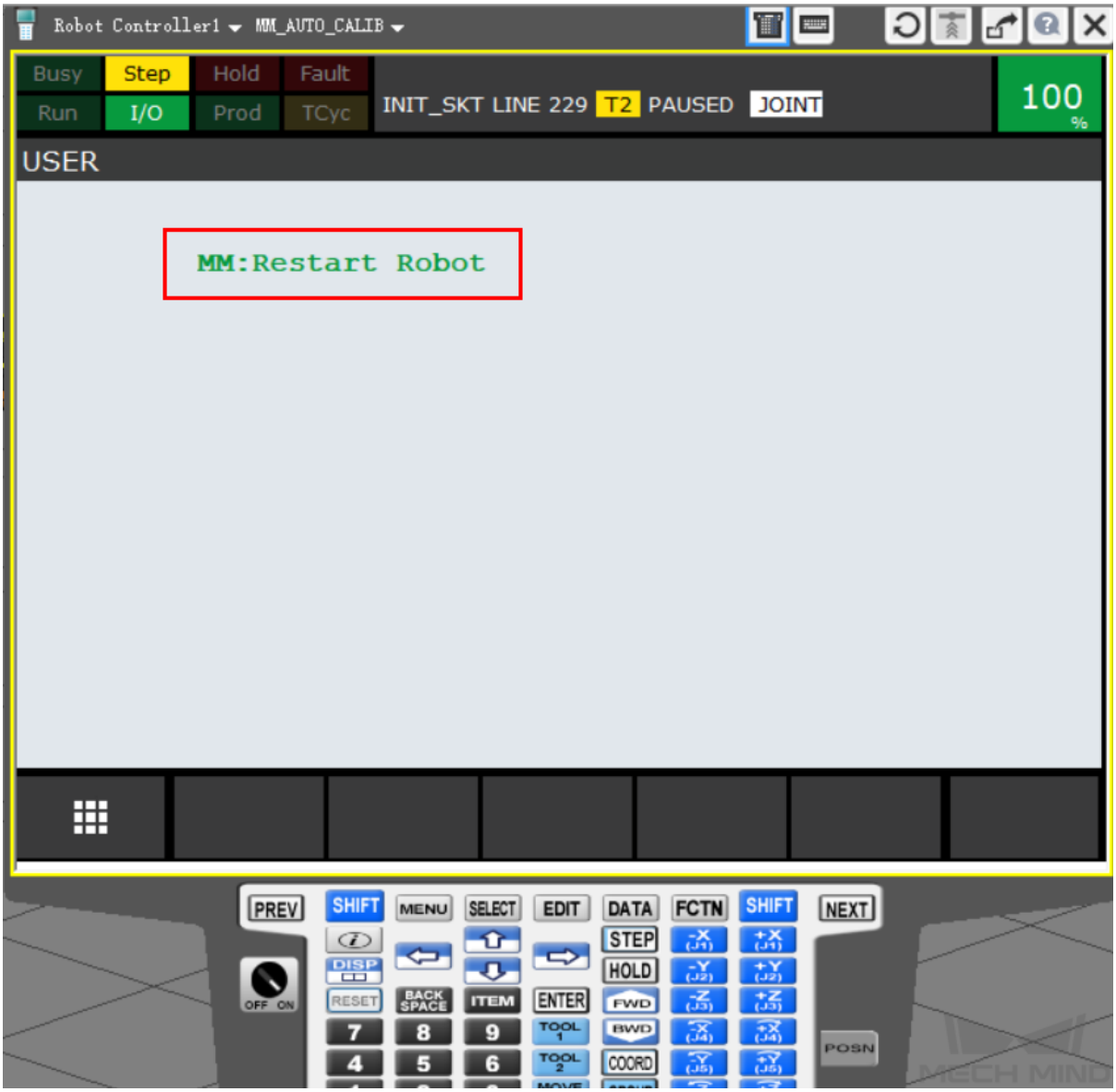
4. Press and hold either one of the deadman switches on the back of the teach pendant.

5. Press the `SHIFT` and `FWD` keys at the same time to run line 1.



6. After running line 1 of the program, press `MENU` and then select *USER*.

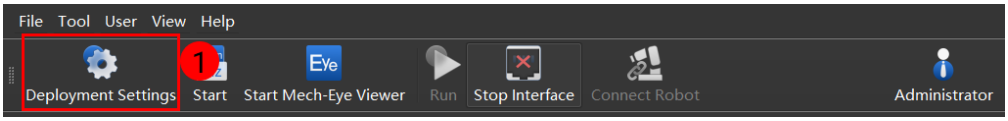
7. If the message "MM:Restart Robot" appear on the screen, the program can be executed successfully on the robot. Please restart the robot later.

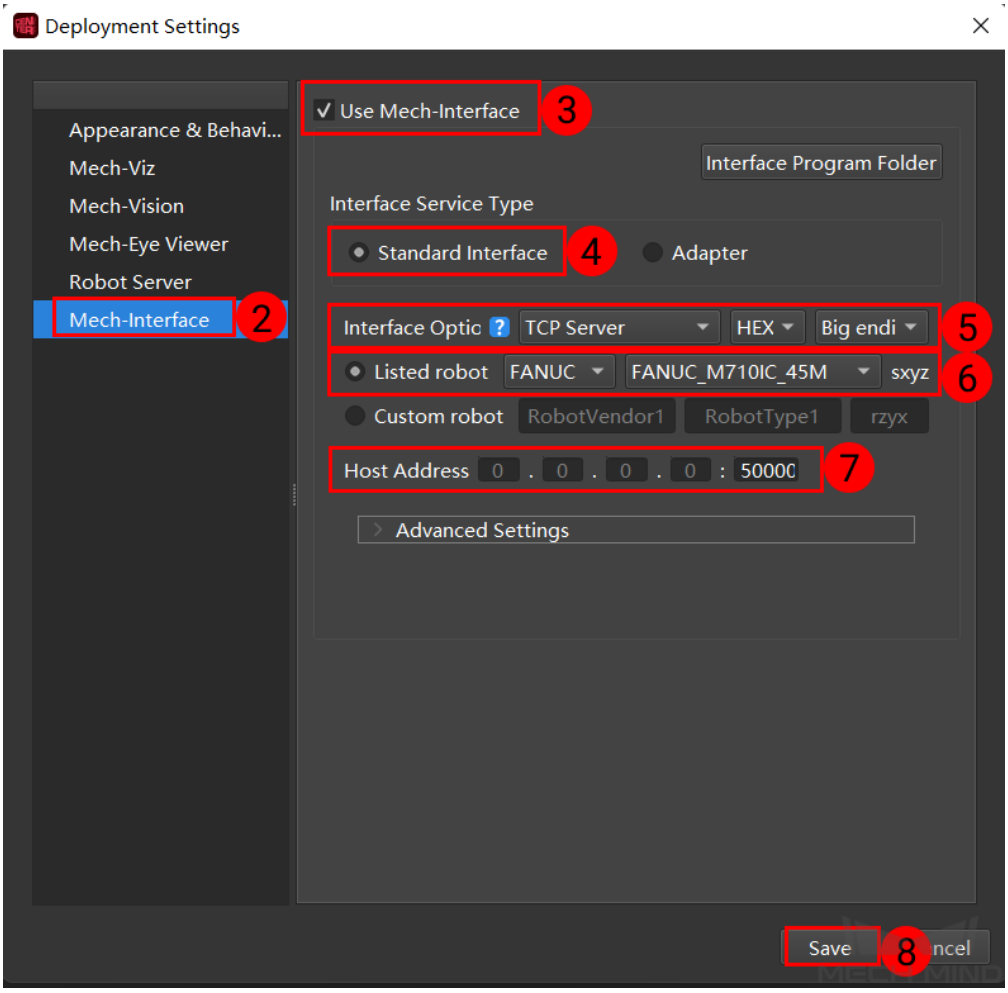


Test Robot Connection

Configuration in Mech-Center

1. Open Mech-Center and click on *Deployment Settings*.
2. Select *Mech-Interface*
3. Check **Use Mech-Interface**.
4. Select **Standard Interface**.
5. Select **TCP Server**, **HEX** and **Big endian** in Interface Options.
6. Select the robot model and click on *Save* to save the configurations.

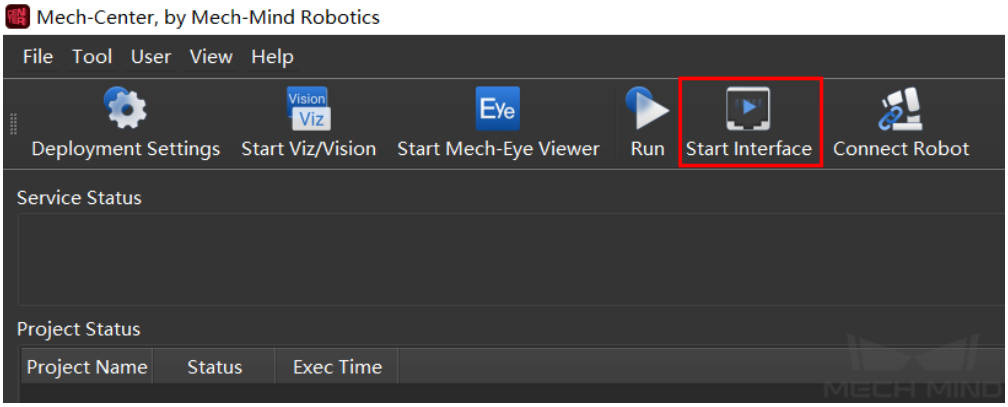




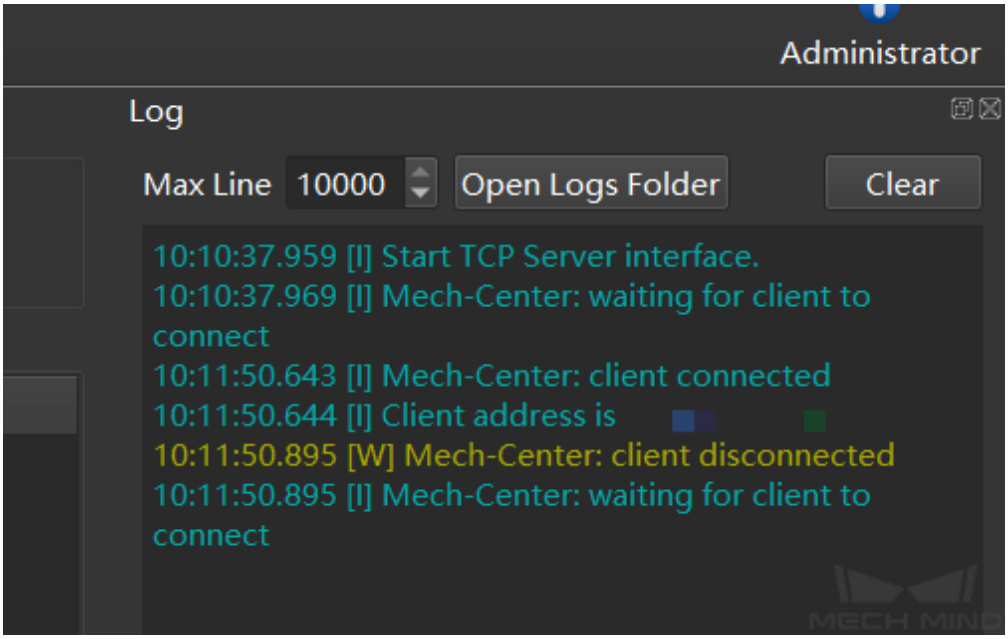
Note: The default port number is 50000. If it is modified, please modify the corresponding code in the robot program when initializing communication.

Test the Connection

1. Start TCP Server interface in Mech-Center.



2. Run the first command of the program **MM_AUTO_CALIB** again according to the instructions mentioned before. If the message as shown below appear in the **Log** panel of Mech-Center, then the robot can be connected successfully.



2.3.2 FANUC Calibration Program

This section introduces the process of calibrating the camera extrinsic parameters using the calibration instruction.

The process consists of 4 steps:

- *Select the Calibration Program*
- *Teach the Calibration Start Point*
- *Run the Calibration Program*
- *Start Calibration in Mech-Vision*

Before proceeding, please make sure that:

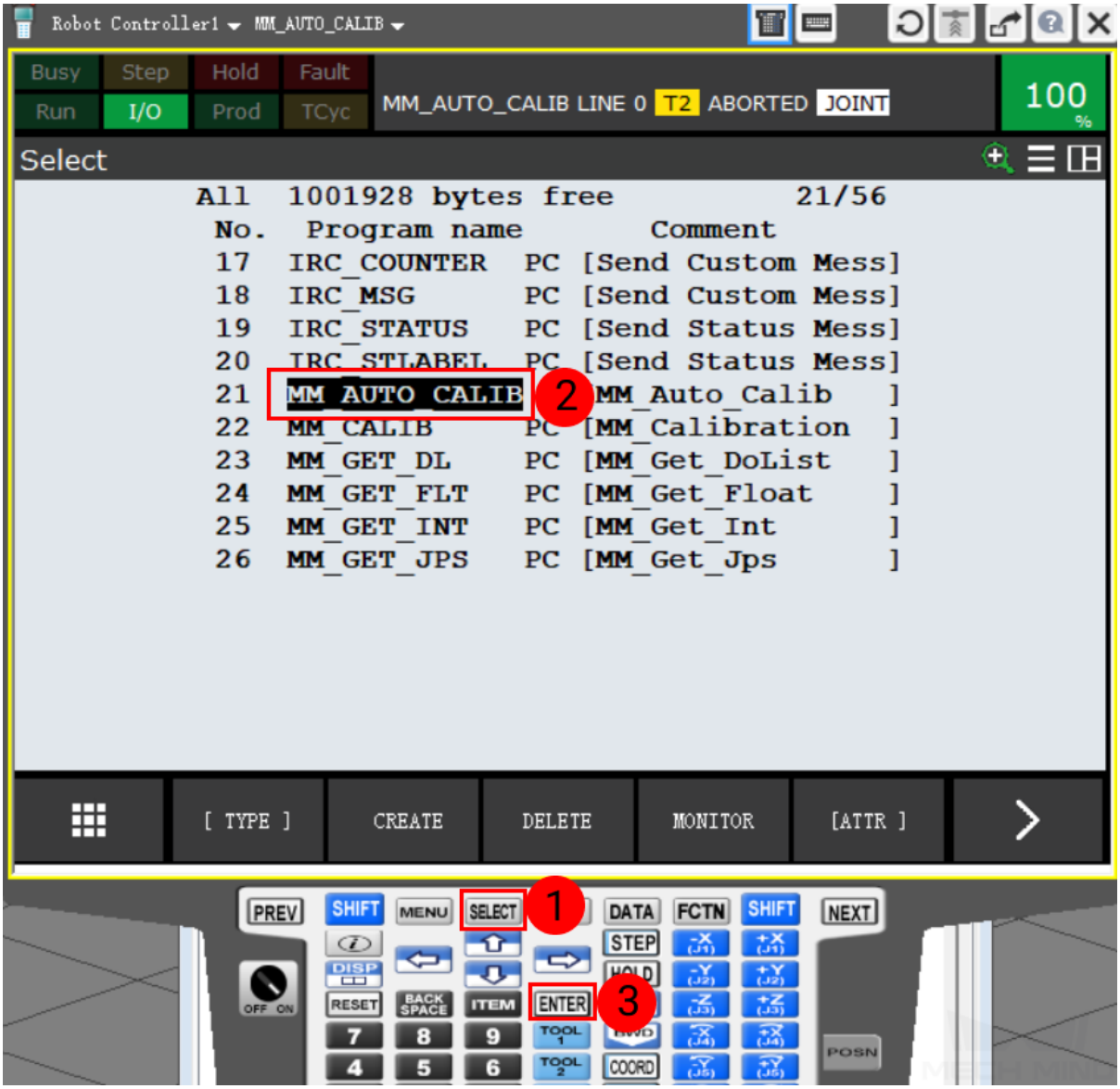
- You have *loaded the Standard Interface program* onto the robot and can establish communcation with Mech-Center.
- You are familiar with the contents in calibration_guide.

The calibration process introduced in this section is applicable to scenarios where standard interface is used to communicate and the extrinsic parameters need to be calibrated multiple times.

Calibration Process

Select the Calibration Program

Press the `SELECT` on the teach pendant, and then select **MM_AUTO_CALIB** and press `ENTER` to open the program.


Robot Controller1
MM_AUTO_CALIB
Busy
Step
Hold
Fault
Run
I/O
Prod
TCyc
MM_AUTO_CALIB LINE 0 T2 ABORTED JOINT
100 %
Select
All 1001928 bytes free 21/56
No. Program name Comment
17 IRC_COUNTER PC [Send Custom Mess]
18 IRC_MSG PC [Send Custom Mess]
19 IRC_STATUS PC [Send Status Mess]
20 IRC_STLABEL PC [Send Status Mess]
21 MM_AUTO_CALIB 2 MM_Auto_Calib]
22 MM_CALIB PC [MM_Calibration]
23 MM_GET_DL PC [MM_Get_DoList]
24 MM_GET_FLT PC [MM_Get_Float]
25 MM_GET_INT PC [MM_Get_Int]
26 MM_GET_JPS PC [MM_Get_Jps]
[TYPE]
CREATE
DELETE
MONITOR
[ATTR]
PREV
SHIFT
MENU
SELECT
1
DATA
FCTN
SHIFT
NEXT
STEP
DISP
RESET
BACK SPACE
ITEM
ENTER
3
TOOL 1
TOOL 2
COORD
OFF ON
POSN

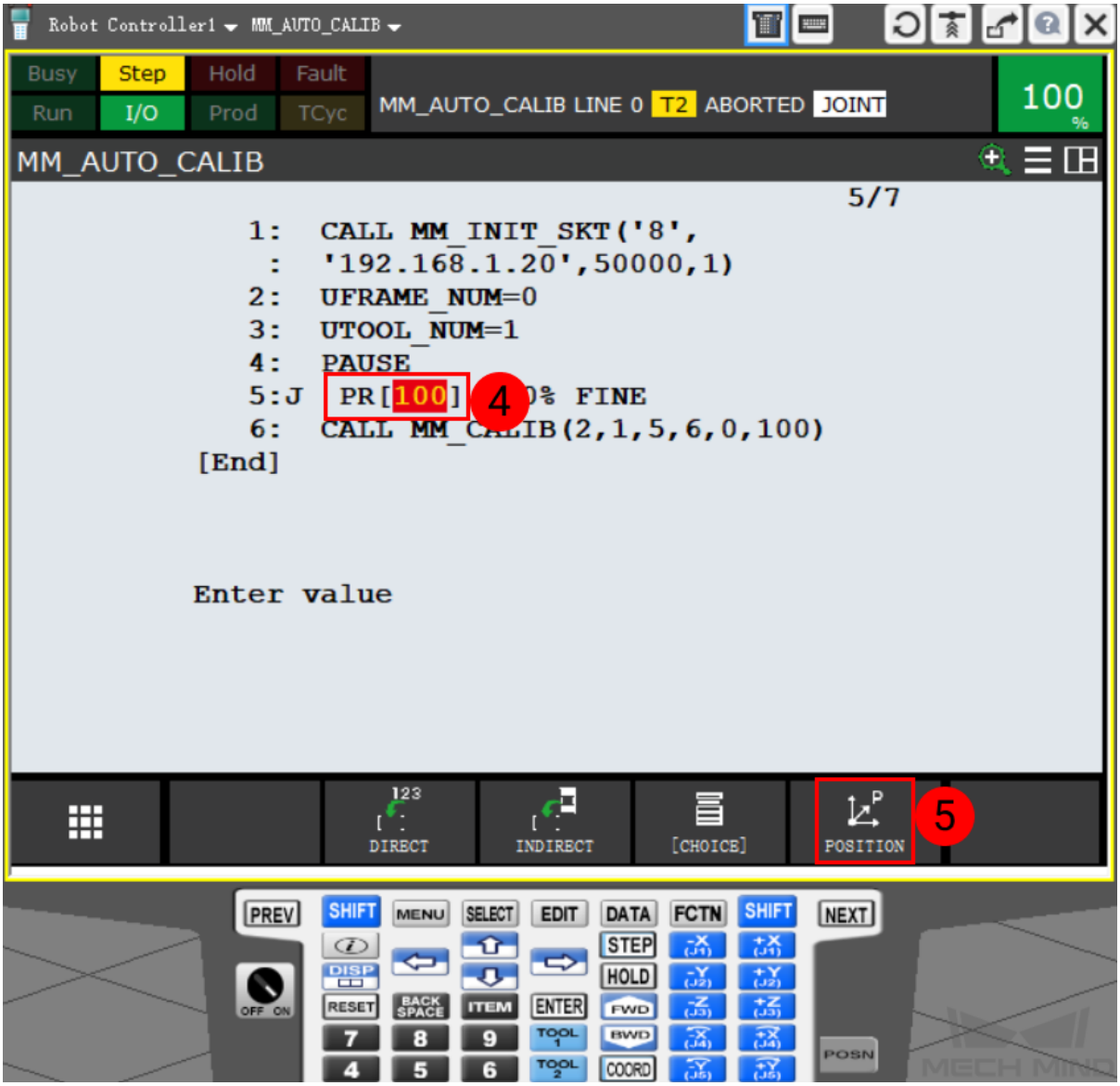
Teach the Calibration Start Point

1. After opening the program, press the `STEP` key on the teach pendant to switch into **Step** mode. Then the *Step* icon on the screen will turn yellow.

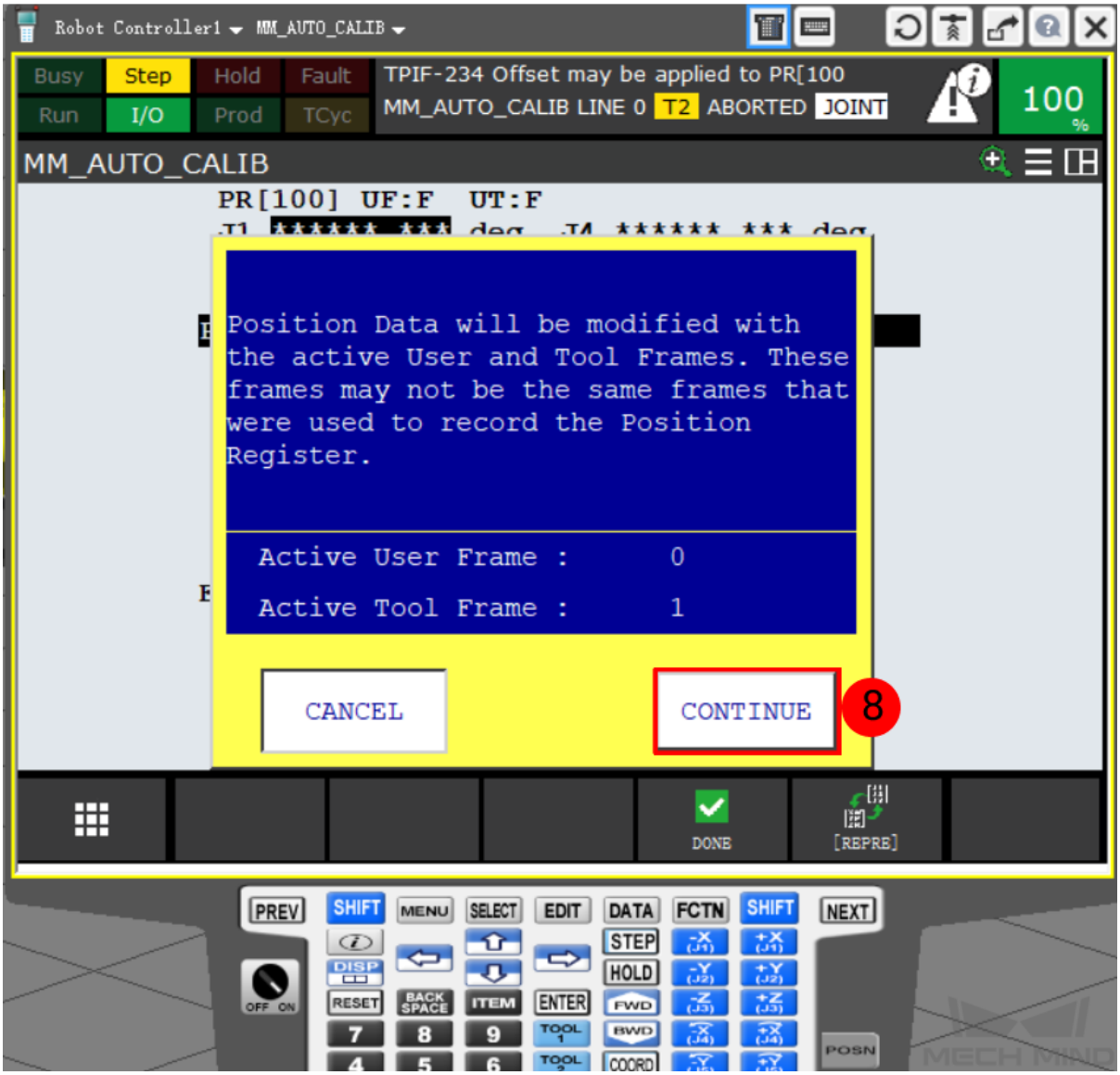
2. Move the cursor to the 5th line. Press and hold either of the deadman switches on the back of the teach pendant, and then press the `SHIFT` and `FWD` keys at the same time to run the command on the 5th line, as shown below.


2

3. Select [**100**], and go to *Position* → *REPRE* → *Cartesian*.

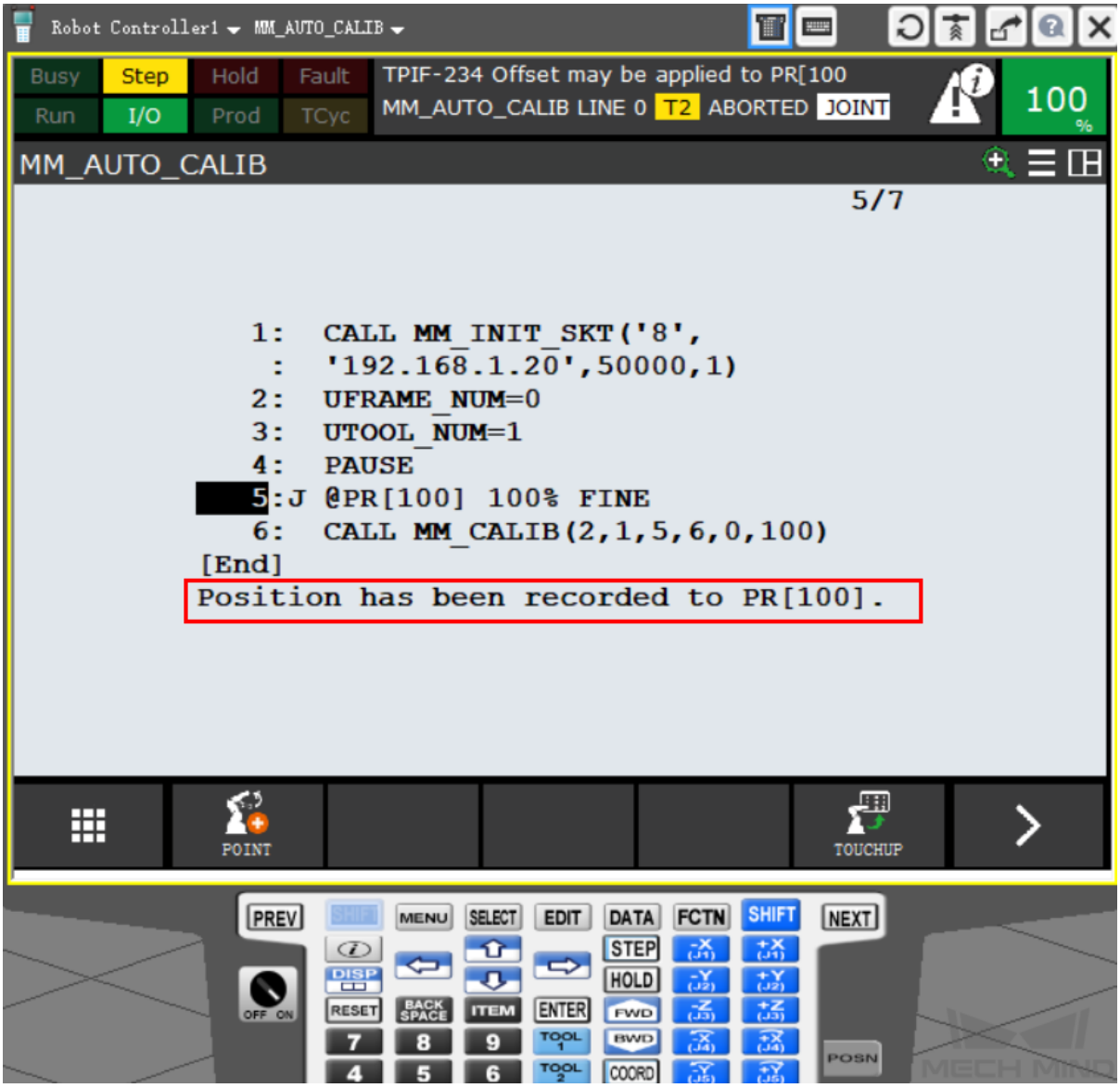

Robot Controller1
MM_AUTO_CALIB
Busy
Step
Hold
Fault
Run
I/O
Prod
TCyc
MM_AUTO_CALIB LINE 0 T2 ABORTED JOINT
100 %
MM_AUTO_CALIB
5/7
1: CALL MM_INIT_SKT('8',
: '192.168.1.20',50000,1)
2: UFRAME_NUM=0
3: UTOOL_NUM=1
4: PAUSE
5:J PR[100] 4 0% FINE
6: CALL MM_CALIB(2,1,5,6,0,100)
[End]
Enter value
DIRECT
INDIRECT
[CHOICE]
POSITION
5
PREV
SHIFT
MENU
SELECT
EDIT
DATA
FCTN
SHIFT
NEXT
STEP
HOLD
FWD
BWD
COORD
RESET
BACK SPACE
ITEM
ENTER
POSN
OFF ON

4. Press on *Continue* in the pop-up window and then press on *DONE*.

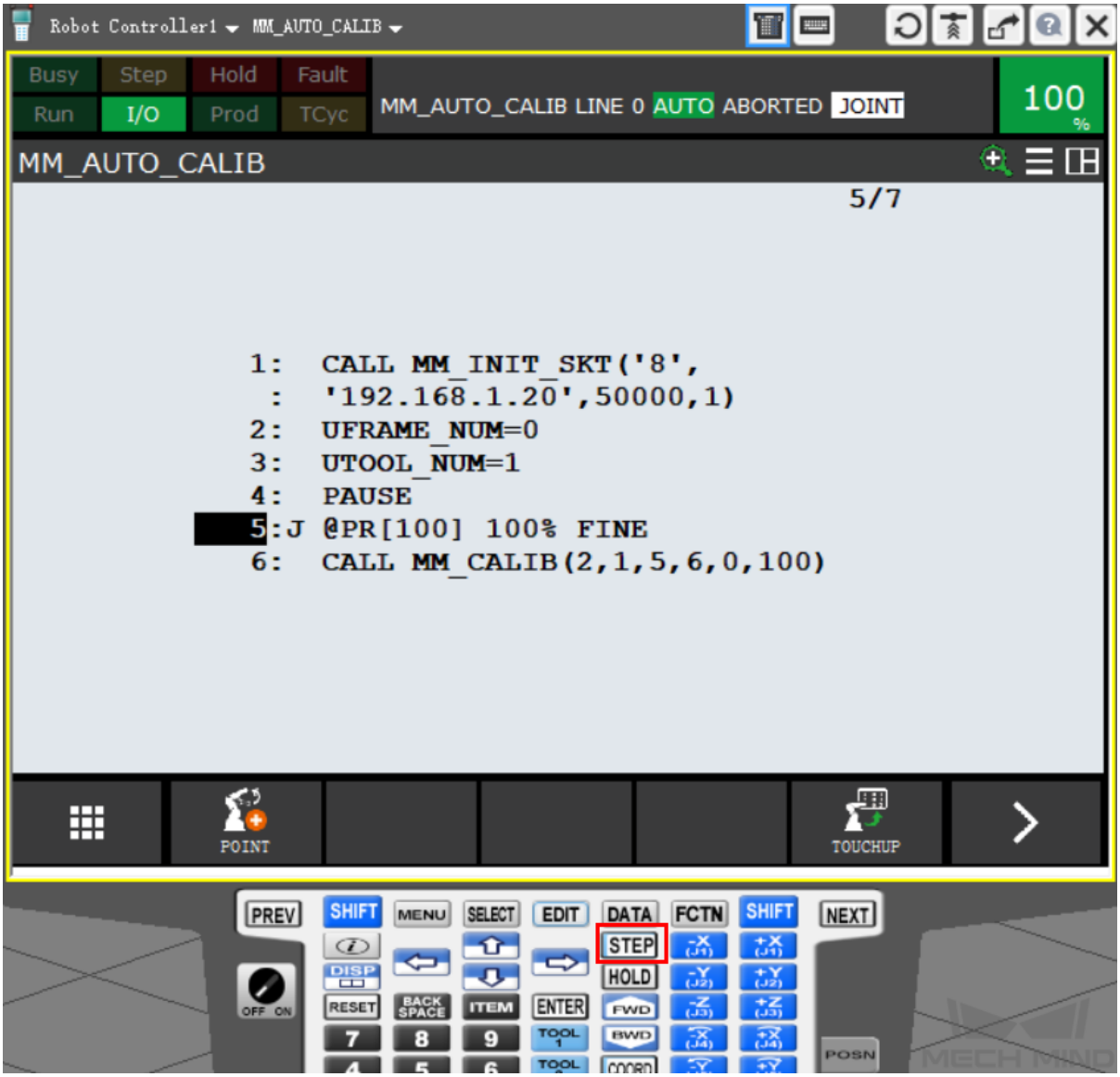

Robot Controller1
MM_AUTO_CALIB
Busy
Step
Hold
Fault
Run
I/O
Prod
TCyc
TPIF-234 Offset may be applied to PR[100
MM_AUTO_CALIB LINE 0 T2 ABORTED JOINT
100
%
MM_AUTO_CALIB
PR[100] UF:F UT:F
Position Data will be modified with the active User and Tool Frames. These frames may not be the same frames that were used to record the Position Register.
Active User Frame : 0
Active Tool Frame : 1
CANCEL
CONTINUE
8
DONE
[REPRE]
PREV
SHIFT
MENU
SELECT
EDIT
DATA
FCTN
SHIFT
NEXT
STEP
HOLD
FWD
BWD
COORD
ENTER
ITEM
BACK SPACE
RESET
DISP
OFF ON
POSN

5. Select the number at the beginning of the 5th line, as shown below. Then press the `SHIFT` key and *TOUCHUP* on the screen together. A message "Position has been recorded to PR[100]" will appear.


Robot Controller1
MM_AUTO_CALIB
Busy
Step
Hold
Fault
Run
I/O
Prod
TCyc
TPIF-234 Offset may be applied to PR[100
MM_AUTO_CALIB LINE 0 T2 ABORTED JOINT
100 %
MM_AUTO_CALIB
5/7
1: CALL MM_INIT_SKT('8',
: '192.168.1.20',50000,1)
2: UFRAME_NUM=0
3: UTOOL_NUM=1
4: PAUSE
5: PR[100] 100% FINE
6: CALL MM_CALIB(2,1,5,6,0,100)
[End]
Position has not been recorded
10
POINT
TOUCHUP
11
PREV
SHIFT
MENU
SELECT
EDIT
DATA
FCTN
SHIFT
11
STEP
HOLD
FWD
BWD
COORD
ENTER
ITEM
RESET
BACK SPACE
DISP
OFF ON
POSN

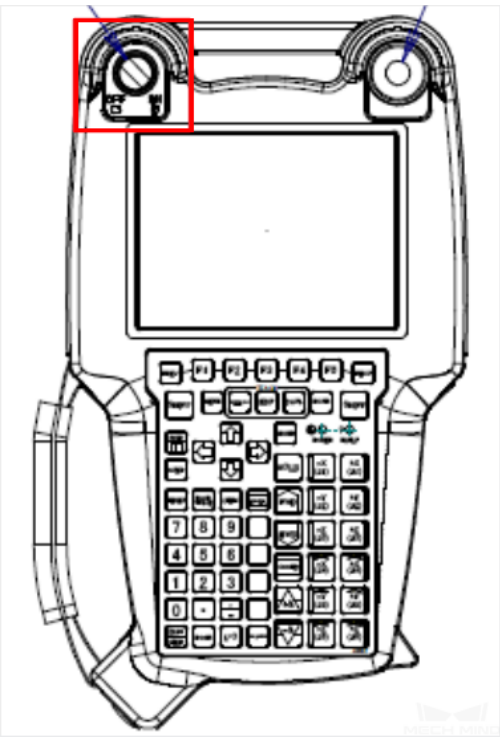


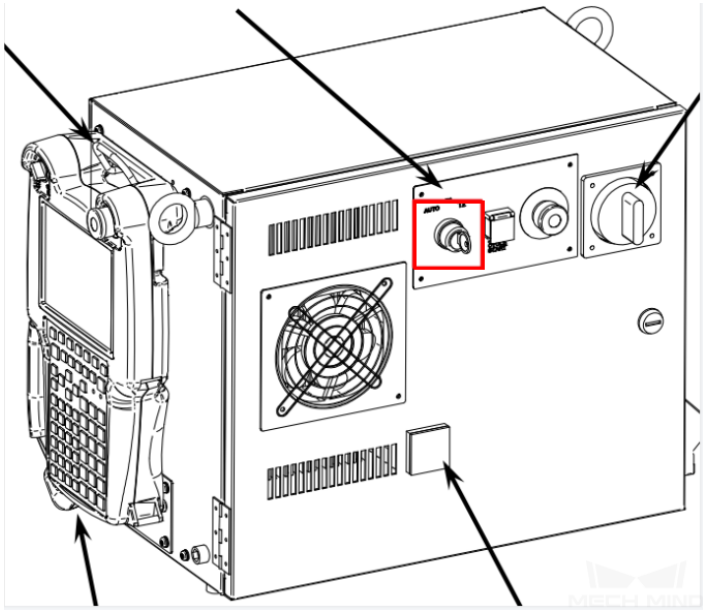
6. Press STEP key on the teach pendant to exit the **Step** mode, then the *Step* icon on the screen will turn grey.


Robot Controller1
MM_AUTO_CALIB
Busy
Step
Hold
Fault
Run
I/O
Prod
TCyc
MM_AUTO_CALIB LINE 0 AUTO ABORTED JOINT
100 %
MM_AUTO_CALIB
5/7
1: CALL MM_INIT_SKT('8',
: '192.168.1.20',50000,1)
2: UFRAME_NUM=0
3: UTOOL_NUM=1
4: PAUSE
5:J @PR[100] 100% FINE
6: CALL MM_CALIB(2,1,5,6,0,100)
POINT
TOUCHUP
PREV
SHIFT
MENU
SELECT
EDIT
DATA
FCTN
SHIFT
NEXT
STEP
HOLD
FWD
BWD
DISP
RESET
BACK SPACE
ITEM
ENTER
OFF ON
POSN

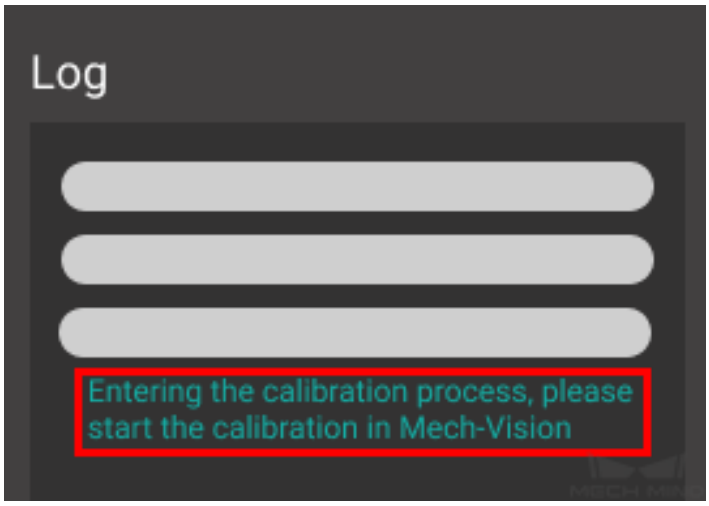
Run the Calibration Program

1. Swith to AUTO mode: turn the switch on the teach pendant to **OFF** and then turn the switch key on the controller to **AUTO**, as shown below.



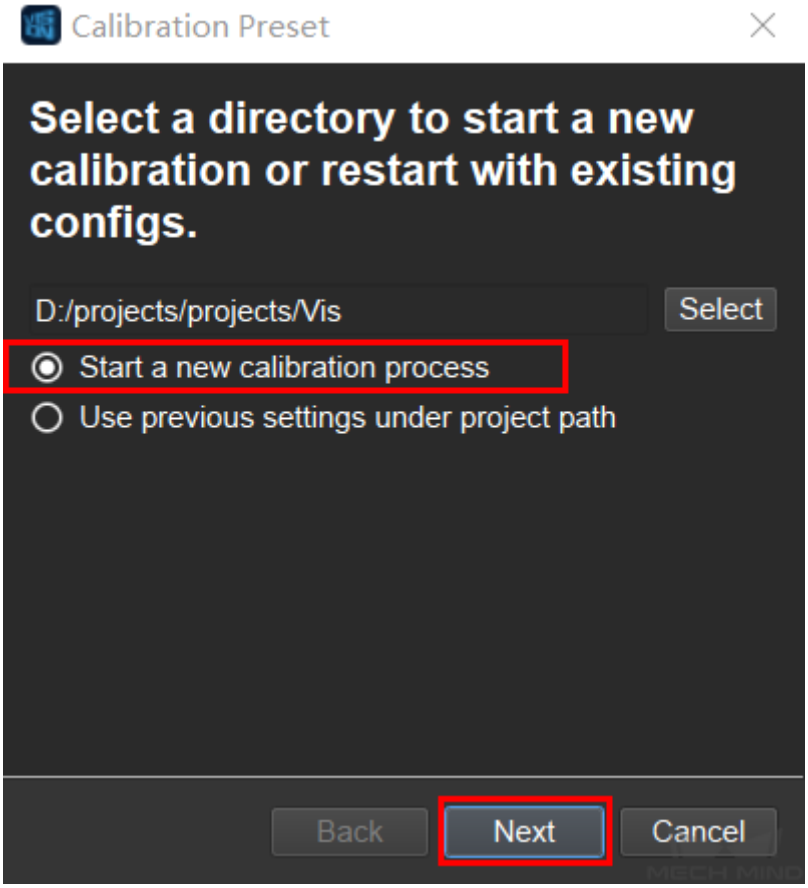


2. Press the green button on the controller to auto-run the calibration program. If the following message appear in Mech-Center **Log** panel, you can start calibration in Mech-Vision.

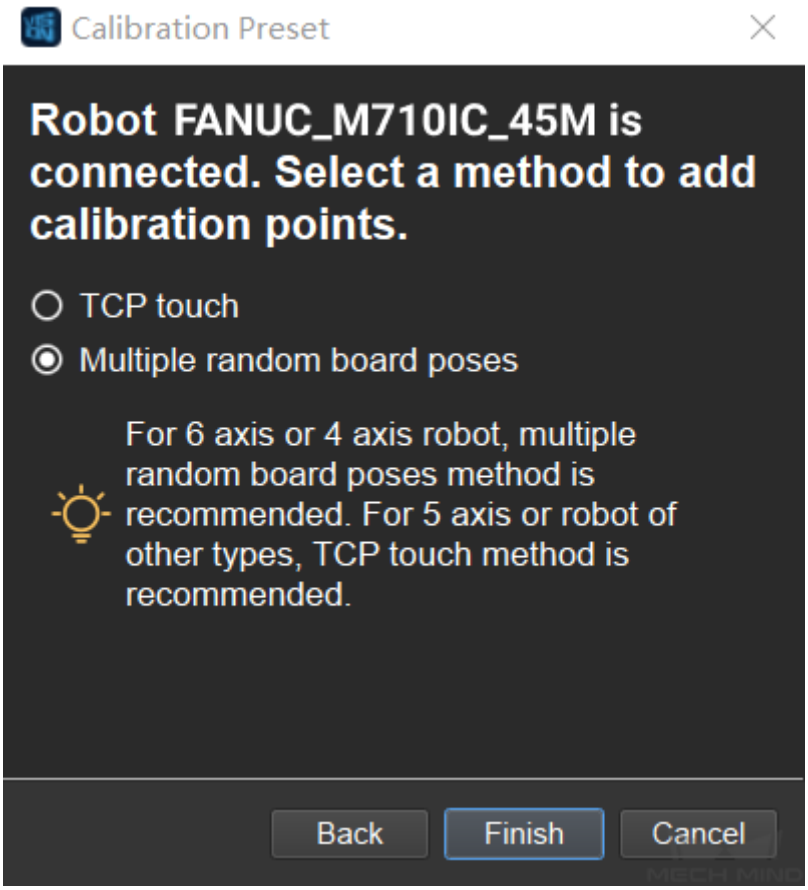


Start Calibration in Mech-Vision

1. In Mech-Vision, go to *Camera* → *Camera Calibration* → *Standard*.
2. Follow the instructions in Mech-Vision to complete the following configuration:
    1. Select **Start a new calibration process**;



2. Select the camera mounting method;
3. Select **Multiple random board poses** for adding calibration points.



Note: If after selecting the camera mounting method, the window says **No robot is connected**, the connection between the robot and Mech-Center is not properly established. Please re-run the robot program.

3. Follow the instructions in Mech-Vision to finish the calibration.

Note: In **5 Add Marker-Images and Poses** after you click on *Move Robot along Trajectory and Add Board Images*, if the robot does not reach the next calibration point within 60 seconds, Mech-Vision will report a timeout error and stop the calibration process. In such case, please select and run **MM_AUTO_CALIB** on the teach pendant again, and restart the calibration process in Mech-Vision.

2.3.3 FANUC Example Program

This section introduces the example program provided with Mech-Center and the operations required to perform an actual pick-and-place task.

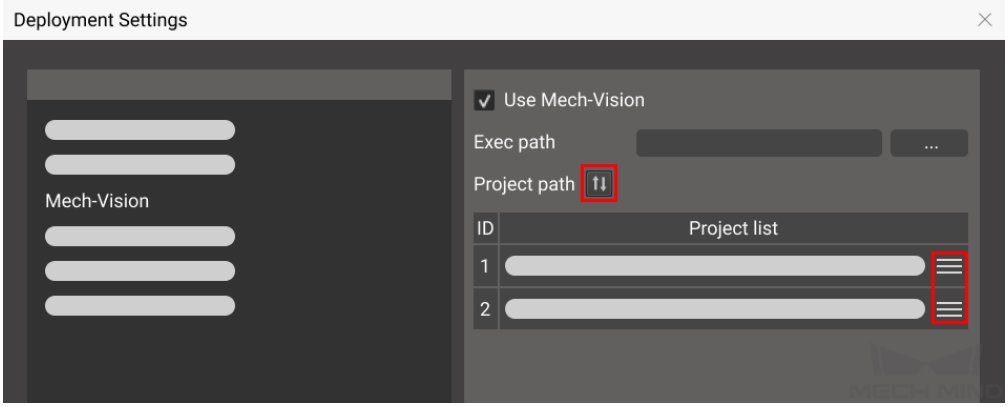
The example program **mm_sample** can be found in *XXXX/Mech-Center/mech-interface/fanuc*.

Check the section corresponding to your own application setup:

- *Obtain Vision Results from Mech-Vision*
- *Obtain Planned Path from Mech-Viz*

Before running the program, please make sure that:

- You have *loaded the Standard Interface program* onto the robot and can establish communcation with Mech-Center.
- You have completed the extrinsic parameter calibration with *the calibration program* or by manually adding calibration points.
- Mech-Vision and Mech-Viz projects are created and set to autoload.
- The **Project list** in *Mech-Center* → *Deployment Settings* → *Mech-Vision* is synced by clicking on  , and the order of Mech-Vision projects have been adjusted according to actual needs.



- The TCP has been correctly specified.
- The robot speed is set to a low value, so that the operator can notice any unexpected behavior before accidents occur.

Obtain Vision Results from Mech-Vision

```
1:  !FUNCTION:Eye to Hand simple pick ;
2:  !MechMind,2022-05-30 ;
3:   ;
4:  !SET Tool ;
5:  UTOOL_NUM=1 ;
6:  !Move to HOME Position ;
7:J P[1] 100% FINE    ;
8:  !Move to Camera capture Position ;
```

(continues on next page)

(continued from previous page)

```
 9:L P[2] 3000mm/sec FINE    ;
10:  !Set IP address and Port ;
11:  CALL MM_INIT_SKT('8','192.168.1.20',50000,1) ;
12:  WAIT    .10(sec) ;
13:  !Set Vision Recipe ;
14:  //CALL MM_SET_MOD(1,1) ;
15:  !Run Vision Project ;
16:  CALL MM_START_VIS(1,1,2) ;
17:  WAIT   1.00(sec) ;
18:  CALL MM_GET_VIS(1,50,51,52) ;
19:  IF (R[52]<>1100) THEN ;
20:  PAUSE ;
21:  ENDIF ;
22:  CALL MM_GET_POS(1,60,70,80) ;
23:L PR[60] 800mm/sec CNT100 Offset,PR[1]    ;
24:L PR[60] 800mm/sec FINE    ;
25:  !Add object grasping logic here ;
26:   ;
27:L PR[60] 800mm/sec CNT100 Offset,PR[1]    ;
28:  !Add transition point ;
29:L P[3] 800mm/sec FINE    ;
30:  !Move to DROP Position ;
31:L P[4] 800mm/sec CNT100 Offset,PR[2]    ;
32:L P[4] 200mm/sec FINE    ;
33:  !Add object releasing logic here ;
34:   ;
35:L P[4] 800mm/sec CNT100 Offset,PR[2]    ;
36:  !Move to HOME Position ;
37:J P[1] 100% FINE    ;
```

Program Logic

1. Move the robot to HOME position.
2. Move the robot to the image capturing pose.
3. Initialize communication with **MM_INIT_SKT**. For detailed information, please refer to *FANUC Standard Interface Commands*.
4. If parameter recipes are used in the Mech-Vision project, the recipe to be used is set with **MM_SET_MOD**. For detailed information, please refer to *FANUC Standard Interface Commands*.
5. Run the Mech-Vision project with **MM_START_VIS**.
6. Wait for 1 second. Under Eye-In-Hand, this **WAIT** instruction is required to make sure the robot stays still until image acquisition is completed. Under Eye-To-Hand, this **WAIT** instruction is not needed if there is a motion procedure between **MM_START_VIS** and **MM_GET_VIS**
7. Obtain the vision results from Mech-Vision.
8. Check if the returned status code indicates any error. If an error code is returned, the program is paused.
9. Move the robot to the picking pose and perform picking.

10. Move the robot to a waypoint between the picking pose and placing pose.
11. Move the robot to the set placing pose and perform placing.
12. Return the robot to HOME position.

The following parts should be modified according to your actual needs.

Define HOME position

Please set the HOME position in register **P[1]**.

Define the TCP

The TCP in this example is defined as **UTOOL_NUM=1**. Please change the value **1** according to the actual TCP values.

Teach the Image Capturing Pose

Please record the image capturing pose in register **P[2]**.

Teach the Waypoint(s)

Waypoints are intermediate poses between the picking pose and placing pose. They are used to ensure that the robot doesn't collide with the surrounding when moving between the picking and placing poses.

The waypoints in this example is saved in **P[3]**. You can add one or more waypoints.

Teach the Placing Pose

Please record the placing pose in register **P[4]**.

Define Z-Offset from Picking/Placing Pose

Z-offset distances relative to the tool frame from the picking/placing pose are used to ensure collision doesn't occur when the robot is approching or departing the picking/placing pose.

Adjust the following commands according to your actual needs.

- **L PR[60] 800mm/sec CNT100 Offset,PR[1]**: the Z-offset when approching the picking pose is saved in **PR[1]**.
- **L PR[60] 800mm/sec CNT100 Offset,PR[1]**: the Z-offset when departing the picking pose is saved in register **PR[1]**.
- **L P[4] 800mm/sec CNT100 Offset,PR[2]**: the Z-offset when approching placing pose is saved in register **PR[2]**.
- **L P[4] 800mm/sec CNT100 Offset,PR[2]**: the Z-offset when departing the placing pose is saved in register **PR[2]**.

Add Object Grasping and Releasing Logics

Add logic for controlling the tool action when picking or placing the object.

Obtain Planned Path from Mech-Viz

```
 1:  !FUNCTION:Eye to Hand simple pick ;
 2:  !and place with Mech-Viz ;
 3:  !MechMind,2022-05-30 ;
 4:   ;
 5:  !SET Tool ;
 6:  UTOOL_NUM=1 ;
 7:  !Move to HOME Position ;
 8:J P[1] 100% FINE    ;
 9:  !Move to Camera capture Position ;
10:L P[2] 3000mm/sec FINE    ;
11:  !Set IP address and Port ;
12:  CALL MM_INIT_SKT('8','192.168.1.20',50000,1) ;
13:  WAIT    .10(sec) ;
14:  !Set Vision Recipe ;
15:  //CALL MM_SET_MOD(1,1) ;
16:  !Run Viz Project ;
17:  CALL MM_START_VIZ(1) ;
18:  WAIT    .10(sec) ;
19:  !set branch exitport ;
20:  //CALL MM_SET_BCH(1,1) ;
21:  !get planned path ;
22:  CALL MM_GET_VIZ(2,50,51,52,53) ;
23:  IF (R[53]<>2100) THEN ;
24:  PAUSE ;
25:  ENDIF ;
26:  CALL MM_GET_POS(1,60,70,80) ;
27:  CALL MM_GET_POS(2,61,71,81) ;
28:  CALL MM_GET_POS(3,62,72,82) ;
29:  !follow the planned path to pick ;
30:L PR[60] R[70]mm/sec FINE    ;
31:L PR[61] R[71]mm/sec FINE    ;
32:  !Add object grasping logic here ;
33:   ;
34:L PR[62] R[72]mm/sec FINE    ;
35:  !Add transition point ;
36:L P[3] 800mm/sec FINE    ;
37:  !Move to DROP Position ;
38:L P[4] 800mm/sec CNT100 Offset,PR[2]    ;
39:L P[4] 200mm/sec FINE    ;
40:  !Add object releasing logic here ;
41:   ;
42:L P[4] 800mm/sec CNT100 Offset,PR[2]    ;
43:  !Move to HOME Position ;
44:J P[1] 100% FINE    ;
```

Program Logic

1. Move the robot to HOME position.
2. Move the robot to the image capturing pose.
3. Initialize communication with **MM_INIT_SKT**. For detailed information, please refer to *FANUC Standard Interface Commands*.
4. If parameter recipes are used in the Mech-Vision project, the recipe to be used is set with **MM_SET_MOD**.
5. Run the Mech-Viz project with **MM_START_VIZ**.
6. Obtain the planned path from Mech-Viz.
7. Check if the returned status code indicates any error. If an error code is returned, the program is paused.
8. Store obtained target points in the planned path to **PR[60]**, **PR[61]**, and **PR[62]**. For detailed information, please refer to *FANUC Standard Interface Commands*.
9. Move the robot along the planned path and perform picking.
10. Move the robot to a waypoint between the picking pose and placing pose.
11. Move the robot to the set placing pose and perform placing.
12. Return the robot to HOME position.

The following parts should be modified according to your actual needs.

Define HOME position

Please set the HOME position in register **P[1]**.

Define the TCP

The TCP in this example is defined as **UTOOL_NUM=1**. Please change the value **1** according to the actual TCP values.

Teach the Image Capturing Pose

Please record the image capturing pose in register **P[2]**.

Teach the Waypoint(s)

Waypoints are intermediate poses between the picking pose and placing pose. They are used to ensure that the robot doesn' t collide with the surrounding when moving between the picking and placing poses.

The waypoints in this example is saved in **P**[**3**]. You can add one or more waypoints.

Teach the Placing Pose

Please record the placing pose in register **P**[**4**].

Add Object Grasping and Releasing Logics

Add logic for controlling the tool action when picking or placing the object.

2.3.4 FANUC Standard Interface Commands

The FANUC Standard Interface provides the following procedures:

- *Initialize Communication*
- *Start Mech-Vision Project*
- *Get Vision Result*
- *Start Mech-Viz Project*
- *Get Planned Path*
- *Obtain Pose*
- *Obtain Joint Positions*
- *Switch Mech-Vision Recipe*
- *Select Mech-Viz Branch*
- *Set Move Index*
- *Get Software Status*
- *Input Object Dimensions to Mech-Vision*
- *Get DO Signal List/ Set DO Signal List*
- *Input TCP to Mech-Viz*
- *Calibration*

When programming the FANUC robot, please pay attention to the following:

- Multiple parameters should be separated by commas.
- All parameters should be defined as local variables in the program file.
- Parameters can be defined as Input or Output parameters.

This Standard Interface is over the TCP/IP protocol.

Initialize Communication

```
MM_INIT_SKT (C_Tag,Ip_Addr,Svr_Port,Time_Out)
```

This procedure is used to set the host IP address, port number, and wait time for TCP/IP communication.

Parameters

- Input Parameters

Name	Description
C_Tag	Client port number, i.e. the port number string of the robot
Ip_Addr	The IP address of the IPC
Svr_Port	Server port number; the default port number is 50000
Time_Out	Wait time in minutes before stopping connection attempt

Example

```
CALL MM_INIT_SKT ('1', '192.168.1.20', 50000, 5);
```

When running the example, the specified client port number is 1, the host IP address should be set to 192.168.1.20, the port number should be set to 50000, and the wait time is 5 minutes.

Start Mech-Vision Project

```
MM_START_VIS (Job,Pos_Num_Need,SendPos_Type)
```

This procedure is for applications that use Mech-Vision but not Mech-Viz.. It runs the corresponding Mech-Vision project to acquire and process data.

Parameters

- Input Parameters

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Please go to *Deployment Settings* → *Mech-Vision* to check and adjust the number.
Pos_Num_Need	Number of poses for Mech-Vision to send, from 0 to 20, where 0 means "send all".
SendPos_Type	Set the image capturing pose for the robot to send, from 0 to 2
	0: Do not send image capturing pose (for Eye To Hand) 1: Send image capturing pose as joint positions 2: Send image capturing pose as robot flange pose

Example

```
CALL MM_START_VIS (1,1,1)
```

This example triggers Mech-Vision No. 1 project to run; the Mech-Vision No. 1 project is expected to send back 1 pose; and the robot will send image capturing pose as joint to Mech-Center.

Get Vision Result

```
MM_GET_VIS (Job,Reg_Lst_Data,Reg_Pos_Num,Reg_Status)
```

This procedure is for applications that use Mech-Vision but not Mech-Viz. It obtains the vision result from the corresponding Mech-Vision project.

Parameters

- Input Parameter

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Please go to *Deployment Settings* → *Mech-Vision* to check and adjust the number.

- Output Parameters

Name	Description
Reg_Lst_Data	Data Register, indicating whether all vision result has been sent, 0 or 1
	0: NOT all vision result has been sent (more on the way) 1: All vision result has been sent
Reg_Pos_Num	Data Register for storing the number of received poses, from 1 to 20
MM_Status	Data Register for storing status code, refer to the standard_interface_status_codes

Example

```
CALL MM_GET_VIS (1,50,51,52)
```

This example obtains the vision result from Mech-Vision project No.1. Whether all vision result has been sent is stored in register **R[50]**, the number of poses received is stored in register **R[51]**, and the status code is stored in register **R[52]**.

Start Mech-Viz Project

```
MM_START_VIZ (SendPos_Type)
```

This procedure is for applications that use both Mech-Vision and Mech-Viz. It runs the corresponding Mech-Viz project (which triggers the corresponding Mech-Vision project to run), and sets the initial joint positions of the simulated robot in Mech-Viz.

Parameter

- Input Parameter

Name	Description
SendPos_Type	The initial joint positions for the simulated robot in Mech-Viz, 0 or 1
	0: Set the initial joint positions of the simulated robot to [0,0,0,0,0,0] 1: Set the initial joint positions of the simulated robot to the current joint positions of the real robot

Note: When the scene contains object models that obstruct the robot to move from [0,0,0,0,0,0] to the first target point, this parameter must be set to 1.

Example

```
CALL MM_START_VIZ (1)
```

This example runs the corresponding Mech-Viz project, and sets the initial joint positions of the simulated robot to the current joint positions of the real robot.

Get Planned Path

```
MM_GET_VIZ (Jps_Pos,Reg_Lst_Data,Reg_Pos_Num,Reg_VPos_Num,Reg_Status)
```

This procedure obtains the planned path from Mech-Viz.

Parameters

- Input Parameter

Name	Description
Jps_Pos	Whether Mech-Viz should send target points as joint positions or TCPs, 1 or 2
	1: Mech-Viz sends joint positions 2: Mech-Viz sends TCPs

- Output Parameters

Name	Description
Reg_Lst_Data	Data Register, indicating whether all target points have been sent, 0 or 1
	0: NOT all target points have been sent (more on the way) 1: All target points have been sent
Reg_Pos_Num	Data Register for storing the number of received target points which range from 1-20
Reg_VPos_Num	Data Register for storing the position of the first visual_move target point in the path
	Example path: move-1, move-2, visual_move-3, move-3, visual_move-2 In this path, the position of the first visual_move target point is 3. If the path does not contain visual_move target point, the return value is 0.
Reg_Status	Data Register for storing status code, refer to the standard_interface_status_codes

Example

```
CALL MM_GET_VIZ (2,50,51,52,53)
```

This example obtains the planned path from Mech-Viz in the form of TCPs. Whether all target points have been sent is stored in register **R**[**50**], the number of target points received is stored in register **R**[**51**], the position of the visual_move target point is stored in register **R**[**52**], and the status code is stored in register **R**[**53**].

Obtain Pose

```
MM_GET_POS (Serial, Pr_Num, Reg_Label, Reg_Speed)
```

This procedure stores a pose returned by Mech-Vision or a target point (as TCP) returned by Mech-Viz in the specified Position Register.

Parameters

- Input Parameter

Name	Description
Serial	Specify the index of the pose to be stored

- Output Parameters

Name	Description
Pr_Num	Position Register for storing the specified pose
Label	Data Register for storing the label corresponding to the specified pose
Pose_Speed	Data Register for storing the speed corresponding to the specified pose

Example

```
CALL MM_GET_POS (1,60,61,62)
```

This example stores the first received pose to register **PR[60]**, the corresponding label to register **R[61]**, and the corresponding speed to register **R[62]**.

Obtain Joint Positions

```
MM_GET_JPS (Serial,Pr_Num,Reg_Label,Reg_Speed)
```

This procedure stores a set of joint positions returned by Mech-Viz in the specified Position Registers.

Note: As Mech-Vision does not output joint position data, this subprogram can only be used with Mech-Viz.

Parameters

- Input Parameter

Name	Description
Serial	Specify the index of the set of joint positions to be stored

- Output Parameters

Name	Description
Pr_Num	Position Register for storing the specified set of joint positions
Reg_Label	Data Register for storing the label corresponding to the specified set of joint positions
Reg_Speed	Data Register for storing the speed corresponding to the specified set of joint positions

Example

```
CALL MM_GET_JPS (1,60,61,62)
```

This example stores the first set of received joint positions to **PR[60]**, the corresponding label to register **R[61]**, and the corresponding speed to register **R[62]**.

Switch Mech-Vision Recipe

```
MM_SET_MOD (Job,Model_Num)
```

This procedure specifies which parameter recipe of the Mech-Vision project to use. For more information on parameter recipe, please see parameter_recipe_configuration.

Note:

- This procedure must be called BEFORE **MM_START_VIZ**.

Parameters

- Input Parameters:

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center* → *Deployment Settings* → *Mech-Vision*
Model_Num	The number of a parameter recipe in the Mech-Vision project, from 1 to 99

Example

```
CALL MM_SET_MOD (2,2)
```

This example switches the parameter recipe used to No. 2 in Mech-Vision project No. 2.

Select Mech-Viz Branch

```
MM_SET_BCH (Branch_Num,Export_Num)
```

This procedure is used to select along which branch the Mech-Viz project should proceed. Such branching is achieved by adding branch_by_service_message Task(s) to the project. This subprogram specifies which out port such Task(s) should take.

Note:

- **MM_START_VIZ** must be called BEFORE this procedure.
- When the next Task to be executed in Mech-Viz is a **branch_by_service_message** Task, Mech-Viz will wait for this procedure to send the out port number it should take.
- The name of all **branch_by_service_message** Tasks in the Mech-Viz project must be changed to numbers between 1 and 99, and the names should be unique among all tasks in the project.

Parameters

- Input Parameters

Name	Description
Branch_Num	Name of the **branch_by_service_message** Task, from 1 to 99
Export_Num	The number of the out port to take, from 1 to 99

Example

```
CALL MM_SET_BCH (1,3)
```

This example tells Mech-Viz to take out port **3** for the **branch_by_service_message** Task named **1**.

Set Move Index

```
MM_SET_IDX (Skill_Num,Index_Num)
```

This procedure sets the value for the Current Index parameter of Mech-Viz Tasks. Tasks that have this parameter include move_list, move_grid, custom_pallet_pattern, and smart_pallet_pattern.

Note:

- **MM_START_VIZ** must be called BEFORE this procedure.
- The name of all Tasks with index parameters in the Mech-Viz project must be changed to numbers between 1 and 99, and the names should be unique among all tasks in the project.

Parameters

- Input Parameters

Name	Description
Skill_Num	Name of the Task, from 1 to 99
Index_Num	Value for the Current Index parameter when the Task is executed

Example

```
CALL MM_SET_IDX (2,10)
```

This example sets the Current Index value to 9 for the Task named **2**. When the Task is executed, the Current Index value will be added 1 and become 10.

Get Software Status

```
MM_GET_STAT (MM_Status:OUT)
```

This procedure is currently capable of checking whether Mech-Vision is ready to run projects. In the future, this procedure can be used for obtaining the execution status of Mech-Vision, Mech-Viz and Mech-Center.

Parameter

- Output Parameter

Name	Description
MM_Status	Data Register for storing the status code, refer to the standard_interface_status_codes

Example

```
CALL MM_GET_STAT (70)
```

This example obtains the status code and stores it in register **R[70]**.

Input Object Dimensions to Mech-Vision

```
MM_SET_BS (Job:IN,Lenght,Width,Height)
```

This procedure inputs object dimensions to the Mech-Vision project.

Note:

- This procedure must be called BEFORE **MM_START_VIS**.

Parameters

- Input Parameters

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center* → *Deployment Settings* → *Mech-Vision*
Lenght	Length of object in mm
Width	Width of object in mm
Height	Height of object in mm

Example

```
CALL MM_SET_BS (1,11.0,12.0,13.0)
```

This example sets the object dimensions in the read_object_dimensions Step in the Mech-Vision project No. 1 to 11*12*13 mm.

Get DO Signal List/ Set DO Signal List

- Get DO Signal List

  ```
  MM_GET_DL
  ```

- Set DO Signal Lis

  ```
  MM_SET_DL
  ```

These two procedures obtain the planned DO Signal list for controlling multiple sections of a sectioned vacuum gripper and set the DOs accordingly.

Note:

- **MM_GET_VIZ** must be called BEFORE this procedure.
- Please deploy the Mech-Viz project based on the template project in *Mech-Center/tool/viz_project/suction_zone*, and set the suction cup configuration file in the Mech-Viz project.

Parameters

No parameters.

Example

```
CALL MM_GET_DL
```

```
CALL MM_SET_DL
```

These two examples obtain the DO signal list planned by Mech-Viz, store it in **Do_Port[i]**, and input the values to the corresponding digital outputs.

Input TCP to Mech-Viz

```
MM_SET_POS (X,Y,Z,W,P,R)
```

This procedure inputs TCP data to the outer_move Task.

Note:

- This procedure must be called BEFORE **MM_START_VIZ**.
- Please deploy the Mech-Viz project based on the template project in *Mech-Centertoolviz_projectouter_move*, and put the **outer_move** Task at a proper position in the workflow.

Parameter

- Input parameter

Name	Description
X	Variable 1 of the pose (mm)
Y	Variable 2 of the pose (mm)
Z	Variable 3 of the pose (mm)
W	Variable 4 of the pose (arc degree)
P	Variable 5 of the pose (arc degree)
R	Variable 6 of the pose (arc degree)

Example

```
CALL MM_SET_POS (100.1,200.2,300.3,90.0,180.0,0.0)
```

This example sends the pose of TCP **(100.1,200.2,300.3,90.0,180.0,0.0)** to the **outer_move** task in the Mech-Viz project.

Calibration

```
MM_CALIB (Move_Type,PosJps,WaitTime,AxisNum,AxisVal,Reg_CalibPos)
```

This procedure is used for hand-eye calibration (camera extrinsic parameter calibration). It automates the calibration process in conjunction with the **Camera Calibration** function in Mech-Vision. For detailed instructions, see *FANUC Calibration Program*.

Parameters

- Input Parameters

Name	Description
Move_Type	Motion type, 1 or 2
	1: Linear motion 2: Joint moton
PosJps	Pose as flange pose or joint positions, 1 or 2
	1: flange pose 2: Joint positions
WaitTime	Wait time between poses in minutes
AxisNum	The axis number of the robot
AxisVal	Data of the external 7th axis in mm (Optional; input 0 when there is no external axis)
Reg_CalibPos	The Position Register used in MM_AUTO_CALIB; PR[100] by default

Example

```
CALL MM_CALIB (2,1,5,6,0,100)
```

This example moves a 6-axis robot in Joint motion type, receives pose data in the form of flange pose, and sets the wait time between two poses to 5 minutes. Moreover, the robot does not have an external 7th axis. The register **PR[100]** is used to store the received pose data.

2.3.5 FANUC Error Messages

The following errors may occur while running the Standard Interface program on the robot.

MM:Robot_Internal_Error

Error occurred while the robot program attempts to call the system function.

Troubleshooting

Please refer to *KAREL Reference Manual* and check the program.

MM:Robot_Socket_Closed

Error occurred when the robot program called the **MSG_CONNECT** procedure to establish communication with Mech-Center.

Troubleshooting

- Check if the hardware are properly connected.
- Check if the Standard Interface is started in Mech-Center.
- Check the IP addresses of the robot and the IPC, and if the port number is configured correctly.
- Check if the firewall is turned off on the IPC.
- Contact Mech-Mind Technical Support for further assistance.

MM:Robot_Argument_Error

When calling a Mech-Mind Standard Interface procedure, arguments provided are not correct.

Troubleshooting

Please refer to *FANUC Standard Interface Commands* and input the correct arguments accordingly.

MM:Robot_CMD_Error

The command code returned to the robot does not match the one sent to Mech-Center.

Troubleshooting

The sequence of command sending and receiving is problematic. Please contact Mech-Mind Technical Support for further assistance.

MM:IPC_Return_Error

Returned status code is an error code. Please check Mech-Center' s log.

Troubleshooting

- Please refer to the standard_interface_status_codes for the specific error.
- Please contact Mech-Mind Technical Support for further assistance.

2.4 KUKA

This section introduces the Standard Interface for KUKA robots.

2.4.1 KUKA Setup Instructions

This section introduces the process of loading the Standard Interface program onto a KUKA robot.

The process consists of 4 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program Files*
- *Test Robot Connection*

Please have a flash drive ready at hand.

Check Controller and Software Compatibility

Compatibility requirements are as follows:

- Robot: 6-axis KUKA robot
- Controller model: KUKA KR C4
- Controller system software version: KSS 8.2 to 8.6
- Add-on software package: Ethernet KRL (V 2.2.8 or above)

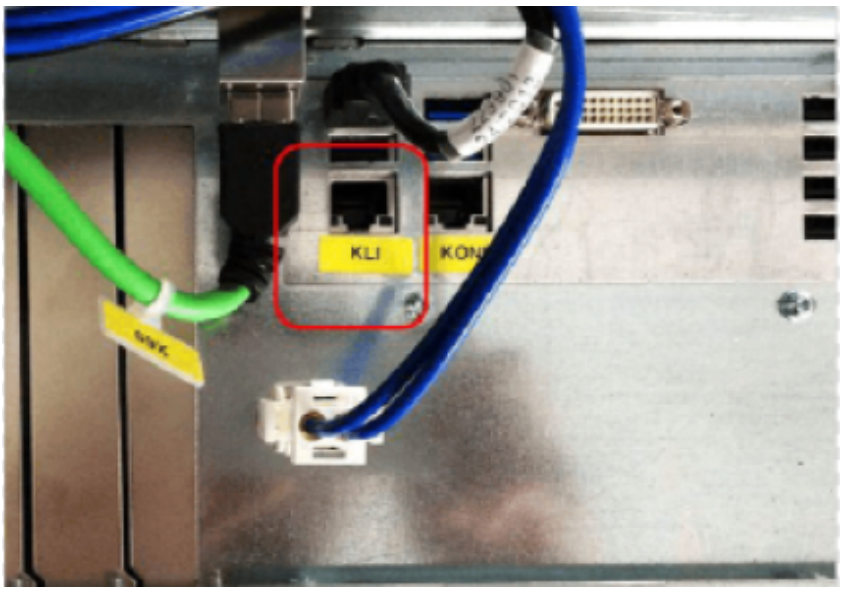
Note: All teach pendant actions in this chapter are performed on KSS 8.6. The specific steps and menu selections may differ slightly in older versions of system software.

Setup the Network Connection

Hardware Connection

Plug the Ethernet cable into:

- An Ethernet port on the IPC
- The X66 port on KR C4 compact and KLI port on other KR C4 controllers

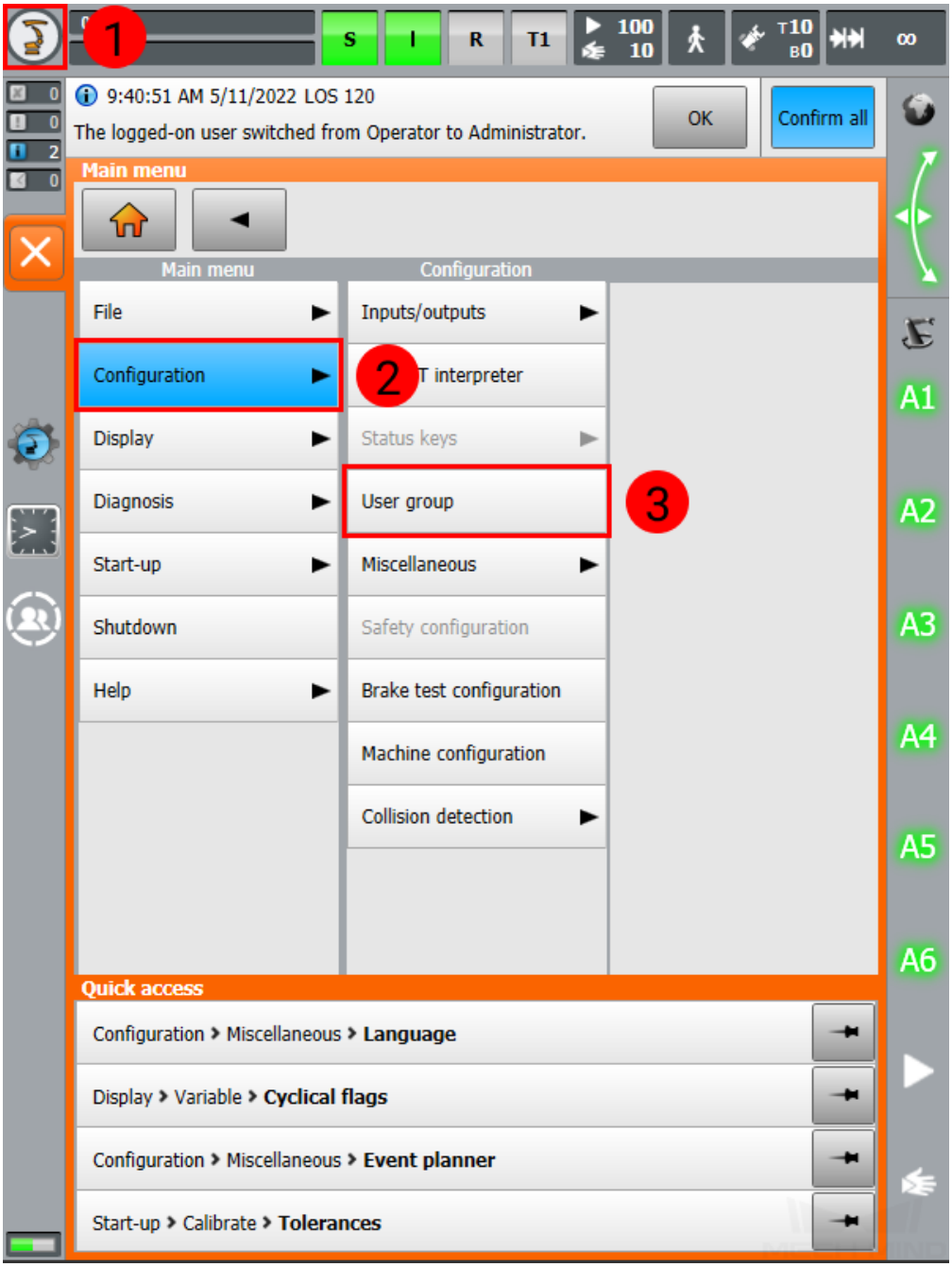


IP Configuration

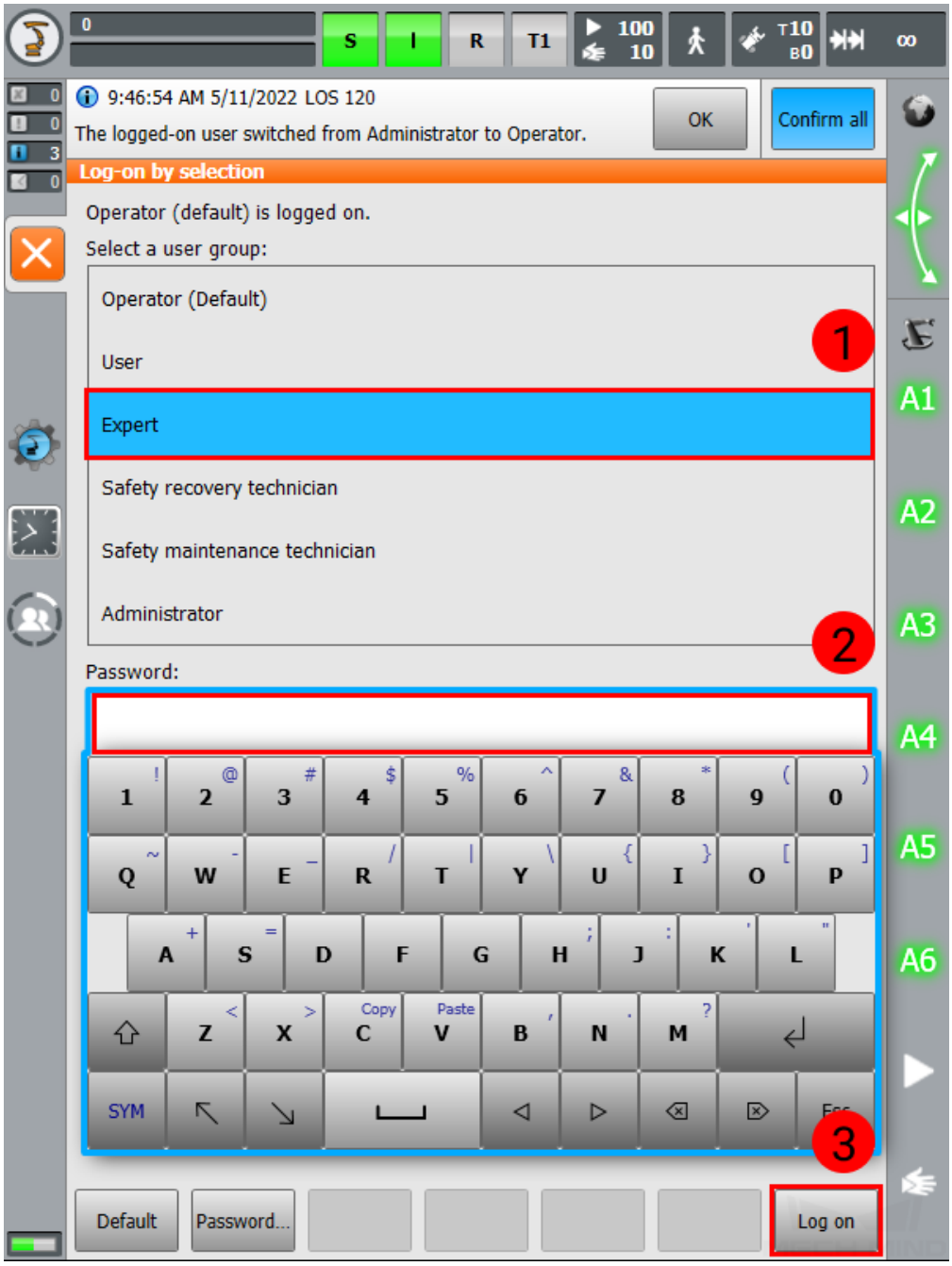
To allow communication between the IPC and the robot controller, both must have an IP address in the same subnet. This means that the first three numbers of the IP addresses should be the same. For example, 192.168.100.1 and 192.168.100.2 are in the same subnet.

1. Check the IP address of the IPC: please use the *ipconfig* command in Command Prompt or PowerShell to check the IP address.
2. Switch to expert mode:

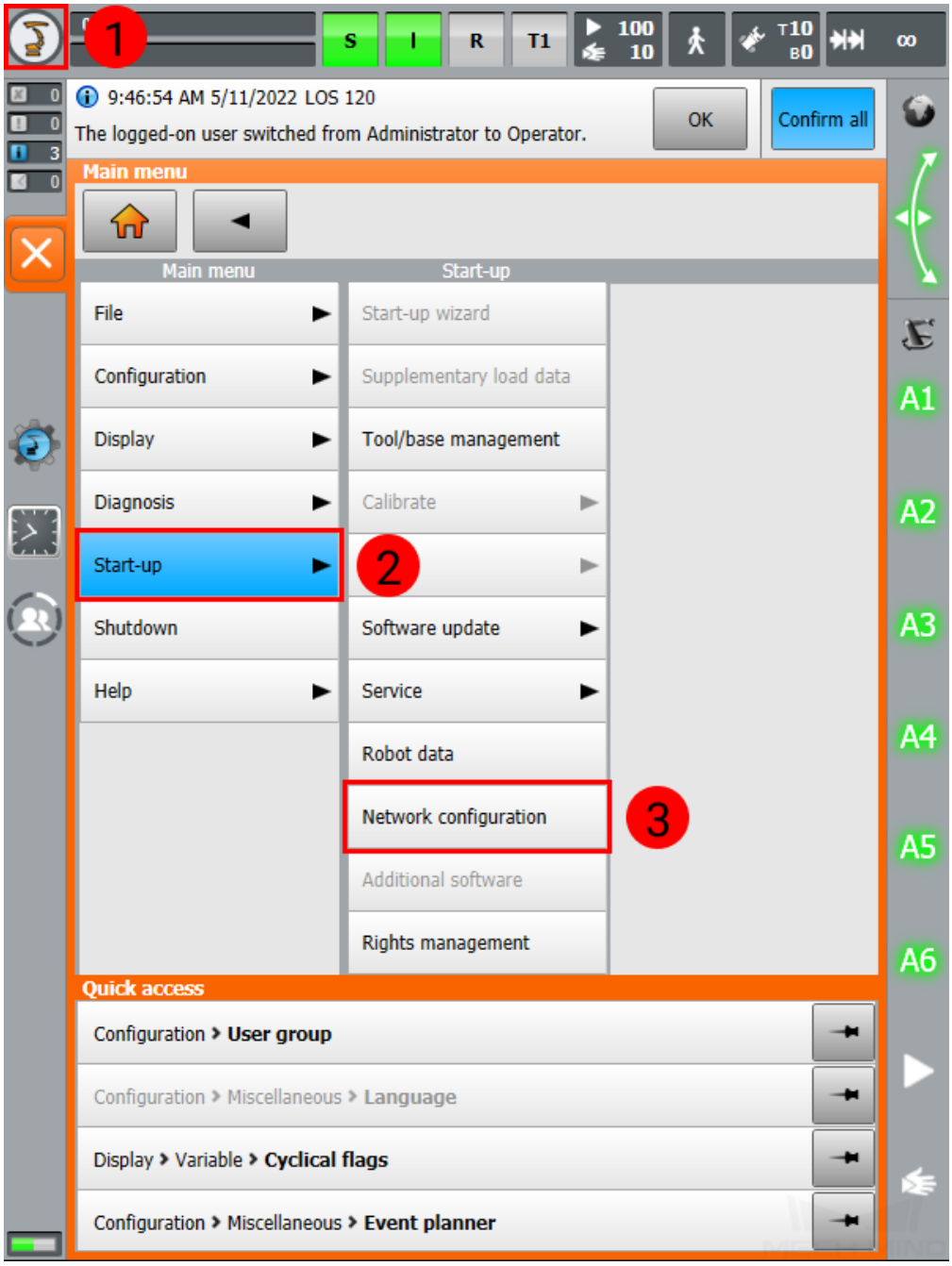
1. Press on  , and then select *Configuration* → *User group*.



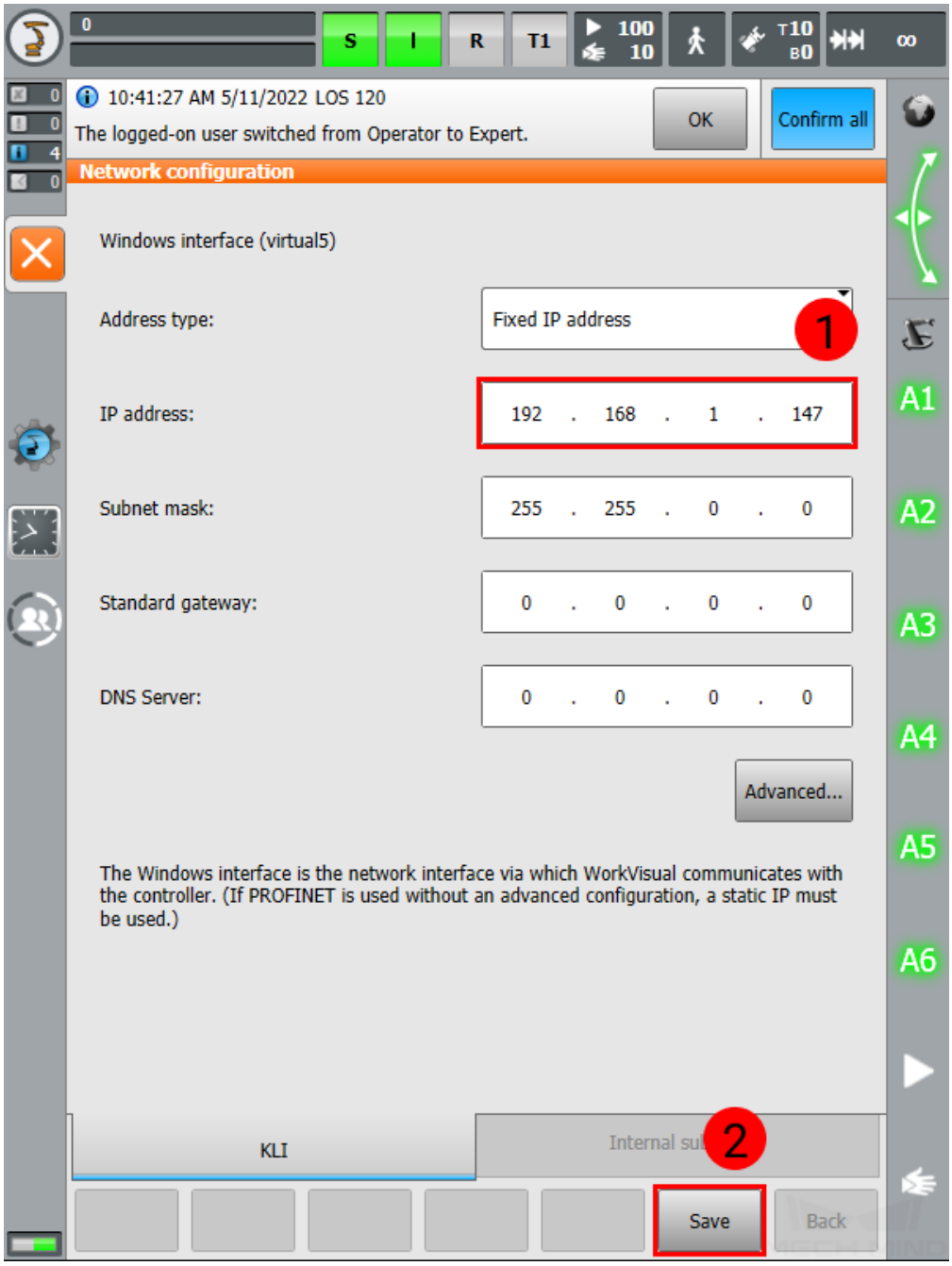
2. Select **Expert**, enter the password (the default password is **kuka**), and press on *Log on*.

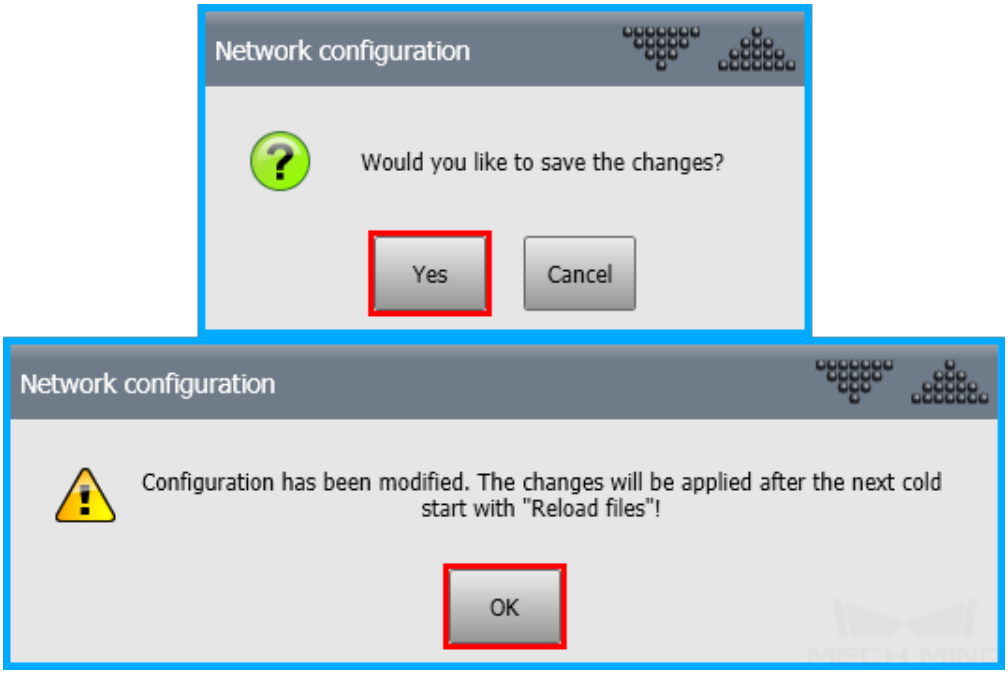


3. Press on  , and then select *Start-up* → *Network configuration*.



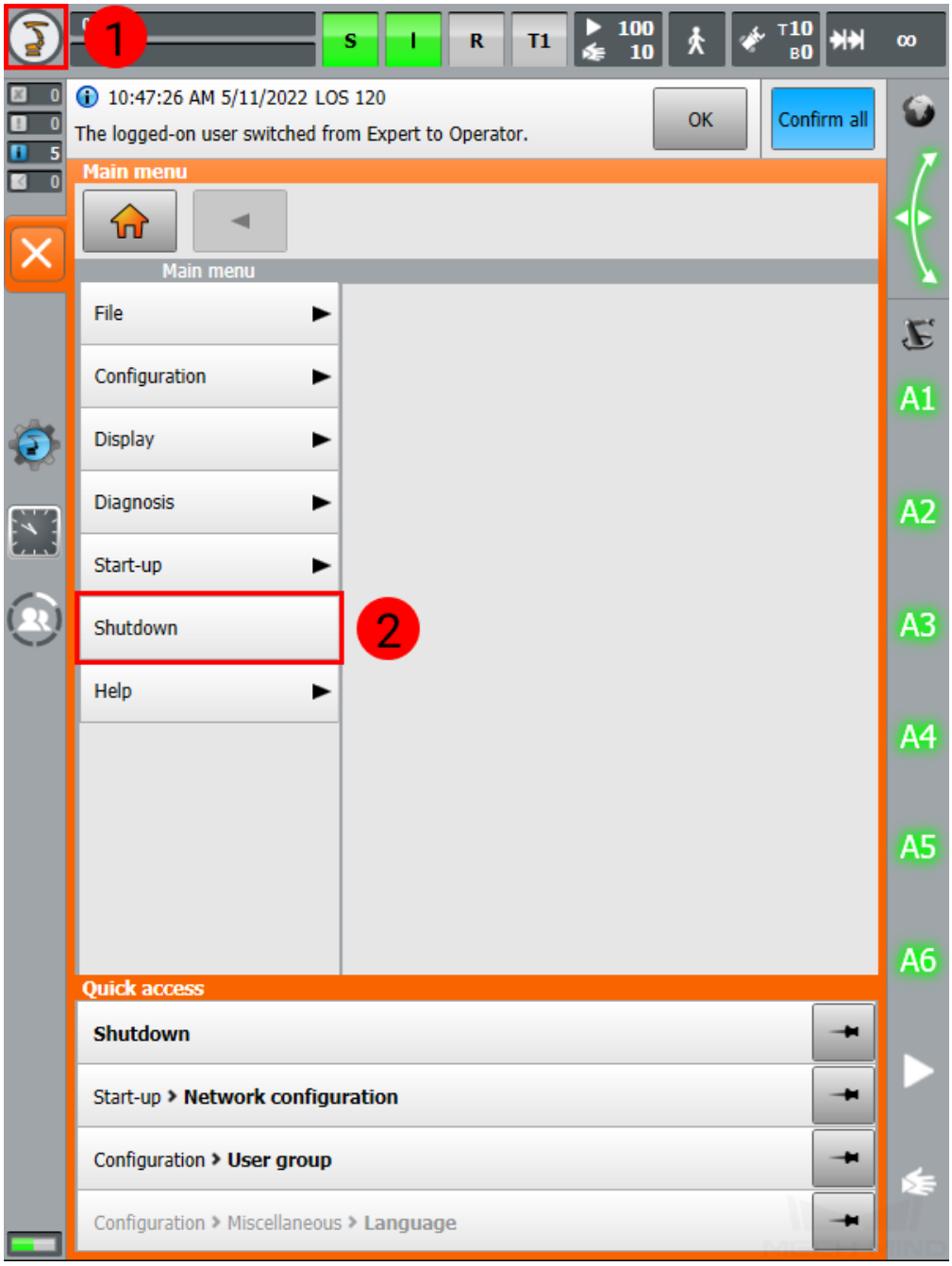
4. Input an **IP address** in the same subnet as that of the IPC, and then press on *Save*. Press on *Yes* and *OK*, respectively, in the next two pop-up windows.


10:41:27 AM 5/11/2022 LOS 120
The logged-on user switched from Operator to Expert.
OK
Confirm all
Network configuration
Windows interface (virtual5)
Address type:
Fixed IP address
1
IP address:
192 . 168 . 1 . 147
Subnet mask:
255 . 255 . 0 . 0
Standard gateway:
0 . 0 . 0 . 0
DNS Server:
0 . 0 . 0 . 0
Advanced...
The Windows interface is the network interface via which WorkVisual communicates with the controller. (If PROFINET is used without an advanced configuration, a static IP must be used.)
KLI
Internal su
2
Save
Back
A1
A2
A3
A4
A5
A6

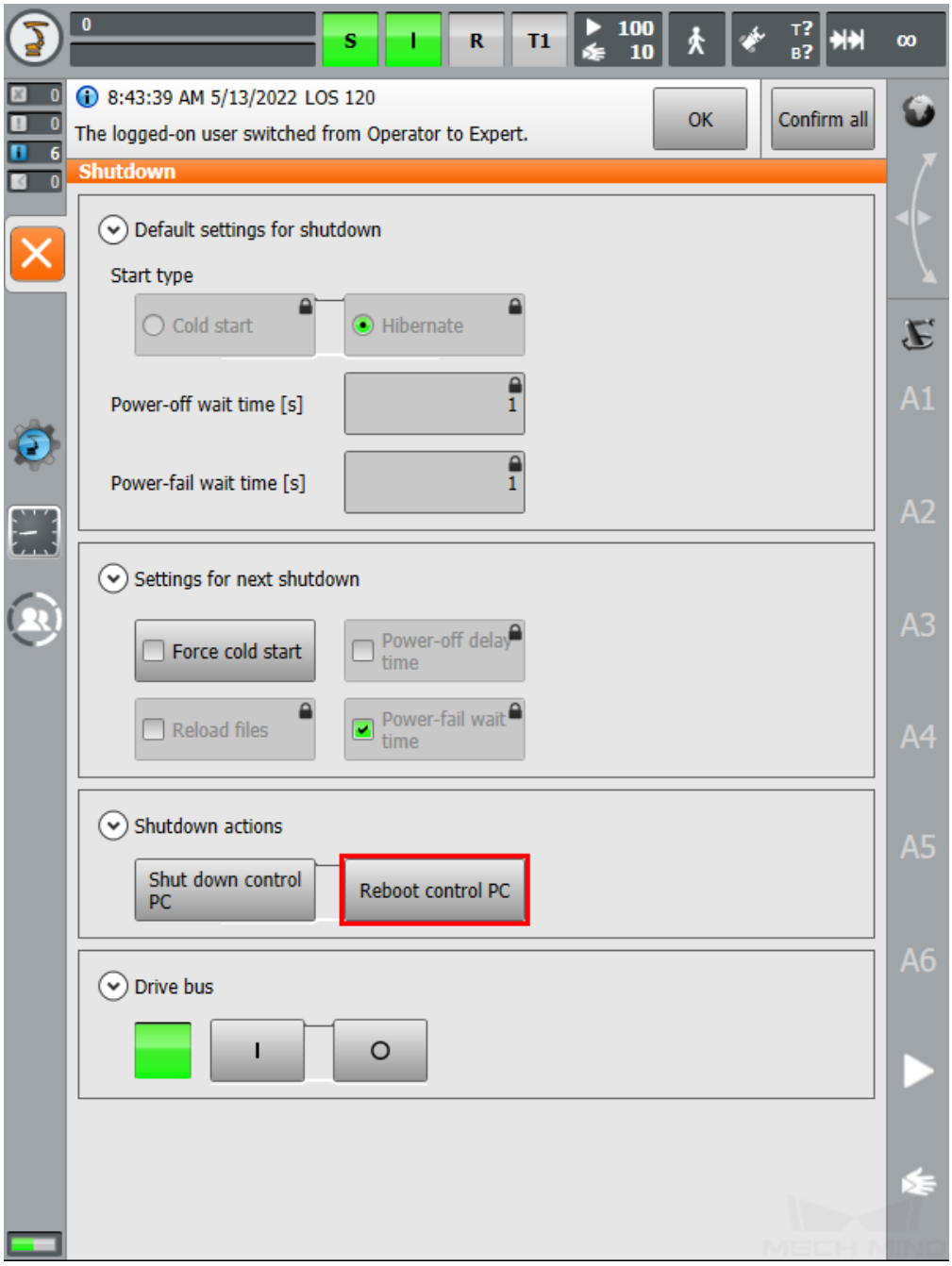


5. Restart the robot to finish setting the IP address:

   1. Press on  , and select **Shutdown**.



2. Press on *Reboot control PC*.


8:43:39 AM 5/13/2022 LOS 120
The logged-on user switched from Operator to Expert.
OK
Confirm all
Shutdown
Default settings for shutdown
Start type
Cold start
Hibernate
Power-off wait time [s]
1
Power-fail wait time [s]
1
Settings for next shutdown
Force cold start
Power-off delay time
Reload files
Power-fail wait time
Shutdown actions
Shut down control PC
Reboot control PC
Drive bus
I
O

Load the Program Files

Prepare the Files

The program files are stored in the installation directory of Mech-Center. The default directory is *C:/Mech-Mind/Mech-Center*.

Navigate to *xxx/Mech-Center/mech_interface/kuka*, and copy the following files to your flash drive.

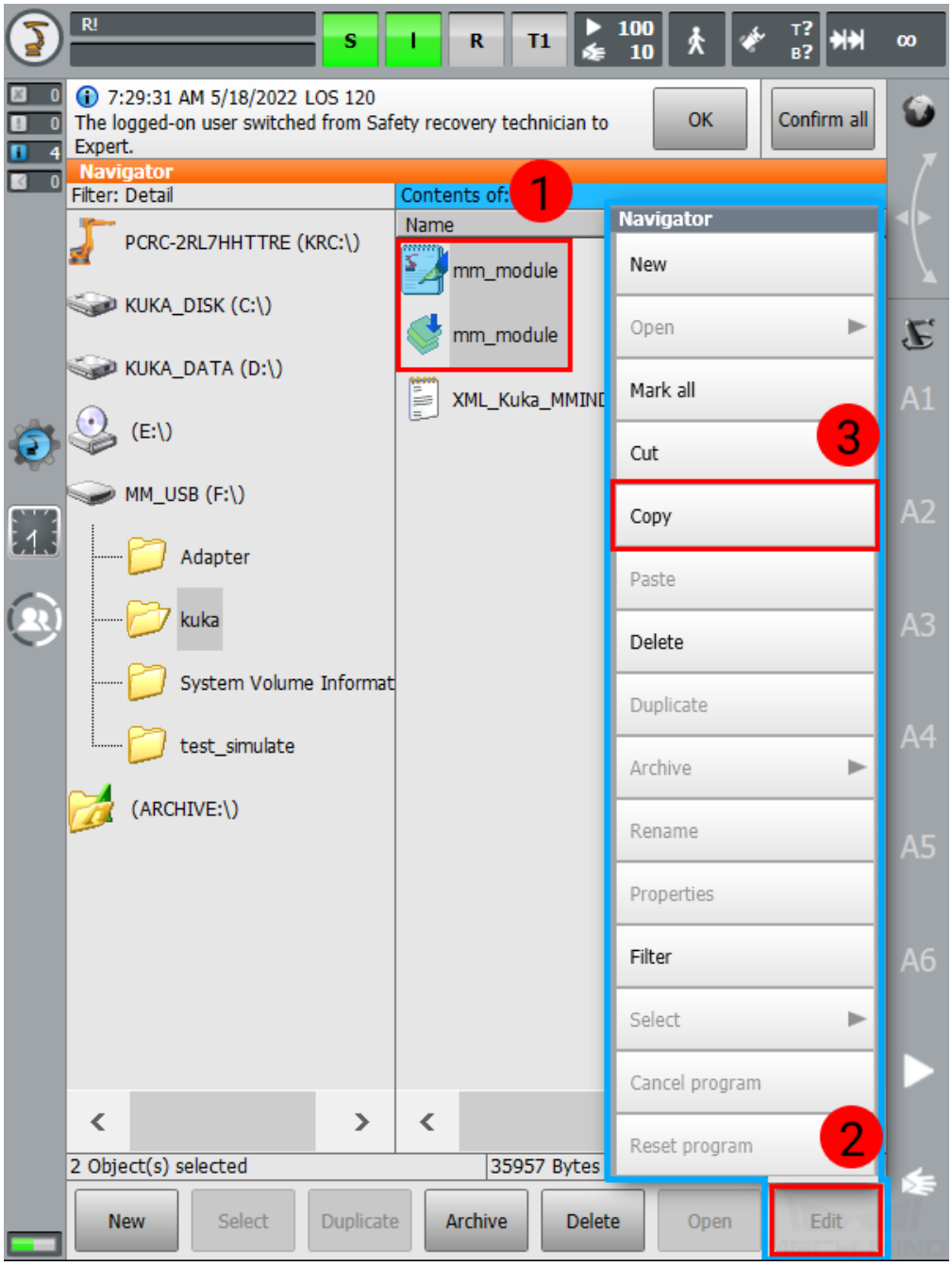
- **mm_module.src**
- **mm_module.dat**
- **XML_Kuka_MMIND.xml**

Load the Files to the Robot

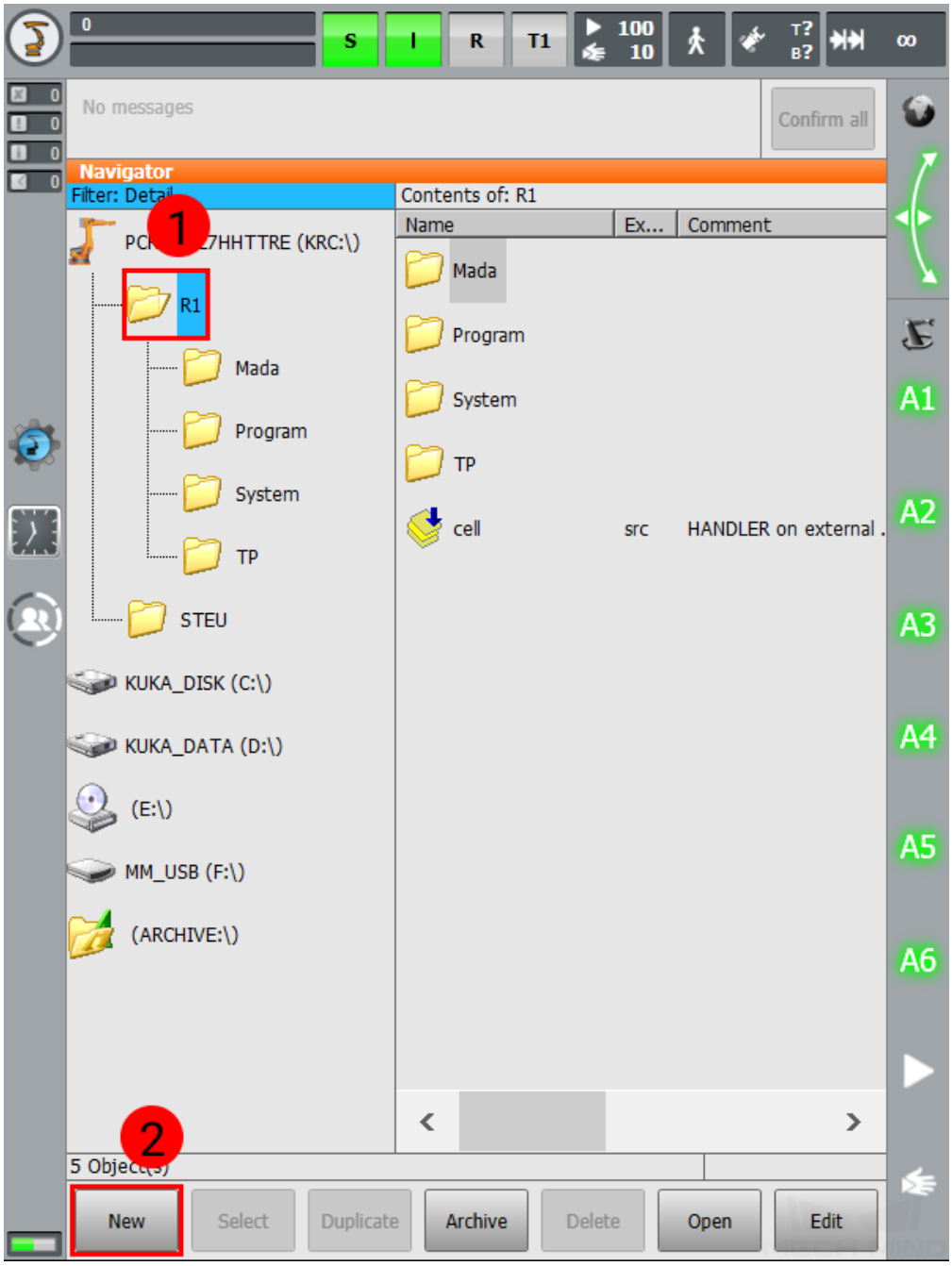
Note: Make sure you have switched to expert mode on the teach pendant. For instructions, see step 2 in **IP Configuration**.

---

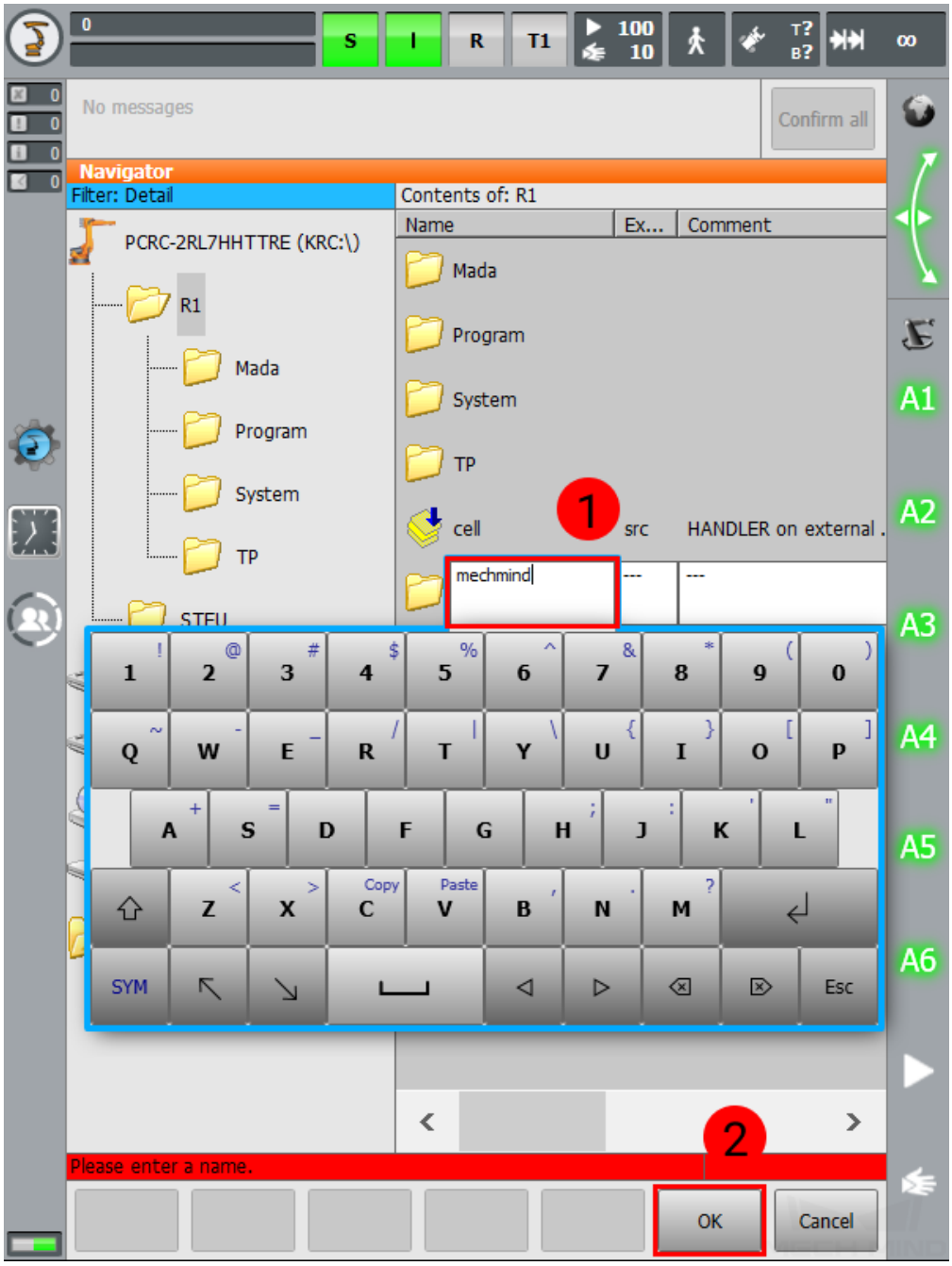
1. Plug the flash drive to the controller.
2. Select the flash drive, and locate the above files.
3. Long-press and select **mm_module.src** and **mm_module.dat**, press on *Edit*, and then select **Copy**.



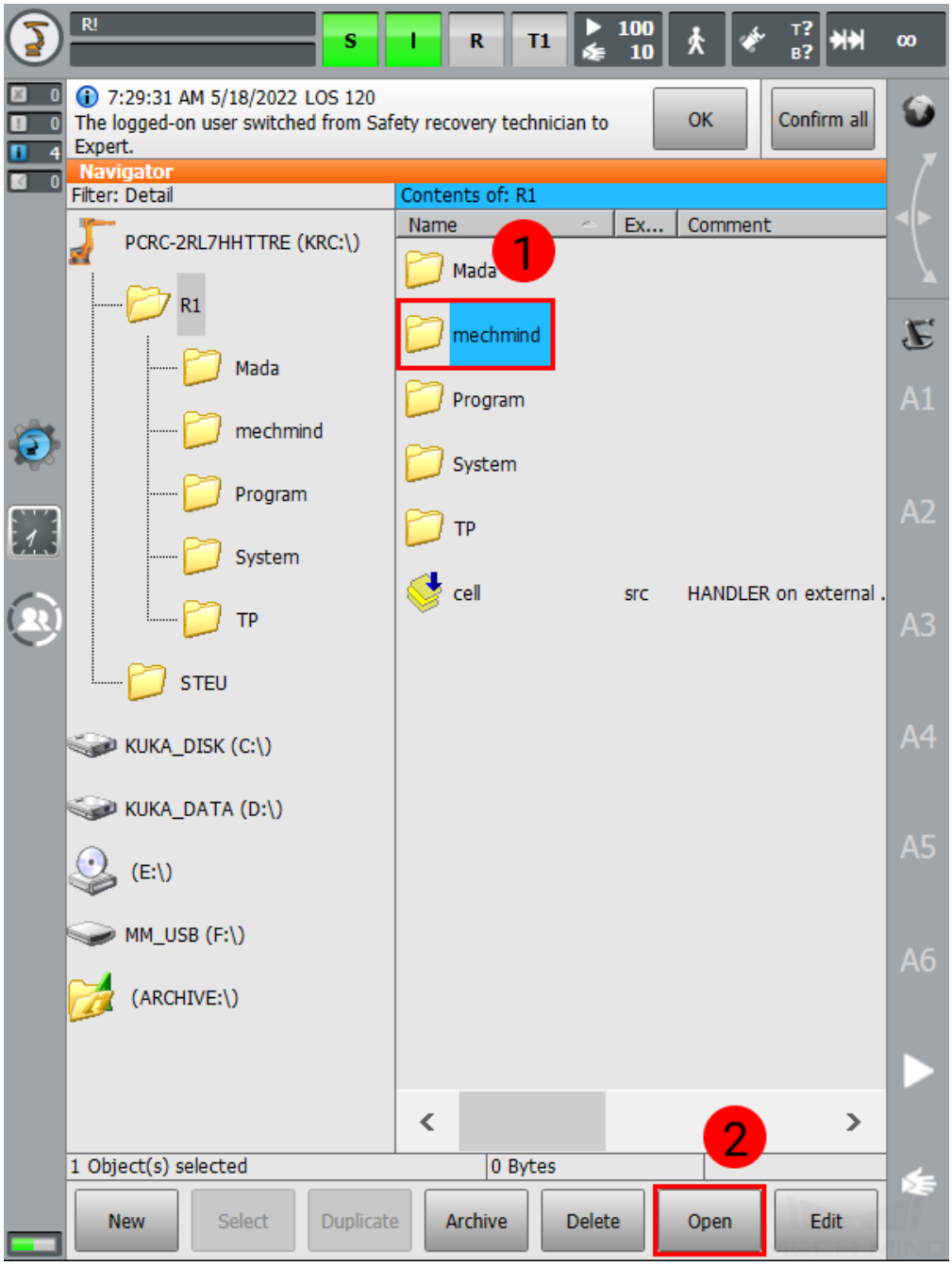
4. Navigate to *KRC:/R1*, and press on *New*.



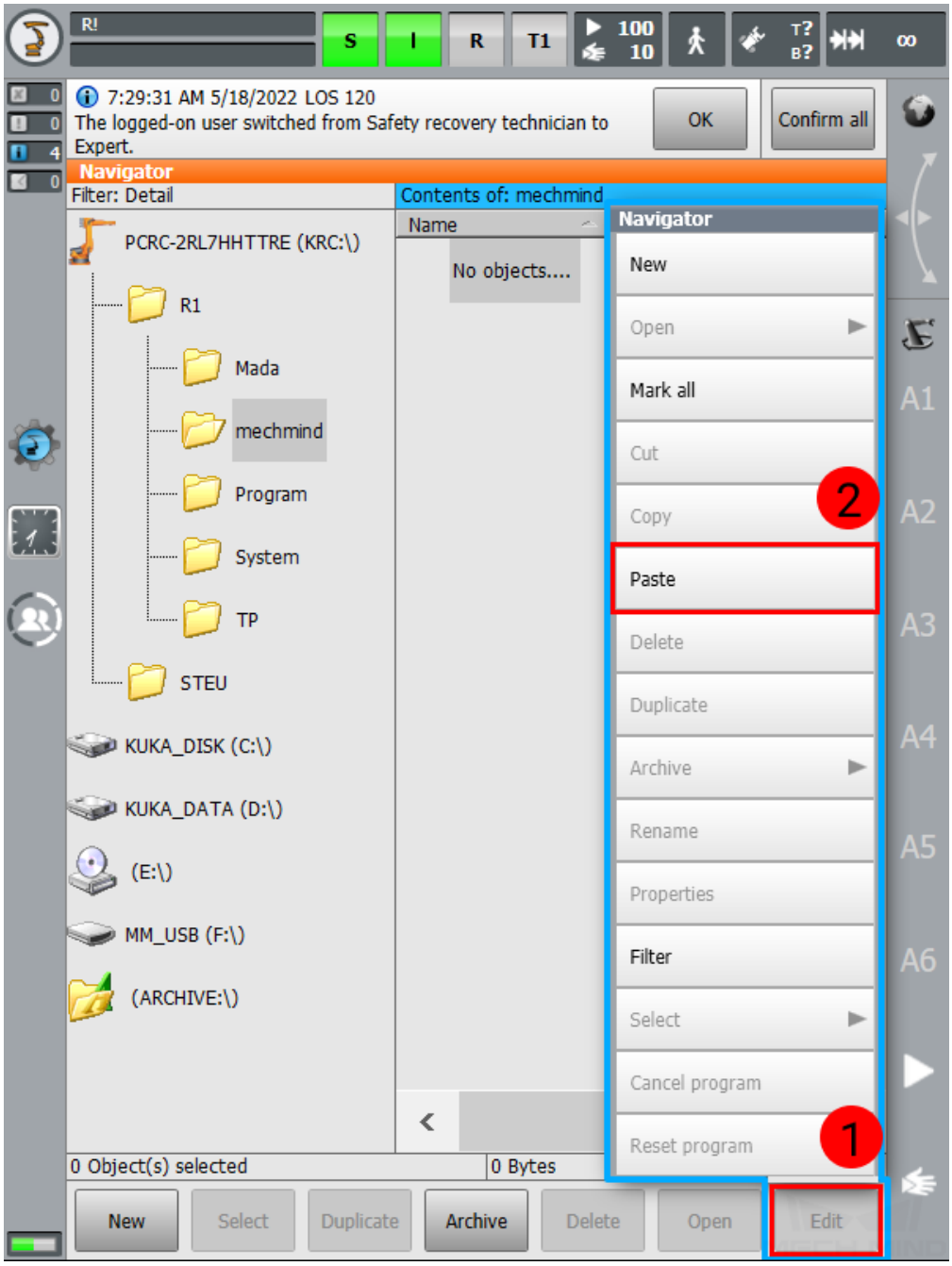
5. Input **mechmind** for the folder name, and press on *OK*.



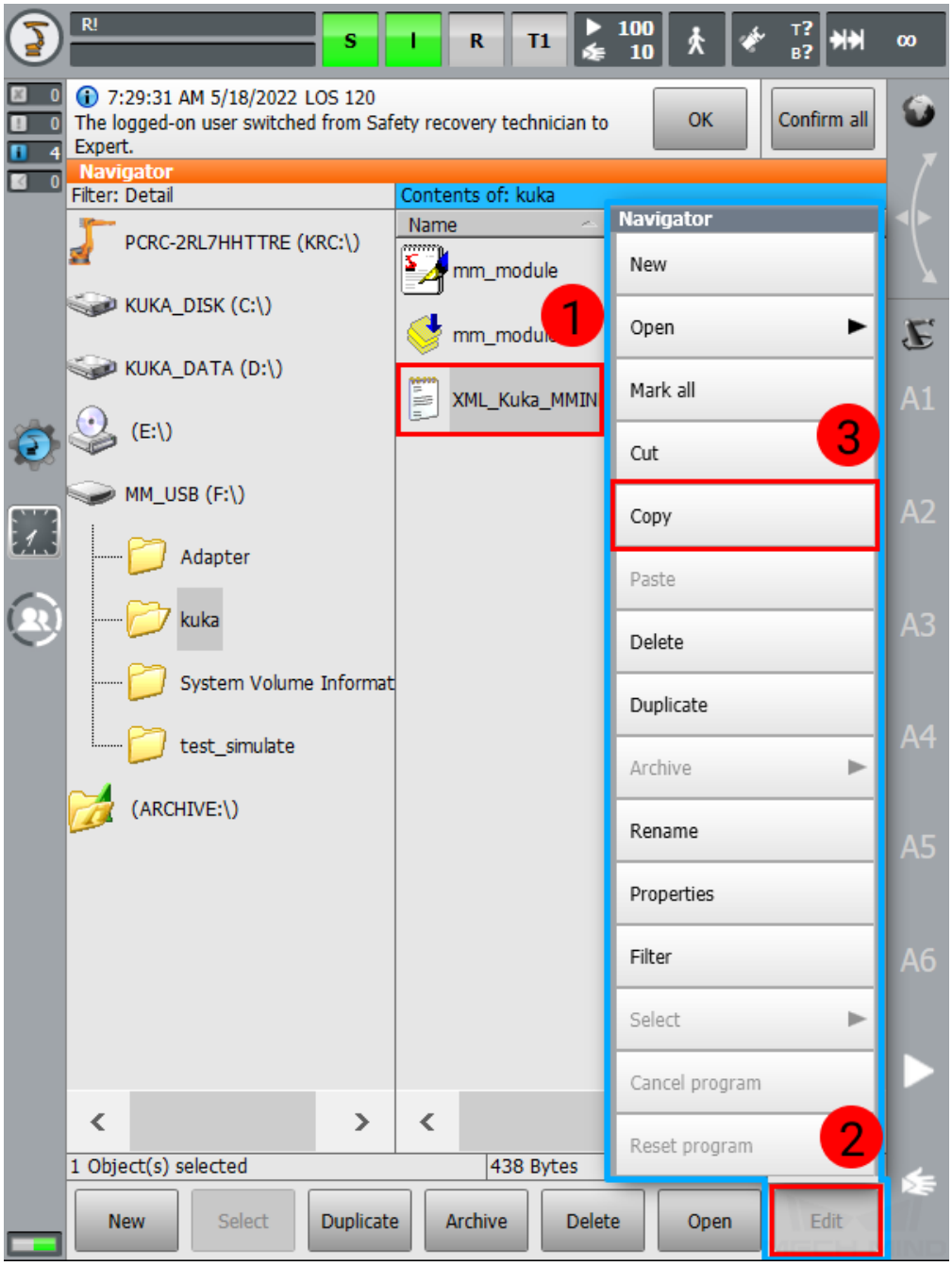
6. Select the **mechmind** folder, and press on **Open**.



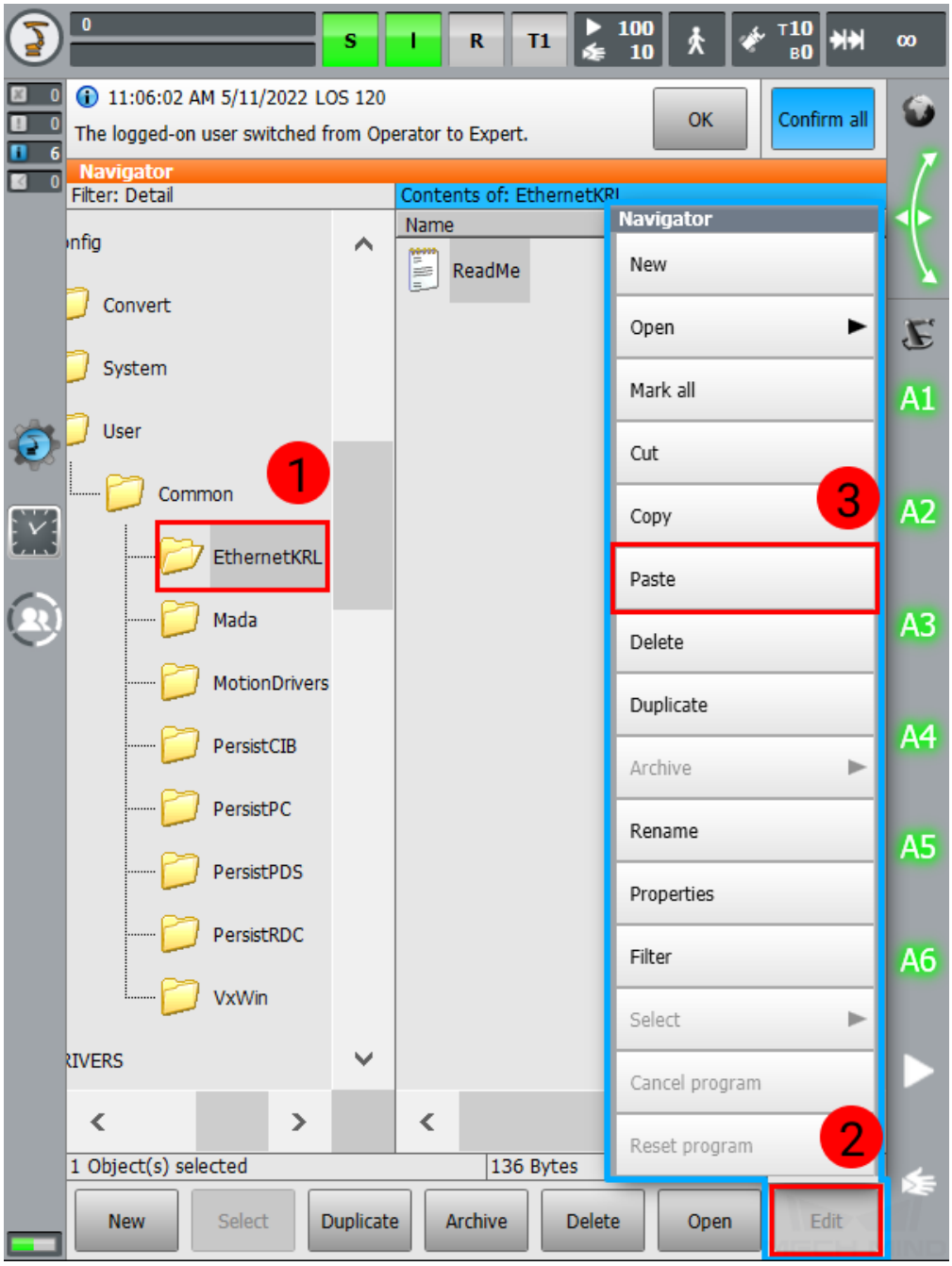
7. Press on *Edit*, and then select **Paste**.



8. Navigate back to the flash drive, and copy **XML_Kuka_MMIND.xml**.

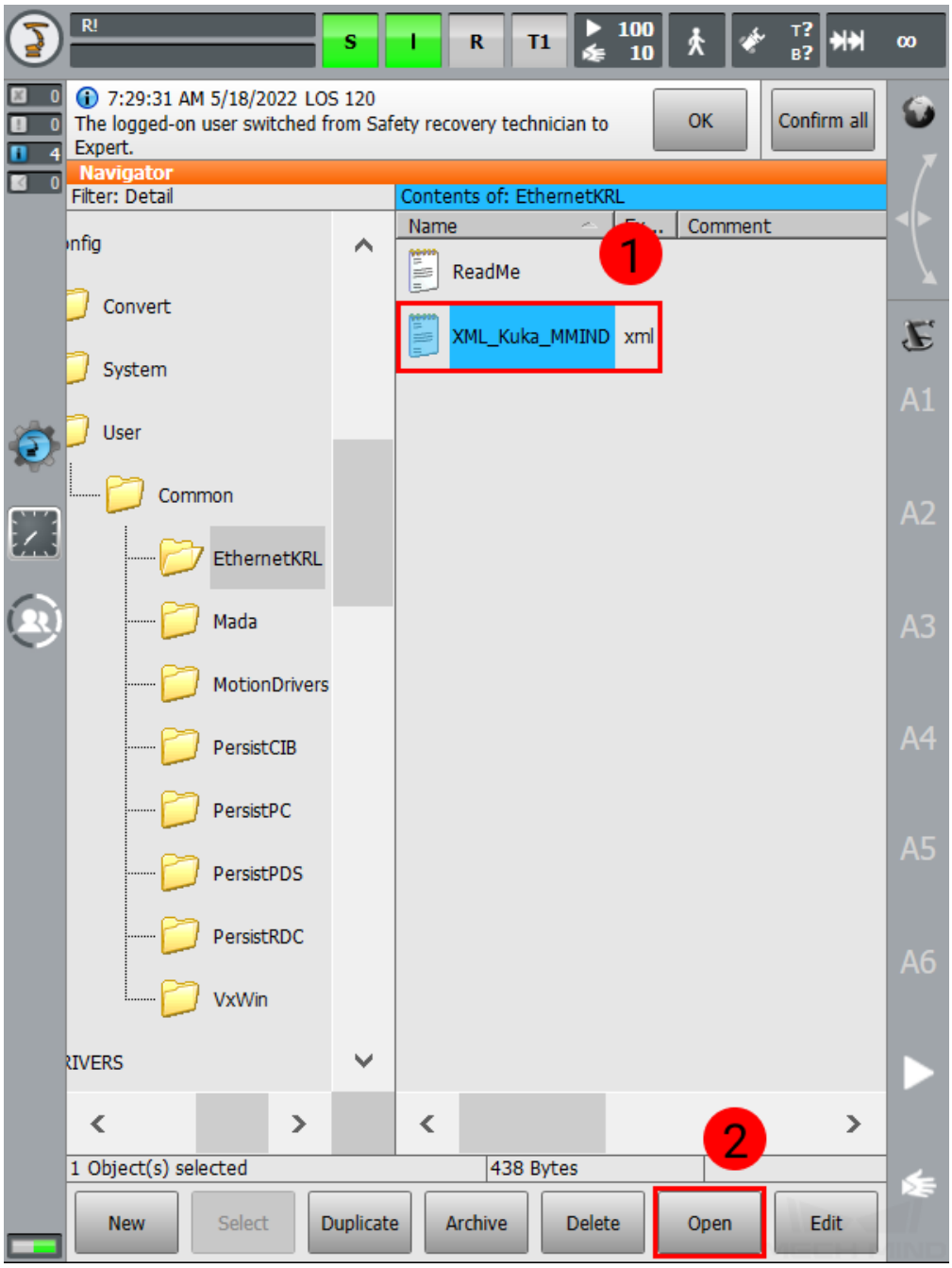


9. Navigate to *C:/KRC/ROBOTER/Config/User/Common/EthernetKRL*, press on *Edit*, and then select **Paste**.

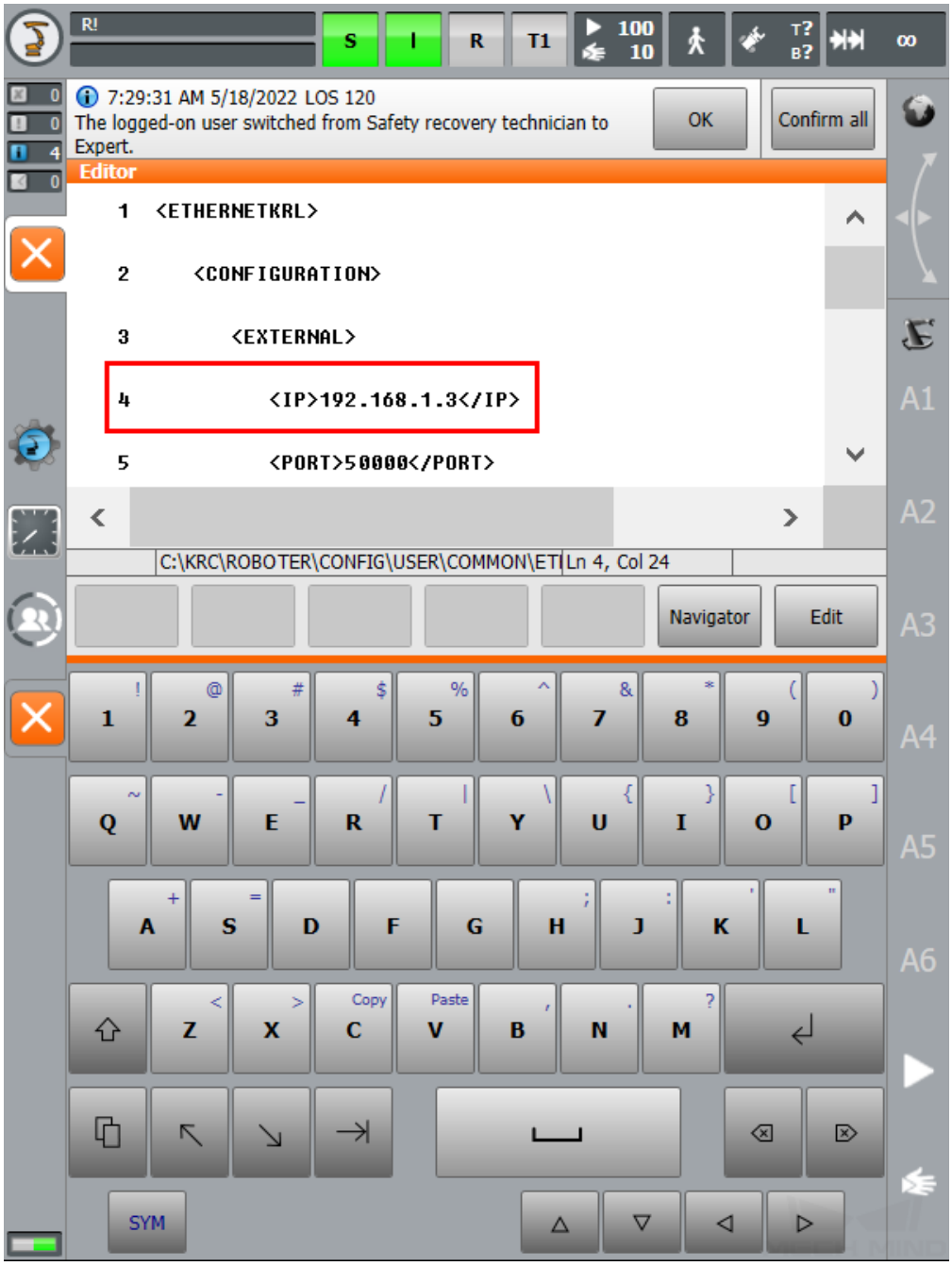

11:06:02 AM 5/11/2022 LOS 120
The logged-on user switched from Operator to Expert.
OK
Confirm all
Navigator
Filter: Detail
Contents of: EthernetKRL
Convert
System
User
Common
EthernetKRL
Mada
MotionDrivers
PersistCIB
PersistPC
PersistPDS
PersistRDC
VxWin
Name
ReadMe
New
Open
Mark all
Cut
Copy
Paste
Delete
Duplicate
Archive
Rename
Properties
Filter
Select
Cancel program
Reset program
1 Object(s) selected
136 Bytes
New
Select
Duplicate
Archive
Delete
Open
Edit
1
2
3

Modify IP Address in XML file

1. Select **XML_Kuka_MMIND.xml**, and press on *Open*.



2. Press on line 4 where it says **<IP>192.168.1.1<IP>**, and press  to change the IP address to the IPC' s actual IP address.



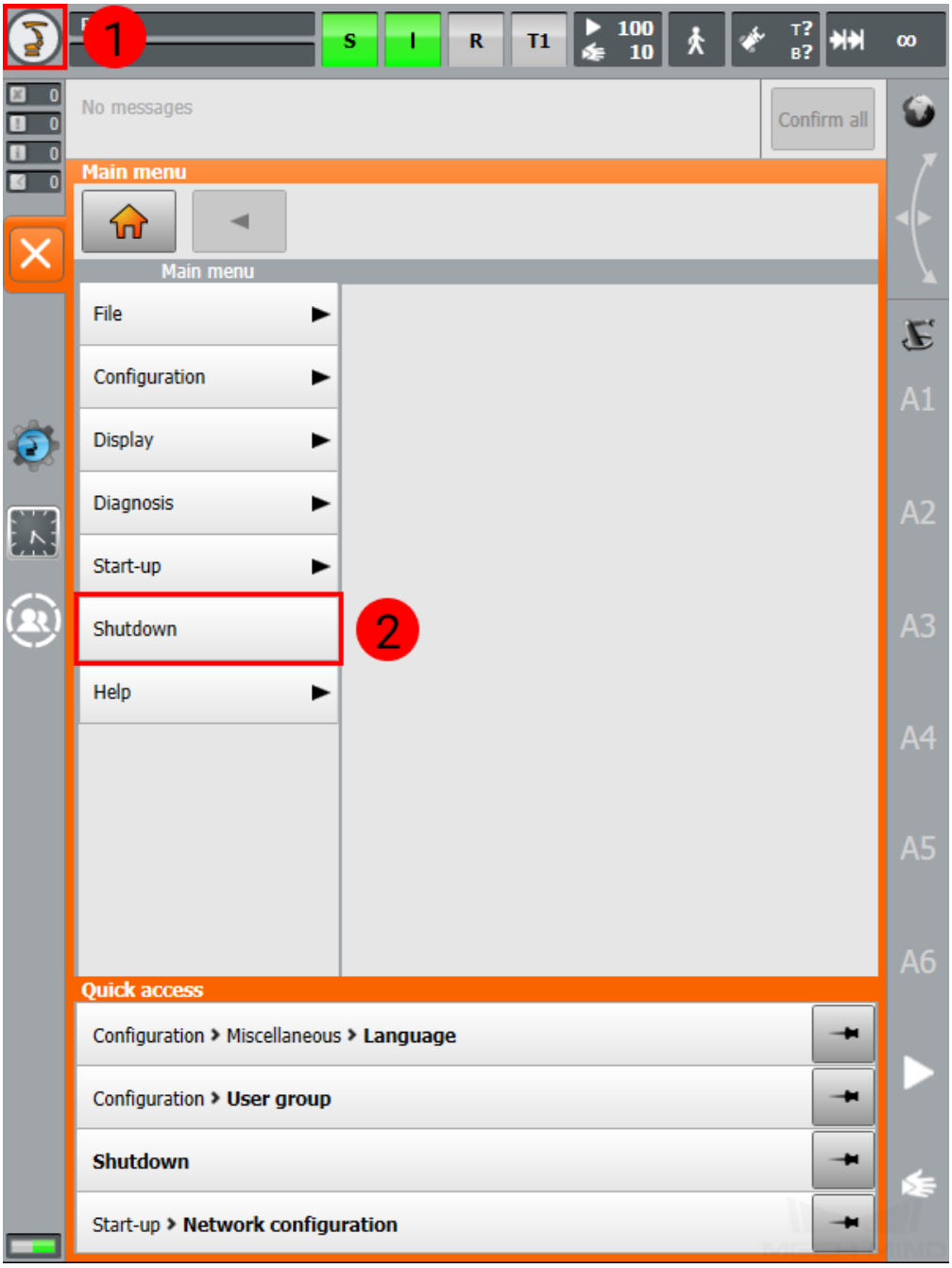
3. Check line 5 to see if the port number is correct. If not, change it to the IPC' s actual port number.

4. Press on  and then *Yes* to save the changes.

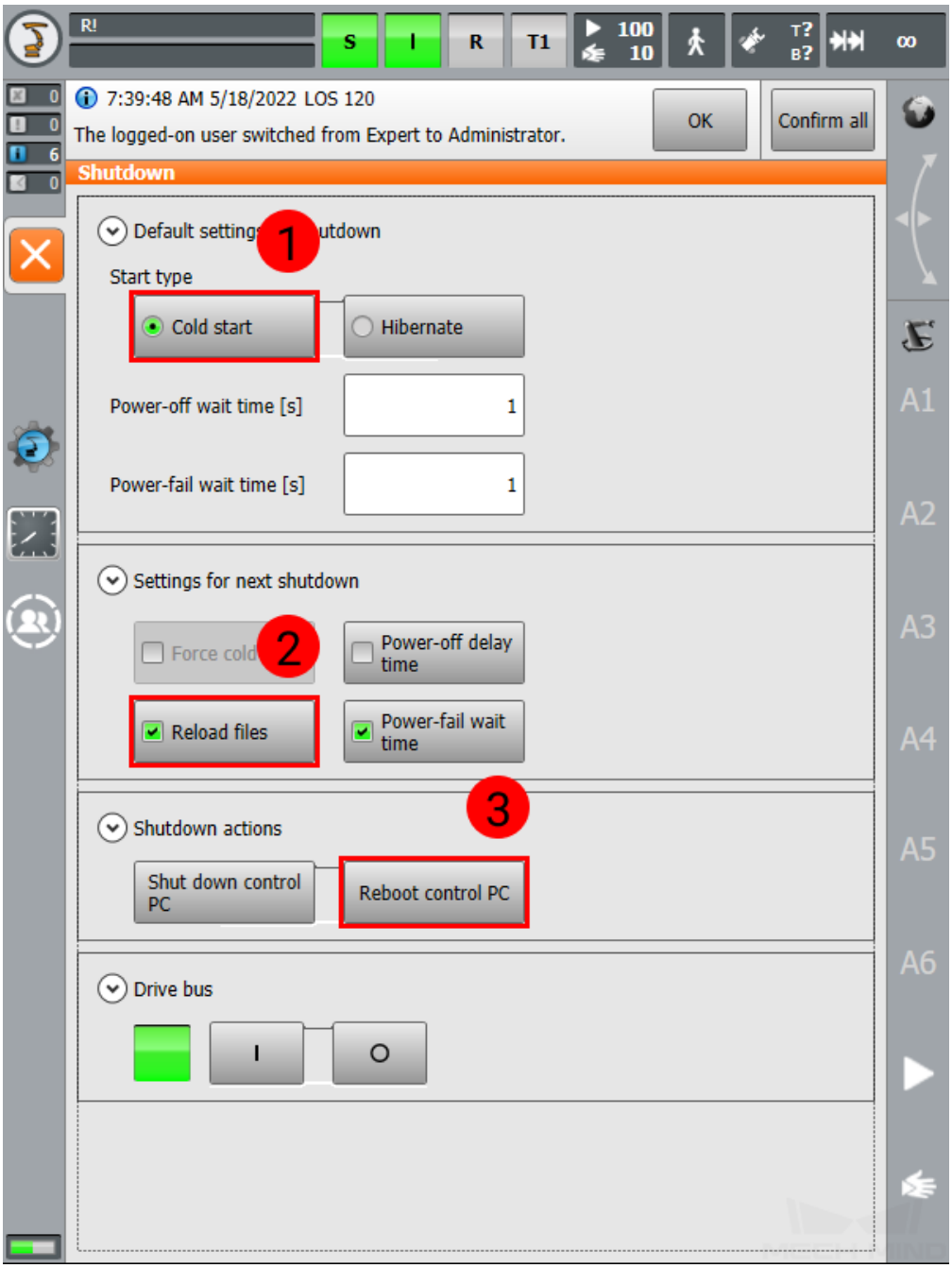

7:29:31 AM 5/18/2022 LOS 120
The logged-on user switched from Safety recovery technician to Expert.
OK
Confirm all
Editor
1 <ETHERNETKRL>
<CONFIGURATION>
3 <EXTERNAL>
4 <IP>192.168.1.3</IP>
5 <PORT>50000</PORT>
1
7:38:46 AM 5/18/2022 XEdit 19
C:\KRC\ROBOTER\CONFIG\USER\COMMON\ETHERNETKRL\XML_KUKA_MMIND.XML modified. Save changes?
Yes
No
Cancel
2
SYM

Restart the Robot

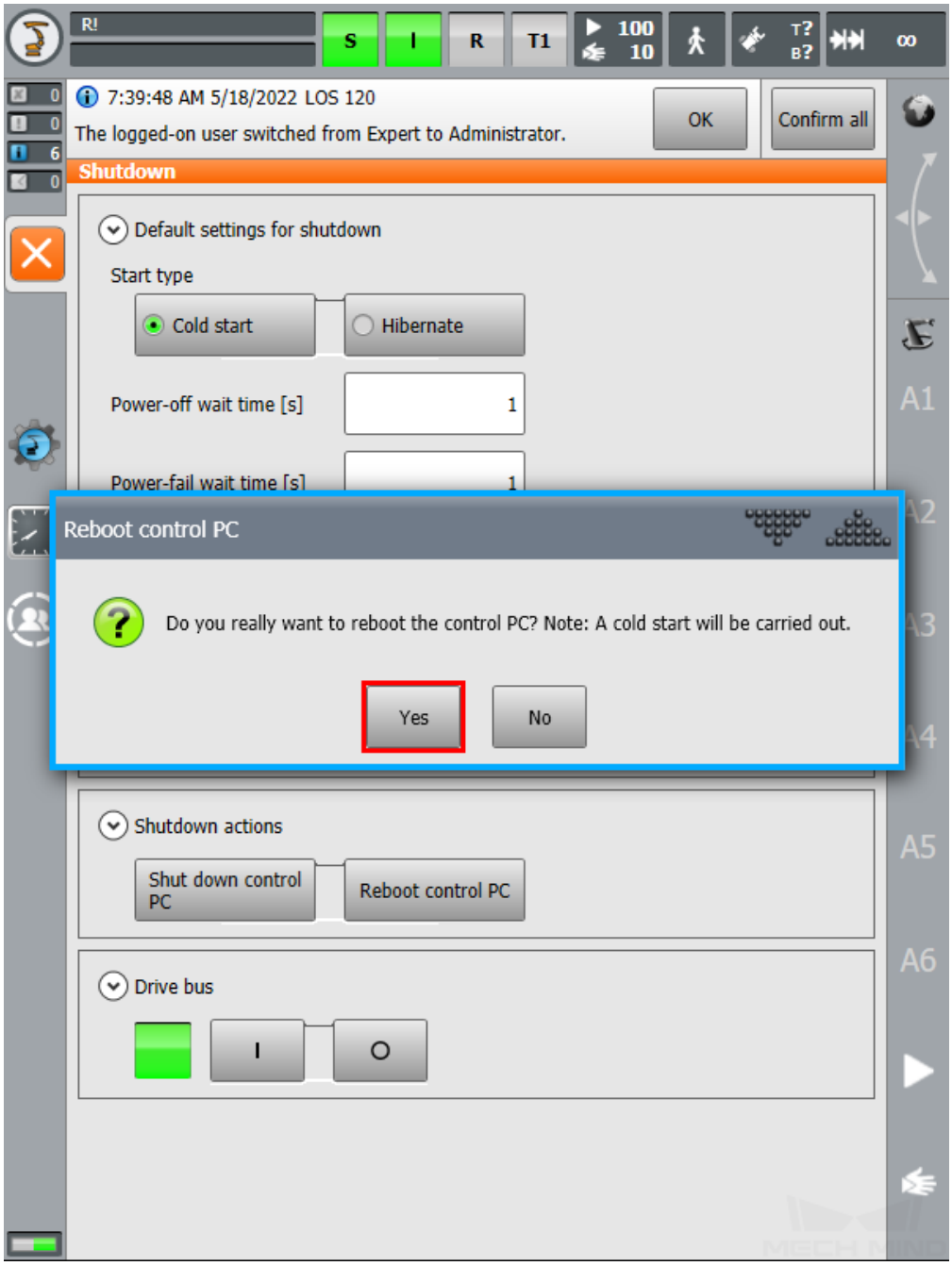
1. Switch to administrator mode following the instructions of step 2 in **IP Configuration**. The default password is **kuka**.
2. Press on  , and select **Shutdown**.



3. Select **Cold start** and **Reload files**, and then press on *Reboot control PC*.



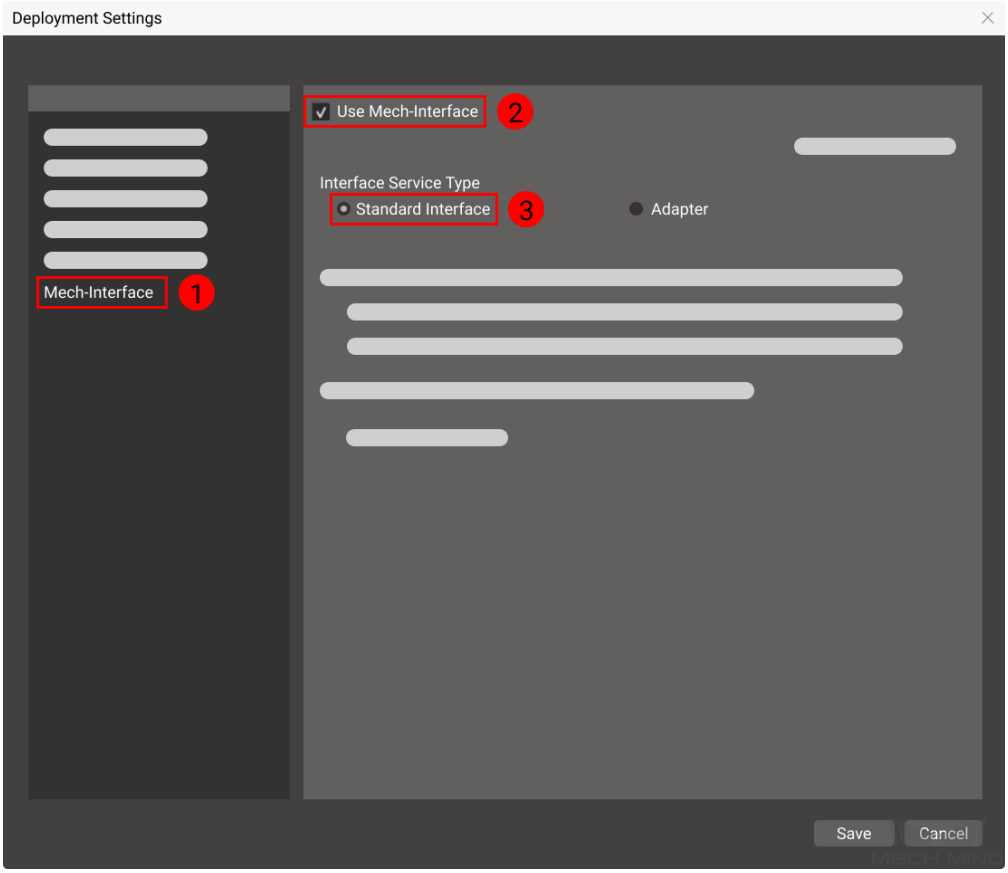
4. Press on *Yes* in the pop-up window to restart the robot.


7:39:48 AM 5/18/2022 LOS 120
The logged-on user switched from Expert to Administrator.
OK
Confirm all
Shutdown
Default settings for shutdown
Start type
Cold start
Hibernate
Power-off wait time [s]
1
Power-fail wait time [s]
1
Reboot control PC
Do you really want to reboot the control PC? Note: A cold start will be carried out.
Yes
No
Shutdown actions
Shut down control PC
Reboot control PC
Drive bus

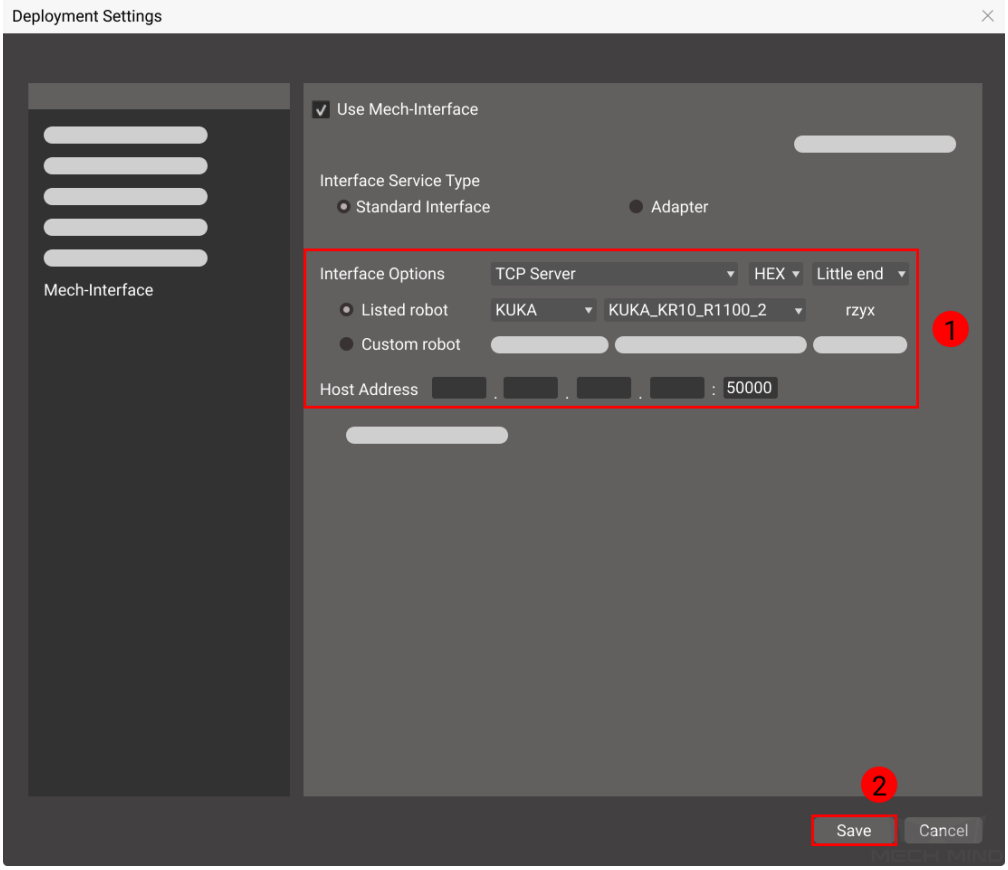
Test Robot Connection

Configure Mech-Interface in Mech-Center

1. Open Mech-Center and click on *Deployment Settings*.
2. Go to **Mech-Interface**, check **Use Mech-Interface** and select **Standard Interface**.



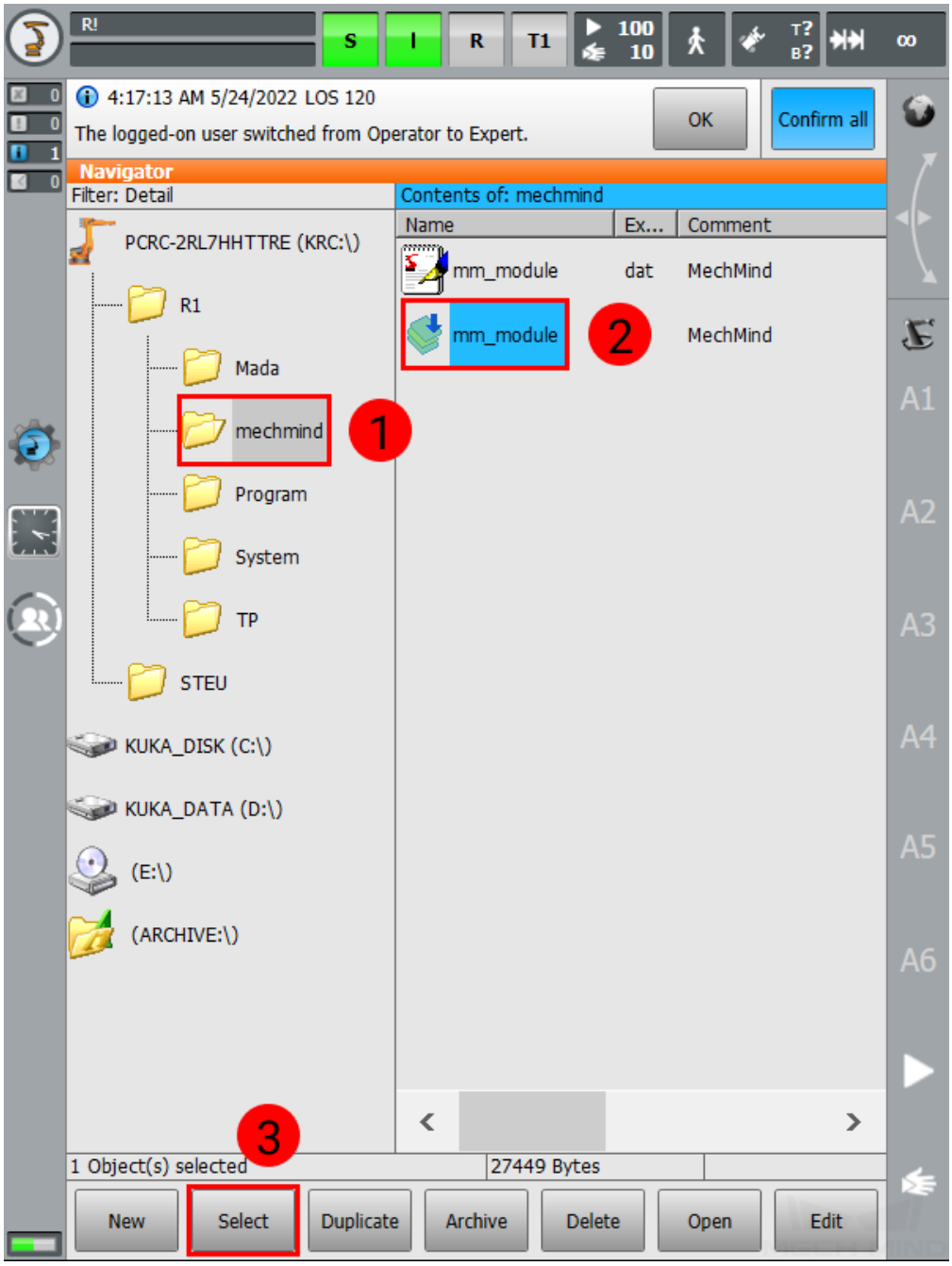
3. Set the following fields:
   - **Interface Option**: Set to **TCP Server**, **HEX**, and **Little endian**.
   - **Listed robot**: Select the robot model you are using.
   - **Host Address**: The default port number is **50000**. If you need to change the port number, make sure to change it accordingly in the **XML_Kuka_MMIND.xml** on the teach pendant as well.
4. Click on *Save*.



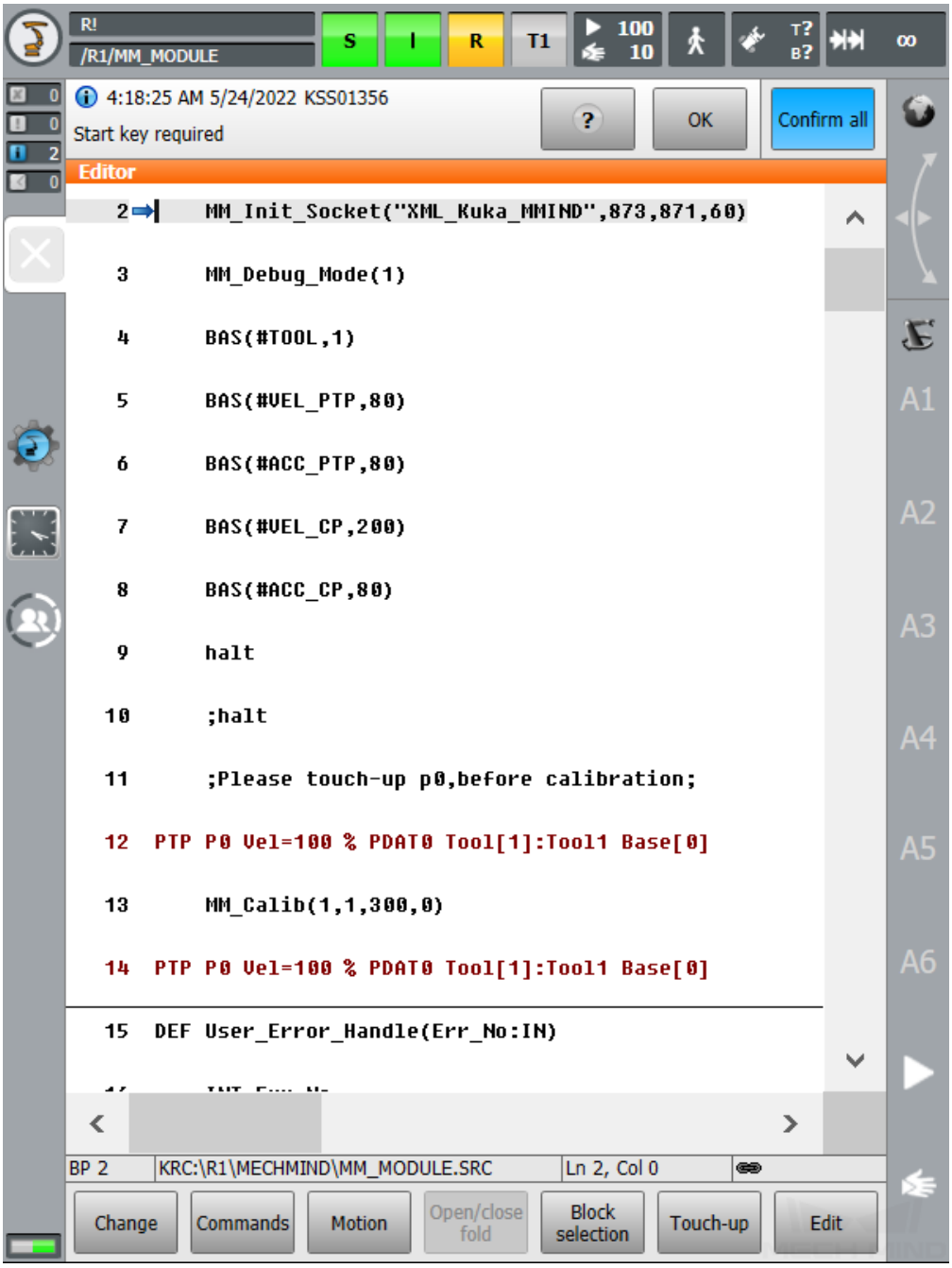
5. Click on *Start Interface* in the Toolbar.

Run the Robot Program

1. On the teach pendant, open the **mechmind** folder, select **mm_module.src**, and then press on *Select*.



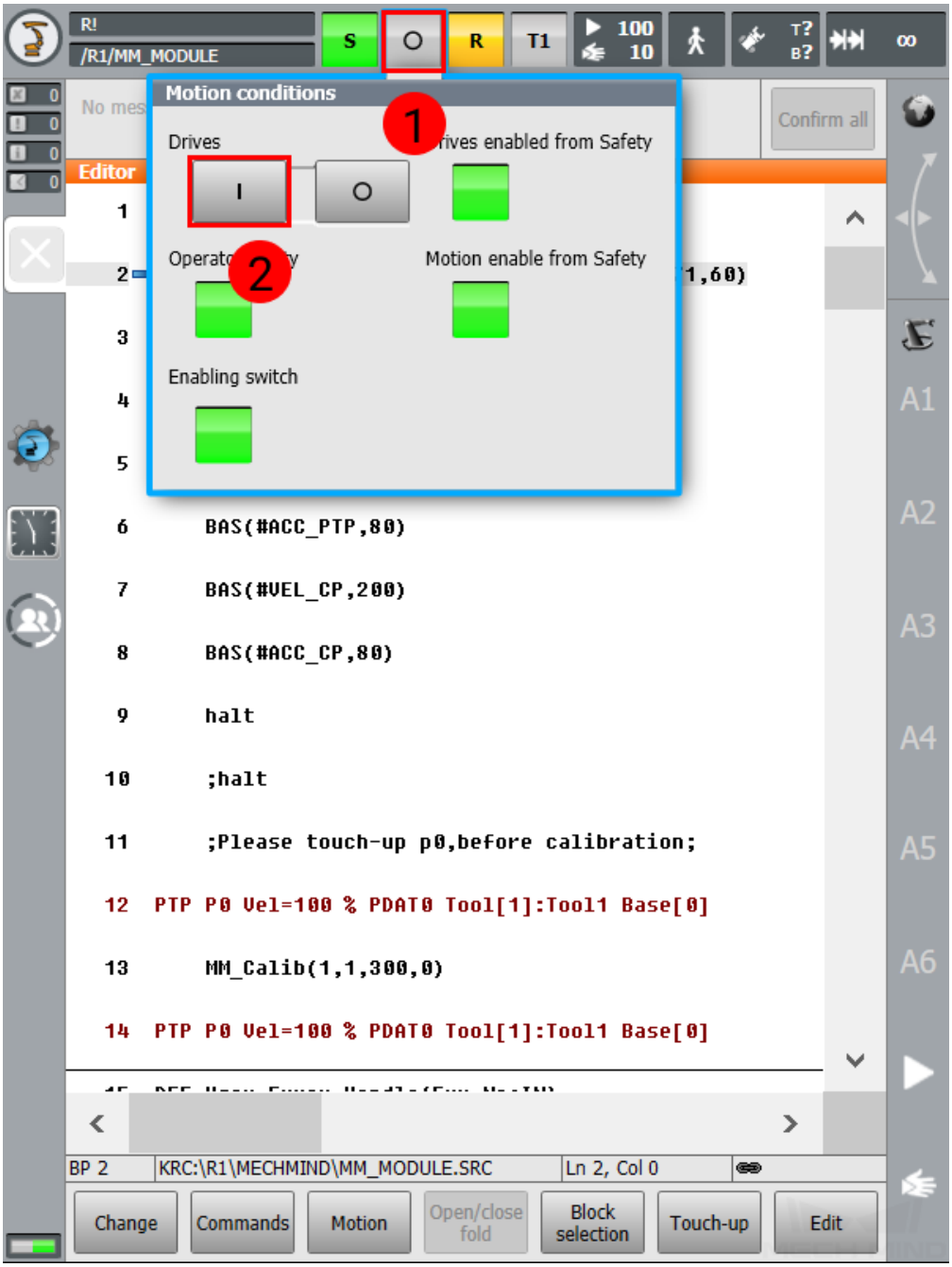
2. The following should appear on the screen.



3. Turn the key switch to horizontal, select **T1** on the screen, and then turn the switch back to vertical.

4. Check the icon to the right of  :

   - If it looks like  , then skip this step.
   - If it looks like  , then press on it and select  in the drop-down window.



5. Press on the enabling switch (either one of three) on the back of the pendant and  on the front at the same time to move the robot back to P0 position. When the screen displays a message saying **Programmed path reached (BCO)**, and  turns red, release the enabling switch and  .

Note: Set an appropriate speed for the robot before moving it, and observe its motion carefully to avoid accidents.

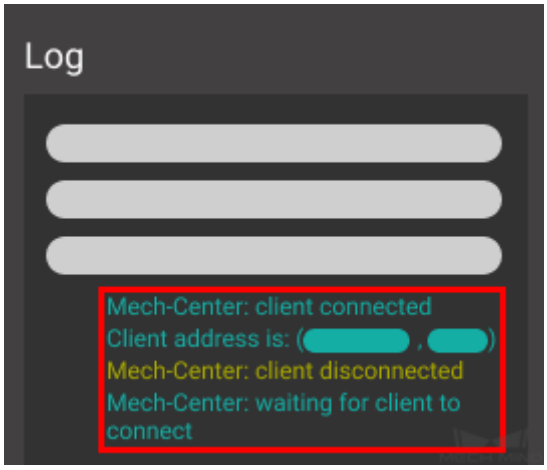
6. Switch to **AUTO** mode as described in step 3, and press on  to start running the full-control program (

should turn green).

7. The robot can be successfully connected if Mech-Center' s **Log** panel displays the following messages:

   - **Mech-Center: client connected**

- A message showing the **client address**
- **Mech-Center: client disconnected**
- **Mech-Center: waiting for client to connect**



2.4.2 KUKA Calibration Program

This section introduces the process of calibrating the camera extrinsic parameters using the calibration subprogram.

The process consists of 4 steps:

- *Select the Calibration Program*
- *Teach the Calibration Start Point*
- *Run the Calibration Program*
- *Start Calibration in Mech-Vision*

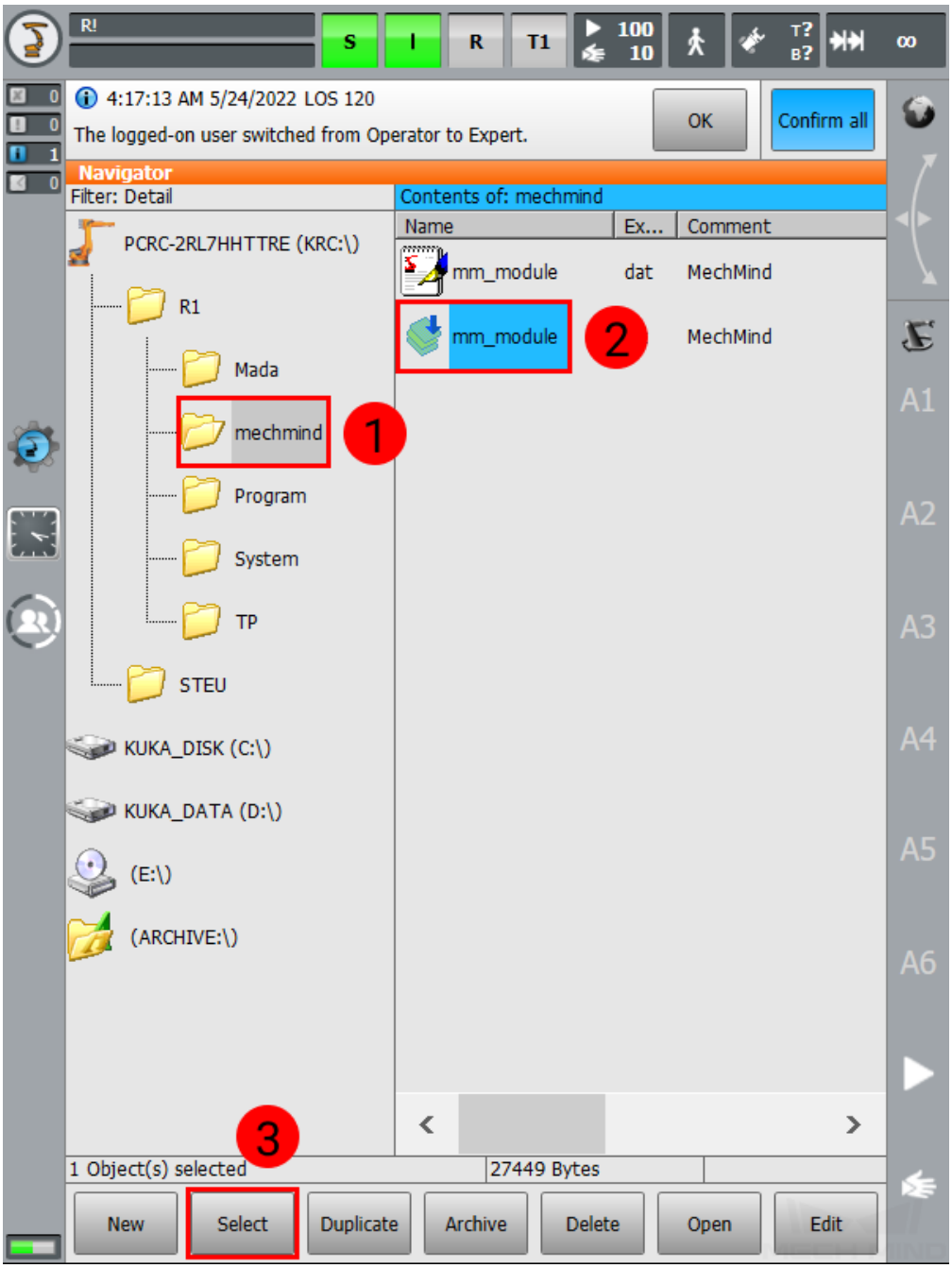
Before proceeding, please make sure that:

- You have *loaded the Standard Interface program* onto the robot and can establish communcation with Mech-Center.
- You are familiar with the contents in calibration_guide.

Note: This section is intended for scenarios where the communication between the robot and Mech-Center is established through Standard Interface, and calibration has to be performed frequently.

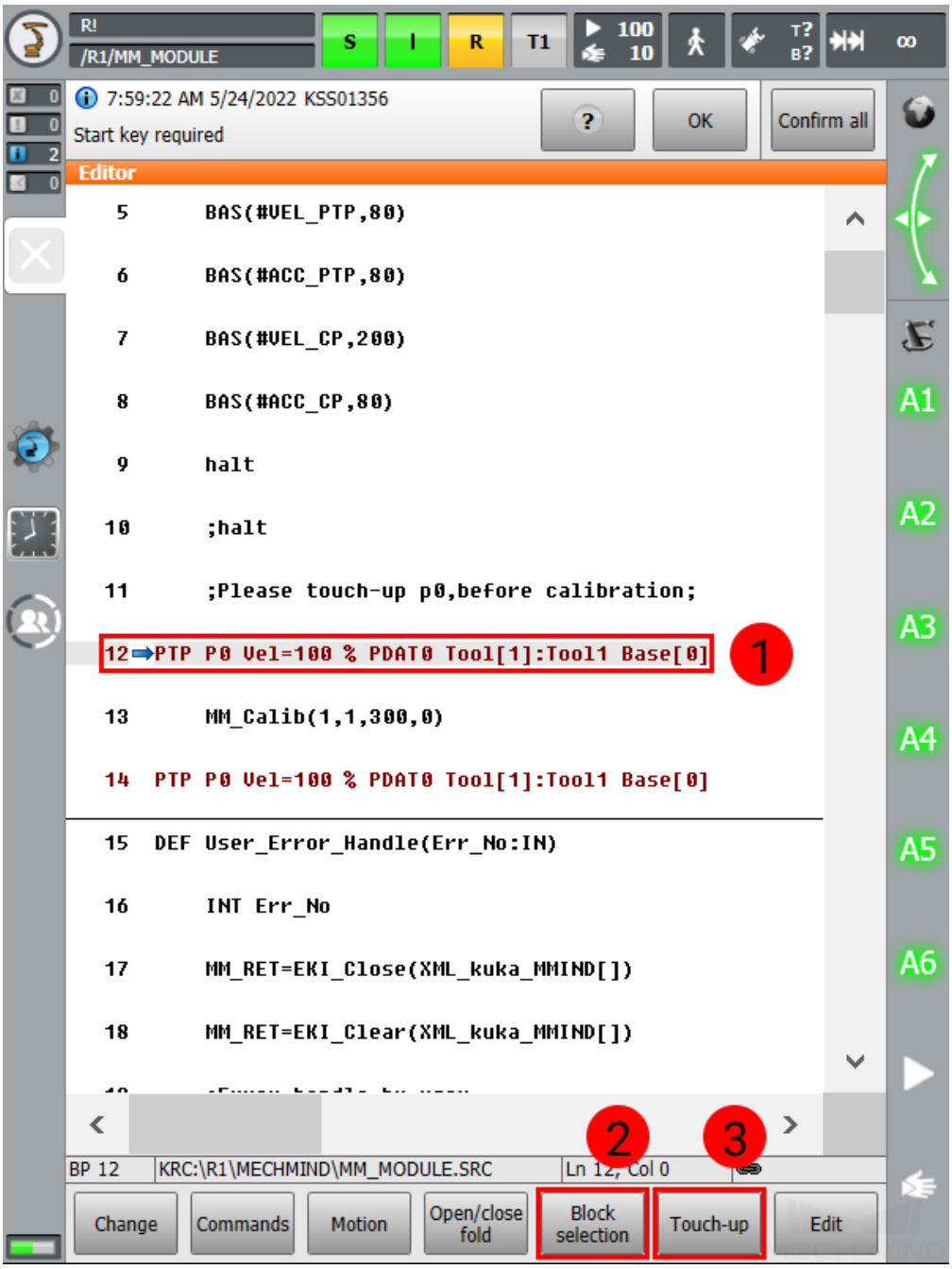
Select the Calibration Program

On the teach pendant, navigate to *KRC:/R1/mechmind*, select **mm_module.src**, and then press on *Select*.

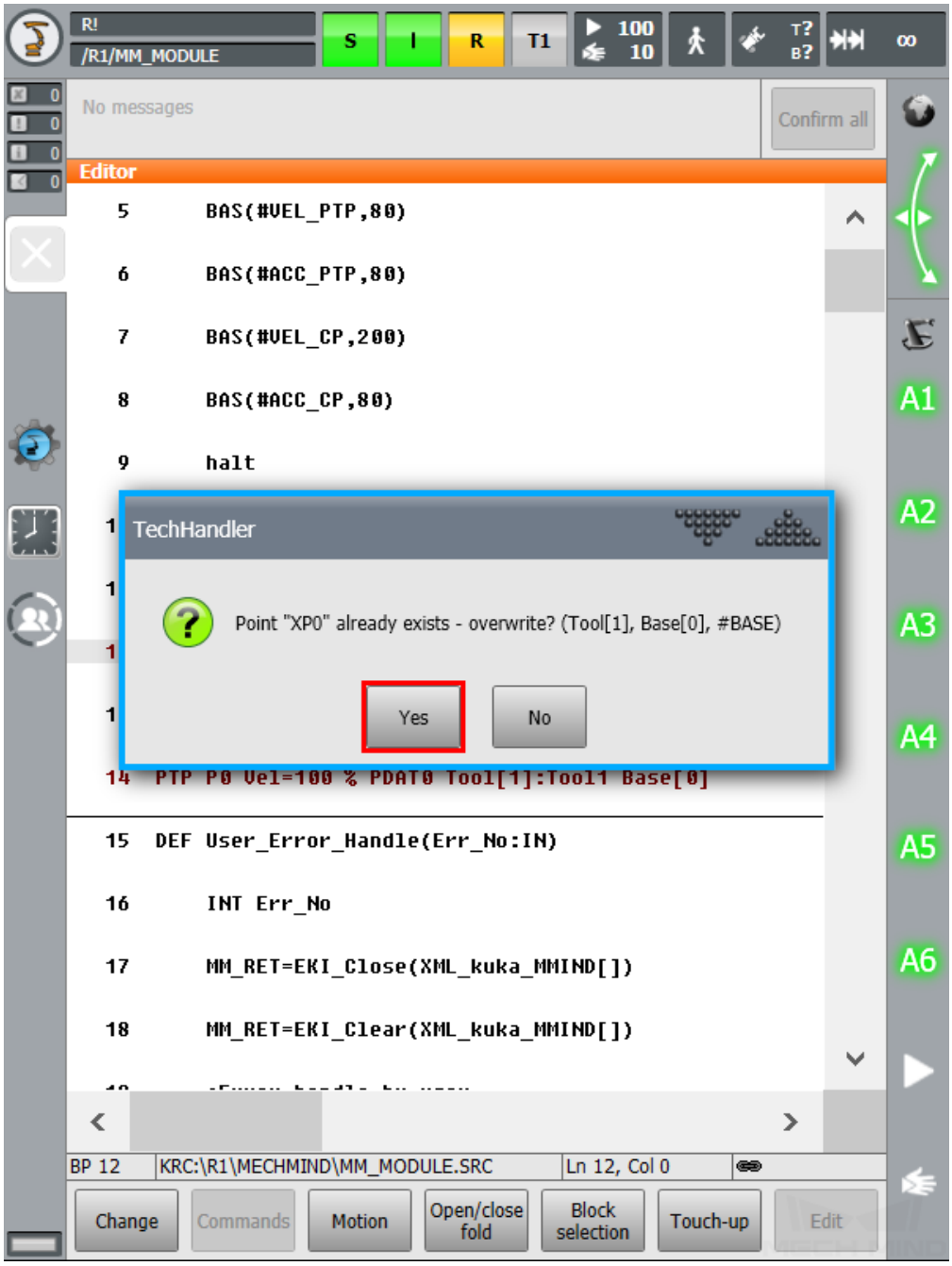


Teach the Calibration Start Point

1. Move the robot to the start point for the calibration.
2. On the teach pendant, move the cursor to Line 12, and press on *Block selection* and then *Touch-up*.

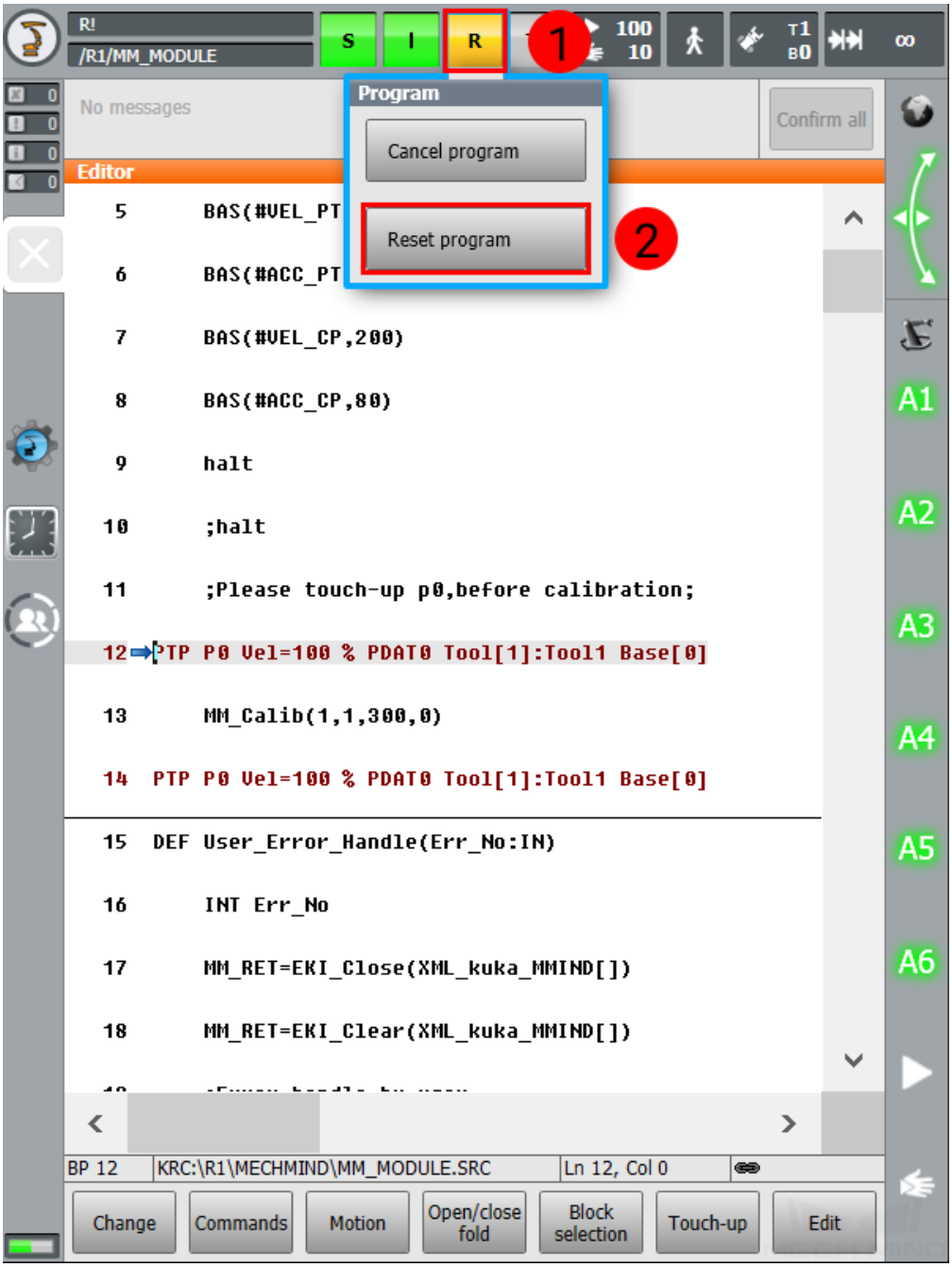


3. Press on *Yes* in the pop-up window to finish teaching the initial point for calibration.


R!
/R1/MM_MODULE
S
I
R
T1
100
10
No messages
Confirm all
Editor
5 BAS(#VEL_PTP,80)
6 BAS(#ACC_PTP,80)
7 BAS(#VEL_CP,200)
8 BAS(#ACC_CP,80)
9 halt
TechHandler
Point "XP0" already exists - overwrite? (Tool[1], Base[0], #BASE)
Yes
No
14 PTP P0 Vel=100 % PDAT0 Tool[1]:Tool1 Base[0]
15 DEF User_Error_Handle(Err_No:IN)
16 INT Err_No
17 MM_RET=EKI_Close(XML_kuka_MMIND[])
18 MM_RET=EKI_Clear(XML_kuka_MMIND[])
BP 12
KRC:\R1\MECHMIND\MM_MODULE.SRC
Ln 12, Col 0
Change
Commands
Motion
Open/close fold
Block selection
Touch-up
Edit
A1
A2
A3
A4
A5
A6

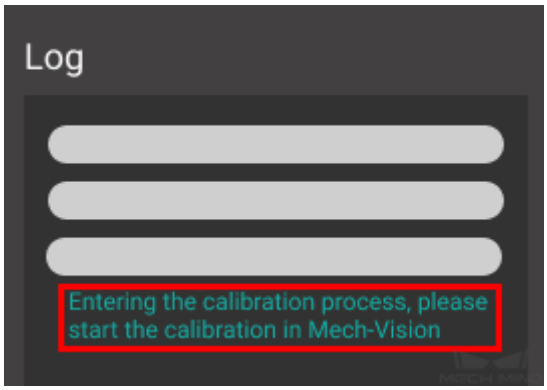
Run the Calibration Program

1. Press on  , select *Reset program*, and then press  to run the calibration program.



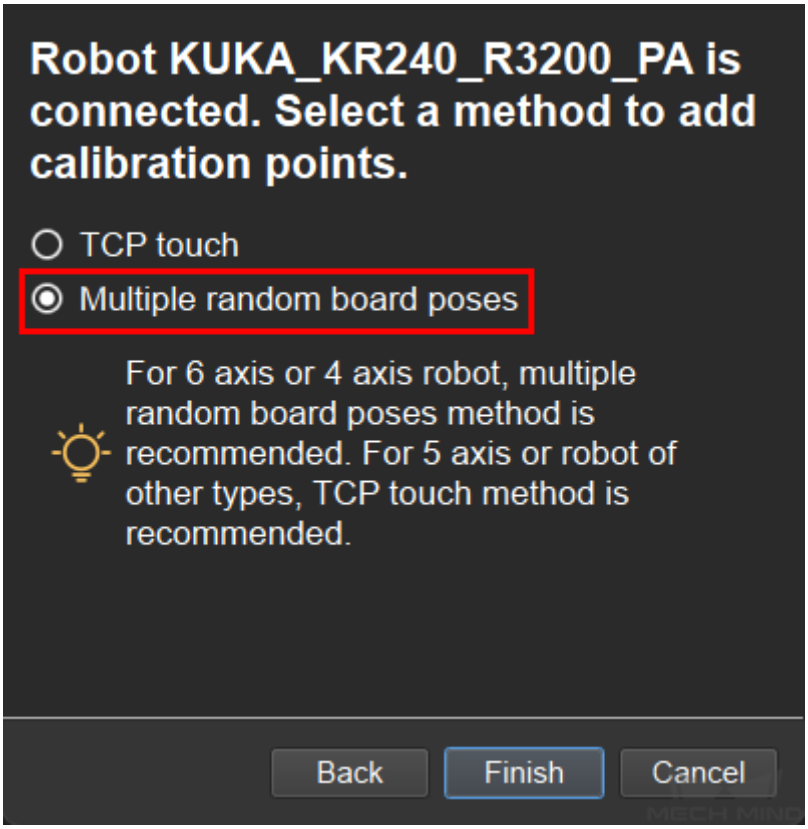
2. Proceed to the next section when the following are displayed:
   - On the teach pendant: a message saying **Calibration Start!**

- In Mech-Center **Log** panel: **Entering the calibration process, please start the calibration in Mech-Vision**

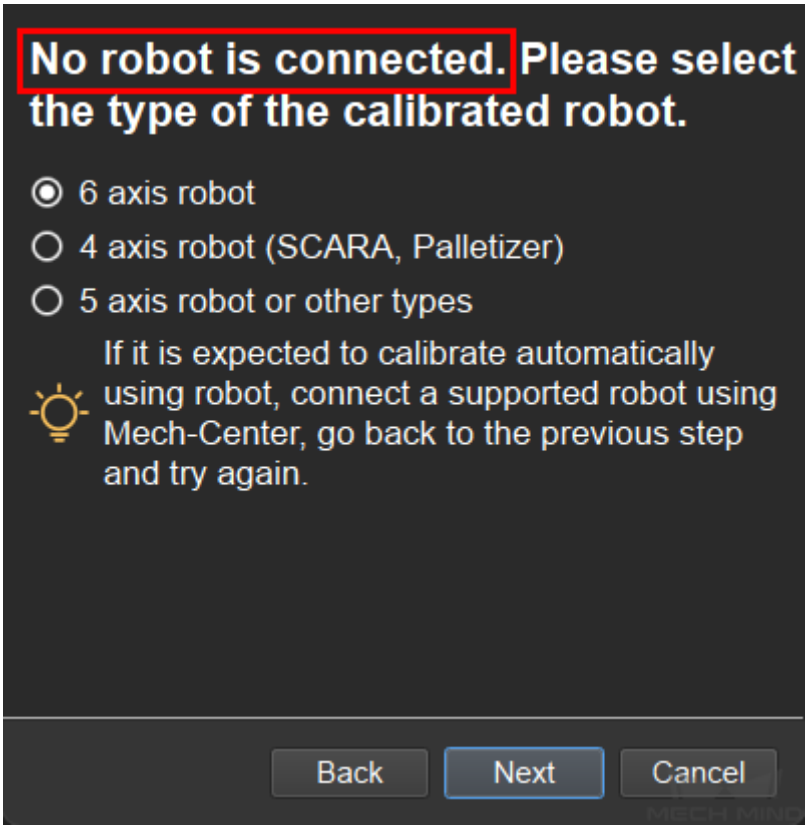


Start Calibration in Mech-Vision

1. In Mech-Vision, click on *Camera Calibration (Standard)* in the Toolbar, or select *Camera* → *Camera Calibration* → *Standard* from the Menu Bar.
2. Follow the instructions in Mech-Vision to complete the following configuration:
    1. Select **Start a new calibration process**;
    2. Select the camera mounting method;
    3. Select **Multiple random board poses** for adding calibration points.



Note: If after selecting the camera mounting method, the window says **No robot is connected**, the connection between the robot and Mech-Center is not properly established. Please re-run the robot program.



3. Follow the instructions in Mech-Vision to finish the calibration.

Note: In **5 Add Marker-Images and Poses** after you click on *Move Robot along Trajectory and Add Board Images*, if the robot does not reach the next calibration point within 60 seconds, Mech-Vision will report a timeout error and stop the calibration process. In such case, please select and run **mm_module.src** on the teach pendant again, and restart the calibration process in Mech-Vision.

2.4.3 KUKA Example Program

This section introduces the example program provided with Mech-Center and the operations required to perform an actual pick-and-place task.

The following example program files can be found in *Mech-Center/mech_interface/kuka*:

- **MM_SAMPLE01.dat**
- **MM_SAMPLE01.src**
- **MM_SAMPLE02.dat**
- **MM_SAMPLE02.src**

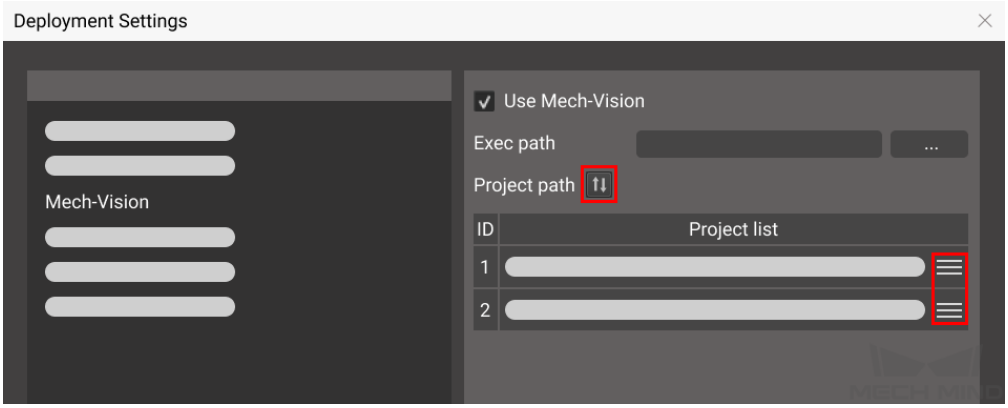
SAMPLE01 obtains vision results from Mech-Vision; **SAMPLE02** obtains planned path from Mech-Viz.

Check the section corresponding to your own application setup:

- *Obtain Vision Results from Mech-Vision*
- *Obtain Planned Path from Mech-Viz*

Before running the program, please make sure that:

- You have *loaded the Standard Interface program* onto the robot and can establish communcation with Mech-Center.
- You have completed the extrinsic parameter calibration with *the calibration program* or by manually adding calibration points.
- Mech-Vision and Mech-Viz projects are created and set to autoload.
- The **Project list** in *Mech-Center* → *Deployment Settings* → *Mech-Vision* is synced by clicking on  , and the order of Mech-Vision projects have been adjusted according to actual needs.



- The TCP has been correctly specified.
- The robot speed is set to a low value, so that the operator can notice any unexpected behavior before accidents occur.

Obtain Vision Results from Mech-Vision

```
&ACCESS RVO
&REL 5
DEF MM_SAMPLE01 ( )
  ;FUNCTION:Eye to Hand simple pick and place with Mech-Vision
  ;mechmind, 2022-5-31
  INT Job
  INT Pos_Num
  INT Last_Data
  INT MM_Status
  E6POS MM_pick
  E6POS MM_waypoint
  E6POS MM_camera_capture
```

(continues on next page)

(continued from previous page)

```
  E6POS MM_drop
  INT MM_Label
  INT MM_Speed
  ;FOLD PTP HOME Vel=100 % DEFAULT;%{PE}%R 8.3.44,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:, 5:100, 7:DEFAULT
$BWDSTART=FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS(#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
  ;ENDFOLD

  BAS(#TOOL,1)
  LIN MM_camera_capture
  ;Set ip address of IPC
  MM_Init_Socket("XML_Kuka_MMIND",873,871,60)
  wait sec 0.1
  ;Set vision recipe
  MM_Switch_Model(1,1)
  ;Run vision project
  MM_Start_Vis(1,1,2)
  wait sec 1
  MM_Get_VisData(1,Pos_Num,MM_Status)
  IF MM_Status<> 1100 THEN
    halt
  ENDIF
  MM_Get_Pose(1,MM_pick,MM_Label,MM_Speed )
MM_pick.z=MM_pick.z+100
LIN MM_pick
MM_pick.z=MM_pick.z-100
  LIN MM_pick
  ;Add object grasping logic here.
  LIN_REL {z -100}#TOOL
 LIN MM_waypoint
MM_drop.z=MM_drop.z+100
LIN MM_drop
MM_drop.z=MM_drop.z-100
  LIN MM_drop
  ;Add object releasing logic here.
  LIN_REL {z -100}#TOOL
    ;FOLD PTP HOME Vel=100 % DEFAULT;%{PE}%R 8.3.44,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:, 5:100, 7:DEFAULT
$BWDSTART=FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS(#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
  ;ENDFOLD
END
```

Program Logic

1. Move the robot to HOME position.
2. Move the robot to the image capturing pose.
3. Initialize communication with **MM_Init_Socket**.
4. If parameter recipes are used in the Mech-Vision project, the recipe to be used is set with **MM_Switch_Model**.
5. Run the Mech-Vision project with **MM_Start_Vis**.
6. Wait for 1 second. Under Eye-In-Hand, this **WAIT SEC** instruction is required to make sure the robot stays still until image acquisition is completed. Under Eye-To-Hand, this **WAIT SEC** instruction can be replaced with motion commands.
7. Obtain the vision results from Mech-Vision.
8. Check if the returned status code indicates any error. If an error code is returned, the program is halted.
9. Move the robot to the picking pose and perform picking.
10. Move the robot to a waypoint between the picking pose and placing pose.
11. Move the robot to the set placing pose and perform placing.
12. Return the robot to HOME position.

The following parts should be modified according to your actual needs.

Define the TCP

Change **BAS(#TOOL,1)** in line 25 to the tool coordinate system to which the actual TCP is saved.

Teach the Image Capturing Pose

Record the image capturing pose in **MM_camera_capture** in line 26.

Teach the Waypoint(s)

Waypoints are intermediate poses between the picking pose and placing pose. They are used to ensure that the robot doesn' t collide with the surrounding when moving between the picking and placing poses.

You can add one or more waypoints to **MM_waypoint** in line 46.

Teach the Placing Pose

Record the placing pose in **MM_drop** in line 48.

Define Z-Offset from Picking/Placing Pose

Z-offset distances relative to the tool frame from the picking/placing pose are used to ensure collision doesn't occur when the robot is approching or departing the picking/placing pose.

Adjust the following commands according to your actual needs.

- **MM_pick.z=MM_pick.z+100** in line 40: the Z-offset when approching the picking pose is set to **100**. Robot will move to 100 mm above the picking pose.
- **LIN_REL {z -100}#TOOL** in line 45: the Z-offset when departing the picking pose is set to **100**. Robot will move to 100 mm above the picking pose.
- **MM_drop.z=MM_drop.z+100** in line 47: the Z-offset when approching placing pose is set to **100**. Robot will move to 100 mm above the placing pose.
- **LIN_REL {z -100}#TOOL** in line 52: the Z-offset when departing the placing pose is set to **100**. Robot will move to 100 mm above the placing pose.

Add Object Grasping and Releasing Logics

Add logic for controlling the tool action when picking the object to line 44.

Add logic for controlling the tool action when placing the object to line 51.

Define HOME position

Teach the HOME position before running the program.

Obtain Planned Path from Mech-Viz

```
1  &ACCESS RVO
2  &REL 2
3  DEF MM_SAMPLE02 ( )
4  INT Job
5    INT Pos_Num
6    INT VisPos_Num
7    INT Last_Data
8    INT MM_Status
9    DECL E6POS MM_movepoint[20]
10   E6POS MM_waypoint
11   E6POS MM_camera_capture
12   E6POS MM_drop
13   DECL INT MM_Label[20]
14   INT MM_Speed[20]
15   INT count
16   ;FOLD PTP HOME Vel=100 % DEFAULT;%{PE}%R 8.3.44,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME,␣
↪3:, 5:100, 7:DEFAULT
```

(continues on next page)

(continued from previous page)

```
$BWDSTART=FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS(#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
  ;ENDFOLD

  BAS(#TOOL,1)
  LIN MM_camera_capture
  ;Set ip address of IPC
  MM_Init_Socket("XML_Kuka_MMIND",873,871,60)
  wait sec 0.1
  ;Set vision recipe
  ;MM_Switch_Model(1,1)
  ;Run Viz project
  MM_Start_Viz(1)
  wait sec 0.1
  ;set branch exitport
  ;MM_Set_Branch(1,1)
  ;get planned path
  MM_Get_VizData(2,Last_Data,Pos_Num,VisPos_Num,MM_Status)
  IF MM_Status<> 2100 THEN
    halt
  ENDIF
  FOR count=1 TO Pos_Num
  MM_Get_Pose(count,MM_movepoint[count],MM_Label[count],MM_Speed[count])
  ENDFOR
  ;follow the planned path to pick
  FOR count=1 TO Pos_Num
  LIN MM_movepoint[count]
  IF count==VisPos_Num THEN
   ;add object grasping logic here
  ENDIF
  ENDFOR
  ;go to drop location
  LIN MM_waypoint
MM_drop.z=MM_drop.z+100
  LIN MM_drop
MM_drop.z=MM_drop.z-100
  LIN MM_drop
  ;Add object releasing logic here.
  LIN_REL  {z -100}#TOOL
    ;FOLD PTP HOME Vel=100 % DEFAULT;%{PE}%R 8.3.44,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP,␣
↪2:HOME, 3:, 5:100, 7:DEFAULT
$BWDSTART=FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS(#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
  ;ENDFOLD
END
```

Program Logic

1. Move the robot to HOME position.
2. Move the robot to the image capturing pose.
3. Initialize communication with **MM_Init_Socket**.
4. If parameter recipes are used in the Mech-Vision project, the recipe to be used is set with **MM_Switch_Model**.
5. Run the Mech-Viz project with **MM_Start_Viz**.
6. Obtain the planned path from Mech-Viz.
7. Check if the returned status code indicates any error. If an error code is returned, the program is halted.
8. Store obtained target points in the planned path to **MM_movepoint**[] with a FOR loop.
9. Move the robot along the planned path with a FOR loop and perform picking.
10. Move the robot to a waypoint between the picking pose and placing pose.
11. Move the robot to the set placing pose and perform placing.
12. Return the robot to HOME position.

The following parts should be modified according to your actual needs.

Define the TCP

Change **BAS(#TOOL,1)** in line 25 to the tool coordinate system to which the actual TCP is saved.

Teach the Image Capturing Pose

Record the image capturing pose in **MM_camera_capture** in line 26.

Teach the Waypoint(s)

Waypoints are intermediate poses between the picking pose and placing pose. They are used to ensure that the robot doesn' t collide with the surrounding when moving between the picking and placing poses.

You can add one or more waypoints to **MM_waypoint** in line 53.

Teach the Placing Pose

Record the placing pose in **MM_drop** in line 55.

Add Object Grasping and Releasing Logics

Add logic for controlling the tool action when picking the object to line 49.

Add logic for controlling the tool action when placing the object to line 58.

Define HOME position

Teach the HOME position before running the program.

2.4.4 KUKA Standard Interface Commands

The KUKA Standard Interface provides the following subprograms:

- *Initialize Communication*
- *Start Mech-Vision Project*
- *Get Vision Result*
- *Start Mech-Viz Project*
- *Get Planned Path*
- *Obtain Pose*
- *Obtain Joint Positions*
- *Switch Mech-Vision Recipe*
- *Select Mech-Viz Branch*
- *Set Move Index*
- *Get Software Status*
- *Input Object Dimensions to Mech-Vision*
- *Get DO Signal List*
- *Input TCP to Mech-Viz*
- *Calibration*

When writing your own program, pay attention to the following:

- Multiple parameters should be separated by commas.
- All parameters should be defined as runtime variables in the program file.
- Parameters can be defined as IN or OUT parameters.

This Standard Interface is over the TCP/IP protocol.

Initialize Communication

```
MM_Init_Socket(XML_Name[]:IN,Alive_Flag:IN,Recv_Flag:IN,Time_Out:IN)
```

This subprogram sets the name of the XML file used for setting up the TCP/IP communication, flag for successful communication, flag for successful data reception, and wait time before the program stops trying to establish the communication.

Parameters

- IN parameters

Name	Description
XML_Name[]	Name of the XML file, case-sensitive
Alive_Flag	ALIVE flag number in the XML file
	When the flag is set to ON, the communication is successfully established
Recv_Flag	RECEIVE flag number in the XML file
	When the flag is set to ON, the robot has successfully received data
Time_Out	Wait time in seconds before stopping connection attempt

Example

```
MM_Init_Socket("XML_Kuka_MMIND",873,871,60)
```

This example sets the XML file for setting up communication as *XML_Kuka_MMMIND*, the flag for successful communication as 873, the flag for successful data reception as 871, and wait time as 60 seconds.

Start Mech-Vision Project

```
MM_Start_Vis(Job:IN,Pos_Num_Need:IN,SendPos_Type:IN)
```

This subprogram is for applications that use Mech-Vision but not Mech-Viz. It runs the corresponding Mech-Vision project to acquire and process data.

Parameters

- IN parameters

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center → Deployment Settings → Mech-Vision*
Pos_Num_Need	Number of poses for Mech-Vision to send, from 0 to 20, where 0 means "obtain all".
SendPos_Type	Set the image capturing pose for the robot to send, from 0 to 2
	0: Do not send image capturing pose (for Eye To Hand) 1: Send image capturing pose as joint positions 2: Send image capturing pose as robot flange pose

Example

```
MM_Start_Vis(1,1,1)
```

This example runs Mech-Vision project No. 1, and asks the project to send over 1 pose, and the robot sends its joint positions when this subprogram is called as the image capturing pose to Mech-Center.

Get Vision Result

```
MM_Get_VisData(Job:IN,Last_Data:OUT,Pos_Num:OUT,MM_Status:OUT)
```

This subprogram is for applications that use Mech-Vision but not Mech-Viz. It obtains the vision result from the corresponding Mech-Vision project.

Parameters

- IN parameter

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center* → *Deployment Settings* → *Mech-Vision*

- OUT parameters

Name	Description
Last_Data	Variable, indicating whether all vision result has been sent, 0 or 1
	0: NOT all vision result has been sent (more on the way) 1: All vision result has been sent
Pos_Num	Variable for storing the number of received poses
MM_Status	Variable for storing status code, refer to the standard_interface_status_codes

Example

```
MM_Get_VisData(1,Last,Pose_Num,Status)
```

This example obtains the vision result from Mech-Vision project No. 1. Whether all vision result has been sent is stored in **Last**, the number of poses received is stored in **Pose_Num**, and the status code is stored in **Status**.

Start Mech-Viz Project

```
MM_Start_Viz(SendPos_Type:IN)
```

This subprogram is for applications that use both Mech-Vision and Mech-Viz. It runs the corresponding Mech-Viz project (which triggers the corresponding Mech-Vision project to run), and sets the initial joint positions of the simulated robot in Mech-Viz.

Parameter

- IN parameter

Name	Description
SendPos_Type	The initial joint positions for the simulated robot in Mech-Viz, 0 or 1
	0: Set the initial joint positions of the simulated robot to [0,0,0,0,0,0] 1: Set the initial joint positions of the simulated robot to the current joint positions of the real robot

Note: When the scene contains object models that obstruct the robot to move from [0,0,0,0,0,0] to the first target point, this parameter must be set to 1.

Example

```
MM_Start_Viz(1)
```

This example runs the corresponding Mech-Viz project, and sets the initial joint positions of the simulated robot to the current joint positions of the real robot.

Get Planned Path

```
MM_Get_VizData(Jps_Pos:IN,Last_Data:OUT,Pos_Num:OUT,VisPos_Num:OUT,MM_Status:OUT)
```

This subprogram obtains the planned path from Mech-Viz.

Parameters

- IN parameter

Name	Description
Jps_Pos	Whether Mech-Viz should send target points as joint positions or TCPs, 1 or 2
	1: Mech-Viz sends joint positions 2: Mech-Viz sends TCPs

- OUT parameters

Name	Description
Last_Data	Variable, indicating whether all target points have been sent, 0 or 1
	0: NOT all target points have been sent (more on the way) 1: All target points have been sent
Pos_Num	Variable for storing the number of received target points
VisPos_Num	Variable for storing the position of the first visual_move target point in the path
	Example path: move-1, move-2, visual_move-3, move-3, visual_move-2 In this path, the position of the first visual_move target point is 3. If the path does not contain visual_move target point, the return value is 0.
MM_Status	Variable for storing status code, refer to the standard_interface_status_codes

Example

```
MM_Get_VizData(2,Last,Pose_Num,VisPose_Num,Status)
```

This example obtains the planned path from Mech-Viz in the form of TCPs. Whether all target points have been sent is stored in **Last**, the number of target points received is stored in **Pose_Num**, the position of the visual_move target point is stored in **VisPose_Num**, and the status code is stored in **Status**.

Obtain Pose

```
MM_Get_Pose(Serial:IN,MM_P:OUT,MM_Label:OUT,MM_Speed:OUT)
```

This subprogram stores a pose returned by Mech-Vision or a target point (as TCP) returned by Mech-Viz in the specified variable.

Parameters

- IN parameter

Name	Description
Serial	Specify the index of the pose to be stored

- OUT parameters

Name	Description
MM_P	Variable for storing the specified pose
MM_Label	Variable for storing the label corresponding to the specified pose
MM_Speed	Variable for storing the speed corresponding to the specified pose

Example

```
MM_Get_Pose(1,XP1,Label,Pose_Speed)
```

This example stores the first received pose to **XP1**, the corresponding label to **Label**, and the corresponding speed to **Pose_Speed**.

Obtain Joint Positions

```
MM_Get_Jps(Serial:IN,MM_J:OUT,MM_Label:OUT,MM_Speed:OUT)
```

This subprogram stores a set of joint positions returned by Mech-Viz in the specified variable.

Note: As Mech-Vision does not output joint position data, this subprogram can only be used with Mech-Viz.

Parameters

- IN parameter

Name	Description
Serial	Specify the index of the set of joint positions to be stored

- OUT parameters

Name	Description
MM_J	Variable for storing the specified set of joint positions
MM_Label	Variable for storing the label corresponding to the specified set of joint positions
MM_Speed	Variable for storing the speed corresponding to the specified set of joint positions

Example

```
MM_Get_Jps(1,JP1,Label,Pose_Speed)
```

This example stores the first set of received joint positions to **JP1**, the corresponding label to **Label**, and the corresponding speed to **Pose_Speed**.

Switch Mech-Vision Recipe

```
MM_Switch_Model(Job:IN,Model_Number:IN)
```

This subprogram specifies which parameter recipe of the Mech-Vision project to use. For more information on parameter recipe, please see parameter_recipe_configuration.

Note:

- This subprogram must be called BEFORE **MM_Start_Vis**.
- The corresponding Mech-Vision project must have parameter recipes already configured and saved.

Parameters

- IN parameters

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center* → *Deployment Settings* → *Mech-Vision*
Model_Number	The number of a parameter recipe in the Mech-Vision project, from 1 to 99

Example

```
MM_Switch_Model(2,2)
```

This example switches the parameter recipe used to No. 2 in Mech-Vision project No. 2.

Select Mech-Viz Branch

```
MM_Set_Branch(Branch_Num:IN,Export_Num:IN)
```

This subprogram is used to select along which branch the Mech-Viz project should proceed. Such branching is achieved by adding branch_by_service_message Task(s) to the project. This subprogram specifies which out port such Task(s) should take.

Note:

- **MM_Start_Viz** must be called BEFORE this subprogram.
- When the next Task to be executed in Mech-Viz is a **branch_by_service_message** Task, Mech-Viz will wait for this subprogram to send the out port number it should take.
- The name of all **branch_by_service_message** Tasks in the Mech-Viz project must be changed to numbers between 1 and 99, and the names should be unique among all tasks in the project.

Parameters

- IN parameters

Name	Description
Branch_Num	Name of the **branch_by_service_message** Task, from 1 to 99
Export_Num	The number of the out port to take, from 1 to 99

Example

```
MM_Set_Branch(1,3)
```

This example tells Mech-Viz to take out port 3 for the **branch_by_service_message** Task named **1**.

Set Move Index

```
MM_Set_Index(Skill_Num:IN,Index_Num:IN)
```

This subprogram sets the value for the Current Index parameter of Mech-Viz Tasks. Tasks that have this parameter include move_list, move_grid, custom_pallet_pattern, and smart_pallet_pattern.

Note:

- **MM_Start_Viz** must be called BEFORE this subprogram.
- The name of all Tasks with index parameters in the Mech-Viz project must be changed to numbers between 1 and 99, and the names should be unique among all tasks in the project.

Parameters

- IN parameters

Name	Description
Skill_Num	Name of the Task, from 1 to 99
Index_Num	Value for the Current Index parameter when the Task is executed

Example

```
MM_Set_Index(2,10)
```

This example sets the Current Index value to 9 for the Task named **2**. When the Task is executed, the Current Index value will be added 1 and become 10.

Get Software Status

```
MM_Get_Status(MM_Status:OUT)
```

This subprogram is currently capable of checking whether Mech-Vision is ready to run projects. In the future, this subprogram can be used for obtaining the execution status of Mech-Vision, Mech-Viz and Mech-Center.

Parameter

- OUT parameter

Name	Description
MM_Status	Variable for storing the status code, refer to the standard_interface_status_codes

Example

```
MM_Get_Status(Status)
```

This example obtains the status code and stores it in **Status**.

Input Object Dimensions to Mech-Vision

```
MM_Set_BoxSize(Job:IN,Lenght:IN,Width:IN,Height:IN)
```

This subprogram inputs object dimensions to the Mech-Vision project.

Note:

- This subprogram must be called BEFORE **MM_Start_Vis**.

Parameters

- IN parameters

Name	Description
Job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center* → *Deployment Settings* → *Mech-Vision*
Lenght	Length of object in mm
Width	Width of object in mm
Height	Height of object in mm

Example

```
MM_Set_BoxSize(1,500,300,200)
```

This example sets the object dimensions in the read_object_dimensions Step in the Mech-Vision project No. 1 to 500*300*200 mm.

Get DO Signal List

```
MM_Get_DoList()
MM_Set_DoList()
```

These two subprograms obtain the planned DO Signal list for controlling multiple sections of a sectioned vacuum gripper and set the DOs accordingly.

Note:

- **MM_Get_VizData** must be called BEFORE this subprogram.
- Please deploy the Mech-Viz project based on the template project in *Mech-Center/tool/viz_project/suction_zone*, and set the suction cup configuration file in the Mech-Viz project.

Parameters

No parameters.

Example

```
MM_Get_DoList()
MM_Set_DoList()
```

These two examples obtain the DO signal list planned by Mech-Viz, store it in **Do_Port[i]**, and input the values to the corresponding digital outputs.

Input TCP to Mech-Viz

```
MM_Set_Pos(Out_Pos:IN)
```

This subprogram inputs TCP data to the outer_move Task.

Note:

- This subprogram must be called BEFORE **MM_Start_Viz**.
- Please deploy the Mech-Viz project based on the template project in *Mech-Centertoolviz_projectouter_move*, and put the **outer_move** Task at a proper position in the workflow.

Parameter

- IN parameter

Name	Description
Out_Pos	Variable for storing the TCP data to be sent to Mech-Viz

Example

```
MM_Set_Pos(XP50)
```

This example sends the TCP data stored in **XP50** to the **outer_move** task in the Mech-Viz project.

Calibration

```
MM_CALIB(Move_Type:IN,PosJps:IN,WaitTime:IN,E1:IN)
```

This subprogram is used for hand-eye calibration (camera extrinsic parameter calibration). It automates the calibration process in conjunction with the **Camera Calibration** function in Mech-Vision. For detailed instructions, see *KUKA Calibration Program*.

Parameters

- IN parameters

Name	Description
Move_Type	Motion type, 1 or 2
	1: LIN 2: PTP
PosJps	Pose as flange pose or joint positions, 1 or 2
	1: flange pose 2: Joint positions
WaitTime	Wait time between poses in seconds
E1	Data of the external 7th axis in mm

Example

```
MM_CALIB(2,1,300,0)
```

This example moves the robot in PTP type, receives pose data in the form of flange pose, and sets the wait time between two poses to 300 seconds. This robot does not have an external axis installed.

2.4.5 KUKA Error Messages

The following errors may occur while running the Standard Interface program on the robot.

ROBOT_SOCKET_TIMEOUT

MM_Flag_Recv is called to check data reception. This error is reported if no data are received within the specified wait time.

Troubleshooting

- Check if the order of calling **EKI_Send** and **MM_Flag_Recv** is correct.
- Check if the Standard Interface is started in Mech-Center.
- Check if the wait time input through the **MM_Init_Socket** command is too short.

ROBOT_SOCKET_CLOSED

MM_Flag_Alive detected that communication could not be established within the specified wait time.

Troubleshooting

- Check if the hardware are properly connected.
- Check if the Standard Interface is started in Mech-Center.
- Check the IP addresses of the robot and the IPC, and if the port number is configured correctly.
- Check if the firewall is turned off on the IPC.
- Contact Mech-Mind Technical Support for further assistance.

ROBOT ARGUMENT ERROR

When calling a Mech-Mind Standard Interface subprogram, arguments provided are not correct.

Troubleshooting

Please refer to *KUKA Standard Interface Commands* and input the correct arguments accordingly.

ROBOT_CMD_ERROR

The command code received does not match the one sent.

Troubleshooting

The sequence of command sending and receiving is problematic. Please contact Mech-Mind Technical Support for further assistance.

Error Codes

Returned status code is an error code. Please check Mech-Center' s log.

Troubleshooting

- Please refer to the standard_interface_status_codes for the specific error.
- Please contact Mech-Mind Technical Support for further assistance.

2.5 Kawasaki

This section introduces the Standard Interface for Kawasaki robots.

2.5.1 Kawasaki Setup Instructions

This section introduces the process of loading the Standard Interface program onto a Kawasaki robot.

The process consists of 4 steps:

- *Check Controller and Software Compatibility*
- *Setup the Network Connection*
- *Load the Program Files*
- *Test Robot Connection*

Please have a flash drive ready at hand.

Note: A USB 2.0 flash drive is recommended. Otherwise, the robot controller may not recognize the flash drive.

Check Controller and Software Compatibility

- Controller: E and F series
- Controller system software version: no requirement
- Additional controller software options: not required
- Mech-Center: latest version recommended

Setup the Network Connection

Hardware Connection

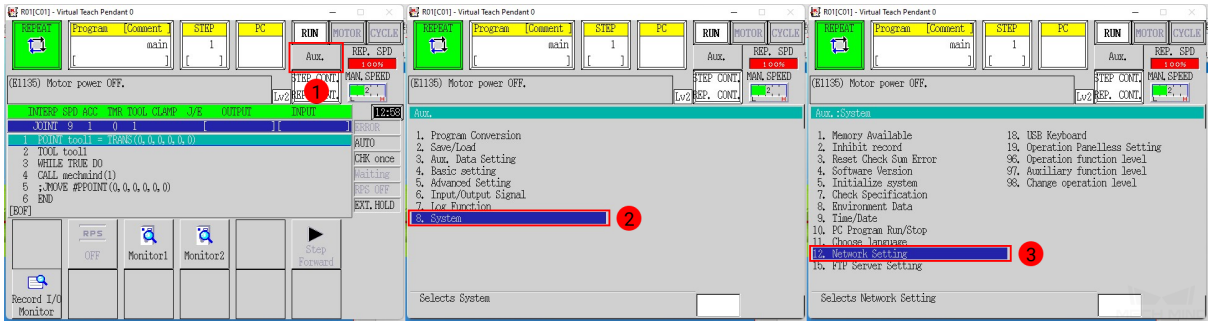
Plug the Ethernet cable into:

- An Ethernet port on the IPC
- The Ethernet port inside the accessory panel on the front of the controller

IP Configuration

To allow communication between the IPC and the robot controller, both must have an IP address in the same subnet. This means that the first three numbers of the IP addresses should be the same. For example, 192.168.100.1 and 192.168.100.2 are in the same subnet.

1. Check the IP address of the IPC: please use the *ipconfig* command in Command Prompt or PowerShell to check the IP address.
2. On the teach pendant, press on *Aux.*, and then select *8. System* → *12. Network Setting*.

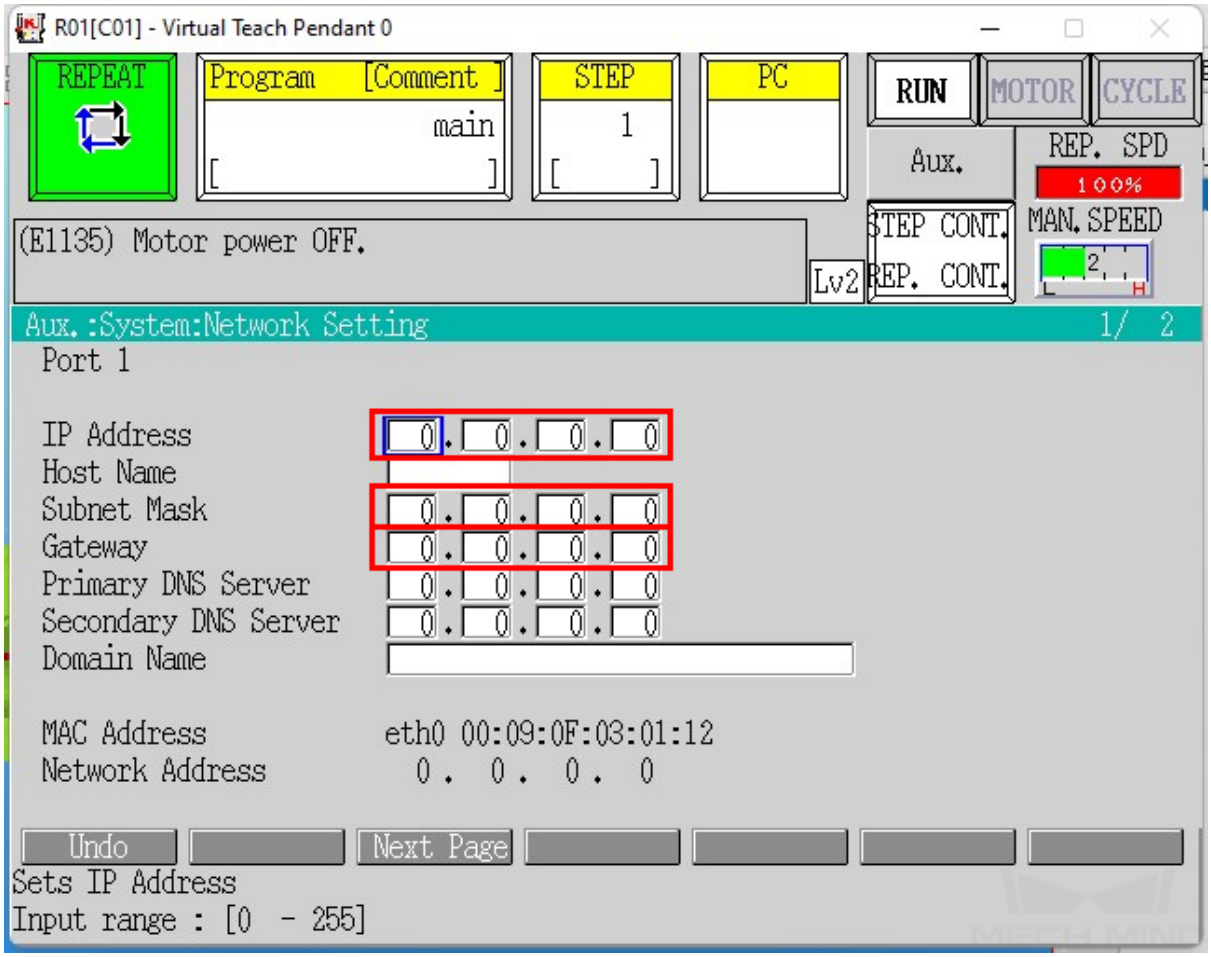


3. Set **IP Address** to one in the same subnet as that of the IPC.
4. Set **Subnet Mask** to **255.255.255.0**.

Note: If the IP address is set to either 192.168.0.1 or 192.168.1.1, please set **Subnet Mask** to **255.255.0.0** instead.

---

5. If you are using a network gateway, the gateway address should also be set. Please consult your IT support for help.



6. Press the `ENTER` key to confirm, and then restart the controller.

Load the Program Files

Prepare the Files

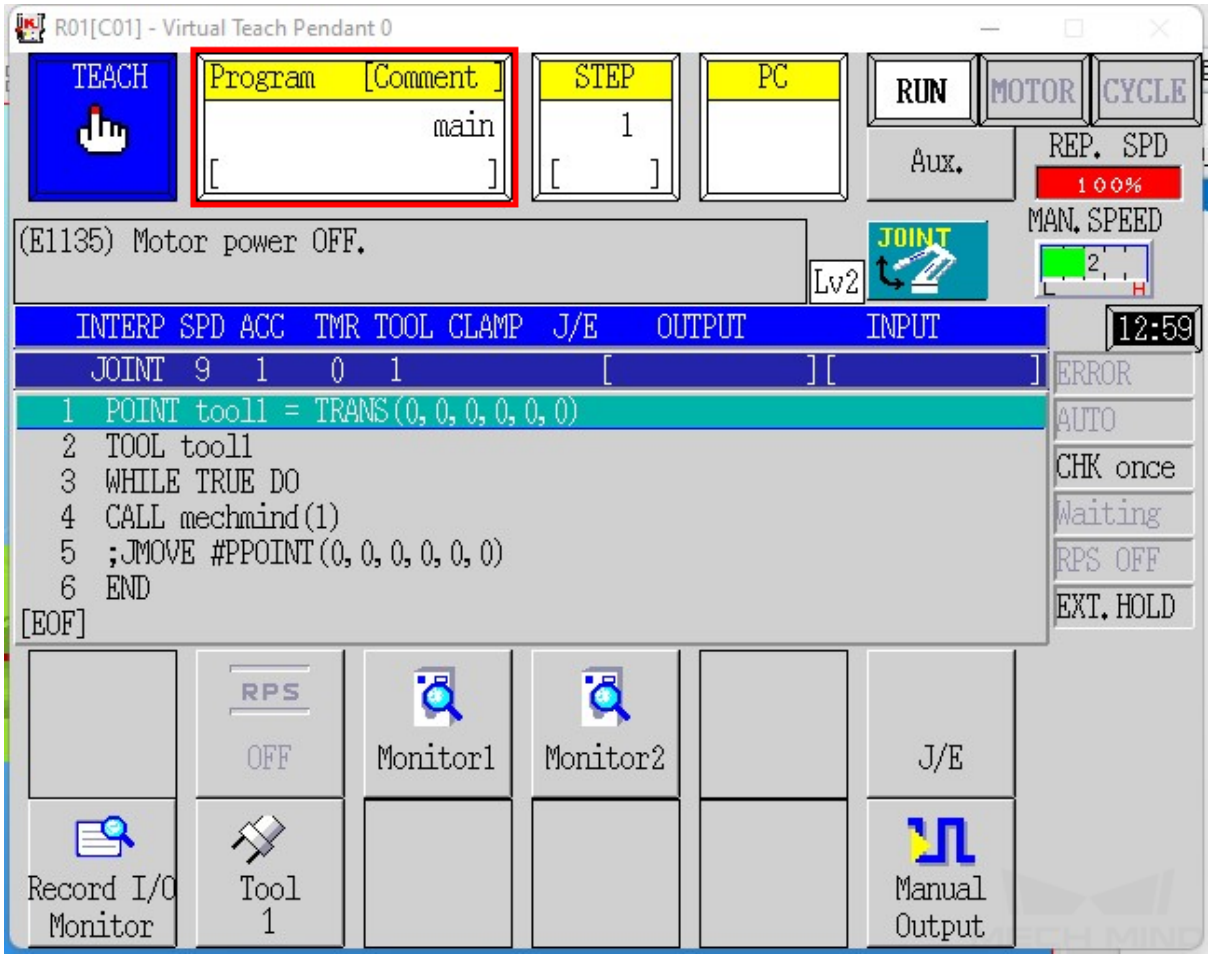
The program files are stored in the installation directory of Mech-Center. The default directory is *C:/Mech-Mind/Mech-Center*.

Navigate to *xxx/Mech-Center/mech_interface/kawasaki*, and copy **mm_module.as** to your flash drive.

Note: Copy the file to the root directory of the flash drive. Do not put it in another folder or rename it.

Load the Files to the Robot

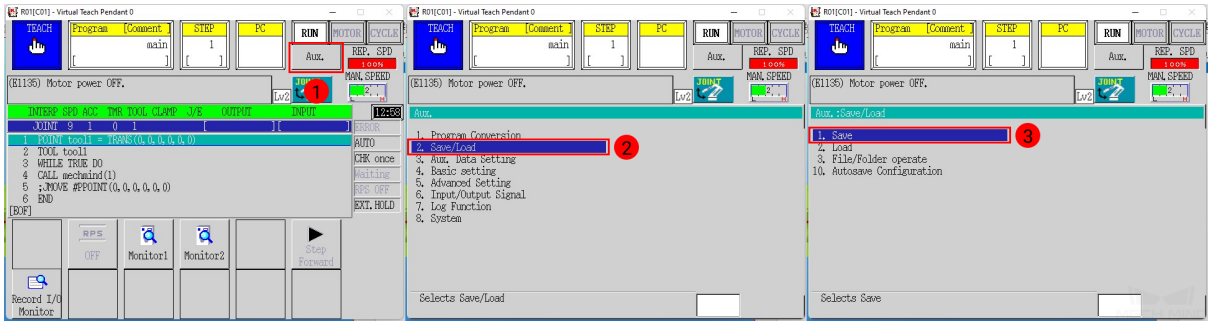
1. Insert the flash drive to the USB port inside the accessory panel on the controller.
2. Check the **Program** area to see if any robot control program is running. If so, backup and exit the program according to the following instructions.



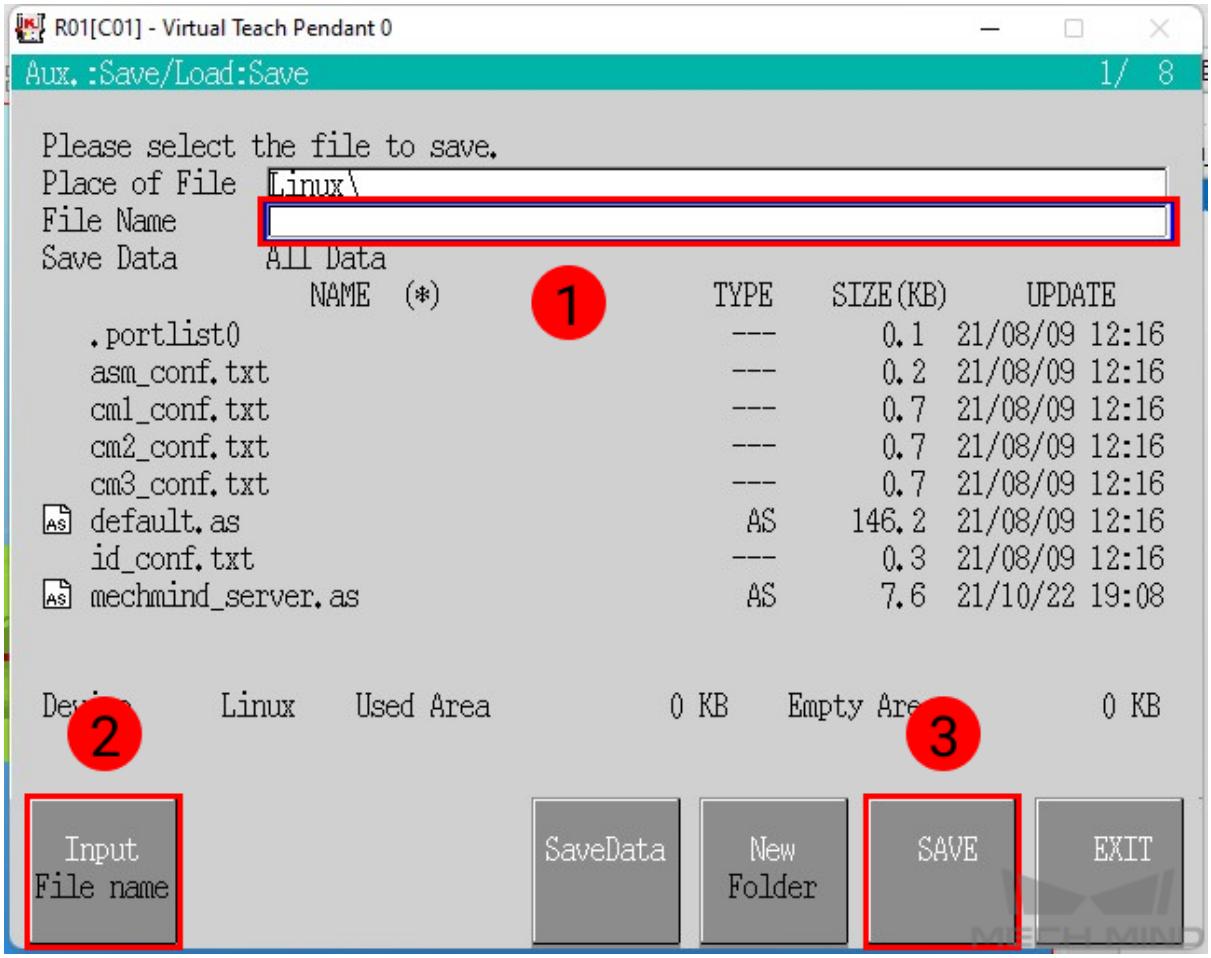
If there is a program in the **Program** area, please create backup first.

Follow the steps below to back up all files except for system logs to the flash drive.

1. Press on *Aux.*, and select *2. Save/Load* → *1. Save*.

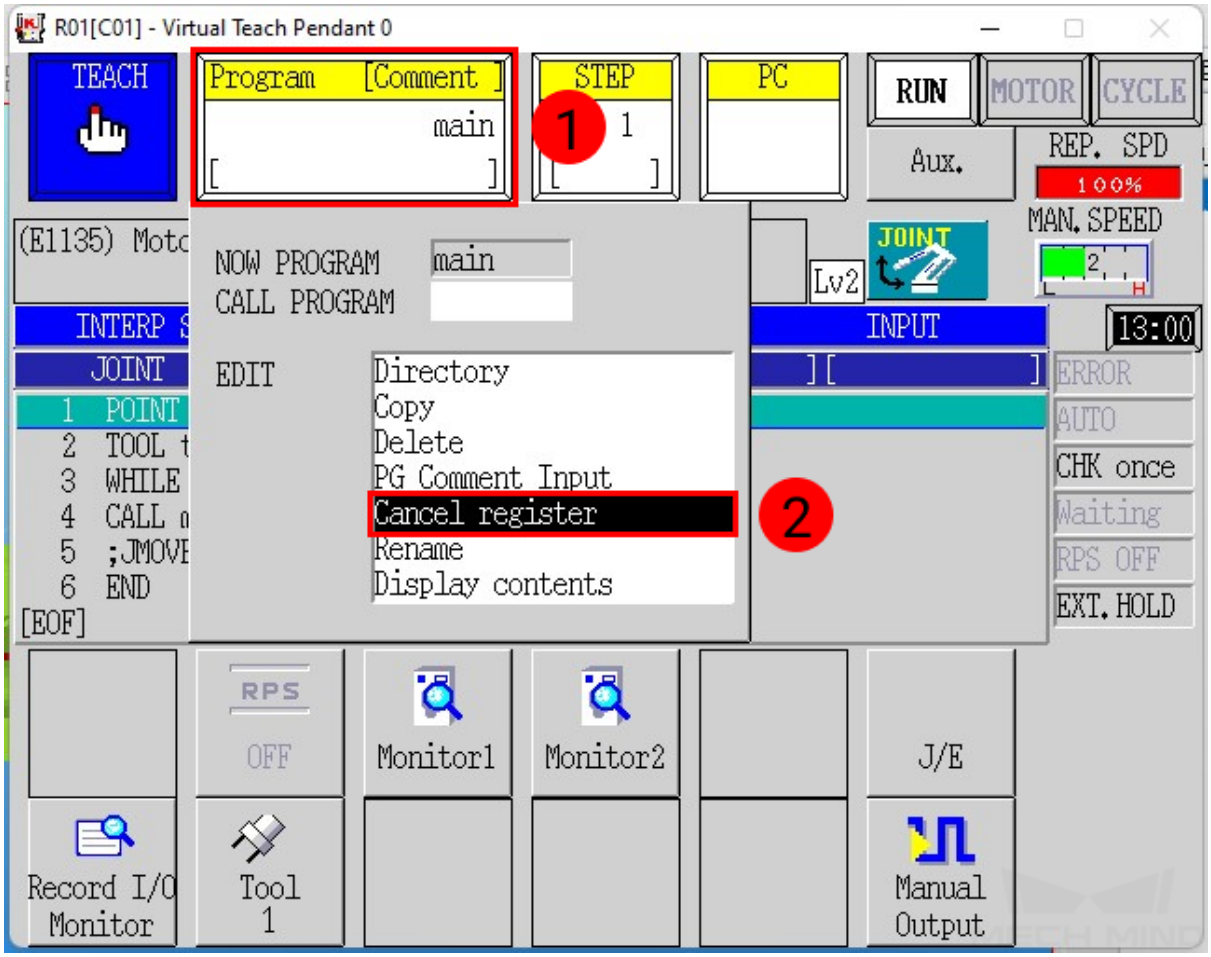


2. Press on *Input File name* to input a **File Name** for the backup file, and then press on *SAVE*.

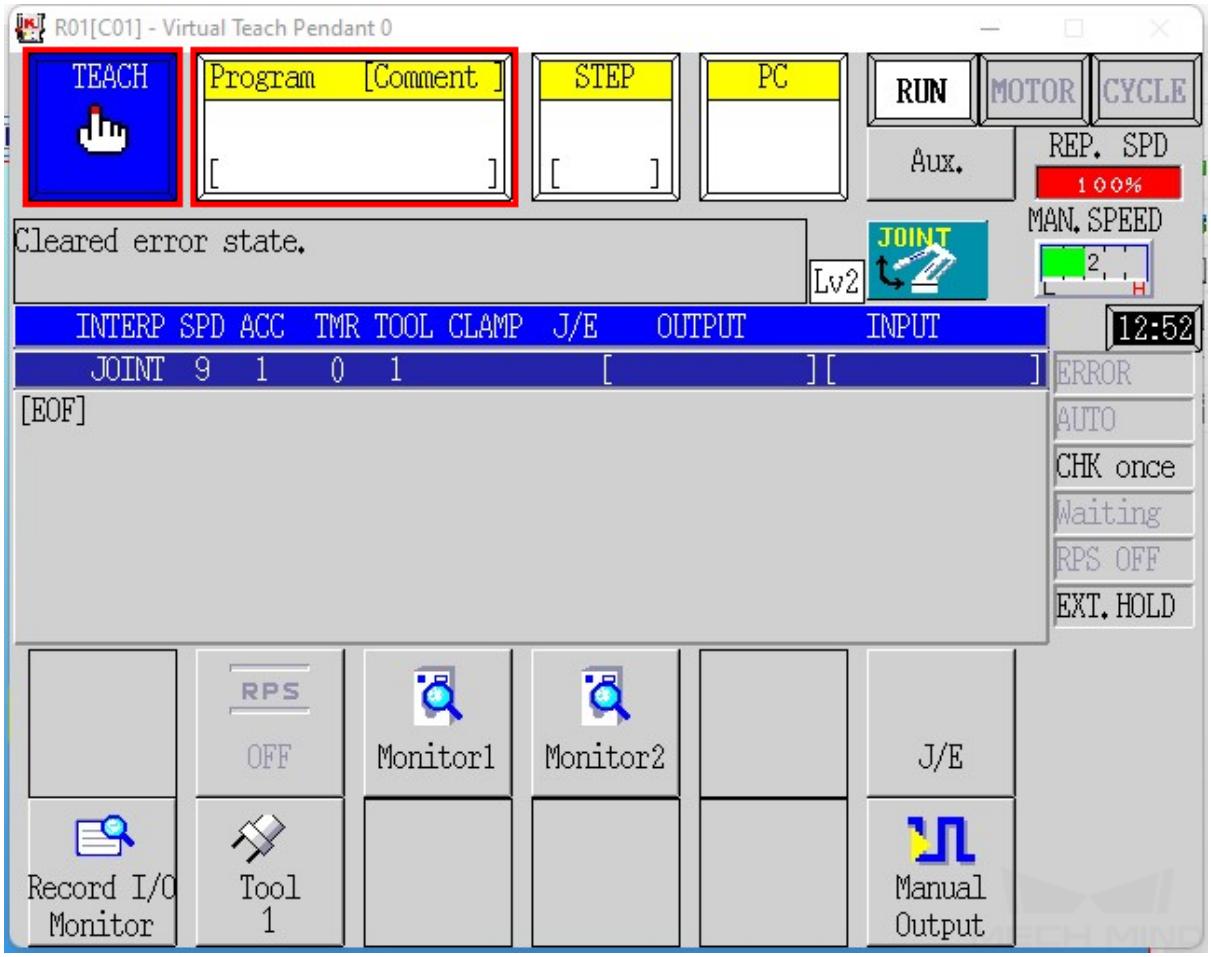


Cancel robot control program.

1. Press on the **Program** area, and select **Cancel register** in the drop down menu.

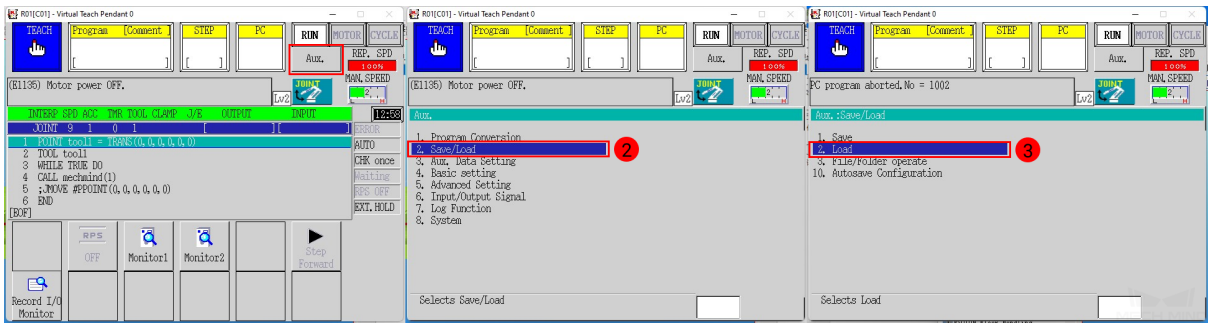


3. Make sure that the robot is in teach mode, and make sure that the **Program** area has nothing listed.

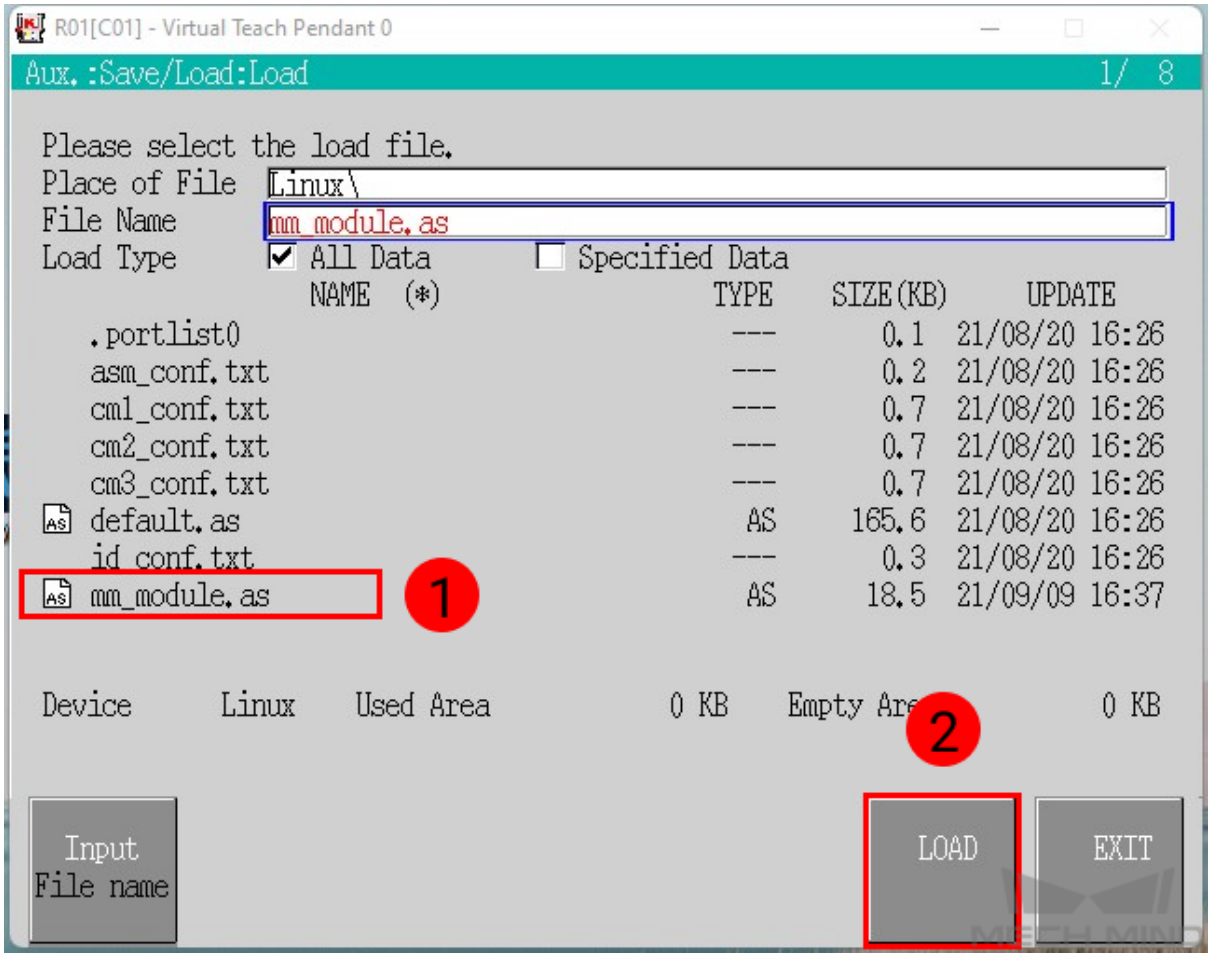


Note: To switch the robot to teach mode, turn the Teach/Repeat switch on the controller to **TEACH**, and the teach lock switch on the teach pendant to **ON**.

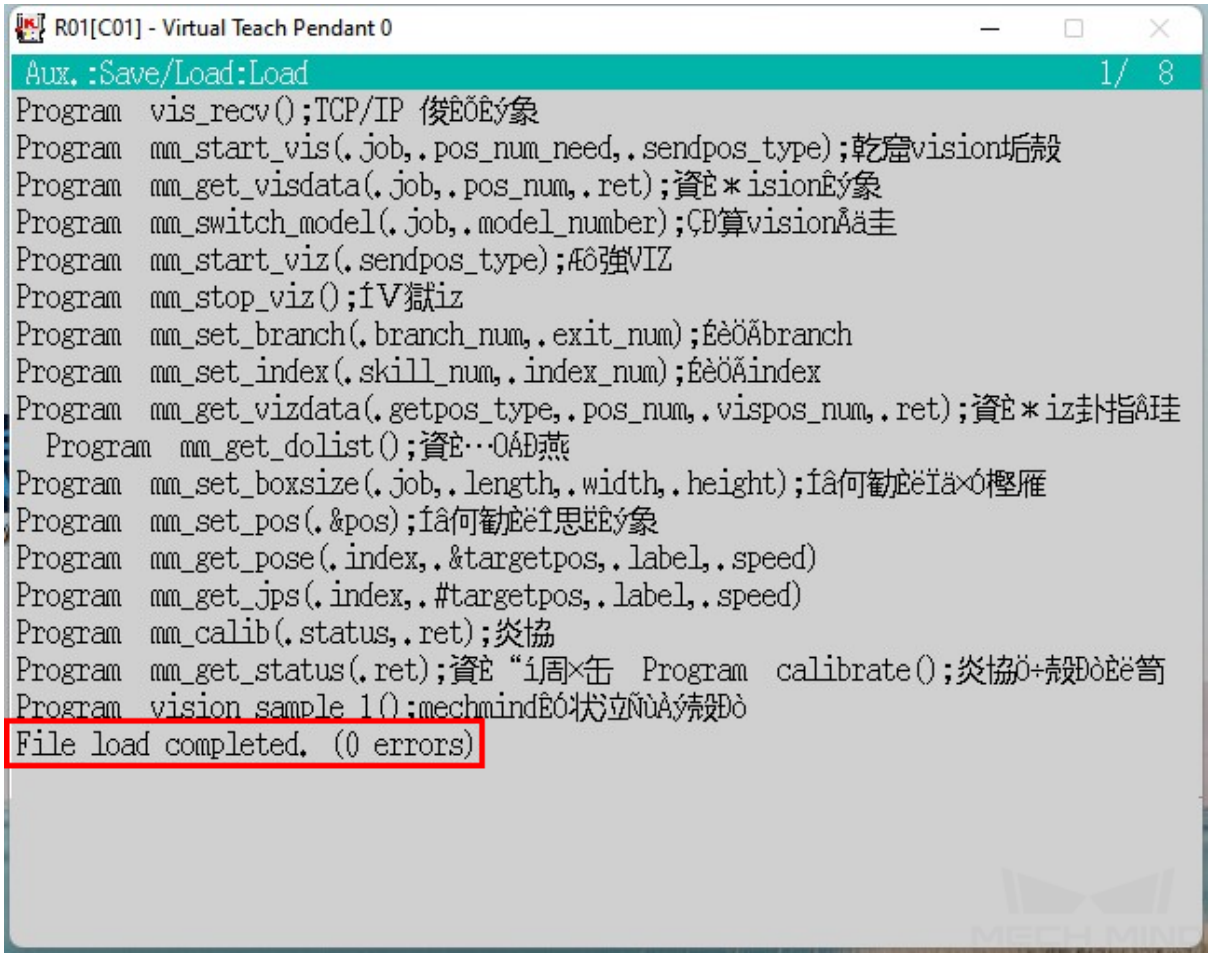
4. Press on *Aux.*, and select *2. Save/Load* → *2. Load*.



5. Press on **mm_module.as** in the file list twice to select it, and then press on *LOAD*.



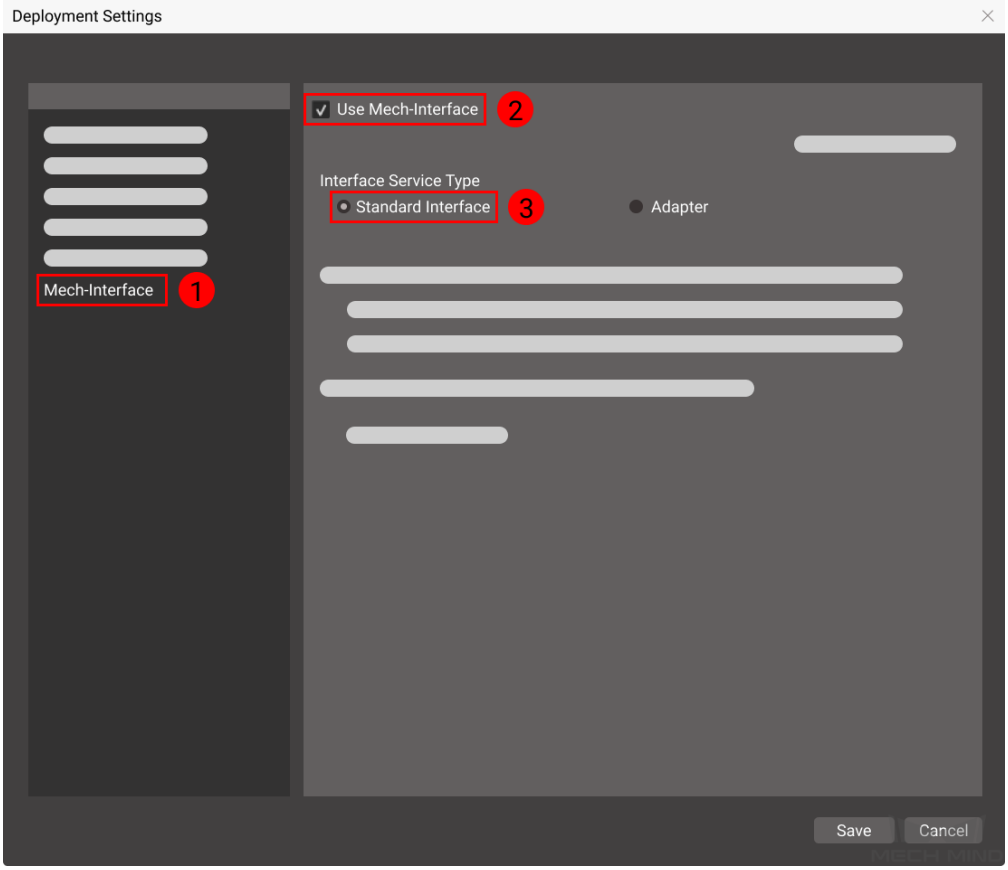
6. When loading completes, make sure no errors occurred during loading, and press on the R key to exit.



Test Robot Connection

Configure Mech-Interface in Mech-Center

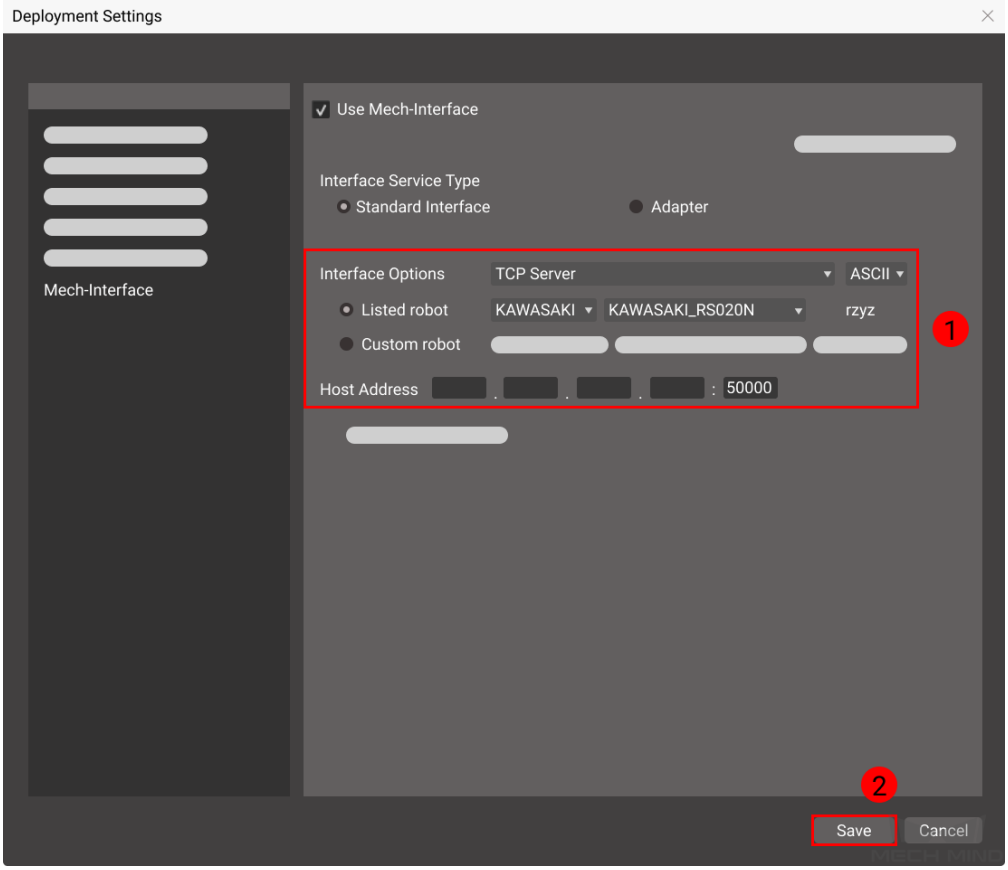
1. Open Mech-Center and click on *Deployment Settings*.
2. Go to **Mech-Interface**, check **Use Mech-Interface** and select **Standard Interface**.



3. Set the following fields:
   - **Interface Option**: Set to **TCP Server** and **ASCII**.
   - **Listed robot**: Select the robot model you are using.

     Note: If the robot you are using is not listed, select **Custom robot** instead. Make sure to set Euler angle order to **rzyz**.

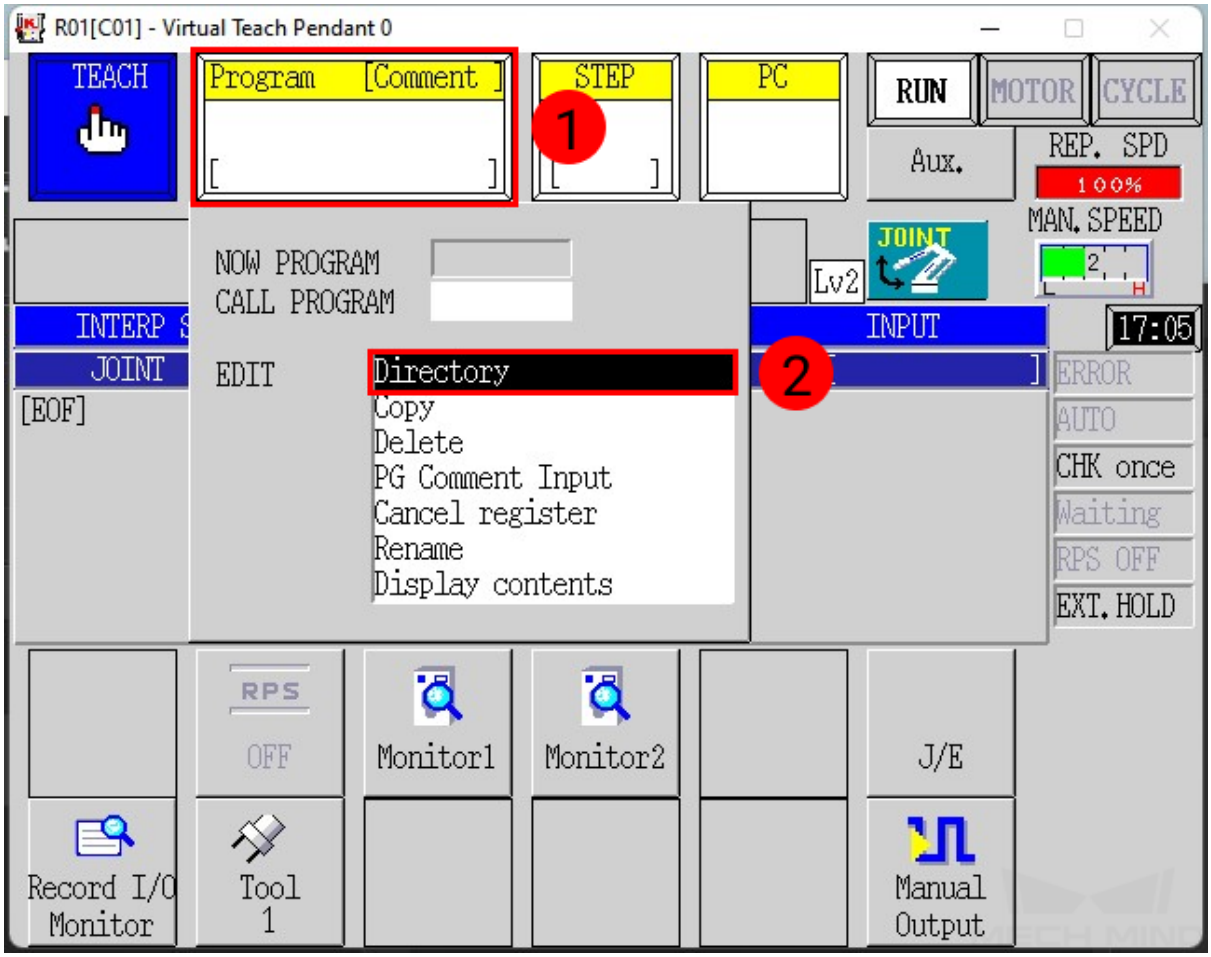
   - **Host Address**: The default port number is **50000**. If you need to change the port number, make sure it falls between 8192 and 65535.
4. Click on *Save*.



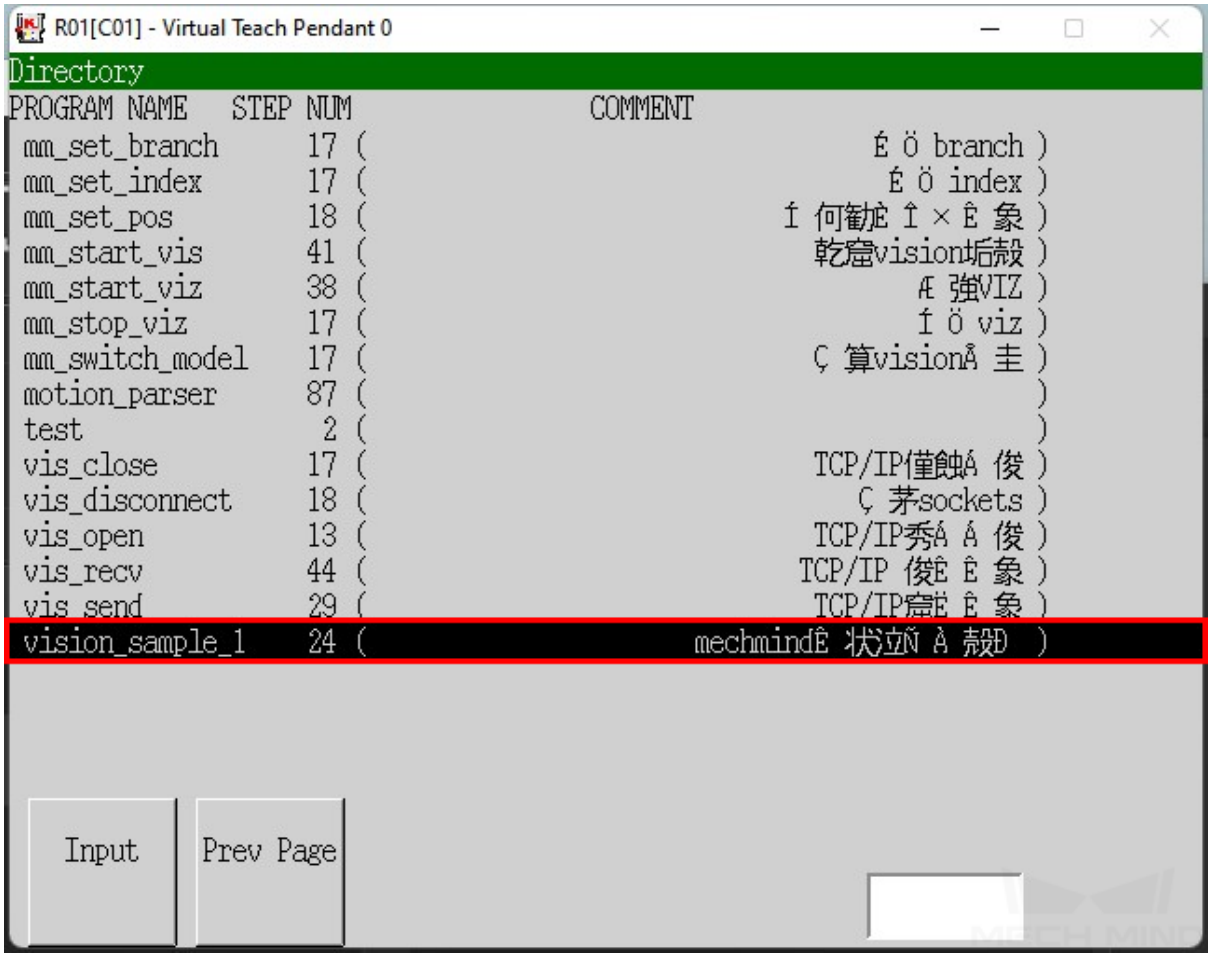
5. Click on *Start Interface* in the Toolbar.

Modify and Run Robot Program

1. On the teach pendant, press on the **Program** area, select **Directory**.

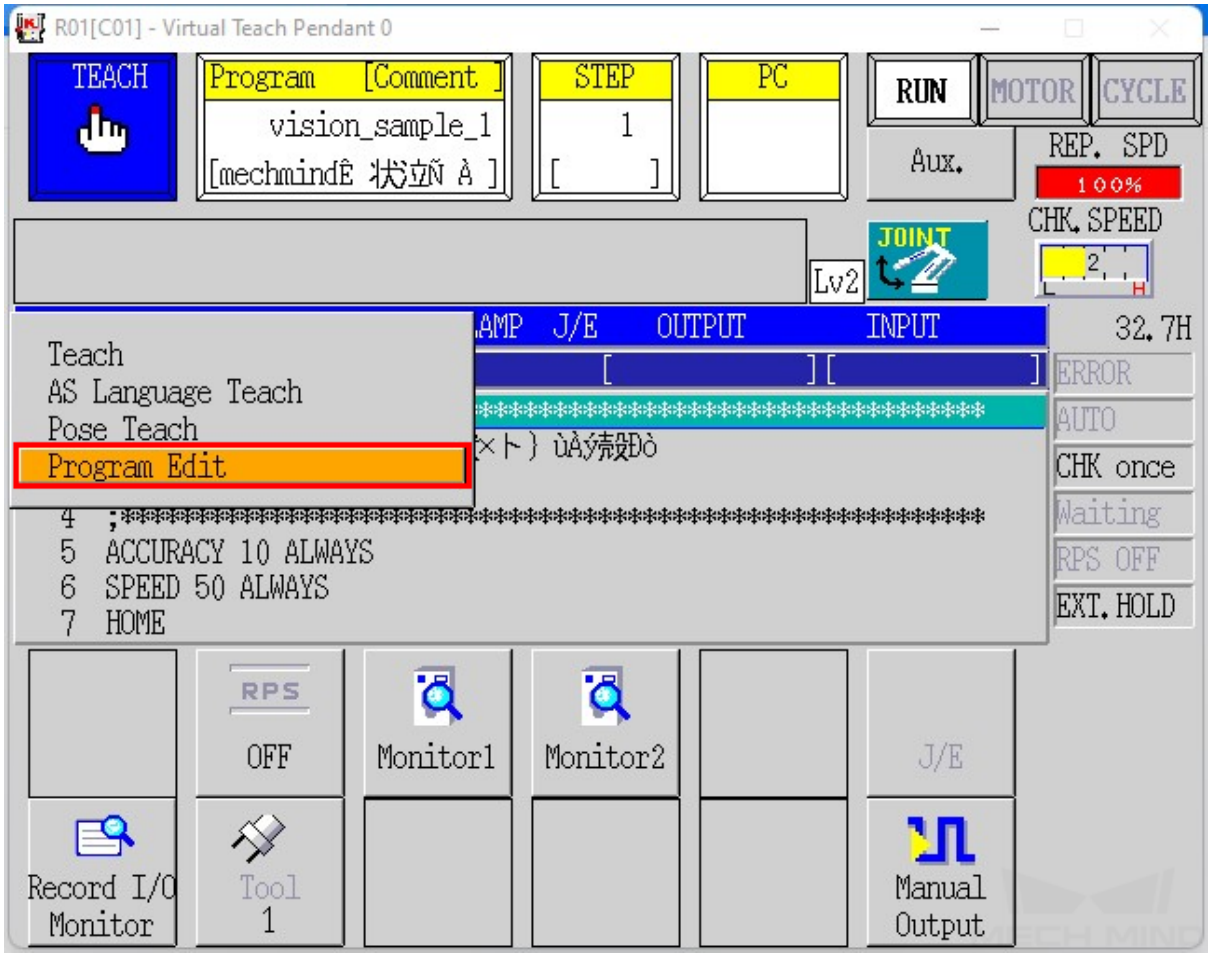


2. Select **vision_sample _1** in the list, and then press the `ENTER` key to confirm.

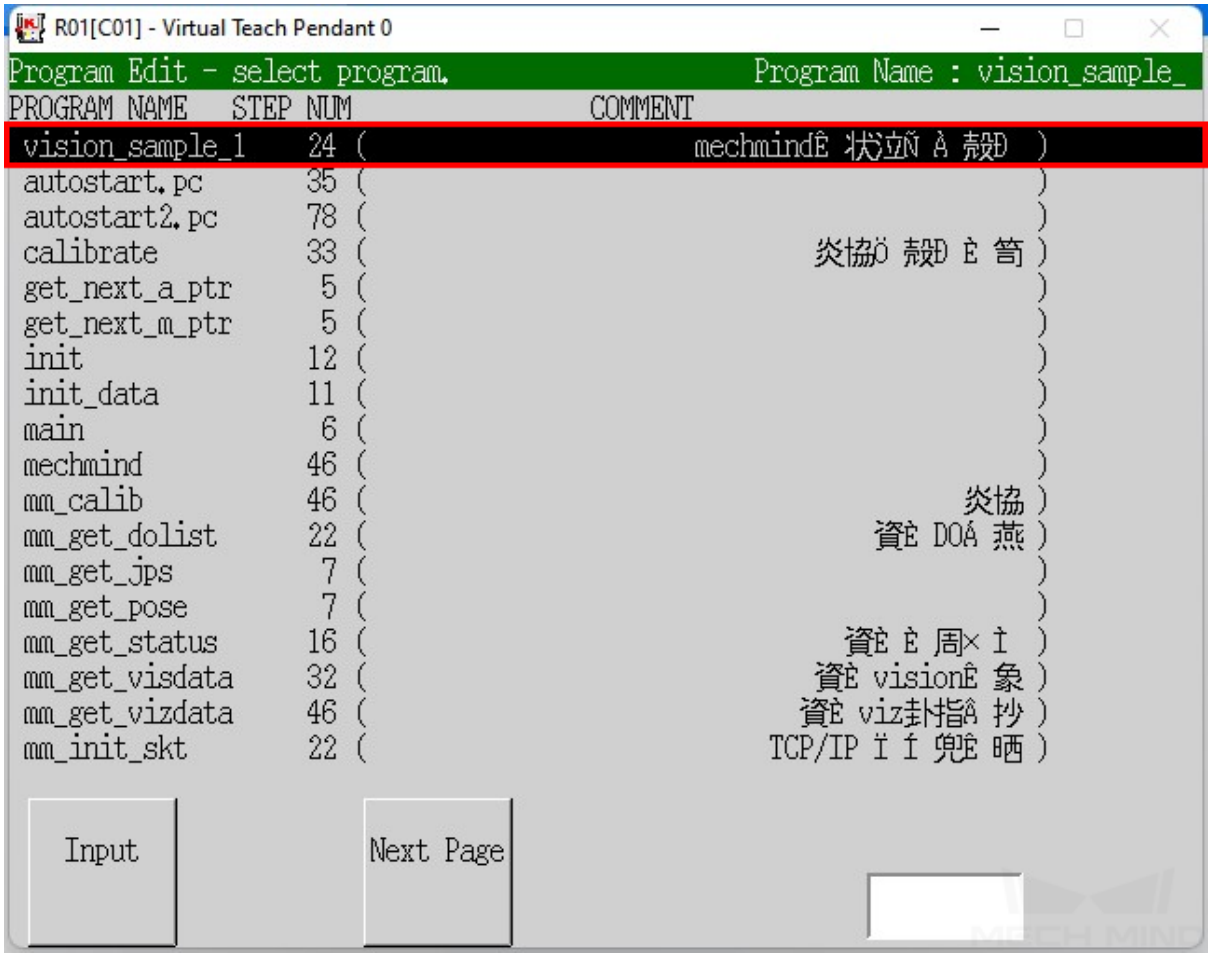


3. Modify the IP address and port number in the program:

   - Press the `J/E` key, select **Program Edit** in the pop-up menu, and then press the `ENTER` key to confirm.

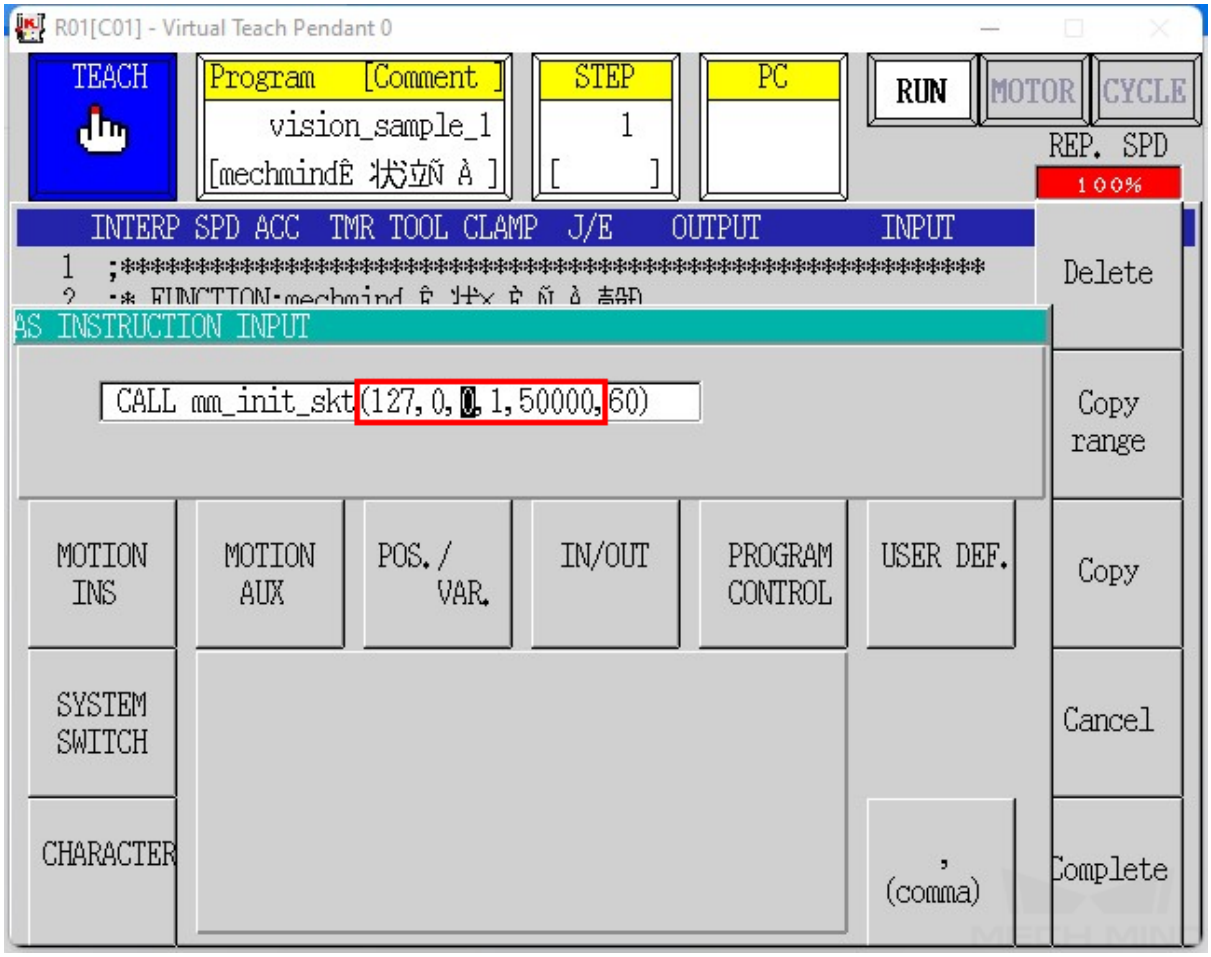


- Select **vision_sample_1** in the list, and then press the ENTER key to confirm.

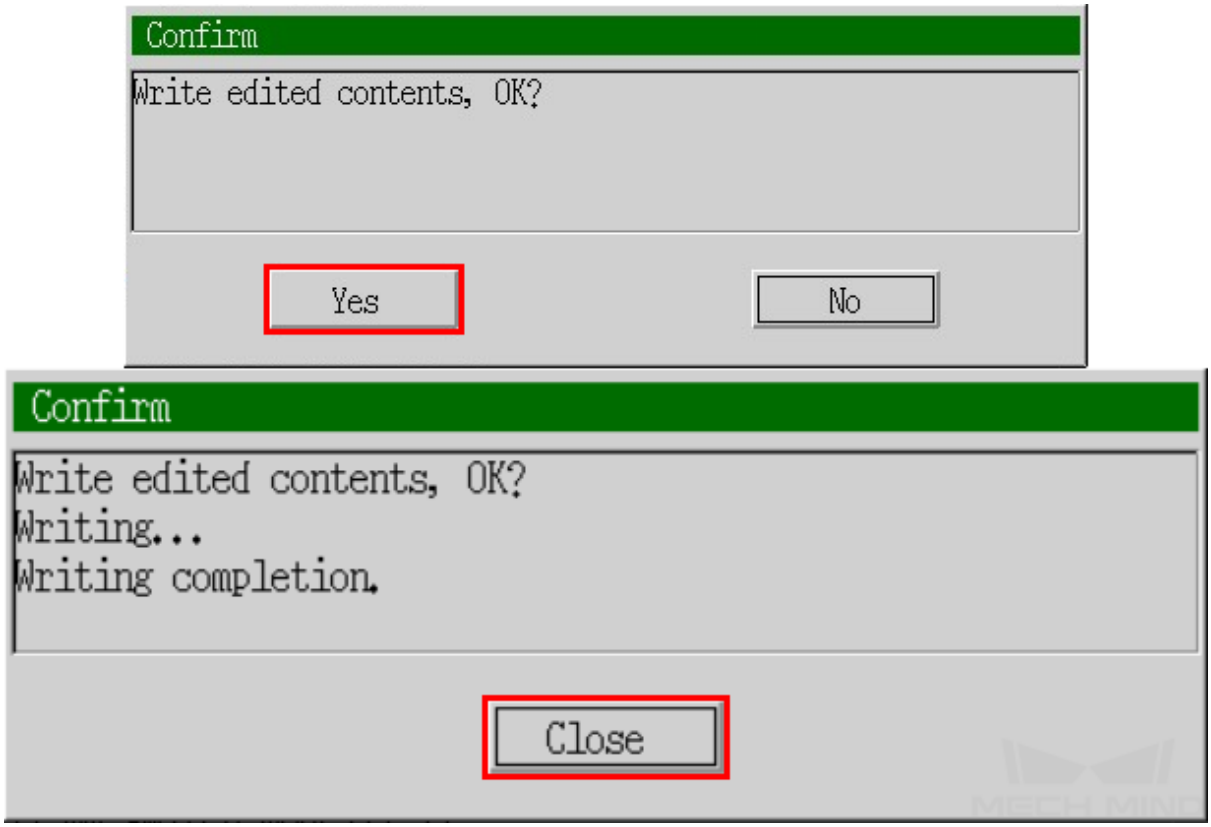


- Use the arrow keys on the teach pendant to move the cursor to Line 10, and then press the `ENTER` key to confirm.

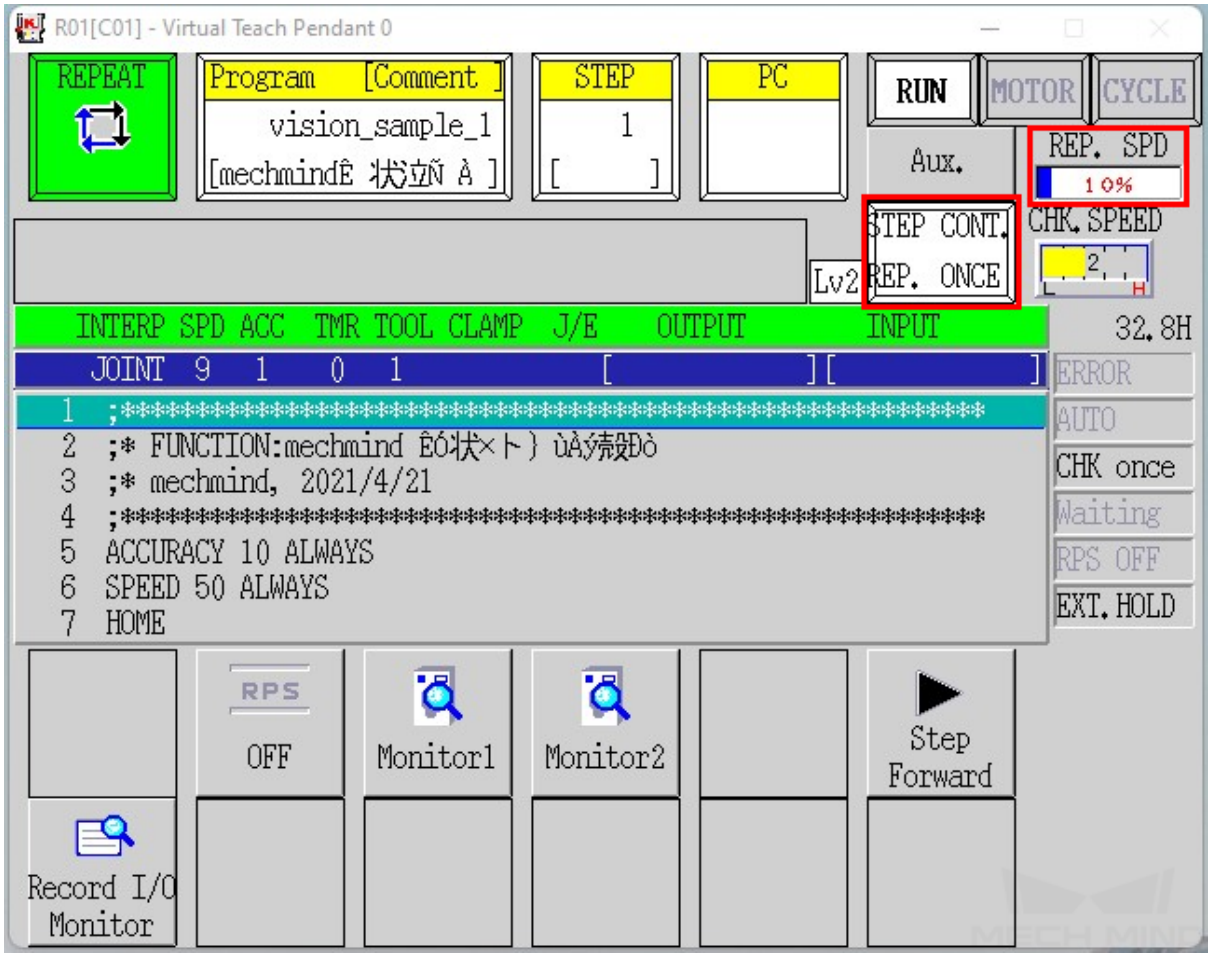
- Use the arrow keys and number keys to change:
  - **127,0,0,1** to the IPC’ s actual IP address
  - **50000** to the port number set in Mech-Center



- Press `ENTER` to save the changes, and then press `R` to exit.
- In the pop-up windows, press *Yes* and *Close*, respectively.

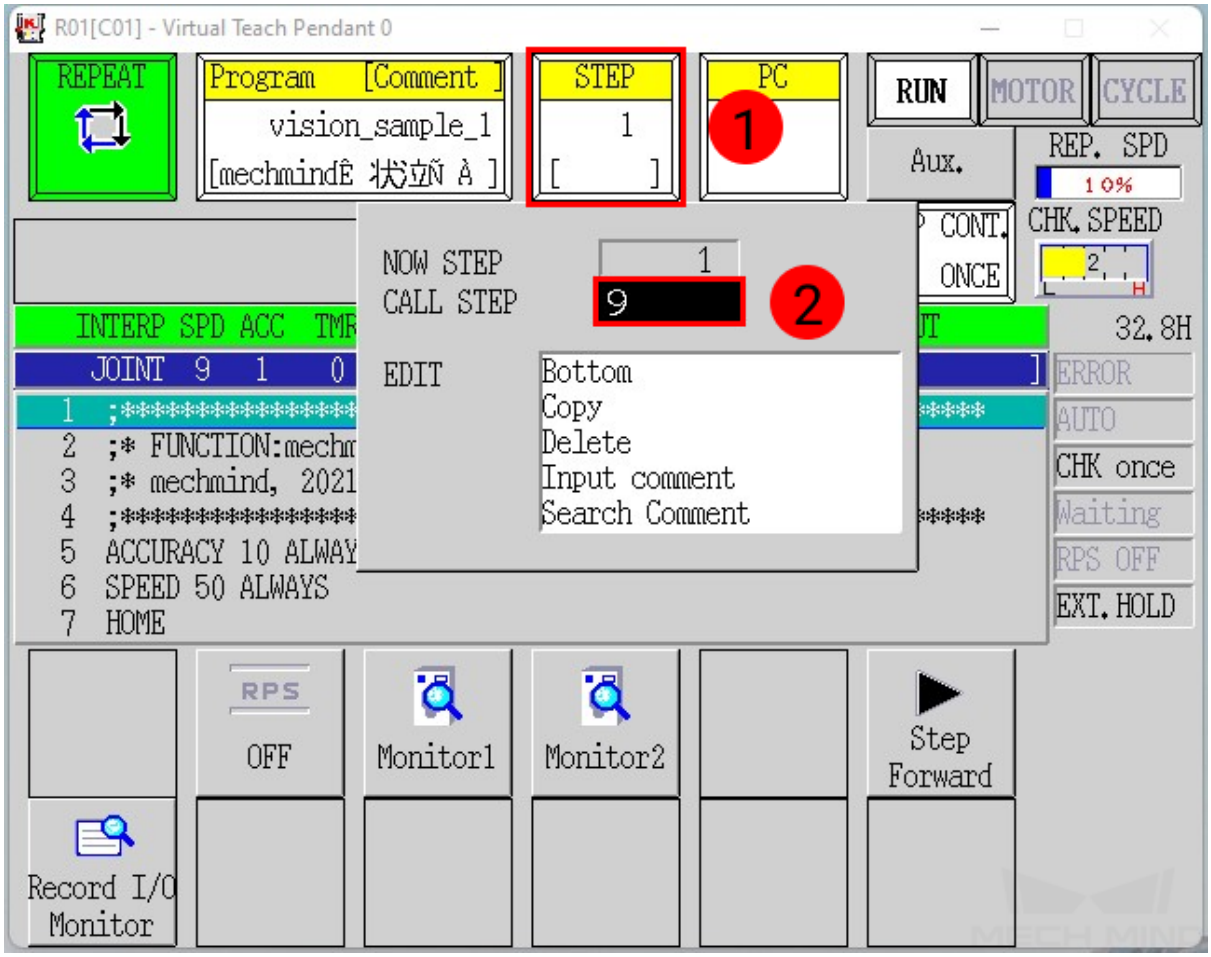


4. Switch the robot to repeat mode, press on **RPT. SPD** to adjust the repeat speed to **10%**. Press on the white button below *Aux.*, and change the drop-down options to **STEP CONT** and **REPEAT ONCE**.

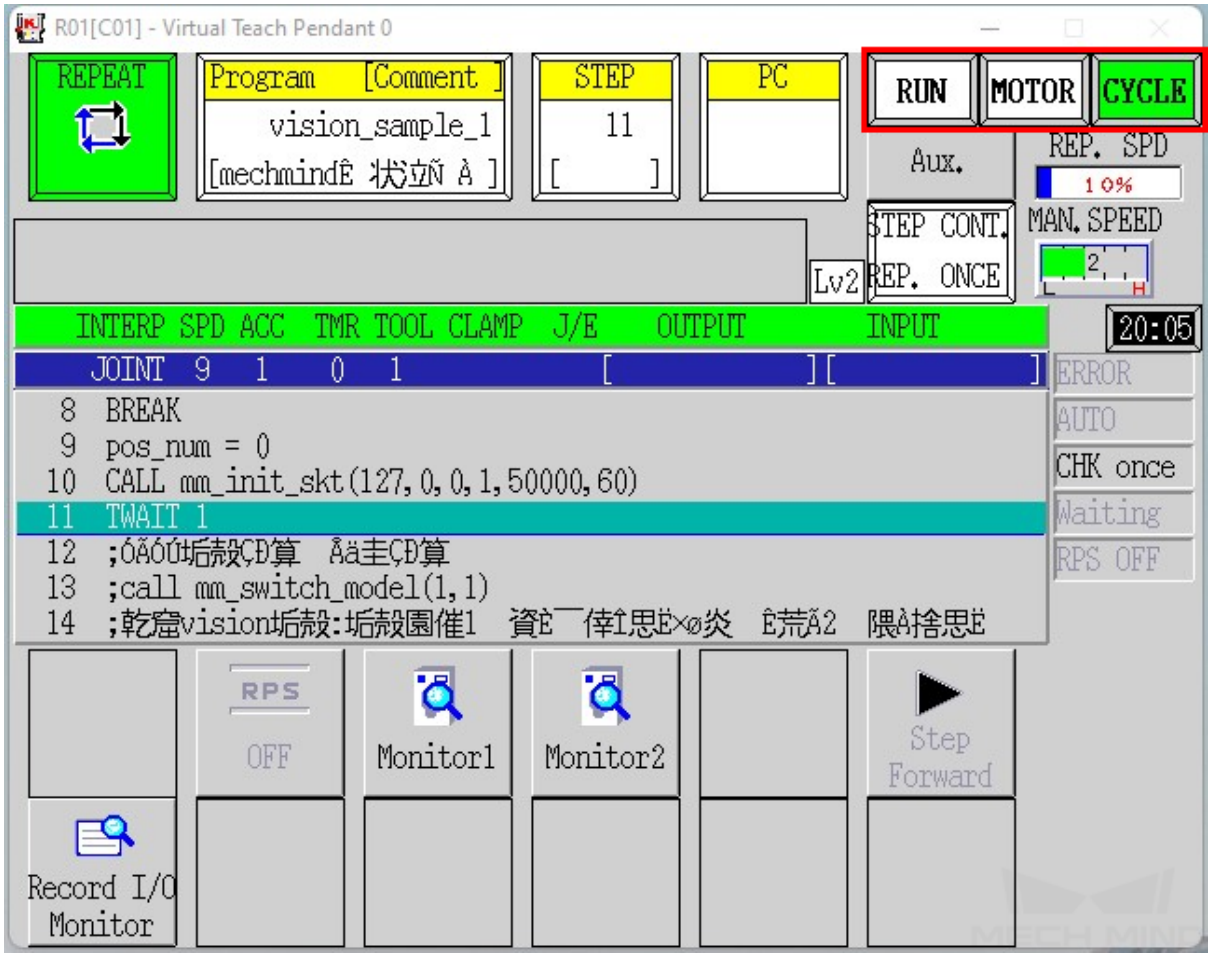


Note: To switch the robot to repeat mode, turn the Teach/Repeat switch on the controller to **REPEAT**, and the teach lock switch on the teach pendant to **OFF**.

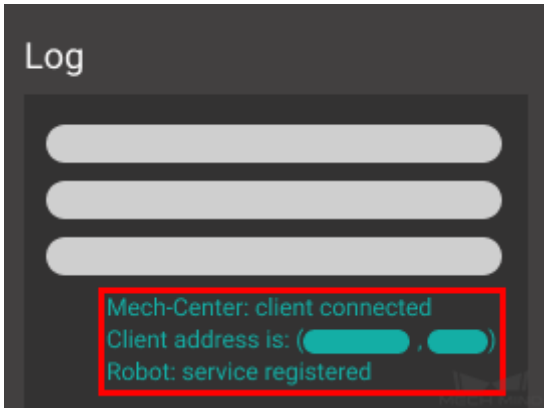
5. If the program is run from STEP 1, the robot will return to HOME position first according to the command in Line 7. To avoid this when testing the robot' s connection with Mech-Center, press on the **STEP** area, press the number key `9`, and then press `ENTER` to confirm.



6. Press on *MOTOR* while holding down the `A` key to power the motor.
7. Press on *CYCLE* while holding down the `A` key to run the program.
8. If *RUN* does not turn green, press the `RUN/HOLD` key while holding down the `A` key.



9. The robot is successfully connected if Mech-Center' s **Log** panel displays the following messages:
   - **Mech-Center: client connected**
   - A message showing the **client address**
   - **Robot: server registered**



2.5.2 Kawasaki Calibration Program

This section introduces the process of calibrating the camera extrinsic parameters using the calibration program.

The process consists of 4 steps:

- *Select the Calibration Program*
- *Teach the Calibration Start Point*
- *Run the Calibration Program*
- *Start Calibration in Mech-Vision*

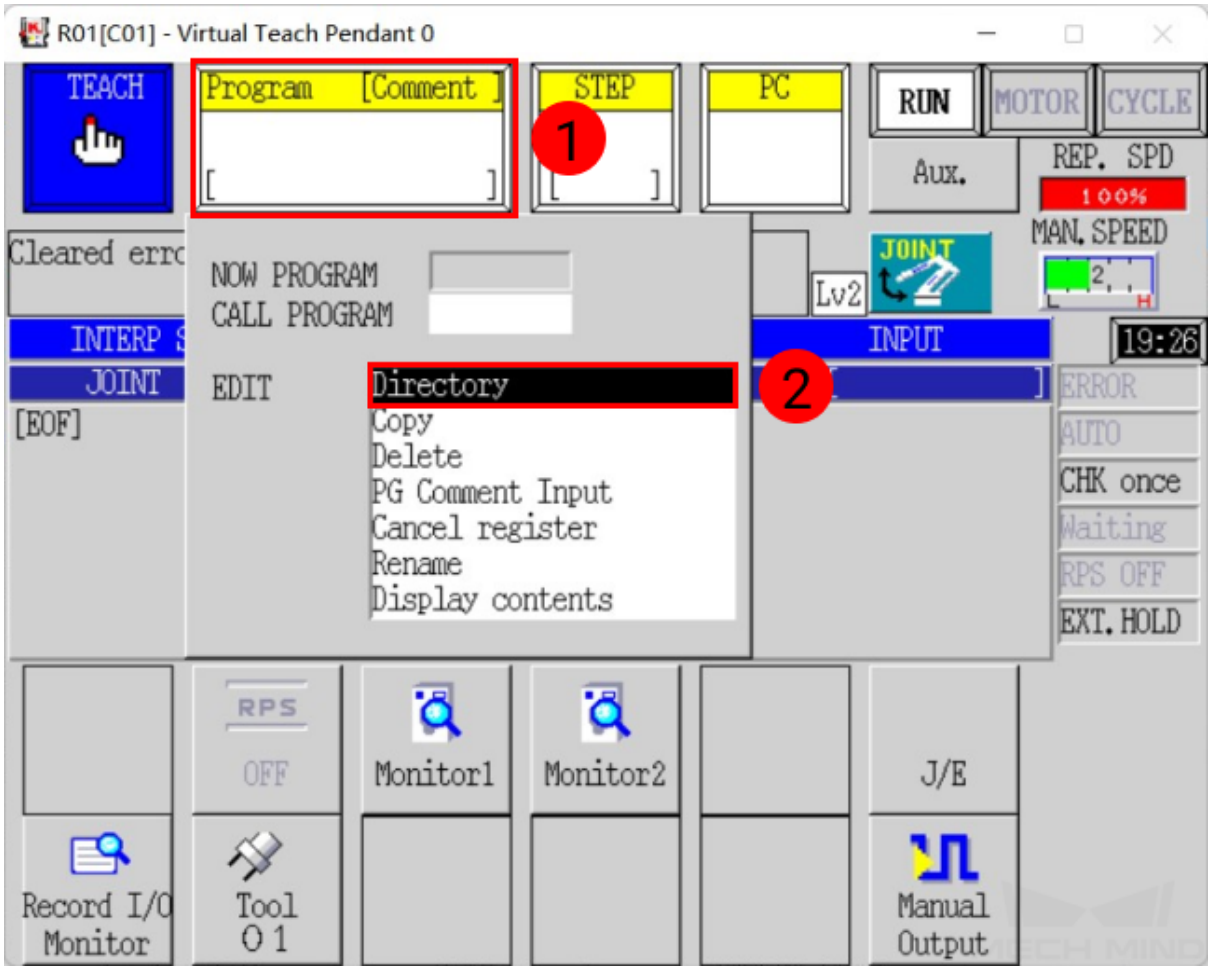
Before proceeding, please make sure that:

- You have *loaded the Standard Interface program* onto the robot and can establish communcation with Mech-Center.
- You are familiar with the contents in calibration_guide.

Note: This section is intended for scenarios where the communication between the robot and Mech-Center is established through Standard Interface, and calibration has to be performed frequently.

Select the Calibration Program

1. Switch the robot to teach mode, press on the **Program** area, and select **Directory**.

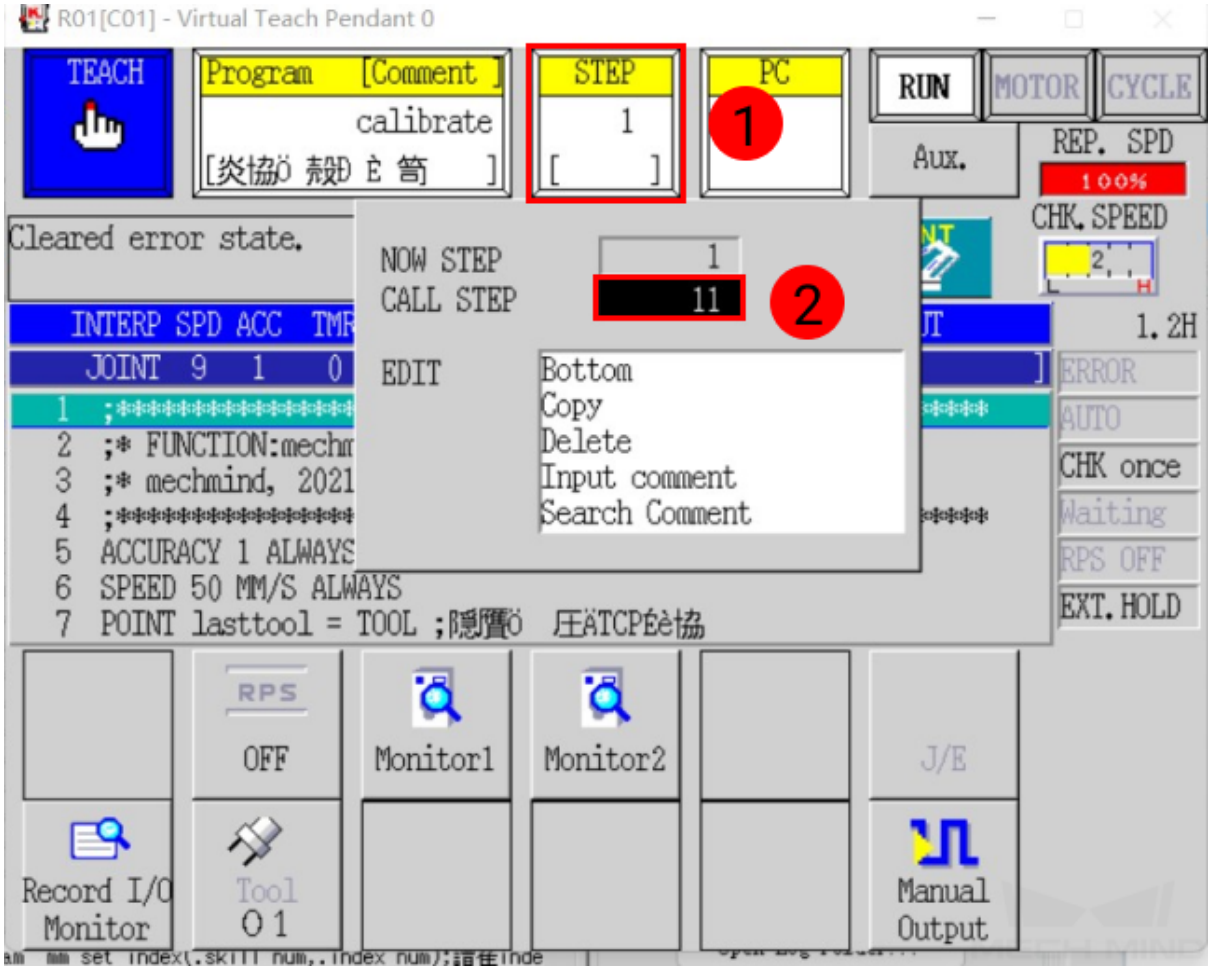


Note: To switch the robot to teach mode, turn the Teach/Repeat switch on the controller to **TEACH**, and the teach lock switch on the teach pendant to **ON**.

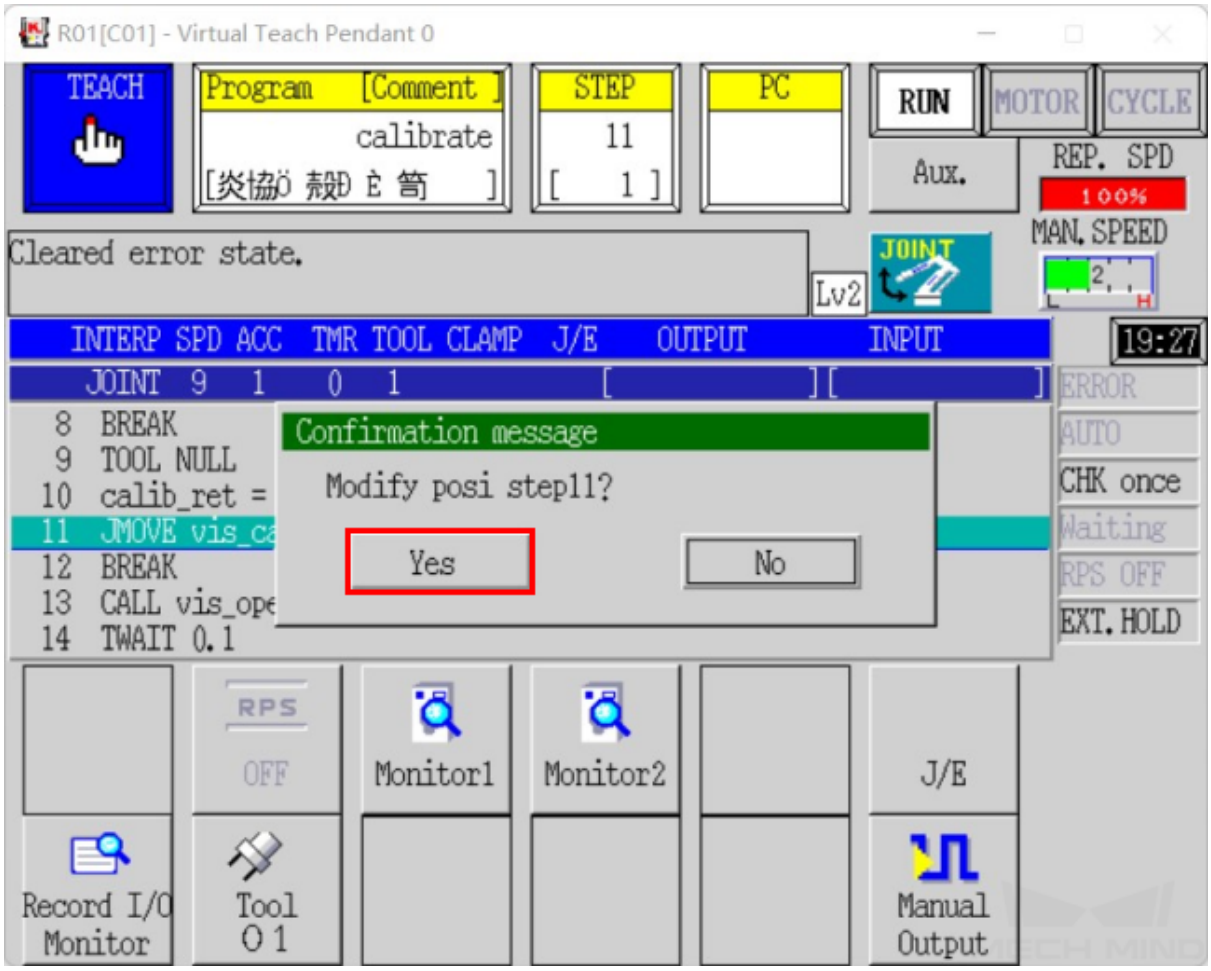
2. Select **calibrate** from the list, and press `ENTER` to confirm.

Teach the Calibration Start Point

1. Move the robot to the start point for the calibration.
2. Press on the **STEP** area, enter **11** with the number key, and then press `ENTER` to confirm.

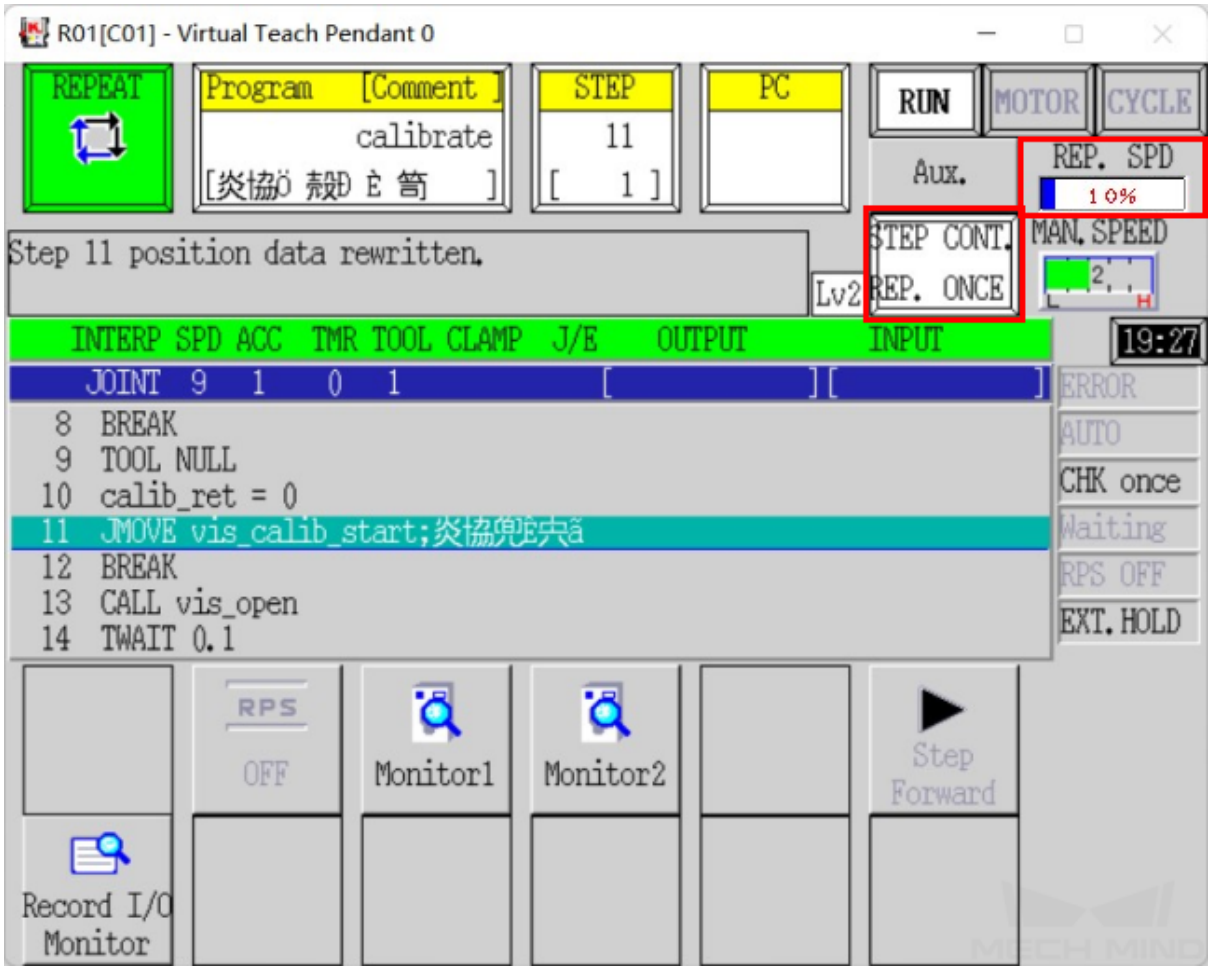


3. Press down the `A` key and `POS MOD` key at the same time, and press on *Yes* to modify the position in STEP 11.



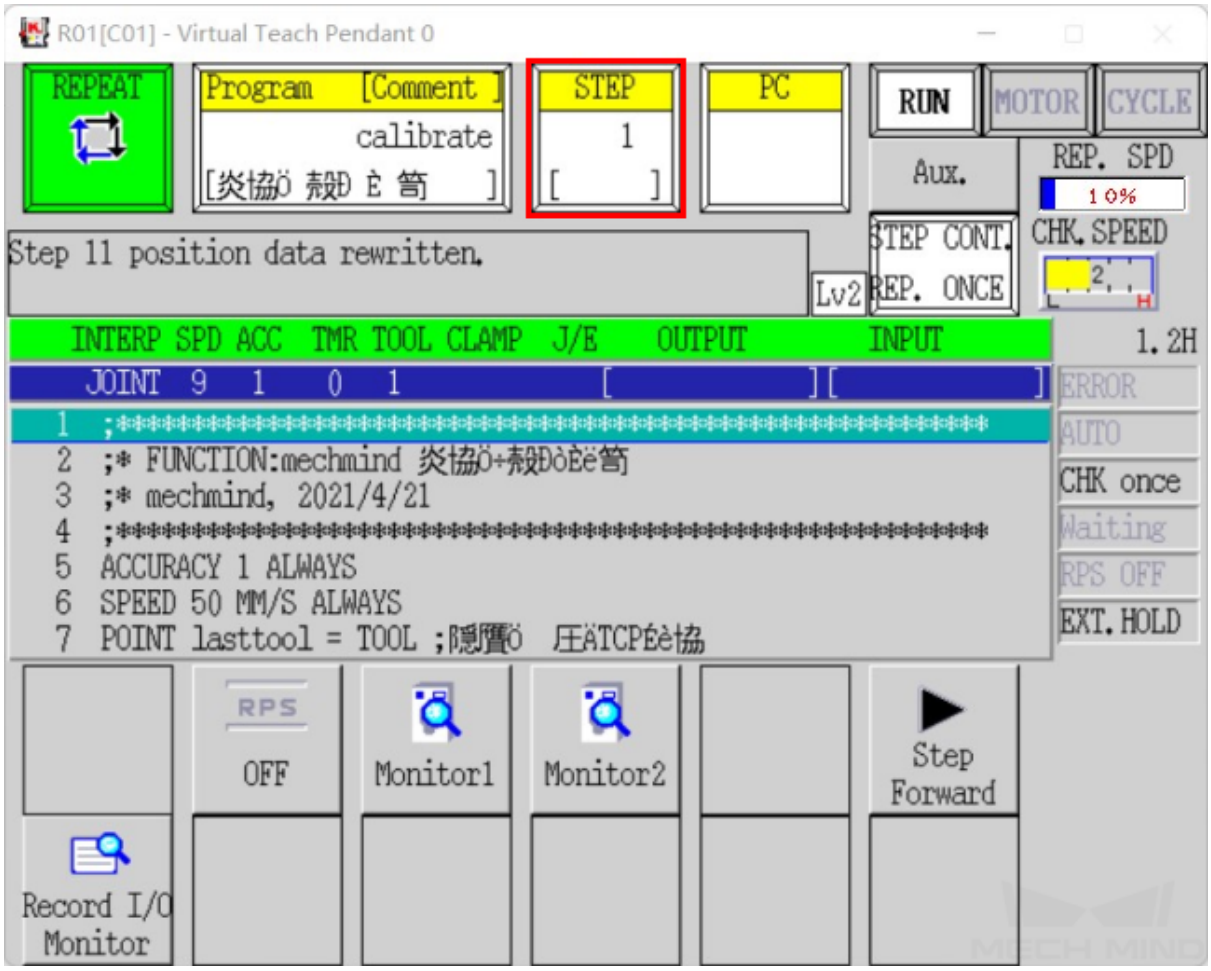
Run the Calibration Program

1. Switch the robot to repeat mode, press on **RPT. SPD** to adjust the repeat speed to **10%**. Press on the white button below *Aux.*, and change the drop-down options to **STEP CONT** and **REPEAT ONCE**.

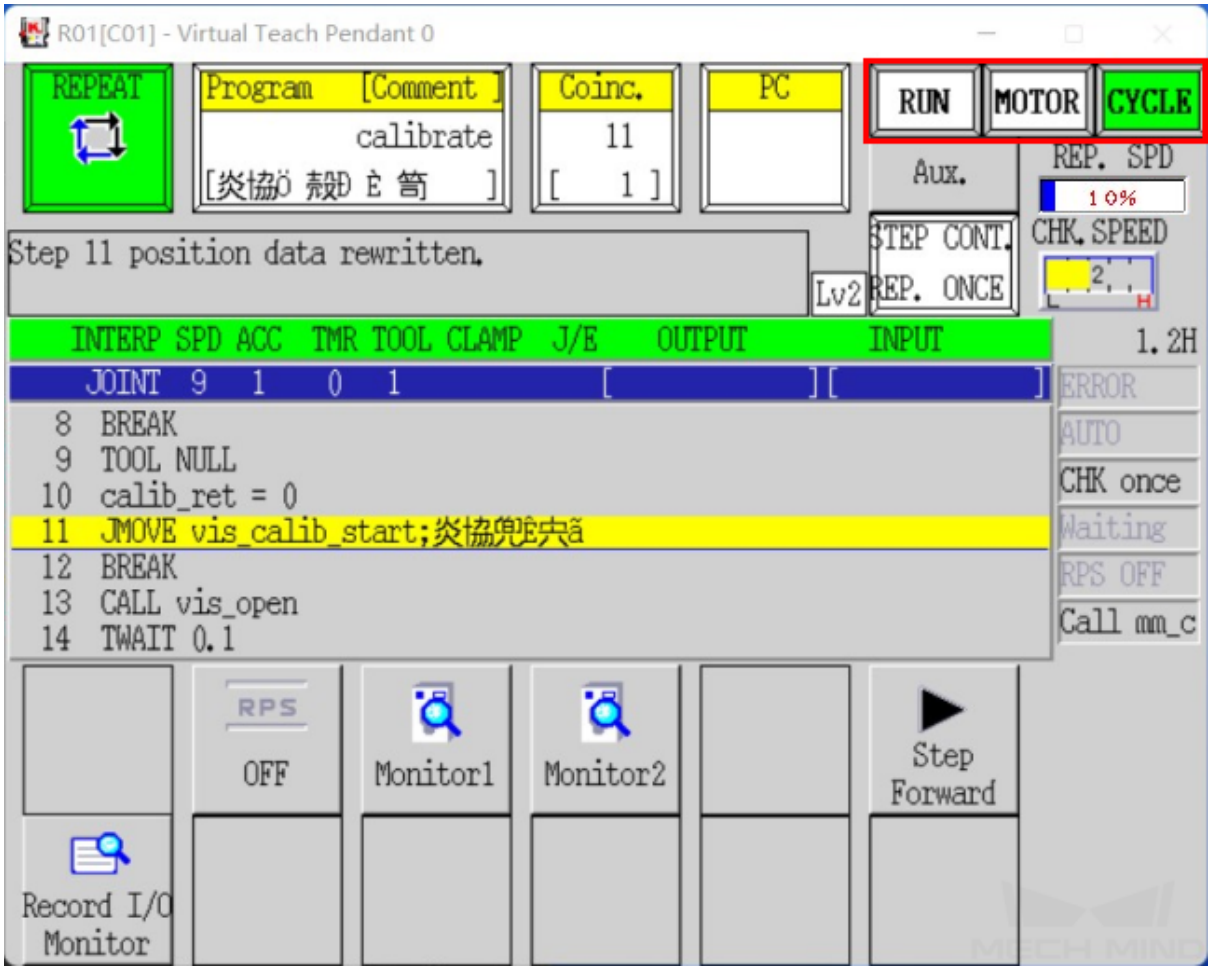


Note: To switch the robot to repeat mode, turn the Teach/Repeat switch on the controller to **REPEAT**, and the teach lock switch on the teach pendant to **OFF**.

2. Press on the **STEP** area, enter **1** with the number key, and then press `ENTER` to confirm.

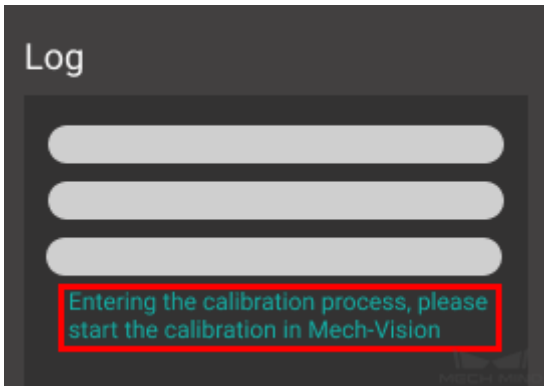


3. Press on *MOTOR* while holding down the `A` key to power the motor.
4. Press on *CYCLE* while holding down the `A` key to run the program.
5. If *RUN* does not turn green, press the `RUN/HOLD` key while holding down the `A` key.



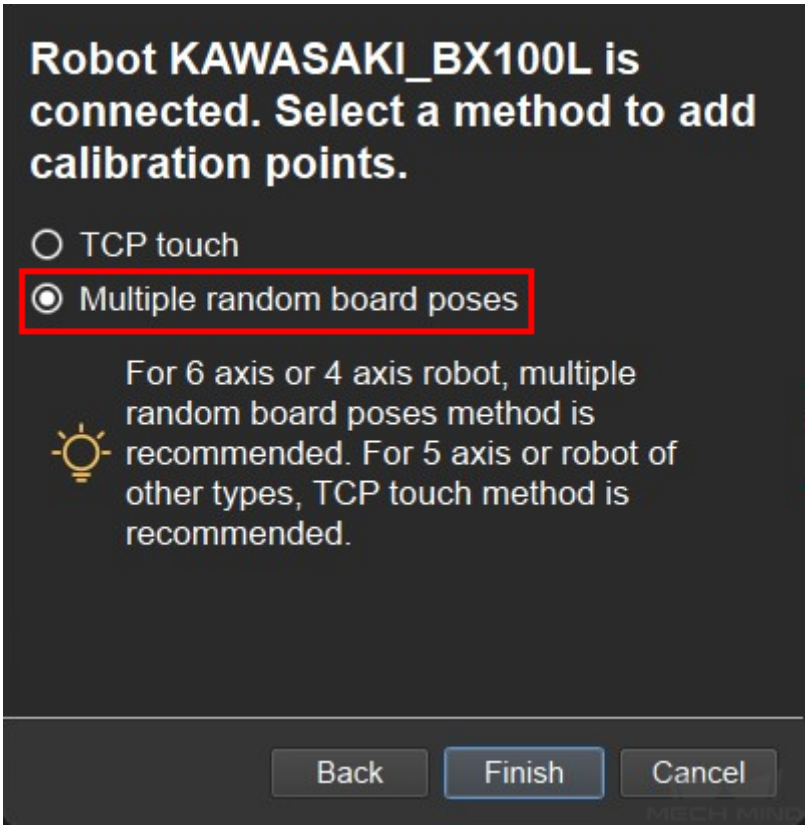
6. Proceed to the next section when the following message is displayed in Mech-Center's **Log** panel:

   Entering the calibration process, please start the calibration in Mech-Vision

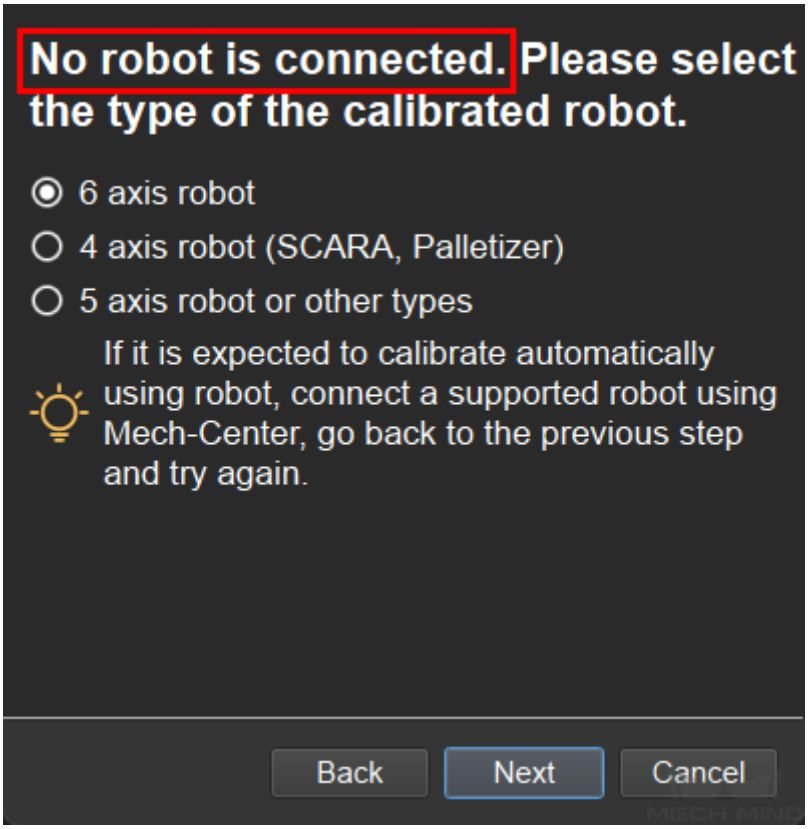


Start Calibration in Mech-Vision

1. In Mech-Vision, click on *Camera Calibration (Standard)* in the Toolbar, or select *Camera* → *Camera Calibration* → *Standard* from the Menu Bar.
2. Follow the instructions in Mech-Vision to complete the following configuration:
   1. Select **Start a new calibration process**;
   2. Select the camera mounting method;
   3. Select **Multiple random board poses** for adding calibration points.



Note: If after selecting the camera mounting method, the window says **No robot is connected**, the connection between the robot and Mech-Center is not properly established. Please re-run the robot program.



3. Follow the instructions in Mech-Vision to finish the calibration.

Note: In **5 Add Marker-Images and Poses** after you click on *Move Robot along Trajectory and Add Board Images*, if the robot does not reach the next calibration point within 60 seconds, Mech-Vision will report a timeout error and stop the calibration process. In such case, please select **calibration** in the program directory and run this program again, and restart the calibration process in Mech-Vision.

2.5.3 Kawasaki Example Program

This section introduces the example program provided with Mech-Center and the operations required to perform an actual pick-and-place task.

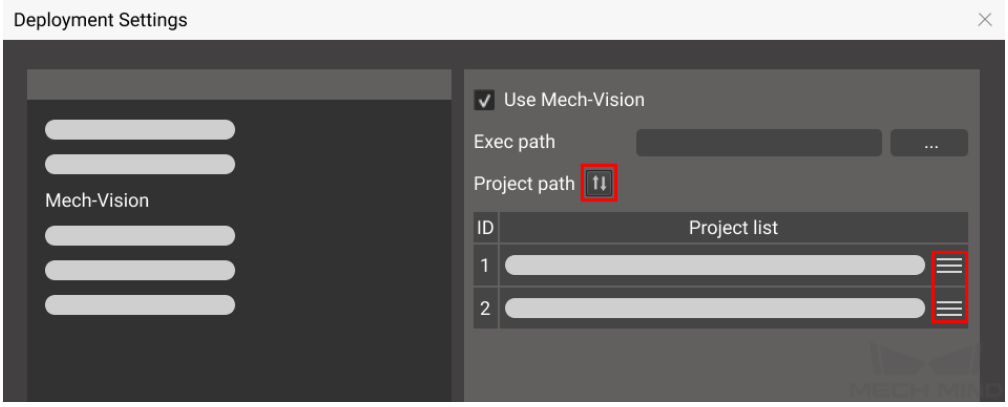
The example program **mm_sample.as** can be found in *Mech-Center/mech_interface/kawasaki*. It contains two parts: **vision_sample_1** obtains vision results from Mech-Vision; **vision_sample_2** obtains planned path from Mech-Viz.

Check the section corresponding to your own application setup:

- *Obtain Vision Results from Mech-Vision*
- *Obtain Planned Path from Mech-Viz*

Before running the program, please make sure that:

- You have *loaded the Standard Interface program* onto the robot and can establish communcation with Mech-Center.
- You have completed the extrinsic parameter calibration with *the calibration program* or by manually adding calibration points.
- Mech-Vision and Mech-Viz projects are created and set to autoload.
- The **Project list** in *Mech-Center* → *Deployment Settings* → *Mech-Vision* is synced by clicking on  , and the order of Mech-Vision projects have been adjusted according to actual needs.



- The TCP has been correctly specified.
- The robot speed is set to a low value, so that the operator can notice any unexpected behavior before accidents occur.

Obtain Vision Results from Mech-Vision

```
.PROGRAM vision_sample_1()
;*********************************************************
;* FUNCTION:simple pick and place with Mech-Vision
;* mechmind, 2022-5-1
;*********************************************************
  accuracy 1 always
  speed 30 always
  TOOL gripper ;set TCP
  Home ;move robot home position
  lmove camera_capture ;move to camera capture position
  break
  pos_num = 0
  ;Set ip address of IPC
  call mm_init_skt(127,0,0,1,50000,60)
  twait 0.1
  ;Set vision recipe
  ;call mm_switch_model(1,1)
  ;Run vision project
  call mm_start_vis(1,1,2)
  twait 1
  call mm_get_visdata(1,pos_num,ret2)
```

(continues on next page)

(continued from previous page)

```
  if ret2 <> 1100
      halt
  end
  call mm_get_pose(1,&pick[1],label[l],speed[1])
  LAPPRO pick[1],100
  LMOVE pick[1]
  break
  ;Add object grasping logic here.
  ldepart 100
  lmove waypoint[1]
  lappro drop[1],100
  lmove drop[1]
  ;Add object releasing logic here.
  ldepart 100
  HOME
.END
```

Program Logic

1. Move the robot to HOME position.
2. Move the robot to the image capturing pose.
3. Initialize communication with **mm_init_skit**.
4. If parameter recipes are used in the Mech-Vision project, the recipe to be used is set with **mm_switch_model**.
5. Run the Mech-Vision project with **mm_start_vis**.
6. Wait for 1 second. Under Eye-In-Hand, this **TWAIT** instruction is required to make sure the robot stays still until image acquisition is completed. Under Eye-To-Hand, this **TWAIT** instruction can be replaced with **LMOVE** or **JMOVE**.
7. Obtain the vision results from Mech-Vision.
8. Check if the returned status code indicates any error. If an error code is returned, the program is halted.
9. Move the robot to the picking pose and perform picking.
10. Move the robot to a waypoint between the picking pose and placing pose.
11. Move the robot to the set placing pose and perform placing.
12. Return the robot to HOME position.

The following parts should be modified according to your actual needs.

Define the TCP

Change the value of **gripper** in line 8 to the actual TCP values.

Teach the Image Capturing Pose

Record the image capturing pose in **camera_capture** in line 10.

Teach the Waypoint(s)

Waypoints are intermediate poses between the picking pose and placing pose. They are used to ensure that the robot doesn't collide with the surrounding when moving between the picking and placing poses.

You can add one or more waypoints to **waypoint**[] in line 31.

Teach the Placing Pose

Record the placing pose in **drop[1]** in line 33.

Define Z-Offset from Picking/Placing Pose

Z-offset distances relative to the tool frame from the picking/placing pose are used to ensure collision doesn't occur when the robot is approching or departing the picking/placing pose.

Adjust the following commands according to your actual needs.

- **LAPPRO pick[1],100** in line 26: the Z-offset when approching the picking pose is set to **100**. Robot will move to 100 mm above the picking pose.
- **ldepart 100** in line 30: the Z-offset when departing the picking pose is set to **100**. Robot will move to 100 mm above the picking pose.
- **lappro drop[1],100** in line 32: the Z-offset when approching placing pose is set to **100**. Robot will move to 100 mm above the placing pose.
- **ldepart 100** in line 35: the Z-offset when departing the placing pose is set to **100**. Robot will move to 100 mm above the placing pose.

Add Object Grasping and Releasing Logics

Add logic for controlling the tool action when picking the object to line 29.

Add logic for controlling the tool action when placing the object to line 34.

Define HOME position

The HOME position need to be set on the teach pendant in *Aux.* → *4. Basic setting* → *2. Home Position* beforehand.

Obtain Planned Path from Mech-Viz

```
.PROGRAM vision_sample_2()
;*********************************************************
;* FUNCTION:simple pick and place with Mech-Viz
;* mechmind, 2022-5-1
;*********************************************************
  accuracy 1 always
  speed 30 always
  TOOL gripper ;set TCP
  Home ;move robot home position
  LMOVE camera_capture ;move to camera_capture position
  break
  pos_num = 0
  ;Set ip address of IPC
  call mm_init_skt(127,0,0,1,50000,60)
  twait 0.1
  ;Set vision recipe
  ;call mm_switch_model(1,1)
  ;Run Viz project
  call mm_start_viz(1)
  twait 0.1
  ;set branch exitport
  ;call mm_set_branch(1,1)
  ;get planned path
  call mm_get_vizdata(2,pos_num,vispos_num,ret1)
  if ret1 <> 2100
      halt
  end
  for count=1 to pos_num
    call mm_get_pose(count,&movepoint[count],label[count],speed[count])
  end
  ;follow the planned path to pick
  for count =1 to pos_num
    speed speed[count]
    LMOVE movepoint[count]
    if count == vispos_num then
        ;add object grasping logic here
    end
  end
  ;go to drop location
  ldepart 100
  lmove waypoint[1]
  lappro drop[1],100
  lmove drop[1] ;drop point
  ;add object releasing logic here
  ldepart 100
  HOME
.END
```

Program Logic

1. Move the robot to HOME position.
2. Move the robot to the image capturing pose.
3. Initialize communication with **mm_init_skit**.
4. If parameter recipes are used in the Mech-Vision project, the recipe to be used is set with **mm_switch_model**.
5. Run the Mech-Viz project with **mm_start_viz**.
6. Obtain the planned path from Mech-Viz.
7. Check if the returned status code indicates any error. If an error code is returned, the program is halted.
8. Store obtained target points in the planned path to **&movepoint**[] with a **FOR** loop.
9. Move the robot along the planned path with a **FOR** loop and perform picking.
10. Move the robot to a waypoint between the picking pose and placing pose.
11. Move the robot to the set placing pose and perform placing.
12. Return the robot to HOME position.

The following parts should be modified according to your actual needs.

Define the TCP

Change the value of **gripper** in line 8 to the actual TCP values.

Teach the Image Capturing Pose

Record the image capturing pose in **camera_capture** in line 10.

Teach the Waypoint(s)

Waypoints are intermediate poses between the picking pose and placing pose. They are used to ensure that the robot doesn' t collide with the surrounding when moving between the picking and placing poses.

You can add one or more waypoints to **waypoint**[] in line 41.

Teach the Placing Pose

Record the placing pose in **drop**[**1**] in line 43.

Add Object Grasping and Releasing Logics

Add logic for controlling the tool action when picking the object to line 29.

Add logic for controlling the tool action when placing the object to line 34.

Define HOME position

The HOME position need to be set on the teach pendant in *Aux.* → *4. Basic setting* → *2. Home Position* beforehand.

2.5.4 Kawasaki Standard Interface Commands

The Kawasaki Standard Interface provides the following subroutines:

- *Initialize Communication*
- *Start Mech-Vision Project*
- *Get Vision Result*
- *Start Mech-Viz Project*
- *Get Planned Path*
- *Obtain Pose*
- *Obtain Joint Positions*
- *Switch Mech-Vision Recipe*
- *Select Mech-Viz Branch*
- *Set Move Index*
- *Get Software Status*
- *Input Object Dimensions to Mech-Vision*
- *Get DO Signal List*
- *Input TCP to Mech-Viz*
- *Calibration*

When writing your own program, pay attention to the following:

- Multiple parameters should be separated by commas.
- All parameters should be defined as local variables.
- Parameters can be defined as input or output parameters.
- Input arguments can be constants, global variables or local variables; output arguments can be global variables or local variables.

This Standard Interface is over the TCP/IP protocol.

Initialize Communication

```
mm_init_skt(.ip1,.ip2,.ip3,.ip4,.port,.time_out)
```

This subroutine sets the IP address and port number of the IPC and wait time before the program stops trying to establish the communication.

Parameters

- Input parameters

Name	Description
.ip1 - .ip4	IP address of the IPC
.port	Port number of the IPC, the default is 50000
.time_out	Wait time in seconds before stopping connection attempt

Example

```
CALL mm_init_skt(192,168,1,1,50000,60)
```

This example sets the IP address and port number of the IPC to 192.168.1.1:50000 and wait time to 60 seconds.

Start Mech-Vision Project

```
mm_start_vis(.job,.pos_num_need,.sendpos_type)
```

This subroutine is for applications that use Mech-Vision but not Mech-Viz. It runs the corresponding Mech-Vision project to acquire and process data.

Parameters

- Input parameters

Name	Description
.job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center → Deployment Settings → Mech-Vision*
.pos_num_need	Number of poses for Mech-Vision to send, from 0 to 20, where 0 means "send all".
.sendpos_type	Set the image capturing pose for the robot to send, from 0 to 2
	0: Do not send image capturing pose (for Eye To Hand) 1: Send image capturing pose as joint positions 2: Send image capturing pose as robot flange pose

Example

```
CALL mm_start_vis(1,1,1)
```

This example runs Mech-Vision project No. 1, and asks the project to send over 1 pose, and the robot sends its joint positions when this subroutine is called as the image capturing pose to Mech-Center.

Get Vision Result

```
mm_get_visdata(.job,.pos_num,.ret)
```

This subroutine is for applications that use Mech-Vision but not Mech-Viz. It obtains the vision result from the corresponding Mech-Vision project.

Parameters

- Input parameter

Name	Description
.job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center* → *Deployment Settings* → *Mech-Vision*

- Output parameters

Name	Description
.pos_num	Variable for storing the number of received poses
.ret	Variable for storing status code, refer to the standard_interface_status_codes

Example

```
CALL mm_get_visdata(1,posnum,statuscode)
```

This example obtains the vision result from Mech-Vision project No. 1. The number of poses received is stored in **posnum**, and the status code is stored in **statuscode**.

Start Mech-Viz Project

```
mm_start_viz(.sendpos_type)
```

This subroutine is for applications that use both Mech-Vision and Mech-Viz. It runs the corresponding Mech-Viz project (which triggers the corresponding Mech-Vision project to run), and sets the initial joint positions of the simulated robot in Mech-Viz.

Parameter

- Input parameter

Name	Description
.sendpos_type	The initial joint positions for the simulated robot in Mech-Viz, 0 or 1
	0: Set the initial joint positions of the simulated robot to [0,0,0,0,0,0] 1: Set the initial joint positions of the simulated robot to the current joint positions of the real robot

Note: When the scene contains object models that obstruct the robot to move from [0,0,0,0,0,0] to the first target point, this parameter must be set to 1.

Example

```
CALL mm_start_viz(1)
```

This example runs the corresponding Mech-Viz project, and sets the initial joint positions of the simulated robot to the current joint positions of the real robot.

Get Planned Path

```
mm_get_vizdata(.getpos_type,.pos_num,.vispos_num,.ret)
```

This subroutine obtains the planned path from Mech-Viz.

Parameters

- Input parameter

Name	Description
.getpos_type	Whether Mech-Viz should send target points as joint positions or TCPs, 1 or 2
	1: Mech-Viz sends joint positions 2: Mech-Viz sends TCPs

- Output parameters

Name	Description
.pos_num	Variable for storing the number of received target points
.vispos_num	Variable for storing the position of the first visual_move target point in the path
	Example path: move-1, move-2, visual_move-3, move-3, visual_move-2 In this path, the position of the first visual_move target point is 3. If the path does not contain visual_move target point, the return value is 0.
.ret	Variable for storing status code, refer to the standard_interface_status_codes

Example

```
CALL mm_get_vizdata(2,posnum,vis_index,statuscode)
```

This example obtains the planned path from Mech-Viz in the form of TCPs. The number of target points received is stored in **posnum**, the position of the visual_move target point is stored in **vis_index**, and the status code is stored in **statuscode**.

Obtain Pose

```
mm_get_pose(.index,.&targetpos,.label,.speed)
```

This subroutine stores a pose returned by Mech-Vision or a target point (as TCP) returned by Mech-Viz in the specified variable.

Parameters

- Input parameter

Name	Description
.index	Specify the index of the pose to be stored

- Output parameters

Name	Description
.&targetpos	Variable for storing the specified pose
	Must add "&" before the variable name to indicate the variable as transformation values
.label	Variable for storing the label corresponding to the specified pose
.speed	Variable for storing the speed corresponding to the specified pose

Example

```
CALL mm_get_pose(1,.&pt[1],pt_label[1],pt_speed[1])
```

This example stores the first received pose to **pt[1]**, the corresponding label to **pt_label[1]**, and the corresponding speed to **pt_speed[1]**.

Obtain Joint Positions

```
mm_get_jps(.index,.#targetpos,.label,.speed)
```

This subroutine stores a set of joint positions returned by Mech-Viz in the specified variable.

Note: As Mech-Vision does not output joint position data, this subroutine can only be used with Mech-Viz.

Parameters

- Input parameter

Name	Description
.index	Specify the index of the set of joint positions to be stored

- Output parameters

Name	Description
.#targetpos	Variable for storing the specified set of joint positions
	Must add "#" before the variable name to indicate the variable as joint displacement values
.label	Variable for storing the label corresponding to the specified set of joint positions
.speed	Variable for storing the speed corresponding to the specified set of joint positions

Example

```
CALL mm_get_jps(1,.#pt[1],pt_label[1],pt_speed[1])
```

This example stores the first set of received joint positions to **pt[1]**, the corresponding label to **pt_label[1]**, and the corresponding speed to **pt_speed[1]**.

Switch Mech-Vision Recipe

```
mm_switch_model(.job,.model_number)
```

This subroutine specifies which parameter recipe of the Mech-Vision project to use. For more information on parameter recipe, please see parameter_recipe_configuration.

Note:

- This subroutine must be called BEFORE **mm_start_vis**.
- The corresponding Mech-Vision project must have parameter recipes already configured and saved.

Parameters

- Input parameters

Name	Description
.job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center → Deployment Settings → Mech-Vision*
.model_number	The number of a parameter recipe in the Mech-Vision project, from 1 to 99

Example

```
CALL mm_switch_model(2,2)
```

This example switches the parameter recipe used to No. 2 in Mech-Vision project No. 2.

Select Mech-Viz Branch

```
mm_set_branch(.branch_num,.exit_num)
```

This subroutine is used to select along which branch the Mech-Viz project should proceed. Such branching is achieved by adding branch_by_service_message Task(s) to the project. This subroutine specifies which out port such Task(s) should take.

Note:

- **mm_start_viz** must be called BEFORE this subroutine.
- When the next Task to be executed in Mech-Viz is a **branch_by_service_message** Task, Mech-Viz will wait for this subroutine to send the out port number it should take.
- The name of all **branch_by_service_message** Tasks in the Mech-Viz project must be changed to numbers between 1 and 99, and the names should be unique among all tasks in the project.

Parameters

- Input parameters

Name	Description
.branch_num	Name of the **branch_by_service_message** Task, from 1 to 99
.exit_num	The number of the out port to take, from 1 to 99

Example

```
CALL mm_set_branch(1,3)
```

This example tells Mech-Viz to take out port 3 for the **branch_by_service_message** Task named **1**.

Set Move Index

```
mm_set_index(.skill_num,.index_num)
```

This subroutine sets the value for the Current Index parameter of Mech-Viz Tasks. Tasks that have this parameter include move_list, move_grid, custom_pallet_pattern, and smart_pallet_pattern.

Note:

- **mm_start_viz** must be called BEFORE this subroutine.
- The name of all Tasks with index parameters in the Mech-Viz project must be changed to numbers between 1 and 99, and the names should be unique among all tasks in the project.

Parameters

- Input parameters

Name	Description
.skill_num	Name of the Task, from 1 to 99
.index_num	Value for the Current Index parameter when the Task is executed

Example

```
CALL mm_set_index(2,10)
```

This example sets the Current Index value to 9 for the Task named **2**. When the Task is executed, the Current Index value will be added 1 and become 10.

Get Software Status

```
mm_get_status(.ret)
```

This subroutine is currently capable of checking whether Mech-Vision is ready to run projects. In the future, this subroutine can be used for obtaining the execution status of Mech-Vision, Mech-Viz and Mech-Center.

Parameter

- Output parameter

Name	Description
.ret	Variable for storing the status code, refer to the standard_interface_status_codes

Example

```
CALL mm_get_status(statuscode)
```

This example obtains the status code and stores it in **statuscode**.

Input Object Dimensions to Mech-Vision

```
mm_set_boxsize(.job,.length,.width,.height)
```

This subroutine inputs object dimensions to the Mech-Vision project.

Note:

- This subroutine must be called BEFORE **mm_start_vis**.

Parameters

- Input parameters

Name	Description
.job	Mech-Vision Project ID, from 1 to 99
	Can check and adjust in *Mech-Center* → *Deployment Settings* → *Mech-Vision*
.length	Length of object in mm
.width	Width of object in mm
.height	Height of object in mm

Example

```
CALL mm_set_boxsize(1,500,300, 200)
```

This example sets the object dimensions in the read_object_dimensions Step in the Mech-Vision project No. 1 to 500*300*200 mm.

Get DO Signal List

```
mm_get_dolist()
```

This subroutine obtains the planned DO Signal list for controlling multiple sections of a sectioned vacuum gripper.

Note:

- **mm_get_vizdata** must be called BEFORE this subroutine.
- Please deploy the Mech-Viz project based on the template project in *Mech-Center/tool/viz_project/suction_zone*, and set the suction cup configuration file in the Mech-Viz project.

Parameters

No parameters.

Example

```
CALL mm_get_dolist
```

This example obtains the DO signal list planned by Mech-Viz and stores it in **setdo**[] array. The first array element is **setdo**[**0**] and the last is **setdo**[**io_index-1**].

Input TCP to Mech-Viz

```
mm_set_pos(.&pos)
```

This subroutine inputs TCP data to the outer_move Task.

Note:

- This subroutine must be called BEFORE **mm_start_viz**.
- Please deploy the Mech-Viz project based on the template project in *Mech-Centertoolviz_projectouter_move*, and put the **outer_move** Task at a proper position in the workflow.

Parameter

- Input parameter

Name	Description
.&pos	Variable for storing the TCP data to be sent to Mech-Viz
	Must add "&" before the variable name to indicate the variable as transformation values

Example

```
call mm_set_pos(&pos)
```

This example sends the TCP data stored in **&pos** to the **outer_move** task in the Mech-Viz project.

Calibration

```
calibrate()
```

This subroutine is used for hand-eye calibration (camera extrinsic parameter calibration). It automates the calibration process in conjunction with the **Camera Calibration** function in Mech-Vision. For detailed instructions, see *Kawasaki Calibration Program*.

2.5.5 Kawasaki Error Messages

The following errors may occur while running the Standard Interface program on the robot.

Note: If robot reports an error message and the program is halted, please cancel the program and select it from the directory again after problem is resolved.

Error: Invalid TCP socket, Error: vis_socket invalid ID

When the program calls the **TCP_SEND** or **TCP_CLOSE** functions, the value of **vis_socket** is incorrect.

Troubleshooting

- Check the program to see if **TCP_CONNECT** is called before **TCP_SEND** or **TCP_CLOSE**. **TCP_CONNECT** must be called first to assign value to **vis_socket**.
- Check if the variable **vis_socket** is used in other user programs.

Error: TCP_CLOSE fault

Calling the **TCP_CLOSE** function failed.

Troubleshooting

- Check if the hardware are properly connected.
- Check if the Standard Interface is started in Mech-Center.
- Contact Mech-Mind Technical Support for further assistance.

Error: TCP connect fault

Calling the **TCP_CONNECT** function failed.

Troubleshooting

- Check if the hardware are properly connected.
- Check if the Standard Interface is started in Mech-Center.
- Check the IP addresses of the robot and the IPC, and if the port number is configured correctly.
- Check if the firewall is turned off on the IPC.
- Contact Mech-Mind Technical Support for further assistance.

Error: TCP_SEND null string

The string sent when calling the **TCP_SEND** function is empty.

Troubleshooting

Check if the variable **$vis_data[1]** is used in other user programs.

Error: TCP_SEND fault

Communication failed when calling the **TCP_SEND** function.

Troubleshooting

- Check the parameters of **TCP_SEND**.
- Contact Mech-Mind Technical Support for further assistance.

Error: TCP_SEND retry counter exceed limit

Communication failed when calling the **TCP_SEND** function, and retry times exceeded set limit.

Troubleshooting

- Check the parameters of **TCP_SEND**.
- Contact Mech-Mind Technical Support for further assistance.

Error: TCP_RECV fault

Communication failed when calling the **TCP_RECV** function.

Troubleshooting

- Check the parameters of **TCP_RECV**.
- Check if the Standard Interface is started in Mech-Center.
- Contact Mech-Mind Technical Support for further assistance.

Error: TCP_RECV retry counter exceed limit

Communication failed when calling the **TCP_RECV** function, and retry times exceeded set limit.

Troubleshooting

- Check the parameters of **TCP_RECV**.
- Check if the Standard Interface is started in Mech-Center.
- Contact Mech-Mind Technical Support for further assistance.

Error: CMD error

The command code received does not match the one sent.

Troubleshooting

The sequence of command sending and receiving is problematic. Please contact Mech-Mind Technical Support for further assistance.

Error: IPC error

Returned status code is an error code. Please check Mech-Center' s log.

Troubleshooting

- Please refer to the standard_interface_status_codes for the specific error.
- Please contact Mech-Mind Technical Support for further assistance.

2.6 EtherNet/IP - KEYENCE PLC

This section provides information on setting up communication between a KEYENCE PLC and Mech-Mind Software Suite via EtherNet/IP.

2.6.1 Overview

- *Hardware and Software Requirements*
- *Configure IPC and Initiate Communication*
- *Install EDS file and Configure Communication*
- *Import Example Program and Download to PLC*
- *Test with Mech-Vision/Mech-Viz Project*

2.6.2 Hardware and Software Requirements

Hardware

- KEYENCE PLC:
  - KV-8000 series
  - KV-8000A
  - KV-7500
  - KV-5500
  - Other models with a KV-EP21V or KV-NC1EP EtherNet/IP Unit
- USB Type A Male to Type B Male cable
- AC 220 V to DC 24 V power adapter
- HMS IXXAT INpact EIP Slave PCIe interface card installed on the IPC in Mech-Mind Vision System
- Switch
- Ethernet cables

Attention: A KV-8000 model is used in the example below.

Software

- KEYENCE PLC programming software KV STUDIO V11.41
- Mech-Mind Software Suite: Mech-Center 1.5.1 or above, Mech-Vision 1.5.0 or above, and Mech-Viz 1.5.0 or above
- VCI V4 (driver software for HMS IXXAT INpact 40 interface card)
- HMS IPconfig software
- Mech-Mind EDS file:
  - File name: **005A002B003A0100.EDS**
  - Location: `Mech-Mind/Mech-Center/mech_interface/EthernetIP`
- Example programs:
  - CameraSignalsMove.mod
  - CameraTest.mod
  - ObtainPose.kfb

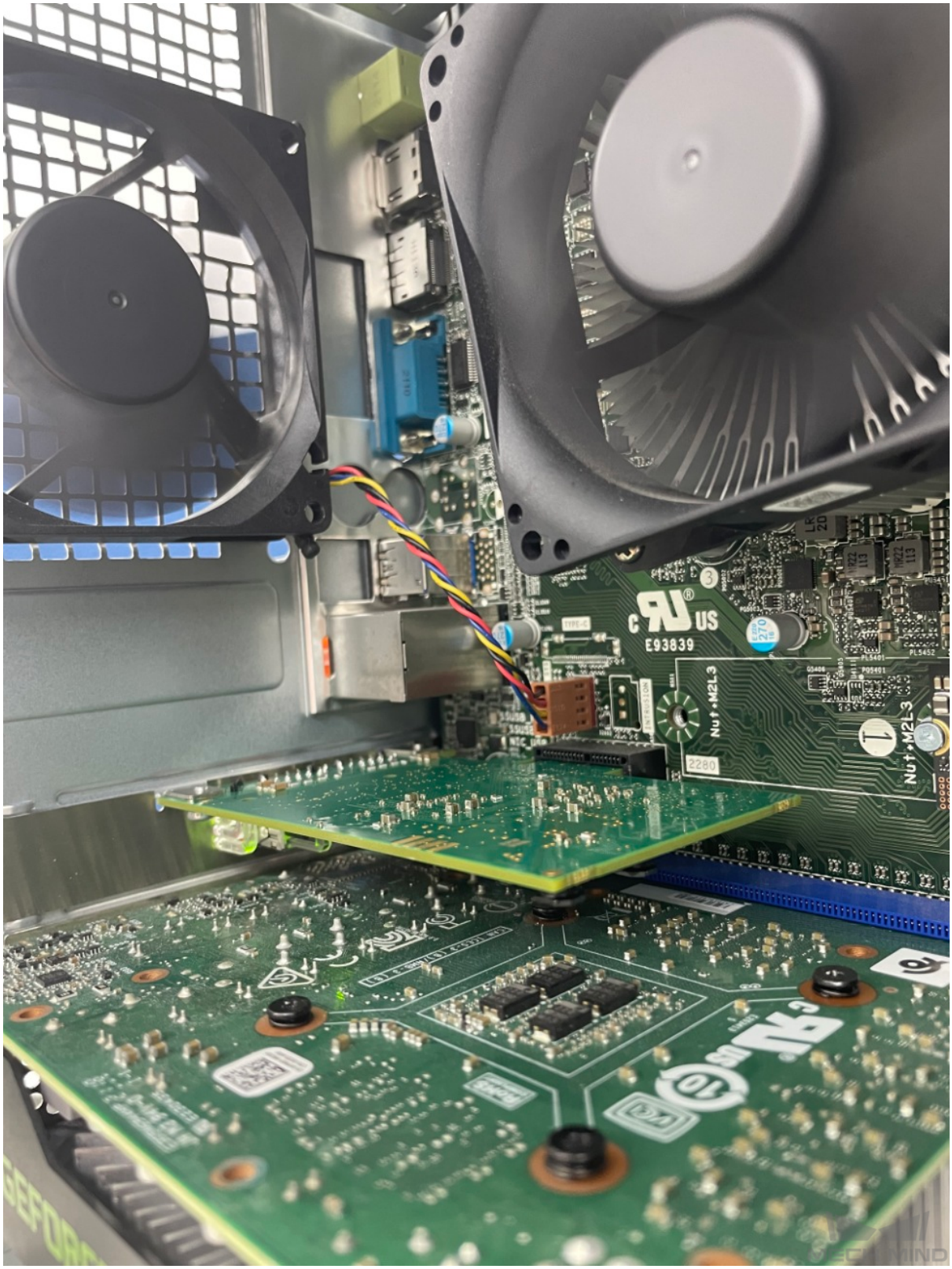
  The files are stored in `Mech-Mind/Mech-Center/mech_interface/documents/CN/基恩士 EtherNet IP 编程指南`. Please copy and paste all files to the computer with KV STUDIO installed.

Note: Connect the Mech-Mind Vision System IPC, computer with KV STUDIO installed, and PLC to the same router.

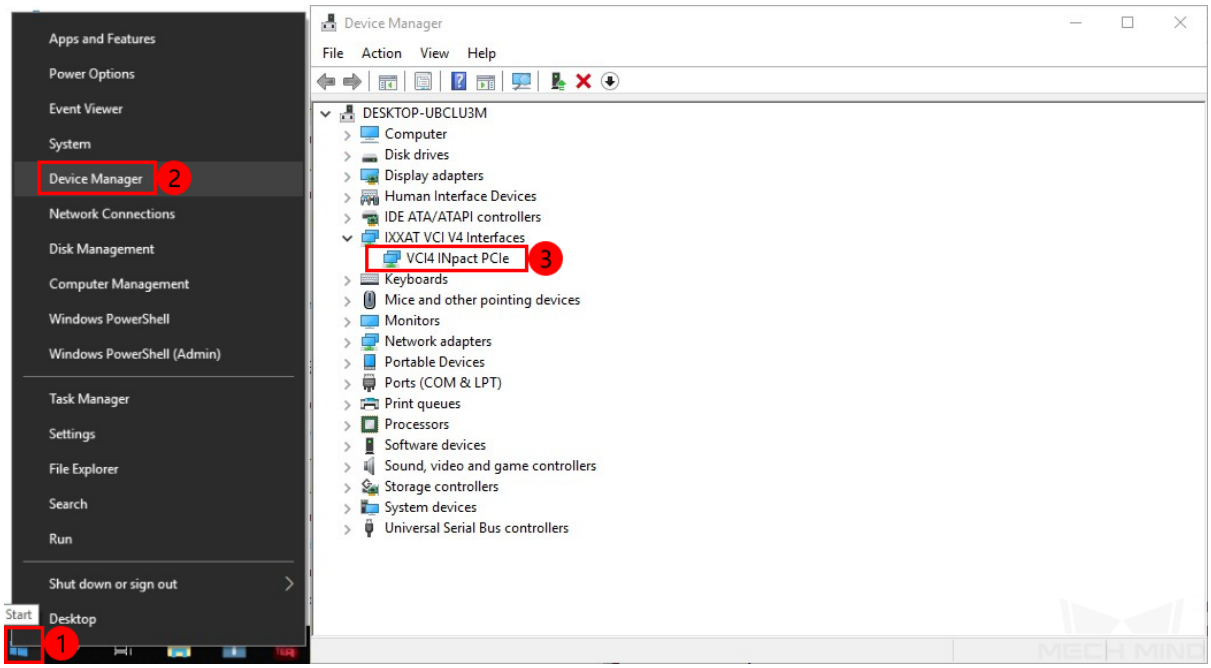
2.6.3 Configure IPC and Initiate Communication

Check PCI-e Card and Driver Software

1. Please make sure that the INpact EIP Slave PCIe interface card has been pressed into the PCI-e slot of the IPC, as shown below.

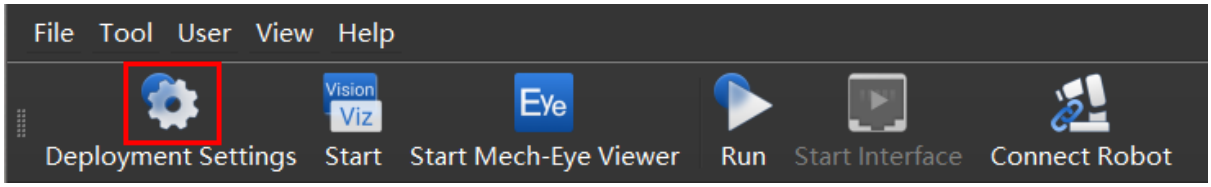

E93839
Nut•M2L-3
Nut•M2L-3

2. Start the IPC, go to *Start* → *Device Manager* and check if the driver software **VCI4 INpact PCIe** has been installed.

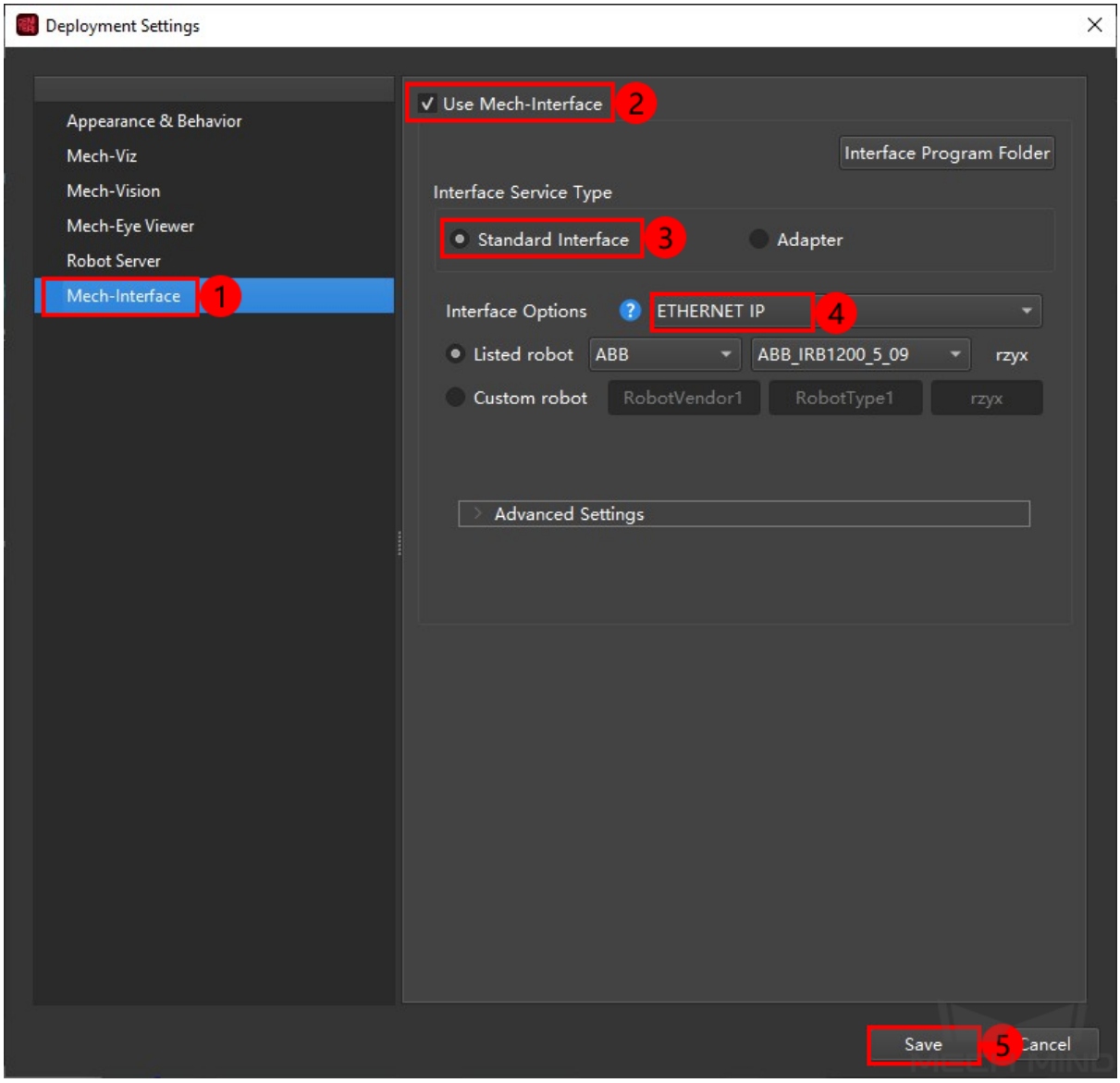


Configure Mech-Interface in Mech-Center

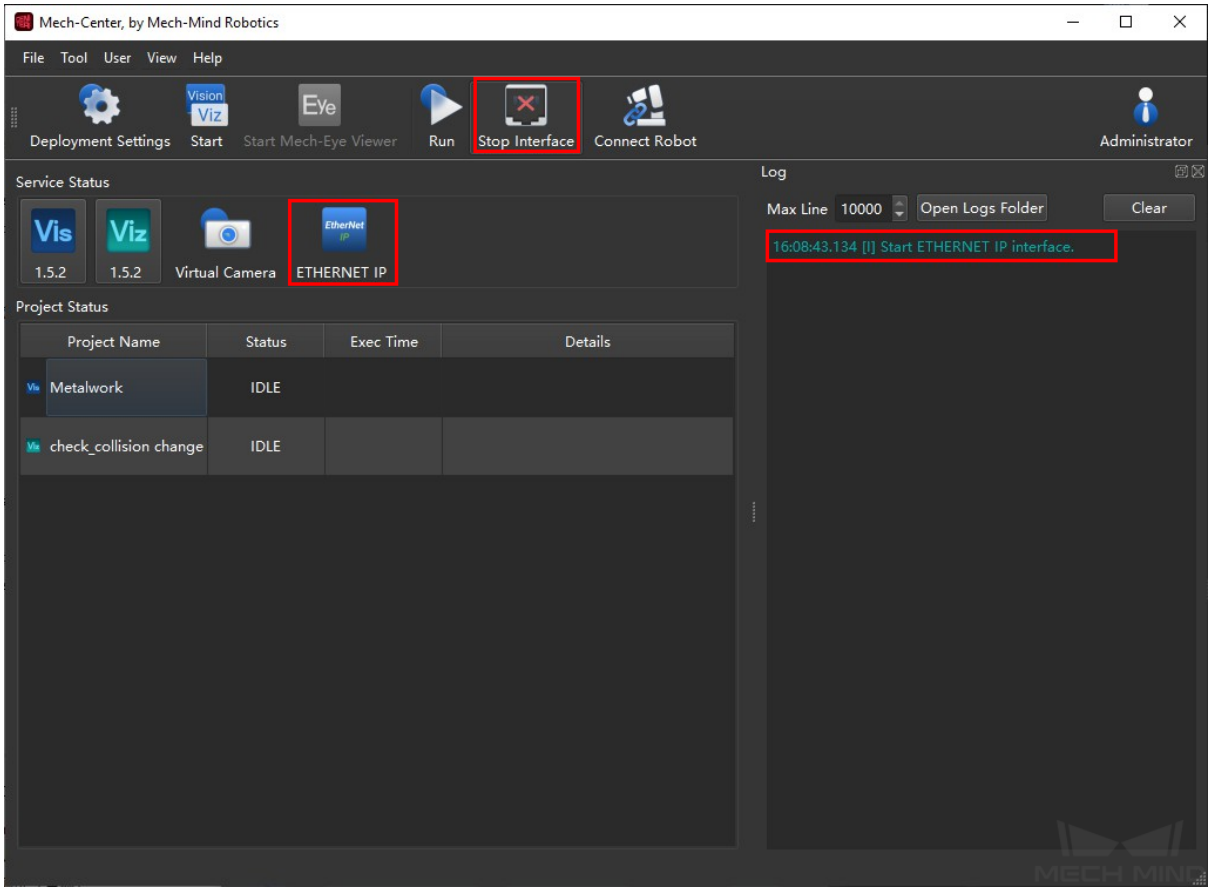
1. Open Mech-Center, and click on *Deployment Settings*.



2. Go to **Mech-Interface**, check **Use Mech-Interface** and select *Standard Interface* → *ETHERNET IP*. Click on *Save* to complete configurations.



3. Click on *Start Interface* in the Toolbar. Then an ETHERNET IP icon will display in the service status bar.

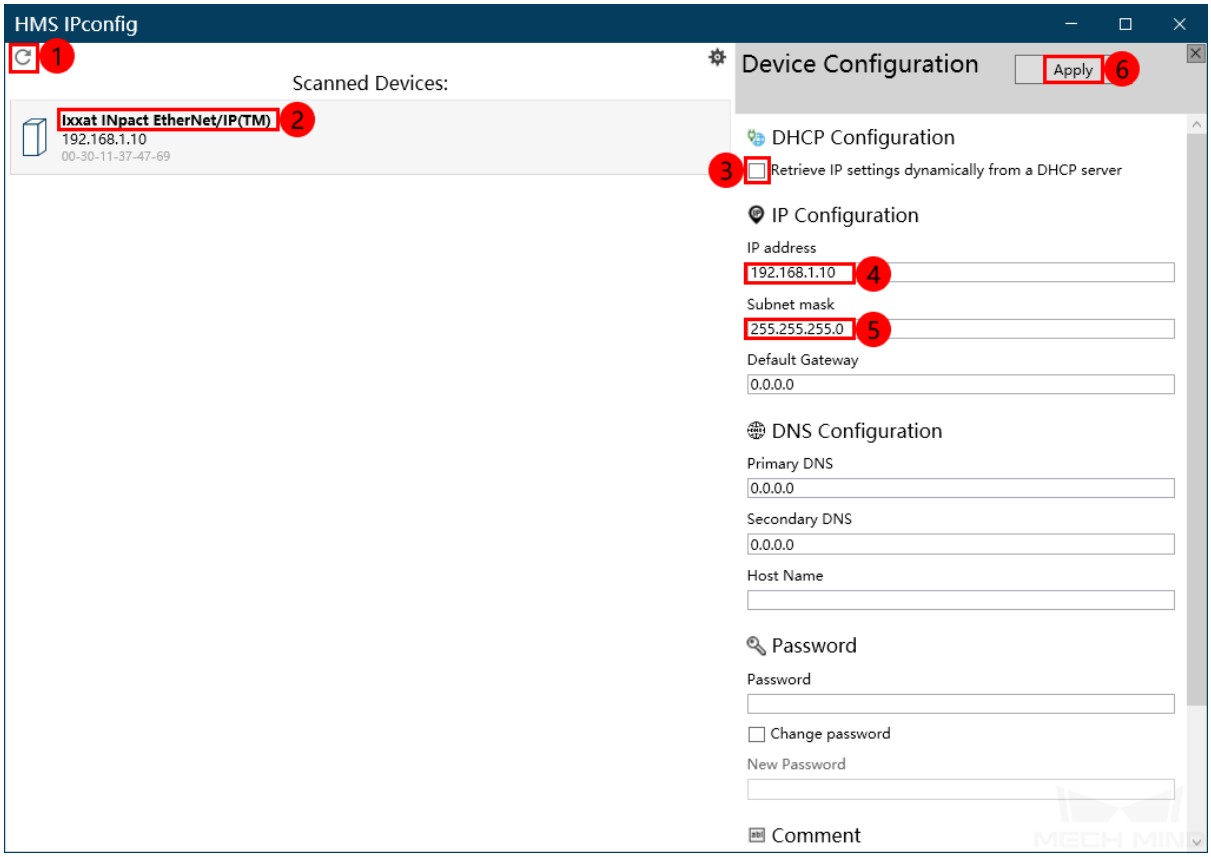


Configure IP address of the PCI-e Card

1. Download and install **HMS IPconfig software** on the IPC first. Use an Ethernet cable to connect the network ports of the IPC and the INpact EIP Slave PCIe.

   > **Attention:** After configurating the IP and initiating communication successfully, the Ethernet cable used here can be removed.

2. Open HMS IPconfig, click on  and select **Ixxat INpact EtherNet/IP(TM)**. Then uncheck **Retrieve IP settings dynamically from a DHCP server** and enter the IP address and subnet mask, as shown below. After configuration, click on *Apply*.

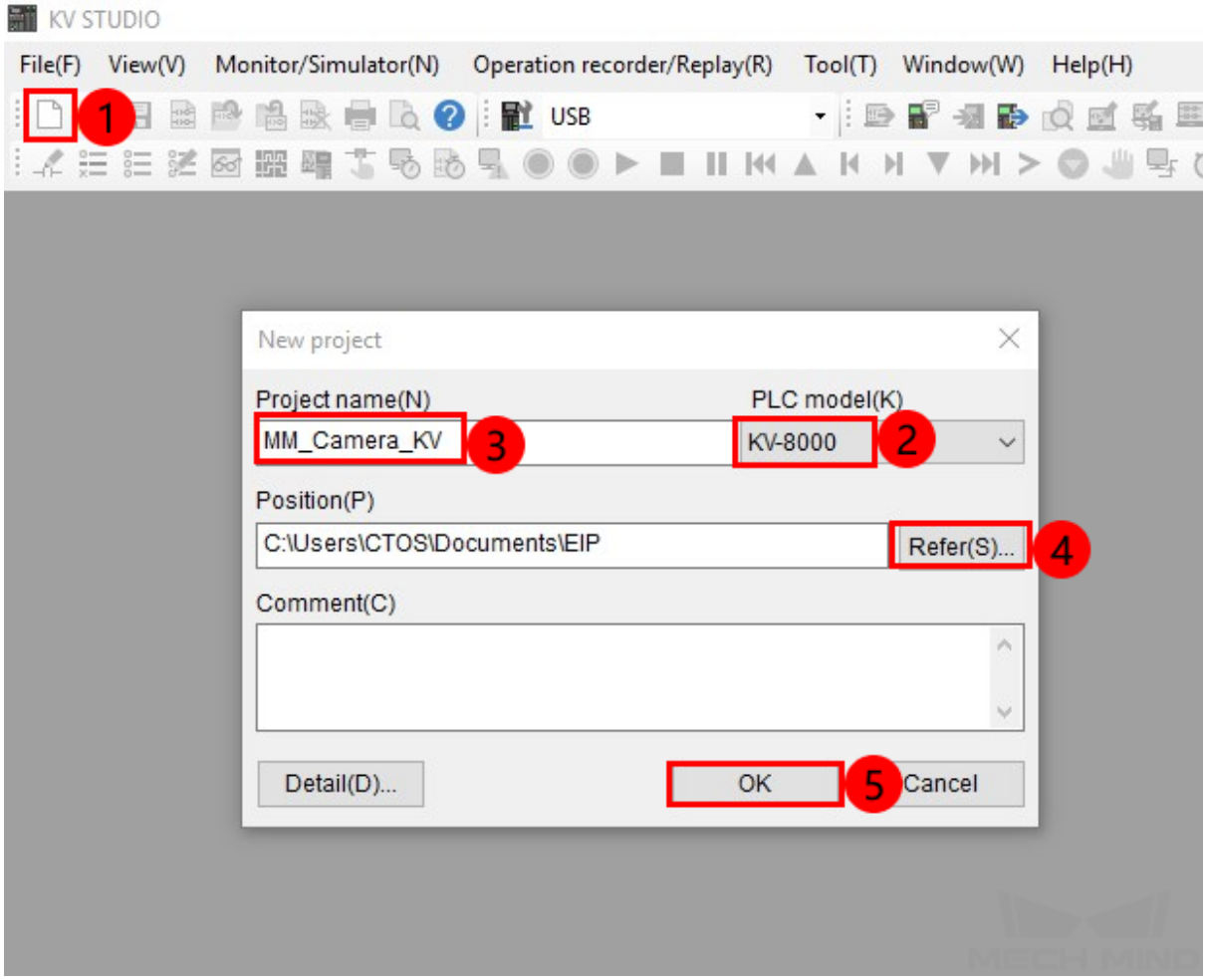


Attention: The IP address should be the same as which is configured in the PLC.

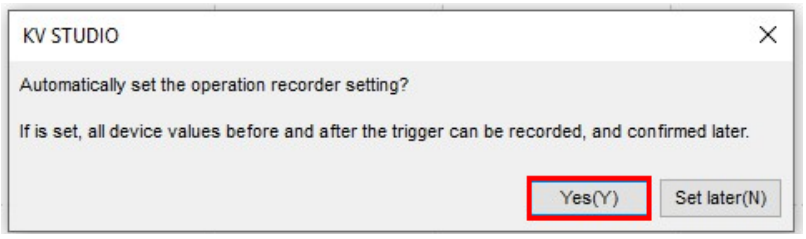
2.6.4 Install EDS file and Configure Communication

Create PLC Project

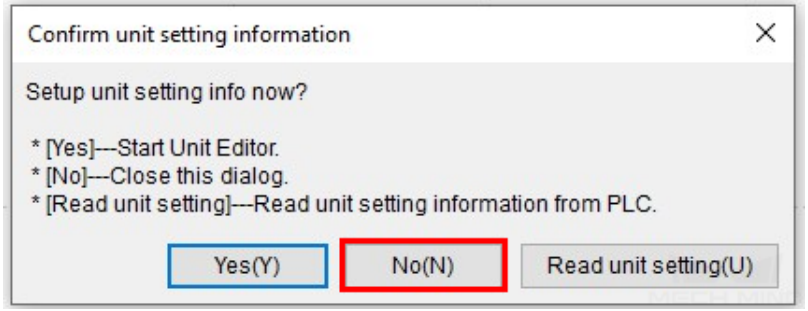
1. Open KV STUDIO, click on  , select the **PLC model** according to the model in use, and name the project. Then click on *Refer* to select a location to store the project, and click on *OK* to save settings.



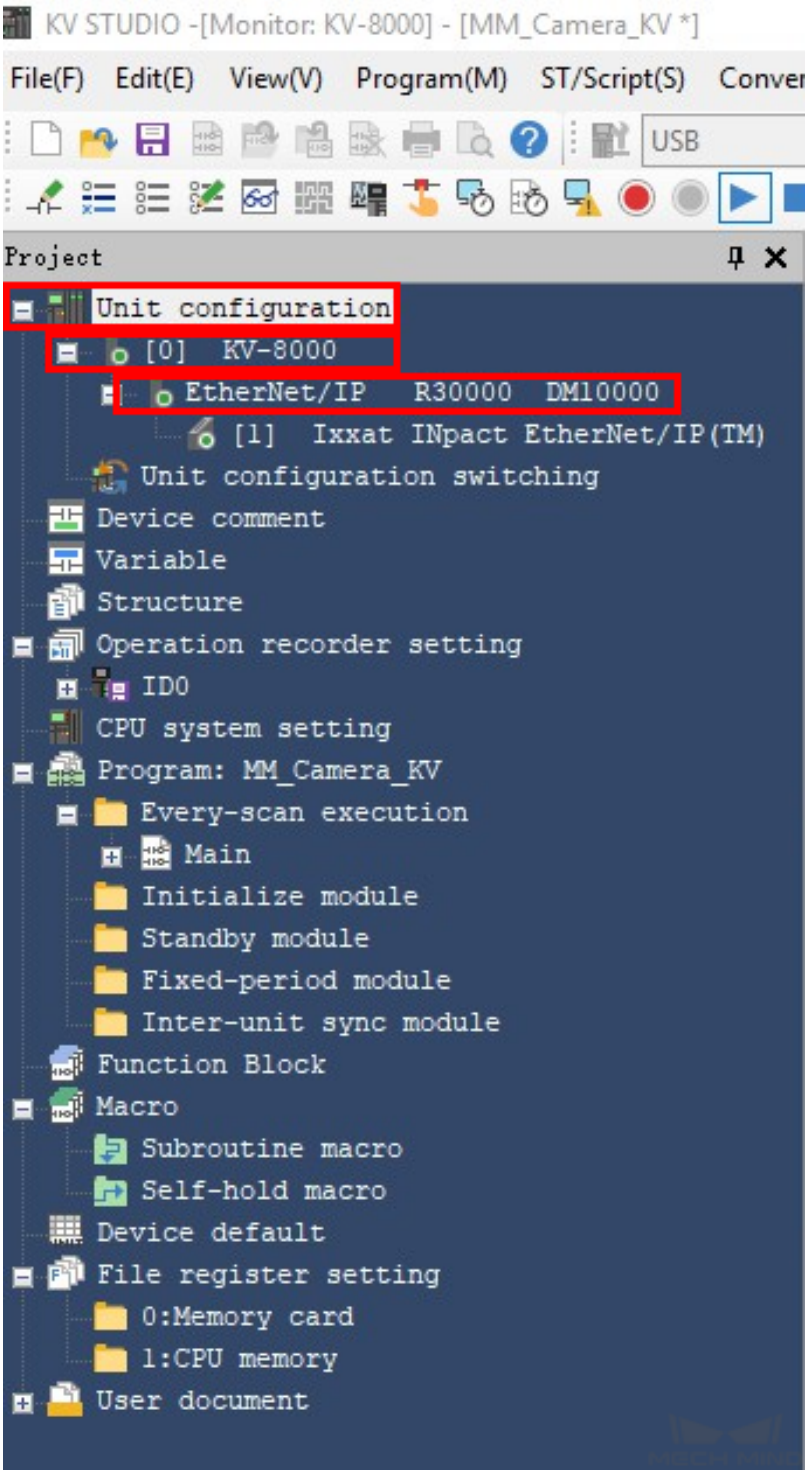
2. If the message **"Automatically set the operation recorder setting?"** pops up, select *Yes*.



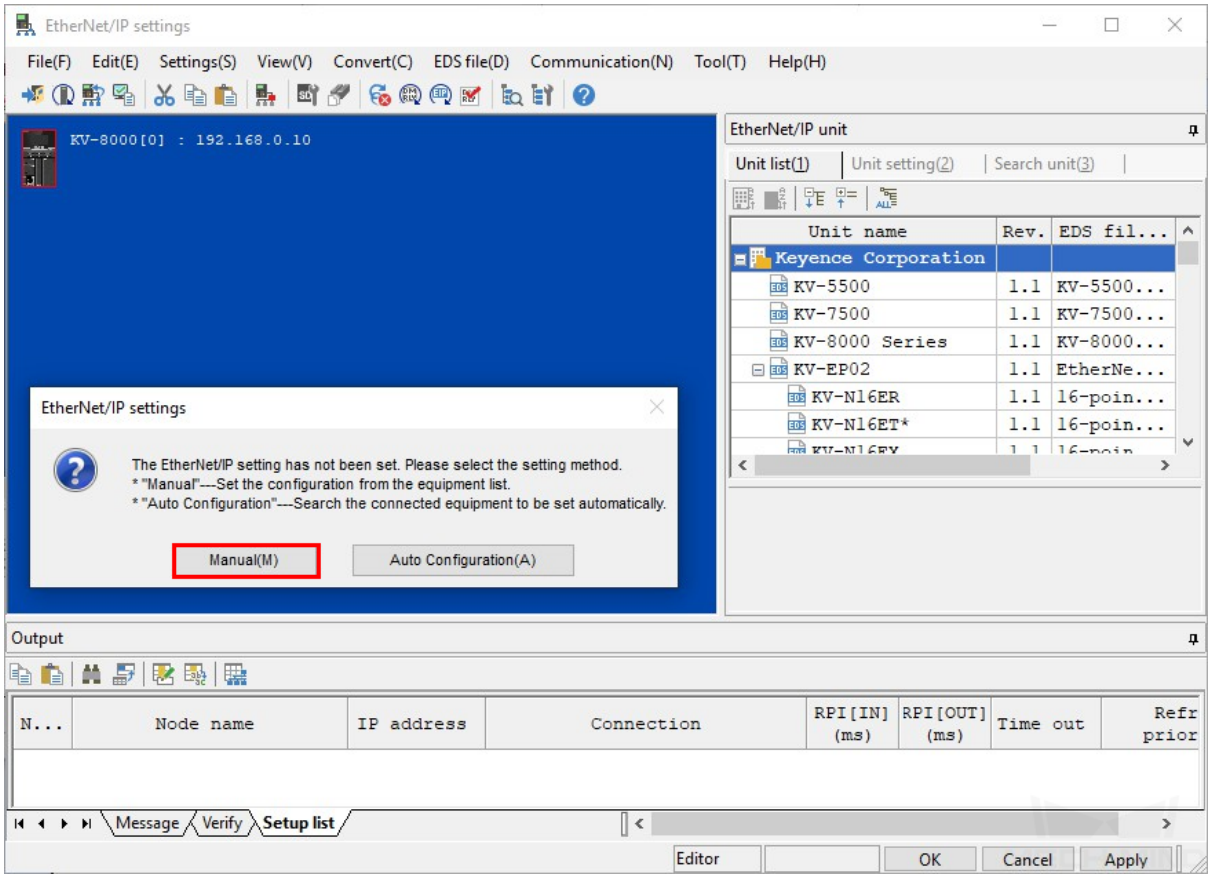
3. If the message **"Setup unit setting info now?"** pops up, select *No*.



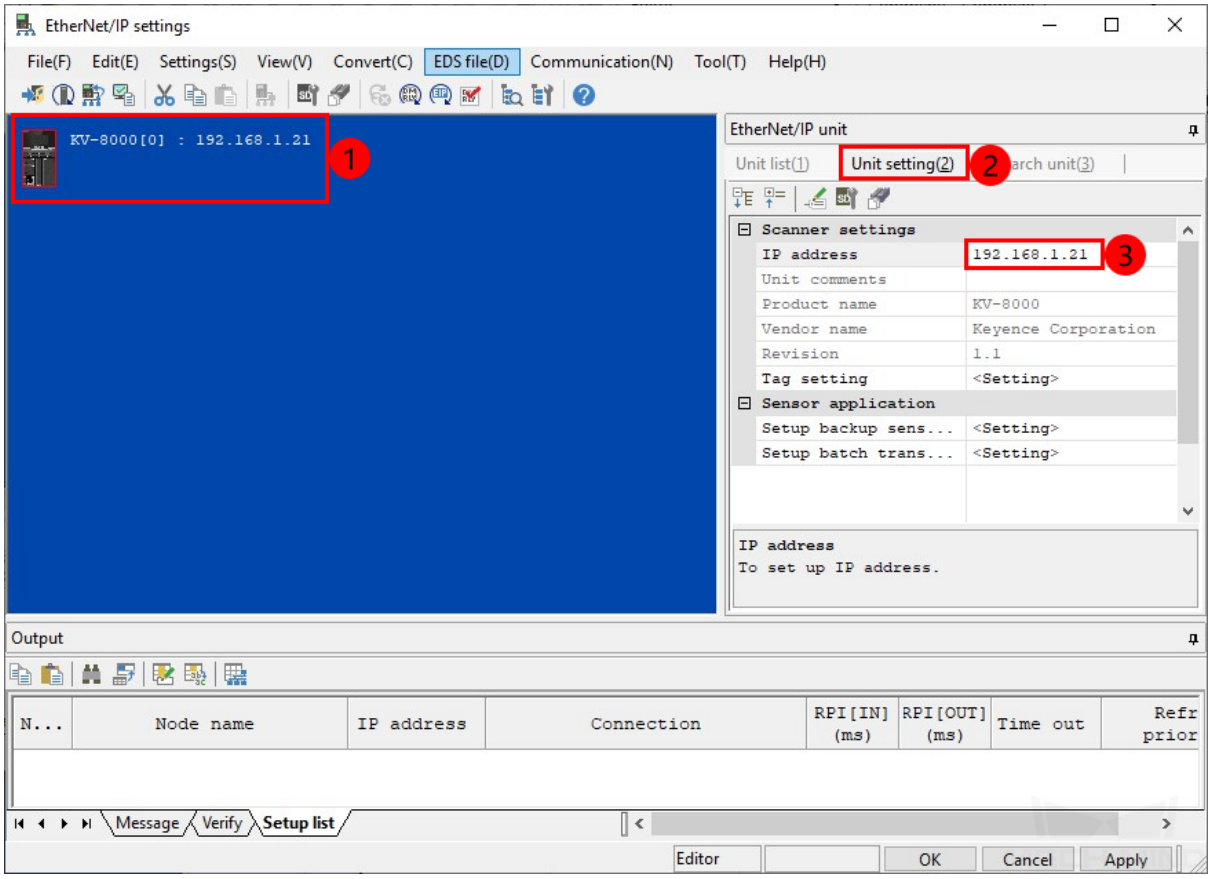
4. Go to *Unit configuration→ KV 8000*, double click on **EtherNet/IP R30000 DM10000**.



5. An **EtherNet/IP settings** window will appear as shown below. Select *Manual* in the pop-up window.

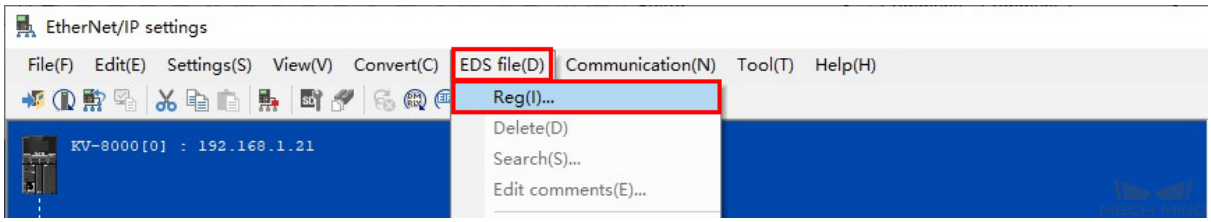


6. Select **KV-8000**, click on *Unit setting* and then set the **IP address** of the PLC.

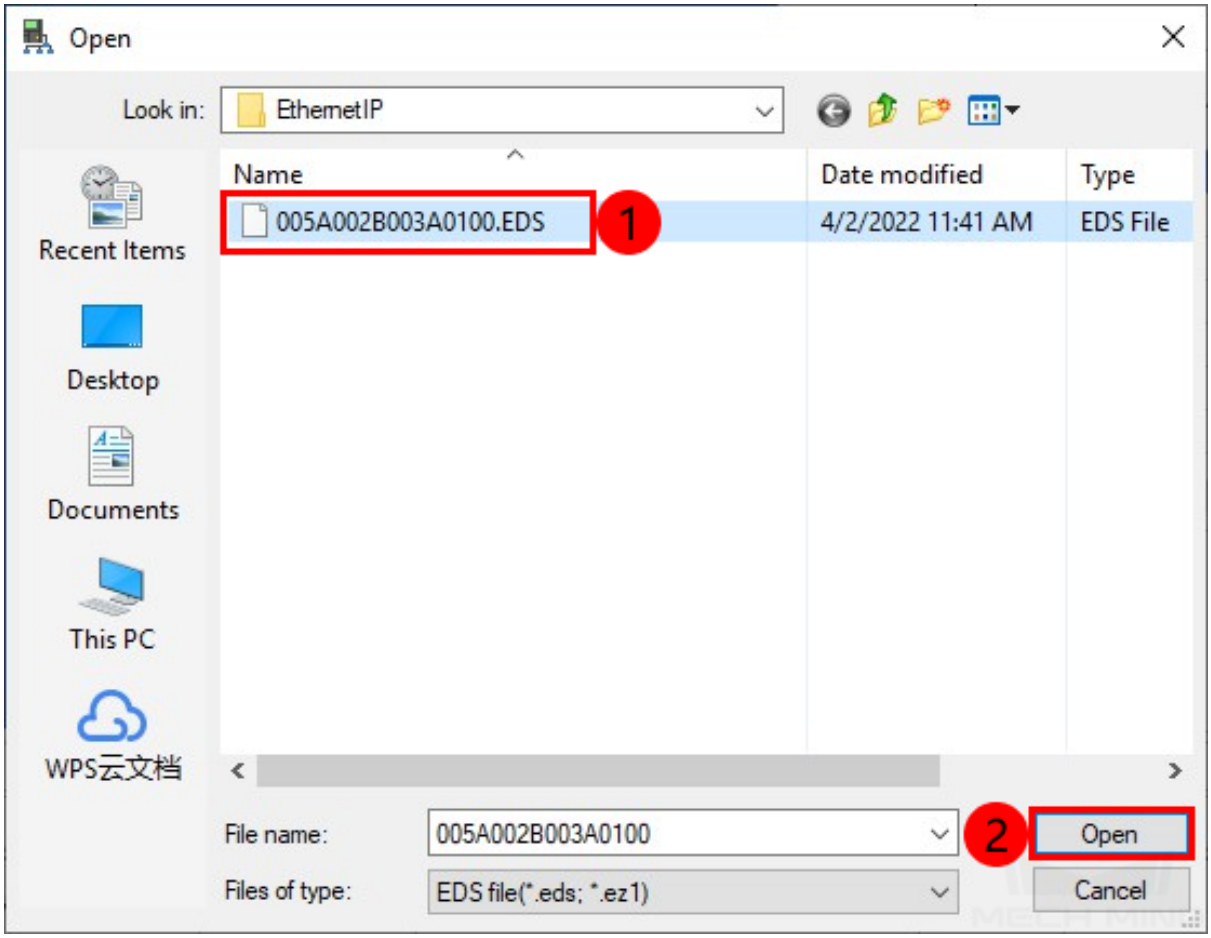


Install EDS File and Configure Network

1. Click on *EDS file* in the **EtherNet/IP settings** window, and select **Reg**. An **Open** window will pop up.

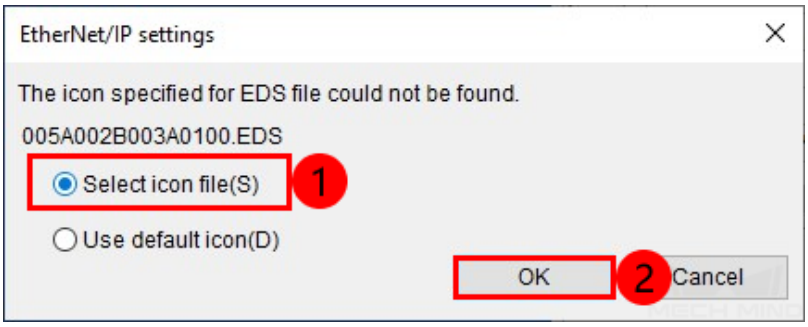


2. Locate and select the EDS file and click on *Open*.

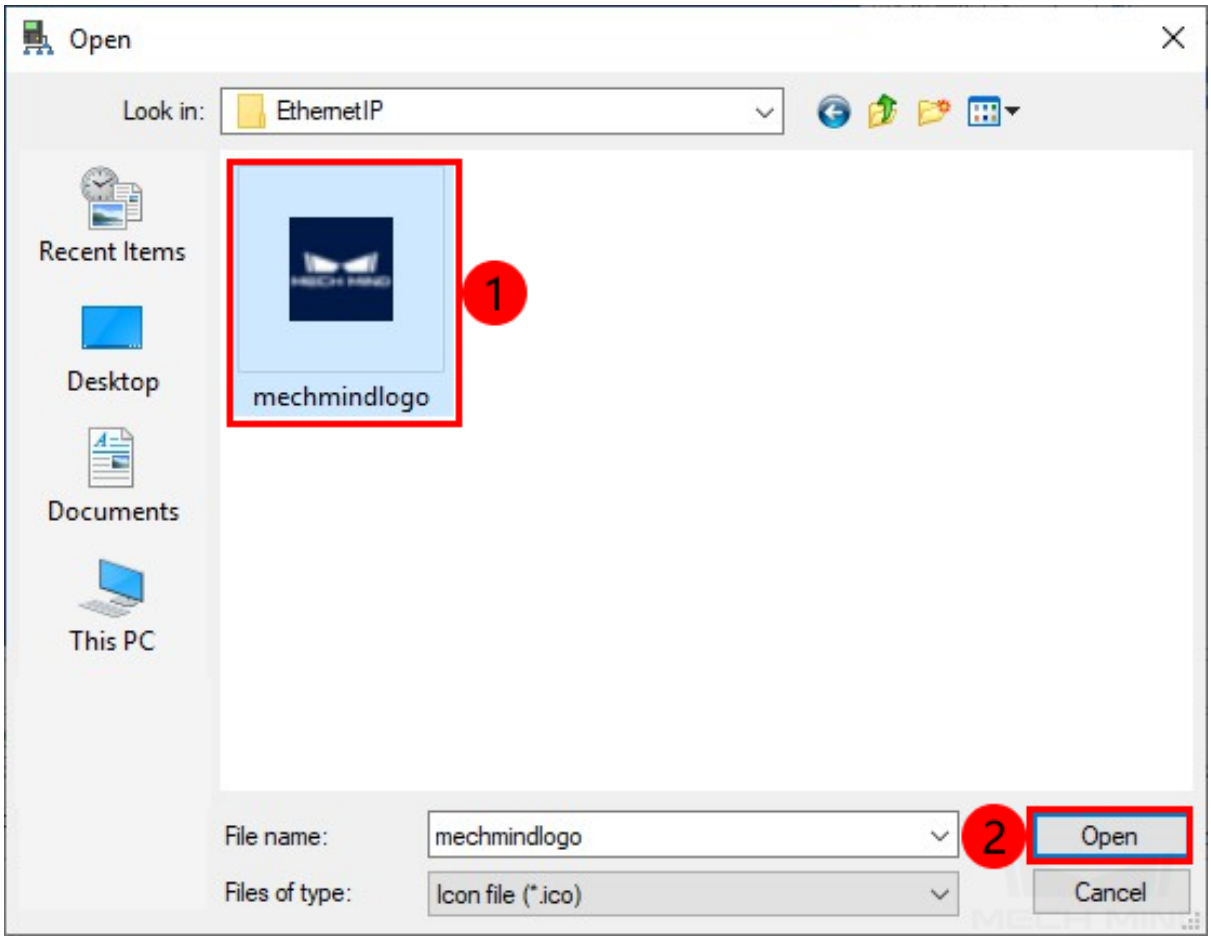


> **Attention:** The EDS file is stored in the folder where Mech-Center is installed. The default path is `Mech-Mind/Mech-Center/mech_interface/ETHERNETIP`. If KV STUDIO is not installed on the same IPC where Mech-Center is installed, you can copy and paste the **ETHERNETIP** folder to the PC with KV STUDIO installed.

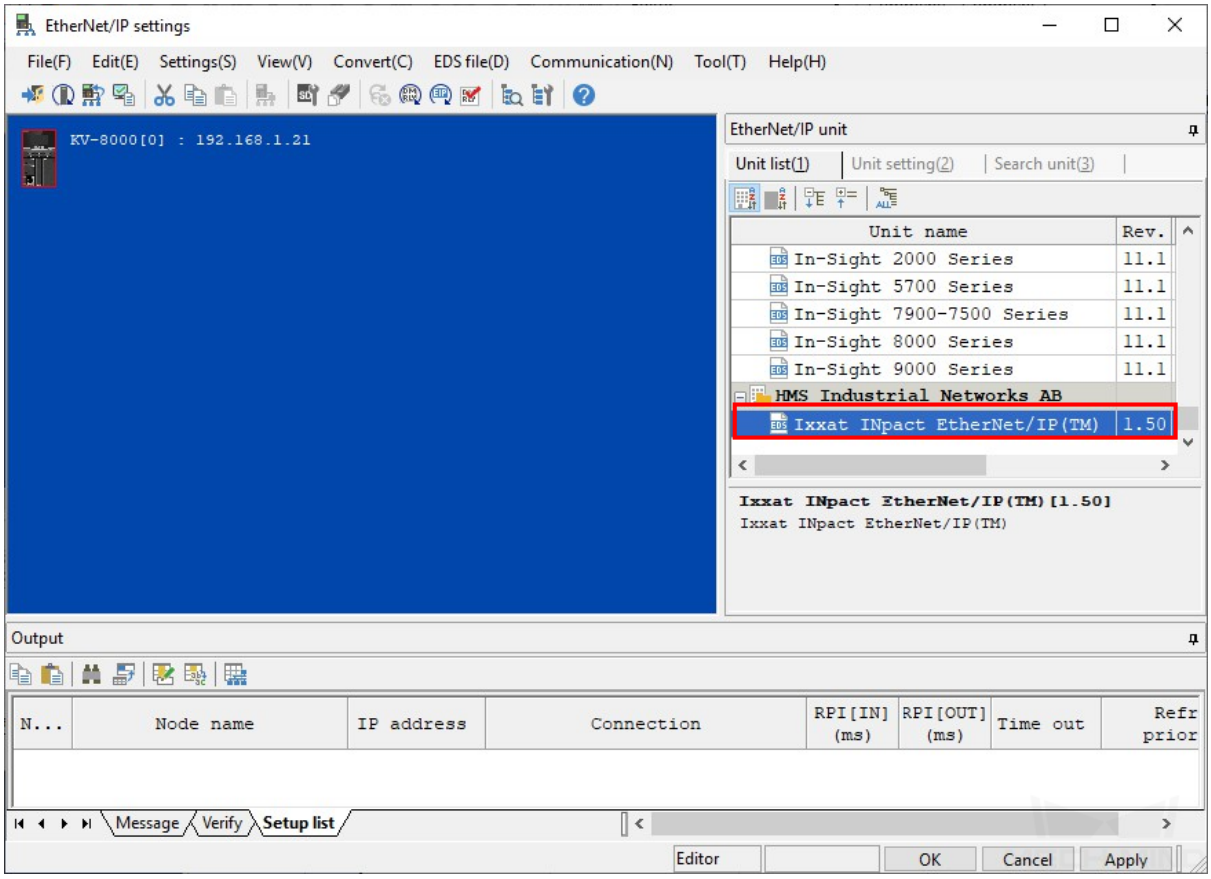
3. If the message **"The icon specified for EDS file could not be found."** pops up, select **Select icon files** and then click on *OK*.



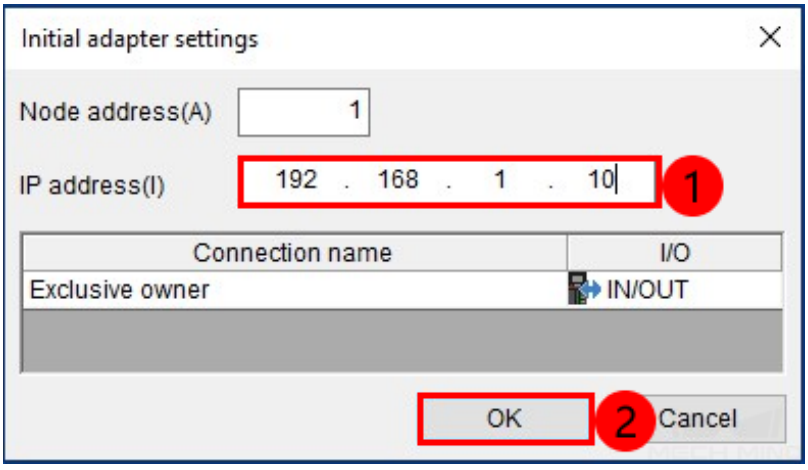
4. After selecting the icon files, click on *Open*.



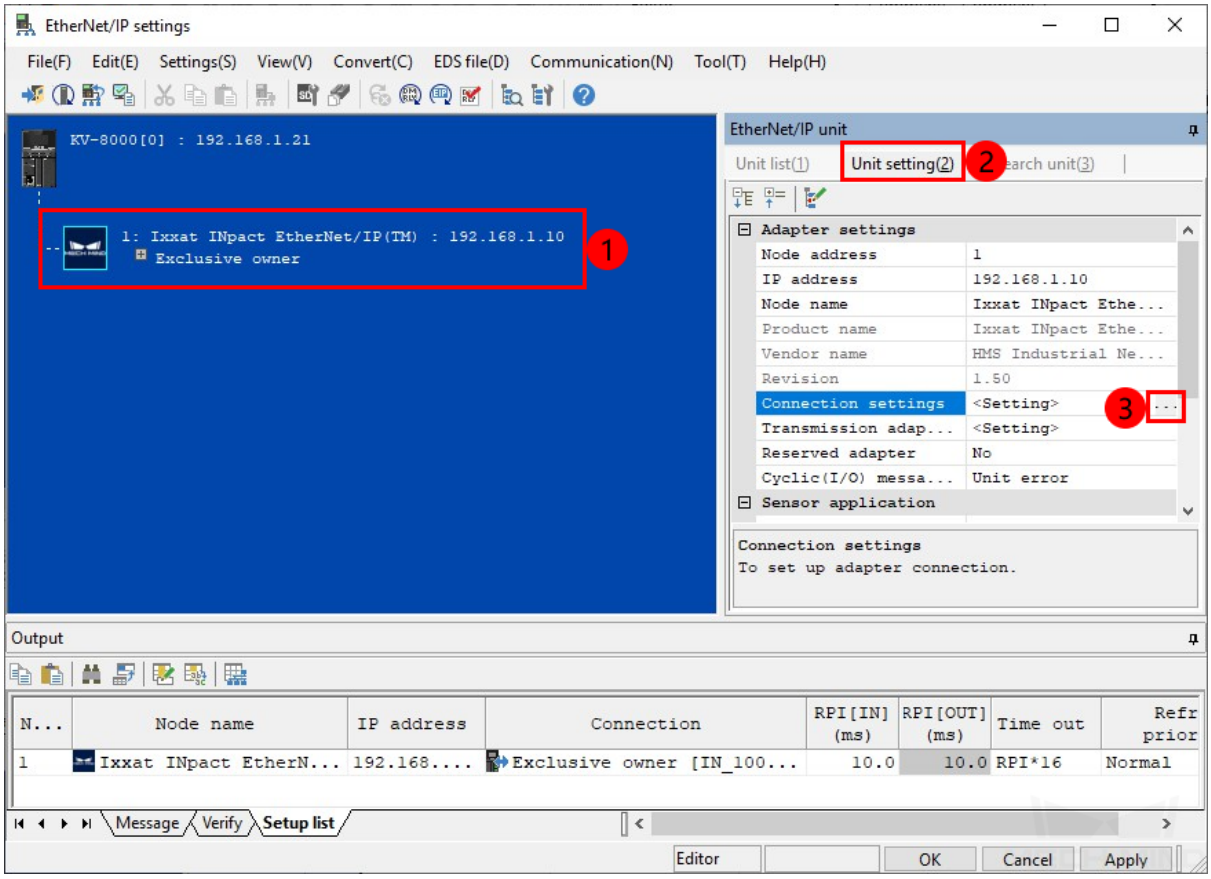
5. If the EDS file is registered successfully, **Ixxat INpact EtherNet/IP(TM)** will appear in the Unit list. Double click it to connect to the EtherNet/IP network.



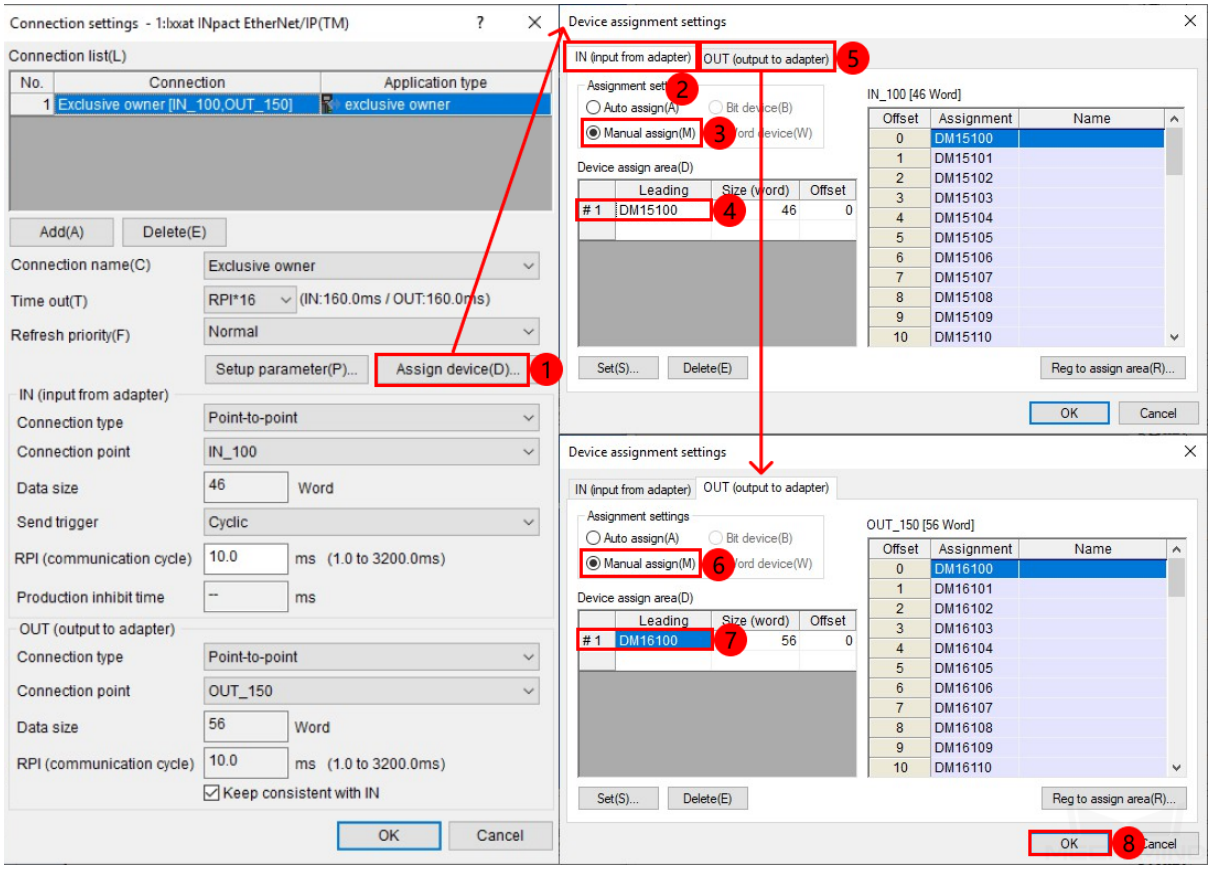
6. Set the IP address in the **Initial adapter settings** window, set the IP address of the IPC as same as which is set in HMS IPconfig, and then click on *OK*.



7. Select **Ixxat INpact EtherNet/IP(TM)**, click on *Unit setting(2)* and then click on .. next to **Connection settings**.

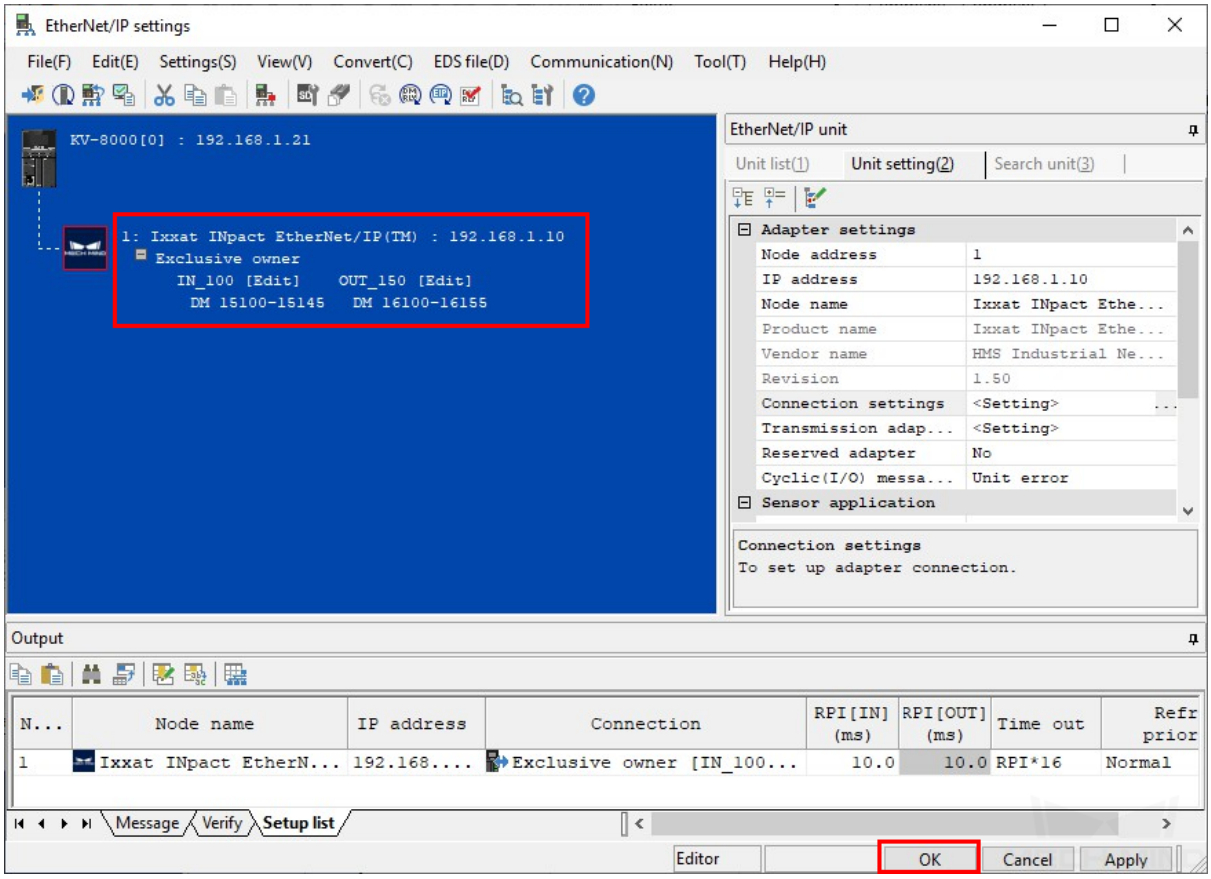


8. Now you can see a **Connection settings** window. Click on *Assign device* and then assign the **IN** and **OUT** manually, as shown below. Go back to the **EtherNet/IP settings** window after assigning.

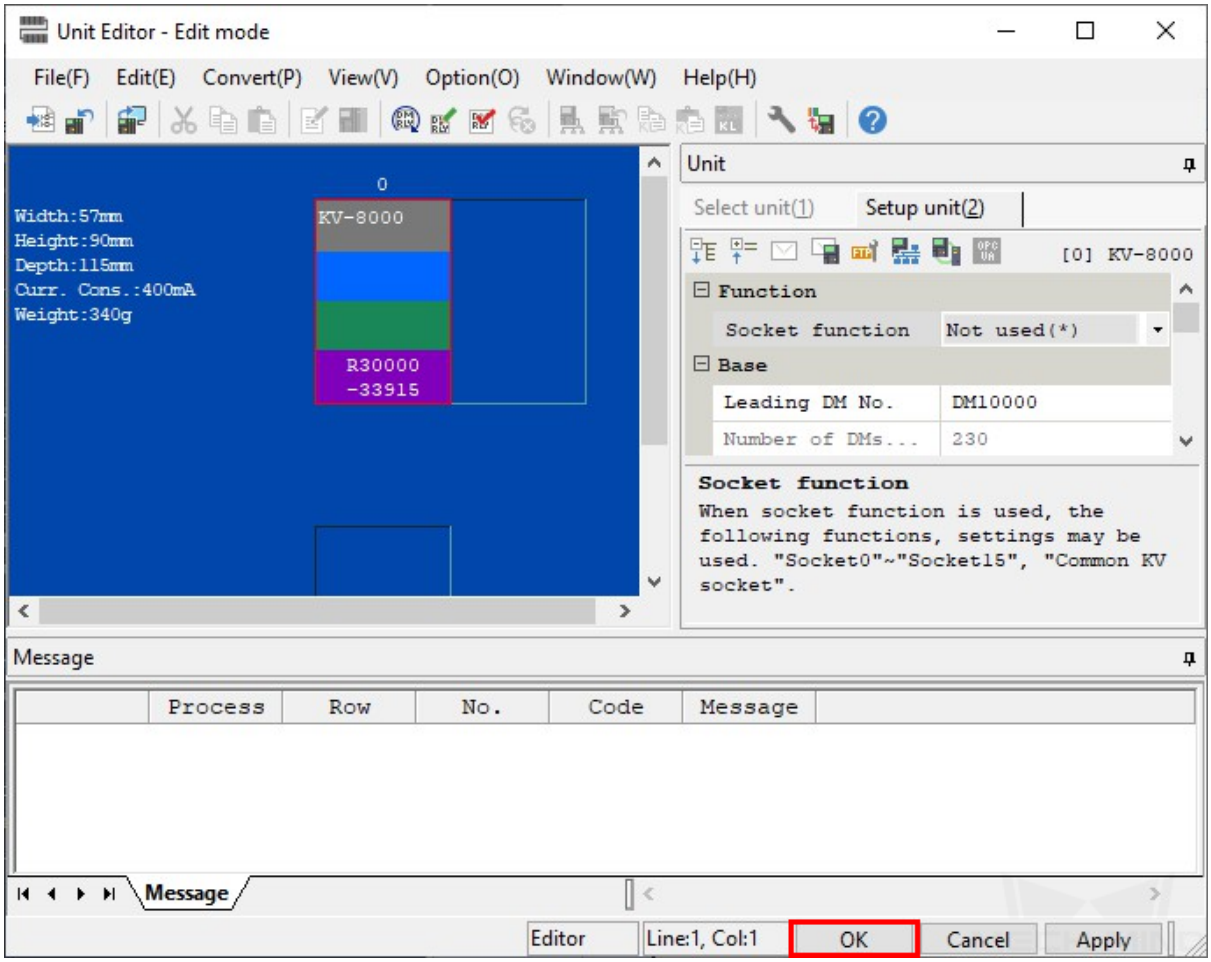


Note: The Input and Output in the figure are applicable to this example only.

9. Now the information of the server appear in the **EtherNet/IP settings** window. Click on *OK*.

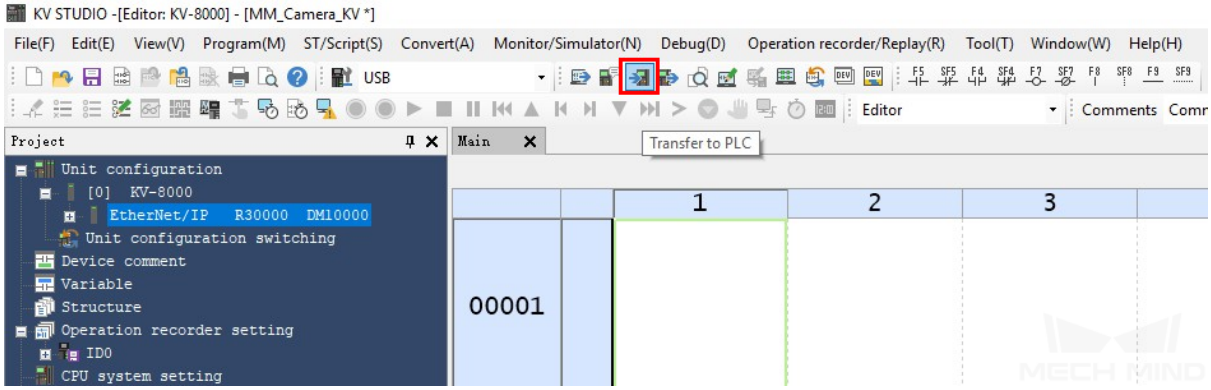


10. Click on *OK* in the **Unit Editor** window.

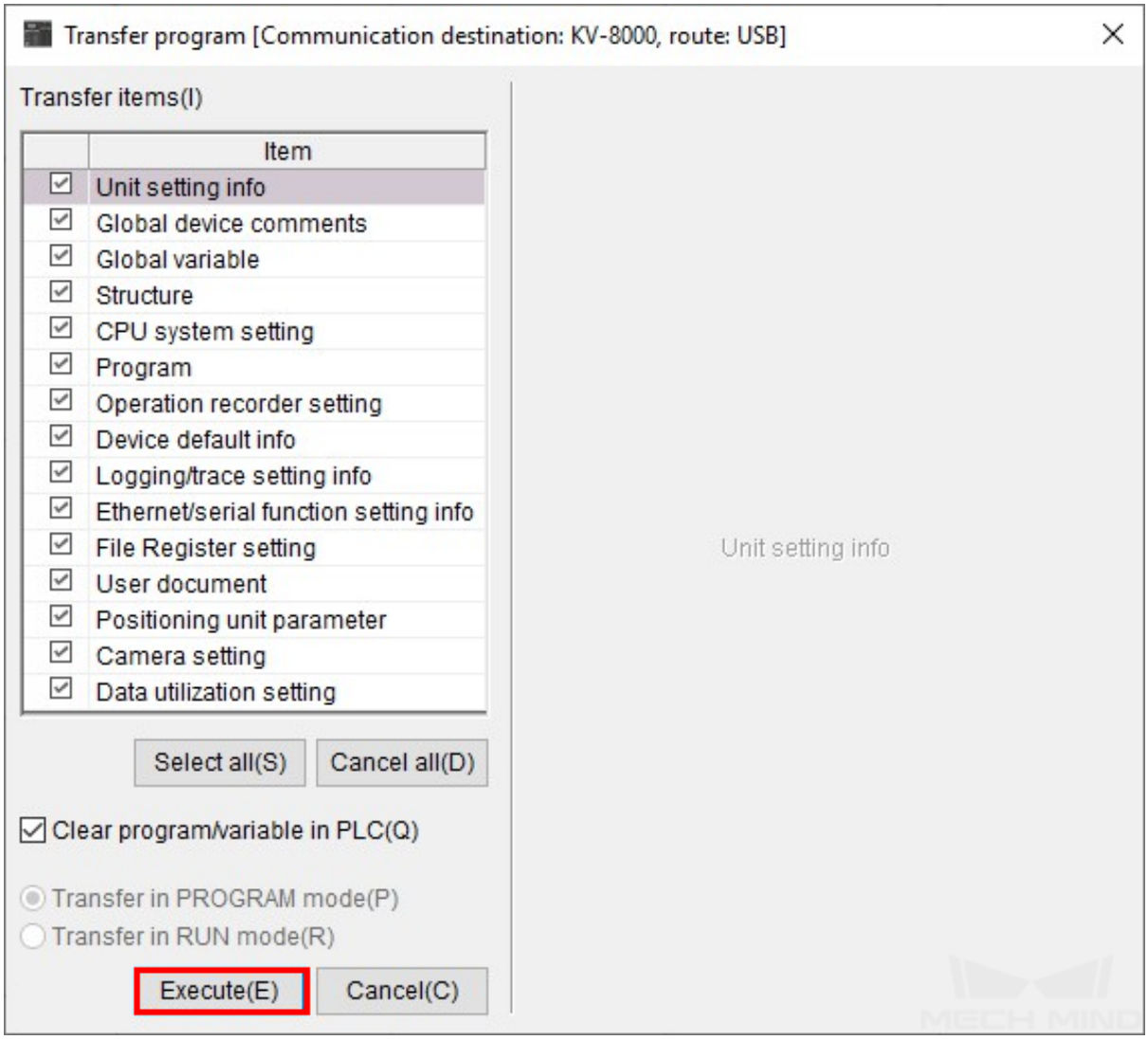


Download Hardware Configuration to PLC

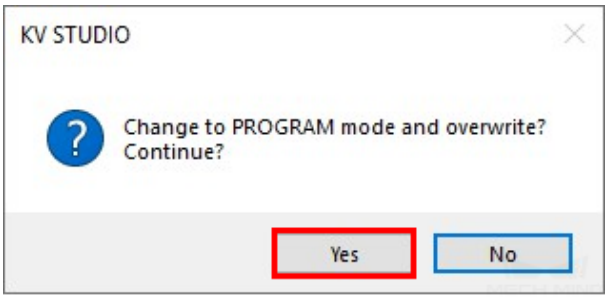
1. Go back to the main interface of KV STUDIO, click on  button in the menu bar.



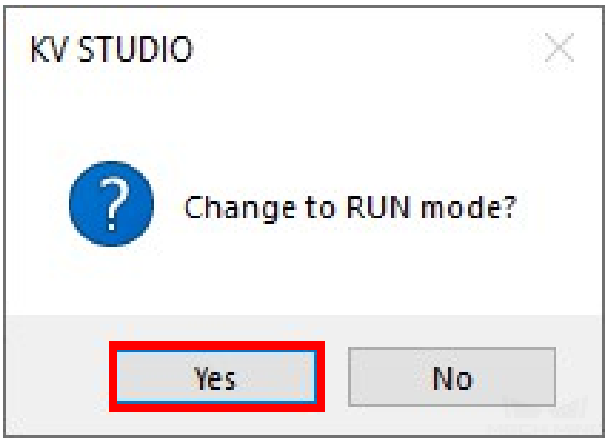
2. Keep the default settings, and click on *Execute* in the **Transfer program** window.



3. Select *Yes* in the pop-up window showing the message **"Change to PROGRAM mode and overwrite?"** .

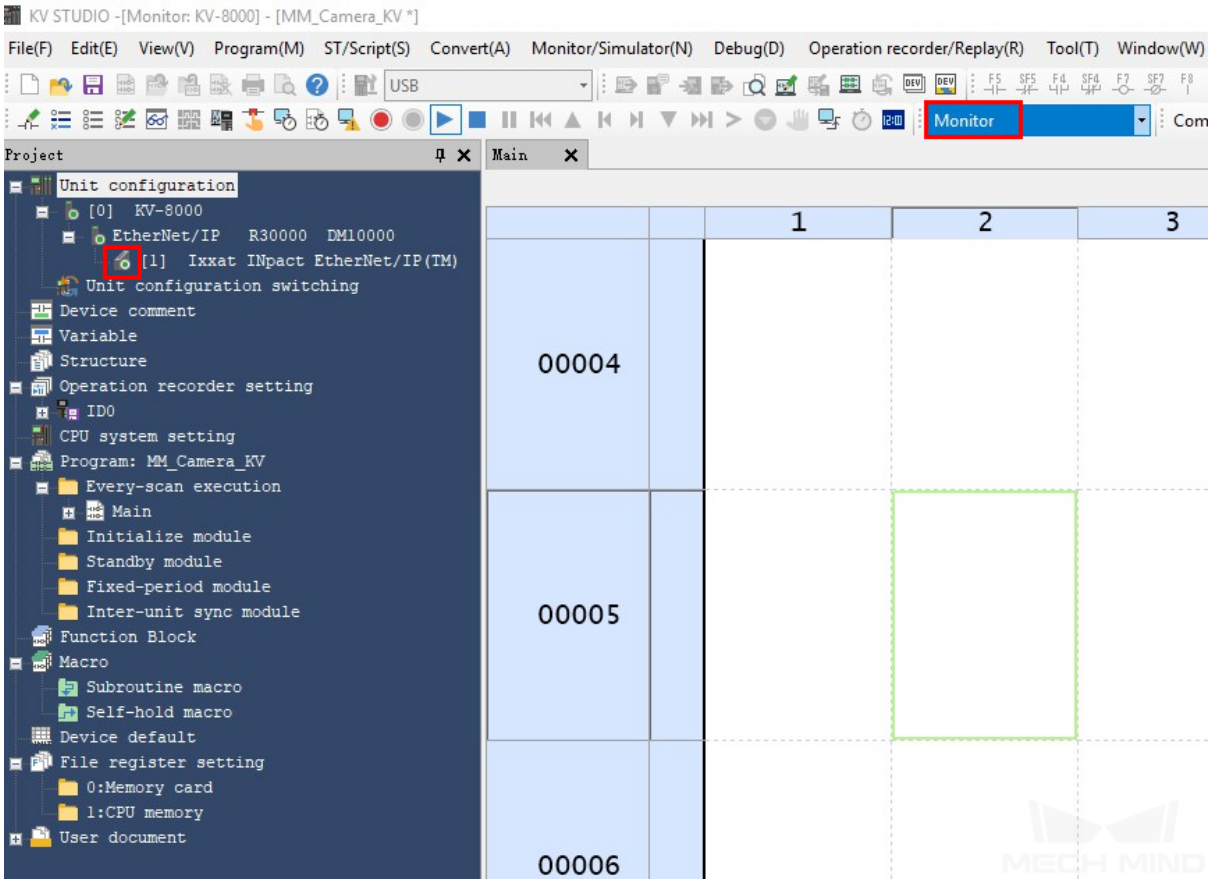


4. Select *Yes* in the pop-up window showing the message **"Change to RUN mode?"** .



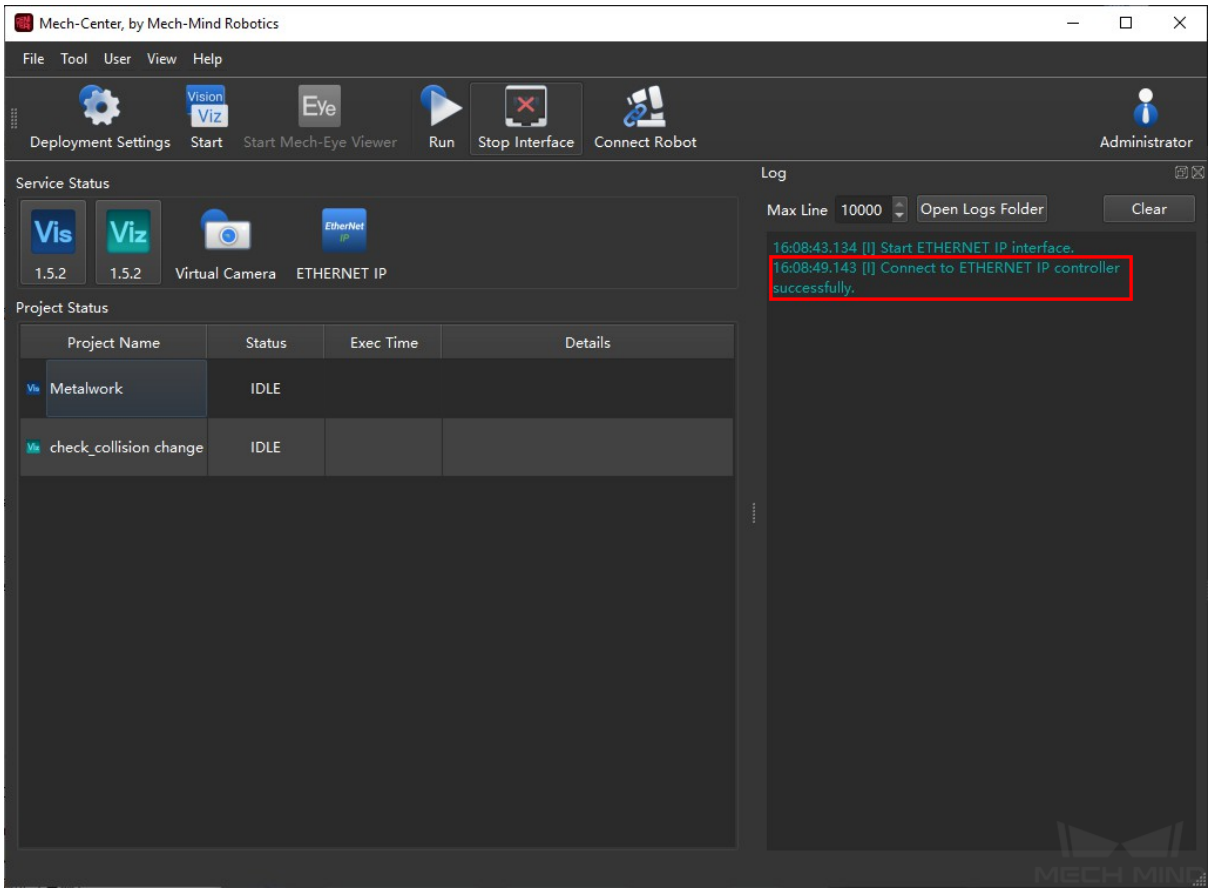
Check Communication

1. If the PLC is connected successfully, the status of **Ixxat INpact EtherNet/IP(TM)** will be displayed as online in **Monitor** mode.



2. The PLC is successfully connected to Mech-Center if the following message is displayed in Mech-Center **Log** panel:

Connect to ETHERNET IP controller successfully



Note: If you don’t see this log message, please check if:

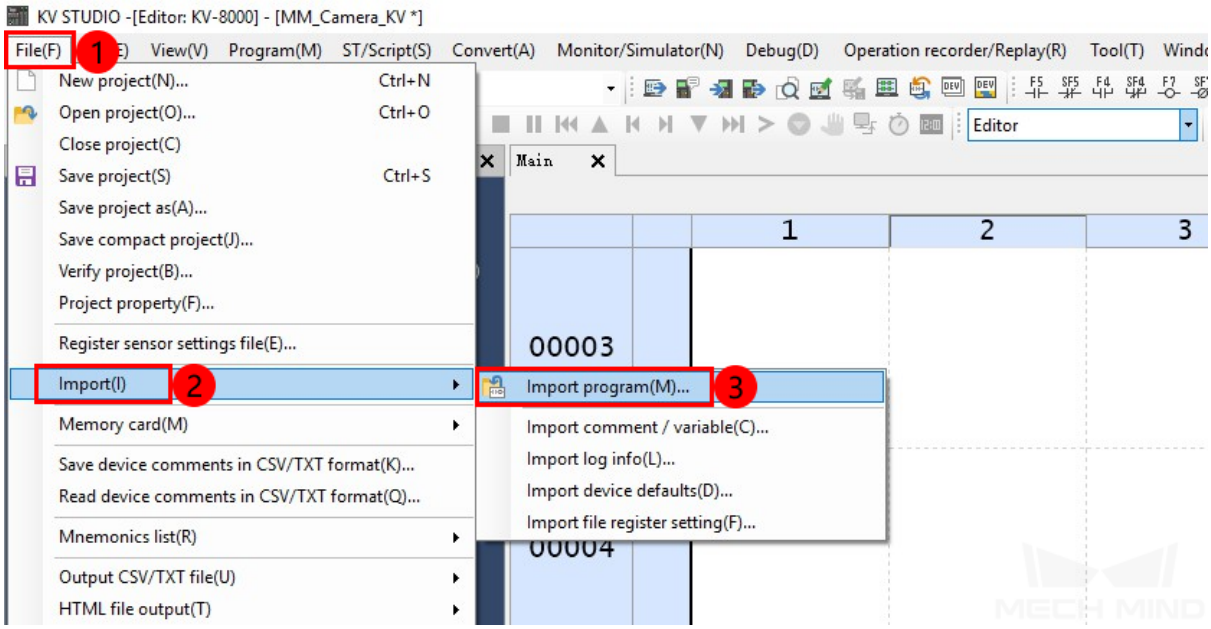
- The hardware are properly connected;
- If Mech-Interface has been started by clicking on *Start Interface* in the Toolbar;
- If the hardware configuration has been downloaded to the PLC.

2.6.5 Import Example Program and Download to PLC

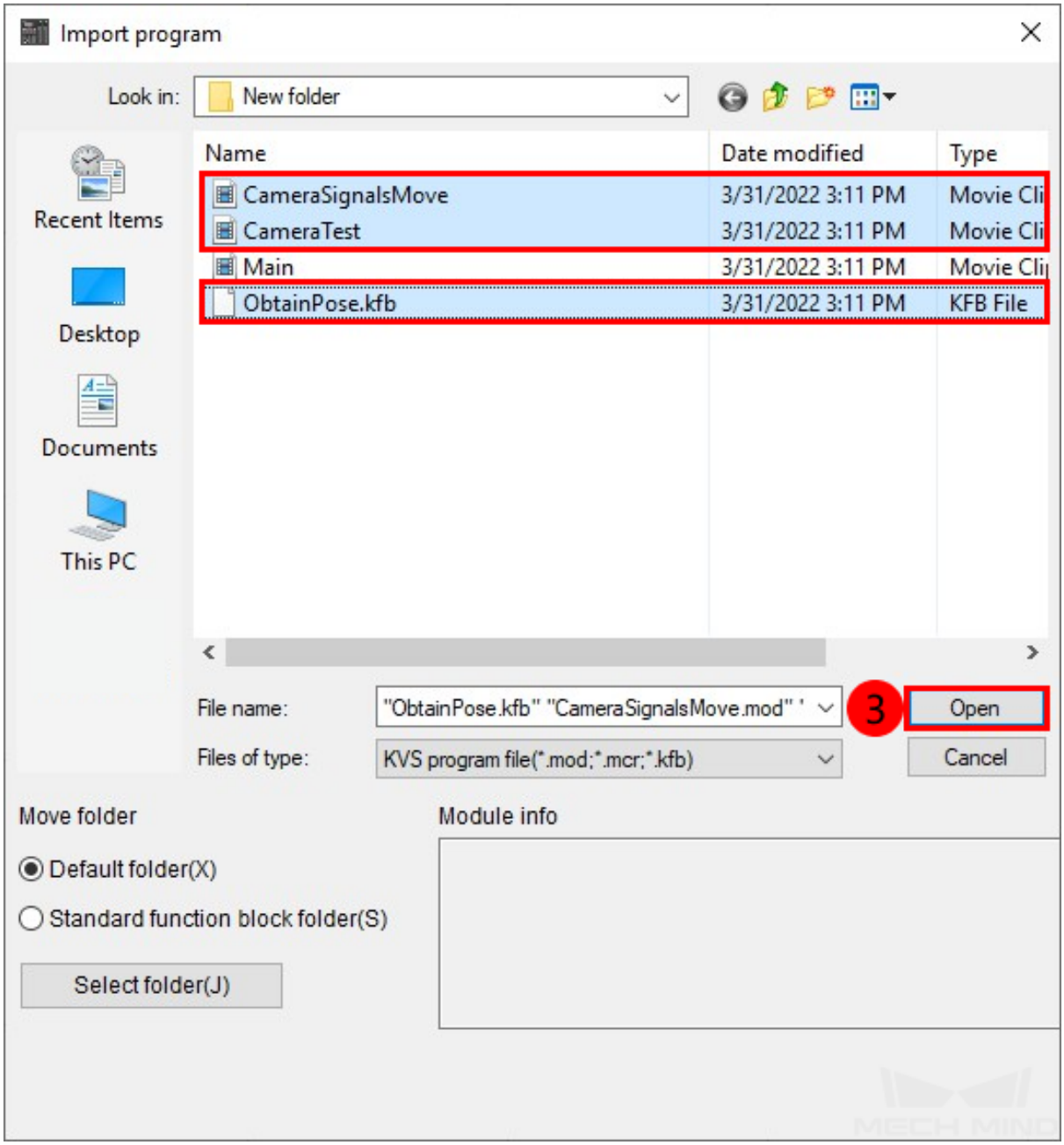
Note: Before you add the example program to a project already in use, it is recommended to import it to a new project and test it first. In the following steps, the project created earlier is used to import and test the example program.

Import Example Program Files

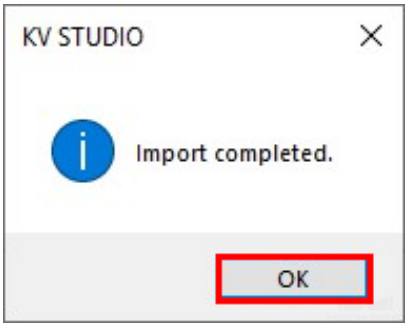
1. In the main interface of KV STUDIO, go to *File* → *Import* → *Import program*.



2. In the pop-up **Import program** window, select the **CameraSignalsMove.mod**, **CameraTest.mod**, and **ObtainPose.kfb** files, and then click on *Open*.

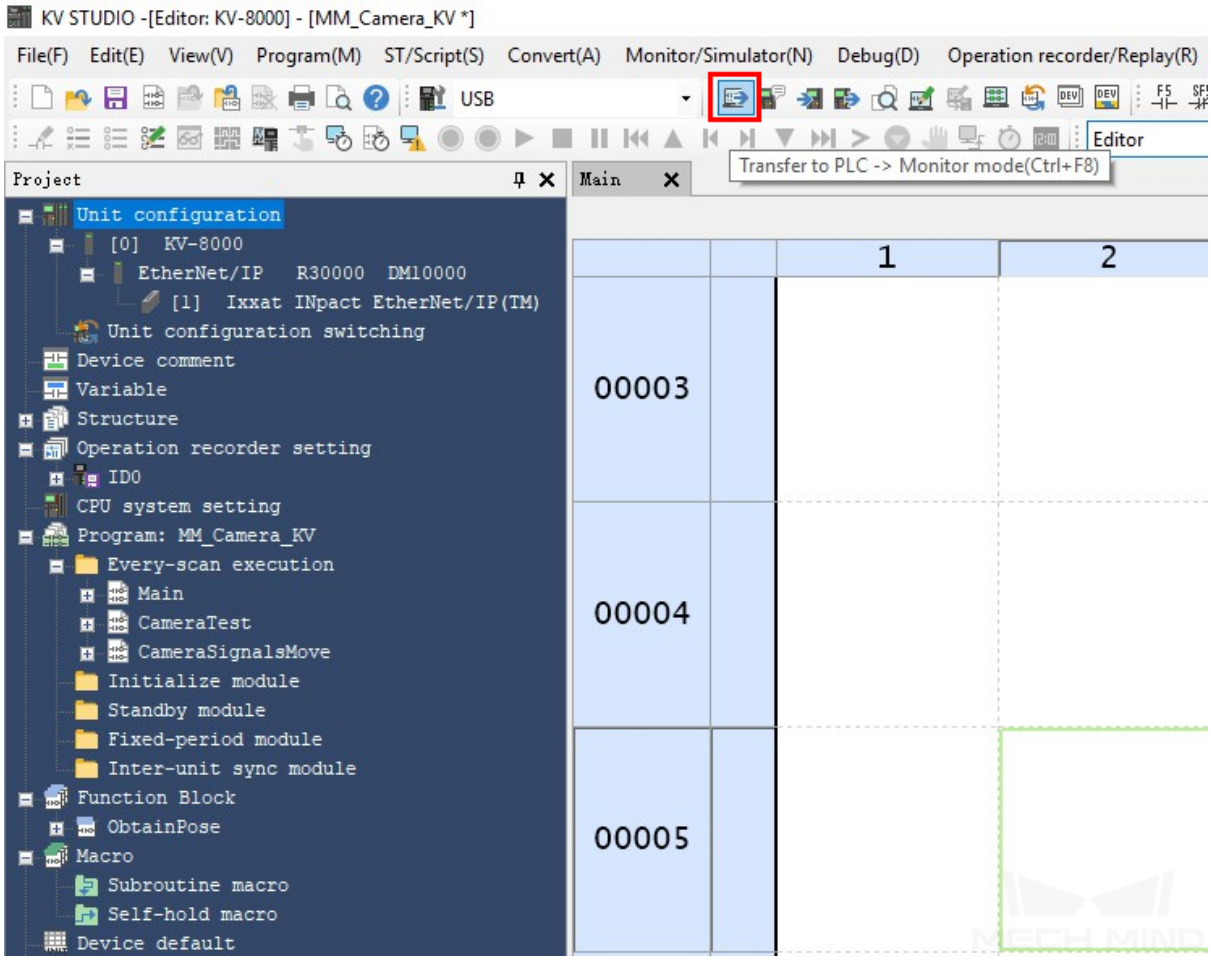


3. Click on *OK* in the **Import completed** window.

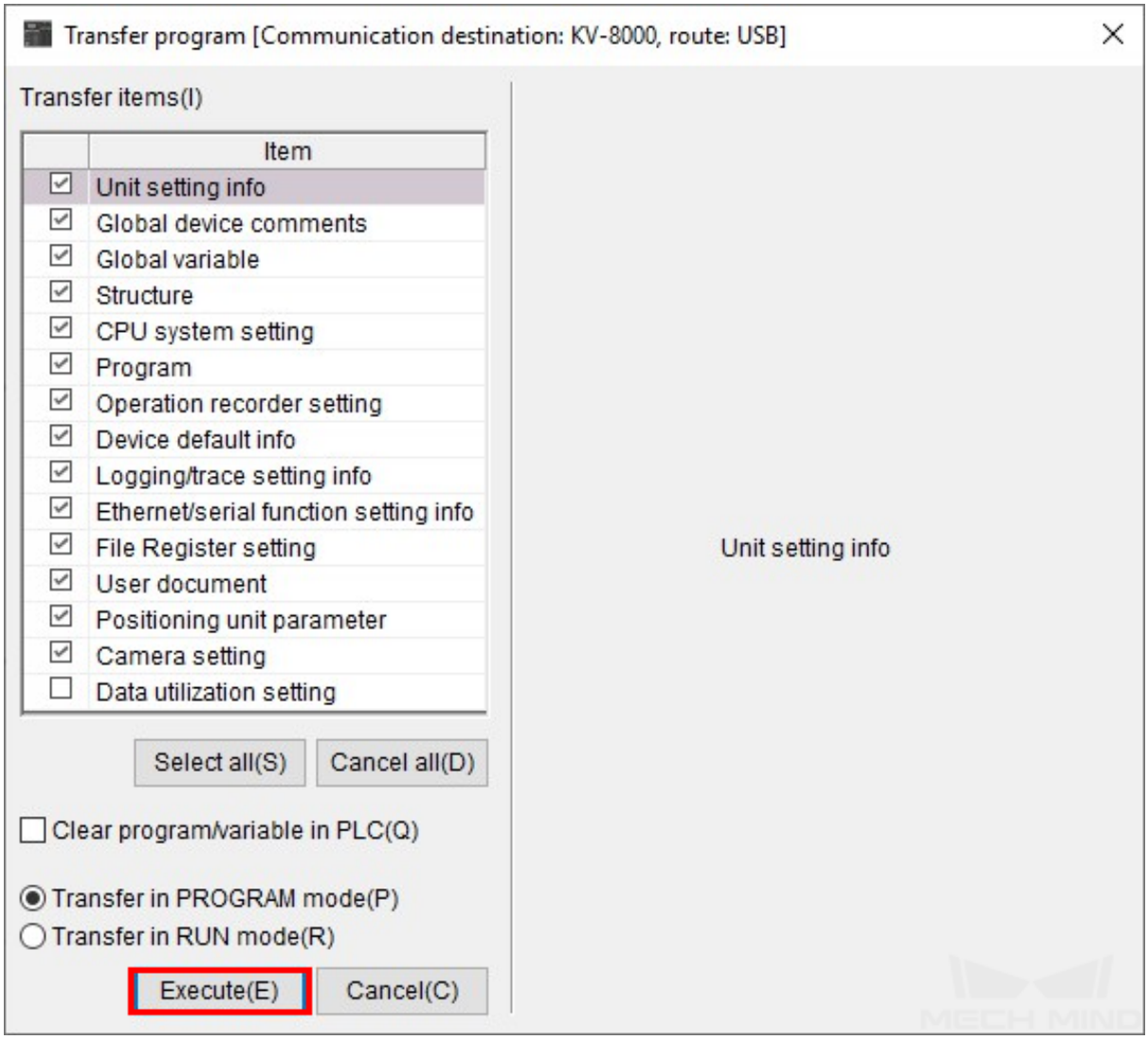


Download PLC Program

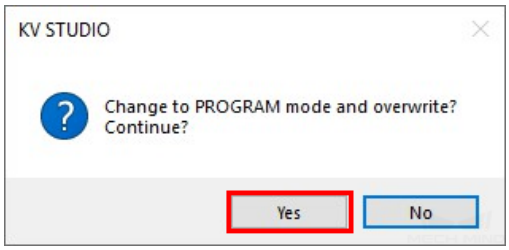
1. Go back to the main interface of KV STUDIO, click on  button.



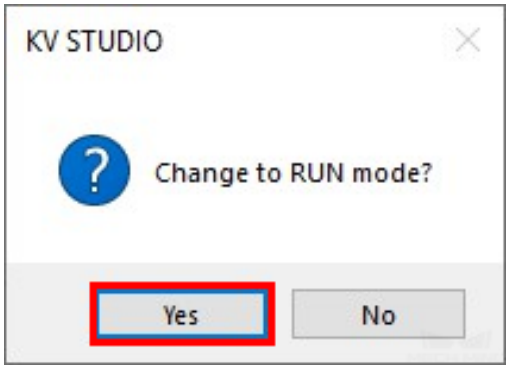
2. Keep the default settings, and click on *Execute* in the **Transfer program** window.



3. Select *Yes* in the pop-up window showing the message **"Change to PROGRAM mode and overwrite?"** .



4. Select *Yes* in the pop-up window showing the message **"Change to RUN mode?"** .

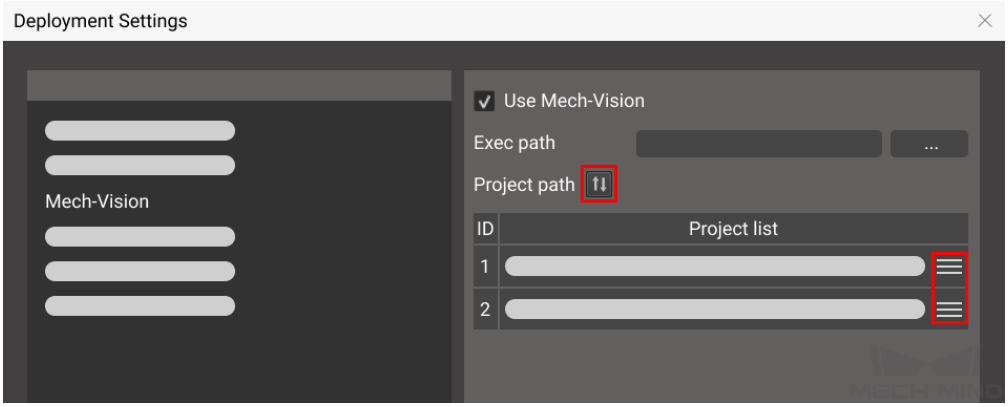


2.6.6 Test with Mech-Vision/Mech-Viz Project

This section introduces how to run the Mech-Vision/Mech-Viz project and obtain data from the project using the **ObtainPose** FB. For detailed information on the modules, please refer to standard_interface_development_profinet.

Prerequisites

- Mech-Vision project(s):
  - Executable
  - Set to autoload
  - The **Project list** in *Mech-Center* → *Deployment Settings* → *Mech-Vision* is synced by clicking on  , and the order of Mech-Vision projects have been adjusted according to actual needs.

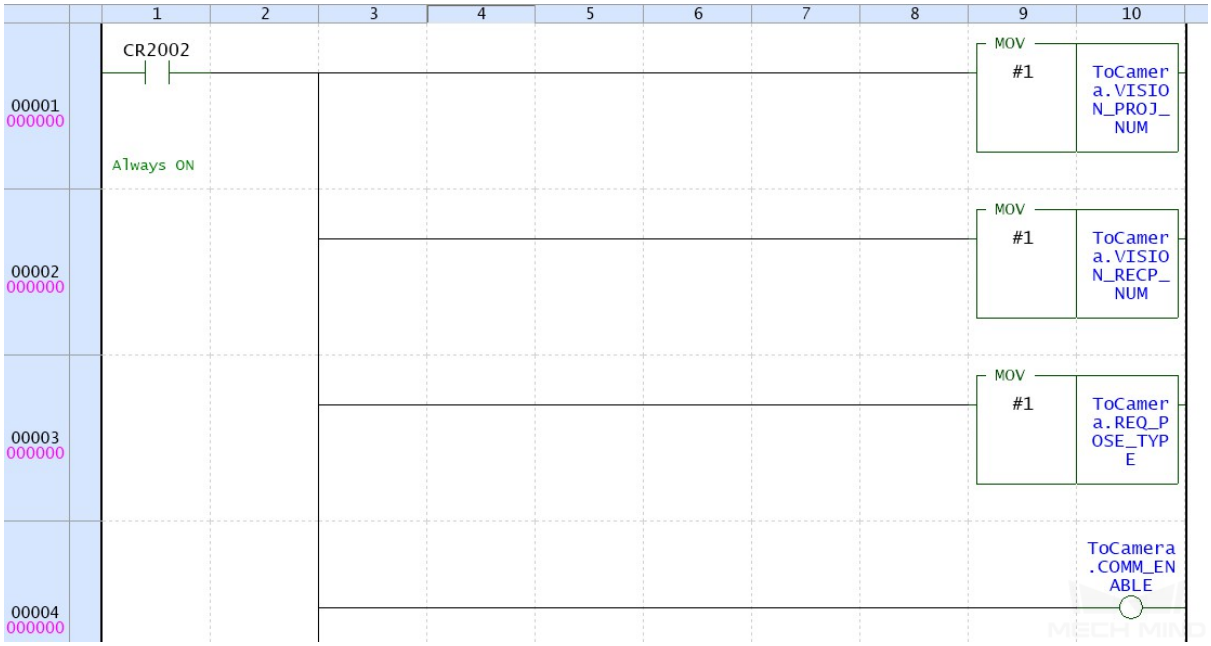


- Mech-Viz project:
  - Executable
  - Set to autoload
  - Contains a branch_by_service_message Task that has been renamed to **1**.

Run Mech-Vision Project and Obtain Vision Points

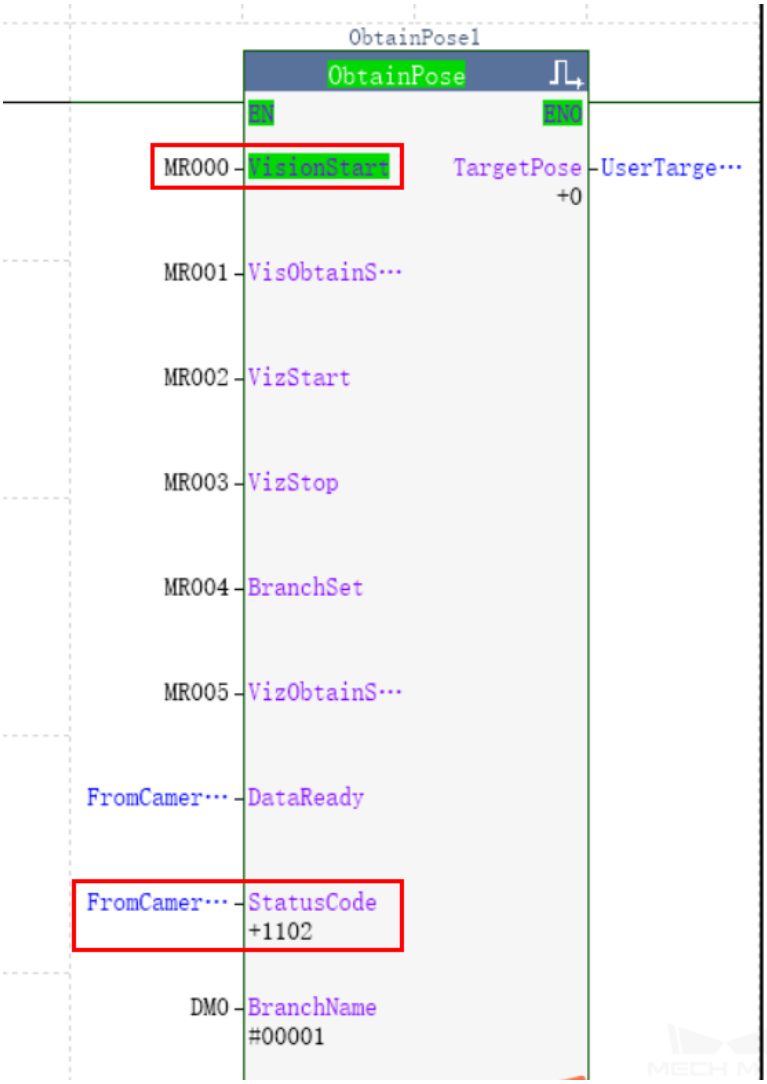
Parameter Settings

1. Set the **ToCamera.COM_ENABLE** to be **always ON**.
2. Double click on the **MOV** module, set the Mech-Vision project ID the same as the one set in **Deployment Settings** in Mech-Center. For example, if the monitor value is changed to **1**, then Mech-Vision project No. 1 in the **Project list** of Mech-Center will be started.
3. Set the number of vision points to be sent by Mech-Vision. The default value of **REQ_POSE_NUM** is 0, which means the Mech-Vision project will send all the vision points.



Start Mech-Vision Project

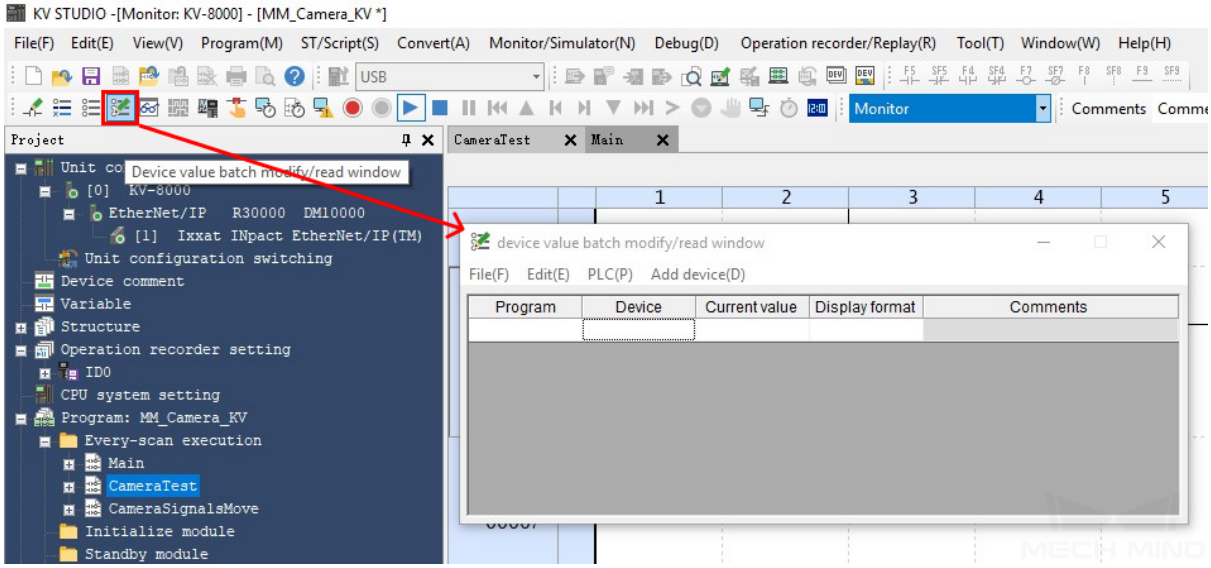
1. Go back to main interface of KV STUDIO, go to *Project*→ *Program* and double click on **CameraTest**. Double click on **VisionStart** in the FB **ObtainPose** to set the value to **1** and therefore start Mech-Vision project. Then double click again to reset the value to **0**.
2. Check returned status code: check the monitor value of **StatusCode**. **1102** represents that the Mech-Vision project was started successfully. For other values, please refer to standard_interface_status_codes for the corresponding error.



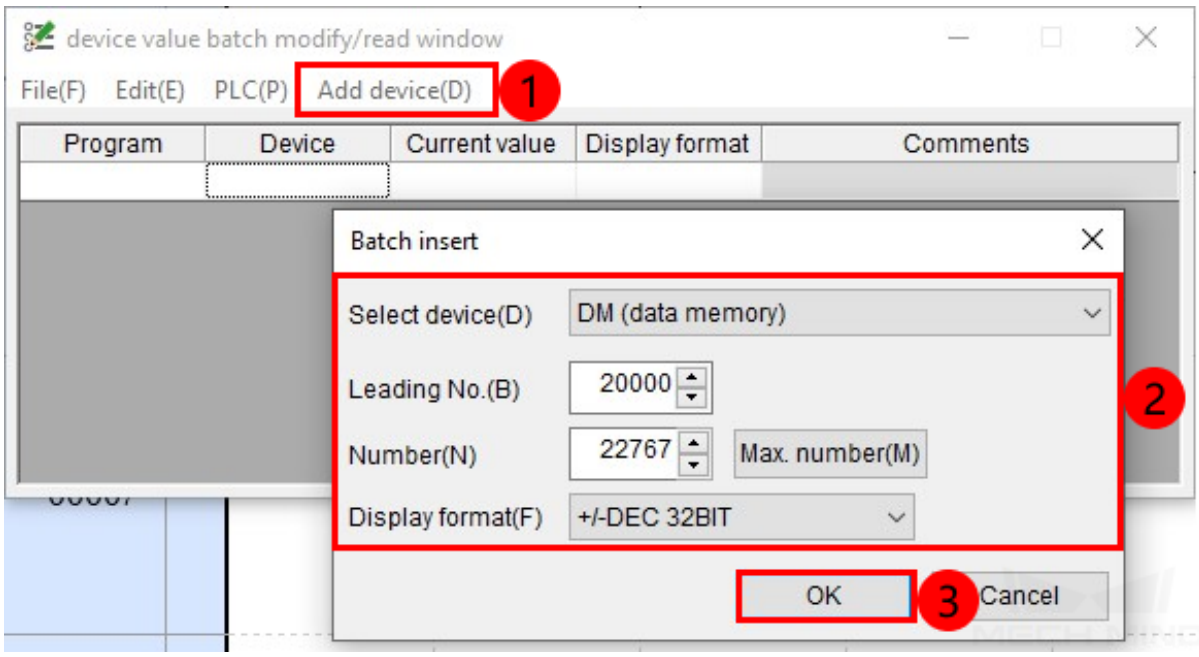
Obtain Vision Points from Mech-Vision

1. After the status code **1102** is returned, double click on **VisObtainStart** in the FB **ObtainPose** to set the value to **1** and therefore obtain vision points from Mech-Vision. Then double click again to reset the value to **0**. The result is shown as below. The value of SendPoseNum is 2, which means 2 vision points are obtained from Mech-Vision.

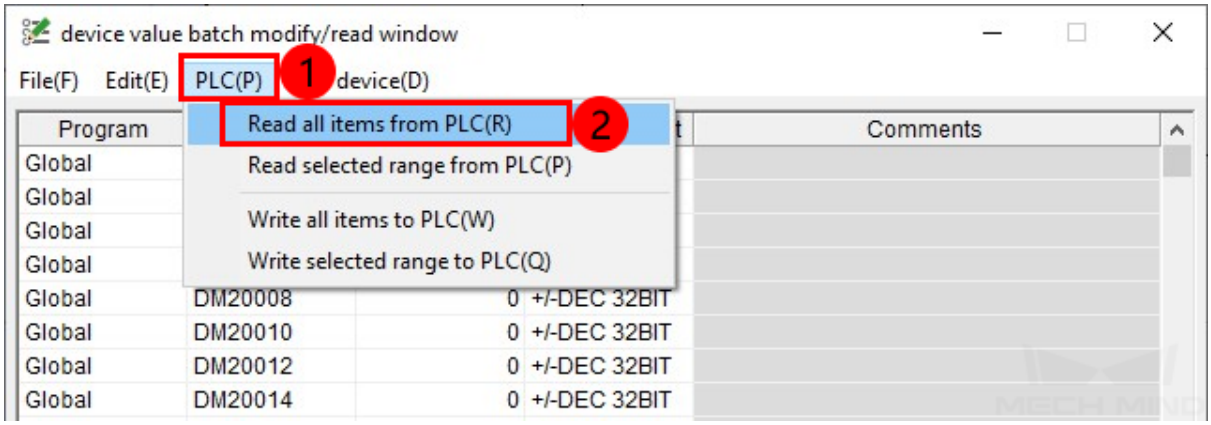
2. In the main interface of KV STUDIO, click on  button to open the **device value batch modify/read window**.



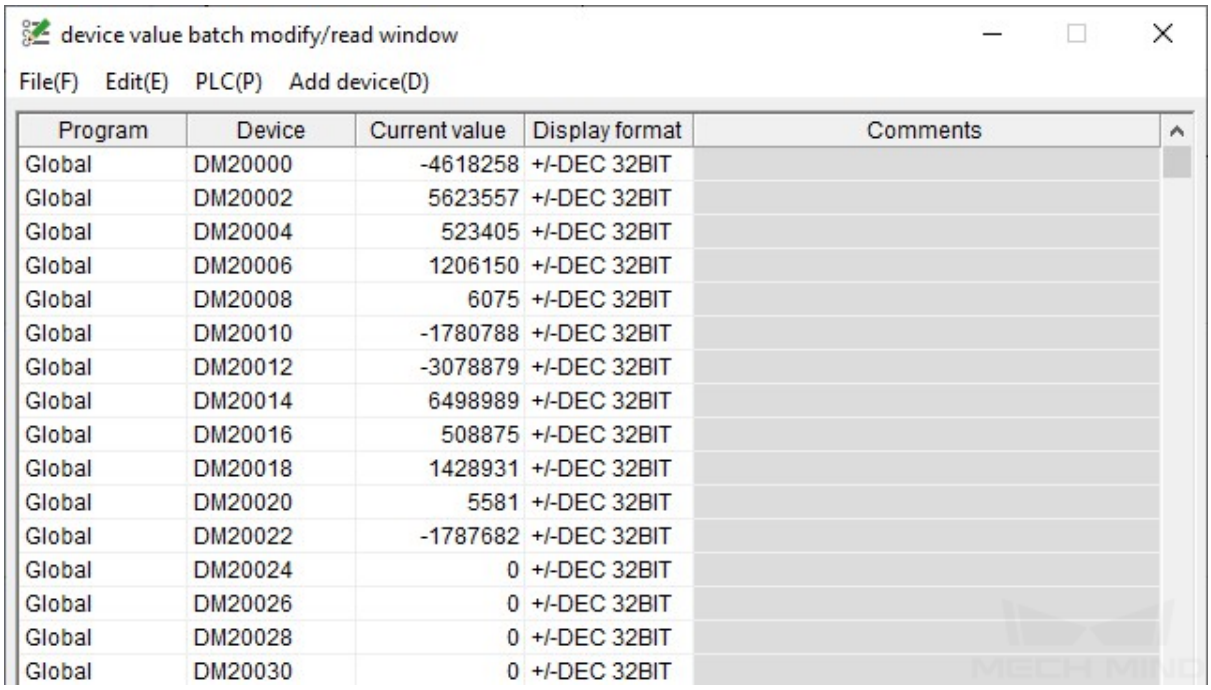
3. Select *Add device→ Batch insert* and configure the device, leading No., Number, and Display format. Click on *OK* after configurations.



4. Select *PLC→ Read all items from PLC.*



Hint: This example received 2 poses. Divide the transferred values by 10000 to obtain the actual pose data.



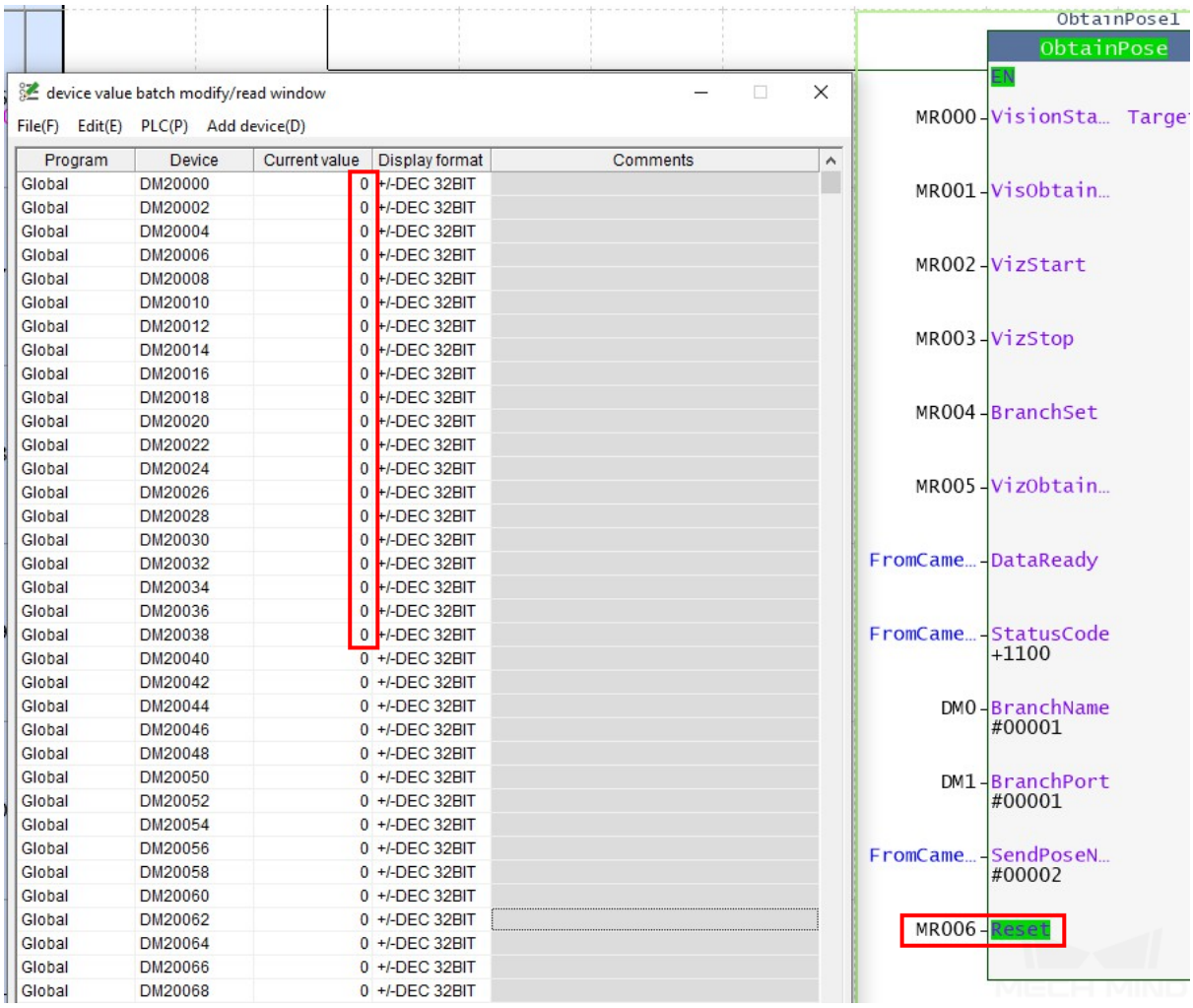
Run Mech-Viz Project and Obtain Planned Path

Parameter Settings

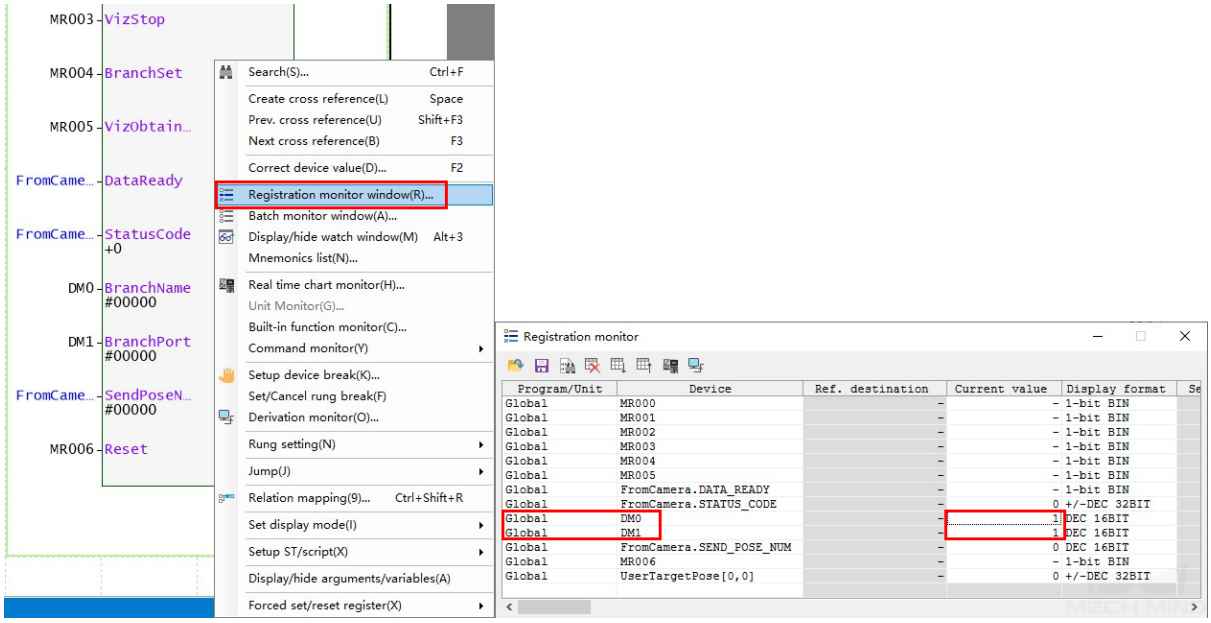
1. Double click on **Reset** in the **ObtainPose** FB to set the value to **1**. Then double click again to reset the value to **0**.

Hint: Open the **device value batch modify/read window** to check if the previously obtained

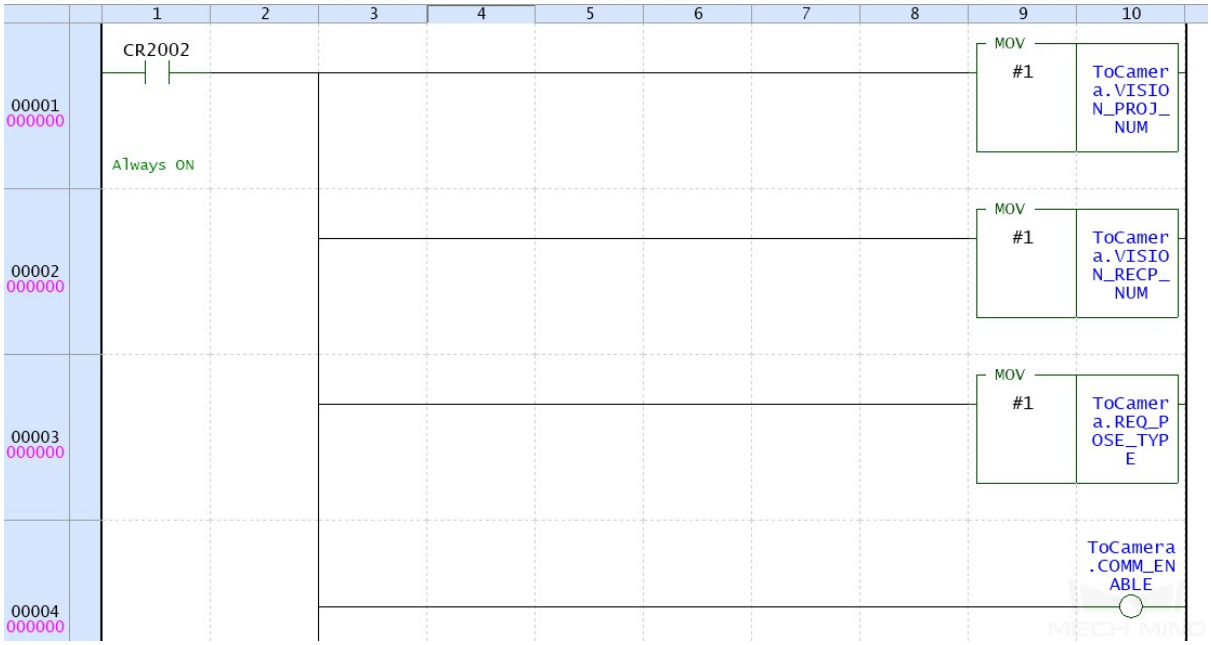
vision data has been cleared successfully.



2. Modify the value of DM0 register, set the value of **BranchName** to **1**.
3. Modify the value of DM1 register, set the value of **BranchPort** to **1** , the Mech-Viz project will proceed along out port 1 of Task 1.

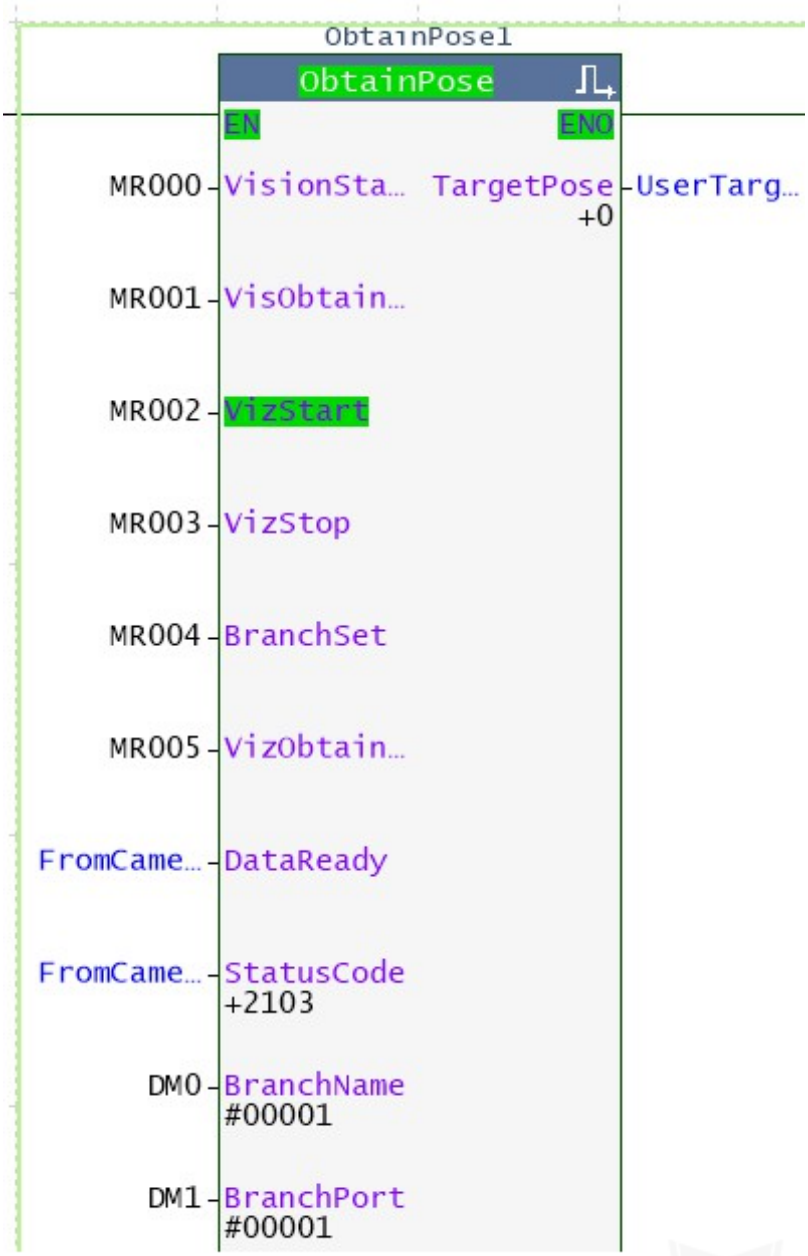


4. Set the value of REQ_POSE_TYPE to **1**. This asks Mech-Viz to send joint positions (instead of TCP data).



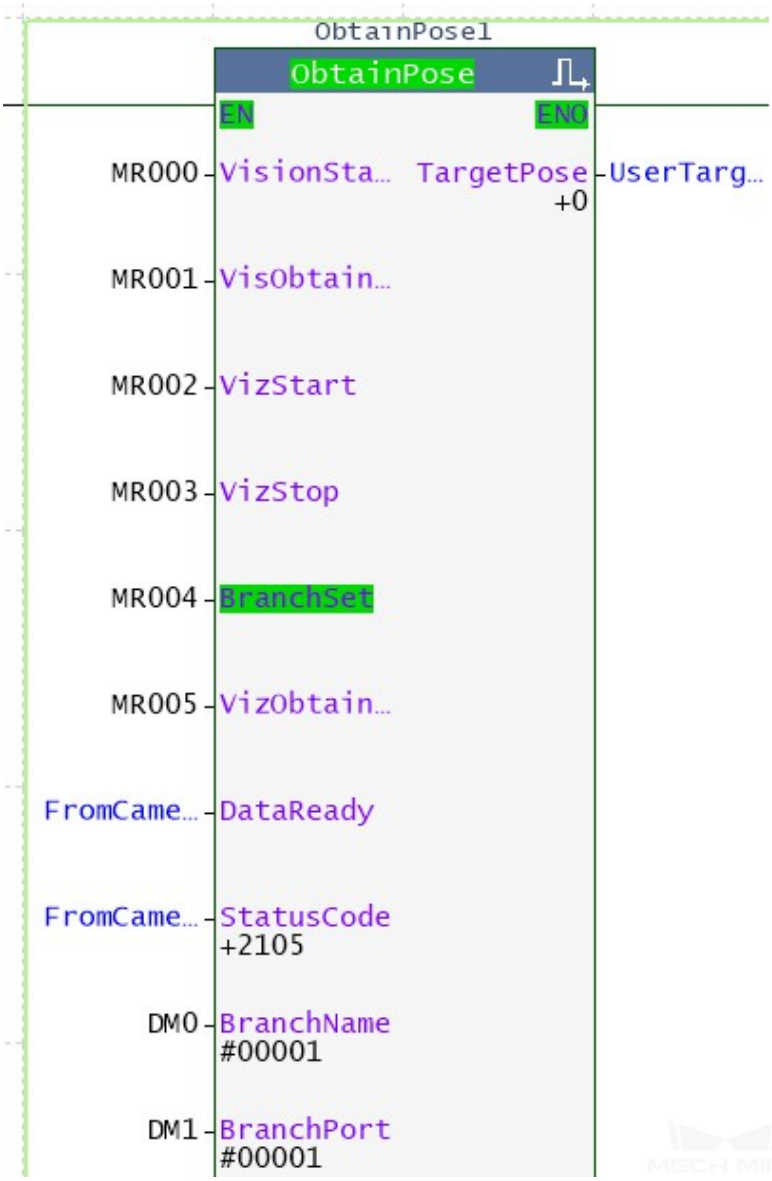
Start Mech-Viz Project

1. Double click on **VizStart** in the **ObtainPose** FB to set the value to **1** and therefore start Mech-Viz project. Then double click again to reset it to **0**.
2. If the value returned by the variable **StatusCode** is **2103**, it represents that the Mech-Viz project was started successfully. For other values, please refer to standard_interface_status_codes for the corresponding error.



Select Branch in the Mech-Viz Project

1. Double click on **BranchSet** in the **ObtainPose** FB to set the value to **1** and therefore select branch in the Mech-Viz project. Then double click again to reset it to **0**.
2. If the value returned by the variable **StatusCode** is **2105**, it represents that the branch was selected successfully. For other values, please refer to standard_interface_status_codes for the corresponding error.



Obtain Planned Path

1. Double click on **VizObtainStart** in the **ObtainPose** FB to set the value to **1** and therefore obtain planned path from Mech-Viz project. Then double click again to reset it to **0**.
2. If the value returned by the variable **StatusCode** is **2100**, it represents that planned path was obtained successfully. For other values, please refer to standard_interface_status_codes for the corresponding error. The value of **SendPoseNum** shows how many target points were received, and the target points are stored in **TargetPose**.

3. Go back to **device value batch modify/read window**, the 10 poses are shown as below. Please divide the transferred values by 10000 to obtain the actual pose data.

device value batch modify/read window

File(F) Edit(E) PLC(P) Add device(D)

Program	Device	Current value	Display format	Comments
Global	DM20000	740000	+/-DEC 32BIT	
Global	DM20002	574700	+/-DEC 32BIT	
Global	DM20004	-521200	+/-DEC 32BIT	
Global	DM20006	80000	+/-DEC 32BIT	
Global	DM20008	246500	+/-DEC 32BIT	
Global	DM20010	900000	+/-DEC 32BIT	
Global	DM20012	1292518	+/-DEC 32BIT	
Global	DM20014	825172	+/-DEC 32BIT	
Global	DM20016	-259905	+/-DEC 32BIT	
Global	DM20018	35766	+/-DEC 32BIT	
Global	DM20020	338369	+/-DEC 32BIT	
Global	DM20022	1856696	+/-DEC 32BIT	
Global	DM20024	1291698	+/-DEC 32BIT	
Global	DM20026	791000	+/-DEC 32BIT	
Global	DM20028	-232961	+/-DEC 32BIT	
Global	DM20030	35098	+/-DEC 32BIT	
Global	DM20032	345613	+/-DEC 32BIT	
Global	DM20034	1856680	+/-DEC 32BIT	
Global	DM20036	740000	+/-DEC 32BIT	
Global	DM20038	684700	+/-DEC 32BIT	
Global	DM20040	-31200	+/-DEC 32BIT	
Global	DM20042	0	+/-DEC 32BIT	
Global	DM20044	246500	+/-DEC 32BIT	
Global	DM20046	900000	+/-DEC 32BIT	
Global	DM20048	740000	+/-DEC 32BIT	
Global	DM20050	574700	+/-DEC 32BIT	
Global	DM20052	-31200	+/-DEC 32BIT	
Global	DM20054	0	+/-DEC 32BIT	
Global	DM20056	246500	+/-DEC 32BIT	
Global	DM20058	900000	+/-DEC 32BIT	
Global	DM20060	600000	+/-DEC 32BIT	
Global	DM20062	574700	+/-DEC 32BIT	
Global	DM20064	-31200	+/-DEC 32BIT	
Global	DM20066	0	+/-DEC 32BIT	
Global	DM20068	246500	+/-DEC 32BIT	
Global	DM20070	900000	+/-DEC 32BIT	

Global	DM20072	500000	+/-DEC 32BIT
Global	DM20074	574700	+/-DEC 32BIT
Global	DM20076	-31200	+/-DEC 32BIT
Global	DM20078	0	+/-DEC 32BIT
Global	DM20080	246500	+/-DEC 32BIT
Global	DM20082	900000	+/-DEC 32BIT
Global	DM20084	400000	+/-DEC 32BIT
Global	DM20086	574700	+/-DEC 32BIT
Global	DM20088	-31200	+/-DEC 32BIT
Global	DM20090	0	+/-DEC 32BIT
Global	DM20092	246500	+/-DEC 32BIT
Global	DM20094	900000	+/-DEC 32BIT
Global	DM20096	0	+/-DEC 32BIT
Global	DM20098	718643	+/-DEC 32BIT
Global	DM20100	-52061	+/-DEC 32BIT
Global	DM20102	0	+/-DEC 32BIT
Global	DM20104	233418	+/-DEC 32BIT
Global	DM20106	900000	+/-DEC 32BIT
Global	DM20108	0	+/-DEC 32BIT
Global	DM20110	684720	+/-DEC 32BIT
Global	DM20112	-31191	+/-DEC 32BIT
Global	DM20114	0	+/-DEC 32BIT
Global	DM20116	246471	+/-DEC 32BIT
Global	DM20118	900000	+/-DEC 32BIT
Global	DM20120	0	+/-DEC 32BIT
Global	DM20122	0	+/-DEC 32BIT
Global	DM20124	0	+/-DEC 32BIT
Global	DM20126	0	+/-DEC 32BIT
Global	DM20128	0	+/-DEC 32BIT
Global	DM20130	0	+/-DEC 32BIT

2.7 EtherNet/IP - OMRON PLC

This section provides information on setting up communication between an OMRON PLC and Mech-Mind Software Suite via EtherNet/IP.

2.7.1 Overview

- *Hardware and Software Requirements*
- *Configure IPC and Initiate Communication*
- *Install EDS file and Configure Communication*
- *Import Example Program and Download to PLC*
- *Test with Mech-Vision/Mech-Viz Project*

2.7.2 Hardware and Software Requirements

Hardware

- OMRON PLC:
  - CJ2H-CPU6[]-EIP series
  - CJ2M-CPU3[] series
  - Other models with a CJ1W-EIP21 or CS1W-EIP21 EtherNet/IP Unit
- USB Type A Male to Type B Male cable
- Power supply unit
- HMS IXXAT INpact EIP Slave PCIe interface card installed on the IPC in Mech-Mind Vision System
- Switch
- Ethernet cables

Attention: An CJ2H CPU65-EIP model and CJ1W-PA205R power supply unit are used in the example below. The UNIT NO. is set to 0, and the NODE No. is set to 15 (in HEX; 21 in decimal numeral system).

Software

- CX-Programmer V9.70
- Mech-Mind Software Suite: Mech-Center 1.5.1 or above, Mech-Vision 1.5.0 or above, and Mech-Viz 1.5.0 or above
- VCI V4 (driver software for HMS IXXAT INpact 40 interface card)
- HMS IPconfig software
- Mech-Mind EDS file:
  - File name: **005A002B003A0100.EDS**
  - Location: `Mech-Mind/Mech-Center/mech_interface/EthernetIP`
- Example programs:
  - AUTOEXEC.OBJ
  - PROGRAMS.IDX
  - SYMBOLS.SYM
  - COMMENTS.CMT

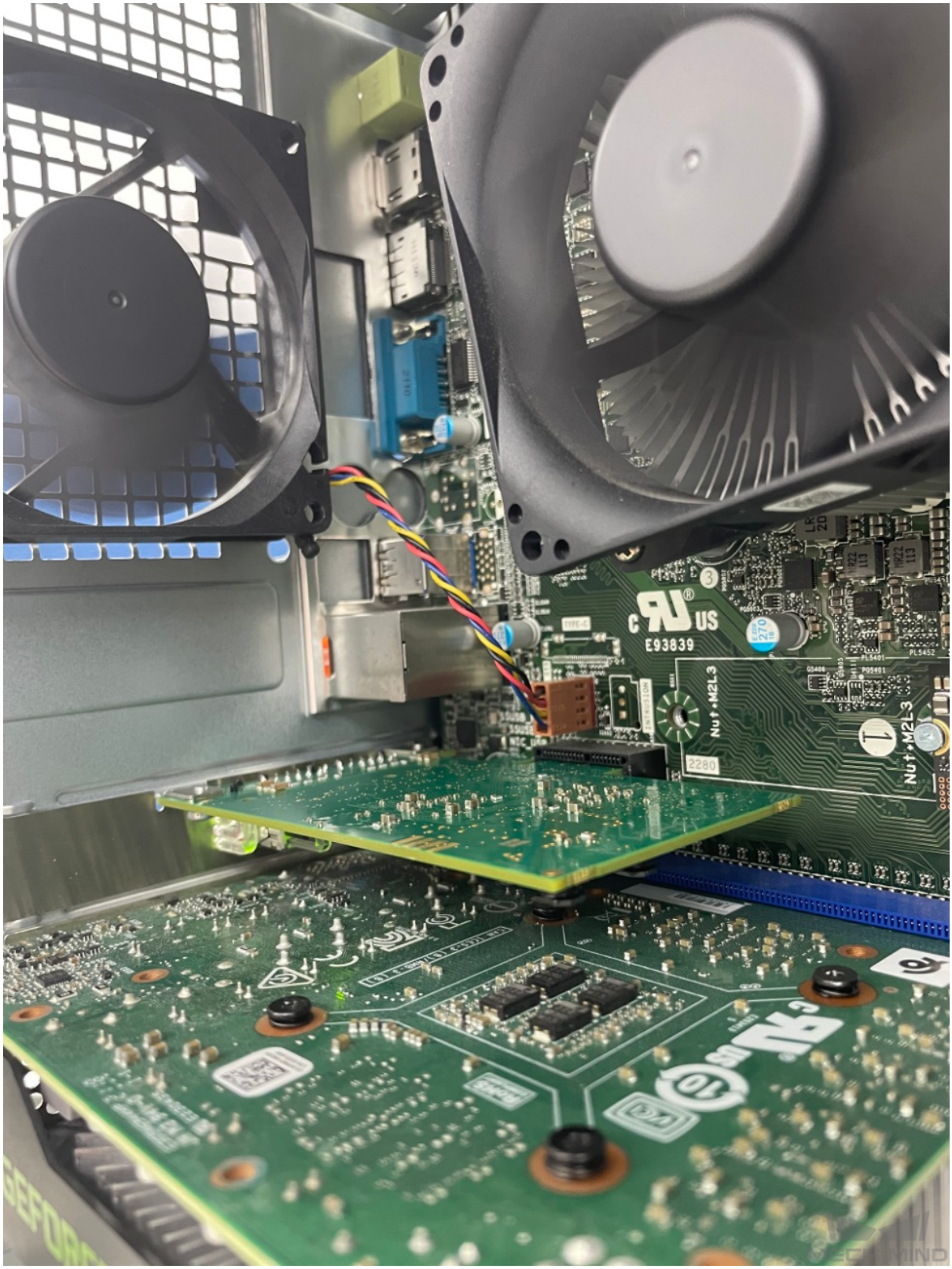
  The files are stored in `Mech-Mind/Mech-Center/mech_interface/documents/CN/欧姆龙 EtherNet IP 编程指南`. Please copy and paste all files to the computer with CX-Programmer installed.

Note: Connect the Mech-Mind Vision System IPC, computer with CX-Programmer installed, and PLC to the same router.

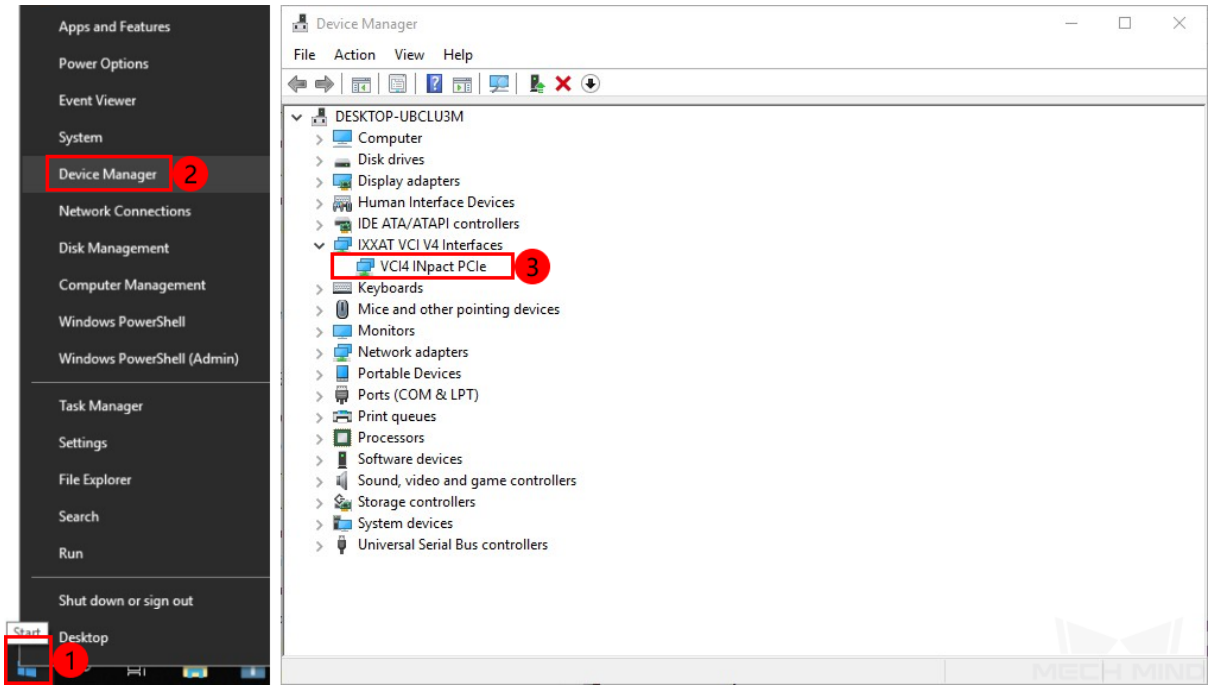
2.7.3 Configure IPC and Initiate Communication

Check PCI-e Card and Driver Software

1. Please make sure that the INpact EIP Slave PCIe interface card has been pressed into the PCI-e slot of the IPC, as shown below.

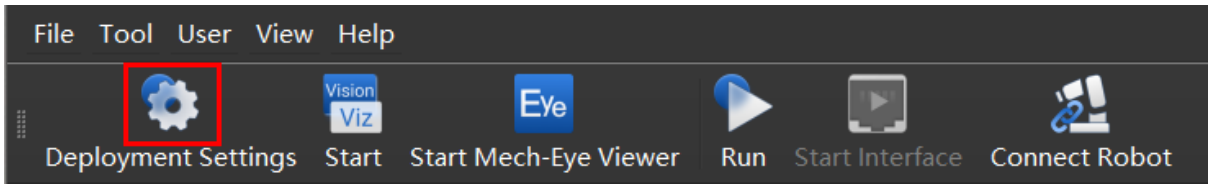

E93839
Nut•M2L-3
Nut•M2L-3

2. Start the IPC, go to *Start* → *Device Manager* and check if the driver software **VCI4 INpact PCIe** has been installed.

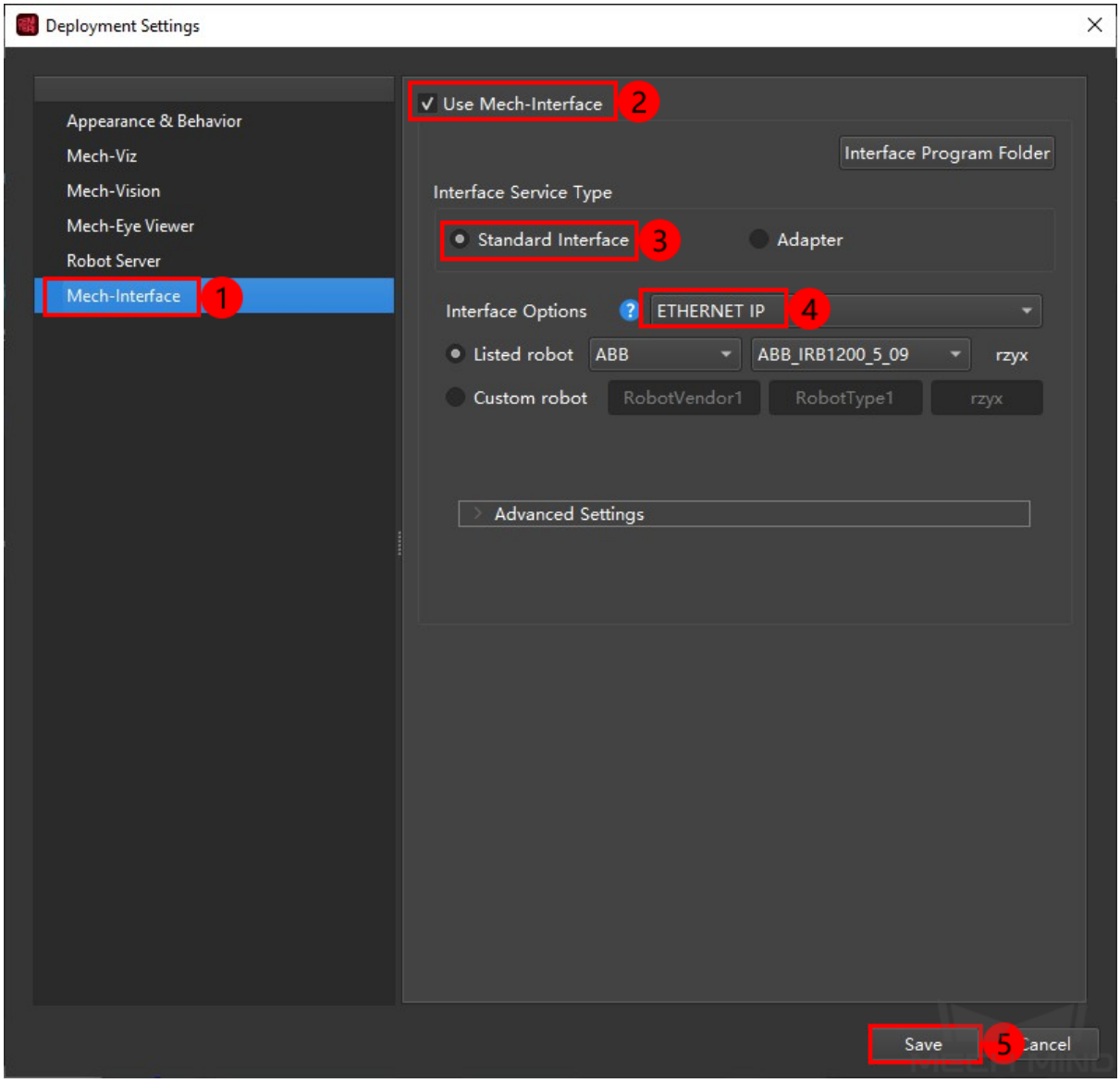


Configure Mech-Interface in Mech-Center

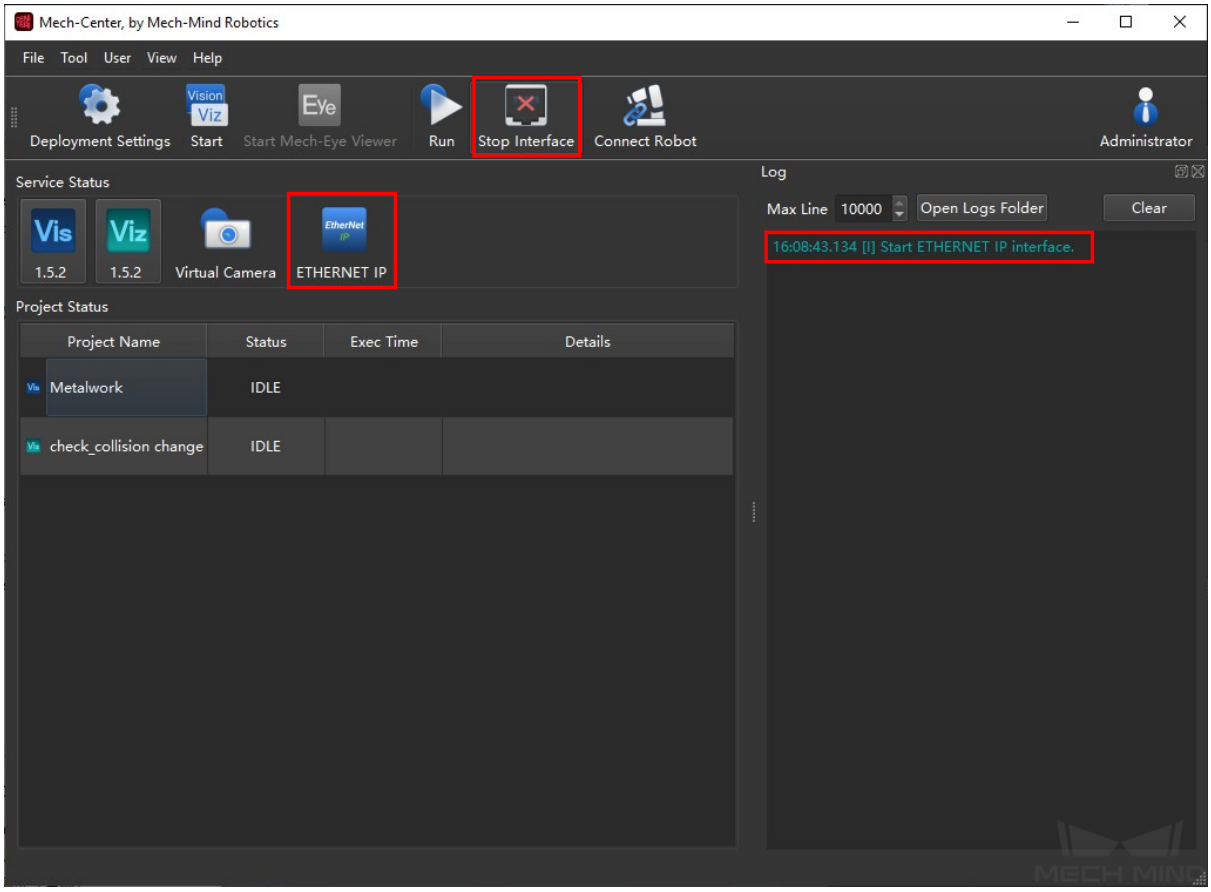
1. Open Mech-Center, and click on *Deployment Settings*.



2. Go to **Mech-Interface**, check **Use Mech-Interface** and select *Standard Interface* → *ETHERNET IP*. Click on *Save* to complete configurations.



3. Click on *Start Interface* in the Toolbar. Then an ETHERNET IP icon will display in the service status bar.

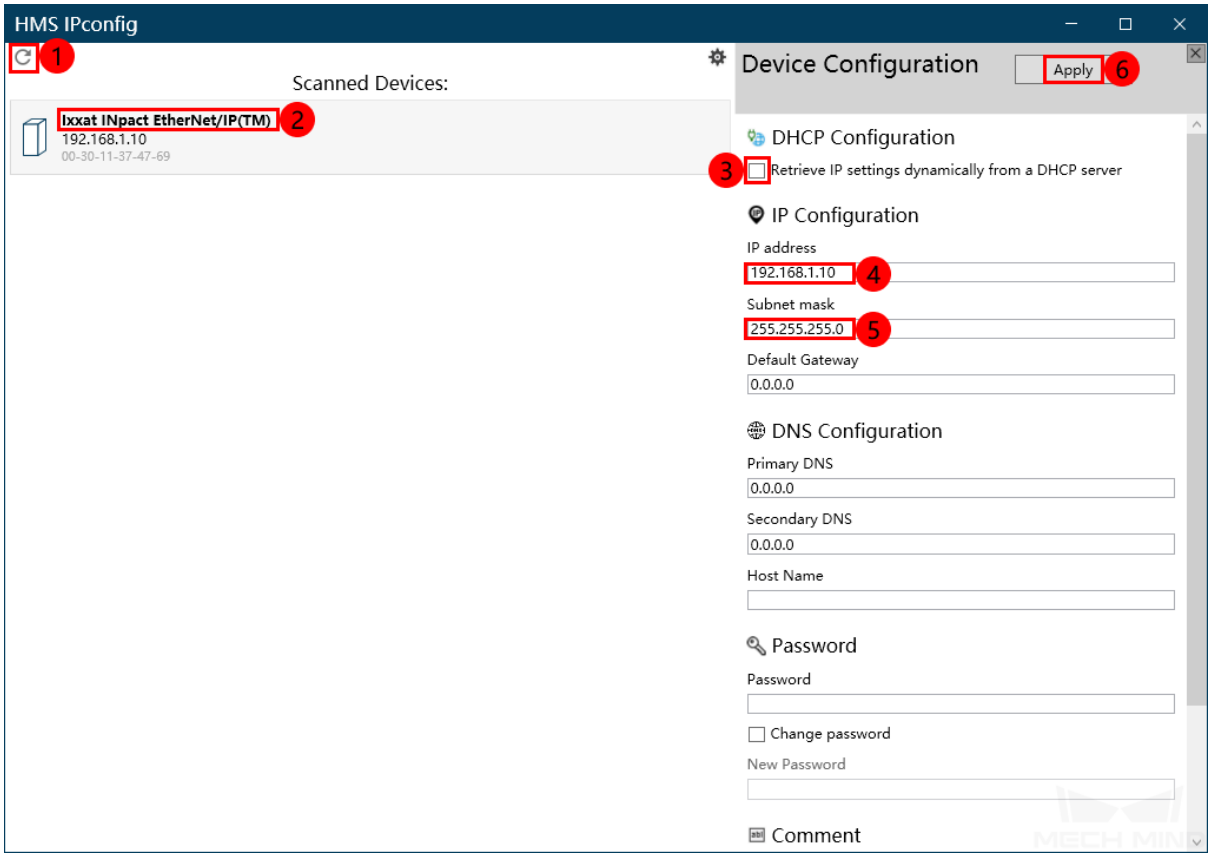


Configure IP address of the PCI-e Card

1. Download and install **HMS IPconfig software** on the IPC first. Use an Ethernet cable to connect the network ports of the IPC and the INpact EIP Slave PCIe.

   > **Attention:** After configurating the IP and initiating communication successfully, the Ethernet cable used here can be removed.

2. Open HMS IPconfig, click on  and select **Ixxat INpact EtherNet/IP(TM)**. Then uncheck **Retrieve IP settings dynamically from a DHCP server** and enter the IP address and subnet mask, as shown below. After configuration, click on *Apply*.

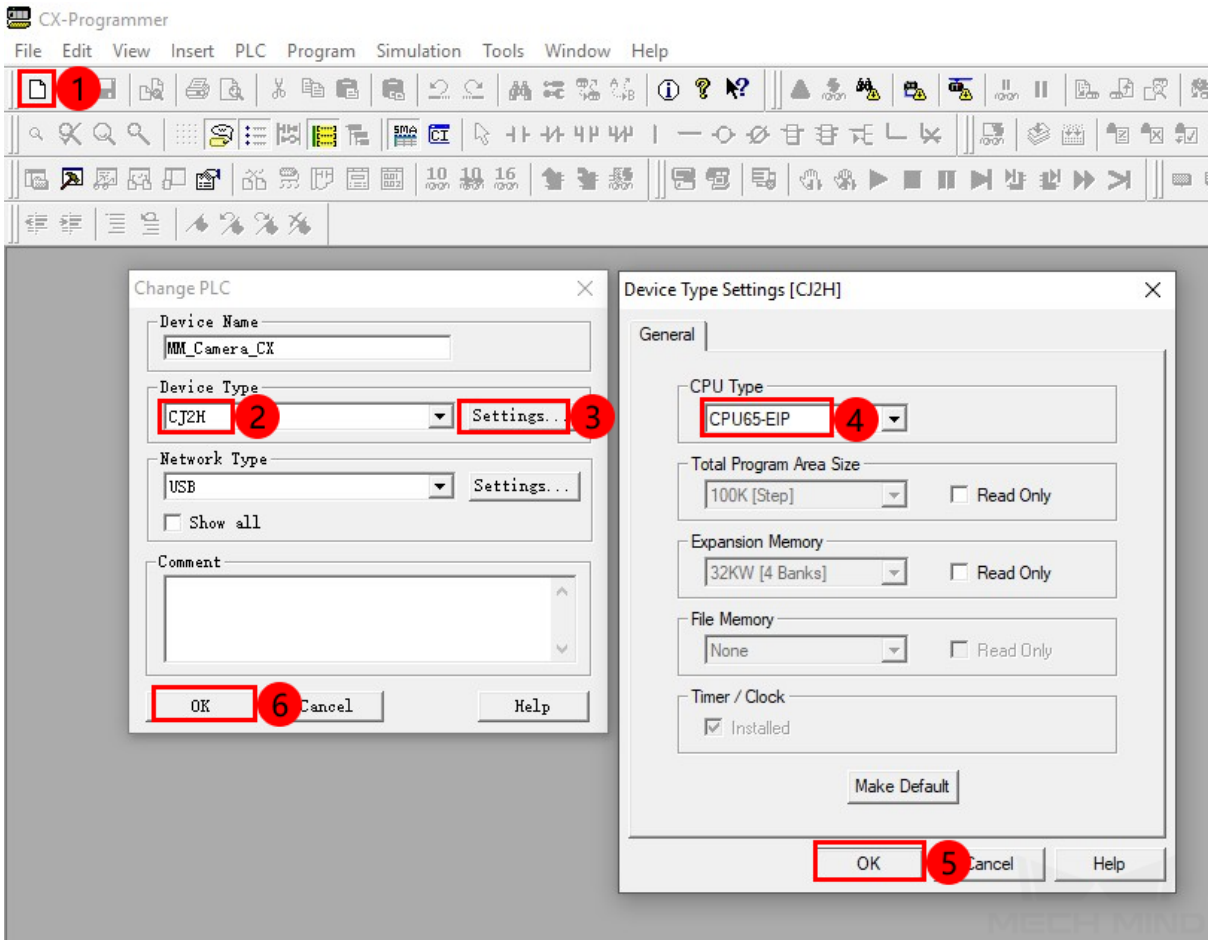


> **Attention:** The IP address should be the same as which is configured in the PLC.

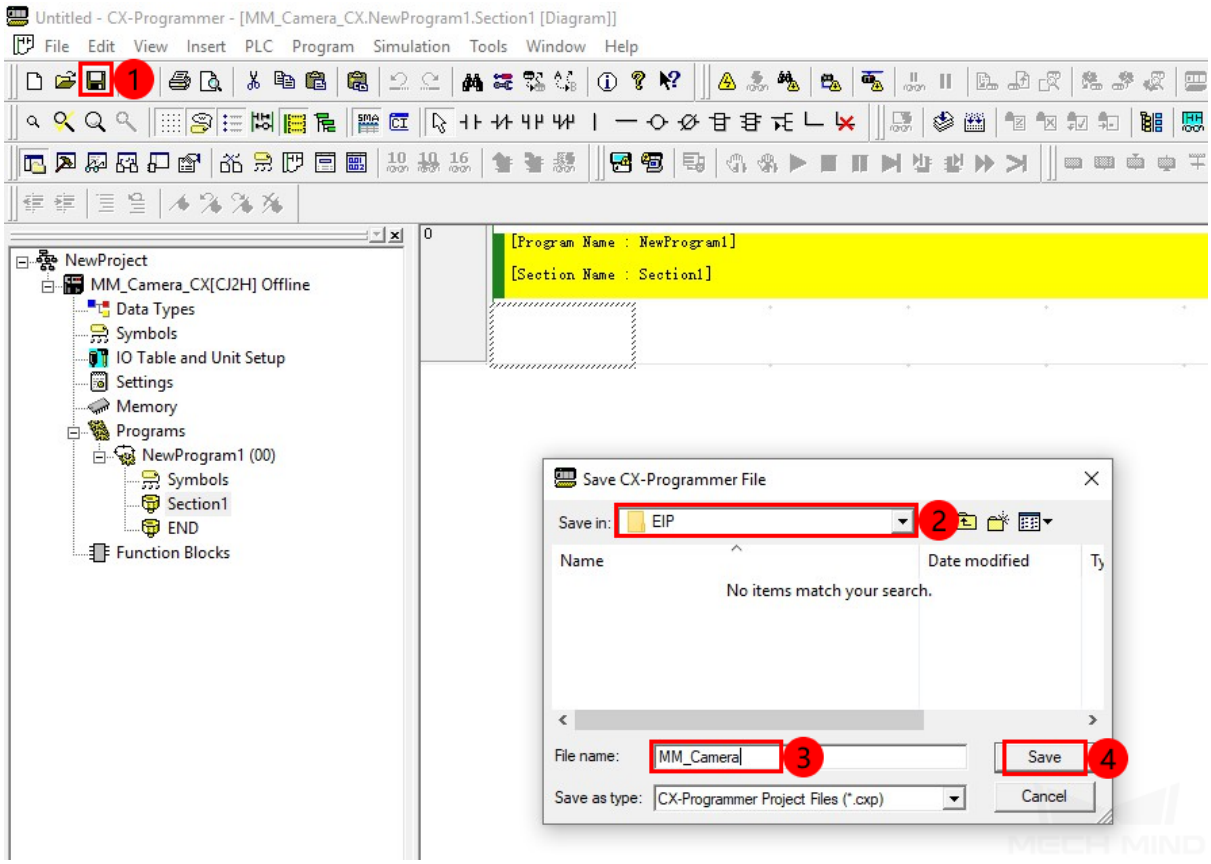
2.7.4 Install EDS file and Configure Communication

Create PLC Project

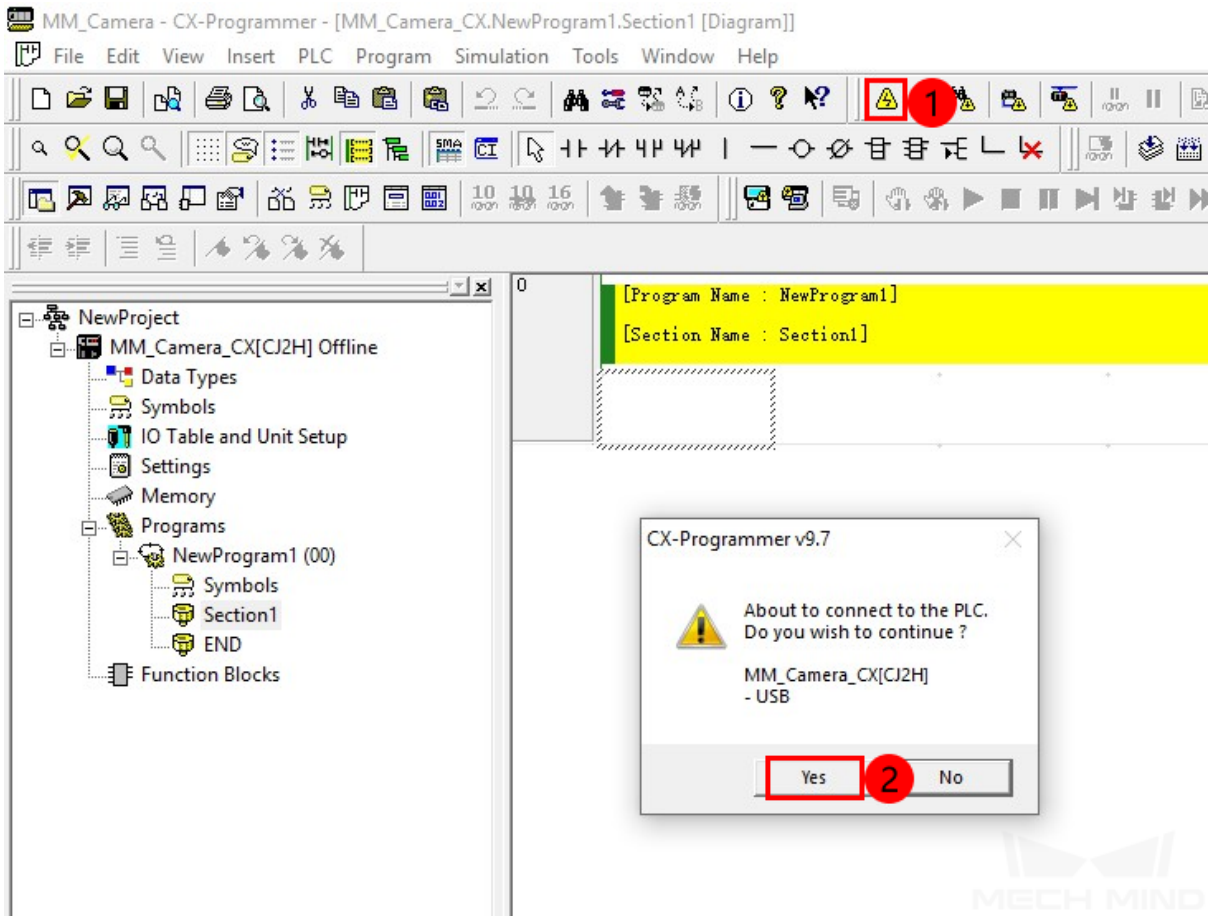
1. Open the CX-Programmer software, click on  in the menu bar and then a **Change PLC** window will pop up. Select the **Device Type** according to the model in use, and then click on *Settings*. Select the **CPU** type in the pop-up **Device Type Settings** window. Click on *OK* to save the changes.



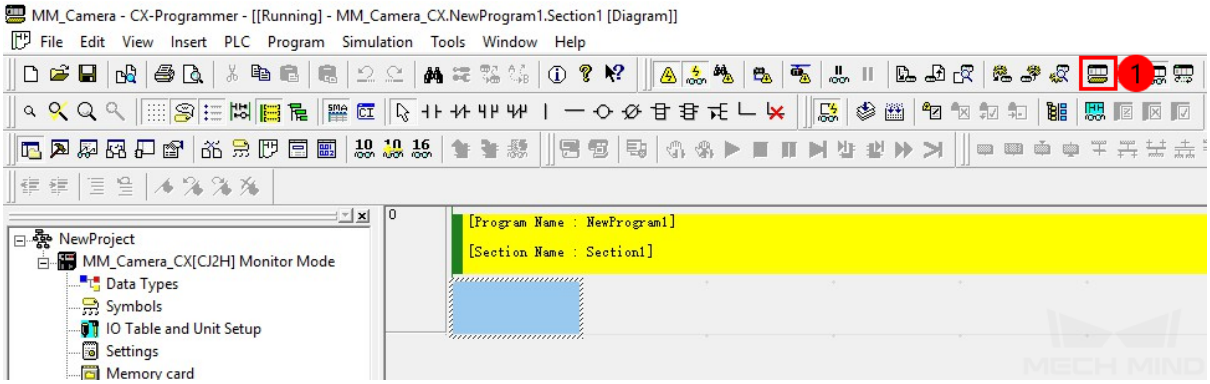
2. Click on  in the menu bar, a **Save CX-Programmer File** window will pop up. Select a folder to save the project file, name the file, and then click on *Save*.

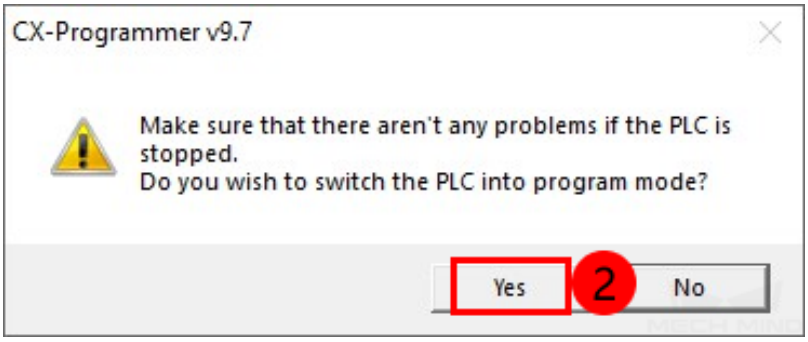


3. Click on  in the menu bar, and then select *Yes* in the pop-up window to connect to the PLC.

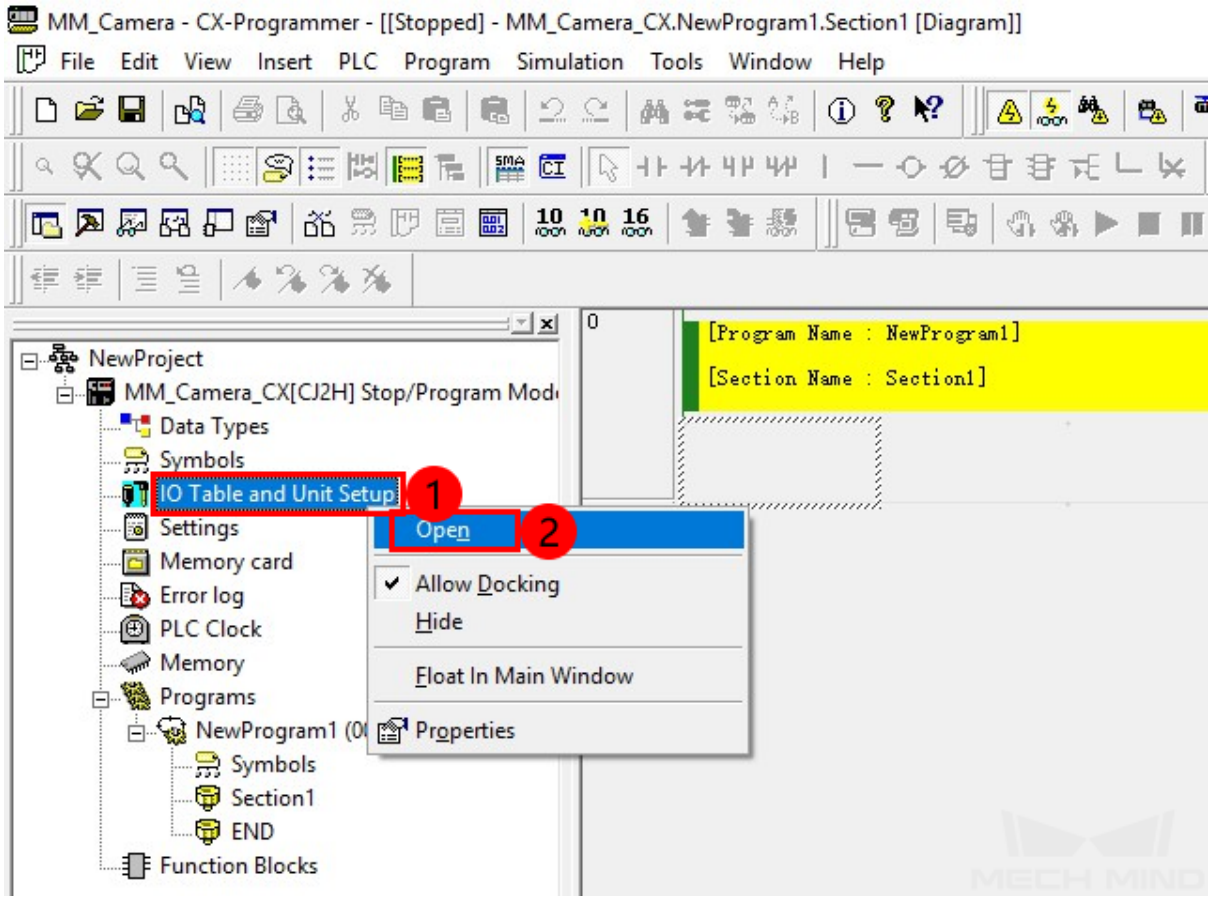


4. Click on  in the menu bar. Make sure that there aren't any problems if the PLC is stopped, and then select *Yes*.

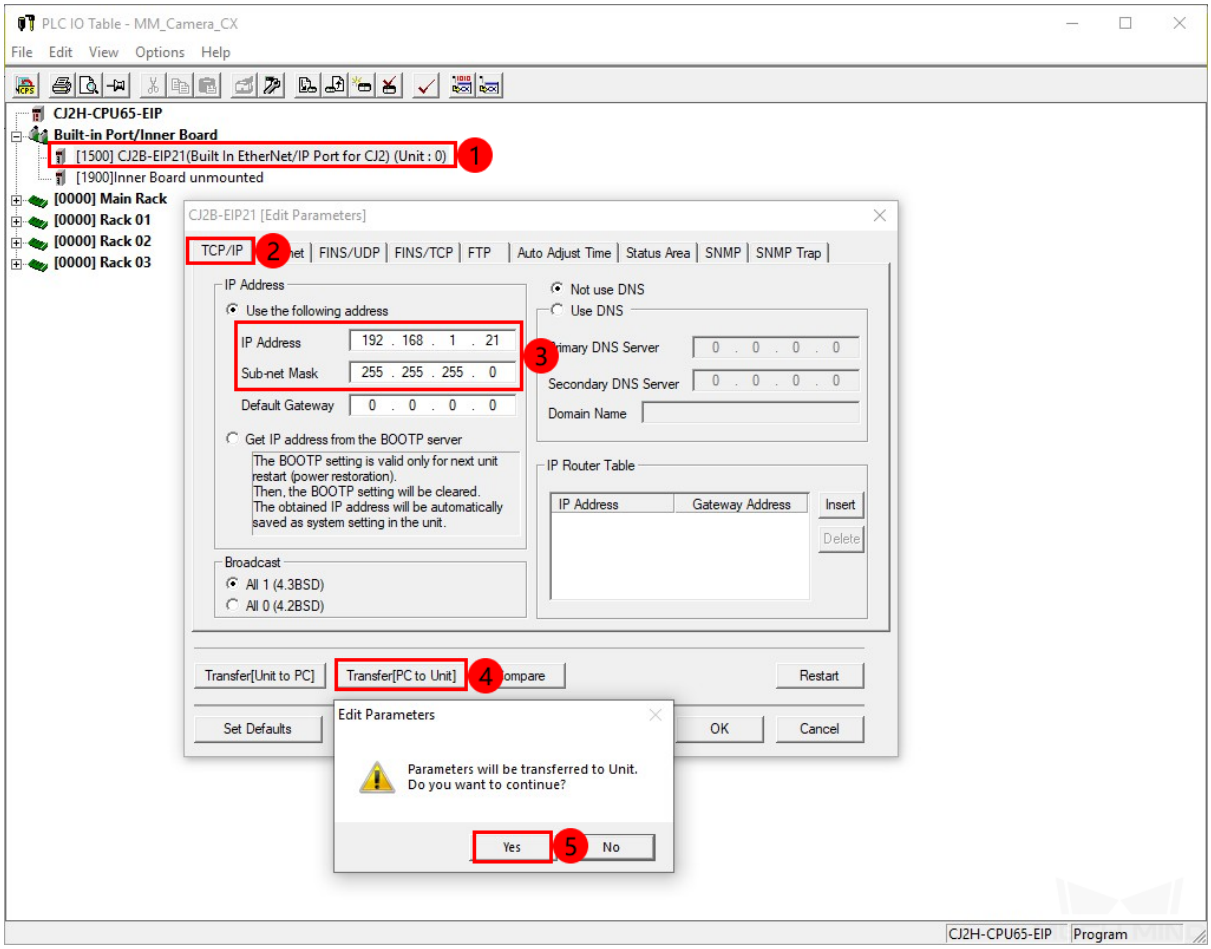




5. Select *IO Table and Unit Setup* → *Open* in the project workspace to open the **PLC IO Table**.



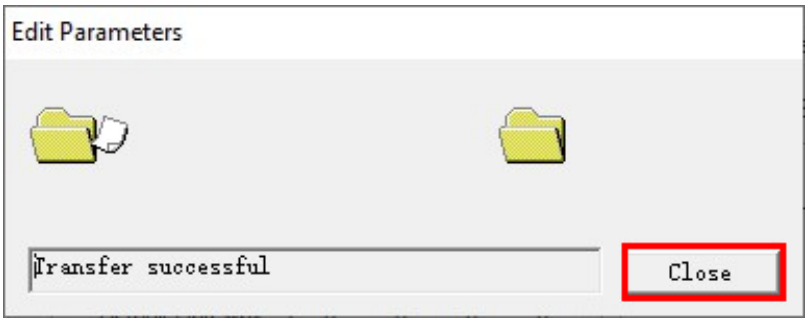
6. Go to **Built-in Port/Inner Board** and select **CJ2B-EIP21**. Select *TCP/IP* in the **Edit Parameters** window, enter the IP address and subnet mask. Then click on *Transfer[PC to Unit]*, select *Yes* in the pop-up window to transfer parameters.



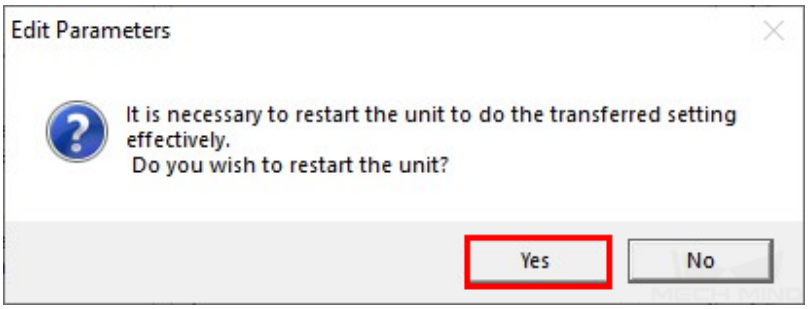
Note:

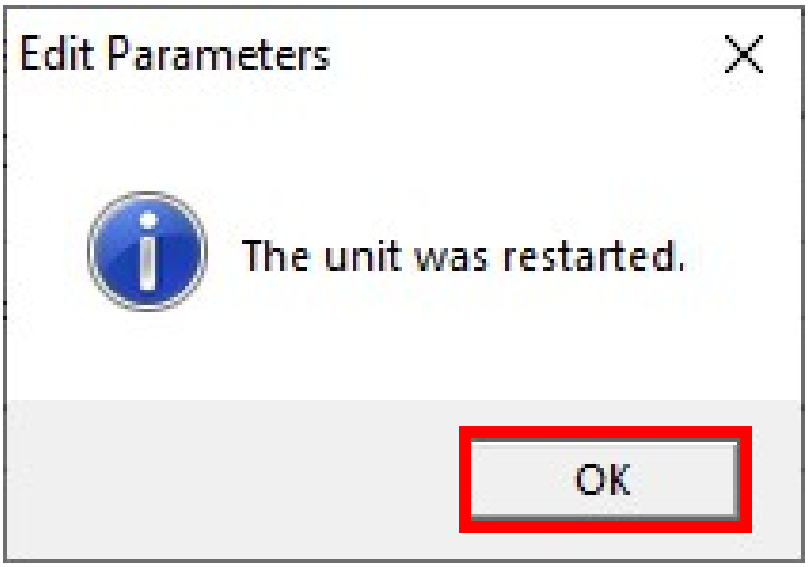
- The instructions in this step is based on a specific model of PLC. Please configure according to the actual PLC model you are using.
- The last number of the IP address should be the same as the NODE No. set on the PLC.

7. After transfering the parameters successfully, click on *Close*.



8. Select *Yes* to restart the unit, and click on *OK* in the pop-up window.

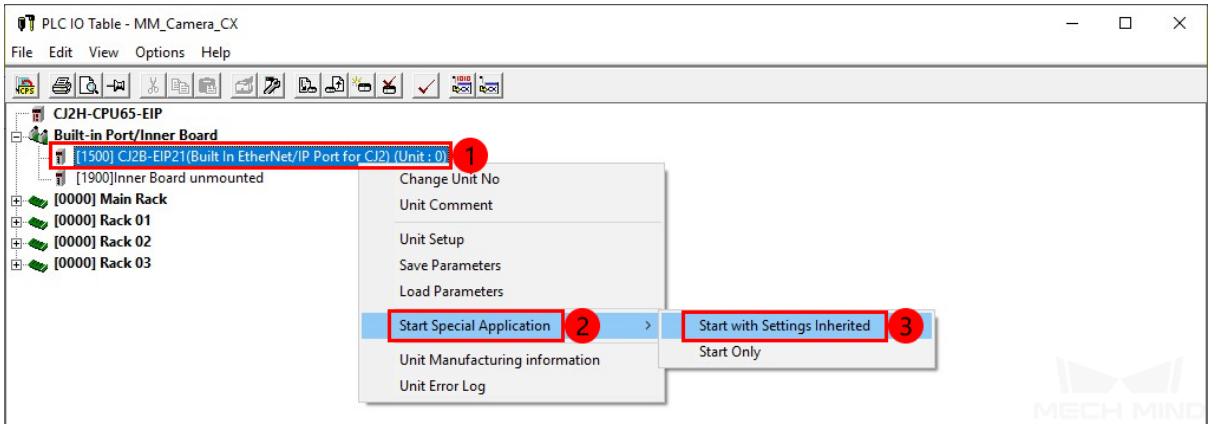




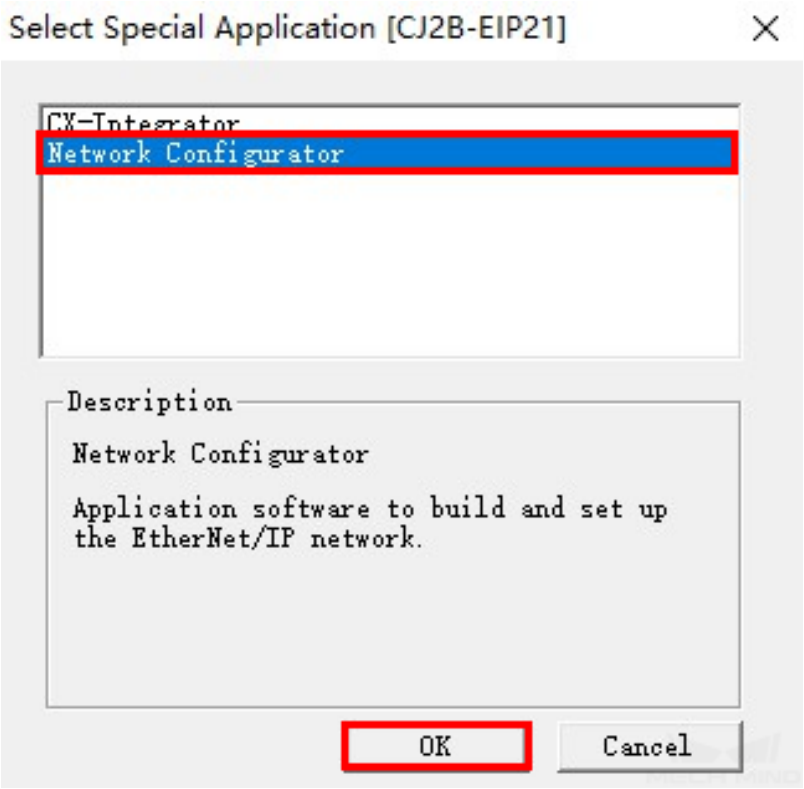
9. Click on *OK* in the **Edit Parameters** window to complete configuration.

Install EDS File and Configure Network

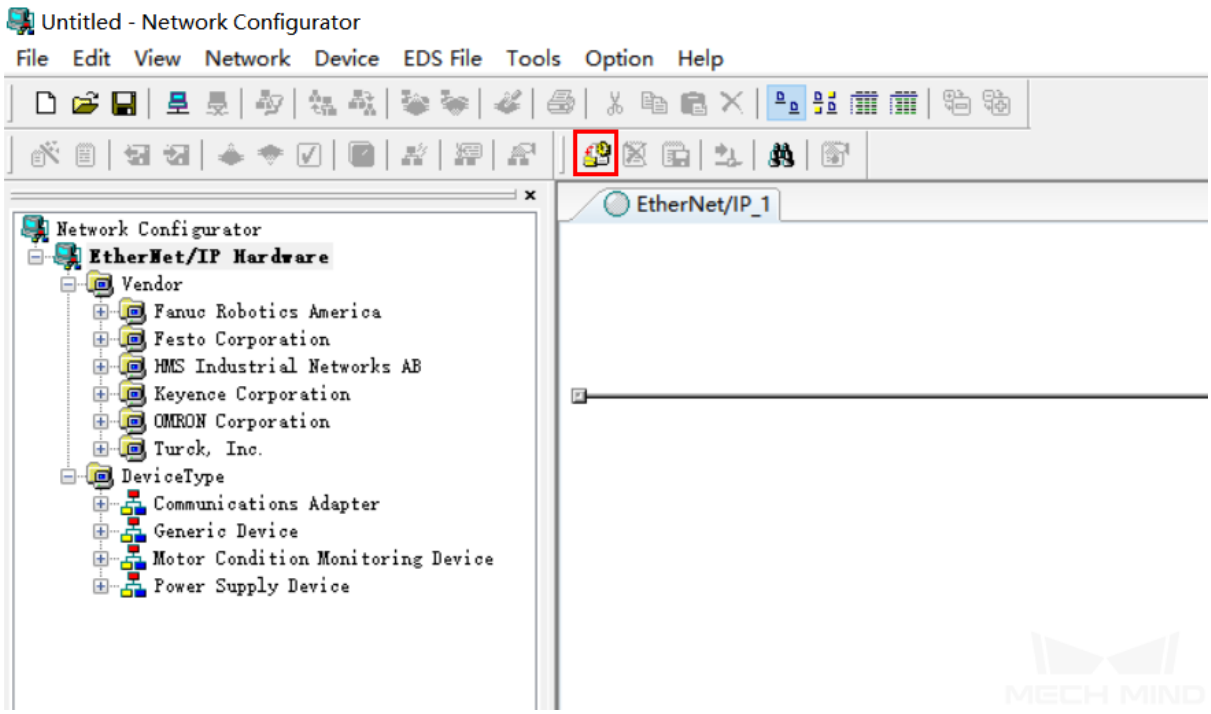
1. Right click on *CJ2B-EIP21* in the **PLC IO Table**, and then select *Start Special Application* → *Start with Settings Inherited*.



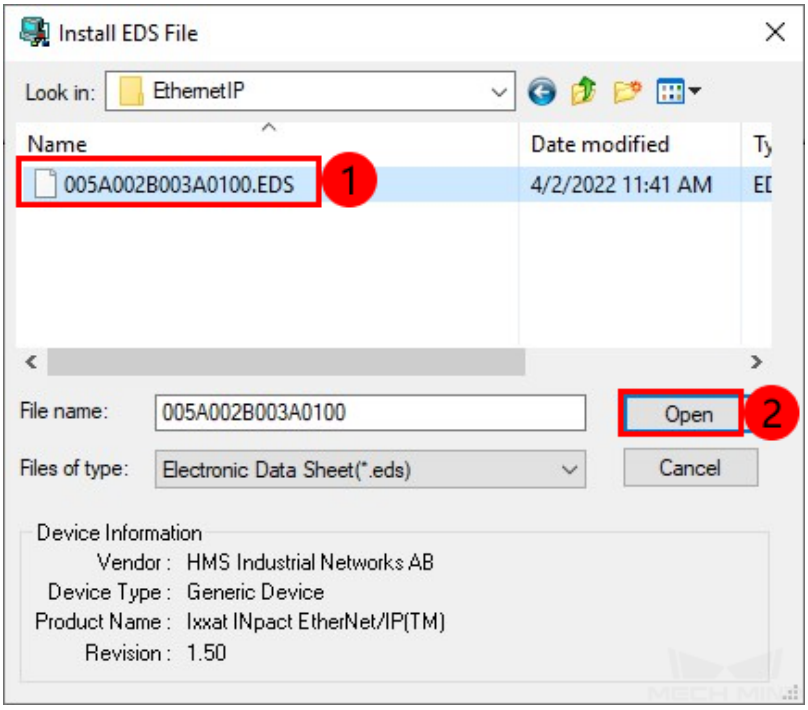
2. Select **Network Configurator** in the pop-up window, and click on *OK* to open the **Untitled - Network Configurator** window.



3. Click on  in the menu bar, and an **Install EDS File** file will pop up.

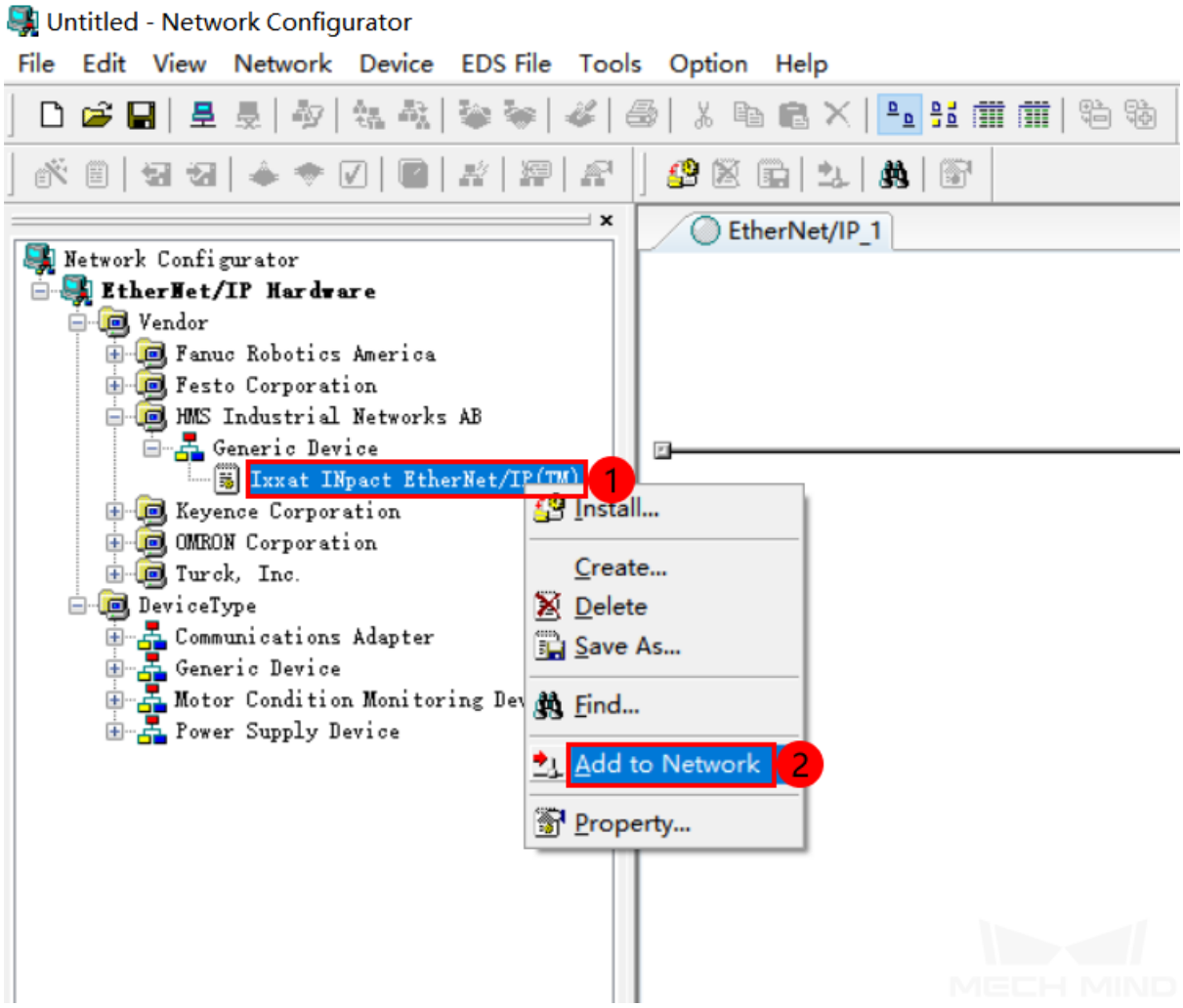


4. Locate and select the EDS file and click on *Open*.

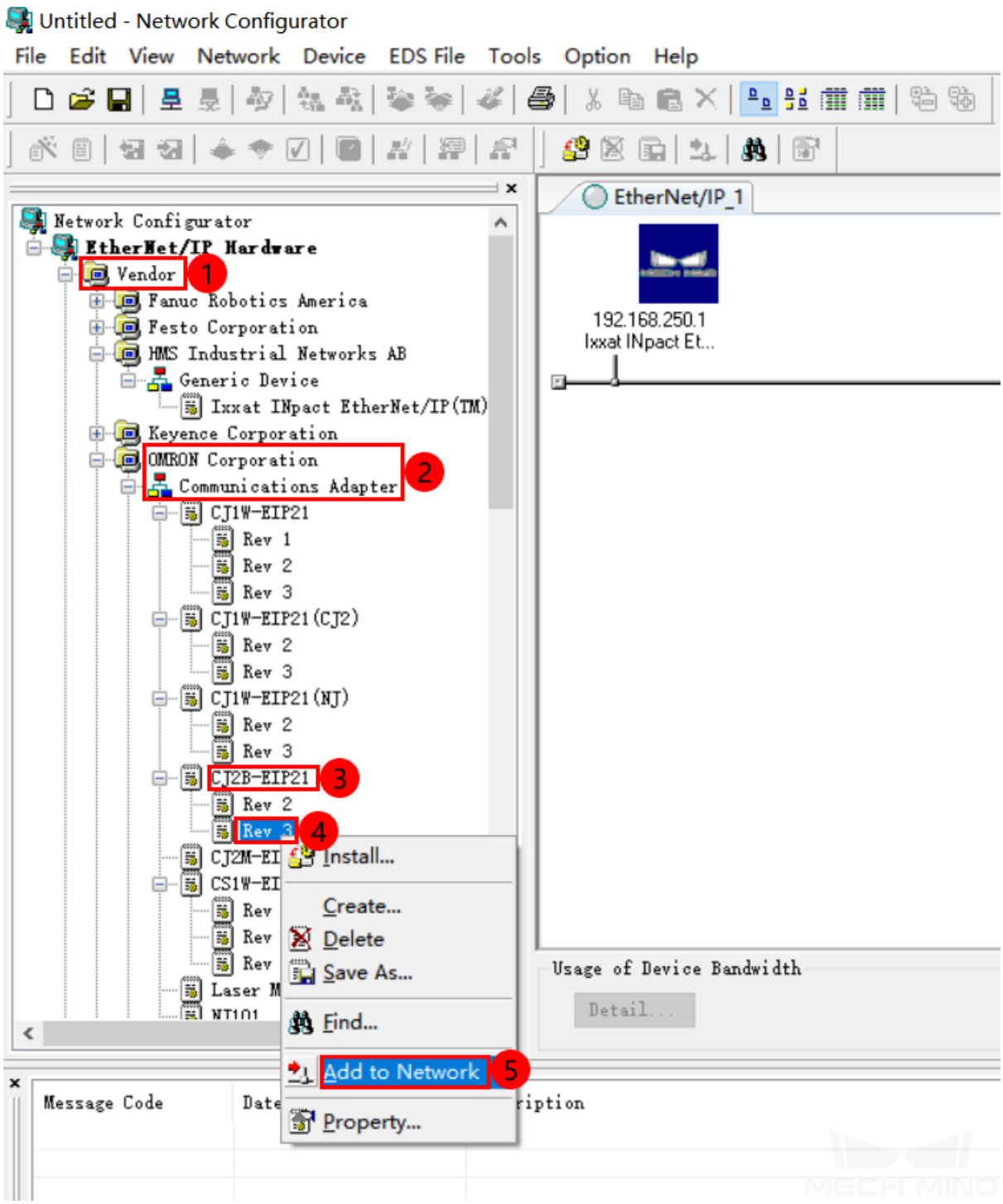


> **Attention:** The EDS file is stored in the folder where Mech-Center is installed. The default path is `Mech-Mind/Mech-Center/mech_interface/ETHERNETIP`. If CX-Programmer is not installed on the same IPC where Mech-Center is installed, you can copy and paste the **ETHERNETIP** folder to the PC with CX-Programmer installed.

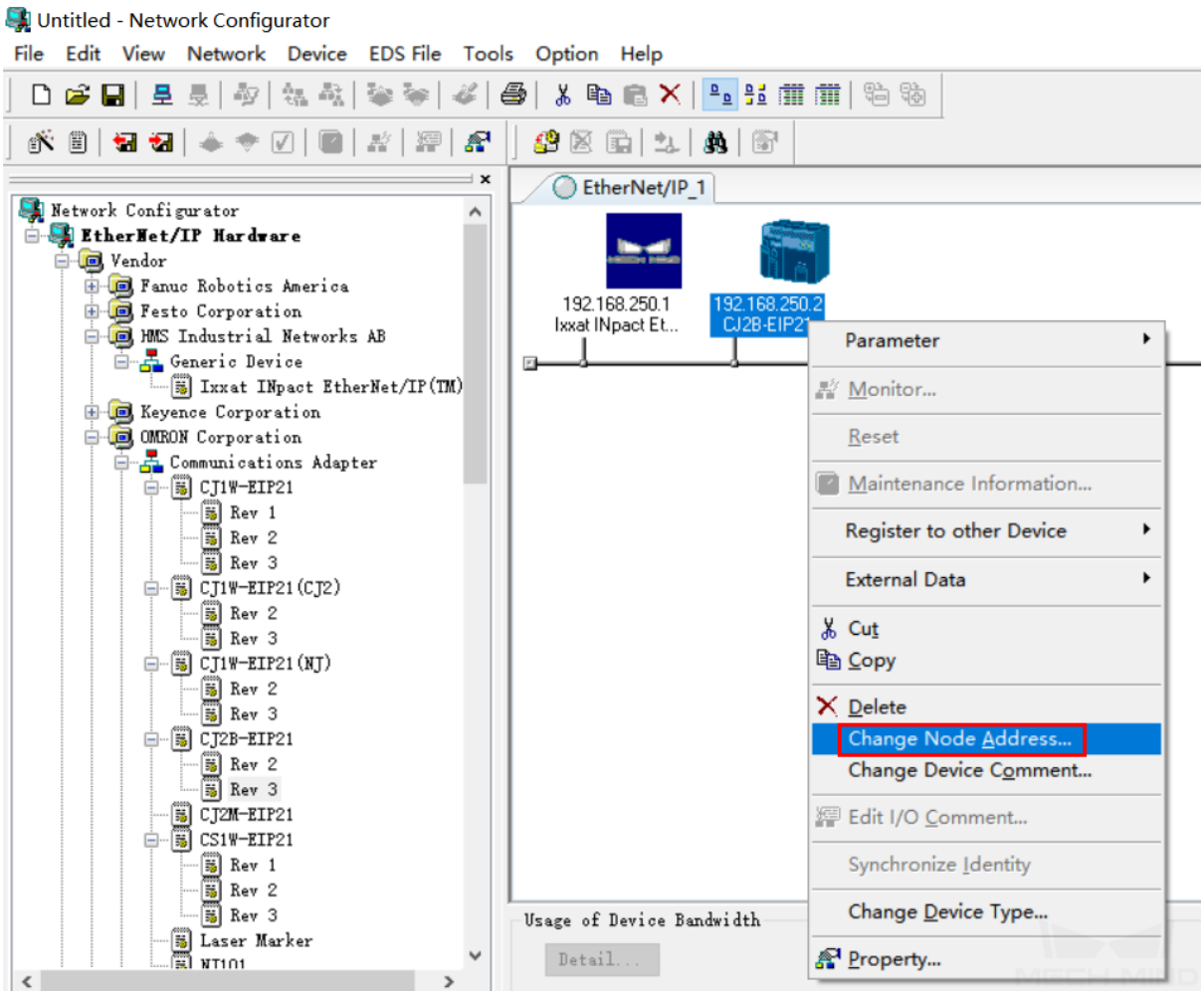
5. In the **Untitled - Network Configurator** panel, go to *EtherNet→ IP HardWare→ Vendor→ HMS Industrial Networks AB→ Generic Device*. Right click on **Ixxat INpact EtherNet/IP(TM)** and select **Add to Network** in the context menu.

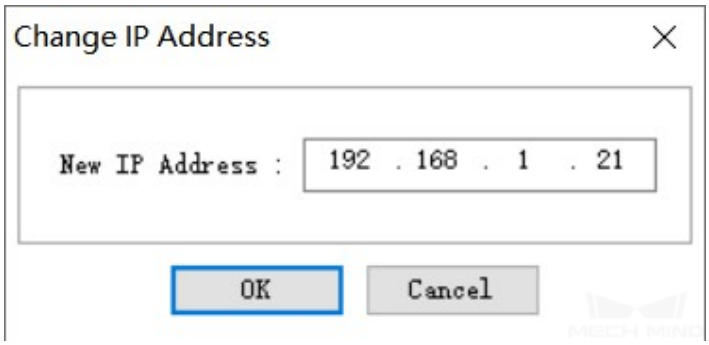


6. Then go to *EtherNet/IP Hardware→ Vendor→ OMRON Corporation→ Communications Adapter→ CJ2B-EIP21*. Right click on **Rev 3** and select **Add to Network** in the context menu.

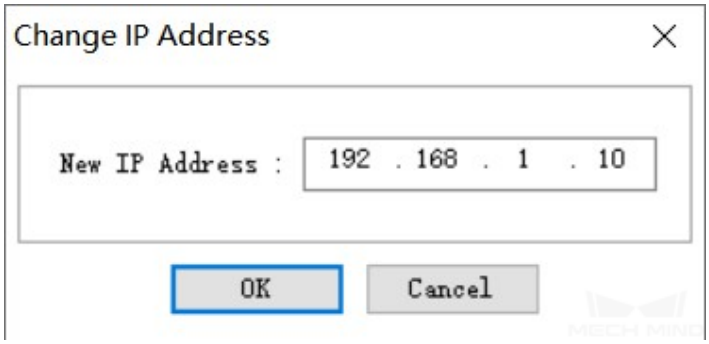


7. In **EtherNet/IP_1** window, right click on the **SJ2B-EIP21** device icon, select **Change Node Adress** in the context menu, and then configure the IP address to 192.168.1.21.

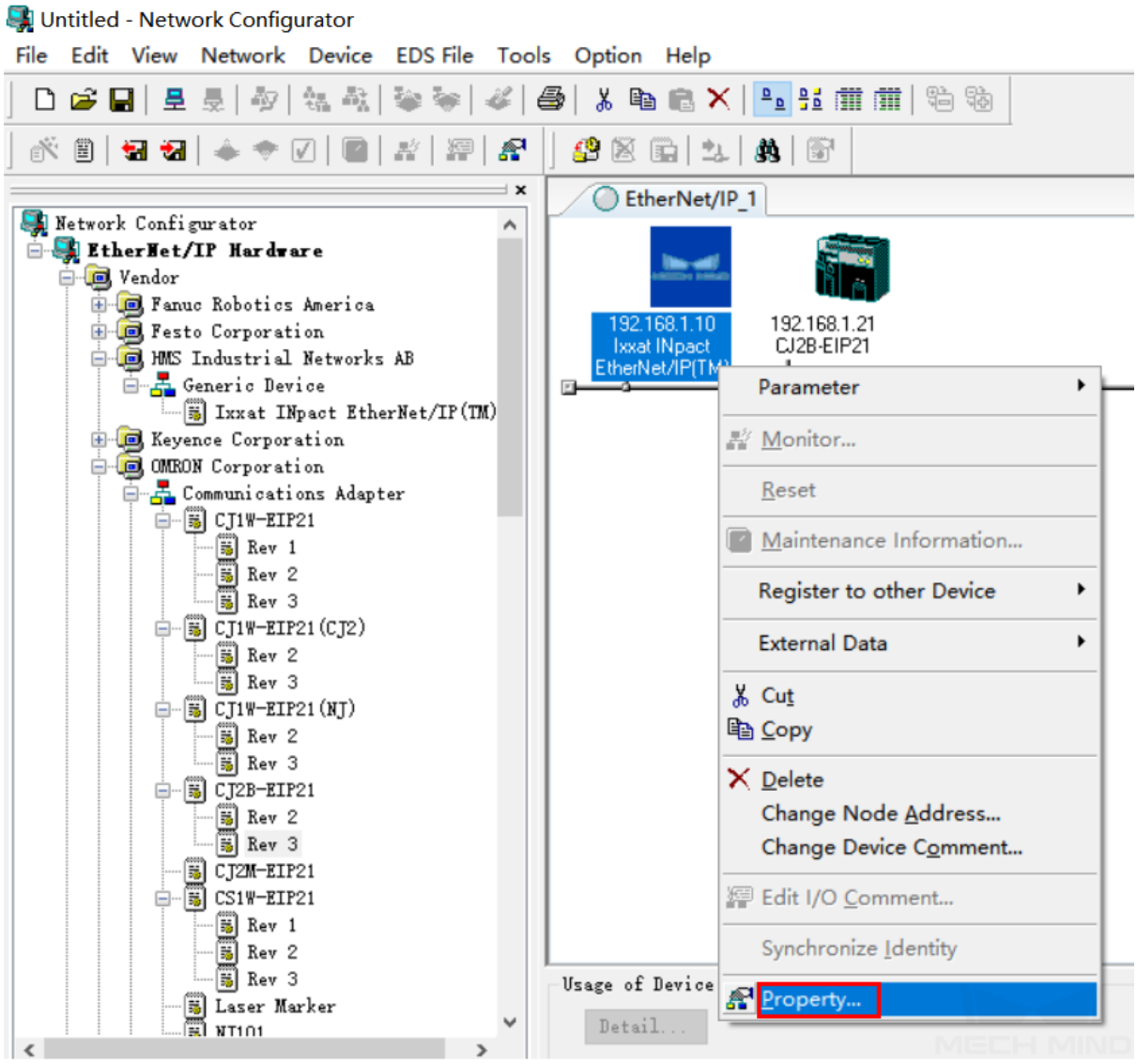




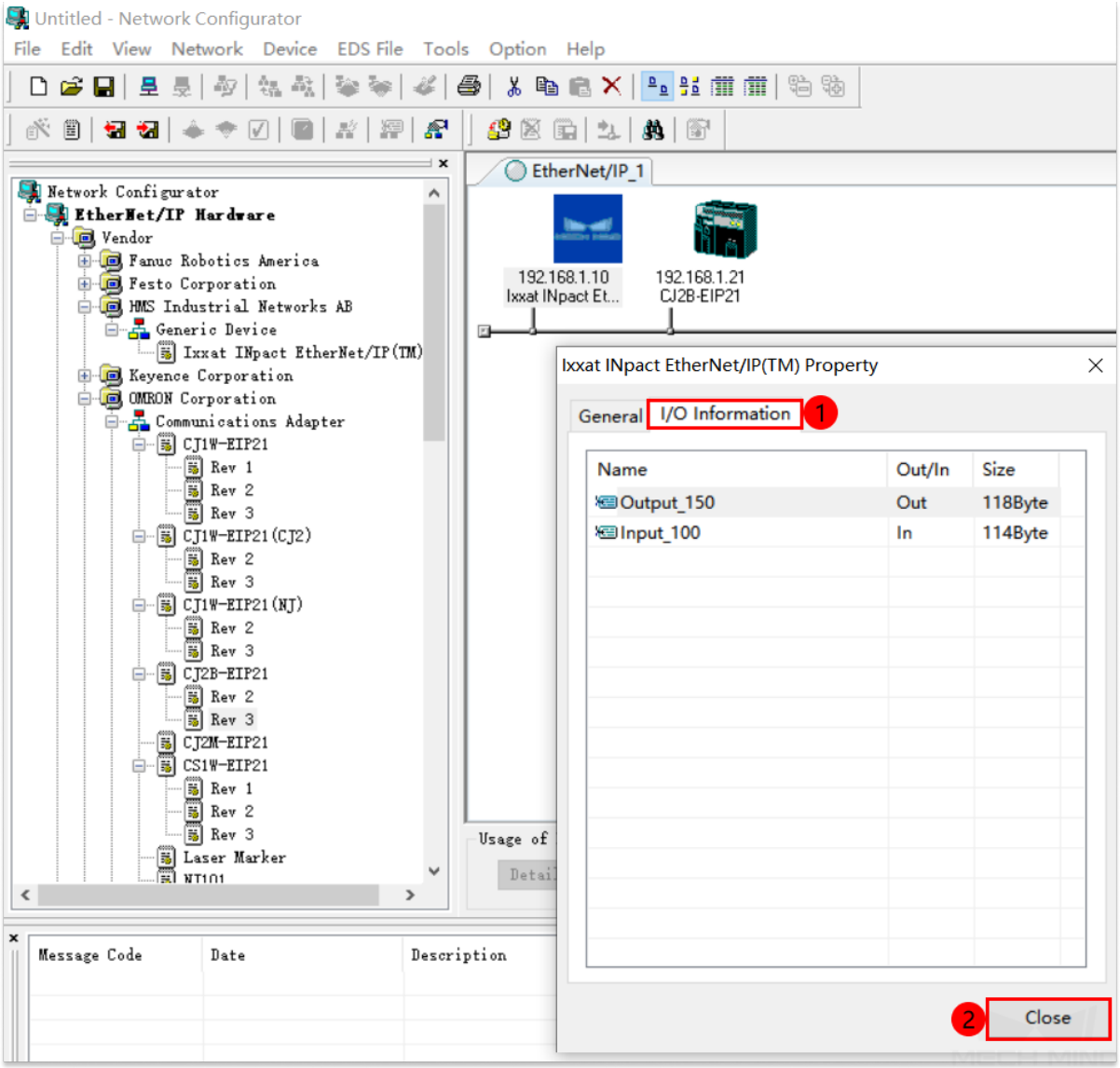
8. Right click on Mech-Mind visual device icon, select **Change Node Adress** in the context menu, and then configure the IP address to 192.168.1.10.



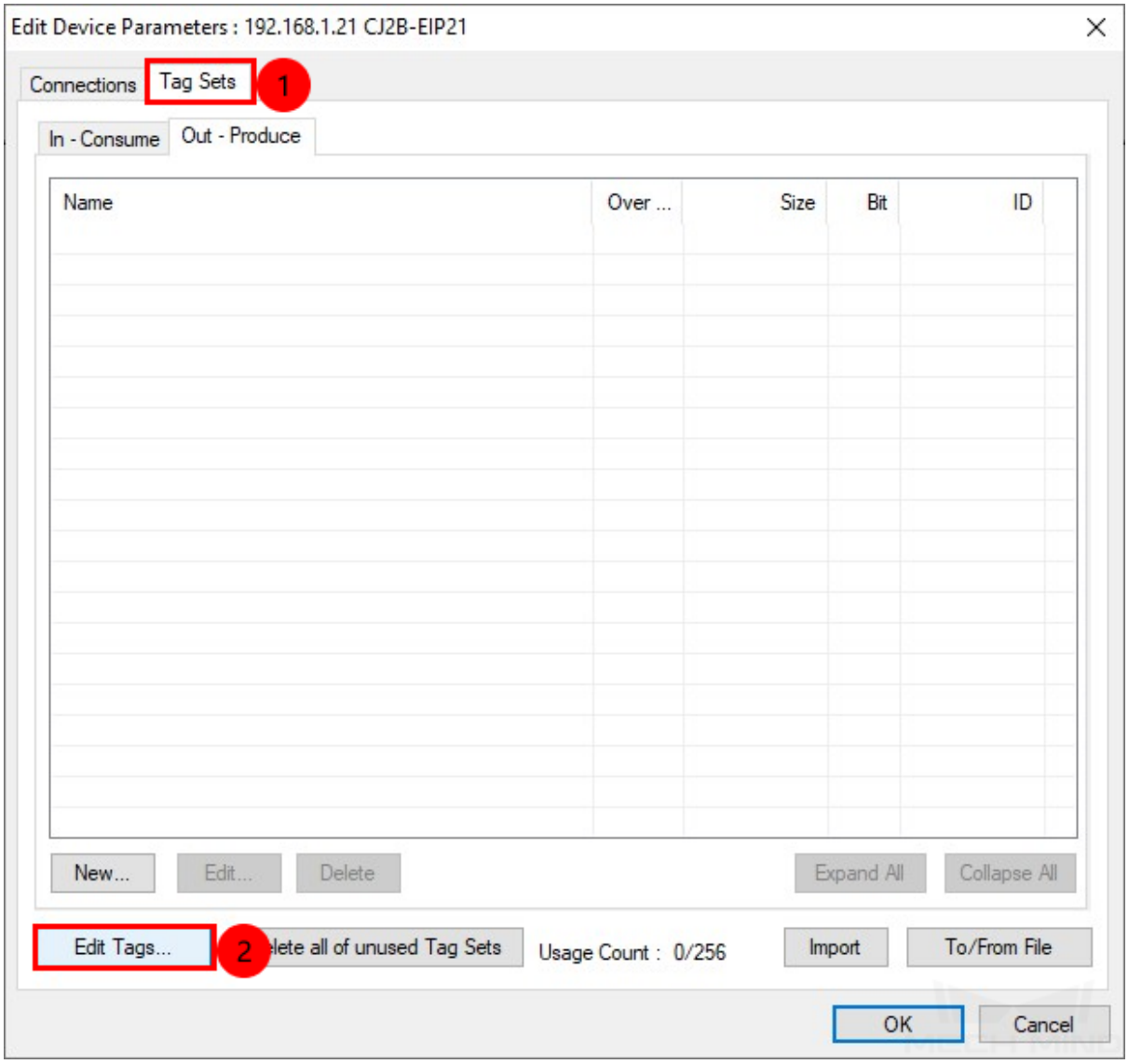
9. Right click on Mech-Mind visual device icon, and select **Property**.



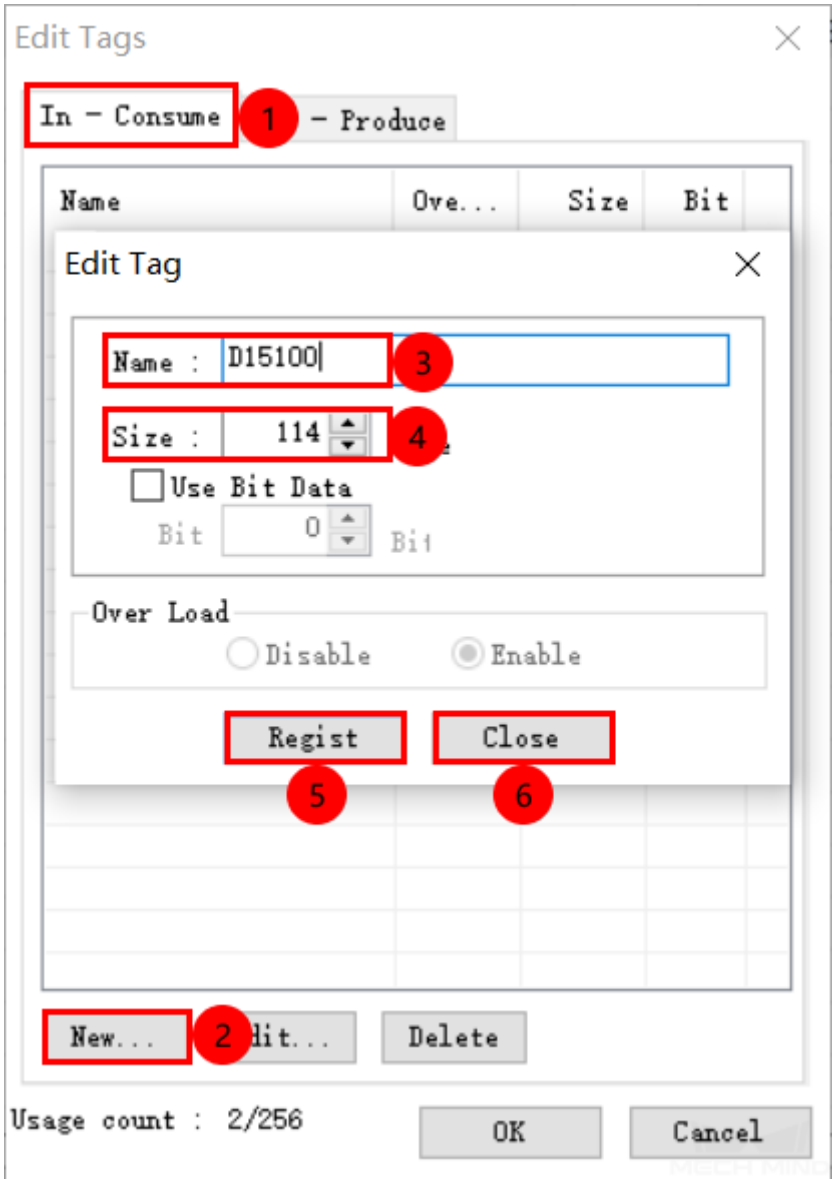
10. In the **Property** window, click on *I/O Information* to check the I/O size, and then click on *Close*.



11. In **EtherNet/IP_1** window, double click on the **CJ2B-EIP21** icon. Select *Tag Sets* in the pop-up **Edit Device Parameters** window, and then click on *Edit Tags*.

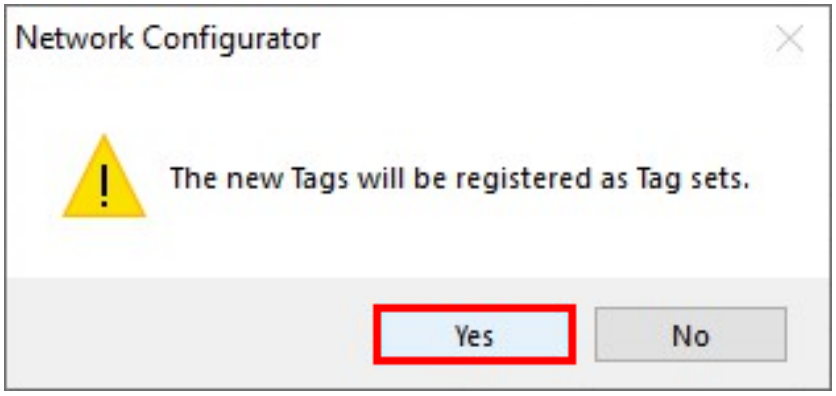


12. Select *In-Consume*, and click on *New*. Then edit the tag according to actual situation. Click on *Regist* and *Close*.

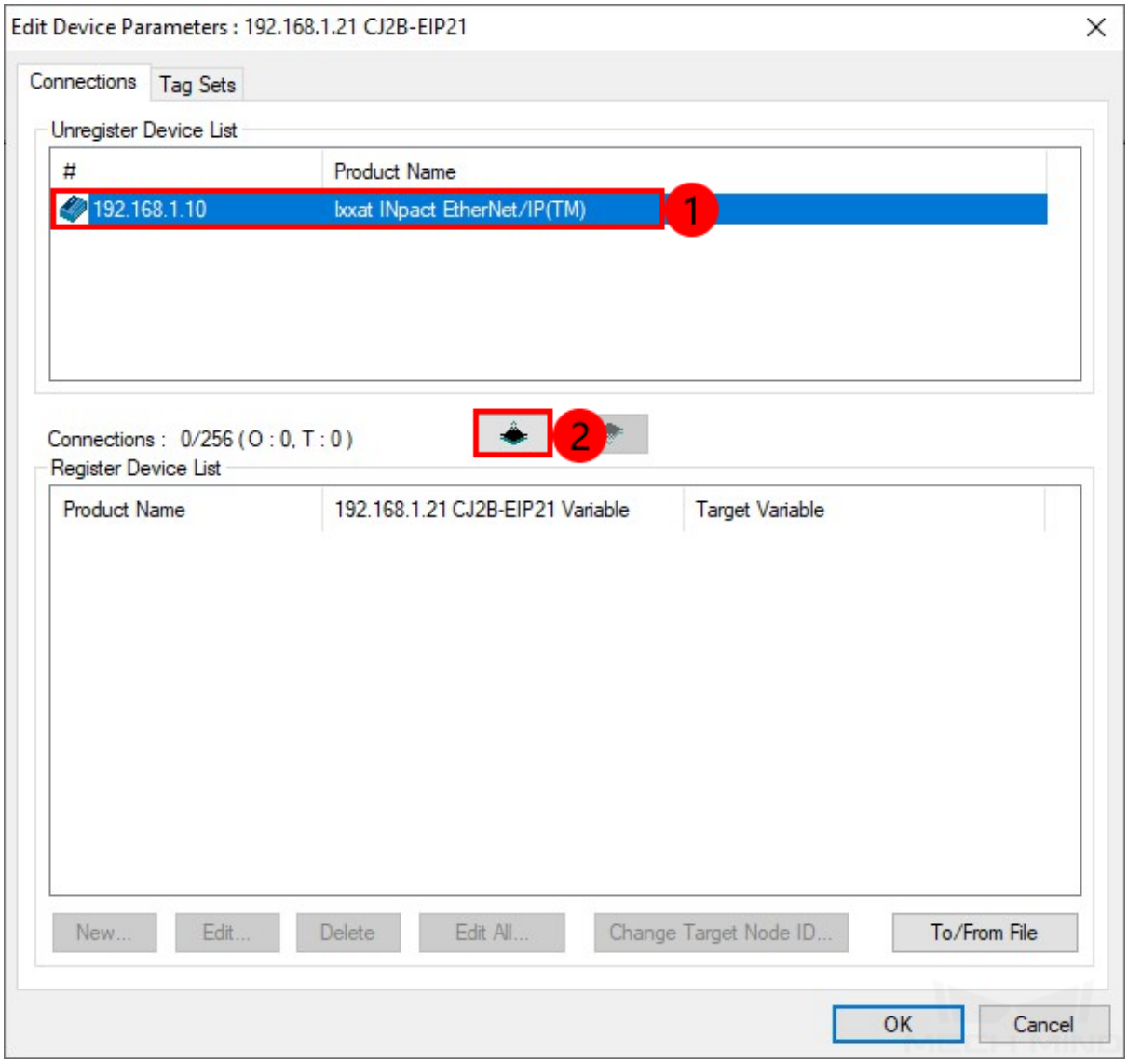


13. Select *Out-Produce* and edit the tag in the same way. Click on *OK* in the **Edit Tags** window in the end.

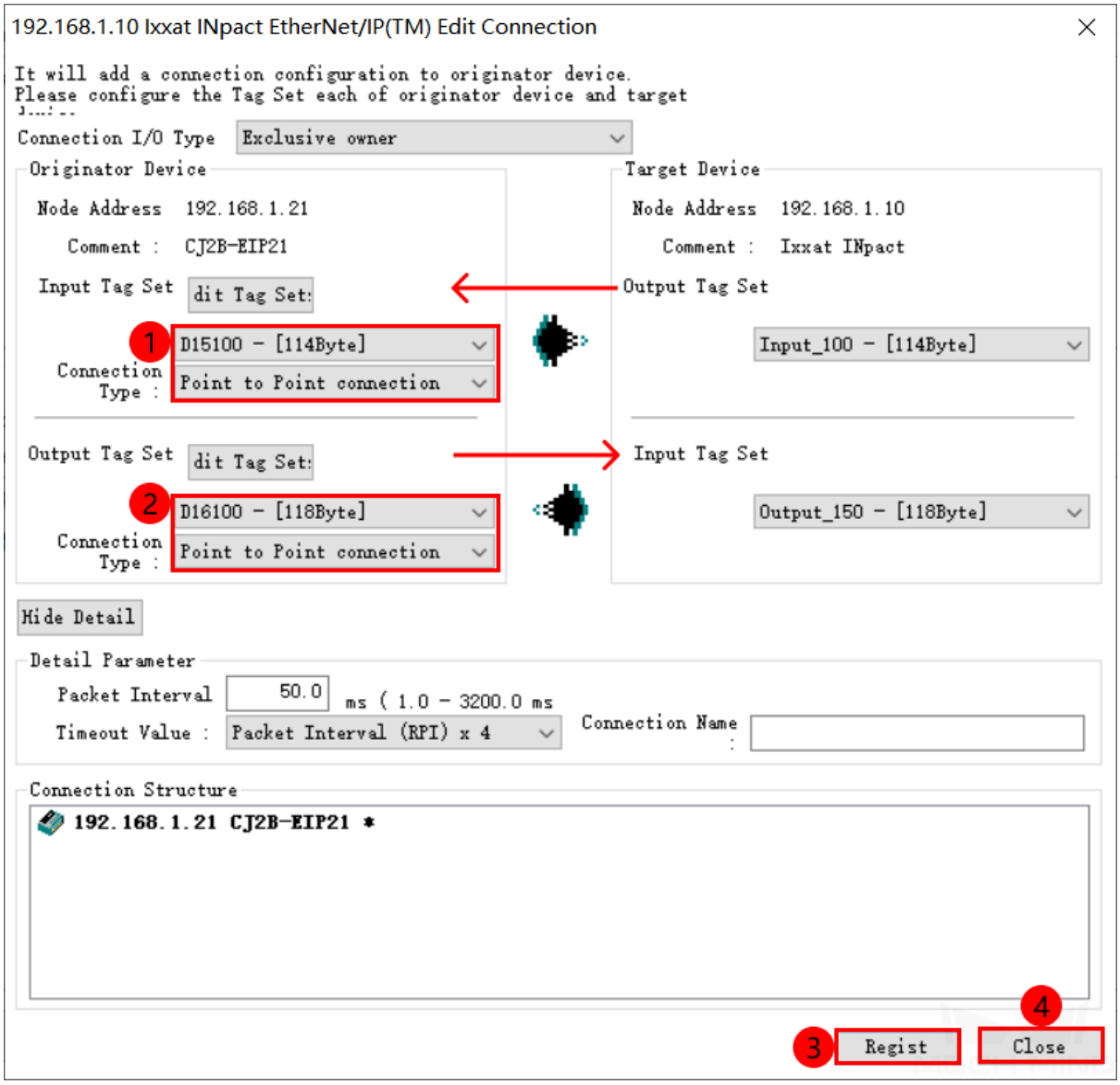
14. Select *Yes* in the pop-up window.



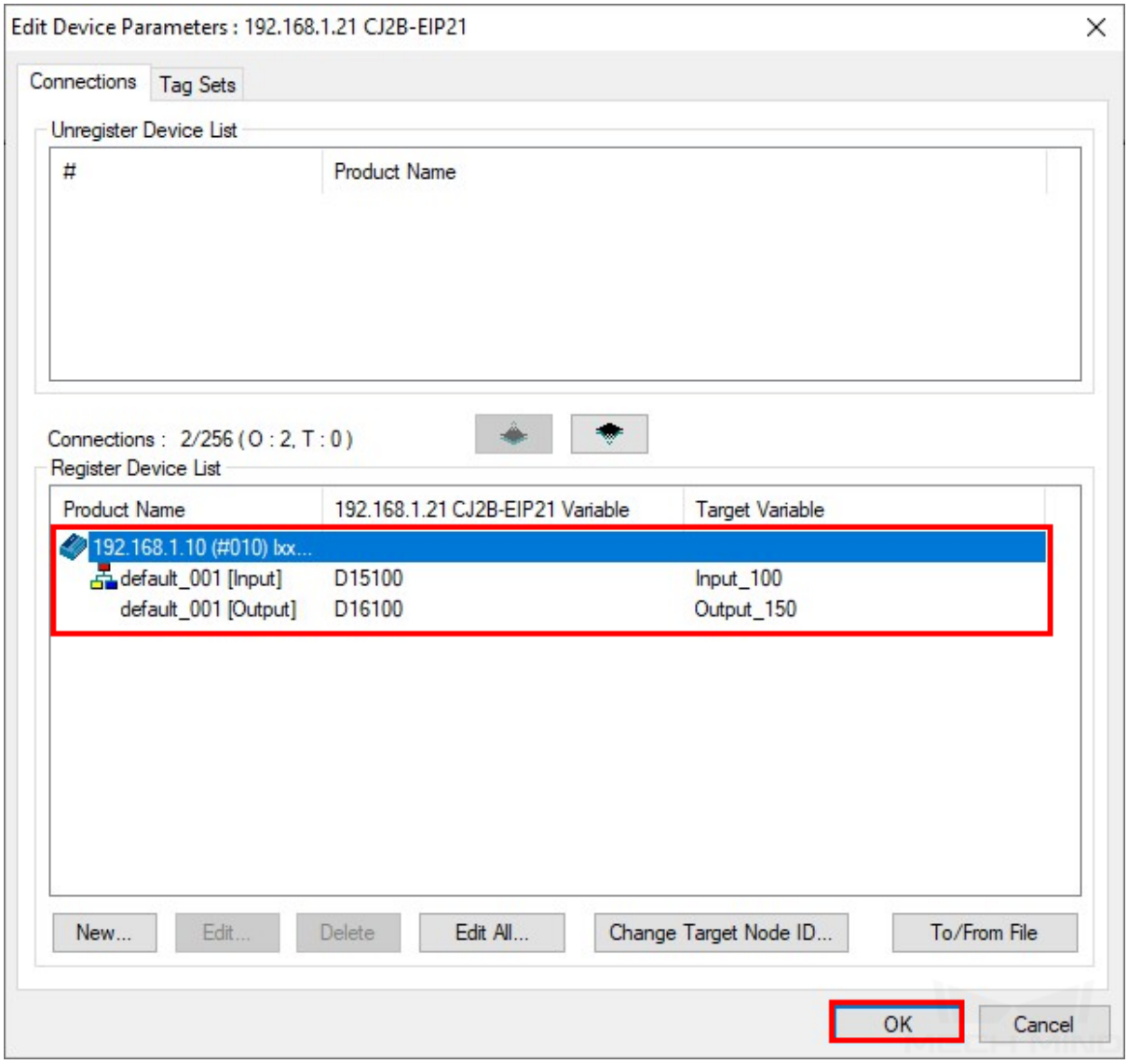
15. Select **Ixxat INpact EtherNet/IP(TM)** in the **Unregister Device List**, click on  . Then double click on **Ixxat INpact EtherNet/IP(TM)** in the **Register Device List**.



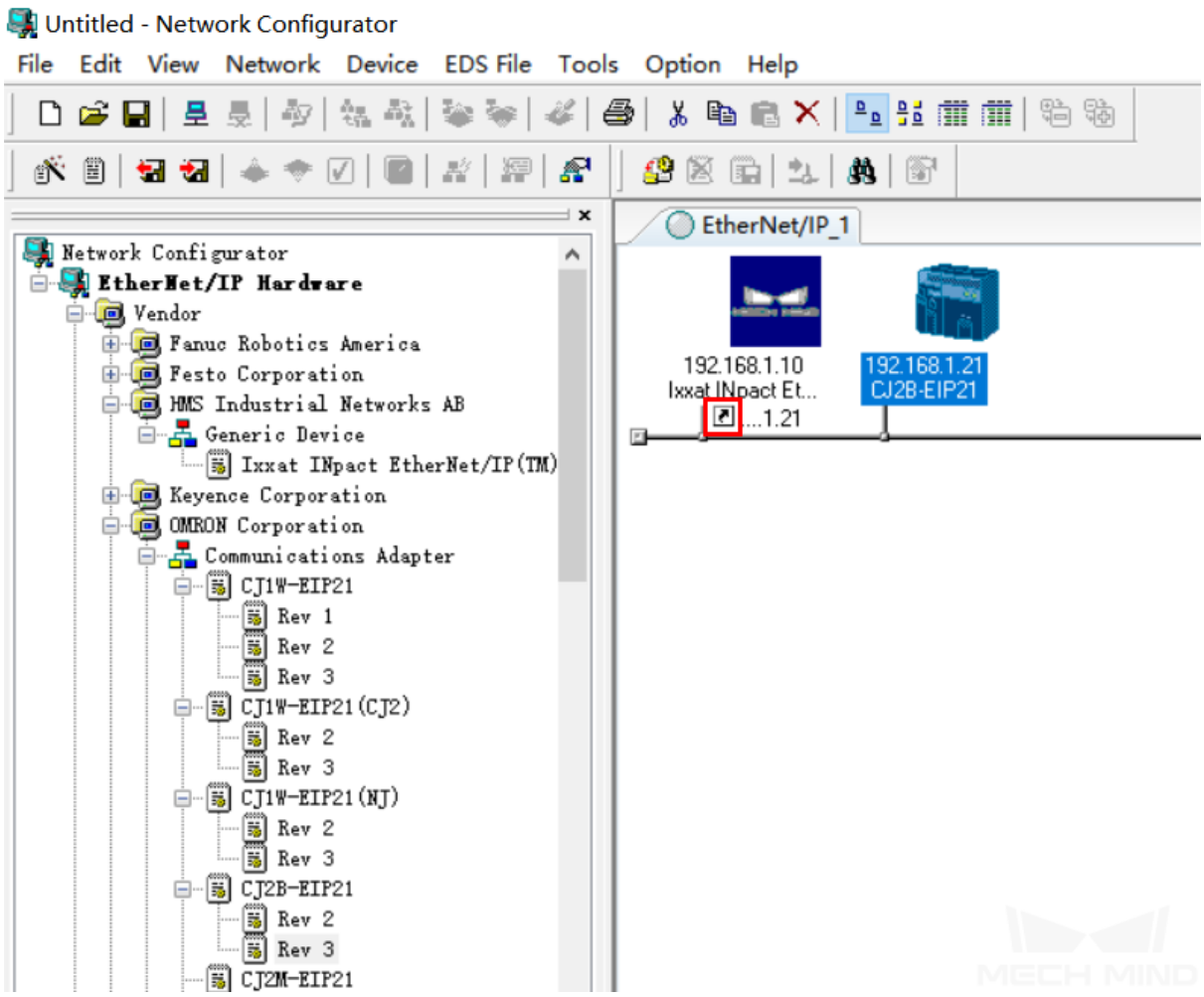
16. A **Ixxat INpact EtherNet/IP(TM) Edit Connection** window showing Input and Output will appear. Configure the tag sets as shown below. Click on *Regist* and *Close* to close the window.



17. Now the tag sets will appear in the **Register Device List**, then click on *OK*.

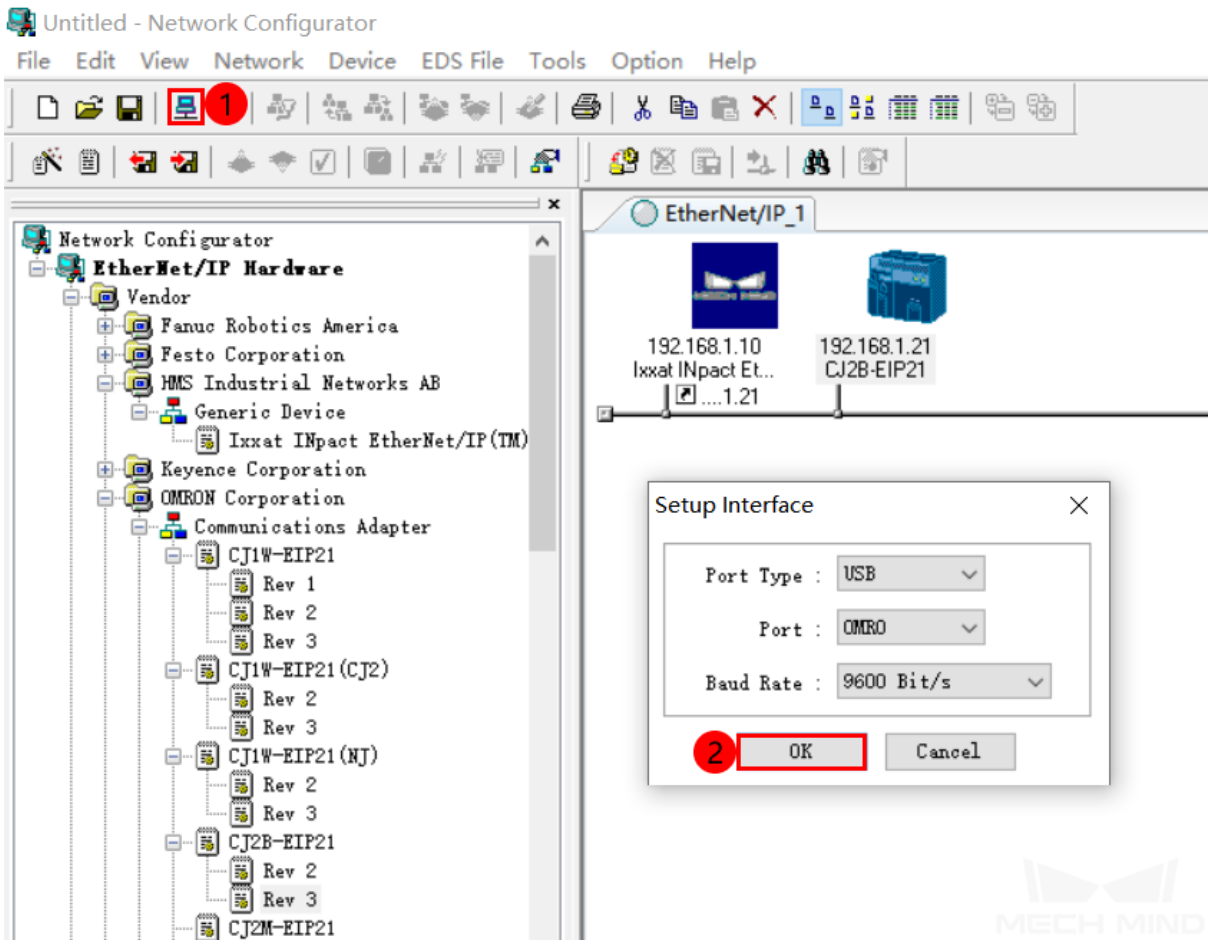


18. After the IPC is connected, a small arrow will appear as shown below.

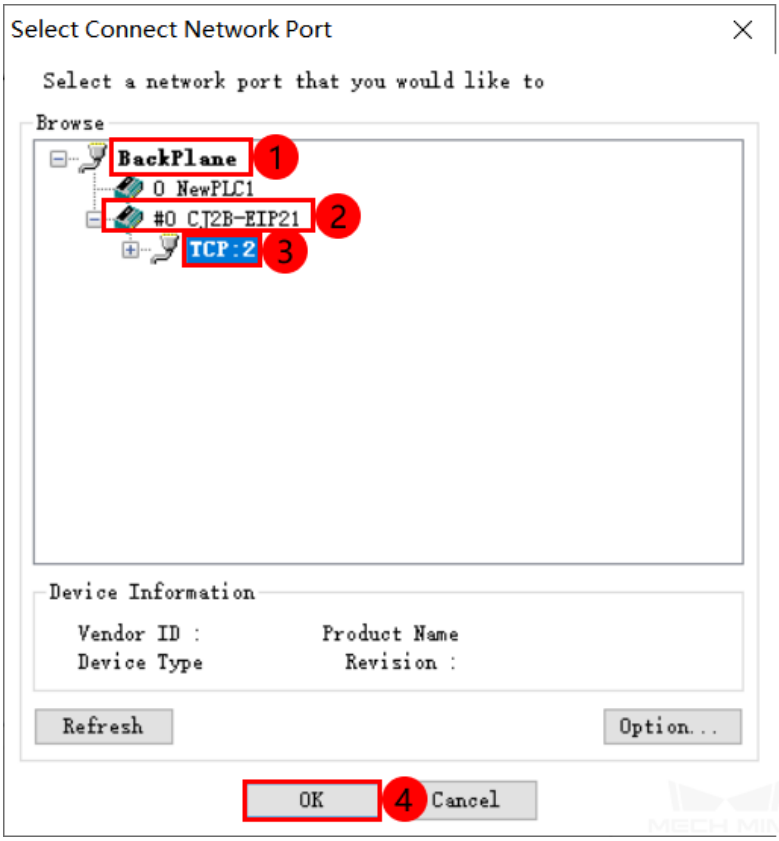


Download Hardware Configuration to PLC

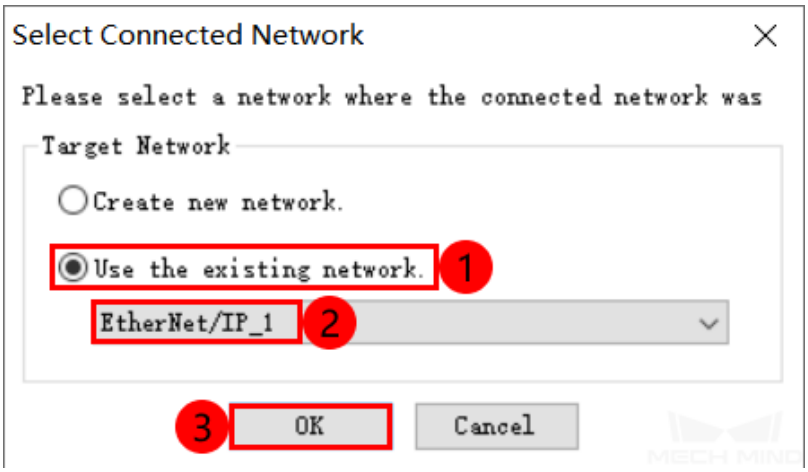
1. Click on  in the **Network Configurator** window. Then select *OK* in the pop-up **Setup Interface** window.



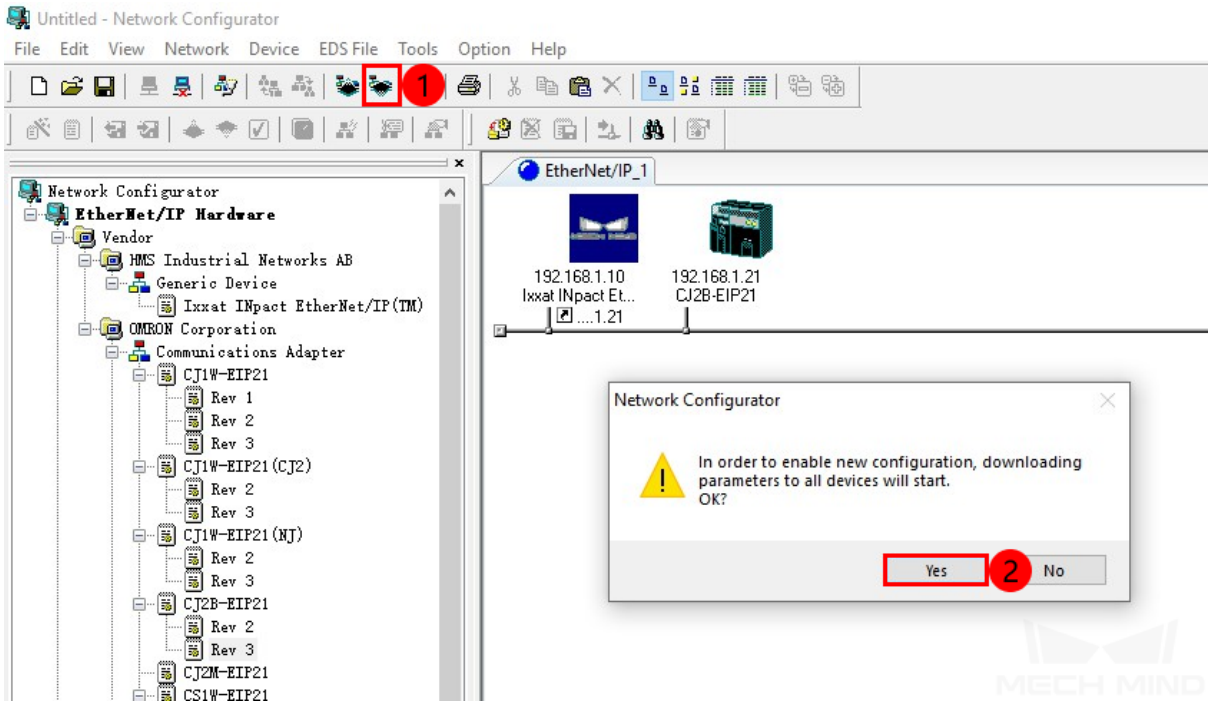
2. Now you can see a **Select Connect Network Port** window. Select *BackPlane→ #0 CJ2B-EIP21→TCP:2*, and then click on *OK*.



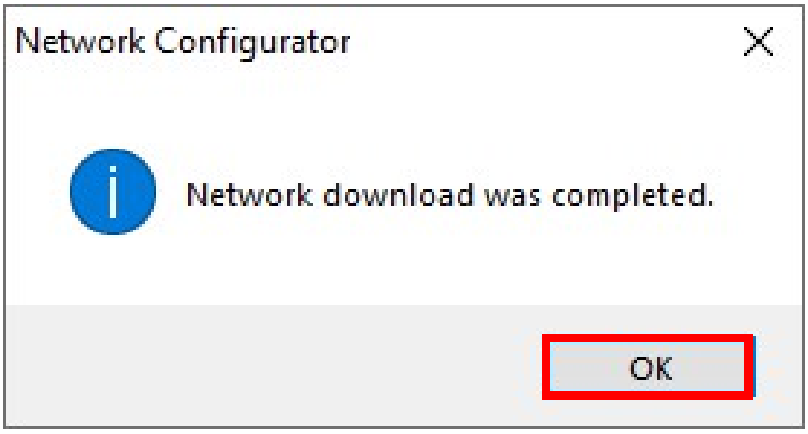
3. Select *Using the existing network→ EtherN IP_1* in the **Select Connected Network** window and click on *OK*.



4. Click on  in the **Network Configurator** window, and select *Yes* to start dowloading parameters.

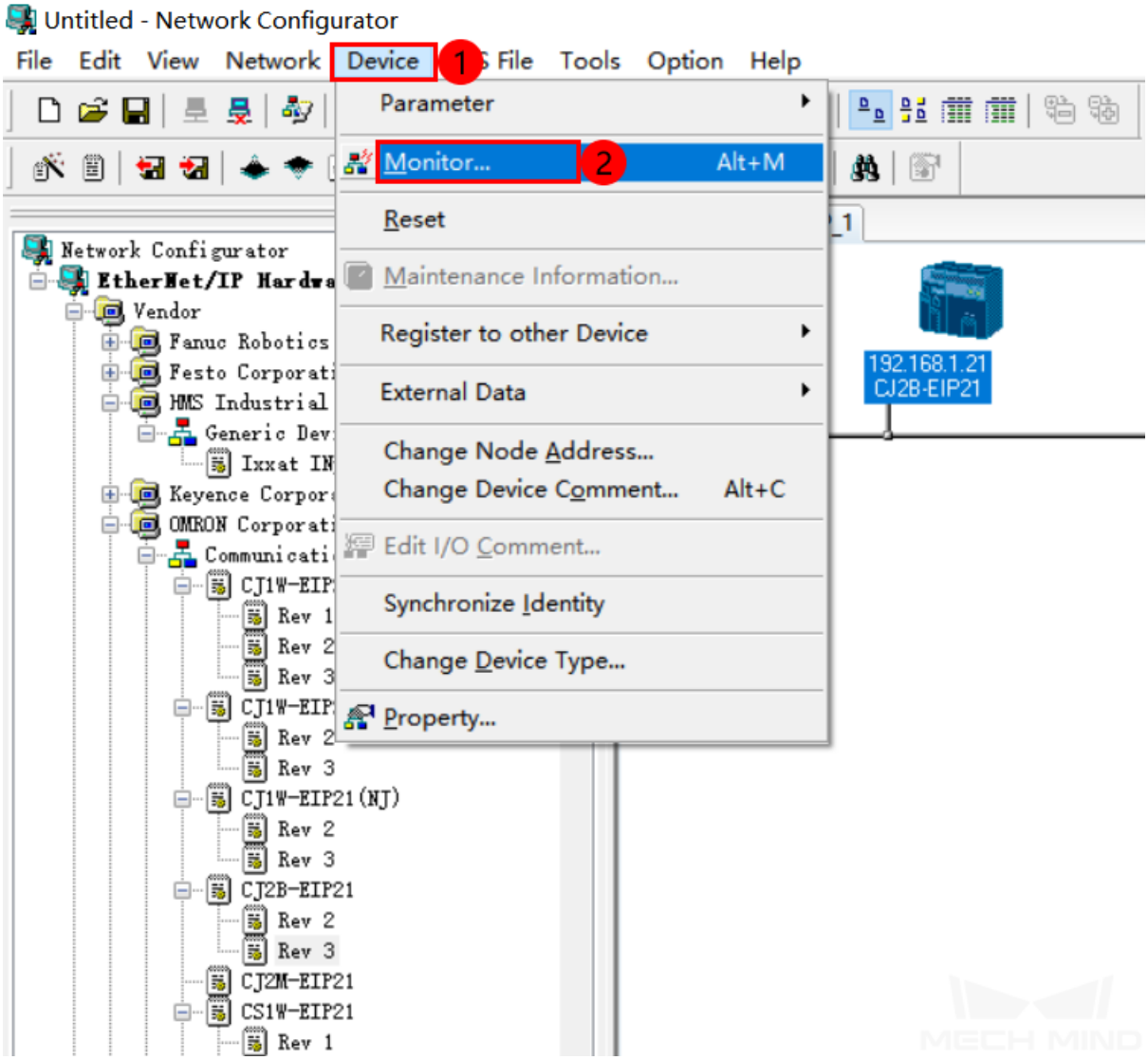


5. After the dowload is completed, click on *OK*.

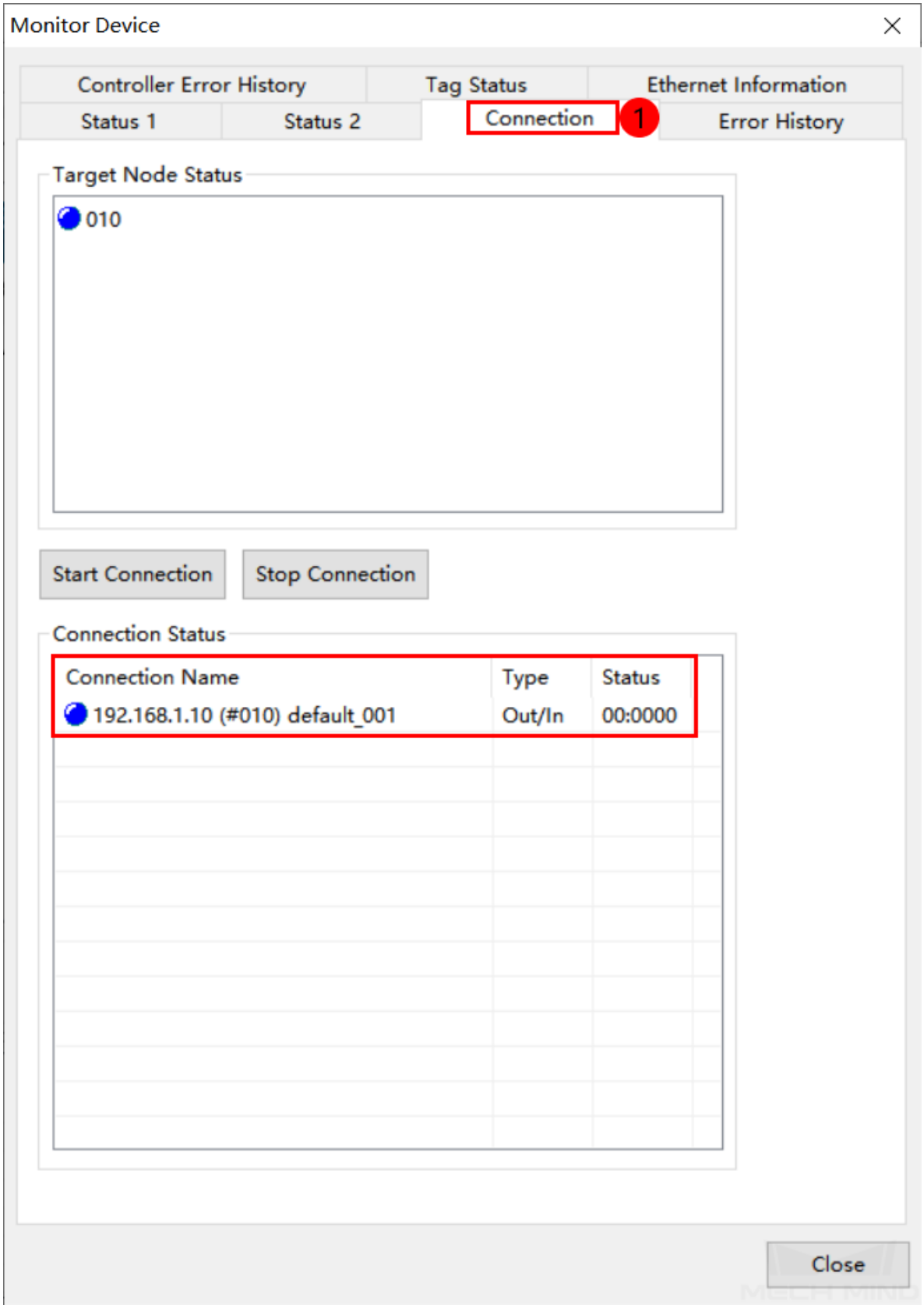


Check Communication

1. In the **Untitled-Network Configurator** window, go to *Device* → *Monitor*.

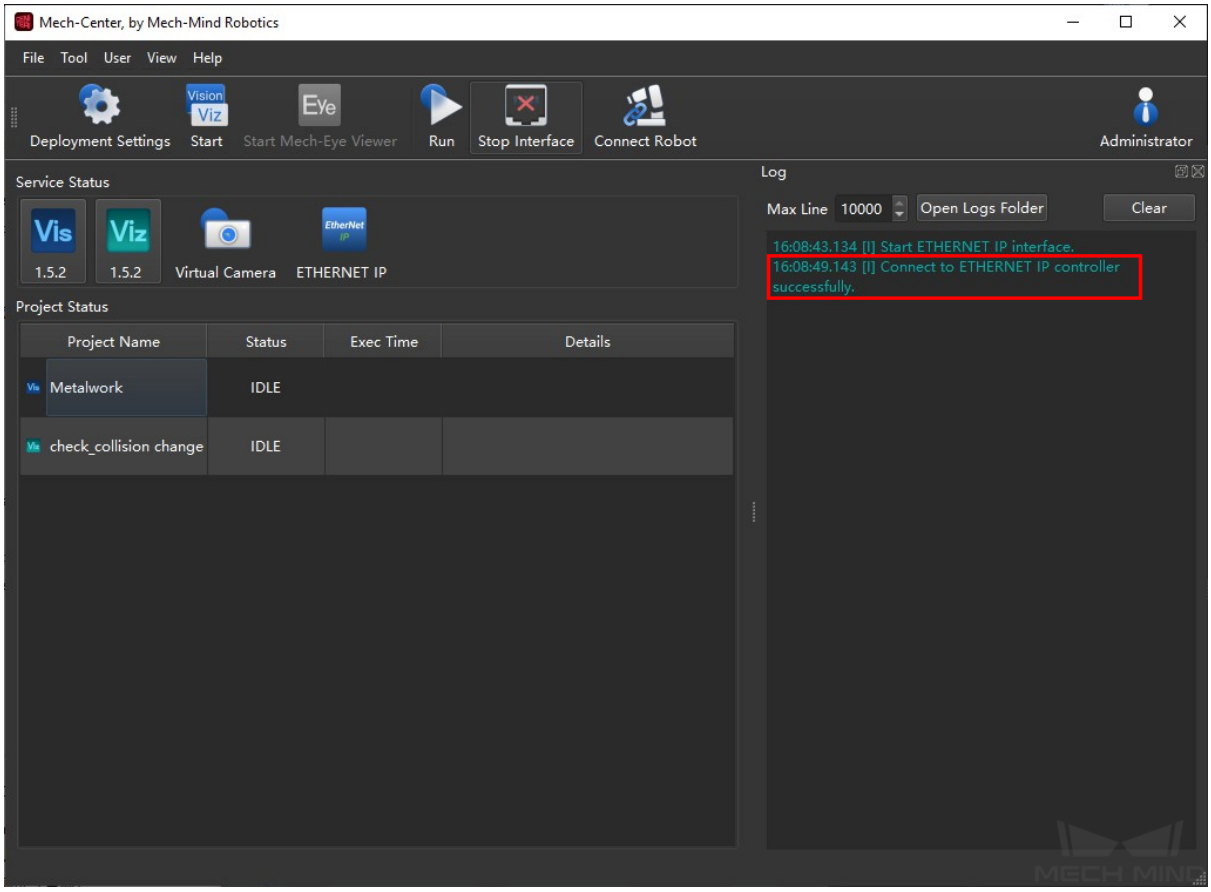


2. A **Monitor Device** window will appear. Click on *Connection*. If the connection is successful, the status indicator will be blue.


Monitor Device
Controller Error History
Tag Status
Ethernet Information
Status 1
Status 2
Connection
1
Error History
Target Node Status
010
Start Connection
Stop Connection
Connection Status
Connection Name
Type
Status
192.168.1.10 (#010) default_001
Out/In
00:0000
Close

3. The PLC is successfully connected to Mech-Center if the following message is displayed in Mech-Center **Log** panel:

   Connect to ETHERNET IP controller successfully



Note: If you don't see this log message, please check if:

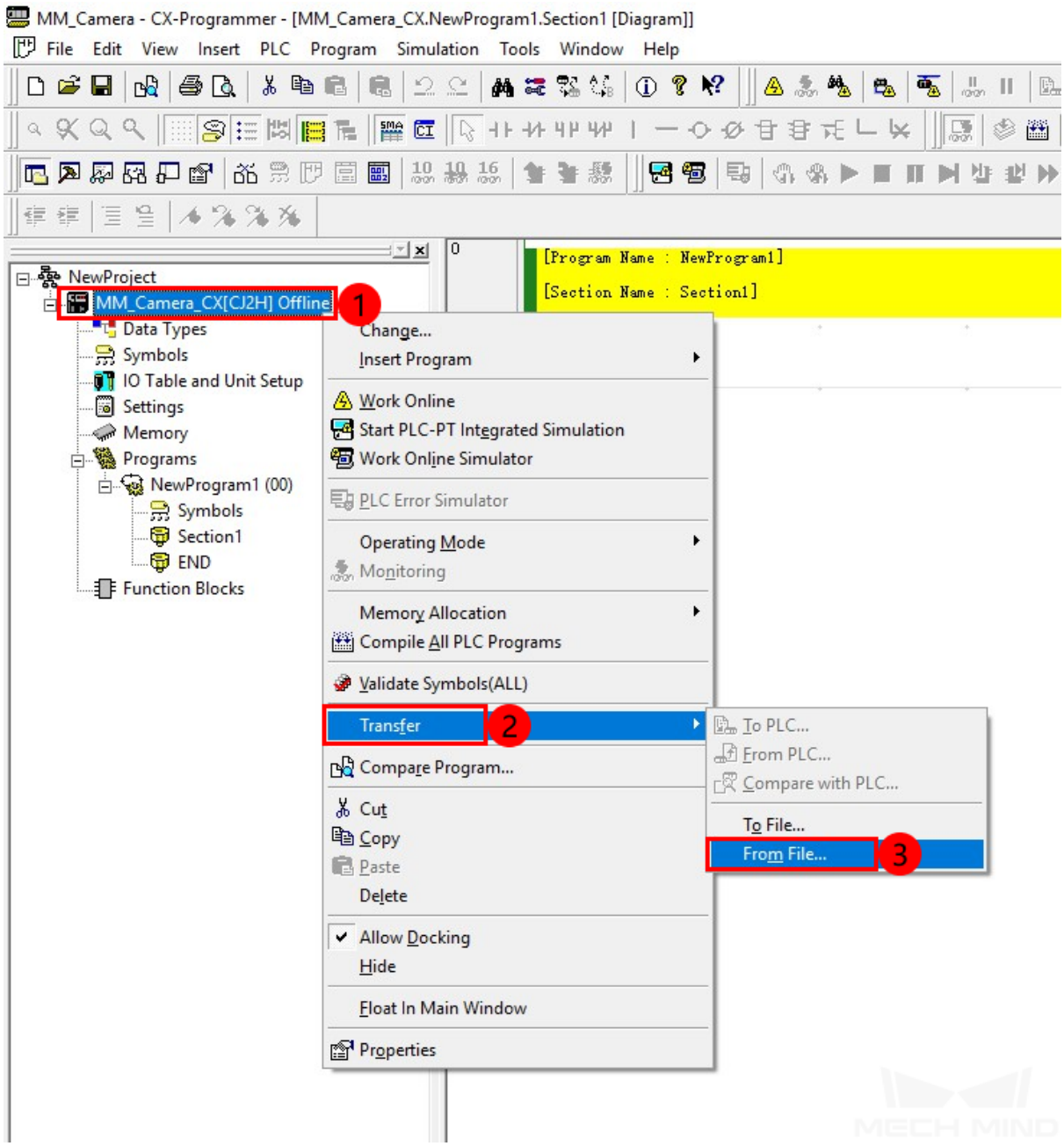
- The hardware are properly connected;
- If Mech-Interface has been started by clicking on *Start Interface* in the Toolbar;
- If the hardware configuration has been downloaded to the PLC.

2.7.5 Import Example Program and Download to PLC

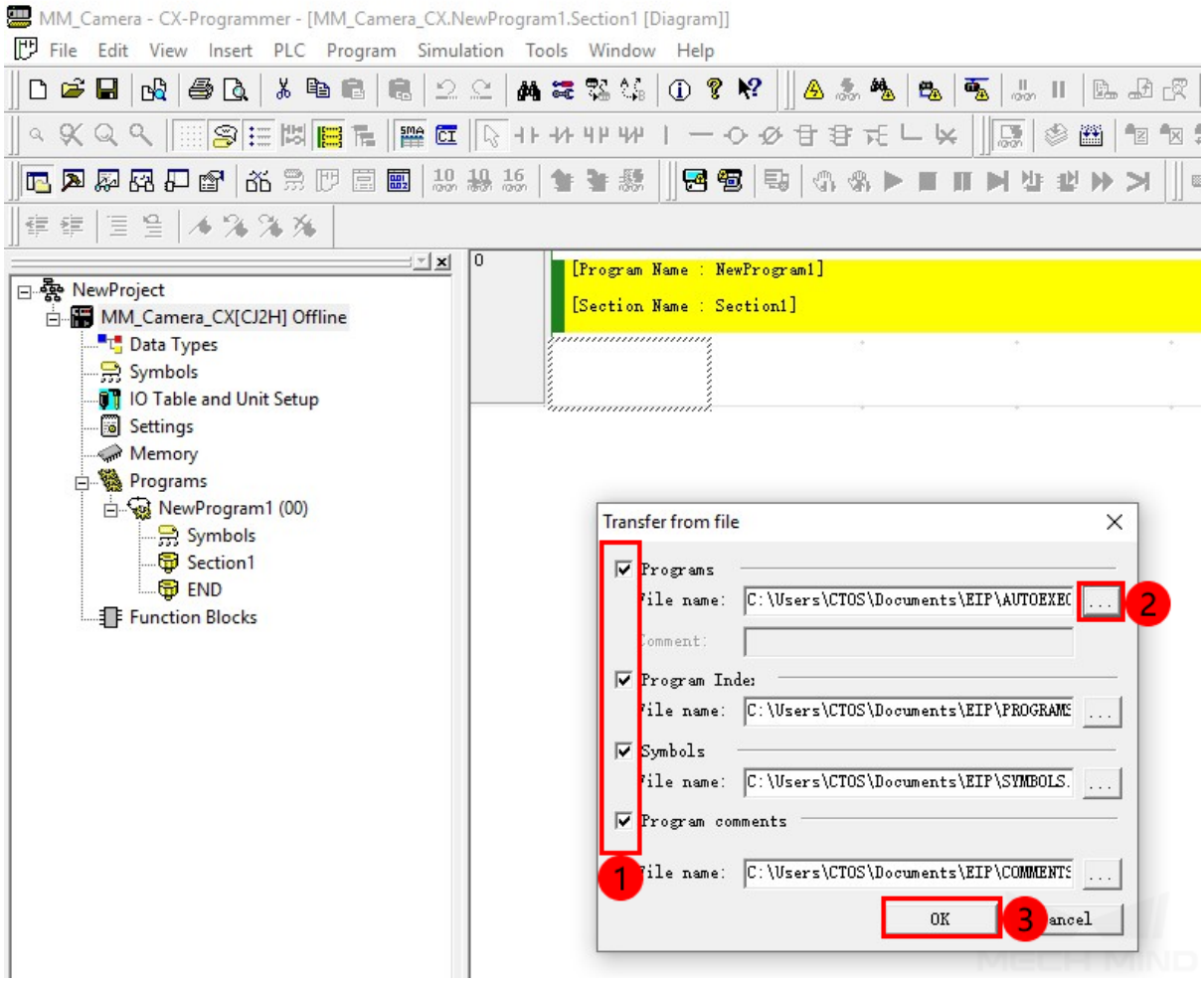
Note: Before you add the example program to a project already in use, it is recommended to import it to a new project and test it first. In the following steps, the project created earlier is used to import and test the example program.

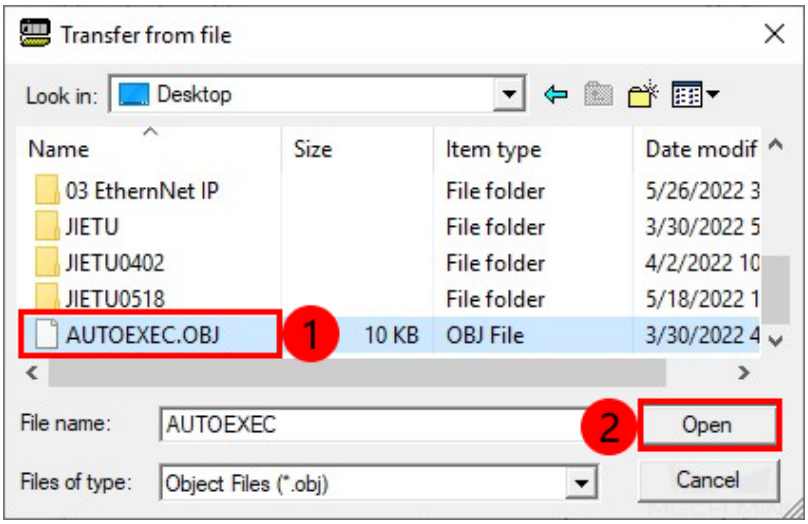
Import Example Program Files

1. Open CX-Programmer, right click on *MM_Camera_CX[CJ2H] Offline* and select *Transfer* → *From File*.

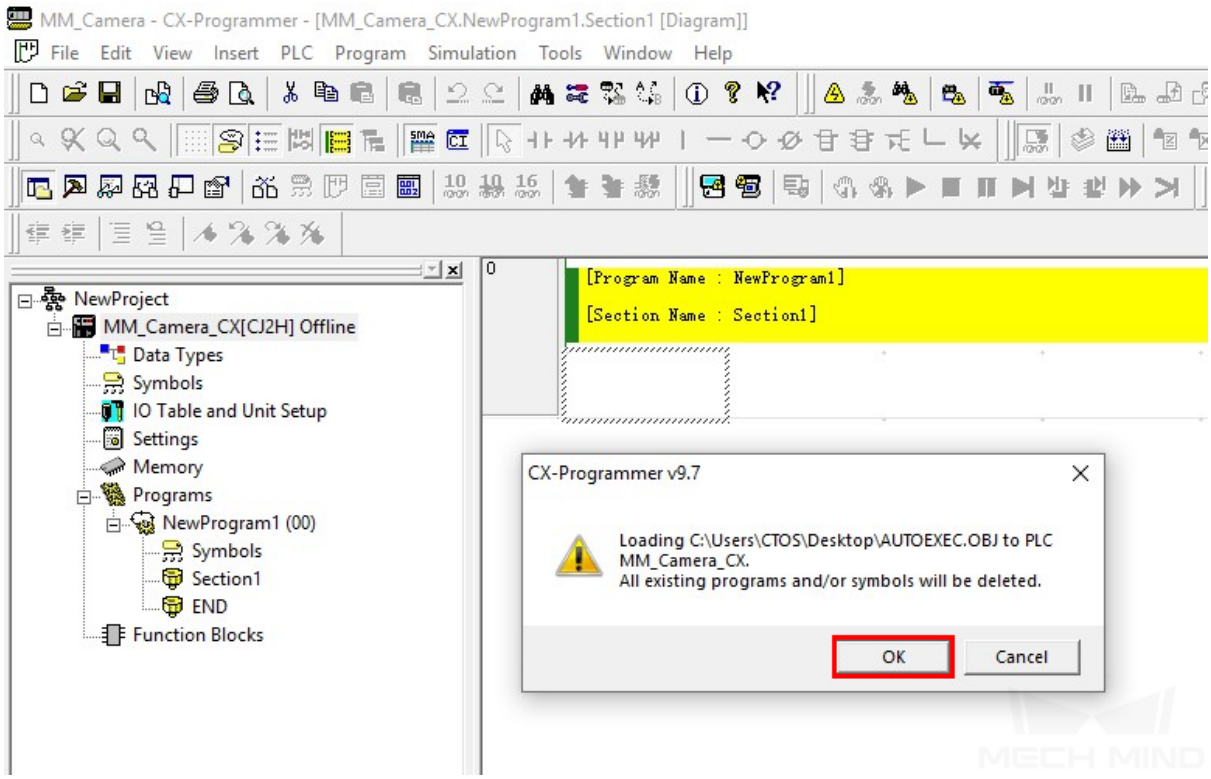


2. Check **Programs**, **Program Index**, **Symbols** and **Program comments**. Then click on ··· next to the **File name** under **Programs**, select the OBJ file, and click on *Open*. If other files are stored in the same folder, they will be auto filled. Click on *OK* in the end.



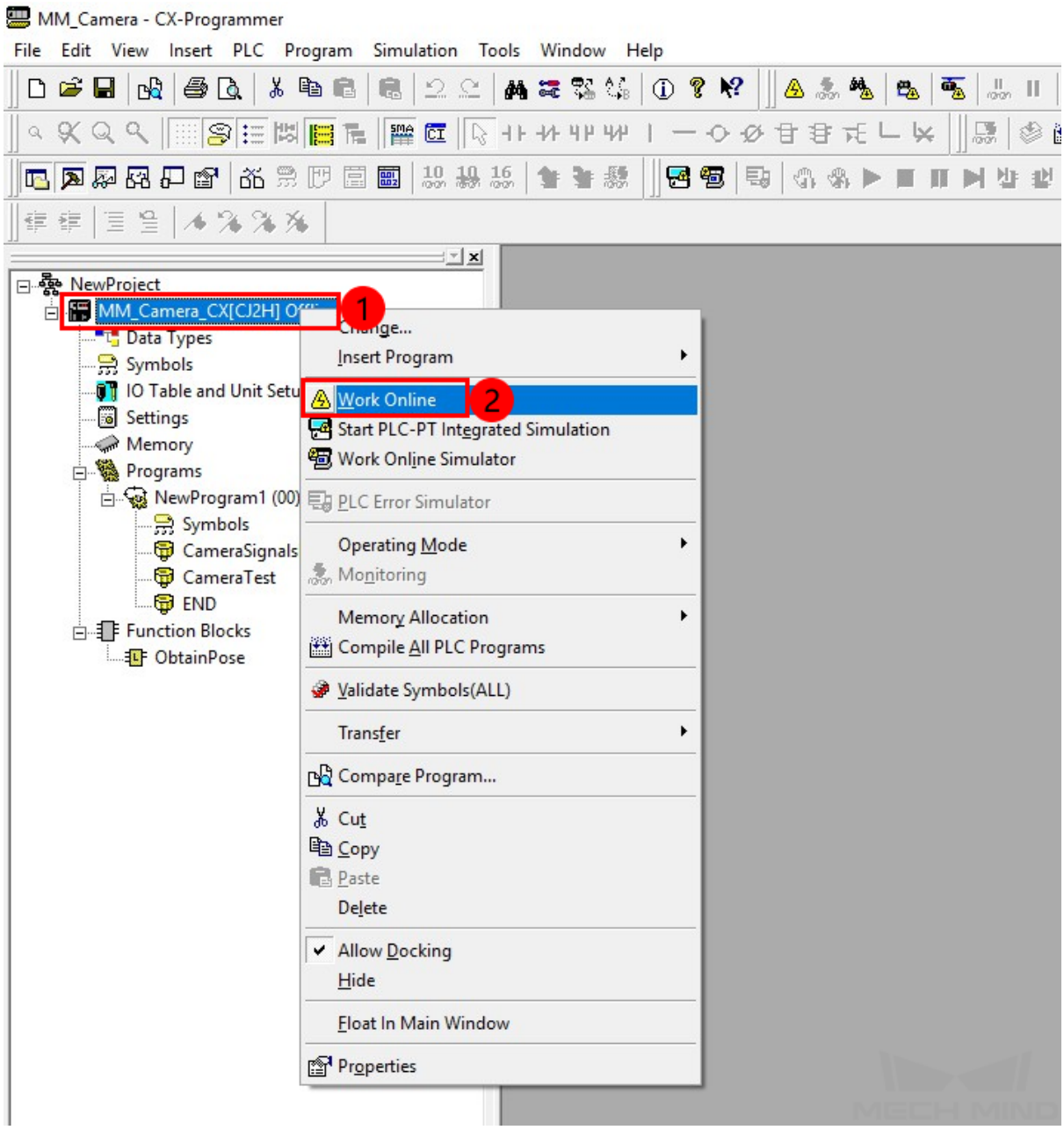


3. Select *OK* in the pop-up window to load the example program.

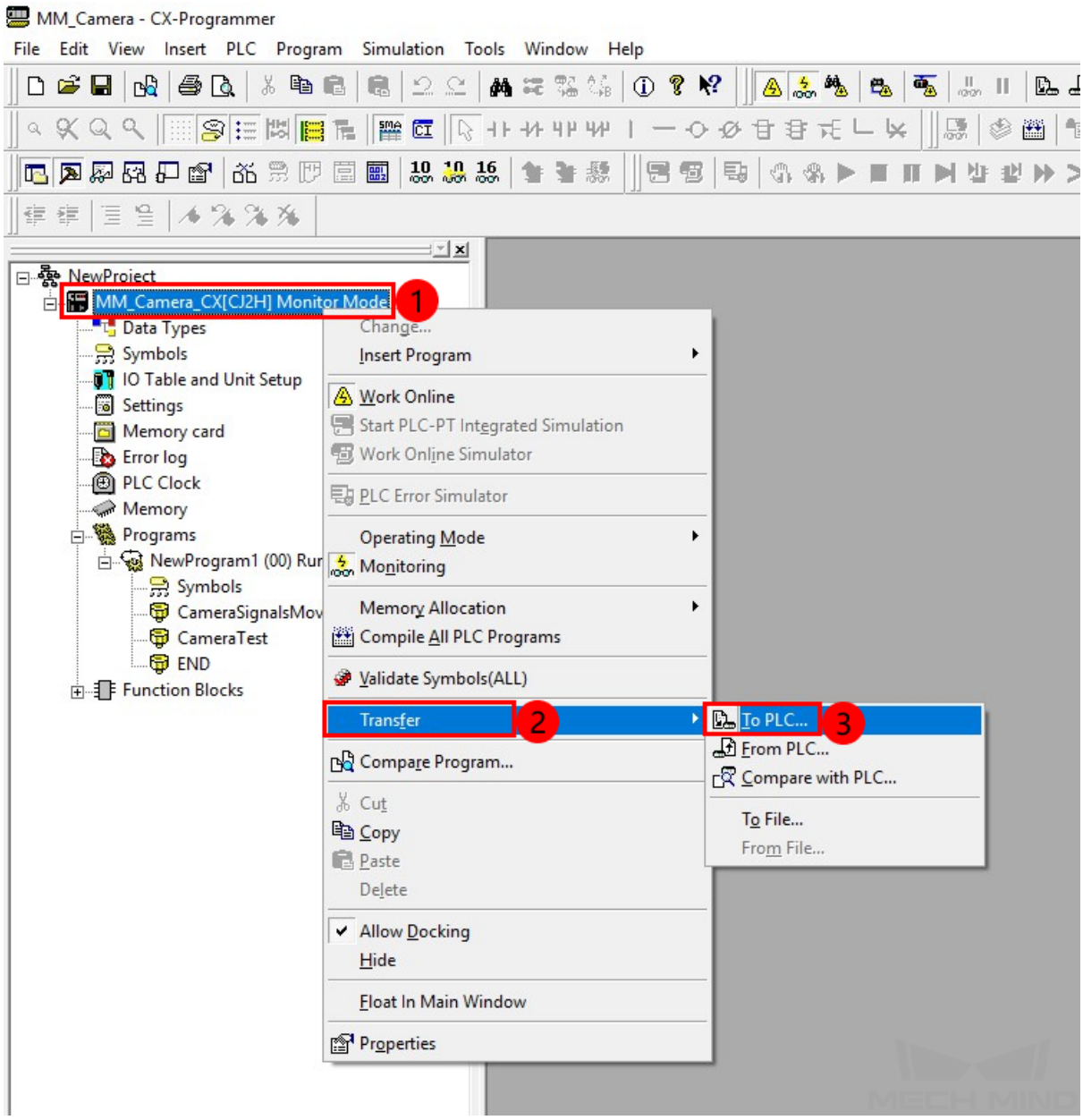


Download PLC Program

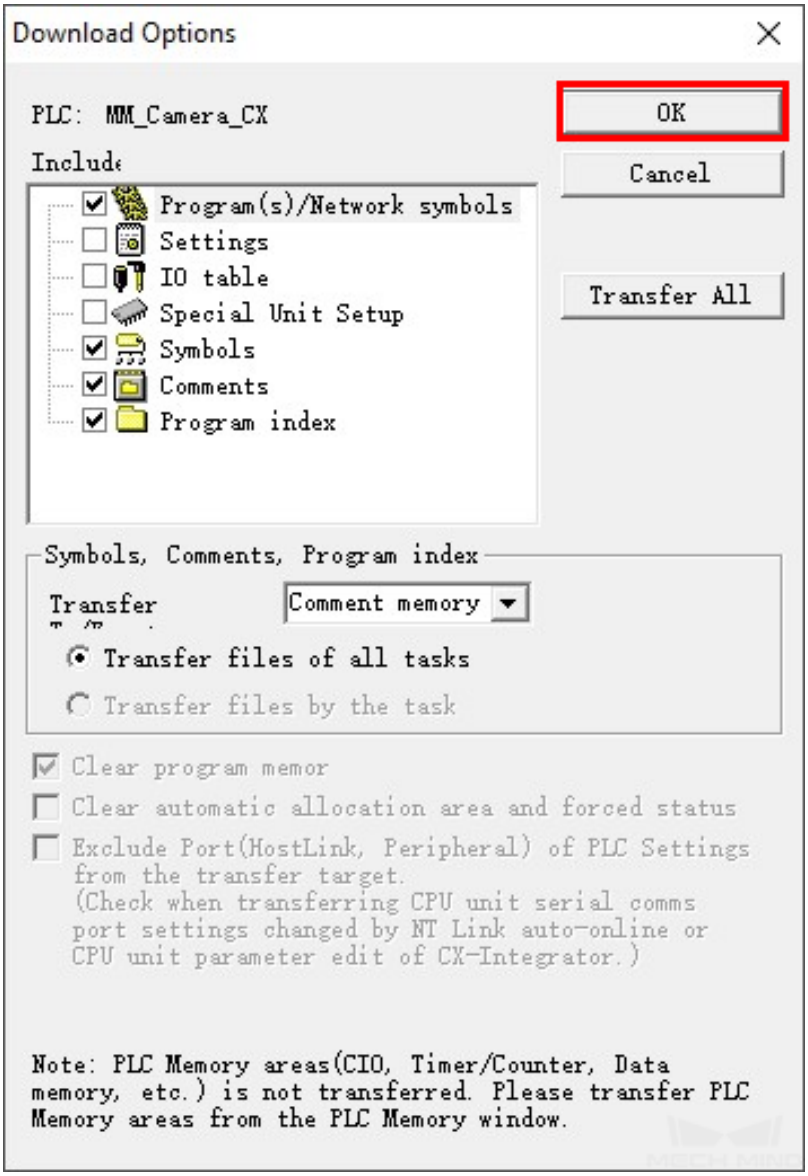
1. Right click on *MM_Camera_CX[CJ2H] Offline* and select **Work Online** in the context menu to switch the project to monitor mode.



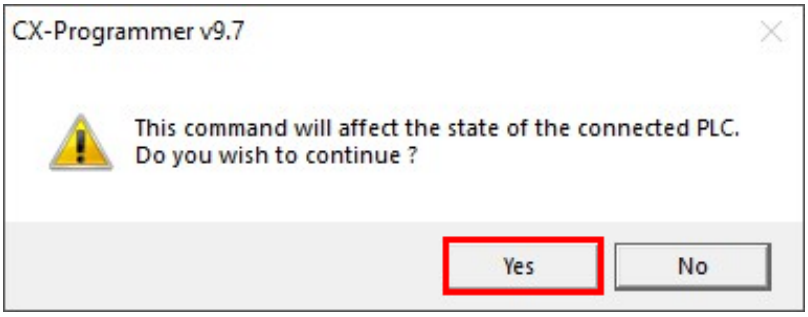
2. Right click on *MM_Camera_CX[CJ2H] Monitor Mode* and select *Transfer* → *To PLC*.

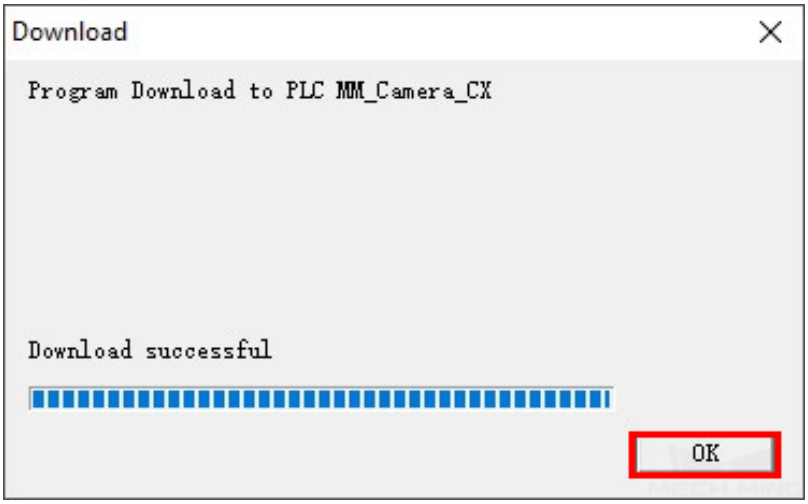


3. Click on *OK* in the **Download Options** window.



4. Click on *Yes* if the safety of connected devices can be ensured. Click on *OK* after dowloading the program successfully.



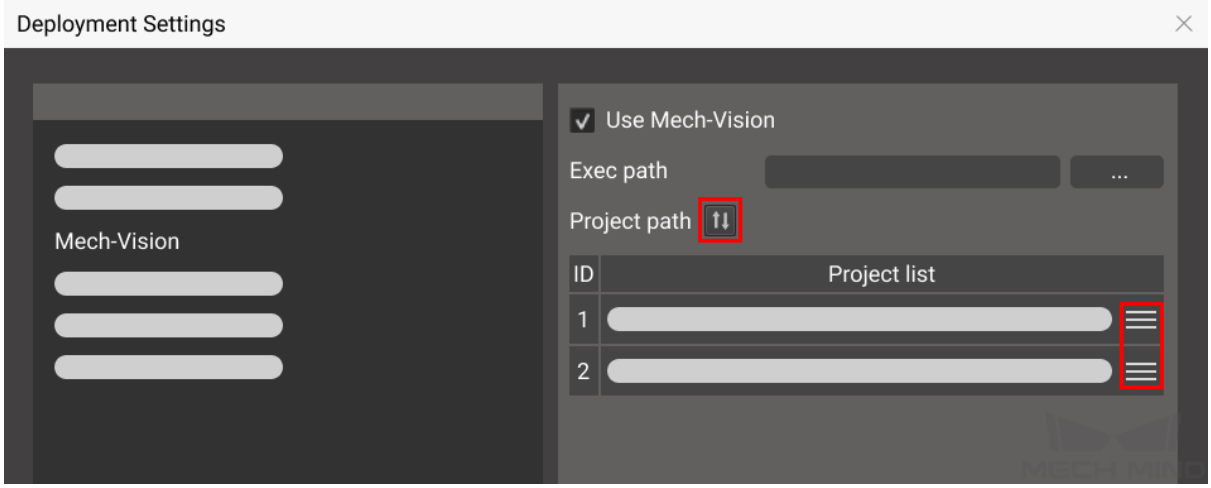


2.7.6 Test with Mech-Vision/Mech-Viz Project

This section introduces how to run the Mech-Vision/Mech-Viz project and obtain data from the project using the **ObtainPose** FB. For detailed information on the modules, please refer to standard_interface_development_profinet.

Prerequisites

- Mech-Vision project(s):
  - Executable
  - Set to autoload
  - The **Project list** in *Mech-Center* → *Deployment Settings* → *Mech-Vision* is synced by clicking on  , and the order of Mech-Vision projects have been adjusted according to actual needs.

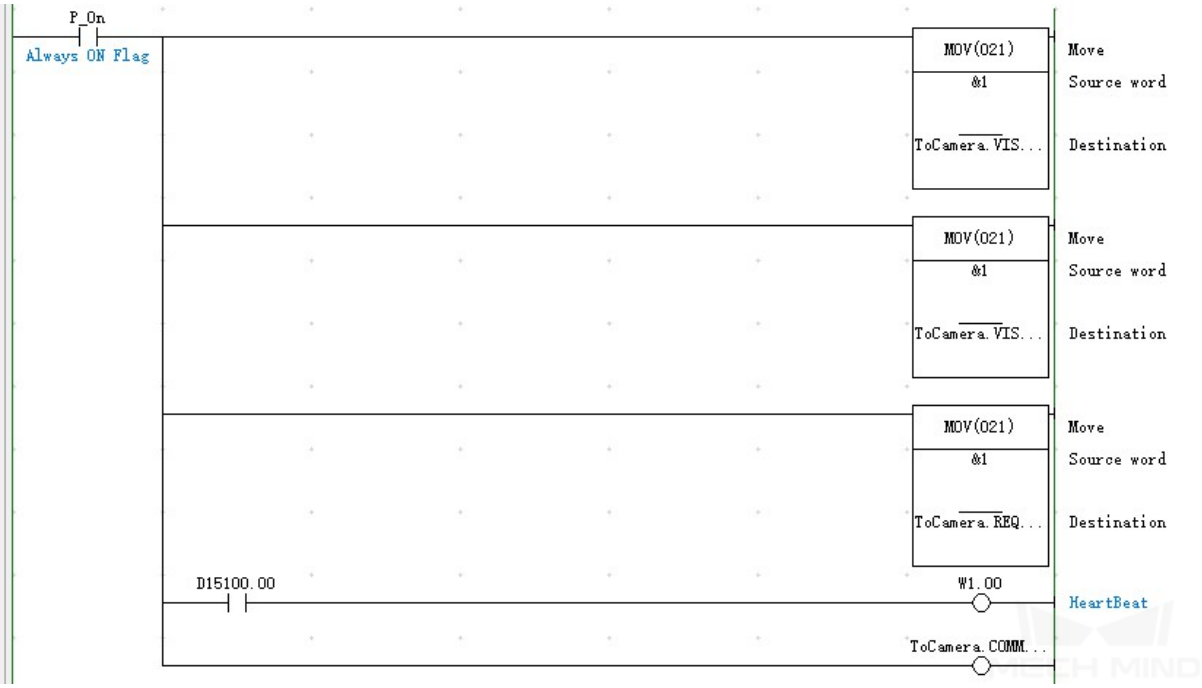


- Mech-Viz project:
  - Executable
  - Set to autoload
  - Contains a branch_by_service_message Task that has been renamed to **1**.

Run Mech-Vision Project and Obtain Vision Points

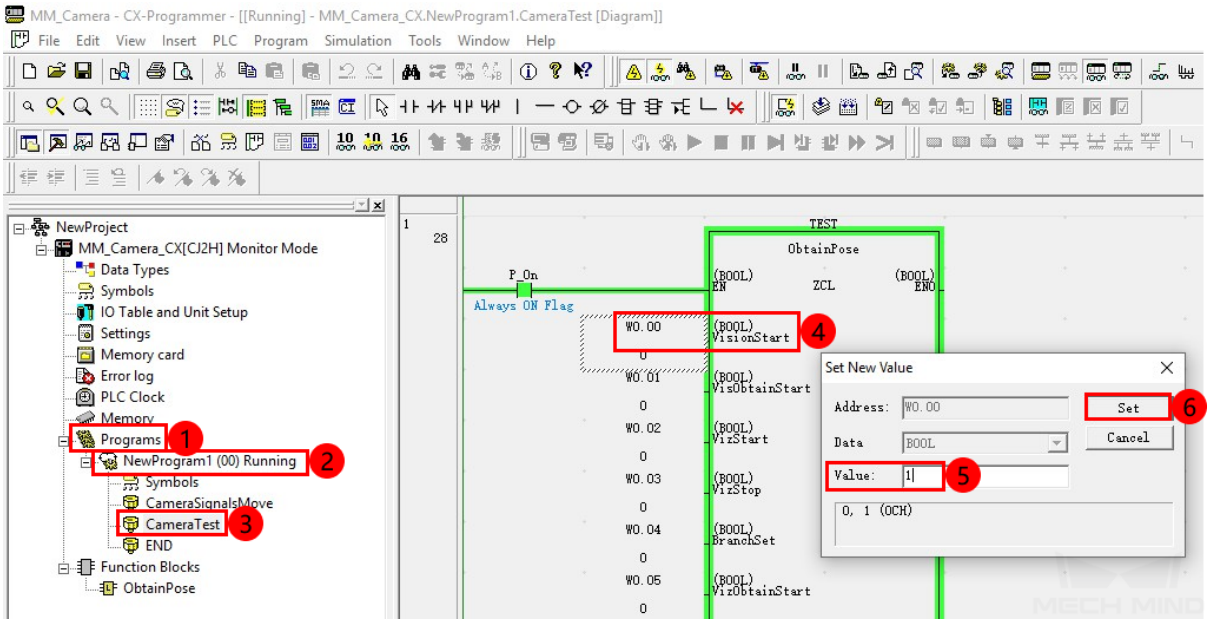
Parameter Settings

1. Set the **ToCamera.COM_ENABLE** to be **always ON**.
2. Double click on the **MOV** module, set the Mech-Vision project ID the same as the one set in **Deployment Settings** in Mech-Center. For example, if the monitor value is changed to **1**, then Mech-Vision project No. 1 in the **Project list** of Mech-Center will be started.
3. Set the number of vision points to be sent by Mech-Vision. The default value of **REQ_POSE_NUM** is 0, which means the Mech-Vision project will send all the vision points.

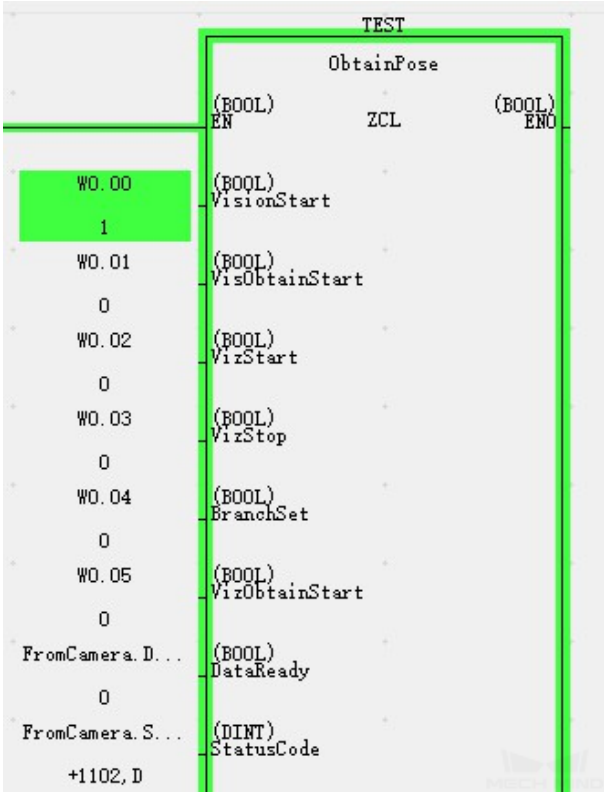


Start Mech-Vision Project

1. Open CX-Programmer, go to *Programs → NewProgram1(00) Running* and double click on **CameraTest**. Double click on the input port of **VisionStart** in the FB **ObtainPose**. Set the value to **1** in the **Set New Value** window and then click on *Set* to start Mech-Vision project and trigger camera to capture images. Then reset the value to **0**.

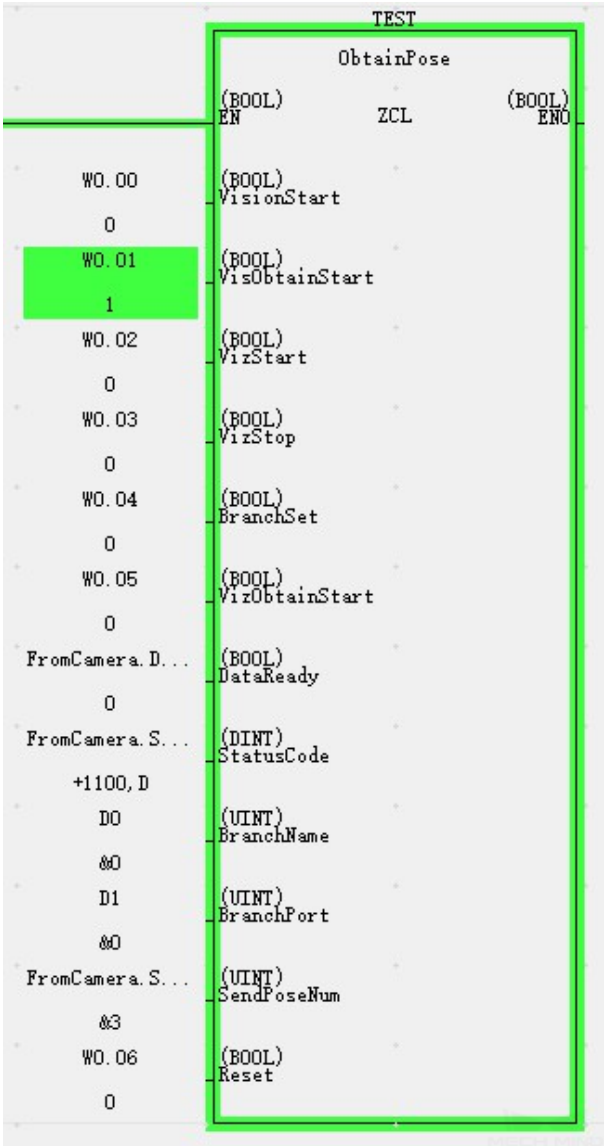


2. Check returned status code: check the monitor value of **StatusCode**. **1102** represents that the Mech-Vision project was started successfully. For other values, please refer to standard_interface_status_codes for the corresponding error.

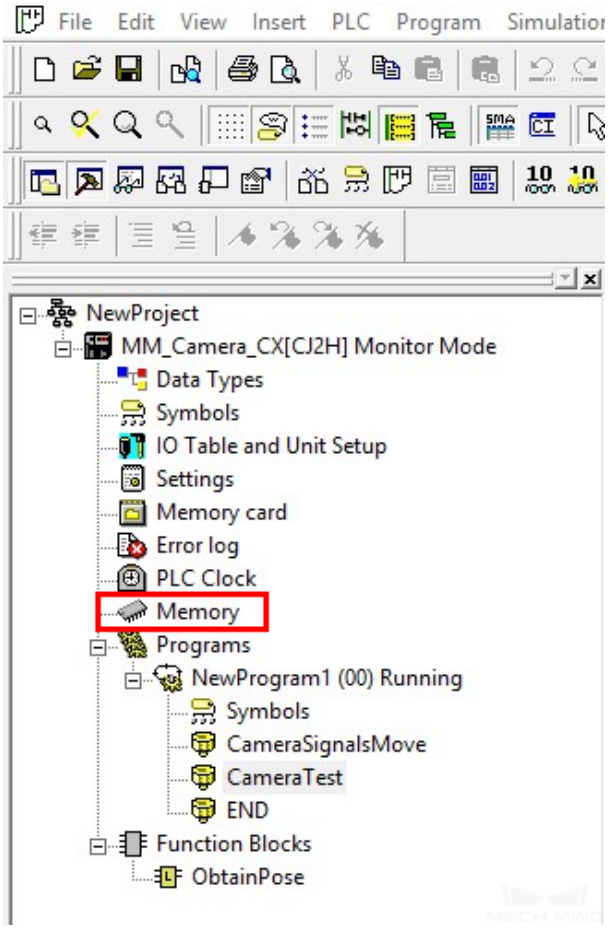


Obtain Vision Points from Mech-Vision

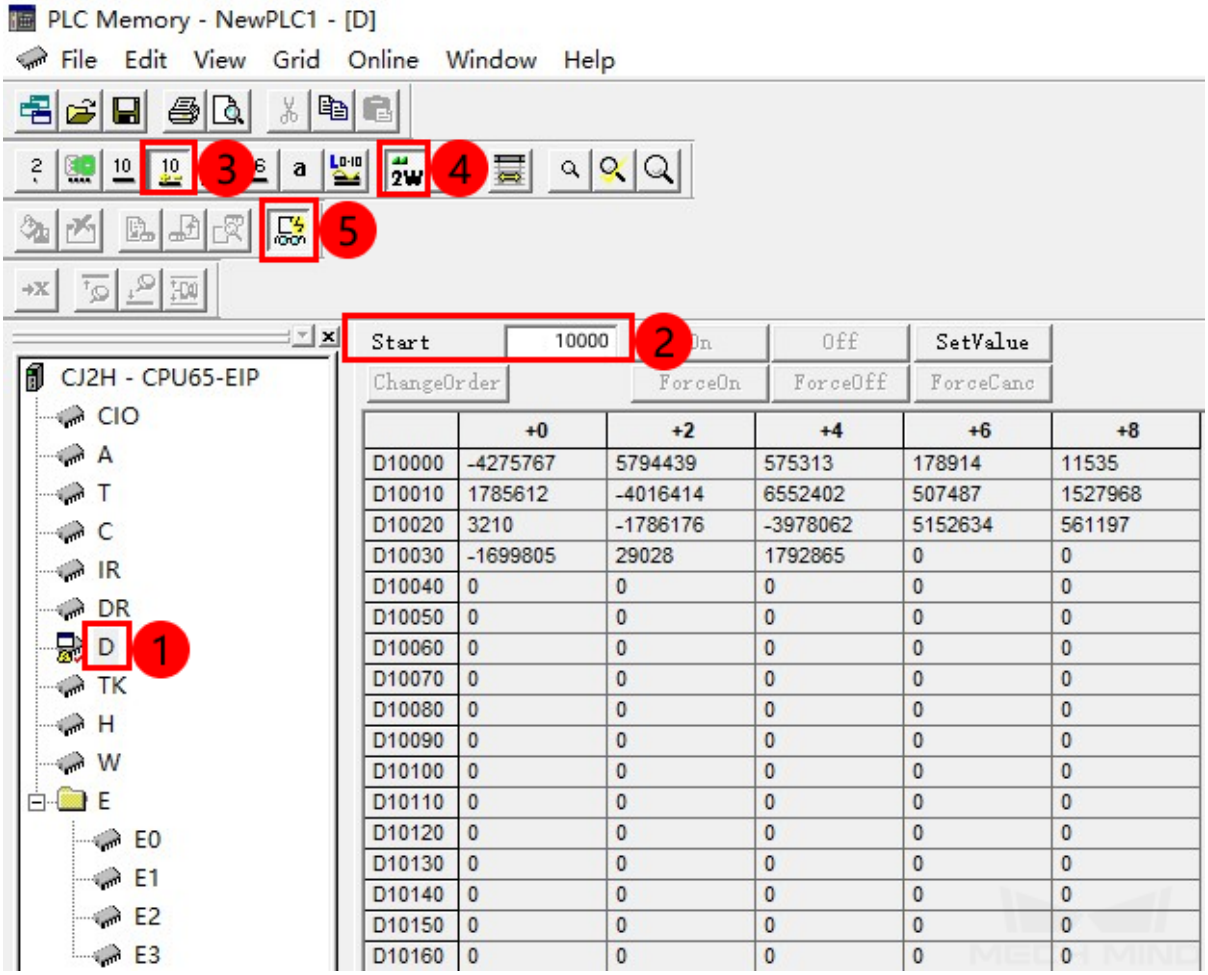
1. After the status code **1102** is returned, double click on the input port of **VisObtainStart** in the FB **ObtainPose**, and set the value to **1** in the **Set New Value** window and then click on *Set* to obtain vision points from Mech-Vision. Then reset the value to **0**. The result is shown as below. The value of SendPoseNum is 3, which means 3 vision points are obtained from Mech-Vision.



2. Double click on **Memory** in the project workspace, and a **PLC Memory** window will appear.



3. Double click on **D**, set the **Start** to **10000**, click on  ,  to set the data format as signed decimal and double word, and then click on  to start monitoring.

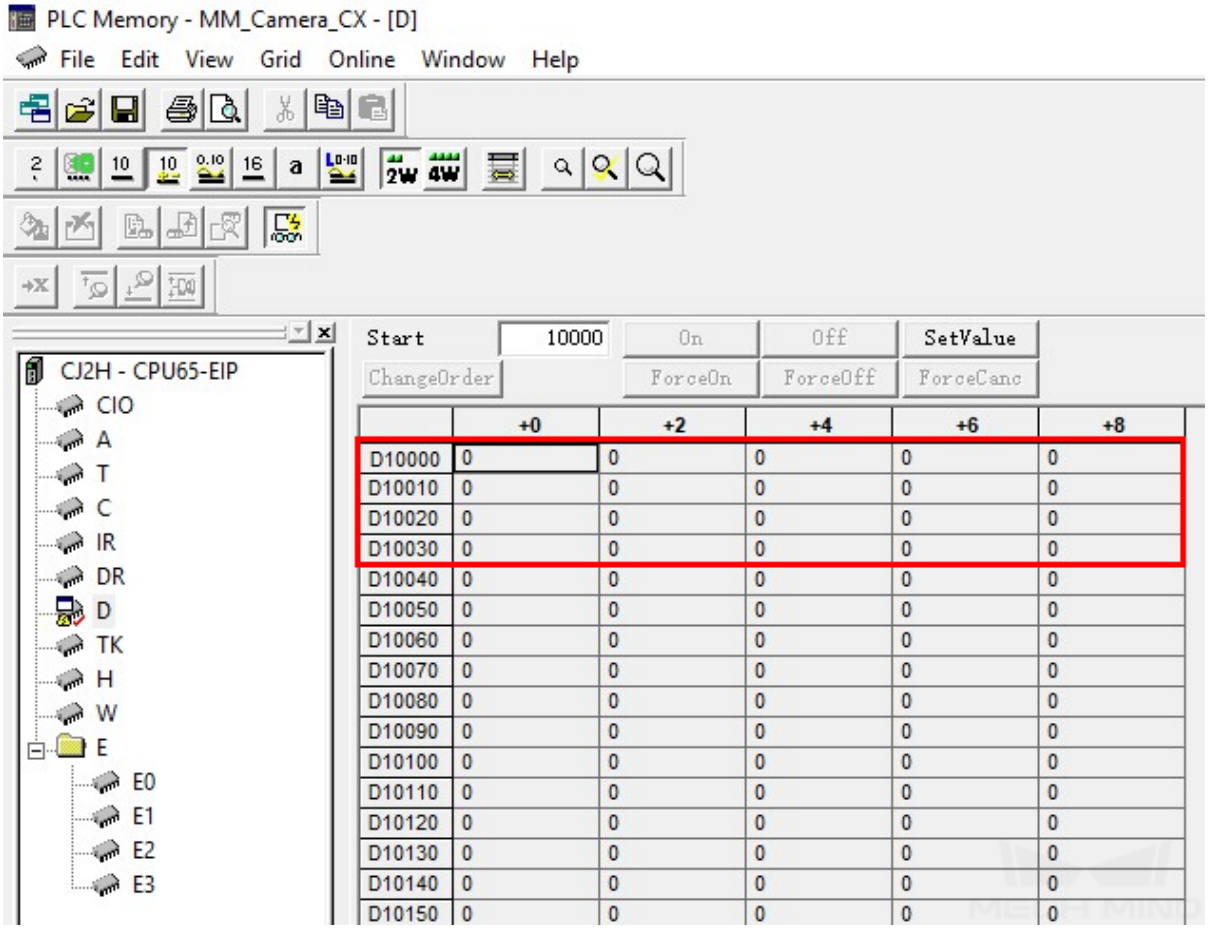


Hint: This example received 3 poses. Divide the transferred values by 10000 to obtain the actual pose data.

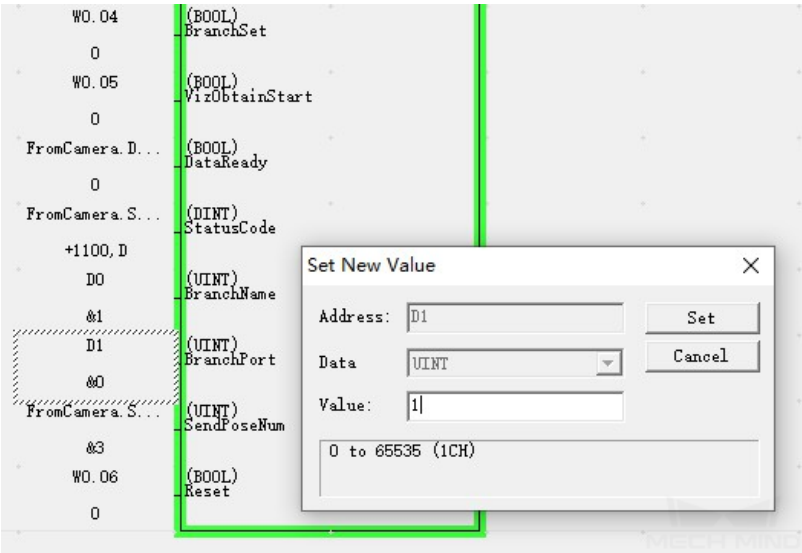
Run Mech-Viz Project and Obtain Planned Path

Parameter Settings

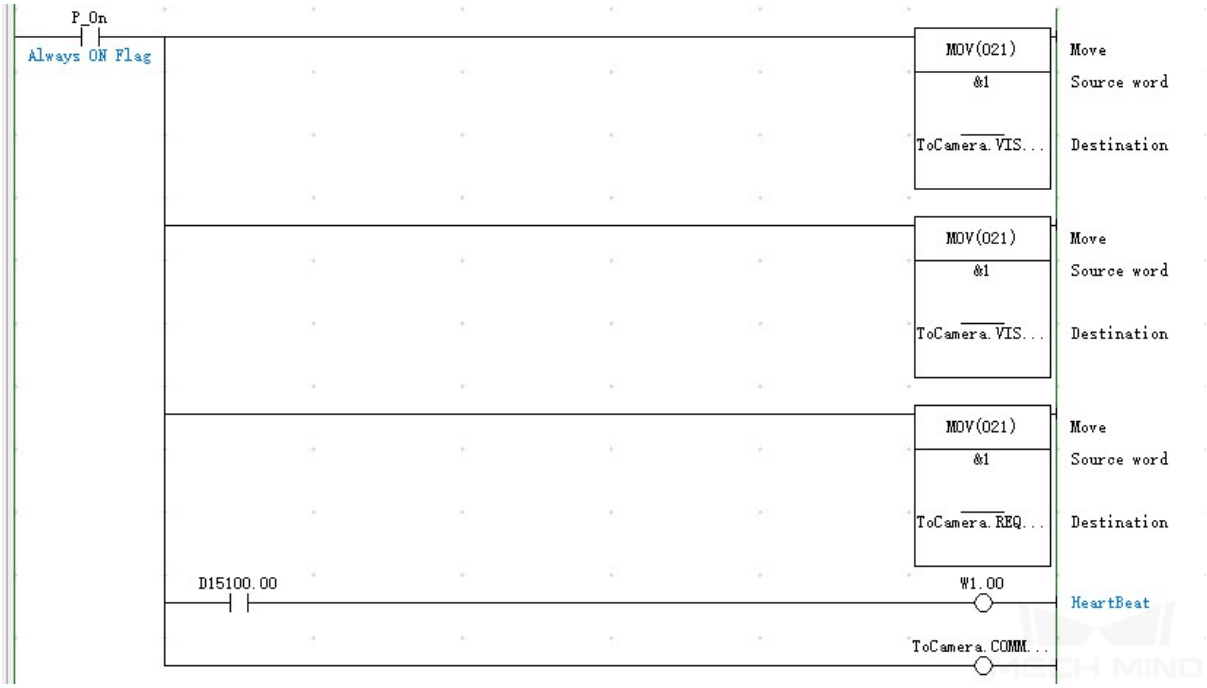
1. Open CX-Programmer, double click on input port of **Reset** in the **ObtainPose** FB, set the value to **1** in the **Set New Value** window and then click on *Set* to clear the previously obtained vision data. Then reset the value to **0**.



2. Modify the value of D0 register, set the value of **BranchName** to **1**.
3. Modify the value of D1 register, set the value of **BranchPort** to **1** , the Mech-Viz project will proceed along out port 1 of Task 1.

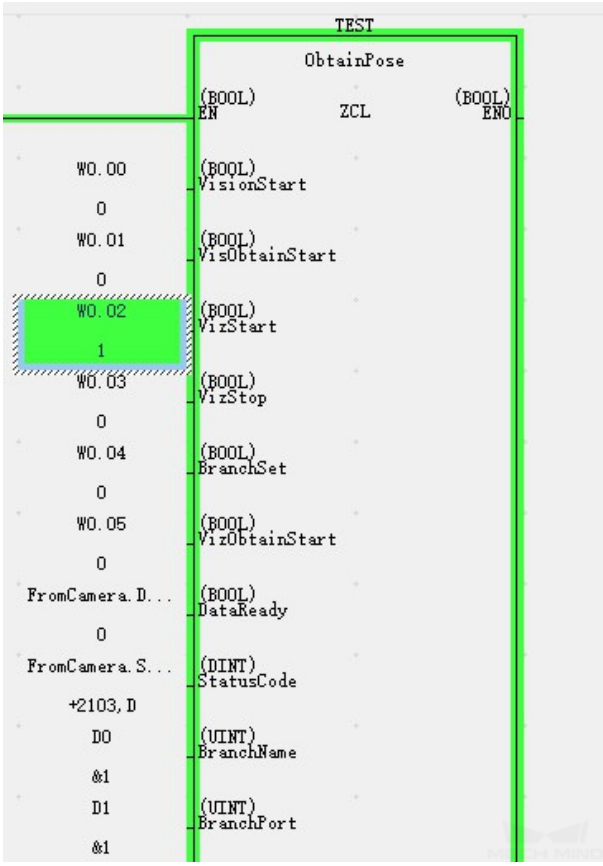


4. Set the value of REQ_POSE_TYPE to **1**. This asks Mech-Viz to send joint positions (instead of TCP data).



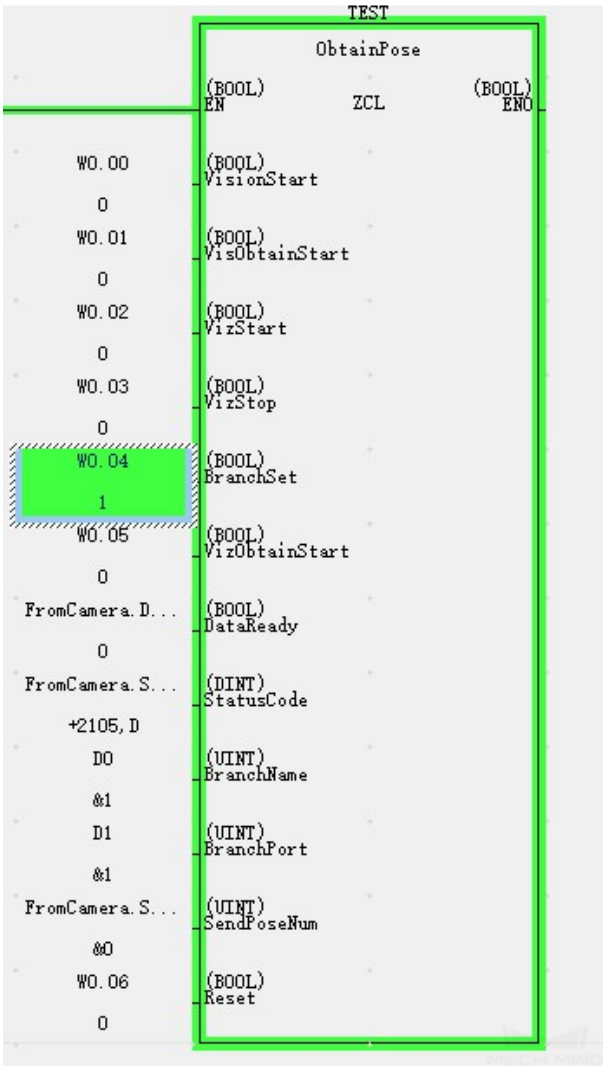
Start Mech-Viz Project

1. Double click on the input port of **VizStart** in the **ObtainPose** FB, set the value to **1** in the **Set New Value** window to start Mech-Viz project and then reset it to **0**.
2. If the value returned by the variable **StatusCode** is **2103**, it represents that the Mech-Viz project was started successfully. For other values, please refer to standard_interface_status_codes for the corresponding error.



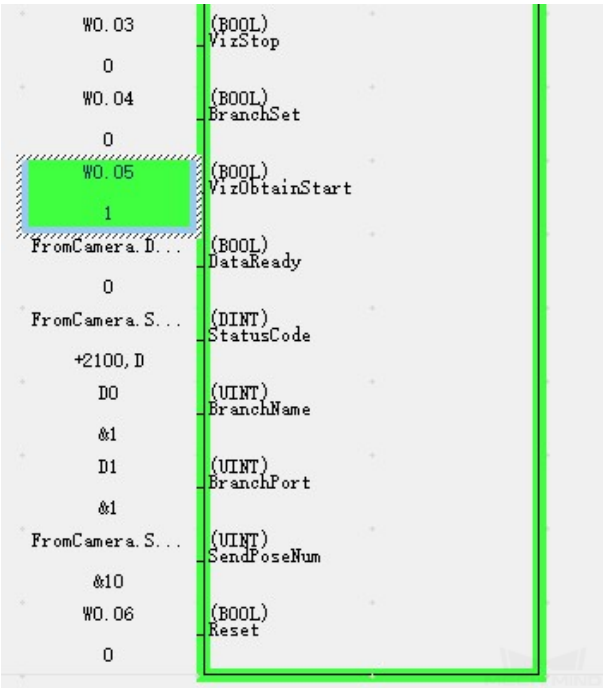
Select Branch in the Mech-Viz Project

1. Double click on the input port of **BranchSet** in the **ObtainPose** FB, set the value to **1** in the **Set New Value** window to select branch in the Mech-Viz project and then reset it to **0**.
2. If the value returned by the variable **StatusCode** is **2105**, it represents that the branch was selected successfully. For other values, please refer to standard_interface_status_codes for the corresponding error.

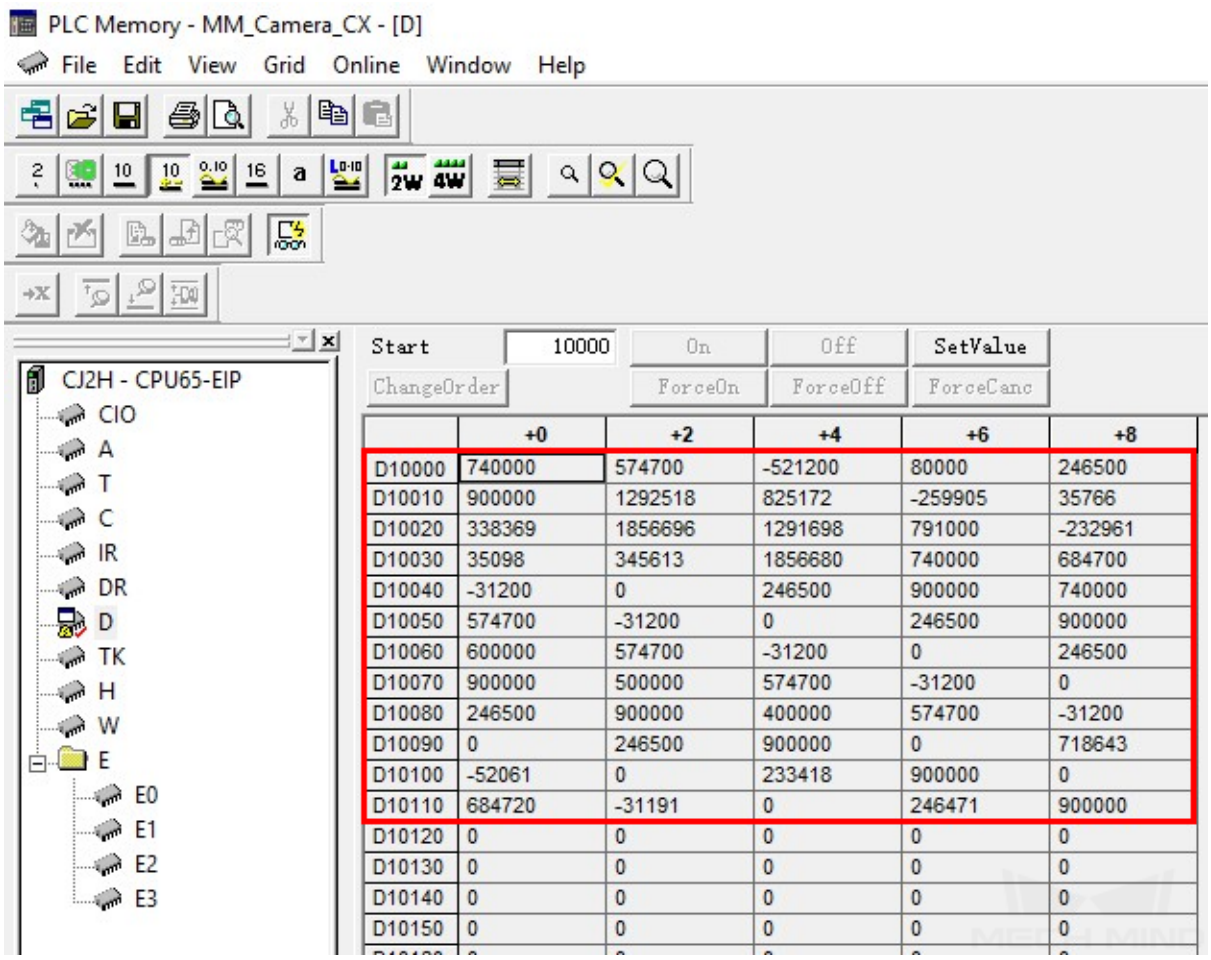


Obtain Planned Path

1. Double click on the input port of **VizObtainStart** in the **ObtainPose** FB, set the value to **1** in the **Set New Value** window to obtain planned path from Mech-Viz project and then reset it to **0**.
2. If the value returned by the variable **StatusCode** is **2100**, it represents that planned path was obtained successfully. For other values, please refer to standard_interface_status_codes for the corresponding error. The value of **SendPoseNum** shows how many target points were received, and the target points are stored in **TargetPose**.



3. Go back to **PLC Memory** window, the 10 poses are shown as below. Please divide the transferred values by 10000 to obtain the actual pose data.



2.8 PROFINET - Siemens SIMATIC S7 PLC

This section provides information on setting up communication between a Siemens SIMATIC S7 PLC and Mech-Mind Software Suite via PROFINET.

2.8.1 Overview

- *Hardware and Software Requirements*
- *Configure IPC and Initiate Communication*
- *Install GSD file and Configure Communication*
- *Import Example Program and Download to PLC*
- *Test with Mech-Vision/Mech-Viz Project*

2.8.2 Hardware and Software Requirements

Hardware

- Siemens SIMATIC S7 PLC:
  - S7-300 (with PROFINET interface or CP 343-1 integrated to function as a PROFINET IO controller)
  - S7-400 (with PROFINET interface or CP 443-1 integrated to function as a PROFINET IO controller)
  - S7-1200
  - S7-1500
- AC 220 V to DC 24 V power adapter
- HMS IXXAT INpact 40 interface card installed on the IPC in Mech-Mind Vision System
- Switch
- Ethernet cables

Software

- Siemens TIA Portal V15.1
- Mech-Mind Software Suite: Mech-Center 1.5.1 or above, Mech-Vision 1.5.0 or above, and Mech-Viz 1.5.0 or above.
- VCI V4 (driver software for HMS IXXAT INpact 40 interface card)
- Mech-Mind GSD file:
  - File name: **GSDML-V2.35-Mech-Mind Robotics Technologies Ltd-MechMind-PIR-20220315.xml** (The version number and date in the file name may differ.)
  - File location: *Mech-Mind/Mech-Center/mech_interface/PROFINET* (the installation directory of Mech-Center)

  Note: Copy this file to the computer with Siemens TIA Portal installed.

- Example programs:
  - File name: **PLC sample.zip**
  - File location: *Mech-Mind/Mech-Center/mech_interface/documents/CN/Siemens PROFINET 编程指南* (the installation directory of Mech-Center)

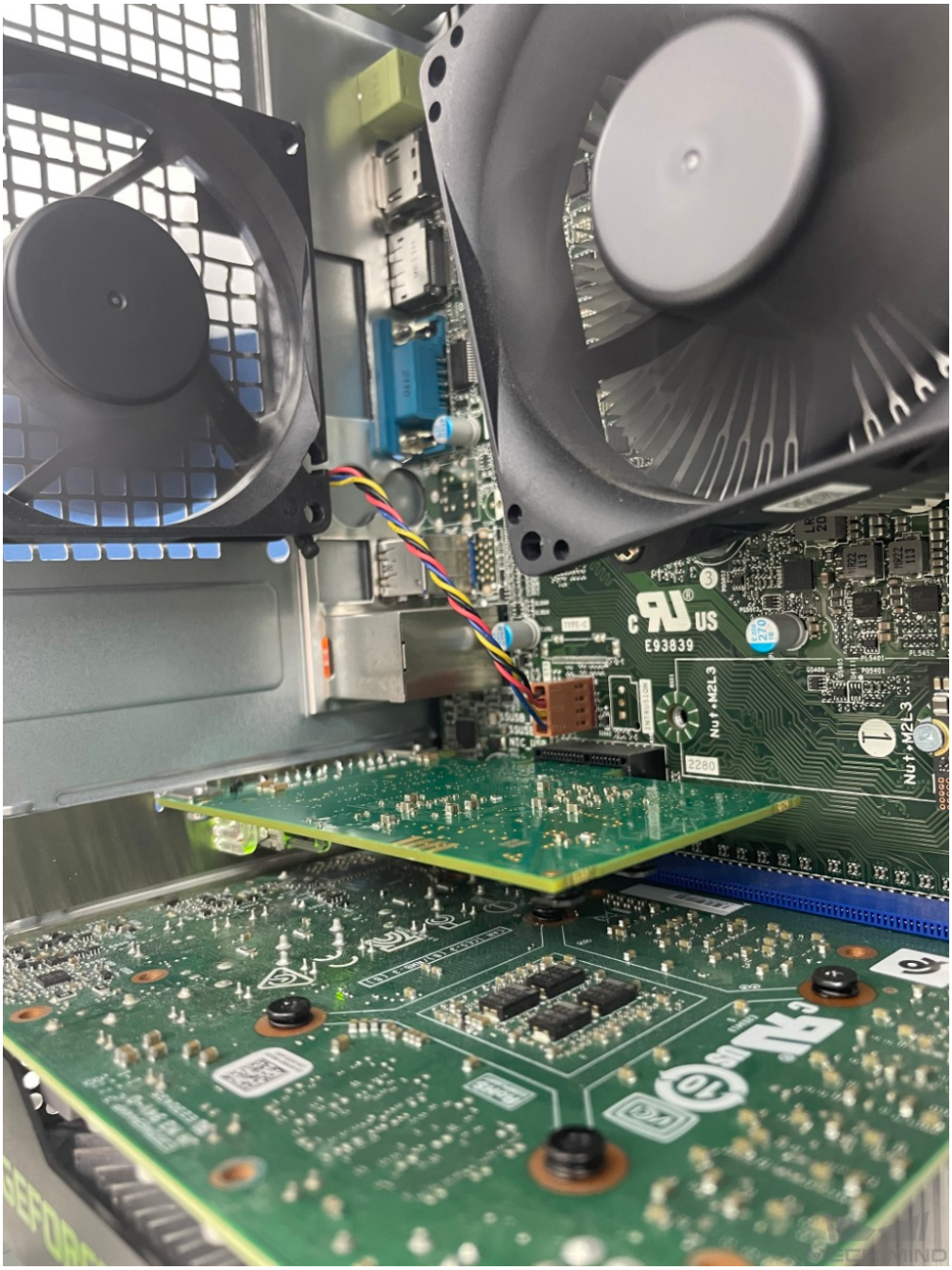
  Note: Copy this ZIP file to the computer with Siemens TIA Portal installed, and unzip it to get the following files: **Camera_IO.scl**, **ObtainPose.scl**, **PLCTags.xlsx**.

Note: Connect the Mech-Mind Vision System IPC, computer with Siemens TIA Portal installed, and PLC to the same router.

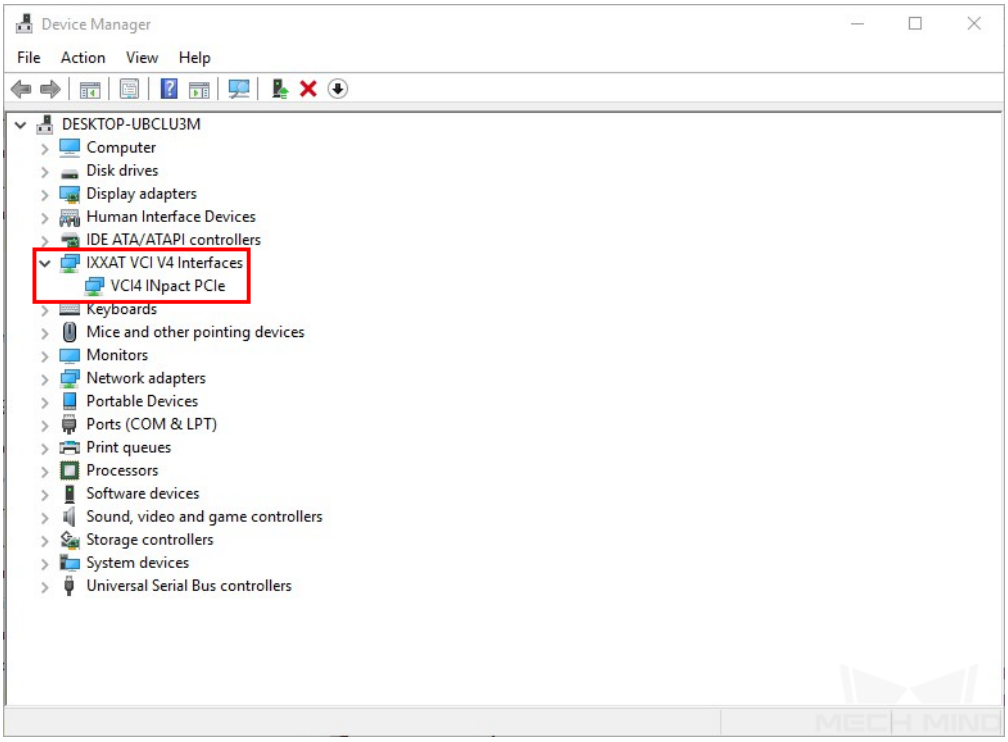
2.8.3 Configure IPC and Initiate Communication

Check PCI-e Card and Driver Software

1. Check the PCI-e slot on the IPC and make sure the HMS IXXAT INpact 40 interface card is installed.

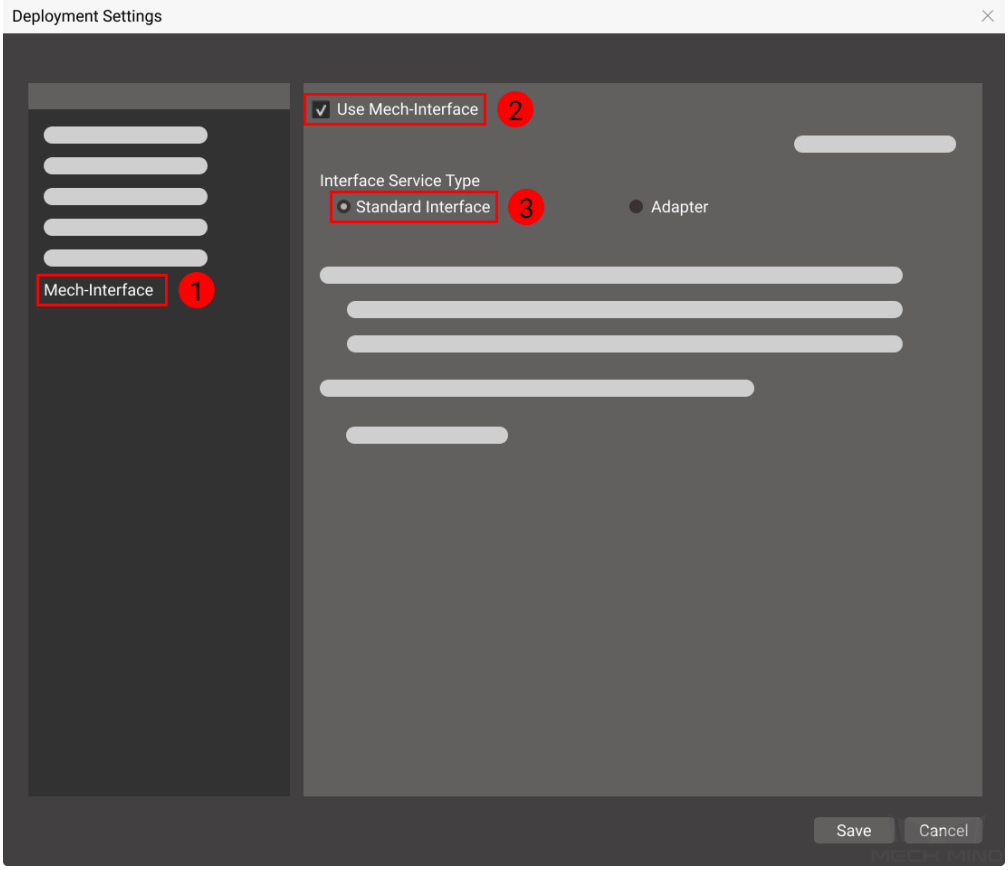


2. Make sure the driver software is installed on the IPC: in **Device Manager**, find **VCI4 INpact PCIe** under **IXXAT VCI V4 Interfaces**.

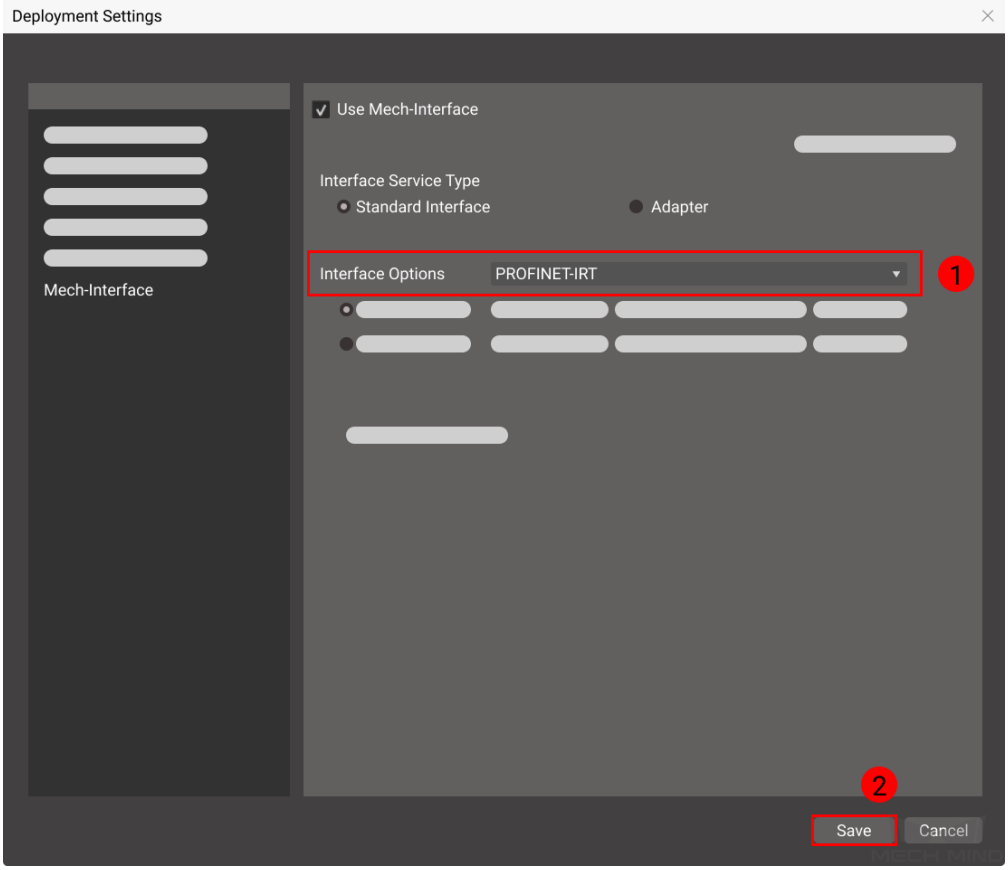


Configure Mech-Interface in Mech-Center

1. Open Mech-Center, and click on *Deployment Settings*.
2. Go to **Mech-Interface**, check **Use Mech-Interface** and select **Standard Interface**.



3. In **Interface Options**, select **PROFINET-IRT**, and then click on *Save*.



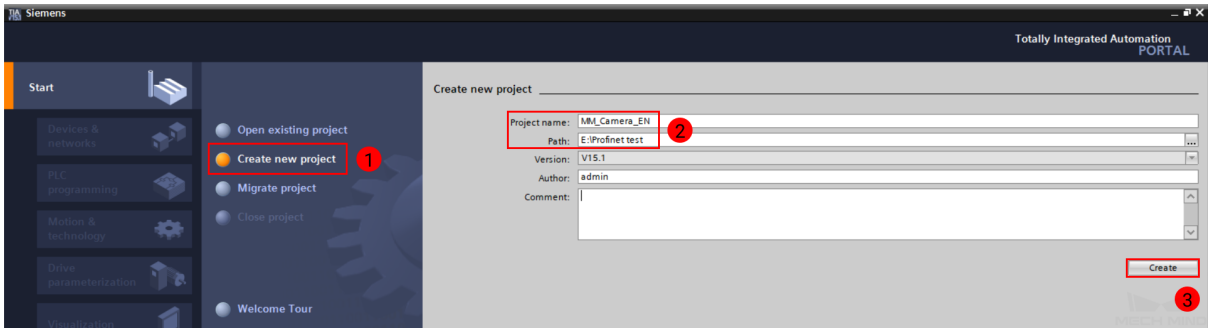
4. Click on *Start Interface* in the Toolbar.

2.8.4 Install GSD file and Configure Communication

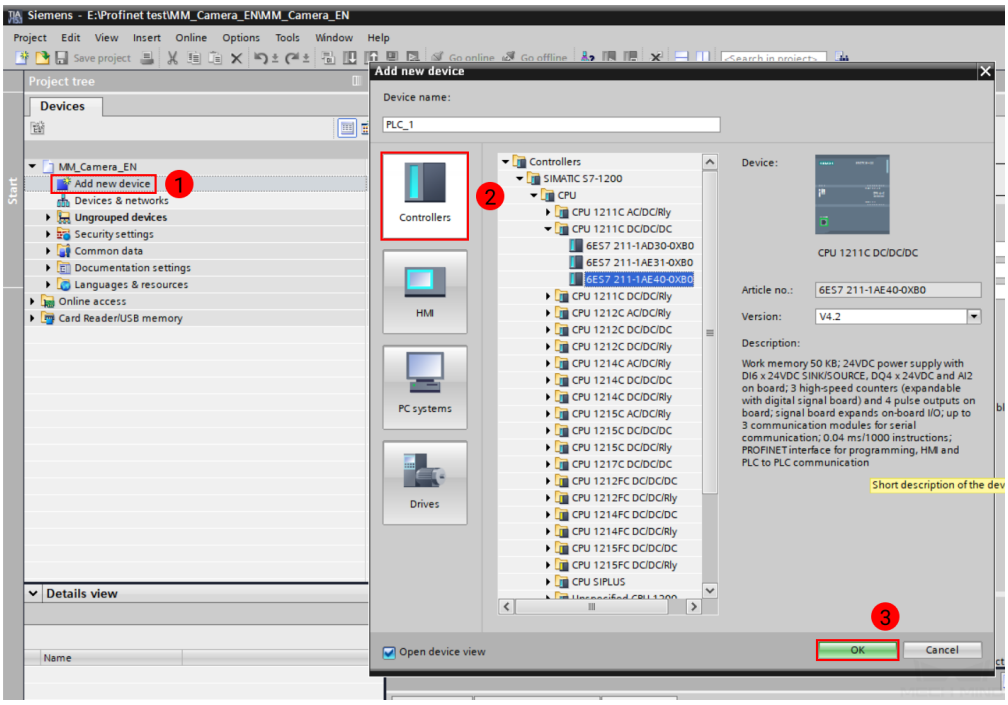
Create PLC Project and Set IP Address

Hint: If needed, click on *Save project* to save the changes to the project.

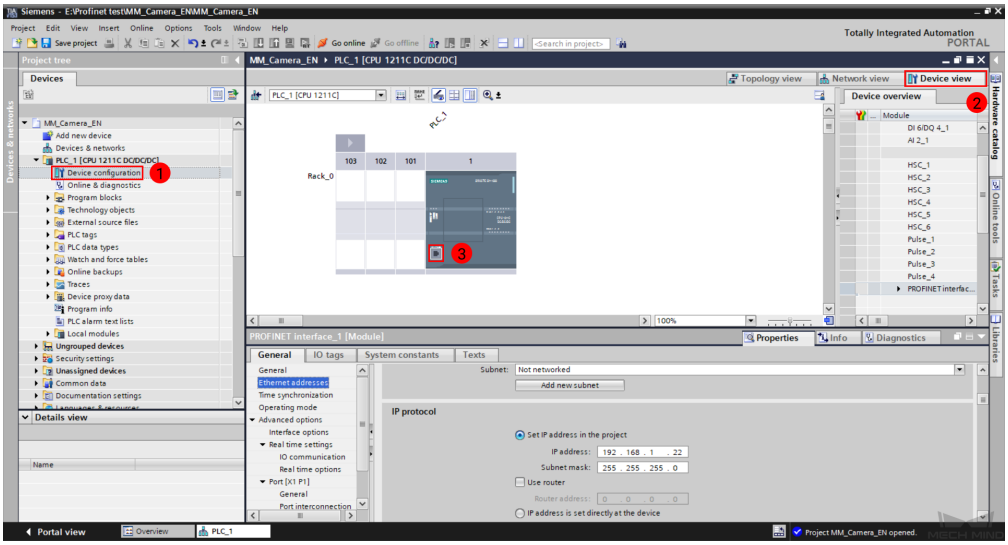
1. Open TIA Portal and click on **Create new project**.
2. Put in **Project name** and **Path**, and then click on *Create*. Click on **Open the project view** in the pop-up page.



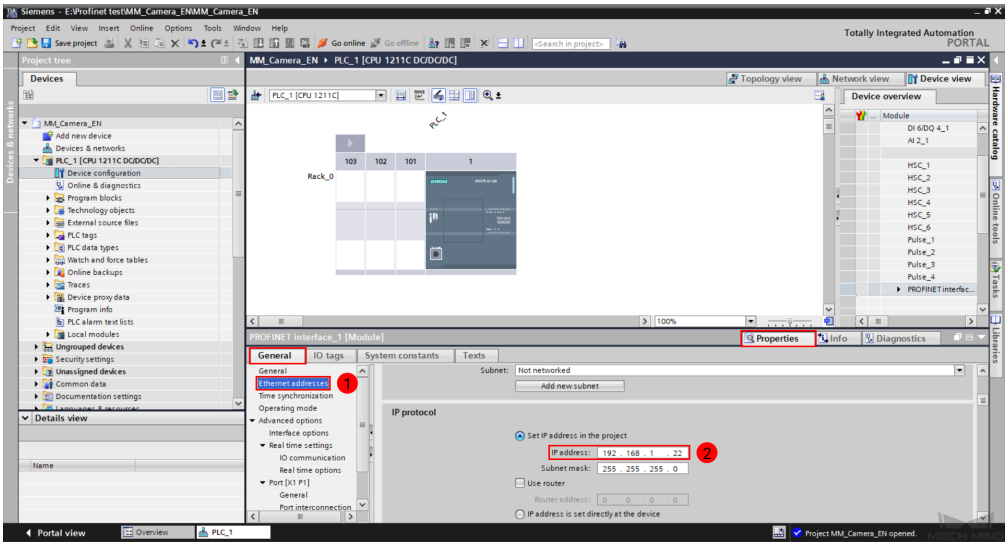
3. In the **Project tree** panel, double-click on **Add new device**. In the new window, click on *Controller*, and find the CPU module you are using, and name it in **Device name**. Click on *OK* to confirm adding the device. Here, the device is named **PLC_1**.



4. In **Device view**, click on  is **PLC_1**. If you don' t see the following interface, first click on **Device configuration** under **PLC_1** in the **Project tree** panel, and then click on **Device view** tab on the right.

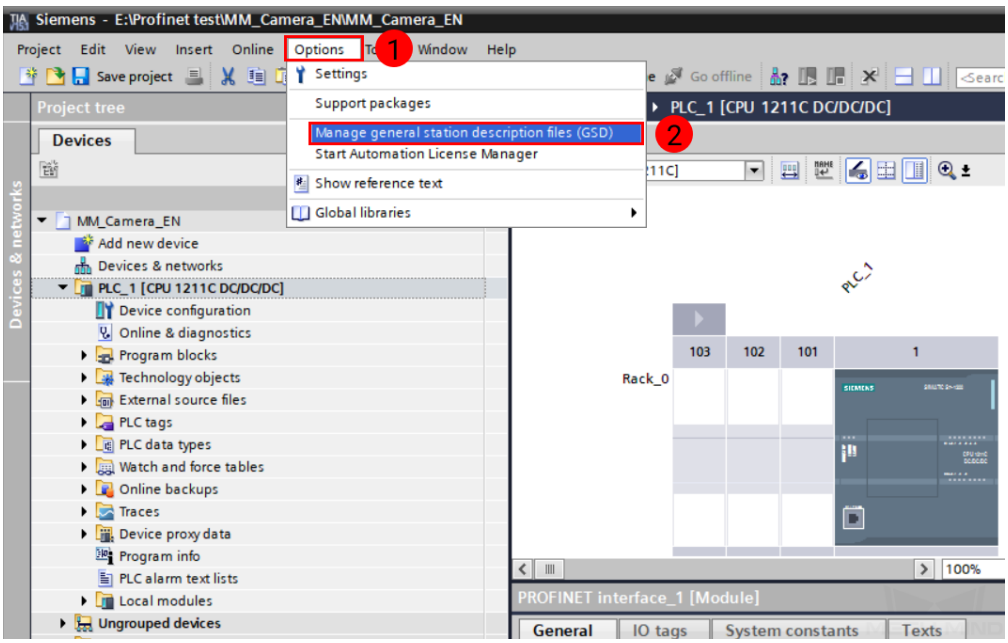


5. Down in the *Properties* → *General* tab, click on **Ethernet address** to set the IP address of the PLC.

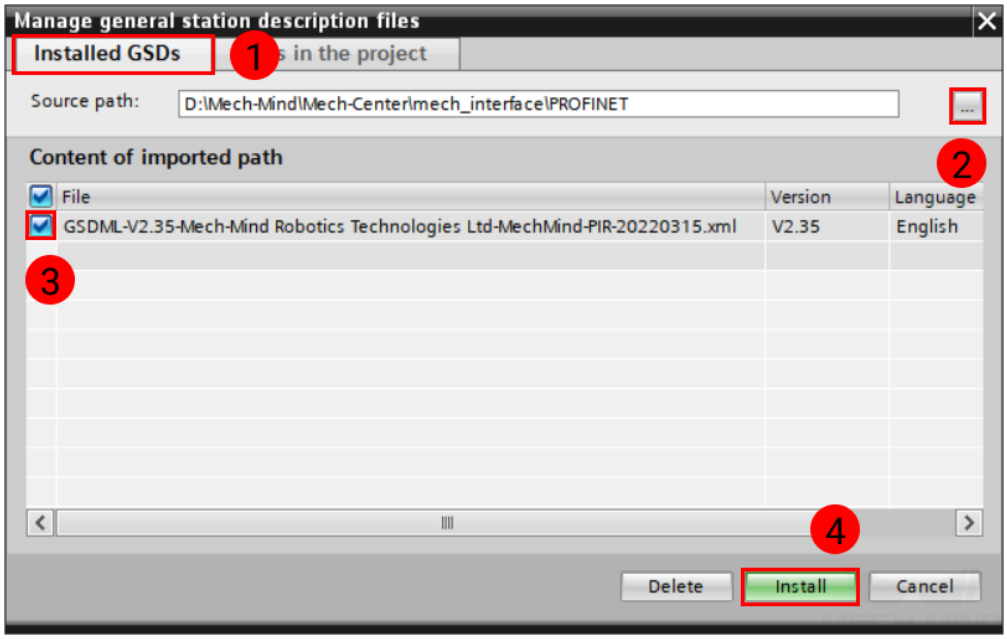


Install GSD File and Configure Network

1. In the menu bar, select *Options* → *Manage general station description files (GSD)*.

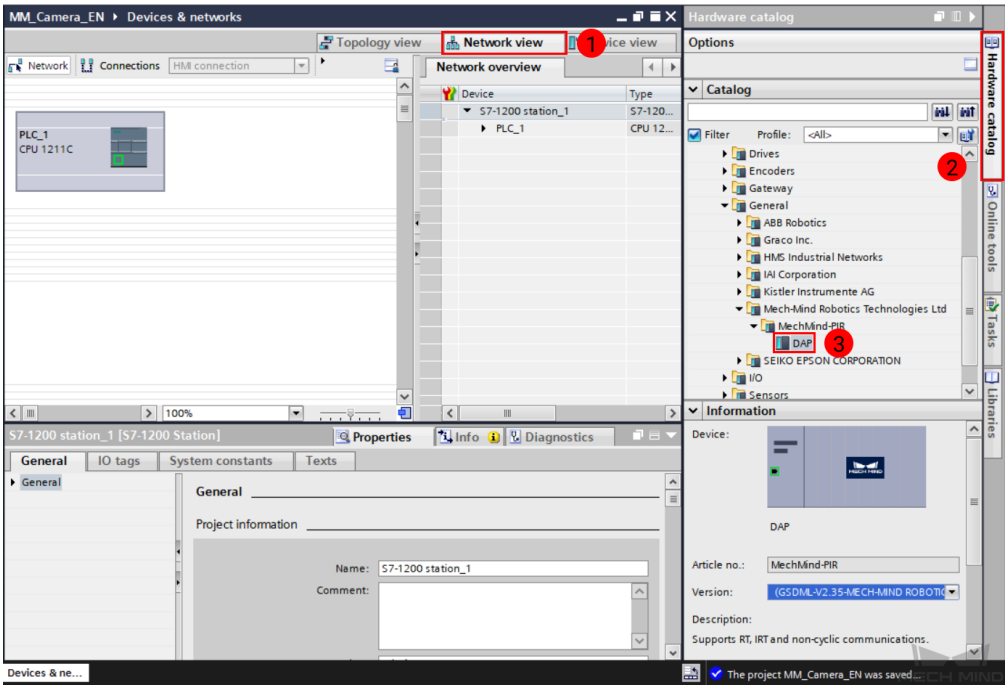


2. In the pop-up window under **Installed GSDs** tab, click on ··· to the right of **Source path**. Locate the path where the Mech-Mind GSD file is stored. Check this file in **Content of imported path**, and then click on *Install*. Close the window after the installation completes.

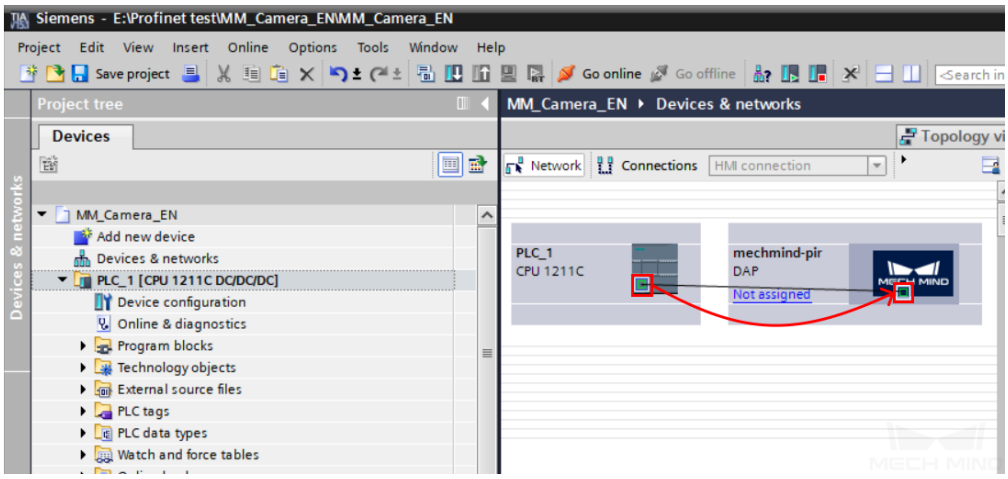


3. Return to **Device configuration**, and go to **Network view** tab. Click on **Hardware catalog**,

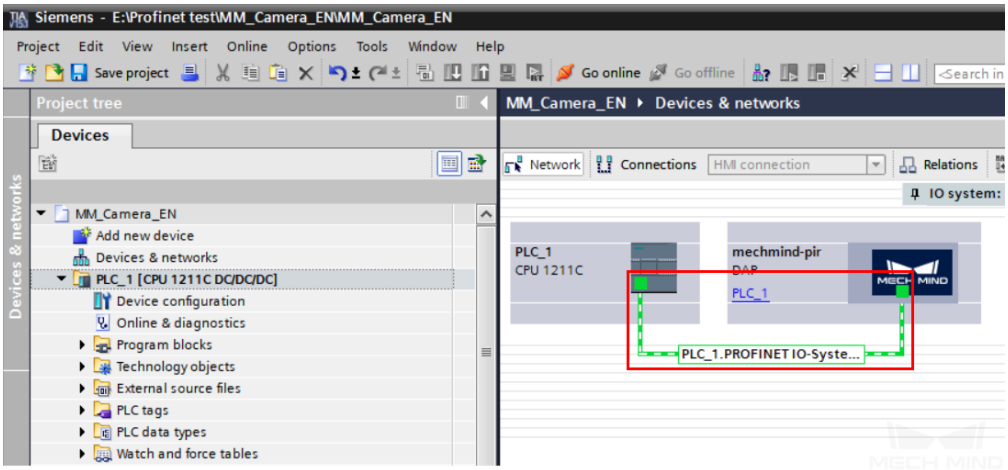
and double-click on **DAP** under *Other field devices/PROFINET IO/General/Mech-Mind Robotics Technologies Ltd/MechMind-PIR* to display **mechmind-pir** in **Network view**.



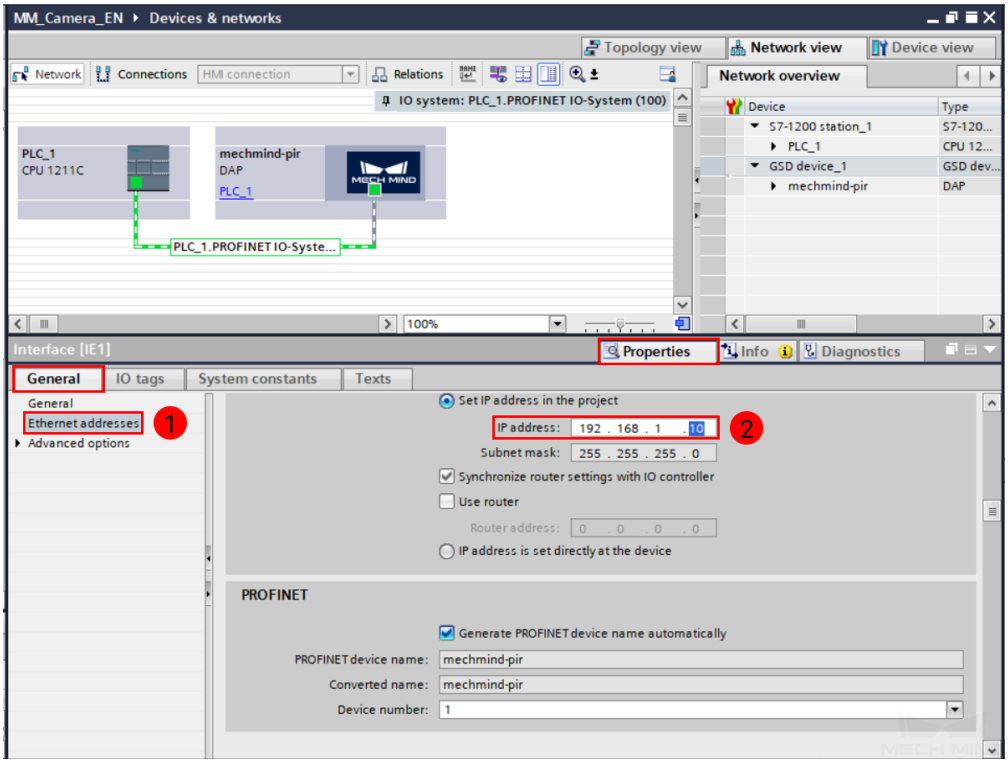
4. In **Network view**, click on  in **PLC_1** and drag it to  in **mechmind-pir**, and release the button when a black connection line appears.



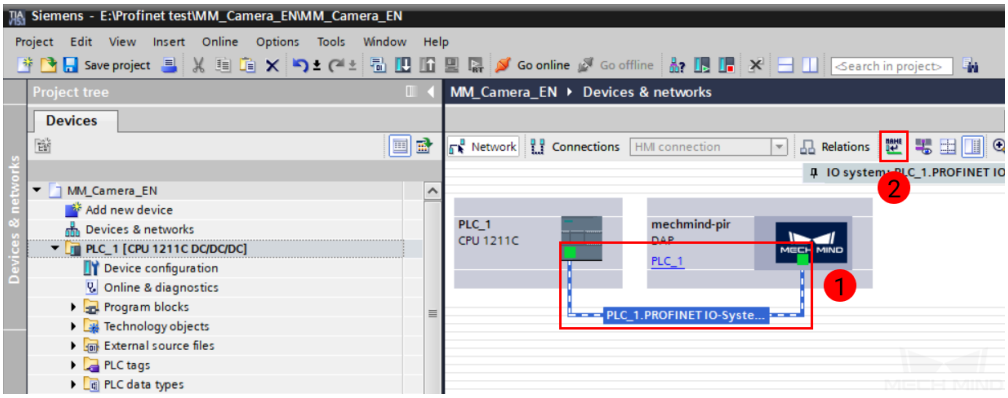
A successful connection should look like this:



5. Click on  in **mechmind-pir**, down in the *Properties* → *General* tab, click on **Ethernet address** to set the IP address of the IPC. This IP address should be in the same subnet as that of the PLC. Make sure **Generate PROFINET device name automatically** is checked.

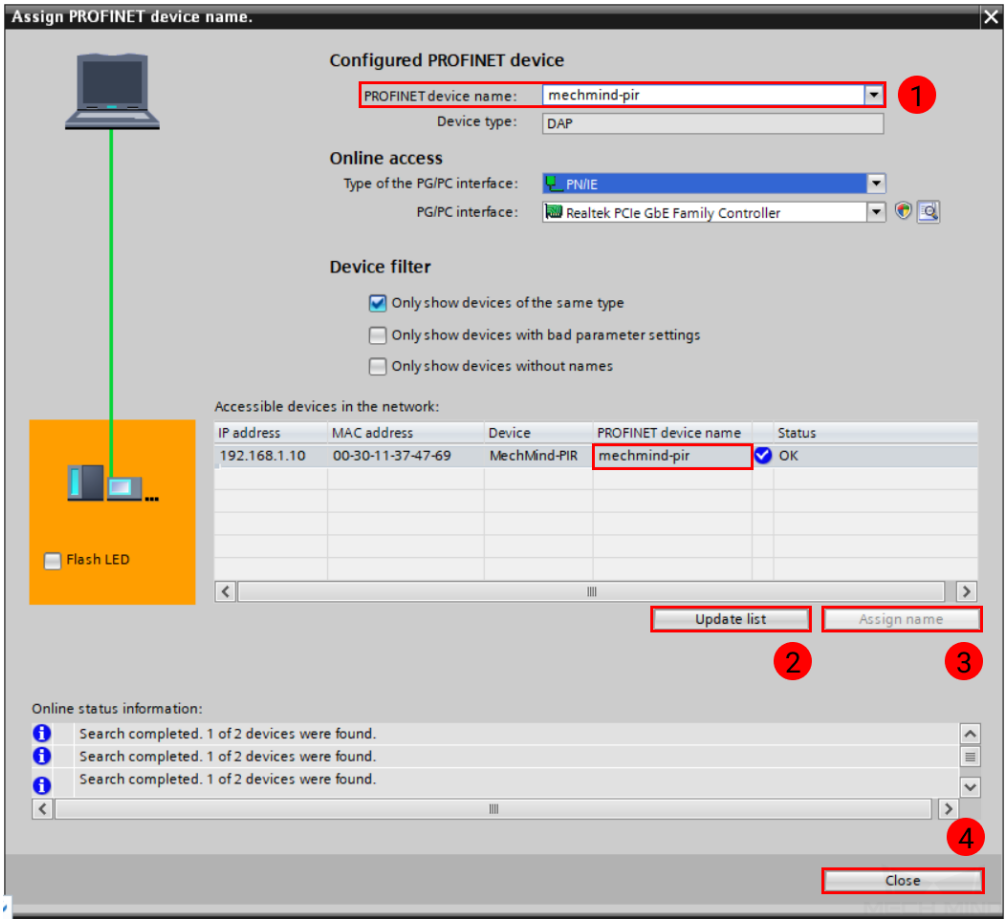


6. Click on the green connection line, and then click on  .

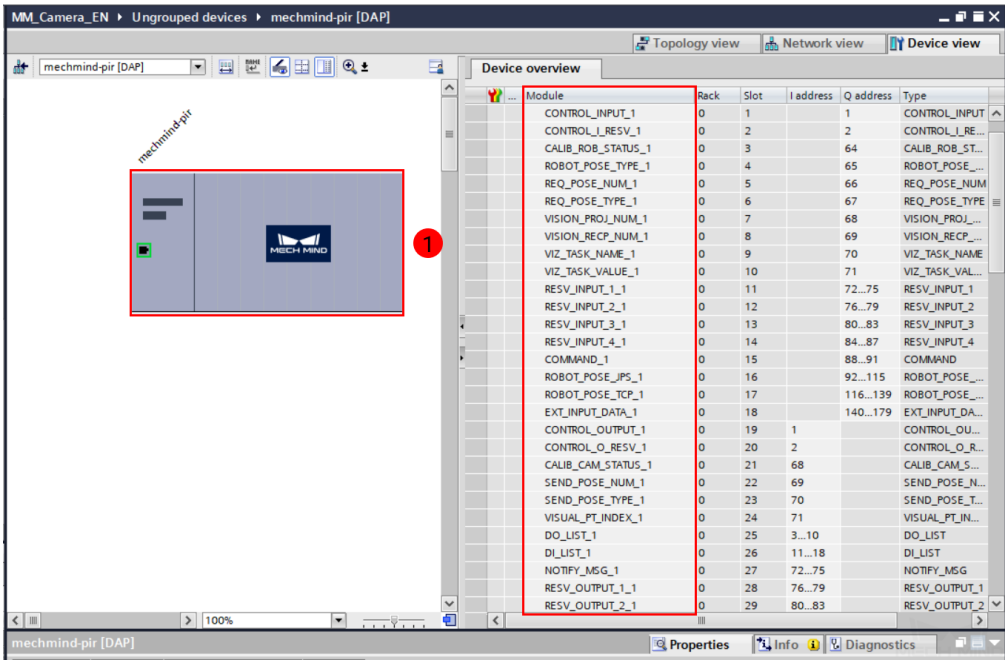


7. In the pop-up window, select **mechmind-pir** for **PROFINET device name**, and then click on *Update list*.

8. When the device appears in the list, check whether the **PROFINET device name** is **mechmind-pir**. If not, click on *Assign name*. Once **Status** is **OK**, click on *Close* to close the window.

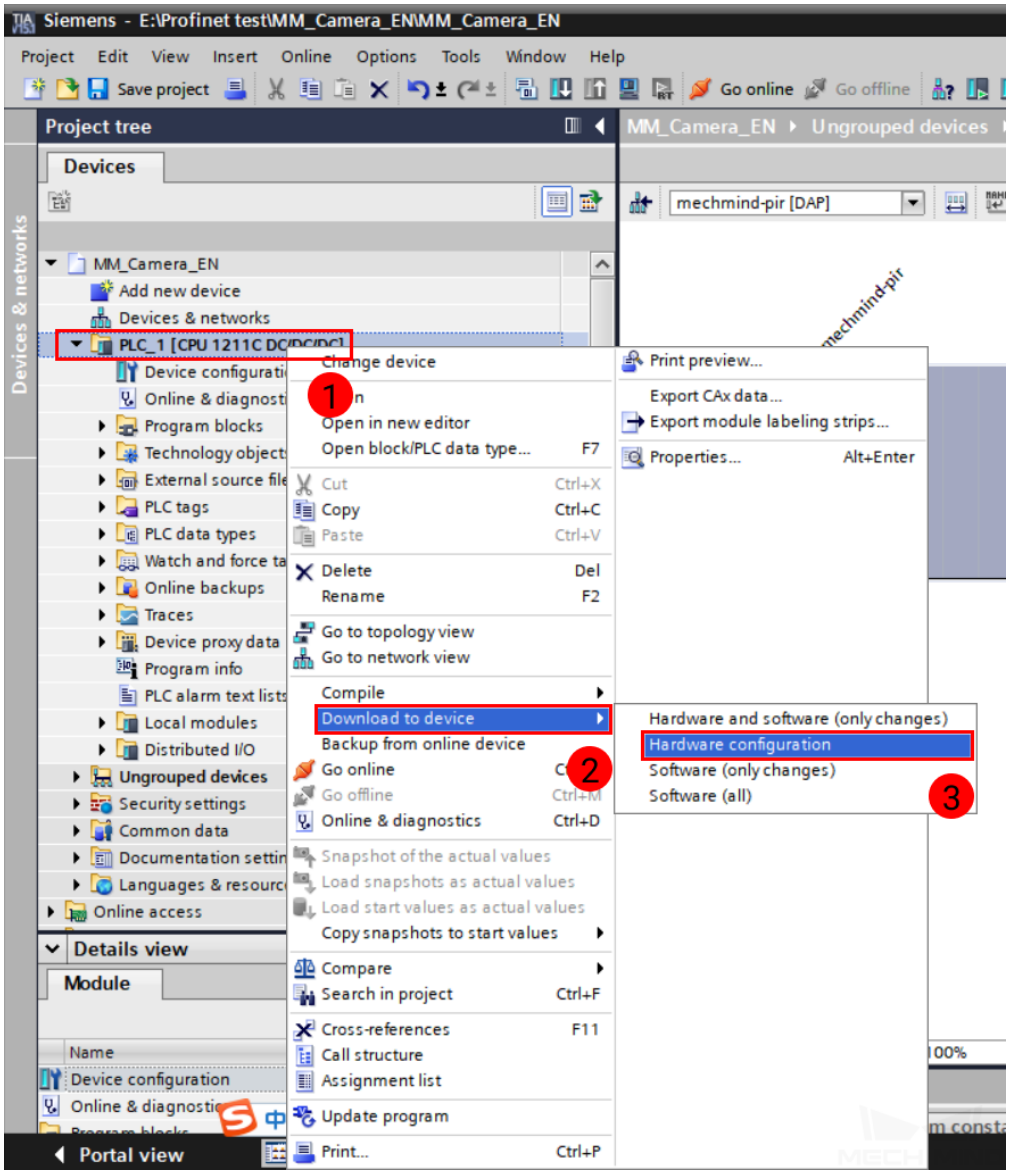


9. Double-click on **mechmind-pir** to enter **Device view**. You can see all the available modules listed.

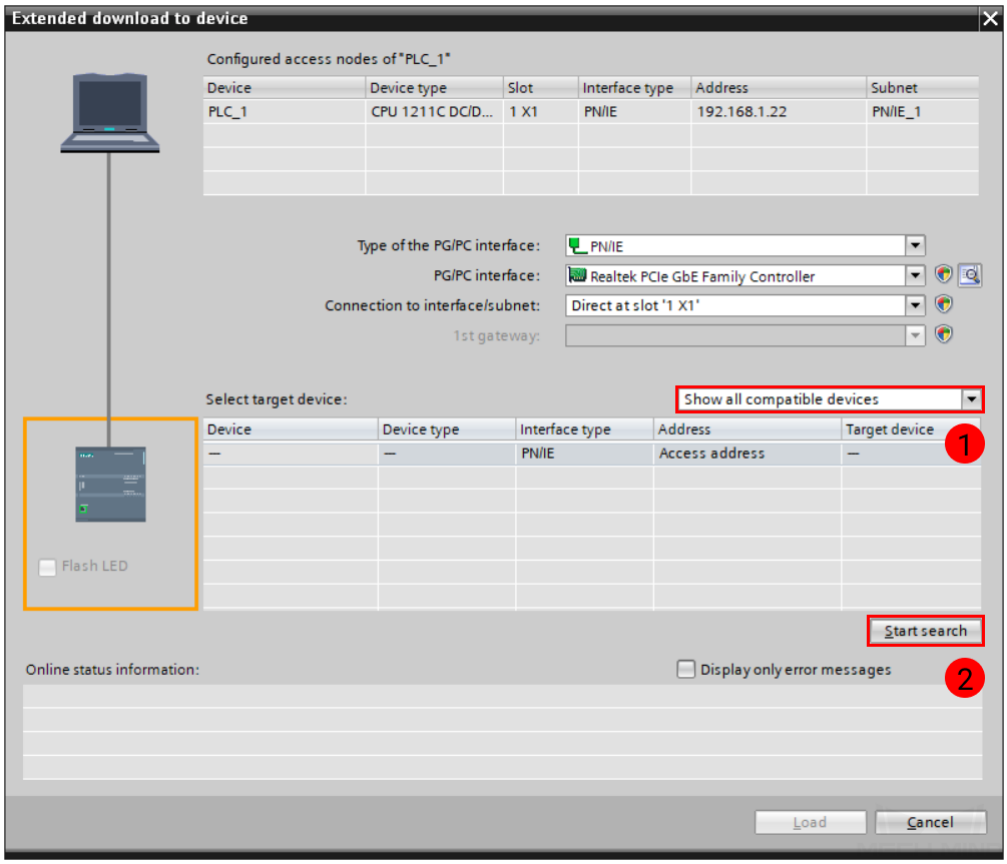


Download Hardware Configuration to PLC

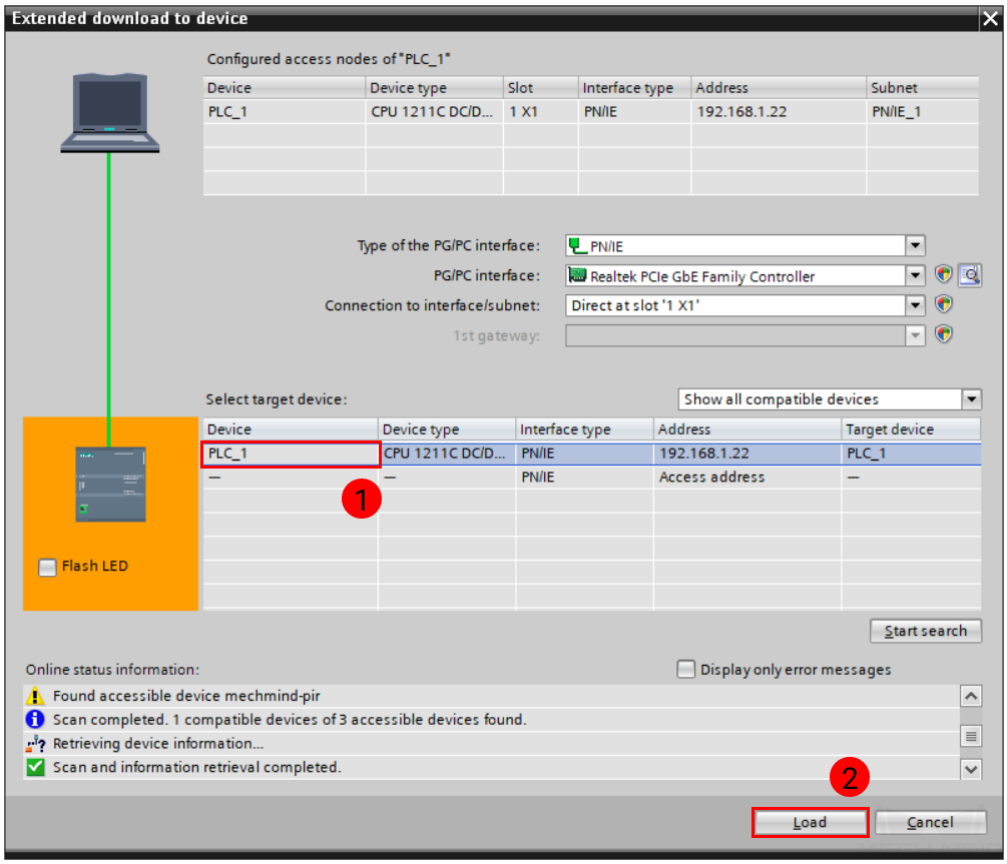
1. In **Project tree** panel, right-click on **PLC_1**, and select *Download to device* → *Hardware configuration*.



2. In the pop-up window, select **Show all compatible devices** for **Select target device**, and then click on *Start search*.

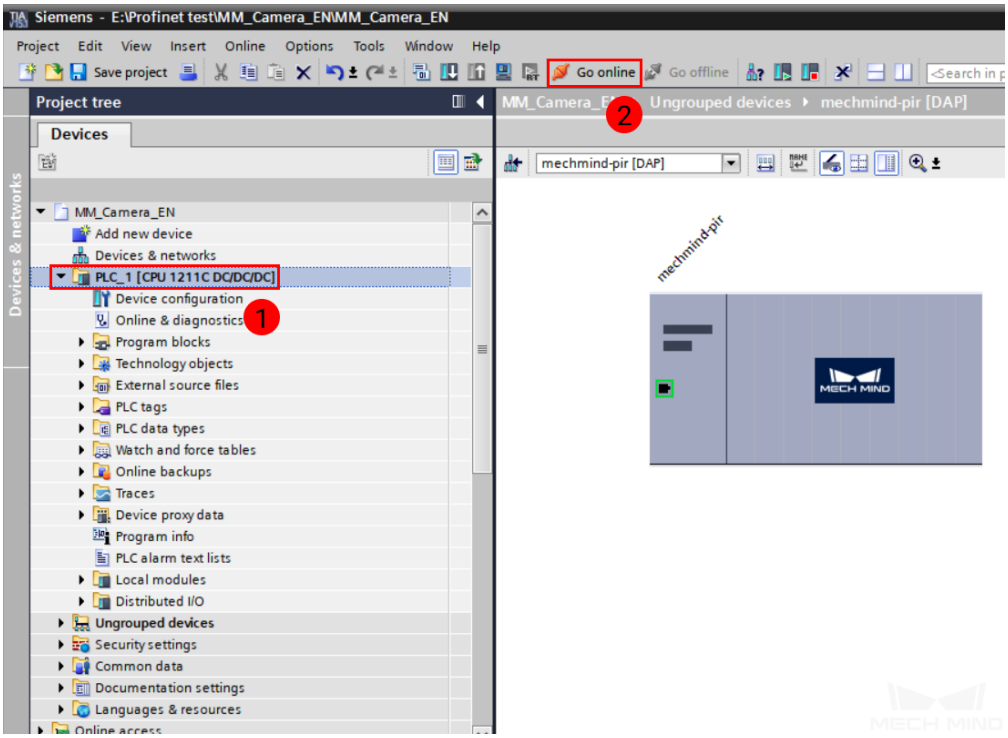


3. Select the corresponding device in the search result, and click on *Load*.

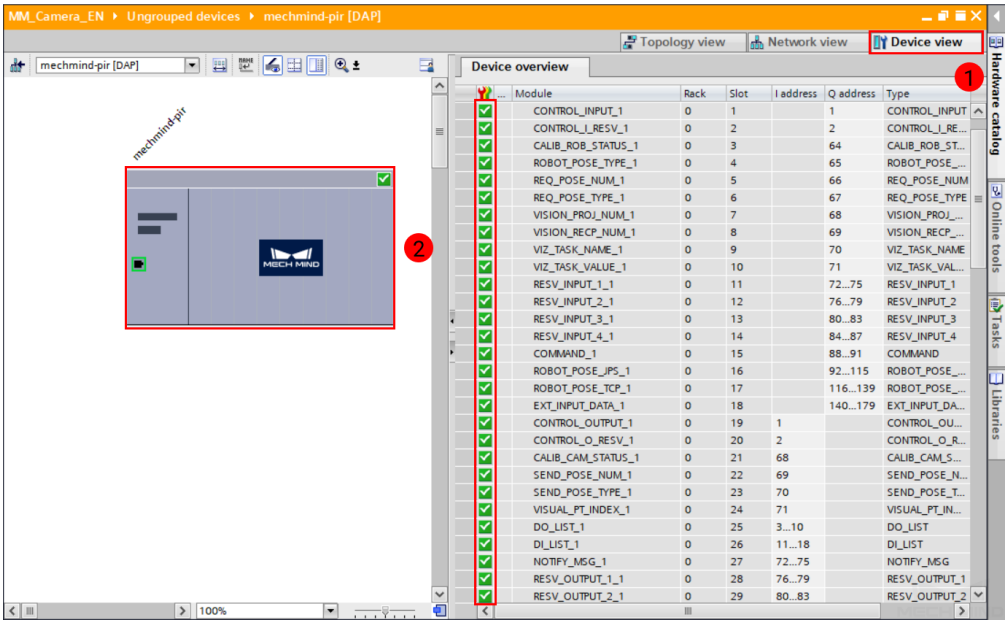


Check Communication

1. Return to the project, and click on **PLC_1** in the **Project tree** panel. Then, click on *Go online* in the toolbar.



2. In the **Project tree** panel, click on **Device configuration**, and then click on **Device view** tab on the right. Select **mechmind-pir**. In **Device overview**, check marks with a green background in front of module names indicate normal connection.



3. The PLC is successfully connected to Mech-Center if the following message is displayed in Mech-Center **Log** panel:

   Connect to PROFINET-IRT controller successfully

Note: If you don' t see this log message, please check if:

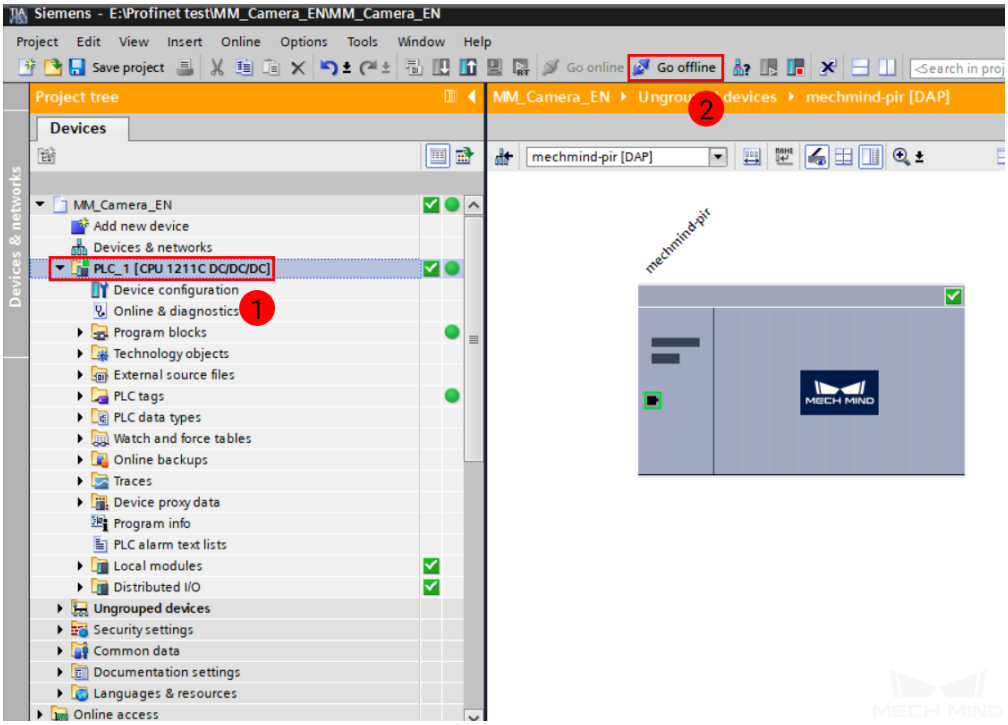
- The hardware are properly connected;
- If Mech-Interface has been started by clicking on *Start Interface* in the Toolbar;
- If the hardware configuration has been downloaded to the PLC.

2.8.5 Import Example Program and Download to PLC

Note: Before you add the example program to a project already in use, it is recommended to import it to a new project and test it first. In the following steps, the project created earlier is used to import and test the example program.

Import Example Program Files

1. Select **PLC_1** in the **Program tree** panel, and then click on *Go offline* in the toolbar.



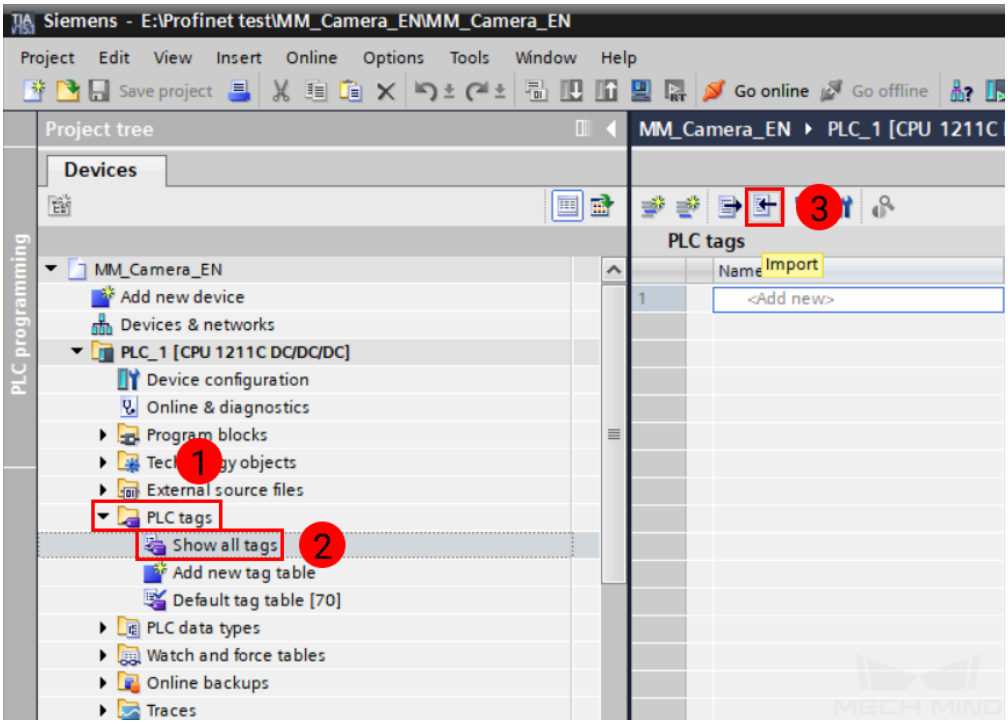
2. In **Network view**, double-click on **mechmind-pir** to enter **Device view**. Change the I addresses and Q addresses according to your actual needs. Here, **500** is used as the lowest module start address.

Note: For a module occupying multiple bytes, the addresses assigned must be continuous, and the module start address must be of an even number.

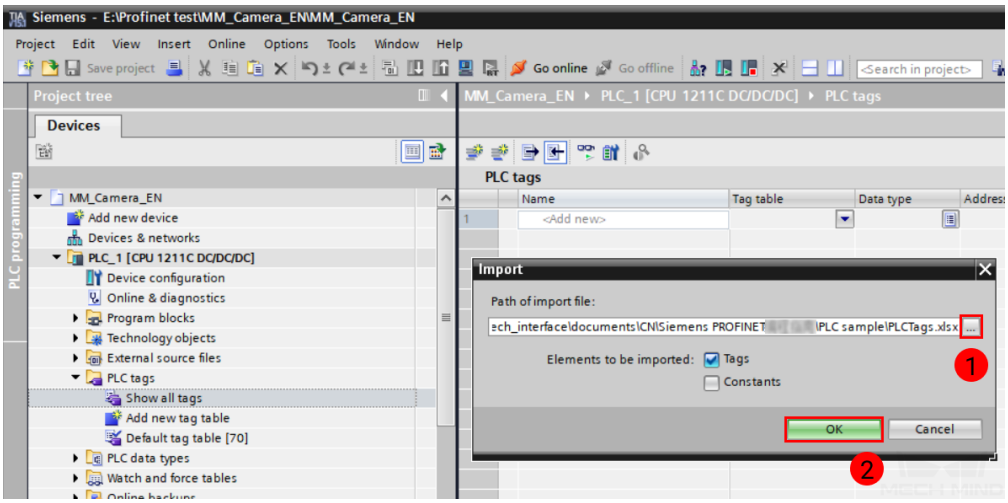
Device overview

Module	Rack	Slot	I address	Q address	Type	Article no.	Firmware	Comment
CONTROL_INPUT_1	0	1		500	CONTROL_INPUT			
CONTROL_I_RESV_1	0	2		501	CONTROL_I_RESV			
CALIB_ROB_STATUS_1	0	3		502	CALIB_ROB_STATUS			
ROBOT_POSE_TYPE_1	0	4		503	ROBOT_POSE_TYPE			
REQ_POSE_NUM_1	0	5		504	REQ_POSE_NUM			
REQ_POSE_TYPE_1	0	6		505	REQ_POSE_TYPE			
VISION_PROJ_NUM_1	0	7		506	VISION_PROJ_NUM			
VISION_RECP_NUM_1	0	8		507	VISION_RECP_NUM			
VIZ_TASK_NAME_1	0	9		508	VIZ_TASK_NAME			
VIZ_TASK_VALUE_1	0	10		509	VIZ_TASK_VALUE			
RESV_INPUT_1_1	0	11		510...513	RESV_INPUT_1			
RESV_INPUT_2_1	0	12		514...517	RESV_INPUT_2			
RESV_INPUT_3_1	0	13		518...521	RESV_INPUT_3			
RESV_INPUT_4_1	0	14		522...525	RESV_INPUT_4			
COMMAND_1	0	15		526...529	COMMAND			
ROBOT_POSE_JPS_1	0	16		530...553	ROBOT_POSE_JPS			
ROBOT_POSE_TCP_1	0	17		554...577	ROBOT_POSE_TCP			
EXT_INPUT_DATA_1	0	18		578...617	EXT_INPUT_DATA			
CONTROL_OUTPUT_1	0	19	500		CONTROL_OUTPUT			
CONTROL_O_RESV_1	0	20	501		CONTROL_O_RESV			
CALIB_CAM_STATUS_1	0	21	502		CALIB_CAM_STATUS			
SEND_POSE_NUM_1	0	22	503		SEND_POSE_NUM			
SEND_POSE_TYPE_1	0	23	504		SEND_POSE_TYPE			
VISUAL_PT_INDEX_1	0	24	505		VISUAL_PT_INDEX			
DO_LIST_1	0	25	506...513		DO_LIST			
DI_LIST_1	0	26	514...521		DI_LIST			
NOTIFY_MSG_1	0	27	522...525		NOTIFY_MSG		Article number	
RESV_OUTPUT_1_1	0	28	526...529		RESV_OUTPUT_1			
RESV_OUTPUT_2_1	0	29	530...533		RESV_OUTPUT_2			
RESV_OUTPUT_3_1	0	30	534...537		RESV_OUTPUT_3			
STATUS_CODE_1	0	31	538...541		STATUS_CODE			
TARGET_POSE_1	0	32	542...565		TARGET_POSE			
TARGET_LABEL_1	0	33	566...569		TARGET_LABEL			
TARGET_SPEED_1	0	34	570...573		TARGET_SPEED			
EXT_OUTPUT_DATA_1	0	35	574...613		EXT_OUTPUT_DATA			
	0	36						

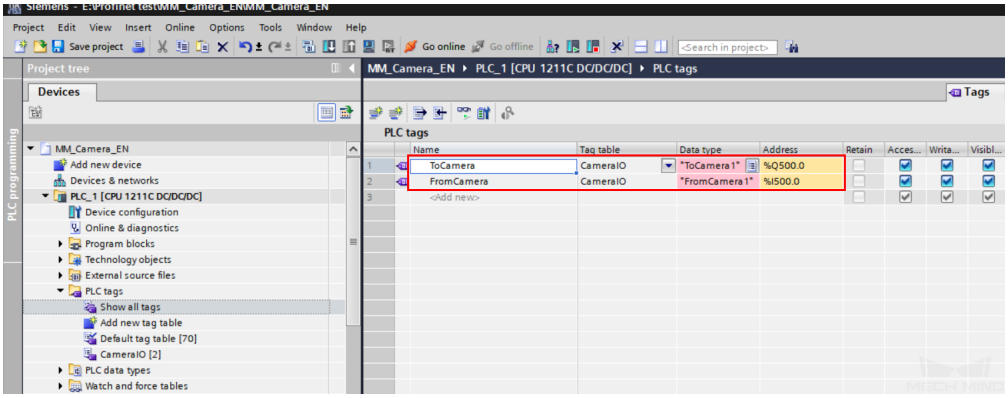
3. In **Project tree** panel, double-click on **PLC tags** under **PLC_1**, and then double-click on **Show all tags** to open the **PLC tags** window. Then, click on  to import tags.



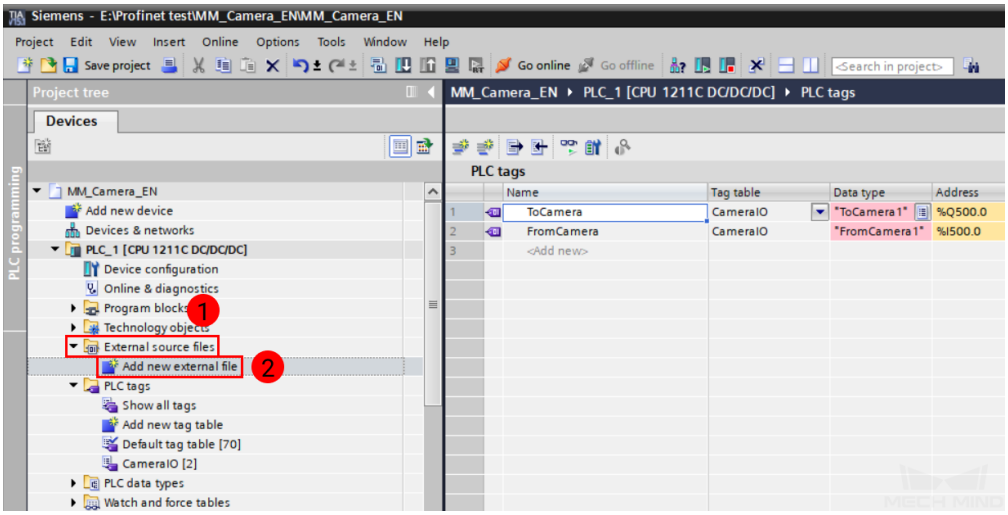
4. Click on ··· to the right of the input field, and locate the **PLCTags.elsx** file. Click on OK to import the PLC tags.



5. The following tags should be imported. Change the address of **ToCamera** and **FromCamera** to the same as those of **CONTROL_INPUT_1** and **CONTROL_OUTPUT_1** modules, respectively.

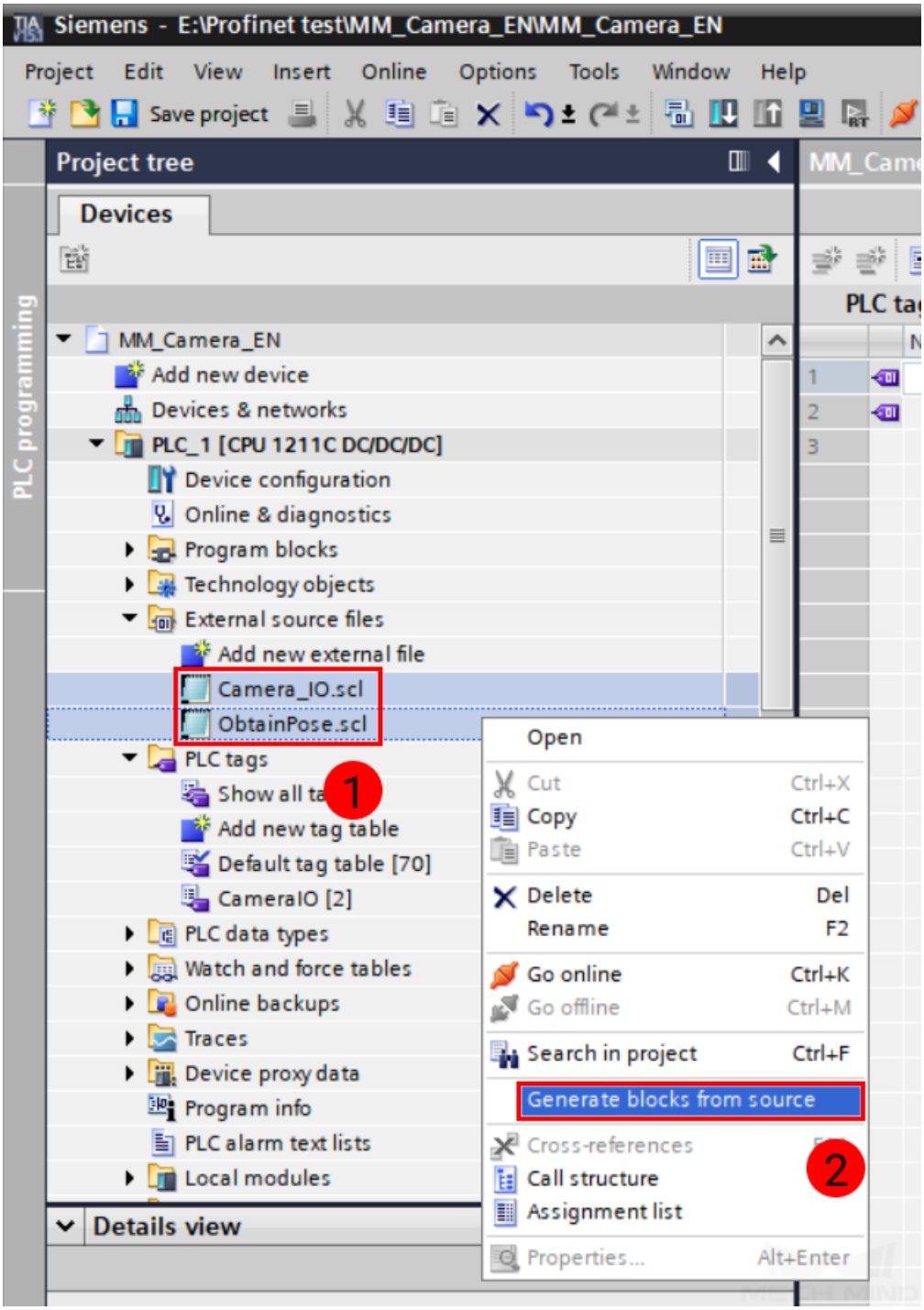


6. In the **Project tree** panel, double-click on **External source files**, and then double-click on **Add new external file**.

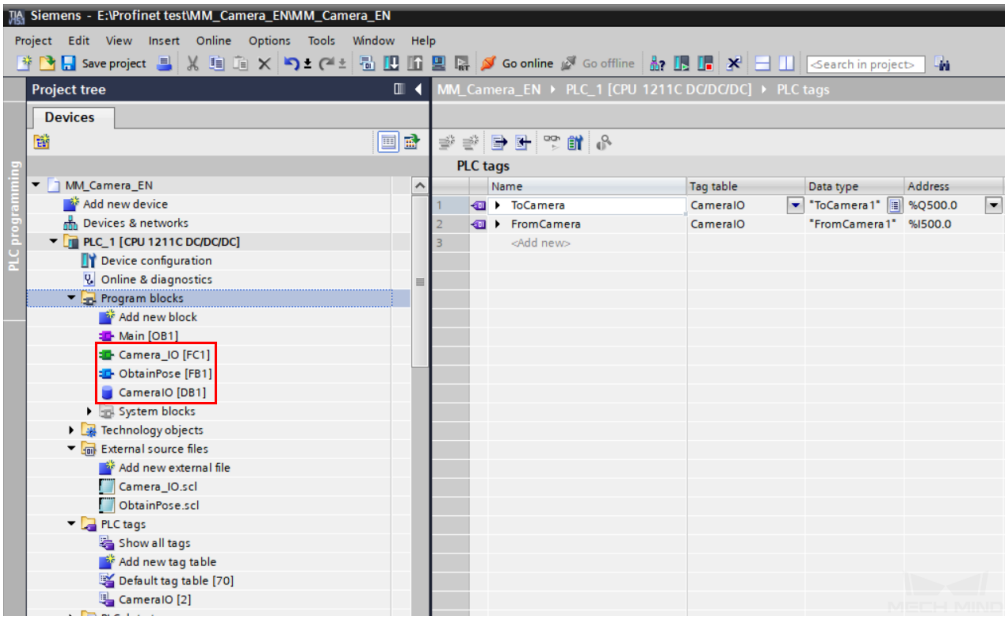


7. In the pop-up window, locate and select the **Camera_IO.scl** and **ObtainPose.scl** files. Click on *Open* to import these files.

8. Select the two imported external files and then right-click on these files. In the right-click menu, select **Generate blocks from source**.

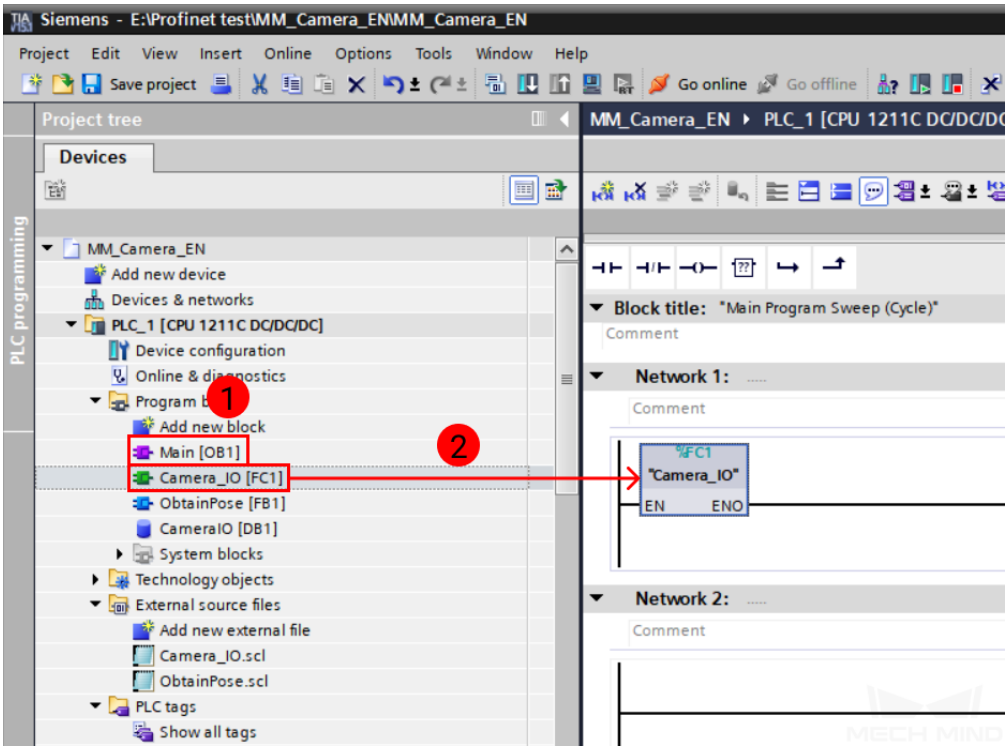


9. Three program blocks should be generated: **Camera_IO** FC, **ObtainPose** FB, and **CameraIO** DB.



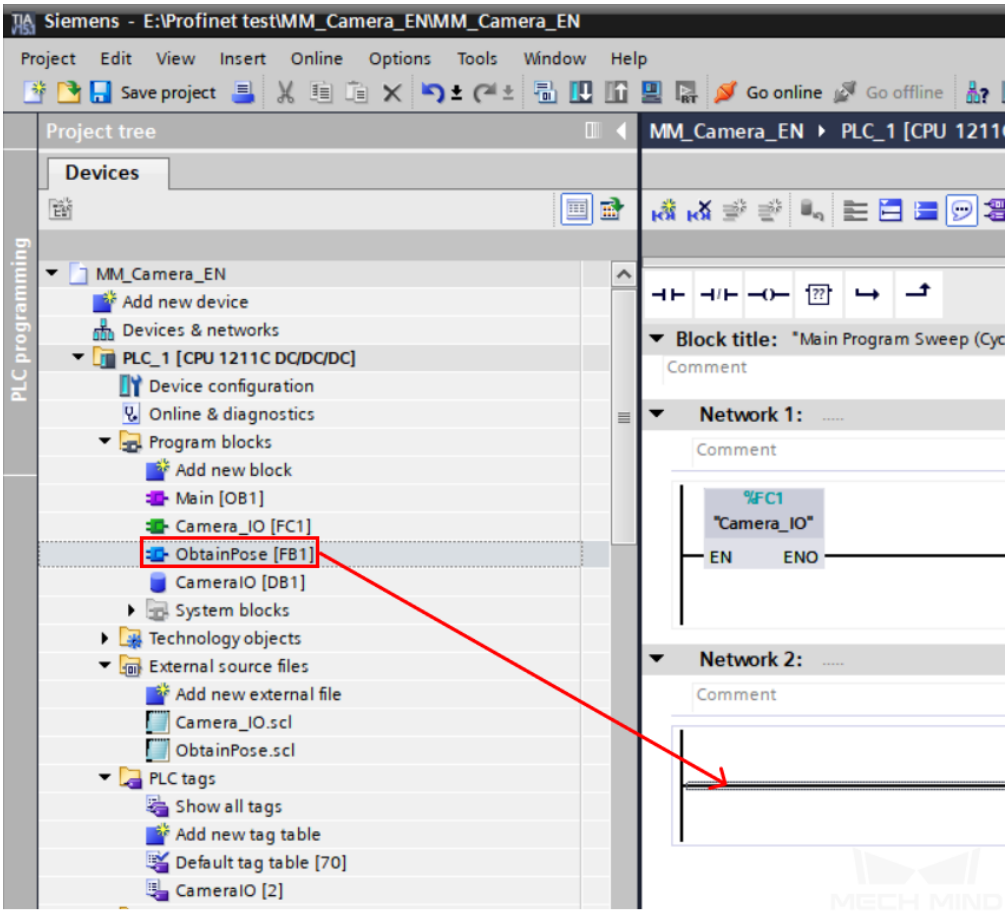
2.8.6 Build Program and Download to PLC

1. In the **Program tree** panel, double-click on the **Main** OB in **Program blocks** to open it. Then, select **Camera_IO** FC and drag it to **Network 1**.

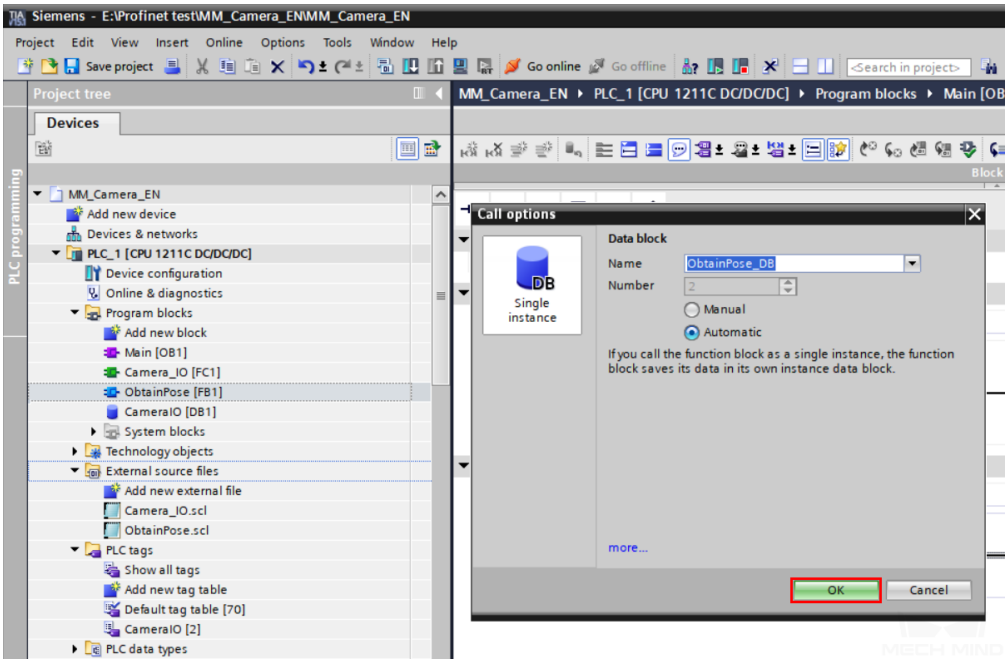


2. Select **ObtainPose** FB and drag it to **Network 2**. A window will pop-up when you release the

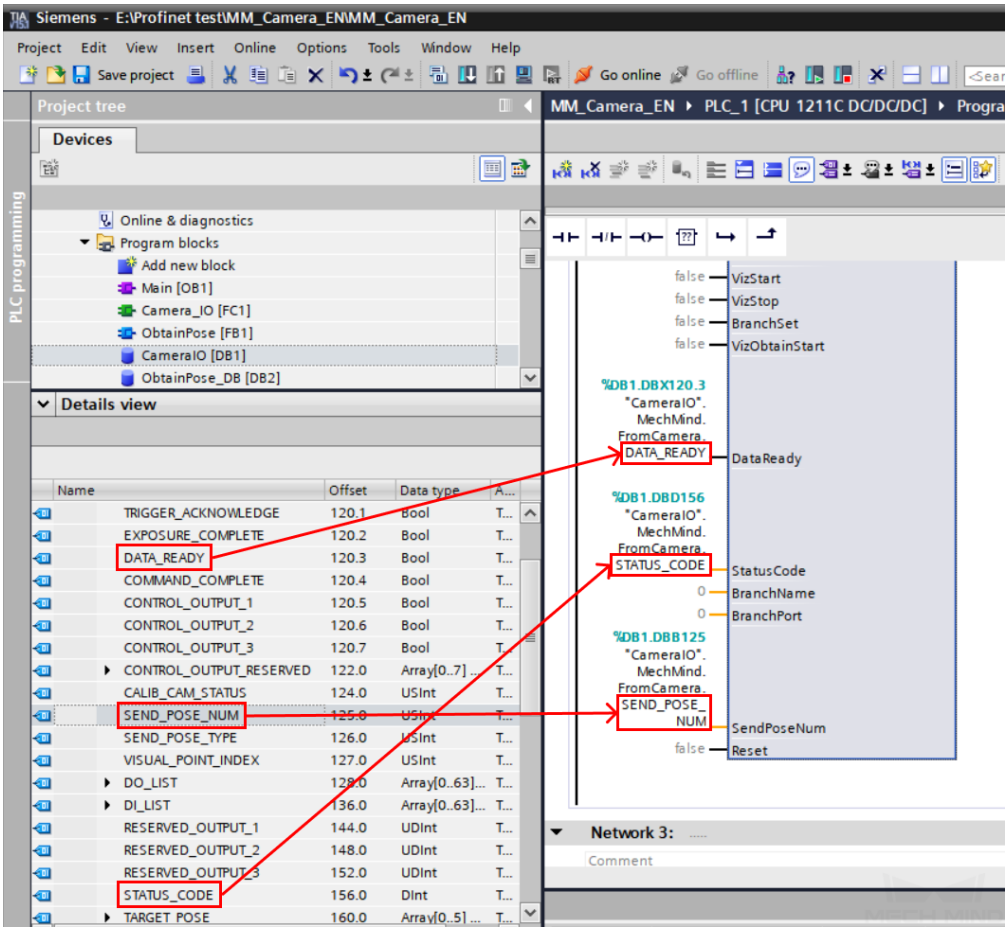
mouse button.



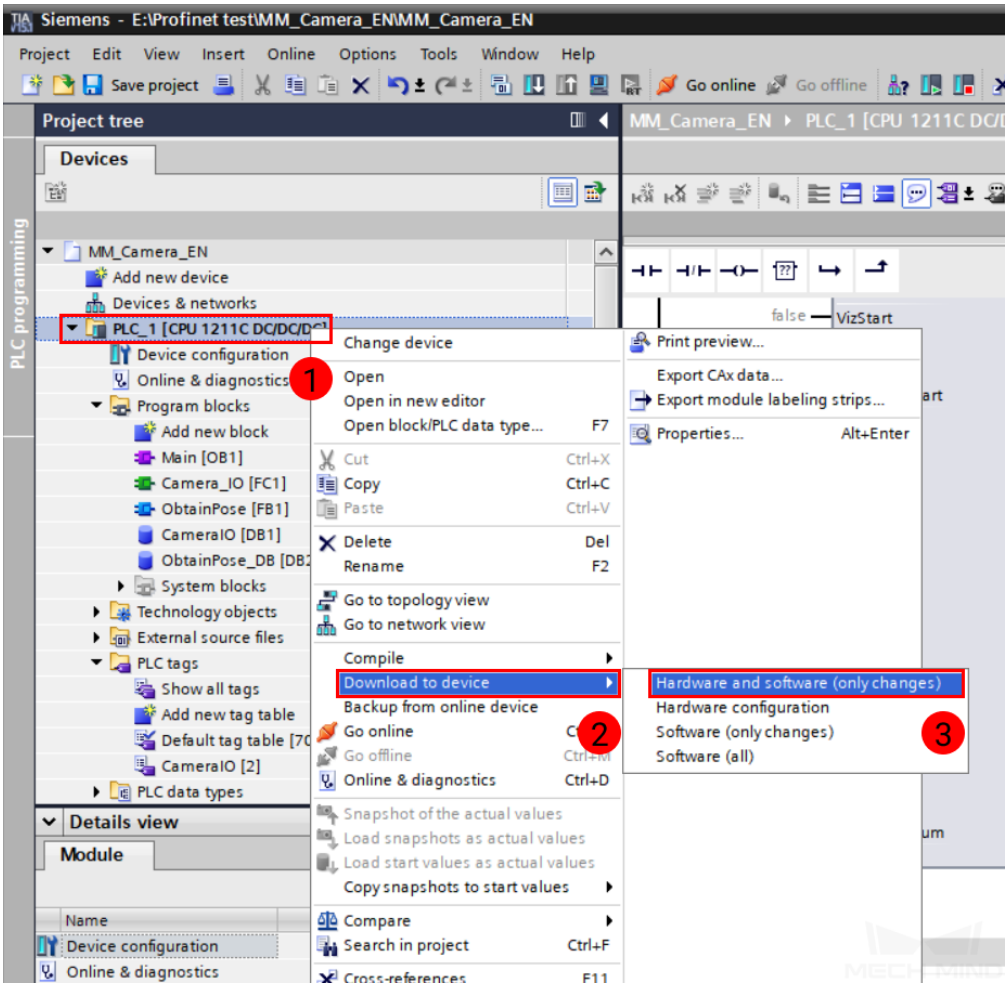
3. In the pop-up window, keep the default options and click on *OK*.



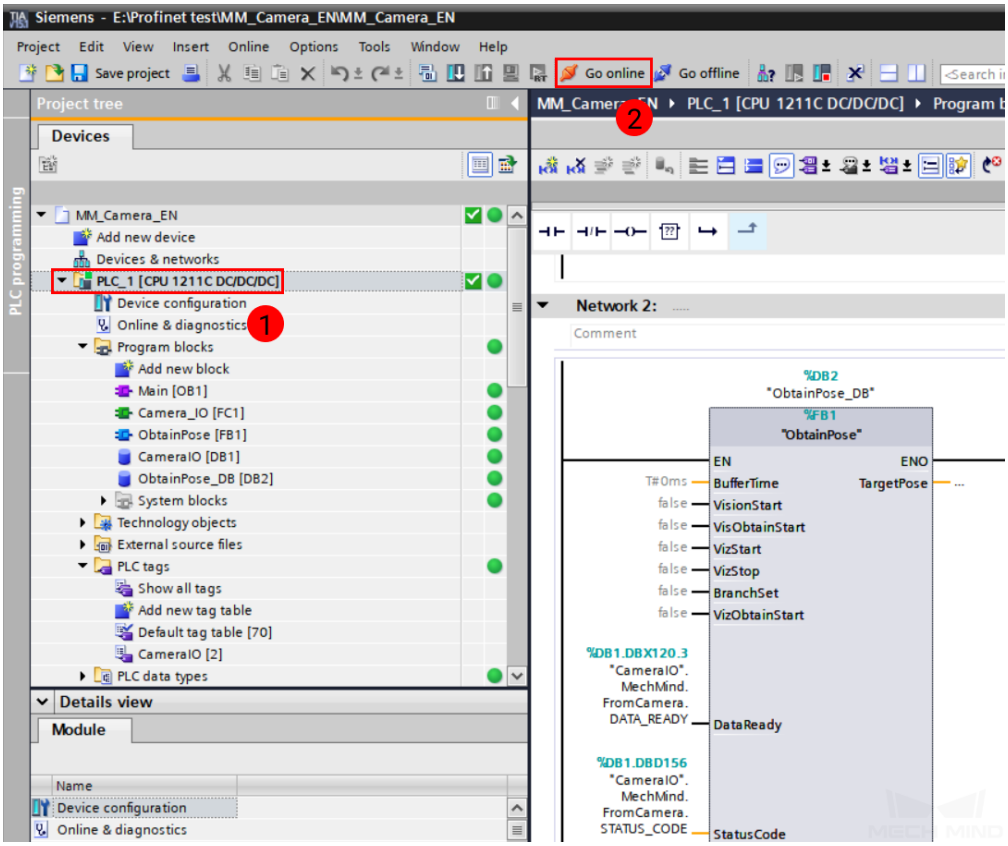
4. Click on **CameraIO** DB, and under the **Details view**, drag **DATA_READY**, **SEND_POSE_NUM** and **STATUS_CODE** to the input ports with the same names on the left side of **ObtainPose** FB.



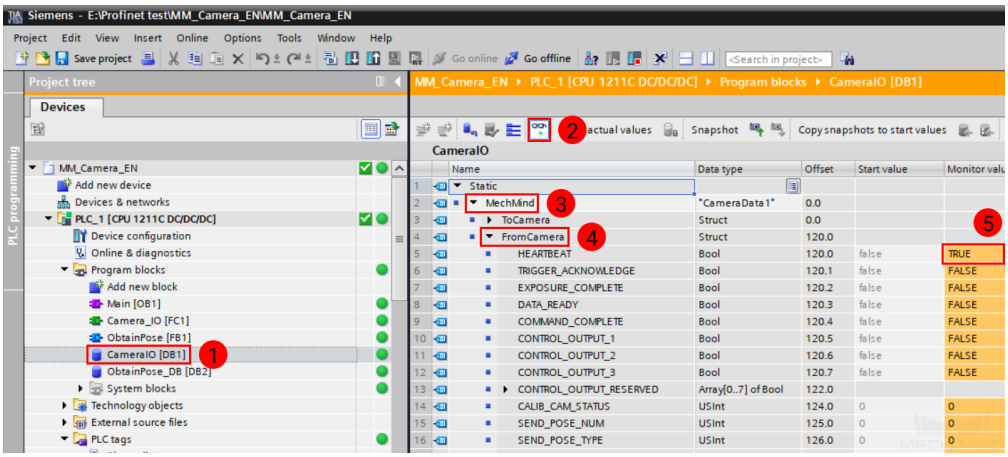
5. In the **Program tree** panel, right-click on **PLC_1**, and select *Download to device* → *Hardware and software (only changes)*. Refer to steps 2 and 3 in *Download Hardware Configuration to PLC* above and download the program to the PLC.



6. After downloading, select **PLC_1** in the **Program tree** panel, and click on *Go online*.



7. In the **Program tree** panel, double-click on *Program blocks* → *CameraIO* and then click on  . Double-click to expand *MechMind* → *FromCamera*. If the monitor value of **HEARTBEAT** keeps changing, then the I/Q addresses of modules in **mechmind-pir** have been successfully transferred to the **CameraIO** FB.

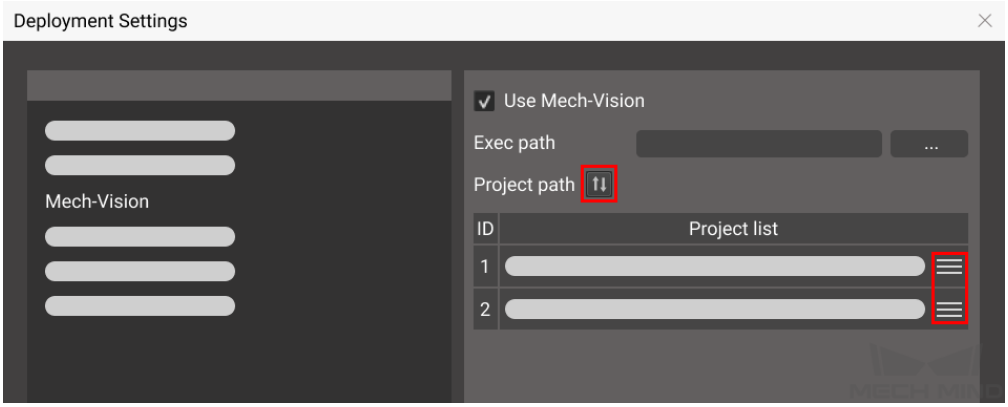


2.8.7 Test with Mech-Vision/Mech-Viz Project

This section introduces how to run the Mech-Vision/Mech-Viz project and obtain data from the project using the **ObtainPose** FB. For detailed information on the modules, please refer to standard_interface_development_profinet.

Prerequisites

- Mech-Vision project(s):
  - Executable
  - Set to autoload
  - The **Project list** in *Mech-Center* → *Deployment Settings* → *Mech-Vision* is synced by clicking on  , and the order of Mech-Vision projects have been adjusted according to actual needs.



- Mech-Viz project:
  - Executable
  - Set to autoload
  - Contains a branch_by_service_message Task that has been renamed to **1**.

Run Mech-Vision Project and Obtain Vision Points

1. Enable communication: in the **CameraIO** DB, double-click on the monitor value of **COMM_ENABLE** in **ToCamera**. If a warning message pops up, click on *Yes* to confirm changing the monitor value.
2. Set Mech-Vision project ID: change the monitor value of **VISION_PROJ_NUM** to the ID of the Mech-Vision project you wish to run. For example, if the monitor value is changed to **1**, then Mech-Vision project No. 1 in the **Project list** of Mech-Center will be started.
3. Set the number of vision points to be sent by Mech-Vision: change the monitor value of **REQ_POSE_NUM**. If the value is set to **0**, the Mech-Vision project will send all the vision points.

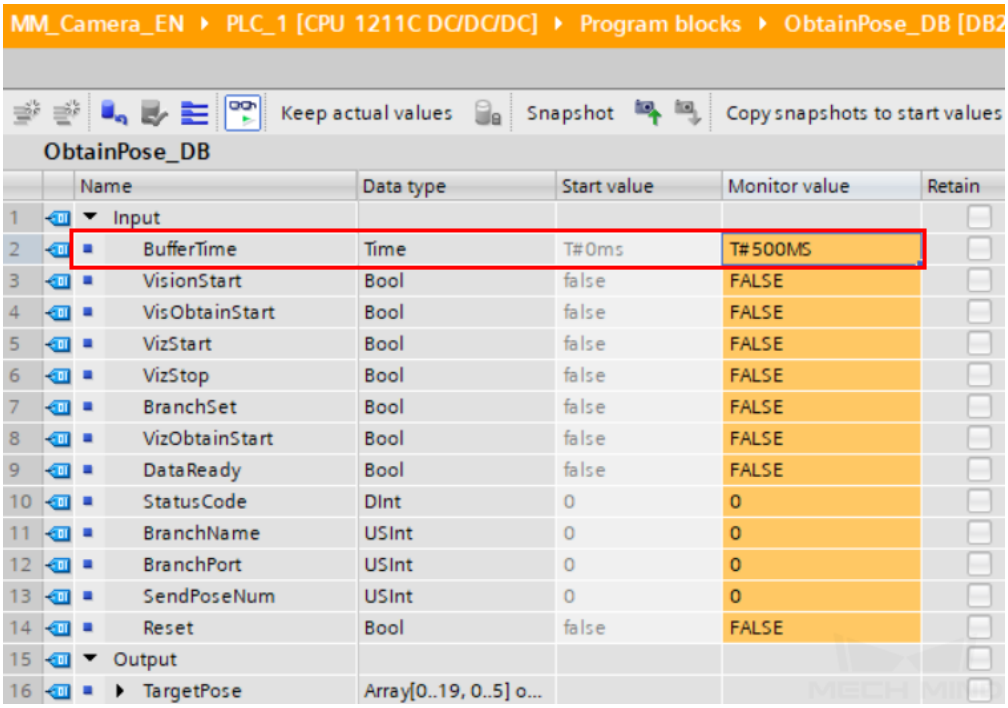
MM_Camera ▸ PLC_1 [CPU 1211C DC/DC/DC] ▸ Program blocks ▸ CameraIO [DB1]

Keep actual values | Snapshot | Copy snapshots to start values

CameraIO

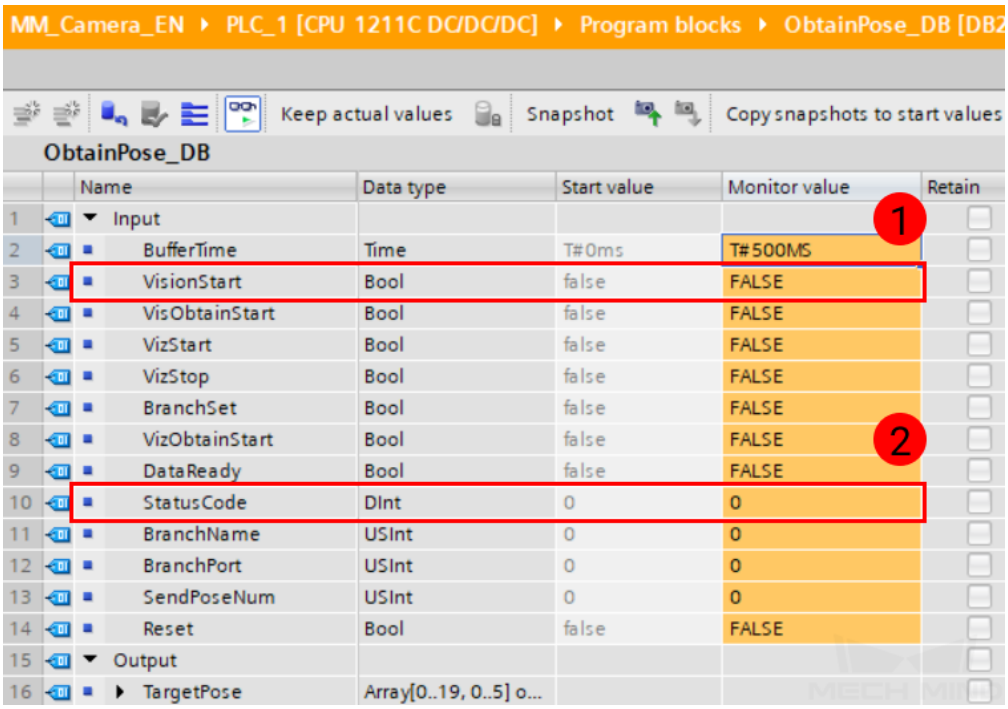
	Name	Data type	Offset	Start value	Monitor value
1	Static				
2	MechMind	"CameraData1"	0.0		
3	ToCamera	Struct	0.0		
4	COMM_ENABLE	Bool	0.0	false	TRUE
5	TRIGGER	Bool	0.1	false	FALSE
6	RESET_EXPOSURE	Bool	0.2	false	FALSE
7	DATA_ACKNOWLEDGE	Bool	0.3	false	FALSE
8	CLEAR_NOTIFY	Bool	0.4	false	FALSE
9	CONTROL_INPUT_1	Bool	0.5	false	FALSE
10	CONTROL_INPUT_2	Bool	0.6	false	FALSE
11	CONTROL_INPUT_3	Bool	0.7	false	FALSE
12	CONTROL_INPUP_RESERVED	Array[0..7] of Bool	2.0		
13	CALIB_ROB_STATUS	USInt	4.0	0	0
14	ROBOT_POSE_TYPE	USInt	5.0	0	0
15	REQ_POSE_NUM	USInt	6.0	0	0
16	REQ_POSE_TYPE	USInt	7.0	0	0
17	VISION_PROJ_NUM	USInt	8.0	0	1
18	VISION_RECIPE_NUM	USInt	9.0	0	0
19	VIZ_TASK_NAME	USInt	10.0	0	0
20	VIZ_TASK_VALUE	USInt	11.0	0	0
21	RESERVED_INPUT_1	UDInt	12.0	0	0
22	RESERVED_INPUT_2	UDInt	16.0	0	0
23	RESERVED_INPUT_3	UDInt	20.0	0	0
24	RESERVED_INPUT_4	UDInt	24.0	0	0
25	COMMAND	DInt	28.0	0	0
26	ROBOT_POSE_JPS	Array[0..5] of DInt	32.0		
27	ROBOT_POSE_TCP	Array[0..5] of DInt	56.0		
28	EXT_INPUT_DATA	Array[0..9] of DInt	80.0		

4. Set the hold time of signals sent to the IPC: double-click on **ObtainPose_DB** DB, and click on  . Under **Input** group, change the monitor value of **BufferTime** to **500MS**.



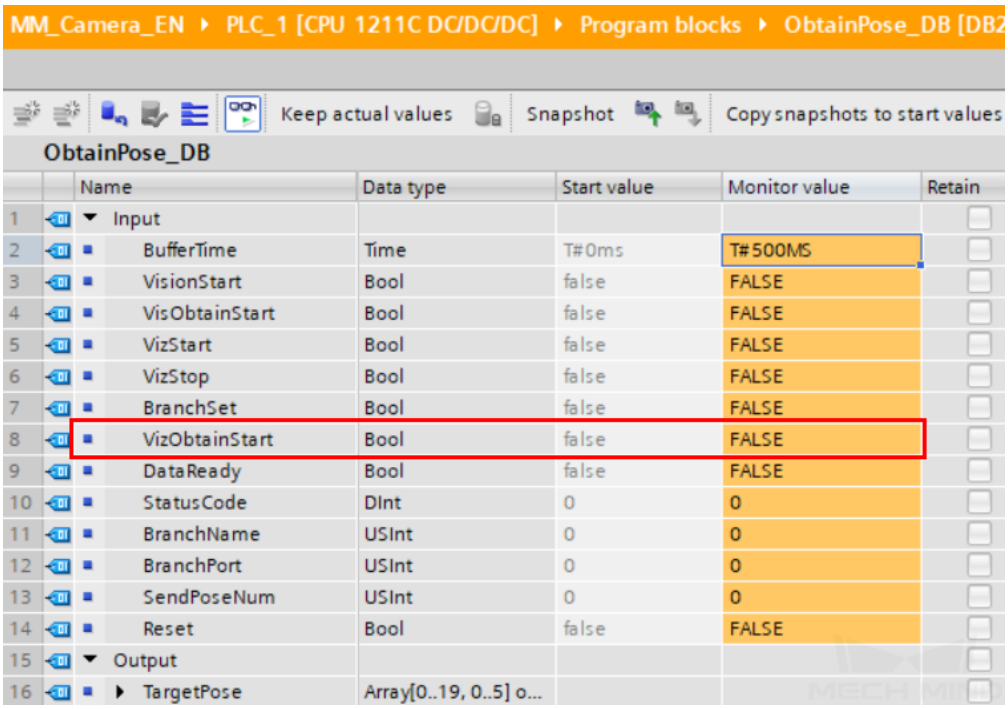
	Name	Data type	Start value	Monitor value	Retain
1	Input				
2	BufferTime	Time	T#0ms	T#500MS	
3	VisionStart	Bool	false	FALSE	
4	VisObtainStart	Bool	false	FALSE	
5	VizStart	Bool	false	FALSE	
6	VizStop	Bool	false	FALSE	
7	BranchSet	Bool	false	FALSE	
8	VizObtainStart	Bool	false	FALSE	
9	DataReady	Bool	false	FALSE	
10	StatusCode	DInt	0	0	
11	BranchName	USInt	0	0	
12	BranchPort	USInt	0	0	
13	SendPoseNum	USInt	0	0	
14	Reset	Bool	false	FALSE	
15	Output				
16	TargetPose	Array[0..19, 0..5] o...			

5. Start the Mech-Vision project: in the **ObtainPose_DB** DB, double-click on the monitor value of **VisionStart** to change it to **TRUE**. Then, reset the monitor value to **FALSE**.
6. Check returned status code: check the monitor value of **StatusCode**. **1102** represents that the Mech-Vision project was started successfully. For other values, please refer to standard_interface_status_codes for the corresponding error.



	Name	Data type	Start value	Monitor value	Retain
1	Input				
2	BufferTime	Time	T#0ms	T#500MS	
3	VisionStart	Bool	false	FALSE	
4	VisObtainStart	Bool	false	FALSE	
5	VizStart	Bool	false	FALSE	
6	VizStop	Bool	false	FALSE	
7	BranchSet	Bool	false	FALSE	
8	VizObtainStart	Bool	false	FALSE	
9	DataReady	Bool	false	FALSE	
10	StatusCode	DInt	0	0	
11	BranchName	USInt	0	0	
12	BranchPort	USInt	0	0	
13	SendPoseNum	USInt	0	0	
14	Reset	Bool	false	FALSE	
15	Output				
16	TargetPose	Array[0..19, 0..5] o...			

7. Obtain vision points from the Mech-Vision project: in the **ObtainPose_DB** DB, double-click on the monitor value of **VisObtainStart** to change it to **TRUE**. Then, reset the monitor value to **FALSE**.



	Name	Data type	Start value	Monitor value	Retain
1	Input				
2	BufferTime	Time	T#0ms	T#500MS	
3	VisionStart	Bool	false	FALSE	
4	VisObtainStart	Bool	false	FALSE	
5	VizStart	Bool	false	FALSE	
6	VizStop	Bool	false	FALSE	
7	BranchSet	Bool	false	FALSE	
8	VizObtainStart	Bool	false	FALSE	
9	DataReady	Bool	false	FALSE	
10	StatusCode	DInt	0	0	
11	BranchName	USInt	0	0	
12	BranchPort	USInt	0	0	
13	SendPoseNum	USInt	0	0	
14	Reset	Bool	false	FALSE	
15	Output				
16	TargetPose	Array[0..19, 0..5] o...			

8. Check the received vision points: the monitor value of **SendPoseNum** shows how many vision points were received, and the vision points are stored in **TargetPose**. Divide the monitor values by 10000 to obtain the actual pose data.

MM_Camera ▸ PLC_1 [CPU 1211C DC/DC/DC] ▸ Program blocks ▸ ObtainPose_DB [DB2]

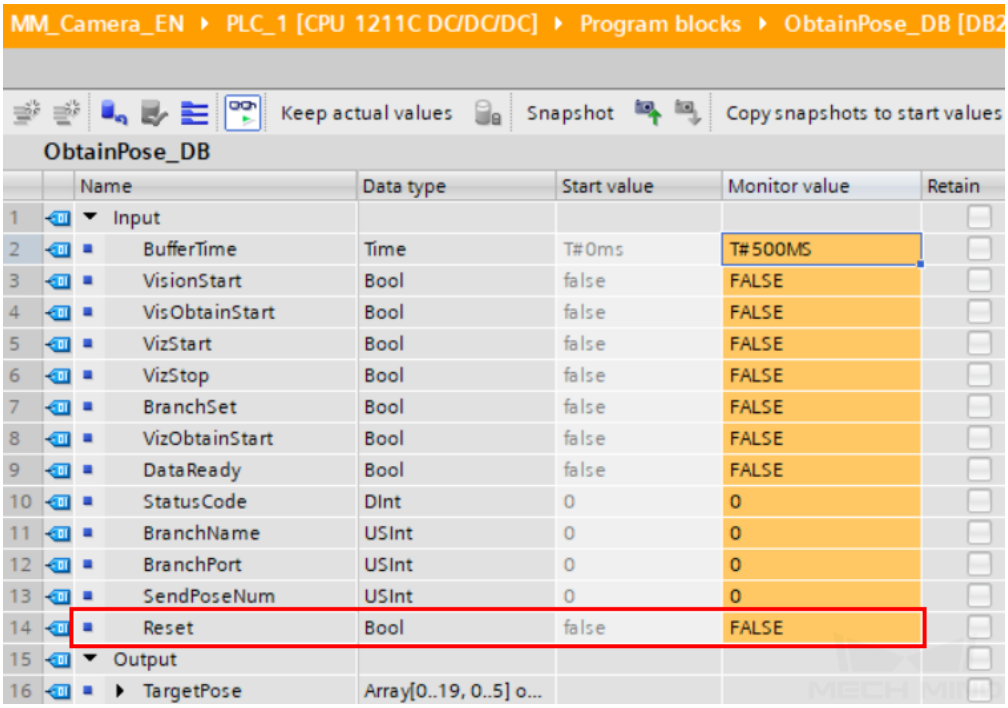
Keep actual values | Snapshot | Copy snapshots to start values

ObtainPose_DB

	Name	Data type	Start value	Monitor value	Retain
1	Input				
2	BufferTime	Time	T#0ms	T#500MS	
3	VisionStart	Bool	false	FALSE	
4	VisObtainStart	Bool	false	FALSE	
5	VizStart	Bool	false	FALSE	
6	VizStop	Bool	false	FALSE	
7	BranchSet	Bool	false	FALSE	
8	VizObtainStart	Bool	false	FALSE	
9	DataReady	Bool	false	FALSE	
10	StatusCode	DInt	0	1100	
11	BranchName	USInt	0	0	
12	BranchPort	USInt	0	0	
13	SendPoseNum	USInt	0	2	
14	Reset	Bool	false	FALSE	
15	Output				
16	TargetPose	Array[0..19, 0..5] o...			
17	TargetPose[0,0]	DInt	0	-4618258	
18	TargetPose[0,1]	DInt	0	5623557	
19	TargetPose[0,2]	DInt	0	523405	
20	TargetPose[0,3]	DInt	0	1206150	
21	TargetPose[0,4]	DInt	0	6075	
22	TargetPose[0,5]	DInt	0	-1780788	
23	TargetPose[1,0]	DInt	0	-3078879	
24	TargetPose[1,1]	DInt	0	6498989	
25	TargetPose[1,2]	DInt	0	508875	
26	TargetPose[1,3]	DInt	0	1428931	
27	TargetPose[1,4]	DInt	0	5581	
28	TargetPose[1,5]	DInt	0	-1787682	
29	TargetPose[2,0]	DInt	0	0	
30	TargetPose[2,1]	DInt	0	0	

Run Mech-Viz Project and Obtain Planned Path

1. Clear last obtained data: in the **ObtainPose_DB** DB, double-click on the monitor value of **Reset** to change it to **TRUE**. Then, reset the monitor value to **FALSE**.



	Name	Data type	Start value	Monitor value	Retain
1	Input				
2	BufferTime	Time	T#0ms	T#500MS	
3	VisionStart	Bool	false	FALSE	
4	VisObtainStart	Bool	false	FALSE	
5	VizStart	Bool	false	FALSE	
6	VizStop	Bool	false	FALSE	
7	BranchSet	Bool	false	FALSE	
8	VizObtainStart	Bool	false	FALSE	
9	DataReady	Bool	false	FALSE	
10	StatusCode	DInt	0	0	
11	BranchName	USInt	0	0	
12	BranchPort	USInt	0	0	
13	SendPoseNum	USInt	0	0	
14	Reset	Bool	false	FALSE	
15	Output				
16	TargetPose	Array[0..19, 0..5] o...			

2. Set the branch to take in the Mech-Viz project:
   - Set Task name: in the **ObtainPose_DB** DB, change the monitor value of **BranchName** to **1**. This tells Mech-Viz that you are trying to select the out port for the Task named **1**.
   - Select out port: change the monitor value of **BranchPort** to the out port you would like the Mech-Viz project to take in Task 1. For example, if you set the value of **BranchPort** to **1**, the Mech-Viz project will proceed along out port 1 of Task 1.

MM_Camera ▸ PLC_1 [CPU 1211C DC/DC/DC] ▸ Program blocks ▸ ObtainPose_DB [DB2

Keep actual values | Snapshot | Copy snapshots to start val

ObtainPose_DB

	Name	Data type	Start value	Monitor value
1	Input			
2	BufferTime	Time	T#0ms	T#500MS
3	VisionStart	Bool	false	FALSE
4	VisObtainStart	Bool	false	FALSE
5	VizStart	Bool	false	FALSE
6	VizStop	Bool	false	FALSE
7	BranchSet	Bool	false	FALSE
8	VizObtainStart	Bool	false	FALSE
9	DataReady	Bool	false	FALSE
10	StatusCode	DInt	0	0
11	BranchName	USInt	0	1
12	BranchPort	USInt	0	1
13	SendPoseNum	USInt	0	0
14	Reset	Bool	false	FALSE
15	Output			
16	TargetPose	Array[0..19, 0..5] of DInt		

1

2

3. Set data type: in the **CameraIO** DB, change the monitor value of **REQ_POSE_TYPE** to **1**. This asks Mech-Viz to send joint positions (instead of TCP data).

MM_Camera ▸ PLC_1 [CPU 1211C DC/DC/DC] ▸ Program blocks ▸ CameraIO [DB1]

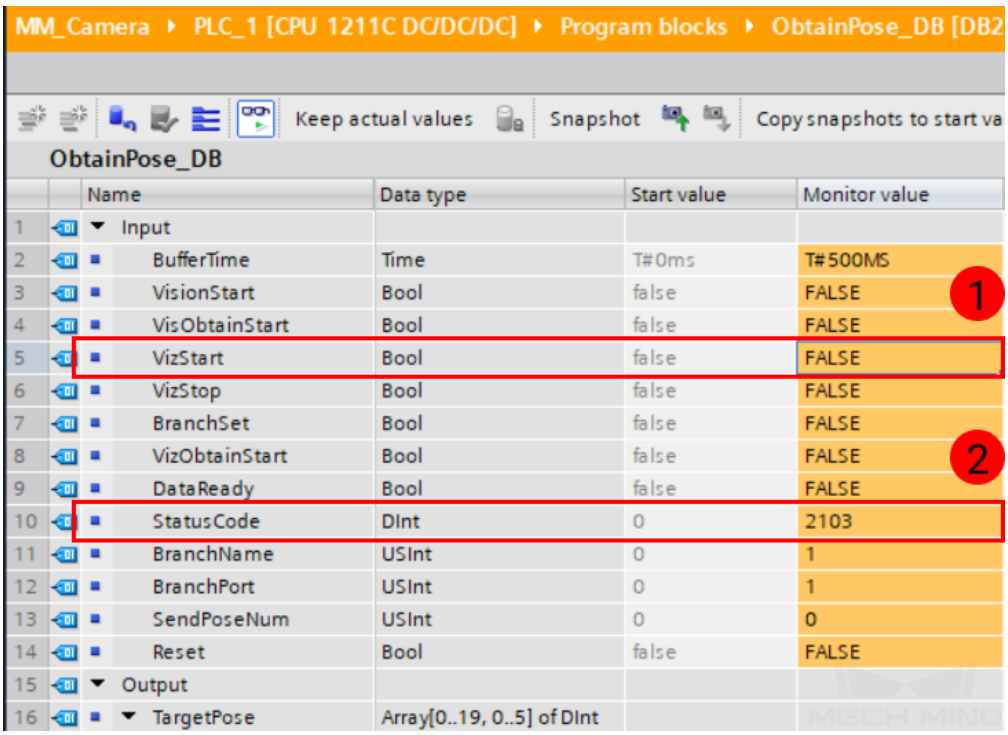
Keep actual values | Snapshot | Copy snapshots to start values | Load

CameraIO

	Name	Data type	Offset	Start value	Monitor value
1	Static				
2	MechMind	"CameraData1"	0.0		
3	ToCamera	Struct	0.0		
4	COMM_ENABLE	Bool	0.0	false	TRUE
5	TRIGGER	Bool	0.1	false	FALSE
6	RESET_EXPOSURE	Bool	0.2	false	FALSE
7	DATA_ACKNOWLEDGE	Bool	0.3	false	FALSE
8	CLEAR_NOTIFY	Bool	0.4	false	FALSE
9	CONTROL_INPUT_1	Bool	0.5	false	FALSE
10	CONTROL_INPUT_2	Bool	0.6	false	FALSE
11	CONTROL_INPUT_3	Bool	0.7	false	FALSE
12	CONTROL_INPUP_RESERVED	Array[0..7] of Bool	2.0		
13	CALIB_ROB_STATUS	USInt	4.0	0	0
14	ROBOT_POSE_TYPE	USInt	5.0	0	0
15	REQ_POSE_NUM	USInt	6.0	0	0
16	REQ_POSE_TYPE	USInt	7.0	0	1
17	VISION_PROJ_NUM	USInt	8.0	0	1
18	VISION_RECIPE_NUM	USInt	9.0	0	0

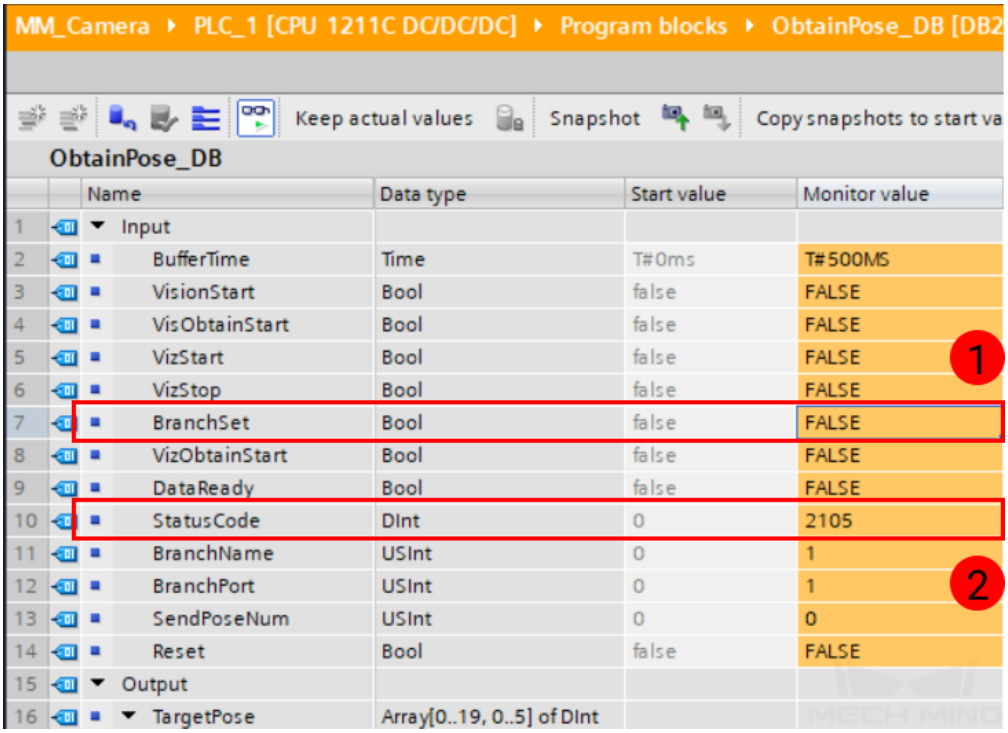
4. Start the Mech-Viz project: in the **ObtainPose_DB** DB, double-click on the monitor value of **VizStart** to change it to **TRUE**. Then, reset the monitor value to **FALSE**.

5. Check returned status code: check the monitor value of **StatusCode**. **2103** represents that the Mech-Viz project was started successfully. For other values, please refer to standard_interface_status_codes for the corresponding error.



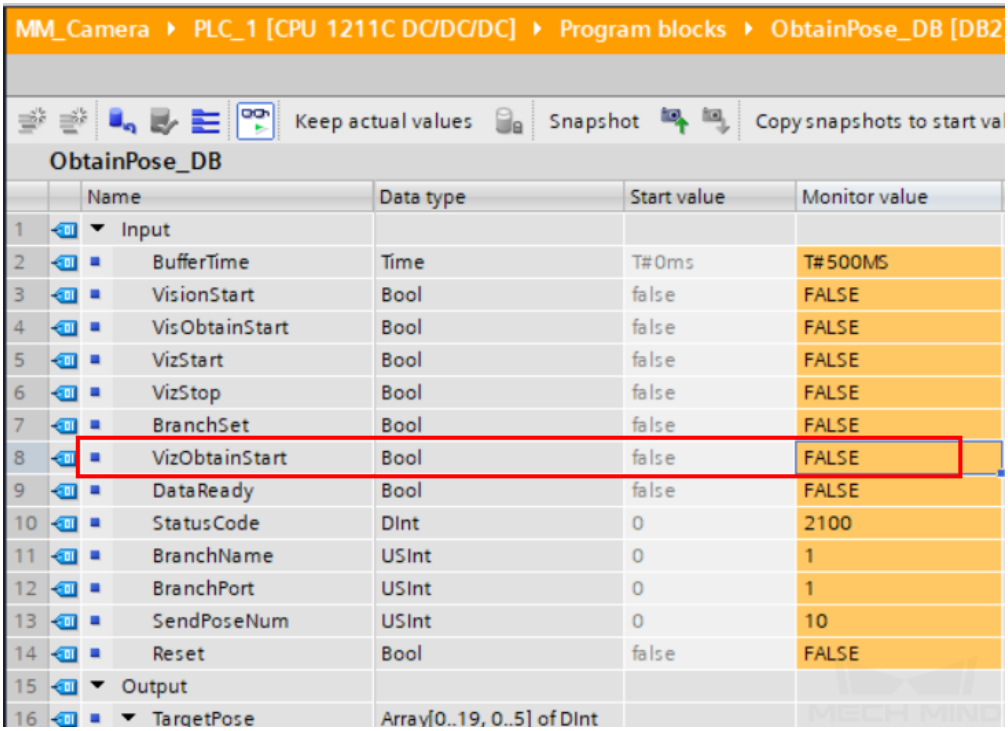
6. Select branch in the Mech-Viz project: in the **ObtainPose_DB** DB, double-click on the monitor value of **BranchSet** to change it to **TRUE**. Then, reset the monitor value to **FALSE**. This asks the Mech-Viz project to proceed along the branch set in step 2.

7. Check returned status code: check the monitor value of **StatusCode**. **2105** represents that the branch was selected successfully. For other values, please refer to standard_interface_status_codes for the corresponding error.



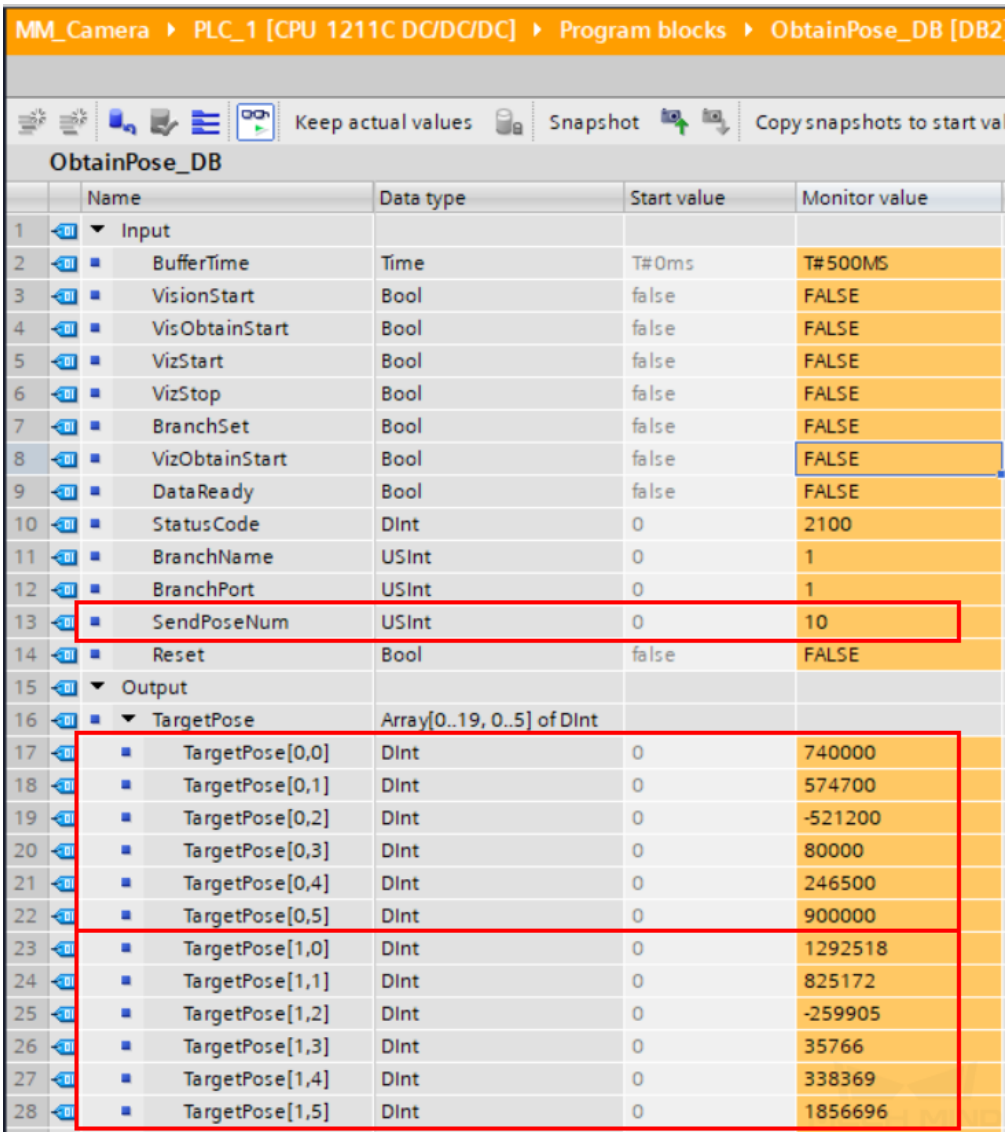
	Name	Data type	Start value	Monitor value
1	Input			
2	BufferTime	Time	T#0ms	T#500MS
3	VisionStart	Bool	false	FALSE
4	VisObtainStart	Bool	false	FALSE
5	VizStart	Bool	false	FALSE
6	VizStop	Bool	false	FALSE
7	BranchSet	Bool	false	FALSE
8	VizObtainStart	Bool	false	FALSE
9	DataReady	Bool	false	FALSE
10	StatusCode	DInt	0	2105
11	BranchName	USInt	0	1
12	BranchPort	USInt	0	1
13	SendPoseNum	USInt	0	0
14	Reset	Bool	false	FALSE
15	Output			
16	TargetPose	Array[0..19, 0..5] of DInt		

8. Obtain planned path from the Mech-Viz project: in the **ObtainPose_DB** DB, double-click on the monitor value of **VizObtainStart** to change it to **TRUE**. Then, reset the monitor value to **FALSE**.



	Name	Data type	Start value	Monitor value
1	Input			
2	BufferTime	Time	T#0ms	T#500MS
3	VisionStart	Bool	false	FALSE
4	VisObtainStart	Bool	false	FALSE
5	VizStart	Bool	false	FALSE
6	VizStop	Bool	false	FALSE
7	BranchSet	Bool	false	FALSE
8	VizObtainStart	Bool	false	FALSE
9	DataReady	Bool	false	FALSE
10	StatusCode	DInt	0	2100
11	BranchName	USInt	0	1
12	BranchPort	USInt	0	1
13	SendPoseNum	USInt	0	10
14	Reset	Bool	false	FALSE
15	Output			
16	TargetPose	Array[0..19, 0..5] of DInt		

9. Check the received target points: the monitor value of **SendPoseNum** shows how many target points were received, and the target points are stored in **TargetPose**. Divide the monitor values by 10000 to obtain the actual pose data.



	Name	Data type	Start value	Monitor value
1	Input			
2	BufferTime	Time	T#0ms	T#500MS
3	VisionStart	Bool	false	FALSE
4	VisObtainStart	Bool	false	FALSE
5	VizStart	Bool	false	FALSE
6	VizStop	Bool	false	FALSE
7	BranchSet	Bool	false	FALSE
8	VizObtainStart	Bool	false	FALSE
9	DataReady	Bool	false	FALSE
10	StatusCode	DInt	0	2100
11	BranchName	USInt	0	1
12	BranchPort	USInt	0	1
13	SendPoseNum	USInt	0	10
14	Reset	Bool	false	FALSE
15	Output			
16	TargetPose	Array[0..19, 0..5] of DInt		
17	TargetPose[0,0]	DInt	0	740000
18	TargetPose[0,1]	DInt	0	574700
19	TargetPose[0,2]	DInt	0	-521200
20	TargetPose[0,3]	DInt	0	80000
21	TargetPose[0,4]	DInt	0	246500
22	TargetPose[0,5]	DInt	0	900000
23	TargetPose[1,0]	DInt	0	1292518
24	TargetPose[1,1]	DInt	0	825172
25	TargetPose[1,2]	DInt	0	-259905
26	TargetPose[1,3]	DInt	0	35766
27	TargetPose[1,4]	DInt	0	338369
28	TargetPose[1,5]	DInt	0	1856696