HIWIN® MIKROSYSTEM

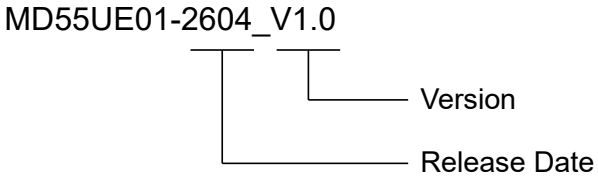
Application Note

E Series EtherCAT Drive Complete Setup with HIMC3 iA Studio

www.hiwinmikro.tw
MD55UE01-2604_V1.0

Revision History

The version of the manual is also indicated on the bottom of the front cover.



Release Date	Version	Applicable Product	Revision Contents
Apr. 20th, 2026	1.0	E Series EtherCAT Drive	First edition.

Related Documents

Through related documents, users can quickly understand the positioning of this manual and the correlation between manuals and products. Go to HIWIN MIKROSYSTEM's official website → Download → Manual Overview for details (https://www.hiwinmikro.tw/Downloads/ManualOverview_EN.htm).

Preface

This manual explains the operation of software iA Studio when E series EtherCAT drive is used with HIMC3 motion controller. For detailed information on E series servo drive, please refer to the related user manuals.

Specifications of Software/Hardware

Name	Version of Software/Firmware
E1 Series EtherCAT Drive	Software (Thunder): 1.14.11.0 or above Firmware: 2.14.9 or above ESI file: HIWIN_MIKROSYSTEM_EDxF_20250725 or above
E2 Series EtherCAT Drive	Software (Thunder): 1.14.11.0 or above Firmware: 3.14.9 / 4.14.9 or above ESI file: HIWIN_MIKROSYSTEM_EDxF_20250725 or above
HIMC3 Motion Controller	Software (iA Studio): 3.3.0 or above Firmware: 3.3.0 or above

Table of Contents

1. Communication and Module Setup

1.1 Connect to controller

In “Connection Setting” window, users can connect iA Studio to controller via specified communication type.

1.1.1 Connection setting

1. Click **Controller** on the menu bar and click **Connection Setting** to open “Connection Setting” window.

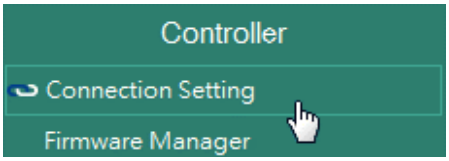


Figure 1.1.1.1 Connection Setting

2. Select the port (**CN3** or **CN4**) for connecting the controller to the computer.

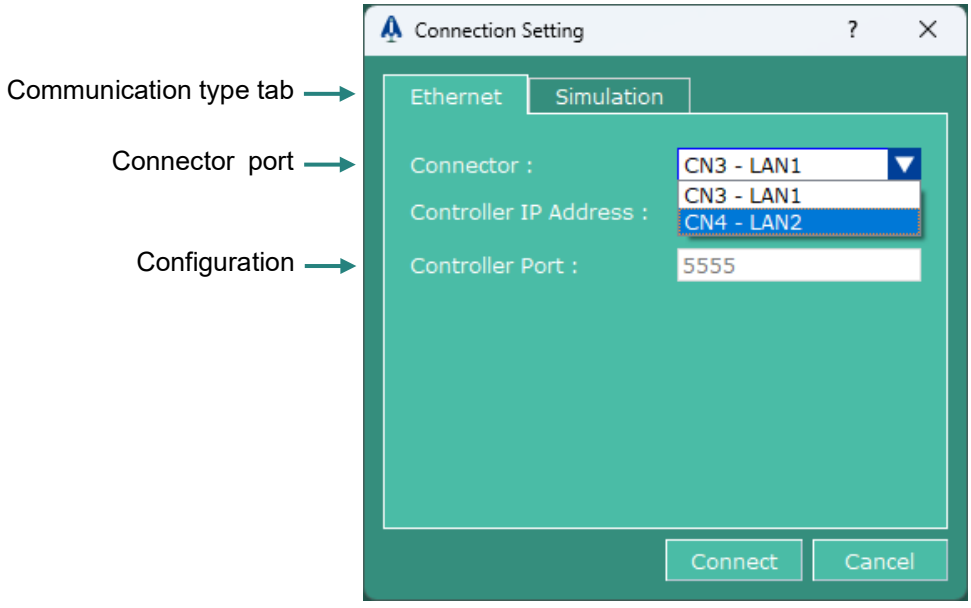


Figure 1.1.1.2 “Connection Setting” window: EtherCAT

3. If using CN3, the computer's network card and the controller must be configured within the same IP subnet, e.g., 192.168.0.XXX. If using CN4, IP is fixed at 169.254.188.20.

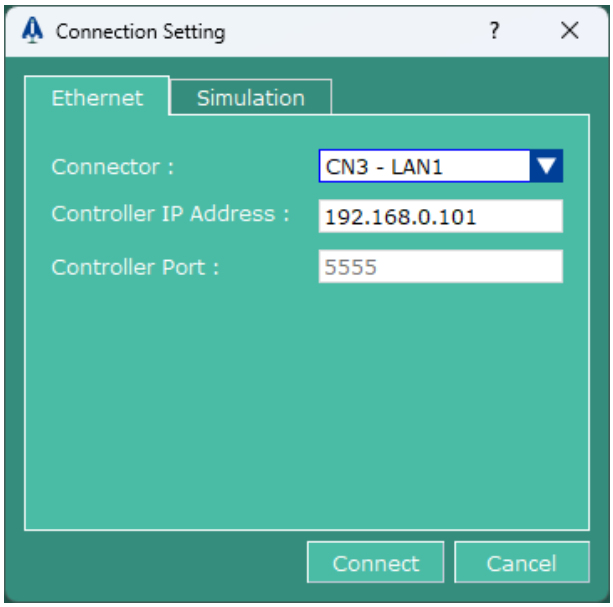


Figure 1.1.1.3 CN3 - Self-defined IP connection

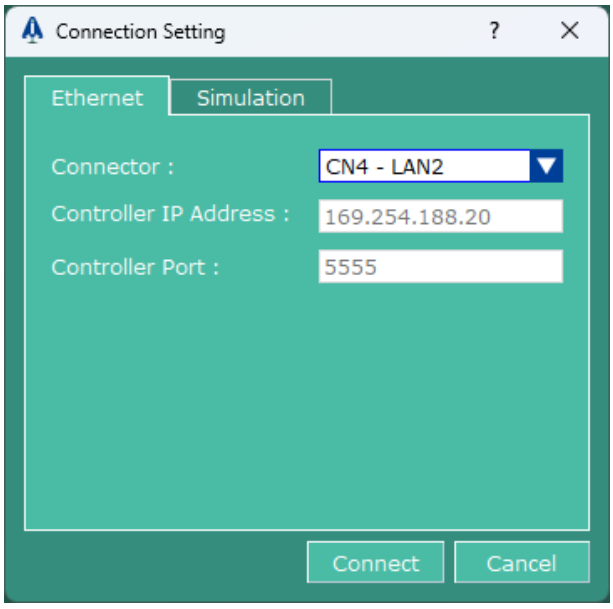


Figure 1.1.1.4 CN4 - Fixed IP connection

Table 1.1.1.1 "Connection Setting" window

Communication Type Tab	Description
Ethernet	Connect to controller via TCP/IP.
Simulation	Connect to simulator.

1.1.2 Connect to controller via Ethernet

Controller can be connected via Ethernet. Follow the steps below to establish connection.

1. Select **Ethernet** tab in “Connection Setting” window.
2. Enter controller IP address and IP port.
3. Click **Connect** button to initialize the connection. A pop-up window will appear to indicate the connecting progress.

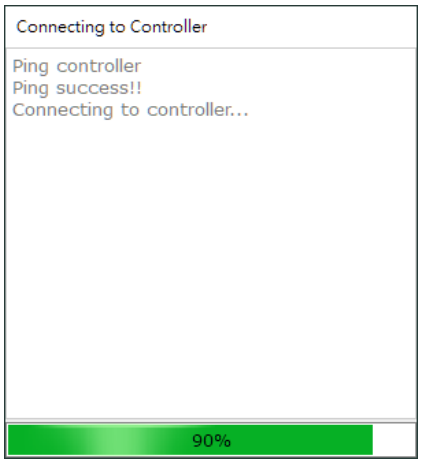


Figure 1.1.2.1 Connecting progress pop-up window

“Connection Setting” window and pop-up window will close automatically after connection is successfully established. If connection cannot be established, an error dialog will appear. When it appears, please check if the communication cable is properly connected to the controller.

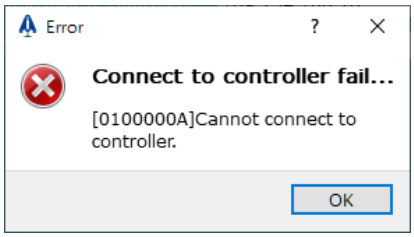


Figure 1.1.2.2 Fail to connect to the controller

2. Parameters Setup

2.1 Configure controller

1. Click **Project** on the menu bar and click **Configuration Wizard**.

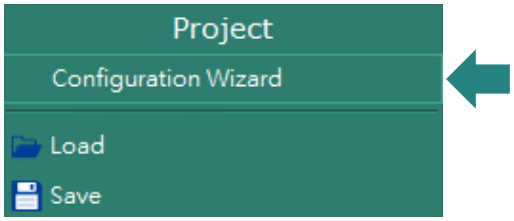


Figure 2.1.1 Configuration Wizard

2. Click  to scan the drive(s).

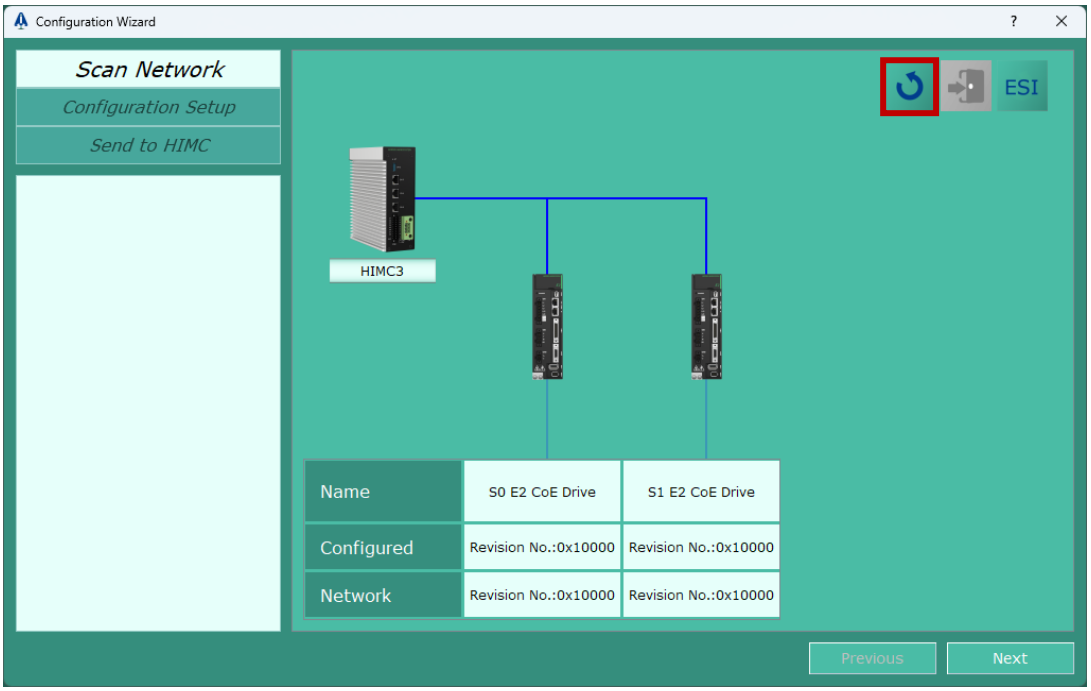


Figure 2.1.2 Scan the drive(s)

2.2 Install ESI file

Important

(1) ESI filename for E series servo drive: HIWIN_MIKROSYSTEM_EDxF_yyyymmdd.xml

(2) E series servo drive's ESI file can be obtained in the installation path of drive's Thunder software (**Thunder/doc/ESI Files**).

1. Click ESI in "Configuration Wizard" window.

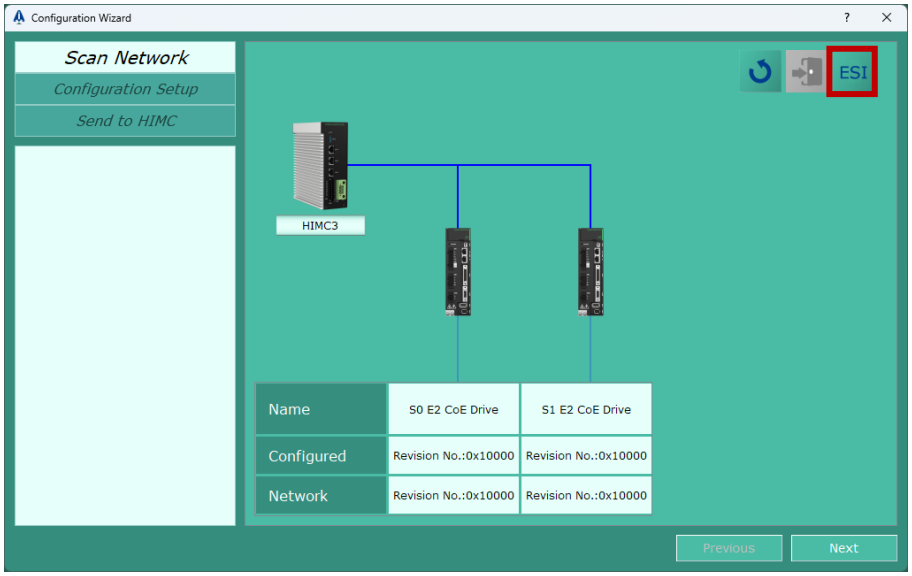


Figure 2.2.1 Set up ESI file

2. Click **Add** to add ESI file corresponding to the firmware version.

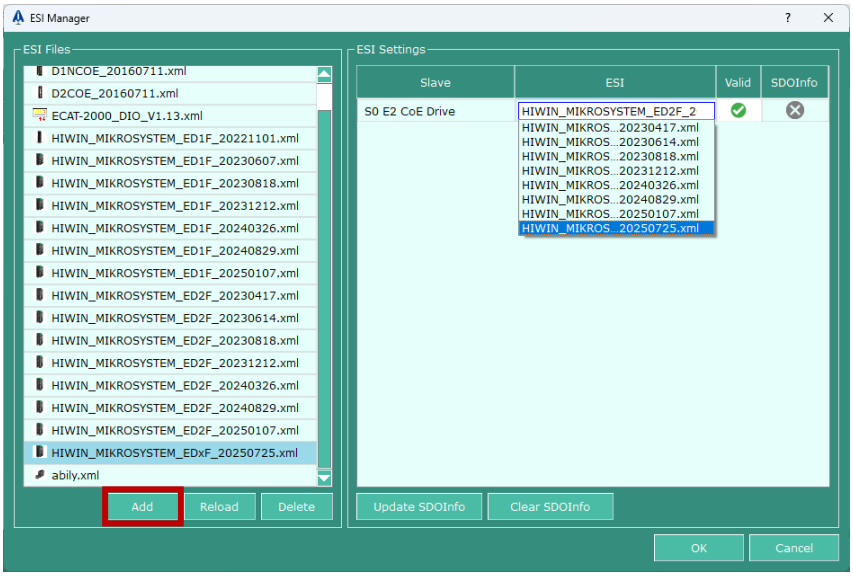


Figure 2.2.2 Add ESI file

3. Select the ESI file (.xml) to be loaded and click **Load**.

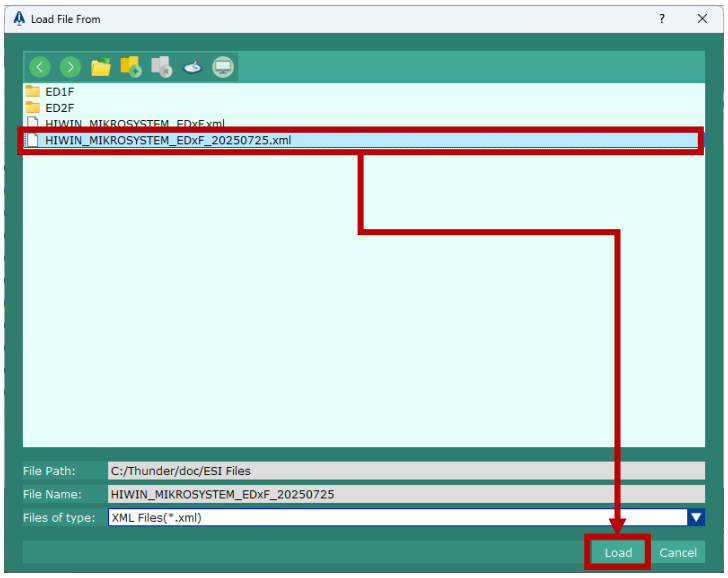


Figure 2.2.3 Load ESI file

4. Select the desired ESI file (.xml) version and click **OK** to complete the installation.

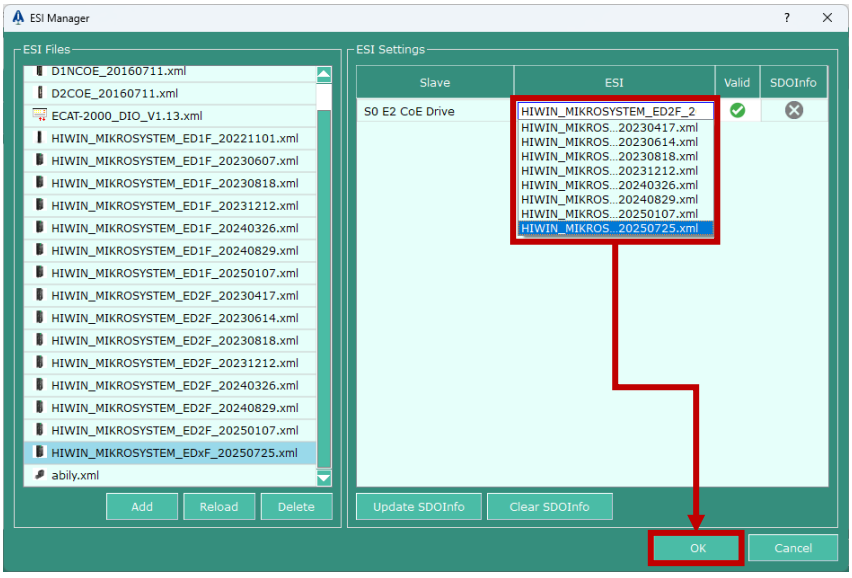


Figure 2.2.4 Select ESI file

2.3 Set up motion control axis

1. Click  in front of the slave axis in “Configuration Setup” tab.

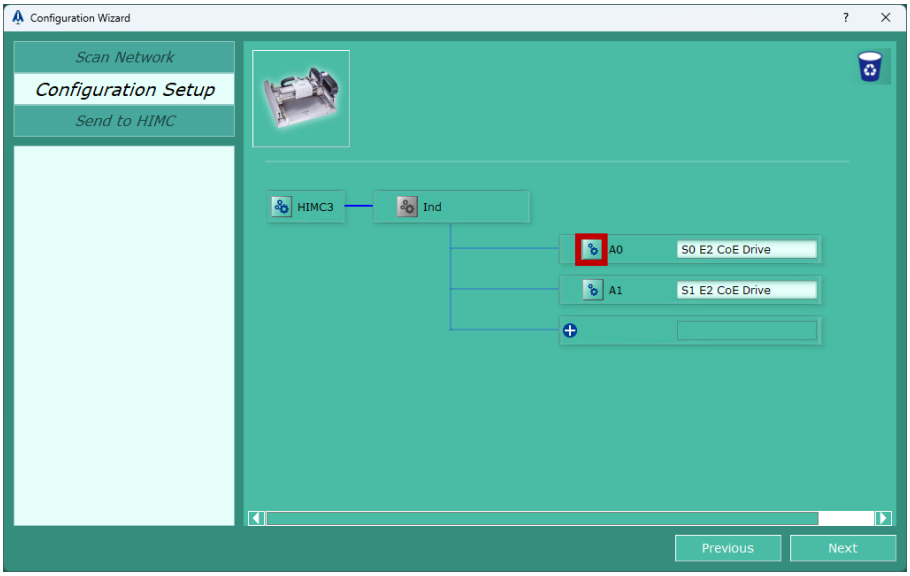


Figure 2.3.1

2. Set axis-related parameters in “Axis” tab, such as operation mode, motor type, drive setting, safety setting, etc.

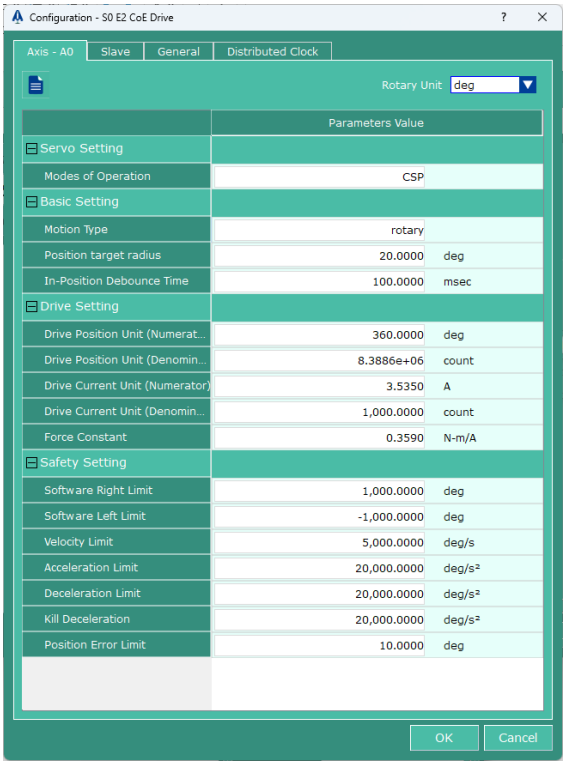


Figure 2.3.2

3. Check if Motion Type is correct.

Important

For non-rotary direct-drive architectures (including applications combining rotary motors and screws), the screw feed constant must be set via the electronic gear ratio in the drive and then converted to linear control units. Therefore, Motion Type must be set to Linear when configuring the controller to prevent control anomalies.

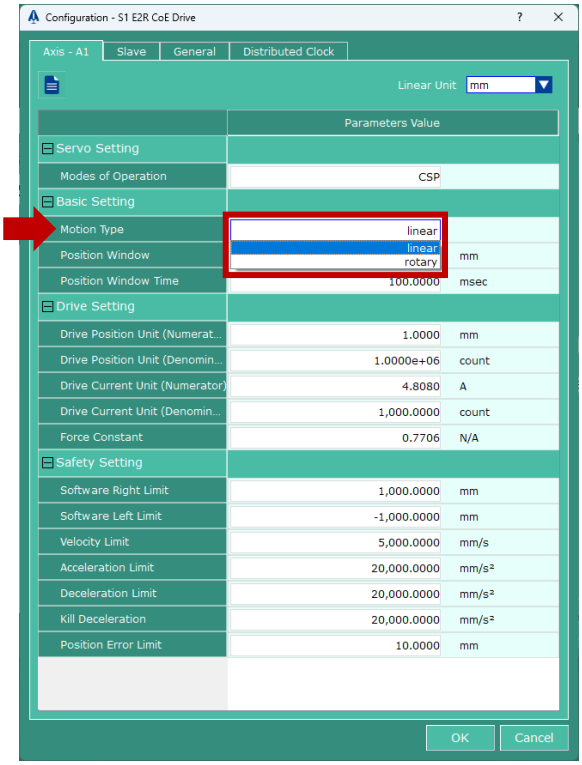


Figure 2.3.3

4. Check if the control units of the drive and controller are the same.

Important

If the electronic gear ratio of the equipment mechanism is already set in the drive, simply confirm that the control units of the drive and controller are the same. If the electronic gear ratio of the equipment mechanism is set to 1:1 in the drive, the conversion of the mechanism's electronic gear ratio must be set in the controller.

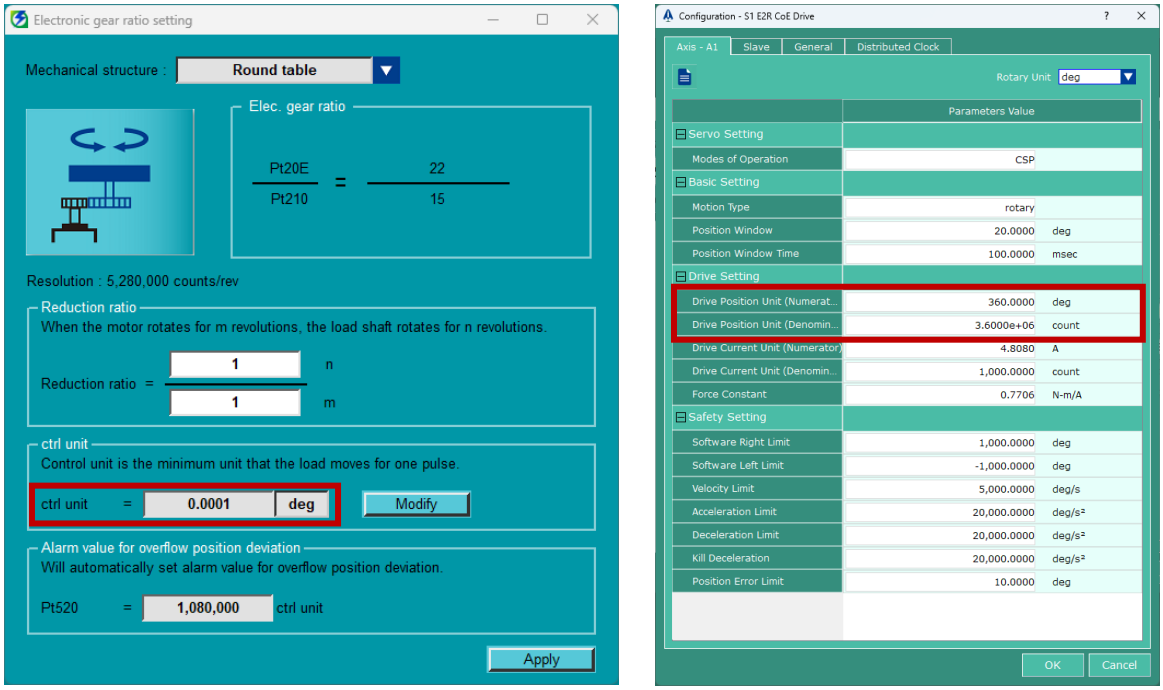


Figure 2.3.4

5. Switch to “Send to HIMC” tab. After confirming the modified parameters are correct, click **Send to HIMC** to complete the setup.

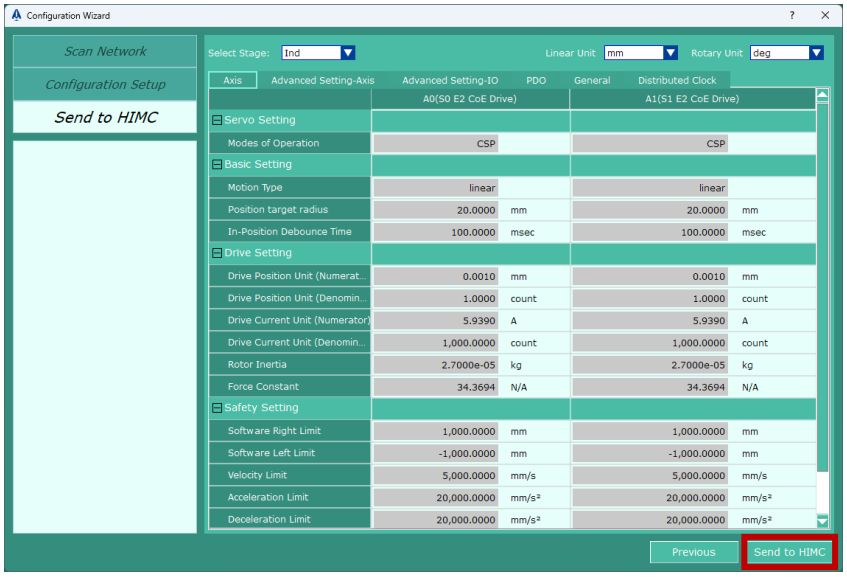


Figure 2.3.5

2.4 PDO mapping

Important

(1) After clicking any PDO group, the default PDO objects for that group will be displayed on the right side of “Edit PDO Map Settings” window.

(2) Click **Add PDO Entry** to add other objects to the group; click **Delete PDO Entry** to remove existing objects from the group.

(3) The maximum number of RxPDO and TxPDO objects for E series servo drive is 10 each.

1. After switching to “Slave” tab, click  of PDO Mapping.

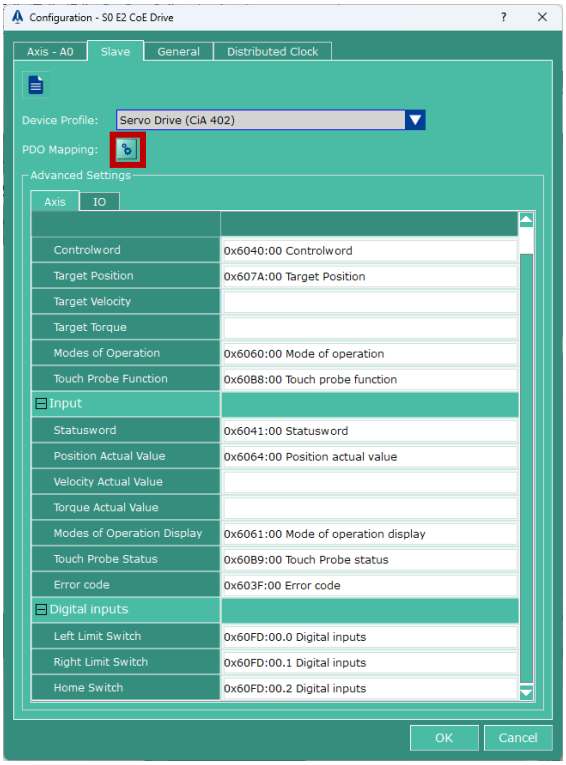


Figure 2.4.1

2. Select the desired default PDO. To adjust the setting, click  in the upper right corner.

   After completing the setting, click **OK**.

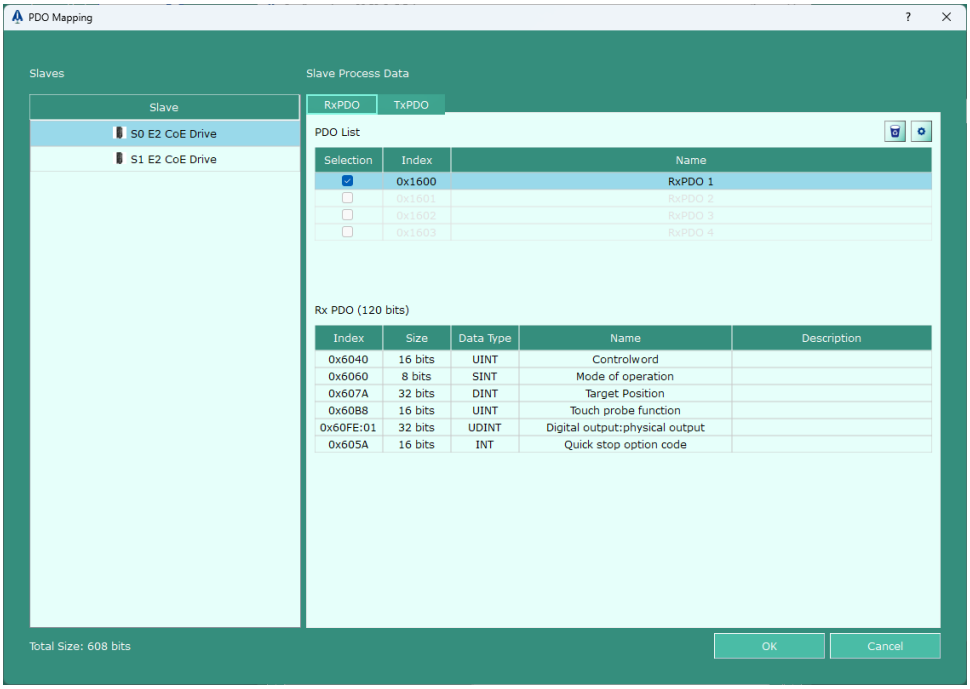


Figure 2.4.2 Select default PDO

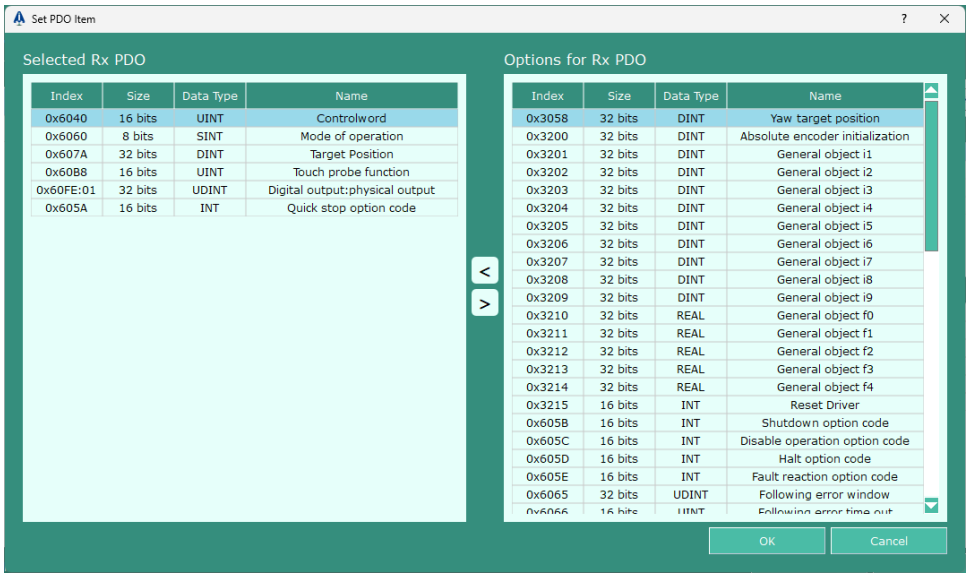


Figure 2.4.3 Select self-defined PDO

3. Finally, perform PDO mapping in “Slave” tab.

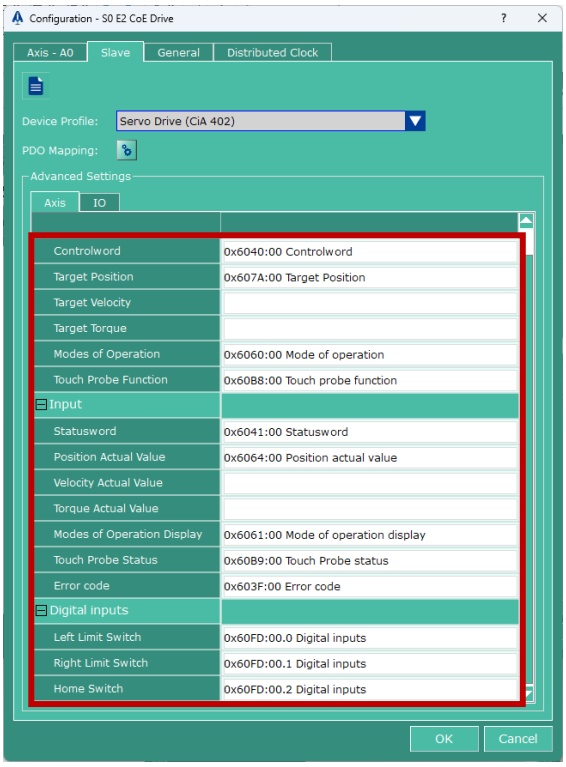


Figure 2.4.4

3. Test Run

This chapter will introduce how to perform test run by iA Studio interface and by the simple programs of HMPL.

Important

This manual only introduces basic functions. For other functions, please refer to "HIMC Series iA Studio User Guide" and "HIMC Series HMPL User Guide."

3.1 Enable

1. Click **Tools** on the menu bar and click **Motion Manager**.

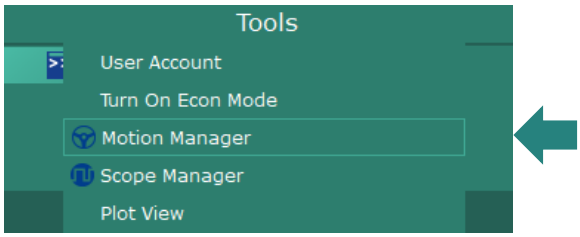


Figure 3.1.1 Motion Manager

2. Click **Enable**/**Disable** in "Motion Manager" window to enable/disable the motor. Observe the indicator light of Enable to check whether the operation is successful.

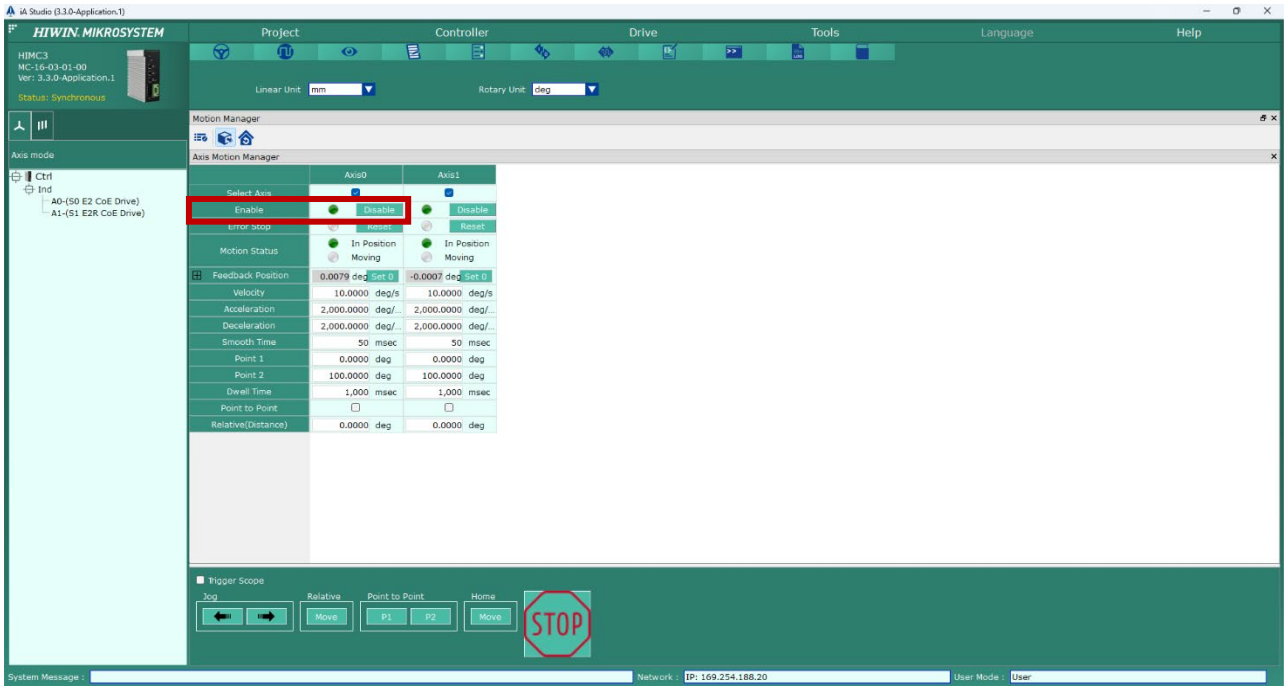


Figure 3.1.2 "Motion Manager" window - Enable/Disable

3.2 Homing

1. Click  in “Motion Manager” window to switch to the homing function window.

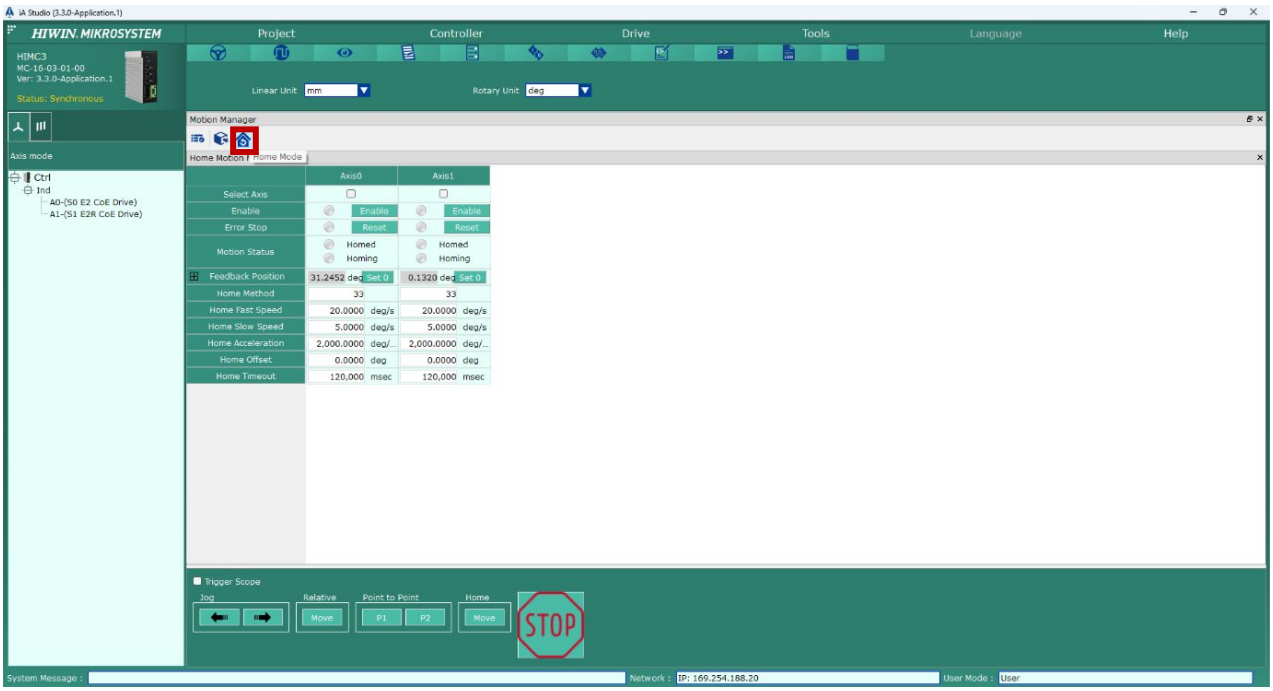


Figure 3.2.1 “Motion Manager” window - Homing

2. Set homing parameters based on application requirements, including homing method, velocity, acceleration, offset, and so on.

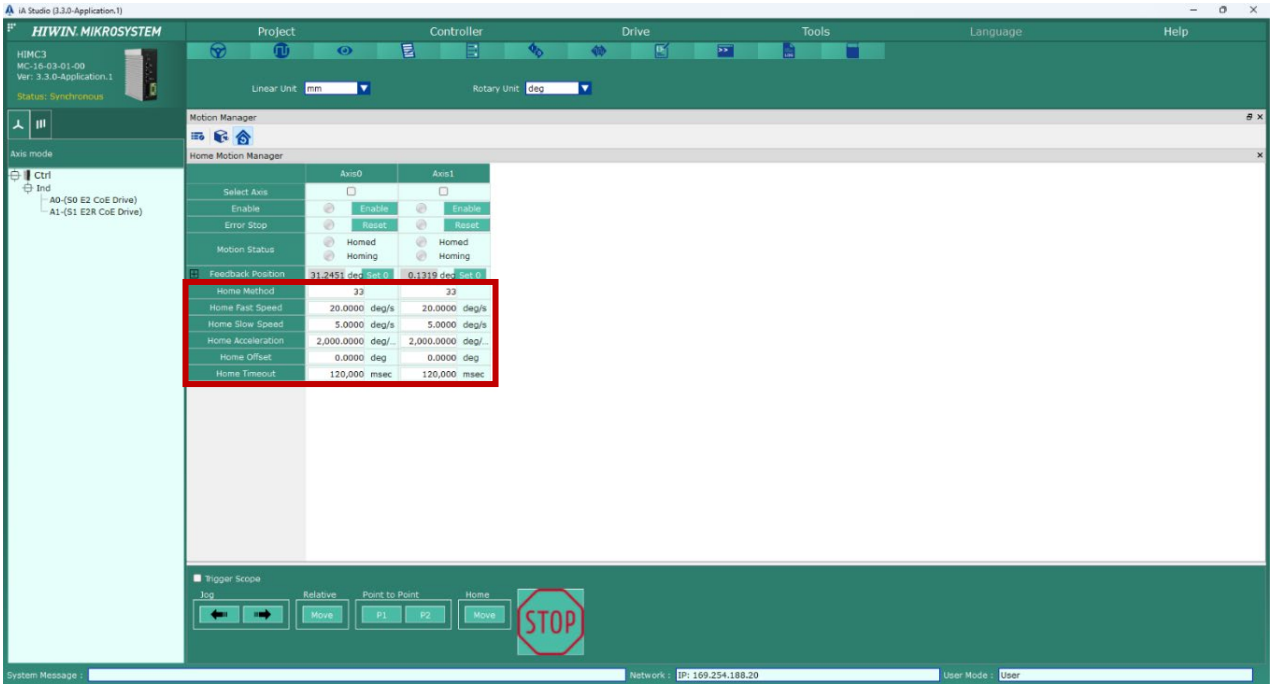


Figure 3.2.2 Homing conditions setting

3. Select the axis to be homed, click **Enable**, and click **Move** of Home.

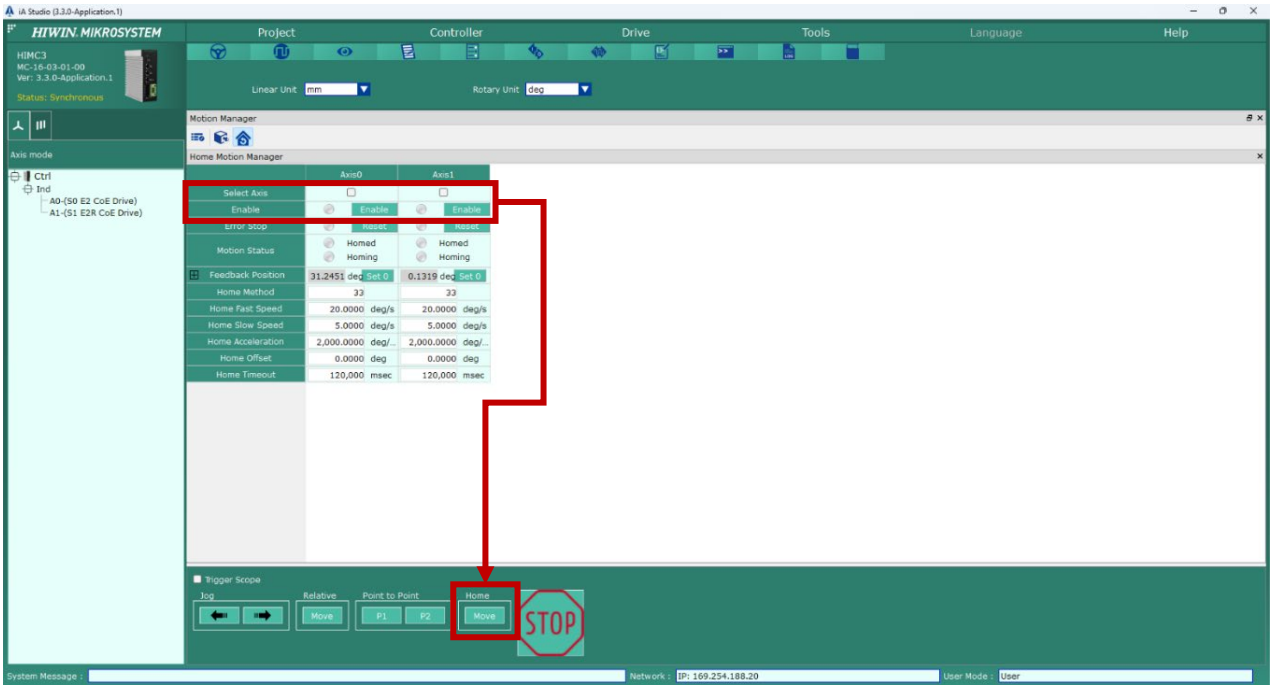


Figure 3.2.3 Enable homing

4. When Homed lights up in green, homing is done.

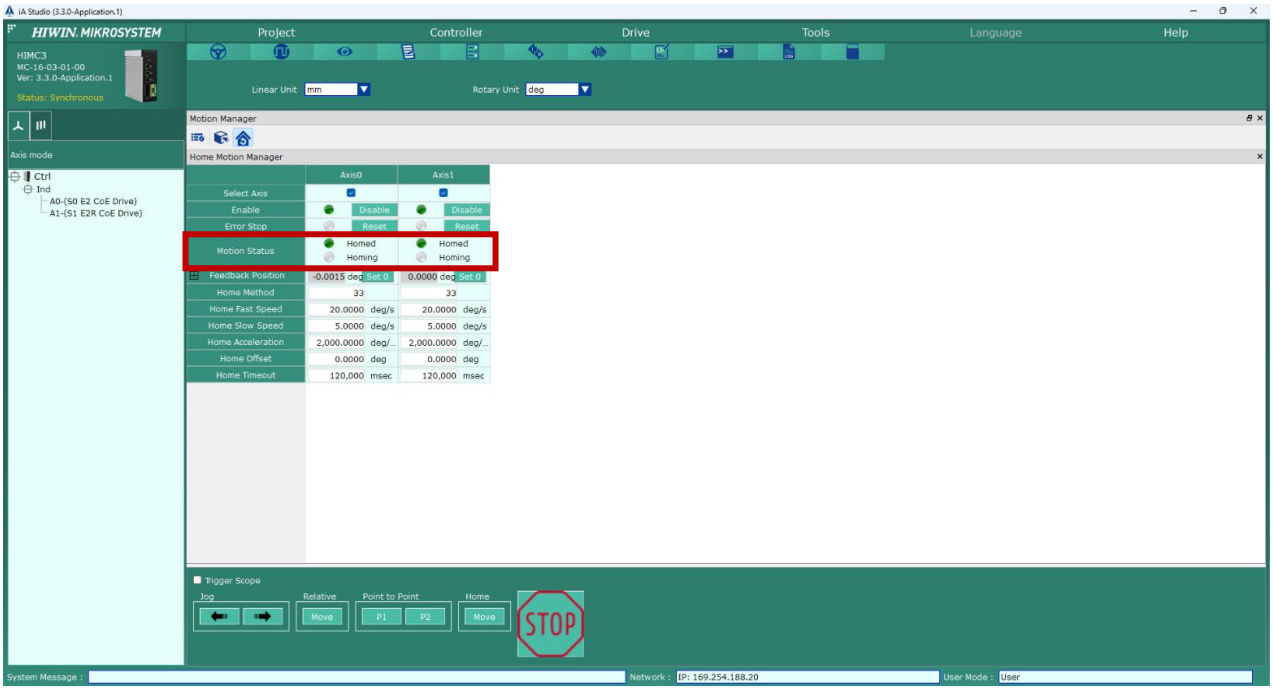


Figure 3.2.4 Homing completed

3.3 Jog

Click **the arrows of Jog** in “Motion Manager” window to perform forward or reverse jog.

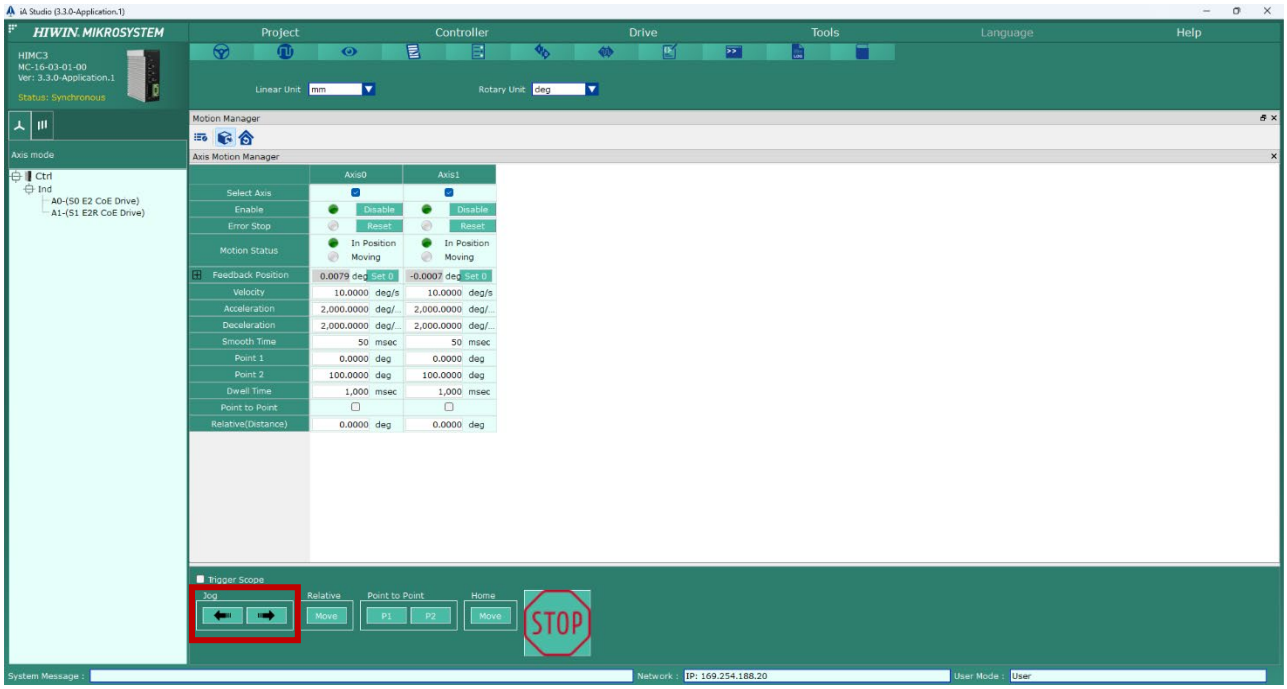


Figure 3.3.1 “Motion Manager” window - Jog

3.4 Relative move

In “Motion Manager” window, set **Relative(Distance)** and click **Move** of Relative to perform relative move.

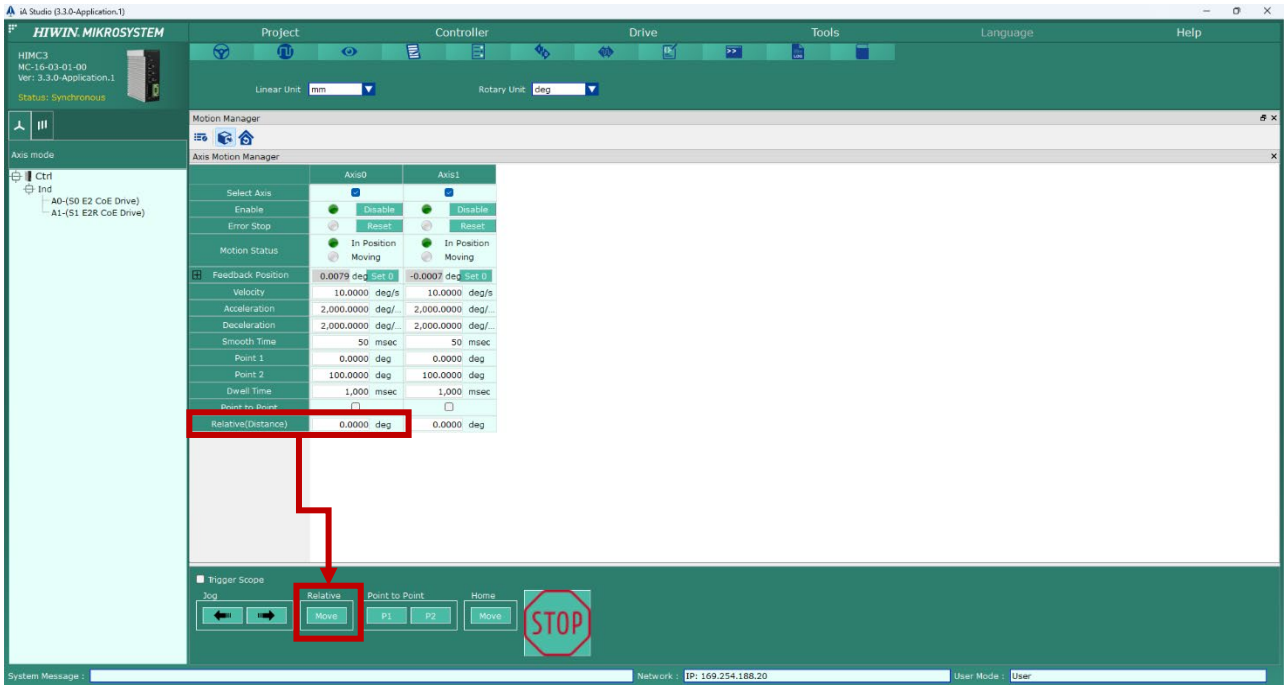


Figure 3.4.1 “Motion Manager” window - Relative move

3.5 Point to Point motion

1. Set **Point 1**, **Point 2**, and **Dwell Time** in “Motion Manager” window.
2. Click **P1** or **P2** of Point to Point to move the motor to the position of Point 1 or Point 2.
3. Check **Point to Point** and click P1 or P2 to make the motor repeatedly move between the positions of Point 1 and Point 2.

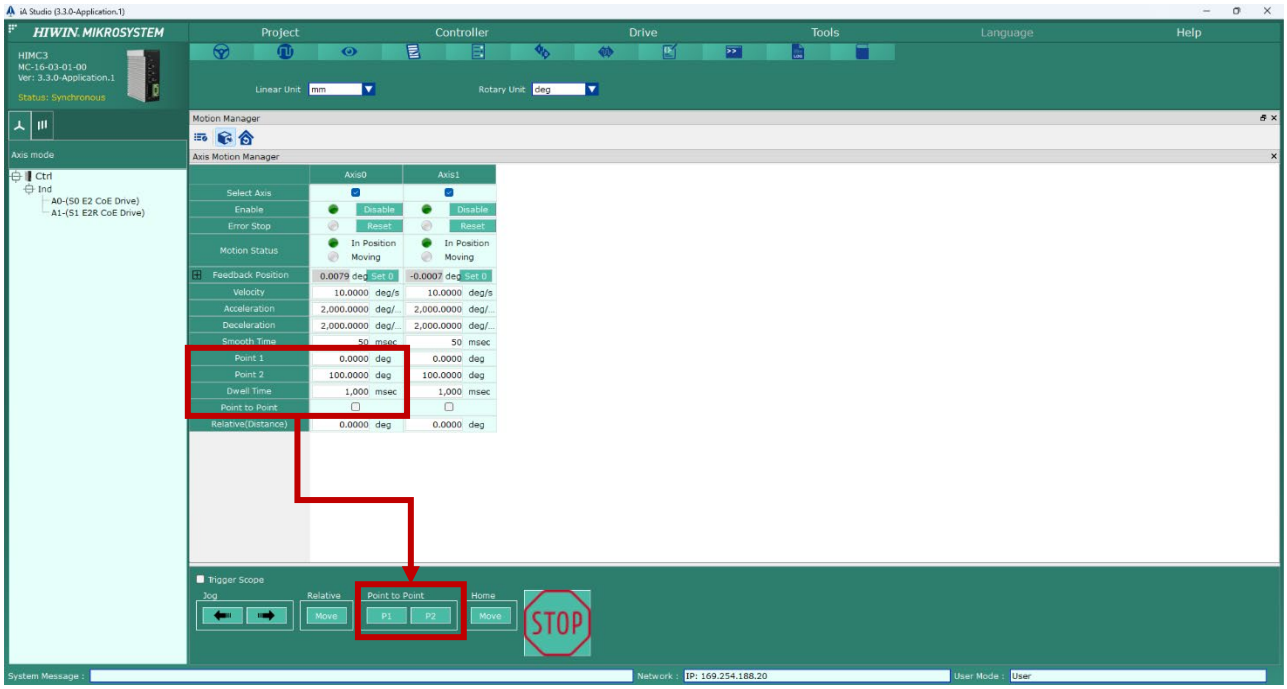


Figure 3.5.1 “Motion Manager” window - Point to Point motion

3.6 HMPL Editor

1. Click **Tools** on the menu bar and click **HMPL Editor** to open HMPL editing window.

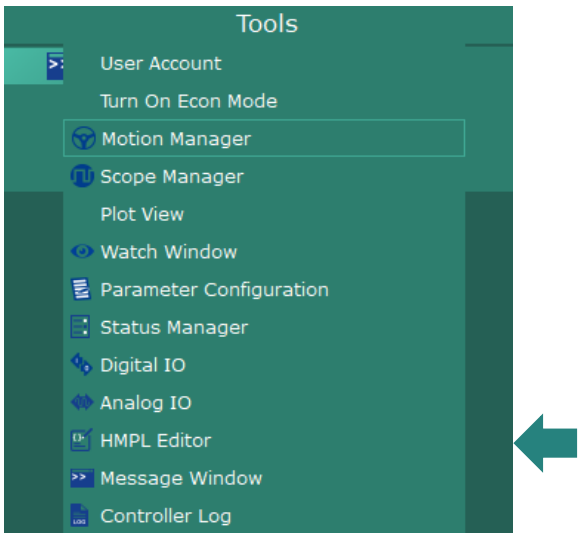


Figure 3.6.1 HMPL Editor

2. Double-click the task to open the program editing page.

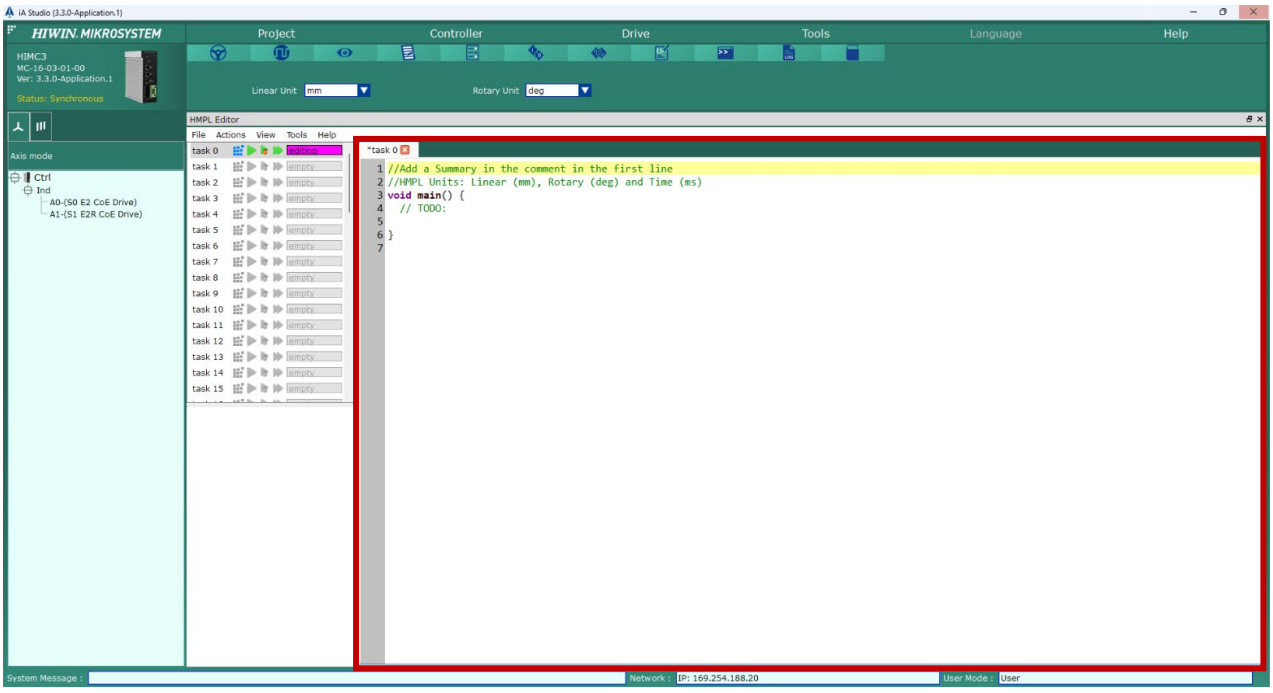


Figure 3.6.2 Program editing

3. Click  (F7) to compile the written program.

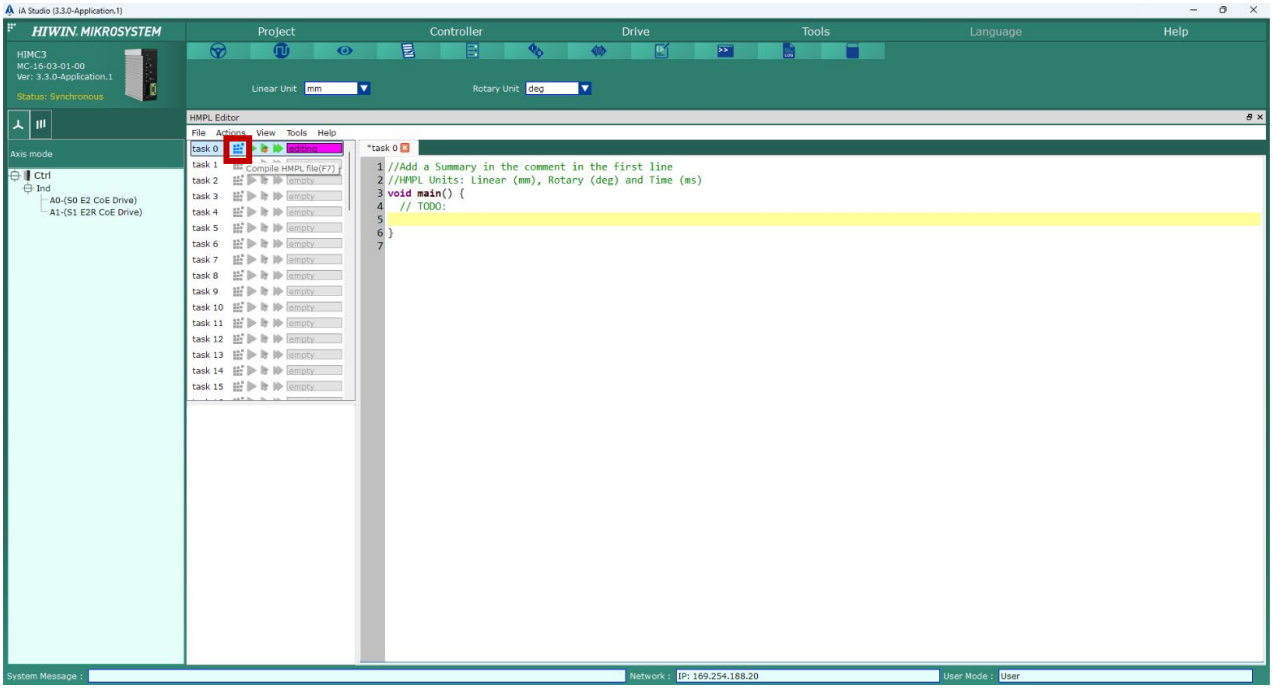


Figure 3.6.3 Program compiling

4. Click  to save the task to the controller.

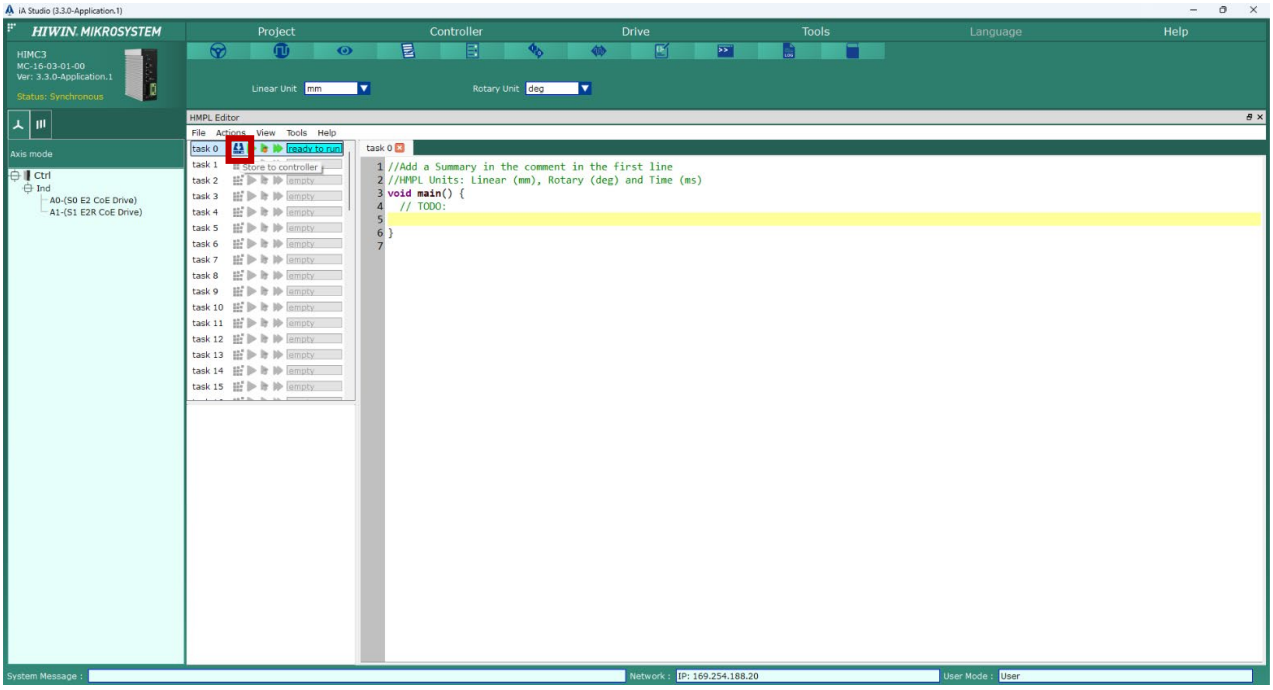


Figure 3.6.4 Program written to controller

5. Click  (Ctrl+F5) to execute the program in the task.

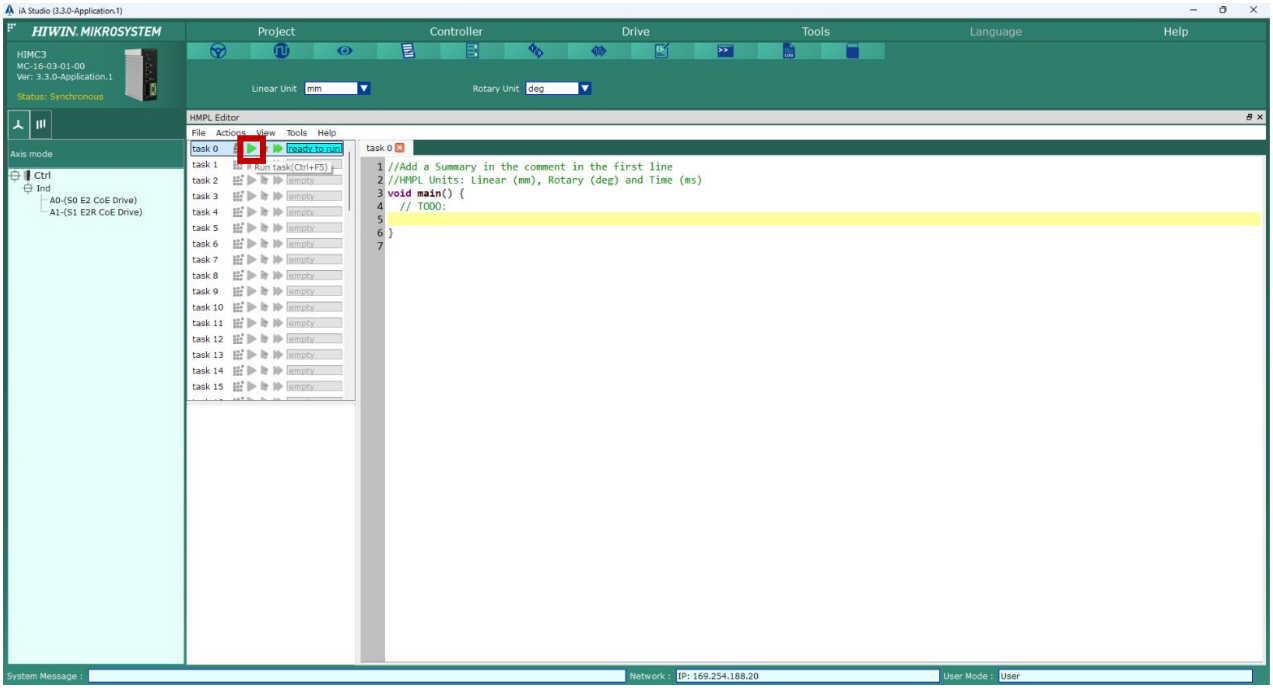


Figure 3.6.5 Program executing

3.6.1 Point to Point motion

1. Start a new task.
2. Write a point-to-point program in the task (set P1/P2 position, velocity, acceleration/deceleration, etc.).
3. Compile the program and write it to the controller.
4. Execute the program.

```
int axis_x = 0;  //  Axis index
int P1 = 0;  //  P1 position  Unit: linear (mm), rotary (deg)
int P2 = 30; //  P2 position  Unit: linear (mm), rotary (deg)
int Dwell_Time = 1000;  //  Dwell time (ms)

void main()
{
    SetVel(axis_x, 50);  //  Set velocity  Unit: linear (mm/s), rotary (deg/s)
    SetAcc(axis_x, 2000);  //  Set acceleration  Unit: linear (mm/s^2), rotary (deg/s^2)
    SetDec(axis_x, 2000);  //  Set deceleration  Unit: linear (mm/s^2), rotary (deg/s^2)

    Enable(axis_x);  //  Enable the axis
    Till(IsEnabled(axis_x));  //  Wait until it is enabled
    Sleep(500);

    while(IsEnabled(axis_x)){
    MoveAbs(axis_x, P1);    //  Move to P1
    Till(IsInPos(axis_x));  //  Wait until it is in-position
    Sleep(Dwell_Time);      //  Dwell time
    MoveAbs(axis_x, P2);    //  Move to P2
    Till(IsInPos(axis_x)); //  Wait until it is in-position
    Sleep(Dwell_Time);     //  Dwell time
    }
}
```

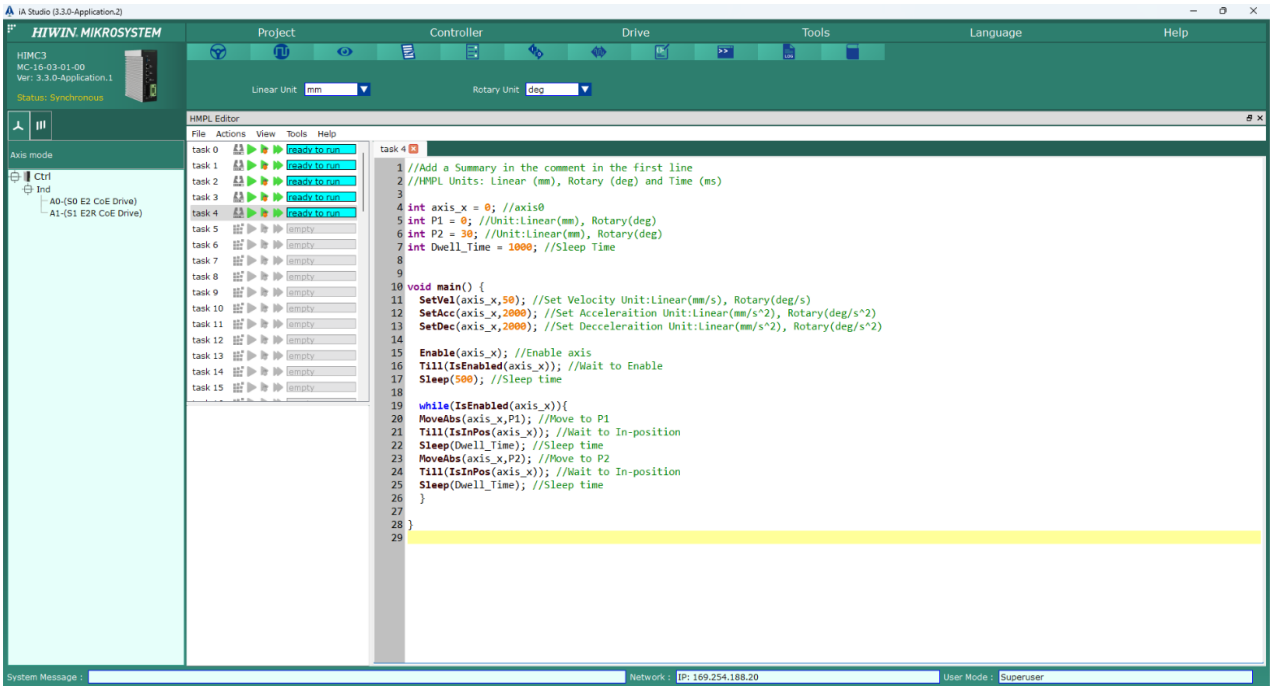


Figure 3.6.1.1 Point-to-point program written to controller

3.6.2 Homing

Important

Users can load the example program from the installation path of iA Studio:

C:\Program Files (x86)\HIWIN MIKROSYSTEM\iA Studio (version)\examples\HMPL

1. Start a new task.
2. Write a homing program in the task (set homing method, velocity, etc.).
3. Compile the program and write it to the controller.
4. Execute the program.

```
void main()
{
    int axis_id = 0;  //  Axis index
    int home_method = 33;  //  Homing method
    double fast_vel = 20;  //  Homing velocity (search for Limit Switch)  Unit: linear (mm/s), rotary (deg/s)
    double slow_vel = 2;   //  Homing velocity (search for Index)  Unit: linear (mm/s), rotary (deg/s)
    double acc = 2000;  //  Homing acceleration  Unit: linear (mm/s^2), rotary (deg/s^2)
    double home_offsets = 0;  //  Home offset  Unit: linear (mm), rotary (deg)
    int time_out = 10000;  //  Timeout (ms)

    Enable(axis_id);
    Till(IsEnabled(axis_id));

    SetHomedStatus(axis_id, false);  //  Set homing status
    SetHomeMethod(axis_id, home_method);  //  Set homing method
    SetHomeSwitchVel(axis_id, fast_vel);  //  Set homing velocity (search for Limit Switch)
    SetHomeZeroVel(axis_id, slow_vel);    //  Set homing velocity (search for Index)
    SetHomeAcc(axis_id, acc);  //  Set homing acceleration
    SetHomeOffset(axis_id, home_offsets);  //  Set home offset
    SetHomeTimeout(axis_id, time_out);  //  Set timeout
    int result = MoveHome(axis_id);
    Till(!IsHoming(axis_id));  //  Wait until homing is done

    if (!IsHoming(axis_id)) {
        Print("Home success.");
```

```
    } else {
        Print("Home fail:%d.", result);
    }
}
```

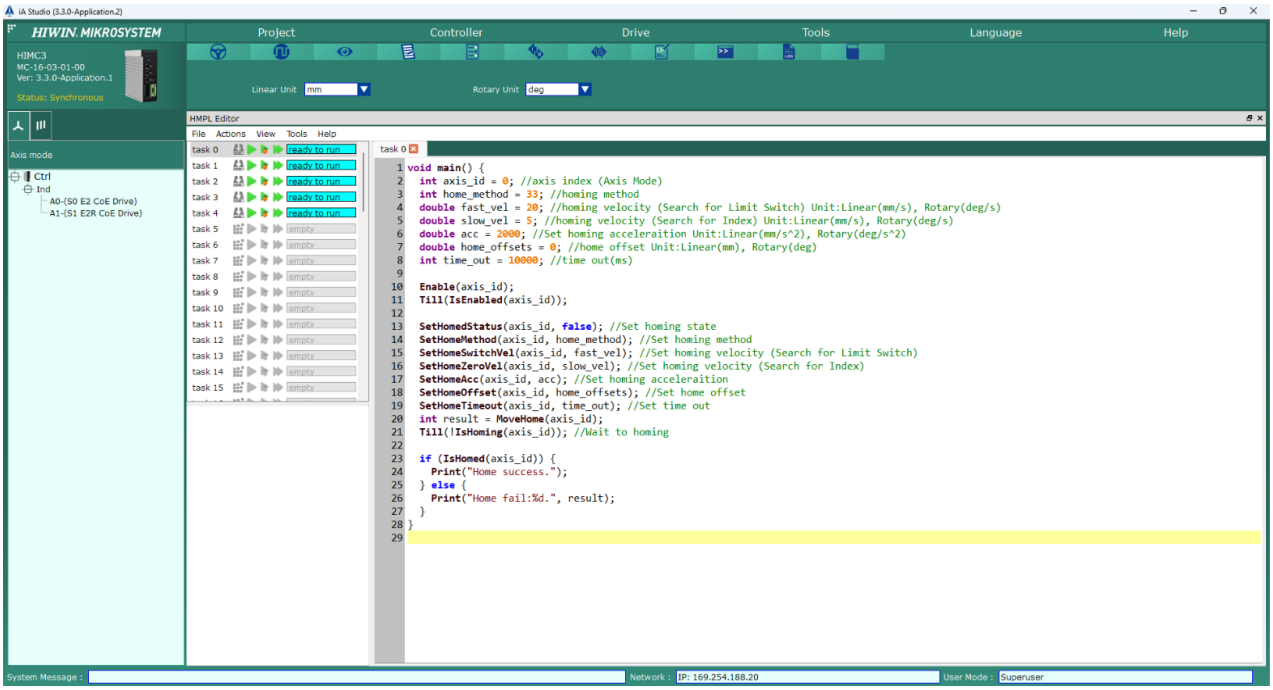


Figure 3.6.2.1 Homing program written to controller

4. Other Application Settings

4.1 Error compensation function setting

Tolerances will be generated during installation and movement of the mechanism. To improve accuracy, users can utilize measurement tools such as laser measuring instruments, dial indicators, and calibration plates to obtain error values and perform error compensation. For detailed explanations of the error compensation function, error compensation functions, and example programs, please refer to chapter 14 **Dynamic Error Compensation functions** in "HIMC Series HMPL User Guide."

4.1.1 1D error compensation

1D error compensation performs single-axis compensation by referencing one axis. The following example demonstrates Y-axis compensation by referencing X axis with a start compensation position of -100, an interval of 200, and 5 compensation points. The sequential compensation values are -0.2, 0.1, -0.3, 0.4, and -0.1. For detailed program example, refer to Example 1: One-dimensional dynamic error compensation in chapter 14 **Dynamic Error Compensation functions** of "HIMC Series HMPL User Guide."

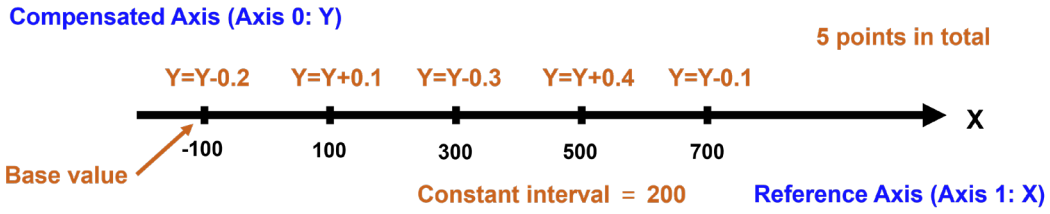


Figure 4.1.1.1 Position diagram of 1D error compensation

Set up and enable compensation via HMPL:

```
void main() {
    //  Set up error map
    double data[5] = {-0.2, 0.1, -0.3, 0.4, -0.1};
    SetUserTable(1, 5, data);  //  Write data to user table

    SetupComp(
      0,    //  axis to be compensated
      1,    //  start index in user table
      -100, //  base value
      200,  //  interval
      5,    //  number of points (base data included)
```

```
      1    //  reference axis (input)
    );
    EnableComp(0);  //  Enable compensation on axis 0
    Enable(0);  //  Enable axis 0 to activate compensation
}
```

Disable compensation via HMPL:

```
void main() {
    Disable(0);  //  Disable axis 0
    Till(!IsEnabled(0));
    DisableComp(0);  //  Disable compensation on axis 0
}
```

4.1.2 2D error compensation

2D error compensation involves simultaneously referencing two axes, commonly seen in XYZ planes, where X-axis, Y-axis or Z-axis compensation is performed by referencing both X and Y axes. The following example demonstrates Y-axis compensation with simultaneous reference to both X and Y axes. For detailed program example, refer to Example 2: Two-dimensional dynamic error compensation in chapter 14 **Dynamic Error Compensation functions** of "HIMC Series HMPL User Guide."

Important

The controller has only one table, so both axes will share the same table during compensation. When building compensation values, pay attention to the start position and number of points of each axis written to the table to avoid overwriting each other's values.

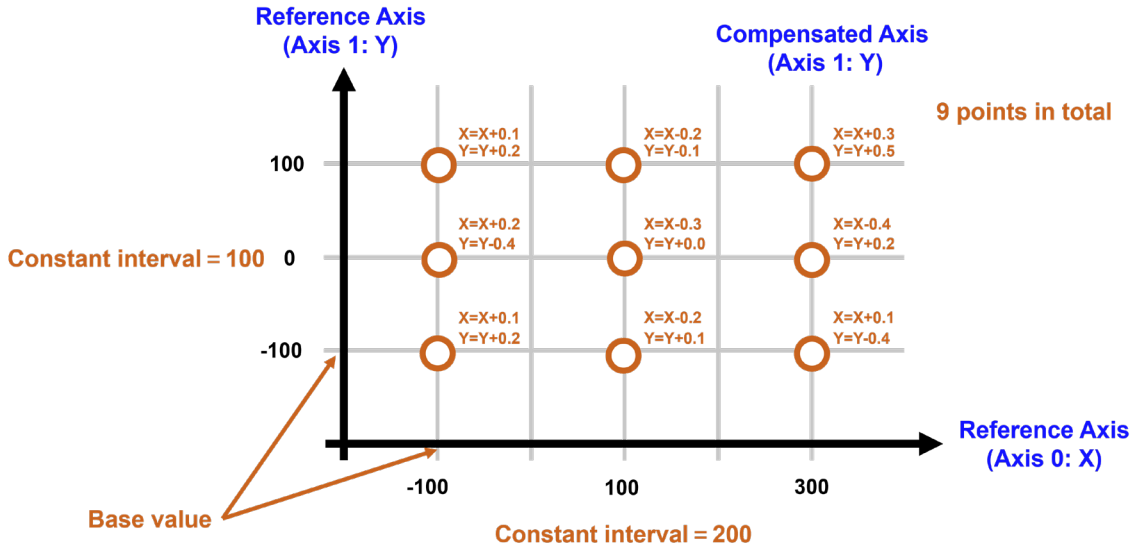


Figure 4.1.2.1 Position diagram of 2D error compensation

Set up and enable compensation via HMPL:

```
void main() {
    //  Set up error map
    double datay[9] = {0.2, 0.1, -0.4, -0.4, 0.0, 0.2, 0.2, -0.1, 0.5};  //  compensation
values (from left to right, bottom to top)
    double datax[9] = {0.1, -0.2, 0.1, 0.2, -0.3, -0.4, 0.1, -0.2, 0.3};  //  compensation
values (from left to right, bottom to top)
    SetUserTable(1, 9, datay);  //  Write data to user table
    SetUserTable(10, 9, datay);  //  Write data to user table

    double base[2] = {-100, -100};
    double interval[2] = {200, 100};
    int num_pt[2] = {3, 3};
    int ref_axis[2] = {0, 1};
    SetupComp2D(
      1,    //  axis to be compensated (axis 1: Y)
      1,    //  start index in user table
      base, //  base value
      interval,  //  interval
      num_pt,    //  number of points (base data included)
      ref_axis     //  reference axis (input)
    );

    SetupComp2D(
      0,    //  axis to be compensated (axis 0: X)
      10,    //  start index in user table
      base, //  base value
      interval,  //  interval
      num_pt,    //  number of points (base data included)
      ref_axis     //  reference axis (input)
    );

    EnableComp(0);  //  Enable compensation on (axis 0: X)
    Enable(0);  //  Enable (axis 0: X) to activate compensation
    EnableComp(1);  //  Enable compensation on (axis 1: Y)
    Enable(1);  //  Enable (axis 1: Y) to activate compensation
}
```

Disable compensation via HMPL:

```
void main() {
    Disable(1);  //  Disable axis 1
    Till(!IsEnabled(1));
    DisableComp(1);  //  Disable compensation on axis 1
    Disable(0);  //  Disable axis 0
    Till(!IsEnabled(0));
    DisableComp(0);  //  Disable compensation on axis 0
}
```

To check the compensation values, monitor the variables hcv.axis[0].pos_fb_comp (compensated position) and hcv.axis[0].pos_fb (original encoder position) in Scope Manager.

Important

(1) Compensated position = Original encoder position + Compensation value

(2) Modify the number within the brackets [ ] to monitor different axes.

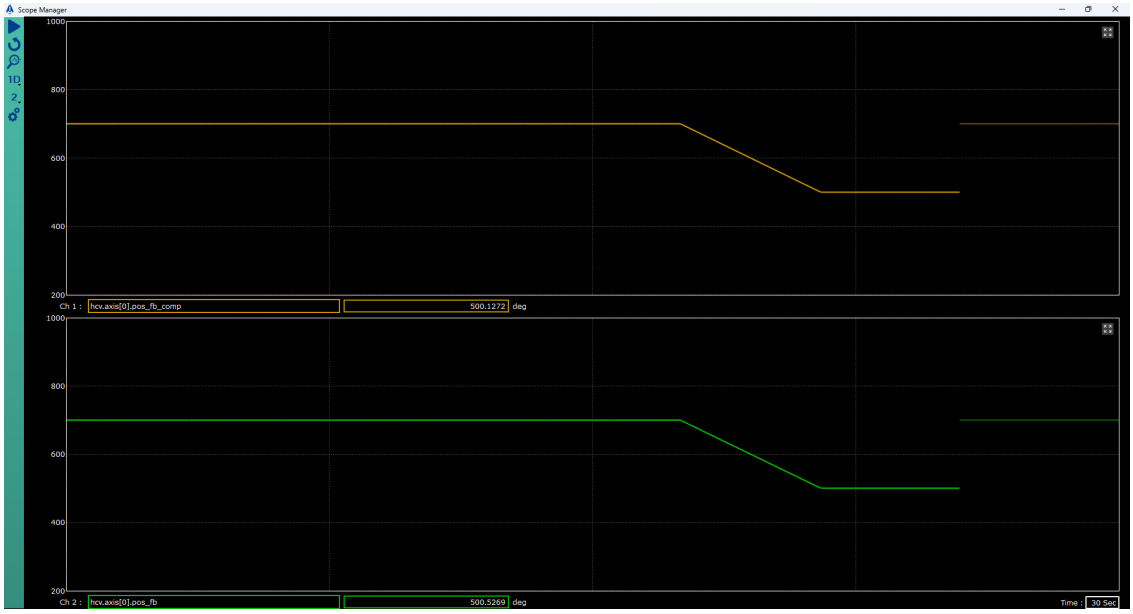


Figure 4.1.2.1 Encoder feedback and compensation values monitoring