

QEC Start Guide 8: 3 Axis Stepper Motor (CiA402 PP Mode)

In this guide, we will show you how to use the EtherCAT Master QEC-M-01P and the QEC-RXXMP3S Series (EtherCAT Slave, 3-axis Stepper Motor Controller). We will be operating CiA402 Profile Position (PP) mode.

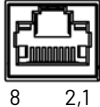
Notes QEC's PoE (Power over Ethernet)

In QEC product installations, users can easily distinguish between PoE and non-PoE: if the RJ45 house is red, it is PoE type, and if the RJ45 house is black, it is non-PoE type.



PoE (Power over Ethernet) is a function that delivers power over the network. QEC can be equipped with an optional PoE function to reduce cabling. In practice, PoE is selected based on system equipment, so please pay attention to the following points while evaluating and testing:

- The PoE function of QEC is different and incompatible with EtherCAT P, and the PoE function of QEC is based on PoE Type B, and the pin functions are as follows:

	Pin #	Signal Name	Pin #	Signal Name
	1	LAN1_TX+	2	LAN1_TX-
	3	LAN1_RX+	4	VS+
	5	VP+	6	LAN1_RX-
	7	VS- (GND)	8	VP- (GND)

* PoE LAN with the Red Housing; Regular LAN with Black Housing.

* L4, L5, L7, L8 pins are option, for RJ45 Power IN/OUT.

- When connecting PoE and non-PoE devices, make sure to disconnect Ethernet cables at pins 4, 5, 7, and 8 (e.g., when a PoE-supported QEC EtherCAT master connects with a third-party EtherCAT slave).

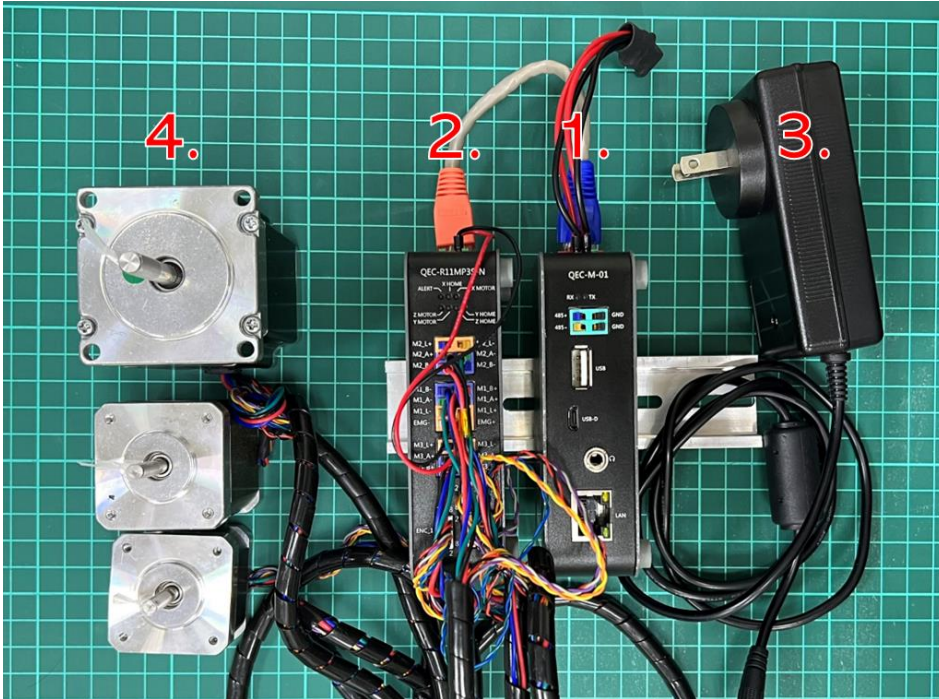


- QEC's PoE power supply is up to 24V/3A.

Connection and wiring hardware

The following devices are used here:

1. QEC-M-01P (EtherCAT Master/PoE)
2. QEC-R11MP3S (EtherCAT Slave, 3-axis Stepper Motor Controller) & LAN cable
3. 24V power supply & EU-type terminal cable
4. 4-wire two-phase bipolar 42 stepper motors with encoder interface * 3 (refer to [86STEP](#) | [86Duino](#))



QEC-M-01P

The QEC EtherCAT master with PoE functions.

1. Using the EtherCAT Out port (top side) connected to the EtherCAT In port of QEC-R11MP3S via RJ45 cable (powered by PoE).
2. Connect to Vs+/Vs- and Vp+/Vp- power supplies via EU terminals for 24V power. And provide power to QEC-R11MP3S-N via the PoE function.

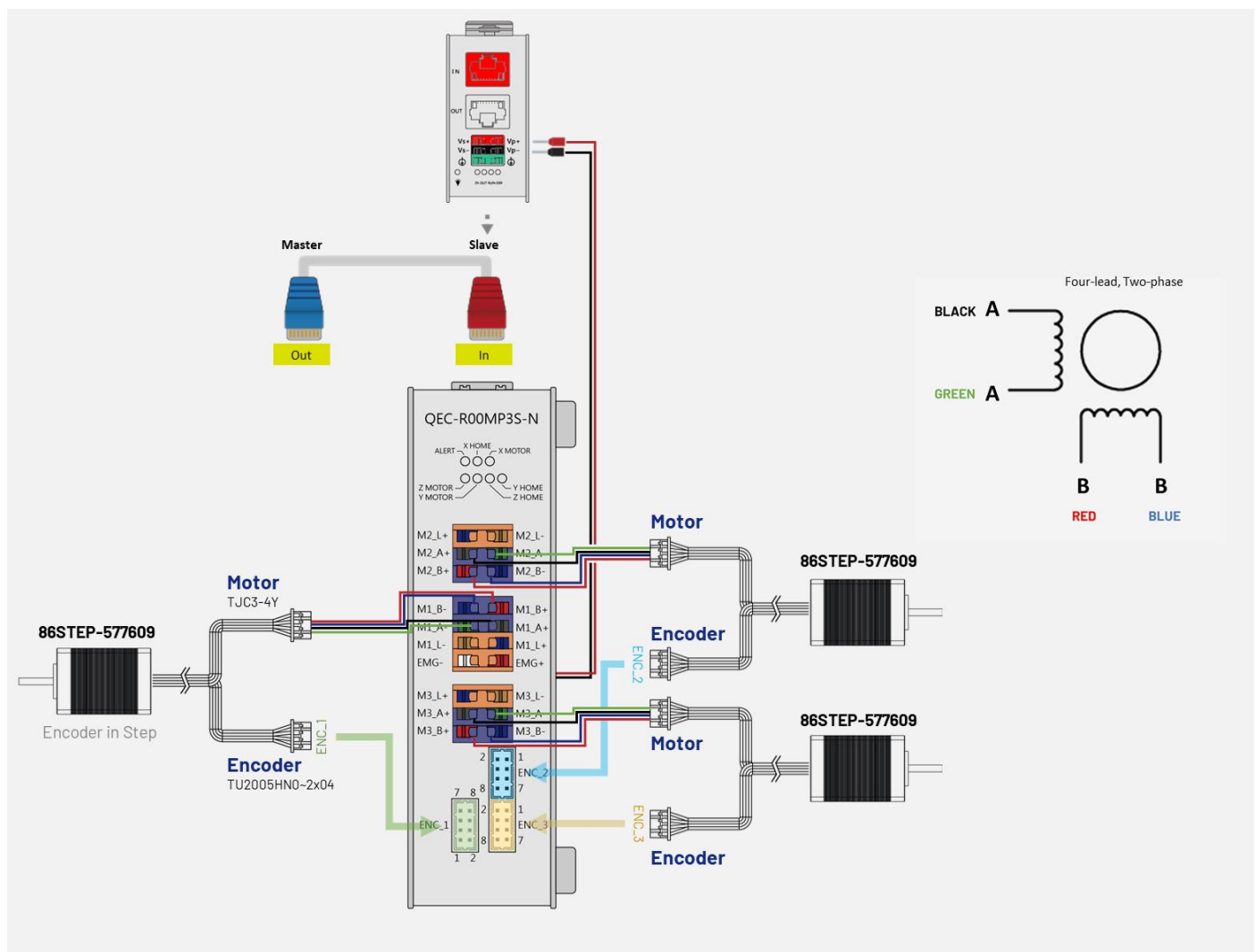


QEC-R11MP3S-N

QEC-R11MP3S-N with PoE functionality.

- Connect Vp+ to EMG+; Vp- to EMG-. (Power supply to the motor)
- Connect three 4-wire, bidirectional 42-step motors to the M1/M2/M3 axis of the QEC-R11MP3S-N, with wiring as follows:
 - A+ (black) to M_A+; A- (green) to M_A-.
 - B+ (red) to M_B+; B- (blue) to M_B-.
- Connect the encoders of the three 4-wire, bidirectional 42-step motors to ENC_1/ ENC_2/ ENC_3 of the QEC-R11MP3S-N respectively.

* Note: The wiring colors of the motor are the same as the terminal colors of the QEC-R11MP3S-N, users can follow the color for wiring reference.



Software/Development Environment: 86Duino IDE

Download 86duino IDE from <https://www.qec.tw/software/>.

QEC

Quicker, Easier Control



Download

The open-source 86Duino Software (IDE) makes it easy to write code and upload it to the QEC. Refer to the [Getting Started page](#) for Installation instructions.

86Duino_Coding_500_Beta_20230926_13

Download

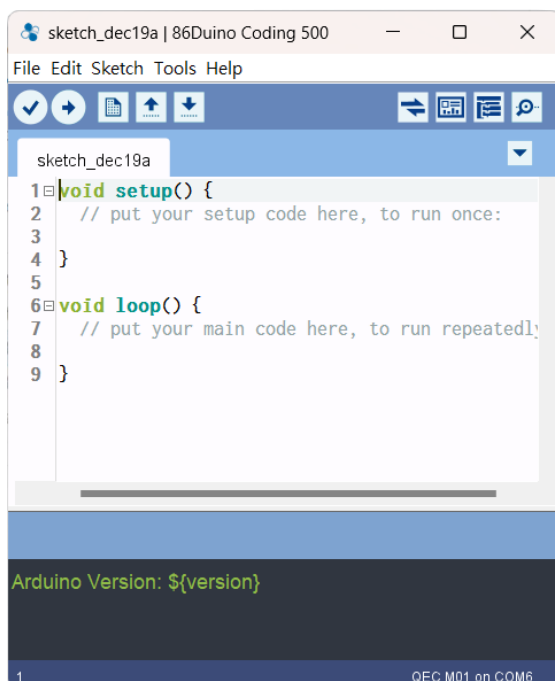
About how to update the QEC Master (QEC-M series products) with the latest version of the 86Duino IDE, please see [this page](#).

After downloading, please unzip the downloaded zip file, no additional software installation is required, just double-click 86duino.exe to start the IDE.



***Note:** If Windows displays a warning, click Details once and then click the Continue Run button once.

86Duino Coding IDE 500+ looks like below.

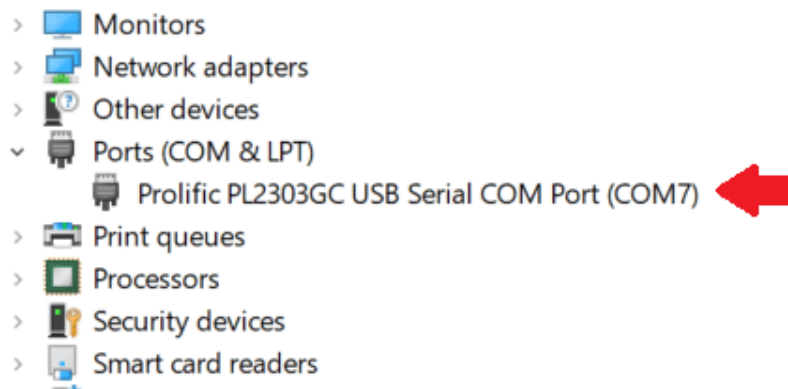
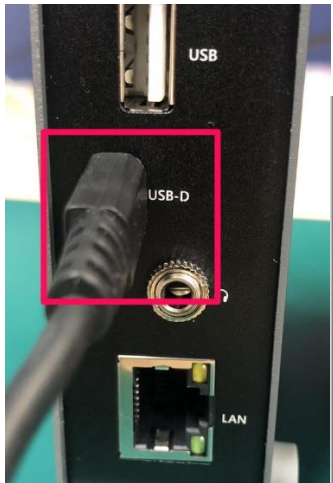


Connect to your PC and set up the environment

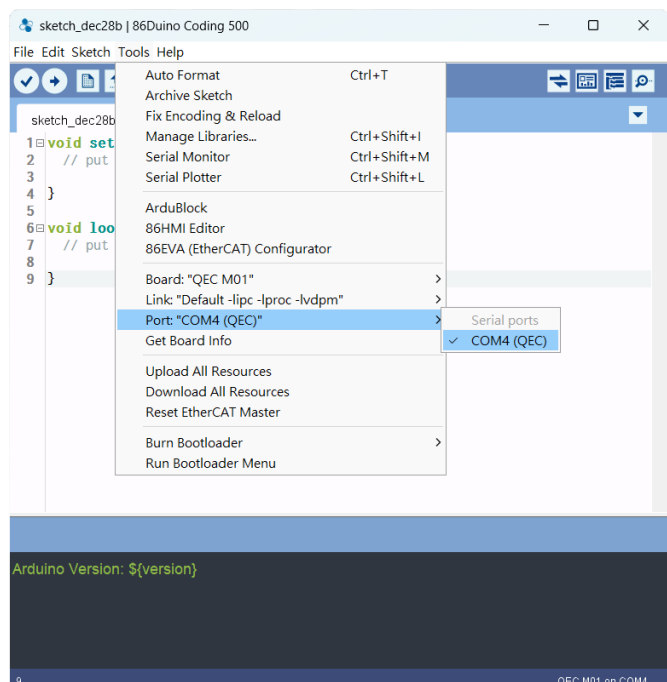
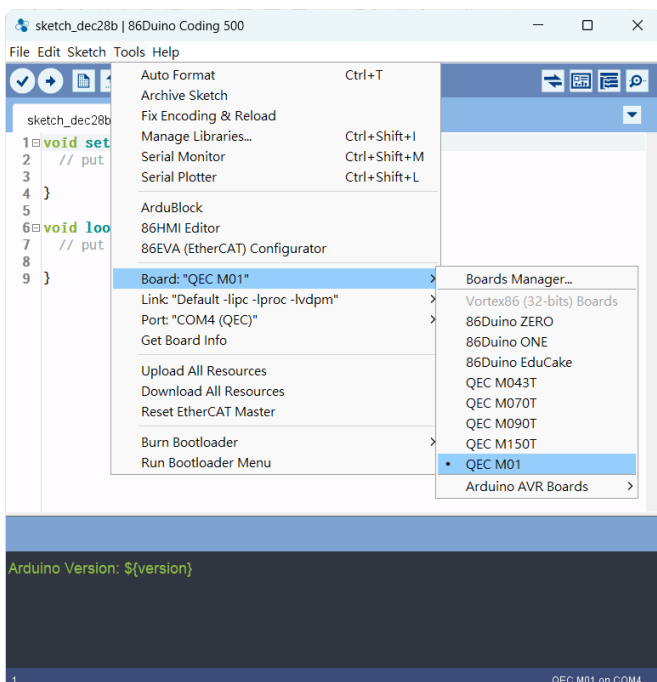
Follow the steps below to set up the environment:

1. Connect the QEC-M-01P to your PC via a Micro USB to USB cable (86Duino IDE installed).
2. Turn on the QEC power.
3. Open "Device Manager" (select in the menu after pressing Win+X -> "Ports (COM & LPT)" in your PC and expand the ports; you should see that the "Prolific PL2303GC USB Serial COM Port (COMx)" is detected; if not, you will need to install the required drivers.

(For Windows PL2303 driver, you can download [here](#))



4. Open the 86Duino IDE.
5. Select the correct board: In the IDE's menu, select Tools > Board > QEC-M-01 (or the QEC-M master model you use).
6. Select Port: In the IDE's menu, select Tools > Port and select the USB port to connect to the QEC-M master (in this case, COM4 (QEC)).



Development Method 1: Write code

The EtherCAT master (QEC-M-01P) and the HID slave (QEC-R11MP3S-N) can be configured and programmed via the EtherCAT library in the 86Duino IDE. The Arduino development environment has two main parts: `setup()` and `loop()`; `setup()` corresponds to initialization which do once, and `loop()` corresponds to main programs which do repeat. Before operating the EtherCAT network, you must configure it once. The process should be from Pre-OP to OP mode in EtherCAT devices.

In this example, we achieve synchronous motion control through the EtherCAT communication protocol combined with DC mode. We use the QEC-R11MP3S controller to manage three-axis stepper motors. The motion control adopts the Profile Position (PP) mode of the CiA402 standard, which is a position control mode based on predefined paths. By calling the `profilePositionBegin()` function, we set the motor to move to a specific position, 81920, at a speed and acceleration of 10,000. After sending the command, we wait for the driver's Control word to confirm receipt of the command and verify that the motor has reached the predetermined position. Once completed, we reset the target position to 0 and move at the same speed and acceleration, allowing the motor to move back and forth between these two points.

For third-party slave support of CiA402 objects:

When operating EtherCAT CiA402 objects, you can use the dedicated EtherCAT CiA402 library.

```
#include "Ethercat.h" // Include the EtherCAT Library

#define MOTORS      (3) // Define the number of motors

EthercatMaster master; // Create an EtherCAT Master Object
EthercatDevice_CiA402 motor[MOTORS]; // Create the EtherCAT Slave Objects for CiA402 motor

int pp_state[MOTORS]; // Array to hold the profile position state of each motor
int pp_done; // Variable to track the completion of profile positions

// Callback function for cyclic updates
void myCallback() {
    for (int i = 0; i < MOTORS; i++) {
        motor[i].update(); Slave1.update(); // Manually call update();
    }
}

void setup(void) {
    // Initialize the EtherCAT Master. If successful, all slaves enter PRE OPERATIONAL state
    master.begin(ECAT_ETH_1);

    for (int i = 0; i < MOTORS; i++) {
        // Attach CiA402 motor to the EtherCAT Master
        // First Parameter: EtherCAT node number, Second Parameter: Stepper Motor number
```



```

motor[i].attach(0, i, master);
motor[i].setDc(1000000); // Set Distributed Clocks Synchronization Cycle time
motor[i].driveSetMode(CIA402_PP_MODE); // Set motor drive operation mode to Profile Position Mode
}

master.attachCyclicCallback(myCallback); // Attach a cyclic callback for EtherCAT Master
// Start the EtherCAT Master. If successful, all slaves enter OPERATIONAL state
master.start(1000000); // Set the cycle time to 1 second (1000000 microseconds)

for (int i = 0; i < MOTORS; i++) {
    motor[i].driveEnable(); // Enable motor (CiA402 state set to OPERATION_ENABLED)
}
}

void loop() {
    for (int i = 0; i < MOTORS; i++) {
        // State machine for each motor's profile position control
        switch (pp_state[i]) {
            case 0:
                // Begin moving the motor to the target position with specified velocity and acceleration
                if (motor[i].profilePositionBegin(81920, 10000, 10000) == 0)
                    pp_state[i]++; // Advance state if command successful
                break;
            case 1:
                // Check if the motor's control word is ready for next command
                if ((motor[i].driveGetControlword() & (1 << 4)) == 0)
                    pp_state[i]++; // Advance state if control word is ready
                break;
            case 2:
                // Check if the motor has reached its target position
                if (motor[i].driveIsTargetReached())
                    pp_state[i]++; // Advance state if target reached
                break;
            case 3:
                // Command motor to return to original position
                if (motor[i].profilePositionBegin(0, 10000, 10000) == 0)
                    pp_state[i]++; // Advance state if command successful
                break;
            case 4:
                // Check again if control word is ready for next command
                if ((motor[i].driveGetControlword() & (1 << 4)) == 0)
                    pp_state[i]++; // Advance state if control word is ready



```

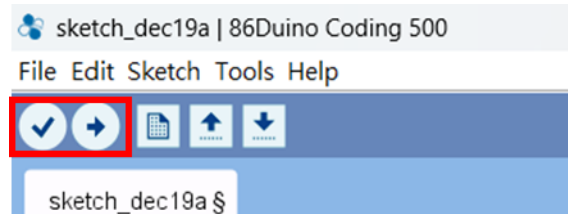
```

    break;
case 5:
    // Final state to check if motor has returned to the initial position
    if (motor[i].driveIsTargetReached()) {
        pp_state[i] = 0; // Reset state machine for this motor
        pp_done++; // Increment count of motors that have completed the profile position sequence
    }
    break;
}
}

// Reset for all motors if all have completed their moves
if (pp_done == MOTORS) {
    pp_done = 0; // Reset the done counter
    for (int i = 0; i < MOTORS; i++)
        pp_state[i] = 0; // Reset all states to 0
}
}

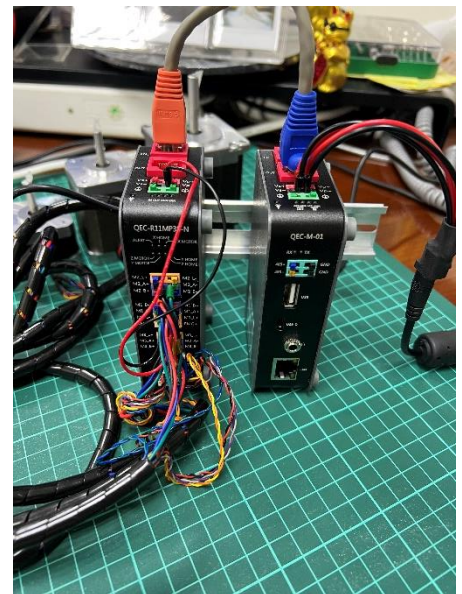
```

Note: Once the code is written, click on the toolbar to  compile, and to confirm that the compilation is complete and error-free, you can click  to upload. The program will run when the upload is complete.



After the upload is complete, you can see all three stepper motors move to the specified position and then return to the initial position.

This process will repeat continuously.



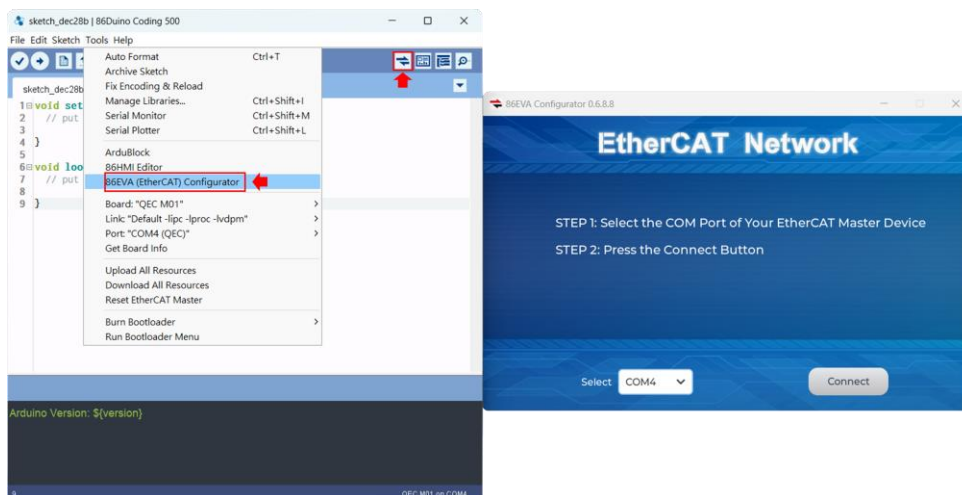
Development Method 2: Use 86EVA with code

86EVA is a graphical EtherCAT configurator based on the EtherCAT Library in the 86Duino IDE and is one of the development kits for 86Duino. The user can use it to configure the EtherCAT network quickly and start programming.

In this example, we achieve synchronous motion control through the EtherCAT communication protocol combined with DC mode. We use the QEC-R11MP3S controller to manage three-axis stepper motors. The motion control adopts the Profile Position (PP) mode of the CiA402 standard, which is a position control mode based on predefined paths. By calling the profilePositionBegin() function, we set the motor to move to a specific position, 81920, at a speed and acceleration of 10,000. After sending the command, we wait for the driver's Control word to confirm receipt of the command and verify that the motor has reached the predetermined position. Once completed, we reset the target position to 0 and move at the same speed and acceleration, allowing the motor to move back and forth between these two points.

Step 1: Turn on 86EVA and scan

The 86EVA tool can be opened via the following buttons.

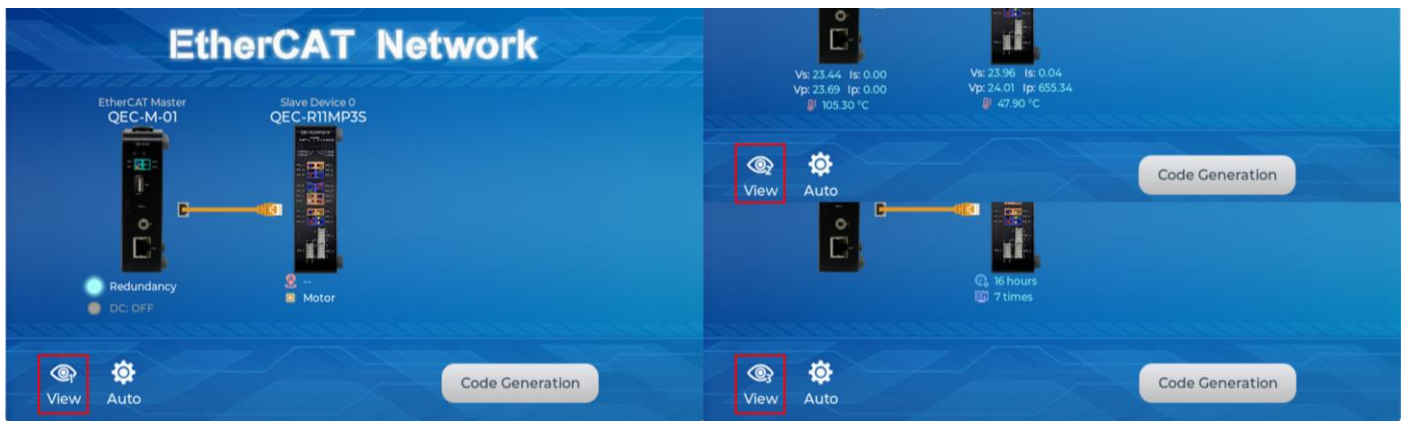


Once you have confirmed that the correct COM port has been selected of QEC-M-01P, press the Connect button to start scanning the EtherCAT network.



The connected devices will be displayed after the EtherCAT network has been scanned.

Press the "View" button in the lower left corner to check the device's status (Voltage, Current, and Temperature; View2) and operating time (Hours; View3).



Step 2: Set the parameters

Press twice on the scanned device image to enter the corresponding parameter setting screen.



QEC-M-01

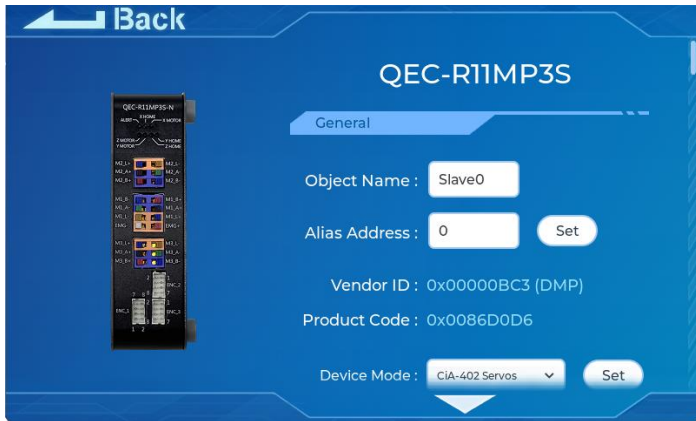
Press twice on the image of the QEC-M-01 to see the parameter settings.

This example will use the default settings and not change any settings; please click "Back" in the upper left corner to return.



QEC-R11MP3S-N

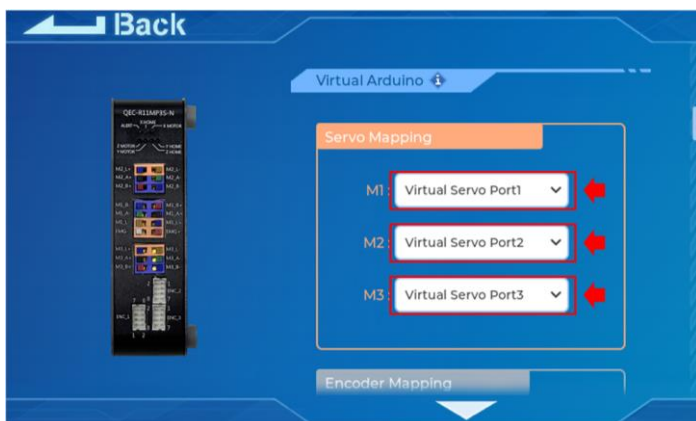
Press twice on the image of the QEC-R11MP3S to see the parameter settings.



Please note the "Device Mode" field in "General," which has two options: CiA-402 Servos and G-code Machine. This example uses the CiA-402 Servos mode. (Note, if the default is not set to CiA-402 Servos mode, please refer to [Troubleshooting: QEC-RXXMP3S cannot control the motor.](#))



Continue to the "Servo Mapping" section. Among them, we select "Virtual Servo Port1" from the dropdown menu for M1, "Virtual Servo Port2" for M2, and "Virtual Servo Port3" for M3.



After finishing, click "Back" in the upper left corner to return.



This action configures the M1 (first axis motor), M2 (second axis motor), and M3 (third axis motor) of the QEC-R11MP3S as the virtual Servo Port1, virtual Servo Port2, and virtual Servo Port3 of the EVA, respectively.

Step 3: Generate the code

Once you've set your device's parameters, go back to the home screen and press the "Code Generation" button in the bottom right corner.

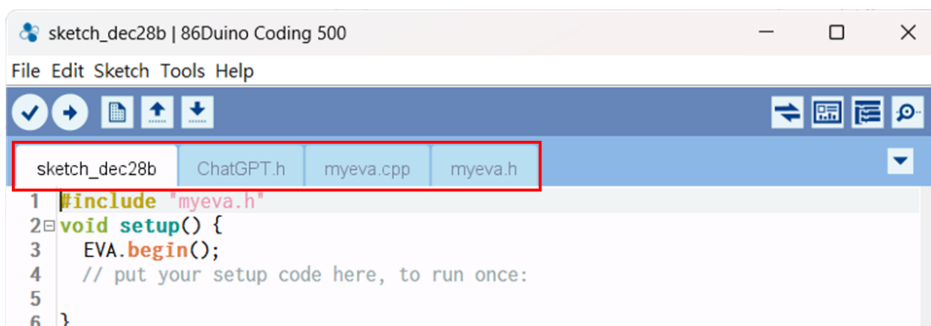


When you're done, double-click the OK button to turn off 86EVA, or it will close in 10 seconds.



The generated code and files are as follows:

- sketch_dec28b: Main Project (.ino, depending on your project name)
- ChatGPT.h: Parameters to provide to ChatGPT referred
- myeva.cpp: C++ program code of 86EVA
- myeva.h: Header file of 86EVA



Additional note: After 86EVA generates code, the following code will be automatically generated in the main program (.ino), and any of them missing will cause 86EVA not to work.

1. #include "myeva.h" : Include EVA Header file
2. EVA.begin() in setup() ; : Initialize the EVA function

Step 4: Write the code

In this example, we achieve synchronous motion control through the EtherCAT communication protocol combined with DC mode. We use the QEC-R11MP3S controller to manage three-axis stepper motors. The motion control adopts the Profile Position (PP) mode of the CiA402 standard, which is a position control mode based on predefined paths. By calling the `profilePositionBegin()` function, we set the motor to move to a specific position, 81920, at a speed and acceleration of 10,000. After sending the command, we wait for the driver's Control word to confirm receipt of the command and verify that the motor has reached the predetermined position. Once completed, we reset the target position to 0 and move at the same speed and acceleration, allowing the motor to move back and forth between these two points.

```
#include "myeva.h"

#define MOTORS      3 // Define the number of motors
EthercatDevice_CiA402* motor[MOTORS]; // Array to hold pointers to EtherCAT Slave Objects for CiA402
motors

int pp_state[MOTORS]; // Array to hold the profile position state of each motor
int pp_done = 0; // Variable to track the completion of profile positions, initialized to 0

void setup() {
    EVA.begin();
    // Assign pointers returned by cia402GetServo() directly to the motor array elements
    motor[0] = VirtualServo1.cia402GetServo();
    motor[1] = VirtualServo2.cia402GetServo();
    motor[2] = VirtualServo3.cia402GetServo();
    for (int i = 0; i < MOTORS; i++) {
        motor[i]->setDc(1000000); // Use arrow operator for pointer access
        motor[i]->driveEnable(); // Use arrow operator for pointer access
    }
}



void loop() {
    for (int i = 0; i < MOTORS; i++) {
        // State machine for each motor's profile position control
        switch (pp_state[i]) {
            case 0:
                if (motor[i]->profilePositionBegin(81920, 10000, 10000) == 0)
                    pp_state[i]++; // Advance state if command successful
                break;
            case 1:
                if ((motor[i]->driveGetControlword() & (1 << 4)) == 0)
```

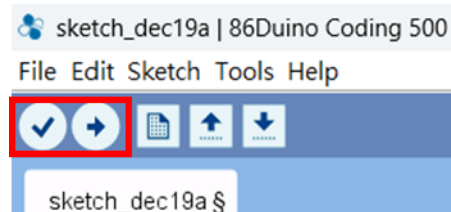
```

    pp_state[i]++; // Advance state if control word is ready
    break;
case 2:
    if (motor[i]->driveIsTargetReached())
        pp_state[i]++; // Advance state if target reached
    break;
case 3:
    if (motor[i]->profilePositionBegin(0, 10000, 10000) == 0)
        pp_state[i]++; // Advance state if command successful
    break;
case 4:
    if ((motor[i]->driveGetControlword() & (1 << 4)) == 0)
        pp_state[i]++; // Advance state if control word is ready
    break;
case 5:
    if (motor[i]->driveIsTargetReached()) {
        pp_state[i] = 0; // Reset state machine for this motor
        pp_done++; // Increment count of motors that have completed the profile position sequence
    }
    break;
}
}

if (pp_done == MOTORS) {
    pp_done = 0; // Reset the done counter
    for (int i = 0; i < MOTORS; i++)
        pp_state[i] = 0; // Reset all states to 0
}
}

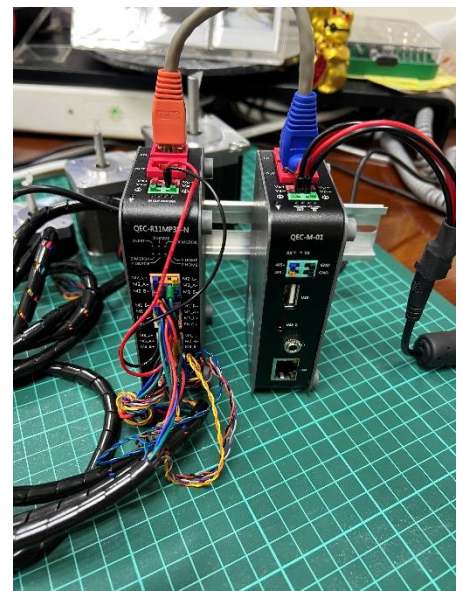
```

Note: Once the code is written, click on the toolbar to  compile, and to confirm that the compilation is complete and error-free, you can click  to upload. The program will run when the upload is complete.



After the upload is complete, you can see all three stepper motors move to the specified position and then return to the initial position.

This process will repeat continuously.



Troubleshooting

Old environment of your QEC-M-01:

If you cannot upload your sketch successfully, please try to update your QEC EtherCAT Master. This update covers the following three updates: Bootloader, EtherCAT Firmware, and EtherCAT Tools.

Now, we will further explain how to proceed with the update:

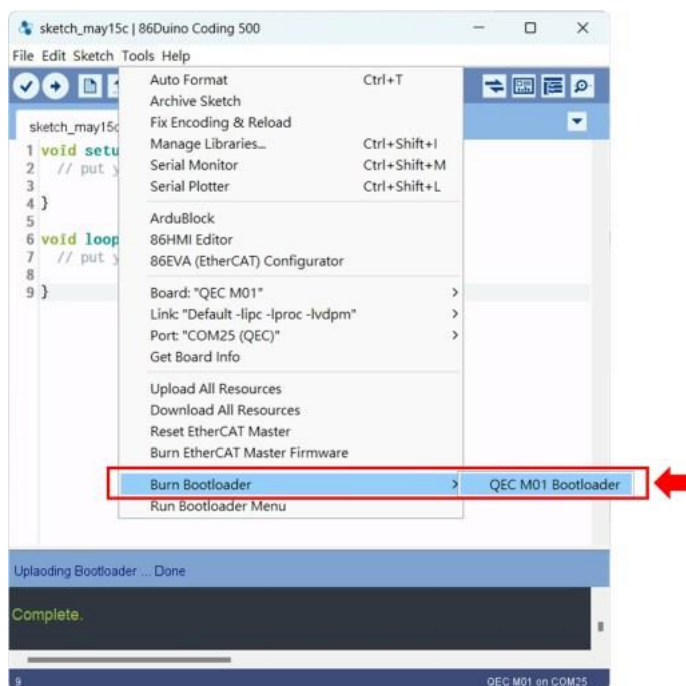
Step 1: Setting up QEC-M

1. Download and install 86Duino IDE 500 (or a newer version): You can download it from [Software](#).
2. Connect the QEC-M: Use a USB cable to connect the QEC-M to your computer.
3. Open 86Duino IDE: After the installation is complete, open the 86Duino IDE software.
4. Select Board: From the IDE menu, choose "Tools" > "Board" > "QEC-M-01" (or the specific model of QEC-M you are using).
5. Select Port: From the IDE menu, choose "Tools" > "Port" and select the USB port to which the QEC-M is connected.

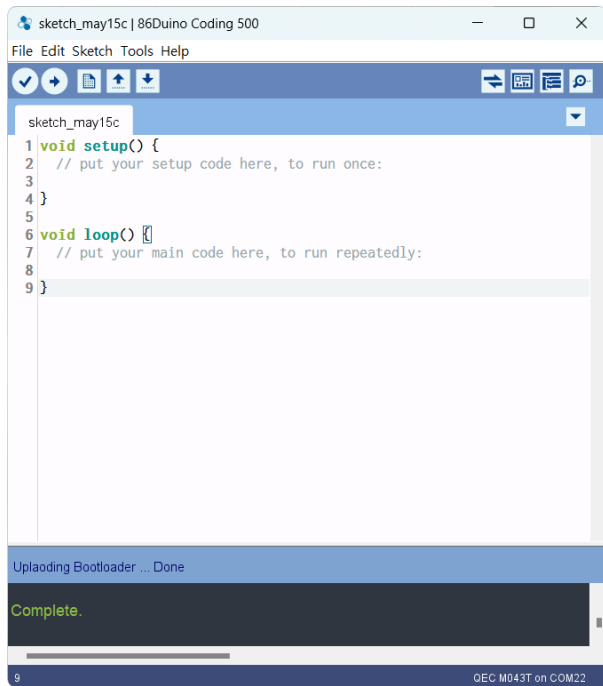
Step 2: Click "Burn Bootloader" button

After connecting to your QEC-M product, go to "Tools"> "Burn Bootloader". The currently selected QEC-M name will appear. Clicking on it will start the update process, which will take approximately 5-20 minutes.

QEC-M-01:



Step 3: Complete the Update



After completing the above steps, your QEC-M has been successfully updated to the latest version of the development environment.

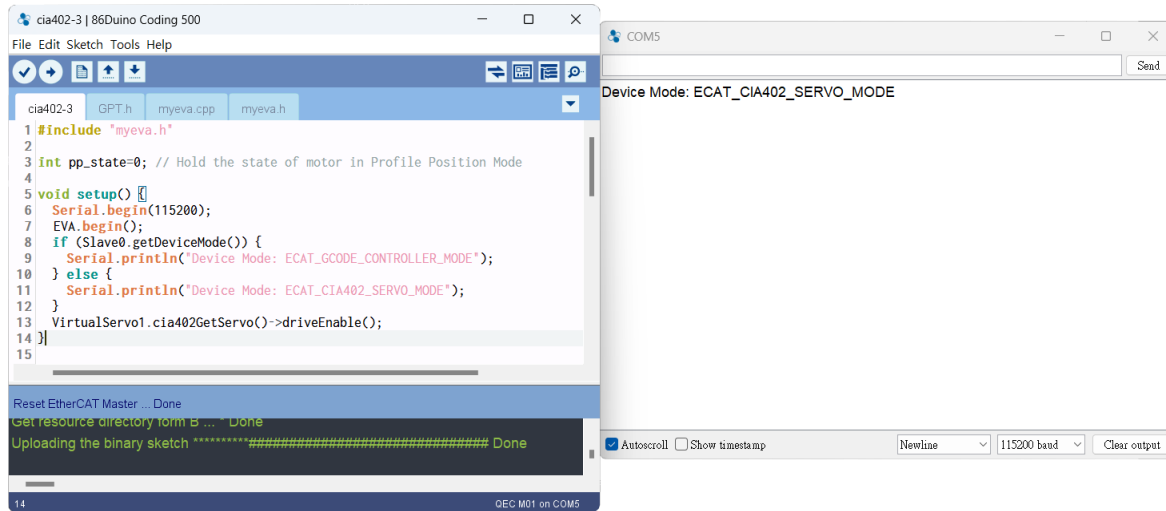
QEC-RXXMP3S cannot control the motor:

Mode setting error:

On the QEC-RXXMP3S, users can operate two types of motion control methods: CiA-402 and G-code. Users must use the corresponding command set and control strategy because CiA-402 and G-code modes are not interchangeable.

Users can check the current mode of the QEC-RXXMP3S through the code `getDeviceMode()`; or by viewing it on the 86EVA.

To check using the 86Duino IDE:



To view using 86EVA:



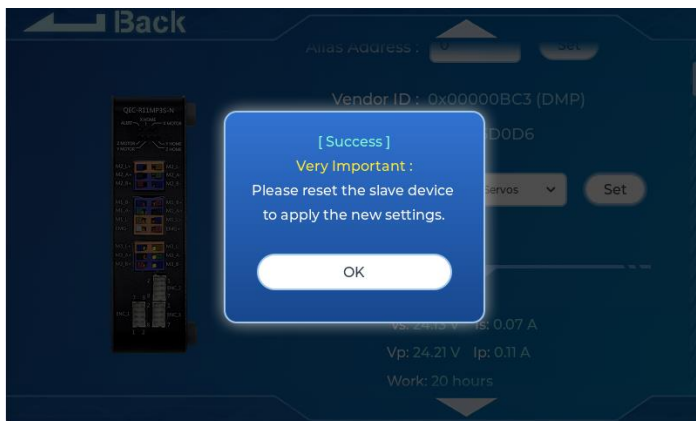
Changing the Mode of QEC-RXXMP3S

Below is the method to change the mode of QEC-RXXMP3S.

1. You can change the current control mode of QEC-RXXMP3S through the "Device Mode" dropdown menu in 86EVA.



2. After selecting, click the "Set" button on the right. A success notification window will appear, reminding the user that it is **necessary to power off and restart the slave device**.



3. After powering off and restarting the QEC-RXXMP3S, reopen the 86EVA tool through the QEC master station and enter the settings page of QEC-RXXMP3S, you will be able to see the motion control mode you have set (CiA-402 Servos or G-code Machine).



After completing the steps above, your QEC-RXXMP3S slave has been successfully updated to your desired motion control mode; please ensure that restarting the slave device after switching modes is very important, as this is a necessary condition for the changes to take effect.

For more information and sample requests, please write to info@icop.com.tw, call your nearest [ICOP branch](#), or contact our [official global distributor](#).