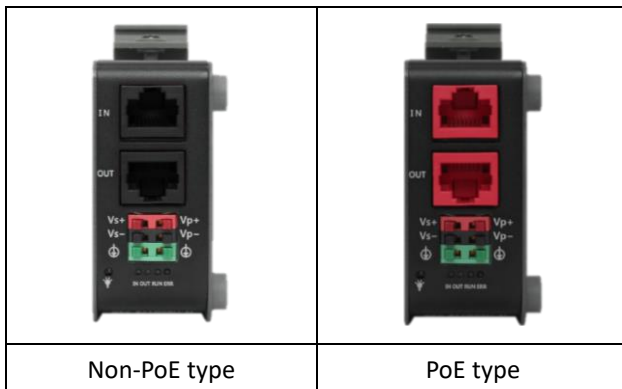


# QEC Start Guide 10: RS-485 Basic Transmission

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 G-code mode.

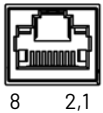
## Notes QEC's PoE (Power over Ethernet)

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-R11HU9S-N (EtherCAT HID Slave, supports 2 UART, 1 MPG, 1 Keypad, 1 LCM)
3. 24V power supply & EU-type terminal cable
4. RS-485 cable & LAN cable



## QEC-M-01P

QEC EtherCAT master with PoE function.

1. Using the EtherCAT Out port (top side) connected to the EtherCAT In port of QEC-R11HU9S via RJ45 cable (powered by PoE).
2. Connect to Vs+/Vs- and Vp+/Vp- power supplies via EU terminals for 24V power.



## QEC-R11HU9S-N

- On QEC-R11HU9S-N, COM1 and COM2 signals RS-485- are on pin 1, RS-485+ are on pin 2.

No.	Pin Assignment	No.	Pin Assignment
1	RS485-	6	DSR
2	RS485+/RXD	7	RTS
3	TXD	8	CTS
4	DTR	9	VCC
5	GND	-	-

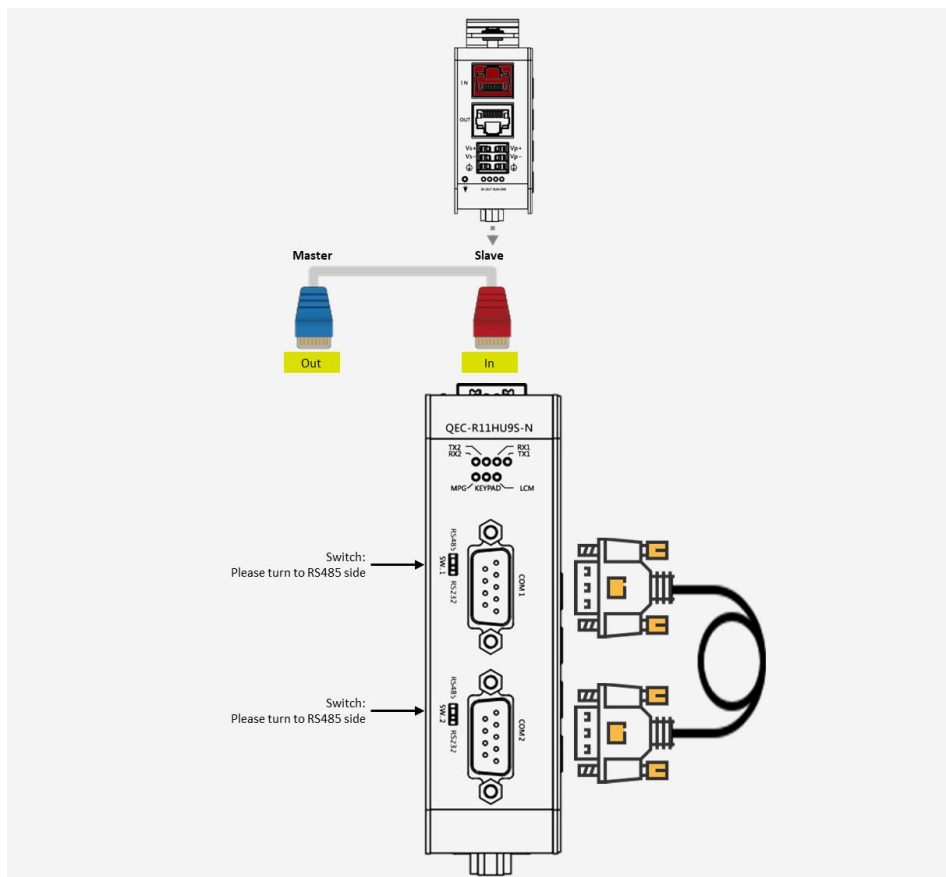
\* Note: RS232 and RS485 cannot be used simultaneously.

- This example is for using RS-485.

Note: Next to the two UART Ports, there is a switch each that can be adjusted through the direction of this switch to switch between RS-232 function or RS-485 function. After adjusting, please restart the power of the device.



- Connect COM1 and COM2 on the QEC-R11HU9S-N through the RS-485 transmission line.



**Note:** The RS-485 supported by the QEC-RXXHU series requires users to add a 120-ohm termination resistor according to the specific scenario to optimize signal quality. Termination resistors can match the impedance of RS-485, reducing signal reflection, which is especially important in long-distance or high-interference environments. When designing an RS-485 network, considering the network length and the number of devices to determine the need for termination resistors is key to ensuring communication reliability and efficiency.

# 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**

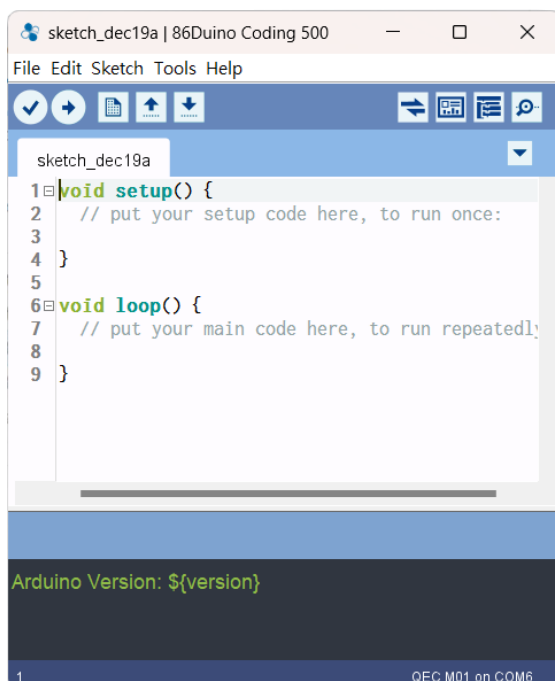
About how to update the QEC Master (QEC-M series products) with the latest version of the 86Duino IDE, please see [this page](#).

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.

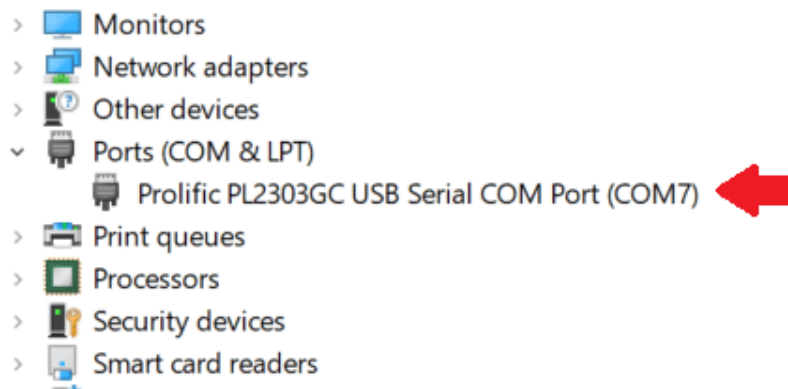
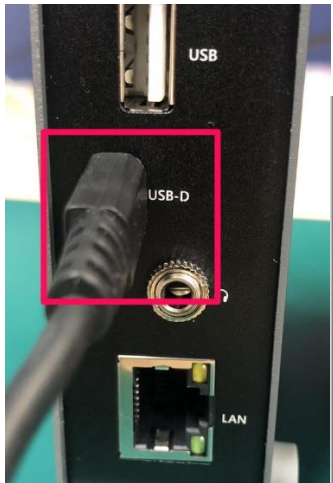


# Connect to your PC and set up the environment

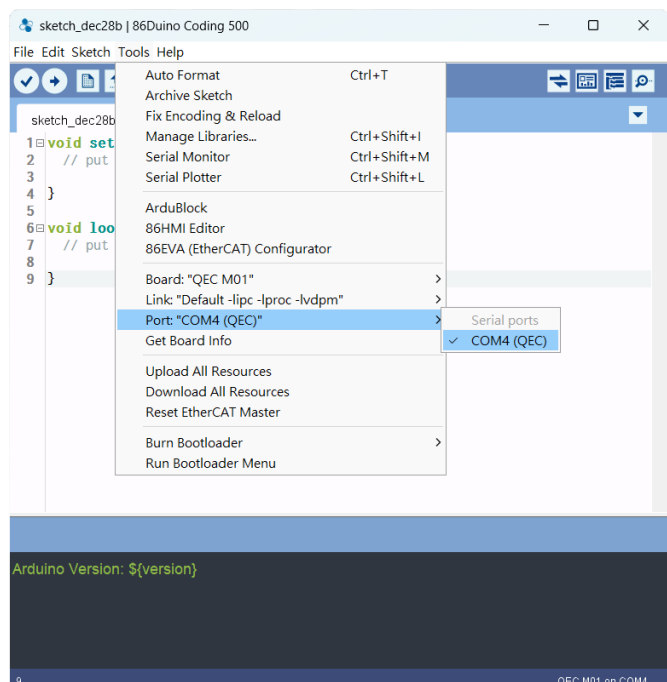
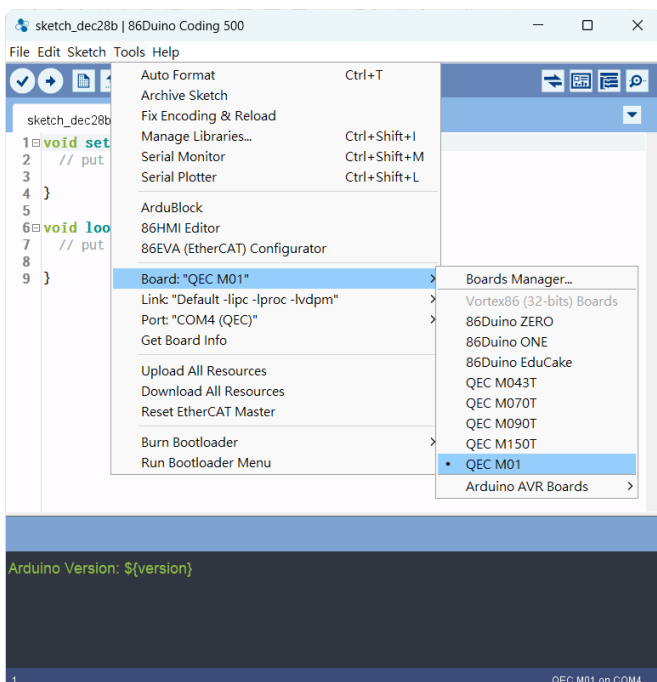
Follow the steps below to set up the environment:

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-R11HU9S-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()`, which correspond to initialization and main programs. Before operating the EtherCAT network, you must configure it once. The process should be from Pre-OP to OP mode in EtherCAT devices.

**In the following example, we utilized the Ethercat library and the RTCZero library to integrate EtherCAT communication and real-time clock functionality. Initially, we configured two RS485 ports on the QEC-R11HU9S with specific baud rates and data formats. Then, in the main `loop()` function, we continuously send the current date and time from one port (COM1) every second and read it back from the other port (COM2). This setup can be used for applications requiring synchronized time stamps or logs that need precise time data, conducted through RS485 communication.**

When using QEC Slave, you can use the dedicated QEC Ethercat Slave Library. For example, QEC-R11HU9S can be used [EthercatDevice\\_QECRXXHU Class](#). The example code is as follows:

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

EthercatMaster EcatMaster; // Create an EtherCAT Master Object
EthercatDevice_QECR11HU9S Slave1; // Create an EtherCAT Slave Objects for QEC-R11HU9S
RTCZero rtc; // Create an RTC instance

int incomingByte = 0; // Variable for storing incoming serial data
char dateString[100]; // Array to hold the formatted date string

// Callback function for cyclic updates
void myCallback() {
    Slave1.update(); // Since it is non-blocking, manually call update();
}

void setup() {
    Serial.begin(115200); // Initialize serial communication at 115200 baud rate
    EcatMaster.begin(); // Initialize the EtherCAT Master. If successful, all slaves enter PRE OPERATIONAL
state
    Slave1.attach(0, EcatMaster); // Attach QECR11HU9S to the EtherCAT Master at position 0
    EcatMaster.attachCyclicCallback(myCallback); // Attach a cyclic callback for EtherCAT Master
    EcatMaster.start(1000000, ECAT_FREERUN_AUTO); // Start the EtherCAT Master. If successful, all slaves
enter OPERATIONAL state; Free-run Mode, and the parameter 1000000 sets the cycle time in nanoseconds

    // Set up RS485 ports on the slave device
    Slave1.uartSetBaud(COM1, 115200); // Set baud rate for COM1
    Slave1.uartSetFormat(COM1, SERIAL_8N1); // Set data format for COM1
```

```



Slave1.uartSetBaud(COM2, 115200); // Set baud rate for COM2
Slave1.uartSetFormat(COM2, SERIAL_8N1); // Set data format for COM2
rtc.begin(); // Initialize the real-time clock
}
int slen; // Variable to store the length of the date string
void loop() {
    // Format the current date and time into a string
    sprintf(dateString, "%d/%02d/%02d - %02d:%02d:%02d", rtc.getYear() + 2000, rtc.getMonth(), rtc.getDay(),
rtc.getHours(), rtc.getMinutes(), rtc.getSeconds()); // Assemble a string of year, month, day, hour,
minute, second
    slen = strlen(dateString); // Calculate the length of the date string

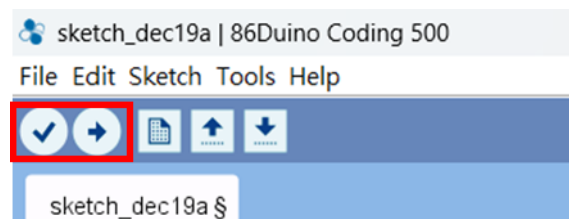
    Slave1.uartSend(COM1, (void*)dateString, slen); // COM Port Test: Send from EtherCAT HID COM1 to HID COM2

    // Display received characters
    for (int i = 0; i < slen + 1; i++)
        Serial.print((char)Slave1.uartRead(COM2)); // Read and print the received data from COM2
    Serial.println(); // Print a newline for readability

    delay(1000); // Wait for a second before the next loop iteration
}

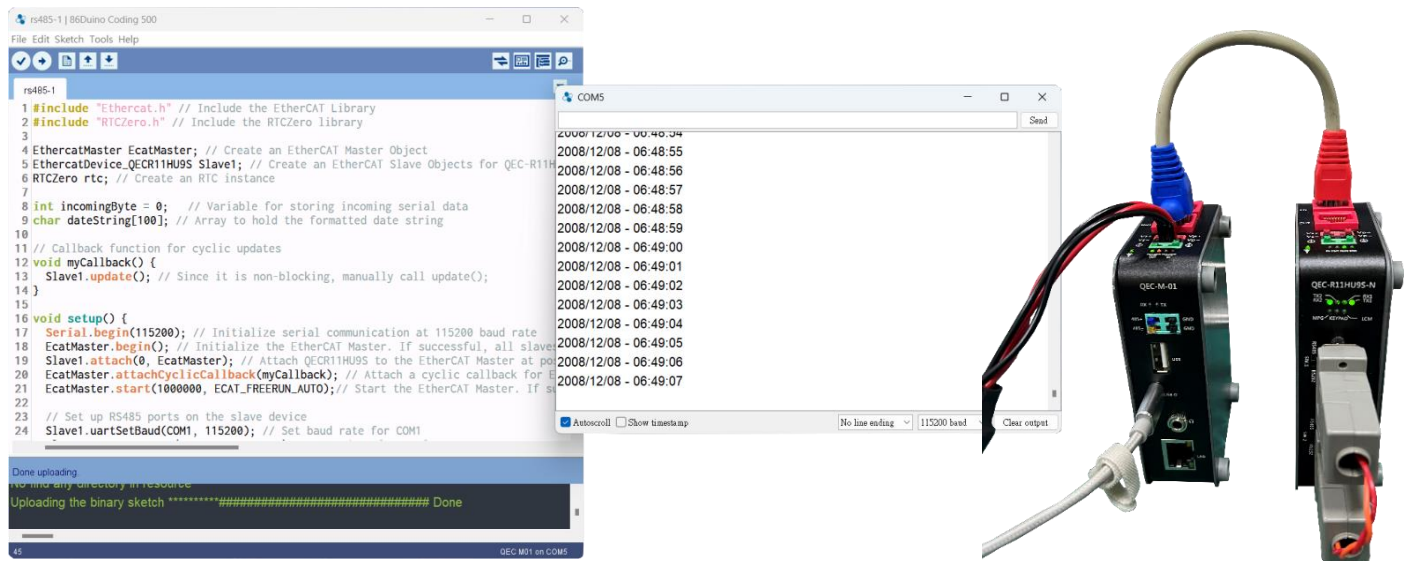
```

**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 uploading, you can view the current date and time received from RS485 COM2 through the Serial Monitor of the 86Duino IDE. This time is sent from RS485 COM1 and is updated every second. This setup can be used for applications requiring synchronized time stamps or logs that need precise time data, conducted through RS485 communication.





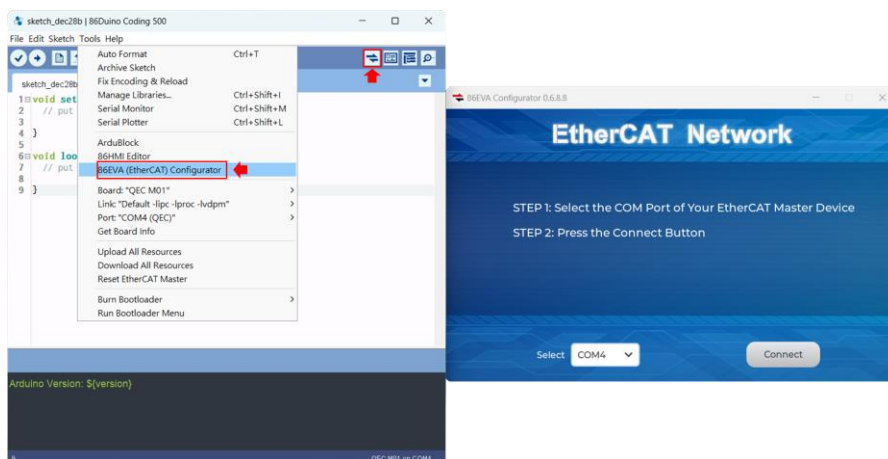
# Development Method 2: Use 86EVA with code

86EVA is a graphical EtherCAT configuration tool based on the EtherCAT Library in the 86Duino IDE, part of the development kit of the 86Duino IDE. Users can configure the EtherCAT master (QEC-M-01P) and HID slave (QEC-R11HU9S-N) through 86EVA, and then start programming.

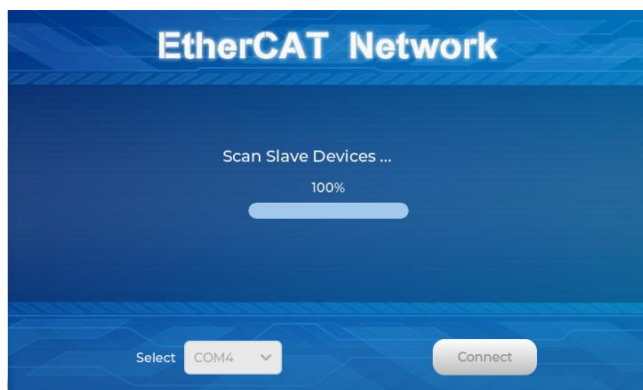
In the following example, we utilized the Ethercat library and the RTCZero library to integrate EtherCAT communication and real-time clock functionality. Initially, we configured two RS485 ports on the QEC-R11HU9S with specific baud rates and data formats. Then, in the main `loop()` function, we continuously send the current date and time from one port (COM1) every second and read it back from the other port (COM2). This setup can be used for applications requiring synchronized time stamps or logs that need precise time data, conducted through RS485 communication.

## Step 1: Turn on 86EVA and scan

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

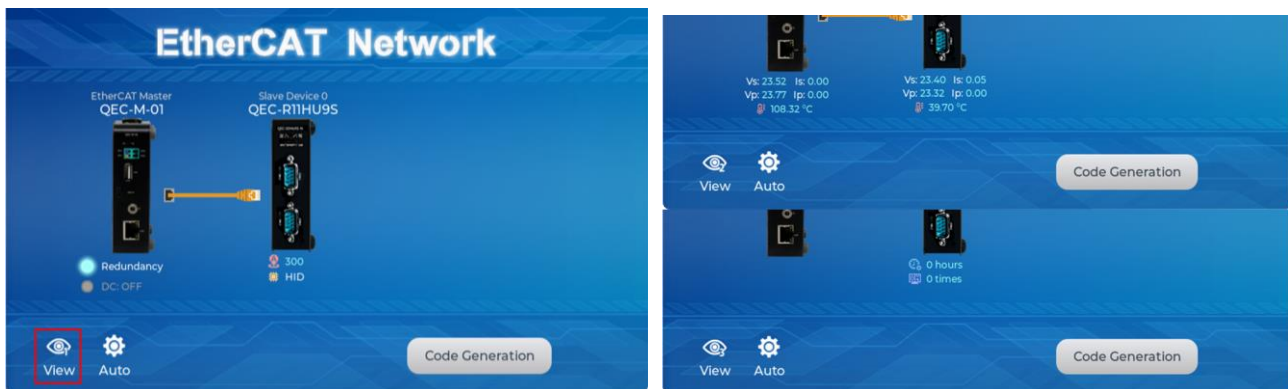


Once you have confirmed that the correct COM port has been selected of QEC-M-01P, press the Connect button to start scanning the EtherCAT network.



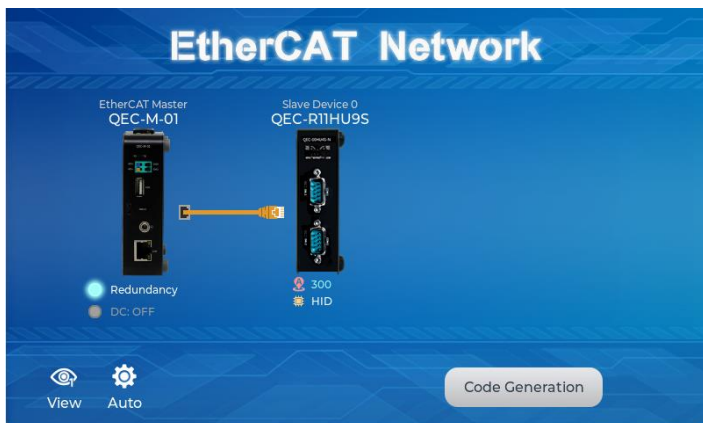
The connected devices will be displayed after the EtherCAT network has been scanned.

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.

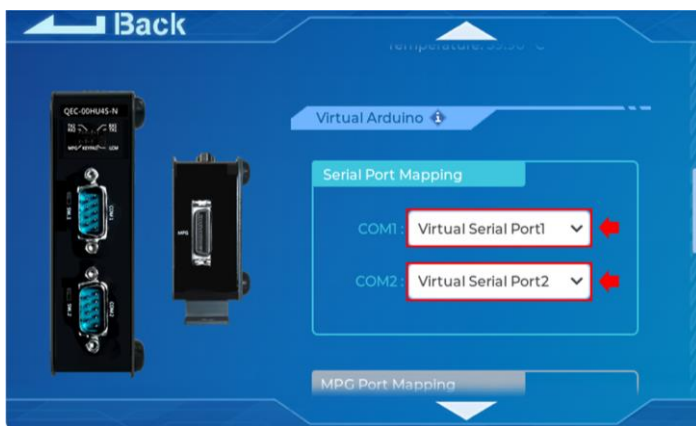


## QEC-R11HU9S-N

Press twice on the image of the QEC-R11HU9S to see the parameter settings.



Go to the "Serial Port Mapping" area. Among them, we select "Virtual Serial Port1" in the drop-down box of COM1 in "Serial Port Mapping". And then, select "Virtual Serial Port2" in the drop-down box of COM2 in "Serial Port Mapping".



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



These actions are to set the QEC-R11HU9S's COM1 to virtual serial port1 of EVA, and COM2 to virtual serial port2 of EVA.

## 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.

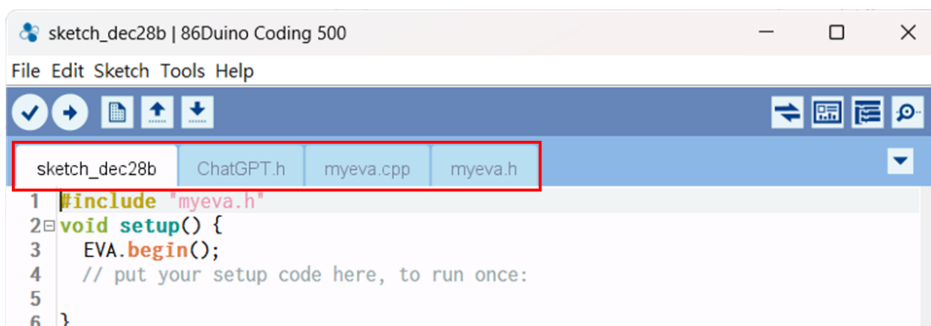


When you're done, double-click the OK button to turn off 86EVA, or it will close in 10 seconds.



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 the following example, we utilized the Ethercat library and the RTCZero library to integrate EtherCAT communication and real-time clock functionality. Initially, we configured two RS485 ports on the QEC-R11HU9S with specific baud rates and data formats. Then, in the main `loop()` function, we continuously send the current date and time from one port (COM1) every second and read it back from the other port (COM2). This setup can be used for applications requiring synchronized time stamps or logs that need precise time data, conducted through RS485 communication.

```
#include "myeva.h"
#include "RTCZero.h" // Include the RTCZero library
RTCZero rtc; // Create an RTC instance

int incomingByte = 0; // Variable for storing incoming serial data
char dateString[100]; // Array to hold the formatted date string



void setup() {
    EVA.begin();
    Serial.begin(115200);
    VirtualSerial1.begin(115200);
    VirtualSerial2.begin(115200);
    rtc.begin(); // Initialize the real-time clock
    Serial.println("...OK");
}

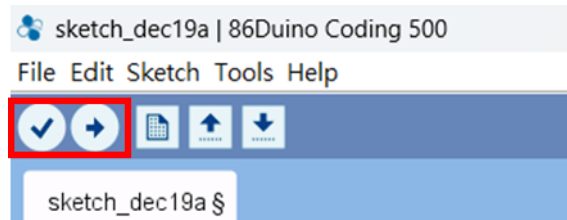
int slen; // Variable to store the length of the date string
void loop() {
    // Format the current date and time into a string
    sprintf(dateString, "%d/%02d/%02d - %02d:%02d:%02d", rtc.getYear() + 2000, rtc.getMonth(), rtc.getDay(),
    rtc.getHours(), rtc.getMinutes(), rtc.getSeconds()); // Assemble a string of year, month, day, hour,
    minute, second
    slen = strlen(dateString); // Calculate the length of the date string

    VirtualSerial1.print(dateString); // COM Port Test: Send from EtherCAT HID COM1 to HID COM2

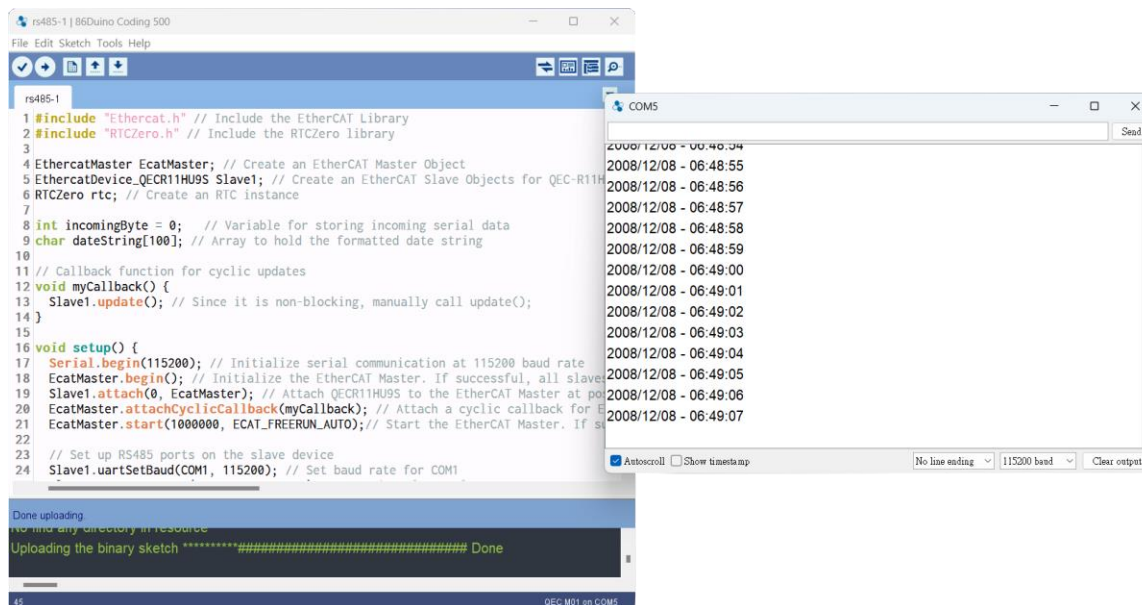
    // Display received characters
    for (int i = 0; i < slen + 1; i++)
        Serial.print((char)VirtualSerial2.read()); // Read and print the received data from COM2
    Serial.println(); // Print a newline for readability

    delay(1000); // Wait for a second before the next loop iteration
}
```

**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.



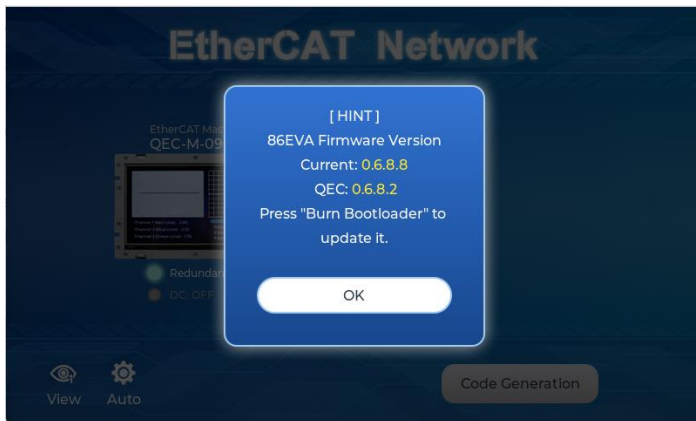
After uploading, you can view the current date and time received from RS485 COM2 through the Serial Monitor of the 86Duino IDE. This time is sent from RS485 COM1 and is updated every second. This setup can be used for applications requiring synchronized time stamps or logs that need precise time data, conducted through RS485 communication.



# Troubleshooting

## QEC-M-01 cannot successfully upload code

When you are unable to successfully upload code, please open 86EVA to check if your QEC EtherCAT Master's environment is abnormal. As shown in the figure below, please try updating your QEC EtherCAT Master's environment, which will include the following three items: Bootloader, EtherCAT firmware, and EtherCAT tool.



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

### Step 1: Setting up QEC-M

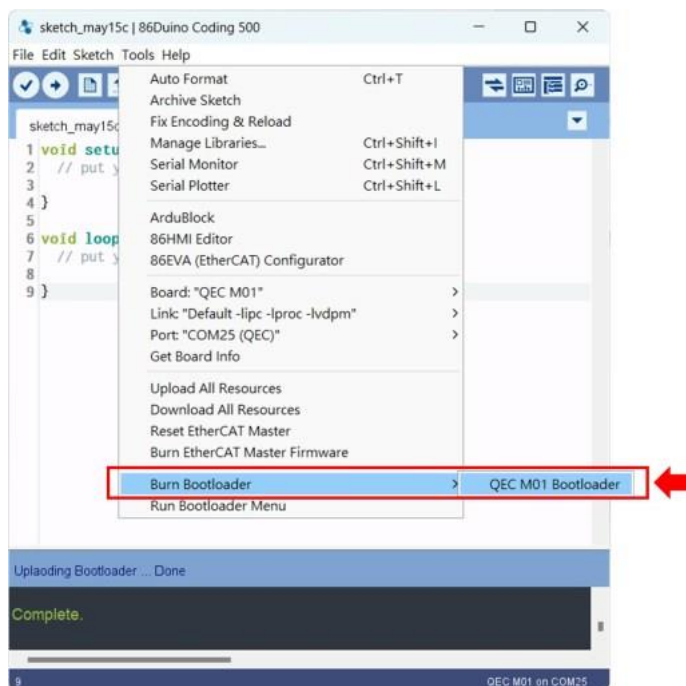
1. Download and install 86Duino IDE 500 (or a newer version): You can download it from [Software](#).
2. Connect the QEC-M: Use a USB cable to connect the QEC-M to your computer.
3. Open 86Duino IDE: After the installation is complete, open the 86Duino IDE software.
4. Select Board: From the IDE menu, 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.



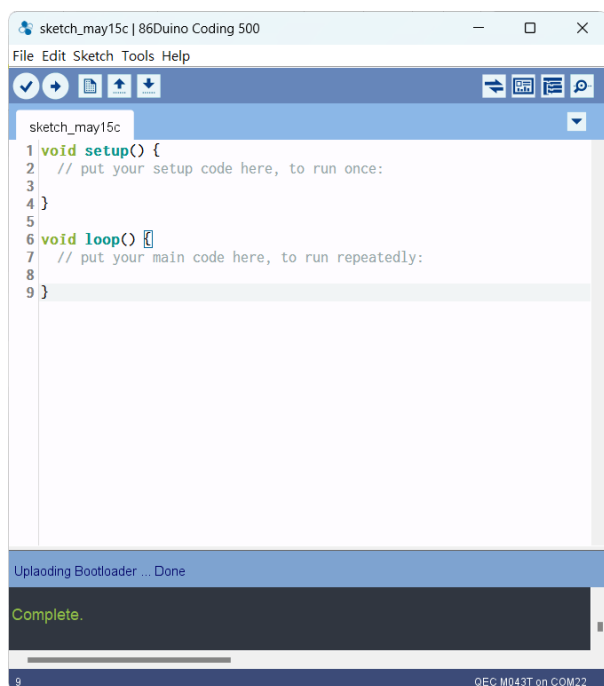
## Step 2: Click “Burn Bootloader” button

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.

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