Start Guide

QEC-R11HU9S : EtherCAT Gateway RS232/485 + Keypad + LCM + MPG with 86EVA

86Duino Coding IDE 501

EtherCAT Library

(Version 2.0)

Revision

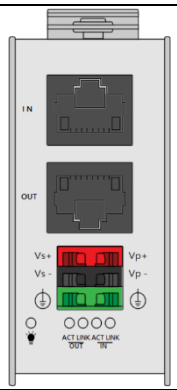
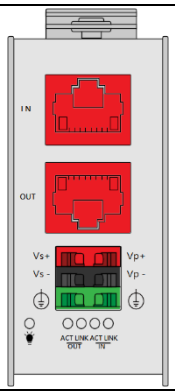
Date	Version	Description
2024/2/2	Version1.0	New release.
2024/10/8	Version1.1	Split the development steps into two documents.
2025/11/5	Version2.0	Combine all functions of HU9S in one document.

Preface

In this guide, we will show you how to use the EtherCAT MDevice **QEC-M-01** and the **QEC-R11HU9S** series (EtherCAT SubDevice, with RS232/485, Keypad, LCM, and MPG Hand wheel).

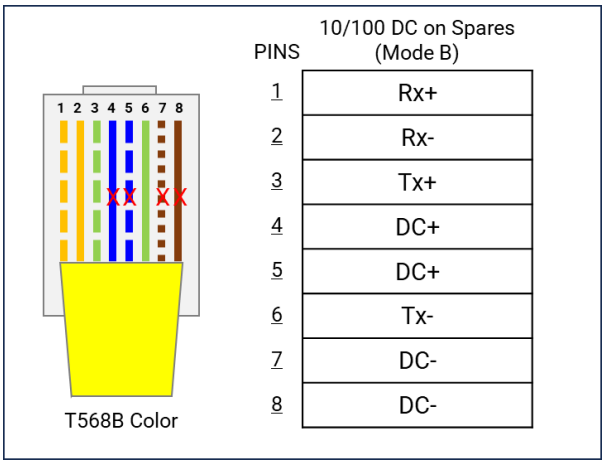
Notes QEC's PoE (Power over Ethernet)

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.

	
Non-PoE type	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:

1. 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:



PINS	10/100 DC on Spares (Mode B)
1	Rx+
2	Rx-
3	Tx+
4	DC+
5	DC+
6	Tx-
7	DC-
8	DC-

2. 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 MDevice connects with a third-party EtherCAT SubDevice).

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

1. Connection and wiring hardware

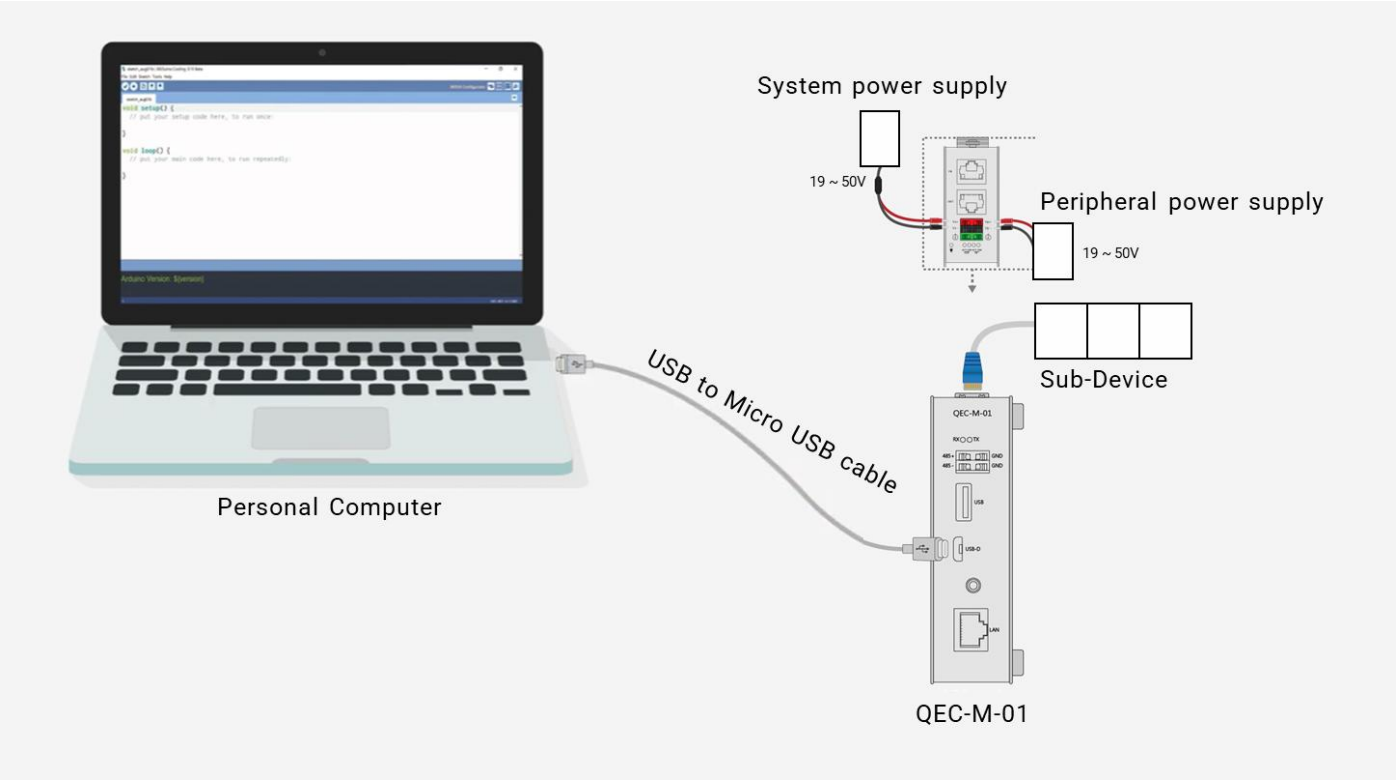
The following devices are used here:

1. QEC-M-01 (EtherCAT MDevice)
2. QEC-R11HU9S series (EtherCAT SubDevice, with RS232/485, Keypad, LCM, and MPG Hand wheel)
3. 24VDC power supply & EU-type terminal cable & LAN cable

1.1 QEC-M-01

QEC-M-01 is an EtherCAT MDevice.

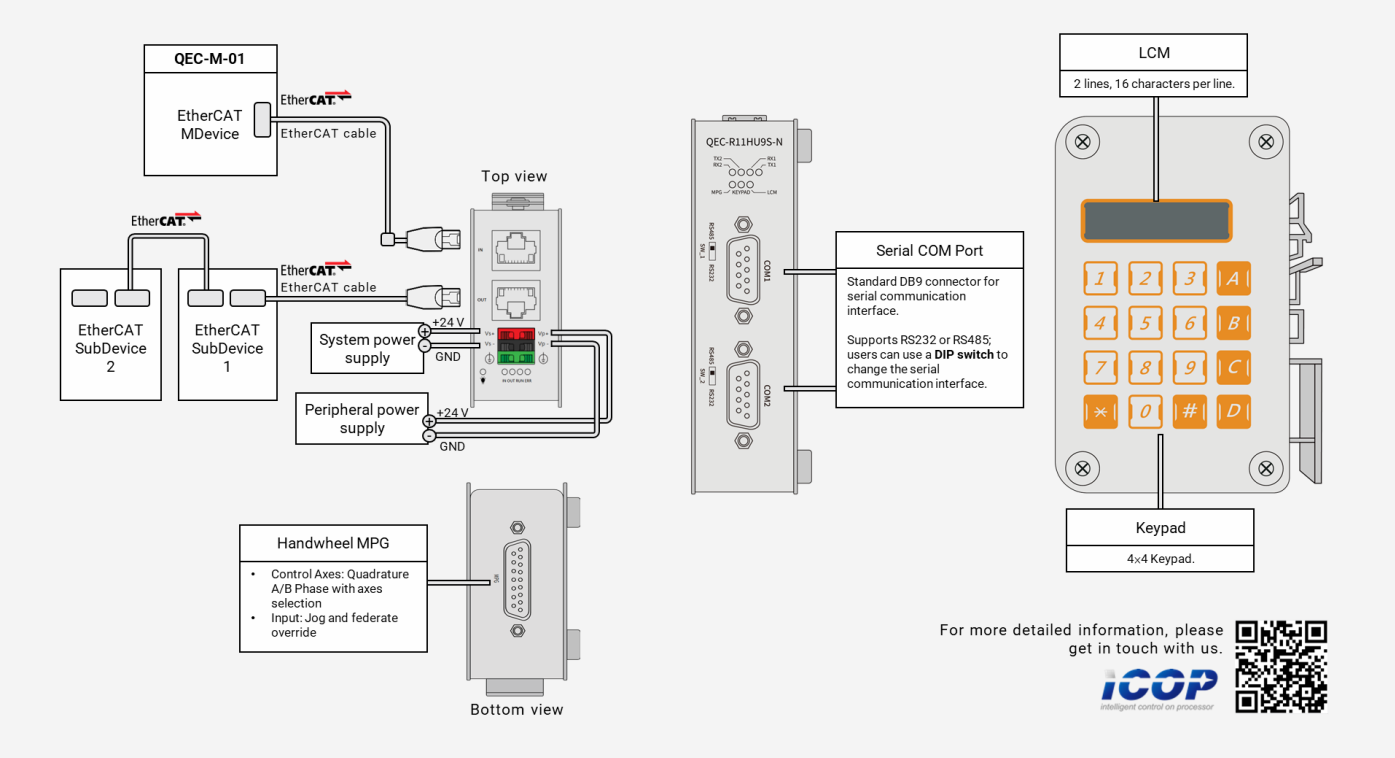
1. Power Supply: Connect to Vs+/Vs- and Vp+/Vp- power supplies via EU terminals for 24V power.
2. EtherCAT Connection: Using the EtherCAT Out port (On the top side) connected to the EtherCAT In port of EtherCAT SubDevice via RJ45 cable.



1.2 QEC-R11HU9S

The **QEC-R11HU9S** is an EtherCAT SubDevice module with RS232/485, Keypad, LCM, and MPG Hand wheel.

The diagram shows a typical wiring example with a QEC MDevice and an EtherCAT network.

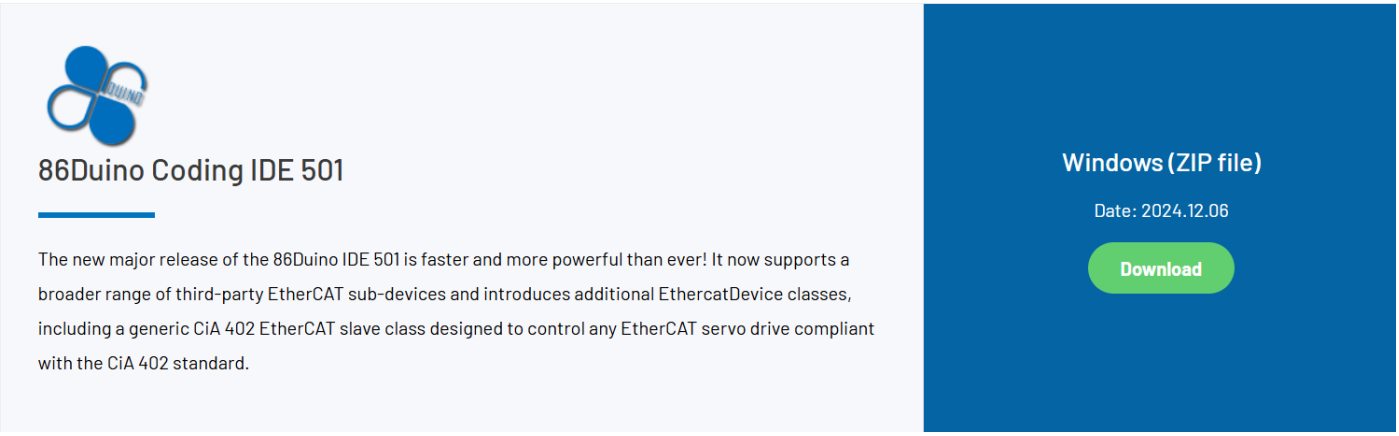


Connections are grouped by function:

- EtherCAT: MDevice → IN; OUT for daisy-chain
- Power & Grounding:
  - VS+/VS−: system power +24 V/GND
  - VP+/VP−: field I/O power +24 V/GND
- Serial COM: DB9; DIP switch selects RS-232/RS-485
  For RS-485, use termination/bias and a shared reference
  - Data transfer rate (bps): 2400,4800,9600,14400,19200,38400,57600,115200
  - Data width (bit): 5/6/7/8
  - Hardware Flow Control: CTS/RTS
- Handwheel MPG
  - Control Axes: Quadrature A/B Phase with axes selection
  - Input: Jog and federate override
- Keypad: Matrix 4×4
- LCM: 2 lines, 16 characters per line
- Indicators: PWR, RUN, LINK, ERROR, TX/RX/MPG/KEYPAD/LCM status

2. Software/Development Environment

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



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

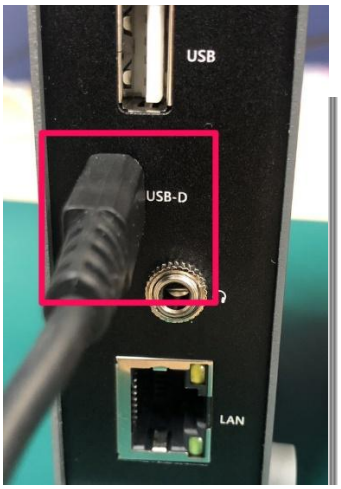
*__Note__: If Windows displays a warning, click Details once and then click the Continue Run button once.

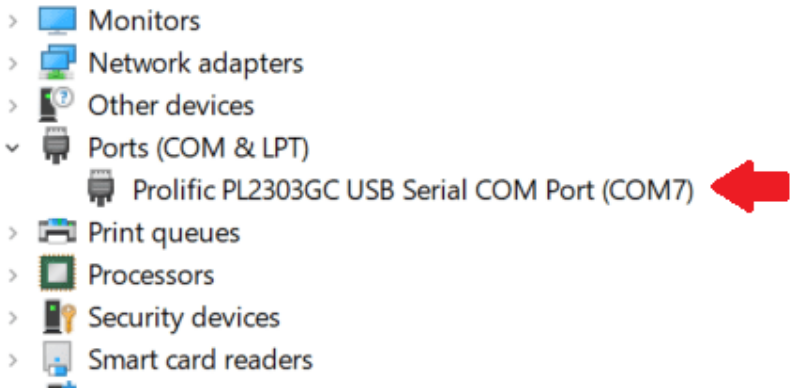
86Duino Coding IDE 501+ looks like below.

3. Connect to PC and set up the environment

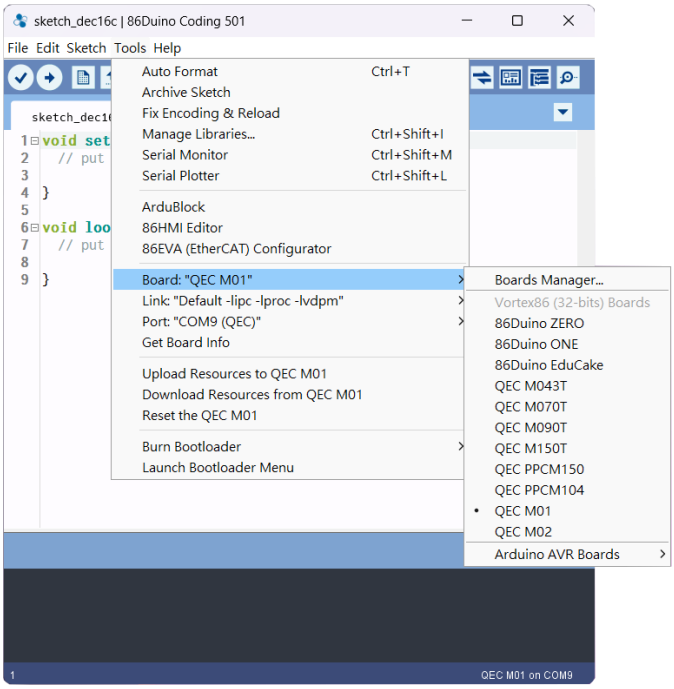
Follow the steps below to set up the environment:

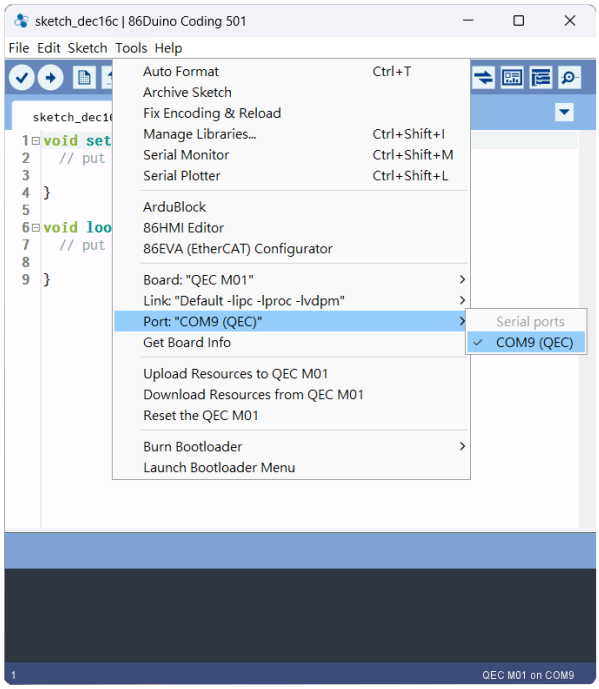
1. Connect the QEC-M-01 to your PC via a Micro USB to USB cable (86Duino IDE installed).
2. Turn on the QEC power.
3. Open “**Device Manager**” (select in the menu after pressing Win+X) ->” **Ports (COM & LPT)**” in your PC and expand the ports; you should see that the “**Prolific PL2303GC USB Serial 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-M01**” (or the QEC MDevice model you use).
6. Select Port: In the IDE's menu, select “**Tools**” > “**Port**” and select the USB port to connect to the QEC MDevice (in this case, COM9 (QEC)).





4. Use 86EVA with code

This example shows how to operate the EtherCAT MDevice (QEC-M-01) and the QEC-R11HU9S series (EtherCAT SubDevice, with RS232/485, Keypad, LCM, and MPG Hand wheel) through the 86Duino IDE's graphical low-code programming tool, 86EVA.

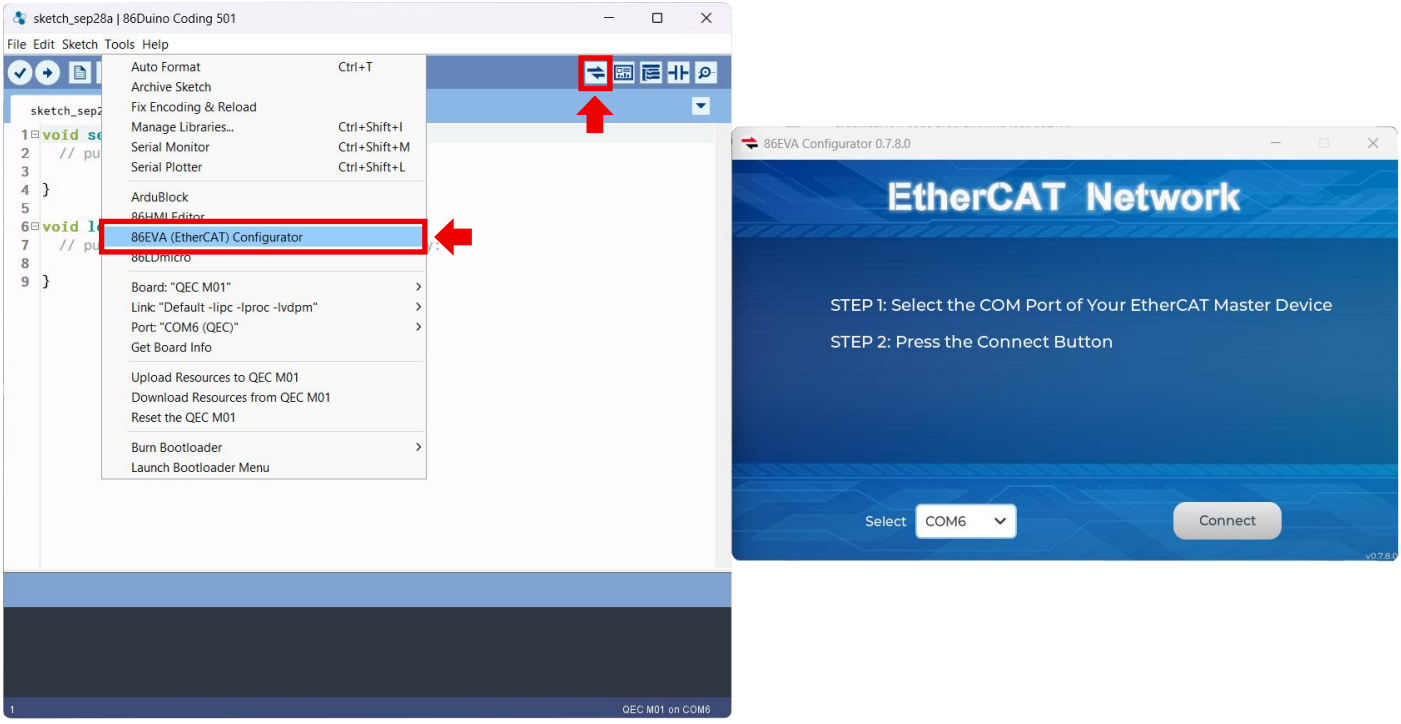
Software Tools Description:

- **86EVA (EVA, EtherCAT-Based Virtual Arduino)**:
  is a graphical EtherCAT configuration tool based on the EtherCAT Library in the 86Duino IDE and is one of the development kits for 86Duino.

We will present the Serial COM Port (RS232/RS485), Keypad and LCM, and MPG hand wheel usage in the following section by steps.

Step 1: Turn on 86EVA and scan

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



Please select the correct COM port and then click the "**Connect**" button.

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

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

Step 2: Set the parameters

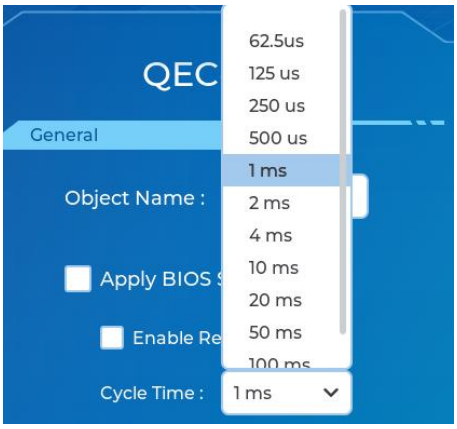
Press twice on the scanned device image to enter the corresponding parameter setting screen.

Step 2.1: QEC-M-01

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

Please check the following configures.

1. Turn off the "**Apply BIOS Settings**".
2. Select "**1ms**" to the Cycle Time.



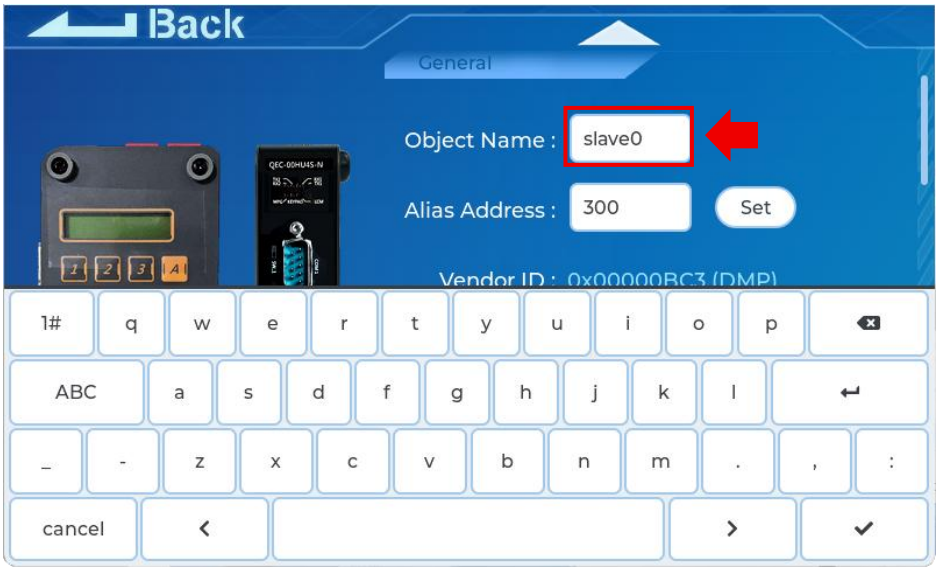
Click "**Back**" in the upper left corner to return.

Step 2.2: QEC-R11HU9S

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

The page will show the Object Name, Alias Address, Vendor ID, Product Code, Virtual Arduino Mapping, and Virtual Servo Configuration parameters.

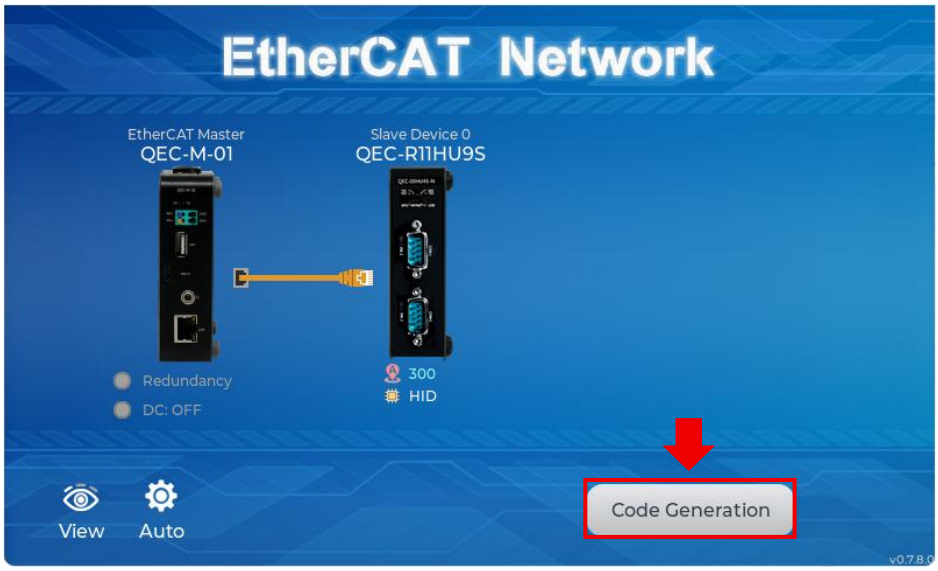
Please change the Object Name to “**slave0**”.
It’ll appear a keyboard after you click the Object Name.



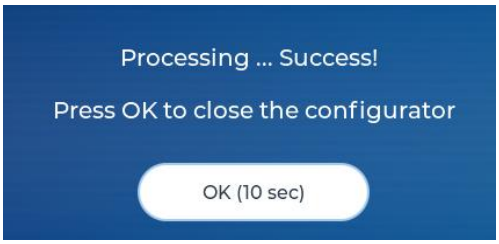
Click "**Back**" in the upper left corner to return.

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.

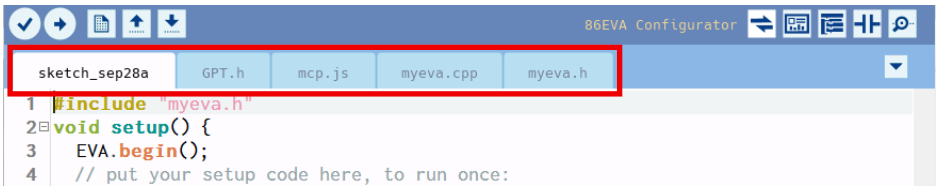


When you're done, double-click the "**OK**" button to turn off 86EVA, or it will close in 10 seconds.



The generated code and files are as follows:

- sketch_sep10b: Main Project (.ino, depending on your project name)
- 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 code

Before operating the EtherCAT network, you must configure it once. The process should be from Pre-OP to OP mode in EtherCAT devices. 86EVA will automatically handle the EtherCAT State Machine in the background.

The programming code from 86EVA are set as the following by default:

- QEC-R11HU9S module: `EthercatDevice_QECR11HU9S` object.
- EtherCAT mode: ECAT_SYNC.

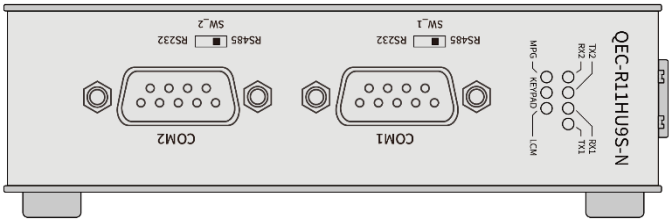
And here is the setting by users:

- EtherCAT Cycle time: 1 millisecond.
- The `EthercatMaster` object (“master”) represents the QEC-M-01, while the `EthercatDevice_QECR11HU9S` object (“slave0”) represents the QEC-R11HU9S module.

We will present the Serial COM Port (RS232/RS485), Keypad and LCM, and MPG hand wheel usage in the following section by steps.

Step 4.1: Serial COM Port (RS232/485)

In this section, we will read the data from the Serial Monitor in the 86Duino IDE and transfer it from COM1 (RS-232) to COM2 (RS-232). After COM2 receives the data, we print it on the Serial Monitor.



Wiring

- Use the front DB9 connector (see diagram for pinout).
- Connect COM1 to COM2; you can connect to the external device's TX ↔ RX, RX ↔ TX, and GND ↔ GND.
- Keep cable length reasonable; share a common reference (GND) with the device.

A. In Setup Function:

In the `setup()` function, initialize communication, enter OP, and configure COM1 and COM2. Follow the steps below:

1. Initialize Serial Communication
    - Start serial communication at a baud rate of 115200.
2. Start the 86EVA
    - Use the `EVA.begin()` function to start and initialize the EtherCAT network.
3. Configure the COM Port (RS-232)
    - Use the `uartSetBaud(COM, 115200)` function to setup COM port to baud rate 115200.
    - Use the `uartSetFormat(COM, SERIAL_8N1)` function to set COM port to SERIAL_8N1 format.

B. In Loop Function:

In the `loop()` function, keep EtherCAT/UART services pumping; forward any USB-Serial byte from the Serial Monitor to **COM1**, and non-blocking read back from **COM2**.

1. Logic
    - Call `update()` every cycle to service EtherCAT mailbox/UART queues.
    - If the Serial Monitor has a byte (`Serial.available()`), read it and send to COM1 via `uartWrite(COM1, byte)`.
    - Poll COM2 non-blockingly: if `uartQueryRxQueue(COM2) > 0`, fetch bytes with `uartRead(COM2)` and return them to the Serial monitor using `Serial.print()`.

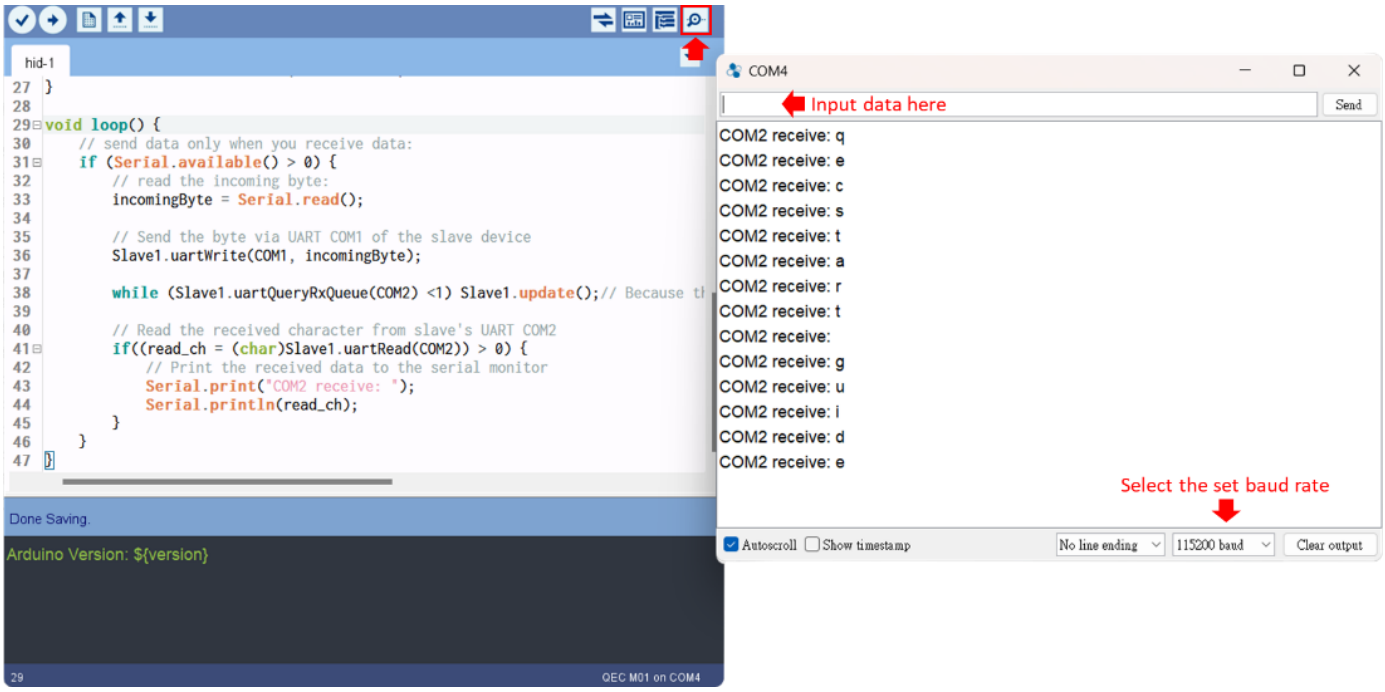
2. Code Logic Summary
    - Use `Serial.available()` to check PC-side RX FIFO; use `Serial.read()` to get incoming bytes.
    - Use `slave0.uartWrite(COM1, b)` to transmit to the RS-232 on COM1; use `slave0.uartRead(COM2)` to read COM2.
    - Use `Serial.print()` to echo COM2 data back to the PC to preserve binary data.
    - Keep all remote serial I/O inside `loop()` and call `slave0.update()` frequently to maintain EtherCAT mailbox service and UART queue progress.

The example code is as follows:

```
#include "myeva.h"

int incomingByte = 0;
char read_ch;

void setup() {
  Serial.begin(115200);
  EVA.begin();

  slave0.uartSetBaud(COM1, 115200);
  slave0.uartSetFormat(COM1, SERIAL_8N1);
  slave0.uartSetBaud(COM2, 115200);
  slave0.uartSetFormat(COM2, SERIAL_8N1);
}

void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    slave0.uartWrite(COM1, incomingByte);

    if (slave0.uartQueryRxQueue(COM2) < 1)
      slave0.update();

    if ((read_ch = (char)slave0.uartRead(COM2)) > 0) {
      Serial.print("COM2 receive: ");
      Serial.println(read_ch);
    }
  }
}
```

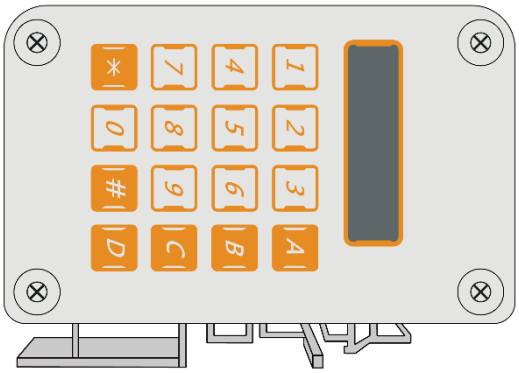
*__Note:__ Once the code is written, click on the toolbar to  compile, and to confirm that the compilation is complete and error-free, you can click  to upload.

After uploading, you can input a number or letter to the Serial Monitor in 86Duino IDE. All data will transfer from COM1 to COM2. After COM2 receives the data, we print it on the Serial Monitor, as in the image below.



Step 4.2: Keypad + LCM

In this section, we read keypad input and print it at specific positions on the LCM. The buzzer beeps whenever a key is pressed.



- # clears the LCM and sets the print row to **Row 1**.
- * clears the LCM and sets the print row to **Row 2**.
- Keys 0–9 print at columns **1–10**; keys A–D print at columns **11–14** of the current row.

A. In Setup Function:

In the `setup()` function, initialize EtherCAT communication, keypad, and LCM. Follow the steps below:

1. Initialize Serial Communication
    - Start serial communication at a baud rate of 115200.
2. Start the 86EVA
    - Use the `EVA.begin()` function to start and initialize the EtherCAT network.
3. Initialize the Keypad and LCM
    - Use the `keypadClear()` function to initialize the keypad.
    - Use the `lcmClear()` function to initialize the lcm.

B. In Loop Function:

In the `loop()` function, read the keypad non-blockingly and print to the LCM.

1. Logic
    - Read key: `key = slave0.keypadRead();` valid keys are {'0'..'9', 'A'..'D', '#', '*'}. Any other value = no key → skip.
    - Digits '0'..'9': row = lcmY, col = key - '0' + 1 ('0'→1 … '9'→10) → `lcmGotoXY(col, lcmY)` → `lcmWrite(key)`.
    - Letters 'A'..'D': row = lcmY, col = key - 'A' + 11 (11..14) → `lcmGotoXY(col, lcmY)` → `lcmWrite(key)`.
    - '#': set lcmY = 1 and `lcmClear();`
    - '*': set lcmY = 2 and `lcmClear()`.
    - Buzzer: trigger `buzzer(3000, 200)` only when the key is valid.

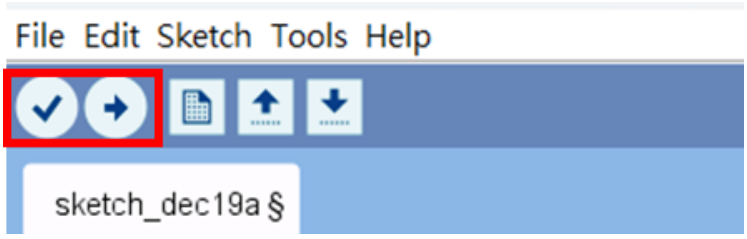
- Service: call `slave0.update()` every loop to maintain EtherCAT/mailbox.

2. Code Logic Summary
   - Use `slave0.keypadRead()` to fetch a key.
   - Use `lcmGotoXY(x, y)` then `lcmWrite(ch)` to print at a specific column/row (1-based).
   - Keep all I/O inside `loop()` and call `slave0.update()` frequently to maintain EtherCAT timing.

The example code is as follows:

```
#include "myeva.h"

int lcmY = 1;

void setup() {
  EVA.begin();

  slave0.keypadClear();
  slave0.lcmClear();
}

void loop() {
  char keyPadInput = slave0.keypadRead();

  if (keyPadInput >= '0' && keyPadInput <= '9') {
    slave0.lcmGotoXY(keyPadInput - '0' + 1, lcmY);
    slave0.lcmWrite(keyPadInput);
  } else if (keyPadInput >= 'A' && keyPadInput <= 'D') {
    slave0.lcmGotoXY(keyPadInput - 'A' + 11, lcmY);
    slave0.lcmWrite(keyPadInput);
  } else if (keyPadInput == '#') {
    lcmY = 1;
    slave0.lcmClear();
  } else if (keyPadInput == '*') {
    lcmY = 2;
    slave0.lcmClear();
  }

  if (keyPadInput != 0) slave0.buzzer(3000, 200);

  slave0.update();
}
```

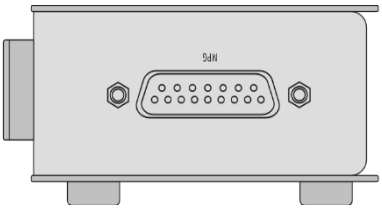
***Note:** Once the code is written, click on the toolbar to compile, and to confirm that the compilation is complete and error-free, you can click to upload.



After uploading, you can press the keypad on the QEC-R11HU9S-N's side and printing on the specific positions of LCM according to it. Buzzer will buzz when the keypad is pressed; among them, '#' is a clear LCM display and sets the print position to the first row, and '*' is a clear LCM display and sets the print position to the second row.

Step 4.3: MPG Hand wheel

In this section, we will read the MPG data and status for QEC-R11HU9S and print EMG, Enable, Axis, Ratio, Raw, and Logical data through the Serial Monitor of the 86Duino IDE.



No.	Pin Assignment	No.	Pin Assignment
1	VCC	9	AXIS_B0
2	A	10	AXIS_B1
3	A-	11	AXIS_B2
4	B	12	MULTIPLE_B0
5	B-	13	MULTIPLE_B1
6	C	14	EMERGENCY
7	C-	15	LED
8	GND	-	-

A. In Setup Function:

In the `setup()` function, initialize EtherCAT communication, keypad, and LCM. Follow the steps below:

1. Initialize Serial Communication
    - Start serial communication at a baud rate of 115200.
2. Start the 86EVA
    - Use the `EVA.begin()` function to start and initialize the EtherCAT network.

B. In Loop Function:

In the `loop()` function, we will continuously poll MPG signals and print them; keep EtherCAT services pumping.

1. Logic
    - Call `update()` every cycle to service EtherCAT mailbox/queues.
    - Read MPG signals from the SubDevice:
        - `mpgReadEmergencyStop()` → emergency stop status (EMG).
        - `mpgReadEnableSwitch()` → enable/hold switch status.
        - `mpgReadAxis()` → currently selected axis (e.g., 0/1/2…).
        - `mpgReadRatio()` → handwheel ratio (e.g., ×1/×10/×100).
        - `mpgReadEncoderRaw()` → raw encoder counts.

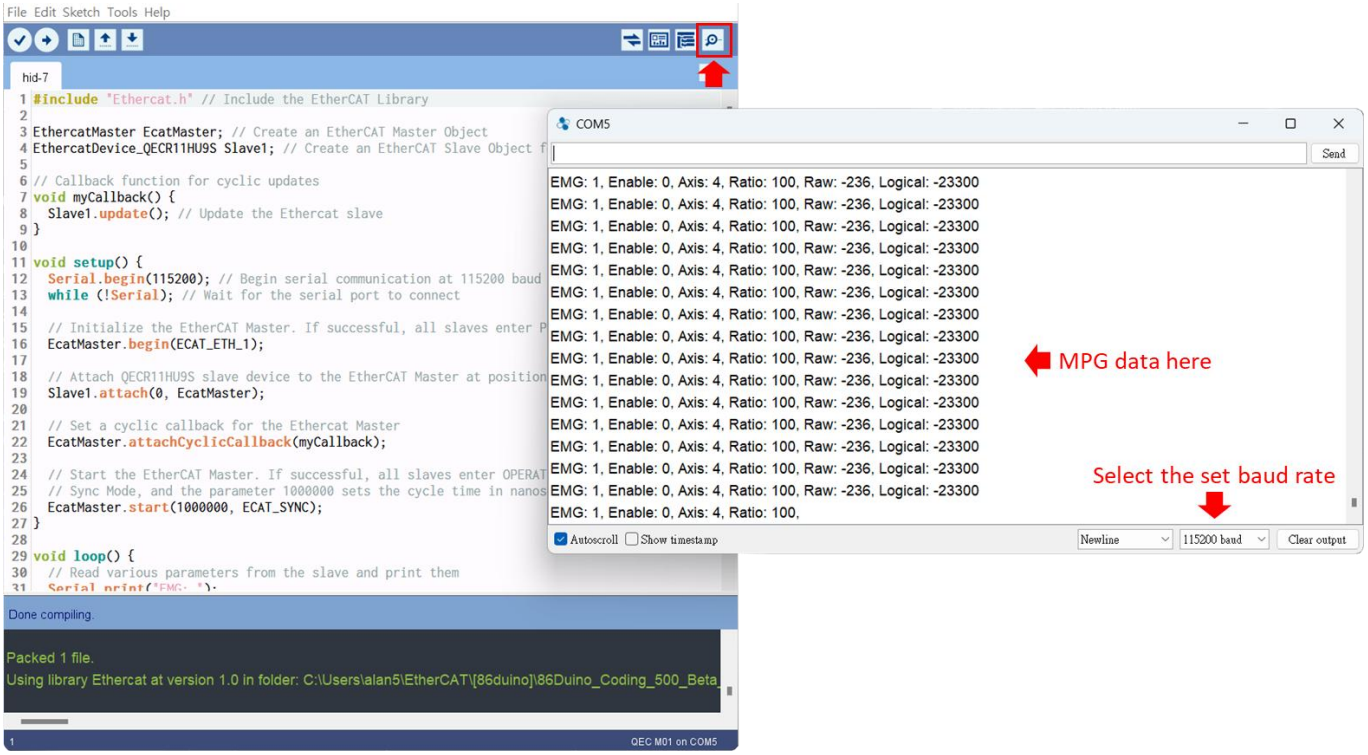
    - `mpgReadEncoder()` → logical/processed encoder value.
  - Using `Serial.print()` to print the values to the Serial Monitor for live diagnostics.
2. Code Logic Summary
  - Use `Serial.print`/`println()` to format a single diagnostic line per cycle.
  - Use MPG accessors (`mpgRead*()`) to fetch each field individually.
  - Call `slave0.update()` each iteration so the MPG/encoder values stay fresh and mailbox traffic progresses.

The example code is as follows:

```
#include "myeva.h"

void setup() {
  Serial.begin(115200);
  EVA.begin();
}

void loop() {
  // Read various parameters from the slave and print them
  Serial.print("EMG: ");
  Serial.print(slave0.mpgReadEmergencyStop()); // Read and print the
emergency stop status
  Serial.print(", Enable: ");
  Serial.print(slave0.mpgReadEnableSwitch()); // Read and print the enable
switch status
  Serial.print(", Axis: ");
  Serial.print(slave0.mpgReadAxis()); // Read and print the axis information
  Serial.print(", Ratio: ");
  Serial.print(slave0.mpgReadRatio()); // Read and print the ratio
  Serial.print(", Raw: ");
  Serial.print(slave0.mpgReadEncoderRaw()); // Read and print the raw encoder
value
  Serial.print(", Logical: ");
  Serial.println(slave0.mpgReadEncoder()); // Read and print the logical
encoder value

  slave0.update();
}
```

*__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.

After the upload is completed, you can read the data and status of the MPG of QEC-R11HU9S, and view EMG, Enable, Axis, Ratio, Raw, Logical and other data through the Serial Monitor of 86Duino IDE.



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 MDevice's environment is abnormal. As shown in the figure below, please try updating your QEC EtherCAT MDevice'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:

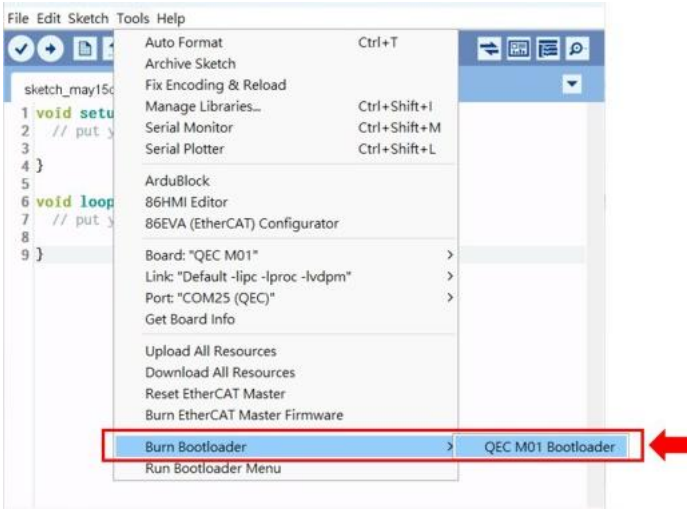
Step 1: Setting up QEC-M

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.

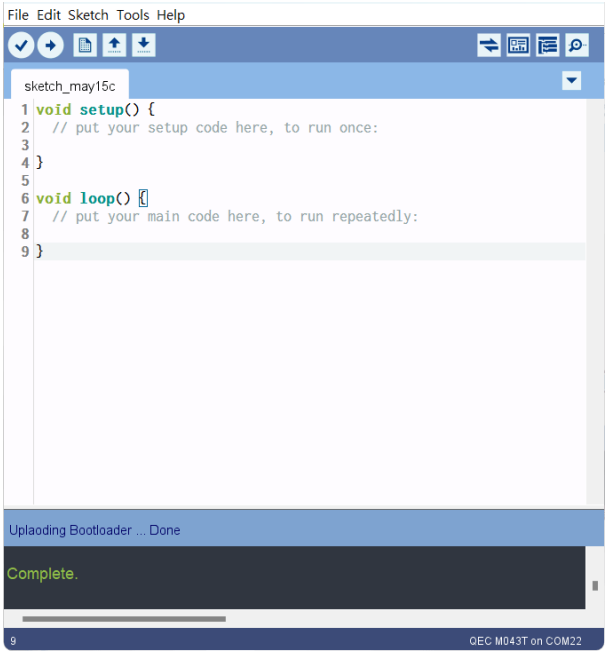
Step 2: Click “Burn Bootloader” button

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.

Warranty

This product is warranted to be in good working order for a period of one year from the date of purchase. Should this product fail to be in good working order at any time during this period, we will, at our option, replace or repair it at no additional charge except as set forth in the following terms. This warranty does not apply to products damaged by misuse, modifications, accident or disaster. Vendor assumes no liability for any damages, lost profits, lost savings or any other incidental or consequential damage resulting from the use, misuse of, originality to use this product. Vendor will not be liable for any claim made by any other related party. Return authorization must be obtained from the vendor before returned merchandise will be accepted. Authorization can be obtained by calling or faxing the vendor and requesting a Return Merchandise Authorization (RMA) number. Returned goods should always be accompanied by a clear problem description.