# Start Guide

## QEC-R11CFFG : EtherCAT Remote Digital I/O, Analog I/O, and Serial

86Duino Coding IDE 501

EtherCAT Library

(Version 1.0)

# Revision

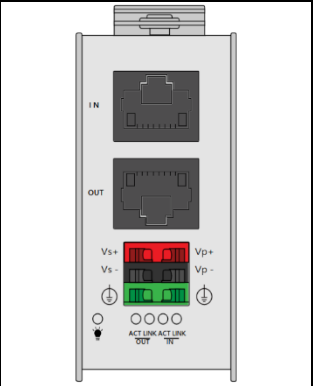
| Date | Version | Description |
|---|---|---|
| **2025/9/23** | Version1.0 | New Release. |

# Preface

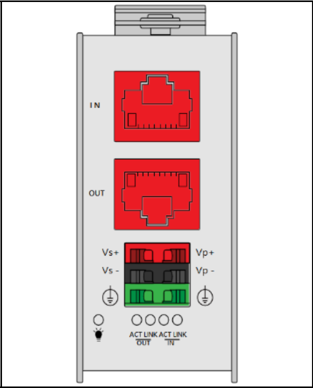
In this guide, we will show you how to use the EtherCAT MDevice QEC-M-01 and the QEC-R11CFFG series (EtherCAT Compound I/O module, with DIO + AIO + Serial COM port).

## 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.



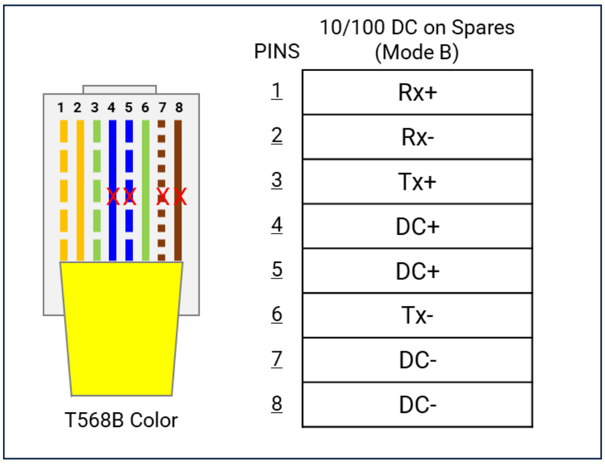
Non-PoE type



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- |

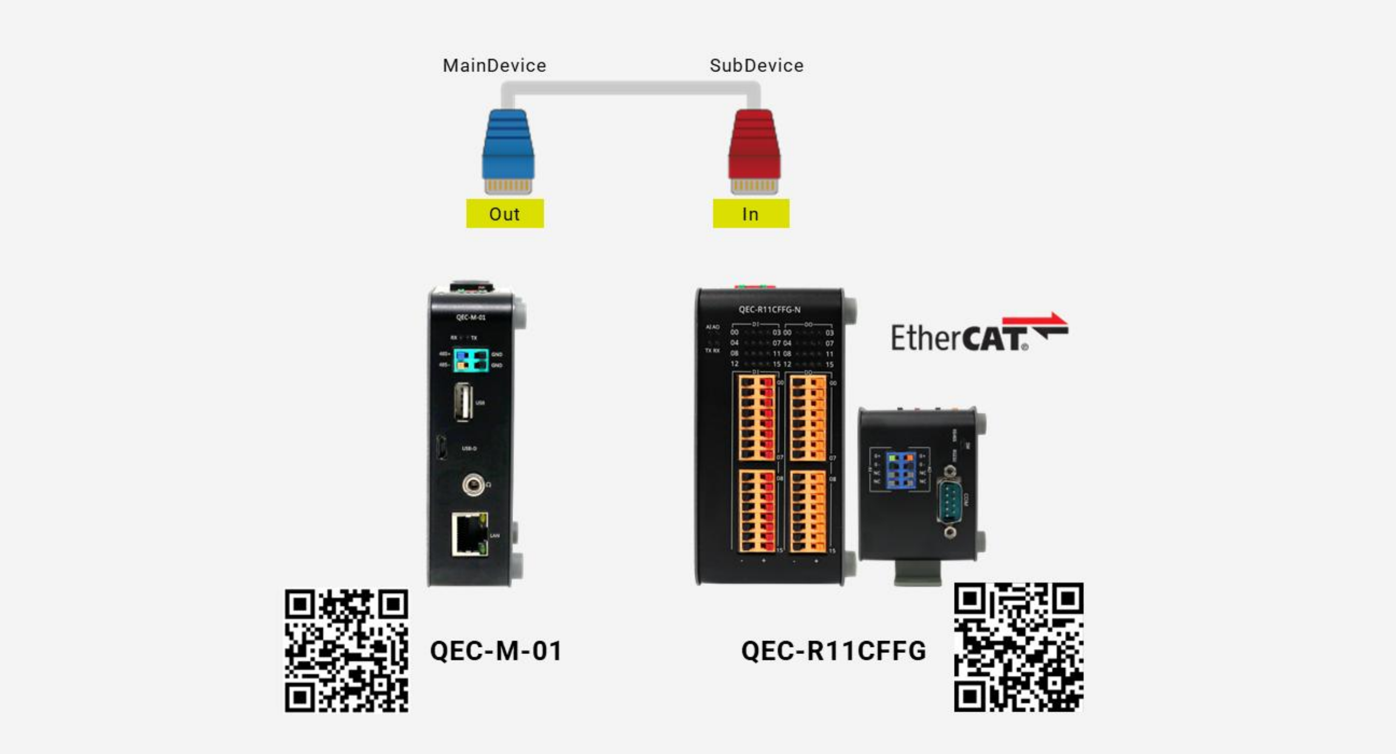
2. When connecting PoE and non-PoE devices, make sure to disconnect Ethernet cables at pins 4, 5, 7, and 8 (e.g., when a PoE-supported QEC EtherCAT MDevice connects with a third-party EtherCAT SubDevice).

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

# 1. Connection and wiring hardware

The following devices are used here:

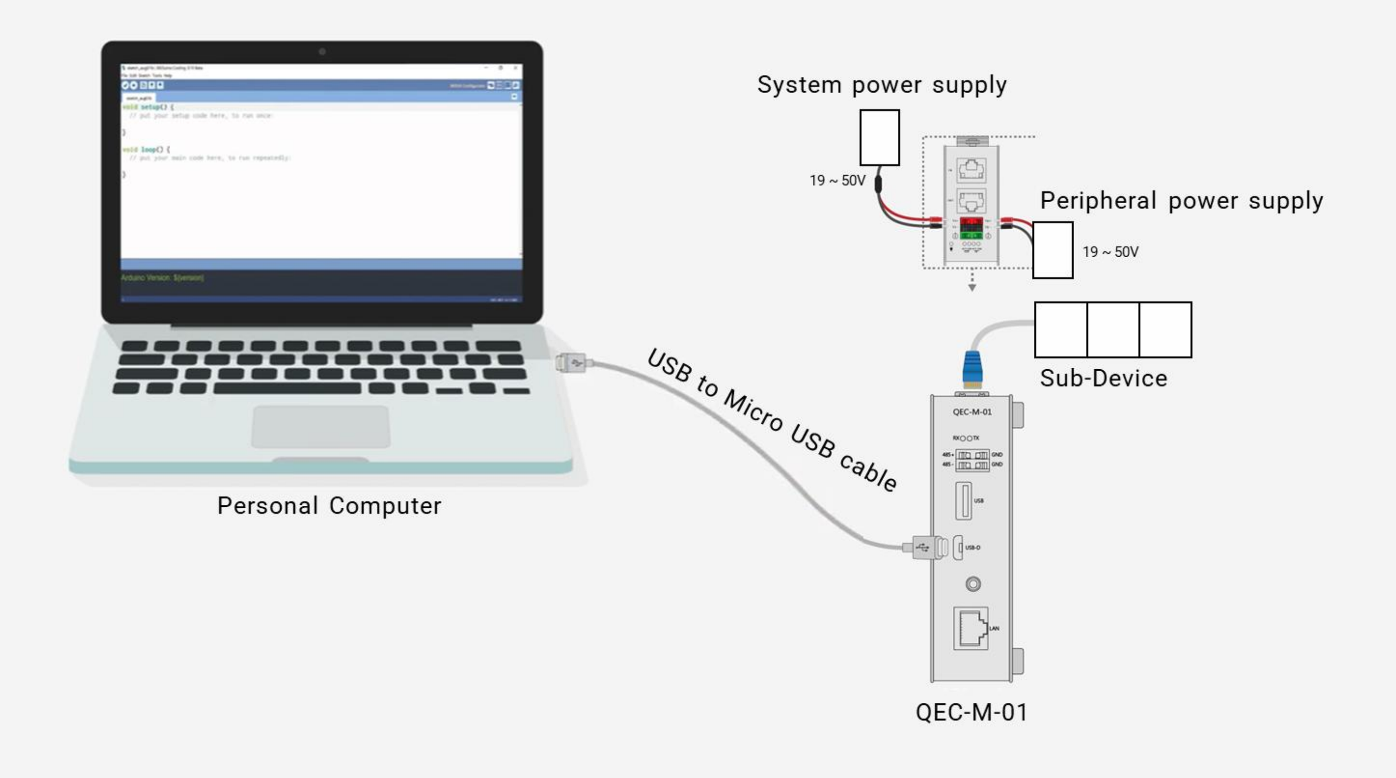
1. QEC-M-01 (EtherCAT MDevice)
2. QEC-R11CFFG series (EtherCAT Compound I/O module, with DIO + AIO + Serial COM port)
3. 24VDC power supply & EU-type terminal cable & LAN cable



## 1.1 QEC-M-01

QEC 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.

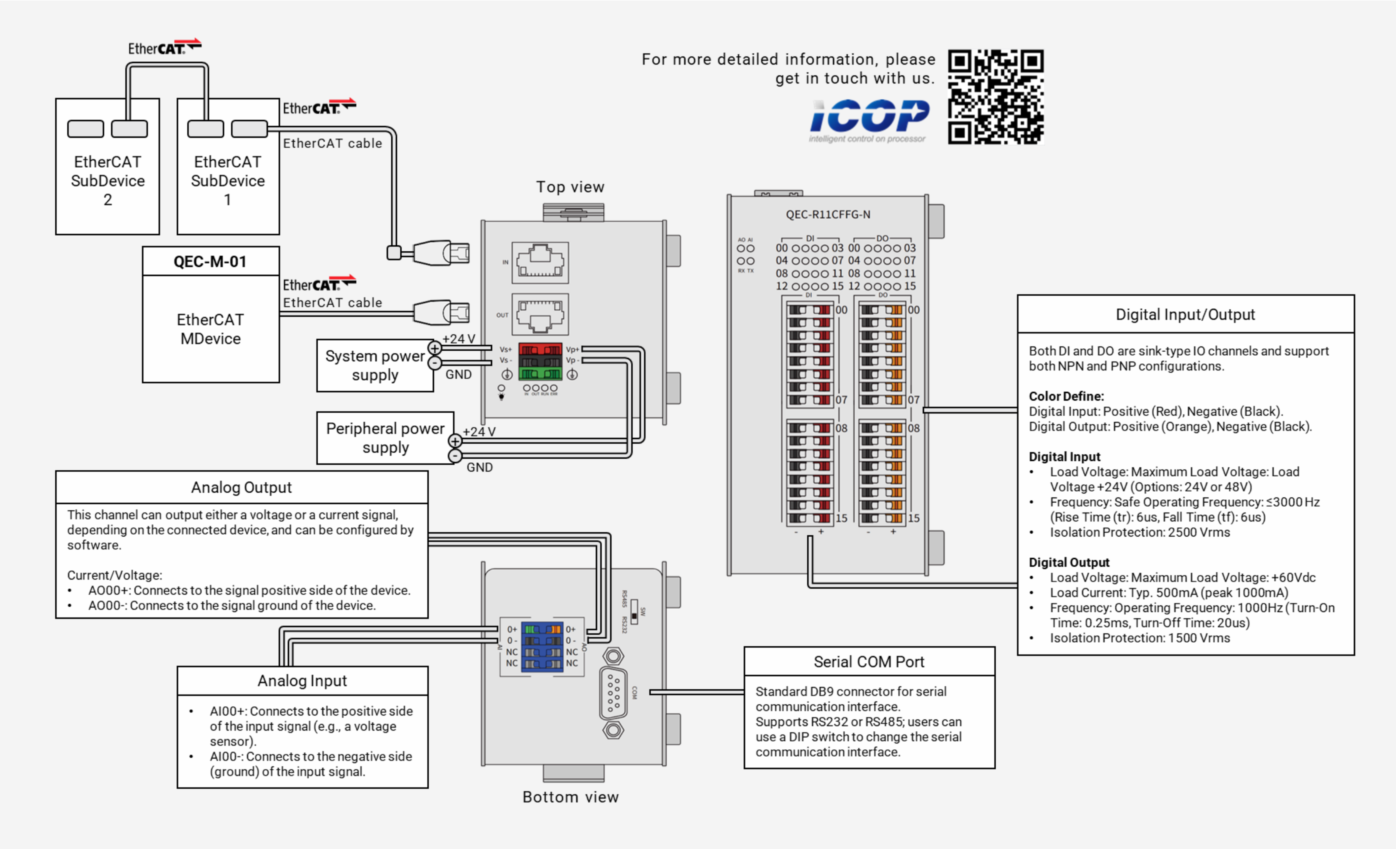


```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## 1.2 QEC-R11CFFG

The **QEC-R11CFFG** is an EtherCAT SubDevice that combines isolated 32-ch Digital I/O (DI16/DO16), analog I/O slots (voltage/current per configuration), and a serial COM port (RS-232/RS-485 selectable).

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



Digital Input/Output

Both DI and DO are sink-type IO channels and support both NPN and PNP configurations.

**Color Define:**
Digital Input: Positive (Red), Negative (Black).
Digital Output: Positive (Orange), Negative (Black).

**Digital Input**

- Load Voltage: Maximum Load Voltage: Load Voltage +24V (Options: 24V or 48V)
- Frequency: Safe Operating Frequency: ≤3000 Hz (Rise Time (tr): 6us, Fall Time (tf): 6us)
- Isolation Protection: 2500 Vrms

**Digital Output**

- Load Voltage: Maximum Load Voltage: +60Vdc
- Load Current: Typ. 500mA (peak 1000mA)
- Frequency: Operating Frequency: 1000Hz (Turn-On Time: 0.25ms, Turn-Off Time: 20us)
- Isolation Protection: 1500 Vrms

Analog Output

This channel can output either a voltage or a current signal, depending on the connected device, and can be configured by software.

Current/Voltage:

- AO00+: Connects to the signal positive side of the device.
- AO00-: Connects to the signal ground of the device.

Analog Input

- AI00+: Connects to the positive side of the input signal (e.g., a voltage sensor).
- AI00-: Connects to the negative side (ground) of the input signal.

Serial COM Port

Standard DB9 connector for serial communication interface.
Supports RS232 or RS485; users can use a DIP switch to change the serial communication interface.

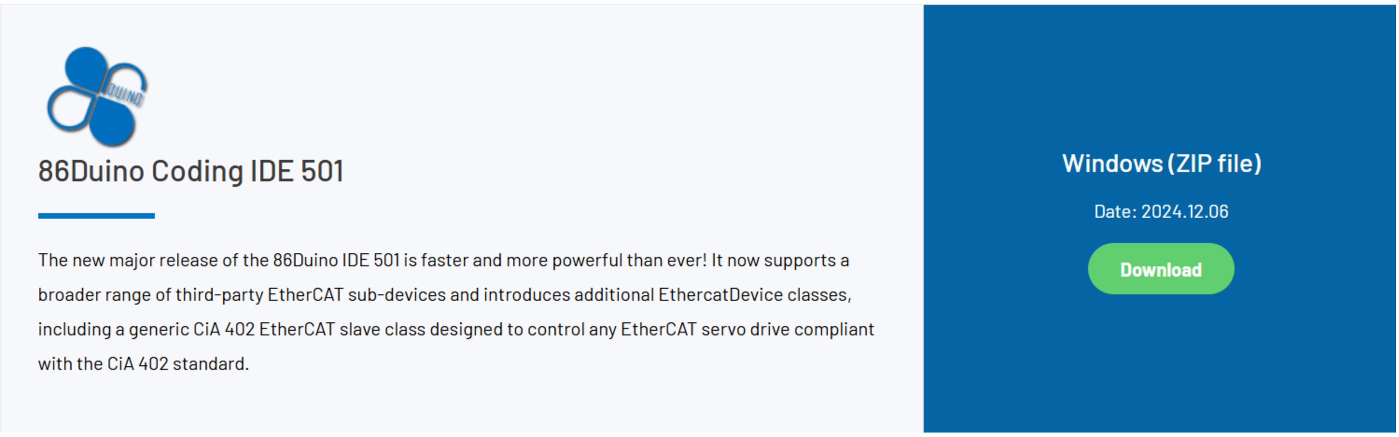
Connections are grouped by function:

- 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
- Digital I/O:
  - DI 16 (sink type): supports NPN/PNP; safe op ≤ 3 kHz, tr/tf 6 µs; input +24 V (options 24/48 V).
  - DO 16: up to +60 Vdc, typ. 500 mA (peak 1 A); 1 kHz (Ton 0.25 ms / Toff 20 µs). Add flyback/snubber for inductive loads.
  - Terminal groups: DI00–07 / DI08–15, DO00–07 / DO08–15, each with + / –.
  - Isolation: DI 2500 Vrms; DO 1500 Vrms.
- Analog I/O:
  - AO voltage/current (software-selectable): AO00+ to device positive, AO00– to return.
  - AI: AI00+ to sensor positive, AI00– to sensor return.

- Serial COM: DB9; DIP switch selects RS-232/RS-485.
  For RS-485, use termination/bias and a shared reference.
- Indicators: PWR / RUN / ERR / L/A (see later section).
- Color legend: DI + Red / − Black; DO + Orange / − Black.

# 2. Software/Development Environment

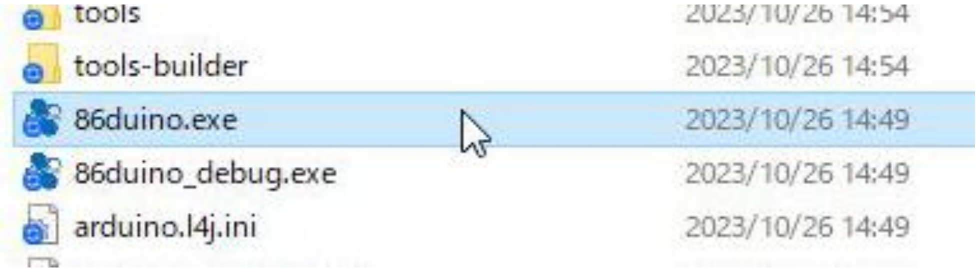
Download 86duino IDE from https://www.qec.tw/software/.



86Duino Coding IDE 501

The new major release of the 86Duino IDE 501 is faster and more powerful than ever! It now supports a broader range of third-party EtherCAT sub-devices and introduces additional EthercatDevice classes, including a generic CiA 402 EtherCAT slave class designed to control any EtherCAT servo drive compliant with the CiA 402 standard.

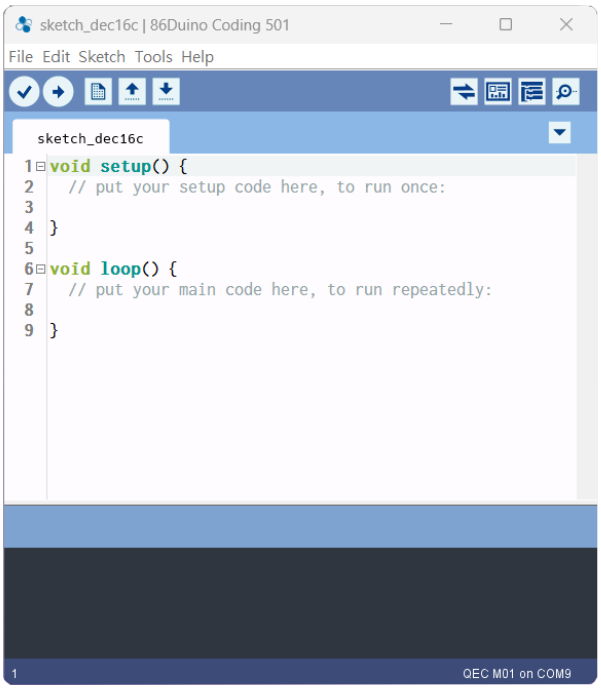
After downloading, please unzip the downloaded zip file, no additional software installation is required, just double-click 86duino.exe to start the IDE.



| | |
|---|---|
| tools | 2023/10/26 14:54 |
| tools-builder | 2023/10/26 14:54 |
| 86duino.exe | 2023/10/26 14:49 |
| 86duino_debug.exe | 2023/10/26 14:49 |
| arduino.l4j.ini | 2023/10/26 14:49 |

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

86Duino Coding IDE 501+ looks like below.



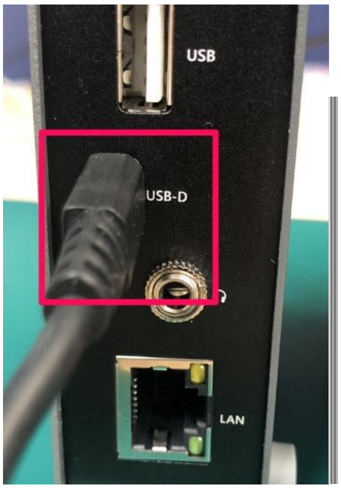
sketch_dec16c | 86Duino Coding 501
File Edit Sketch Tools Help
sketch_dec16c
1
QEC M01 on COM9

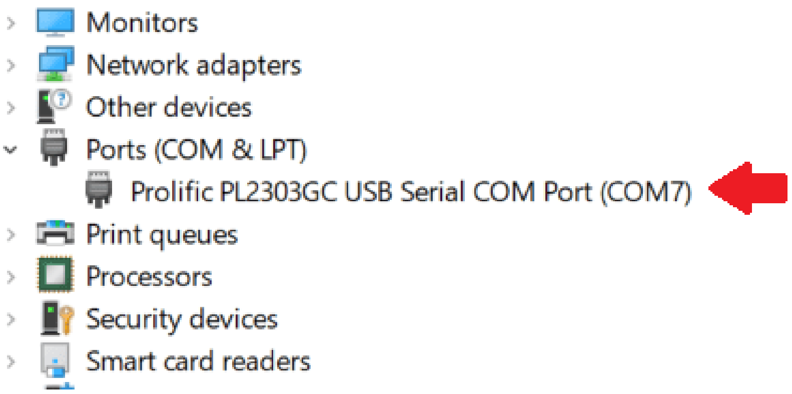
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

# 3. Connect to PC and set up the environment

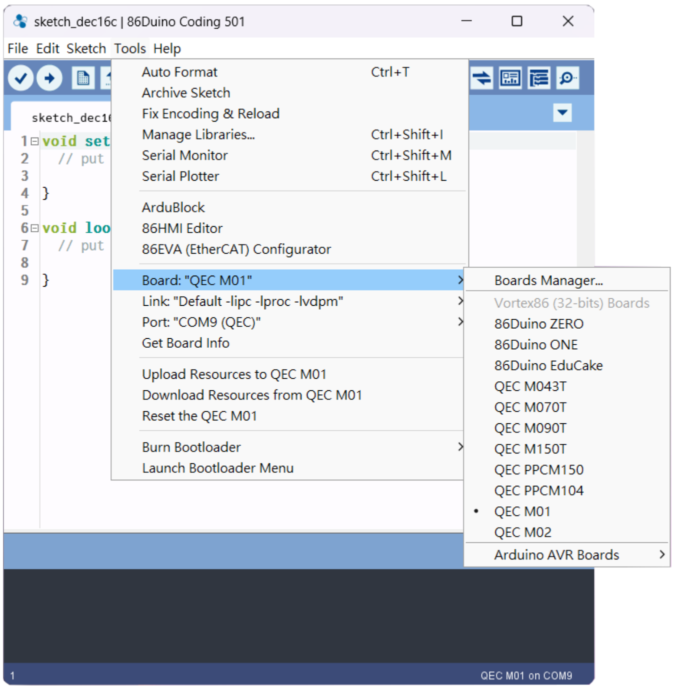
Follow the steps below to set up the environment:

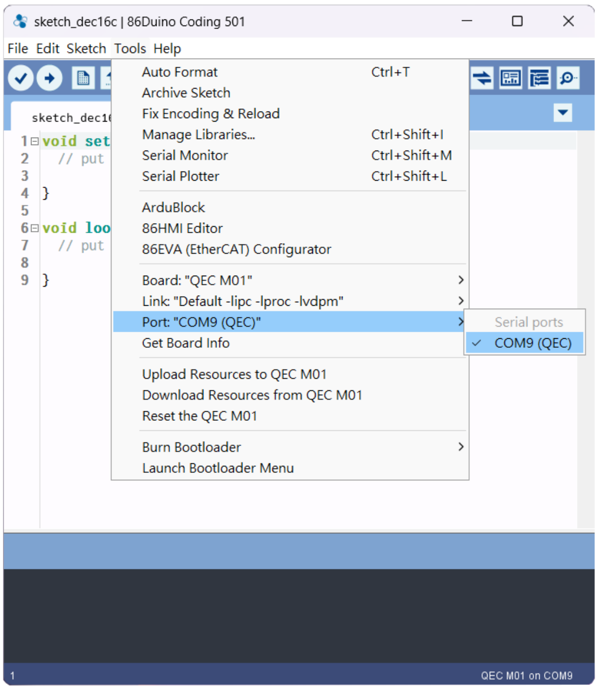
1. Connect the QEC-M-01 to your PC via a Micro USB to USB cable (86Duino IDE installed).
2. Turn on the QEC power.
3. Open “**Device Manager**” (select in the menu after pressing Win+X) ->” **Ports (COM & LPT)**” in your PC and expand the ports; you should see that the “**Prolific PL2303GC USB Serial 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)).




sketch_dec16c | 86Duino Coding 501
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
ArduBlock
86HMI Editor
86EVA (EtherCAT) Configurator
Board: "QEC M01"
Link: "Default -lipc -lproc -lvdpm"
Port: "COM9 (QEC)"
Get Board Info
Upload Resources to QEC M01
Download Resources from QEC M01
Reset the QEC M01
Burn Bootloader
Launch Bootloader Menu
Serial ports
✓ COM9 (QEC)
QEC M01 on COM9

# 4. Write code

The EtherCAT MDevice (QEC-M-01) and the QEC-R11CFFG (EtherCAT Compound I/O module, with DIO + AIO + Serial COM port) 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.

The following program sets:

- EtherCAT Cycle Time: 1 millisecond.
- EtherCAT Mode: ECAT_SYNC.

The EthercatMaster object (“master”) represents the QEC-M-01, while the EthercatDevice_QECR11CFFG object (“slave0”) represents the QEC-R11CFFG module.

We will present the Digital I/O, Analog I/O, and Serial COM Port (RS232/RS485) usage in the following section by steps.

## 4.1 Digital I/O

In this section, we will read digital input DI00 (e.g., button) and mirror its state to digital output DO00 (e.g., LED/load).

**A. In Setup Function:**

In the `setup()` function, initialize communication and configure the bring the EtherCAT network up to OP mode. Follow the steps below:

1. Initialize Serial Communication
   - Start serial communication at a baud rate of 115200.
2. Start the EtherCAT MDevice
   - Use the `begin()` function to begin the EtherCAT MDevice and set the EtherCAT state machine to the PRE-OPERATIONAL state.
3. Attach the QEC-R11CFFG EtherCAT SubDevice
   - Use the `attach()` function to attach the EtherCAT SubDevice to the EtherCAT Network. Set the node number and the specific MDevice.
4. Start the EtherCAT MDevice
   - Use the `start(1000000, ECAT_SYNC)` function to switch the EtherCAT state machine to the OPERATIONAL state. Set the cycle time to 1ms and ECAT_SYNC mode.

**B. In Loop Function:**

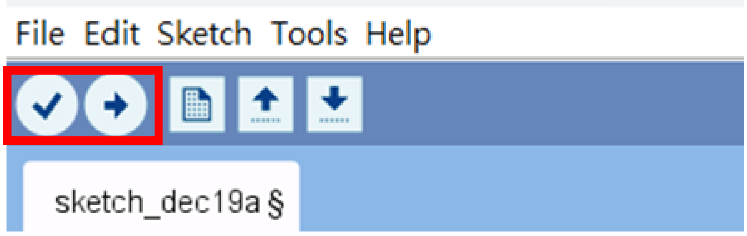
In the `loop()` function, read DI00 and mirror its state to DO00 continuously.

1. Logic
   - If `digitalRead(0)` is HIGH, set `digitalWrite(0, HIGH);` otherwise set `digitalWrite(0, LOW)`.
2. Code Logic Summary
   - Use `digitalRead(n)` to read DI channel n (0–15).
   - Use `digitalWrite(n, HIGH/LOW)` to drive DO channel n (0–15).

The example code is as follows:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave0;

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

  Serial.print("Begin: ");
  Serial.println(master.begin());
  Serial.print("Slave: ");
  Serial.println(slave0.attach(0, master));

  Serial.print("Start: ");
  Serial.println(master.start(1000000, ECAT_SYNC));
}

void loop() {
  if (slave0.digitalRead(0) == HIGH)
    slave0.digitalWrite(0, HIGH);
  else
    slave0.digitalWrite(0, LOW);
}
```

***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.



## 4.2 Analog I/O

In this section, we will read analog input AI00 (voltage) and sweep analog output AO00 from 0 V → 5 V in 1-V steps every second.

**A. In Setup Function:**

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

1. Initialize Serial Communication
   - Start serial communication at a baud rate of 115200.
2. Start the EtherCAT MDevice
   - Use the `begin()` function to begin the EtherCAT MDevice and set the EtherCAT state machine to the PRE-OPERATIONAL state.
3. Attach the QEC-R11CFFG EtherCAT SubDevice
   - Use the `attach()` function to attach the EtherCAT SubDevice to the EtherCAT Network. Set the node number and the specific MDevice.
4. Start the EtherCAT MDevice
   - Use the `start(1000000, ECAT_SYNC)` function to switch the EtherCAT state machine to the OPERATIONAL state. Set the cycle time to 1ms and ECAT_SYNC mode.
5. Configure Analog I/O Modes
   - Use the `aiPinMode(0, 0)` function to set AI00 input range to ±22 V (default).
   - Use the `aoPinMode(0, 0)` function to set AO00 to voltage output 0–5 V (default).

**B. In Loop Function:**

In the `loop()` function, print the measured AI voltage and step the AO voltage.

1. Logic
   - Read AI00 with `voltageRead(0)` and print the value.
   - Write AO00 with `voltageWrite(0, i)`, where i sweeps 0 → 5 → 0 (in volts), updating every 1 s.
2. Code Logic Summary
   - Use `voltageRead(ch)` to get the analog input voltage of channel ch.
   - Use `voltageWrite(ch, volts)` to set the analog output voltage of channel ch.

**C. AI / AO Range Options**

Use `aiPinMode(ch, mode)` / `aoPinMode(ch, mode)` to select input/output ranges or signal type (per device options).

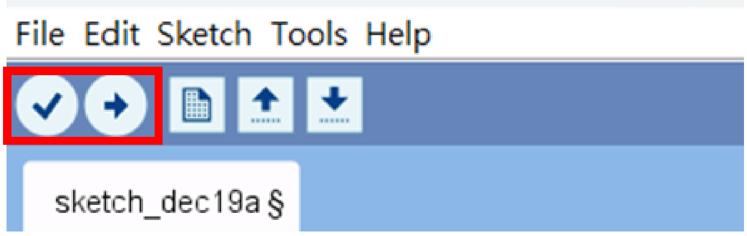
1. Default
   - ECAT_AI_PIN_MODE_N22_TO_P22_V (±22 V).
   - ECAT_AO_PIN_MODE_0_TO_5_V (0–5 V).

2. Analog Input (AI) voltage ranges
    - ECAT_AI_PIN_MODE_N22_TO_P22_V (0) — ±22 V *(default)*
    - ECAT_AI_PIN_MODE_N20_TO_P20_V (1) — ±20 V
    - ECAT_AI_PIN_MODE_N11_TO_P11_V (2) — ±11 V
    - ECAT_AI_PIN_MODE_N10_TO_P10_V (3) — ±10 V
    - ECAT_AI_PIN_MODE_N5_TO_P5_V (4) — ±5 V
    - ECAT_AI_PIN_MODE_0_TO_22_V (5) — 0–22 V
    - ECAT_AI_PIN_MODE_0_TO_20_V (6) — 0–20 V
    - ECAT_AI_PIN_MODE_0_TO_11_V (7) — 0–11 V
    - ECAT_AI_PIN_MODE_0_TO_10_V (8) — 0–10 V
3. Analog Output (AO) ranges / types
    - ECAT_AO_PIN_MODE_0_TO_5_V (0) — 0–5 V *(default)*
    - ECAT_AO_PIN_MODE_0_TO_10_V (1) — 0–10 V
    - ECAT_AO_PIN_MODE_N5_TO_P5_V (2) — ±5 V
    - ECAT_AO_PIN_MODE_N10_TO_P10_V (3) — ±10 V
    - ECAT_AO_PIN_MODE_3p5_TO_23p5_mA (4) — 3.5–23.5 mA (current)
    - ECAT_AO_PIN_MODE_0_TO_20_mA (5) — 0–20 mA (current)
    - ECAT_AO_PIN_MODE_0_TO_24_mA (6) — 0–24 mA (current)
    - ECAT_AO_PIN_MODE_N24_TO_P24_mA (7) — ±24 mA (current)
    - ECAT_AO_PIN_MODE_0_TO_6_V (8) — 0–6 V
    - ECAT_AO_PIN_MODE_0_TO_12_V (9) — 0–12 V
    - ECAT_AO_PIN_MODE_N6_TO_P6_V (10) — ±6 V
    - ECAT_AO_PIN_MODE_N12_TO_P12_V (11) — ±12 V
    - ECAT_AO_PIN_MODE_4_TO_20_mA (12) — 4–20 mA (current)

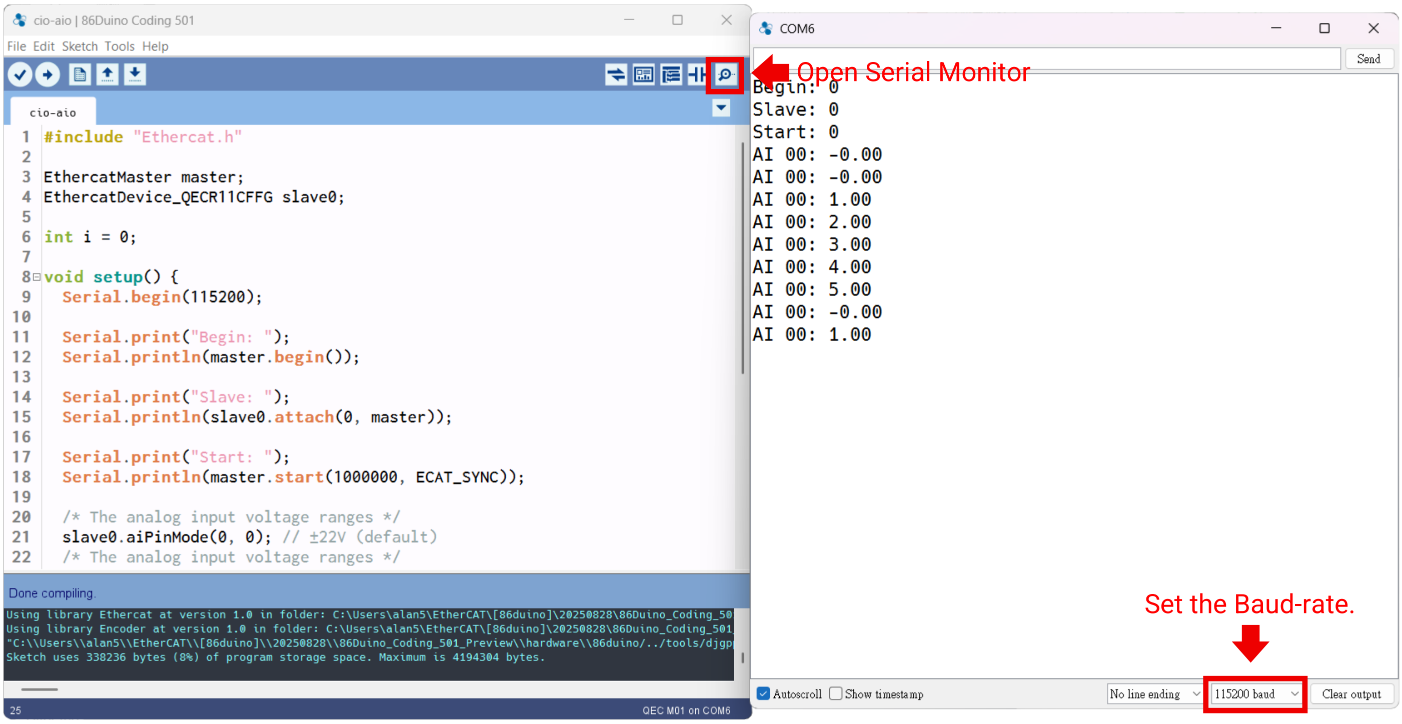
The example code is as follows:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave0;

int i = 0;

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

  Serial.print("Begin: ");
  Serial.println(master.begin());

  Serial.print("Slave: ");
  Serial.println(slave0.attach(0, master));

  Serial.print("Start: ");
  Serial.println(master.start(1000000, ECAT_SYNC));

  /* The analog input voltage ranges */
  slave0.aiPinMode(0, 0); // ±22V (default)
  /* The analog input voltage ranges */
  slave0.aoPinMode(0, 0); // Voltage output 0 to +5 V (default)

  delay(1000);
}

void loop() {
  // Print Analog Input Voltage value
  Serial.print("AI 00: ");
  Serial.println(slave0.voltageRead(0));

  // Cycle Analog Output Voltage from 0 to 5V
  slave0.voltageWrite(0, i);
  i++;
  if (i == 6) i = 0;

  delay(1000);
}
```

***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 you successfully upload the program to the QEC-M-01, you can open the Serial Monitor on the 86Duino IDE. Please check that the Serial baud rate is the same as your setting.



Open Serial Monitor

Set the Baud-rate.

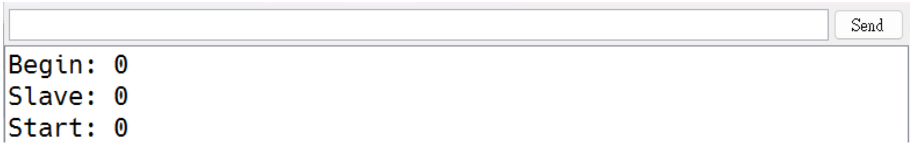
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave0;

int i = 0;

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

  Serial.print("Begin: ");
  Serial.println(master.begin());

  Serial.print("Slave: ");
  Serial.println(slave0.attach(0, master));

  Serial.print("Start: ");
  Serial.println(master.start(1000000, ECAT_SYNC));

  /* The analog input voltage ranges */
  slave0.aiPinMode(0, 0); // ±22V (default)
  /* The analog input voltage ranges */
```

Done compiling.

```
Using library Ethercat at version 1.0 in folder: C:\Users\alan5\EtherCAT\[86duino]\20250828\86Duino_Coding_50
Using library Encoder at version 1.0 in folder: C:\Users\alan5\EtherCAT\[86duino]\20250828\86Duino_Coding_501
"C:\\Users\\alan5\\EtherCAT\\[86duino]\\20250828\\86Duino_Coding_501_Preview\\hardware\\86duino/../tools/djgp
Sketch uses 338236 bytes (8%) of program storage space. Maximum is 4194304 bytes.
```

```
Begin: 0
Slave: 0
Start: 0
AI 00: -0.00
AI 00: -0.00
AI 00: 1.00
AI 00: 2.00
AI 00: 3.00
AI 00: 4.00
AI 00: 5.00
AI 00: -0.00
AI 00: 1.00
```

If the EtherCAT communication configuration is successful, the Serial Monitor will print “0” for each status for EtherCAT.



```
Begin: 0
Slave: 0
Start: 0
```

It will print the analog input AI00 value (Voltage) to the serial monitor.

```
AI 00: -0.00
AI 00: -0.00
AI 00: 1.00
AI 00: 2.00
AI 00: 3.00
AI 00: 4.00
AI 00: 5.00
AI 00: -0.00
AI 00: 1.00
```

## 4.3 Serial COM Port

In this section, we forward bytes from the USB Serial Monitor to COM1 (RS-232) on QEC-R11CFFG. In this test, no data is expected to return from RS-232, so the code only transmits and optionally prints anything that happens to arrive.

**Wiring**

- Use the front DB9 connector (see diagram for pinout).
- Connect 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. Follow the steps below:

1. Initialize Serial Communication
   - Start serial communication at a baud rate of 115200.
2. Start the EtherCAT MDevice
   - Use the `begin()` function to begin the EtherCAT MDevice and set the EtherCAT state machine to the PRE-OPERATIONAL state.
3. Attach the QEC-R11CFFG EtherCAT SubDevice
   - Use the `attach()` function to attach the EtherCAT SubDevice to the EtherCAT Network. Set the node number and the specific MDevice.
4. Start the EtherCAT MDevice
   - Use the `start(1000000, ECAT_SYNC)` function to switch the EtherCAT state machine to the OPERATIONAL state. Set the cycle time to 1ms and ECAT_SYNC mode.
5. Configure the COM Port (RS-232)
   - Use the `uartSetBaud(COM1, 115200)` function to setup COM1 to baud rate 115200.
   - Use the `uartSetFormat(COM1, SERIAL_8N1)` function to set COM1 to SERIAL_8N1 format.

**B. In Loop Function:**

In the `loop()` function, keep EtherCAT/COM services running and forward any USB Serial byte from the Serial Monitor to COM1.

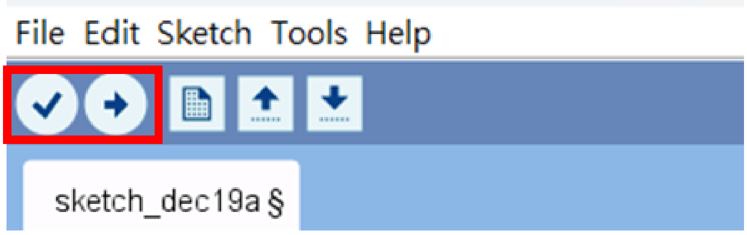
1. Logic
   - Call `update()` each cycle to update the UART queues on EtherCAT/mailbox.
   - If Serial Monitor has a byte, read it and send it to COM1 with `uartWrite(COM1, byte)`.
   - Since this test has no RS-232 reply, the code still tries a non-blocking `uartRead(COM1)` and prints only if a byte is present.

2. Code Logic Summary
    - Use `Serial.available()` to check the RX FIFO, and `Serial.read()` to get data from Serial.
    - Use `slave0.uartRead(COM1)` / `uartWrite(COM1)` to receive/transmit bytes to the Serial COM port on QEC-R11CFFG.
    - Keep remote serial I/O in `loop()` to do EtherCAT mailbox service.

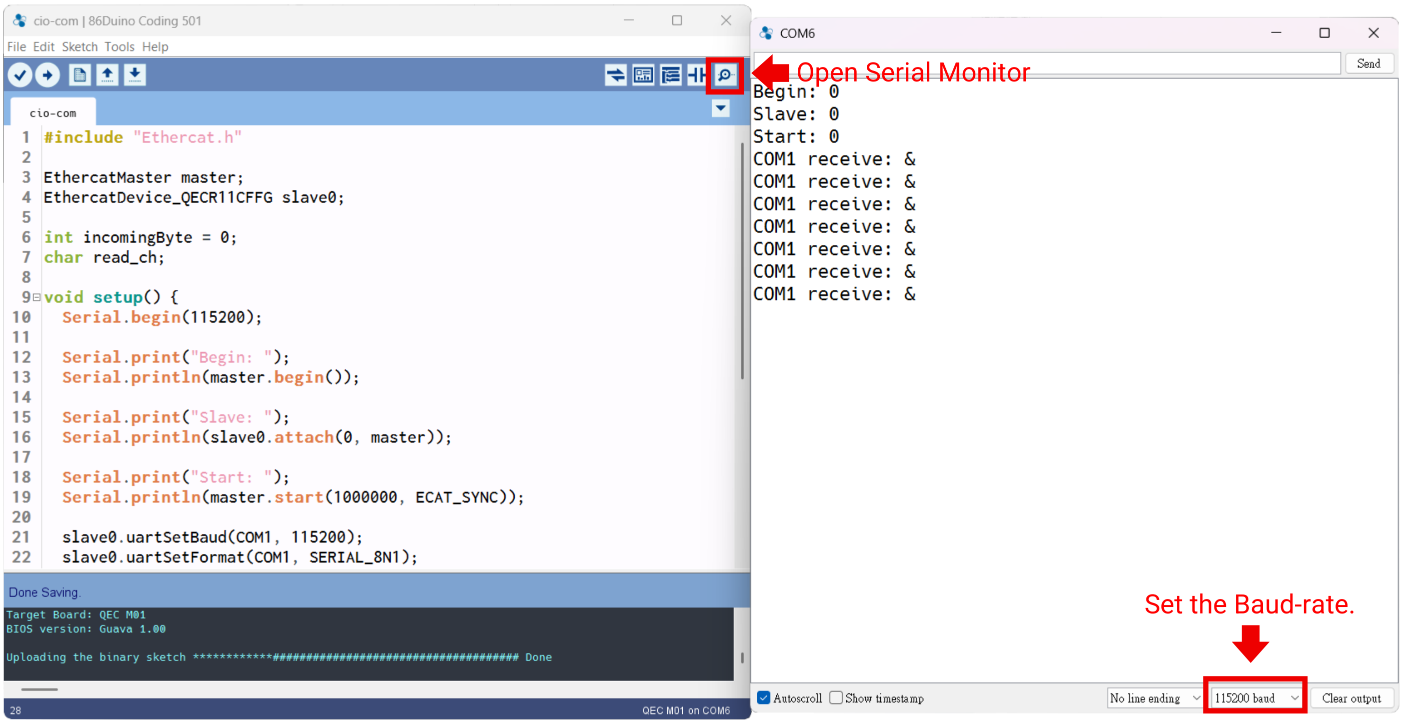
The example code is as follows:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave0;

int incomingByte = 0;
char read_ch;

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

  Serial.print("Begin: ");
  Serial.println(master.begin());

  Serial.print("Slave: ");
  Serial.println(slave0.attach(0, master));

  Serial.print("Start: ");
  Serial.println(master.start(1000000, ECAT_SYNC));

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

}

void loop() {
  slave0.update();

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

    if ((read_ch = (char)slave0.uartRead(COM1)) > 0) {
      Serial.print("COM1 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 you successfully upload the program to the QEC-M-01, you can open the Serial Monitor on the 86Duino IDE. Please check that the Serial baud rate is the same as your setting.



```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave0;

int incomingByte = 0;
char read_ch;

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

  Serial.print("Begin: ");
  Serial.println(master.begin());

  Serial.print("Slave: ");
  Serial.println(slave0.attach(0, master));

  Serial.print("Start: ");
  Serial.println(master.start(1000000, ECAT_SYNC));

  slave0.uartSetBaud(COM1, 115200);
  slave0.uartSetFormat(COM1, SERIAL_8N1);
```

```
Done Saving.
Target Board: QEC M01
BIOS version: Guava 1.00

Uploading the binary sketch ************##################################### Done
```

28 QEC M01 on COM6

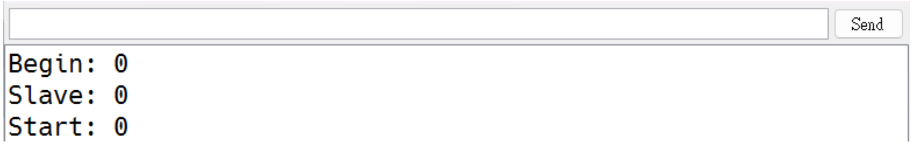
Open Serial Monitor

```
Begin: 0
Slave: 0
Start: 0
COM1 receive: &
COM1 receive: &
COM1 receive: &
COM1 receive: &
COM1 receive: &
COM1 receive: &
COM1 receive: &
```

Set the Baud-rate.

☑ Autoscroll □ Show timestamp No line ending 115200 baud Clear output

If the EtherCAT communication configuration is successful, the Serial Monitor will print “0” for each status for EtherCAT.



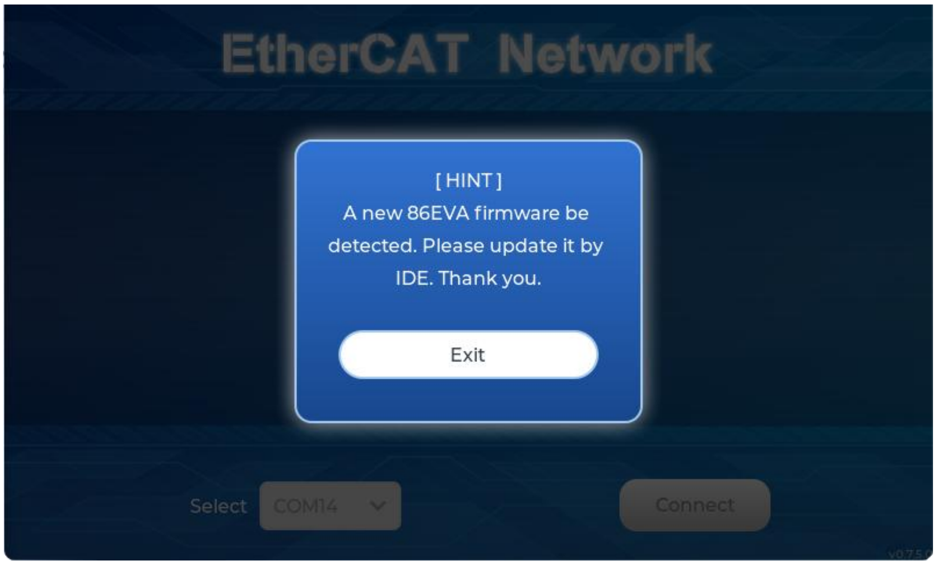
```
Begin: 0
Slave: 0
Start: 0
```

It will print the value to the serial monitor if QEC-R11CFFG’s COM port receives data.

# 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.



EtherCAT Network

[ HINT ]
A new 86EVA firmware be detected. Please update it by IDE. Thank you.

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

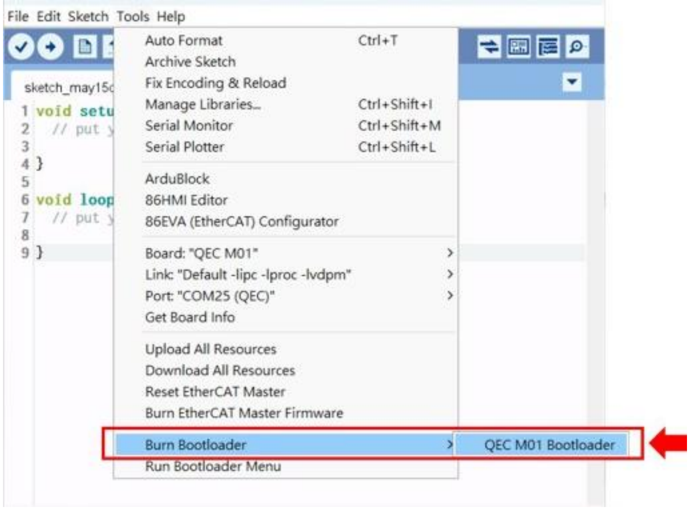
### Step 1: Setting up QEC-M

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.

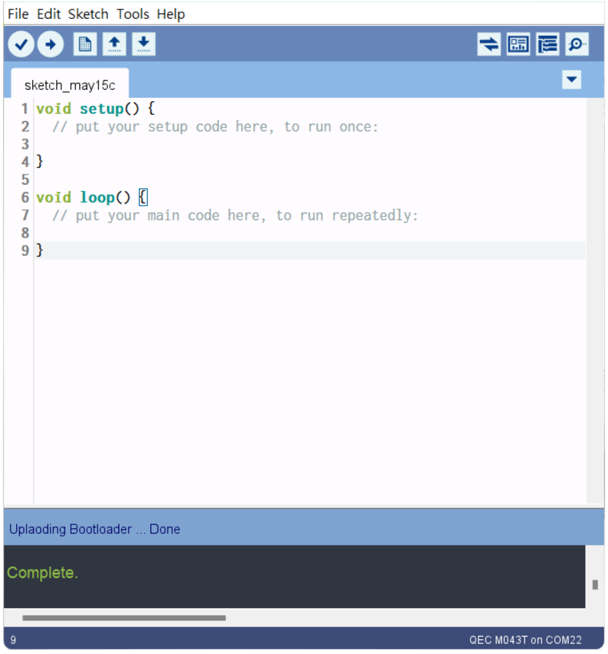
## Step 2: Click “Burn Bootloader” button

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



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
Uplaoding Bootloader ... Done
Complete.
```

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.

All Trademarks appearing in this manuscript are registered trademark of their respective owners.
All Specifications are subject to change without notice.

©ICOP Technology Inc. 2025