



User Manual

QEC-M-01

DM&P Vortex86EX2 Processor

EtherCAT Master System

(Revision 2.2)

REVISION

| DATE | VERSION | DESCRIPTION |
|------------|-------------|---------------------------------|
| 2022/06/24 | Version1.0A | First Release. |
| 2023/03/28 | Version1.1A | Modified Start Guide. |
| 2023/08/08 | Version2 | Updated Product Specifications. |
| 2023/11/05 | Version2.1 | Updated Hardware system. |
| 2024/01/14 | Version2.2 | Updated Getting Started. |

COPYRIGHT

The information in this manual is subject to change without notice for continuous improvement in the product. All rights are reserved. The manufacturer assumes no responsibility for any inaccuracies that may be contained in this document and makes no commitment to update or to keep current the information contained in this manual.

No part of this manual may be reproduced, copied, translated or transmitted, in whole or in part, in any form or by any means without the prior written permission of ICOP Technology Inc.

©Copyright 2024 ICOP Technology Inc.

Ver.2.2 January, 2024

TRADEMARKS ACKNOWLEDGMENT

ICOP® is the registered trademark of ICOP Corporation. Other brand names or product names appearing in this document are the properties and registered trademarks of their respective owners. All names mentioned herewith are served for identification purpose only.

For more detailed information or if you are interested in other ICOP products, please visit our official websites at:

- Global: www.icop.com.tw
- USA: www.icoptech.com
- Japan: www.icop.co.jp
- Europe: www.icoptech.eu
- China: www.icop.com.cn

For technical support or drivers download, please visit our websites at:

- https://www.icop.com.tw/resource_entrance

For EtherCAT solution service, support or tutorials, 86Duino Coding IDE 500+ introduction, functions, languages, libraries, etc. Please visit the QEC website:

- QEC: <https://www.qec.tw/>

This Manual is for the QEC series.

SAFETY INFORMATION

- Read these safety instructions carefully.
- Please carry the unit with both hands and handle it with caution.
- Power Input voltage +19 to +50VDC Power Input (Typ. +24VDC)
- Make sure the voltage of the power source is appropriate before connecting the equipment to the power outlet.
- To prevent the QEC device from shock or fire hazards, please keep it dry and away from water and humidity.
- Operating temperature between -20 to +70°C/-40 to +85°C (Option).
- When using external storage as the main operating system storage, ensure the device's power is off before connecting and removing it.
- Never touch un-insulated terminals or wire unless your power adaptor is disconnected.
- Locate your QEC device as close as possible to the socket outline for easy access and avoid force caused by the entangling of your arms with surrounding cables from the QEC device.
- If your QEC device will not be used for a period of time, make sure it is disconnected from the power source to avoid transient overvoltage damage.

WARNING!



DO NOT ATTEMPT TO OPEN OR TO DISASSEMBLE THE CHASSIS (ENCASING) OF THIS PRODUCT. PLEASE CONTACT YOUR DEALER FOR SERVICING FROM QUALIFIED TECHNICIAN.

CONTENT

| | |
|--|----|
| CONTENT | iv |
| Ch. 1 General Information | 1 |
| 1.1 Introduction | 2 |
| 1.1.1 QEC-M Systems Diagram | 3 |
| 1.1.2 Software Support | 3 |
| 1.2 Specifications | 4 |
| 1.3 Dimension | 5 |
| 1.4 Mounting Instruction..... | 6 |
| 1.5 Ordering Information..... | 7 |
| 1.5.1 Ordering Part Number:..... | 7 |
| Ch. 2 Hardware System | 8 |
| 2.1 General Technical Data | 9 |
| 2.2 General Summary | 9 |
| 2.2.1 EtherCAT Interface | 10 |
| 2.2.2 Power Connector..... | 11 |
| 2.2.3 Power and Connection Status LEDs..... | 12 |
| 2.2.4 RS-485 | 13 |
| 2.2.5 USB | 13 |
| 2.2.6 Micro USB..... | 14 |
| 2.2.7 Audio | 14 |
| 2.2.8 Giga LAN..... | 15 |
| 2.2.9 DIN-Rail installation..... | 15 |
| 2.3 Wiring to the Connector | 16 |
| 2.3.1 Connecting the wire to the connector | 16 |
| 2.3.2 Removing the wire from the connector..... | 16 |
| Ch. 3 Hardware Installation | 17 |
| 3.1 DIN-Rail installation | 18 |
| 3.2 Removing QEC-M-01 Unit..... | 19 |
| Ch. 4 Getting Started..... | 20 |
| 4.1 Package Contents..... | 22 |
| 4.2 Hardware Configuration | 22 |
| 4.2.1 Plug in the power supply..... | 22 |
| 4.3 Software/Development Environment | 23 |
| 4.4 Connect to your PC and set up the environment..... | 24 |
| 4.5 EtherCAT Development Method..... | 26 |
| 4.5.1 Implementing the EtherCAT State Machine | 26 |
| 4.5.2 Process Data Objects (PDO) Functions..... | 29 |

| | | |
|----------|--|----|
| 4.5.3 | Cyclic Callback..... | 32 |
| 4.5.4 | Distributed Clock (DC)..... | 35 |
| 4.5.5 | Use 86EVA with code | 38 |
| 4.6 | Troubleshooting..... | 46 |
| 4.6.1 | Old environment of your QEC-M-01: | 46 |
| Ch. 5 | Software Function..... | 48 |
| 5.1 | Software Description | 49 |
| 5.2 | EtherCAT Function List | 50 |
| 5.2.1 | EthercatMaster Class Functions..... | 50 |
| 5.2.2 | EthercatDevice Class General Functions | 52 |
| 5.3 | Additional Resources | 54 |
| Warranty | | 55 |



Ch. 1

General Information

[1.1 Introduction](#)

[1.2 Specifications](#)

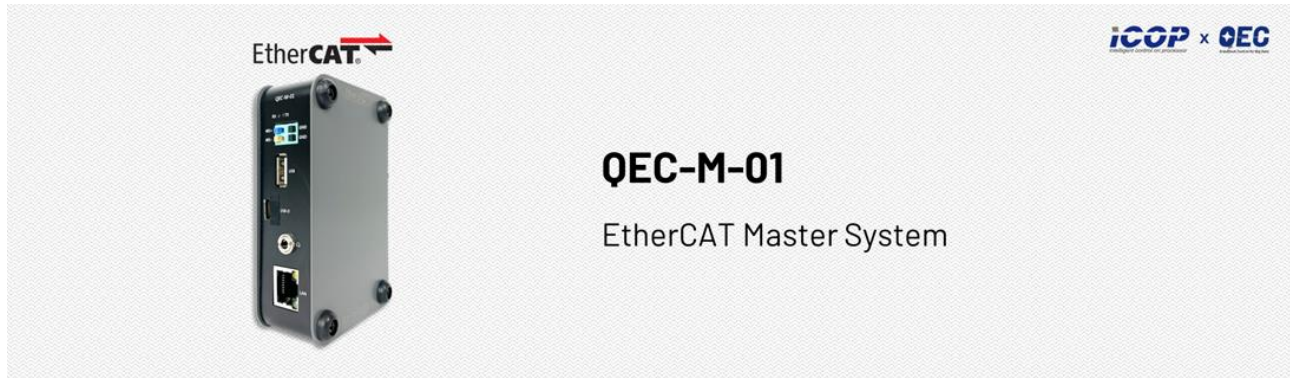
[1.3 Dimension](#)

[1.4 Mounting Instruction](#)

[1.5 Ordering Information](#)

1.1 Introduction

ICOP's QEC-M-01 is an EtherCAT master system based on Vortex86EX2 processor. The development environment uses industrial-Arduino software, 86Duino IDE, supporting EtherCAT API, which performs real-time field monitoring and big data collection and supports graphical programming tools, making it easy to hire software engineers and shorten the market time.



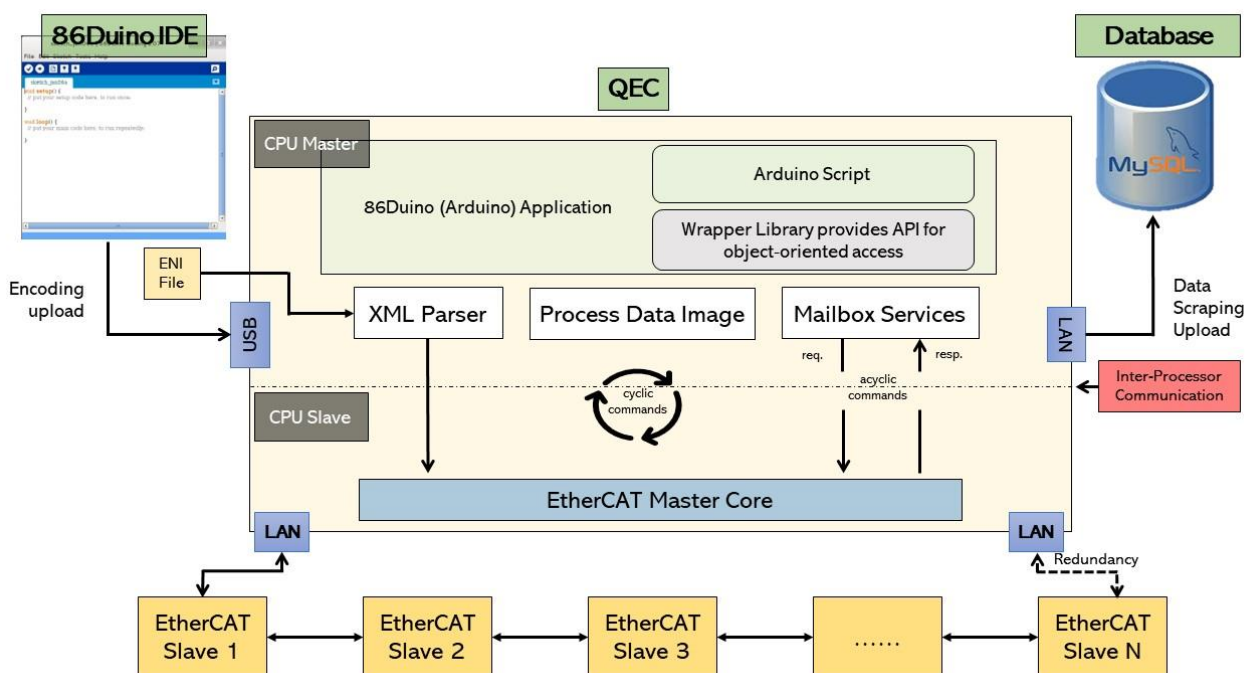
The QEC master is highly compatible with third-party EtherCAT devices for communication, such as servo, I/O, etc. For taking full advantage of EtherCAT, It supports PDO, CoE, FoE, DC, and EtherCAT cable redundancy to use other EtherCAT slaves flexibly. The QEC master has precise synchronization (min.125 μ s), and its 86Duino IDE provides less than 1 μ s jitter time in the minimum cycle time; it could apply to highly synchronized and precision automatic applications, like motion control and I/O control. (Read More: [EtherCAT Master's Benchmark – QEC](#))

QEC-M-01 has a built-in high endurance 2GB SLC eMMC, designed to provide a stable and reliable operating system. Users can upload the developed executable files and required images or data, such as HMI images, to the QEC-M-01's SLC via the 86Duino IDE without affecting the performance of the master system.

QEC-M-01 can also monitor hardware information on temperature, voltage, and current. These features allow users to track the system's carbon footprint and estimate its lifespan. QEC-M-01's dimension is 107.45 x 77.39 x 34 mm, could be mounted via Din-Rail. Operating temperature is from -20°C to +70°C; it can be placed directly in the field outdoors and ensures that the machine can work in harsh environments.

QEC-M-01 has two networks for EtherCAT Cable Redundancy, one Giga LAN for external network connection, RS485 signal pins, HD Audio, and USB; All provide an off-the-shelf API to use. Users can quickly collect data over EtherCAT and transfer data to a server via Giga LAN, and then big data can easily to construct by MySQL Library.

1.1.1 QEC-M Systems Diagram



1.1.2 Software Support

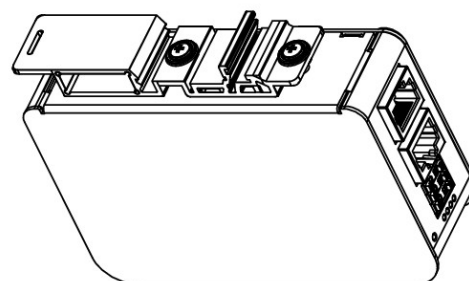
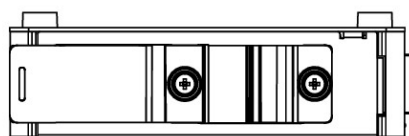
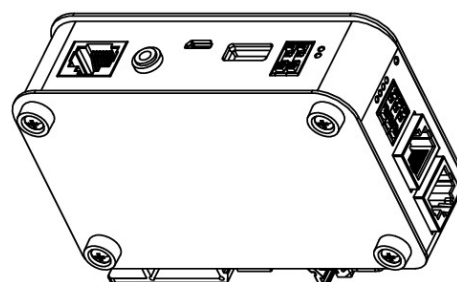
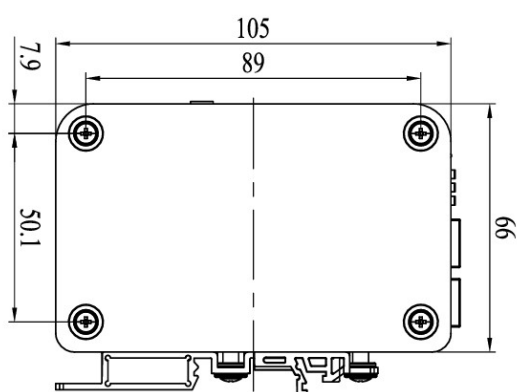
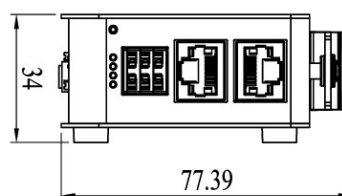
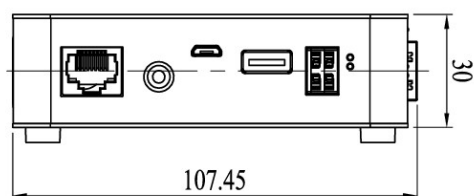
The 86Duino integrated development environment (IDE) software makes it easy to write code and upload it to QEC-M, and it runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Arduino IDE, Processing, DJGPP, and other open-source software. You can Download here: <https://www.qec.tw/software/>.



1.2 Specifications

| | | |
|-----------------------|---|---|
| CPU | DM&P Vortex86EX2 Processor, Master 533MHz/Slave 400MHz | |
| Memory | 512MB/1GB DDRIII Onboard | |
| Storage | 32MB SPI Flash/2GB SLC eMMC | |
| LAN | 1Gbps Ethernet RJ45 x1 10/100Mbps Ethernet RJ45 x2 for EtherCAT | |
| I/O Connector | Power DC Input/Output Connector x1 USB 2.0 Host x1 Micro USB (Type-B) x1 (Upload/Debug only) | Audio Connector x1 RS485 x1 RJ45 x3 |
| Protocol | EtherCAT (EtherCAT Master Functions: PDO, CoE, DC, Cable-redundancy, etc.) | |
| Ethernet Standard | IEEE 802.3 | |
| Control Cycle Time | 125 μ s (min.) | |
| Power Connector | 6-pin Power Input /Output | |
| Power Requirement | 4-pin Power Input/Output & 2-pin FGND | |
| Power Consumption | 5W | |
| Operating Temperature | -20 to +70°C/-40 to +85°C (Option) | |
| Dimension | 107.45 x 66 x 34mm (Without DIN-Rail) | |
| Weight | 270 g | |
| Mounting | DIN-Rail | |
| Certifications | CE, FCC, VCCI | |
| Internal Monitoring | Temperature, Voltage, Current | |
| Certifications | CE, FCC, VCCI | |
| Software Support | 86Duino Coding IDE 500+ (The environment is written in Java and based on Arduino IDE, Processing, DJGPP, and other open-source software) | |

1.3 Dimension

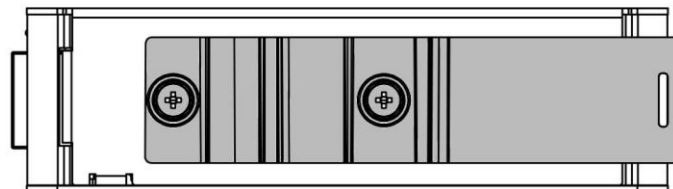
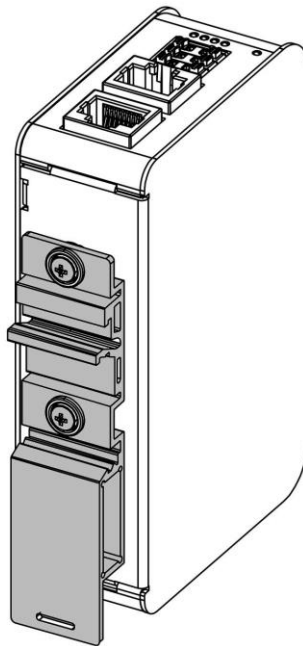


(Unit: mm)

1.4 Mounting Instruction

QEC-M-01 is an easy-install design to help you maintain your modules easily. Please refer to [Ch.3.1 DIN-Rail installation](#).

- **DIN-Rail**



1.5 Ordering Information

| Type | LCD size | (Below is the customization function, the unfilled fields do not need to be filled in; if the customer does not require, it will be directly shipped standard material number, such as QEC-M-01) | | | | | | | |
|-------|----------|--|---|---------|---------|---|------------|---|---------|
| | | PoE | - | Feature | | - | Wide Temp. | - | Coating |
| | | | | Memory | Storage | | | | |
| QEC-M | 01 | X | | X | X | | X | | X |

1. Type: Code 1~4

M: EtherCAT Master

2. LCD size: Code 5~6

01: without LCD

3. PoE: Code 7

P: RJ45 PoE Device, Red Plastic Housing

None or E: RJ45 w/o power, Black Plastic Housing

(Standard: None)

4. Feature: Code 8~9

X (Memory): G: 1G DDRIII / M: 512M DDRIII (Standard: 512MB)

X (Storage): 1, 2, 4: eMMC size (Standard: 2G)

5. Wide Temp.: Code 10

X: -40 to +85°C / R: -20 to +70°C (Standard: -20 to +70°C)

6. Coating: Code 11

C: Yes / N: Normal

QEC-M-01 X - XX - X - X

1.5.1 Ordering Part Number:

- **QEC-M-01**: Vortex86EX2 Processor 533MHz-based EtherCAT Master System
- **QEC-M-01P**: Vortex86EX2 Processor 533MHz-based EtherCAT Master System/PoE
- **QEC-M-01P-G4-X-C**: Vortex86EX2 Processor 533MHz-based EtherCAT Master System/PoE/1G DDRIII Memory/4G eMMC Storage/Wide Temp./Coating

Ch. 2

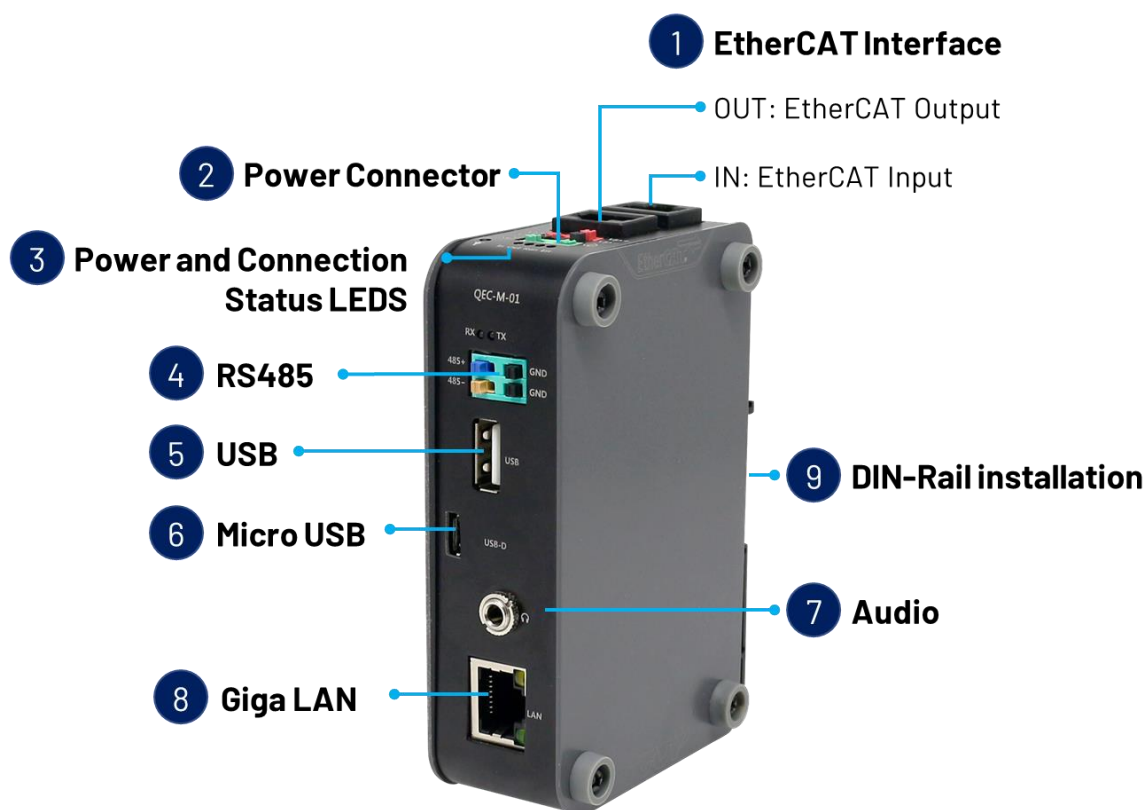
Hardware System

[2.1 General Technical Data](#)

[2.2 Connector Summary](#)

[2.3 Wiring to the Connector](#)

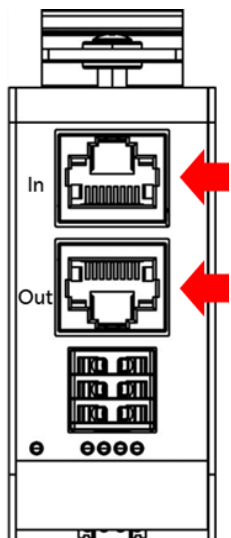
2.1 General Technical Data



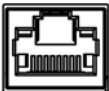
2.2 General Summary

| No. | Description | | Type Narrative | Pin # |
|-----|----------------------------------|-----|--|-------|
| 1 | EtherCAT Interface | IN | External RJ45 Connector (Gold finger) | 8-pin |
| | | OUT | | 8-pin |
| 2 | Power Connector | | Terminal Block Interface | 6-pin |
| 3 | Power and Connection Status LEDs | | External Status LEDs | - |
| 4 | RS-485 | | Terminal Block Interface | 4-pin |
| 5 | USB | | Standard USB 2.0 | - |
| 6 | Micro USB | | Micro USB (Type-B) | - |
| 7 | Audio | | HD Audio | - |
| 8 | Giga LAN | | External RJ45 Connector (Gold finger) | 8-pin |
| 9 | DIN-Rail | | - | - |

2.2.1 EtherCAT Interface



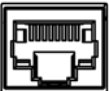
EC IN

|  8 2,1 | 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.

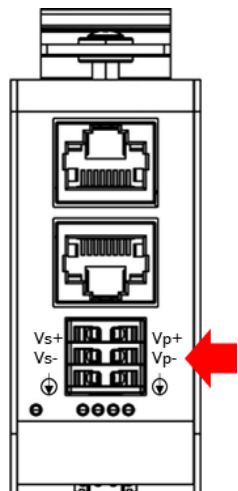
EC OUT

|  1,2 8 | Pin # | Signal Name | Pin # | Signal Name |
|--|-------|-------------|-------|-------------|
| | 1 | LAN2_TX+ | 2 | LAN2_TX- |
| | 3 | LAN2_RX+ | 4 | VS+ |
| | 5 | VP+ | 6 | LAN2_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.

2.2.2 Power Connector

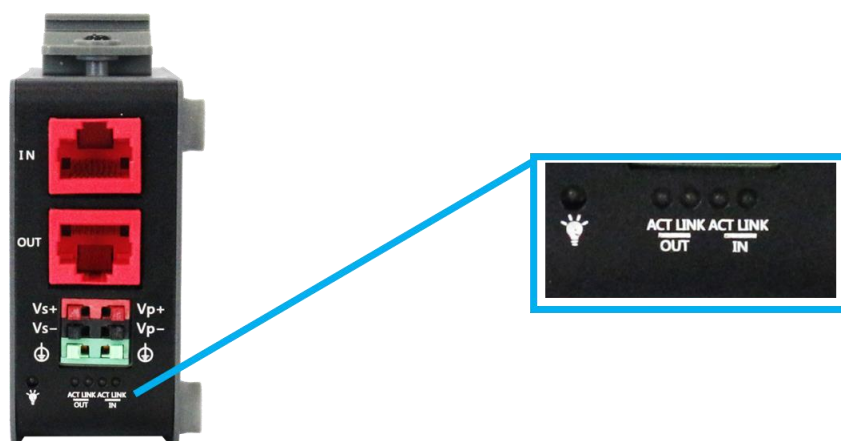


Vs for system power; Vp for peripheral power and backup power.

|  | Pin # | Signal Name | Pin # | Signal Name |
|--|-------|-------------|-------|-------------|
| | 1 | Vs+ | 2 | Vp+ |
| | 3 | Vs- (GND) | 4 | Vp- (GND) |
| | 5 | F.G | 6 | F.G |


* Power Input voltage +19 to +50VDC Power Input (Typ. +24VDC)

2.2.3 Power and Connection Status LEDs



Power Status LED

Power input is 24V (typical), 12V or 48V. The LED status provide high/low voltage warning.

| Notation | Color | States | Description |
|---|-------------|--------------|--|
| PWR  | Green / Red | Green LED On | 1. Voltage $\leq 48V$ and Voltage $\geq 19V$ |
| | | Green LED On | 1. Voltage $< 50V$ and Voltage $> 48V$ |
| | | Red LED On | 2. Voltage $< 19V$ and Voltage $< 17V$ |
| | | Red LED On | Voltage $\geq 50V$ and Voltage $\leq 17V$ |

* Vs power status will be displayed first.

Connection Status LEDs


There are two LED for each LAN ports. Please refer to the table below for the LAN port LED indications.

| LAN | Notation | Color | States | Description |
|-----|----------|--------|--------------------|--------------------|
| OUT | ACT | Green | Green LED Blinking | Data Activity |
| | LINK | Orange | Orange LED On | 100Mbps Connection |
| IN | ACT | Green | Green LED Blinking | Data Activity |
| | LINK | Orange | Orange LED On | 100Mbps Connection |

2.2.4 RS-485

A set of RS485+ and RS485- pins and 2 pins GND.



|  | Pin # | Signal Name | Pin # | Signal Name |
|---|-------|-------------|-------|-------------|
| | 2 | RS485+ | 1 | GND |
| | 4 | RS485- | 3 | GND |

To setup RS-485 pins, please refer to [Serial](#).

2.2.5 USB

Standard USB 2.0 with Hot-plug.

You can plug in the Keyboard, Mouse, or USB disk to control the QEC-M-01.



For drive USB, you can refer to the following hyperlinks:

- [SD library](#): read USB disk.
- [Keyboard Controller Example](#)
- [Mouse Controller Example](#)

2.2.6 Micro USB

The Micro USB is mainly for programming upload.



For getting started of QEC-M-01 with its software, please see [Ch.4 Getting Started](#).

2.2.7 Audio




HD Audio (Line-Out).

For use HD Audio function, please refer to [Audio Library](#).

2.2.8 Giga LAN



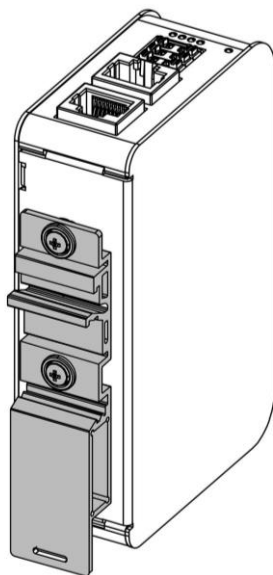
There are three LAN ports in QEC-M-01, two for EtherCAT communication and one for external Ethernet work.

|  | Pin # | Signal Name | Pin # | Signal Name |
|---|-------|-------------|-------|-------------|
| | L1 | GTX+ | L2 | GTX- |
| | L3 | GRX+ | L4 | GTXC+ |
| | L5 | GTXC- | L6 | GRX- |
| | L7 | GRXD+ | L8 | GRXD- |

The EtherCAT Lan on the QEC-M divides into Input and Output for cable redundancy. To drive GigaLAN, you can refer to [Ethernet library](#) or [Modbus Library](#).

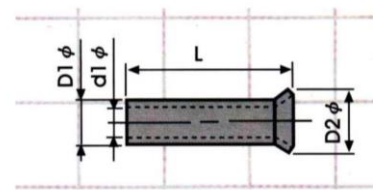
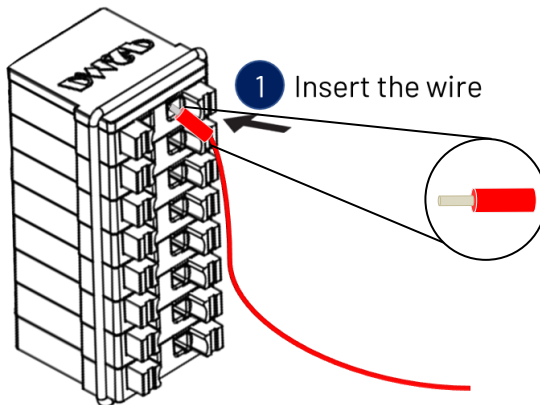
2.2.9 DIN-Rail installation

Please refer to [Ch.3.1 DIN-Rail installation](#).



2.3 Wiring to the Connector

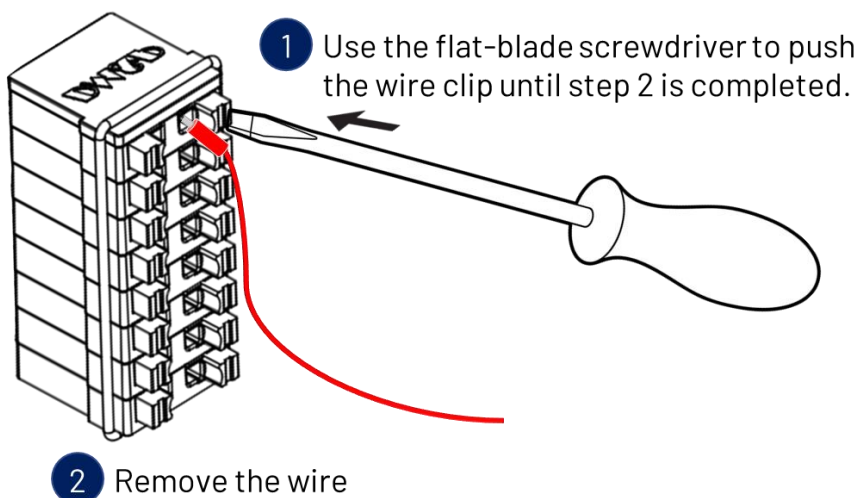
2.3.1 Connecting the wire to the connector



Insulated Terminals Dimensions (mm)

| Position | L | ØD1 | Ød1 | ØD2 |
|-----------|------|-----|-----|-----|
| CN 0.5-6 | 6.0 | 1.3 | 1.0 | 1.9 |
| CN 0.5-8 | 8.0 | 1.3 | 1.0 | 1.9 |
| CN 0.5-10 | 10.0 | 1.3 | 1.0 | 1.9 |

2.3.2 Removing the wire from the connector





Ch. 3

Hardware Installation

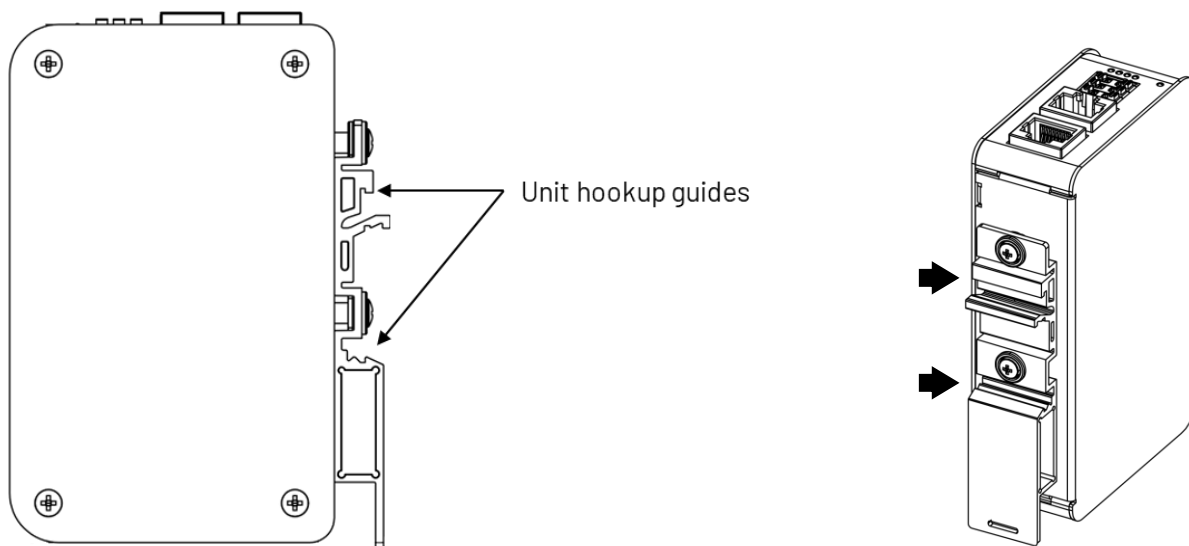
[3.1 DIN-Rail installation](#)

[3.2 Removing QEC-M-01 Unit](#)

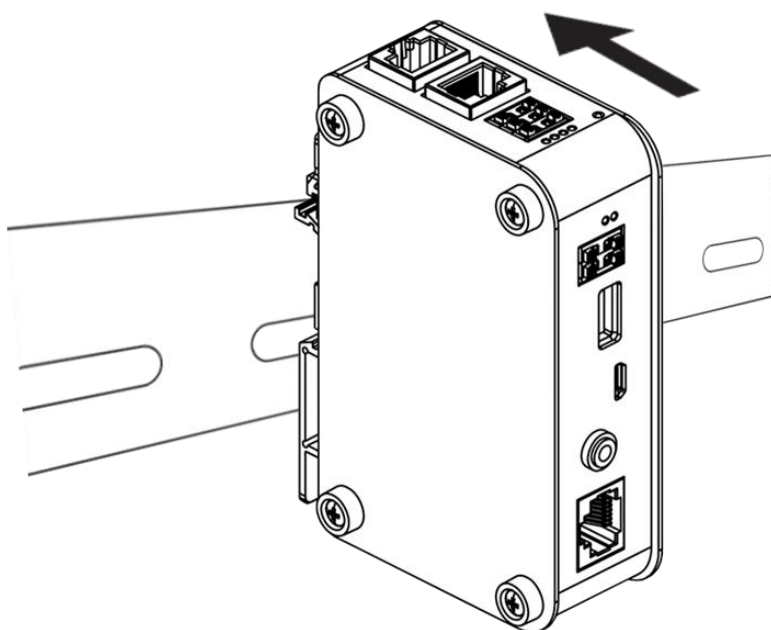
This section describes how to install QEC-M-01. Please turn OFF the power supply before you mount QEC-M-01. Always mount QEC-M-01 one at a time.

3.1 DIN-Rail installation

Slide in the QEC-M-01 on the hookup guides and press the QEC-M-01 with a certain amount of force against the DIN track until the DIN track mounting hook lock into place.



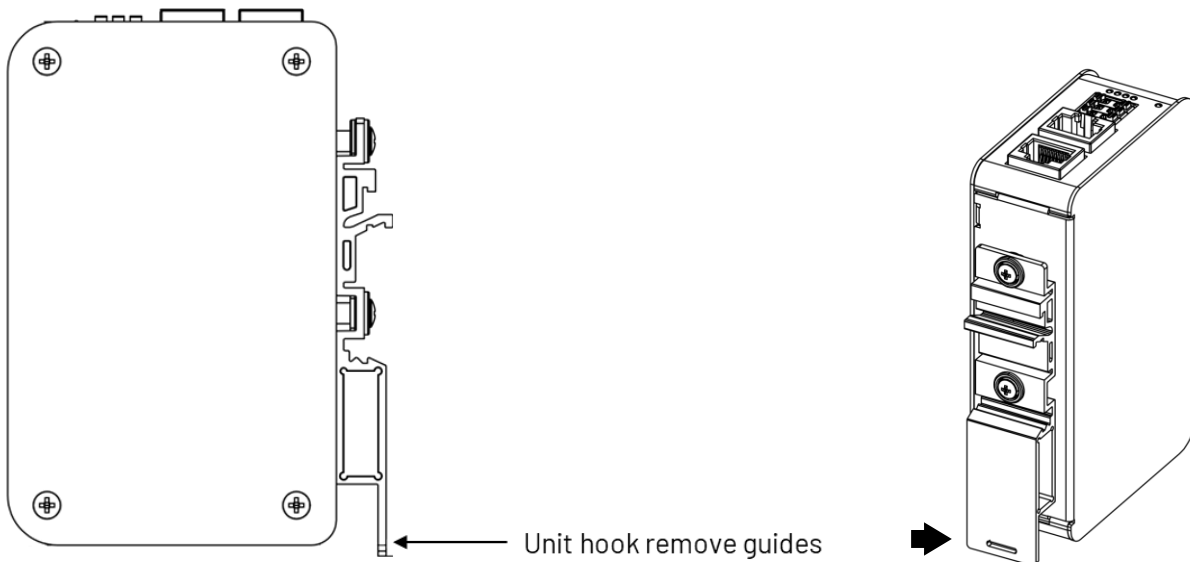
When you mount the QEC-M-01, releasing the DIN track mounting hook on the QEC-M-01 is unnecessary. After you mount the QEC-M-01, make sure it is locked to the DIN track.



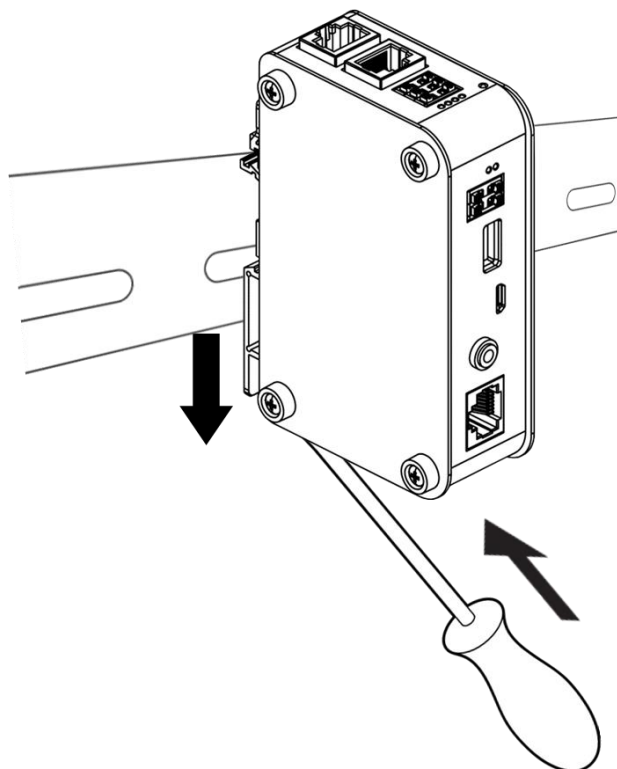
Note: Always turn OFF the Unit power supply before connecting and removing the QEC-M-01.

3.2 Removing QEC-M-01 Unit

Use a flat-blade screwdriver to remove the DIN Track mounting hook on the unit.



Pull down the flat-blade screwdriver against the DIN track until the mounting hook being removed from the track.





Ch. 4

Getting Started

[4.1 Package Contents](#)

[4.2 Hardware Configuration](#)

[4.3 Software/Development Environment](#)

[4.4 Connect to your PC and set up the environment](#)

[4.5 EtherCAT Development Method](#)

[4.6 Troubleshooting](#)

This chapter explains how to start with QEC-M-01 and its software, 86Duino Coding IDE.

Note. 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



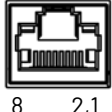
PoE type

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. 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).



2. 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.

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

4.1 Package Contents

The package includes the following items:

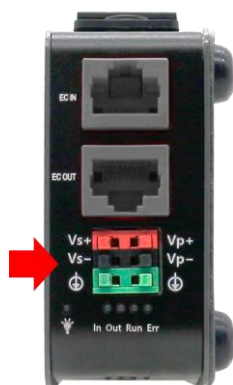
- QEC-M-01
- Cable-set

Please get in touch with our sales channels if any of the package items are missing or damaged. Also, feel free to reuse the shipping materials and carton for further storing and shipping needs in the future,

4.2 Hardware Configuration

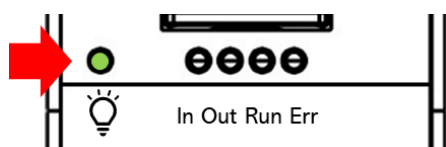
The development environment will be pre-installed before the QEC-M is shipped to customers. Users must download the software (see [4.3 Software/Development Environment](#)) and follow this user manual to set up the device.

4.2.1 Plug in the power supply



There are two groups of power supplies in QEC-M-01, Vs and Vp. The voltage requirement for both supplies' ranges from 19V to 50V wide voltage.

After powering it on, the power LED lights up (green).



4.3 Software/Development Environment

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

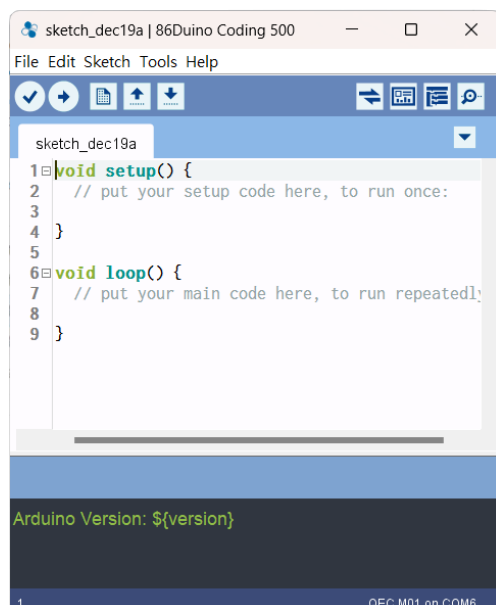


After downloading, please unzip the downloaded zip file, no additional software installation is required, just double-click **86duino.exe** to start the IDE.



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

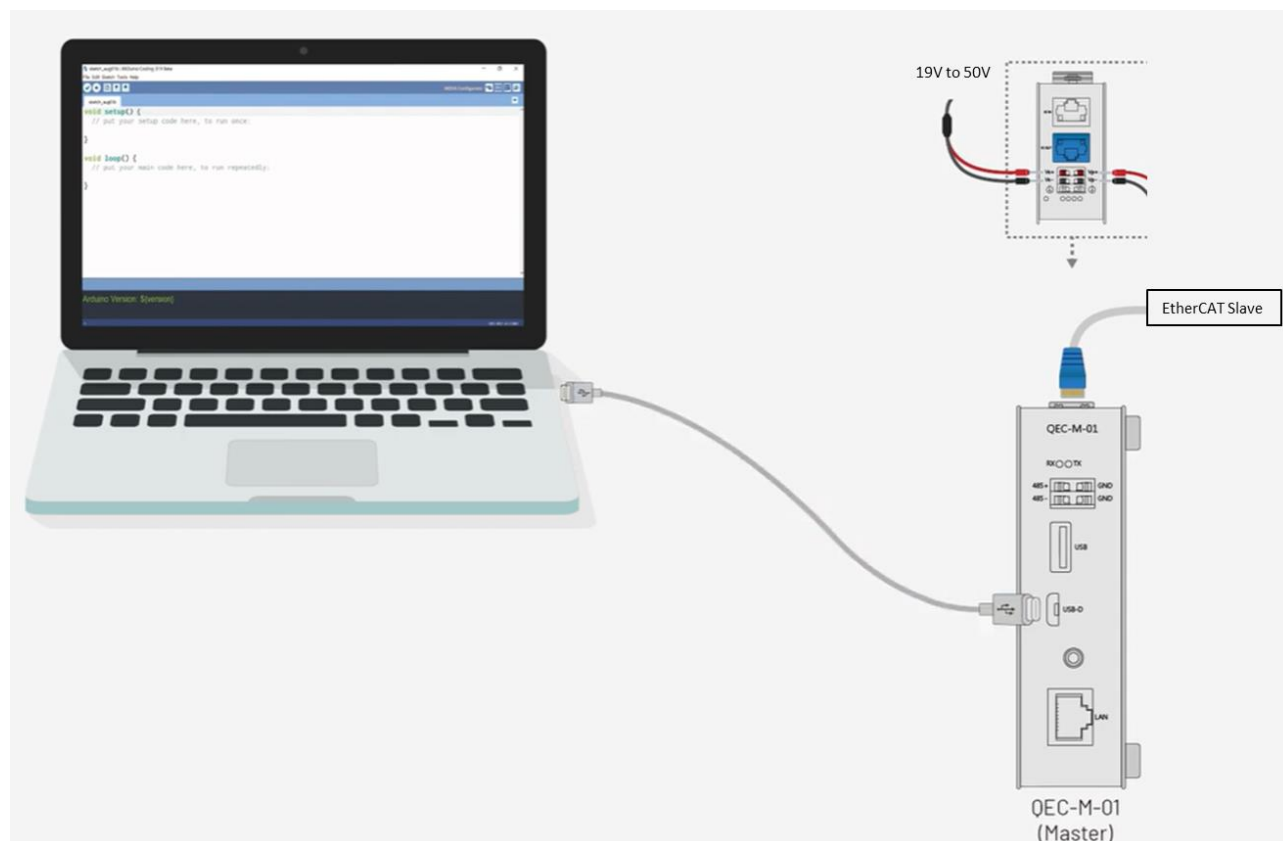
86duino Coding IDE 500+ looks like below.



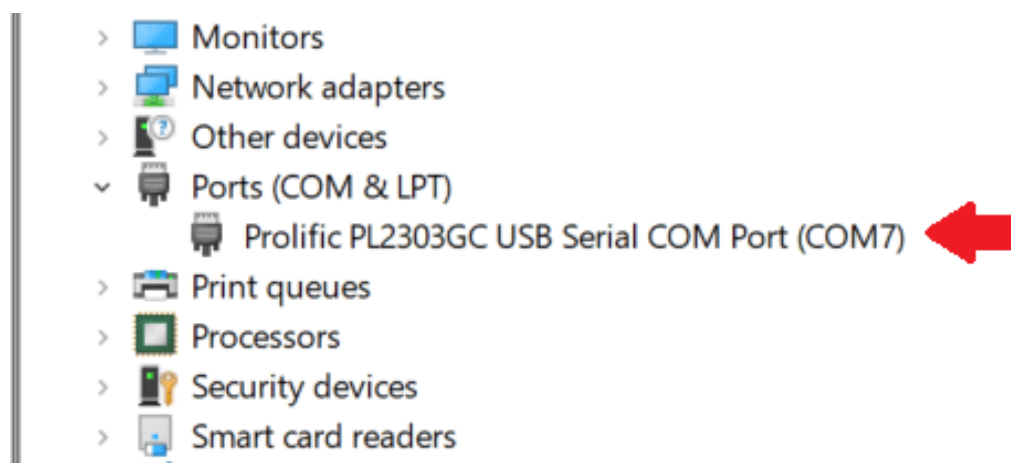
4.4 Connect to your PC and set up the environment

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).

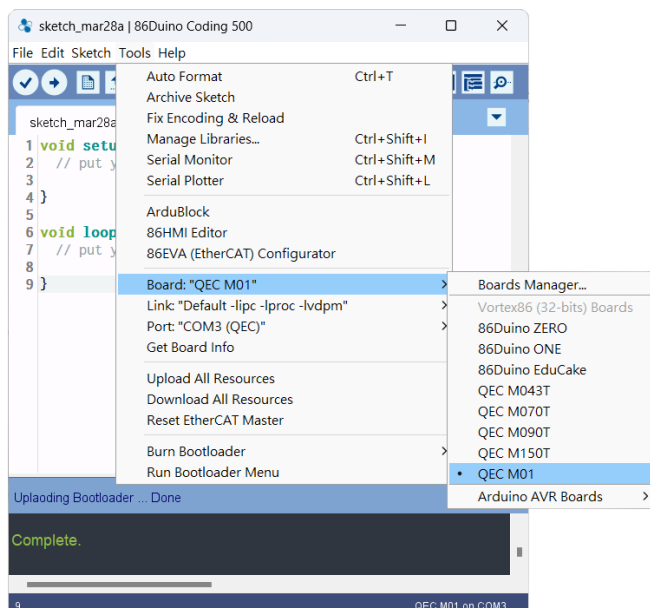


2. Turn on the QEC power.
3. Open "Device Manager" -> "Ports (COM & LPT)" in your PC and expand the ports; you should see that the "Prolific PL2303GC USB Serial COM Port (COMx)" is detected; if not, you will need to install the required drivers.

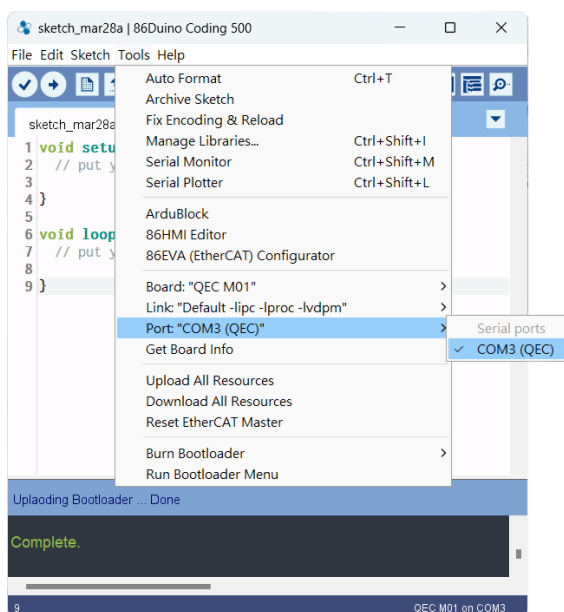


For Windows PL2303 driver, you can download [here](#).

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, COM3 (QEC)).



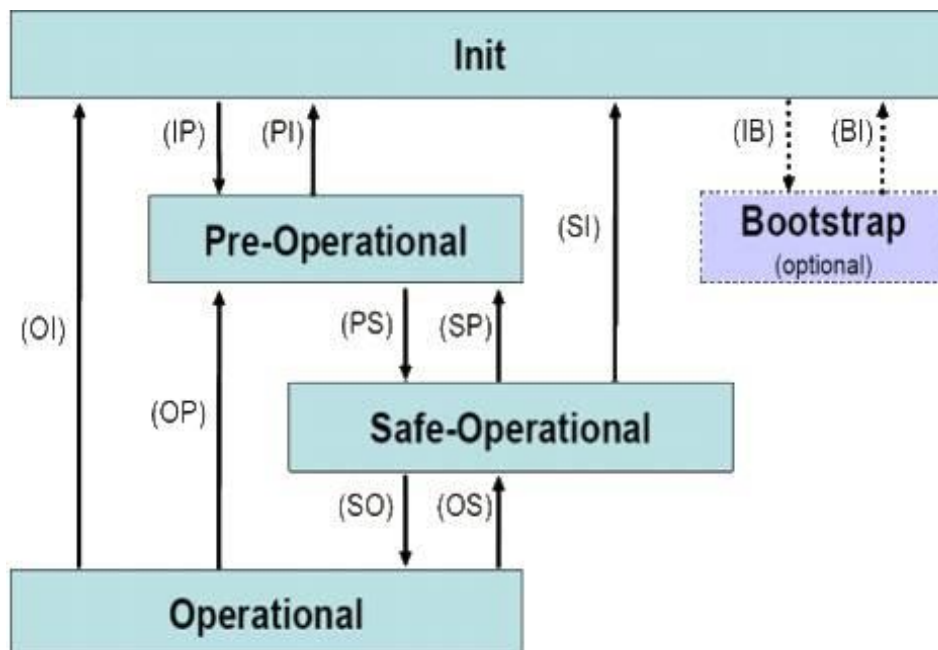
4.5 EtherCAT Development Method

This section introduces two primary methods to configure your EtherCAT Slave devices through the QEC EtherCAT Master: Write code and Use 86EVA with code. Both methods are designed to offer flexibility and efficiency depending on your familiarity and requirements.

4.5.1 Implementing the EtherCAT State Machine

To set up and transition EtherCAT Slave devices through various operational states using the QEC EtherCAT Master. It is crucial for understanding the state transitions and operational flow within an EtherCAT network.

The state of the EtherCAT slave is controlled via the EtherCAT State Machine (ESM). Depending upon the state, different functions are accessible or executable in the EtherCAT slave. Specific commands must be sent by the EtherCAT master to the device in each state, particularly during the bootup of the slave.



A distinction is made between the following states:

- Init
- Pre-Operational
- Safe-Operational and
- Operational
- Boot

The regular state of each EtherCAT slave after bootup is the OP state.

The EtherCAT master (QEC-M-01P) can be configured in the EtherCAT network via the EtherCAT library and programmed with the control action in the 86Duino IDE. The 86Duino 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.

```

1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly
8
9 }

```

Free-run Mode Code Example:

```

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

EthercatMaster master; // Create an EtherCAT Master Object
EthercatDevice_Generic slave; // Create an Generic EtherCAT Slave Object

void setup() {
  // EtherCAT Master Initialize. All slaves will enter PRE-OP state if success.
  master.begin();

  // Specify the EC-Slave number and mount it on the EC-Master.
  slave.attach(0, master);

  // Start EtherCAT Master. All slaves will enter OP state if success.
  // FREERUN Mode, and parameter 1000000 sets the cycle time in nanoseconds
  master.start(1000000, ECAT_FREERUN_AUTO);
}

void loop() {

}

```

SYNC Mode Code Example:

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



EthercatMaster master; // Create an EtherCAT Master Object
EthercatDevice_Generic slave; // Create an Generic EtherCAT Slave Object

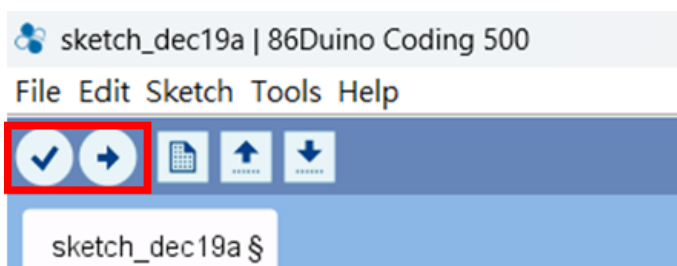
void setup() {
  // EtherCAT Master Initialize. All slaves will enter PRE-OP state if success.
  master.begin();

  // Specify the EC-Slave number and mount it on the EC-Master.
  slave.attach(0, master);

  // Start EtherCAT Master. All slaves will enter OP state if success.
  // Sync Mode, and the parameter 10000000 sets the cycle time in nanoseconds
  master.start(10000000, ECAT_SYNC);
}

void loop() {
}
```

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, and the LED will start flashing.



4.5.2 Process Data Objects (PDO) Functions

The process data transferred by an EtherCAT slave during each cycle (Process Data Objects, PDOs) are user data which the application expects to be updated cyclically or which are sent to the slave.

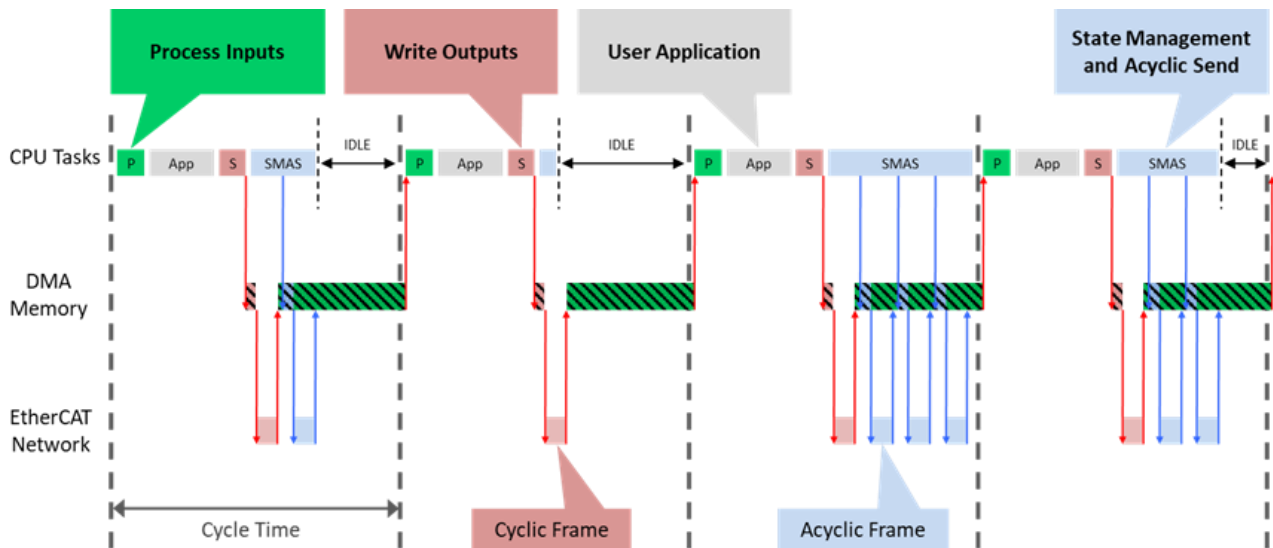
In the 86Duino IDE, the EtherCAT network cyclic part is highly integrated and consisted of certain functions with specific tasks that will not interact with others due to their strict division. Based on this feature, we can measure the time consumption of these master frames and further ensure the accuracy of the cycle process. The tasks include:

| | |
|-------------|---|
| P | Process Inputs Receive and process all cyclic frames. |
| S | Write Outputs Send all cyclic frames. |
| SMAS | State Management and Acyclic Send State machine management and process queued acyclic commands. |
| App | User Application User's application task. To do some calculation by input process data, and create values of output process data. |

The received data (inputs) in the previous EtherCAT frame is first updated in the "Process Inputs" task. The "User Application" task takes its newly received data, calculates the data (outputs), and sends it out to the network by calling the "Write Outputs" task.

The EtherCAT frame can pass the network through Direct Memory Access (DMA). This means the data can be transported without loading the CPU, as well as be received when returning to the master without interruption automatically or sent out when using a physical network.

In the “State Management and Acyclic Send (SMAS)” task, the master state machine and each slave state machine are executed. All slave device states are transferred from INIT to OPERATIONAL during the initial start-up process. The acyclic mailbox communications require another frame with slave-specific commands for reading and writing to the slave. Moreover, to handle these acyclic mailbox communications like CoE, the state machines are needed in regular operation.



EtherCAT Common Slave Driver, read/write the Process Data Input/Output.

PDO Function Code Example:

When using a non-QEC slave, you can use the `EthercatDevice_Generic` function, which can be referenced [EthercatDevice Class](#). We expect the EtherCAT Slave device is Digital IO module.

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

EthercatMaster master; // Create an EtherCAT Master Object
EthercatDevice_Generic slave; // Create an EtherCAT Slave Object



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

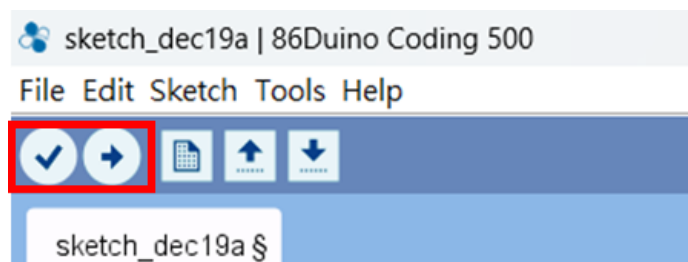
    // Attach the QEC-R11D88H slave device to the EtherCAT Master at position 0
    slave.attach(0, master);

    // Start EtherCAT Master. All slaves will enter OP state if success.
    // Sync Mode, and the parameter 1000000 sets the cycle time in nanoseconds
    master.start(1000000, ECAT_SYNC);
}

void loop() {
    // Write a HIGH value to a bit in the Process Data Output at index 0
    slave.pdoBitWrite(0, HIGH);
    delay(4000); // Wait for 4 seconds

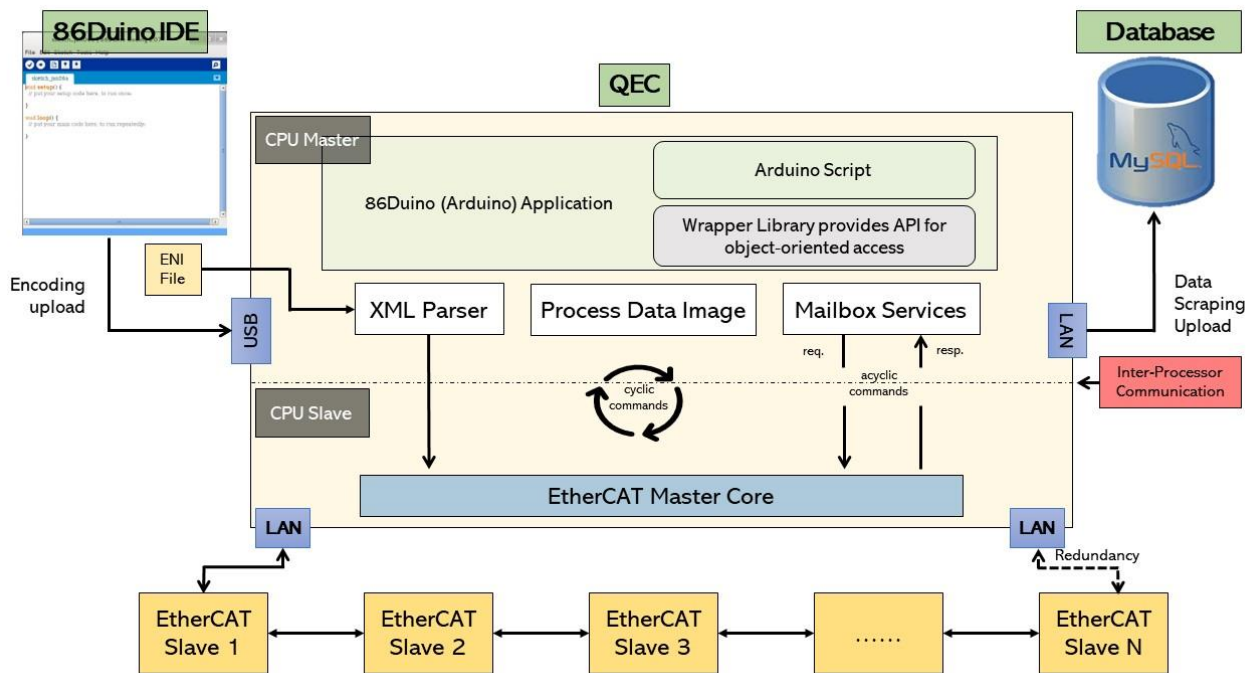
    // Write a LOW value to the same bit in the Process Data Output at index 0
    slave.pdoBitWrite(0, LOW);
    delay(1000); // Wait for 1 second
}
```

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.



4.5.3 Cyclic Callback

Cyclic Callbacks play a crucial role in QEC EtherCAT Master systems, for handling data exchange in the Vortex86EX2 processor. It allows the system to periodically execute specific tasks at regular intervals, ensuring timely data handling and synchronization across the network. This is particularly important in industrial automation where precision and real-time response are key.



Understanding Cyclic Callbacks:

Cyclic Callbacks specify a named Interrupt Service Routine (ISR) to be called when an interrupt occurs. This mechanism is vital for automating tasks within your program, such as reading sensor data or managing user inputs, without the need for continuous polling.

Key Notes on Interrupts:

- Interrupts should be addressed promptly to maintain system responsiveness.
- Avoid using `delay()` in the ISR as it can lead to data loss or system lag.
- Variables modified within an ISR should be declared as **volatile** to ensure correct updates.

About Interrupt Service Routines (ISRs):

- ISRs are unique: They cannot have parameters and do not return values.
- An ISR should be concise and efficient to avoid system bottlenecks.
- In a multi-interrupt system, only one ISR executes at a time, making it crucial to optimize each ISR for speed and efficiency.

EtherCAT Specifics:

- In an EtherCAT context, Cyclic Callbacks are essential for managing PDO (Process Data Object) updates, ensuring that the system captures all critical data points without missing a pulse.
- Unlike in some systems, `delay()` and `millis()` in the 86Duino environment function as expected in ISRs, but it's still best practice to avoid long delays in ISR code.

QEC attach the Cyclic Callback Function Code Example

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

EthercatMaster EcatMaster; // Create an EtherCAT Master Object
EthercatDevice_Generic Slave1; // Create an Generic EtherCAT Slave Object



void myCallback() {
    // put your cyclic Callback function here.
}

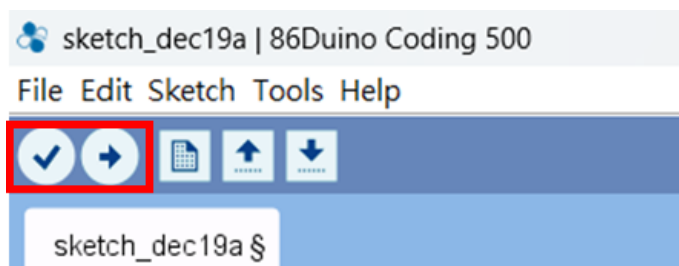
void setup() {
    // EtherCAT Master Initialize. All slaves will enter PRE-OP state if success.
    EcatMaster.begin();

    // Specify the EC-Slave number and mount it on the EC-Master.
    Slave1.attach(0, EcatMaster);

    // Set a cyclic callback for the Ethercat Master
    EcatMaster.attachCyclicCallback(myCallback);
}
```

```
// Start EtherCAT Master. All slaves will enter OP state if success.  
// Sync Mode, and the parameter 1000000 sets the cycle time in nanoseconds  
EcatMaster.start(1000000, ECAT_SYNC);  
}  
  
void loop() {  
  
}
```

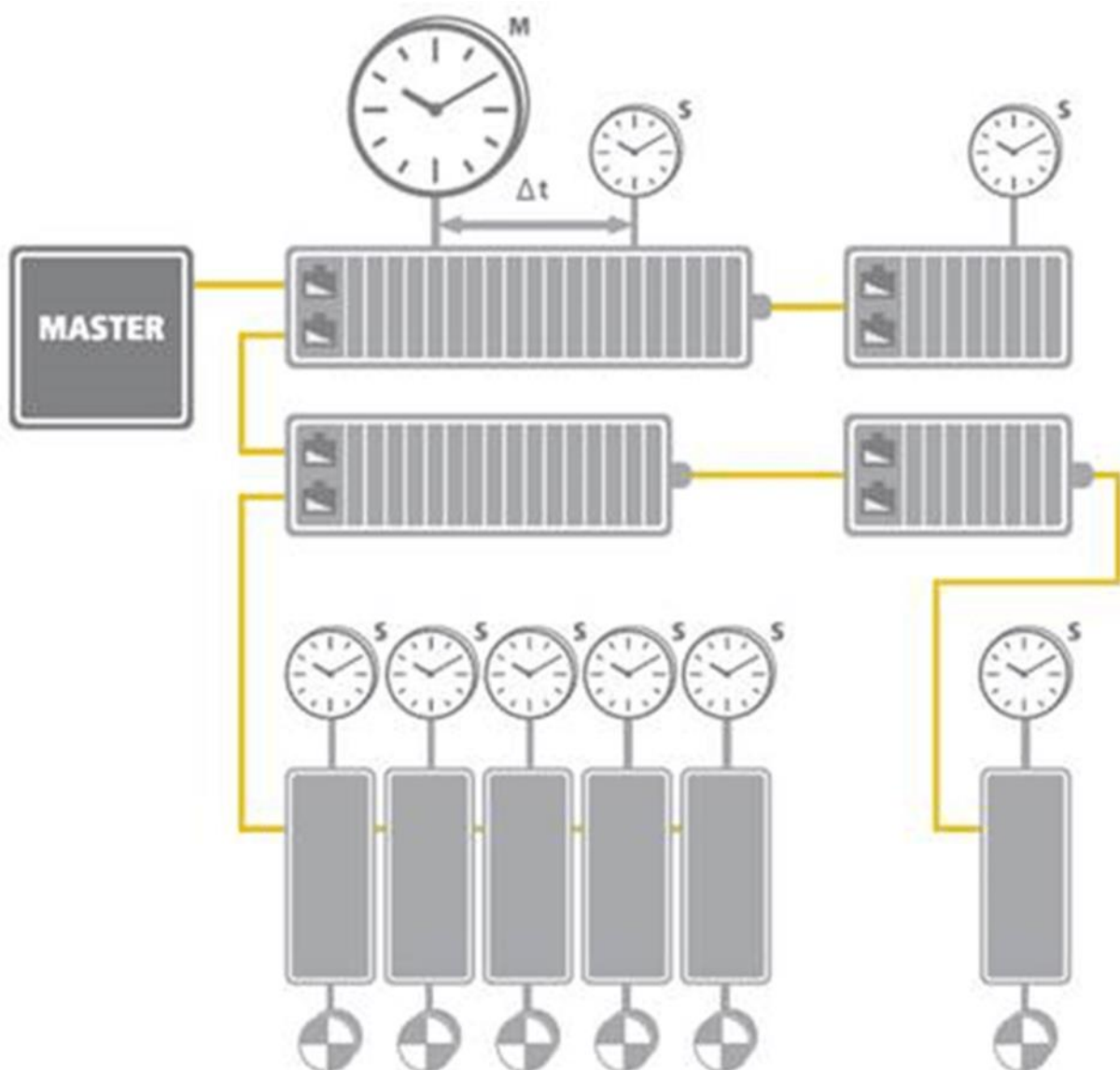
Note: Once the code is written, click on the toolbar to  compile, and to confirm that the compilation is complete and error-free, you can click  to upload. The program will run when the upload is complete, and the LED will start flashing.



4.5.4 Distributed Clock (DC)

In applications with spatially distributed processes requiring simultaneous actions, exact synchronization is particularly important. For example, this is the case for applications in which multiple servo axes execute coordinated movements.

In contrast to completely synchronous communication, whose quality suffers immediately from communication errors, distributed synchronized clocks have a high degree of tolerance for jitter in the communication system. Therefore, the EtherCAT solution for synchronizing nodes is based on such distributed clocks (DC).

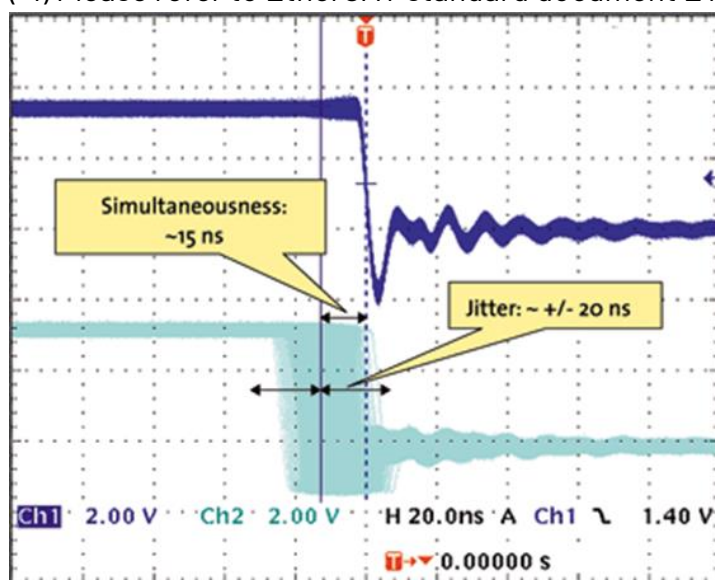


EtherCAT: Illustration of Distributed Clock (DC). (Source of information: <http://www.ethercat.org/>)

The synchronization mechanism of EtherCAT is based on IEEE-1588 Precision Clock Synchronization Protocol and extends the definition to so-called Distributed-clock (DC). To put it simple, every EtherCAT ESC maintains a hardware-based clock and the minimum time interval is 1 nano-second (64 bits in total). The time maintained by EC-Slave is called Local system time.

With accurate internet time synchronization mechanism and dynamic time compensation mechanism (*1), EtherCAT DC technology can guarantee that the time difference among every EC-Slave local system time is within ± 20 nano-seconds. The following diagram is a scope view of two slave devices' output digital signals. We can see that the time difference between the I/O signal from two EC-Slaves is around 20 nano-seconds.

(*1) Please refer to EtherCAT standard document ETG1000.4



Synchronicity and Simultaneousness: Scope view of two distributed devices with 300 nodes and 120 m of cable between them. (Source of information: <http://www.ethercat.org/>)

QEC Set DC Code Example

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

EthercatMaster EcatMaster; // Create an EtherCAT Master Object
EthercatDevice_Generic Slave1; // Create a Generic EtherCAT Slave Object
EthercatDevice_Generic Slave2; // Create a Generic EtherCAT Slave Object

void myCallback() {
    // put your cyclic Callback function here.
}

void setup() {
    // EtherCAT Master Initialize. All slaves will enter PRE-OP state if success.
    EcatMaster.begin();



    // Specify the EC-Slave number and mount it on the EC-Master.
    Slave1.attach(0, EcatMaster);
    Slave1.setDc(1000000); // 1000000 ns = 1 ms

    // Specify the EC-Slave number and mount it on the EC-Master.
    Slave2.attach(1, EcatMaster);
    Slave2.setDc(1000000); // 1000000 ns = 1 ms

    // Set a cyclic callback for the Ethercat Master
    EcatMaster.attachCyclicCallback(myCallback);

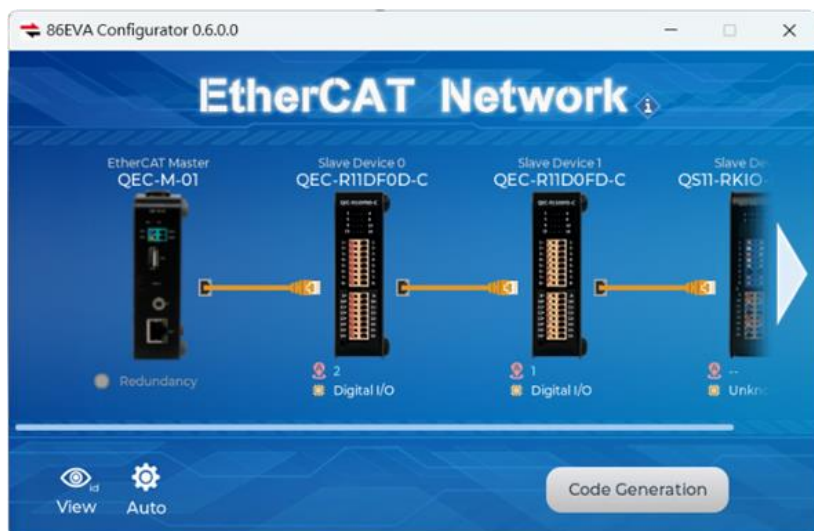
    // Start EtherCAT Master. All slaves will enter OP state if success.
    // Sync Mode, and the parameter 1000000 sets the cycle time in nanoseconds
    EcatMaster.start(1000000, ECAT_SYNC);
}

void loop() {
}
```

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, and the LED will start flashing.

4.5.5 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.



The following information about 86EVA will focus on QEC EtherCAT Slave devices, with features including:

1. Automatically generated Arduino language (via EtherCAT-Based Virtual Arduino)
2. Automatically scan for network devices.
3. EtherCAT Master/Primary Settings:
 - Set Master Object Name
 - Set Cycle Time
 - Set Redundancy Options
 - Optional ENI file
4. EtherCAT Slave/Secondary Settings:
 - Set Secondary Object Name
 - Set Slave Alias
 - Slave I/O Mapping can be set
 - Display secondary device information
 - View internal information, including:
 - a. Voltage (V)
 - b. Current (A)
 - c. Temperature (C)
 - d. Startup time (hr)

Example

This example is using a QEC-M-01P to control the QEC-R11D88H-N (EtherCAT Slave with 8-ch Digital Input and 8-ch Digital Output). The following devices are used here:

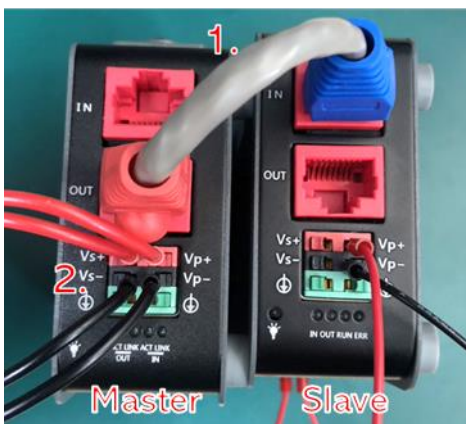
1. QEC-M-01P (EtherCAT Master/PoE)
2. QEC-R11D88H-N (EtherCAT Slave 8-ch digital input and 8-ch digital output/PoE)
3. 24V power supply
4. 24V LED



QEC-M-01P

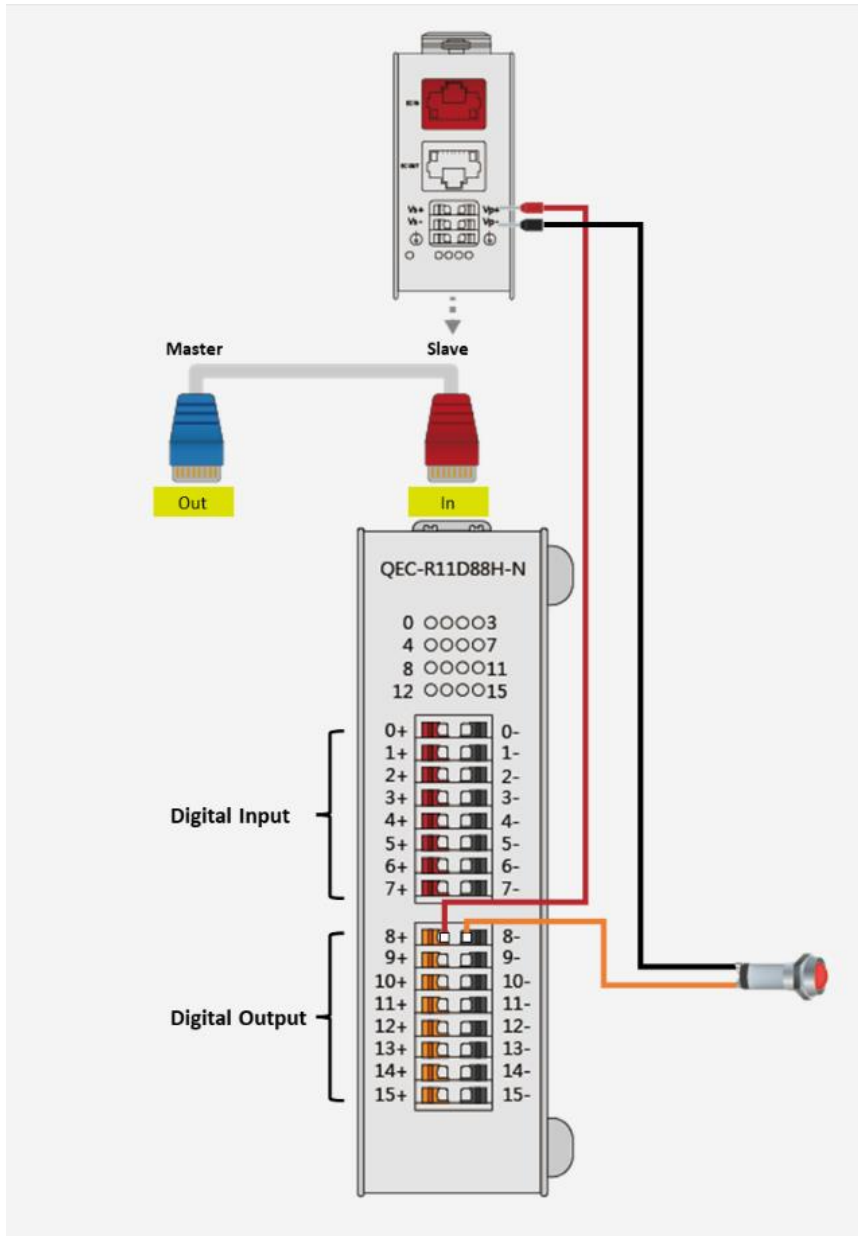
QEC EtherCAT master with PoE function.

1. Using the EtherCAT Out port (top side) connected to the EtherCAT In port of QEC-R11D88H via RJ45 cable (powered by PoE).
2. Connect to Vs+/Vs- and Vp+/Vp- power supplies via EU terminals for 24V power.



QEC-R11D88H-N

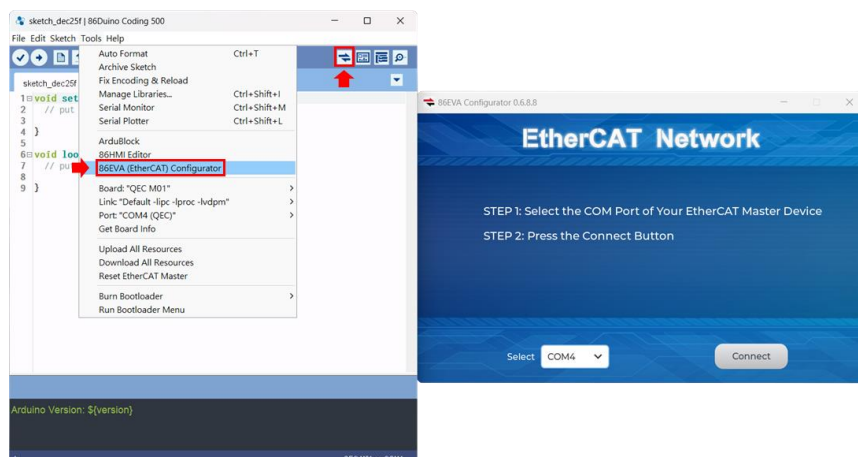
- Connect from VP+ to DO 0+.
- Connect the 24V LED+ terminal to DO 0-.
- Connect the 24V LED- end to the VP-.



The following example is setting Pin0 of Digital Output to HIGH for 4 seconds and then changing it to LOW for 1 second.

Step 1: Turn on 86EVA and scan

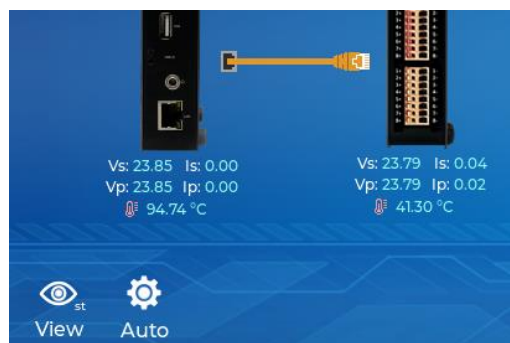
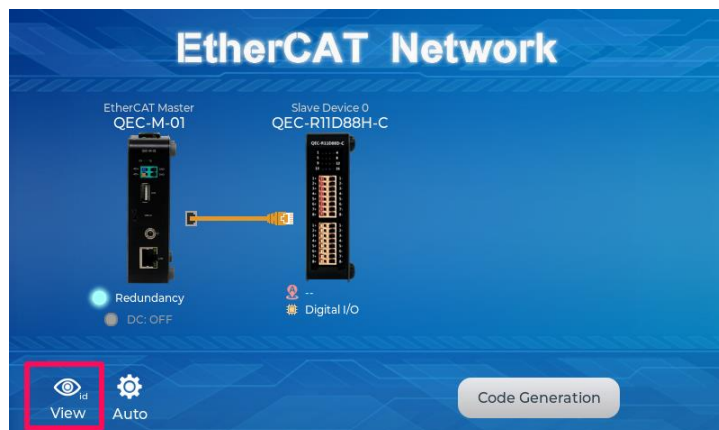
The 86EVA tool can be opened via the following buttons.



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.



Step 2: Set the parameters

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

QEC-M-01:

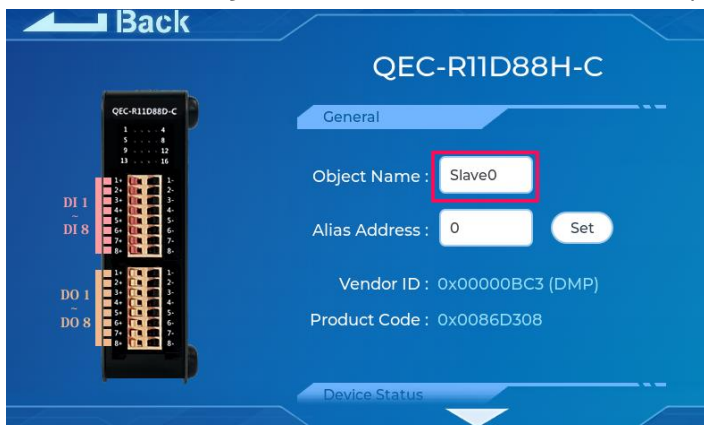
Click on the image of the QEC-M-01 to see the parameter settings.

If you are developing for the first time, please use the preset settings first and click "**Back**" in the upper left corner to return.

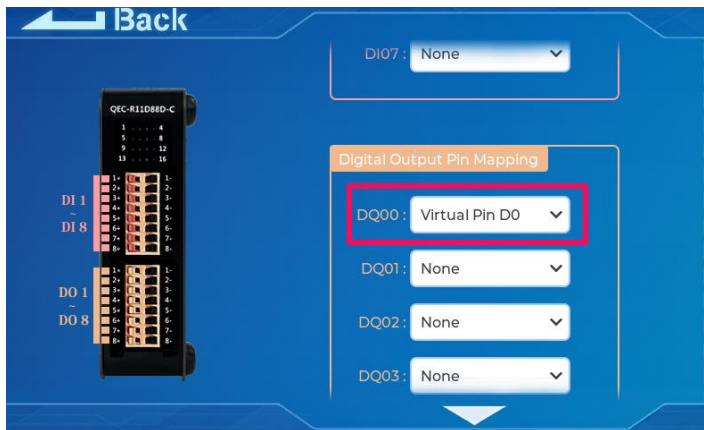


QEC-R11D88H-N:

Click on the image of the QEC-R11D88H to see the parameter settings.



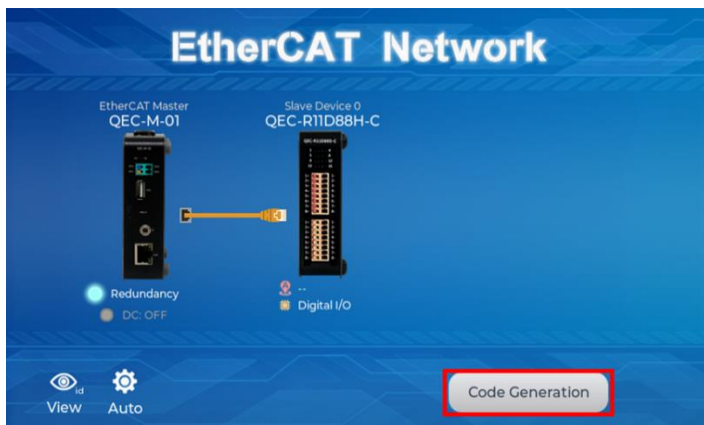
Continue down to the **"Digital Output Pin Mapping"** area. Among them, we select **"Virtual Pin D0"** in the drop-down box of DQ00 of Digital Output Pin Mapping and click **"Back"** in the upper left corner to return.



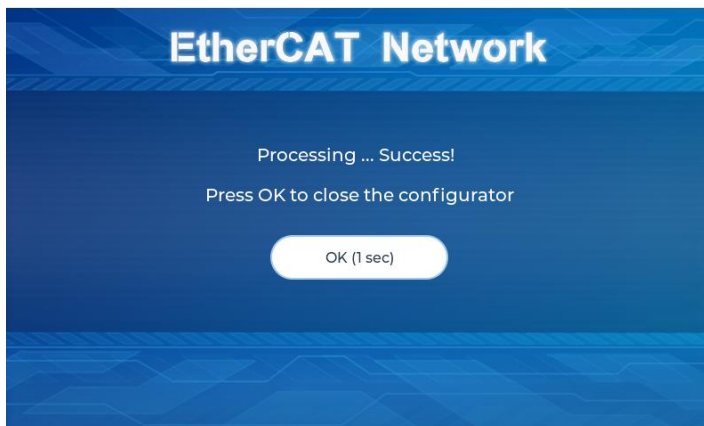
This action is to set the Digital Output Pin0 of QEC-R11D88H to the virtual D0 pin of EVA.

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.

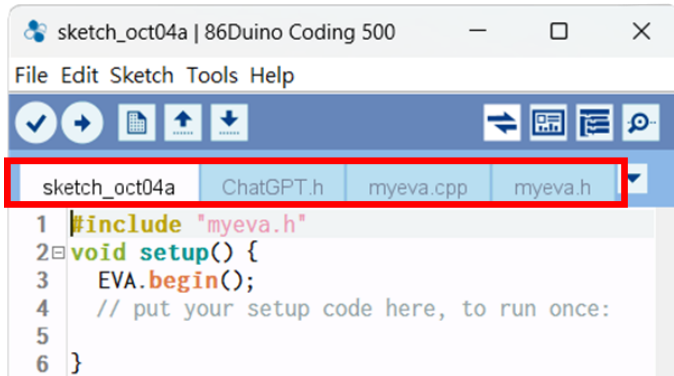


When you're done, double-click the OK button to turn off 86EVA, or it will close in 10 seconds.



The generated code and files are as follows:

- sketch_oct04a: 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



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



The following example is setting the D0 pin of EVA (Pin0 for Digital Output) to HIGH for 4 seconds and then changing it to LOW for 1 second.

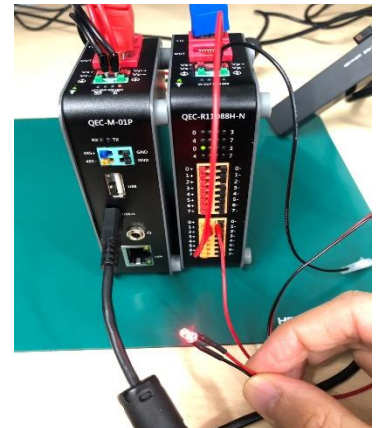
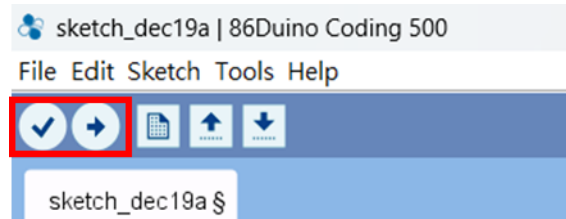
```
#include "myeva.h" // Include EVA function

void setup() {
  EVA.begin(); // Initialize EVA function.
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
  // Set EVA D0 pin to HIGH
  EVA.digitalWrite(0, HIGH);
  delay(4000); // Wait for 4 seconds

  // Set EVA D0 pin to LOW
  EVA.digitalWrite(0, LOW);
  delay(1000); // Wait for 1 second
}
```

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, and the LED will start flashing.



4.6 Troubleshooting

4.6.1 Old environment of your QEC-M-01:

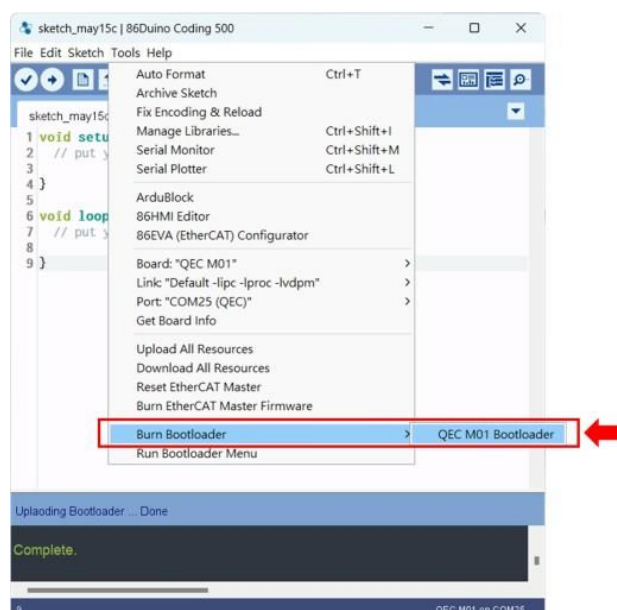
If you cannot upload your sketch successfully, please try to update your QEC EtherCAT Master. This update covers the following three updates: Bootloader, EtherCAT Firmware, and EtherCAT Tools. Now, we will further explain how to proceed with the update:

Step 1: Setting up QEC-M

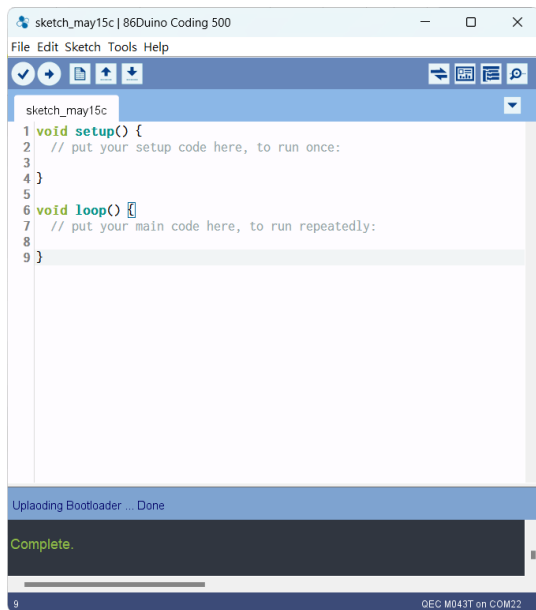
1. Download and install 86Duino IDE 500 (or a newer version): You can download it from [Software](#).
2. Connect the QEC-M: Use a USB cable to connect the QEC-M to your computer.
3. Open 86Duino IDE: After the installation is complete, open the 86Duino IDE software.
4. Select Board: From the IDE menu,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



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.



Ch. 5

Software Function

[5.1 Software Description](#)

[5.2 EtherCAT Function List](#)

[5.3 Additional Resources](#)

5.1 Software Description

The 86Duino Coding IDE 500+ developed by the QEC team designed specifically for industrial-field control systems, bringing simple and powerful functions into related industrial fields through the open-source Arduino.

Please visit [gec.tw](https://www.gec.tw) for 86Duino Coding IDE 500+ details.



You can Download here: <https://www.gec.tw/software/>.

5.2 EtherCAT Function List

The list below introduces the functions of our QEC series products.

Please visit [EtherCAT Master API User Manual](#) for API Function details.

5.2.1 EthercatMaster Class Functions

Initialization Functions:

- `begin()` - EtherCAT Master Initialize.
- `end()` - Shutdown EtherCAT Master.
- `isRedundancy()` - Check EtherCAT uses redundancy or not.
- `libraryVersion()` - The version of EtherCAT Master library.
- `firmwareVersion()` - The version of EtherCAT firmware.

Access to slave information Functions:

Get Slave Information (SlaveCount, VendorID, ProductCode, RevisionNumber, SerialNumber, AliasAddress) on EtherCAT bus.

- `getSlaveCount()` - Get EtherCAT Slave Count Number on EtherCAT bus.
- `getVendorID()` - Get the EtherCAT Slave Vendor ID on EtherCAT bus.
- `getProductCode()` - Get the EtherCAT Slave Product Code on EtherCAT bus.
- `getRevisionNumber()` - Get the EtherCAT Slave revision Number on EtherCAT bus.
- `getSerialNumber()` - Get the EtherCAT Slave Serial Number on EtherCAT bus.
- `getAliasAddress()` - Get the EtherCAT Slave Alias Address on EtherCAT bus.
- `getSlaveNo()` - Obtain the sequential ID of the specified slave based on the alias address/supplier number/product number/amendment number/serial number.

Control Functions:

- `start()` - Start EtherCAT Master. All slaves will enter OP state if successful.
There are three modes: Auto Freerun / Manually Freerun / Sync.
- `stop()` - Stop EtherCAT Master.
- `getSystemTime()` - Get system time of current cycle. (The unit is nanosecond)
- `getWorkingCounter()` - Get working counter of current cycle.
- `getExpectedWorkingCounter()` - Get expected working counter. (Fixed value)

Access to master information Functions:

Get Master information, including System/Peripheral power voltage or current.

- `getSystemPowerVoltage()` - Get the System Power Voltage Value of Master. (unit: V)
- `getSystemPowerCurrent()` - Get the System Power Current Value of Master. (unit: A)
- `getPeripheralPowerVoltage()` - Get the Peripheral Power Voltage Value of Master. (unit: V)
- `getPeripheralPowerCurrent()` - Get the Peripheral Power Current Value of Master. (unit: A)

Advanced EtherCAT Master Functions:

The following APIs are still under development and are not recommended for use.

- `attachCyclicCallback()` - Register Cyclic Callback Function.
- `detachCyclicCallback()` - Unregister Cyclic Callback Function.
- `attachErrorCallback()` - Register Error Callback Function.
- `detachErrorCallback()` - Unregister Error Callback Function.

5.2.2 EthercatDevice Class General Functions

Initialization Functions:

- `attach()` - Specify the EC-Slave number and mount it on the EC-Master.
- `detach()` - Uninstall the slave object.

Access to slave information Functions:

- `getVendorID()` - Get EtherCAT Slave Count Number on EtherCAT bus.
- `getProductCode()` - Get the EtherCAT Slave Product Code on EtherCAT bus.
- `getRevisionNumber()` - Get the EtherCAT Slave revision Number on EtherCAT bus.
- `getSerialNumber()` - Get the EtherCAT Slave Serial Number on EtherCAT bus.
- `getAliasAddress()` - Get the EtherCAT Slave Alias Address on EtherCAT bus.
- `getSlaveNo()` - Get EtherCAT Slave Number on EtherCAT bus.
- `readSII()` - Read EEPROM Info.
- `readSII8()` - Read EEPROM Info. (uint8_t)
- `readSII16()` - Read EEPROM Info. (uint16_t)
- `readSII32()` - Read EEPROM Info. (uint32_t)
- `writeSII()` - Write EEPROM.
- `writeSII8()` - Write EEPROM. (uint8_t)
- `writeSII16()` - Write EEPROM. (uint16_t)
- `writeSII32()` - Write EEPROM. (uint32_t)

Process Data Objects (PDO) Functions:

- `pdoBitWrite()` - Write Bit of Process Data Output.
- `pdoBitRead()` - Read Bit of Process Data Input.
- `pdoGetOutputBuffer()` - Get Slave Process Data Output Pointer.
- `pdoGetInputBuffer()` - Get Slave Process Data Input Pointer.
- `pdoWrite()` - Write Slave Process Data Output.
- `pdoWrite8()` - Write Slave Process Data Output. (uint8_t)
- `pdoWrite16()` - Write Slave Process Data Output. (uint16_t)
- `pdoWrite32()` - Write Slave Process Data Output. (uint32_t)
- `pdoRead()` - Read Slave Process Data Input.
- `pdoRead8()` - Read Slave Process Data Input. (uint8_t)
- `pdoRead16()` - Read Slave Process Data Input. (uint16_t)
- `pdoRead32()` - Read Slave Process Data Input. (uint32_t)

CANopen over EtherCAT (CoE) Functions:

- `sdoDownload()` - (CoE) Write the object to EtherCAT Slave device.
- `sdoDownload8()` - (CoE) Write the object to EtherCAT Slave device. (unit8_t)
- `sdoDownload16()` - (CoE) Write the object to EtherCAT Slave device. (unit16_t)
- `sdoDownload32()` - (CoE) Write the object to EtherCAT Slave device. (unit32_t)
- `sdoUpload()` - (CoE) Read the object from EtherCAT Slave device to EtherCAT Master.
- `sdoUpload8()` - (CoE) Read the object from EtherCAT Slave device to EtherCAT Master. (unit8_t)
- `sdoUpload16()` - (CoE) Read the object from EtherCAT Slave device to EtherCAT Master. (unit16_t)
- `sdoUpload32()` - (CoE) Read the object from EtherCAT Slave device to EtherCAT Master. (unit32_t)
- `getODlist()` - (CoE) Get the object list from EtherCAT Slave device to EtherCAT Master.
- `getObjectDescription()` - (CoE) Get the object list's Description from EtherCAT Slave device to EtherCAT Master.
- `getEntryDescription()` - (CoE) Get the entry description with Slave index.

File over EtherCAT (FoE) Functions:

- `readFoE()` - Read FoE.
- `writeFoE()` - Write FoE.

Distributed Clock (DC) Functions:

- `setDc()` - Configure Distributed Clock (DC) parameters.

Others Functions:

- `getState()` - Get EtherCAT State.
- `setState()` - Set EtherCAT State.

5.3 Additional Resources

If you want to learn more about the libraries available in the 86Duino IDE or explore the details of the 86Duino programming language, please visit the following links:

- **86Duino IDE Libraries:** Find an extensive list of libraries supported by 86Duino IDE, along with detailed documentation and examples at <https://www.qec.tw/86duino/libraries/>.
- **86Duino Language Reference:** Learn about the 86Duino programming language, including its syntax, functions, and usage in the official 86Duino Language Reference at <https://www.qec.tw/86duino/86duino-language-reference/>.

These resources provide valuable information for both beginners and experienced developers using the 86Duino platform. By exploring these links, you can harness the full potential of 86Duino IDE and create innovative projects.

Happy coding with 86Duino IDE!

The text of the 86Duino reference is a modification of [the Arduino reference](#) and is licensed under a [Creative Commons Attribution-ShareAlike 3.0 License](#). Code samples in the reference are released into the public domain.

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. 2024