スタートガイド

QEC-R11HU9S：EtherCAT ゲートウェイ RS232/485 + キーパッド + LCM + MPG（86EVA 編）

86Duino Coding IDE 501

EtherCAT ライブラリ

(Version 2.0)

改訂履歴

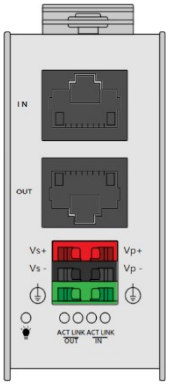
日付	バージョン	内容
2024/2/2	Version1.0	初版リリース。
2024/10/8	Version1.1	開発手順を 2 つの文書に分割。
2025/11/5	Version2.0	HU9S の全機能を 1 つの文書に統合。

はじめに

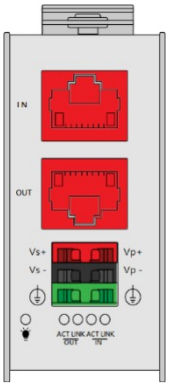
本ガイドでは、EtherCAT M デバイス **QEC-M-01** と **QEC-R11HU9S** シリーズ（RS232/485、キーパッド、LCM、MPG ハンドホイールを備えた EtherCAT Sub デバイス）の使用方法を説明します。

QEC の PoE（Power over Ethernet）に関する注意事項

QEC 製品では、PoE タイプと非 PoE タイプを簡単に見分けることができます。RJ45 コネクタのハウジングが赤色の場合は PoE タイプ、黒色の場合は非 PoE タイプです。



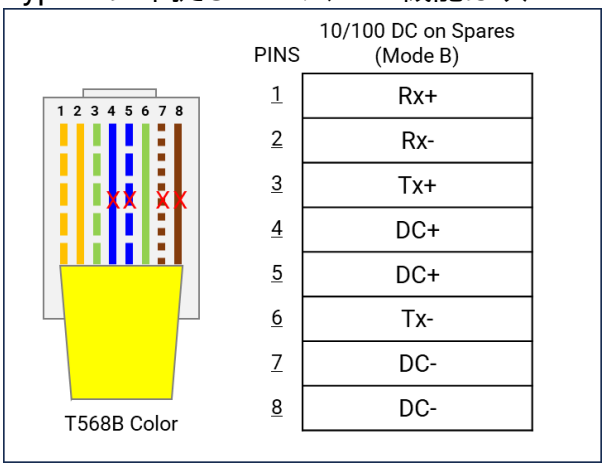
非 PoE タイプ



PoE タイプ

PoE（Power over Ethernet）は、ネットワークケーブル経由で電力を供給する機能です。QEC では配線を削減するためにオプションで PoE 機能を搭載できます。実際の運用では、PoE の採用はシステム構成機器に応じて判断されるため、評価・テストの際は次の点に注意してください。

1. QEC の PoE 機能は EtherCAT P とは異なり、互換性はありません。QEC の PoE 機能は PoE Type B に準拠しており、ピン機能は次のとおりです。



2. PoE 対応機器と非 PoE 機器を接続する場合は、必ずイーサネットケーブルの 4、5、7、8 番ピンを切断してください（例：PoE 対応の QEC EtherCAT M デバイスを他社製の EtherCAT Sub デバイスと接続する場合）。

3. QEC の PoE 給電能力は最大 24V/3A です。

1. ハードウェアの接続と配線

ここでは次の機器を使用します。

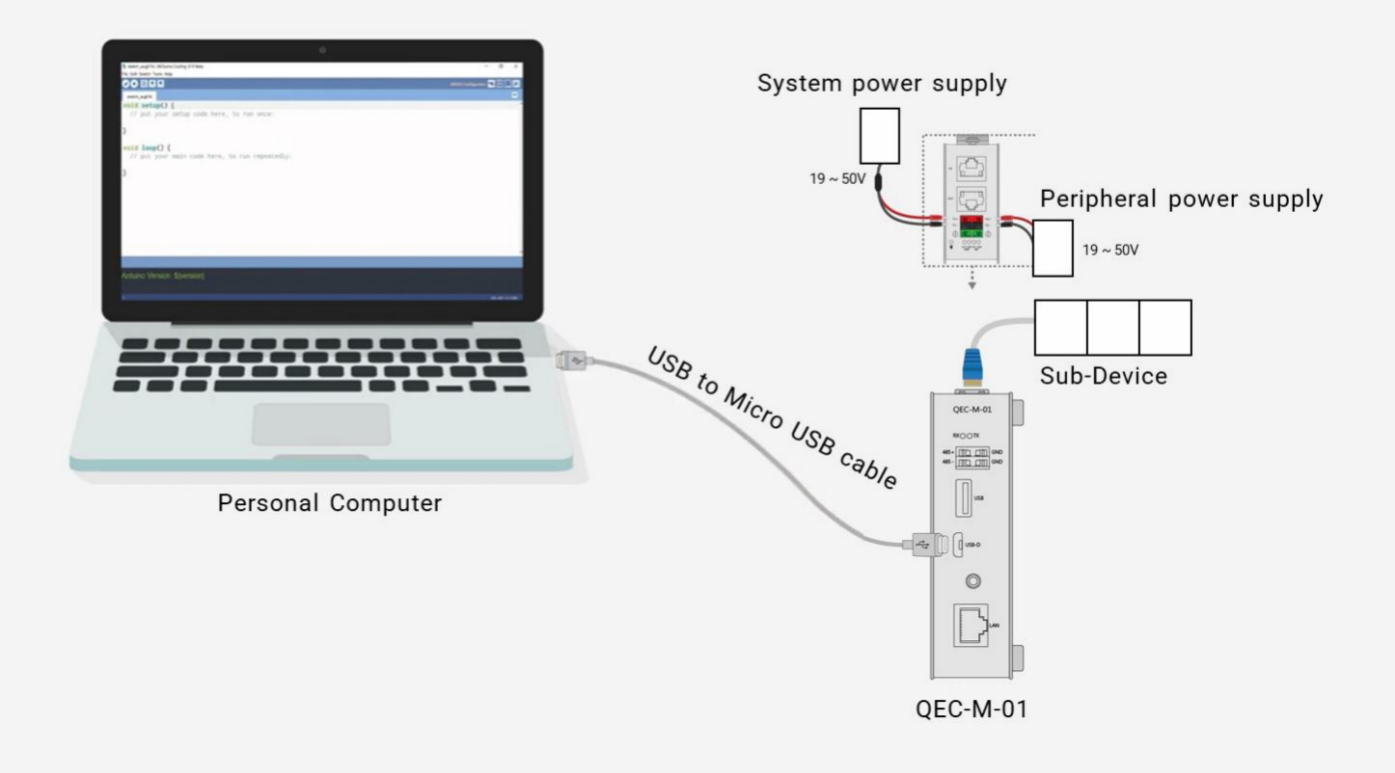
1. QEC-M-01（EtherCAT M デバイス）
2. QEC-R11HU9S シリーズ（RS232/485、キーパッド、LCM、MPG ハンドホイールを備えた EtherCAT Sub デバイス）
3. 24VDC 電源、EU タイプ端子ケーブル、LAN ケーブル



1.1 QEC-M-01

QEC-M-01 は EtherCAT M デバイスです。

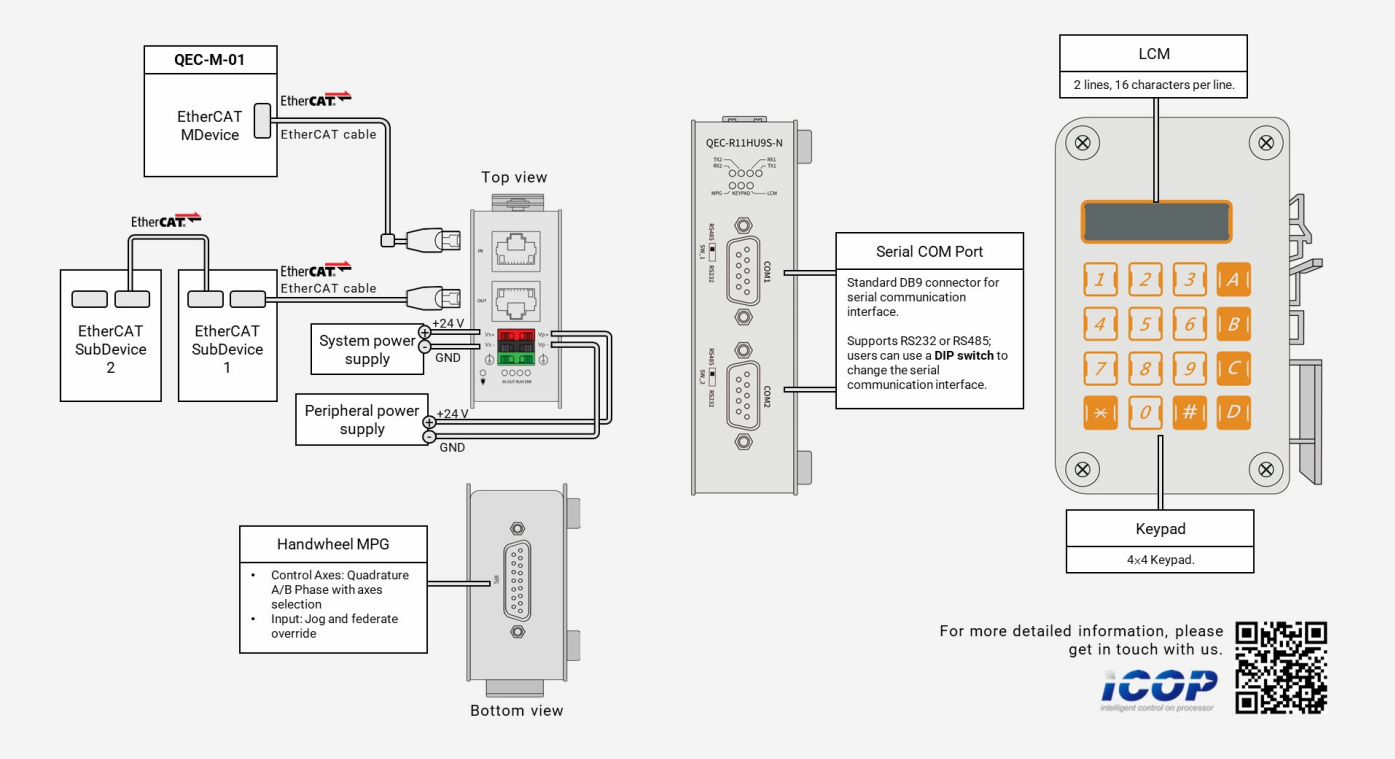
1. 電源：EU 端子を介して Vs+/Vs- および Vp+/Vp- に 24V 電源を接続します。
2. EtherCAT 接続：EtherCAT Out ポート（上側）から RJ45 ケーブルで EtherCAT Sub デバイスの EtherCAT In ポートに接続します。



1.2 QEC-R11HU9S

QEC-R11HU9S は、RS232/485、キーパッド、LCM、MPG ハンドホイールを備えた EtherCAT Sub デバイスモジュールです。

下図は、QEC M デバイスおよび EtherCAT ネットワークとの代表的な配線例です。

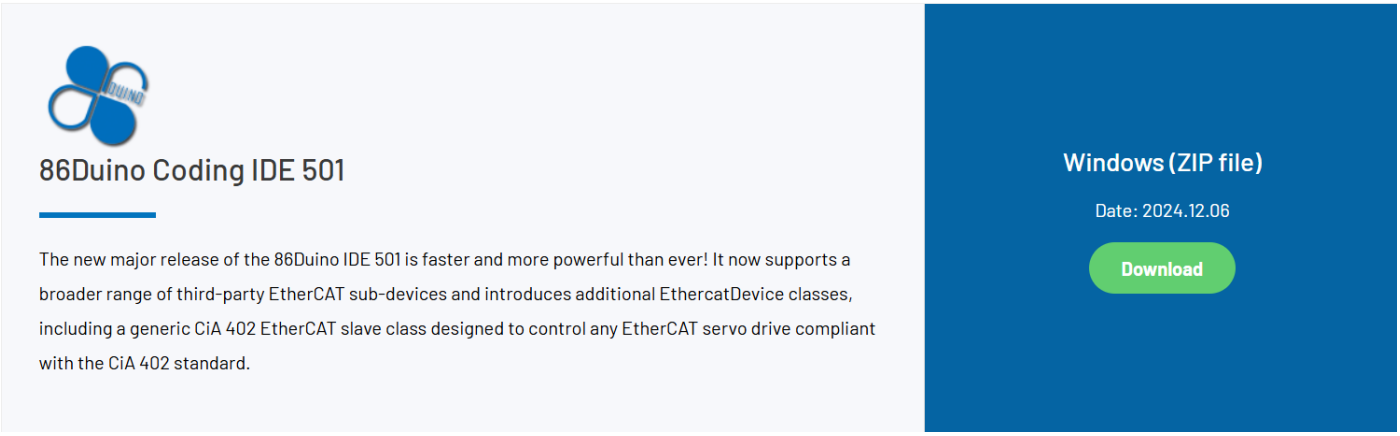


接続は機能ごとに次のように分かれています。

- EtherCAT：M デバイス → IN、OUT はデイジーチェーン接続用
- 電源・接地：
  - VS+/VS−：システム電源 +24 V/GND
  - VP+/VP−：フィールド I/O 電源 +24 V/GND
- Serial COM: DB9; DIP switch selects RS-232/RS-485
  RS-485 では終端抵抗／バイアス抵抗を使用し、基準電位（GND）を共通にしてください
  - データ転送速度（bps）：2400、4800、9600、14400、19200、38400、57600、115200
  - データ幅（ビット）：5/6/7/8
  - ハードウェアフロー制御：CTS/RTS
- ハンドホイール MPG
  - 制御軸：A/B 相直交信号（軸選択付き）
  - 入力：ジョグおよび送り速度オーバーライド
- キーパッド：4×4 マトリクス
- LCM：2 行、1 行 16 文字
- インジケーター：PWR、RUN、LINK、ERROR、TX/RX/MPG/KEYPAD/LCM ステータス

2. ソフトウェア／開発環境

86Duino IDE は https://www.qec.tw/software/ からダウンロードしてください。

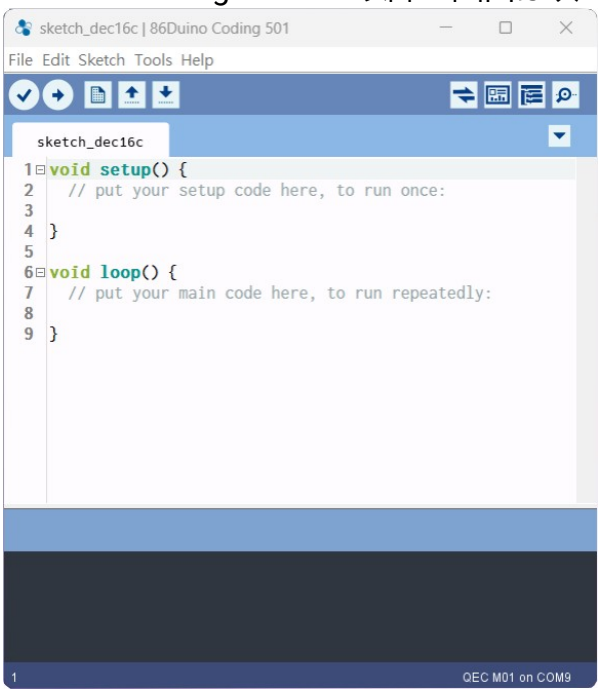


ダウンロードした zip ファイルを解凍してください。追加のソフトウェアをインストールする必要はなく、86duino.exe をダブルクリックするだけで IDE が起動します。



※注意：Windows で警告が表示された場合は、「詳細情報」を 1 回クリックしてから「実行」ボタンを 1 回クリックしてください。

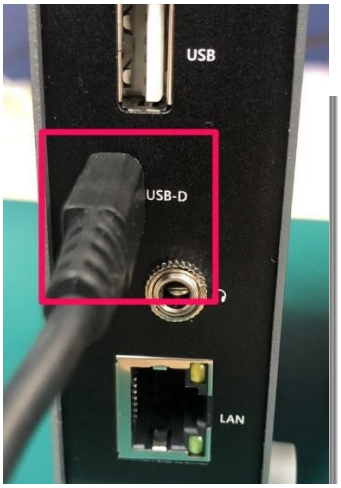
86Duino Coding IDE 501 以降の画面は次のとおりです。



3. PC との接続と環境設定

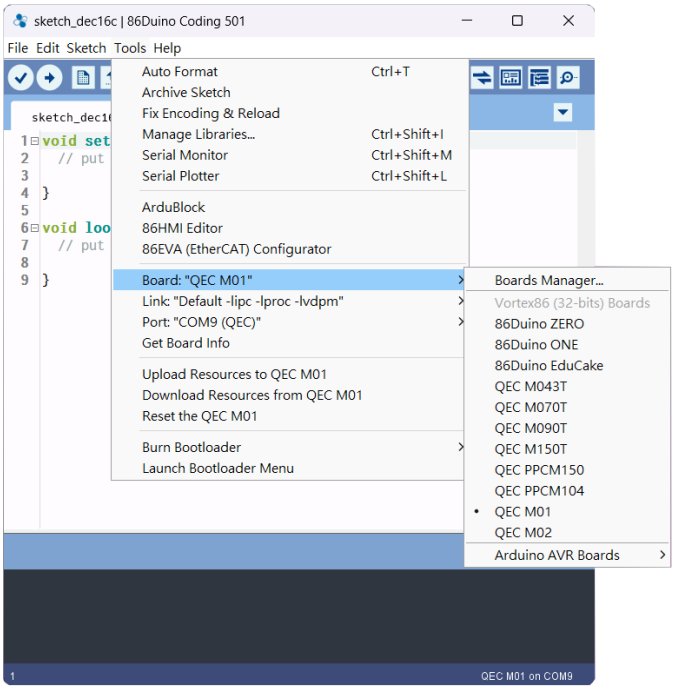
次の手順で環境を設定します。

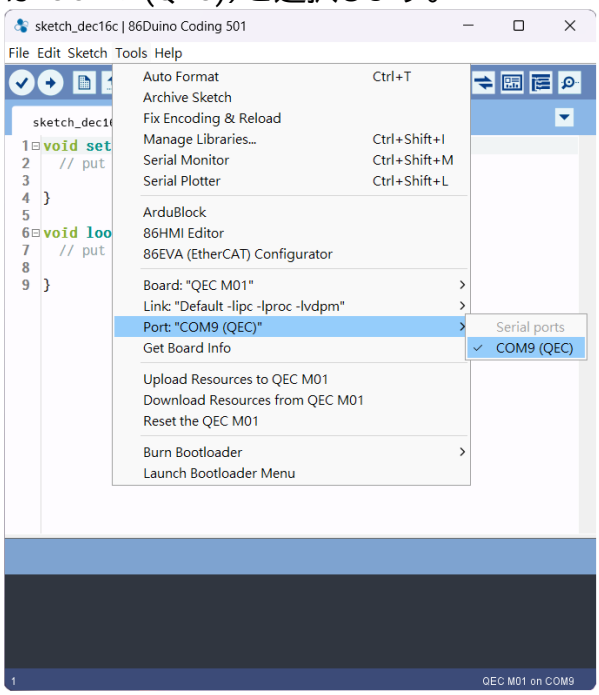
1. Micro USB－USB ケーブルで、QEC-M-01 を PC（86Duino IDE をインストール済み）に接続します。
2. QEC の電源を入れます。
3. PC で「**Device Manager（デバイス マネージャー）**」（Win+X キーを押して表示されるメニューから選択）を開き、「**Ports (COM & LPT)（ポート（COM と LPT））**」を展開します。「**Prolific PL2303GC USB Serial COM Port (COMx)**」が検出されていることを確認してください。表示されない場合は、必要なドライバーをインストールする必要があります。
   （Windows 用 PL2303 ドライバーはこちらからダウンロードできます）





4. 86Duino IDE を起動します。
5. ボードの選択：IDE のメニューから「**Tools（ツール）**」>「**Board（ボード）**」>「**QEC-M-01**」（または使用している QEC M デバイスの型番）を選択します。
6. ポートの選択：IDE のメニューから「**Tools（ツール）**」>「**Port（シリアルポート）**」を選択し、QEC M デバイスが接続されている USB ポート（この例では COM9 (QEC)）を選択します。





4. 86EVA とコードの使用

この例では、86Duino IDE のグラフィカルなローコード開発ツール 86EVA を使用して、EtherCAT M デバイス（QEC-M-01）と QEC-R11HU9S シリーズ（RS232/485、キーパッド、LCM、MPG ハンドホイールを備えた EtherCAT Sub デバイス）を操作する方法を説明します。

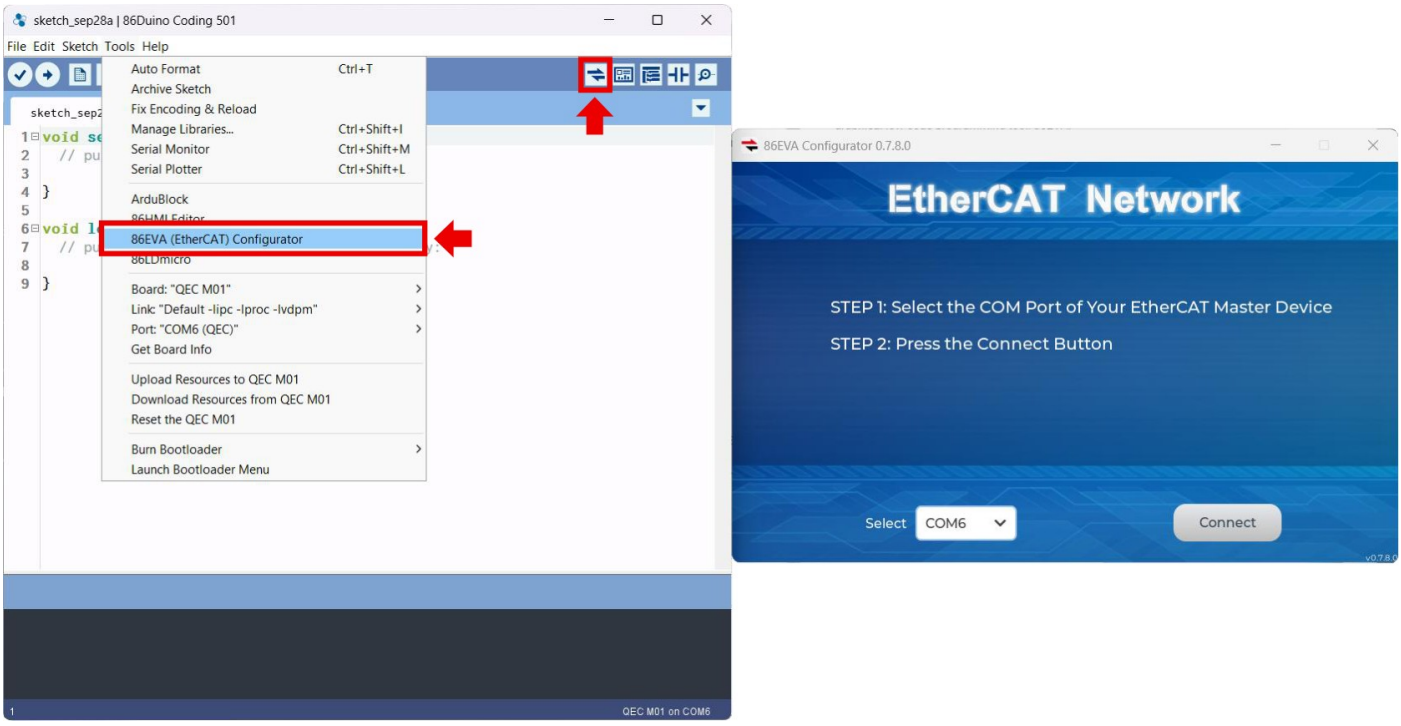
ソフトウェアツールの説明：

- **86EVA（EVA、EtherCAT-Based Virtual Arduino）：**
  86Duino IDE の EtherCAT ライブラリをベースにしたグラフィカルな EtherCAT 設定ツールで、86Duino の開発キットの 1 つです。

以下では、シリアル COM ポート（RS232/RS485）、キーパッドと LCM、MPG ハンドホイールの使い方を順を追って説明します。

ステップ 1：86EVA を起動してスキャンする

86EVA ツールは次のボタンから起動できます。



正しい COM ポートを選択し、「**Connect（接続）**」ボタンをクリックしてください。



QEC-M-01 の正しい COM ポートが選択されていることを確認したら、Connect ボタンを押して EtherCAT ネットワークのスキャンを開始します。



EtherCAT ネットワークのスキャンが完了すると、接続されているデバイスが表示されます。



ステップ 2：パラメーターを設定する

スキャンされたデバイスの画像をダブルクリックすると、対応するパラメーター設定画面が開きます。

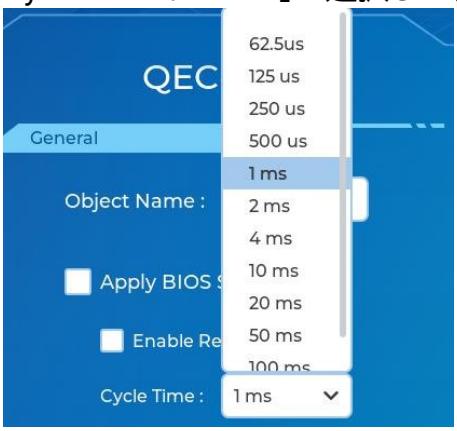
ステップ 2.1：QEC-M-01

QEC-M-01 の画像をダブルクリックして、パラメーター設定を表示します。



次の設定を確認してください。

1. 「**Apply BIOS Settings**」をオフにします。
2. Cycle Time に「**1ms**」を選択します。



左上の「**Back（戻る）**」をクリックして元の画面に戻ります。



ステップ 2.2：QEC-R11HU9S

QEC-R11HU9S の画像をダブルクリックして、パラメーター設定を表示します。



この画面には、Object Name、Alias Address、Vendor ID、Product Code、Virtual Arduino Mapping、Virtual Servo Configuration の各パラメーターが表示されます。

Object Name を「**slave0**」に変更してください。
Object Name をクリックすると、キーボードが表示されます。

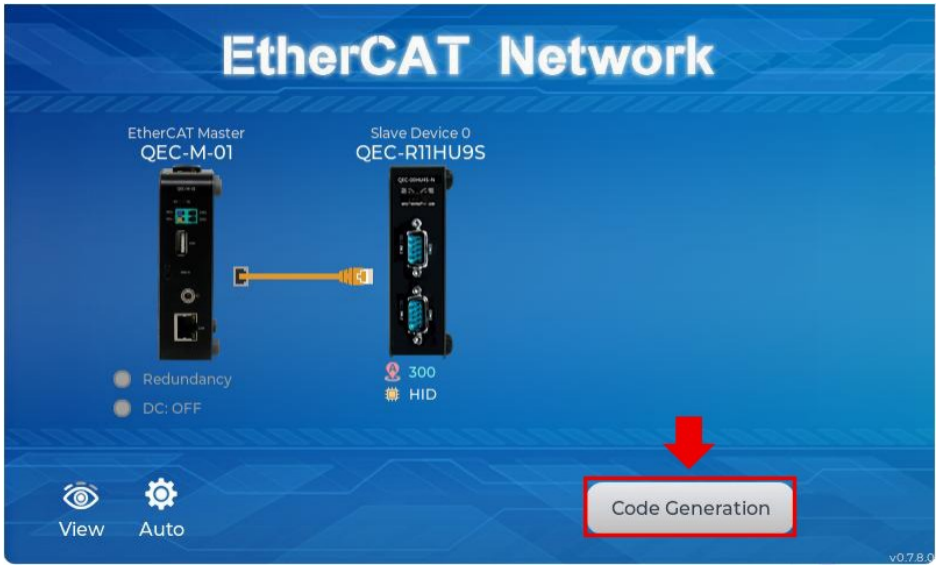


左上の「**Back（戻る）**」をクリックして元の画面に戻ります。

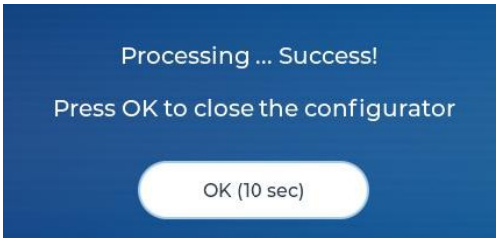


ステップ 3：コードを生成する

デバイスのパラメーターを設定したら、ホーム画面に戻り、右下の「**Code Generation（コード生成）**」ボタンを押します。

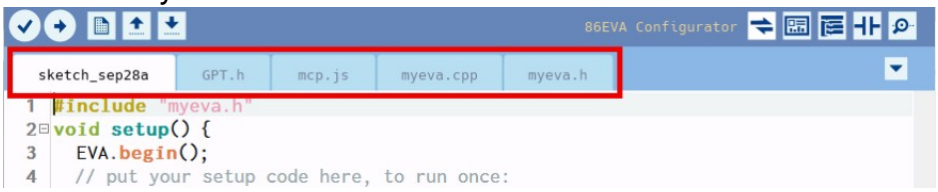


生成が完了したら「**OK**」ボタンをダブルクリックして 86EVA を閉じます。操作しない場合も 10 秒後に自動的に閉じます。



生成されるコードとファイルは次のとおりです。

- sketch_sep10b：メインプロジェクト（.ino、ファイル名はプロジェクト名により異なります）
- myeva.cpp：86EVA の C++ プログラムコード
- myeva.h：86EVA のヘッダーファイル



※補足：

86EVA でコードを生成すると、メインプログラム（.ino）に次のコードが自動的に追加されます。いずれかが欠けていると 86EVA は動作しません。

1. `#include “myeva.h”`：EVA ヘッダーファイルのインクルード
2. `setup()` 内の `EVA.begin();`：EVA 機能の初期化

ステップ 4：コードを記述する

EtherCAT ネットワークを動作させる前に、一度設定を行う必要があります。EtherCAT デバイスは Pre-OP モードから OP モードへ移行させます。86EVA はバックグラウンドで EtherCAT ステートマシンを自動的に処理します。

86EVA が生成するプログラムコードは、デフォルトで次のように設定されています。

- QEC-R11HU9S モジュール：EthercatDevice_QECR11HU9S オブジェクト
- EtherCAT モード：ECAT_SYNC

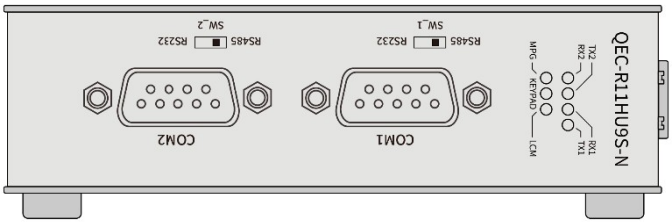
ユーザーが設定した項目は次のとおりです。

- EtherCAT サイクルタイム：1 ミリ秒
- EthercatMaster オブジェクト（“master”）は QEC-M-01 を表し、EthercatDevice_QECR11HU9S オブジェクト（“slave0”）は QEC-R11HU9S モジュールを表します。

以下では、シリアル COM ポート（RS232/RS485）、キーパッドと LCM、MPG ハンドホイールの使い方を順を追って説明します。

ステップ 4.1：シリアル COM ポート（RS232/485）

このセクションでは、86Duino IDE のシリアルモニタからデータを読み取り、COM1（RS-232）から COM2（RS-232）へ転送します。COM2 がデータを受信したら、それをシリアルモニタに表示します。



配線

- 前面の DB9 コネクタを使用します（ピン配置は図を参照）。
- COM1 と COM2 を接続します。外部機器と接続する場合は、TX ↔ RX、RX ↔ TX、GND ↔ GND をそれぞれ接続します。
- ケーブルは適切な長さにし、機器と基準電位（GND）を共通にしてください。

A. setup 関数：

`setup()` 関数では、通信を初期化して OP モードに移行し、COM1 と COM2 を設定します。次の手順で行います。

1. シリアル通信の初期化
    - ボーレート 115200 でシリアル通信を開始します。
2. 86EVA の開始
    - `EVA.begin()` 関数を使用して、EtherCAT ネットワークを開始・初期化します。
3. COM ポート（RS-232）の設定
    - `uartSetBaud(COM, 115200)` 関数を使用して、COM ポートのボーレートを 115200 に設定します。
    - `uartSetFormat(COM, SERIAL_8N1)` 関数を使用して、COM ポートを SERIAL_8N1 フォーマットに設定します。

B. loop 関数：

`loop()` 関数では、EtherCAT／UART の処理を継続しながら、シリアルモニタから受け取った USB シリアルのバイトを **COM1** に転送し、**COM2** からノンブロッキングで読み戻します。

1. ロジック
    - 毎サイクル `update()` を呼び出して、EtherCAT メールボックス／UART キューを処理します。
    - シリアルモニタにバイトがある場合（`Serial.available()`）は、それを読み取り `uartWrite(COM1, byte)` で COM1 に送信します。
    - COM2 をノンブロッキングでポーリングします。`uartQueryRxQueue(COM2) > 0` の場合は `uartRead(COM2)` でバイトを取得し、`Serial.print()` でシリアルモニタに返します。

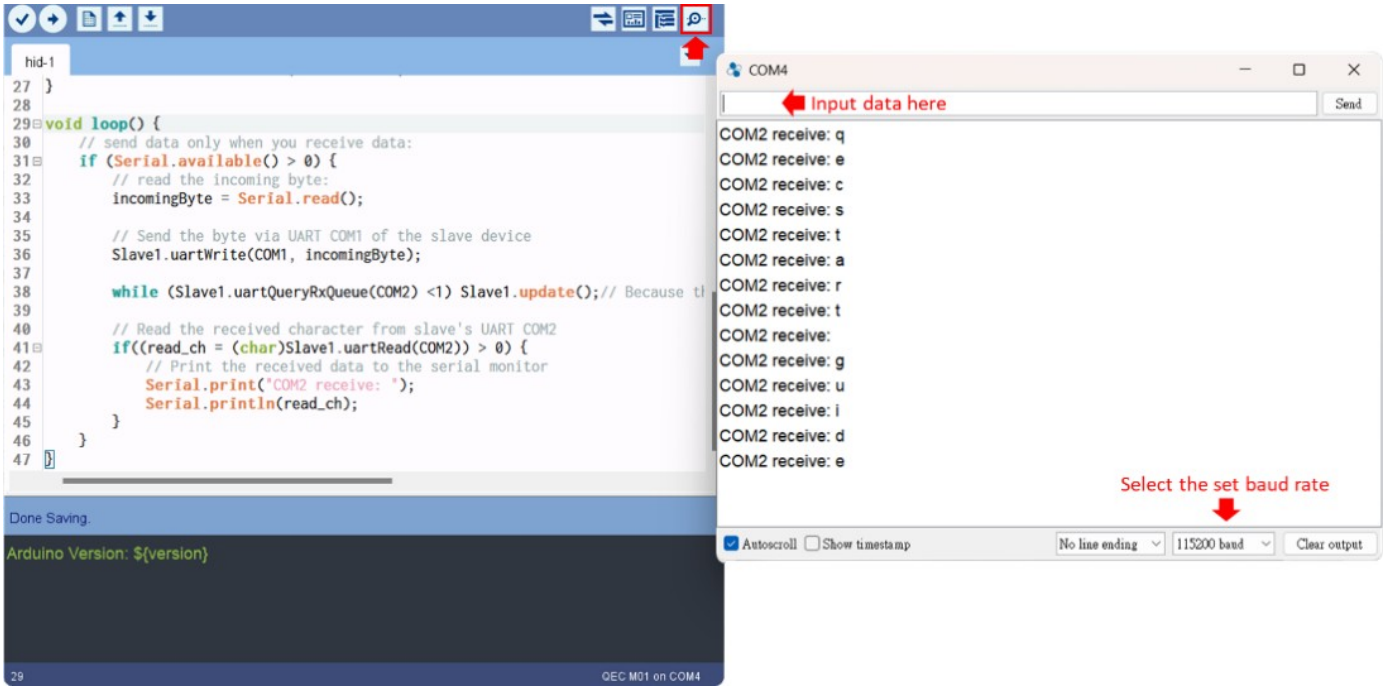
2. コードロジックのまとめ
    - `Serial.available()` で PC 側の RX FIFO を確認し、`Serial.read()` で受信バイトを取得します。
    - `slave0.uartWrite(COM1, b)` で COM1 の RS-232 に送信し、`slave0.uartRead(COM2)` で COM2 を読み取ります。
    - バイナリデータを保持するため、`Serial.print()` で COM2 のデータを PC に返します。
    - リモートシリアル I/O の処理はすべて `loop()` 内に記述し、EtherCAT メールボックスの処理と UART キューの進行を維持するため、`slave0.update()` を頻繁に呼び出します。

サンプルコードは次のとおりです。

```
#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);
    }
  }
}
```

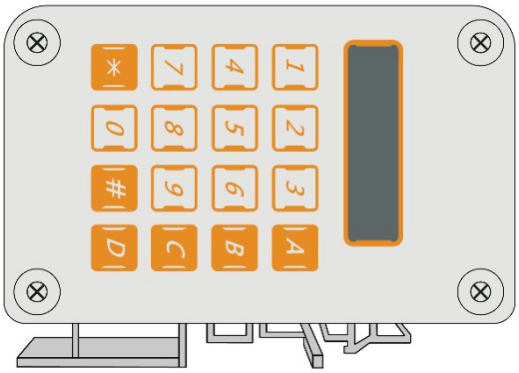
※注意：コードを記述したら、ツールバーの  をクリックしてコンパイルします。コンパイルが完了しエラーがないことを確認したら、 をクリックして書き込みます。

書き込み後、86Duino IDE のシリアルモニタに数字や文字を入力すると、すべてのデータが COM1 から COM2 に転送されます。COM2 がデータを受信すると、下図のようにシリアルモニタに表示されます。



ステップ 4.2：キーパッド + LCM

このセクションでは、キーパッドの入力を読み取り、LCM の指定した位置に表示します。キーが押されるたびにブザーが鳴ります。



- # で LCM をクリアし、表示行を **1 行目** にします。
- * で LCM をクリアし、表示行を **2 行目** にします。
- キー 0～9 は現在の行の **1～10** 桁目に、キー A～D は **11～14** 桁目に表示されます。

A. setup 関数：

`setup()` 関数では、EtherCAT 通信、キーパッド、LCM を初期化します。次の手順で行います。

1. シリアル通信の初期化
    - ボーレート 115200 でシリアル通信を開始します。
2. 86EVA の開始
    - `EVA.begin()` 関数を使用して、EtherCAT ネットワークを開始・初期化します。
3. キーパッドと LCM の初期化
    - `keypadClear()` 関数を使用して、キーパッドを初期化します。
    - `lcmClear()` 関数を使用して、LCM を初期化します。

B. loop 関数：

`loop()` 関数では、キーパッドをノンブロッキングで読み取り、LCM に表示します。

1. ロジック
    - キーの読み取り：`key = slave0.keypadRead();`。有効なキーは {'0'..'9', 'A'..'D', '#', '*'} です。それ以外の値はキー入力なしとしてスキップします。
    - 数字 '0'..'9'：行 = lcmY、桁 = key - '0' + 1（'0'→1 … '9'→10）→ `lcmGotoXY(col, lcmY)` → `lcmWrite(key)`
    - 文字 'A'..'D'：行 = lcmY、桁 = key - 'A' + 11（11..14）→ `lcmGotoXY(col, lcmY)` → `lcmWrite(key)`
    - '#'：lcmY = 1 にして `lcmClear();` を実行
    - '*'：lcmY = 2 にして `lcmClear()` を実行
    - ブザー：有効なキーが押されたときのみ `buzzer(3000, 200)` を実行
    - サービス処理：EtherCAT／メールボックスを維持するため、毎ループ `slave0.update()` を呼び出し

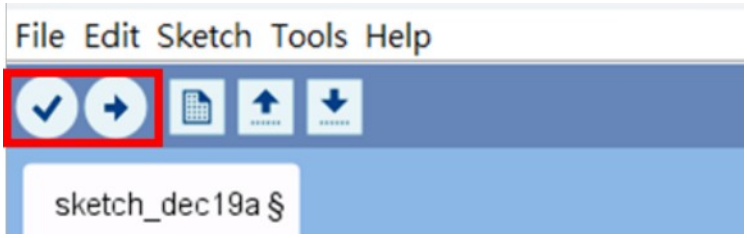
2. コードロジックのまとめ

- `slave0.keypadRead()` でキーを取得します。
- `lcmGotoXY(x, y)` の後に `lcmWrite(ch)` を使用して、指定した桁／行（1 始まり）に表示します。
- I/O の処理はすべて `loop()` 内に記述し、EtherCAT のタイミングを維持するため `slave0.update()` を頻繁に呼び出します。

サンプルコードは次のとおりです。

```
#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();
}
```

※注意：コードを記述したら、ツールバーの  をクリックしてコンパイルします。コンパイルが完了しエラーがないことを確認したら、 をクリックして書き込みます。

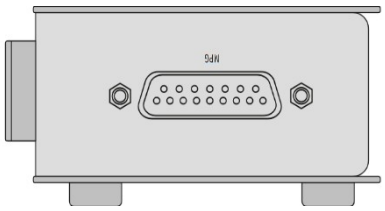


書き込み後、QEC-R11HU9S-N 側面のキーパッドを押すと、押したキーに応じて LCM の指定位置に表示されます。キーパッドを押すとブザーが鳴ります。'#' は LCM の表示をクリアして表示位置を 1 行目に、'*' は LCM の表示をクリアして表示位置を 2 行目にします。



ステップ 4.3：MPG ハンドホイール

このセクションでは、QEC-R11HU9S の MPG のデータとステータスを読み取り、
EMG、Enable、Axis、Ratio、Raw、Logical の各データを 86Duino IDE のシリアルモニタに表示します。



No.	ピン割り当て	No.	ピン割り当て
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. setup 関数：

`setup()` 関数では、EtherCAT 通信を初期化します。次の手順で行います。

1. シリアル通信の初期化
    - ボーレート 115200 でシリアル通信を開始します。
2. 86EVA の開始
    - `EVA.begin()` 関数を使用して、EtherCAT ネットワークを開始・初期化します。

B. loop 関数：

`loop()` 関数では、EtherCAT の処理を継続しながら、MPG の信号を継続的にポーリングして表示します。

1. ロジック
    - 毎サイクル `update()` を呼び出して、EtherCAT メールボックス／キューを処理します。
    - Sub デバイスから MPG の信号を読み取ります：
        - o `mpgReadEmergencyStop()` → 非常停止の状態（EMG）
        - o `mpgReadEnableSwitch()` → イネーブル／ホールドスイッチの状態
        - o `mpgReadAxis()` → 現在選択されている軸（例：0/1/2…）
        - o `mpgReadRatio()` → ハンドホイールの倍率（例：×1/×10/×100）
        - o `mpgReadEncoderRaw()` → エンコーダーの生カウント値
        - o `mpgReadEncoder()` → 論理（処理済み）エンコーダー値
    - `Serial.print()` で値をシリアルモニタに表示し、リアルタイムで確認します。
2. コードロジックのまとめ
    - `Serial.print/println()` で、1 サイクルごとに 1 行の診断情報を出力します。

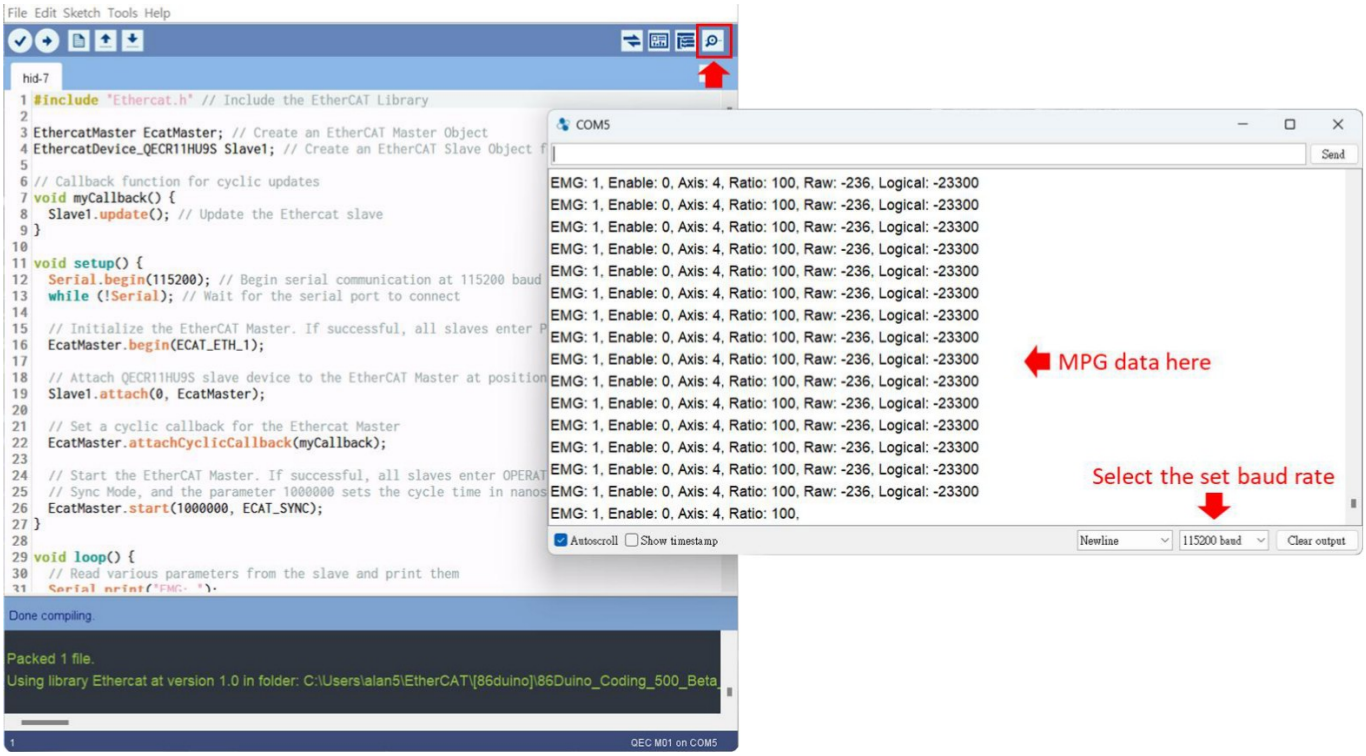
- MPG アクセサー（`mpgRead*()`）で各項目を個別に取得します。
- MPG／エンコーダーの値を最新に保ち、メールボックス通信を進めるため、毎回 `slave0.update()` を呼び出します。

サンプルコードは次のとおりです。

```
#include "myeva.h"

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

void loop() {
  // スレーブから各種パラメーターを読み取って表示
  Serial.print("EMG: ");
  Serial.print(slave0.mpgReadEmergencyStop()); // 非常停止の状態を読み取って表示
  Serial.print(", Enable: ");
  Serial.print(slave0.mpgReadEnableSwitch()); // イネーブルスイッチの状態を読み取って表示
  Serial.print(", Axis: ");
  Serial.print(slave0.mpgReadAxis()); // 軸の情報を読み取って表示
  Serial.print(", Ratio: ");
  Serial.print(slave0.mpgReadRatio()); // 倍率を読み取って表示
  Serial.print(", Raw: ");
  Serial.print(slave0.mpgReadEncoderRaw()); // エンコーダーの生の値を読み取って表示
  Serial.print(", Logical: ");
  Serial.println(slave0.mpgReadEncoder()); // 論理エンコーダー値を読み取って表示

  slave0.update();
}
```

※注意：コードを記述したら、ツールバーの  をクリックしてコンパイルします。コンパイルが完了しエラーがないことを確認したら、 をクリックして書き込みます。

書き込みが完了すると、QEC-R11HU9S の MPG のデータとステータスを読み取り、
EMG、Enable、Axis、Ratio、Raw、Logical などのデータを 86Duino IDE のシリアルモニタで確認できます。


File Edit Sketch Tools Help
hid-7
#include "Ethercat.h" // Include the EtherCAT Library
EthercatMaster EcatMaster; // Create an EtherCAT Master Object
EthercatDevice_QECR11HU9S Slave1; // Create an EtherCAT Slave Object f
// Callback function for cyclic updates
void myCallback() {
Slave1.update(); // Update the Ethercat slave
}
void setup() {
Serial.begin(115200); // Begin serial communication at 115200 baud
while (!Serial); // Wait for the serial port to connect
// Initialize the EtherCAT Master. If successful, all slaves enter P
EcatMaster.begin(ECAT_ETH_1);
// Attach QECR11HU9S slave device to the EtherCAT Master at position
Slave1.attach(0, EcatMaster);
// Set a cyclic callback for the Ethercat Master
EcatMaster.attachCyclicCallback(myCallback);
// Start the EtherCAT Master. If successful, all slaves enter OPERAT
// Sync Mode, and the parameter 1000000 sets the cycle time in nanos
EcatMaster.start(1000000, ECAT_SYNC);
}
void loop() {
// Read various parameters from the slave and print them
Done compiling.
Packed 1 file.
Using library Ethercat at version 1.0 in folder: C:\Users\alan5\EtherCAT\[86duino]\86Duino_Coding_500_Beta_
QEC M01 on COM5
COM5
Send
EMG: 1, Enable: 0, Axis: 4, Ratio: 100, Raw: -236, Logical: -23300
MPG data here
Select the set baud rate
Autoscroll
Show timestamp
Newline
115200 baud
Clear output

トラブルシューティング

QEC-M-01 にコードを書き込めない場合

コードを正常に書き込めない場合は、86EVA を開いて QEC EtherCAT M デバイスの環境に異常がないか確認してください。下図のように異常がある場合は、QEC EtherCAT M デバイスの環境を更新してください。更新対象は Bootloader、EtherCAT ファームウェア、EtherCAT ツールの 3 項目です。



以下、更新の手順を説明します。

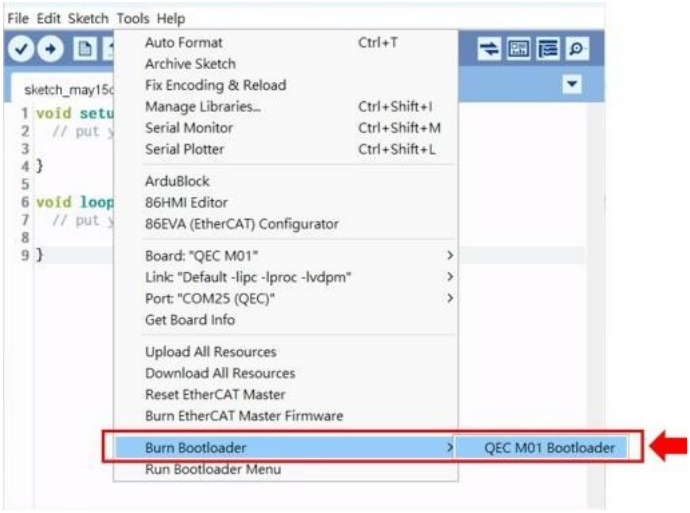
ステップ 1：QEC-M の準備

1. 86Duino IDE 500 以降（またはそれより新しいバージョン）をダウンロードしてインストールします。
   ダウンロードは Software ページから行えます。
2. QEC-M の接続：USB ケーブルで QEC-M をコンピューターに接続します。
3. 86Duino IDE の起動：インストールが完了したら、86Duino IDE を起動します。
4. ボードの選択：IDE のメニューから「**Tools（ツール）**」>「**Board（ボード）**」>「**QEC-M-01**」（または使用している QEC-M の型番）を選択します。
5. ポートの選択：IDE のメニューから「**Tools（ツール）**」>「**Port（シリアルポート）**」を選択し、QEC-M が接続されている USB ポートを選択します。

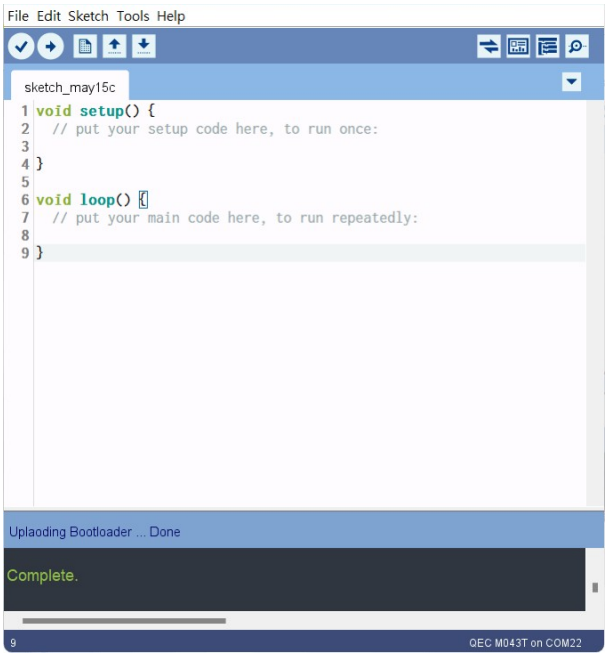
ステップ 2：「Burn Bootloader（ブートローダを書き込む）」をクリック

QEC-M 製品を接続したら、「**Tools（ツール）**」>「**Burn Bootloader（ブートローダを書き込む）**」を選択します。
現在選択されている QEC-M の名前が表示されます。これをクリックすると更新が始まります。更新には約 5～20 分かかります。

- QEC-M-01：



ステップ 3：更新の完了



以上の手順で、QEC-M の開発環境が最新バージョンに更新されます。

保証

本製品は、ご購入日から 1 年間、正常に動作することを保証します。この期間中に本製品が正常に動作しなくなった場合は、以下の条件に定める場合を除き、当社の判断により無償で交換または修理いたします。本保証は、誤用、改造、事故または災害により損傷した製品には適用されません。販売元は、本製品の使用、誤用、または使用不能により生じたいかなる損害、逸失利益、逸失した節約、その他の付随的または結果的損害についても一切の責任を負いません。また、販売元はその他の関係者からのいかなる請求についても責任を負いません。返品をお受けするには、事前に販売元から返品承認を得る必要があります。返品承認は、販売元に電話または FAX で連絡し、返品承認番号（RMA 番号）を請求することで取得できます。返品の際は、必ず問題の内容を明記した説明を添付してください。

本書に記載されているすべての商標は、それぞれの所有者の登録商標です。
すべての仕様は予告なく変更されることがあります。
©ICOP Technology Inc. 2025

日本語版資料は、英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。