QEC スタートガイド 8:3 軸ステッピングモーター(CiA402 PP モード)

本ガイドでは、EtherCAT M デバイス QEC-M-01P と QEC-RXXMP3S シリーズ(EtherCAT Sub デバイス、3 軸ステッピングモーターコントローラー)の使用方法を説明します。CiA402 のプロファイル位置(PP)モードで動作させます。

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。



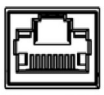
非 PoE タイプ



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。



ピン番号	信号名	ピン番号	信号名
1	LAN1_TX+	2	LAN1_TX-
3	LAN1_RX+	4	VS+
5	VP+	6	LAN1_RX-
7	VS- (GND)	8	VP- (GND)

* PoE 対応 LAN は赤色ハウジング、通常の LAN は黒色ハウジングです。

* L4、L5、L7、L8 ピンはオプションで、RJ45 の電源入出力用です。

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)。

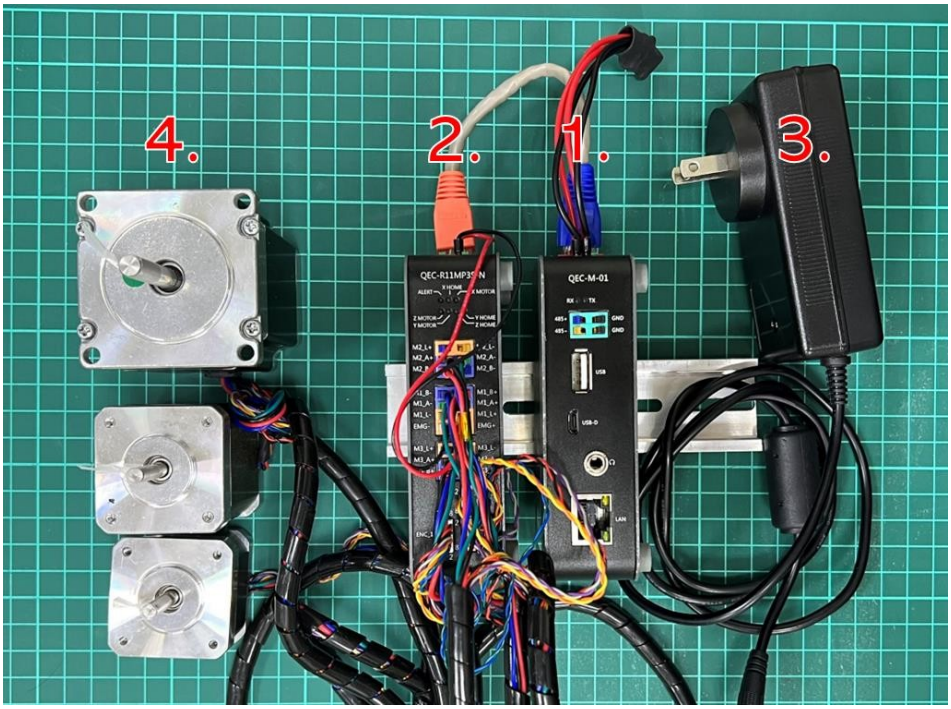


3. QEC の PoE 給電能力は最大 24V/3A です。

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。

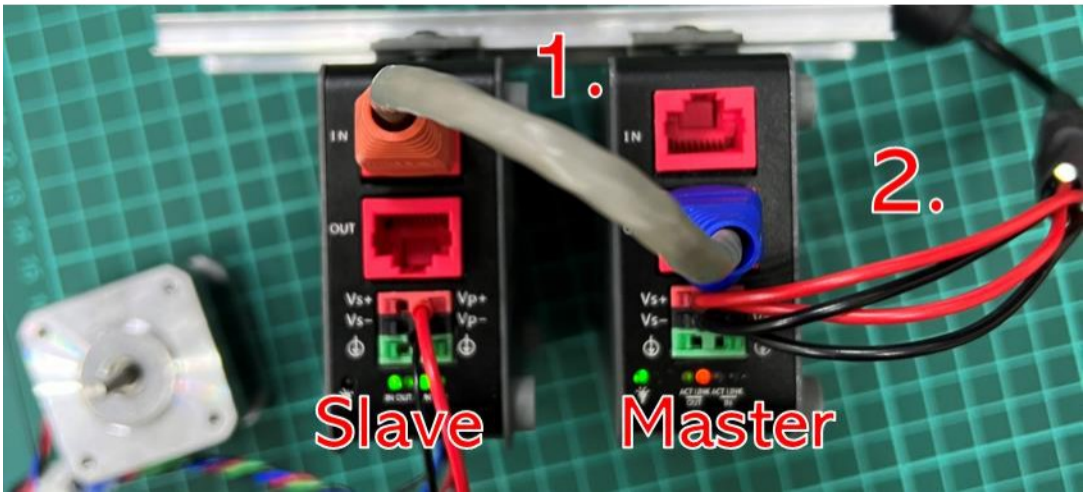
1. QEC-M-01P（EtherCAT M デバイス／PoE）
2. QEC-R11MP3S（EtherCAT Sub デバイス、3 軸ステッピングモーターコントローラー）、LAN ケーブル
3. 24V 電源、EU タイプ端子ケーブル
4. エンコーダーインターフェース付き 4 線式 2 相バイポーラ 42 ステッピングモーター ×3（86STEP | 86Duino を参照）



QEC-M-01P

PoE 機能付きの QEC EtherCAT M デバイスです。

1. EtherCAT Out ポート（上側）から RJ45 ケーブルで QEC-R11MP3S の EtherCAT In ポートに接続します（PoE で給電）。
2. EU 端子を介して Vs+/Vs- および Vp+/Vp- に 24V 電源を接続します。また、PoE 機能で QEC-R11MP3S-N に給電します。

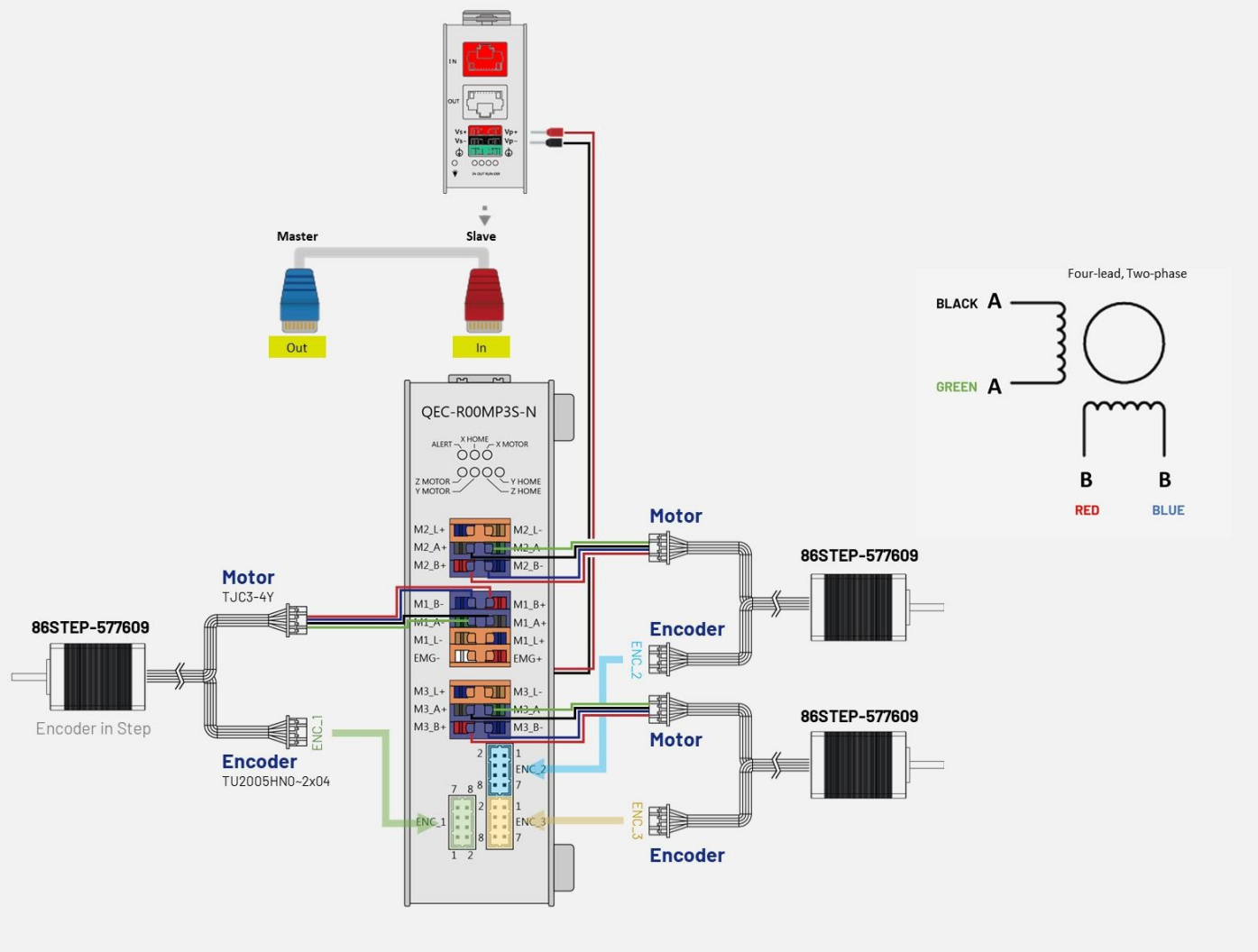


QEC-R11MP3S-N

PoE 機能付きの QEC-R11MP3S-N です。

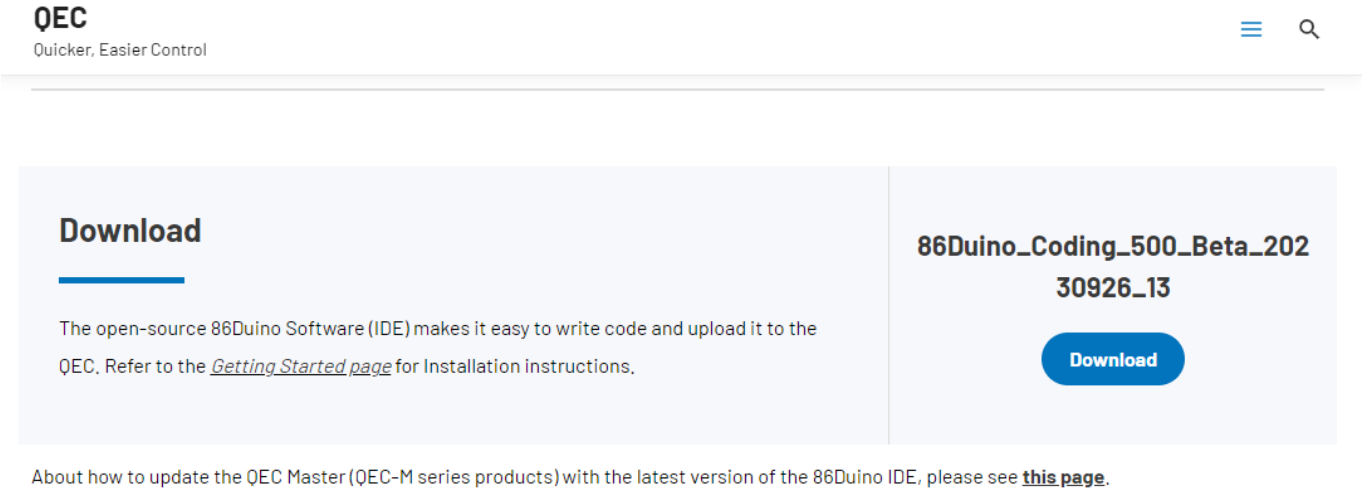
- Vp+ を EMG+ に、Vp- を EMG- に接続します(モーターへの電源供給)。
- 3 台の 4 線式 2 相 42 ステッピングモーターを、次のように QEC-R11MP3S-N の M1／M2／M3 軸に接続します。
  - i. A+(黒)を M_A+ に、A-(緑)を M_A- に接続します。
  - ii. B+(赤)を M_B+ に、B-(青)を M_B- に接続します。
- 3 台の 4 線式 2 相 42 ステッピングモーターのエンコーダーを、それぞれ QEC-R11MP3S-N の ENC_1／ENC_2／ENC_3 に接続します。

※注意:モーターの配線色は QEC-R11MP3S-N の端子の色と同じなので、色を目安に配線できます。



ソフトウェア／開発環境:86Duino IDE

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

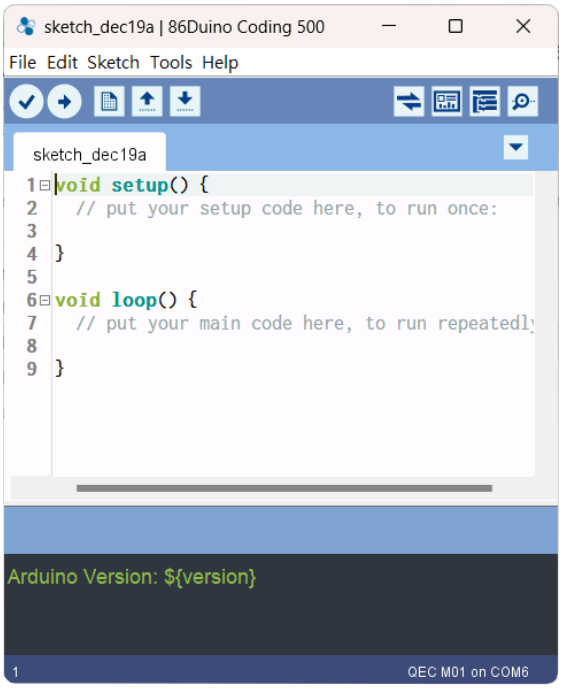


ダウンロードした zip ファイルを解凍してください。追加のソフトウェアをインストールする必要はなく、86duino.exe をダブルクリックするだけで IDE が起動します。



※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。

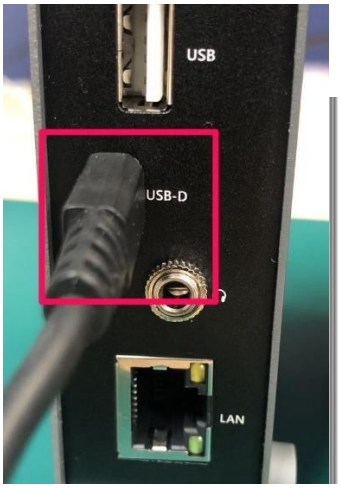
86Duino Coding IDE 500 以降の画面は次のとおりです。



PC との接続と環境設定

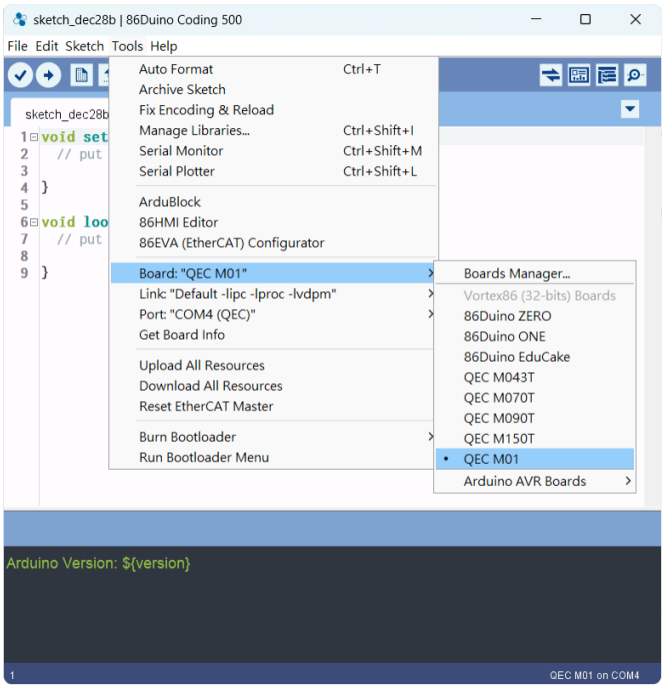
次の手順で環境を設定します。

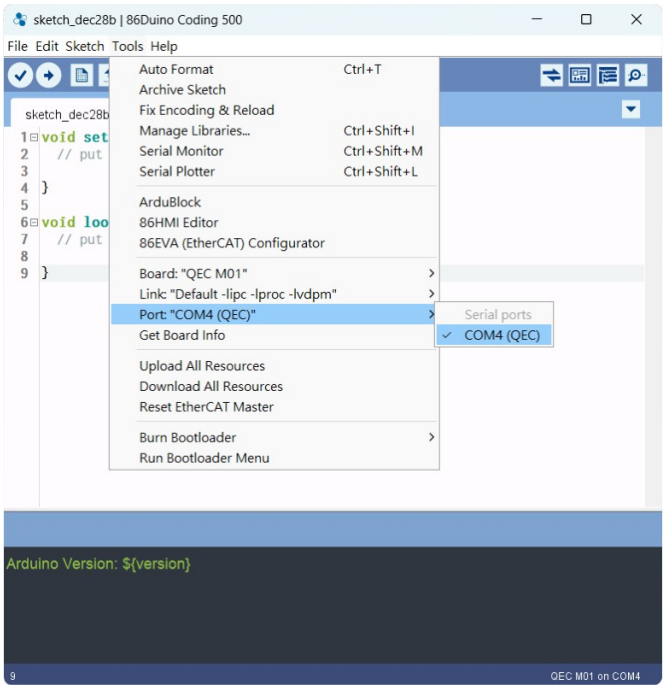
1. Micro USB－USB ケーブルで、QEC-M-01P を PC(86Duino IDE をインストール済み)に接続します。
2. QEC の電源を入れます。
3. PC で「Device Manager(デバイス マネージャー)」(Win+X キーを押して表示されるメニューから選択)を開き、「Ports (COM & LPT)(ポート(COM と LPT))」を展開します。「Prolific PL2303GC USB Serial COM Port (COMx)」が検出されていることを確認してください。表示されない場合は、必要なドライバーをインストールする必要があります。(Windows 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 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 M デバイスが接続されている USB ポート(この例では COM4 (QEC))を選択します。





開発方法 1:コードを記述する

EtherCAT M デバイス(QEC-M-01P)と Sub デバイス(QEC-R11MP3S-N)は、86Duino IDE の EtherCAT ライブラリで設定とプログラミングができます。Arduino の開発環境は主に `setup()` と `loop()` の 2 つの部分で構成されます。`setup()` は一度だけ実行される初期化処理、`loop()` は繰り返し実行されるメインプログラムに対応しています。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。

この例では、EtherCAT 通信プロトコルと DC モードを組み合わせて同期モーション制御を行います。QEC-R11MP3S コントローラーで 3 軸のステッピングモーターを制御します。モーション制御には、あらかじめ定義した経路に基づく位置制御モードである CiA402 規格のプロファイル位置(PP)モードを使用します。`profilePositionBegin()` 関数を呼び出して、速度・加速度 10,000 でモーターを特定の位置(81920)へ移動させます。コマンドの送信後、ドライバーのコントロールワードでコマンドの受信を確認し、モーターが所定の位置に到達したことを確かめます。完了したら目標位置を 0 に戻し、同じ速度と加速度で移動させることで、モーターを 2 点間で往復させます。

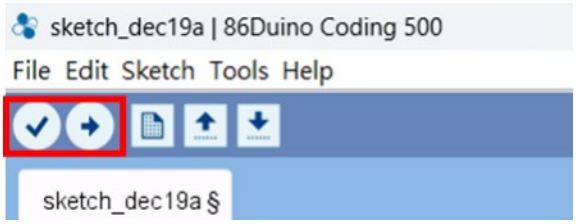
CiA402 オブジェクトに対応した他社製 Sub デバイスの場合:
EtherCAT の CiA402 オブジェクトを操作する場合は、専用の EtherCAT CiA402 ライブラリを使用できます。

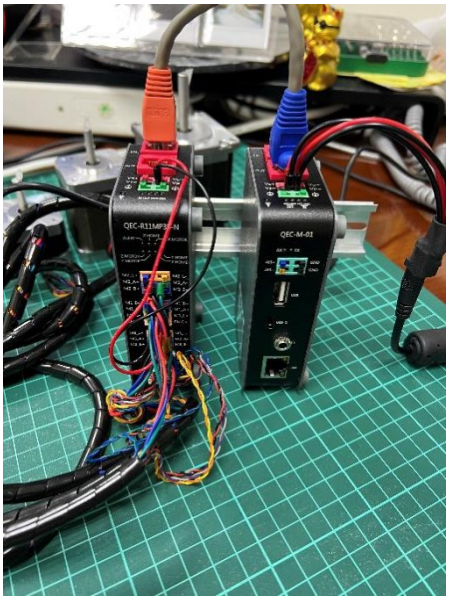
```
#include "Ethercat.h" // EtherCAT ライブラリをインクルード


#define MOTORS      (3) // モーターの数を定義


EthercatMaster master; // EtherCAT Mデバイスのオブジェクトを作成
EthercatDevice_CiA402 motor[MOTORS]; // CiA402 モーター用の EtherCAT Subデバイスオブジェクトを作成


int pp_state[MOTORS]; // 各モーターのプロファイル位置の状態を保持する配列
int pp_done; // プロファイル位置動作の完了を管理する変数


// 周期更新用のコールバック関数
void myCallback() {
  for (int i = 0; i < MOTORS; i++) {
    motor[i].update(); Slave1.update(); // update() を手動で呼び出す
  }
}


void setup(void) {
  // EtherCAT Mデバイスを初期化。成功すると、すべての Subデバイスが PRE OPERATIONAL 状態になる
  master.begin(ECAT_ETH_1);


  for (int i = 0; i < MOTORS; i++) {
    // CiA402 モーターを EtherCAT Mデバイスに接続
    // 第 1 引数:EtherCAT ノード番号、第 2 引数:ステッピングモーター番号
    motor[i].attach(0, i, master);
```

```
    motor[i].setDc(1000000); // 分散クロック（DC）の同期サイクルタイムを設定
    motor[i].driveSetMode(CIA402_PP_MODE); // モータードライブの動作モードをプロファイル位置モードに設定
  }

  master.attachCyclicCallback(myCallback); // EtherCAT Mデバイスに周期コールバックを登録
  // EtherCAT Mデバイスを開始。成功すると、すべての Subデバイスが OPERATIONAL 状態になる
  master.start(1000000); // サイクルタイムを 1ms（1,000,000 ns）に設定

  for (int i = 0; i < MOTORS; i++) {
    motor[i].driveEnable(); // モーターを有効化（CiA402 の状態を OPERATION_ENABLED に設定）
  }
}

void loop() {
  for (int i = 0; i < MOTORS; i++) {
    // 各モーターのプロファイル位置制御用ステートマシン
    switch (pp_state[i]) {
      case 0:
        // 指定した速度と加速度で、モーターの目標位置への移動を開始
        if (motor[i].profilePositionBegin(81920, 10000, 10000) == 0)
          pp_state[i]++; // コマンドが成功したら次の状態へ
        break;
      case 1:
        // モーターのコントロールワードが次のコマンドを受け付けられるか確認
        if ((motor[i].driveGetControlword() & (1 << 4)) == 0)
          pp_state[i]++; // コントロールワードの準備ができたら次の状態へ
        break;
      case 2:
        // モーターが目標位置に到達したか確認
        if (motor[i].driveIsTargetReached())
          pp_state[i]++; // 目標位置に到達したら次の状態へ
        break;
      case 3:
        // モーターに元の位置へ戻るよう指令
        if (motor[i].profilePositionBegin(0, 10000, 10000) == 0)
          pp_state[i]++; // コマンドが成功したら次の状態へ
        break;
      case 4:
        // コントロールワードが次のコマンドを受け付けられるか再確認
        if ((motor[i].driveGetControlword() & (1 << 4)) == 0)
          pp_state[i]++; // コントロールワードの準備ができたら次の状態へ
```

```
        break;
      case 5:
        // モーターが初期位置に戻ったかを確認する最終状態
        if (motor[i].driveIsTargetReached()) {
          pp_state[i] = 0; // このモーターのステートマシンをリセット
          pp_done++; // プロファイル位置シーケンスを完了したモーターの数を加算
        }
        break;
    }
  }

  // すべてのモーターが移動を完了したら全体をリセット
  if (pp_done == MOTORS) {
    pp_done = 0; // 完了カウンターをリセット
    for (int i = 0; i < MOTORS; i++)
      pp_state[i] = 0; // すべての状態を 0 にリセット
  }
}
```

※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プログラムが実行されます。





書き込みが完了すると、3 台のステッピングモーターがすべて指定位置へ移動し、初期位置に戻る様子を確認できます。
この動作が繰り返されます。

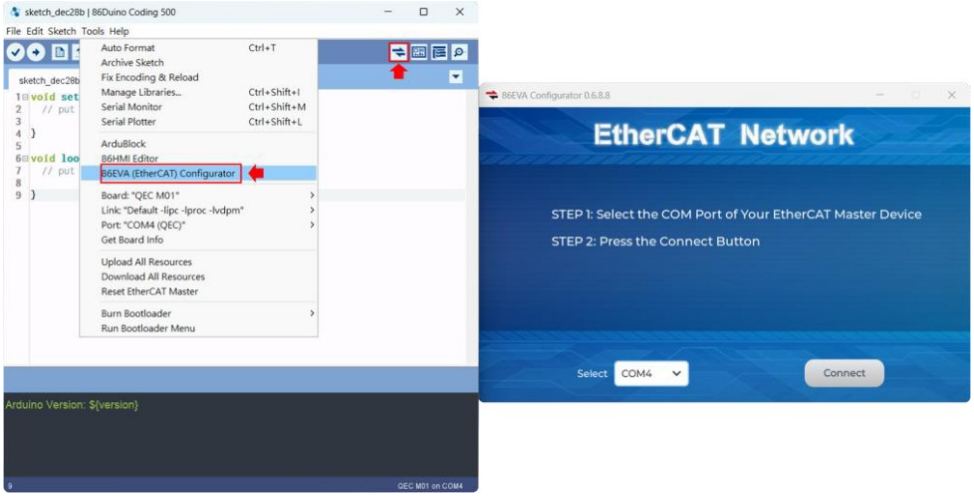
開発方法 2:86EVA とコードを使用する

86EVA は、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。EtherCAT ネットワークをすばやく設定して、プログラミングを始められます。

この例では、EtherCAT 通信プロトコルと DC モードを組み合わせて同期モーション制御を行います。QEC-R11MP3S コントローラーで 3 軸のステッピングモーターを制御します。モーション制御には、あらかじめ定義した経路に基づく位置制御モードである CiA402 規格のプロファイル位置(PP)モードを使用します。`profilePositionBegin()` 関数を呼び出して、速度・加速度 10,000 でモーターを特定の位置(81920)へ移動させます。コマンドの送信後、ドライバーのコントロールワードでコマンドの受信を確認し、モーターが所定の位置に到達したことを確かめます。完了したら目標位置を 0 に戻し、同じ速度と加速度で移動させることで、モーターを 2 点間で往復させます。

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。

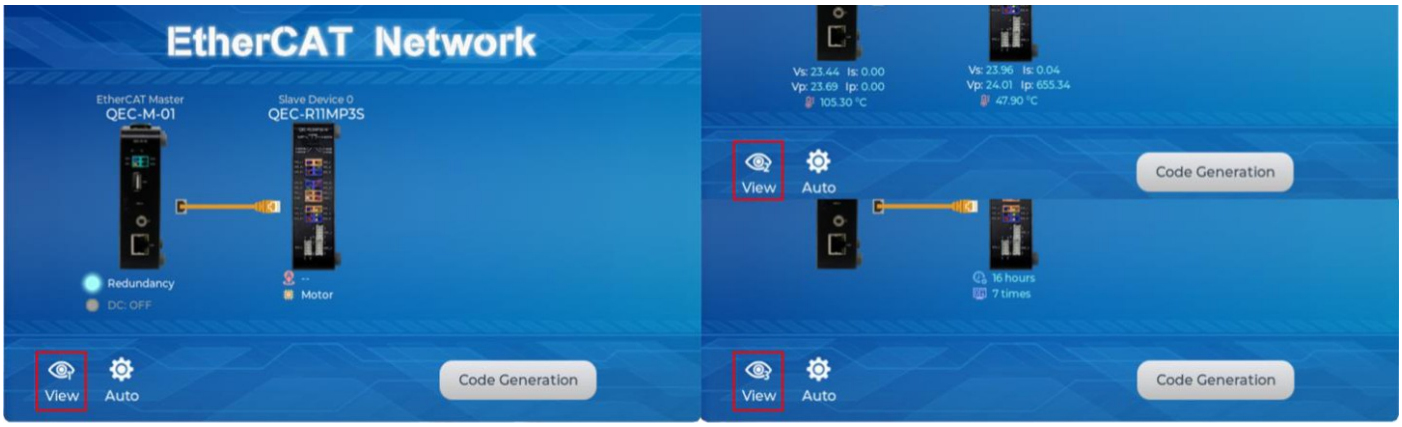


QEC-M-01P の正しい COM ポートが選択されていることを確認したら、Connect ボタンを押して EtherCAT ネットワークのスキャンを開始します。



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。

左下の「View」ボタンを押すと、デバイスの状態(電圧、電流、温度:View2)と稼働時間(時間:View3)を確認できます。



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。



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。

この例ではデフォルト設定のまま変更しません。左上の「Back(戻る)」をクリックして元の画面に戻ってください。



QEC-R11MP3S-N

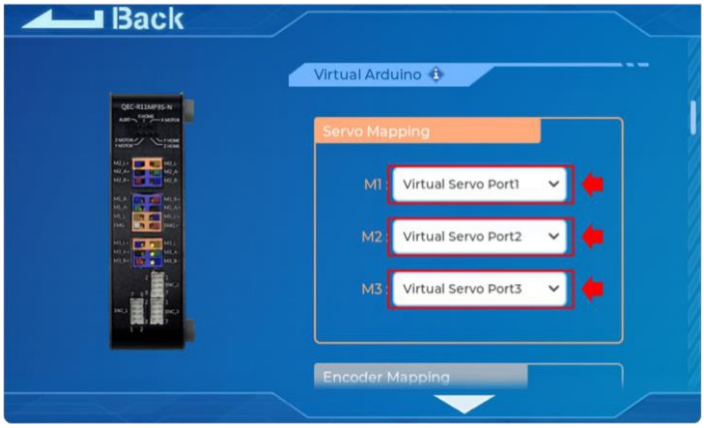
QEC-R11MP3S の画像をダブルクリックして、パラメーター設定を表示します。



「General」の「Device Mode」欄に注意してください。選択肢は CiA-402 Servos と G-code Machine の 2 つで、この例では CiA-402 Servos モードを使用します(デフォルトが CiA-402 Servos モードに設定されていない場合は、トラブルシューティング:QEC-RXXMP3S でモーターを制御できない を参照してください)。



「Servo Mapping」セクションに進み、M1 のドロップダウンで「Virtual Servo Port1」を、M2 で「Virtual Servo Port2」を、M3 で「Virtual Servo Port3」を選択します。



設定が終わったら、左上の「Back(戻る)」をクリックして元の画面に戻ります。



これにより、QEC-R11MP3S の M1(1 軸目のモーター)、M2(2 軸目のモーター)、M3(3 軸目のモーター)が、それぞれ EVA の Virtual Servo Port1、Virtual Servo Port2、Virtual Servo Port3 に割り当てられます。

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。

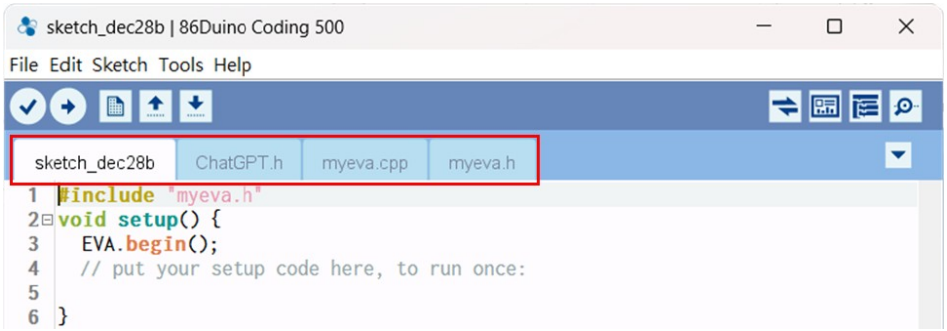


生成が完了したら OK ボタンをダブルクリックして 86EVA を閉じます。操作しない場合も 10 秒後に自動的に閉じます。



生成されるコードとファイルは次のとおりです。

- sketch_dec28b: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)
- ChatGPT.h:ChatGPT に参照させるためのパラメーター
- 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

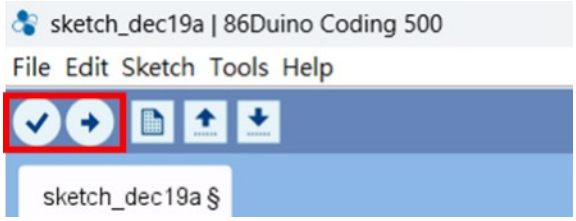
ステップ 4:コードを記述する

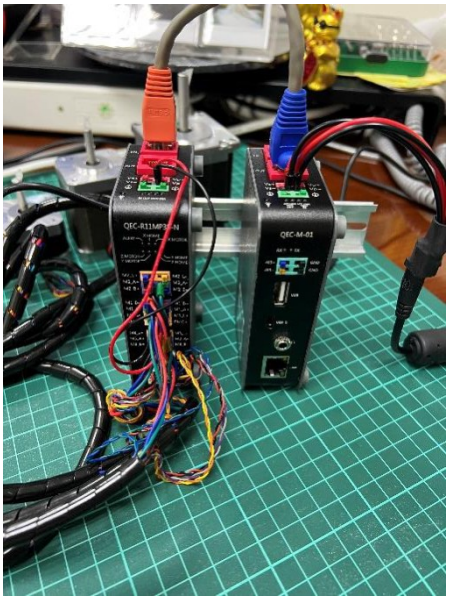
この例では、EtherCAT 通信プロトコルと DC モードを組み合わせて同期モーション制御を行います。QEC-R11MP3S コントローラーで 3 軸のステッピングモーターを制御します。モーション制御には、あらかじめ定義した経路に基づく位置制御モードである CiA402 規格のプロファイル位置(PP)モードを使用します。`profilePositionBegin()` 関数を呼び出して、速度・加速度 10,000 でモーターを特定の位置(81920)へ移動させます。コマンドの送信後、ドライバーのコントロールワードでコマンドの受信を確認し、モーターが所定の位置に到達したことを確かめます。完了したら目標位置を 0 に戻し、同じ速度と加速度で移動させることで、モーターを 2 点間で往復させます。

```
#include "myeva.h"

#define MOTORS      3 // モーターの数を定義
EthercatDevice_CiA402* motor[MOTORS]; // CiA402 モーター用の EtherCAT Sub デバイスオブジェクトへのポインターを保持する配列

int pp_state[MOTORS]; // 各モーターのプロファイル位置の状態を保持する配列
int pp_done = 0; // プロファイル位置動作の完了を管理する変数(初期値 0)

void setup() {
  EVA.begin();
  // cia402GetServo() が返すポインターをモーター配列の要素に直接代入
  motor[0] = VirtualServo1.cia402GetServo();
  motor[1] = VirtualServo2.cia402GetServo();
  motor[2] = VirtualServo3.cia402GetServo();
  for (int i = 0; i < MOTORS; i++) {
    motor[i]->setDc(1000000); // ポインターのアクセスにはアロー演算子を使用
    motor[i]->driveEnable(); // ポインターのアクセスにはアロー演算子を使用
  }
}

void loop() {
  for (int i = 0; i < MOTORS; i++) {
    // 各モーターのプロファイル位置制御用ステートマシン
    switch (pp_state[i]) {
      case 0:
        if (motor[i]->profilePositionBegin(81920, 10000, 10000) == 0)
          pp_state[i]++; // コマンドが成功したら次の状態へ
        break;
      case 1:
        if ((motor[i]->driveGetControlword() & (1 << 4)) == 0)
          pp_state[i]++; // コントロールワードの準備ができたら次の状態へ
        break;
```

```
      case 2:
        if (motor[i]->driveIsTargetReached())
          pp_state[i]++; // 目標位置に到達したら次の状態へ
        break;
      case 3:
        if (motor[i]->profilePositionBegin(0, 10000, 10000) == 0)
          pp_state[i]++; // コマンドが成功したら次の状態へ
        break;
      case 4:
        if ((motor[i]->driveGetControlword() & (1 << 4)) == 0)
          pp_state[i]++; // コントロールワードの準備ができたら次の状態へ
        break;
      case 5:
        if (motor[i]->driveIsTargetReached()) {
          pp_state[i] = 0; // このモーターのステートマシンをリセット
          pp_done++; // プロファイル位置シーケンスを完了したモーターの数を加算
        }
        break;
    }
  }

  if (pp_done == MOTORS) {
    pp_done = 0; // 完了カウンターをリセット
    for (int i = 0; i < MOTORS; i++)
      pp_state[i] = 0; // すべての状態を 0 にリセット
  }
}
```

※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プログラムが実行されます。





書き込みが完了すると、3 台のステッピングモーターがすべて指定位置へ移動し、初期位置に戻る様子を確認できます。
この動作が繰り返されます。

トラブルシューティング

QEC-M-01 の環境が古い場合:

スケッチを正常に書き込めない場合は、QEC EtherCAT M デバイス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。

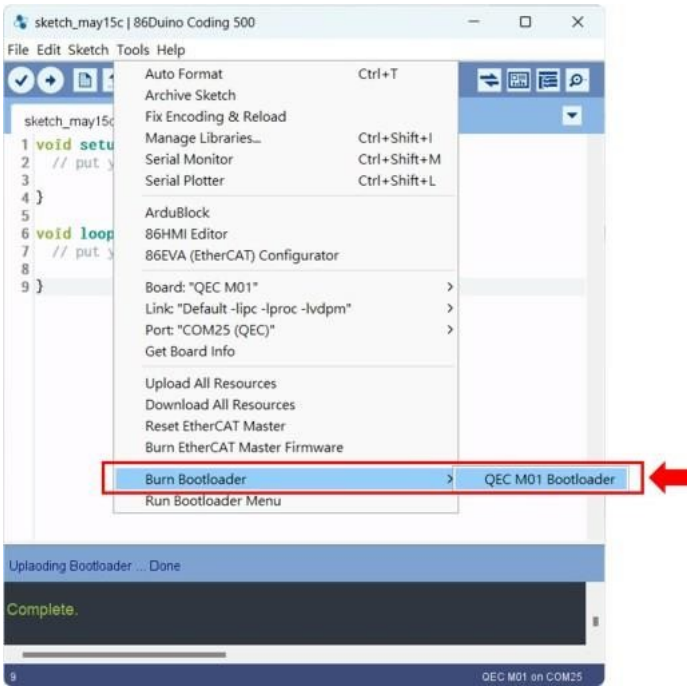
ステップ 1:QEC-M の準備

1. 86Duino IDE 500(またはそれ以降のバージョン)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。

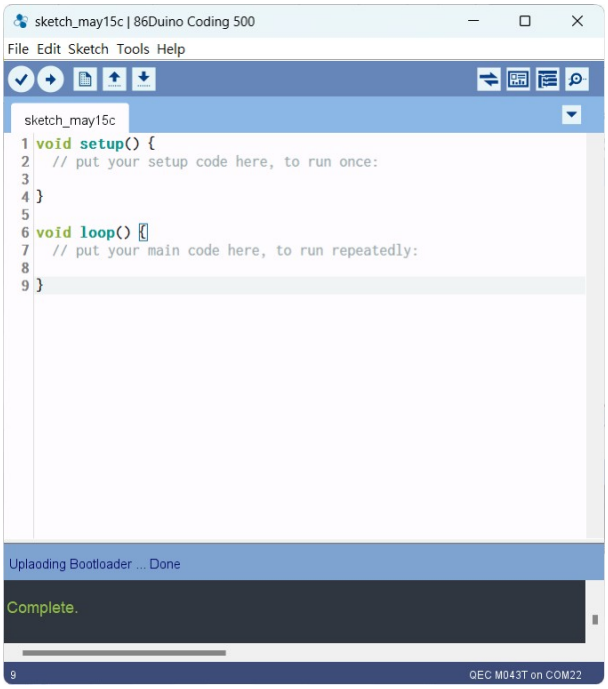
ステップ 2:「Burn Bootloader(ブートローダを書き込む)」をクリック

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。

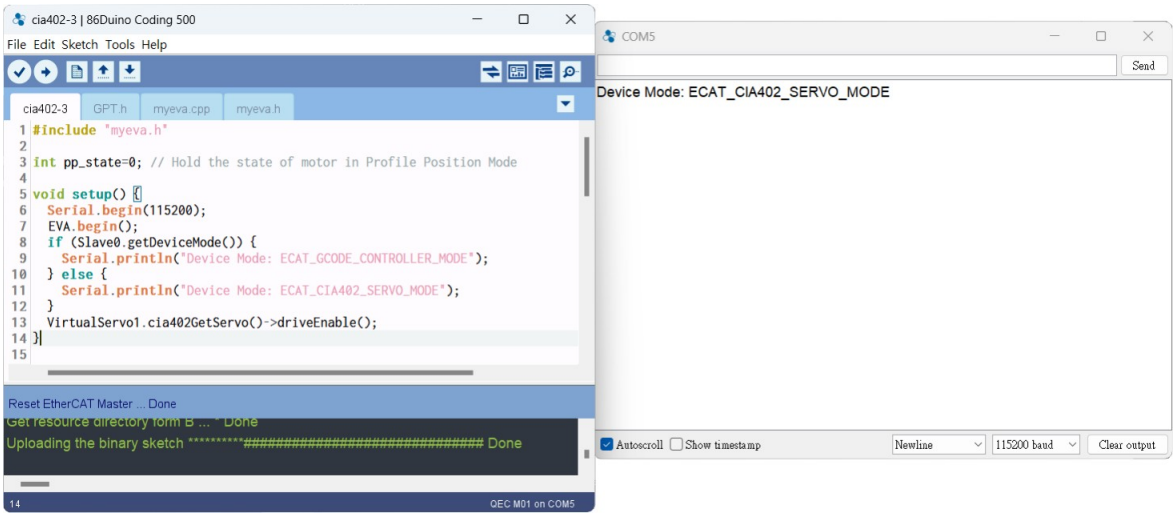
QEC-RXXMP3S でモーターを制御できない場合：

モード設定の誤り：

QEC-RXXMP3S では、CiA-402 と G コードの 2 種類のモーション制御方式を使用できます。CiA-402 モードと G コードモードには互換性がないため、それぞれに対応したコマンドセットと制御方法を使用する必要があります。

QEC-RXXMP3S の現在のモードは、コード `getDeviceMode();` または 86EVA で確認できます。

86Duino IDE で確認する場合：



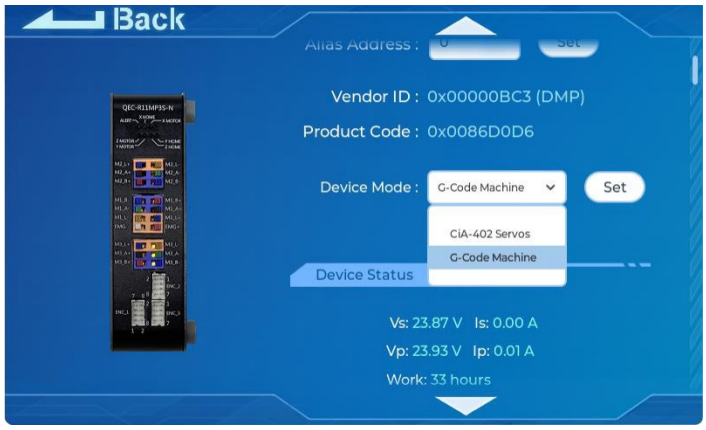
86EVA で確認する場合：



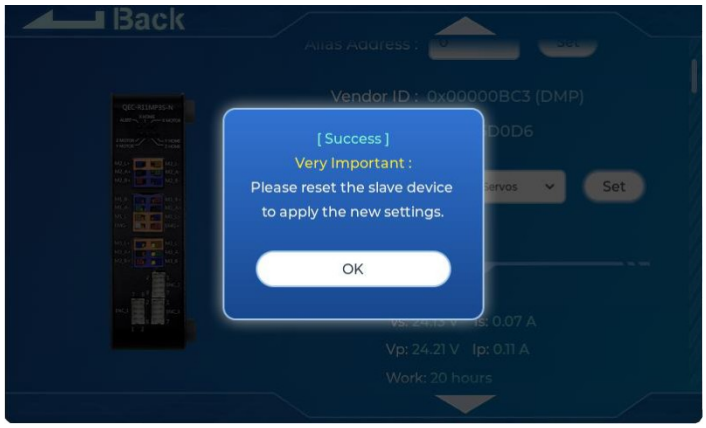
QEC-RXXMP3S のモードの変更

QEC-RXXMP3S のモードを変更する方法は次のとおりです。

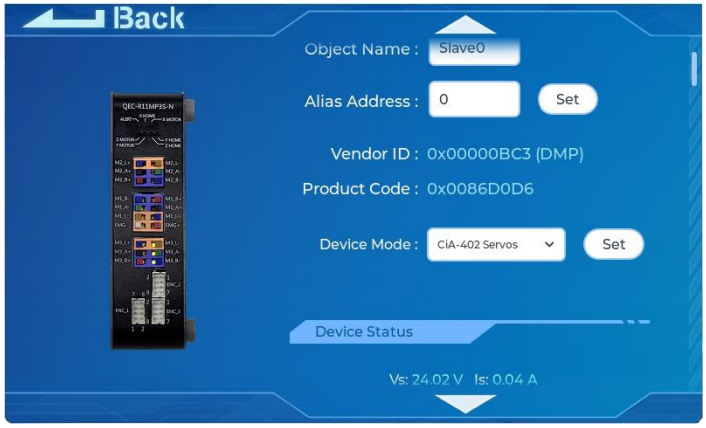
1. 86EVA の「Device Mode」ドロップダウンで、QEC-RXXMP3S の現在の制御モードを変更できます。



2. 選択したら、右側の「Set」ボタンをクリックします。成功を知らせるウィンドウが表示され、**Sub デバイスの電源を切って再起動する必要がある** ことが通知されます。



3. QEC-RXXMP3S の電源を切って再起動した後、QEC M デバイスから 86EVA ツールを再度開き、QEC-RXXMP3S の設定画面に入ると、設定したモーション制御モード（CiA-402 Servos または G-code Machine）を確認できます。



以上の手順で、QEC-RXXMP3S Sub デバイスのモーション制御モードが希望のモードに更新されます。変更を反映させるには再起動が必須のため、モードを切り替えた後は必ず Sub デバイスを再起動してください。

詳細情報やサンプルのご依頼は、info@icop.com.tw までメールでお問い合わせいただくか、最寄りの ICOP 拠点 にお電話いただくか、正規販売代理店 にお問い合わせください。

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。