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・ArduBlock 編）

86Duino Coding IDE 501

EtherCAT ライブラリ

(Version 2.0)

改訂履歴

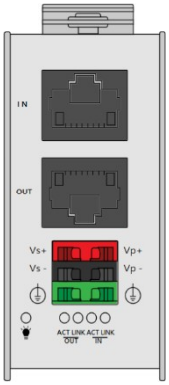
日付	バージョン	内容
2024/2/2	Version1.0	初版リリース。
2024/10/8	Version1.1	開発手順を 2 つの文書に分割。
2025/11/6	Version2.0	HU9S の全機能を 1 つの文書に統合。

はじめに

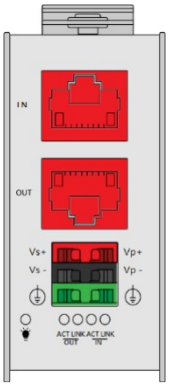
本ガイドでは、EtherCAT M デバイス **QEC-M-01** と **QEC-R11HU9S** シリーズ（RS232/485、キーパッド、LCM、MPG ハンドホイールを備えた EtherCAT Sub デバイス）の使用方法を説明します。

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。



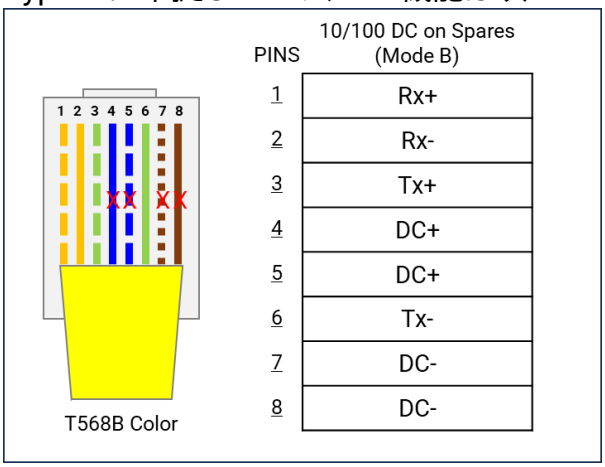
非 PoE タイプ



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。



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

2. 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。

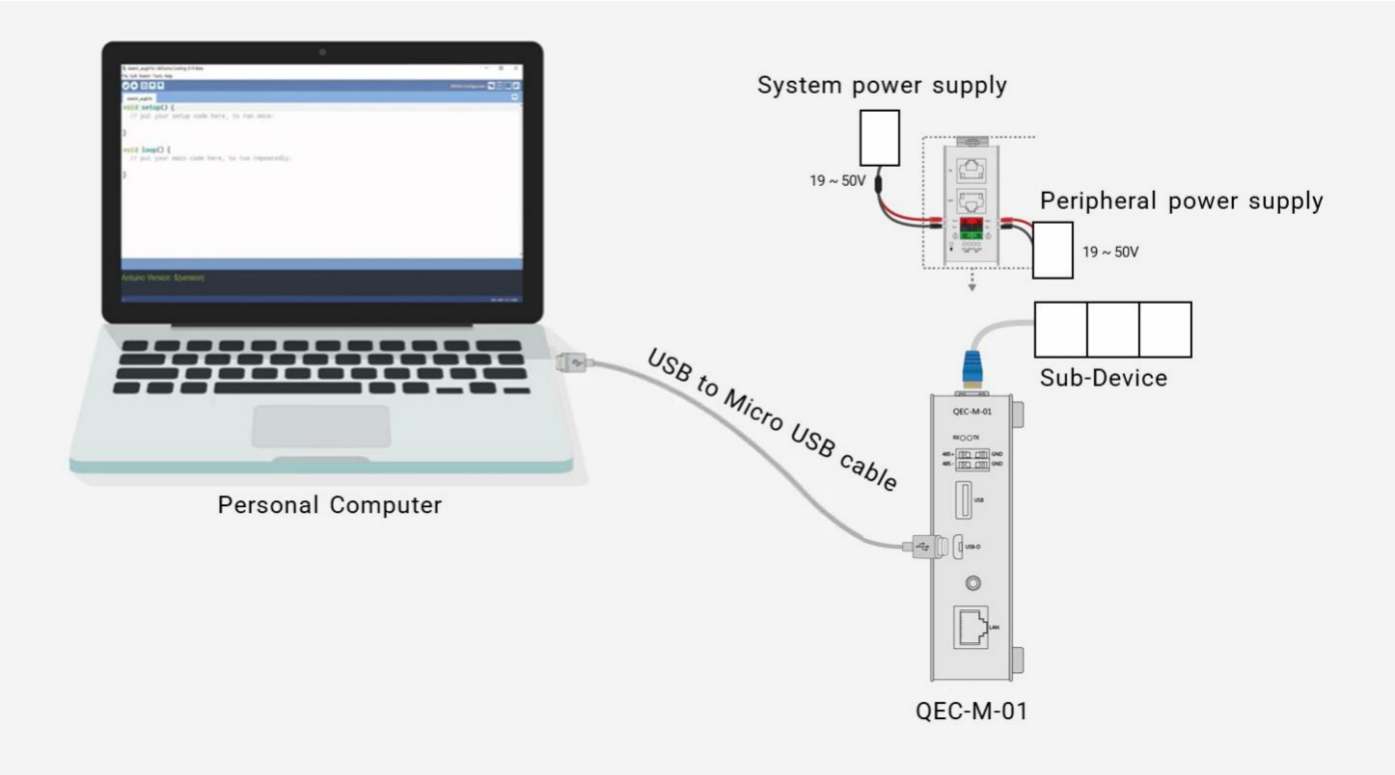
1. QEC-M-01（EtherCAT M デバイス）
2. QEC-R11HU9S シリーズ（RS232/485、キーパッド、LCM、MPG ハンドホイールを備えた EtherCAT Sub デバイス）
3. 24VDC 電源、EU タイプ端子ケーブル、LAN ケーブル



1.1 QEC-M-01

QEC-M-01 は EtherCAT M デバイスです。

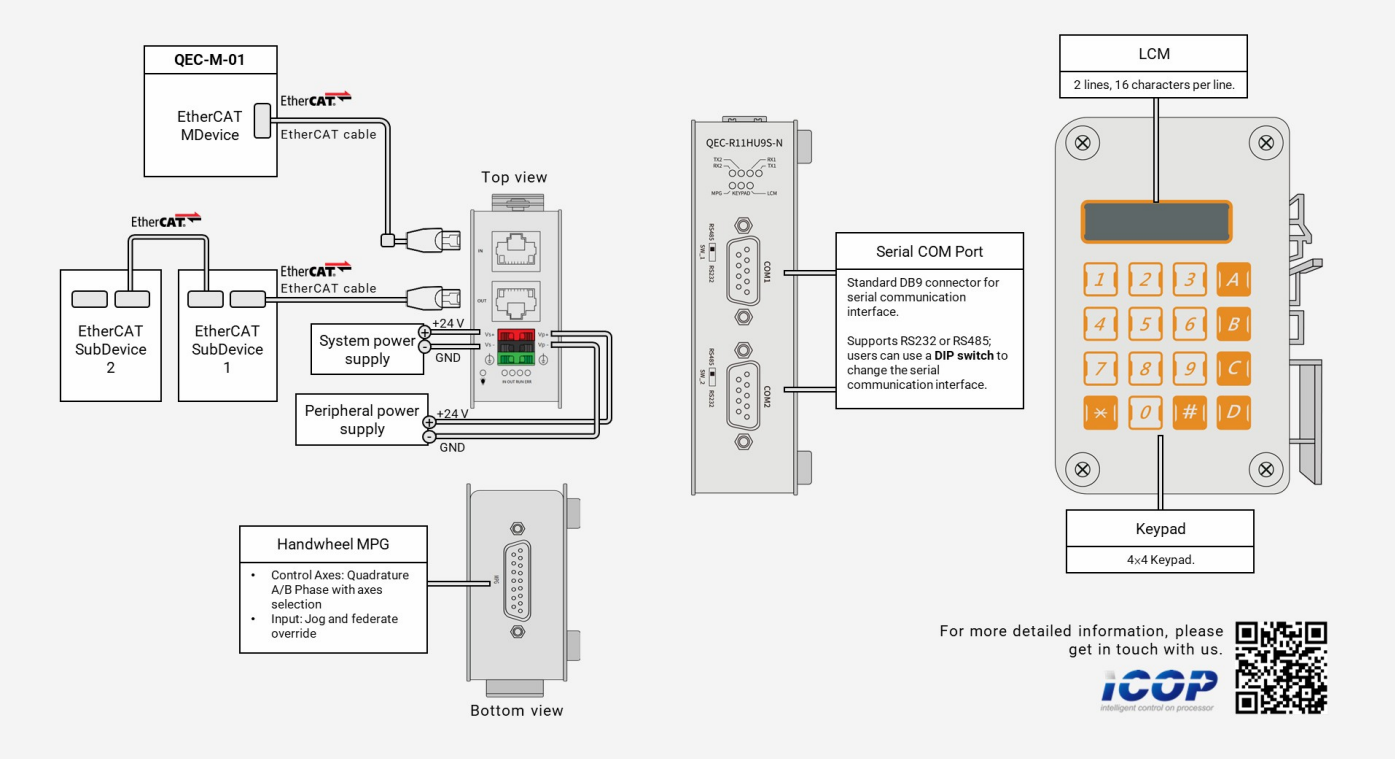
1. 電源：EU 端子を介して Vs+/Vs- および Vp+/Vp- に 24V 電源を接続します。
2. EtherCAT 接続：EtherCAT Out ポート（上側）から RJ45 ケーブルで EtherCAT Sub デバイスの EtherCAT In ポートに接続します。



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。

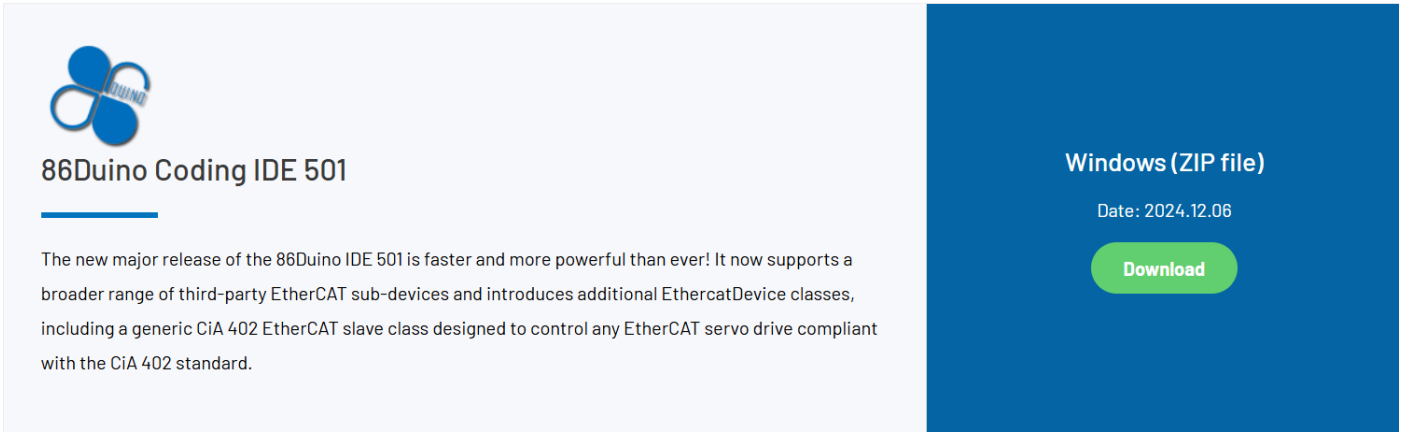


接続は機能ごとに次のように分かれています。

- 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。

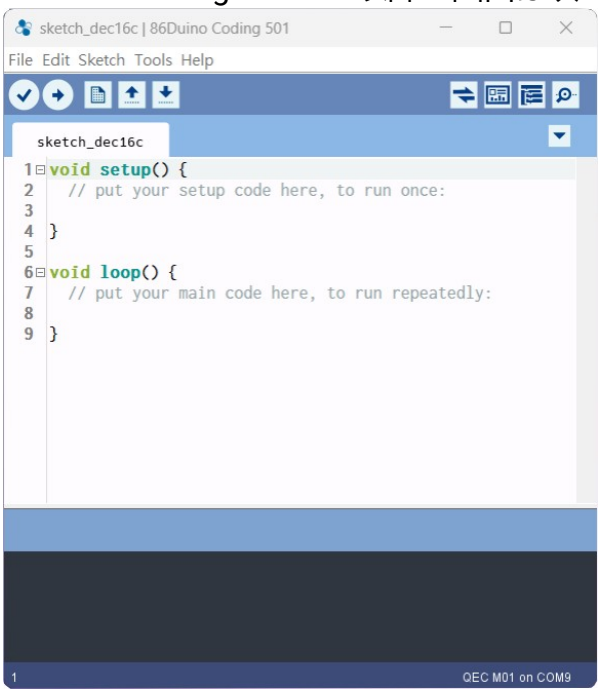


ダウンロードした zip ファイルを解凍してください。追加のソフトウェアをインストールする必要はなく、86duino.exe をダブルクリックするだけで IDE が起動します。



※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。

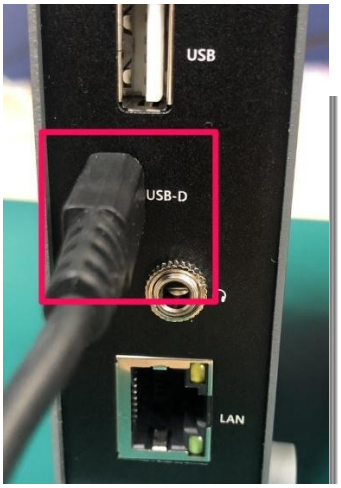
86Duino Coding IDE 501 以降の画面は次のとおりです。



3. PC との接続と環境設定

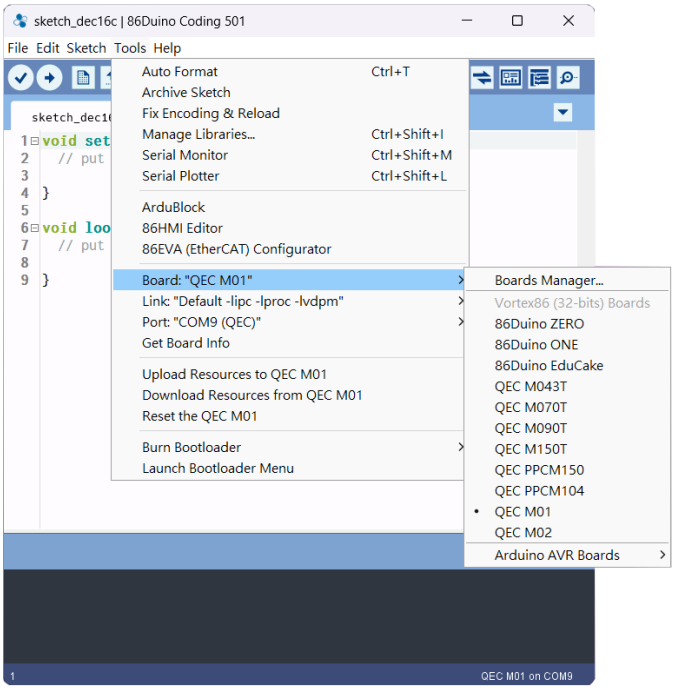
次の手順で環境を設定します。

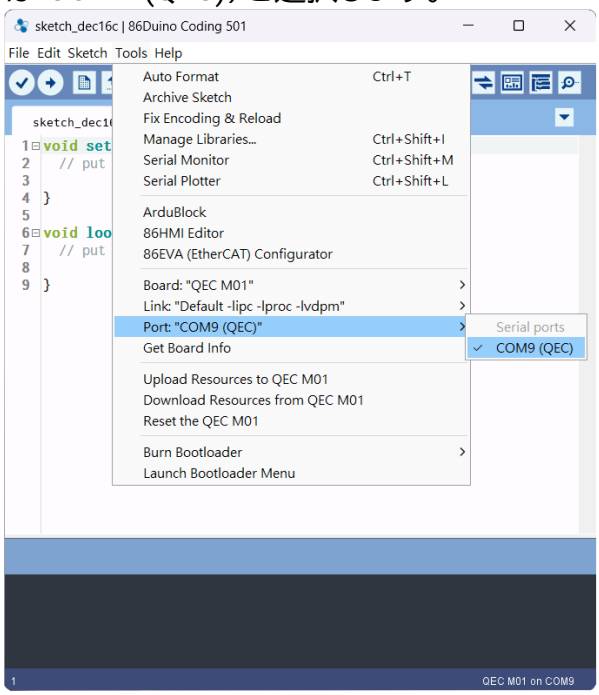
1. Micro USB－USB ケーブルで、QEC-M-01 を PC（86Duino IDE をインストール済み）に接続します。
2. QEC の電源を入れます。
3. PC で「**Device Manager（デバイス マネージャー）**」（Win+X キーを押して表示されるメニューから選択）を開き、「**Ports (COM & LPT)（ポート（COM と LPT））**」を展開します。「**Prolific PL2303GC USB Serial COM Port (COMx)**」が検出されていることを確認してください。表示されない場合は、必要なドライバーをインストールする必要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

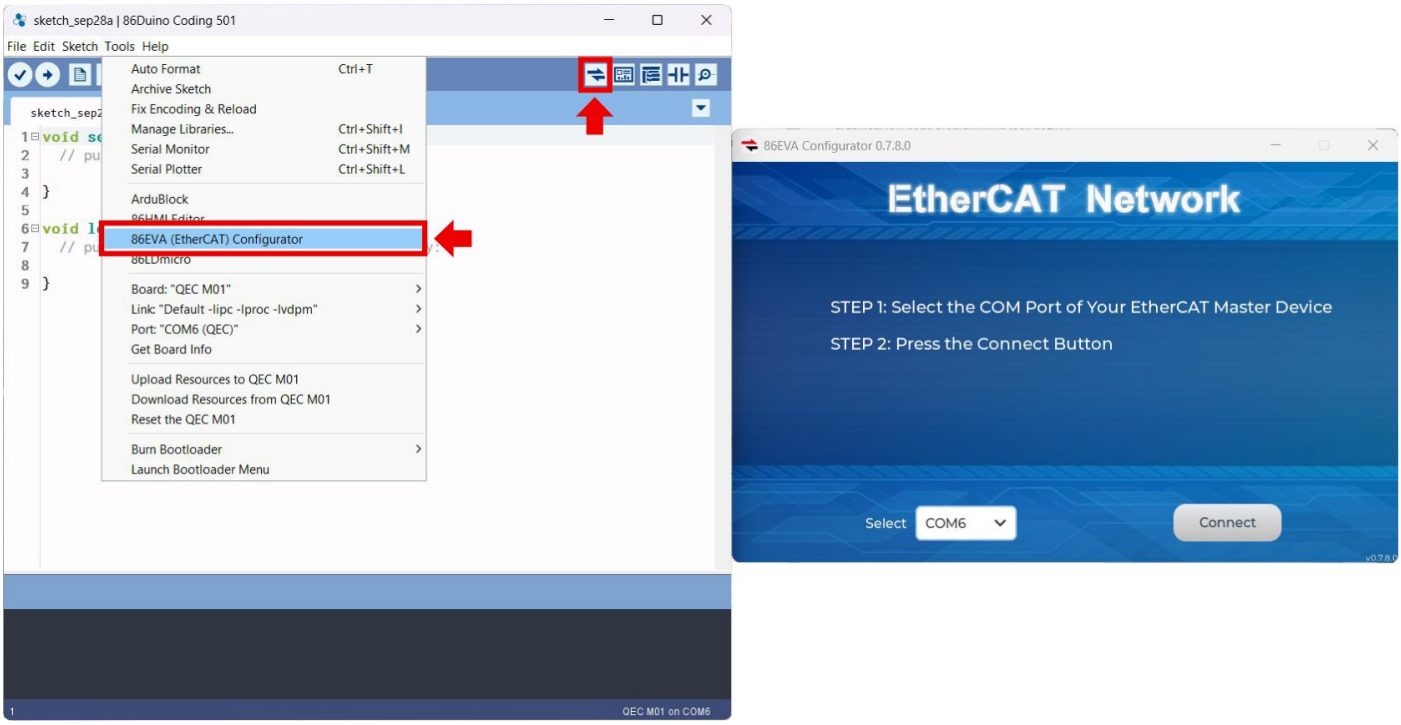
ソフトウェアツールの説明：

- **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。
- **ArduBlock：**
  プログラミングと I/O 制御のためのグラフィカルインターフェースです。上海在住のクリエイター David Li 氏が開発した Arduino IDE 向けのサードパーティ製ソフトウェアで、IDE に組み込んで動作させる必要があります。ArduBlock はグラフィカルなブロックをコードに変換し、最終的にメインプログラムを 86Duino Coding IDE に生成して、コンパイルと書き込みを行います。

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 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。

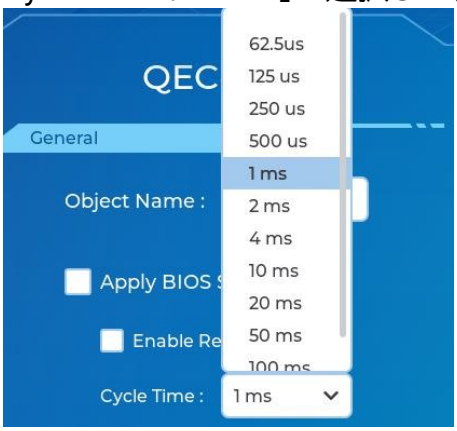
ステップ 2.1：QEC-M-01

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。



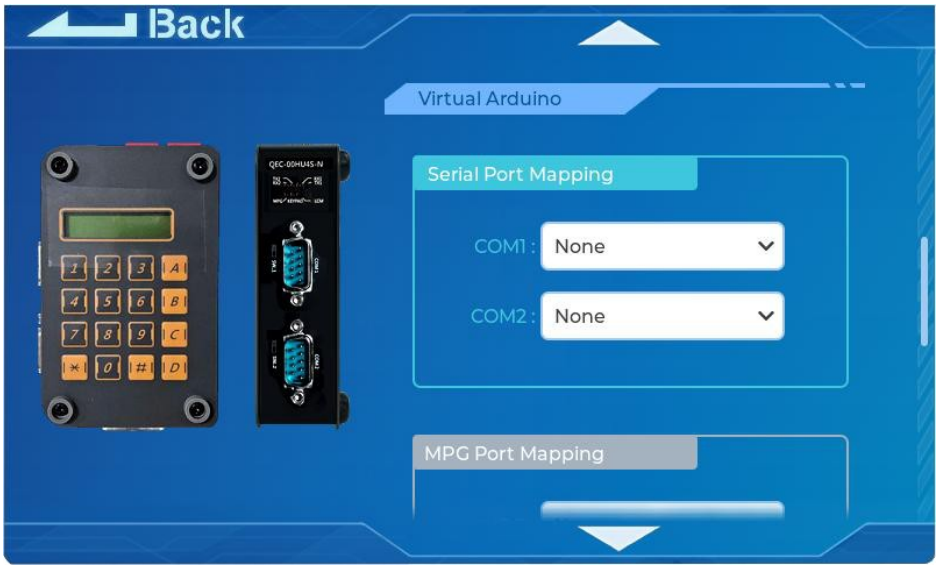
ステップ 2.2：QEC-R11HU9S

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。



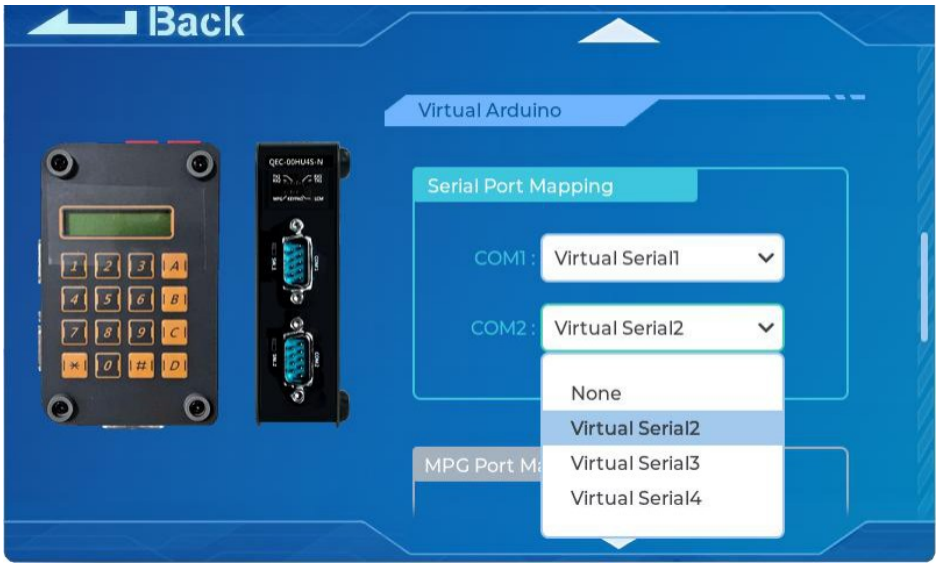
この画面には、Object Name、Alias Address、Vendor ID、Product Code、Virtual Arduino Mapping、Virtual Servo Configuration の各パラメーターが表示されます。

下にスクロールして「**Virtual Arduino**」エリアに進みます。

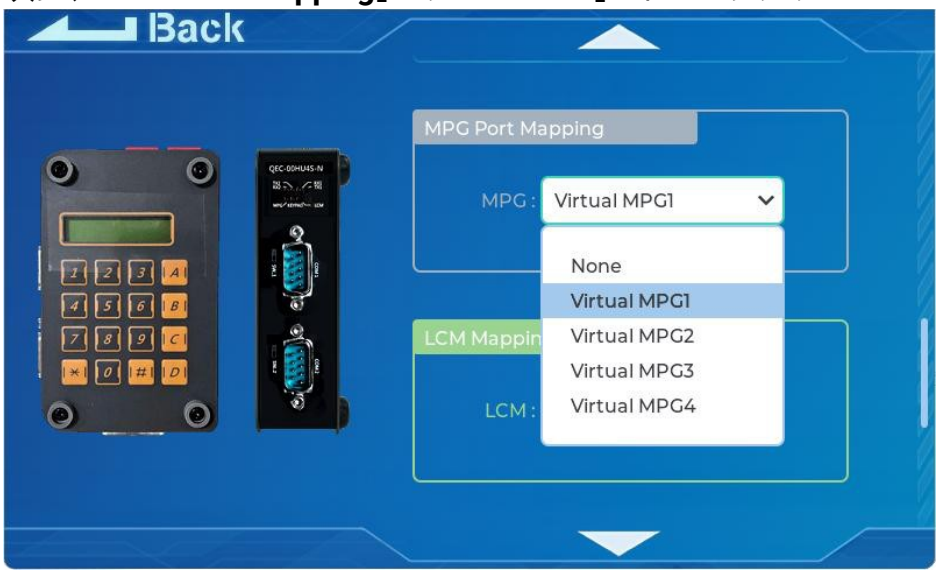


QEC-R11HU9S の Virtual Arduino には、**Serial Port Mapping**、**MPG Port Mapping**、**LCM Mapping**、**Keypad Mapping**、**Buzzer Pin Mapping** の 5 つのエリアがあります。

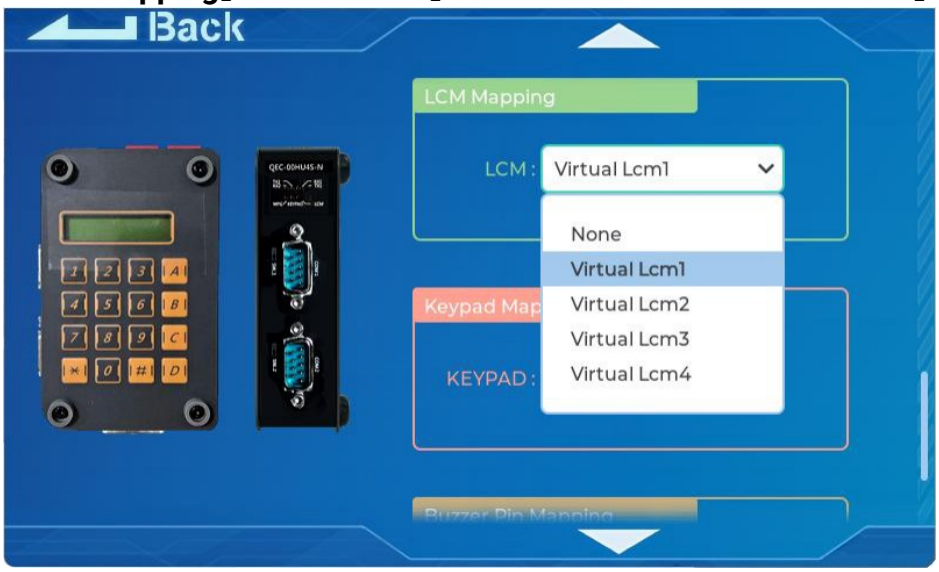
まず、「**Serial Port Mapping**」エリアの「**COM1**」と「**COM2**」のドロップダウンで「**Virtual Serial1/2**」を選択します。



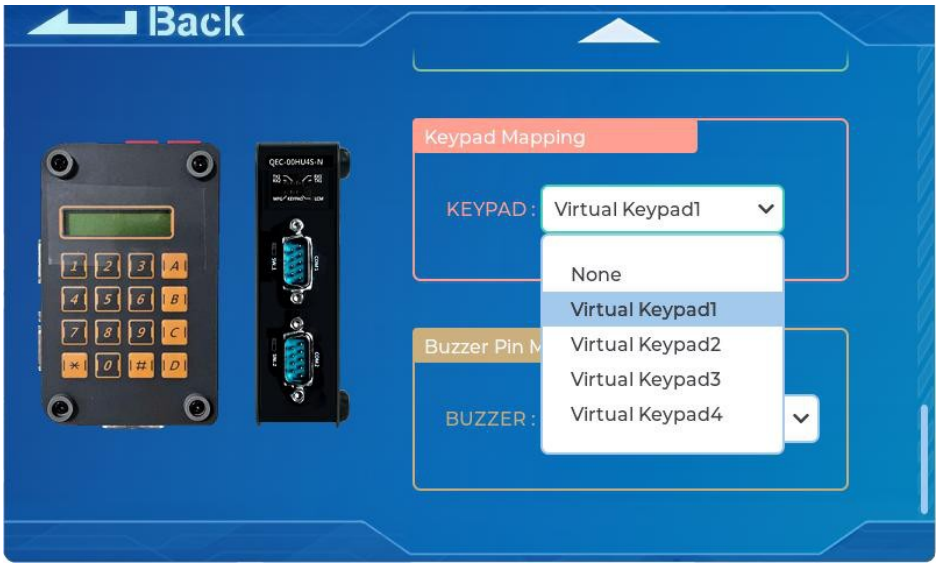
次に、「**MPG Port Mapping**」エリアの「**MPG**」のドロップダウンで「**Virtual MPG1**」を選択します。



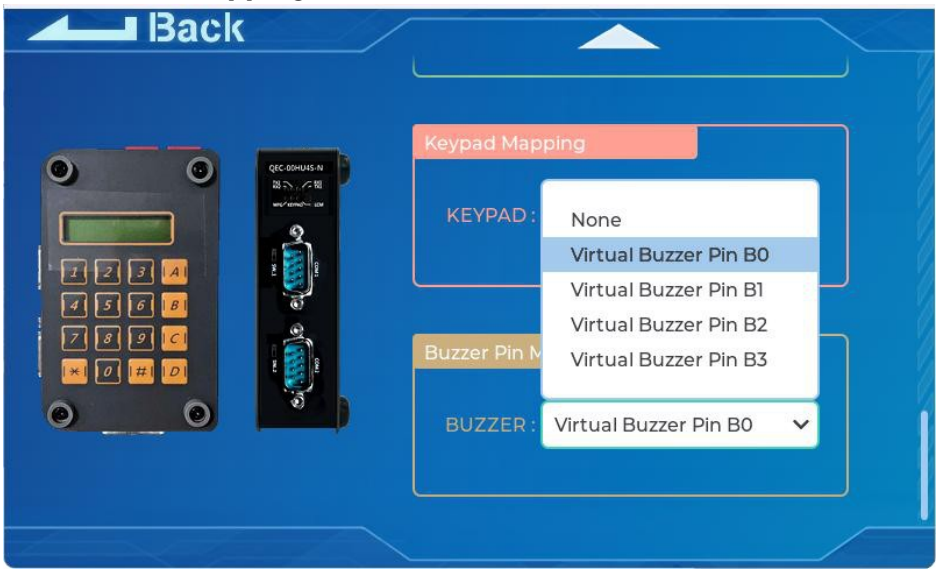
「**LCM Mapping**」エリアの「**Lcm**」のドロップダウンで「**Virtual Lcm1**」を選択します。



「**Keypad Mapping**」エリアの「**KEYPAD**」のドロップダウンで「**Virtual Keypad1**」を選択します。



「**Buzzer Pin Mapping**」エリアの「**BUZZER**」のドロップダウンで「**Virtual Buzzer Pin B0**」を選択します。



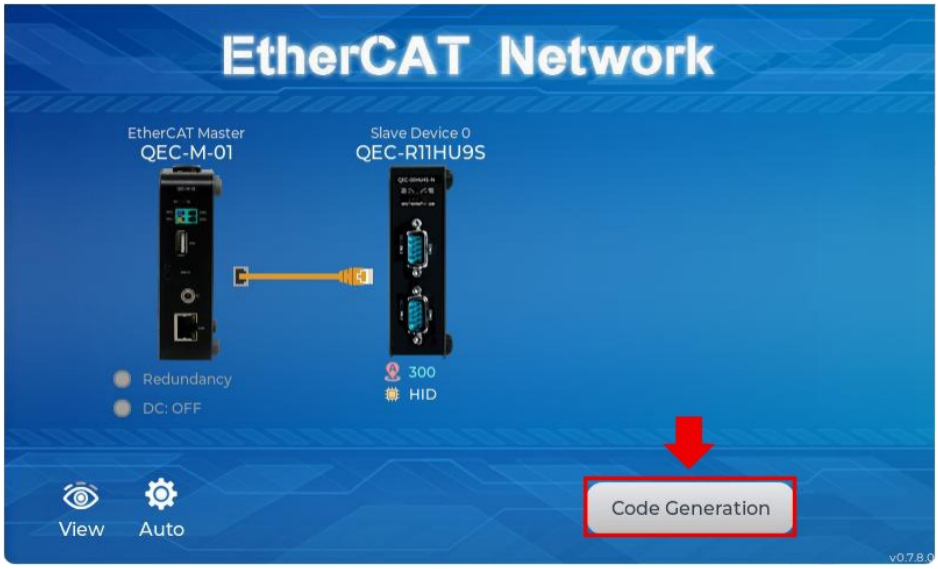
これらの設定とマッピングは、すべて ArduBlock ツールで使用するためのものです。

次に、左上の「**Back（戻る）**」をクリックして元の画面に戻ります。

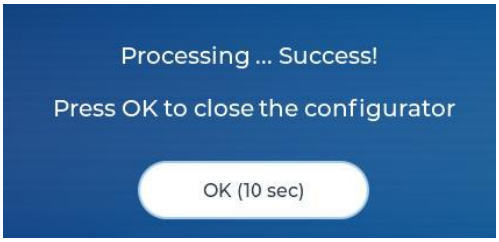


ステップ 3：コードを生成する

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。

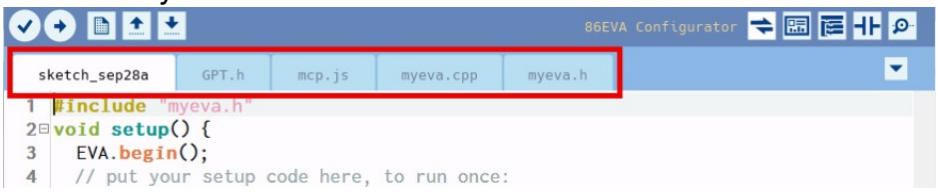


生成が完了したら「**OK**」ボタンをダブルクリックして 86EVA を閉じます。操作しない場合も 10 秒後に自動的に閉じます。



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：ArduBlock を起動して設定する

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。

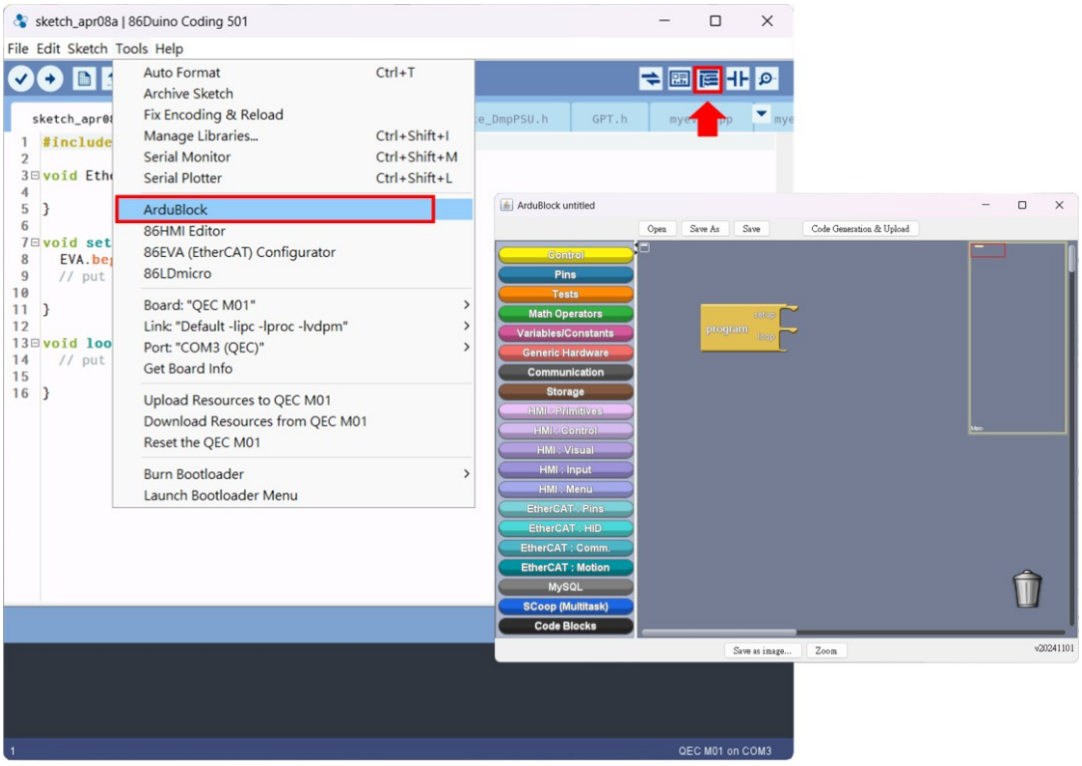
86EVA が生成するプログラムコードは、デフォルトで次のように設定されています。

- 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。

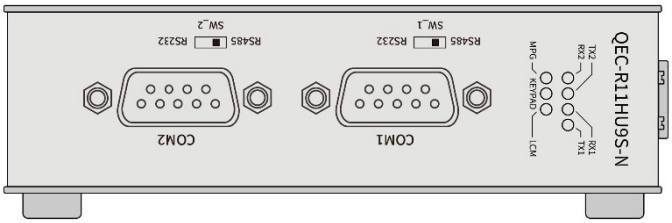
86EVA で Virtual Arduino のピンを設定したら、ArduBlock を起動します。



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）

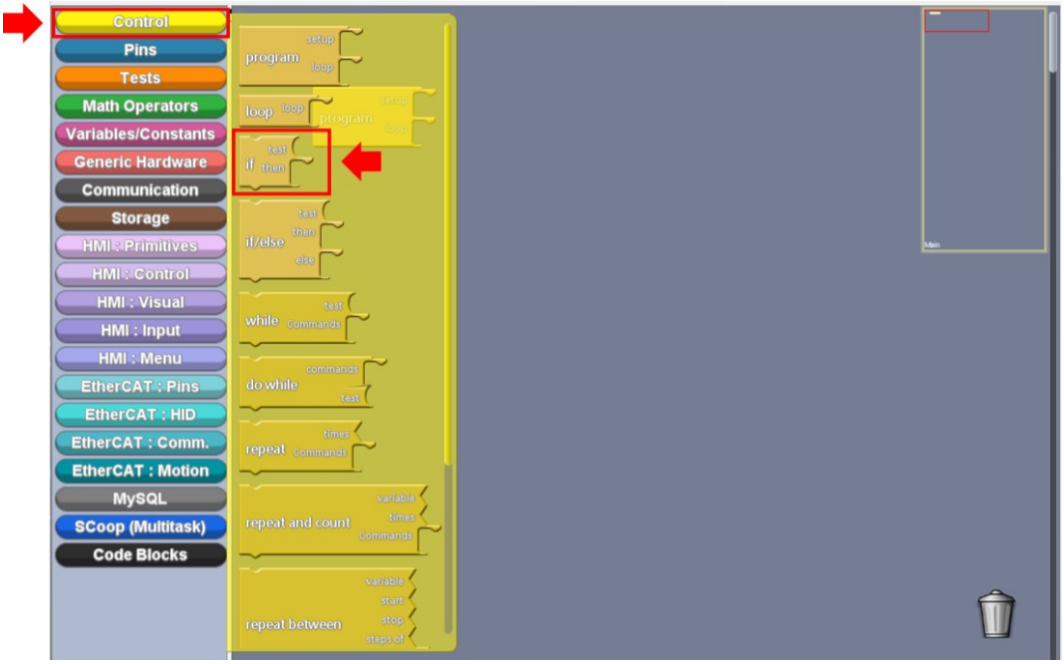
このセクションでは、86Duino IDE のシリアルモニタからデータを読み取り、COM1（RS-232）から COM2（RS-232）へ転送します。COM2 がデータを受信したら、それをシリアルモニタに表示します。



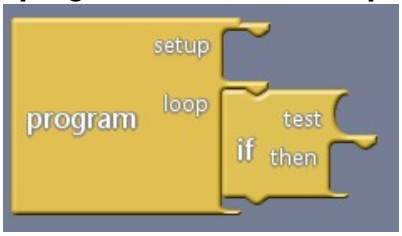
配線

- 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。

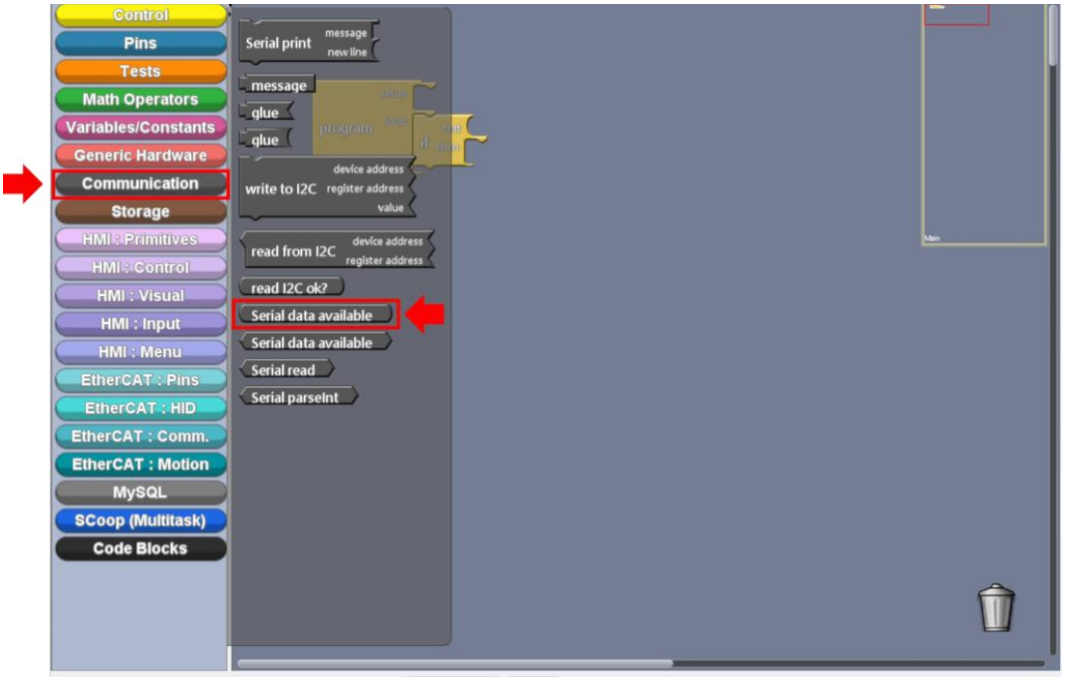
まず、「**Control**」クラスの「**if**」ブロックをプログラムのメインループに配置し、シリアルモニタが使用可能かどうかを判定します。



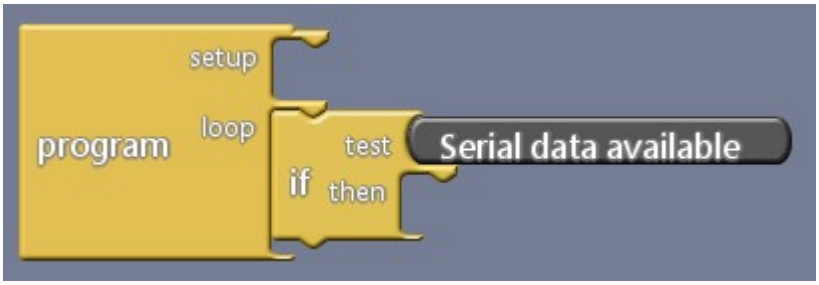
「**program**」ブロックの「**loop**」エリアに配置します。



次に、「**Communication**」クラスの「**Serial data available**」ブロックを使用します。



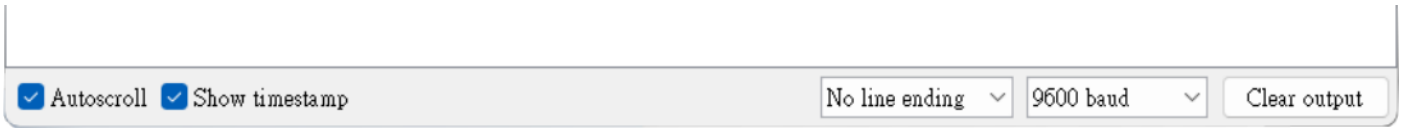
「**if**」ブロックの「**test**」エリアに配置します。



「**if**」は「**Serial data available**」を判定し、true の場合にプログラムを実行します。

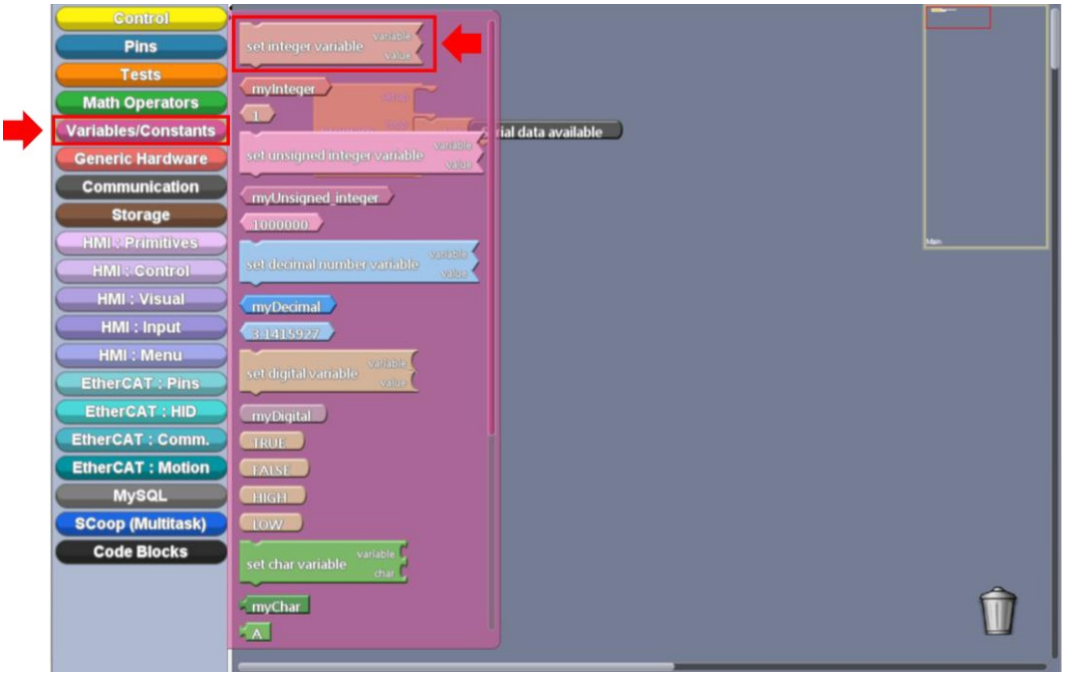
※注意：

シリアルモニタのボーレートのデフォルトは 9600 です。



「**if**」ブロックの then エリアで、HID の RS232 通信の処理を行います。

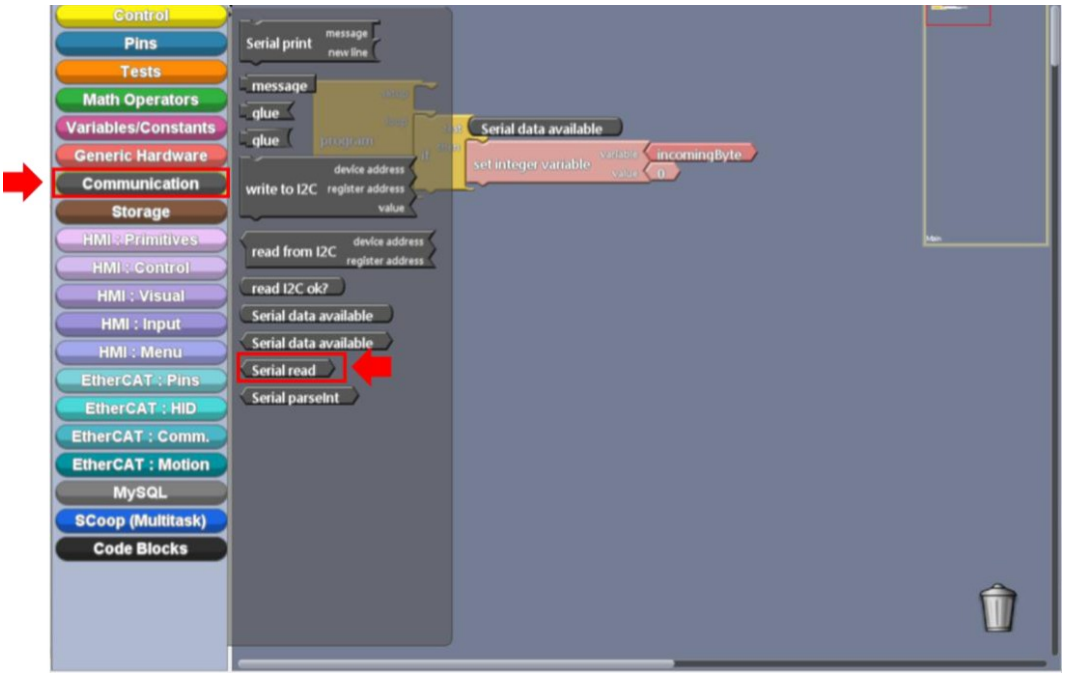
「**Variables/Constants**」クラスの「**set integer variable**」ブロックを使用して、シリアルモニタから受け取る入力値の変数を作成します。



「**if**」ブロックの「**then**」エリアに配置し、変数名を「**incomingByte**」に変更します。



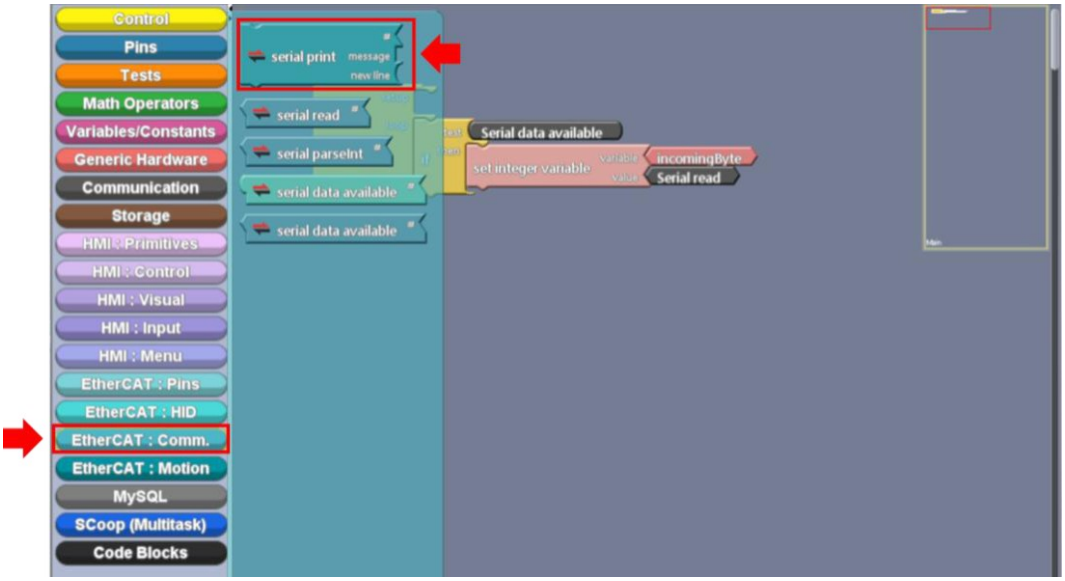
「**Communication**」クラスの「**Serial read**」ブロックを、「**set integer variable**」ブロックの値エリアに配置します。



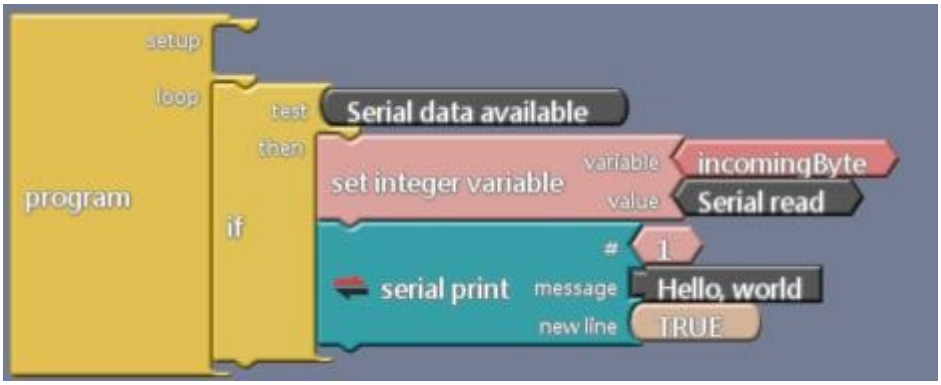
下図のようになります。



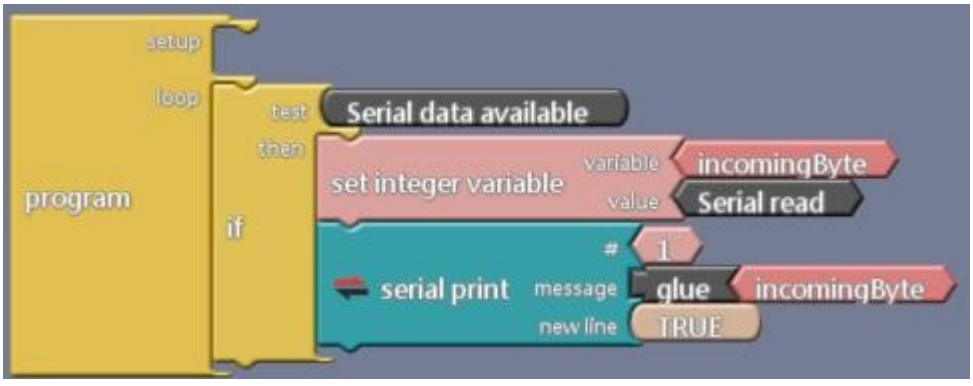
次に、「**EtherCAT : Comm.**」クラスの「**serial print**」ブロックを使用して、シリアルモニタからの入力データを「**Virtual Serial Port1**」経由で送信します（HU9S の COM1 には「**Virtual Serial Port1**」を選択しています）。



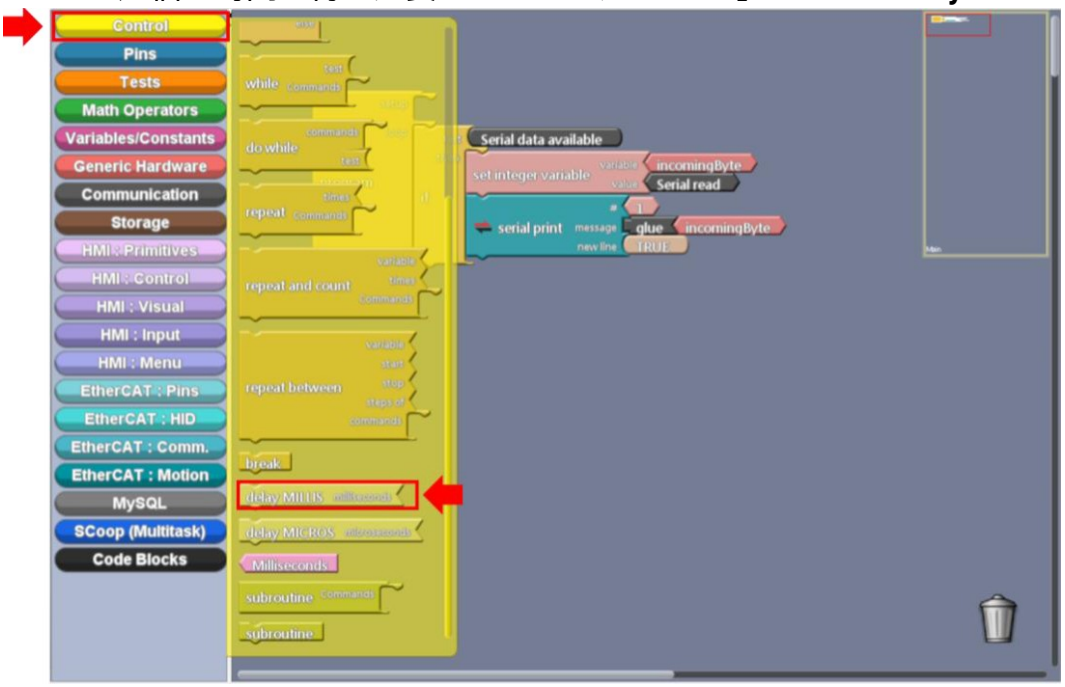
下図のようになります。



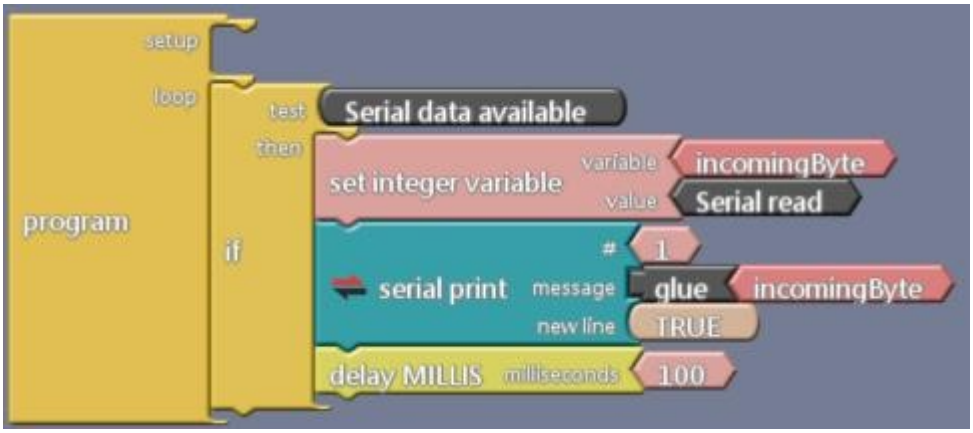
メッセージの内容を「**Hello, world**」から変数「**incomingByte**」に変更する必要があります。
「**Communication**」クラスの「**glue**」ブロックを使用して、「**incomingByte**」を「**serial print**」のメッセージエリアにつなげます。



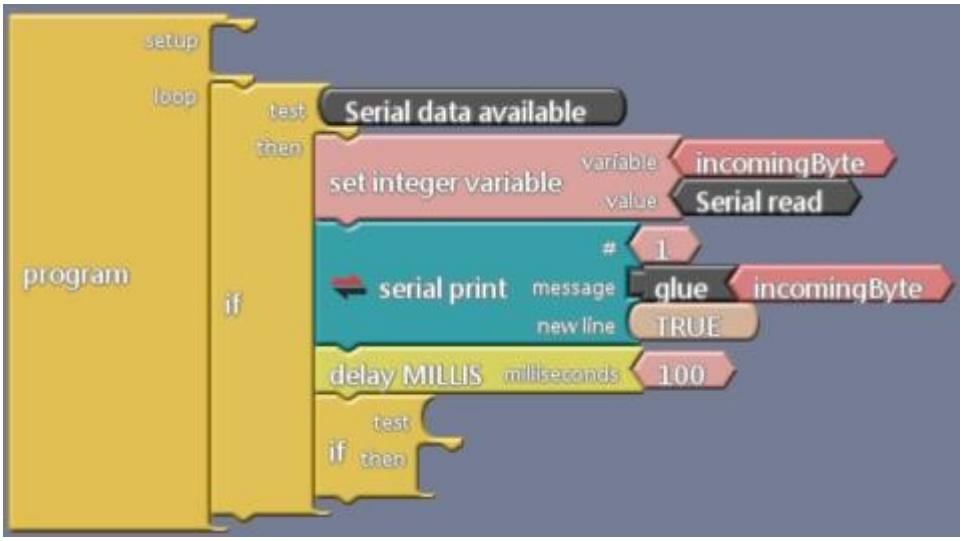
シリアル通信の時間を待つ必要があるため、「**Control**」クラスの「**delay MILLIS**」ブロックを配置します。



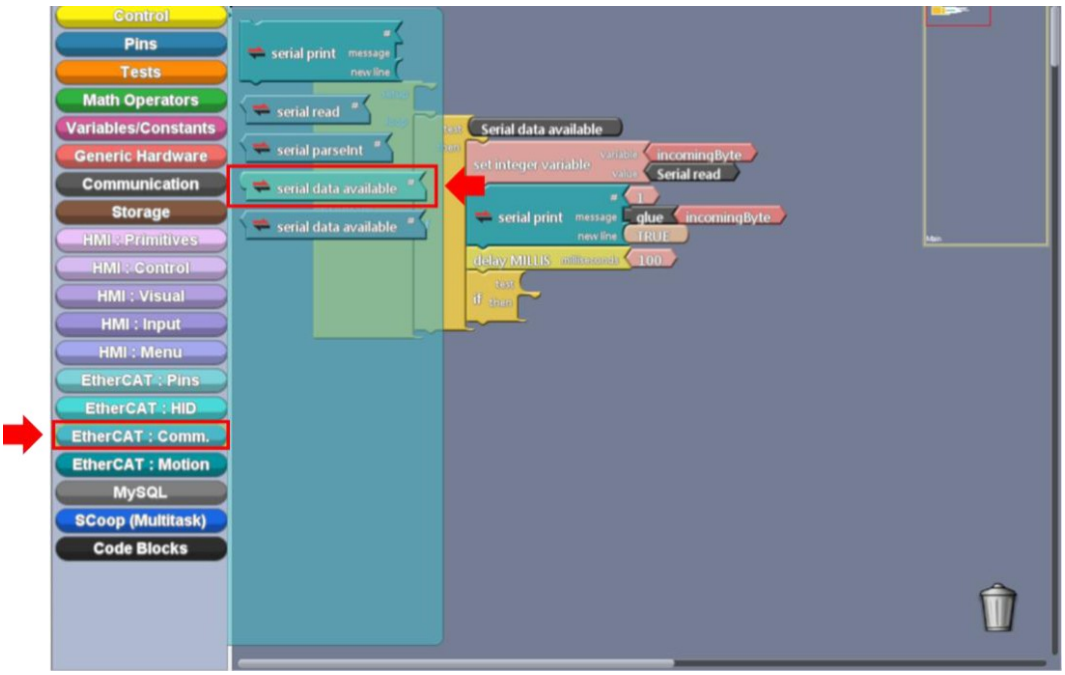
この例では 100 ミリ秒待機します。



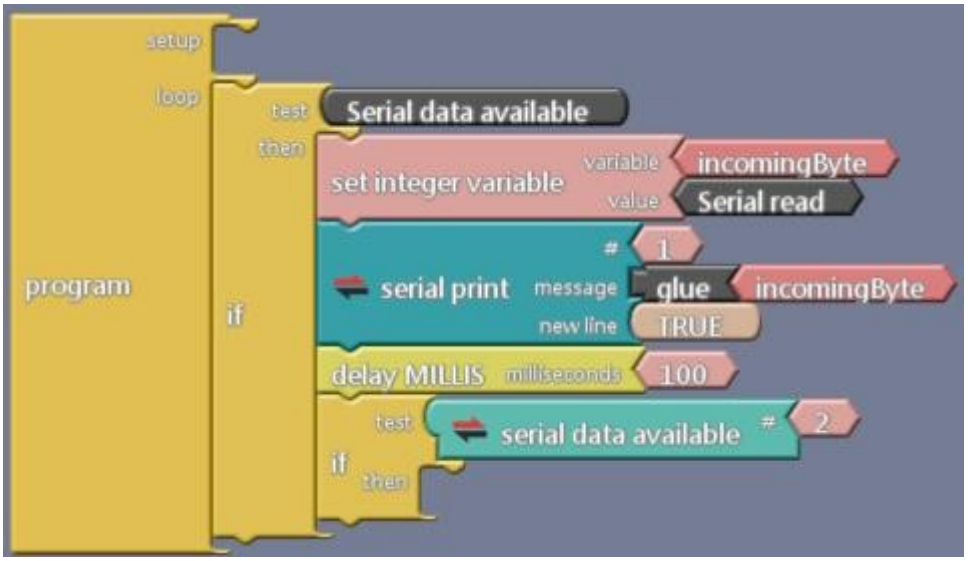
次に、「**Virtual Serial Port2**」（HU9S の COM2 には「**Virtual Serial Port2**」を選択しています）が「**Virtual Serial Port1**」からデータを受信したかどうかを判定します。そのため、「**delay MILLIS**」ブロックの後に「**if**」ブロックを作成します。



「**if**」ブロックの test エリアには、「**EtherCAT : Comm.**」クラスの「**serial data available**」ブロックを使用します。

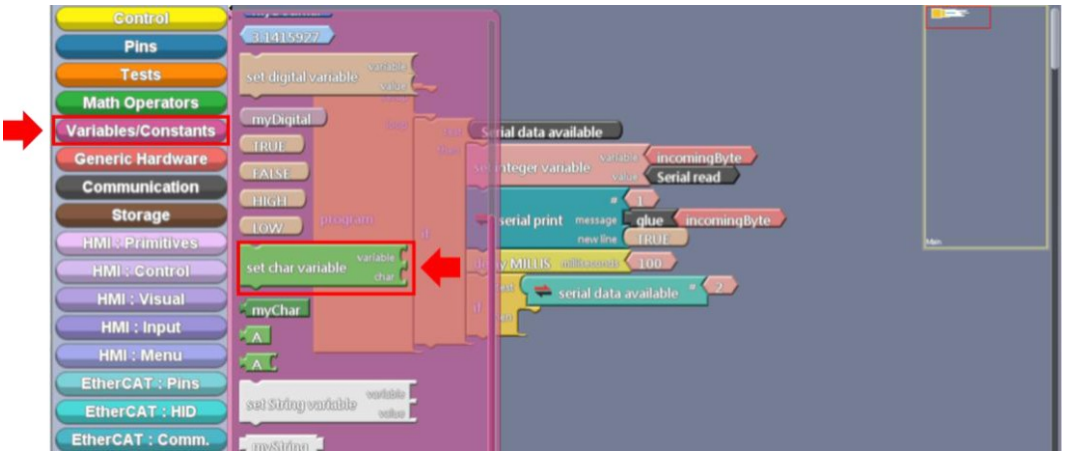


「#」の番号を 2 に設定し、「**serial data available**」が「**Virtual Serial Port 2**」を読み取るようにします。



「**if**」は「**Virtual Serial Port2**」が使用可能かどうかを判定し、true の場合にプログラムを実行します。

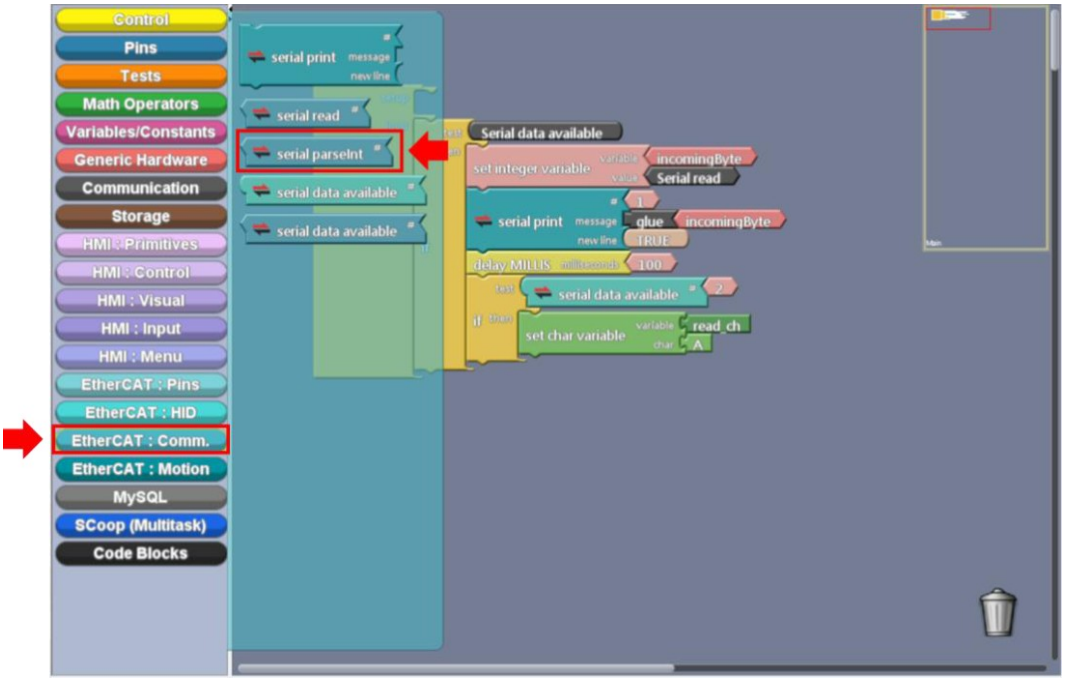
「**if**」ブロックの「**then**」エリアでは、「**Virtual Serial Port2**」が受信したデータを読み取り、「**read_ch**」という char 型変数に格納します。
まず、「**Variables/Constants**」クラスの「**set char variable**」ブロックを使用し、変数名を「**read_ch**」に設定します。



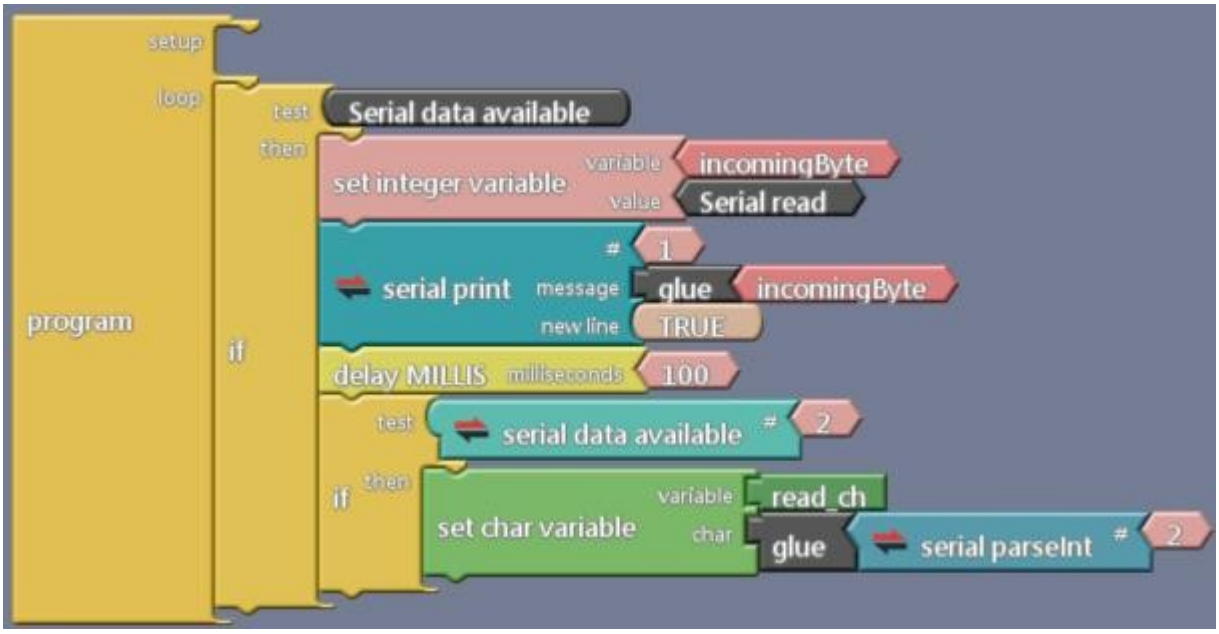
下図のようになります。



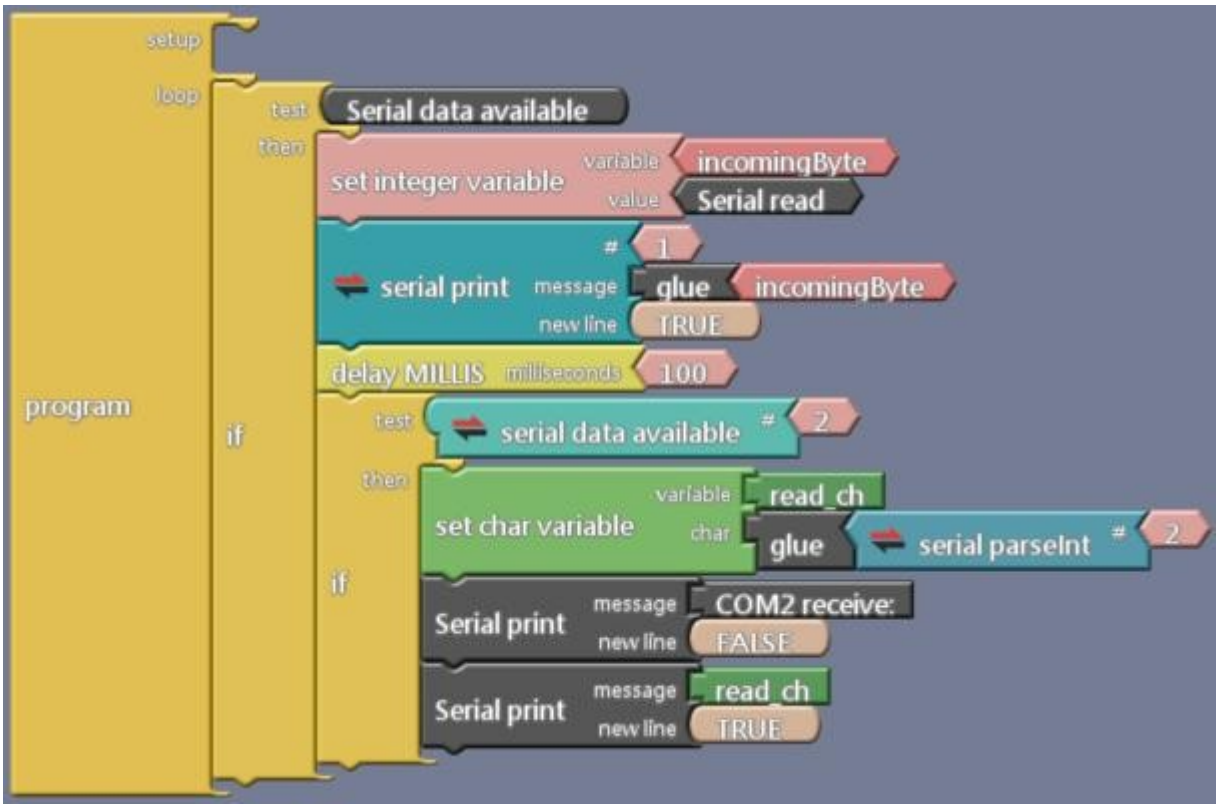
「**Communication**」クラスの「**glue**」ブロックを使用して、「**EtherCAT : Comm**」クラスの「**serial parseInt**」ブロックで取得した「**Virtual Serial Port2**」の受信データを、「**set char variable**」の char エリアにつなげます。



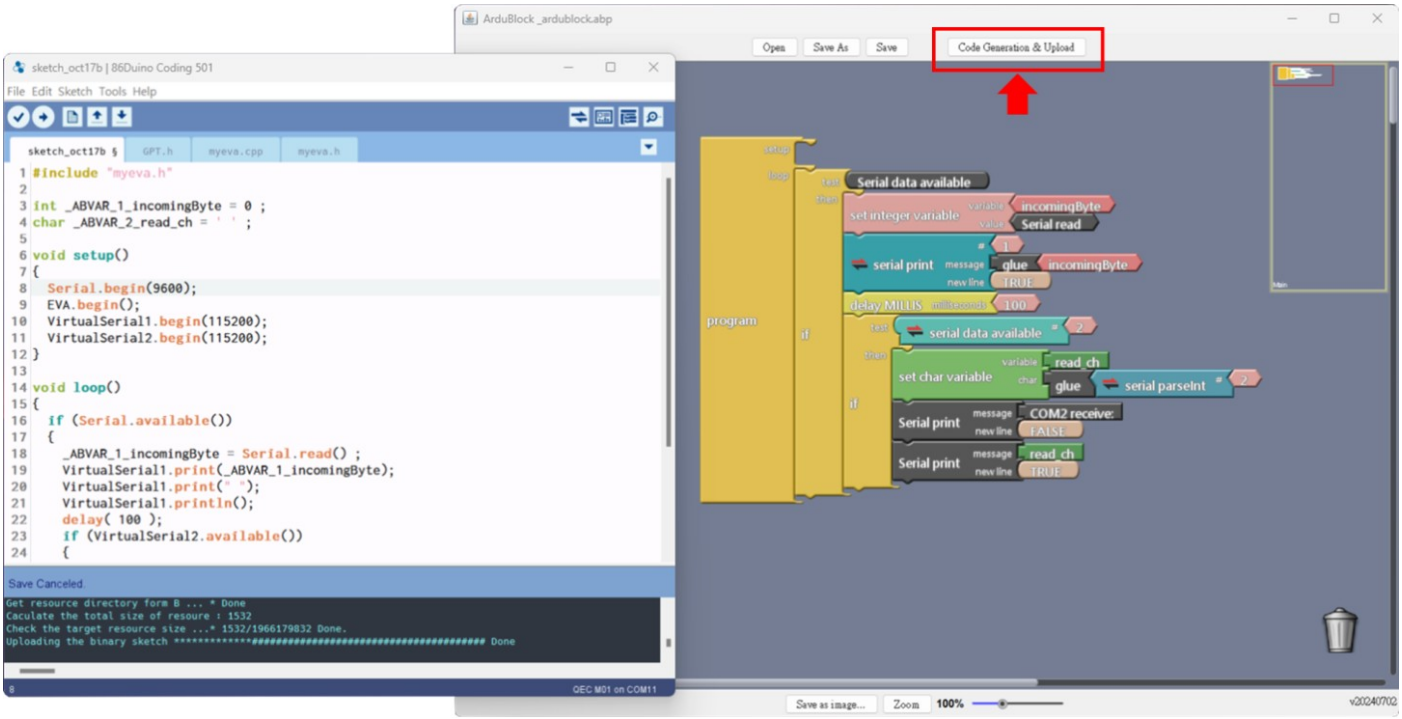
下図のようになります。



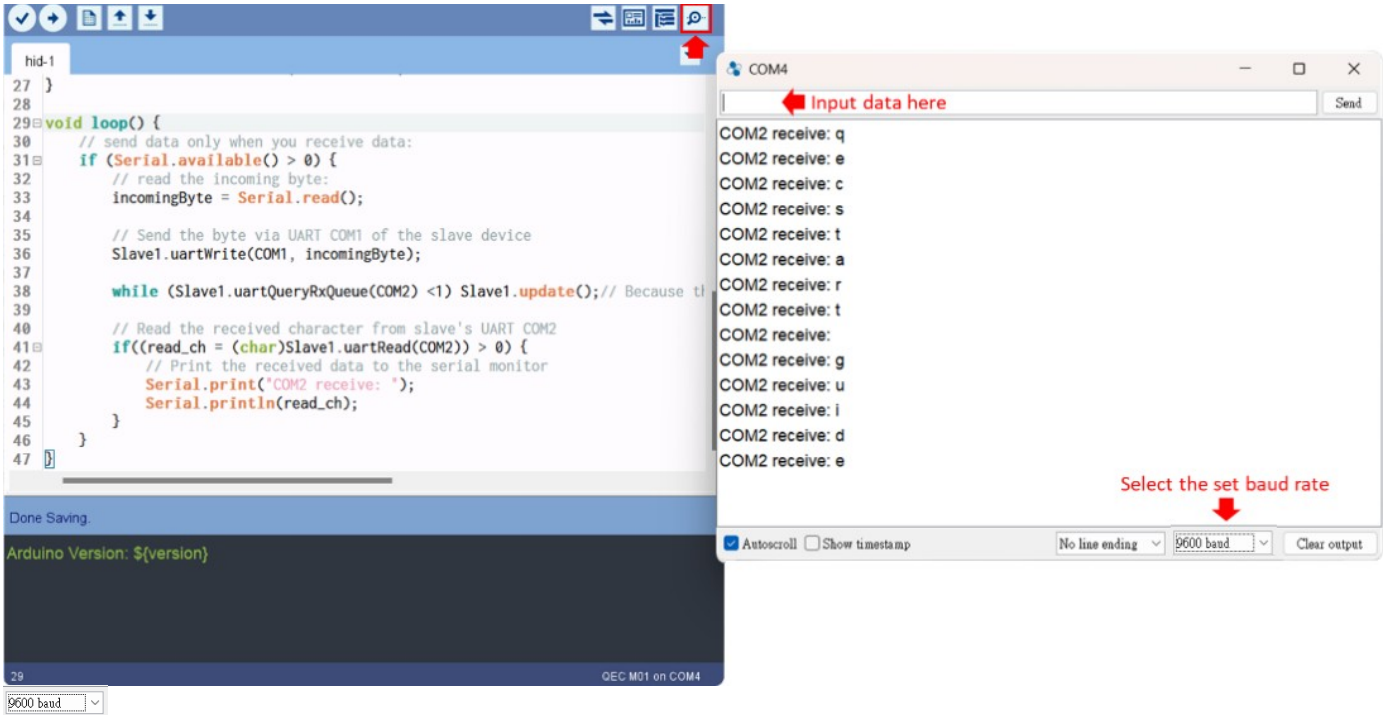
「**Serial print**」ブロックを使用して、char 型変数「**read_ch**」（「**Virtual Serial Port2**」から受信したデータ）をシリアルモニタに表示します。



完成したら「**Code Generation & Upload**」ボタンをクリックすると、ArduBlock がプログラムコードを自動生成し、QEC-M-01 に書き込みます。

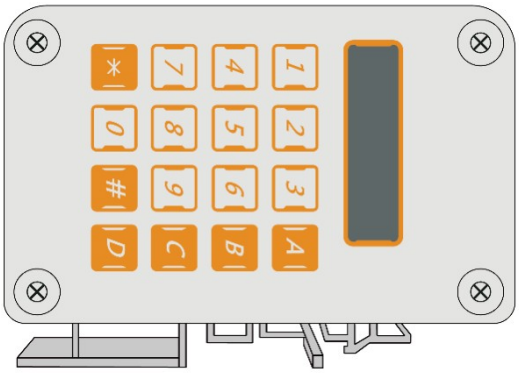


書き込み後、86Duino IDE のシリアルモニタに数字や文字を入力すると、すべてのデータが COM1 から COM2 に転送されます。COM2 がデータを受信すると、下図のようにシリアルモニタに表示されます。



ステップ 4.2：キーパッド + LCM + ブザー

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。

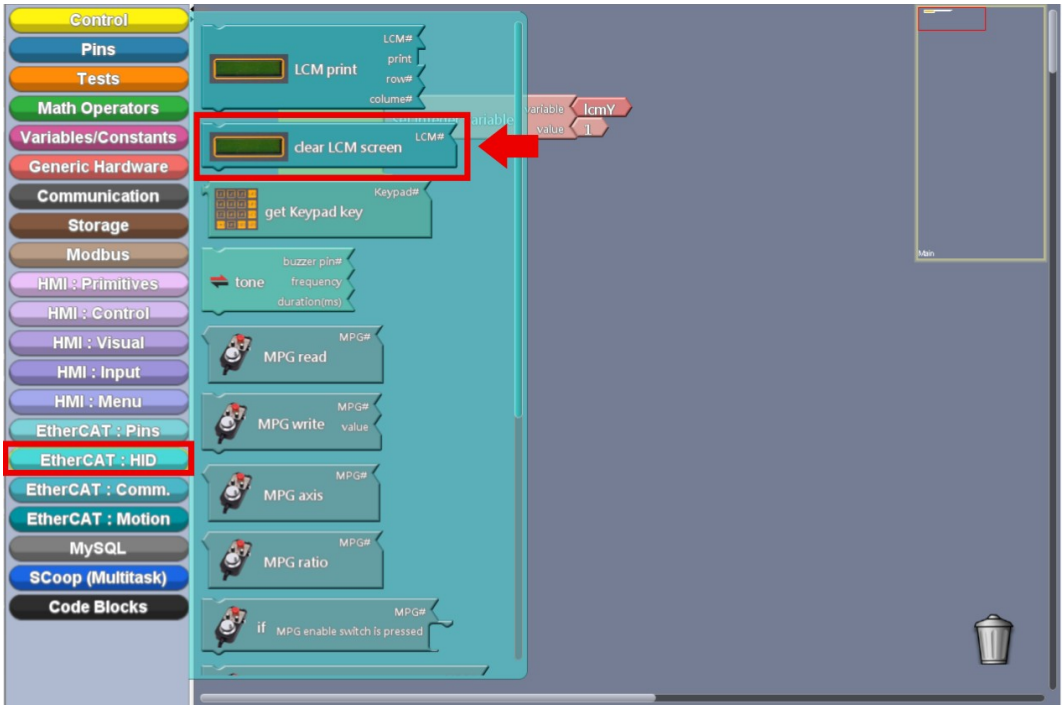
まず、「**program**」ブロックの「**setup**」エリアで、「**Variables/Constants**」クラスの「**set integer variable**」ブロックを使用して、行番号を記録する変数を作成します。



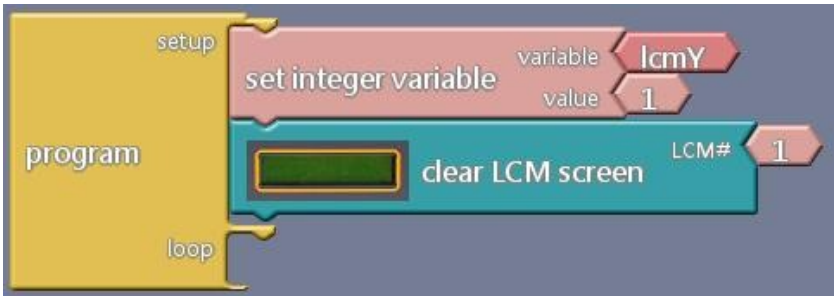
変数名を「**lcmY**」に、値を 1 に変更します。



「**EtherCAT : HID**」クラスの「**clear LCM screen**」ブロックを使用して、「**Virtual Lcm1**」経由で LCM を初期化します（HU9S の LCM には「**Virtual Lcm1**」を選択しています）。

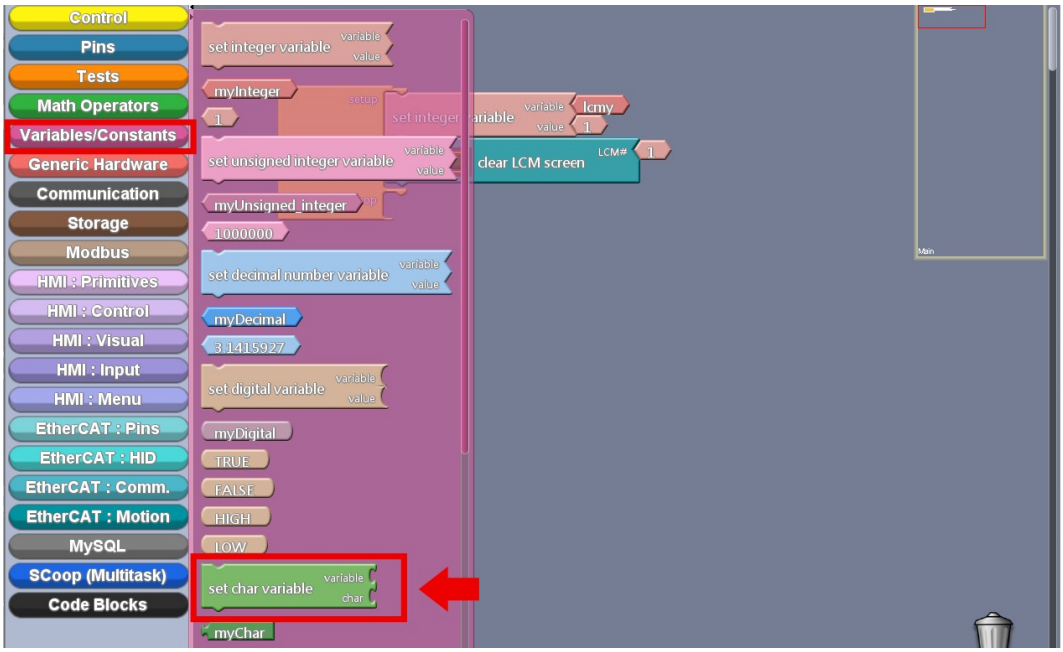


下図のようになります。

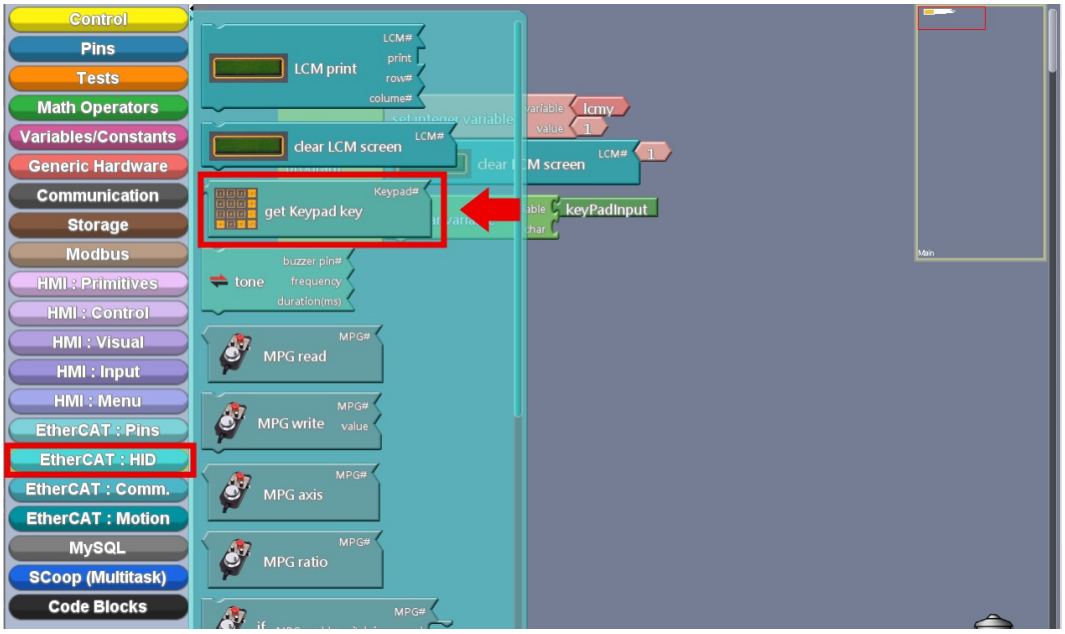


次に、「**program**」ブロックの「**loop**」エリアに進みます。

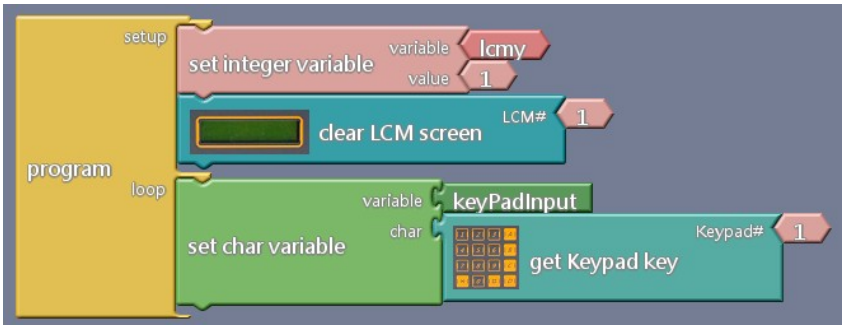
「**Variables/Constants**」クラスの「**set char variable**」ブロックを使用して、キーパッドの入力値を記録する変数を作成します。



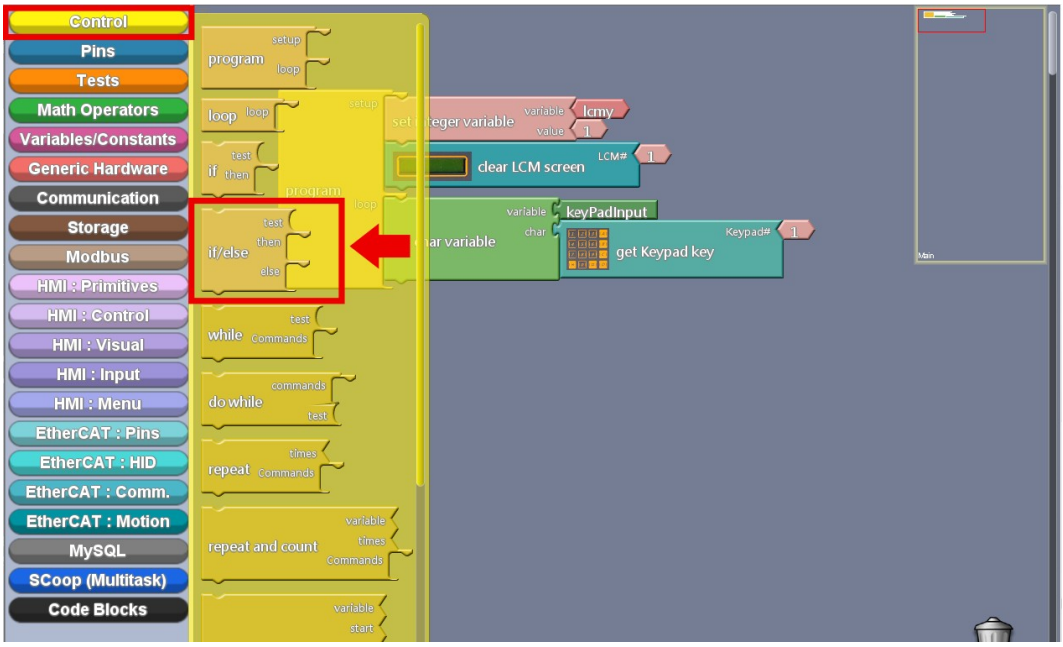
変数名を「**keyPadInput**」に変更します。そして「**keyPadInput**」の「**char**」エリアに、「**EtherCAT : HID**」クラスの「**get Keypad key**」ブロックを配置します（HU9S のキーパッドには「**Virtual Keypad1**」を選択しています）。



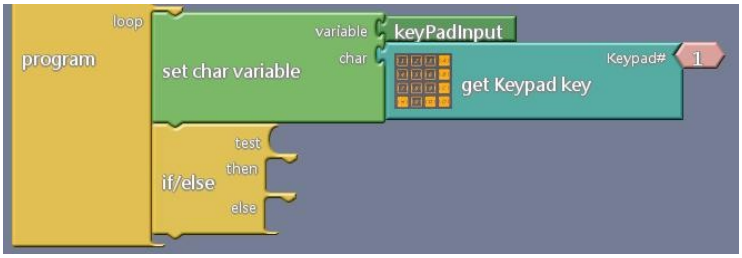
下図のようになります。



「**Control**」クラスの「**if/else**」ブロックをプログラムのメインループに配置し、キーパッドの入力値を判定します。

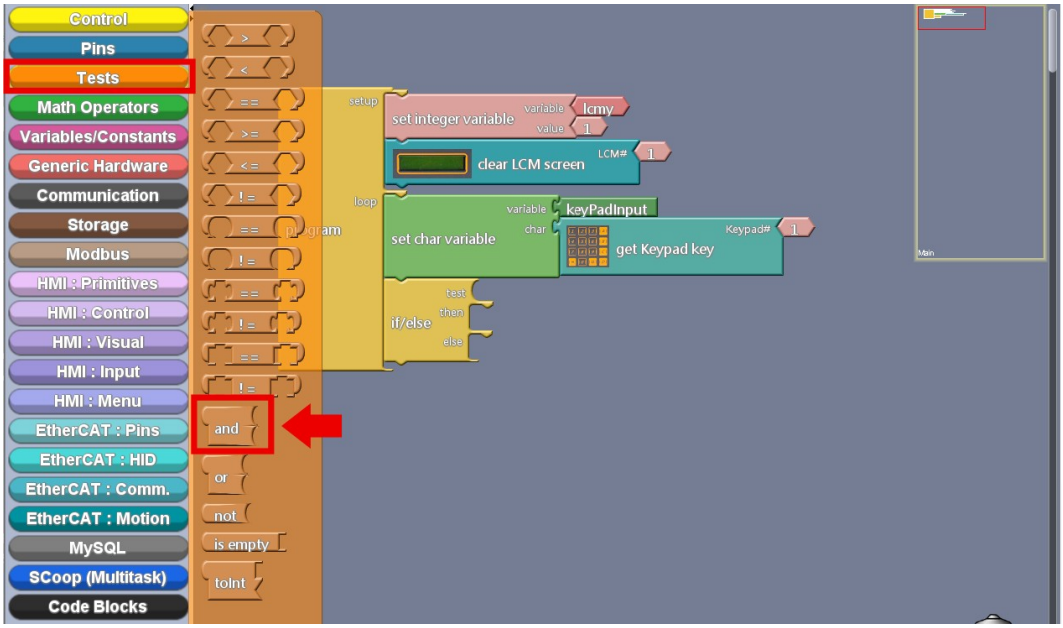


下図のようになります。



まず、キーパッドの入力値が数字 '0'..'9' かどうかを判定します。行 = lcmY、桁 = key - '0' + 1（'0'→1 … '9'→10）とします。

「**Tests**」クラスの「**and**」ブロックを、「**if/else**」ブロックの「**test**」エリアに配置します。

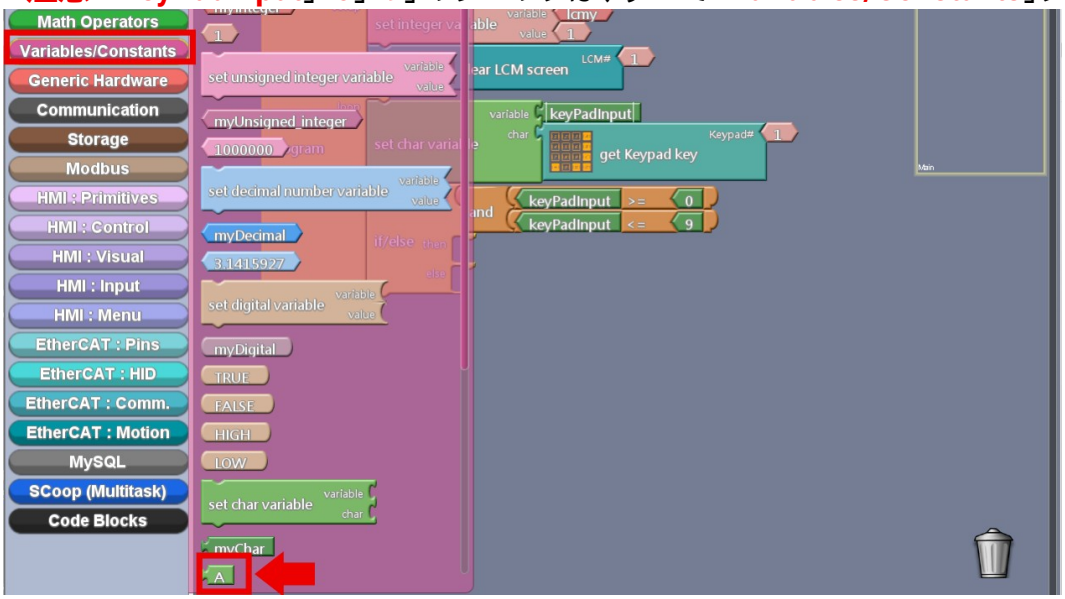


「**>=**」と「**<=**」のブロックを使用して、「**keyPadInput**」の値が **'0'** から **'9'** の間かどうかを判定します。

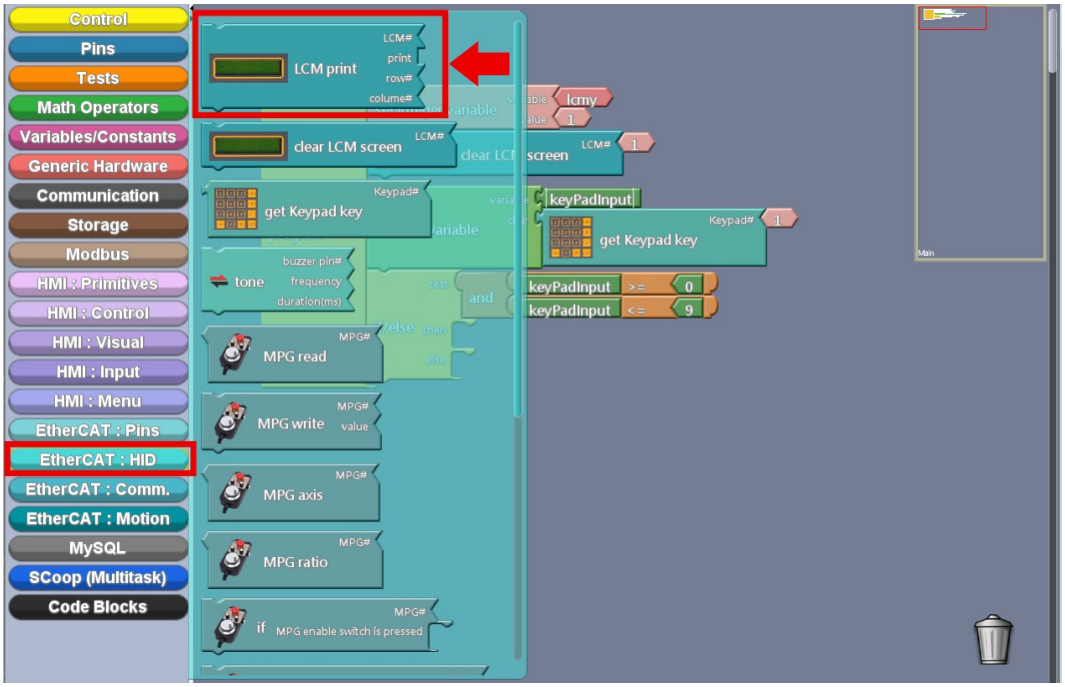
下図のようになります。



※注意：「**keyPadInput**」「**0**」「**9**」のブロックは、すべて「**Variables/Constants**」クラスの「**char**」変数です。

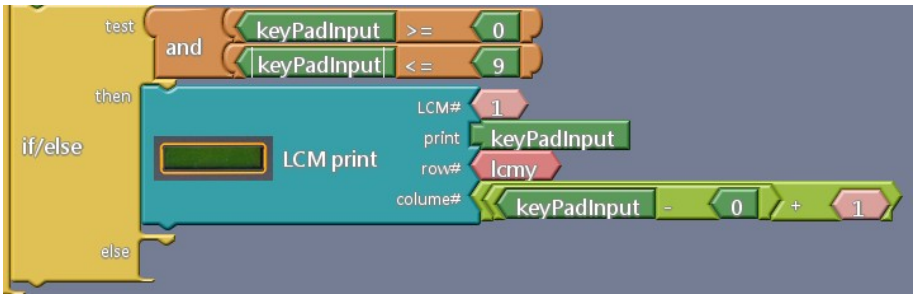


「**EtherCAT : HID**」クラスの「**LCM print**」ブロックを、「**if/else**」ブロックの「**then**」エリアに配置します。

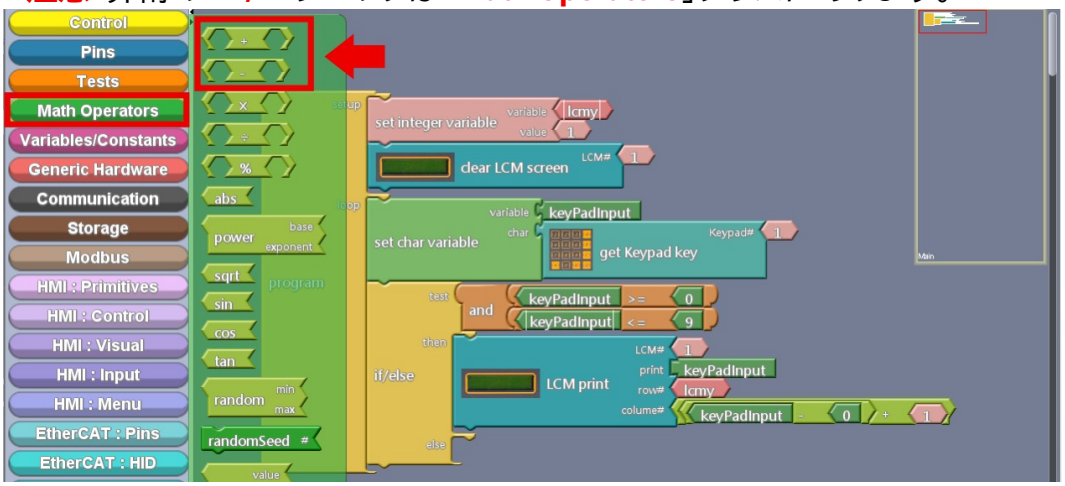


「**Virtual Lcm1**」を使用し、「**keyPadInput**」の値を、行 =「**lcmY**」、桁 = **“KeyPadInput” - '0' + 1** の位置に表示します。

下図のようになります。

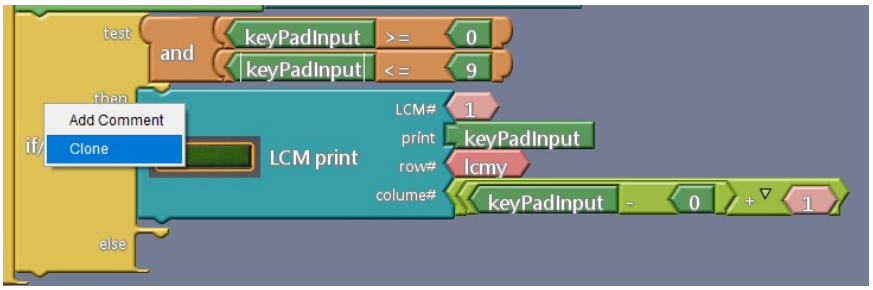


※注意：算術の **“+/-”** ブロックは「**Math Operators**」クラスにあります。



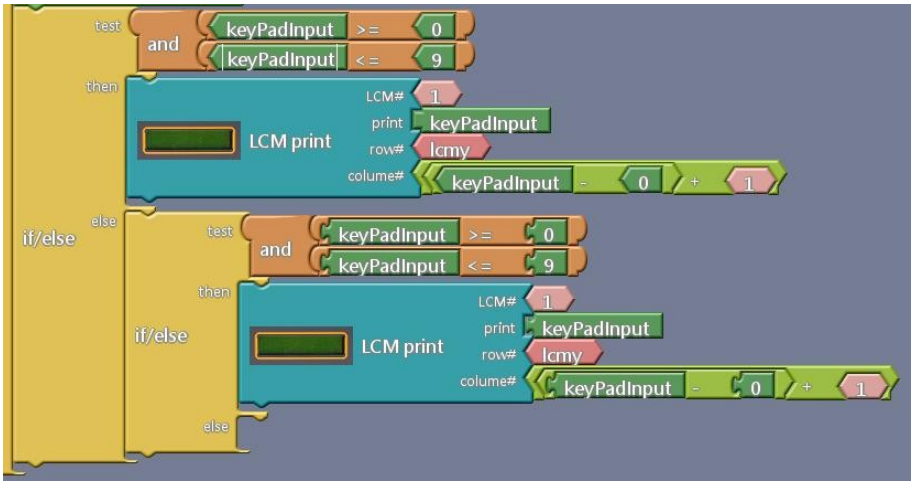
次に、「**if/else**」ブロックの「**else**」エリアで、キーパッドの入力値が文字 'A'..'D' かどうかを判定します。行 = lcmY、桁 = key - 'A' + 11（11..14）とします。

「**if/else**」ブロックを右クリックすると、「**Add Comment**」と「**Clone**」の 2 つの項目が表示されます。



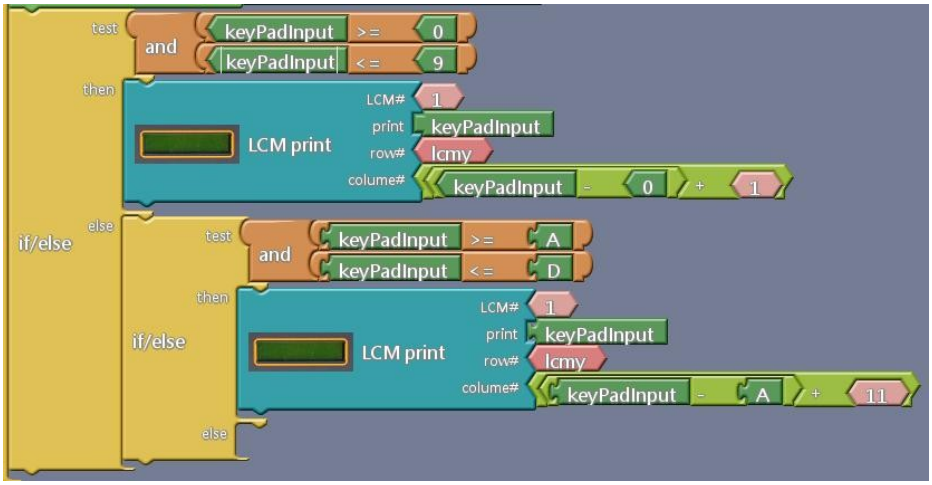
「**Clone**」をクリックして、設定済みの「**if/else**」内のブロックをすべて複製し、必要に応じてパラメーターを調整します。複製した「**if/else**」ブロックは、元の「**if/else**」ブロックの「**else**」エリアに配置します。

下図のようになります。



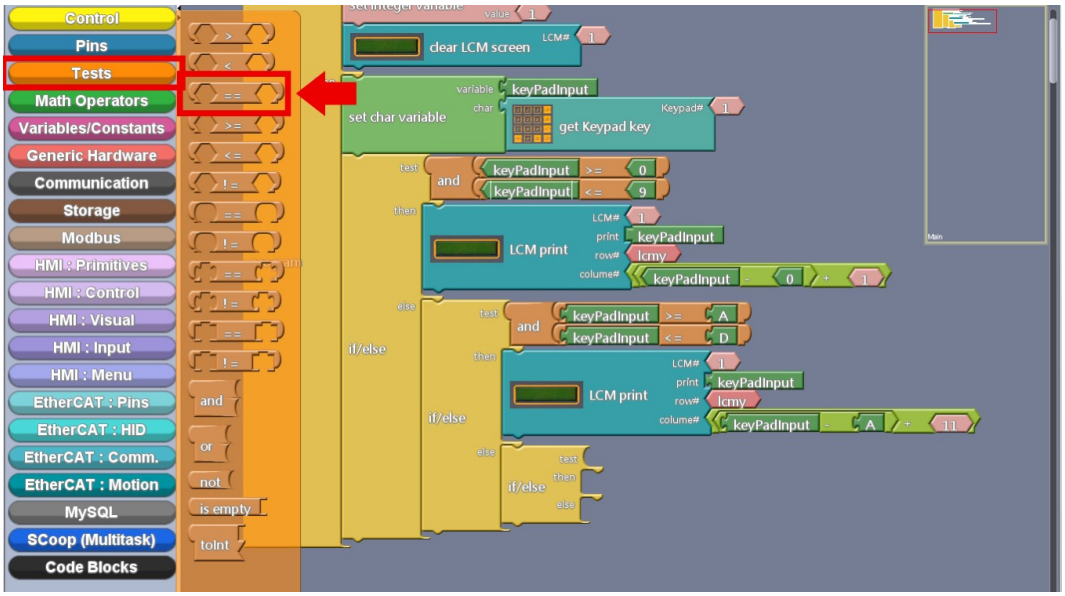
「**test**」エリアの「**and**」ブロックで、判定条件を「**keyPadInput**」の値が‘**0**’と‘**9**’の間から、‘**A**’と‘**D**’の間に変更します。
次に、「**column**」の値を“**KeyPadInput**” - '0' + 1 から“**KeyPadInput**” - 'A' + 11 に変更します。

下図のようになります。



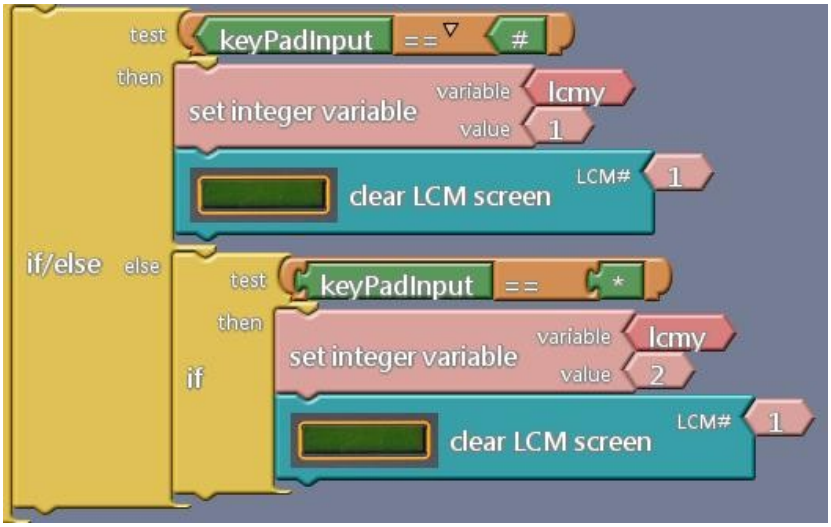
キーパッドの入力値「**#**」と「*****」を判定するため、もう 1 つ「**if/else**」ブロックを作成します。このブロックは、‘A’～‘D’を判定する「**if/else**」ブロックの「**else**」エリアに配置します。

「**Test**」クラスの「**==**」ブロックを使用して、「**keyPadInput**」の値が「**#**」か「*****」かを判定します。



まず、「**if/else**」ブロックの「**test**」エリアで「**keyPadInput**」の値が「**#**」かどうかを判定し、「**then**」エリアで「**lcmY**」を 1 に変更して「**clear LCM screen**」ブロックで LCM をクリアします。次に「**else**」エリアにもう 1 つ「**if**」ブロックを追加して、「**keyPadInput**」の値が「*****」かどうかを判定します。該当する場合は「**lcmY**」を 2 に設定し、「**clear LCM screen**」ブロックで LCM をクリアします。

下図のようになります。



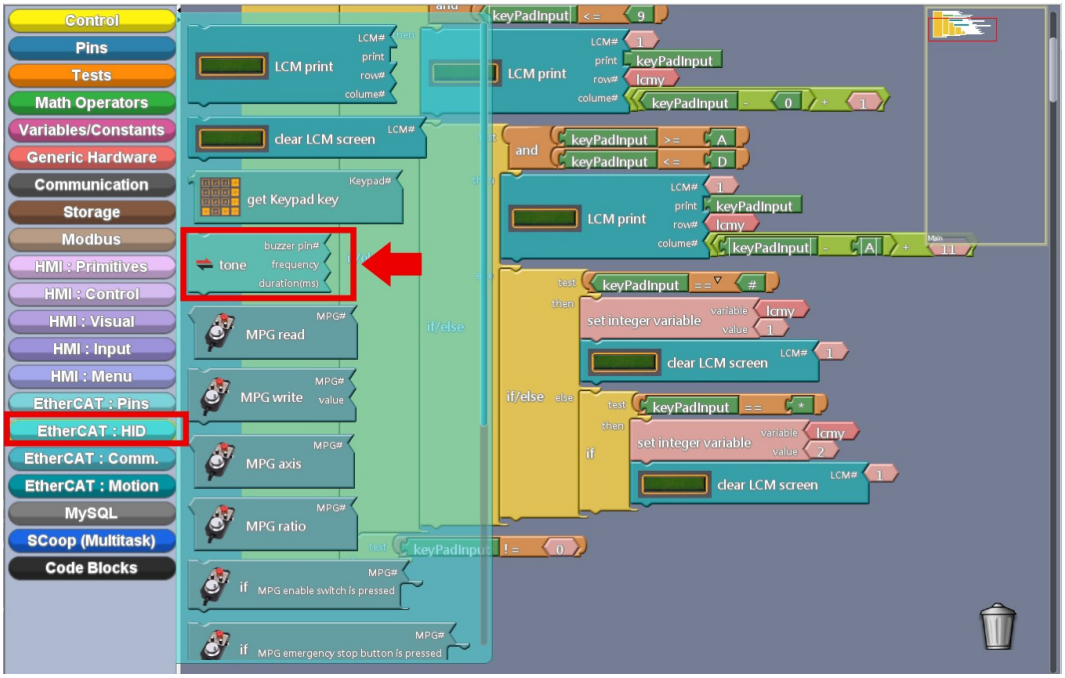
最後に、キーが押されたときだけ、周波数 3000 Hz で 200 ms ブザーを鳴らします。

「**Control**」クラスの「**if**」ブロックをプログラムのメインループに配置し、キーパッドが押されたかどうかを判定します。
「**if**」ブロックの「**test**」エリアで「**!=**」ブロックを使用し、「**keyPadInput**」の値が 0 でないかどうかを判定します。

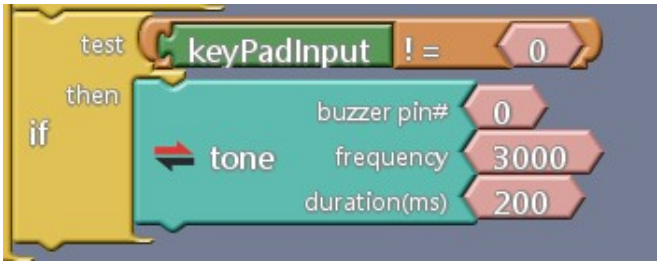
下図のようになります。



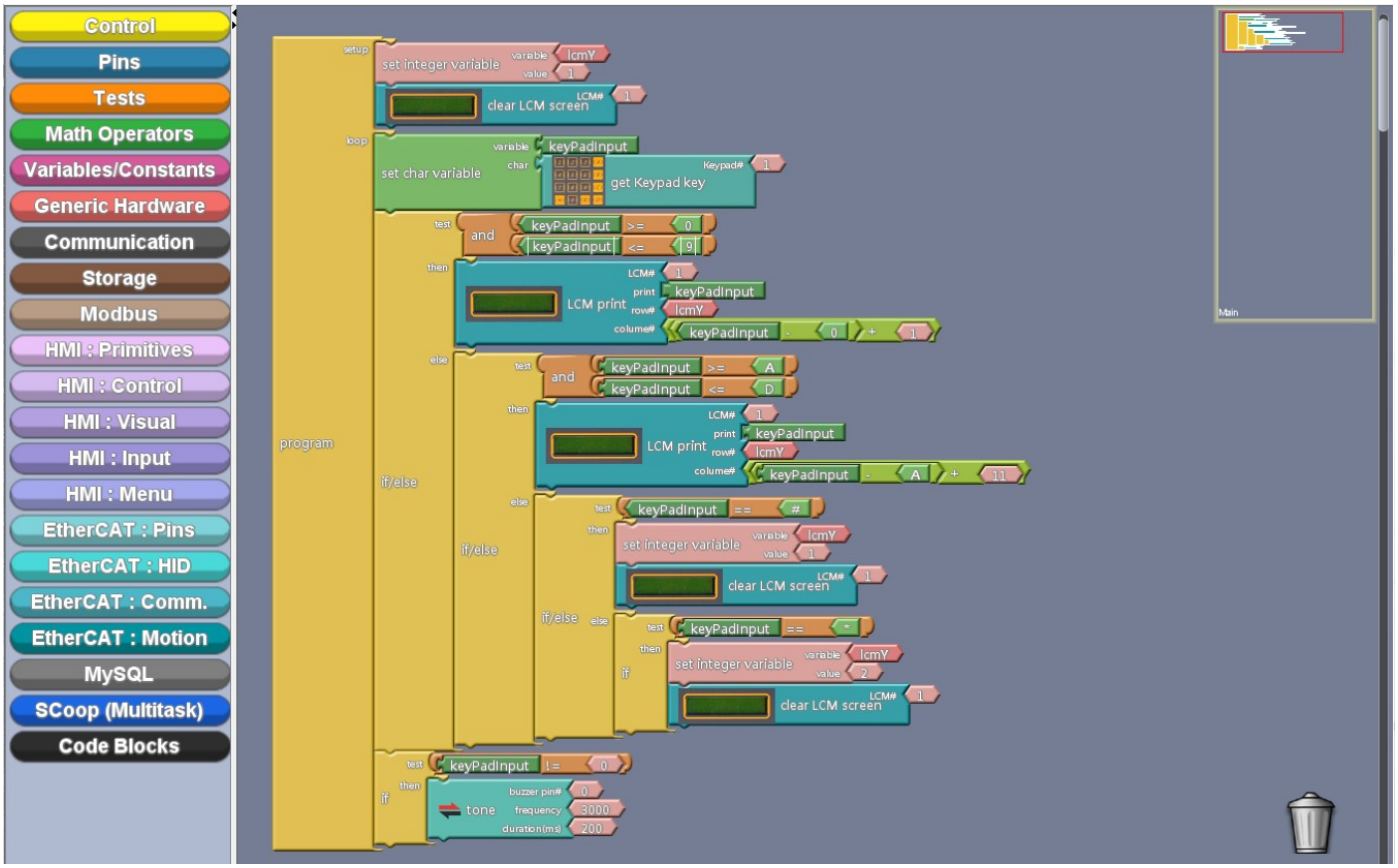
次に、「**EtherCAT : HID**」クラスの「**tone**」ブロックを「**if/else**」ブロックの「**then**」エリアに配置します。



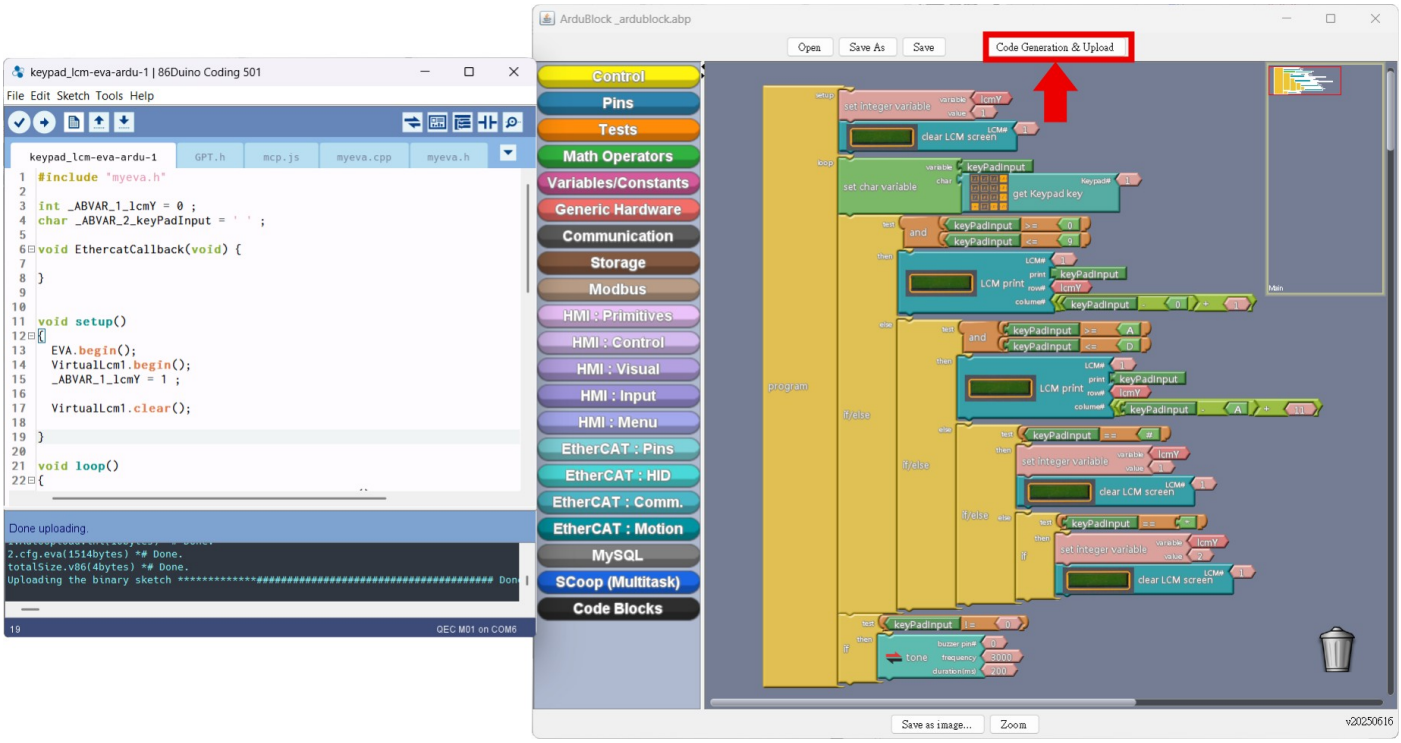
ブザーの周波数を 3000 Hz、時間を 200 ms に設定します。
下図のようになります。



完成したブロック全体は下図のとおりです。



完成したら「**Code Generation & Upload**」ボタンをクリックすると、ArduBlock がプログラムコードを自動生成し、QEC-M-01 に書き込みます。



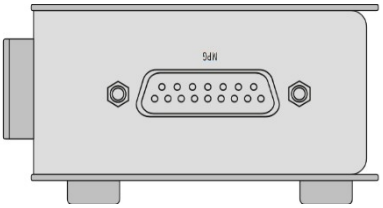
書き込み後、QEC-R11HU9S-N 側面のキーパッドを押すと、押したキーに応じて LCM の指定位置に表示されます。キーパッドを押すとブザーが鳴ります。
それぞれの動作は次のとおりです。

- # 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。



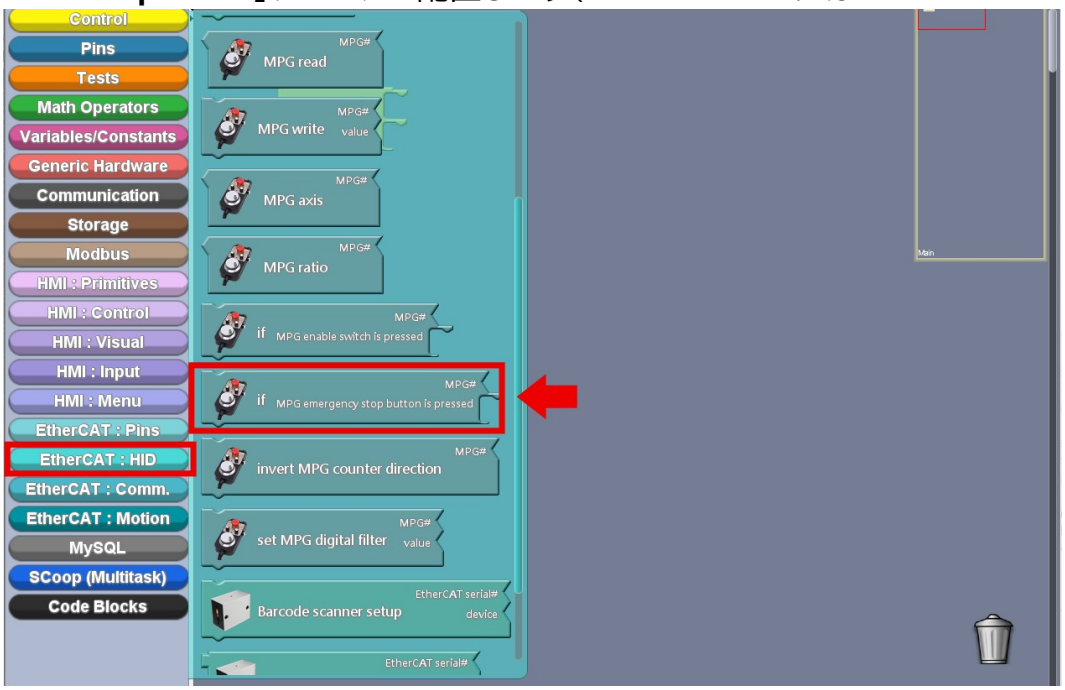
ステップ 4.3：MPG ハンドホイール

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 の各データを 86Duino IDE のシリアルモニタに表示します。

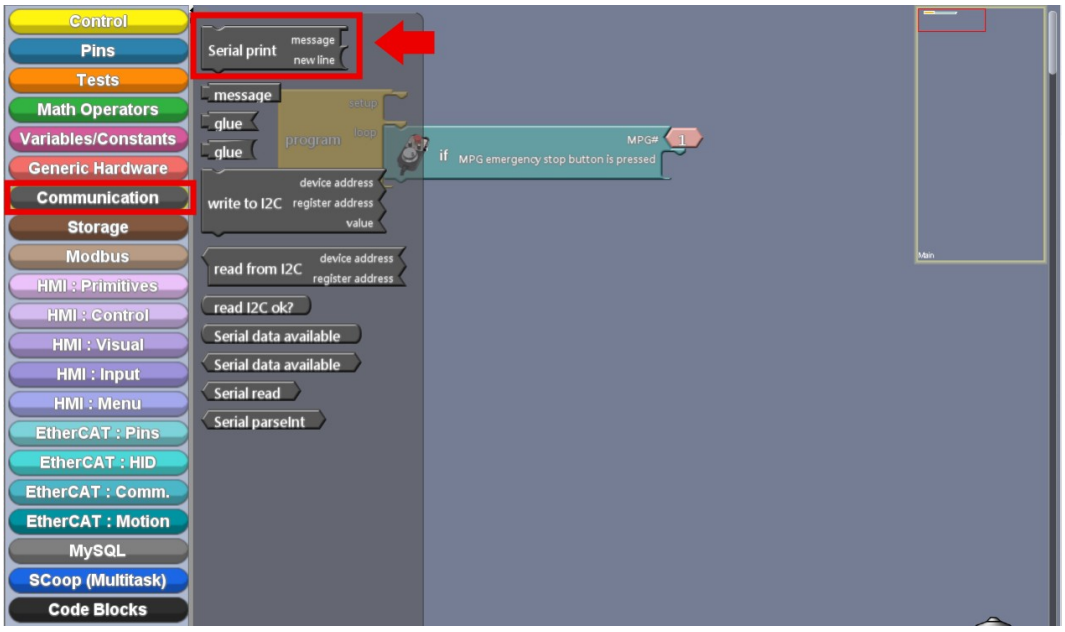


No.	ピン割り当て	No.	ピン割り当て
1	VCC	9	AXIS_B0
2	A	10	AXIS_B1
3	A-	11	AXIS_B2
4	B	12	MULTIPLE_B0
5	B-	13	MULTIPLE_B1
6	C	14	EMERGENCY
7	C-	15	LED
8	GND	-	-

まず、「**program**」ブロックの「**loop**」エリアに、「**EtherCAT : HID**」クラスの「**if MPG emergency stop button is pressed**」ブロックを配置します（HU9S の MPG には「**Virtual MPG1**」を選択しています）。

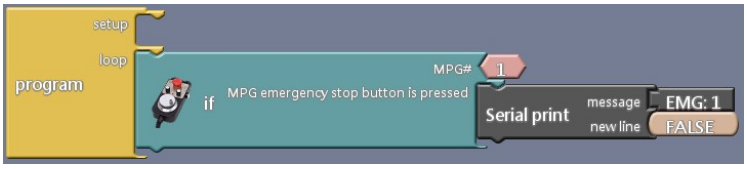


「**Communication**」クラスの「**Serial print**」ブロックを、「**if MPG emergency stop button is pressed**」の中に配置します。

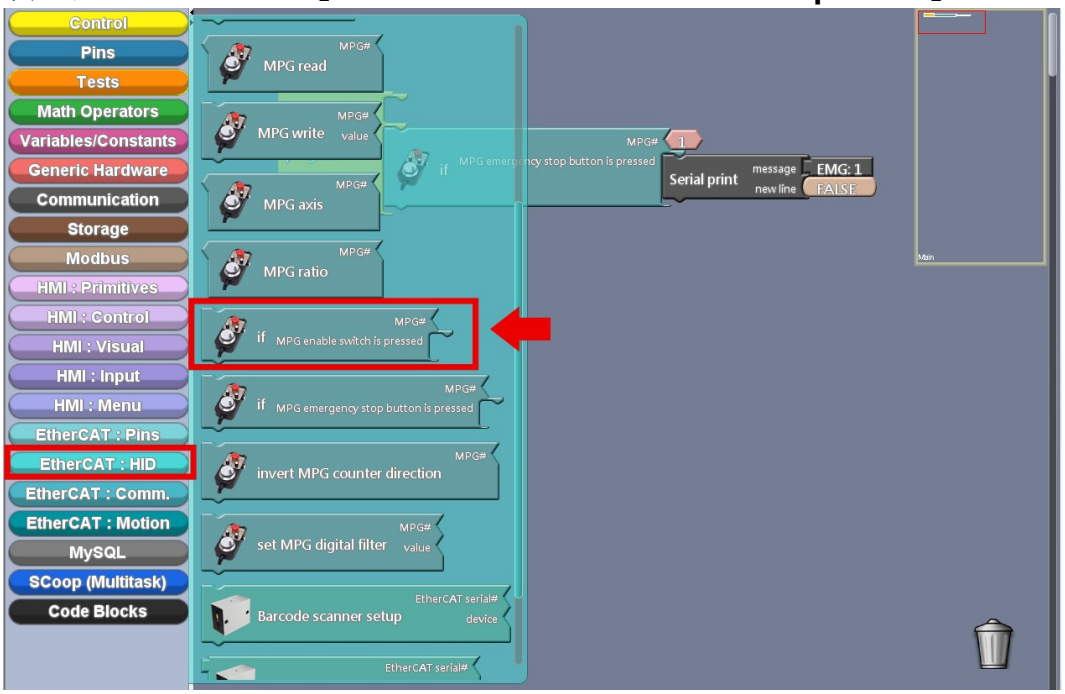


「**Serial print**」ブロックの「**message**」を「**EMG: 1**」に、「**new line**」を「**FALSE**」に設定します。

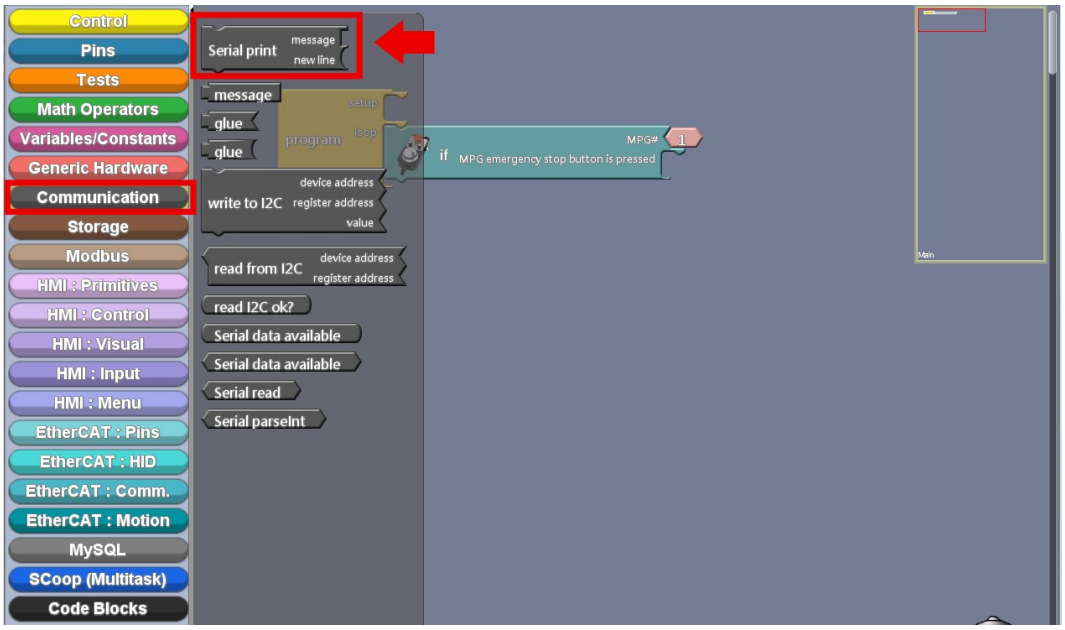
下図のようになります。



次に、「**EtherCAT : HID**」クラスの「**if MPG enable switch is pressed**」ブロックを使用します。

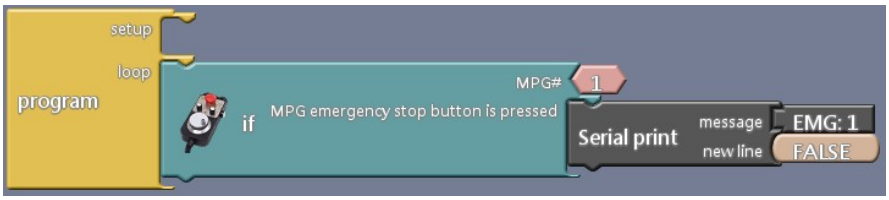


同様に、「**Communication**」クラスの「**Serial print**」ブロックを「**if MPG enable switch is pressed**」の中に配置します。



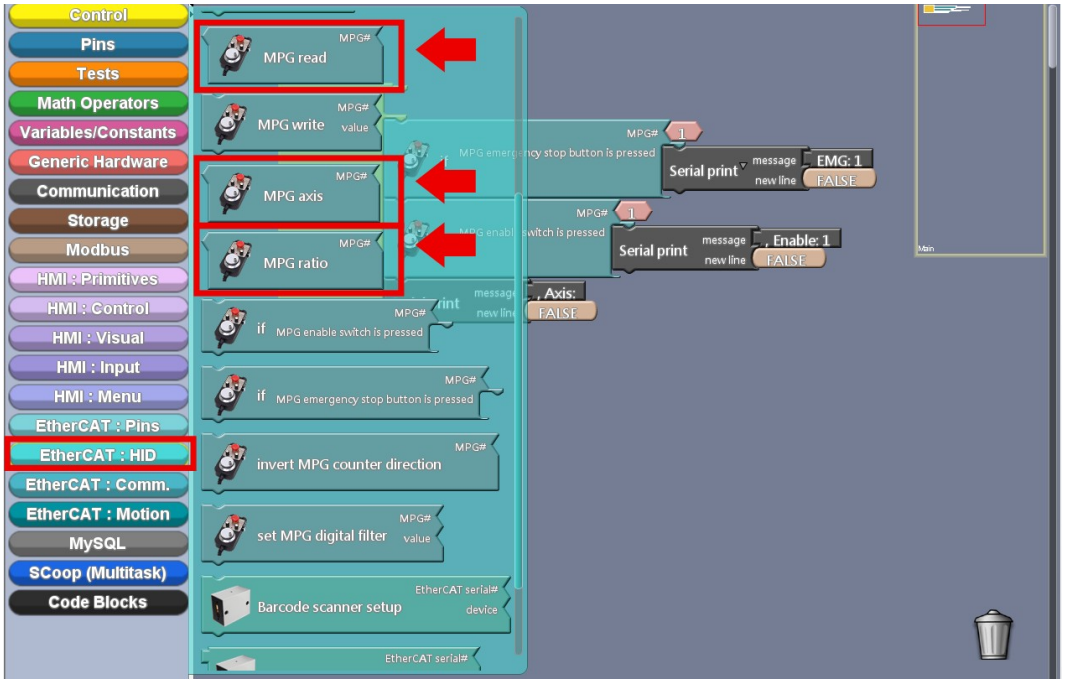
「**Serial print**」ブロックの「**message**」を「**, Enable: 1**」に、「**new line**」を「**FALSE**」に設定します。

下図のようになります。

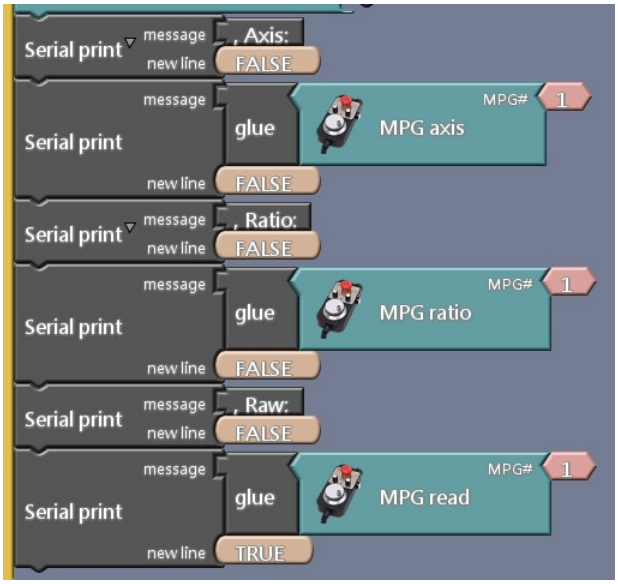


次に、軸、倍率、生データを表示します。

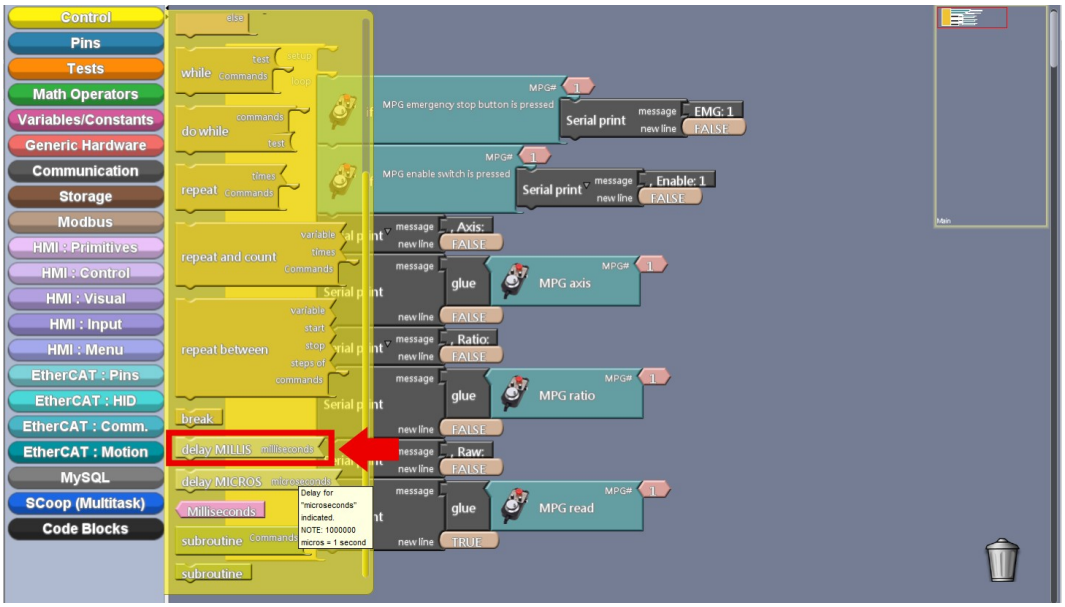
「**Serial print**」ブロックを 3 つ使用し、「**message**」をそれぞれ「**, Axis:**」「**, Ratio:**」「**, Raw:**」に設定します。
さらに、「**EtherCAT : HID**」クラスの「**MPG axis**」「**MPG ratio**」「**MPG read**」の各ブロックを使用します。



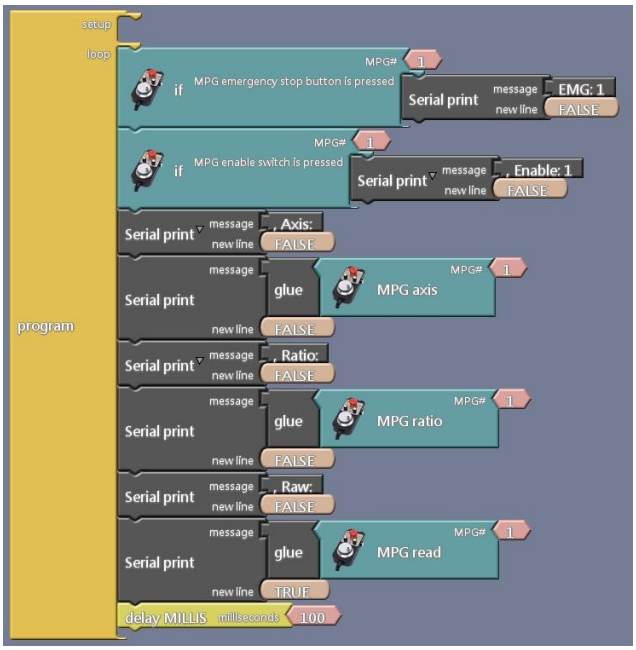
「**Serial print**」ブロックと「**EtherCAT : HID**」ブロックを「**program**」ブロックの「**loop**」エリアに配置します。メッセージの内容は「**EtherCAT : HID**」ブロックに変更する必要があります。
「**Communication**」クラスの「**glue**」ブロックを使用して、「**EtherCAT : HID**」ブロックを「**Serial Print**」ブロックの「**message**」エリアにつなげます。
下図のようになります。



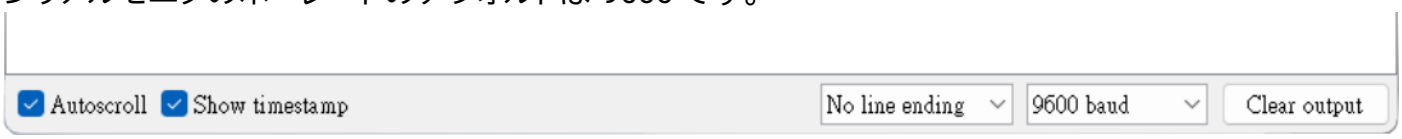
次に、MPG の応答時間を待つ必要があるため、すべてのブロックの後に「**delay MILLIS**」ブロックを配置します。



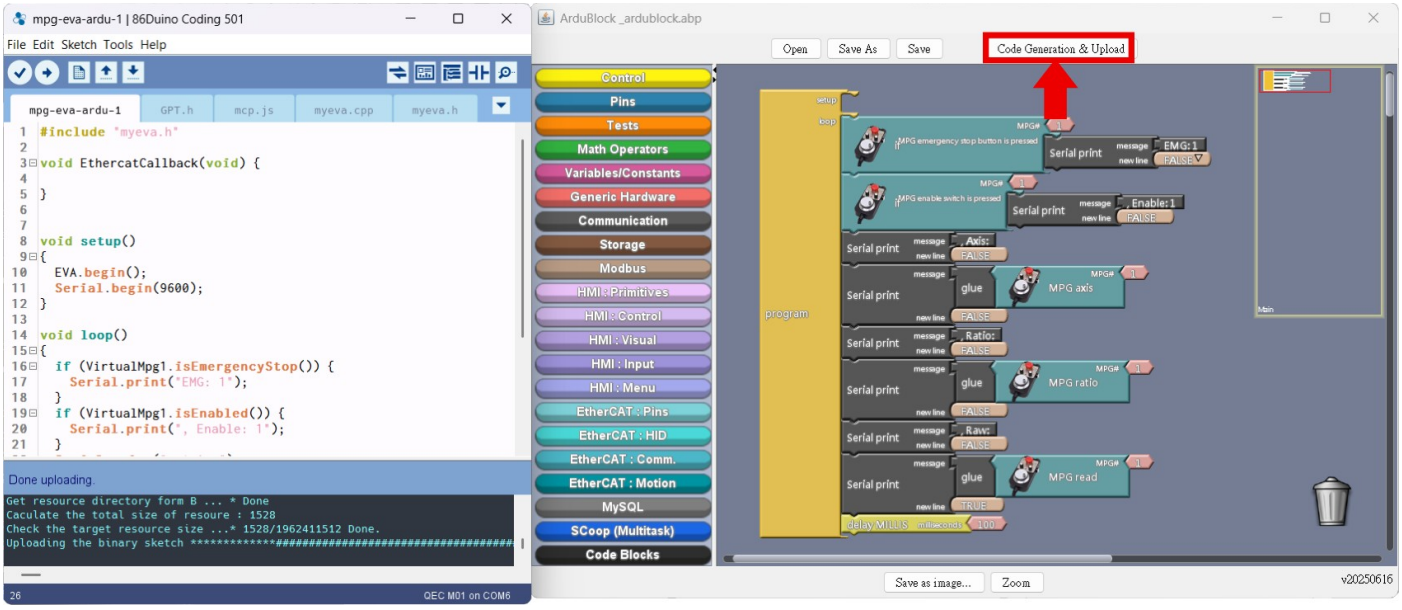
「**delay MILLIS**」ブロックに 100 ms を設定します。
下図のようになります。



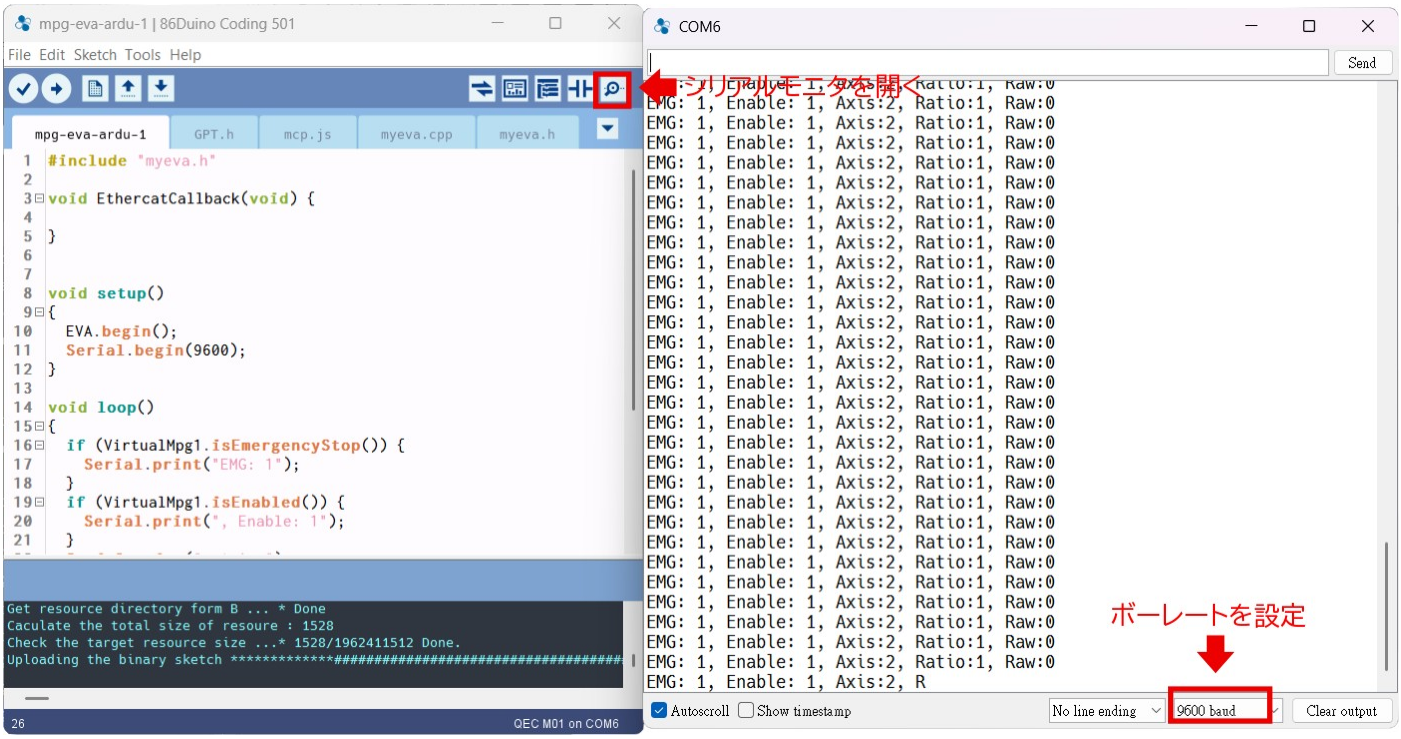
※注意：
シリアルモニタのボーレートのデフォルトは 9600 です。



完成したら「**Code Generation & Upload**」ボタンをクリックすると、ArduBlock がプログラムコードを自動生成し、QEC-M-01 に書き込みます。



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 のデータを 86Duino IDE のシリアルモニタで確認できます。



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。

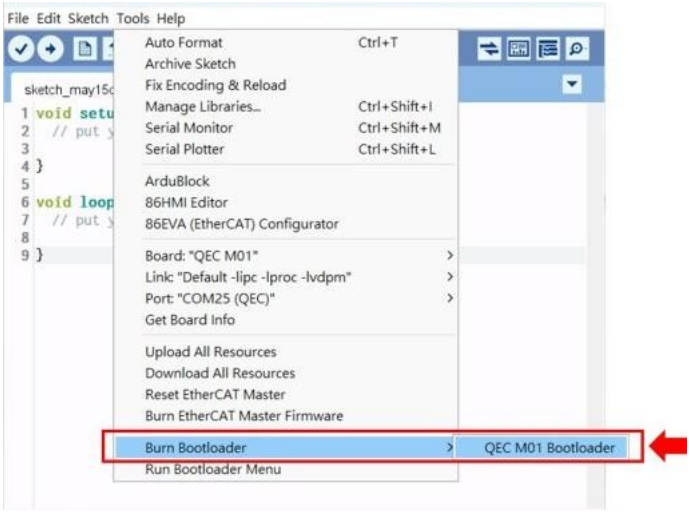
ステップ 1：QEC-M の準備

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。

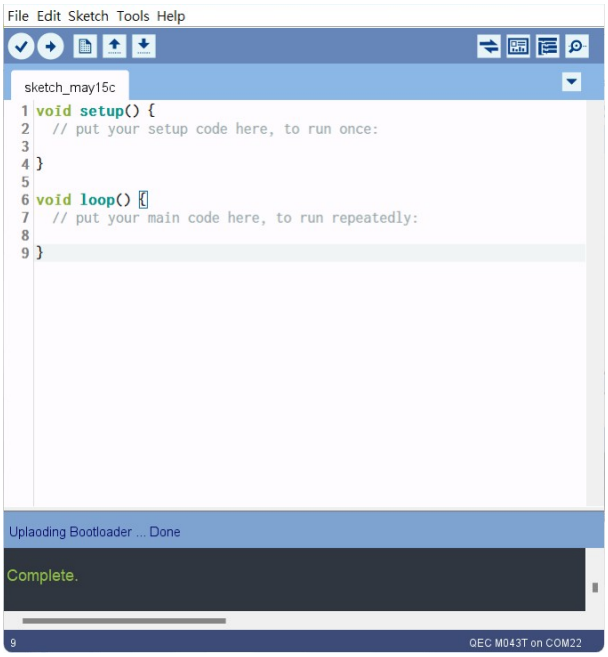
ステップ 2：「Burn Bootloader（ブートローダを書き込む）」をクリック

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。