---電磁弁基板 Daigaku シリーズ制御用ライブラリ仕様書---

ver.2019_9_28

目次

1. このライブラリについて

このライブラリは、電磁弁基板「Daigaku_8_V1」（以下 D_8_1）を制御するためのライブラリです。マイコンは esp32（**ＥＳＰ３２－ＤｅｖＫｉｔＣ　ＥＳＰ－ＷＲＯＯＭ－３２開発ボード**）を使用する事を前提に、Arduino で書かれています。

また、開発環境は vscode で、platformio が導入されている事を前提に説明を進めますので、こちらも各自導入してください。

2. 使用方法

2-1. おまじないと書き換えが必要な部分

~おまじない~ ※書き換え箇所 main.cpp

まず、ピンアサイン等が定義されたライブラリをインクルードします。

```
#include <Arduino.h>
#include "daigaku.h"
/*---次のライブラリをインクルードしてください。ディスコードからダウンロードして！---*/
#include "RoboMotherMINI.h"            //ピンアサインが定義されている。
```

次に、関数 setup 内に次の文を追加してください。

```
void setup() {
  /*---次の文を追加してください。---*/
  SPI.begin(SCLK,MISO,MOSI,-1);       //SPI 通信の開始。
  //SCLK:SPI 通信のクロックのピンアサイン(RoboMotherMINI.h で定義されている。)
  //MISO,MOSI:SPI 通信のバスのピンアサイン(RoboMotherMINI.h で定義されている。)
  //-1:本来ここに、スレーブセレクトのピンアサインを書くが、daigaku.h 側で定義済みであるから、-1(定義なし)と表記した。
}
```

~基板の枚数によって…~ ※書き換え箇所 daigaku.h

扱う D_8_1 の枚数によって次の値を書き換えてください。

```
/*---ALL_BYTE は基板の枚数によって変えてください。---*/
#define ALL_BYTE 1                    //ALL_BYTE...送受信のデータ量:1~3[BYTE]
```

2-2. Daigaku クラス

実際に電磁弁やスイッチを扱うには、Daigaku クラスを使用します。各種関数や変数の使い方を簡単に示します。

~初期化~

```
Daigaku name(RCLK,CS,MISO);
//name:任意のインスタンス名
//RCLK:シフトレジスタ内部のストレージレジスタに与えるクロック(DEV9 or DEV10)
//CS:スレーブセレクト(DEV9 or DEV10) ※DEV1~10 までRoboMotherMINI.h で定義済み。
//MISO:SPI のバス                    ※RoboMotherMINI.h で定義済み。
```

name は任意の名前で良いです。MISO はそのまま表記してください。

RCLK,CS にはピンアサインを渡します。

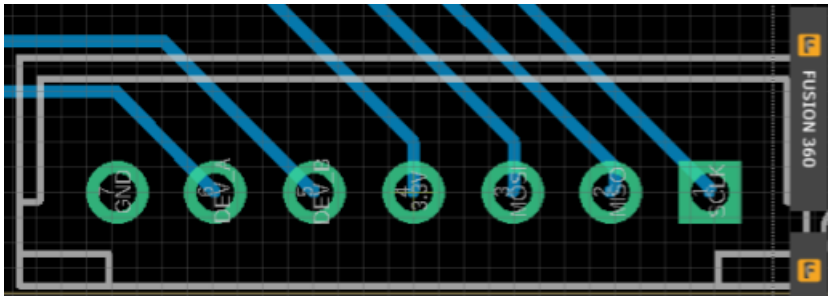
RCLK のバスに、DEV_A を使う基板の場合は DEV10 を、DEV_B を使う基板の場合は DEV9 を渡してください。DEV_B がぶった切られてる基板の場合、RCLK のバスには DEV_A が使用されていますから、DEV10 を渡してください。以下に、DEV_B がぶった切られてる基板の場合の初期化例を示します。

```
Daigaku name(DEV9,DEV10,MISO);
//RCLK のバスにDEV9 を用いているので、CS のバスにはDEV10 を用いた。
```

※使うマザーのバージョンによって異なるそうです。DEV10 や DEV9 と変更して両方試しても通信できない場合、お知らせください。

参考までに、マザーと D_8_1 をつなぐ XH コネクタのピンアサインを示します。

ピン番号 6 が DEV_A でピン番号 5 が DEV_B です。



~配列：send_data~

```
unsigned char send_data[ALL_BYTE * 8];
```

先ほど書き換えた ALL_BYTE×8 がこの配列の要素数になります。

オンにしたいポートと配列の添え字が対応しています。

```
name.send_data[num] = val;
//num:オンにしたいポートの番号 - 1
//val: 1:オン 0:オフ
```

例えば、ポート 5 とポート 7 をオンにポート 3 をオフにする場合、次のように書きます。

```
name.send_data[4] = 1;
name.send_data[6] = 1;
name.send_data[2] = 0;
name.shift();                    //ポートの状態を電磁弁の出力に反映。
```

~配列：sw_status~

```
unsigned char sw_status[ALL_BYTE * 8];
```

先ほど書き換えた ALL_BYTE×8 がこの配列の要素数になります。

関数 shift が実行されると各ポートのスイッチの状態を次のように取得できます。

```
name.shift();
printf("---ポート 1~8 までの状態を取得--¥n");//※ALL_BYTE が 1 の時です。
for(num = 0;num < SW_BYTE * 8;num++){
        printf("ポート%d:%d¥n",num + 1,name.sw_status[num]);
}

//num:スイッチのポートの番号 - 1
//1:オン 0:オフ
```

関数 shift が実行されない限り、sw_status には前回の値が残り続けます。

※基本的に sw_status の中身は書き換えないでください。

~関数：shift~

```
void shift(void);
```

send_data に格納されている各ポートの状態を、電磁弁の出力に反映します。

sw_status に各ポートのスイッチの状態を代入します。

※send_data に状態を入れるだけでは電磁弁は起動しません。send_data を書き換えるたびに必ずこの関数を呼び出してください。

```
//各ポートの状態を電磁弁に反映する。また、スイッチの状態を取得する。
name.shift();
```

~関数：reset~

```
void reset(void);
```

この関数を呼び出すと、send_data は全て初期値（0）となり、電磁弁もすべてオフとなります。

※sw_status の値は変化しません。つまりスイッチ入力には一切関係ない関数です。

```
//電磁弁をリセットする。
name.reset();
```

3. 基板の使い方注意点

マイコン起動時に各 IO ピンから意図しない信号が出ます。この影響を Daigaku シリーズの電磁弁基板は諸受けます。何も対策しないとマイコン起動時に、全ての電磁弁が勝手に開放される恐れがあるので、次の手順で基板を操作してください。

① すべての電源がオフであることを確認する。

② マイコンの電源を入れる。

③ 電磁弁（基板）の電源を入れる。

これだけの事です。マイコンの電源と電磁弁の電源が同じ場合、電磁弁基板の VDD ポートの手前にスイッチを付けて使用してください。