RMC-RA4M1 (rev.1.0b) 基板
実習マニュアル

※マイコンカー走行プログラム「kit23_ra4m1_msd」、「kit23_ra4m1_servo_tyosei」も掲載しています。

※本マニュアルは、2023 年度のマイコンカー技術講習会で配布された基板「RMC-RA4M1(rev.1.0b)」について解説しています。

※2024 年 3 月頃に販売される予定の基板「RMC-R4M1(rev.2.0)」とは、端子の配置が大幅に異なりますので、ご注意ください。

万が一「本マニュアル」による損失・損害が発生した時には、マニュアル制作団体、制作者はいかなる場合も責任を負いません。個人の免責が取れる範囲内であらかじめ了承した上でご使用くださるようにお願いをいたします。

0.04	P3 の図について、誤り: DAC AN002 P002 A3(~D17) AN009 P014 A2(D16) → 正: DAC AN009 P014 A2(~D16) AN002 P002 A3(D17) に訂正 その他、誤字・脱字の修正
0.05	ArduinoIDE に RMC-RA4M1(rev.1.0b)ボードを追加する方法を、ボードマネージャに変更
0.06	マイコンカー走行プログラム「kit23_ra4m1_msd.ino」、「kit23_ra4m1_servo_tyosei」を追記
0.07	AREF 端子を本マニュアルでは「+5V」としていたのを、「AREF」に変更、「aqm1248graphic.ino」の追加
0.08	「eeprom.ino」の追加

第 0.08 版
2024.02.04
ジャパンマイコンカーラリー実行委員会

目 次

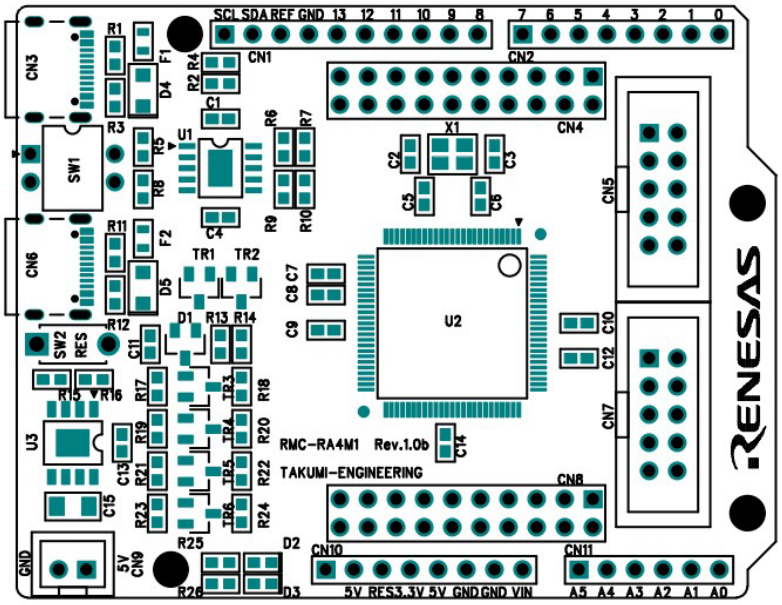
1. 概要

RMC-RA4M1(rev.1.0b)ボードと Arduino Uno R4 の仕様を、下記に示します。

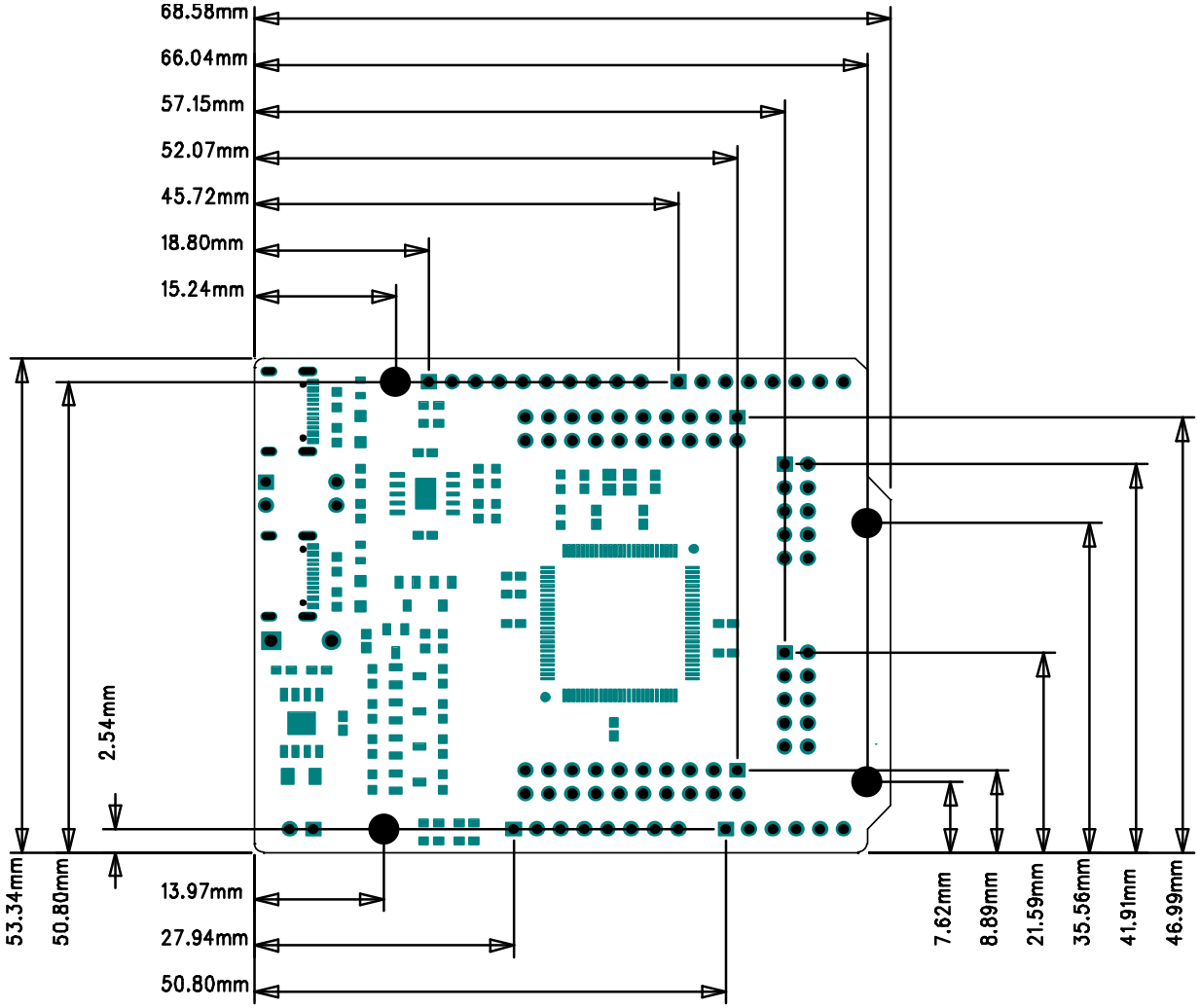
基板名	RMC-RA4M1 rev.1.0b	Arduino Uno R4 MINIMA
写真		
マイコン	ルネサスエレクトロニクス製 RA4M1 100 ピン	ルネサスエレクトロニクス製 RA4M1 64 ピン
USB	TypeC × 2 個 CN6(右中)は Renesas Flash Programmer V3 書き込み用 CN3(右上)は、Arduino IDE 用	TypeC × 1個
入力電圧	CN9 または 5V 端子に、5V±10%入力 ※VIN 端子は、未接続	入力電圧(VIN):6～24 V または、5V 端子に 5V±10%入力
クロック	48MHz	48MHz
フラッシュ ROM	256KB	256KB
RAM	32KB	32KB
データフラッシュ (EEP-ROM)	8KB	8KB
I/O	**●デジタル I/O 端子:69 端子** ※D0～D22 の 22 本、CN5 の 6 本、 CN7 の 8 本、CN4 の 16 本、CN8 の 17 本 **●デジタル入力端子:3 端子** ※CN5 の(P214)、(P215)、CN4 の(P200)	**デジタル I/O 端子:20 端子(全端子)**
PWM 出力端子	**I/O 端子の中で 16 端子**	**I/O 端子の中で 14 端子 (ホームページで公式に PWM 端子とされているのは 6 端子)**
アナログ入力	**I/O 端子の中でアナログ入力端子:25 端子**	I/O 端子の中でアナログ入力端子:6 端子 **(A0～A5 端子)**
microSD コネクタ	**あり**	なし
LED	2 個(P205 と P305 に接続)	3 個(電源用、D0、D1 に接続)
ディップ スイッチ	**2bit**(P307 と P306 に接続)	なし
コネクタ	**未実装**(部品面に付けるか、半田面に付けるか、オスか、メスかは、ユーザーで選べる)	実装済み

2. 仕様

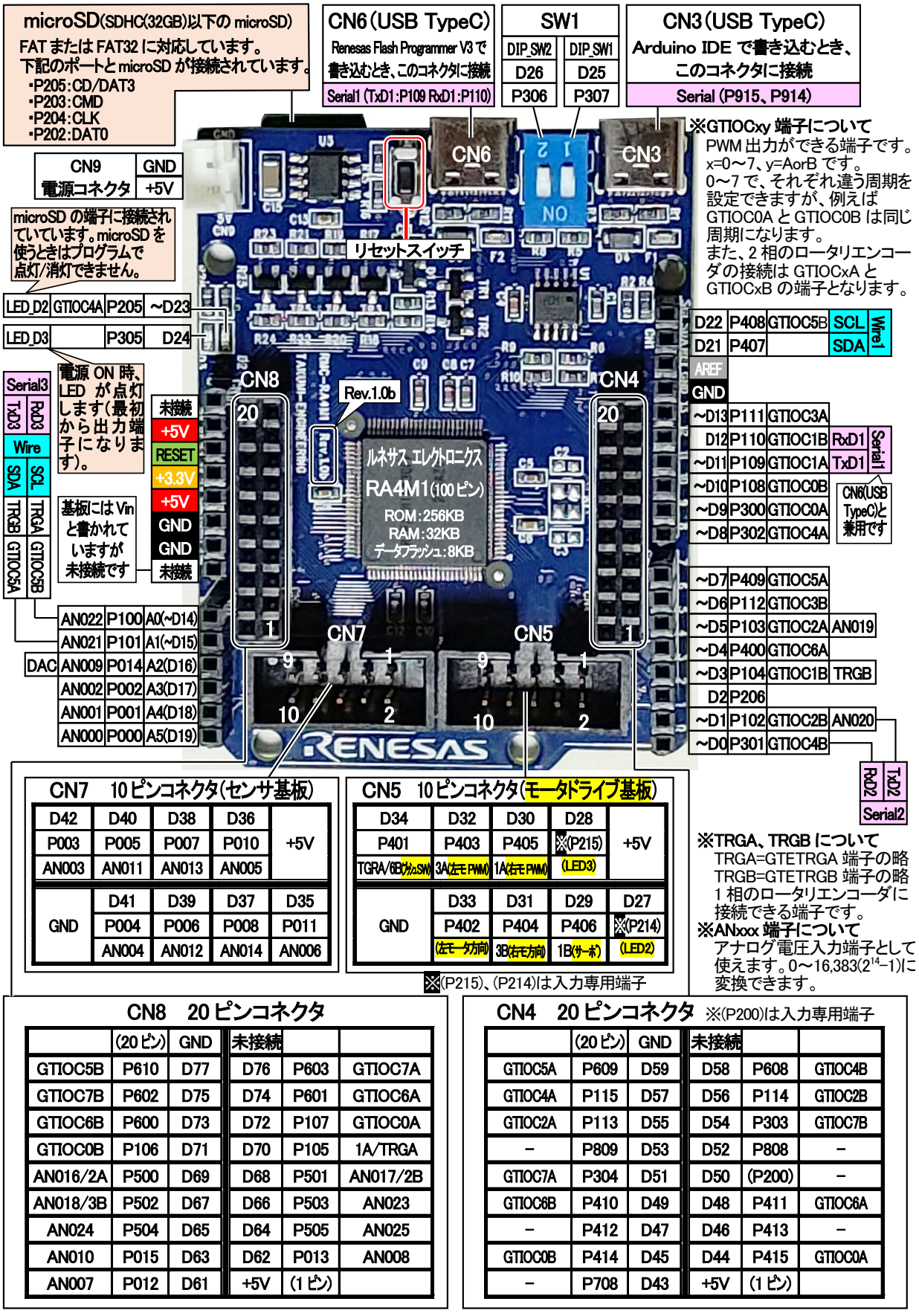
2.1. シルク図



2.2. 寸法



2.3. 外観



CN7 10ピンコネクタ(センサ基板)

D42	D40	D38	D36	+5V
P003	P005	P007	P010	
AN003	AN011	AN013	AN005	
GND	D41	D39	D37	D35
	P004	P006	P008	P011
	AN004	AN012	AN014	AN006

CN5 10ピンコネクタ(モータドライブ基板)

D34	D32	D30	D28	+5V
P401	P403	P405	※(P215)	
TGRA/6B(カムSW)	3A(左モ PWM)	1A(右モ PWM)	(LED3)	
GND	D33	D31	D29	D27
	P402	P404	P406	※(P214)
	(左モータ方向)	3B(右モ方向)	1B(サーボ)	(LED2)

※(P215)、(P214)は入力専用端子

CN8 20ピンコネクタ

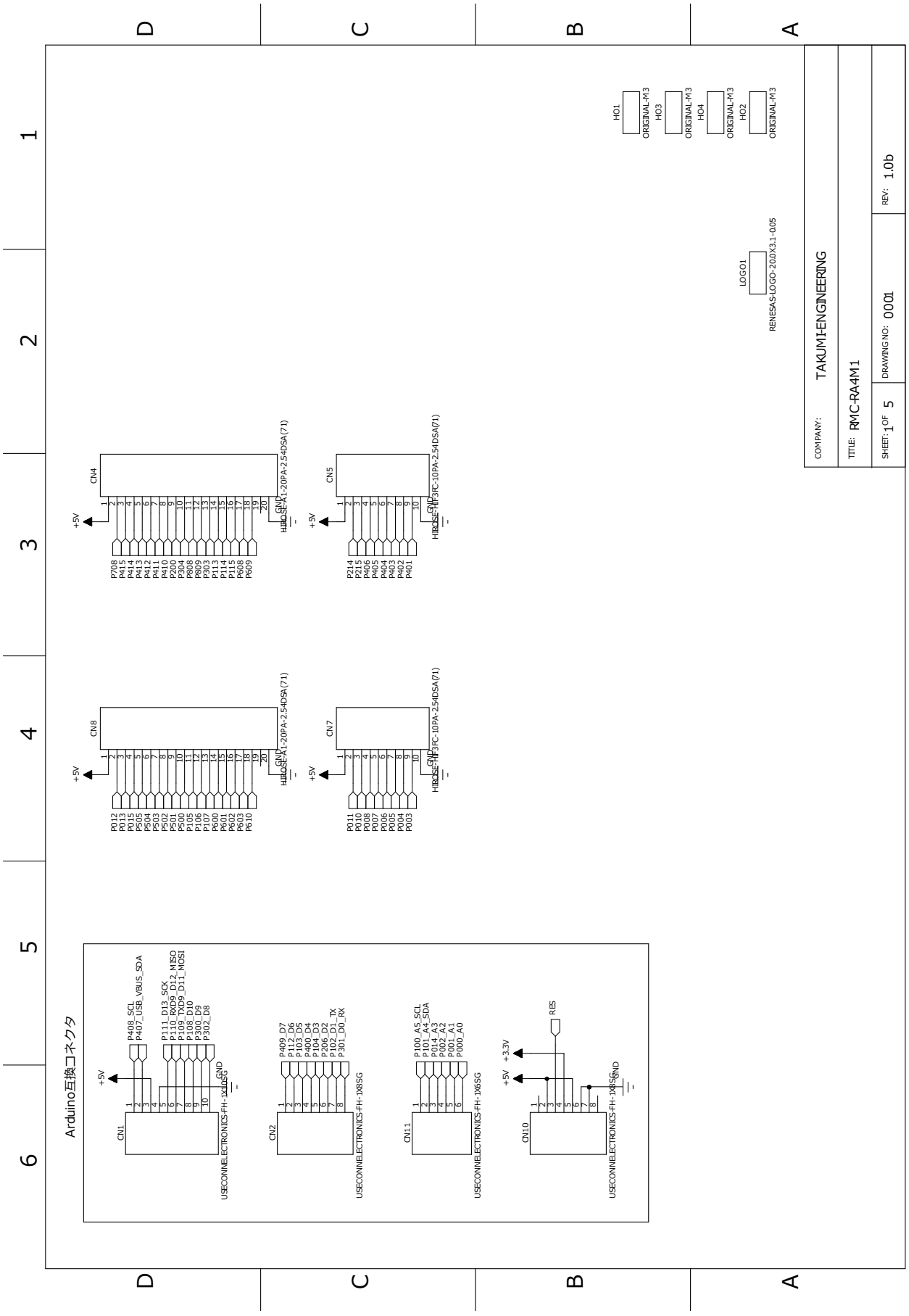
	(20ピン)	GND	未接続		
GTIOC5B	P610	D77	D76	P603	GTIOC7A
GTIOC7B	P602	D75	D74	P601	GTIOC6A
GTIOC6B	P600	D73	D72	P107	GTIOC0A
GTIOC0B	P106	D71	D70	P105	1A/TRGA
AN016/2A	P500	D69	D68	P501	AN017/2B
AN018/3B	P502	D67	D66	P503	AN023
AN024	P504	D65	D64	P505	AN025
AN010	P015	D63	D62	P013	AN008
AN007	P012	D61	+5V	(1ピン)	

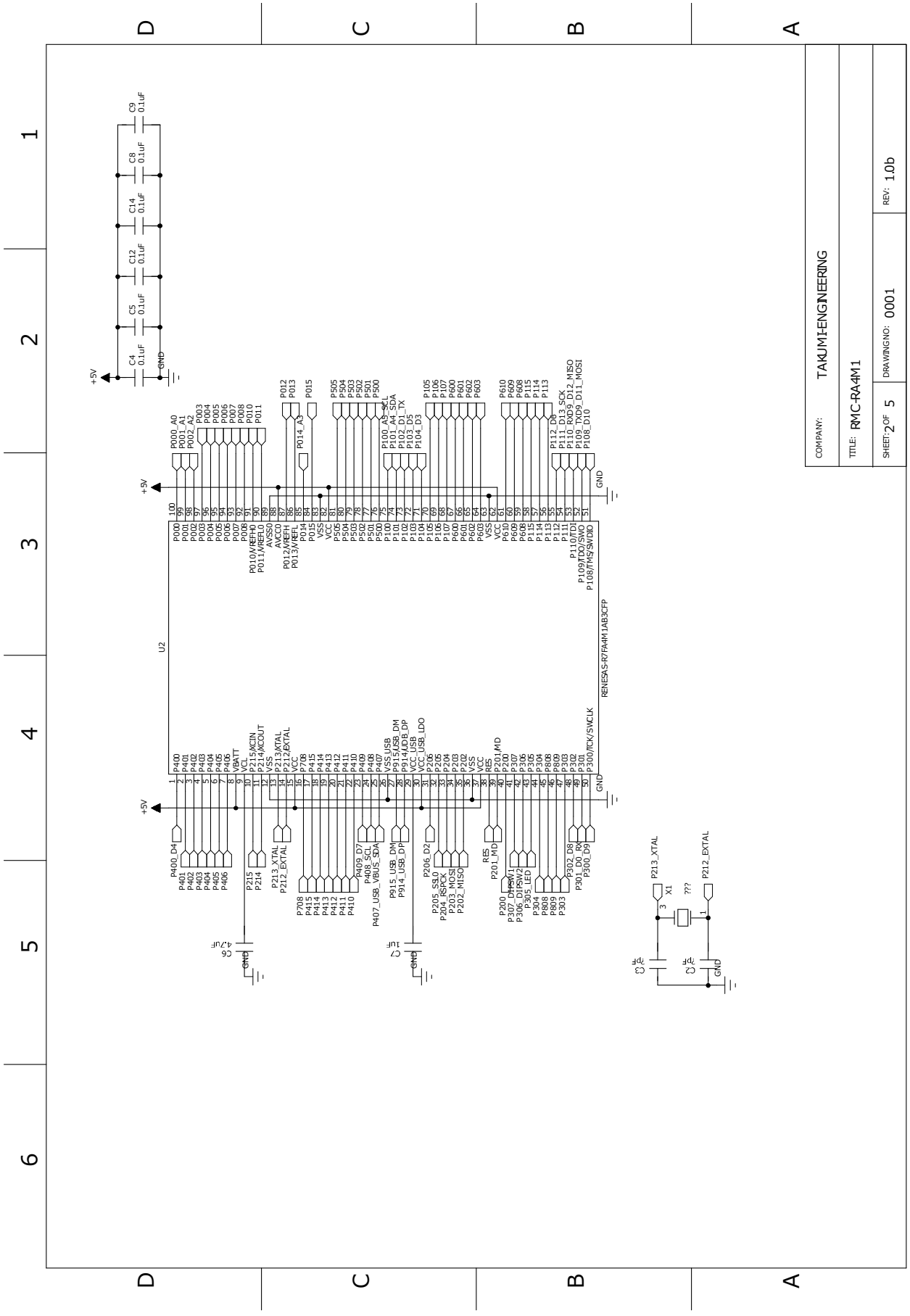
CN4 20ピンコネクタ ※(P200)は入力専用端子

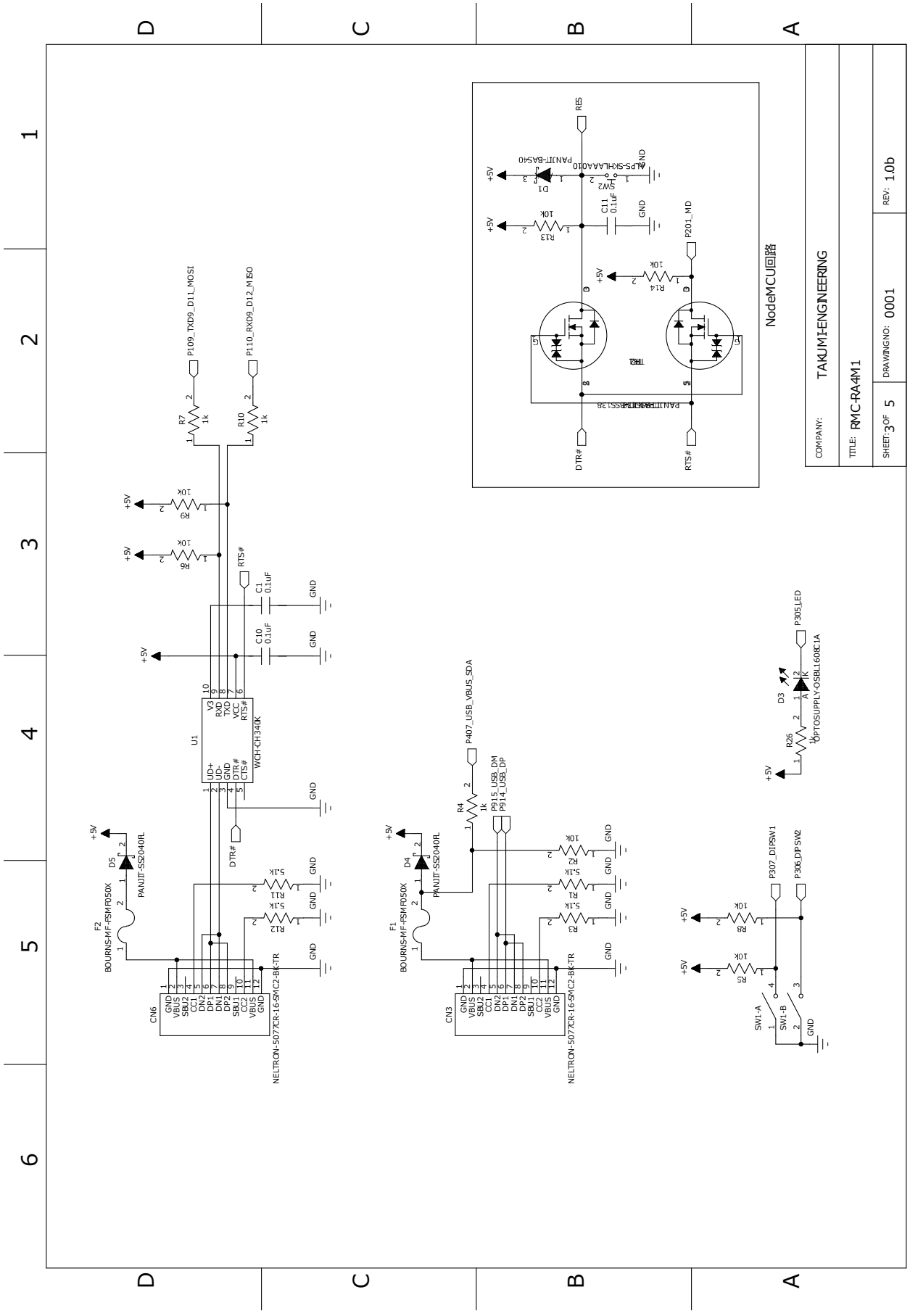
	(20ピン)	GND	未接続		
GTIOC5A	P609	D59	D58	P608	GTIOC4B
GTIOC4A	P115	D57	D56	P114	GTIOC2B
GTIOC2A	P113	D55	D54	P303	GTIOC7B
–	P809	D53	D52	P808	–
GTIOC7A	P304	D51	D50	(P200)	–
GTIOC6B	P410	D49	D48	P411	GTIOC6A
–	P412	D47	D46	P413	–
GTIOC0B	P414	D45	D44	P415	GTIOC0A
–	P708	D43	+5V	(1ピン)	

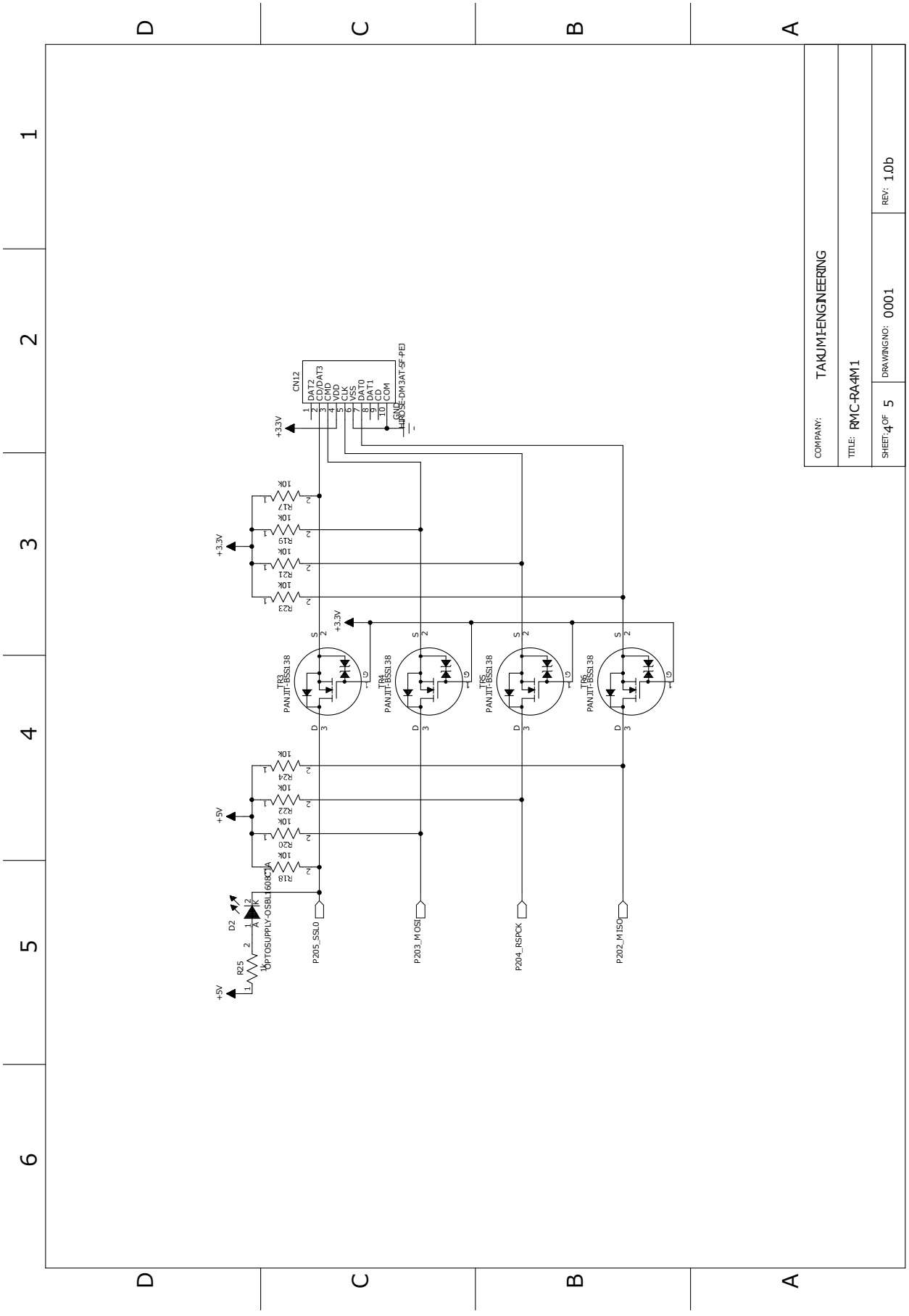
※RMC-RAM1 Rev.1.0b 基板のピン配置です。Rev.2.0 とは全く異なります。

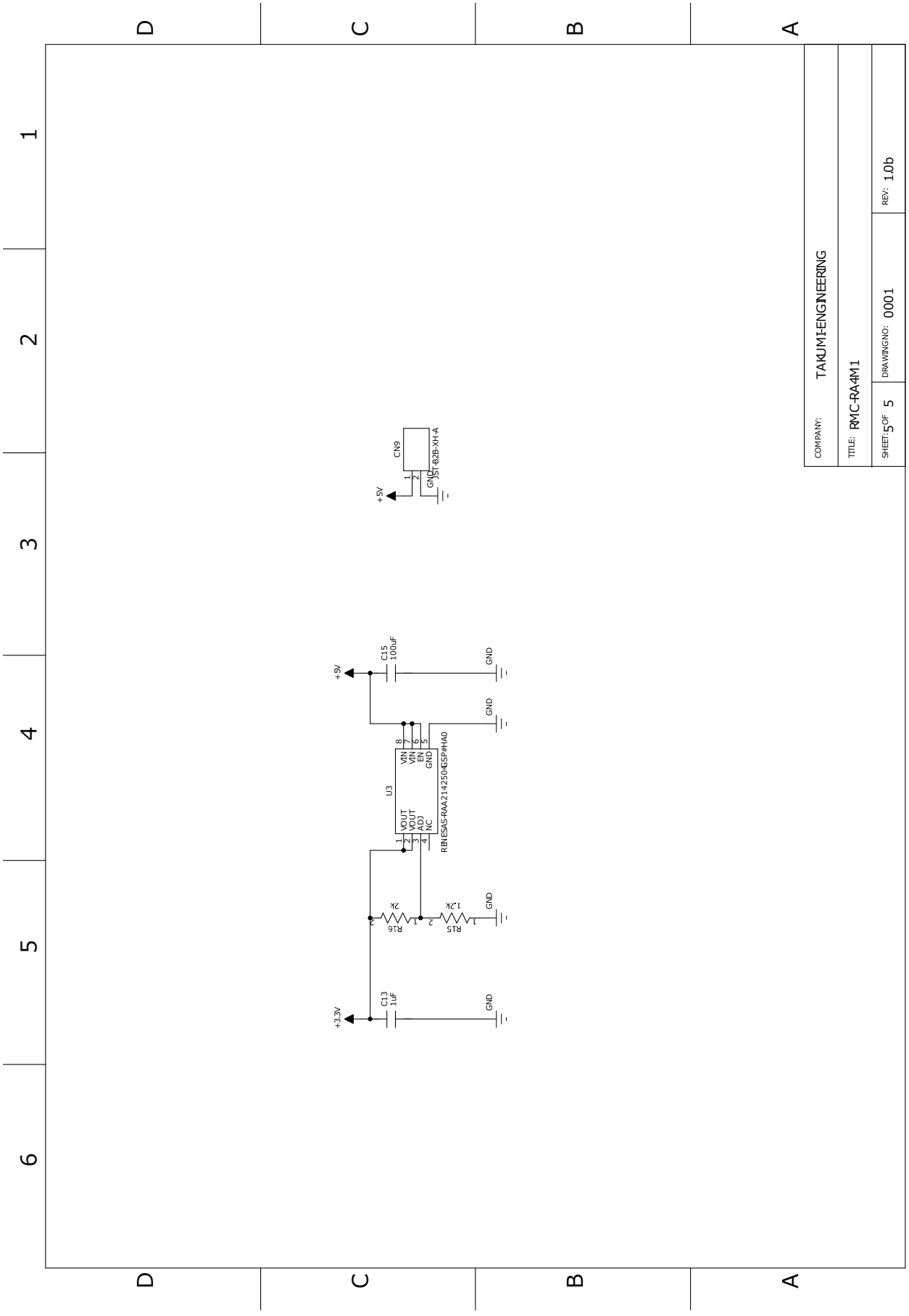
2.4. 回路図


Arduino互換コネクタ
CN1
CN2
CN11
CN10
CN4
CN5
CN8
CN7
HO1
HO3
HO4
HO2
LOGO1
RENESAS-LOGO-200X3.1-005
COMPANY: TAKUMI-ENGINEERING
TITLE: RMC-RA4M1
SHEET: 1 OF 5
DRAWING NO: 0001
REV: 1.0b


COMPANY: TAKUMI-ENGINEERING
TITLE: RMC-RA4M1
SHEET: 2 OF 5
DRAWING NO: 0001
REV: 1.0b
U2
RENESAS-R7FA4M1AB3CFP
C4 0.1uF
C5 0.1uF
C12 0.1uF
C14 0.1uF
C8 0.1uF
C9 0.1uF
C6 4.7uF
C7 1uF
C3 ??
C2 ??
X1 ??
+5V
GND

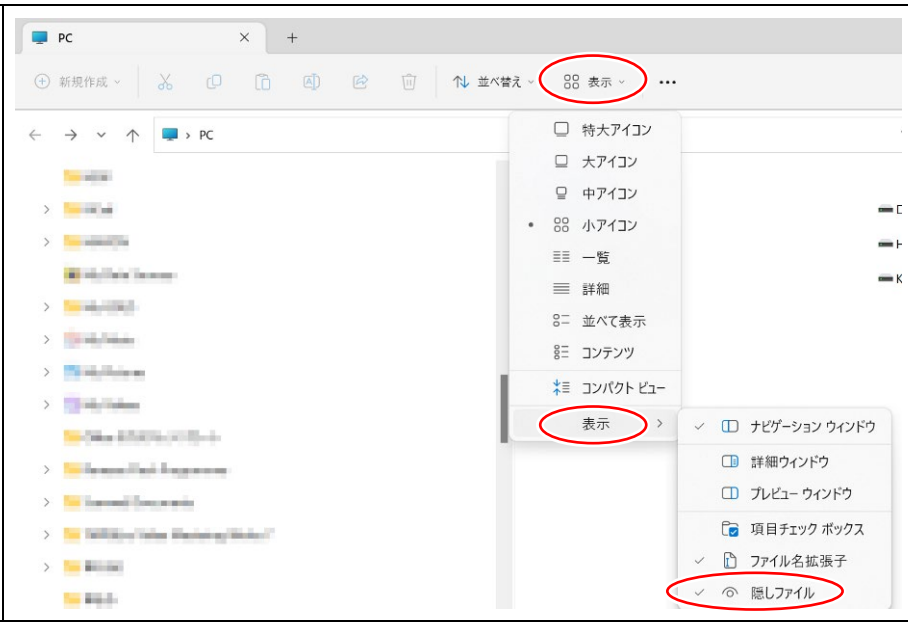

NodeMCU回路
COMPANY: TAKUMI-ENGINEERING
TITLE: RMC-RA4M1
SHEET: 3 OF 5
DRAWINGNO: 0001
REV: 1.0b
U1
WCH-CH340K
CN6
CN3
NELTRON-507KCR-16SMC2-BK-TR
BOURNS-MF-FSMF050X
PANJIT-SSZ040RL
P109_TXD9_D11_MOSI
P110_RXD9_D12_MISO
P407_USB_VBUS_SDA
P915_USB_DM
P914_USB_DP
P305_LED
P307_DIPSW1
P306_DIPSW2
DTR#
RTS#
RES
P201_MD
GND
+5V


COMPANY: TAKUMI-ENGINEERING
TITLE: RMC-RA4M1
SHEET: 4 OF 5
DRAWING NO: 0001
REV: 1.0b
CN12
DAT2
CD/DAT3
CMD
VDD
CLK
VSS
DAT0
DAT1
CD
COM
GND
HRSE-DM3AT-SF-PEJ
+3.3V
+5V
R27 10K
R19 10K
R21 10K
R23 10K
R24 10K
R22 10K
R20 10K
R18 10K
TR3 PANJIT-BSS138
TR4 PANJIT-BSS138
TR5 PANJIT-BSS138
TR6 PANJIT-BSS138
D2
OPTOSUPPLY-OSBL1608C-UA
R25 1K
P205_SSL0
P203_MOSI
P204_RSPCK
P202_MISO

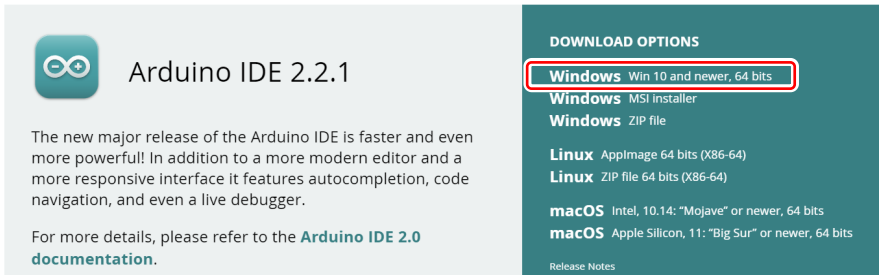

COMPANY: TAKUMI-ENGINEERING
TITLE: RMC-RA4M1
SHEET: 5 OF 5
DRAWING NO: 0001
REV: 1.0b
CN9
SM2B-SHF-1AA
GND
+5V
C15
100uF
U3
RENESAS-RAA214250-GSP#HA0
VIN
VIN
EN
GND
VOUT
VOUT
ADJ
NC
R16
2k
R15
1.2k
C13
1uF
+3.3V

3. 開発環境を整える

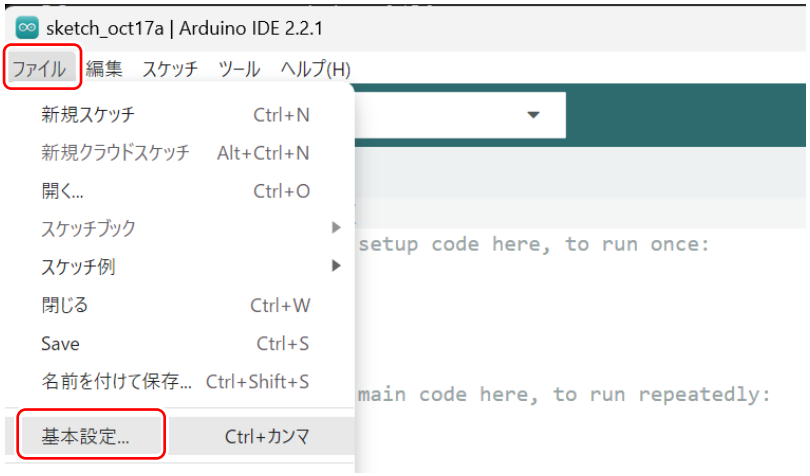
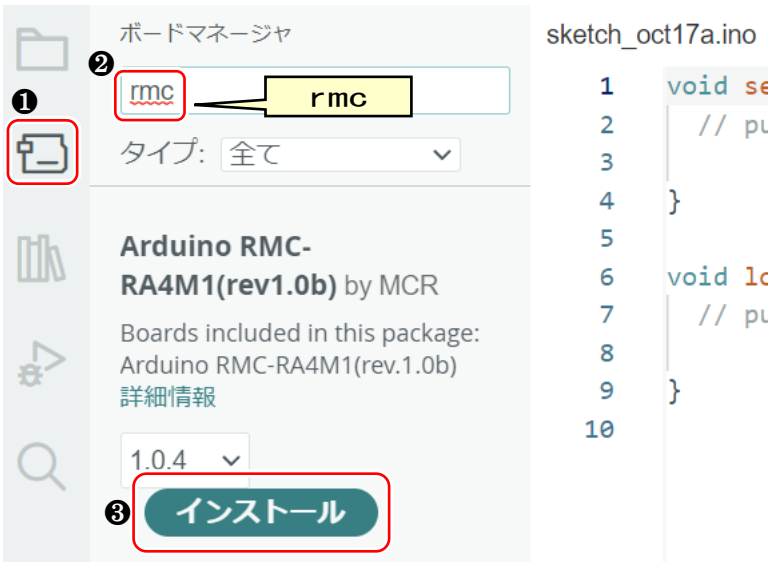
3.1. Windows の設定

1		これから、隠しフォルダ・ファイルを操作するので、隠しフォルダ・ファイルが見られるように設定しておきます。 デスクトップの PC アイコンをダブルクリックするか、Win＋E キーを押してエクスプローラーウィンドウを表示します。 「表示」タブ→「表示」→「隠しファイル」にチェックを入れます。

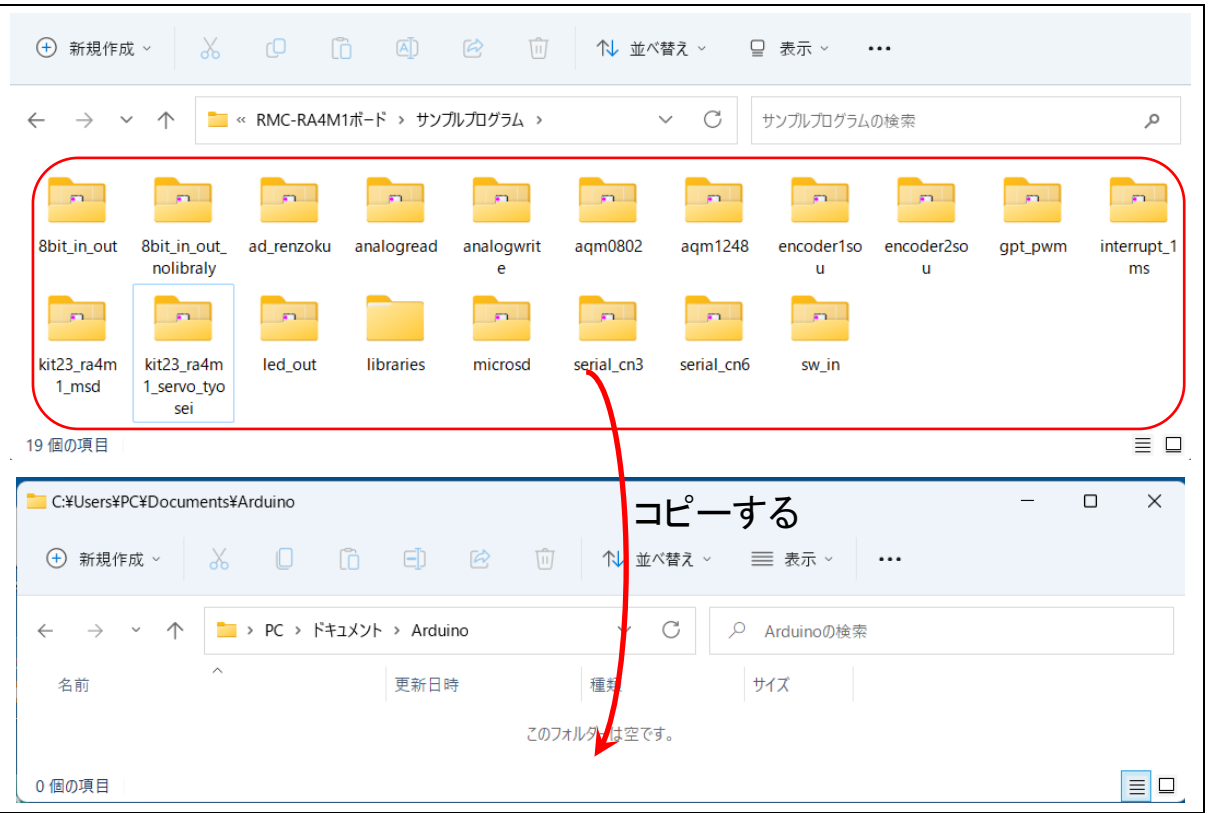
3.2. 開発環境のダウンロード

1		ブラウザで「Arduino IDE」と検索し、Arduino IDE ダウンロードサイト（https://www.arduino.cc/en/software）を開きます。Arduino IDE 2.x.x をダウンロード、インストールします。 ※Arduino IDE Ver.1.xx.xx には対応していません。

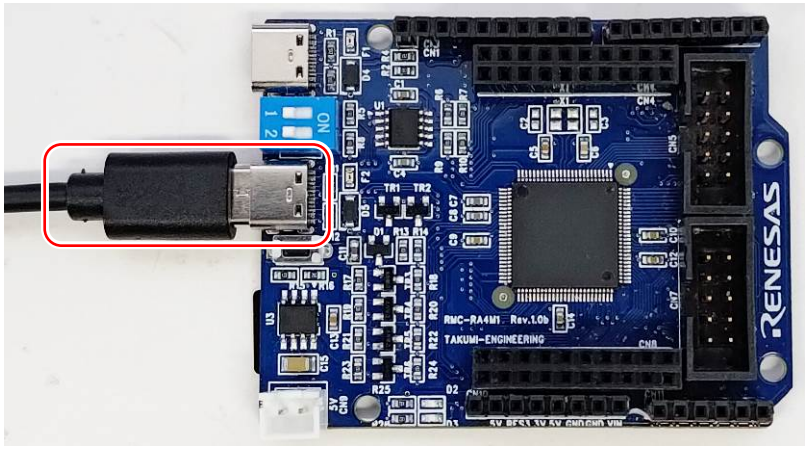
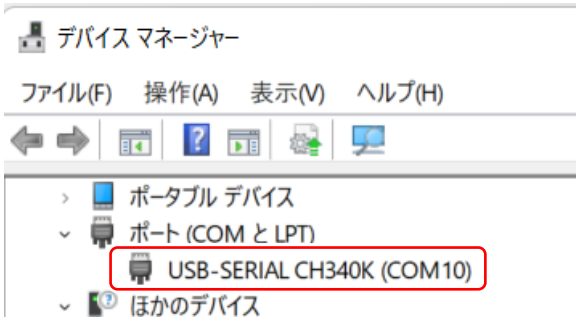
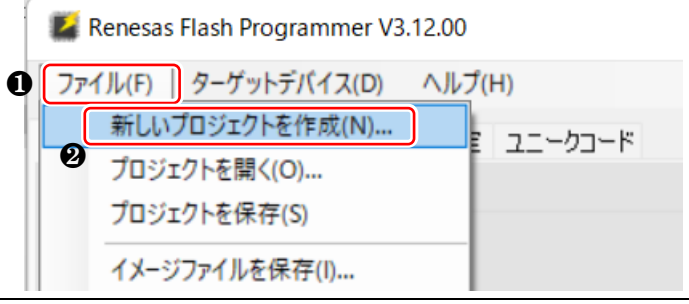
3.3. Arduino IDE に「RMC-RA4M1(rev.1.0b)」を追加する

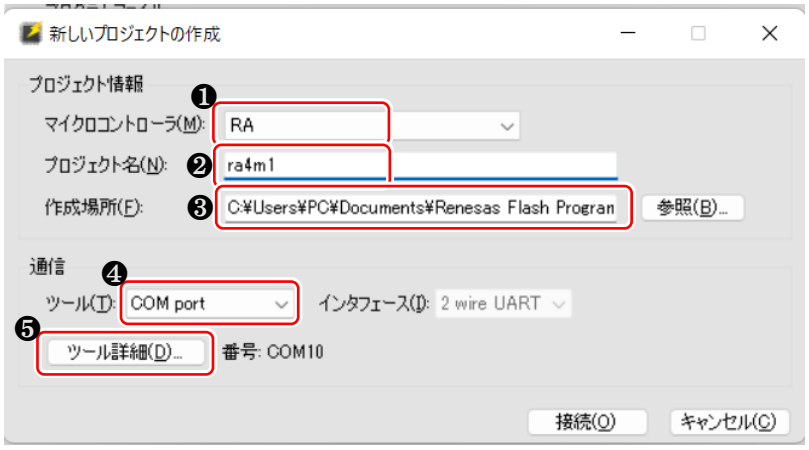
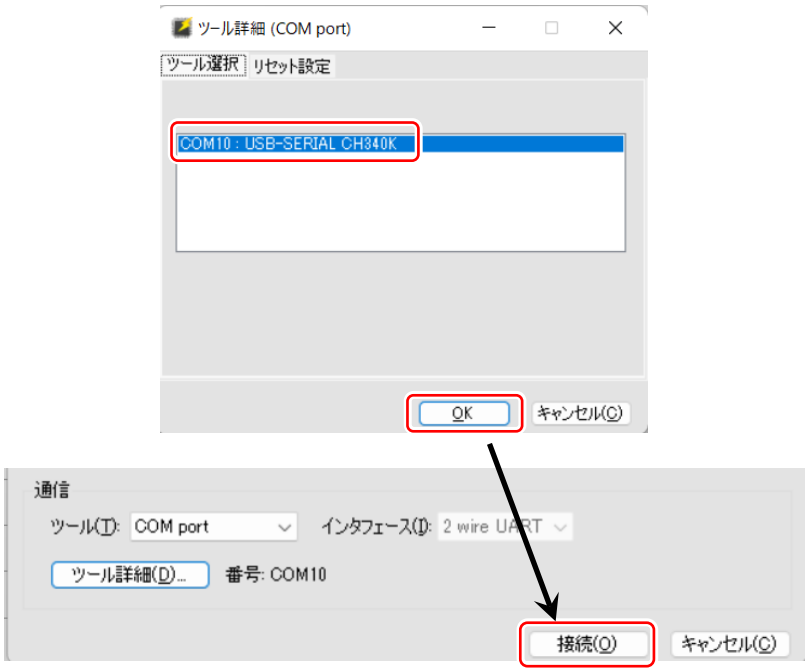
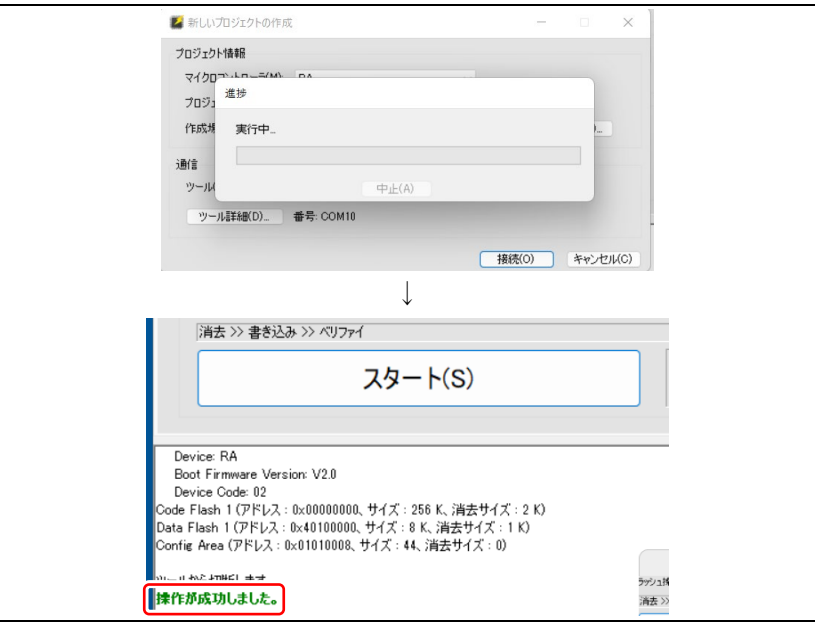
1		Arduino IDE を立ち上げます。 「ファイル→基本設定」を選択します。
2	 https://j-mcr.net/arduino_mcr/rmc_ra4m1_rev10b_install.json ※RMC-RA4M1(rev.2.0)は、URL が異なります。下記の URL になります。 https://j-mcr.net/arduino_mcr/rmc_ra4m1_rev20_install.json	「追加のボードマネージャの URL」に左のような URL を追加して、OK をクリックします。 ※URL の QR コード コピペして、ボードマネージャの URL に追加して下さい。
3		Arduino IDE を立ち上げます。 ❶左側の上から2個目の「  」をクリックします。 ❷検索欄に「rmc」と入力します。 ❸「Arduino RMC-RA4M1(rev1.0b)」の『インストール』ボタンをクリックして、インストールしてください。 ❹再度「  」をクリックして、ボードマネージャの表示を消します。

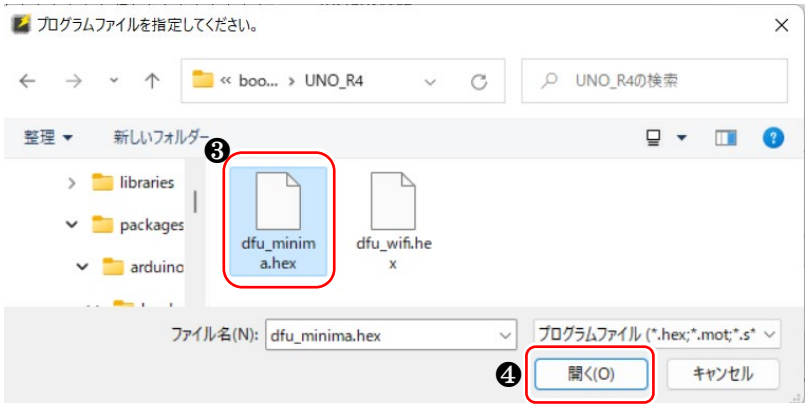
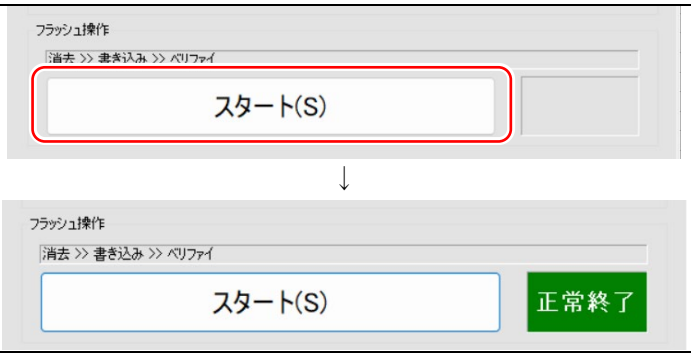
3.4. サンプルプログラムをコピーする

1	
	「RMC-RA4M1 ボード」→「サンプルプログラム」のフォルダのデータを、 マイドキュメントの「Arduino」フォルダへコピーします。

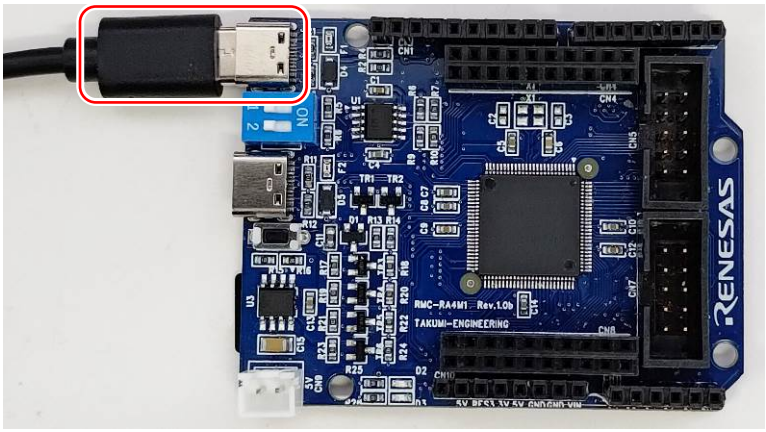
3.5. ファームウェアを書き込む

1		RMC-RA4M1 ボードの中心の USB コネクタと、パソコンを接続します。
2		デバイスマネージャーを開きます。 開き方が分からない場合は、Win＋Rキーのファイル名を指定して実行で、 **mmc devmgmt.msc** (mmc の次の文字はスペース) と入力して実行してください。 「ポート(COM と LPT)」部分に、 「**USB-SERIAL CH340K (COM●)**」 と表示されているか確認してください。 ●部分は 1～255 の番号です。 ！マークなどが出て、ドライバーが認識されていない場合は、「CH340K ドライバー」などで検索し、ドライバーをインストールしてください。
3		ブラウザで「renesas rfp」と検索し、Renesas Flash Programmer のサイト (https://www.renesas.com/jp/ja/software-tool/renesas-flash-programmer-programming-gui)を開きます。 「Renesas Flash Programmer V3.12.00 Windows」をダウンロード、インストールします。 ※「V3.12.00」の波線部分は、異なることがあります。
4		スタート→「Renesas Electronics Utilities」→「Renesas Flash Programmer V3」を立ち上げます。 ❶ファイル ❷新しいプロジェクトを作成 を選択します。

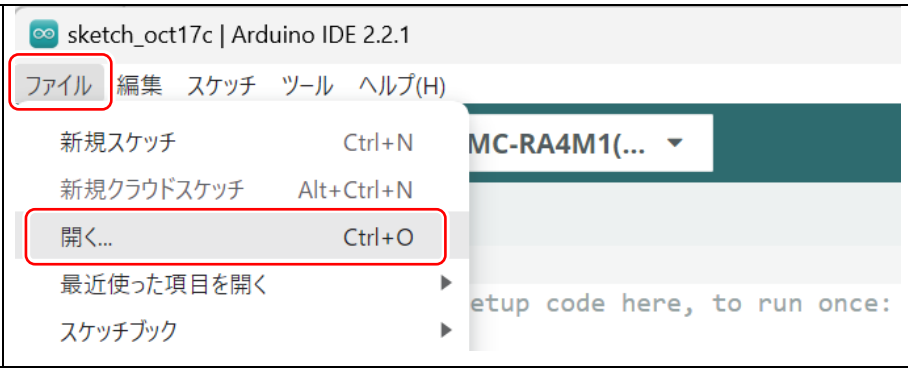
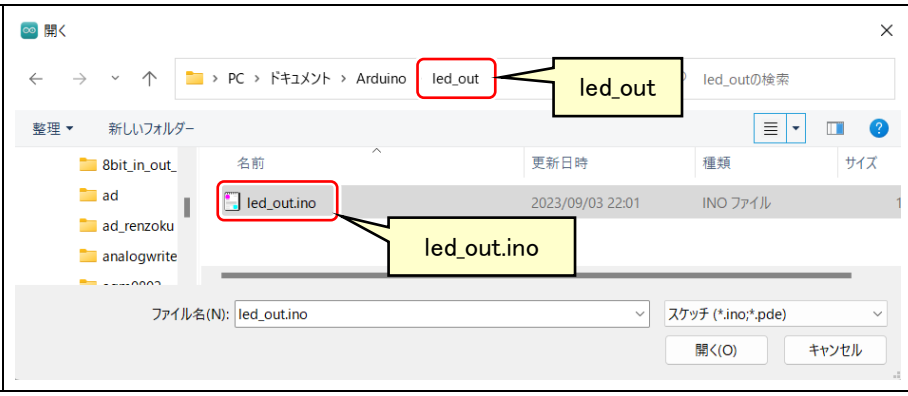
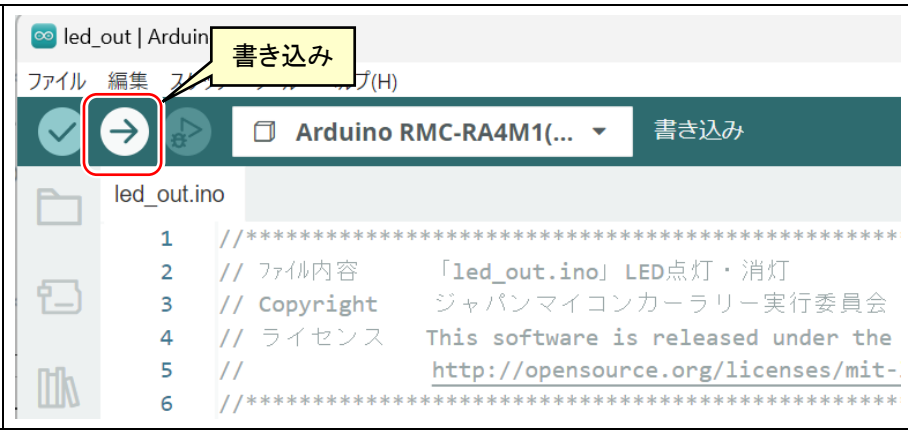
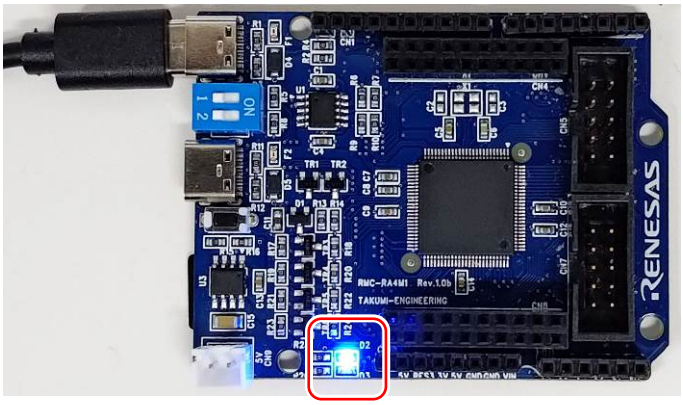
5		左のように入力、設定します。 ❶マイクロコントローラ「RA」 ❷プロジェクト名「ra4m1」など ※プロジェクト名は、任意の文字で構いません。 ❸作成場所：**変更しません** ❹ツール「COM port」 ❺「ツールの詳細」をクリックします。
6		ポート選択で、 「COM● USB-SERIAL CH340K」を選択し、「OK」をクリックします。 「接続」をクリックします。
7		「実行中」と表示され「接続が成功しました。」と表示されます。 ●通信に失敗した場合 ・「実行中」が数秒続いて、失敗する場合、「実行中」が表示されている間に、RMC-RA4M1ボードのリセットスイッチを押してみてください。接続されることがあります。 ・すぐにエラーが出る場合は、CH340K の COM ポートが正しく接続されていません。ドライバーを入れ直すなど、確認してください。

8	 ↓ ❷C:¥Users¥【ユーザー名】¥AppData¥Local¥Arduino15¥packages¥rmc_ra4m1_10b¥hardware¥ra4m1_10b¥1.x.x¥bootloaders¥UNO_R4 フォルダを開く ↓ 	❶参照をクリックします。 ❷左のフォルダを開きます。 ※「【ユーザー名】」は、Windows のユーザー名を選択します。 ※「1.x.x」は、インストールしたバージョンを開きます。 ❸「dfu_minima.hex」を選択します。 ❹「開く」をクリックします。
9		「スタート」をクリックします。 ブートローダーが書き込まれ、「正常終了」と表示されれば、完了です。 書き込みがうまくいかないときは、「実行中」と表示されているときに RMC-RA4M1 ボードのリセットボタンを押してみてください。

3.6. Arduino IDE でプログラムを書き込む

1		RMC-RA4M1 ボードに USB コネクタを接続します。 USB ケーブルは、基板角のコネクタ(左写真の位置)に差し込みます。 ArduinoIDE を使うときは、このコネクタになります。
2		❶「▼」をクリックします ❷「他のボードとポートを選択」を選択します。 ❸「Arduino RMC-RA4M1 (rev.1.0b)」を選択します。 ❹「USB DFU」を選択します。番号(左図では 5-1.3.1)は何番でも構いません。 最後に「OK」をクリックします。

※DFU は、Device Firm ware Upgrade のことで、USB でプログラムを書き込むことができる通信規格のことです。

3		「ファイル」→「開く」を選択します。
4		「led_out」フォルダの「led_out.ino」を開きます。
5		書き込みボタンをクリックして、プログラムを書き込みます。
6		写真□部分の 2 個の LED が交互に点灯すれば、書き込み成功です。

※ファームウェアを書き込んでからArduinoIDE でプログラムを書き込むと、ポートが「USB DFU」から「USB シリアルデバイス(COM○)」になり、TeraTerm や ArduinoIDE のシリアルモニタなどで、パソコンと RMC-RA4M1 ボードでシリアル通信することができます。

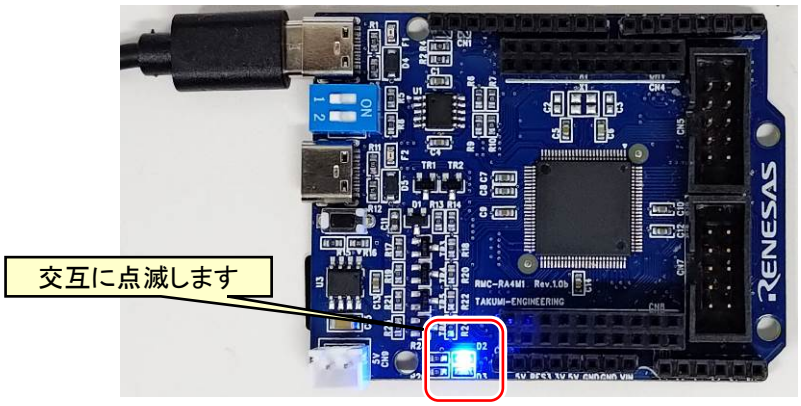
4. 演習

4.1. 演習 1　マイコンボード上の LED の点滅「led_out.ino」

マイコンボード上の LED 2 個を点滅させます。

4.1.1. 配線

特にありません。マイコンボード上の LED を点灯させます。



4.1.2. プログラム

```
 1 :  //********************************************************************************
 2 :  // ﾌｧｲﾙ内容       「led_out.ino」LED 点灯・消灯
 3 :  // Copyright     ジャパンマイコンカーラリー実行委員会
 4 :  // ライセンス    This software is released under the MIT License.
 5 :  //               http://opensource.org/licenses/mit-license.php
 6 :  //********************************************************************************
 7 :
 8 :  void setup() {
 9 :    pinMode( 23, OUTPUT ); // LED D2
10 :    pinMode( 24, OUTPUT ); // LED D3
11 :  }
12 :
13 :  void loop() {
14 :    digitalWrite( 23, 1 );  // 1 で消灯
15 :    digitalWrite( 24, 0 );  // 0 で点灯
16 :    delay( 500 );
17 :
18 :    digitalWrite( 23, 0 );
19 :    digitalWrite( 24, 1 );
20 :    delay( 500 );
21 :  }
```

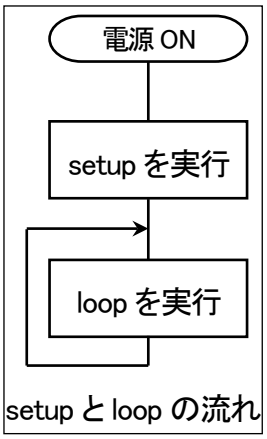
pinMode 関数は、端子を入力にするか、出力にするか設定する関数です。23 番ピンと 24 番ピンの LED 端子を出力(OUTPUT)に設定しています。

digitalWrite 関数は、端子から電圧を出力する命令です。23 番ピンの LED へ"1"(5V)を出力、24 番ピンの LED へ"0"(0V)を出力しています。
23 番ピンと 24 番ピンの LED は､5V を加えると消灯、0V を加えると点灯します。

delay 関数は、時間稼ぎをする関数で ms 単位で設定します。今回は 500ms(0.5 秒)、この行で一時停止し、500ms 経つと次の行へ行きます。

4.1.3. プログラムの解説

Arduino のプログラムは、setup 関数と loop 関数があります。
setup 関数はマイコンの動作を開始するときに、最初の1回だけ実行される関数です。端子の入出力設定や、接続している機器の初期化を行います。
loop 関数は setup 関数を実行後、繰り返し実行する関数です。
流れ図を右図に示します。



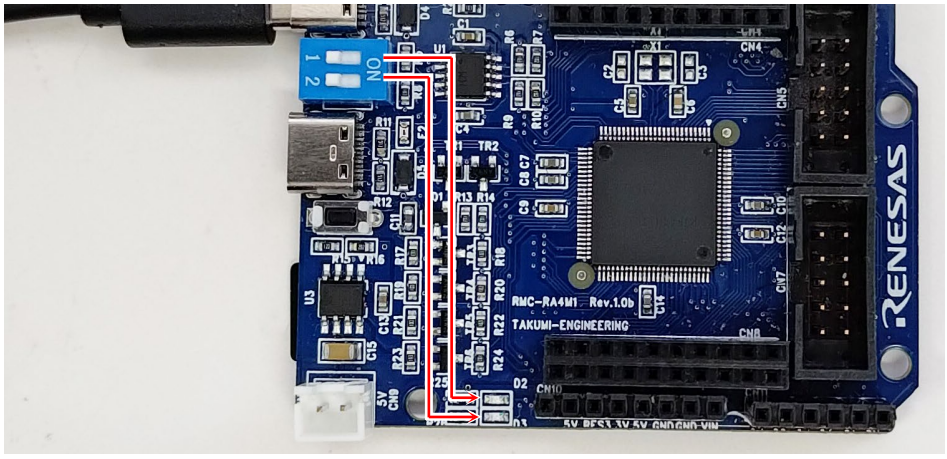
setup と loop の流れ

4.2. 演習 2　マイコンボード上のディップスイッチの入力「sw_in.ino」

マイコンボード上のディップスイッチの値を取得して、LED へ出力します。

4.2.1. 配線

特にありません。マイコンボード上のディップスイッチが ON になると、LED を点灯させます。



4.2.2. プログラム

```
1 :  //********************************************************************************
2 :  // ﾌｧｲﾙ内容       「sw_in.ino」ディップスイッチの値　取得
3 :  // Copyright     ジャパンマイコンカーラリー実行委員会
4 :  // ライセンス    This software is released under the MIT License.
5 :  //               http://opensource.org/licenses/mit-license.php
6 :  //********************************************************************************
7 :
8 :  void setup() {
9 :    pinMode( 23, OUTPUT ); // LED D2
10 :   pinMode( 24, OUTPUT ); // LED D3
11 :   pinMode( 25, INPUT );  // ディップスイッチ 1
12 :   pinMode( 26, INPUT );  // ディップスイッチ 2
13 :  }
14 :
15 :  void loop() {
16 :    int sw1, sw2;
17 :
18 :    sw1 = digitalRead( 25 );  // ON 側で 0 OFF 側で 1
19 :    sw2 = digitalRead( 26 );
20 :
21 :    digitalWrite( 23, sw1 );  // 0 で点灯 1 で消灯
22 :    digitalWrite( 24, sw2 );
23 :  }
```

ディップスイッチは 25 番ピンと 26 番ピンに接続されていて、pinMode で INPUT にします。

ディップスイッチは ON 側（基板内側）で"0"(0V)、逆側（基板外側）で"1"(5V)になります。sw1 変数と sw2 変数に代入します。

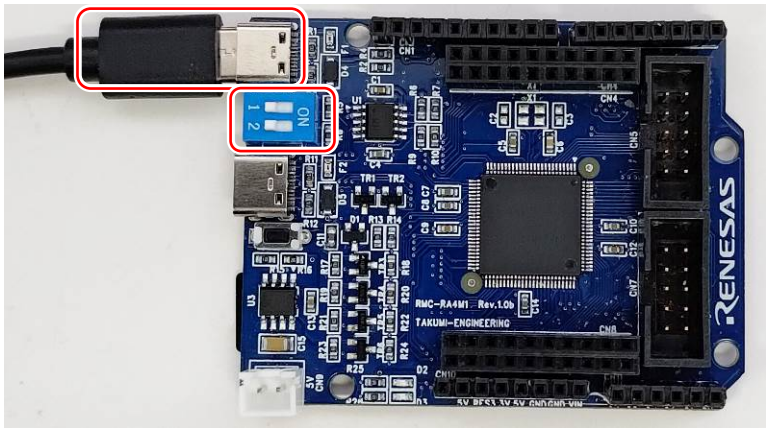
sw1 変数と sw2 変数の値を LED へ出力します。LED は"0"で点灯するので、ディップスイッチを ON 側にすると LED が点灯することになります。

4.3. 演習 3 シリアル通信（CN3 使用(ArduinoIDE 書き込み用)）「serial_cn3.ino」

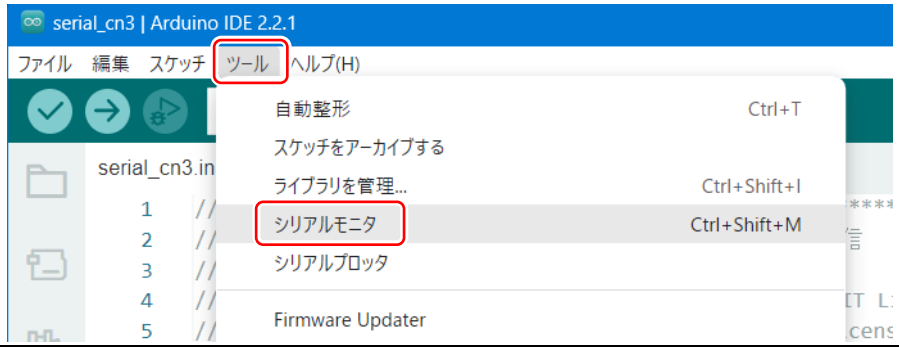
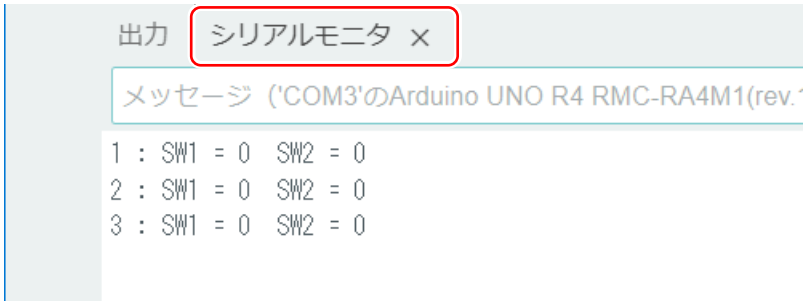
USB コネクタの CN3（端にある USB コネクタ)からシリアル出力して、ディップスイッチの状態をパソコンなどのシリアル通信ソフトに表示します。

4.3.1. 配線

特にありません。マイコボード上の CN3 とパソコンを接続します。
シリアル通信を行いますので、USB ケーブルを接続して、シリアルモニタを表示させておきます。



4.3.1. シリアルモニタを表示させる

1		Arduino IDE の「ツール」→「シリアルモニタ」を選択します。
2		Arduino IDE の下部に「シリアルモニタ」が表示されるので、選択します。 マイコンからシリアル通信で送られてくるデータが表示されます。 シリアルモニタの表示が不要になったら、「シリアルモニタ ×」の×マークをクリックして、シリアルモニタを閉じます。

4.3.2. プログラム

```
 1 :  //*****************************************************************************
 2 :  // ﾌｧｲﾙ内容      「serial_cn3.ino」CN3 の USB とシリアル通信
 3 :  // Copyright    ジャパンマイコンカーラリー実行委員会
 4 :  // ライセンス   This software is released under the MIT License.
 5 :  //              http://opensource.org/licenses/mit-license.php
 6 :  //*****************************************************************************
 7 :
 8 :  void setup() {
 9 :    pinMode( 25, INPUT );  // ディップスイッチ 1
10 :    pinMode( 26, INPUT );  // ディップスイッチ 2
11 :
12 :    Serial.begin( 9600 );
13 :
14 :    while ( !Serial ) {
15 :      // 接続されるまで待つ
16 :    }
17 :
18 :    Serial.println( "¥n¥nCN3 シリアル通信");
19 :  }
20 :
21 :  void loop() {
22 :    static int lp = 1;
23 :    int sw1, sw2;
24 :
25 :    sw1 = digitalRead( 25 );  // ON 側で 0 OFF 側で 1
26 :    sw2 = digitalRead( 26 );
27 :
28 :    Serial.print( lp );
29 :    Serial.print( " : " );
30 :    Serial.print( "SW1 = " );
31 :    Serial.print( sw1 );
32 :    Serial.print( "  SW2 = " );
33 :    Serial.println( sw2 );
34 :    delay( 1000 );
35 :    lp++;
36 :  }
```

Serial.begin(通信速度)で、USB の CN3 とパソコンを接続して、シリアル通信を行うことができます。通信速度は、9600bps に設定します。
シリアル通信ソフトの多くが、初期通信速度が 9600bps なので、9600bps にしておきます。

接続されるまで while ループで待ちます。通信が接続されたらループを抜けて次に行きます。

println で、シリアルポートへ出力します。今回は、「CN3 シリアル通信」という文字を出力します。println は、出力した後改行します。ただの print は改行しません。
「¥n」は改行命令で、「¥n」が出てきた部分で改行します。

ローカル変数は、関数が終了すると値が破棄されます。よって、lp 変数は毎回 1 になりますが、static を付けることによって、破棄せずに保持しておきます。loop 関数を繰り返すごとに lp 変数は 35 行で＋１します。
sw1 変数、sw2 変数は毎回、値が破棄されますが 25 行目、26 行目で毎回値を更新しているので、破棄されても問題ありません。

Serial.print(変数)とすると、変数の値をシリアル出力します。
Serial.print("文字列")とすると、文字列をそのまま表示します。
33 行は、Serial.println なので、改行します。
シリアルモニタへの出力例を下記に示します。

```
CN3 シリアル通信
1 : SW1 = 0  SW2 = 0
2 : SW1 = 1  SW2 = 0
3 : SW1 = 0  SW2 = 1
中略
10 : SW1 = 1  SW2 = 0
11 : SW1 = 1  SW2 = 1
```

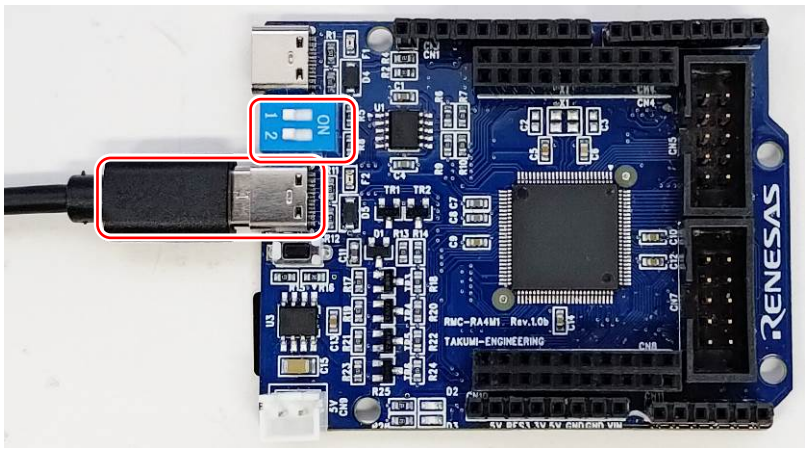
delay(1000)で 1000ms(１秒) 待って、lp 変数を＋１して、loop 関数を終わります。loop 関数は繰り返し実行されるので、lp 変数は loop 関数を繰り返すごとに＋１されます。

4.4. 演習 4 シリアル通信（CN6 使用(Renesas Flash Programmer 書き込み用)）「serial_cn6.ino」

USB コネクタの CN6（中にある USB コネクタ)からシリアル出力して、ディップスイッチの状態をパソコンなどのシリアル通信ソフトに表示します。

4.4.1. 配線

特にありません。マイコンボード上の CN6 とパソコンを接続します。
シリアル通信を行いますので、USB ケーブルを接続して、シリアルモニタを表示させておきます。



4.4.2. プログラム

```
 1 :  //*****************************************************************************
 2 :  // ﾌｧｲﾙ内容      「serial_cn6.ino」CN6 の USB とシリアル通信
 3 :  // Copyright    ジャパンマイコンカーラリー実行委員会
 4 :  // ライセンス   This software is released under the MIT License.
 5 :  //              http://opensource.org/licenses/mit-license.php
 6 :  //*****************************************************************************
 7 :
 8 :  void setup() {
 9 :    pinMode( 25, INPUT );  // ディップスイッチ 1
10 :    pinMode( 26, INPUT );  // ディップスイッチ 2
11 :
12 :    Serial1.begin( 9600 );  // Serial1 = CN6 の USB コネクタとシリアル接続
13 :
14 :    while ( !Serial1 ) {
15 :      // 接続されるまで待つ
16 :    }
17 :
18 :    Serial1.println( "¥n¥nCN6 シリアル通信");
19 :  }
20 :
21 :  void loop() {
22 :    static int lp = 1;
23 :    int sw1, sw2;
24 :
25 :    sw1 = digitalRead( 25 );  // ON 側で 0 OFF 側で 1
26 :    sw2 = digitalRead( 26 );
27 :
28 :    Serial1.print( lp );
29 :    Serial1.print( " : " );
30 :    Serial1.print( "SW1 = " );
31 :    Serial1.print( sw1 );
32 :    Serial1.print( "  SW2 = " );
33 :    Serial1.println( sw2 );
34 :    delay( 1000 );
35 :    lp++;
36 :  }
```

Serial1 にすると、CN6（基板の中側）の USB コネクタから、シリアル出力します。
プログラムの動作は、「serial_cn3.ino」と同じです。

※「Serial」と「Serial1」の併用について

「Serial」と「Serial1」は併用して問題ありません。併用すると、USB ケーブル 2 本を RMC-RA4M1 ボードとパソコン間に接続することになります。通信ソフト(シリアルモニタ)も2つ起動させて、別々に通信することができます。
プログラム例を下記に示します。

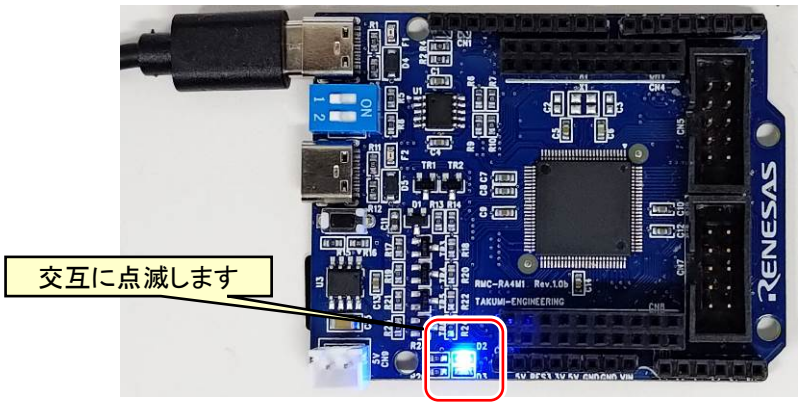
```
void setup() {
  Serial.begin( 9600 );
  Serial1.begin( 9600 );
}

void loop() {
  static int i = 1;
  Serial.println( i );    // Serial に出力
  Serial1.println( i );   // Serial1 に出力
  i++;
  delay( 1000 );
}
```

4.5. 演習 5　1ms ごとの割り込み処理「interrupt_1ms.ino」

loop 関数を実行しながら、1ms ごとに割り込みを発生させます。

4.5.1. 配線

特にありません。マイコンボード上の LED を点灯させます。



4.5.1. プログラム

```
 1 :  //**************************************************************************
 2 :  // ﾌｧｲﾙ内容      「interrupt_1ms.ino」1ms ごとの割り込み
 3 :  // Copyright    ジャパンマイコンカーラリー実行委員会
 4 :  // ライセンス   This software is released under the MIT License.
 5 :  //              http://opensource.org/licenses/mit-license.php
 6 :  //**************************************************************************
 7 :
 8 :  //  インクルード宣言
 9 :  #include "FspTimer.h"
10 :
11 :  // プロトタイプ宣言
12 :  void AGTCallback(timer_callback_args_t __attribute((unused)) * p_args);
13 :
14 :  // グローバル変数の宣言
15 :  FspTimer fsp_timer;       // 割り込み関係
16 :  uint32_t cnt;
17 :
18 :  void setup() {
19 :    pinMode( 23, OUTPUT ); // LED D2
20 :    pinMode( 24, OUTPUT ); // LED D3
21 :
22 :    // AGT 1ms ごとの割り込み処理の設定 PCLKB=24MHz ∴TIMER_SOURCE_DIV_1(1 分周)なら、1/(24e6*1) * 24000 = 1ms
23 :    fsp_timer.begin(TIMER_MODE_PERIODIC, AGT_TIMER, 1, 23999, 1,
                                                  (timer_source_div_t)TIMER_SOURCE_DIV_1, AGTCallback);
24 :    IRQManager::getInstance().addPeripheral(IRQ_AGT, (void*)fsp_timer.get_cfg());
25 :    fsp_timer.open();
26 :  }
27 :
28 :  void loop() {
29 :    if( cnt >= 1000 ) {
30 :      cnt = 0;
31 :    }
32 :    if( cnt < 500 ) {
33 :      digitalWrite( 23, 1 );
34 :      digitalWrite( 24, 0 );
35 :    } else {
36 :      digitalWrite( 23, 0 );
37 :      digitalWrite( 24, 1 );
38 :    }
39 :  }
40 :
41 :  // 割り込み処理  1ms ごとに実行
42 :  void AGTCallback(timer_callback_args_t __attribute((unused)) * p_args) {
43 :    cnt++;
44 :  }
```

RA4M1 マイコンに AGT(The Asynchronous General Purpose Timer:非同期汎用タイマ) という機能があり、その機能で 1ms ごとに割込を発生させます。
AGT の設定を行うライブラリをインクルードします。

1ms ごとに実行される関数を、プロトタイプ宣言します。

FspTimer クラスで fsp_timer インスタンスを作成します。

23～25 行が、AGT で 1ms ごとの割込を発生させるプログラム部分です。□で囲った２カ所を変更することによって、割り込み周期を変更させることができます。

cnt 変数は、割り込みプログラムで＋１しているので、1ms ごとに＋１していく変数です。
1000 以上ということは、1000ms たったら、という意味になります。

cnt 変数が 0～499 なら 33 行、34 行を実行、
それ以外(cnt 変数が 500～999)なら、36 行、37 行を実行します。

割り込みプログラムです。
1ms に１回、実行されます。

4.5.2. プログラムの解説

FspTimer クラスの begin で、割り込み周期と割り込み関数名を設定します。

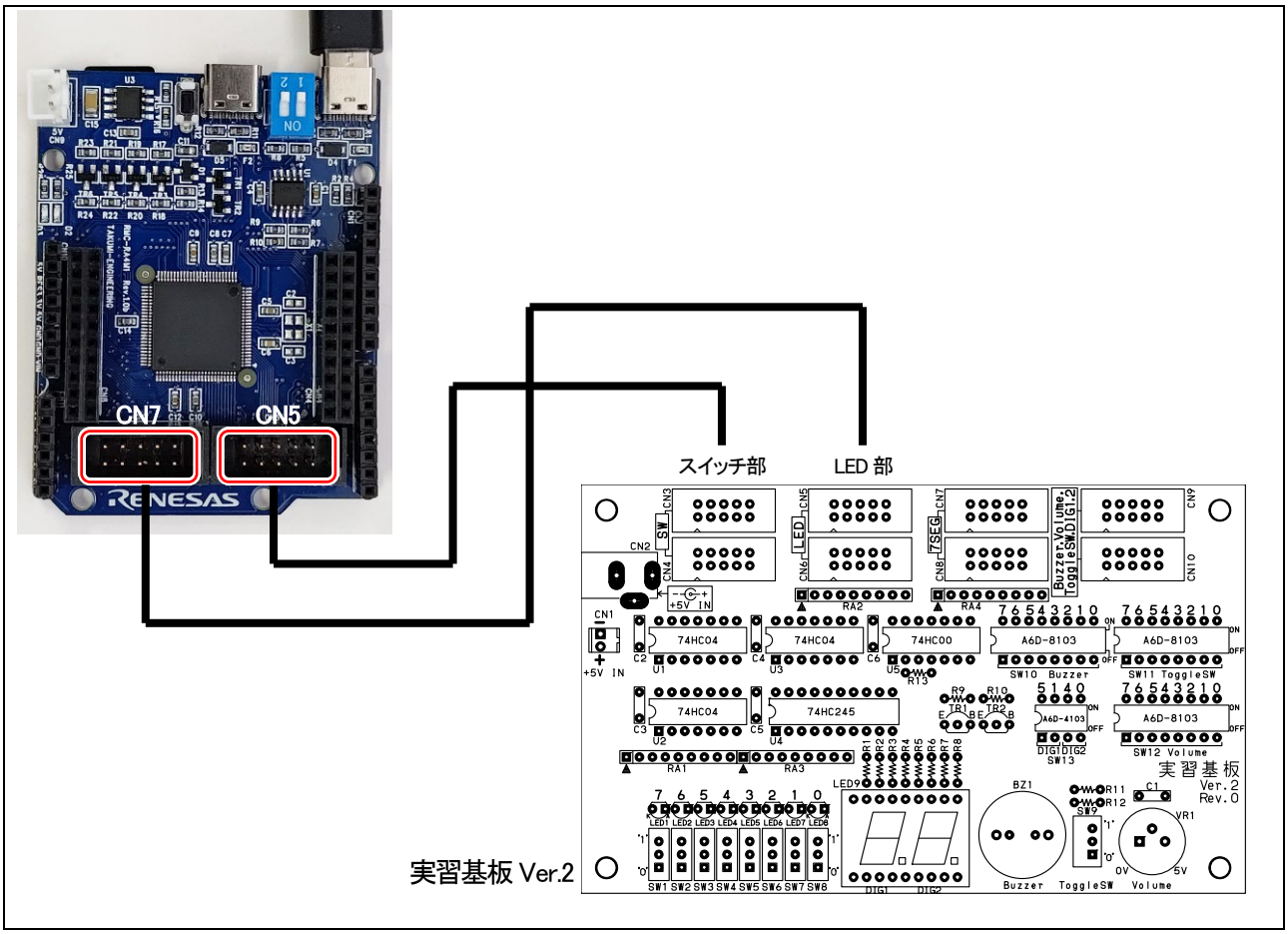
使い方	FspTimer :: begin(TIMER_MODE_PERIODIC, AGT_TIMER, 1, 値, 1, (timer_source_div_t)分周比, 割り込み関数名);
分周比	TIMER_SOURCE_DIV_1 : カウンタの周波数を 1 分周で使用 TIMER_SOURCE_DIV_2 : カウンタの周波数を 2 分周で使用 TIMER_SOURCE_DIV_8 : カウンタの周波数を 8 分周で使用
値	0～65,535
割り込み関数名	割り込み関数の関数名を設定します。
計算方法	割り込み周期の計算方法を示します。 カウンタは 24MHz で動作しています。割り込み周期は、 割り込み周期＝1/(24MHz÷分周比)×値 となります。それぞれの分周比で設定できる最大の割り込み周期は、 １分周で設定できる最大割り込み周期：1/（24MHz÷1 分周）× 65535 ＝ 2.730625ms ２分周で設定できる最大割り込み周期：1/（24MHz÷2 分周）× 65535 ＝ 5.461250ms ８分周で設定できる最大割り込み周期：1/（24MHz÷8 分周）× 65535 ＝ 21.845000ms となります。例えば、割り込み周期が 1ms であれば 1 分周、4ms であれば 2 分周、20ms であれば 8 分周にします。21.845000ms を超える割り込み周期の設定はできません。 値の計算方法は、 値 ＝ 設定したい割り込み周期 ÷ { 1/(24MHz÷分周比) } で計算できます。 例 1： 1ms にしたいとき … 値 ＝ 1ms ÷ { (1/(24MHz÷1 分周) } ＝ 24,000 例 2： 5ms にしたいとき … 値 ＝ 5ms ÷ { (1/(24MHz÷2 分周) } ＝ 60,000 例 3：10ms にしたいとき … 値 ＝ 10ms ÷ { (1/(24MHz÷8 分周) } ＝ 30,000 ※プログラムで設定するときは 1 小さい値を設定します。 例）24000→23999 60000→59999
使用例	・例１のとき fsp_timer.begin(TIMER_MODE_PERIODIC, AGT_TIMER, 1, 23999, 1, (timer_source_div_t) TIMER_SOURCE_DIV_1, AGTCallback); ・例２のとき fsp_timer.begin(TIMER_MODE_PERIODIC, AGT_TIMER, 1, 59999, 1, (timer_source_div_t) TIMER_SOURCE_DIV_2, AGTCallback); ・例３のとき fsp_timer.begin(TIMER_MODE_PERIODIC, AGT_TIMER, 1, 29999, 1, (timer_source_div_t) TIMER_SOURCE_DIV_8, AGTCallback);
備考	AGT は、PCLKB というクロックで動きます。 メインの動作周波数は 48MHz ですが PCLKB はメイン周波数の２分周で動作するという設定になっているので、AGT は 48MHz の 1/2 の 24MHz で動作します。

4.6. 演習 6　10ピンコネクタを使用したデータの入出力「8bit_in_out.ino」

マイコンボードの CN5 に接続されているディップスイッチ 8bit を読み込んで、マイコンボードの CN7 に接続されている LED 8bit へ出力します。

4.6.1. 配線

マイコンボードの CN5 と実習基板 Ver.2 のスイッチ部、マイコンボードの CN7 と実習基板 Ver.2 の LED 部をフラットケーブルで接続します。



4.6.2. プログラム

```
1 :  //********************************************************************************
2 :  // ﾌｧｲﾙ内容      「8bit_in_out.ino」8bit 単位で入力、出力
3 :  // Copyright    ジャパンマイコンカーラリー実行委員会
4 :  // ライセンス   This software is released under the MIT License.
5 :  //              http://opensource.org/licenses/mit-license.php
6 :  //********************************************************************************
7 :
8 :  // プロトタイプ宣言
9 :  unsigned char dipsw( void );
10 :  void led( unsigned char value );
11 :
```

```
12 :  void setup() {
13 :    // CN5 DIPSW 8bit
14 :    pinMode( 27, INPUT );  // D27 は入力のみ
15 :    pinMode( 28, INPUT );  // D28 は入力のみ
16 :    pinMode( 29, INPUT );
17 :    pinMode( 30, INPUT );
18 :    pinMode( 31, INPUT );
19 :    pinMode( 32, INPUT );
20 :    pinMode( 33, INPUT );
21 :    pinMode( 34, INPUT);
22 :
23 :    // CN7 LED 8bit
24 :    pinMode( 35, OUTPUT );
25 :    pinMode( 36, OUTPUT );
26 :    pinMode( 37, OUTPUT );
27 :    pinMode( 38, OUTPUT );
28 :    pinMode( 39, OUTPUT );
29 :    pinMode( 40, OUTPUT );
30 :    pinMode( 41, OUTPUT );
31 :    pinMode( 42, OUTPUT );
32 :  }
33 :
34 :  void loop() {
35 :    unsigned char sw;
36 :
37 :    sw = dipsw();
38 :    led( sw );
39 :  }
40 :
41 :  unsigned char dipsw( void ) {
42 :    unsigned char c;
43 :
44 :    c  = (digitalRead( 27 ) << 7);
45 :    c |= (digitalRead( 28 ) << 6);
46 :    c |= (digitalRead( 29 ) << 5);
47 :    c |= (digitalRead( 30 ) << 4);
48 :    c |= (digitalRead( 31 ) << 3);
49 :    c |= (digitalRead( 32 ) << 2);
50 :    c |= (digitalRead( 33 ) << 1);
51 :    c |= (digitalRead( 34 ) << 0);
52 :
53 :    return c;
54 :  }
55 :
56 :  void led( unsigned char value ) {
57 :
58 :    digitalWrite( 35, (value & 0x80) != 0 );
59 :    digitalWrite( 36, (value & 0x40) != 0 );
60 :    digitalWrite( 37, (value & 0x20) != 0 );
61 :    digitalWrite( 38, (value & 0x10) != 0 );
62 :    digitalWrite( 39, (value & 0x08) != 0 );
63 :    digitalWrite( 40, (value & 0x04) != 0 );
64 :    digitalWrite( 41, (value & 0x02) != 0 );
65 :    digitalWrite( 42, (value & 0x01) != 0 );
66 :  }
```

CN5 の 27 番ピン～34 番ピンは、ディップスイッチに接続するので、入力にします。
※27、28 番ピンは入力専用端子で、出力端子にはできません。

CN7 の 35 番ピン～42 番ピンは、LED に接続するので、出力にします。

CN5 の 8bit の情報を sw 変数へ代入し、
sw 変数の値を CN7 の 8bit へ出力します。

例えば、27～34 番ピンに”1”が入力されているとします。
・D27 が 1 なので、左に 7bit シフトすると 1000 0000 になります。
・D28 が 1 なので、左に 6bit シフトすると 0100 0000 になります。
・D29 が 1 なので、左に 5bit シフトすると 0010 0000 になります。
・D30 が 1 なので、左に 4bit シフトすると 0001 0000 になります。
・D31 が 1 なので、左に 3bit シフトすると 0000 1000 になります。
・D32 が 1 なので、左に 2bit シフトすると 0000 0100 になります。
・D33 が 1 なので、左に 1bit シフトすると 0000 0010 になります。
・D34 が 1 なので、左に 0bit シフトすると 0000 0001 になります。
それらをすべて OR すると、「1111 1111 になります。

value の bit7 をチェックします。
例えば、value が 1010 0111 なら 0x80 で AND して

```
    value 1010 0111
& )       1000 0000
--------------------
          1000 0000 … (A)
```

「(0 以外の値) != 0」は「1」という値を持ちます。
「 (0) != 0」は「0」という値を持ちます。
よって(A)が 0 以外なら、″1″(5V)を出力、
(A)が 0 なら″0″を出力します。
他も bit6～bit0 までチェックして、0 または 1 を出力します。

4.7. 演習 7 Arduino ライブラリを使用しないデータの入出力「8bit_in_out_nolibraly.ino」

演習6と同じ動作ですが、Arduino ライブラリの digitalWrite 関数と digitalRead 関数を使わず、マイコンのポートから直接入出力します。実行速度が高速になります。

4.7.1. 配線

前回の「4.6. 10ピンコネクタを使用したデータの入出力「8bit_in_out.ino」」と同様の結線です。

4.7.2. プログラム

```
 1 : //**************************************************************************
 2 : // ﾌｧｲﾙ内容      「8bit_in_out_nolibraly.ino」 8bit 単位で入力、出力(Adruino ライブラリ未使用)
 3 : // Copyright    ジャパンマイコンカーラリー実行委員会
 4 : // ライセンス   This software is released under the MIT License.
 5 : //              http://opensource.org/licenses/mit-license.php
 6 : //**************************************************************************
 7 :
 8 : // プロトタイプ宣言
 9 : unsigned char dipsw( void );
10 : void led( unsigned char value );
11 :
12 : void setup() {
13 :   // CN5 DIPSW 8bit
14 :   pinMode( 27, INPUT );  // D27 は入力のみ
15 :   pinMode( 28, INPUT );  // D28 は入力のみ
16 :   pinMode( 29, INPUT );
17 :   pinMode( 30, INPUT );
18 :   pinMode( 31, INPUT );
19 :   pinMode( 32, INPUT );
20 :   pinMode( 33, INPUT );
21 :   pinMode( 34, INPUT);
22 :
23 :   // CN7 LED 8bit
24 :   pinMode( 35, OUTPUT );
25 :   pinMode( 36, OUTPUT );
26 :   pinMode( 37, OUTPUT );
27 :   pinMode( 38, OUTPUT );
28 :   pinMode( 39, OUTPUT );
29 :   pinMode( 40, OUTPUT );
30 :   pinMode( 41, OUTPUT );
31 :   pinMode( 42, OUTPUT );
32 : }
33 :
34 : void loop() {
35 :   unsigned char sw;
36 :
37 :   sw = dipsw();
38 :   led( sw );
39 : }
40 :
```

```
41 :  unsigned char dipsw( void ) {
42 :    unsigned char c;
43 :
44 :    c  = (R_PORT2->PIDR_b.PIDR14) << 7;
45 :    c |= (R_PORT2->PIDR_b.PIDR15) << 6;
46 :    c |= (R_PORT4->PIDR_b.PIDR6 ) << 5;
47 :    c |= (R_PORT4->PIDR_b.PIDR5 ) << 4;
48 :    c |= (R_PORT4->PIDR_b.PIDR4 ) << 3;
49 :    c |= (R_PORT4->PIDR_b.PIDR3 ) << 2;
50 :    c |= (R_PORT4->PIDR_b.PIDR2 ) << 1;
51 :    c |= (R_PORT4->PIDR_b.PIDR1 ) << 0;
52 :
53 :    return c;
54 :  }
55 :
56 :  void led( unsigned char value ) {
57 :    R_PORT0->PODR_b.PODR11 = ( (value & 0x80) != 0 );  // 式が真なら 1、偽なら 0 の値を持つ
58 :    R_PORT0->PODR_b.PODR10 = ( (value & 0x40) != 0 );
59 :    R_PORT0->PODR_b.PODR8  = ( (value & 0x20) != 0 );
60 :    R_PORT0->PODR_b.PODR7  = ( (value & 0x10) != 0 );
61 :    R_PORT0->PODR_b.PODR6  = ( (value & 0x08) != 0 );
62 :    R_PORT0->PODR_b.PODR5  = ( (value & 0x04) != 0 );
63 :    R_PORT0->PODR_b.PODR4  = ( (value & 0x02) != 0 );
64 :    R_PORT0->PODR_b.PODR3  = ( (value & 0x01) != 0 );
65 :  }
```

digitalRead 関数を使用せずに、直接ポートに入力されている電圧("0"(0V)か"1"(5V)か)を読み込みます。
digitalRead 関数を使用するより、高速に処理することができます。

digitalWrite 関数を使用せずに、直接ポートに"0"(0V)または"1"(5V)を出力します。
digitalWrite 関数を使用するより、高速に処理することができます。

4.7.3. プログラムの解説

Arduino ライブラリの digitalWrite 関数、digitarRead 関数は、RA4M1 マイコンのレジスタから読み込み・書き込みするより、実行時間がかかります。

例えば Advanced Class のプログラムを作るときに、PD 制御を 0.1ms(100μs)で実行するとき、データの入出力だけで 100μs の大半を使ってしまっては、センサ値の読み込みもれや、PD 値の計算が 100μs ごとにできない、など起こってしまいます。そこで、ポートの入出力は、直接、RA4M1 マイコンのレジスタから読み込み・書き込みをすると、高速に処理することができます。実測した時間を下記に示します。

Arduino ライブラリ使用 実測で 22μs	Arduino ライブラリを使用せず直接ポートから入力 実測で 3μs

c = (digitalRead(27) << 7);
c |= (digitalRead(28) << 6);
c |= (digitalRead(29) << 5);
c |= (digitalRead(30) << 4);
c |= (digitalRead(31) << 3);
c |= (digitalRead(32) << 2);
c |= (digitalRead(33) << 1);
c |= (digitalRead(34) << 0);
	c = (R_PORT2->PIDR_b.PIDR14) << 7;
	c |= (R_PORT2->PIDR_b.PIDR15) << 6;
	c |= (R_PORT4->PIDR_b.PIDR6) << 5;
	c |= (R_PORT4->PIDR_b.PIDR5) << 4;
	c |= (R_PORT4->PIDR_b.PIDR4) << 3;
	c |= (R_PORT4->PIDR_b.PIDR3) << 2;
	c |= (R_PORT4->PIDR_b.PIDR2) << 1;
	c |= (R_PORT4->PIDR_b.PIDR1) << 0;

記述の仕方について下記に示します。

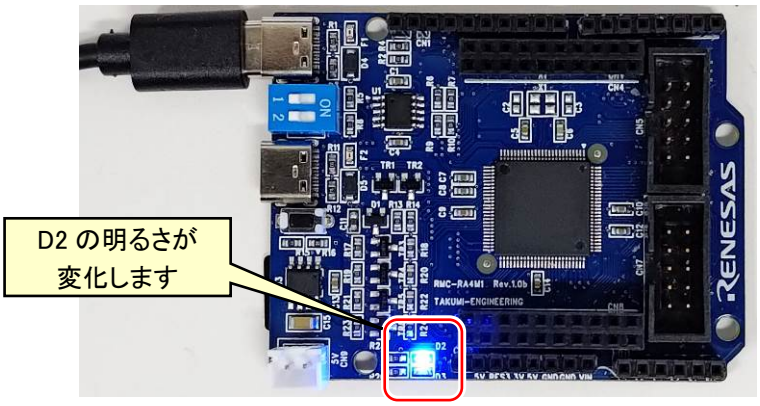
	プログラムの書き方	例	説明
入力	R_PORT●->PIDR_b.PIDR■	i = R_PORT2->PIDR_b.PIDR15;	P215 端子に入力されている論理(0 または 1)を読み込ます。
出力	R_PORT●->PODR_b.PODR■	R_PORT0->PODR_b.PODR6 = 1;	P006 端子から 0 または 1(0V または 5V)を出力します。 ※「06」の十の桁が 0 なら、PODR 側は1の桁のみ記載します。

4.8. 演習 8　analogWrite を使用した PWM 出力「analogwrite.ino」

analogWrite 関数は、端子から PWM 信号を出力して擬似的に 0～5V のアナログ電圧を出力しているかのように振る舞う機能です。「analog」と書かれていますが、アナログ電圧 0～5V は出力しません。
ただし、A2 端子は、D/A 変換器を使用して実際に 0～5V を出力します。

4.8.1. 配線

特にありません。マイコンボード上の LED D2 を点灯させます。



4.8.2. プログラム

```
 1 :  //*****************************************************************************
 2 :  // ﾌｧｲﾙ内容       「analogWrite.ino」 PWM 信出力
 3 :  // Copyright     ジャパンマイコンカーラリー実行委員会
 4 :  // ライセンス    This software is released under the MIT License.
 5 :  //               http://opensource.org/licenses/mit-license.php
 6 :  //*****************************************************************************
 7 :
 8 :  void setup() {
 9 :    digitalWrite( 24, 1 );  // LED D3 消灯
10 :  }
11 :
12 :  void loop() {
13 :    static int i = 0;
14 :    analogWrite( 23, 255 - i );  // LED D2  PWM 出力
15 :    delay( 10 );
16 :    i++;
17 :    if( i >= 255 ) {
18 :      i = 0;
19 :    }
20 :  }
```

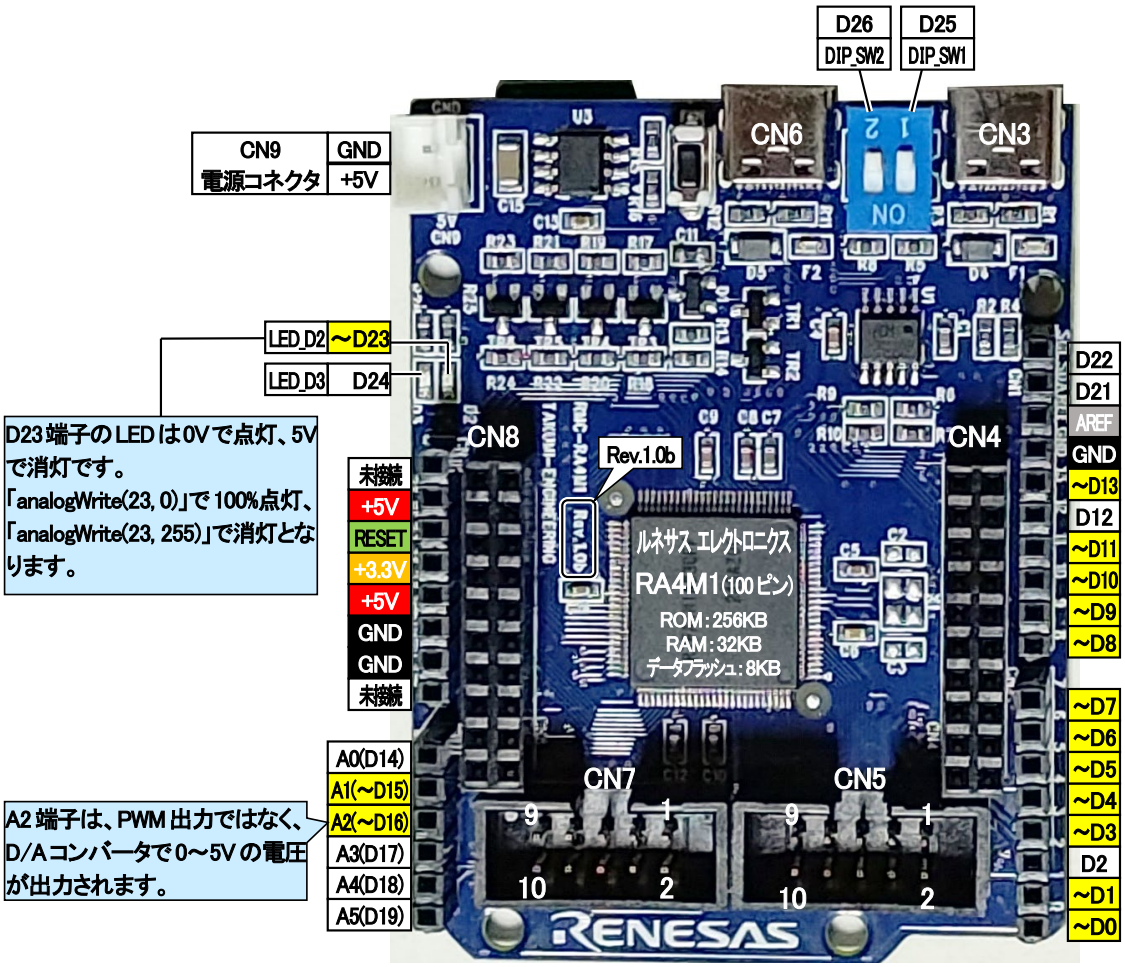
D24 端子は使用しないので、消灯しておきます。
D24 端子の LED は、"1"(5V)で消灯します。

D23 端子へ analogWrite して、LED の光量を変化させます。
最初は 255(5V)、10ms ごとに値が－１していきます。

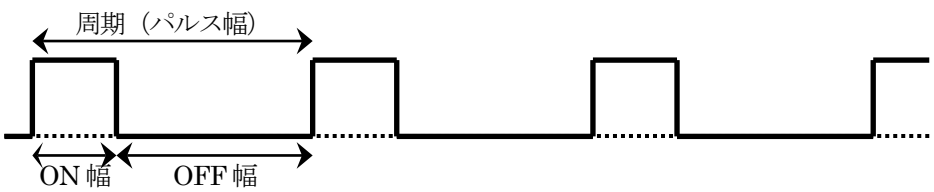
4.8.3. プログラムの解説

(1) 端子

analogWrite 関数を使用できるPWM 端子を下記に示します。黄色部分の「～」マークが付いている端子が、PWM を使用できる端子です。



(2) analogWrite 関数の使い方

使い方	analogWrite(端子 , ON の割合 0～255);
端子	0、1、3～11、13、A1、A2
ON の割合	下図の様にパルス幅(ON 幅＋OFF 幅)に対する、ON 幅の割合を指定します。 0 で 0%(常に 0V 出力)、255 で 100%(常に 5V 出力)となります。 例えば 128 だと、5V を 50%出力、0V を 50%出力となります。平均すると(128/255)×5V＝2.5V を擬似的に出力していることになります（平均が 2.5V で、実際に 2.5V 出力している訳ではありません）。 

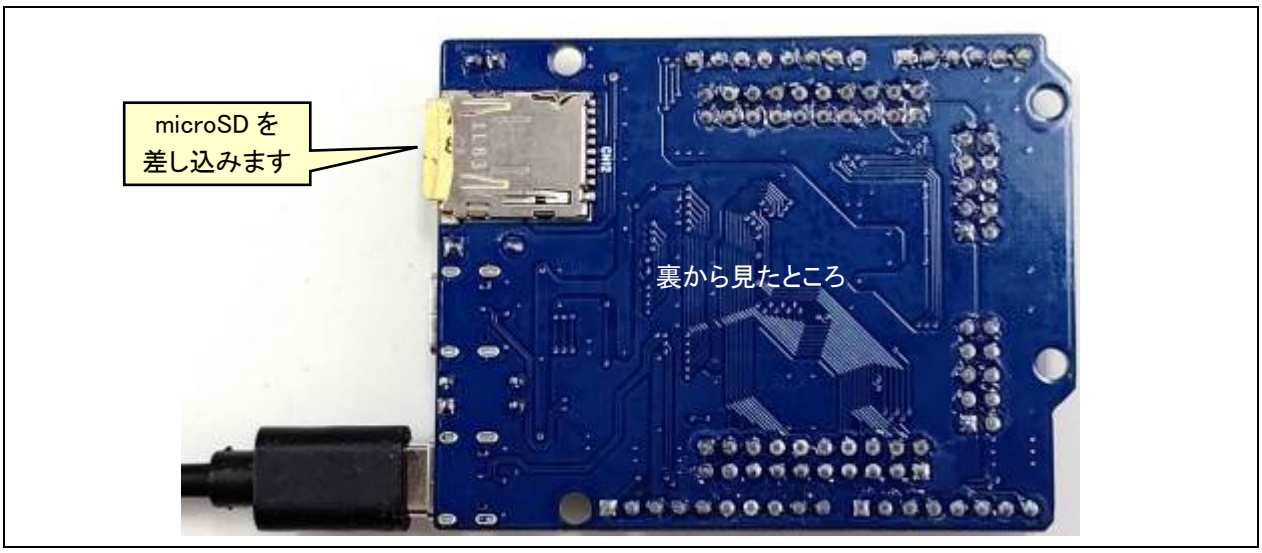
使用例	analogWrite (0 , 255); // D0 端子から、255(5V)を出力する
	analogWrite (1 , 128); // D1 端子から、(128/255)×5V＝2.5V を出力する

4.9. 演習 9　microSD へファイルの読み書き「microsd.ino」

RMC-RA4M1 ボードに搭載されている microSD へ、ファイルの読み書きを行います。microSD コネクタへ 32GB (SDHC)以下の microSD を差し込んでください。なお、microSD のフォーマット形式は、FAT または FAT32 のみに対応しています。NTFS や exFAT フォーマットには対応していません（エラーで読み書きできません）。

4.9.1. 配線

特にありません。RMC-RA4M1 ボードの microSD コネクタへ microSD を差し込みます。
差し込んだ microSD へファイルの読み書きを行います。
シリアル通信を行いますので、USB ケーブルを接続して、シリアルモニタを表示させておきます。



4.9.2. SD ライブラリの追加

1	-> candidates: [SPI] C:\Users\[redacted]\AppData\Local\Arduino15\package C:\Users\[redacted]\Documents\Arduino\microsd\micro #include <SD.h> ^~~~~ compilation terminated.	「検証・コンパイル」をしたときに、左のような「SD.h」に関するエラーが表示された場合、SD ライブラリを追加しなければいけません。 SD ライブラリの追加について、説明します。
2		❶左側の上から3個目の「  」をクリックします。 ❷検索欄に「sd」と入力します。 ❸「SD by Arduino, SparkFun」が表示されますので、「インストール」ボタンをクリックして、インストールしてください。 追加できたら「検証・コンパイル」をもう一度実行して、エラーが出ないことを確認してください。

4.9.3. プログラム

```
1 :  //****************************************************************************
2 :  // ﾌｧｲﾙ内容      「microsd.ino」microSD ファイル書き込み、読み込み
3 :  // Copyright    ジャパンマイコンカーラリー実行委員会
4 :  // ライセンス   This software is released under the MIT License.
5 :  //              http://opensource.org/licenses/mit-license.php
6 :  //****************************************************************************
7 :
8 :  // インクルード
9 :  #include <SPI.h>
10 :  #include <SD.h>
11 :
12 :  // グローバル変数宣言
13 :  File microSD;  // microSD のファイルアクセス用
14 :
```

microSD を制御するために、「SPI.h」と「SD.h」をインクルードします。

File クラスで microSD インスタンスを作成します。

```
15 :  void setup() {
16 :    Serial.begin( 9600 );
17 :
18 :    while( !Serial ) {
19 :      // シリアル接続されるまで待つ
20 :    }
21 :
22 :    Serial.println( "" );
23 :    Serial.print( "1 microSD を確認中です。
24 :
25 :
26 :    if( !SD.begin( CS ) ) {      // SD カードのチップセレクトライン それ以外は pins_arduino.h で定義
27 :      Serial.println( "microSD が認識されません!" );
28 :      while( 1 );  // 終了
29 :    } else {
30 :      // microSD 認識 OK!
31 :      Serial.println( "microSD を認識しました。" );
32 :    }
33 :
34 :    // microSD へ書き込み
35 :    microSD = SD.open( "test.txt", FILE_WRITE );
36 :    if( microSD != 0 ) {  // エラーがない
37 :      microSD.println( "テストです。" );  // ファイルの末尾に追加で書き込む
38 :      microSD.close();
39 :      Serial.println( "2 microSD に書き込みができました。" );
40 :    } else {
41 :      Serial.println( "2 ファイル書き込み(FILE_WRITE)ができませんでした。" );
42 :      while( 1 );
43 :    }
44 :
45 :    // microSD から読み込み
46 :    microSD = SD.open( "test.txt", FILE_READ );
47 :    if( microSD != 0 ) {  // エラーがないなら
48 :      Serial.println( "3 ファイルの内容を表示します。" );
49 :      Serial.println( "------------------------------ ここから" );
50 :      while ( microSD.available() ) {      // ファイルにデータがあれば
51 :        Serial.write( microSD.read() );    // データを読み込んでシリアルに出力
52 :      }
53 :      microSD.close();
54 :    } else {
55 :      Serial.println("3 ファイル読み込み(FILE_READ)ができませんでした。");
56 :      while( 1 );
57 :    }
58 :
59 :    Serial.println( "------------------------------ ここまで" );
60 :    Serial.println( "4 プログラムを終了します。" );
61 :  }
62 :
63 :  void loop() {
64 :    // 何もしない
65 :  }
```

SD.begin で SD カードと通信するための初期設定を行います。エラーになると戻り値が 0 になります。このとき「if(!0)」と同じ意味になります。「!0」は 1 なので、if 文が成り立ち、エラーと表示して、「while(1);」の無限ループでプログラムを終了します。

microSD を認識すると、認識しましたと表示して、次へ行きます。

ファイル名「test.txt」でファイルを開きます。開き方は「FILE_WRITE」なので、書き込みモードになります。

ファイルが開けたら、
「microSD.println」で、ファイルの末尾に追加で文字列を書き込みます。
「microSD.close();」で、ファイル処理を終了します。close しないと、正しくファイルが書き込めませんので必ず close してください。

ファイル名「test.txt」でファイルを開きます。開き方は「FILE_READ」なので、読み込みモードになります。

「microSD.avilable()」はファイルに読み込むデータがあるかチェックする関数です。データがあれば、「microSD.read()」でファイルから 1 バイト読み込みます。すべて read すると microSD.avilable()は 0 となり、読み込みを終了します。

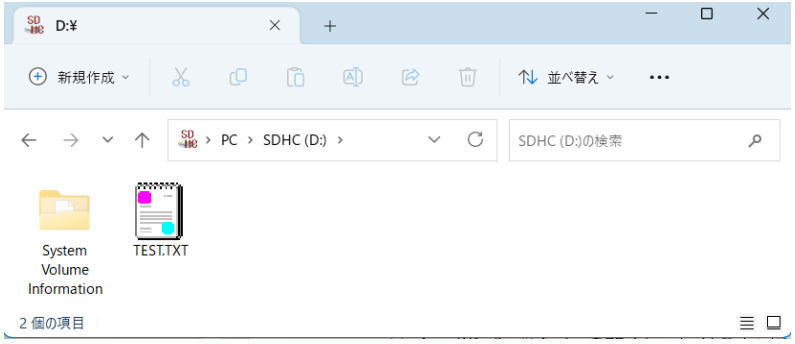
4.9.4. プログラムの解説

プログラムを実行すると、microSD の「test.txt」ファイルの末尾に、「テストです。」と文字を追加します。
シリアルモニタには、下記のように表示されます。

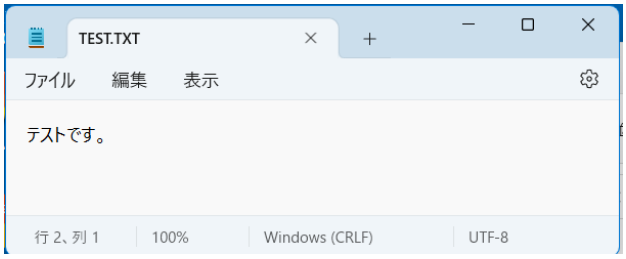
```
1 microSD を確認中です... microSD を認識しました。
2 microSD に書き込みができました。
3 ファイルの内容を表示します。
-------------------------------- ここから
テストです。
-------------------------------- ここまで
4 プログラムを終了します。
```

「test.txt」の内容を表示します。
プログラムを実行するたびに、「テストです。」を追加していくので、何度も実行すると「テストです。」が複数行、表示されます。

microSD をパソコンで開くと、下記のように「test.txt」ファイルが表示されます。



「test.txt」をダブルクリックして開くと、メモ帳などで、ファイルの内容が開きます。



microSD が挿入されていなかったり、microSD のフォーマットが FAT や FAT32 でない場合は、次のようにシリアルモニターにエラーが表示されます。

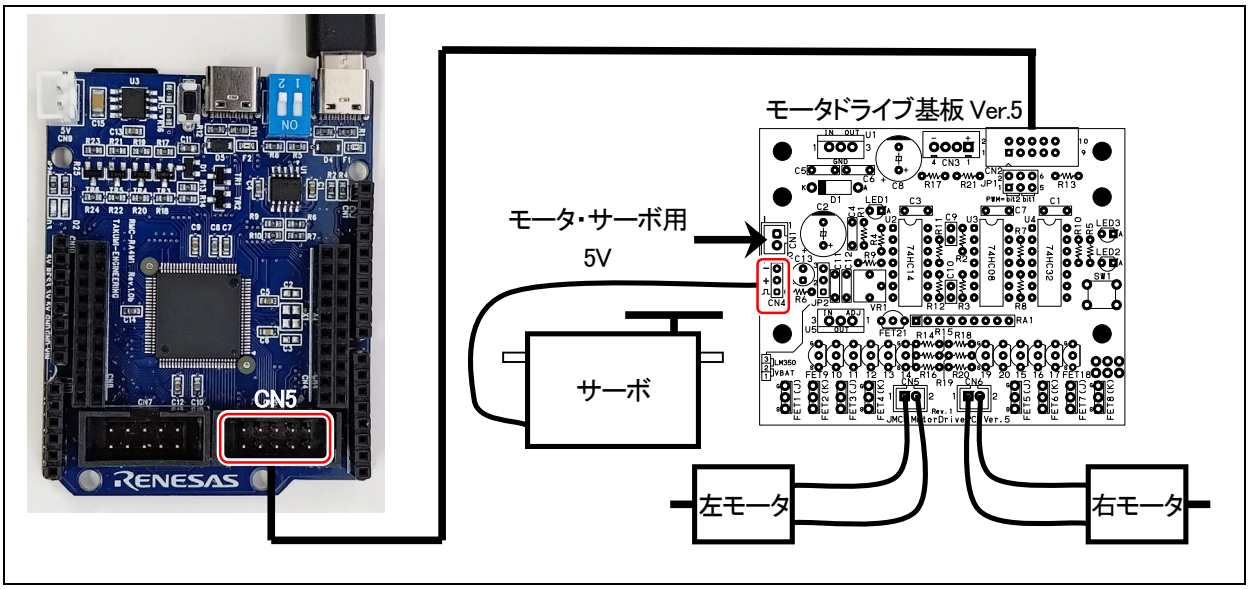
```
1 microSD を確認中です... microSD が認識されません!
```

4.10. 演習 10 モータドライブ基板(Ver.5)のモータ、サーボ制御(GPT を使用した PWM)「gpt_pwm.ino」

RMC-RA4M1 ボードの CN5 とモータドライブ基板 Ver.5 を接続し、サーボと左右モータを制御します。

4.10.1. 配線

RMC-RA4M1 ボードの CN5 とモータドライブ基板 Ver.5 をフラットケーブルで接続します。



4.10.2. プログラム

```
 1 :  //********************************************************************************
 2 :  // ﾌｧｲﾙ内容      「gpt_pwm.ino」モータドライブ基板へ PWM 出力でサーボ、モータを制御
 3 :  // Copyright    ジャパンマイコンカーラリー実行委員会
 4 :  // ライセンス   This software is released under the MIT License.
 5 :  //              http://opensource.org/licenses/mit-license.php
 6 :  //********************************************************************************
 7 :
 8 :  // インクルード宣言
 9 :  #include "mcr_gpt_lib.h"
10 :
11 :  // 定数の宣言
12 :  #define PWM_CYCLE     47999           //  16ms/{1/(HOCO/16)} = 0.016 /{1/(48e6/16)} = 48000
13 :  #define SERVO_CENTER  4500            // 1.5ms/{1/(HOCO/16)} = 0.0015/{1/(48e6/16)} =  4500
14 :  #define HANDLE_STEP   26              // １度あたりの値
15 :
16 :  // プロトタイプ宣言
17 :  void motor(int accele_l, int accele_r);
18 :  void handle(int angle);
19 :  unsigned char dipsw_get(void);
20 :
```

```
21 :  void setup() {
22 :    pinMode( 25, INPUT );                // DIPSW1
23 :    pinMode( 26, INPUT );                // DIPSW2
24 :
25 :    setGPTterminal( 4, 6 );              // P406(GTIOC1B):サーボ PWM
26 :    setGPTterminal( 4, 5 );              // P405(GTIOC1A):左モータ PWM
27 :    setGPTterminal( 4, 3 );              // P403(GTIOC3A):右モータ PWM
28 :
29 :    GTIOC1A = 0;
30 :    GTIOC1B = SERVO_CENTER;
31 :    startPWM_GPT1( GTIOCA | GTIOCB , DIV16, PWM_CYCLE );  // GPT1 GTIOCA と B を使用 分周:16 周期:PWM_CYCLE
32 :
33 :    GTIOC3A = 0;
34 :    startPWM_GPT3( GTIOCA          , DIV16, PWM_CYCLE );  // GPT3 GTIOCA を使用    分周:16 周期:PWM_CYCLE
35 :
36 :    // CN5 モータドライブ基板
37 :    //pinMode( 27 , OUTPUT );          // LED2 rev.1.0b は入力端子なので LED は点灯しない
38 :    //pinMode( 28 , OUTPUT );          // LED3 rev.1.0b は入力端子なので LED は点灯しない
39 :    //pinMode( 29 , OUTPUT );          // サーボ PWM なので設定しない
40 :    //pinMode( 30 , OUTPUT );          // 右モータ PWM なので設定しない
41 :    pinMode(31, OUTPUT);               // 右モータ方向
42 :    //pinMode( 32 , OUTPUT );          // 左モータ PWM なので設定しない
43 :    pinMode(33, OUTPUT);               // 左モータ方向
44 :    pinMode(34, INPUT);                // プッシュスイッチ
45 :  }
46 :
47 :  void loop() {
48 :
49 :    handle( 0 );
50 :    motor( 0, 0 );
51 :    delay( 1000 );
52 :
53 :    handle(10 );
54 :    motor( 0, 25 );
55 :    delay( 1000 );
56 :
57 :    handle( -10 );
58 :    motor( 25, 0 );
59 :    delay( 1000 );
60 :
61 :    handle( 20 );
62 :    motor( 0, -50 );
63 :    delay( 1000 );
64 :
65 :    handle( -20 );
66 :    motor( -50, 0 );
67 :    delay( 1000 );
68 :  }
69 :
```

```
 70 :  //*****************************************************************
 71 :  // モータ速度制御
 72 :  // 引数    左モータ:-100～100、右モータ:-100～100
 73 :  //         0 で停止、100 で正転 100%、-100 で逆転 100%
 74 :  // 戻り値 なし
 75 :  //*****************************************************************
 76 :  void motor(int accele_l, int accele_r) {
 77 :    int sw_data;
 78 :
 79 :    sw_data = dipsw_get() + 7;  // 7～10
 80 :    accele_l = accele_l * sw_data / 10;
 81 :    accele_r = accele_r * sw_data / 10;
 82 :
 83 :    // 左モータ制御
 84 :    if(accele_l >= 0) {
 85 :      R_PORT4->PODR_b.PODR2 = 0;
 86 :      accele_l = (long)(PWM_CYCLE + 1) * accele_l / 100;
 87 :      GTIOC3A = accele_l;
 88 :    } else {
 89 :      R_PORT4->PODR_b.PODR2 = 1;
 90 :      accele_l = (long)(PWM_CYCLE + 1) * (-accele_l) / 100;
 91 :      GTIOC3A = accele_l;
 92 :    }
 93 :
 94 :    // 右モータ制御
 95 :    if(accele_r >= 0) {
 96 :      R_PORT4->PODR_b.PODR4 = 0;
 97 :      accele_r = (long)(PWM_CYCLE + 1) * accele_r / 100;
 98 :      GTIOC1A = accele_r;
 99 :    } else {
100 :      R_PORT4->PODR_b.PODR4 = 1;
101 :      accele_r = (long)(PWM_CYCLE + 1) * (-accele_r) / 100;
102 :      GTIOC1A = accele_r;
103 :    }
104 :  }
105 :
106 :  //*****************************************************************
107 :  // サーボハンドル操作
108 :  // 引数    サーボ操作角度：-90～90
109 :  //         -90 で左へ 90 度、0 でまっすぐ、90 で右へ 90 度回転
110 :  //*****************************************************************
111 :  void handle(int angle) {
112 :    // サーボが左右逆に動く場合は、「-」を「+」に替えてください
113 :    GTIOC1B = SERVO_CENTER - angle * HANDLE_STEP;
114 :  }
115 :
116 :  //*****************************************************************
117 :  // ディップスイッチ値読み込み
118 :  // 戻り値 スイッチ値 0～3
119 :  //*****************************************************************
120 :  unsigned char dipsw_get(void) {
121 :    unsigned char sw;
122 :
123 :    sw = (!(R_PORT3->PIDR_b.PIDR7) << 1) + (!(R_PORT3->PIDR_b.PIDR6));
124 :
125 :    return sw;
126 :  }
```

4.10.1. プログラムの解説

(1) GPT とは

RA4M1 マイコンの GPT とは、汎用タイマ(General Purpose Timer)のことで、PWM 波形を出力したり、エンコーダのパルスを入力したりすることができる機能のことです。GPT は 0～7 番まで8個あります。
ChatGPT(Generative Pre-trained Transformer)とはまったく関係ありません。ChatGPT が発表される前からルネサスの GPT 機能は存在しています。

今回は、GPT を使用して PWM 波形を出力します。PWM は、左モータ、右モータ、サーボモータを制御するために 3 つの端子から出力します。
Arduino ライブラリの analogWrite 関数でも PMW 波形を出力することができますが、周期の設定方法が公開されていなかったり、分解能が 0～255 までしか設定できないなどあるので、独自のライブラリ「mcr_gpt_lib」を使用します。

(2)「mcr_gpt_lib」を使って PWM 波形を出力する

①GPT で PWM を使用するために「mcr_gpt_lib.h」を呼び出します。

```
 9 :  #include "mcr_gpt_lib.h"
```

②GPT で PWM 波形を出力する端子を設定します。RMC-RA4M1 ボードの端子を見て、GTIOCxy(x=0～7、y=A または B)端子であれば、PWM 波形を出力することができます。このとき、GTIOCxA と GTIOCxB の端子(x は同じ数字)は、PWM 波形の周期は必ず同じになります。例えば、GTIOC1A と GTIOC1B は同じ周期になります。
モータドライブ基板は 10 ピンコネクタの、4 番ピンがサーボ PWM、5 番ピンが左モータ PWM、7 番ピンが右モータ PWM となります。モータドライブ基板は、RMC-RA4M1 ボードの CN5 に接続します。サーボにつながる CN5 の 4 番ピンは GTIOC1B 端子なので、PWM 波形を出力することができます。この端子は P406 なので、
「setGPTterminal(4, 6);」として GTIOC1B 端子として使用することを宣言します。
同様に、CN5 の 5 番ピン、CN5 の 7 番ピンも PWM 波形出力端子として使用することを設定します。

```
25 :    setGPTterminal( 4, 6 );            // P406(GTIOC1B):サーボ PWM
26 :    setGPTterminal( 4, 5 );            // P405(GTIOC1A):左モータ PWM
27 :    setGPTterminal( 4, 3 );            // P403(GTIOC3A):右モータ PWM
```

このときの注意点は、例えば GTIOC1B 端子は、P406(D29 端子)、P104(D3 端子)、P110(D12 端子)がありますが、1 端子しか GTIOC1B 端子に設定できません。下記に記述してはいけない例を示します。

```
    setGPTterminal( 4,  6 );            // P406(GTIOC1B) GTIOC1B 端子は１つしか設定できません！
    setGPTterminal( 1,  4 );            // P104(GTIOC1B)
    setGPTterminal( 4, 10 );            // P410(GTIOC1B)
```

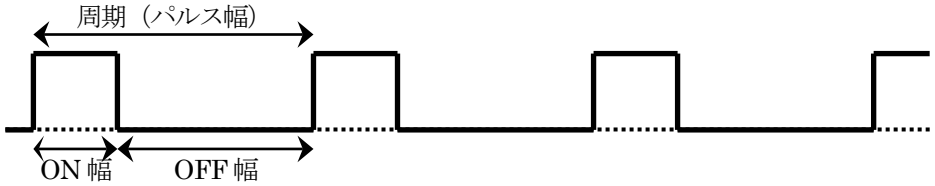
(3) PWM 波形を出力する

端子の設定、PWM 波形を出力するプログラムは、29 行～31 行です。

```
29 :    GTIOC1A = 0;
30 :    GTIOC1B = SERVO_CENTER;
31 :    startPWM_GPT1( GTIOCA | GTIOCB , DIV16, PWM_CYCLE );  // GPT1 GTIOCA と B を使用 分周:16 周期:PWM_CYCLE
```

3 行のプログラムについて説明します。 ※説明の順番の関係で 31 行、29 行と 30 行の順に説明します。

①startPWM_GPTx 関数

使い方	startPWM_GPTx(使用する端子 , 分周比 , PWM 周期の値); x にはチャネル番号 0～7 までの数字が入ります。実行すると GPT がスタートし PWM 波形が出力されます。
使用する端子	※x=チャネル番号です。x には 0～7 が入ります。 GTIOCxA 端子を PWM 出力にするとき ： GTIOCA（数字は入れません） GTIOCxB 端子を PWM 出力にするとき ： GTIOCB（数字は入れません） 例：GTIOC1A 端子を PWM 出力にするとき ： GTIOCA GTIOC1A 端子と GTIOC1B 端子を PWM 出力にするとき ： GTIOCA 縦棒（たてぼう） GTIOCB
分周比	DIV1 ： 1 分周　　DIV64 ： 64 分周 DIV4 ： 4 分周　　DIV256 ： 256 分周 DIV16 ： 16 分周　　DIV1024 ： 1024 分周 上記の６つのいずれかを設定します。どれを設定するかは、「PWM 周期の値」で説明します。
PMW 周期の値	PWM 波形の周期を設定します。周期とは下図のように、波形の ON 幅＋OFF 幅のことです。  PWM 周期の値の計算方法を下記に示します。まず、分周比をどの値にするかを決めます。 PWM 周期の値は最大 65536 です。各分周比で設定できる最大周期を下記に示します。 ※プログラムで設定するときは、１小さい値を設定します。例：計算が 65536 なら設定値は 65535 となります。 DIV1 ： 1.3653[ms] ※65536 × { 1 / (48MHz / 1 }) DIV4 ： 5.4613[ms] ※65536 × { 1 / (48MHz / 4 }) DIV16 ： 21.8453[ms] ※65536 × { 1 / (48MHz / 16 }) DIV64 ： 87.3813[ms] ※65536 × { 1 / (48MHz / 64 }) DIV256 ： 349.5253[ms] ※65536 × { 1 / (48MHz / 256 }) DIV1024 ： 1398.1013[ms] ※65536 × { 1 / (48MHz / 1024 }) 例えば、周期を 16ms にしたい場合は、最大周期が 16ms 以上、かつ小さい DIV 値を選びます。上記の最大周期より、16ms 以上の「DIV16」を選びます。 DIV1024 の 1398.1013[ms]を超える周期は設定できません。 PWM 周期の値の計算方法を、下記に示します。 PWM 周期の値 ＝ 設定したい PWM 周期 / { 1/(48MHz/分周比) } 例えば、周期を 16ms にしたければ、DIV は 16 として、 PWM 周期の値 ＝ 16ms / {1/(48MHz / 16) } = 0.016 / { 1 / (48×10^6 / 16) } = 48000 設定値は、タイマの仕様上、１小さい値を設定するので、最終的な設定値は 47999 となります。

例１	CN4 の D48(P411 GTIOC6A)と D49(P410 GTIOC6B)から周期 1ms の PWM 波形を出力します。 まず、周期 1ms なので DIV は１となります。 PWM 周期の値 ＝ 1ms / {1/(48MHz / 1) } = 1×10^{-3} / { 1 / (48×10^{6} / 16) } = 48000 設定値は１小さい 47999 となります。 最終的なプログラムは、 `setGPTterminal( 4 , 11 );` // P411(GTIOC6A)端子 `setGPTterminal( 4 , 10 );` // P410(GTIOC6B)端子 `startPWM_GPT6(GTIOCA | GTIOCB , DIV1 , 47999 );` となります。
例２	CN8 の D77(P610 GTIOC5B)から周期 25ms の PWM 波形を出力します。 まず、周期 25ms なので DIV は 64 となります。 PWM 周期の値 ＝ 25ms / {1/(48MHz / 64) } = 0.025 / { 1 / (48×10^{6} / 64) } = 18750 設定値は１小さい 18749 となります。 最終的なプログラムは、 `setGPTterminal( 6 , 10 );` // P610(GTIOC5B)端子 `startPWM_GPT5( GTIOCB , DIV64 , 18749 );` となります。

②ON 幅の設定

使い方	GTIOCxy = ON 幅の設定 x = GPT のチャンネル 0～7 y = A または B
ON 幅について	ON 幅の設定を設定します。1 当たりの PWM 波形の ON 幅は下記の通りです。 startPWM_GPTx 関数で設定したときの分周比で設定値が変わります。 DIV1 : 0.02083[μs] = 20.83[ns] = { 1/(24MHz/ 1) } DIV4 : 0.08333[μs] = 83.33[ns] = { 1/(24MHz/ 4) } DIV16 : 0.33333[μs] = 333.33[ns] = { 1/(24MHz/ 16) } DIV64 : 1.33333[μs] = 1333.33[ns] = { 1/(24MHz/ 64) } DIV256 : 5.33333[μs] = 5333.33[ns] = { 1/(24MHz/ 256) } DIV1024 : 21.33333[μs] = 21333.33[ns] = { 1/(24MHz/1024) } 例えば、DIV4 のとき、GTIOCxy の値を１つ増やすと、ON 幅が 83.33[ns]増えます。

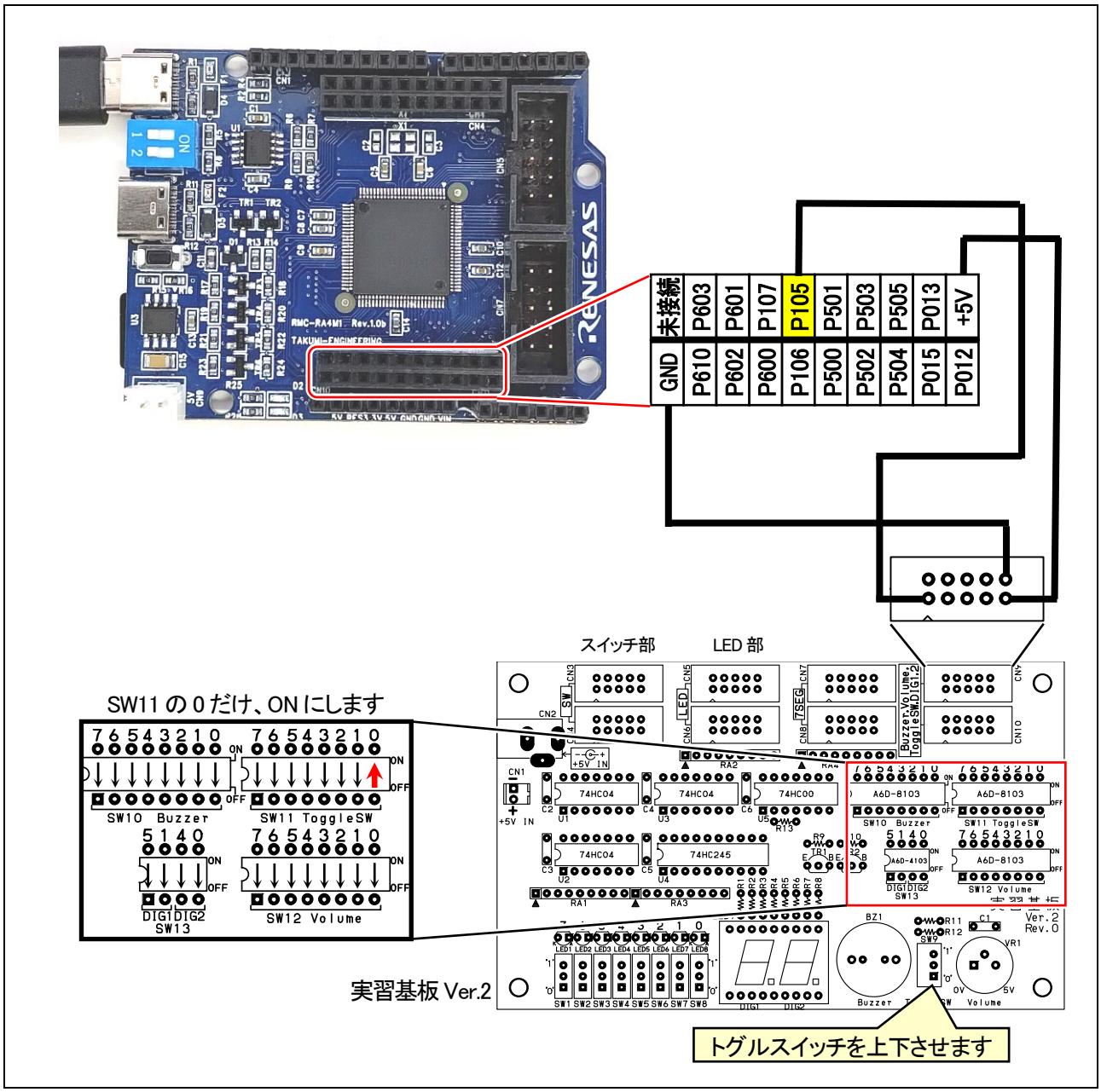
例	`setGPTterminal( 4, 6 );` // P406 GTIOC1B :サーボ PWM として使用
	`setGPTterminal( 4, 5 );` // P405 GTIOC1A :左モータ PWM として使用
	`GTIOC1A = 0;` // 初期値を設定
	`GTIOC1B = 4500;` // 初期値を設定
	`startPWM_GPT1( GTIOCA | GTIOCB , DIV16, 47999 );` // GTIOC1A と GTIOC1B を使用
	`int i = 0;`
	`while( 1 ) {`
	`  GTIOC1A = i;` // DIV16 なので 1 あたり 333.33[ns]ずつ ON 幅を増やす
	`  delay( 1 );`
	`  i++;`
	`  if( i > 48000 ) {`
	`  i = 0;`
	`  }`
	`}`

4.11. 演習 11　1相のロータリエンコーダからパルスを入力する「encoder1sou.ino」

RA4M1 マイコンの GPT を使用して1相のロータリエンコーダからパルスを入力します。

4.11.1. 配線

CN8(20 ピンコネクタ)の D70(P105)を、1 相のロータリエンコーダのパルス出力端子に接続します。ロータリエンコーダが無い場合は、下図のように実習基板 Ver.2 のトグルスイッチと接続してください。
シリアル通信を行いますので、USB ケーブルを接続して、シリアルモニタを表示させておきます。



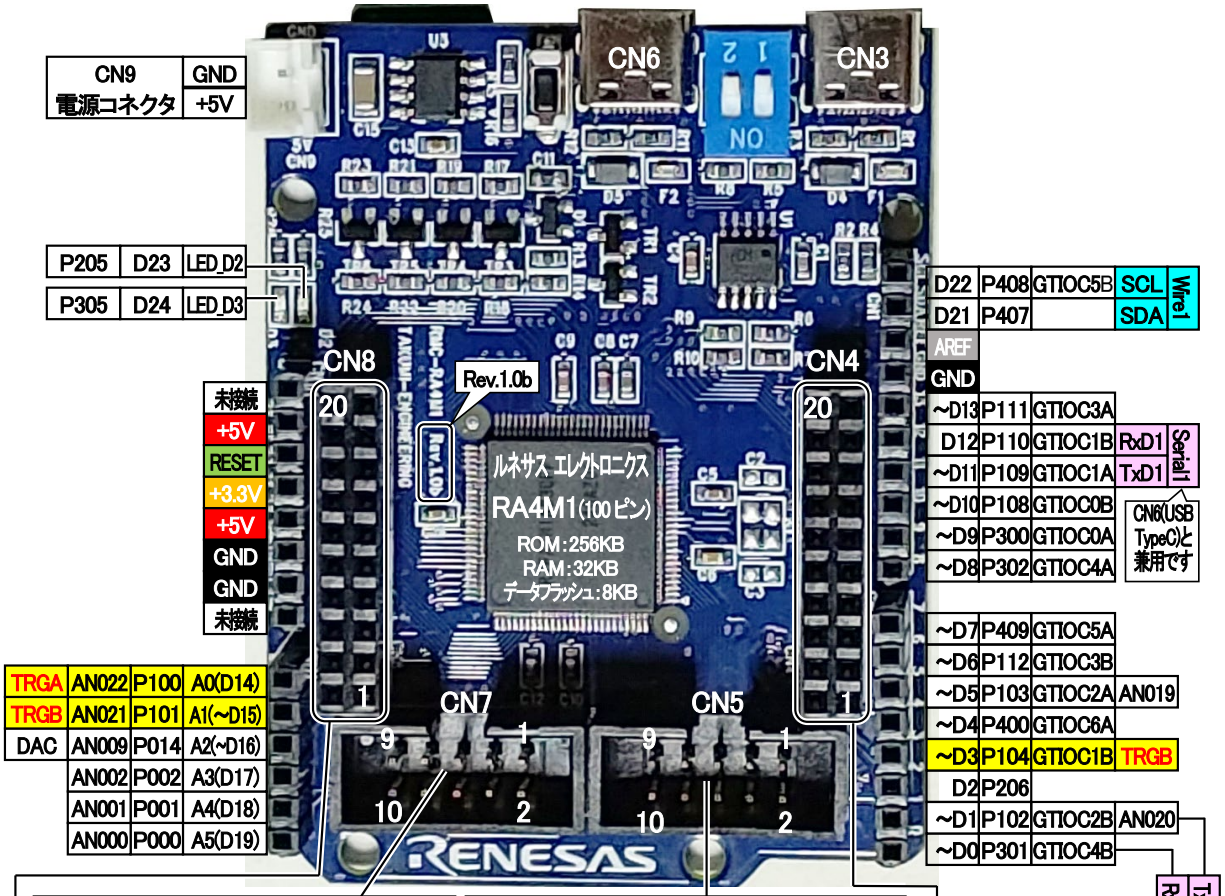
4.11.2. プログラム

```
 1 : //*****************************************************************************
 2 : // ﾌｧｲﾙ内容      「encoder1sou.ino」1 相のロータリエンコーダ　パルス入力
 3 : // Copyright    ジャパンマイコンカーラリー実行委員会
 4 : // ライセンス   This software is released under the MIT License.
 5 : //              http://opensource.org/licenses/mit-license.php
 6 : //*****************************************************************************
 7 :
 8 : // インクルード宣言
 9 : #include "mcr_gpt_lib.h"
10 :
11 : void setup() {
12 :   startGPT1_1SouEncoder( GTETRGA , 1, 5 );  // 1 相エンコーダ GPT1 P105 の GTETRGA を使用
13 :
14 :   Serial.begin(9600);                       // シリアル通信 速度 9600bps
15 :
16 :   while ( !Serial ) {
17 :     // 接続されるまで待つ
18 :   }
19 :   Serial.print( "¥n¥n ロータリエンコーダ 1 相確認プログラム¥n" );
20 : }
21 :
22 : void loop() {
23 :   static int lp = 1;
24 :
25 :   Serial.print( lp );
26 :   Serial.print( " : " );
27 :   Serial.println( GPT1_CNT );
28 :   lp++;
29 :   delay( 1000 );
30 : }
```

4.11.3. プログラムの解説

(1) 1 相のロータリエンコーダのパルス入力として使える端子

1 相のロータリエンコーダのパルス入力として使える端子は、GTETRGA 端子(略して TGRA 端子)または GTETRGB 端子(略して TGRB 端子)です。下図に、GTETRGA 端子、GTETRGB 端子(黄色部分)を示します。



CN7 10ピンコネクタ(センサ基板)

D42	D40	D38	D36	+5V
P003	P005	P007	P010	
AN003	AN011	AN013	AN005	
GND	D41	D39	D37	D35
	P004	P006	P008	P011
	AN004	AN012	AN014	AN006

CN5 10ピンコネクタ(モータドライブ基板)

D34	D32	D30	D28	+5V
P401	P403	P405	(P215)	
TGRA/6Bﾊﾟﾙｽ入力SW	3A(左ﾓｰﾀPWM)	1A(右ﾓｰﾀPWM)	(LED3)	
GND	D33	D31	D29	D27
	P402	P404	P406	(P214)
	(左ﾓｰﾀ方向)	3B(右ﾓｰﾀ方向)	1B(ｻｰﾎﾞ)	(LED2)

※(P215)、(P214)は入力専用端子

CN8 20 ピンコネクタ

		GND	未接続		
GTIOC5B	P610	D77	D76	P603	GTIOC7A
GTIOC7B	P602	D75	D74	P601	GTIOC6A
GTIOC6B	P600	D73	D72	P107	GTIOC0A
GTIOC0B	P106	D71	D70	P105	1A/TRGA
AN016/2A	P500	D69	D68	P501	AN017/2B
AN018/3B	P502	D67	D66	P503	AN023
AN024	P504	D65	D64	P505	AN025
AN010	P015	D63	D62	P013	AN008
AN007	P012	D61	+5V		

CN4 20 ピンコネクタ ※(P200)は入力専用端子

		GND	未接続		
GTIOC5A	P609	D59	D58	P608	GTIOC4B
GTIOC4A	P115	D57	D56	P114	GTIOC2B
GTIOC2A	P113	D55	D54	P303	GTIOC7B
–	P809	D53	D52	P808	–
GTIOC7A	P304	D51	D50	(P200)	–
GTIOC6B	P410	D49	D48	P411	GTIOC6A
–	P412	D47	D46	P413	–
GTIOC0B	P414	D45	D44	P415	GTIOC0A
–	P708	D43	+5V		

(2)「mcr_gpt_lib」を使って 1 相のロータリエンコーダからパルスを入力する

①GPT で 1 相のロータリエンコーダからパルスを入力するために「mcr_gpt_lib.h」を呼び出します。

```
9 :  #include "mcr_gpt_lib.h"
```

②GPT で 1 相のロータリエンコーダからパルスを入力して、パルスをカウントします。
RMC-RA4M1 ボードの端子を見て、GTETRGA 端子(略して TGRA 端子)または GTETRGB 端子(略して TGRB 端子)から1相のロータリエンコーダのパルスを入力することができます。
今回は、CN8 の D70 端子(P105、GTETRGA 端子)を、パルス入力端子にしています。端子と GPT のチャンネルは関係なく、GPT は 0～7 のどれを使っても構いません。ただし、GPT のチャンネル 0 と1は、エンコーダ値を 2^{32}-1(4,294,967,295)までカウントすることができるのでチャネル 0 または 1 の使用がおすすめです。ちなみに、チャネル 2～7 は 2^{16}-1(65,535)までカウントすることができます。
今回のプログラムを下記に示します。

```
12 :    startGPT1_1SouEncoder( GTETRGA , 1, 5 );  // 1 相エンコーダ GPT1 P105 の GTETRGA を使用
```

(3) startGPTx_1SouEncoder 関数の使い方(x＝0～7)

使い方	startGPTx_1SouEncoder(使用する端子の GTETRG(GTETRGA または GTETRGB) , ポート上位１桁 , ポート下位２桁); x にはチャネル番号 0～7 までの数字が入ります。実行すると GPT がスタートし 1 相のパルスをカウントします。
使用する端子の GTETRG	GTETRGA または GTETRGB を指定します。
ポート上位１桁	例えば P105 なら下線部分の 1 になります。
ポート下位２桁	例えば P105 なら波線部分の 5 になります。
例	startGPT2_1SouEncoder(GTETRGB , 1, 4); // 1 相エンコーダ GPT2 P104 の GTETRGB を使用

(4) カウント値の読み込み方

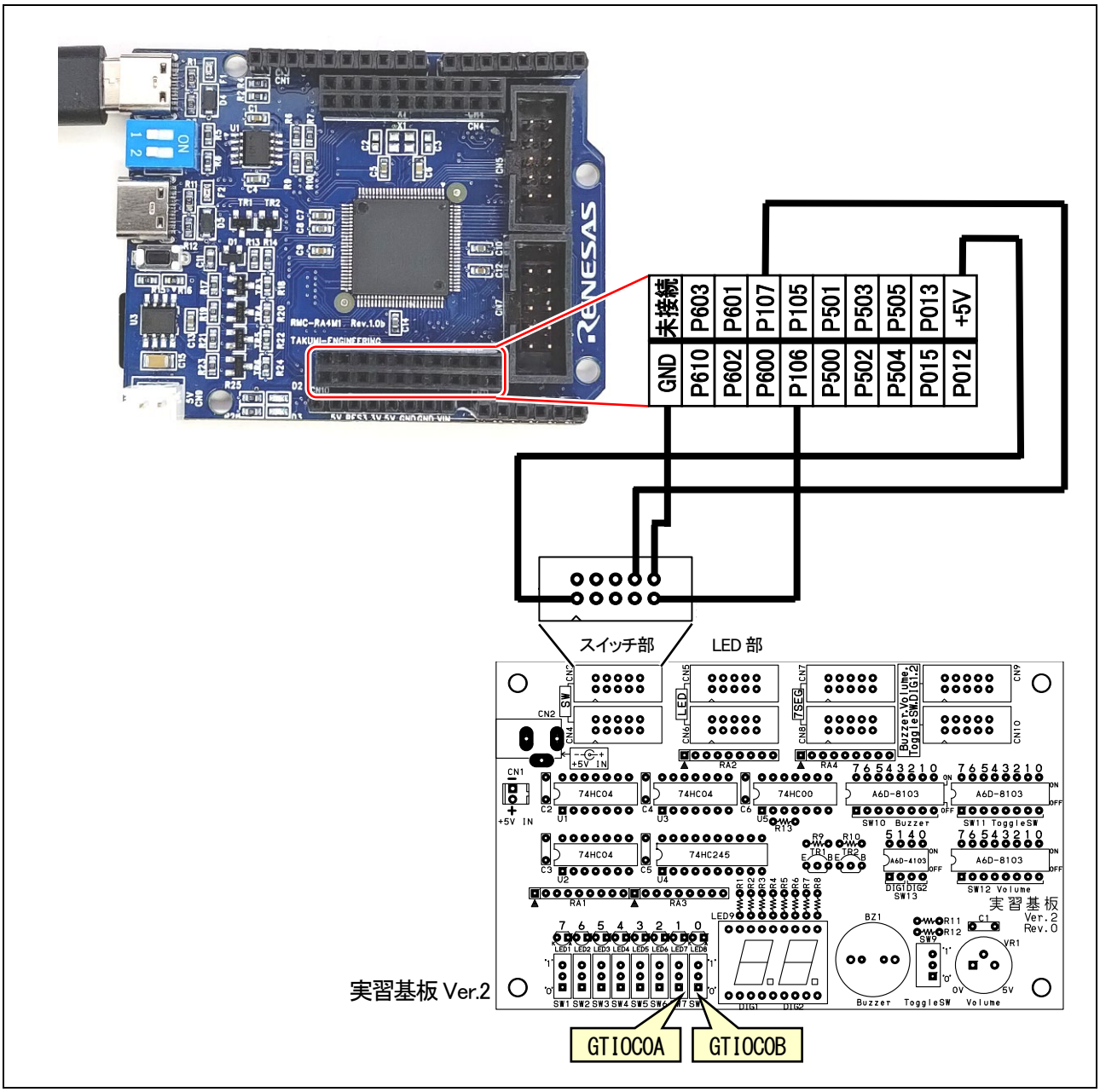
使い方	「GPTx_CNT」の値を読み込むと、カウント値が読み込まれます。 x にはチャネル番号 0～7 までの数字が入ります。 チャネル 0～1、は 0～2^{32}-1 (4,294,967,295) の範囲でカウントすることができます。 チャネル 2～7、は 0～2^{16}-1 (65,535) の範囲でカウントすることができます。
例	i = GPT2_CNT; // GPT2 のカウント値を変数 i に代入

4.12. 演習 12　2相のロータリエンコーダからパルスを入力する「encoder2sou.ino」

RA4M1 マイコンの GPT を使用して 2 相のロータリエンコーダからパルスを入力します。

4.12.1. 配線

CN8(20 ピンコネクタ)の D71(P106)と D72(P107)を、2 相のロータリエンコーダのパルス出力端子に接続します。2 相のロータリエンコーダが無い場合は、下図のように実習基板 Ver.2 のスイッチ部分と接続してください。
シリアル通信を行いますので、USB ケーブルを接続して、シリアルモニタを表示させておきます。



4.12.2. プログラム

```
 1 :  //*****************************************************************************
 2 :  // ﾌｧｲﾙ内容      「encoder2sou.ino」2 相のロータリエンコーダ　パルス入力
 3 :  // Copyright    ジャパンマイコンカーラリー実行委員会
 4 :  // ライセンス   This software is released under the MIT License.
 5 :  //              http://opensource.org/licenses/mit-license.php
 6 :  //*****************************************************************************
 7 :
 8 :  // インクルード宣言
 9 :  #include "mcr_gpt_lib.h"
10 :
11 :  void setup() {
12 :    startGPT0_2SouEncoder( 1, 7 , 1, 6);  // 2 相エンコーダ 0A=P107 0B=P106 を使用
13 :
14 :    Serial.begin(9600);                   // シリアル通信 速度 9600bps
15 :
16 :    while ( !Serial ) {
17 :      // 接続されるまで待つ
18 :    }
19 :    Serial.print( "¥n¥n ロータリエンコーダ２相確認プログラム¥n" );
20 :  }
21 :
22 :  void loop() {
23 :    static int lp = 1;
24 :
25 :    Serial.print( lp );
26 :    Serial.print( " : " );
27 :    Serial.println( INT_GPT0_CNT );
28 :    lp++;
29 :    delay( 1000 );
30 :  }
```

4.12.3. プログラムの解説

(1)「mcr_gpt_lib」を使って 2 相のロータリエンコーダからパルスを入力する

①GPT で 2 相のロータリエンコーダからパルスを入力するために「mcr_gpt_lib.h」を呼び出します。

```
 9 :  #include "mcr_gpt_lib.h"
```

②GPT で 2 相のロータリエンコーダからパルスを入力して、パルスをカウントします。
RMC-RA4M1 ボードの端子を見て、GTIOCxA 端子と GTIOCxB 端子から2相のロータリエンコーダのパルスを入力することができます。(x=0～7)
今回は、CN8 の D71 端子(P106、GTIOC0B 端子)と D72 端子(P107、GTIOC0A 端子)を、パルス入力端子にしています。端子と GPT のチャンネルは関係なく、GPT は 0～7 のどれを使っても構いません。ただし、GPT のチャンネル 0 と1は、エンコーダ値を 0～2^{32}-1(4,294,967,295)までカウントすることができるのでチャネル 0 または 1 の使用がおすすめです。チャネル 2～7 は 0～2^{16}-1(65,535)までカウントすることができます。
今回のプログラムを下記に示します。

```
12 :    startGPT0_2SouEncoder( 1, 7 , 1, 6 );  // 2 相エンコーダ 0A=P107 0B=P106 を使用
                               ①  ②   ③  ④                    ①②      ③④
```

(2) startGPTx_2SouEncoder 関数の使い方(x＝0～7)

使い方	startGPTx_2SouEncoder(GTIOCxA のポート上位１桁, GTIOCxA のポート下位２回桁, GTIOCxB のポート上位１桁, GTIOCxB のポート下位２回桁); x にはチャネル番号 0～7 までの数字が入ります。実行すると GPT がスタートし 2 相のパルスをカウントします。
GTIOCxA のポート上位１桁	ロータリエンコーダの A 相を接続する、GTIOCxA のポートを設定します。 例えば P107 端子であれば、「1」になります。
GTIOCxA のポート下位２桁	ロータリエンコーダの A 相を接続する、GTIOCxA のポートを設定します。 例えば P107 端子であれば、「7」になります。 ※十の位が０の場合、０は省略してください。
GTIOCxB のポート上位１桁	ロータリエンコーダの B 相を接続する、GTIOCxB のポートを設定します。 例えば P106 端子であれば、「1」になります。
GTIOCxB のポート下位２桁	ロータリエンコーダの B 相を接続する、GTIOCxB のポートを設定します。 例えば P106 端子であれば、「6」になります。 ※十の位が０の場合、０は省略してください。
例	startGPT7_2SouEncoder(6 , 3 , 6 , 2); //2 相エンコーダ GPT7 P603(GTIOC7A),P602(GTIOC7B)を使用

(3) カウント値の読み込み方

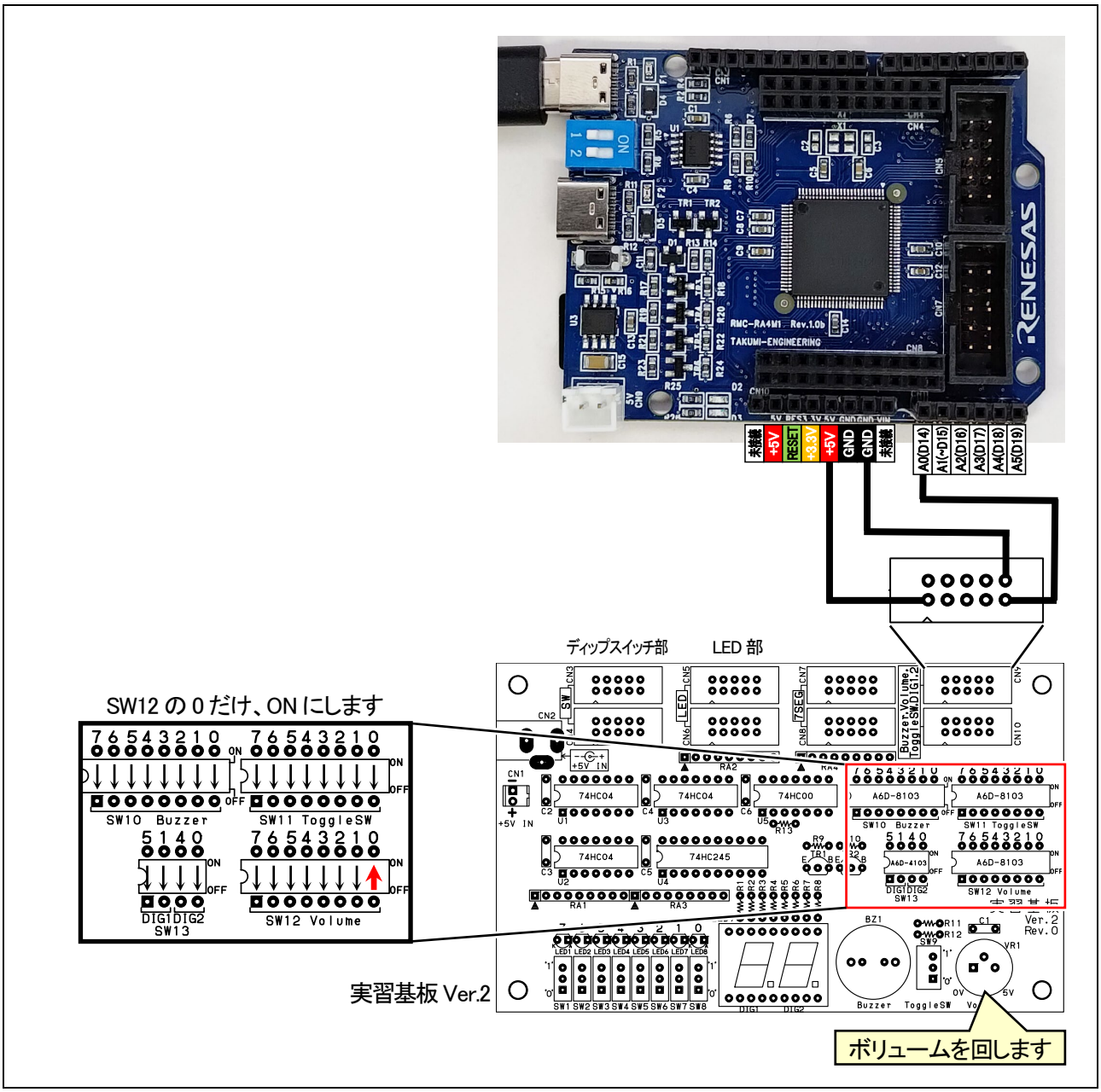
使い方	符号なしのとき：「GPTx_CNT」の値を読み込むと、カウント値が読み込まれます。 チャネル 0～1 は 0～2^{32}-1（4,294,967,295）の範囲でカウントすることができます。 チャネル 2～7 は 0～2^{16}-1（65,535）の範囲でカウントすることができます。 符号ありのとき：「INT_GPT0_CNT」の値を読み込むと、カウント値が読み込まれます。 チャネル 0～1 は-2,147,483,648 ～ +2,147,483,647 の範囲でカウントすることができます。 チャネル 2～7 は-32,768 ～ +32,767 の範囲でカウントすることができます。 x にはチャネル番号 0～7 までの数字が入ります。
例	i = INT_GPT0_CNT; // GPT0 のカウント値(符号有り)を変数 i に代入

4.13. 演習 13 A/D 変換「analogread.ino」

0～5V の電圧をデジタル値に A/D 変換します。

4.13.1. 配線

A0 端子と実習基板 Ver.2 のボリュームと接続します。
シリアル通信を行いますので、USB ケーブルを接続して、シリアルモニタを表示させておきます。



4.13.2. プログラム

```
 1 :  //*****************************************************************************
 2 :  // ﾌｧｲﾙ内容      「analogread.ino」A/D 変換(0～16,383)
 3 :  // Copyright    ジャパンマイコンカーラリー実行委員会
 4 :  // ライセンス   This software is released under the MIT License.
 5 :  //              http://opensource.org/licenses/mit-license.php
 6 :  //*****************************************************************************
 7 :
 8 :  void setup() {
 9 :    analogReadResolution( 14 ); // A/D 変換 14bit(0～16,383)
10 :    Serial.begin( 9600 );
11 :  }
12 :
```

A/D 変換の分解能を設定します。
設定値は、8、10、12、14 のいずれかです。
analogReadResolution 関数を実行しないときの初期値は 10 です。

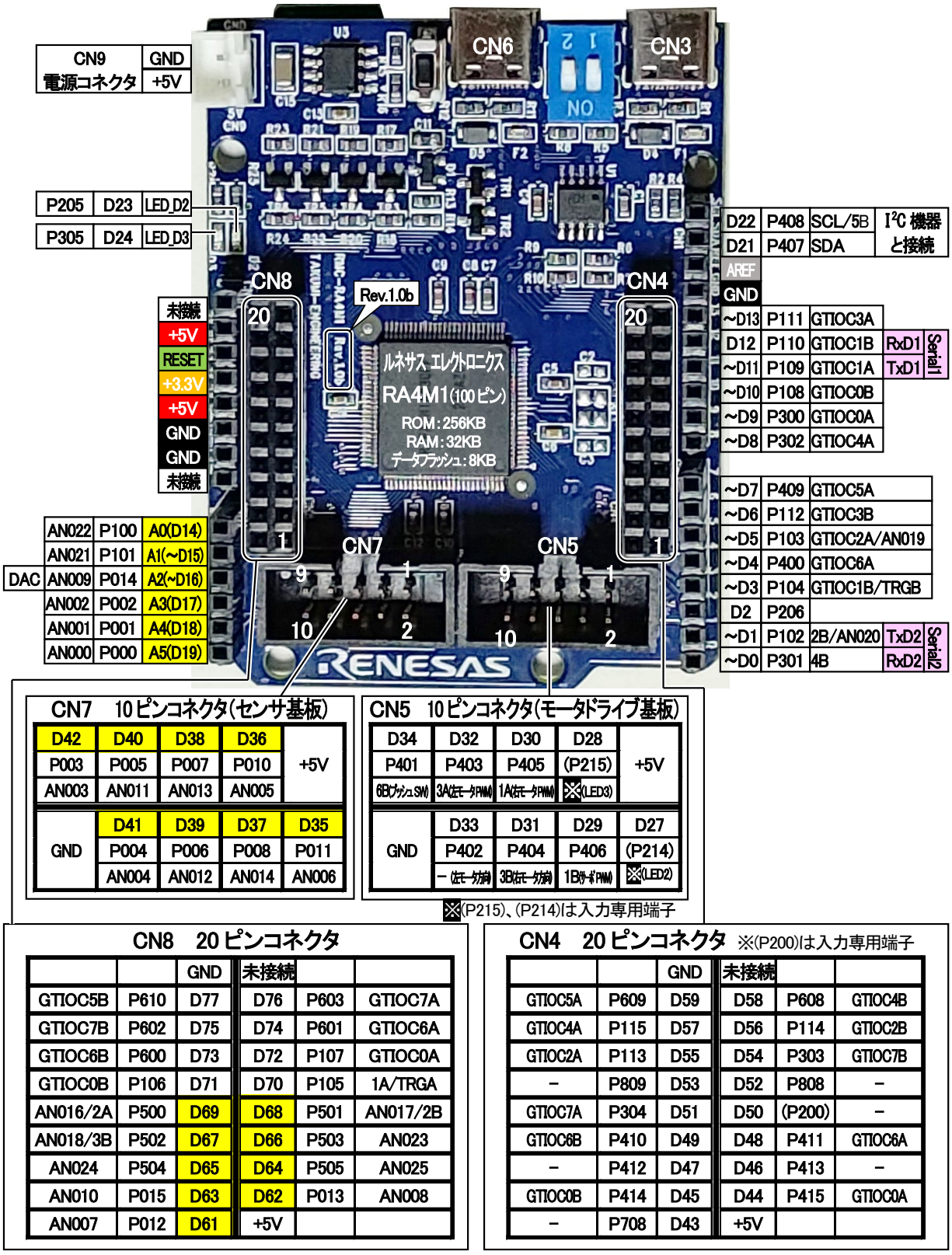
8 を設定したとき…0～5V を 0～(　　1111 1111)$_2$=0～　255 に変換します。
10 を設定したとき…0～5V を 0～(　 11 1111 1111)$_2$=0～ 1,023 に変換します。
12 を設定したとき…0～5V を 0～(　1111 1111 1111)$_2$=0～ 4,095 に変換します。
14 を設定したとき…0～5V を 0～(11 1111 1111 1111)$_2$=0～16,383 に変換します。

```
13 :  void loop() {
14 :    static int lp = 1;
15 :
16 :    Serial.print( lp );
17 :    Serial.print( " : " );
18 :    Serial.println( analogRead( A0 ) );  // A/D 変換結果をシリアル出力
19 :    delay( 1000 );
20 :    lp++;
21 :  }
```

analogRead 関数で A0 端子に入力されている電圧 0～5V を 0～16,383 に変換し、シリアルで出力します。出力は、
1 : 12345
2 : 9876
と、1000ms ごとにシリアルモニタに出力されます。
最初の数字は連番(1～)で、2 つ目の数字は A0 の A/D 値です。

4.13.3. プログラムの解説

下記の黄色部分がアナログ入力端子です。



CN7　10ピンコネクタ(センサ基板)

D42	D40	D38	D36	+5V
P003	P005	P007	P010	
AN003	AN011	AN013	AN005	
GND	D41	D39	D37	D35
	P004	P006	P008	P011
	AN004	AN012	AN014	AN006

CN5　10ピンコネクタ(モータドライブ基板)

D34	D32	D30	D28	+5V
P401	P403	P405	(P215)	
6B(プッシュSW)	3A(左モータPWM)	1A(右モータPWM)	※(LED3)	
GND	D33	D31	D29	D27
	P402	P404	P406	(P214)
	ー(左モータ方向)	3B(右モータ方向)	1B(サーボPWM)	※(LED2)

※(P215)、(P214)は入力専用端子

CN8　20ピンコネクタ

		GND	未接続		
GTIOC5B	P610	D77	D76	P603	GTIOC7A
GTIOC7B	P602	D75	D74	P601	GTIOC6A
GTIOC6B	P600	D73	D72	P107	GTIOC0A
GTIOC0B	P106	D71	D70	P105	1A/TRGA
AN016/2A	P500	D69	D68	P501	AN017/2B
AN018/3B	P502	D67	D66	P503	AN023
AN024	P504	D65	D64	P505	AN025
AN010	P015	D63	D62	P013	AN008
AN007	P012	D61	+5V		

CN4　20ピンコネクタ ※(P200)は入力専用端子

		GND	未接続		
GTIOC5A	P609	D59	D58	P608	GTIOC4B
GTIOC4A	P115	D57	D56	P114	GTIOC2B
GTIOC2A	P113	D55	D54	P303	GTIOC7B
ー	P809	D53	D52	P808	ー
GTIOC7A	P304	D51	D50	(P200)	ー
GTIOC6B	P410	D49	D48	P411	GTIOC6A
ー	P412	D47	D46	P413	ー
GTIOC0B	P414	D45	D44	P415	GTIOC0A
ー	P708	D43	+5V		

例えば、D35 端子の AD 値を読みたいときのプログラムは、下記のようになります。

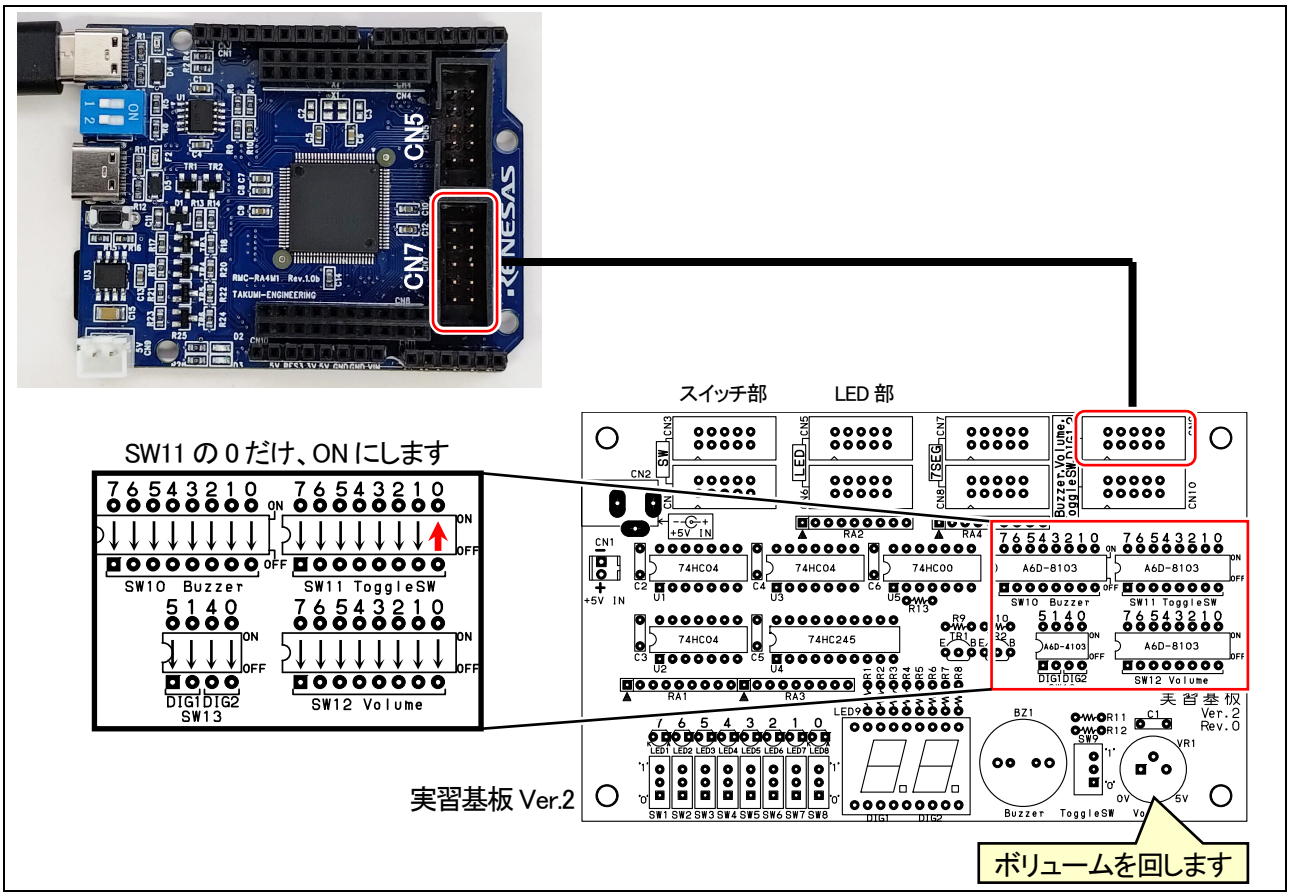
```
a = analogRead( D35 );
```

4.14. 演習 14 連続スキャンモードを使用した A/D 変換「ad_renzoku.ino」

マイコンの連続スキャンモードを使用して、常に複数の端子を A/D 変換します。専用ライブラリを使用するので、analogWrite 関数を使ったときより、読み込む時間が短くなります。

4.14.1. 配線

RMC-RA4M1 ボードの CN7 と実習基板 Ver.2 のボリュームと接続します。
シリアル通信を行いますので、USB ケーブルを接続して、シリアルモニタを表示させておきます。



4.14.2. プログラム

```
1 :  //********************************************************************************
2 :  // ﾌｧｲﾙ内容       「ad_renzoku.ino」A/D 変換 連続スキャンモード
3 :  // Copyright     ジャパンマイコンカーラリー実行委員会
4 :  // ライセンス    This software is released under the MIT License.
5 :  //               http://opensource.org/licenses/mit-license.php
6 :  //********************************************************************************
7 :
8 :  // インクルード宣言
9 :  #include "mcr_ad_lib.h"
10 :
11 :  // グローバル変数宣言
12 :  mcr_ad ad;
13 :
```

連続スキャンモードで A/D 変換できるライブラリをインクルードします。

mcr_ad クラスで ad インスタンスを作成します。

```
14 :  void setup() {
15 :    ad.useCh( 6 );  // D35 端子
16 :    ad.useCh( 5 );  // D36 端子
17 :    ad.useCh( 14 ); // D37 端子
18 :    ad.useCh( 13 ); // D38 端子
19 :    ad.useCh( 12 ); // D39 端子
20 :    ad.useCh( 11 ); // D40 端子
21 :    ad.useCh( 4 );  // D41 端子
22 :    ad.useCh( 3 );  // D42 端子
23 :    ad.start();
24 :
25 :    Serial.begin( 9600 );
26 :    while (!Serial) {
27 :      // 接続されるまで待つ
28 :    }
29 :    Serial.print( "¥n¥nA/D 変換 連続スキャンモード¥n" );
30 :  }
31 :
32 :  void loop() {
33 :    char buff[16];
34 :
35 :    digitalWrite( 24, 0 );
36 :    sprintf( buff, "AN006=%5d  ", getAD_006 );
37 :    Serial.print( buff );
38 :    sprintf( buff, "AN005=%5d  ", getAD_005 );
39 :    Serial.print( buff );
40 :    sprintf( buff, "AN014=%5d  ", getAD_014 );
41 :    Serial.print( buff );
42 :    sprintf( buff, "AN013=%5d  ", getAD_013 );
43 :    Serial.print( buff );
44 :    sprintf( buff, "AN012=%5d  ", getAD_012 );
45 :    Serial.print( buff );
46 :    sprintf( buff, "AN011=%5d  ", getAD_011 );
47 :    Serial.print( buff );
48 :    sprintf( buff, "AN004=%5d  ", getAD_004 );
49 :    Serial.print( buff );
50 :    sprintf( buff, "AN003=%5d  ", getAD_003 );
51 :    Serial.print( buff );
52 :    Serial.print( "¥n" );
53 :    digitalWrite( 24, 1 );
54 :
55 :    delay( 1000 );
56 :  }
```

連続スキャンモードで使用する端子を設定します。6 = AN006 端子のことです。

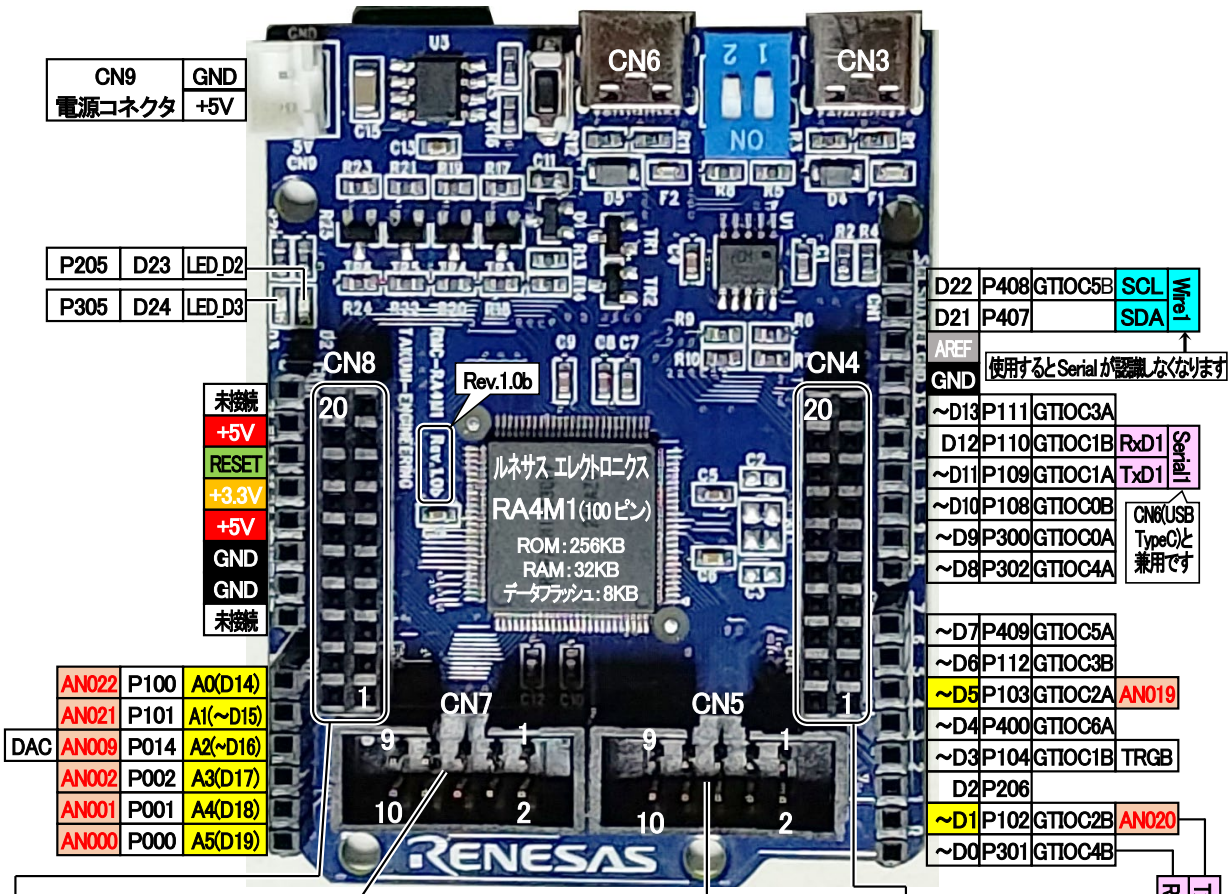
使用する端子を設定したのち、連続スキャンモードで A/D 変換をスタートします。

連続スキャンモードの A/D 値の読み込みは、getAD_xxx を使います。xxx は端子番号です。AN006 端子なら、「getAD_006」から読み込みます。

Serial.print で、CN7 の 8 端子分の A/D 値をシリアルモニタに出力します。

4.14.3. プログラムの解説

下記のオレンジ色部分がアナログ入力端子のチャネル番号です。プログラムでは、チャネル番号(AN022 など)を指定します。A0 などの端子番号は使用しません。



CN7　10ピンコネクタ(センサ基板)

D42	D40	D38	D36	+5V
P003	P005	P007	P010	
AN003	AN011	AN013	AN005	
GND	D41	D39	D37	D35
	P004	P006	P008	P011
	AN004	AN012	AN014	AN006

CN5　10ピンコネクタ(モータドライブ基板)

D34	D32	D30	D28	+5V
P401	P403	P405	(P215)	
6B(プッシュSW)	3A(左モータPWM)	1A(右モータPWM)	※(LED3)	
GND	D33	D31	D29	D27
	P402	P404	P406	(P214)
	ー(左モータ方向)	3B(右モータ方向)	1B(サーボPWM)	※(LED2)

※(P215)、(P214)は入力専用端子

CN8　20ピンコネクタ

		GND	未接続		
GTIOC5B	P610	D77	D76	P603	GTIOC7A
GTIOC7B	P602	D75	D74	P601	GTIOC6A
GTIOC6B	P600	D73	D72	P107	GTIOC0A
GTIOC0B	P106	D71	D70	P105	1A/TRGA
AN016/2A	P500	D69	D68	P501	AN017/2B
AN018/3B	P502	D67	D66	P503	AN023
AN024	P504	D65	D64	P505	AN025
AN010	P015	D63	D62	P013	AN008
AN007	P012	D61	+5V		

CN4　20ピンコネクタ　※(P200)は入力専用端子

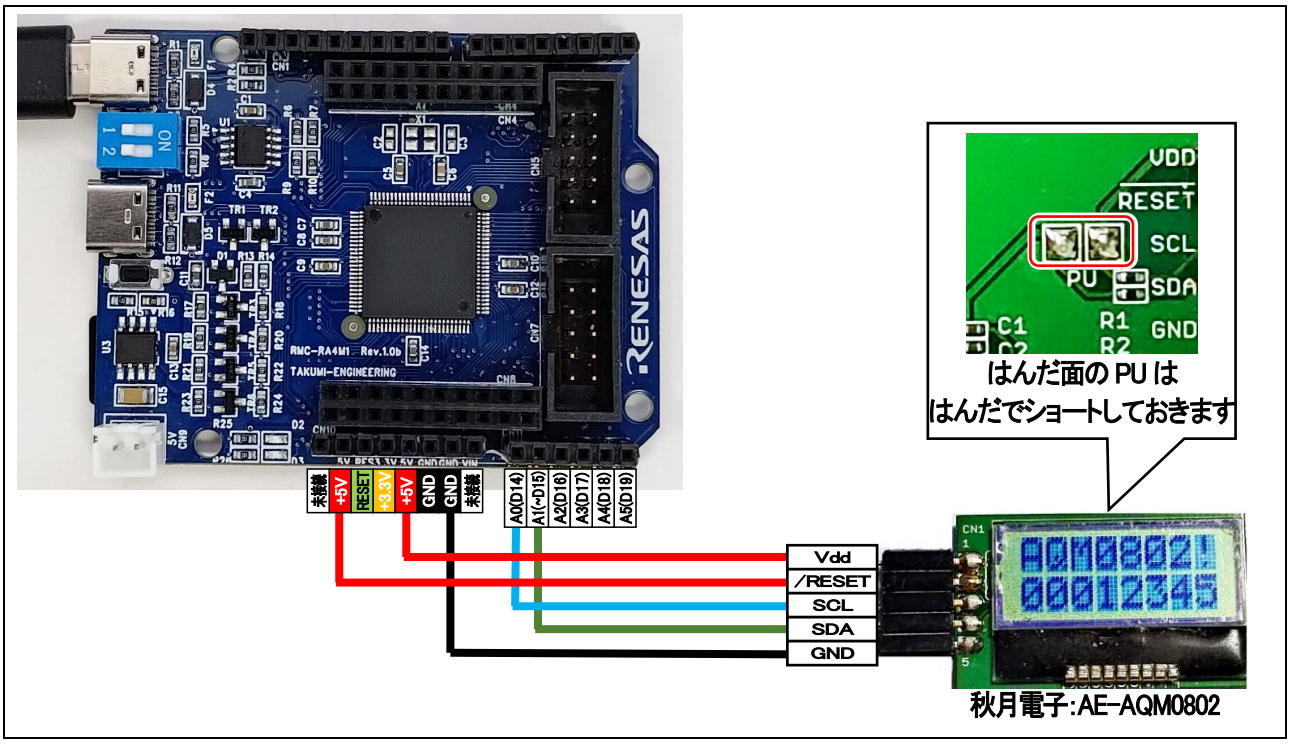
		GND	未接続		
GTIOC5A	P609	D59	D58	P608	GTIOC4B
GTIOC4A	P115	D57	D56	P114	GTIOC2B
GTIOC2A	P113	D55	D54	P303	GTIOC7B
ー	P809	D53	D52	P808	ー
GTIOC7A	P304	D51	D50	(P200)	ー
GTIOC6B	P410	D49	D48	P411	GTIOC6A
ー	P412	D47	D46	P413	ー
GTIOC0B	P414	D45	D44	P415	GTIOC0A
ー	P708	D43	+5V		

4.15. 演習 15 I2C 液晶 AQM0802 の表示「aqm0802.ino」

I2C 液晶の AQM0802 液晶(秋月電子製:AE-AQM0802 基板など)に文字を表示します。

4.15.1. 配線

RMC-RA4M1 ボードの A0 端子と液晶の SCL 端子、A1 端子と SDA 端子を接続します。



4.15.2. プログラム

```
 1 :  //**************************************************************************
 2 :  // ﾌｧｲﾙ内容      「aqm0802.ino」AQM0802 液晶(I2C)
 3 :  // Copyright    ジャパンマイコンカーラリー実行委員会
 4 :  // ライセンス   This software is released under the MIT License.
 5 :  //              http://opensource.org/licenses/mit-license.php
 6 :  //**************************************************************************
 7 :
 8 :  // インクルード宣言
 9 :  #include "mcr_aqm0802.h"  // A0:SCL A1:SDA
10 :
11 :  // グローバル変数宣言
12 :  MCR_AQM0802 lcd;          // MCR_AQM0802 クラス
13 :
14 :  void setup() {
15 :    lcd.begin( 11 );        // AQM0802 液晶使用開始 コントラスト=11
16 :  }
17 :
18 :  void loop() {
19 :    static int lp = 1;
20 :
21 :    lcd.setPosition( 0,0 );
22 :    lcd.printf( "AQM0802!" );
23 :    lcd.setPosition( 0,1 );
24 :    lcd.printf( "%08d", lp );
25 :    delay(10);
26 :    lp++;
27 :  }
```

I2C 液晶 AQM0802 を制御するライブラリを読み込みます。

MCR_AQM0802 クラスで lcd インスタンスを作成します。

液晶を初期化します。カッコの中には、液晶のコントラストを設定します。
設定値は 1～63 です。1 が薄く（ほぼ見えません)、63 が濃く(真っ黒になります)なります。5V の場合、実験値で 11 としました（個体差で違うことがありますので、微調整をお願いします)。

「lcd.setPosition(x, y);」で表示する位置を指定します。
x=0～7、y=0～1 で、左上が、x=0、y=0 です。

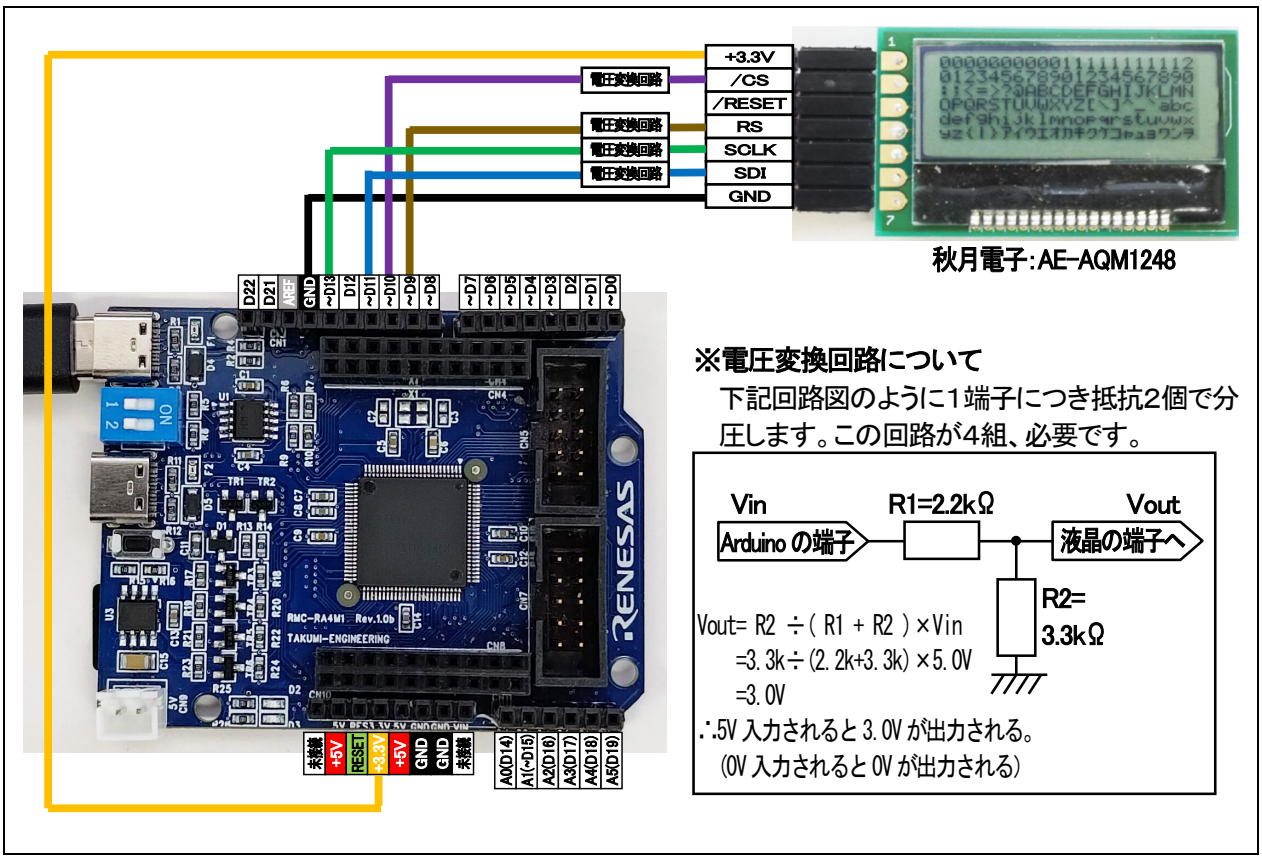
Arduino の print の書式ではなく、
C 言語の **printf** の書式となります。

4.16. 演習 16 I2C グラフィック液晶 AQM1248 の文字表示「aqm1248.ino」

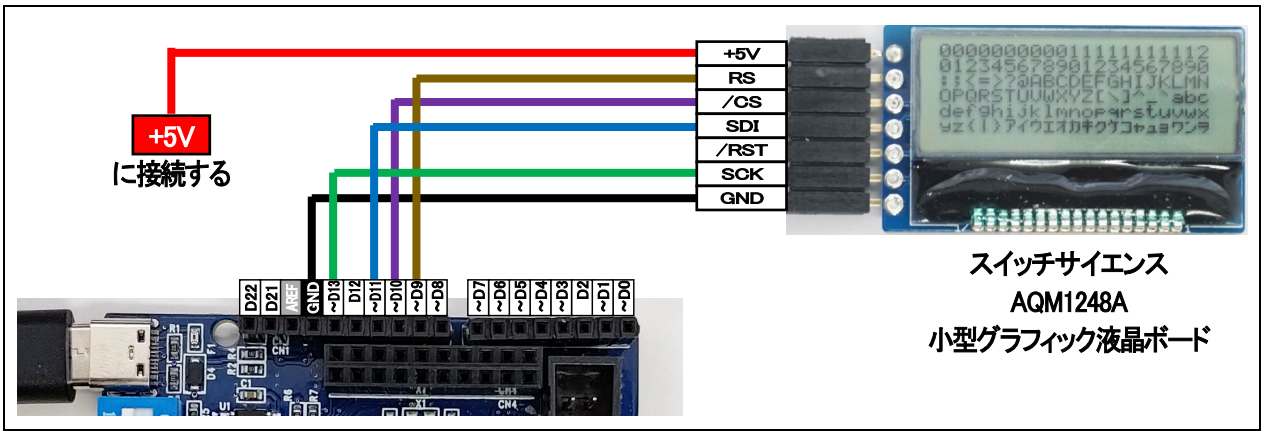
I2C グラフィック液晶の AQM1248 液晶（スイッチサイエンス：AQM1248A 小型グラフィック液晶ボード、または秋月電子製：AE-AQM 1248 基板など）に文字を表示します。ただし、秋月電子製の基板は、3.3V で動作するため、端子に 5V を加えると壊れてしまいます。秋月電子製の基板は、電圧変換用抵抗が必要になります。スイッチサイエンス製の基板は、5V で動作するため、電圧変換用抵抗は必要ありません。
※秋月電子 AE-AQM1248 ： https://akizukidenshi.com/catalog/g/gK-07007/
※スイッチサイエンス SSCI-026086 ： https://www.switch-science.com/products/2608

4.16.1. 配線

RMC-RA4M1 ボードの D9、D10、D11、D13 端子と、AE-AQM1248 基板を、下図のように接続します。



スイッチサイエンスの AQM1248A 小型グラフィック液晶ボードは、下図のように接続します。電源は 5V に接続します。



4.16.2. プログラム

```
 1 :  //************************************************************************
 2 :  // ﾌｧｲﾙ内容      「aqm1248.ino」I2C グラフィック液晶 AQM1248 の制御（文字）
 3 :  // Copyright    ジャパンマイコンカーラリー実行委員会
 4 :  // ライセンス   This software is released under the MIT License.
 5 :  //              http://opensource.org/licenses/mit-license.php
 6 :  //************************************************************************
 7 :
 8 :  // インクルード宣言
 9 :  #define SPI SPI1              // RMC-RA4M1 rev.1.0b は SPI ではなく SPI1 を使用
10 :  #include <MGLCD.h>            // 出典；しなぷすのハード製作記 https://synapse.kyoto/lib/MGLCD/page001.html
11 :  #include <MGLCD_SPI.h>
12 :  #include <SPI.h>
13 :
14 :  // グローバル変数の宣言
15 :  MGLCD_AQM1248A_SPI MGLCD( MGLCD_SpiPin2(10, 9), 100000L );  // (10=CS ピン,9=RS ピン), MAX_FREQ の設定
16 :
17 :  void setup()
18 :  {
19 :    // LCD の初期化
20 :    while( MGLCD.Reset() );          // CS=10pin  RS=9pin  SCLK=13pin  SDI=11pin
21 :    if( strlen("ｱ") != 1 ) MGLCD.SetCodeMode( MGLCD_CODE_UTF8 );  // 半角カナ表示の有効化
22 :    MGLCD.SetVolumeResistor( 28 );  // コントラストの調整 0～63
23 :    MGLCD.ClearScreen();
24 :
25 :    MGLCD.Locate( 0, 0 );
26 :    MGLCD.print( "00000000011111111112" );
27 :    MGLCD.print( "01234567890123456789" );
28 :
29 :    for( char i = ':'; i<= '}'; i++ ) {
30 :      MGLCD.print( i );
31 :    }
32 :    MGLCD.print( "ｱｲｳｴｵｶｷｸｹｺｬｭｮﾜﾝﾗ" );        // 半角カタカナ表示
33 :  }
34 :
35 :  void loop()
36 :  {
37 :    // 特に何もしない
38 :  }
```

I2C グラフィック液晶 AQM1248 を制御するライブラリを読み込みます。「しなぷすのハード製作記」に掲載されているライブラリを使わせていただきました。すばらしいライブラリの公開、ありがとうございます。

CS ピンと RS ピンはここで指定できます。
CS ピンは 10 ピン、RS ピンは 9 ピンに指定します。
SCK ピンと SDI ピンは、変更できません(SPI の機能を使います)。

横 21 文字、縦 6 文字表示することができます。
Locate で表示する位置を指定でき、0 文字目、0 行目から始まります。
左上が(0,0)、右下が(20,5)となります。

4.16.3. プログラムの解説

プログラムを実行すると、右写真のように横21文字、縦6文字分、表示されます。
I2C グラフィック液晶 AQM1248 を制御するライブラリは、「しなぷすのハード製作記」にあるライブラリを使わせていただきました。すばらしいライブラリの公開をありがとうございます。
ライブラリの使い方は、下記を参照してください。

https://synapse.kyoto/hard/MGLCD_AQM1248A/page001.html

4.17. 演習 17 I2C グラフィック液晶 AQM1248 のグラフィック表示「aqm1248graphic.ino」

演習 16 の AQM1248 液晶に、グラフィックを表示します。
演習 16 と同様に、「しなぷすのハード製作記」にあるライブラリを使わせていただきました。すばらしいライブラリの公開をありがとうございます。
グラフィック関係は、下記のホームページを参考にさせていただきました。
https://synapse.kyoto/hard/MGLCD_AQM1248A/page016.html

4.17.1. 配線

演習 14 と同じです。

4.17.2. プログラム

```
1 ： //*****************************************************************************
2 ： // ﾌｧｲﾙ内容      「aqm1248graphic.ino」I2C グラフィック液晶 AQM1248 の制御(グラフィック)
3 ： // Copyright    ジャパンマイコンカーラリー実行委員会
4 ： // ライセンス   This software is released under the MIT License.
5 ： //              http://opensource.org/licenses/mit-license.php
6 ： //*****************************************************************************
7 ：
8 ： // インクルード宣言
9 ： #define SPI SPI1              // RMC-RA4M1 rev.1.0b は SPI ではなく SPI1 を使用
10 ： #include <MGLCD.h>            // 出典；しなぷすのハード製作記  https://synapse.kyoto/lib/MGLCD/page001.html
11 ： #include <MGLCD_SPI.h>
12 ： #include <SPI.h>
13 ：
14 ： // グローバル変数の宣言
15 ： MGLCD_AQM1248A_SPI MGLCD( MGLCD_SpiPin2(10, 9), 100000L );  // (10=CS ピン,9=RS ピン), MAX_FREQ の設定
16 ：
17 ： void setup()
18 ： {
19 ：   // LCD の初期化
20 ：   while( MGLCD.Reset() );          // CS=10pin  RS=9pin  SCLK=13pin  SDI=11pin
21 ：   if( strlen("ｱ") != 1 ) MGLCD.SetCodeMode( MGLCD_CODE_UTF8 );  // 半角カナ表示の有効化
22 ：   MGLCD.SetVolumeResistor( 28 );  // コントラストの調整 0～63
23 ：   MGLCD.ClearScreen();
24 ： }
25 ：
26 ： void loop()
27 ： {
28 ：   // 点を打つ
29 ：   MGLCD.SetPixel( 10, 10 );          // x , y , color(1:点灯 0:消灯 省略時は 1 になる)
30 ：   MGLCD.SetPixel( 11, 11 );
31 ：   MGLCD.SetPixel( 12, 12 );
32 ：   delay( 500 );
33 ：
34 ：   // 線を引く
35 ：   MGLCD.Line( 20, 10, 20, 40 );      // 始点x , 始点y , 終点x , 終点y , color(1:点灯 0:消灯 省略時は1になる)
36 ：   delay( 500 );
37 ：
```

SetPixel 関数で、指定した座標に点を打つことができます。
3 個目の引数は、1 で点灯、0 で消灯させます。省略すると点灯になります。今回のプログラムは省略しているので点灯させます。

Line 関数で、(始点 x,始点 y)と(終点 x,終点 y)を結ぶ直線を引きます。5 個目の引数は、1 で点灯、0 で消灯、省略すると点灯になります。

```
38 :    // 長方形を描く(塗りつぶし無し)
39 :    MGLCD.Rect( 30, 10, 40, 20 );      // 始点x , 始点y , 終点x , 終点y , color(1:点灯 0:消灯 省略時は1になる)
40 :    delay( 500 );
41 :
```

Rect 関数で、(始点 x,始点 y)と(終点 x,終点 y)を対角とする長方形を描きます。5 個目の引数は、1 で点灯、0 で消灯、省略すると点灯になります。長方形の中は塗りつぶしません(外形線のみ描きます)。

```
42 :    // 長方形を描く(塗りつぶし有り)
43 :    MGLCD.FillRect( 50, 30, 60, 40 ); // 始点x , 始点y , 終点x , 終点y , color(1:点灯 0:消灯 省略時は1になる)
44 :    delay( 500 );
45 :
```

FillRect 関数で、(始点 x,始点 y)と(終点 x,終点 y)を対角とする長方形を描き、長方形の中を塗りつぶします。5 個目の引数は、1 で点灯、0 で消灯、省略すると点灯になります。

```
46 :    // 円を描く(塗りつぶし無し)
47 :    MGLCD.Circle( 80, 20, 10 );        // x , y , 半径r , color(1:点灯 0:消灯 省略時は1になる)
48 :    delay( 500 );
49 :
```

Circle 関数で、(x,y)を中心とする、半径 r の円を描きます。円の中は塗りつぶしません。4 個目の引数は、1 で点灯、0 で消灯、省略すると点灯になります。

※真円しか描けませんが、横：縦の比率が 1:0.864 なので横長の楕円になってしまいます

```
50 :    // 円を描く(塗りつぶし有り)
51 :    MGLCD.FillCircle( 110, 20, 10 );  // x , y , 半径 , color(1:点灯 0:消灯 省略時は1になる)
52 :    delay( 5000 );
53 :
```

FillCircle 関数で、(x,y)を中心とする、半径 r の円を描き、円の中を塗りつぶします。4 個目の引数は、1 で点灯、0 で消灯、省略すると点灯になります。

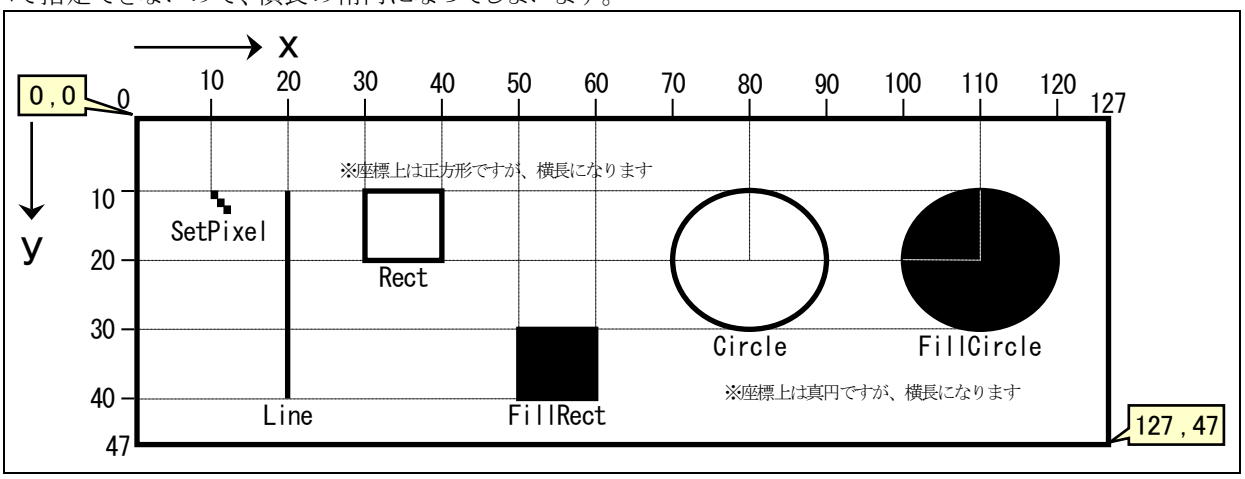
※真円しか描けませんが、横：縦の比率が 1:0.864 なので横長の楕円になってしまいます

```
54 :    MGLCD.ClearScreen();               // 画面クリア
55 :    delay( 500 );
56 :  }
```

4.17.3. プログラムの解説

AQM1248 液晶は、横 128 ピクセル、縦 48 ピクセルです。座標は左上が(x=0 , y=0)、右下が(x=127 , y=47)になります。下図に今回のプログラムを実行ときのイメージを示します。

横:縦の比率が 1:0.864 なので、横に長くなります。プログラム上は正方形を描いても横長に、真円を描いても横長の楕円になります。四角は縦横比を計算し正方形に見えるようにプログラムできますが、円は真円しかプログラムで指定できないので、横長の楕円になってしまいます。



4.18. 演習 18　EEPROM(データフラッシュメモリ)「eeprom.ino」

マイコン内蔵の EEP-ROM(Electrically Erasable Programmable Read-Only Memory)にパラメータを書き込み、読み込みます。

EEP-ROM は、利用者が内容を書き換え可能な ROM なので、電源を切っても消えません。パラメータなどを保存しておけます。RA マイコンは、EEP-ROM のことを、データフラッシュメモリと呼びます。

RA マイコンの EEP-ROM は、標準で 1,000,000 回(最低保障回数 100,000 回)、書き込みを行うことができます。

容量は、8KB(8192 バイト)あり、0～8191 番地に値を読み書きできます。 ※型によって書き込めるパラメータ数は代わります。例えば、int 型は 4 バイトなので、2048 個のデータを保存できます。

4.18.1. 配線

特にありません。マイコン内部の EEP-ROM に書き込んでいるパラメータを呼び出し、＋1した値を書き込みます。

4.18.2. プログラム

```
 1 : //**************************************************************************
 2 : // ﾌｧｲﾙ内容      「eeprom.ino」データフラッシュメモリ(EEP-ROM)に保存、読み込み
 3 : // Copyright    ジャパンマイコンカーラリー実行委員会
 4 : // ライセンス   This software is released under the MIT License.
 5 : //              http://opensource.org/licenses/mit-license.php
 6 : //**************************************************************************
 7 : /*
 8 : // ※RA マイコンは、EEP-ROM のことを、データフラッシュメモリと呼びます。
 9 :    標準で 1,000,000 回 (最低保障回数 100,000 回)、書き込みを行うことができます。
10 : */
11 :
12 : // インクルード
13 : #include <EEPROM.h>
14 :
15 : // シンボル定義
16 : #define EEPROM_CHECK ((int)0x20240201)  // EEP-ROM にデータが書き込まれているかチェック用
17 :
18 : void setup() {
19 :   int i;
20 :
21 :   Serial.begin(9600);
22 :   while ( !Serial ) {
23 :     // 接続されるまで待つ
24 :   }
25 :   Serial.print( "¥n¥n" );
26 :
27 :   EEPROM.get( 0, i ); // 0 番地から値読み込み
28 :
29 :   if( i != EEPROM_CHECK ) {
30 :     //もしデータ書き込まれていないなら
31 :     Serial.print( "EEP-ROM を初期化します...  " );
32 :     EEPROM.put( 0 , EEPROM_CHECK ); // 0 番地  チェック用データ  書き込み
33 :     EEPROM.put( 4 , (int)1 );       // 4 番地  データ  書き込み
34 :     Serial.println( "初期化しました。" );
```

EEP-ROM のライブラリを読み込みます。

0～3 番地に 0x20240201 が書き込まれていれば、これから読み込む番地に過去に書き込んだ値があると判断します。
なければ、不定の値が入っていると判断し、使う番地の値を初期化します。

0～3 番地に 0x20240201 が書き込まれていないときの処理です。

```
35 :    } else {
36 :      // データが書き込まれていたら読み込み、＋１した値を書き込む
37 :      EEPROM.get( 4 , i );
38 :      Serial.print( "4番地の値は " );
39 :      Serial.print( i );
40 :      Serial.println( " です。" );
41 :
42 :      i++;
43 :      EEPROM.put( 4 , i );    // 4番地　データ　書き込み　次実行すると値は＋１されている
44 :    }
45 :  }
46 :
47 :  void loop() {
48 :    // 何もしない
49 :  }
```

前回書き込んだ 4～7 番地の値(int)を読み込み、シリアル出力します。

読み込んだ値を＋1して、書き込みます。

4.18.3. プログラムの解説

EEP-ROM へ書き込みは、put 関数を使います。型によって EEP-ROM に何バイト書き込むか変わります。例えば 0 番地に int 型で書き込んだ場合、0～3 番地に書き込まれます。違う値を書き込む場合は 4 番地以降を使います。型とバイト数の関係は「6.1. 型指定子の範囲について」を参照してください。

```
  EEPROM.put( 番地(0～8191) , 書き込む値 );

例) EEPROM.put( 0 , 5 );  // 0番地～3番地に5を書き込む
                          数値は何も指定しないと int 型になり 4 バイトの値になりアドレスを４つ使用します
    EEPROM.put( 4, (float)1.2345 );  // 4～7 番地に float 型で 1.2345 を書き込む
    EEPROM.put( 8, (char)0xff );     // 8番地に 0xff を書き込む。char 型は 1 バイトなので 8 番地しか使いません
```

EEP-ROM からの読み込みは、get 関数を使います。

```
  EEPROM.get( 番地(0～8191) , 読み込んだ値を書き込む変数 ) ;

例) int i;
    float f;
    char c;
    EEPROM.get( 0 , i );   // 0～3 番地の int 値を変数 i に読み込む
    EEPROM.get( 4 , f );   // 4～7 番地の float 値を変数 f に読み込む
    EEPROM.get( 8 , c );   // 8 番地の char 値を変数 c に読み込む
```

put 関数、get 関数の実行時間を測定しました。測定方法は D13 端子を"1"にして関数を実行、その後 D13 端子を"0"にして、オシロスコープで"1"の時間を測定しました。

EEPROM.put(0 , 0xffffffff); // 0xff を書き込む
delay(50);
digitalWrite(13, 1);
EEPROM.put(0 , 0x12345678); **// 実行時間 44ms**
digitalWrite(13, 0);
	EEPROM.put(0 , 0xffffffff);　　// 0xff を書き込む
	delay(50);
	digitalWrite(13, 1);
	EEPROM.get(0 , i); **// 実行時間 120μs**(i は unsigned int 型)
	digitalWrite(13, 0);

EEPROM.put(0 , (unsigned char)0xff); // 0xff を書き込む
delay(50);
digitalWrite(13, 1);
EEPROM.put(0 , (unsigned char)0xff); **// 実行時間 44ms**
digitalWrite(13, 0);
	EEPROM.put(0 , (unsigned char)0xff); // 0xff を書き込む
	delay(50);
	digitalWrite(13, 1);
	EEPROM.get(0 , c); **//実行時間 30.5μs**(c は unsigned char 型)
	digitalWrite(13, 0);

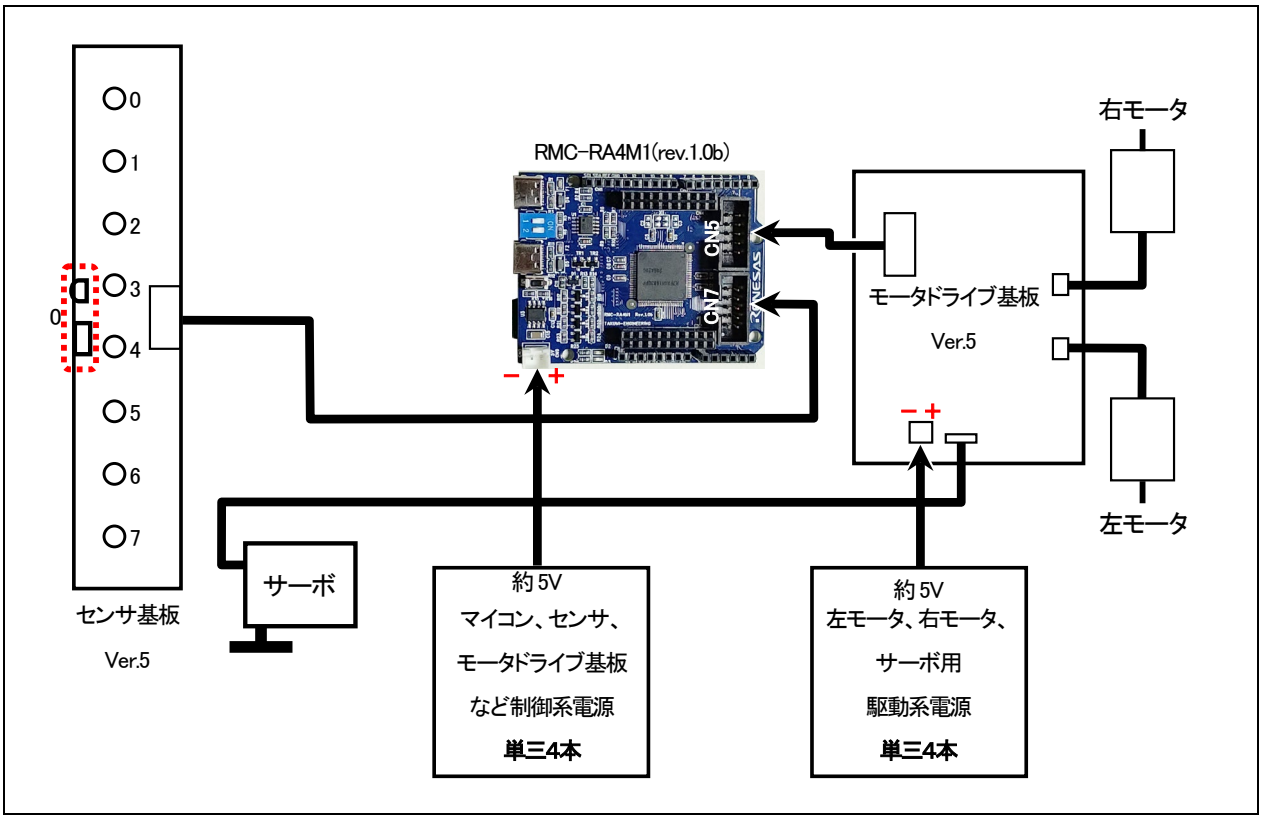
書き込みは実測で、int 型(4 バイト)で 44ms、unsigned char 型(1バイト)で 44ms かかりました。
読み込みは実測で、int 型(4 バイト)で 120μs、unsigned char 型(1バイト)で 30.5μs かかりました。

5. マイコンカー走行プログラム

5.1. ベーシッククラスマイコンカー サーボセンタ・サーボ確認プログラム「kit23_ra4m1_servo_tyosei」

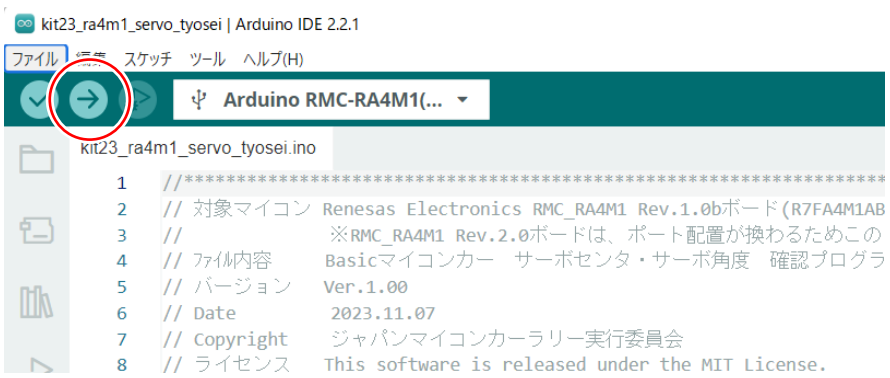
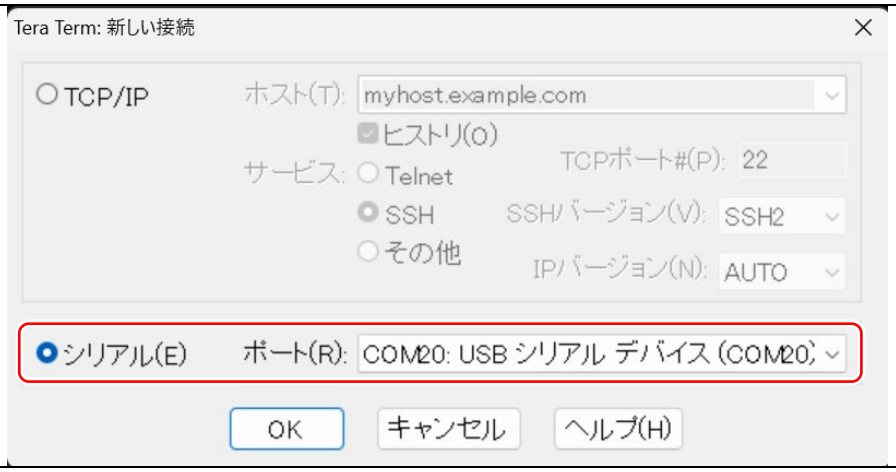
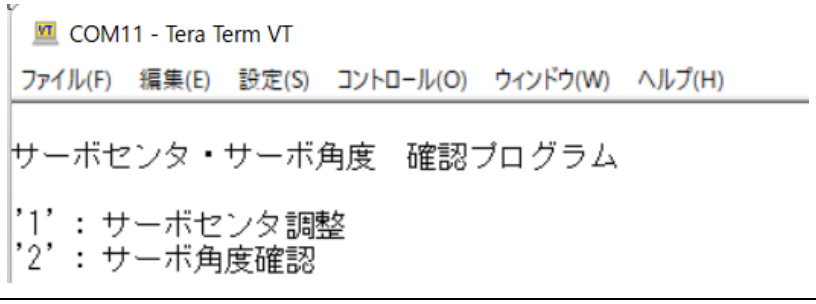
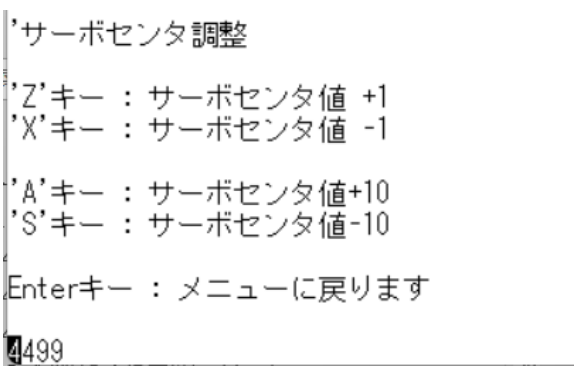
5.1.1. 配線

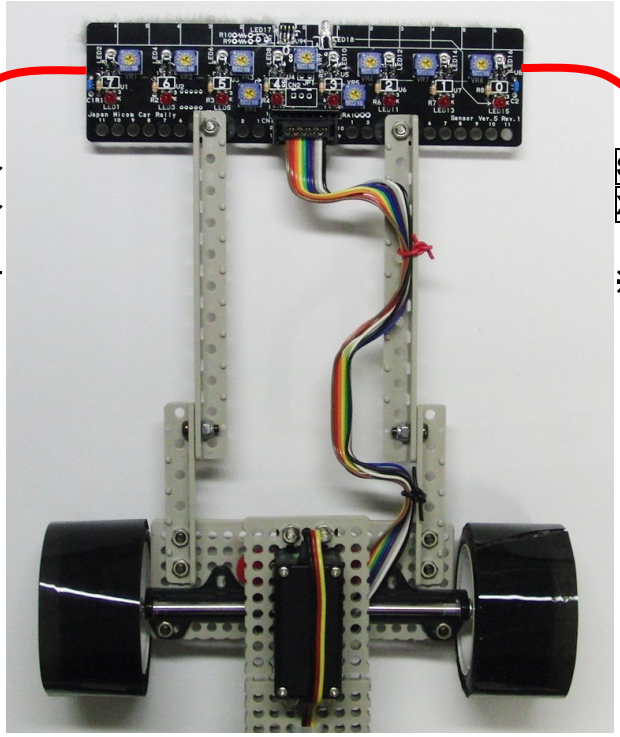
ベーシッククラスマイコンカーのサーボセンタと角度調整を確認するプログラムです。
RMC-RA4M1 ボードの CN5 とモータドライブ基板 Ver.5、CN7 とセンサ基板 Ver.5 を接続します。



5.1.2. 調整の仕方

※通信ソフト「TeraTerm」(https://ttssh2.osdn.jp/)を使用します。パソコンに入っていない場合は、あらかじめインストールをお願いします。ArduinoIDE のシリアルモニタでは、一回一回改行しないといけないので、実用的ではありません。

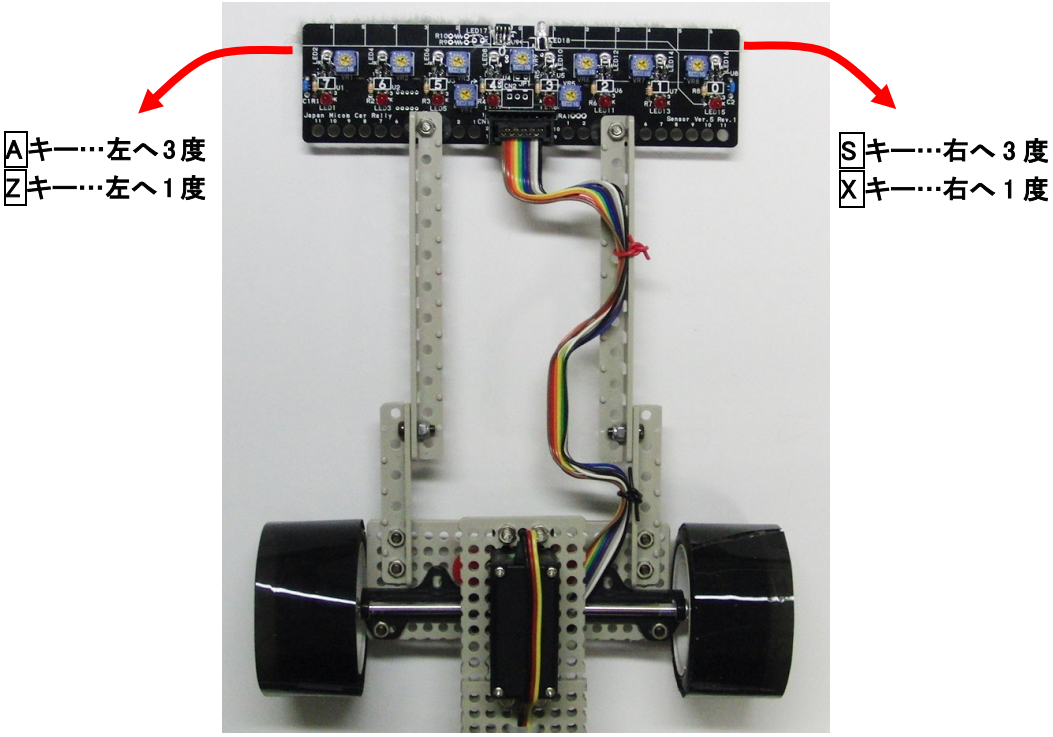
1		「 kit23_ra4m1_servo_tyosei.ino 」を開き、RMC-RA4M1 ボードにプログラムを書き込みます。
2		TeraTerm を立ち上げます。 「シリアル」を選び、ポートは、RMC-RA4M1 ボードが接続されているポートを選びます。
3		正しく接続されると、左画面のように表示されます。 数字の「1」キーを押すと、サーボセンタ調整を行えます。 「2」キーを押すと、サーボ角度調整を行います。 まず、サーボセンタ調整を行うので、「1」キーを押します。
4		左のように表示されます。

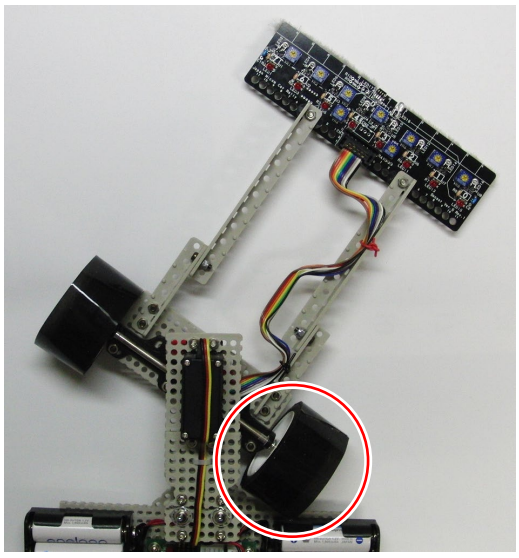
5	 A、S、Z、X キーをそれぞれ押し続けるとサーボが動きます。**キーを使ってサーボがまっすぐ向く角度に調整してください。** ※もし左右が逆に動く場合は、「kit23_ra4m1_msd.ino」プログラムの一部を変更する必要があります。詳しくは、「kit23_ra4m1_msd.ino」プログラムの解説を参照して下さい。

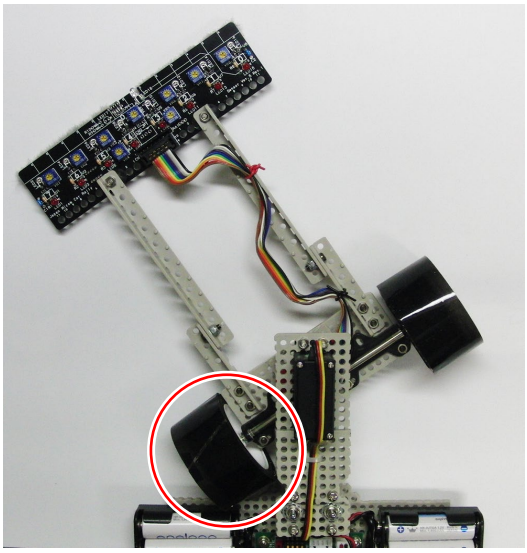
6	COM11 - Tera Term VT ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H) 1 'サーボセンタ調整 'Z'キー : サーボセンタ値 +1 'X'キー : サーボセンタ値 -1 'A'キー : サーボセンタ値+10 'S'キー : サーボセンタ値-10 Enterキー : メニューに戻ります 4469	まっすぐに調整できたら、Tera Term の数字を見ます。この値がこのマイコンカーの**サーボセンタ（SERVO_CENTER）**です。メモしておいてください。 エンターキーを押して、メニューに戻ります。

7	COM11 - Tera Term VT ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H) サーボセンタ・サーボ角度 確認プログラム '1' : サーボセンタ調整 '2' : サーボ角度確認	サーボ角度確認を行うので、「2」キーを押します。

8	COM11 - Tera Term VT ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H) 'サーボ角度確認 'Z'キー : サーボ角度 +1 'X'キー : サーボ角度 -1 'A'キー : サーボ角度 +3 'S'キー : サーボ角度 -3 Enterキー : メニューに戻ります	左のように表示されます。

9	 A、S、Z、X キーをそれぞれ押すとサーボが動きます。**右はどこまでハンドルを曲げることができるかを調べます。左も同様に調べます。**

10		まず S キー、X キーで右の限界を見つけます。タイヤを手で回して、シャーシにぶつからないか確かめてください。シャーシにぶつかるようなら Z キーでもう少し小さくしてください。

11	COM11 - Tera Term VT ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H) 'サーボ角度確認 'Z'キー : サーボ角度 +1 'X'キー : サーボ角度 -1 'A'キー : サーボ角度 +3 'S'キー : サーボ角度 -3 Enterキー : メニューに戻ります 40	Tera Term の数値を見ます。これが現在ハンドルを右へ曲げている角度です。**40 度曲げていると言うことが分かりました。** **右が 40 度だからといって左が-40 度とは限りません。必ず左右確かめます。**
12		今度は A キー、Z キーで逆の左の限界を見つけます。タイヤを手で回して、シャーシにぶつからないか確かめてください。シャーシにぶつかるようなら X キーでもう少し小さくしてください。
13	COM11 - Tera Term VT ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H) 'サーボ角度確認 'Z'キー : サーボ角度 +1 'X'キー : サーボ角度 -1 'A'キー : サーボ角度 +3 'S'キー : サーボ角度 -3 Enterキー : メニューに戻ります -38	Tera Term の数値を見ます。これが現在ハンドルを左へ曲げている角度です。**-38 度曲げていると言うことが分かりました。** 次は、走行プログラム「kit23_ra4m1_msd.ino」を書き込み、マイコンカーを走らせます。 **Tera Term は終了しておきます。**

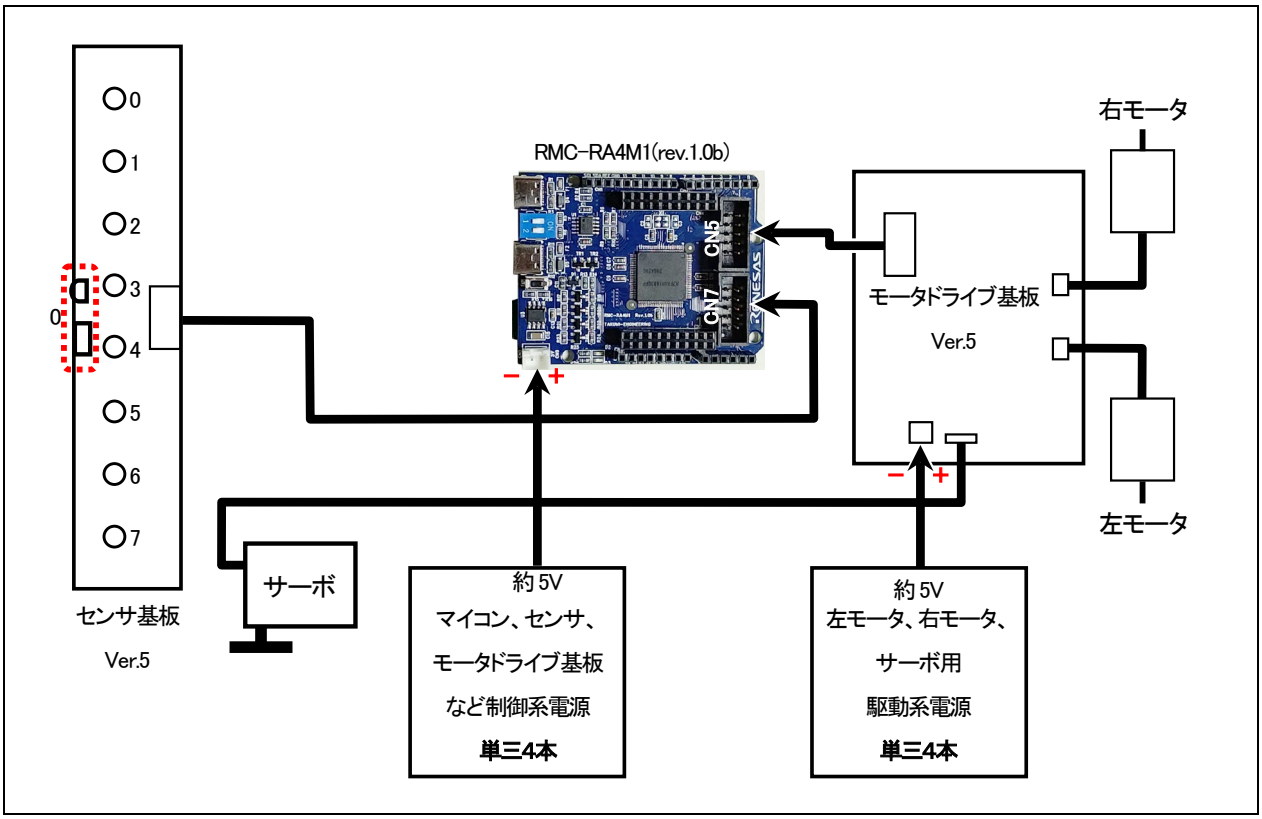
下記をメモしておきます。

サーボセンタ値		例）4469
右角度の限界値		例）40
左角度の限界値		例）-38

5.2. ベーシッククラスマイコンカー走行プログラム「kit23_ra4m1_msd」

5.2.1. 配線

ベーシッククラスマイコンカー走行プログラムです。
RMC-RA4M1 ボードの CN5 とモータドライブ基板 Ver.5、CN7 とセンサ基板 Ver.5 を接続します。

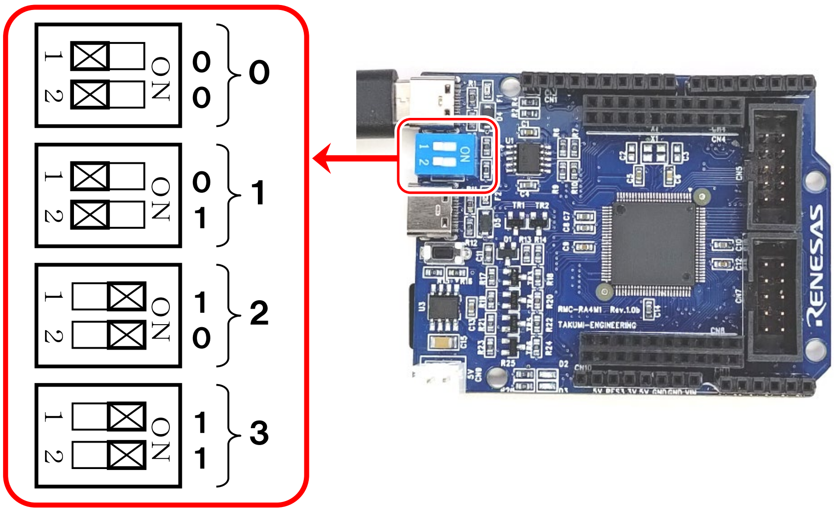


5.2.2. 調整・走行

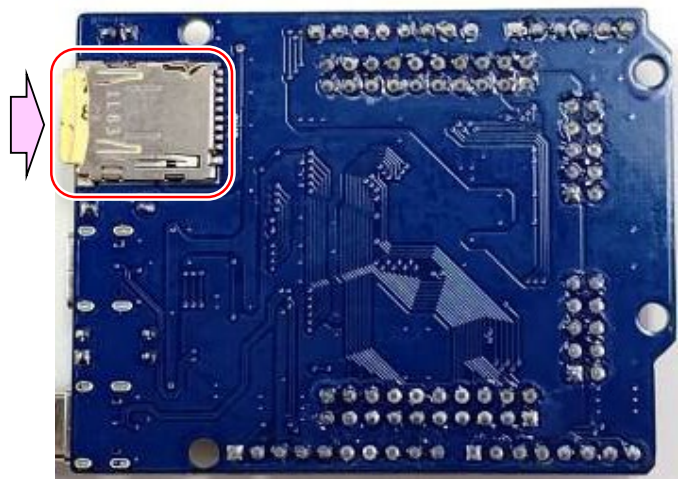
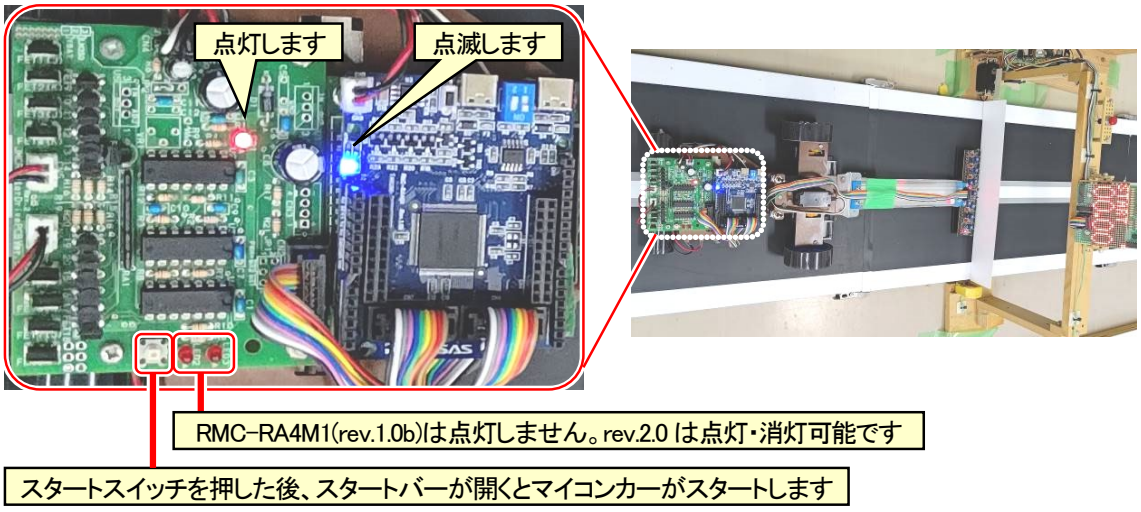
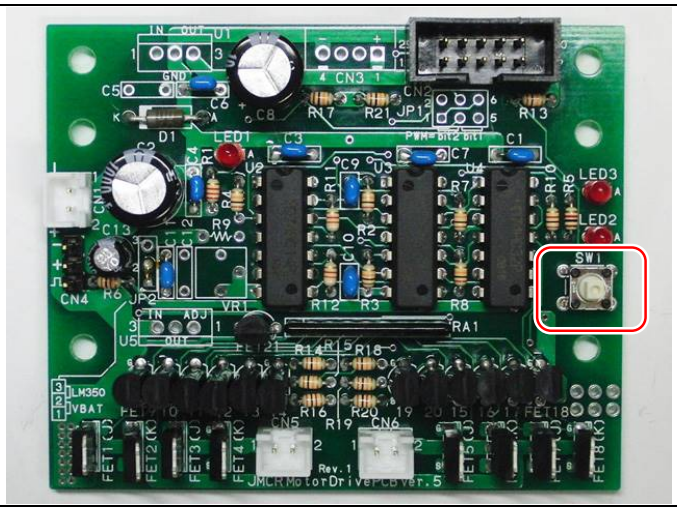
<table>
<tr>
<td>1</td>
<td>
<pre>
25   // モータ、サーボのPWM関係
26   #define PWM_CYCLE   47999   // 16ms/(1/HOCO*
27   #define SERVO_CENTER   4499   // 1.5ms/(1/HOCO*
28   #define HANDLE_STEP   26   // 1度あたりの値
～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～
562   case 23:
563   // クロスライン後のトレース、クランク検出
564   if(sensor_inp(MASK4_0) == 0xf0) {
565   // 左クランクと判断→左クランククリア処理へ
566   led_out(1); // led_out(0x1);
567   handle(-41);
568   motor(0, 50);
569   pattern = 31;
570   cnt1 = 0;
571   break;
572   }
573   if(sensor_inp(MASK0_4) == 0x0f) {
574   // 右クランクと判断→右クランククリア処理へ
575   led_out(1); // led_out(0x2);
576   handle(37);
577   motor(50, 0);
578   pattern = 41;
579   cnt1 = 0;
580   break;
581   }
</pre>
</td>
<td>「kit23_ra4m1_msd.ino」を開きます。

「 kit23_ra4m1_servo_tyosei.ino」で見つけた値3つを、自分のマイコンカーに合わせて調整します。

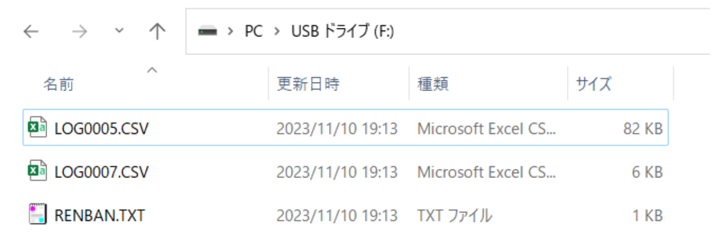
・27 行のサーボセンタ値
・567 行の左限界角度
・576 行の右限界角度

変更したら、RMC-RA4M1 ボードにプログラムを書き込んで下さい。</td>
</tr>
<tr>
<td>2</td>
<td colspan="2"> 

マイコンボードのディップスイッチ 2bit で、プログラムで motor 関数に設定している PWM を 0.7～1.0 倍にします。
具体的には、
　　実際に出力される PWM＝motor 関数で設定している値×（ディップスイッチの値＋７）÷１０
となります。
　例えば、motor(100, 80)とプログラムしていて、ディップスイッチの値が2の場合、
　　左モータに実際に出力される PWM＝100×(2＋7)÷10＝90%
　　右モータに実際に出力される PWM＝ 80×(2＋7)÷10＝72%
となります。</td>
</tr>
</table>

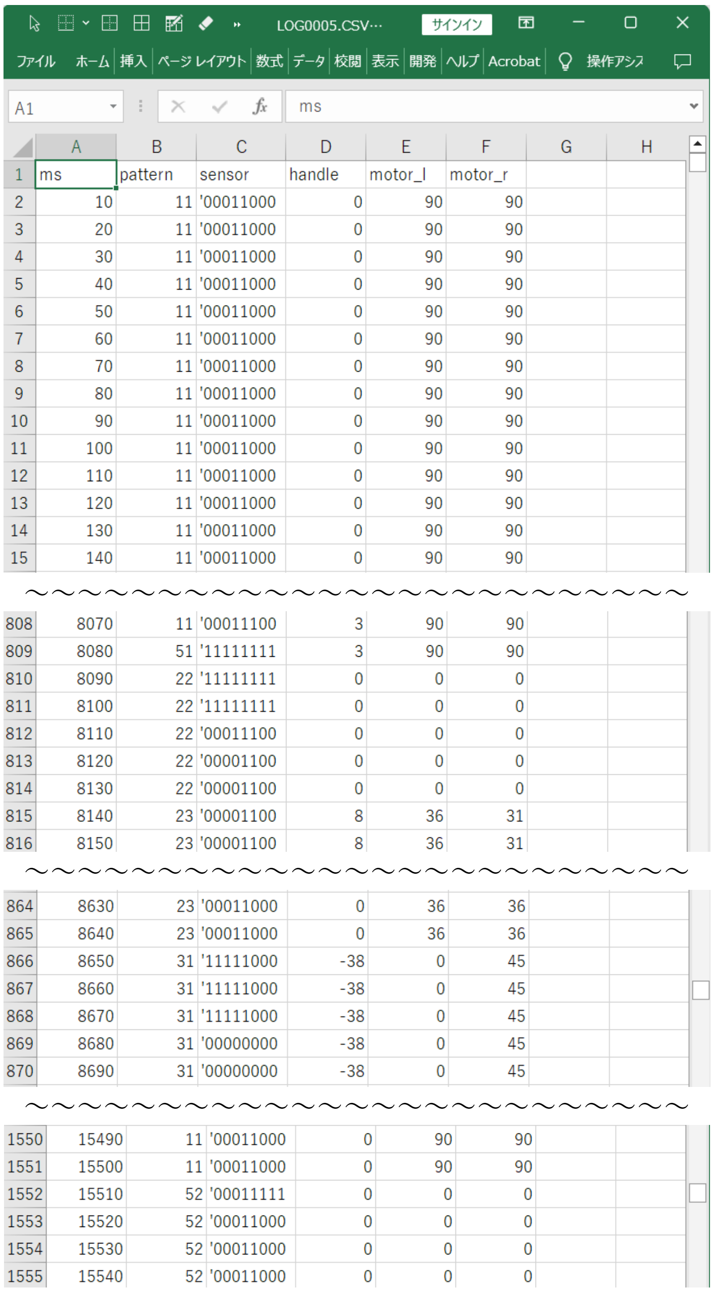
3		ログ保存用の microSD をセットします。 microSD は SDHC(32GB 以下)を使用してください。 また、あらかじめパソコンで FAT32 でフォーマットしておいてください。NTFS や exFAT は対応していません。
4	 マイコンカーをコースにセットします。 ・スタートバーがある場合は、スタートバーにセットします。スタートスイッチを押した後、スタートバーが開くとマイコンカーがスタートします。 ・スタートバーがない場合は、スタートスイッチを押すとすぐにスタートします。 **※モータドライブ基板の LED2、LED3 は、RMC-RA4M1(rev.1.0b)ボードで使用するときは点灯しません(入力端子に割当たっているため)。RMC-RA4M1 rev.2.0 基板は点灯します。**	
5		走行し終わったら、**電源を切る前に、必ずモータドライブ基板のスイッチを押してください**。microSD のログファイルクローズ処理を行います。LED がゆっくりした点滅になります。 スイッチ押下後、マイコンカーの電源スイッチを切ってください。 **ファイルクローズ処理を行わないで電源を切ると、ログファイルが保存されません。**

6



microSDドライブを開くと、「LOGxxxx.csv」というファイルができています。これが走行ログファイルです。「xxxx」は数字で連番になります。
「RENBAN.TXT」ファイルは連番の番号を設定します。連番を変更したいときは、このファイルを開き、連番の番号を設定してください。

7



	A	B	C	D	E	F
1	ms	pattern	sensor	handle	motor_l	motor_r
2	10	11	'00011000	0	90	90
3	20	11	'00011000	0	90	90
4	30	11	'00011000	0	90	90
5	40	11	'00011000	0	90	90
6	50	11	'00011000	0	90	90
7	60	11	'00011000	0	90	90
8	70	11	'00011000	0	90	90
9	80	11	'00011000	0	90	90
10	90	11	'00011000	0	90	90
11	100	11	'00011000	0	90	90
12	110	11	'00011000	0	90	90
13	120	11	'00011000	0	90	90
14	130	11	'00011000	0	90	90
15	140	11	'00011000	0	90	90
～～～						
808	8070	11	'00011100	3	90	90
809	8080	51	'11111111	3	90	90
810	8090	22	'11111111	0	0	0
811	8100	22	'11111111	0	0	0
812	8110	22	'00011100	0	0	0
813	8120	22	'00001100	0	0	0
814	8130	22	'00001100	0	0	0
815	8140	23	'00001100	8	36	31
816	8150	23	'00001100	8	36	31
～～～						
864	8630	23	'00011000	0	36	36
865	8640	23	'00011000	0	36	36
866	8650	31	'11111000	-38	0	45
867	8660	31	'11111000	-38	0	45
868	8670	31	'11111000	-38	0	45
869	8680	31	'00000000	-38	0	45
870	8690	31	'00000000	-38	0	45
～～～						
1550	15490	11	'00011000	0	90	90
1551	15500	11	'00011000	0	90	90
1552	15510	52	'00011111	0	0	0
1553	15520	52	'00011000	0	0	0
1554	15530	52	'00011000	0	0	0
1555	15540	52	'00011000	0	0	0

csv ファイルをダブルクリックすると、エクセルなどの表計算ソフトが立ち上がります。立ち上がらない場合は、表計算ソフトから csv ファイルを開いてください。
ログファイルが開かれ、走行ログを見ることができます。
A 列:時間[ms]
B 列:パータン番号
C 列:センサ値(2進数)
D 列:ハンドル角度
E 列:左モータ PWM 値
F 列:右モータ PWM 値
です。

【解説】

8080ms は、センサが「1111 1111」でクロスラインに入ったところですが、センサが右斜めに入ったらしく、パターンが右ハーフライン検出の 51 番になってしまいました。

8090ms は、パターン 52 でクロスラインチェックを入れているので、クロスラインと判断し直して、パターン 21 に移ります。22 番は 21 番から来るので、21 番はログに残っていないだけで、21 番には入っていると想像できます。

8650ms は、左クランクを見つけて、ハンドル-38、左モータ 0%、右モータ 45%で回しているところです。

15510ms は、右ハーフラインを見つけて 51 番、52 番に移ったところです。51 番はすぐに 52 番に移動するのでログには記録されていません。

5.2.3. サーボが左右逆の動きをした場合

1

```
983  //*****************************************************************
984  // サーボハンドル操作
985  // 引数    サーボ操作角度：-90～90
986  //          -90で左へ90度、0でまっすぐ、90で右へ90度回転
987  //*****************************************************************
988  void handle( int angle )
989  {
990    log_raw.ang = angle;
991    // サーボが左右逆に動く場合は、「-」を「+」に替えてください
992    GTIOC1B = SERVO_CENTER - angle * HANDLE_STEP;
993  }
```

「マイナス」を「プラス」に替えます

フタバなどのサーボを Basic マイコンカー走行プログラムで動かすと、サーボの動きが左右が逆になることがあります。このときは、handle 関数の「マイナス」を「プラス」にしてください。

```
GTIOC1B = SERVO_CENTER - angle * HANDLE_STEP;
                       ↓
GTIOC1B = SERVO_CENTER + angle * HANDLE_STEP;
```

6. 付録

6.1. 型指定子の範囲について

下記プログラムを実行して、各データ型が何バイトか調べました。変数や型のメモリサイズを調べるための演算子「sizeof」を使用しました。

```
void setup() {
  Serial.begin( 9600 );
  while ( !Serial ) {
    // 接続されるまで待つ
  }
  Serial.print( " char : " );
  Serial.println( sizeof( char ) );
  Serial.print( " short int : " );
  Serial.println( sizeof( short int) );
  Serial.print( " int  : " );
  Serial.println( sizeof( int ) );
  Serial.print( " long : " );
  Serial.println( sizeof( long ) );
  Serial.print( " long long : " );
  Serial.println( sizeof( long long ) );
  Serial.print( " float : " );
  Serial.println( sizeof( float ) );
  Serial.print( " double : " );
  Serial.println( sizeof( double ) );
  Serial.print( " long double : " );
  Serial.println( sizeof( long double ) );
}

void loop() {
  // 何もしない
}
```

実行した結果、各データ型とサイズ、範囲を下表に示します。

	Arduino UNO R4 RMC-RA4M1(rev.1.0b)	Arduino UNO R3	R8C/38A、35A
char	1 バイト -128 ～ 127	1 バイト -128 ～ 127	1 バイト -128 ～ 127
short int	2 バイト -32,768 ～ 32,767	2 バイト -32,768 ～ 32,767	2 バイト -32,768 ～ 32,767
int **※下記参照**	**4 バイト** **-2,147,483,648 ～ 2,147,483,647**	**2 バイト** **-32,768 ～ 32,767**	**2 バイト** **-32,768 ～ 32,767**
long	4 バイト -2,147,483,648 ～ 2,147,483,647	4 バイト -2,147,483,648 ～ 2,147,483,647	4 バイト -2,147,483,648 ～ 2,147,483,647
long long	8 バイト -9,223,372,036,854,775,808 ～ 9,223,372,036,854,775,807	8 バイト -9,223,372,036,854,775,808 ～ 9,223,372,036,854,775,807	8 バイト -9,223,372,036,854,775,808 ～ 9,223,372,036,854,775,807
float	4 バイト 3.4E-38 ～ 3.4E+38	4 バイト 3.4E-38 ～ 3.4E+38	4 バイト 3.4E-38 ～ 3.4E+38
double および long double	8 バイト 1.7E-308 ～ 1.7E+308	4 バイト 3.4E-38 ～ 3.4E+38	8 バイト 1.7E-308 ～ 1.7E+308

※データ型のについて

C言語のデータ型の大きさは処理系(メーカーやマイコンの種類)によって異なります。RMC-RA4M1(rev.1.0b)の int のサイズと、R8C/38A の int のサイズは異なります。

マイコンを変更しても、扱える整数の型の大きさを変えたくないときは、サイズを決めた整数型を使用します。この型はC言語では「stdint.h」をインクルードすれば使用可能です。ArduinoIDE はインクルードしなくても使用可能です。符号有りの型は「int○_t」(○＝8,16,32 など)、符号無しの型は「uint○_t」(○＝8,16,32 など)となります。

6.2. float の実行時間について

下記プログラムを実行して、実行速度を調べました。調べ方は、測定したいプログラムを実行する直前で端子を"1"(5V)にして、実行後に"0"(0V)にします。"1"の幅をオシロスコープで観測し時間を計ります。

❶int、long の計算時間測定
※int は、下記 long を int に変更
※R8C は、ほぼ同等のプログラムに変更

```
void setup() {
  long data[182];

  pinMode( 2 , OUTPUT );

  Serial.begin( 9600 );
  while( !Serial ) {
  }

  digitalWrite( 2 , 1 );  // スタート
  for( int i=1; i<=181; i++ ) {
      data[i] = (long)i * (long)i; // 最大181*181=32761
  }
  digitalWrite( 2 , 0 );  // 終了

  // 正しく計算されているかシリアルモニタで確認
  for( int i=1; i<=181; i++ ) {
    Serial.print( i );
    Serial.print( " : " );
    Serial.println( data[i] );
  }
}

void loop() {

}
```

❷float、double の計算時間測定【乗除算】
※double は、下記 float を double に変更
※R8C は、ほぼ同等のプログラムに変更

```
void setup() {
  float data[361];

  pinMode( 2 , OUTPUT );

  Serial.begin( 9600 );
  while( !Serial ) {
  }

  digitalWrite( 2 , 1 );  // スタート
  for( int i=0; i<=360; i++ ) {
    data[i] = (float)i * PI;
    data[i] = data[i]  / 180.0;
  }
  digitalWrite( 2 , 0 );  // 終了

  // 正しく計算されているかシリアルモニタで確認
  for( int i=0; i<=360; i++ ) {
    Serial.print( i );
    Serial.print( " : " );
    Serial.println( data[i], 10 );
  }
}

void loop() {

}
```

❸float、double の計算時間測定【sin】
※double は、下記 float を double に変更
※R8C は、ほぼ同等のプログラムに変更

```
void setup() {
  float data[361];

  pinMode( 2 , OUTPUT );

  Serial.begin( 9600 );
  while( !Serial ) {
  }

  digitalWrite( 2 , 1 );  // スタート
  for( int i=0; i<=360; i++ ) {
    data[i] = sin( (float)i * PI / 180 );
  }
  digitalWrite( 2 , 0 );  // 終了


  // 正しく計算されているかシリアルモニタで確認
  for( int i=0; i<=360; i++ ) {
    Serial.print( i );
    Serial.print( " : " );
    Serial.println( data[i], 10 );
  }
}

void loop() {

}
```

実行したときの処理時間を下表に示します。

		Arduino UNO R4、 RMC-RA4M1(rev.1.0b)	Arduino UNO R3	R8C/38A、 R8C/35A
❶	int 型の かけ算	0.027ms 1 倍とすると	0.2ms 7.4 倍	0.45ms 16.7 倍
	long 型の かけ算	0.027ms 1 倍とすると	1.1ms 40.7 倍	1.4ms 51.9 倍
❷	float 型で 乗除算	1.1ms 1 倍とすると	16ms 14.5 倍	180ms 163 倍
	double 型で 乗除算	6.4ms 1 倍とすると	16ms 2.5 倍	180ms 28.1 倍
❸	float 型で sin 計算	26ms※1 1 倍とすると	55ms 2.1 倍	1750ms 67.3 倍
	下記※の方法で float 型で sin 計算	9ms※2 1 倍とすると	55ms 6.1 倍	1750ms 194 倍
	double 型で sin 計算	26ms 1 倍とすると	55ms 2.1 倍	1750ms 67.3 倍

※RMC-RA4M1 の sin 計算は、予想以上に時間がかかりました(※1 部分)。これは sin 関数を使用する math ライブラリが FPU(浮動小数点演算装置)を使用して計算していないものと考えられます。sin 関数を自作しているサイト(https://hiroyukichishiro.com/sin-cos-tan-functions-in-c-language/)の関数を参考に実行すると、同等のプログラムで、実測で約 9ms で実行しました(※2 部分)。

7. 参考文献

・Renesas RA4M1 グループ ユーザーズマニュアル ハードウェア編 Rev.1.00 2020.03
https://www.renesas.com/jp/ja/products/microcontrollers-microprocessors/ra-cortex-m-mcus/ra4m1-32-bit-microcontrollers-48mhz-arm-cortex-m4-and-lcd-controller-and-cap-touch-hmi

・スイッチサイエンス Arduino Uno R4 Minima のホームページ
https://www.switch-science.com/products/9000

・I2C 液晶の使い方
LCD の使い方(AE-AQM0802)のホームページ
https://nobita-rx7.hatenablog.com/entry/28696592

・グラフィック液晶の使い方
しなぷすのハード製作記のホームページ
https://synapse.kyoto/lib/MGLCD/page001.html
https://synapse.kyoto/hard/MGLCD_AQM1248A/page001.html
https://synapse.kyoto/hard/MGLCD_AQM1248A/page016.html