RMC-RA4M1 (rev.2.0) 基板
実習マニュアル

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。

第 1.10 版
2024.08.11
ジャパンマイコンカーラリー実行委員会

目　次

1. 概要 1

2. 仕様 2

2.1. シルク図 2
2.2. 寸法 2
2.3. 外観 3
2.4. 回路図 4

3. 開発環境を整える 9

3.1. Windows の設定 9
3.2. 開発環境のダウンロード 9
3.3. Arduino IDE に「RMC-RA4M1(rev.2.0)」を追加する 9
3.4. サンプルプログラムをコピーする 10
3.5. ファームウェアを書き込む 11
3.6. Arduino IDE でプログラムを書き込む 14

4. 演習 16

4.1. 演習 1　マイコンボード上の LED の点滅「led_out.ino」 16
4.1.1. 配線 16
4.1.2. プログラム 16
4.1.3. プログラムの解説 17
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」 19
4.2.1. 配線 19
4.2.2. プログラム 19
4.2.3. プログラムの解説 20
4.3. 演習 3　シリアル通信（CN3 使用(ArduinoIDE 書き込み用)）「serial_cn3.ino」 21
4.3.1. 配線 21
4.3.2. シリアルモニタを表示させる 21
4.3.3. プログラム 22
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」 23
4.4.1. 配線 23
4.4.2. プログラム 24
4.5. 演習 5　1ms ごとの割り込み処理「interrupt_1ms.ino」 25
4.5.1. 配線 25
4.5.2. プログラム 26
4.5.3. プログラムの解説 27
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」 28
4.6.1. 配線 28
4.6.2. プログラム 28
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」 30
4.7.1. 配線 30
4.7.2. プログラム 30
4.7.3. プログラムの解説 31
4.8. 演習 8　analogWrite を使用した PWM 出力「analogwrite.ino」 32
4.8.1. 配線 32

4.8.2. プログラム 32
4.8.3. プログラムの解説 33
4.9. 演習 9　microSD へファイルの読み書き「microsd.ino」 34
4.9.1. 配線 34
4.9.2. SD ライブラリの追加 35
4.9.3. プログラム 35
4.9.4. プログラムの解説 37
4.9.5. Arduino UNO R4 と互換端子(D10～D13 端子)に接続して microSD を使用する 38
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」 39
4.10.1. 配線 39
4.10.2. プログラム 39
4.10.3. プログラムの解説 42
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」 45
4.11.1. 配線 45
4.11.2. プログラム 46
4.11.3. プログラムの解説 47
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」 48
4.12.1. 配線 48
4.12.2. プログラム 49
4.12.3. プログラムの解説 49
4.13. 演習 13　A/D 変換「analogread.ino」 51
4.13.1. 配線 51
4.13.2. プログラム 52
4.13.3. プログラムの解説 53
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」 54
4.14.1. 配線 54
4.14.2. プログラム 54
4.14.3. プログラムの解説 56
4.15. 演習 15　I2C 液晶　AQM0802 の表示「aqm0802.ino」 57
4.15.1. 配線 57
4.15.2. プログラム 58
4.16. 演習 16　I2C 液晶　AQM0802 の表示 Wire1 を使用「aqm0802_wire1.ino」 59
4.16.1. 配線 59
4.16.2. プログラム 59
4.16.3. プログラムの解説 59
4.17. 演習 17　I2C 液晶　AQM0802 の表示 ソフトウェア I2C を使用「aqm0802_soft_i2c.ino」 60
4.17.1. 配線 60
4.17.2. プログラム 60
4.17.3. プログラムの解説 60
4.18. 演習 18　グラフィック液晶　AQM1248 の文字表示「aqm1248.ino」 61
4.18.1. 配線 61
4.18.2. プログラム 62
4.18.3. プログラムの解説 62
4.18.4. マイコン内蔵の SPI 機能を使用せずに、ソフトウェア SPI で動作させる 63
4.19. 演習 19　グラフィック液晶　AQM1248 のグラフィック表示「aqm1248graphic.ino」 64
4.19.1. 配線 64
4.19.2. プログラム 64
4.19.3. プログラムの解説 65
4.20. 演習 20　EEP-ROM(データフラッシュメモリ)「eeprom.ino」 66
4.20.1. 配線 66

4.20.2. プログラム 66
4.20.3. プログラムの解説 67

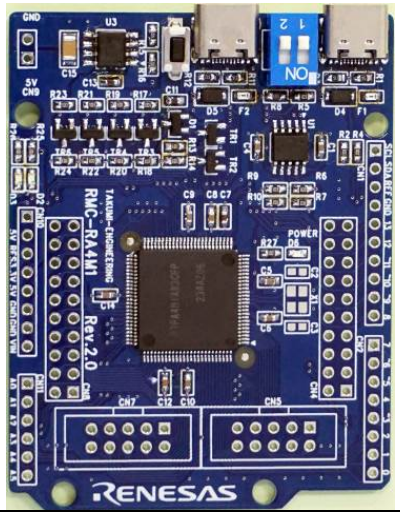
5. 付録 68

5.1. 型指定子の範囲について 68
5.2. int、long、float の実行時間について 69

6. 参考文献 70

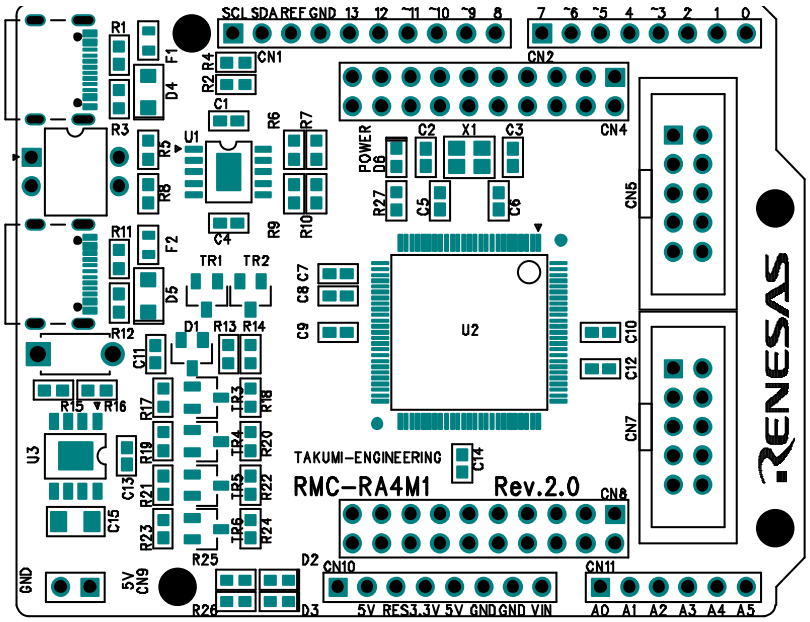
1. 概要

RMC-RA4M1(rev.2.0)ボードと Arduino Uno R4 の仕様を、下記に示します。

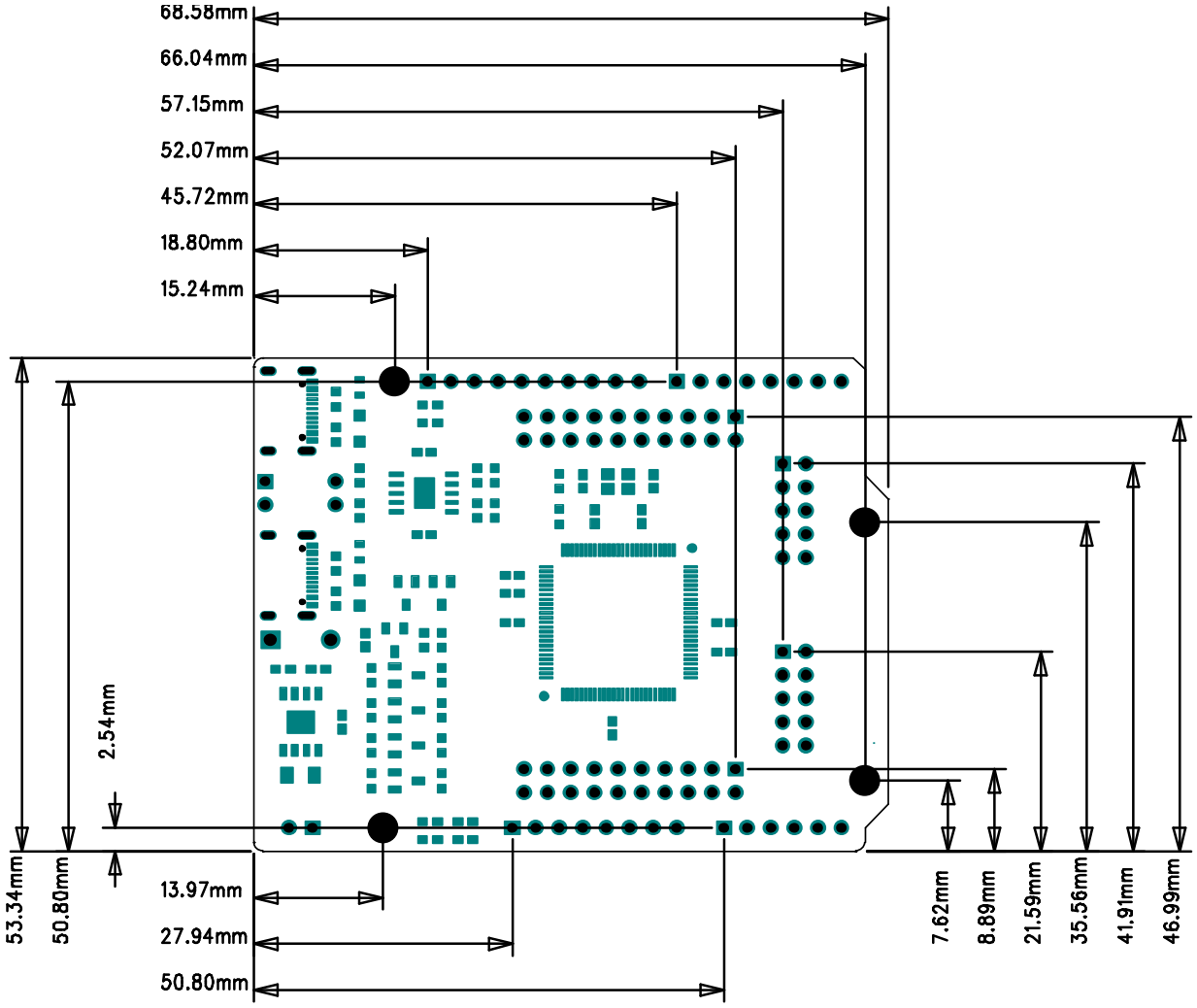
基板名	RMC-RA4M1 rev.2.0	Arduino Uno R4 MINIMA
写真		
マイコン	ルネサスエレクトロニクス製 RA4M1 100 ピン	ルネサスエレクトロニクス製 RA4M1 64 ピン
USB	TypeC × 2 個 CN6(中のコネクタ)は Renesas Flash Programmer V3 書き込み用 CN3(端のコネクタ)は、Arduino IDE 用	TypeC × 1個
入力電圧	CN9 または 5V 端子に、5V±10%入力 ※VIN 端子は、未接続	入力電圧(VIN):6～24 V または、5V 端子に 5V±10%入力
クロック	48MHz	48MHz
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～D13 の 14 本、A0～A5 の 6 本、 CN5 の 8 本、CN7 の 8 本、 CN4 の 15 本、CN8 の 18 本 **●デジタル入力端子:3 端子** ※CN5 の(P214)、(P215)、(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 PWM 出力端子:16 端子**	**I/O 端子の中で 14 端子 (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	3 個(電源用、D13、D23 に接続)	3 個(電源用、D0、D1 に接続)
ディップ スイッチ	2bit(D25、D26 に接続)	なし
購入時の 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

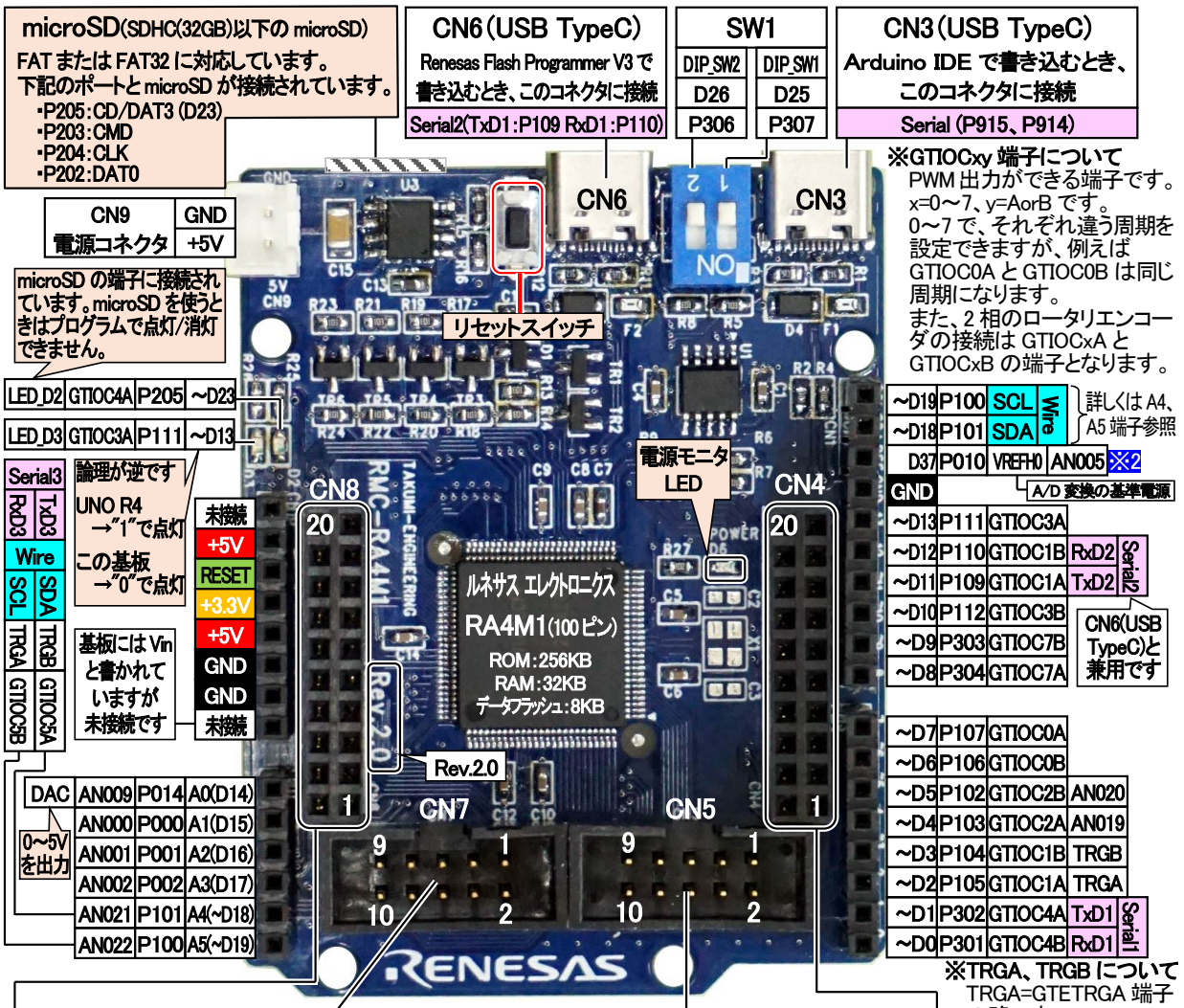
2.1. シルク図



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
P004	P006	P008	P011	
AN004	AN012	AN014	AN006	
GND	D41	D39	D37(※2)	D35
	P005	P007	P010	P012
	AN011	AN013	AN005	AN007

CN5　10 ピンコネクタ(モータドライブ基板)

D34	D32	D30	D28	+5V
P003	P401	P403	P405	
AN003(プッシュ SW)	TGRA/6B(左モ PWM)	3A(右モ PWM)	1A(LED3)	
GND	D33	D31	D29	D27
	P400	P404	P406	P402
	6A(左モータ方向)	3B(右モ方向)	1B(サーボ)	(LED2)

※2…D37 は 2 端子あります　※3…D54 は CN3(USB)の電圧端子です。使用するときは動作するか検証してください

CN8　20 ピンコネクタ

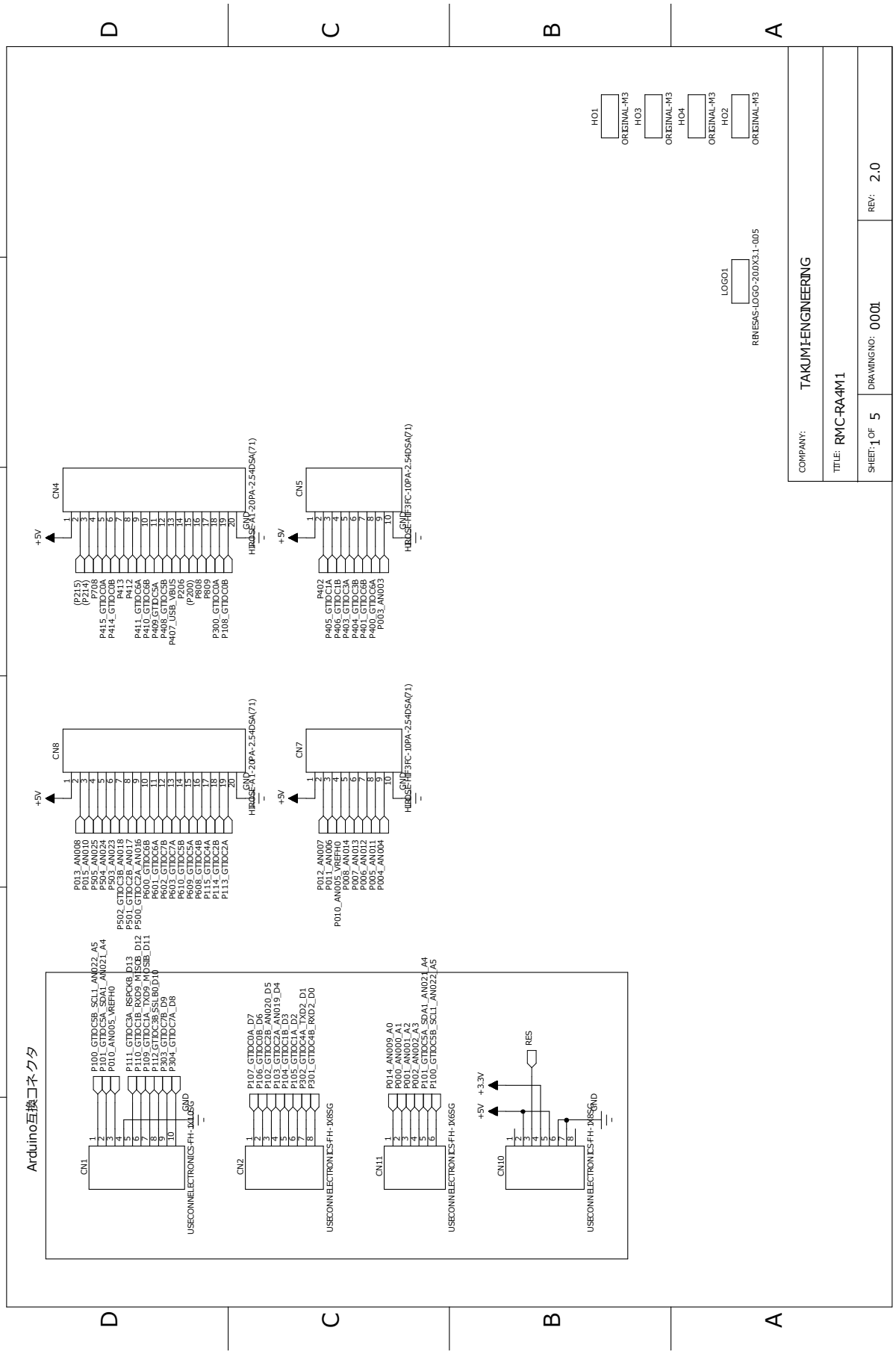
	(20 ピン)	GND			
GTIOC2B	P114	D77	D78	P113	GTIOC2A
GTIOC4B	P608	D75	D76	P115	GTIOC4A
GTIOC5B	P610	D73	D74	P609	GTIOC5A
GTIOC7B	P602	D71	D72	P603	GTIOC7A
GTIOC6B	P600	D69	D70	P601	GTIOC6A
AN017/2B	P501	D67	D68	P500	AN016/2A
AN023	P503	D65	D66	P502	AN018/3B
AN025	P505	D63	D64	P504	AN024
AN008	P013	D61	D62	P015	AN010
			+5V	(1 ピン)	

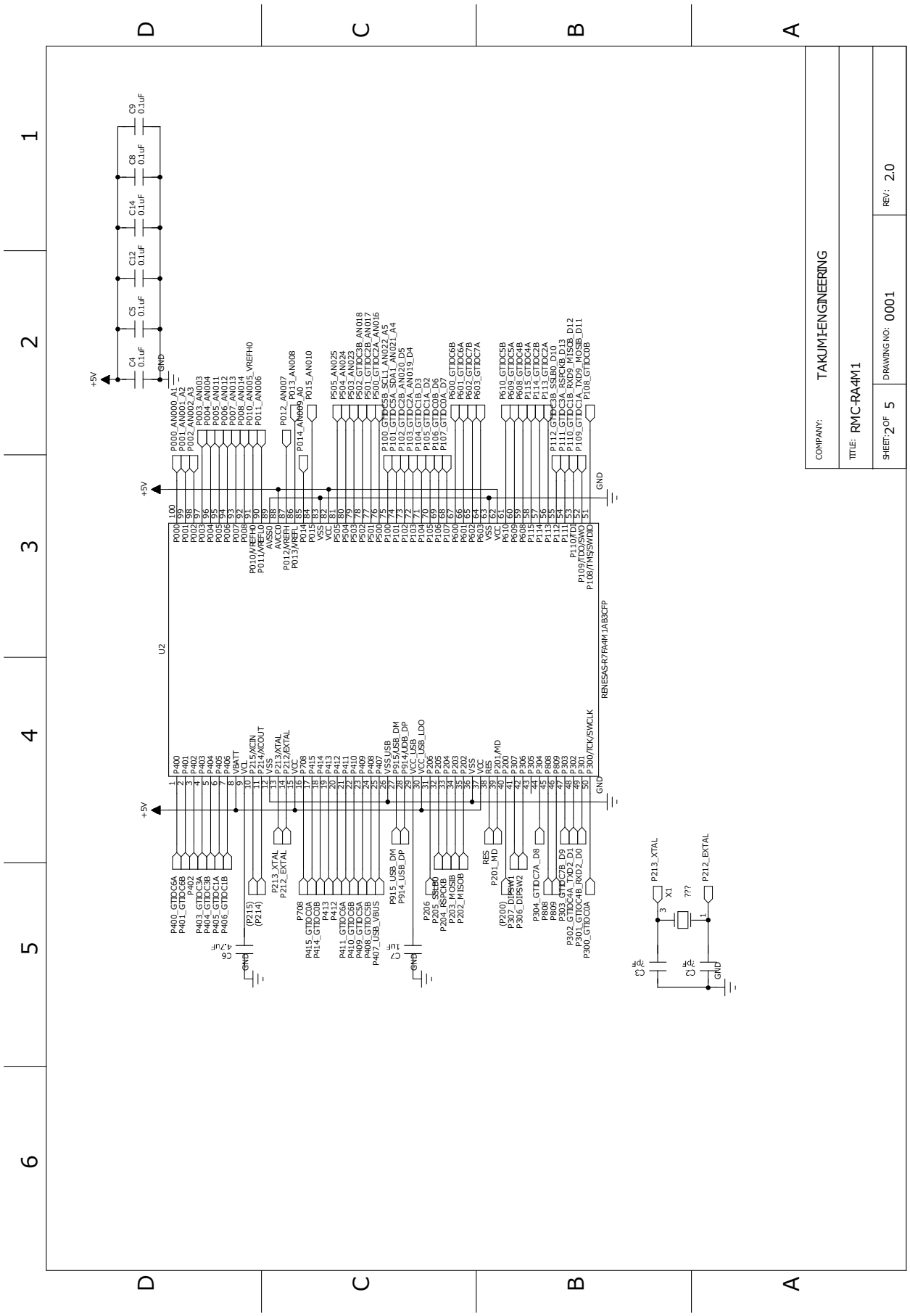
CN4　20 ピンコネクタ　※(P000)は入力専用端子

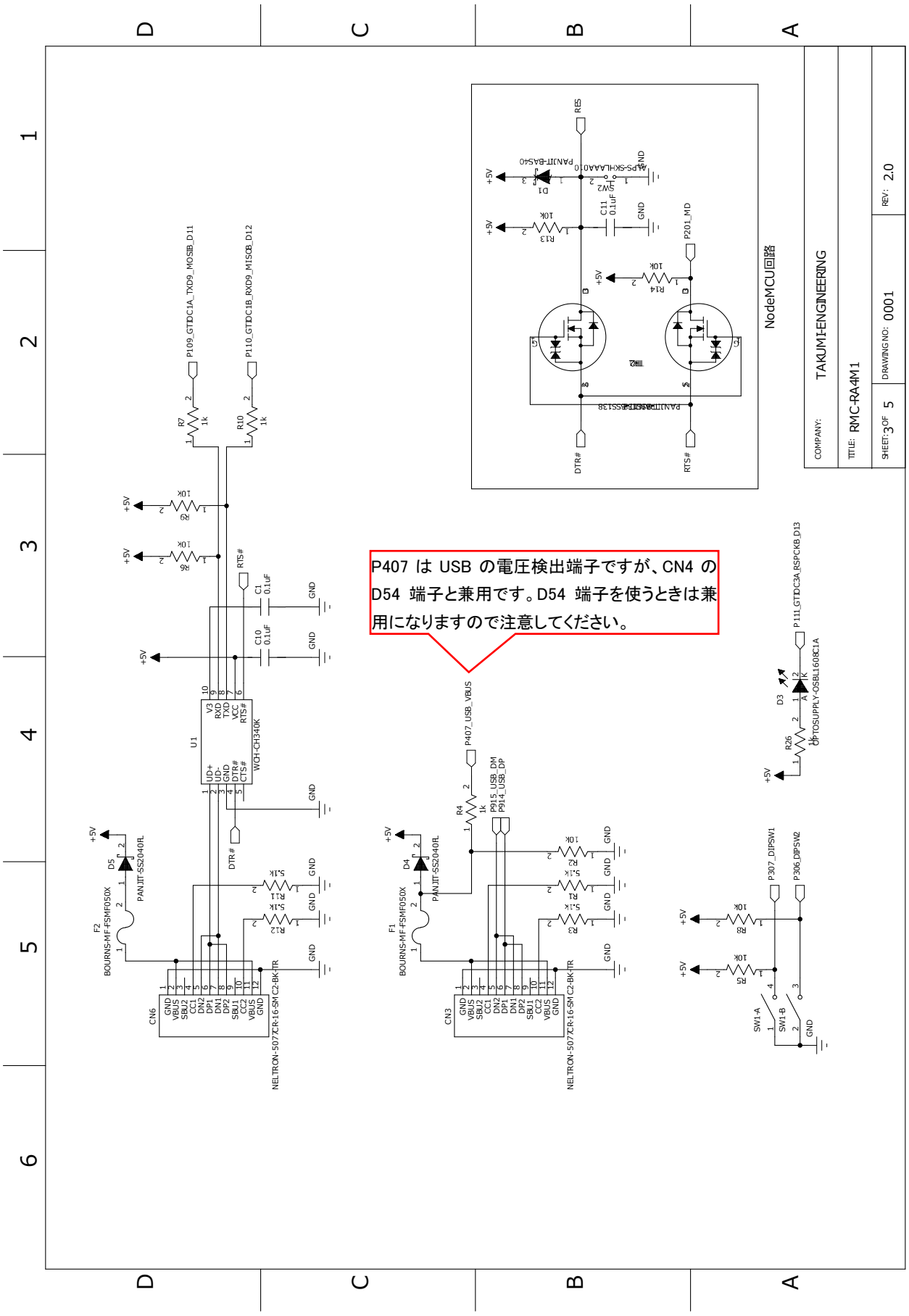
	(20 ピン)	GND			
GTIOC0A	P300	D59	D60	P108	GTIOC0B
–	P808	D57	D58	P809	–
–	P206	D55	D56	(P200)	–
Wire1 SCL1	GTIOC5B P408	D53	D54	P407(※3)	– SDA1 Wire1
GTIOC6B	P410	D51	D52	P409	GTIOC5A
–	P412	D49	D50	P411	GTIOC6A
GTIOC0B	P414	D47	D48	P413	–
–	P708	D45	D46	P415	GTIOC0A
–	(P215)	D43	D44	(P214)	–
			+5V	(1 ピン)	

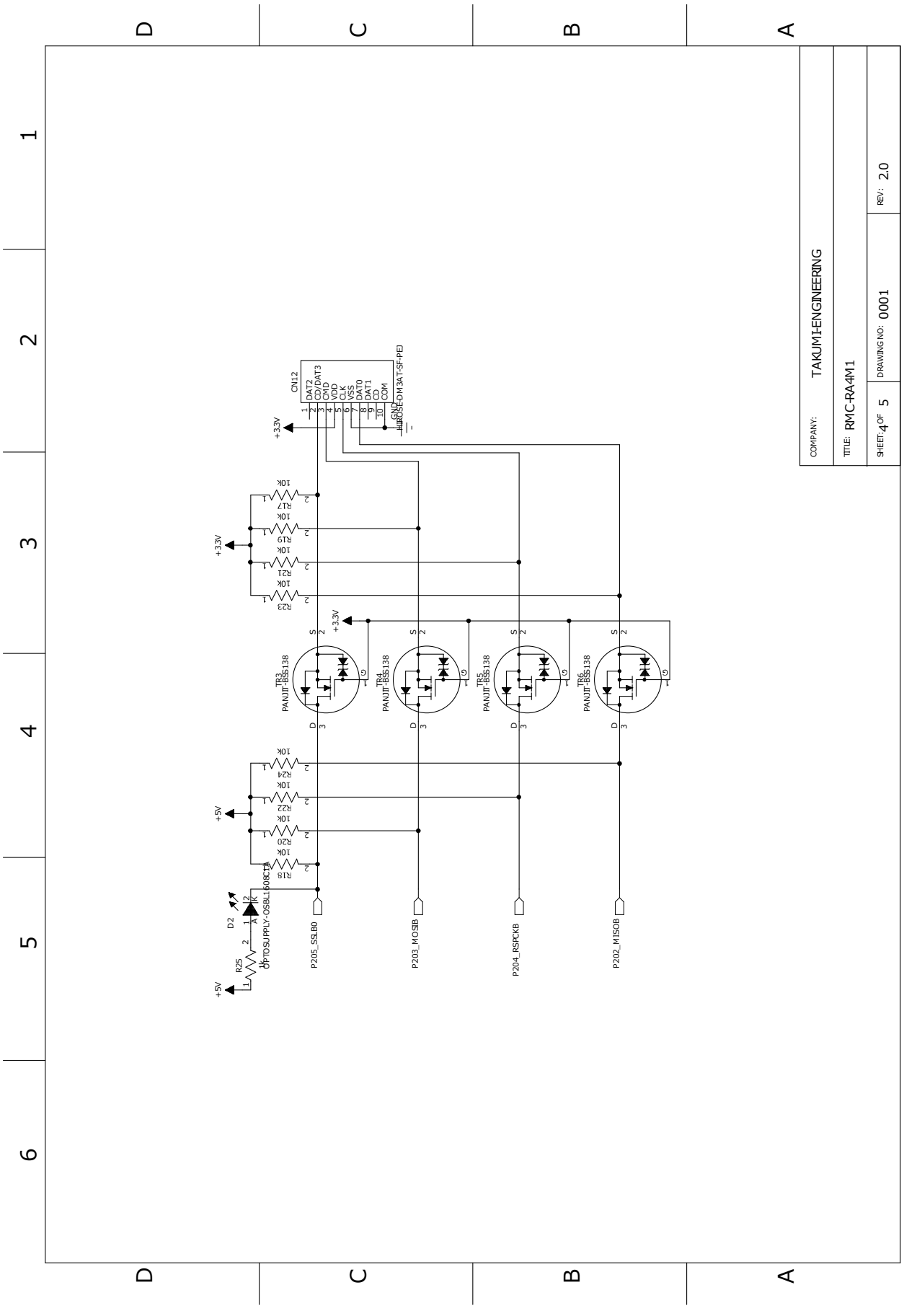
RMC-RAM1 Rev.2.0 基板のピン配置です。Rev.1.0b とは全く異なります。

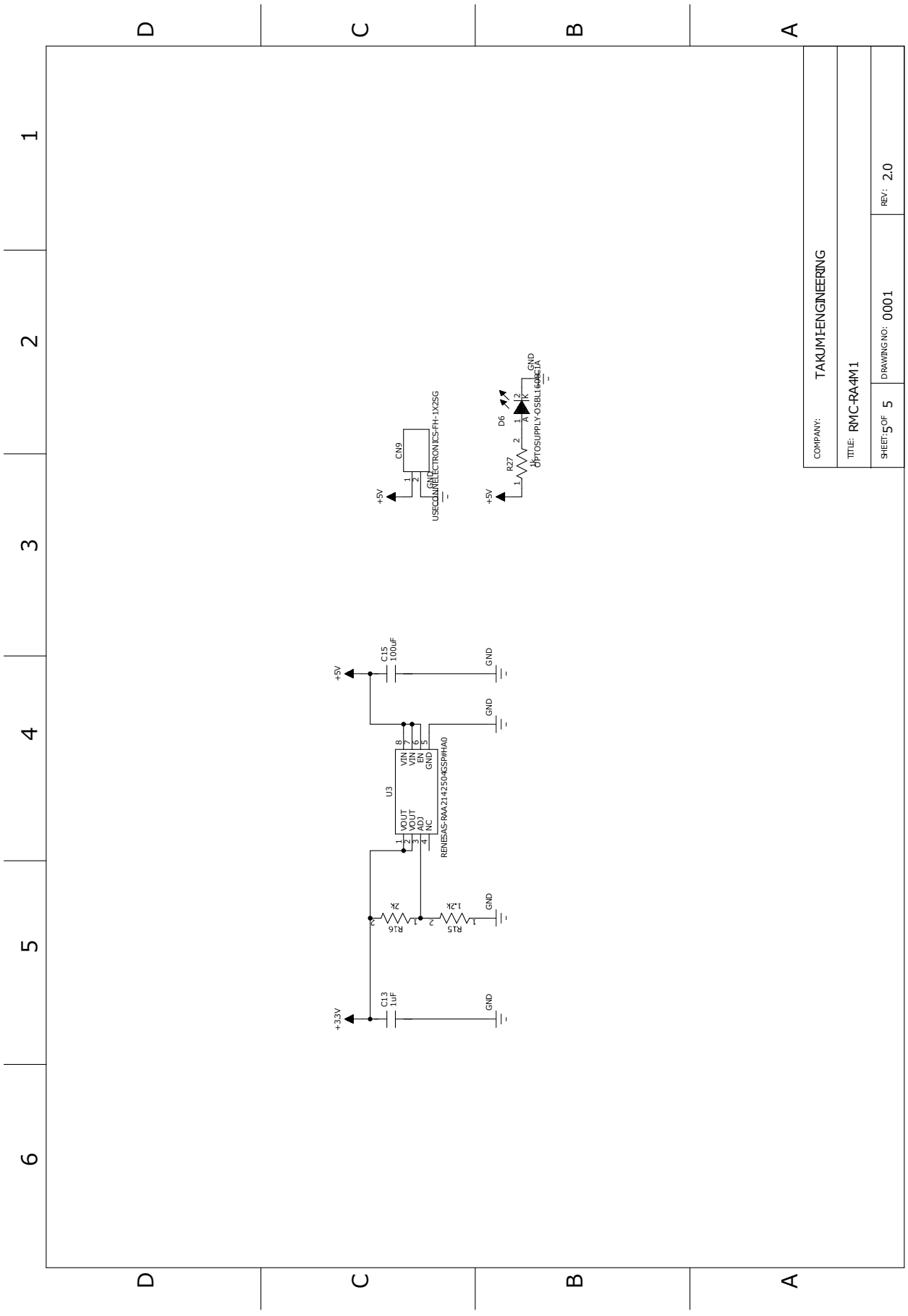
2.4. 回路図


Arduino互換コネクタ
CN1
CN2
CN4
CN5
CN7
CN8
CN10
CN11
USECONNELECTRONICS-FH-1X10SG
USECONNELECTRONICS-FH-1X8SG
USECONNELECTRONICS-FH-1X6SG
USECONNELECTRONICS-FH-1X8SG
HIROSE-AI-20PA-2.54DSA(71)
HIROSE-HIF3FC-10PA-2.54DSA(71)
+5V
+3.3V
GND
RES
HO1
HO2
HO3
HO4
ORIGINAL-M3
LOGO1
RENESAS-LOGO-20.0X3.1-0.05
COMPANY: TAKUMI-ENGINEERING
TITLE: RMC-RA4M1
SHEET: 1 OF 5
DRAWING NO: 0001
REV: 2.0


COMPANY: TAKUMI-ENGINEERING
TITLE: RMC-RA4M1
SHEET: 2 OF 5
DRAWING NO: 0001
REV: 2.0
U2
RENESAS:R7FA4M1AB3CFP
+5V
GND
C4 0.1uF
C5 0.1uF
C12 0.1uF
C14 0.1uF
C8 0.1uF
C9 0.1uF
X1
P213_XTAL
P212_EXTAL

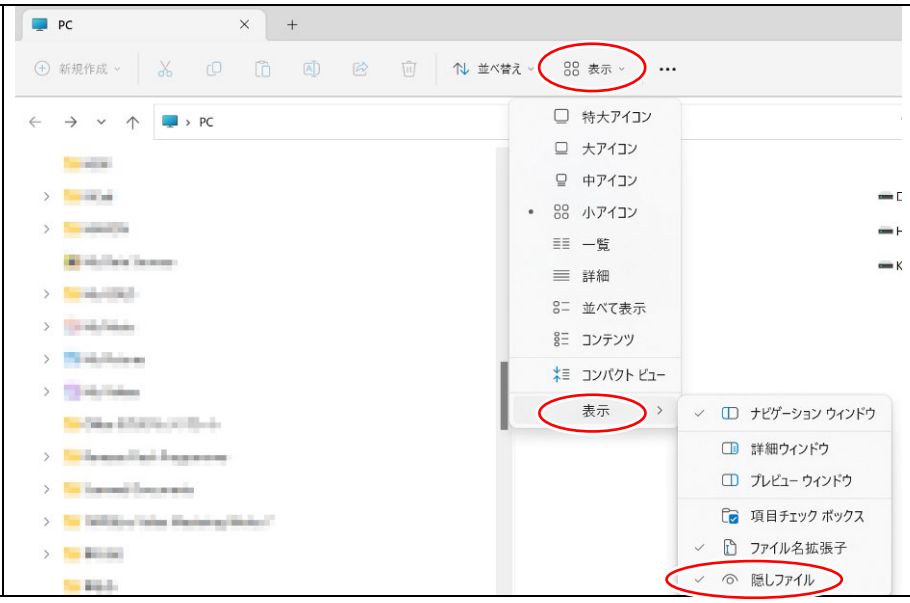

P407 は USB の電圧検出端子ですが、CN4 の D54 端子と兼用です。D54 端子を使うときは兼用になりますので注意してください。
NodeMCU回路
COMPANY: TAKUMI-ENGINEERING
TITLE: RMC-R4M1
SHEET: 5 OF 5
DRAWING NO: 0001
REV: 2.0


COMPANY: TAKUMI-ENGINEERING
TITLE: RMC-R4M1
SHEET: 4 OF 5
DRAWING NO: 0001
REV: 2.0
CN12
DAT2
CD/DAT3
CMD
VDD
CLK
VSS
DAT0
DAT1
CD
COM
GND
+3.3V
+5V
R25
D2
TR3
TR4
TR5
TR6
PANJIT-BSS138
P205_SSLB0
P203_MOSIB
P204_RSPCKB
P202_MISOB

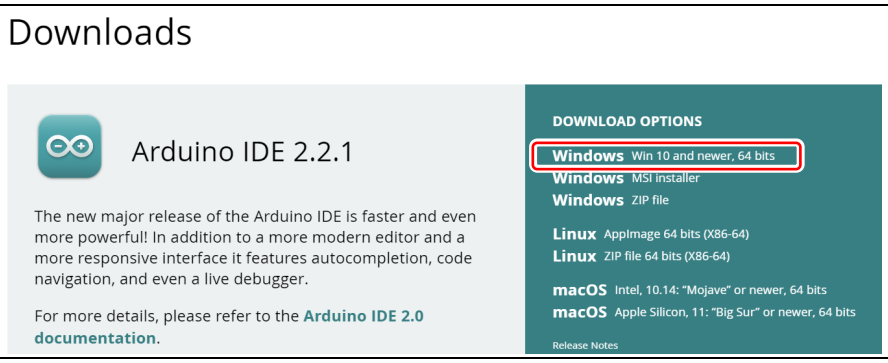

COMPANY: TAKUMI-ENGINEERING
TITLE: RMC-RA4M1
SHEET: 5 OF 5
DRAWING NO: 0001
REV: 2.0
CN9
USECONN/ELECTRONICS-FH-1X2SG
D6
OPTOSUPPLY-OSBL1608C1A
R27
U3
RENESAS-RAA2142504GSP#HA0
C15 100uF
C13 1uF
R15 1.2K
R16 2K
+5V
+3.3V
GND

3. 開発環境を整える

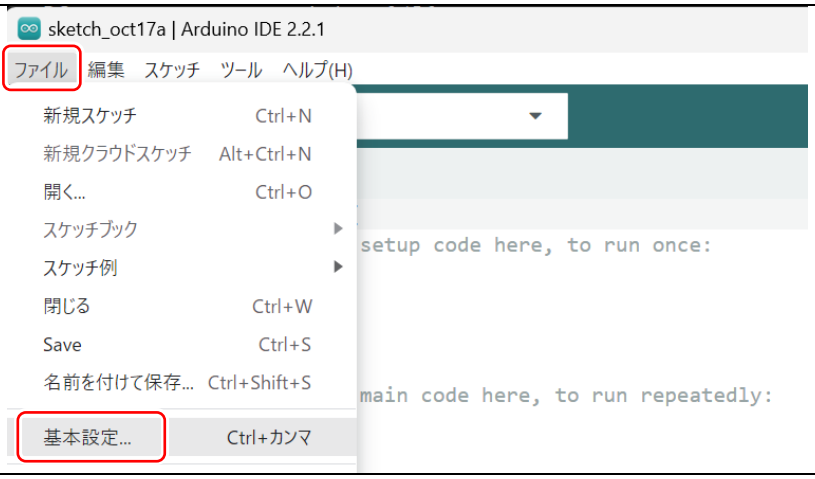
3.1. Windows の設定

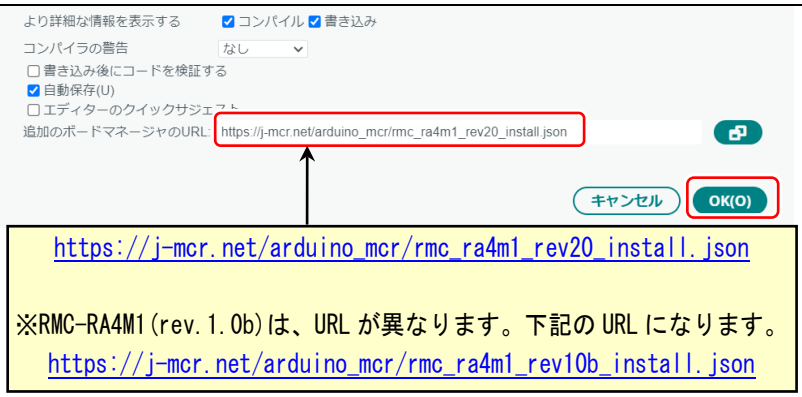
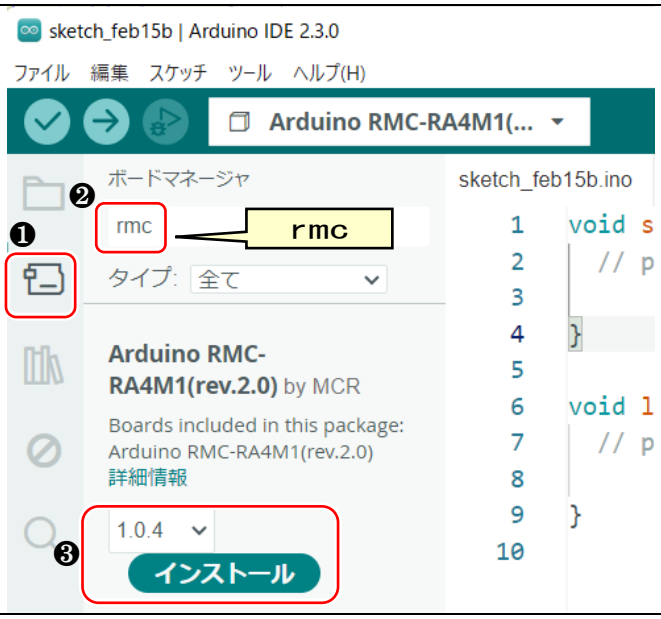
1		これから、隠しフォルダ・ファイルを操作するので、隠しフォルダ・ファイルが見られるように設定しておきます。 デスクトップの PC アイコンをダブルクリックするか、Win＋E キーを押してエクスプローラーウィンドウを表示します。 「表示」タブ→「表示」→「隠しファイル」にチェックを入れます。

3.2. 開発環境のダウンロード

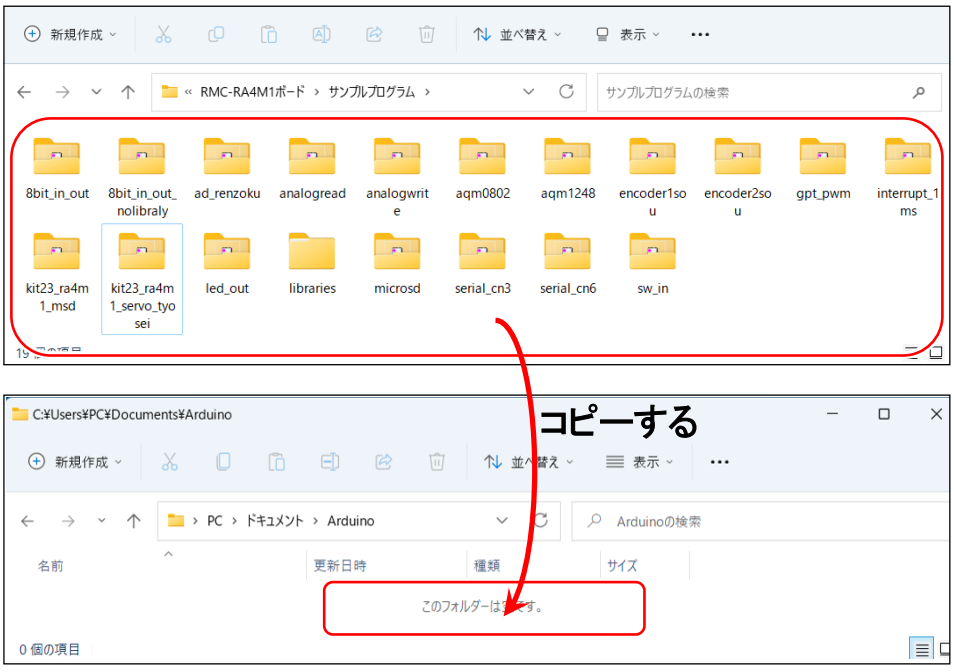
1		ブラウザで「Arduino IDE」と検索し、Arduino IDE ダウンロードサイト（https://www.arduino.cc/en/software）を開きます。Arduino IDE 2.x.x をダウンロード、インストールします。

3.3. Arduino IDE に「RMC-RA4M1(rev.2.0)」を追加する

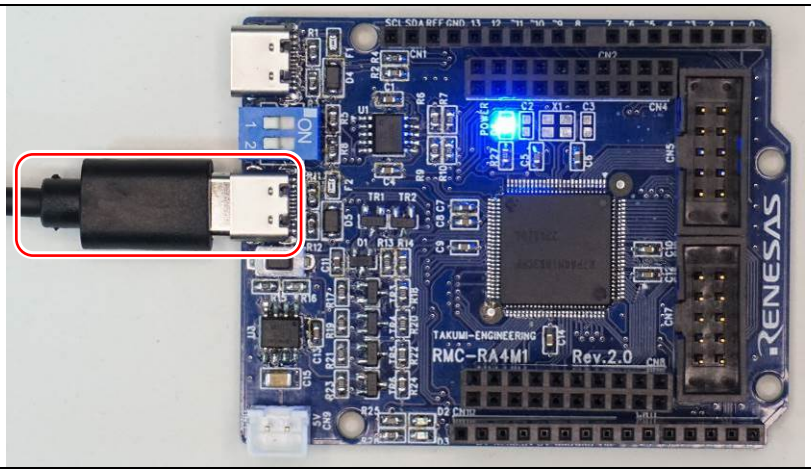
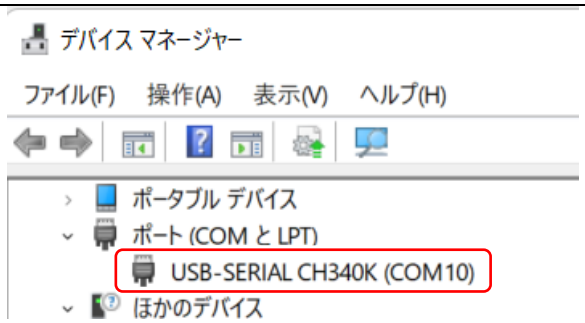
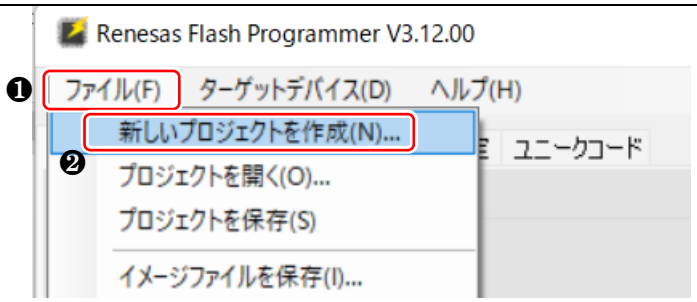
1		Arduino IDE を立ち上げます。 「ファイル→基本設定」を選択します。

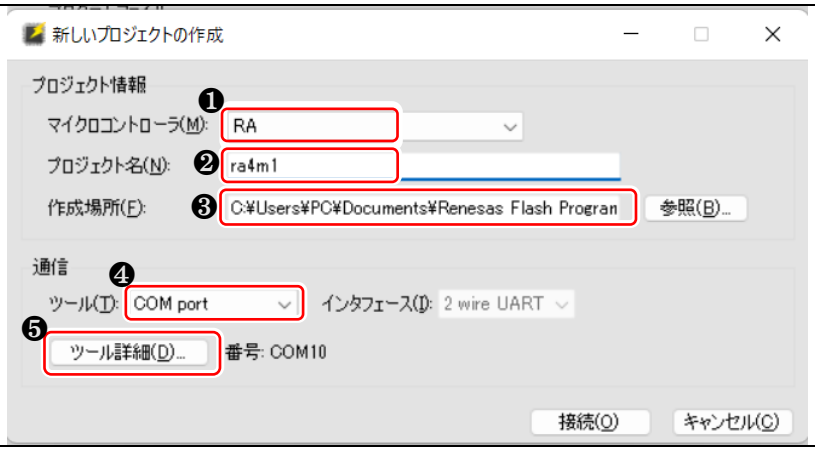
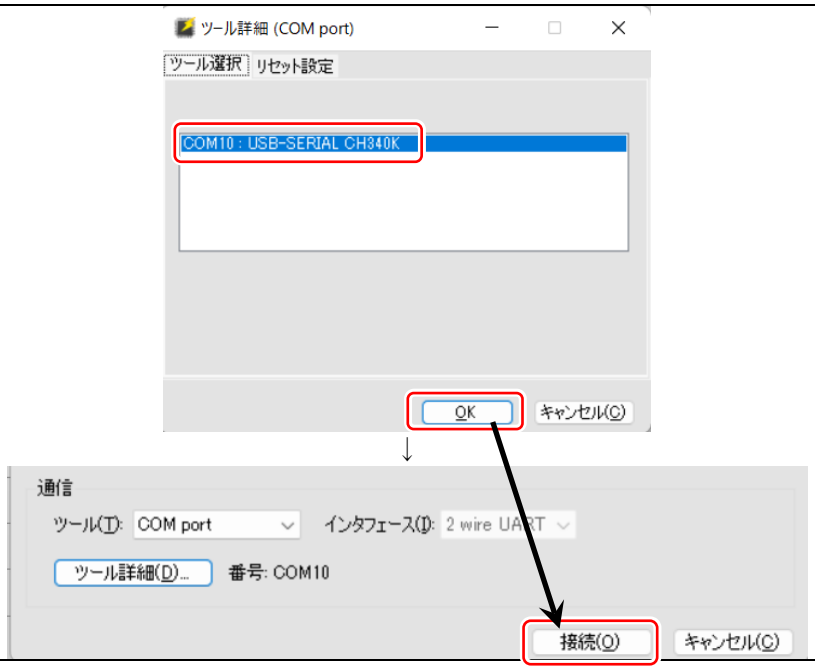
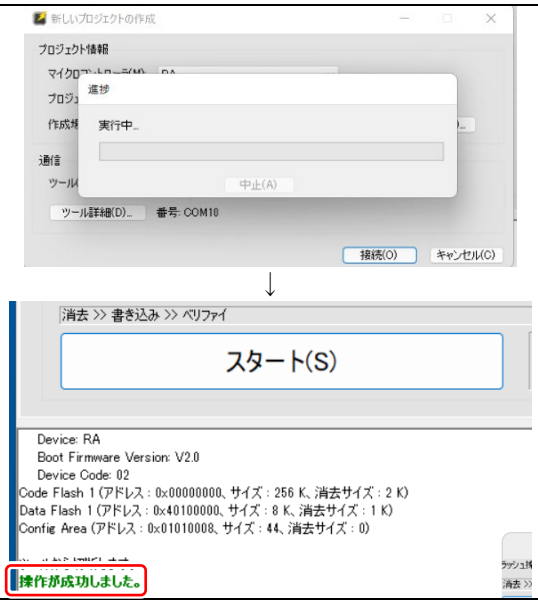
2		「追加のボードマネージャのURL」に左のような URL を追加して、OK をクリックします。
3		Arduino IDE を立ち上げます。 ❶左側の上から２個目の「 」をクリックします。 ❷検索欄に「rmc」と入力します。 ❸「 Arduino RMC-RA4M1 (rev2.0)」の『インストール』ボタンをクリックして、インストールしてください。 ❹再度「 」をクリックして、ボードマネージャの表示を消します。

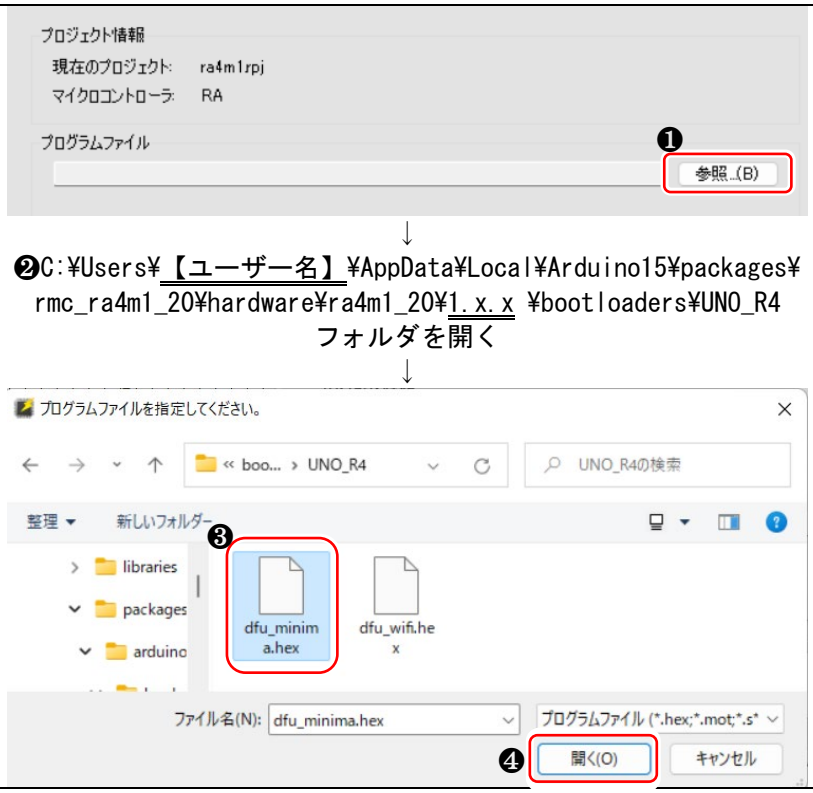
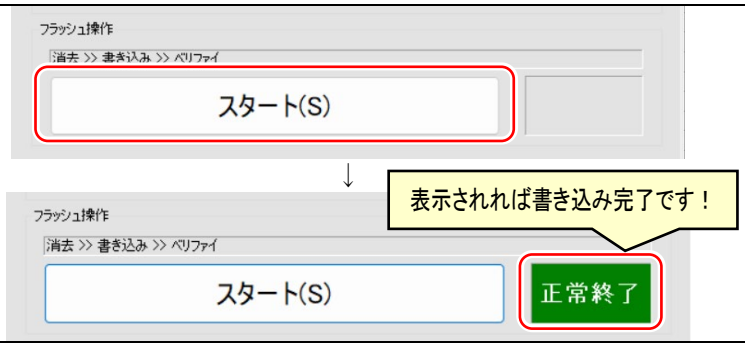
3.4. サンプルプログラムをコピーする

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。

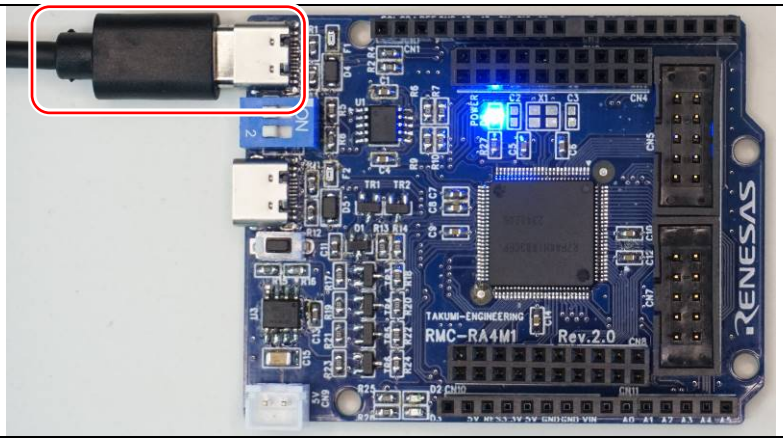
3.5. ファームウェアを書き込む

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。

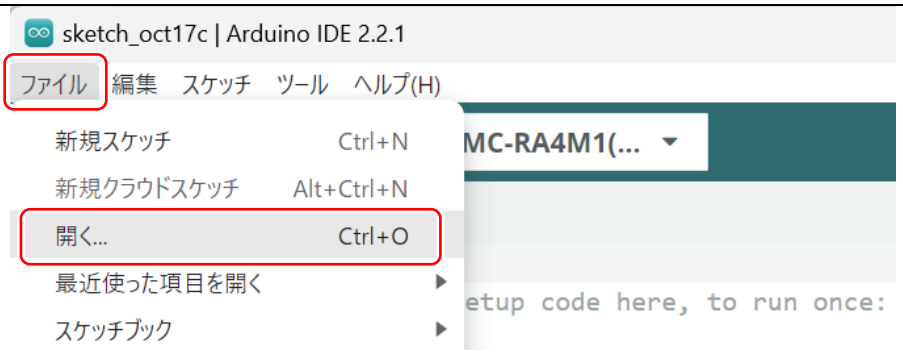
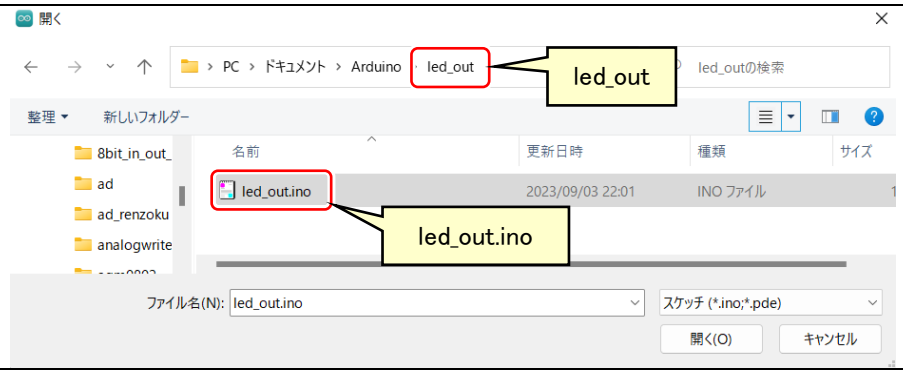
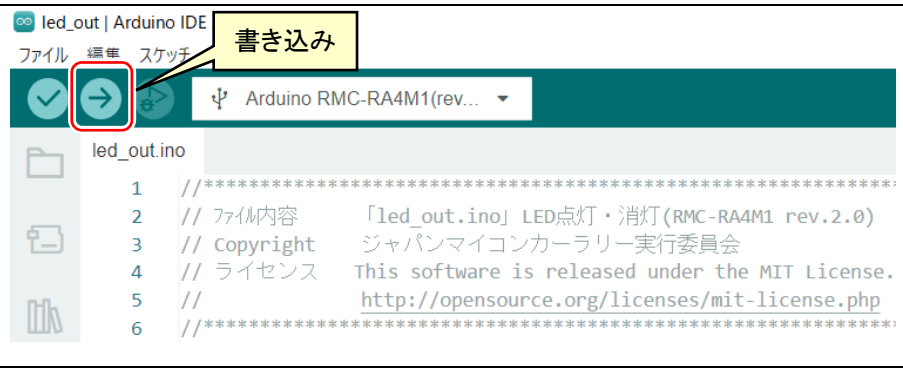
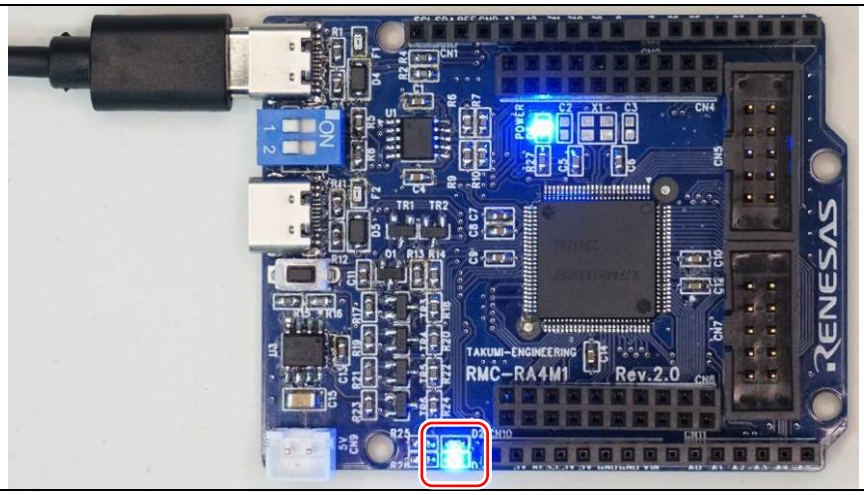
5		左のように入力、設定します。 ❶マイクロコントローラ「RA」 ❷プロジェクト名「ra4m1」など ※プロジェクト名は、任意の文字で構いません。 ❸作成場所：**変更しません** ❹ツール「COM port」 ❺「ツールの詳細」をクリックします。
6		ポート選択で、 「COM● USB-SERIAL CH340K」を選択し、「OK」をクリックします。 「接続」をクリックします。
7		「実行中」と表示され「接続が成功しました。」と表示されます。 ●通信に失敗した場合 ・「実行中」が数秒続いて、失敗する場合、「実行中」が表示されている間に、RMC-RA4M1ボードのリセットスイッチを押してみてください。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		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 でプログラムを書き込むときは、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.2.0)」を選択します。 ❹「USB DFU」を選択します。番号(左図では 3-1.2)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だ最初は「USB DFU」ですが、1回以上 ArduinoIDE でプログラムを書き込むと「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。

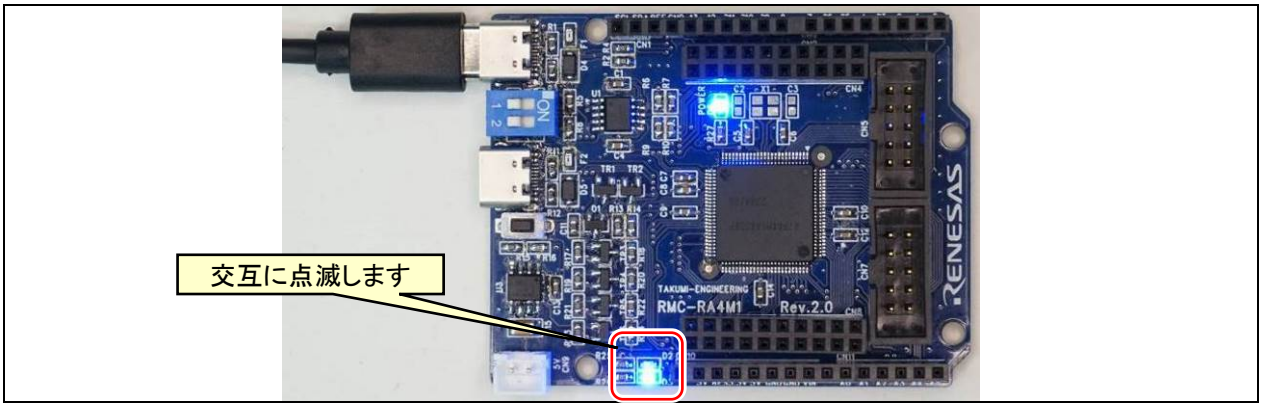
4. 演習

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(RMC-RA4M1 rev.2.0)
 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( 13, OUTPUT ); // LED D3
11 :  }
12 :
13 :  void loop() {
14 :    digitalWrite( 23, 1 );  // 1 で消灯
15 :    digitalWrite( 13, 0 );  // 0 で点灯
16 :    delay( 500 );
17 :
18 :    digitalWrite( 23, 0 );
19 :    digitalWrite( 13, 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 13 番ピンの LED 端子を出力(OUTPUT)に設定しています。

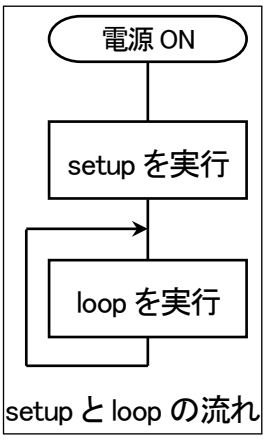
digitalWrite 関数は、端子から電圧を出力する命令です。23 番ピンの LED へ"1"(5V)を出力、13 番ピンの LED へ"0"(0V)を出力しています。
23 番ピンと 13 番ピンの LED は、5V を加えると消灯、0V を加えると点灯します。

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

(1) setup と loop の動作

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

(2) 端子の入出力設定(pinMode 関数)

マイコンのプログラムで最初にしなければいけないことは、マイコンの端子を入力にするか、出力にするかを設定することです。pinMode 関数で設定します。
※ポート(port)は港のことで、データが出入りすることから名付けられました。

使い方	pinMode(端子 , 入出力方向);	
端子	0～78、A0～A5　※同一の端子がある。例えば「A0」は「14」である。	
入出力方向	OUTPUT	端子を出力に設定する。 例えば LED は、マイコンの端子から LED へ電圧(0 か 5V か)を出力して光らせるので、出力に設定する。
	INPUT	端子を入力に設定する。 例えばスイッチは、スイッチの電圧をマイコンの端子で入力して、状態(0V が入力されているのか、5V なのか)を判断するので、入力に設定する。
	INPUT_PULLUP	端子を入力に設定し、マイコン内蔵のプルアップ抵抗（10k～50kΩの抵抗値）を有効にする。

使用例	pinMode(11 , OUTPUT); // D11 を出力端子とする　D は省略する
	pinMode(0 , INPUT); // D0 を入力端子とする
	pinMode(A0 , OUTPUT); // A0 を出力端子とする 「A0」を「14」としても OK　A は書く
	// A0～A5 端子はアナログ電圧を入力できる端子だが、
	// デジタル値の入力・出力も可能である。
	pinMode(13 , OUTPUT); // D13 を出力端子とする	

(3) 端子から出力(digitalWrite)

digitalWrite 関数は、pinMode で出力に設定した端子から"0"または"1"を出力する関数です。
"0" = 0V = LOW(ロー:「低い」の意味)
"1" = 5V = HIGH(ハイ:「高い」の意味)
となります。

使い方	digitalWrite(端子 , 出力);
端子	0～78、A0～A5　※同一の端子がある。例えば「A0」は「14」である。
出力	端子から 0V を出力したいとき、「0」または「LOW」とする。 端子から 5V を出力したいとき、「1」または「HIGH」とする。
注意事項	pinMode で出力端子に設定した端子しか使用できない。
使用例	digitalWrite (13 , 1);　// D13 から"1"(電圧でいうと 5V)を出力する。 digitalWrite (A0 , 0);　// A0 から"0"(電圧でいうと 0V)を出力する。 digitalWrite (0 , HIGH);　// D0 からハイ電圧(5V = "1")を出力する。0V は LOW。
備考	0V または 5V のどちらかしか出力することができない。 0～5V の任意の電圧を出力したい場合は、analogWrite 関数を使用する。(ただし、2.5V を出力しているように見せかけるだけで、実際は 0V または 5V が出力されている)。

(4) 時間稼ぎ(delay)

delay 関数は、指定した時間だけ一時停止する関数です。

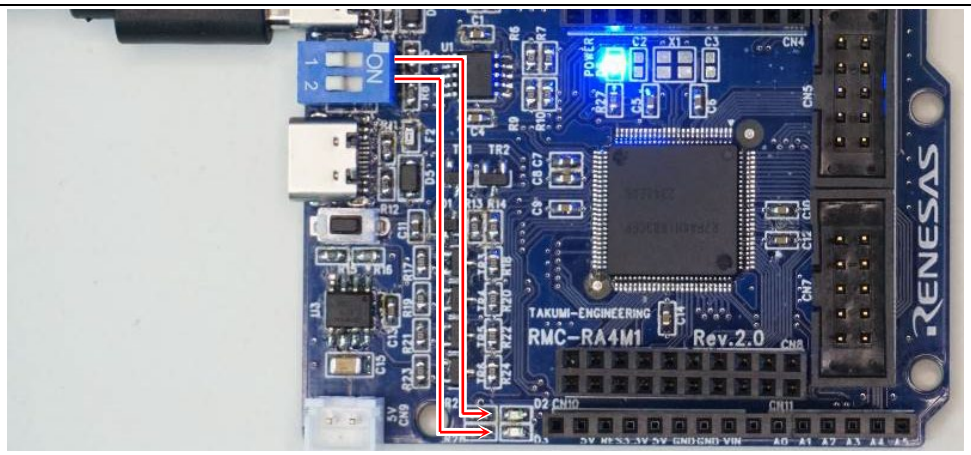
使い方	delay(時間[ms]);
時間[ms]	0～4,294,967,295ms の範囲で指定する。　※$(4,294,967,295)_{10}=(ffff\ ffff)_{16}$
注意事項	一時停止中は何もできない。例えば、delay 実行中はスイッチのチェックなどができない。
使用例	delay(1000);　// 1000ms (＝１秒) 一時停止

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(RMC-RA4M1 rev.2.0)
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( 13, 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( 13, 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。

4.2.3. プログラムの解説

(1) 端子の電圧を入力(digitalRead)

pinMode で入力に設定した端子から、端子に入力されている論理(0 か 1 か)を読み込む関数です。
①端子に Vcc×0.2V(Vcc=5.0V なら 1.0V)以下の電圧が入力されていると、"0"(＝LOW)と判断します。
②端子に Vcc×0.8V(Vcc=5.0V なら 4.0V)以上の電圧が入力されていると、"1"(＝HIGH)と判断します。
③上記①～②の間の電圧が入力された場合(Vcc=5.0V の場合、1.0～4.0V が入力された場合)は、しきい値("0"と"1"が切り替わる瞬間)の電圧が分かりません(保証されていないという)。手元にある基板で実験したところ約 3.26V をしきい値として"0"、"1"が変わりました。ただし、メーカーが保証していませんので、マイコンの個体差によってしきい値の電圧が異なる可能性があります。

使い方	digitalRead(端子);
端子	0～78、A0～A5　※同一の端子がある。例えば A0＝14 である。

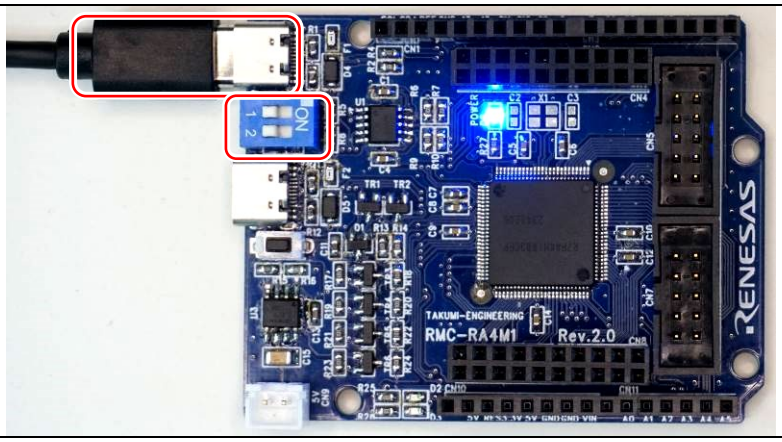
使用例	i = digitalRead (13);　// D13 端子に入力されている論理を読み込み変数 i に代入する
	　　// Vcc=5.0V の場合、1.0V 以下で"0"、4.0V 以上で"1"となる
	j = digitalRead (A0);　// A0 端子に入力されている論理を読み込み変数 j に代入する

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

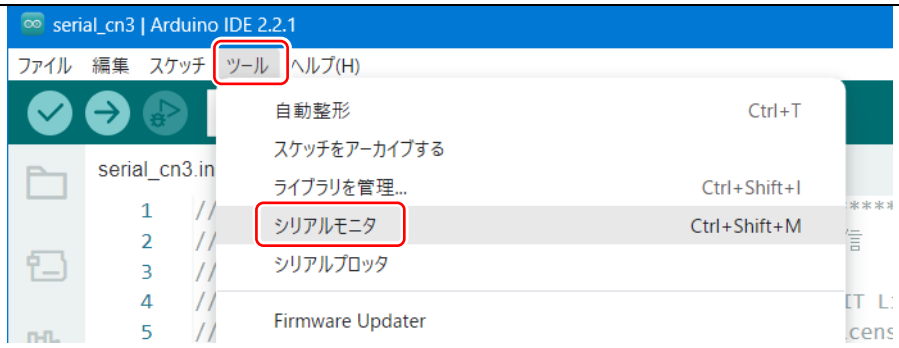
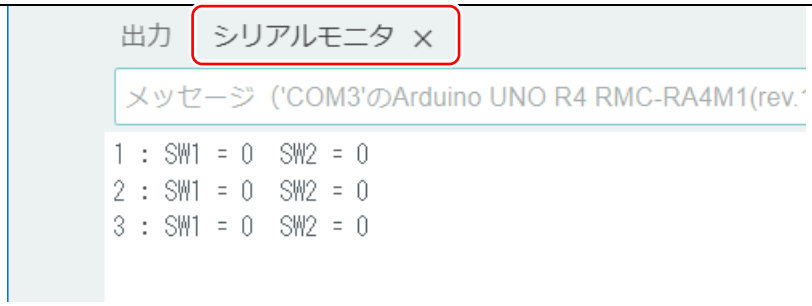
USB コネクタの CN3（端にある USB コネクタ)からシリアル出力して、ディップスイッチの状態をパソコンなどのシリアル通信ソフトに表示します。

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。



4.3.2. 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。

4.3.3. 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(RMC-RA4M1 rev.2.0)
 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( "CN3 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、CN3 の USB コネクタ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。多くのシリアル通信ソフトの初期の通信速度が 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、文字を出力した後、改行します。ただの print は改行しません。

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 行目で毎回値を代入しているので、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
```

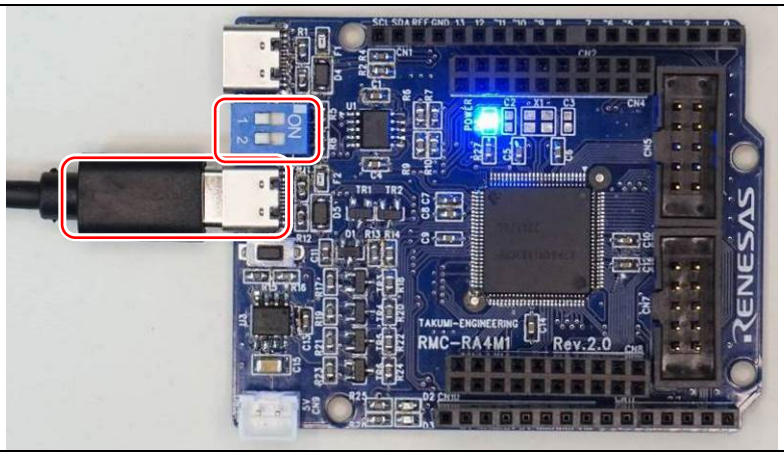
delay(1000)で 1000ms(1 秒) 待って、lp 変数を＋1 して、loop 関数を終わります。loop 関数は繰り返し実行されるので、lp 変数は loop 関数を繰り返すごとに＋1 されます。

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(RMC-RA4M1 rev.2.0)
 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 :    Serial2.begin( 9600 );  // Serial2 = CN6 の USB コネクタとシリアル接続
13 :
14 :    while ( !Serial2 ) {
15 :      // 接続されるまで待つ
16 :    }
17 :
18 :    Serial2.println( "CN6 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 1 OFF 側で 0
26 :    sw2 = !digitalRead( 26 );
27 :
28 :    Serial2.print( lp );
29 :    Serial2.print( " : " );
30 :    Serial2.print( "SW1 = " );
31 :    Serial2.print( sw1 );
32 :    Serial2.print( "  SW2 = " );
33 :    Serial2.println( sw2 );
34 :    delay( 1000 );
35 :    lp++;
36 :  }
```

Serial2 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「Serial2」の併用について

「Serial」と「Serial2」は併用可能です。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。

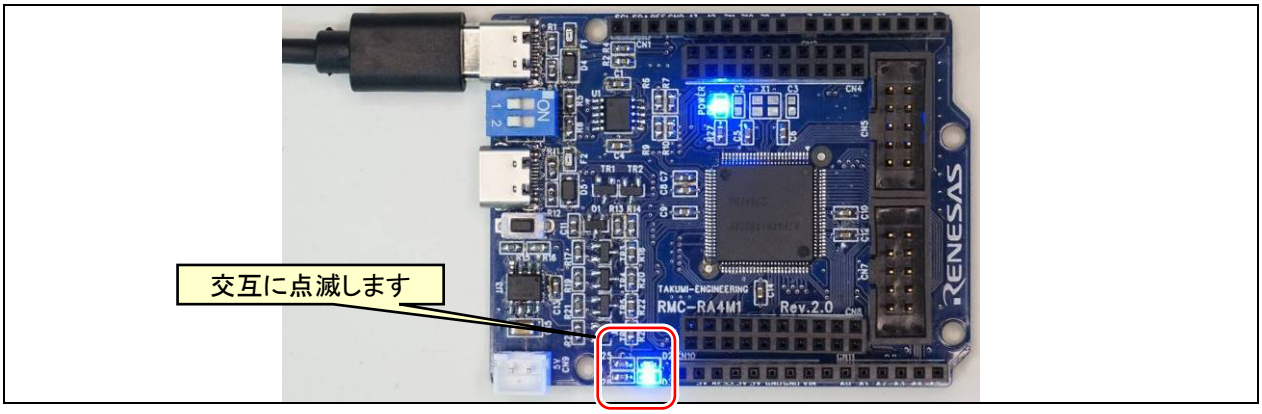
```
void setup() {
  Serial.begin( 9600 );
  Serial2.begin( 9600 );
}

void loop() {
  static int i = 1;
  Serial.println( i );    // Serial に出力
  Serial2.println( i );   // Serial2 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。



4.5.2. プログラム

```
1 :  //*****************************************************************************
2 :  // ﾌｧｲﾙ内容      「interrupt_1ms.ino」 1ms ごとの割り込み(RMC-RA4M1 rev.2.0)
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 :  unsigned long cnt;
17 :
18 :  void setup() {
19 :    pinMode( 23, OUTPUT ); // LED D2
20 :    pinMode( 13, 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( 13, 0 );
35 :    } else {
36 :      digitalWrite( 23, 0 );
37 :      digitalWrite( 13, 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 ごとに割り込みを発生させます。
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。

4.5.3. 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。

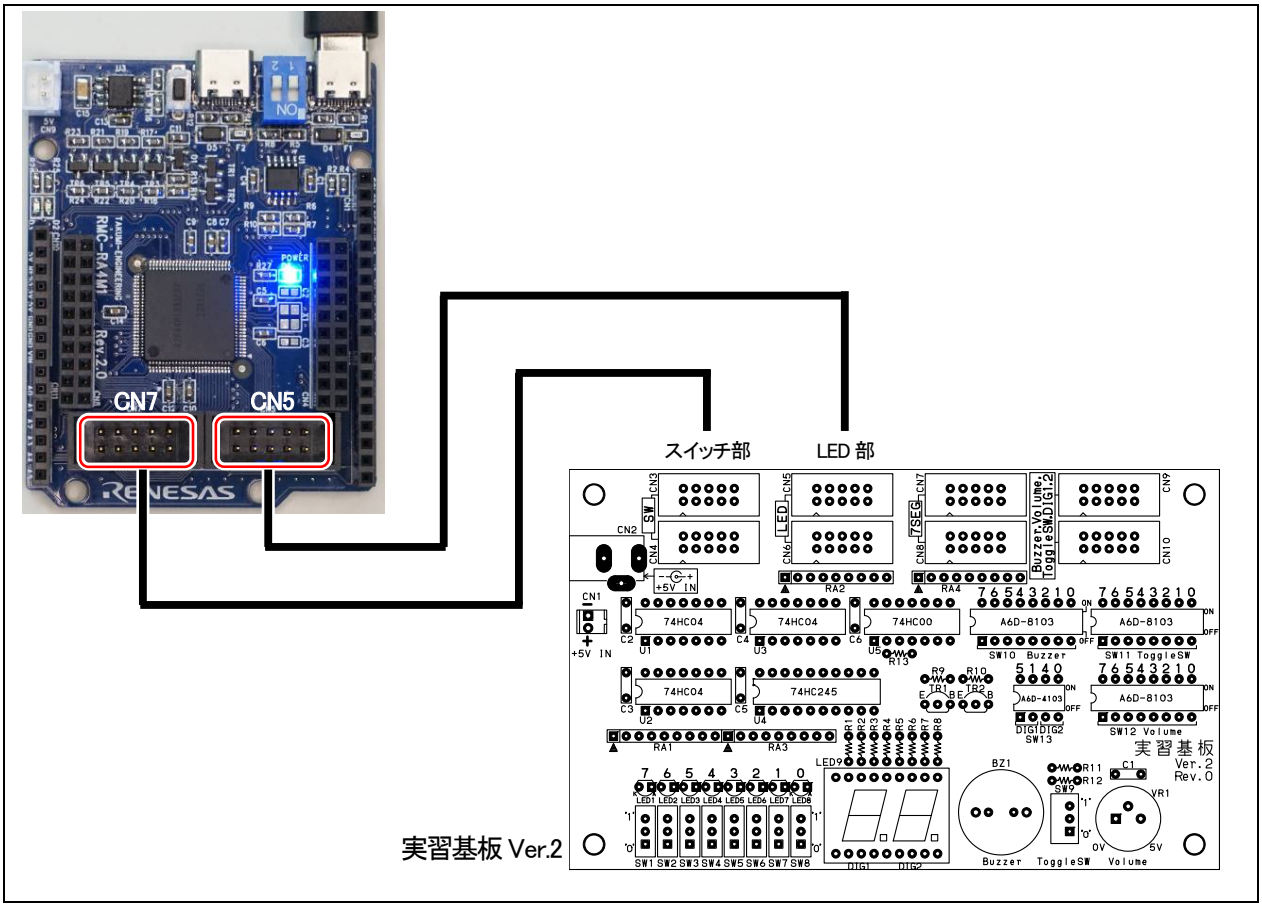
使い方	FspTimer :: begin(TIMER_MODE_PERIODIC, AGT_TIMER, 1, 値, 1, (timer_source_div_t)分周比, 割り込み関数名);
分周比	TIMER_SOURCE_DIV_1 : カウンタの周波数を 1 分周で使用 TIMER_SOURCE_DIV_2 : カウンタの周波数を 2 分周で使用 TIMER_SOURCE_DIV_8 : カウンタの周波数を 8 分周で使用
値	0～65,535
割り込み関数名	割り込み関数の関数名を設定します。
計算方法	割り込み周期の計算方法を示します。 AGT のカウンタは 24MHz で動作しています。割り込み周期は、 割り込み周期 ＝ 1/(24MHz÷分周比)×値 となります。値の最大は 65,535 です。 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）× 65536 ＝ 2.730667ms ２分周で設定できる最大割り込み周期：1/（24MHz÷2 分周）× 65536 ＝ 5.461333ms ８分周で設定できる最大割り込み周期：1/（24MHz÷8 分周）× 65536 ＝ 21.845333ms 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。21.845333ms 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 CN7 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 CN5 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 CN7 と実習基板 Ver.2 のスイッチ部、マイコンボードの CN5 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(RMC-RA4M1 rev.2.0)
 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 :    // CN7 DIPSW 8bit
14 :    pinMode( 35, INPUT );
15 :    pinMode( 36, INPUT );
16 :    pinMode( 37, INPUT );
17 :    pinMode( 38, INPUT );
18 :    pinMode( 39, INPUT );
19 :    pinMode( 40, INPUT );
20 :    pinMode( 41, INPUT );
21 :    pinMode( 42, INPUT );
22 :
23 :    // CN5 LED 8bit
24 :    pinMode( 27, OUTPUT );
25 :    pinMode( 28, OUTPUT );
26 :    pinMode( 29, OUTPUT );
27 :    pinMode( 30, OUTPUT );
28 :    pinMode( 31, OUTPUT );
29 :    pinMode( 32, OUTPUT );
30 :    pinMode( 33, OUTPUT );
31 :    pinMode( 34, 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( 35 ) << 7);
45 :    c |= (digitalRead( 36 ) << 6);
46 :    c |= (digitalRead( 37 ) << 5);
47 :    c |= (digitalRead( 38 ) << 4);
48 :    c |= (digitalRead( 39 ) << 3);
49 :    c |= (digitalRead( 40 ) << 2);
50 :    c |= (digitalRead( 41 ) << 1);
51 :    c |= (digitalRead( 42 ) << 0);
52 :
53 :    return c;
54 :  }
55 :
56 :  void led( unsigned char value ) {
57 :
58 :    digitalWrite( 27, (value & 0x80) != 0 );
59 :    digitalWrite( 28, (value & 0x40) != 0 );
60 :    digitalWrite( 29, (value & 0x20) != 0 );
61 :    digitalWrite( 30, (value & 0x10) != 0 );
62 :    digitalWrite( 31, (value & 0x08) != 0 );
63 :    digitalWrite( 32, (value & 0x04) != 0 );
64 :    digitalWrite( 33, (value & 0x02) != 0 );
65 :    digitalWrite( 34, (value & 0x01) != 0 );
66 :  }
```

CN7 の 35 番ピン～42 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。

CN5 の 27 番ピン～34 番ピンは、LED に接続するので、出力にします。

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

例えば、35～42 番ピンに”1”が入力されているとします。
・D35 が 1 なので、左に 7bit シフトすると 1000 0000 になります。
・D36 が 1 なので、左に 6bit シフトすると 0100 0000 になります。
・D37 が 1 なので、左に 5bit シフトすると 0010 0000 になります。
・D38 が 1 なので、左に 4bit シフトすると 0001 0000 になります。
・D39 が 1 なので、左に 3bit シフトすると 0000 1000 になります。
・D40 が 1 なので、左に 2bit シフトすると 0000 0100 になります。
・D41 が 1 なので、左に 1bit シフトすると 0000 0010 になります。
・D42 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。
例では(A)が 0 以外なので"1"(5V)を出力します。
もし(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「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)(RMC-RA4M1 rev.2.0)
 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 :    // CN7 DIPSW 8bit
14 :    pinMode( 35, INPUT );
15 :    pinMode( 36, INPUT );
16 :    pinMode( 37, INPUT );
17 :    pinMode( 38, INPUT );
18 :    pinMode( 39, INPUT );
19 :    pinMode( 40, INPUT );
20 :    pinMode( 41, INPUT );
21 :    pinMode( 42, INPUT );
22 :
23 :    // CN5 LED 8bit
24 :    pinMode( 27, OUTPUT );
25 :    pinMode( 28, OUTPUT );
26 :    pinMode( 29, OUTPUT );
27 :    pinMode( 30, OUTPUT );
28 :    pinMode( 31, OUTPUT );
29 :    pinMode( 32, OUTPUT );
30 :    pinMode( 33, OUTPUT );
31 :    pinMode( 34, 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_PORT0->PIDR_b.PIDR12) << 7;
45 :    c |= (R_PORT0->PIDR_b.PIDR11) << 6;
46 :    c |= (R_PORT0->PIDR_b.PIDR10) << 5;
47 :    c |= (R_PORT0->PIDR_b.PIDR8 ) << 4;
48 :    c |= (R_PORT0->PIDR_b.PIDR7 ) << 3;
49 :    c |= (R_PORT0->PIDR_b.PIDR6 ) << 2;
50 :    c |= (R_PORT0->PIDR_b.PIDR5 ) << 1;
51 :    c |= (R_PORT0->PIDR_b.PIDR4 ) << 0;
52 :
53 :    return c;
54 :  }
55 :
56 :  void led( unsigned char value ) {
57 :    R_PORT4->PODR_b.PODR2 = ( (value & 0x80) != 0 );  // 式が真なら 1、偽なら 0 の値を持つ
58 :    R_PORT4->PODR_b.PODR5 = ( (value & 0x40) != 0 );
59 :    R_PORT4->PODR_b.PODR6 = ( (value & 0x20) != 0 );
60 :    R_PORT4->PODR_b.PODR3 = ( (value & 0x10) != 0 );
61 :    R_PORT4->PODR_b.PODR4 = ( (value & 0x08) != 0 );
62 :    R_PORT4->PODR_b.PODR1 = ( (value & 0x04) != 0 );
63 :    R_PORT4->PODR_b.PODR0 = ( (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)で実行したいときに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(35) << 7);
c |= (digitalRead(36) << 6);
c |= (digitalRead(37) << 5);
c |= (digitalRead(38) << 4);
c |= (digitalRead(39) << 3);
c |= (digitalRead(40) << 2);
c |= (digitalRead(41) << 1);
c |= (digitalRead(42) << 0);
	c = (R_PORT0->PIDR_b.PIDR12) << 7;
	c |= (R_PORT0->PIDR_b.PIDR11) << 6;
	c |= (R_PORT0->PIDR_b.PIDR10) << 5;
	c |= (R_PORT0->PIDR_b.PIDR8) << 4;
	c |= (R_PORT0->PIDR_b.PIDR7) << 3;
	c |= (R_PORT0->PIDR_b.PIDR6) << 2;
	c |= (R_PORT0->PIDR_b.PIDR5) << 1;
	c |= (R_PORT0->PIDR_b.PIDR4) << 0;

RA4M1 マイコンのレジスタから、端子から入力、端子から出力する記述について下記に示します。

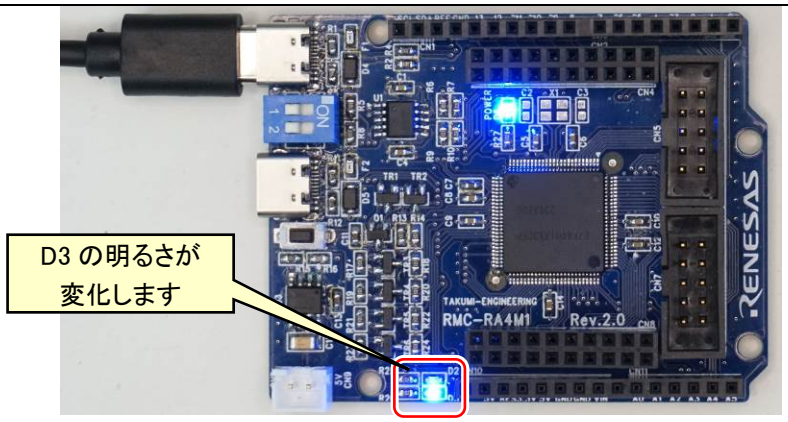
	プログラムの書き方	例	説明
入力	R_PORT●->PIDR_b.PIDR■	i = R_PORT0->PIDR_b.PIDR12;	P012端子に入力されている論理(0または1)を読み込みます。
出力	R_PORT●->PODR_b.PODR■	R_PORT4->PODR_b.PODR2 = 1;	P402端子から0または1(0Vまたは5V)を出力します。 ※「06」の十の桁が0なら、PODR側は1の桁のみ記載します。

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、A0 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 D3 の明るさを変化させます。



4.8.2. プログラム

```
 1 :  //*****************************************************************************
 2 :  // ﾌｧｲﾙ内容       「analogWrite.ino」 PWM 信出力(RMC-RA4M1 rev.2.0)
 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 :  }
10 :
11 :  void loop() {
12 :    static int i = 0;
13 :    analogWrite( 13, 255 - i );  // LED D2  PWM 出力
14 :    delay( 10 );
15 :    i++;
16 :    if( i >= 256 ) {
17 :      i = 0;
18 :    }
19 :  }
```

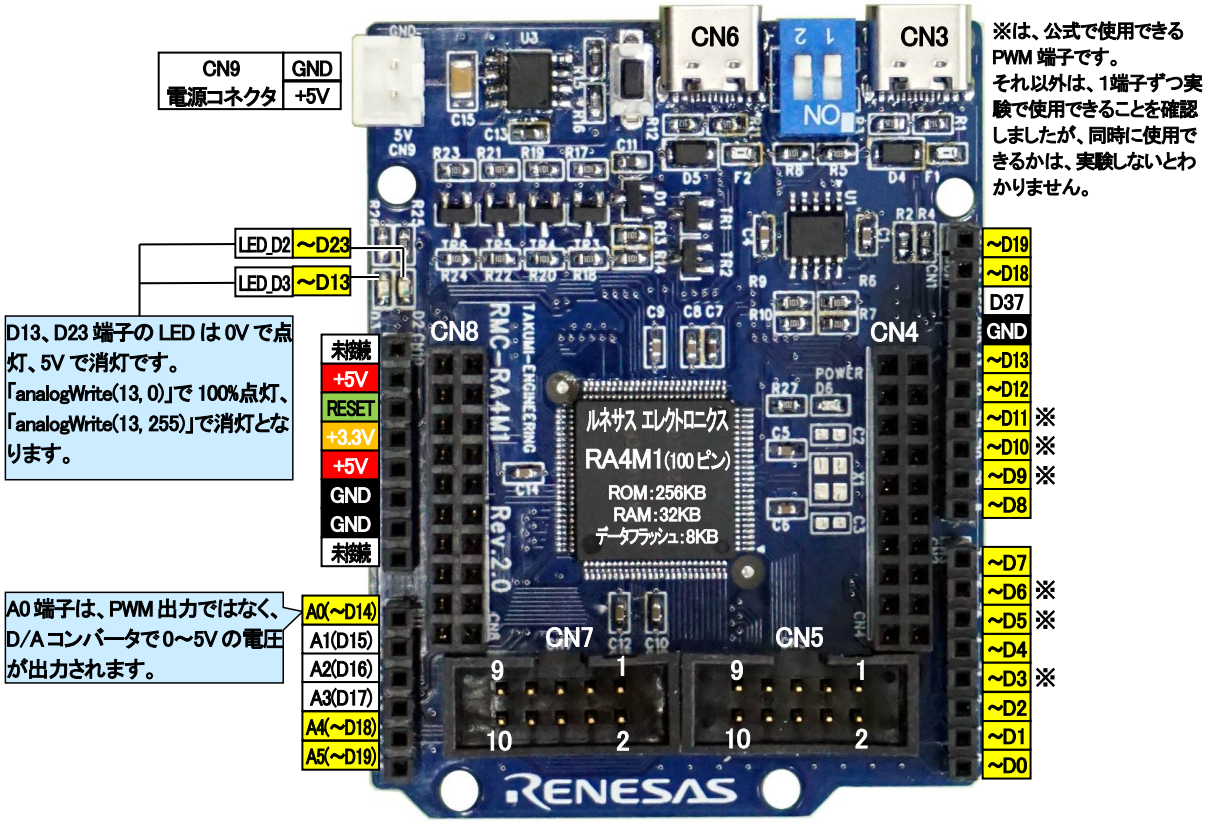
D13 端子へ analogWrite して、LED の光量を変化させます。
255 がデューティ比 100%です。
D13 端子の LED は"1"で消灯、"0"で点灯なので、0%のとき 255,100%のとき 0 になるよう 255 で引いています。
最初は 255(5V)で、10ms ごとに値が－1 していきます。

4.8.3. プログラムの解説

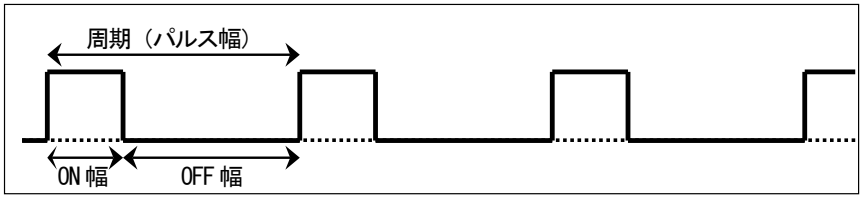
(1) 端子

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。ただし、これらの端子は同時に出力できない端子があります。

※公式には、D3、D5、D6、D9、D10、D11 端子が PWM 出力端子です。その他の PWM 端子は、1端子ずつ analogWrite 関数で PWM 出力することを確認しましたが、同時に使用できるかは実験しないとわかりません。同時に使用できるかは実験して確かめてください。確実に出力したいときは「gpt_pwm.ino」を使用してください。



(2) analogWrite 関数の使い方

使い方	analogWrite(端子 , ON の割合 0～255);
端子	0～13、A0、A4、A5
ON の割合	●A0 端子以外 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 (ON と OFF の平均が 2.5V で、実際に 2.5V 出力している訳ではありません)。  ●A0 端子 0～5V を出力します。0 で 0V、255 で 5V を出力します。

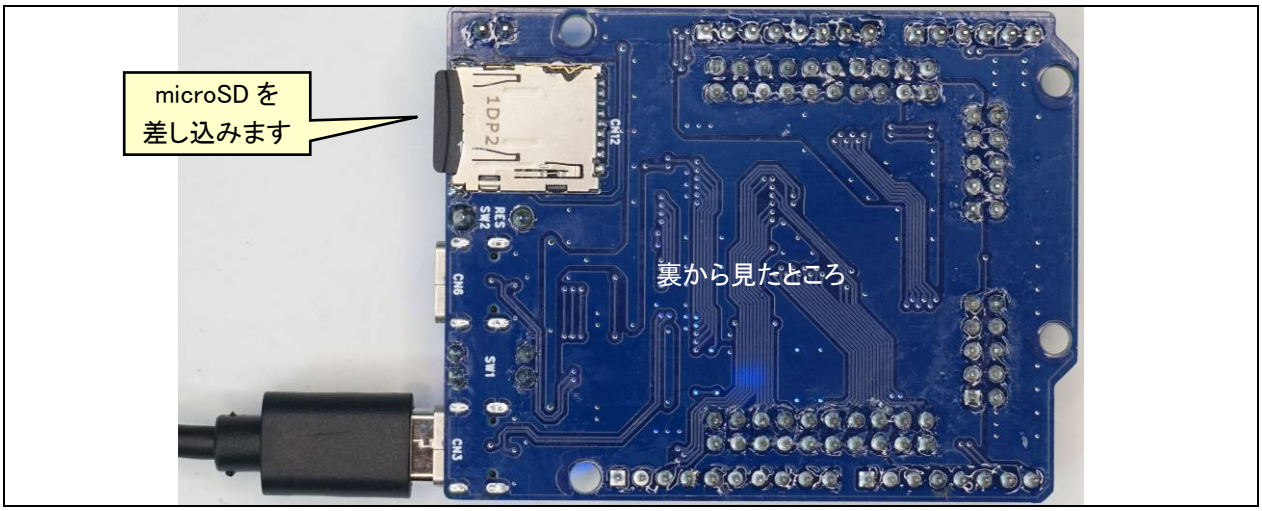
使用例	analogWrite (0 , 255); // D0 端子から、255(5V)を出力する
	analogWrite (1 , 128); // D1 端子から、(128/255)×5V＝2.5V を出力する

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

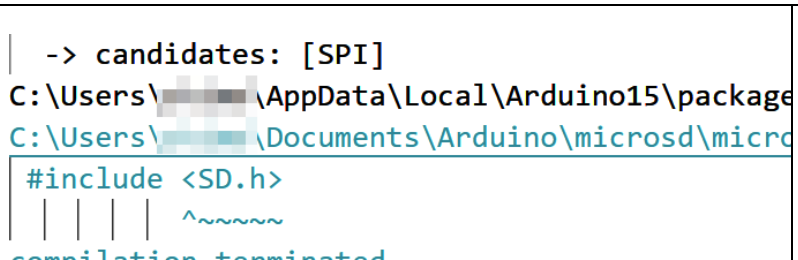
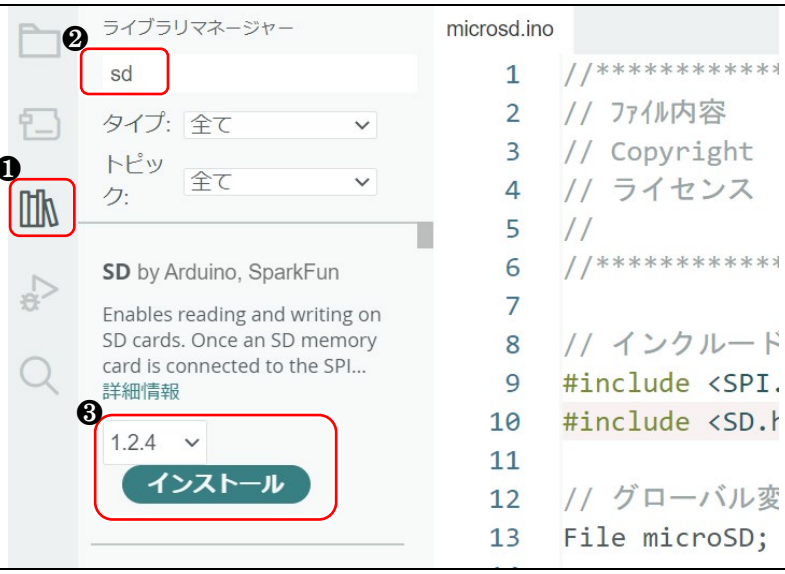
RMC-RA4M1 ボードに搭載されている microSD へ、ファイルの読み書きを行います。microSD コネクタへ 32GB (SDHC)以下の microSD を差し込んでください。なお、microSD のフォーマット形式は、FAT または FAT32 のみに対応しています。NTFS や exFAT フォーマットには対応していません（エラーで読み書きできません）。

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		「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(RMC-RA4M1 rev.2.0)
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 :  ArduinoSPI SPI(MISO1, MOSI1, SCK1, FORCE_SPI1_MODE); // RMC-RA4M1 の microSD 用 SPI を選択(SPI の各端子は pins_arduino.h で定義)
14 :  //ArduinoSPI SPI(MISO, MOSI, SCK, FORCE_SPI_MODE);    // Arduino UNO R4 の SPI を使用(MISO=12,MOSI=11,SCK=13)
15 :  File microSD;  // microSD のファイルアクセス用
```

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

RMC-RA4M1 ボードのマイコンには、SPI が 2 チャンネルあります。
基板搭載の microSD を使用するときは、13 行目を使用します(MISO1,MOSI1,SCK1)。
Arduino UNO R4 の SPI(MISO=12,MOSI=11,SCK=13)を使用するときは、14 行目を使用します。
どちらか 1 つを有効にしてください。どちらかの SPI の定義がないと、エラーになります。

File クラスで microSD インスタンスを作成します。

```
17 :  void setup() {
18 :    Serial.begin( 9600 );
19 :
20 :    while( !Serial ) {
21 :      // シリアル接続されるまで待つ
22 :    }
23 :
24 :    Serial.println( "" );
25 :    Serial.print( "1 microSD を確認中です... " );
26 :
27 :    if( !SD.begin( 2e6, CS1 ) ) {    // 通信速度Hz(e6=10の6乗),microSDのCS1(pins_arduino.hで定義)  Arduino UNO R4は「CS」にする
```

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。
引数は、最初が通信速度[Hz]、2 つ目が CS 端子を設定します。通信速度の「2e6」は「2×10^6」の意味になります。基板搭載の microSD を使用するときは「CS1」を選択してください。

```
28 :      Serial.println( "microSD が認識されません!" );
29 :      while( 1 );  // 終了
30 :    } else {
31 :      // microSD 認識 OK!
32 :      Serial.println( "microSD を認識しました。" );
33 :    }
34 :
35 :    // microSD へ書き込み
36 :    microSD = SD.open( "test.txt", FILE_WRITE );
37 :    if( microSD != 0 ) {  // エラーがないなら
38 :      microSD.println( "テストです。" );  // ファイルの末尾に追加で書き込む
39 :      microSD.close();
40 :      Serial.println( "2 microSD に書き込みができました。" );
```

microSD を認識しないと、無限ループでプログラムはここで停止します。

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_READ」なので、読み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「テストです。」の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。

```
41 :    } else {
42 :      Serial.println( "2 ファイル書き込み(FILE_WRITE)ができませんでした。" );
43 :      while( 1 );
44 :    }
45 :
46 :    // microSD から読み込み
47 :    microSD = SD.open( "test.txt", FILE_READ );
48 :    if( microSD != 0 ) {  // エラーがないなら
49 :      Serial.println( "3 ファイルの内容を表示します。" );
50 :      Serial.println( "------------------------------ ここから" );
51 :      while ( microSD.available() ) {     // ファイルにデータがあれば
52 :        Serial.write( microSD.read() );   // データを読み込んでシリアルに出力
53 :      }
54 :      microSD.close();
```

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。

「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。

```
55 :     } else {
56 :       Serial.println( "3 ファイル読み込み(FILE_READ)ができませんでした。" );
57 :       while( 1 );
58 :     }
59 :
60 :     Serial.println( "------------------------------ ここまで" );
61 :     Serial.println( "4 プログラムを終了します。" );
62 :   }
63 :
64 :   void loop() {
65 :     // 何もしない
66 :   }
```

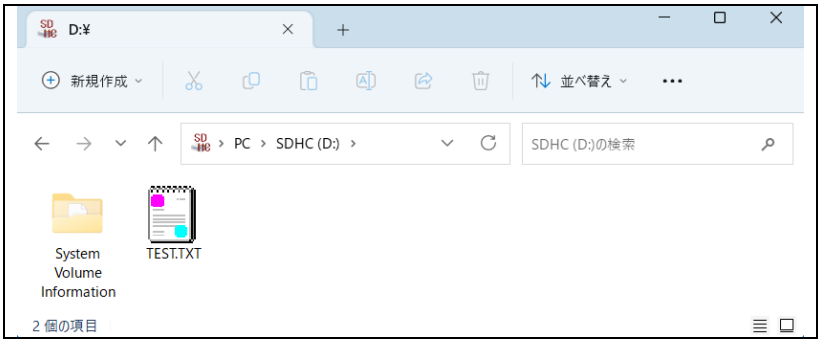
4.9.4. プログラムの解説

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。

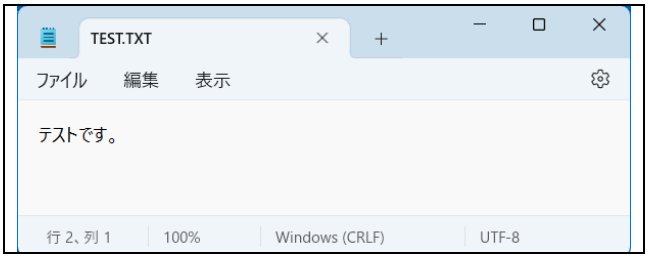
```
1 microSD を確認中です... microSD を認識しました。
2 microSD に書き込みができました。
3 ファイルの内容を表示します。
------------------------------ ここから
テストです。
------------------------------ ここまで
4 プログラムを終了します。
```

「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。



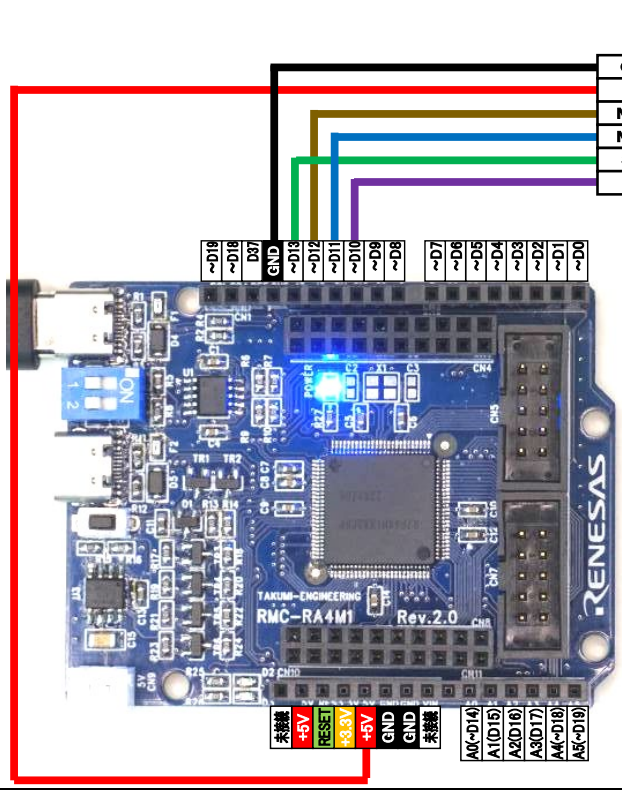
microSD が挿入されていなかったり、microSD のフォーマットが FAT や FAT32 でない場合は、次のようにシリアルモニターにエラーが表示されます。

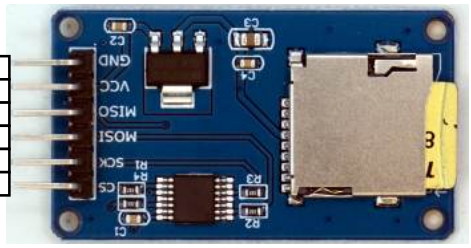
```
1 microSD を確認中です... microSD が認識されません!
```

4.9.5. Arduino UNO R4 と互換端子(D10～D13 端子)に接続して microSD を使用する

SD カードモジュールを使用する配線、プログラムを説明します。
SD カードモジュールは「arduino microsd」などで検索すると、基板が検索されます。

(1) 配線





SD カードモジュール

RMC-RA4M1(Arduino UNO R4)と SD カードモジュールの接続は下記のようにします。
D13 — SCK
D12 — MISO
D11 — MOSI
D10 — CS

(2) プログラム

「microsd.ino」を下記のように改造します。
13行目を、コメント(先頭に「//」を追加)してください。行ごと、削除しても構いません。
14行目のコメント(「//」を削除して、プログラムを有効にします。
27行目の「CS1」を「CS」に変更します(「1」を削除)。「CS」は 10 番ピンに指定されています。CS 端子は変更可能です。この場合、「CS」部分を端子番号(4 など)に変更してください。SD カードモジュールとの配線も変更してください。

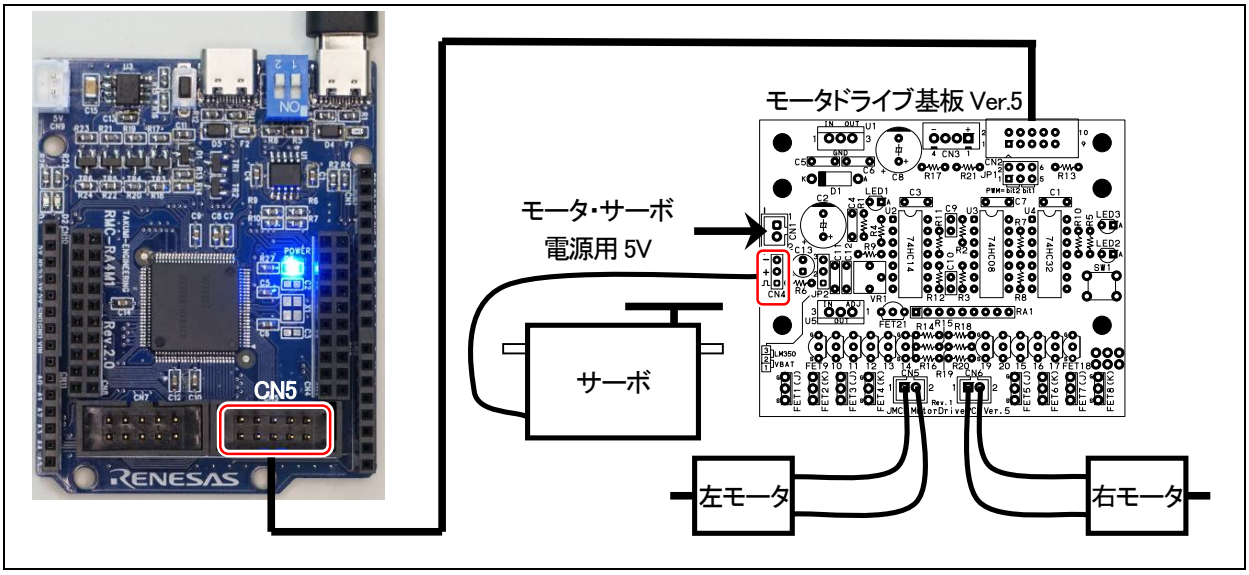
```
13 :  //ArduinoSPI SPI(MISO1, MOSI1, SCK1, FORCE_SPI1_MODE); // この行をコメント（無効）にする)
14 :  ArduinoSPI SPI(MISO, MOSI, SCK, FORCE_SPI_MODE);        // コメントをとって、この行を有効にする
中略
27 :    if( !SD.begin( 2e6, CS ) ) {    //  「CS1」を「CS」に変更する
```

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(RMC-RA4M1 rev.2.0)
 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 :  void led_out( unsigned char led );
21 :
```

周期を設定します。
サーボ PWM 周期は 16ms です。16ms は計算すると 48,000 となり設定値は１小さい 47,999 を設定します。

```
22 :  void setup() {
23 :    pinMode( 25, INPUT );                 // DIPSW1
24 :    pinMode( 26, INPUT );                 // DIPSW2
25 :
26 :    setGPTterminal( 4, 6 );               // P406(GTIOC1B):サーボ PWM
27 :    setGPTterminal( 4, 1 );               // P401(GTIOC6B):左モータ PWM
28 :    setGPTterminal( 4, 3 );               // P403(GTIOC3A):右モータ PWM
29 :
30 :    GTIOC1B = SERVO_CENTER;
31 :    startPWM_GPT1( GTIOCB , DIV16, PWM_CYCLE );  // GPT1 GTIOCB を使用 分周:16 周期:PWM_CYCLE
32 :
33 :    GTIOC6B = 0;
34 :    startPWM_GPT6( GTIOCB , DIV16, PWM_CYCLE );  // GPT6 GTIOCB を使用 分周:16 周期:PWM_CYCLE
35 :
36 :    GTIOC3A = 0;
37 :    startPWM_GPT3( GTIOCA , DIV16, PWM_CYCLE );  // GPT3 GTIOCA を使用 分周:16 周期:PWM_CYCLE
38 :
39 :    // CN5 モータドライブ基板
40 :    pinMode( 27 , OUTPUT );           // LED2
41 :    pinMode( 28 , OUTPUT );           // LED3
42 :    //pinMode( 29 , OUTPUT );         // サーボ PWM なので設定しない
43 :    //pinMode( 30 , OUTPUT );         // 右モータ PWM なので設定しない
44 :    pinMode(31, OUTPUT);              // 右モータ方向
45 :    //pinMode( 32 , OUTPUT );         // 左モータ PWM なので設定しない
46 :    pinMode(33, OUTPUT);              // 左モータ方向
47 :    pinMode(34, INPUT);               // プッシュスイッチ
48 :  }
49 :
50 :  void loop() {
51 :
52 :    led_out( 0x0 );
53 :    handle( 0 );
54 :    motor( 0, 0 );
55 :    delay( 1000 );
56 :
57 :    led_out( 0x1 );
58 :    handle(10 );
59 :    motor( 0, 25 );
60 :    delay( 1000 );
61 :
62 :    led_out( 0x2 );
63 :    handle( -10 );
64 :    motor( 25, 0 );
65 :    delay( 1000 );
66 :
67 :    led_out( 0x3 );
68 :    handle( 20 );
69 :    motor( 0, -50 );
70 :    delay( 1000 );
71 :
72 :    led_out( 0x2 );
73 :    handle( -20 );
74 :    motor( -50, 0 );
75 :    delay( 1000 );
76 :  }
77 :
```

モータドライブ基板 LED2:消灯 LED3:消灯
ハンドル角度:0 度
左モータ:0%　右モータ:0%
これらを 1000ms 動作させます。

モータドライブ基板 LED2:消灯 LED3:点灯
ハンドル角度:右 10 度
左モータ:0%　右モータ:正転 25%
これらを 1000ms 動作させます。

モータドライブ基板 LED2:点灯 LED3:消灯
ハンドル角度:左 10 度
左モータ:25%　右モータ:0%
これらを 1000ms 動作させます。

モータドライブ基板 LED2: 点灯 LED3:点灯
ハンドル角度:右 20 度
左モータ:0%　右モータ:逆転 50%
これらを 1000ms 動作させます。

モータドライブ基板 LED2:点灯 LED3:消灯
ハンドル角度:左 20 度
左モータ:逆転 50%　右モータ:0%
これらを 1000ms 動作させます。

```
78 : //*****************************************************************
79 : // モータ速度制御
80 : // 引数    左モータ:-100～100、右モータ:-100～100
81 : //         0 で停止、100 で正転 100%、-100 で逆転 100%
82 : // 戻り値 なし
83 : //*****************************************************************
84 : void motor(int accele_l, int accele_r) {
85 :   int sw_data;
86 :
87 :   sw_data = dipsw_get() + 7;  // 7～10
88 :   accele_l = accele_l * sw_data / 10;
89 :   accele_r = accele_r * sw_data / 10;
90 :
91 :   // 左モータ制御
92 :   if(accele_l >= 0) {
93 :     R_PORT4->PODR_b.PODR0 = 0;
94 :     accele_l = (long)(PWM_CYCLE + 1) * accele_l / 100;
95 :     GTIOC6B = accele_l;
96 :   } else {
97 :     R_PORT4->PODR_b.PODR0 = 1;
98 :     accele_l = (long)(PWM_CYCLE + 1) * (-accele_l) / 100;
99 :     GTIOC6B = accele_l;
100 :  }
101 :
102 :  // 右モータ制御
103 :  if(accele_r >= 0) {
104 :    R_PORT4->PODR_b.PODR4 = 0;
105 :    accele_r = (long)(PWM_CYCLE + 1) * accele_r / 100;
106 :    GTIOC3A = accele_r;
107 :  } else {
108 :    R_PORT4->PODR_b.PODR4 = 1;
109 :    accele_r = (long)(PWM_CYCLE + 1) * (-accele_r) / 100;
110 :    GTIOC3A = accele_r;
111 :  }
112 : }
113 :
114 : //*****************************************************************
115 : // サーボハンドル操作
116 : // 引数    サーボ操作角度：-90～90
117 : //         -90 で左へ 90 度、0 でまっすぐ、90 で右へ 90 度回転
118 : //*****************************************************************
119 : void handle(int angle) {
120 :   // サーボが左右逆に動く場合は、「-」を「+」に替えてください
121 :   GTIOC1B = SERVO_CENTER - angle * HANDLE_STEP;
122 : }
123 :
124 : //*****************************************************************
125 : // ディップスイッチ値読み込み
126 : // 戻り値 スイッチ値 0～3
127 : //*****************************************************************
128 : unsigned char dipsw_get(void) {
129 :   unsigned char sw = ( !(R_PORT3->PIDR_b.PIDR6) << 1 ) + ( !(R_PORT3->PIDR_b.PIDR7) );
130 :
131 :   return sw;
132 : }
133 :
```

```
134 :  //*****************************************************************
135 :  // モータドライブ基板の LED 制御
136 :  // 引数　スイッチ値 LED2:bit1 LED3:bit0　"0":消灯 "1":点灯
137 :  // 例)0x3→LED2:ON LED3:ON　0x2→LED2:ON LED3:OFF
138 :  //*****************************************************************
139 :  void led_out( unsigned char led )
140 :  {
141 :    R_PORT4->PODR_b.PODR2 = (led & 0x02) != 0 ? 1 : 0;
142 :    R_PORT4->PODR_b.PODR5 = (led & 0x01) != 0 ? 1 : 0;
143 :  }
```

4.10.3. 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 PWM を出力します。

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い、端子を細かく設定したいということで、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。

```
26 :    setGPTterminal( 4, 6 );              // P406(GTIOC1B):サーボ PWM
27 :    setGPTterminal( 4, 1 );              // P401(GTIOC6B):左モータ PWM
28 :    setGPTterminal( 4, 3 );              // P403(GTIOC3A):右モータ PWM
```

このときの注意点は、同じ GTIOCxy の番号を重複して選択できないということです。例えば今回のサーボ PWM 端子の P406 は、GTIOC1B 端子です。他に GTIOC1B 端子は、P104(D3 端子)、P110(D12 端子)にもありますが、1 端子しか GTIOC1B 端子に設定することができません。下記に記述してはいけない例を示します。

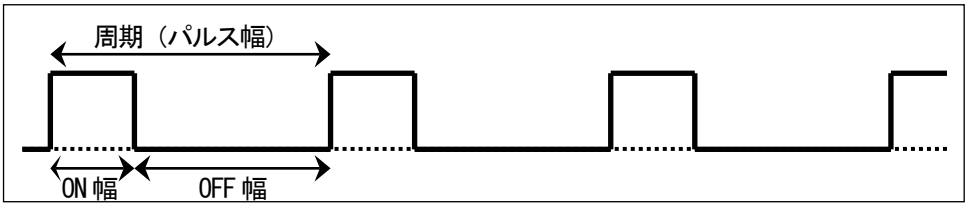
```
setGPTterminal( 4,  6 );             // P406(GTIOC1B) GTIOC1B 端子は１つしか設定できません！
setGPTterminal( 1,  4 );             // P104(GTIOC1B)
setGPTterminal( 4, 10 );             // P410(GTIOC1B)
```

③PWM 波形の ON 幅、周期を設定します。この部分のプログラムは、30 行～37 行です。

```
30 :    GTIOC1B = SERVO_CENTER;                          サーボ PWM の ON 幅の設定
31 :    startPWM_GPT1( GTIOCB , DIV16, PWM_CYCLE );      サーボ PWM の周期設定
32 :
33 :    GTIOC6B = 0;                                     左モータ PWM の ON 幅の設定
34 :    startPWM_GPT6( GTIOCB , DIV16, PWM_CYCLE );      左モータ PWM の周期設定
35 :
36 :    GTIOC3A = 0;                                     右モータ PWM の ON 幅の設定
37 :    startPWM_GPT3( GTIOCA , DIV16, PWM_CYCLE );      右モータ PWM の周期設定
```

startPWM_GPTx(x＝0～7)関数、GTIOCxy(x＝0～7、y＝A または B)について説明します。

(3) startPWM_GPTx 関数の使い方

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 x に設定したチャネル番号 0～7 の 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 65535 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。上記より、「DIV16」を選びます。 最大の PWM 周期は DIV1024 の 1398.1013[ms]です。これ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 D50(P411:GTIOC6A)と D51(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} / 1) } = 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
	GTIOC6A = 0; // ON 幅を設定　次で説明
	GTIOC6B = 0; // ON 幅を設定　次で説明
	startPWM_GPT6(GTIOCA | GTIOCB , DIV1 , 47999);
	(たてぼう)
	となります。

例２	CN8 の D73(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
	GTIOC5B = 0; // ON 幅を設定　次で説明
	startPWM_GPT5(GTIOCB , DIV64 , 18749);
	となります。

(4) ON 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	先ほど計算した PWM 周期の値と同じ値を設定すると 100%になります。0 にすると 0%になります。 例えば上記例１と同様に、分周比＝DIV1、周期＝48000 を設定して、1ms の周期の PWM 波形を GTIOC6A 端子と GTIOC6B 端子に出力しているとします。 GTIOC6A 端子の ON 幅を 0.1ms にしたければ、0.1ms は周期 1ms の 0.1 倍なので、 周期 48000 の 0.1 倍である 4800 を設定します。設定するときは 1 小さい値とします。次のようになります。 GTIOC6A = 4799; GTIOC6B 端子の ON 幅を 0.5ms にしたければ、0.5ms は周期 1ms の 0.5 倍なので、 周期 48000 の 0.5 倍である 24000 を設定します。設定するときは 1 小さい値とします。次のようになります。 GTIOC6B = 23999;

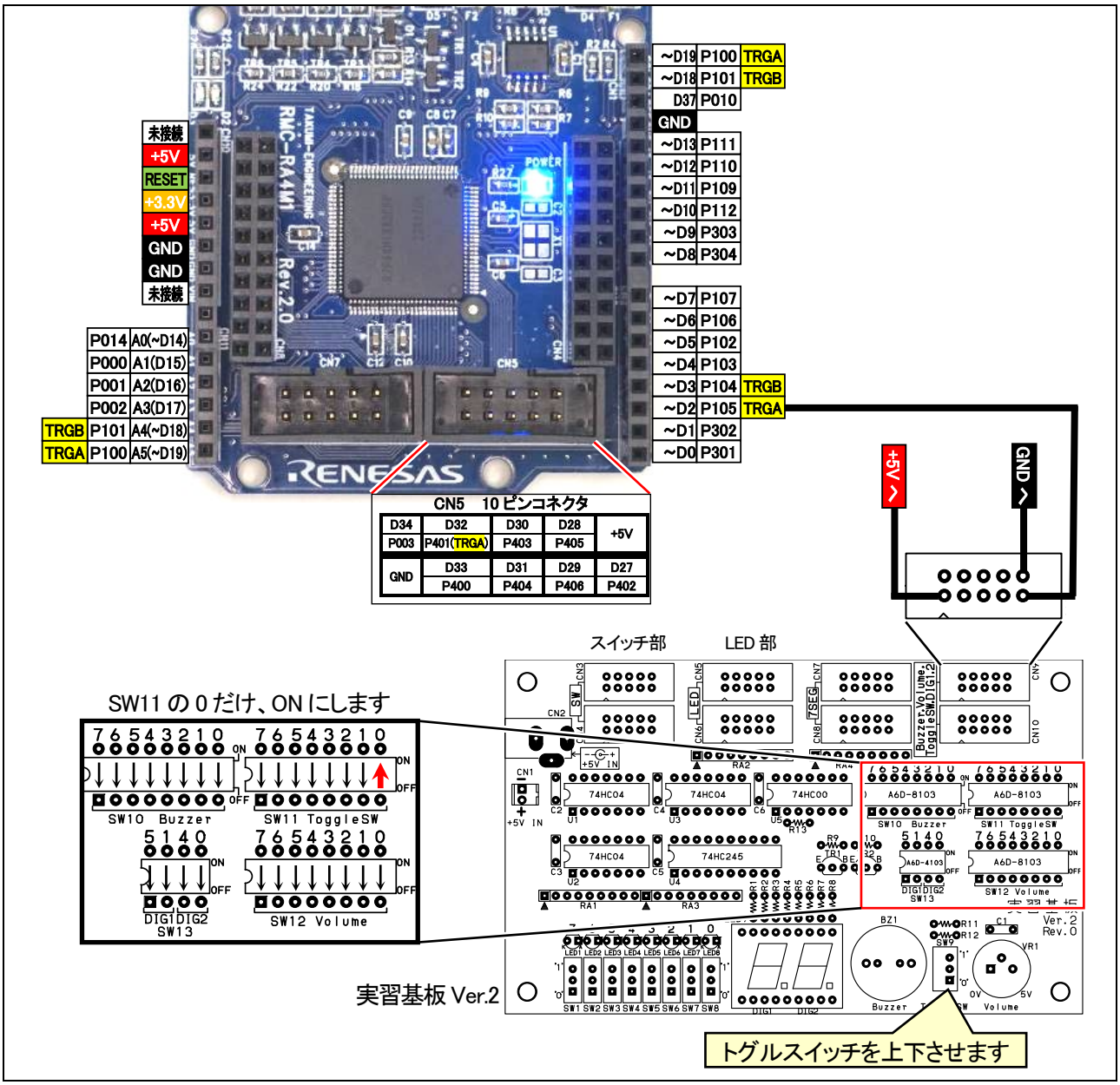
例	// CN5 の D32(P401)端子から、周期 16ms の PWM 波形を出力
	setGPTterminal(4, 1); // P401(GTIOC6B):左モータ PWM として使用
	GTIOC6B = 0; // ON 幅を設定
	startPWM_GPT6(GTIOCB , DIV16, 47999); // 周期 16ms
	
	int i = 0;
	while(1) {
	GTIOC6B = i; // ON 幅の設定(本当は1小さい値を設定しないといけませんが、例なので1小さくしていません)
	delay(1000); // 1000ms のディレイ
	i = i + 4800;
	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

D2(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。
1相のロータリエンコーダは、下図黄色部分の「TRGA」、または「TRGB」と書かれた端子に接続することができます。



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(RMC-RA4M1 rev.2.0)
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(D2 端子)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.println( "ロータリエンコーダ 1 相確認プログラム" );
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。使える端子は、配線図の TRGA 端子と TRGB 端子(黄色部分)です。

(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、TGRA 端子または 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、D2 端子(P105、TRGA 端子)を、パルス入力端子にしています。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。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(D2 端子)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。左桁が「0」の場合、0 は書きません。
例	startGPT2_1SouEncoder(GTETRGA , 4, 1); // 1 相エンコーダ GPT2 P401 の GTETRGA を使用

(4) カウント値の読み込み方

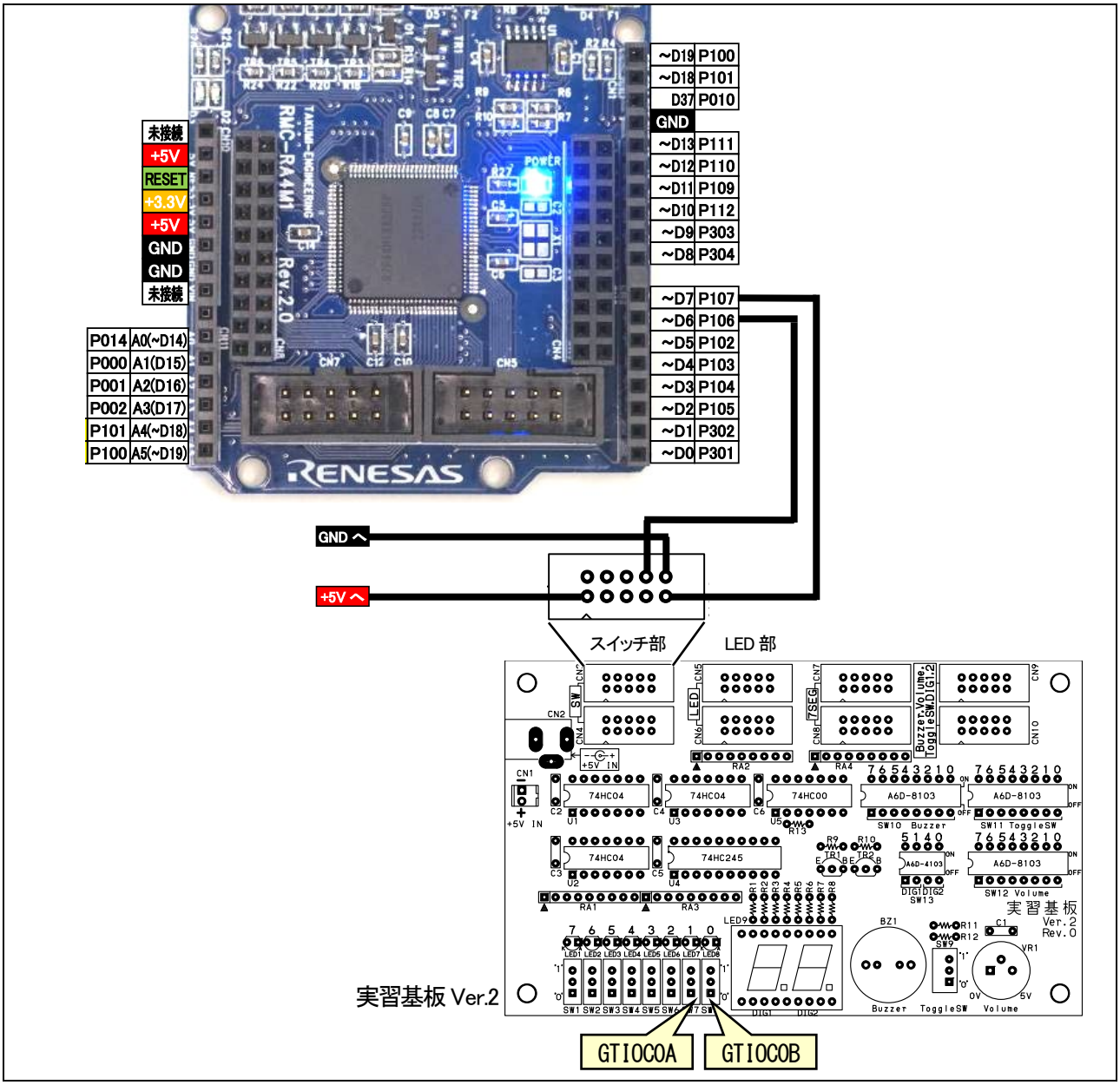
使い方	「GPTx_CNT」の値を読み込むと、カウント値が読み込まれます。 x にはチャネル番号 0～7 までの数字が入ります。 チャネル 0～1、は 0～2^{32}-1 (4,294,967,295) の範囲でカウントすることができます。 チャネル 2～7、は 0～2^{16}-1 (65,535) の範囲でカウントすることができます。
例	i = GPT2_CNT; // GPT2 のカウント値を変数 i に代入

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

D6(P106)と D7(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

```
//*************************************************************************
// 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(RMC-RA4M1 rev.2.0)
// Copyright    ジャパンマイコンカーラリー実行委員会
// ライセンス   This software is released under the MIT License.
//              http://opensource.org/licenses/mit-license.php
//*************************************************************************

// インクルード宣言
#include "mcr_gpt_lib.h"

void setup() {
  startGPT0_2SouEncoder( 1, 7 , 1, 6);  // 2 相エンコーダ 0A=P107(D7 端子) 0B=P106(D6 端子)を使用

  Serial.begin(9600);                   // シリアル通信 速度 9600bps

  while ( !Serial ) {
    // 接続されるまで待つ
  }
  Serial.println( "ロータリエンコーダ２相確認プログラム" );
}

void loop() {
  static int lp = 1;

  Serial.print( lp );
  Serial.print( " : " );
  Serial.println( INT_GPT0_CNT );
  lp++;
  delay( 1000 );
}
```

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。

```
#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 の番号)。x の番号は同じ端子を使用します。例えば、GTIOC2A 端子と GTIOC2B 端子などです。GTIOC1A 端子と GTIOC2B 端子など、数字が違う端子は使用できません。
今回は、D6 端子(P106、GTIOC0B 端子)と D7 端子(P107、GTIOC0A 端子)を、パルス入力端子にしています。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。

```
startGPT0_2SouEncoder( 1, 7 , 1, 6);  // 2相エンコーダ 0A=P107(D7 端子) 0B=P106(D6 端子)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桁, GTIOCxB のポート上位１桁, GTIOCxB のポート下位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 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	●符号なしで読み込むとき 「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。 ●符号ありで読み込むとき 「INT_GPT0_CNT」の値を読み込むと、符号ありで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 の数字が入ります。

例	i = INT_GPT0_CNT; // GPT0 のカウント値(符号あり)を変数 i に代入
	
	j = GPT5_CNT; // GPT5 のカウント値(符号なし)を変数 j に代入
	※変数 j も unsigned int などの符号なしの型にする必要があります。

(4) カウント値のクリアの仕方

カウント値のクリアは、GPTx_CNT に代入してください。INT_GPTx_CNT に代入するとエラーになります。

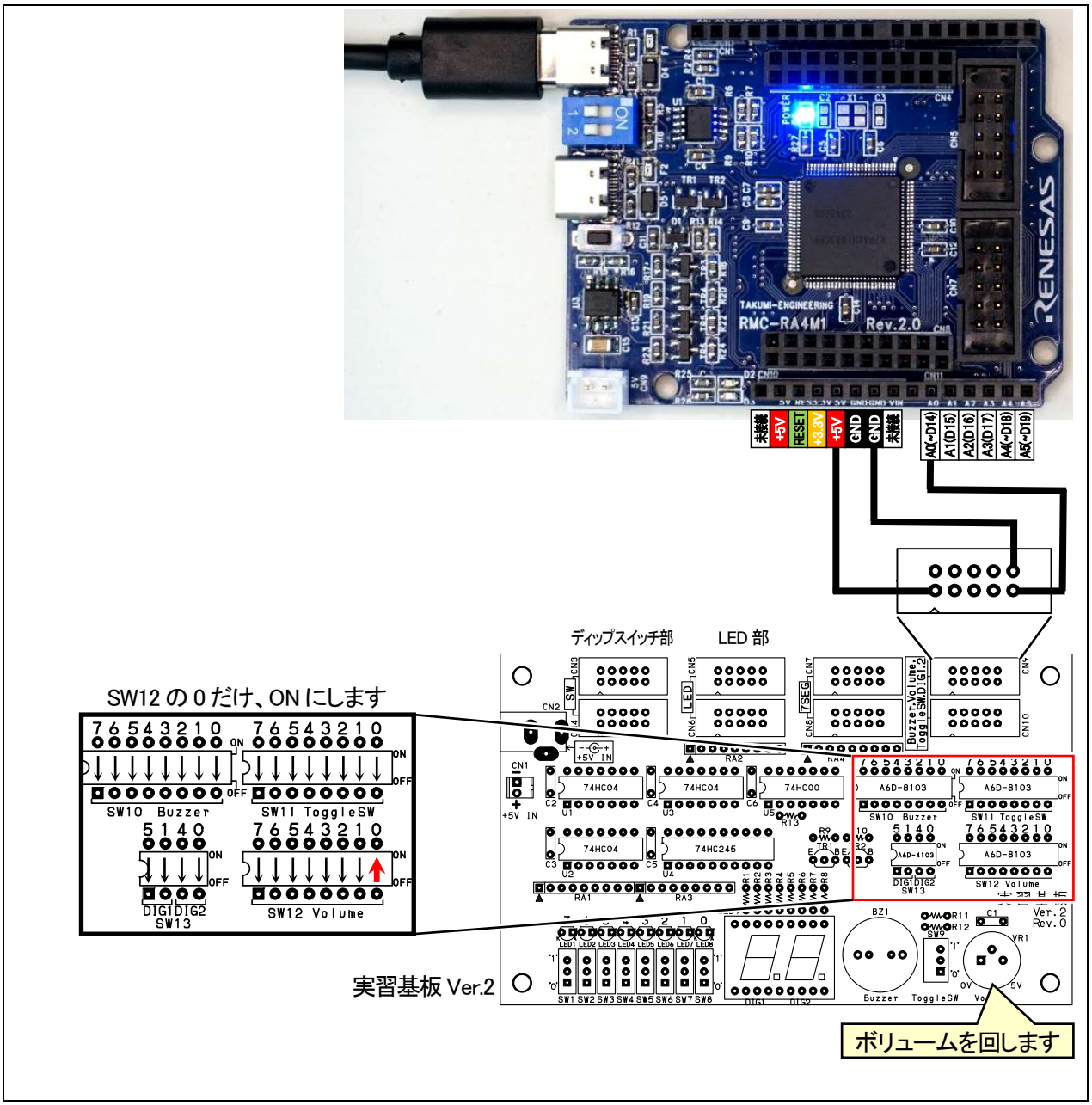
例	INT_GPT0_CNT = 0; // エラーになります！
	GPT0_CNT = 0; // 正しく実行できます。

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)(RMC-RA4M1 rev.2.0)
 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 :    while( !Serial ) {
13 :      // シリアル接続されるまで待つ
14 :    }
15 :  }
16 :
17 :  void loop() {
18 :    static int lp = 1;
19 :
20 :    Serial.print( lp );
21 :    Serial.print( " : " );
22 :    Serial.println( analogRead( A0 ) );  // A/D 変換結果をシリアル出力
23 :    delay( 1000 );
24 :    lp++;
25 :  }
```

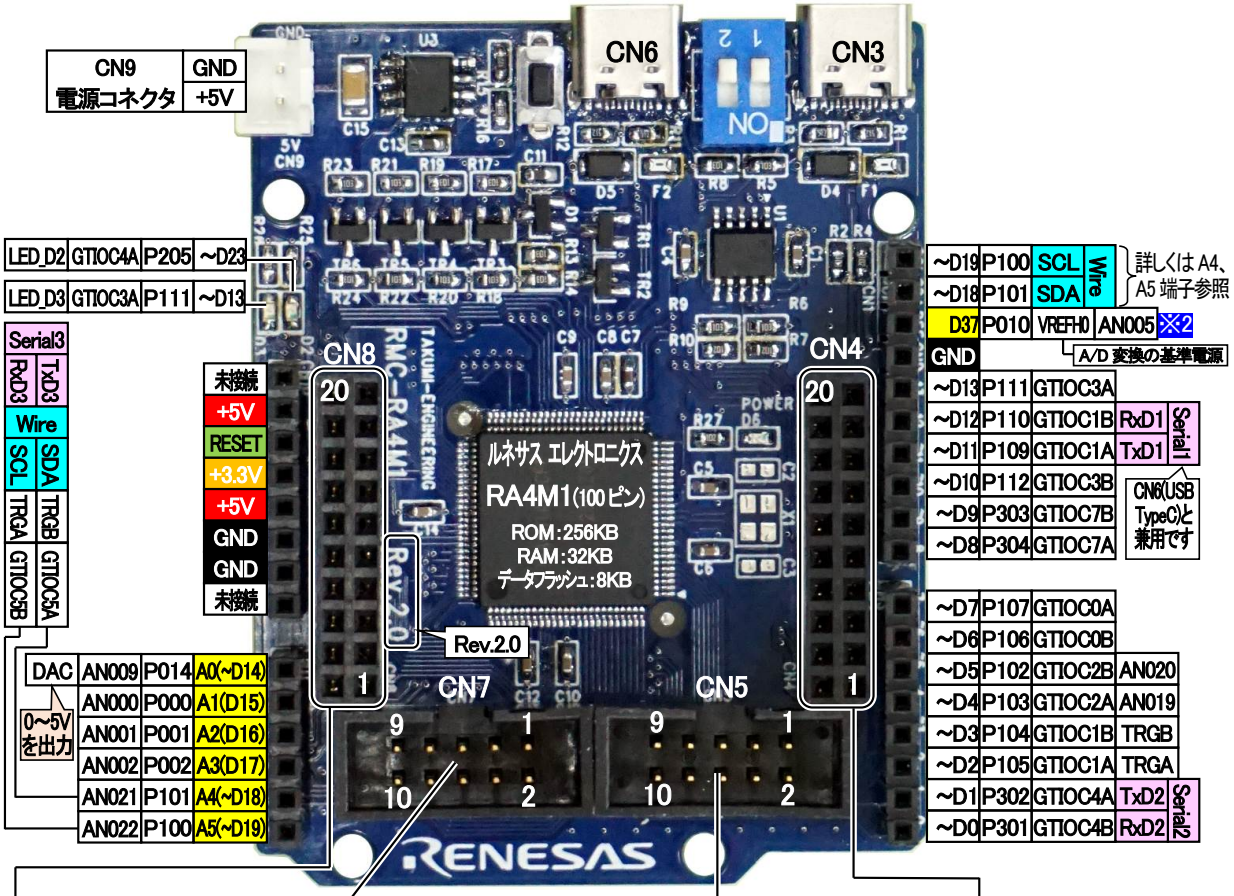
A/D 変換の分解能を設定します。
設定値は、8、10、12、14 のいずれかです。
analogReadResolution 関数を実行しないときの初期値は 10 です。

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。

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、シリアルモニタへ出力します。出力は、
1 : 12345
2 : 9876
などと、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 端子の A/D 値(0～16,383)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
P004	P006	P008	P011	
AN004	AN012	AN014	AN006	
GND	D41	D39	D37(※2)	D35
	P005	P007	P010	P012
	AN011	AN013	AN005	AN007

CN5　10ピンコネクタ(モータドライブ基板)

D34	D32	D30	D28	+5V
P003	P401	P403	P405	
AN003(プッシュSW)	TGRA/6B(左モPWM)	3A(右モPWM)	1A(LED3)	
GND	D33	D31	D29	D27
	P400	P404	P406	P402
	6A(左モータ方向)	3B(右モ方向)	1B(サーボ)	(LED2)

※2…D37は2端子あります

CN8　20ピンコネクタ

	(20ピン)	GND			
GTIOC2B	P114	D77	D76	P115	GTIOC4A
GTIOC4B	P608	D75	D74	P609	GTIOC5A
GTIOC5B	P610	D73	D72	P603	GTIOC7A
GTIOC7B	P602	D71	D70	P601	GTIOC6A
GTIOC6B	P600	D69	D68	P500	AN016/2A
AN017/2B	P501	D67	D66	P502	AN018/3B
AN023	P503	D65	D64	P504	AN024
AN025	P505	D63	D62	P015	AN010
AN008	P013	D61	+5V	(1ピン)	

CN4　20ピンコネクタ ※(P000)は入力専用端子

	(20ピン)	GND			
GTIOC0A	P300	D59	D58	P809	-
-	P808	D57	D56	(P200)	-
-	P206	D55	D54	P407	-
GTIOC5B	P408	D53	D52	P409	GTIOC5A
GTIOC6B	P410	D51	D50	P411	GTIOC6A
-	P412	D49	D48	P413	-
GTIOC0B	P414	D47	D46	P415	GTIOC0A
-	P708	D45	D44	(P214)	-
-	(P215)	D43	+5V	(1ピン)	

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。

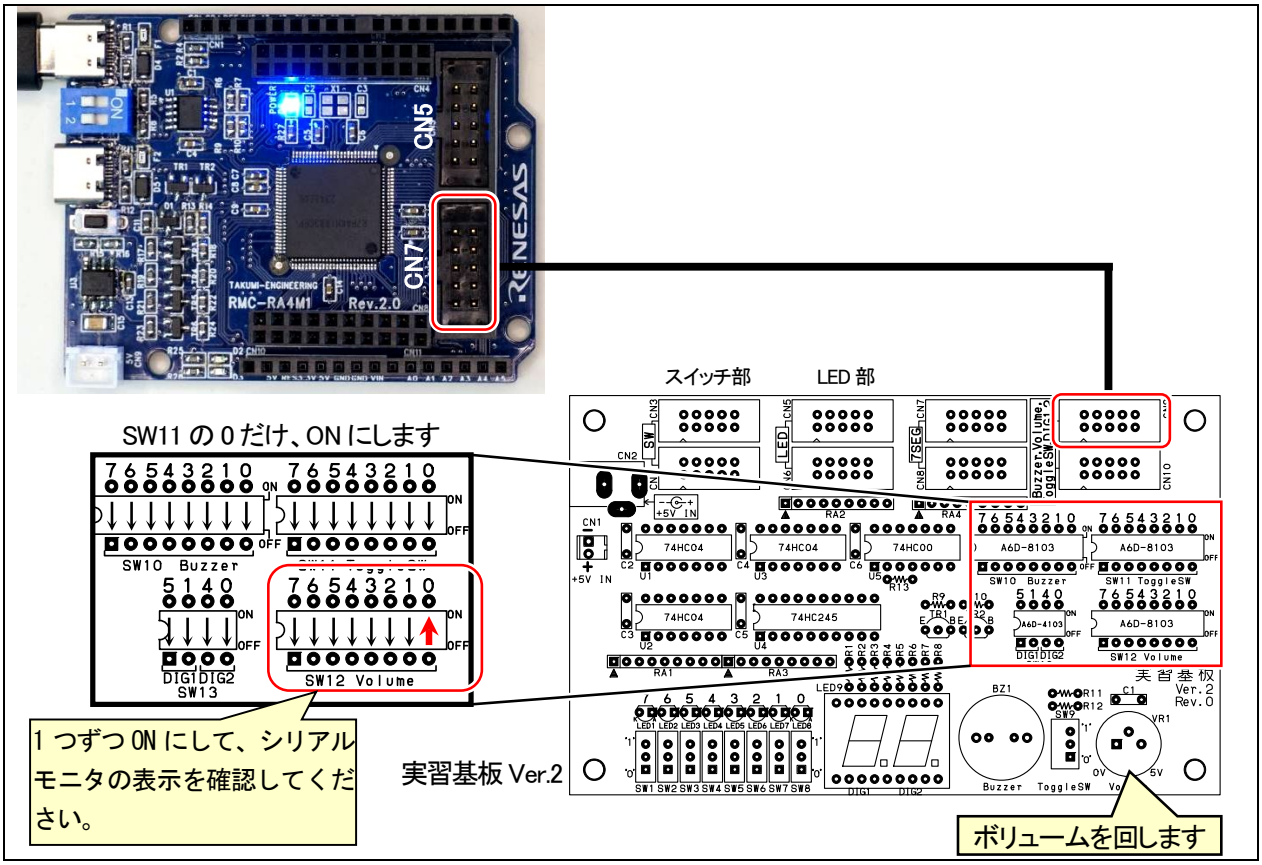
```
a = analogRead( D35 );
```

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(RMC-RA4M1 rev.2.0)
 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( 7 );  // D35 端子
16 :    ad.useCh( 6 );  // D36 端子
17 :    ad.useCh( 5 );  // D37 端子
18 :    ad.useCh( 14 ); // D38 端子
19 :    ad.useCh( 13 ); // D39 端子
20 :    ad.useCh( 12 ); // D40 端子
21 :    ad.useCh( 11 ); // D41 端子
22 :    ad.useCh( 4 );  // 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 :
36 :    sprintf( buff, "AN007=%5d  ", AD_007 );
37 :    Serial.print( buff );
38 :    sprintf( buff, "AN006=%5d  ", AD_006 );
39 :    Serial.print( buff );
40 :    sprintf( buff, "AN005=%5d  ", AD_005 );
41 :    Serial.print( buff );
42 :    sprintf( buff, "AN014=%5d  ", AD_014 );
43 :    Serial.print( buff );
44 :    sprintf( buff, "AN013=%5d  ", AD_013 );
45 :    Serial.print( buff );
46 :    sprintf( buff, "AN012=%5d  ", AD_012 );
47 :    Serial.print( buff );
48 :    sprintf( buff, "AN011=%5d  ", AD_011 );
49 :    Serial.print( buff );
50 :    sprintf( buff, "AN004=%5d  ", AD_004 );
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。7 = AN007 端子のことです。
今回は、AN007,AN006,AN005,AN014,AN013,
AN012,AN011,AN004 を設定します。
何個設定しても構いません。

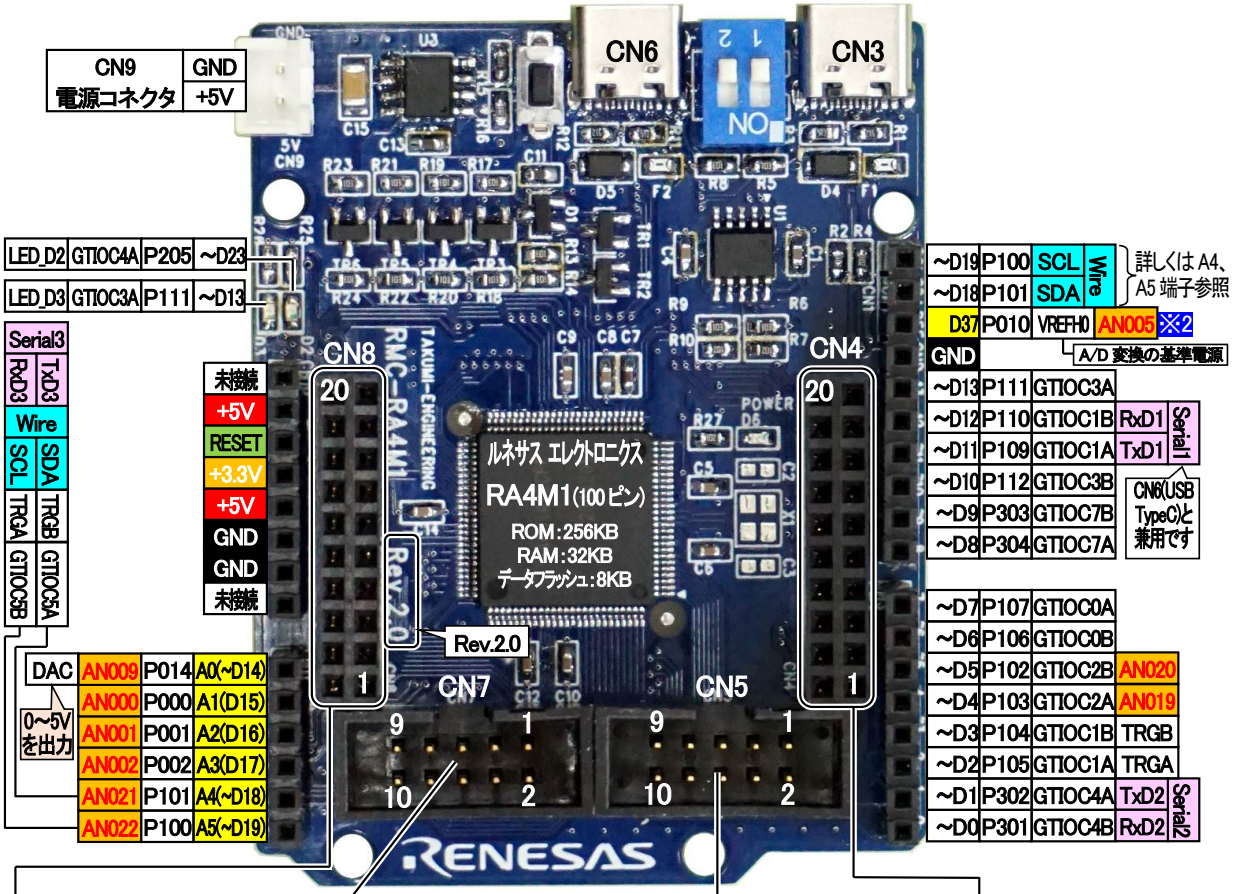
使用する端子を設定したのち、連続スキャンモードで A/D 変換をスタートします。

連続スキャンモードの A/D 値の読み込みは、
AD_xxx を使います。xxx は端子番号です。
AN007 端子なら、「AD_007」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(AN009 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
P004	P006	P008	P011	
AN004	AN012	AN014	AN006	
GND	D41	D39	D37(※2)	D35
	P005	P007	P010	P012
	AN011	AN013	AN005	AN007

CN5　10ピンコネクタ(モータドライブ基板)

D34	D32	D30	D28	+5V
P003	P401	P403	P405	
AN003(プッシュSW)	TGRA/6B(左モPWM)	3A(右モPWM)	1A(LED3)	
GND	D33	D31	D29	D27
	P400	P404	P406	P402
	6A(左モータ方向)	3B(右モ方向)	1B(サーボ)	(LED2)

※2…D37 は 2 端子あります

CN8　20 ピンコネクタ

	(20 ピン)	GND	D78	P113	GTIOC2A
GTIOC2B	P114	D77	D76	P115	GTIOC4A
GTIOC4B	P608	D75	D74	P609	GTIOC5A
GTIOC5B	P610	D73	D72	P603	GTIOC7A
GTIOC7B	P602	D71	D70	P601	GTIOC6A
GTIOC6B	P600	D69	D68	P500	AN016/2A
AN017/2B	P501	D67	D66	P502	AN018/3B
AN023	P503	D65	D64	P504	AN024
AN025	P505	D63	D62	P015	AN010
AN008	P013	D61	+5V	(1 ピン)	

CN4　20 ピンコネクタ　※(P000)は入力専用端子

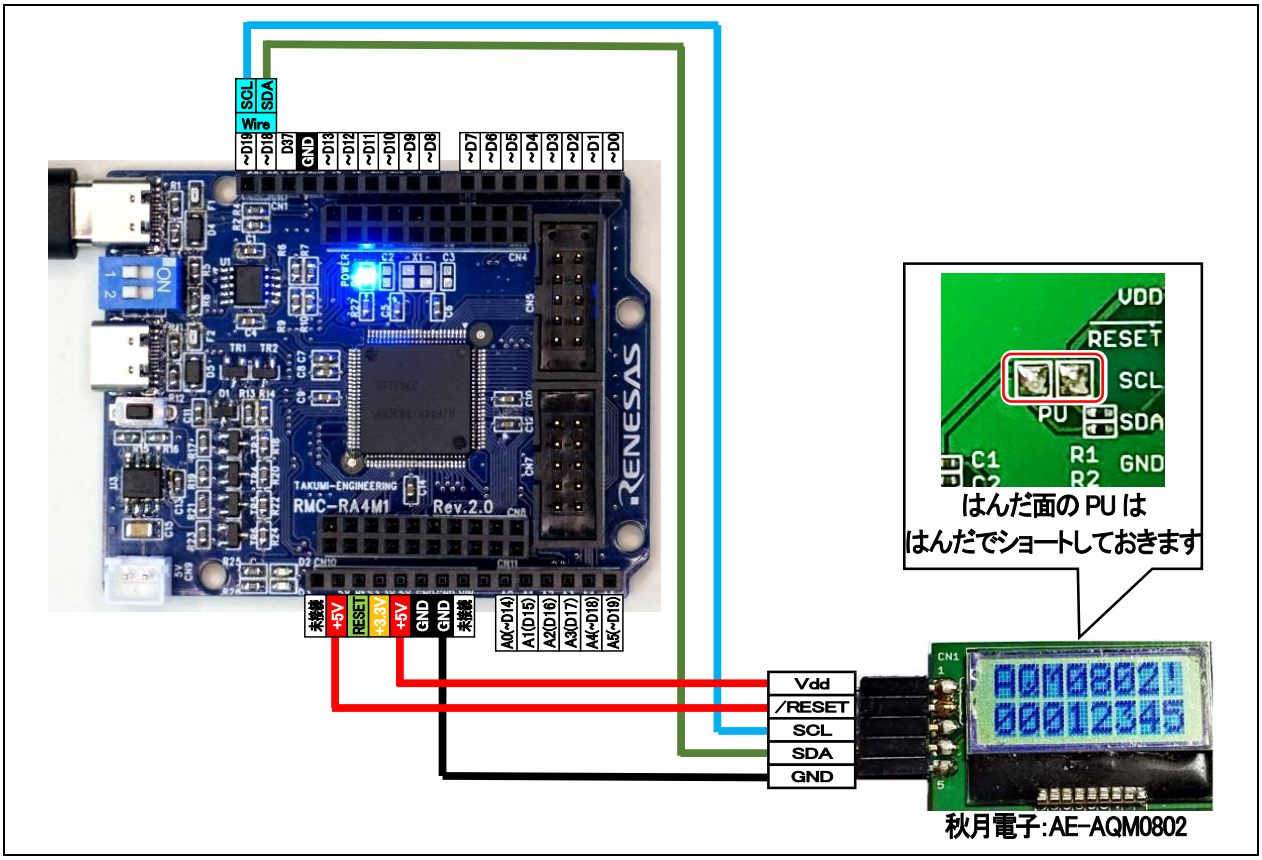
	(20 ピン)	GND	D60	P108	GTIOC0B
GTIOC0A	P300	D59	D58	P809	–
–	P808	D57	D56	(P200)	–
–	P206	D55	D54	P407	–
GTIOC5B	P408	D53	D52	P409	GTIOC5A
GTIOC6B	P410	D51	D50	P411	GTIOC6A
–	P412	D49	D48	P413	–
GTIOC0B	P414	D47	D46	P415	GTIOC0A
–	P708	D45	D44	(P214)	–
–	(P215)	D43	+5V	(1 ピン)	

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 SCL 端子・SDA 端子と、液晶の SCL 端子・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)(RMC-RA4M1 rev.2.0)
 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"  // SCL(A5)端子と SDA(A4)端子に接続
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( 10 );        // AQM0802 液晶使用開始 コントラスト=10
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 10 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 液晶 AQM0802 の表示 Wire1 を使用「aqm0802_wire1.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。前回の演習と同じ内容ですが、配線を Wire1 の D53 端子と D54 端子に接続します。

4.16.1. 配線

RMC-RA4M1 ボードの D53 端子と液晶の SCL 端子、D54 端子と液晶の SDA 端子を接続します。

4.16.2. プログラム

```
 1 :  //************************************************************************
 2 :  // ﾌｧｲﾙ内容      「aqm0802_wire1.ino」 AQM0802 液晶(I2C)(RMC-RA4M1 rev.2.0)
 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_wire1.h"  // SCL1(D53)端子と SDA1(D54)端子に接続
10 :
11 :  // グローバル変数宣言
12 :  MCR_AQM0802_WIRE1 lcd1;      // クラス MCR_AQM0802 でインスタンス(オブジェクト)lcd1 を生成
13 :
14 :  void setup() {
15 :    lcd1.begin( 10 );          // AQM0802 液晶使用開始 コントラスト=10
16 :  }
17 :
18 :  void loop() {
19 :    static int lp = 1;
20 :
21 :    lcd1.setPosition( 0,0 );
22 :    lcd1.printf( "AQM0802!" );
23 :    lcd1.setPosition( 0,1 );
24 :    lcd1.printf( "%08d", lp );
25 :    delay(10);
26 :    lp++;
27 :  }
```

マイコンボードの D53 端子を SCL、D54 端子を SDA とするライブラリを読み込みます。これらの端子は Wire1 の端子です。

MCR_AQM0802_WIRE1 クラスで lcd1 インスタンスを作成します。

4.16.3. プログラムの解説

Wire1 を使うと、CN3 に USB コネクタを挿しても COM として認識しなくなるため、シリアル通信や Arduino IDE からのプログラム書き込みができなくなるようです。

このとき、リセットスイッチをダブルクリック(連続でカチャ、カチャと押す)すると、COM ポートの代わりに USB DFU(Device Firm ware Upgrade)となり、書き込みができるようになります。

また、USB を接続しているときに液晶が点灯して、USB を抜いたときに液晶が動作しない場合、SCL、SDA 端子のプルアップ抵抗を 2kΩ程度にしてください。秋月電子製:AE-AQM0802 基板は、基板内蔵抵抗が 10kΩなので USB を抜くと、液晶が動作しなくなります。

4.17. 演習 17 I2C 液晶 AQM0802 の表示 ソフトウェア I2C を使用「aqm0802_soft_i2c.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。前回の演習と同じ内容ですが、ソフトウェア I2C を使い、入力専用端子以外であればどの端子でも接続可能です。

4.17.1. 配線

RMC-RA4M1 ボードの D59 端子と液晶の SCL 端子、D58 端子と液晶の SDA 端子を接続します。

4.17.2. プログラム

```
 1 :  //**************************************************************************
 2 :  // ﾌｧｲﾙ内容      「aqm0802_soft_i2c.ino」 AQM0802 液晶(I2C)ソフトウェア I2C 版 (RMC-RA4M1 rev.2.0)
 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_soft_i2c.h"  // ソフトウェア I2C
10 :
11 :  // グローバル変数宣言
12 :  MCR_AQM0802_SOFT_I2C lcd;           // クラス MCR_AQM0802_SOFT_I2C でインスタンス(オブジェクト)lcd を生成
13 :
14 :  void setup() {
15 :    lcd.begin( D58, D59, 10 );       // AQM0802 液晶使用開始 SDA 端子,SCL 端子,コントラスト=10 ※入力専用端子は使えません
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 :  }
```

SDA 端子、SCL 端子、コントラストを設定します。
端子は、入力専用端子は使用できません。
例えば D56(P200)は入力専用端子なので、使用できません。

4.17.3. プログラムの解説

ソフトウエア I2C は、入力専用端子以外であればどの端子でも使用可能ですが、プログラムの負荷が大きくなります(実行速度が遅くなります)。

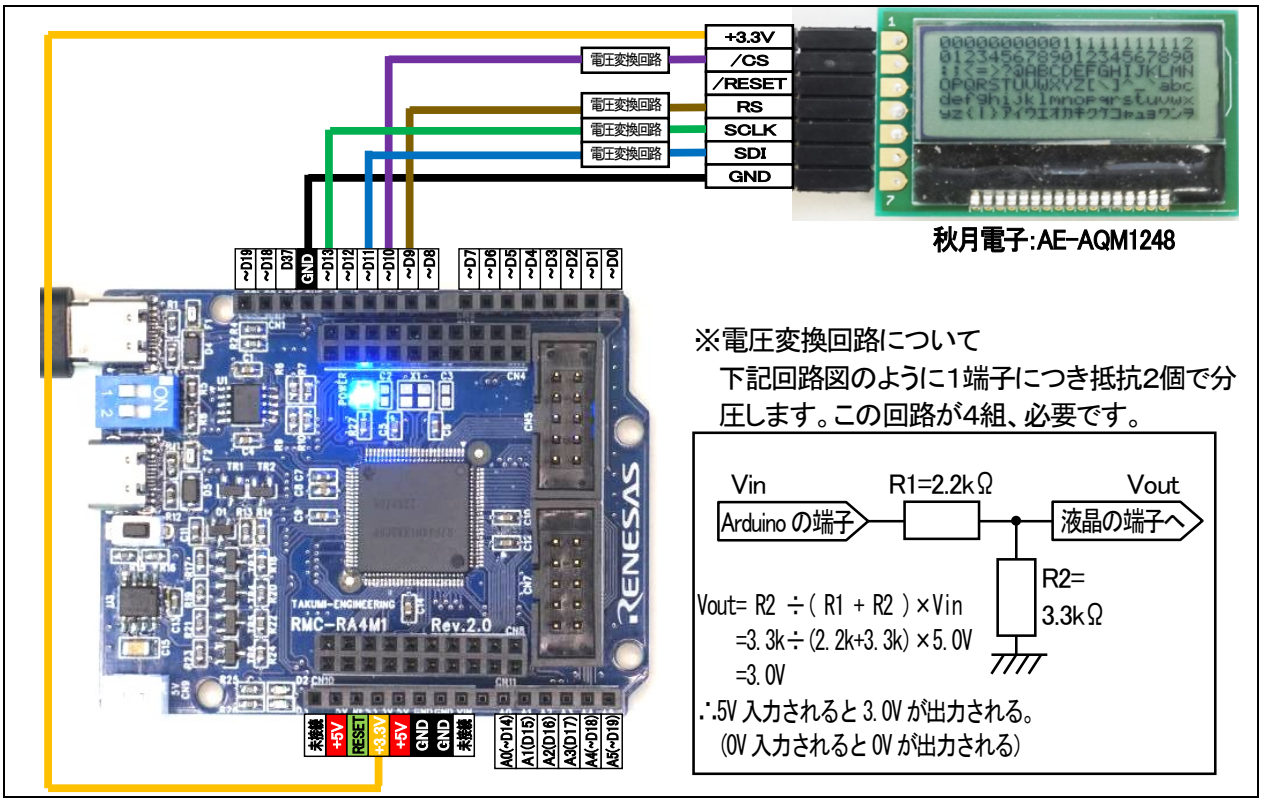
入力専用端子は、D56(P200)、D44(P214)、D43(P215)です。これらの端子は使用できません。

4.18. 演習 18　グラフィック液晶　AQM1248 の文字表示「aqm1248.ino」

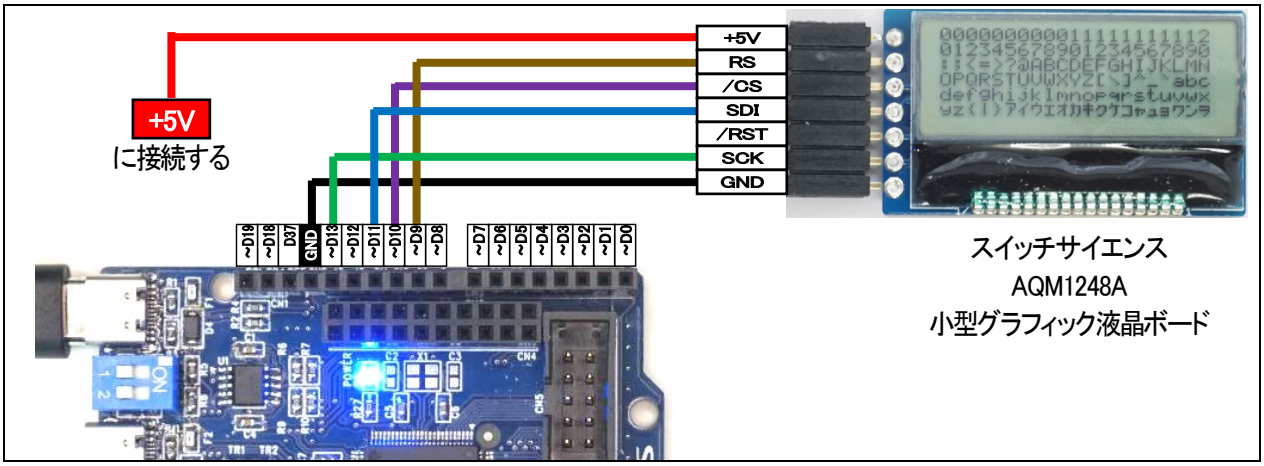
グラフィック液晶の AQM1248 液晶(スイッチサイエンス:AQM1248A 小型グラフィック液晶ボード、または秋月電子製:AE-AQM 1248 基板など)に文字を表示します。ただし、秋月電子製の基板は、3.3V で動作するため、端子に 5V を加えると壊れてしまいます。秋月電子製の基板は、電圧変換用抵抗が必要になります。スイッチサイエンス製の基板は 5V で動作するため(基板に電圧変換回路を搭載しています)、電圧変換用抵抗は必要ありません。
※秋月電子 AE-AQM1248 : https://akizukidenshi.com/catalog/g/gK-07007/
※スイッチサイエン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

4.18.1. 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。



4.18.2. プログラム

```
 1 :  //************************************************************************
 2 :  // ﾌｧｲﾙ内容       「aqm1248.ino」グラフィック液晶 AQM1248 の制御(RMC-RA4M1 rev.2.0)
 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 <MGLCD.h>                  // 出典；しなぷすのハード製作記  https://synapse.kyoto/lib/MGLCD/page001.html
10 :  #include <MGLCD_SPI.h>
11 :  #include <SPI.h>
12 :
13 :  // グローバル変数の宣言
14 :  ArduinoSPI SPI(MISO, MOSI, SCK, FORCE_SPI_MODE);          // Arduino UNO R4 の SPI を使用(MISO=12,MOSI=11,SCK=13)
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ﾇ" );        // 半角カタカナ表示
33 :  }
34 :
35 :  void loop()
36 :  {
37 :    // 特に何もしない
38 :  }
```

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。

RMC-RA4M1 ボードのマイコンには、SPI が 2 チャンネルあります。
Arduino UNO R4 互換の SPI(MISO=12,MOSI=11,SCK=13)を使用するときは、14 行目のように記述します。

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。

4.18.3. 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。

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

4.18.4. マイコン内蔵の SPI 機能を使用せずに、ソフトウェア SPI で動作させる

SPI (Serial Peripheral Interface) はクロックに同期させてデータの通信を行う同期式シリアル通信のひとつです。※出典：https://www.rohm.co.jp/electronics-basics/micon/mi_what8

マイコン内蔵 SPI 機能を用いると、高速に AQM1248 や microSD を制御することができます。RA マイコンには、SPI 機能が 2 チャネル内蔵されています。ただし、Arduino ライブラリが同時に使える SPI は 1 つです。microSD を使用するときは必ず SPI 機能が必要なため、さらに SPI 機能を使う AQM1248 制御はできません。

そこで、プログラムで擬似的に SPI と同等の動作をさせるソフトウェア SPI を使うことにより、microSD は SPI、AQM1248 はソフトウェア SPI というように、両方同時に使用することができます。マイコン内蔵 SPI とソフトウェア SPI の特徴を下表に示します。

内容	マイコン内蔵 SPI	ソフトウェア SPI
プログラムの大きさ	○ ソフトウェア SPI より少ない容量	△ 一般的にはマイコン内蔵 SPI を使ったときよりプログラム容量が大きくなる
速度	◎ 最速 MHz 単位の速度でデータを送れる	× とても遅い 数 kHz くらいでしかデータを送れない
端子	△ SPI が使える専用端子(D11,D12,D13 など)にしか接続できない	◎ プログラムでどの端子を使うか選択するので、どの端子でも使用可能

「しなぷすのハード製作記」には、ソフトウェア SPI で AQM1248 を制御するライブラリ、方法が公開されています。ソフトウェア SPI にプログラムを改造する部分を下記に示します。
出典：https://synapse.kyoto/hard/MGLCD_AQM1248A/page013.html

```
 9 :  #include <MGLCD.h>      // 出典；しなぷすのハード製作記  https://synapse.kyoto/lib/MGLCD/page001.html
10 :  // #include <MGLCD_SPI.h>  コメントにする
11 :  // #include <SPI.h>        コメントにする
12 :
13 :  // グローバル変数の宣言
14 :  // ArduinoSPI SPI(MISO, MOSI, SCK, FORCE_SPI_MODE);                 コメントにする
15 :  // MGLCD_AQM1248A_SPI MGLCD( MGLCD_SpiPin2(10, 9), 100000L );  コメントにする
16 :  MGLCD_AQM1248A_SoftwareSPI MGLCD(MGLCD_SpiPin4( 13, 11, 10, 9) , 0 );  追加
                                                    ①  ②  ③  ④  ⑤
                                                  SCK 端子  MOSI 端子  CS 端子  DI 端子  ウエイト
```

4.19. 演習 19　グラフィック液晶　AQM1248 のグラフィック表示「aqm1248graphic.ino」

演習 17 の AQM1248 液晶に、グラフィック(点、線、四角、円)を表示します。
演習 17 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

4.19.1. 配線

演習 17 と同じです。

4.19.2. プログラム

```
 1 ： //*****************************************************************************
 2 ： // ﾌｧｲﾙ内容      「aqm1248graphic.ino」グラフィック液晶 AQM1248 の制御(グラフィック) (RMC-RA4M1 rev.2.0)
 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 <MGLCD.h>              // 出典；しなぷすのハード製作記 https://synapse.kyoto/lib/MGLCD/page001.html
10 ： #include <MGLCD_SPI.h>
11 ： #include <SPI.h>
12 ：
13 ： // グローバル変数の宣言
14 ： ArduinoSPI SPI(MISO, MOSI, SCK, FORCE_SPI_MODE);            // Arduino UNO R4 の SPI を使用(MISO=12,MOSI=11,SCK=13)
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

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。

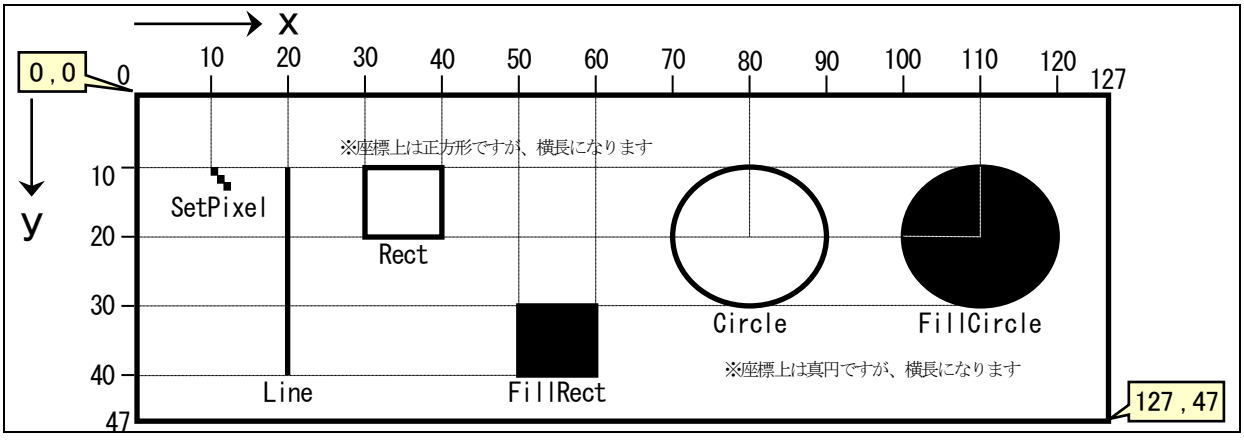
※真円しか描けませんが、横：縦の比率が 1:0.864 なので横長の楕円になってしまいます

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

4.19.3. 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。



4.20. 演習 20　EEP-ROM(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んだり、読み込んだりします。

EEP-ROM は、プログラムで内容を書き換え可能な ROM のことで、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。※プログラムが入っているメモリを、コードフラッシュメモリといいます。コードフラッシュメモリはパソコンからは書き換えることができますが、プログラムから内容の書き換えはできません。

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(読み込み回数は制限がありません)。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。

4.20.1. 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。

4.20.2. 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(RMC-RA4M1 rev.2.0)
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 :
26 :    EEPROM.get( 0, i ); // 0～3 番地から値読み込み
27 :
28 :    if( i != EEPROM_CHECK ) {
29 :      //もしデータ書き込まれていないなら
30 :      Serial.print( "初めての使用と判断し、EEP-ROM を初期化します...  " );
31 :      EEPROM.put( 0 , EEPROM_CHECK ); // 0～3 番地　チェック用データ　書き込み
32 :      EEPROM.put( 4 , (int)1 );       // 4～7 番地　データ　書き込み
33 :      Serial.println( "初期化しました。" );
34 :    }
```

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、使う番地に初期値を代入します。

0～3 番地に 0x20240201 が書き込まれていないときの処理です。

```
35 :    // 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
36 :    EEPROM.get( 4 , i );
37 :    Serial.print( "4番地の値は " );
38 :    Serial.print( i );
39 :    Serial.println( " です。" );
40 :
41 :    i++;
42 :    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
43 :  }
44 :
45 :  void loop() {
46 :    // 何もしない
47 :  }
```

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。
次に読み込まれる値は、＋1した値になります。

4.20.3. 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「5.1. 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 型の値を変数 i に読み込む
    EEPROM.get( 4 , f );   // 4～7 番地の float 型の値を変数 f に読み込む
    EEPROM.get( 8 , c );   // 8 番地の char 型の値を変数 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); //初期値 0xffffffff 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); //初期値 0xffffffff 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); //初期値 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); //初期値 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. 付録

5.1. 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	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 の int のサイズと、R8C/38A や Arduino UNO R3（AVR）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。

5.2. int、long、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**	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)。

6. 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

・ルネサス半導体トレーニングセンター　C言語入門コーステキスト　第 1 版
・電波新聞社　マイコン入門講座　大須賀威彦著　第 1 版
・ソフトバンク(株)　新C言語入門シニア編　林晴比古著　初版
・共立出版(株)　プログラマのための ANSI C 全書　L.Ammeraal 著 吉田敬一・竹内淑子・吉田恵美子訳　初版

マイコンカーラリー、販売部品、各種マニュアルについての詳しい情報は、マイコンカーラリー販売サイトをご覧ください。

https://www2.himdx.net/mcr/

RA マイコンについての詳しい情報は、ルネサス エレクトロニクスのホームページをご覧ください。

https://www.renesas.com/

JMCR2015 以前の大会記録や、JMCR2016 以降のマイコンカーの写真や情報など掲載されているホームページです。

https://j-mcr.net/

改訂履歴

1.00	作成
1.01	誤字・脱字の修正
1.10	ハードディスク損傷で最新データが消失し、古いデータから再度作り直し