

ルネサスエレクトロニクス製

**R8C/38A マイコン**

**R8C/35A マイコン**

**制御ライブラリ**

**解説マニュアル**

※R8C/35A について

本マニュアルでは、主に R8C/38A マイコンを使った場合について説明していますが、R8C/35A もほぼ同じです。マニュアル内の「38」は、「35」と読み替えて、進めてください。

第 1.04 版

2018.03.13

株式会社日立ドキュメントソリューションズ

# 注意事項 (rev.6.0H)

## 著作権

- ・本マニュアルに関する著作権は株式会社日立ドキュメントソリューションズに帰属します。
- ・本マニュアルは著作権法および、国際著作権条約により保護されています。

## 禁止事項

ユーザーは以下の内容を行うことはできません。

- ・第三者に対して、本マニュアルを販売、販売を目的とした宣伝、使用、営業、複製などを行うこと
- ・第三者に対して、本マニュアルの使用権を譲渡または再承諾すること
- ・本マニュアルの一部または全部を改変、除去すること
- ・本マニュアルを無許可で翻訳すること
- ・本マニュアルの内容を使用しての、人命や人体に危害を及ぼす恐れのある用途での使用

## 転載、複製

本マニュアルの転載、複製については、文書による株式会社日立ドキュメントソリューションズの事前の承諾が必要です。

## 責任の制限

本マニュアルに記載した情報は、正確を期すため、慎重に制作したのですが万一本マニュアルの記述誤りに起因する損害が生じた場合でも、株式会社日立ドキュメントソリューションズはその責任を負いません。

## その他

- ・本マニュアルに記載の情報は本マニュアル発行時点のものであり、株式会社日立ドキュメントソリューションズは、予告なしに、本マニュアルに記載した情報または仕様を変更することがあります。製作に当たりましては、最新の内容を確認いただきますようお願いします。
- ・すべての商標および登録商標は、それぞれの所有者に帰属します。

## 連絡先

株式会社 日立ドキュメントソリューションズ

〒135-0016 東京都江東区東陽六丁目 3 番 2 号 イースト 21 タワー

E-mail:himdx.m-carrally.dd@hitachi.com

# 目次

1. 概要.....	1
2. サンプルプログラム.....	2
2.1 開発環境.....	2
2.2 ワークスペース(プログラム)を開く.....	3
2.3 ファイル構成.....	4
2.4 サンプルプログラム.....	5
2.4.1 サンプルプログラムの内容.....	5
2.4.2 実行の仕方.....	6
2.4.3 sample1.c を実行する.....	8
2.4.4 sample2.c を実行する.....	9
2.4.5 sample3.c を実行する.....	10
2.4.6 sample4.c を実行する.....	11
2.4.1 sample5.c を実行する.....	12
2.4.2 sample6.c を実行する.....	13
2.4.3 sample7.c を実行する.....	14
2.4.4 sample8.c を実行する.....	14
2.5 ライブラリを使った環境の構築方法.....	15
3. ライブラリ関数.....	21
3.1 クロックに関する関数.....	21
3.1.1 外付けクリスタル値のセット.....	21
3.1.2 外付けクリスタルに切り替え.....	21
3.2 ポートに関する関数.....	22
3.2.1 ポートの入出力設定.....	22
3.2.2 ポートにデータ出力(ポート単位で出力).....	22
3.2.3 ポートからデータ入力(ポート単位で入力).....	22
3.2.4 端子にデータ出力(1bit ごとに出力).....	23
3.2.5 端子からデータ入力(1bit ごとに入力).....	23
3.2.6 端子のプルアップ制御.....	23
3.3 A/D 変換に関する関数.....	24
3.3.1 A/D 変換.....	24
3.4 タイマに関する関数.....	24
3.4.1 タイマ( $\mu$ s 単位).....	24
3.4.2 タイマ(ms 単位).....	25
3.5 マイコンの動作に関する関数.....	25
3.5.1 ストップ.....	25
3.5.2 全体割り込みの許可.....	25
3.5.3 全体割り込みの禁止.....	26
3.6 タイマ RB に関する関数.....	26
3.6.1 タイマ RB 設定(インターバル割り込み).....	26
3.7 タイマ RF に関する関数(R8C/35A は未対応です).....	27
3.7.1 タイマ RF 設定(インターバル割り込み).....	27
3.8 タイマ RG に関する関数(R8C/35A は未対応です).....	28
3.8.1 タイマ RG 設定(インターバル割り込み).....	28
3.9 タイマ RC に関する関数.....	29

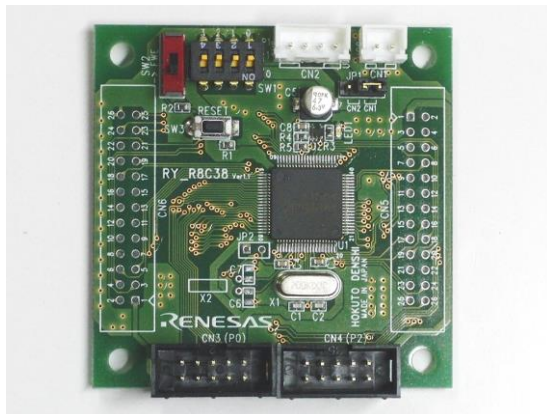
## 目次

3.9.1 タイマ RC 設定(周波数出力) .....	29
3.9.2 周波数の設定(周波数出力モード時) .....	29
3.10 タイマ RD に関する関数 .....	30
3.10.1 タイマ RD 設定(PWM 出力) .....	30
3.10.2 PWM 波形出力 .....	31
3.10.3 PWM 波形の出力状態取得 .....	31
3.11 マイコンボードの動作に関する関数 .....	31
3.11.1 LED 点灯 .....	31
3.11.2 ディップスイッチ値取得 .....	31
3.12 液晶に関する関数 .....	32
3.12.1 液晶初期化 .....	32
3.12.2 液晶に表示する位置の指定 .....	33
3.12.3 printf 文と同じ書式で液晶に文字を表示 .....	33
3.12.4 液晶に文字を表示 .....	33
3.12.5 液晶に 10 進数を表示 .....	34
3.12.6 液晶に 16 進数を表示 .....	34
3.13 printf 文、scanf 文に関する関数 .....	34
3.13.1 printf 文、scanf 文を使う初期設定 .....	34
<b>4. 参考文献 .....</b>	<b>35</b>

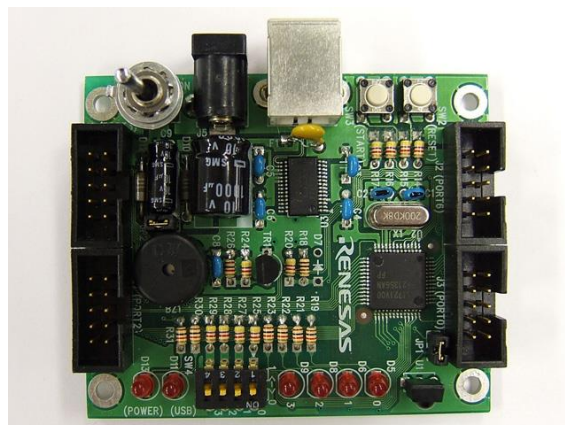
1. 概要

# 1. 概要

本マニュアルは、R8C/38A マイコン、または R8C/35A マイコンの制御ライブラリの内容、使用例を解説しています。制御ライブラリは、R8C/38A マイコン、または R8C/35A マイコンの内蔵周辺機能の設定をひとまとめにして関数として用意したものです。



▲RY\_R8C38 ボード(マイコンはルネサス エレクトロニクス製の R8C/38A)



▲RMC-R8C35A ボード(マイコンはルネサス エレクトロニクス製の R8C/35A)

本マニュアルでは、ジャパンマイコンカーラーの承認ボードである「RY\_R8C38 ボード」、および「RMC-R8C35A ボード」を使った場合について説明しています。特徴を下記に示します。

本マニュアルで説明するマイコンボード	マイコン	仕様やサンプルプログラムについて	購入先
RY_R8C38	ルネサスエレクトロニクス製 R8C/38A	<a href="http://www2.himdx.net/mcr/product/download.html">http://www2.himdx.net/mcr/product/download.html</a> ↓ 各種基板に関する資料 ↓ マイコン実習マニュアル(R8C/38A 版)	マイコンカーラー販売サイト <a href="http://www2.himdx.net/mcr/">http://www2.himdx.net/mcr/</a>
RMC-R8C35A	ルネサスエレクトロニクス製 R8C/35A	<a href="http://www2.himdx.net/mcr/product/download.html">http://www2.himdx.net/mcr/product/download.html</a> ↓ ミニマイコンカーVer.2 に関する資料 ↓ RMC-R8C35A ボード ↓ ミニマイコンカー製作キット Ver.2 マイコン実習マニュアル R8C/35A 版	マイコンカーラー販売サイト <a href="http://www2.himdx.net/mcr/">http://www2.himdx.net/mcr/</a>

## 2. サンプルプログラム

### 2.1 開発環境

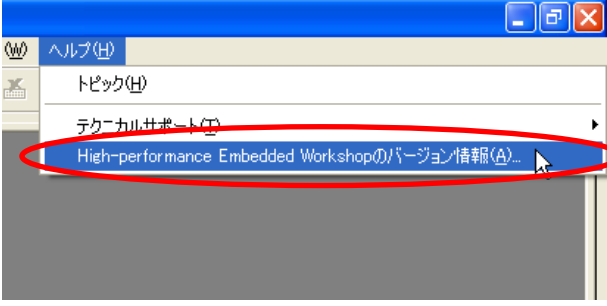
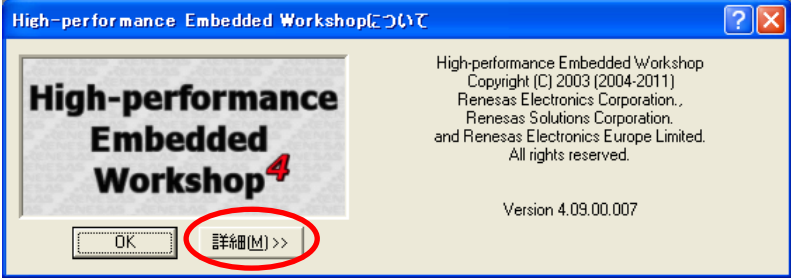
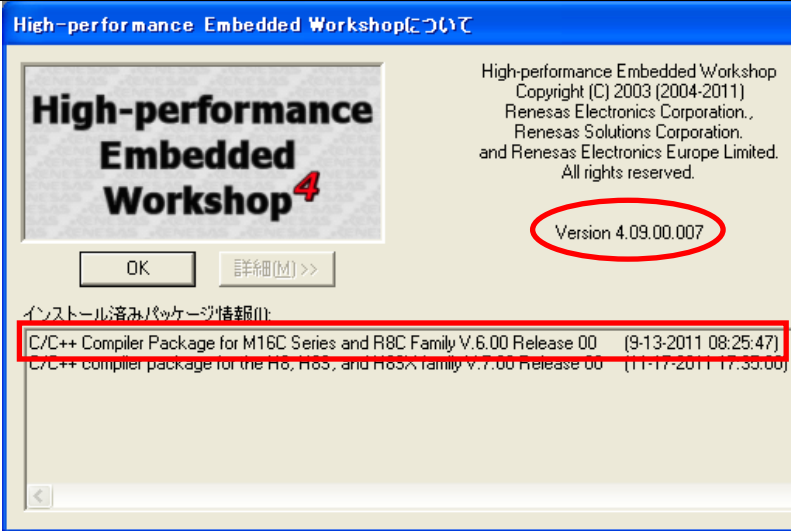
本マニュアルでは、ルネサス統合開発環境(無償評価版)を使用します。ルネサス統合開発環境やその他ファイルの入手、インストール、操作方法については、マイコンカーラーサイトにある「ルネサス統合開発環境 操作マニュアル(R8C/38A 版)」を参照してください。

ルネサス統合開発環境 操作マニュアル(R8C/38A 版)は、  
<http://www2.himdx.net/mcr/product/download.html#ide>



R8C/38A マイコン(RY\_R8C38 ボード)に関する資料より、ダウンロードできます。

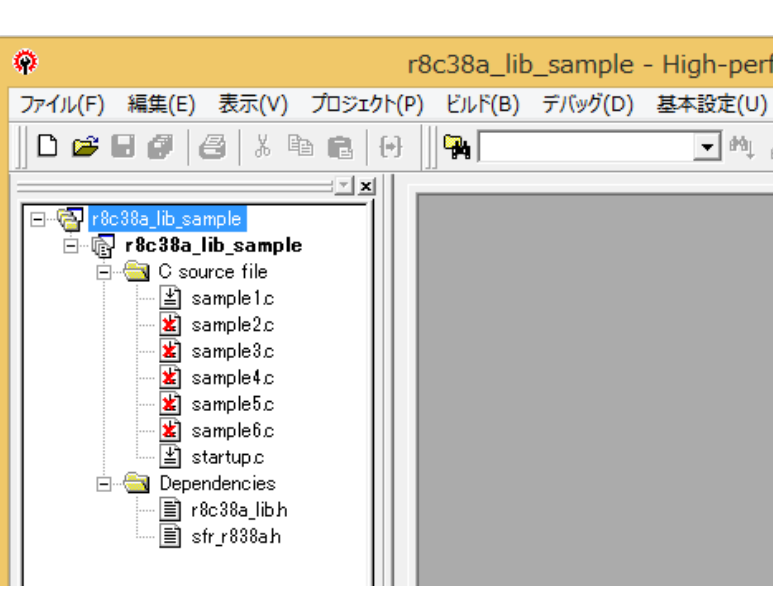
ルネサス統合開発環境のバージョンが古いと、サンプルプログラムが開けません。既にルネサス統合開発環境がインストールされている場合は、次の方法で確認してください。

1		<p>「ヘルプ→バージョン情報」をクリックします。</p>
2		<p>「詳細&gt;&gt;」をクリックします。</p>
3		<ul style="list-style-type: none"> <li>•Version が 4.09.00.007 より小さい場合</li> <li>•C/C++ Compiler Package for M16C Series and R8C Family のバージョンが、V.6.00 Release 00 より小さい場合は、最新版をダウンロード、インストールしてください。</li> </ul> <p>H8 や RX など、他のマイコンのコンパイラパッケージが入っていても問題ありません。</p>

## 2.2 ワークスペース(プログラム)を開く

ルネサス統合開発環境でのファイルの開き方、操作方法を説明します。

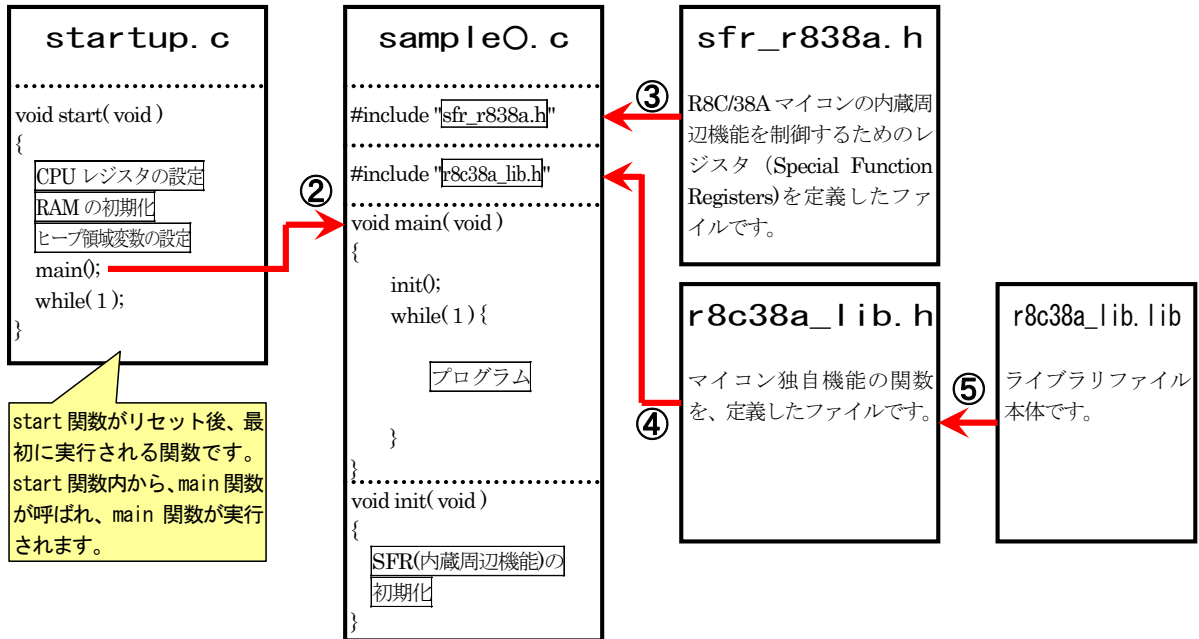
1		<p>●<b>R8C/38A の場合</b> 「C ドライブ → workspace → r8c38a_lib_sample」フォルダにある、「r8c38a_lib_sample.hws」を実行します。</p> <p>●<b>R8C/35A の場合</b> 「C ドライブ → workspace → r8c35a_lib_sample」フォルダにある、「r8c35a_lib_sample.hws」を実行します。</p>
---	--	---

2		<p>ルネサス統合開発環境が立ち上がります。 左側にあるリストが、プログラムファイルになります。 詳しい操作方法は、「ルネサス統合開発環境 操作マニュアル(R8C/38A 版)」を参照してください。</p> <p>※hws ファイルを開けないというメッセージがでた場合は、ルネサス統合開発環境の最新版をルネサスエレクトロニクスのホームページからダウンロードして、インストールしてください。</p>
---	---	--

## 2. サンプルプログラム

## 2.3 ファイル構成

今回のプロジェクトのファイル構成を下図に示します。



プログラムの動きを、下記に示します。

①	マイコンの電源が入ると、start 関数が実行されます。start 関数では、CPU レジスタの設定など、マイコンを動かすための設定を行います。
②	①が終わると、main 関数を実行します。
③	sample0.c は、sfr_r838a.h ファイルをインクルードしてファイルを取り込みます。このファイルは、R8C/38A マイコンの内蔵周辺機能を制御するためのレジスタ (Special Function Registers)を定義したファイルです。
④	sample0.c は、r8c38a_lib.h ファイルをインクルードしてファイルを取り込みます。このファイルは、内蔵周辺機能を設定するための関数などを定義したファイルです。
⑤	ライブラリの関数が呼び出されると、ルネサス統合開発環境が自動で呼び出します。r8c38a_lib.lib はツールチェーンで登録します。登録方法は後述します。



## 2.4 サンプルプログラム

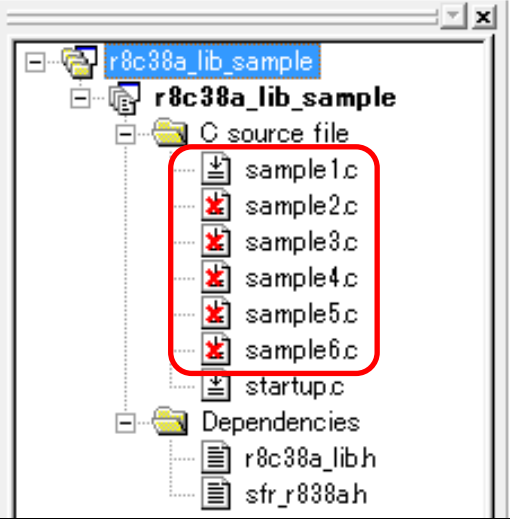

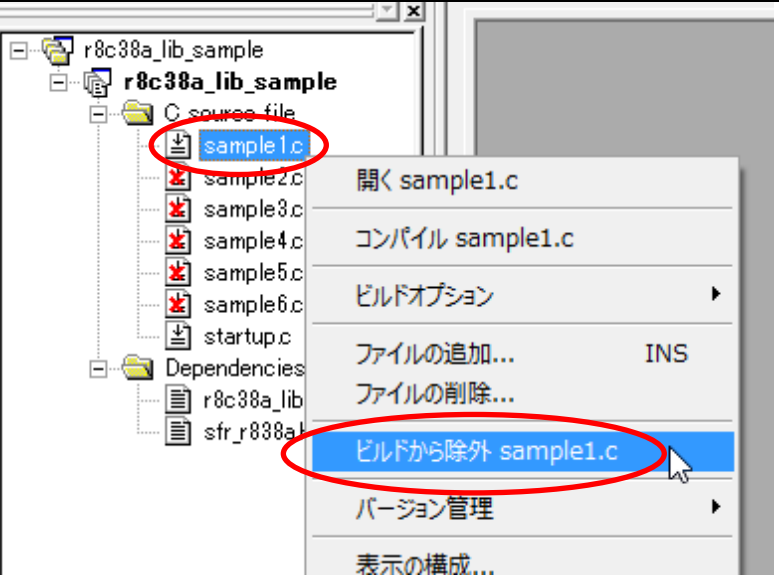
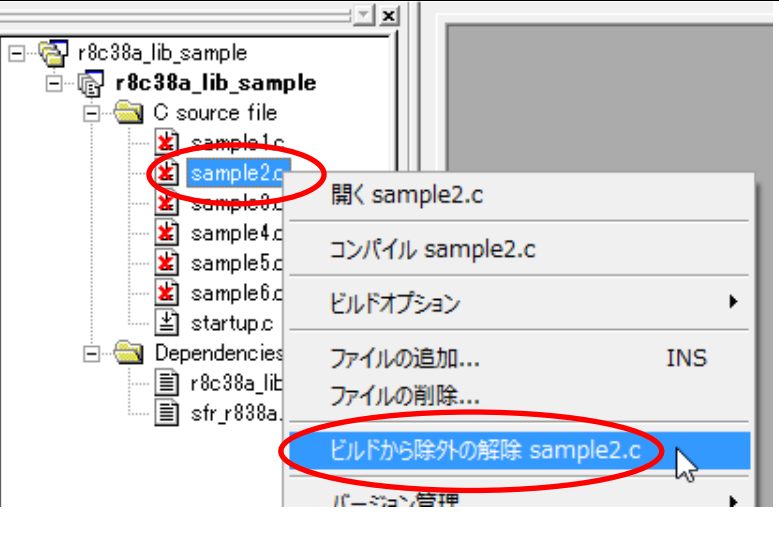
### 2.4.1 サンプルプログラムの内容

サンプルプログラムの内容を下記に示します。

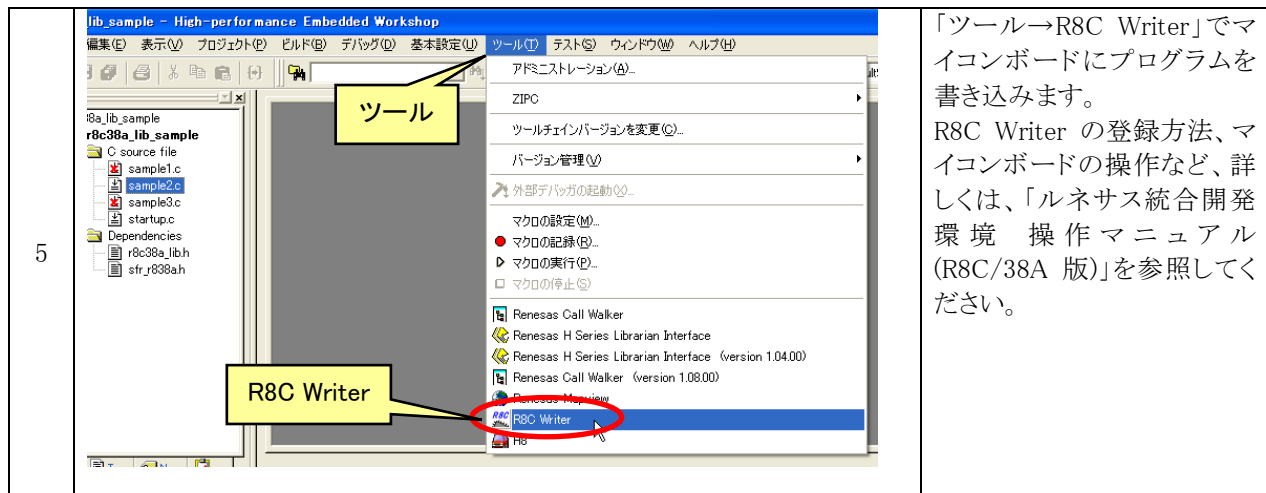
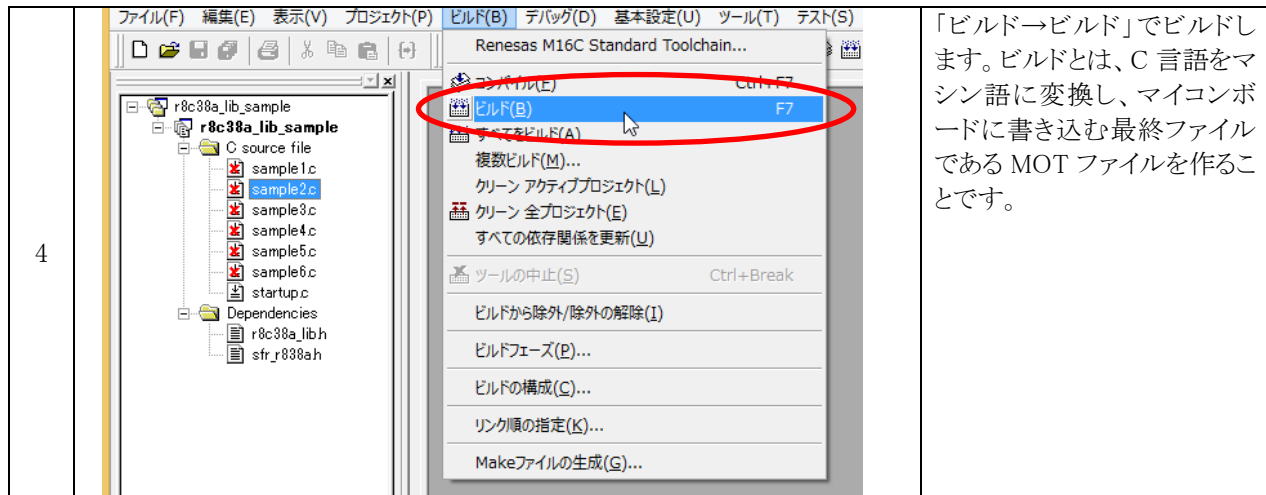
プログラム名	割り込み	R8C 38A 対応	R8C 35A 対応	内容
sample1.c	未使用	○	○	マイコンボードの LED を点滅させるプログラムです。割り込みを使いません。
sample2.c	使用	○	○	マイコンボードのディップスイッチの値を、ポート 2 へ出力するプログラムです。 割り込みで、マイコンボードの LED を点滅させます。
sample3.c	使用	○	○	液晶を制御するプログラムです。 割り込みで、マイコンボードの LED を点滅させます。
sample4.c	使用	○	×	sample2.c と同様のプログラムですが、sample2.c とは違う割り込みを使用しています。
sample5.c	使用	○	×	sample2.c や sample4.c と同様のプログラムですが、sample2.c や sample4.c とは違う割り込みを使用しています。
sample6.c	未使用	○	×	指定した周波数を出力するプログラムです。 端子は 19 端子から選ぶことができます。 ※同時に出力はできません。19 端子中、1 端子から選ぶことができます。
sample7.c	未使用	○	○	PWM 波形を出力するプログラムです。 2 組 3 波形、合計 6 波形を出力することができます。 チャンネル 0 は p2_1 端子、p2_2 端子、p2_3 端子から PWM 波形を出力できます。周期は共通で、各端子から任意のデューティ比の波形を出力することができます。 チャンネル 1 が p2_5 端子、p2_6 端子、p2_7 端子です。周期、デューティ比はチャンネル 0 と同様です。 例えば、16ms 周期の波形をチャンネル 0 に設定して p2_1 端子、p2_2 端子、p2_3 端子からそれぞれ ON 幅、70%、50%、30%の波形を出力、1ms 周期の波形をチャンネル 1 に設定して p2_5 端子、p2_6 端子、p2_7 端子からそれぞれ ON 幅、80%、60%、40%の波形を出力などできます。
sample8.c	未使用	○	○	printf 文を使ったプログラムです。

## 2. サンプルプログラム

## 2.4.2 実行の仕方

1		<p>sample1.c～sample6.c で、有効にできるファイルは 1 つだけです。ファイル名の左側にあるアイコンマークに× (  ) がある場合は、無効です。左画面は、sample1.c が有効な状態です。例えば今回は、sample2.c を有効にしたいと思います。</p>
2		<p>「sample1.c」で右クリックして、「ビルドから除外」を選択します。</p>
3		<p>「sample2.c」で右クリックして、「ビルドから除外の解除」を選択します。</p>

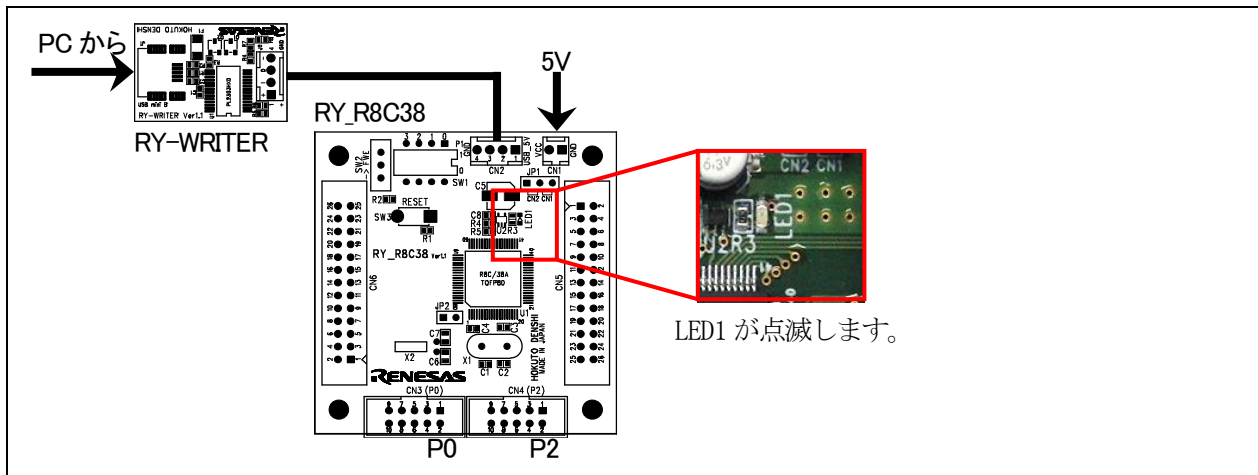
## 2. サンプルプログラム



2. サンプルプログラム

2.4.3 sample1.c を実行する

接続例を下記に示します。

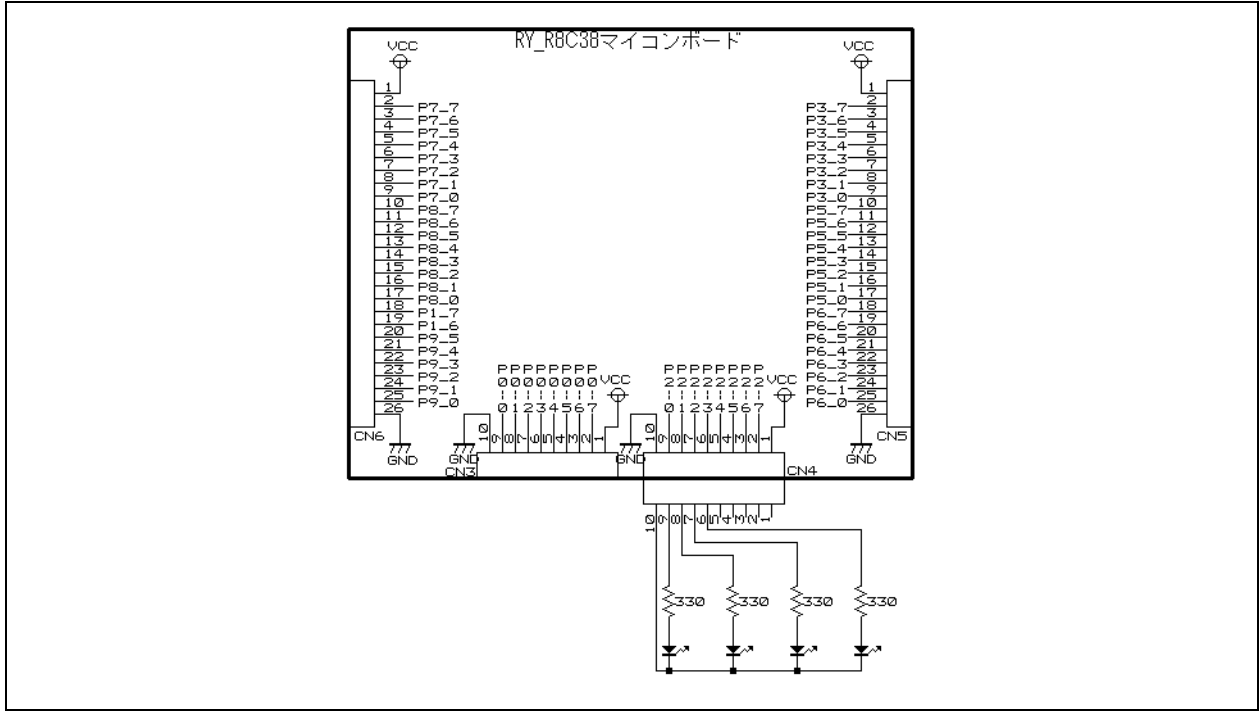


sample1.c を有効にして、ビルド、書き込みをして、実行してください。

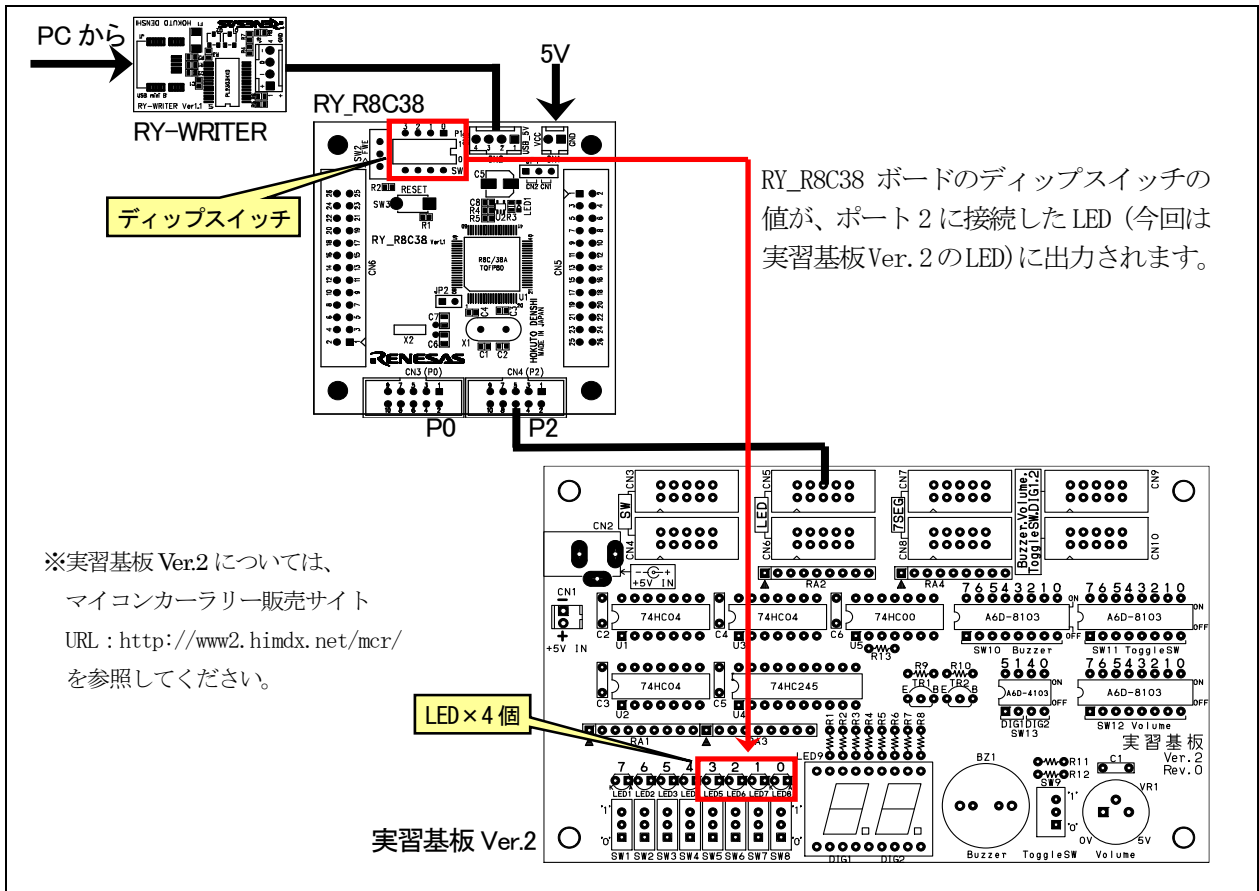
2. サンプルプログラム

2.4.4 sample2.c を実行する

回路図を下記に示します。P2\_3~P2\_0 に LED を接続します。



接続例を下記に示します。実習基板 Ver.2 を使うと、実習基板 Ver.2 の LED 部分と RY\_R8C38 ボードの CN4 (ポート 2) を直結できます。

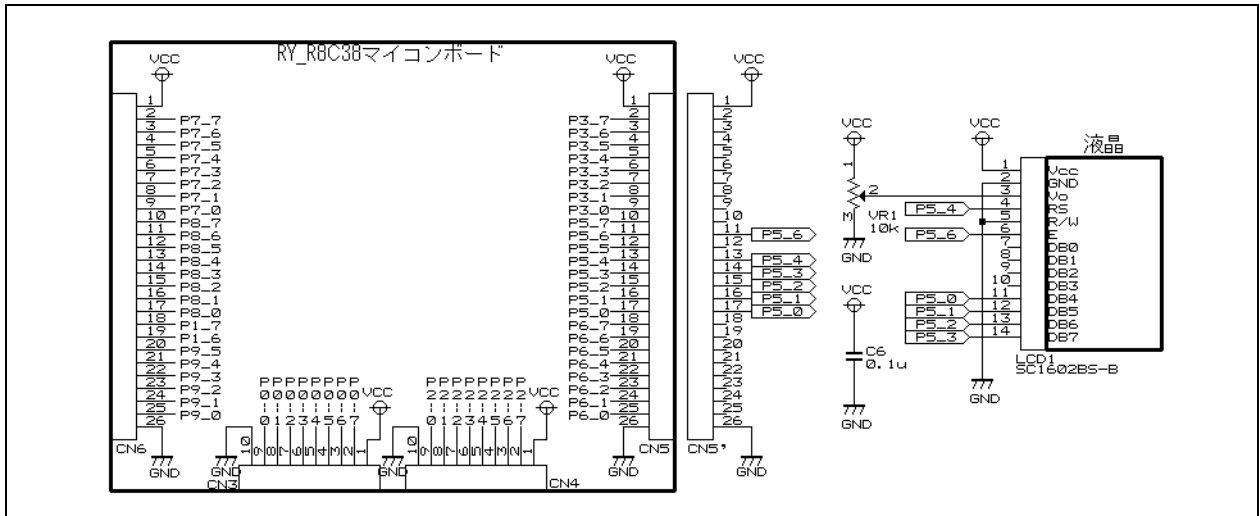


sample2.c を有効にして、ビルド、書き込みをして、実行してください。

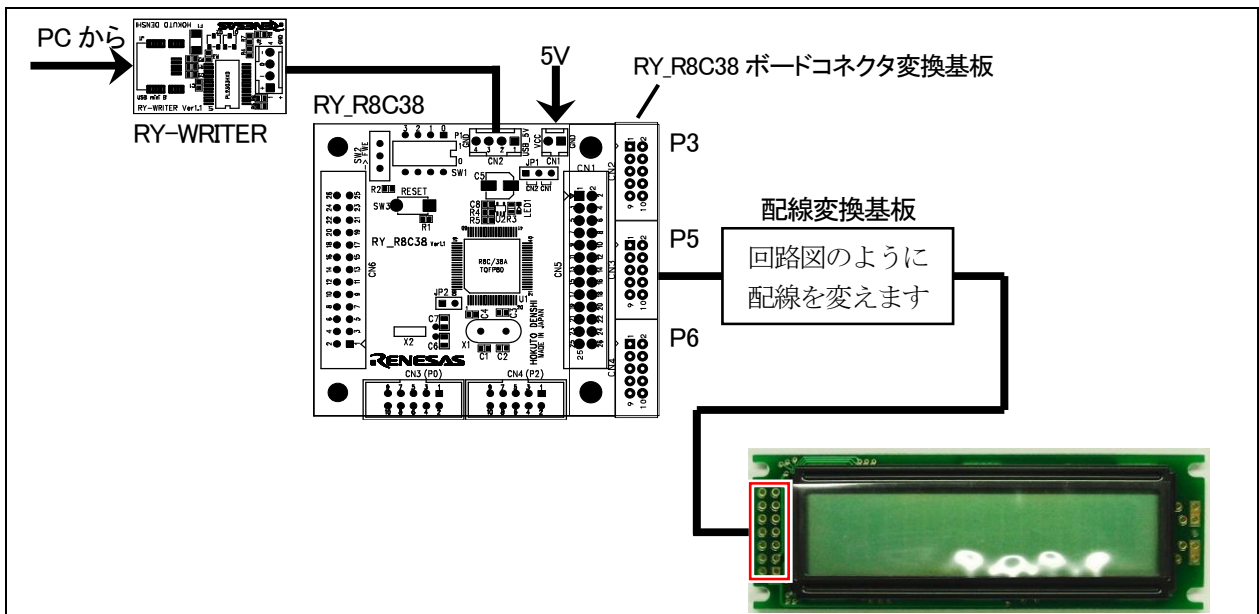
2. サンプルプログラム

2.4.5 sample3.c を実行する

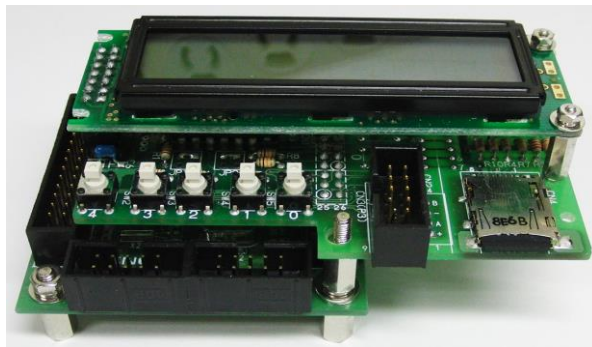
回路図を下記に示します。ポート 5 に液晶を接続します。



接続例を下記に示します。



または、RY\_R8C38 ボードの CN5 に液晶・microSD 基板を接続すれば、配線の変換は必要ありません。



▲液晶・microSD 基板を接続したところ

※RY\_R8C38 ボードコネクタ変換基板、液晶・microSD 基板については、マイコンカーラー販売サイト URL : <http://www2.himdx.net/mcr/> を参照してください。

sample3.c を有効にして、ビルド、書き込みをして、実行してください。

### 2.4.6 sample4.c を実行する

sample2.c と同等の配線です。sample2.c のプログラムとは違う割り込みを使ったサンプルプログラムです。  
例えば、

```
set_timer_b( INTERVAL_INT, 1000 );  
set_timer_f( INTERVAL_INT, 100 );
```

とすると、タイマ RB で 1.000ms の割り込みを発生、タイマ RF で 0.1ms ごとの割り込みを発生させることができます。

割り込み先は下記のようにプログラムします。

```
#pragma interrupt intTRB( vect = 24 )  
void intTRB( void )  
{  
    タイマ RB 割り込み処理  
    例えば「set_timer_b( INTERVAL_INT, 1000 )」を実行すると  
    1.000ms ごとにこの関数を実行します。  
}  
  
#pragma interrupt intTRF( vect = 16 )  
void intTRF( void )  
{  
    タイマ RF 割り込み処理  
    例えば「set_timer_f( INTERVAL_INT, 100 )」を実行すると  
    0.100ms ごとにこの関数を実行します。  
}
```

## 2. サンプルプログラム

## 2.4.1 sample5.c を実行する

sample2.c や sample4.c と同等の配線です。sample2.c や sample4.c のプログラムとは違う割り込みを使ったサンプルプログラムです。

例えば、

```
set_timer_b( INTERVAL_INT, 1000 );
set_timer_f( INTERVAL_INT, 100 );
set_timer_g( INTERVAL_INT, 500 );
```

とすると、タイマ RB で 1.000ms の割り込みを発生、タイマ RF で 0.1ms ごとの割り込みを発生、タイマ RG で 0.5ms ごとの割り込みを発生させることができます。

割り込み先は下記のようにプログラムします。

```
#pragma interrupt intTRB( vect = 24 )
void intTRB( void )
{
    タイマ RB 割り込み処理
    例えば「set_timer_b( INTERVAL_INT, 1000 )」を実行すると
    1.000ms ごとにこの関数を実行します。
}

#pragma interrupt intTRF( vect = 16 )
void intTRF( void )
{
    タイマ RF 割り込み処理
    例えば「set_timer_f( INTERVAL_INT, 100 )」を実行すると
    0.100ms ごとにこの関数を実行します。
}

#pragma interrupt intTRG( vect = 43 )
void intTRG( void )
{
    imfa_trgsr = 0;           // この行を必ず入れてください

    タイマ RG 割り込み処理
    例えば「set_timer_g( INTERVAL_INT, 500 )」を実行すると
    0.500ms ごとにこの関数を実行します。
}
```



## 2.4.2 sample6.c を実行する

指定した端子から設定した周波数のパルスを出力します。圧電サウンダーなどから音を鳴らすときに便利です。使用する端子の設定方法を下記に示します。

```
set_timer_c( FREQUENCY_OUT, TRC_F_P0_3 );
```

      部分に、出力したい端子を設定します。端子は下記の 19 端子から選ぶことができます。

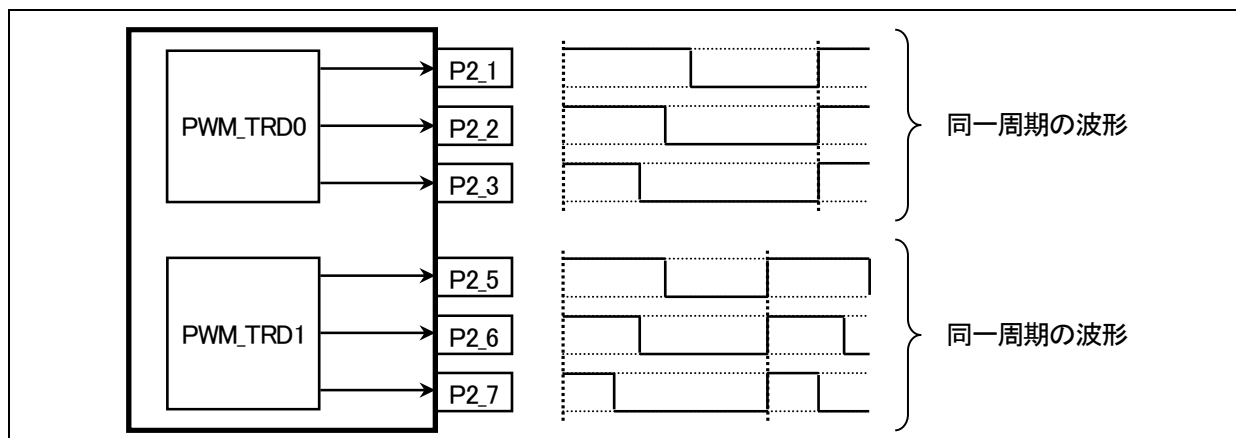
ポート 0	TRC_F_P0_3 , TRC_F_P0_4 , TRC_F_P0_5 , TRC_F_P0_6 , TRC_F_P0_7
ポート 1	TRC_F_P1_0 , TRC_F_P1_2 , TRC_F_P1_3
ポート 2	TRC_F_P2_0 , TRC_F_P2_1 , TRC_F_P2_2
ポート 3	TRC_F_P3_4 , TRC_F_P3_5
ポート 5	TRC_F_P5_2 , TRC_F_P5_3 , TRC_F_P5_4
ポート 6	TRC_F_P6_5 , TRC_F_P6_6 , TRC_F_P6_7

設定した周波数を出力するには、trc\_f 関数を使います。引数に周波数を入れます。例えば、440Hz のパルスを出力したときのプログラムを下記に示します。

```
trc_f( 440 ); // ピッチ 440Hz
```

## 2.4.3 sample7.c を実行する

PWM 波形を出力します。サーボモータや DC モータを制御するときに便利です。下図のようにチャンネル 0 とチャンネル 1 があり、それぞれ 3 つの端子から PWM 波形を出力することができます(最大、同時に 6 つの端子から PWM 波形を出力することができます)。端子の変更はできません。波形出力のイメージ図を下記に示します。



PWM\_TRD0 と PWM\_TRD1 の周期の設定方法を下記に示します。

```
set_timer_d( PWM_TRD0, 1000 ); // タイマ RD ch0 PWM 設定 周期 1000 μs
// ch0 は p2_1, p2_2, p2_3 端子の周期を決めます
set_timer_d( PWM_TRD1, 500 ); // タイマ RD ch1 PWM 設定 周期 500 μs
// ch1 は p2_5, p2_6, p2_7 端子の周期を決めます
```

それぞれの端子から PWM 波形を出力するプログラムを下記に示します。

```
trd_pwm_p2_1( 2500 ); // p2_1 端子から、25.00%の ON 幅の PWM 波形を出力
trd_pwm_p2_2( 5000 ); // p2_2 端子から、50.00%の ON 幅の PWM 波形を出力
trd_pwm_p2_3( 7500 ); // p2_3 端子から、75.00%の ON 幅の PWM 波形を出力

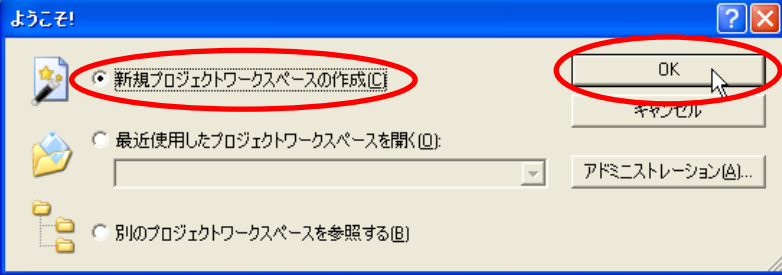


trd_pwm_p2_5( 2500 ); // p2_5 端子から、25.00%の ON 幅の PWM 波形を出力
trd_pwm_p2_6( 5000 ); // p2_6 端子から、50.00%の ON 幅の PWM 波形を出力
trd_pwm_p2_7( 7500 ); // p2_7 端子から、75.00%の ON 幅の PWM 波形を出力
```

## 2.4.4 sample8.c を実行する

printf 文を実行するプログラムです。

## 2.5 ライブラリを使った環境の構築方法

プロジェクト(ワークスペース)を新規に作る時、ライブラリを組み込む方法を説明します。

<p>1</p>		<p>ルネサス統合開発環境を立ち上げます。          「新規プロジェクトワークスペースの作成」を選択、「OK」をクリックします。</p>
<p>2</p>		<p>下記のように設定します。</p> <ul style="list-style-type: none"> <li>●プロジェクトタイプ 「C source startup Application」</li> <li>●ワークスペース名 自由に付けてください。</li> <li>●プロジェクト名 自由に付けてください。</li> <li>●CPU 種別 「M16C」</li> <li>●ツールチェーン 「Renesas M16C Standard」</li> </ul> <p>OKをクリックします。</p>
<p>3</p>		<p>下記のように設定します。</p> <ul style="list-style-type: none"> <li>●ツールチェーンバージョン 「6.00.00」または、それ以上</li> <li>●CPU シリーズ 「R8C/Tiny」</li> <li>●CPU グループ R8C/38A の場合「38C」 R8C/35A の場合「35C」</li> </ul> <p>次へをクリックします。</p>

2. サンプルプログラム

4		<p>下記のように設定します。</p> <ul style="list-style-type: none"> <li>●ROM サイズ R8C/38A の場合「128K」 R8C/35A の場合「32K」</li> </ul> <p><b>完了</b>をクリックします。</p>
---	--	---

5		<p><b>OK</b>をクリックします。</p>
---	--	---------------------------

6		<p>プロジェクト名と同じ C ファイル以外を、消します。 それぞれの C ファイルを選択して、<b>DEL</b>キーを押します。 Dependencies に登録されているファイルは、自動で消えますので、削除する必要はありません。</p>
---	--	---

2. サンプルプログラム

7

エクスプローラー(マイコンコンピュータ)で、「C ドライブ→Wrokspace→今回のワークスペース→今回のプロジェクト」のフォルダを開きます。  
 今回の例では、「C:\Workspace\sample\sample」になります。  
 プロジェクトと同じ名前のCファイル以外の次のファイルを削除します。

- 拡張子 C ファイル  
(プロジェクトと同じ名前のCファイルは消しません)
- 拡張子 H ファイル
- 拡張子 INC ファイル
- 拡張子 TXT ファイル

※ 詳細表示の種類でソートすると見やすいです。

8

「C:\Workspace\r8c38a\_lib\_sample\r8c38a\_lib\_sample」フォルダにある


- r8c38a\_lib.h
- sfr\_r838a.h
- startup.c

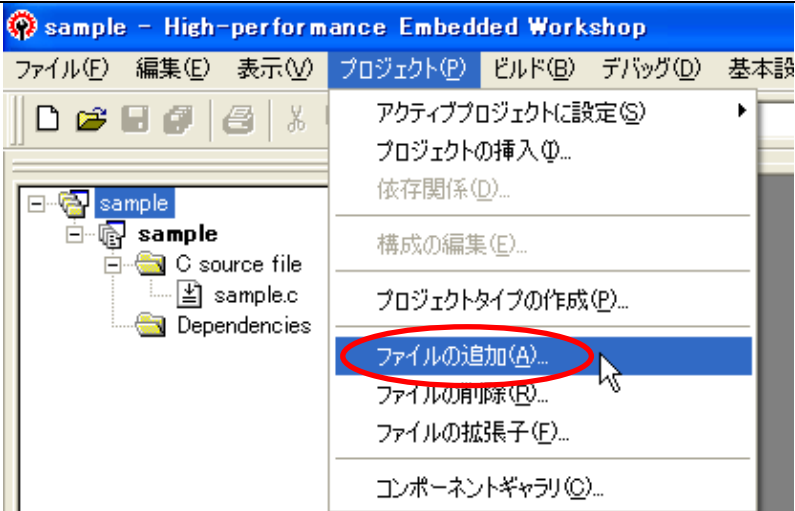
を、「C:\Workspace\sample\sample」フォルダ、または先ほど作ったプロジェクトのフォルダにコピーします。

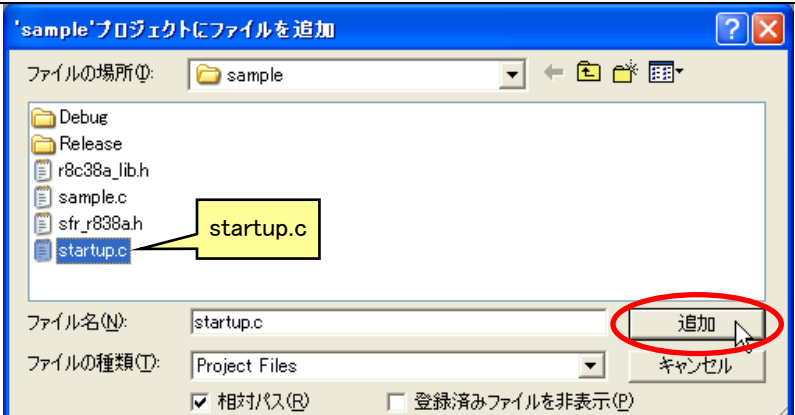
9

コピーしたところです。

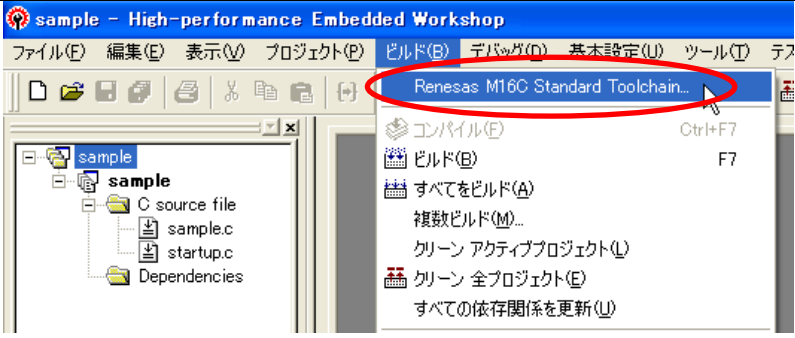
2. サンプルプログラム

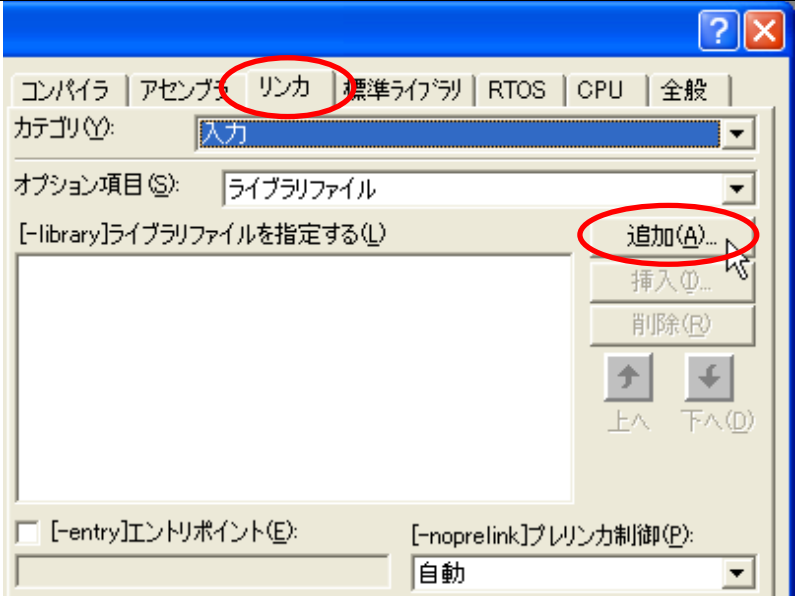
10		<p>「C:\Workspace\r8c38a_lib_sample\r8c38a_lib_sample\Debug」フォルダにある ●r8c38a_lib.lib を、 「C:\Workspace\sample\sample\Debug」フォルダ、または先ほど作ったプロジェクトのフォルダの Debug フォルダにコピーします。</p> <p>※R8C/35A の場合は、 「C:\Workspace\r8c35a_lib_sample\r8c35a_lib_sample\Debug」フォルダにある ●r8c35a_lib.lib をコピーしてください。</p>
----	--	---

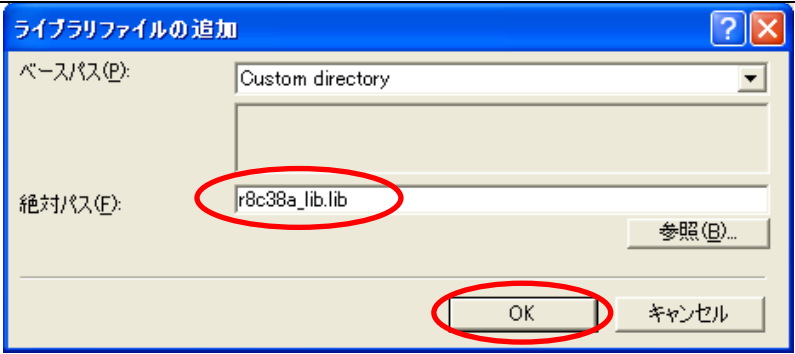
11		<p>「プロジェクト→ファイルの追加」を選択します。</p>
----	---	--------------------------------

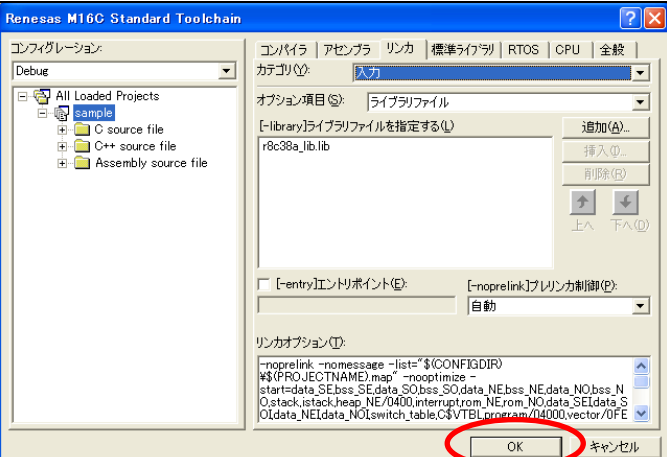
12		<p>「startup.c」を選択、追加をクリックします。</p>
----	--	-----------------------------------

2. サンプルプログラム

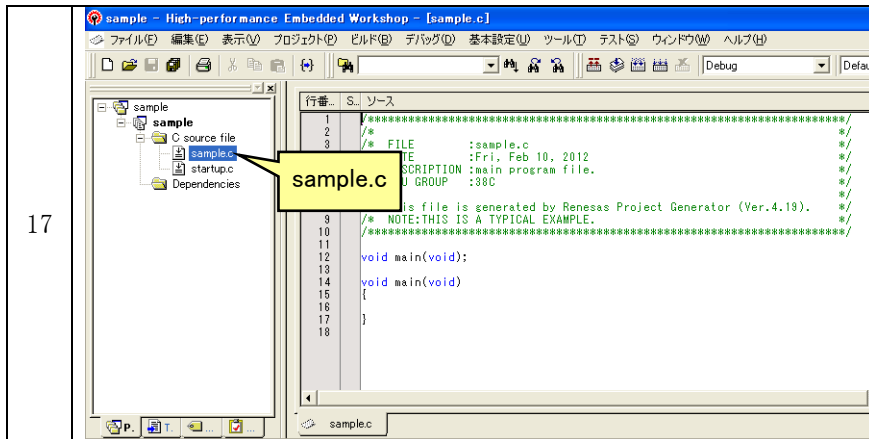
13		<p>「ビルド → Renesas M16C Standard Toolchain」を選択します。</p>
----	--	---

14		<p>「リンカ」タブを選択します。 追加をクリックします。</p>
----	---	---------------------------------------

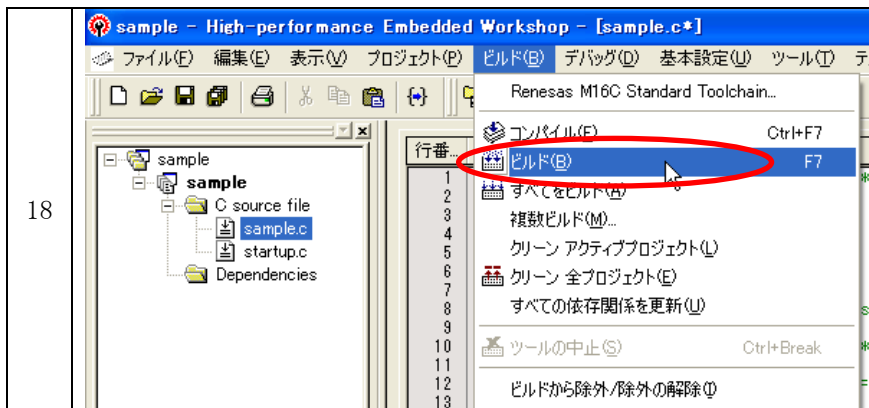
15		<p>絶対パスに次のファイル名を入力します。 えるあいびー てん えるあいびー r8c38a_lib.lib</p> <p>OKをクリックします。</p> <p>※R8C/35A の場合は、 えるあいびー てん えるあいびー r8c35a_lib.lib</p>
----	--	---

16		<p>OKをクリックします。</p>
----	---	--------------------

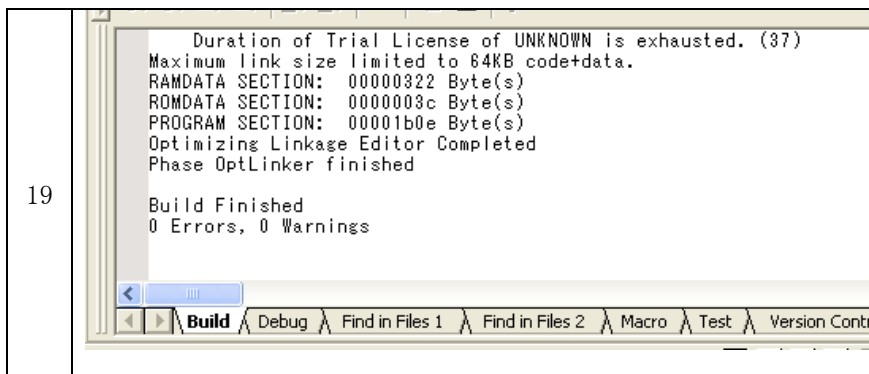
2. サンプルプログラム



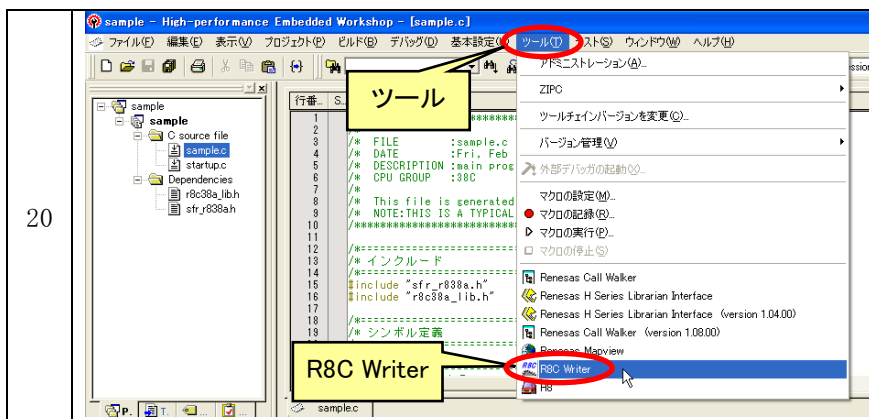
「sample.c」をダブルクリックして、プログラムを作りましょう。



プログラムが完成したら、「ビルド→ビルド」でビルドします。



結果が、0 Errors, 0 Warnings なら、プログラムに文法的な間違いはありません。



「ツール→R8C Writer」でプログラムを書き込んで、動作を確認しましょう！



## 3. ライブラリ関数

### 3.1 クロックに関する関数

#### 3.1.1 外付けクリスタル値のセット

書式	<code>void set_clk( long clk );</code>
内容	外付けクリスタルの周波数をセットします。
引数	外付けクリスタルの周波数
戻り値	無し
注意点	<code>init_xin_clk</code> 関数を実行する前に実行してください。 なお、初期値は 20MHz です。20MHz の場合は、設定する必要はありません。
使用例	<code>set_clk( 10e6 );</code> // クリスタルが 10MHz の場合 (10e6=10×10 <sup>6</sup> ) <code>init_xin_clk();</code> // CPU の動作クロックを XIN クロックにする

#### 3.1.2 外付けクリスタルに切り替え

書式	<code>void init_xin_clk( void );</code>
内容	マイコンの動作クロックを低速オンチップオシレータから外付け (XIN) クリスタルに切り替えます。 マイコンの電源を入れたとき、マイコン内蔵の低速オンチップオシレータで動作します。このオシレータの動作クロックは 125kHz と遅いので、外付け (XIN) クリスタルで動作に切り替えます。RY_R8C38 ボードの場合は 20MHz のクリスタルが実装されているので、本関数を実行後は、マイコンの動作が 20MHz になります。 また、本関数を実行すると、合わせてタイマ RA に値を設定します。そのため、これ以降のプログラムでタイマ RA は使えません。タイマ RA は、 <code>timer_ms</code> 関数で使います。
引数	無し
戻り値	無し
使用例	<code>init_xin_clk();</code> // CPU の動作クロックを XIN クロックにする

## 3. ライブラリ関数

## 3.2 ポートに関する関数

## 3.2.1 ポートの入出力設定

書式	<code>void pd( int port, unsigned char data );</code>
内容	マイコンのポートの入出力設定を行います。 マイコン起動時は、全ポート入力になっています。pdは「Port Direction」の略です。
引数	ポート番号：0～9 設定値：端子を入力にしたい場合は、該当 bit を"0"にします。出力にしたい場合は、該当 bit を"1"にします。
戻り値	無し
使用例	<code>pd( 1, 0xf0 );</code> // 設定値 1111 0000 なのでポート 1 の bit7～4 は出力、 // bit3～0 は入力になります。

## 3.2.2 ポートにデータ出力(ポート単位で出力)

書式	<code>void pout( int port, unsigned char data );</code>
内容	マイコンの出力に設定している端子からデータを出力します。設定はポート単位です。 端子が入力の場合は、設定しても何も起こりません。
引数	ポート番号：0～9 出力値：端子から出力したい値を設定します。該当 bit を"1"にすると、"1"(5V)が出力されます。該当 bit を"0"にすると、"0"(0V)が出力されます。
戻り値	無し
使用例	<code>pout( 1, 0xaa );</code> // 設定値 1010 1010 なのでポート 1 の bit7, 5, 3, 1 から"1"が、 // bit6, 4, 2, 0 から"0"が出力されます。

## 3.2.3 ポートからデータ入力(ポート単位で入力)

書式	<code>unsigned char pin( int port );</code>
内容	ポート(端子)の電圧を入力します。 端子が出力の場合は、現在出力している電圧が入力されます。
引数	ポート番号：0～9
戻り値	入力端子の場合は、ポートの入力電圧が戻り値になります。 出力端子の場合は、出力している電圧が戻り値になります。
使用例	<code>pd( 1, 0xf0 );</code> // ポート 1 の bit7～4 は出力、bit3～0 は入力 <code>pout( 1, 0xa0 );</code> // ポート 1 に"1010 0000"を出力 <code>d = pin( 1 );</code> // bit3～0 に"0011"が入力されている場合、bit3～0 にはこの値が // 入力されます。bit7～4 は現在出力している"1010"が入力されま // す。よって変数 d には、"10100011"(0xa3)が代入されます。

## 3. ライブラリ関数

## 3.2.4 端子にデータ出力(1bit ごとに出力)

書式	<code>void pin_out(int port, int pin, int out );</code>
内容	マイコンの出力に設定している端子からデータを出力します。設定は端子ごとに行います。端子が出力の場合は、設定しても何も起こりません。
引数	ポート番号 : 0~9 端子番号 : 0~7 出力値 : 端子から出力したい値を設定します。"1"を設定すると、"1"(5V)が出力されます。"0"を設定すると、"0"(0V)が出力されます。
戻り値	無し
使用例	<code>pin_out( 1, 5, 1); // ポート1のbit5から"1"を出力</code>

## 3.2.5 端子からデータ入力(1bit ごとに入力)

書式	<code>int pin_in( int port, int pin );</code>
内容	端子の電圧を入力します。端子が出力の場合は、現在出力している電圧が入力されます。
引数	ポート番号 : 0~9 端子番号 : 0~7
戻り値	●端子が入力の場合 1:"1"(5V)が入力されている 0:"0"(0V)が入力されている ●端子が出力の場合 1:"1"(5V)が出力されている 0:"0"(0V)が出力されている
使用例	<code>c = pin_in( 1, 0 ); // ポート1のbit0の状態を入力します</code>

## 3.2.6 端子のプルアップ制御

書式	<code>void set_pullup( int port, int sw );</code>
内容	マイコンには、プルアップ抵抗が内蔵されています。抵抗値は、25~100k $\Omega$ です。標準値は50k $\Omega$ です。 そのプルアップ抵抗をONにするか、OFFにするか設定します。この機能は、端子が入力の時だけ有効です(端子が出力の時は、抵抗はOFFになります)。 設定は、ポート単位で行います。1端子だけの設定はできません。
引数	ポート番号 : 0~9 設定値 : "0" : bit7~0のプルアップ抵抗をOFFにする "1" : bit7~0のプルアップ抵抗をONにする
戻り値	無し
使用例	<code>set_pullup( 1, 1 ) // ポート1のプルアップをONにする</code>

## 3. ライブラリ関数

## 3.3 A/D 変換に関する関数

## 3.3.1 A/D 変換

書式	<code>int get_ad( int ch );</code>
内容	A/D 変換を行います。R8C/38A マイコンには A/D 変換器が内蔵されていて、0~5V(マイコンボードの電源電圧)を 0~1023( $2^{10}-1$ )の値に変換することができます。
引数	A/D 変換器の入力端子選択 : 0~19 番号と端子の関係を下記に示します。 0:AN0 (P0_7) 1:AN1 (P0_6) 2:AN2 (P0_5) 3:AN3 (P0_4) 4:AN4 (P0_3) 5:AN5 (P0_2) 6:AN6 (P0_1) 7:AN7 (P0_0) 8:AN8 (P1_0) 9:AN9 (P1_1) 10:AN10(P1_2) 11:AN11(P1_3) 12:AN12(P7_0) 13:AN13(P7_1) 14:AN14(P7_2) 15:AN15(P7_3) 16:AN16(P7_4) 17:AN17(P7_5) 18:AN18(P7_6) 19:AN19(P7_7)
戻り値	A/D 変換値 : 0~1023 次の計算で、端子に何 V の電圧が入力されているか分かります。 入力電圧 = $5 \times \text{A/D 変換値} (0 \sim 1023) \div 1023$ ※5 は電源電圧です。 例えば、A/D 変換値が 100 のとき、入力電圧は、 入力電圧 = $5 \times 100 \div 1023 \approx 0.489\text{V}$ int 型や long 型は小数点は扱えません。float 型は集数点も扱えますが、処理が遅くなります。よって、5V を 5000 として扱えば、 入力電圧 = $5000 \times 100 \div 1023 \approx 489$ となり、小数第二位までを int 型で高速で処理することが出来ます。
使用例	<pre>i = get_ad( 0 ); // AN0(P0_7) 端子の電圧を A/D 変換する pout( 1, i &gt;&gt; 2 ); // A/D 変換値を 0~255 にして、ポート 1 に出力</pre>

## 3.4 タイマに関する関数

3.4.1 タイマ( $\mu\text{s}$  単位)

書式	<code>void timer_us( unsigned int timer_set );</code>
内容	$\mu\text{s}$ 単位で時間稼ぎをします。本関数実行中は、割り込みを禁止します。
引数	時間 unsigned int 型なので、0~65535 まで設定できます。
戻り値	無し
注意点	本関数は、init_xin_clk 関数で外付け(XIN)クリスタルに切り替え、クリスタル値が 20MHz の場合に約 $1\mu\text{s}$ になります。その他の場合は、 $1\mu\text{s}$ になりません。 また、時間は正確ではありません。目安として使用してください。
使用例	<pre>pin_out( 1, 5, 1 ); timer_us( 10 ); // 約 <math>10\mu\text{s}</math> の時間稼ぎ pin_out( 1, 5, 0 );</pre>

## 3. ライブラリ関数

## 3.4.2 タイマ(ms 単位)

書式	<code>void timer_ms( long timer_set );</code>
内容	ms 単位で時間稼ぎをします。本関数実行中も割り込みは受け付けます。
引数	時間 long 型なので、0~2147483647 (約 2147484 秒=35791.4 分=約 596.5 時間)まで設定できます。
戻り値	無し
注意点	タイマ RA を使用して時間を計ります。精度は、外付けクリスタルの精度に依存します。本関数を実行する前に、 <code>init_xin_clk</code> 関数で、外付け (XIN) クリスタルに切り替えてください。また、外付けクリスタルが 20MHz 以外の場合は、 <code>init_xin_clk</code> 関数を実行する前に <code>set_clk</code> 関数でクリスタル値をセットしてください。
使用例	<pre>pin_out( 1, 5, 1 ); timer_ms( 10 );           // 10ms の時間稼ぎ pin_out( 1, 5, 0 );</pre>

## 3.5 マイコンの動作に関する関数

## 3.5.1 ストップ

書式	<code>void stop( void );</code>
内容	マイコンの動作クロックを停止させて低消費電力モードへ以降、マイコンの動作をストップさせます。解除は、マイコンをリセットしてください。
引数	無し
戻り値	無し
使用例	<pre>stop();                   // マイコン動作ストップ</pre>

## 3.5.2 全体割り込みの許可

書式	<code>void ei( void );</code>
内容	マイコン全体の割り込みを許可します。割り込みを使用する場合、使用する内蔵周辺機能の割り込みを許可した後、全体割り込みを許可する必要があります。 ei は「enable interrupt」の略です。
引数	無し
戻り値	無し
使用例	<pre>ei();                     // 全体割り込み許可</pre>

## 3. ライブラリ関数

## 3.5.3 全体割り込みの禁止

書式	<code>void di( void );</code>
内容	マイコン全体の割り込みを禁止します。 di は「disable interrupt」の略です。
引数	無し
戻り値	無し
使用例	<code>di(); // 全体割り込み禁止</code>

## 3.6 タイマ RB に関する関数

※現在、タイマ RB の設定は、INTERVAL\_INT モードのみ対応しています。

## 3.6.1 タイマ RB 設定(インターバル割り込み)

書式	<code>int set_timer_b( int mode(動作モード), long data(設定値) );</code>
内容	タイマ RB をインターバル割り込みモードに設定します。
引数	動作モード: 「INTERVAL_INT」を設定します。 設定後、ei 関数で全体の割り込みを許可してください。 設定値: 割り込み発生間隔を $\mu s$ 単位で設定します。
戻り値	1:設定完了 0:設定不良
メモ	割り込み発生間隔について ①6553 $\mu s$ 以下の設定 1 $\mu s$ ごとに設定可能です。 ②26214 $\mu s$ 以下の設定 400ns の分解能で設定可能です。小数点が出た場合は切り捨てて設定されます。 例) 19999(19.999ms)を設定した場合は 400ns で割り切れる 19998(19.998ms)になります。 ③26214 $\mu s$ を超える設定 設定できません。 26214 $\mu s$ 以下で割り込みを発生させて、割り込みプログラム内でタイミングを取ってください。 例) 10 秒ごとに処理したい→1000 $\mu s$ (1ms)ごとに割り込みを発生させ、10,000 回ごとに処理させる、など
使用例	<pre>set_timer_b( INTERVAL_INT, 1000 ); // 1000us(1ms)ごとに割り込みを発生 ei(); // 割り込み許可  #pragma interrupt intTRB( vect = 24 ) // 24がタイマRB割り込みです。「vect=24」は変更できません void intTRB( void ) {     <span style="border: 1px solid black; padding: 2px;">割り込みプログラム(1msごとに実行されます)</span> }</pre>

## 3.7 タイマ RF に関する関数(R8C/35A は未対応です)

※現在、タイマ RF の設定は、INTERVAL\_INT モードのみ対応しています。

## 3.7.1 タイマ RF 設定(インターバル割り込み)

書式	<code>int set_timer_f( int mode(動作モード), long data(設定値) );</code>
内容	タイマ RF をインターバル割り込みモードに設定します。
引数	動作モード: 「INTERVAL_INT」を設定します。 設定後、ei 関数で全体の割り込みを許可してください。 設定値 : 割り込み発生間隔を $\mu s$ 単位で設定します。
戻り値	1:設定完了 0:設定不良
メモ	割り込み発生間隔について ①6553 $\mu s$ 以下の設定 1 $\mu s$ ごとに設定可能です。 ②26214 $\mu s$ 以下の設定 400ns の分解能で設定可能です。小数点が出た場合は切り捨てて設定されます。 例) 19999 (19.999ms) を設定した場合は 400ns で割り切れる 19998 (19.998ms) になります。 ③104857 $\mu s$ を超える設定 1600ns の分解能で設定可能です。小数点が出た場合は切り捨てて設定されます。 例) 29999 (29.999ms) を設定した場合は 1600ns で割り切れる 29998.4 (29.9984ms) になります。 ④104857 $\mu s$ を超える設定 設定できません。 104857 $\mu s$ 以下で割り込みを発生させて、割り込みプログラム内でタイミングを取ってください。 例) 10 秒ごとに処理したい→1000 $\mu s$ (1ms) ごとに割り込みを発生させ、10,000 回ごとに処理させる、など
使用例	<pre>set_timer_f( INTERVAL_INT, 1000 ); // 1000us(1ms) ごとに割り込みを発生 ei(); // 割り込み許可  #pragma interrupt intTRF( vect = 16 ) // 16がタイマRF割り込みです。「vect=16」は変更できません void intTRF( void ) {     [割り込みプログラム(1ms ごとに実行されます)] }</pre>

## 3.8 タイマ RG に関する関数(R8C/35A は未対応です)

※現在、タイマ RG の設定は、INTERVAL\_INT モードのみ対応しています。

## 3.8.1 タイマ RG 設定(インターバル割り込み)

書式	<code>int set_timer_g( int mode(動作モード), long data(設定値) );</code>
内容	タイマ RG をインターバル割り込みモードに設定します。
引数	動作モード: 「INTERVAL_INT」を設定します。 設定後、ei 関数で全体の割り込みを許可してください。 設定値 : 割り込み発生間隔を $\mu s$ 単位で設定します。
戻り値	1:設定完了 0:設定不良
メモ	割り込み発生間隔について ①13107 $\mu s$ 以下の設定 1 $\mu s$ ごとに設定可能です。 ②26214 $\mu s$ 以下の設定 400ns の分解能で設定可能です。小数点が出た場合は切り捨てて設定されます。 例) 19999 (19.999ms) を設定した場合は 400ns で割り切れる 19998 (19.998ms) になります。 ③104857 $\mu s$ を超える設定 1600ns の分解能で設定可能です。小数点が出た場合は切り捨てて設定されます。 例) 29999 (29.999ms) を設定した場合は 1600ns で割り切れる 29998.4 (29.9984ms) になります。 ④104857 $\mu s$ を超える設定 設定できません。 104857 $\mu s$ 以下で割り込みを発生させて、割り込みプログラム内でタイミングを取ってください。 例) 10 秒ごとに処理したい → 1000 $\mu s$ (1ms) ごとに割り込みを発生させ、10,000 回ごとに処理させる、など
使用例	<pre> set_timer_g( INTERVAL_INT, 1000 ); // 1000us(1ms) ごとに割り込みを発生 ei(); // 割り込み許可  #pragma interrupt intTRG( vect = 43 ) // 43 がタイマ RG 割り込みです。「vect=43」は変更できません void intTRG( void ) {     imfa_trgsr = 0; // TRG 割り込みの場合、この行を必ず入れてください      <span style="border: 1px solid black; padding: 2px;">割り込みプログラム(1ms ごとに実行されます)</span> } </pre>



## 3. ライブラリ関数

## 3.9 タイマ RC に関する関数

※現在、タイマ RC の設定は、FREQUENCY\_OUT モードのみ対応しています。

## 3.9.1 タイマ RC 設定(周波数出力)

書式	<code>int set_timer_c( int mode(動作モード), long data(設定値) );</code>												
内容	タイマ RC を周波数出力モードに設定します。												
引数	<p>動作モード：「FREQUENCY_OUT」を設定します。            設定値：波形を出力する端子を設定します。            端子は下記の 19 端子を設定することができます。</p> <table border="1"> <tr> <td>ポート 0</td> <td>TRC_F_P0_3, TRC_F_P0_4, TRC_F_P0_5, TRC_F_P0_6, TRC_F_P0_7</td> </tr> <tr> <td>ポート 1</td> <td>TRC_F_P1_0, TRC_F_P1_2, TRC_F_P1_3,</td> </tr> <tr> <td>ポート 2</td> <td>TRC_F_P2_0, TRC_F_P2_1, TRC_F_P2_2</td> </tr> <tr> <td>ポート 3</td> <td>TRC_F_P3_4, TRC_F_P3_5</td> </tr> <tr> <td>ポート 5</td> <td>TRC_F_P5_2, TRC_F_P5_3, TRC_F_P5_4</td> </tr> <tr> <td>ポート 6</td> <td>TRC_F_P6_5, TRC_F_P6_6, TRC_F_P6_7</td> </tr> </table> <p>※例) P0_3 端子にしたいとき「TRC_F_P0_3」を設定します。</p>	ポート 0	TRC_F_P0_3, TRC_F_P0_4, TRC_F_P0_5, TRC_F_P0_6, TRC_F_P0_7	ポート 1	TRC_F_P1_0, TRC_F_P1_2, TRC_F_P1_3,	ポート 2	TRC_F_P2_0, TRC_F_P2_1, TRC_F_P2_2	ポート 3	TRC_F_P3_4, TRC_F_P3_5	ポート 5	TRC_F_P5_2, TRC_F_P5_3, TRC_F_P5_4	ポート 6	TRC_F_P6_5, TRC_F_P6_6, TRC_F_P6_7
ポート 0	TRC_F_P0_3, TRC_F_P0_4, TRC_F_P0_5, TRC_F_P0_6, TRC_F_P0_7												
ポート 1	TRC_F_P1_0, TRC_F_P1_2, TRC_F_P1_3,												
ポート 2	TRC_F_P2_0, TRC_F_P2_1, TRC_F_P2_2												
ポート 3	TRC_F_P3_4, TRC_F_P3_5												
ポート 5	TRC_F_P5_2, TRC_F_P5_3, TRC_F_P5_4												
ポート 6	TRC_F_P6_5, TRC_F_P6_6, TRC_F_P6_7												
戻り値	1:設定完了 0:設定不良												

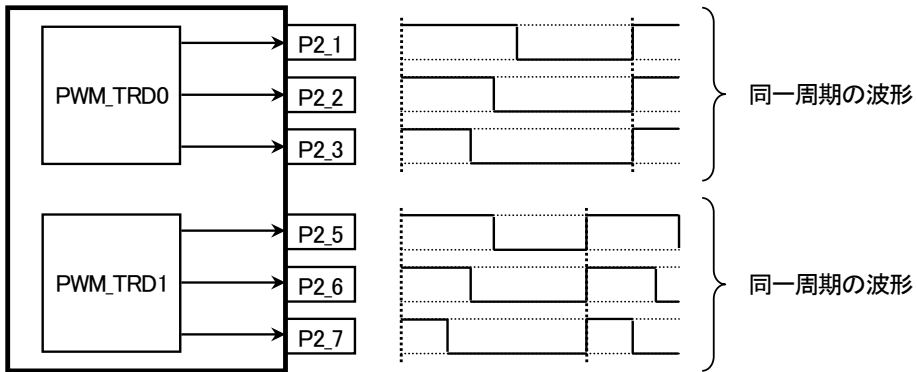
## 3.9.2 周波数の設定(周波数出力モード時)

書式	<code>int trc_f( int frequency )</code>
内容	FREQUENCY_OUT 動作モードで指定した端子から指定した周波数の波形を出力します。
引数	<p>周波数を設定します。設定できる範囲は、10~10MHz までです。</p> <p>①305Hz 以上の周波数を指定した場合            50ns の分解能で設定できます。            例えば 400Hz は、周期が <math>1/400=2.5\text{ms}</math>、50ns で割ると 50000(割り切れる)ので、正確な 400Hz を出力することができます。            例えば 401Hz は、周期が <math>1/401=2.493\cdots\text{ms}</math>、50ns で割ると 49875.31<math>\cdots</math>(割り切れない)ので小数点を切り捨てた周波数となります。</p> <p>②152Hz 以上の周波数を指定した場合            100ns の分解能で設定できます。</p> <p>③76Hz 以上の周波数を指定した場合            200ns の分解能で設定できます。</p> <p>④38Hz 以上の周波数を指定した場合            400ns の分解能で設定できます。</p> <p>④10Hz 以上の周波数を指定した場合            3200ns の分解能で設定できます。</p>
戻り値	1:設定完了 0:設定不良
使用例	<pre>set_timer_c( FREQUENCY_OUT, TRC_F_P0_3 ); // P0_3 端子を周波数出力端子にする  trc_f( 440 ); // 440Hz を P0_3 端子から出力する timer_ms( 100 ); // 100ms 待つ trc_f( 0 ); // 0Hz を P0_3 端子から出力する("0"のまま) timer_ms( 900 ); // 900ms 待つ</pre>

## 3.10 タイマ RD に関する関数

※現在、タイマ RD の設定は、PWM 出力モードのみ対応しています。

## 3.10.1 タイマ RD 設定(PWM 出力)

書式	<code>int set_timer_d( int mode(動作モード), long data(設定値) );</code>
内容	タイマ RD を PWM 波形出力モードに設定します。
引数	<p>動作モード: 「PWM_TRD0」または「PWM_TRD1」を設定します。  それぞれ端子が 3 つあります。TRD0 と TRD1 を使用すれば最大で PWM 波形を 6 つ出力することができます。イメージ図を下記に示します。端子は変更できません。</p>  <p>設定値 : PWM 周期を <math>\mu s</math> 単位で設定します。</p>
メモ	<p>PWM 周期の設定値について</p> <ol style="list-style-type: none"> <li>①13107 <math>\mu s</math> 以下の設定 1 <math>\mu s</math> 単位で設定可能です。</li> <li>②26214 <math>\mu s</math> 以下の設定 400ns の倍数で設定可能です。割り切れない場合は切り捨てて設定されます。 例) 26213 <math>\mu s</math> を設定した場合は 400ns で割り切れる 26212.8 <math>\mu s</math> になります。</li> <li>③104857 <math>\mu s</math> 以下の設定 1600ns の倍数で設定可能です。割り切れない場合は切り捨てて設定されます。 例) 100002 <math>\mu s</math> を設定した場合は 1600ns で割り切れる 100001.6 <math>\mu s</math> になります。</li> <li>④104857 <math>\mu</math> を超える設定 設定できません。</li> </ol>
戻り値	1:設定完了 0:設定不良

## 3. ライブラリ関数

## 3.10.2 PWM 波形出力

書式	<code>int trd_pwm_p2_x( int pwm );</code> ※x=1, 2, 3, 5, 6, 7
内容	p2_x 端子から、PWM 波形を出力します。
引数	PWM 波形の ON 幅の割合を設定します。0~10000 まで設定し、10000=100.00%のことです。 波形の周期は、 <code>set_timer_d(PWM_TRD0or1, 周期)</code> で設定した時間になります。  例) <code>trd_pwm_p2_2( 1234 );</code> // p2_2 端子から ON 幅 12.34%の波形を出力
戻り値	1:設定完了 0:設定不良

## 3.10.3 PWM 波形の出力状態取得

書式	<code>int get_trd_p2_x( void );</code> ※x=1, 2, 3, 5, 6, 7
内容	p2_x 端子から出力されている PWM 波形の状態("0"か"1"か)を取得します。
引数	なし
戻り値	1:"1"を出力中 0:"0"を出力中 例) <code>p0_1 = get_trd_p2_1();</code> // p2_1 の PWM 端子の状態を p0_1 へ出力
メモ	例えば、 <code>trd_pwm_p2_1</code> 関数で PWM 波形を出力しているとき、p2_1 端子は PWM 出力回路につながれているので、「p2」や「p2_1」では、端子の状態は読み取れません。読み取っても常に"0"になります。 例) <code>p0_1 = p2_1;</code> // <code>trd_pwm_p2_1</code> 関数で PWM 波形を出力中、これを実行しても <code>p2_1="0"</code> になる

## 3.11 マイコンボードの動作に関する関数

## 3.11.1 LED 点灯

書式	<code>void led_out( unsigned char led );</code>
内容	マイコンボード上の LED を点灯、消灯させます。
引数	●RY_R8C38 0:消灯 1:点灯 ●RMC-R8C35A 0~15 該当のビットが 0:消灯 該当のビットが 1:点灯
戻り値	無し
使用例	<code>led_out ( 1 );</code>

## 3.11.2 ディップスイッチ値取得

書式	<code>unsigned char dipsw_get( void );</code>
内容	マイコンボード上のディップスイッチ値を取得します。
引数	無し
戻り値	0~15
使用例	<code>c = dipsw_get();</code> // DIPSW が"1010"なら、 <code>c=10</code> が代入される

3. ライブラリ関数

3.12 液晶に関する関数

(株)秋月電子通商などで販売されている LCD キャラクタディスプレイモジュール (16×2 行) を制御することができます。

液晶に関する関数を使うと、ROM 容量が 10KB 程度増えます。

3.12.1 液晶初期化

書式	<pre>void lcd_init( unsigned char *p_e ,int b_e ,unsigned char *p_rs ,int b_rs ,                unsigned char *p_db7,int b_db7,unsigned char *p_db6,int b_db6,                unsigned char *p_db5,int b_db5,unsigned char *p_db4,int b_db4 );</pre>
内容	液晶をえるように、初期化します。
引数	<p>液晶の E 端子に接続しているポート : &amp;p0~&amp;p9 ※ポートは「&amp;」を付けて指定します                  液晶の E 端子に接続しているポートのビット : 7~0                  液晶の RS 端子に接続しているポート : &amp;p0~&amp;p9                  液晶の RS 端子に接続しているポートのビット : 7~0                  液晶の D7 端子に接続しているポート : &amp;p0~&amp;p9                  液晶の D7 端子に接続しているポートのビット : 7~0                  液晶の D6 端子に接続しているポート : &amp;p0~&amp;p9                  液晶の D6 端子に接続しているポートのビット : 7~0                  液晶の D5 端子に接続しているポート : &amp;p0~&amp;p9                  液晶の D5 端子に接続しているポートのビット : 7~0                  液晶の D4 端子に接続しているポート : &amp;p0~&amp;p9                  液晶の D4 端子に接続しているポートのビット : 7~0</p>
戻り値	無し
注意点	本ライブラリでは、液晶の RW 端子は使いません。 表示されたかの確認はせずに、1 文字表示すると約 10ms 待つてから、次の表示をします。
接続例	
使用例	<pre>lcd_init( &amp;p5, 6, &amp;p5, 4, &amp;p5, 3, &amp;p5, 2, &amp;p5, 1, &amp;p5, 0 );</pre> <p style="text-align: center;">① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫</p> <p>①液晶の E 端子に接続しているポート      ②液晶の E 端子に接続しているポートのビット                  ③液晶の RS 端子に接続しているポート      ④液晶の RS 端子に接続しているポートのビット                  ⑤液晶の D7 端子に接続しているポート      ⑥液晶の D7 端子に接続しているポートのビット                  ⑦液晶の D6 端子に接続しているポート      ⑧液晶の D6 端子に接続しているポートのビット                  ⑨液晶の D5 端子に接続しているポート      ⑩液晶の D5 端子に接続しているポートのビット                  ⑪液晶の D4 端子に接続しているポート      ⑫液晶の D4 端子に接続しているポートのビット</p>

## 3. ライブラリ関数

## 3.12.2 液晶に表示する位置の指定

書式	<code>void lcd_position( char x ,char y );</code>
内容	液晶に表示する位置を指定します。
引数	横の位置(x) : 0~19 ※0 がいちばん左、19 がいちばん右です 縦の位置(y) : 0~ 1 ※0 が上、1 が下です
戻り値	無し
使用例	<code>lcd_position( 0, 0 ); // 液晶の(x:0, y:0)の位置から</code> <code>lcdPrintf( "R8C/38A Library" ); // 文字を表示します</code>

## 3.12.3 printf 文と同じ書式で液晶に文字を表示

書式	<code>int lcdPrintf( const char *format, ... );</code>
内容	液晶に文字を表示します。
引数	printf 関数と同様です。浮動小数点(%f, %e)は使えません。
戻り値	正常時 : 出力した文字列 異常時 : 負の数
使用例	<code>int a = 64 , b = -64;</code> <code>char data[] = { "Hello_World!" }; ※_はスペース</code>  <code>lcdPrintf( "%8d", a ); // "_____64" ※_はスペース</code> <code>lcdPrintf( "%8d", b ); // "_____ -64"</code> <code>lcdPrintf( "%8u", a ); // "_____64"</code> <code>lcdPrintf( "%8u", b ); // "___65472"</code> <code>lcdPrintf( "%8o", a ); // "_____100"</code> <code>lcdPrintf( "%8o", b ); // "___177700"</code> <code>lcdPrintf( "%8x", a ); // "_____40"</code> <code>lcdPrintf( "%8x", b ); // "_____ffc0"</code> <code>lcdPrintf( "%8c", a ); // "_____@"</code> <code>lcdPrintf( "%16s", data); // "_____Hello_World!"</code>

## 3.12.4 液晶に文字を表示

書式	<code>void lcd_put_str( const char *str );</code>
内容	液晶に文字を表示します。printf 文の"%s"と同じです。
引数	文字列を設定します。
戻り値	無し
使用例	<code>lcd_position( 0, 0 ); // 液晶の(x:0, y:0)の位置から表示</code> <code>lcd_put_str( "ABCabc" );</code>

## 3. ライブラリ関数

## 3.12.5 液晶に 10 進数を表示

書式	<code>void lcd_put_num( long value, int keta );</code>
内容	液晶に 10 進数を表示します。printf 文の"%d"と同じです。
引数	値：-2147483648 ~ 2147483647 (long 型の範囲) 桁：1~10 ※値が負の数の場合は、「'-'+桁で指定した桁数の表示」となります。
戻り値	無し
使用例	<pre>i = 1234; lcd_position( 0, 0 ); // 液晶の(x:0,y:0)の位置から表示 lcd_put_num( i, 8 ); // "00001234"を表示 i = -i; lcd_position( 0, 1 ); // 液晶の(x:0,y:1)の位置から表示 lcd_put_num( i, 8 ); // "-00001234"を表示</pre>

## 3.12.6 液晶に 16 進数を表示

書式	<code>void lcd_put_hex( unsigned long value, int keta );</code>
内容	液晶に 16 進数を表示します。printf 文の"%x"と同じです。
引数	値：0x00000000 ~ 0xffffffff 桁：1~8
戻り値	無し
使用例	<pre>i = 0x1234; lcd_position( 0, 0 ); // 液晶の(x:0,y:0)の位置から表示 lcd_put_hex( i, 8 ); // "00001234"を表示</pre>

## 3.13 printf 文、scanf 文に関する関数

## 3.13.1 printf 文、scanf 文を使う初期設定

書式	<code>void init_uartx_printf( int sp );</code> ※x=0 または 2
内容	printf 文、scanf 文を使う初期設定を行います。 信号線は、 init_uart0_printf … 送信信号 P1_4、受信信号 P1_5 (プログラムを書き換えている線) init_uart2_printf … 送信信号 P3_7、受信信号 P3_4 を使用します。
引数	SPEED_4800…通信速度を 4800bps に設定します。 SPEED_9600…通信速度を 9600bps に設定します。 SPEED_19200…通信速度を 19200bps に設定します。 SPEED_38400…通信速度を 38400bps に設定します。 ※その他は、ビット数 8bit、パリティなし、ストップビット 1bit 固定です。
戻り値	無し
使用例	<pre>init_uart0_printf( SPEED_9800 ); printf( "Hello World!\n" ); // P1_4 端子からシリアル信号を出力</pre>

## 4. 参考文献

- ルネサス エレクトロニクス(株)  
R8C/38C グループ ユーザーズマニュアル ハードウェア編 Rev.1.10
- ルネサス エレクトロニクス(株)  
R8C/35A グループ ハードウェアマニュアル Rev.0.40
- ルネサス エレクトロニクス(株)  
M16C シリーズ,R8C ファミリー用 C/C++コンパイラパッケージ V.6.00  
C/C++コンパイラユーザーズマニュアル Rev.1.00
- ルネサス エレクトロニクス(株)  
High-performance Embedded Workshop V.4.09 ユーザーズマニュアル Rev.1.00
- 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