## BCRX7\_1M マイコン開発セット マニュアル

第2版 2020.12.17/2016.8.30 E2 liteの使用を追加

### 【 製品概要 】

本マニュアルはBCRX7\_1M CPUボードのソフトウエア開発を行うために必要なソフトウエア インストゥール手順、添付CDのサンプルプログラムの動作について解説されています。特に新しい統合 開発環境CS+ for CCにおける開発方法について多く記述してあります。「コード生成」機能でR L78のように簡単に、FPU内蔵で驚異の演算速度をご体験下さい。

※本CPUボード開発にはルネサスエレクトロニクス社製E1またはE2 liteが必要です。



#### 1. 開発環境、事前準備

- 1-1. 開発環境
  - a:開発セット 同梱物
  - b:BCRX7\_1M CPUボードの特徴
  - c: E1エミュレータ (デバッカ)
  - d:無償のCS+、RX用Cコンパイラのダウンロード
  - e: CDコピー、デバイスドライバのインストゥール
- 1-2 動作、デバック
  - a:CS+起動、コンパイル、書き込み、動作
  - b:新しいプログラムを作る CS+ 操作
    - b-1:初めにIOポートの設定
      - b-2: プログラムの書き込み、書く場所の注意
      - b-3:デバッカの設定がデフォルトはエミュレータなので注意
      - b-4:E1から電源供給
    - b-5:クロック発生回路を設定する必要があります
  - c:その他
    - c-1:動作中に変数の変化を見るには?
    - c-2:サンプルを走らせるときにvect.hの重複するアドレスを削除
    - c-3:三角関数math.hはインクルードもCS+の設定も必要
    - c-4:割り込みが入っているか?周期は?簡単なチェック方法
    - c-5:既存のプログラムを雛形として新しいプログラムを作る
    - c-6:コード生成と見落としがちな注意点

# 2. サンプルプログラム

- 2-1. sample1 出力ポートのON, OFF
- 2-2. sample2 SIO (USB) でパソコンとのやりとり
- 2-3. sample3 A∕D変換をUSB出力
- 2-4. sample4 割り込み
- 2-5. sample5 PWM出力
- 2-6. sample6 三角、対数、平方根関数を使う
- 2-7. sample7 D/Aにsin演算した正弦波を出力する

### 1-1. 開発環境

#### a:開発セット同梱物

BCRX7\_1M CPUボード SIO-USB絶縁変換器 DVD(サンプルプログラム、デバイスドライバ、ドキュメント) マニュアル(本誌) 電源ケーブル、USB(フルサイズ)ケーブル ※開発に必要ないませるエレクトロニクス社創デバッカ51または

※開発に必要なルネサスエレクトロニクス社製デバッカE1またはE2 Iiteは同封されておりません。別途必要です。E1は2019年末に製造中止となりました。



b : BCRX7\_1M CPUボードの特徴(R5F571MLCDFP 100ピン搭 載)

●ルネサス独自のRX CPUコア、内部32ビットデータバス幅マイクロコンピュータ。3.3V 2 40MHz動作可能。従来のRX製品に搭載されたコアとの互換性を踏襲しながらも更に強力に進化した RXv2コアを採用し、フラッシュ内蔵マイコンとして最高クラスとなる1044coremarkを実 現。フラッシュメモリ向けに最適化したキャッシュ(AFU)により240MHzノーウェイト相当のフ ラッシュメモリアクセスが可能。

●FPU 単精度浮動小数点数(32ビット)IEEE754に準拠したデータタイプ、および例外

●メモリ容量 内蔵フラッシュROM 4Mバイト、内蔵RAM512Kバイト 内蔵データフラッシュ 64Kバイト

●A/Dコンバータ : 12ビット分解能×22 変換速度0.48μsec/1ch(ADCLK=6 0MHz)

●D/Aコンバータ:12ビット分解能×1

●外部バス拡張機能:あり(外部にデータバス、アドレスバス等出力できます)

● I /Oポート:入出力:78、入力:1、プルアップ抵抗:78 オープンドレイン出力:78 5V トレラント:17

●タイマ:16ビットタイマパルスユニット(TPUa)(16ビット×6チャンネル)、ポートアウトプ ットイネーブル3(POE3a)、汎用PWMタイマ(GPTa)(16ビット×4チャンネル)、プログラ マブルパルスジェネレータ(PPG)、8ビットタイマ(TMRb)(8ビット×2チャンネル)×2ユニ ット、コンペアマッチタイマ(CMT)(16ビット×2)×2ユニット、コンペアマッチタイマW(CM TW)(32ビット×1チャンネル)×2ユニット、リアルタイムクロック(RTCd)、ウオッチドッグ タイマ(WDTA)、独立ウオッチドグタイマ(IWDTAa)

●イーサネットコントローラ(ETHERC)2チャンネル、USB2. 0FSホスト/ファンクション モジュール(USBb)1ポート、シリアルコミュニュケーションインターフェイス(SCIg、SCI h)9チャンネル、FIFO内蔵シリアルコミュニュケーションインターフェイス(SCIFA)4チャ ンネル、IICバスインターフェイス(RIICa)2チャンネル、CANモジュール(CAN)3チャ ンネル、シリアルペリフェラルインターフェイス(RSPIa)2チャンネル、クワッドシリアルペリフ ェラルインターフェイス(QSPI)1チャンネル

●シリアルサウンドインターフェイス(SSI)2チャンネル、サンプリングレートコンバータ(SRC)、 MMCホストインターフェイス(MMCIF)、パラレルデータキャプチャユニット(PDC)、温度セン サ等内蔵。

●オンチップデバッキングシステム:(FINEインターフェイス)

●動作周囲温度:-40~+85℃

●EEPROM: 25LC256(32Kバイト) 電源OFFでもデータ保持。 ※オプション(実 装品はご相談下さい)

●電源 2.7V~3.6V 単一 40mA(240MHz動作 TYPE)

E1デバッカを使用して動作させる時E1から3.3Vの電源を供給できます。 デバック時など200mA以内の使用であれば他に用意する必要はありません。

●クリスタル:メイン 12MHz(×4逓倍で50MHz作成)実装済み。

●デバックコネクタ:E1用(FINEインターフェイス)デバックコネクタ実装済み。

- ●基板サイズ 64×48×13(H)mm
- ●基板仕上げ 金メッキ RoHS指令準拠 基板、部品、半田付け全ての工程でRoHS指令準拠仕様。

## 基板大きさ(部品面)



# 省略

# 1-2 動作、デバック

## a:CS+ for CC 起動、コンパイル、書き込み、動作



CDに添付しているサンプルプログラムを使って、コンパイル、書き込み、動作の方法を示します。

CS+ for CCを起動します。ここでは例としてRX7\_1Asample¥sample1を動作させます。基板上のLED D1が点滅するプログラムです。

ファイル → ファイルを開く → sample1. mtpjをダブルクリックします。

(2) ファイルを開く

— → × ↑ 📜 « Wa	orkSpace > RX71M_sample > sample1 >	ٽ ~	sample1の検索	م
整理▼ 新しいフォルダー			∎== = <b>▼</b>	
🚡 デスクトップ 🖈 🛆	名前	更新日時	種類	サイズ
🗎 ドキュメント 🖈	cg_src	2016/04/20 17:03	ファイル フォルダー	
🤽 ダウンロード 🕺 🗉	DefaultBuild	2016/04/19 17:34	ファイル フォルダー	
🍋 ピクチャ 🛛 🖈	Sample1.mtpj	2016/04/20 17:03	MTPJ ファイル	17,427 K

プロジェクトツリーが表示されます。

E1は設定済みです。	
🔯 sample1 - RX E1(Serial) - (	S+ for CC -
ファイル(F) 編集(E) 表示(V) フ	ロジェクト(P)
🚳 スタート(S) 🔒 🗐 🗿	
プロジェクト・ツリー	ąχ
ĝ 🕜 🙎  🗃	
■ R5F571MLCxFP (マイク) ■ コード生成(設計ツール) ● CC-RX (ビルド・ツール) ● RX E1(Serial) (デバッグ ■ コード生成 ■ コード生成 ■ コード生成 ■ コード生成 ■ コード生成 ■ 「」cgdbsct.c ■ 「」cgrcg_tbrc.c ■ 「」cgsbrk.c ■ 「」cgvecttbl.c	-םארכםי יש-גע

-

もし、E2liteを使う場合は、右クリックで→使用するデバックツール→E2 liteを選択しま す。



通信方式はJTAGになってしまうので、FINEに切り替えて下さい。

sample1. cが中央に表示されます。とりあえず、実行してみます。E1のケーブルを基板のCN 1に挿入します。電源はE1から供給しますので、不要です。(写真ご参考)



「デバック・ツールヘプログラムを転送」をクリック。



上手く転送できると、今まで表示されていなかったプログラムの絶対番地が表示されます。E1から電源 がCPU基板に供給されます。



ここまでいかなかった場合、E1のデバイスドライバインストゥールをご検証願います。

次に、プログラムを動作させます。「CPUリセット後、プログラムを実行」をクリック。

	CPUリセット後、プログラムを実行します。
E1のRUN(糸 されます。	录LED)が点灯し、基板のD1が点滅したら動作しています。CS+の右下部にも表示
■RUN	実行中 ■RX E1(Serial) ⑦計測中
ここまで確認でき	きましたら、一度止めます。
へルプ(H)	
, 🖓 🗅 m 🤇	
	実行中のプログラムを停止します。 (Shift+F5)
main関数の	lwaitの数値2箇所を1桁0を増やしてみます。
ffc005f7 ffc005fe	PORTF.PODR.BYTE = 0x55; PORTJ.PODR.BYTE = 0x55;
ffc00605	main_wait(5000000);
ffc0060d ffc0060f ffc00611 ffc00613 ffc00615 ffc00617	PORTO.PODR.BYTE = 0×aa; PORT1.PODR.BYTE = 0×aa; PORT2.PODR.BYTE = 0×aa; PORT3.PODR.BYTE = 0×aa; PORT4.PODR.BYTE = 0×aa; PORT5.PODR.BYTE = 0×aa;
ffc00619 ffc00620 ffc00627 ffc00635 ffc00635 ffc00643 ffc00643 ffc00651 ffc00658 ffc0065f	PORT6.PODR.BYTE = 0xaa; PORT7.PODR.BYTE = 0xaa; PORT8.PODR.BYTE = 0xaa; PORT9.PODR.BYTE = 0xaa; PORTA.PODR.BYTE = 0xaa; PORTB.PODR.BYTE = 0xaa; PORTC.PODR.BYTE = 0xaa; PORTD.PODR.BYTE = 0xaa; PORTE.PODR.BYTE = 0xaa; PORTF.PODR.BYTE = 0xaa; PORTF.PODR.BYTE = 0xaa; PORTJ.PODR.BYTE = 0xaa;

main\_wait(5000000);

\*

ffc00666

1

8

セーブして

E) 表	表示(V)	プロジェ	クト(P)	ビルド(B	) デバッ	ッグ(D)	ツー
1		<b>x b b</b>	50	aa 🙇 🙈		-	10
	ファイ	ルに指定	された項	目を保存	<sub>ノ</sub> ます。	(Ctrl+S	5)
「ビノ	ルド後、	デバック	・ツール	へプログラ	ir main.c ムを転送	* 127 J	リハナ



「CPUリセット後、プログラムを実行」をクリック。 LEDの点滅が先ほどより、遅くなったのが目視できましたでしょうか?

次に、ブレークポイントの設定を行ってみます。一度、プログラムを停止させます。 ブレークポイントを2点設定しました。手のマークの部分をマウスでダブルクリック。

99	ffc00606	138	main_wait(5000000);	
100				-
101				,
102	ffc0060e	1	PORTO.PODR.BYTE = 0xaa;	
103	ffc00610	1	PORT1.PODR.BYTE = 0xaa:	
104	ffc00612	1	PORT2.PODR.BYTE = 0xaa:	
105	ffc00614	i	PORT3.PODR.BYTE = 0xaa:	
106	ffc00616	i	PORT4, PODR, BYTE = 0xaa:	
107	ffc00618	1	PORT5, PODR, BYTE = 0xaa:	
108				
109	ffc0061a	1	PORT6 PODR BYTE = Oxaa:	
110	ffc00621	- i	PORTZ PODR BYTE = Oxaa:	
111	ffc00628		PORTS PODR BYTE = Oxaa:	
112	ffc0062f		PORTS PODR BYTE = Oxaa:	
113	ffc00636		PORTA PODR BYTE = Oxaa:	
114	ffc0063d		PORTE PODE BYTE = Oxaa:	
115	ffc00644		PORTC PODR BYTE = Oxaa:	
116	ffc0064b		PORTD PODR BYTE = Oxaa:	
117	ffc00652		PORTE PODR BYTE = Oxaa;	
118	ffc00659		PORTE PODR BYTE = Ovaa:	
110	ffc00660		PORT L PODR BYTE = 0xaa;	
120	1100000		/ / / / / / / / / / / / / / / / / / /	/
121	ffc00668	1.00	main wait(50000000)	
121	11000000	0	mail: mail: (00000000),	

「CPUリセット後、プログラムを実行」します。

	PORT8.PODR.BYTE = 0x55; PORT9.PODR.BYTE = 0x55; PORTA.PODR.BYTE = 0x55; PORTB.PODR.BYTE = 0x55; PORTC.PODR.BYTE = 0x55; PORTD.PODR.BYTE = 0x55; PORTE.PODR.BYTE = 0x55; PORTF.PODR.BYTE = 0x55; PORTJ.PODR.BYTE = 0x55;	•
178	main_wait(5000000);	

プログラムの実行はブレークポイントで停止し、LED D1 は PORTD. PODR. BYTE= 0×55;命令によりPD0=1となるので、点灯します。

更に「プログラムを現在の位置から実行」をクリックすると、もう一つのブレークポイントで停止し、P ORTD. PODR. BYTE=0xaa;命令実行によりPD0=0になり、LEDは消灯します。



以上が、プログラムのコンパイル、E1へのダウンロード、実行、修正、ブレークポイント設定、動作の 概要です。

# **b**:新しいプログラムを作る

省略

## **b**-1:初めに I O ポートの設定

ここが従来のRXマイコン開発と変わった部分になります。コード生成ツールが完備してIO設定に限らず、あらゆるペリフェラルの初期設定をプログラムレスで開発できるようになりました(2016.9)。 RL78のCS+開発と同じように行えます。

ここではPORTCのPCOを出力設定します。コード生成(設計ルーツ)→ 周辺機能 → I / Oポー トを選択。PORTCのPCOを出力にチェック。

 $\backslash \setminus$ 

🔮 test_sample - CS+ for CC - [周辺機能					
ファイル(F) 編集(E) 表示(V) プロジェクト(F	P) ビルド(B) デバッグ(D) ツール	L(T) DAYF	ウ(W) ヘルプ(H)		
🏽 🙉 スタート(S)   🛃 🗐 🗿 🖁 🕷 🛍	♥ ♥ ■ ■		🔍 👻 😽 😽 DefaultBuild	• 🕺	i 🚮 🕰 🤭 i 📵 🕑 🖸
プロジェクト・ツリー + ×	🚰 プロパティ 💯 周辺機能*				
2 🕼 🚨 🛛	1 コードを生成する 👗	A K <sup>m</sup> (1)	a z z z n n n n n	3 <u>6 6</u> 6 -	1 <i>2277</i> #7
■ test_sample (プロジェクト)* ^					
- R5F571MLCxFP (マイクロコント	Port0 Port1 Port2 Port	B Port4   F	Ports   PortA   PortB   PortC   Por	tD   PortE   Por	tJ
□- 1型 コード生成 (設計ツール) ■ 2→ 端子図	○ 使用しない ○ 入力	● 出力	□ 内蔵プルアップ CMOS 出力	~	1を出力 高駆動出力
日-222 周辺機能	-PC1 ④ 使用しない 〇 入力	〇 出力	<ul> <li>内蔵ゴルアップ CMOS 出力</li> </ul>	× 🗆	1を出力 高駆動出力
<ul> <li>■ 電圧検出回路</li> <li>■ クロック周波数精度測定</li> </ul>	-PC2 ● 使用しない 〇 入力	〇出力	□ 内蔵ブルアップ CMOS 出力	× .	1を出力 高駆動出力
	-PC3 ● 使用しない 〇 入力	〇出力	<ul> <li>内蔵ブルアップ CMOS 出力</li> </ul>	×	1を出力 高駆動出力
	-PC4	〇出力	<ul> <li>内蔵ブルアップ CMOS 出力</li> </ul>		1を出力 高駆動出力
	-PC5 ● 使用しない 〇 入力	〇出力	□ 内蔵ブルアップ CMOS 出力		1を出力 高駆動出力
■ <u>いのハート</u> ■ ■ マルチファンクションタイマノ	-PC6 ● 使用しない 〇 入力	<ul><li>○ 出力</li></ul>	回 内蔵ゴルアップ CMOS 出力		1を出力□ 高販勧出力
● ● 汎用PWMタイマ	-PC7	0 414			
	● 使用しない ● 人力	〇 出力	〇 内蔵フルアップ CMOS 出力	× _	1を出力 高駆動出力

これでコードを生成するをクリックするとPORTCのPCOを出力に設定する初期化プログラムが自動 生成されます。

# b-2:プログラムの書き込み、書く場所の注意

省略

## b-3:デバッカの設定がデフォルトはエミュレータなので注意





E1からの電源供給が遮断されている状態で「使用するデバック・ツール」→ RX E1を選択して下 さい。

# b-4:E1から電源供給

省略

綺麗にLEDが点滅できたと思います。しかし、sample1と比べると遅い。実はまだ設定が十分で はありません。

b-5:クロック発生回路を設定する必要があります

省略

これでCPUはやっと240MHzで動作し、sample1とほぼ同じ間隔でLEDが点滅するのが確認できます。

c:その他

c-1:動作中に変数の変化を見るには?

省略

#### c-2:サンプルを走らせるときにvect. hの重複するアドレスを削除

ルネサス等から提供されているライブラリを自分で新規に製作したプロジェクトで走らそうとするときに、 vect. hの内容が重複定義となり、エラーが出る場合があります。vect. hの中の定義を削除し て下さい。

```
以下、RX2_1Aのsample3.cの中で定義されている割込み。
```

```
* Function Name: dsadi0 isr
* Description : DSAD channel0 interrupt rutine
* Arguments
           : none
* Return Value : none
*****
#pragma interrupt dsadi0_isr(vect = VECT(DSAD, DSADI0))
static void dsadi0_isr(void)
{
   /* Read conversion data of DSAD channel0
   DSADDR0
              Delta-Sigma Data Register 0
   b31-b0
            Holding A/D results. Read only register. */
   g_dsad_data[0] = (int32_t)DSAD.DSADDR0;
   /* Clear Interrupt Request Register assigned DSADI0. */
```

```
IR(DSAD, DSADI0) = 0;
```

}

vect. hの中をコメントにしないと2重定義でエラーがでます。

```
/*
```

```
// DSAD DSADI0
```

```
#pragma interrupt (Excep_DSAD_DSADI0(vect=207))
void Excep_DSAD_DSADI0(void);
```

// DSAD DSADI1
#pragma interrupt (Excep\_DSAD\_DSADI1(vect=208))
void Excep\_DSAD\_DSADI1(void);

// DSAD DSADI2
#pragma interrupt (Excep DSAD DSADI2(vect=209))

### void Excep\_DSAD\_DSADI2(void);

#### // DSAD DSADI3

#pragma interrupt (Excep\_DSAD\_DSADI3(vect=210))
void Excep\_DSAD\_DSADI3(void);

#### // DSAD DSADI4

#pragma interrupt (Excep\_DSAD\_DSADI4(vect=211))
void Excep\_DSAD\_DSADI4(void);

#### // DSAD DSADI5

#pragma interrupt (Excep\_DSAD\_DSADI5(vect=212))
void Excep\_DSAD\_DSADI5(void);

### // DSAD DSADI6

#pragma interrupt (Excep\_DSAD\_DSADI6(vect=213))
void Excep\_DSAD\_DSADI6(void);
\*/

# c-3:三角関数math.hはインクルードもCS+の設定も必要

sample6は三角、対数、平方根関数を使用しますが、ソースファイルにインクルードを記入するだけでなく、

#include <math.h>

//sqr 等演算を行うのに必要 math.h 有効と共にこの表記も必要。

Sample6 (ノロジェクト)	4 E-F	
■ R5F521A8BxFP (マイクロ	一 標準フィブフリの使用・構築方法	標準フイブフリ・ファイル作成オブション変更時
へCC-RX (ビルド・ツール)	▲ 標準フ1フフリ こ / ポニリ 押ま	e/eeeV i
A RY E1(Serial) (デパッグ・	フイフフリ 傍放 いい 堪勢対象 ホライブラリ	U(U89)(-lang=c)
	/ 悟奈内家のパイノア	(±()(=head=\SubOption2)
白 リファイル	リノダイム・Jイノリリと相対にする etyme b(OSD (ODD)を右対にする	(alu)(=nead=runtime)
dbsct.c	Ctype.n(Co3/C33)を有効にする 	(101% (+()(_bd=))
e intora c	math(1)(000/000)を方効にする	(a) (a) The automatic (a) (a) (a) (b) (b) (c) (c) (c) (c) (c) (c) (c) (c) (c) (c
	that 11:1(003/033/2有久)(293)	()()2
resetprg.c	studaig1(0007007/21) ctdia b(090/090)を右効にする	(t()(=headtetdia)
🖆 sbrk.c	stdlib h(C89/C99)を有効にする	(#(.)(=head=stdlih)
e vecttbl.c	string b(CB9/C99)を有効(Cする	(#(.)(-head=string)
I iodofino h	ins(EC++)を有効にする	()()
iddenne.n	new(EC++)を有効にする	(\$(,)(-head=new)
- Sbrk.h	complex(EC++)を有効にする	いいえ
- 🔄 stacksct.h	string(EC++)を有効にする	いいえ
- Urvedefine.h	オプジェクト	
6 week b	出力フォルダ	%BuildModeName%
vect.n	出力ファイル名	%ProjectName%.llb
sample6.c	機能縮小版入出力関数を生成する	いいえ
	プログラム 領域のセクション 名	P
	定数領域のセクション名	C
	初期化データ領域のセクション名	D
	未初期化データ領域のセクション名	B
	リテラル 領域のセクション 名	L
	switch 文分岐テーブル 領域のセクション 名	W
	初期値なし変数をアライメント4のセクションに配置する	いいえ
	初期値あり変数をアライメント4のセクションに配置する	いいえ
		1.0.17

CC-RX(ビルド・ツール) → ライブラリ・ジェネレート・オプション → math.hを有効にす る→はい としてください。

# c-4:割り込みが入っているか?周期は?簡単なチェック方法

sample4で1msec毎の定周期タイマー割り込みを使用しています。これがちゃんと割り込まれているか、周期、割り込み中の消費時間が分かる方法があります。8ビットタイマ→TMR0→10000  $\mu$  sec=1msec毎にCMIA0割り込みが入るように設定されています。

🞑 sample4 - RX E1(Serial) - CS+ for CC - [周辺機能]			
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバ	ッグ(D) ッ	ソール(T) ウインドウ(W) ヘルプ(H)	
🊳 スタート(S) 📴 🖬 🕼 🖄 🐚 🖄 🔍 🛱 🌉	88. -	🕶 🛛 100% 📼 🖬 🚮 Defaul	tBuild 🔹 🗸 🕴 🖏 🖎 🐂 I 💿 💿 🐂 🎯 I 🖘 🖓
プロジェクト・ツリー	<b>д х</b>	「 逆アセンブル1 「 プロパティ 「 r cg t	mmr user.c 📝 r cg main.c 📝 r cg tmr.c 🧏 周辺機能
2 🕜 🙎   🗃		◎ フートを生成する   素 ☆ 次 座 🥑	• * * • • • • • • • • • • • • • • • • •
Isample4 (プロジェクト)     In (プロジェクト)     In (アリン・ション・     In (アリン・     In (アリン・	^		
■ R5F571MLCxFP (マイクロコントローラ)		-カウント設定	
		クロックソース	PCLK V 60000 (kHz)
□ 22 周辺機能		外部クロック端子TMCI1	P12 V
		カウンタクリア	コンペアマッチ Aによりクリア 🗸
		外部リセット端子TMRIO	P20 V
	_	コンペアマッチ Aの 値(TCORA)	1000 µs v (実際の値: 1000)
	-	☐ S12AD A/D変換開始要求	
<b>⊡</b>		コンペアマッチ Bの 値(TCORB)	20 µs (実際の値: 20)
		-TMO0出力設定 □ TMO0出力許可	
			P22
■ 🔮 マルチファンクションタイマパルスユニット3		コンパアマッチム時の出力しパル	<u>∽</u> 変化(な()
→ ポートアウトプットイネーブル3		コンパアマッチ日時の出力レベル	☆(130000) 変化しない
■ ~ 汎用PWM91 マ ■ 🔮 16ビットタイマパルスコニット			2.130000
■ ■ プログラマブルパルスジェネレータ		-割り込み設定	司(CMIAO) 優先順位しびU45
			可(CMIRO) 優先順位レベル5
		- TONITオーバフロー割り込みを許可)	

「コード生成」で r \_ c g \_ t m r . c と r \_ c g \_ t m r \_ u s e r . c ファイルが新たに生成されま すが、 r \_ c g \_ t m e r . cの方は初期化設定=メインの初めに 1 回だけコールします、 r \_ c g \_ t m r \_ u s e r . cのほうにユーザーがプログラムを書き込むべき CM I A O 割り込み関数が生成されま す。

<b>吃</b> 逆	アセンブル 📑 プロ	1パティ 🥑	rjogitmrjuser.c 🗹 rjogitmr.c 🖉 rjogitmr.c 두
50   1	b)   🔿 🗠 🖍   力	<u>54</u> •	
行	# アドレス	<b>1</b>	
53			···/**********************************
54			* Function Name: r_tmr_cmiaU_interrupt
55			* Description : None
55			* Arguments : None
5/			* Keturn value : None
00			
09			HIT FAST INTERRUPT VECTOR - VECTOR PERID INIBUS
61			[#pragma interrupt r_tmr_cmra0_interrupt(vect=vect(Perib, INTB128), Fint)
62			Herse interrupt r tmr omial interrupt(vect=VECT(DEDID_INTR120))
62			(webt-vebt(rehtb))
64	ff-0128	2	static void r tmr smial interrupt(void)
65	1100100.	4	
66			/* Start user code. Do not edit comment generated here */
67		-	
68	ffc0138	f	PORTC.PODR.BIT.BO = 1; //LED1 ON
70	ffc0138	9	<pre>if(int_timer != 0){int_timer;}</pre>
/1	CC 0100		
72	TTCU139		PUKIC.PUUK.BII.BU = U; //LEUI UFF
74			
75			/* End user code. Do not edit comment generated here */
76			

r\_tmr\_cmia0\_interrupt割り込みの中でポートを立てて、抜け出す前に下ろしてい

ます。

この波形をオシロスコープで観測します。



TBS 1064 - 13:17:10 2015/08/11

細い筋が割込み周期です。ほぼ1msec毎になっています。また、1つの波形を時間軸を拡大すると



割り込みプログラムで50nsec×1.8 ≒ 90nsec程度の時間を消費していることが分かり ます。このポートがHでない部分がメインルーチンのプログラムが走っている時間です。

# c-5:既存のプログラムを雛形として新しいプログラムを作る

# 省略

⑤ t e s t \_ s a m p l e. c を削除。これで完成です。気になる方は、D e f a u l t B u i l d ホル ダの中の t e s t \_ s a m p l e ファイル群も削除して下さい。

プロジェクト・ツリー	9 ×
2 0 2 2	_
) 🖟 test_sample2 (プロ	コジェクト)*
R5F521A8BxFP (	マイクロコン
- CC-RX (ビルド・	ツー <mark>ル</mark> )
RX E1(Serial) (5	「バッグ・ツー
占 🍱 ファイル	
↓ ビルド・ツール	生成ファイル
- intprg.c	
sbrk.c	
	.c
test_sample.c	
- 🔄 iodefine.h	
- 🔄 sbrk.h	
📲 stacksct.h	
📲 typedefine.h	
vect.h	

# c-6:コード生成と見落しがちな注意点

例えば、下記のように割り込み要求を変更し→「コードを生成する」で



全てのプログラムソースが上書きされるのですが、

出力
M0409004:cg_src¥r_cg_sci.hを上書きしました。↓ M0409004:cg_src¥r_cg_s12ad.cを上書きしました。↓ M0409004:cg_src¥r_cg_s12ad_user.cを上書きしました。↓ M0409004:cg_src¥r_cg_s12ad.hを上書きしました。↓ M0409003:ファイルの生成を完了しました。↓ [EOF]

# 省略

コード生成で変更したつもりになっていても、これを忘れると変わっていないので、ご注意下さい。

# 2. サンプルプログラム

サンプルプログラムは全てコンパイル、動作確認済みです。CN1にE1のケーブルを差し、「デバック ツールへ<sub>|</sub>プログラムをダウンロード」後

1 = = = = = = = = = = = = = = = = = = =
<mark>デバッグ・ツール ヘプログラムをダウンロードします。</mark> / 端子配置図
CPUリセット後、実行で動作します。
: 👦 🗅 🍋 🔘 💌 🚱 🕬   🖘 Çı 🛎   🔏
CPUリセット後、プログラムを実行します。

電源はE1から供給しますので、新たに準備する必要はありません。

### sample1 ポートのON/OFF

#### 【 概要 】

出力可能な全ポートのON/OFFを繰り返します。PORTCのPCOに接続された基板上のLED D 1が点滅します。

#### 【周辺機能の説明】

サンプルプログラムは周辺機能→I/Oポート、ポートOからポートJまで出力可能なポートは全て出力 に設定し、「コード生成」してあります。この機能により、ユーザーはポートの初期設定を文章で記入す る必要が無く、「コード生成」で自動的に作成され、電源投入時、自動的に実行されます。

ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) ツール(T) ウインドウ(W) ヘルプ(H)	
38 スタート(S) 😺 🗟 🕼 🖄 🕾 🖄 🕫 🔍 🔍 📾 🌉 🔍 🔹 100% 👻 🗑 🚱 DefaultBuild 🔹 🔦	
プロジェクト・ツリー 🕴 🗶 端子配置表 🥤 r_og_main.c 📝 r_og_dbsct.c 🖄 プロパティ 冠 周辺機能 💋 端子配置	Ø
2 ② 2 図 - Fを生成する 点 Q 恋 性 学 編 品 品 詳 知 ③ ③ ③ ④ ③ ③ ③ ③ ③ ③ ③ ③ ③ ④ ③ ◎ ④ ◎ ④ ◎	<b>_ &amp; &amp; 7</b>
Port0 Port1 Port2 Port3 Port4 Port5 PortA PortB PortC PortD PortE Po	rtJ
回: 当 コード生成 (設計ツール) ○ 使用しない ○ 入力 ● 出力 □ 内蔵 プルアップ CMOS 出力 ▼ □	]1を出力
<ul> <li>● 2 クロック発生回路</li> <li>● 2 クロック発生回路</li> <li>● 使用しない ○ 入力</li> <li>● 出力</li> <li>● 内蔵ブルアップ</li> <li>CMOS 出力</li> <li>▼</li> </ul>	]1を出力
■ ● 電圧硬田回路 - ● クロック周波数構度測定回路 ● 谐差響 1氏減増能 のほ	]1を出力
● 割り込みコントローラ ● 割し込みコントローラ ● ジバス = 0 使用しない ○ 入力 ● 出力 ○ 内蔵ブルアップ CMOS 出力 ▼	]1を出力
● ● DMAコントローラ ● データトランスファコントローラ ロコ	]1を出力
- <sup>●</sup> イベントリンクコントローラ ● <sup>●</sup> //Oポート ● <sup>●</sup> マルチファンクションタイマパルスコーット3	]1を出力

```
【 プログラム 】
```

### \*\*\*\*\*

```
Global variables and functions
```

```
*****
```

1/\* Start user code for global. Do not edit comment generated here \*/

```
②void main_wait(long ltime)
```

```
{
```

```
while(ltime != 0)
{
    ltime--;
}
```

```
}
```

```
/* End user code. Do not edit comment generated here */
```

```
void R_MAIN_UserInit(void);
```

```
*****
* Function Name: main
* Description : This function implements main function.
* Arguments
          : None
* Return Value : None
*****
void main(void)
{
  R MAIN UserInit();
   /* Start user code. Do not edit comment generated here */
3
   while (1U)
  {
      (4)
             PORT0.PODR.BYTE = 0x55;
             PORT1.PODR.BYTE = 0x55;
             PORT2.PODR.BYTE = 0x55;
             PORT3.PODR.BYTE = 0x55;
             PORT4.PODR.BYTE = 0x55;
             PORT5.PODR.BYTE = 0x55;
             PORT6.PODR.BYTE = 0x55;
             PORT7.PODR.BYTE = 0x55;
             PORT8.PODR.BYTE = 0x55;
             PORT9.PODR.BYTE = 0x55;
             PORTA.PODR.BYTE = 0x55;
             PORTB.PODR.BYTE = 0x55;
             PORTC.PODR.BYTE = 0x55;
```

	PORTD.PODR.BYTE = $0x55$ ;
	PORTE.PODR.BYTE = $0x55$ ;
	PORTF.PODR.BYTE = $0x55$ ;
	PORTJ.PODR.BYTE = $0x55$ ;
5	main_wait(5000000);
6	PORT0.PODR.BYTE = 0xaa;
	PORT1.PODR.BYTE = 0xaa;
	PORT2.PODR.BYTE = 0xaa;
	PORT3.PODR.BYTE = 0xaa;
	PORT4.PODR.BYTE = 0xaa;
	PORT5.PODR.BYTE = 0xaa;
	PORT6.PODR.BYTE = 0xaa;
	PORT7.PODR.BYTE = 0xaa;
	PORT8.PODR.BYTE = 0xaa;
	PORT9.PODR.BYTE = 0xaa;
	PORTA.PODR.BYTE = 0xaa;
	PORTB.PODR.BYTE = 0xaa;
	PORTC.PODR.BYTE = 0xaa;
	PORTD.PODR.BYTE = 0xaa;
	PORTE.PODR.BYTE = 0xaa;
	PORTF.PODR.BYTE = 0xaa;
	PORTJ.PODR.BYTE = 0xaa;
$\bigcirc$	main_wait(5000000);

/\* End user code. Do not edit comment generated here \*/ 【 解説 】

# 省略

}

②void main\_wait(long ltime)
{
 while(ltime != 0)
 {
 ltime--;
 }

}

LEDのON, OFFを人間の目で確認するためには時間の早すぎるON, OFFではだめで、数100 msecの時間(ウエイト)を作るためのプログラムです。

```
③ /* Start user code. Do not edit comment generated here */
while (1U)
{
```

プログラムは/\* Start user code. 以下に記入してください。

4

PORT0.PODR.BYTE = 0x55; PORT1.PODR.BYTE = 0x55; PORT2.PODR.BYTE = 0x55; PORT3.PODR.BYTE = 0x55; PORT4.PODR.BYTE = 0x55; ; PORTC.PODR.BYTE = 0x55;

PC7	PC6	PC5	PC4	PC3	PC2	P C 1	PC0
0	1	0	1	0	1	0	1

(5) main\_wait(500000);

500000という数を減算して0になるまでの間、ポートが0×55に保たれます。LED D1も 点灯が保たれます。

- 6
- PORT0.PODR.BYTE = 0xaa; PORT1.PODR.BYTE = 0xaa; PORT2.PODR.BYTE = 0xaa; PORT3.PODR.BYTE = 0xaa; PORT4.PODR.BYTE = 0xaa; PORT5.PODR.BYTE = 0xaa;

0x55のビット反転数0xaaを出力しています。

Oxaa=10101010Bです。1ビット毎に1を立てています。PORTCのPCOに繋がれているLED D1もデータ'0'で電流が止まり、消灯します。

-		÷					
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PCO
1	0	1	0	1	0	1	0

0 x 5 5 と 0 x a a を交互に出す理由は、仮に隣のポートと接触しているとレベルの変化がありませんの で、例えばLEDが点滅しません。それによりハードウエアの異常が検出できます。(隣は必ず異なる論理 なので0, 1でも1, 0でも接触していると0になります。弊社出荷検査にて使用しています)

⑦ main\_wait(5000000);消灯している間も点灯同様に時間を保持しています。

### 2-2 sample2 SIO (USB) でパソコンとのやりとり

#### 【 概要 】

USB出力をパソコンと接続し、データのやり取りを行います。お手数ですが、テラタームやハイパータ ーミナルなどのターミナルプログラムを使用しますので、無い方は、ネットで検索し、インストゥール願 います。例ではテラタームで行います。ボーレートは9600bpsに設定して下さい。

USB-SIO絶縁変換器基板とCPUボードを接続します。USBケーブルでパソコンとつなげると、 USB基板側に電源が入ります。



スタート→右クリック→デバイスマネージャー → ポート(COMとLPT)でUSB Serial Port (COMxx)があることを確認して下さい (Windows10の場合)。例ではCOM4とな っています。



Tera Tremをシリアルポート COM4 →OKとします。

Fera Term: 新しい	接続	x
O TOP/IP	ホスト(T) 192.168.11.21	~
	<ul> <li></li></ul>	
	●SSH SSHバージョン(V): SSH2	$\sim$
	○その他 プロトコル(C): UNSP	EC Y
<ul><li>●シリアル</li></ul>	ポート(R): COM4: USB Serial Port (COM4)	~
	OK キャンセル ヘルプ(H)	

#### 設定→シリアルポート→ボーレート9600として下さい。

Tera Term: シリアルポート 設ち	Ē		x
ポート(P):	COM4	~	ОК
ボー・レート(B):	9600	~	
データ(D):	8 bit	~	キャンセル
パリティ(A):	none	~	
ストップ(S):	1 bit	~	ヘルプ(H)
フロー制御(F):	none	~	
送信遅延 0 ミリ利	♡/字(C) 0	= =,	Ⅰ秒/行(L)

CS+でsample2を開き、デバック・ツールヘプログラムダウンロード→CPUリセット後、プロ グラム実行。



USB Test、、 と表示され、PCのキーボードを何か押すたびに、押した文字が表示されると動作としてはOKです。

プログラムはパソコンのキーボードを押した文字がCPU基板に送信され、それを返信(エコーバック) し、表示されるようになっています。 【周辺機能の説明】



ここではSCI1を使用しています。青箱のふたが空いたように見える機能が使用されている機能です。

SCI1をクリック → 一般設定 調歩同期式、TXD1端子としてP16, RXD1端子としてP1 5が選択されています。

🚰 วีน/งริง	🐴 逆アセンブル	/1 🗹 r_o;	g_sci	.c 🗹 r_cg	_sci_us	er.c	🗹 r_o	og_m	ain.c	2月 唐	辺相	機能	
🐻 コードを生	成する 👗 📋	<i>🕷 </i> 🖾 🛙	<b>1</b>	) & &	<b>t</b> \$0	3	a 🐼	Ø	0	00	0		æ
SCI0 SCI1	SCI2 SCI3	SCI5 S	CI6	SCI12									
一般設定(	设定	90 O											
-機能設定 -													
() 使用	しない												
• 調歩	同期式			送信/受	を信		~						
〇 マルチ	·プロセッサモ <i>ー</i> ド			送信	送信								
עםל 🔾	り同期式 !			送信 🗸									
0 22-	トカードインタフ:	1 – J		送信									
() 簡易	120バス												
() 簡易	SPIバス 👎			スレーブ	送信/	受信	$\sim$						
- 端子設定													
TXD1	P16	· · · · ·	-	RXD1	P15				~				
SSDA1	P16			SSCL1	P15				$\sim$				
SMOSI1	P16			SMIS O1	P1 5				$\sim$				

ボーレートは9600bpsです。8ビットデータ、パリティなし、1ビットストップビット、ハードウ エアフロー制御なし。

SCI0 SCI1 SCI2 SCI3 SCI5 SCI6	SCI12		
一般設定設定			
-スタートビット 検出設定			
● RXD1 端子のLowレベル	─ RXD1 端子の立ち <sup>-</sup>	下がりエッシ	и /
-データ・ビット 長設定			
○ 9ピット	<ul> <li>๑ ยะีง</li> </ul>		○ 7ビット
-パリティ設定	0		~
● パリティなし	○ 偶数パリティ		○ 奇数パリティ
- ストップビット 設定			
	0 25%		
- テーダ転送方回設定 ● ISBファーフト			
転送りロック	内部クロック	~	
基本クロック	1ビット期間の16サイク	7.Jb 🗸	
ビットレート	9600	~	(bps) (実際の値: 9615.385, エラー: 0.16%)
□ ビットレートモジュレーション機能有効			
SCKI 端子機能	SCK1を使用しない	~	P17 🗸 🕐
- ノイズフィルタ設定			
□ノイズ除去機能を使用する			
ノイズフィルタクロック	1分周のクロック	~	60000000 (Hz)
-ハードウェアフロー制御設定			
● 禁止	🔾 CTS 🙂		🔘 rts 😲
CTS1/RTS1 端子	P1 4	$\sim$	
-データ処理設定			
送信データ処理	割り込みサービスルー	チンで処理	里する 🗸
受信データ処理	割り込みサービスルー	チンで処理	⊈र्वे ✓
- 割り込み 設定			
TXII 優先順位	レベル15	~	
RXII 優先順位	レベル5	~	
✓ エラー割り込み許可(ERI1)	1		
TEI1. ERI1 優先順位 (グループBLO)	レベル15		

この条件で「コード生成」が行われ、以下の2つの関数が生成されています。



r \_\_ c g \_\_ s c i . c は電源投入時に自動的に実行される void R\_SCI1\_Create(void)関数と、ユーザーが使い初めに1回だけコールする void R\_SCI1\_Start(void)関数が自動生成されています。

# 省略

割込みコントローラも設定する必要があります。

# 省略

# 【 プログラム 】

#### void main(void)

{

R\_MAIN\_UserInit();

/\* Start user code. Do not edit comment generated here \*/

①R\_SCI1\_Start();

②R\_SCI1\_Serial\_Receive(rx\_data,1);
rx\_flg = 0;
tx\_end\_flg = 0;

PORTC.PODR.BIT.B0 = 0; //LED1 OFF ③R\_SCI1\_Serial\_Send(String\_0,37); //Opening message //送信終了待ち ④tx\_end\_wait(); while (1U) 5 { 6  $if(rx_flg == 1)$ { PORTC.PODR.BIT.B0 = 1; //LED1 ON  $rx_flg = 0;$ //受信フラグクリア  $\bigcirc$ R\_SCI1\_Serial\_Receive(rx\_data,1); //1文字受信 R\_SCI1\_Serial\_Send(rx\_data,1); //1文字送信 } }

【解説】

# 省略

2-3 sample3 A/D変換をUSB出力

#### 【 動作概要 】

AN000(P40 CN4 16番)、CPU内部温度、基準電圧を入力とし、A/D変換した値をUS Bからパソコンに送り、表示しています。

SCOM4:9600baud - Tera Term VT	-	•	x
ファイル(F) 編集(E) 設定(S) コントロール(O) ウインドウ(W) ヘルプ(H)			
AD0 = 2.996V TEMP = 117.86d STD_VOLT = 1.255 V			~
ADO = 2.996V TEMP = 116.88d STD_VOLT = 1.254 V			
ADO = 2.985V TEMP = 117.86d STD_VOLT = 1.254 V			
AD0 = 2.998V IEMP = 118.06d SID_VOLI = 1.254 V			
ADO = 2.996V TEMP = 118.06d STD_VOLT = 1.252 V			
ADO = 2.987V TEMP = 117.67d STD_VOLT = 1.252 V			
ADO = 2.995V TEMP = 117.67d STD_VOLT = 1.252 V			
ALU = 3.000V TEMP = 117.86d STD_VULT = 1.253 V			
ADU = 2.987V TEMP = 117.86d STU_VULT = 1.254 V			
ADU = 2.995V TEMP = 118.26d STD_VULT = 1.252 V			
ADU = 3.001V TEMP = 117.67d STD_VULT = 1.252 V			
ADU = 2.990V TEMP = 117.86d STU_VULT = 1.252 V			
ADU = 2.995V TEMP = 117.67d STD_VULT = 1.252 V			
ADU = 2.996V TEMP = 118.06d STD_VULT = 1.255 V			
ADU = 2.987V TEMP = 118.45d STU_VULT = 1.252 V			
ADU = 2.999V IEMP = 118.26d SID_VULI = 1.253 V			
ADU = 3.000V TEMP = 118.65d STD_VULT = 1.253 V			
ADU = 2.988V TEMP = 118.26d STU_VULT = 1.254 V			
ADU = 2.996V TEMP = 118.400 STU_VULT = 1.203 V			
ADU = 3.002V IEMM = 118.26d SIU_VULI = 1.253 V			
ADU = 2.9960 IEMM = 118.200 SIU_VULI = 1.252 V			
ADU = 2.3367 IEMM = 117.366 SIL_VULT = 1.252 V			=
ADU = 2.9967 IEMM = 118.268 SID_VULT = 1.255 V			-
			~

#### 【周辺機能の説明】

SCI1に加えて、12ビットA/Dコンバータを使用しています。



ANOOO入力のための設定です。S12ADOを選択しています。

省略

温度センサ、内部基準電圧を読み込むためのS12AD1の設定です。

省略

# 【 プログラム 】

voi	d main(void)								
{									
	R_MAIN_UserInit();								
	/* Start user code. Do not edit comment generated here */								
	R_SCI1_Start();	//SCI1動	乍開始						
1	R S12AD0 Start();	//AD0動作	乍開始						
2	R_S12AD1_Start();	//AD1動作	乍開始						
	R_SCI1_Serial_Receive(rx_ rx_flg = 0; tx_end_flg = 0;	_data,1);							
	PORTC.PODR.BIT.B0 = 0;	i	//LED1 OFF						
	R_SCI1_Serial_Send(String tx_end_wait();	g_0,37);	//Opening message //送信終了まち						
	while (1U) {								
3	S12AD.ADCSR S12AD1.ADCS	.BIT.ADST R.BIT.ADS	T = 1; T = 1;	//ADスタート //AD1スタート					
4	while(S12AD.A	DCSR.BIT.	ADST)						

ad0 = S12AD.ADDR0; //AD値

;

8	fdata3 = stvolt/(4095/3.3);	//データ→電圧捜	與算 type1.25V 1.20~1.30V
9	<pre>sprintf(tx_buffer,"AD0 = %.3fV TEMP = %.2fd</pre>	$STD_VOLT = \%.3 f V = n r$	",fdata1,fdata2,fdata3);
10	R_SCI1_Serial_Send(tx_buffer,sized	f(tx_buffer)); //デー	タをUSB出力
	tx_end_wait();	//送信終了まち	

PORTC.PODR.BIT.B0 = 1;	//LED1 ON
main_wait(1000000);	
PORTC.PODR.BIT.B0 = 0;	//LED1 OFF
main_wait(1000000);	

# 【解説】

}



ウォッチ1	
🕘   🌒   🛃 🖏 🗙   🕯	表記(N)▼ 🔤
ウォッチ式	値
📷 SCI1.SSR	0x84
🗉 🔍 rx_data	
😜 ad0	124 (0x007c)
🔍 templ	1567 (0x061f)
😜 stvolt	1557 (0x0615)
🖃 🔍 tx_buffer	"ADO = 0.100V…
€ [0]	'A' (0x41)
😜 [1]	'D' (0x44)
😜 [2]	'0' (0x30)
😜 [3]	''(0x20)
😜 [4]	'=' (0x3d)
😜 [5]	''(0x20)
😜 [6]	'0' (0x30)
😜 [7]	'.' (Ox2e)
😜 [8]	'1' (0x31)
😜 [9]	'0' (0x30)
😜 [10]	'0' (0x30)
😜 [11]	'V' (0x56)
😜 [12]	''(0x20)
😜 [13]	'T' (0x54)
😜 [14]	'E' (0x45)
👻 [15]	'M' (0x4d)
😜 [16]	'P' (0x50)
😺 [17]	''(0x20)
😜 [18]	'=' (0x3d)

## 2-4 sample4 割り込み

#### 【 動作概要 】

sample4を動作させます。ここでは定周期割り込みについてサンプルを示します。sample3のループ時間を割込みを使って正確に1秒とし、データを出力しています。新たに経過秒を示しています。

ファイル(F) 編集(F) 影	設定(5) コントロール(0)	บ่∡่นหี่บ่เพ่า ∧แป้(เ	-1)	/
ファイル(F)         編集(E)         数           /0         Test         Beyond the           00         = 0.082V         TEMP =           00         = 0.082V         TEMP =           00         = 0.087V         TEMP =           00         = 0.081V         TEMP =           00         = 0.087V         TEMP =           00         = 0.082V         TEMP =	酸定(S) コントロール(O)       P river 2016.9     = 31.78d STD_VOLT     = 32.95d STD_VOLT     = 32.95d STD_VOLT     = 33.15d STD_VOLT     = 33.35d STD_VOLT     = 33.35d STD_VOLT     = 33.55d STD_VOLT     = 33.54d STD_VOLT     = 33.74d STD_VOLT     = 33.94d STD_VOLT     = 34.13d STD_VOLT     = 34.13d STD_VOLT     = 34.13d STD_VOLT     = 33.74d STD_VOLT     = 34.13d STD_VOLT     = 34.74d STD	ウインドウ(W) ヘルプ(H = 1.263 V Timer = = 1.262 V Timer = = 1.262 V Timer = = 1.262 V Timer = = 1.263 V Timer = = 1.263 V Timer = = 1.264 V Timer = = 1.264 V Timer = = 1.262 V Timer = = 1.262 V Timer = = 1.262 V Timer = = 1.261 V Timer = = 1.264 V Timer = = 1.261 V Timer = = 1.263 V Timer = = 1.263 V Timer = = 1.263 V Timer = = 1.264 V Timer = = 1.264 V Timer = = 1.264 V Timer = = 1.263 V Timer = = 1.264 V Time	4)       1       2       3       4       5       6       7       8       9       10       11       12       13       14       15       16       17       18       19	

オシロスコープがあれば PC0 CN6 20番を観測すると、以下のような1msec毎の波形が観測 できます。



【周辺機能の説明】

s a m p Ⅰ e 3 で使用している周辺機能	に加えて、8ビットタイマ	7 TMR0を使って、	1msec 定
周期割り込みを実現させています。	/		/
🔯 sample4 - RX E1(Serial) - CS+ for CC - [プロジェクト・ツリー]			
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デパッグ(D) ツ	ール(T) ウインドウ(W) ヘルプ(H)	/	,
🚳 スタート(S) 退 🖩 🗿 🐰 🐚 🛍 🔊 唑 🖁 🐥	🕶 🛛 100% 👻 🗖 🐻 Default	uild 🕘 🐁 😽 🛛	🖲 🕞 🕞 🔭 🎯 🖘 Çe
プロジェクト・ツリー 🛛 🕈 🗙	「 逆アセンブル1 プラロパティ Mrcgtm	ruser.c 🏹 r.cg main.c 🏹 r.g tmr.c.	🍸 r. cg. sci user.c 🧏 周辺
2 🞯 🙎   🗷			<u></u>
- R5F571MLCxFP (マイクロコントローラ) ヘ			
□ 1−ド生成 (設計ツ−ル)		2	
▲ / 端子図	<b>7</b>	PCLK	▼ 60000 (kHz)
□ 2000 mm 0 20000 mm 0 2000 mm 0 20000 mm 0 200000 mm 0 200000 mm 0 200000000	外部クロック端子TMCII	P12 V	
■ 🔮 電圧検出回路	カウンタクリア	コンペアマッチ Acよりクリア	~
	外部リヤット端子TMRIO	P20	
	コンペアマッチ Aの 値(TCORA)	1000 us v (東	上際の値:1000)
● 割り込みコントローフ =	☐ S12AD A/D変換開始要求		
	<u>-</u> いいの い し ((ICORB)	20 us (1	際の値:20)
		μο (3	(MO) 12. 207
	-TMOO出力設定		
	L TMOO出力許可		
■ マイルノファンションショイマハルスユニット3	TMOO認子	P22 V	
■ ● 汎用PWMタイマ	コンペアマッチA時の出力レベル	変化しない	×
в € 16ビットタイマパルスユエット	コンペアマッチB時の出力レベル	変化しない	× .
□ □ ブログラマブルパルスジェネレータ	- 割り込み設定		
	✓ TCORAコンペアマッチ割り込みを許可	(CMIAO) 優先順位レベル15	~
- TMR1	□ TCORBコンペアマッチ割り込みを許可	(CMIBO) 優先順位 レベル15	Y
<sup>©</sup> TMR2	□ TCNTオーパフロー割り込みを許可(C	MO) 優先順位レベル15	Y
TMR3			

「コード生成」で r\_cg\_tmer.c r\_cg\_tmer\_user.c

2つのプログラムが自動作成されます。

## 【 プログラム 】

sample3と比べて変わった所だけ書きます。

void main(void)

{

```
R_MAIN_UserInit();
```

/\* Start user code. Do not edit comment generated here \*/

```
R_SCI1_Start(); //SIO初期化
```

R_S12AD0_Start();	//AD初期化
R_S12AD1_Start();	//AD初期化

① R\_TMR0\_Start(); //8 ビットタイマー初期化

;

```
r_cg_tmer_user.c の割り込みの部分
```

static void r\_tmr\_cmia0\_interrupt(void)

{

/\* Start user code. Do not edit comment generated here \*/

(5) **PORTC.PODR.BIT.B0 = 1;** //LED1 ON

```
6 if(int_timer != 0){int_timer--;}
```

PORTC.PODR.BIT.B0 = 0; //LED1 OFF

/\* End user code. Do not edit comment generated here \*/

}

## 【解説】

# 省略

このように、割り込みで時間を作り、メインで使用すれば、極めて正確なタイマーが多数作成可能です。 定周期割り込みはその中に様々なプログラムを作成することも可能で、機能的なプログラム作成に不可欠 な知識、要素です。

## 2ー5 sample5 PWM出力 【 動作 】

汎用PWMタイマを使って、PWM波形を作ります。GTIOCOA端子 P23 CN5 20番、G TIOCOB端子 P17 CN5 16番に出力されます。



PWM波形は上図のように、周期が変わらず、設定値によってH、Lの幅の比率が変化します。この出力でLEDやモーターをドライブすると明るさや速度を変えることが出来るので、現代では様々な用途に使われています。

#### 【周辺機能の説明】

汎用PWMタイマ→GPTOでタイマ周期レジスタを10µsecに設定。f(周波数)=1/周期で 周波数は100KHzとなります。周期レジスタ(GTPRO)の値が自動的に599になります。一般 的に変化の応答を早くしたいときは周期を短くし、分解能を重視したいときは周期を長くします。

# 省略

コンペアマッチレジスタGTCCRAに書く値を0~599まで変えることによりPWMの幅が変ります。 初期値として99をセットしています。PWM出力端子をP23, P17に設定しています。

🚰 プロパティ 🐁 逆アセンブル1 🥑 r_c	sg_main.c 📝 r_cg_gpt.c 🧏 周辺機能		
🔞 コードを生成する   🍒 🔒 🕷 😤	e 🛛 🕹 🕹 😂 💷 🕲 🔍 🕲 🖉	1 🔿 Ø	🌼 🔲 🔗 🔗 🍼 🍼
-コンペアマッチレジスタGTCCRA、GTIC	DCOA端子設定		
GTCCRA機能	コンペアマッチ	~	]
コンペアマッチ 値 (GTCCRA)	99		
バッファ動作	バッファ動作しない	¥	
GTIOCOA端子機能	PWM出力端子	~	P23 (B) 🗸
🗌 ノイズフィルタ	PCLKA	~	
開始/停止時の出力レベル	開始時0出力、停止時0出力	~	]
コンペアマッチ時の出力レベル	0出力	~	]
周期の終わり時の出力レベル	1出力	~	]
-コンペアマッチレジスなGICCRB GIIC	COB端子設定		
GTCCRB機能	コンペアマッチ	~	
コンペアマッチ 値 (GTCCRB)	100		<i>.</i>
バッファ動作	パッファ 動作しない	~	
GTIOC0B端子機能	PWM出力端子	~	P17 (B) 🗸
🗌 ノイズフィルタ	PCLKA	~	
開始/停止時の出力レベル	開始時0出力、停止時0出力	~	
コンペアマッチ時の出力レベル	0出力	~	]
周期の終わり時の出力レベル	1出力	~	]

なおP17, P23は周辺機器→I/Oポート→(I/Oとしては)使用しない を選択する必要があり ます。

📓 r_cg_main.c 🔛 ブロバテイ	☑ 端子郎	置表 gr_og.gpt.c gg端子配置図 加速 间辺機能
🐻 コードを生成する   👗 🗋	a 🖄 🕷	■ ♣ ♣ 苯 💴 ⑧ ④ ◎ ◎ ④ ④ ⑨ ◎ 의 용 용 중 중 백 중 ଅ 🍕 🦄
Port0 Port1 Port2 Port	3 Port4 F	Port5   PortA   PortB   PortC   PortD   PortE   PortJ
-P20	~	
● 使用しない ○ 入力	● 出力	□ 内蔵ブルアップ CMOS 出力 · 1を出力 · 高駆動出力
- P21		
○ 使用しない ○ 入力	• 出力	内蔵ブルアップ CMOS 出力 🔹 🔽 1を出力 🗸 高駆動出力
- P22		
	<b>●</b> 出力	□ 内蔵グルアップ CMOS 出力 V □ 1を出力 Z 高駆動出力
	Сшл	
-P23		
● 使用しない ○ 人力 🤇	• 〇 出刀 (	■ 内蔵JルアップCMUS出力 V 「1を出力」高駆動出力
- P24		以下の端子と競合しています。この機能を使用する場合は競合する機能の設定を無効にしてください。
🔹 🔘 使用しない 🗋 入力	◉ 出力	P23はGTIOCOAで使われています。
- P25		
○ 使用しない ○ 入力	① 出力	□ 内蔵グルアップ CMOS 出力 V 1を出力 S駆動出力
D06	<u> </u>	
	<u>а</u> ш+	
	●五月	
-P27	~	
● 使用しない ● 入力	• 出力	内蔵ブルアップ CMOS 出力

# 【 プログラム 】

void main(void)

{

R\_MAIN\_UserInit();

/\* Start user code. Do not edit comment generated here \*/

 R\_SCI1\_Start();
 //SCI1スタート

 ①
 R\_GPT0\_Start();
 //汎用PWMタイマースタート

R\_SCI1\_Serial\_Receive(rx\_data,1); rx\_flg = 0; tx\_end\_flg = 0;

```
PORTC.PODR.BIT.B0 = 0;
```

//LED1 OFF

R\_SCI1\_Serial\_Send(String\_0,37);
tx\_end\_wait();

//Opening message //送信終了待ち

2	while (1U)	
	{	
	if(GPT0.C	GTCCRA != 599)
	{	
3		GPT0.GTCCRA++;
	}	
	else	
	{	
4		GPT0.GTCCRA = 0;
	}	
	if(GPT0.C	FTCCRB = 0
	{	
5		GPT0.GTCCRB;
	}	
	else	
	{	
6		GPT0.GTCCRB = 599;
	}	
~		
(7)	main_wai	t(100000);
	<u>`</u>	
	}	

/\* End user code. Do not edit comment generated here \*/
}

【 解説 】



従来のRXのイニシャルプログラム作成時はレジスタライトプロテクション有効、解除の頻繁な書き込み が必要でしたし、PWMを使うためのレジスタの詳細を掴んでからでないとプログラムが作成できません でしたが、このようにRL78と全く同じようにRXマイコンの開発が行える環境がそろいました。

# 2-6 三角、対数、平方根関数を使う

## 【 概要 】

| og、sin、√ 演算を行い、演算結果の確認とその速度を測定します。

### 【周辺機能の説明】

CS+側の設定はc-3↓をご参照ください。

c-3:三角関数math.hはインクルードもCS+の設定も必要

# 【 プログラム 】

) ()#include <math.h>

②double d1,d2,d3;
③short s1,s2,s3;

(4)#define PI 3.14159265

;

#### void main(void)

{

# R\_MAIN\_UserInit(); /\* Start user code. Do not edit comment generated here \*/

#### //重複省略

5	PORTC.PODR.BIT.B0 = 1;	//時間マーカーON
6	d1 = log10(10000);	
7	PORTC.PODR.BIT.B0 = 0;	//時間マーカーOFF
8	d2 = sin((PI/180)*45);	
	PORTC.PODR.BIT.B0 = 1;	//時間マーカーON
9	d3 = sqrt(2);	
	PORTC.PODR.BIT.B0 = 0;	//時間マーカーOFF
10	s1 = d1;	
	s2 = d2;	
	s3 = d3;	





32ビット浮動小数点データを16ビット整数にキャストしています。それぞれ、4,0,1となるはずです。例えば演算結果をDAコンバータに出力する場合、浮動小数点のままでは設定できません。小数点以下何桁まで使用したいかに応じて、doubleデータを加工してからshortに移せば最大の精度、 有効数値を得ることが出来ます。

74971					* *
🗷   🍭 🤽 🎒 🗙	表記(N)▼				
ウォッチ式	値	型情報(バイト数)	アドレス	×т	
🔍 tx_end_flg	'' (0x00)	uint8_t(1)	0x0000002d		
👻 rx_flg	'' (0x00)	uint8_t (1)	0x0000002c		
🖽 😔 rx_data		unsigned char…	0x00000004		
📷 SCI1.SCR	0x50	IOR(1)	0x0008a022		
📷 SCI1.SSR	0x84	IOR(1)	0x0008a024		
😜 d 1	4.000000E+000	float (4)	0×00000460		
😔 d2	7.071068E-001	float (4)	0×00000464		
😜 d3	1.414214E+000	float (4)	0x00000468		
😜 s1	4 (0x0004)	short(2)	0x00000054		
😜 s2	0 (0x0000)	short (2)	0×00000056		
😜 s3	1 (0x0001)	short (2)	0x00000058		

演算結果はそれぞれd1、d2、d3に入ります。正しいですね。

演算速度ですが、 | og 10(10000)が約550nsec、sin(45°)が350nsec、√ 2が100nsec程度かかるようでした。



TBS 1064 - 17:41:37 2016/09/29

例えばRL78(32MHz)では約220 $\mu$ sec、sin(45°)が130 $\mu$ sec、 $\sqrt{2}$ が100  $\mu$ sec程度ですので、それぞれ400倍、371倍、1000倍も速いことになります。マイコンに必 要な能力が演算処理速度の場合、RXを使用するのが圧倒的に有利であることが分かります。RX21A がRX71M、RX630に比べて遅いのはFPU内蔵、非内蔵の差と思われます。

or omgræden					
CPU クロッ	ク	log	sin	$\checkmark$	
RX71M 2	4 0 M H z	550nsec	350nsec	100nsec	
RX630 1	0 0 M H z	1. 8µsec	800nsec	1. 2 <i>µ</i> sec	
R X 2 1 A	5 0 M H z	ЗЗµѕес	2. 5µsec	Зμѕес	
RL78	3 2 M H z	220µsec	130µsec	1000 <i>µ</i> sec	

CPU別演算速度例

2-7 D/Aにsin演算した正弦波を出力する

#### 【 概要 】

RX71Mがもつ、1ch 12ビットD/Aにsin演算結果(最大±1)を0-3.3Vに変換し出 カします。正弦波オシレーターになります。



【周辺機能の説明】

D/Aコンバータの設定は以下の通りです。

【 プログラム 】

```
①unsigned short sin_data[370],kakudo;
;
void main(void)
£
②//演算して結果をメモリにセーブ
                 for(kakudo = 0;kakudo < 360 ;kakudo++)</pre>
                 {
                         d2 = sin((PI/180)*kakudo);
                                                   //1から-1まで変動
                         d2 +=1;
                                                   //オフセット+1→ 0~2の変化になる
                         d2 *= 2047.5;
                                                   //2を最大電圧3.3Vにする。
                                                   //sin信号をDAOUT P05
                         sin_data[kakudo] = d2;
                 }
③//演算結果をD/Aに出力
```

```
PORTC.PODR.BIT.B0 = 1; //LED1 ON
for(kakudo = 0;kakudo < 360 ;kakudo++)
{
DA.DADR1 = sin_data[kakudo]; //sin信号をDAOUT P03
}
PORTC.PODR.BIT.B0 = 0; //LED1 OFF
```

}

【解説】

省略

格納された配列のデータを1個ずつ読み込んでD/Aに出力しています。この方式で82KHz程度の正 弦波が作成できています。人間の可聴帯域程度であれば十分使用可能です。

それぞれはそれぞれの会社の登録商標です。

1. 本文章に記載された内容は弊社有限会社ビーリバーエレクトロニクスの調査結果です。

2. 本文章に記載された情報の内容、使用結果に対して弊社はいかなる責任も負いません。

3. 本文章に記載された情報に誤記等問題がありましたらご一報いただけますと幸いです。

4. 本文章は許可なく転載、複製することを堅くお断りいたします。

#### お問い合わせ先:

〒350-1213 埼玉県日高市高萩1141-1
TEL 042 (985) 6982
FAX 042 (985) 6720
Homepage: http://beriver.co.jp
e-mail: info@beriver.co.jp
有限会社ビーリバーエレクトロニクス (Beyond the river Inc. 20160930)