BCRX630__144 マイコン開発セット マニュアル

第1版2015.12.17

第1版

【 製品概要 】

本マニュアルはBCRX630_144 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。サンプル プログラムはルネサスエレクトロニクス社が無償で提供する統合開発環境CS+ for CCとCコン パイラを使用します。(HEW + Cコンパイラのサンプル、マニュアル(pdf)も同封されています)。 本CPUボード開発にはルネサスエレクトロニクス社製E1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:BCRX630_144 CPUボードの特徴
 - c:デバッカE1
 - 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D2XX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:ブレークポイント設定、レジスタ、変数参照概要
 - c:新しいプログラムを作る

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) でパソコンとやりとり
	2-2-1 sample21 SIO(RS232C)でパソコンとやりとり
	2-2-2 sample22 EEPROM(25LC256)読み書き
2 — 3.	sample3 A/D変換をUSB出力、RS232C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,cos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

RX630_144 CPUボード CD (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) ハードウエアマニュアル 電源ケーブル Kケーブル (RS232C用)



※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は同封されておりません。別途必要です。

b:BCRX630_144 CPUボードの特徴

●RXアーキテクチャコア(32ビットシングルチップCISC 最大165DMIPS 100MHz 動作時)、144ピン、R5F5630DDDFB搭載。

●32ビット単精度浮動小数点(IEEE754準拠)、2種類の積和演算器(メモリ間、レジスタ間)、 32ビット乗算器(最速1クロックで実行)、5段パイプラインのCISCハーバードアーキテクチャ ●外部クリスタルメイン12.5MHz(最大8逓倍100MHz動作)、サブ32.768KHz搭載。 ●大容量メモリ内蔵 FLASH 1.5MByte、RAM 128KByte

●コンパクト73×73mmサイズにUSB(FT232RL)、RS232C(ADM3202 2ch)、 EEPROM (25LC256 32KByte) IC搭載。

●動作電圧3.3V TYPE 50mA(100MHz動作時)

●豊富な周辺機能

I /Oポート:入出力117、入力1、内部プルアップ抵抗117、オープンドレイン117、5Vトレラント53、A/D変換器:12ビット、21ch、D/A変換器:10ビット2ch、リアルタイムクロック内蔵、シリアル(SCI):12ch、CANモジュール:3ch、外部バス拡張(16ビット)、DMA、強力なタイマ:MTU2(16bit×6ch)、ウオッチドグタイマ、コンペアマッチタイマ(16bit×2ch)、温度センサ等。

●デバッカE1によるデバック用コネクタ搭載。FINE接続。

基板大きさ



c:E1デバッカ



概要

E1 エミュレータは、ルネサス主要マイコンに対応したオンチップデバッギングエミュレータです。基本的 なデバッグ機能を有した低価格の購入しやすい開発ツールで、フラッシュプログラマとしても使用可能で す。

C 言語ソースデバックが可能で、1 行実行、ブレークポイント設定、変数、レジスタ、メモリ参照等々、従 来であれば高価な ICE しか出来なかった機能が、安価に実現されています。また、使い方もHEW(統合 開発環境)のE8a と同じで、経験があれば半日で、無くても1日で必要な操作を会得することが出来ると 思います。

マイコンとの通信として、シリアル接続方式(FINE)とJTAG 接続方式の2種類に対応しています。使用可能なデバッグインタフェースは、ご使用になるマイコンにより異なります。 また、基本デバッグ機能に加え、ホットプラグイン機能(動作中のユーザシステムに後から E1 エミュレー タを接続して、プログラムの動作確認を行うことが可能)を搭載しているため、プログラムのデバッグ・性能評価 に大きく貢献できます。

対応MPU

- V850 ファミリ
- RX ファミリ
- RL78 ファミリ
- R8C ファミリ
- 78K ファミリ



E1を購入するとCDが添付されていて、ドライバーのインストールとセルフチェックを行った後に、ネットから開発環境HEWとCコンパイラのダウンロードを行います。

E1/	E1/E20 USBドライバのインストール						
E	1またはE20エミュレータを	ホストマシンに接続する前に、E1/E20用のUSBドライバをインストールします。 🚅	<u>ンフォルダ</u> にあるEIUSBDRIVER.exe をクリックしてください。				
E1/	′E20のセルフチョ	ニック					
j t	ご購入後、製品が正しく動作することを、こ <u>のフォルダ</u> にあるE1/E20自己設断プログラムE1E20SCP-awを実行して確認してびざい。 また、ご使用中に起動プスN等の問題が発生した場合にも、本プログラムを使用して自己診断を行ってびざい。 セルプチェックプログラムの実施手順については、 <mark>こちら</mark> を参照がざい・、被撃による、修理、交換については、ルネサス販性または特別店にお問い合わせびざい。						
対応	なソフトウェアの・	インストール					
4	本製品を使用するためには、ご使用になるマイコンに含わせたソフトウェアをインストールする必要があります。						
	対応マイコン コンパイワ(必要) デパッガ(必要) プラッシュブロ ^{グラマ}						
	RXファミリ	RXファミリ用C/C++コンパン(ラバッケージ 無護計価度 (ダウンロードページへ) 無償評価版のダウンロードページへシャンプします。製品版をお持ちの方は製 版を使用してださい。	RXファミリ用エミュレータンフトウェア インストールブログラムを格納人にフォルダがオープンしますので、そのフォル ダにあるHewInstMan.evaを実行してください、インストールが始まります。	フラッシュ間部シールキット 度は2010 第一次シンコートペーシー) 無償評価版のダウンロードページへジャンプにます。 彩版を使用してがさい。			

c:無償版CS+、RX用Cコンパイラのダウンロード

プログラムの開発はルネサスエレクトロニクス社の統合開発環境HEWでC言語を用い動作させること ができます。CD添付のサンプルプログラムはこの環境下で作成されています。無償版をダウンロードし て使用します。

ネット検索で→「RXファミリコンパイラダウンロード」などで以下の画面を表示。

製品情報 :	アプリケーション	2 Fty Online	お問合も	セ /サポート	ご購入/サ		
ホーム 製品 開発	環境 コーディングツー	-ル コンパイラ/アセ	ンブラ				
RXファミリ用C/C++コンパイラパッケージ 更新動型論: 🕹 Share: 🗗 💟							
- 志山日					**P/++ -P		
		アンリケーションノートハリ			月午Q/リハート		
コーディングツール							
コンパイラ/アセンブラ	22件のうち1-10件を表示	しています。	表示件	數 10 🔻 💽	1 2 3 ≥		
RXフズリ用C/C++コンパイ ラパッケージ	◎ 分類	◎ソプトウェア名	◎ 登録日	說明	備考		
SuperH ファミリ用C/C++コ ンパイラパッケージ RH850ファミリ用Cコンパイラ パッケージ	統合開発環境 e² studi O	統合開発環境 e ^a studio 4.2 インストーラ (オンライン用)	Dec.15.15	Eclipseベースの統合開 発環境です。 コンパイラは別製品のた め、別途インストールが 必要です。			
V850ファミリ用Cコンパイラパ ッケージ V850用ソフトウェアパッケー ジ [SP850] M32Rファミリ用C/C++コン パイラパッケージ [M3T-	統合開発環境 e ² studi O	統合開発環境 e ² studio 4.2.0.012 イン ストーラ (オフライン用)	Dec.15.15	Eclipseペースの統合開 発環境です。 コンパイラは別製品のた め、別途インストールが 必要です。			
CC32R] RL78ファミリ用Cコンパイラパ ッケージ RL78、78Kファミリ用Cコンパ イラパッケージ R8C, M16Cファミリ用 C/C++コンパイラパッケージ 78K0R用ソフトウェアパッケ ージ [SP78K0] 78K0用ソフトウェアパッケ ジ [SP78K0] 78K0S用ソフトウェアパッケ	CS+ ()⊟CubeSuite+)	【無償評価版】統合開発 環境 CS+ for CC V3.02.00 (一括ダウン ロード版)	Oct.20.15	CS+ パッケージに含ま れるサブパッケージで す。 デバッガおよび無償評価 版二ンパイラを含みま す。 CubeSuite+からのアッ ブデートにも使用できま す。 対応マイコン: RH850フ ァミリ、RXファミリ、RL70 ファミリ	Б У З		
 -ジ [SP78K0S] 78K4用ソフトウェアパッケージ ジ [SP78K4] H8SX,H85,H87ァミリ用 C/C++コンパイラパッケージ 740ファミリ用Cコンパイラパッケージ ケージ [M3T-ICC740] 740ファミリ用アセンブラパッケージ ゲージ [M3T-SRA74] 4500シリーズ用アプソリュー 	CS+ ()⊟CubeSuite+)	【無償評価版】統合開発 環境 CS+ for CC V3.02.00 (分割ダウン ロード版)	Oct.20.15	CS+ パッケージに含ま れるサブパッケージで す。 デバッガおよび無償評価 版コンパイラを含みま す。 CubeSuite+からのアッ ブデートにも使用できま す。 対応マイコン: RH850フ	Б У		

CS+ Cコンパイラ同梱をダウンロードします。

無償版は60日経過後、リンクサイズが128KBと制限されます。[\] 統合開発環境CS+ for CCとCコンパイラがインストゥールされます。 d:開発セット添付CDコピー、デバイスドライバD2XXのインストゥール

事前にCDの中のホルダを例えばC:¥WrokSpace¥にコピーしてください。WorkSpaceはCS+をインストゥールすると自動形成されます。

名前	更新日時	種類	サイズ
🐌 RX630_CS+_sample	2015/12/17 17:05	ファイル フォル	
👢 RX630_HEW_sample	2014/03/13 17:30	ファイル フォル	
🐌 USBDRV	2014/02/25 11:41	ファイル フォル	
👢 Windows	2014/02/25 11:44	ファイル フォル…	
🗼 ドキュメント	2014/03/28 11:15	ファイル フォル	

RX630_CS+_sampleがCS+ for CC 用のサンプルソフトです。

初めて、BCRX630_144 CPUボードをパソコンにUSBミニケーブルで接続するとOSが FT232RLのデバイスドライバを要求してきます。WindowsXPの場合、下記の手順に従って デバイスドライバのインストゥールを行ってください。Windows7、8、10では自動検索されて、 CDを挿入する間がないものもあります。その結果、正常にインストゥールできなかった場合、コントロ ールパネル→システム→デバイスマネージャーで個別のデバイスマネージャーを開き、下記方法を参考に 再設定する必要があります。

以下省略

1-2 動作、デバック a:CS+起動、コンパイル、書き込み、動作



CS+ for CCを起動します。ここでは例としてRX630_CS+_sample¥sample1_ccを動作させます。E1はUSBでパソコンと接続され、E1とCPUボードのCN3がハーネスで接続して下さい。電源はE1から供給しますので電源ケーブルの接続は不要です。写真参照。



拡張子mtpjファイルをダブルクリック。



読み込み案内が表示され

進捗状況	
1	プロジェクトを読み込んでいます。
	キャンセル

例えば以下のように表示されます。



ここまで表示されない場合、E1のインストゥールが正常に終了していない可能性があります。再度確認 願います。

次に、「デバック・ツールヘプログラムをダウンロード」させます。

🔘 sample1_cc - CS+ for CC - [プロパティ]		
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(ト	4)
🆚 スタート(S) 📴 🔠 🐰 🖻 🖄 🕫 🕫 🖉 🙈	- 100% - 😽 😽 DefaultBuild	 ★ ★
		デバッグ・ツールヘプログラムをダウンロードします。

「CPUリセット後、プログラム実行」をクリック。

🖓 🗣 🐂 I 💷 🕑 🕨	🕽 🛞 ee Če 두 🙀
	CPUリセット後、プログラムを実行します。

基板上のLED D4が点滅したら、プログラムが動作しました。

▶RUN ⇔実行中 ■RX E1(Serial) (3計測中)

動作中は右下に「RUN」、実行中と表示されます。

LEDの点滅時間を変更してみます。2箇所のwaitの数値100000に0を追加してみます。

🚰 ਹੈਹ/ੴਰ 🍸 sample1_cc.c		
111 🛐 🔿 🤉 📭 カラム	•	
行 堂		
122	main_wait(1000000);	- /
123 124 125 126 127 128 129 130 131 132 133 134 135 136 137	PORTO.POOR.BYTE = 0xaa; PORT1.POOR.BYTE = 0xaa; PORT2.POOR.BYTE = 0xaa; PORT3.POOR.BYTE = 0xaa; PORT4.POOR.BYTE = 0xaa; PORT5.POOR.BYTE = 0xaa; PORT6.POOR.BYTE = 0xaa; PORT6.POOR.BYTE = 0xaa; PORT9.POOR.BYTE = 0xaa; PORT9.POOR.BYTE = 0xaa; PORT9.POOR.BYTE = 0xaa; PORT4.POOR.BYTE = 0xaa; PORT4.POOR.BYTE = 0xaa; PORT5.POOR.BYTE = 0xaa; PORT5.POOR.BYTE = 0xaa; PORT5.POOR.BYTE = 0xaa;	
138 139 140	PORTE.PODR.BYTE = 0xaa; PORTE.PODR.BYTE = 0xaa;	
141 142 143 144 145 146 147	PORTF.POOR.BYTE = 0xaa; PORTJ.POOR.BYTE = 0xaa; PORTK.POOR.BYTE = 0xaa; PORTL.POOR.BYTE = 0xaa; main_wait(1000000);	

セーブします。



🖓 🗅, 🐂 I 💿 🕟 🕟	🔊 🚳 Se Če Çe 🕅
	CPUリセット後、プログラムを実行します。

LEDの点滅が遅くなったでしょうか? 遅くなったら正常です。プログラムにエラーがあるとダウンロ ードされません。

以上のように、プログラム開発は「エディタ (プログラム作成)」→「セーブ」→「コンパイル」→「エラ ーが無いことを確認」→「書き込み」→結果によって頭の「エディタ」に戻る繰り返しになります。

エディタは使い慣れたものでも使用可能です。その場合、CS+のエディタは使えなくなります。

b:ブレークポイント設定、レジスタ、変数参照概要

以下省略

c:新しいプログラムを作る

以下省略

2. サンプルプログラム

2-1 sample1 出力ポートのON, OFF

/* */ /* FILE ∶sample1.c */ /* DATE :Tue, Feb 11, 2014 */ /* DESCRIPTION :Main Program */ /* CPU TYPE :RX630 */ /* */ /* This file is generated by Renesas Project Generator (Ver. 4.53). */ /* NOTE: THIS IS A TYPICAL EXAMPLE. */ /* */

#ifdef __cplusplus

//#include "typedefine.h"
#ifdef __cplusplus
//#include <ios> // Remove the comment when you use ios
//_SINT ios_base::Init::init_cnt; // Remove the comment when you use ios
#endif
void main(void);

```
extern "C" {
void abort(void);
}
#endif
②void main_wait(long ltime)
{
        while(ltime != 0)
        {
                ltime--;
        }
}
③void main(void)
{
  /* ---- Disable maskable interrupts ---- */
   clrpsw_i();
   /* ---- Stopping the peripherals which start operations ---- */
   R_INIT_StopModule();
   /* ---- Initialization of the non-existent ports ---- */
   R_INIT_NonExistentPort();
   /* ---- Initialization of the clock ---- */
   R_INIT_Clock();
(4)
        PORTO. PDR. BYTE = 0xff;
                                                          //全部出力 ポート検査用
        PORT1. PDR. BYTE = 0xff;
                                                          //全部出力 ポート検査用
        PORT2. PDR. BYTE = 0xff;
                                                          //全部出力 ポート検査用
        PORT3. PDR. BYTE = 0x3f;
                                                          //全部出力 ポート検査用
        //P36, 37 XTAL, EXTAL を出力ポートに設定してはいけない 外部 xtal 発振しなくなる。
        PORT4. PDR. BYTE = 0xff;
                                                          //全部出力 ポート検査用
        PORT5. PDR. BYTE = 0xff;
                                                          //全部出力 ポート検査用
        PORT6. PDR. BYTE = 0xff;
                                                          //全部出力 ポート検査用
        PORT7. PDR. BYTE = 0xff;
                                                          //全部出力 ポート検査用
        PORT8. PDR. BYTE = 0xff;
                                                          //全部出力 ポート検査用
        PORT9. PDR. BYTE = 0xff;
                                                          //全部出力 ポート検査用
        PORTA. PDR. BYTE = 0xff;
                                                          //全部出力 ポート検査用
```

PORTB. PDR. BYTE	=	0xff;
PORTC. PDR. BYTE	=	0xff;
PORTD. PDR. BYTE	=	0xff;
PORTE. PDR. BYTE	=	0xff;
PORTF. PDR. BYTE	=	0xff;
PORTJ. PDR. BYTE	=	0xff;
PORTK. PDR. BYTE	=	0xff;
PORTL. PDR. BYTE	=	0xff;

//全部出力 ポート検査用 //全部出力 ポート検査用

(5)	whi	le(1)	

{ ⑥

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;

PORTO. 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
PORTK.	PODR.	BYTE	=	0x55
PORTL.	PODR.	BYTE	=	0x55
	PORTB. PORTC. PORTD. PORTE. PORTF. PORTJ. PORTL.	PORTB. PODR. PORTC. PODR. PORTD. PODR. PORTE. PODR. PORTF. PODR. PORTJ. PODR. PORTK. PODR. PORTL. PODR.	PORTB. PODR. BYTE PORTC. PODR. BYTE PORTD. PODR. BYTE PORTE. PODR. BYTE PORTF. PODR. BYTE PORTJ. PODR. BYTE PORTK. PODR. BYTE PORTL. PODR. BYTE	PORTB. PODR. BYTE=PORTC. PODR. BYTE=PORTD. PODR. BYTE=PORTF. PODR. BYTE=PORTJ. PODR. BYTE=PORTK. PODR. BYTE=PORTL. PODR. BYTE=

 \bigcirc

main_wait(1000000);

8	PORTO. 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;
                PORTK. PODR. BYTE = 0xaa;
                PORTL. PODR. BYTE = 0xaa;
                main_wait(1000000);
   }
}
#ifdef __cplusplus
void abort(void)
{
}
#endif
 【解説】
本プログラムは、出荷検査のポート検査用に作られたもので、全てのポートをON, OFFさせています。
RX630を外部12.5MHz、8逓倍 100MHzで動作させています。
①#include "iodefine.h"
各レジスタの定義が入っているヘッダです。
②void main_wait(long ltime)
{
        while(ltime != 0)
        {
                ltime--;
        }
```

}

```
13
```

メイン関数で使用するLEDのON、OFF時間を設定するウエイトです。

③void main(void) { メインルーチンです。

(4) PORTO. PDR. BYTE = 0xff;

//全部出力 ポート検査用

ディレクション(方行)レジスタに全ビットに1を立てて、全て出力にしています。0を設定すると入力 になります。

(5) while(1)

{ ここから無限ループです。

6 PORTO. PODR. BYTE = 0x55;
 0 x 5 5 を出力しています。0 x 5 5 = 0 b 0 1 0 1 0 1 0 1 です。1ビットおきに1を立てています。

⑦ main_wait(1000000);
 ウエイトです。1000000 が0になるまで戻ってきません。

(8) PORTO. PODR. BYTE = 0xaa;

0 x a a を出しています。2進数で書くと0 b 0 1 0 1 0 1 0 1 0 です。前が0 x 5 5 でしたが2進数表 記では0 b 0 1 0 1 0 1 0 1 となります。お互いを反転させた値をポートに書き込んでいます。

 $0 \times a a = \sim 0 \times 55;$

出荷検査は断線、接触を検査しますが、断線は信号が振れないことで検出できます。接触は例えば隣のポート同士が接触していて、この数値を出力すると、どちらかは必ずOで、ハード的に出力がOに収束されるため、信号が上下に振れません。それでエラーが分かるという考え方です。単なるポートのON, OF FであればOxff、Oを交互に書き込めばよいのですが、それでは隣との接触が検出出来ないというわけです。

2-2 sample2 SIO (USB) でパソコンとやりとり /* */ /* FILE */ ∶sample1.c /* DATE :Tue, Feb 11, 2014 */ /* DESCRIPTION :Main Program */ /* CPU TYPE :RX630 */ /* */ /* This file is generated by Renesas Project Generator (Ver. 4.53). */ /* NOTE: THIS IS A TYPICAL EXAMPLE. */ /* */ /* SIO (USB) でパソコンとやりとり 【動作】 フリーのターミナルソフト「テラターム」と本基版をUSBケーブルで接続し、やりとりを行います。 【 接続 】添付のUSBミニケーブルで本基版とパソコンを接続して下さい。 【 事前設定 】E1プローブをRX630_144 CPUボードのCN3に挿入する。 【 注意 】「テラターム」はダウンロード、インストゥールしてください。使い慣れたターミナルソフト があればそれで結構です。FTDI FT232RLのデバイスドライバは事前にインストゥールしてく ださい。速度等は9600bps、データ8ビット、パリティノン、ストップ1です。 */ #include <machine.h> #include "iodefine.h" #include "r_init_clock.h" #include "r_init_non_existent_port.h" #include "r_init_stop_module.h" #include "sio RX630.c" unsigned char cf; #define CR 0x0d #define LF 0x0a void main(void) { /* ---- Disable maskable interrupts ---- */ clrpsw_i(); /* ---- Stopping the peripherals which start operations ---- */ R_INIT_StopModule();

/* ---- Initialization of the non-existent ports ---- */
R_INIT_NonExistentPort();
/* ---- Initialization of the clock ---- */

R_INIT_Clock();

//SIO6 USB 動作開始

<pre>② char_out6(CR); char_out6(LF); char_out6('R'); char_out6('X');</pre>	//キャリッジリタ //ラインフィート //USB	マーン
<pre>char_out6(UF); char_out6(LF); char_out6('R'); char_out6('X');</pre>	//キャリノンリン //ラインフィート //USB	
char_out6('R'); char_out6('X');	//USB	
char_out6('X');		1 文字出力
	//USB	1 文字出力
char_out6('6');	//USB	1 文字出力
char_out6('3');	//USB	1 文字出力
char_out6('0');	//USB	1 文字出力
char_out6('');	//USB	1 文字出力
char_out6('U');	//USB	1 文字出力
char_out6('S');	//USB	1 文字出力
char_out6('B');	//USB	1 文字出力
char_out6(CR);	//USB	1 文字出力
char_out6(LF);	//USB	1 文字出力

	while(1)			
	{			
3		cf = char_in6();	//USB	パソコンから 1 文字入力
		char_out6(cf);	//USB	パソコンへ1文字出力
	}			

}

【 解説 】

事前準備として、本基版とパソコンを添付のUSBミニケーブルで接続して下さい。基板に電源が入り、 しばらくすると仮想COMの番号が表示されますので、それでテラタームを立ち上げて下さい。 下記例ではCOM43番。9600bpsです。

	Tera Term: 新しい	接続	
	© TCP/IP	たスト(T): <mark>BRE-DC1 □ ピストリ(O)</mark> ナービス: ○ Telnet TCPが ◎ SSH SSH/ 「ージョ ● その他 プロトコ/	 ►#(P): 22 ✓(V): SSH2 ↓L(C): UNSPEC
	● シリアル	 K—– h(R): COM16: BT Port (CC COM16: BT Port (CC COM17: BT Port) (CO COM17: BT Port) (CO COM18: BT Port (CO COM21: BT Port) (CO COM21: BT Port) (CO COM21: BT Port) (CO 	M16) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M17) (M1
Install C	₽ ++Builder6 pcbe	COM23: BT Port (CO COM23: BT Port (CO COM24: BT Port (CO COM25: BT Port (CO >∋=−17COM40: BT Port (CO	M23) M24) M25) Zv2.61 readst

sample2を動作させると画面にRX630 USBと表示されます。パソコンのキーボードで押した文字や数字がパソコンから本基版に送信され、基板側はその文字を返信して、画面に表示されます。



① init_rx630_sio();

//SIO(USB)イニシャル

SIOの初期化を行っています。関数はsio_RX630. cの中にあります。RXの命令の特徴であるレジスタライトプロテクション命令などを使用して作成してあります。詳細は興味のある方向けに最後に示します。

(2) char_out6(CR);

//キャリッジリターン

テラターム画面上に「RX630 USB」と表示させています。前後に改行、ラインフィードを入れて います。

 3 cf = char_in6(); char_out6(cf);
 1文字受信して、それを送信しています。 //USB パソコンから1文字入力
//USB パソコンへ1文字出力

init_rx630_sio()関数の詳細 void init_rx630_sio(void) { //SI01 動作開始

(1) SYSTEM. PRCR. WORD = 0xA502; SYSTEM. MSTPCRB. BIT. MSTPB31 = 0; SYSTEM. MSTPCRB. BIT. MSTPB28 = 0; SYSTEM. MSTPCRB. BIT. MSTPB25 = 0; SYSTEM. PRCR. WORD = 0xA500;

//

(2) MPC. PWPR. BIT. BOWI = 0; //PFS レジスタ書き込み 1
 MPC. PWPR. BIT. PFSWE = 1; //PFS レジスタ書き込み 2許可

//SCI0

MPC. P20PFS.BIT.PSEL = 0xa; MPC.P21PFS.BIT.PSEL = 0xa;

//SCI3

MPC. P25PFS. BIT. PSEL = 0xa; MPC. P23PFS. BIT. PSEL = 0xa;

//SCI6

(4)

MPC. POOPFS. BIT. PSEL = 0xa; クション

MPC. PO1PFS. BIT. PSEL = 0xa;

MPC. PWPR. BIT. PFSWE = 0; MPC. PWPR. BIT. BOWI = 1;

③ PORT2. PMR. BYTE = 0x2b; PORT0. PMR. BYTE = 0x03;

//SCI0 RS232C_0 CN4

SCIO. SCR. BYTE = 0; //クリア SCIO. BRR = 162; //ボーレート (50MHz/32*BPS)-1 //9600≒162, 38400≒40, 115200≒13, 230400≒7, これ以上は駄目でした

SCIO.SMR.BYTE = 0; //8bit,1-stop,parity-none,async SCIO.SCR.BYTE = 0x70; //シリアルコントロールレジスタクリア、内部ボーレート

//IPR[219] フラグを立てるため、RIE をアクティブにしている

//レジスタライトプロテクション解除 PRC1

//SI00 モジュールストップ解除

//SI03 モジュールストップ解除

//SI06 モジュールストップ解除

//レジスタライトプロテクション有効

//RXDO 端子割り振り

//TXD0 端子割り振り

//RXD3 端子割り振り

//TXD3 端子割り振り

//RXD6 端子割り振り

//PFS レジスタ書き込み1禁止

//PFS レジスタ書き込み2禁止

//周辺装置

//周辺装置

//TXD6 端子割り振り マルチピンファン

//SCI3 RS232C_3 CN5

	SCI3. SCR. BYTE = 0;	//クリア			
	SCI3.BRR = 162;	//ボーレート(50MH z /32*BPS)−1			
	//9600≒162, 38400≒40, 115200≒13, 230400≒7, これ以上は駄目でした				
	SCI3. SMR. BYTE = 0;				
	//8bit,1-stop,parity-none,async				
	SCI3. SCR. BYTE = $0x70$;	//シリアルコントロールレジスタクリア、内部ボーレート			
		//IPR[219]フラグを立てるため、RIE をアクティブにしている			
//SCI6	USB CN8				
	SCI6. SCR. BYTE = $0;$	//クリア			
	SCI6. BRR = 162 ;	//ボーレート(50MH z /32*BPS)−1			
	//9600≒162, 38400≒40, 11	5200≒13,230400≒7,これ以上は駄目でした			
	SCI6. SMR. BYTE = $0;$				
	//8bit, 1-stop, parity-none, async				
	SCI6. SCR. BYTE = $0x70$;	//シリアルコントロールレジスタクリア、内部ボーレート			
}		//IPR[227]フラグを立てるため、RIE をアクティブにしている			

【解説】



2-6 三角、対数、平方根関数を使う

/* */ /* FILE ∶sample6.c */ /* DATE :Tue, Feb 18, 2014 */ /* DESCRIPTION :Main Program */ /* CPU TYPE :RX630 */ /* */ /* This file is generated by Renesas Project Generator (Ver.4.53). */ /* NOTE: THIS IS A TYPICAL EXAMPLE. */ /* */

/*

三角関数

【 動作 】log、sin、ルートの演算を行い、double に浮動小数点格納されます。数値と演算 時間を確認します。

【 接続 】特になし
 【 事前設定 】特になし
 【 注意 】特になし

*/

#include <machine.h> #include ~iodefine.h~ #include ~r_init_clock.h~ #include ~r_init_non_existent_port.h~ #include ~r_init_stop_module.h~ #include ~sio_RX630.c~ ()#include <math.h> (2)double d1, d2, d3: short s1, s2, s3; (3)#define PI 3.14159265 #define CR 0x0d #define LF 0x0a

省略

void main(void) { **省略(既説)**

4 PORT7. PODR. BIT. B0 = 1; //時間マーカーON $d1 = \log 10 (10000);$ PORT7. PODR. BIT. BO = 0; //時間マーカーON PORT7. PODR. BIT. BO = 1; //時間マーカーON d2 = sin((PI/180)*45);PORT7. PODR. BIT. B0 = 0; //時間マーカーON PORT7. PODR. BIT. BO = 1; //時間マーカーON d3 = sqrt(2);PORT7. PODR. BIT. B0 = 0; //時間マーカーON 5 s1 = d1; s2 = d2;s3 = d3;

while (1U)

- {
- ſ

}

}

【 解説 】

(1)#include <math.h>

三角関数や、対数、平方根を使うためにはmath.hをインクルードする必要があります。加えて

🗿 sample6_cc - CS+ for CC - [プロ	パティ]	
ファイル(F) 編集(E) 表示(V) プロジ	^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)	
🚳 スタート(S) 📮 🗃 🔉 🐁	B ウ ペ 器 単 単 ・ 100% - 夏 夏 夏 DefaultBuild - ★ 夏 □, い ◎ ● ○ い ④ □ = 0 	ja c₂ ja
プロジェクト・ツリー 🛛 🕈 🗙	□ プロ/行イ I Sample6 cc.c	
2 🕜 🙎 🔳		
B- Sample6 cc (プロジェクト)		
		檀進ライブラリ・ファイル作成(オプ)
	- 標準ライブラリ	
CC-RX (EILF·9-IL)	ライブラリ構成	C(C89)(-lang=c)
	構築対象のライブラリ	力スタム(-head= <suboption>)</suboption>
☆ 🖾 ファイル	ランタイム・ライブラリを有効にする	はい(-head=runtime)
dbsct c	ctype.h(C89/C99)を有効にする	いいえ
	math h(CB9/C99)を有効にする	(at ()(-head=math)
intprg.c	mathf.h(289/299)を有刻にする	いいえ
- Setprg.c	stdarg,h(CB9/C99)を有以に9.6	(t)()Z
	statio IN (397/099)を有効にする	(#()(=head=stdio)
vecttbl.c	string (C897/C997)2 47 x/IC 90 string (C897/C997)2 47 x/IC 90	(4.(.)(-head-string)
indofina h	ins (FR)+1/k addic to A	()()ž
	new(EC++)を有効にする	(t()(-head=new)
🖼 sbrk.h	complex(EC++)を有効にする	いいえ
	string(EC++)を有効にする	いいえ
typedefine.h	▲ オブジェクト	
west h	出力フォルダ	%BuildModeName%
	出力ファイル名	%ProjectName%.lib
- T_INIT_CIOCK.C	機能縮小版人出力関数を生成する	()()ž
- 🖆 r_init_non_existent_po	フロクラム視域のセクション名	P
🚽 r init stop module.c	た数理機のセンタイン	0
sample6 cc c	10月11日 「フロックリンコン」を	B
	本が効応して、実界数のビクションで	
	switch 文分岐テレゴル領域のセガション名	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	(J(J)Ž
	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
	conct的就変動をプライイントルのわりについて記録する	11117
	標準ライブラリの使用・構築方法 標準ライブラリの使用・構築方法を選択します。	
	↓ \ 共通オブション / コンパイル・オブション / アセンブル・オブション / リンク・オブション / ヘキサ出力オブション / ラ イブラリ・ジェネレ ・	- ト・オプ ション /

以下省略

一度「リセット後実行」させ、その後、停止させると右上のワッチウインドウに演算結果が表示されます。

ľ	ノオツナ1				4	×
	🗷 🧶 🗏 🛞 🗙	表記(N)- 👼				
	ウォッチ式	値	型情報(バイト数)	アドレス	7-	ŧ
	🔍 d1	4.000000E+000	float (4)	0x00000418		
	🔍 d2	7.071068E-001	float (4)	0x0000041c		
	😜 d3	1.414214E+000	float (4)	0×00000420		
	😜 s1	4 (0x0004)	short(2)	0×00000004		
	😜 s2	0 (0x0000)	short(2)	0×00000006		
	😜 s3	1 (0x0001)	short(2)	0×00000008		

上からd1、d2、d3の10進数倍精度データです。答えは合っていますね。

(5) s1 = d1; s2 = d2; s3 = d3;

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。⑤は浮動小数 点データを整数の16ビットに設定(キャスト)しています。1000番地からs1, s2, s3です。 結果を見ると小数点部分が欠落して設定されていることが分かると思います。(上図 下 s1、s2、s 小数点以下何桁まで使用したいか、ということで、doubleデータを加工してからshortに移せば最大の精度、有効数値が得られます。

P70の端子をオシロで観測すると、図のような波形が得られます。 $| og の演算が約1.8 \mu sec$ 、 sinが0.8 μ sec、平方根が1.2 μ secくらいでしょうか。



以下はHEW+Cコンパイラの場合なのですが、



TDS 2012 - 18:58:24 2014/03/05

P 7 0 の端子をオシロで観測すると、図のような波形が得られます。 | o g の演算が約40 μ s e c 、 s i n が 2 6 μ s e c 、平方根が 5 μ s e c くらいでしょうか。

I o g で 2 2 倍違います! 同じ C P U ボード(動作クロック 1 0 0 M H z) でにわかに信じられないような速度の違いが開発環境、Cコンパイラの違いで出ました。

32MHz動作のRL78104マイコンで | og10(10000)が約220 μ sec、sin(4 5°)が130 μ sec、√2が100 μ sec程度かかるので、演算に関してRXはクロック比以上の 劇的な速さが得られるのが分かります。

3)

※RXは倍精度演算の速度です。RL78は単精度演算の速度です。 ※RL78の演算、ポート制御は従来のH8、R8Cマイコン等に比べて何倍も高速です。速度比較詳細 は無償ダウンロード出来る弊社「RL78104の開発セットマニュアル抜粋」でご確認下さい。



RX630は分解能10ビットD/A出力を2ch持っています。そこにsin(), cos()の0~3 60°を演算し、D/A出力し、電圧をみてみます。いわゆる、正弦波発振器と同じ出力が得られます。

以下省略

演算とDA出力を分離したことにより、



25.1387kHzという正弦(sin)波、余弦(cos)波が得られました。この周波数はクリス タルの精度で、極めて安定しています。周波数を低くするには1データ出力毎にウエイトを入れることで 可能です。人間の可聴帯域はほぼカバーすることが出来ます。CR発振器が苦手な超低周波信号も高精度、 高安定で作成できます。 WindowsXP®、WindowsVist®、Windows7®はマイクロソフト社の登録商標です。 フォース®は弊社の登録商標です。

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. 20151218