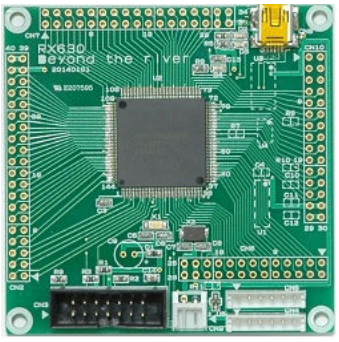
ＢＣＲＸ６３０＿１４４　マイコン開発セット　マニュアル

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

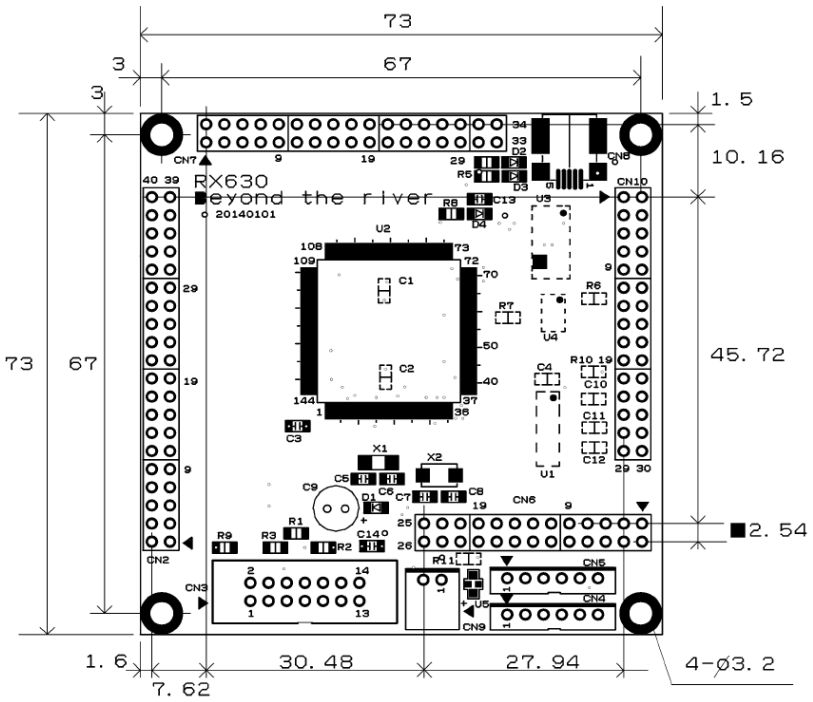
ＲＸ６３０＿１４４　ＣＰＵボード
ＣＤ（サンプルプログラム、デバイスドライバ、ドキュメント）
マニュアル（本誌）
ハードウエアマニュアル
電源ケーブル
Ｋケーブル（ＲＳ２３２Ｃ用）

※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。

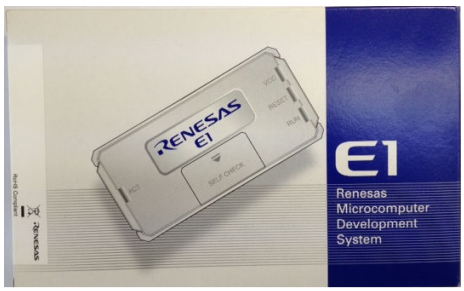
ｂ：ＢＣＲＸ６３０＿１４４　ＣＰＵボードの特徴

●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/E20 USBドライバのインストール

E1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。このフォルダにあるE1USBDRIVER.exeをクリックしてください。

E1/E20のセルフチェック

ご購入後、製品が正しく動作することを、このフォルダにあるE1/E20自己診断プログラムE1E20SCP.exeを実行して確認してください。
また、ご使用中に起動しない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ください。

セルフチェックプログラムの実施手順については、こちらを参照ください。故障による、修理、交換については、ルネサス販社または特約店にお問い合わせください。

対応ソフトウェアのインストール

本製品を使用するためには、ご使用になるマイコンに合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	コンパイラ(必要)	デバッガ(必要)	フラッシュプログラマ (任意)
RXファミリ	RXファミリ用C/C++コンパイラパッケージ 無償評価版 [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ジャンプします。製品版をお持ちの方は製品版を使用してください。	RXファミリ用エミュレータソフトウェア インストールプログラムを格納したフォルダがオープンしますので、そのフォルダにあるHewInstMan.exeを実行してください。インストールが始まります。	フラッシュ開発ツールキット 無償評価版 [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ジャンプします。製品版をお持ちの方は製品版を使用してください。

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。

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。

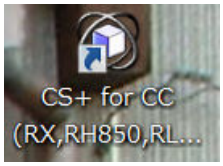
ＲＸ６３０＿ＣＳ＋＿ｓａｍｐｌｅがＣＳ＋　ｆｏｒ　ＣＣ　用のサンプルソフトです。

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

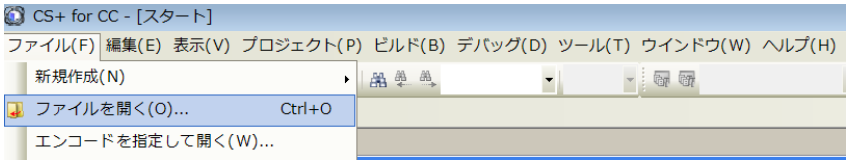
１－２　　動作、デバック

ａ：ＣＳ＋起動、コンパイル、書き込み、動作

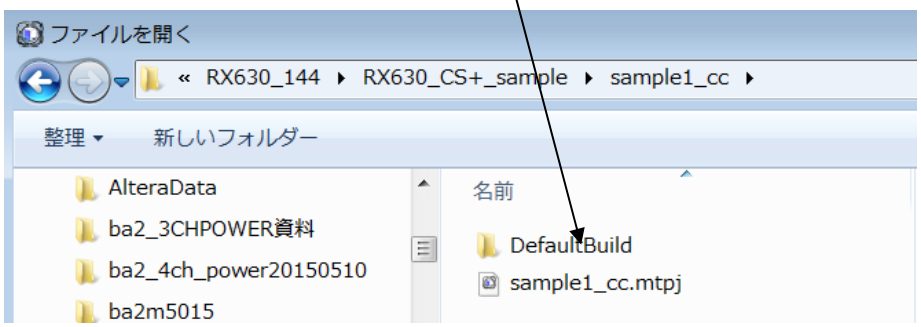


ＣＳ＋　ｆｏｒ　ＣＣを起動します。ここでは例としてＲＸ６３０＿ＣＳ＋＿ｓａｍｐｌｅ￥ｓａｍｐｌｅ１＿ｃｃを動作させます。Ｅ１はＵＳＢでパソコンと接続され、Ｅ１とＣＰＵボードの ＣＮ３がハーネスで接続して下さい。電源はＥ１から供給しますので電源ケーブルの接続は不要です。写真参照。

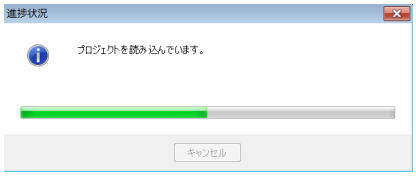
ファイル→ファイルを開く



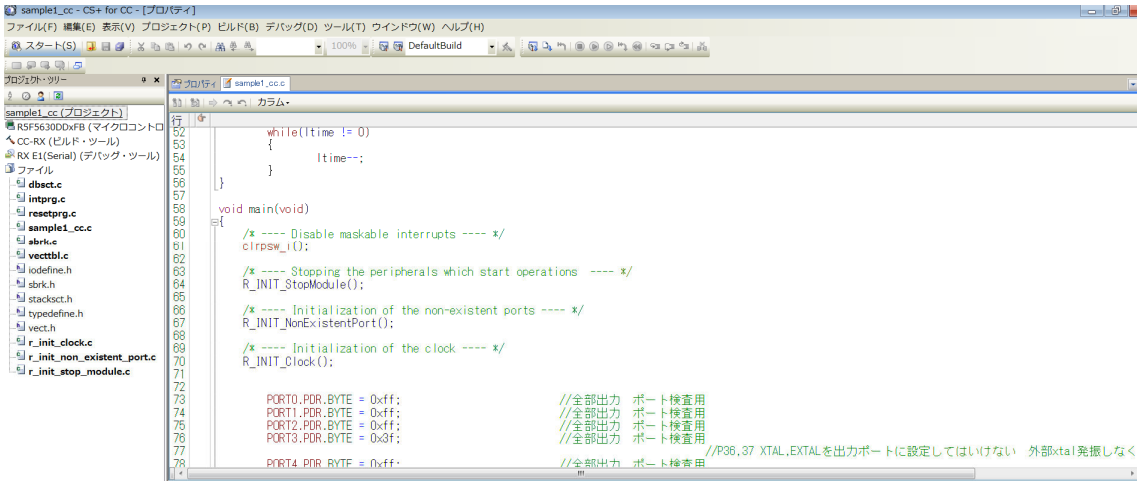
拡張子ｍｔｐｊファイルをダブルクリック。



読み込み案内が表示され



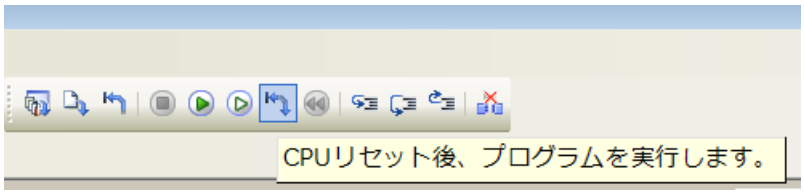
例えば以下のように表示されます。



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・ツールへプログラムをダウンロード」させます。

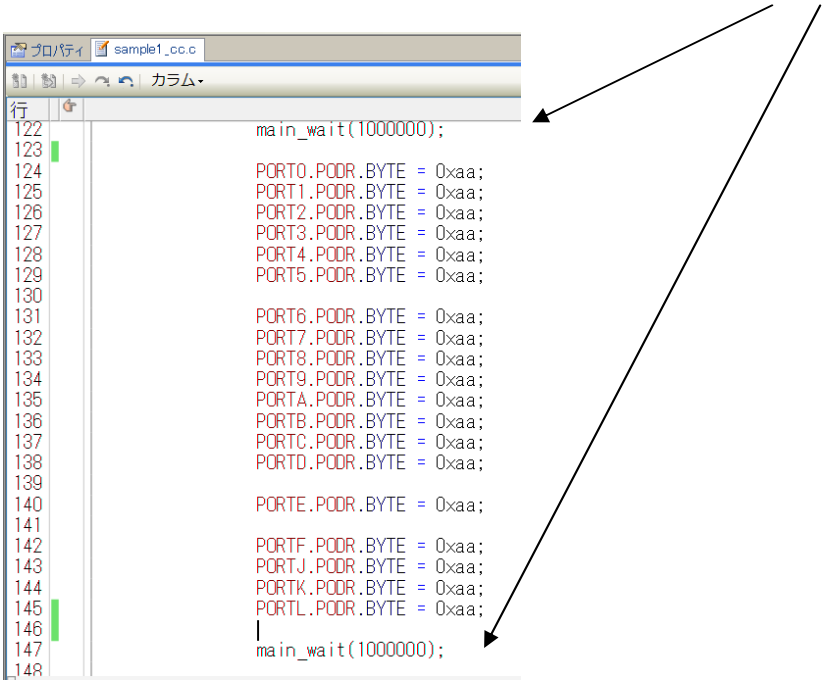
「ＣＰＵリセット後、プログラム実行」をクリック。



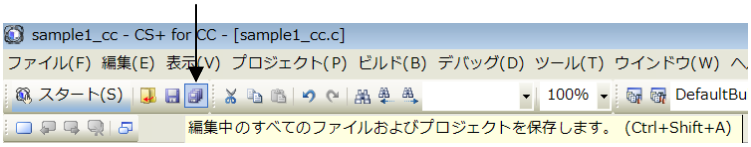
基板上のＬＥＤ　Ｄ４が点滅したら、プログラムが動作しました。

動作中は右下に「ＲＵＮ」、実行中　と表示されます。

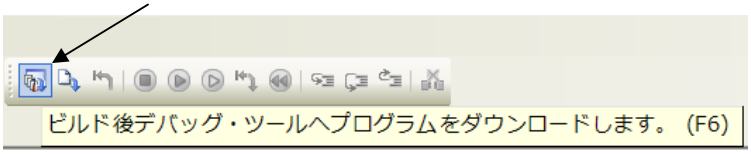
ＬＥＤの点滅時間を変更してみます。２箇所のｗａｉｔの数値１０００００００に０を追加してみます。



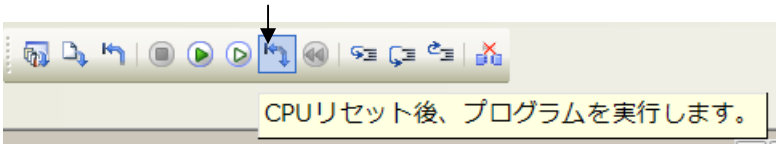
セーブします。



コンパイル＆ダウンロードします。



リセット＆実行して下さい。



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.                                    */
/*                                                                     */
/***********************************************************************/


#include <machine.h>
①#include "iodefine.h"
#include "r_init_clock.h"
#include "r_init_non_existent_port.h"
#include "r_init_stop_module.h"



//#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);
#ifdef __cplusplus
```

```
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       PORT0.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;                          //全部出力　ポート検査用
    PORTC.PDR.BYTE = 0xff;                          //全部出力　ポート検査用
    PORTD.PDR.BYTE = 0xff;                          //全部出力　ポート検査用
    PORTE.PDR.BYTE = 0xff;                          //全部出力　ポート検査用

    PORTF.PDR.BYTE = 0xff;                          //全部出力　ポート検査用
    PORTJ.PDR.BYTE = 0xff;                          //全部出力　ポート検査用
    PORTK.PDR.BYTE = 0xff;                          //全部出力　ポート検査用
    PORTL.PDR.BYTE = 0xff;                          //全部出力　ポート検査用


⑤   while(1)
  {
⑥             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;
              PORTK.PODR.BYTE = 0x55;
              PORTL.PODR.BYTE = 0x55;

⑦             main_wait(1000000);


⑧             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;
                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--;
        }
}
```

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　　PORT0.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　　PORT0.PODR.BYTE = 0x55;
０ｘ５５を出力しています。０ｘ５５＝０ｂ０１０１０１０１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　　PORT0.PODR.BYTE = 0xaa;
０ｘａａを出しています。２進数で書くと０ｂ０１０１０１０１です。前が０ｘ５５でしたが２進数表記では０ｂ０１０１０１０１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。

０ｘａａ　＝　～０ｘ５５；

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０で、ハード的に出力が０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，ＯＦＦであれば０ｘｆｆ、０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。

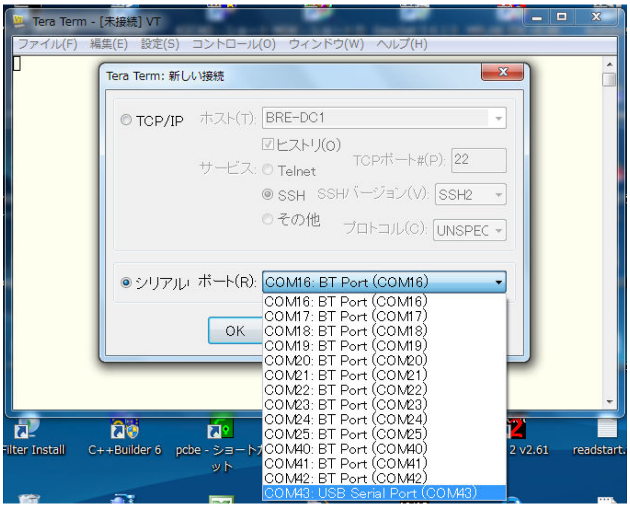
２－２　ｓａｍｐｌｅ２　ＳＩＯ（ＵＳＢ）でパソコンとやりとり

```
/***********************************************************************/
/*                                                                     */
/*  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();
```

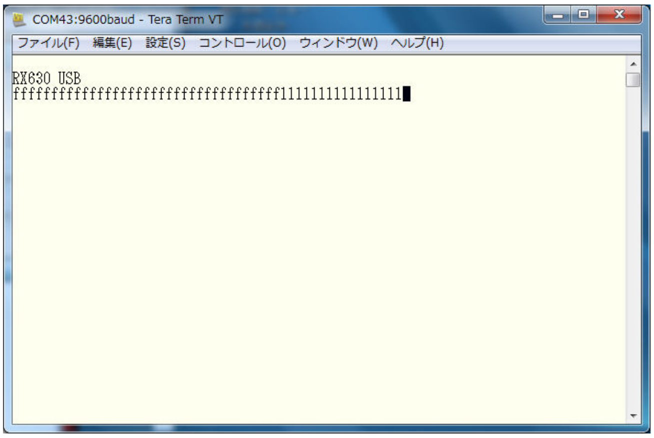
```
    /* ---- Initialization of the non-existent ports ---- */
    R_INIT_NonExistentPort();

    /* ---- Initialization of the clock ---- */
    R_INIT_Clock();


//SIO6 USB 動作開始

①       init_rx630_sio();                                   //SIO(USB)イニシャル


②       char_out6(CR);                                      //キャリッジリターン
        char_out6(LF);                                      //ラインフィード
        char_out6('R');                                             //USB  1 文字出力
        char_out6('X');                                             //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。



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();　　　　　　　//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

１文字受信して、それを送信しています。

ｉｎｉｔ＿ｒｘ６３０＿ｓｉｏ（）関数の詳細

```
void init_rx630_sio(void)
{
//SIO1 動作開始

①       SYSTEM.PRCR.WORD = 0xA502;          //レジスタライトプロテクション解除　PRC1
        SYSTEM.MSTPCRB.BIT.MSTPB31 = 0;    //SIO0 モジュールストップ解除
        SYSTEM.MSTPCRB.BIT.MSTPB28 = 0;    //SIO3 モジュールストップ解除
        SYSTEM.MSTPCRB.BIT.MSTPB25 = 0;    //SIO6 モジュールストップ解除
        SYSTEM.PRCR.WORD = 0xA500;         //レジスタライトプロテクション有効

//
②       MPC.PWPR.BIT.B0WI = 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;                  //RXD0　端子割り振り
        MPC.P21PFS.BIT.PSEL = 0xa;                  //TXD0　端子割り振り

//SCI3
        MPC.P25PFS.BIT.PSEL = 0xa;                  //RXD3　端子割り振り
        MPC.P23PFS.BIT.PSEL = 0xa;                  //TXD3　端子割り振り

//SCI6
        MPC.P00PFS.BIT.PSEL = 0xa;                  //TXD6　端子割り振り　マルチピンファン
クション
        MPC.P01PFS.BIT.PSEL = 0xa;                  //RXD6　端子割り振り

        MPC.PWPR.BIT.PFSWE = 0;            //PFS レジスタ書き込み１禁止
        MPC.PWPR.BIT.B0WI = 1;             //PFS レジスタ書き込み２禁止

③       PORT2.PMR.BYTE = 0x2b;                      //周辺装置
        PORT0.PMR.BYTE = 0x03;                      //周辺装置

//SCI0  RS232C_0 CN4
        SCI0.SCR.BYTE = 0;                          //クリア
④       SCI0.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
        SCI0.SMR.BYTE = 0;
        //8bit,1-stop,parity-none,async
        SCI0.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

//SCI3  RS232C_3 CN5
```

```
        SCI3.SCR.BYTE = 0;                          //クリア
        SCI3.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
        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 (50MHｚ／32*BPS)-1

        //9600≒162,38400≒40,115200≒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　】

以下省略

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"
①#include <math.h>

②double d1,d2,d3;
short s1,s2,s3;

③#define PI 3.14159265

#define CR 0x0d
#define LF 0x0a
```

省略

```
void main(void)
{
```

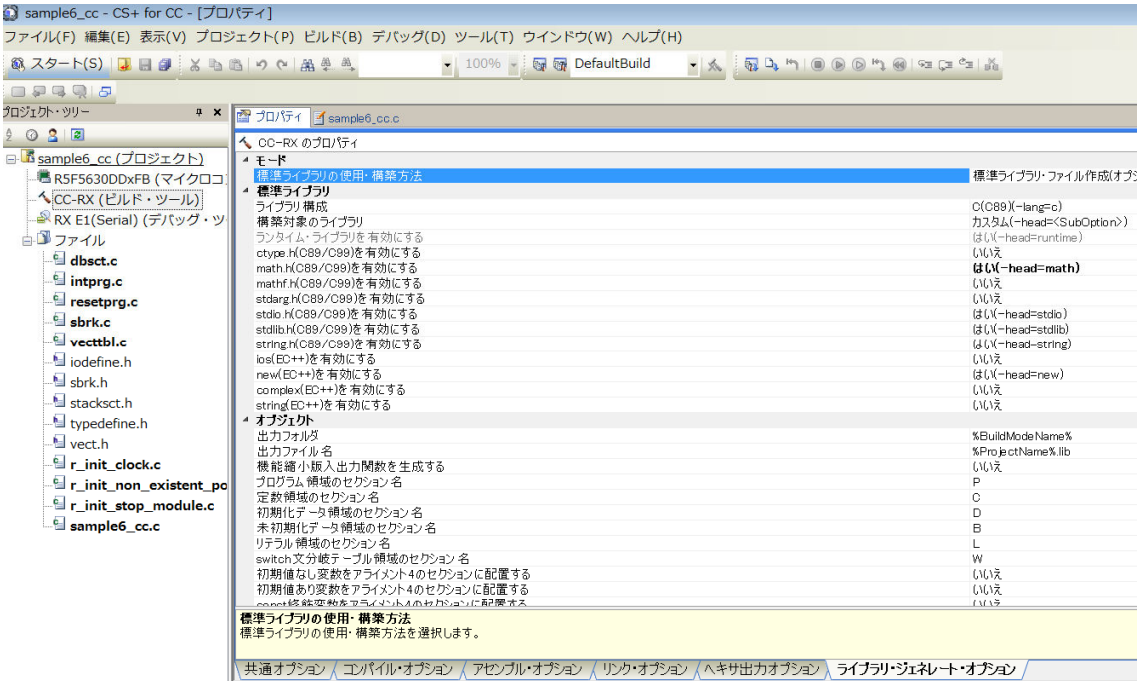
省略（既説）

```
④                   PORT7.PODR.BIT.B0 = 1;              //時間マーカーON
                    d1 = log10(10000);
                    PORT7.PODR.BIT.B0 = 0;              //時間マーカーON

                    PORT7.PODR.BIT.B0 = 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.B0 = 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)
    {


    }

}
```

【　解説　】

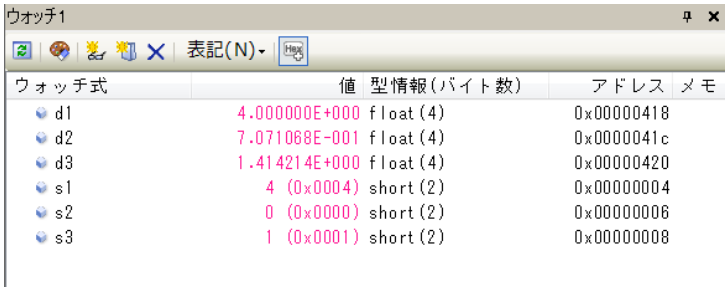
①#include <math.h>

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



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。



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。

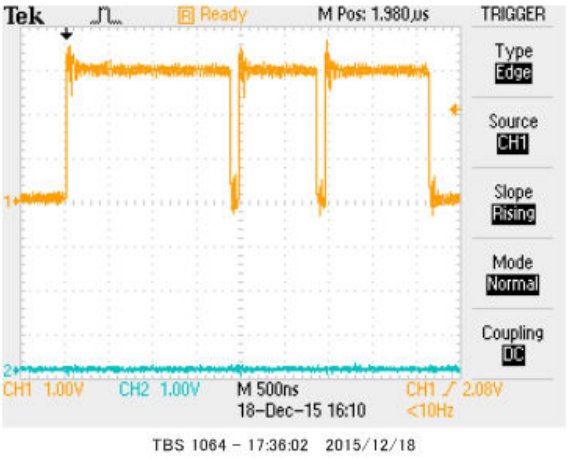
⑤
```
s1 = d1;
s2 = d2;
s3 = d3;
```

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

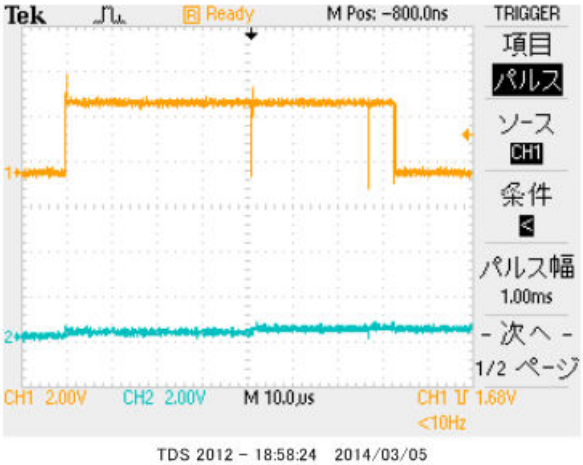
３）

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。ｌｏｇの演算が約１．８μｓｅｃ、ｓｉｎが０．８μｓｅｃ、平方根が１．２μｓｅｃ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、



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。ｌｏｇの演算が約４０μｓｅｃ、ｓｉｎが２６μｓｅｃ、平方根が５μｓｅｃくらいでしょうか。

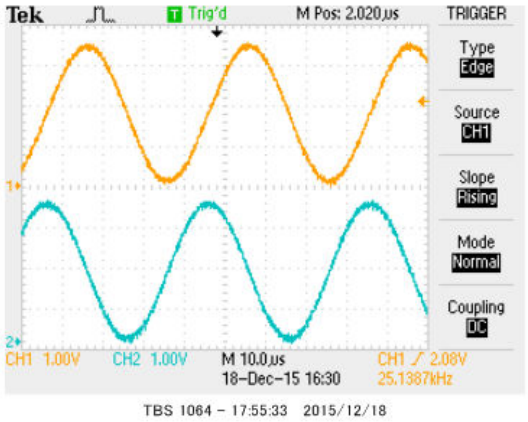
ｌｏｇで２２倍違います！　同じＣＰＵボード（動作クロック１００ＭＨｚ）でにわかに信じられないような速度の違いが開発環境、Ｃコンパイラの違いで出ました。

３２ＭＨｚ動作のＲＬ７８１０４マイコンでｌｏｇ１０（１００００）が約２２０μｓｅｃ、ｓｉｎ（４５°）が１３０μｓｅｃ、√２が１００μｓｅｃ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。

※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。

２－７　Ｄ／Ａコンバータ　sin、cos 値を出力してみる

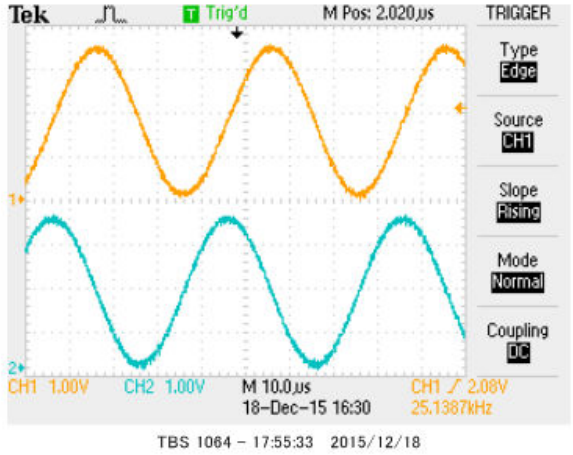
【　動作　】



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０～３６０°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。

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　