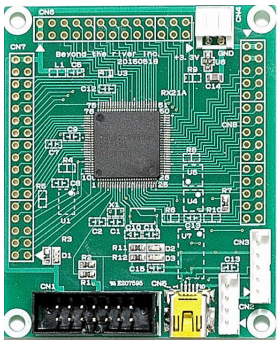
ＢＣＲＸ２１Ａ　マイコン開発セット　マニュアル

初版　２０１５．８．５

【　製品概要　】

本マニュアルはＢＣＲＸ２１Ａ　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+における開発方法について多く記述してあります。
※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：ＢＣＲＸ２１Ａ　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：＃ｉｎｃｌｕｄｅ　ｉｏｄｅｆｉｎｅ．ｈを忘れると何も出来ない
　　ｂ－２：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－３：Ｅ１から電源供給
　　ｂ－４：ＷＤＴ動作、リセット電圧選択はＯＦＳ０、１で設定する
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

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、ｃｏｓ演算した正弦波を出力する
　　ポート入力設定注意！

１−１．開発環境

ａ：開発セット同梱物

ＢＣＲＸ２１Ａ　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）

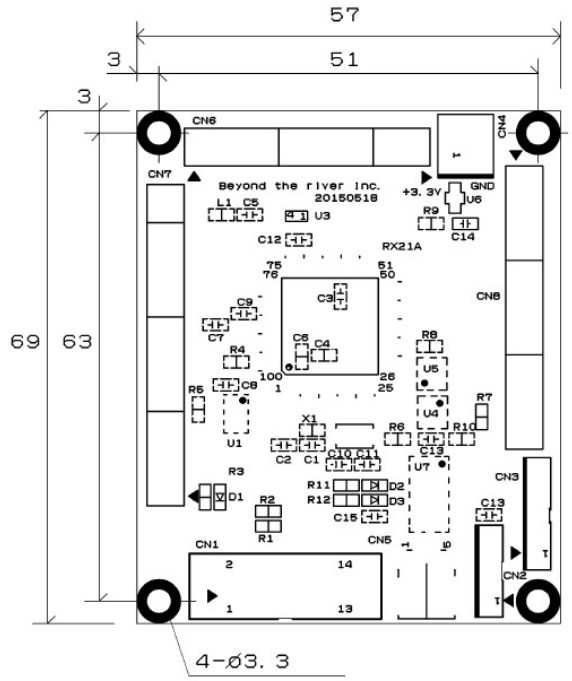
電源ケーブル、ＵＳＢ（ミニ）ケーブル

※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。

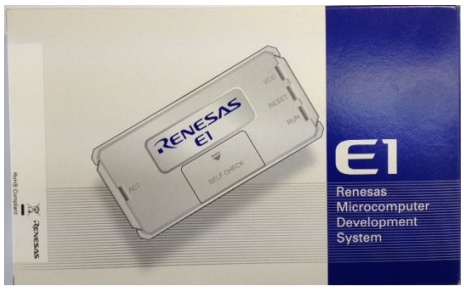
ｂ：ＢＣＲＸ２１Ａ　ＣＰＵボードの特徴

●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　５０ＭＨｚ動作可能。５段パイプラインＣＩＳＣハーバードアーキテクチャ
●メモリ容量　内蔵フラッシュＲＯＭ　５１２Ｋバイト、内蔵ＲＡＭ６４Ｋバイト　内蔵データフラッシュ８Ｋバイト
●Ａ／Ｄコンバータ　：２４ビット分解能×７　　変換速度９２μｓｅｃ、１２．２ＫＨｚ（ＤＳＡＤＣＬＫ＝２５ＭＨｚ）、入力範囲±０．５９０４Ｖ（これ以上、以下の電圧を加えないで下さい）
●Ａ／Ｄコンバータ　：１０ビット分解能×７　　変換速度　２μｓｅｃ（ＡＤＣＬＫ＝２５ＭＨｚ）
　　入力範囲０－３．３Ｖ
●Ｄ／Ａコンバータ：１０ビット分解能×２
●外部バス拡張機能：なし（外部にデータバス、アドレスバス等出力できません）
●Ｉ／Ｏポート：入出力６６、入力１、
●タイマ：マルチファンクションタイマパルスユニット２（１６ビットタイマ×６）、８ビットタイマ（８ビット×２）×２、コンペアマッチタイマ（１６ビットタイマ×２）×２、ウオッチドッグタイマ×１、独立ウオッチドグタイマ×１、ＲＴＣｃ　（リアルタイムクロック）
●シリアルコミュニュケーションインターフェイス×５ｃｈ、ＩＩＣバス×２ｃｈ、ＩｒＤＡバスインターフェイス、シリアルペリフェラルインターフェイス
●オンチップデバッキングシステム：（ＪＴＡＧおよびＦＩＮＥインターフェイス）
●動作周囲温度：－４０～＋１０５℃
●ＵＳＢポート：　１ｃｈ（ＳＩＯ１使用、フォトカプラ絶縁）　ＦＴＤＩ社　ＦＴ２３２ＲＬ使用
●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　３０ｍＡ（５０ＭＨｚ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　１２．５ＭＨｚ（×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。
●ΔΣＡ／Ｄ用外部基準電源：１．２００Ｖ（８ｐｐｍ／℃）実装済み。
●基板サイズ　６９×５７×１３（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）



ｃ：Ｅ１エミュレータ



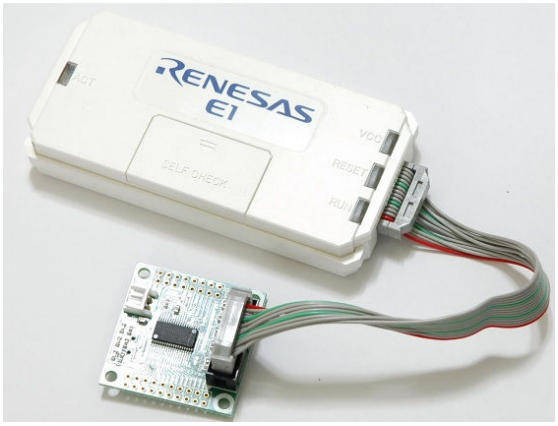
概要

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ＣＳ＋とＣコンパイラのダウンロードを行います。

ｄ：無償版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ＣＳ＋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　コンパイラ」の検索で表示されます。　　これを使用します。

現在（２０１５．８．７）ＲＸ用の開発環境は３種類あります。本開発セットで使用するのは１番上のＣＳ＋環境です。

「ダウンロード」をクリックします。ＣＳ＋用をダウンロードします。

CS+ (旧CubeSuite+)	【無償評価版】統合開発環境 CS+ for CC V3.01.00 (一括ダウンロード版)	Apr.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+からのアップ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、RL78ファミリ
CS+ (旧CubeSuite+)	【無償評価版】統合開発環境 CS+ for CC V3.01.00 (分割ダウンロード版)	Apr.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+からのアップ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、RL78ファミリ

いずれかのＣＳ＋　ｆｏｒ　ＣＣをダウンロードし、指示に従い展開して下さい。統合開発環境とＣコンパイラが同時にダウンロードされます。なお、ＣＳ＋は以前、ＣｕｂｅＳｕｉｔｅ＋という名称でしたが、２０１４年にＣS+となりました。大きな変更点は

１．ＣＳ＋　ｆｏｒ　ＣＡ，ＣＸ、ＲＸ、ＣＳ＋　ｆｏｒ　ＣＣ　とＣＰＵ別に２つに分割されました。

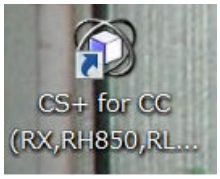
２．設定等も変更されています。但し、上位互換性はあり、ＣｕｂｅＳｕｉｔｅ＋で作成されたソフトはＣＳ＋　ｆｏｒ　ＣＣでコンパイル、実行可能です。

ｅ：開発セット添付ＣＤコピー、デバイスドライバのインストゥール

省略

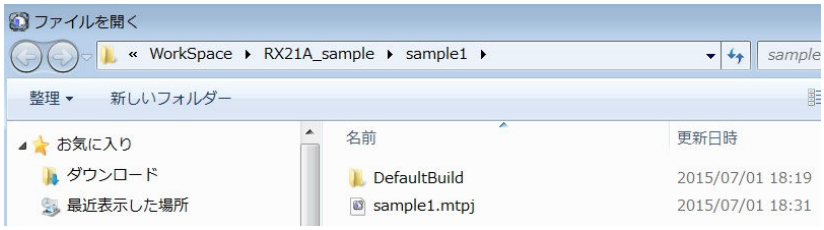
１－２　　動作、デバック

ａ：ＣＳ＋　ｆｏｒ　ＣＣ　起動、コンパイル、書き込み、動作

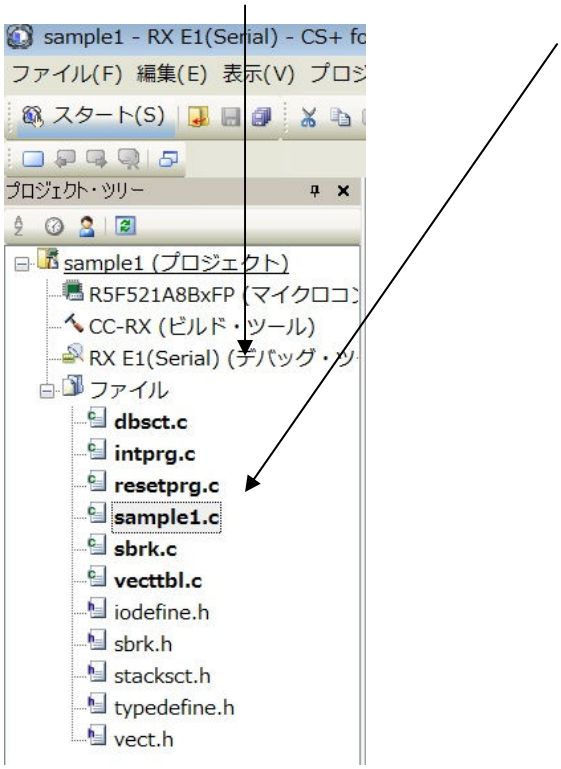


ＣＤに添付しているサンプルプログラムを使って、コンパイル、書き込み、動作の方法を示します。

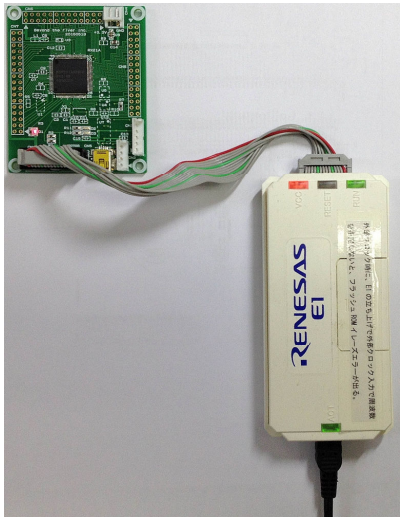
ＣＳ＋　ｆｏｒ　ＣＣを起動します。ここでは例としてＲＸ２１Ａ＿ｓａｍｐｌｅ￥ｓａｍｐｌｅ１を動作させます。基板上のＬＥＤ　Ｄ１が点滅するプログラムです。
ファイル　→　ファイルを開く　→　ｓａｍｐｌｅ１．ｍｔｐｊをダブルクリックします。



プロジェクトツリーが表示されます。ｓａｍｐｌｅ１．ｃをダブルクリック。
Ｅ１は設定済みです。

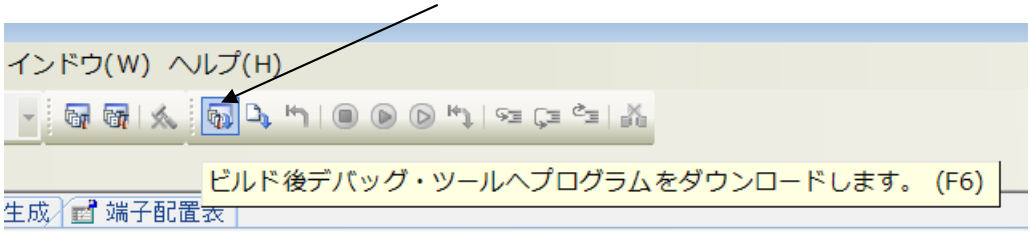


ｓａｍｐｌｅ１．ｃが中央に表示されます。とりあえず、実行してみます。Ｅ１のケーブルを基板のＣＮ１に挿入します。電源はＥ１から供給しますので、不要です。（写真ご参考）



（Ｅ１に貼ってある文字「外部クロック時にＥ１の立ち上げで外部クロック入力、周波数を指定しないとフラッシュＲＯＭイレーズエラーが出る」場合があります）

「ビルド後、デバック・ツールへプログラムを転送」をクリック。

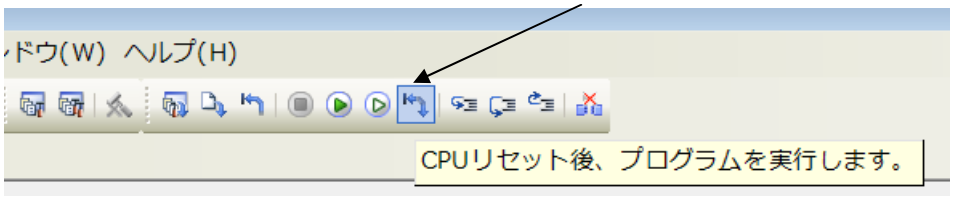


上手く転送できると、今まで表示されていなかったプログラムの絶対番地が表示されます。Ｅ１から電源がＣＰＵ基板に供給されます。　　（Ｅ１　ＶＣＣ　オレンジＬＥＤ点灯）

```
38                  /*****************************************
39                  * Function Name: main
40                  * Description  : The main loop
41                  * Arguments    : none
42                  * Return Value : none
43                  *****************************************
44      fff805e8    void main(void)
45                  {
46                      /* System initialization */
47      fff805ea        system_init();
48
49                  //port test
50                  //P3.5は入力専用なので波形が出ません。
51
52                          //PORTJ.PDR.BIT.B1 = 1;
53      fff805f0            PORT0.PDR.BYTE = 0xff;
54      fff805f7            PORT1.PDR.BYTE = 0xff;
55      fff805fe            PORT2.PDR.BYTE = 0xff;
56      fff80605            PORT3.PDR.BYTE = 0x3f;
57      fff8060d            PORT4.PDR.BYTE = 0xff;
58      fff80614            PORT5.PDR.BYTE = 0xff;
59      fff8061b            PORTA.PDR.BYTE = 0xff;
60      fff80622            PORTB.PDR.BYTE = 0xff;
61      fff80629            PORTC.PDR.BYTE = 0xff;
62      fff80630            PORTE.PDR.BYTE = 0xff;
63      fff80637            PORTJ.PDR.BYTE = 0x0a;
64      fff8063f            PORTH.PDR.BYTE = 0xff;
```

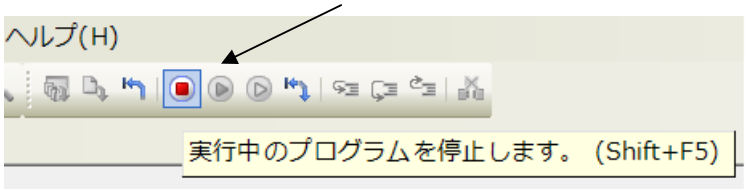
ここまでいかなかった場合、Ｅ１のインストゥールをご検証願います。

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。



Ｅ１のＲＵＮ（緑ＬＥＤ）が点灯し、基板のＤ１が点滅したら動作しています。ＣＳ＋の右下部にも表示されます。

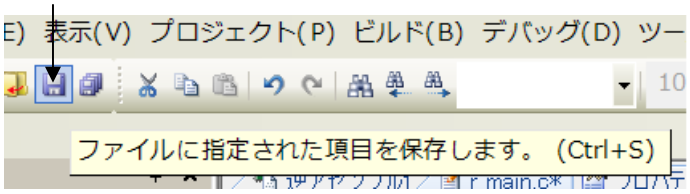
ここまで確認できましたら、一度止めます。



ｍａｉｎ関数のｌｗａｉｔの数値２箇所を１桁０を増やしてみます。

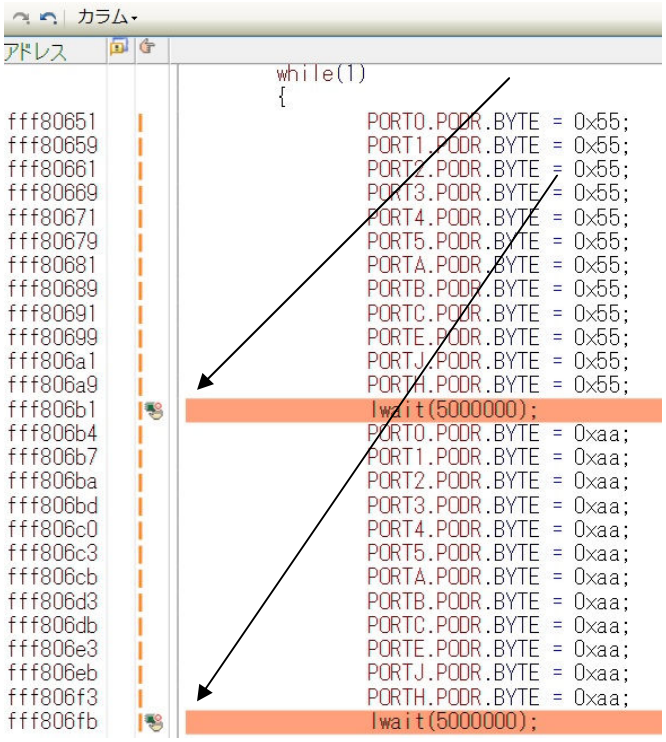
```
    while(1)
    {
            PORT0.PODR.BYTE = 0x55;
            PORT1.PODR.BYTE = 0x55;
            PORT2.PODR.BYTE = 0x55;
            PORT3.PODR.BYTE = 0x55;
            PORT4.PODR.BYTE = 0x55;
            PORT5.PODR.BYTE = 0x55;
            PORTA.PODR.BYTE = 0x55;
            PORTB.PODR.BYTE = 0x55;
            PORTC.PODR.BYTE = 0x55;
            PORTE.PODR.BYTE = 0x55;
            PORTJ.PODR.BYTE = 0x55;
            PORTH.PODR.BYTE = 0x55;
            lwait(5000000);
            PORT0.PODR.BYTE = 0xaa;
            PORT1.PODR.BYTE = 0xaa;
            PORT2.PODR.BYTE = 0xaa;
            PORT3.PODR.BYTE = 0xaa;
            PORT4.PODR.BYTE = 0xaa;
            PORT5.PODR.BYTE = 0xaa;
            PORTA.PODR.BYTE = 0xaa;
            PORTB.PODR.BYTE = 0xaa;
            PORTC.PODR.BYTE = 0xaa;
            PORTE.PODR.BYTE = 0xaa;
            PORTJ.PODR.BYTE = 0xaa;
            PORTH.PODR.BYTE = 0xaa;
            lwait(5000000);
```

セーブして

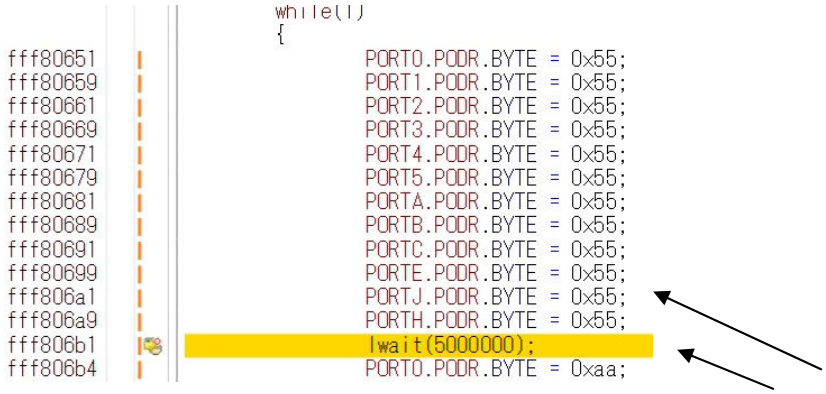


さきほどの、
「ビルド後、デバック・ツールへプログラムを転送」をクリック。
「ＣＰＵリセット後、プログラムを実行」をクリック。
ＬＥＤの点滅が先ほどより、遅くなったのが目視できましたでしょうか？

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。

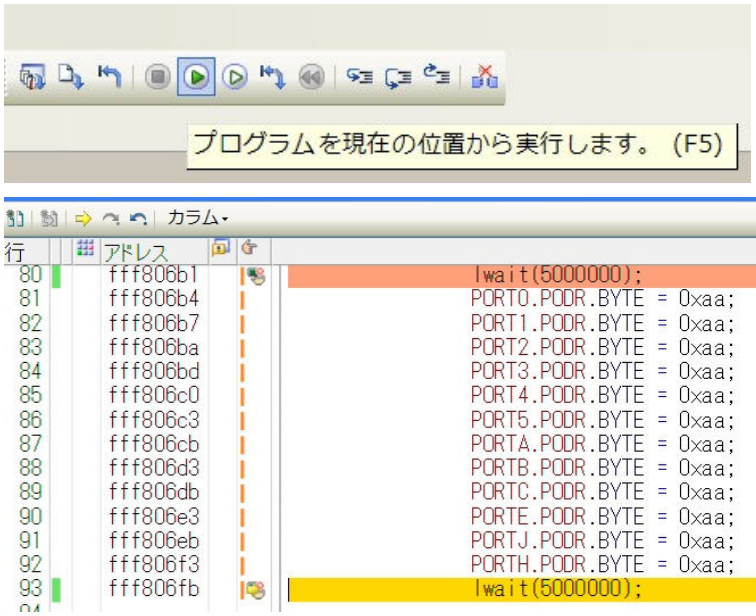


「ＣＰＵリセット後、プログラムを実行」します。



プログラムの実行はブレークポイントで停止し、ＬＥＤ　Ｄ１　は　ＰＯＲＴＪ．ＰＯＤＲ．ＢＹＴＥ＝０ｘ５５；命令によりＰＪ１＝０となるので、消灯します。

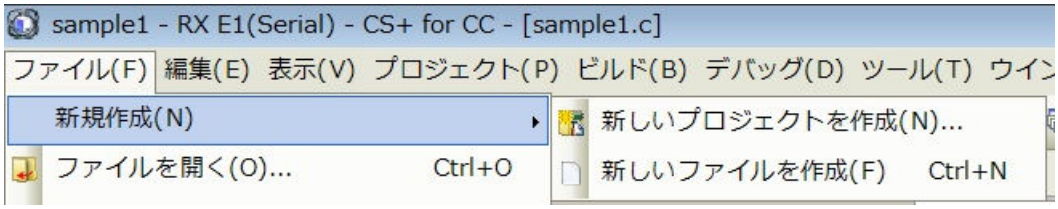
更に「プログラムを現在の位置から実行」をクリックすると、もう一つのブレークポイントで停止し、ＰＯＲＴＪ．ＰＯＤＲ．ＢＹＴＥ＝０ｘａａ；命令実行により、ＬＥＤは点灯します。



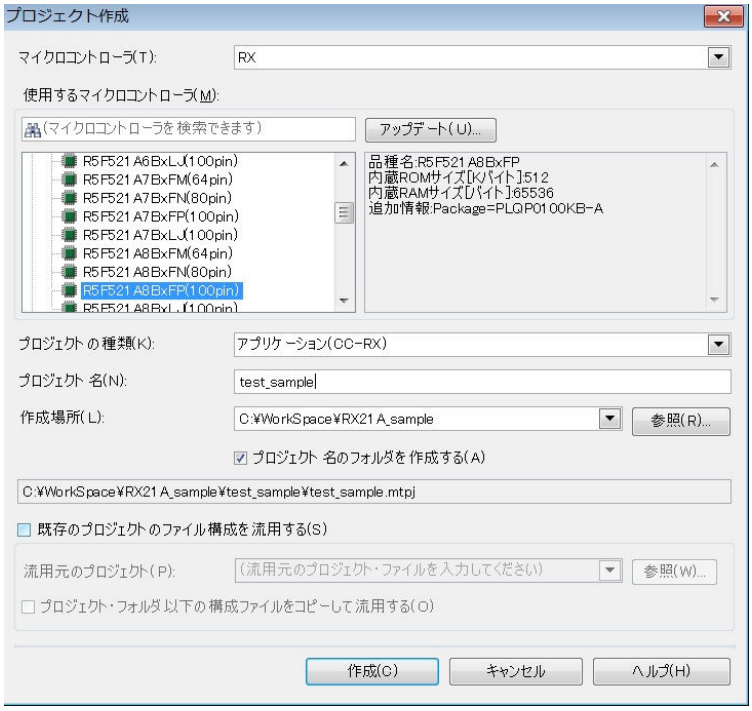
以上が、プログラムのコンパイル、Ｅ１へのダウンロード、実行、修正、ブレークポイント設定、動作の概要です。

ｂ：新しいプログラムを作る

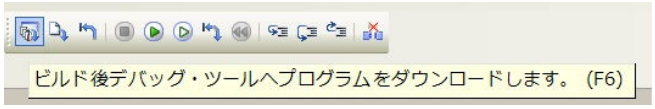
ＣＳ＋での新規プログラム開発方法と注意点を書きます。



ファイル→新しいプロジェクトを作成。ＣＰＵをＲ５Ｆ５２１Ａ８ＢｘＦＰを選択。



プロジェクト名をｔｅｓｔ＿ｓａｍｐｌｅとしました。「作成」をクリック。

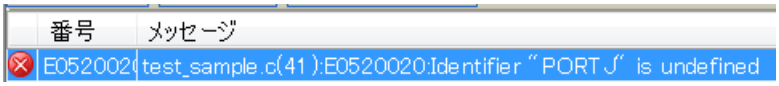


ビルド後デバック・ツールへプログラムをダウンロードクリック→エラーなしでダウンロード出来ますが、当然、何も実行されません。

ｓａｍｐｌｅ１．ｃを参考にもっとも簡単なプログラムを記入してみます。基板上のＬＥＤを点滅させるプログラムです。

```
void lwait(long wdata)
{
        while(wdata != 0)
        {
                wdata--;
        }

}

void main(void)
{
    /* System initialization */
    system_init();

        PORTJ.PDR.BIT.B1 = 1;

        while(1)
        {
                PORTJ.PODR.BIT.B1 = 1;
                lwait(500000);
                PORTJ.PODR.BIT.B1 = 0;
                lwait(500000);
        }

}
```

ビルドすると　ＰＯＲＴＪが見つからないエラーが出ます。



ｂ－１：＃ｉｎｃｌｕｄｅ　ｉｏｄｅｆｉｎｅ．ｈを忘れると何も出来ない

頭のほうに#include　"iodefine.h"　の１行を追記して下さい。またその下にクロック設定の sys.c をインクルード、sys.c ファイルをｓａｍｐｌｅ１等からこのｔｅｓｔ＿ｓａｍｐｌｅにコピーして下さい。

```
#include "iodefine.h"
#include "sys.c"
```

これでエラーも出ず、ダウンロードできるはずです。ところが、「リセット後、実行」させてもＬＥＤは光りません。

ｂ－２：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

省略

ｂ－３：Ｅ１から電源供給

省略

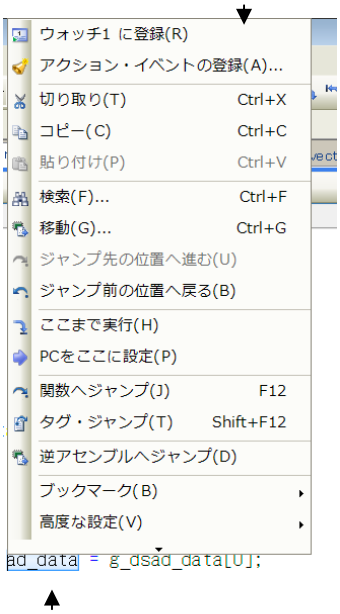
ｂ－４：ＷＤＴ動作、リセット電圧選択はＯＦＳ０、１で設定する

省略

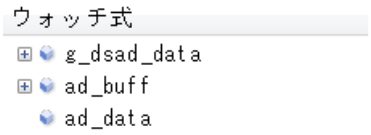
ｃ：その他

ｃ－１：動作中に変数の変化を見るには？

例としてｓａｍｐｌｅ３で使用しているａｄ＿ｄａｔａをウオッチ１に登録します。左クリックを押しながらマウスで文字を囲み、右クリック　ウオッチ１に登録を選択。

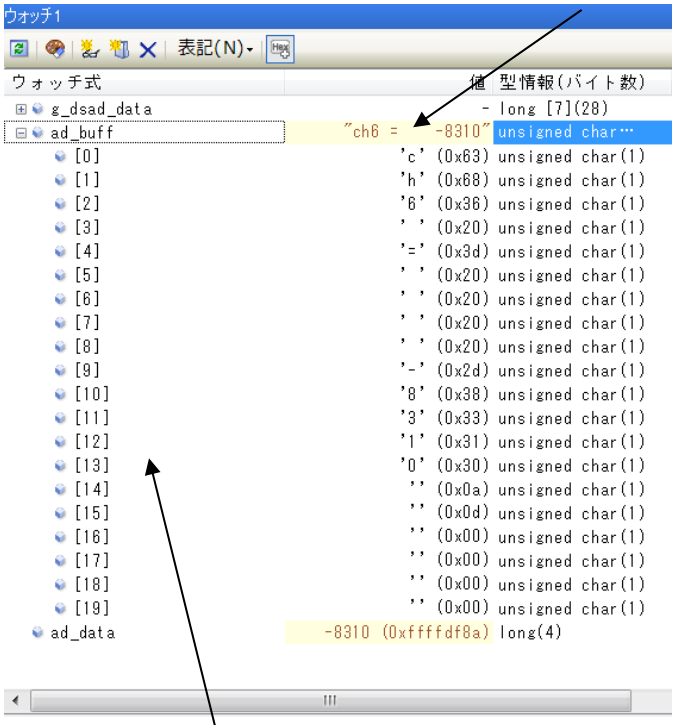


左クリックを押しながら、マウスで選択。右のウオッチ式に変数名が表示されるとＯＫです。



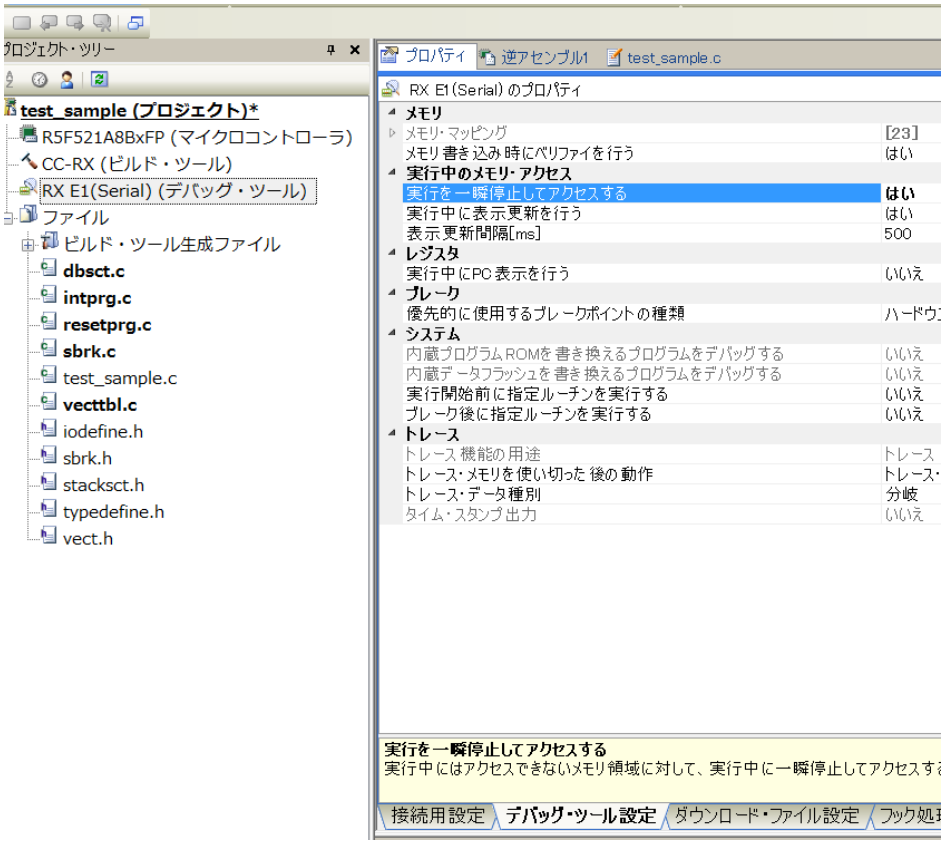
なお、ａｄ＿ｂｕｆｆのように頭に＋が付く変数は配列で、プラスをクリックすることにより個々の値も見ることが出来ます。

配列全体の値


ウォッチ1
表記(N)
ウォッチ式
値
型情報(バイト数)
g_dsad_data
- long [7](28)
ad_buff
"ch6 = -8310"
unsigned char…
[0] 'c' (0x63) unsigned char(1)
[1] 'h' (0x68) unsigned char(1)
[2] '6' (0x36) unsigned char(1)
[3] ' ' (0x20) unsigned char(1)
[4] '=' (0x3d) unsigned char(1)
[5] ' ' (0x20) unsigned char(1)
[6] ' ' (0x20) unsigned char(1)
[7] ' ' (0x20) unsigned char(1)
[8] ' ' (0x20) unsigned char(1)
[9] '-' (0x2d) unsigned char(1)
[10] '8' (0x38) unsigned char(1)
[11] '3' (0x33) unsigned char(1)
[12] '1' (0x31) unsigned char(1)
[13] '0' (0x30) unsigned char(1)
[14] '' (0x0a) unsigned char(1)
[15] '' (0x0d) unsigned char(1)
[16] '' (0x00) unsigned char(1)
[17] '' (0x00) unsigned char(1)
[18] '' (0x00) unsigned char(1)
[19] '' (0x00) unsigned char(1)
ad_data
-8310 (0xffffdf8a) long(4)

個々の値

更に、デバック・ツール設定→実行中のメモリアクセス→実行を一瞬停止してアクセスする→はいにします。



これにより、動作中に変数が変化する様子を見ることが出来ます。

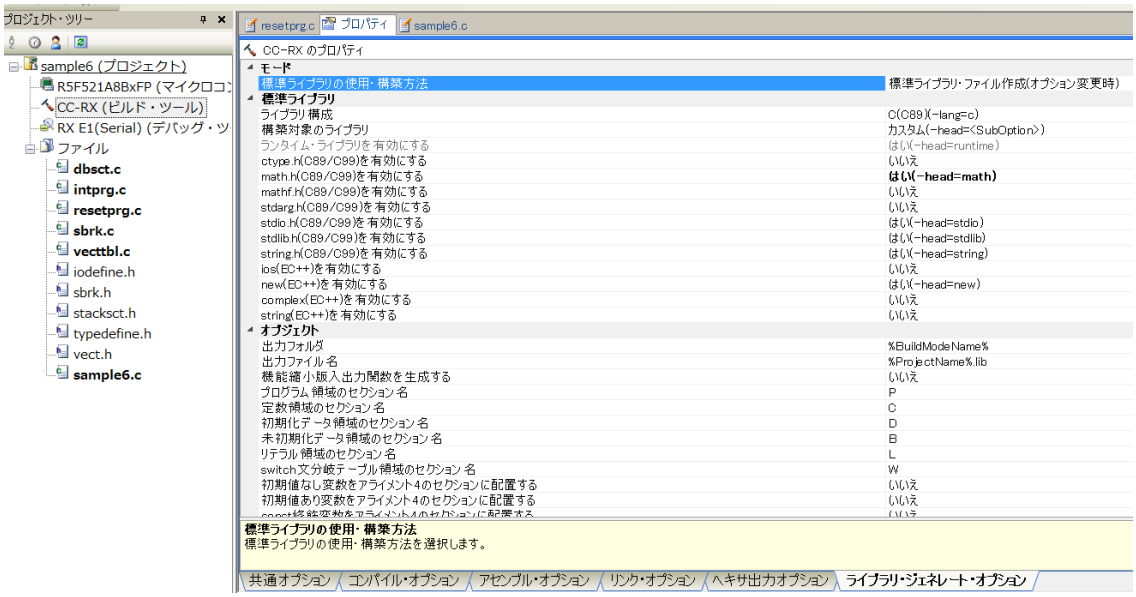
ｃ－２：サンプルを走らせるときにｖｅｃｔ．ｈの重複するアドレスを削除

省略

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。
```



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

省略

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

２．サンプルプログラム

ｓａｍｐｌｅ１　ポートのＯＮ/ＯＦＦ

【　概要　】

ポートのＯＮ/ＯＦＦを繰り返します。基板上のＬＤ１が点滅します。

【　プログラム　】

```
①void lwait(long wdata)
{
        while(wdata != 0)
        {
                wdata--;
        }

}

/***********************************************************************
* Function Name: main
* Description   : The main loop
* Arguments      : none
* Return Value : none
***********************************************************************/
②void main(void)
{
    /* System initialization */
③       system_init();                                          //外部12.5MH z 内部50MH z 動作
設定

//port test
//P3.5は入力専用なので波形が出ません。

        //PORTJ.PDR.BIT.B1 = 1;
④       PORT0.PDR.BYTE = 0xff;
        PORT1.PDR.BYTE = 0xff;
        PORT2.PDR.BYTE = 0xff;
        PORT3.PDR.BYTE = 0x3f;
        PORT4.PDR.BYTE = 0xff;
        PORT5.PDR.BYTE = 0xff;
        PORTA.PDR.BYTE = 0xff;
        PORTB.PDR.BYTE = 0xff;
        PORTC.PDR.BYTE = 0xff;
        PORTE.PDR.BYTE = 0xff;
        PORTJ.PDR.BYTE = 0x0a;
        PORTH.PDR.BYTE = 0xff;
```

```
        while(1)
        {
⑤               PORT0.PODR.BYTE = 0x55;
                PORT1.PODR.BYTE = 0x55;
                PORT2.PODR.BYTE = 0x55;
                PORT3.PODR.BYTE = 0x55;
                PORT4.PODR.BYTE = 0x55;
                PORT5.PODR.BYTE = 0x55;
                PORTA.PODR.BYTE = 0x55;
                PORTB.PODR.BYTE = 0x55;
                PORTC.PODR.BYTE = 0x55;
                PORTE.PODR.BYTE = 0x55;
                PORTJ.PODR.BYTE = 0x55;
                PORTH.PODR.BYTE = 0x55;
⑥               lwait(500000);
⑦               PORT0.PODR.BYTE = 0xaa;
                PORT1.PODR.BYTE = 0xaa;
                PORT2.PODR.BYTE = 0xaa;
                PORT3.PODR.BYTE = 0xaa;
                PORT4.PODR.BYTE = 0xaa;
                PORT5.PODR.BYTE = 0xaa;
                PORTA.PODR.BYTE = 0xaa;
                PORTB.PODR.BYTE = 0xaa;
                PORTC.PODR.BYTE = 0xaa;
                PORTE.PODR.BYTE = 0xaa;
                PORTJ.PODR.BYTE = 0xaa;
                PORTH.PODR.BYTE = 0xaa;
                lwait(500000);

        }

}
```

【　解説　】

```
①void lwait(long wdata)
{
        while(wdata != 0)
        {
                wdata--;
        }

}
```

時間を作り出している関数です。ｗｄａｔａが０になるまで関数から戻りません。数値が大きいほど時間も長くなります。

```
/***********************************************************************
* Function Name: main
* Description   : The main loop
* Arguments     : none
* Return Value : none
***********************************************************************/
②void main(void)
{
ここからメインです。

    /* System initialization */
③     system_init();                                    //外部12.5MH z 内部50MH z 動作設定
```
ＣＰＵを５０ＭＨｚで動作させています。ｓｙｓ．ｃの中に関数はあります。

```
④          PORT0.PDR.BYTE = 0xff;
           PORT1.PDR.BYTE = 0xff;
```
ポートを出力ポートに設定しています。

```
⑤                    PORT0.PODR.BYTE = 0x55;
```
ポートに０ｘ５５＝０１０１０１０１Ｂを出力しています。

```
⑥                    lwait(500000);
```
ＬＥＤの点滅を人の目で確認出来る速度にするための待ち時間です。

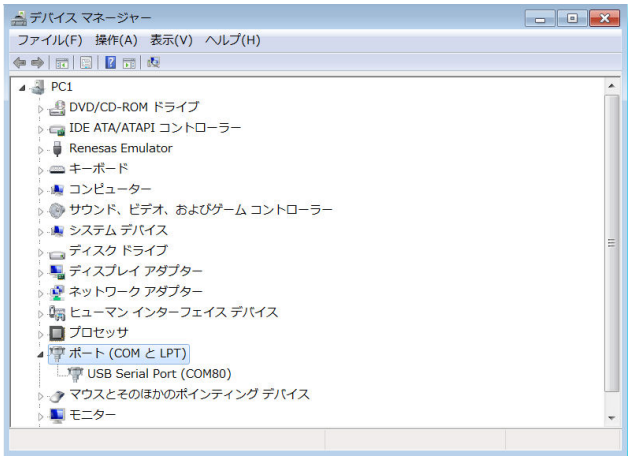
```
⑦                    PORT0.PODR.BYTE = 0xaa;
```
ポートに０ｘａａ＝１０１０１０１０Ｂを出力しています。０ｘ５５に比べすべてのビットが反転し、ＬＥＤが点滅して見えます。仮に隣のポートと接触しているとレベルの変化がありませんので、例えばＬＥＤが点滅しません。それによりハードウエアの異常が検出できます。（隣は必ず異なる論理なので０，１でも１，０でも０になります）

２－２　ｓａｍｐｌｅ２　ＳＩＯ（ＵＳＢ）でパソコンとのやりとり

【　概要　】

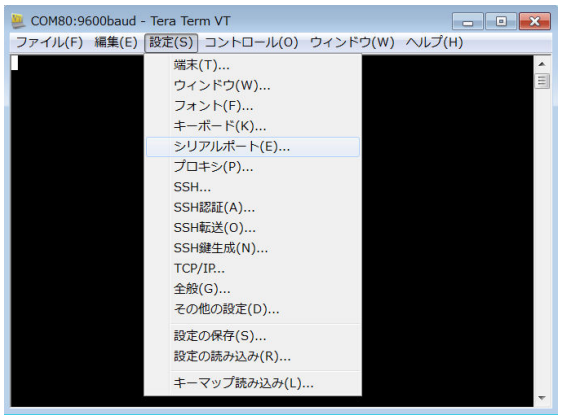
ＵＳＢ出力をパソコンと接続し、データのやり取りを行います。お手数ですが、テラタームやハイパーターミナルなどのターミナルプログラムを使用しますので、無い方は、ネットで検索し、インストゥール願います。例ではテラタームで行います。ボーレートは３８４００ｂｐｓに設定して下さい。

ＵＳＢミニケーブルでパソコンとつなげると、ＵＳＢ側に電源が入ります。コントロールパネル→全てのコントロールパネル項目→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。例ではＣＯＭ８０となっています。



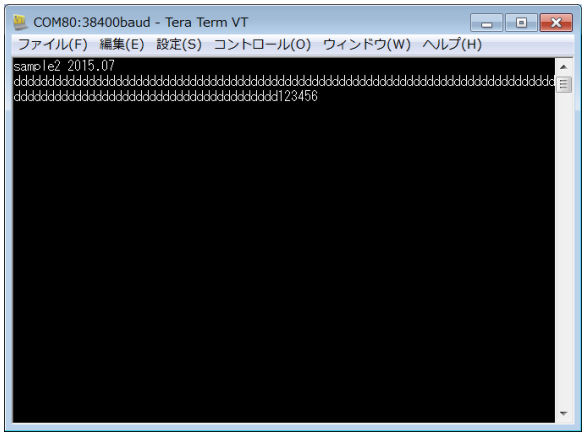
Ｔｅｒａ　Ｔｒｅｍをシリアルポート　ＣＯＭ８０　→ＯＫとします。

設定→シリアルポート→ボーレート３８４００として下さい。



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

グラム実行。



ｓａｍｐｌｅ２　２０１５．０７　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。

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。

【　プログラム　】

```
void main(void)
{
   //System initialization

            system_init();                                                    //外部12.5MH z 内部
50MH z 動作設定

  //SIO initial

①           init_sio();

②           char_out1('s');
            char_out1('a');
            char_out1('m');
            char_out1('p');
            char_out1('l');
            char_out1('e');
            char_out1('2');
            char_out1(' ');
            char_out1('2');
            char_out1('0');
            char_out1('1');
            char_out1('5');
            char_out1('.');
            char_out1('0');
```

```
        char_out1('7');
③       char_out1(CR);
        char_out1(LF);

        while(1)
        {
④               char_out1(char_in1());

        }


}
```

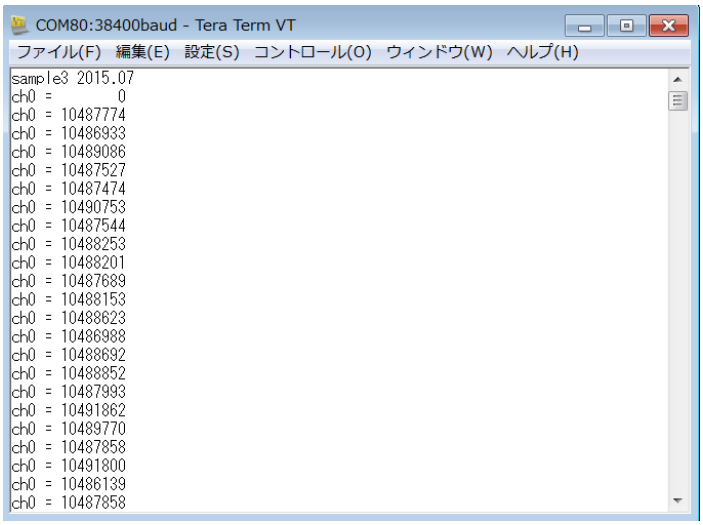
【　解説　】

省略

２－３　ｓａｍｐｌｅ３　Ａ／Ｄ変換をＵＳＢ出力

【　動作概要　】

ＡＮＤＳ０Ｎ（ＣＮ７　２３）　ＡＮＤＳ０Ｐ（ＣＮ７　２２）を入力とし、Ａ／Ｄ変換した値をＵＳＢからパソコンに送ります。±０．５９０４Ｖがフルスケールです。これ以上の電圧を入力しないで下さい。



【　プログラム　】

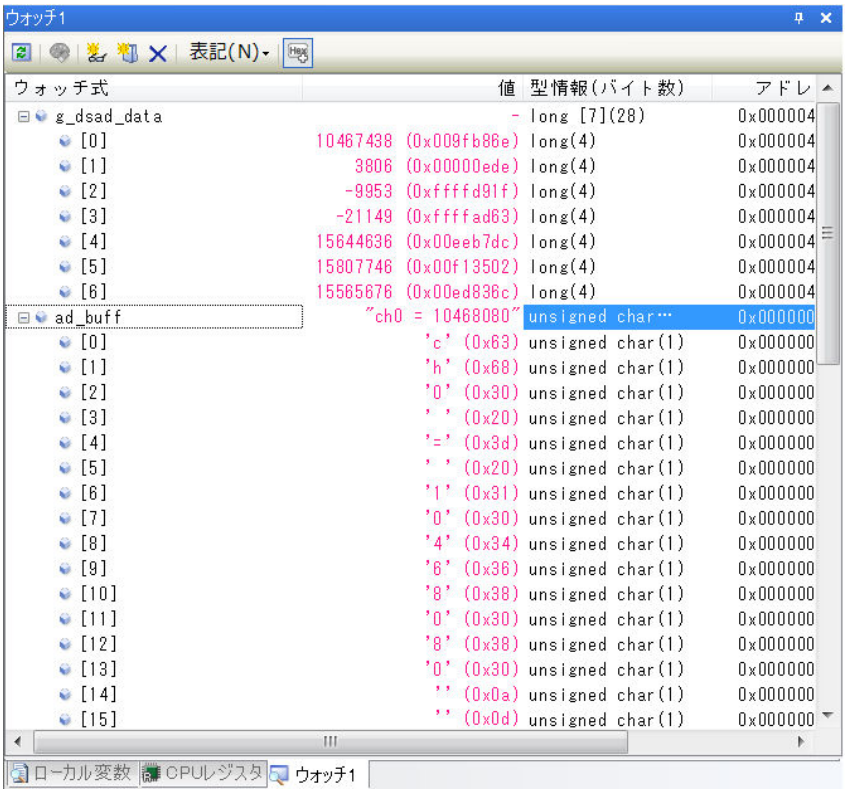
```
void main(void)
{
unsigned char cf,loop;

   //System initialization

①      clrpsw_i();

      //System initialization
      system_init();                          //外部12.5MH z 内部50MH z 動作設定

      //DSAD initialization
②      dsad_init();                          //ΔΣ24bit　A/Dイニシャライズ

      //MTU2a-channel0 (MTU0) initialization
③      mtu0_init();                                                    //

      //ELC initialization
④      elc_init();                           //イベントリンクコントローラ　ΔΣ24bitデータ変換→


  //SIO initial
```

```
        init_sio();

        char_out1('s');
        char_out1('a');
        char_out1('m');
        char_out1('p');
        char_out1('l');
        char_out1('e');
        char_out1('3');
        char_out1(' ');
        char_out1('2');
        char_out1('0');
        char_out1('1');
        char_out1('5');
        char_out1('.');
        char_out1('0');
        char_out1('7');
        char_out1(CR);
        char_out1(LF);

        //
⑤       setpsw_i();

    //Starts MTU0 and DSAD conversion
    MTU.TSTR.BIT.CST0 = 1;


        while(1)
        {
⑥       ad_data = g_dsad_data[0];

                sprintf(ad_buff,"ch0 = %8d¥n¥r",ad_data);

                cf = 1;loop = 0;
                while(cf != 0)
                {
                        cf = ad_buff[loop];
                        char_out1(cf);
                        loop++;

                }

                lwait(1000000);
```

```
            }
}

⑦#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;
}
```

【　解説　】

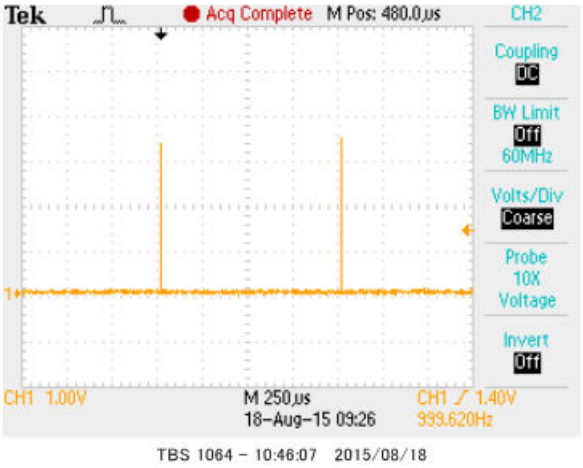
省略



２－４　ｓａｍｐｌｅ４　割り込み

【　動作概要　】

ｓａｍｐｌｅ４を動作させます。オシロスコープがあればＰＪ１　ＣＮ７　１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。



ΔΣ２４ｂｉｔＡ／ＤのデータをＵＳＢで出力する動作はｓａｍｐｌｅ３と同じです。
違いは
①ｓａｍｐｌｅ３のウエイト

```
/                       lwait(1000000);

                        wtime = 1000;
                        while(wtime != 0)
                                ;
```

を止め、割り込みで時間を作成してあります。１ｍｓｅｃ×１０００＝１Ｓ毎にデータを出力します。

②１ｍｓｅｃ毎にデータを加算し、平均値を出力しています。

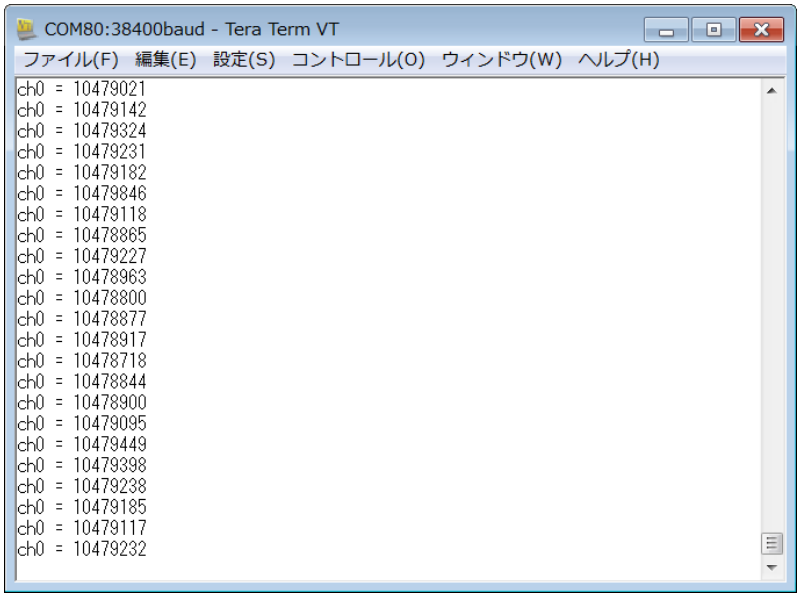
【　プログラム　】

省略

【　解説　】

省略

１００回の平均化で上位５桁まで、１ｄｉｇｉｔのちらつきで測定出来ました。


COM80:38400baud - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
ch0 = 10479021
ch0 = 10479142
ch0 = 10479324
ch0 = 10479231
ch0 = 10479182
ch0 = 10479846
ch0 = 10479118
ch0 = 10478865
ch0 = 10479227
ch0 = 10478963
ch0 = 10478800
ch0 = 10478877
ch0 = 10478917
ch0 = 10478718
ch0 = 10478844
ch0 = 10478900
ch0 = 10479095
ch0 = 10479449
ch0 = 10479398
ch0 = 10479238
ch0 = 10479185
ch0 = 10479117
ch0 = 10479232

２－５　ｓａｍｐｌｅ５　PWM出力
２－６　三角、対数、平方根関数を使う

【　概要　】

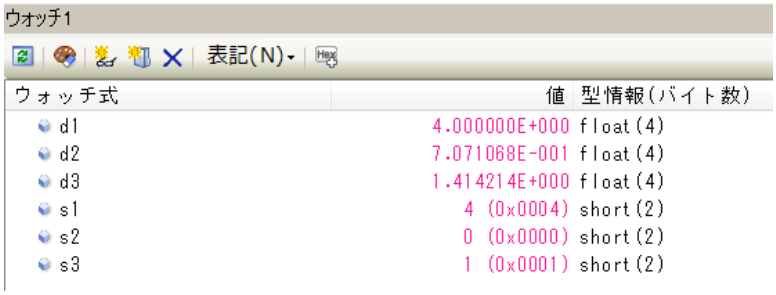
ｌｏｇ、ｓｉｎ、√　演算を行い、演算結果の確認とその速度を測定します。

【　プログラム　】

省略

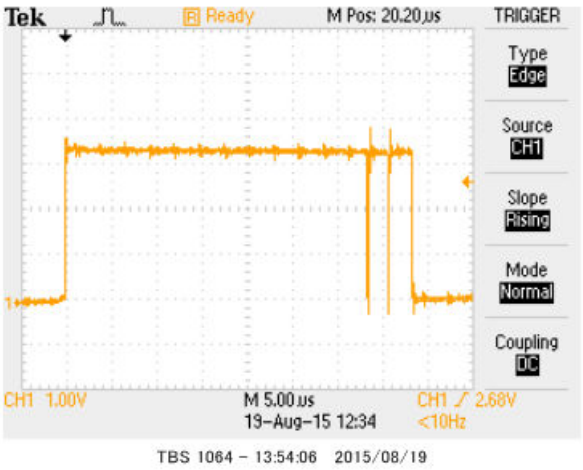
【　解説　】

省略

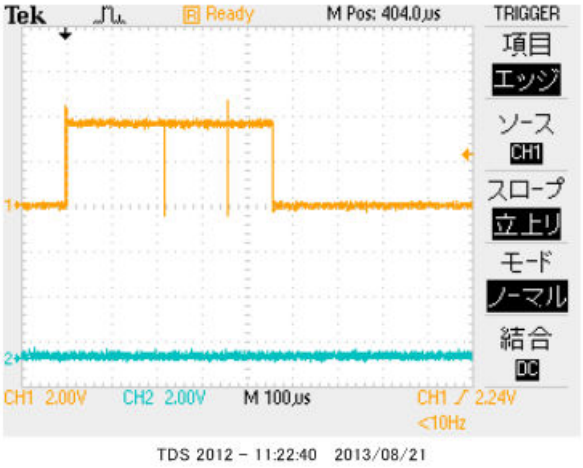


演算速度ですが、
ｌｏｇ１０（１００００）が約３３μｓｅｃ、ｓｉｎ（４５°）が２．５μｓｅｃ、√２が３μｓｅｃ程度かかるようでした。例えばＲＬ７８（３２ＭＨｚ）では約２２０μｓｅｃ、ｓｉｎ（４５°）が１３０μｓｅｃ、√２が１００μｓｅｃ程度ですので、それぞれ６．６倍、５２倍、３３倍も速いことになります。
マイコンに必要な能力が演算処理速度の場合、ＲＸを使用するのが圧倒的に有利であることが分かります。

ＲＸ２１Ａ（５０ＭＨｚ）



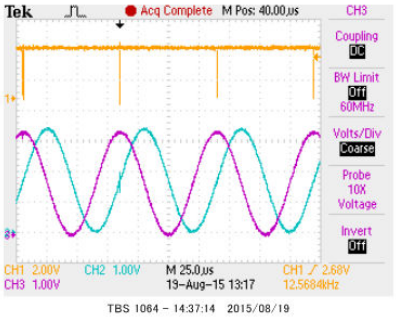
ＲＬ７８（３２ＭＨｚ）



２－７　Ｄ／Ａにｓｉｎ，ｃｏｓ演算した正弦波を出力する

【　概要　】

ＲＸ２１Ａがもつ、２ｃｈ　１０ビットＤ／Ａにｓｉｎ，ｃｏｓ演算結果（最大±１）を０－３．３Ｖに変換し出力します。正弦波オシレーターになります。



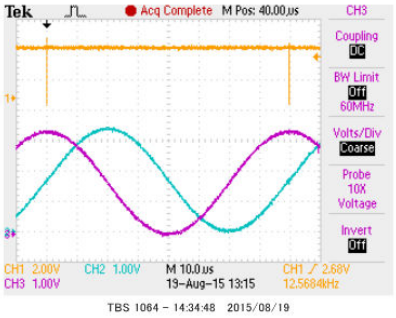
【　プログラム　】

省略

【　解説　】

省略

０から３５９度まで、先に演算したデータをＤ／Ａコンバータにセットしています。１２．５ＫＨｚ程度の正弦波が得られています。毎回、演算をしていると速い波形は出せません。



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　