ＲＬ７８　マイコン学習セット　マニュアル　入門編

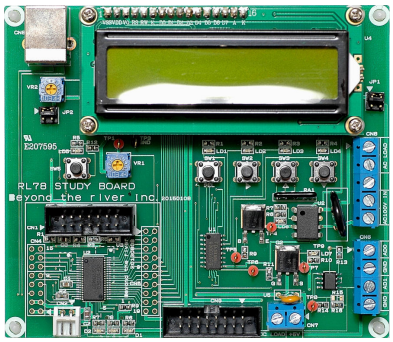
第１版２０１５．４．１６　　**第１版**

【　製品概要　】

本マニュアルはＲＬ７８／Ｉ１Ａ　Ｒ５Ｆ１０７ＤＥ（３８ピン）マイコンを使ったマイコン学習セットの開発環境構築、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。

入門編ではマイコンの基本的なハードウエアのアクセス方法、プログラムの書き方をサンプルプログラムを参考に学び、習熟度をチェックするために、演習プログラムの課題を自分で考えます。
また、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。

※本学習セット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．学習環境、事前準備

１－１．　学習環境

ａ：学習セット　同梱物

ｂ：ＢＣＲＬ７８１０７　ＣＰＵ部の特徴

ｃ：Ｅ１エミュレータ（デバッカ）

ｄ：無償のＣＳ＋、ＲＬ７８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

ｆ：ＲＬ７８とＨ８／３００Ｈ、Ｒ８Ｃの速度比較

ｆ－１：ポートアクセス速度の比較

ｆ－２：乗除演算速度の比較

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：Ａ／Ｄ、リセット、ウオッチドッグ設計上の注意点

ｂ－２：自動生成されたプログラム

ｂ－３：Ｅ１から電源供給

ｂ－４：コード生成後の初期値の変更

ｂ－５：変数を見る

ｂ－６：変数変化を実行中に確認する

２．サンプルプログラム

２－１．キー入力　ｓａｍｐｌｅ１

　　プログラム　　：キー入力で LED を点灯

　　演習プログラム　：キー入力で LED を点灯、押されていても一定時間で消す

２－２．ＵＳＢ通信 ｓａｍｐｌｅ２

　　プログラム　　：ＡＢＣＤをパソコン側に送信

　　演習プログラム　：キー入力でＡＢＣＤと送信

２－３．Ａ／Ｄ変換 ｓａｍｐｌｅ３

　　プログラム　　：Ａ／Ｄ変換データをパソコン側に送信

　　演習プログラム　：Ａ／Ｄ値を０－５Ｖに換算しパソコン側に送信。

２－４．ＰＷＭ　　ｓａｍｐｌｅ４

　　プログラム　　：ＬＥＤ輝度連続可変

　　演習プログラム　：ＬＥＤ輝度階段状可変

２－５．割り込み　ｓａｍｐｌｅ５

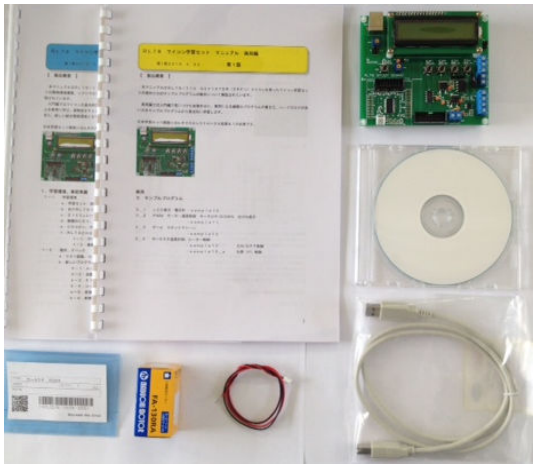
　　プログラム　　：割り込みでＬＤ１点灯

　　演習プログラム　：メインでＬＤ１点滅、割り込みでＬＤ２点灯、消灯

１−１．学習環境

ａ：学習セット同梱物

ＲＬ７８学習ボード	１
ＣＤ（サンプルプログラム、デバイスドライバ、ドキュメント）	１
マニュアル（本誌） 入門、実用	各１
電源ケーブル、ＵＳＢケーブル	各１
モーター	１
サーミスタ	１



※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。
但し、プログラムの検討、コンパイルは、無料のＣＳ＋（後述）で行うことが出来ます。

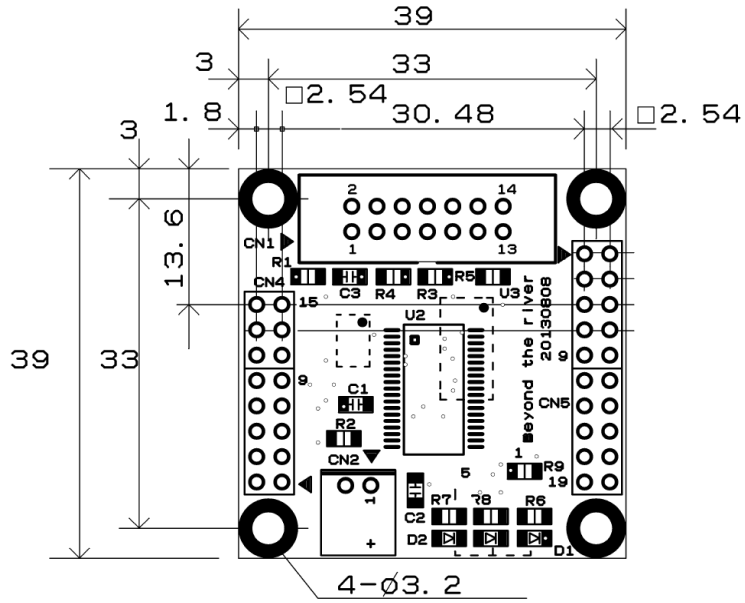
複数の人間の学習において
Ａ．Ｅ１＋本ボード＋ＣＳ＋インストゥール済みパソコンを用意
Ｂ．プログラムの検討、コンパイルは他のパソコンで行い、実行だけＡのパソコンに席を移る
といった使い方で、人数分用意しなくても効率よく学習することは可能だと思います。もちろん、各人に各台数あるのが、時間的な効率は一番良いです。

ｂ：ＢＣＲＬ７８１０７　ＣＰＵボード部の特徴
学習ボードのマイコン部分は弊社ＢＣＲＬ７８１０７ＣＰＵボードと同じです。
●高性能、低消費電力、低コストな新設計ＲＬ７８コアを使用。１．３９ＤＭＩＰＳ/ＭＨｚ、４６μＡ／ＭＨｚ。３２ＭＨｚ±１%の高精度内蔵オシレータ　※１
●ＲＬ７８/Ｉ１Ａ（Ｒ５Ｆ１０７ＤＥ）は産業、インフラ、情報アプリケーションに特化した強力な周辺機能（高性能ＰＷＭタイマ、ＬＩＮ－ｂｕｓ、ＤＡＬＩ通信機能）を搭載。３８ピン。
●内蔵高速オシレータ　　３２ＭＨｚ（２．７～５．５Ｖ）。最小命令実行時間３１．２５ｎｓｅｃ。
●内蔵低速オシレーター　１５ＫＨｚ（ＴＹＰ）　ＣＰＵクロックとしては使用不可。
●メモリ容量　フラッシュＲＯＭ６４Ｋバイト、ＲＡＭ４Ｋバイト、データフラッシュ４Ｋバイト。
電源を切ってもデータが保持されるＥＥＰＲＯＭ　２５ＬＣ２５６（容量３２、７６８ＢＹＴＥ）搭載　ライブラリ添付※２
●基板大きさ、超小型３９×３９×１５ｍｍ
●動作電圧電流　３．３Ｖ～５．５Ｖ、１６ｍＡ　ＴＹＰＥ（５Ｖ、ＵＳＢ使用、３２ＭＨｚ動作時）
最低２．７Ｖから動作可能（ＢＣＲＬ７８１０７Ｓタイプ　※２）
●豊富な周辺機能

Ｉ／Ｏポート　合計３４、Ａ／Ｄ変換器：１０ビット分解能　１１ｃｈ、プログラマブルゲインアンプ　６ｃｈ、ＵＡＲＴ　３ｃｈ（１ｃｈはＬＩＮ－ｂｕｓ、ＤＭＸ５１２、ＤＡＬＩ通信対応）
タイマ８ｃｈ（PWM出力３ｃｈ、１ｎｓｅｃ分解能可能、６４ＭＨｚＰＬＬ＋ディザリング）、乗除算・積和演算器内蔵、オンチップデバック機能内蔵
●ＵＳＢ搭載　ミニＢコネクタ、ドライバＩＣ　ＦＴＤＩ社　ＦＴ２３２ＲＬ搭載。※２
●デバッカＥ１によるデバック用コネクタ搭載。Ｃ言語による１行実行、ブレークポイント、変数参照等可能です。

※１　速度比較は本マニュアル　１－１　ｆ：ＲＬ７８とＨ８／３００Ｈ、Ｒ８Ｃの速度比較をご参照下さい。
※２　学習ボードはＣＰＵ＋デバック用コネクタ、ＵＳＢインターフェイス＋ＥＥＰＲＯＭ搭載のＢＣＲＬ７８１０７Ｍが使用されています。

ＣＰＵ部大きさ（部品面）



ＵＳＢミニＢコネクタ、ＦＴ２３２ＲＬ、２５ＬＣ２５６は裏面搭載。

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。

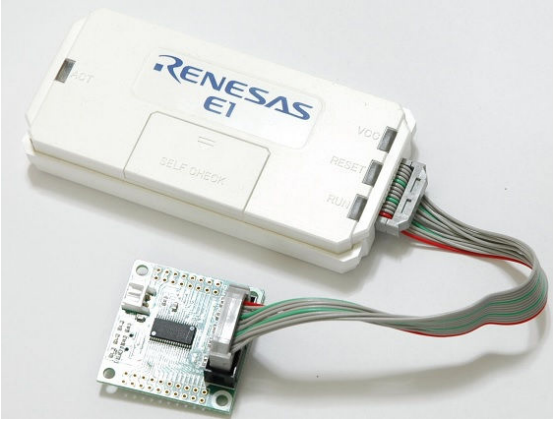
Ｃ言語ソースデバックが可能で、１行実行、ブレークポイント設定、変数、レジスタ、メモリ参照等々、従来であれば高価なＩＣＥしか出来なかった機能が、安価に実現されています。変数をウオッチ窓に登録し、実行中を含めて数値を見ながらデバック出来ます。

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。

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ＲＬ７８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→「ＲＬ７８　無償コンパイラ」の検索で表示されます。

いずれかのＣＳ＋　ｆｏｒ　ＣＡ，ＣＸ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．ＲＬ７８用はＣＳ＋　ｆｏｒ　ＣＡ，ＣＸ、ＲＸ用はＣＳ＋　ｆｏｒ　ＣＣ　と別環境に分割されました。

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　ＣＡ，ＣＸでコンパイル、実行可能です。

３．２０１５　３／２０に出たＣＳ＋　ｆｏｒ　ＣＣ　はＲＬ７８の開発が行えますが、１，２の上位互換性が無いので本学習セットには使用できません。ご注意願います。（２０１５．５．７）

以下省略

ｆ：ＲＬ７８とＨ８／３００Ｈ、Ｒ８Ｃの速度比較

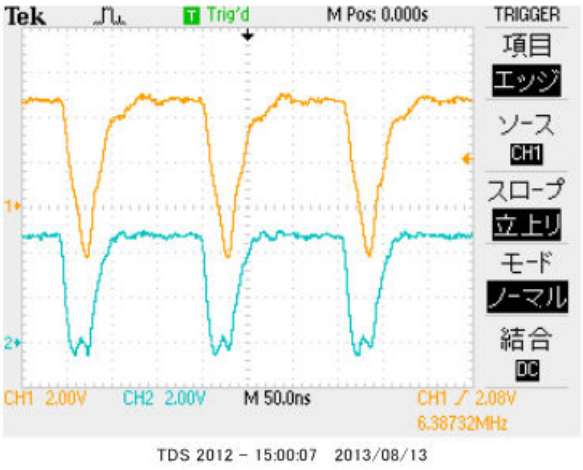
ＲＬ７８は、製造中止がアナウンスされているＨ８／３０４８の替わりに検討される方も多いと思われますが、実行速度はどうなのでしょうか？　開発環境を含めて以前より進化していなければ使う意味がないとお考えの方も多いかと思われます。

ｆ－１　ポートアクセス速度比較

単純なポートアクセスプログラムで比較してみます。
ＲＬ７８のポートを１，０繰り返すプログラムです。

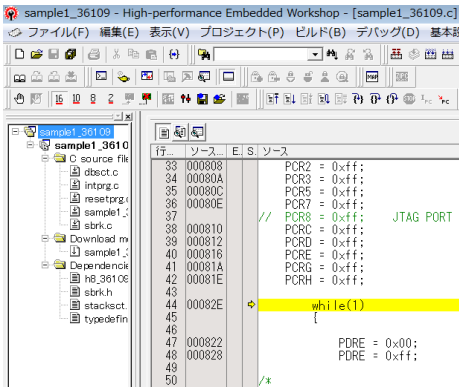
```
64                          void main(void)
65                          {
66      00187               |   R_MAIN_UserInit();
67                          |   /* Start user code. Do not edit comment generated here */
68                          |   while (1U)
69                          |   {
70      0018b               |       P2 = 0x00;
71      0018d               |       P2 = 0xff;
72                          |
73                          |
74      00190               |   }
75                          |   /* End user code. Do not edit comment generated here */
76      00192               | }
```

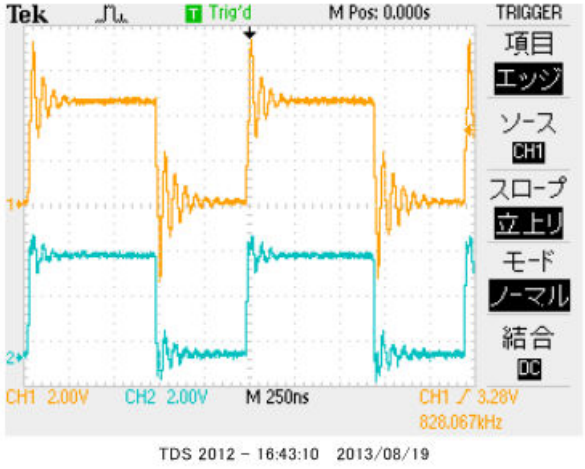
オシロスコープでＰ２０、Ｐ２１波形を観測すると６．３８７３２ＭＨｚという周波数でポートの１，０を繰り返すことが分かります。（クロック３２ＭＨｚ）



この命令の詳細は
ｗｈｉｌｅ（１Ｕ）
｛
Ｐ２　＝　０ｘ００；　　//ポートを０にする
Ｐ２　＝　０ｘｆｆ；　　//ポートを１にする
｝　　//上行にジャンプする
という３つの動作を行っています。波形が１から０に落ちて、上がる手前の時間が１命令の実行時間です。波形上約３０ｎｓｅｃ程度なので、カタログ値　３１．２５ｎｓｅｃと大きく相違は無いように思います。１クロックで１命令実行はＲＩＳＣ並みですね。１の時間が０に比べて長いのはポートを１にする、上行にジャンプするの２命令実行しているからです。

Ｈ８／３００Ｈコアを代表してＨ８／３６１０９を使用しました。基板名ＢＣＨ８３６１４０９。ＨＥＷで同じ意味のコードを書き込みテストします。Ｈ８／３００ＨコアはＨ８／３０４８やＨ８／３０５２と同じです。





ポートＥを繰り返し、０、１しています。波形を観測すると８２８．０６７ＫＨｚとなりました。

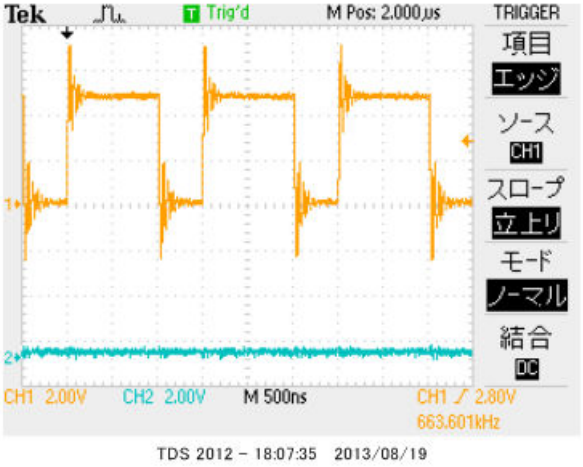
６．３８７３２ＭＨｚ÷８２８．０６７ＫＨｚ≒７．７倍高速という驚きの結果になりました。（クロック２０ＭＨｚ）クロックを同じにしても、４．８倍違います。

次にＲ８Ｃを評価します。Ｒ８Ｃ/Ｍ１２Ａ（クロック２０ＭＨｚ）を使用して比較してみます。

```
645 0EBE3
646
647 0EBE5
648 0EBE9
649 0EBE9
650 0EBED
651 0EBF1
```

```
asm("FCLR I");      //マスカブル割り込み禁止

    while(1)    //test
    {
        p1_0 = 0;
        p1_0 = 1;
    }
```



６６３．６０１ＫＨｚとなりました。

ｆ－２　乗除演算速度の比較

演算速度はどの程度違うでしょうか？　３２ｂｉｔの乗算、除算を行ってみました。
演算前にポートを立てて、演算後にポートを下ろすことにより、演算実行時間をオシロで観測しています。

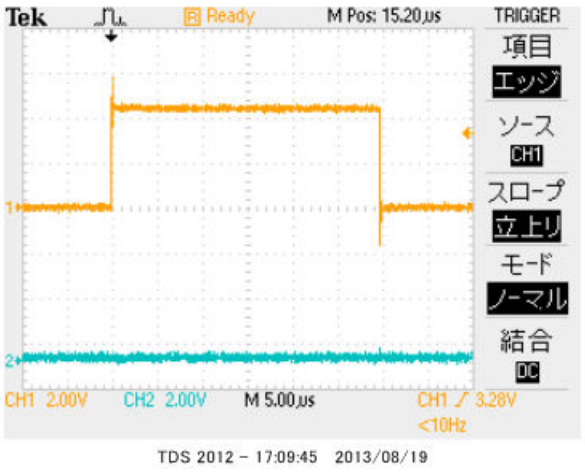
Ｈ８－３６１０９　約３０μｓｅｃでした。

```
000874          while(1)
                {
00082E              ldata1 = 1234;
000838              ldata2 = 5678;

000840              PDRE = 0xff;
00084E              ldata3 = ldata1 * ldata2;
00085E              ldata4 = ldata2 / ldata1;
00086E              PDRE = 0;
                /*
                    PDR1 = 0;
                    PDR2 = 0;
                    PDR3 = 0;
                    PDR5 = 0;
                    PDR7 = 0;
                //  PDR8 = 0;
                    PDRC = 0;
                    PDRD = 0;
```

sample1...

Address	+0	+4	+8	+C
FFE40C	006AE9BC	00000004	00000000	00000000

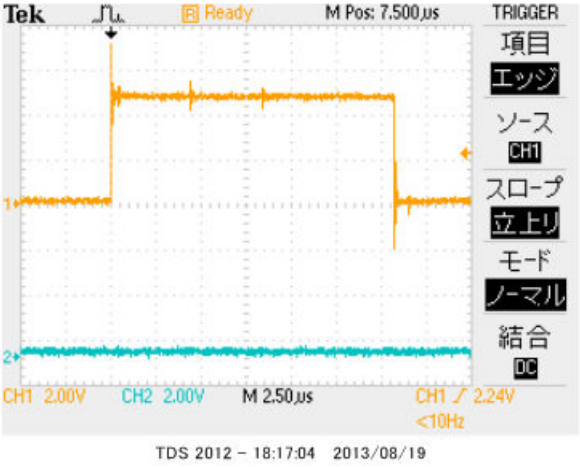


Ｒ８Ｃ／Ｍ１２Ａの場合　約１５．５μｓｅｃでした。

```
0EBE3       asm("FCLR I");      //マスカブル割り込み禁止

0EBE5       while(1)    //test
0EBE9       {
0EBE9           ldata1 = 1234;
0EBF3           ldata2 = 5678;

0EBFD           p1_0 = 1;
0EC01           ldata3 = ldata1 * ldata2;
0EC1F           ldata4 = ldata2 / ldata1;
0EC3D           p1_0 = 0;
0EC41       }
```



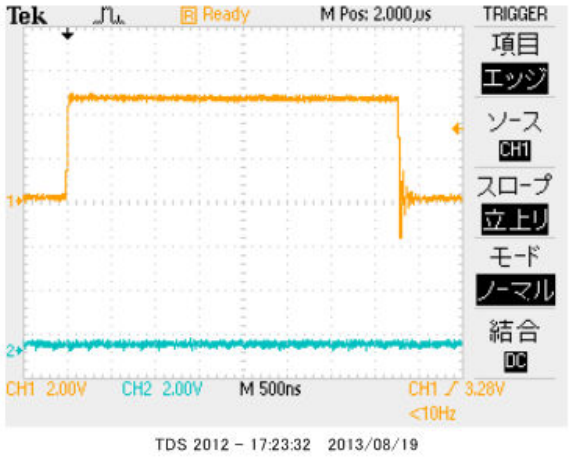
ＲＬ７８の場合　約３．８μｓｅｃでした。

ソースファイル

```
64
65                          unsigned long ldata1,ldata2,ldata3,ldata4;
66
67                          void main(void)
68                          {
69      001f1                   R_MAIN_UserInit();
70                              /* Start user code. Do not edit comment generated here */
71                              while (1U)
72                              {
73      001f5                       ldata1 = 1234;
74      001ff                       ldata2 = 5678;
75      00209                       P2 = 0xff;
76      0020c                       ldata3 = ldata1 * ldata2;
77      0022b                       ldata4 = ldata2 / ldata1;
78      0024a                       P2 = 0x0;
```

ソース＋逆アセンブラ

```
73:                     ldata1 = 1234;
   001f5    30d204      MOVW    AX,#4D2H
   001f8    bfbaef      MOVW    !_ldata1,AX
   001fb    f6          CLRW    AX
   001fc    bfbcef      MOVW    !0EFBCH,AX
74:                     ldata2 = 5678;
   001ff    302e16      MOVW    AX,#162EH
   00202    bfbeef      MOVW    !_ldata2,AX
   00205    f6          CLRW    AX
   00206    bfc0ef      MOVW    !0EFC0H,AX
75:                     P2 = 0xff;
   00209    cd02ff      MOV     P2,#0FFH
76:                     ldata3 = ldata1 * ldata2;
   0020c    afbaef      MOVW    AX,!_ldata1
   0020f    bdd8        MOVW    _@STBEG,AX
   00211    afbcef      MOVW    AX,!0EFBCH
   00214    bdda        MOVW    _@RTARG2,AX
   00216    afbeef      MOVW    AX,!_ldata2
   00219    bddc        MOVW    _@RTARG4,AX
   0021b    afc0ef      MOVW    AX,!0EFC0H
   0021e    fd8501      CALL    !@@lsmul
   00221    adda        MOVW    AX,_@RTARG2
   00223    bfc4ef      MOVW    !0EFC4H,AX
   00226    add8        MOVW    AX,_@STBEG
   00228    bfc2ef      MOVW    !_ldata3,AX
77:                     ldata4 = ldata2 / ldata1;
   0022b    afbeef      MOVW    AX,!_ldata2
   0022e    bdd8        MOVW    _@STBEG,AX
   00230    afc0ef      MOVW    AX,!0EFC0H
   00233    bdda        MOVW    _@RTARG2,AX
   00235    afbaef      MOVW    AX,!_ldata1
   00238    bddc        MOVW    _@RTARG4,AX
   0023a    afbcef      MOVW    AX,!0EFBCH
   0023d    fdb001      CALL    !@@ludiv
   00240    adda        MOVW    AX,_@RTARG2
   00242    bfc8ef      MOVW    !0EFC8H,AX
   00245    add8        MOVW    AX,_@STBEG
   00247    bfc6ef      MOVW    !_ldata4,AX
78:                     P2 = 0x0;
   0024a    f402        CLRB    P2
```



以上の結果をまとめると

ＣＰＵコア	クロック	ポートアクセス	乗除演算
ＲＬ７８	３２ＭＨｚ	６．３８ＭＨｚ	３．８μｓｅｃ
Ｈ８－３００Ｈ	２０ＭＨｚ	０．８２ＭＨｚ	３０μｓｅｃ
Ｒ８Ｃ	２０ＭＨｚ	０．６６ＭＨｚ	１５．５μｓｅｃ
結論		ＲＬ７８がＨ８－３００Ｈの７．７倍、Ｒ８Ｃの９．６倍高速。	ＲＬ７８がＨ８－３００Ｈの７．８倍、Ｒ８Ｃの４倍高速。

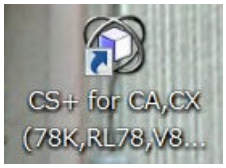
※測定結果はいずれも弊社製品比較です。

一般に設計が新しいCPUの方が、製造プロセスが微細化されている分、同じ機能であれば安価に製造できます。
RL78は従来より優れたアーキテクチャのコアに、乗除・積和演算器、10進補正回路等、高度な機能も内蔵し、かつ、今までより低消費電力、安価を目指して開発されたようです。

結論として、従来、Ｈ８／３０４８等をご使用の方々にも安心して使っていただける性能をもったＣＰＵだと思います。

１－２　　動作、デバック

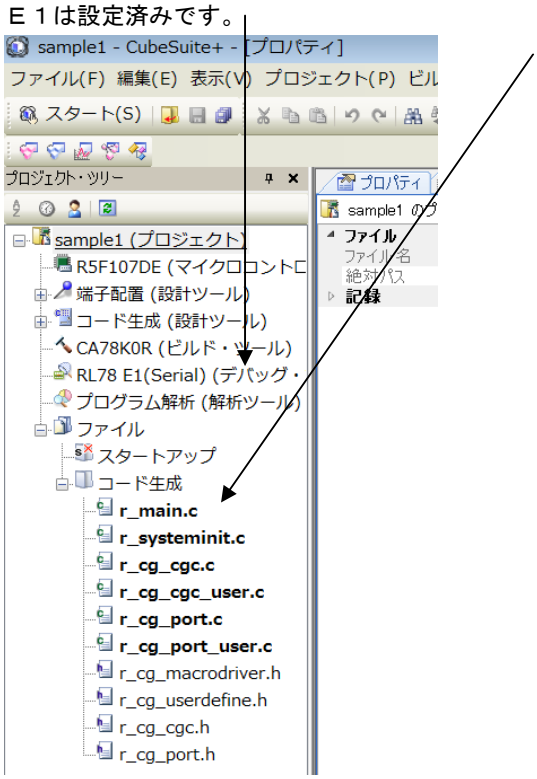
ａ：ＣＳ＋起動、コンパイル、書き込み、動作



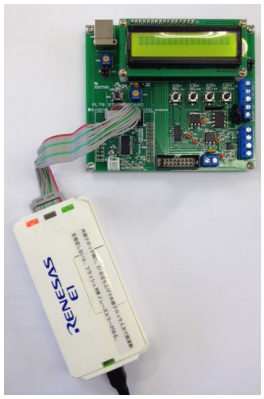
ＣＤに添付しているサンプルプログラムを使って、コンパイル、書き込み、動作の方法を示します。

ＣＳ＋を起動します。ここでは例としてＲＬ７８ＳＴＵＤＹ￥ｓａｍｐｌｅ１を動作させます。キーを押すと上のＬＥＤが点滅するプログラムです。
初めてのときは　ファイル　→　ファイルを開く　→　ｓａｍｐｌｅ１．ｍｔｐｊをダブルクリックします。

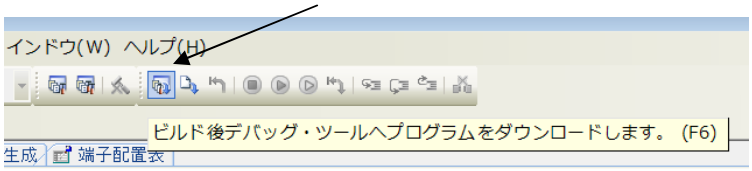
プロジェクトツリーが表示されます。ｒ＿ｍａｉｎ．ｃをダブルクリック。
Ｅ１は設定済みです。



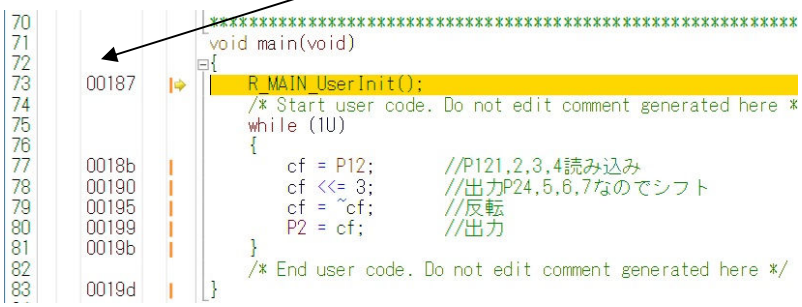
ｒ＿ｍａｉｎ．ｃが中央に表示されます。とりあえず、実行してみます。Ｅ１のケーブルを基板のＣＮ１に挿入します。電源はＥ１から供給しますので、不要です。（写真ご参考）



「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。

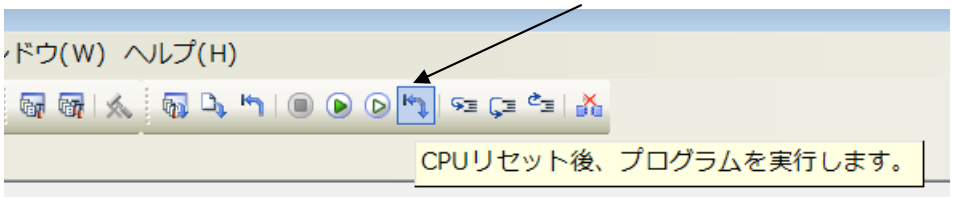


上手く転送できると、今まで表示されていなかったプログラムの絶対アドレスが表示されます。Ｅ１から電源がＣＰＵ基板に供給されます。



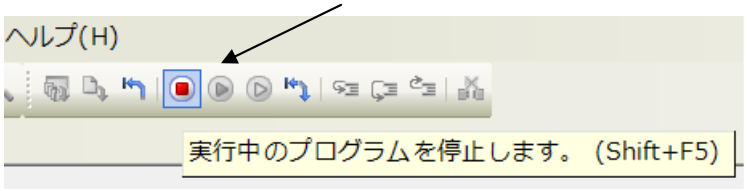
ここまでいかなかった場合、Ｅ１のインストゥールをご検証願います。

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。



キースイッチＳＷ１，２，３，４を押すと上のＬＥＤが点灯したら正常に動作しています。ＣＳ＋の右下部にも表示されます。

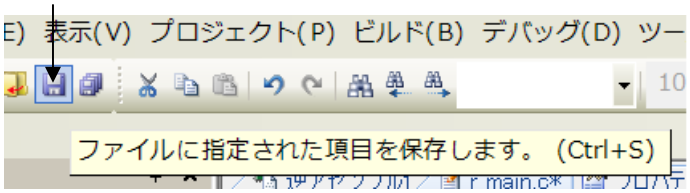
ここまで確認できましたら、一度止めます。



ｍａｉｎ関数のＩｗａｉｔのシフト数値３を２に書き直して

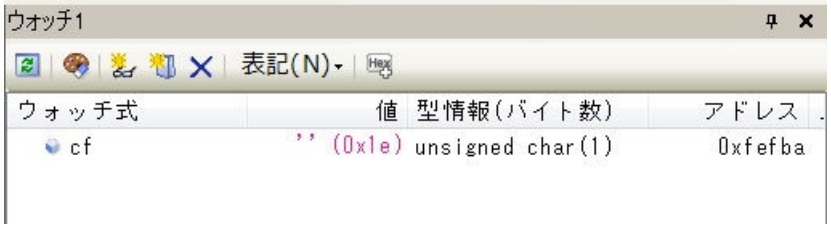
```
71                  void main(void)
72                  {
73      00187           R_MAIN_UserInit();
74                      /* Start user code. Do not edit comment generated here */
75                      while (1U)
76                      {
77      0018b               cf = P12;       //P121,2,3,4読み込み
78      00190               cf <<= 3;       //出力P24,5,6,7なのでシフト
79      00195               cf = ~cf;       //反転
80      00199               P2 = cf;        //出力
81      0019b           }
82                      /* End user code. Do not edit comment generated here */
83      0019d       }
```

セーブして



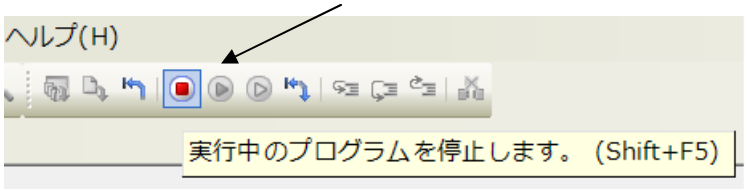
さきほどの、
「ビルド後、デバック・ツールへプログラムを転送」をクリック。
「ＣＰＵリセット後、プログラムを実行」をクリック。
ＬＥＤの点灯が先ほどより、１つずれたのが目視できましたでしょうか？

なお、動作中に右側にあるウオッチ１の変数ｃｆの値がちらちら変化します。



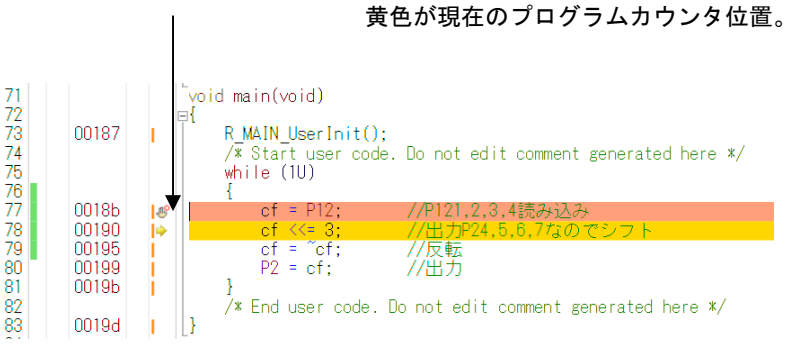
Ｐ１２のデータを読み込んだり、その値を３ビット左にシフトしたり、反転させたりしているので、値が変動します。

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。

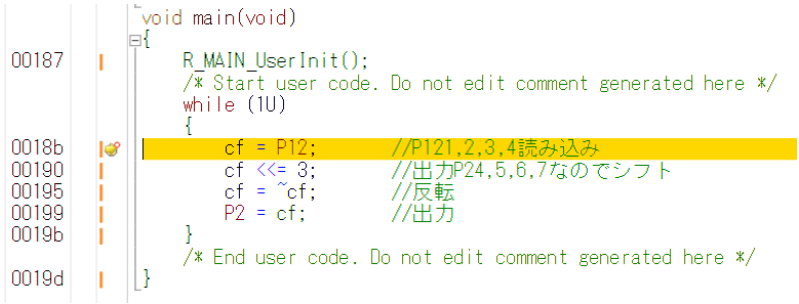


マウスを 0018b 番地にもって行き、左クリックで設定です。手の形が出ます。解除は設定後、再クリック。

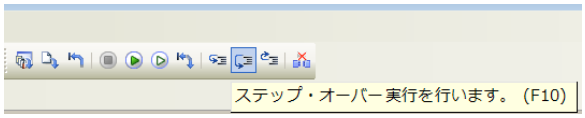
黄色が現在のプログラムカウンタ位置。



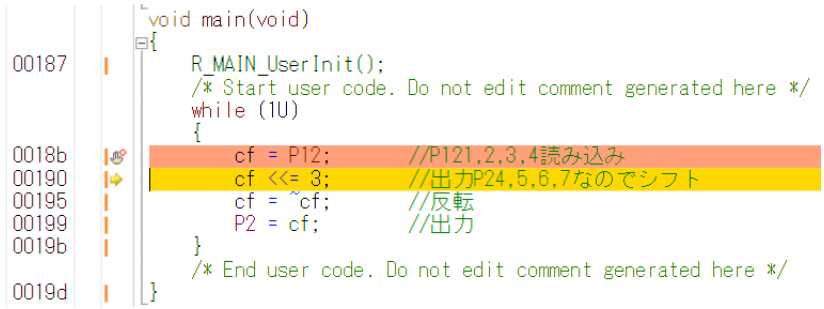
「ＣＰＵリセット後、プログラムを実行」をクリック。先ほど設定した行でプログラムカウンタが停止します。

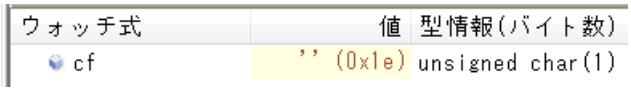


ステップオーバーで１行実行。



プログラムカウンターは 00190 で、１行前のｃｆ＝Ｐ１２が実行されました。





ｃｆの値、０ｘ１ｅがＰ１２を読み込んだ値です。２進数で書くと０ｂ０００１１１１０です。

更にステップオーバーで１行実行。

cf <<= 3;

を実行します。ｃｆの値は０ｘ１ｅより変化しませんが、実際は３ビット左シフトで０ｂ１１１１００００＝０ｘｆ０になっています。

更にステップオーバーで１行実行。

cf＝~cf;　//反転

０ｘｆ０がビット反転しｃｆは０ｘ０ｆとなりました。２進数で書くと０ｂ００００１１１１です。

更にステップオーバーで１行実行。

P2＝cf;　//出力

Ｐ２０	１
Ｐ２１	１
Ｐ２２	１
Ｐ２３	１
Ｐ２４	０
Ｐ２５	０
Ｐ２６	０
Ｐ２７	０

ビットで見ると上記のようにｃｆのデータが出力されます。ＬＥＤ１～４はどれも光りません。Ｕ１　ＴＤ６２００３は１で出力がＯＮし、ＬＥＤがＯＮする回路になっています。

さて、cf<<＝3;

実行後、ｃｆの値が変わらないというおかしな状況がありました。１行プログラムを追加してみます。ｃｆ　＜＜＝　３の後に　ｃｆ　＝　ｃｆです。

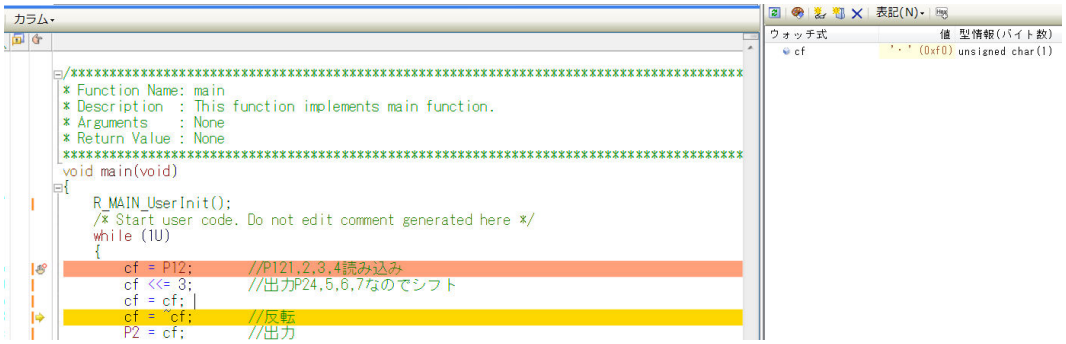
```
                void main(void)
                {
00187           |   R_MAN_UserInit();
                    /* Start user code. Do not edit comment generated here */
                    while (1U)
                    {
0018b           |       cf = P12;         //P121,2,3,4読み込み
00190           |       cf <<= 3;         //出力P24,5,6,7なのでシフト
00195           |       cf = cf;
00198           |       cf = ~cf;         //反転
0019c           |       P2 = cf;          //出力
0019e           |   }
                    /* End user code. Do not edit comment generated here */
001a0           |   }
```

プログラムをセーブ。

「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。

ブレークポイントで停止したら、ｃｆ＝ｃｆまでステップ実行。ウオッチ窓に正しくｃｆ＝０ｘｆ０と表示されました。デバッカ側がｃｆ＜＜＝３の間は値を読み込むことが出来ないようです。



とりあえず、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。

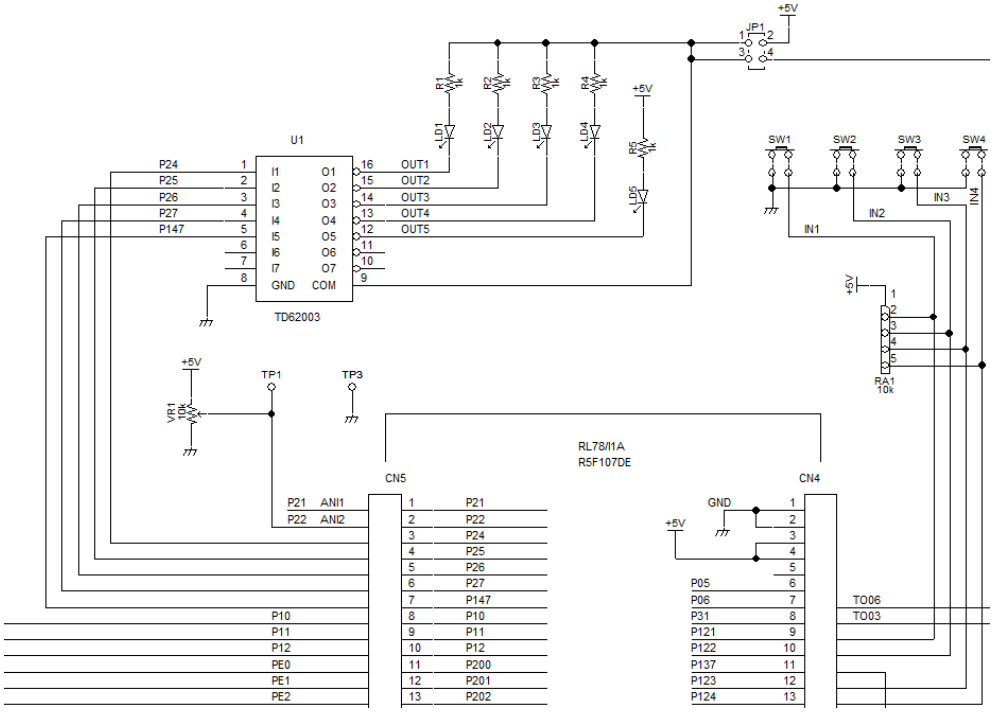
以下省略

２．サンプルプログラム

２－１　ｓａｍｐｌｅ１　キー入力でＬＥＤを点灯

【　概要　】

キーＳＷ１，２，３，４を押すと、ＬＥＤ１，２，３，４がそれぞれ点灯するソフトです。マイコンの基礎であるポートの入出力を学習します。



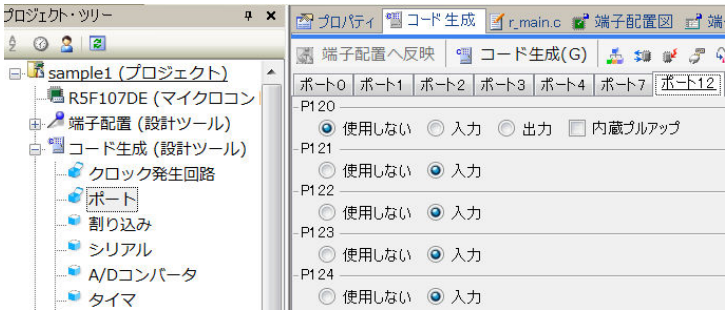
【　ハードウエア　】

キーはＲＡ１でプルアップされており、入力ポートＰ１２１，１２２，１２３，１２４は押されないとき１（＋５Ｖ）、押されたとき０（０Ｖ　ＧＮＤレベル）になります。出力ポートＰ２３，２５，２６，２７はトランジスタバッファＴＤ６２００３の入力に接続されていて、入力Ｉｘが１（＋５Ｖ）で出力Ｏｘがｏｎ（出力が０Ｖ近傍まで落ち、ＬＥＤが点灯します）、０（０Ｖ　ＧＮＤレベル）でＯＦＦ（出力が＋５Ｖ近傍まで上がり、ＬＥＤが消灯します）

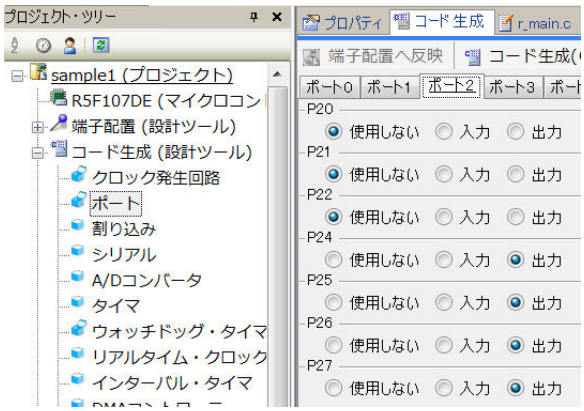
ｘ＝１～７

【　コード生成　ポートの設定　】

あらかじめ、ポートＰ１２１，１２２，１２３，１２４は入力に設定されています。



ポート２４，２５，２６，２７は出力に設定されています。



【　プログラム　】

```
①void main(void)
{
  ②   R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
  ③    while (1U)
    {
④          cf = P12;              //P121,2,3,4読み込み
⑤          cf <<= 3;              //出力P24,5,6,7なのでシフト
⑥          cf = ~cf;              //反転
⑦          P2 = cf;               //出力
    }
⑧    /* End user code. Do not edit comment generated here */
}
```

【　解説　】

①void main(void)

メイン関数です。電源ＯＮで自動的に実行されます。

②　R_MAIN_UserInit();

コード生成によって自動的に作られた初期設定関数をコールしています。この初期設定はメインルーチンの下（同じファイル）のところにあります。

③　while (1U)

以下の{　}の中を無限ループします。１Ｕは数字の１で、ｕｎｓｉｇｎｅｄ型であると明示しています。

④　　cf = P12;　//P121,2,3,4読み込み

ポート１２（１２０～１２４）のデータを読み込んで変数ｃｆにセットしています。

⑤　　cf <<= 3;　//出力P24,5,6,7なのでシフト

ｃｆを左に３ビットシフトしています。例えばｃｆが０ｘ１ｅ（２進数表記　０ｂ０００１１１１０）だった場合、０ｂ１１１１００００＝０ｘｆ０になります。

⑥　　cf = ~cf;　　//反転

反転ですので、例えば０ｘｆ０が０ｘ０ｆになります。

⑦　　P2 = cf;　　//出力

それをＰ２に出力しています。Ｕ１のＴＤ６２００３はトランジスタバッファで入力１で出力ＯＮ，ＬＥ

Ｄ　点灯します。ＬＥＤを光らせるだけでしたら、ポートから直接ＬＥＤを駆動しても可能ですが、Ｕ１の出力はサービスポートにも出力されていて、ポートでは無理な電流まで駆動することが出来ます。
Ｐ２ポート　ロウレベル出力電流最大１ｍＡ　ハイレベル出力電流最大　－０．５ｍＡ
ＴＤ６２００３　出力電流最大５００ｍＡ　ｈＦＥ　１０００（最小）
⑧ /* Start user code. Do not edit comment generated here */

/* End user code. Do not edit comment generated here */
前にも書きましたがこの間に挟むようにプログラムを書けば、途中で「コード生成」を再度行っても、既に書いたプログラムが消えることはありません。ここ以外は消えます。

【　演習　】
以上が理解できた方は、演習問題に進んで下さい。課題は
キー入力でＬＥＤを点灯、押されていても一定時間で消灯です。ｓａｍｐｌｅ１は押している間、継続してＬＥＤが点灯しましたが、ここではある時間をもって消灯してみてください。

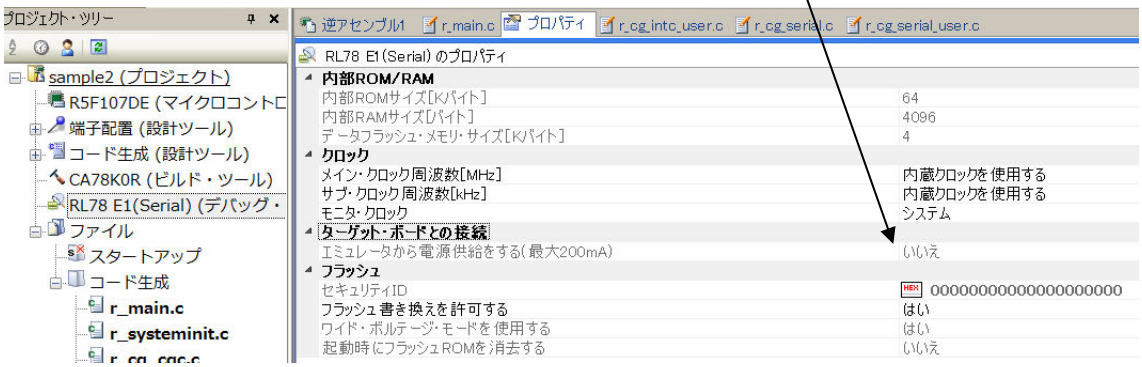
回答例　ｓａｍｐｌｅ１＿ａ

回答例は例であり、この通りに作る必要もありません。１０人居れば１０人、違うプログラムになるのが一般的です。

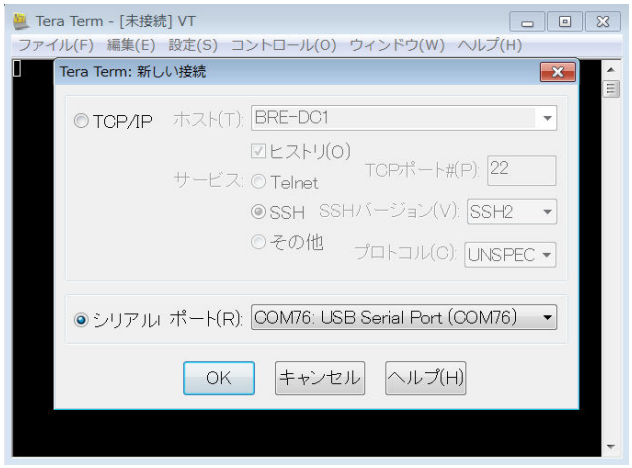
２－２　ｓａｍｐｌｅ２　ＵＳＢ通信　ＡＢＣＤをパソコン側に送信

【　概要　】

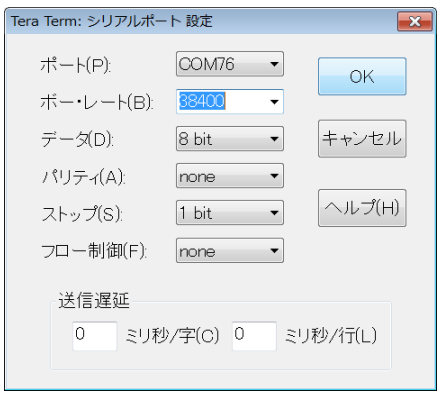
ＵＳＢ出力をパソコンと接続し、データのやり取りを行います。お手数ですが、テラタームやハイパーターミナルなどのターミナルプログラムを使用しますので、無い方は、ネットで検索し、インストゥール願います。例ではテラタームで行います。38400bps に設定して下さい。ＵＳＢケーブルでパソコンとつなげますとＣＰＵ基板、Ｅ１に電源が入ります。このプログラム以降はＥ１の設定　ターゲット・ボードとの接続　エミュレータから電源を供給する　いいえ　に設定されています。



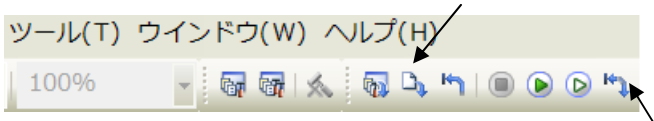
次に、テラタームを立ち上げて下さい。シリアルポートで「ＣＯＭｘｘ：ＵＳＢ　Ｓｅｒｉａｌ　Ｐｏｒｔ」を探して、「ＯＫ」をクリック。



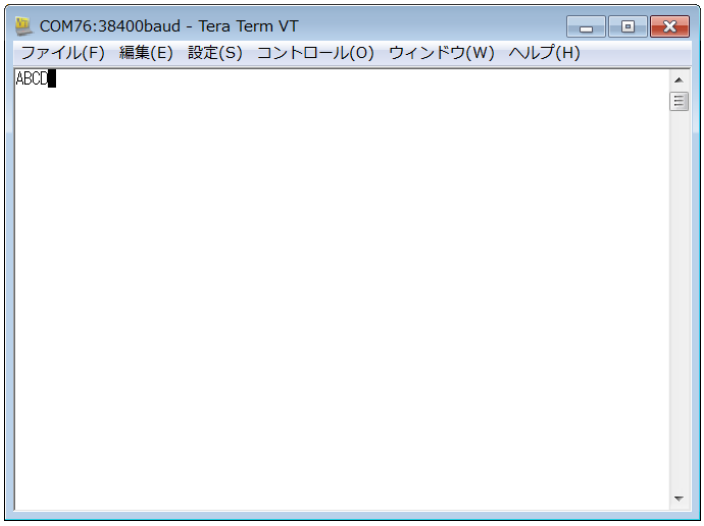
設定→シリアルポート→ボーレイトを３８４００にします。



プログラムをターゲットにダウンロードし

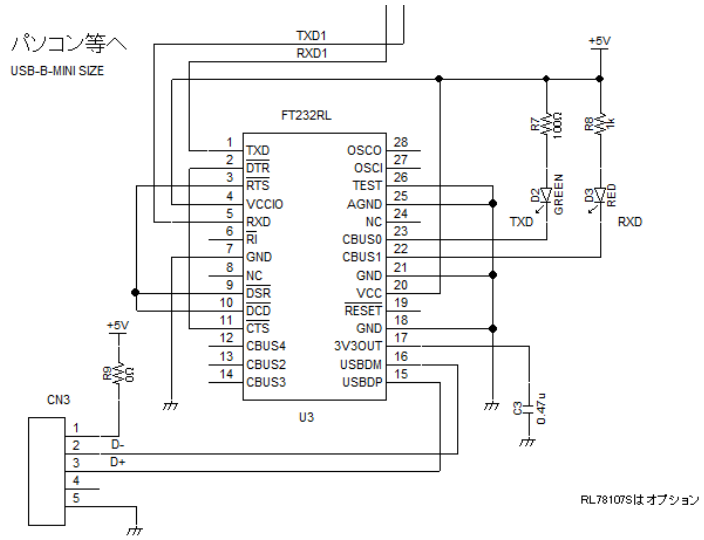


「リセットから実行」で　ＡＢＣＤと表示されれば正常に転送できました。



【　ハードウエア　】

マイコンのシリアルポート　ＴＸＤ１（Ｐ０２）、ＲＸＤ１（Ｐ０３）をＦＴ２３２ＲＬでＵＳＢに変換して、送受信します。難しいＵＳＢのやりとりはＦＴ２３２ＲＬが行いますので、プログラム作成側はＲＬ７８のシリアルアイオーの知識のみで操作できます。



以下省略

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　