ＲＸ２３０　マイコン学習セット　マニュアル　入門編

第１版　２０２２．４．２７

【　製品概要　】

本マニュアルはＲＸ２３０　Ｒ５Ｆ５２３０６ＡＤＦＭ（６４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、ソフトウエアコピー手順、添付ＣＤのサンプルプログラムの動作について解説しています。

●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Ｃ言語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。

●マイコンと各デバイスとのやりとりに標準的なＩ２Ｃ、ＳＰＩインターフェイスを使用、ライブラリの使用法を具体例で示し、解説しています。

●ルネサスエレクトロニクス社の統合開発環境ＣＳ+　ｆｏｒ　ＣＣ　における開発方法について記述してあります。

※本学習セット開発にはルネサスエレクトロニクス社製Ｅ２Ｌｉｔｅ（Ｅ２Ｌ）が別途必要です。

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：ＢＣＲＸ２３０　ＣＰＵ部の特徴

ｃ：Ｅ２　Ｌｉｔｅ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

ｆ：ＲＸ２３０とＲＬ７８、ＲＸ７１Ｍ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：ＰＥ０を出力にする初期設定
ｂ－２：自動生成されたプログラム
ｂ－３：Ｅ２Ｌの設定
ｂ－４：コード生成後の初期設定の変更
ｂ－５：変数変化を実行中に見る　ウオッチ窓の使い方
ｂ－６：新しいプログラムを作る簡単な方法

２．サンプルプログラム

２－１．Ｉ／Ｏ入出力　　ｓａｍｐｌｅ１
プログラム　　：キー入力でＬＥＤが点滅
演習プログラム　　：ｓａｍｐｌｅ１＿ｂ　押されたキーのLEDを点滅

２－２．Ａ／Ｄ変換　　ｓａｍｐｌｅ２
プログラム　　：１２ｂｉｔＡ／Ｄ変換器のデータを有機ＥＬに表示
Ｉ２Ｃインターフェイス
演習プログラム　　：ｓａｍｐｌｅ２＿ａ　Ａ／Ｄ変換の値を平均する

２－３．Ｄ／Ａ変換　　ｓａｍｐｌｅ３
プログラム　　：１２ｂｉｔ　Ｄ／Ａ変換器のアナログ出力で作る三角波
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．ＦＲＡＭ　　ｓａｍｐｌｅ４
プログラム　　：ＦＲＡＭの読み書き　ＳＰＩインターフェイス
演習プログラム　　：データを変えて、読み書きできるか確認

２－５．温度センサ　　ｓａｍｐｌｅ５
プログラム　　：温度センサＭＣＰ９８０１を使う　Ｉ２Ｃインターフェイス
演習プログラム　　：ＭＣＰ９８０１を１０ｂｉｔデータにして温度表示

２－６．ＰＷＭ　　ｓａｍｐｌｅ６
プログラム　　：ＰＷＭを使ってＬＥＤの明るさを変える
演習プログラム　　：明暗の周期を２．５秒から０．２５秒に変える

２－７．ＵＳＢ通信 ｓａｍｐｌｅ７
プログラム　　：パソコン側で打ち込んだキーを受信し、返送する
演習プログラム　　：学習ボードのキーを押してパソコン側にＡＢＣ他を表示する

１−１．学習環境

ａ：学習セット同梱物

ＲＸ２３０学習ボード	１
ＤＶＤ（サンプルプログラム、ドキュメント）	１
マニュアル（本誌） 入門、応用	各１
ＵＳＢケーブル	１
モーター	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Ｅ２Ｌ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．Ｅ２Ｌ＋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、学習ボードを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。

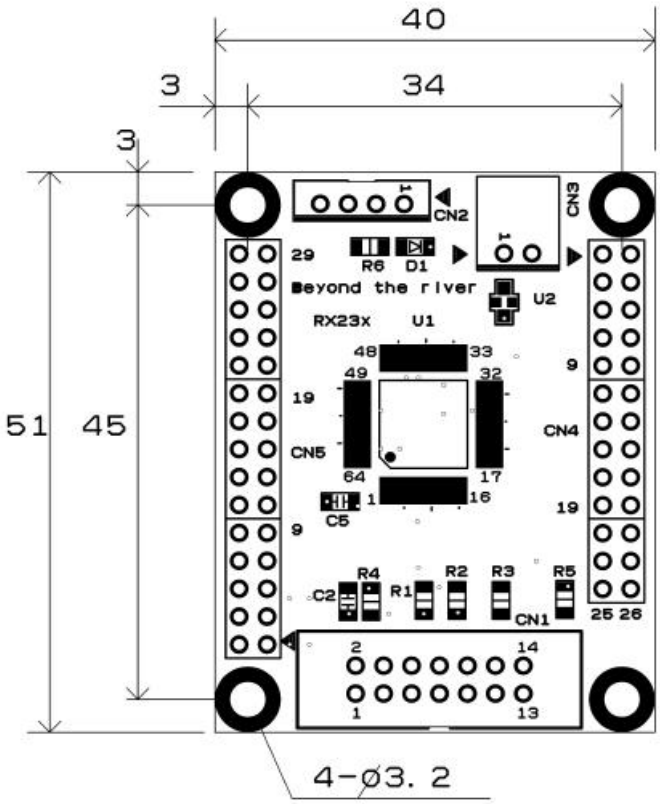
ｂ：ＢＣＲＸ２３０　ＣＰＵボード部の特徴
学習ボードのマイコン部分は弊社ＢＣＲＸ２３０　ＣＰＵボードと同じです。

●ＣＰＵ特徴　Ｒ５Ｆ５２３０６ＡＤＦＭ　　ＲＸ２３０

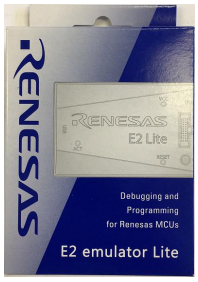
■ 32 ビット RXv2 CPU コア内蔵　最大動作周波数 54MHz／ 88.56 DMIPS の性能(54MHz 動作時)　DSP 強化：32 ビット積和、16 ビット積差命令に対応● FPU 搭載：32 ビット単精度浮動小数点（IEEE754 に準拠）● 除算器（最速 2 クロックで実行）● 高速割り込み● 5 段パイプラインの CISC ハーバードアーキテクチャ● 可変長命令形式：コードを大幅に短縮● オンチップデバッグ回路内蔵● メモリプロテクションユニット（MPU）対応

■64 ピン　10×10mm　0.5mmピッチ　IC
■消費電力低減機能　1.8V ～ 5.5V 動作の単一電源• バッテリバックアップ専用電源で動作可能な RTC• 3 種類の低消費電力モード• ソフトウェアスタンバイ中も動作する LPT（ローパワータイマ）•
■データ転送機能　DMAC：4 チャネル内蔵• DTC：4 種類の転送モード•
■ ELC　割り込みを介さず、イベント信号でモジュール動作が可能• CPU スリープ状態において、モジュール間のリンク動作が可能•
■リセットおよび電源電圧制御　パワーオンリセット（POR）など 8 種類のリセットに対応• 低電圧検出機能（LVD）の設定可能•
■クロック機能　メインクロック発振子周波数：1 ～ 20MHz• 外部クロック入力周波数：～ 20MHz• サブクロック用発振子周波数：32.768kHz• PLL 回路入力 4MHz ～ 12.5MHz• 低速オンチップオシレータ、高速オンチップオシレータ、IWDT 専• 用オンチップオシレータ内蔵　USB 専用 PLL 回路：4MHz、6MHz、8MHz、12MHz• システムクロック 54MHz USB クロック 48MHz の設定可能　32.768kHz RTC 専用クロックの生成• クロック周波数精度測定回路（CAC）内蔵•
■リアルタイムクロック内蔵　補正機能（30 秒、うるう年、誤差）• カレンダカウントモード / バイナリカウントモードを選択可能• 時間キャプチャ機能• 外部端子のイベント入力で時間をキャプチャ•
■独立ウォッチドッグタイマ内蔵　15kHz IWDT 専用オンチップオシレータクロック動作•
■最大 14 本の通信機能を内蔵　USB2.0 ホスト / ファンクション /OTG（ON-The-Go）（1 チャネル）、• フルスピード（12Mbps）、ロースピード（1.5Mbps）、アイソクロナ ス転送、BC（バッテリチャージャ）に対応　ISO11898-1 準拠の CAN（1 チャネル）最大 1Mbps 転送• 多彩な機能に対応した SCI（最大 7 チャネル）調歩同期式モード /• クロック同期式モード / スマートカードインタフェースモード ビットモジュレーション機能による通信誤差低減　IrDA インタフェース（1 チャネル、SCI5 と連携）• I2C バスインタフェース　最大 400kbps 転送　SMBus に対応•（1 チャネル）　RSPI（1 チャネル）　最大 16Mbps 転送• シリアルサウンドインタフェース（1 チャネル）• SD Host I/F（オプション :1ch）SD メモリ /SDIO 1 or 4 ビット SD• バスをサポート 注 . 48 ピン版は 1 ビットモードのみ
■最大 20 本の拡張タイマ機能　16 ビット MTU：インプットキャプチャ、アウトプットコンペア、• 相補 PWM 出力、位相計数モード（6 チャネル）　16 ビット TPU：インプットキャプチャ、アウトプットコンペア、• 位相計数モード（6 チャネル）　8 ビット TMR（4 チャネル）• 16 ビット CMT（4 チャネル）
■ 12 ビット A/D コンバータ内蔵　最小 0.83μs 変換が可能• 24 チャネル• チャネルごとにサンプリング時間を設定可能• 自己診断機能 / アナログ入力断線検出アシスト機能内蔵
■ 12 ビット D/A コンバータ内蔵　2 チャネル
■汎用入出力ポート内蔵　5V トレラント、オープンドレイン、入力プルアップ、駆動能力切• り替え機能
■セキュリティ機能（TSIP-Lite）　暗号エンジンへの不正アクセスを禁止し、成りすまし、改ざんを防止鍵の安全な管理を提供• AES（鍵長 128/256bit）内蔵。ECB,CBC,GCM 他に対応• 真正乱数発生回路内蔵•
■温度センサ内蔵 ■動作周囲温度　—40 ～+ 85 ℃• —40 ～+ 105 ℃
■用途　一般産業、民生機器

ＣＰＵ部大きさ（部品面）



ｃ：Ｅ２　Ｌｉｔｅ　エミュレータ



概要

Ｅ２Ｌｉｔｅエミュレータ（以降Ｅ２Ｌ）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、使い方も比較的短時間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　RL78,RX,RA,RE

ＥＺＬをＰＣのＵＳＢに接続するとｗｉｎｄｏｗｓが自動的にデバイスドライバをインスツールします。続いて、ネットから開発環境ＣｕｂｅＳｕｉｔｅ＋とＣコンパイラの最新版ダウンロードを行います。

ｄ：無償の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ＣＳ+　ｆｏｒ　ＣＣ　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→「ＣＳ＋　無償ダウンロード」の検索で表示されます。

以下省略

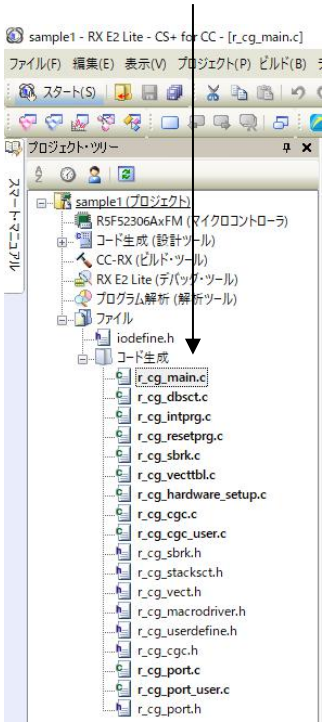
１－２　　動作、デバック

ａ：ＣＳ＋起動、コンパイル、書き込み、動作

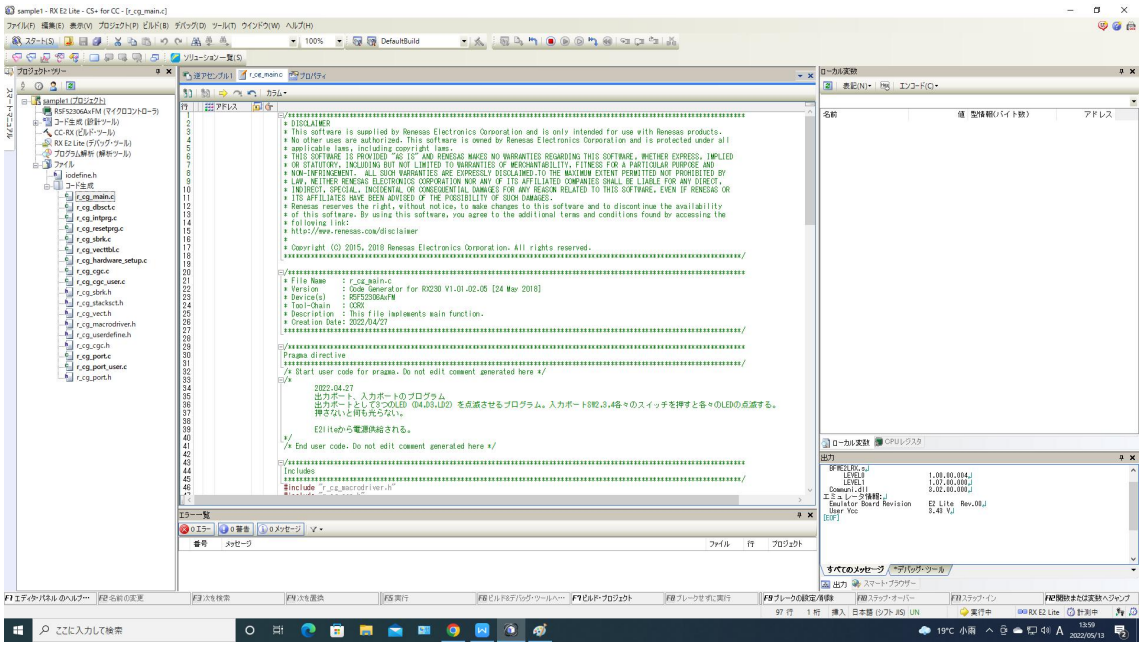
ＣＤに添付しているサンプルプログラムを使って、コンパイル、書き込み、動作の方法を示します。

ＣＳ＋を起動します。ここでは例としてＲＸ２３０ＳＴＵＤＹ￥ｓａｍｐｌｅ１を動作させます。キーを押すと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。



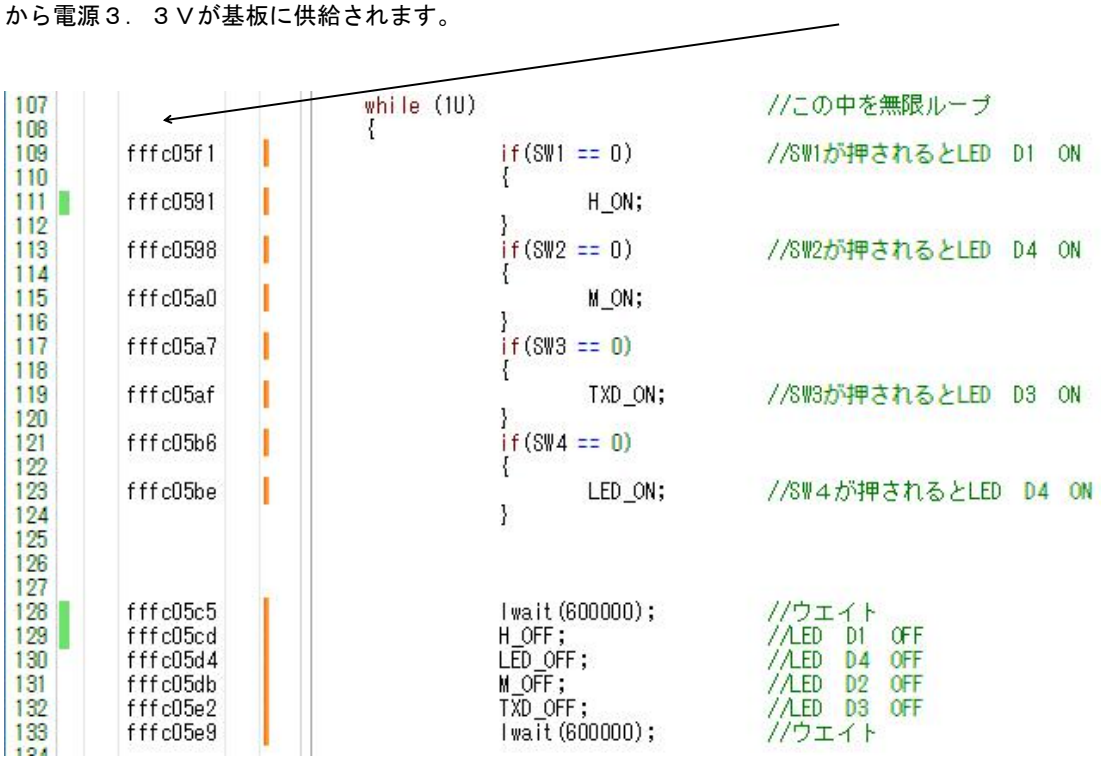
ｒ＿＿ｍａｉｎ．ｃが中央に表示されます。



とりあえず、実行してみます。Ｅ２Ｌのケーブルを基板のＣＮ１に挿入します。電源はＥ２Ｌから供給しますので、ＵＳＢ接続は不要です。（写真ご参考）

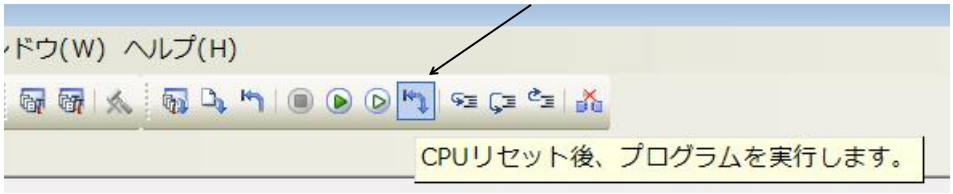
「デバック・ツールへプログラムを転送」をクリック。

　正常に転送できると、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。Ｅ２Ｌから電源３．３Ｖが基板に供給されます。



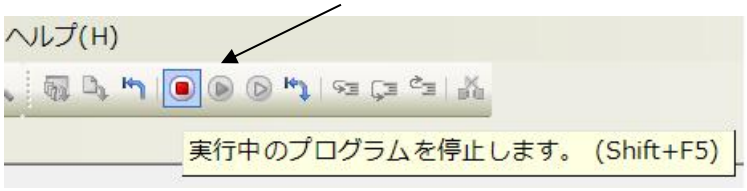
ここまでいかなかった場合、Ｅ２Ｌのインストゥールをご検証願います。

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。

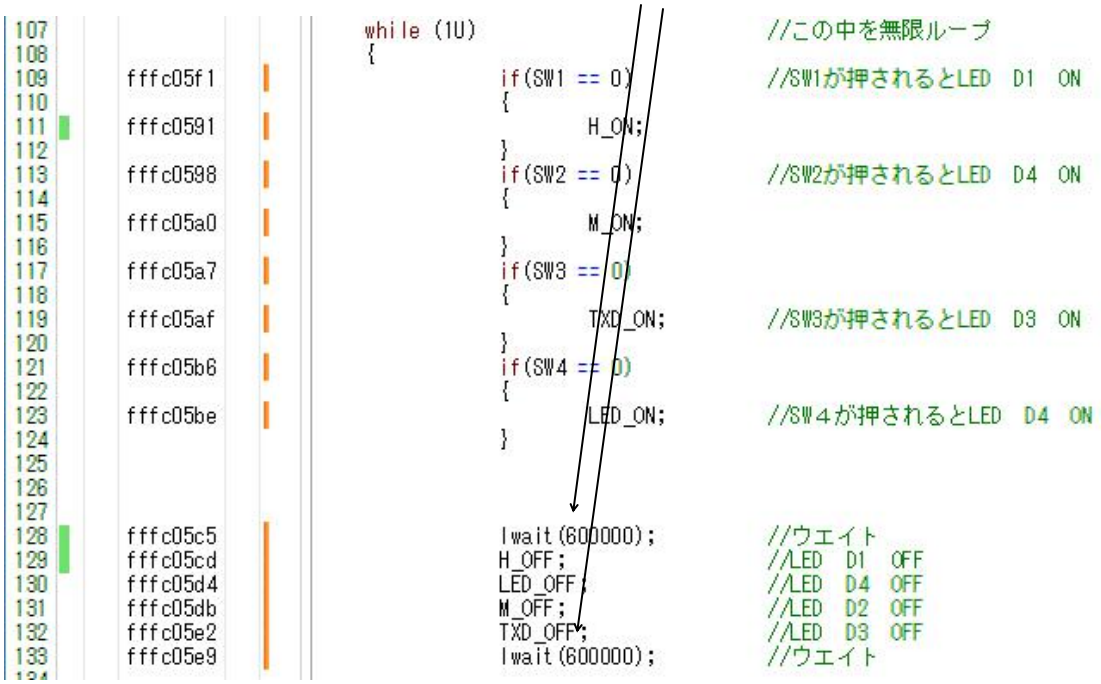


４つのＳＷ１，２，３，４を押すとＬＥＤ　Ｄ１，２，３，４が点滅したら正常に動作しています。ＣＳ＋の右下部「ＲＵＮ　実行中」が表示されます。

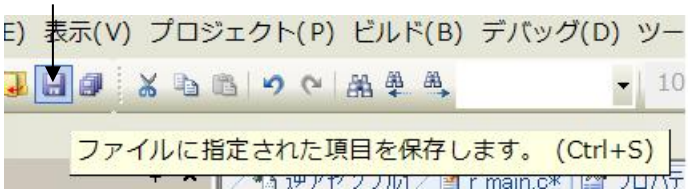
ここまで確認できましたら、一度止めます。



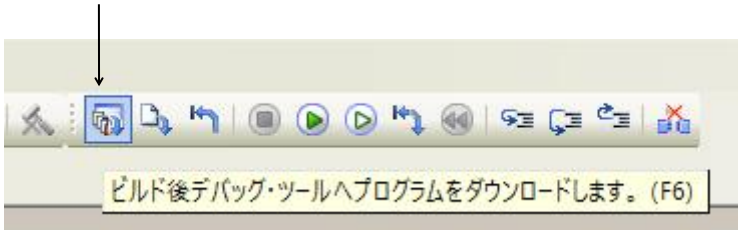
ｍａｉｎ関数のｗａｉｔの数値の２か所、６００００００の６を２に減らして（３倍速）



セーブして

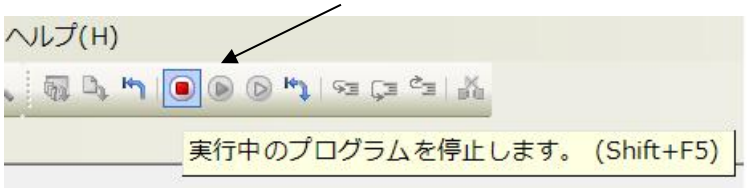


プログラムを変更しましたから「ビルド後、デバック・ツールへプログラムを転送」をクリック。



「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、早く点滅するのが目視できましたでしょうか？

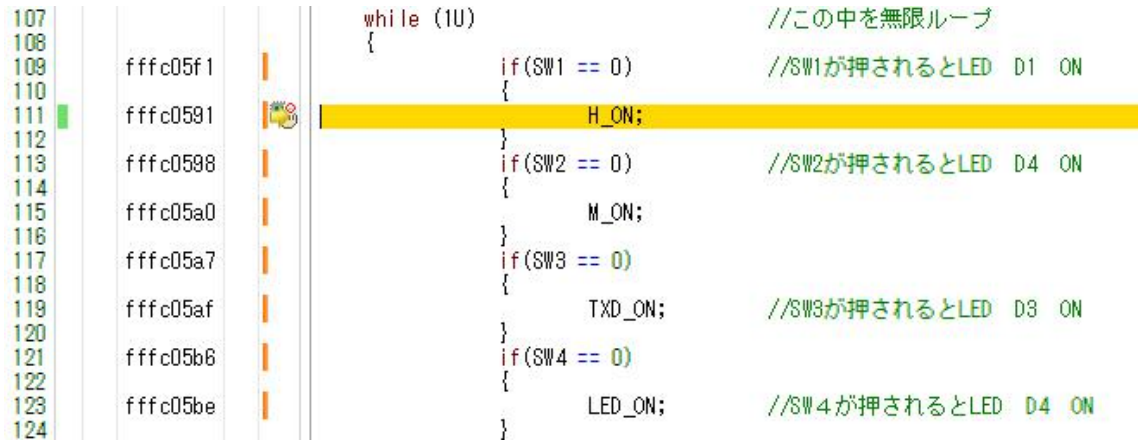
次に、ブレークポイントの設定を行ってみます。一度、プログラムを停止させます。



マウスを 111 番にもって行き、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。

「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ＳＷ１を押すとプログラムカウンタが停止します。黄色は現在のプログラムカウンタの位置です。まだコードは実行されていません。



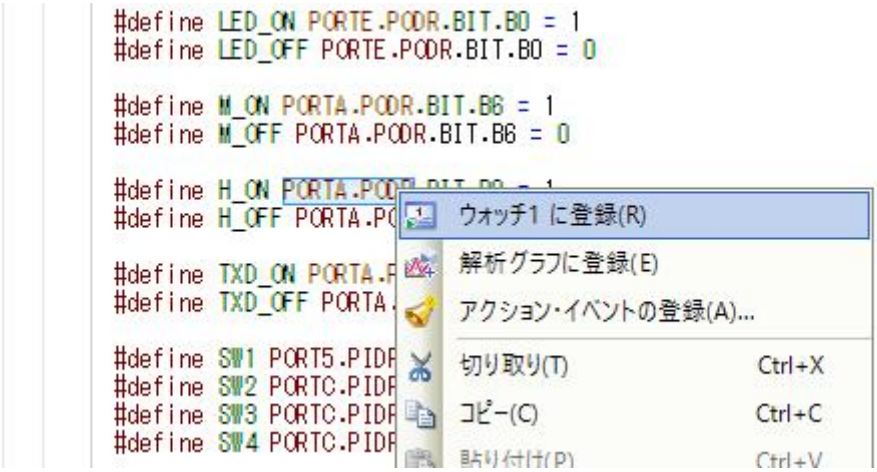
ステップオーバーで１行実行。


```
107                                  while (1U)                          //この中を無限ループ
108                                  {
109     fffc05f1                             if(SW1 == 0)                //SW1が押されるとLED  D1  ON
110                                          {
111     fffc0591                                     H_ON;
112                                          }
113     fffc0598                             if(SW2 == 0)                //SW2が押されるとLED  D4  ON
114                                          {
115     fffc05a0                                     M_ON;
116                                          }
117     fffc05a7                             if(SW3 == 0)
```

プログラムカウンターは 113 で、111 行のＨ＿ＯＮ　が実行されました。ＬＤ１の点灯が確認できると思います。引き続きステップオーバー実行で 130 行まで行くと 129 行の命令　H_OFF が実行されて LED　D1 が消灯するのを確認できます。

```
107                                  while (1U)                          //この中を無限ループ
108                                  {
109     fffc05f1                             if(SW1 == 0)                //SW1が押されるとLED  D1  ON
110                                          {
111     fffc0591                                     H_ON;
112                                          }
113     fffc0598                             if(SW2 == 0)                //SW2が押されるとLED  D4  ON
114                                          {
115     fffc05a0                                     M_ON;
116                                          }
117     fffc05a7                             if(SW3 == 0)
118                                          {
119     fffc05af                                     TXD_ON;             //SW3が押されるとLED  D3  ON
120                                          }
121     fffc05b6                             if(SW4 == 0)
122                                          {
123     fffc05be                                     LED_ON;             //SW４が押されるとLED  D4  ON
124                                          }
125
126
127
128     fffc05c5                             lwait(600000);              //ウエイト
129     fffc05cd                             H_OFF;                      //LED  D1  OFF
130     fffc05d4                             LED_OFF;                    //LED  D4  OFF
131     fffc05db                             M_OFF;                      //LED  D2  OFF
132     fffc05e2                             TXD_OFF;                    //LED  D3  OFF
133     fffc05e9                             lwait(600000);              //ウエイト
```

ポートのデータを見たいときは例えば



＃ｄｅｆｉｎｅ　Ｈ＿ＯＮ　の　ＰＯＲＴＡ．ＰＯＤＲの部分をマウスの左ボタンを押したままドラッグ、離して、右クリックで「ウオッチ１に登録」で現在のＰＯＲＴＡ．ＰＯＤＲの内容がウオッチ窓に表示されます。

ブレークポイントを解除し「プログラムを現在の位置から実行」すると

ウオッチ１窓にあるＰＯＲＴＡ．ＰＯＤＲの値がＳＷ１を押したときだけ、プログラムの進行に伴い、０ｘ１０、０ｘ１１と繰り返すのが確認できると思います。

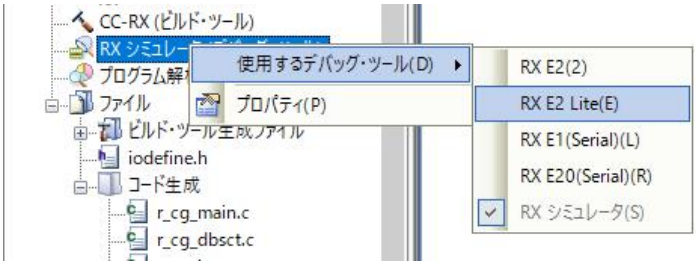
以上、大急ぎでしたが、ＣＳ＋　のプログラムの書き換え、コンパイル、Ｅ２Ｌへのダウンロード、実行、ブレークポイント設定、ウオッチ窓設定、動作の概要でした。

ｂ：新しいプログラムを作る

以下省略

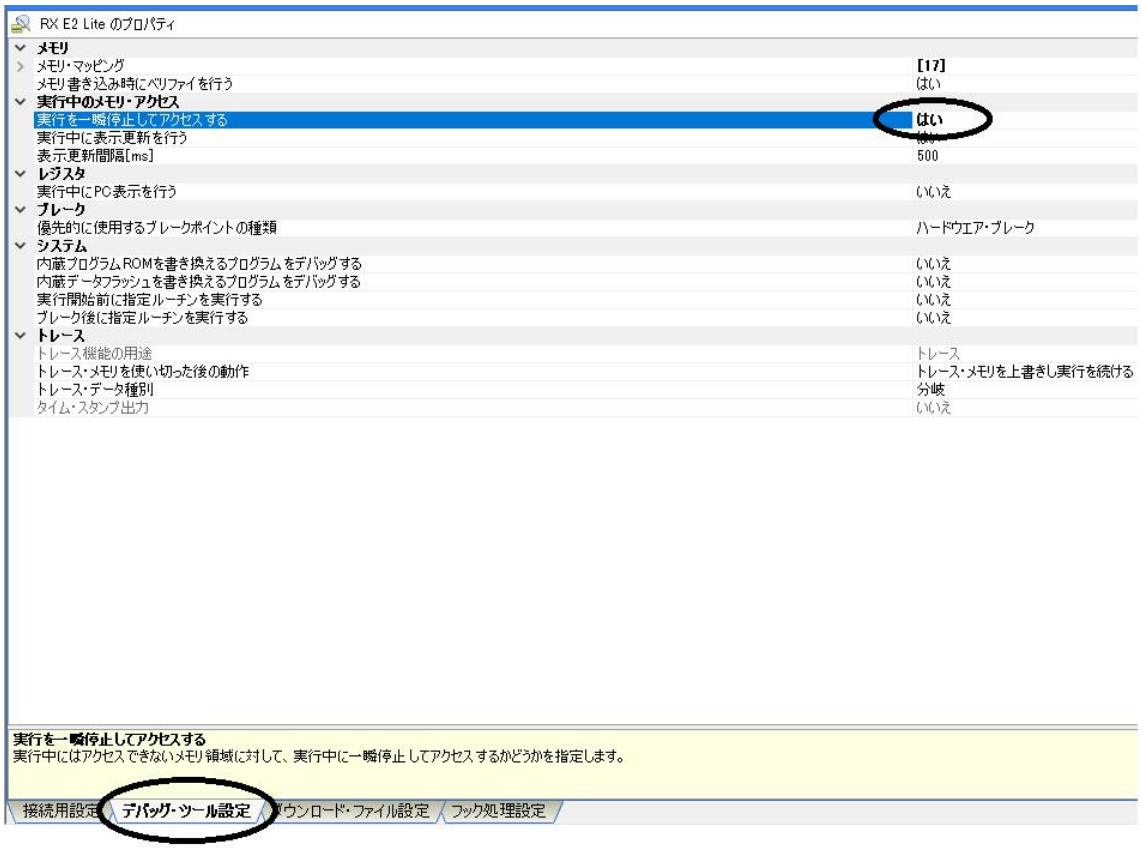
ｂ－３：Ｅ２Ｌの設定

出来上がったプログラムを実機にダウンロード、実行させるデバッカにはＥ２Ｌを使用します。初めはシュミレータになっているのでこれをＲＸ　Ｅ２　Ｌｉｔｅ（Ｅ）に変更します。



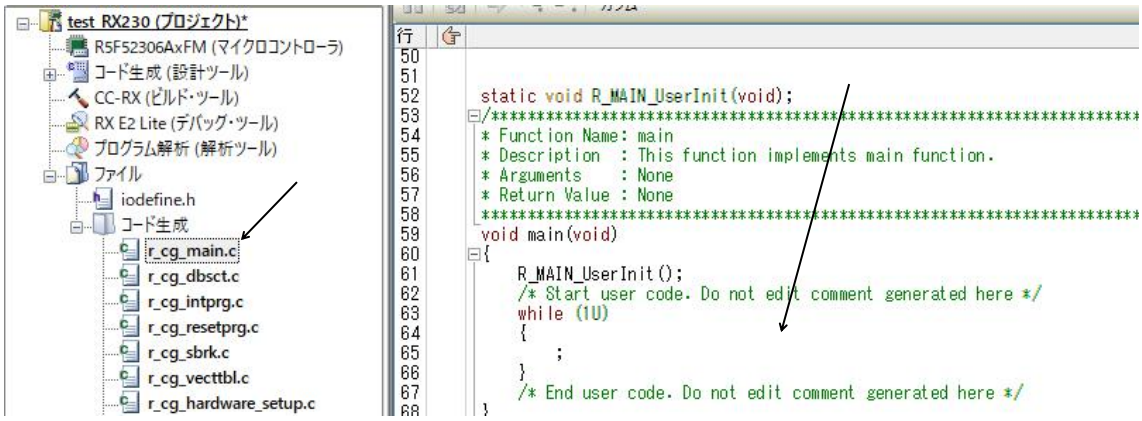
E2 Lite のプロパティ→　接続用設定でクロック、ターゲットボードとの設定を行います。

E2　Lite　のプロパティ→デバックツール設定　実行を一瞬停止してアクセスする→はい　にすると



プログラム実行中のポートのレジスタ値、メモリ、変数の変化等を複数リアルタイムに確認できます。

ここまで設定出来たら、ｒ＿ｃｇ＿ｍａｉｎ．ｃをクリックし、ｍａｉｎプログラムにプログラムを書きます。

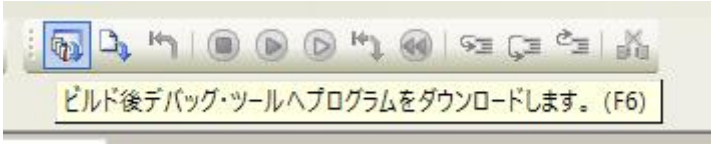


ポートをＯＮ／ＯＦＦする命令を書き、

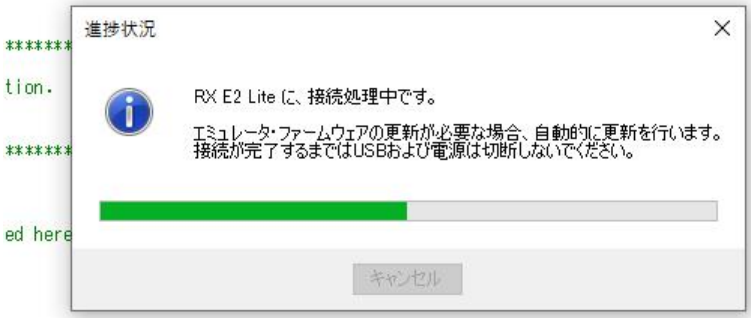
```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
    while (1U)
    {
            PORTE.PODR.BIT.B0 = 1;
            PORTE.PODR.BIT.B0 = 0;
    }
    /* End user code. Do not edit comment generated here */
}
```

Ｅ２Ｌと実機を接続し、

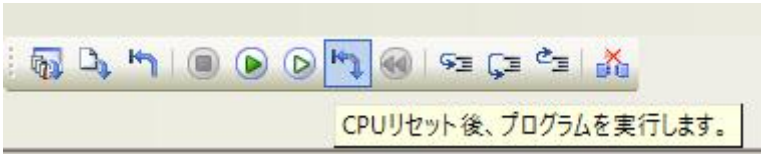
ビルド後、デバックツールへプログラムをダウンロード　をクリック。



プログラムにエラーが無かった場合、接続、ダウンロードが行われます。



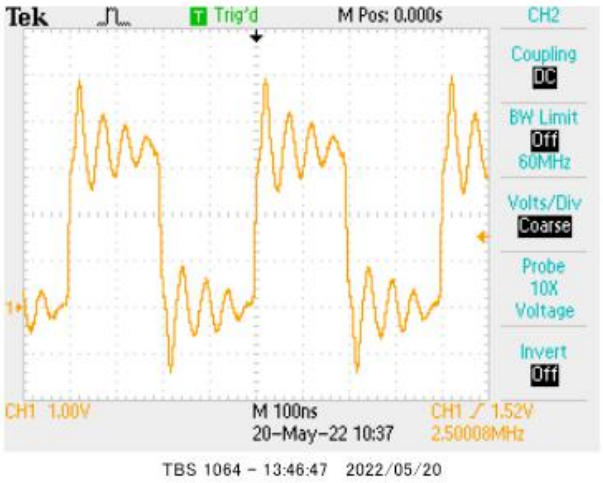
CPU リセット後、プログラムを実行。



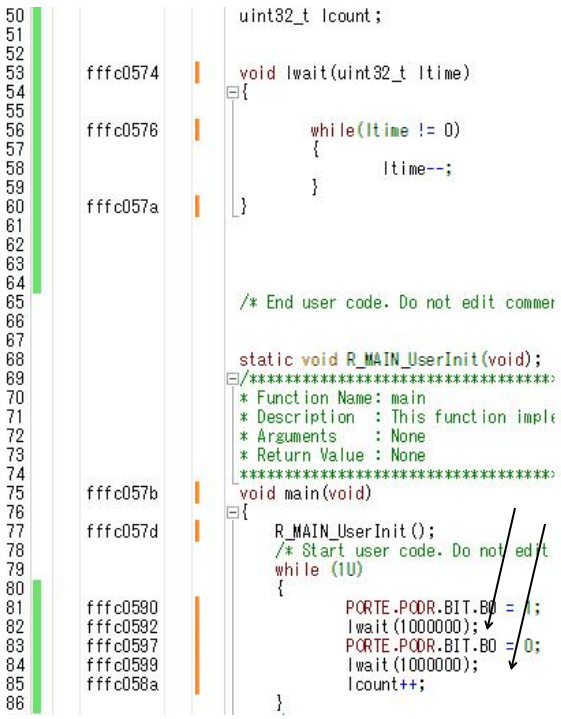
プログラム実行中は赤いマークと

右下にＲＵＮという緑の文字が移動表示され、その右に実行中と黒文字で表示されます。

オシロスコープがあればＴＰ１７を観測すればＰＥ０ポートが０，１を繰り返すのが確認できますが、目視でも出来るようにウエイト関数を入れてみます。



ウエイト関数を入れたプログラム



```
50                      uint32_t lcount;
51
52
53      fffc0574        void lwait(uint32_t ltime)
54                      {
55
56      fffc0576                while(ltime != 0)
57                              {
58                                      ltime--;
59                              }
60      fffc057a        }
61
62
63
64
65                      /* End user code. Do not edit commer
66
67
68                      static void R_MAIN_UserInit(void);
69                      /***********************************
70                      * Function Name: main
71                      * Description  : This function imple
72                      * Arguments    : None
73                      * Return Value : None
74                      ***********************************
75      fffc057b        void main(void)
76                      {
77      fffc057d            R_MAIN_UserInit();
78                          /* Start user code. Do not edit
79                          while (1U)
80                          {
81      fffc0590                    PORTE.PODR.BIT.B0 = 1;
82      fffc0592                    lwait(1000000);
83      fffc0597                    PORTE.PODR.BIT.B0 = 0;
84      fffc0599                    lwait(1000000);
85      fffc058a                    lcount++;
86                          }
```

これをセーブ、コンパイル、ダウンロード、実行させるとＬＥＤ　Ｄ４が目で見てわかるほどの点滅になります。

ｂ－４：コード生成後の初期設定の変更

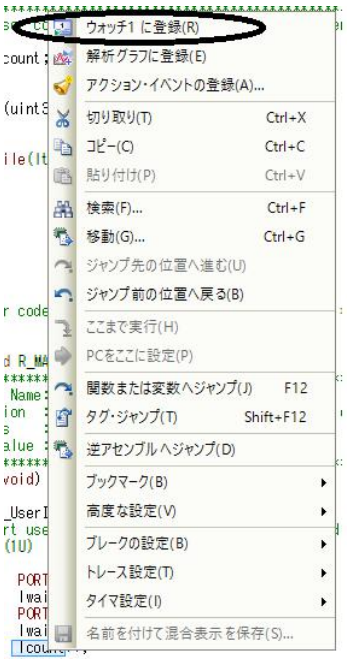
以下省略

ｂ－５：変数変化を実行中に見る　ウオッチ窓の使い方

動作中の変数を見る場合、見たい変数を左ドラッグし、

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
    while (1U)
    {
        PORTE.PODR.BIT.B0 = 1;
        lwait(1000000);
        PORTE.PODR.BIT.B0 = 0;
        lwait(1000000);
        lcount++;
    }
    /* End user code. Do not edit comment generated here */
}
```

右クリック→ウオッチ１に登録



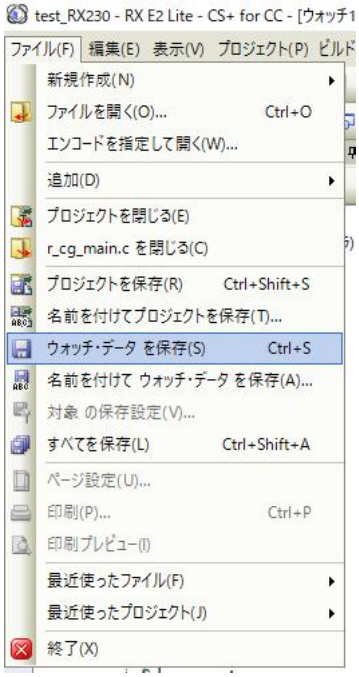
ウオッチ１にｌｃｏｕｎｔが表示され、プログラム実行と共に＋１された表示を見ることが出来ます。

同様方法で変数だけでなく、レジスタの値も読み込みできます。下記例では PORTE の出力レジスタ。

プログラムでＰＥ０が０，１と変化することが目視出来ます。

ｂ－６：ウオッチデータのセーブ、ロード

ウオッチ窓に作成したデータはセーブ、ロードできます。プログラムが停止している状態で、カーソルをウオッチ窓に持っていき、右クリック→ファイル→ウオッチ・データを保存



以下省略

２．サンプルプログラム

２－１　ｓａｍｐｌｅ１　キー入力でＬＥＤが点滅

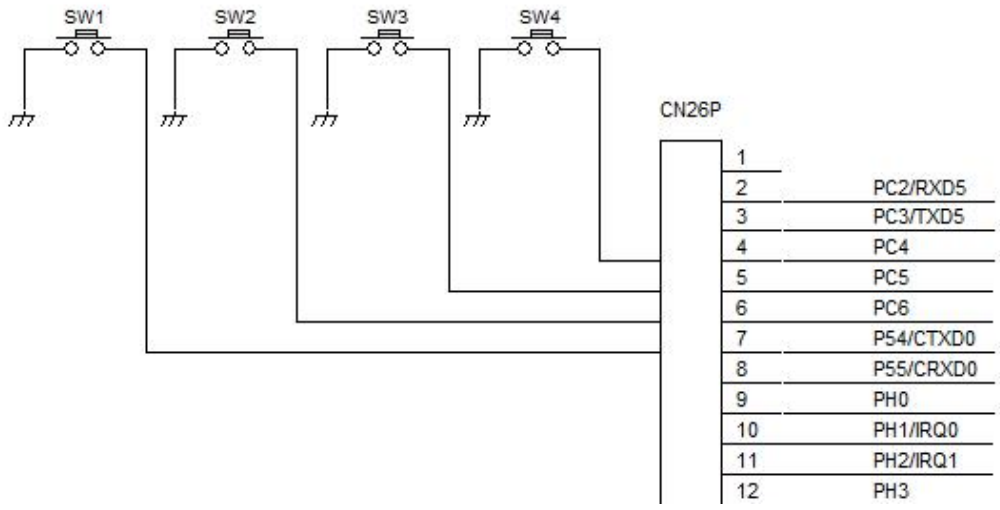
【　概要　】

出力ポート、入力ポートのプログラム

出力ポートとして４つのＬＥＤ（ＬＤ１，ＬＤ２，Ｄ３，Ｄ４）を点滅させるプログラム。入力ポートＳＷ１，２，３，４各々のスイッチを押している間、各々のＬＥＤが点滅します。電源はＥ２Ｌから供給されます。

【　ハードウエア　】

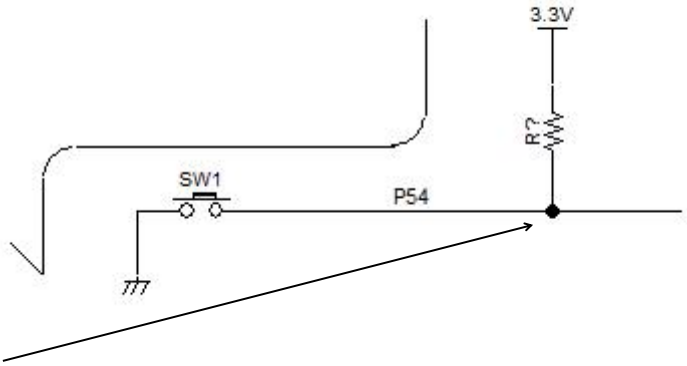
プッシュスイッチ　ＳＷ１，２，３，４は以下のようにＰＣ４，５，６，７に接続されています。



プッシュスイッチは押すと通電するタイプで、この回路図では押すとＧＮＤと同電位＝０Ｖとなります。

ＳＷ１が接続されているＰ５４のＩ／Ｏポート設定は　入力、内蔵プルアップとなっています。

ＳＷ２，３，４が接続されているＰＣ６，５，４も同じです。



Ｒ？　が内蔵プルアップを示します。ＳＷ１が押されると、ＳＷ１は通電しますのでＰ５４電圧レベルは０Ｖになります。離すとＰ５４は電源電圧＝３．３Ｖになります。プログラム的には押されたのを知るためにはＰ５４が０かどうか？を見ればいいことになります。外部に抵抗を付けなくて済むので、コスト安、基板を小さく出来るメリットがあります。

　内蔵プルアップはシリコン上に抵抗を作っているわけですが、抵抗値の精度は１０Ｋ～５０ＫΩと５倍もばらつきます。しかしプルアップという使い方では０または１が検出できればいいので、問題になりません。

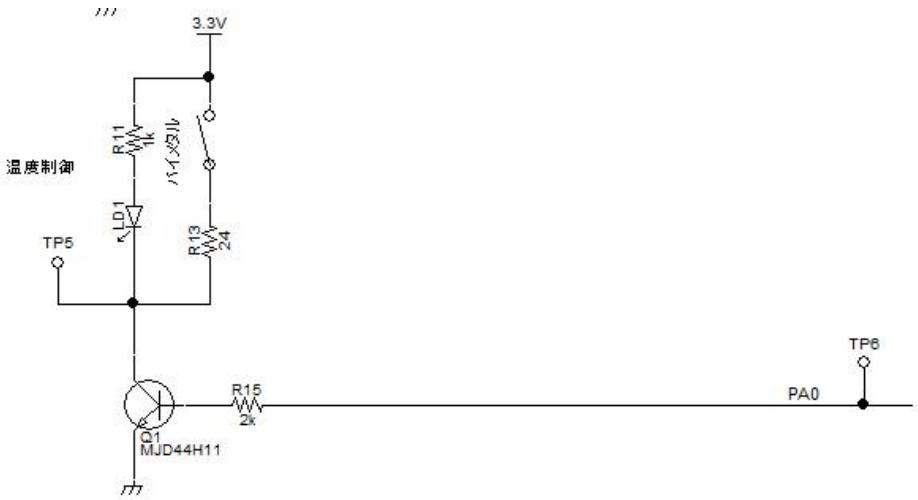
表50.6　　DC特性（4）
条件：1.8V≦VCC = VCC_USB = AVCC0≦5.5V、VSS = AVSS0 = VSS_USB = 0V、T_a = –40～+105°C

項目		記号	min	typ	max	単位	測定条件
入力プルアップ抵抗	全ポート（ポート35以外）	R_U	10	20	50	kΩ	V_{in} = 0V

```
#define SW1 PORT5.PIDR.BIT.B4

        if(SW1 == 0)                    //SW1 が押されると LED　D1　ON
```

```
{
        H_ON;
}
```



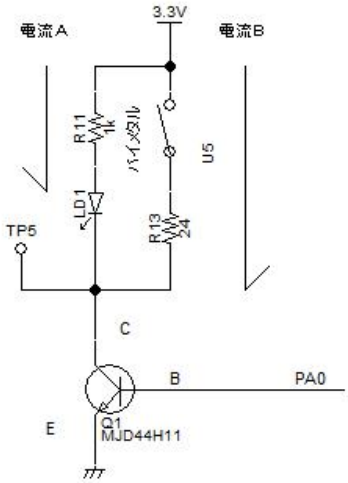
出力はＬＥＤ　Ｄ３，Ｄ４はポート直接、ＬＤ１、ＬＤ２はトランジスタＭＪＤ４４Ｈ１１を介してＬＥＤがドライブされています。トランジスタは電流増幅素子で、ポートが出力できる電流で足りないので追加されています。ポートが出力できる電流はハードいウエアマニュアル　５０．電気的特性　によれば通常で４ｍＡ、高駆動出力時で８ｍＡです。

RX230グループ、RX231グループ　　50. 電気的特性

表50.16　　出力許容電流値（1）
条件：1.8V≦VCC = VCC_USB = AVCC0≦5.5V、VSS = AVSS0 = VSS_USB = 0V、T_a = –40～+85°C

項目			記号	max	単位
出力Lowレベル許容電流 （1端子あたりの平均値）	ポート40～47、ポート03、05、07、ポート36、37		I_{OL}	4.0	mA
	それ以外のポート	通常出力時		4.0	
		高駆動出力時		8.0	

それに対して、Ｑ１にはＲ１３に流す電流Ａと、Ｒ１１に流す電流Ｂの合計が流れます（重ね合わせの理）。



Ｑ１がＯＮの時のＣ　コレクタ　－　Ｅ　エミッタ　間の抵抗を便宜上０Ωとすると、電流Ａは（３．３Ｖ　－　ＬＤ１のＶＦ）／１ＫΩ　です。ＬＤ１のＶＦ　順方向降下電圧を１．８Ｖとすると、
電流Ａ　＝　（３．３－１．８）／１ＫΩ　＝　１．５ｍＡ　となります。
電流Ｂはバイメタルの抵抗を０Ωとすると、３．３Ｖ／２４Ω　＝　１３７．５ｍＡ　となります。
Ｑ１が制御する電流　＝　１．５　＋　１３７．５　＝　１３９ｍＡ　となります。
１３９ｍＡ　＞＞　４ｍＡ　で、ポートだけではドライブできないのが明らかで、トランジスタＱ１が入れられています。ＬＥＤ　Ｄ３，Ｄ４は負荷がＬＥＤだけなので、ポートで直接ドライブしています。

【　出力ポートの設定、コード生成　】

出力ポートはＰＡ０，ＰＡ６，ＰＥ０，ＰＡ４は出力、ＣＭＯＳ出力に設定されています。

【　プログラム　】

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

①#define LED_ON PORTE.PODR.BIT.B0 = 1
#define LED_OFF PORTE.PODR.BIT.B0 = 0

#define M_ON PORTA.PODR.BIT.B6 = 1
#define M_OFF PORTA.PODR.BIT.B6 = 0

#define H_ON PORTA.PODR.BIT.B0 = 1
#define H_OFF PORTA.PODR.BIT.B0 = 0

#define TXD_ON PORTA.PODR.BIT.B4 = 0
#define TXD_OFF PORTA.PODR.BIT.B4 = 1
```

```
#define SW1 PORT5.PIDR.BIT.B4
#define SW2 PORTC.PIDR.BIT.B6
#define SW3 PORTC.PIDR.BIT.B5
#define SW4 PORTC.PIDR.BIT.B4

②void lwait(long time)
{
        while(time != 0)
        {
                time--;
        }
}


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

③void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

   while (1U)                                   //この中を無限ループ
    {
④                   if(SW1 == 0)                //SW1 が押されると LED   D1   ON
                    {
⑤                           H_ON;
                    }
                    if(SW2 == 0)                //SW2 が押されると LED   D4   ON
                    {
                            M_ON;
                    }
                    if(SW3 == 0)
                    {
                            TXD_ON;             //SW3 が押されると LED   D3   ON
                    }
                    if(SW4 == 0)
                    {
                            LED_ON;             //SW 4 が押されると LED   D4   ON
                    }



⑥                   lwait(600000);              //ウエイト
                    H_OFF;                      //LED   D1   OFF
                    LED_OFF;            //LED   D4   OFF
```

```
                M_OFF;                  //LED   D2   OFF
                TXD_OFF;                //LED   D3   OFF
                lwait(600000);          //ウエイト

    }
```

【　解説　】

①#define LED_ON PORTE.PODR.BIT.B0 = 1
ＬＥＤ＿ＯＮをＰＥ０＝１と定義しています。　ＬＥＤ　Ｄ４を点灯するためにＰＥ０ポートに１を出します。

②
```
void lwait(long time)
{
        while(time != 0)
        {
                time--;
        }
}
```

ウエイト関数です。ｔｉｍｅが０になるまでループします。

③
```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

   while (1U)                           //この中を無限ループ
    {
```

電源がＯＮすると末尾にあるシステム初期設定 R_MAIN_UserInit();を行い、ｗｈｉｌｅ（１Ｕ）の中を無限ループします。コード生成で作成したポートの初期設定等はここに来る前のｒ＿ｃｇ＿ｒｅｓｅｔｐｒｇ．ｃ、ｒ＿ｃｇ＿ｈａｒｄｗａｒｅ_ｓｅｔｕｐ．ｃ　等で実行されています。

④
```
                if(SW1 == 0)            //SW1 が押されると LED   D1   ON
                {
```

SW1 PORT5.PIDR.BIT.B4　ポート５４が０（スイッチ１が押されている）なら

⑤
```
                        H_ON;
                }
```

H_ON PORTA.PODR.BIT.B0 = 1　ポートＡ０に１を出します。トランジスタＱ１がＯＮし、ＬＤ１が点灯します。

⑥

```
lwait(600000);          //ウエイト
H_OFF;                  //LED　D1　OFF
LED_OFF;                //LED　D4　OFF
M_OFF;                  //LED　D2　OFF
TXD_OFF;                //LED　D3　OFF
lwait(600000);          //ウエイト
```

ウエイト時間を置いて、出力をオフ　＝　消灯しています。その後、ウエイト時間を置いて再びキーを読みに行きます。

【　演習問題　】

ｓａｍｐｌｅ１の動作を理解できた方は、演習問題に進んで下さい。課題は
ＳＷを押さないとＬＤ２、Ｄ３、Ｄ４が点滅する。ＳＷ１を押すと全ＬＥＤ点滅しない。ＳＷ２を押すとＤ４が消灯、ＳＷ３を押すとＬＤ２が消灯、ＳＷ４押すとＤ３が消灯するプロフラム。

回答例　ｓａｍｐｌｅ１＿ｂ

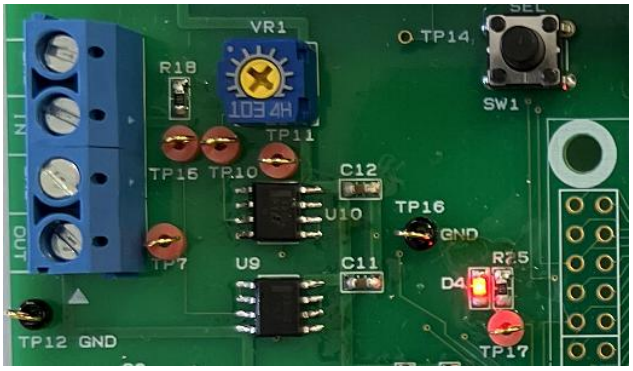
回答例は例であり、課題に合致していれば、書き方がこの通りである必要はありません。

２－２　ｓａｍｐｌｅ２　Ａ／Ｄ変換データを有機ＥＬディスプレイに表示する　Ｉ２Ｃインターフェイス

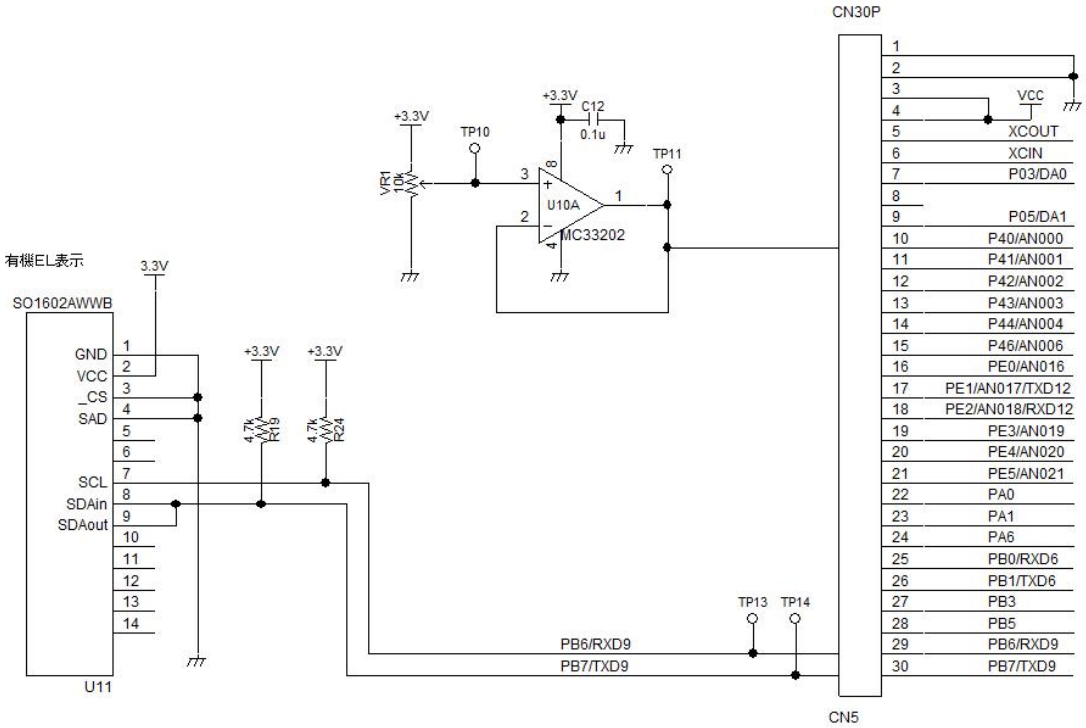
【　概要　】

Ｐ４０／ＡＮ０００入力に入力される電圧をＡ／Ｄ変換器でアナログ→デジタル変換し、ＶＲ１を回すことにより０－３．３Ｖのスパンで Ｉ２Ｃインターフェイスで有機ＥＬディプレイに表示します。

電源はＥ２Ｌｉｔｅから供給されます。



【　ハードウエア　】



ＶＲ１とＡ／Ｄ変換器　ＡＮ０００の間にはオペアンプＭＣ３３２０２によるバッファ（インピーダンス変換器　電圧増幅しない）が１段入っています。この理由はＲＸ２３０マイコンのＡ／Ｄコンバータが０．８３μｓの高速変換を実現するために、信号源インピーダンスが０．５ＫΩ以下であること、という条件があるためです。

43.7　許容信号源インピーダンスについて

本 MCU のアナログ入力は、高速変換 0.83μs を実現するために、信号源インピーダンスが 0.5kΩ 以下の入力信号に対し、変換精度が保証される設計となっています。シングルスキャンモードで 1 端子のみ変換を行

ハードウエアマニュアル　１２ビットＡ／Ｄコンバータ　許容信号源インピーダンス　の項参照　１７４６ページ

オペアンプは入力インピーダンスは大きく、出力インピーダンスは低くという特性を持っているので、このようなバッファでインピーダンス変換する場合にも使われます。

ところで、プログラムを動作させて見ると、ＶＲ１　ＴＰ１０を０Ｖにしても、ＴＰ１１ががゼロにならず、有機ＥＬ表示器もゼロが表示されないという現象があります。

測定してみるとオペアンプ入力のＴＰ１０は０．０００Ｖですが、オペアンプ出力のＴＰ１１は０．０４Ｖ前後でした。理想的なオペアンプでは入力０Ｖの場合、出力も０になるのですが、実際は入力オフセット電圧と入力バイアス電流、レイルツーレイル構造の影響で０になりません。

また、ＶＲ１　ＴＰ１０を最大　約３．３ＶにしてもＴＰ１１が同じにならない現象もあります。

ＭＣ３３２０２　オペアンプは入出力レイルツーレイルオペアンプなのでゲイン１の場合、３．３Ｖ入力に対して出力３．３Ｖ出て欲しい所ですが、実際は出ません。アウトプットボルテージスイング（出力電圧スイング）によれば電源電圧５．０Ｖに対して４．９５Ｖ、負荷電流が増えると４．８５Ｖ

DC ELECTRICAL CHARACTERISTICS (cont.) (V_{CC} = + 5.0 V, V_{EE} = Ground, T_A = 25°C, unless otherwise noted.)

Characteristic	Figure	Symbol	Min	Typ	Max	Unit
Input Offset Current (V_{CM} = 0 V to 0.5 V, V_{CM} = 1.0 V to 5.0 V) T_A = + 25°C T_A = − 40° to +105°C T_A = − 55° to +125°C	−	$\lvert I_{IO} \rvert$	 − − −	 5.0 10 −	 50 100 200	nA
Common Mode Input Voltage Range	−	V_{ICR}	V_{EE}	−	V_{CC}	V
Large Signal Voltage Gain (V_{CC} = + 5.0 V, V_{EE} = − 5.0 V) R_L = 10 kΩ R_L = 600 Ω	7	A_{VOL}	 50 25	 300 250	 − −	kV/V
Output Voltage Swing (V_{ID} = ± 0.2 V) R_L = 10 kΩ R_L = 10 kΩ R_L = 600 Ω R_L = 600 Ω	8, 9, 10	 V_{OH} V_{OL} V_{OH} V_{OL}	 4.85 − 4.75 −	 4.95 0.05 4.85 0.15	 − 0.15 − 0.25	V

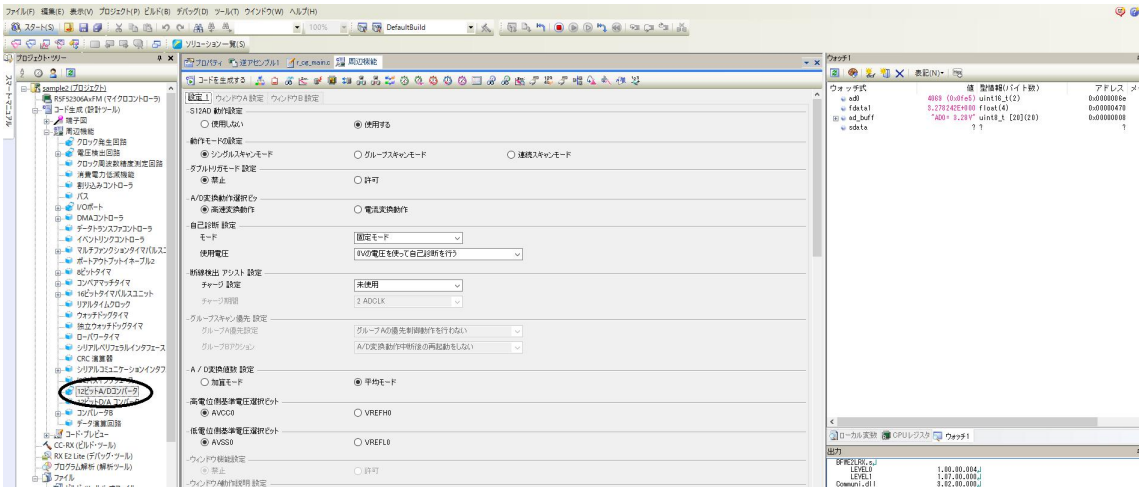
など、電源電圧５．０Ｖに対して出力は電源電圧まで出力されません。
また、出力の最小値も０．０５Ｖ～０．１５Ｖと０Ｖにはならないようです。

Ａ／Ｄコンバータの入力特性上、入出力レイルツーレイルオペアンプを単電源で使用していますが、上下限近辺のデータには注意が必要になります。

表示に使われている有機ＥＬ　ＳＯ１６０２ＡＷＷＢ　ですが、表示素子自身が光りますので、液晶に比べ視認性が良い、応答速度が高速などの特徴があります。インターフェイスはＩ２Ｃで、信号線は２本しか使いません。本機はマイコンから有機ＥＬにデータを出力するだけの結線となっています。

【　コード生成　】

コード生成は１２ビットＡ／Ｄコンバータの項を以下のように設定されています。本サンプルではソフトウエアでＡ／Ｄ変換を開始しています。



【　プログラム　】

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

①          init_SO1602();                          //有機 EL イ二シャル

                    while(1)
                    {

                    ②  S12AD.ADCSR.BIT.ADST = 1;                   //AD スタート
                        while(S12AD.ADCSR.BIT.ADST)                 //変換時間待ち
                                    ;
                    ③          ad0 = S12AD.ADDR0;                   //データ読み込み
                               fdata1 = ad0/(4095/3.3);          //AD データ→電圧換算　3.3V で 4095
                               sprintf(ad_buff,"AD0= %.2fV",fdata1);    //表示バッファにデータ設定
                               LED_ON;                              //マーカー
                    ④          lcd_puts(0,ad_buff);                 // 1 行目　数値表示
                               lcd_bar_disp(1,ad0);                 // 2 行目　バー表示
                               LED_OFF;                             //マーカー
                    }


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

【　解説　】

```
①          init_SO1602();                          //有機 EL イニシャル
```

有機ＥＬのイニシャルを行っています。詳細は#include <lcd_so1602_rx230.c>　　//有機 EL ライブラリをご参照ください。

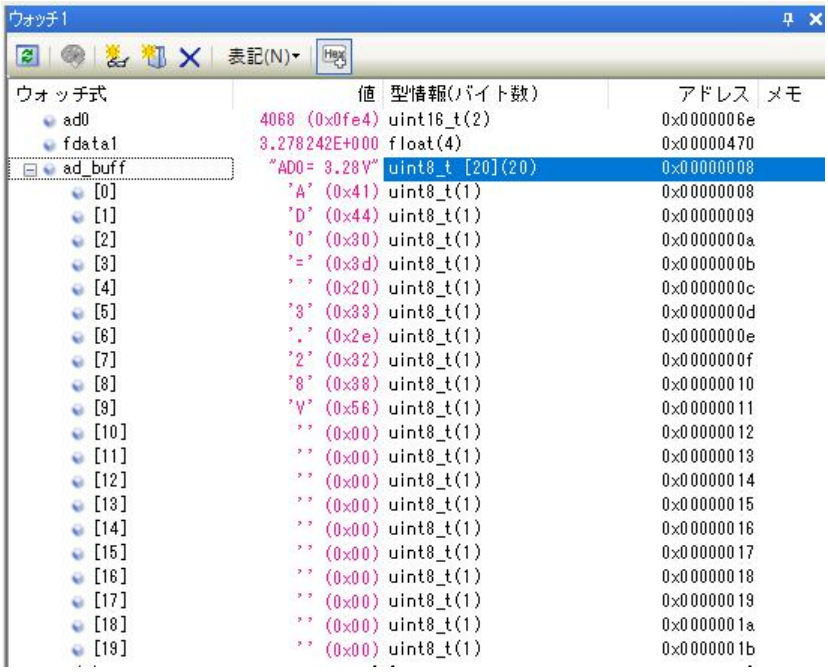
```
                    while(1)
                    {

②          S12AD.ADCSR.BIT.ADST = 1;               //AD スタート
           while(S12AD.ADCSR.BIT.ADST)             //変換時間待ち
                    ;
```

ＡＤＳＴビットを１にしてＡ／Ｄ変換開始。１になると変換終了です。

```
③          ad0 = S12AD.ADDR0;                      //データ読み込み
           fdata1 = ad0/(4095/3.3);                //AD データ→電圧換算　3.3V で 4095
           sprintf(ad_buff,"AD0= %.2fV",fdata1);   //表示バッファにデータ設定
```

データを読み込み、3.3V に換算し、表示のためのアスキーコードに変換してます。内容は右手のウオッチ窓でリアルタイムに確認できます。表示されていない場合、ウオッチ式をインポートして表示させてください。**ｂ－５：変数変化を実行中に見る　ウオッチ窓の使い方**　参照



```
        LED_ON;                         //マーカー
④       lcd_puts(0,ad_buff);            //１行目　数値表示
        lcd_bar_disp(1,ad0);            //２行目　バー表示
        LED_OFF;                        //マーカー
```

ｌｃｄ＿ｐｕｔｓ（）は有機ＥＬに数値を表示する関数です。
ｌｃｄ＿ｂａｒ（）は数値をバーグラフで表示する関数です。
いずれも詳細はライブラリ　ご参照下さい。

【　演習　】

Ａ／Ｄデータを１回ではなく、何回か平均して表示するプログラムを作成してみてください。平均回数を変えると数値のバラツキ、安定性が変化することを確認できると思います。

参考プログラム　ｓａｍｐｌｅ２＿ｂ

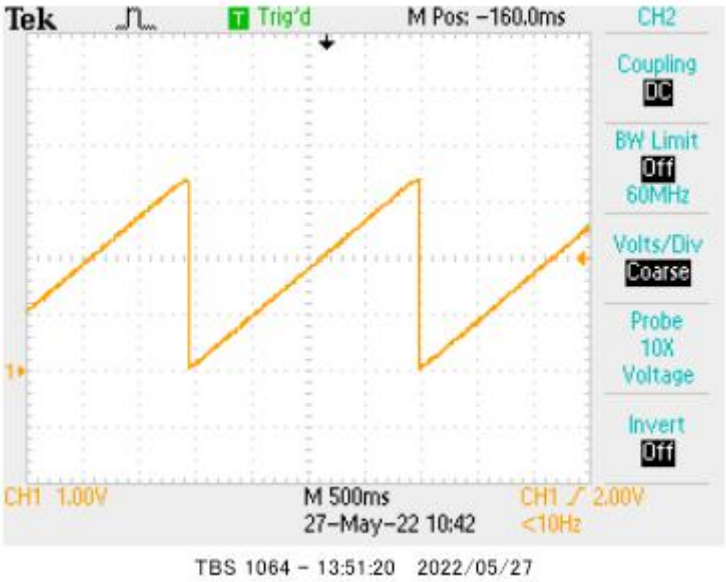
２－３　ｓａｍｐｌｅ３　Ｄ／Ａコンバータで三角波を作る

【　動作概要　】

ＤＡコンバータ、定周期割り込みの使い方を学習します。

１ｍｓｅｃ毎にＤＡ０の電圧レベルを変えて三角波形を作っています。

Ｐ０３／ＤＡ０　ＣＮ８の１番、ＴＰ７で波形観測できます。



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