2022.01.11

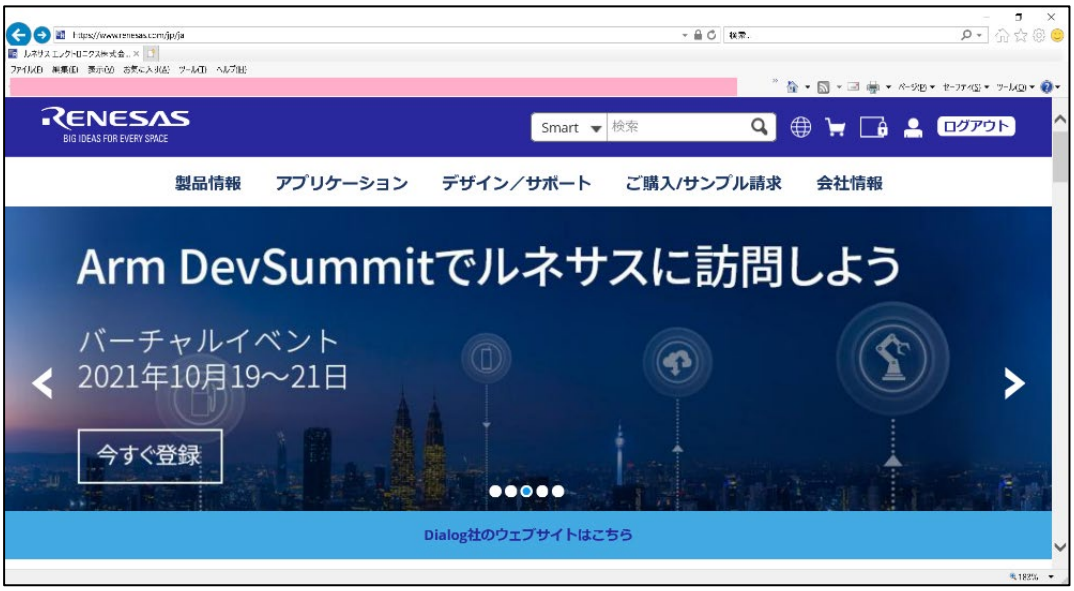
「E2 エミュレータ Lite 使用法」

Ｅ２エミュレータLiteはエミュレータ側からの電源供給に５Vの設定がございません。接続した際にはマイコンボードも３．３Vでご使用いただく事となります。
Ａ／Ｄコンバータのリファレンスなど、ＶＤＤ電圧に依存するプログラムの場合はご注意ください。
（マイコンボードに直接電源を供給する場合は５Vでのご使用が可能です）

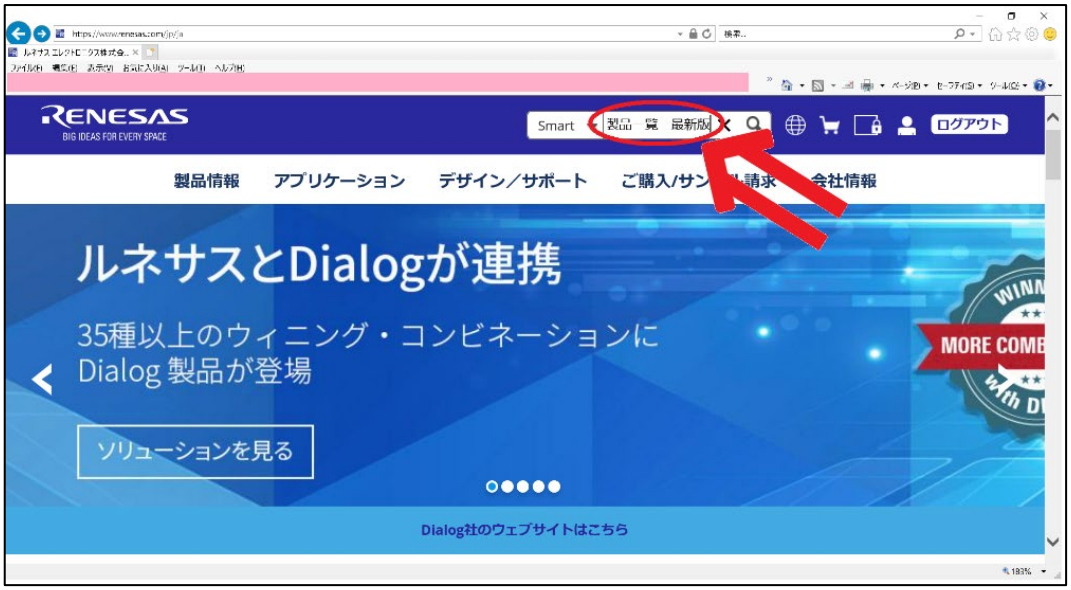
・統合開発環境CS＋のインストール説明

E2エミュレータLiteは統合開発環境ＣＳ＋でお使いいただけます。HEWでは使用できません。
RENESAS 社様ホームページより下記リンクに飛び、ファイルをダウンロードします。
（＊ダウンロードにはユーザー登録が必須となります。）

１）ＲＥＮＥＳＡＳ社ホームページにアクセスします。



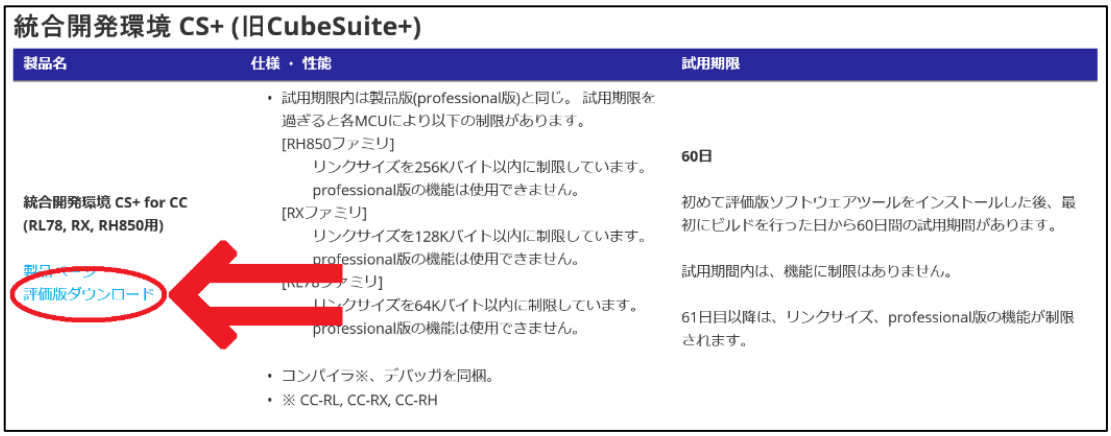
２）検索窓に　**評価版ソフトウェアツール　ダウンロード製品一覧　最新版**　と入力して検索。



3）検索結果が出たらクリック。

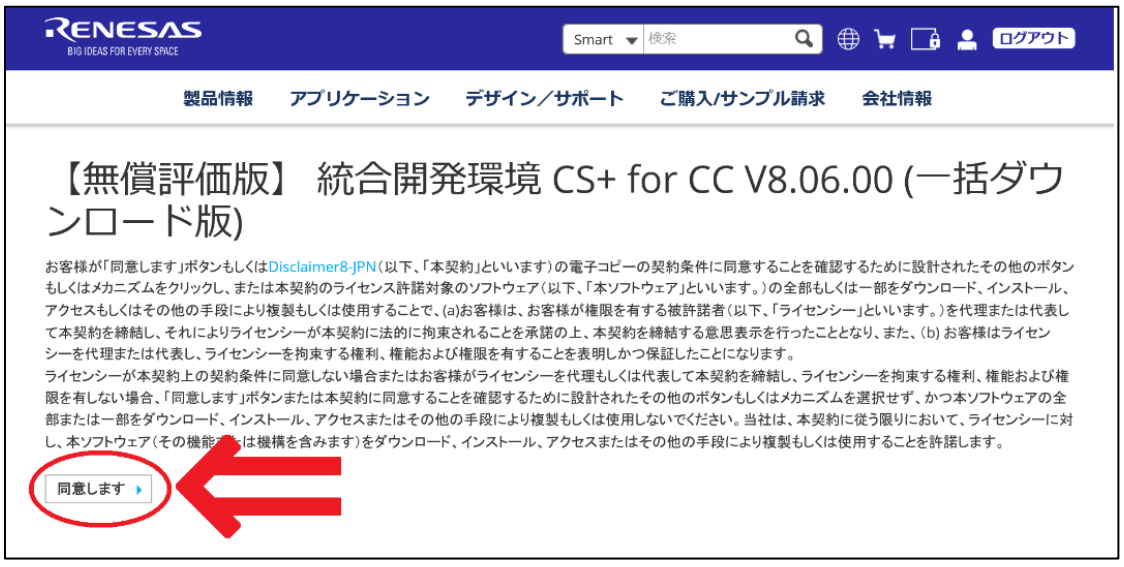
4）このページに飛びます。

5）下にスクロールしてこの画面から「評価版ダウンロード」をクリック。



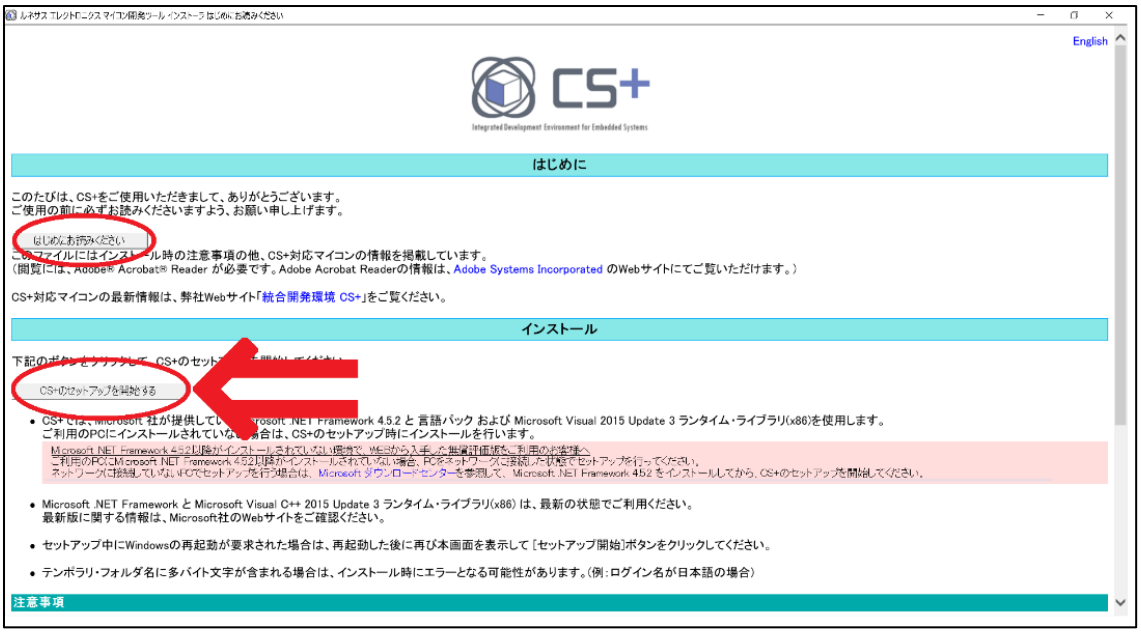
6）【無償評価版】統合開発環境 CS+ for CC V8.06.00(一括ダウンロード版)をクリック。
（「V8.06.00」の部分は最新の物に変更されている場合があります）

7）契約ページで、文章に同意する場合「同意します」をクリック。



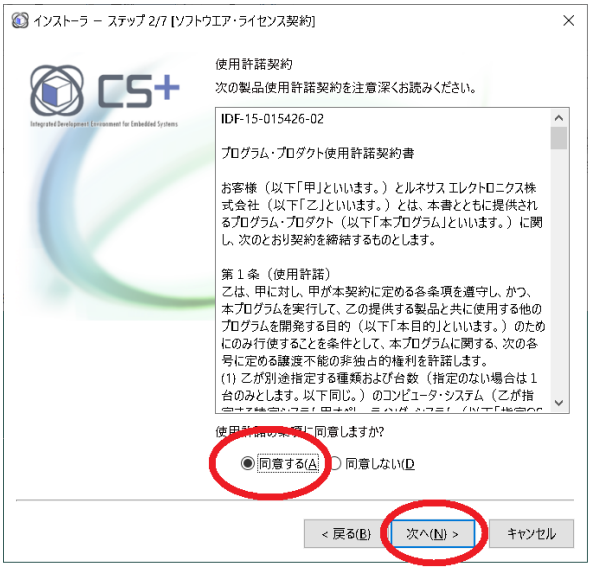
8）同意すると「CSPlus_CC_Package_V8000-doc-j.zip」のダウンロードが始まります。
ダウンロードした zip ファイルをクリックすると下記のファイルが解凍されます。

9）CSplus_CC_Package_V80600.eEXE をクリックすると、インストールが始まります。
「はじめにお読みください」をクリックして説明を読んだ後、「CS+のセットアップを開始する」をクリックしてセットアップを開始します。



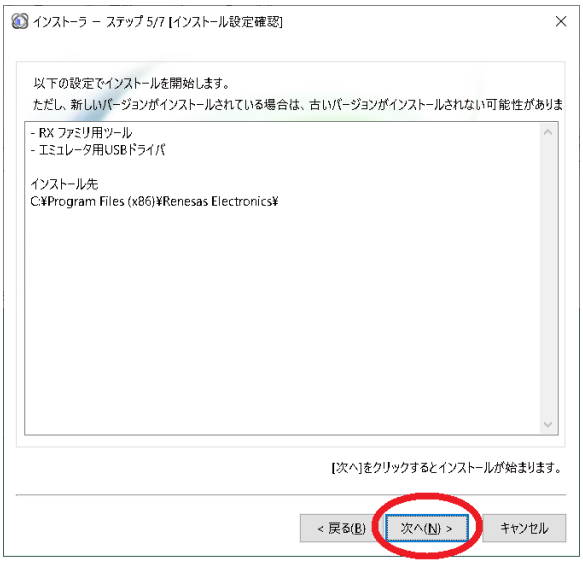
１０）セットアップを開始しますので「次へ」をクリック。

１１）使用許諾契約を読んだ後、「同意する」にチェックを入れて「次へ」をクリック。

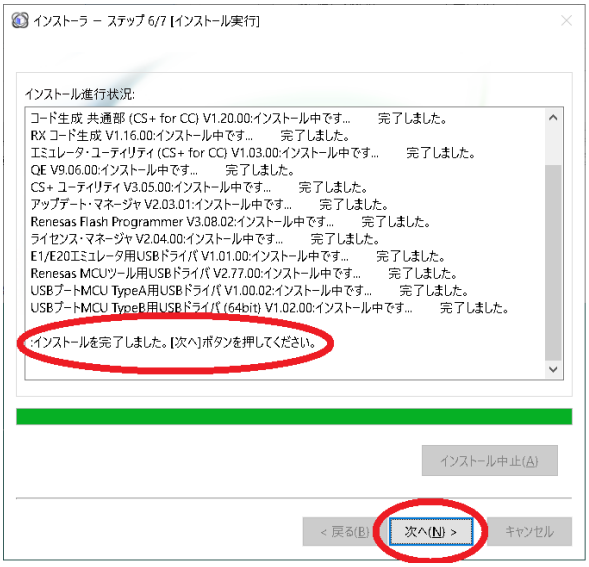


１２）開発ツール選択で「RX ファミリ用ツール」「エミュレータ用 USB ドライバ」にチェックを入れて「次へ」をクリック。

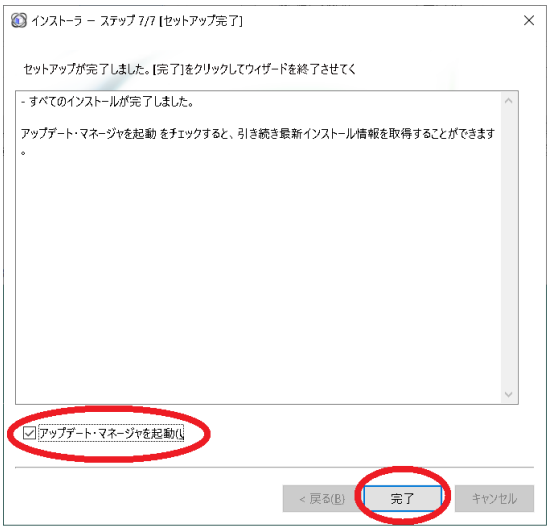
１３）インストール設定を確認して「次へ」をクリック。



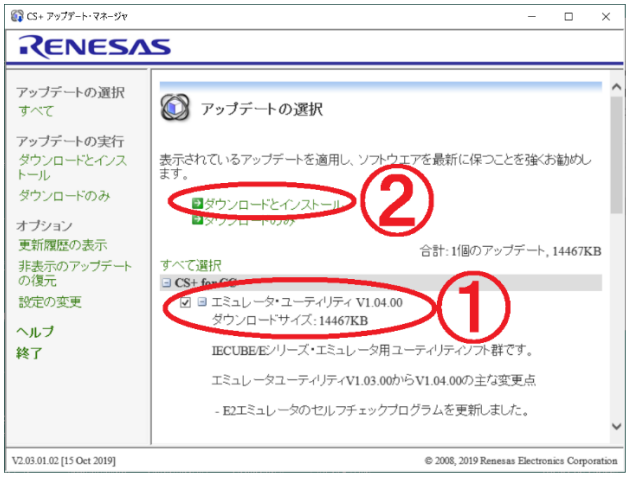
１４）インストールが完了したら、確認して「次へ」をクリック。



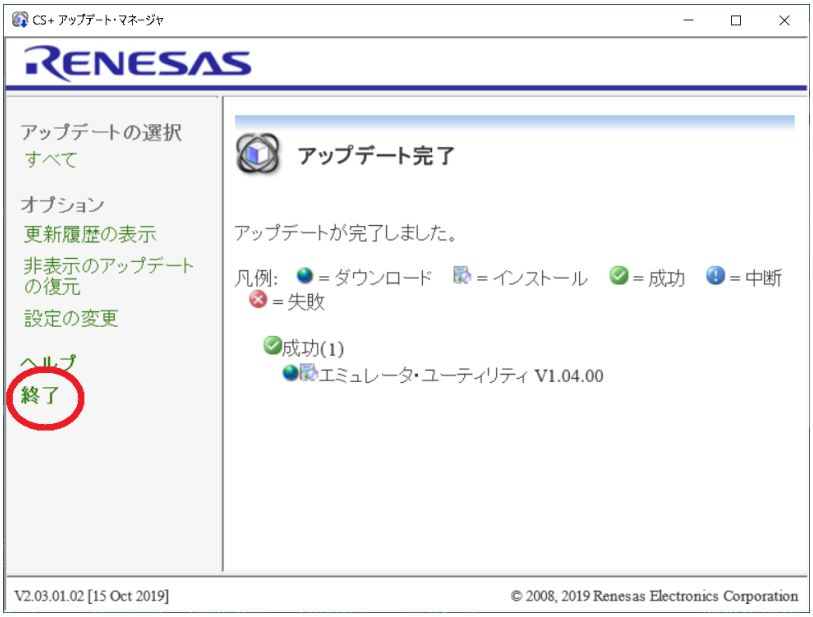
１５）セットアップが完了したのでアップデートを確認します。「アップデート・マネージャを起動」にチェックを入れて「完了」をクリック



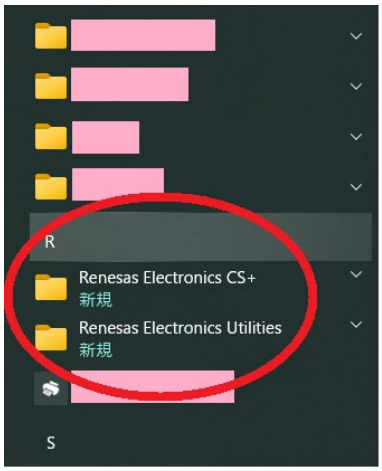
１６）アップデート・マネージャが起動するので①「エミュレータ・ユーティリティ V1.04.00」にチェックを入れて②「ダウンロードとインストール」をクリック。
（「V1.04.00」の数字はダウンロード時の最新の物になっています。）



１７）アップデート完了を確認して「終了」をクリック。



１８）スタートメニューからインストール完了を確認します。



以上でソフトウェアのインストールは終了です。

続いてコンパイルに移ります。

「E2 エミュレータ Lite コンパイル」

にお進みください。

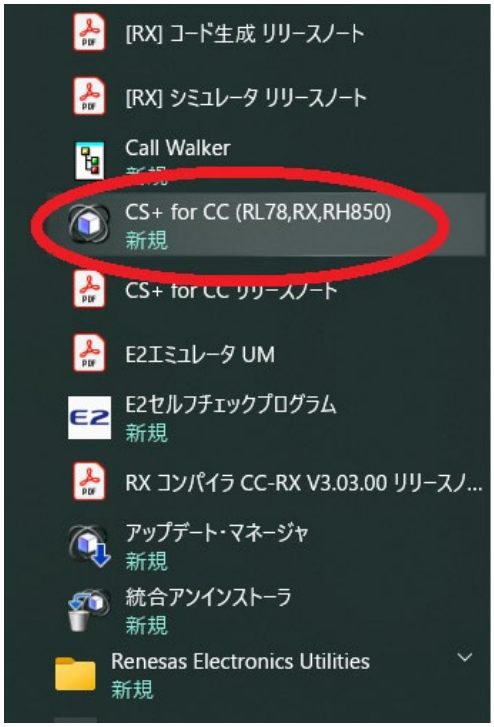
「E2 エミュレータ Lite コンパイル」

●C言語でのサンプルソフトのコンパイル例説明

本CDの「サンプル」フォルダ内の「TEST_LED.c」が、Cコンパイラ用サンプルソースファイルです。

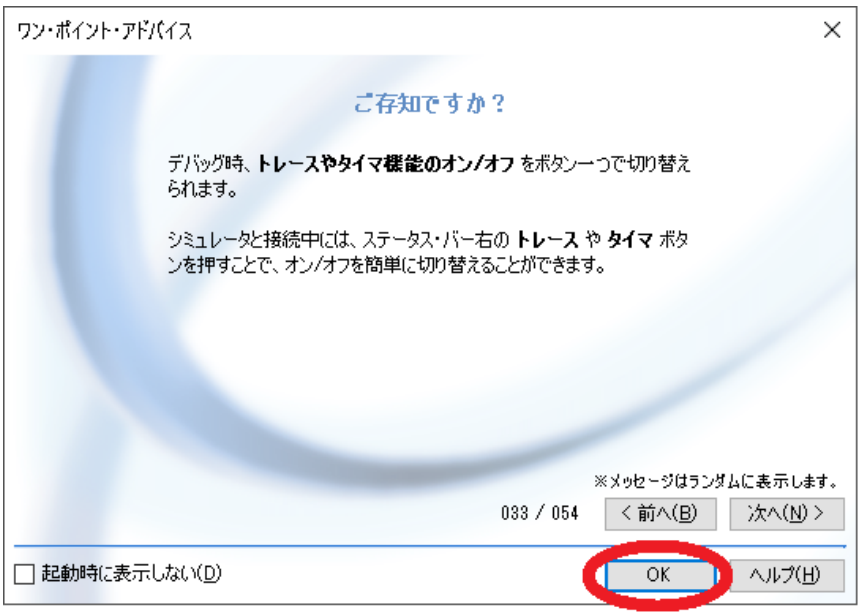
■TEST_LED.c のコンパイル例■

1）フォルダ「Renesas Electronics CS+」中の「CS+ for CC(RL78,RX,RH850)」をクリックして開始します。

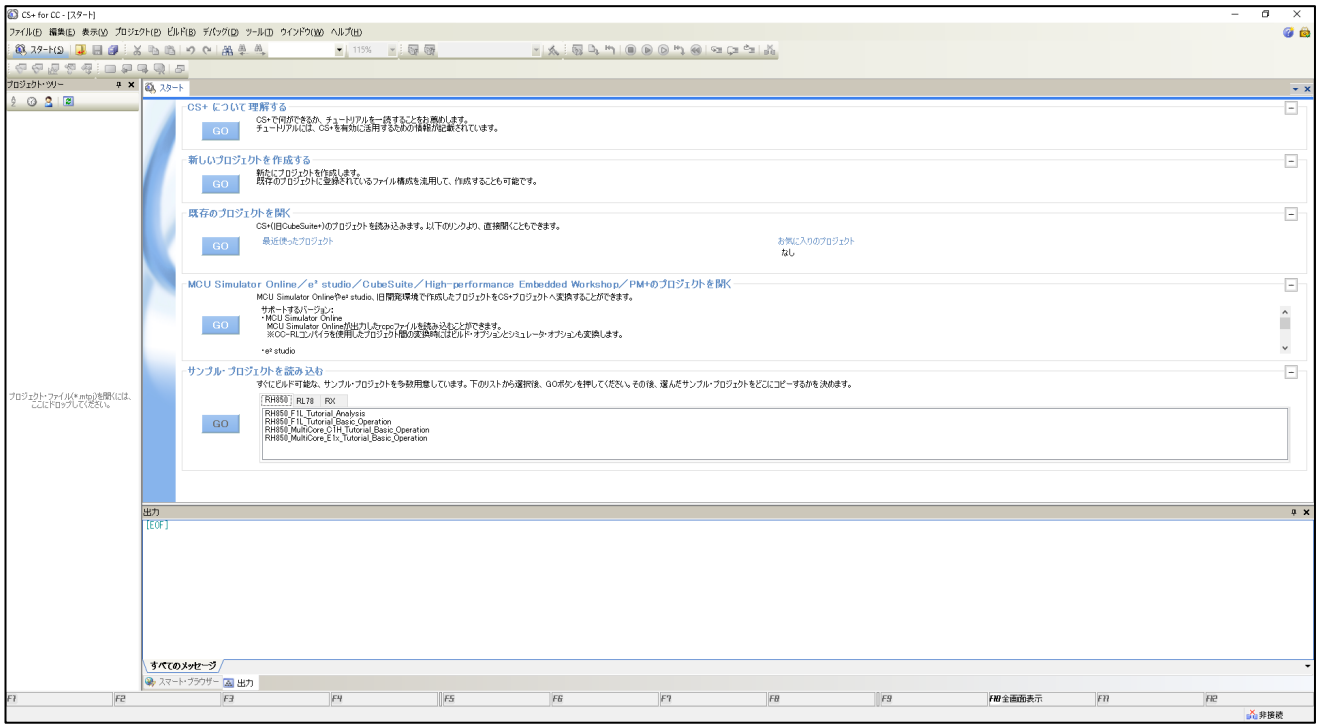


2）こちらが開始画面です。

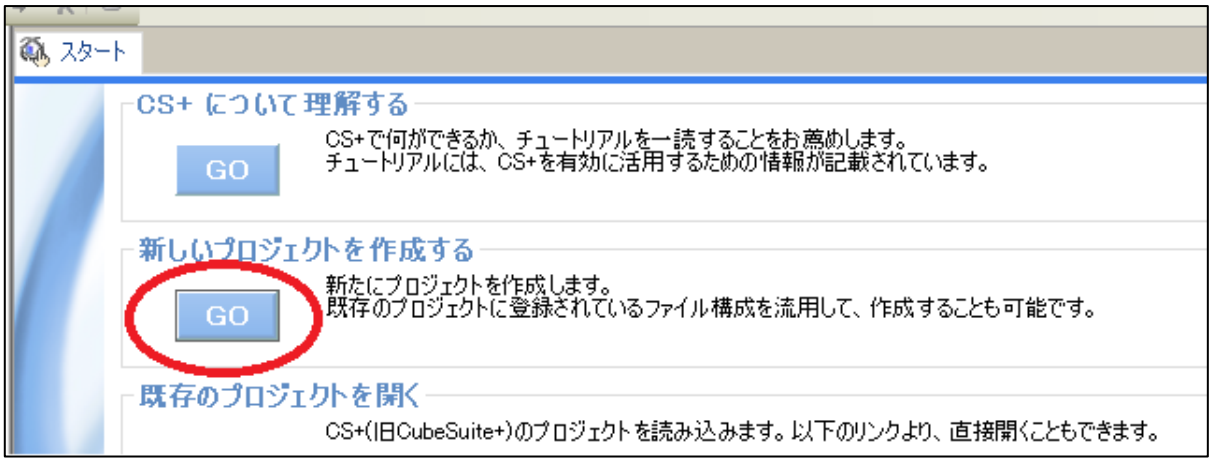
3）次に現れる「ワンポイント・アドバイス」にはこれから使用するに当たって役立つ情報が記載されています。今回はとりあえず「OK」をクリックして開発を開始します。



4）ＣＳ＋が立ち上がると下の画面が出ます。

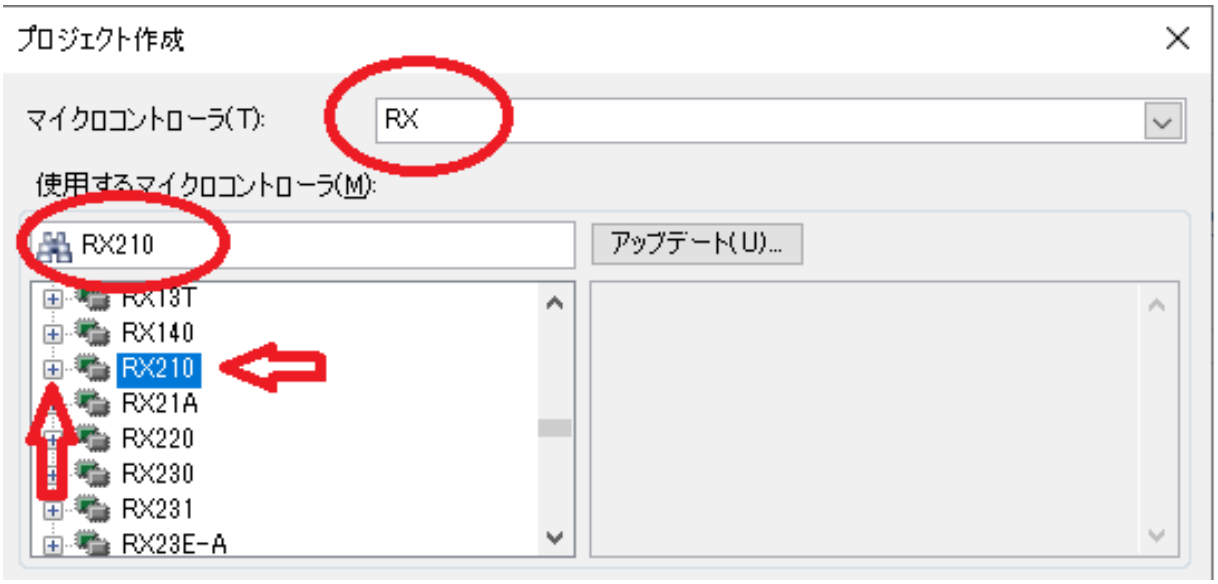


5）プロジェクトを作成します。「新しいプロジェクトを作成する」の「GO」をクリックします。



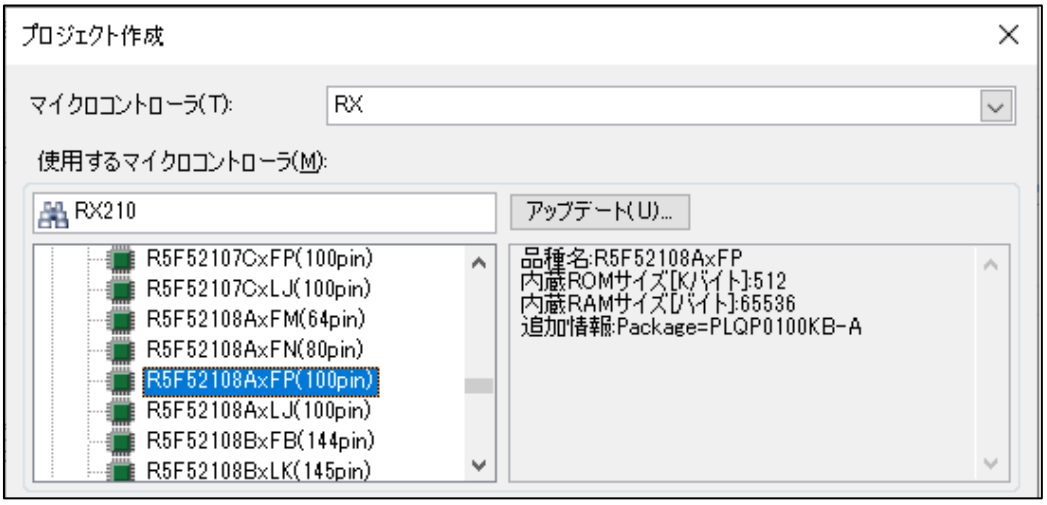
6）「プロジェクト作成」画面でマイクロコントローラ（ＣＰＵ）を選択します。

「マイクロコントローラ（T)」にＲＸ、「使用するマイクロコントローラ（M)」にＲＸ２１０を入力、表示されたＲＸ２１０の左の田マークをクリックします。

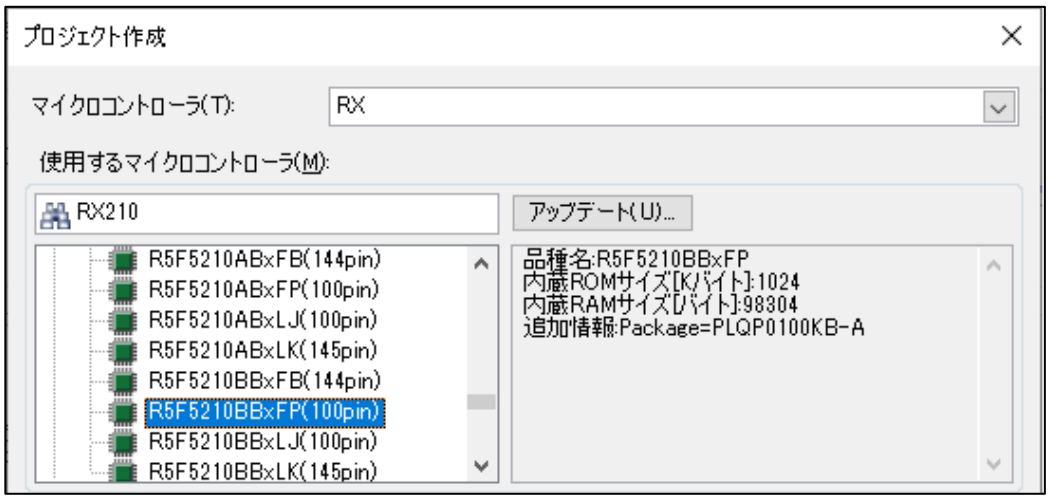


7）マイコンボードに使用しているマイクロコントローラを指定します。

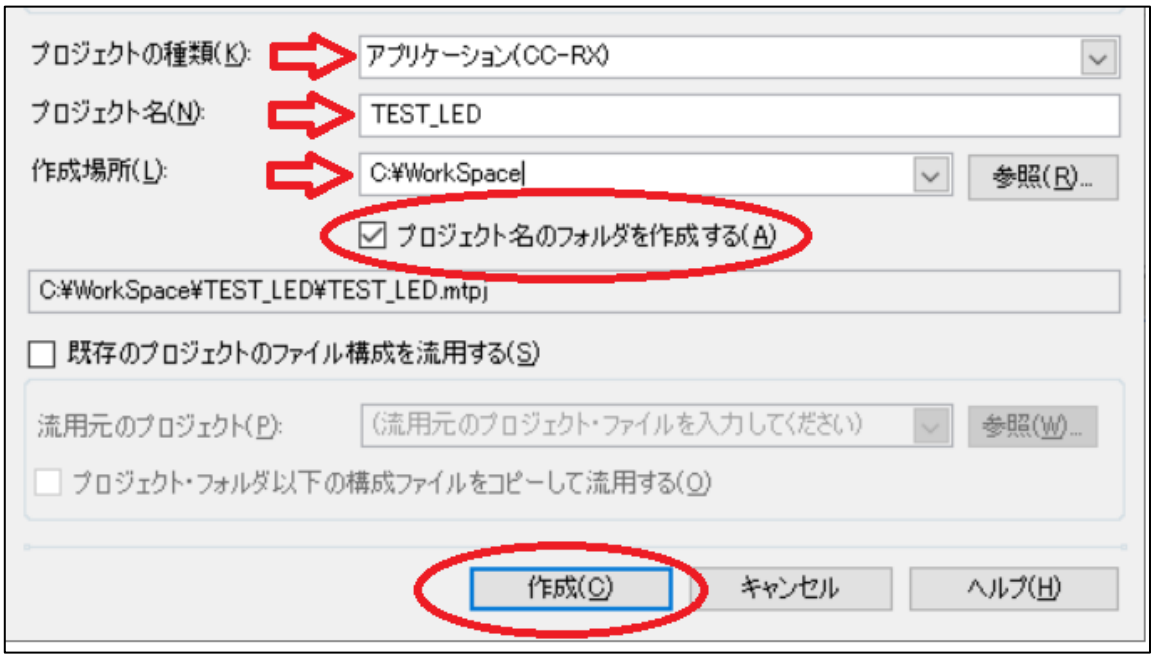
Ｒ５Ｆ５２１０８ＡＤＦＰボードの場合



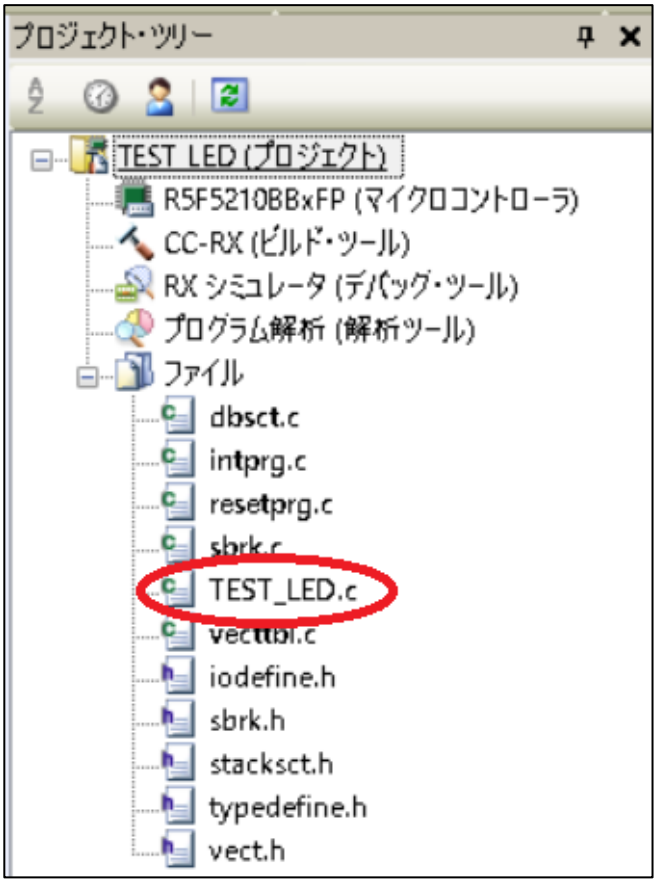
R５F５２１０BBDFPボードの場合



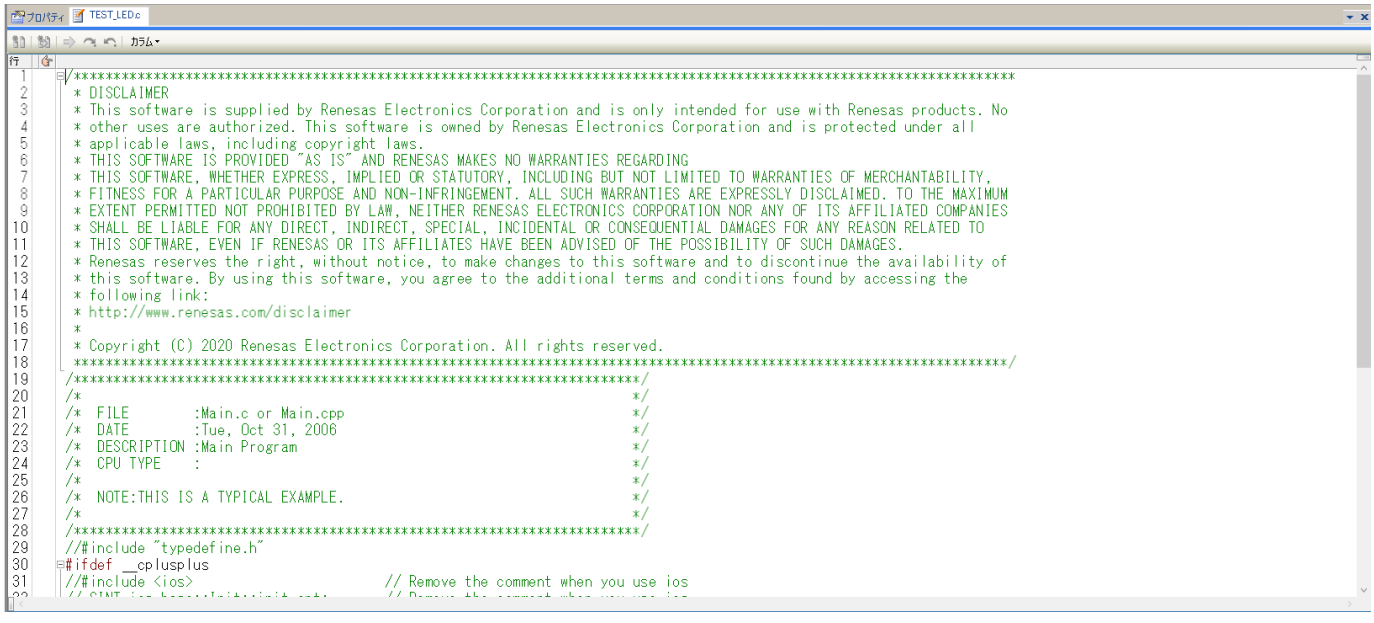
８）「プロジェクト作成」画面の下部でプロジェクトの種類、プロジェクト名、作成場所を指定し、「作成」をクリックします。ここではプロジェクトの種類を「ｱﾌﾟﾘｹｰｼｮﾝ(CC-RX)」プロジェクト名は「TEST_LED」作成場所を「C:¥WorkSpace」として「プロジェクト名のフォルダを作成する」にチェックをします。



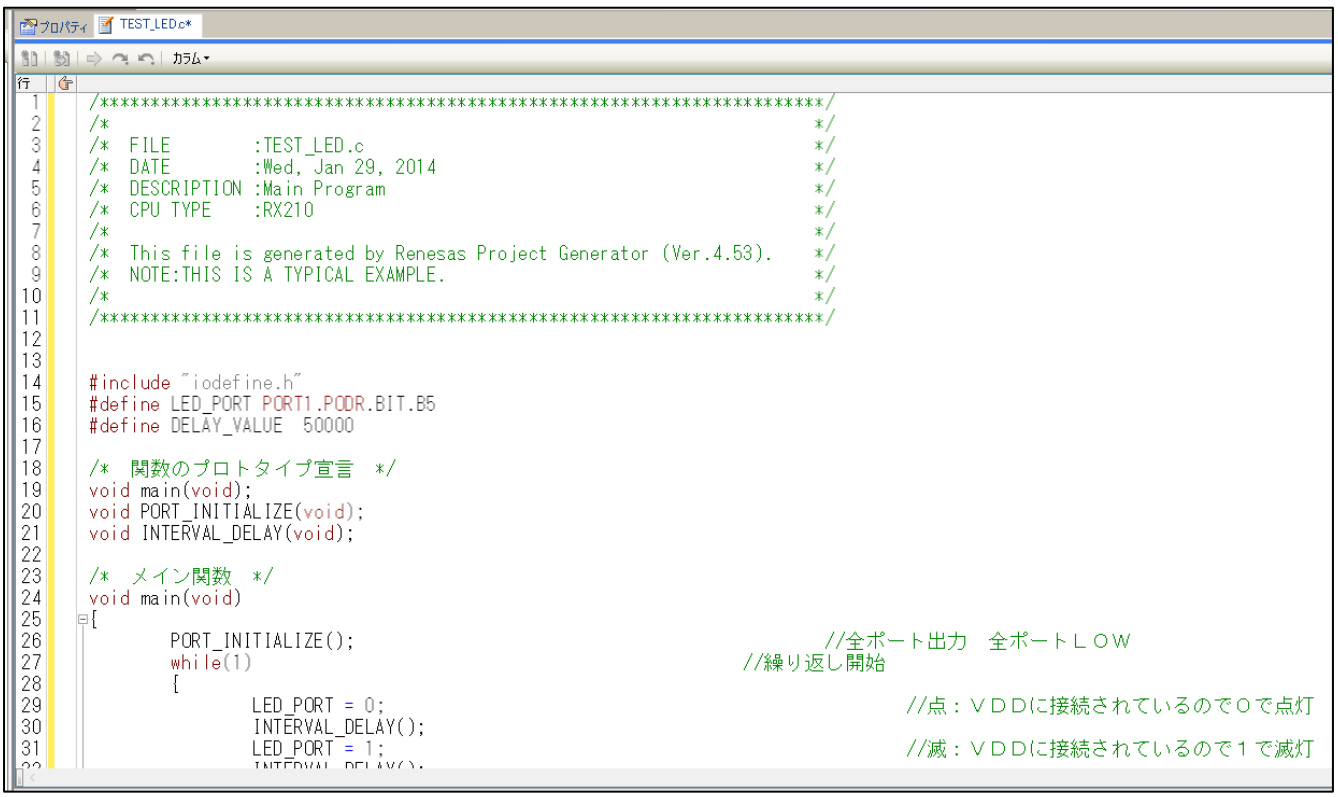
９）左上のプロジェクトツリーを確認すると、「ファイル」
の中に「TEST_LED.c」が生成されています。



１０）TEST_LED.c をダブルクリックすると、下図の様なデフォルトの雛形が現れますが、今回はキット付属の CD-ROM 内のサンプルを使用しますので全て消去します。
（画面内にカーソルを移動させＣｔｒｌ－Ａで全てを選択してからＤＥＬキーで消去してください。

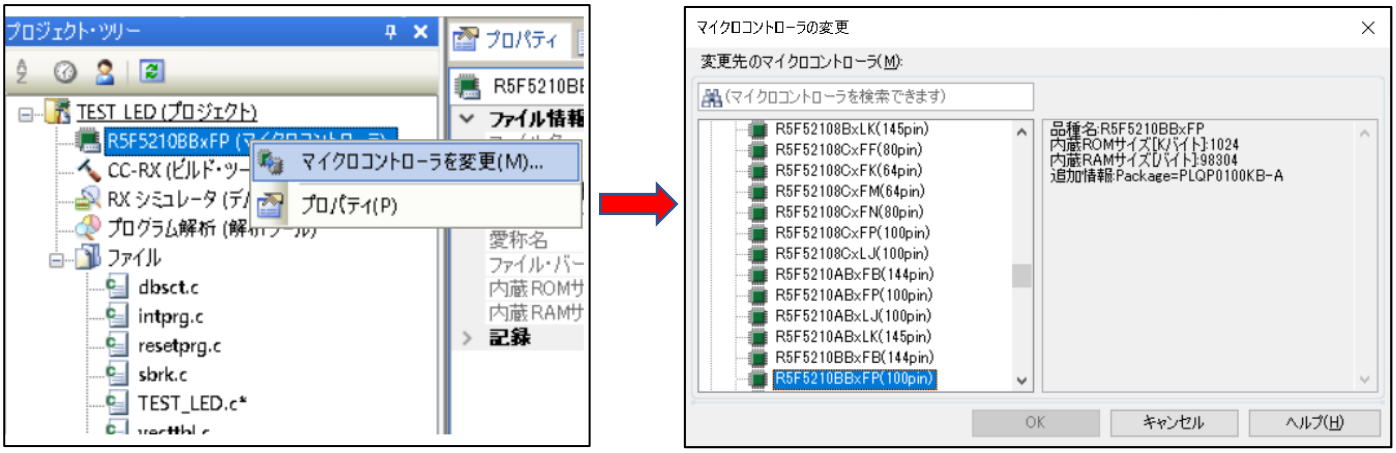


１１）「TEST_LED.c」を編集して、ソフトを作成しますが、キット添付ＣＤ（このＣＤです）の「サンプル」フォルダに「TEST_LED.c」がありますので、表示されているソースを削除し、「TEST_LED.c」を Windows 付属ソフトのメモ帳などで開き、内容をコピーします。
下図は、コピー後の状態です。（ファイルの先頭を表示させています）

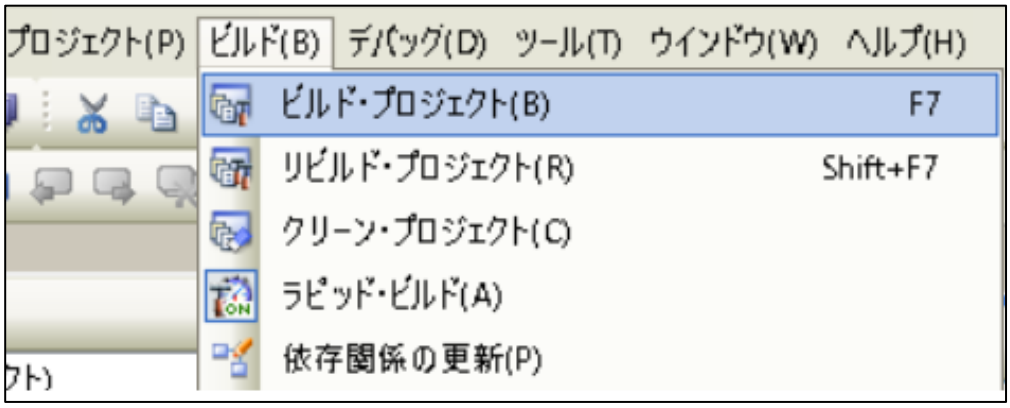


```
/***********************************************************************/
/*                                                                     */
/*  FILE        :TEST_LED.c                                            */
/*  DATE        :Wed, Jan 29, 2014                                     */
/*  DESCRIPTION :Main Program                                          */
/*  CPU TYPE    :RX210                                                 */
/*                                                                     */
/*  This file is generated by Renesas Project Generator (Ver.4.53).    */
/*  NOTE:THIS IS A TYPICAL EXAMPLE.                                    */
/*                                                                     */
/***********************************************************************/


#include "iodefine.h"
#define LED_PORT PORT1.PODR.BIT.B5
#define DELAY_VALUE  50000

/*  関数のプロトタイプ宣言  */
void main(void);
void PORT_INITIALIZE(void);
void INTERVAL_DELAY(void);

/*  メイン関数  */
void main(void)
{
        PORT_INITIALIZE();                                  //全ポート出力　全ポートＬＯＷ
        while(1)                                          //繰り返し開始
        {
                LED_PORT = 0;                                       //点：ＶＤＤに接続されているので０で点灯
                INTERVAL_DELAY();
                LED_PORT = 1;                                       //滅：ＶＤＤに接続されているので１で滅灯
```

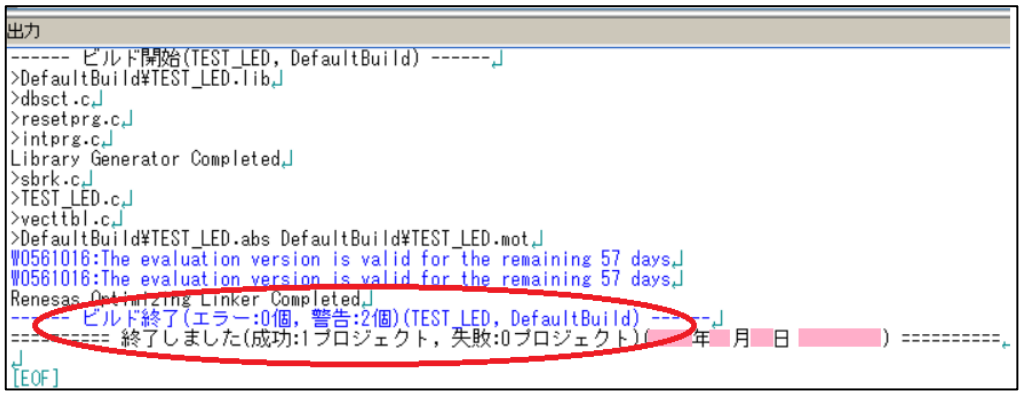
１２）マイコンの指定を間違えていた場合などは下記で修正します。７）をご参照ください。



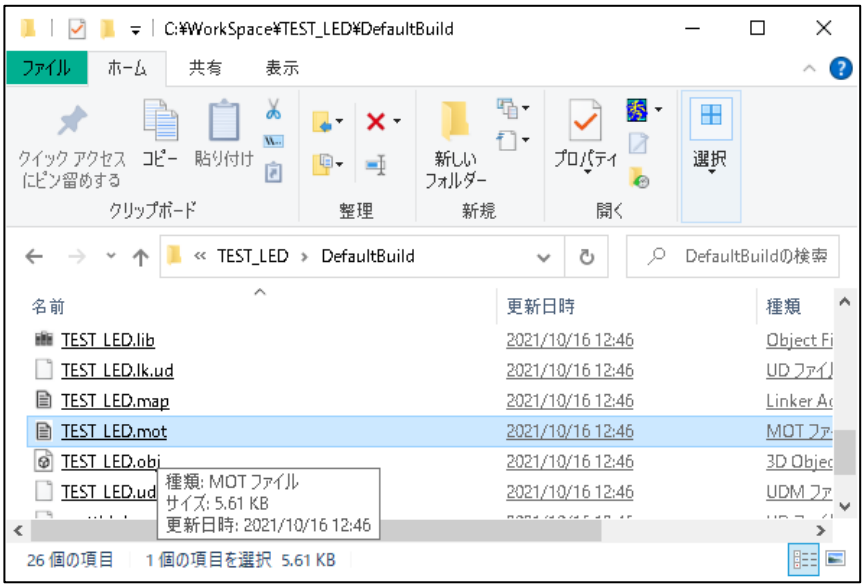
１３）プロジェクトをビルドします。（ソースコード「TEST_LED.c」を実行可能なファイル形式にします）
ツールバー「ビルド（B）」⇒「ビルド・プロジェクト（B）」をクリックします。



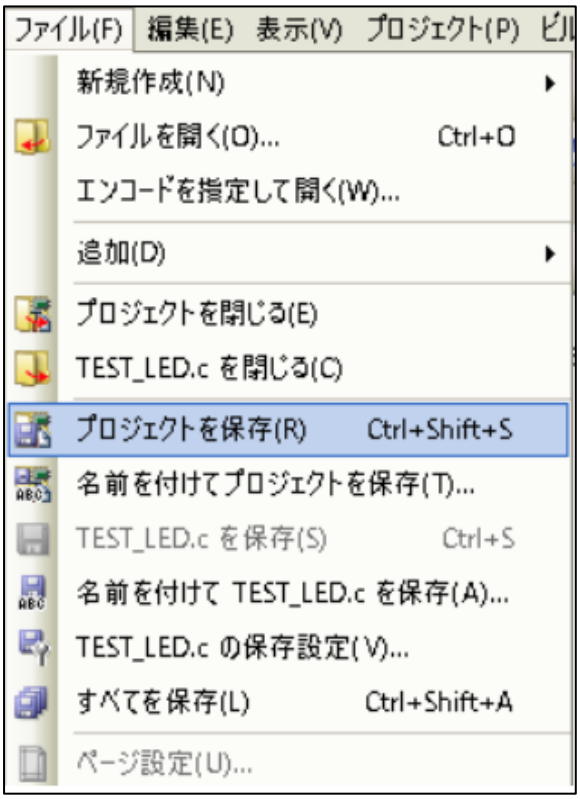
１４）正常に終了すると画面下部の「出力」窓にメッセージが表示されます。
（かかる時間はパソコンの性能により異なります）



１５）実行ファイルが生成されている事を確認します。８）で指定したディレクトリ（フォルダ）に
「TEST_LED.mot」ができている事を確認してください。



１６）一旦ここでプロジェクトを保存しておきましょう。
　　ツールバー「ファイル（Ｆ）」⇒「プロジェクトを保存（Ｒ）」で保存します。

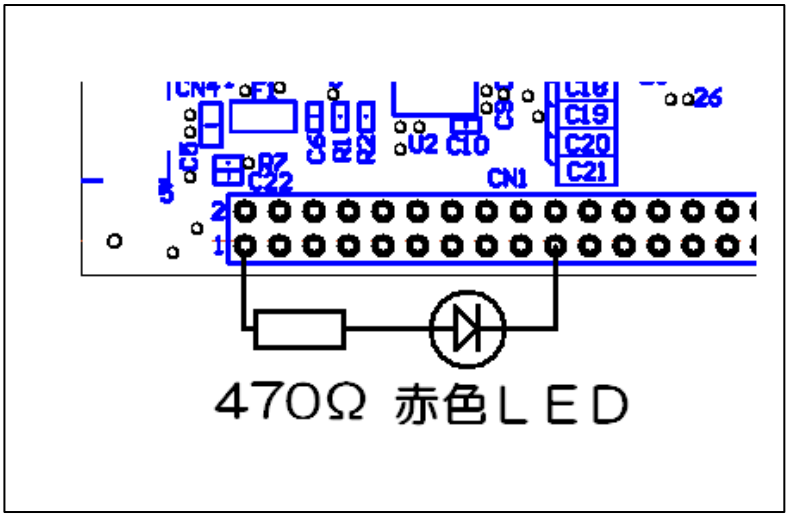


次はE2エミュレータLiteを使ったデバッグに移ります。

「E2 エミュレータ Lite デバッグ」にお進みください。

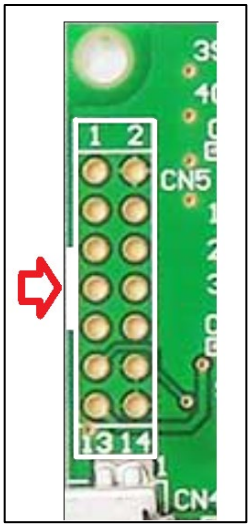
「E2 エミュレータ Lite デバッグ」

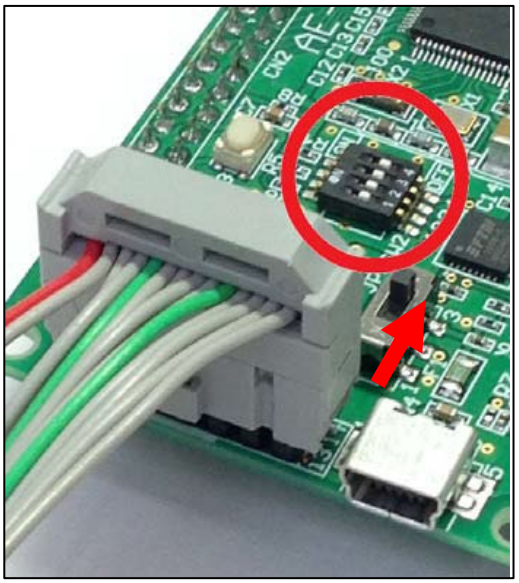
■デバッグ作業確認用テストＬＥＤ回路の接続■（４７０Ω抵抗と赤色ＬＥＤは別途ご用意ください）
ＣＮ１の１番ピン（ＶＤＤ）と１９番ピン（ＰＯＲＴ１．ＢＩＴ５）に４７０Ω抵抗と赤色ＬＥＤを直列に接続します。　１番ピンがアノード側、１９番ピンがカソード側です。



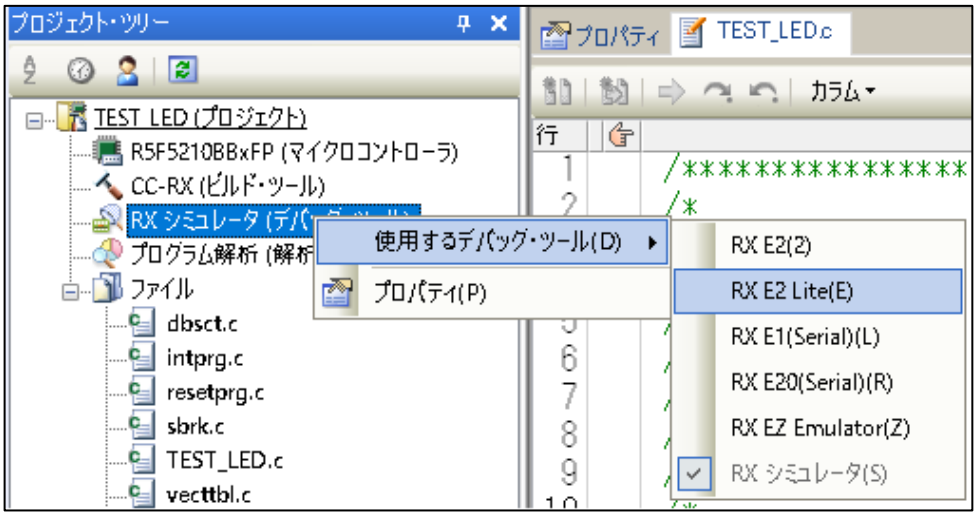
■Ｅ２エミュレータＬite（以下E2Liteと表記）との接続■
以下の説明では電源をE2Liteから供給するものとして説明いたします。
基板上のＳＷ１（スライドスイッチ）は必ず右下画像内矢印の方向（基板内側向き）にスライドさせておいてください。
E2LiteはＵＳＢケーブルでパソコンと接続しておきます。

１）E2Lite をＵＳＢ接続し、マイコンボードとフラットケーブルで接続します。
基板の凸印にコネクタを合わせてください。
ＳＷ２は１～４全てＯＦＦにします。

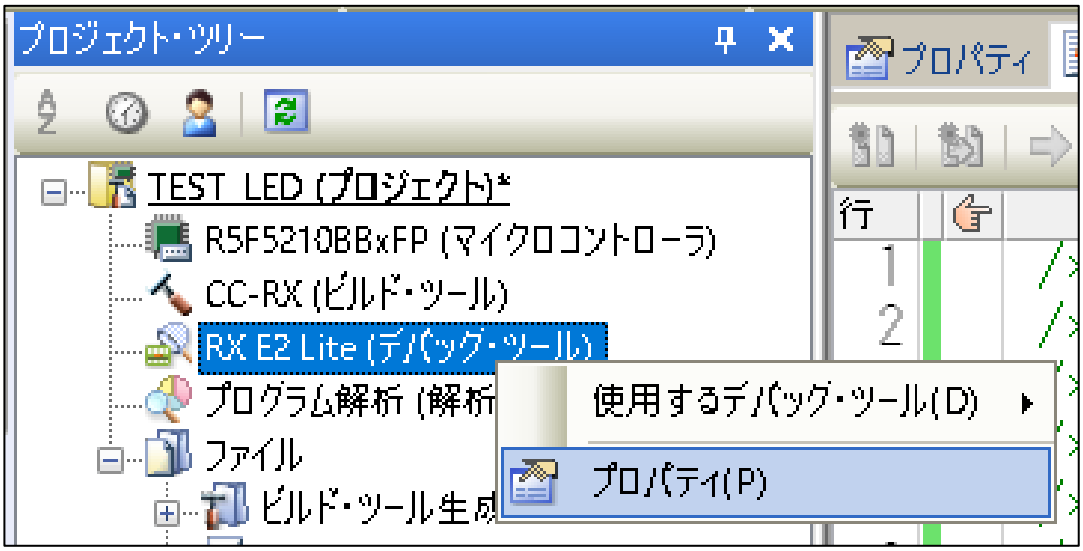




2）ＣＳ＋は「コンパイル編」からの続きで、既に立ち上がっている状態からの説明をいたします。
デバッグ・ツールに E2Lite を指定します。
プロジェクト・ツリーから「ＲＸシミュレータ（デバッグ・ツール）を右クリック、
「使用するデバッグ・ツール(D)」から「RX E2Lite(E)」を指定します。



3）E2Lite のプロパティを設定します。
プロジェクト・ツリーの「RX E2Lite(ﾃﾞﾊﾞｯｸﾞ･ﾂｰﾙ)」を右クリックして表示された
「ﾌﾟﾛﾊﾟﾃｨ(P)」をクリックします。



4）プロパティ画面が表示されます。

左下の「接続用設定」タブの画面です。

・「クロック」内「メイン・クロック・ソース」はＥＸＴＡＬ、

・「メイン・クロック周波数[ＭＨｚ]」＝20（入力すると 20.0000 と表示されます）

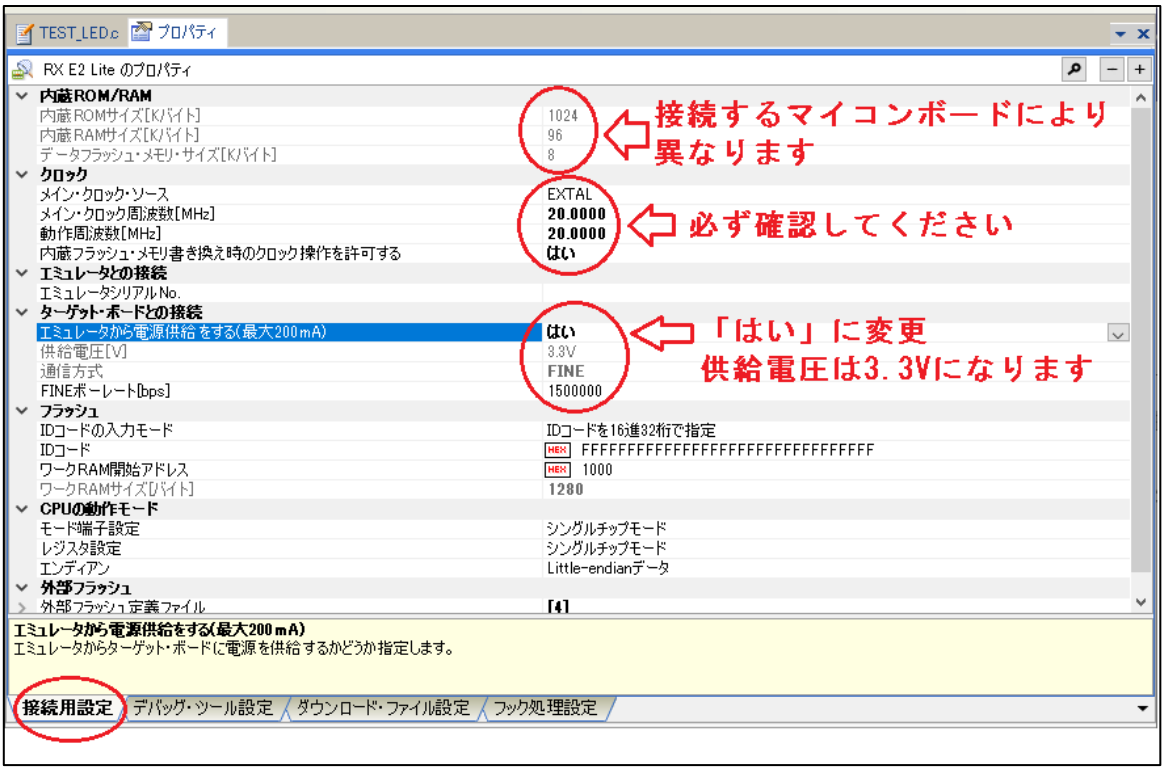
・「動作周波数[ＭＨｚ]」＝20（入力すると 20.0000 と表示されます）

・「内蔵フラッシュ・メモリ書き換え時のクロック操作を許可する」枠内をクリックして現れる右端の「Ｖ」マークをクリックし「はい」に変更

・「エミュレータから電源供給をする（最大 200mA）」枠内をクリックして現れる右端の「Ｖ」マークをクリックして「はい」に変更

・「供給電圧」が３．３Ｖである事を確認

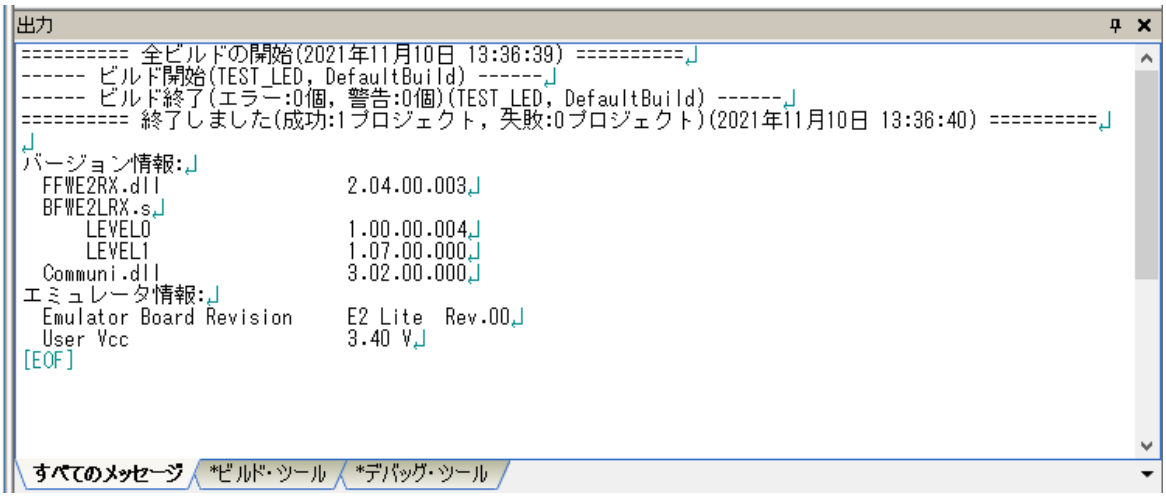
以上を確認し、違っている場合は書き換えてください。



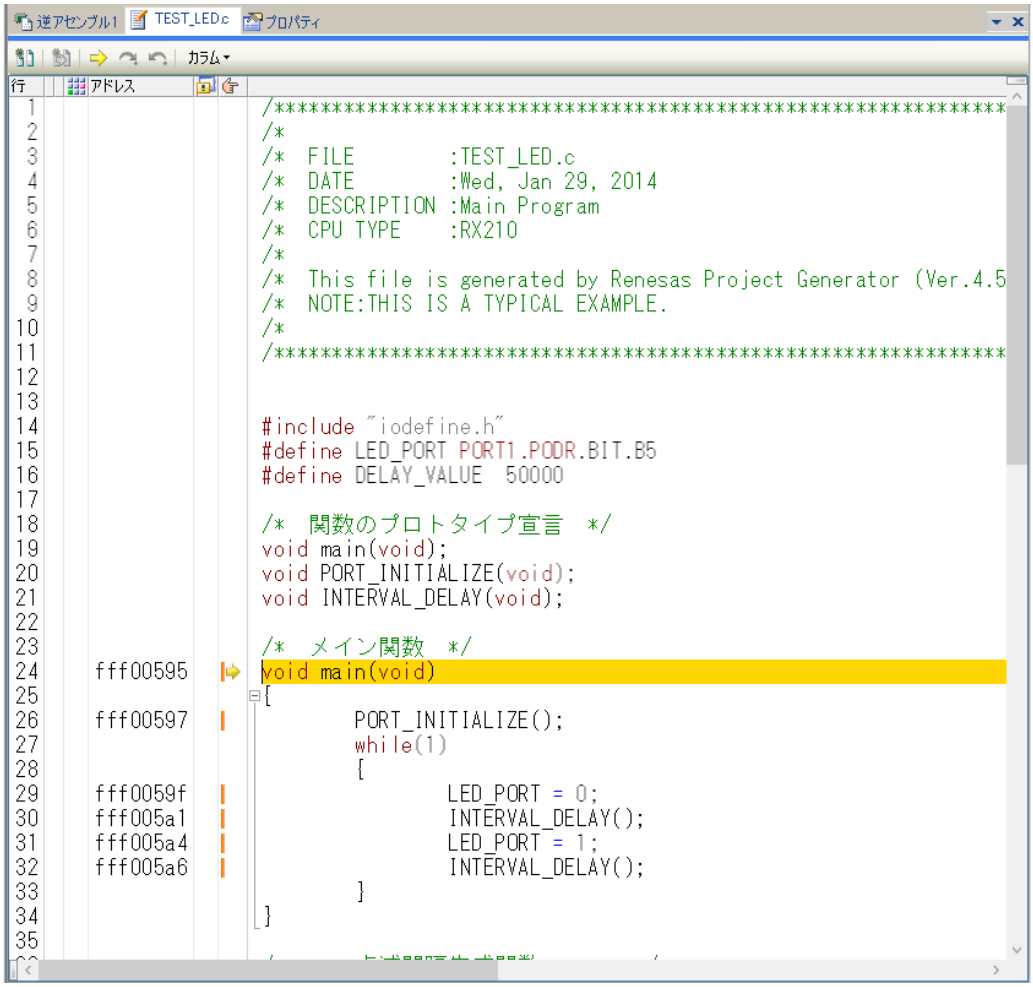
5）ビルドとデバッグ

ツールバー「ﾃﾞﾊﾞｯｸﾞ(D)」⇒「ﾋﾞﾙﾄﾞ&ﾃﾞﾊﾞｯｸﾞ･ﾂｰﾙへﾀﾞｳﾝﾛｰﾄﾞ(B)」をクリック

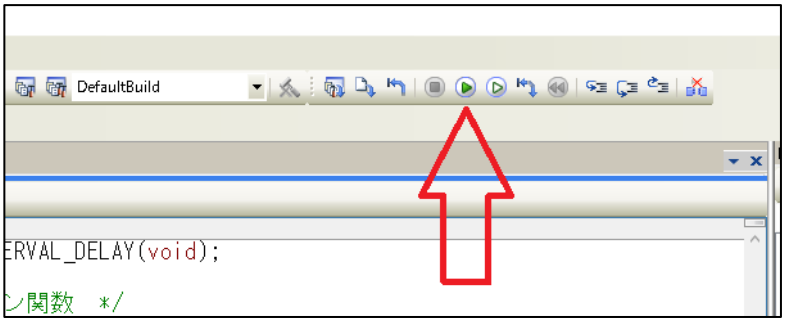
「出力」画面には下記の様に表示されます。



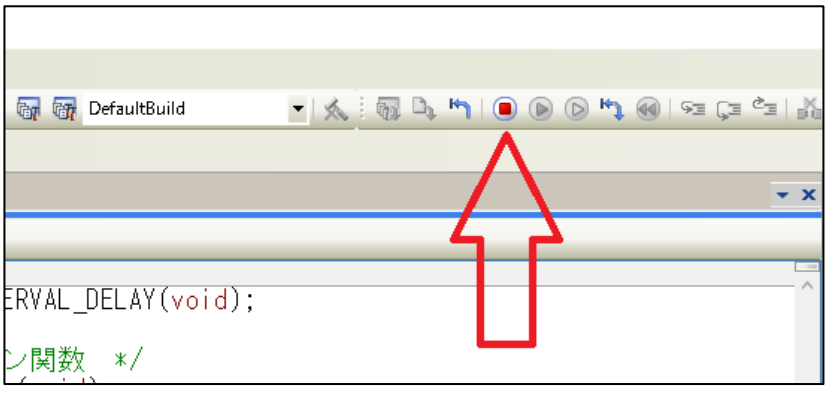
6）「TEST_LED.c」のタブの横には「逆アセンブル」タブが現れます。



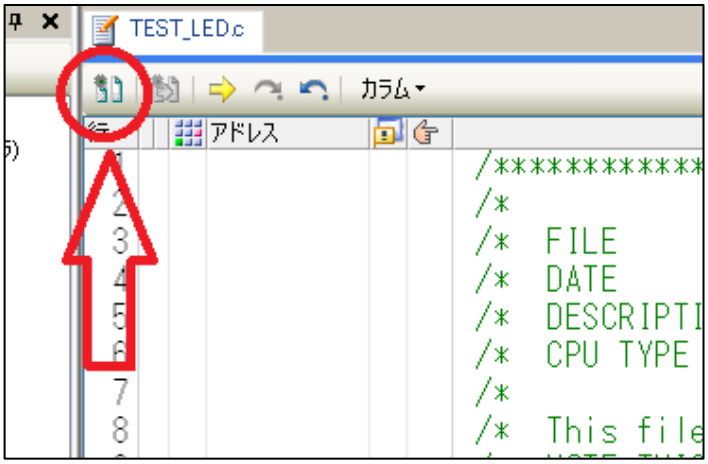
7）ダウンロードが完了するとプログラム実行可能になります。
ツールバー下の「横向き▲（プログラムを現在の位置から実行します）」をクリックするとプログラムが実行されます。



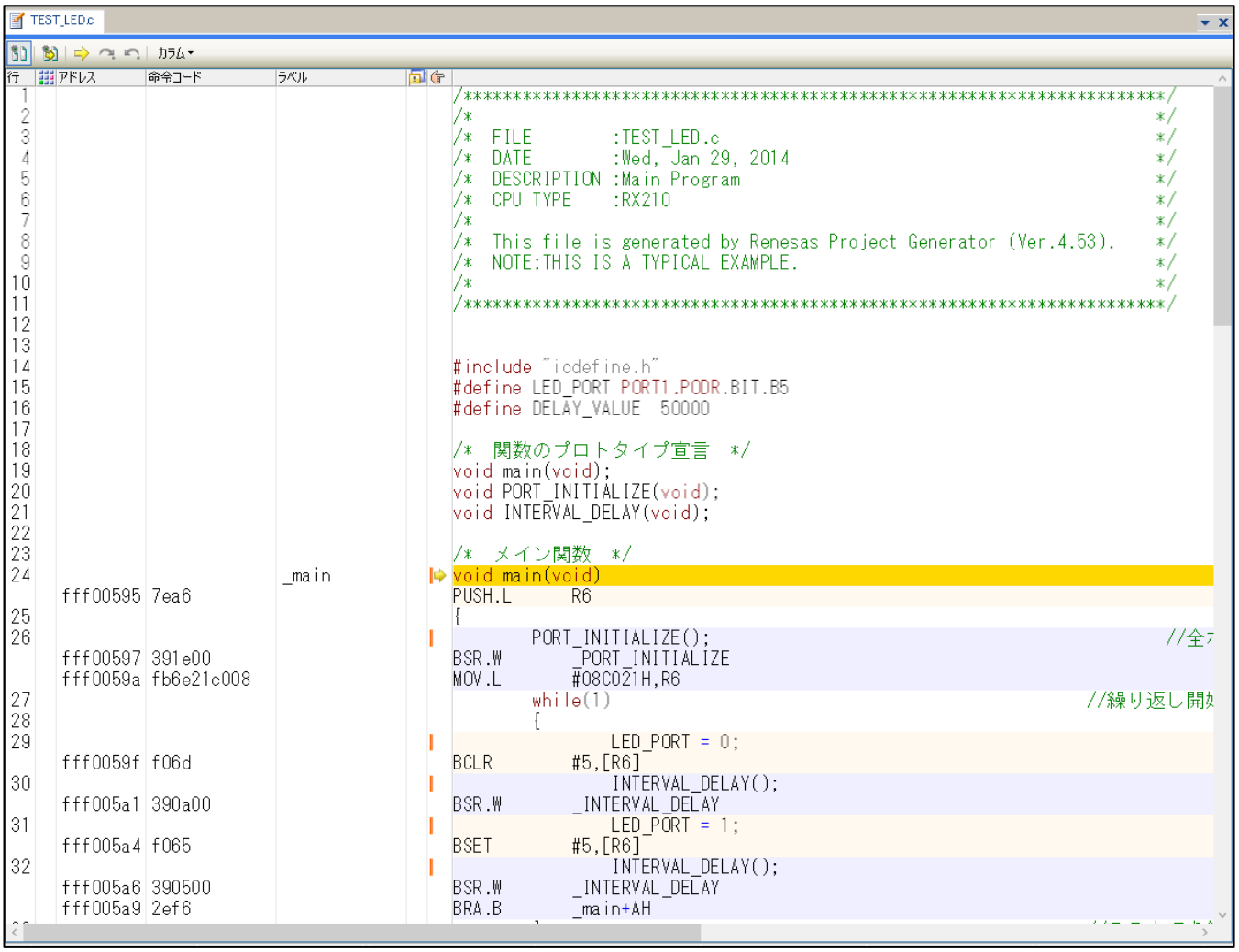
8）実行中にツールバー下の「■（実行中のプログラムを停止します）」をクリックすると停止します。



9）デバッグを行います。「TEST_LED.c」タブの下の（逆アセンブルとソースを混合して表示します）ボタンをクリックして双方を表示させます。



１０）逆アセンブルとソースが同時表示されます。



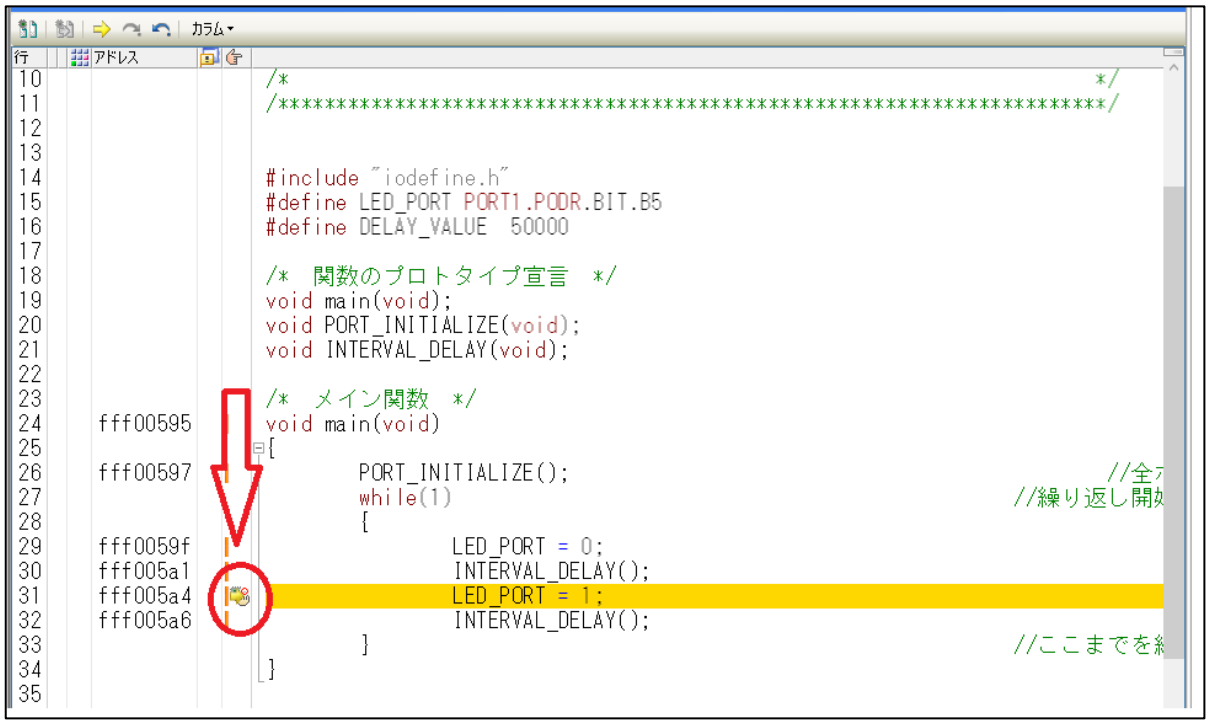
■ブレークポイントの設定■

ソースファイル左のブレークポイント指定欄にカーソルを置くとブレークポイントの印が現れます。その直前までを実行して停止させる事ができます。

下の図の例では、３１行目の「LED_PORT = 1; 」にブレークポイントを置いています。

緑の横向き▲の（プログラムを現在の位置から実行します。）をクリックすると、直前３０行目の「INTERVAL_DELAY(); 」関数を実行終了した時点で停止し、ＬＥＤは点灯したままになります。

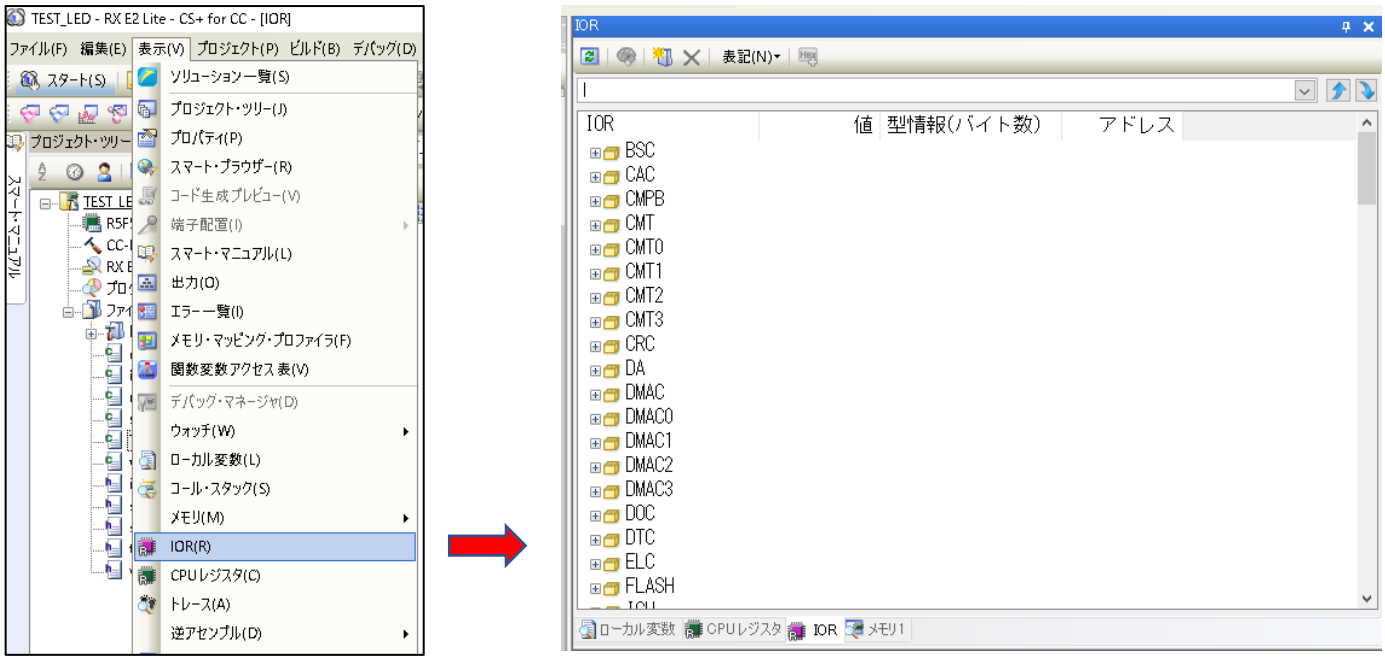
もう一度同じく緑の横向き▲をクリックすると、３１行目を実行（ＬＥＤ消灯）、３２行目を実行（２秒待ち）した後に２９行目（ＬＥＤ点灯）に戻り、再び３０行目を実行終了した時点で停止し、ＬＥＤは点灯したままになります。



■Ｉ／Ｏの確認・変更■

１）ツールバーの「表示」－「ＩＯＲ(R)」で、レジスタの内容が表示されます。

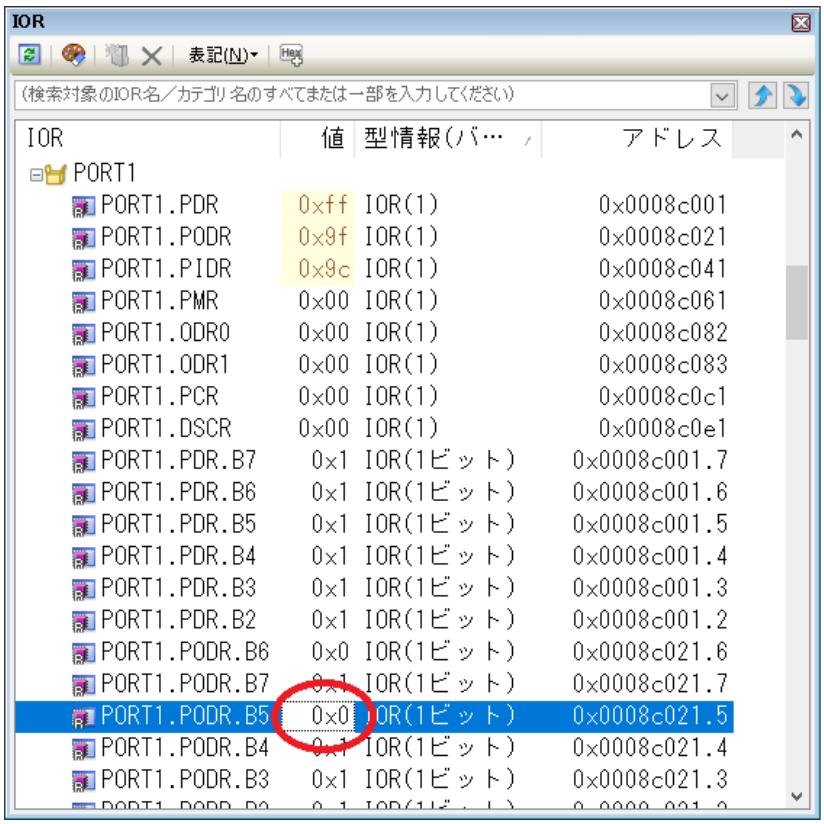
ソースコードの隣に表示されますが、位置や幅は見やすい状態に調節してください。



2）「PORT1」－「PODR」の「Ｂ５」が　ＬＥＤのポートです。（PORT1.PODR.B5）

縦スクロールバーで「ＰＯＲＴ１」まで移動し、左の田印をクリックすると、ＰＯＲＴ１の各レジスタが表示されます。「PORT1.ＰＯＤＲ.B5」桁の「値」をダブルクリックして値を変更Ｅｎｔｅｒキーで確定してください。

ＬＥＤは抵抗を介してＶＤＤに接続されている為、０で点灯、１で消灯です。



３、「■ブレークポイントの設定■」で３２行目にカーソルを置いた場合は、直前３１行目の「LED_PORT = 1; 」を実行して停止しますので、「PORT1.PODR.BIT.B5」は１になり、ＬＥＤは消灯します。

■定数の変更■

1）ツールバー「デバッグ」⇒「デバッグ・ツールから切断」をクリックし、
　ソースファイル冒頭の「DELAY_VALUE」を 50000 から 5000 に変更します。

```
#include "iodefine.h"
#define LED_PORT PORT1.PODR.BIT.B5
#define DELAY_VALUE  50000
```

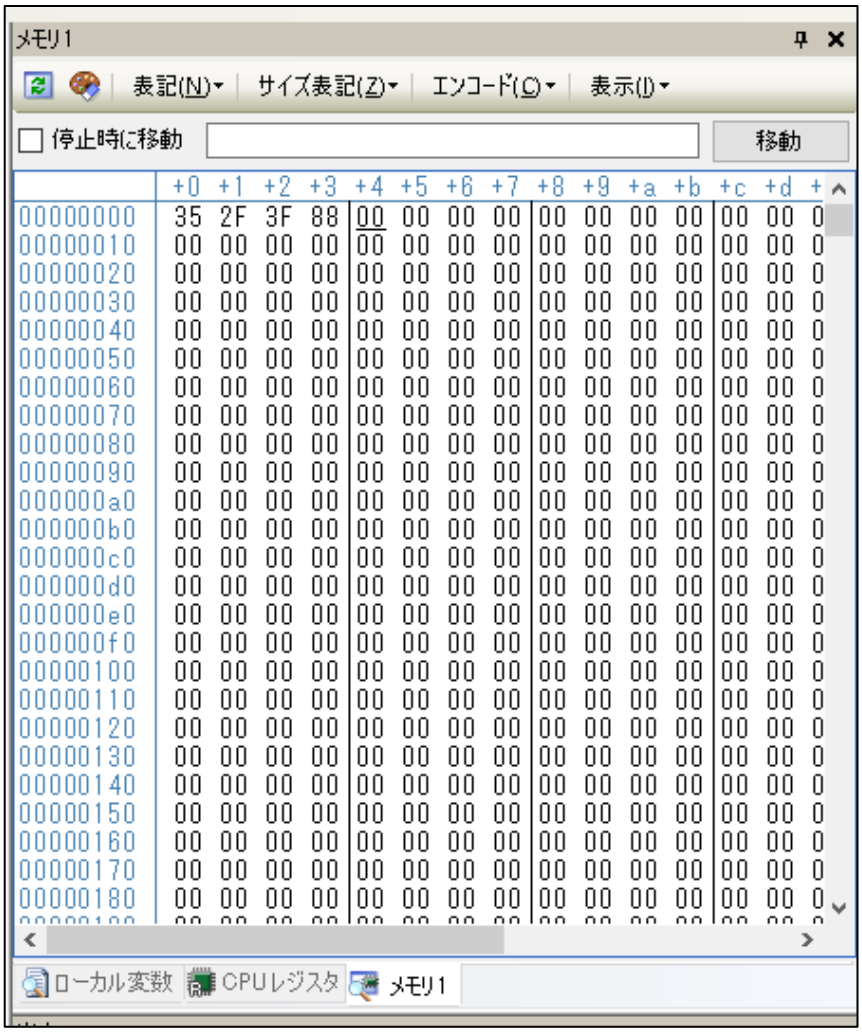
⇒

```
#include "iodefine.h"
#define LED_PORT PORT1.PODR.BIT.B5
#define DELAY_VALUE  5000
```

2）ツールバー「デバッグ」⇒「ビルド＆デバッグ・ツールへダウンロード」をクリック
　緑の横向き▲の（プログラムを現在の位置から実行します。）をクリックすると、点滅が早くなっているのが確認できます。確認できたら■をクリックして停止させます。

■メモリの変更■

1）ソースコードの「/*　点滅間隔生成関数　*/」内の値を直接変更します。
　ツールバー「表示」⇒「メモリ」⇒「メモリ１」でメモリ内容が表示されます。

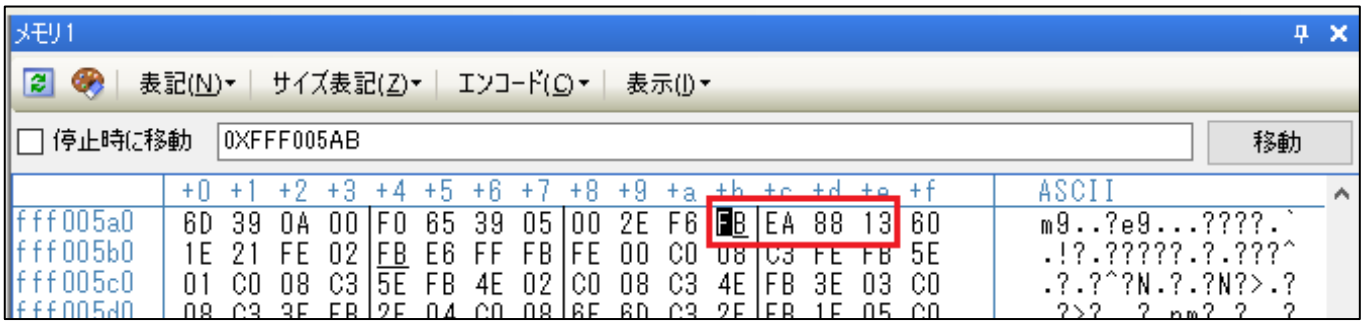


2）変更するメモリのアドレスを表示します。
アドレス「fff005ab」の内容は「fbae8813」となっています。アセンブラは「MOV.L #1388H,R14」です。
この 1388 が４６）で修正した「DELAY_VALUE 5000」（５０００は１６進数の１３８８H）です。
メモリには上位下位が逆に格納されていますので 8813 となっています。

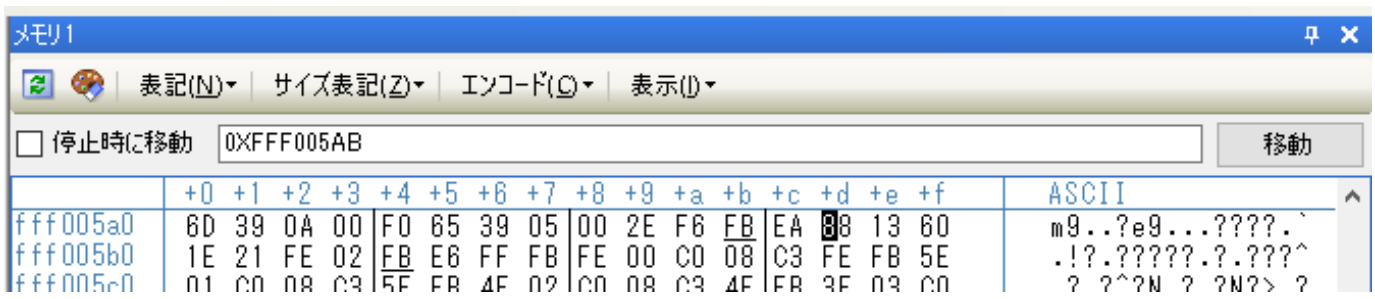
```
36                                      /*      点滅間隔生成関数        */
37                          _INTERVAL_DEL void INTERVAL_DELAY(void)
    fff005ab fbea8813               MOV.L       #1388H,R14
    fff005af 601e                   SUB         #1H,R14
    fff005b1 21fe                   BNE.B       _INTERVAL_DELAY+4H
38                                      {
39                                              unsigned  long L;
40                                              for ( L = 0 ; L < DELAY_VALUE ; L ++ ); //WAIT
41                                      }
    fff005b3 02                     RTS
```

3）「メモリ１」表示窓にアドレス fff005ab を表示させます。
アドレス入力窓に「0xfff005ab」または「0XFFF005AB」(0xFFF005AB)と入力して右の「移動をクリック」（最初の「0X」「0x」は１６進数を表します）

4）アドレス fff005a0 から表示され、指定したアドレス fff005ab がアンダーライン付きで表示されます。
FBEA8813 がメモリに格納されている機械語です。8813 が１６進数の 1388、十進数の 5000 になるので、ここを変更します。



5）変更するアドレス fff005ad の 88 をクリックすると反転表示になりますので、ここから十進数 50000、16 進数で C350 を格納していきます。上位下位が逆になるので 50C3 です。88 の部分から 50C3 と入力してキーボードから **Enter を押して確定**します。



６）メモリを変更した値を確認します。緑色横向き▲で実行し■で停止します。
　　ＬＥＤの点滅周期が遅くなっている事を確認してください。

７）この状態ではＣ言語ソースファイルは変更されていません（マイコン内のメモリは変更されています）。
　　必要に応じてソースファイル「TEST_LED.c」の内容を変更してください。

８）マイコンからユーザインタフェースケーブルをはずし、ＣＮ４（ＵＳＢコネクタ）とパソコンを接続
　　し電源を供給すると、この変更されたメモリの状態でプログラムが実行されます。
　　（このサンプルプログラムの場合、ＳＷ２のＤＩＰスイッチは全てＯＦＦで実行に問題ありません）

９）メニューバーの「ファイル」－「終了」でプログラムを終了してください。
　　（変更を保存する必要がある場合は、保存して終了してください）

より詳しい操作説明はＲＥＮＥＳＡＳ社ホームページのＥ２エミュレータＬｉｔｅのマニュアルをご覧ください。

・次ページからは R5F52108ADFP ボードと R5FT210BBDFP ボードの違いによるプログラミングの相違点を説明いたします。

R5F52108ADFP ボードと R5FT210BBDFP ボードの違いによるプログラミングの相違点

R5F52108ADFP ボード（以下 52108A と記述）と R5F5210BBDFP ボード(以下 5210BB と記述)は以下の部分で違いがあります。プログラミングの際にはご注意ください。

ＣＰＵ	５Ｆ５２１０８ＡＤＦＰ （チップバージョンＡ）	Ｒ５Ｆ５２１０ＢＢＤＦＰ （チップバージョンＢ）
ＲＯＭ	５１２Ｋ	１Ｍ
ＲＡＭ	６４Ｋ	９６Ｋ
システムクロックコントロールレジスタ３ （ＳＣＫＣＲ３）	クロックソースとしてのメインクロック発振器	
	選択不可	選択可
ＰＬＬ電源コントロールレジスタ （ＰＬＬＰＣＲ）	ＰＬＬの電源を切り低消費電力にする機能	
	なし	あり
クロック周波数精度測定回路 ＣＡＣコントロールレジスタ１（ＣＡＣＲ１）	周波数測定クロックとしてのメインクロック発振器出力クロック	
	選択不可	選択可
クロック周波数精度測定回路 ＣＡＣコントロールレジスタ２（ＣＡＣＲ２）	基準信号生成クロックとしてのメインクロック発振器出力クロック	
	選択不可	選択可
動作電力コントロールレジスタ （ＯＰＣＣＲ）	中速動作モード２Ａ、２Ｂ	
	なし	あり
スリープモード復帰クロックソース切り替えレジスタ （ＲＳＴＣＫＣＲ）	スリープモード復帰クロックソースとしてのメインクロック発振器	
	選択不可	選択可
フラッシュＨＯＣＯソフトウェアスタンバイコントロールレジスタ（ＦＨＳＳＢＹＣＲ）	ソフトウェアスタンバイモードのフラッシュメモリの電源供給	
	制御必要	制御不要
５Ｖトレラント対応Ｉ／Ｏポート	Ｐ１２、Ｐ１３，Ｐ１６	Ｐ１２，Ｐ１３，Ｐ１６，Ｐ１７
ＤＣ特性	消費電流特性の改善なし	消費電流特性の改善あり
ＲＯＭ（コード格納用フラッシュメモリ）特性	プログラム・イレース	
	１０００回	１００００回
	データ保持	
	１０００回／１０年	１０００回／30 年 １００００回／１年
コールドスタート時のサブクロック制御回路初期化	必要	必要
１を書き込み必須の ポート方向レジスタ（ＰＤＲ）予約ビット	なし	Ｐ００～Ｐ０２，Ｐ５６ Ｐ６０～Ｐ６７，Ｐ７０～Ｐ７７ Ｐ８０～Ｐ８７，Ｐ９０～Ｐ９７ ＰＦ５，ＰＪ５，ＰＫ２～ＰＫ５ ＰＬ０、ＰＬ１
リアルタイムクロック（ＲＴＣ）のクロック設定手順	なし	ＲＴＣ ソフトウェアリセット実施前に ＰＣＲ２レジスタのｂ７を ０に設定
シリアルペリフェラルインターフェイス ＲＳＰＩビットレジスタ（ＳＰＢＲ）	１６．０Ｍｂｐｓ設定禁止	１６．０Ｍｂｐｓ設定可能

次ページではプログラムに関連する各項目の詳細を説明したします。

■システムクロックコントロールレジスタ３（ＳＣＫＣＲ３）■

このレジスタの書き換えにはプロテクトレジスタ（ＰＲＣＲ）のＰＲＫＥＹビットをＡ５ｈ、該当プロテクトビット（ＰＲＣＲ０）を１にして書き込みを許可し、書き換え後は０にして書き込みを禁止します。

詳細はマニュアル内「プロテクトレジスタ（ＰＲＣＲ）」を参照してください。

・52108A ではメインクロック発振器（２０ＭＨｚ水晶発振子）からの出力をそのままシステムクロックとして選択する事が禁止されています。この発振器出力を使用して２０ＭＨｚのシステムクロックを得るには

①システムクロックコントロールレジスタ（ＳＣＫＣＲ３）のクロックソース選択ビット（ＣＫＳＥＬ）でＰＬＬ回路を選択

②ＰＬＬコントロールレジスタ（ＰＬＬＣＲ）で
分周器比率をＰＬＬＤＩＶで÷４設定、ＰＬＬ回路倍率をＳＴＣでｘ８設定

③システムクロックコントロールレジスタ（ＳＣＫＣＲ）のシステムクロック選択ビット（ＩＣＫ）で２分周を設定

上記で２０ＭＨｚ÷４ｘ８÷２＝２０ＭＨｚのシステムクロックが得られます。

・5210BB ではシステムクロックコントロールレジスタ（ＳＣＫＣＲ３）のクロックソース選択ビット（ＣＫＳＥＬ）で直接メインクロック発振器出力をシステムクロックとして選択できます。

■ＰＬＬ電源コントロールレジスタ（ＰＬＬＰＣＲ）■

このレジスタの書き換えも前項と同じくプロテクトビット（ＰＲＣＲ０）を１にする必要があります。

・52108A にはこのレジスタはありませんがアドレスは存在し、ＣＳ＋上でも定義されています。
レジスタビット０（ＰＬＬＰＣＮＴ）の値は読み出し書き込み共に可能ですが、低消費電力機能はありません。
ビット７～１は予約ビットです。読むと０が読めます。書く場合、０としてください。

■クロック周波数精度測定回路ＣＡＣコントロールレジスタ（ＣＡＣＲ１）■

・52108A は周波数測定クロックとしてメインクロック発振器出力クロックを使用できません。
ＦＭＣＳビット２～０に０００を設定した場合、動作は保証されません。

・5210BB には制約はありません。

■クロック周波数精度測定回路ＣＡＣコントロールレジスタ（ＣＡＣＲ２）■

・52108A は基準信号生成クロックとしてメインクロック発振器出力クロックを使用できません。
ＲＳＣＳビット２～０に０００を設定した場合、動作は保証されません。

・5210BB には制約はありません。

■動作電力コントロールレジスタ（ＯＰＣＣＲ）■

ｂ２～０がＯＰＣＭビットで、クロック（システム、周辺モジュール、外部バス）を制限する事で動作電力をコントロールします。

・52108A ではビット２～０（ＯＰＣＭ）に１００、１０１の設定(中速動作モード２Ａ、２Ｂ)は禁止されています。読み出し書き込み共に可能ですが、動作は保証されません。

■スリープモード復帰クロックソース切り替えレジスタ（ＲＳＴＣＫＣＲ）■

ｂ２～０がＲＳＴＣＫＳＥＬビットで、下記の様に機能します。

・52108A ではシステムクロックにメインクロックを設定できません。それによりスリープモードからの復帰の際はＨＯＣＯ（高速オンチップオシレータ）の一択となります。ビット２～０に００１以外を設定しないで

ください。
・5201BB ではスリープモードからの復帰にＨＯＣＯとメインクロックの二択となります。ビット２～０には００１（ＨＯＣＯ）、０１０（メインクロック）以外を設定しないでください。

■フラッシュＨＯＣＯソフトウェアスタンバイコントロールレジスタ（ＦＨＳＳＢＹＣＲ）■
ｂ２～０がＳＯＦＴＣＵＴビットで、下記の様に機能します。
ビット２＝電源検出回路の動作とＰＯＲ（パワーオンリセット）の低消費電力機能決定ビット。
１で電源検出回路は動作、ＰＯＲ低消費電力機能有効
０で電源検出回路は停止、ＰＯＲ低消費電力機能無効
ビット１＝ＨＯＣＯ（高速オンチップオシレータ）電源供給ビット
１でＨＯＣＯ電源供給ＯＦＦ
０でＨＯＣＯ電源供給ＯＮ
ビット０＝フラッシュメモリへの電源供給ビット（5210BB ではどちらを選択しても大丈夫です）
１でフラッシュメモリ電源ＯＦＦ
０でフラッシュメモリ電源ＯＮ
・52108A ではソフトウェアスタンバイモード時のフラッシュメモリの電源供給を制御する必要があります。プログラム時にビット０を書き込んでください。
・5210BB ではフラッシュメモリの電源供給を考慮する必要がありません。それによりフラッシュメモリへの電源供給関連ビット０は二択同義（don’t care）となります。

■5V トレラント対応Ｉ／Ｏポート■
・52108A ではＰ１２、Ｐ１３、Ｐ１６が５Ｖトレラントです。
・5201BB では上記３ポートに加えてＰ１７が5V トレラントです。

■ポート方向レジスタ（ＰＤＲ）予約ビット■
・52108A では下記制約はありません。
・5210BB では下記ビットに「１」を書き込む事が必須です。
Ｐ００～Ｐ０２、Ｐ５６、Ｐ６０～Ｐ６７、Ｐ７０～Ｐ７７、Ｐ８０～Ｐ８７、Ｐ９０～Ｐ９７、
ＰＦ５、ＰＪ５、ＰＫ２～ＰＫ５、ＰＬ０～ＰＬ１

■リアルタイムクロック（ＲＴＣ）のクロック設定手順■
・52108A では制約はありません。
・5210BB ではＲＴＣソフトウエアリセット実施前にＰＣＲ２レジスタのｂ７を０に設定しください。

■シリアルペリフェラルインターフェイス（ＲＳＰＩ）の設定注意事項■
ＲＳＰＩビットレートレジスタ（ＳＰＢＲ）および
ＲＳＰＩコマンドレジスタのビットレート分周設定ビット（ＢＲＤＶ）
・52108A では上記２つのレジスタを同時に０にしてビットレートに１６Ｍｂｐｓを選択する事はできません。
・5201BB では制約はありません。