（ＲＥＶ.３）

RX220 マイコンの
プログラムを作って、動かしてみよう

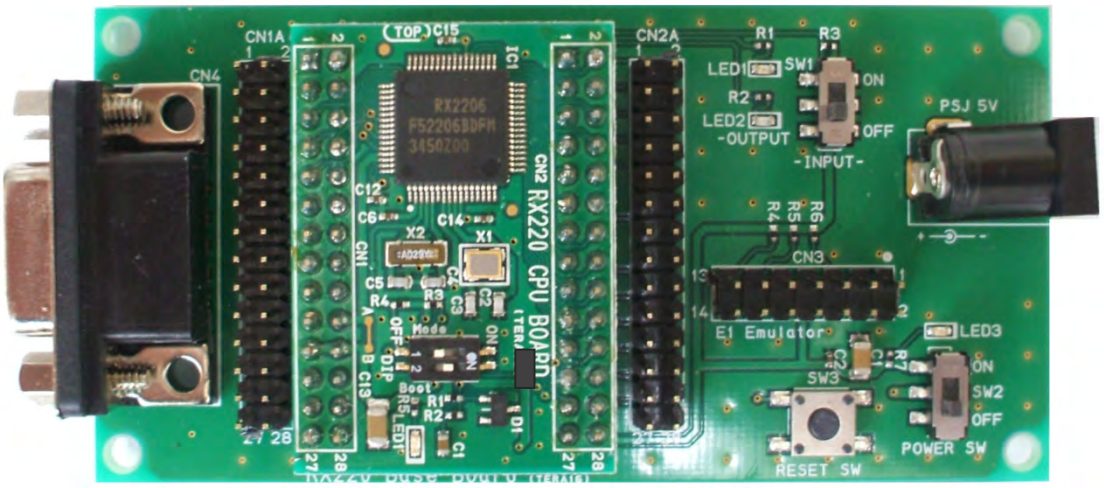
プログラムの作成から動作まで、
一連の操作方法が判ります。

［基礎編］

＊ポイント解説＊

- 開発ソフトのダウンロード操作方法
- プログラムの作成操作方法
- 「RX220 CPU ボード」の設定・操作方法
- 「RX220 Base ボード」の設定・操作方法
- 「E1 エミュレータ」の設定・操作方法
- クロック発生回路と制御プログラム
- サンプルプログラム

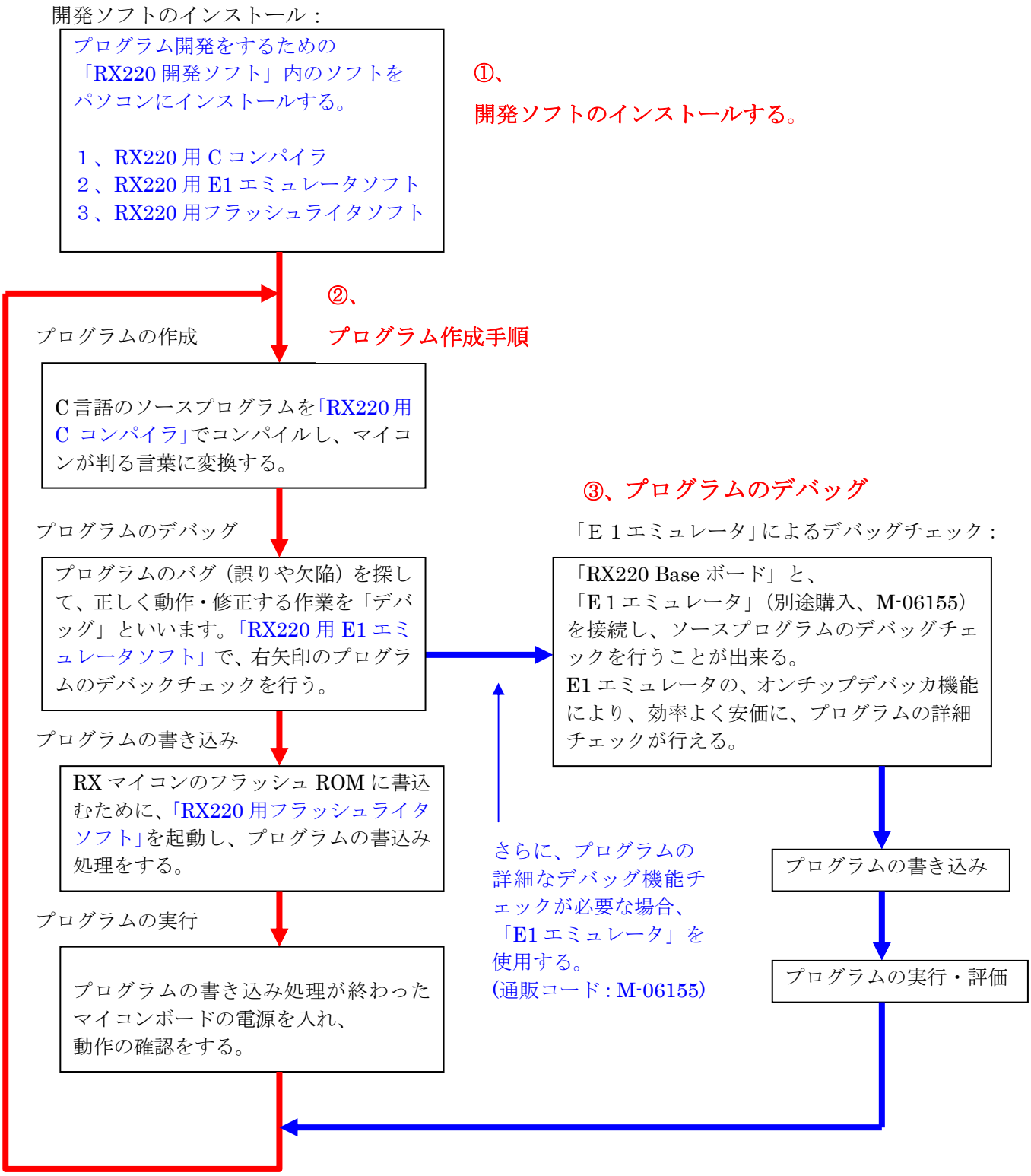
Ｂｙ　（株）秋月電子通商



- 「RX220 CPU ボード」＋「RX220 Base ボード」 -
40mm x 29mm　　95mm x 47mm

■ 開発ソフトのダウンロード手順：

添付 CD の中のフォルダー名「RX220 開発ソフト」を開き、
Windows7 以上の PC(パソコン)に、開発ソフトをダウンロードします。



１、開発ボードの準備：

RX220 マイコンボードは、「RX220　CPU ボード」と「RX220　Base ボード」で、構成されています。
「RX220　CPU ボード」は、40mm x 30mm の超小型 CPU ボードです。「RX220　Base ボード」は、
95mm x 47mm のベース（土台）となるボードで、AC アダプタ端子（５V）、E1 エミュレータ端子、
シリアル書き込み端子が搭載されています。

第１章は、D-Sub 端子（９pin）を使用した、シリアル書き込みによる操作方法を説明しております。
PC（パソコン）と「RX220 Base ボード」の D-Sub 端子（９pin）を接続したプログラム書き込み方法
について、USB－シリアル変換ケーブル（通販コード：M-02747）を使用して実際に即した内容で、
説明致します。
電源は、「RX220 Base ボード」の AC アダプタ端子（５V）から供給します（通販コード：M-06590 等）。

第２章は、E1 エミュレータを使用した書き込み操作方法について、説明しております。
PC の USB 端子から電源をもらうので、「RX220　Base ボード」の AC アダプタ端子（５V）からの
電源供給は不要です。

第３章は、E1 エミュレータを使用したデバッグ機能操作方法について、説明しております。
PC の USB 端子から電源をもらうので、ボード電源の AC アダプタは不要です。

第４章は、「クロック発生回路」について、説明しております。

■ **それでは、「RX220 CPU ボード」に、プログラムを書込み、ボードを動作させる手順まで操作をおこなってみましょう。**

（注意）：かならず、本、手順書を最初から順番に読んでいってください。
（ページを飛ばさないでください）

１、開発ソフトのインストール：

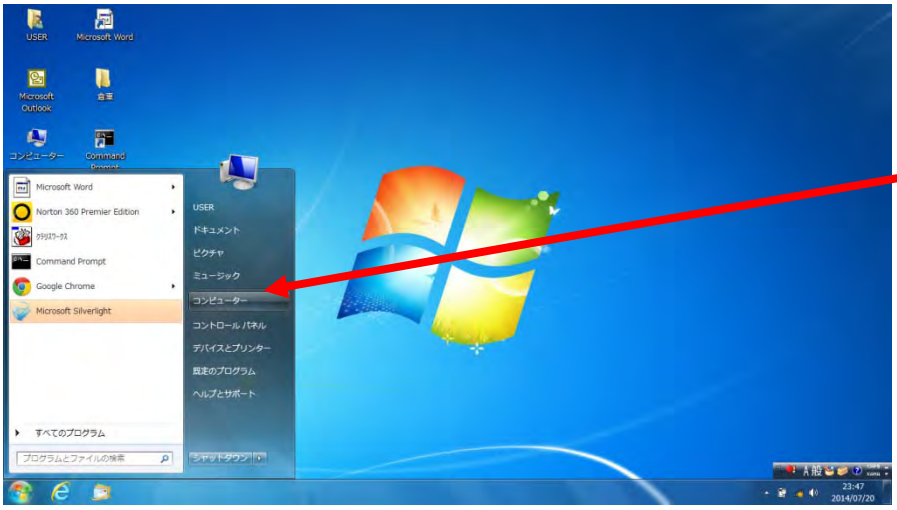
まず初めに、開発ソフトをパソコンにダウンロードします。
PC は、Windows7 以上の搭載機を使用。（Mac OS 機は、使用できません）
ダウンロードするプログラムは、付属ＣＤ内のフォルダー「RX220 開発ソフト」の
「RX220 用 C コンパイラ」、「RX220 用 E1 エミュレータソフト」、
「RX220 用フラッシュライタソフト」の順に、お手持ちのパソコンに、ダウンロードします。
（３本のプログラムは、無償評価版です。注意事項をお読みの上、ご使用してください）

「RX220 用 C コンパイラ」は、C 言語で書いたソースプログラムを
RX220 マイコンが判る言葉に変換するソフトです。
「RX220 用 E1 エミュレータソフト」は、「RX220 CPU ボード」以外に、E1 エミュレータと
「RX220　Base ボード」を使い、オンチップデバッカ機能を実現する為のソフトです。
「RX220 用フラッシュライタソフト」は、パソコンから RX マイコンのフラッシュ ROM に、
プログラムを書込むためのソフトです。PC と「RX220　Base ボード」の D-Sub に接続して、
直接書き込みする方法と、E1 Emulator 端子に E1 エミュレータを接続して書き込みする方法が可能です。
E1 エミュレータ本体は、弊社からも販売中です（通販コード M－06155）。

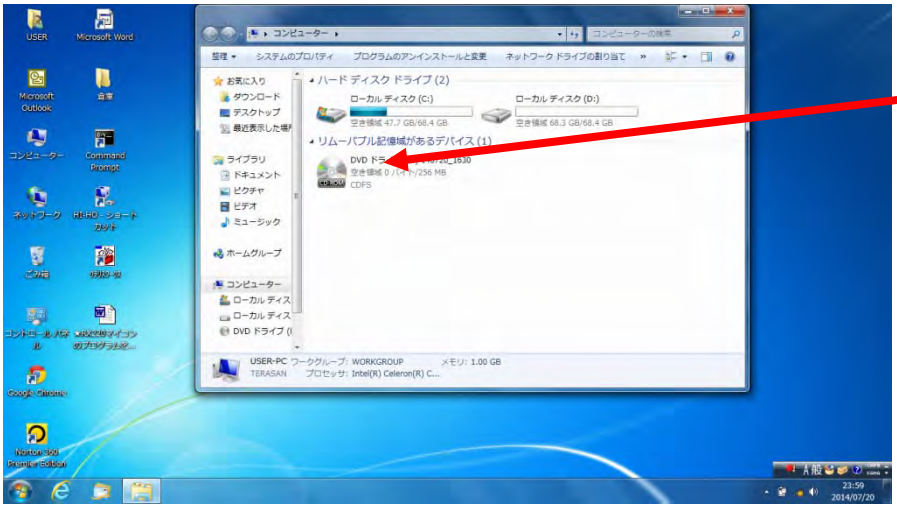
それでは PC に、開発ソフトをダウンロードしてみましょう。

１、まず、ダウンロード対象の PC が、外部機器（インターネット、外付けＤＶＤ、外付けＨＤＤ、ＵＳＢメモリ他）と接続されている場合、全て、外部機器接続を取り外してください。

ＰＣの内蔵ＤＶＤ装置に、今回付属のＣＤ「RX220 開発ソフト」をセットし、
スタート→コンピューター→ＤＶＤドライブ（Ｅ：）と、マウス左をクリックします。



「コンピューター」を、
マウス左で、クリックする。

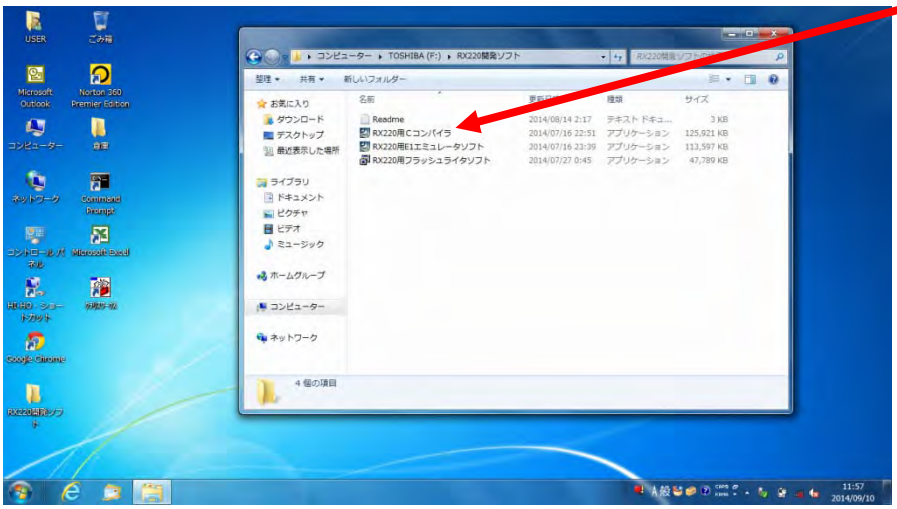


「ＤＶＤドライブ（Ｅ：）」を、
マウス左でクリックする。

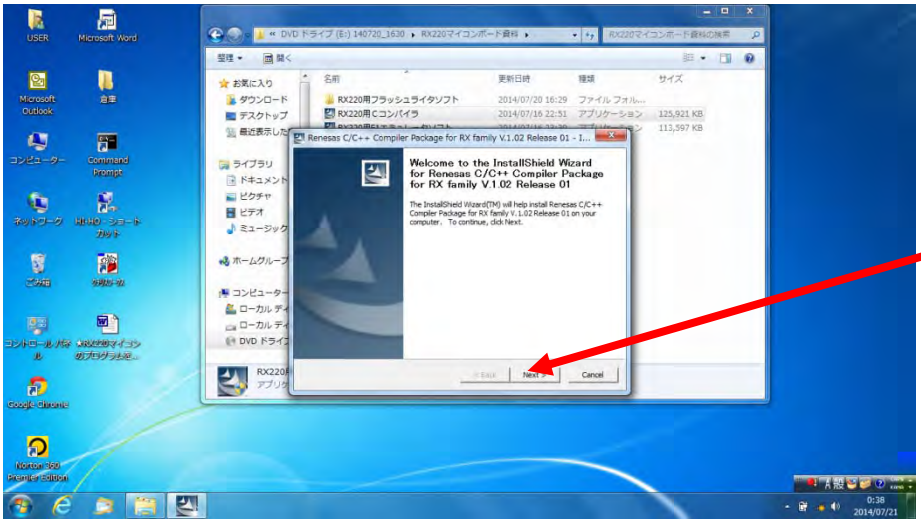
フォルダ「RX220 開発ソフト」を、マウス左クリックします。

まず始めに、

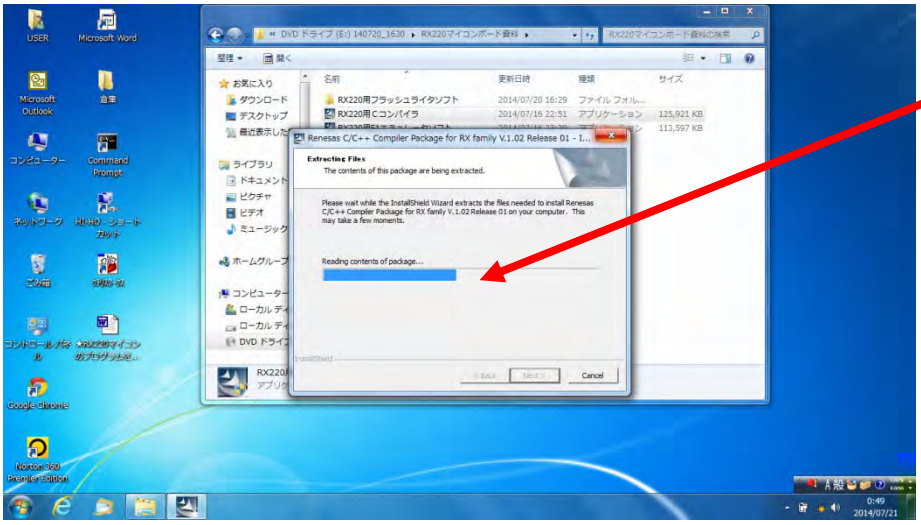
1、「RX220 用Cコンパイラ」をダウンロードする。



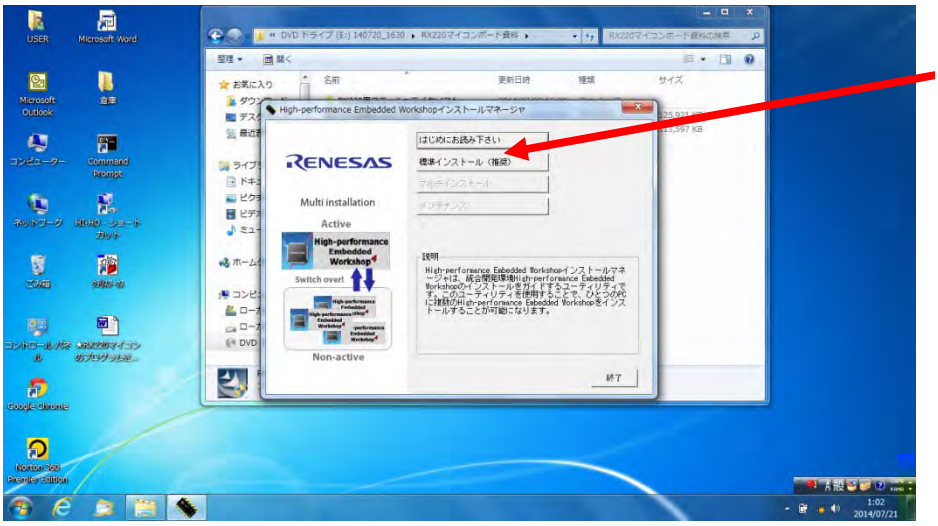
「RX220 用 C コンパイラ」を、マウス左でクリックする。



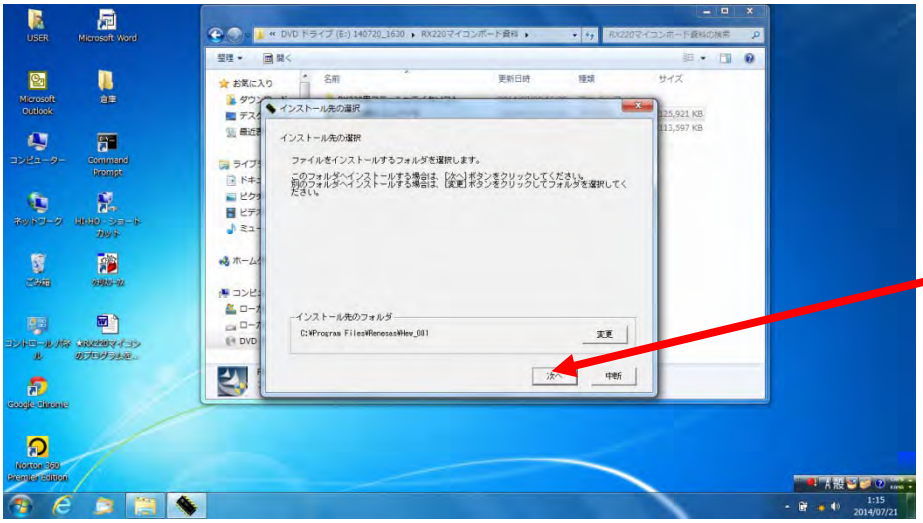
「Ｎｅｘｔ＞」を、
マウス左でクリックする。



プログラムのダウンロードが開始される。

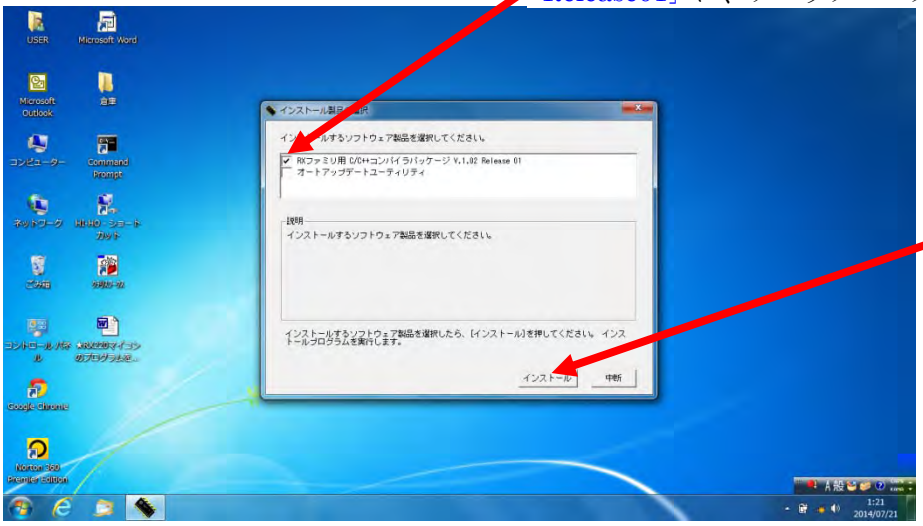


「標準インストール（推奨）」を、
マウス左クリックする。

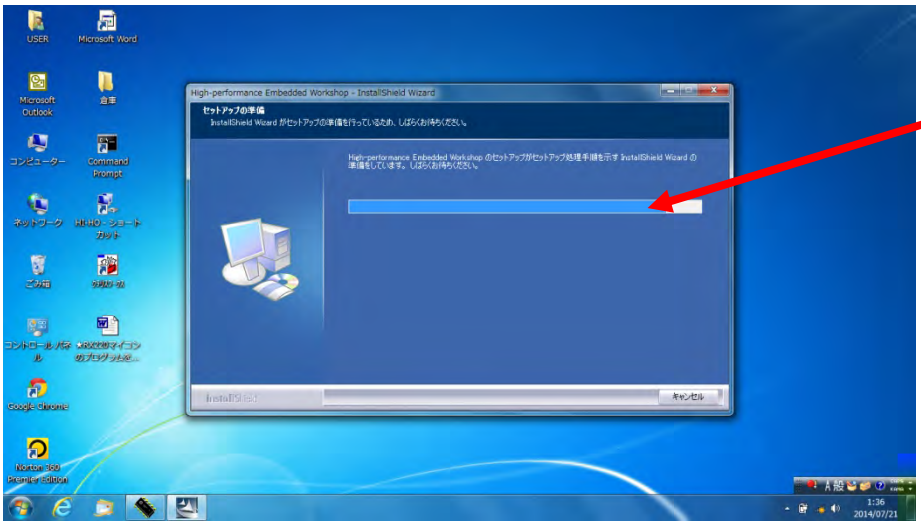


「次へ」を、
マウス左クリックする。

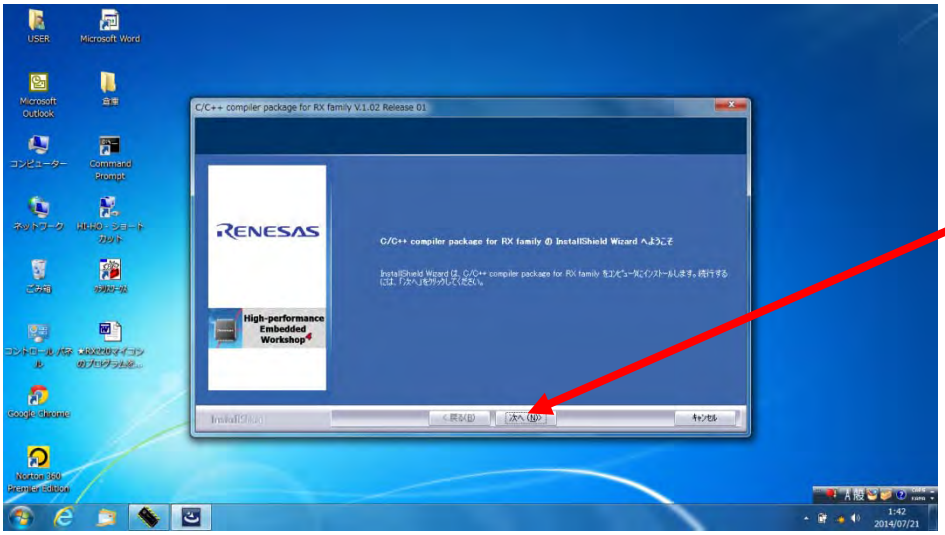
「RX ファミリ用 C/C++コンパイラパッケージ V.1.02 Release01」に、チェックマークをつける。



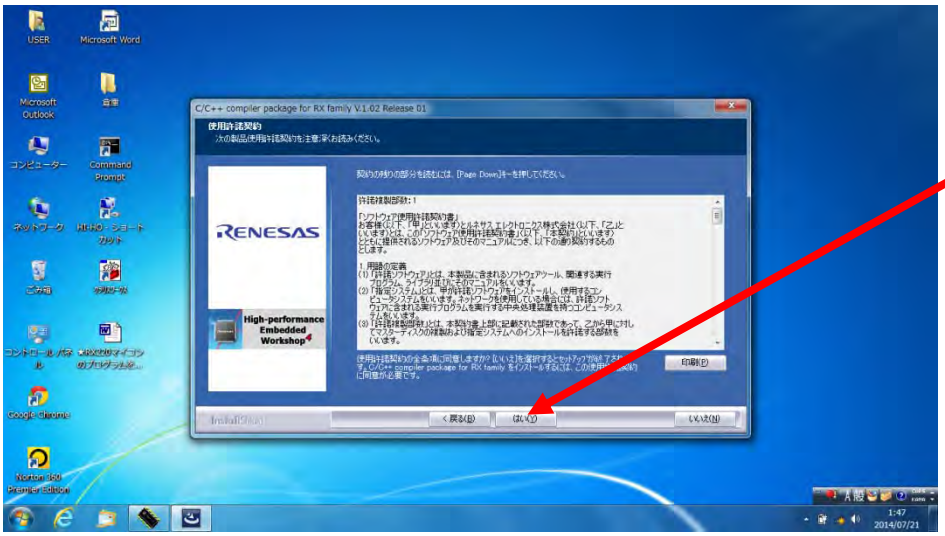
「インストール」を、
マウス左クリックする。



インストールが開始される。

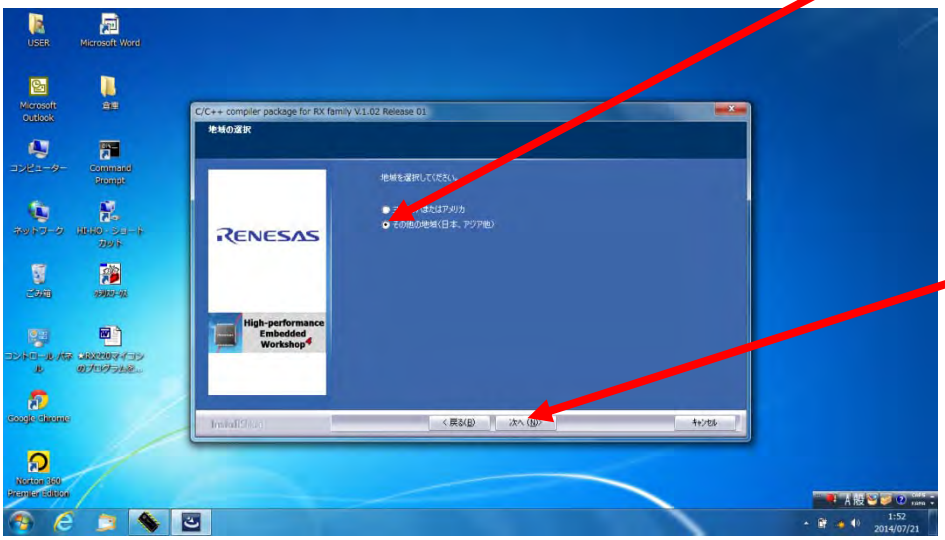


「次へ（N）＞」を
マウス左クリックする。

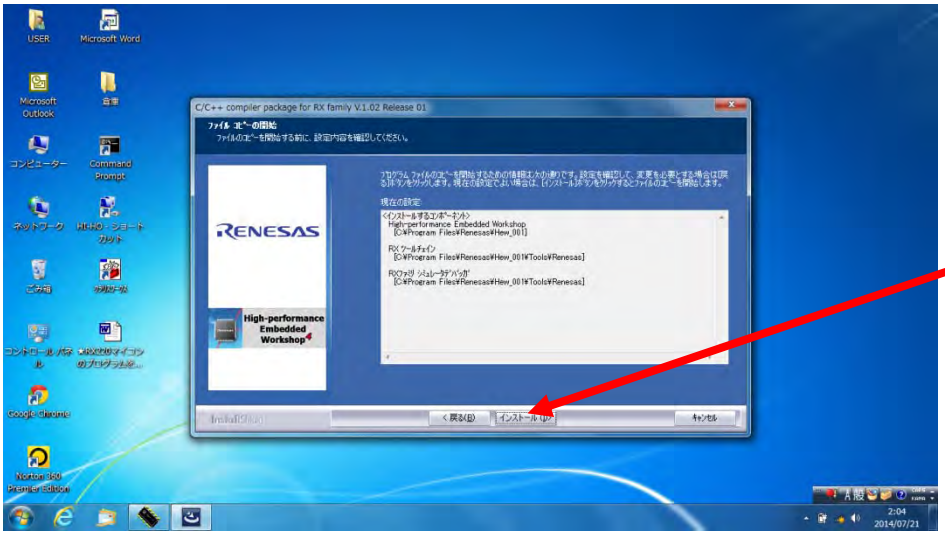


「はい」を
マウス左クリックする。

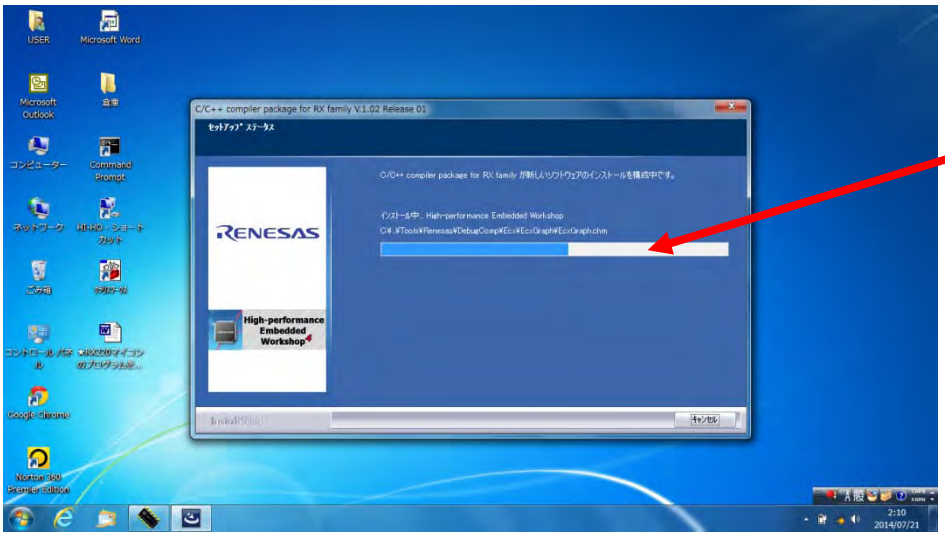
「その他の地域（日本、アジア他）」を、
マウス左クリックし、チェックマークを付ける。



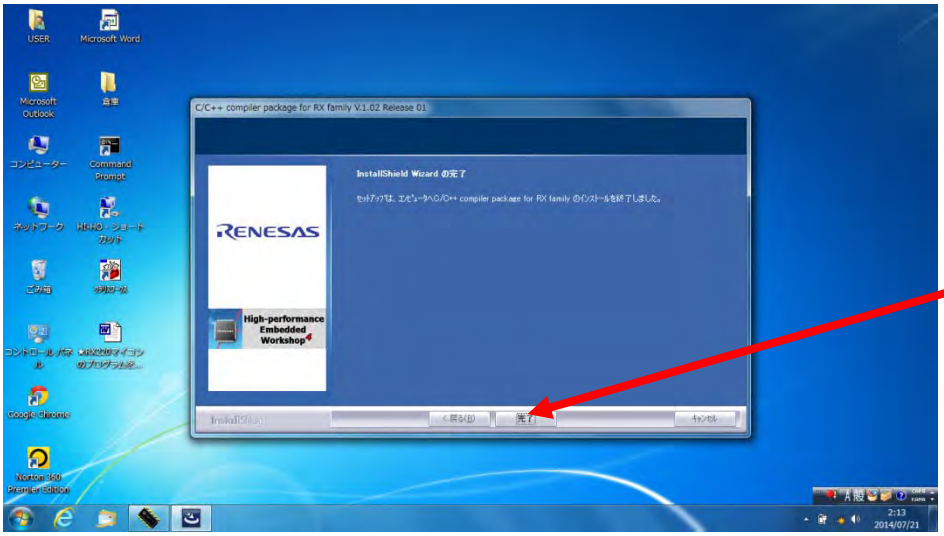
「次へ（N）＞」を
マウス左クリックする。



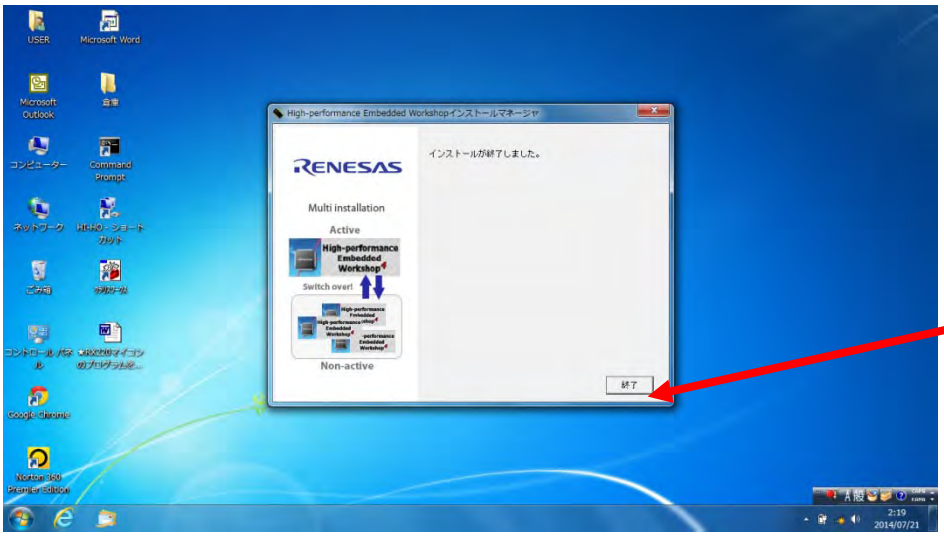
「インストール（I）＞」を
マウス左クリックする。



インストール開始

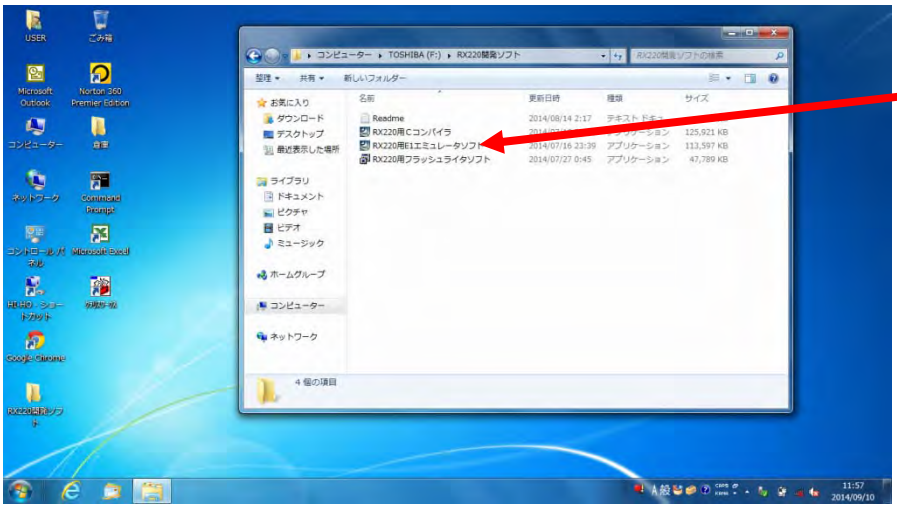


「完了」を
マウス左クリックする。

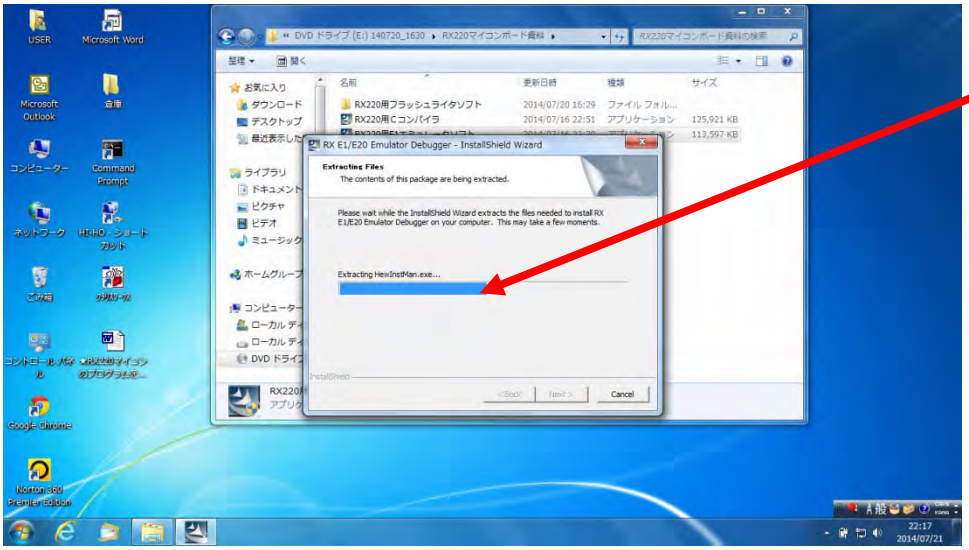


「終了」を
マウス左クリックする。

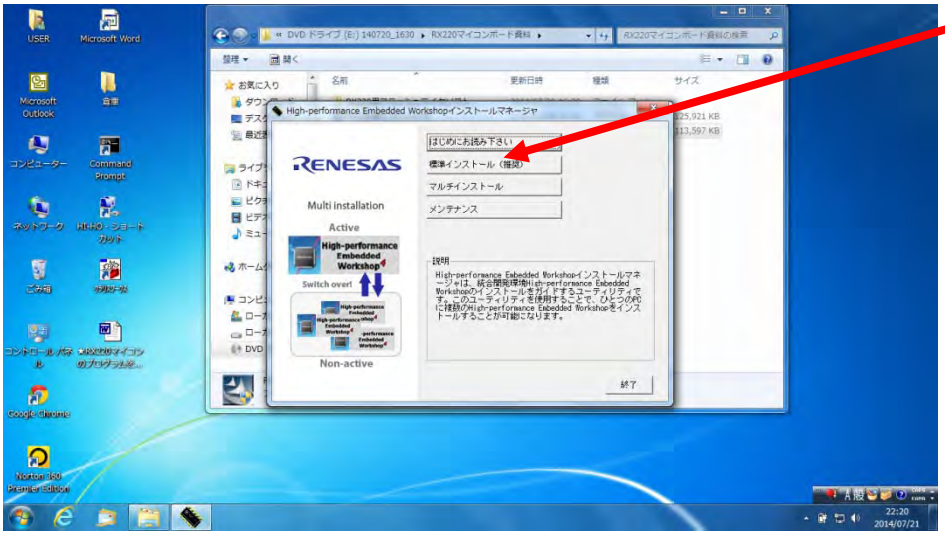
次に、
2、「RX220 用 E1 エミュレータソフト」をダウンロードする。



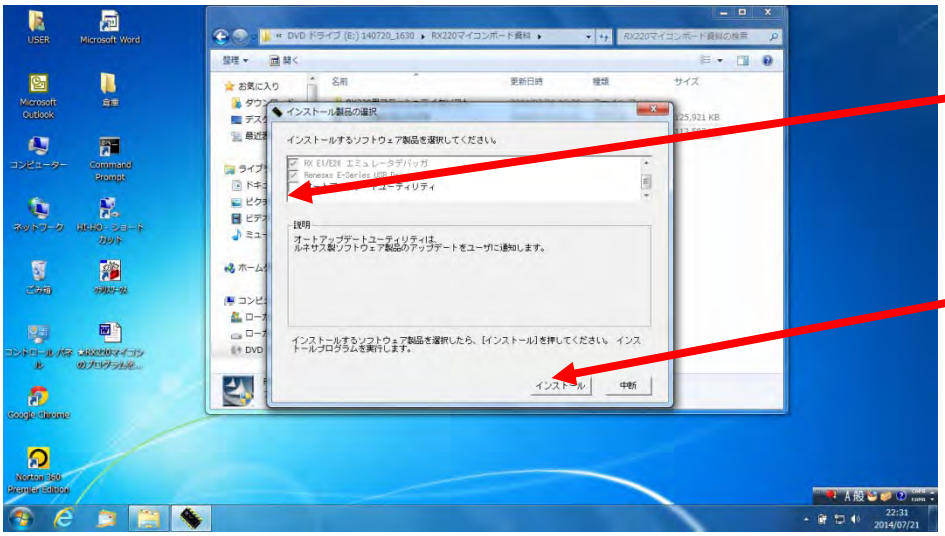
「RX220 用 E1 エミュレータソフト」を、
マウス左クリックする。



ダウンロードが開始される。

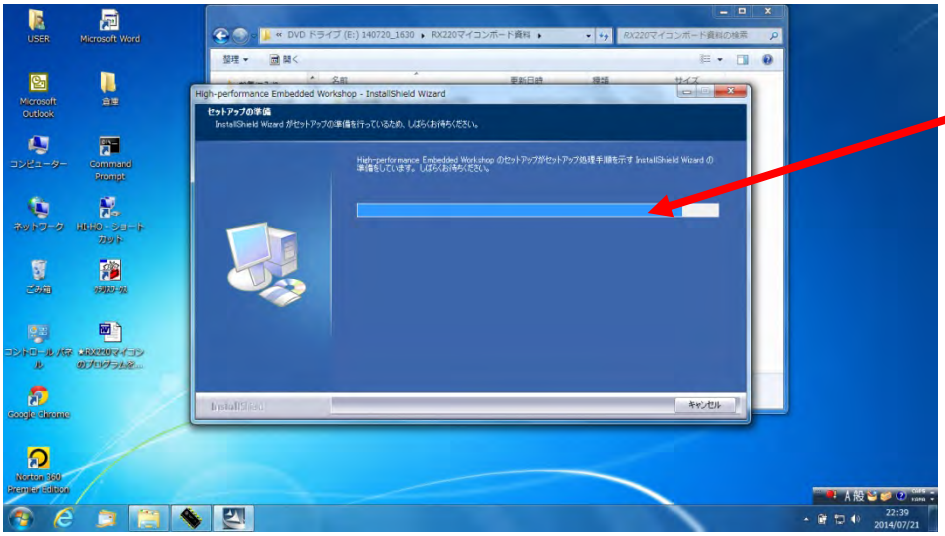


「標準インストール（推奨）」を
マウス左クリックする。

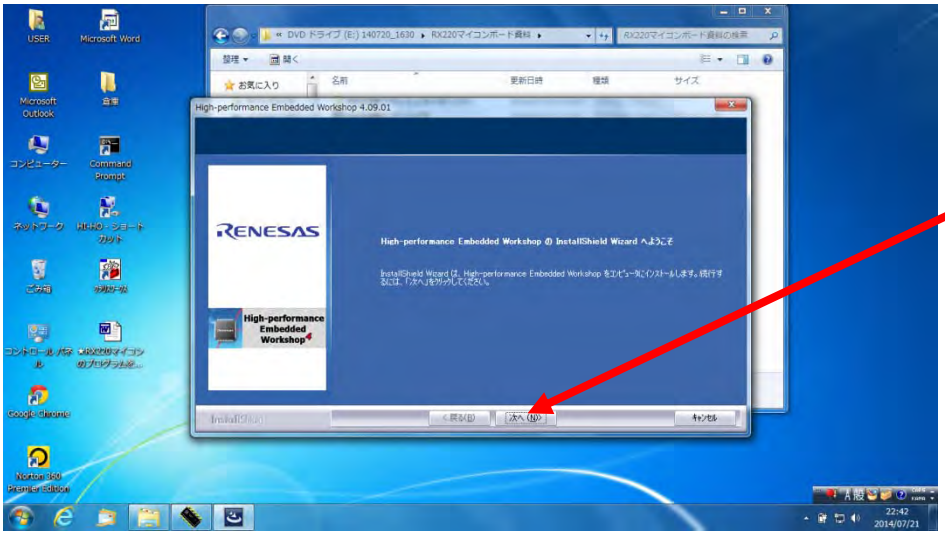


「オートアップデートユーティリティ」
のチェックマークを外す。

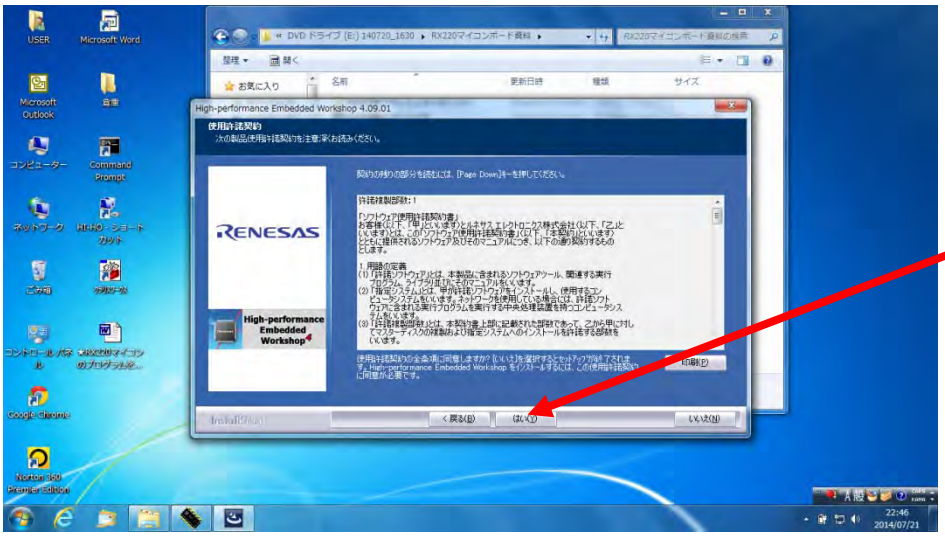
「インストール」を
マウス左クリックする。



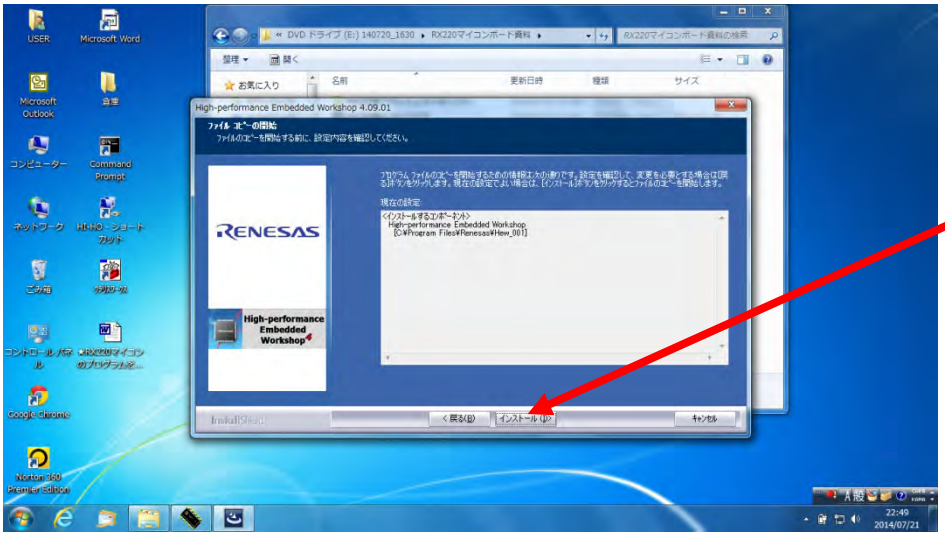
「インストール」が開始される。



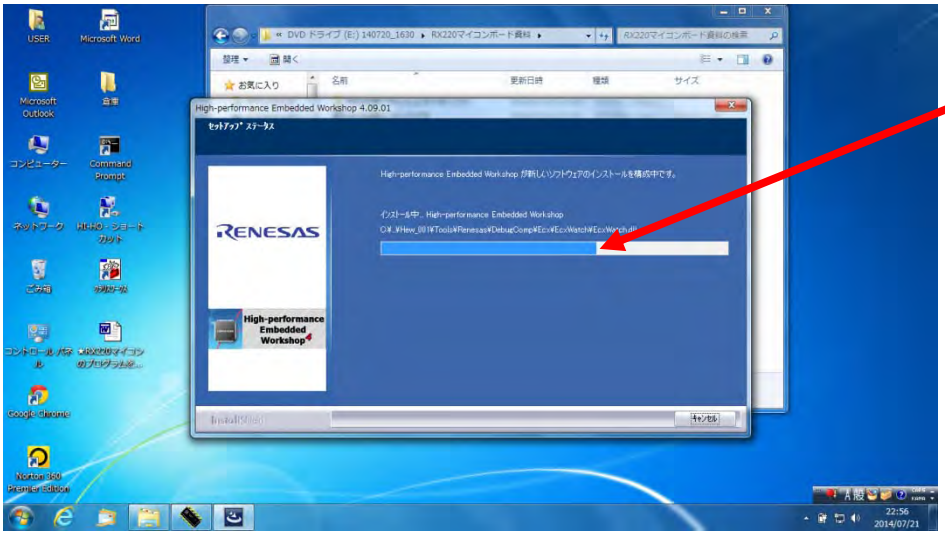
「次へ（N）＞」を
マウス左クリックする。



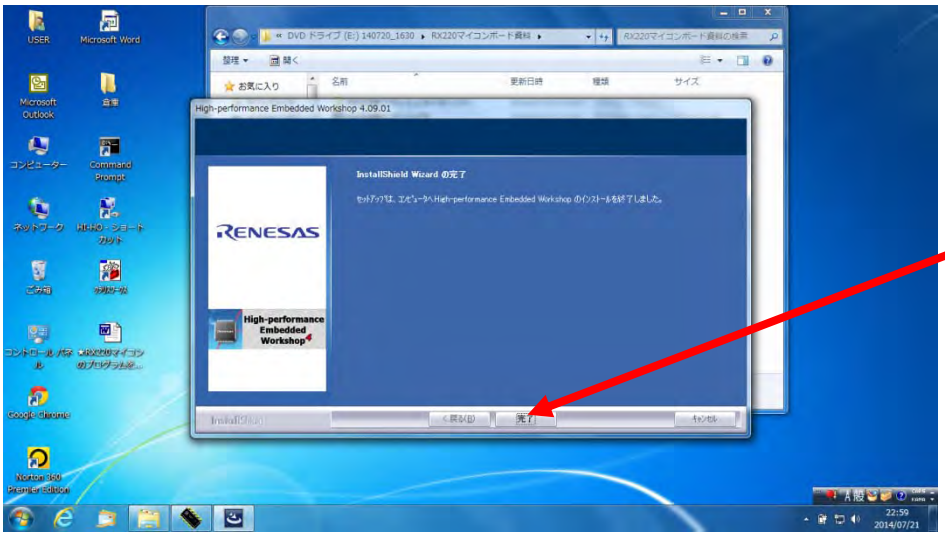
「はい（Y）」を、
マウス左クリックする。



「インストール（Ｉ）＞」を、
マウス左クリックする。

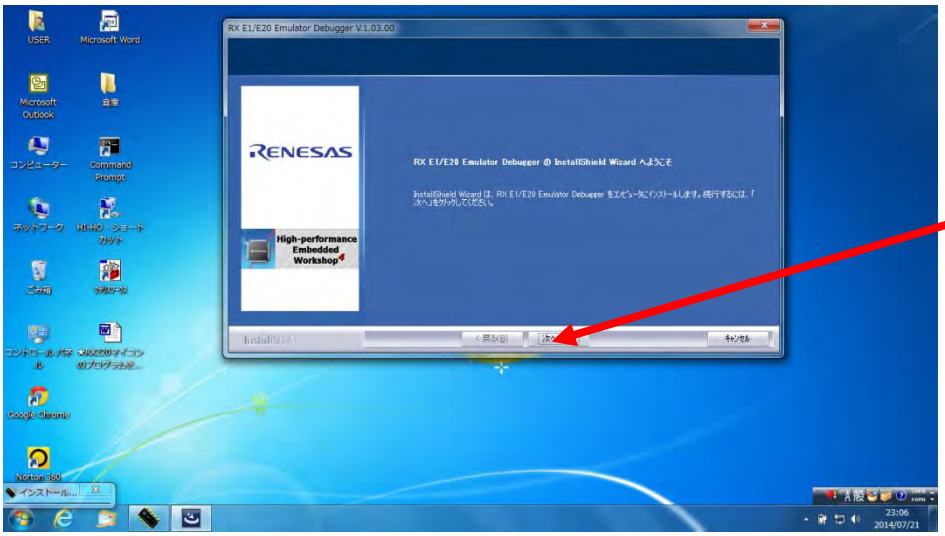


ダウンロードを開始する。

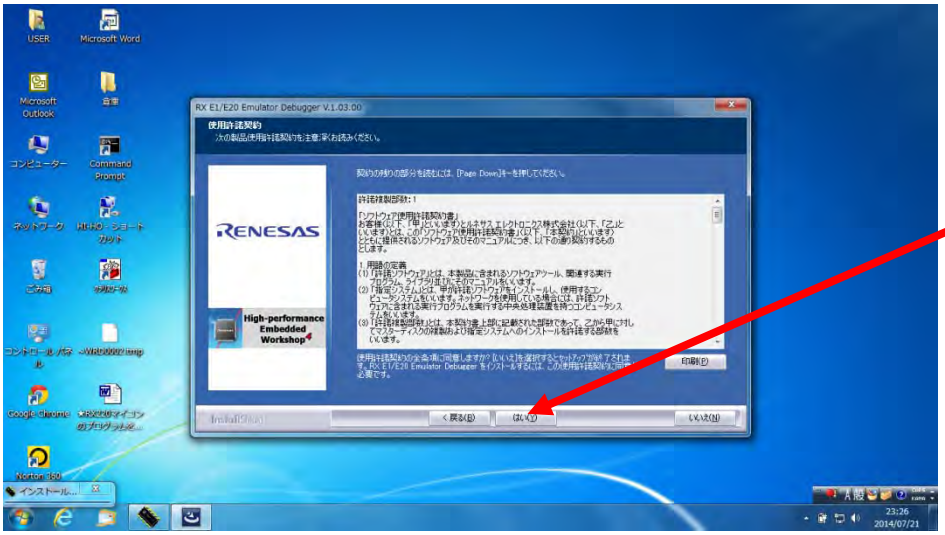


「完了」を
マウス左クリックする。

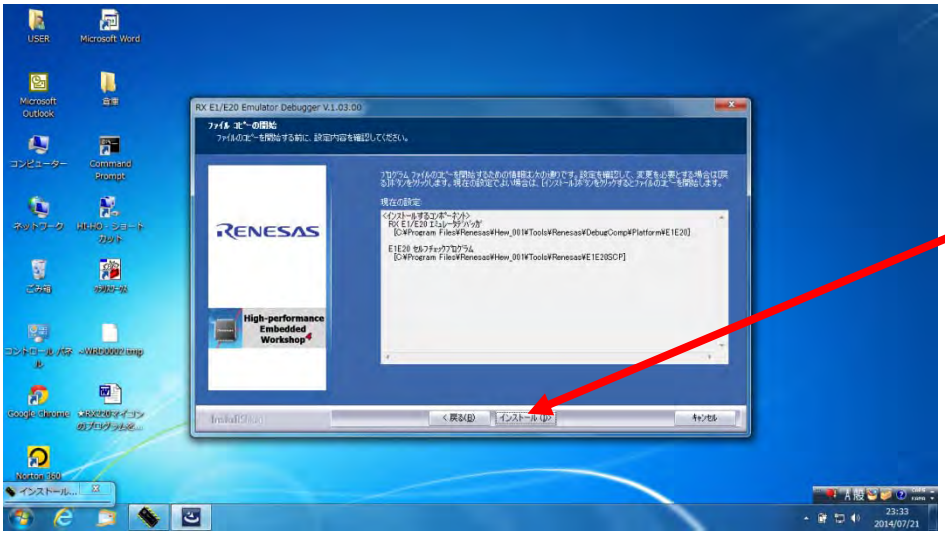
しばらくすると、「RX E1/E20 Emulator Debugger」が立ち上がる。



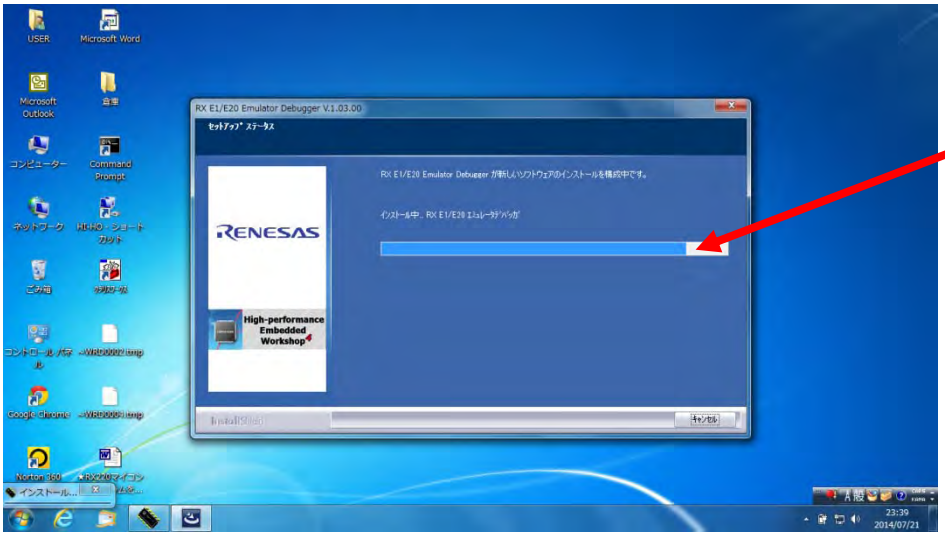
「次へ（N)」を
マウス左クリックする。



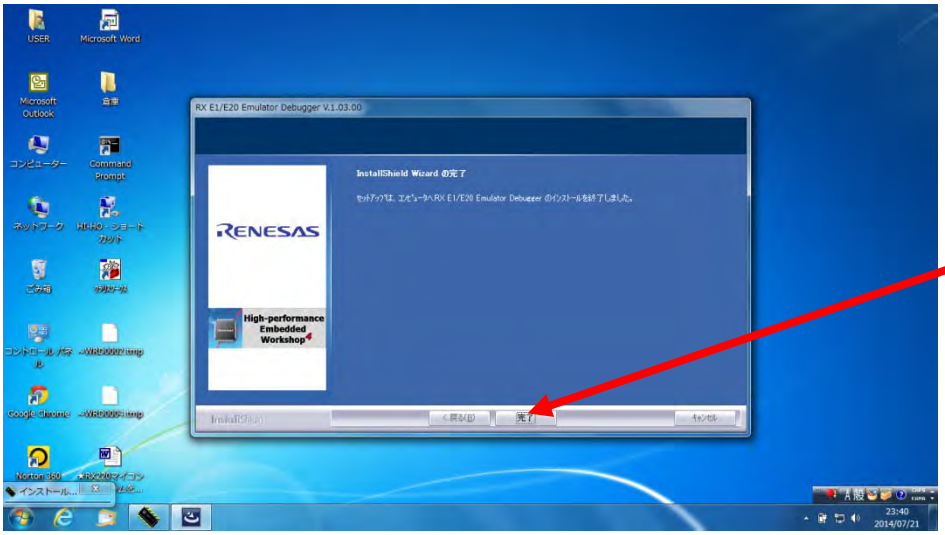
「はい（Y)」を
マウス左クリックする。



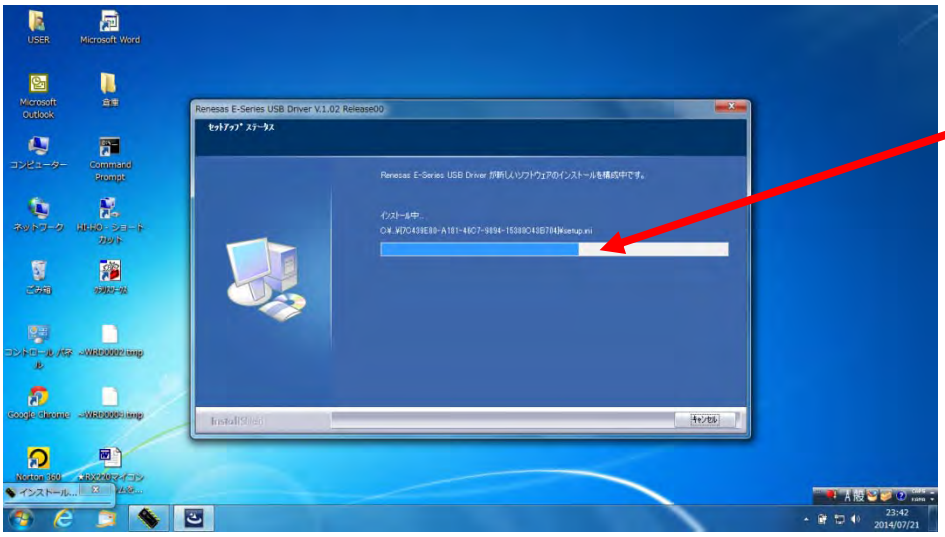
「インストール（I）＞」を
マウス左クリックする。



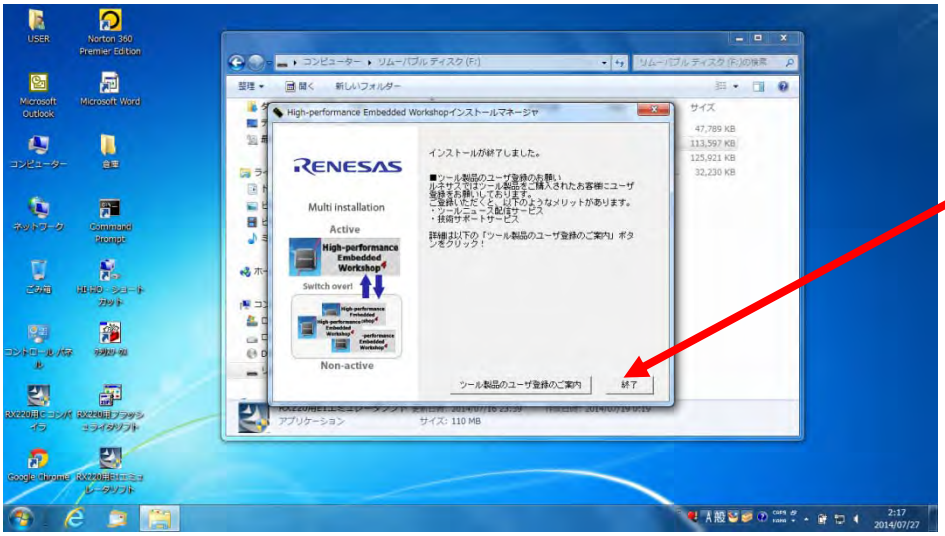
しばらくすると、
ダウンロードが開始される。



「完了」を
マウス左クリックする。

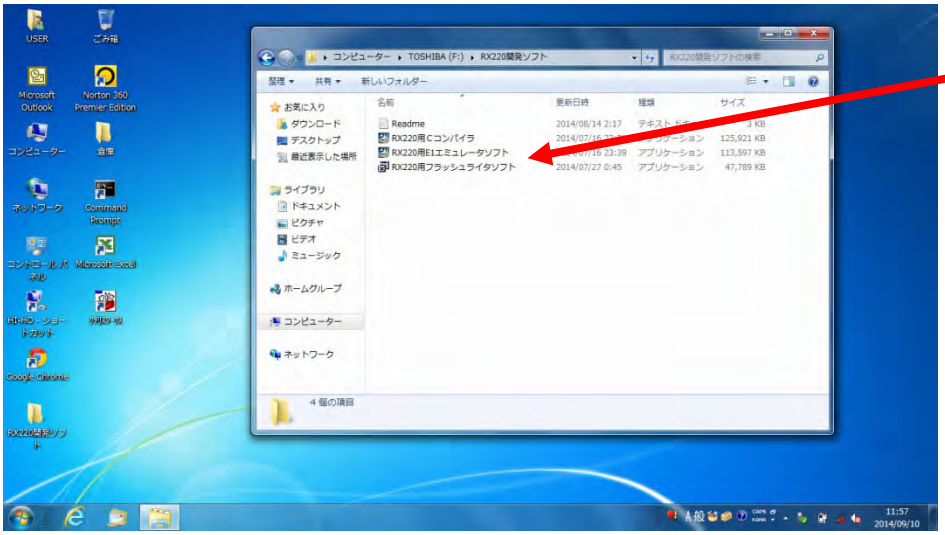


ダウンロードを開始する



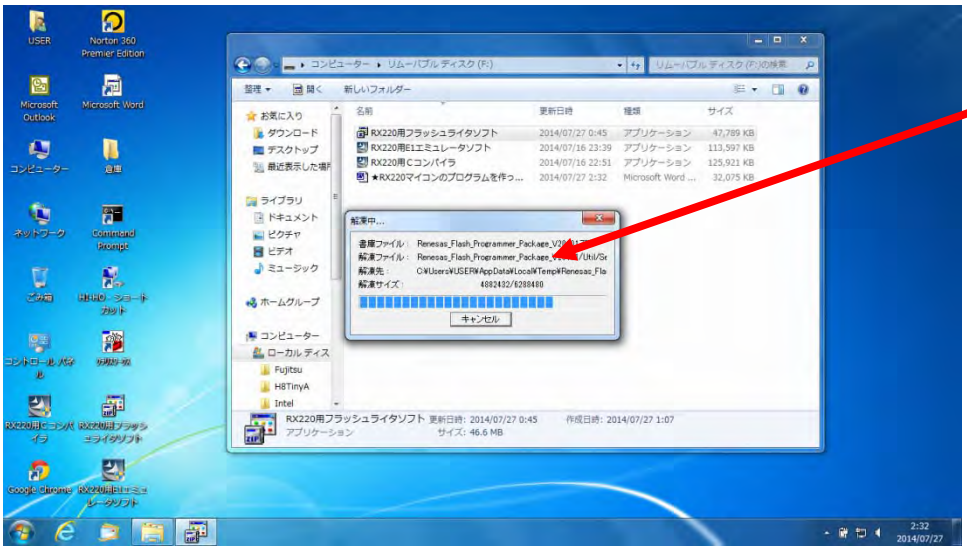
「終了」を
マウス左クリックする。

次に、
3、「RX220 用フラッシュライタソフト」のダウンロードを行う。

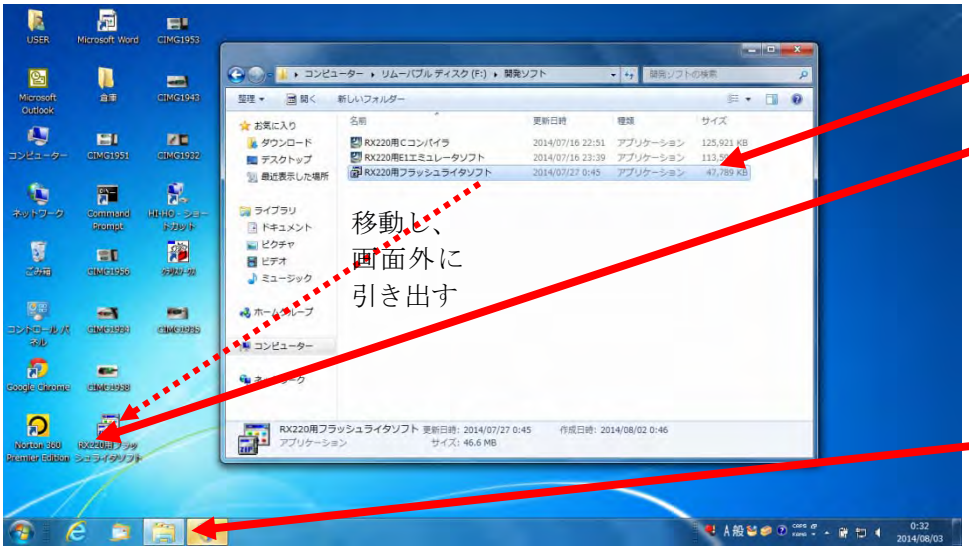


「RX220 用フラッシュライタ」を
マウス左クリックする。

（圧縮ソフトなので、フリーの
解凍ソフトを使いソフトの解凍
作業をおこなってくださ）

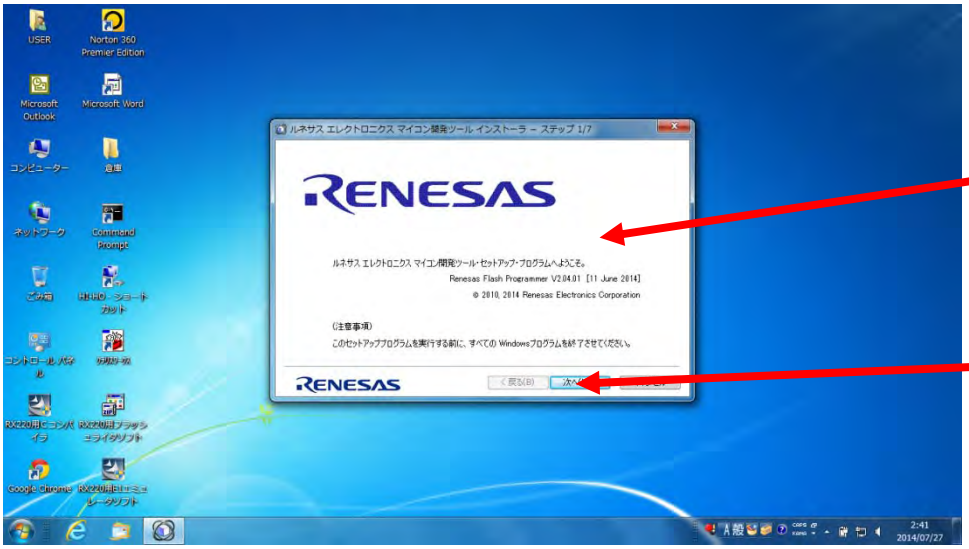


圧縮ソフトなので、
一般的なフリーの解凍ソフト
を使い、「解凍」する。



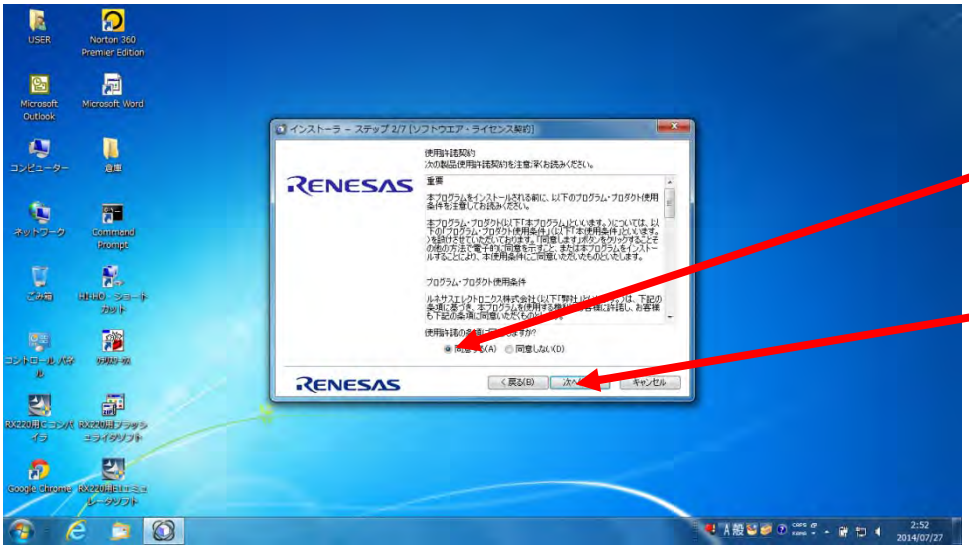
「RX220 用フラッシュライタソフト」を
マウス左クリックした状態で
Windows 画面上に引き出す。

ここをマウス左クリックする。



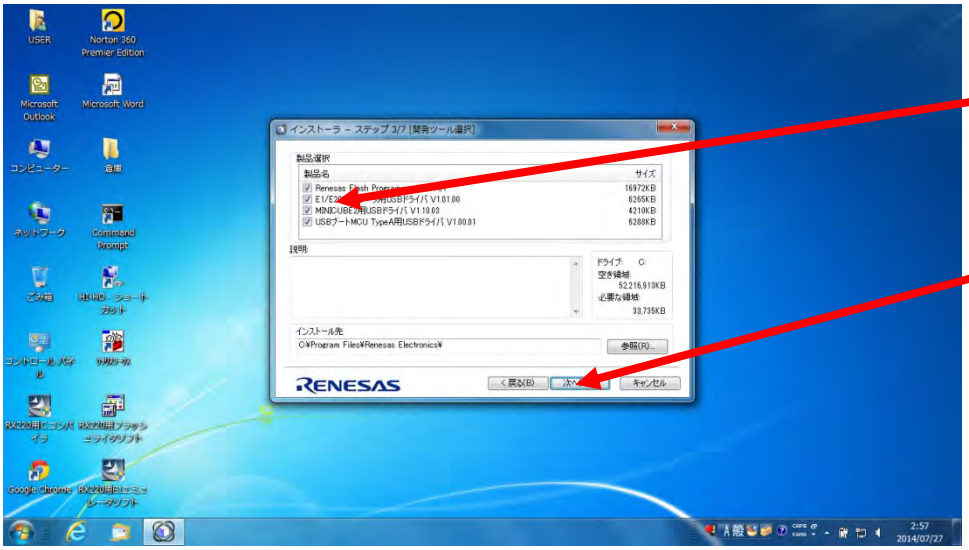
解凍された
「RX220 用
フラッシュライタソフト」

「次へ」を
マウス左クリックする。



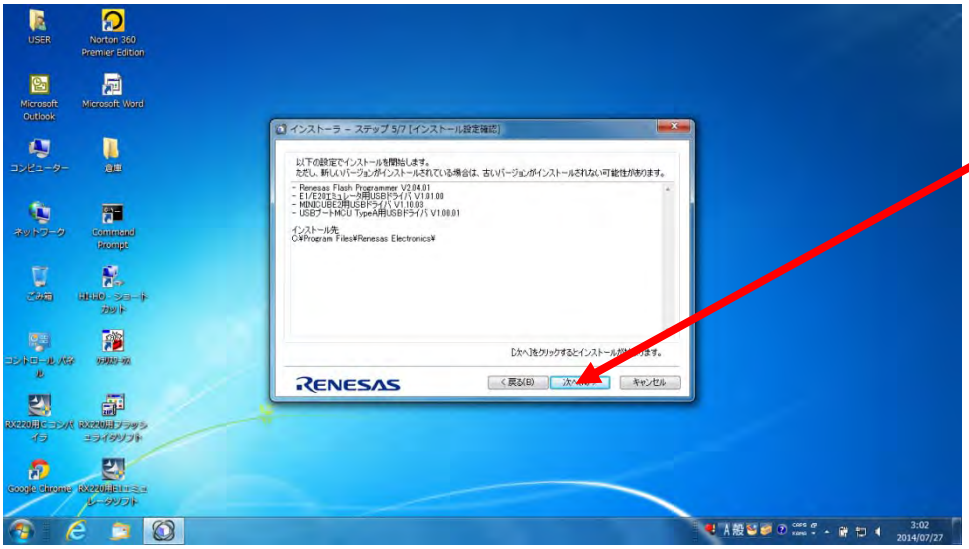
「同意する」に、
チェックマークを付ける。

「次へ」を
マウス左クリックする。

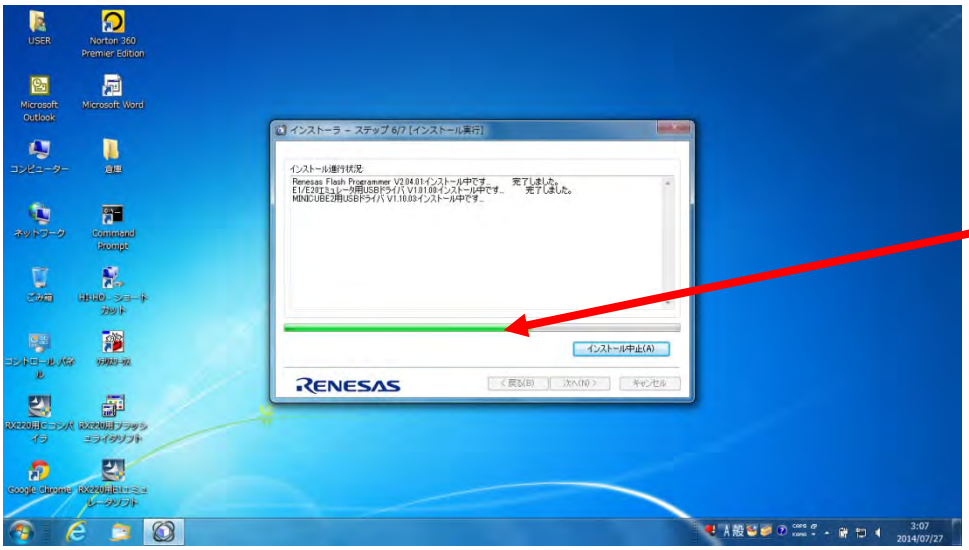


全てに、
チェックマークを付ける

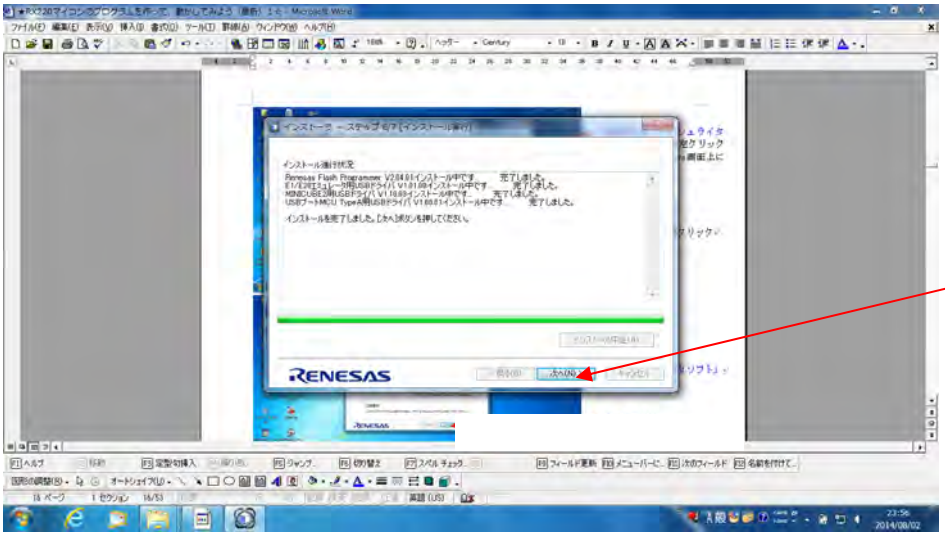
「次へ」を
マウス左クリックする。



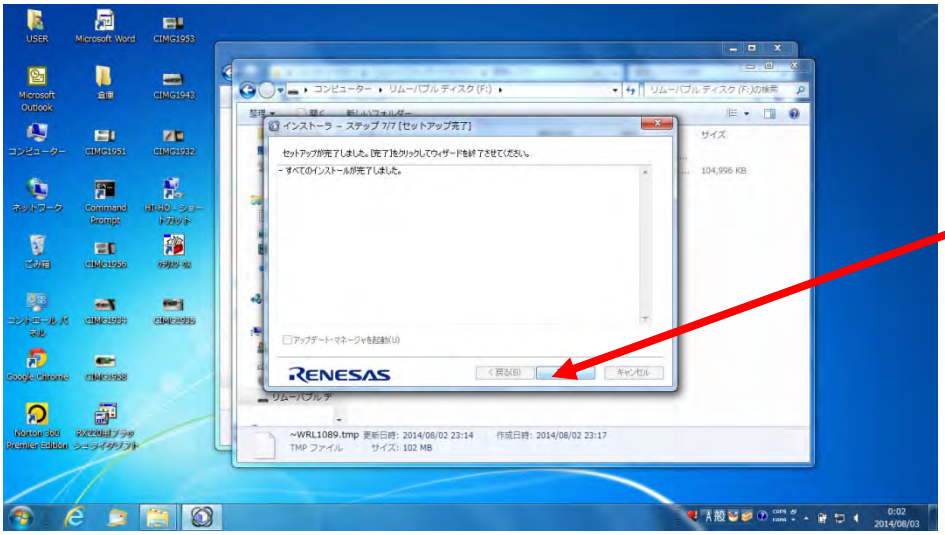
「次へ」を
マウス左クリックする。



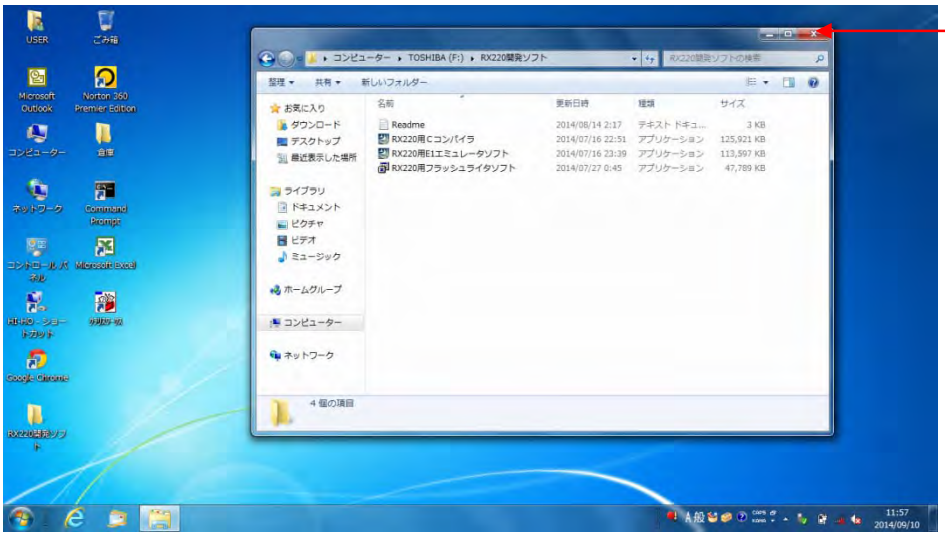
インストールの開始



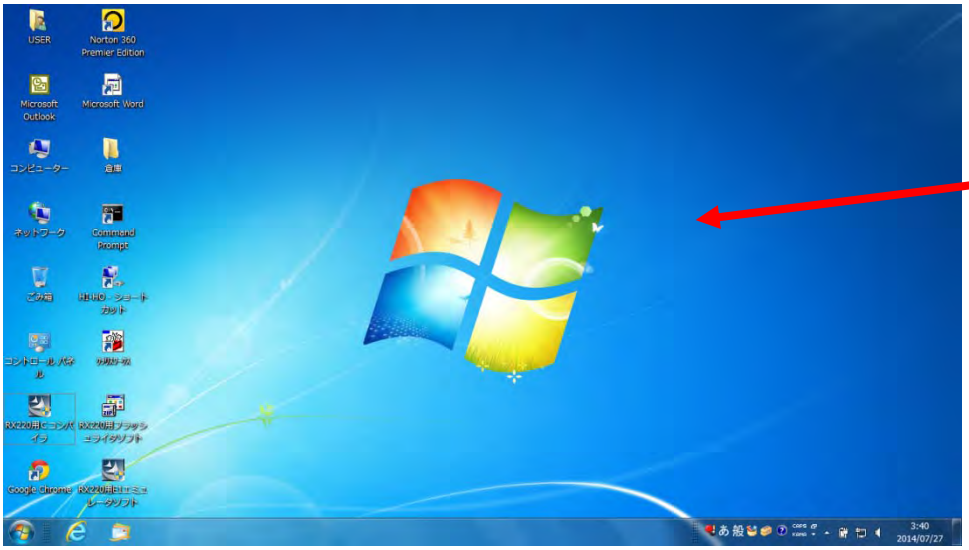
「次へ（N）>」を、
マウス左クリックする。



「完了」をマウス左クリック
する。

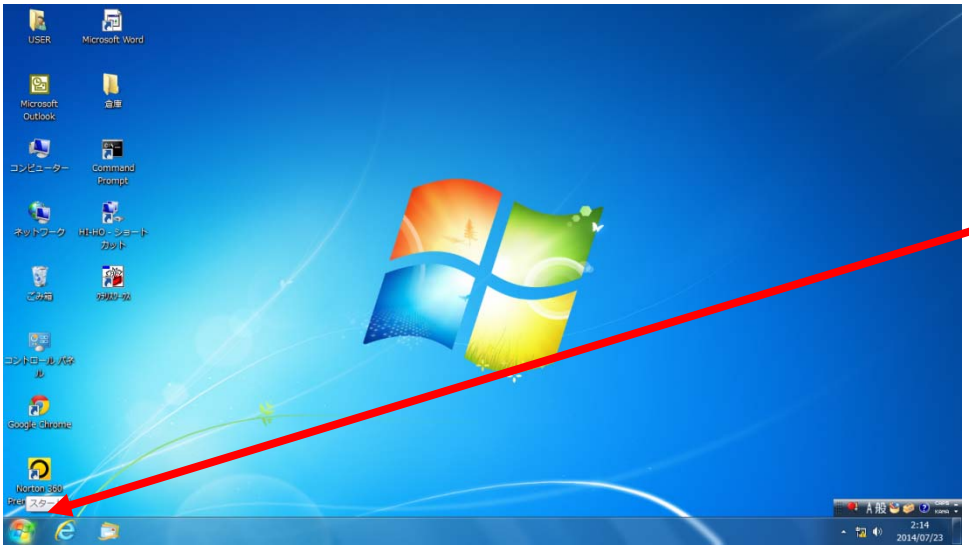


「X」をマウス左クリック
する。

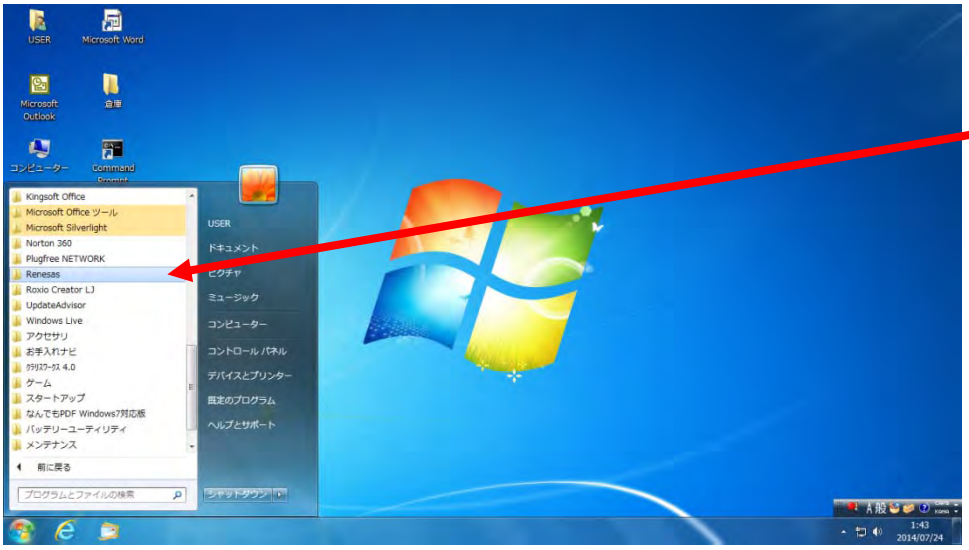


Windows 画面となる。

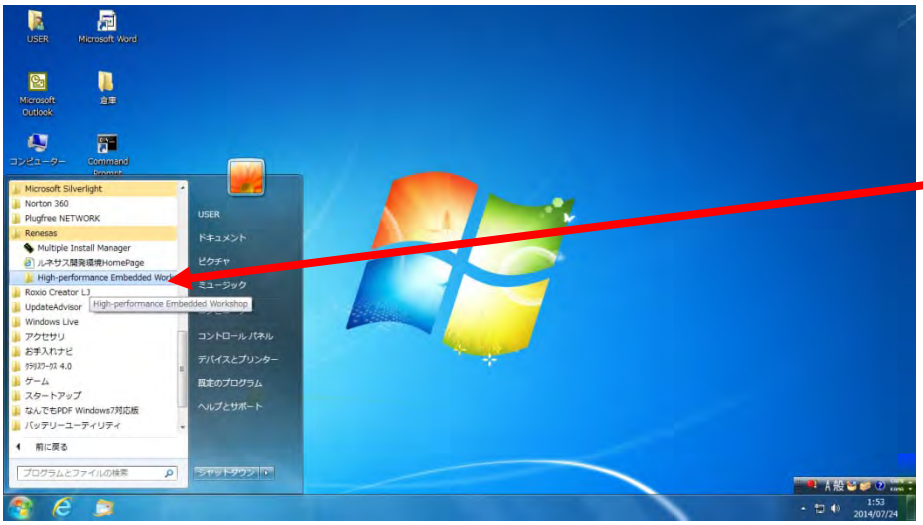
次に、
ダウンロードした「RX220 用 C コンパイラ」を、起動させてみよう。



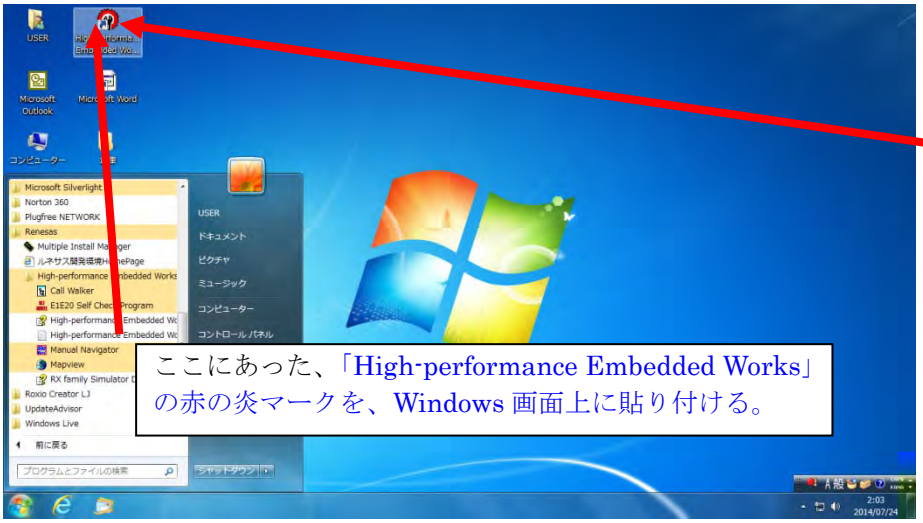
Windows 画面より、マウスの
矢印マークを当て、
「スタート」→
「すべてのプログラム」
と進む。



「Renesas」を
マウス左クリックする。



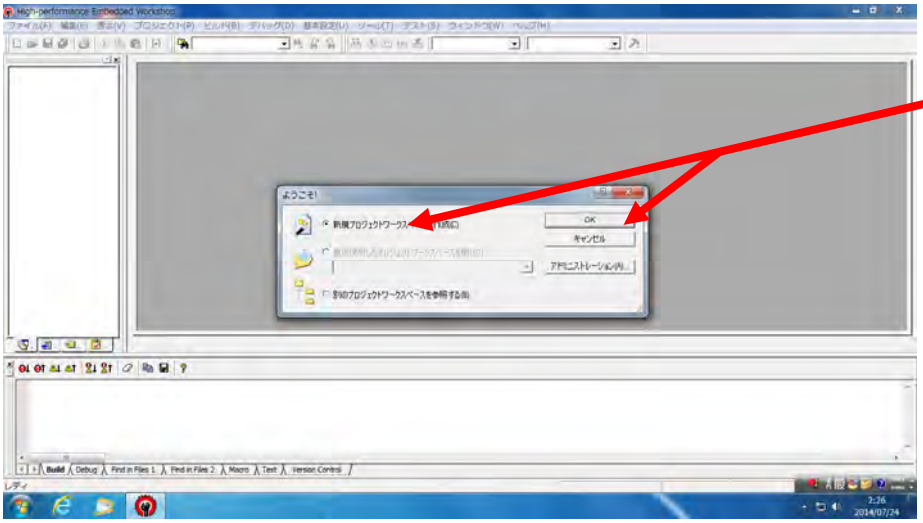
「High-performance Embedded Workshop」をマウス左クリックする。



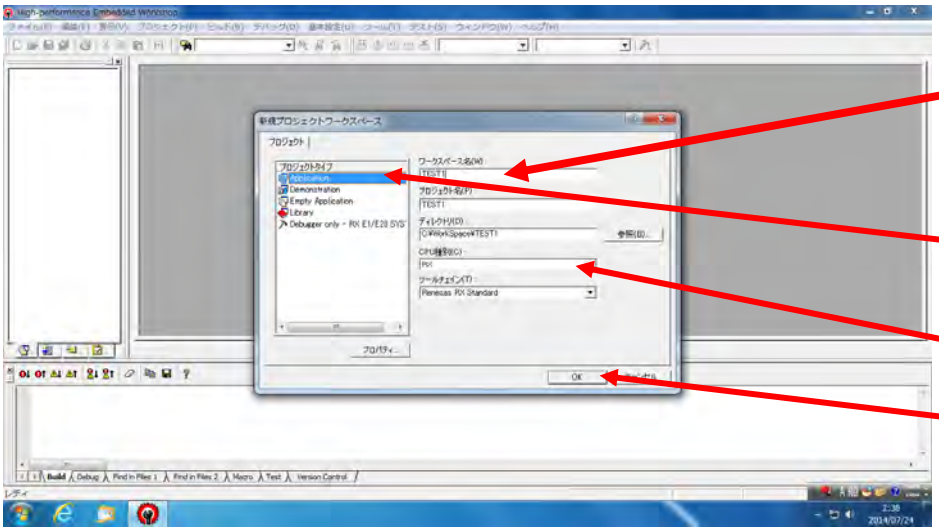
「High-performance Embedded Works」の赤の炎マークを、マウス左をクリックしたまま、Windows 画面上に導き出し、画面上に貼り付ける。

次に、

Windows 画面上の「High-performance Embedded Works」の赤の炎マークをマウス左クリックすると、下記、新規画面となる。



「新規プロジェクトワークスペースの作成（C）」にチェックマークを付け、「OK」をマウス左クリックする。

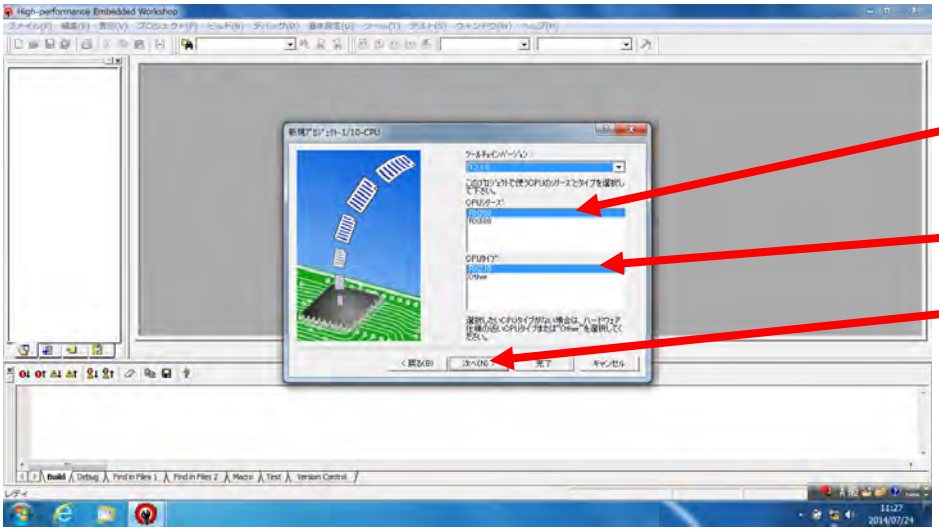


今回は「TEST1」と、入力する。

「Application」をマウス左クリックで、選択する。

「RX」を選択する。

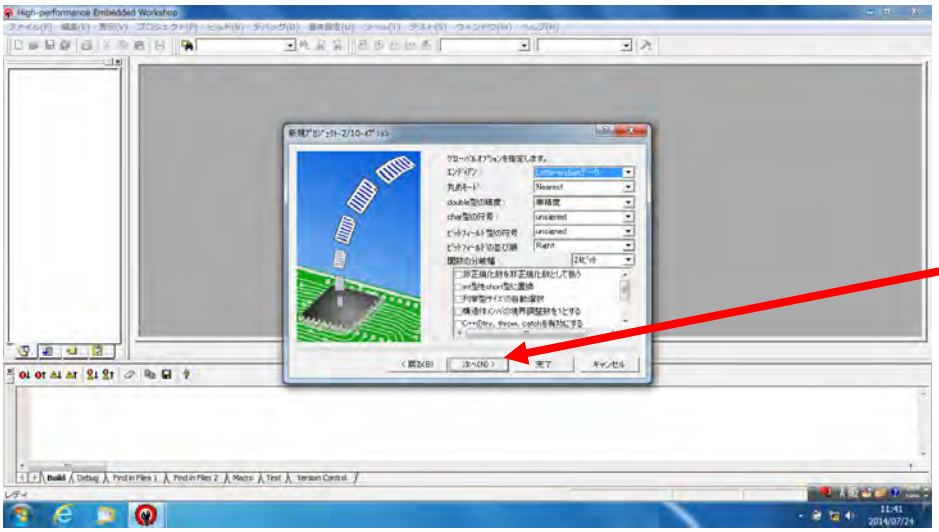
最後に、「OK」をマウス左クリックする。



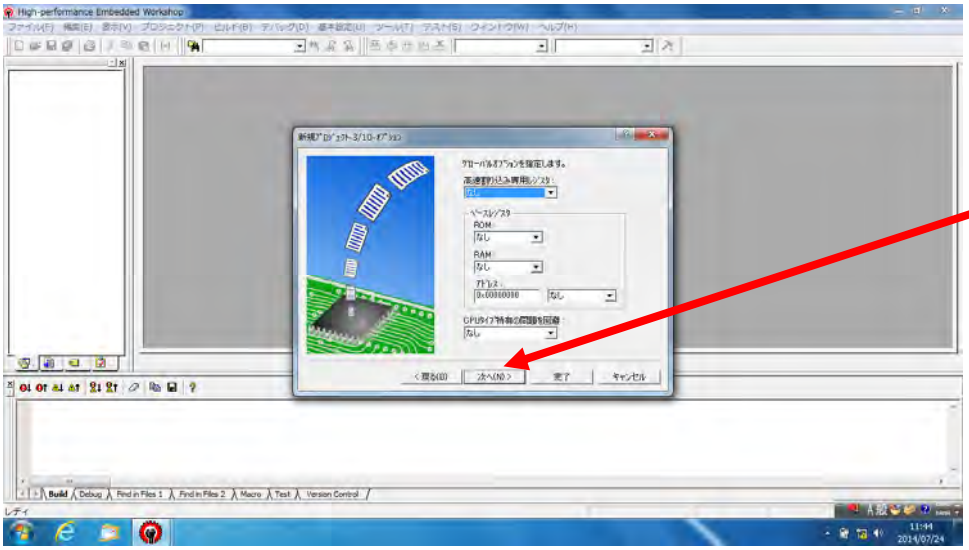
「RX200」を選択する。

「RX210」を選択する。

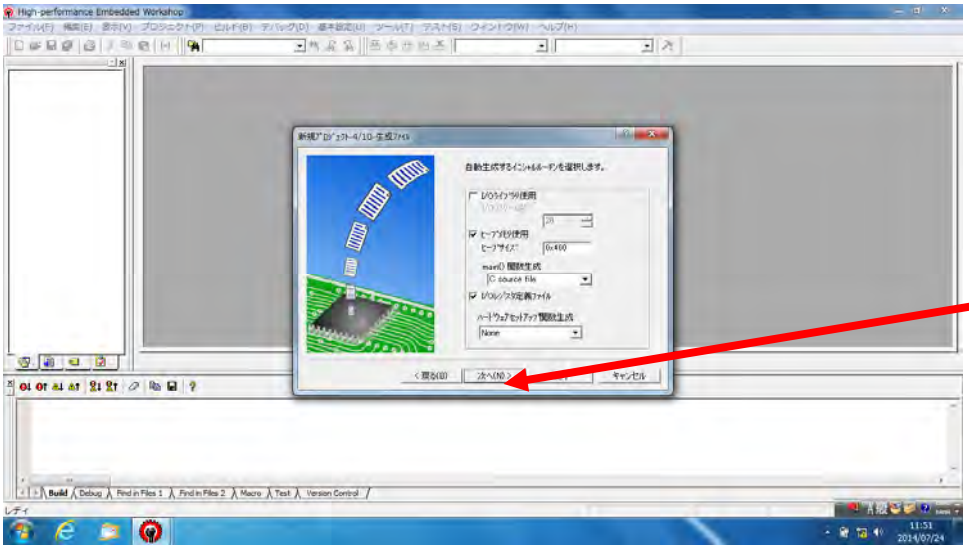
「次へ（N）>」をマウス左クリックする。



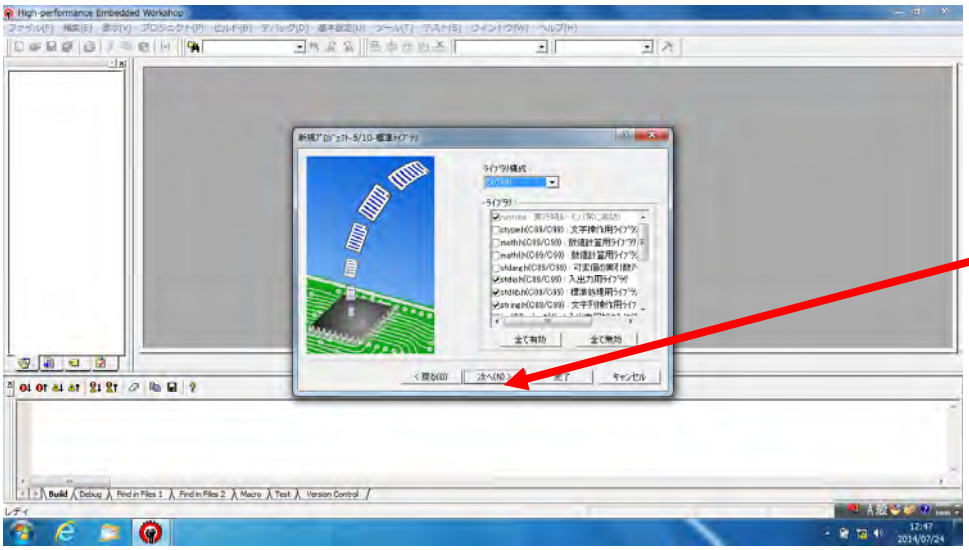
「次へ（N）>」をマウス左クリックする。



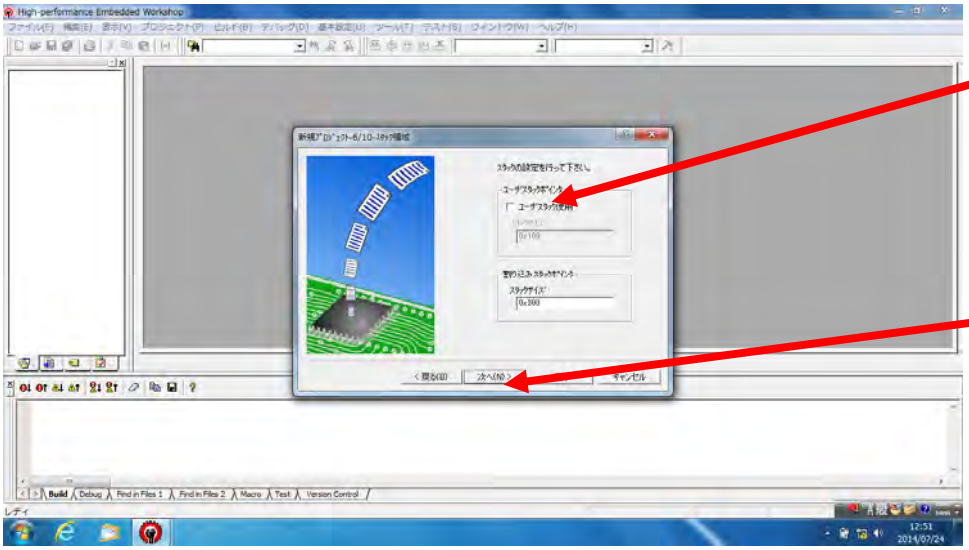
「次へ（N）＞」を
マウス左クリックする。



「次へ（N）＞」を
マウス左クリックする。

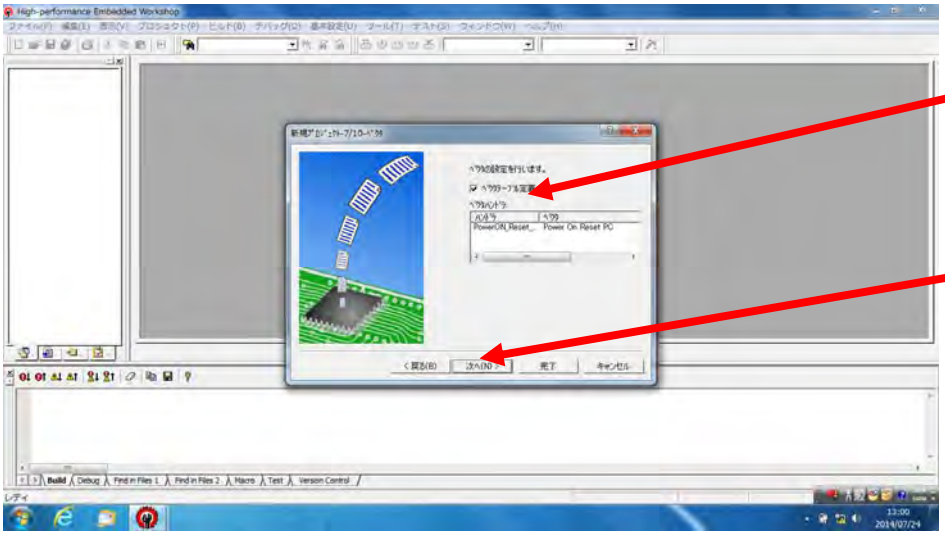


「次へ（N）＞」を
マウス左クリックする。



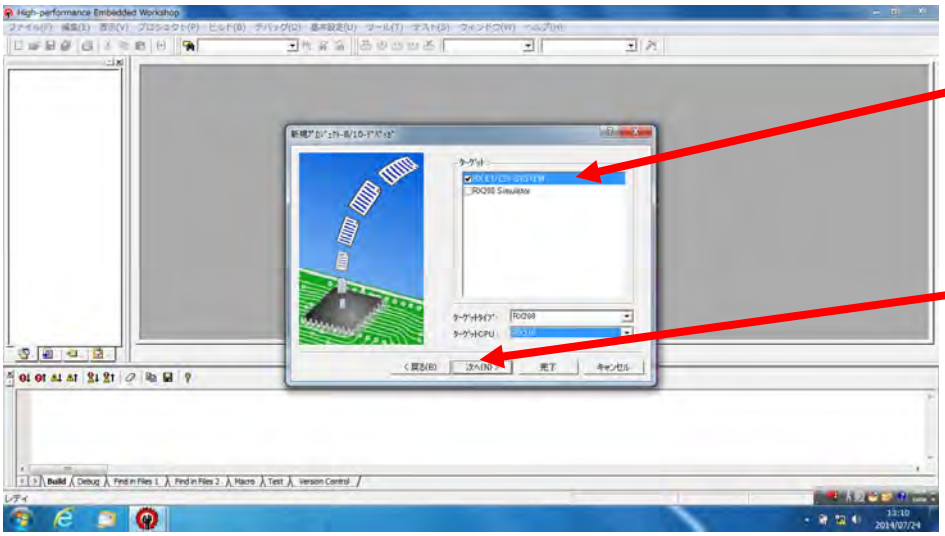
「ユーザスタック使用」の
チェックマークを外す。

「次へ（N）＞」を
マウス左クリックする。



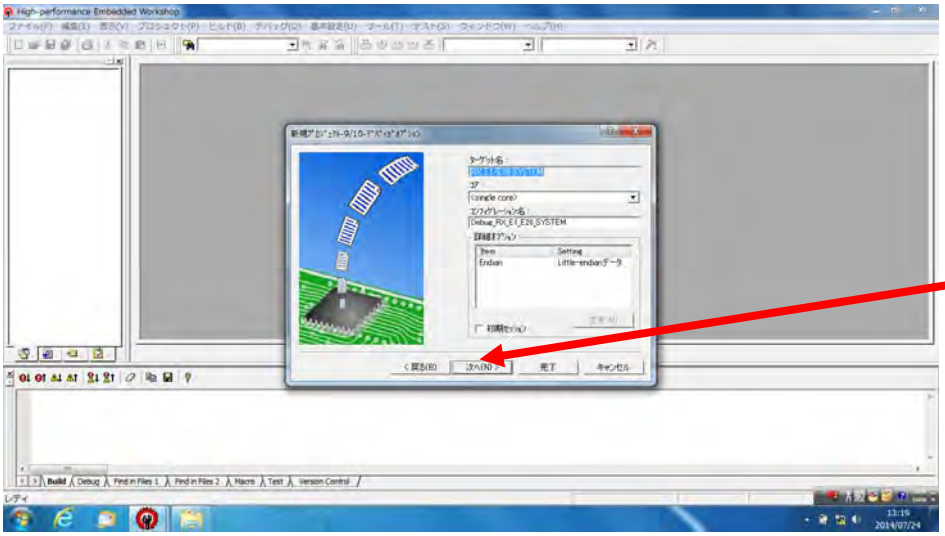
「ベクタテーブル定義」に、
チェックマークを付ける。

「次へ（N）＞」を
マウス左クリックする。

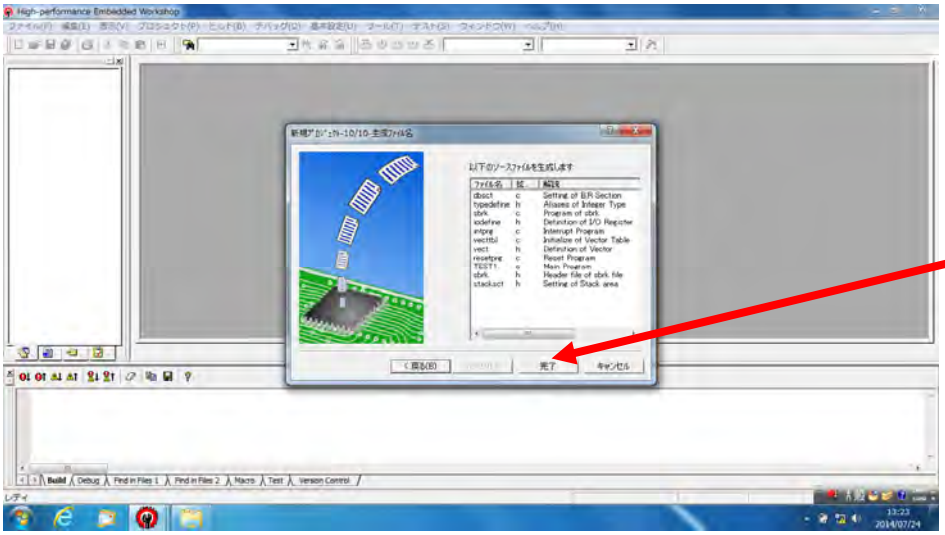


「RX E1/E20 SYSTEM」のみに、
チェックマークを付ける。

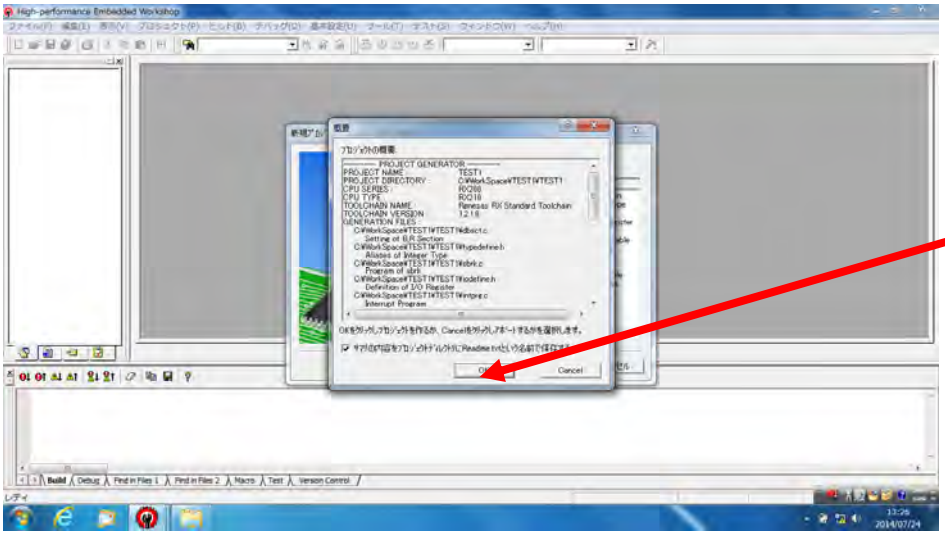
「次へ（N）>」を
マウス左クリックする。



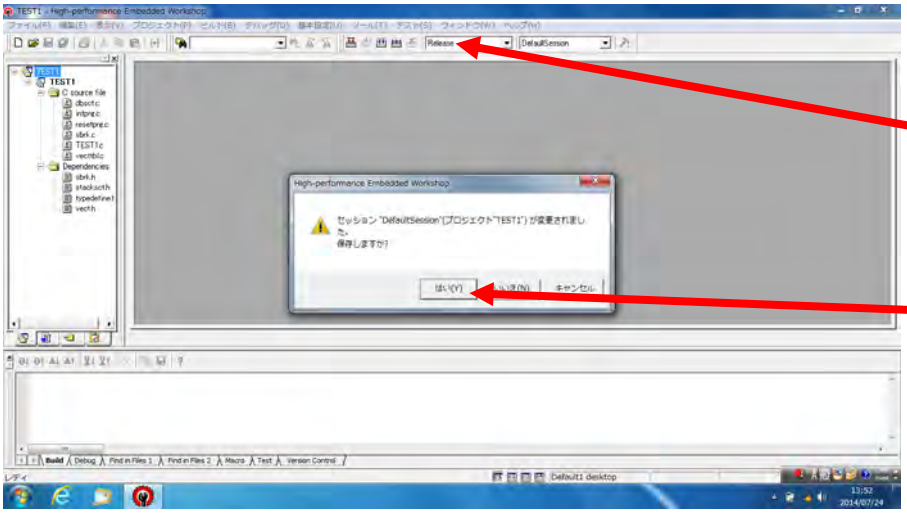
「次へ（N）>」を
マウス左クリックする。



「完了」を
マウス左クリックする。

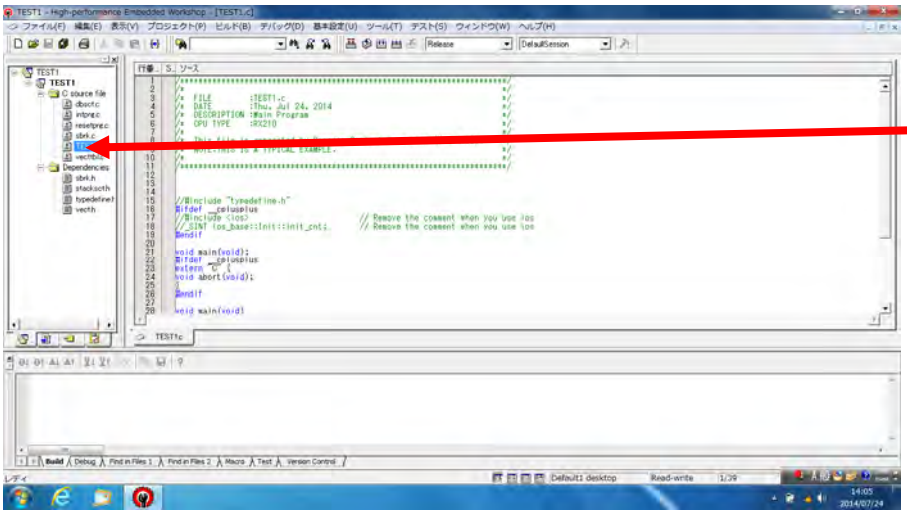


「OK」を
マウス左クリックする。

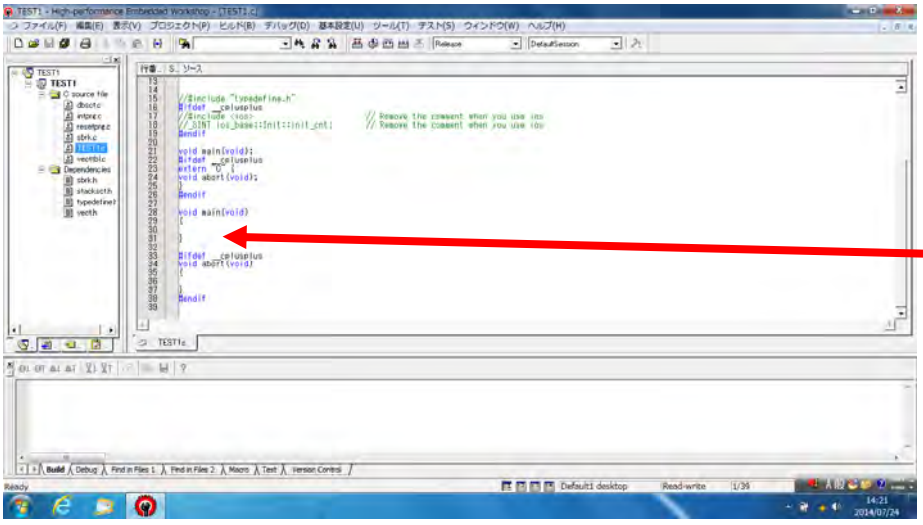


「Debbag」→「Release」に
変更する。
（Debbag 作業をしないので、
Release を選択した）

変更すると、下記画面が表示され、
「はい（Y）」を
マウス左クリックする。



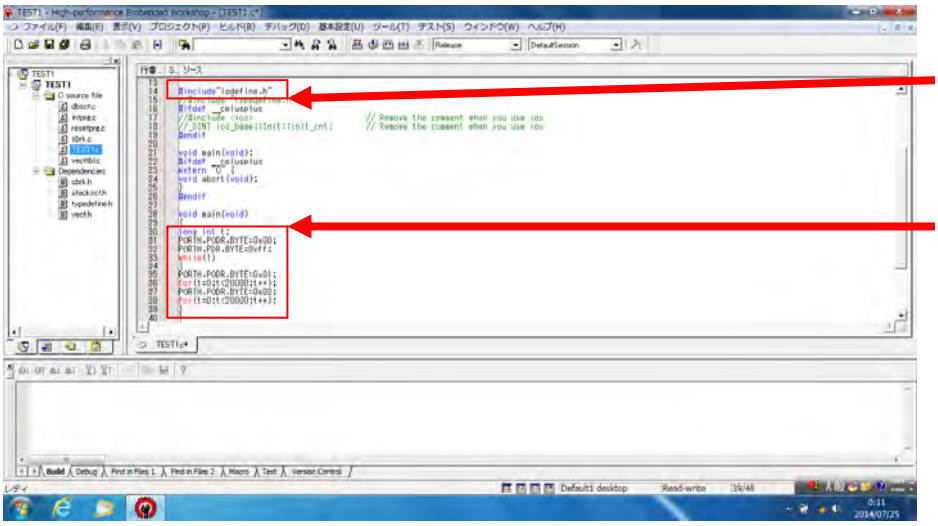
「TEST1.c」を
マウス左ダブルクリックすると、
C 言語開発環境のエディター
画面が表示される。
ユーザのソースプログラム
(main)以外は、既に自動的にプロ
グラムが作成されている。



ここの main 以下のプログラム
に、ユーザの作ったソースプログ
ラムを、書き込んでいく。

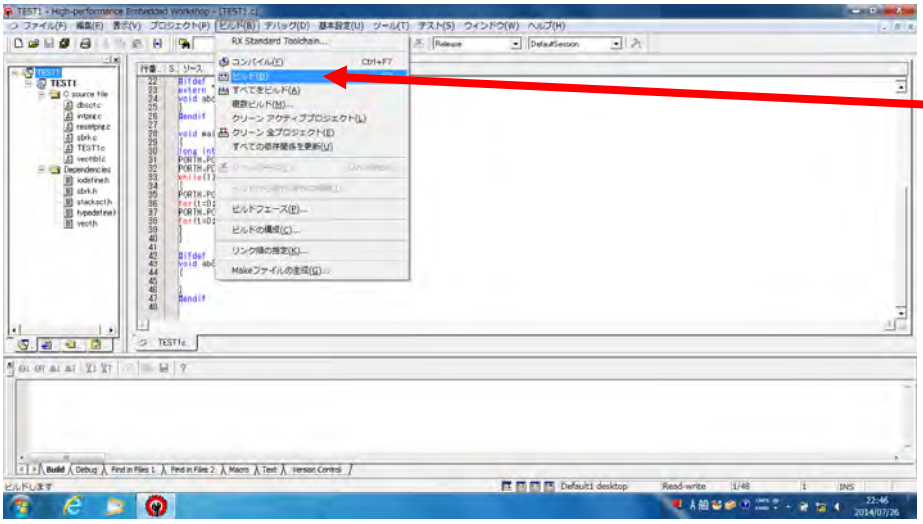
```
Void main(Void)
{

  ←　ここに、書き込んでいく。

}
#ifdef  _cplusplus
```

■LED1(赤)と LED2（緑）を、
交互に約 1 秒点滅させる C 言語
プログラムを作ってみよう。



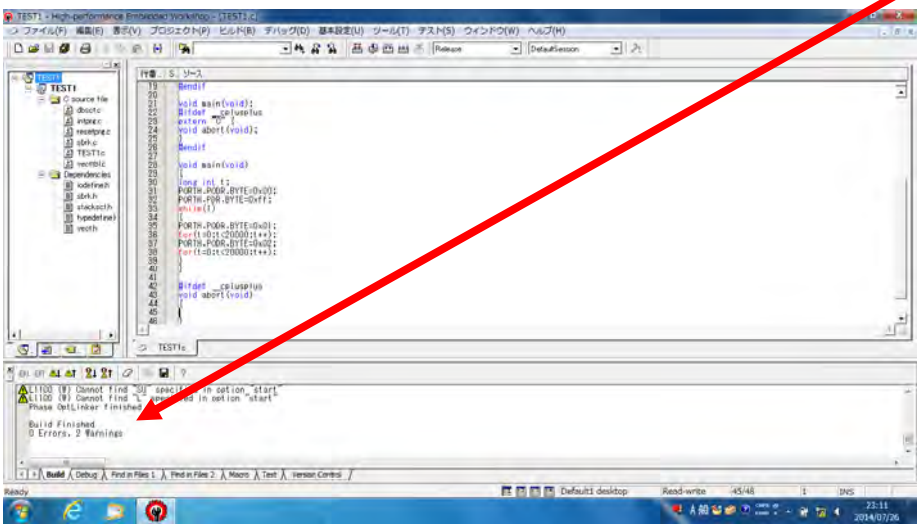
#include”iodefine.h”
を追加する。

下記を、書き込む。
long int t;
PORTH.PODR.BYTE=0x00;
PORTH.PDR.BYTE=0xff;
while(1)
{
PORTH.PODR.BYTE=0x01;
for(t=0;t<20000;t++);
PORTH.PODR.BYTE=0x02;
for(t=0;t<20000;t++);
}

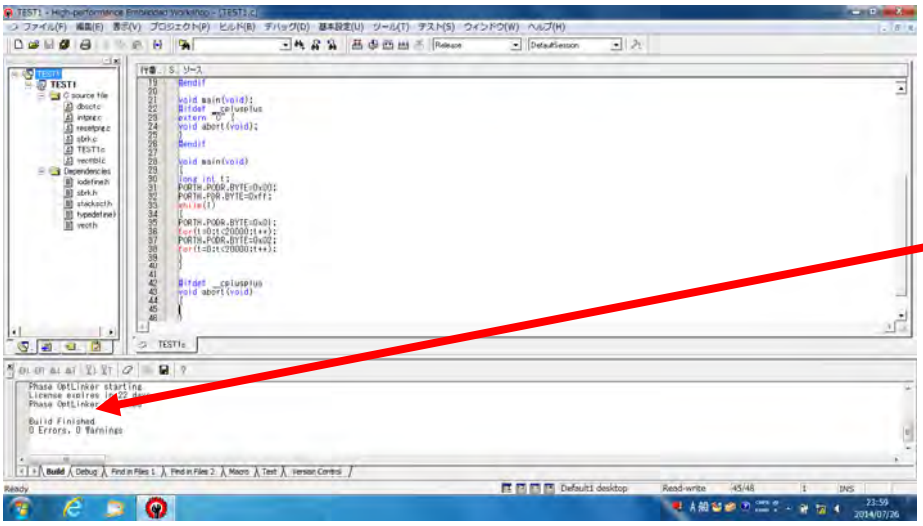


「ビルド」をマウス左クリックする。

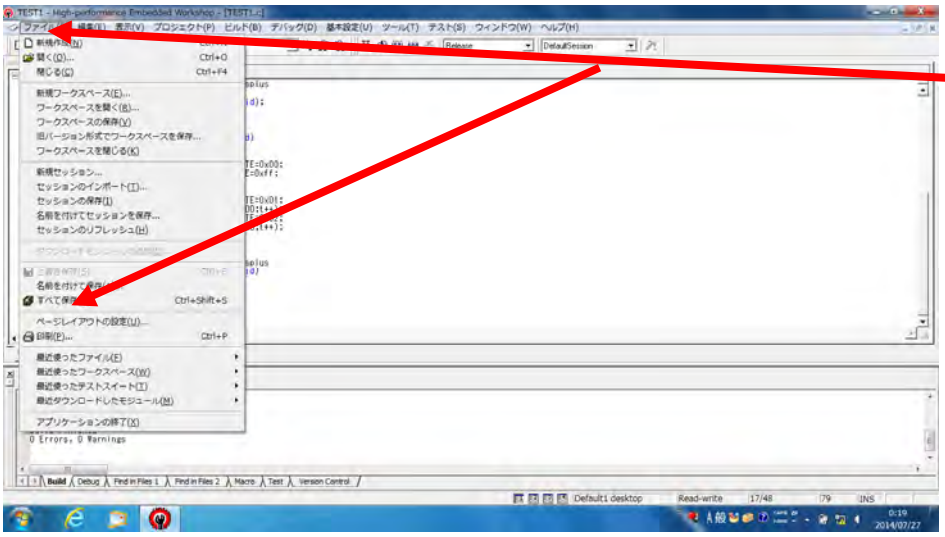
C 言語のコンパイル処理が始まる。
初めは処理時間が２～３分位
かかるが、
２回目からは、瞬時に処理が終わる。
今回は 0 Errors,2 Warnings と表示。
Warnings は、Error ではなく、
警告ですので、チェック後、問題なけ
れば、このままで良いが、表示が気に
なるようなら、
下記の手順で削除する。



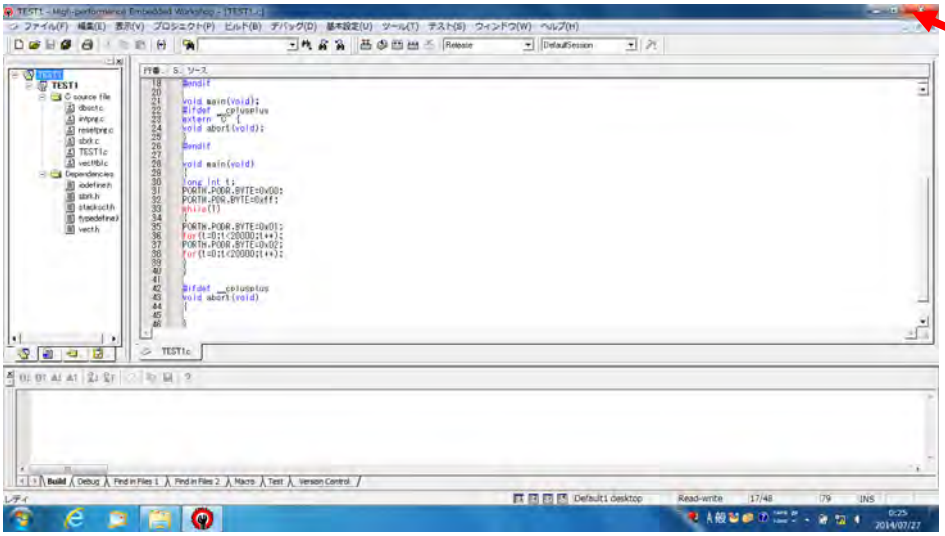
ツールバーからビルド→
”RX Standard Toolchains”を選択。
起動した”RX Standard Toolchains ダ
イアログ”で最適化リンカタブを選
択。カテゴリに”セクション”、設定項
目に”セクション”を選択。
編集ボタンを押下。”セクション設定”
ダイアログが起動。SU,L をそれぞれ
選択して削除ボタンを押下。
最後に OK ボタンを押下して
”セクション設定”ダイアログを閉じ
る。”RX Standard Toolchains”
ダイアログを”OK”ボタンを押下し
閉じる。「ビルド」を再度、実行する。



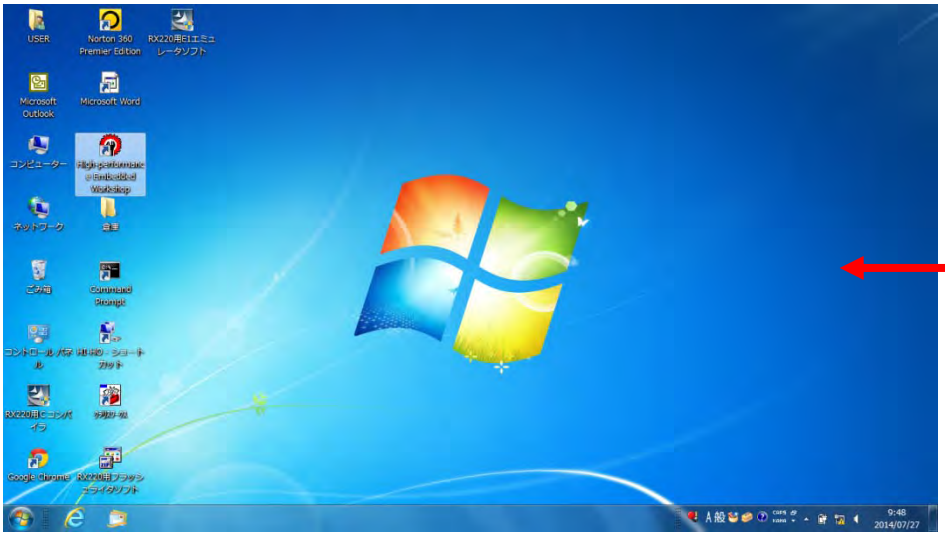
「ビルド」を再実行すると、2 Warnings が 0 Warnings となった（OK）。



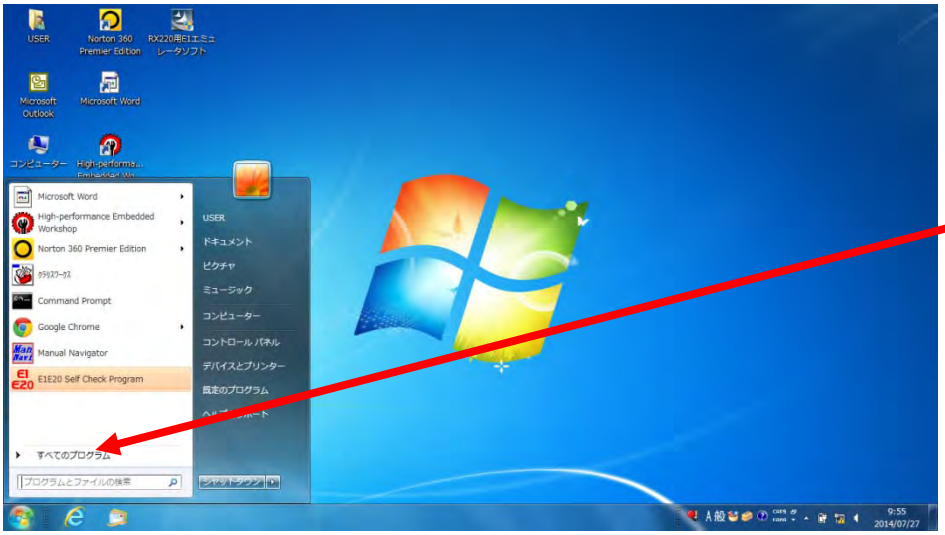
「ファイル」をマウス左クリックし、「すべて保存」をマウス左クリックする。



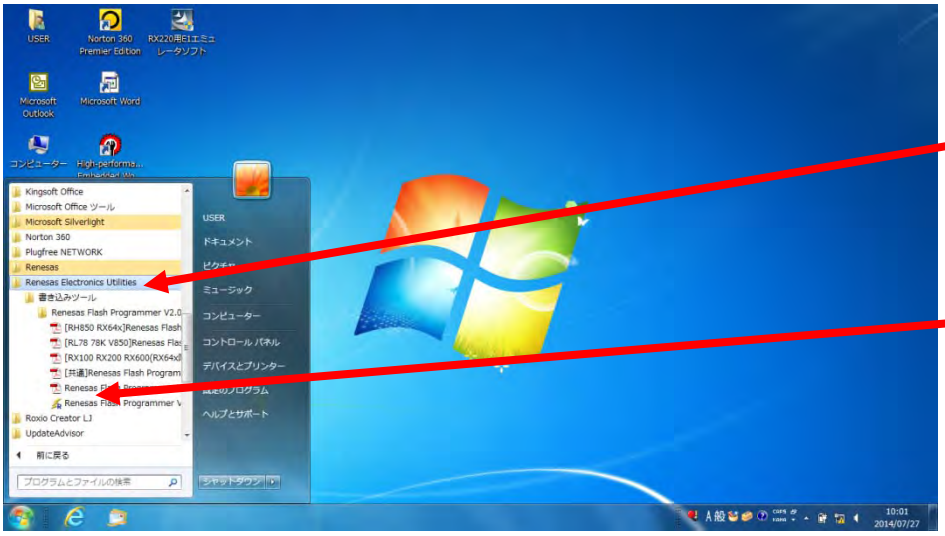
「×」を
マウス左クリックする。



Windows 画面となる。

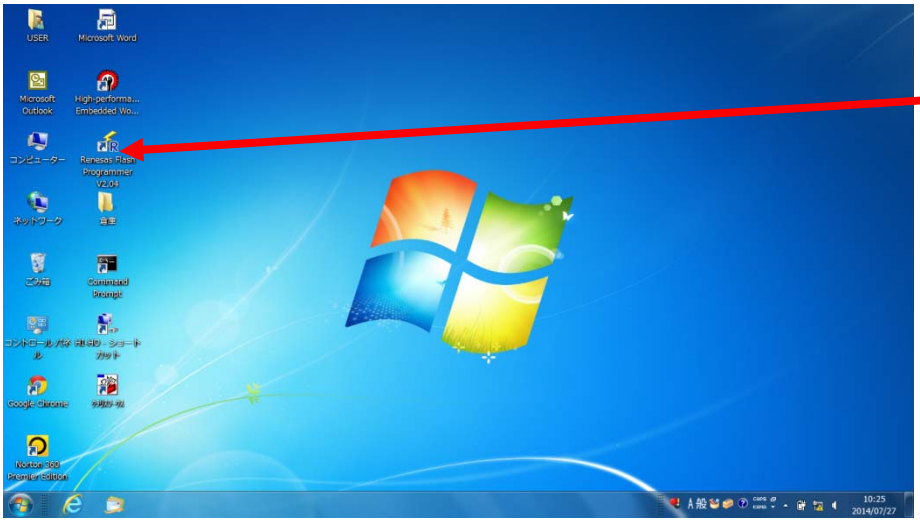


スタート→すべてのプログラム
と、マウス左クリックする。

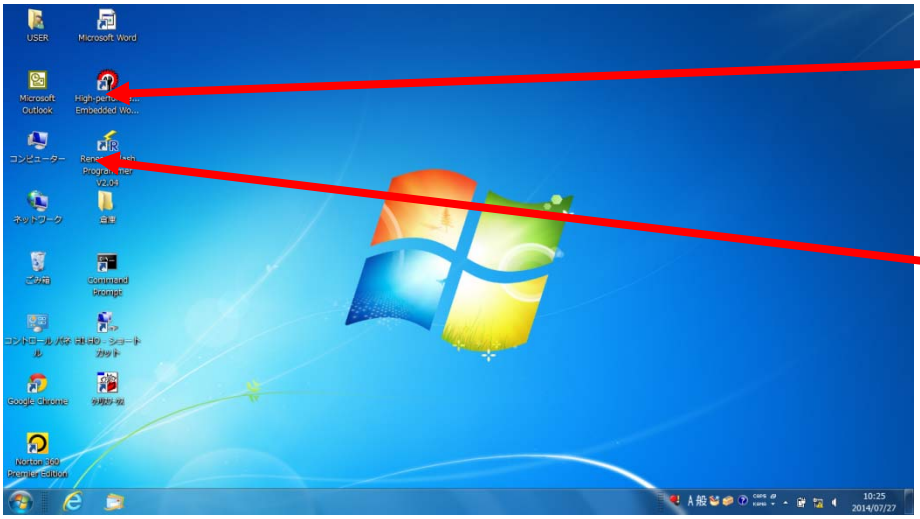


「Renesas Electronics utilities」
→書き込みツール→Renesas Flash Programmer をマウス左クリックする。

稲妻マークに **R** と記載された「Renesas Flash Programmer V2.04」アイコンを、マウス左クリックしたまま Windows 画面に引き出し、画面上に貼り付ける。



稲妻マークに **R** と記載された「Renesas Flash Programmer V2.04」アイコンを Windows 画面上に引き出し、貼り付けたところ。



まとめ：

①、C 言語のソースプログラムを作成する場合には、「High-Performance Embedded Workshop」アイコンを開き、プログラムの作成を行う。

②、作成したプログラムをマイコンに書き込む場合には、「Renesas Flash Programmer V2.04」アイコンを開き、プログラムの書き込みを行う。

（但し、E1 エミュレータでの書き込み時は、本ソフトは使用しません。B、第二章を参照してください）
今後の開発時には、①と②の２個のアイコンのみで、プログラムの開発が出来きます（但し A、第一章のシリアル接続での開発方法のみに適応します）。

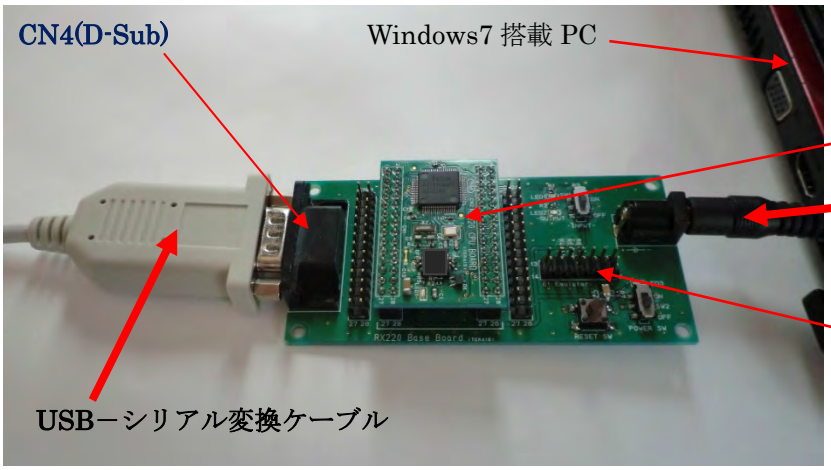
A、第１章：

■ 次に作成した「TEST1」プログラムを、「RX220　CPU ボード」に書き込んでみましょう。

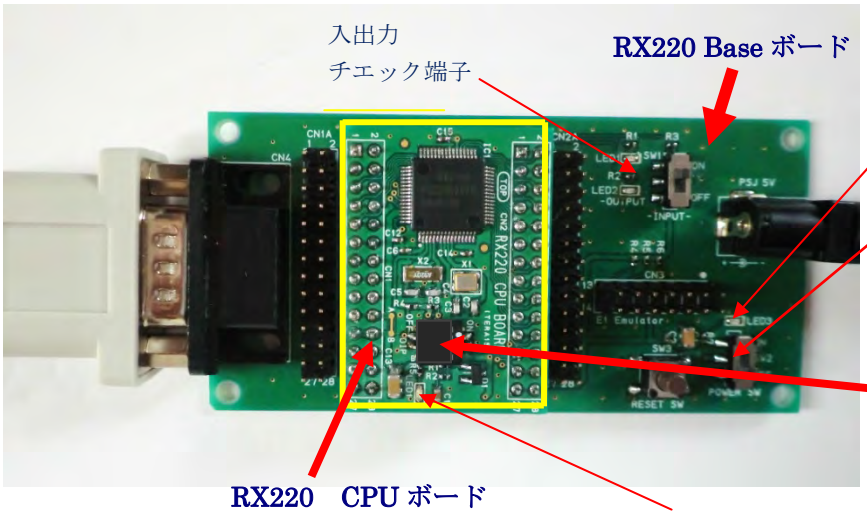
ここでは、「RX220　CPU ボード」と、「RX220　Base ボード」（どちらも、弊社で販売中）を使い、「TEST1」プログラムを書き込んでみます。

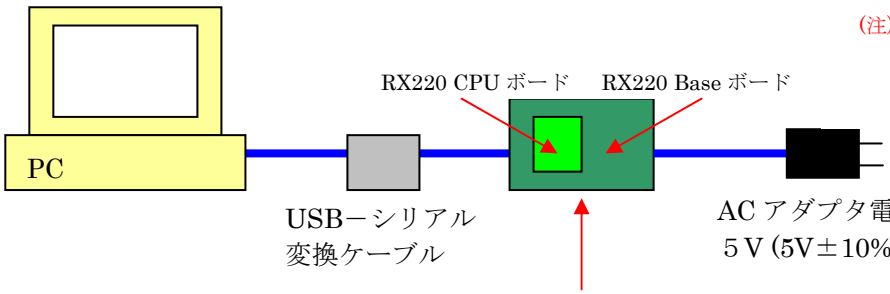
最近の PC は、シリアル端子（９PinD-Sub）が無くなり、USB に置き換わっています。
そこで、USB－シリアル変換ケーブル（通販コード M－02747）を使うと、「RX220　CPU ボード」を、ＵＳＢ端子のみの PC と、接続することが可能です。

USB－シリアル変換ケーブルとして、グレー色、延長ケーブル付（通販コード M-02747）を使用します。



■「RX220　Base ボード」の CN4（D-Sub）を介して、PC（Windows7 以降）のＵＳＢと接続します。



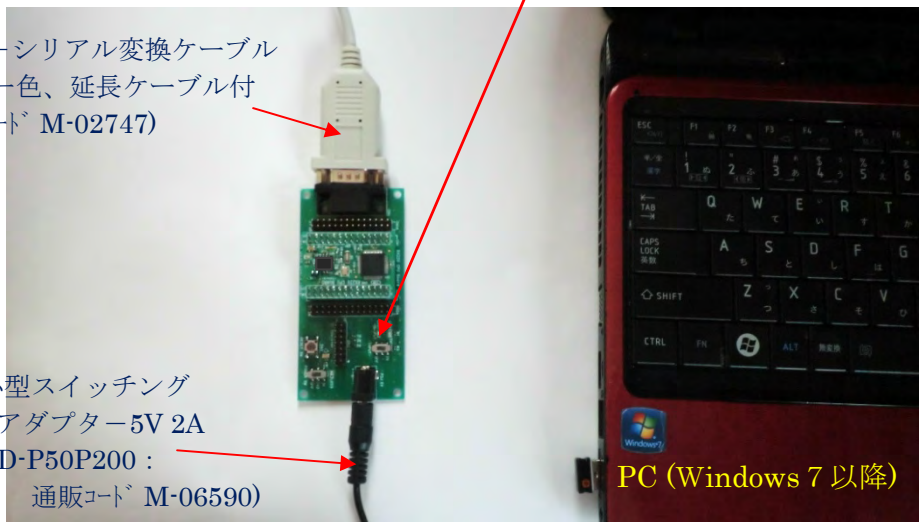


－＜USB－シリアル接続方法＞－

「RX220Base ボード」に、「RX200 CPU ボード」を、上下間違いなくセットします(逆差し注意)。

■ さて、上記の接続が済み、USB－シリアル変換ケーブルが PC のポートを認識していることを確認出来たなら、
「RX220 CPU ボード」の、「DIP」 スイッチを書き込みモード（SW No.1～2、すべて ON 側にします。
次に、「RX220 Base ボード」の電源を ON にします。LED1（赤）、LED3（赤）が点灯し、
プログラムの書き込み待ち状態となります（「ブートモード」）。
(注)：「DIP」スイッチの切り替え時は、必ず電源を OFF にして、
操作を行ってください。

簡易入出力チエック部



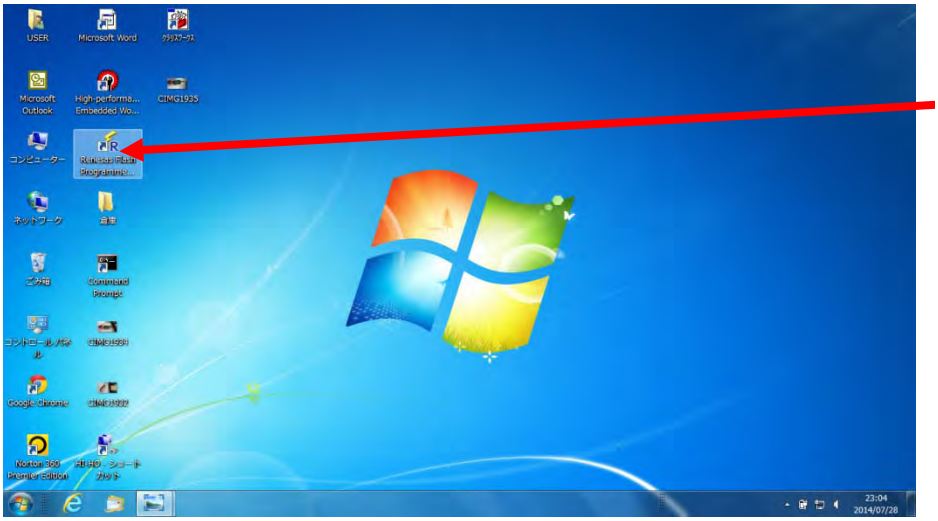
書き込み準備が出来た、
「RX220 CPU ボード」＋「RX220　Base ボード」の状態

簡易入出力チエック部：

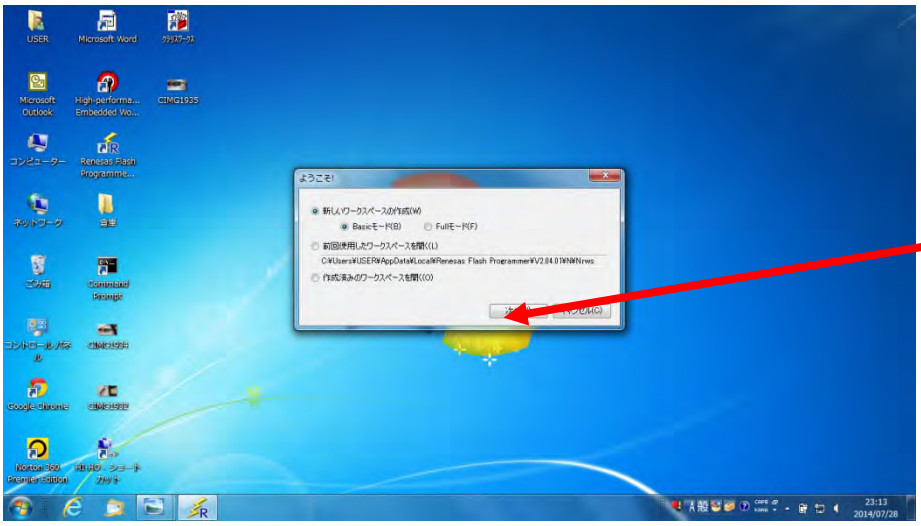
「RX220 Base ボード」上には、簡易的にプログラムの入出力の実行状態の目視チエックが出来る、入力スイッチ（ＳＷ１）と、出力ＬＥＤ１（赤），ＬＥＤ２（緑）が搭載されています。

入力スイッチ（ＳＷ１）を、ON 側にした時、ポートＨの２ビット目、PH2（CN2-17）に、“０”が入力され、ＯＦＦ側にした時、“１”が入力されます。

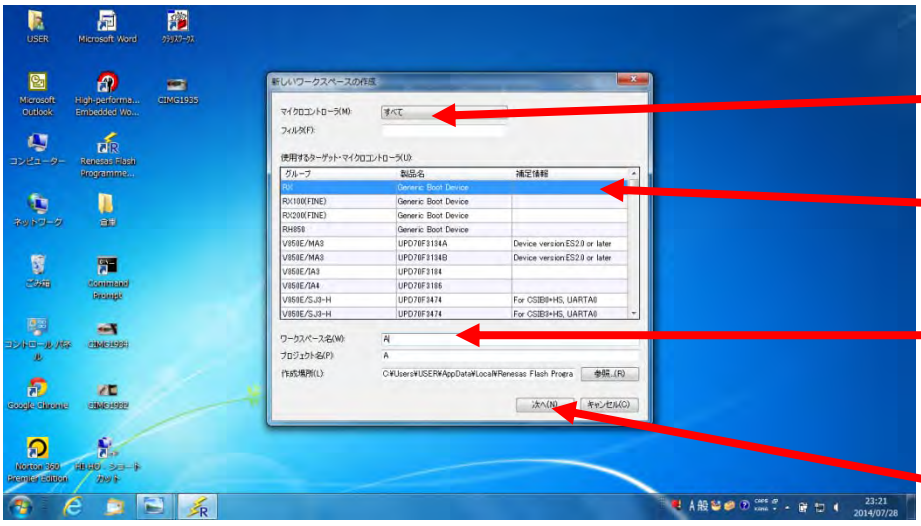
出力 LED1(赤)は、ポートＨの０ビット目、PH0(CN2-15)から“１”が出力されると点灯します。出力 LED2(緑)は、ポートＨの１ビット目、PH1(CN2-16)から“１”が出力されると点灯します。
（“０”が出力されると、消灯します）
[LED が点灯すると、LED 両端電圧は約２Ｖ、低くなります）]



「Renesas Flash Programme…」を、マウス左クリックする。



「次へ」を
マウス左クリックする。

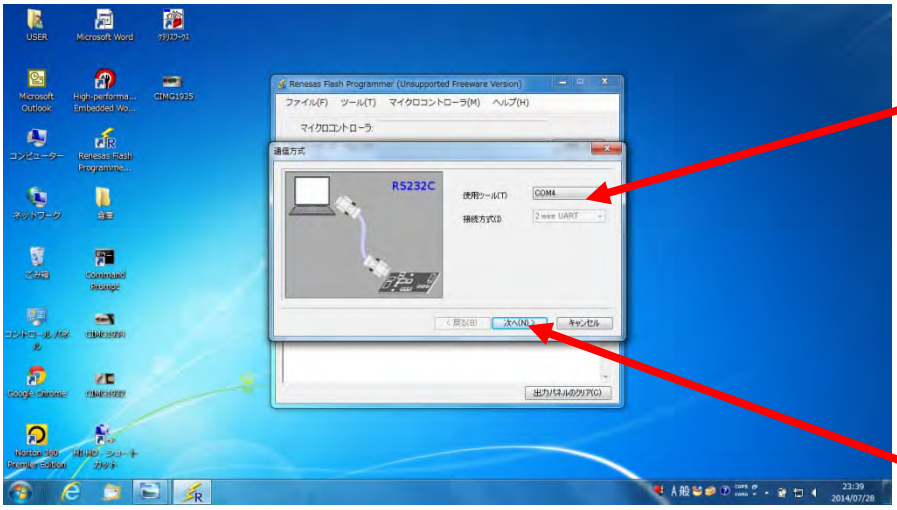


「すべて」を
マウス左クリックする。

「RX」を
マウス左クリックする。

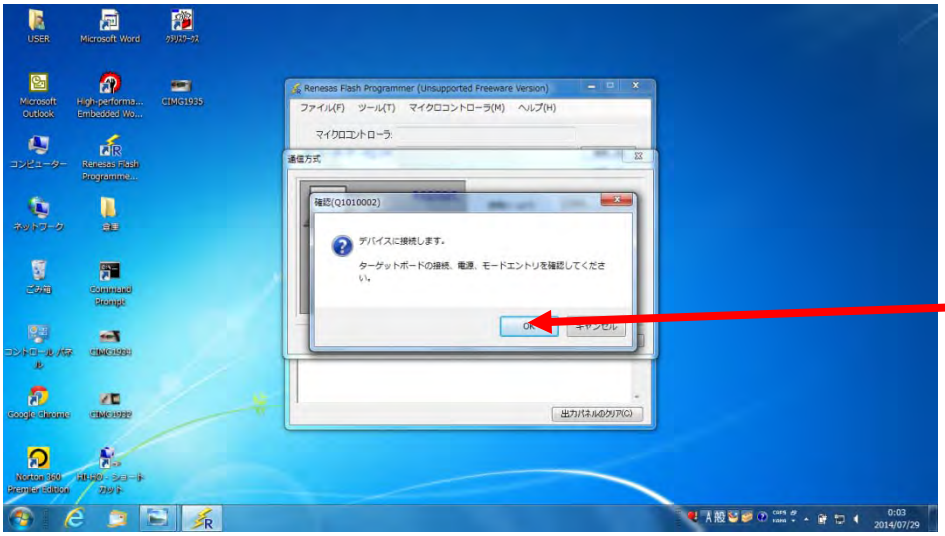
アルファベット、数字を入力する。
（適当なアルファベット、数字でよい）
今回は、「A」と、入力した。

「次へ（N）」を
マウス左クリックする。

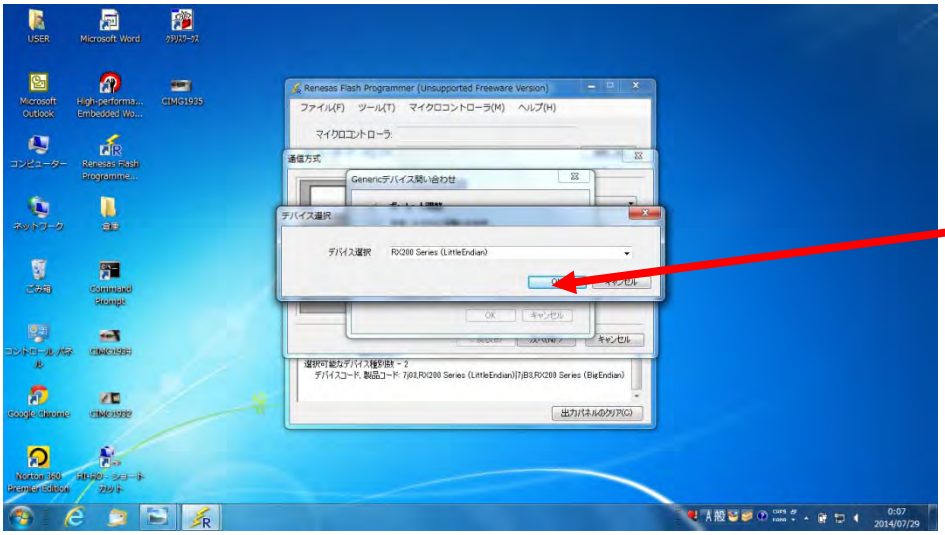


使用ツール（T）は、シリアル接続なので、COM 番号を選択する。
今回の COM4 の４は、PC の PORT ４番が選択されているということです。自動的に PORT 番号が、割り付けされ表示されます。
（COM 番号が表示されない場合は USB ケーブルが正しく接続されていないので、チェック要）

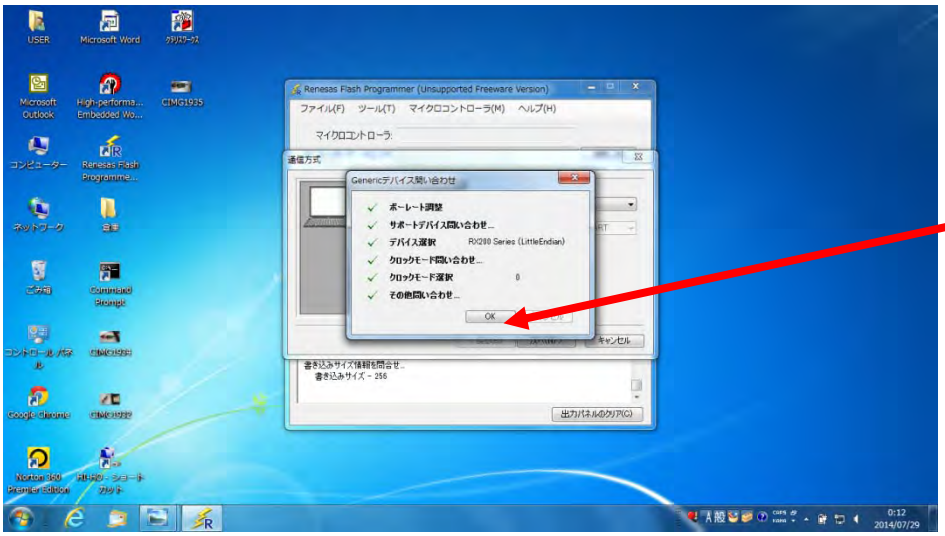
「次へ（N）＞」を
マウス左クリックする。



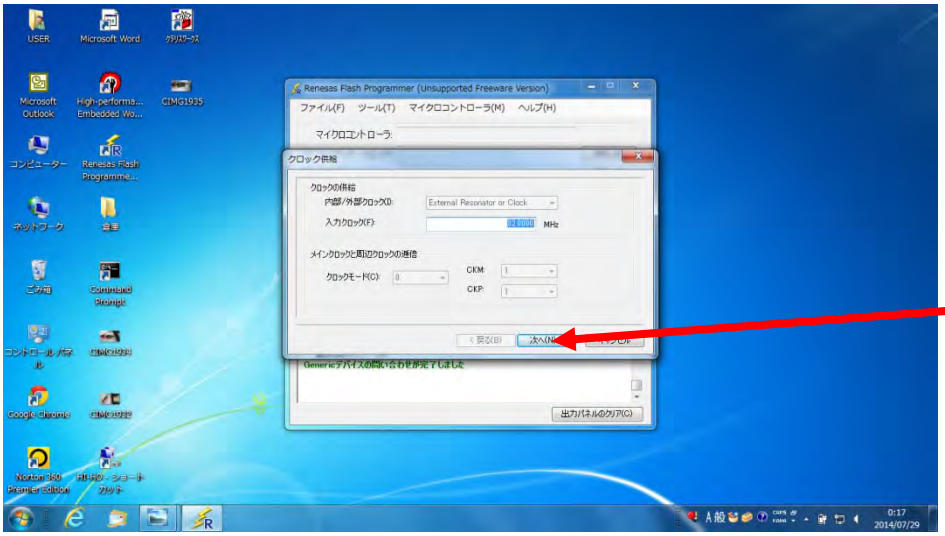
「OK」を
マウス左クリックする。



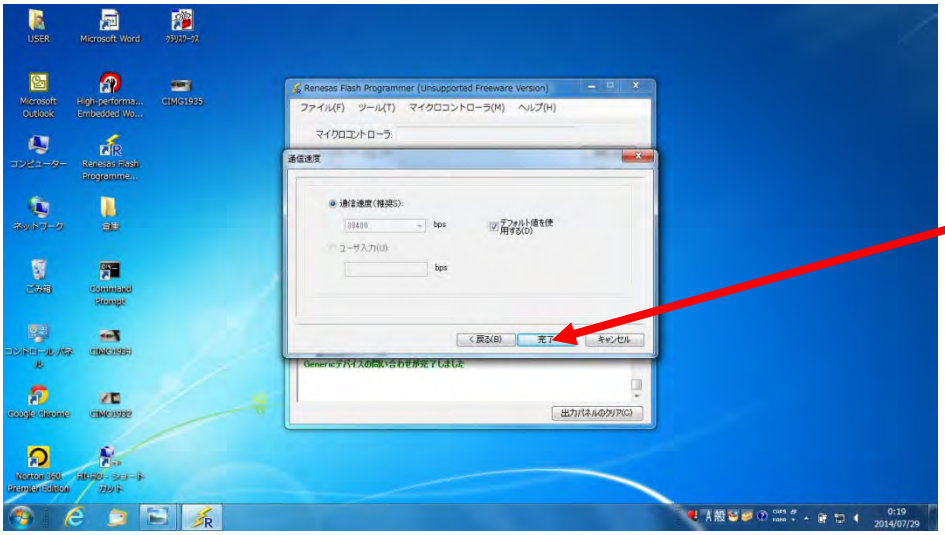
「OK」を
マウス左クリックする。



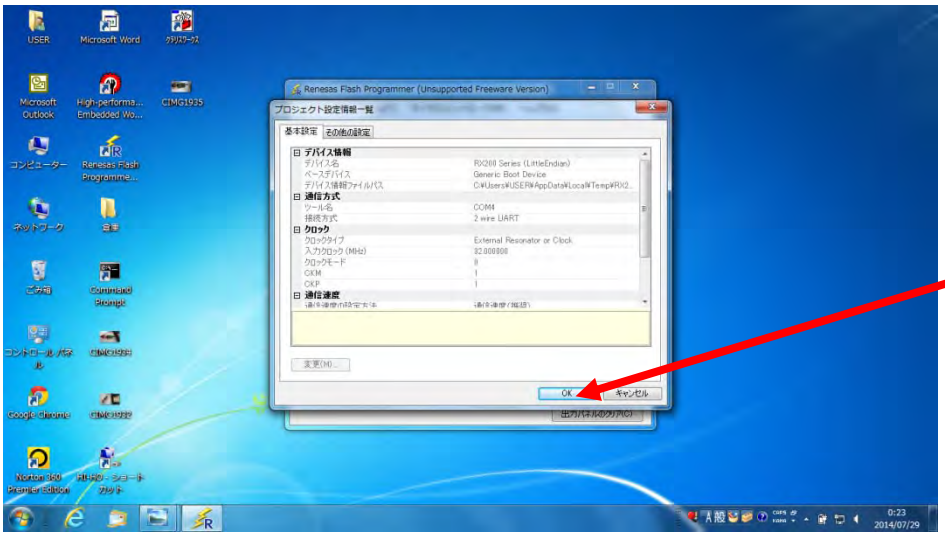
「OK」を
マウス左クリックする。



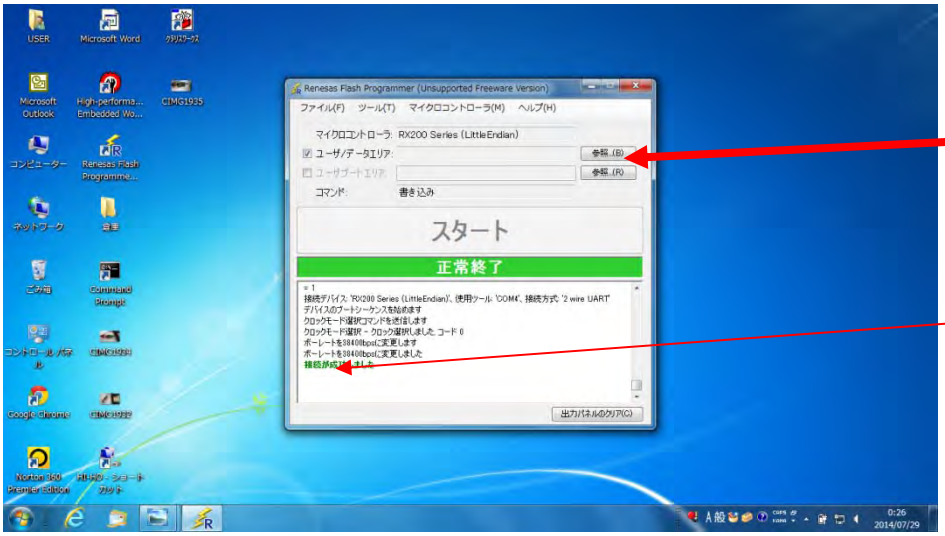
「次へ（N）>」を
マウス左クリックする。



「完了」を
マウス左クリックする。

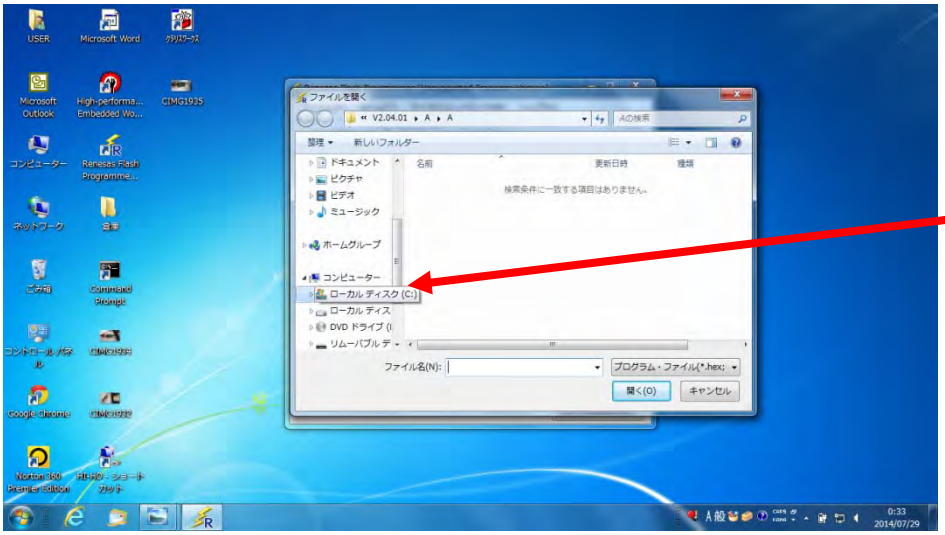


「OK」を
マウス左クリックする。

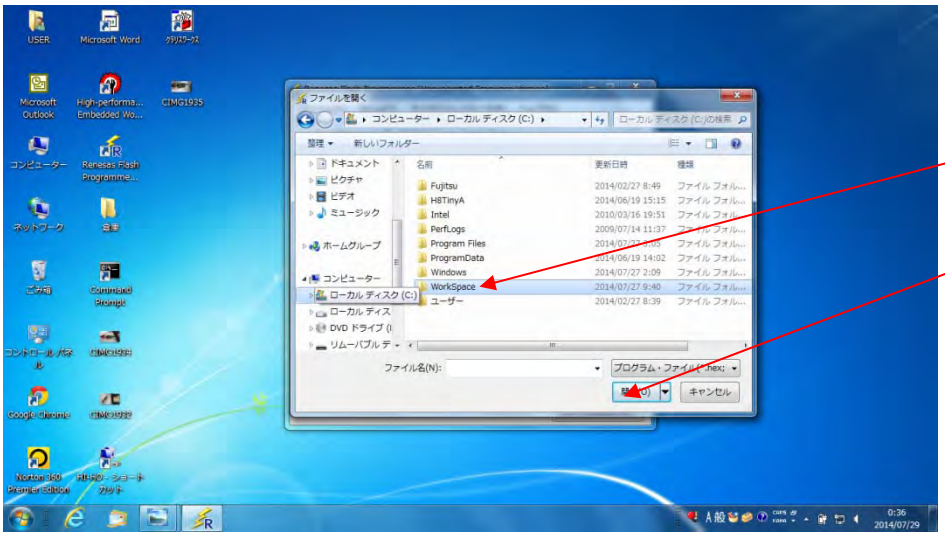


「参照...(B)」を
マウス左クリックする。

RX220 と PC(パソコン)とが、
正常に接続された。

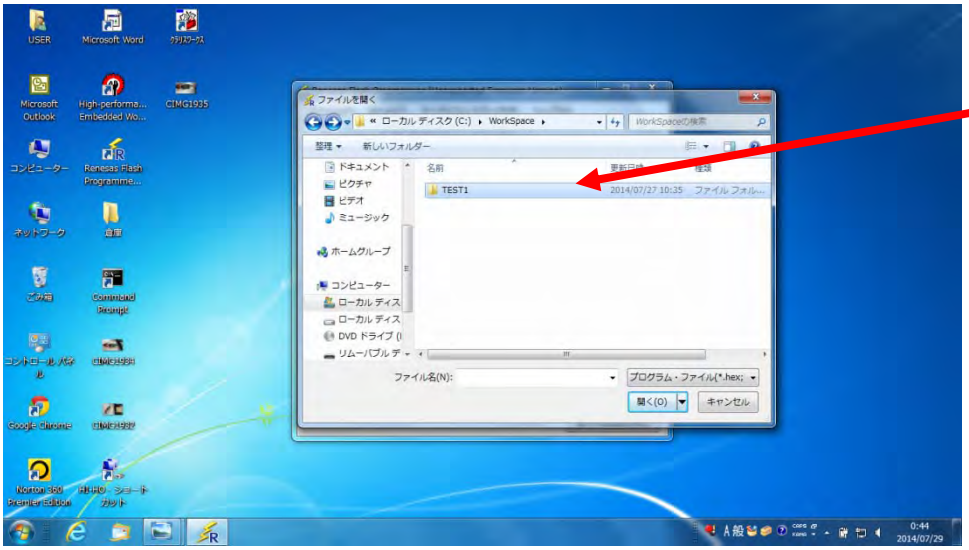


「ローカルディスク（C:)」を
マウス左クリックする。

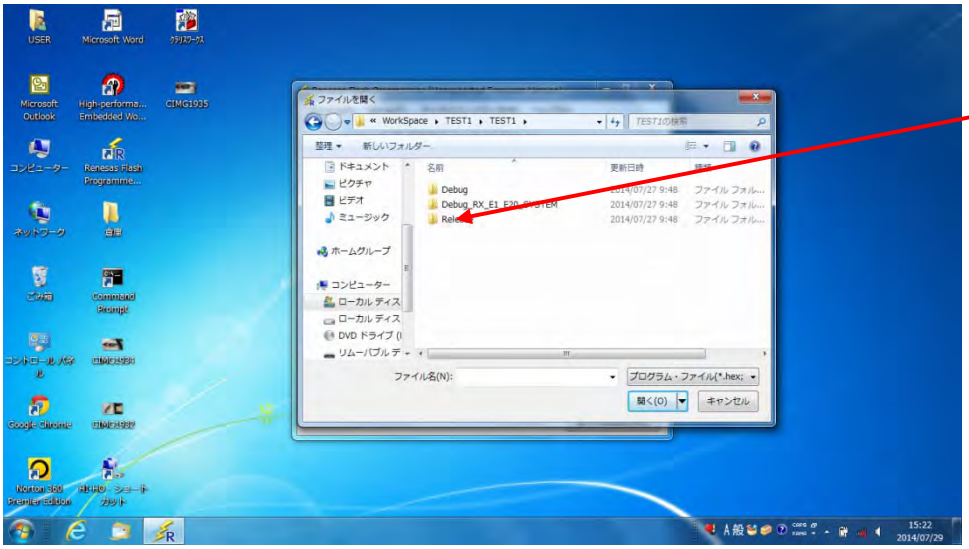


ローカルディスクの内容が、
表示される。

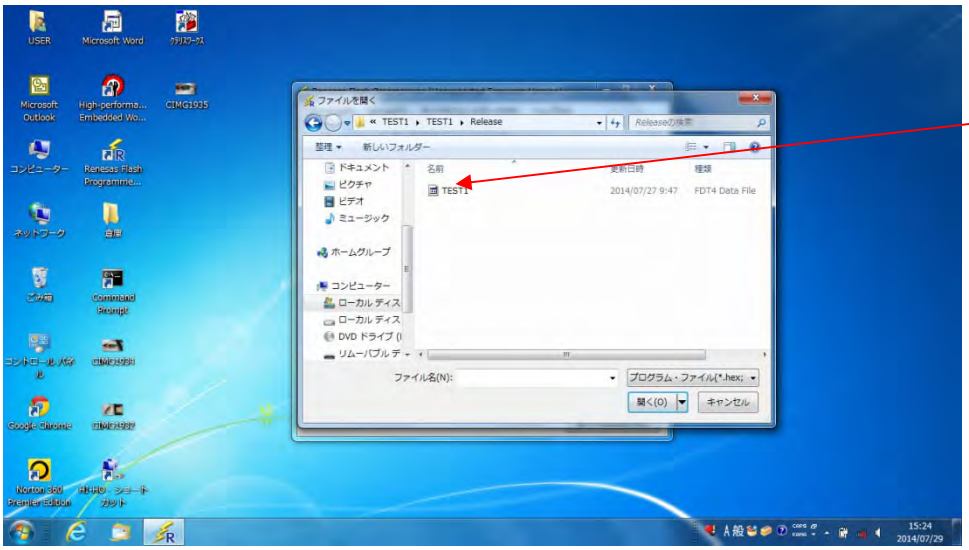
「WorkSpace」をマウス左クリッ
クし、「開く（O)」をマウス
左クリックする。



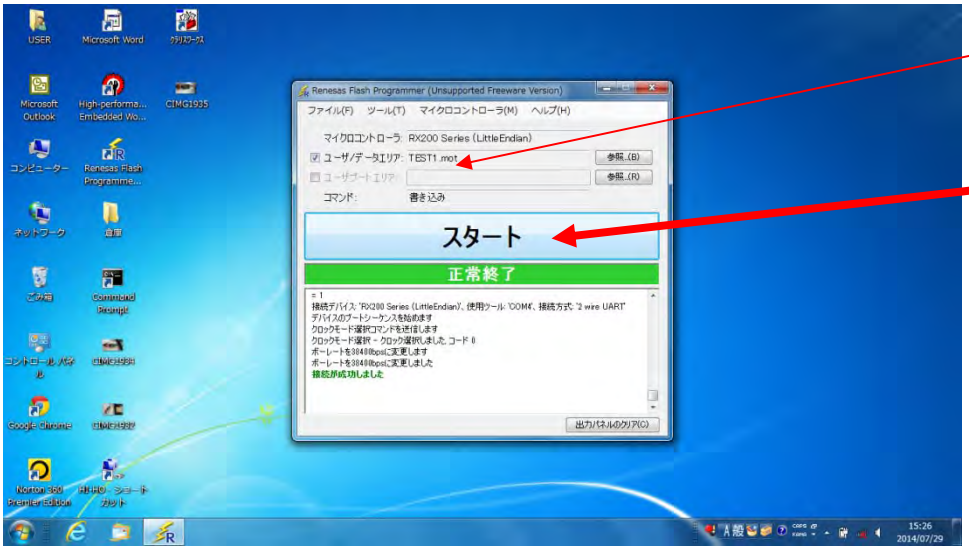
「TEST1」を
マウス左クリックする。



「Release」を
マウス左クリックする。

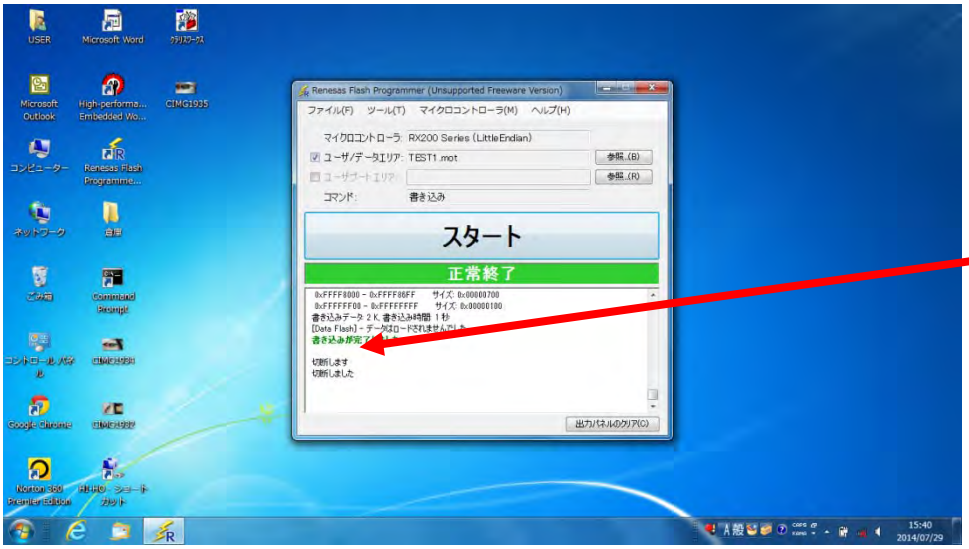


「TEST1」を
マウス左クリックする。



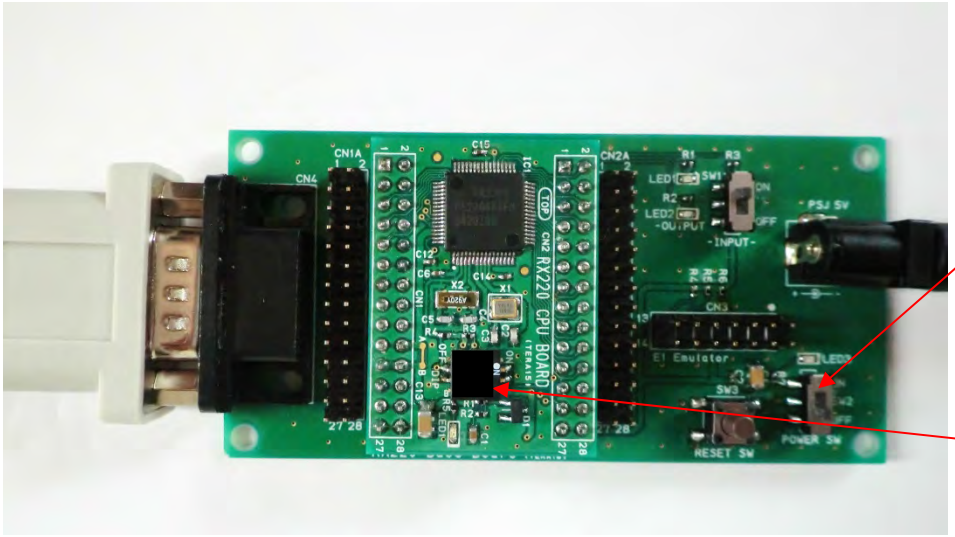
「TEST1.mot」が
表示される。

「スタート」を
マウス左クリックする。



「TEST1.mot」の
書き込みが完了する。
（プログラムの書き込みが
完了する）

プログラムの書き込みモード → 実行モードの切り替え



[ボード操作手順]

「RX220　Base ボード」の
POWER SW を OFF にする。

DIP SW の SW No.1 のみを、
「OFF 側」にする。

プログラムの実行（動作）

プログラムがマイコンのフラッシュ ROM に書き込まれました。次は、書き込まれたプログラムを
マイコンボード上で、実行（動作）してみましょう。

■RX220 マイコンは、RESET 時は、

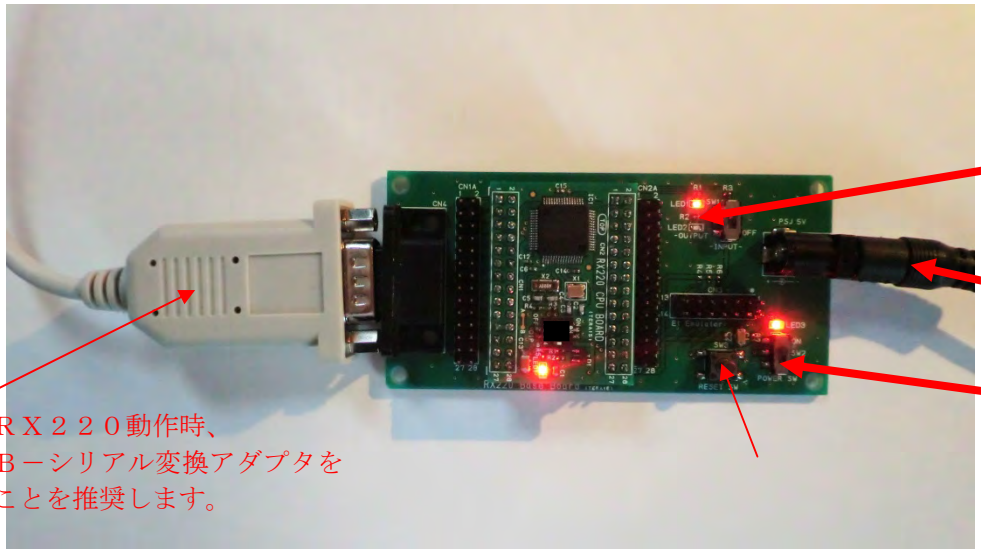
低速オンチップオシレータ（LOCO）の周波数１２５KHｚで動作します。

本プログラムも、周波数１２５KHｚで動作しています。

「RX220 CPU ボード」は、多くのクロック発生回路を内蔵し、きめ細かい周波数、電圧の
低減化制御により、低消費電力化を実現することができます。
下記に、「RX220 CPU ボード」の内蔵するクロック発生回路の種類について、纏めてみました。

■「RX220 CPU ボード」クロック発生回路の種類：

１、メインクロック発振器・・・・２０MHｚ(水晶振動子)
２、サブクロック発振器・・・・・３２．７６８KHｚ（水晶発振子）
３、高速オンチップオシレータ（HOCO）・・・32MHz/36.864MHz/40MHz/50MHz
４、**低速オンチップオシレータ（LOCO）・・・１２５KHｚ**（RESET 時は、本、オシレータが起動する）
５、IWDT 専用オンチップオシレータ・・・・１２５KHｚ



プログラムの作成→プログラムの書き込み→マイコンボードの動作（実行）までの、

操作手順は、以上で完了です。

[コラム]

■「第１章」の補足説明：

「RX220 CPU ボード」に、作成したプログラムを書き込む場合、２通りの方法があります。
第１章のパソコンと USB－シリアルケーブル書き込み（ブートモード書き込み）と、
第２章のパソコンと E1 エミュレータ（別売、M-06155）接続による書き込みです。
今回の「第１章」の操作手順について、要点のみを整理しておきたいと思います。

ます、パソコンと USB－シリアルケーブル（USB－シリアル変換ケーブルとも言う）を「RX220 Base ボード」の D-Sub（CN4）に接続して（但し、「RX220 Base ボード」の POWER SW(SW2)は、必ず OFF にしてください)、JP1 の 1Pin-2Pin,3pin-4pin,5pin-6pin も、それぞれジャンパーピンで接続します。「RX220 CPU ボード」の DIP SW、１pin,2pin を ON（右側にスライドすることにより、ブートモード設定（シリアル書き込み）します）にしてから、「RX220　CPU ボード」を、「RX220 Base ボード」上にセットします（逆差しに注意！）。AC アダプタ（５V、２A）を PSJ 5V 端子に接続して、「RX220 Base ボード」の POWER SW(SW2)を ON にすると、外部電源が印加されます（「RX220 CPU ボード」の LED1（赤)、「RX220 Base ボード」の LED3（赤）が点灯します)。
パソコンの Windows 画面にある「Renesas Flash Programme…」を、起動させ、一連の書き込み操作を行っていきます。
書き込み処理が正常に終わったなら、「RX220 Base ボード」の POWER SW（SW2）を、OFF にします。
「RX220 CPU ボード」の DIP SW の 1Pin を OFF 側にし、「シングルチップモード」（実行）に設定します（DIP SW の 2pin は、ON/OFF どちら側でも可)。
「RX220 Base ボード」の POWER SW（SW2）を ON にすると、作成したプログラムが実行され、「RX220 CPU ボード」が動作し、「RX220 Base ボード」の LED1（赤）,LED2（緑）が交互に点滅を繰り返します。

■「動作モード」の補足説明：

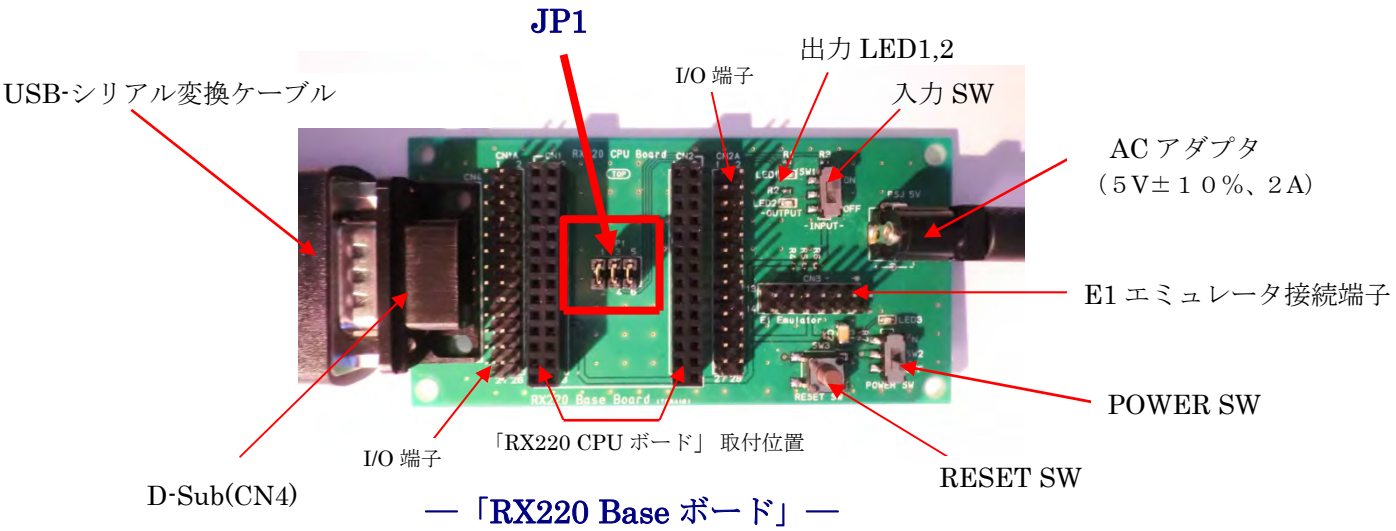
「RX220 CPU ボード」には、動作モード用のスイッチがあります。
これが、DIP スイッチです。プログラムの書き込みを行う時は、
DIP スイッチの 1pin,2pin を ON 側にします。
このモードを、「ブートモード」と言います。

プログラムを実行させる場合には、DIP スイッチの 1Pin を OFF 側します（2Pin は、ON/OFF どちらでも可)。
このモードを、「シングルチップモード」と言います。

注、モード切替えをする場合には、「RX220 Base ボード」の POWER　SW を必ず OFF にしてから、「動作モード」の SW 切り替えをおこなってください。

「ユーザブートモード」のモードもありますが、特殊な使用方法のため、
説明は省略致します。
ご使用に当たっては、ルネサスエレクトロニクスが発行する、
「RX220 グループユーザーズマニュアル（ハードウェア編)」を、ご参照ください。

■「JP1」の補足説明：
「RX220 Base ボード」には、JP1（1-2,3-4,5-6）のピンヘッダがあります。
JP1 の 1-2 をジャンパーピンで接続すると、LED1(赤)と接続します。
ジャンパーピンを外すと、ポート H の 0 bit 目（CN2-15）が開放状態となります。
JP1 の 3-4 をジャンパーピンで接続すると、LED2(緑)と接続します。
ジャンパーピンを外すと、ポート H の 1bit 目（CN2-16）が開放状態となります。
JP1 の 5-6 をジャンパーピンで接続すると、入力 SW(SW1)と接続します。
ジャンパーピンを外すと、ポート H の 2bit 目（CN2-17）が開放状態となります。
LED1（赤）、LED2（緑）は、出力機能のチェック端子用として、入力 SW（SW1）は、
入力機能のチェック端子用として、活用することが出来ます。
ジャンパーピンを外すと、汎用 I/O ポートとして、使用することが出来ます。



■「低速オンチップオシレータ（LOCO）」の補足説明：
「RX220 CPU ボード」を「シングルチップモード」で動作させると、
パワーオンリセット回路が自動的に働き、「RESET」（初期化）されます。
その時のクロック発生回路は、「低速オンチップオシレータ（LOCO）」が
優先的に選択され、周波数１２５KHｚで、「RX220 CPU ボード」が動作
します。（この「RX220 CPU ボード」のプログラムも、LOCO１２５KHｚで動作しています）
メインクロック２０MHｚ　及び　サブクロック３２．７６８KHｚで、
動作させる場合は、LOCO 起動動作（１２５KHｚ）から、メインクロック動作　及び
サブクロック動作に切り替わるように、RX220 マイコンの関連コントロールレジスタを
プログラム上で操作して、クロック発生回路を切り替えなければなりません。
ご使用に当たっては、「ルネサスエレクトロニクス」が発行する、
「RX220 グループユーザーズマニュアル（ハードウェア編）」をご参照ください。

●尚、「D、第４章」に、メインクロック２０MHｚ、
サブクロック３２．７６８KHｚへの遷移プログラムについて、記載しております。

第１章 

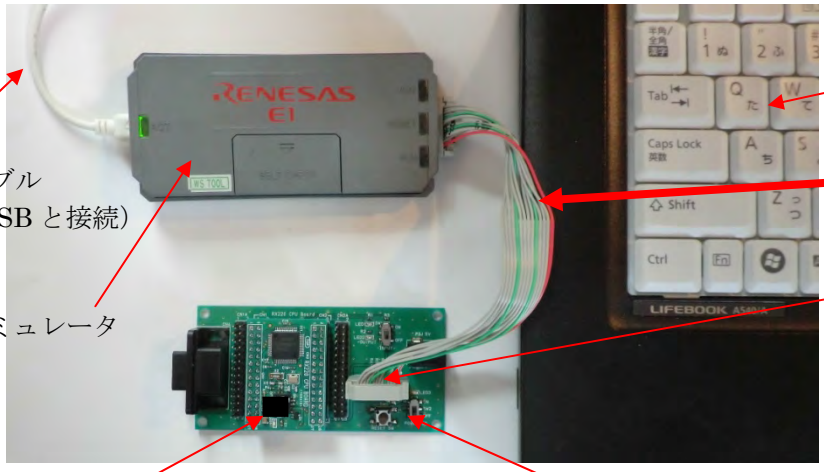
B、第２章：

E1 エミュレータ使用による「TEST1」プログラムの書き込み操作方法について、
説明致します。（E1 エミュレータは、弊社通販コード M－０６１５５にて購入出来ます）

注)、電源（５V）は E1 エミュレータから印加されますので、
必ず、「RX220　Base ボード」の POWER SW は、OFF としておきます（厳守！）。
尚、USB ポートは、十分な電流供給能力のあるものをお使いください。

「RX220　Base ボード」の「E1 Emulator（CN3)」に、E1 エミュレータの
コード（１４本）を接続します。E1 エミュレータの 1 Pin は赤色コードですので、
Pin 番号を間違わないように接続します。（コネクタの逆差しに、注意してください）

E1 エミュレータの ACT 側にはミニ USB、PC 側には、USB の形状を接続します。
「RX220　CPU ボード」の、「DIP」 スイッチは、 1 Pin は OFF 側、
２Pin は ON,OFF どちらでも可です（「シングルチップモード」にする)。



USB ケーブル
（PC の USB と接続）

E1 エミュレータ

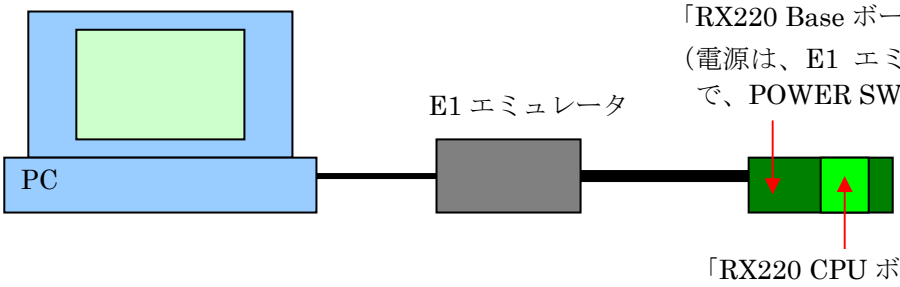
PC（Windows7 以降搭載のパソコン）

赤色コードが 1Pin

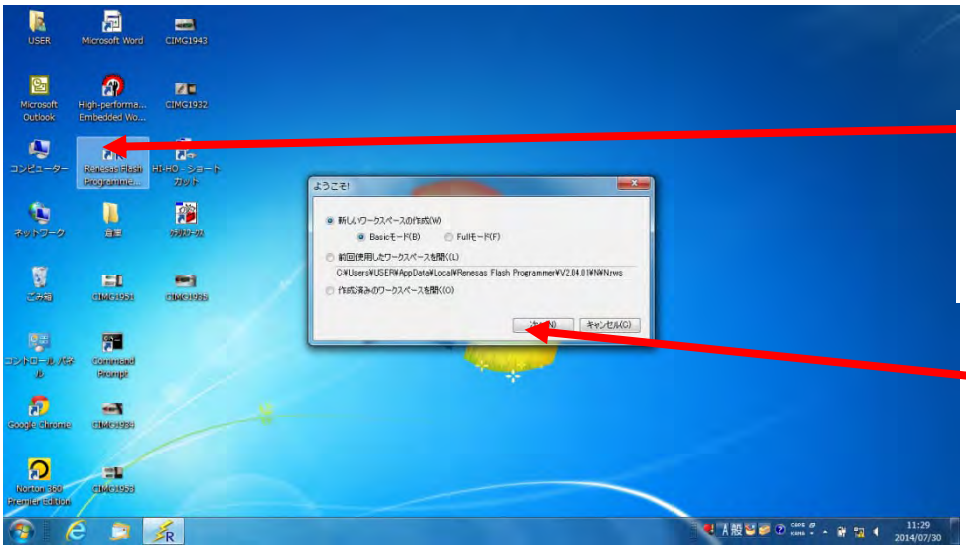
E1 エミュレータのコード(14 本)
をセットする。 1 Pin の位置を、
間違わないように接続すること。

DIP スイッチ(モード SW)の 1 Pin は
OFF 側、
２Pin は、どちらでも可。

必ず、POWER SW (SW2) を OFF とする。

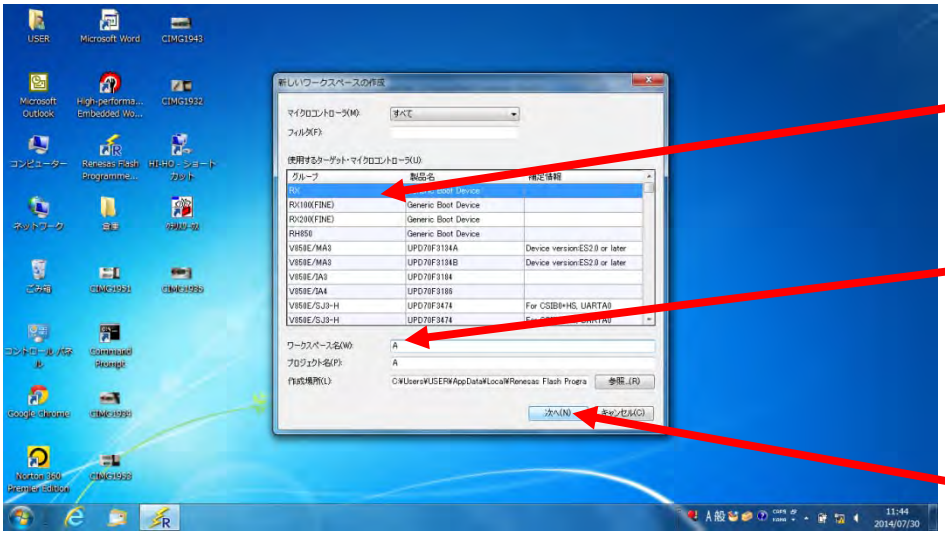


－＜E1 エミュレータ接続方法＞－



「Renesas Flash Programme…」
を、マウス左クリックする。

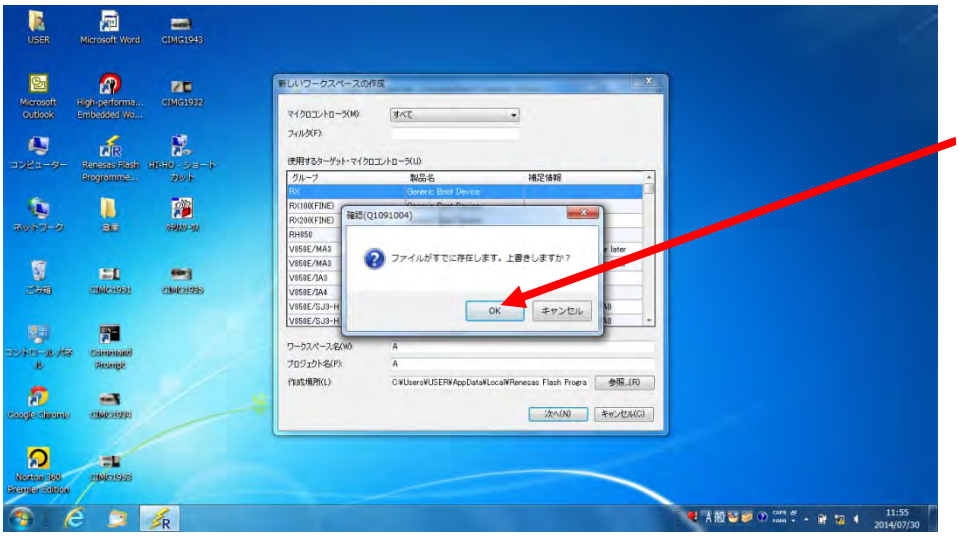
「ようこそ！」画面が
表示され、「次へ（N）」を
マウス左クリックする。



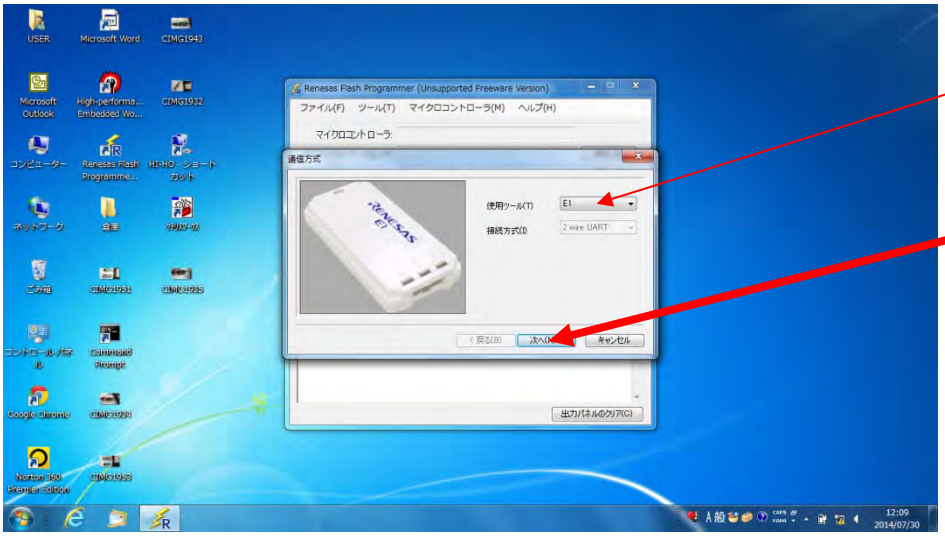
「RX」を
マウス左クリックする。

「A」を入力する。
（数字かアルファベットを
適当に入力する）

「次へ（N）」を
マウス左クリックする。

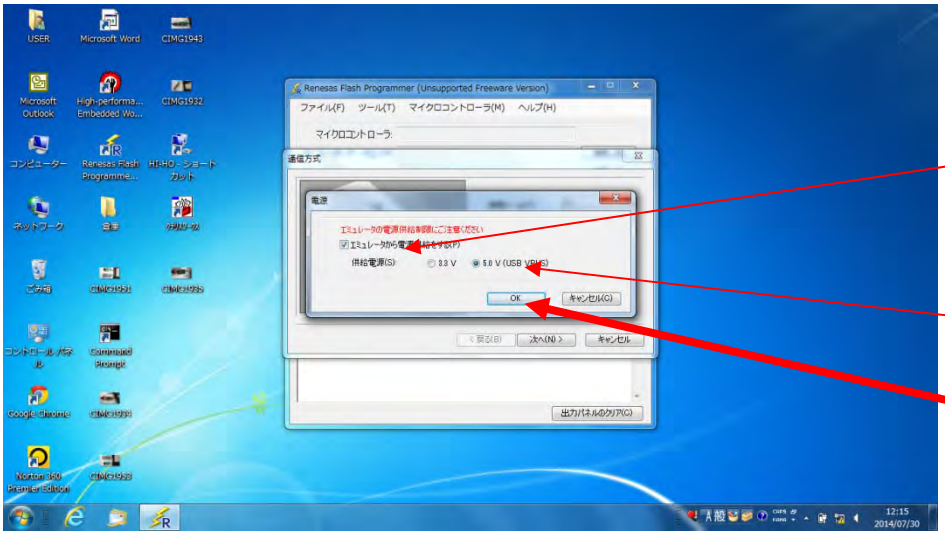


このメッセージが表示されたら、
「OK」をマウス左クリックする。



「E1」を選択する。

「次へ（N）>」を
マウス左クリックする。



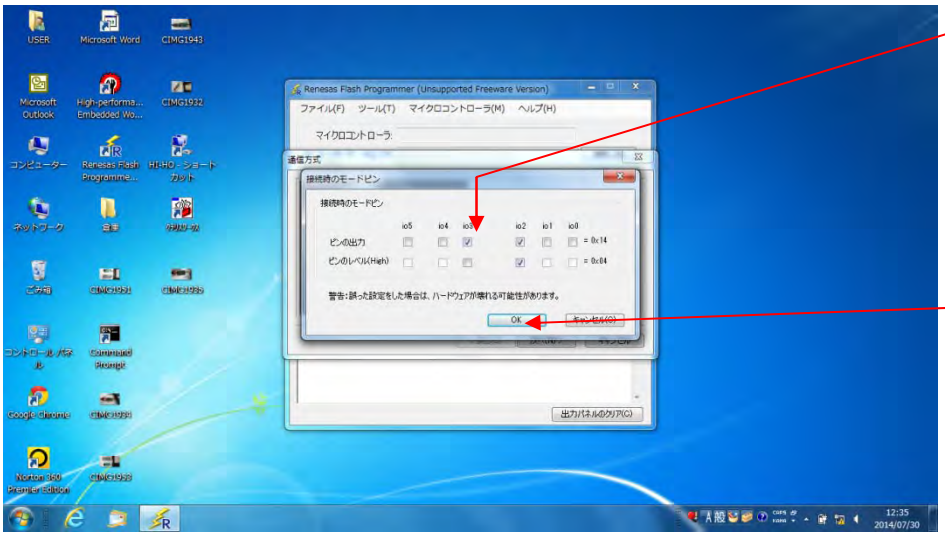
「エミュレータから電源供
給する（P)」にチェック
マークを付ける。

↓

「5.0V(USB VBUS)」に
チェックマークを付ける。

↓

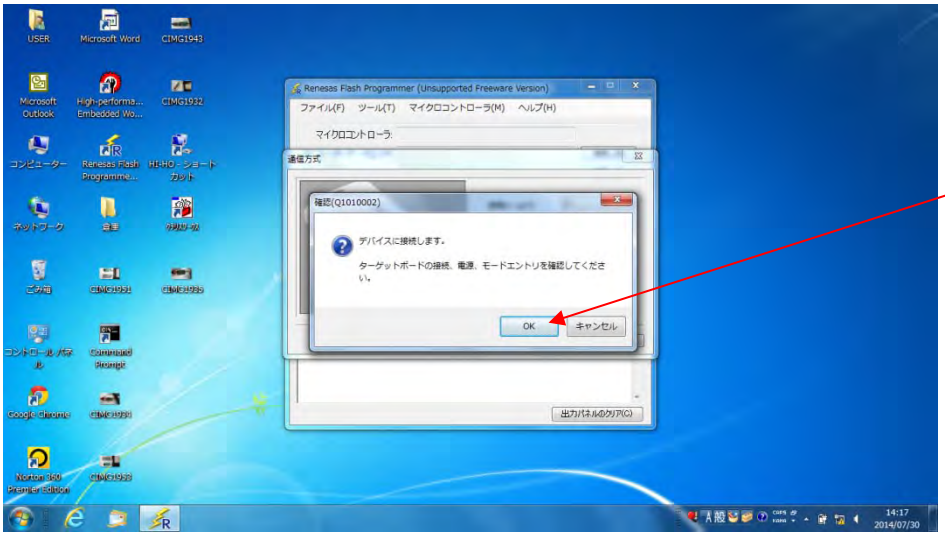
「OK」を
マウス左クリックする。



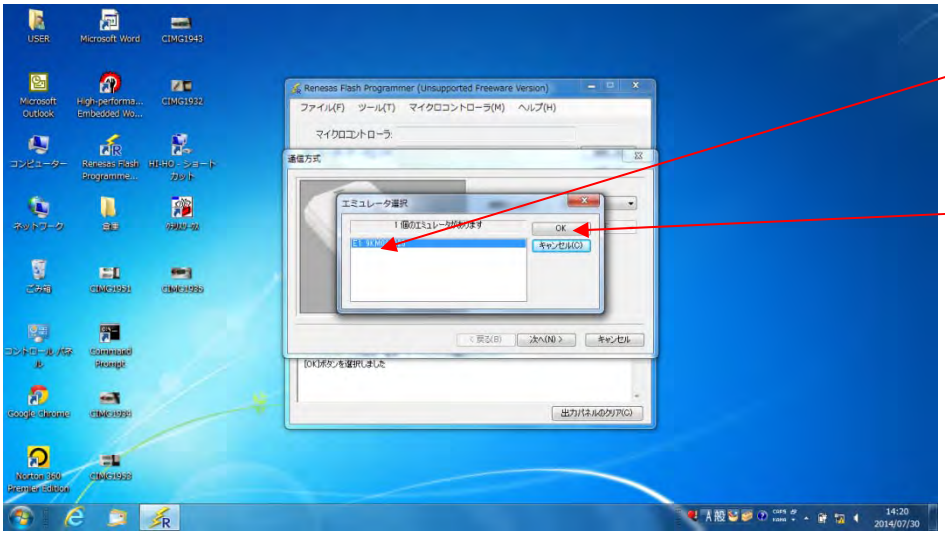
設定注意！：
「ピンの出力」は、io2,io3、
「ピンのレベル(High)」は、
io2にチェックマークを付ける。

↓

「OK」を
マウス左クリックする。



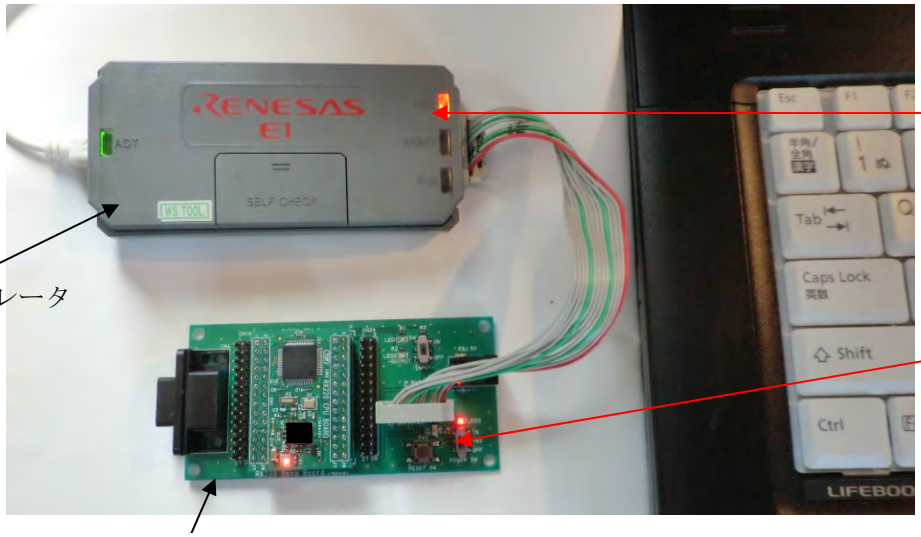
「OK」を
マウス左クリックする。



E1 エミュレータの「S/N No.」
が表示される。

↓

「OK」を
マウス左クリックする。

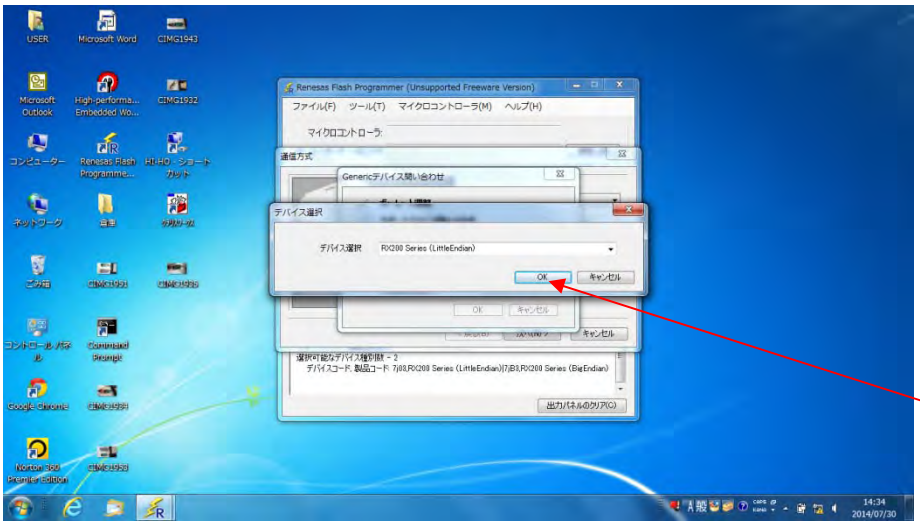


E1 エミュレータ

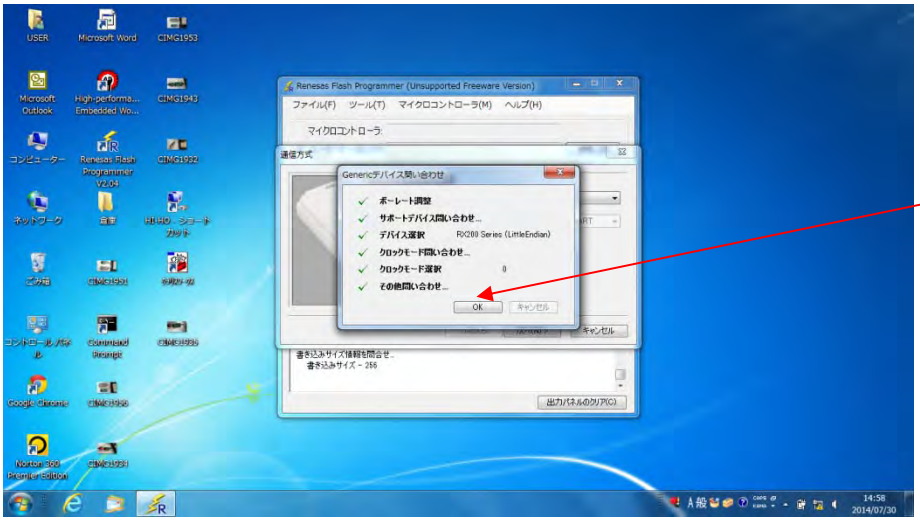
「VCC」が点灯

「Power SW」が OFF であることを確認する(厳守)。
Power SW が OFF でも、E1 エミュレータからの電源印加（５Ｖ）により、LED1（赤）,LED3（赤）が点灯する(ボードの通電状態を表示している)。

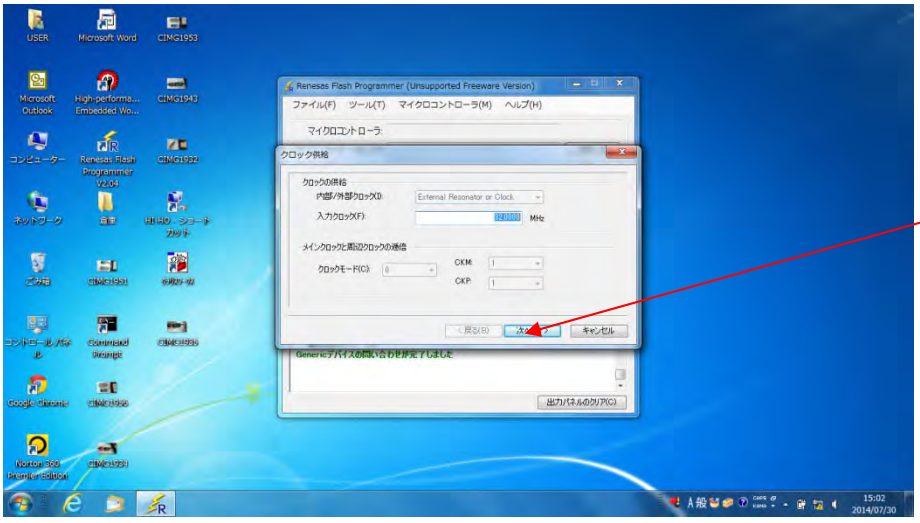
RX220 ボード（RX220 CPU ボード＋RX220　Base ボード）



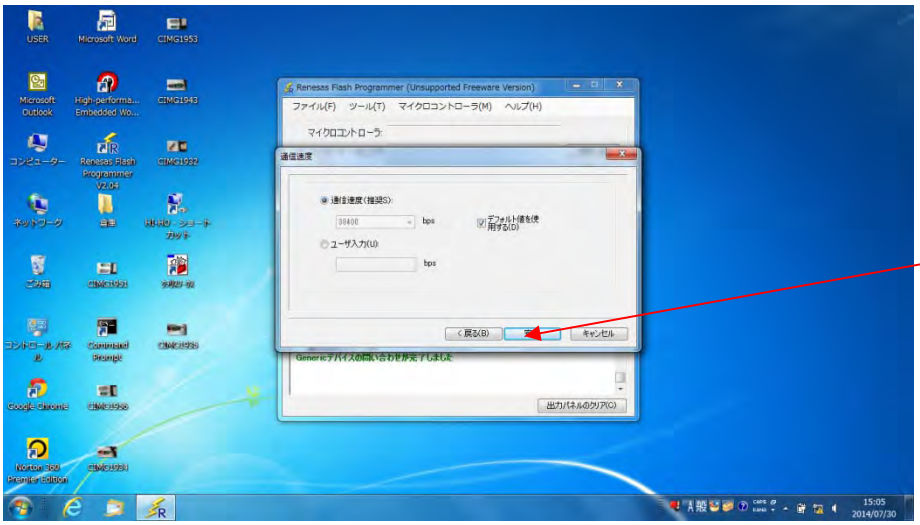
「OK」を
マウス左クリックする。



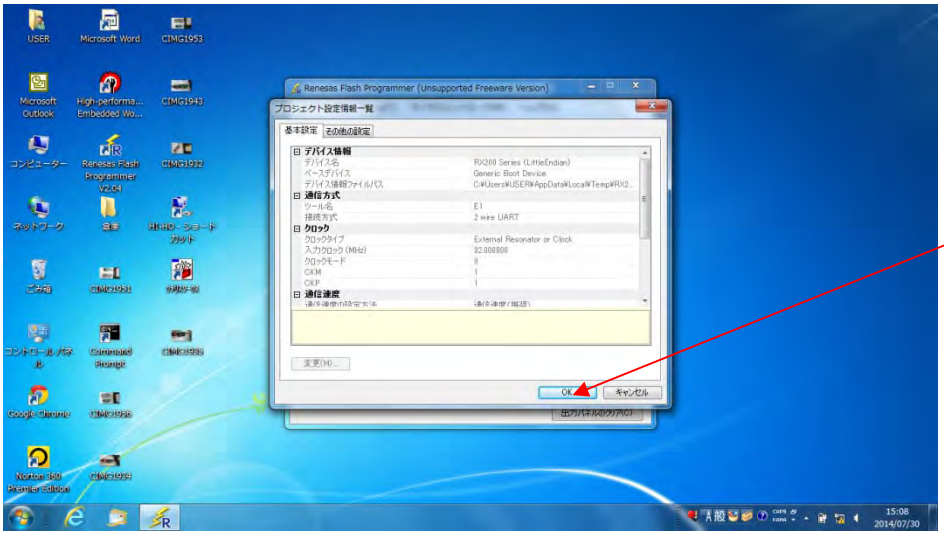
「OK」を
マウス左クリックする。



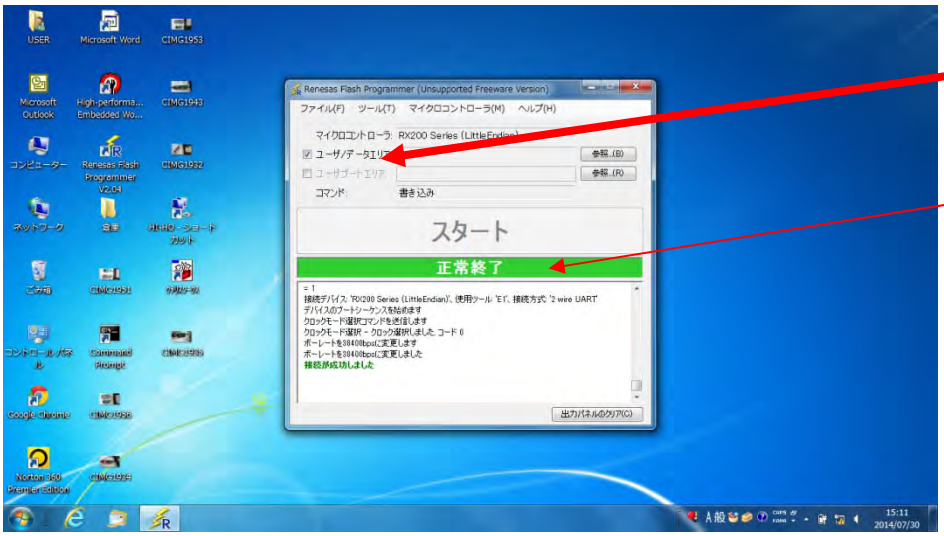
「次へ（N）>」を
マウス左クリックする。



「完了」を
マウス左クリックする。

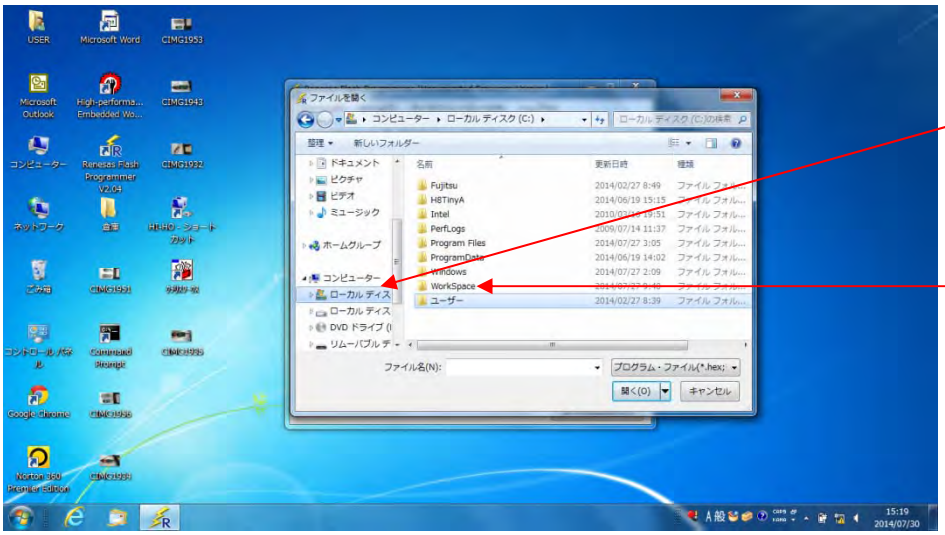


「OK」を
マウス左クリックする。



「ユーザ/ データエリア」にチェックマークを付け、「参照...（B)」を、マウス左クリックする。

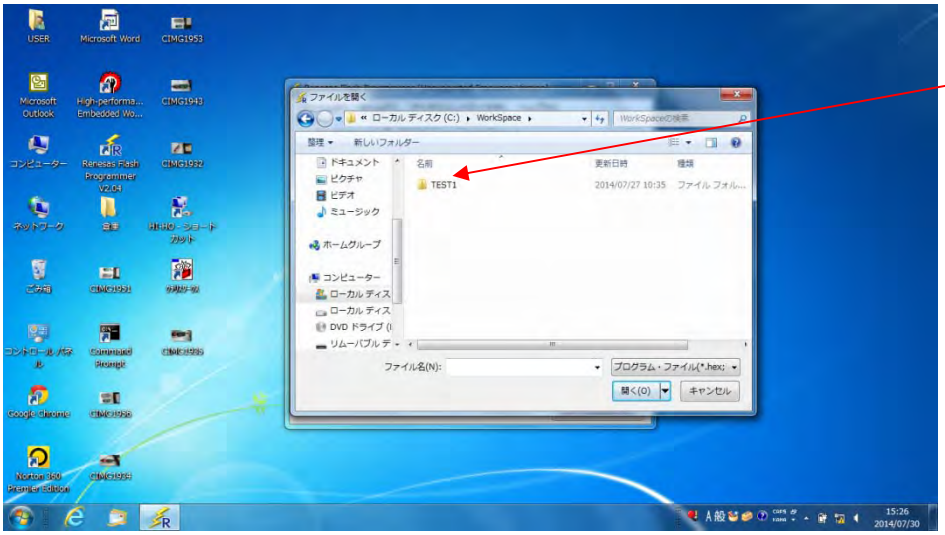
接続が成功しました。



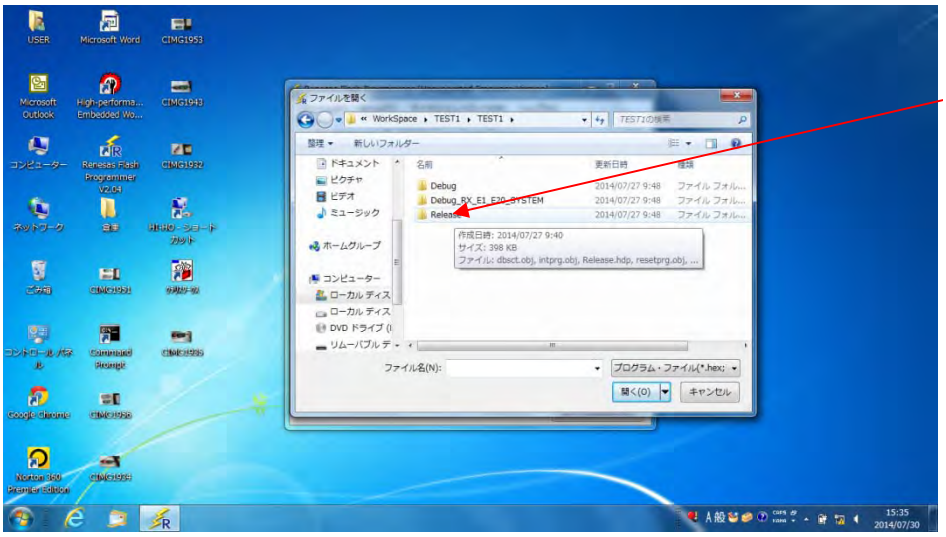
「ローカルディスク（C)」を
マウス左クリックする。

↓

「WorkSpace」を
マウス左クリックする。

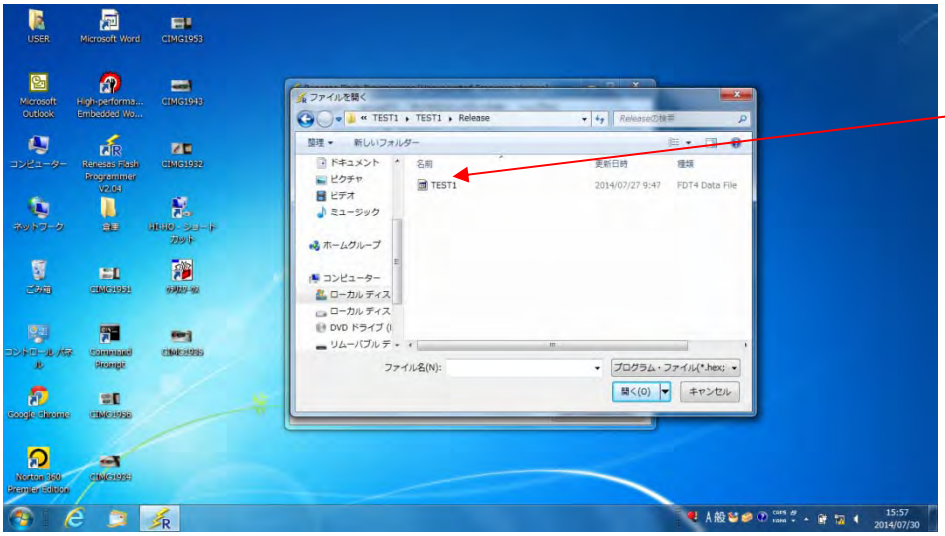


「TEST1」を
マウス左クリックする。



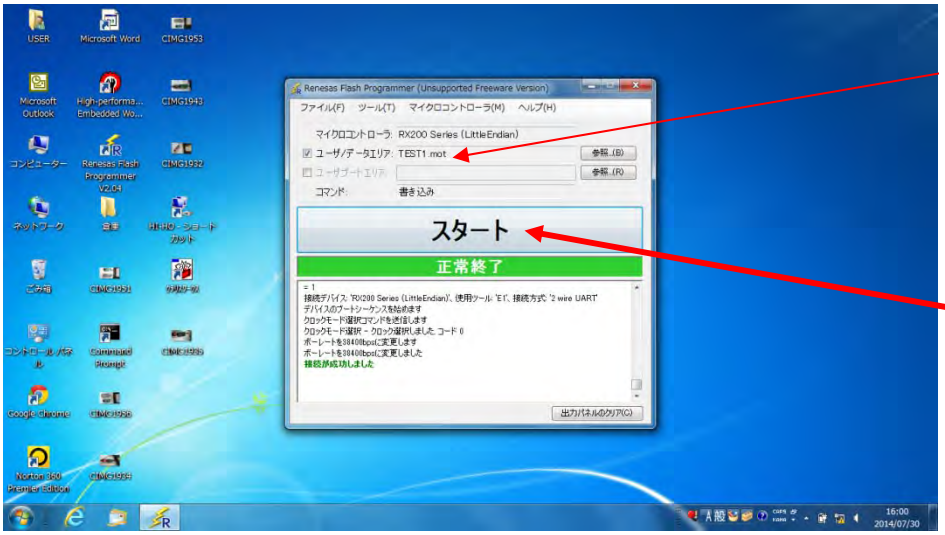
「Release」を
マウス左クリックする。

（Debug 作業を行わないので、
「Release」ファイル選択で、OK）



「TEST1」を
マウス左クリックする。

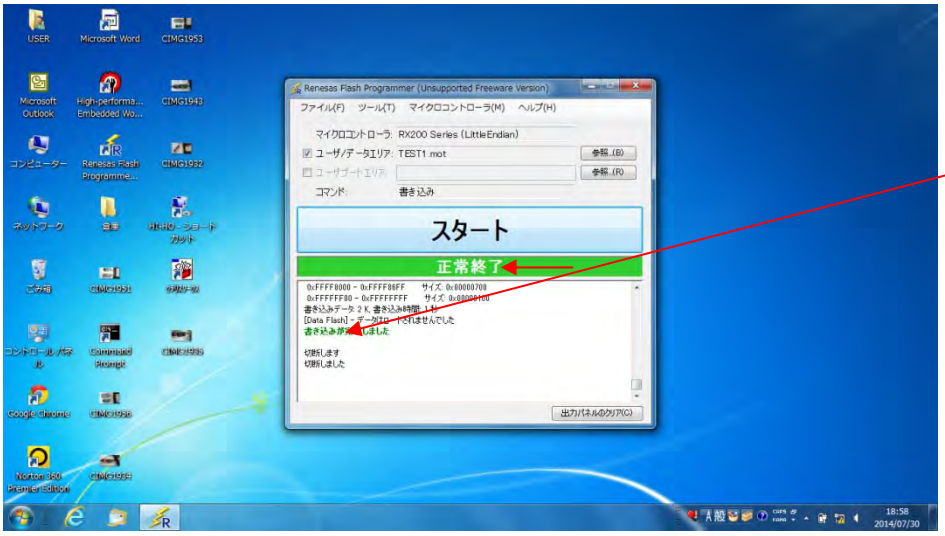
（今回の LED1（赤）、
LED2（緑）の約 1 秒間隔で交
互に点滅を繰り返すプログラ
ムのファイル名が、「TEST1」
です）



ユーザ/データエリアに、
「TEST1.mot」が表示される。

（.mot ファイルが、
最終ファイル(RX220 が理解できる
ファイル）となる。

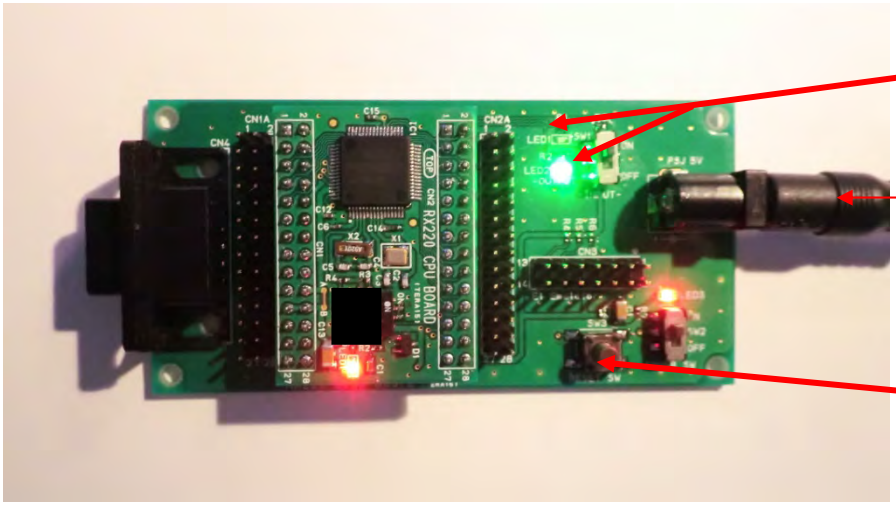
「スタート」を
マウス左クリックする。



「正常終了」・「書き込みが完
了しました」の表示で、E1 エ
ミュレータによる、「TEST1」
ソースプログラムの書き込み
が、正常にマイコンのフラッ
シュ ROM に、書き込まれま
した。

「RX220　Base ボード」の Power SW を、OFF 状態にしておきます。
E1 エミュレータと「RX220　Base ボード」が接続されているコード（14 本）を、取り外します。
「RX220　Base ボード」の PSJ　５V 端子に AC アダプタ（５V±１０%、２A）を接続します。
「RX220　Base ボード」の「DIP」　SW を、１Pin が OFF 側（左側）に（２Pin は、どちら側でも可）になっていることを確認します。（ＳＷをスライドの際には、ピンセットか、精密マイナス（－）ドライバの先で、ＳＷのスライド操作を行うとスムーズに移動ができます）

「RX220 Base ボード」の POWER SW を ON にします。
ハードウェア的に、パワーオンリセット回路が働き、自動的にリセットされ、
LED1（赤）、LED2（緑）が、約１秒間隔で、交互に点滅を繰り返します。



以上で、「E1 エミュレータ」によるプログラムの書き込み操作説明を、終わります。

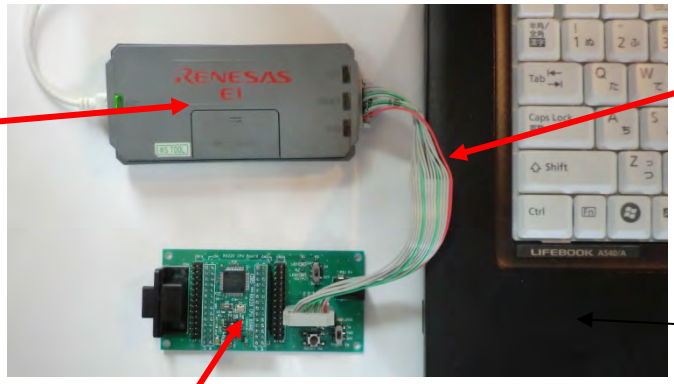
第２章　完

C、第３章：

E1 エミュレータを使用した「デバッグ機能」操作方法について、説明します。
「デバッグ機能」を利用すると、リアルタイムで、プログラムの書き込みはもちろん、

- ●レジスタ値の参照、設定
- ● メモリ値の参照、設定
- ● C 言語変数の参照、設定
- ● プログラムの指定アドレスでのブレーク（ソフトウェアブレーク、オンチップブレーク）
- ● 指定した条件でのプログラムの停止（オンチップブレーク）
- ● プログラム実行履歴の参照（トレース）

と、多くのプログラムの「デバッグ機能」が可能となります。
「ＲＸ２２０　ＣＰＵボード」の「ＤＩＰ」ＳＷは、「１」Pin をＯＦＦ側、
「２」Pin は、どちら側でも可です（「シングルチップモード」にします」。
ＲＸ２２０ボードの電源（５Ｖ）は、Ｅ１エミュレータから印加しますので、
「ＲＸ２２０　Ｂａｓｅボード」の POWER SW（ＳＷ２）は、ＯＦＦとします（厳守）。
ＰＣ(パソコン)のＵＳＢ端子－Ｅ１エミュレーター「ＲＸ２２０ボード (CPU ボード＋Base ボード)」を、
それぞれ接続し、セットします。
ＵＳＢポートの電流供給能力には、十分に余裕のあるものをお使いください。



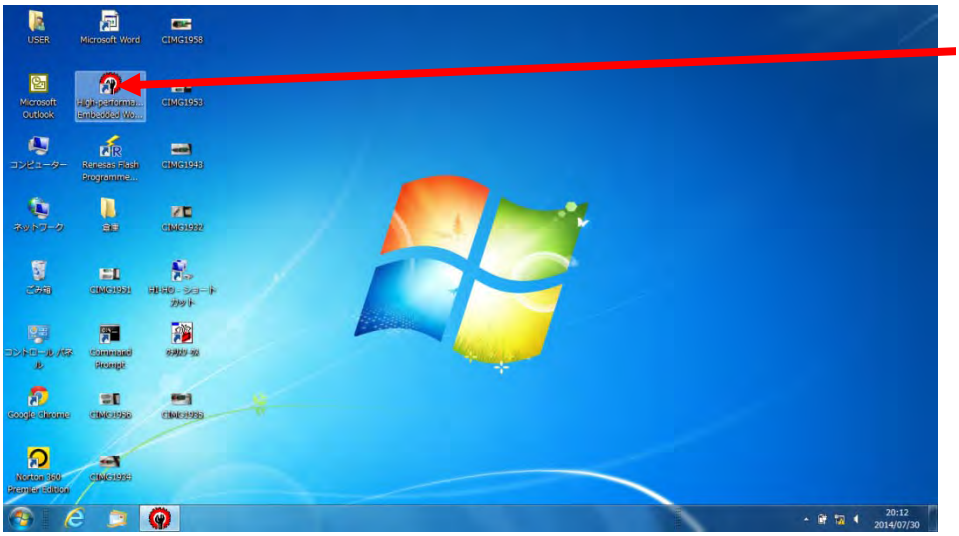
Ｅ１エミュレータ

赤色コードが、
１Pin です。
（逆差しに注意）

ＰＣ(パソコン)
（ＯＳは、Windows7 以降使用）

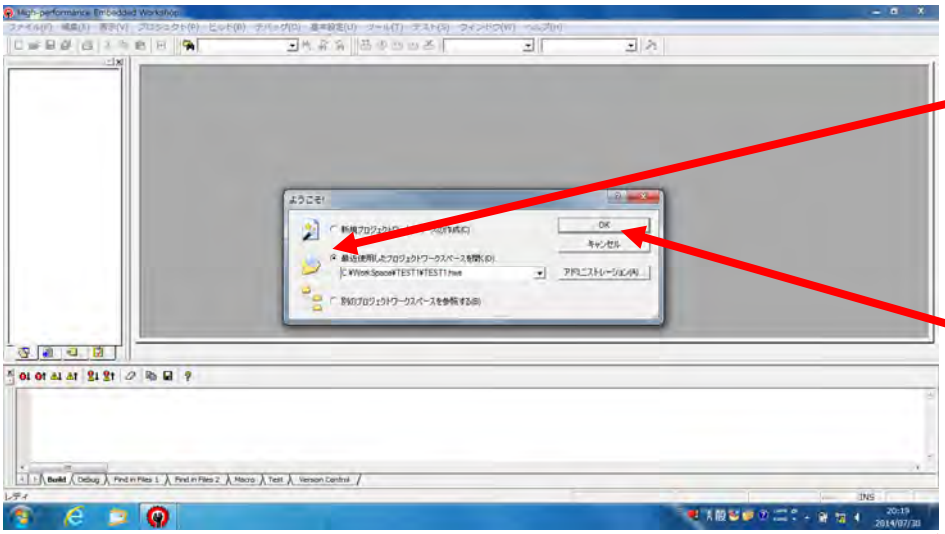
ＲＸ２２０ボード（CPU ボード＋Base ボード）

それでは、**「デバッグ機能」**を動かしてみよう。



Window 画面の
「High-performa…
EmbeddedWo…」を、
マウス左クリックする。

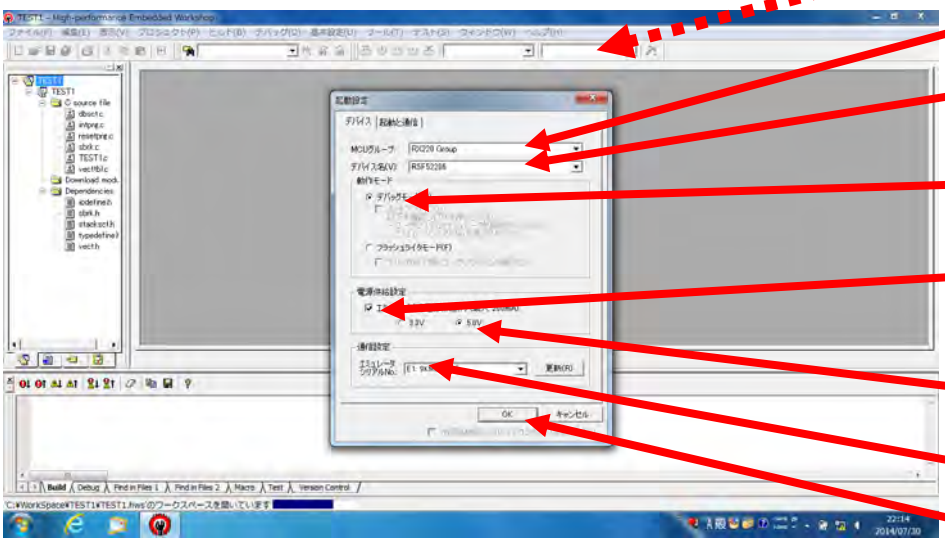
「デバッグ機能」とは？
ソフトウェアに於ける
プログラムのバグ（誤りや
欠陥）を探して、正しく動作
するように修正する機能。



「TEST1. c 」のプログラム
は、作成済みなので、
「最近使用したプロジェクト
ワークスペースを開く
（O）：」に、チェックマーク
を付ける。

「OK」を
マウス左クリックする。

最初に
「SessionRX_E1_E20_SYST」
を選択



「RX220 Group」を選択

「R5F52206」を選択

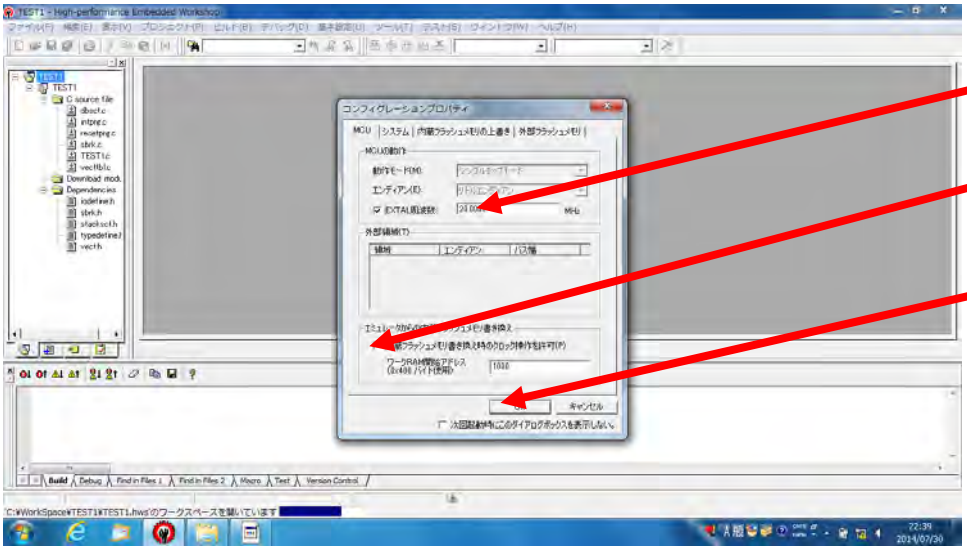
「デバッグモード（D）」に
チェックマークを付ける

「エミュレータから電源供給（P）
(最大 200mA)」に、
チェックマークを付ける

「5.0V」にチェック
マークを付ける

エミュレータの S/N が表示される

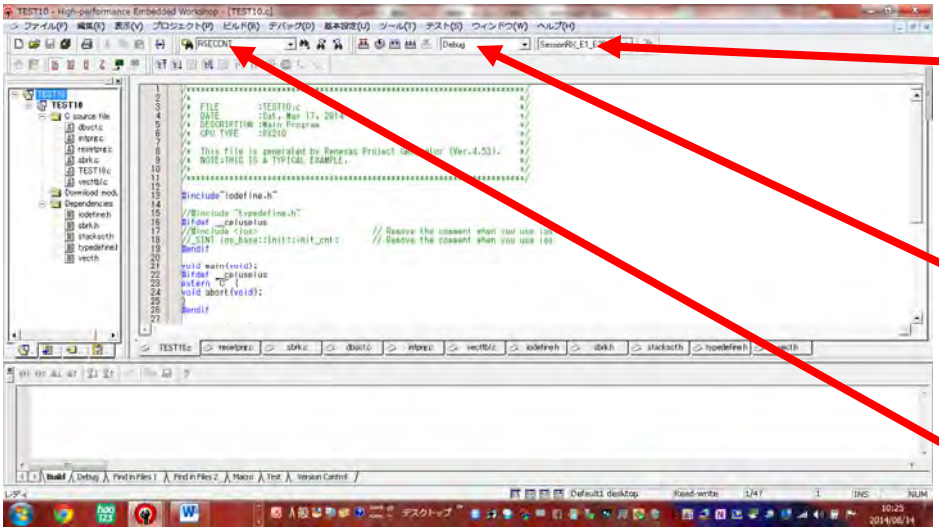
「OK」を、マウス左クリックする



２０ＭＨｚを確認

「内蔵フラッシュメモリ書き換え」にチエックマークを付ける

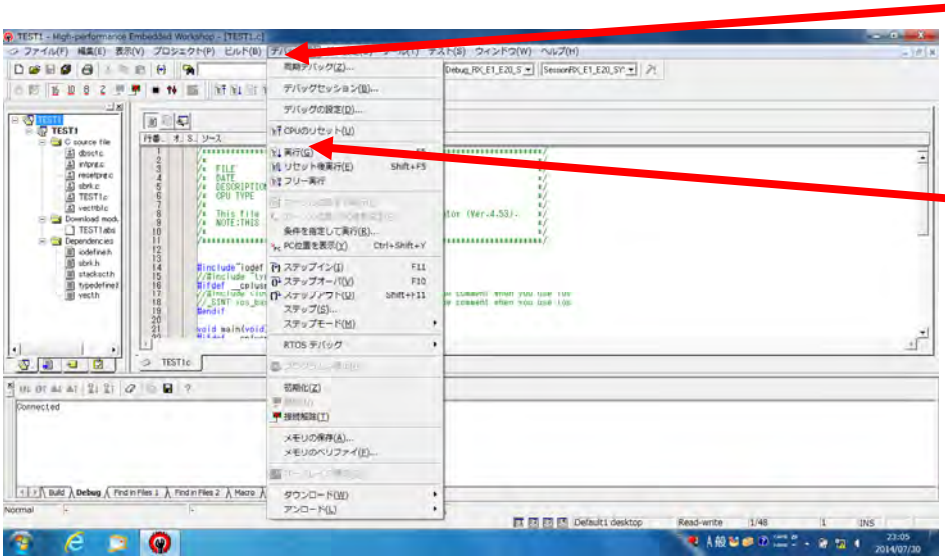
「OK」をマウス左クリックする



未選択の場合は「SessionRX_E1_E20_SYST」を選択する

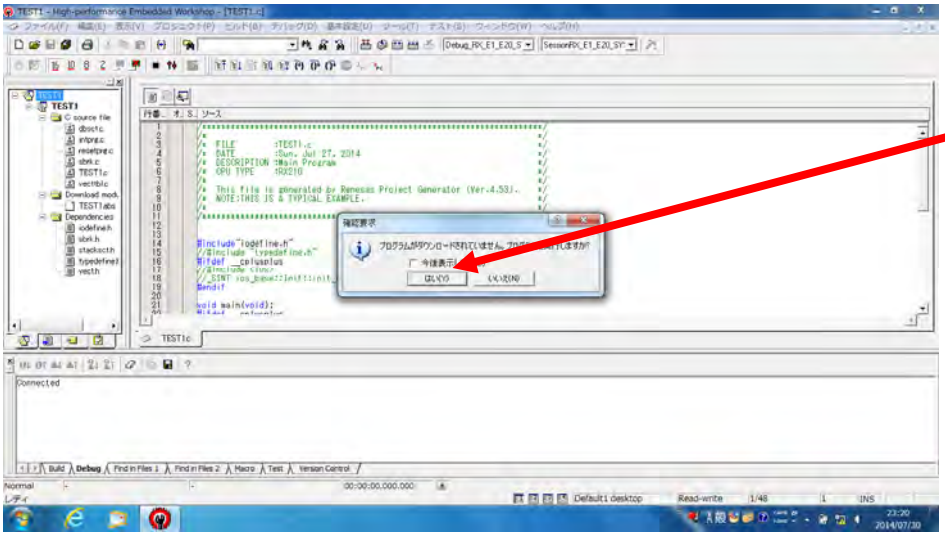
「Debug」を選択する。

今選択されているファイル内で、検索したいコマンド名を入力すると、コマンドを探し出し、表示することが出来る。

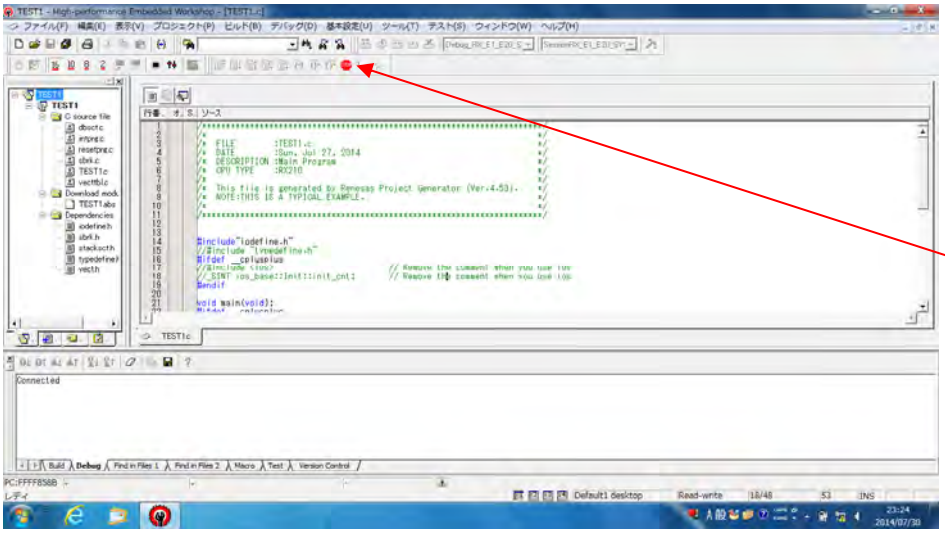


「デバッグ（D)」をマウス左クリックする

「実行（G)」をマウス左クリックする



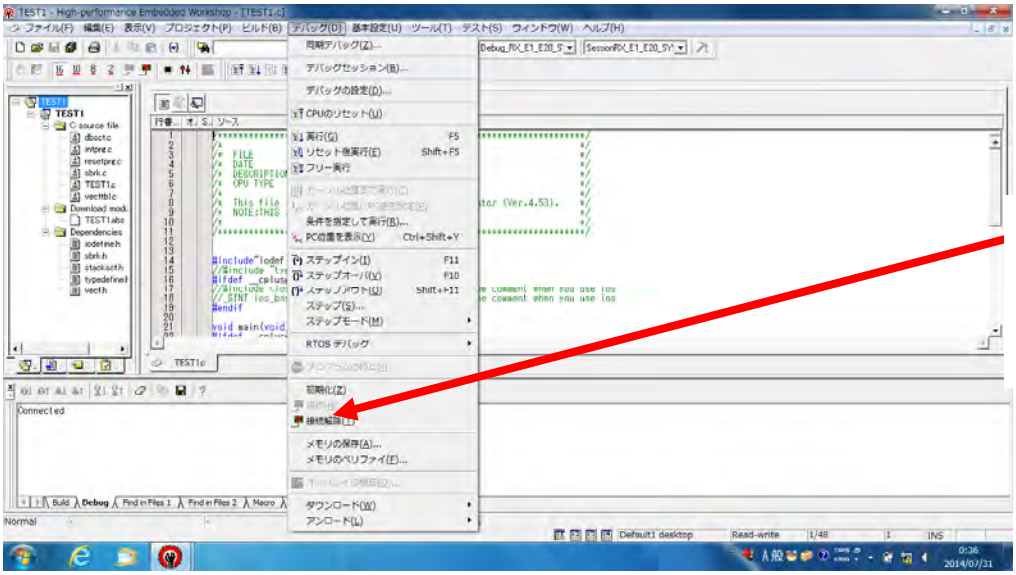
「はい（Y)」を
マウス左クリックする



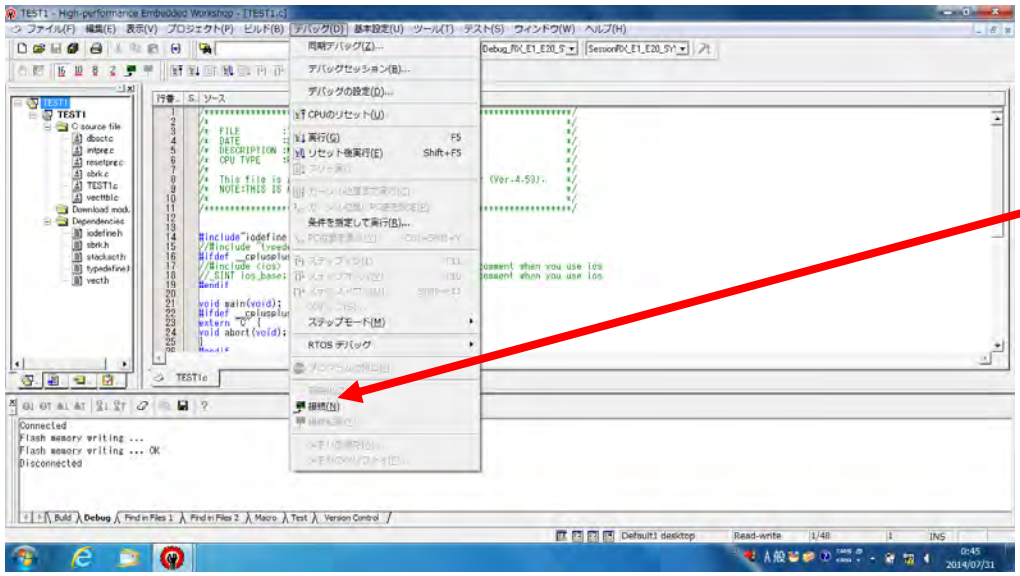
プログラムが実行され、
LED1（赤)、LED2(緑)の
約１秒間隔の交互点滅が繰り返し実行される。

「STOP」をマウス左クリックすると、プログラムの実行が
停止する。

E１エミュレータの「VCC」電源の ON、OFF 操作について：



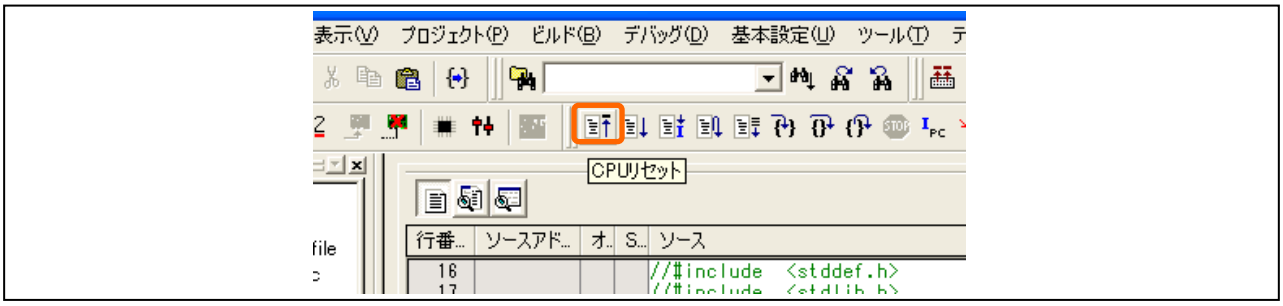
E1 エミュレータの VCC の
電源を、解除（OFF）するには、「デバッグ（D)」→「電源解除(T)」を、
マウス左クリックする。

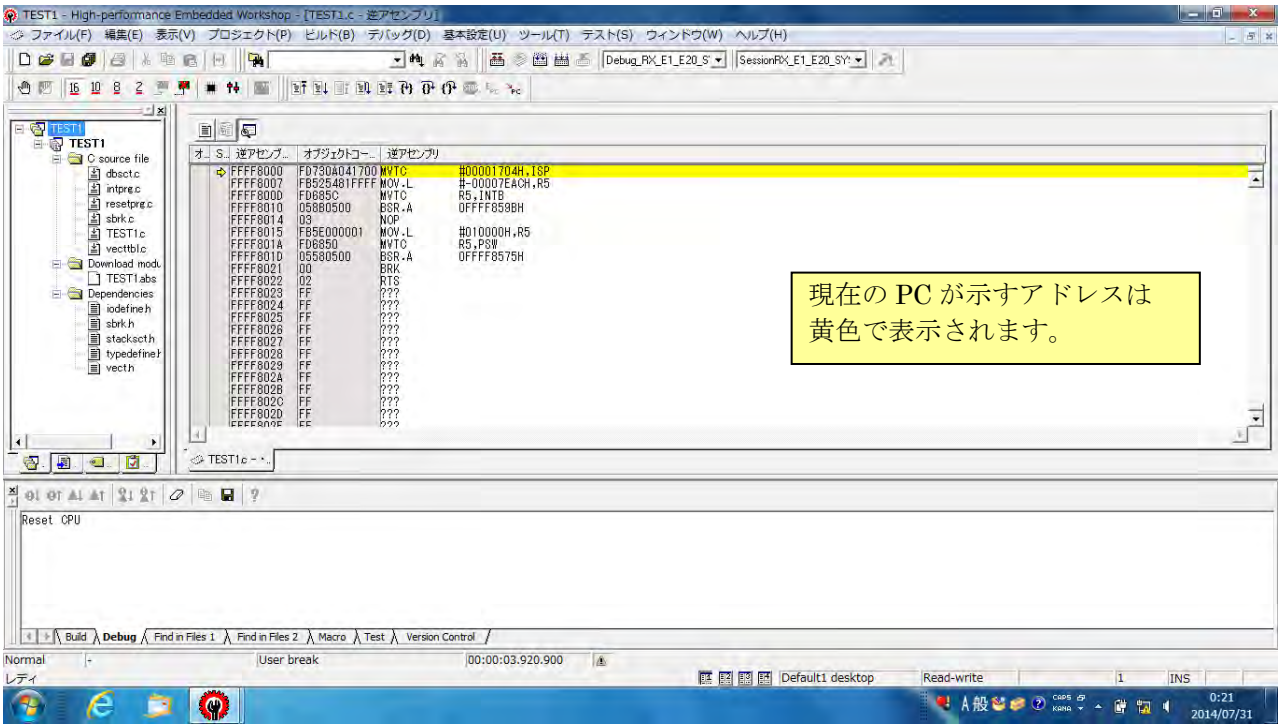


E1 エミュレータの VCC の電源を、接続（ON）するには、「デバッグ（D)」→「接続(N)」を、マウス左クリックする。

CPU のリセット

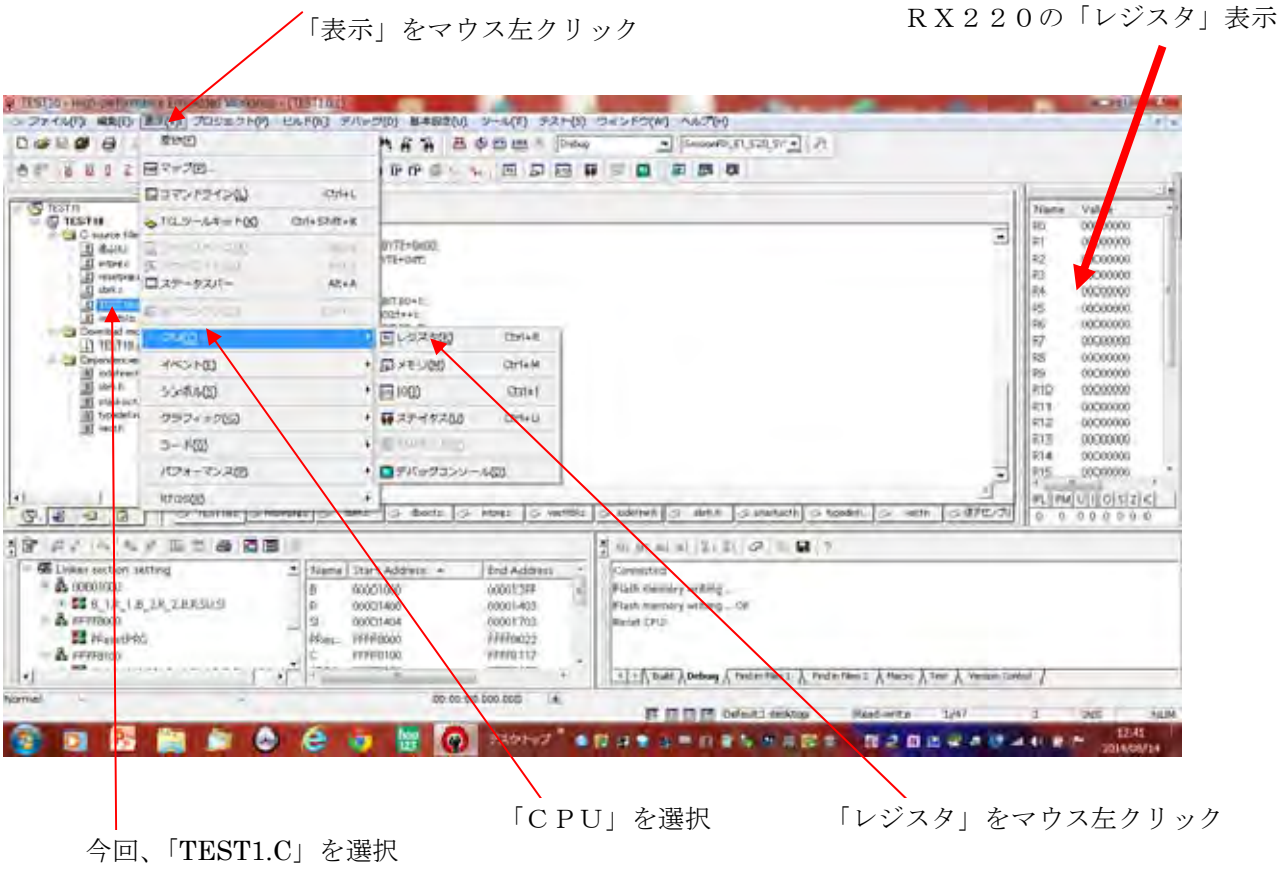
ツールバーの「CPU リセット」ボタンを押すことで、CPU がリセットされ、リセット開始番地のプログラムをウィンドウに表示します。現在の PC が示す行は黄色のラインで表示されます。





■レジスタの内容表示：

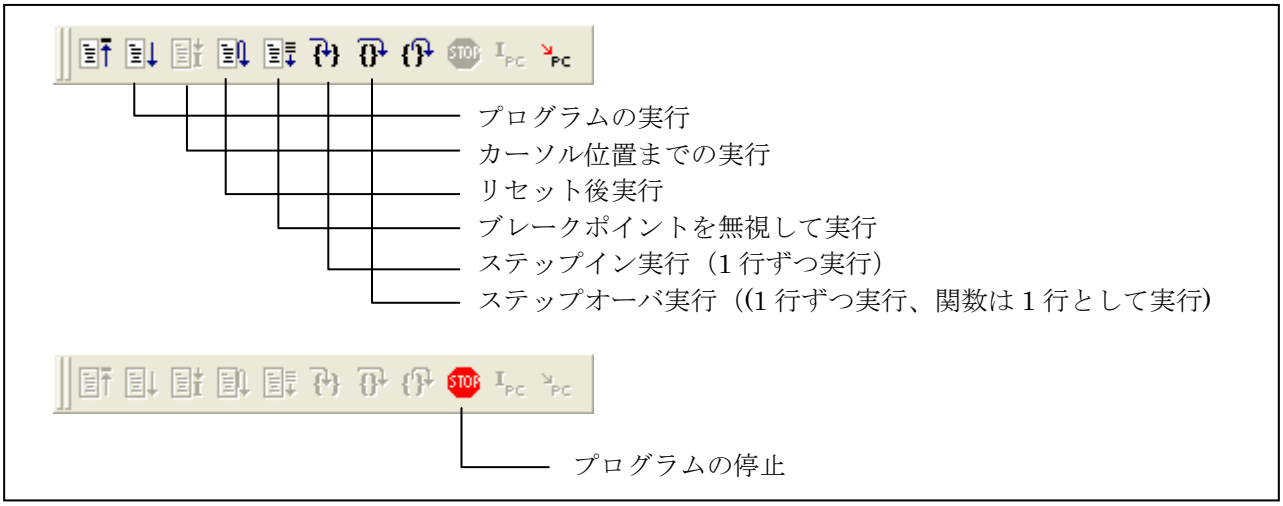
現在動作中のレジスタの内容も、表示することができます。
リアルタイムでプログラムのレジスタ動作状況把握ができ、プログラムの評価に威力を発揮します。



プログラムの実行と停止

プログラムの実行には以下の種類があります。CPU をリセットしてから実行するには、「リセット後実行」ボタンを押します。

ボードでの動作を確認できたら、「停止」ボタンを押してプログラムを停止させます。



そのほかのデバッグ機能

ここで紹介した機能のほかに、以下のようなデバッグ機能があります。これら機能については、E1 のマニュアルおよびアプリケーションノートを参照してください。

- レジスタ値の参照、設定
- メモリ値の参照、設定
- C 言語変数の参照、設定
- プログラムの指定アドレスでのブレーク（ソフトウェアブレーク、オンチップブレーク）
- 指定した条件でのプログラムの停止（オンチップブレーク）
- プログラム実行履歴の参照（トレース）

■オンチップデバッギングエミュレータ E1　導入ガイド
http://documentation.renesas.com/jpn/products/tool/apn/r20an0022jj0101_e1_intr.pdf

■オンチップデバッギングエミュレータ E1　トラブルシューティングガイド
http://documentation.renesas.com/jpn/products/tool/apn/r20an0045jj0100_e1e20_trbl.pdf

第 3 章　完

（参考）

■プログラム集：

1、「RX220 Base ボード」の LED1(赤)、LED2（緑）に於いて、
LED1 が 1 秒間点滅し終わると、LED2 が 1 秒間点滅する、
この繰り返し動作を永遠に繰り返す。

まず、　#include”iodefine.h”　を、下記プログラムの上に、追加記述してください。

追加記述する→　#include"iodefine.h"
//#include "typedefine.h"
#ifdef __cplusplus

下記プログラムを、void main(void) {　この部分に記載する　}　#ifdef __cplusplus

```
long int t;                          (変数 t を定義する)
PORTH.PODR.BYTE=0x00;（ポート H の 4 bit 分(16 進数)を”0”にクリアする）
PORTH.PDR.BYTE=0xff;　（ポート H の 4 bit 分(16 進数)を出力端子に設定）
while(1)　　（While で{} 囲まれたプログラムを、無限に繰り返し実行させる）
{
PORTH.PODR.BIT.B0=1;（ポート H の 0 bit 目から”1”を出力する。”1”=5V）
for(t=0;t<20000;t++);　（0 から 19999 までカウント UP させると、約 1 秒となる。時間稼ぎ）
PORTH.PODR.BIT.B0=0; (ポート H の 0 bit 目から”0”を出力する。”0”＝0 V=GND)
for(t=0;t<20000;t++);　（0 から 19999 までカウント UP させると、約 1 秒となる。時間稼ぎ）
PORTH.PODR.BIT.B1=1; (ポート H の 1bit 目から”1”を出力する。”1”=5V)
for(t=0;t<20000;t++);　（0 から 19999 までカウント UP させると、約 1 秒となる。時間稼ぎ）
PORTH.PODR.BIT.B1=0;　（ポート H の 1bit 目から”0”を出力する。”0”=0V=GND）
for(t=0;t<20000;t++);　（0 から 19999 までカウント UP させると、約 1 秒となる。時間稼ぎ）
}
```

＝Bit 命令応用によるプログラムの作成　(今回、追加するメインプログラム)＝

上記のメインプログラムは、Bit 命令使用により記述したプログラムです。

下記のメインプログラムは、Byte 命令使用により記述したプログラムです。

#include”iodefine.h” ← ①、プログラムの先頭に追加記載します。

②、下記メインプログラムを、{ } の中に記載していきます。

```
long int t;
PORTH.PODR.BYTE=0x00;
PORTH.PDR.BYTE=0xff;
while(1)
{
PORTH.PODR.BYTE=0x01;
for(t=0;t<20000;t++);
PORTH.PODR.BYTE=0x02;
for(t=0;t<20000;t++);
}
```

＝Byte 命令応用によるプログラムの作成 ＝

【コラム】

PORTH.PODR.BIT.B1=1（BYTE 命令）
PORTH.PODR.BYTE=0x01(Bit 命令)

２、「RX220 Base ボード」の LED1(赤)、LED2（緑）に於いて、
SW1 を ON 側にすると、LED1 が 1 秒間隔で点滅を繰り返し（LED2 は消灯)、
SW1 を OFF 側にすると、LED2 が 1 秒間隔で点滅を繰り返す（LED1 は消灯)。
SW1 の ON,OFF を判断して、LED1 なのか LED2 なのかを選択し、点滅させるプログラム。
回路的には、SW1 が ON 側の時、“０”。SW1 が OFF 側の時、“１”となります。

```
#include”iodefine.h”



long int t;
PORTH.PODR.BYTE=0x00;
PORTH.PDR.BYTE=0xfb;
while(1)
{
if(PORTH.PIDR.BIT.B2==1)
{
PORTH.PODR.BIT.B1=1;
for(t=0;t<20000;t++);
PORTH.PODR.BIT.B1=0;
for(t=0;t<20000;t++);
}
else
{
PORTH.PODR.BIT.B0=1;
for(t=0;t<20000;t++);
PORTH.PODR.BIT.B0=0;
for(t=0;t<20000;t++);
}
}
```

＝Bit 命令応用によるプログラムの作成 ＝

【２、の回答：完成プログラム】

```
#include"iodefine.h"          ←（今回、追加記載します）
//#include "typedefine.h"
#ifdef __cplusplus
//#include <ios>                                  // Remove the comment when you use ios
//_SINT ios_base::Init::init_cnt;          // Remove the comment when you use ios00-
#endif

void main(void);
```

```
#ifdef __cplusplus
extern "C" {
void abort(void);
}
#endif

void main(void)
{
long int t;    （変数 t を、定義する）
PORTH.PODR.BYTE=0x00;    （ポート H の 4 bit 分（0 bit～3 bit）を、”0”にクリアする）
PORTH.PDR.BYTE=0xfb;     （ポート H の 2bit 目は入力端子、0,1,3bit 目は、出力端子に設定）
while(1)     （While で{  } 囲まれたプログラムを、無限に繰り返し実行させる）
{
if(PORTH.PIDR.BIT.B2==1)    （ポート H の 2 bit 目の入力端子のデータが、”0”⇔”1”を判断させている）
{                                （ポート H の 2bit 目(入力)が”1”の時、処理は、本 {  } 内を実行する）
PORTH.PODR.BIT.B1=1;     （ポート H の 1bit 目から”1”を出力する。”1”=5V）
for(t=0;t<20000;t++);          （0 から 19999 までカウント UP させると、約 1 秒となる。時間稼ぎ）
PORTH.PODR.BIT.B1=0;     （ポート H の 1 bit 目から”0”を出力する。”0”=0V=GND）
for(t=0;t<20000;t++);          （0 から 19999 までカウント UP させると、約 1 秒となる。時間稼ぎ）
}
else                             （ポート H の 2bit 目(入力)が”0”の時、処理は、本 {  } 内を実行する）
{
PORTH.PODR.BIT.B0=1;    （ポート H の 0 bit 目から”1”を出力する。”1”=5V）
for(t=0;t<20000;t++);          （0 から 19999 までカウント UP させると、約 1 秒となる。時間稼ぎ）
PORTH.PODR.BIT.B0=0;    （ポート H の 0bit 目から”0”を出力する。”0”=0V=GND）
for(t=0;t<20000;t++);          （0 から 19999 までカウント UP させると、約 1 秒となる。時間稼ぎ）
}
}
}
#ifdef __cplusplus
void abort(void)
{
}
#endif
```

■朱記の部分が、今回追加したメインプログラムの部分です。

【コラム】

I/O の入出力を指定するには、**PDR（ポート方向レジスタ）**を使う。
I/O の出力データを格納するには、**PODR（ポート出力データレジスタ）**を使う。
I/O の端子の状態を反映するには、**PIDR（ポート入力データレジスタ）**を使う。

■「RX220　開発ソフト」（無償評価版）：

「RX220 開発ソフト」無償評価版（有効期間９０日間）は、今回添付の開発ソフト以外にも、
「ルネサスエレクトロニクス」のホームページ上から、ダウンロードすることができます。

●IDE、Compier（「RX220 用 C コンパイラ」に対応）に関しては、
【無償評価版】RX ファミリ用 C/C++コンパイラパッケージ V.1.02 Release 01 を
ダウンロードしてインストールしてください。
http://japan.renesas.com/products/tools/evaluation_software/downloads.jsp

●デバッグ（「E1 エミュレータソフト」に対応）に関しては、
E1 エミュレータを使用する場合
以下の URL から RX E1/E20 エミュレータデバッガ V.1.03.00 を
ダウンロードしてインストールしてください。
http://japan.renesas.com/request?SCREEN_ID=ViewGRSDownloadSearch&EXECUTE_ACTION=search&LAYER_ID=2964,167804

●RX220 への書き込み（「RX220 用フラッシュライタソフト」に対応）は、
フラッシュ開発ツールキットをダウンロードしてインストールしてください。
http://japan.renesas.com/request?SCREEN_ID=ViewGRSDownloadSearch&EXECUTE_ACTION=search&LAYER_ID=1050,167804&CATEGORY_ID=1

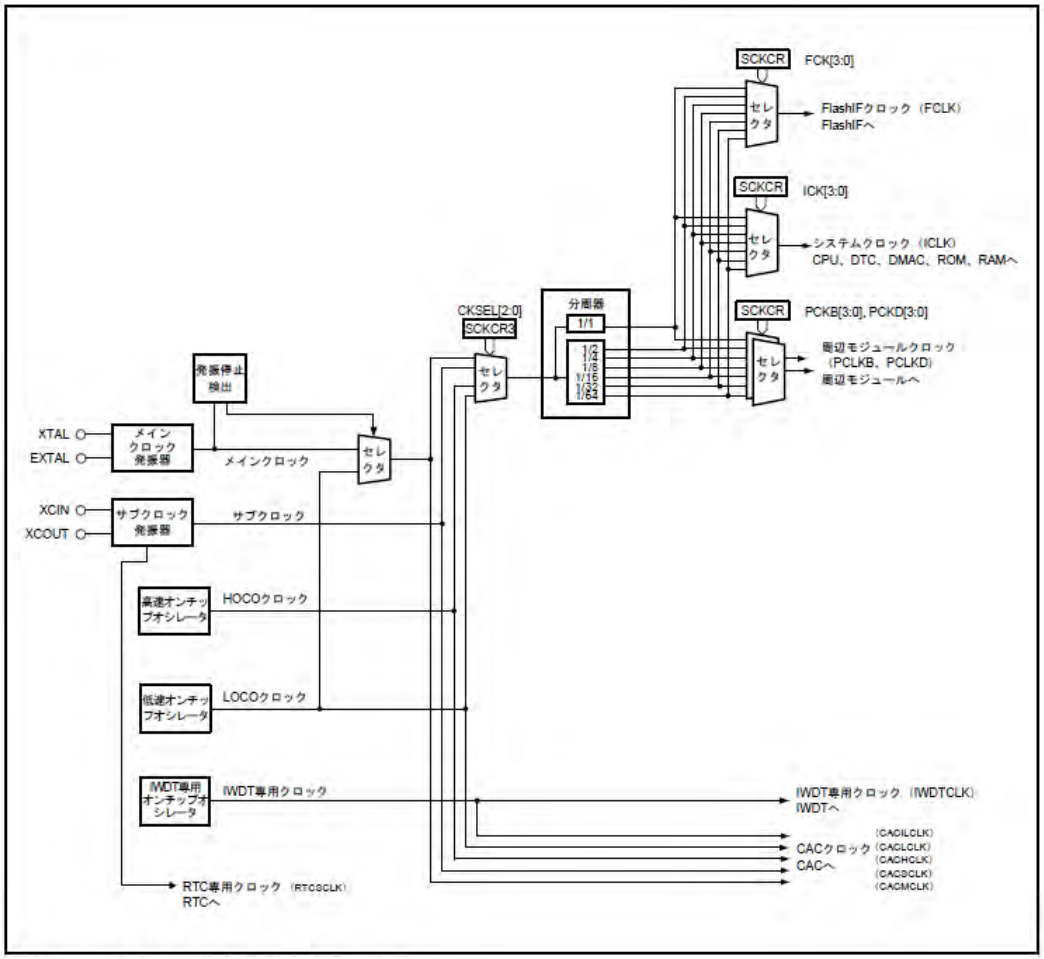
D、第４章：

「RX220 CPU ボード」に搭載の RX220 シリーズのマイコンは、きめ細かな低消費電力化を実現するために、ストップ、スタンバイモードの他に、幾つかのクロック発生回路を内蔵しています。

■RX220 シリーズ「クロック発生回路」の種類：

１、メインクロック発振器・・・・２０MHｚ(水晶振動子)
２、サブクロック発振器・・・・・３２．７６８KHｚ（水晶発振子）
３、高速オンチップオシレータ（HOCO）・・・32MHz/36.864MHz/40MHz/50MHz
４、**低速オンチップオシレータ（LOCO）・・・１２５KHｚ**（RESET 時は、本、オシレータが起動する）
５、IWDT 専用オンチップオシレータ・・・・１２５KHｚ

●「RX220 CPU ボード」の RESET 後は、「低速オンチップオシレータ（LOCO）」周波数１２５KHｚ優先で動作します。
そこで、メインクロック２０MHｚ、サブクロック３２．７６８KHｚ、高速オンチップオシレータ（HOCO）、IWDT 専用オンチップオシレータも選択切り替え出来るようにしなければなりません。
切り替え操作は、プログラムで行わなくてはなりません。



RX220 シリーズ「クロック発生回路」ブロック図

■低速オンチップオシレータ（LOCO）１２５KHｚから、
メインクロック２０MHｚ　及び　サブクロック３２．７６８KHｚに切り替える為には、
下記プログラムを「void main(void)」以降に、追加します。

下記プログラムは、メインクロック発振器、サブクロック発振器は有効とし、
「RX220 CPU ボード」を、メインクロック２０MHｚで、動作させるプログラムです。
サブクロック３２．７６８KHｚで動作させる場合は、「/*クロックソースの選択*/」を、
「//SYSTEM.SCKCR3.WORD = 0x0200;」と、「SYSTEM.SCKCR3.WORD = 0x0300;」に、
変更してください。（「//」が付くか、付かないかの変更です）

ここの指定がないと、コントロール
レジスタにデータが書き込まれない。

```
SYSTEM.PRCR.WORD = 0xa50b;          //クロックソース選択の保護の解除
//SYSTEM.OPCCR.BIT.OPCM = 0x02;     //動作モードの設定。低速/中速
SYSTEM.SCKCR.LONG = 0x00001111;     //マニュアル参照。

/*クロックソースの選択*/
SYSTEM.SCKCR3.WORD = 0x0200;        //大元のクロックにメインクロックを使用する。
//SYSTEM.SCKCR3.WORD = 0x0300;      //大元のクロックにサブクロックを使用する。
```

今回、
メインクロックを選択

今回、
メイン、サブクロック共、
有効にする

```
/*クロックの元栓の設定*/
SYSTEM.MOSCCR.BYTE = 0;     //メインクロック発振器動作
SYSTEM.SOSCCR.BYTE = 0;     //サブクロック発振器動作
//SYSTEM.LOCOCR.BYTE = 1;   //LOCO 動作
//SYSTEM.HOCOCR.BYTE = 1;   //HOCO 動作
//SYSTEM.ILOCOCR.BYTE =1;   //IWDT 専用オシレータ停止
```

各、発振器が選択できる

●今後、メインクロック、サブクロック他を起動させる場合には、
上記の紫文字のプログラム全てを、追加する必要があります。

それでは、LOCO１２５KHｚから、メインクロック２０MHｚに切り替えるプログラムを、
記載することにします。

■前回の「TEST1.ｃ」のプログラムを、下記内容に変更してください。

下記プログラムのシーケンスは、（品名：ＴＥＳＴ１．ｃ）

「RX220　Base ボード」の LED１（赤）、LED2(緑)を、メインクロック２０MHｚを使い、
１６bit タイマ機能で、正確に１秒の交互点滅を繰り返すプログラムです。

下記の黒文字は、「High-performance Embedded」が自動的に作成された部分です。
緑文字は、１秒タイマと入出力端子の制御を行っています。
ピンク文字は、LOCO125KHz からメインクロック 20MHz に、
切り替えるためのプログラムです。緑とピンク色は、今回追加します。

```
/************************************************************************/
/*                                                                      */
/*   FILE            :TEST1.c                                           */
/*   DATE            :Wed, Aug 11, 2014                                 */
/*   DESCRIPTION :Main Program                                          */
/*   CPU TYPE        :RX220                                             */
/*                                                                      */
/*   This file is generated by H.Terashita                              */
/*   NOTE：ＬＥＤ1 秒点滅（メインクロック 20MHz）.                      */
/*                                                                      */
/************************************************************************/
```

この緑色は、ヘッダーファイルです。
今回追加します。

```
#include"iodefine.h"
//#include "typedefine.h"
#ifdef __cplusplus
//#include <ios>                        // Remove the comment when you use ios
//_SINT ios_base::Init::init_cnt;       // Remove the comment when you use ios
#endif
```

この緑色は、16bit コンペアマッチタイマ（ＣＭＴ）
使用にて正確な１秒タイマを作ります。
今回追加します。

```
void main(void);
#ifdef __cplusplus
extern "C" {
void abort(void);
}
#endif

void timer(int msec){//10MCLK
        int cnt = 0;
        MSTP(CMT0)=0;                   //Wakeup CMT0, CMT1
        CMT0.CMCOR = 1249;              //8msec
        CMT0.CMCR.WORD = 0x0000;
        CMT.CMSTR0.BIT.STR0 = 1;
        while(1){
                while(CMT0.CMCNT != 0x00);
                cnt++;
                if(cnt == msec){
                        cnt = 0;
                        CMT.CMSTR0.BIT.STR0 = 0;
                        break;
                }
                while(CMT0.CMCNT == 0x00);
        }
}
```

このピンク記述部分が、メインクロック
20MHz に切り替えるプログラムです。
今回追加します。

```
void main(void)
```

```
{
SYSTEM.PRCR.WORD = 0xa50b;              //クロックソース選択の保護の解除
//SYSTEM.OPCCR.BIT.OPCM = 0x02;         //動作モードの設定。低速/中速
//SYSTEM.SCKCR.LONG = 0x00001412;       //マニュアル参照。

//SYSTEM.SCKCR.BIT.PCKD = 0x00;         //マニュアル参照。
SYSTEM.SCKCR.BIT.PCKB = 0x01;           //マニュアル参照。
//SYSTEM.SCKCR.BIT.BCK = 0x00;          //マニュアル参照。
SYSTEM.SCKCR.BIT.ICK = 0x00;            //マニュアル参照。
//SYSTEM.SCKCR.BIT.FCK = 0x00;          //マニュアル参照。

/*クロックソースの選択*/
SYSTEM.SCKCR3.WORD = 0x0200;            //大元のクロックにメインクロックを使用する。
//SYSTEM.SCKCR3.WORD = 0x0300;          //大元のクロックにサブクロックを使用する。

/*クロックの元栓の設定*/
SYSTEM.MOSCCR.BYTE = 0;         //メインクロック発振器動作
SYSTEM.SOSCCR.BYTE = 0;         //サブクロック発振器動作
//SYSTEM.LOCOCR.BYTE = 1;       //LOCO 動作
//SYSTEM.HOCOCR.BYTE = 1;       //HOCO 動作
//SYSTEM.ILOCOCR.BYTE =1;       //IWDT 専用オシレータ停止


PORTH.PODR.BYTE=0x00;
PORTH.PDR.BYTE=0xff;
while(1)
{
PORTH.PODR.BIT.B0=1;
PORTH.PODR.BIT.B1=0;
timer(1000);
PORTH.PODR.BIT.B0=0;
PORTH.PODR.BIT.B1=1;
timer(1000);                    //timer(ミリ秒)
}

}
#ifdef __cplusplus
void abort(void)
{

}
#endif
```

この緑色は、ポートHの端子状態の設定です。
今回追加します。

1000=1 秒
（例えば、2000 にすると２秒となる）
この緑色は、今回追加します。

次のシーケンスは、リアルタイムクロック(RTC)を使用したシーケンスとなります。
（品名：ＴＥＳＴ２．ｃ）

■シーケンス：
サブクロックの３２．７６８ＫＨｚを使用し、その６４Ｈｚカウンタの
２ＨｚビットをＬＥＤに反映させる事でＬＥＤを点滅させるシーケンスの
プログラムです。結果的に、ＬＥＤ１（赤）、ＬＥＤ２（緑）を
0.5 秒間隔で、交互に点滅表示させることになります。

```
/***********************************************************************/
/*                                                                     */
/*  FILE        :TEST2.c                                               */
/*  DATE        :Wed, Aug 20, 2014                                     */
/*  DESCRIPTION :Main Program                                          */
/*  CPU TYPE    :RX220                                                 */
/*                                                                     */
/*  This file is generated by H.Terashita                              */
/*  NOTE:リアルタイムクロック(RTC) NO.1                                */
/*                                                                     */
/***********************************************************************/


//#include "typedefine.h"
#include "iodefine.h"
#ifdef __cplusplus
//#include <ios>                        // Remove the comment when you use ios
//_SINT ios_base::Init::init_cnt;       // Remove the comment when you use ios
#endif

void main(void);
#ifdef __cplusplus
extern "C" {
void abort(void);
}
#endif

void main(void)
{
unsigned char t = 0;

//初期設定

/*クロックとカウントモードの設定*/
//RTC.RCR3.BIT.RTCDV = 0x01; //低 CL ドライブ能力 RTCDV がヘッダファイルに存在しない
```

青文字を、追加記載してください。

```
RTC.RCR3.BIT.RTCEN = 1; //サブクロック発振器動作開始

RTC.RCR2.BIT.START = 0;//RTC カウントストップ
while(RTC.RCR2.BIT.START != 0);

//RTC.RCR2.BIT.CNTMD = 1;//バイナリカウントモード CNTMD がヘッダファイルに存在しないため、OR で代入する。
RTC.RCR2.BYTE |= 0x80;//バイナリカウントモード

RTC.RCR2.BIT.RESET = 1;//RTC カウンタのリセット
while(RTC.RCR2.BIT.RESET == 1);//リセット完了の待ち


/*時刻の初期設定*/
RTC.RCR2.BIT.START = 0;//RTC カウントストップ
while(RTC.RCR2.BIT.START == 1);
RTC.RCR2.BIT.AADJE = 0;//自動補正機能禁止

RTC.RCR2.BIT.START = 1;//RTC カウントスタート
while(RTC.RCR2.BIT.START == 0);


RTC.RCR1.BYTE = 0x08;//RTC の全割り込み禁止、RTCOUT は 64Hz を出力
//RTC.RCR2.BIT.RTCOE = 1;//RTCOUT 出力許可

PORTH.PODR.BYTE=0x00;//LED ポートゼロクリア
PORTH.PDR.BYTE=0xff;//LED ポート初期設定

while(1)
{
t = RTC.R64CNT.BIT.F2HZ;// t に 2Hz ビットの値(1bit)を格納する
PORTH.PODR.BIT.B0=t;// t の値をそのまま LED に反映させる
PORTH.PODR.BIT.B1=(~t);// t の値に NOT をかけて、LED に反映させる
}
}

#ifdef __cplusplus
void abort(void)
{

}
#endif
```

次に、リアルタイムクロック（ＲＴＣ）を使ったカレンダー機能のシーケンスプログラムです。
（品名：ＴＥＳＴ３．ｃ）
ＲＥＳＥＴ（リセット）後、１秒のＬＥＤ１（赤）が６０回点滅（点灯は３０回）すると、
ＬＥＤ２（緑）の１分が点灯します。ＬＥＤ１（赤）が１２０回点滅（点灯は６０回）すると、
ＬＥＤ２（緑）の２分が消灯します。継続して動作していきます。ＬＥＤ２(緑)は、
奇数回で点灯、偶数回で消灯となります。本プログラムは、月、年まで管理できます。

```
/***********************************************************************/
/*                                                                     */
/*   FILE            :TEST3.c                                          */
/*   DATE            :Wed, Aug 20, 2014                                */
/*   DESCRIPTION :Main Program                                         */
/*   CPU TYPE        :RX220                                            */
/*                                                                     */
/*   This file is generated by    H.Terashita                          */
/*   NOTE:リアルタイムクロック（ＲＴＣ）　NO. 2                        */
/*                                                                     */
/***********************************************************************/


#include "iodefine.h"
//#include "typedefine.h"
#ifdef __cplusplus
//#include <ios>                        // Remove the comment when you use ios
//_SINT ios_base::Init::init_cnt;       // Remove the comment when you use ios
#endif

void main(void);
#ifdef __cplusplus
extern "C" {
void abort(void);
}
#endif

void main(void)
{
        unsigned char t1 = 0;
        unsigned char t2 = 0;


//初期設定
```

青文字を、追加記載してください。

```
/*クロックとカウントモードの設定*/
//RTC.RCR3.BIT.RTCDV = 0x01; //低 CL ドライブ能力 RTCDV がヘッダファイルに存在しない
RTC.RCR3.BIT.RTCEN = 1; //サブクロック発振器動作開始

RTC.RCR2.BIT.START = 0;//RTC カウントストップ
while(RTC.RCR2.BIT.START != 0);

RTC.RCR2.BYTE |= 0x00;//カレンダーカウントモード

RTC.RCR2.BIT.RESET = 1;//RTC カウンタのリセット
while(RTC.RCR2.BIT.RESET == 1);//リセット完了の待ち


/*時刻の初期設定*/
RTC.RCR2.BIT.START = 0;//RTC カウントストップ
while(RTC.RCR2.BIT.START == 1);
RTC.RSECCNT.BYTE = 0; //秒レジスタの初期値
RTC.RMINCNT.BYTE = 0;//分レジスタの初期値
RTC.RHRCNT.BYTE = 0;  //時レジスタの初期値
RTC.RWKCNT.BYTE = 0;  //曜日レジスタの初期値
RTC.RDAYCNT.BYTE = 0;//日レジスタの初期値
RTC.RMONCNT.BYTE = 0;          //月レジスタの初期値
RTC.RYRCNT.WORD = 0;  //年レジスタの初期値


//RTC.RSECCNT.BYTE = 0;//秒レジスタの初期値
//RTC.RSECCNT.BYTE = 0;//秒レジスタの初期値


RTC.RCR2.BIT.AADJE = 0;//自動補正機能禁止

RTC.RCR2.BIT.START = 1;//RTC カウントスタート
while(RTC.RCR2.BIT.START == 0);


RTC.RCR1.BYTE = 0x08;//RTC の全割り込み禁止、RTCOUT は 64Hz を出力
//RTC.RCR2.BIT.RTCOE = 1;//RTCOUT 出力許可

PORTH.PODR.BYTE=0x00;//LED ポートゼロクリア
PORTH.PDR.BYTE=0xff;//LED ポート初期設定

while(1)
{
t1 = RTC.RSECCNT.BYTE & 0x01;//  秒の 1 桁目の最下位ビットを格納する
t2 = RTC.RMINCNT.BYTE & 0x01;//  分の 1 桁目の最下位ビットを格納する

PORTH.PODR.BIT.B0=t1;//  秒の LSB を LED に反映させる
PORTH.PODR.BIT.B1=t2;//  分の LSB を LED に反映させる
```

```
}

}
#ifdef __cplusplus
void abort(void)
{

}
#endif
```

以上で、第４章を終わります。

第４章　 

以上で、説明を終わります。

―（完）―