

# VS-WRC003 での C 言語サンプルプログラム実行方法

ヴイストン株式会社

本説明書は、ロボットプロセッサ「VS-WRC003」を C 言語で開発する際に使用可能なルネサステクノロジより無料配布されている“High-performance Embedded Workshop”を用いて、サンプルプログラムを「VS-WRC003」に書き込む手順について解説したものです。HEW のインストールに関しましては、HEW 開発環境のインストールマニュアルをご覧ください。

## 1. HEW の起動とサンプルプロジェクトの読み込み

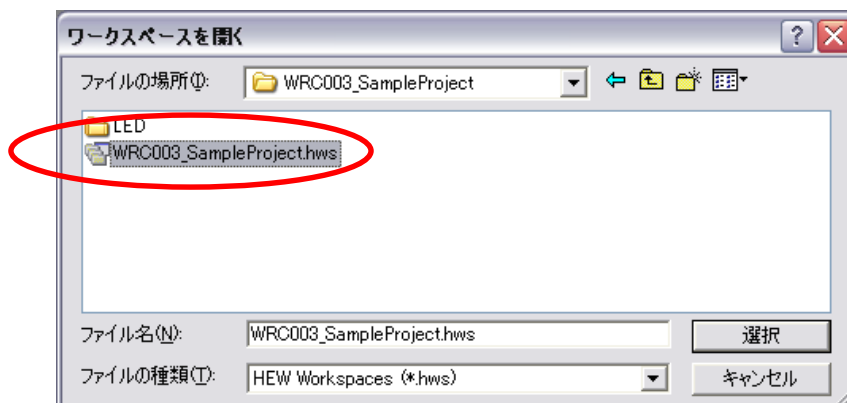
- (1) サンプルプロジェクト[WRC003\_SampleProject]を[C:¥Workspace](※)にコピーします。

(※ : windows が入っているドライブが C:の場合。Workspace フォルダは HEW インストール時に自動で生成されます)

- (2) Windows の[スタートメニュー]から[プログラム]>[Renesas]>[High-performance Embedded Workshop] >[High-performance Embedded Workshop]を起動します。

(※起動時に“ようこそ”画面が表示されますが、キャンセルしてください)

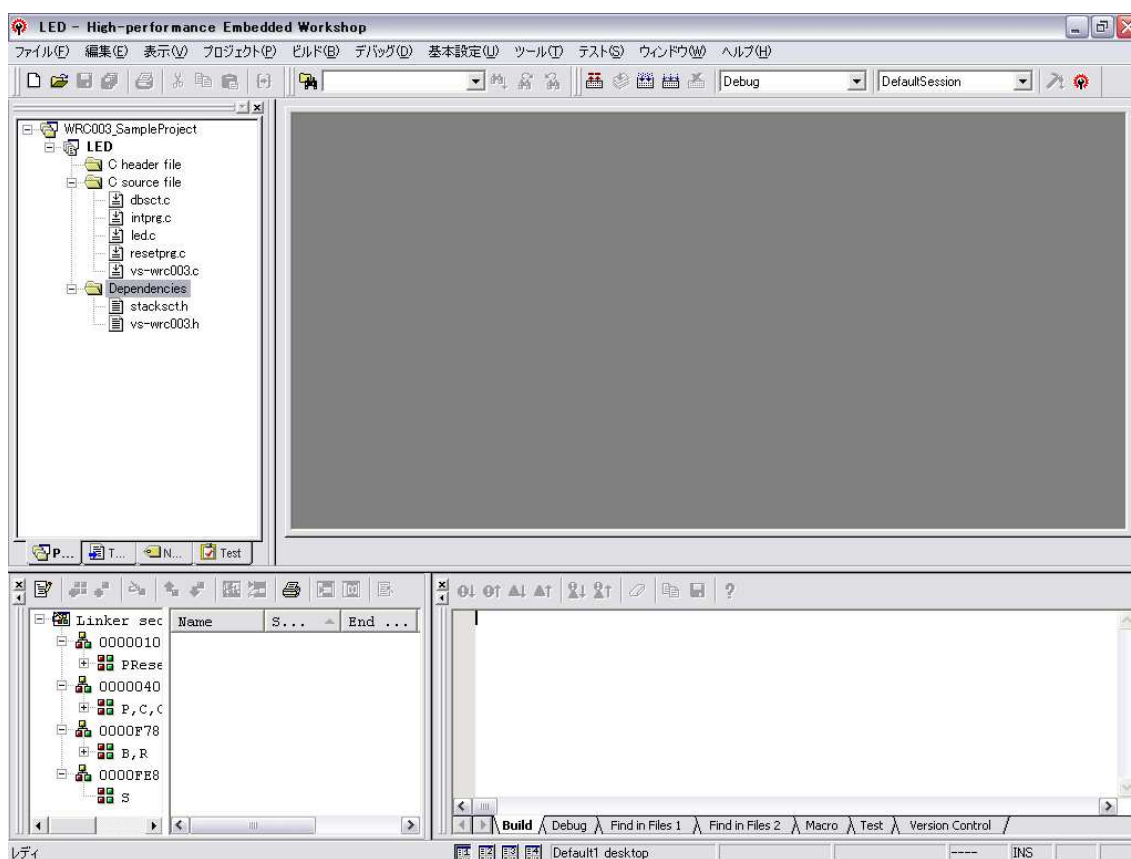
- (3) メニューの[ファイル]から[ワークスペースを開く]を選択し、サンプルプロジェクトフォルダ内にある[WRC003\_SampleProject.hws]を開きます。



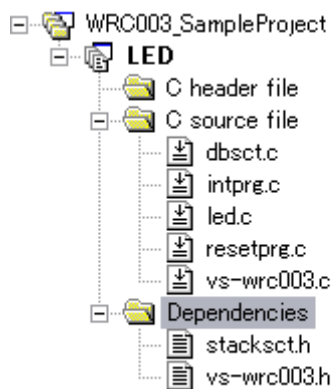
※プロジェクトを開く際にツールチェーンが異なる場合があります、その場合、変更を求められることがあります。 変更する場合は、すべて OK を押すことで変更できます。



(5)プロジェクトを開くと以下のような画面が表示されます。



プロジェクトには以下のファイルが含まれています。



※複数のプロジェクトがある場合、サンプルのワークスペース内の[LED]を右クリックし、プロジェクトをアクティブにします。

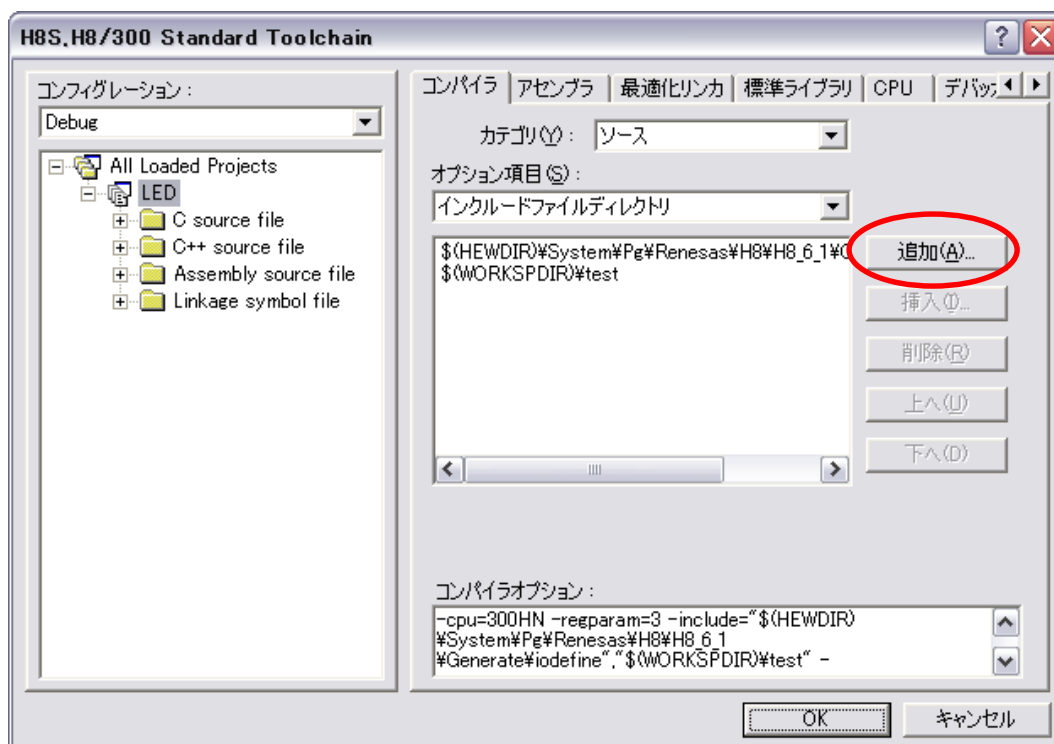
## 2. インクルードファイルディレクトリの設定と、ヘッダファイルの編集

- (1) 外部から持ってきたプロジェクトそのままでは、必要なヘッダファイル（拡張子.h のファイル）が読み込んでいませんので、インクルードファイルディレクトリを設定します。
- (2) メニューの[ビルド]内の[H8S,H8/300 Standard Toolchain..]からツールチェーン（コンパイラ、リンカなどのこと）の設定をします。



- (3) ツールチェーンの設定を開くと以下のようなウィンドが表示されます。

ここで、[コンパイラ]タブ内のカテゴリで[ソース]、オプション項目で[インクルードファイルディレクトリ]を選択し、[追加]を押します。

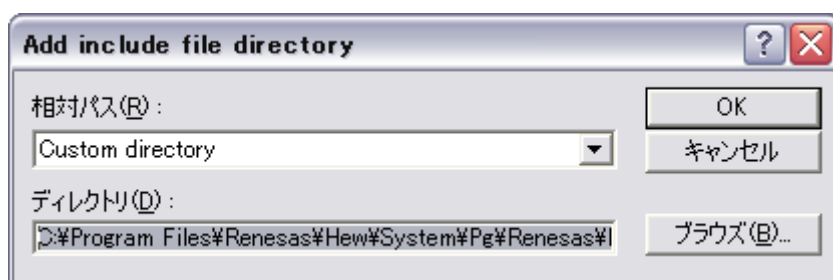


(4)インクルードディレクトリ追加用のウィンドが開いたら、相対パスに[Custom directory]を選択し、ディレクトリに各 CPU 用のインクルードファイルがある[iodefine フォルダ](※)のパスを設定し、[OK]を押します。

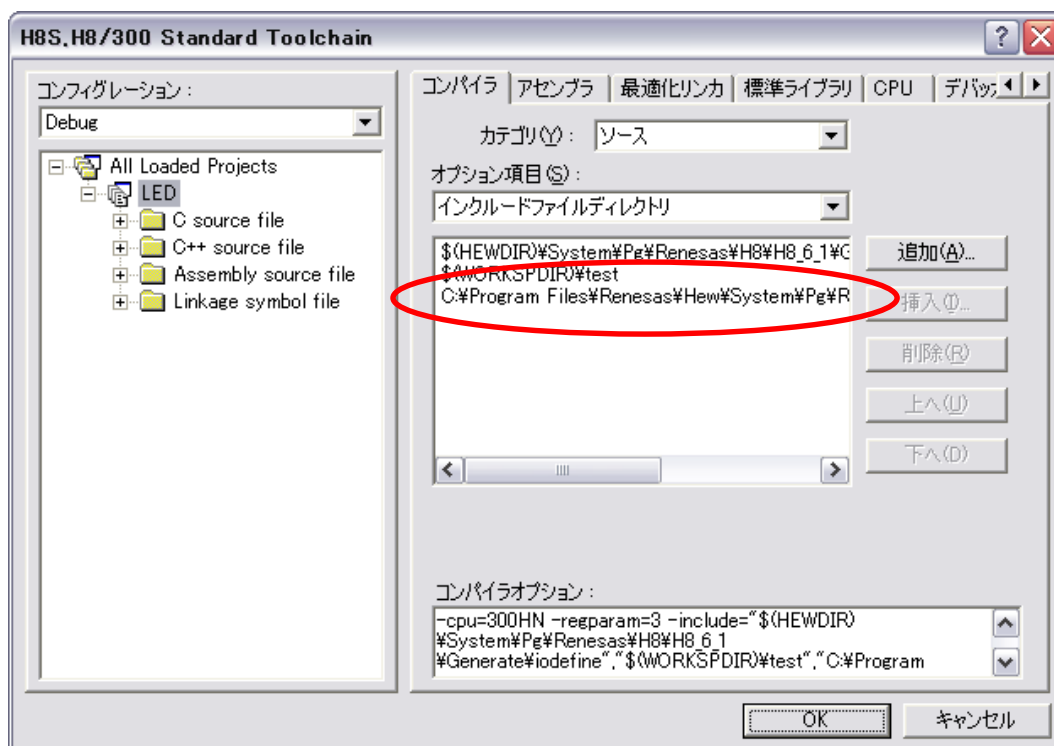
※ インストール時にインストール先のフォルダを変更していない場合、[iodefine フォルダ]は以下のようになります。(windows が入っているドライブが C:の場合)

[C:¥Program Files¥Renesas¥Hew

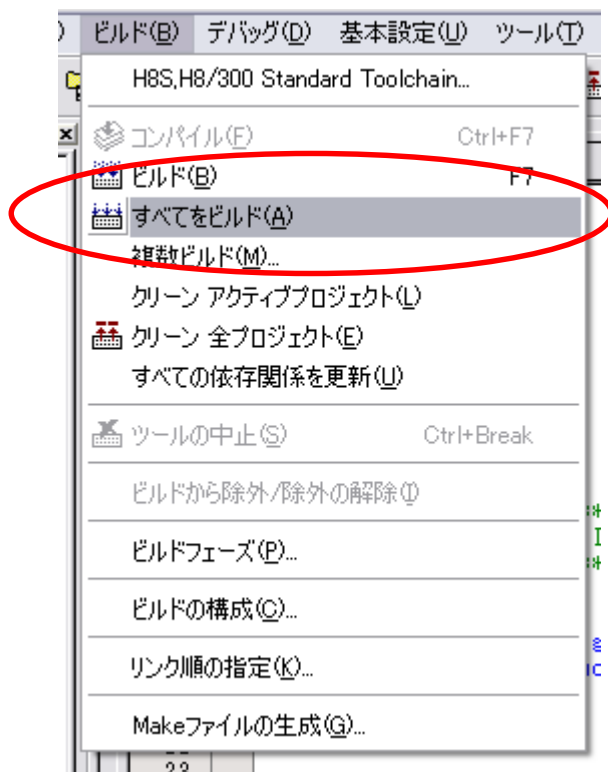
¥System¥Pg¥Renesas¥H8¥H8\_6\_2¥Generate¥iodefine]



(5)設定したフォルダが追加されていたら[OK]をおして、ツールチェインの設定を終了します。



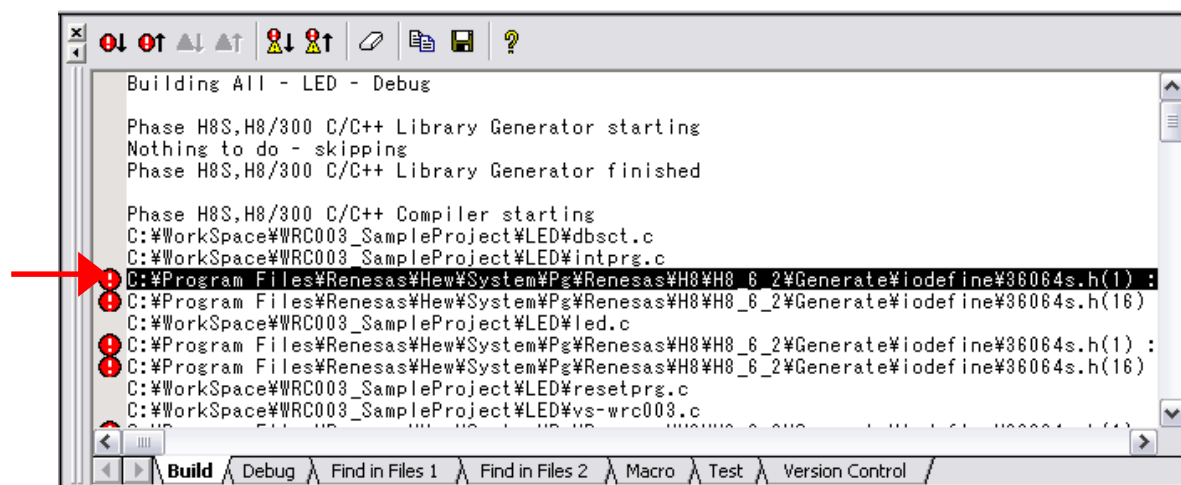
(6)ここで、一旦ビルドをします。ビルドはメニューの[ビルド]内の[すべてをビルド]で行えます。(2回目以降は[ビルド]を選択 or F7 キーを押すでもビルドできます)



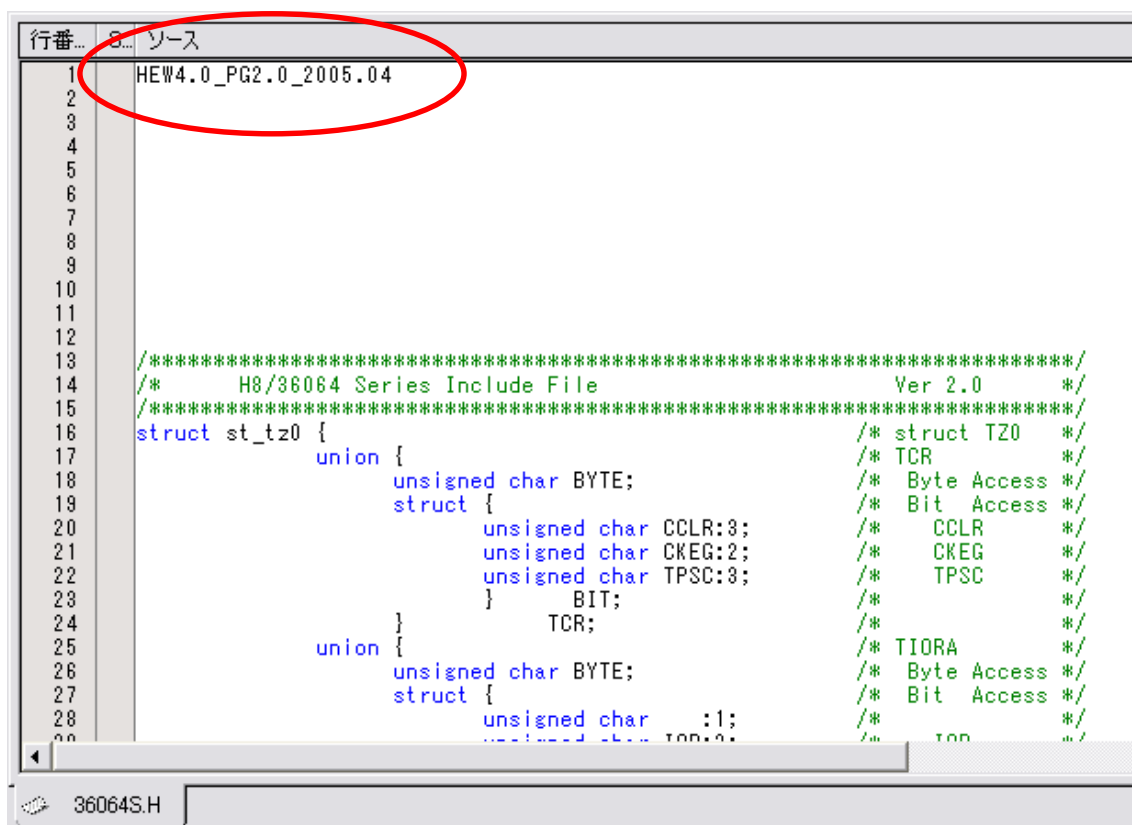
(7)ビルドを行うと画面右下(アウトプットウィンド)にエラーが表示されます。

エラーの原因がどこかを探したいときは、エラーの行をダブルクリックすると画面上のエディタにエラー部分が表示されます。

ここでは、エラーのうち[(HEW のインストールフォルダ)...¥iodefines¥36064s.h(1):\*\*\*\*]  
(一番初めに出力されたエラー) をダブルクリックして表示させます。



(8) エディタに 36064s.h が表示されると思います。この 1 行目にコメントアウトされていない一文がありますので、先頭に[/]を入力（コメントアウト）するか、1 行目を削除してください。

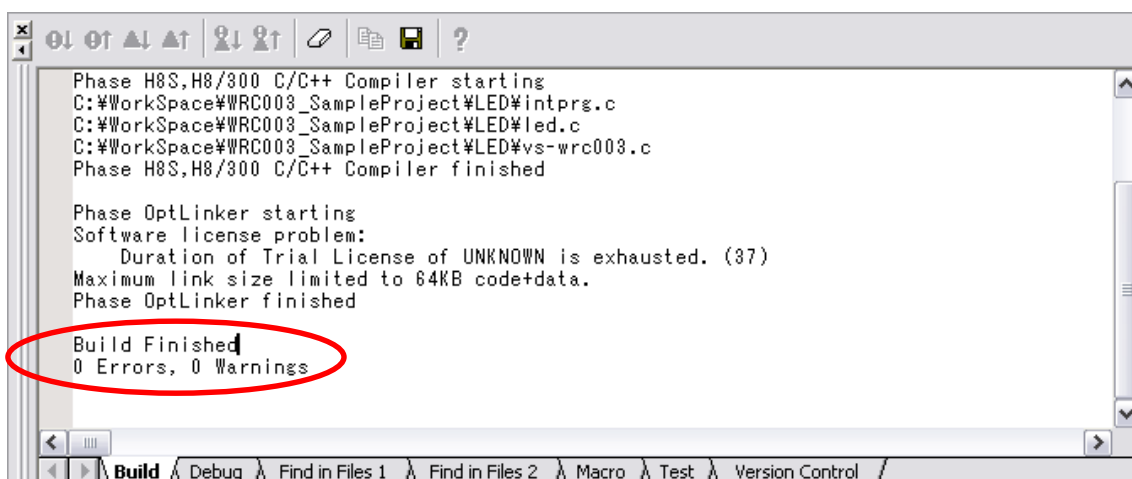


The screenshot shows a code editor window with a line number column on the left. Line 1 is circled in red and contains the text `HEW4.0_PG2.0_2005.04`. The rest of the file contains C code for a structure `st_tz0` with unions for TCR and TIORA, and various bit fields.

```
1 HEW4.0_PG2.0_2005.04
2
3
4
5
6
7
8
9
10
11
12
13
14 /******
15 /*      H8/36064 Series Include File                      Ver 2.0      */
16 /******
17 struct st_tz0 {                                           /* struct TZ0 */
18     union {                                              /* TCR */
19         unsigned char BYTE;                             /* Byte Access */
20         struct {                                         /* Bit Access */
21             unsigned char CCLR:3;                       /* CCLR */
22             unsigned char CKEG:2;                       /* CKEG */
23             unsigned char TPSC:3;                       /* TPSC */
24             BIT;                                          /* */
25         } TCR;                                           /* */
26     }
27     union {                                              /* TIORA */
28         unsigned char BYTE;                             /* Byte Access */
29         struct {                                         /* Bit Access */
30             unsigned char :1;                            /* */
31             unsigned char :1;                            /* */
32         } TIORA;                                         /* */
33     }
34 }
```

(9) コメントアウトし、ファイルを保存したらビルドします。

エラーがなく、ビルドが完了している場合は[**Build Finished 0 Errors, 0 Warnings**]と表示されます。これで、VS-WRC003 に書き込むためのファイル（.mot ファイル）が生成されました。



The screenshot shows a build log window with a toolbar at the top. The log text shows the compiler and linker phases. The final status 'Build Finished 0 Errors, 0 Warnings' is circled in red.

```
Phase H8S,H8/300 C/C++ Compiler starting
C:\Workspace\WRC003_SampleProject\LED\intprg.c
C:\Workspace\WRC003_SampleProject\LED\led.c
C:\Workspace\WRC003_SampleProject\LED\vs-wrc003.c
Phase H8S,H8/300 C/C++ Compiler finished

Phase OptLinker starting
Software license problem:
Duration of Trial License of UNKNOWN is exhausted. (37)
Maximum link size limited to 64KB code+data.
Phase OptLinker finished

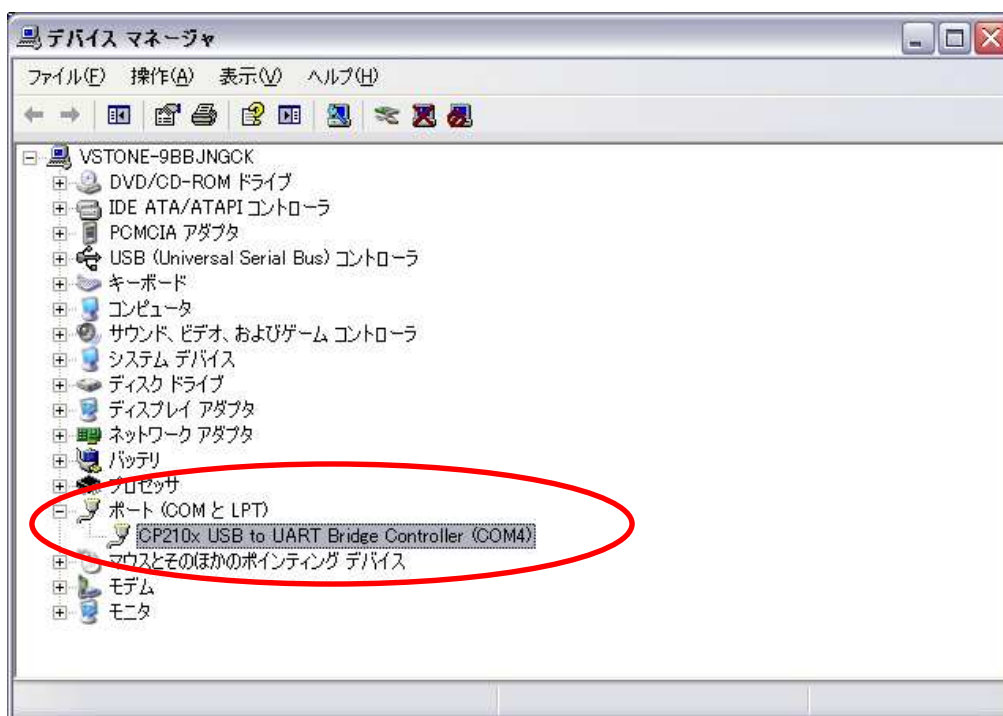
Build Finished
0 Errors, 0 Warnings
```

### 3. VS-WRC003 へのプログラムの書き込み

- (1) VS-WRC003(以下 CPU ボード)には USB シリアル変換 IC が内蔵されています。ですので、書き込みには COM ポートを使用します。まず、CPU ボードと PC を USB ケーブルで接続します。この状態で、**デバイスマネージャ**(※1)を開き **COM ポートの番号**を確認します。(下の画像では COM4) COM9 以降の場合は 19 ページの記載に従って番号を COM8 より小さいものに変更してください。

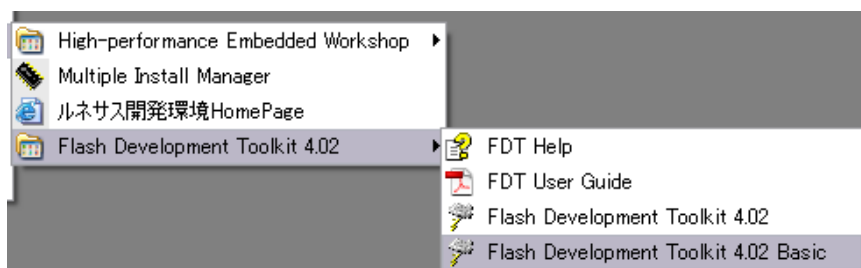
(※1: デバイスマネージャはマイコンピュータのプロパティで[ハードウェア]タブ内の[デバイスマネージャ]を押すと表示されます。)

(※2CP210x のドライバがインストールされていない (デバイスマネージャで以下のような表示がない) 場合は、Beauto ビルダー-NEO のインストールを行ってください)



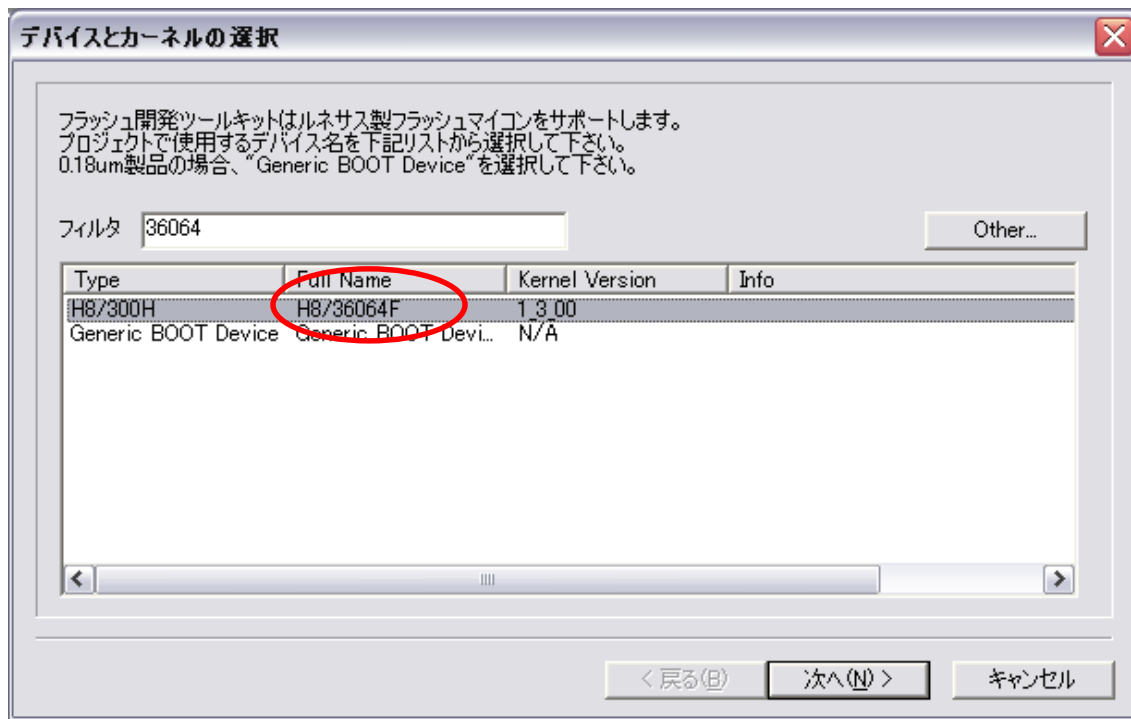
- (2) 次に、書き込み用のツールを開きます。

スタートメニューの[プログラム]>[Renesas]>[Flash Development Toolkit \*.\*]>[ Flash Development Toolkit \*.\* Basic]を開きます。(\*.\*はバージョンの番号)



(3) 初回起動時のみ、以下のオプション画面が表示されます。

CPU ボードが PC に接続されている状態で、フィルタに[36064]と入力し、一番上の[H8/36064F]を選択し、[次へ]を押します。



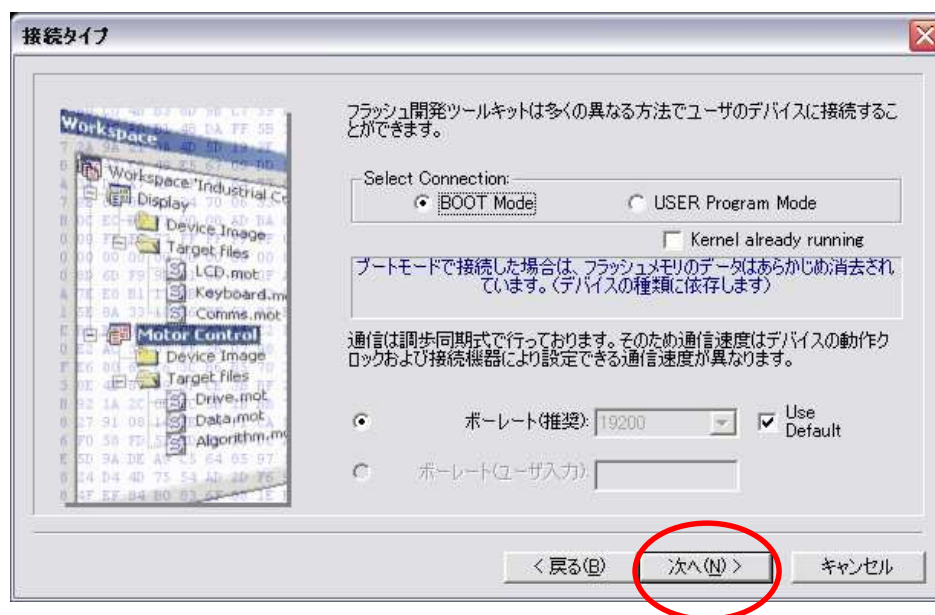
(4) [Select port:] にデバイスマネージャで調べた COM ポートを設定し次へを押します。



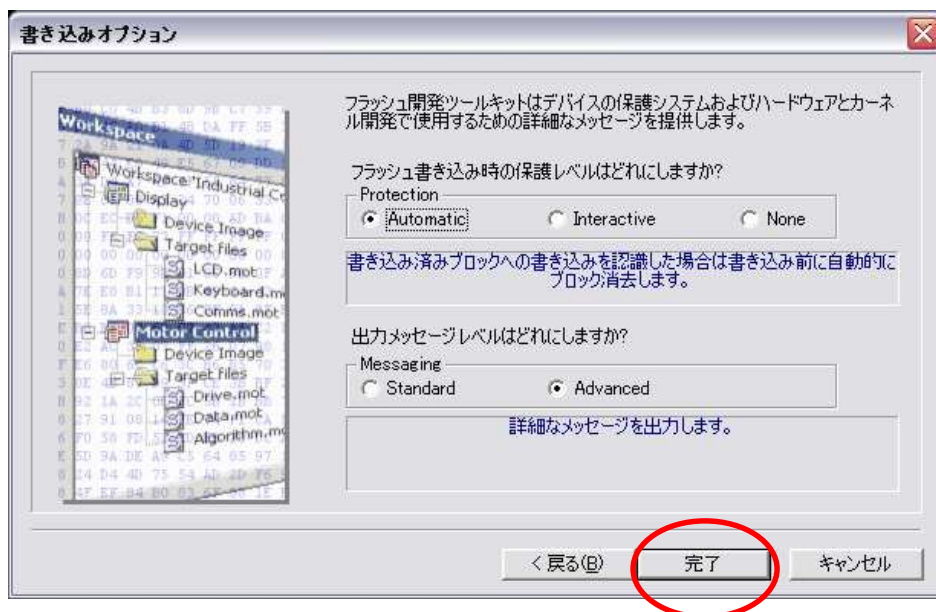
(5)[入力クロック]を[14.7456]に設定し、[次へ]を押します。



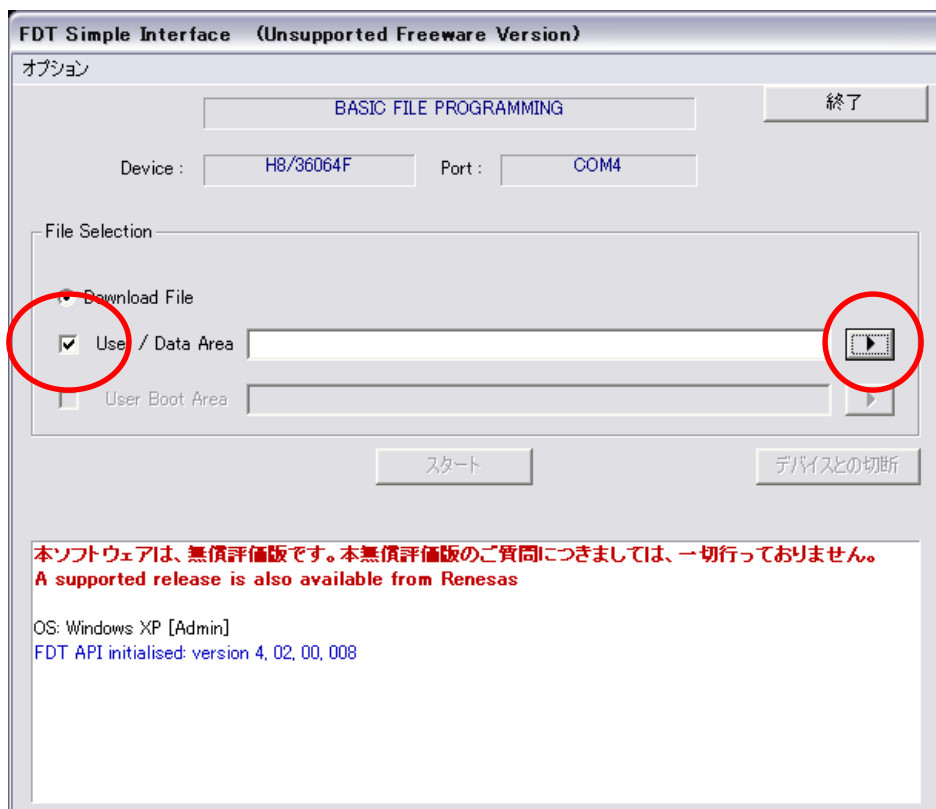
(6)そのままの設定で[次へ]を押します。



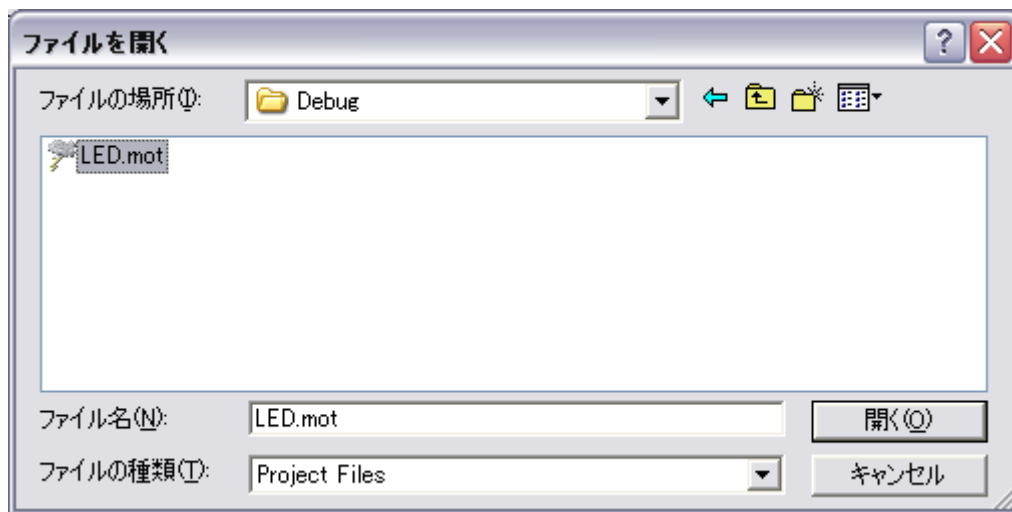
(7)そのままの設定で[完了]を押します。



(8)設定が完了したら、[User/DataArea]をチェックし、右向きの三角のボタンを押し、書き込むファイルを選択します。



(9) 書き込みファイルの[(プロジェクト保存先)¥WRC003\_SampleProject¥LED¥Debug]  
内の[LED.mot]を開きます。

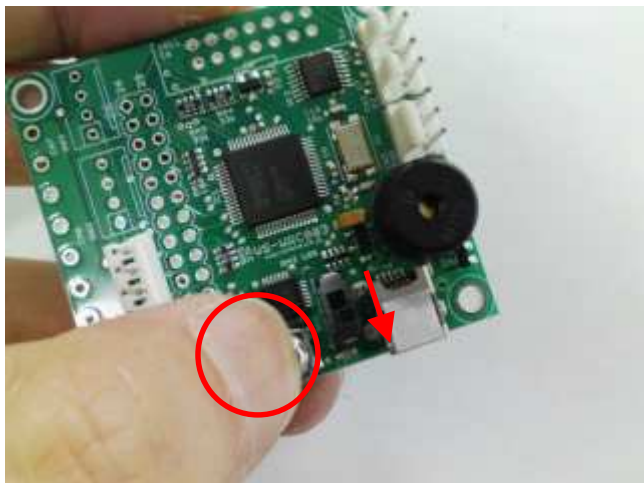


(10)CPU ボードから一度 USB ケーブルを抜き、以下の手順で書き込んでください。

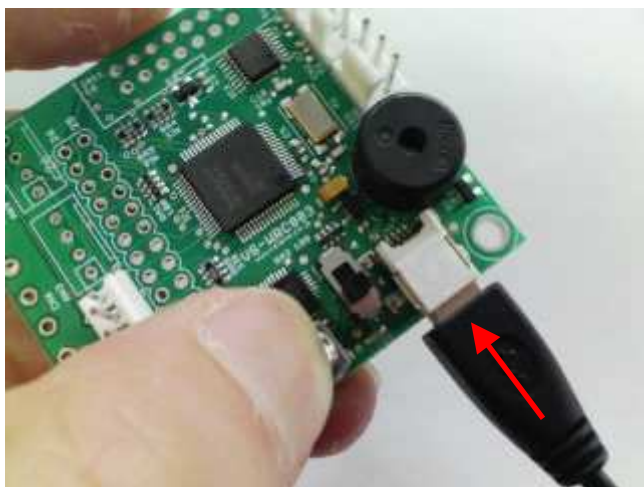
・ 書き込みの手順

①電源スイッチを切ります。

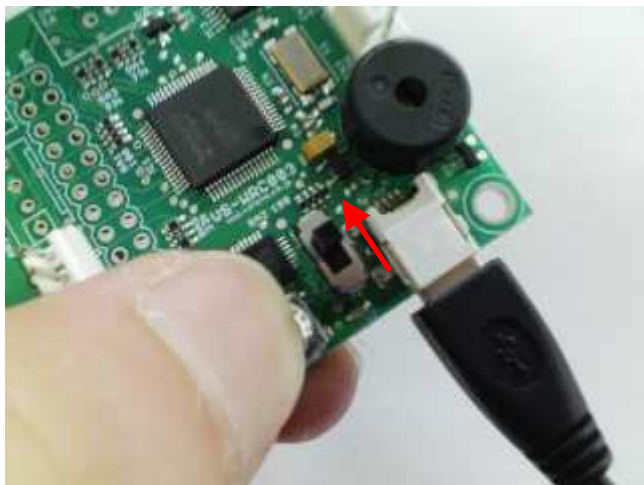
②スイッチを押します。



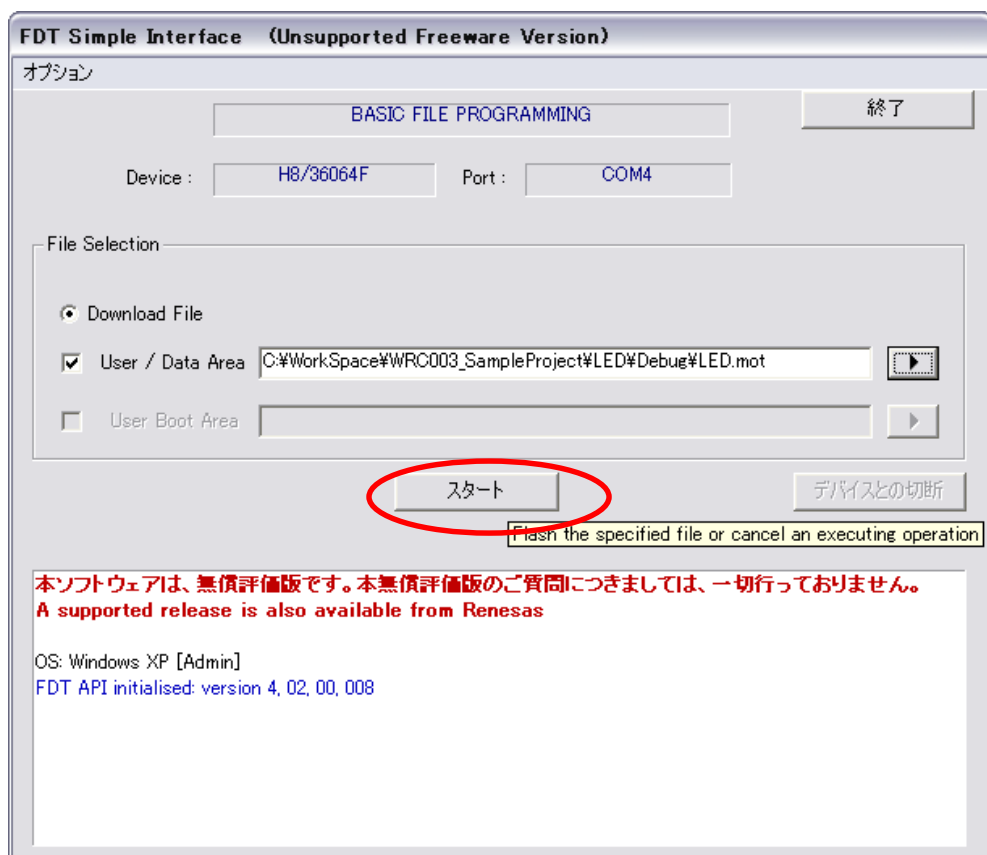
③スイッチを押したまま USB コネクタを差し込みます。



④スイッチを押したまま、電源スイッチを入れます。

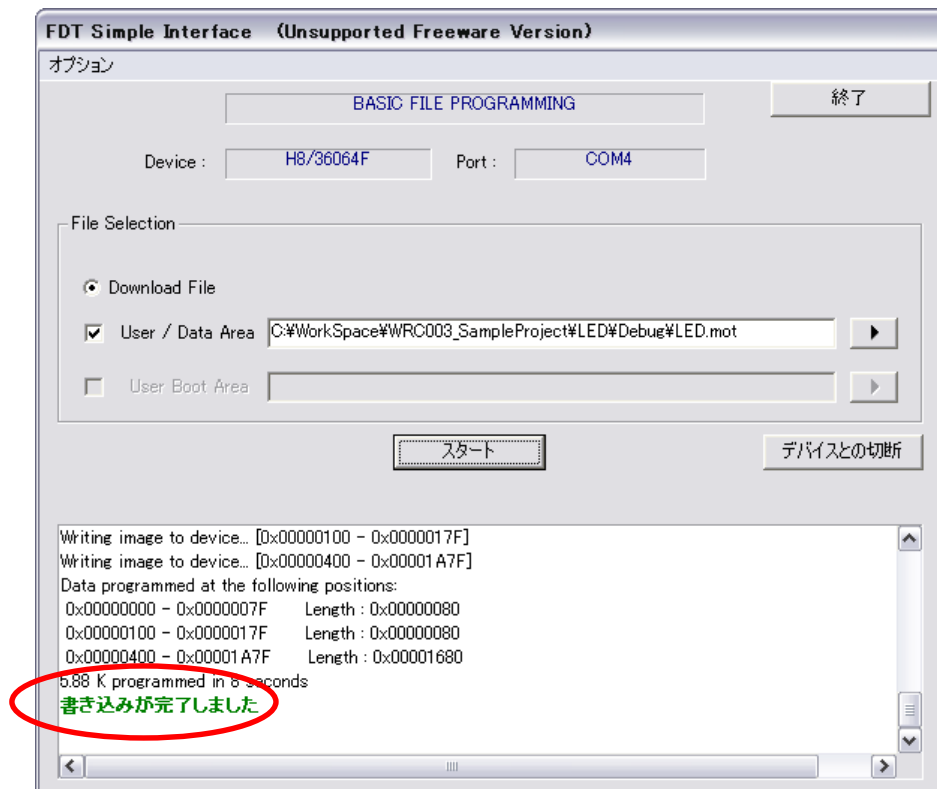


④この状態のまま、FDT のスタートボタンを押します。  
書き込み中はスイッチから手を離さないでください。



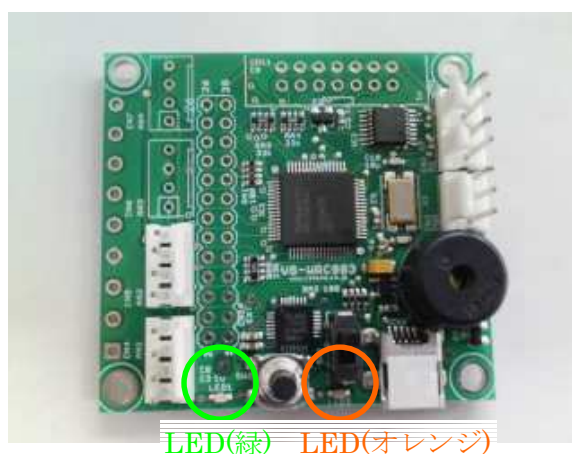
(11)下のように[書き込みが完了しました]と表示されれば、書き込み完了です。

表示されない場合は[デバイスと切断]を押した後、再度 (10)の手順で書き込みを行ってください。



(12)CPU ボードの電源を入れると、書き込んだプログラムが実行されます。

きちんと書き込めている場合、CPU 上のオレンジと緑の LED が交互に点滅します。



※ 設定は次回起動時も保存されていますので、(10)～(11)の手順のみで書き込むことができます。

※ 違うファイルを書き込みたい場合は、(8)～(10)と同じ手順で行ってください。

#### 4. LED 点滅プログラム

サンプルの LED 点滅プログラムは、関数 LED0、Wait0などを使って LED を制御しています。これらの関数はヘッダファイル VS-WRC003.h 内で定義され、プログラム内で利用することが出来ます。

LED 点滅プログラムの main0関数を以下に示します。プログラムを実行する場合、まずこの main0関数から実行されます。LED 点滅プログラムでは、main0は led.c 内にあります。

```
0:      void main(void)
1:      {
2:          //制御周期の設定[単位：Hz 範囲：30.0~]
3:          const BYTE MainCycle = 60;
4:          Init((BYTE)MainCycle);          //CPU の初期設定
5:          InitSci3(CBR_115200,even,1);    //シリアル通信の設定
6:          BuzzerSet(0x80,0x80);           //ブザーの設定
7:          //無限ループ
8:          while(1){
9:              LED(1);          //緑の LED 点灯
10:             Wait(1000);       //1000msec 待つ
11:             LED(2);          //オレンジの LED 点灯
12:             Wait(1000);       //1000msec 待つ
13:         }
14:     }
```

main 関数の各行について説明します。

0：関数の宣言

2～6：各機能の初期設定

ブザー、シリアル通信などの設定をしています。これらについては、各機能を使用するときに再度説明します。

8：メインループ

while 文で書かれた無限ループ内に実行したい処理を記述します。

9～12：実行する処理

上記のプログラムの場合、LED0関数と Wait0関数を利用して LED を交互に点滅させています。各関数について以下で説明します。

○void LED(BYTE LedOn);

LED0関数は CPU ボード上にある 2 つの LED の点灯、消灯を、引数[LedOn]に数値をあたえることで制御することができます。

[LedOn]に与える数値は 2 進数で bit0 が緑、bit1 がオレンジの LED に対応し、1 のとき点灯、0 のとき消灯します。

各数値を指定した時の LED の状態は以下のとおりです。

数値 (10 進数)	bit1	bit0	LED (オレンジ)	LED (緑)
0	0	0	消灯	消灯
1	0	1	消灯	点灯
2	1	0	点灯	消灯
3	1	1	点灯	点灯

○void Wait(int msec);

Wait 0関数は、引数[msec]で与えた時間(単位は msec)だけ待つ関数です。

たとえば、LED 点滅プログラムの中にある Wait(1000)は 1 秒 (=1000msec) だけ待つ処理になります。

### 例題①

スイッチが押されたときに緑の LED を光らせるプログラムを作ってみましょう。  
スイッチが押されているか、いないかの状態の取得には getSW0関数を利用します。

#### ○ BYTE getSW0

getSW0関数を実行すると、スイッチが押されているかどうかを戻り値で判定できます。  
押されている場合は1、押されていない場合は0を返します。

getSW0関数の戻り値を if、else 文で判定し、LED を点灯、消灯することで実現できます。  
Main 関数は以下のようになります。

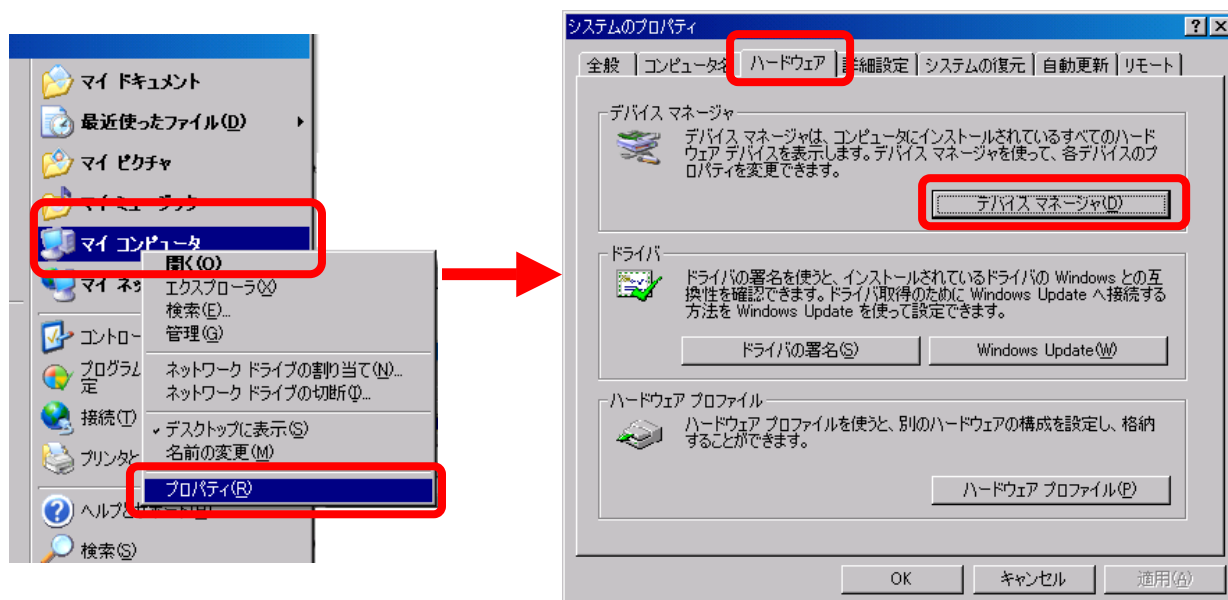
```
0:    void main(void)
1:    {
2:        //制御周期の設定[単位：Hz 範囲：30.0~]
3:        const BYTE MainCycle = 60;
4:        Init((BYTE)MainCycle);           //CPU の初期設定
5:        InitSci3(CBR_115200,even,1);     //シリアル通信の設定
6:        BuzzerSet(0x80,0x80);           //ブザーの設定
7:        //無限ループ
8:        while(1){
9:            if(getSW0 == 1){              //スイッチが押されていたら
10:                LED(1);                  //緑の LED 点灯
11:            }
12:            else{                         //押されていなかったら
13:                LED(0);                  //すべての LED 消灯
14:            }
15:        }
16:    }
```

## トラブル対応：FDT を使用した VS-WRC003 へのプログラム書き込みが失敗する場合

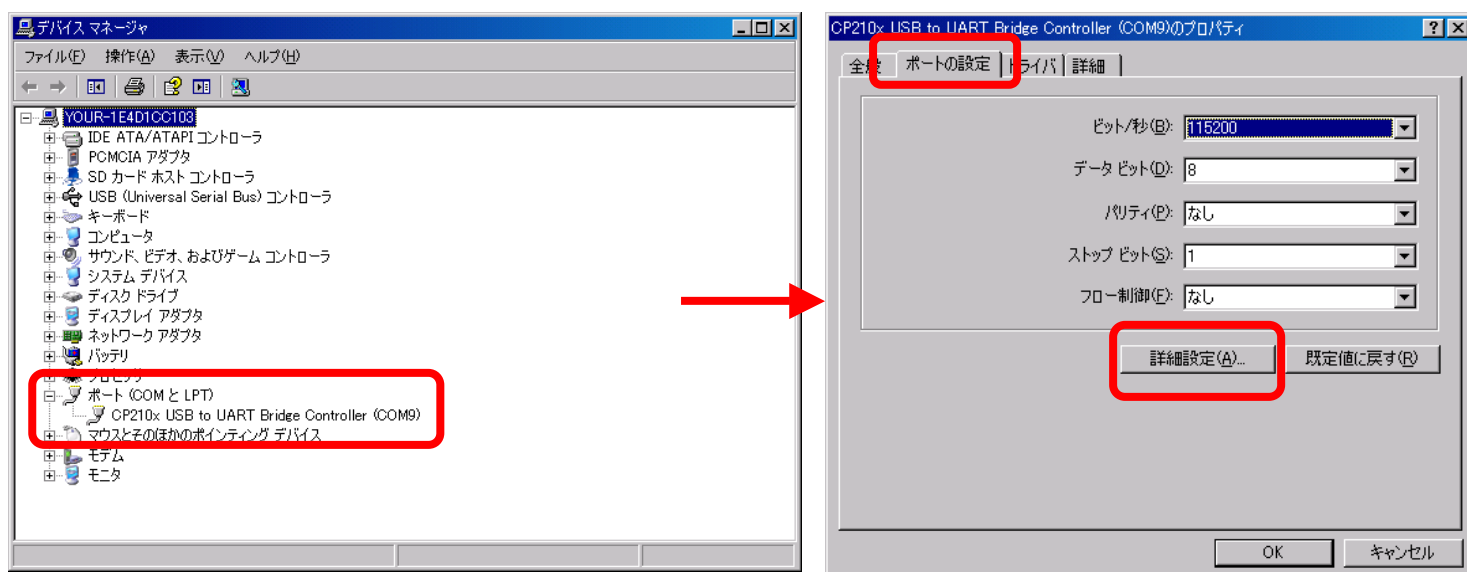
本説明書 8～15 ページの解説に従い、ロボットにプログラムの書き込みを行なった際、ロボットとの接続やボタンの操作などが正しく行われているにもかかわらず書き込みに失敗する場合は、お使いの PC のシリアルポート番号設定を変更することで改善できる場合があります。

シリアルポート番号の変更は、以下の手順で行ないます。

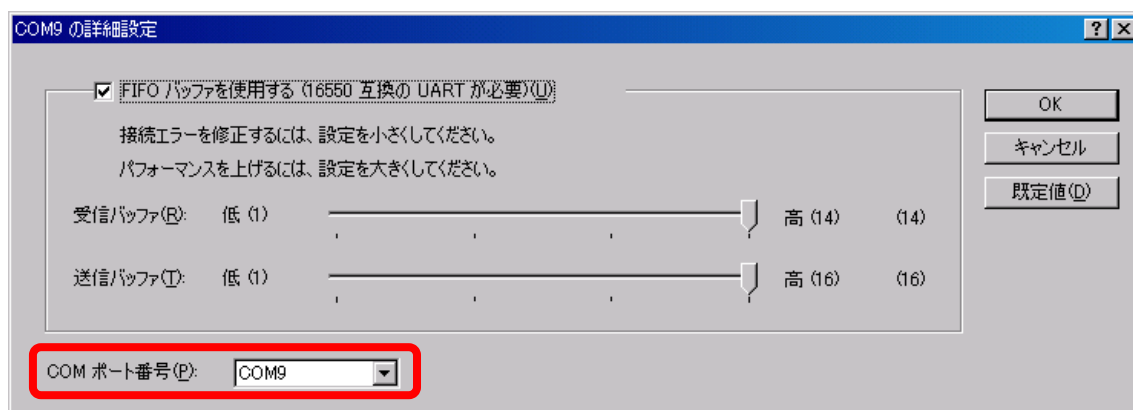
まず、OS のスタートメニューをクリックし、表示されたメニューより「マイコンピュータ」のアイコンを右クリックしてください。右クリックするとポップアップメニューを表示するので、メニュー中の「プロパティ」をクリックしてください（左下図）。クリックすると右下図のウィンドウを表示するので、「ハードウェア」のタブインデックスをクリックしてウィンドウの表示を切り替え、「デバイスマネージャー」のボタンをクリックしてください。



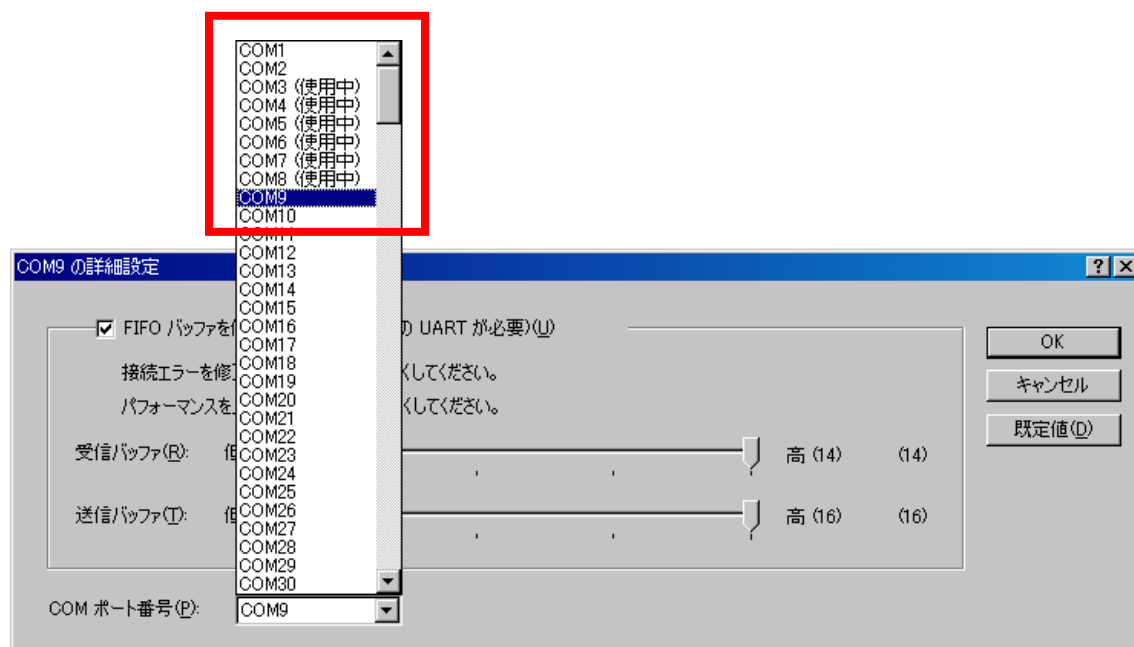
ボタンをクリックするとデバイスマネージャのウィンドウを画面に表示します（下図）。ここで、VS-WRC003 と PC を接続し、VS-WRC003 の電源を ON にしてください。電源を ON にすると、ウィンドウの「ポート (COM と LPT)」の項目に「CP210x USB to UART Bridge Controller (COM?)」という行が追加されます。この「CP210x USB to～」の行をダブルクリックしてください。ダブルクリックすると更に詳細を設定するダイアログを表示するので、「ポートの設定」のタブインデックスをクリックしてウィンドウの表示を切り替え、「詳細設定」のボタンをクリックしてください。



ボタンをクリックすると、以下のダイアログを表示します。このダイアログ下部にある「COM ポート番号(P)」より、シリアルポート番号を変更することができます。

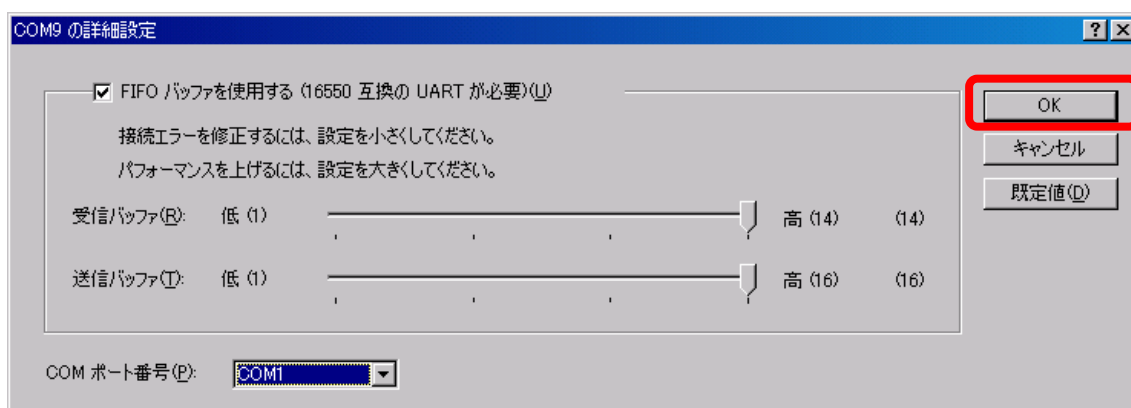


「COM ポート番号(P)」の項目をクリックするとシリアルポート番号の選択が表示されます。こちらより、現在の設定より小さいシリアルポート番号を選択します。

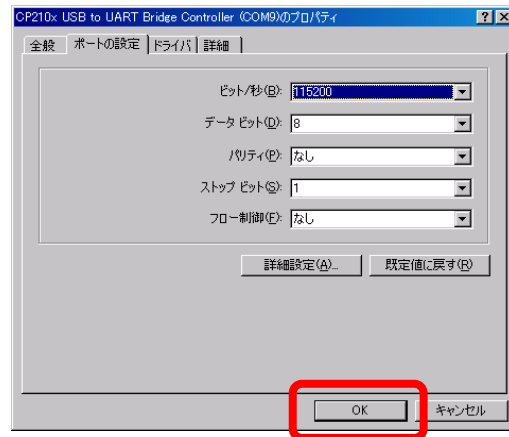


項目に「(使用中)」の表示が無い番号のシリアルポートを選択してください。また、現在の設定より小さいシリアルポート全てに「(使用中)」の表記がある場合、現在実際に使用していない番号を選択しても問題なく利用できる場合があります。

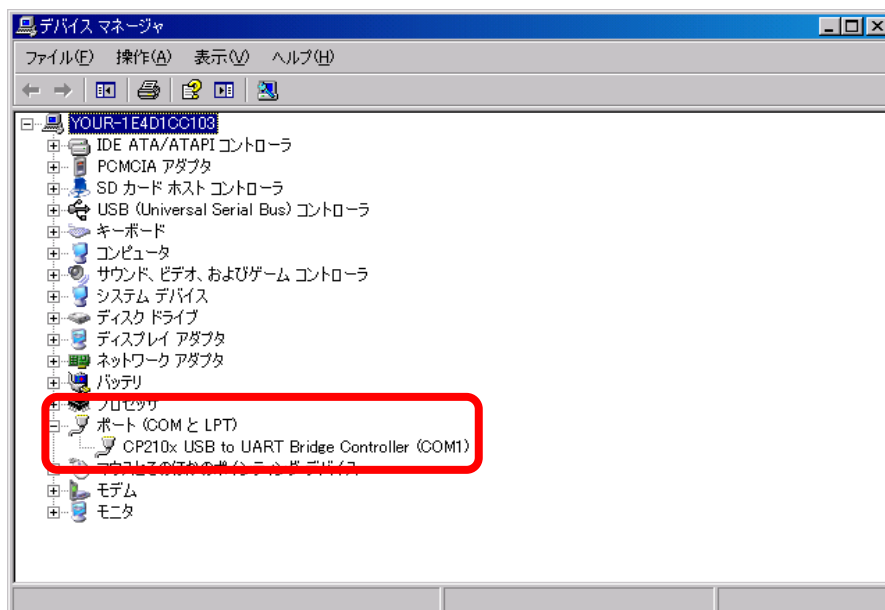
設定を行ったら、「OK」ボタンをクリックして設定を適用しダイアログを閉じてください。



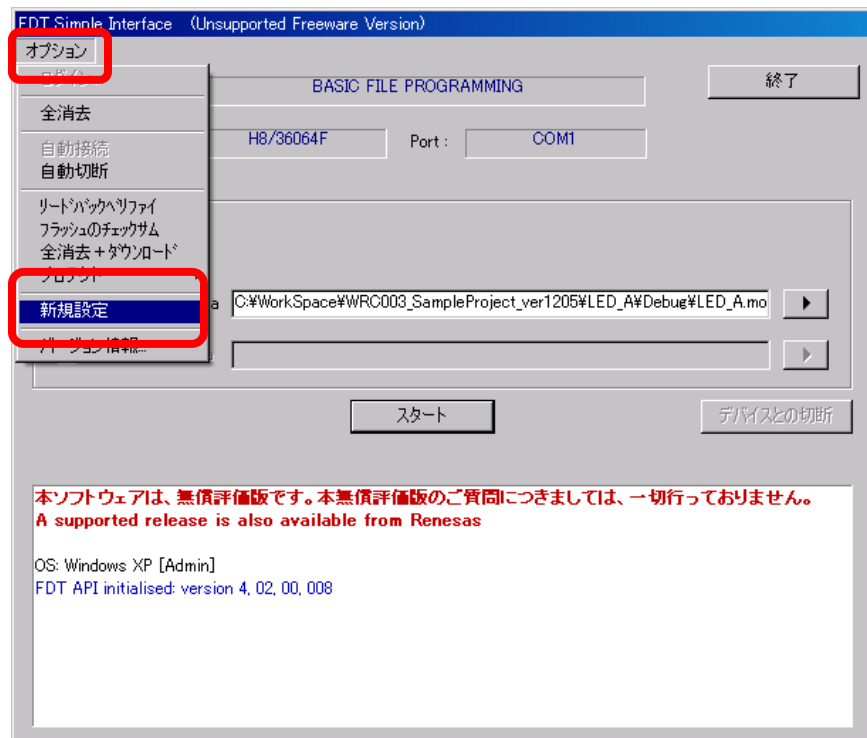
続いて以下のダイアログも「OK」ボタンをクリックし、設定を適用してダイアログを閉じてください。



最後に、一度 VS-WRC003 を PC から抜き差しして、PC に VS-WRC003 を再認識させてください。設定が正しく行なわれていたら、シリアルポート番号が変更した番号に切り替わります。



シリアルポート番号の変更は、FDT の設定にも反映させる必要があります。FDT の設定を変更する場合は、メニューより「オプション」→「新規設定」をクリックしてください。クリックすると、本説明書 9 ページからの記述と同じ画面が表示されるので、説明書の記述に従い、改めて設定を行なってください。



## ・ お問い合わせ

ヴイストーン株式会社

〒554-0024 大阪市此花区島屋 4-4-11

Tel:06-6467-6601 Fax:06-6467-6602

URL: <http://www.vstone.co.jp/>

e-mail: [infodesk@vstone.co.jp](mailto:infodesk@vstone.co.jp)

(2009.2.13)