ARM CortexM3 搭載学習用 CPU ボード

VS-WRC104

取扱説明書

ヴイストン株式会社
（ 2015.7.30 ）

目次

１　はじめに

本説明書は、ARM マイコンを搭載した、ロボット制御学習用の CPU ボード「VS-WRC104」の使用方法について解説したマニュアルです。ご使用になる前に、かならず本説明書をよくお読みいただき、安全にお使いください。

２　仕様

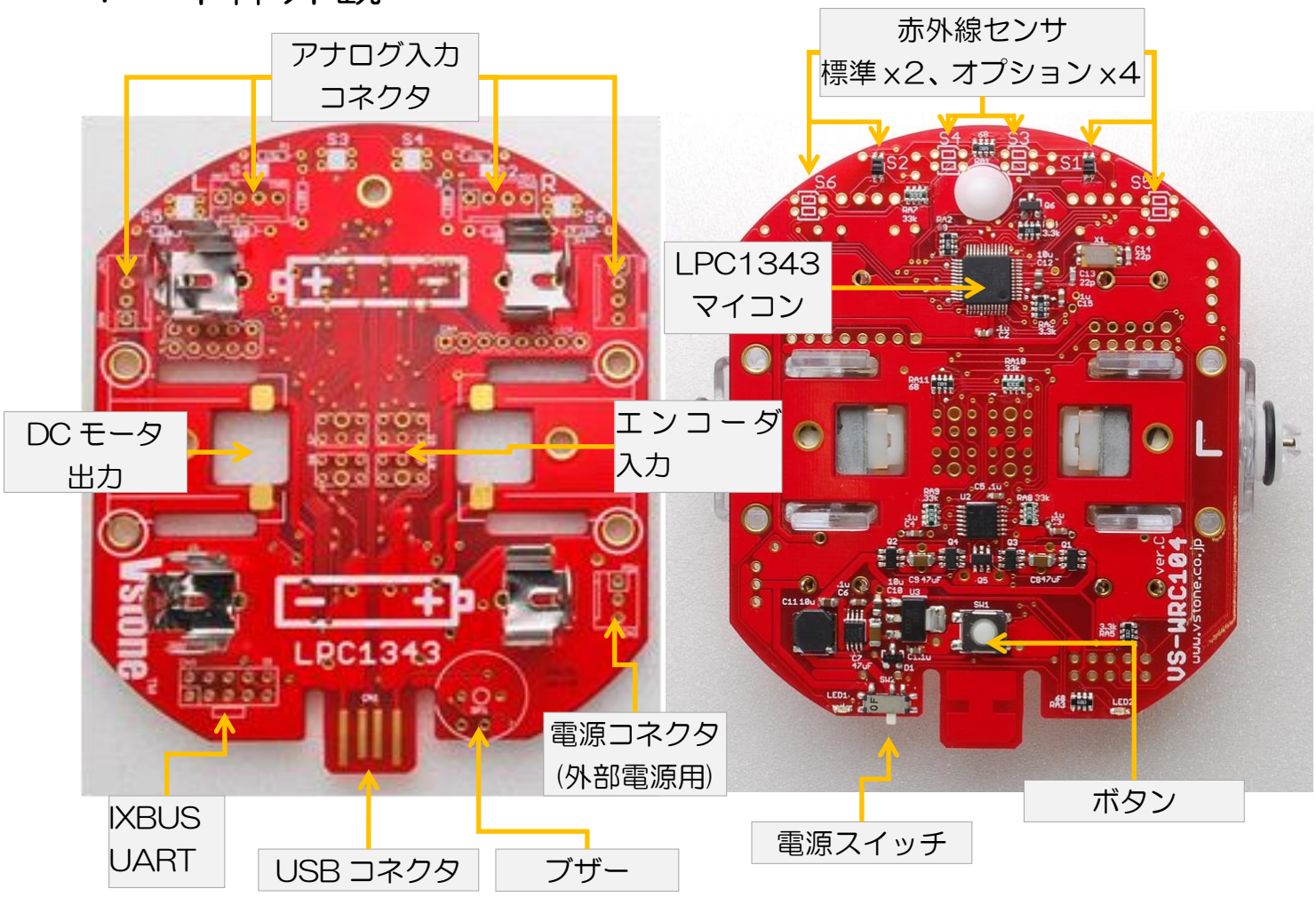
大きさ	長さ 90×幅 92×高さ 23 (mm)
重量	約 115g（電池搭載時）
CPU	ARM CortexM3　LPC1343 （NXP セミコンダクターズ製）
電源	DC　2～4V アルカリ乾電池 2 本　または　ニッケル水素充電池 2～3 本
出力	DC モータ×2　(連続電流 2A まで) LED×2　（オレンジ・緑） 圧電ブザー×1（オプション）
入力	赤外線センサ×2 赤外線センサ x4またはアナログセンサ入力×４ ボタン
インタフェース	USB（HID 準拠）×１、シリアルポート（3.3v レベル）×１ IXBUS（I2C）×１、拡張 IO ポート
オプション	ロータリーエンコーダ拡張セット 赤外線センサ拡張セット IXBUS 拡張ボード（VS-IX001、VS-IX008 など) ユニバーサルボード

3　ご注意

本製品を取り扱う際には、注意事項に従い正しくお使いください。

- VS-WRC104LV（以降　本ボード）に強い衝撃を与えないでください。
- 本ボードを水に濡らしたり、湿気やほこりの多い場所で使用したりしないでください。ショートなどによる故障する恐れがあります。
- 本ボードから煙が発生した場合、すぐに電源をお切りください。
- 本ボードを幼児の近くで使用したり、幼児の手の届くところに保管したりしないでください。
- 動作中、基板上の素子が高温になることがありますので、絶対に触れないでください。
- 基板上の端子（金属部分）に触れると静電気により故障する恐れがあります。かならず基板の縁を触るようにしてください。
- 基板上の端子同士が金属などでショートすると、過電流により故障する可能性があります。
- CN8、CN9 のコネクタは付属しておりません。（拡張オプションとして販売しております。）
- DC モータ出力は FA-130RA タイプモータ（ビュートミニ用）を使用するためのポートです。それ以外のモータも使用可能ですが、モータの種類、使用方法によっては基板の破損につながる可能性があります。

4　本体外観



5　各部詳細

ここでは、基板上の各機能について解説します。

5-1　この章の見方

○ ピン配置

各入出力端子のピン配置を示します。ARM マイコンとつながっている場合、ポート番号を記載します。ポートと ARM マイコンとの間は直接接続されているだけではなく、なにかしらの回路がある場合があります。

○ ブロック

ビュートビルダー2 を使用する際に、その機能を使用するためのブロックです。そのブロックをシーケンスエリアに置くことで、簡単に機能を使用することができます。

ビュートビルダー2 の使用方法の詳細は、ビュートミニ ARM 付属の CD-ROM 内、またはサポートページにあります「ビュートビルダー2 取扱説明書」をご覧ください。

○ C 言語関数

弊社 Web サイトで公開している C 言語サンプルソースの中で定義されている関数になります。その関数を使用することで、数行のプログラムで動作させることが可能になります。
無償の C 言語開発環境 LPCXpresso、C 言語のサンプルソースの使用方法については本マニュアルの 6 章をご覧ください。

○ ポート

各機能を構成する部品が ARM マイコンのどの IO ポートに接続されているかを示します。C 言語で 1 からプログラムを作成する場合に使用します。

5-2　通信コネクタ　（CN1）

通信コネクタは、A タイプの USB 端子となっており、市販の USB 延長ケーブルなどでパーソナルコンピュータ（以降 PC）と接続できます。

使用している ARM マイコンには USB 機能が搭載されているため、USB で PC と接続すると自動的にドライバがインストールされ、使用できる状態となります。

○ ポート

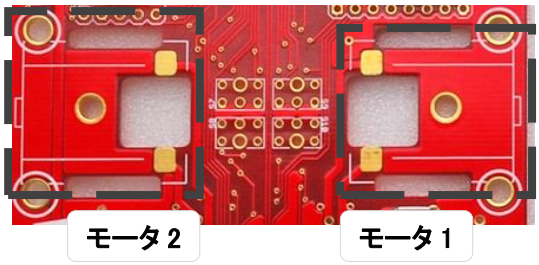
ARM マイコンの USB_DP、USB_DM が接続されています。

本ボードと PC 間の通信をするサンプルプログラム、シリアル通信をおこなう方法などは、サポートページをご覧ください。

5-3　DC モータ出力　（CN3，4）

本ボードには 2ch の DC モータ出力を搭載しています。

このモータ出力はビュート ミニ用モータ（FA-130RA タイプモータ）を使用するためのポートになります。それ以外のモータでも、連続電流 2A までのモータであれば接続可能です。（この場合、モータの種類、使用方法によっては基板が破損する可能性がありますので、十分ご注意ください。）



○ 命令ブロック

移動アクションブロック

モータ制御ブロック

○ C 言語関数

void Mtr_Run_Iv (short mt1 , short mt2 , short mt3 , short mt4 , short mt5 , short mt6);

設定した速度でモータを駆動。呼び出した後はその状態を保持し、制御値に 0 を与えるまで停止しない。

引数：　モータの制御値

0　：フリー（ブレーキ）

0x8000　：フリー（ブレーキ）

時計回り最大値　：0x7FFF　(32767)

反時計回り最大値：0x8001　(-32767)

戻り値：　無し

○ ポート

- CN3（M1）　PIO2_0 、 PIO2_1 （方向出力）　PIO0_8（PWM、16 bit タイマーB0MR0）
- CN4（M2）　PIO2_2 、 PIO2_3 （方向出力）　PIO1_9（PWM、16 bit タイマーB1MR0）

5-4　LED

本ボードには、オレンジ、緑の LED が１個ずつ搭載されています。
それぞれの LED は接続されているポートを LOW レベルにすることで点灯します。

LED1　　**LED2**

○　命令ブロック

LED ブロック(オレンジ)

LED ブロック（緑）

○　C 言語関数

void LED(BYTE LedOn);

CPU ボード上の２つの LED を制御する関数

引数：　　0:　消灯
　　1:　緑　点灯
　　2:　オレンジ　点灯
　　3:　両方　点灯

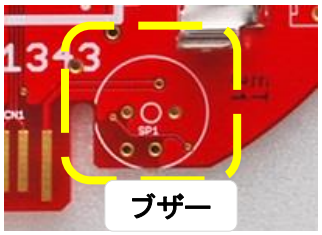
戻り値：　　無し

○　ポート

- LED1（オレンジ）：PIO0_3
- LED2（緑）　　：PIO0_7

5-5　ブザー出力

本ボードには、圧電ブザーがオプションで搭載できます。（エンコーダ・ブザー搭載版は標準装備）単音を出力することが可能です。ブザーは 32bit タイマー1 の割り込みにより PIO ポートをスイッチングして駆動しています。



○ 命令ブロック

ブザーブロック

○ ポート

PIO1_8

○ C 言語関数

void BuzzerSet(BYTE pitch , BYTE vol);

音程、ボリュームの設定。ブザーを鳴らす際の音程とボリュームを設定

引数：　　pitch：音程の設定（0～255、値が大きいほど低い音）

　　vol：ボリュームの設定（0～128）

戻り値：　　無し

void BuzzerStart();

開始。ブザーを鳴らし始める。BuzzerStop()関数を呼ぶまでなり続ける。

引数：　　無し　　戻り値：　　無し

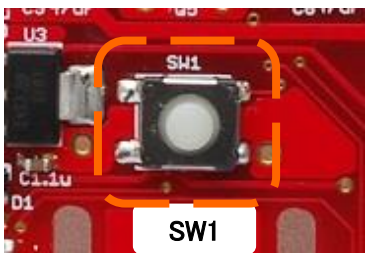
void BuzzerStop();

停止。ブザーを止める。

引数：　　無し　　戻り値：　　無し

5-6　スイッチ入力　（SW1）

本ボードには、1 つの押しボタンが搭載されています。本体後部搭載の電池付近を押しこむことで操作することが可能です。ビュートビルダー2 を使用する場合、このボタンを押すことで、書き込んだシーケンスをスタートさせます。シーケンスを再生中は、スイッチ入力として使用可能です。



○　C 言語関数

BYTE getSW();

ボタン状態取得。CPU ボード上の押しボタンの状態を取得する

戻り値：　　0　：　off

　　1　：　on

○ポート　　PIO0_1

5-7　赤外線センサ入力（S1、2，3，4，5，6）

本ボードには、2ch の赤外線センサが標準搭載されており、オプションの赤外線センサを実装することで、最大で 6ch までの赤外線センサを搭載することが可能です。S3～S6 は次項のアナログセンサ入力と共用ですので、赤外線センサ拡張を行った場合、アナログセンサ入力が使用できなくなります。

○　C 言語関数

UINT AdRead(BYTE ch);

AD 入力値取得。A/D 変換の入力値を取得。

引数：　　チャンネル（0,1,2,3,5,6　=　S1,2,3,45,6）

戻り値：　　A/D 変換の値（0～1023）

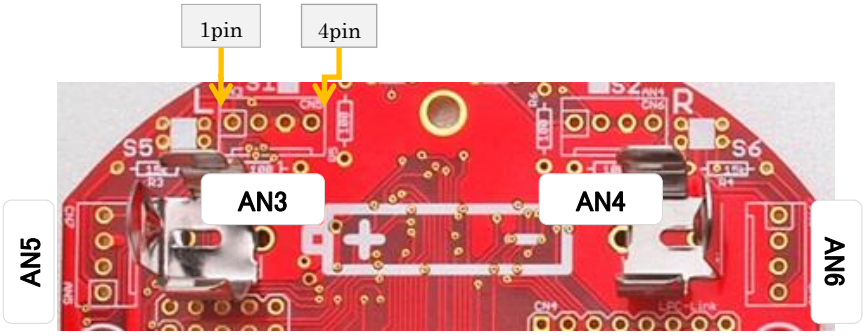
○ ポート

- S1（ch0）　　PIO0_11
- S2（ch1）　　PIO1_0
- S3（ch2）　　PIO1_1
- S4（ch3）　　PIO1_2
- S5（ch5）　　PIO1_4
- S6（ch6）　　PIO1_10

5-8　アナログセンサ入力　（CN5、6、7、8）

本ボードには、4ch のアナログセンサ入力を搭載しています。「ビュート」、「ビュート チェイサー」「ビュートローバー」や VS-WRC003 、VS-WRC003LV に対応したセンサデバイスもそのまま接続できます（ビュートシリーズ用センサ・ケーブルの接続には、別途コネクタをはんだ付けする必要があります）。

自作のアナログ入力デバイスを接続する場合は、各ピンの仕様に従って作成してください。



○ ピン配置

1pin	：	+3.3V	電源、赤外エミッタカソード
2 pin	：	150Ω-GND	1/6W 抵抗（赤外エミッタアノード）を実装可能
3 pin	：	GND	
4 pin	：	信号入力	1/6W 抵抗（+3.3V プルアップ）を実装可能

○ C 言語関数

UINT AdRead(BYTE ch);

AD 入力値取得。A/D 変換の入力値を取得。

引数：　　チャンネル（2,3,4,6　=　CN5,6,7,8）

戻り値：　　A/D 変換の値(0～1023)

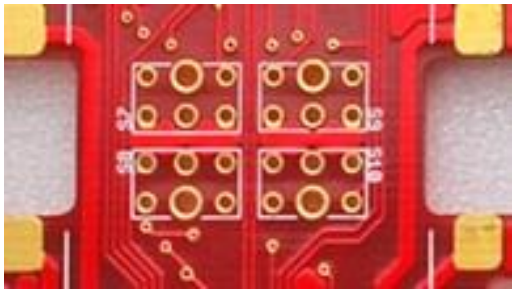
○ ポート

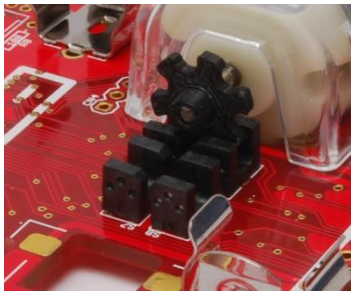
- CN5（AN3、ch2）　　PIO1_1
- CN6（AN4、ch3）　　PIO1_2
- CN7（AN5、ch4）　　PIO1_3
- CN8（AN6、ch6）　　PIO1_10

5-9　ロータリーエンコーダ入力ポート

このポートはモータの回転数を測定する事が可能なロータリーエンコーダ用の入力ポートです。透過型フォトインタラプタを 4 個実装することで正転・逆転の方向と回転数を測定することができます。

分解能はモータ 1 回転辺り 24 パルス、減速比が 6:1、タイヤ径が 12mm ですので、1 パルスあたりπ/12[mm]になります。ただし、モータ軸とタイヤ間、タイヤと地面間に滑りがありますので、使用状況によって正しく測定できない場合があります。





○ 命令ブロック

エンコーダブロック

○ C 言語関数

void InitEncoder();

エンコーダ初期化

エンコーダ拡張を使用する際に、Init()後に 1 度だけ実行する必要がある

引数：無し、戻り値：無し

void ClearEncoder();

エンコーダクリア

エンコーダでカウントしている値を 0 にクリアする

引数：無し、戻り値：無し

void GetEncoder(long *A,long *B);
エンコーダ値取得。エンコーダでカウントしている値取得する
引数：　*A：S1,2 側のカウントを取得する変数のポインタ
　　*B：S3,4 側のカウントを取得する変数のポインタ
戻り値：無し

○ ポート
・S7 （左モータＡ相）　PIO0_2
・S8 （左モータＢ相）　PIO1_5
・S9 （右モータＡ相）　PIO2_11
・S10（右モータＢ相）　PIO2_10

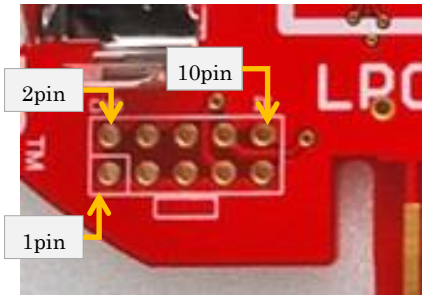
5-10　IXBUS・UART（CN9）

このポートは、IXBUS 拡張基板や Bluetooth モジュール VS-BT001 を接続するためのコネクタです。ビュートビルダー2 で使用する場合は、センサとしてジャイロ/加速度センサ拡張ボード VS-IX001、アナログ入力拡張ボード VS-IX008、8 連赤外線センサボード VS-IX010 を使用することができ、VS-BT001 を接続すると無線でプログラミングを行うことができます。IXBUS 機器をビュートビルダー２上で使用する場合必ず基板上の DIP スイッチの 3，４を OFF にしてください。

VS-IX001、VS-IX008、VS-IX010 の出力はすべて 0～4095 の間で出力されます。

IXBUS 拡張ボードは DIP スイッチの設定により４個まで同時に接続できます。（UART は１機器まで）

C 言語でプログラミングする場合、すべてのボードが使用可能です。（サンプルでは上記記載ボードのみサポート）



○ピン配置
1 pin　：　PIO1_7（TXD）
2 pin　：　PIO1_6（RXD）
3 pin　：　NC
4 pin　：　NC
5 pin　：　PIO0_4（SCL）
6 pin　：　PIO0_5（SDA）
7 pin　：　+5V
8 pin　：　+Vbat
9 pin　：　+3.3V
10 pin　：　GND

○C言語関数（IXBUS）
void i2C_init();
I2C 初期化関数。IXBUS 機器を使用できる状態にする。
引数：無し、戻り値：無し

unsigned char Get_IX008(unsigned char Addr,unsigned int *retdata)

IXBUS 拡張ボード（IX008、IX001、IX010）から８つのセンサ入力を取得する関数。

引数： **Addr**： IXBUS 機器のアドレスを指定。0x90,0x92,0x94,0x96 のいずれかを指定

***retdata**： ８つ分のセンサ入力値を格納する配列のポインタ。0～4095

戻り値：正常に受診した場合１，失敗した場合０

OC 言語関数（UART）

void InitSci3(uint32_t baudrate,BYTE parity ,BYTE stop);

UART 初期化関数。UART 機能を有効にし、ボーレート、パリティビット、ストップビットなどを設定します。

引数： **baudrate**： 通信時のボーレートを指定。VS-BT001 を使用する場合 115200 を指定

parity： パリティのタイプを指定。奇数（odd）偶数（even）またはなし（non）を指定

stop： ストップビットのビット数を指定。１または２。

戻り値：無し

void SciByteTx(unsigned char data);

１バイト送信関数。Tx ポートを通じて通信相手に１バイトのデータを送信する。

引数： **data**： 送信データ

戻り値：無し

unsigned char SciByteRx(unsigned char *data);

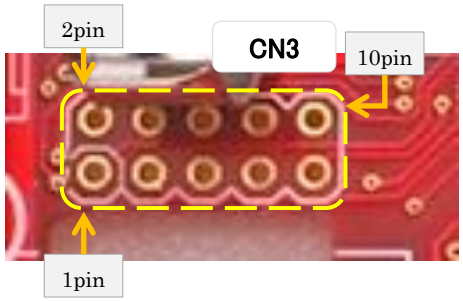
１バイト受信関数。Rx ポートから受信されバッファリングされたデータを１バイト取得する。

引数： ***data**： 受信データを受け取るバイト変数のポインタ

戻り値： 受信バッファが空の場合 0、まだ受信データが有る場合１。戻り値が１の間受信し続けることで、バッファ内すべてのデータを連続して受信できる。

5-11　拡張 IO（CN3）

このポートは、本ボード内では使用していない IO ポートをまとめたコネクタとなります。

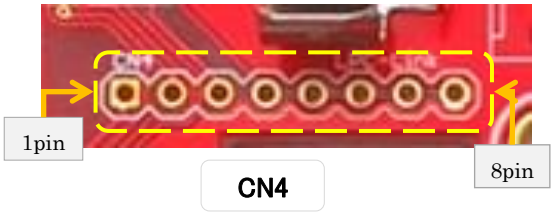


○CN3　ピン配置

1 pin　：　PIO2_4
2 pin　：　PIO2_5
3 pin　：　PIO2_6
4 pin　：　PIO2_7
5 pin　：　PIO2_8
6 pin　：　PIO2_9
7pin　：　PIO3_0
8 pin　：　PIO3_1
9 pin　：　PIO3_2
10 pin　：　PIO3_3

5-12　LPC-Link 接続ポート（CN4）

このポートは、NXP セミコンダクターズ製の評価ボード LPCXpresso などに付属しているデバッガ「LPC-Link」を接続できるポートです。このポートに「LPC-Link」を接続することで、開発環境 LPCXpressoIDE で、デバッグ（ブレークポイントでのプログラム一時停止、マイコン内部のメモリ、変数の確認など）をおこなうことができるようになります。



○ポート

1pin　：　+3.3v
2pin　：　PIO1_3/SWDIO
3pin　：　PIO0_10/SWCLK
4pin　：　PIO0_9/SWO
5pin　：　NC
6pin　：　RES
7pin　：　+5v
8pin　：　GND

６　開発環境の導入とサンプルプログラムの実行

本項では、「VS-WRC104」を C 言語で開発する際に使用可能な、無償配布されている“LPCXpresso”を用いたサンプルプログラムの実行に関して解説をします。LPCXpresso 評価版のご利用は無料ですが、ダウンロードの際にユーザーカウントを作成、またビルドサイズの制限がかかっていますので、インストール後にネットワークを利用した認証作業をする必要があります。

本ボードは「VS-WRC103LV」と互換性（一部ピンの仕様が違います）がありますので、サンプルプログラムは「VS-WRC103LV」のものを使用します。そのため、本文中においても「VS-WRC103LV」のサンプルを使用して解説します。

おおまかな手順は以下の通りです。

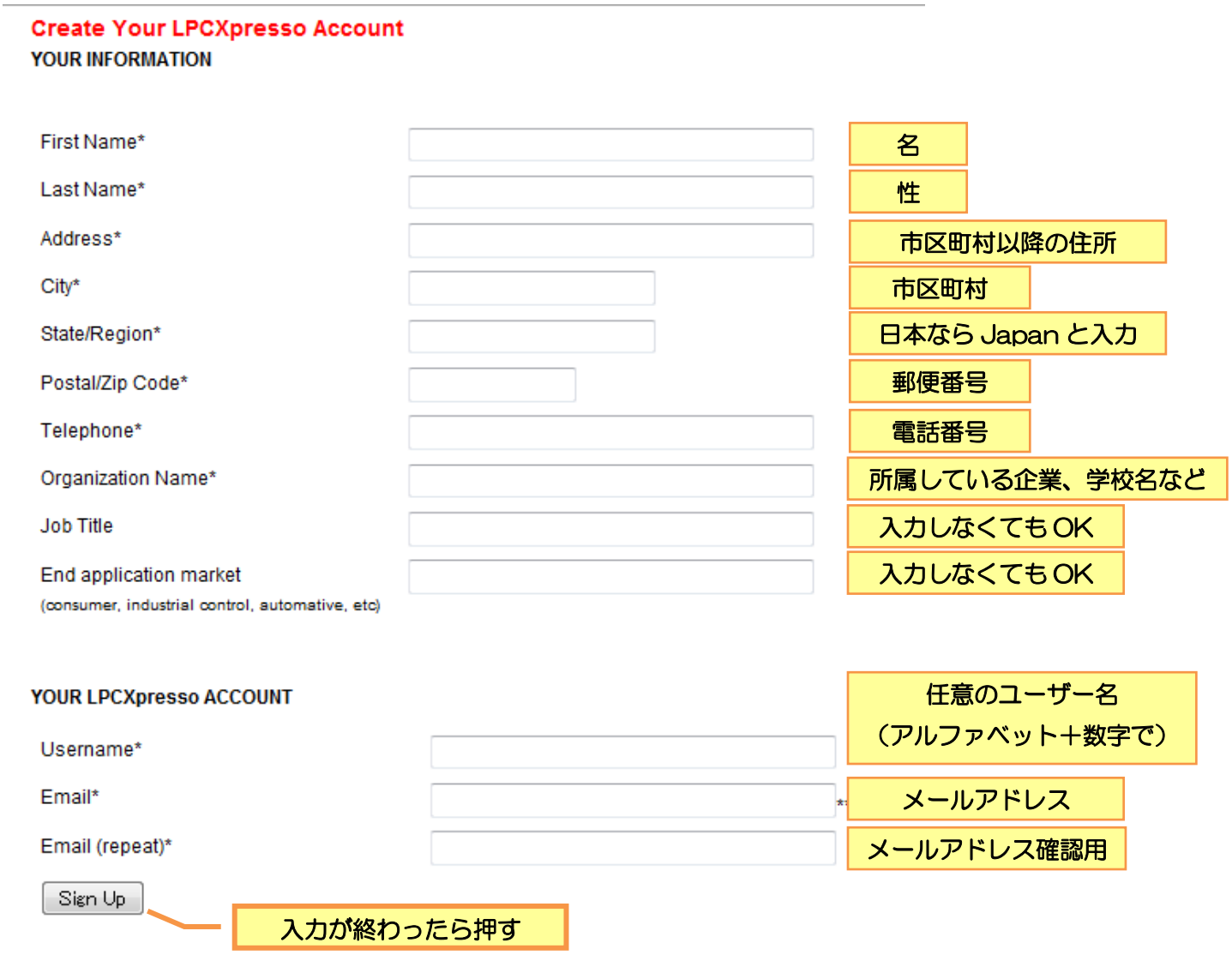
① 開発環境の入手、インストール
② サンプルプロジェクトのダウンロード、ビルド
③ プログラムの書き込み

6-1　開発環境 LPCXpresso の入手

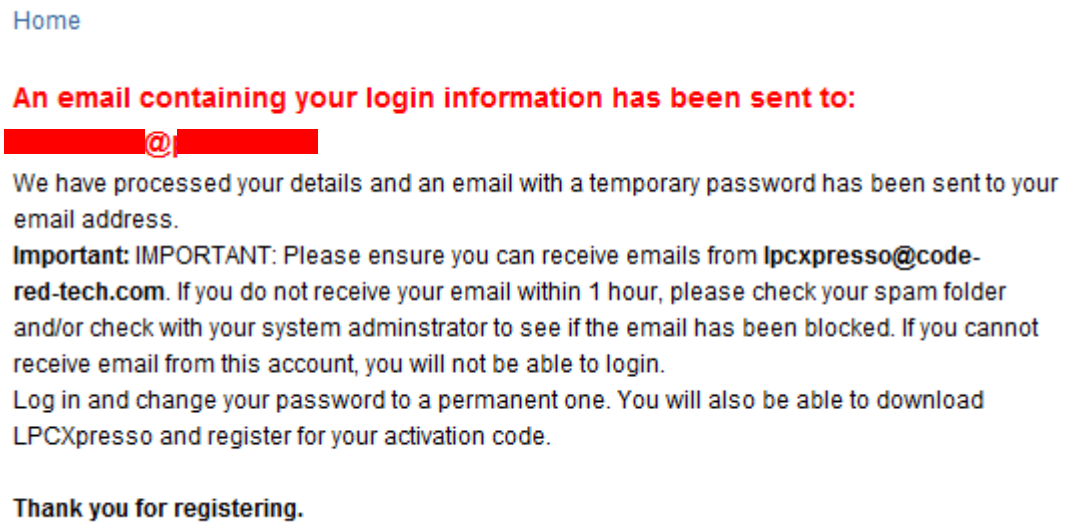
① LPCXpresso は、CodeRed（下記の URL）からダウンロードします。（ダウンロードにはユーザー登録が必要になります）
http://lpcxpresso.code-red-tech.com/LPCXpresso/

② WEB ページを開くと、ログイン画面が現れます。LPCXpresso のユーザーカウントをお持ちで無い場合は**「Create Acount」ボタンをクリック**してアカウントを作成してください。
既にアカウントをお持ちの方は、**ページ左上よりユーザー名とパスワードを入力して「LOG IN」をクリック**して、⑥へお進みください。

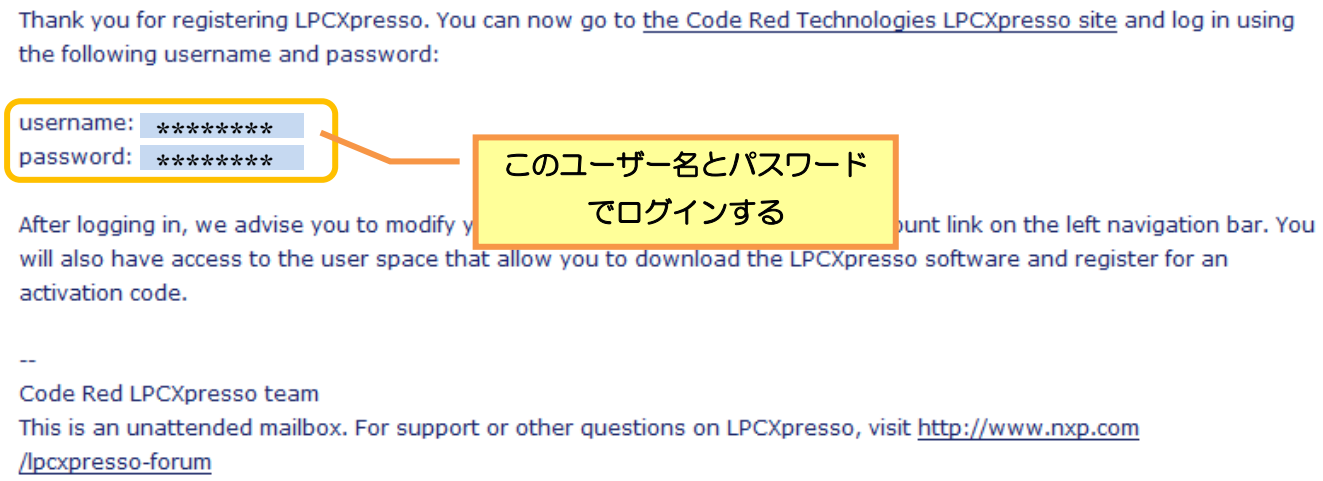
③ アカウントの作成では、以下のような項目を入力してください。



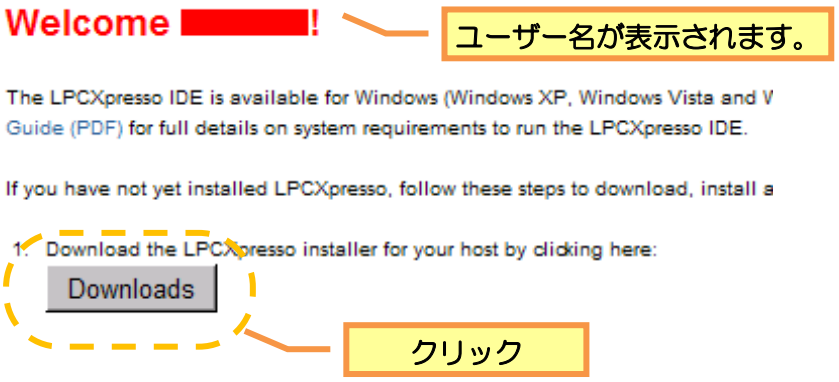
④ 「Sign Up」を押すと、以下のように表示され、入力したメールアドレス宛にメールが届きます。



⑤ 以下のようなメールが届き、登録したユーザー名の下にパスワードが記載されていますので、それを利用して②の画面からログインしてください。

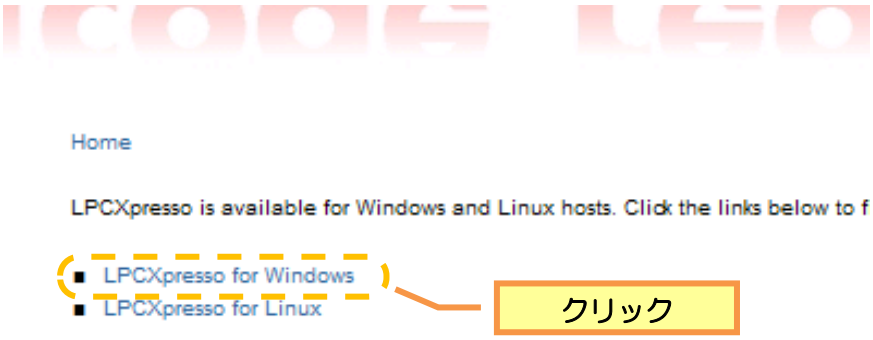


⑥ ログインしたら、ページ中央の「1.Donwload the LPCXpresso installer for your host by clicking here」の下にある「**Download」ボタンをクリック**してください。



⑦ クリックするとページが切り替わるので、そのページから**「LPCXpresso for Windows」をクリック**してください。

※本 CPU ボードは Windwos 環境のみの対応とさせていただいております。



⑧ クリックすると下記のページに切り替わります。このページの最下段より**「LPCXpresso v?.?.? [Current stable release]」というリンクをクリック**してください。（リンクの文字列はソフトウェアのバージョンを表します）

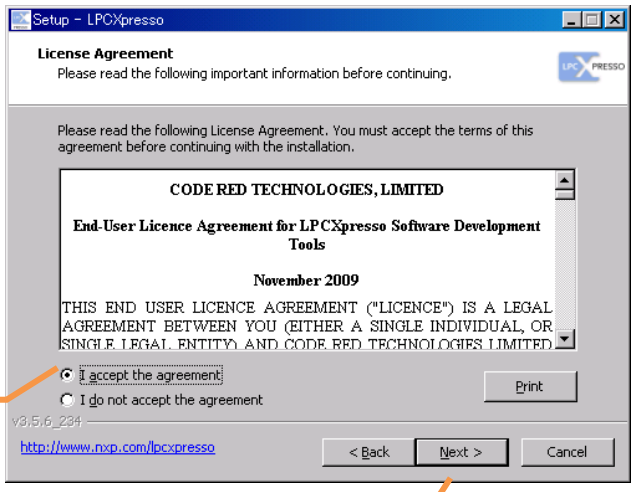
クリックするとダウンロードが始まります。ファイルのサイズが非常に大きいため、ダウンロードに 1 時間程度かかる場合がありますので、ご注意ください。

6-2　LPCXpresso のインストール

　ダウンロードした LPCXpresso を PC にインストールする方法について説明します。
ダウンロードしたインストーラ（exe ファイル）を実行し、以下の手順に沿ってインストールを進めます。
また、インストール中にネットワークに接続することがあります。その際は、ネットワーク接続を許可して作業を進めてください。

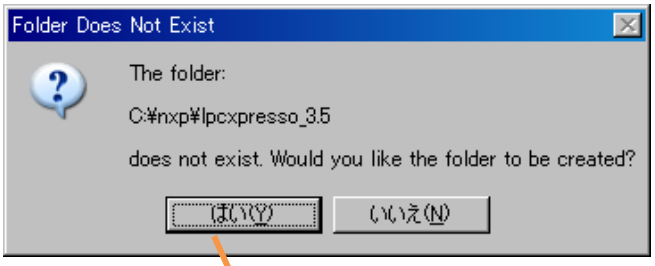
　※Windows のバージョンによって、仕様が若干異なる場合があります。

NEXT を押す

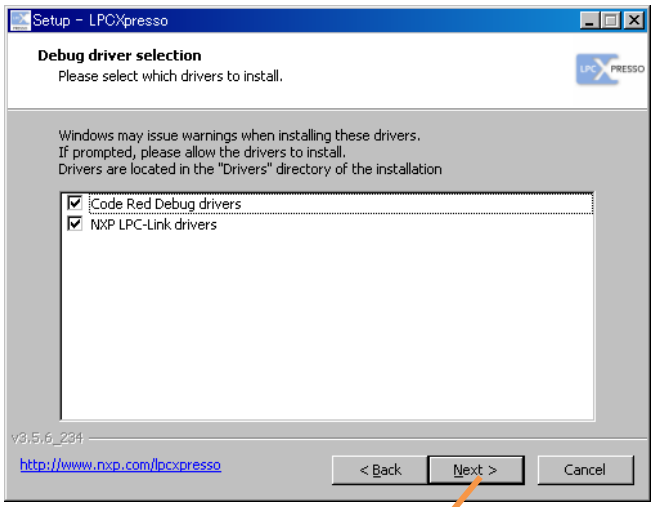


チェックする

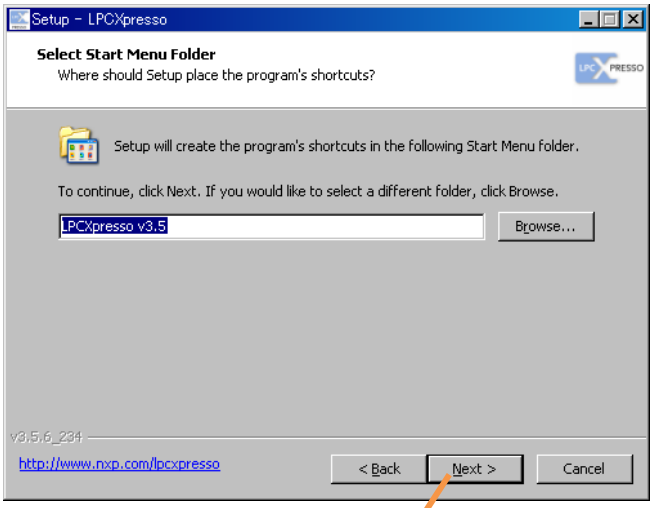
NEXT を押す



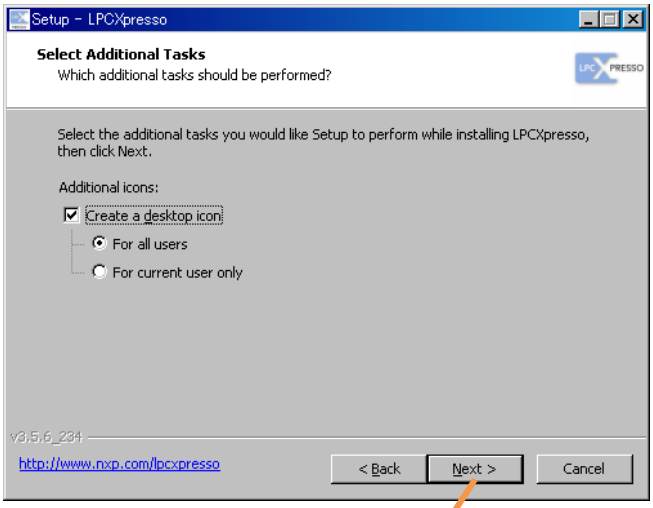
はいを押す



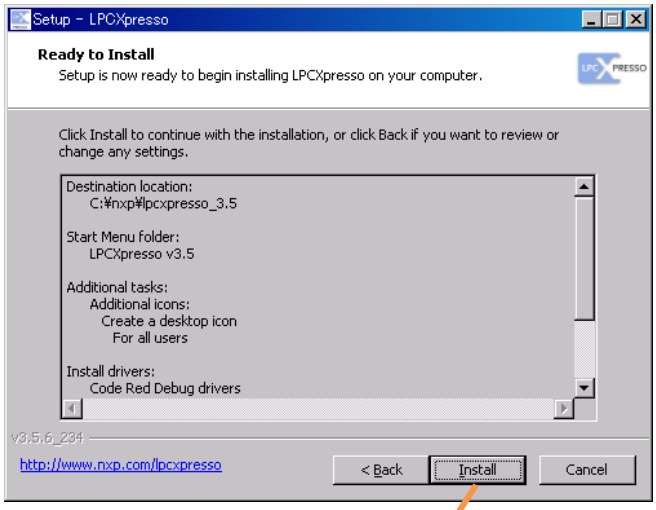
NEXT を押す



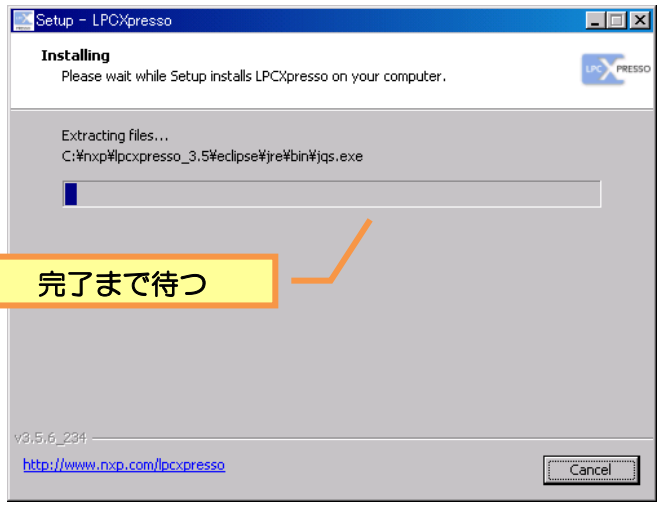
NEXT を押す



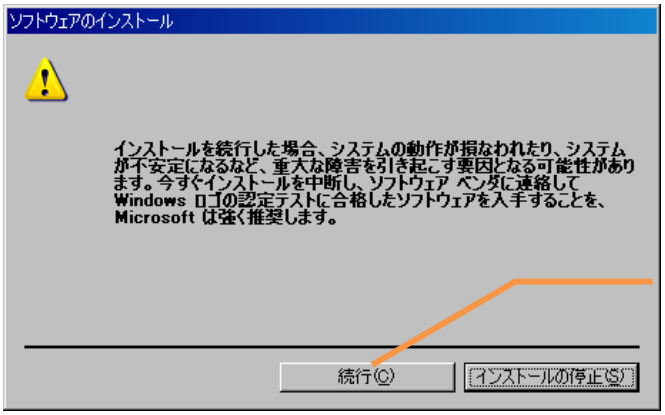
NEXT を押す



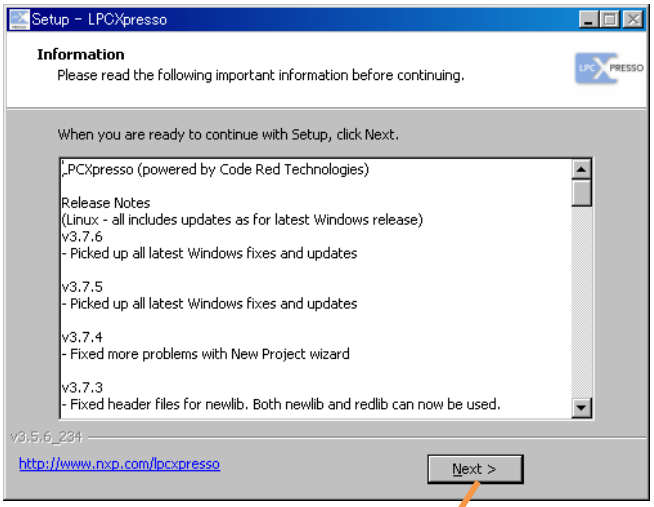
Install を押す

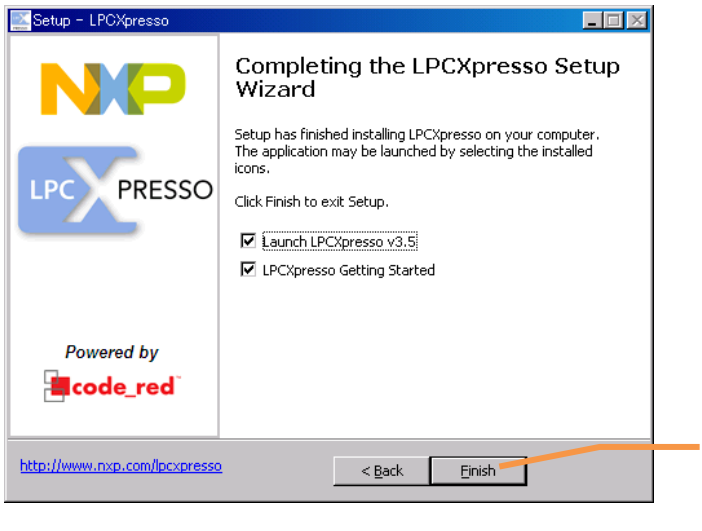


完了まで待つ



警告が数回表示されることがありますが、問題ありませんので、すべて「続行」を押します。





以上で、LPCXpresso のインストールは完了です。

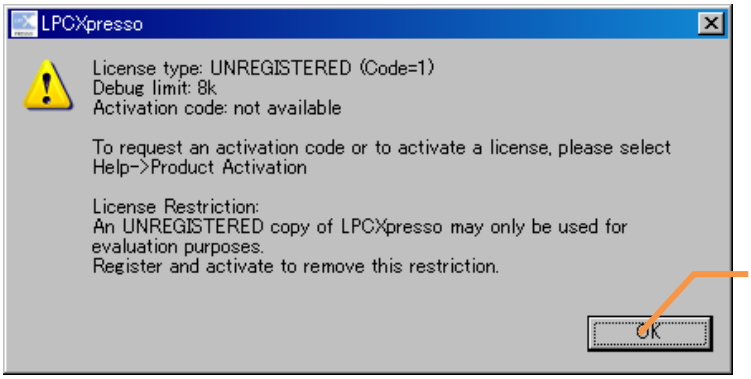
6-3　LPCXpresso の起動と認証

次に、以下の手順に沿って、認証作業を行います。認証作業ではインストールした PC をインターネットに接続している、または、インターネットに接続している別の PC が必要になりあります。

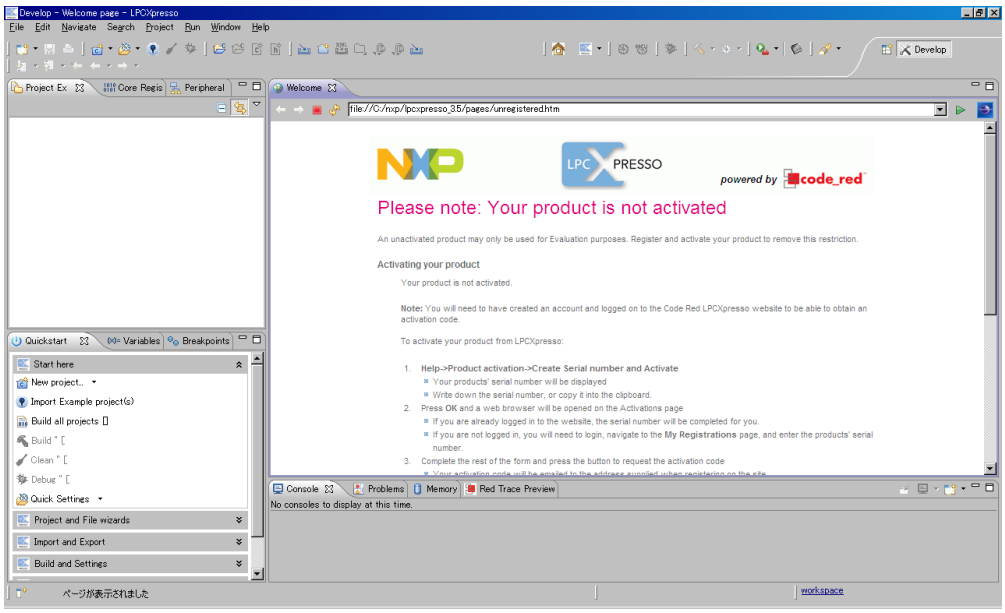
① LPCXpresso を起動します。起動するには、デスクトップ上のアイコンをダブルクリックするか、スタートメニューのすべてのプログラムより「LPCXpresso v*.*」＞「LPCXpresso v*.*」をクリックしてください。（**にはインストールした LPCXpresso のバージョンが数字で入ります）

② 起動すると以下の画面が表示されます。

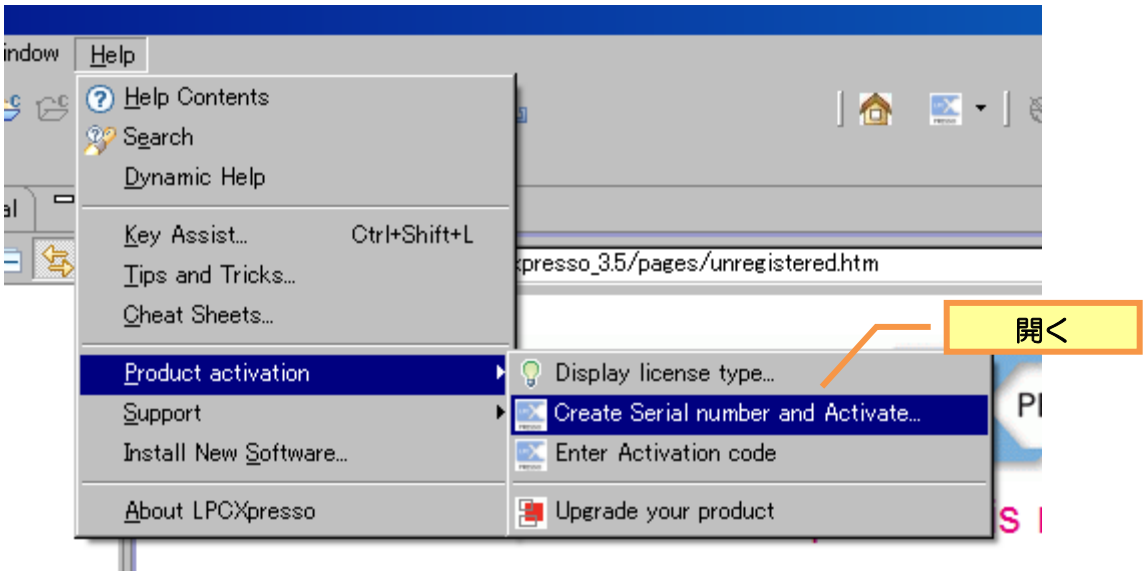
③ 認証を行っていないと以下のような表示がされますので、OK を押します。



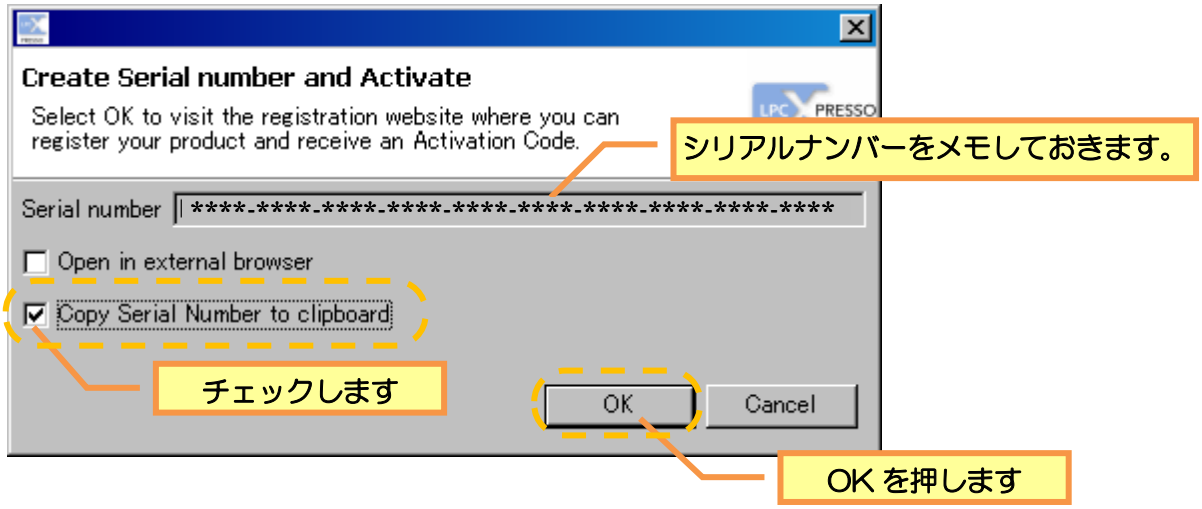
④ 起動すると、以下の編集画面が表示されます。



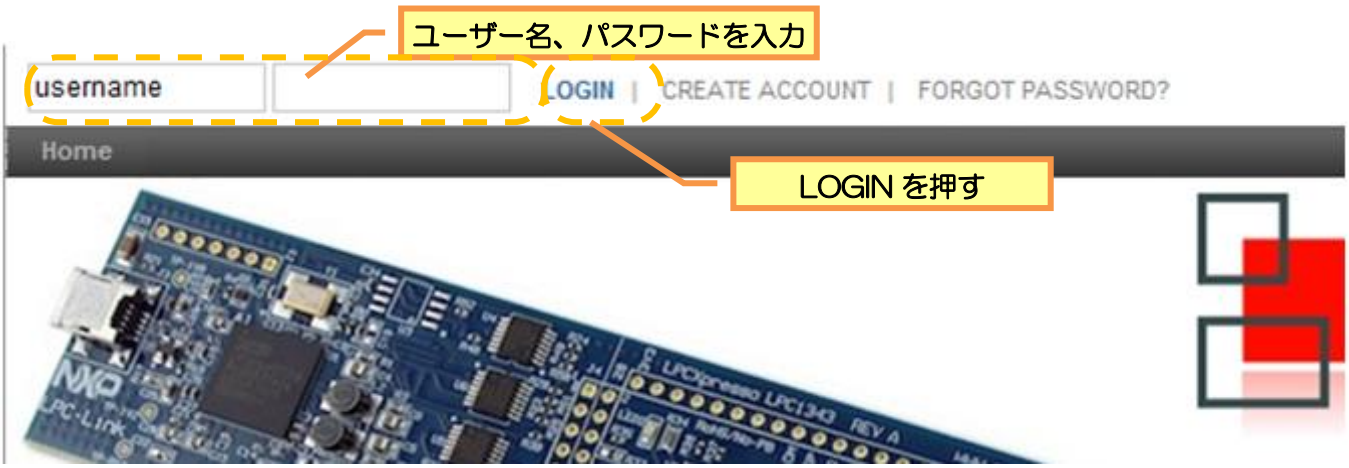
⑤ ツールバーのヘルプから、Help＞Product activation＞Create Serial number and Activate...を開きます。



⑥ 表示されたシリアルナンバーをメモ、コピーするなどして、一時的に保存しておき、OK を押します。



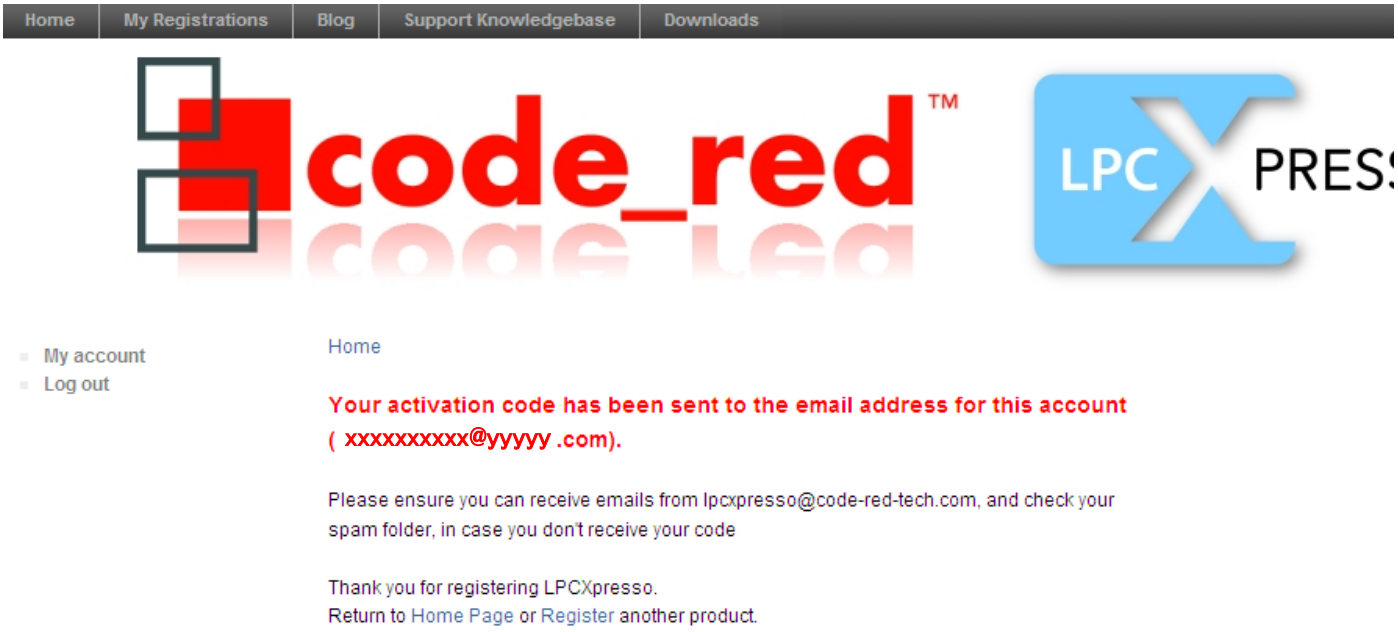
⑦ OK を押すと、以下のログイン画面が開きます。インストーラをダウンロードした際に登録したアカウントのユーザー名、パスワードを入力し、ログインします。
（ログイン済みの場合、⑨の画面が開きますので、⑦、⑧は飛ばします。）



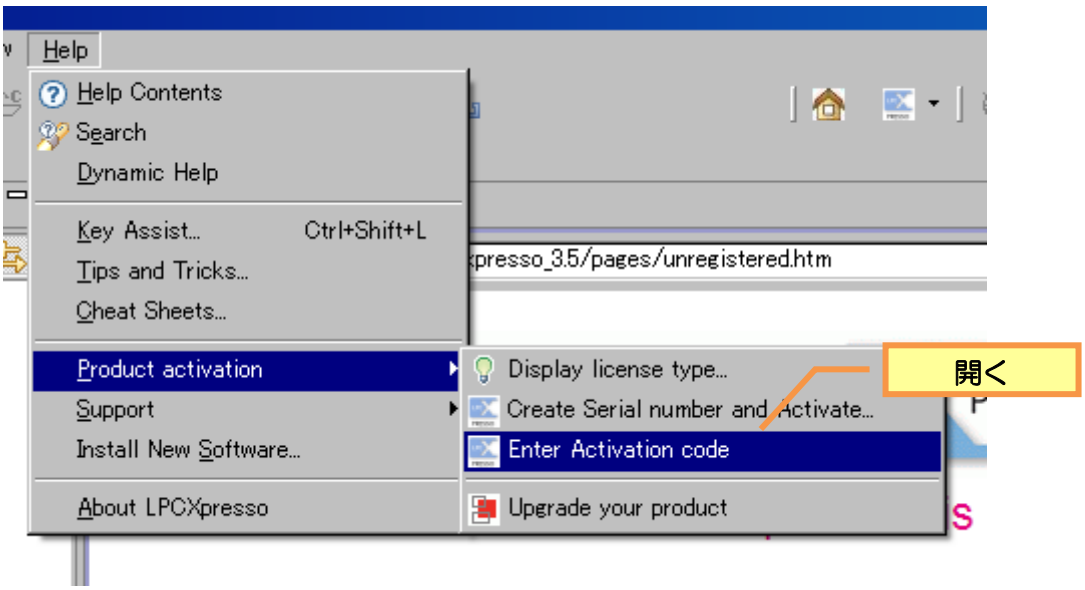
⑧ ログインが完了したら、My　Registrations を押します。

⑨ 先ほどメモしたシリアルナンバーを記入して、「Send me my activation code」ボタンを押します。

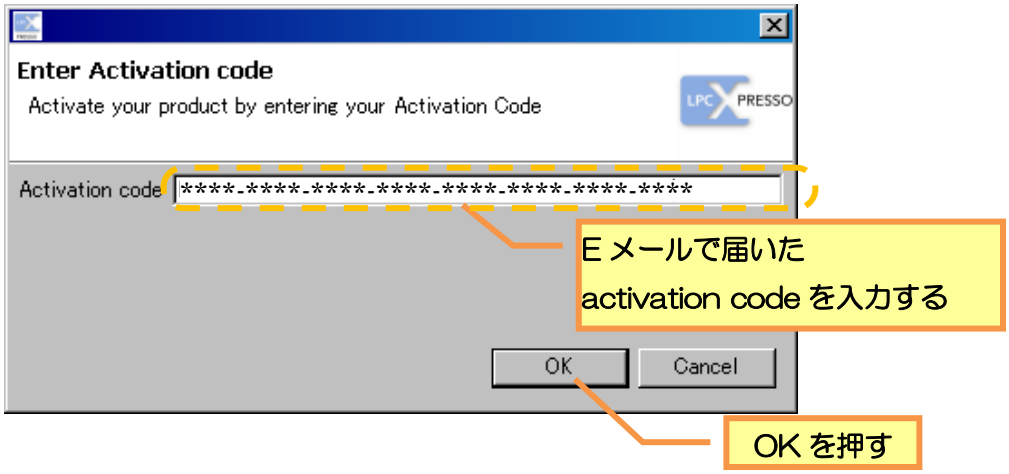
⑩ ボタンを押すと、以下のように表示され、アカウントを登録したメールアドレスに「activation code」が記載されたＥメールが届きます。



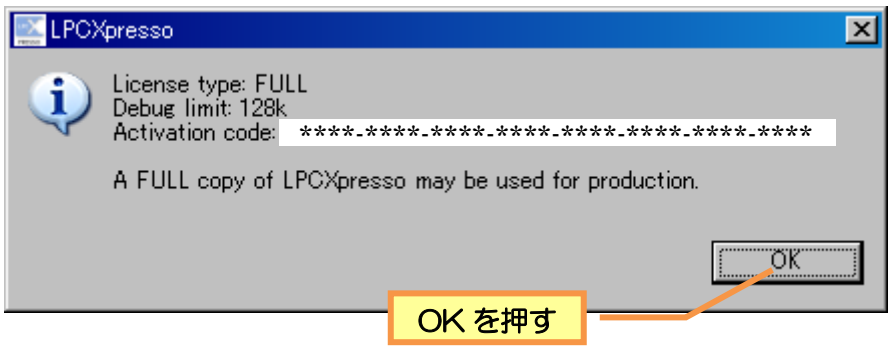
⑪ Help>Product activation>Enter Activation code をクリックし、アクティベーションコード入力画面を開きます。



⑫ Ｅメールに記載のある「activation code」を入力します。



⑬ 認証が正常に完了すると、以下のように表示され、制限が解除されます。OK を押します。



以上で、LPCXpresso の認証作業は完了です。

6-4　サンプルプロジェクトのインポートとビルド

いよいよ、C 言語サンプルプログラムを読み込み、CPU へ書き込みます。ここで使用するサンプルプログラムは、各種設定を行った状態のプロジェクト単位でサンプルを配布していますので、「サンプルプロジェクト」と呼びます。以下の手順に沿って、サンプルプロジェクトのインポートとビルドを行ってください。

(1)ワークスペースフォルダの変更

LPCXpresso をインストールした際に、以下の場所に標準のワークスペースが作られます。**このとき、ユーザー名に全角文字（日本語など）が含まれる場合、正常にビルドできませんので、ワークスペースフォルダの変更を行ってください。**

・WindowsXP の場合

C:¥Documents and Settings¥[ユーザー名※]¥My Documents¥lpcxpresso_3.5¥workspace

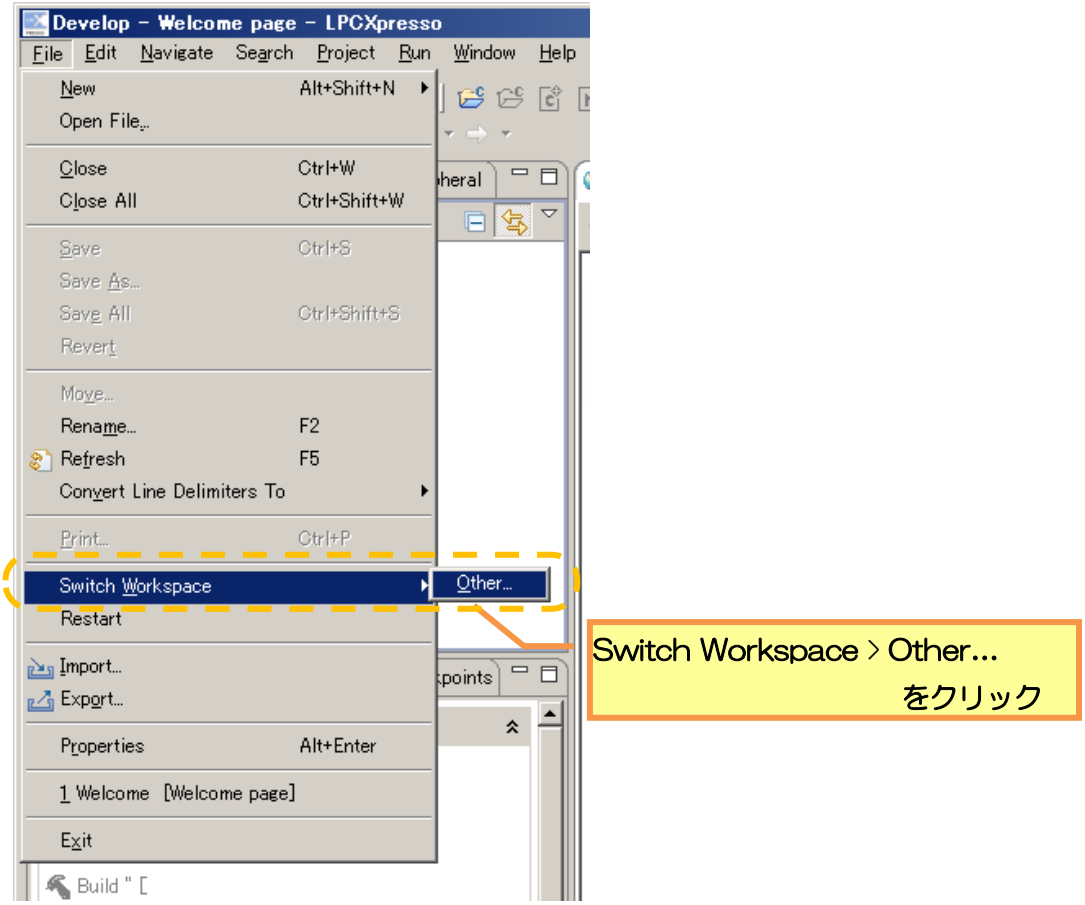
・Windows７の場合

C:¥Users¥[ユーザー名※]¥Documents¥lpcxpresso_3.5¥workspace

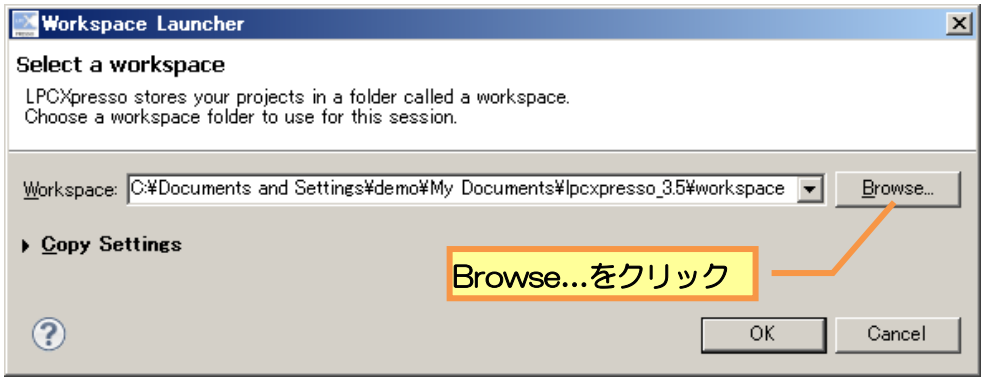
※ユーザー名は Windows にログインしているユーザー名です。

※上記説明がよくわからない場合、以下の手順でワークスペースフォルダの変更を行ってください。

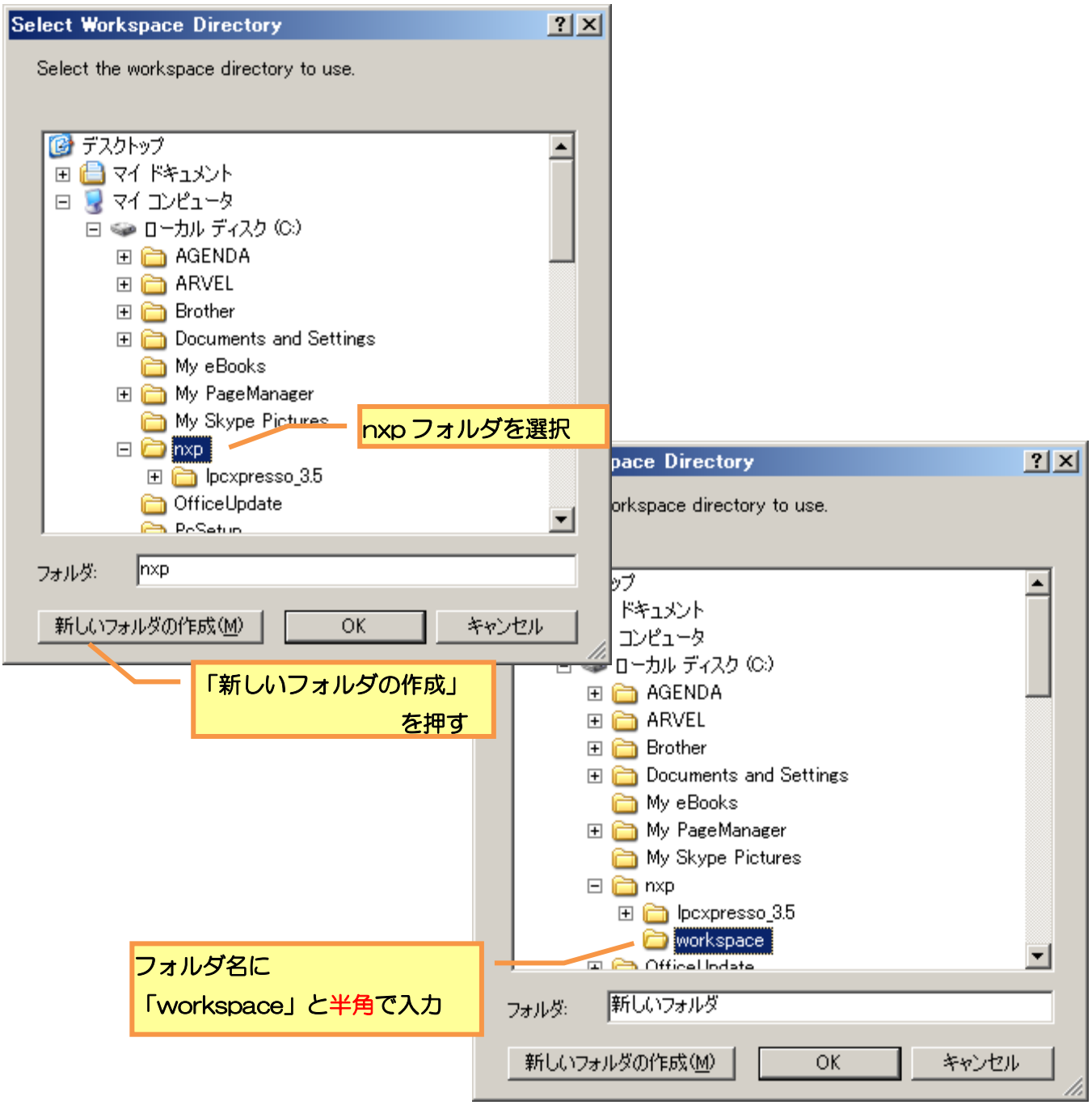
① LPCXpresso のファイルメニューから「Switch Workspace > Other...」を選択しクリックします。



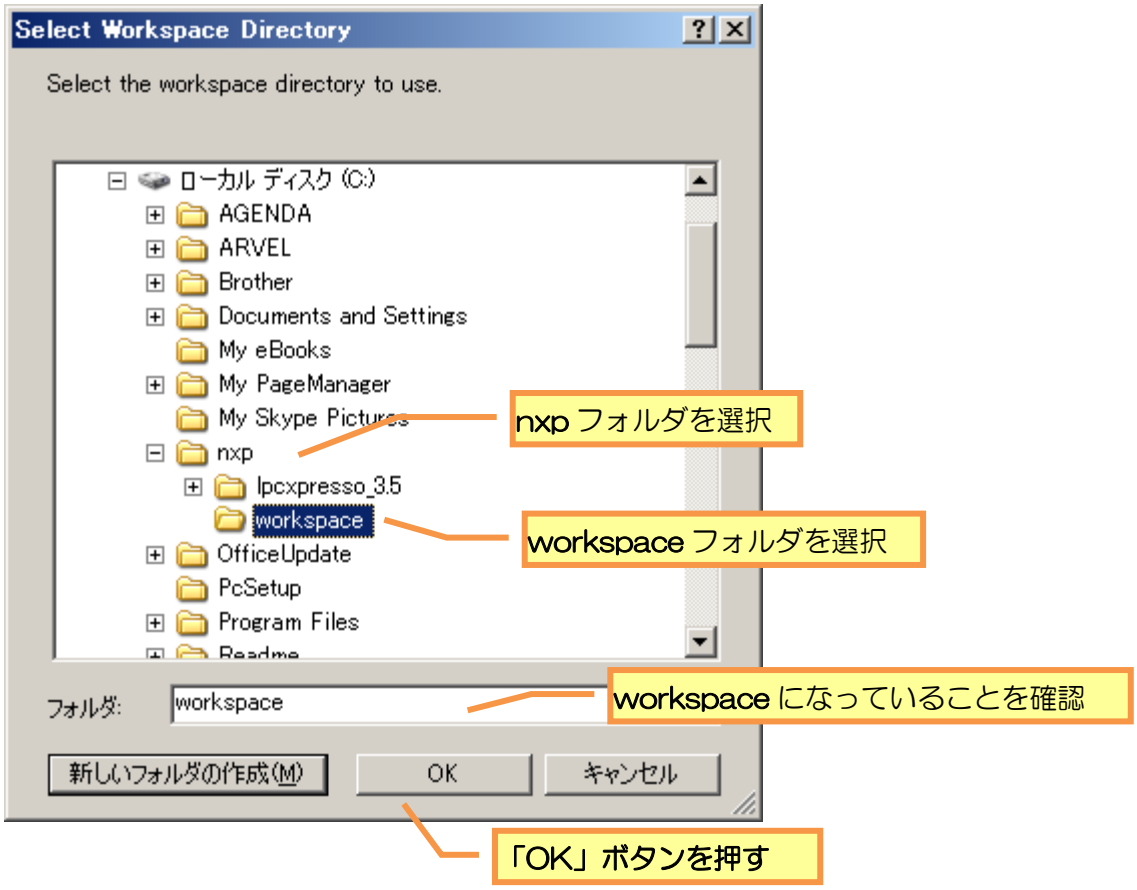
② 「Brows...」をクリックします。



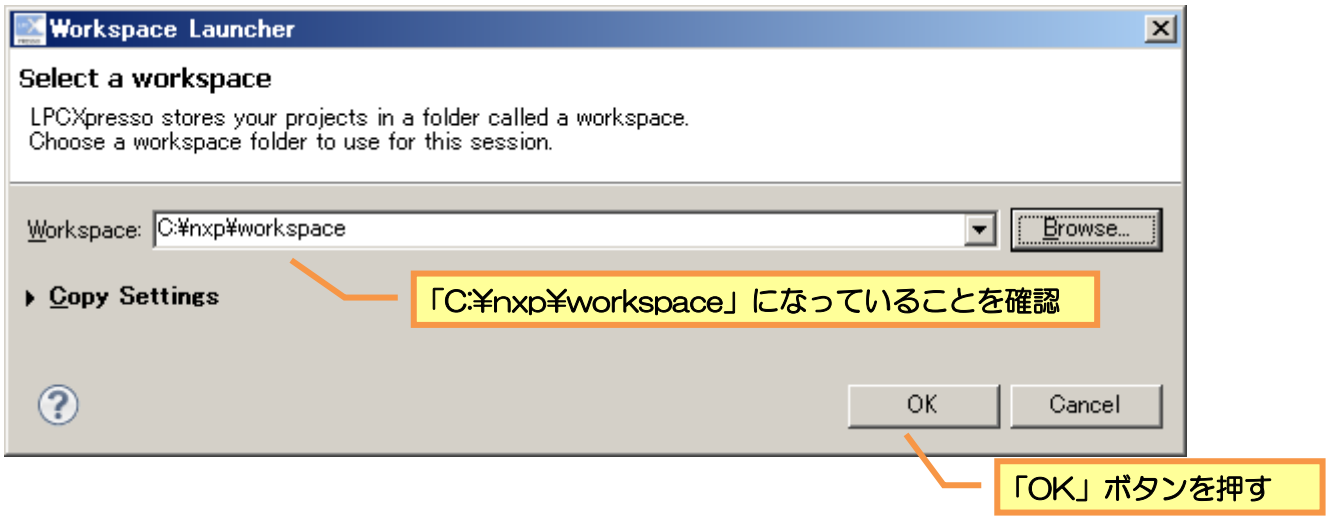
③ マイコンピュータ内の「**ローカルディスク(C:)＞nxp**」フォルダを選択し、「**新しいフォルダの作成**」ボタンをクリックします。作成された「**新しいフォルダ**」の名前を「**workspace**」に変更します。この時、必ず半角文字で入力してください。



④ このままでは、フォルダが反映されていないので、フォルダ名に直接「workspace」と入力するか、以下の手順で、一度別のフォルダをクリックした後に、再度「workspace」フォルダをクリックし、**フォルダ名が「worksoace」になっていることを確認し、「OK」ボタンを押します。**



⑤workspace フォルダが、**「C:¥nxp¥workspace」**になっていることを確認し、**「OK」**ボタンを押します。



以上で、ワークスペースフォルダの変更は完了です。

(2)サンプルプロジェクトのインポート

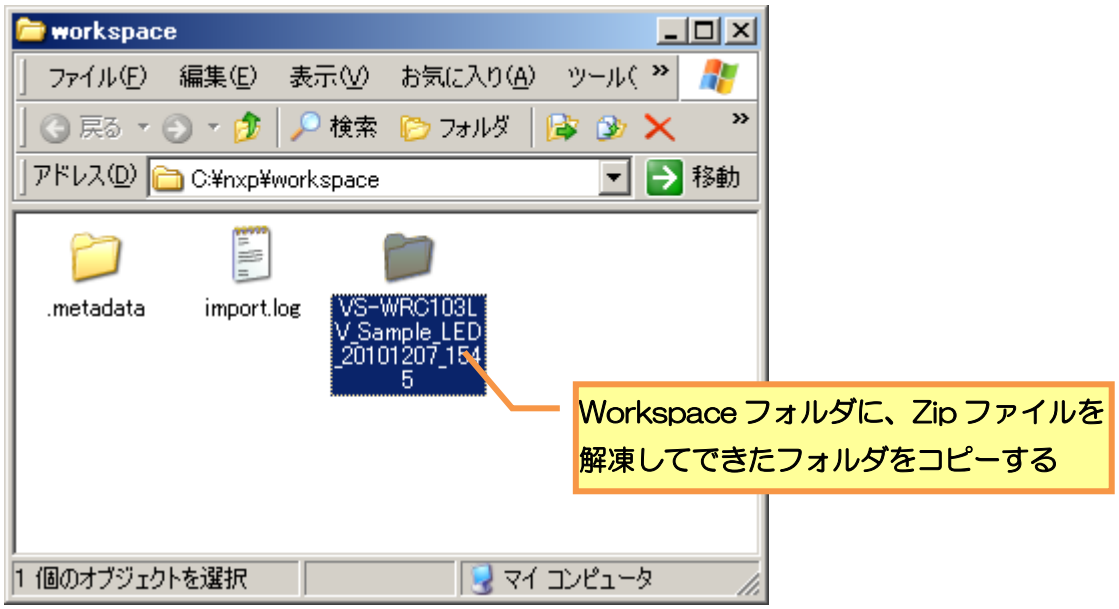
次にワークスペースにサンプルプロジェクトをインポートします。

① 以下の URL より、ビュートミニ ARM サポートページにアクセスし、LED 点滅サンプルプロジェクト「VS-WRC103LV_Sample_LED_******.zip」(****はバージョン)をダウンロードします。

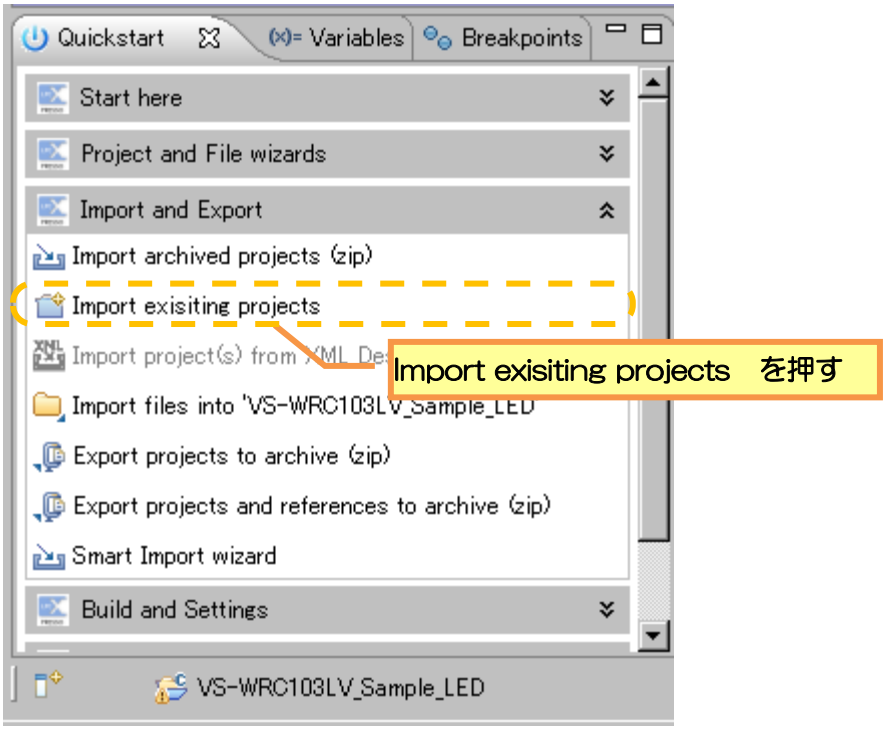
http://www.vstone.co.jp/products/beauto_mini_arm/download.html

※ サポートページで公開しているものが最新になりますが、 ビュートミニ ARM の CD-ROM をお持ちの場合、CD-ROM 内の「C 言語_ARM¥サンプルプロジェクト」フォルダ内にも、各種サンプルを同封しています。

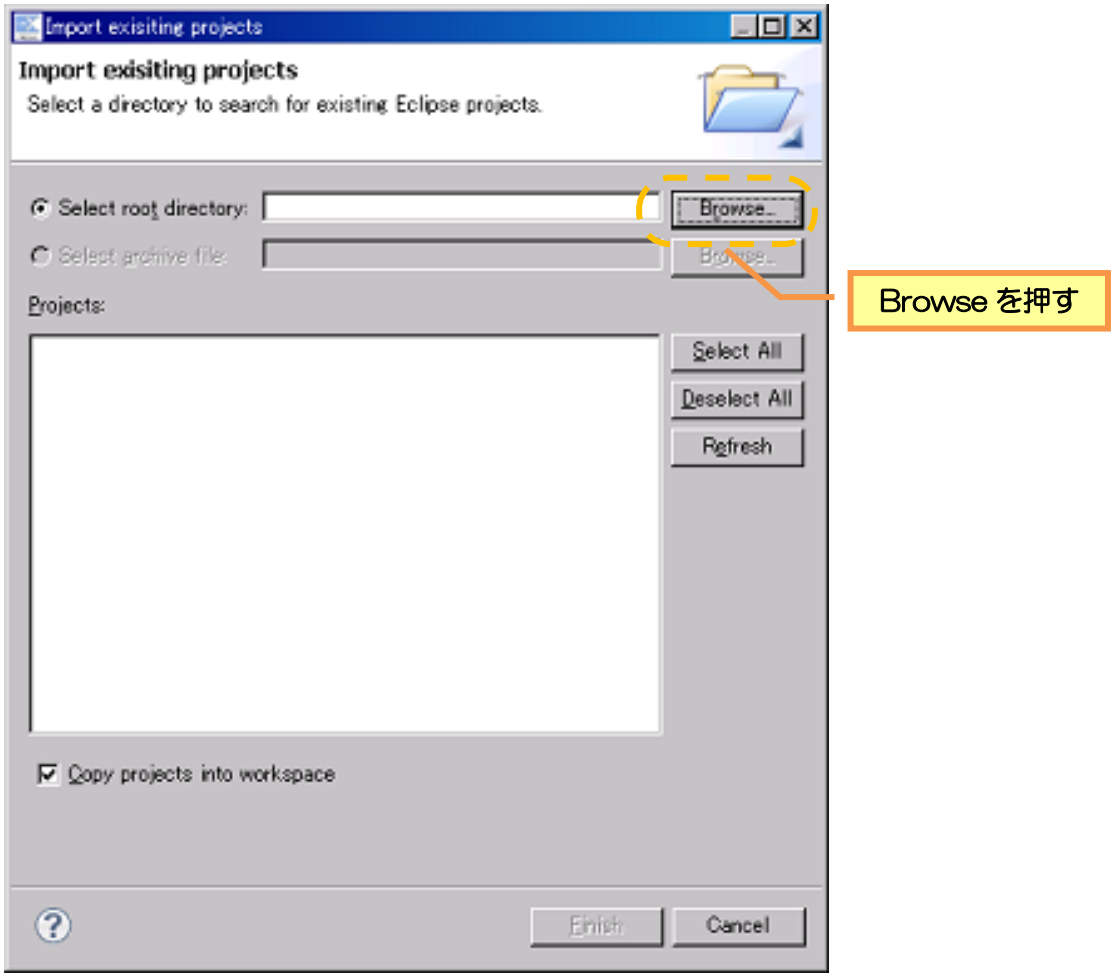
② Zip ファイルを解凍したフォルダをワークスペースフォルダにコピーします。(1)でワークスペースフォルダを変更した場合、「C:¥nxp¥workspace」にコピーします。



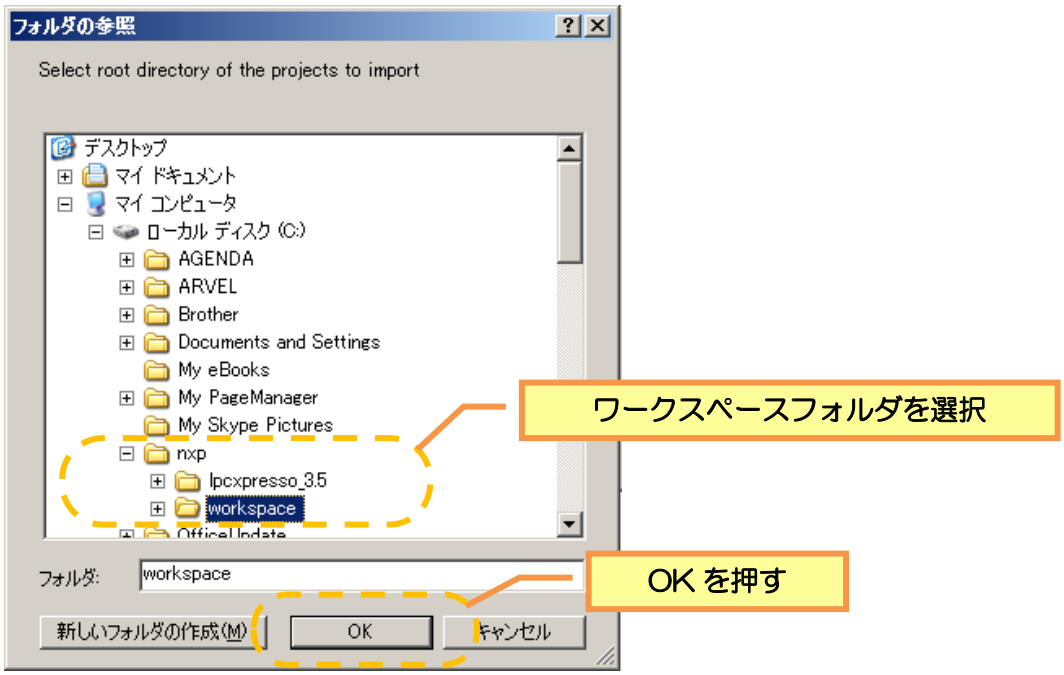
③ LPCXpresso を起動し、画面左下の「Quickstart Panel」の「Import and Export」より「Import existing projects」を開きます。



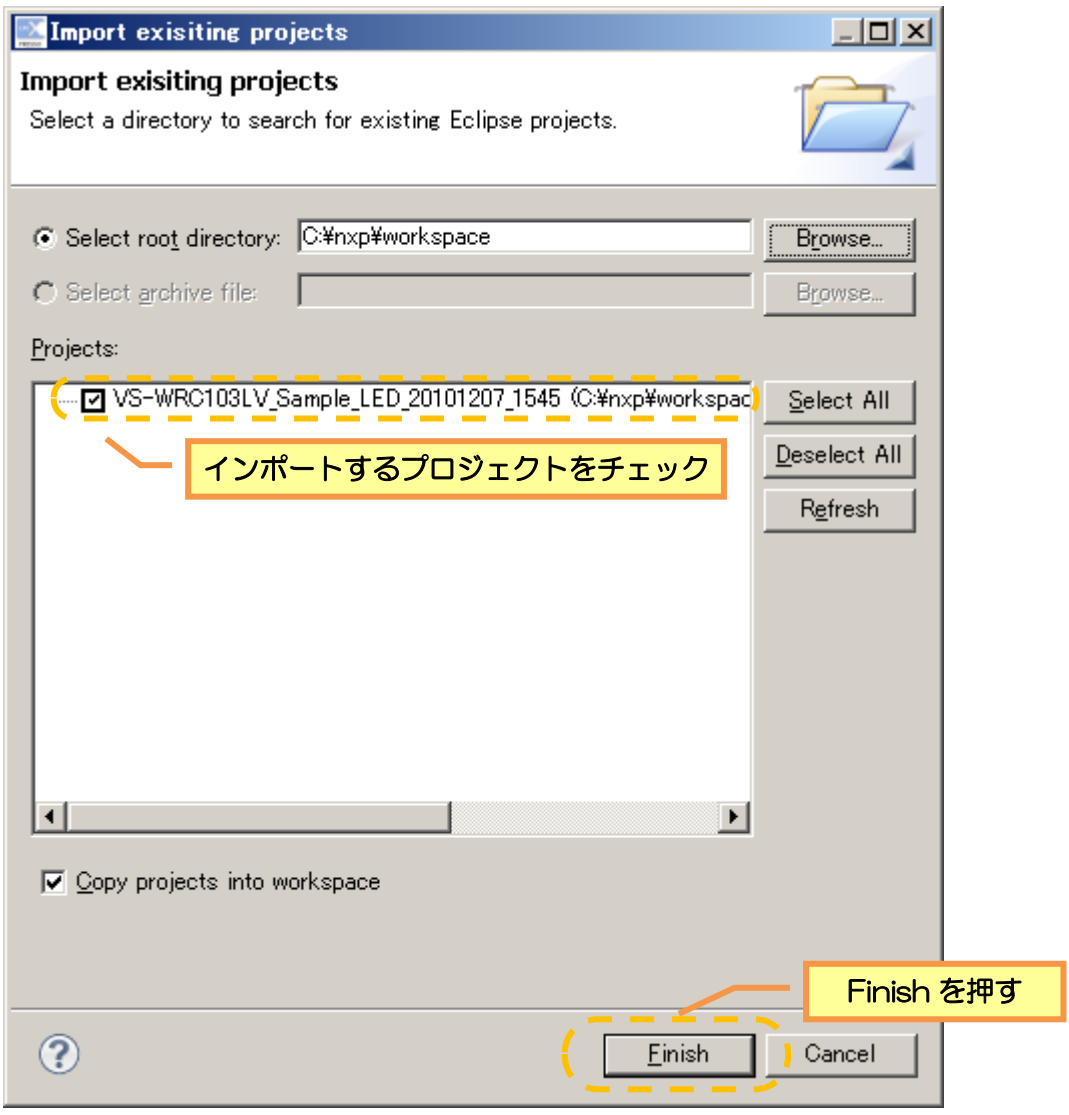
④ 以下のダイアログが表示されるので、「Browse」ボタンを押します。



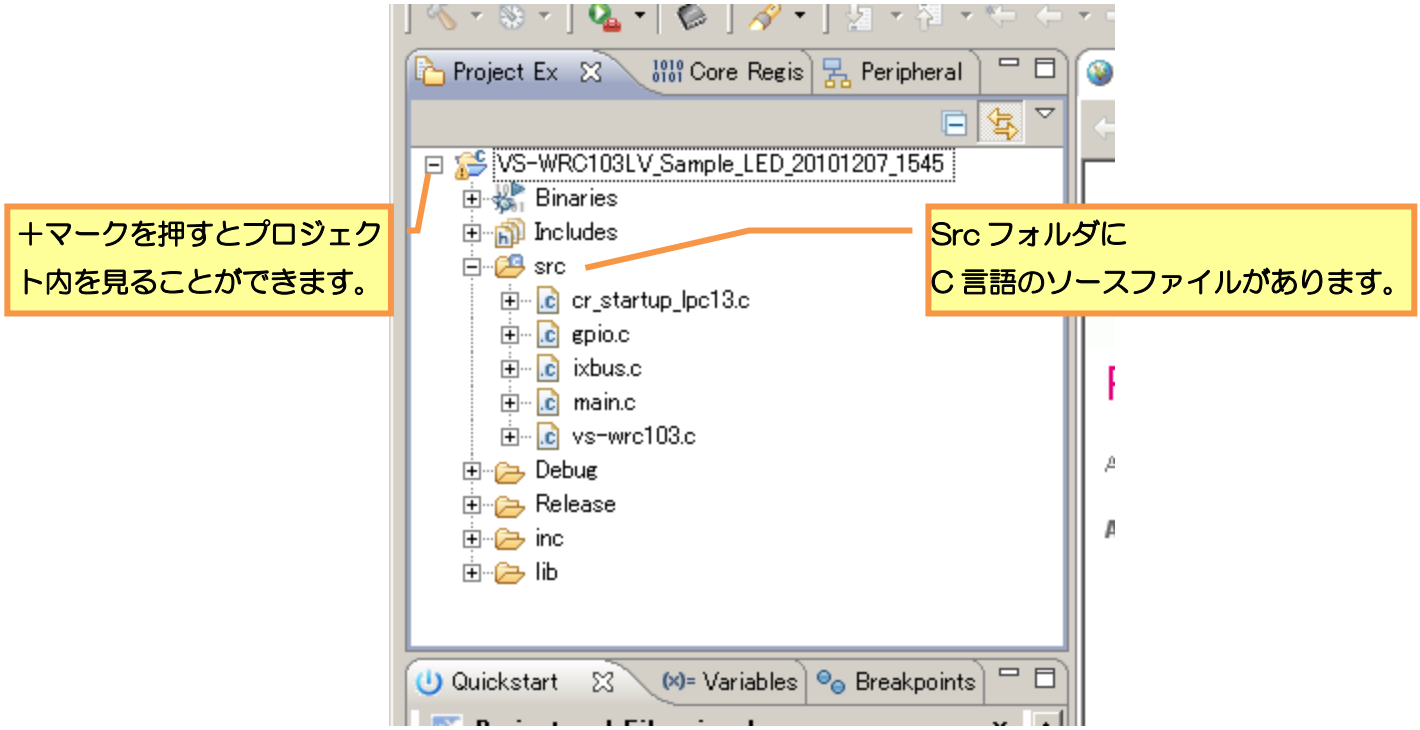
⑤ サンプルプロジェクトを保存した、ワークスペースフォルダ（C:¥nxp¥workspace）を選択し、**OK** を押します。



⑥ インポートするプロジェクト（ここでは、「VS-WRC103LV_Sample_LED_********_****」(****はサンプルのバージョンの数字が入ります)）をチェックし、**「Finish」**を押します。



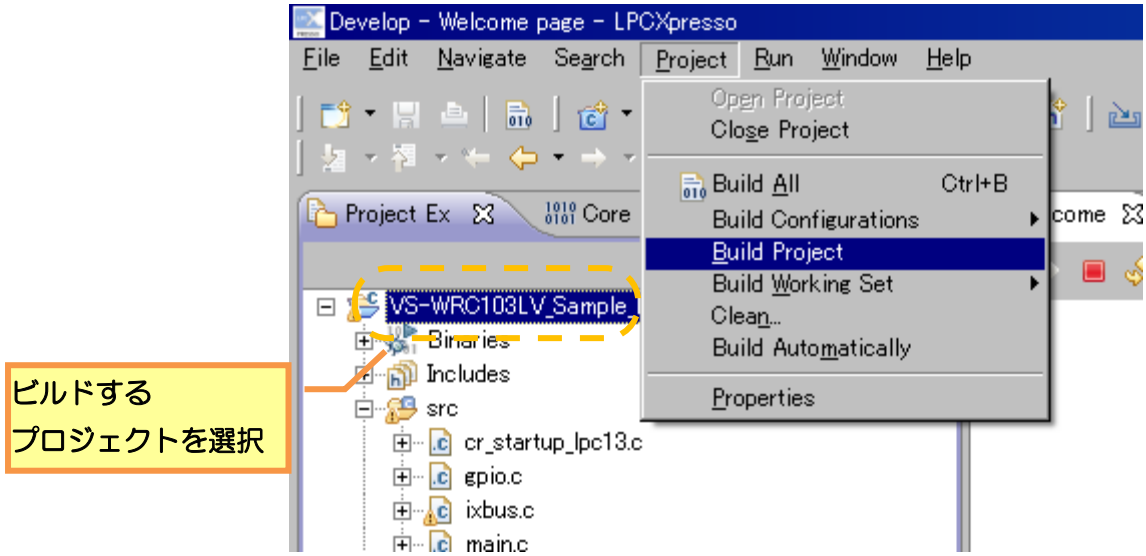
⑦ 「Finish」をクリックすると、画面左上の「Project Ex」にプロジェクトが追加されます。これでプロジェクトの読み込みは完了です。



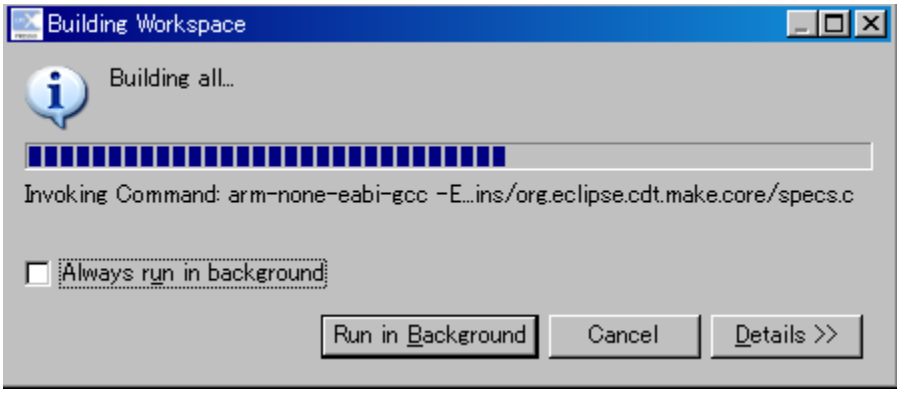
(3) サンプルプロジェクトのビルド

次に、プログラムを CPU ボードに形式に変換する作業のビルドを行います。以下の手順に従って、ビルドを行ってください。

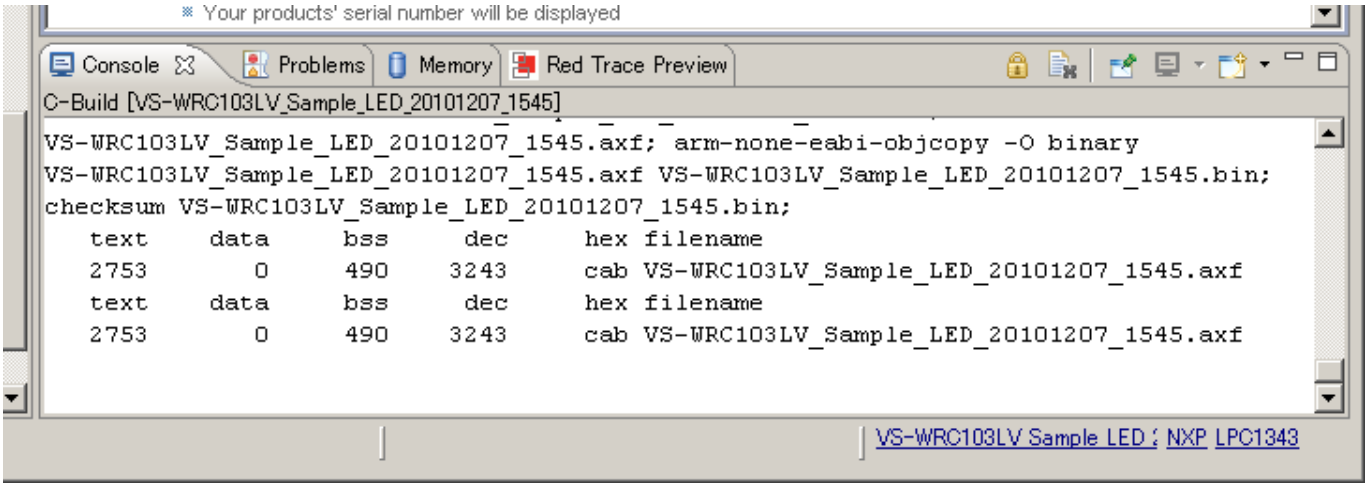
① 「Project Ex」でビルドしたいプロジェクト（ここでは、VS-WRC103LV_Sample_LED）を選択し、Project＞Build Projecrt をクリックしてビルドを開始します。



② ビルドを開始すると以下のダイアログが表示されますので、完了するまで待ちます。



③ 完了すると、ダイアログが閉じ、以下のように表示されます。



※ ここで Error が出た場合、(1)のワークスペースフォルダの変更が、正常に行えてない可能性があります。
再度ワークスペースフォルダの変更を行ってください。
（新しいフォルダの名前を変更で、「workspace」と入力するときは必ず**半角**で入力してください）

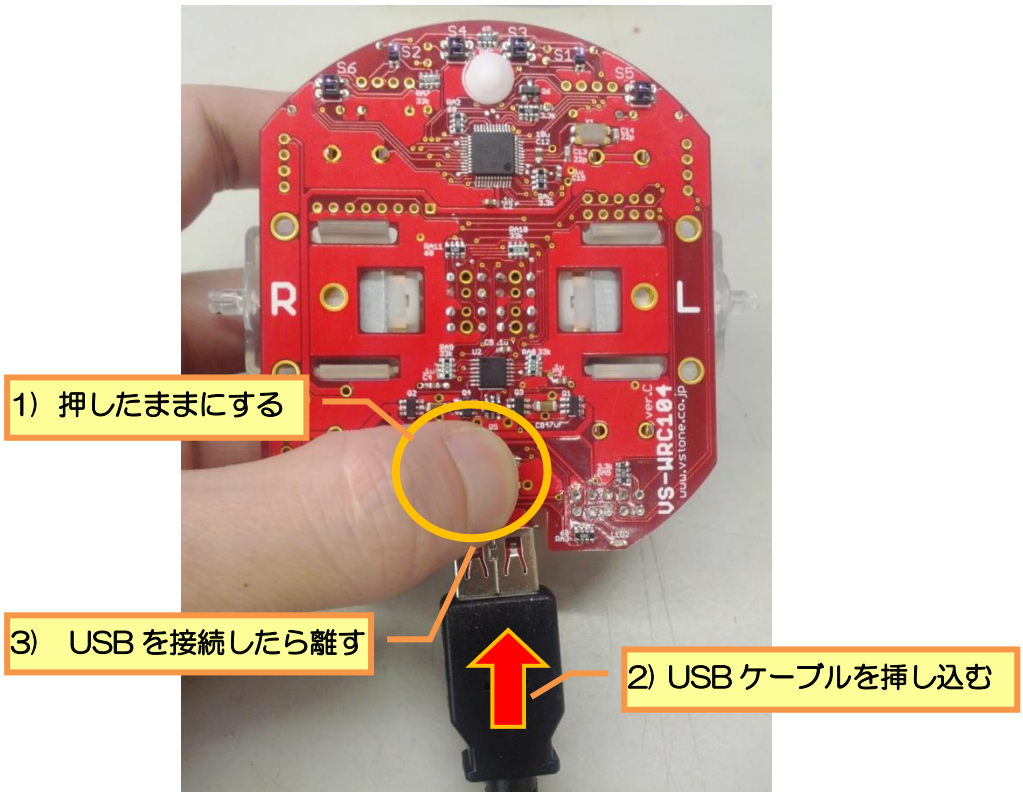
※ C 言語のソースを編集した後に、Error と表示された場合の対処方法は「6-7 エラーの修正」をご参考下さい。

6-5　VS-WRC104 へのプログラムの書き込み

サンプルプロジェクトをビルドすると、**ワークスペースフォルダ　¥ VS-WRC103LV_Sample_LED ¥ Debug** 内に、「**VS-WRC103LV_Sample_LED.bin**」というファイルが作成されます。

このファイルを CPU ボードに書き込むと、作成したプログラムを CPU ボードに実行させることができますので、以下の手順で、書き込みを行ってください。

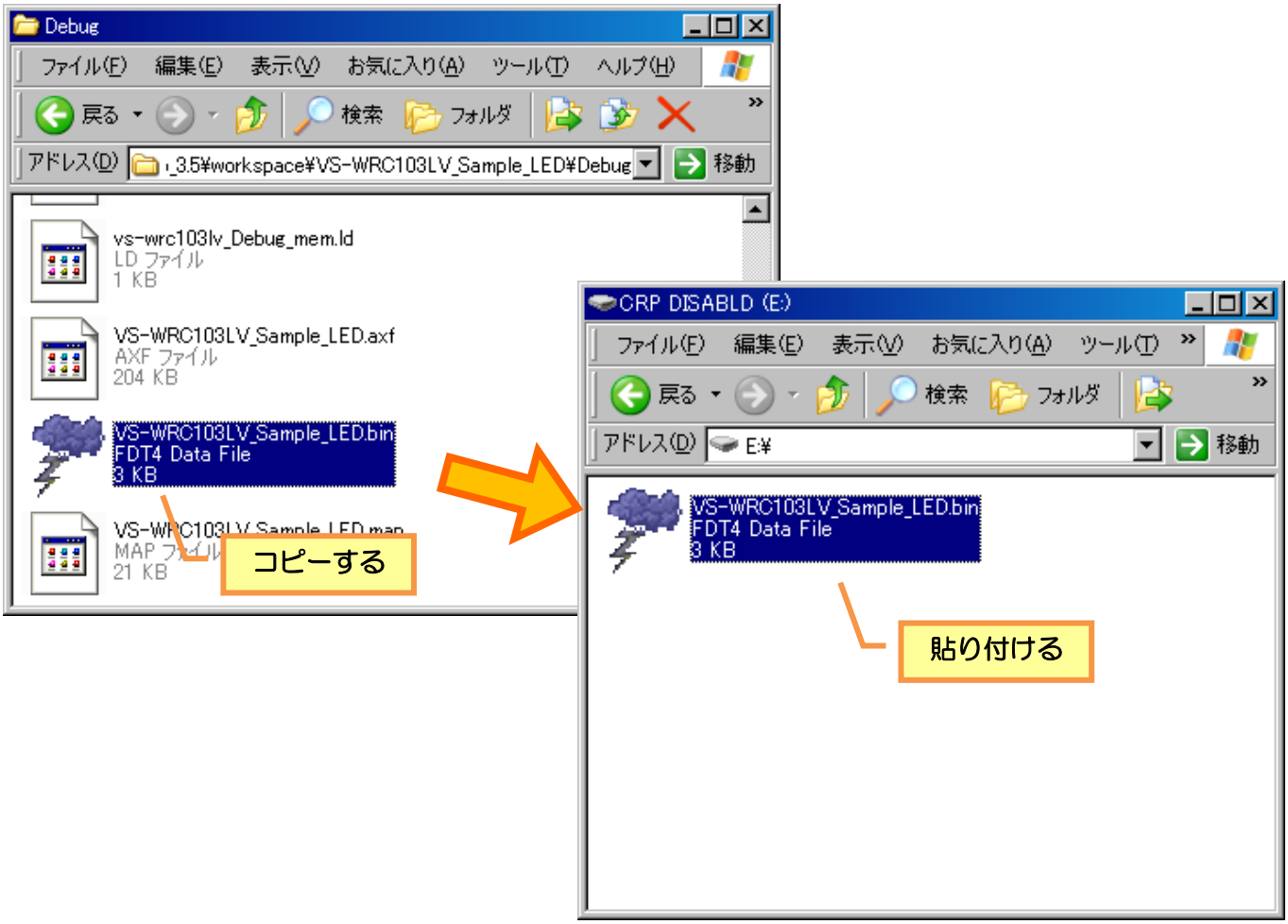
① USB ケーブルを外し、CPU ボードの**全ての電源を OFF** にします。
② **プッシュボタンを押したまま、**PC と USB ケーブルで接続します。
③ USB を接続したらプッシュボタンは離しても OK です。
④ 20 秒程度待つと、PC が CPU ボードを USB メモリなどと同じ「マスストレージデバイス」として認識します。（初めて CPU ボードを PC に接続した場合、認識にしばらく時間がかかります。）



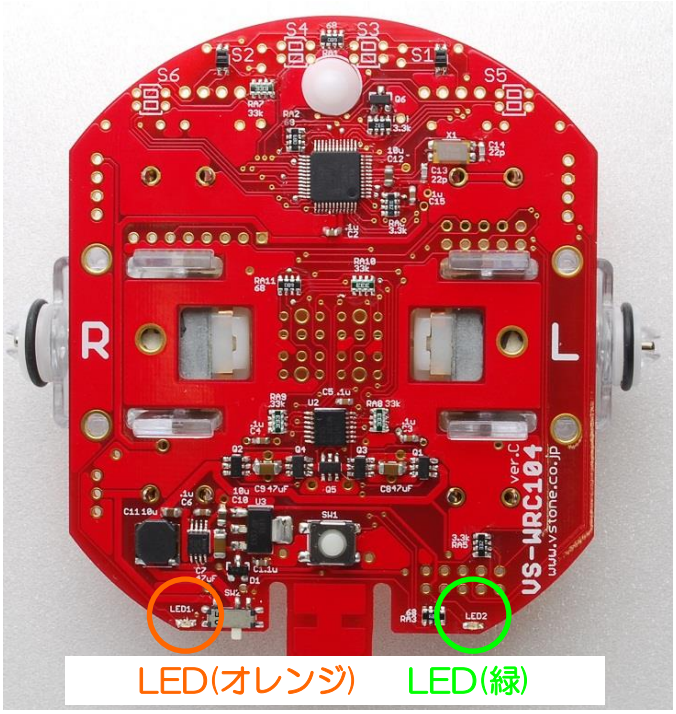
⑤ CPU ボードが PC に認識されると、「**CRP DISABLE**」という名称のドライブ名で表示されます。CPU ボードを接続したら「**マイコンピュータ**」を開いて、この名称のドライブが表示されることを確認し、**ダブルクリックしてドライブを開きます。**

⑥ 「CRP DISABLE」を開くと、中に「**firmware.bin**」というファイルが一つ入っていますので、このファイルを**削除**します。

⑦ ワークスペースフォルダ ￥VS-WRC103LV_Sample_LED￥Debug 内の、
「**VS-WRC103LV_Sample_LED.bin**」を**コピー**して、「**CRP DISABLE**」内に**貼り付け**ます。



⑧ PC から CPU ボードを取り外し、再度電源を入れると、書き込んだプログラムに従って CPU ボードが動作します。LED サンプルプログラムの書き込みが正常に完了していると、オレンジ、緑の LED が交互に点灯します。



以上で、プログラムの書き込み手順は完了です。

6-6　LED 点滅プログラムについて

サンプルの LED 点滅プログラムは、関数 LED()、Wait()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-WRC103LV.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 main()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 main()関数から実行されます。LED 点滅プログラムでは、main()は led.c 内にあります。

```
1:      void main(void)
2:      {
3:              //制御周期の設定［単位：Hz　範囲：30.0~］
4:              const unsigned short MainCycle = 60;
5:
6:              Init(MainCycle);            //CPU の初期設定
7:              //I2C_init(void); //IXBUS 初期化
8:
9:              //ループ
10:             while(1){
11:                     LED(1);         //緑の LED 点灯
12:                     Wait(1000);     //1000msec 待つ
13:                     LED(2);         //オレンジの LED 点灯
14:                     Wait(1000);     //1000msec 待つ
15:             }
16:     }
```

main 関数の各行について説明します。

0：関数の宣言

４～７：各機能の初期設定

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

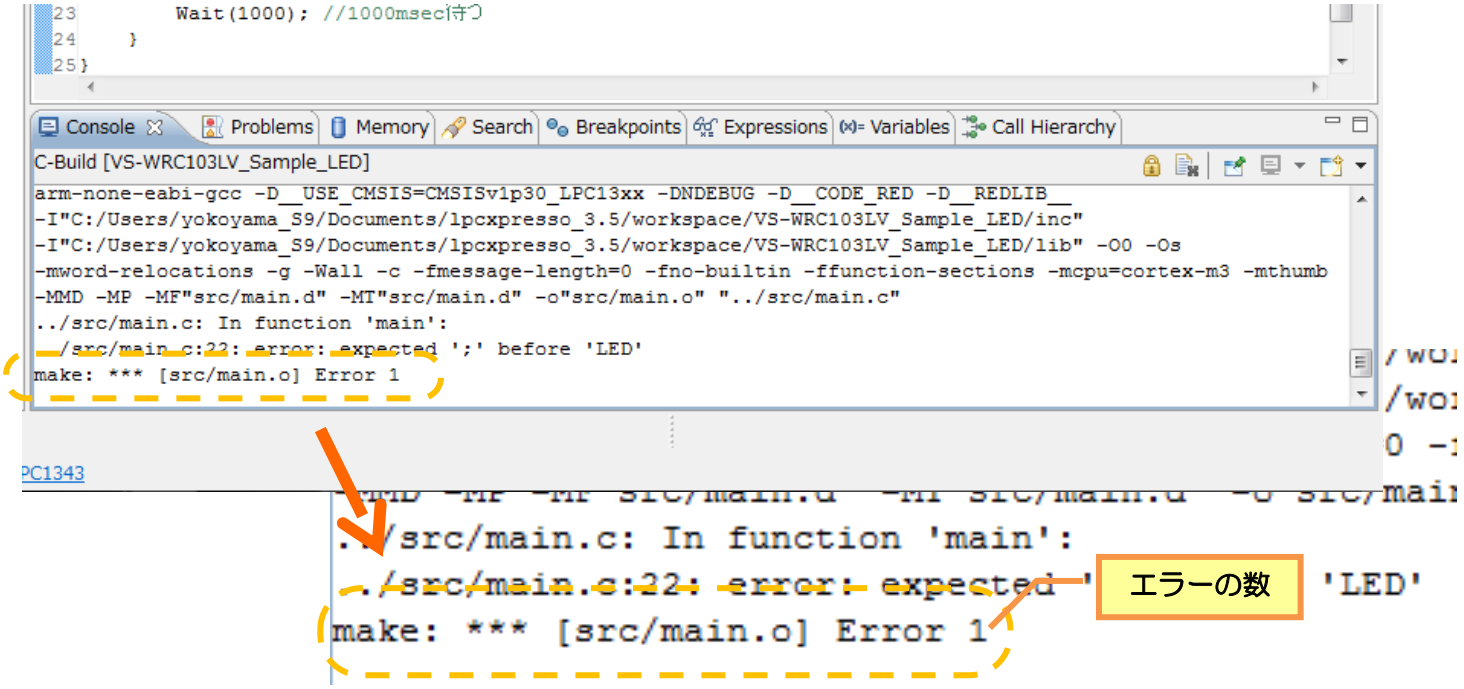
LED()関数と Wait()関数を利用して LED を交互に点滅させています。

各関数については、本マニュアルの５章、またはサンプルプロジェクト内のヘッダファイル「vs-wrc103lv.h」で簡単に解説していますので、そちらをご覧ください。

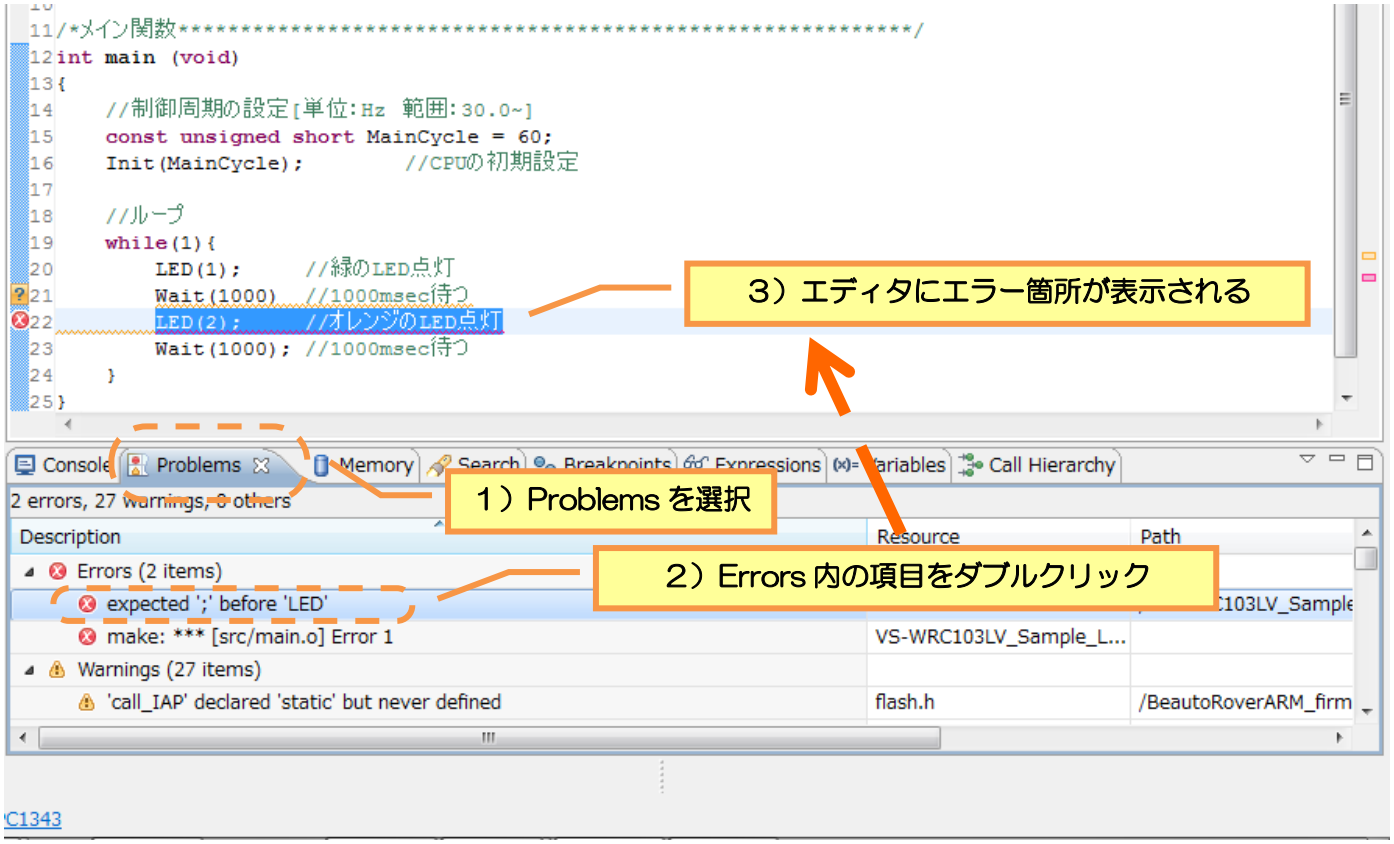
6-7　エラーの修正

サンプルプロジェクトのプログラムを変更し、ビルドすると構文の間違いなどでビルドエラーが発生することがあります。その場合、以下の手順で、エラーの発生した箇所を確認し、修正してください。

①プログラムにエラーがあった場合、コンソールに以下のようなビルド結果が表示されます。
　Error の後の数値は、いくつエラーがあったかその数が表示されます。



②エラーの位置を確認するためには、「Problems」タブを選択し、Errors 内のいずれかのエラーをダブルクリックすると、エディタ上で、エラー箇所が表示されます。



③ここでのエラーは、LED(2);の前のWait(1000);に「；（セミコロン）」がないことによるものですので、そこを修正して再度ビルドします。

```
12 int main (void)
13 {
14     //制御周期の設定[単位:Hz 範囲:30.0~]
15     const unsigned short MainCycle = 60;
16     Init(MainCycle);          //CPUの初期設定
17
18     //ループ
19     while(1){
20         LED(1);       //緑のLED点灯
21         Wait(1000)    //1000msec待つ
22         LED(2);       //オレンジのLED点灯
23         Wait(1000); //1000msec待つ
24     }
25 }
```

④エラーがなくなり、正常にビルドが完了すると以下のように表示されます。

```
12 int main (void)
13 {
14     //制御周期の設定[単位:Hz 範囲:30.0~]
15     const unsigned short MainCycle = 60;
16     Init(MainCycle);          //CPUの初期設定
17
18     //ループ
19     while(1){
20         LED(1);       //緑のLED点灯
21         Wait(1000); //1000msec待つ
22         LED(2);       //オレンジのLED点灯
23         Wait(1000); //1000msec待つ
24     }
25 }
```

Console | Problems | Memory | Search | Breakpoints | Expressions | Variables | Call Hierarchy

C-Build [VS-WRC103LV_Sample_LED]

```
arm-none-eabi-size VS-WRC103LV_Sample_LED.axf; arm-none-eabi-size VS-WRC103LV_Sample_LED.axf;
arm-none-eabi-objcopy -O binary VS-WRC103LV_Sample_LED.axf VS-WRC103LV_Sample_LED.bin; checksum
VS-WRC103LV_Sample_LED.bin;
   text    data     bss     dec     hex filename
   2753       0     490    3243     cab VS-WRC103LV_Sample_LED.axf
   text    data     bss     dec     hex filename
   2753       0     490    3243     cab VS-WRC103LV_Sample_LED.axf
```

Smart Insert | 23 : 33

C1343

以上で、エラーの修正は完了です。エディタに表示した箇所に関連する箇所で問題が起こっている可能性もありますので、その点にも注意してエラーを修正しましょう。

■オプションパーツ、関連商品のご購入は・・・

No.1 の品揃え！　各種オプションパーツ、ロボット関連製品のご購入はコチラ

http://www.vstone.co.jp/robotshop/

楽天・Amazon・Yahoo の各 Web 店舗、または東京、福岡の各ロボットセンター店頭でもロボット関連商品をお買い求めいただけます。

ロボットセンター東京秋葉原店　(東京支店)

〒101-0021

東京都千代田区外神田 1-9-9 内田ビル 4F

ロボットセンターロボスクエア店(福岡支店)

〒814-0001

福岡市早良区百道浜 2-3-2

TNC 放送会館 2F ロボスクエア内

商品に関するお問い合わせ

商品の技術的なご質問は、問題・症状・ご使用の環境などを記載の上メールにてお問い合わせください。

E-mail: **infodesk@vstone.co.jp**　　受付時間　：10:00～17:00（土日祝日は除く）

ヴイストン株式会社

〒555－0012　大阪市西淀川区御幣島 2-15-28

TEL：06-4808-8701　FAX：06-4808-8702

Vstone®

www.vstone.co.jp