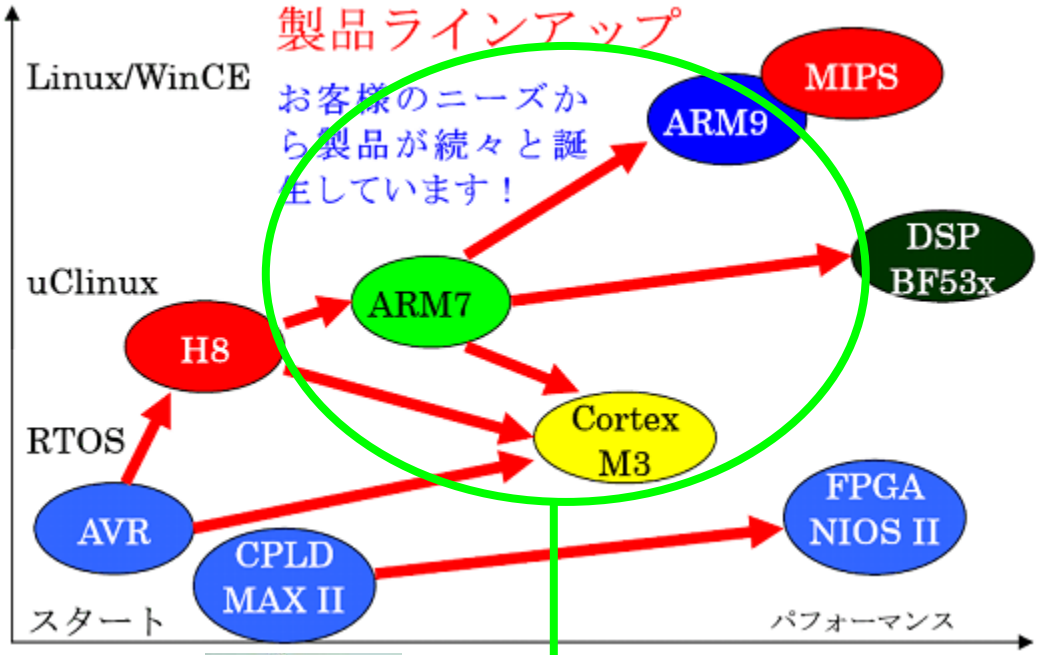
Eclipse+OpenJTAG +OpenOCD で ARM シリーズ開発環境構築

株式会社日昇テクノロジー
http://www.csun.co.jp
info@csun.co.jp

Ver1.4

2011/09/28



OpenJTAG で ARM シリーズを開発可能にする

修正履歴

NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2009/07/31
2	Ver1.1	誤った環境構築用、ワーク用のパスを訂正	2010/07/17
3	Ver1.2	P58～P63 の説明画面の訂正	2010/07/20
4	Ver1.3	フォルダ名誤記訂正 ARM EABI「arm-none-eabi-gcc」確認コマンド誤記訂正 5.2 章 Zylin Embedded CDT のインストール手順変更	2010/11/22
5	Ver1.4	Linux 環境での開発手順を追加	2011/09/28

第一章 背景

近年，ARM プロセッサが急速に広まっています．さらに、ARM9，ARM11，ARM-M，Cortex といった新たなアーキテクチャが次々と発表されています．これらの新しいプロセッサでは、従来から存在したARMが拡張され、さらにARM9とARM11が追加されました．

本書では、従来から広く利用されているGNU クロス開発ツールを、これらの最新アーキテクチャ/命令セットへ対応させます．最新版のGNU ツールを実際に使って、対応したクロス開発環境を作ります．

組込アプリを開発する際に、2点は非常に重要です。一つは統合開発環境（IDE）となり、もう一つはデバッグ環境です。いろんな統合開発環境ツールがありますが、無料で使用できるオープンソースEclipseプラットフォームはJava アプリを含めて、C/C++アプリを開発用の全世界共通のIDEです、今後、ますます多くの使用者が増えていきます。なお、デバッグツールはオープンソースOpen-On-Chip Debugger(OpenOCD)という便利ツールがあります。

Open-On-Chip Debugger(OpenOCD)は、システムプログラミングにおけるデバッグと埋め込まれた対象装置がないかどうかテストされる境界走査を提供することを目指します。

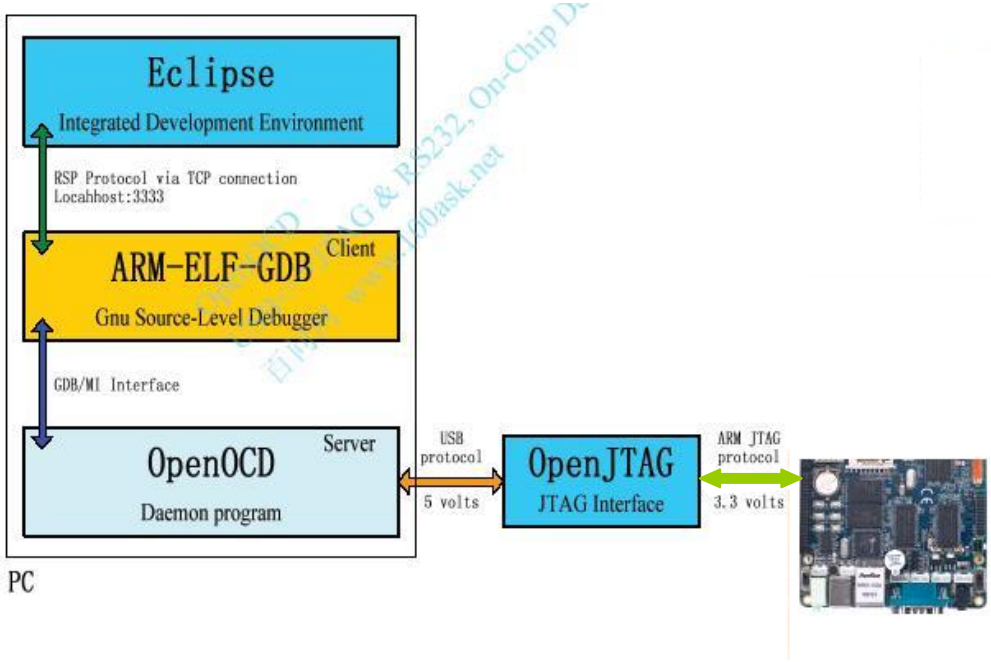
OpenOCDのGDB と接続するためにOpen JTAG コネクトを使う必要です。ただし、パラレルJTAGの転送スピードが遅いので、本書はカスタマイズ化のOpenJTAGを利用し、Eclipse、OpenOCDなどオープンソースとあわせて、ARMシリーズ開発を行います、例えば、ARM基板からダウンロード、書き込み、デバッグを行います。

GDBは以下の基板をデバッグできます。

① ARM7(ARM7TDMIとARM720t)
② ARM9、(ARM920t、ARM922t、 ARM926ej-s、ARM966e-s)
③ XScale(PXA25x、IXP42x)
④ Cortex-M3(LM3とST STM32)

第二章　ARM シリーズ開発の仕組み・イメージ

組込開発のよいプラットフォームは基本的に4つモジュールを含まれます。
① 統合開発環境：IDE（Integrated Development Environment）、例えばEclipse
② クロスコンパイラツールチェイン（Cross Complier tools chain）
③ リモートデバッグツール（例：OpenOCD、Open On-Chip Debugger）
④ 開発用のパソコンとARM基板のコネクタ（JTAG）
全体のイメージは下記の図です。



第三章　用意するもの

以下は Windows 用です。

3.1 ソフトウエアダウンロード URL:

弊社のサーバーから全て必要のものを一括ダウンロードできます。
http://www.dragonwake.com/download/open-jtag/openJTAGIDE.zip
中身に、下記内容を含まれます。
ソフトウエアリスト（名前をクリックしてからツールもそれぞれダウンロードできます）
＊解凍先：お好きな場所を選べられても OK（例：D:¥tmp）

01.arm-2009q1-161-arm-none-eabi.exe
02.openocd-0.1.0.msi
03.yagarto-bu-2.19.1_gcc-4.3.3-c-c++_nl-1.17.0_gi-6.8.50_20090311.exe
04.yagarto-tools-20070303-setup.exe
05.Cygwin
06.jre-6u7-windows-i586-p.exe
07.eclipse-cpp-galileo-win32.zip

3.2　OpenJTAG 用のデバイス(jtag デバッガ、弊社で販売しているもの)

3.2.1　Open-JTAG の購入 URL:

http://www.csun.co.jp/SHOP/200905191.html

3.2.2　デバイスダウンロード URL:

http://www.dragonwake.com/download/open-jtag/open-jtag-driver.zip
①usb-driver(OpenJTAG 用の USB ドライバ)
＊解凍後：D:¥tmp¥open-jtag-driver

3.3　本書使用サンプルソースコード

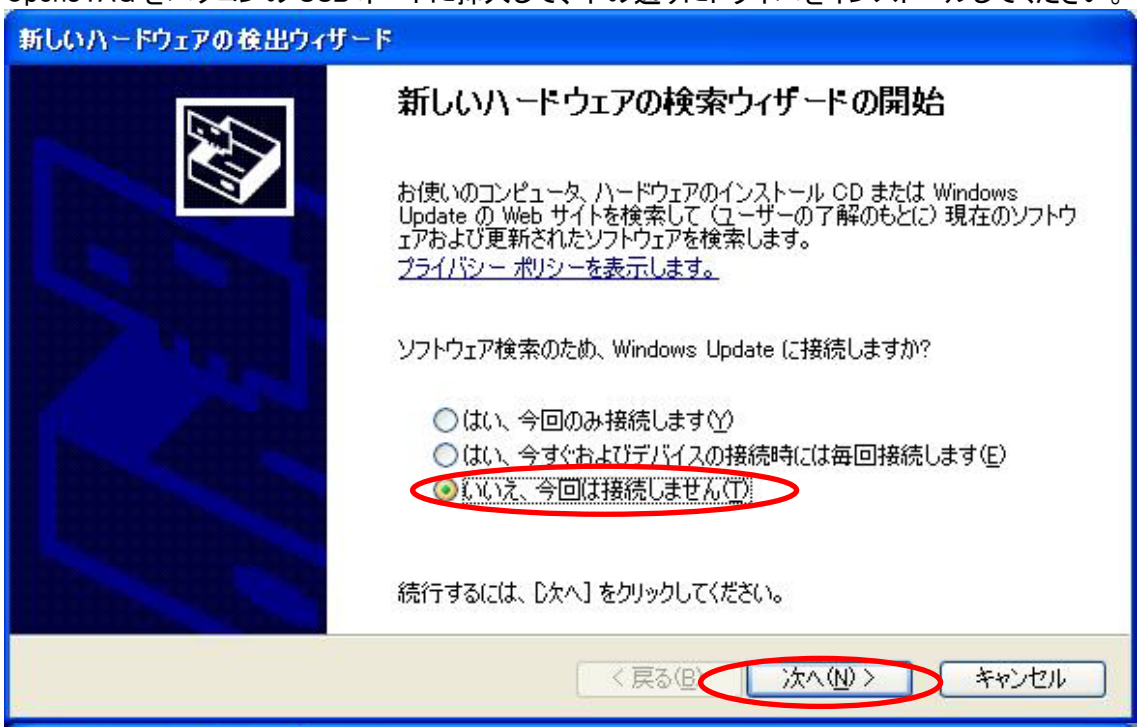
http://www.dragonwake.com/download/open-jtag/gcc_openjtag_eclipse_arm_example.zip
*解凍後：(D:¥OpenJTAG¥ gcc_openjtag_eclipse_arm_example)

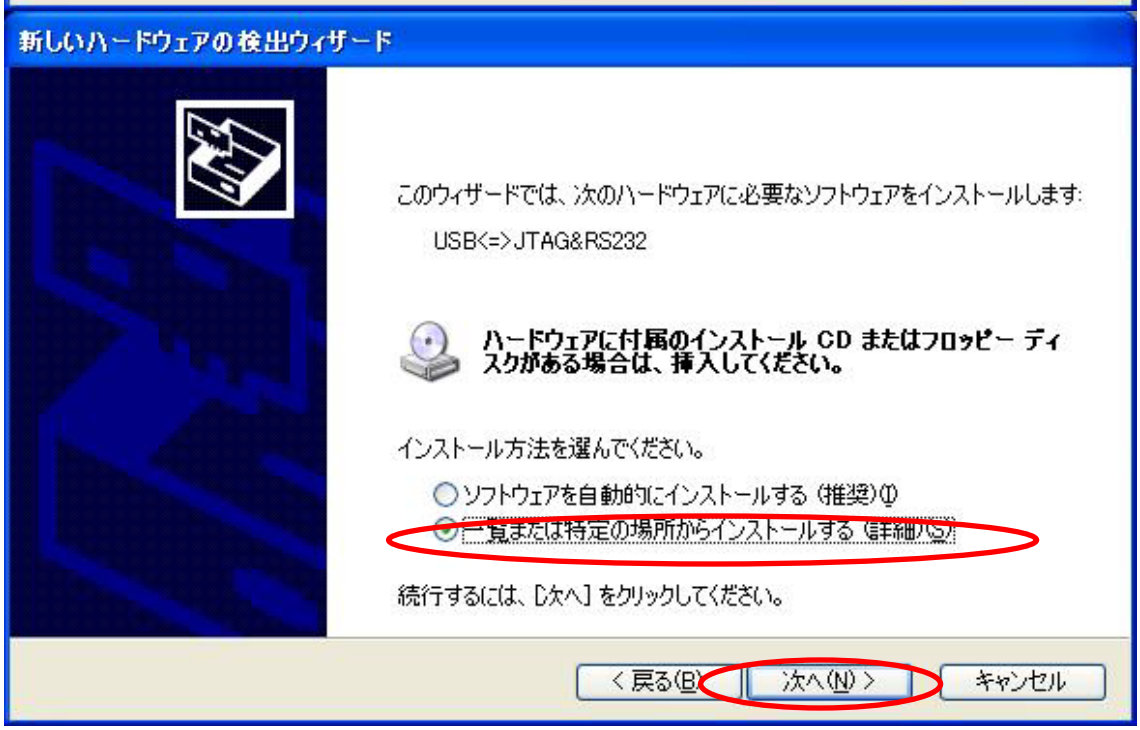
1. 「config」：OpenOCD 用の設定ファイル
2. 「lpc2148」：LPC2148 ボード用のサンプル
3. 「lpc2388」：LPC2388 ボード用のサンプル
4. 「stm32」：stm32FX103 ボード用のサンプル
5. 「mini2440」：mini2440 ボード用のサンプル

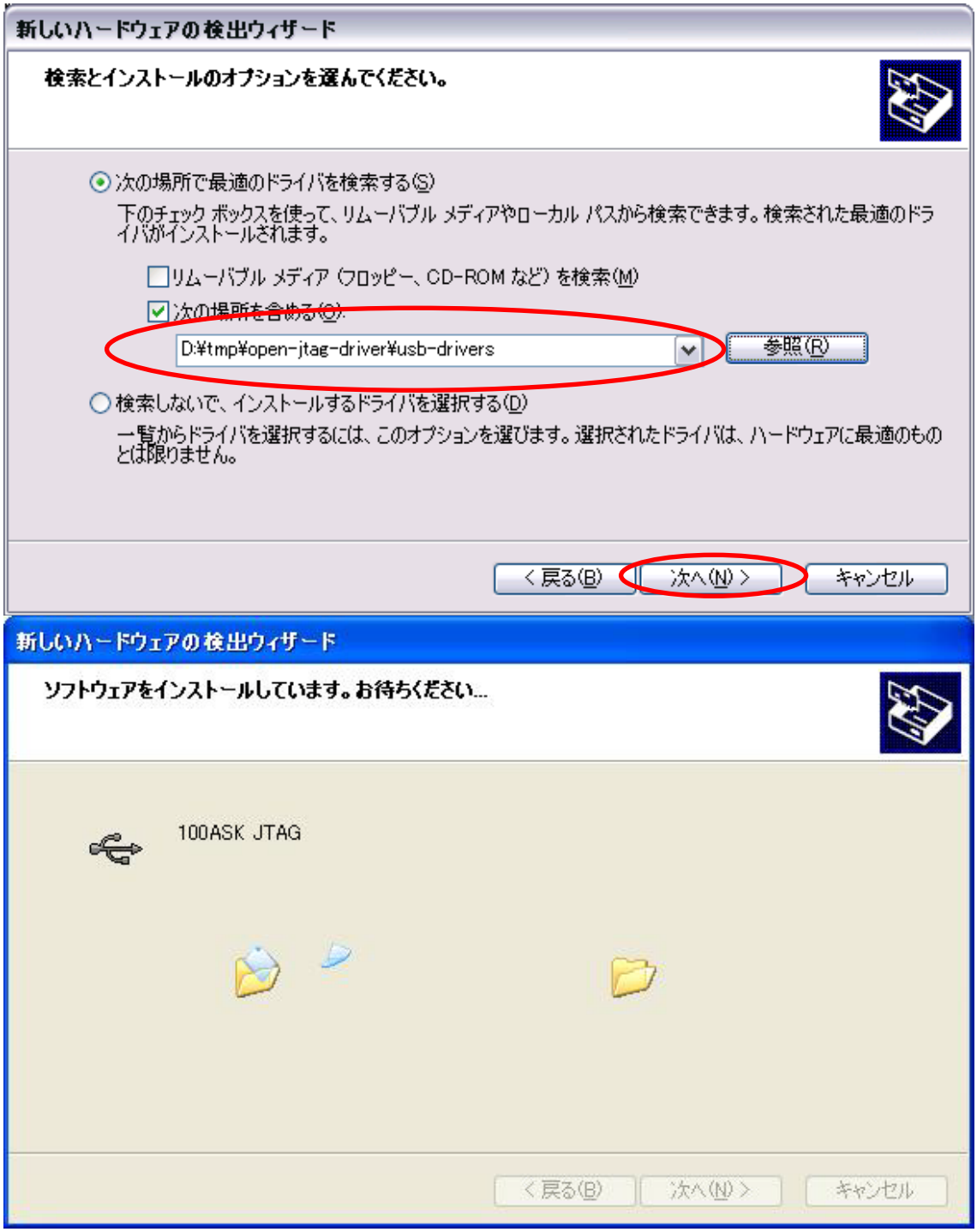
第四章 インストール手順

4.1 OpenJTAG のドライバをインストールする

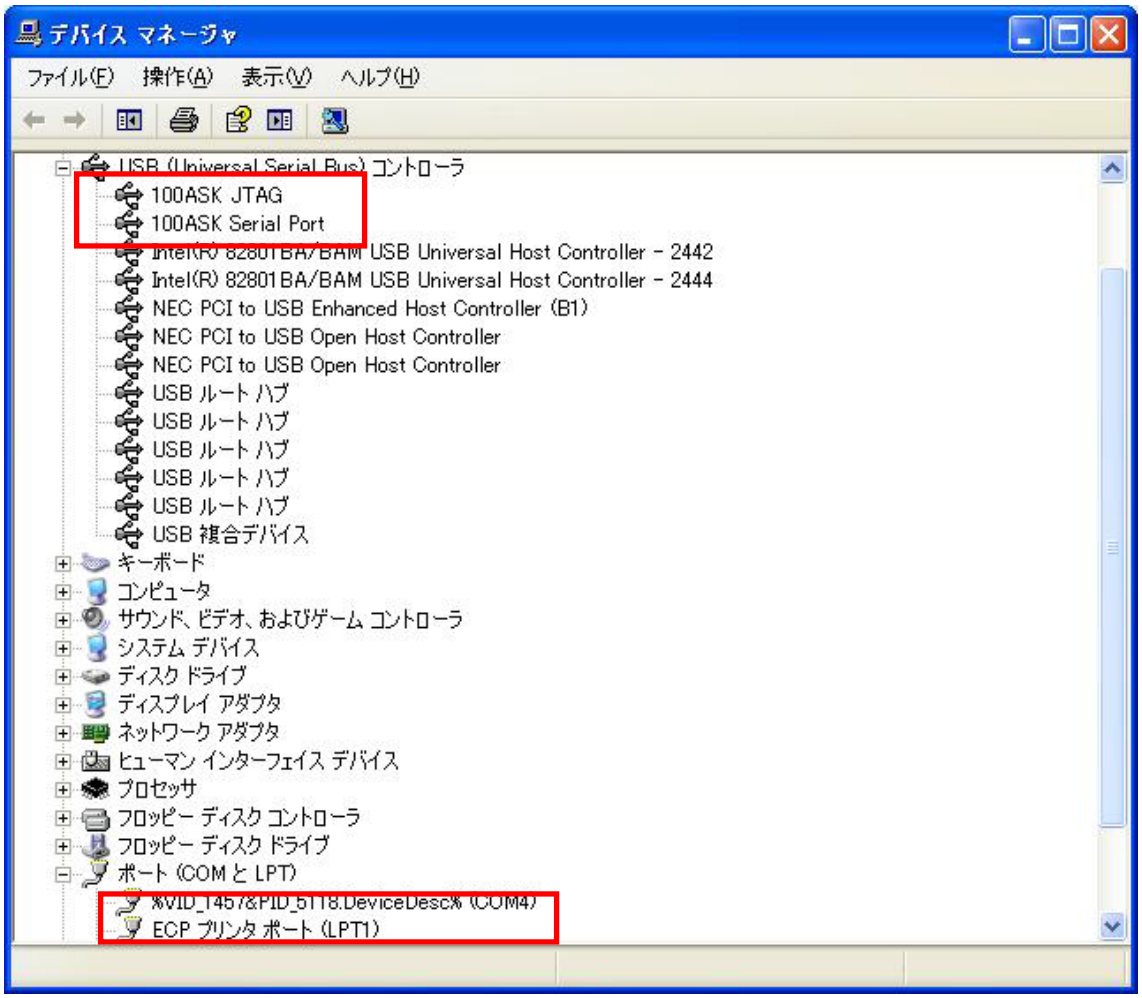
OpenJTAG をパソコンの USB ポートに挿入して、下の通りにドライバをインストールしてください。







USB ドライバのインストールは3回があります。インストール完了すると、デバイスマネージャで三つのデバイスが見えます。



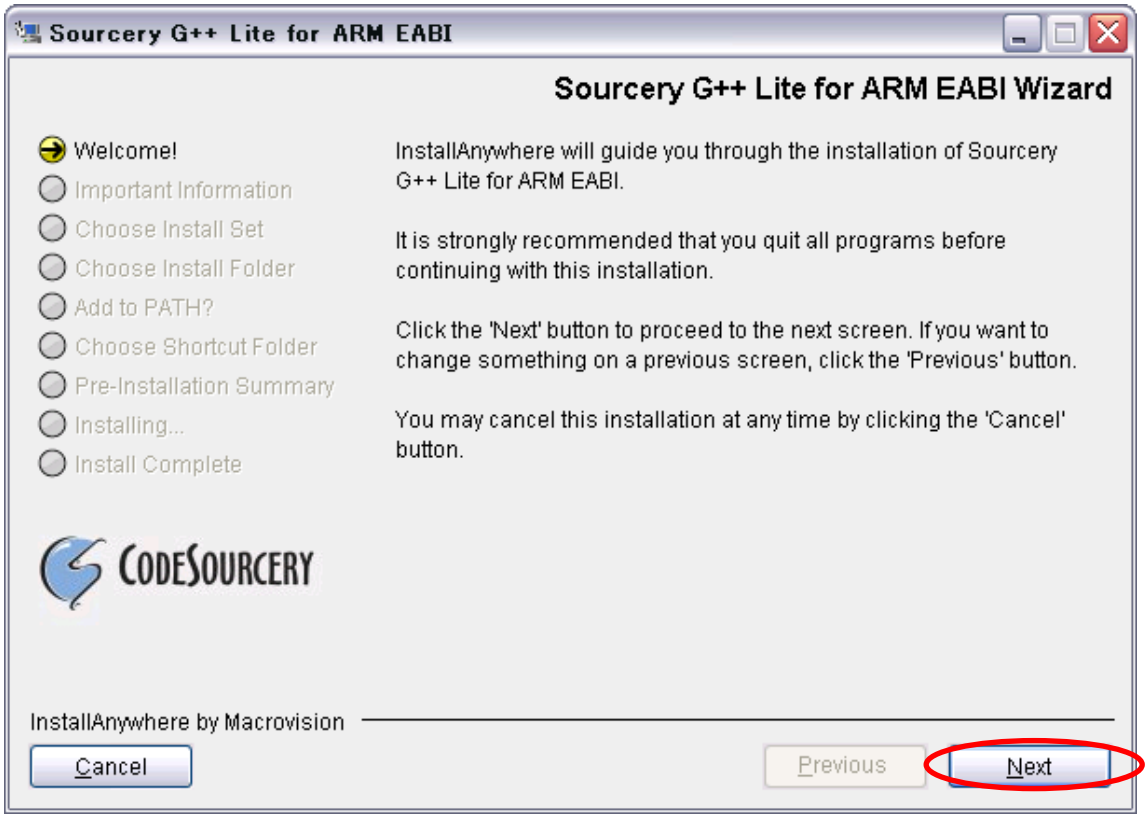
※ OpenJTAG は USB シリアルポートとして使えます。

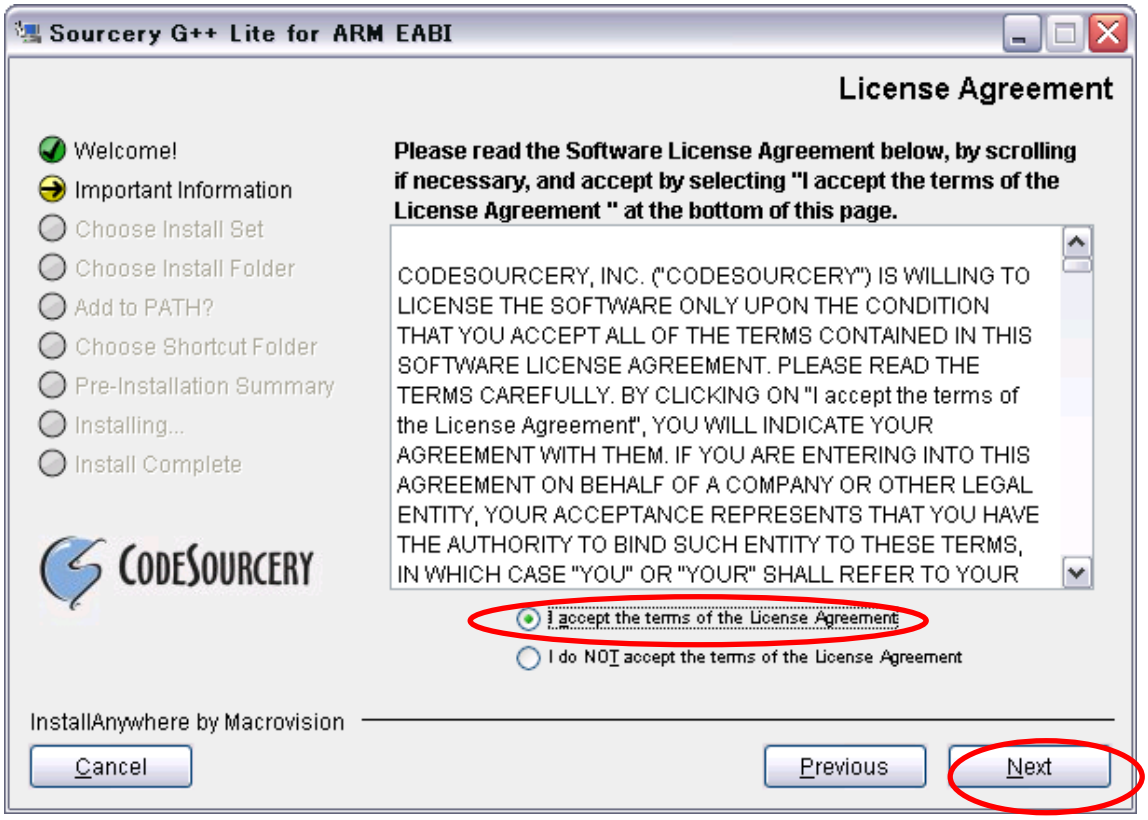
4.2 ソフトウェアをインストールする

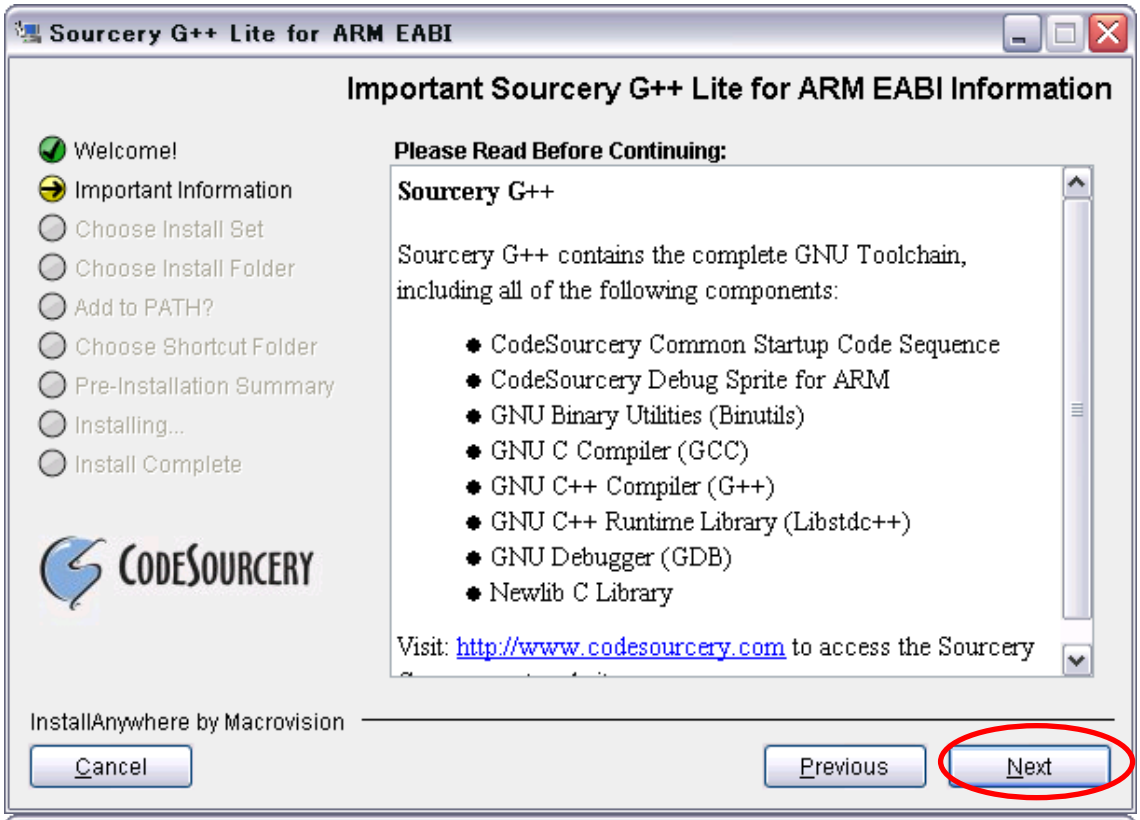
順番で以下のソフトウェアをインストールしてください
01.arm-2009q1-161-arm-none-eabi.exe(EABI を持ってコンパイラ)
02.openocd-0.1.0.msi(jtag デバッガ用ソフト)
03.yagarto-bu-2.19.1_gcc-4.3.3-c-c++_nl-1.17.0_gi-6.8.50_20090311.exe(gcc)
04.yagarto-tools-20070303-setup.exe(各種ユーティリティ)
05.Cygwin
06.jre-6u7-windows-i586-p.exe(Java)
07.eclipse-cpp-galileo-win32.zip(Eclipse)
メモ:①もしバージョン 1.4.2 以上の JRE を既にパソコンにインストールされたら、
06 番の JRE のインストールが不要です。
②05 番の eclipse を解凍して OK、インストール不要

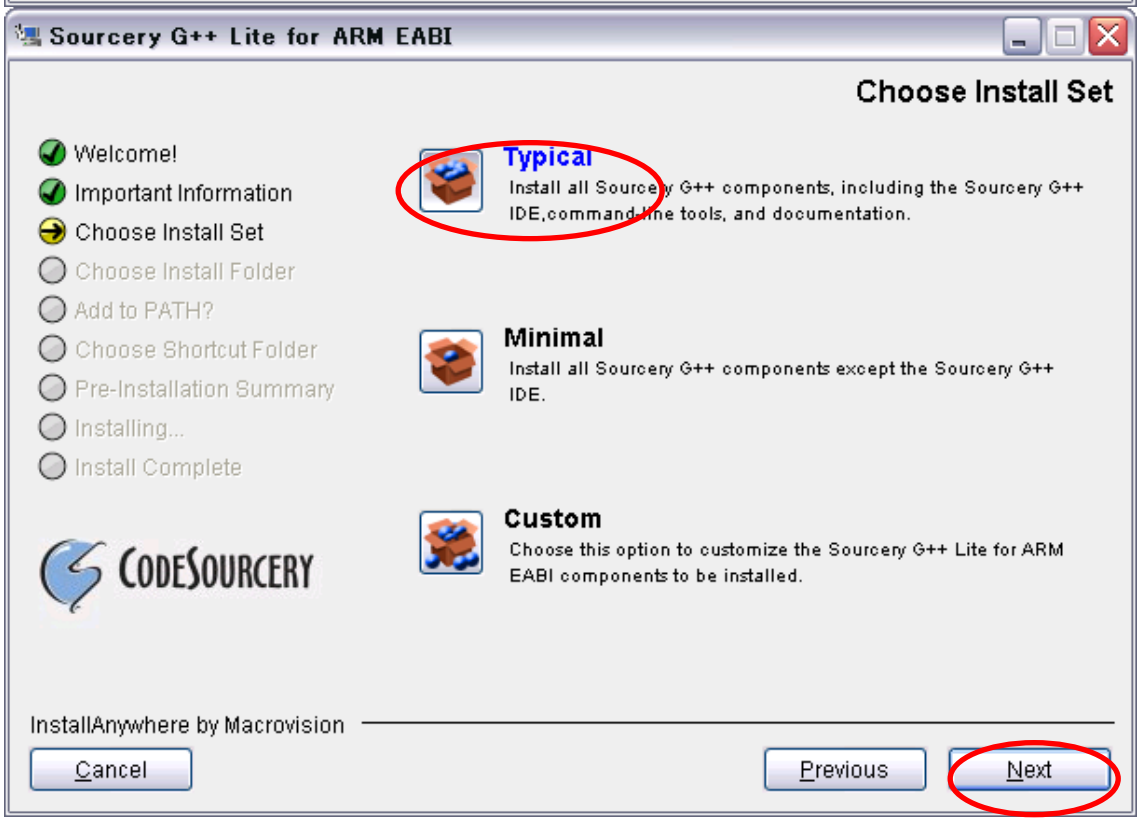
ダウンロードしたインストールファイルは「D:¥tmp¥openJTAGIDE」に格納されます、本書に全てソフトウエアは「D:¥embeded」にインストールされます。

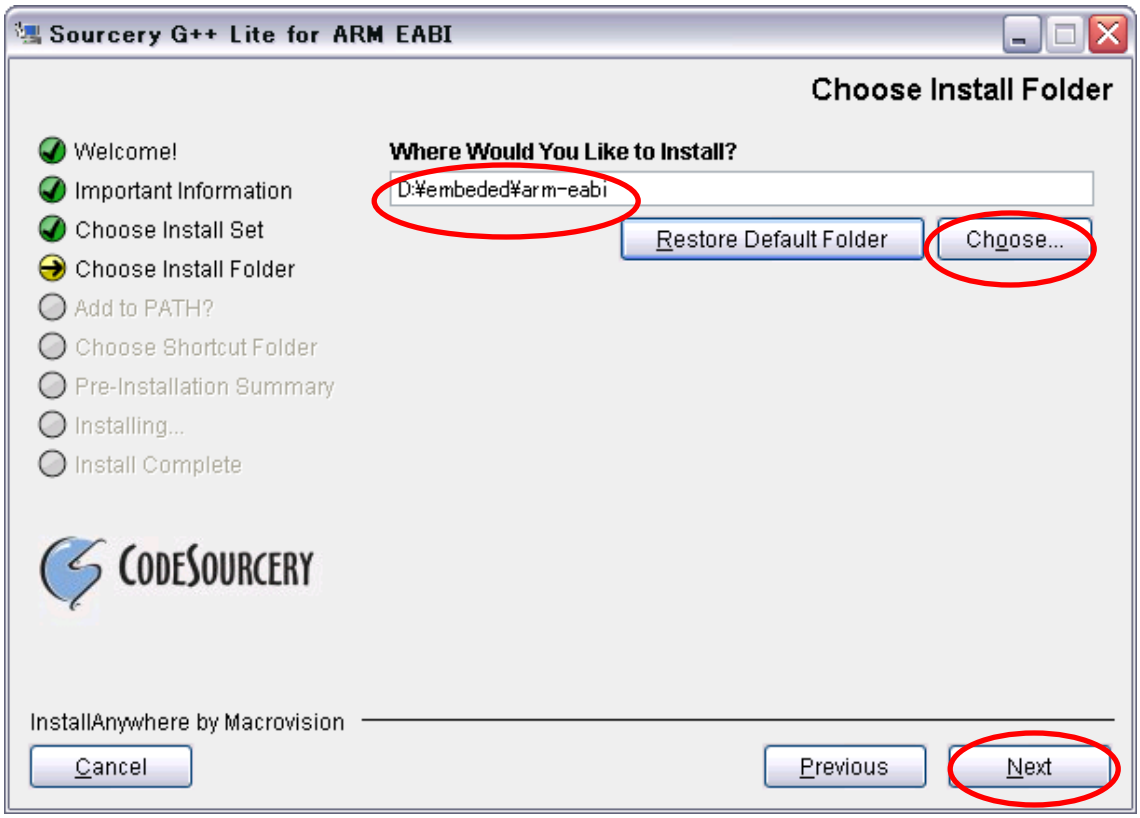
4.2.1 「arm-none-eabi.exe」をインストール




Sourcery G++ Lite for ARM EABI
License Agreement
Welcome!
Important Information
Choose Install Set
Choose Install Folder
Add to PATH?
Choose Shortcut Folder
Pre-Installation Summary
Installing...
Install Complete
CodeSourcery
Please read the Software License Agreement below, by scrolling if necessary, and accept by selecting "I accept the terms of the License Agreement " at the bottom of this page.
CODESOURCERY, INC. ("CODESOURCERY") IS WILLING TO LICENSE THE SOFTWARE ONLY UPON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS SOFTWARE LICENSE AGREEMENT. PLEASE READ THE TERMS CAREFULLY. BY CLICKING ON "I accept the terms of the License Agreement", YOU WILL INDICATE YOUR AGREEMENT WITH THEM. IF YOU ARE ENTERING INTO THIS AGREEMENT ON BEHALF OF A COMPANY OR OTHER LEGAL ENTITY, YOUR ACCEPTANCE REPRESENTS THAT YOU HAVE THE AUTHORITY TO BIND SUCH ENTITY TO THESE TERMS, IN WHICH CASE "YOU" OR "YOUR" SHALL REFER TO YOUR
I accept the terms of the License Agreement
I do NOT accept the terms of the License Agreement
InstallAnywhere by Macrovision
Cancel
Previous
Next

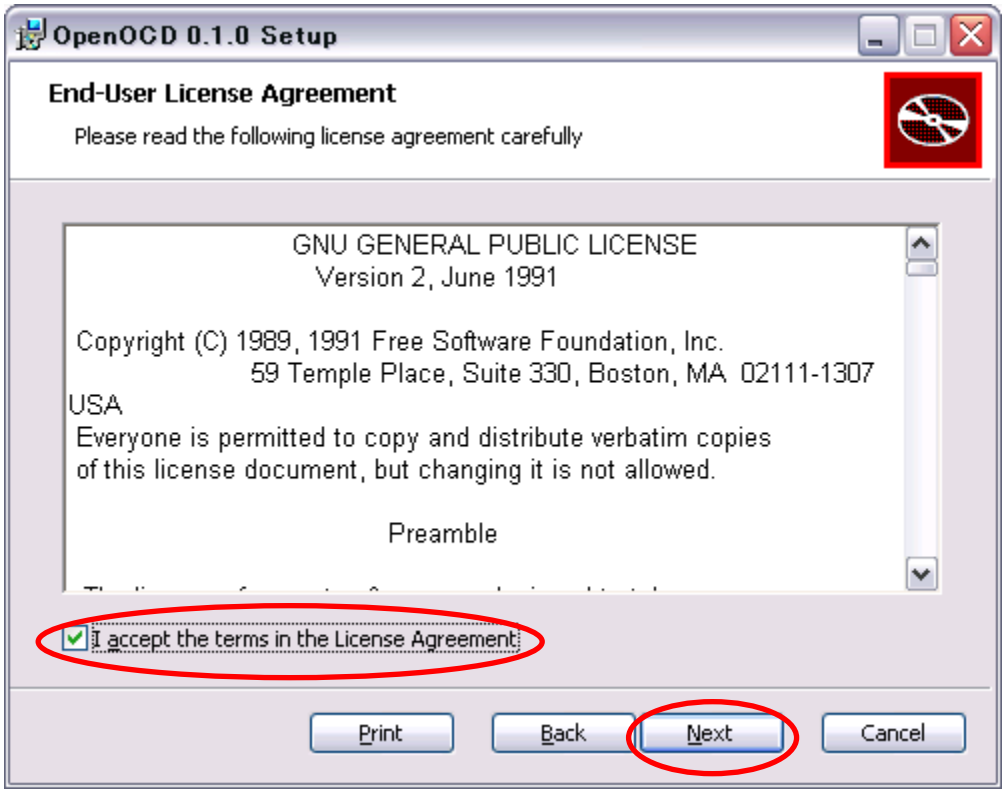

Sourcery G++ Lite for ARM EABI
Important Sourcery G++ Lite for ARM EABI Information
Welcome!
Important Information
Choose Install Set
Choose Install Folder
Add to PATH?
Choose Shortcut Folder
Pre-Installation Summary
Installing...
Install Complete
CodeSourcery
Please Read Before Continuing:
Sourcery G++
Sourcery G++ contains the complete GNU Toolchain, including all of the following components:
CodeSourcery Common Startup Code Sequence
CodeSourcery Debug Sprite for ARM
GNU Binary Utilities (Binutils)
GNU C Compiler (GCC)
GNU C++ Compiler (G++)
GNU C++ Runtime Library (Libstdc++)
GNU Debugger (GDB)
Newlib C Library
Visit: http://www.codesourcery.com to access the Sourcery
InstallAnywhere by Macrovision
Cancel
Previous
Next

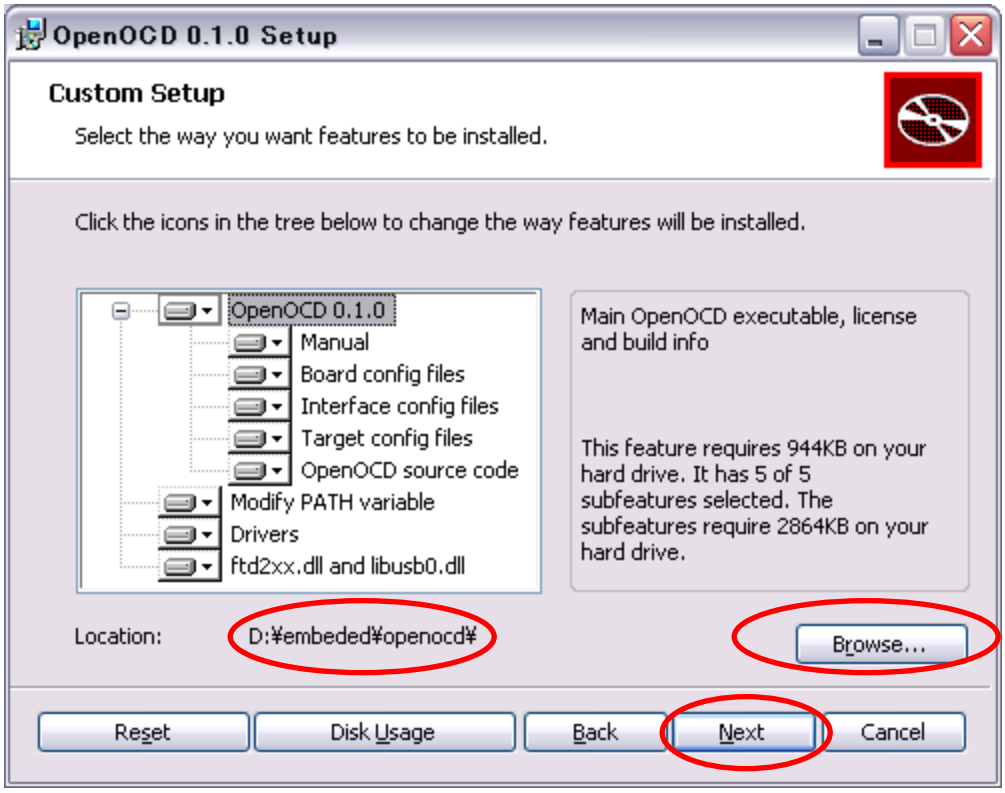

Sourcery G++ Lite for ARM EABI
Choose Install Set
Welcome!
Important Information
Choose Install Set
Choose Install Folder
Add to PATH?
Choose Shortcut Folder
Pre-Installation Summary
Installing...
Install Complete
CodeSourcery
Typical
Install all Sourcery G++ components, including the Sourcery G++ IDE,command-line tools, and documentation.
Minimal
Install all Sourcery G++ components except the Sourcery G++ IDE.
Custom
Choose this option to customize the Sourcery G++ Lite for ARM EABI components to be installed.
InstallAnywhere by Macrovision
Cancel
Previous
Next

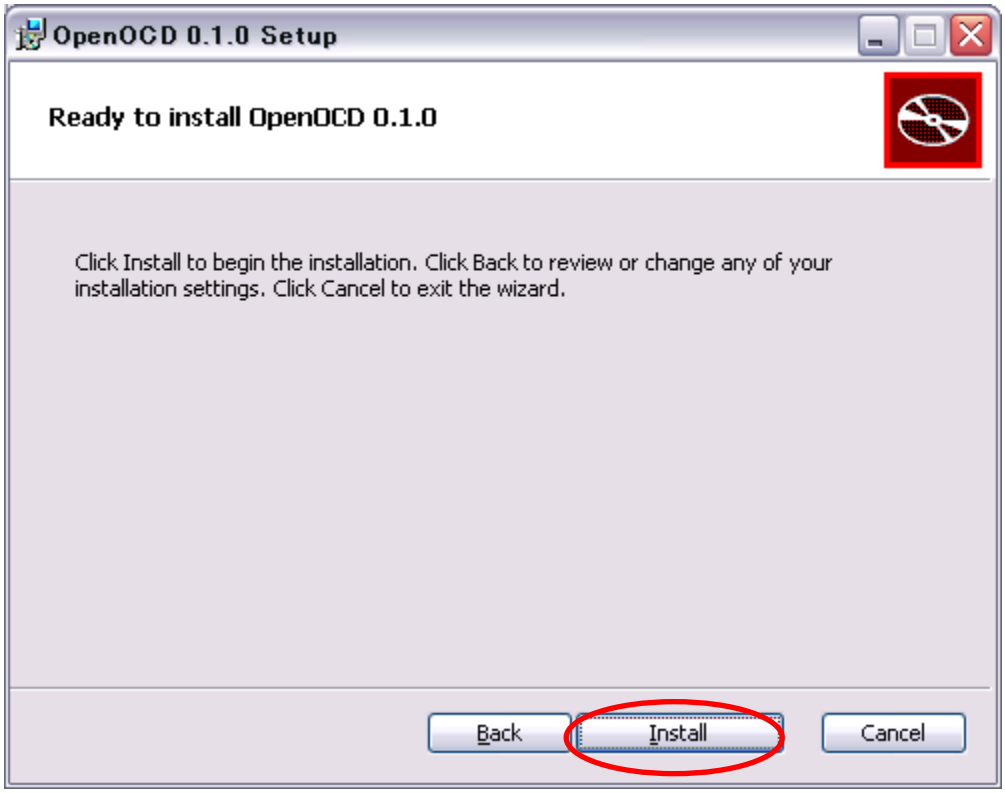


「Next」をクリックすると、インストールが始まります、インストール完了までに何分がかかります、完了までにお待ちください。

4.2.2 「openocd」をインストール

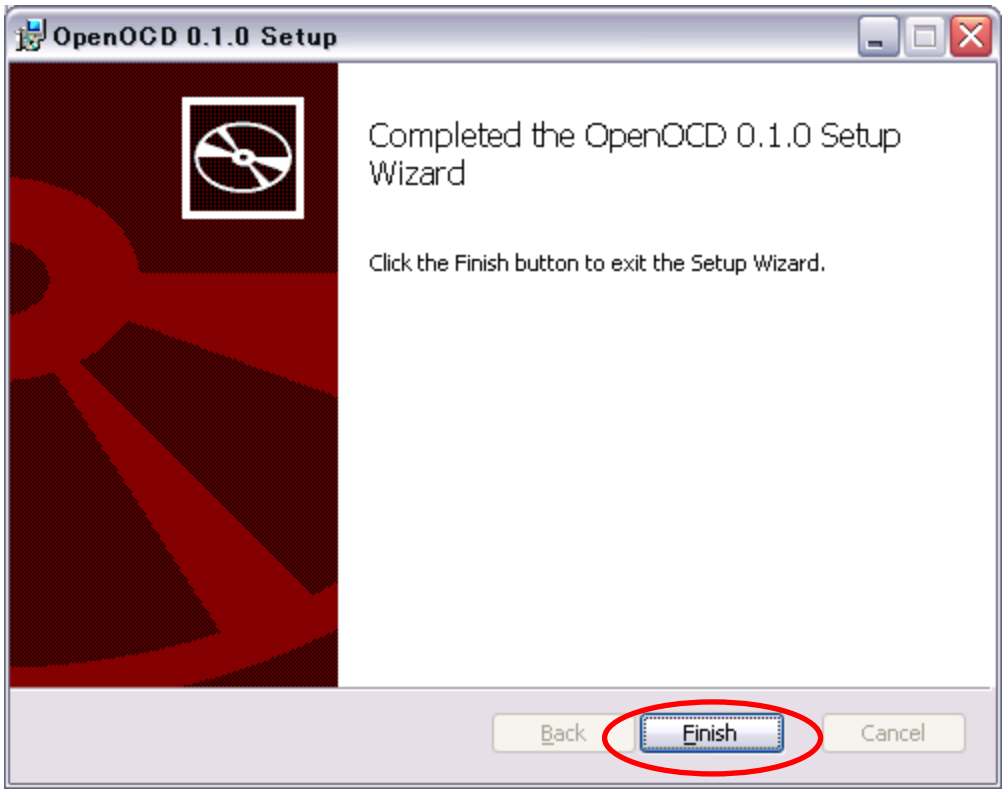

OpenOCD 0.1.0 Setup
End-User License Agreement
Please read the following license agreement carefully
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307
USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
I accept the terms in the License Agreement
Print
Back
Next
Cancel


OpenOCD 0.1.0 Setup
Custom Setup
Select the way you want features to be installed.
Click the icons in the tree below to change the way features will be installed.
OpenOCD 0.1.0
Manual
Board config files
Interface config files
Target config files
OpenOCD source code
Modify PATH variable
Drivers
ftd2xx.dll and libusb0.dll
Main OpenOCD executable, license and build info
This feature requires 944KB on your hard drive. It has 5 of 5 subfeatures selected. The subfeatures require 2864KB on your hard drive.
Location:
D:¥embeded¥openocd¥
Browse...
Reset
Disk Usage
Back
Next
Cancel

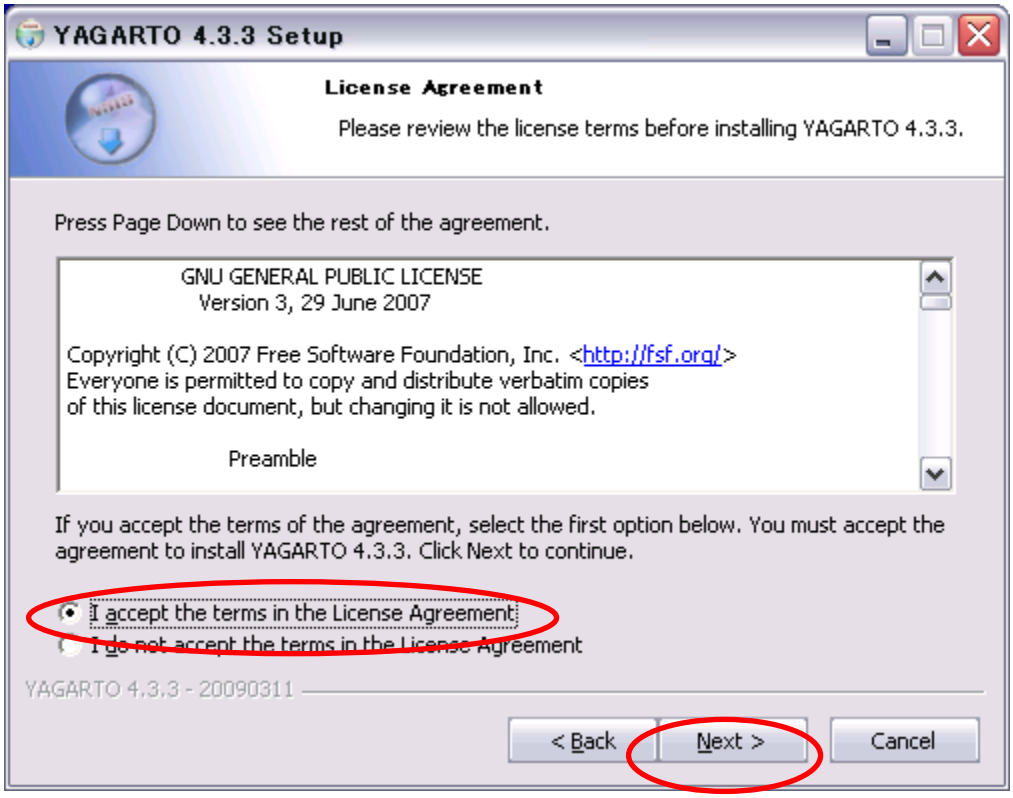


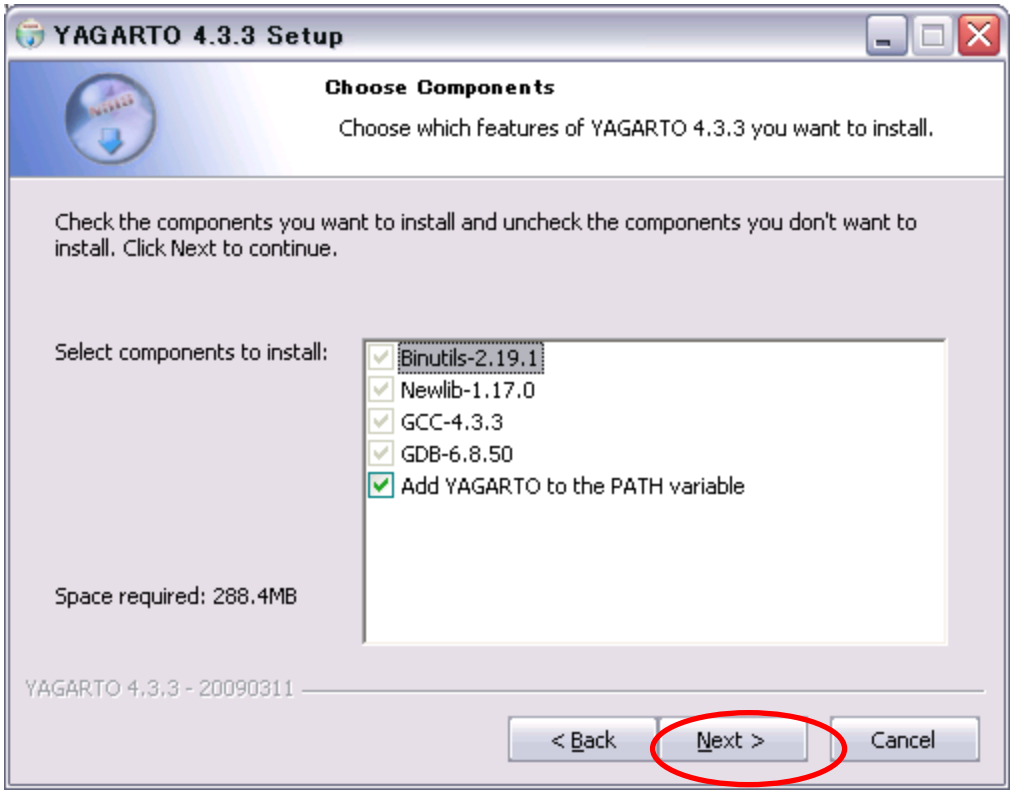
「Install」をクリックすると、インストールが始まります、インストール完了までに何分がかかります、完了までにお待ちください。

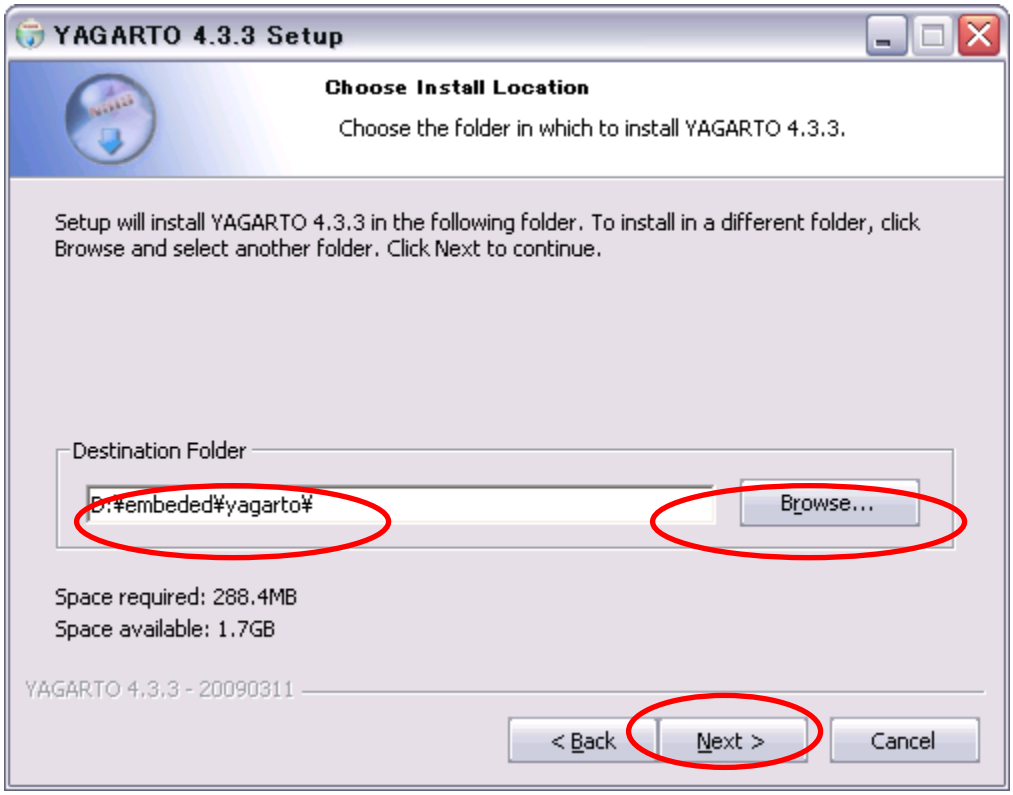
「Finish」をクリックし、OpenOCD のインストールが完了です。


OpenOCD 0.1.0 Setup
Completed the OpenOCD 0.1.0 Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel

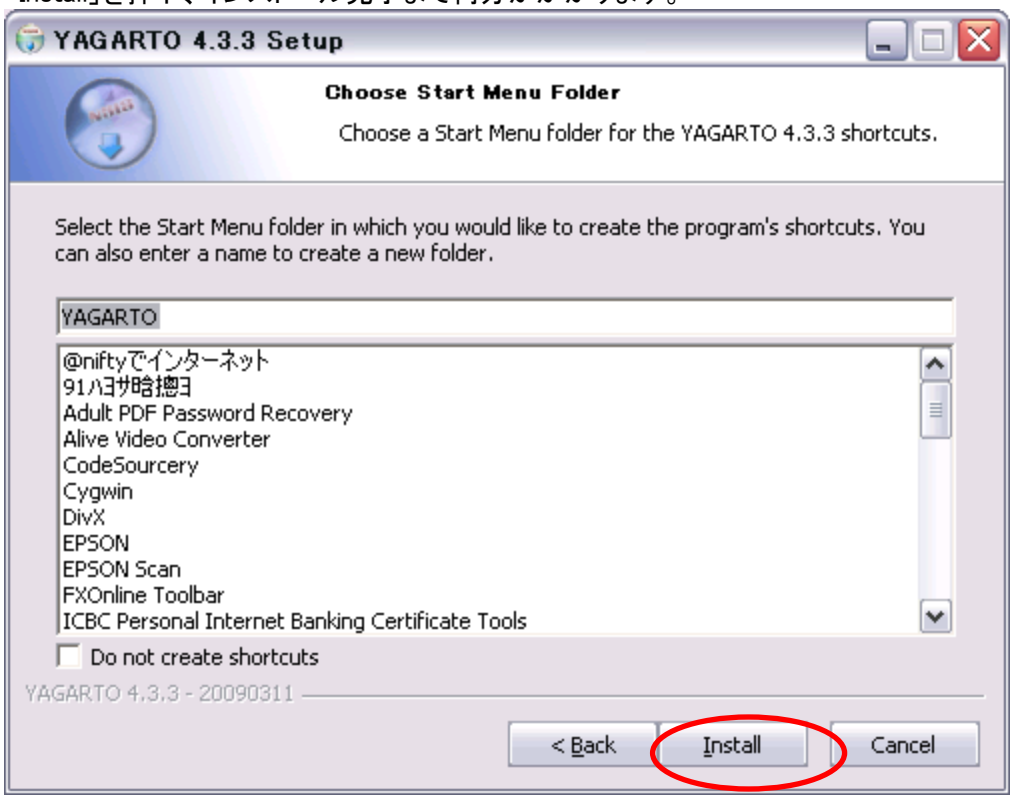
4.2.3 「yagarto」をインストール


YAGARTO 4.3.3 Setup
License Agreement
Please review the license terms before installing YAGARTO 4.3.3.
Press Page Down to see the rest of the agreement.
GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007
Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
If you accept the terms of the agreement, select the first option below. You must accept the agreement to install YAGARTO 4.3.3. Click Next to continue.
I accept the terms in the License Agreement
I do not accept the terms in the License Agreement
YAGARTO 4.3.3 - 20090311
< Back
Next >
Cancel

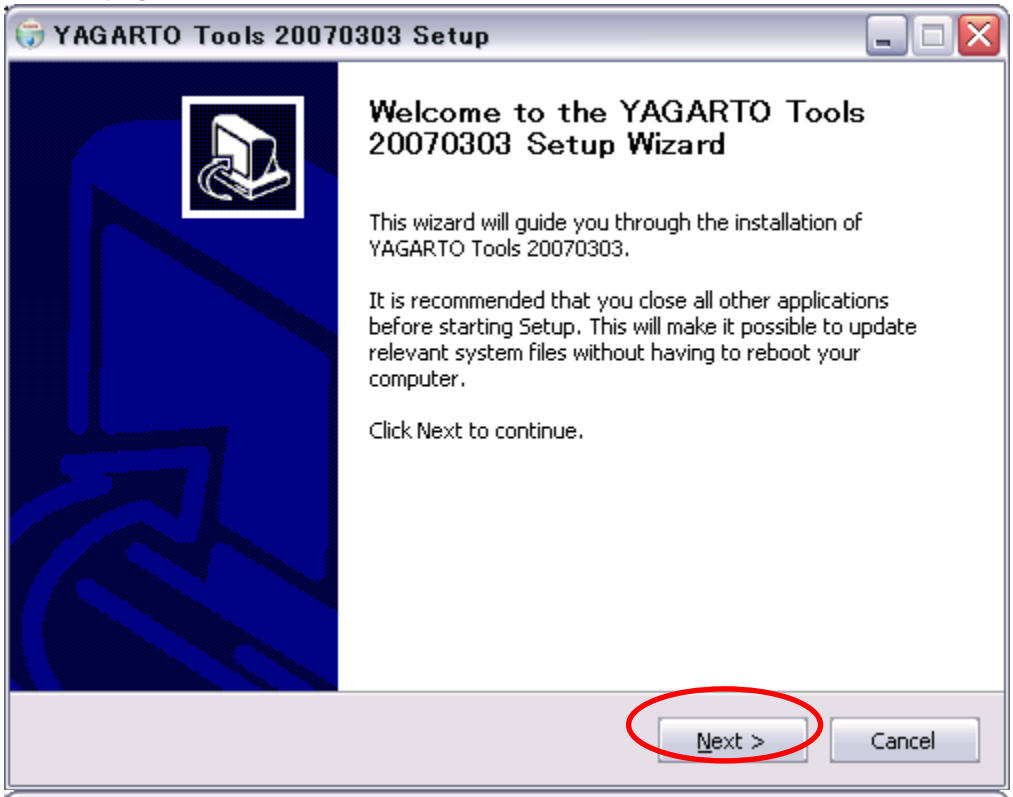

YAGARTO 4.3.3 Setup
Choose Components
Choose which features of YAGARTO 4.3.3 you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Binutils-2.19.1
Newlib-1.17.0
GCC-4.3.3
GDB-6.8.50
Add YAGARTO to the PATH variable
Space required: 288.4MB
YAGARTO 4.3.3 - 20090311
< Back
Next >
Cancel

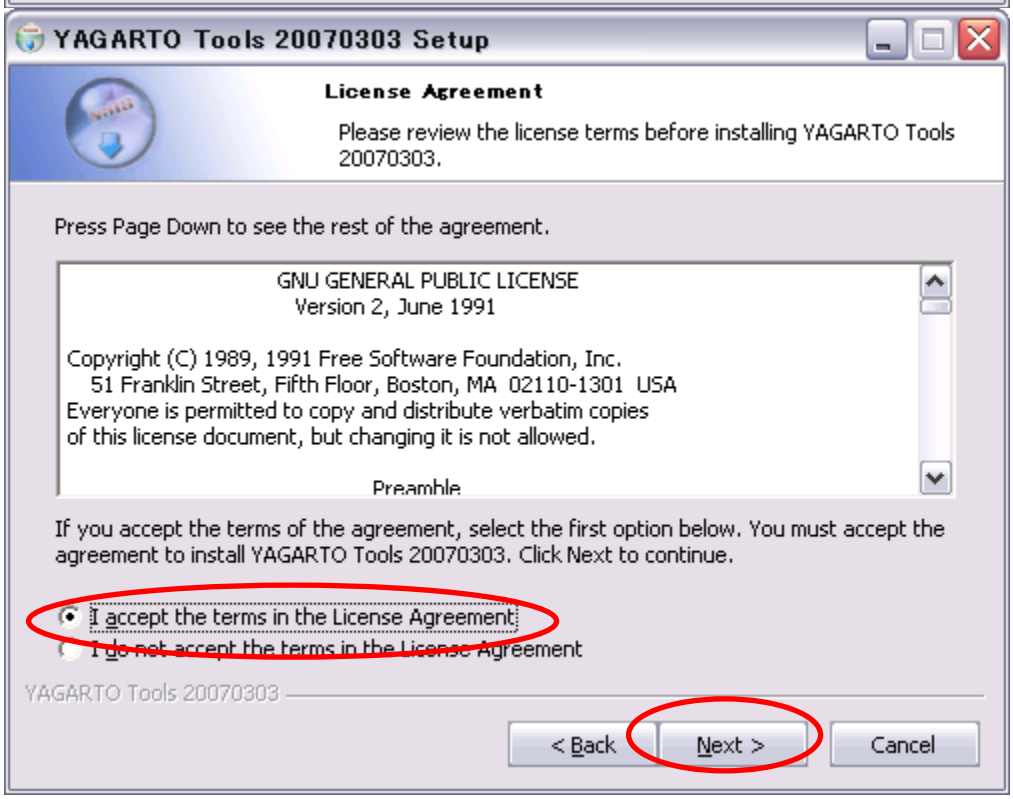

YAGARTO 4.3.3 Setup
Choose Install Location
Choose the folder in which to install YAGARTO 4.3.3.
Setup will install YAGARTO 4.3.3 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
B:¥embeded¥yagarto¥
Browse...
Space required: 288.4MB
Space available: 1.7GB
YAGARTO 4.3.3 - 20090311
< Back
Next >
Cancel

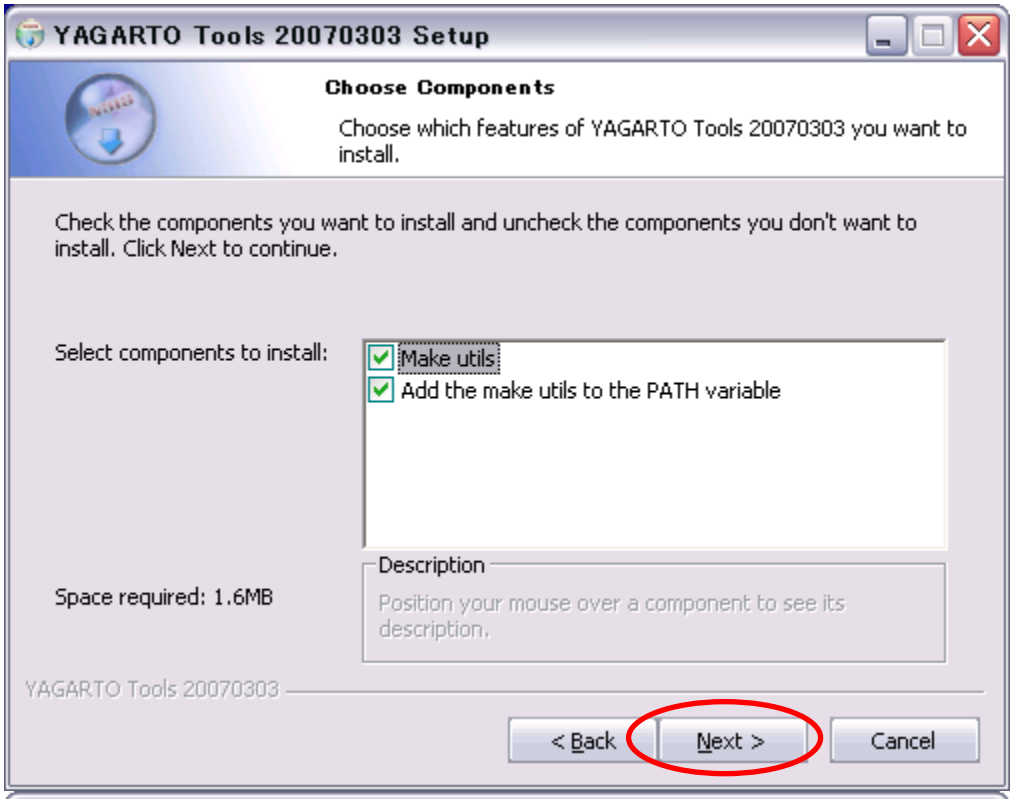
「Install」を押下、インストール完了まで何分がかかります。

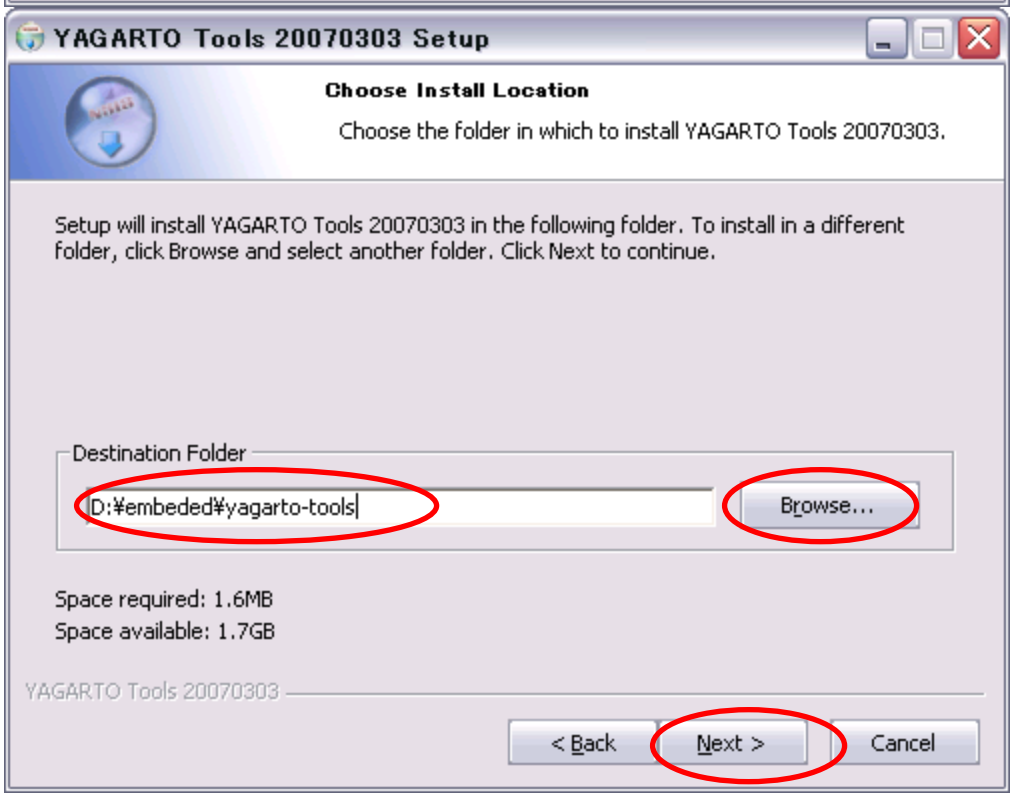


4.2.4 「yagarto-tools」をインストール

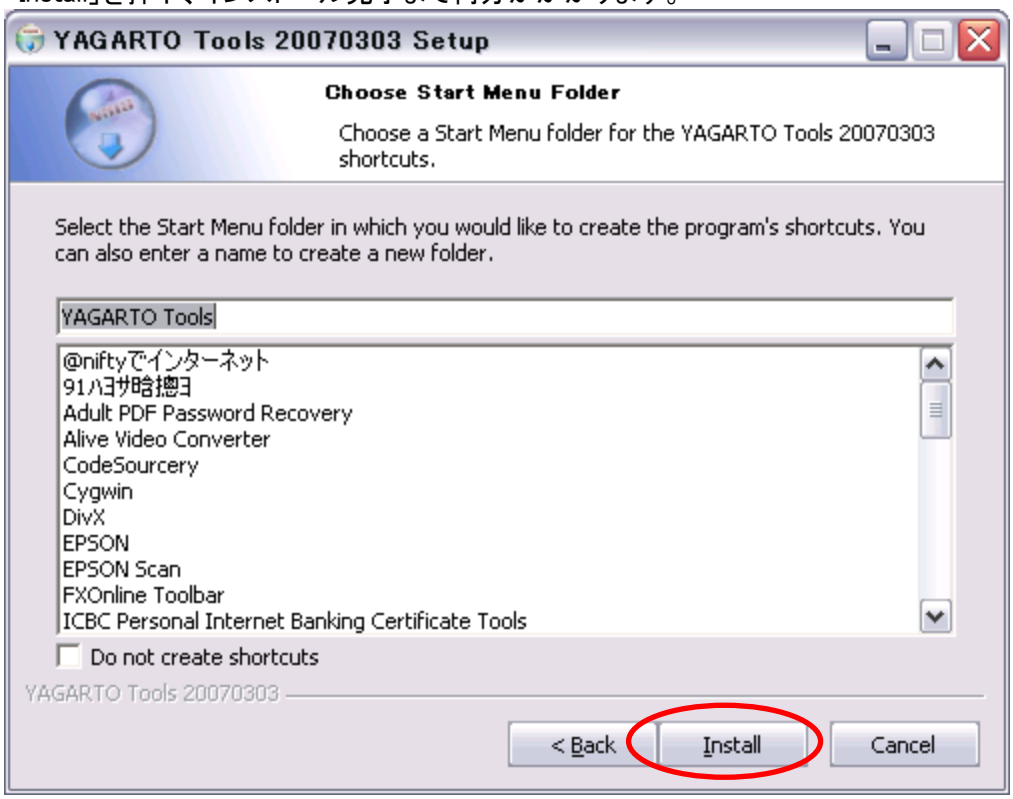





YAGARTO Tools 20070303 Setup
Choose Components
Choose which features of YAGARTO Tools 20070303 you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Make utils
Add the make utils to the PATH variable
Description
Position your mouse over a component to see its description.
Space required: 1.6MB
YAGARTO Tools 20070303
< Back
Next >
Cancel


YAGARTO Tools 20070303 Setup
Choose Install Location
Choose the folder in which to install YAGARTO Tools 20070303.
Setup will install YAGARTO Tools 20070303 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
D:¥embeded¥yagarto-tools
Browse...
Space required: 1.6MB
Space available: 1.7GB
YAGARTO Tools 20070303
< Back
Next >
Cancel

「Install」を押下、インストール完了まで何分がかかります。



4.2.5 Cygwin のダウンロード＆インストール

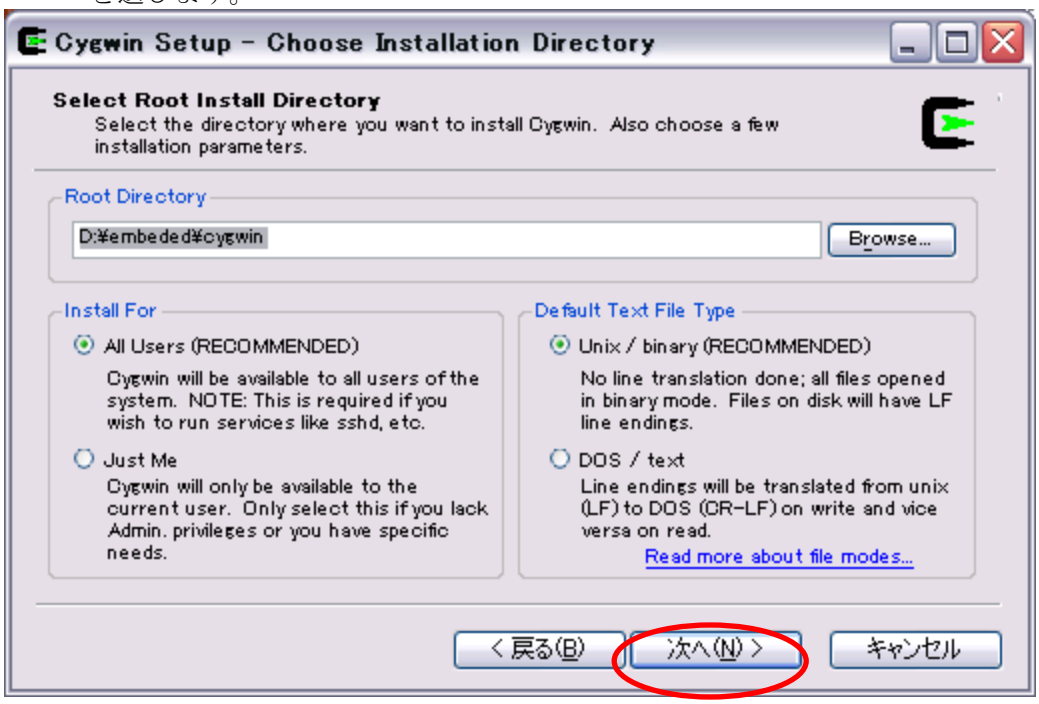
1 インストール

a) setup.exe の DL＆起動

http://cygwin.com/より **setpu.exe** をダウンロードし起動てください。
直接ダウンロードしたファイルを起動しても OK。
以下に注意すべき画面を中心に解説します(より詳しい手順は Cygwin のページを参照したください)。

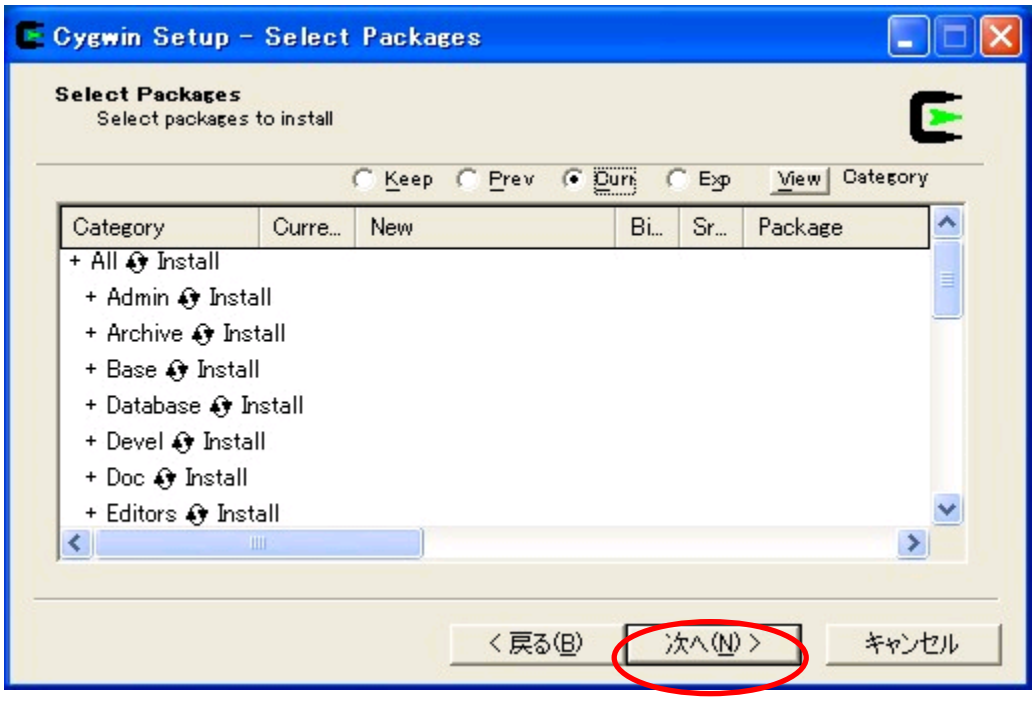
b) テキストファイル形式の選択

以下の「Default Text File Type」に「**Unix**」を選びます(テキストファイルが UNIX 形式(改行が LF)となります)。また Cygwin をインストールするマシンをサーバーとして利用したい場合には、「Install For」で「**All Users**」を選びます。



c）パッケージ選択

パッケージ選択の画面にて Category の「All」を「**Install**」に変更する。この変更によって全パッケージがインストールされることになる



上記の設定が終われば「次へ」をクリックすれば、ダウンロード＆インストールが始まる。

2 インストール後の設定
a) ホームディレクトリの作成

通常、デスクトップにできた Cygwin アイコンをクリックすると、/usr/username がホームディレクトリになるが、Windows の環境変数 HOME が設定されていれば、Cygwin においてもそこがホームディレクトリとなる。この問題を避けるために、コントロールパネルの「システム」 を起動し、「詳細設定」タブの下にある「環境変数」をクリックする。そしてユーザ環境変数に HOME 変数が設定されていれば削除する。
ここでデスクトップの Cygwin アイコンをクリックすればホームディレクトリが自動的に作成される。

次にホームディレクトリの下に各種設定ファイルを編集していく。利用する漢字コードによって設定方法が異なるので、以下では SJIS 環 境を説明する。設定するファイルは、.bash_profile、.bashrc、.inputrc、.vimrc の4つで(全てのファイルはピリ オドから始まるファイルで隠しファイルとなっており、"ls -a" にて確認できる)、以下に示した通りに設定する。既に存在するファイルについては追加し、存在しないファイルについては新規作成する。またファイルの編集 は TeraPad などのエディタを利用すれば便利。注意点として、新規ファイルを保存する際には、「ファイル」→「漢字・改行コードを指定保存」を選び、改行コードに「LF」を選んで保存してください。

b)SJIS 環境に関わる4つファイルの設定

i).bash_profile

```
export TERM=vt100
export LANG=ja_JP.SJIS
export LESSCHARSET=japanese-sjis
```

ii).bashrc

```
alias ls='ls --show-control-chars --color=auto'
alias ct='cygterm &'
function ie {
echo /cygdrive/c/Program¥ Files/Internet¥ Explorer/IEXPLORE "$(cygpath -w $PWD)¥¥$1"
/cygdrive/c/Program¥ Files/Internet¥ Explorer/IEXPLORE "$(cygpath -w $PWD)¥¥$1" &
}
```

上記の ie 関数を設定しておくことにより、XML や XMLtable ファイルをコマンド上から起動可能になる。
例) ie dat.xt #dat.xt を InternetExplore を利用して表示する。

iii).inputrc

```
set convert-meta off
set input-meta on
set output-meta on
```

iv).vimrc

```
set encoding=japan
set fileencodings=sjis
```

4.2.6 「jre」をインストール

一般のソフトウエアと同じようにインストールします。**もしバージョン 1.4.2 以上の JRE を既にパソコンにインストールされたら、インストールが不要です。**

「06.jre-6u7-windows-i586-p.exe」あるいはダウンロードした JRE インストールファイルをクリックし、「D:¥Embeded¥JRE」にインストールする

4.2.7 「Eclipse」をインストール

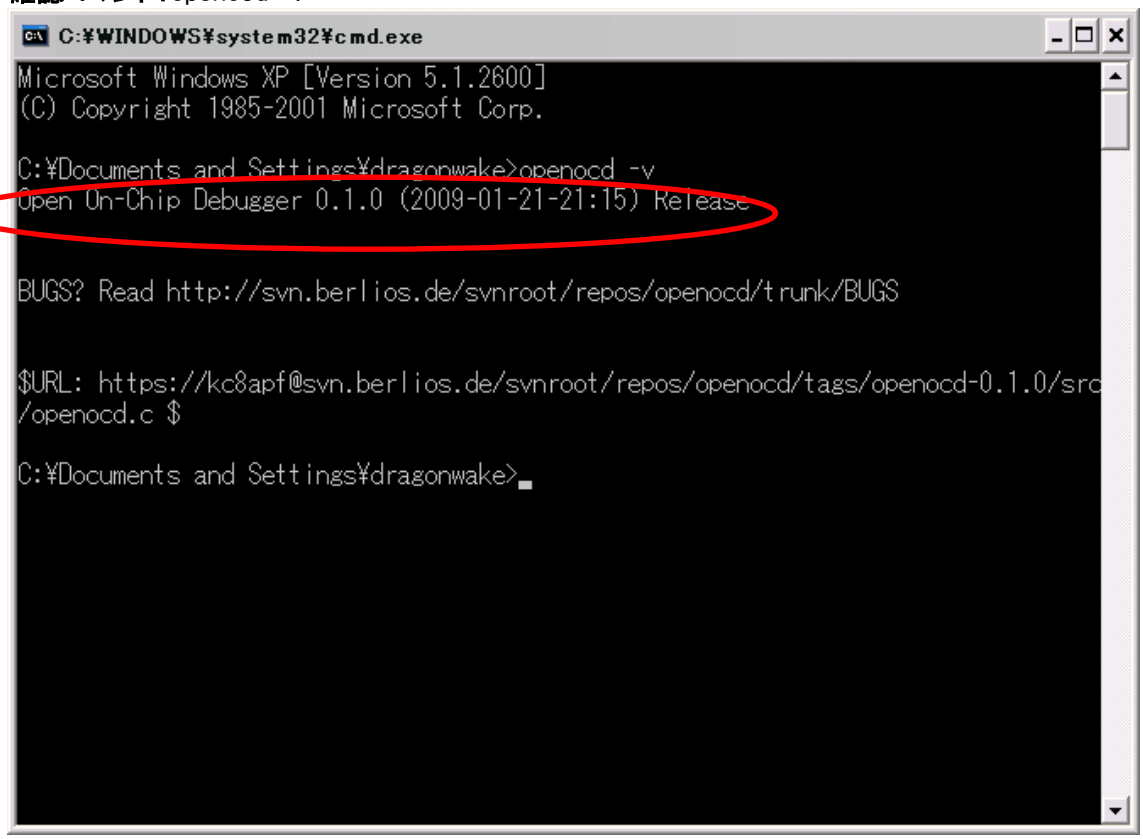
「07.eclipse-cpp-galileo-win32.zip」あるいはダウンロードしたファイルを「D:¥Embeded」解凍

4.3 ソフトウエアインストール後の動作確認

確認方法:「スタート」→「ファイル名を指定して実行」→「cmd」を入力

4.3.1 OpenOCD の確認

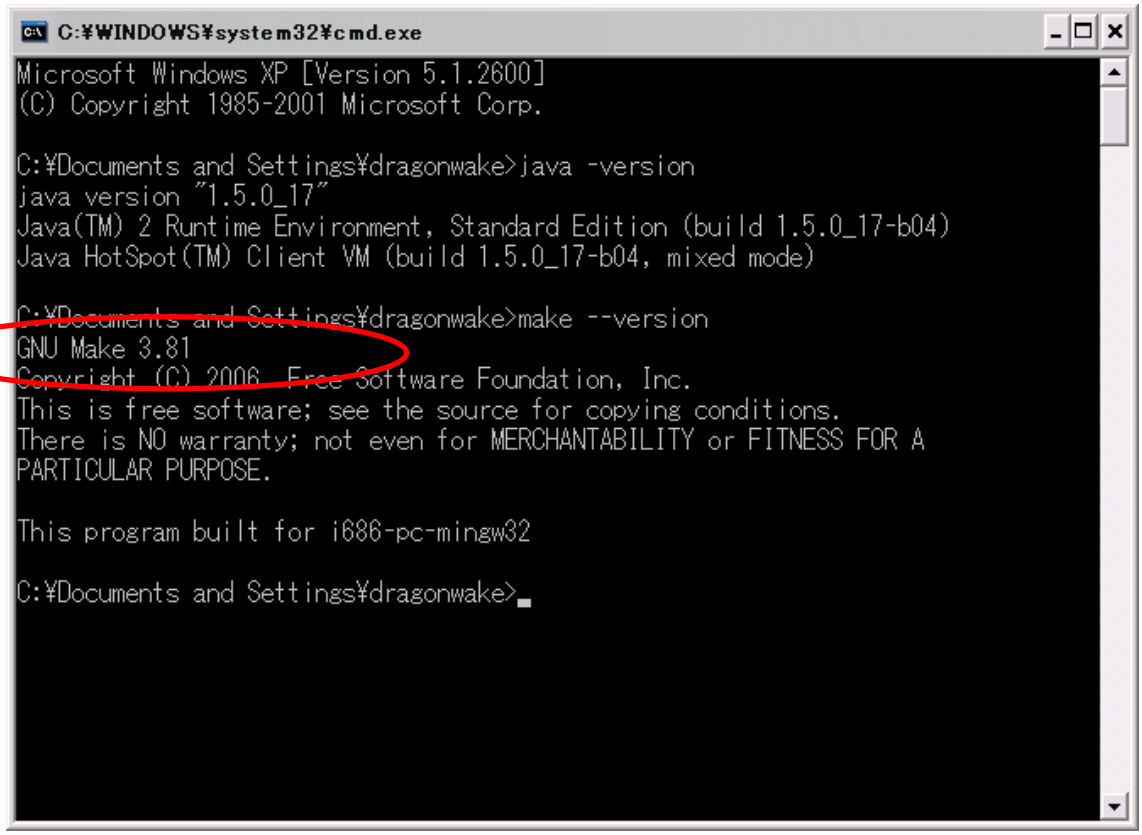
確認コマンド:openocd -v



4.3.2 コンパイラ確認

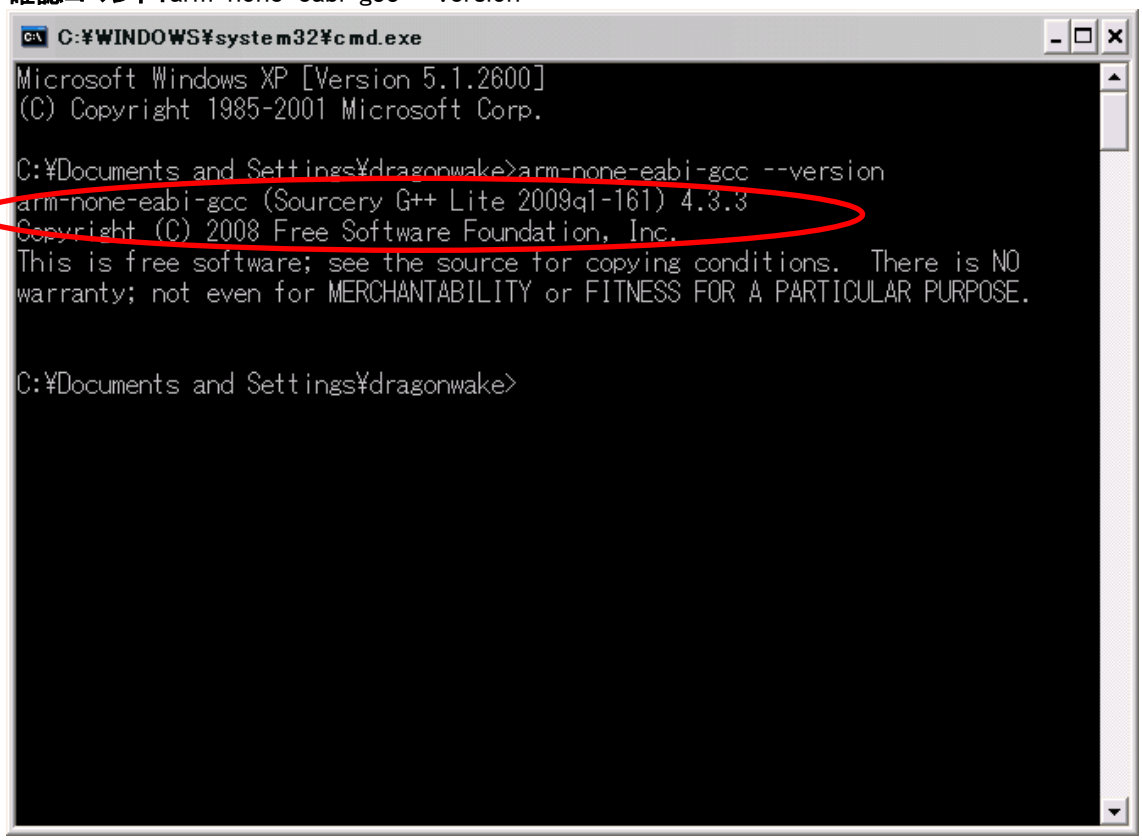
1：GCC コンパイラ「make」確認

確認コマンド：make --version

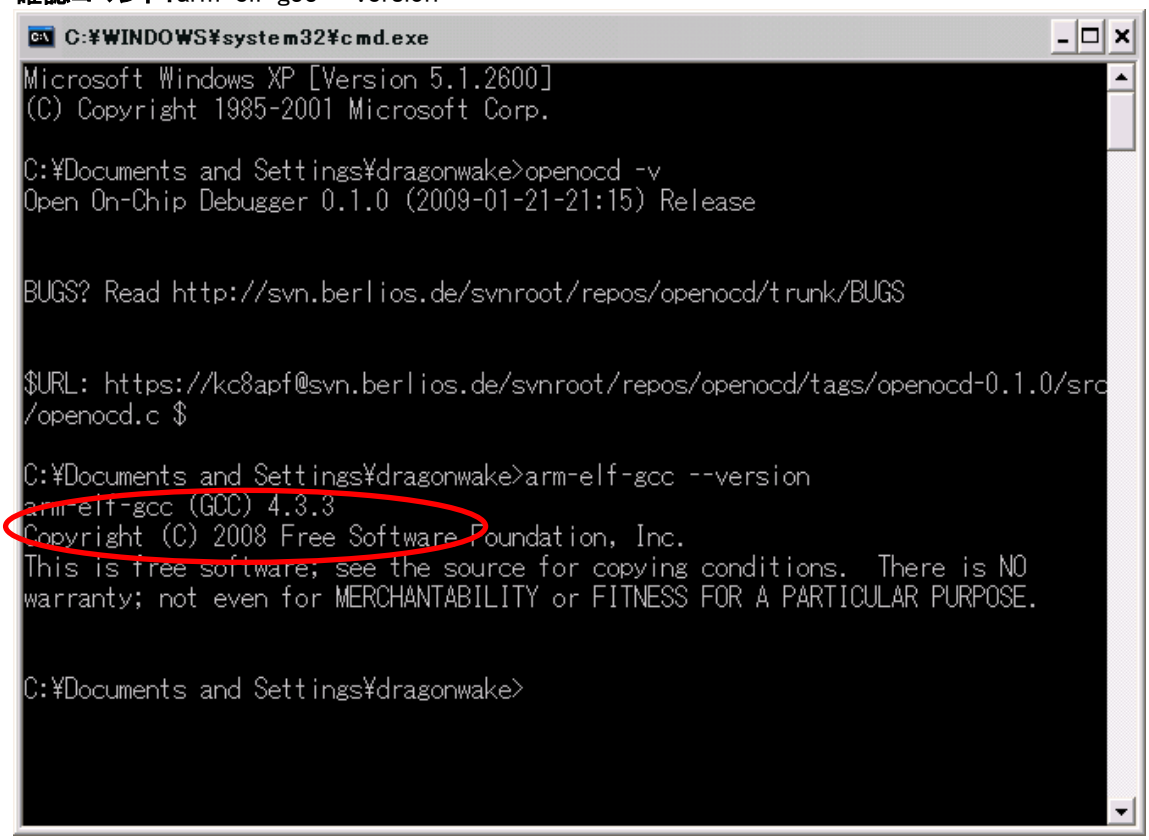


2：ARM EABI「arm-none-eabi-gcc」確認

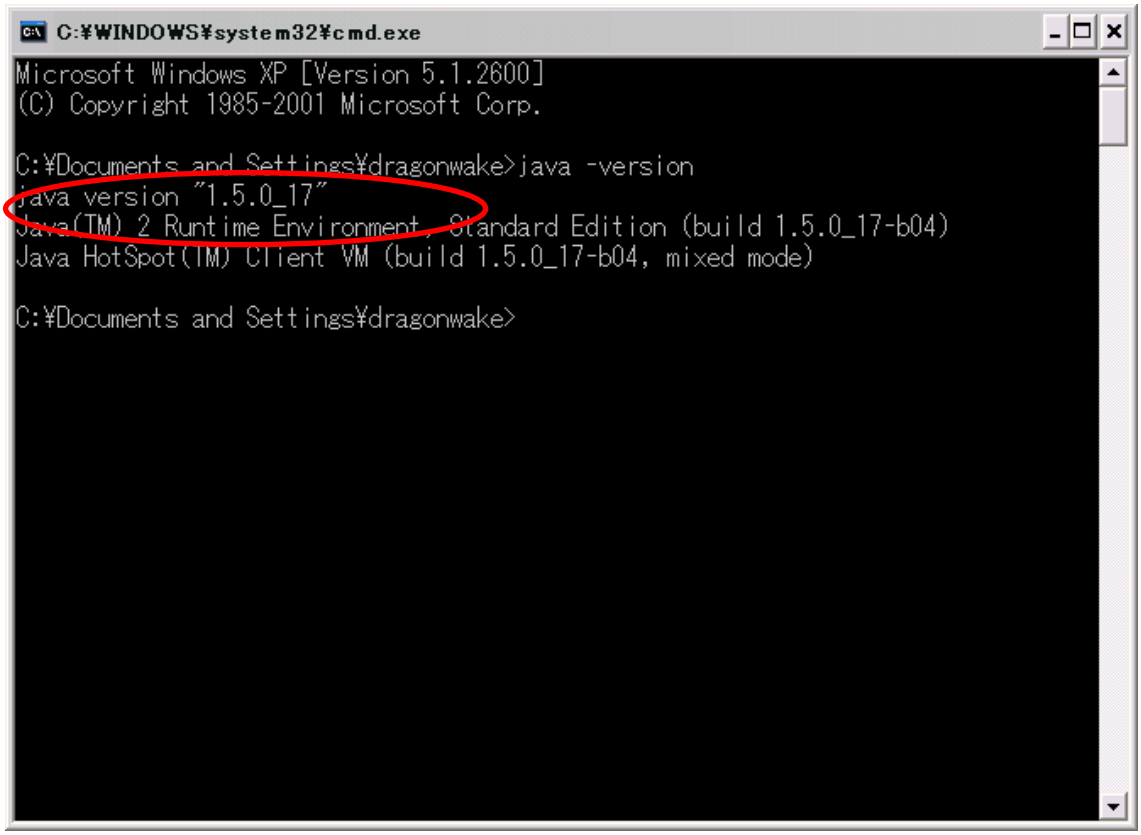
確認コマンド：arm-none-eabi-gcc --version



3：ARM コンパイラ「arm-elf-gcc」確認
確認コマンド：arm-elf-gcc --version

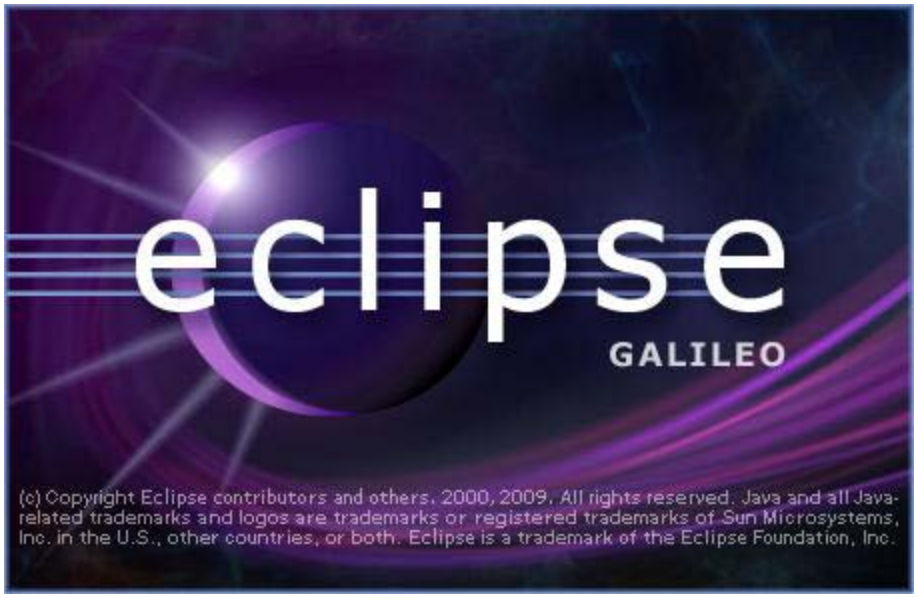


4.JRE バージョン確認:
確認コマンド:
java -version

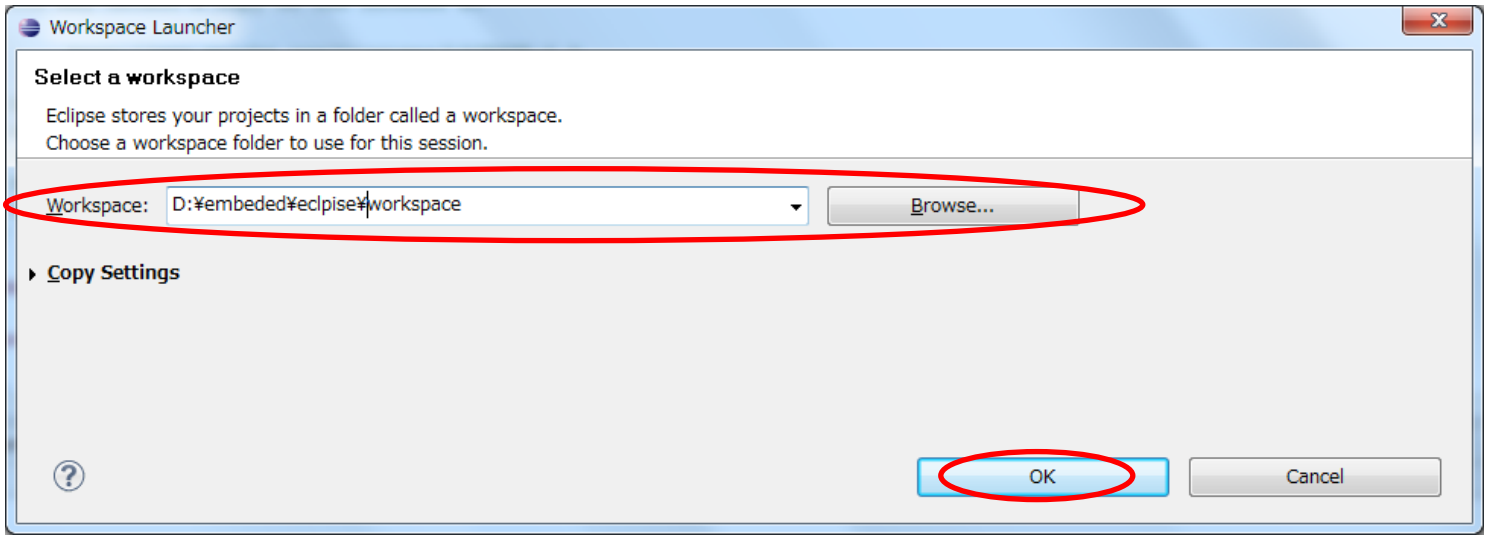


第五章 Eclipse の設定

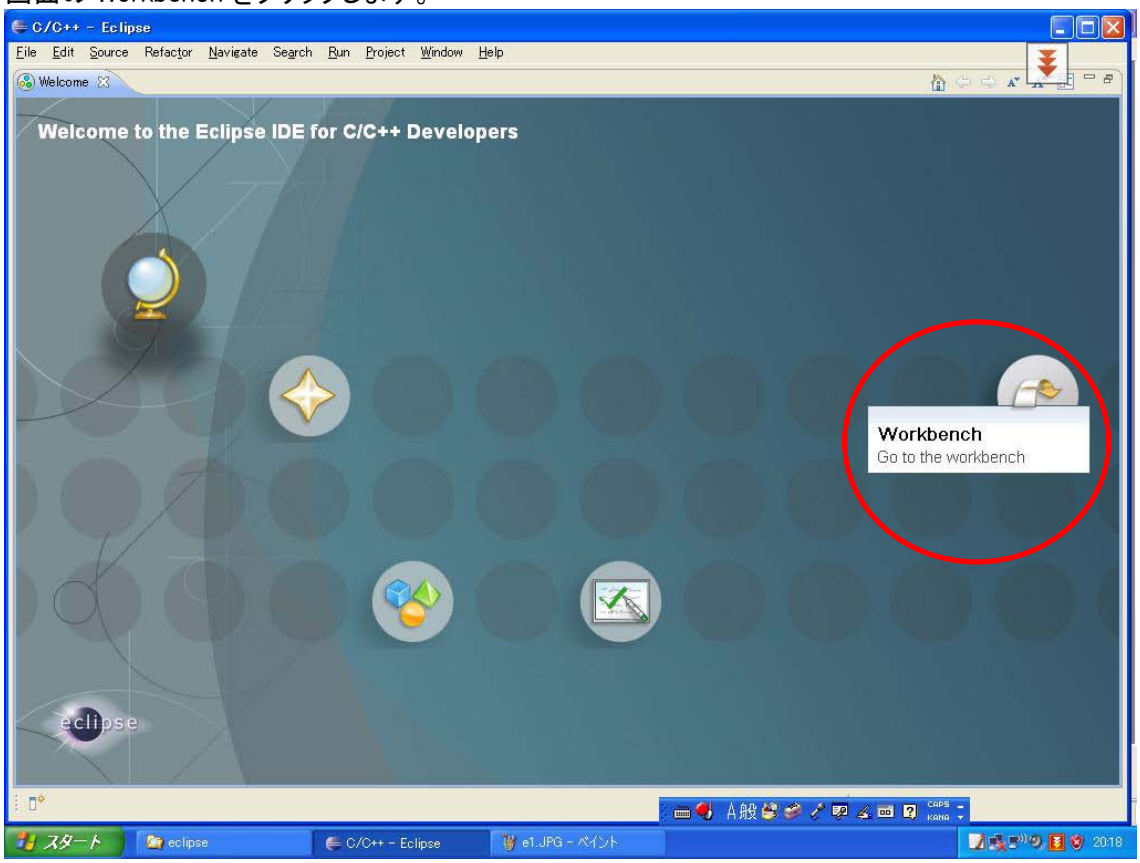
5.1 Eclipse を起動する



最初に Workspace の場所を聞いてきます。デフォルトは名前が長いので、"D:¥embeded¥eclipse¥workspace"に変更しました。



画面の Workbench をクリックします。



注意:サンプルを簡単に導入できるように、ダウンロードして前章で解凍してできた「D:¥OpenJTAG¥gcc_openjtag_eclipse_arm_example」の中の全てのフォルダーを「D:¥embeded¥eclipse¥workspace」へコピーしてください。

5.2 Eclipse プラグイン(Zylin Embedded CDT)インストール

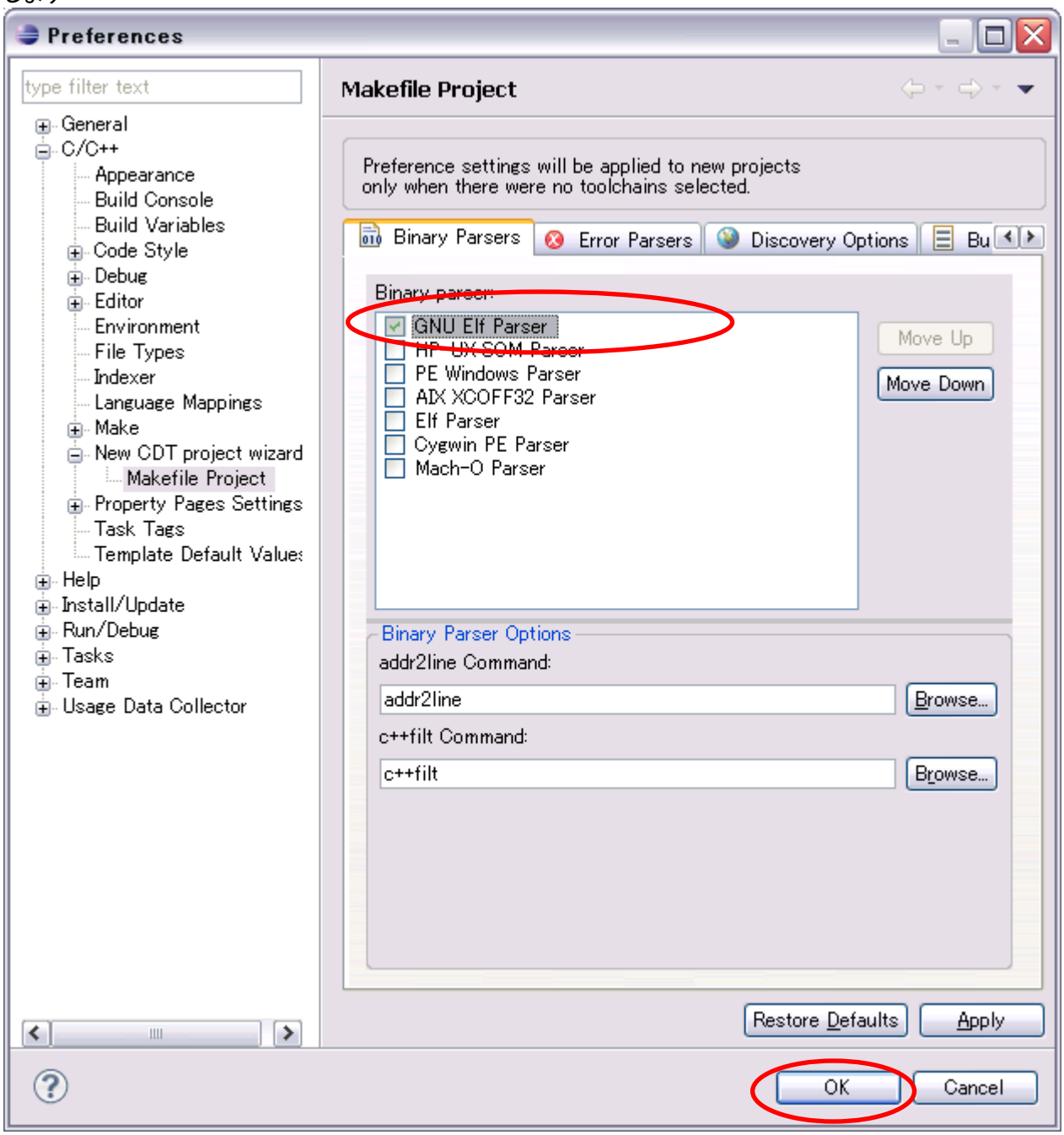
＊Zylin Embedded CDT の 4.81 バージョンは Zylin のホームページから取り下げますので、
弊社から圧縮ファイルをダウンロードしてから Eclipse にコピーしてください。
＊Zylin Embedded CDT の最新版では検証していないため、インストールしないでください。

Zylin Embedded CDT ダウンロード URL:
http://www.dragonwake.com/download/open-jtag/com.zylin.embeddedcdt_4.8.1.zip

上記ファイルを解凍し「Eclispe インストール先」の「plugins」というフォルダーにコピーしてください。
コピーが完了したら、Eclipse を再起動させます。

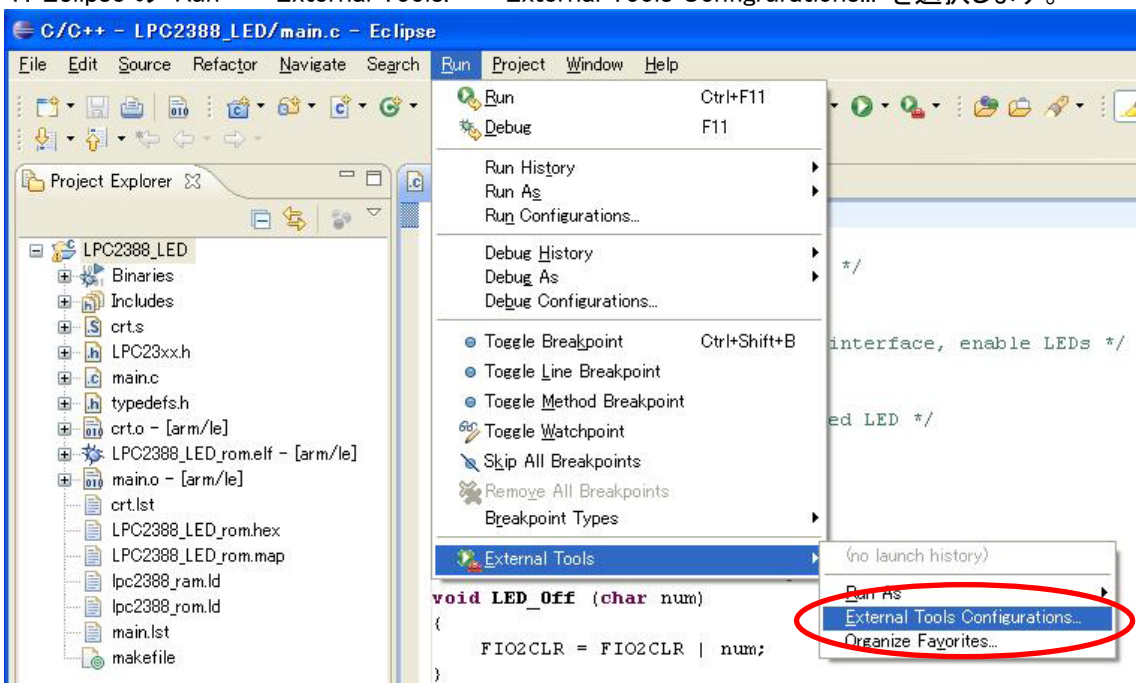
5.3 ビルドの設定

Eclipse の"Window"→"Preferences"を選択し、
Preferences の"C/C++ Build"→"Settings"を選択し"Binary Parsers"タブの"GNU Elf Parser"にチェックを入れて OK ボタンを押します

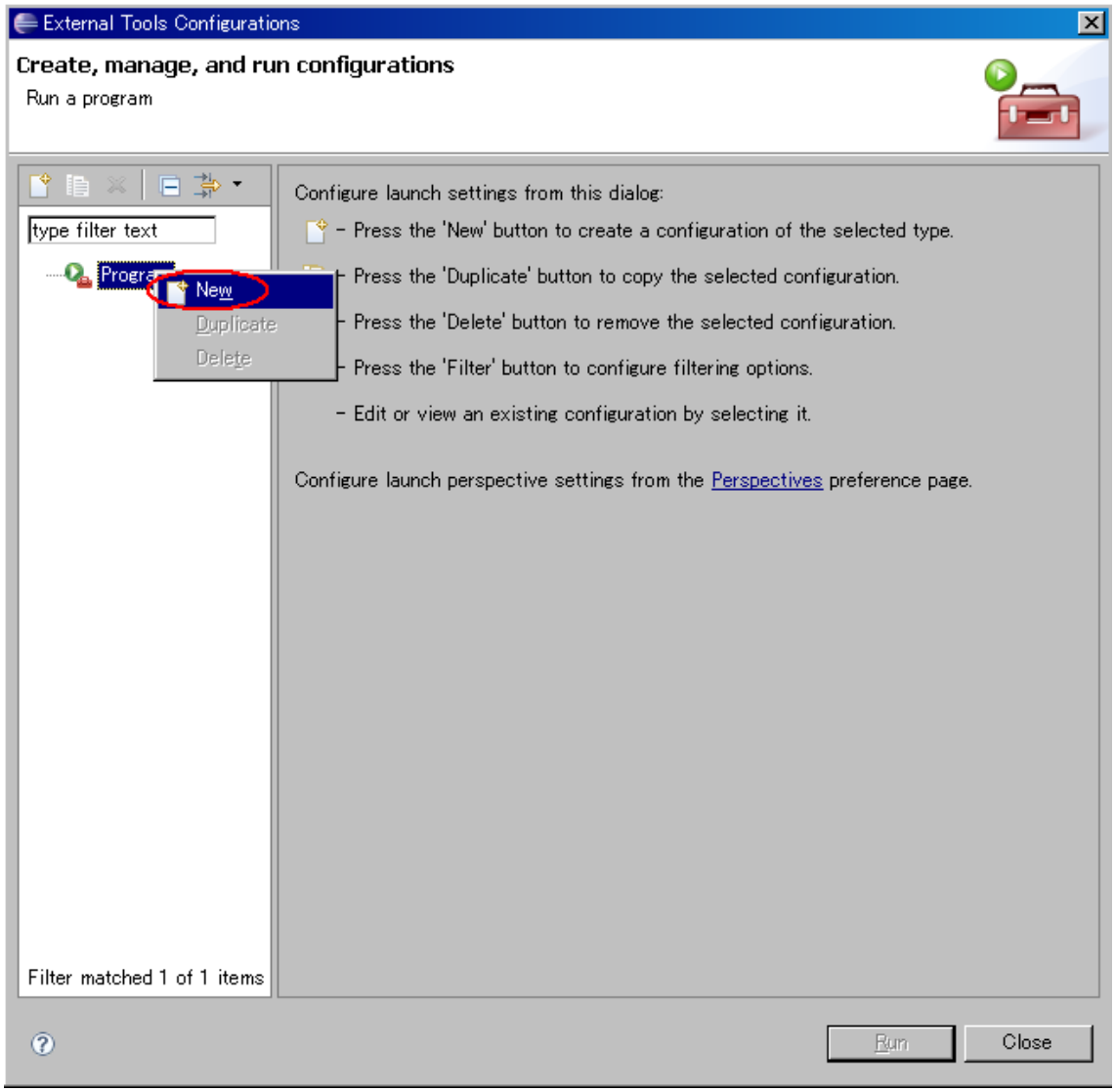


5.4 OpenOCD の設定

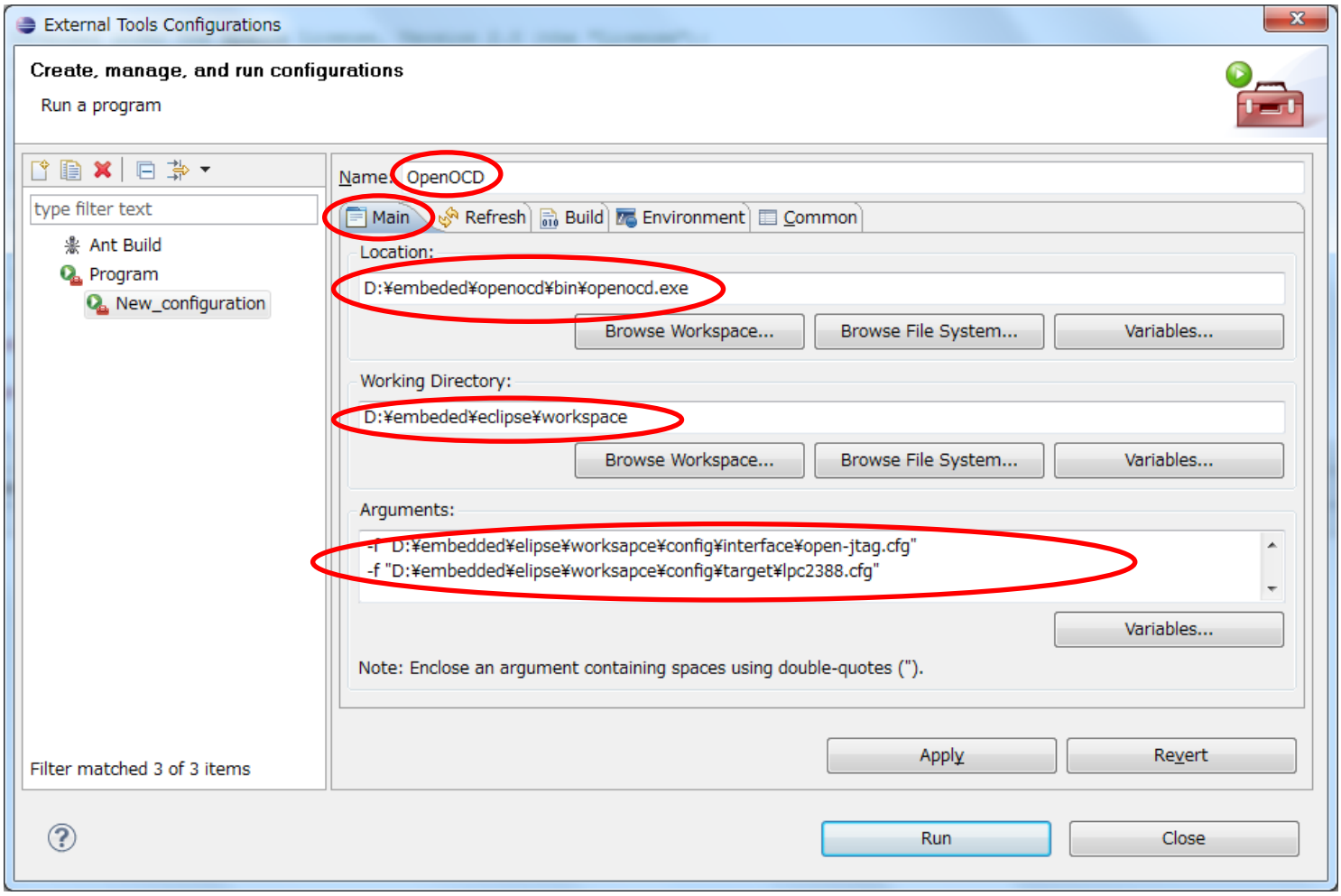
1. Eclipse の"Run"→"External Tools."→"External Tools Configrurations..."を選択します。

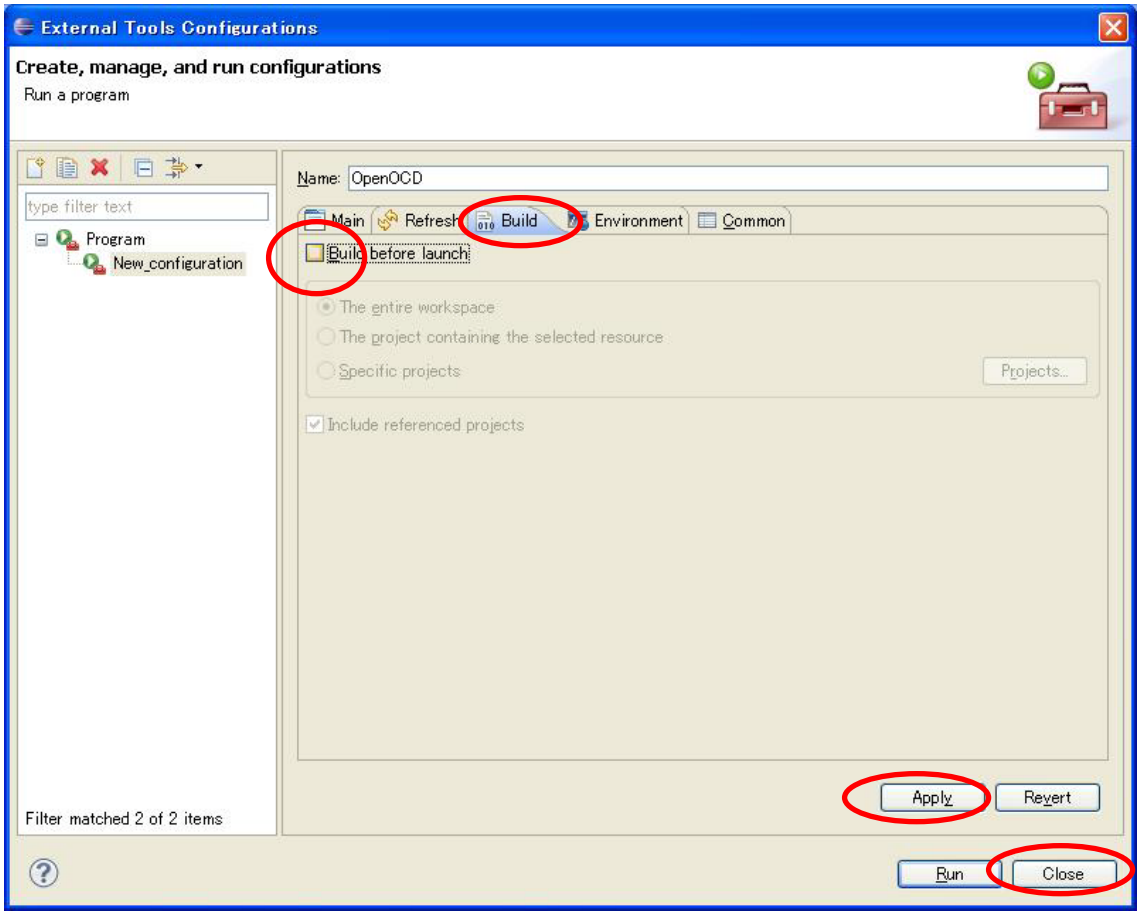


2. External Tools Configrurations の"Program"を右クリックし、"New"を選択します。



3. Main タブの"Name"に適当な名前を入力してください、例えば、"OpenOCD"。
"Location:"に"D:¥embeded¥openocd¥bin¥openocd.exe"、
"Working Directory:"に" D:¥embeded¥eclipse¥workspace "
"Arguments:"に-f "D:¥embeded¥elipse¥worksapce¥config¥interface¥open-jtag.cfg"
-f "D:¥embeded¥elipse¥worksapce¥config¥target¥lpc2388.cfg"と入力します。
＊基板種類(lpc2148,lpc2388,mini2440 など)により、Arguments の設定も異なる。

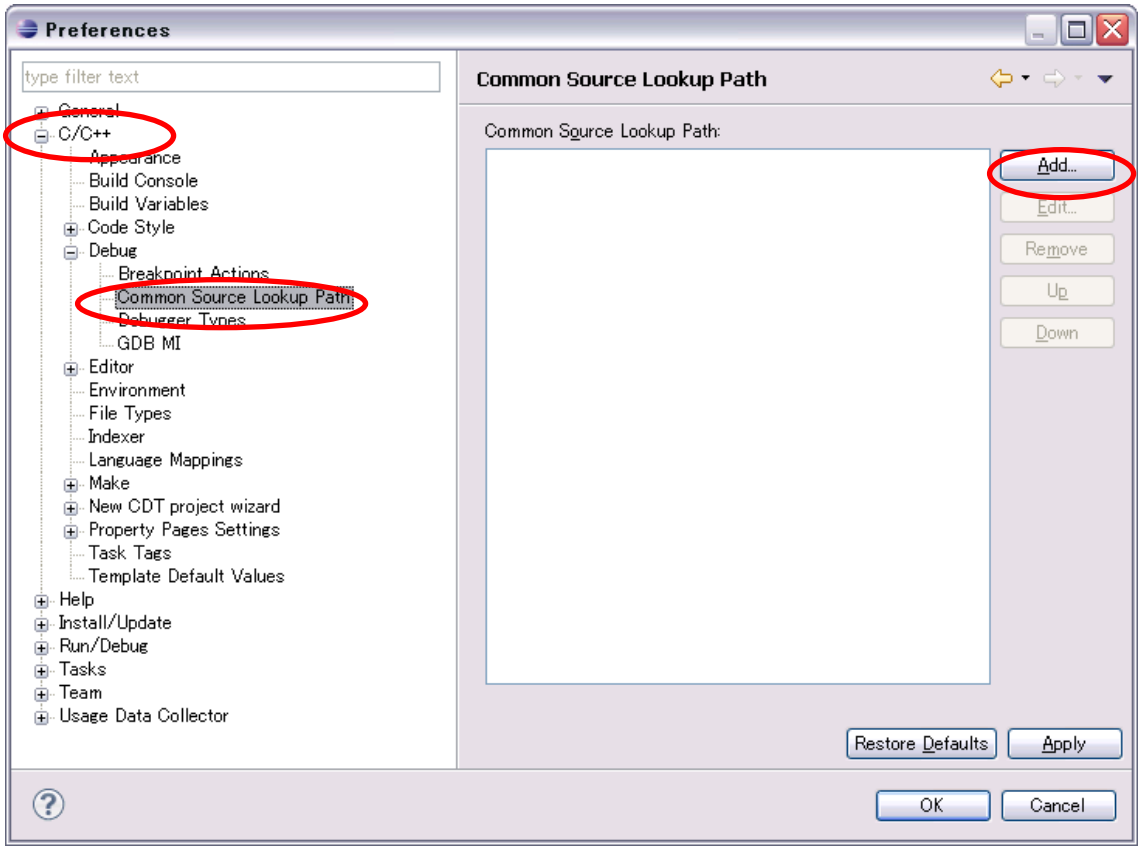




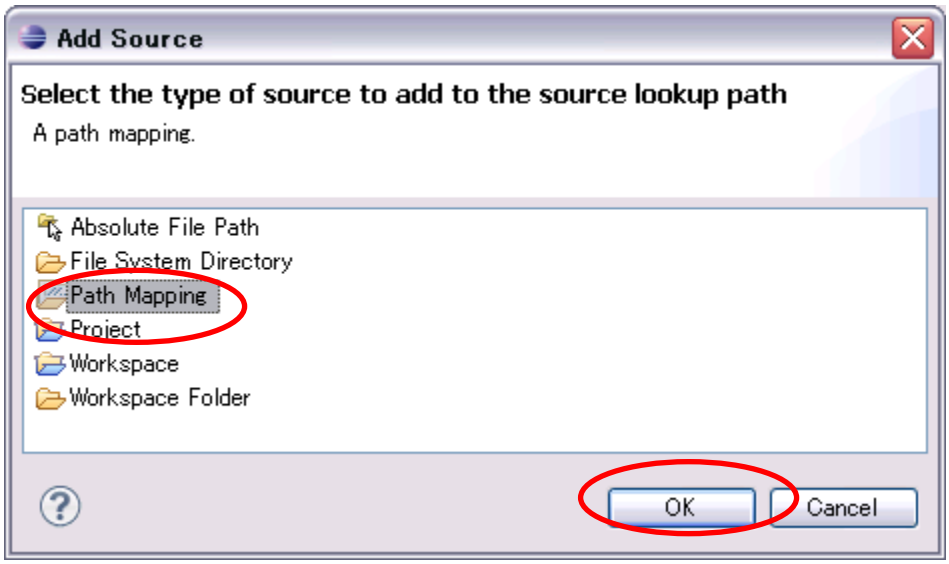
Build タブをクリックし"Build before launch"にチェックを外れます。全てを入力し終えたら"Apply"ボタンを押し、"Close"ボタンを押します。

5.5 Cygwin のソース・ルックアップ・パスの設定

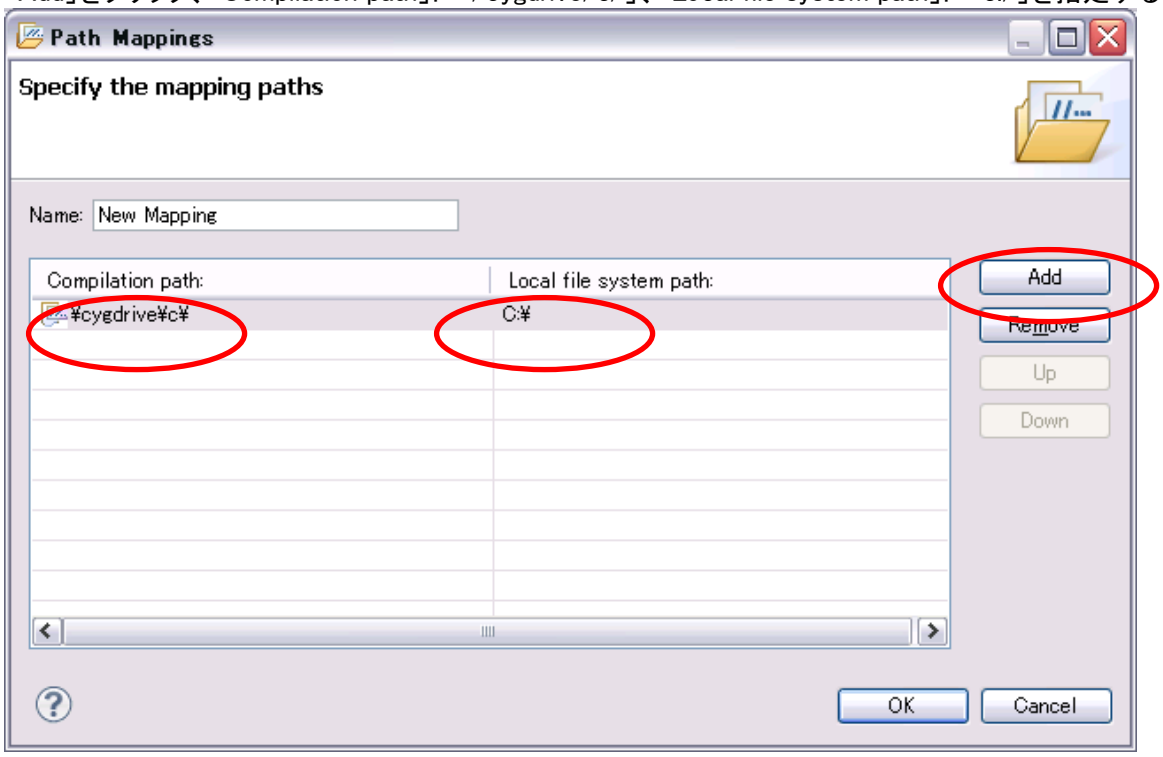
GDB でデバッグを開始すると「Can't find a source file at "/cygdrive/c/project/hoge/hoge.cpp"」なんて怒られる。Cygwin のパスを eclipse が認識できていない様 子。この場合は、ルックアップ・パスの設定が必要。
eclipse のメニュー[ウィンドウ]→[設定]で設定ダイアログを開き、[C/C++]→[Debug]→[Common Source Lookup path]にルッ クアップ設定を追加する。



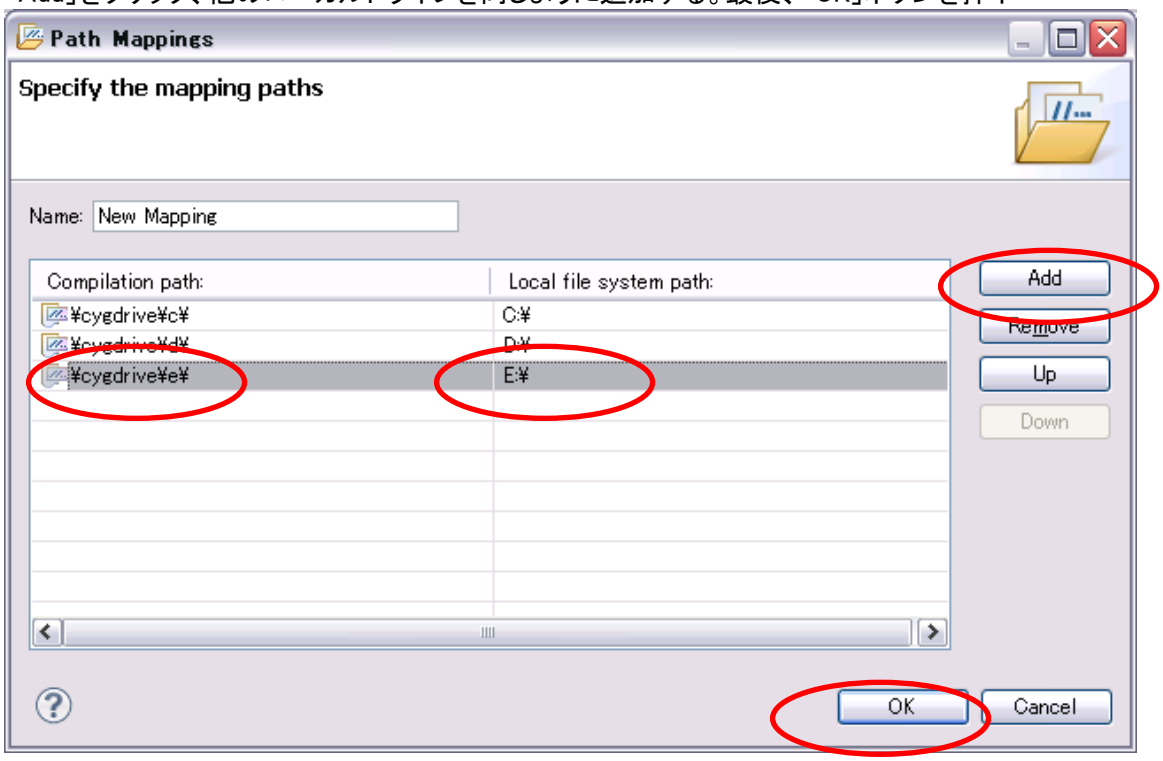
「Path Mapping」を選択、「OK」を押下



「Add」をクリック、「Compilation path」に「/cygdrive/c/」、「Local file system path」に「c:/」を指定する。

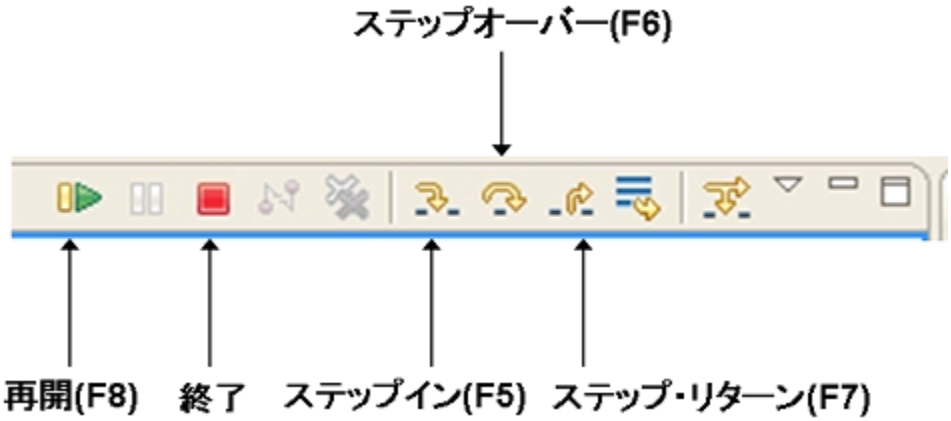


「Add」をクリック、他のローカルドライブを同じように追加する。最後、「OK」ボタンを押下



5.6 Eclipse デバッグ用のコマンド及びショットカットキー一覧

詳しくは Eclipse に関わるドキュメントを参照



ステップ実行において良く使われる操作の一覧を以下に示します。

操作名	ショートカットキー
再開	F8
ステップイン	F5
ステップオーバー	F6
ステップ・リターン	F7

ステップ実行とは関係ありませんが、前回起動したクラスを再度実行したりデバッグする場合は、以下のショートカットキーが便利です。

操作名	ショートカットキー
前回の起動を実行	Ctrl + F11
前回の起動をデバッグ	F11

第六章 ARM シリーズデバッグ手順

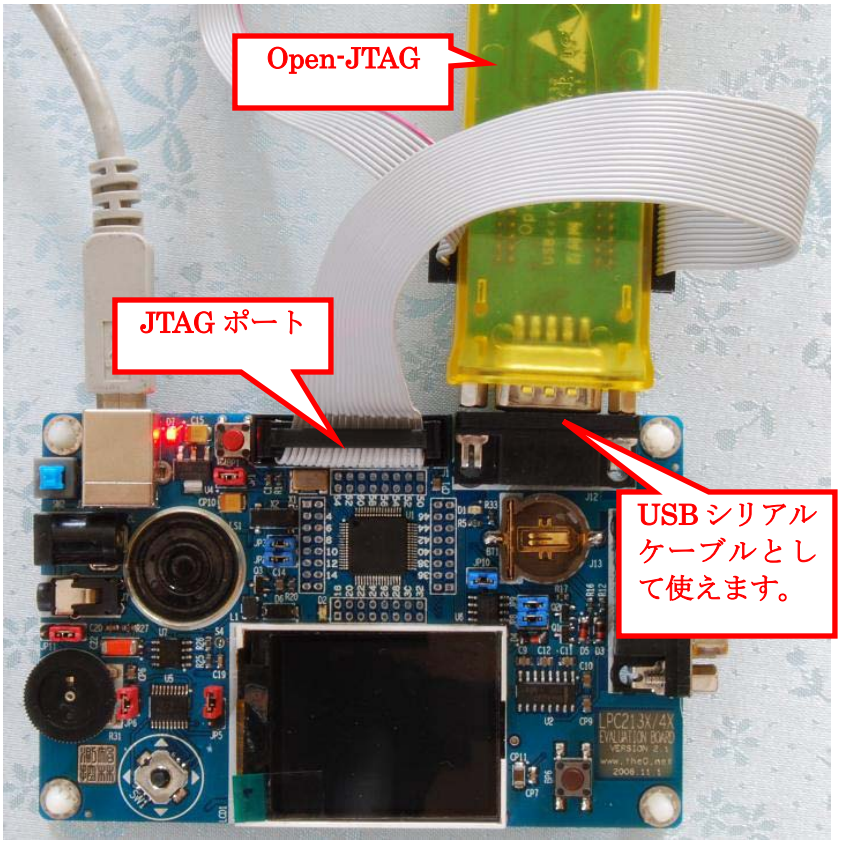
＊本書に使われる ARM シリーズは全て弊社の製品となります。
詳しくは弊社の通販サイト：http://www.csun.co.jp までにご参照ください。
第四章「インストール手順」、第五章「Eclipse の設定」に従って行っていれば、ARM シリーズ、例えば、ARM7（LPC2148、LPC2388）、Contex-M3（STM32F103）、ARM9(MINI2440)のサンプルを導入のうえ、デバッグ手順を説明させて頂きます。

6.1 ARM7 の LPC2148

6.1.1 LPC2148 ボード購入 URL

http://www.csun.co.jp/SHOP/2009071209.html
ボード URL から LPC2148 マニュアルを参照し、Open-JTAG に接続してください。
LPC2148 マニュアルダウンロード URL：
http://www.dragonwake.com/download/LPC2148/LPC2148_manual.pdf

6.1.2 ハードウエア動作確認



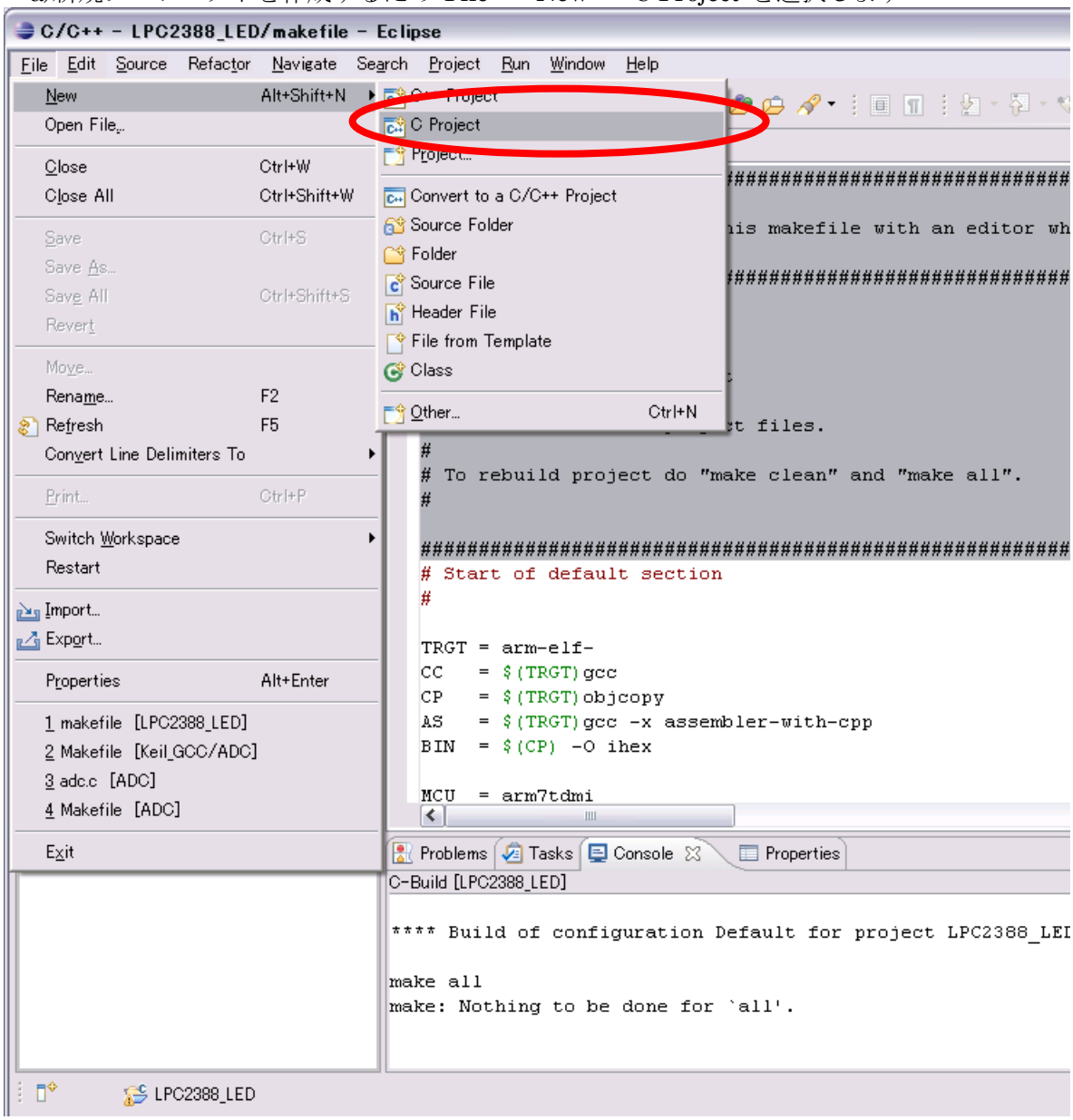
1. OpenJTAG をパソコンの USB ポートに挿入する
2. JTAG ケーブルで OpenJTAG と LPC2148 ボードを繋ぐ
3. LPC2148 ボードに電源を入れる
4. 下記のコマンドを入力します。

openocd -f "D:¥embeded¥eclipse¥workspace¥config¥interface¥open-jtag.cfg" -f "D:¥embeded¥eclipse¥workspace¥config¥target¥lpc2148.cfg"

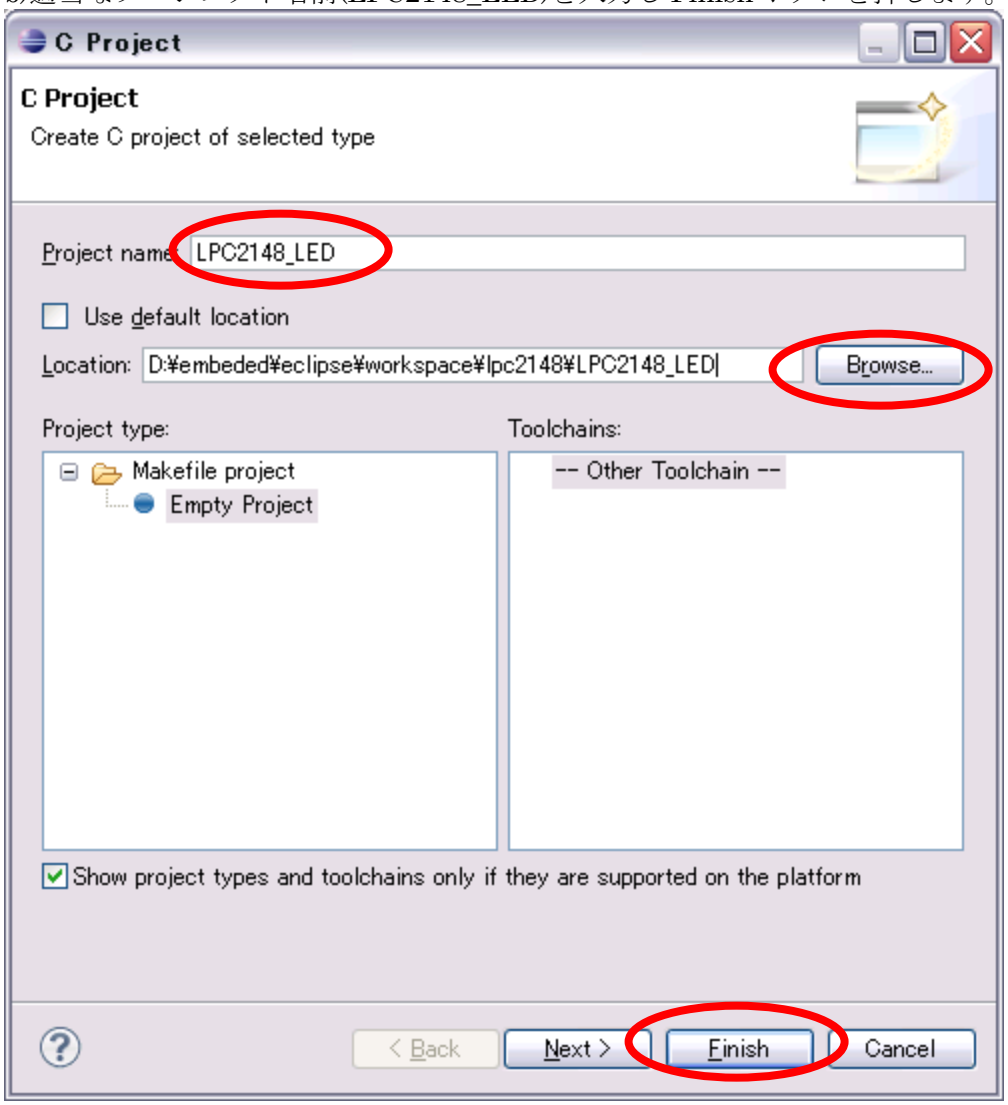
はじめの-f は OpenJTAG のコンフィグファイルを使います。二番目の-f は lpc2148 のコンフィグファイルを使います。

6.1.3 LPC2148 用のサンプル「LED」をデバッグ

サンプルダウンロード URL：
http://www.dragonwake.com/download/LPC2148/Example-2148GCC.rar
*サンプルの導入について、二つ方法があります、一つは新規プロジェクトを作成してからソースを導入、もう一つはダウンロードプロジェクトをインポートします。
＊ここに新規プロジェクトの作成として説明
１．プロジェクト「LPC2148_LED」を作る
a)新規プロジェクトを作成するため"File"→"New"→"C Project"を選択します

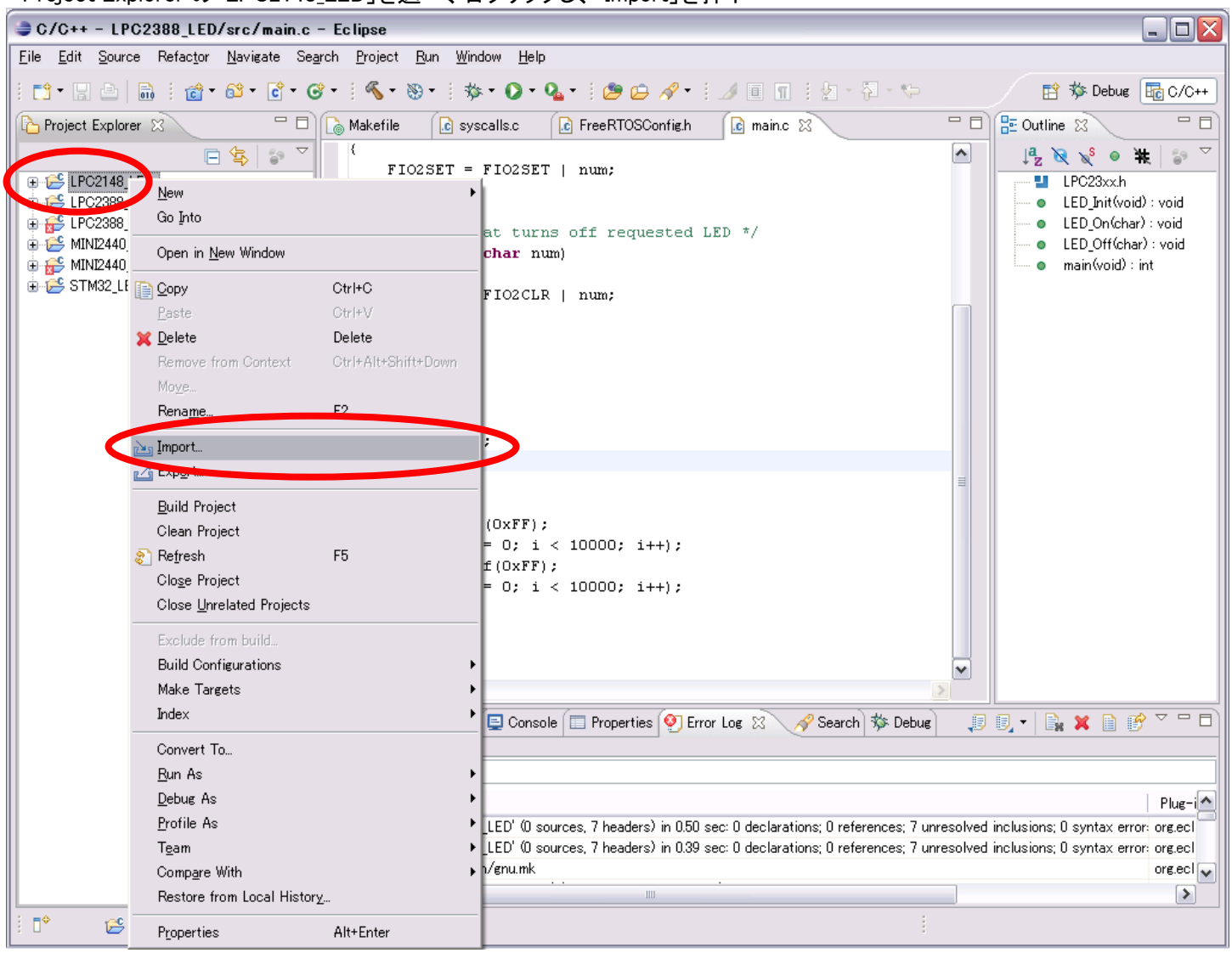


b)適当なプロジェクト名前(LPC2148_LED)を入力し Finish ボタンを押します。

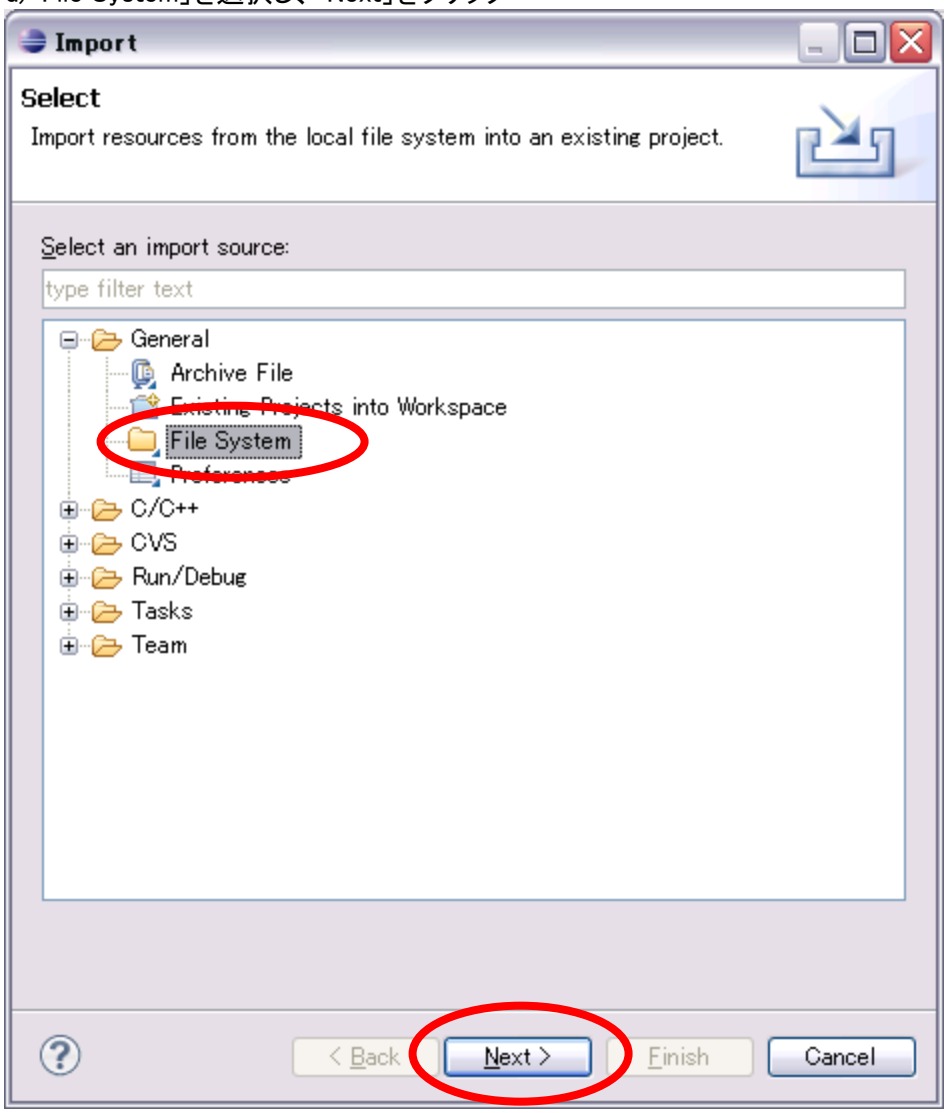


c)サンプルソースを作成されたプロジェクトにインポート

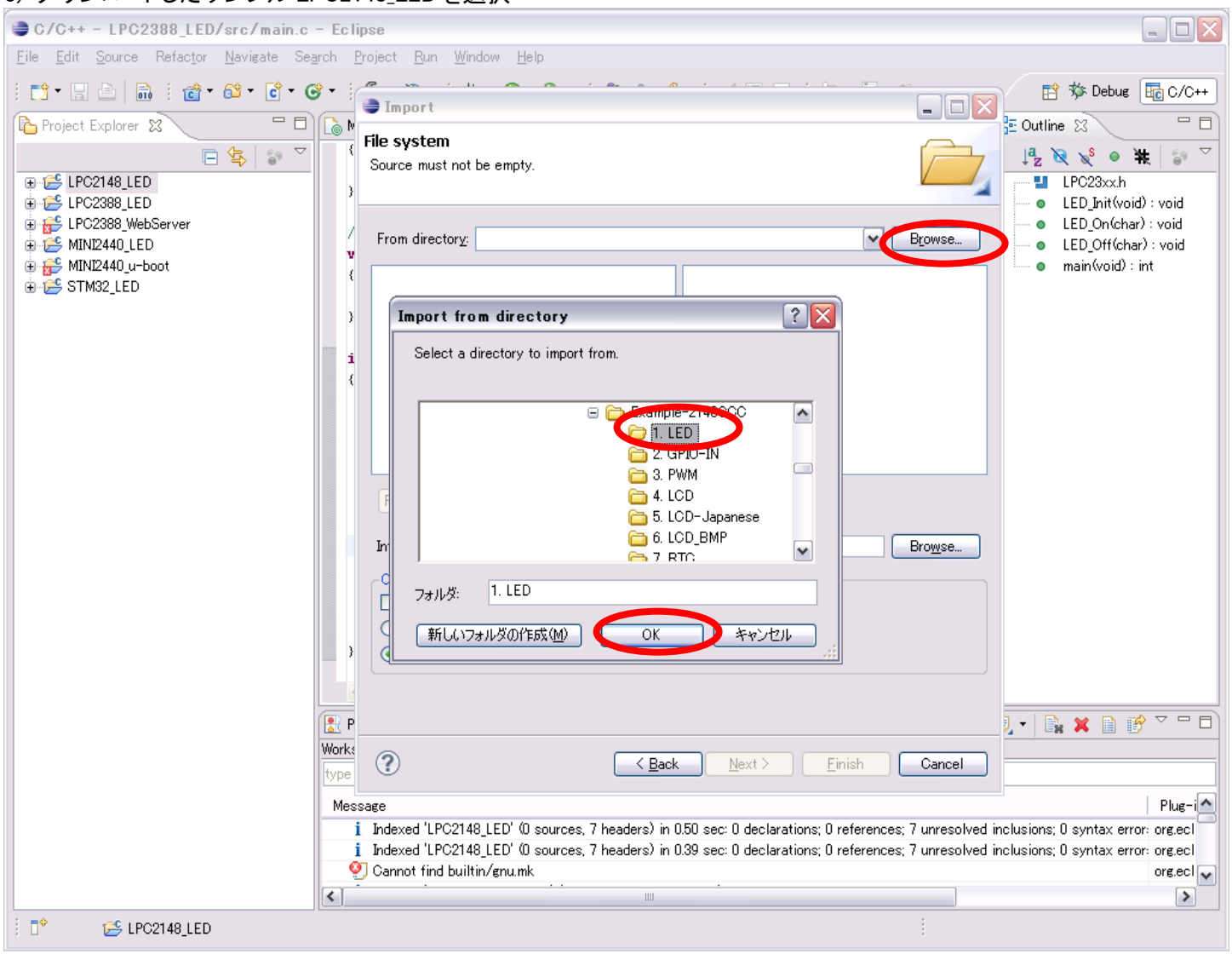
Project Explorer の「LPC2148_LED」を選べ、右クリックし、「Import」を押下



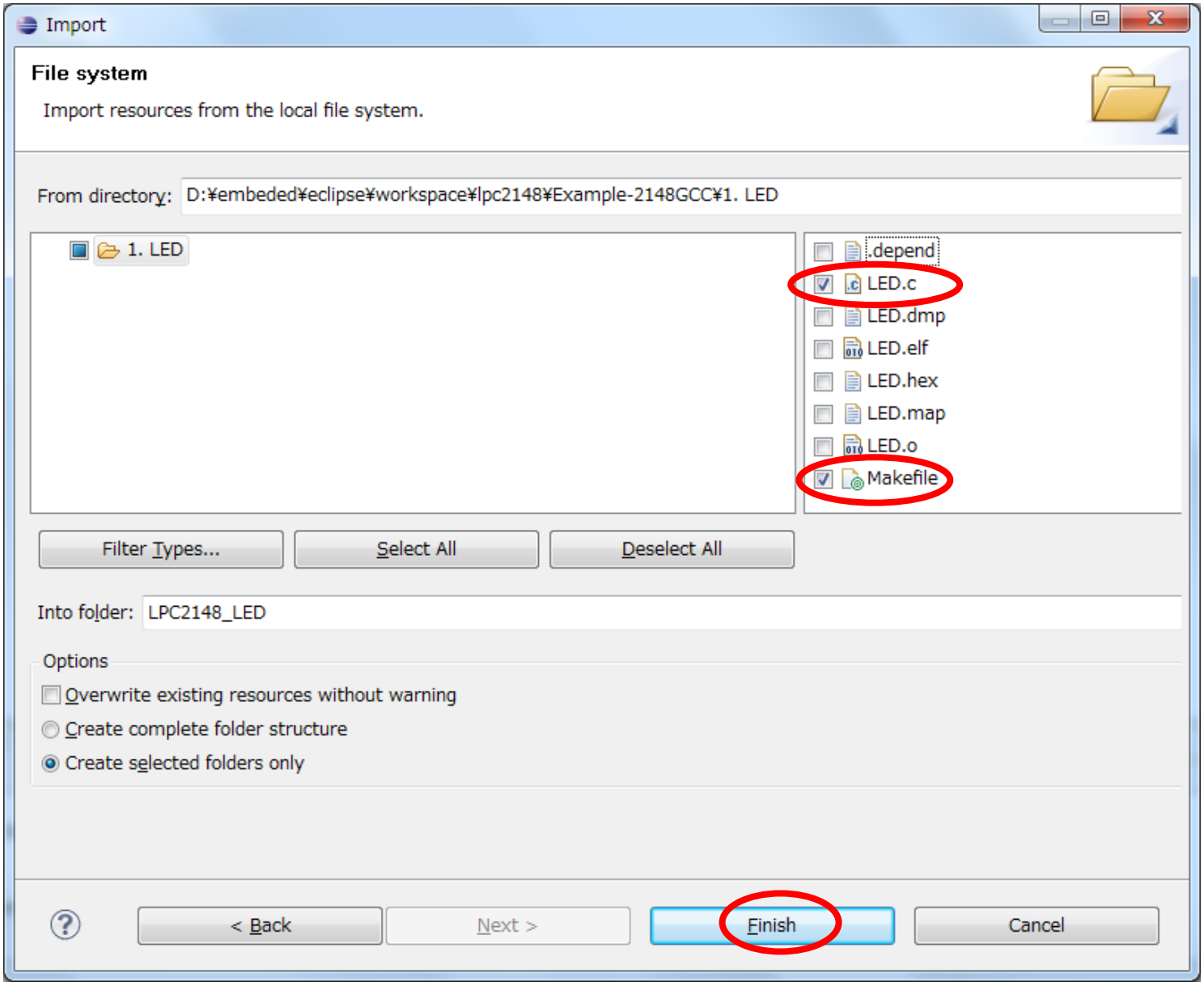
d)「File System」を選択し、「Next」をクリック



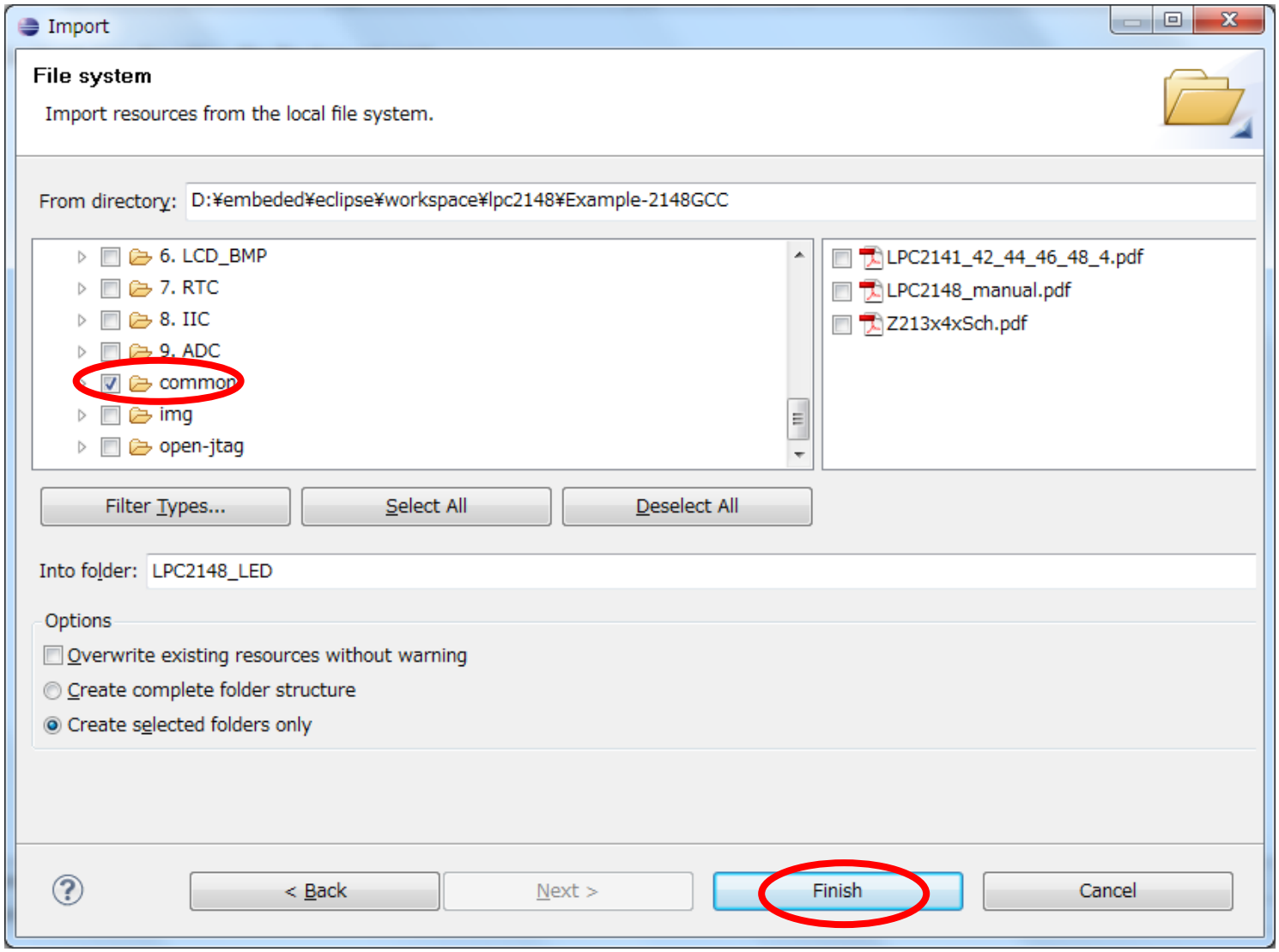
e）ダウンロードしたサンプル LPC2148_LED を選択



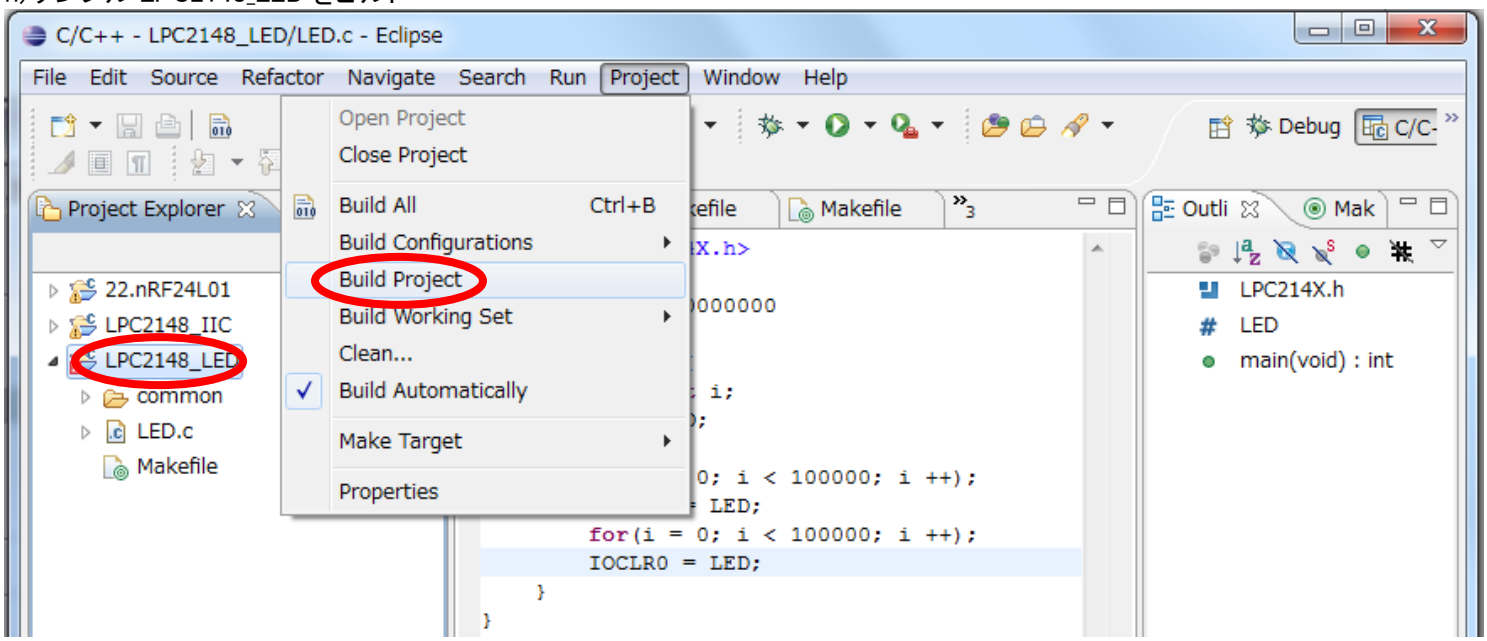
f)ソースファイルと Makefile を導入



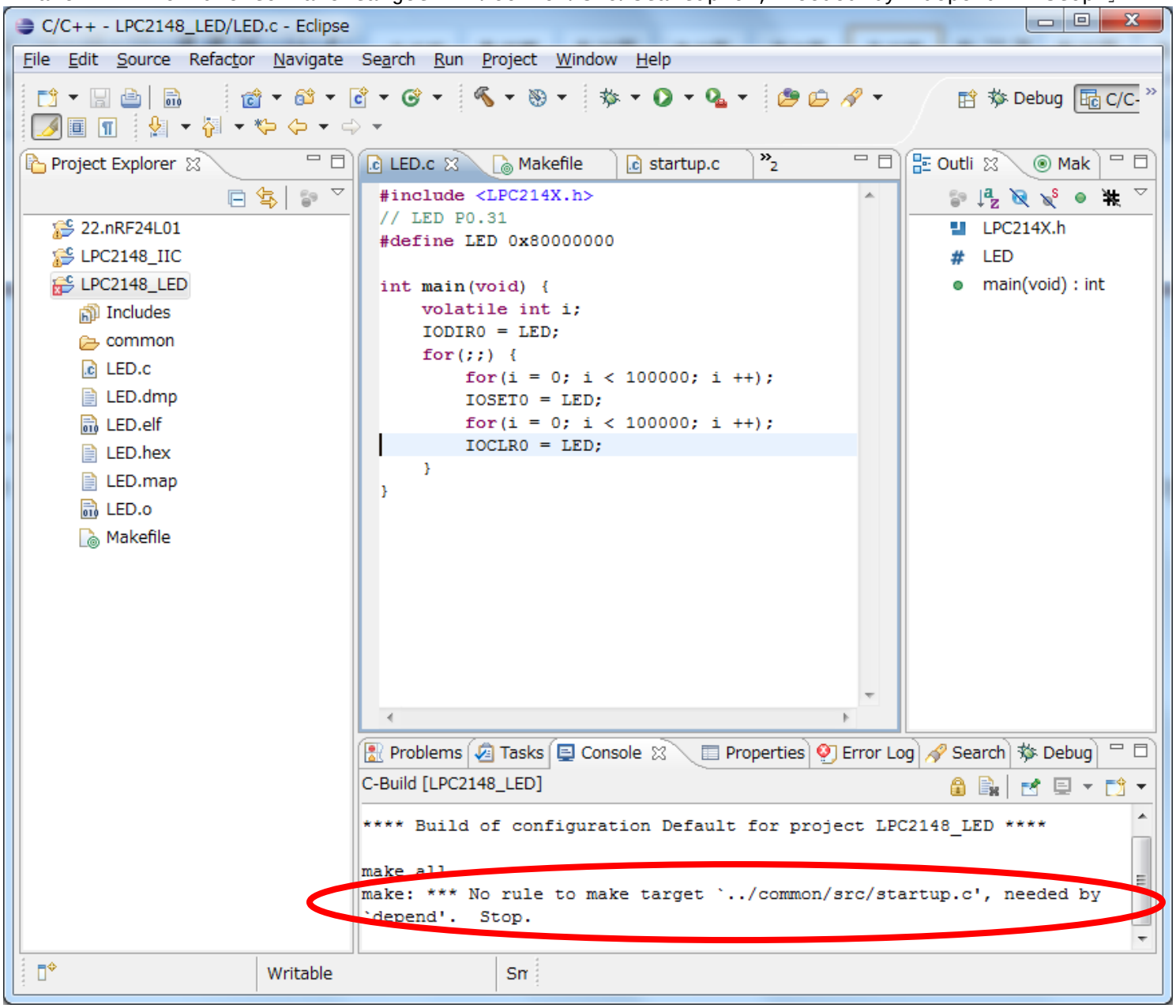
g）共通のソースも導入（他の操作は上記 c）、d）と同じ）



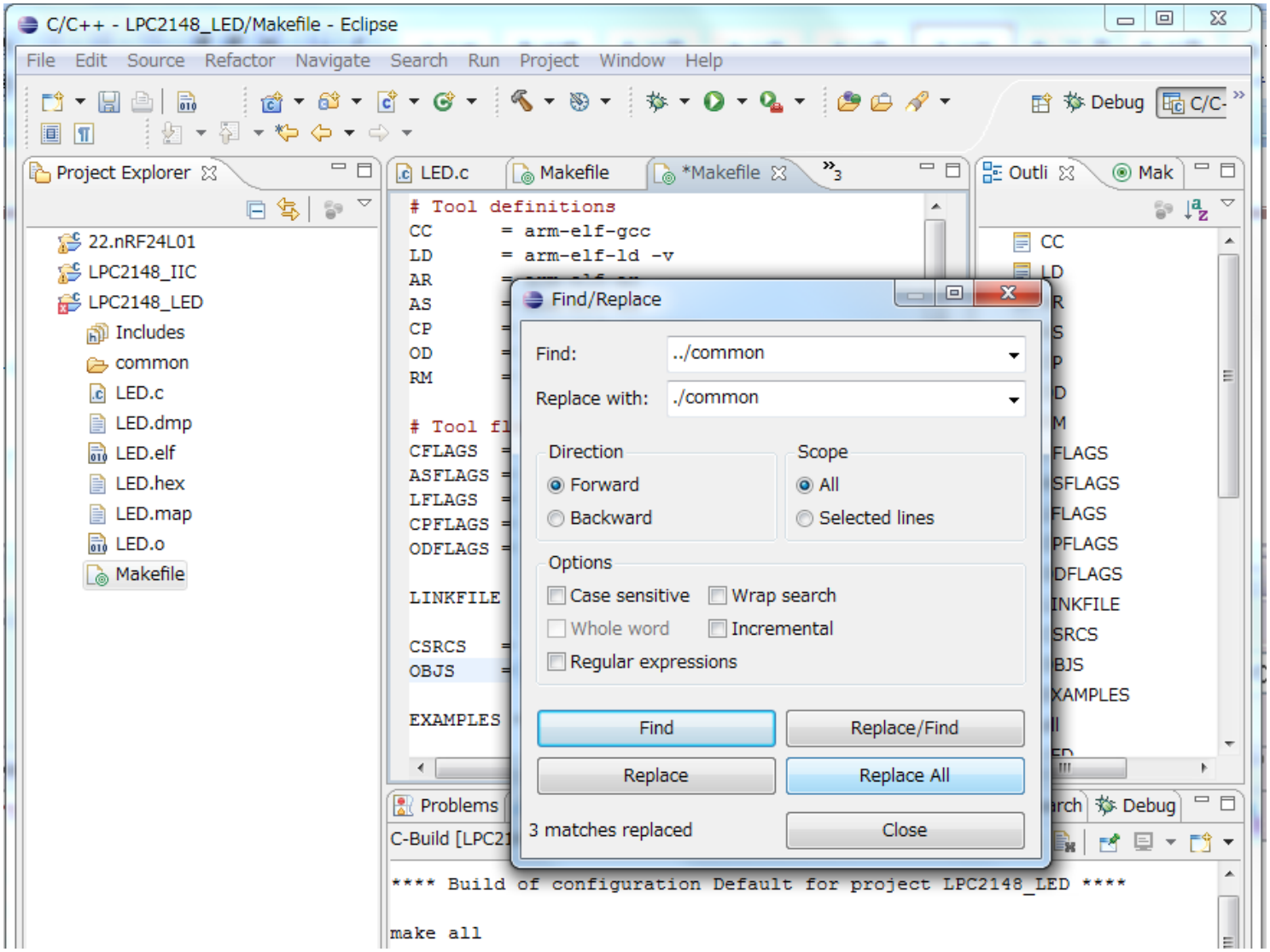
h)サンプル LPC2148_LED をビルド



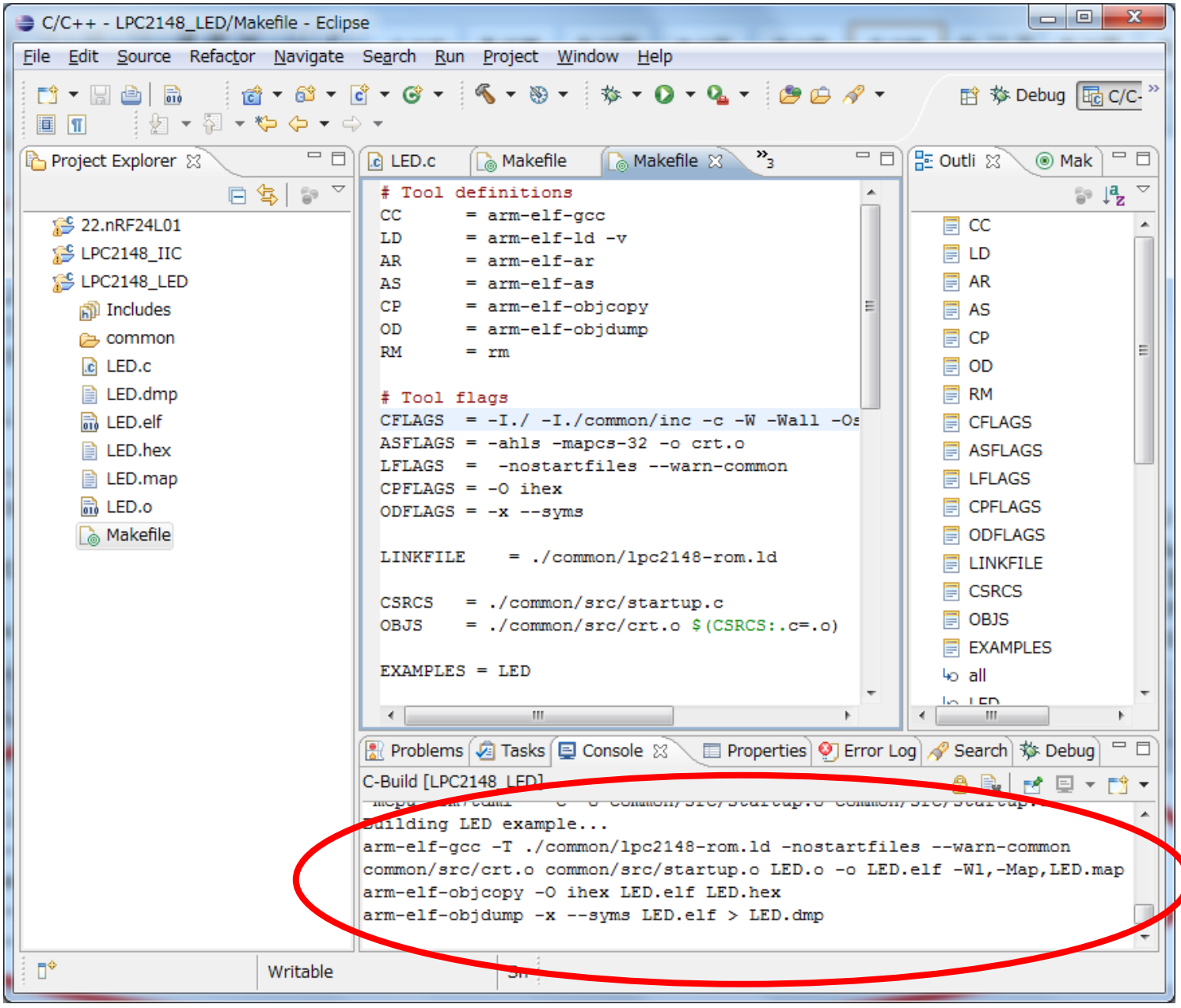
i)ビルドエラーメッセージ（エラーメッセージがなければ、i）を飛ばしてj)を直接参照）
「make: *** No rule to make target `../common/src/startup.c', needed by `depend'. Stop.」



j)「common」こういうソースのパスを「Makefile」に修正
「Edit」→「Find/Replace」
「Find」:「../common」を入力、「Replace with」:「./common」を入力、「Replace All」をクリック
置き換えの後、「File」→「Save」をクリック



k)修正後、再度ビルド（赤丸に囲まれるメッセージが出てきれば、ビルドが成功ということを判明）



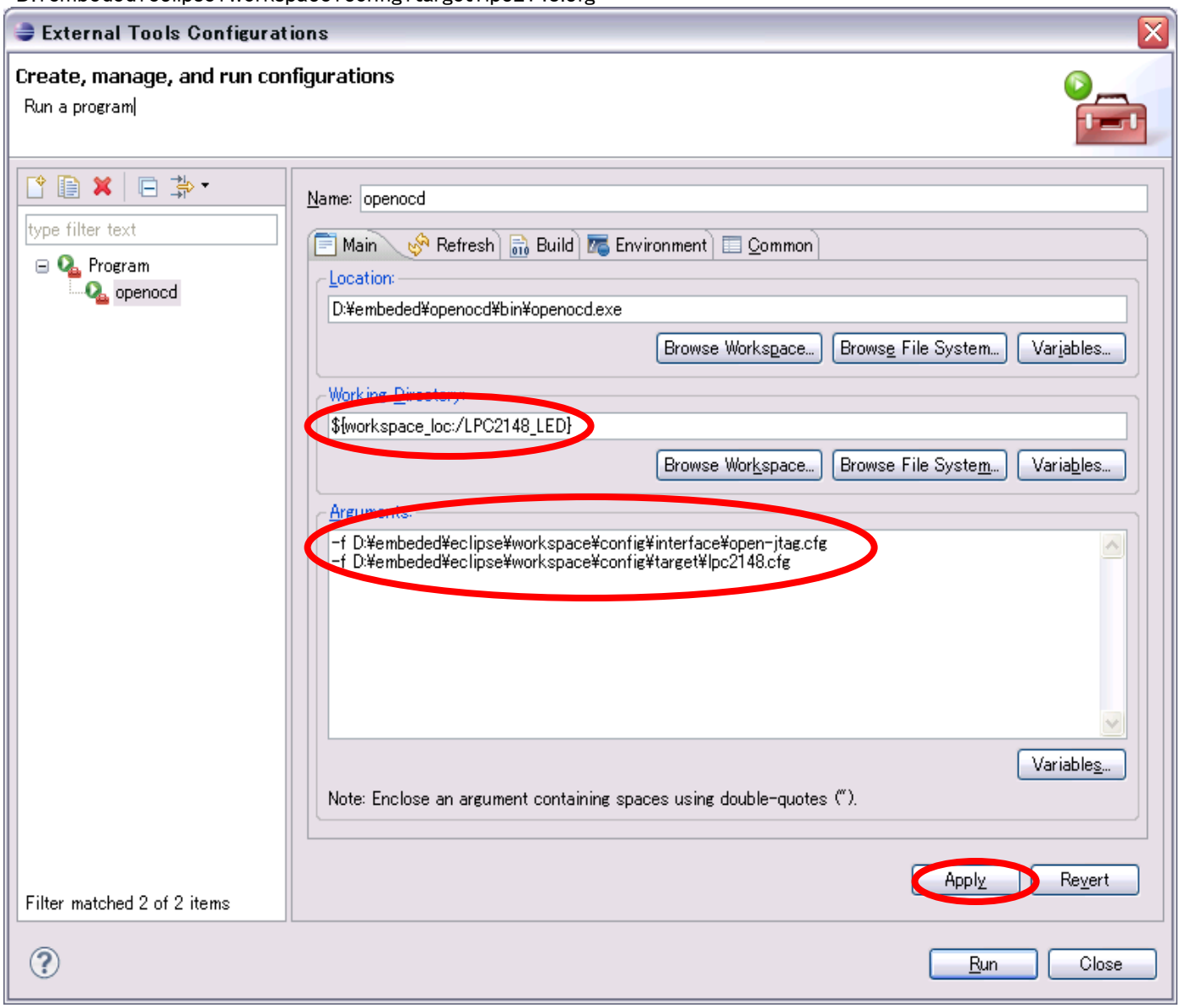
２．openocd 設定の修正

一般の設定は「5.4 OpenOCD の設定」を参照

a)「Working Directory」を「LPC2148_LED」に変更

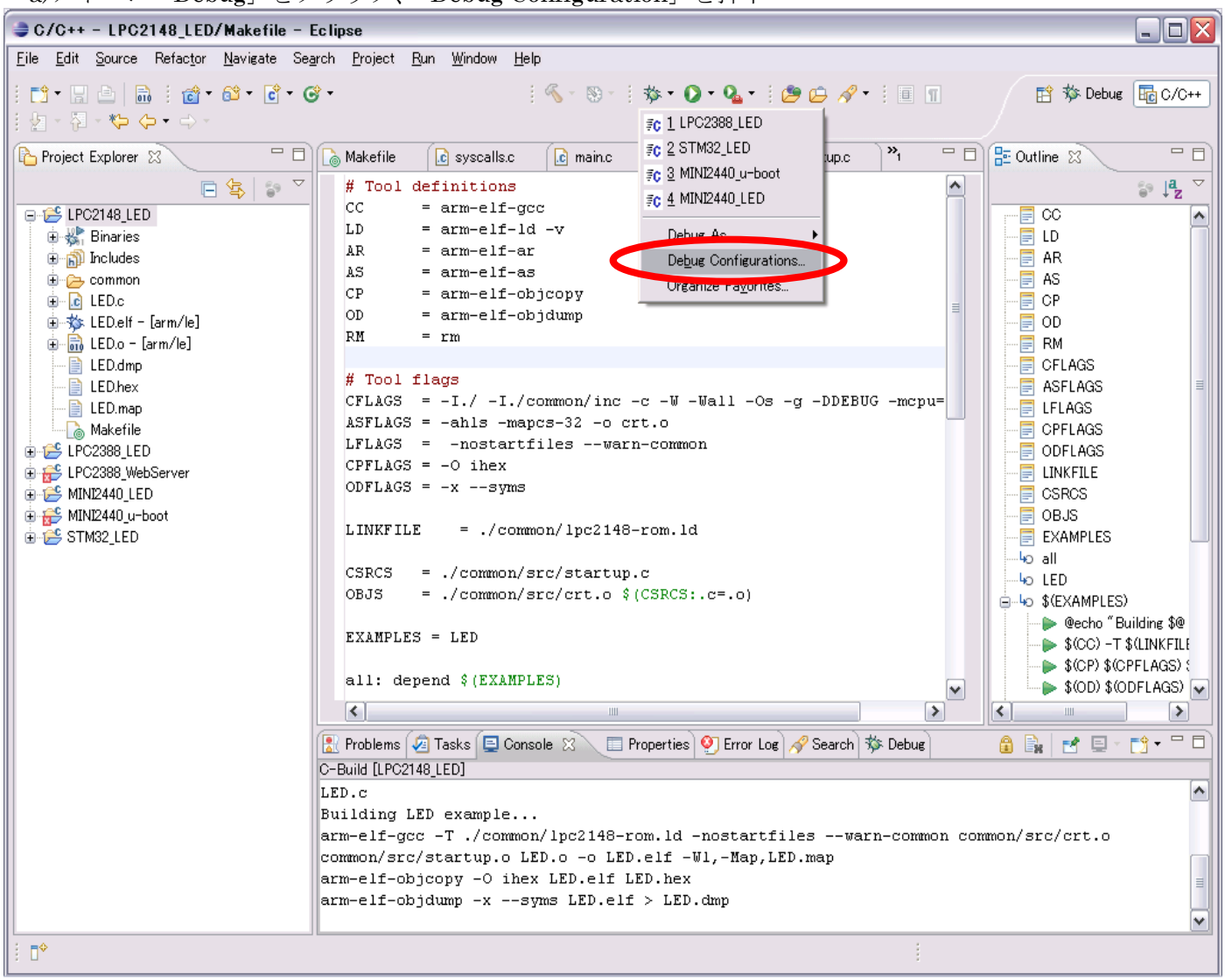
b)「Arguments」を下記のように変更

-f "D:¥embeded¥eclipse¥workspace¥config¥interface¥open-jtag.cfg" -f "D:¥embeded¥eclipse¥workspace¥config¥target¥lpc2148.cfg"

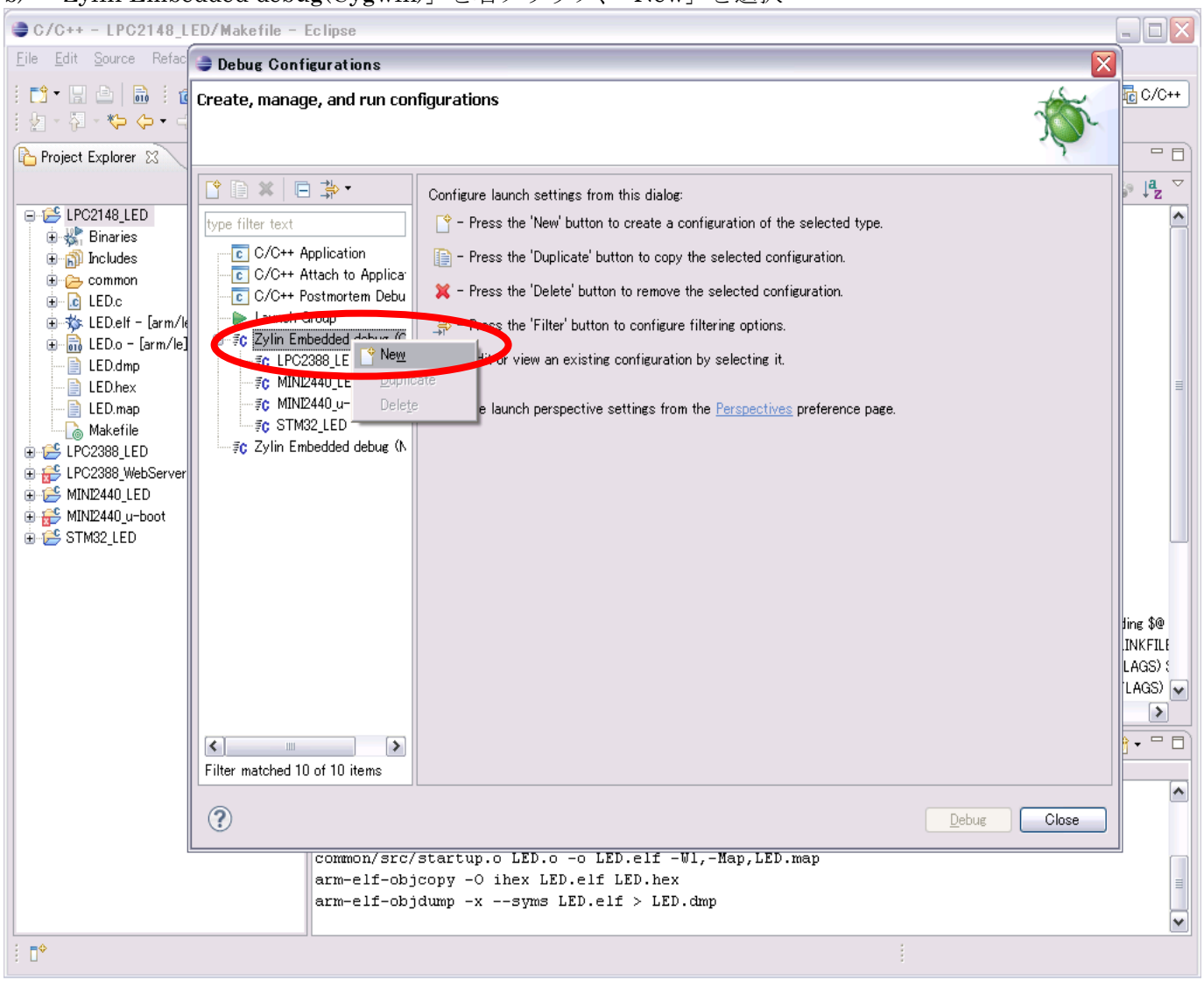


３．LPC2148_LED 用の GDB を追加

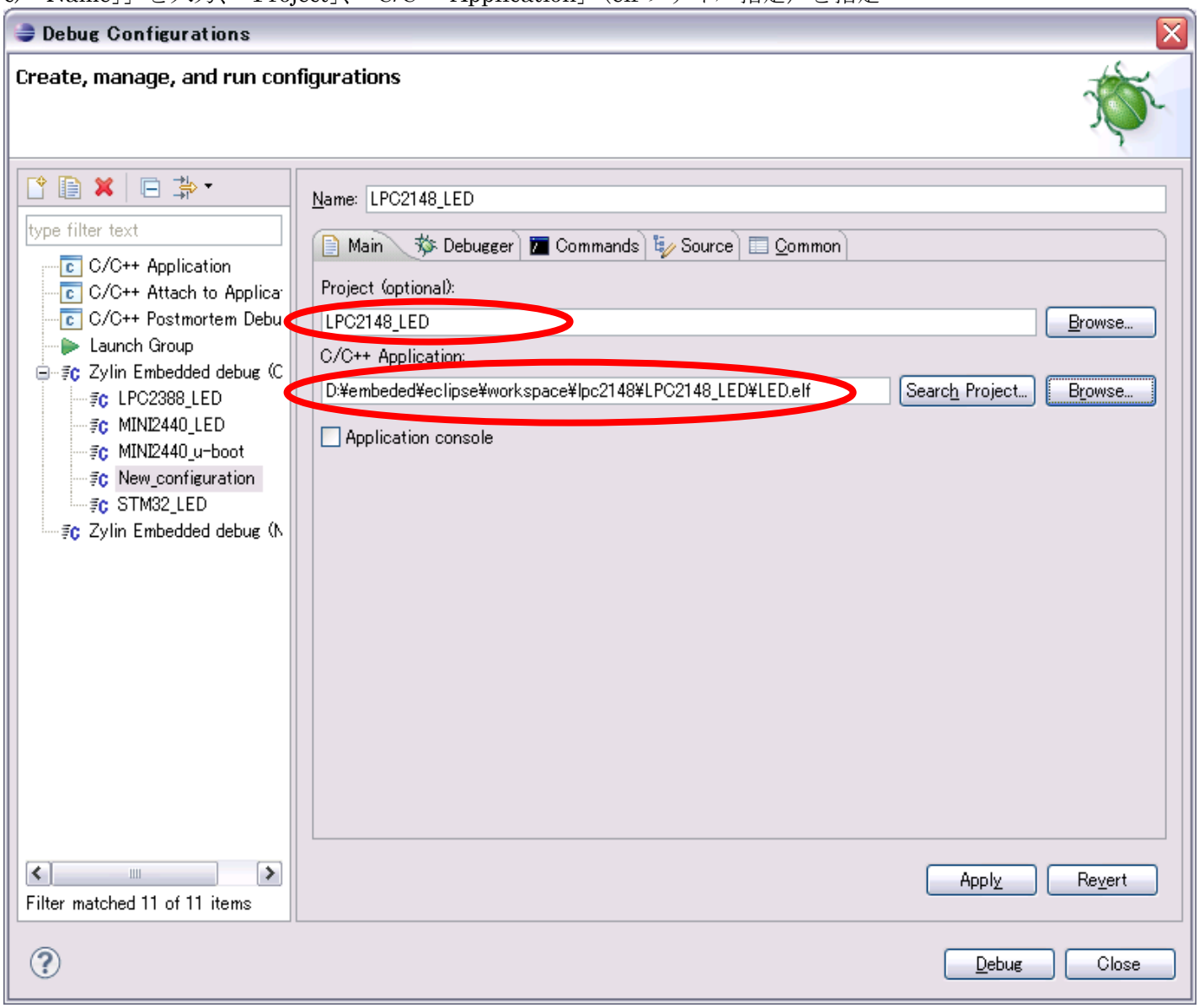
a)アイコン「Debug」をクリック、「Debug Configuration」を押下



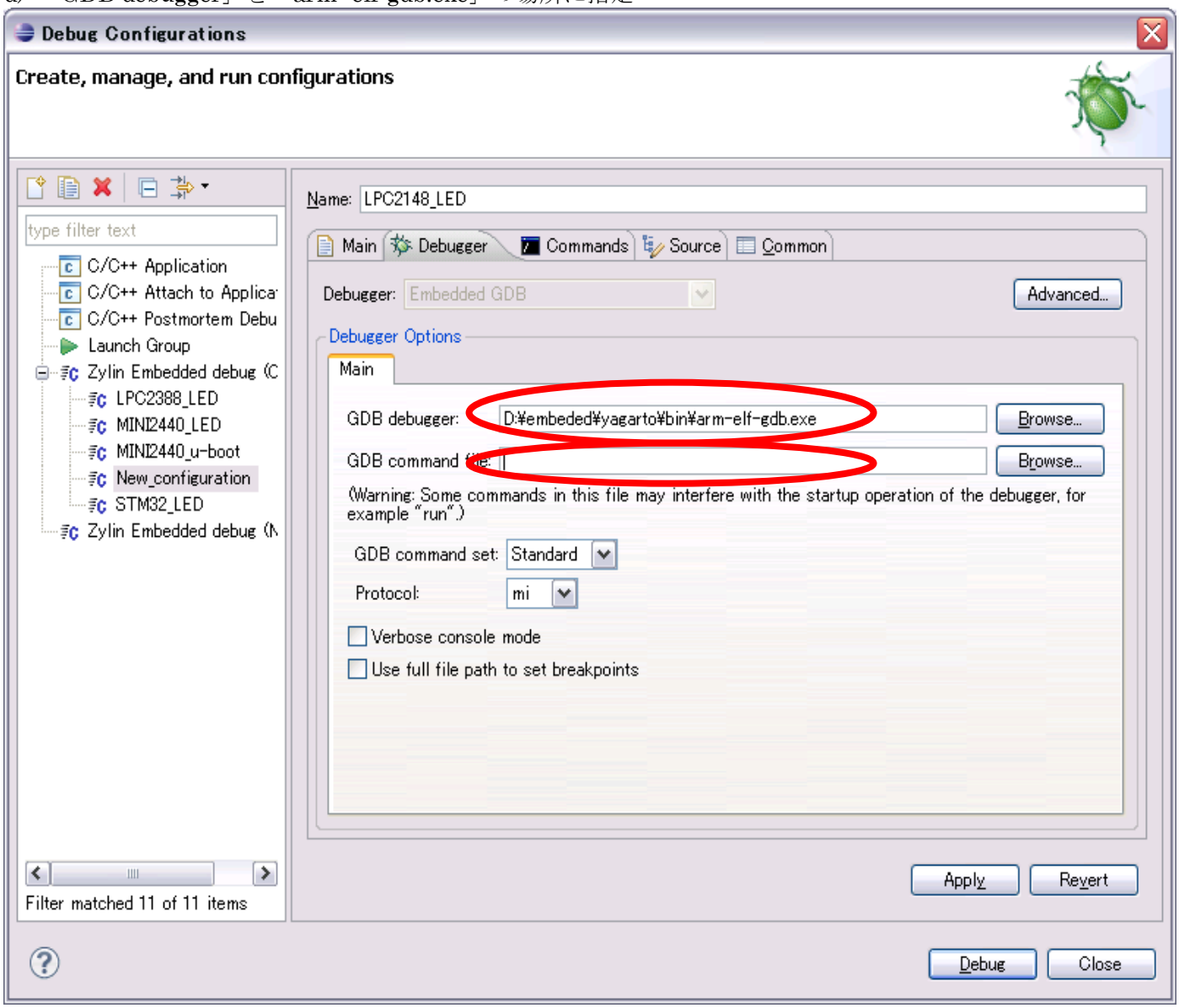
b) 「Zylin Embedded debug(Cygwin)」を右クリック、「New」を選択



c)「Name」」を入力、「Project」、「C/C++ Application」（elf ファイル指定）を指定

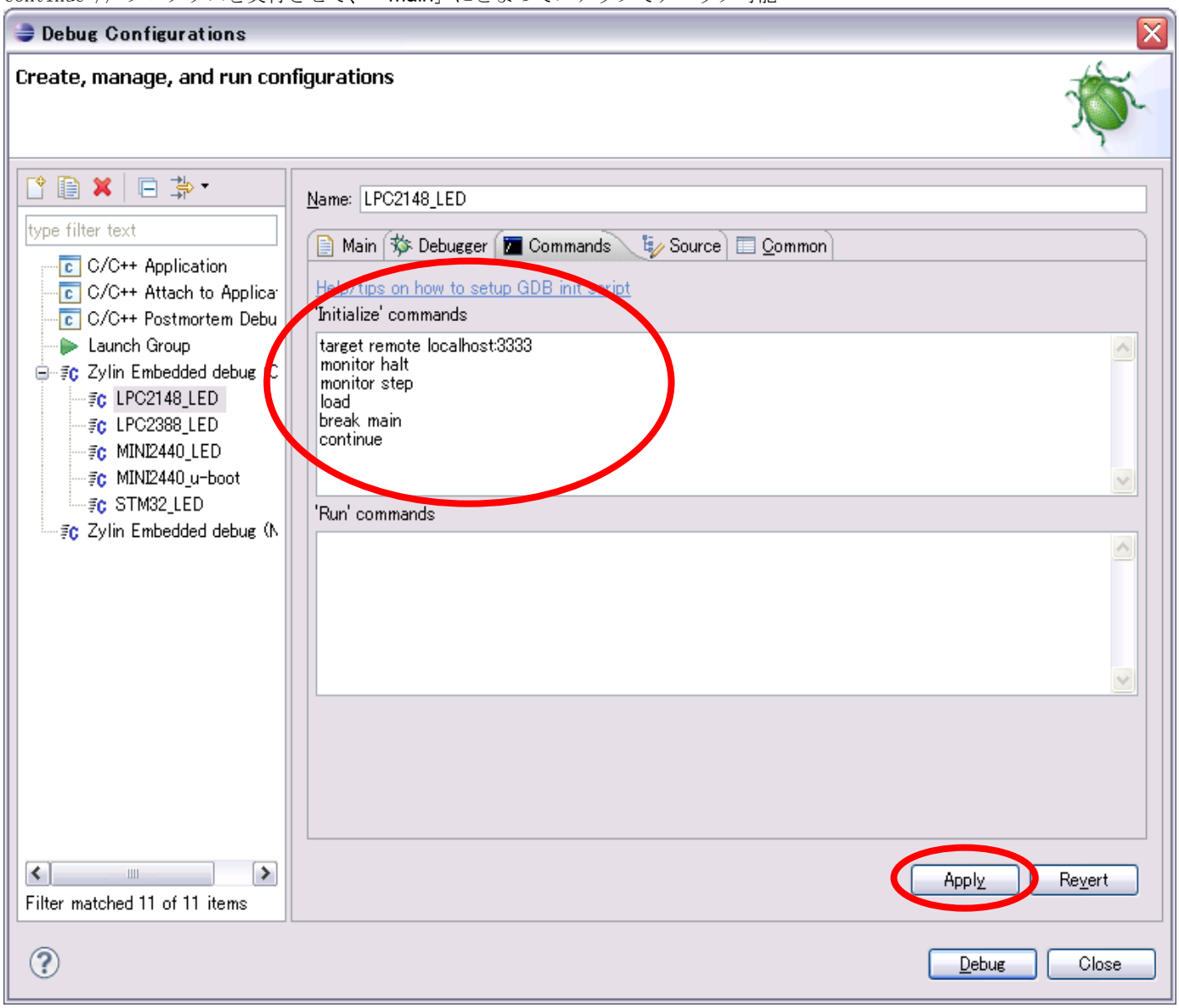


d) 「GDB debugger」を「arm -elf-gdb.exe」の場所に指定



e) 「Commands」を入力（//の右はコメントです、Eclipe にコピーしないでください。）

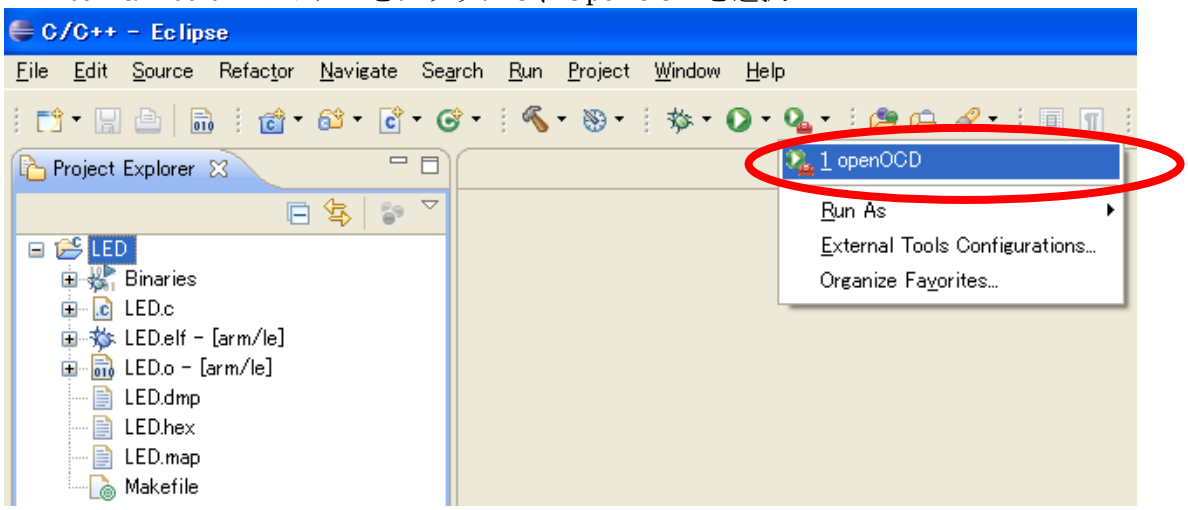
```
target remote localhost:3333 // ローカルポート「3333」と接続（OpenOCDと接続）
monitor halt//ボードの実行を中断させる
monitor step//ステップで実行するように
load // leds_elfをロード，「elf」というフォーマットのファイルにアドレスを含まれる
break main // 「main」関数にブレークポイントを設定
continue // プログラムを実行させて、「main」にとまってステップでデバッグ可能
```



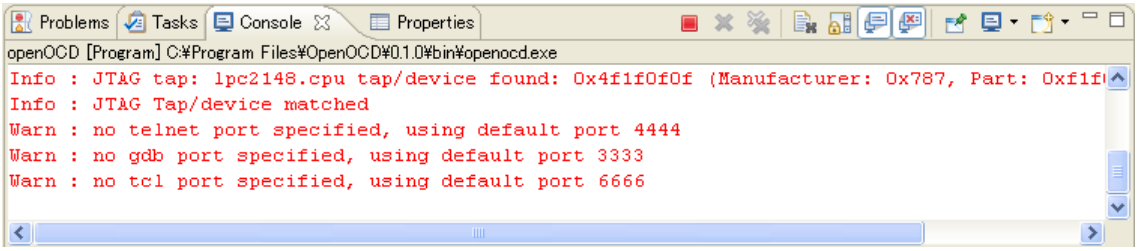
4．デバッグ

a) OpenOCD を起動

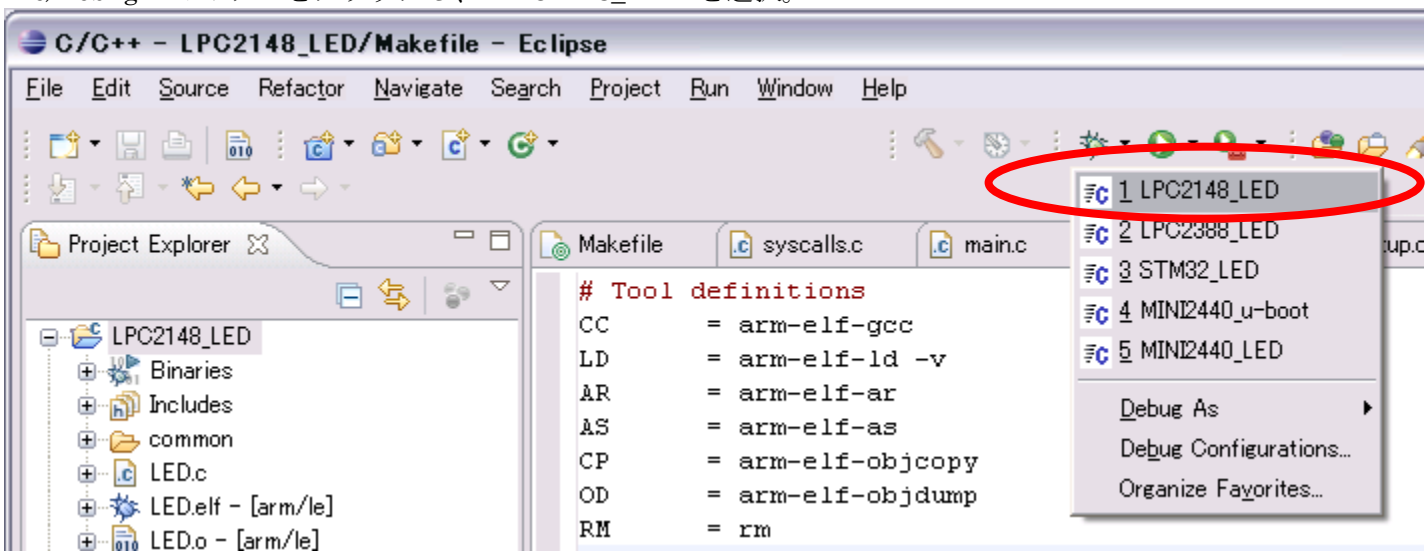
External Tools の▼ボタンをクリックし、OpenOCD を選択



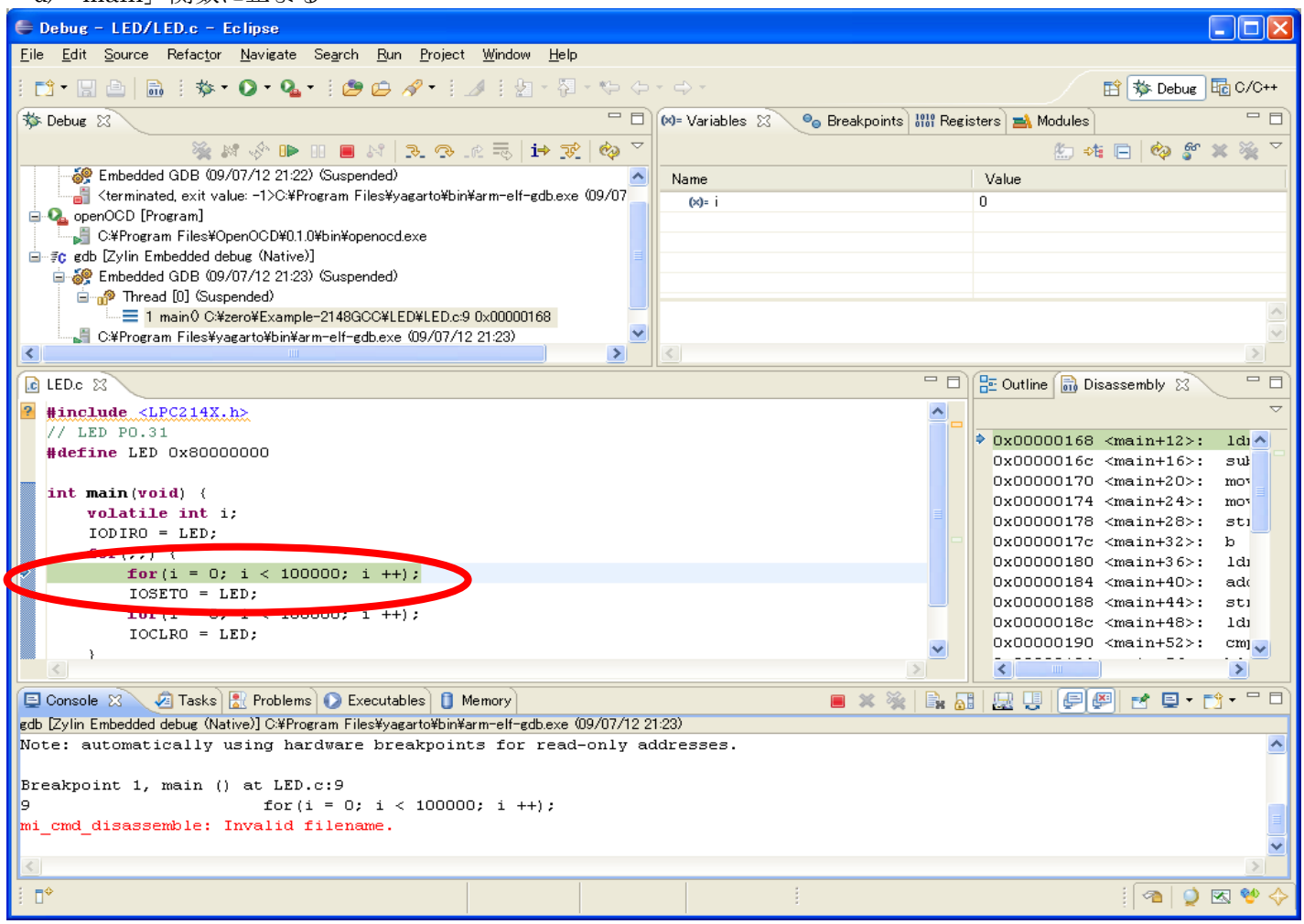
b)Console ウインドに下記のメッセージが出力



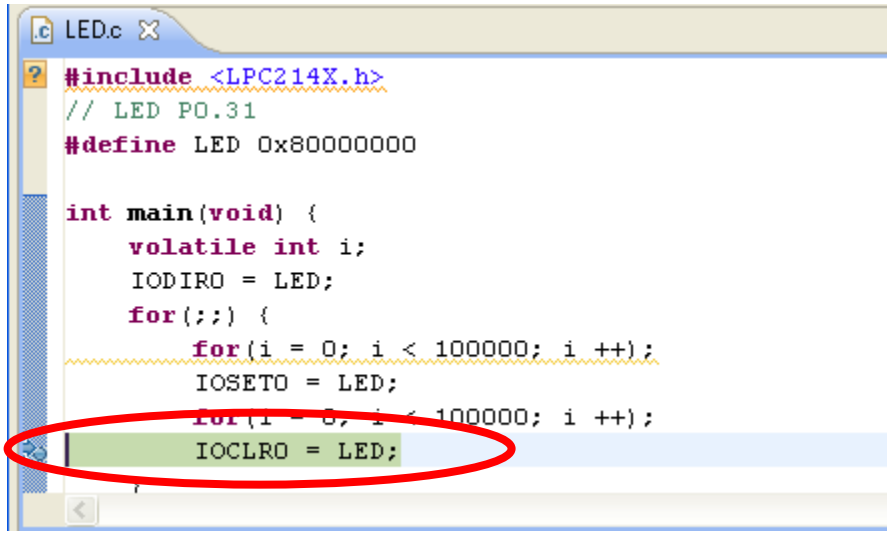
c)Debug の▼ボタンをクリックし、"LPC2148_LED"を選択。



d)「main」関数に止まる



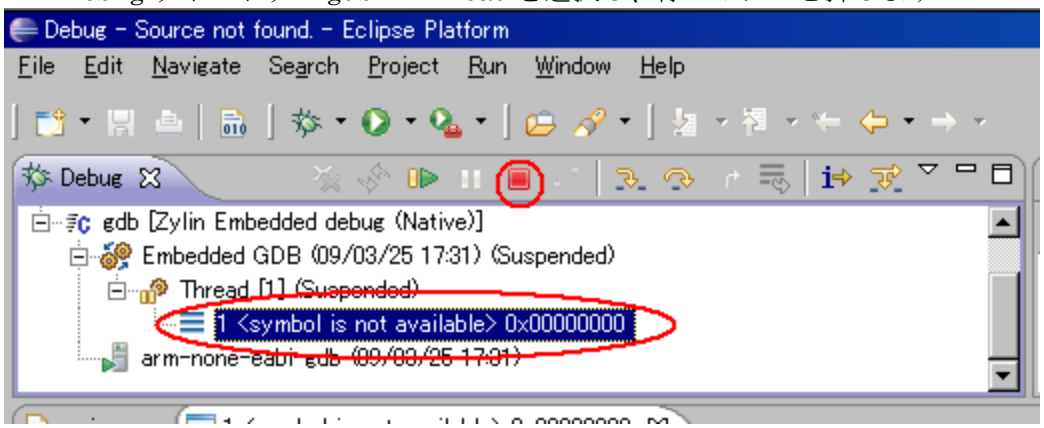
e) ブレークポイントでプログラムが中断した状態から、次のブレークポイントまで実行させたり、1 行ずつ実行させたりできます。コードを繰り返して実行することにより、LED ランプが点滅します。



５．デバッグ終了

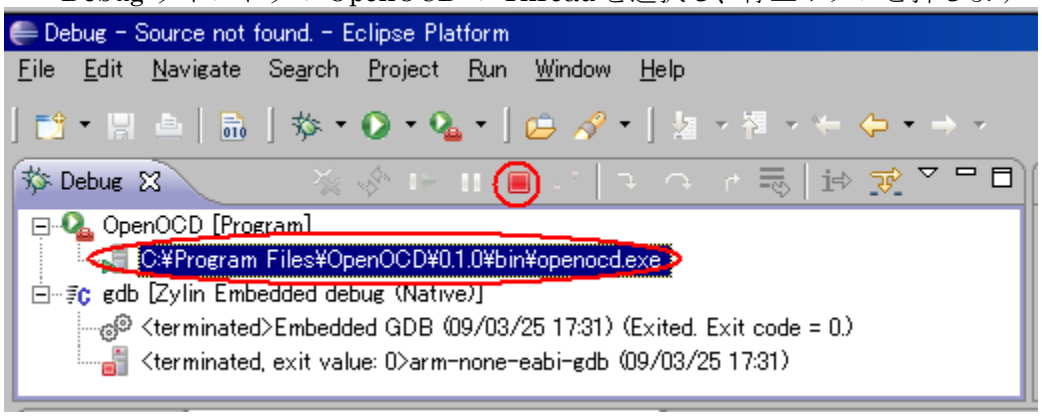
a) gdb の停止

Debug ウインドウの gdb の Thread を選択し、停止ボタンと押します



b) OpenOCD の停止

Debug ウインドウの OpenOCD の Thread を選択し、停止ボタンと押します



c) 電源停止

ターゲットの電源を停止

d) OpenJTAG をターゲットから取り外す

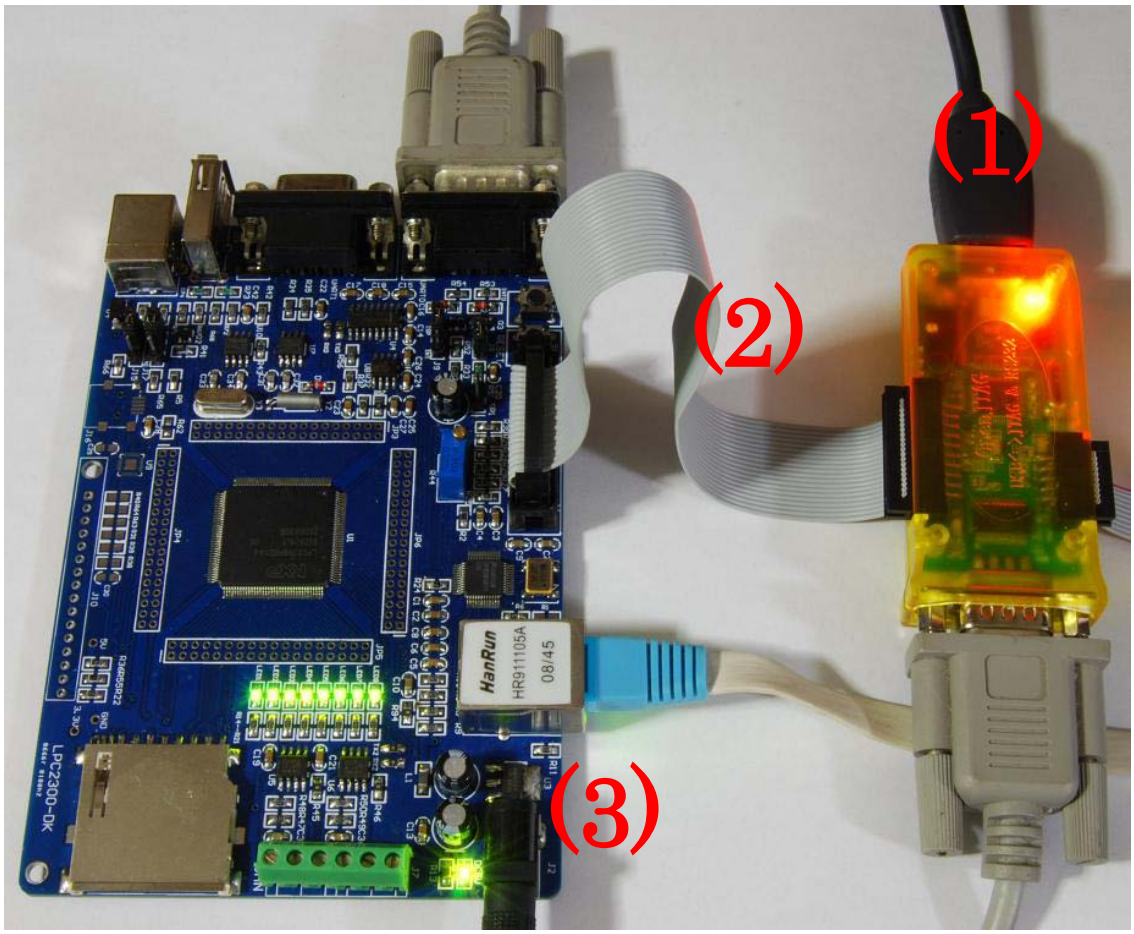
e) 上記が面倒であれば Eclipse を終了しターゲットの電源停止、open-JTAG を取り外しでも OK です。

6.2 ARM7 の LPC2388

6.2.1 LPC2388 ボード購入 URL

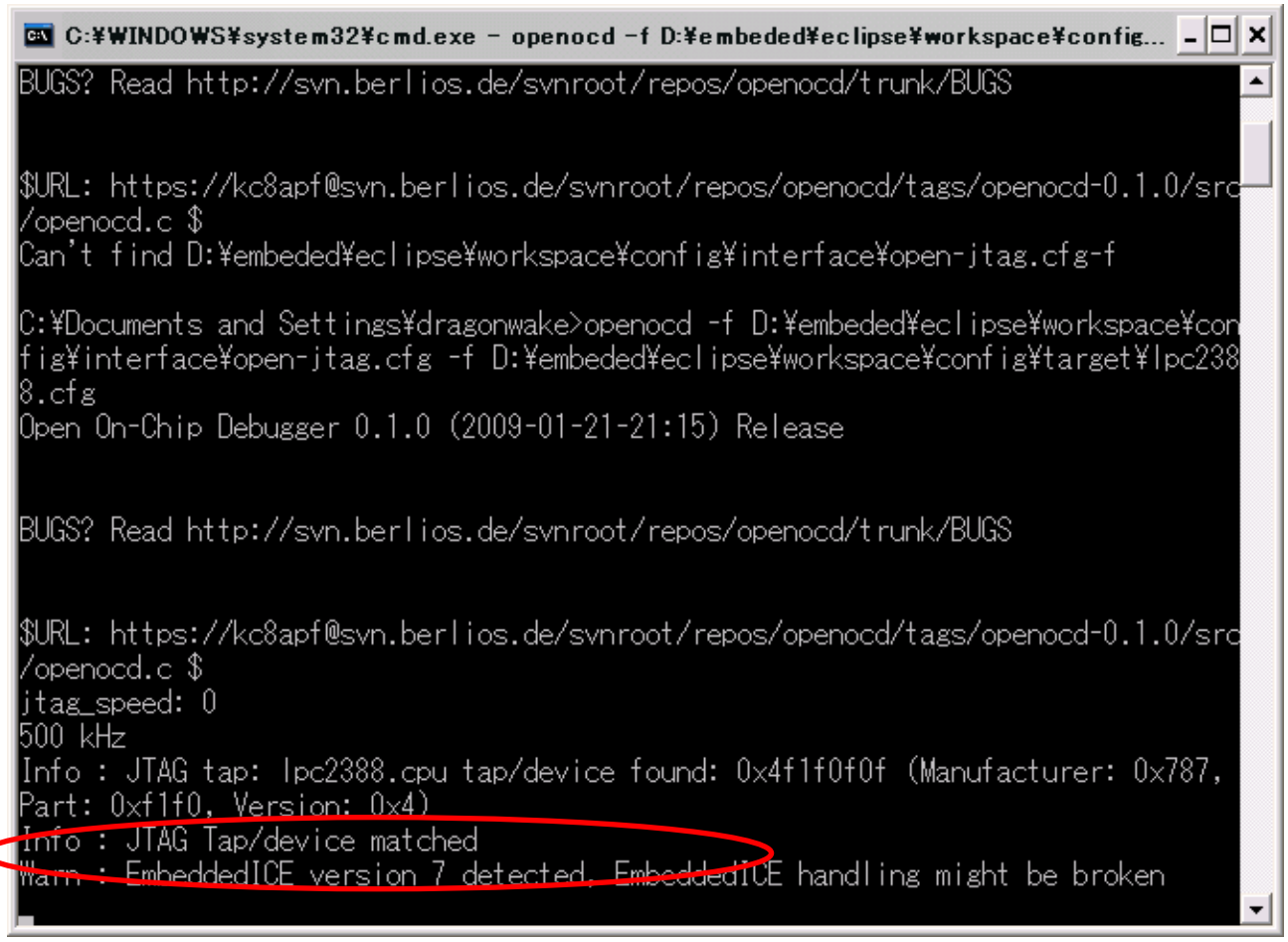
http://www.csun.co.jp/SHOP/200903021.html
ボード URL から LPC2388 マニュアルを参照し、Open-JTAG に接続してください。
LPC2338 マニュアルダウンロード URL :
http://www.dragonwake.com/download/LPC2388/LPC2388_manual.pdf

6.2.2 ハードウエア動作確認



(1). OpenJTAG をパソコンの USB ポートに挿入する
(2). JTAG ケーブルで OpenJTAG と LPC2388 ボードを繋ぐ
(3). LPC2388 ボードに電源を入れる
(4). 下記のコマンドを入力します。
openocd -f " D:¥embeded¥eclipse¥workspace¥config¥interface¥open-jtag.cfg" -f " D:¥embeded¥eclipse¥workspace¥config¥target¥lpc2388.cfg"

※ はじめの-f は OpenJTAG のコンフィグファイルを使います。二番目の-f は lpc2388 のコンフィグファイルを使います。



画面のように"Info：JTAG Tap/device matched"と表示されればＯＫです
（この時点で ARM LPC2388 と通信が出来ています）

(6). その後 CTRL+C を押してデバッグを中止します。

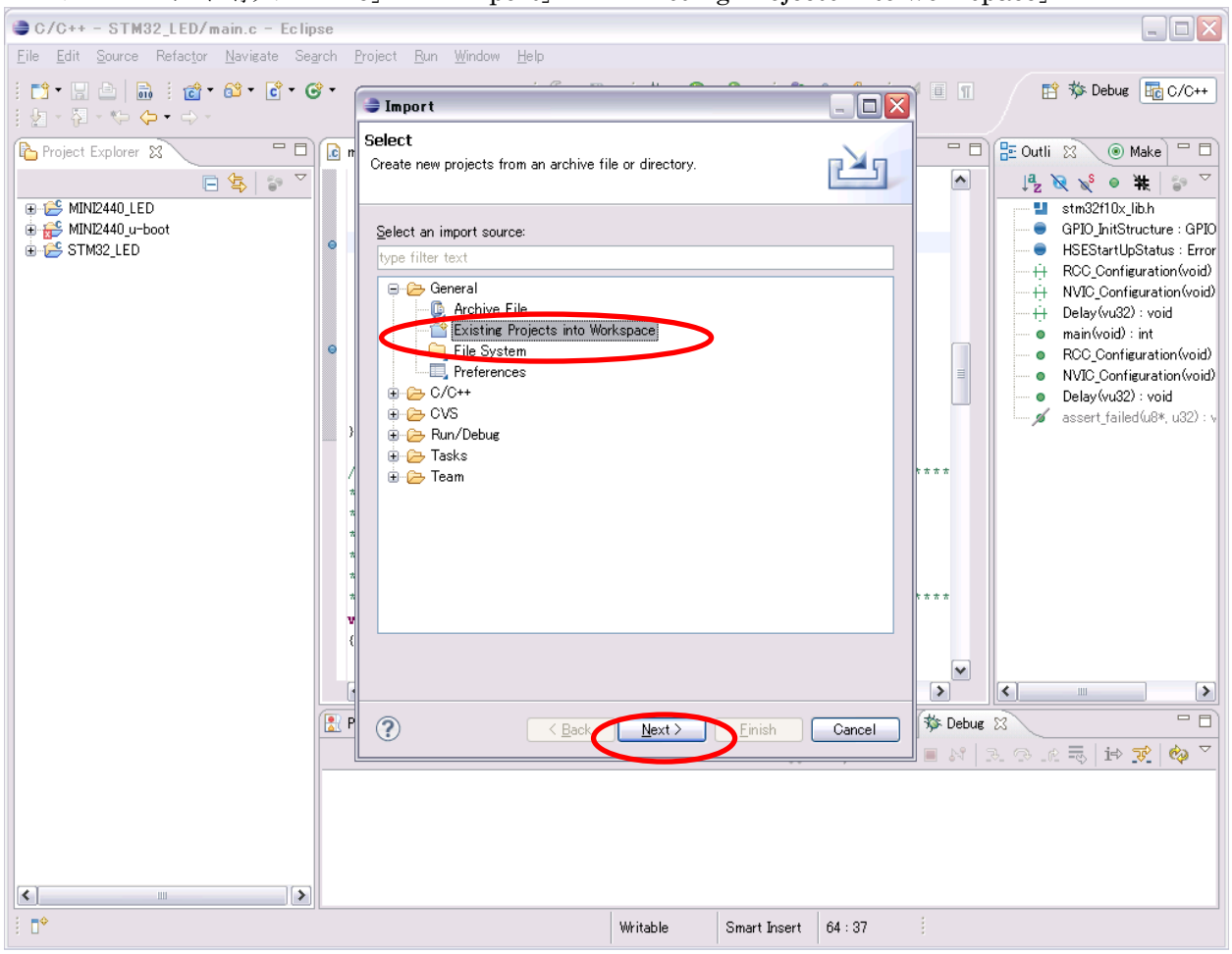
6.2.3 LPC2388 用のサンプル「LED」をデバッグ

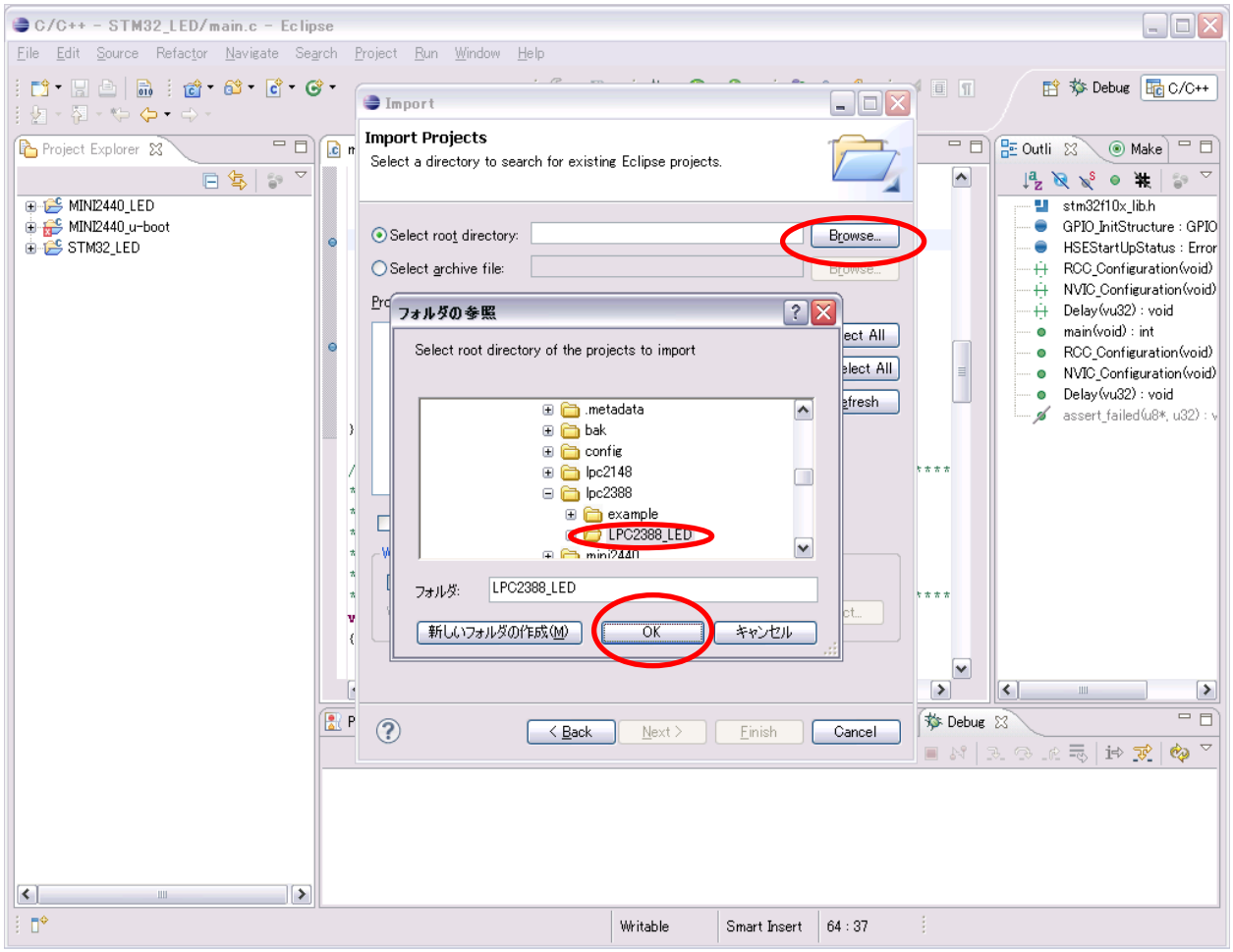
*サンプルの導入について、二つ方法があります、一つは新規プロジェクトを作成してからソースを導入、もう一つはダウンロードプロジェクトをインポートします。

＊新規プロジェクトの作成方法は「6.1.3　LPC2148 用のサンプル「LED」をデバッグ」を参照

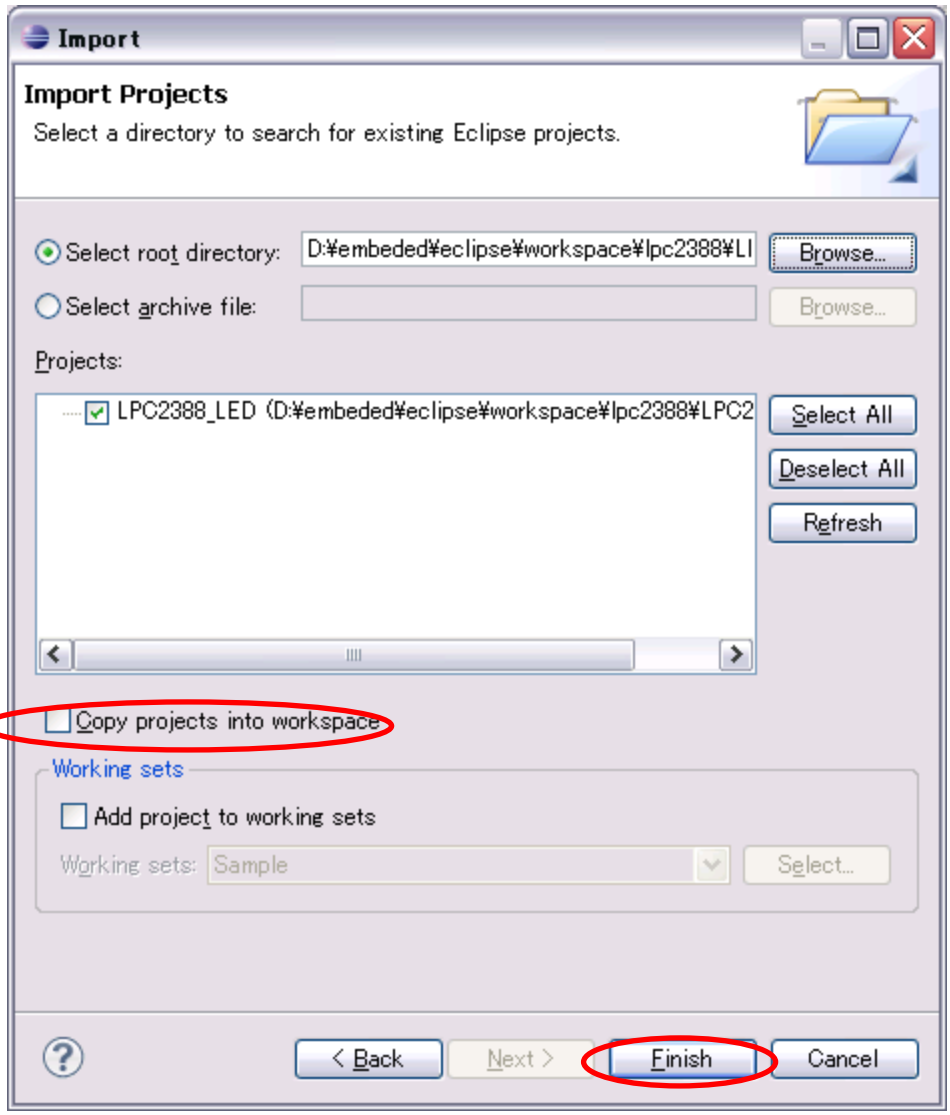
＊ここに既存プロジェクトのインポートとして説明

1．プロジェクト導入：「File」→「Import」→「Existing Projects into workspace」

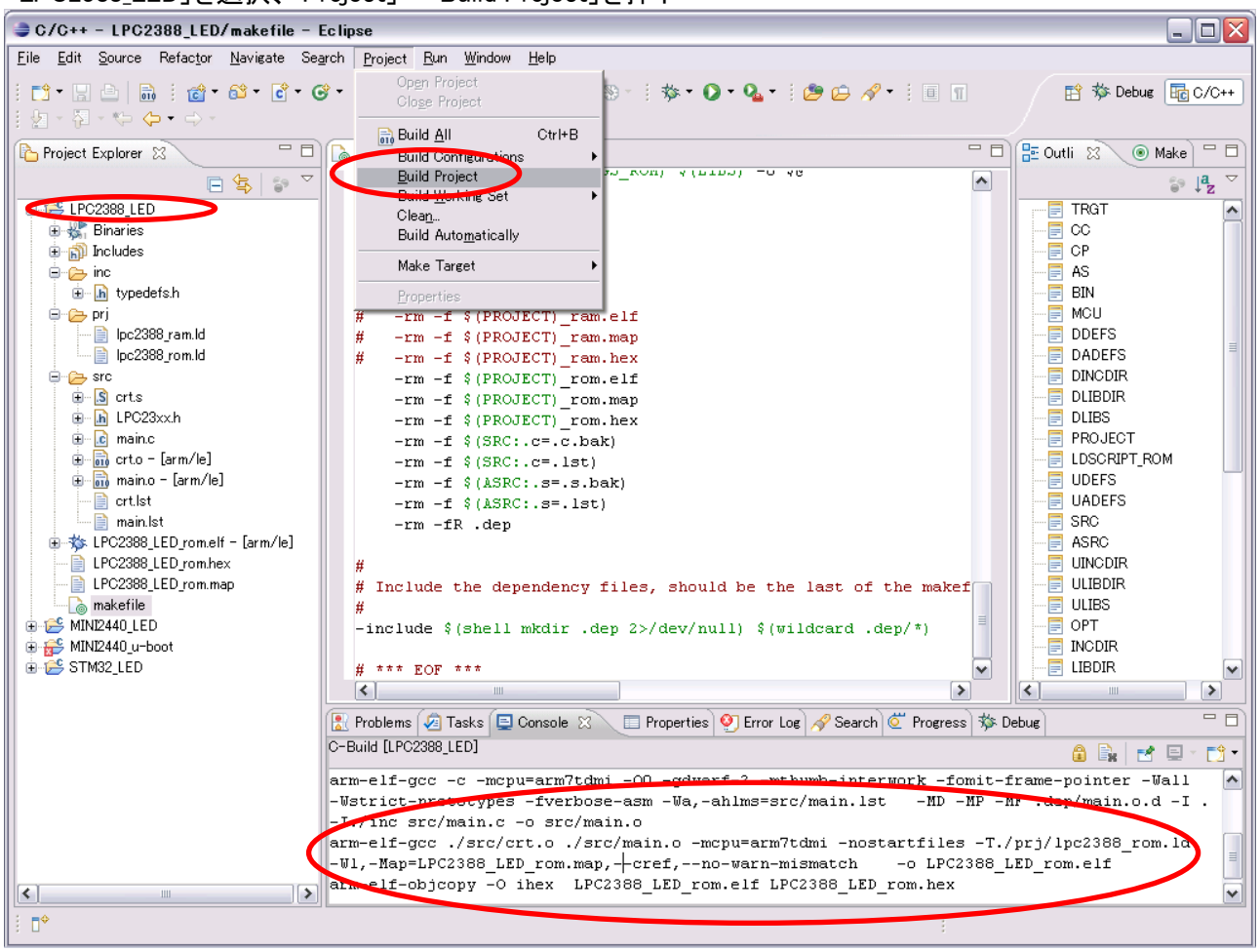



C/C++ - STM32_LED/main.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug C/C++
Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
Browse...
Select archive file:
Project Explorer
MINI2440_LED
MINI2440_u-boot
STM32_LED
Outli
Make
stm32f10x_lib.h
GPIO_InitStructure : GPIO
HSEStartUpStatus : Error
RCC_Configuration(void)
NVIC_Configuration(void)
Delay(vu32) : void
main(void) : int
RCC_Configuration(void)
NVIC_Configuration(void)
Delay(vu32) : void
フォルダの参照
Select root directory of the projects to import
.metadata
bak
config
lpc2148
lpc2388
example
LPC2388_LED
フォルダ:
LPC2388_LED
新しいフォルダの作成(M)
OK
キャンセル
< Back
Next >
Finish
Cancel
Debug
Writable
Smart Insert
64 : 37

既にダウンロードしたプロジェクトを Workspace に入れていますので、ここに「Copy projects into workspace」を外します。Workspace に入れていない場合、「Copy projects into workspace」をチェックしてください。



2. 導入済みのプロジェクトをコンパイル
「LPC2388_LED」を選択、「Project」→「Build Project」を押下



コンパイルが成功すれば、実行ファイルLPC2388_LED_rom.elfとLPC2388_LED_rom.hexを生成されます。

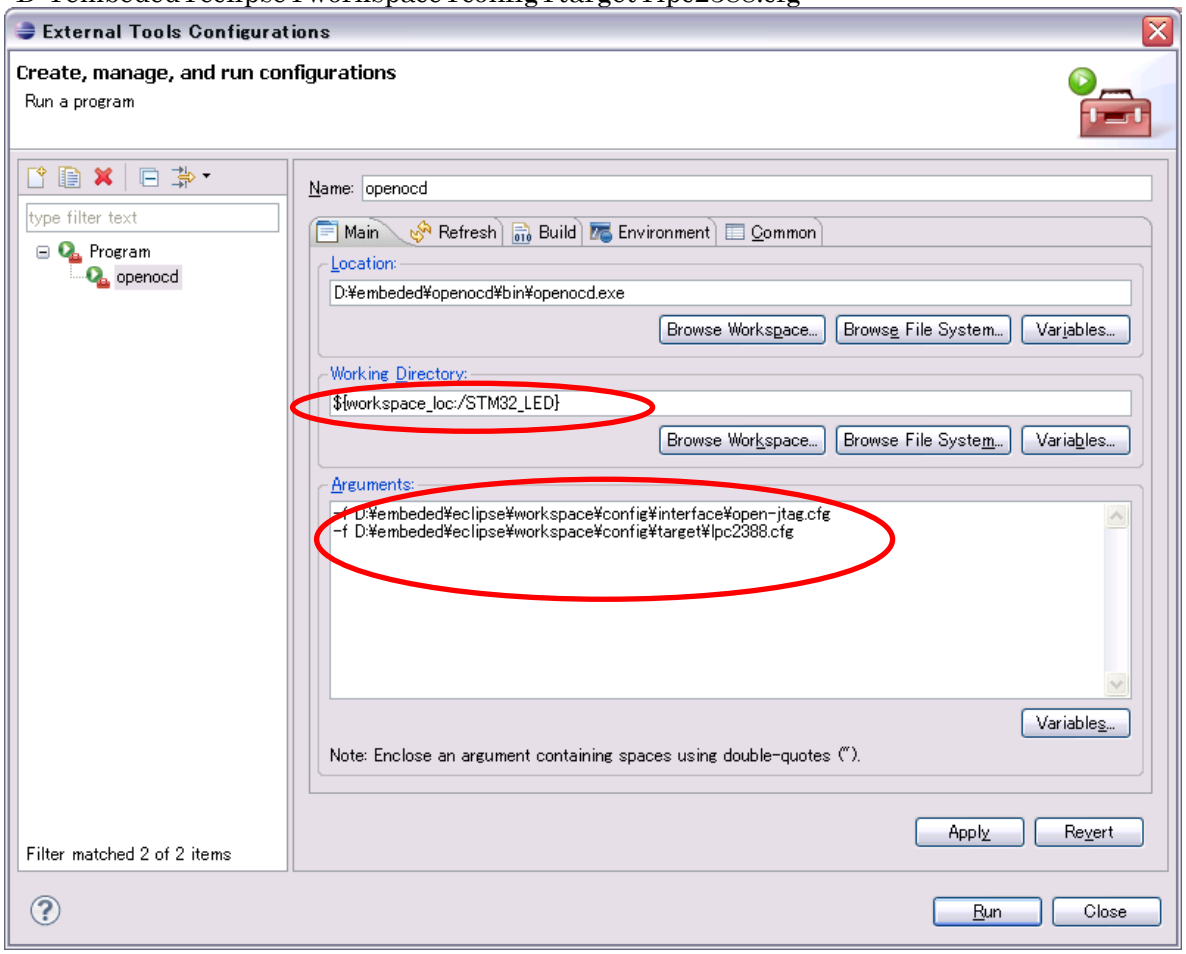
３．openocd 設定の修正

一般の設定は「5.4 OpenOCD の設定」を参照

a)「Working Directory」を「LPC2388_LED」に変更

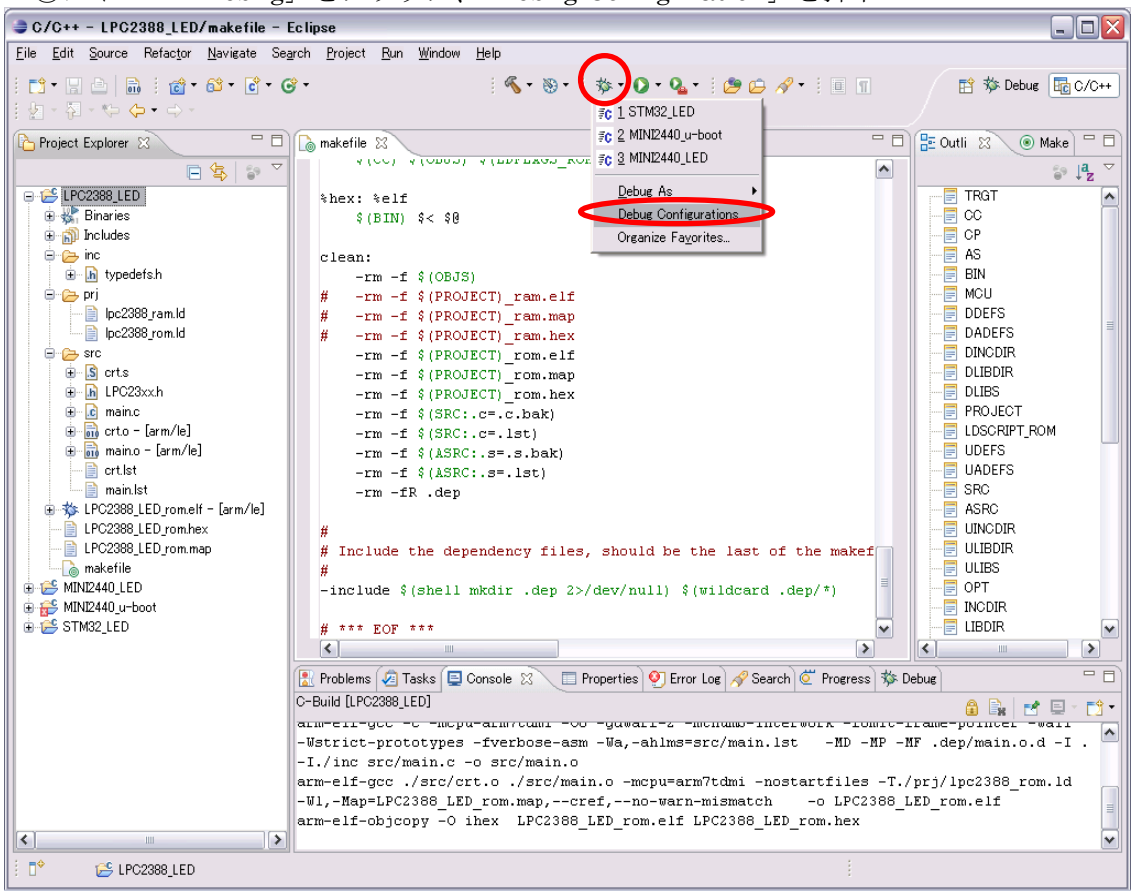
b)「Arguments」を下記のように変更

-f "D:¥embeded¥eclipse¥workspace¥config¥interface¥open-jtag.cfg" -f "D:¥embeded¥eclipse¥workspace¥config¥target¥lpc2388.cfg"

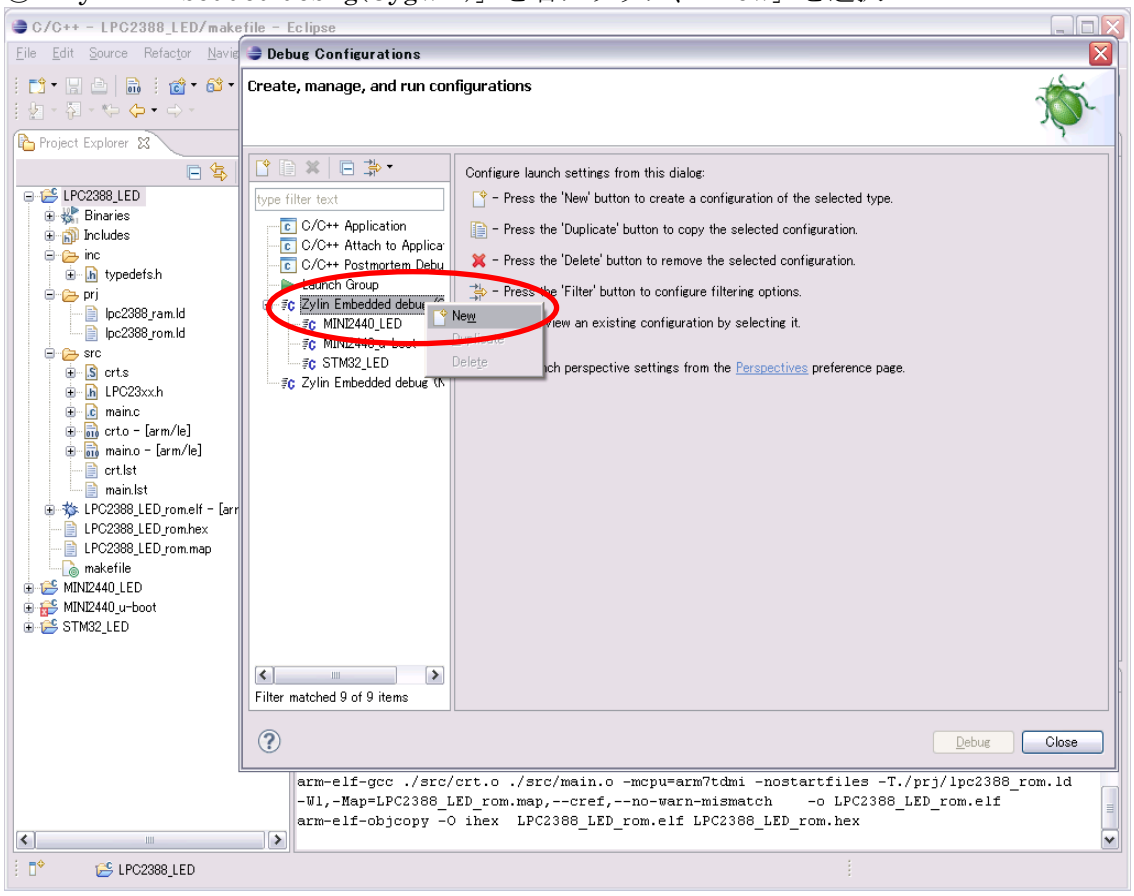


４．LPC2388_LED 用の GDB を追加

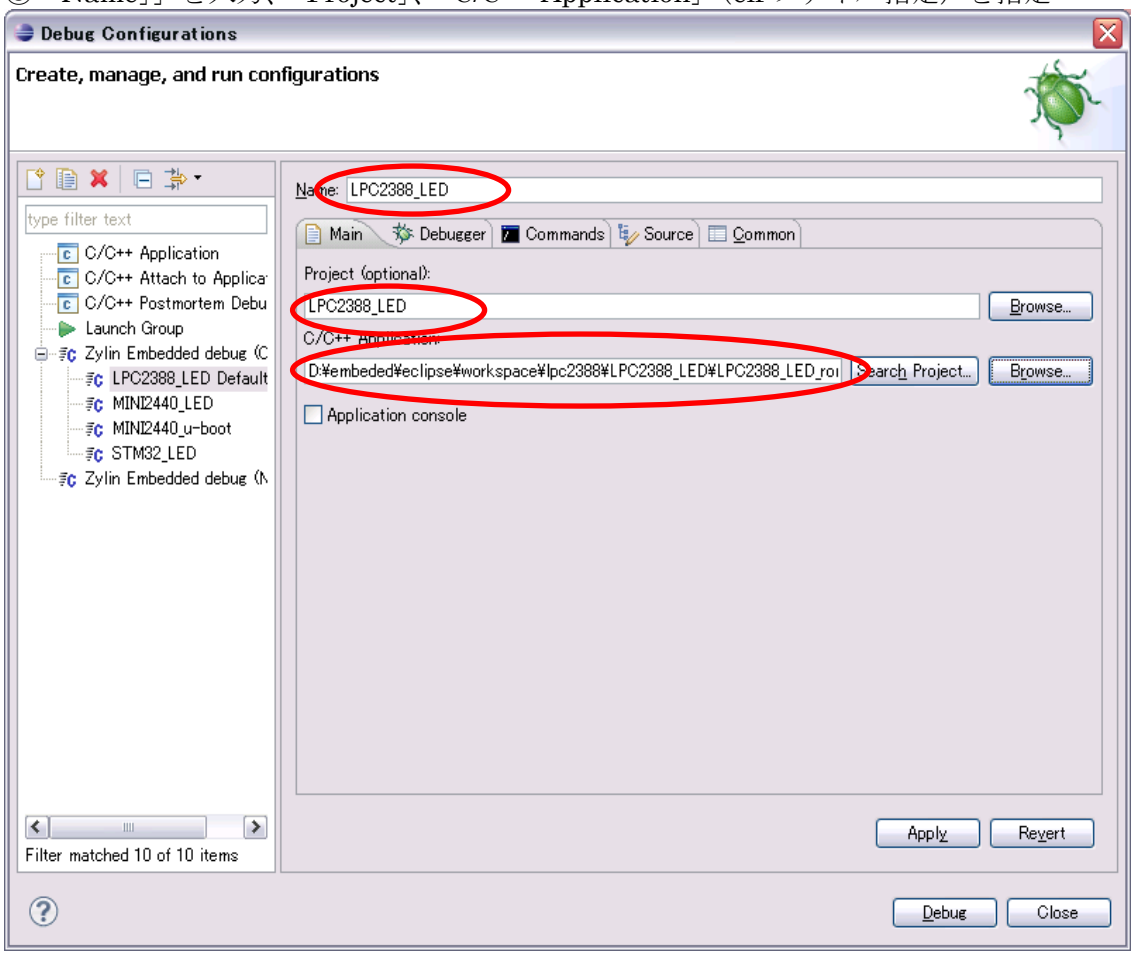
①アイコン「Debug」をクリック、「Debug Configuration」を押下



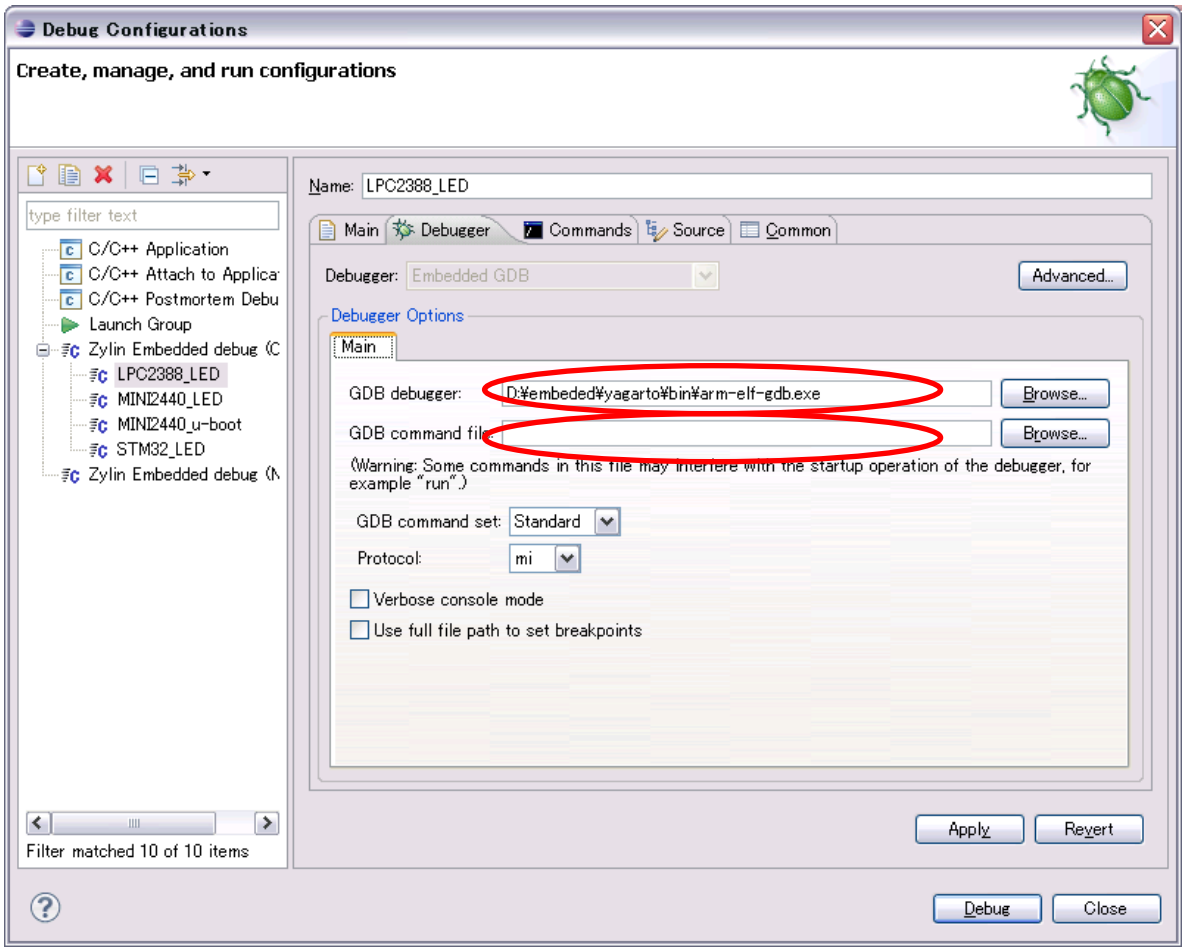
②「Zylin Embedded debug(Cygwin)」を右クリック、「New」を選択



③「Name」」を入力、「Project」、「C/C++ Application」（elf ファイル指定）を指定



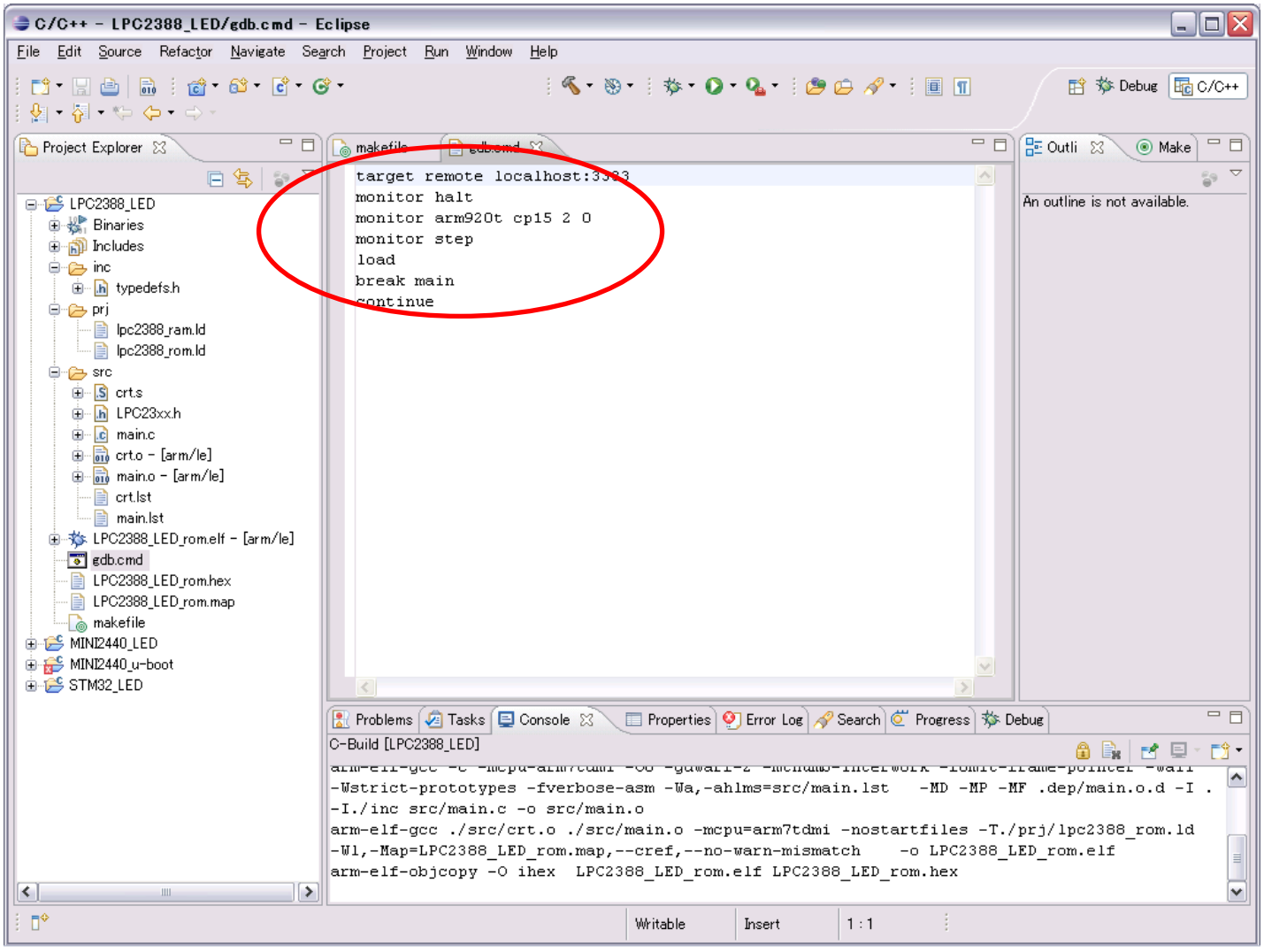
④「GDB debugger」を「arm -elf-gdb.exe」の場所に指定



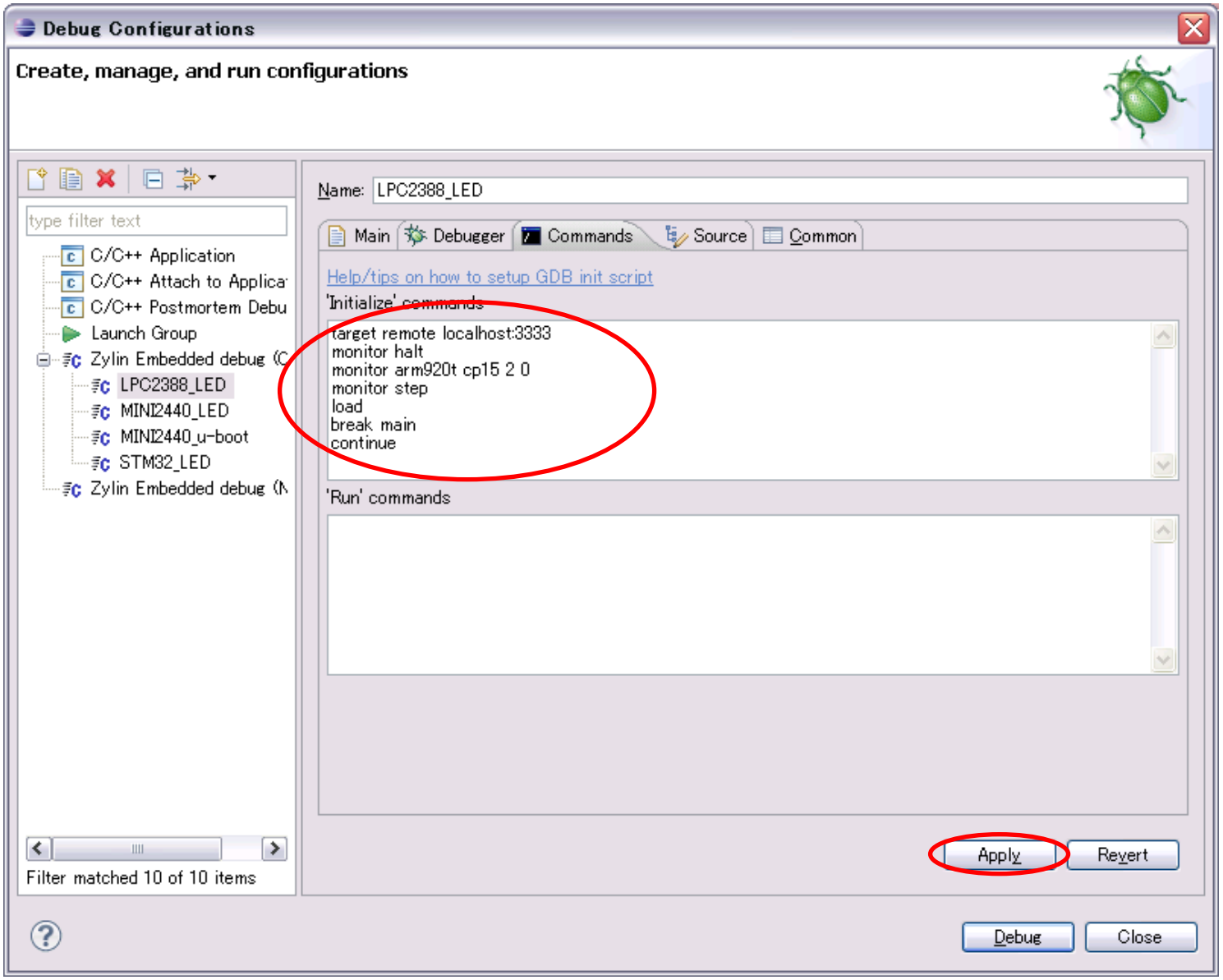
⑤「Commands」を入力

target remote localhost:3333 // ローカルポート「3333」と接続（OpenOCDと接続）
monitor halt//ボードの実行を中断させる
monitor step//ステップで実行するように
load // leds_elfをロード，「elf」というフォーマットのファイルにアドレスを含まれる
break main // 「main」関数にブレークポイントを設定
continue // プログラムを実行させて、「main」にとまってステップでデバッグ可能

「LPC2388_LED」プロジェクトの配下、「gdb.cmd」ファイルの内容をそのままコピーしても OK
「gdb.cmd」を右クリック、「Open With」→「Text Editor」
＊ほかの設定を保存するため、コピー途中で「Apply」ボタンを押して適用します。

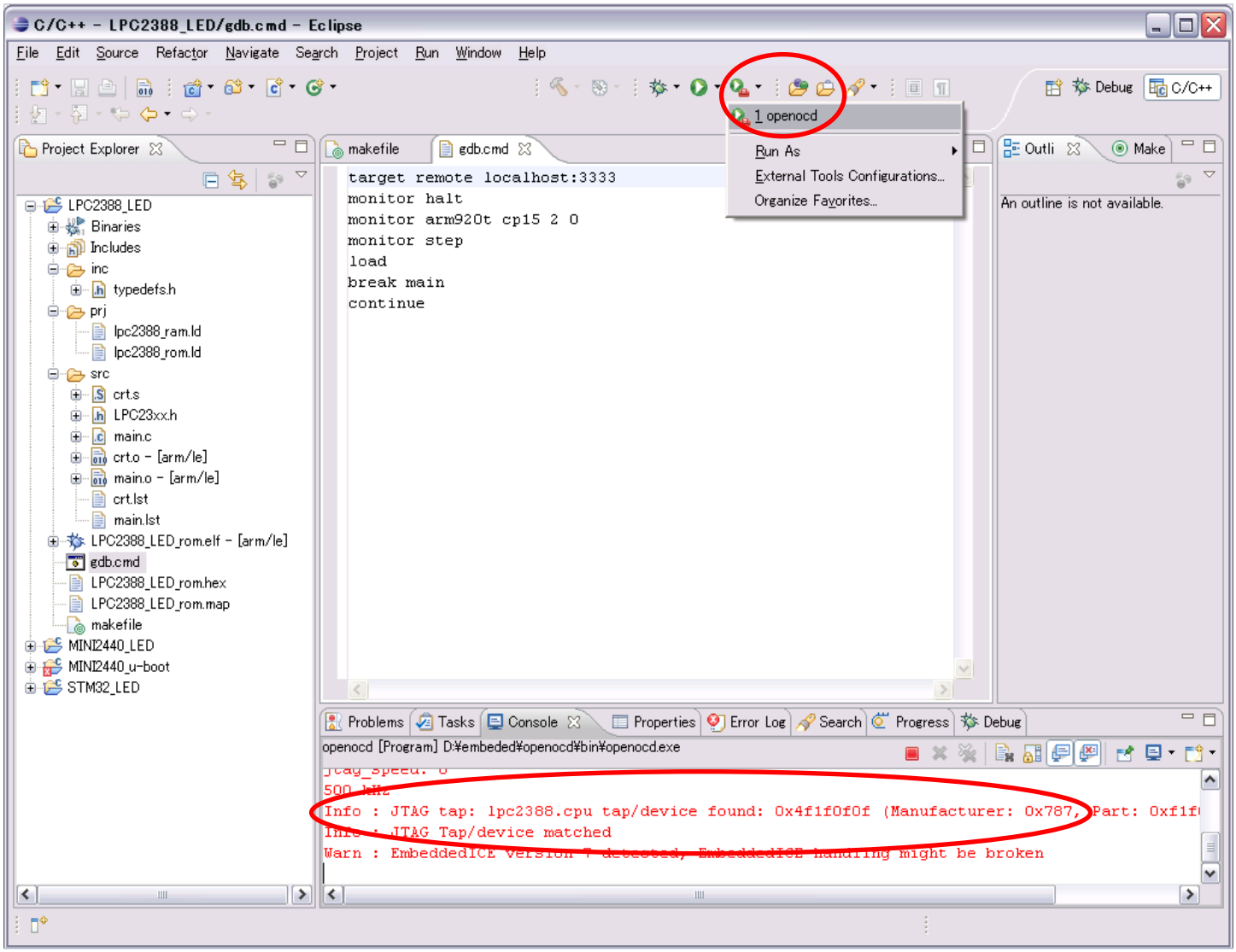


上記のコピー内容を下記「Commands」に貼り付け、「Apply」ボタンを押下



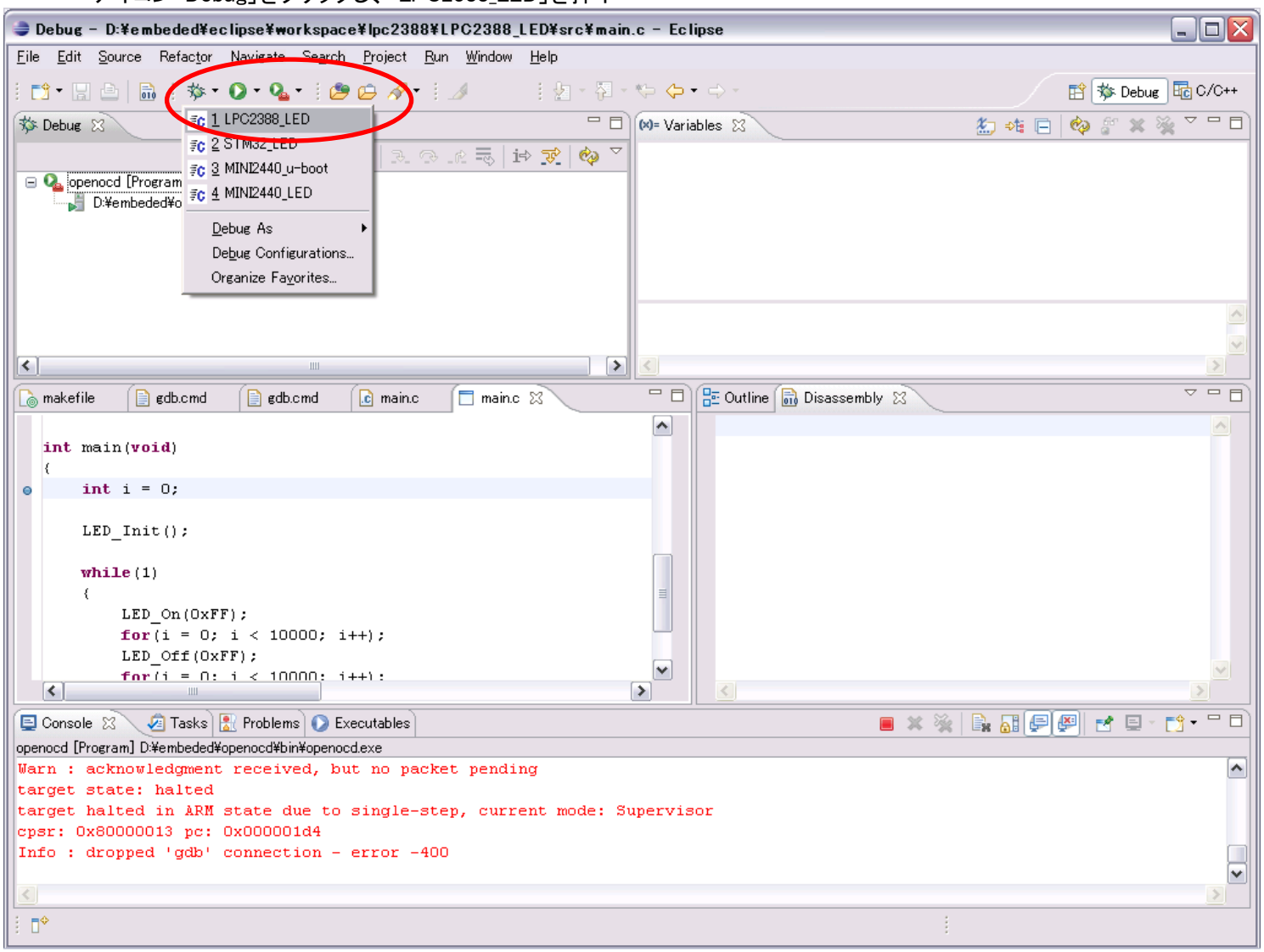
５．デバッグ

① openocd 起動

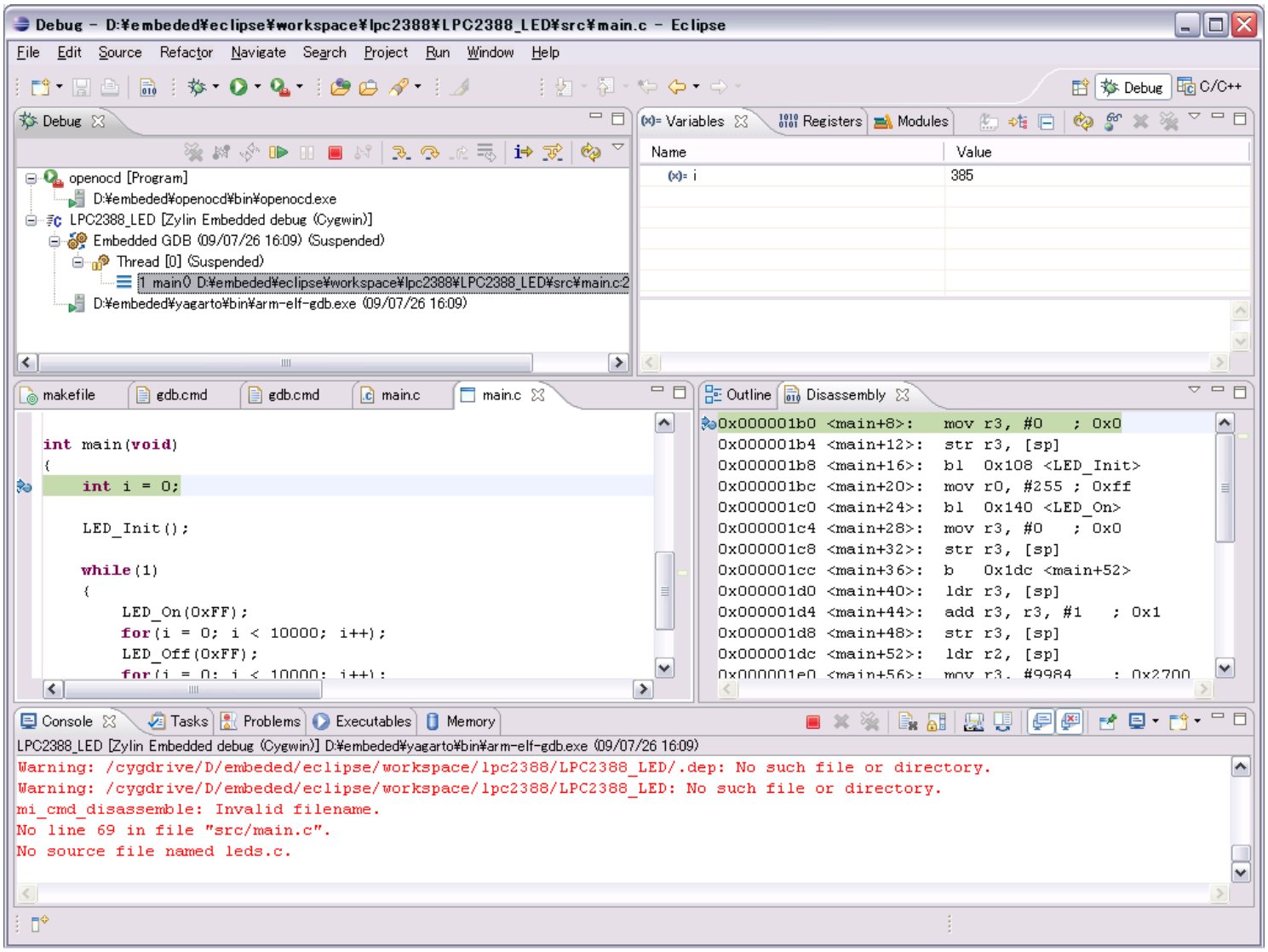


② gdb 起動

アイコン「Debug」をクリックし、「LPC2388_LED」を押下



③ 関数「main」に止まり、ショットカットキー「F6」でステップでデバッグできる



6.3 ARM Cotex-M3 の STM32F103

6.3.1 STM32F103 ボード購入 URL

http://www.csun.co.jp/SHOP/200903019.html

ボード URL から STM32F103 マニュアルを参照し、Open-JTAG に接続してください。

ボード URL から LPC2388 マニュアルを参照し、Open-JTAG に接続してください。

STM32F103 ボードマニュアルダウンロード URL：

http://www.dragonwake.com/download/download-stm32/STM32_manual.pdf

6.3.2 ハードウエア動作確認

(1). OpenJTAG をパソコンの USB ポートに挿入する

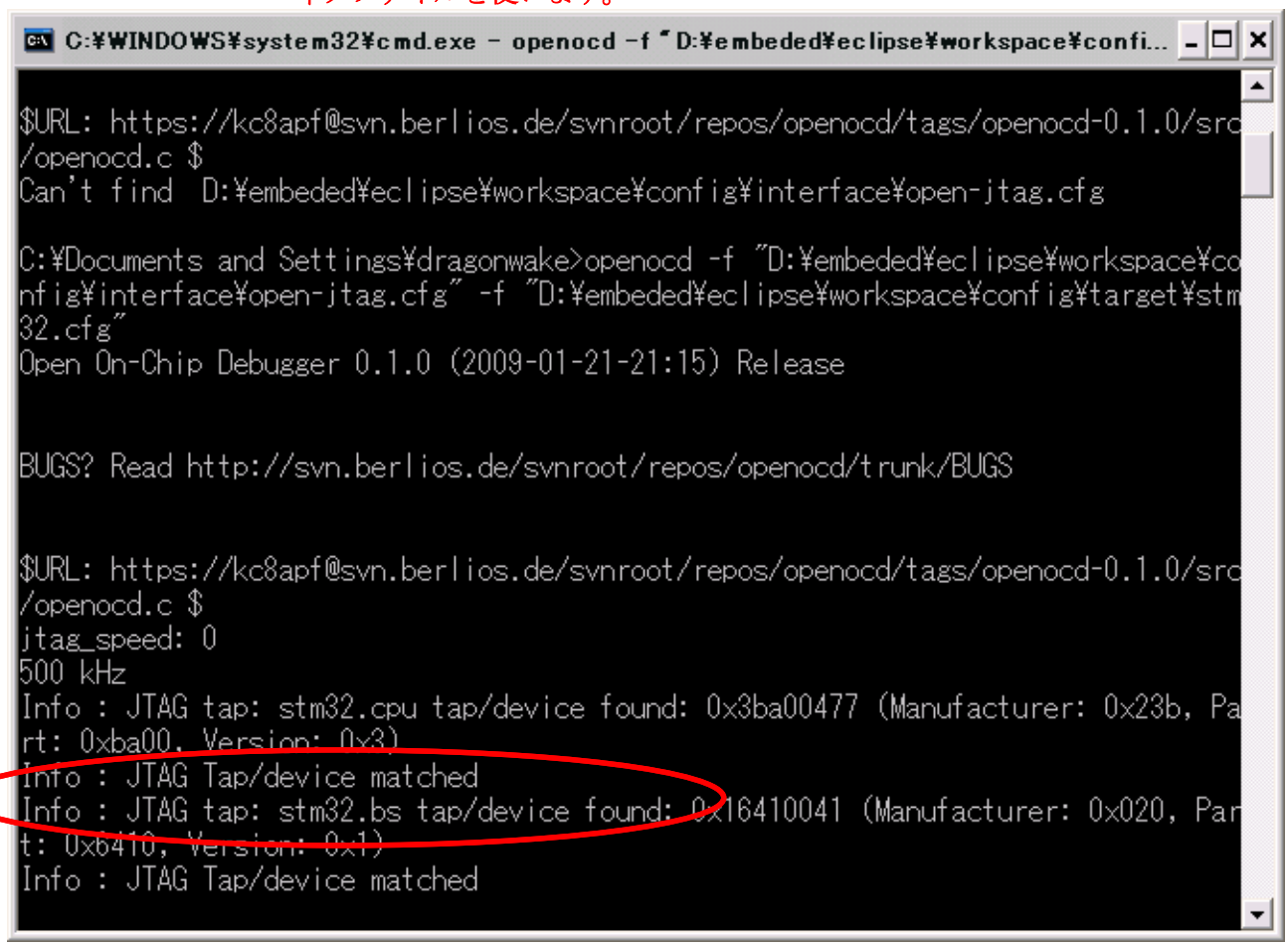
(2). JTAG ケーブルで OpenJTAG と STM32 ボードを繋ぐ

(3). STM32 ボードに電源を入れる

(4). 下記のコマンドを入力します。

```
openocd -f "D:\embeded\eclipse\workspace\config\interface\open-jtag.cfg" -f "D:\embeded\eclipse\workspace\config\target\stm32.cfg"
```

※ はじめの-f は OpenJTAG のコンフィグファイルを使います。二番目の-f は stm32 のコンフィグファイルを使います。



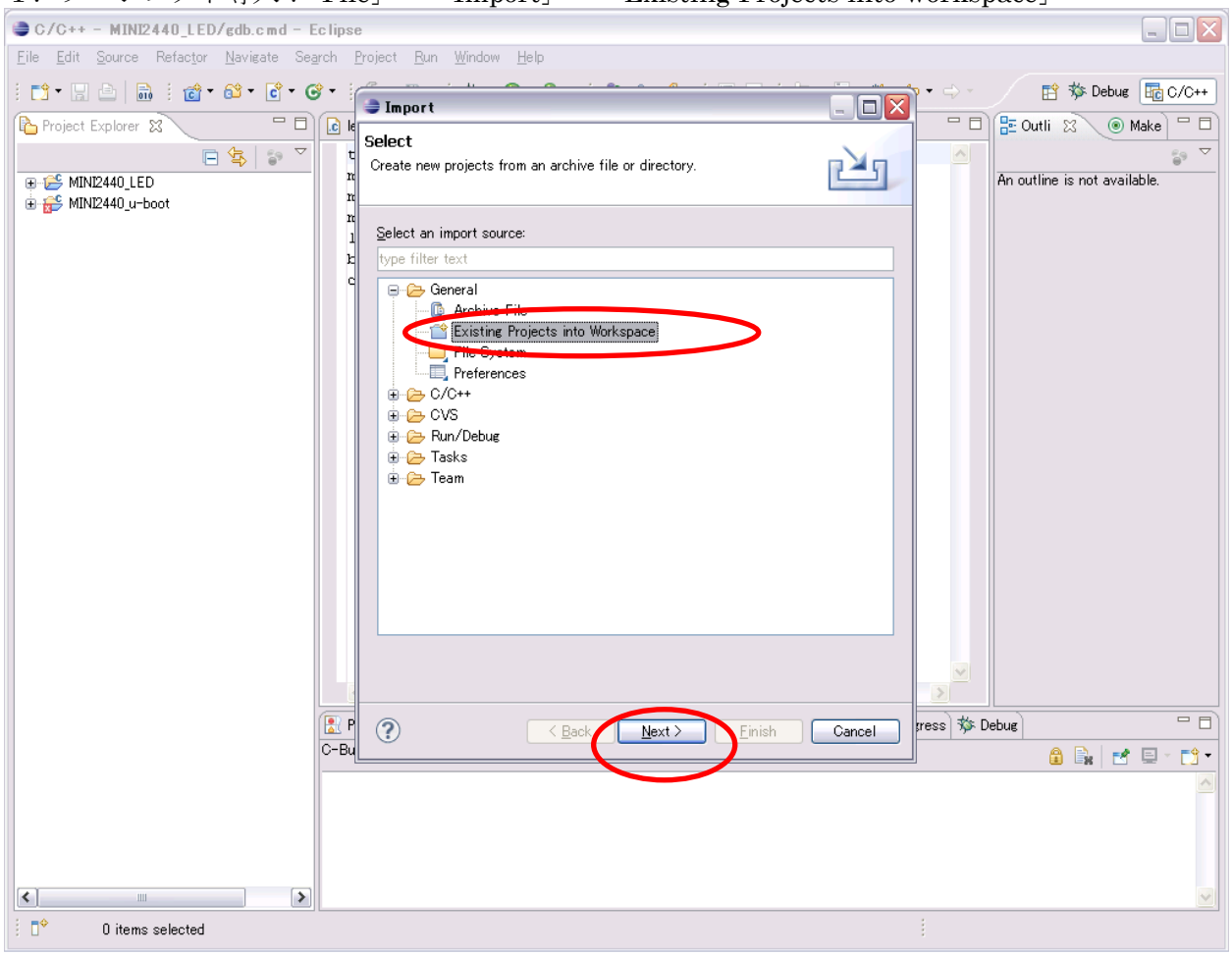
6.3.3 LED サンプルデバッグ

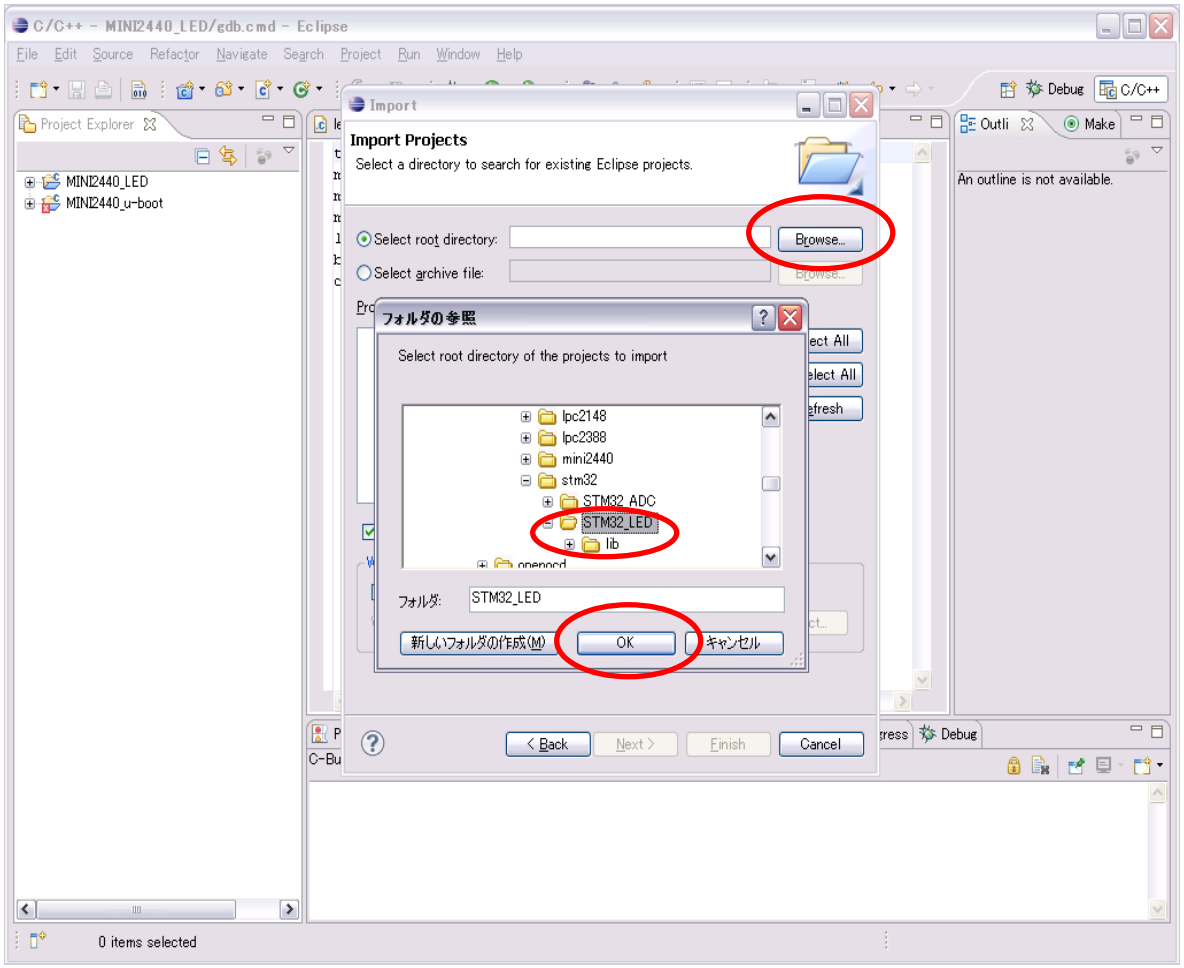
*サンプルの導入について、二つ方法があります、一つは新規プロジェクトを作成してからソースを導入、もう一つはダウンロードプロジェクトをインポートします。

＊新規プロジェクトの作成方法は「**6.1.3 LPC2148 用のサンプル「LED」をデバッグ**」を参照

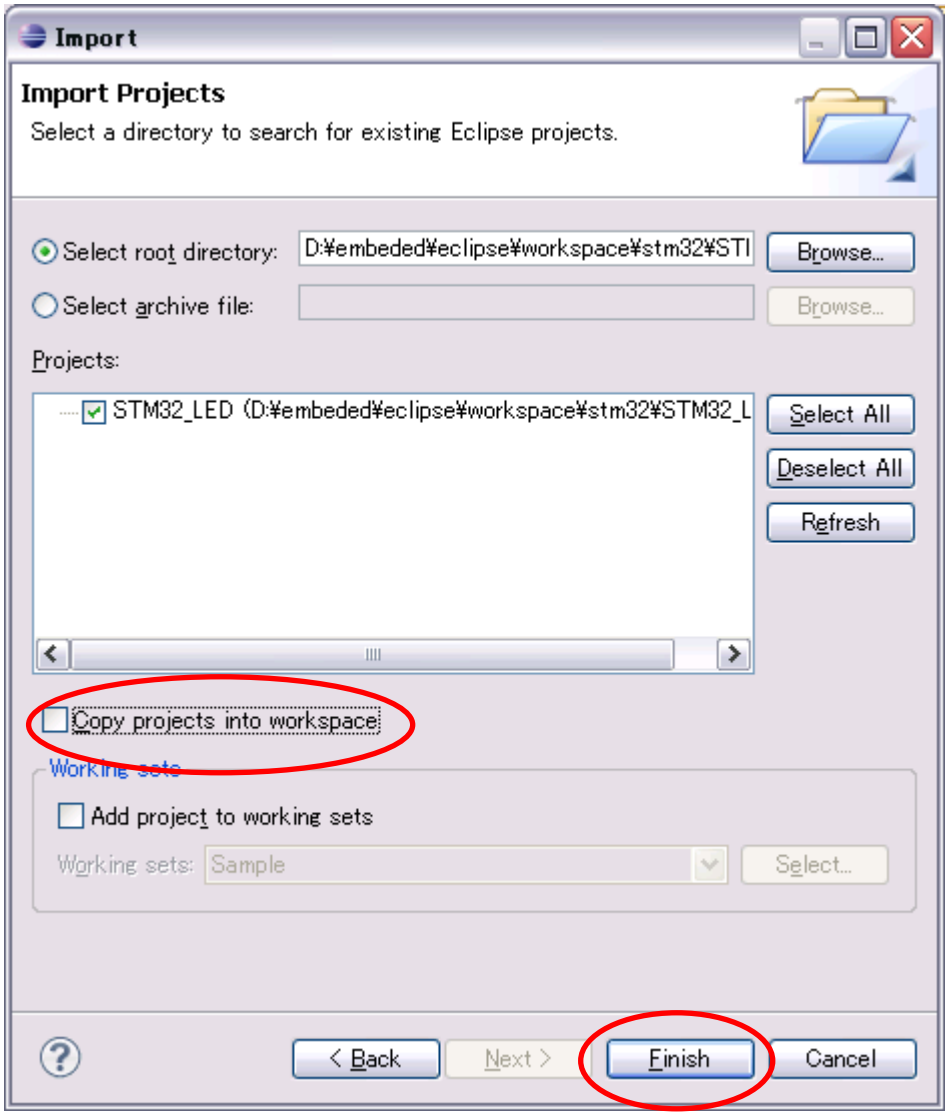
＊ここに既存プロジェクトのインポートとして説明

1．プロジェクト導入：「File」→「Import」→「Existing Projects into workspace」

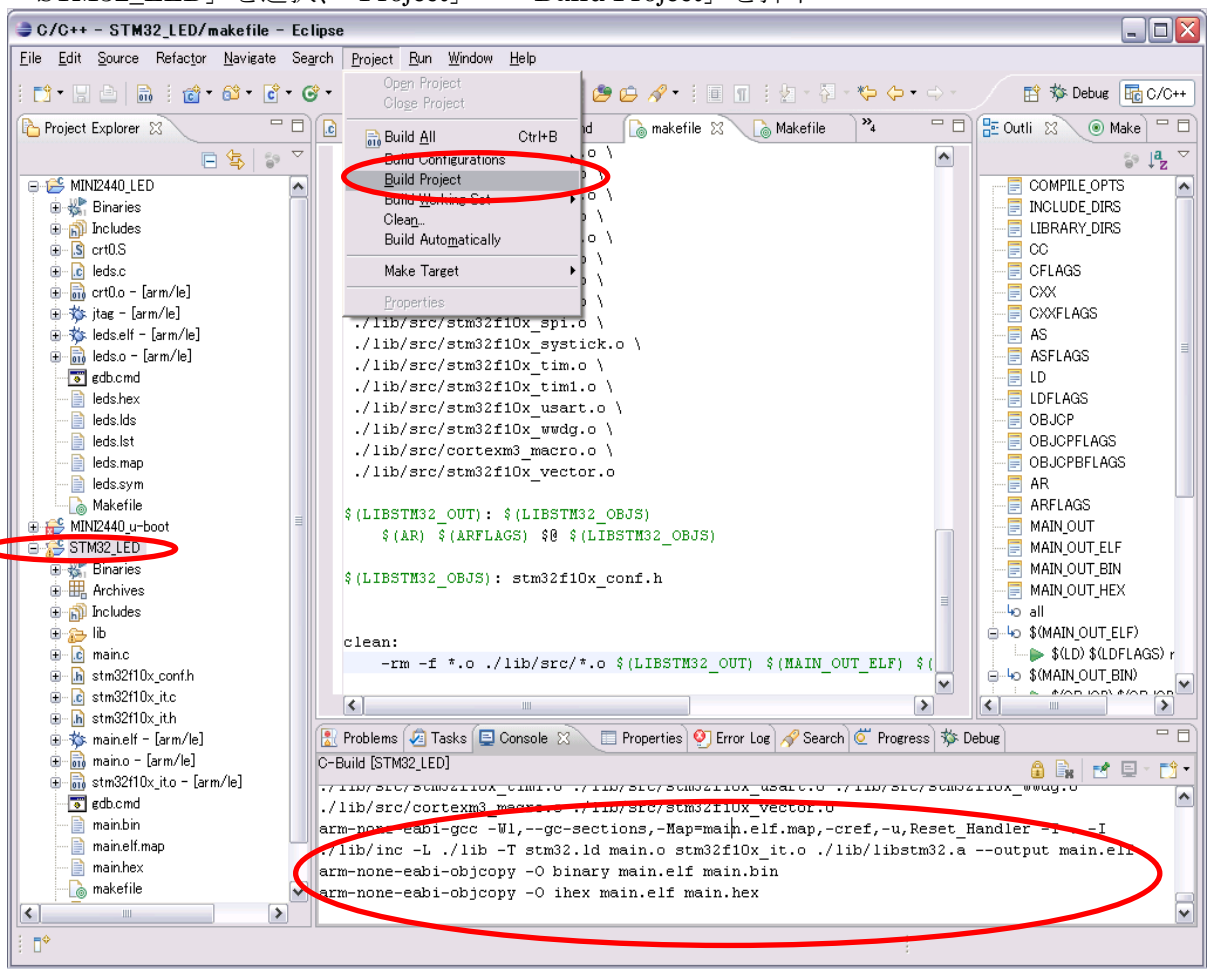



C/C++ - MINI2440_LED/gdb.cmd - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
C/C++
Project Explorer
MINI2440_LED
MINI2440_u-boot
Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
Browse...
Select archive file:
Browse...
フォルダの参照
Select root directory of the projects to import
lpc2148
lpc2388
mini2440
stm32
STM32_ADC
STM32_LED
lib
フォルダ:
STM32_LED
新しいフォルダの作成(M)
OK
キャンセル
Outli
Make
An outline is not available.
< Back
Next >
Finish
Cancel
Debug
0 items selected

既にダウンロードしたプロジェクトを Wokspace に入れていますので、ここに「Copy projects into workspace」を外します。



２．導入済みのプロジェクトをコンパイル
「STM32_LED」を選択、「Project」→「Build Project」を押下



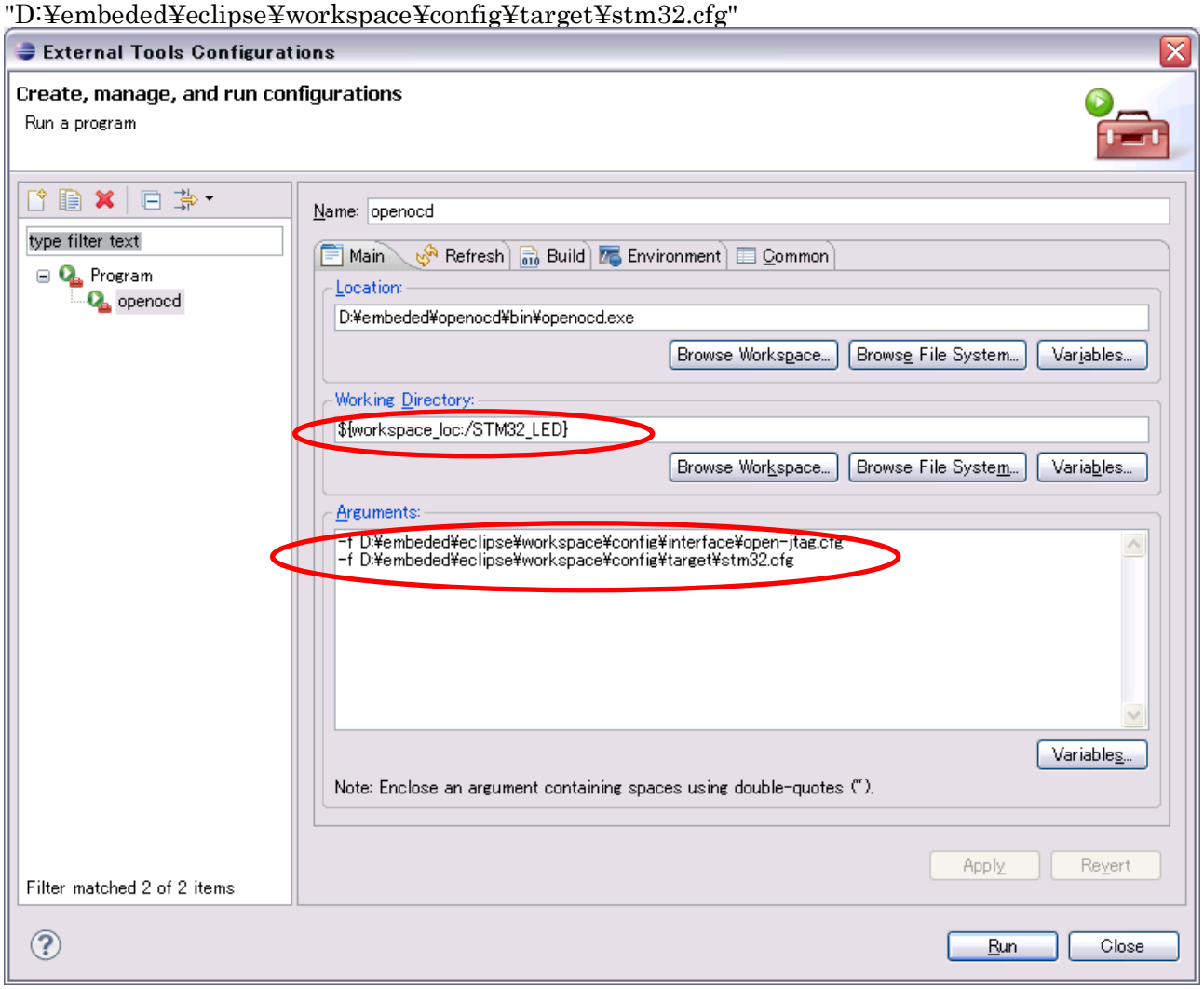
３．openocd 設定の修正

一般の設定は「5.4 OpenOCD の設定」を参照

a)「Working Directory」を「STM32_LED」に変更

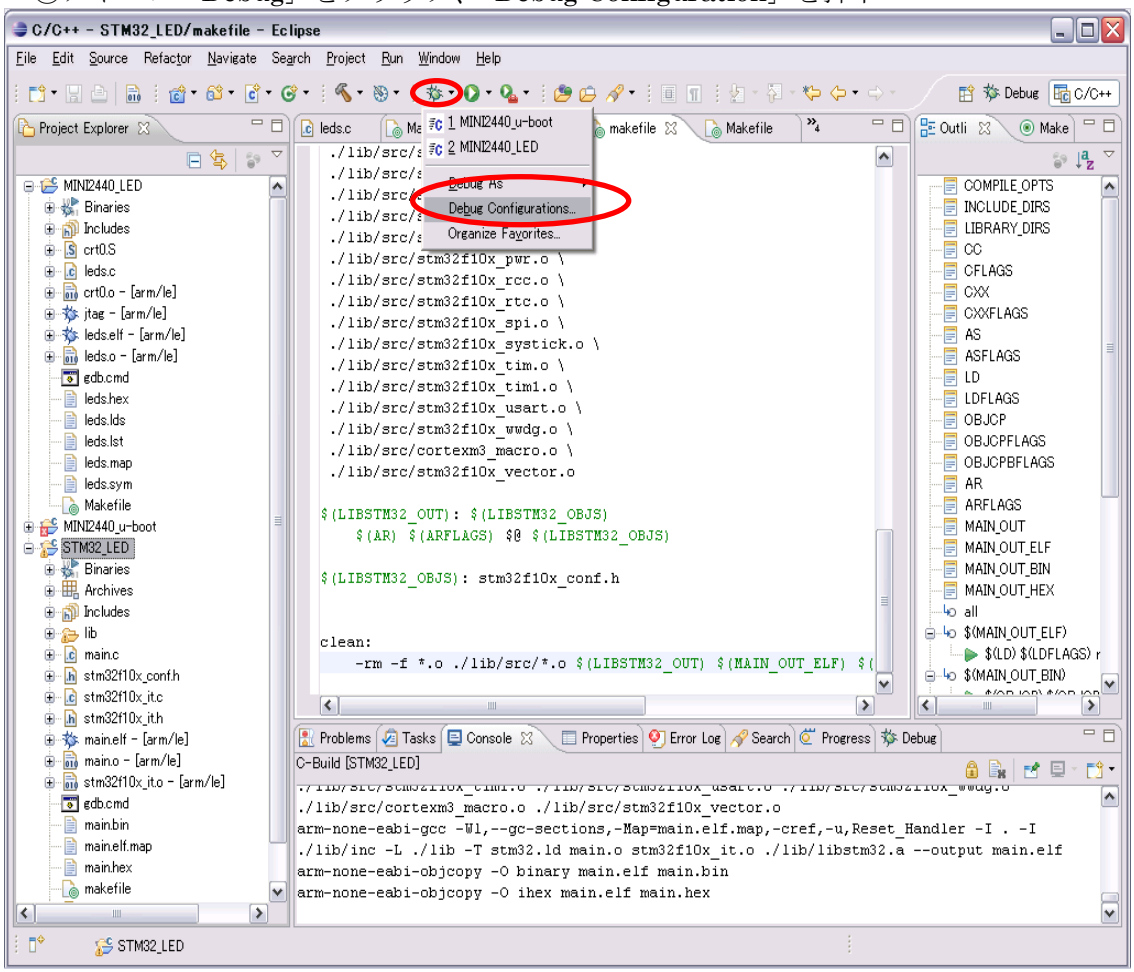
b)「Arguments」を下記のように変更

-f "D:¥embeded¥eclipse¥workspace¥config¥interface¥open-jtag.cfg" -f "D:¥embeded¥eclipse¥workspace¥config¥target¥stm32.cfg"

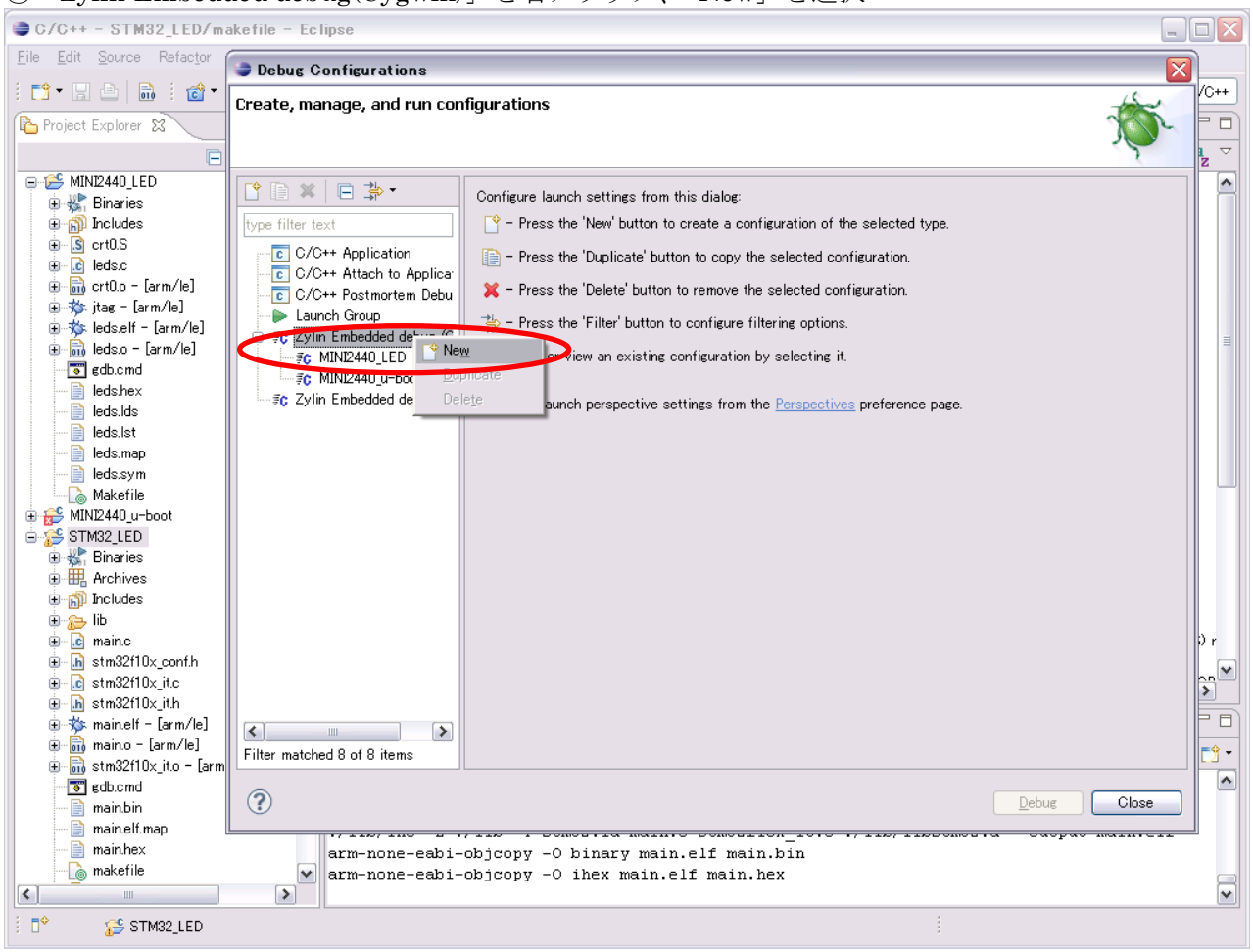


４．STM32_LED 用の GDB を追加

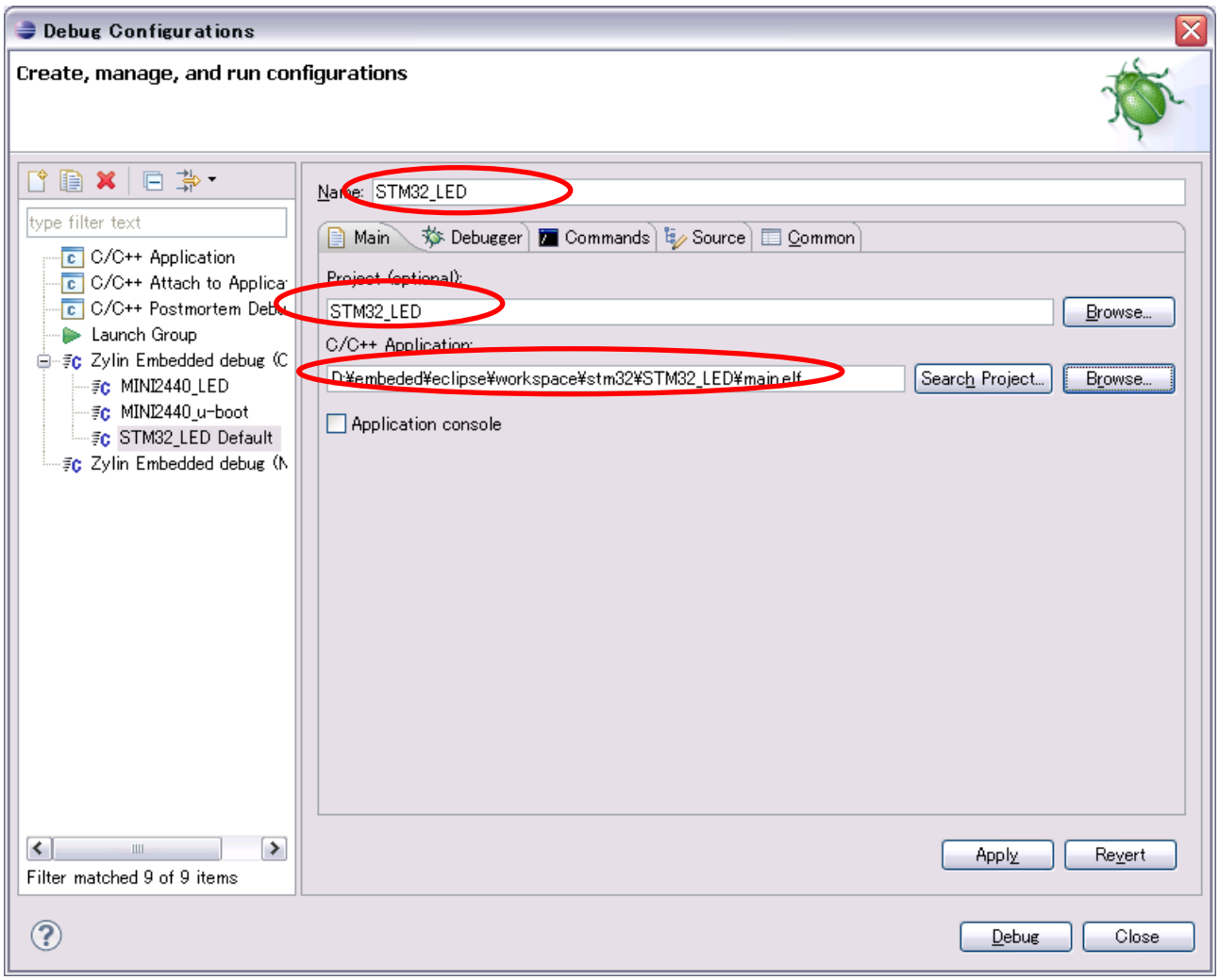
①アイコン「Debug」をクリック、「Debug Configuration」を押下



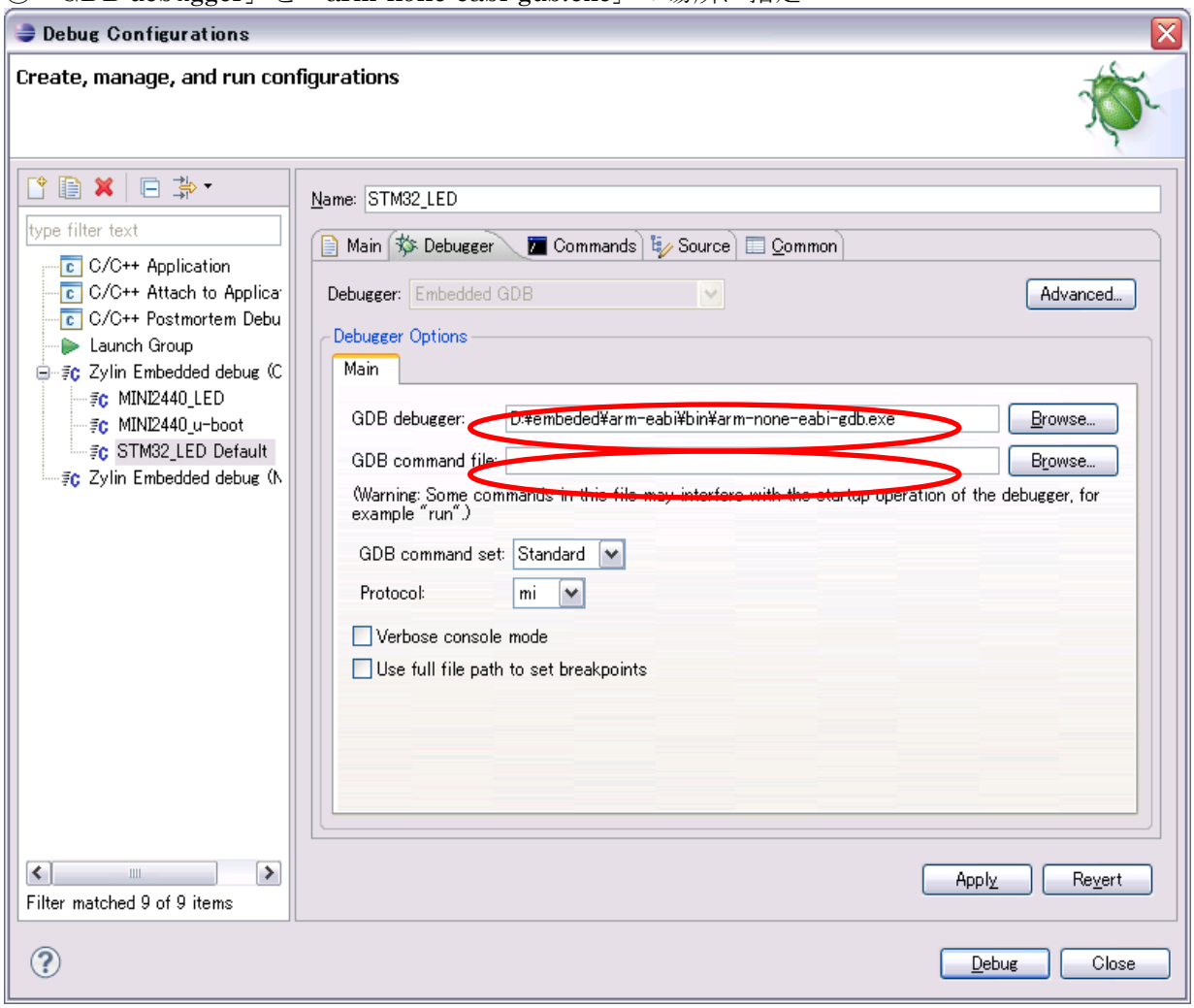
②「Zylin Embedded debug(Cygwin)」を右クリック、「New」を選択



③「Name」」を入力、「Project」、「C/C++ Application」を指定



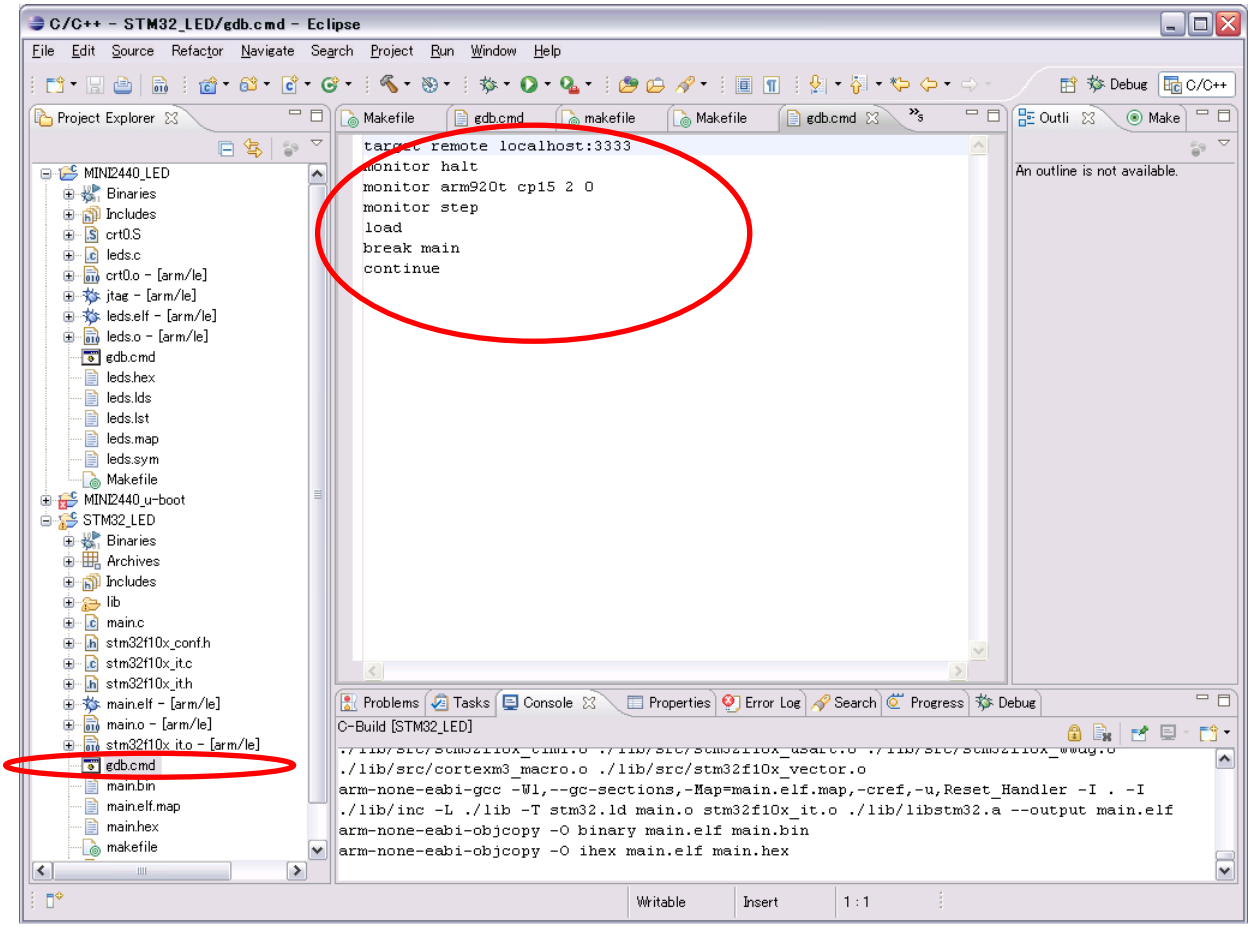
④「GDB debugger」を「arm-none-eabi-gdb.exe」の場所に指定



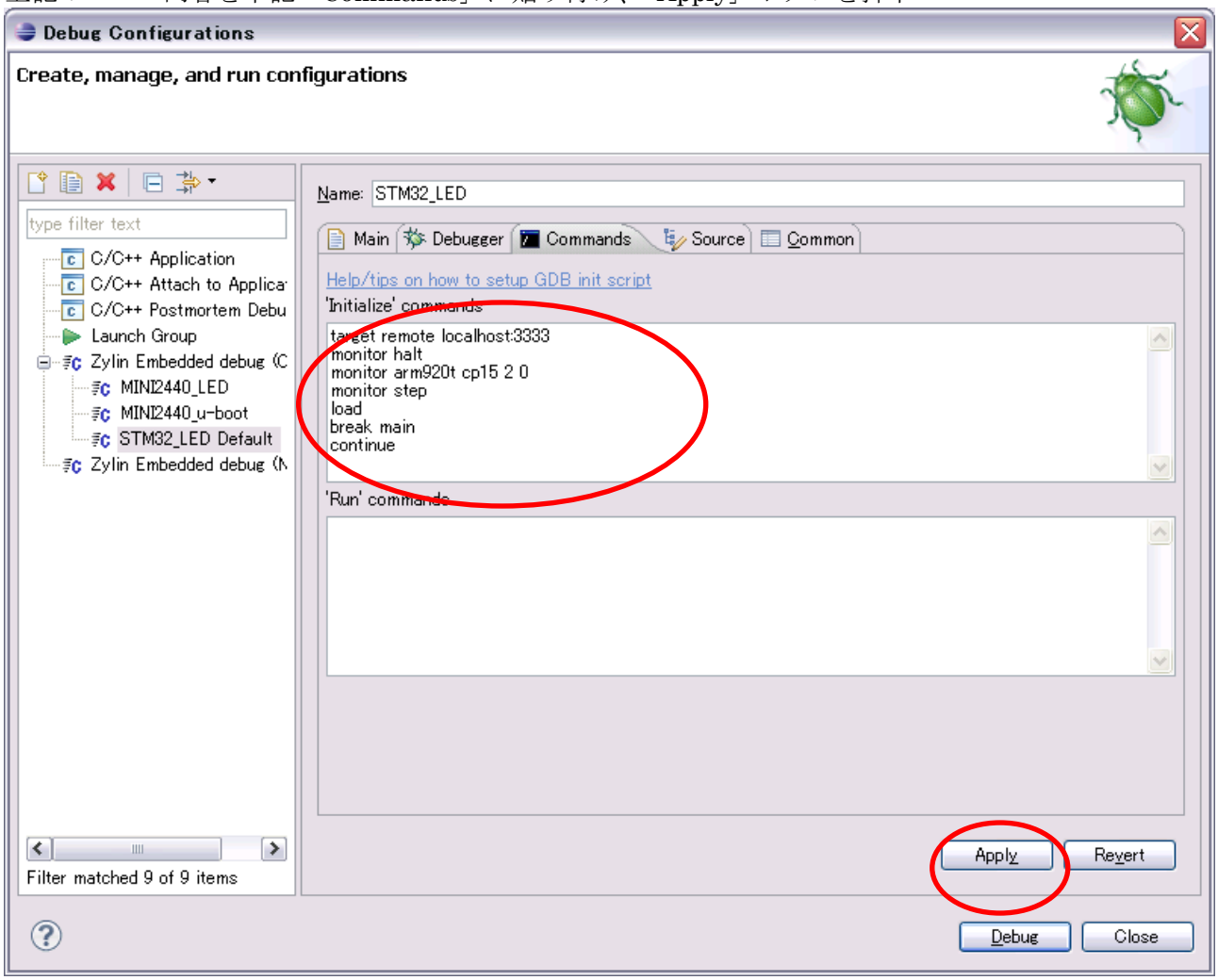
⑤「Commands」を入力

target remote localhost:3333 // ローカルポート「3333」と接続（OpenOCDと接続）
monitor halt//ボードの実行を中断させる
monitor step//ステップで実行するように
load // leds_elfをロード，「elf」というフォーマットのファイルにアドレスを含まれる
break main // 「main」関数にブレークポイントを設定
continue // プログラムを実行させて、「main」にとまってステップでデバッグ可能

「STM32_LED」プロジェクトの配下、「gdb.cmd」ファイルの内容をそのままコピーしても OK
「gdb.cmd」を右クリック、「Open With」→「Text Editor」
＊ほかの設定を保存するため、コピー途中で「Apply」ボタンを押して適用します。

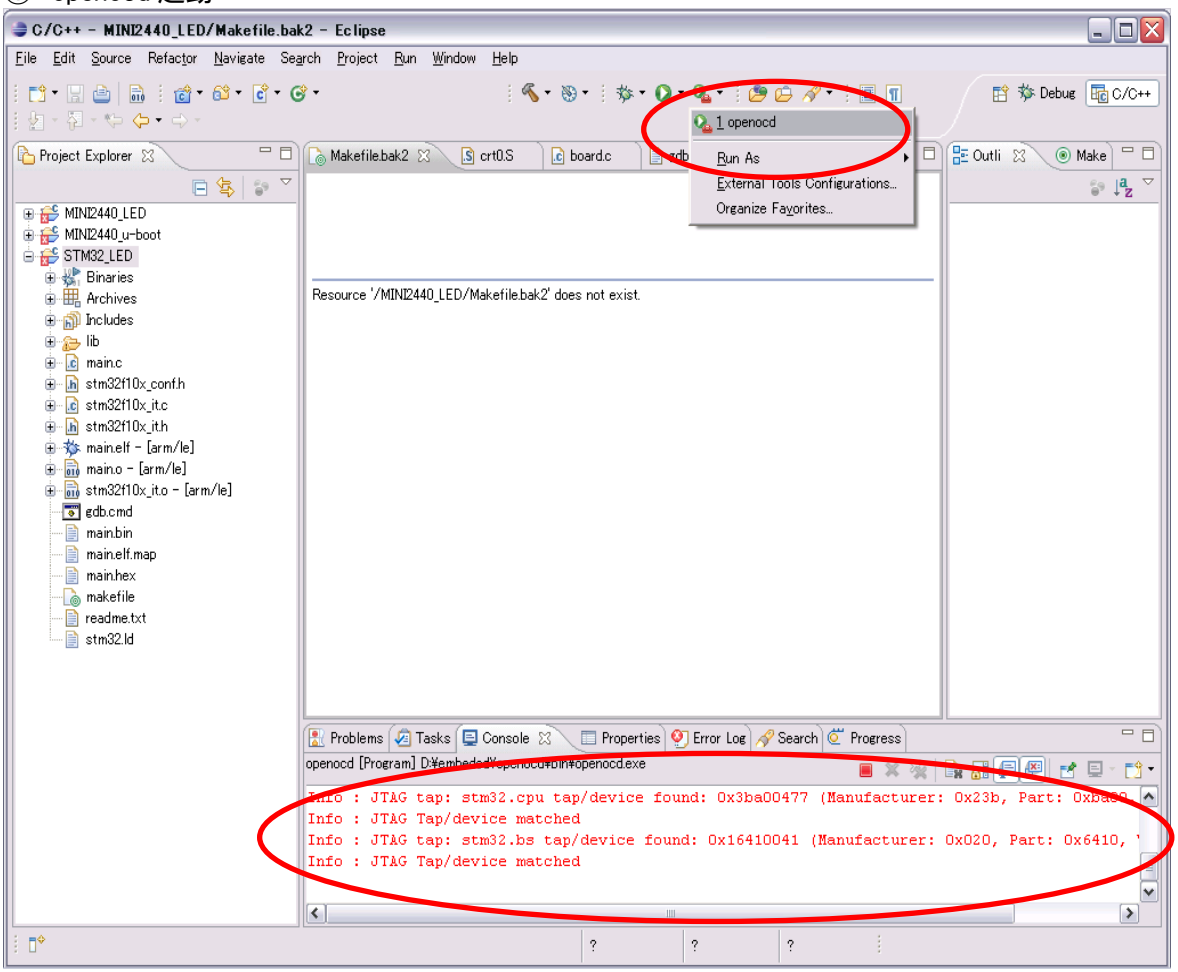


上記のコピー内容を下記「Commands」に貼り付け、「Apply」ボタンを押下

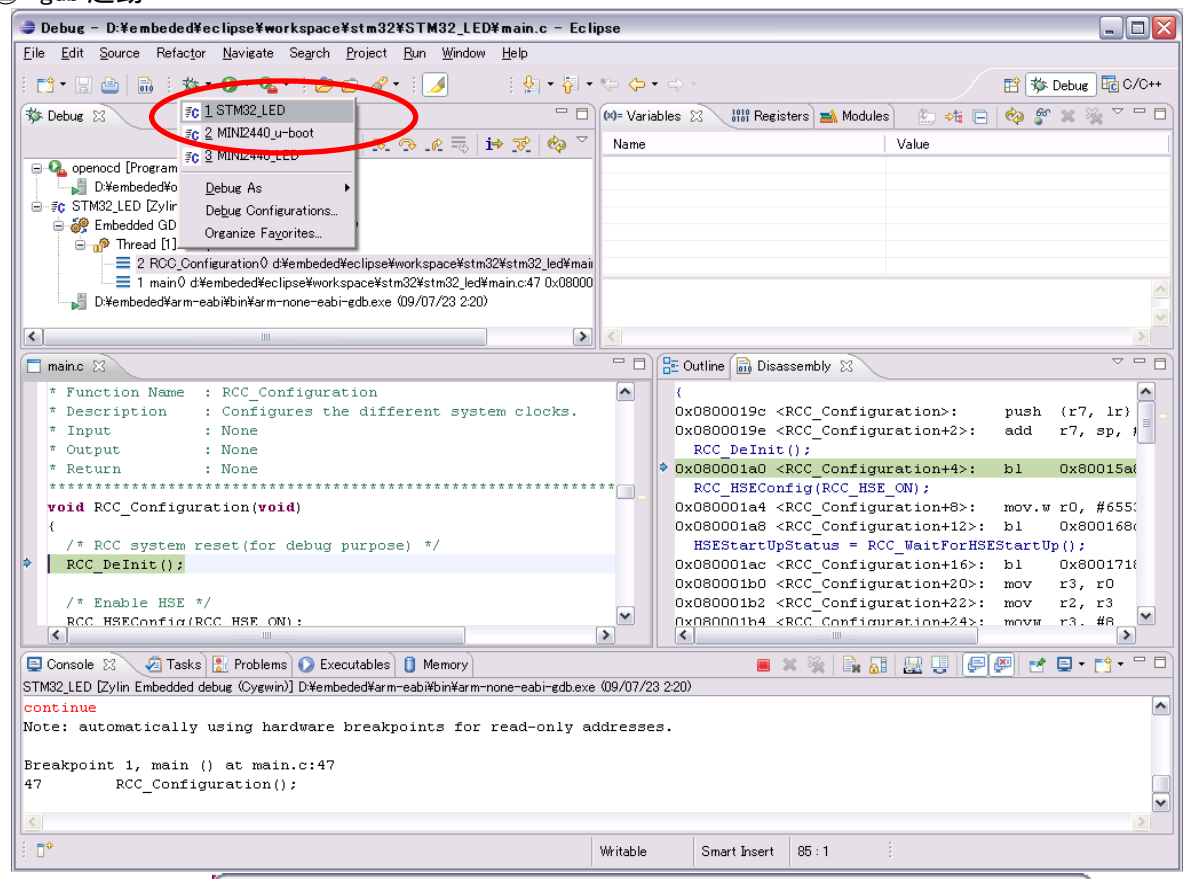


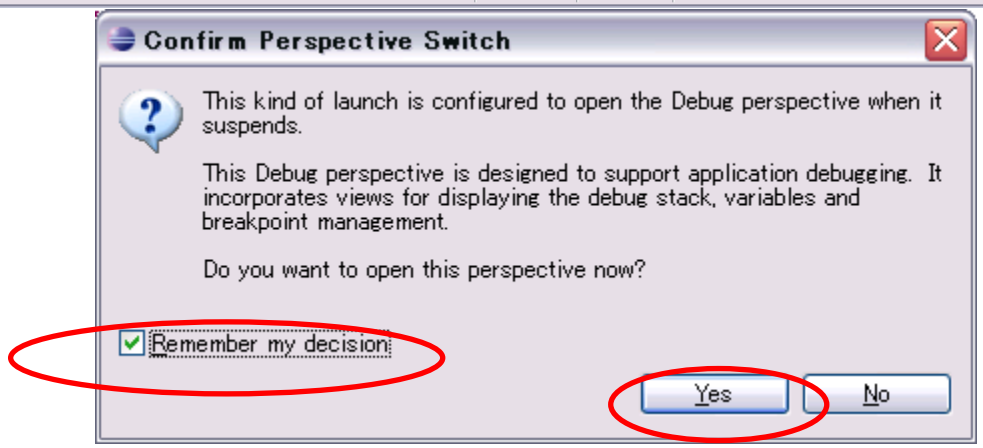
５．デバッグ

① openocd 起動

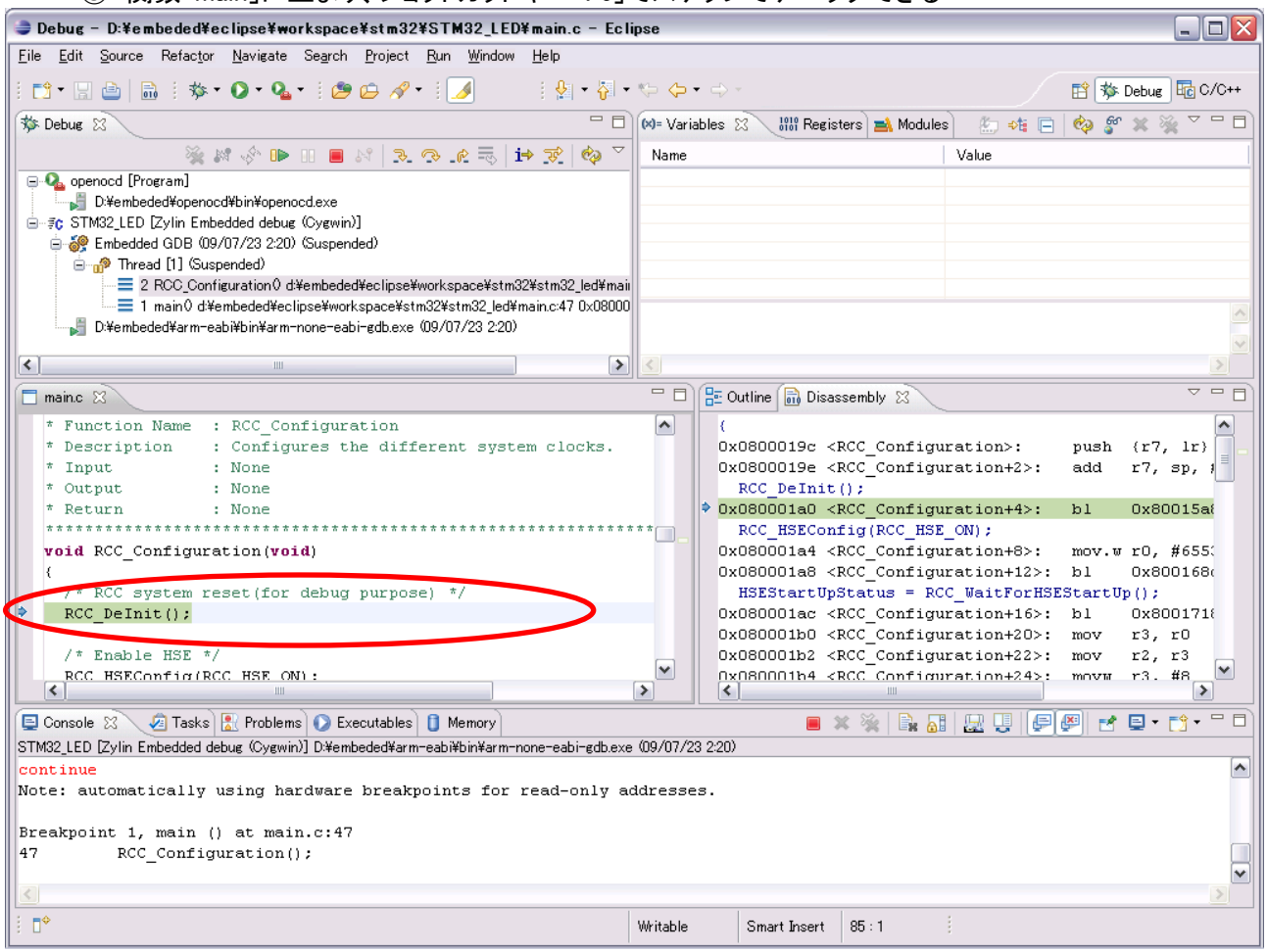


② gdb 起動

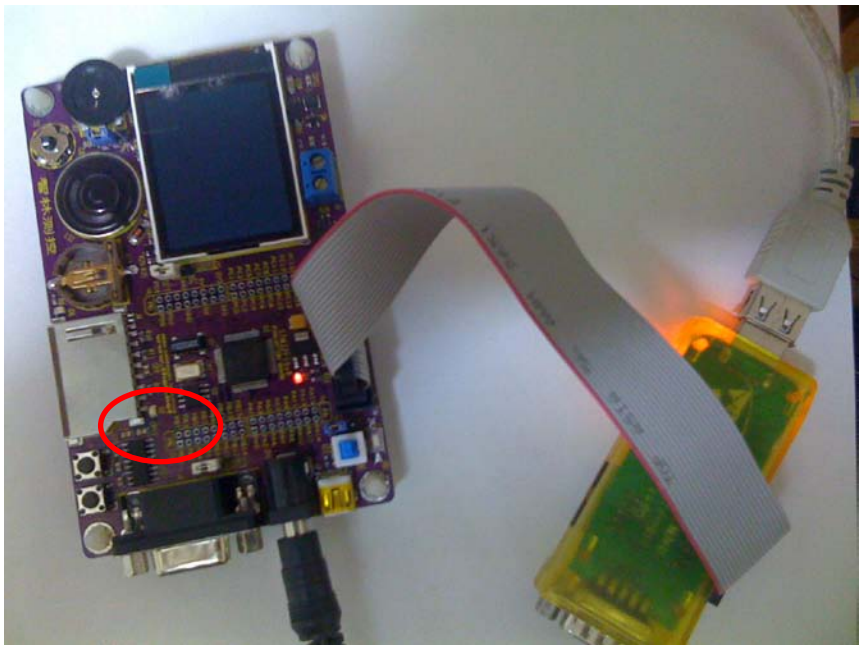

Debug - D:¥embeded¥eclipse¥workspace¥stm32¥STM32_LED¥main.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
1 STM32_LED
2 MINI2440_u-boot
3 MINI2440_LED
Debug As
Debug Configurations...
Organize Favorites...
Variables Registers Modules
Name Value
main.c
Outline Disassembly
Console Tasks Problems Executables Memory
STM32_LED [Zylin Embedded debug (Cygwin)] D:¥embeded¥arm-eabi¥bin¥arm-none-eabi-gdb.exe (09/07/23 2:20)
continue
Note: automatically using hardware breakpoints for read-only addresses.
Breakpoint 1, main () at main.c:47
47 RCC_Configuration();
Writable
Smart Insert
85 : 1


Confirm Perspective Switch
This kind of launch is configured to open the Debug perspective when it suspends.
This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.
Do you want to open this perspective now?
Remember my decision
Yes
No

③ 関数「main」に止まり、ショットカットキー「F6」でステップでデバッグできる



④　ステップでデバッグ
＊関数「main」のコーディング「`GPIO_SetBits(GPIOB, GPIO_Pin_5);`」を実行前、
要は、LED 点灯前に、STM32 ボードの LED が消している



ショットカットキー「F6」を押しながら、関数「main」のコーディング「`GPIO_SetBits(GPIOB, GPIO_Pin_5);`」を実行後、SMT32 ボードの LED が点灯している

6.4 ARM9 の MINI2440

6.4.1 MINI2440 を購入

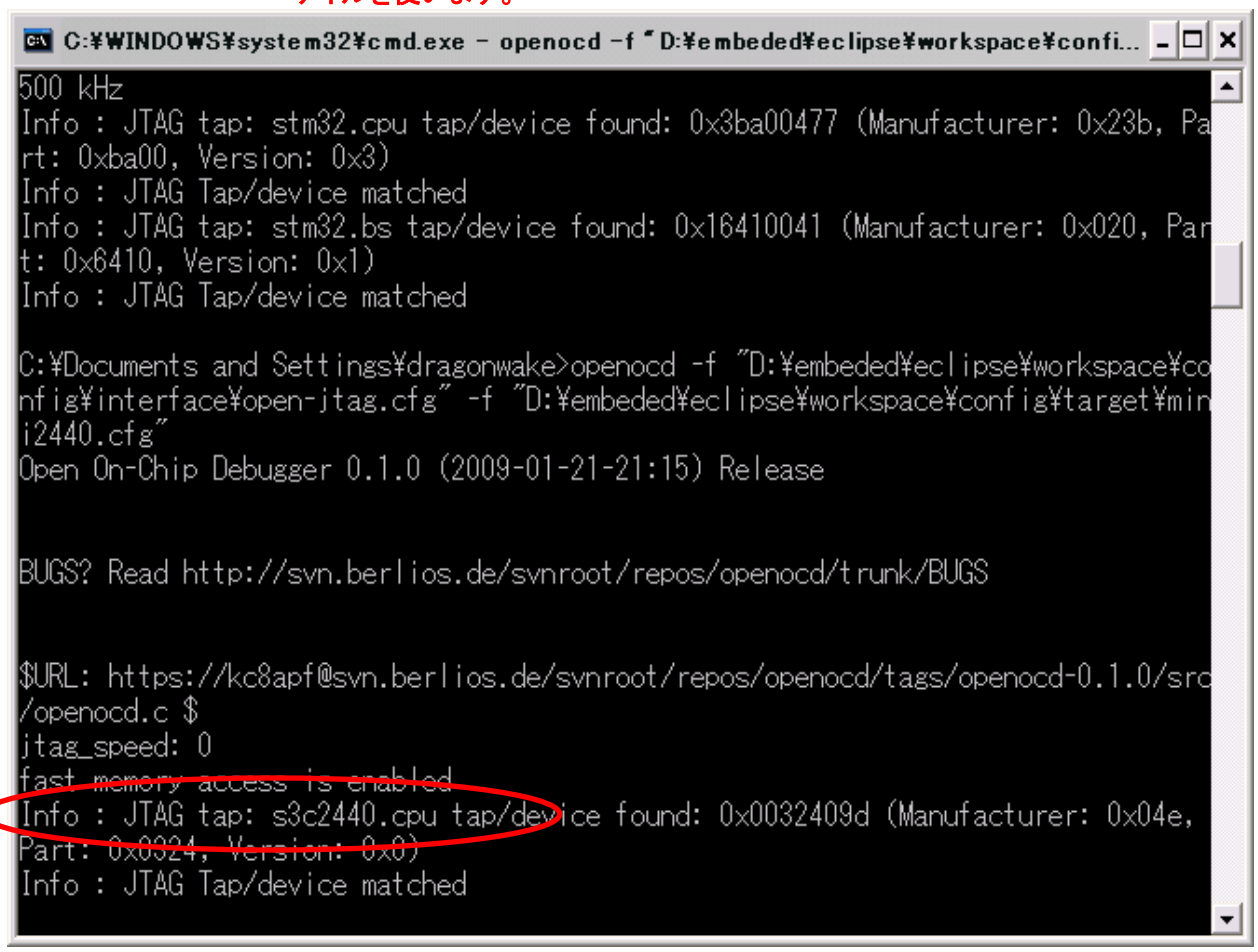
MINI2440 ボード購入 URL：(LCD3.5 とのセット)
http://www.csun.co.jp/SHOP/200812021.html
LPC2148 マニュアルダウンロード URL：
http://www.dragonwake.com/download/arm9-download/linux-2.6.29/MINI2440-linux-2.6.29.pdf

6.4.2 ハードウエア動作確認

1. OpenJTAG をパソコンの USB ポートに挿入する
2. JTAG ケーブルで OpenJTAG と MINI2440 ボードを繋ぐ
3. MINI2440 ボードに電源を入れる
4. 下記のコマンドを入力します。

openocd -f "D:¥embeded¥eclipse¥workspace¥config¥interface¥open-jtag.cfg" -f "D:¥embeded¥eclipse¥workspace¥config¥target¥mini2440.cfg"

※ はじめの-f は OpenJTAG のコンフィグファイルを使います。二番目の-f は mini2440 のコンフィグファイルを使います。



6.4.3 LED サンプルデバッグ

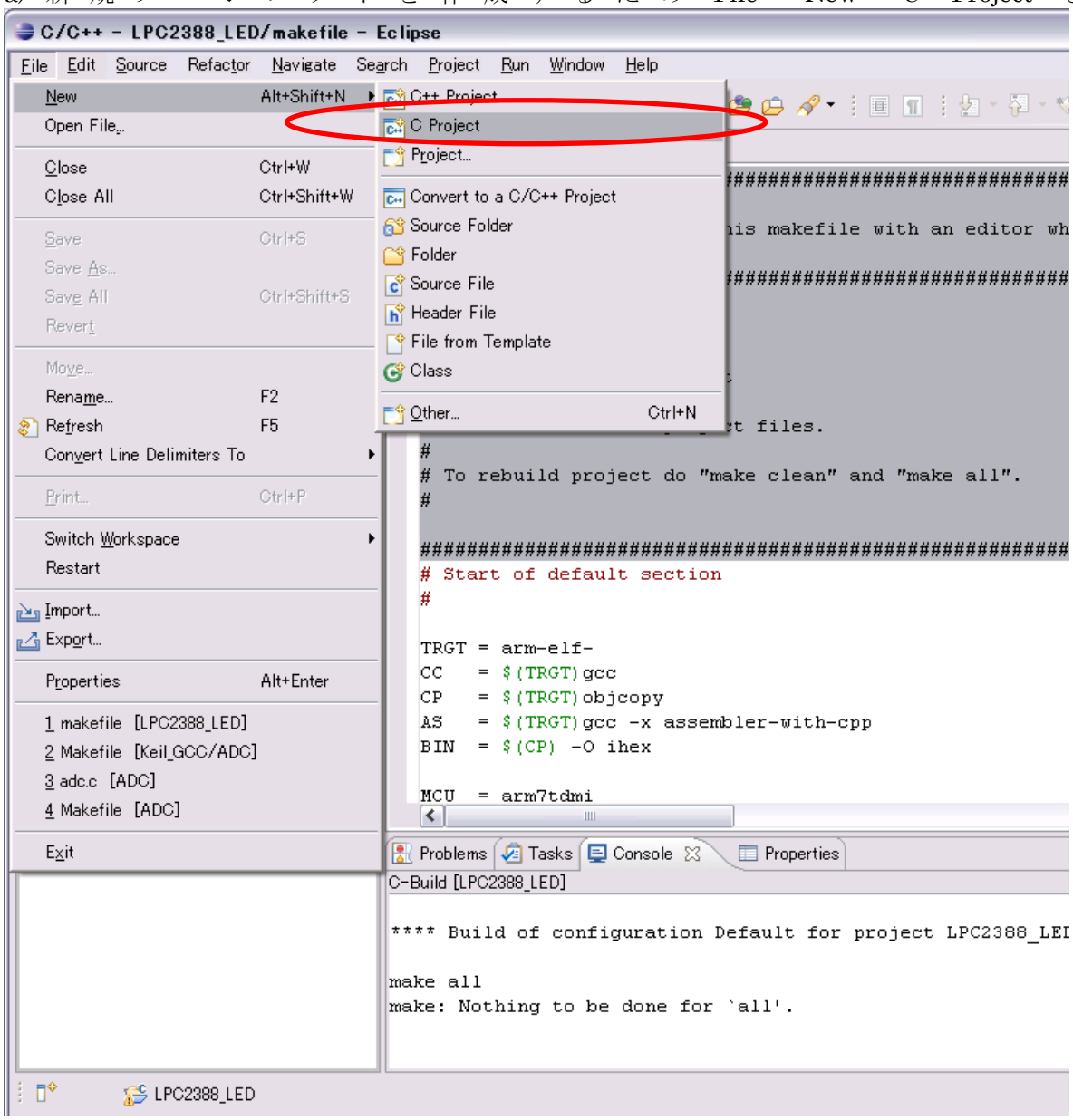
１．サンプルの導入とコンパイル：

サンプルの導入について、二つ方法があります、一つは新規プロジェクトを作成してからソースを導入、もう一つはダウンロードプロジェクトをインポートします。

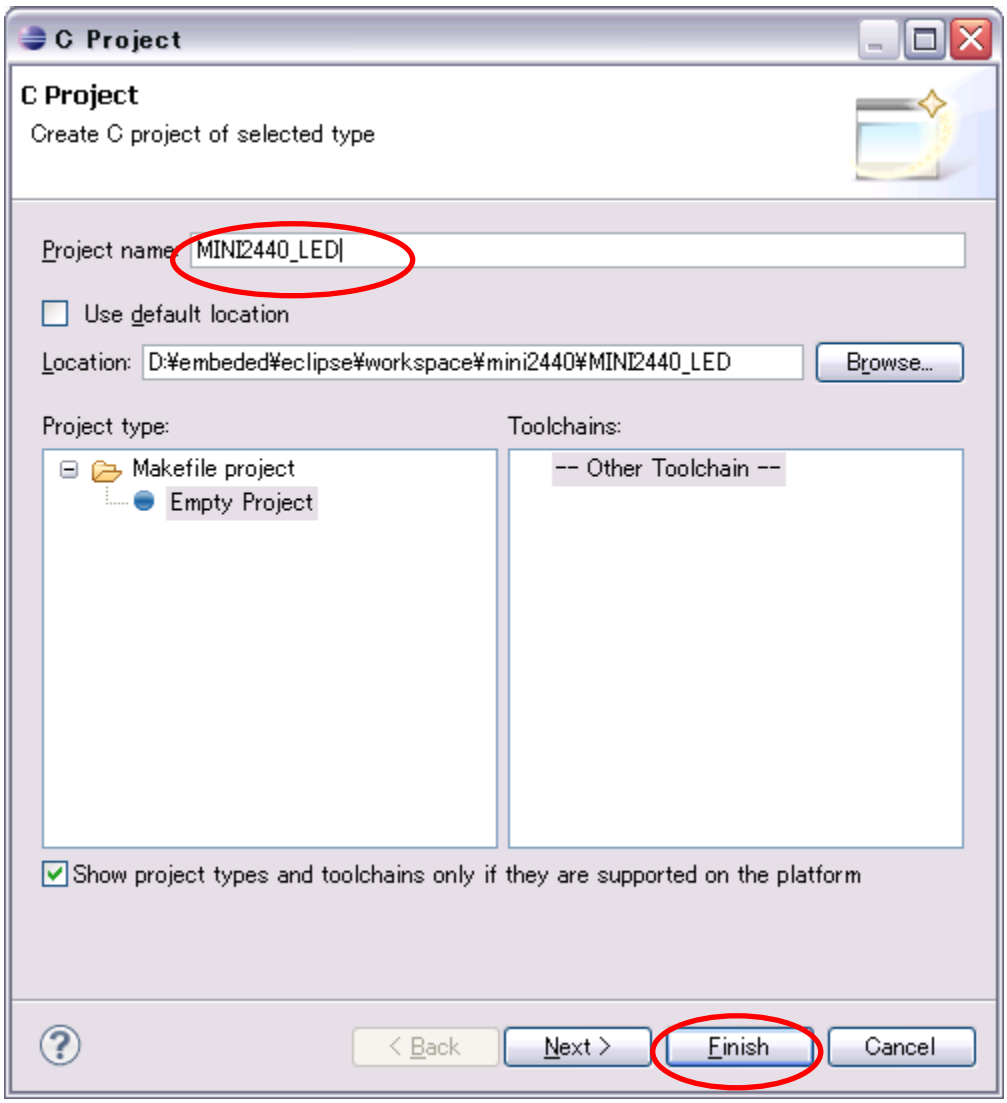
＊ここに新規プロジェクトの作成方法として説明

①プロジェクト「MINI2440_LED」を作る

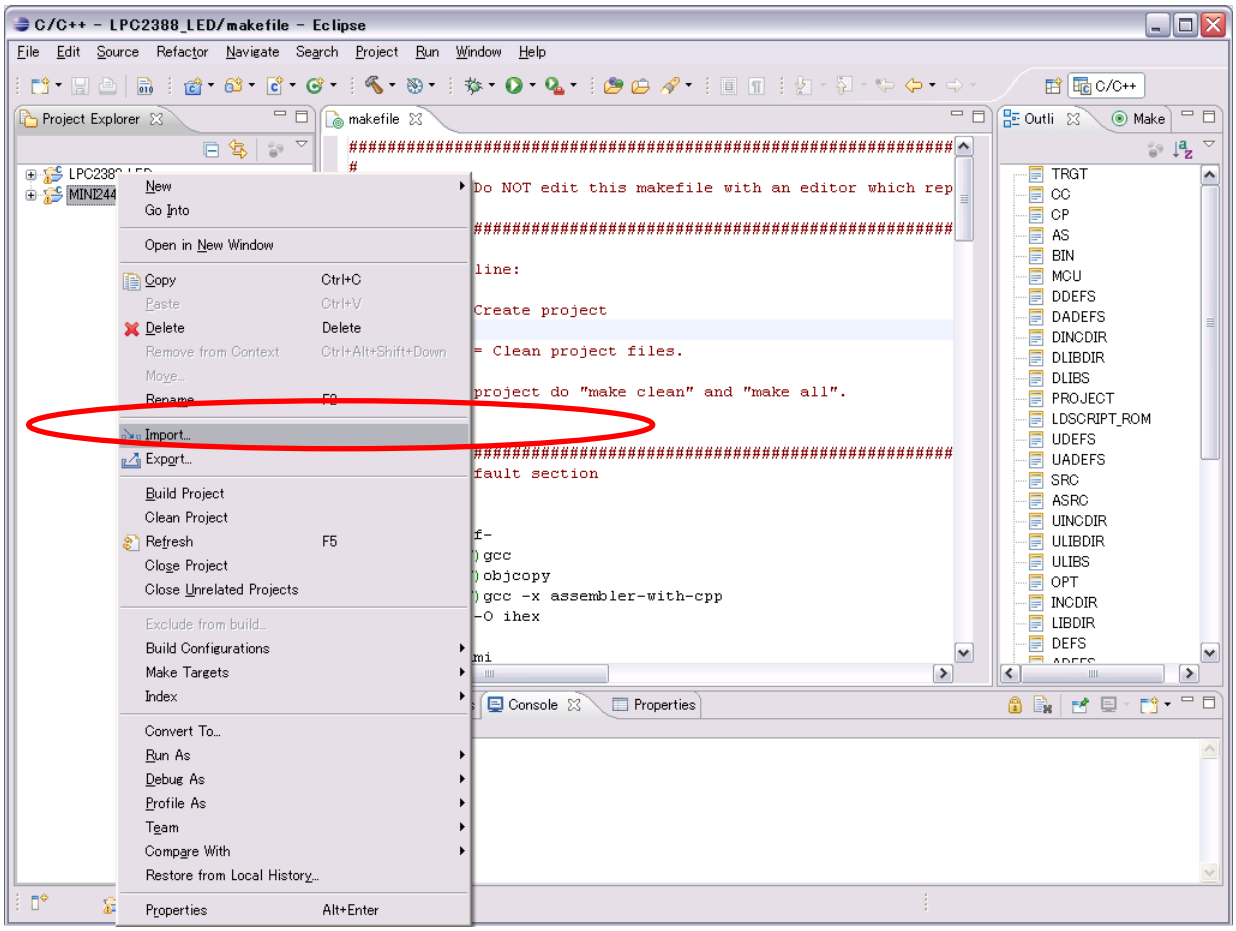
a) 新規プロジェクトを作成するため "File"→"New"→"C Project" を選択します



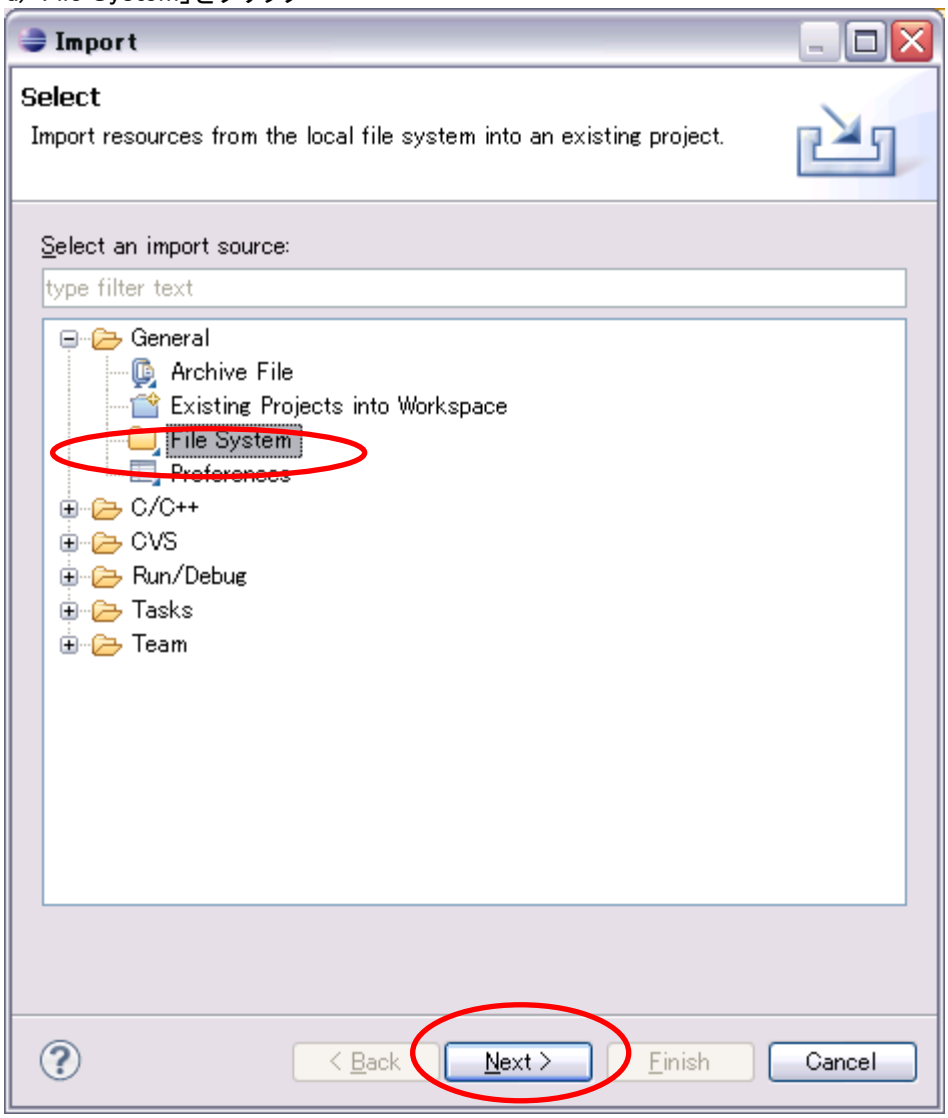
b)適当なプロジェクト名前(MINI2440_LED)を入力し Finish ボタンを押します。



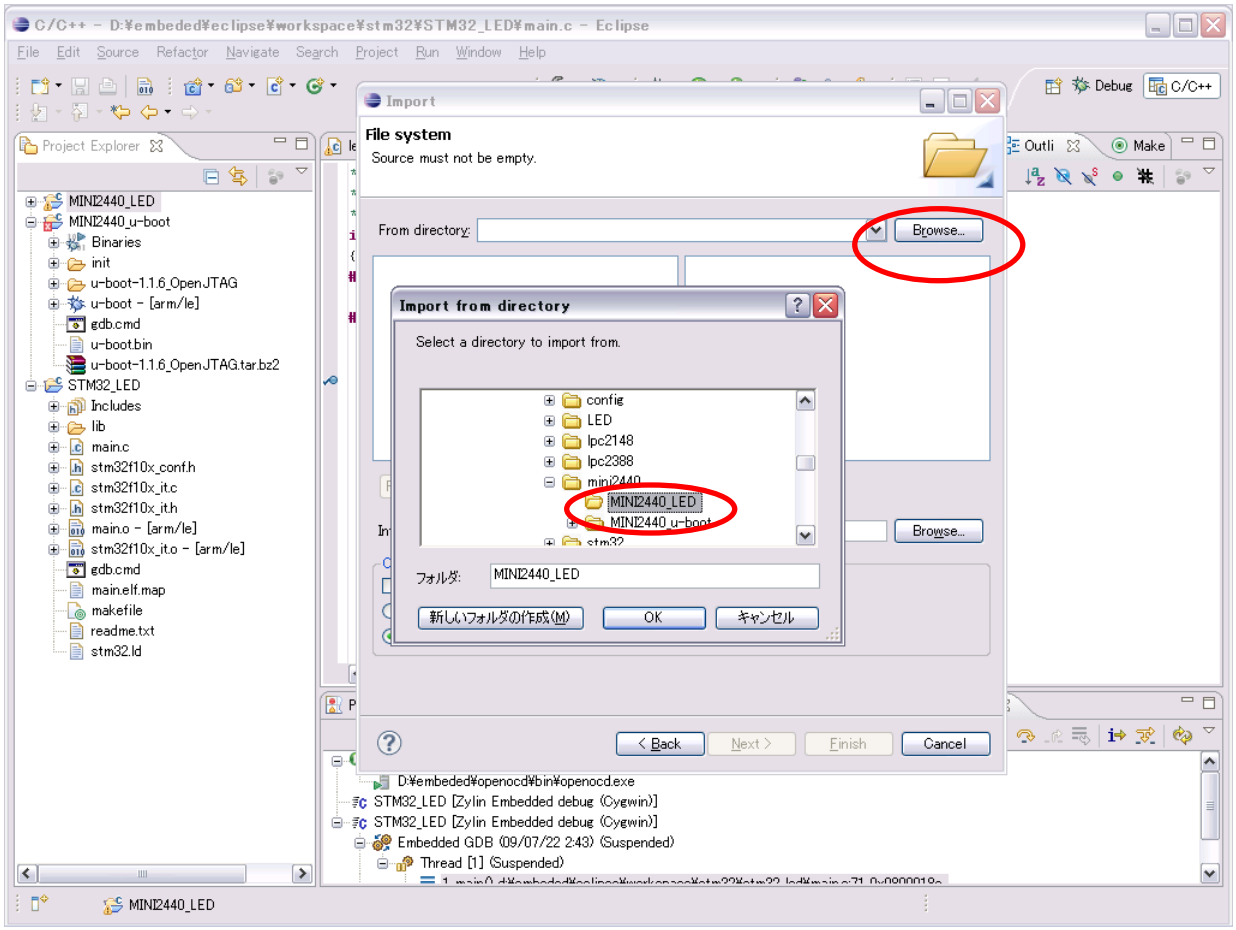
c)サンプルソースを作成されたプロジェクトにインポート
Project Explorer の「MINI2440_LED」を選べ、右クリックし、「Import」を押下



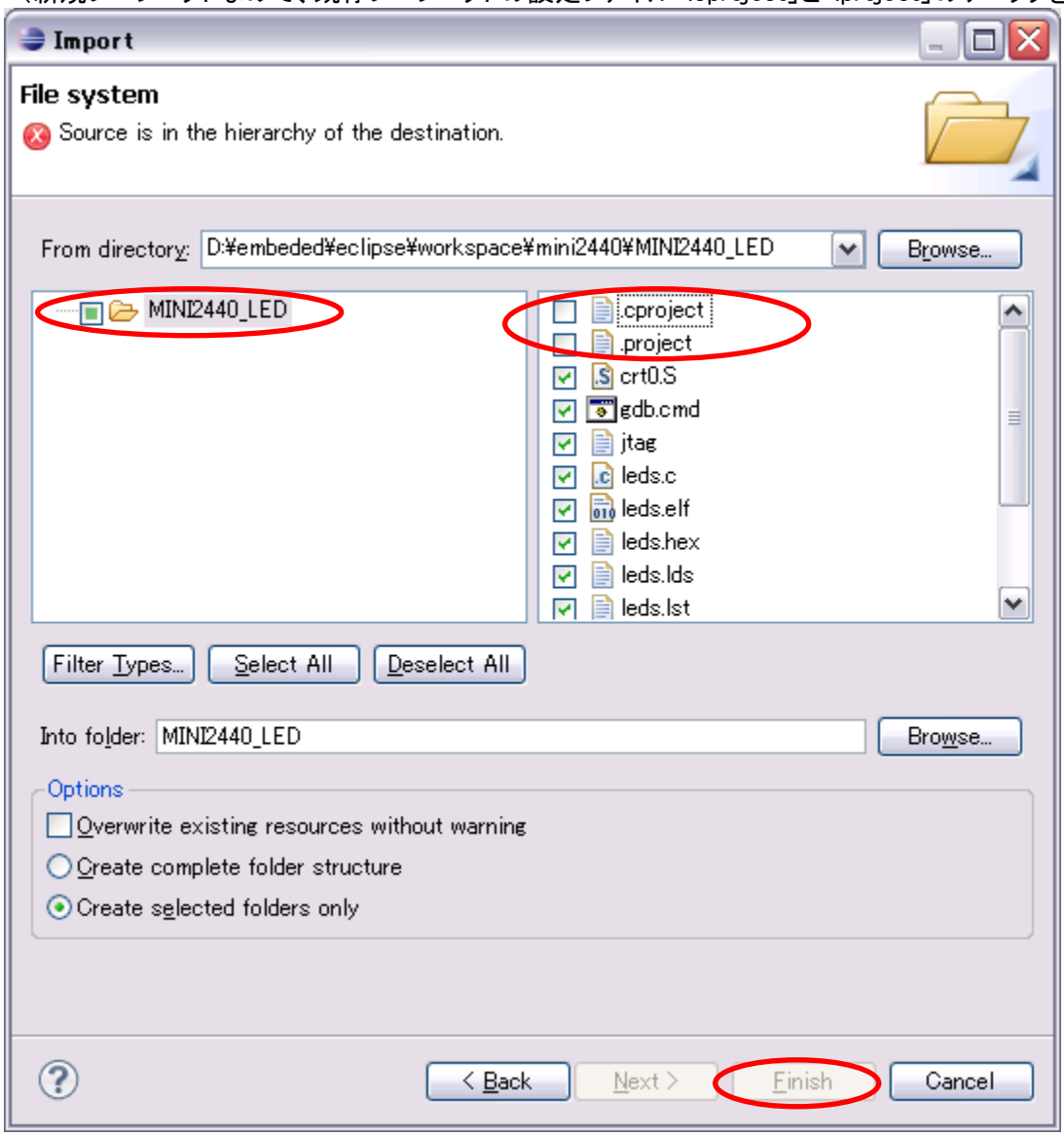
d)「File System」をクリック



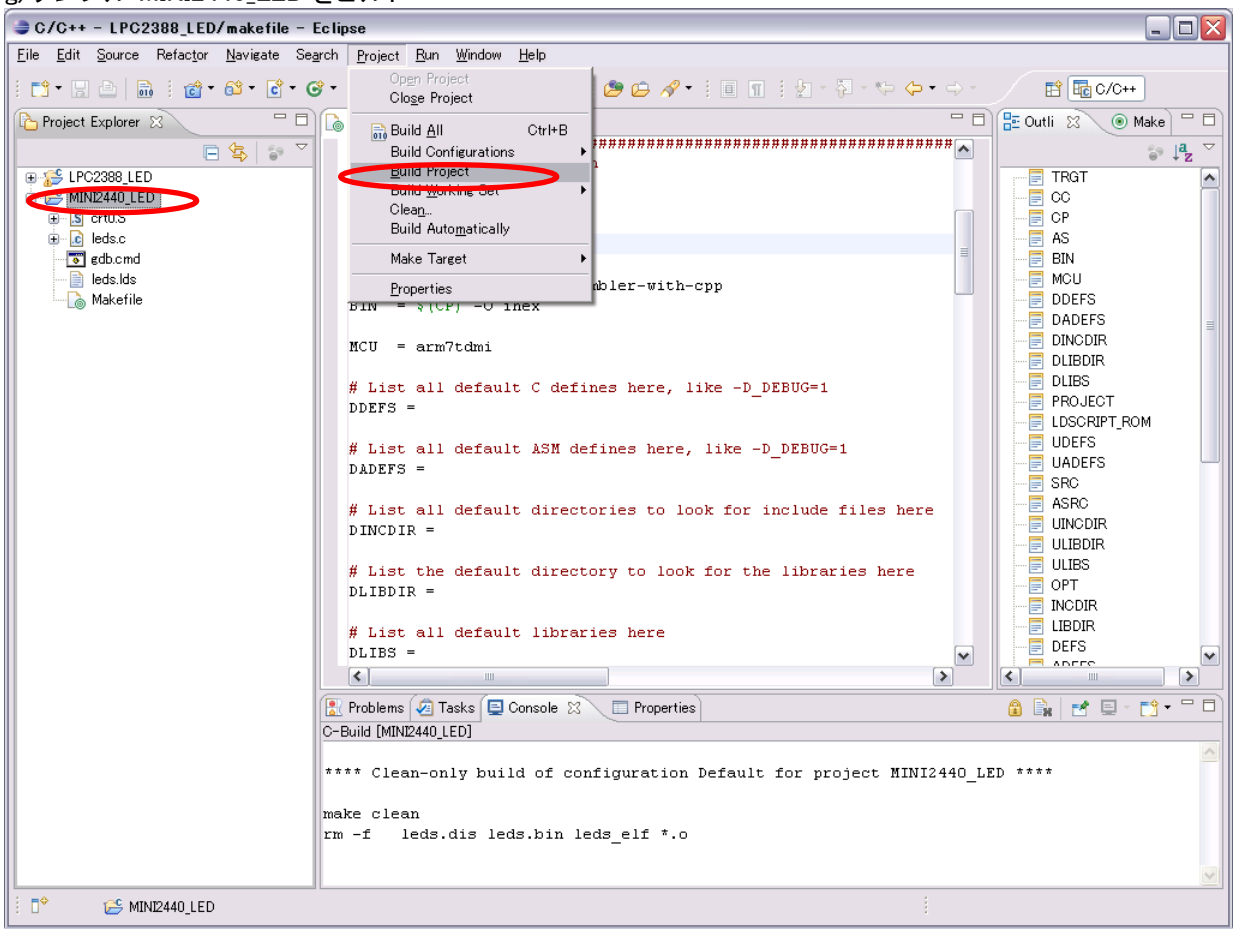
e）ダウンロードしたサンプル MINI2440_LED を選択



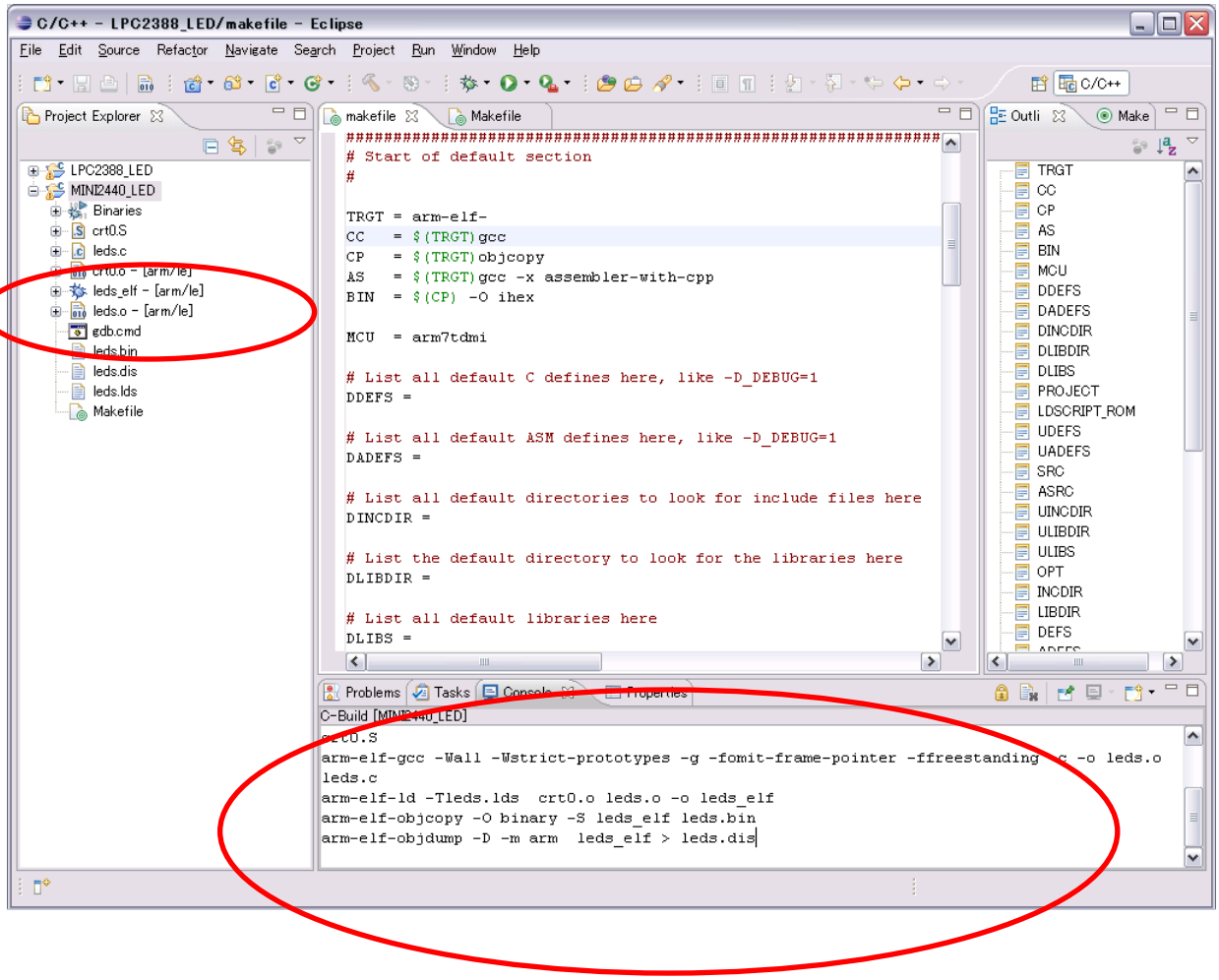
f)選択したソースを導入
（新規プロジェクトなので、既存プロジェクトの設定ファイル「.cproject」と「.project」のチェックを外す）



g)サンプル MINI2440_LED をビルド



h)ビルドが成功かどうかを確認



2. サンプル MINI2440_LED をデバッグ

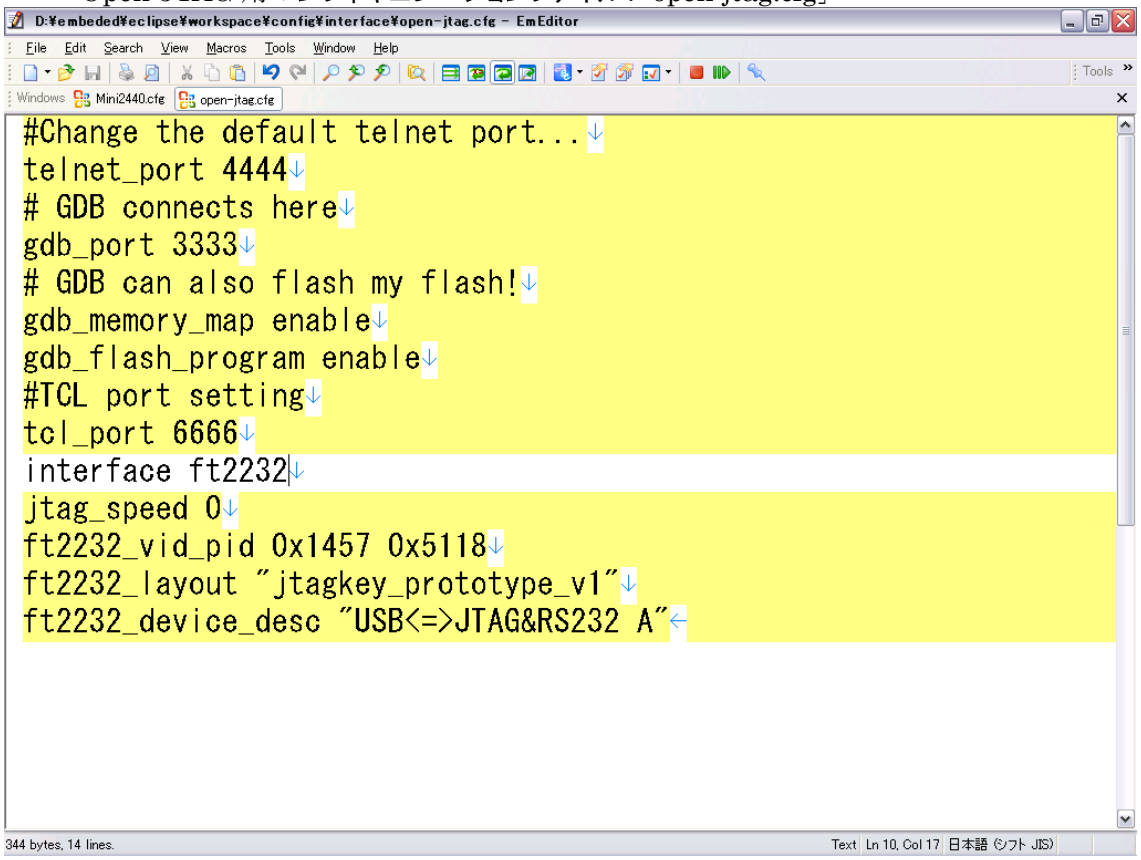
①MINI2440 用の OpenOCD デバッグパラメタを設定

＊ OpenOCD の設定は**「5.4 OpenOCD の設定」**(P23～P26)を参照

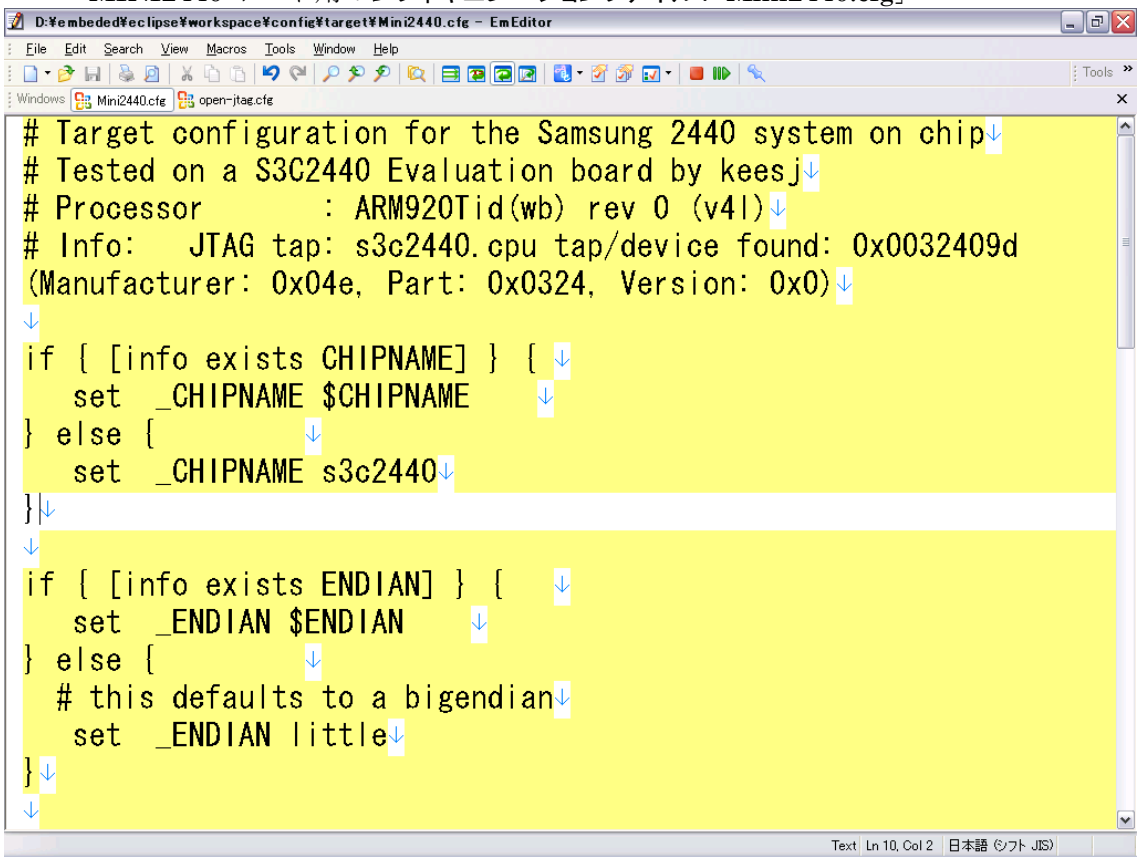
a) MINI2440 用の**コンフィギュレーションファイル内容を確認**

「config/interface」の中：JTAG の種類による**コンフィギュレーションファイル**

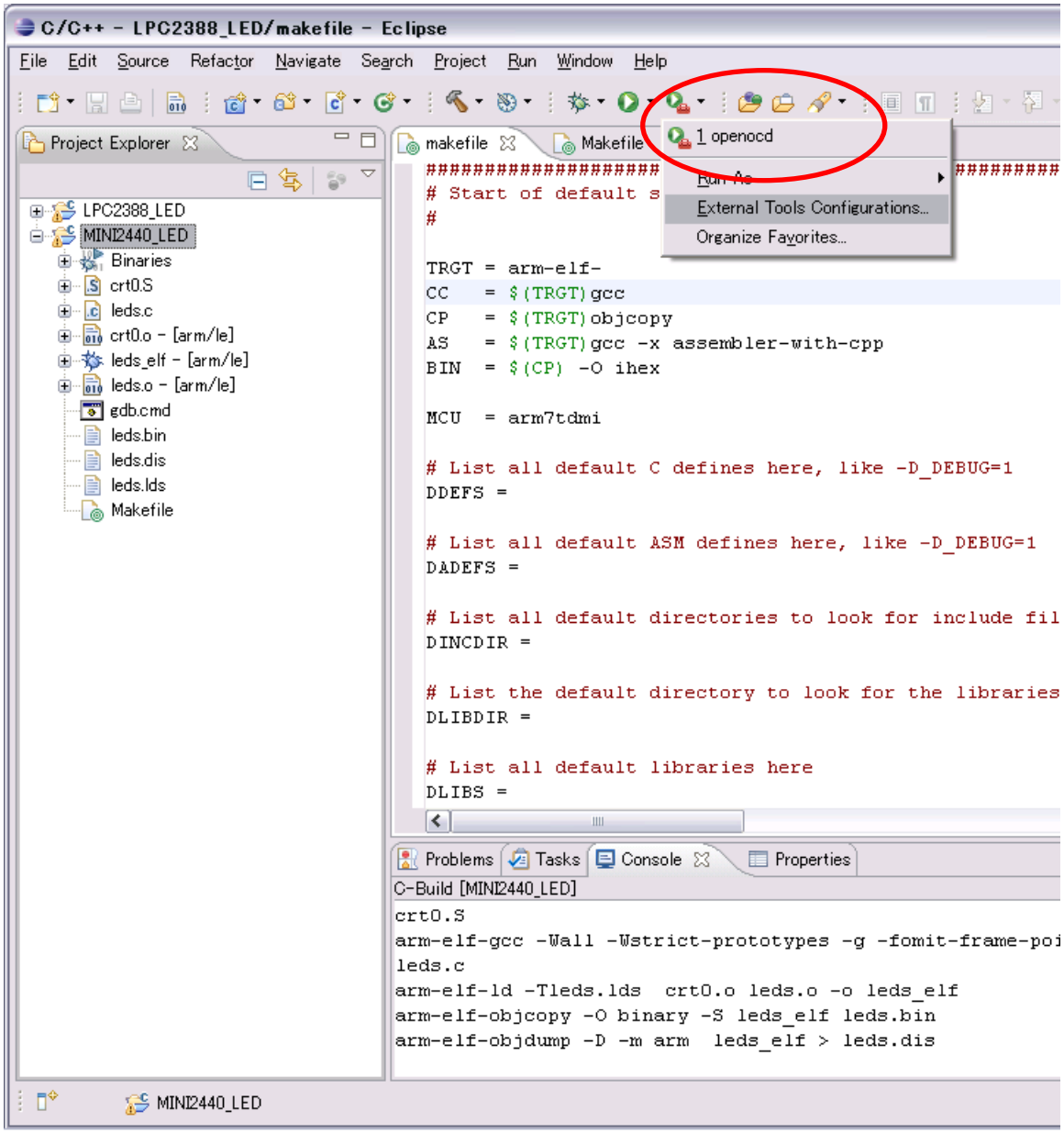
Open-JTAG 用の**ンフィギュレーションファイル**:「open-jtag.cfg」



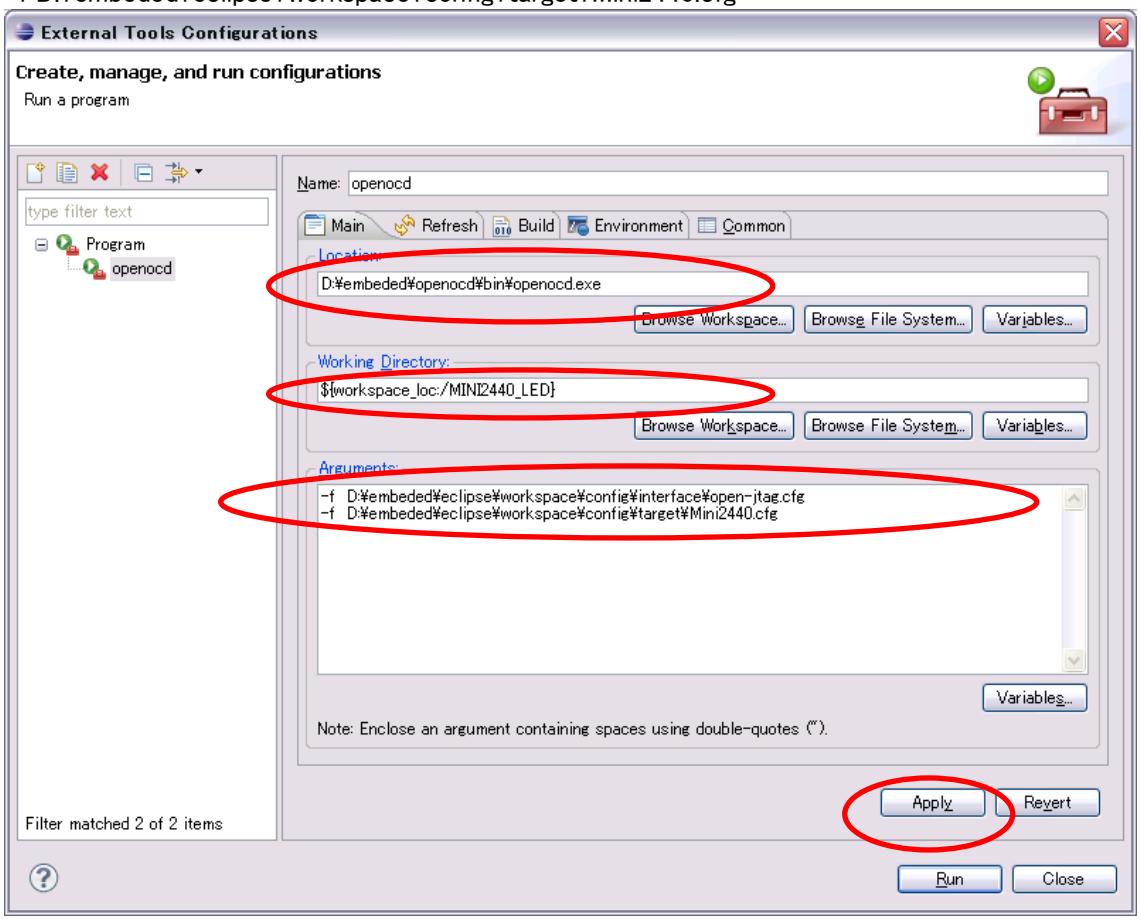
b) config/target」の中：CHIP に関わるコンフィギュレーションファイル
MINI2440 ボード用のコンフィギュレーションファイル:「Mini2440.cfg」



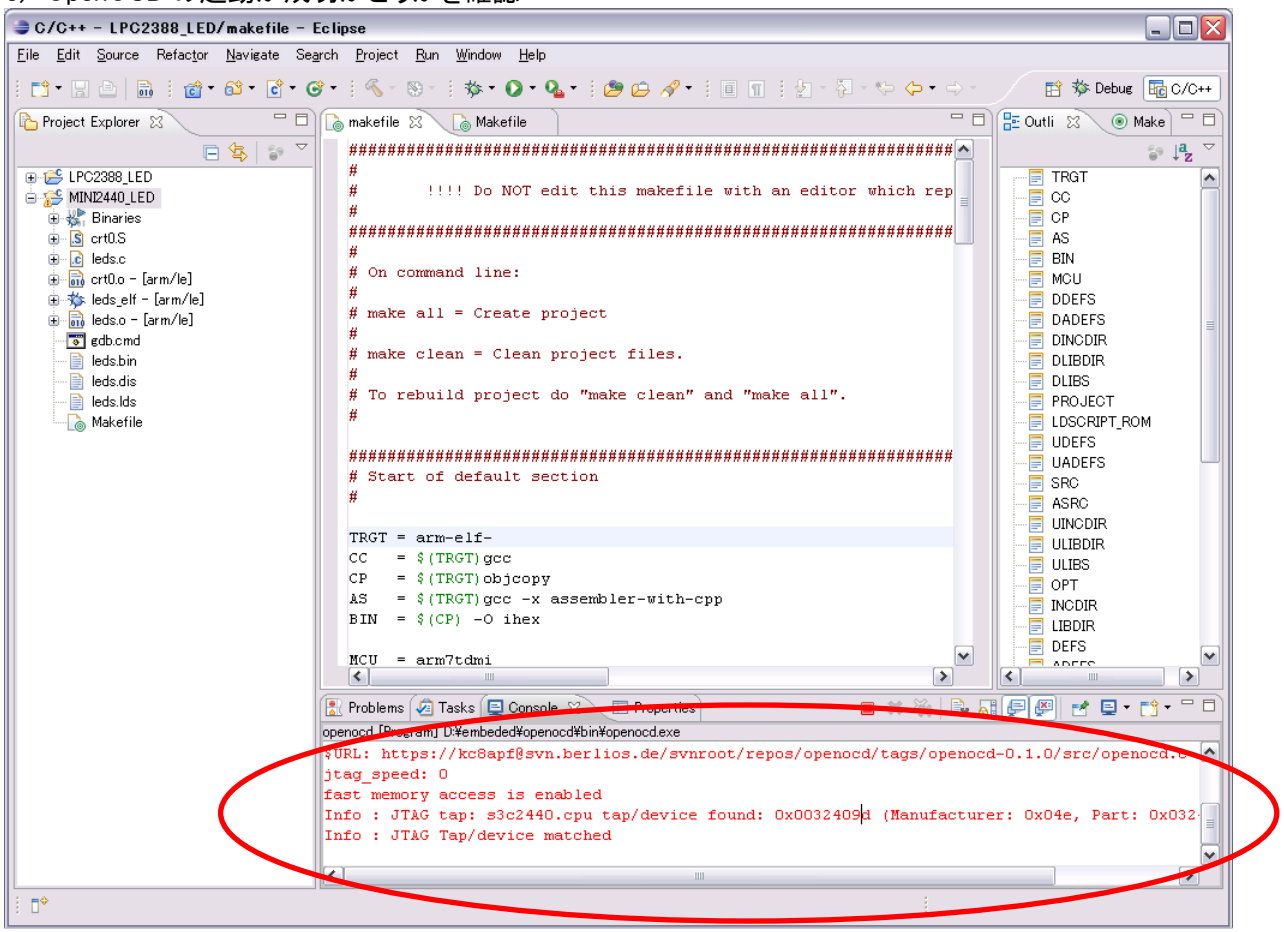
c） OpenOCD の設定画面を呼び出す



d) 「Location」、「Working　Directory」、「Arguments」を設定
＊特に「Arguments」に MINI2440 用のコンフィギュレーションファイルを設定必要
–f D:¥embeded¥eclipse¥workspace¥config¥interface¥open–jtag.cfg
–f D:¥embeded¥eclipse¥workspace¥config¥target¥Mini2440.cfg

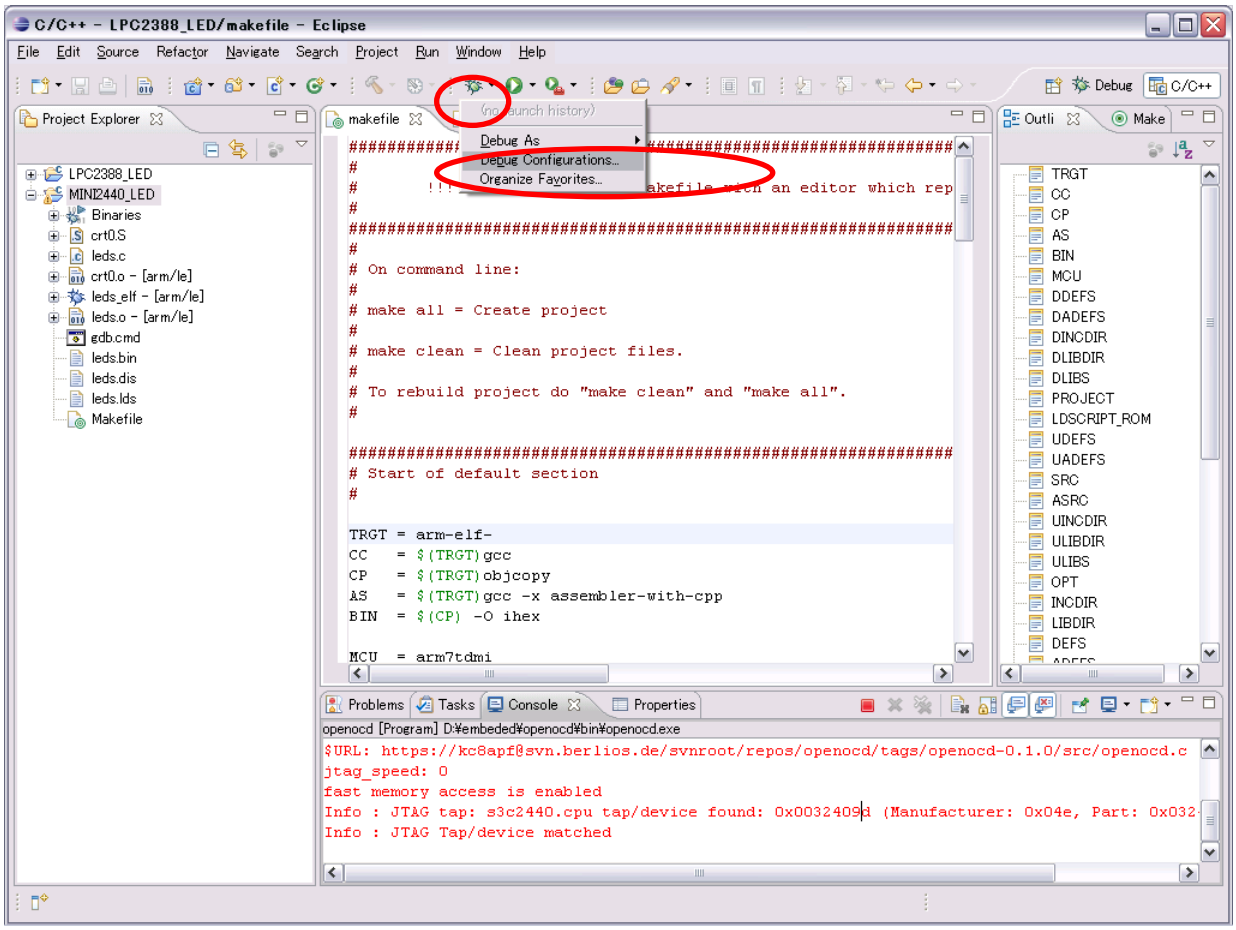


e）OpenOCD の起動が成功かどうかを確認

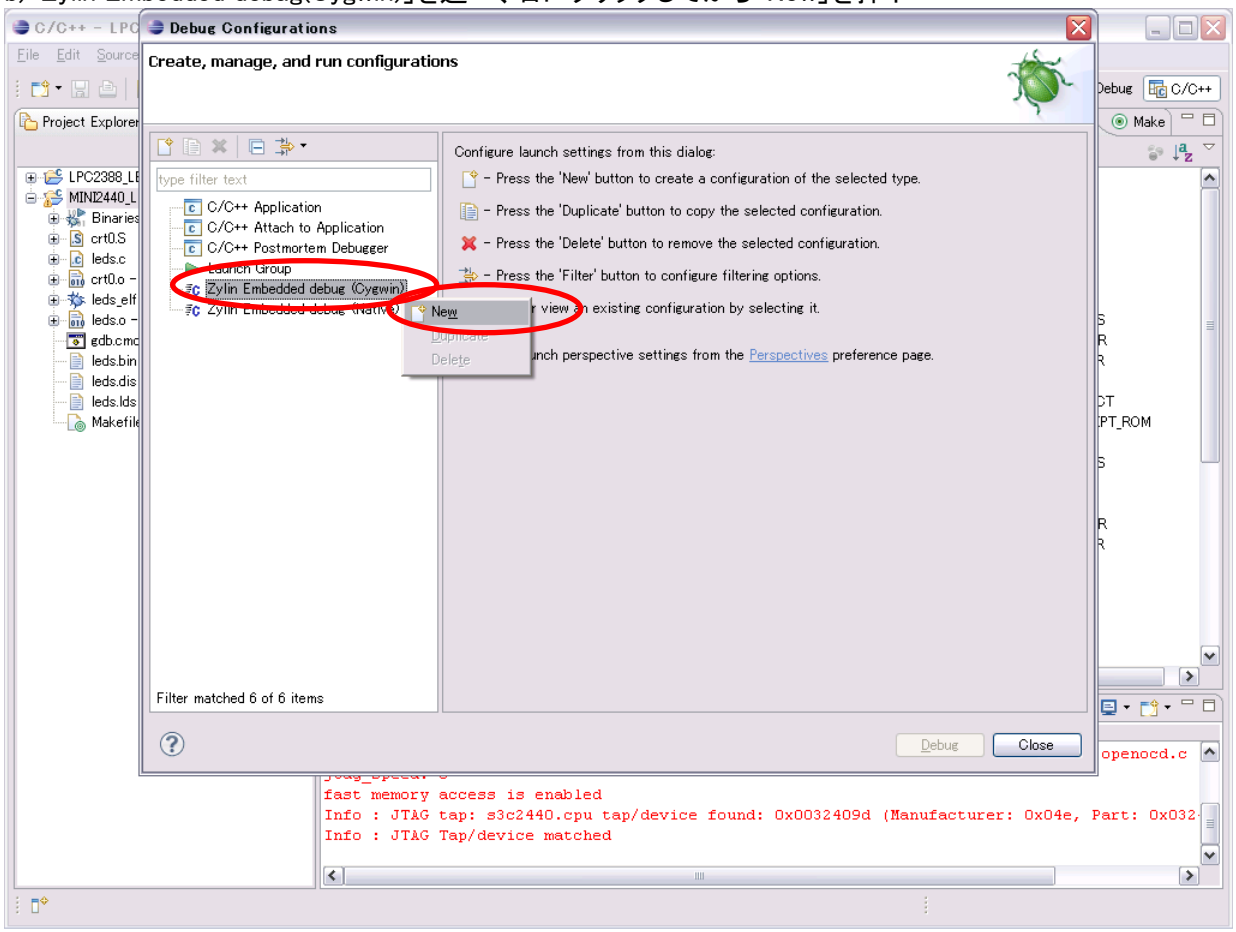


②MINI2440 用のデバッグを設定

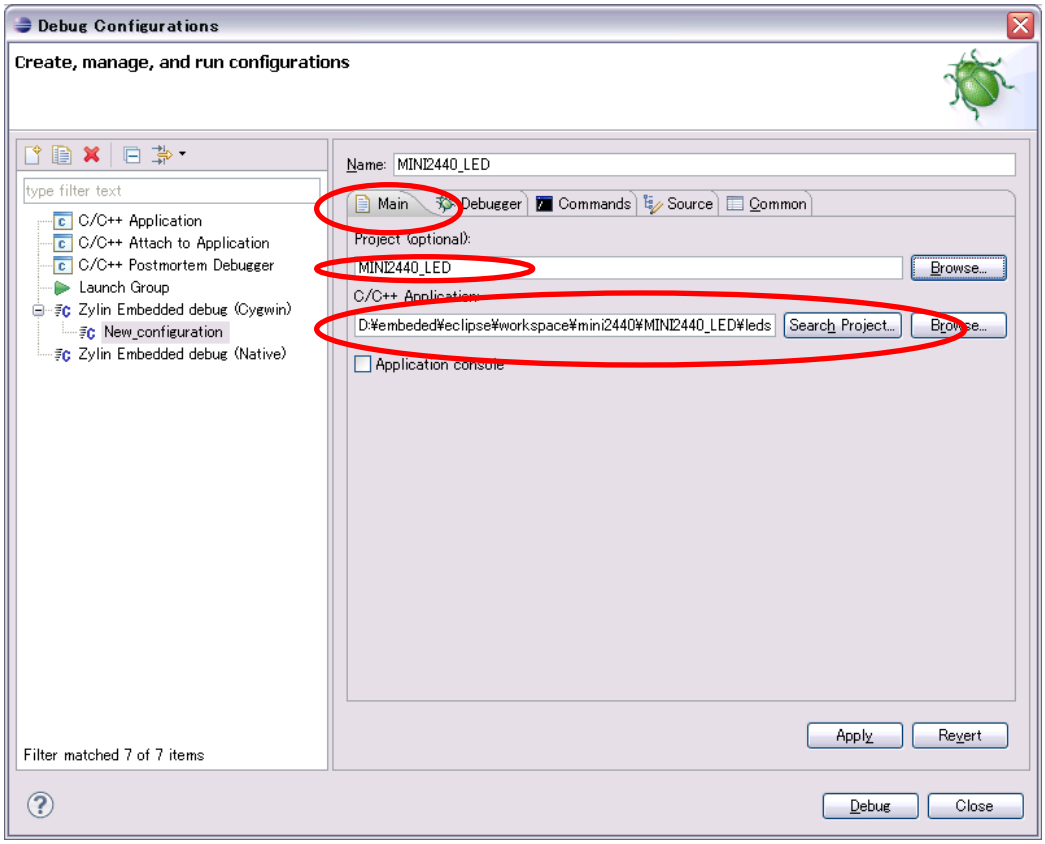
a)デバッグ設定画面を呼び出す



b)「Zylin Embedded debug(Cygwin)」を選べ、右にクリックしてから「New」を押下



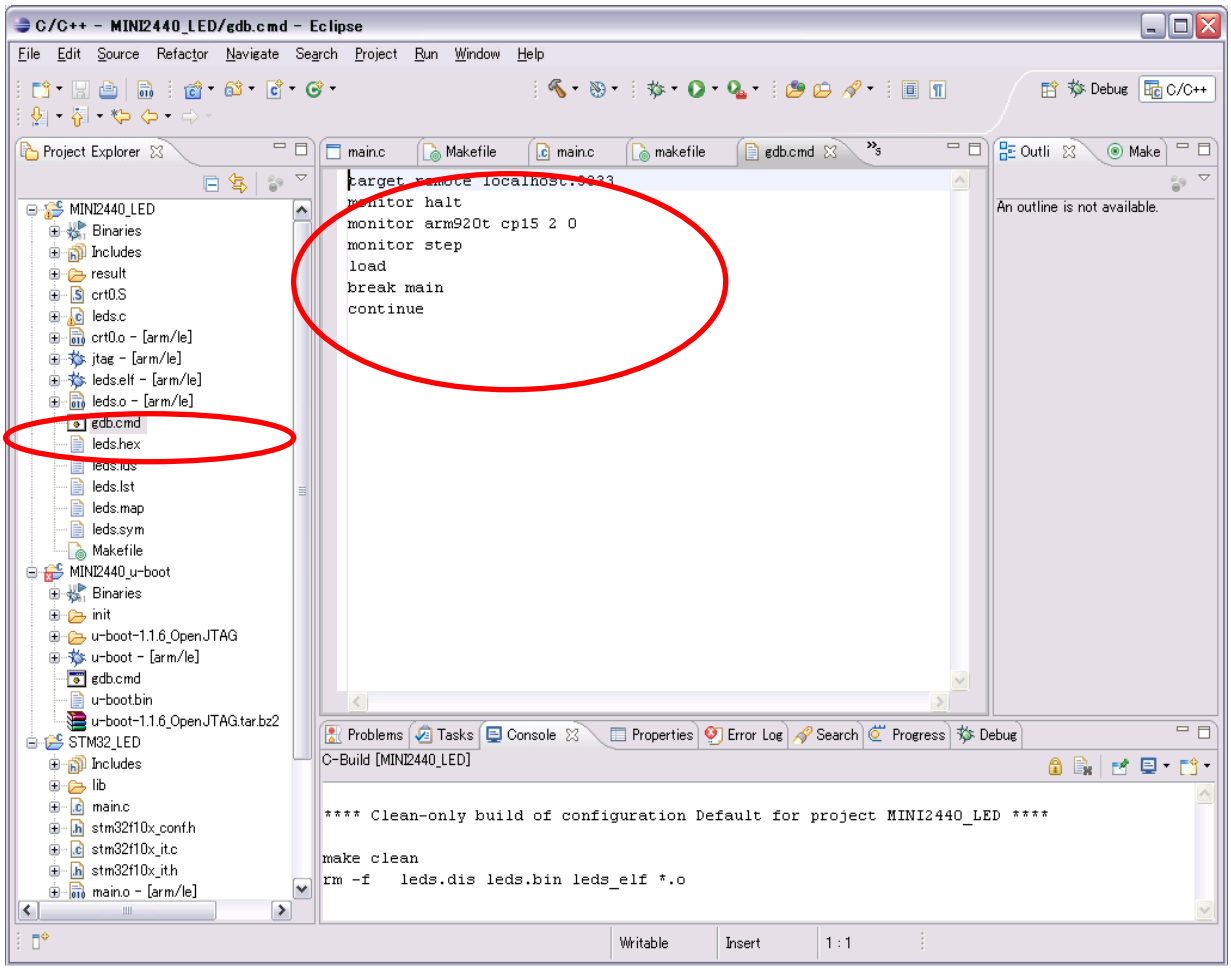
c)デバッグ設定画面の中、「Main」タブ項目を設定



d)「Commonds」タブ項目を設定(//の後にはコマンドのコメントとなり、設定内容に入れない)

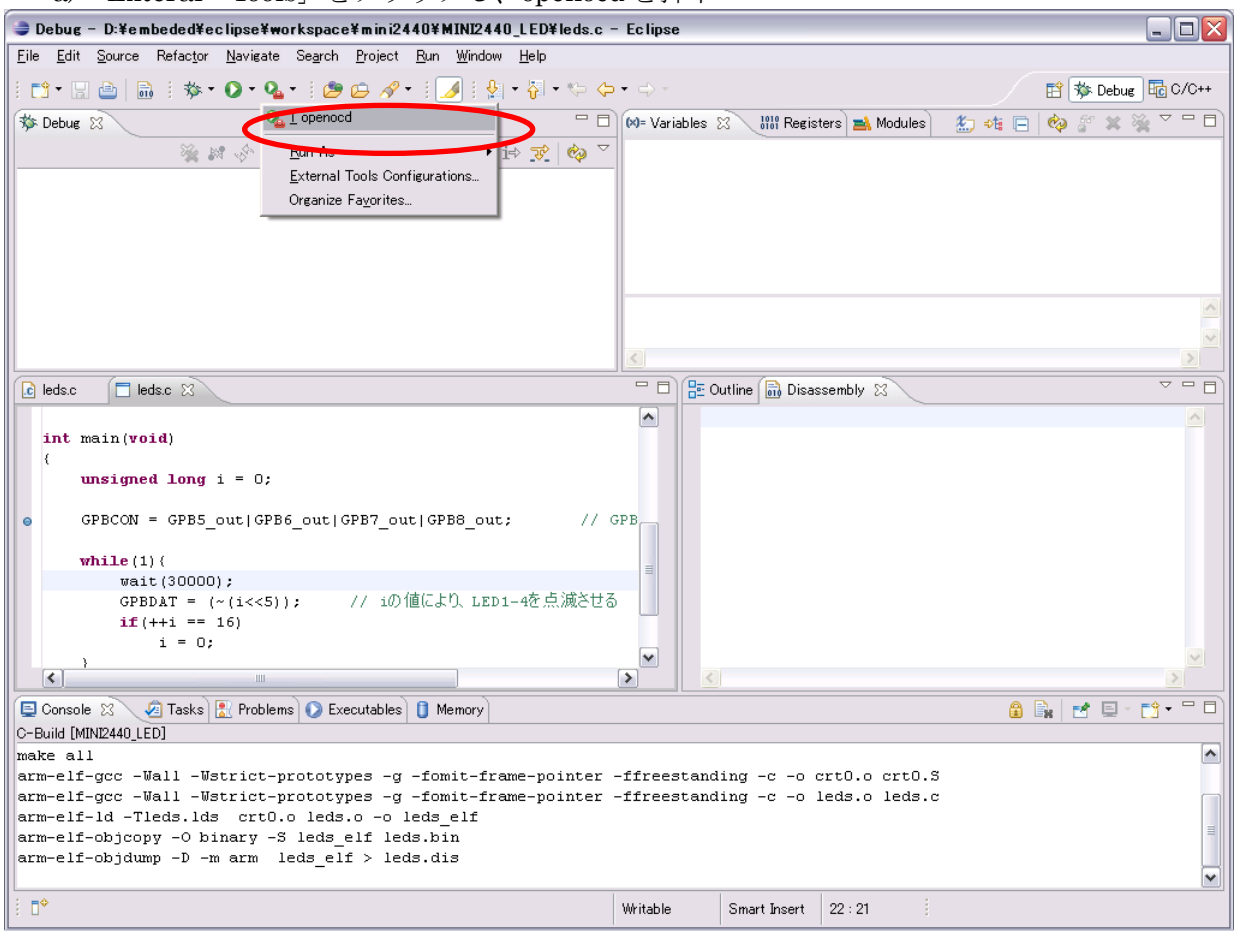
```
target remote localhost:3333 // ローカルポート「3333」と接続（OpenOCDと接続）
monitor halt//ボードの実行を中断させる
monitor arm920t cp15 2 0 // MMU機能をクローズ
monitor step//ステップで実行するように
load // leds_elfをロード，「elf」というフォーマットのファイルにアドレスを含まれる
break main //  「main」関数にブレークポイントを設定
continue // プログラムを実行させて、「main」にとまってステップでデバッグ可能
```

「MINI2440_LED」プロジェクトの配下、「gdb.cmd」ファイルの内容をそのままコピーしても OK
「gdb.cmd」を右クリック、「Open With」→「Text Editor」
＊ほかの設定を保存するため、コピー途中で「Apply」ボタンを押して適用します。

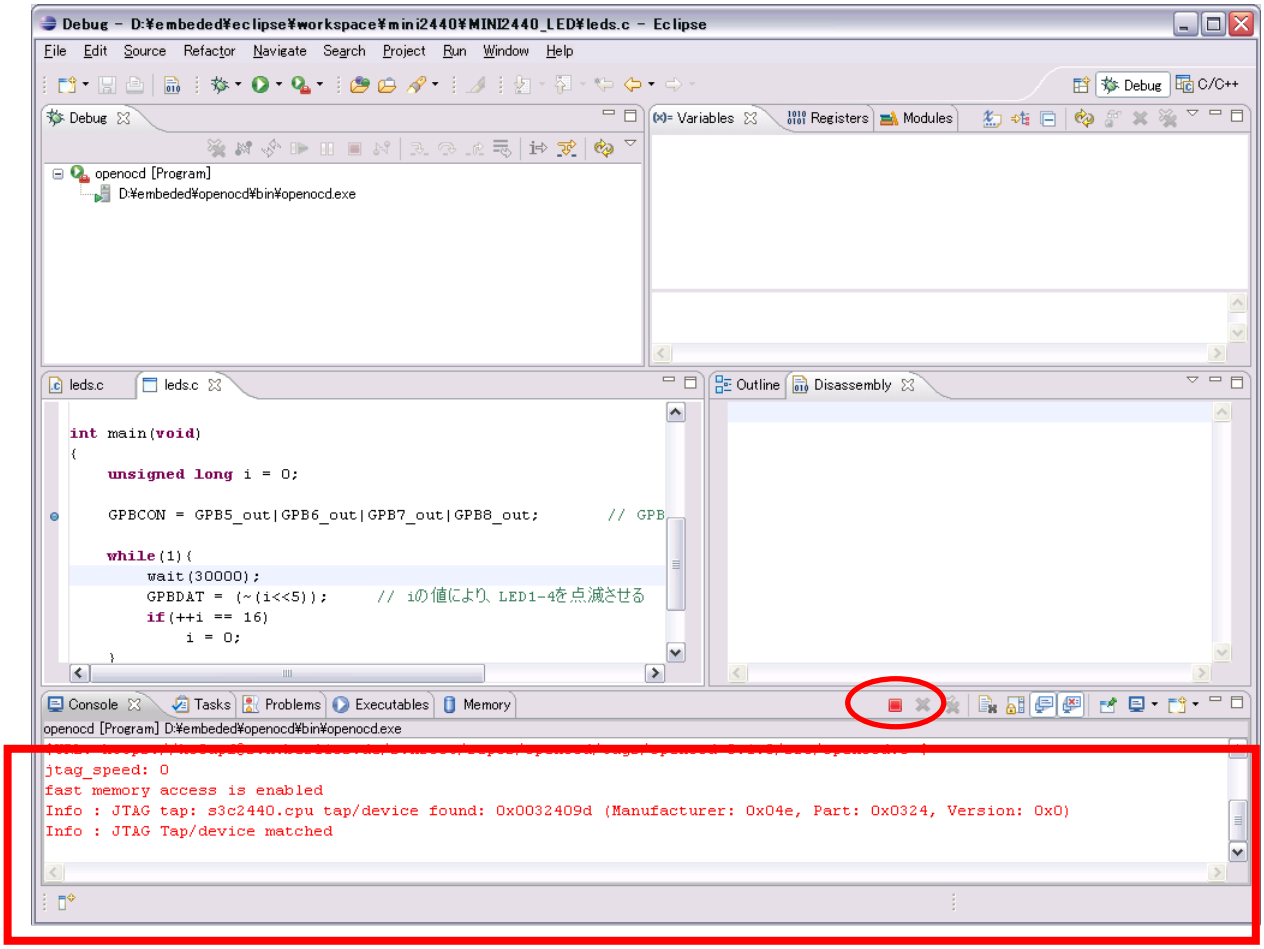


③ OpenOCD を起動させる

a)「Exteral　Tools」をクリックし、openocd を押下

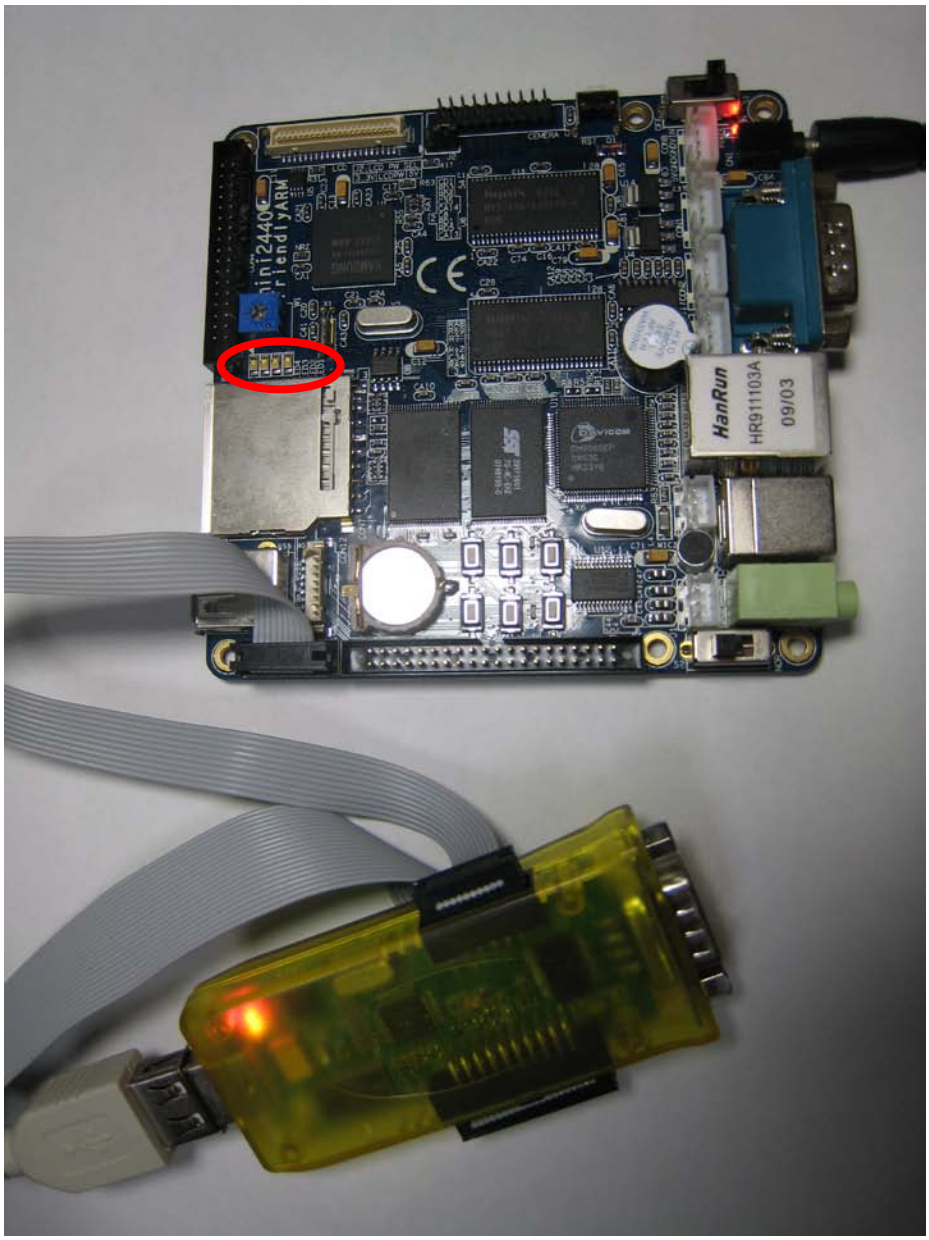


b)成功に起動したことを確認

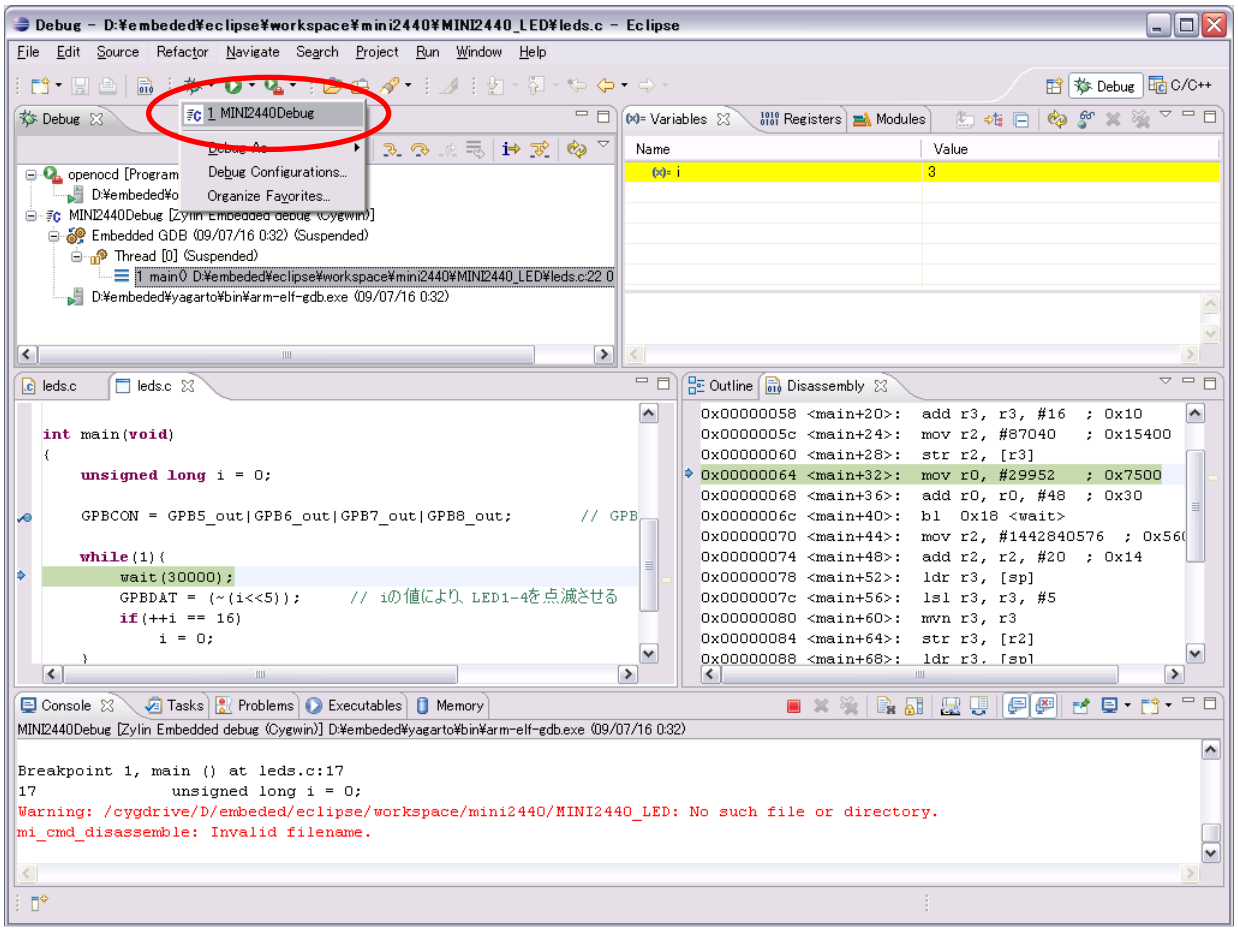


④ Debug　ツール「gdb」が起動前、ARM9 ボードの様子

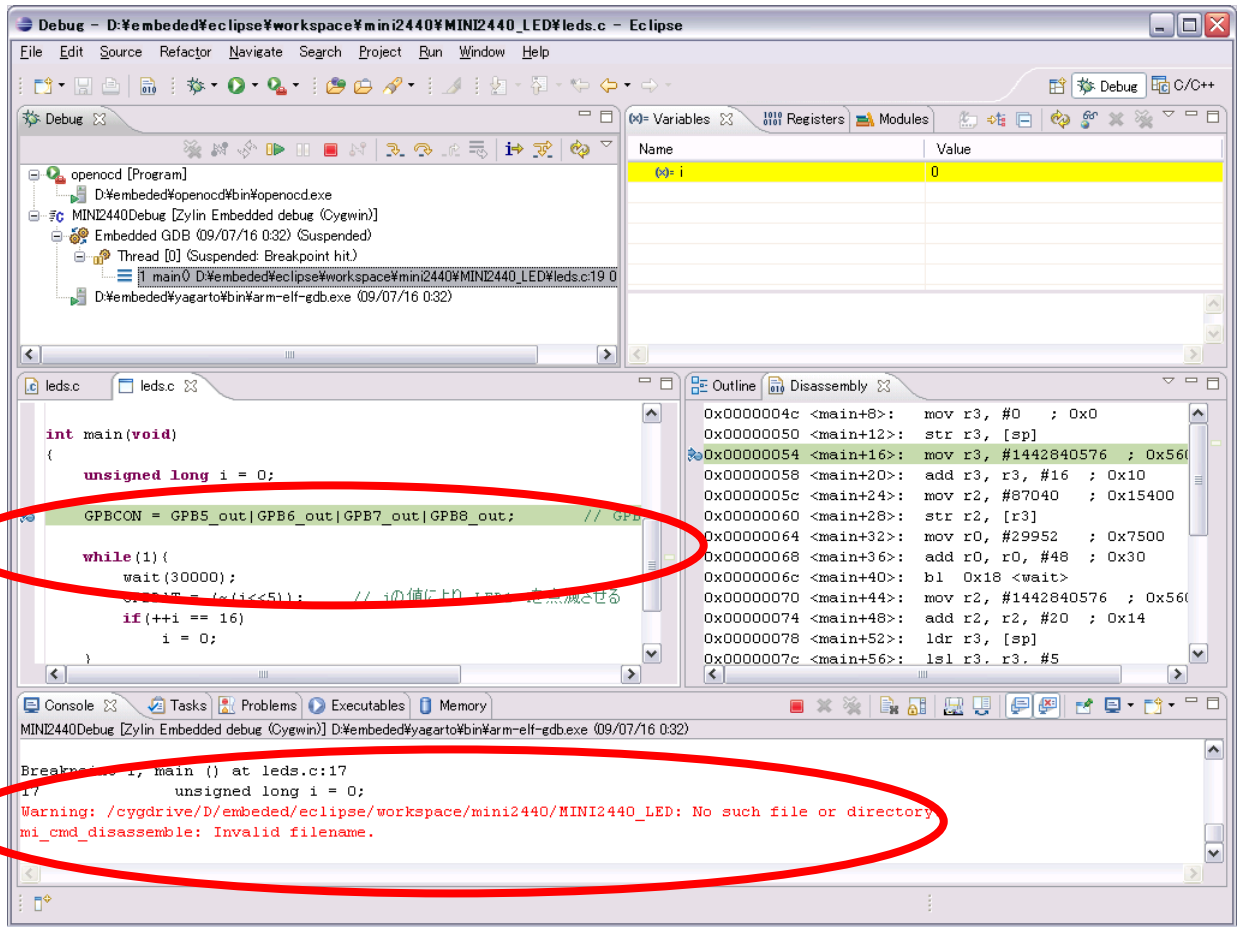
a）4LED が全て消している



⑤ Debug　ツール「gdb」を起動させる



⑥ ウインドに関数「main」の一番最初に止まっている

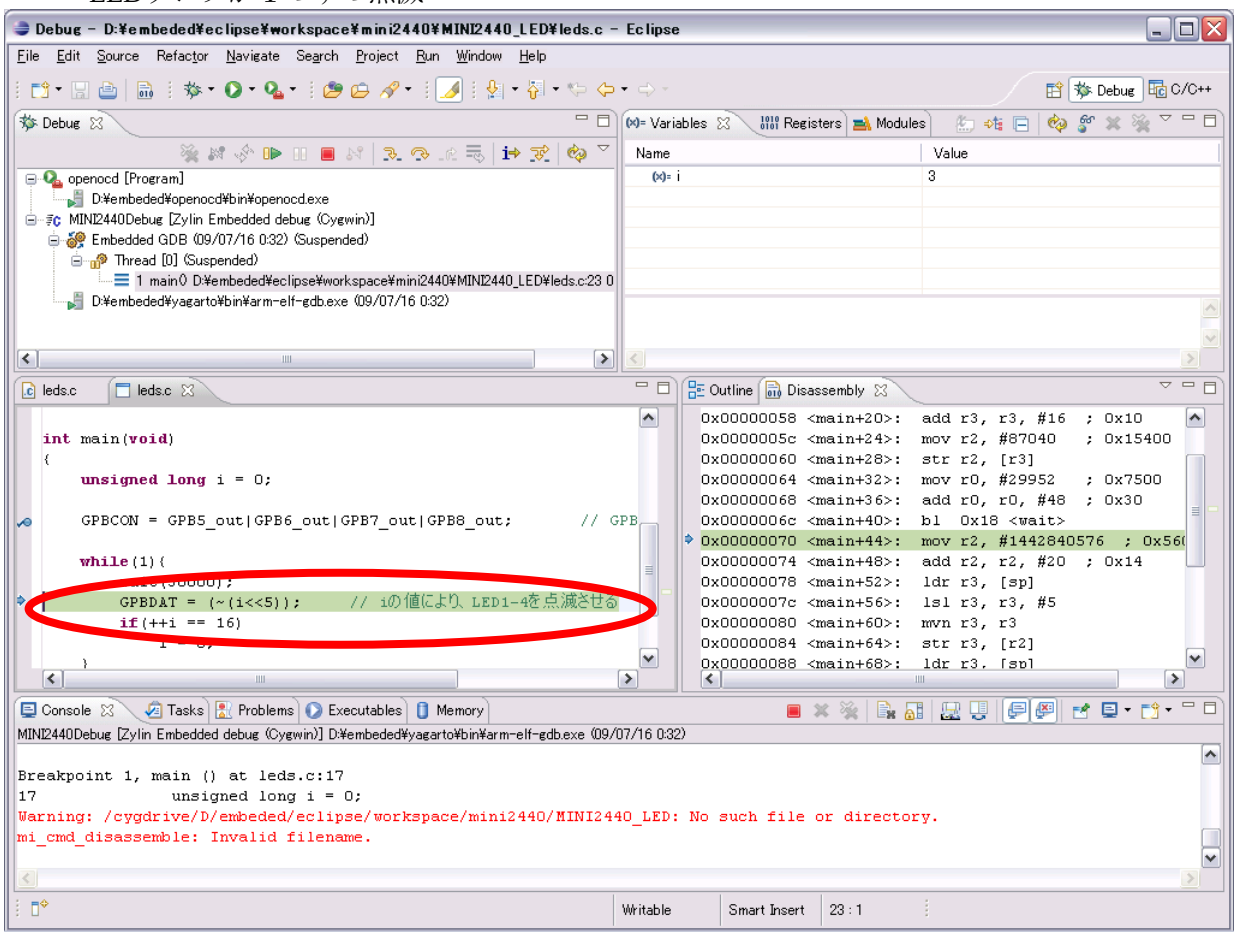


⑦ ステップ実行して Debug（ショットカットキーF6）

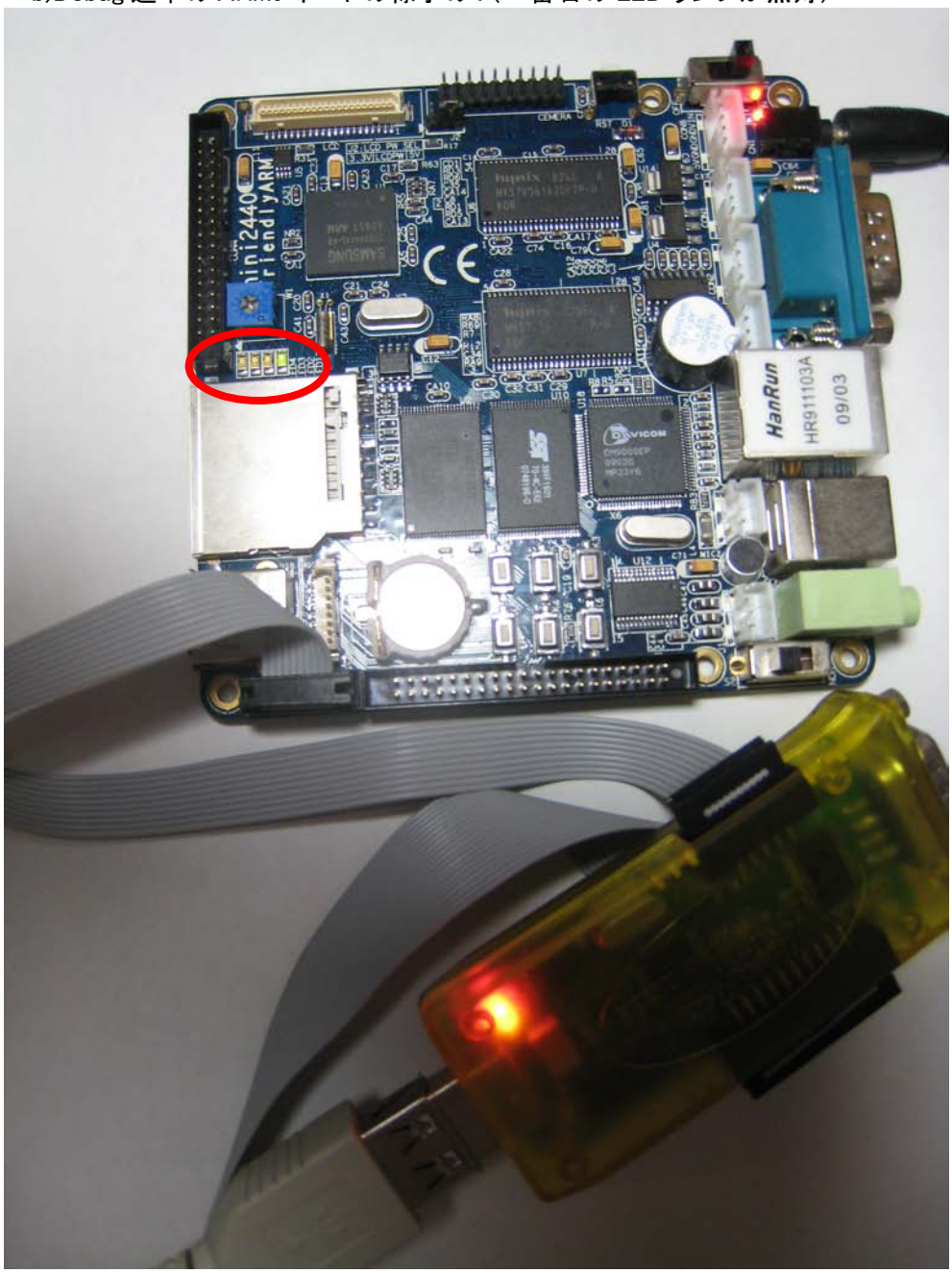
a) ブレークポイントでプログラムが中断した状態から、次のブレークポイントまで実行させたり、1行ずつ実行させたりできます。

コード「GPBDAT = (~(i<<5))」を繰り返して実行することにより、

LEDランプが１つずつ点滅

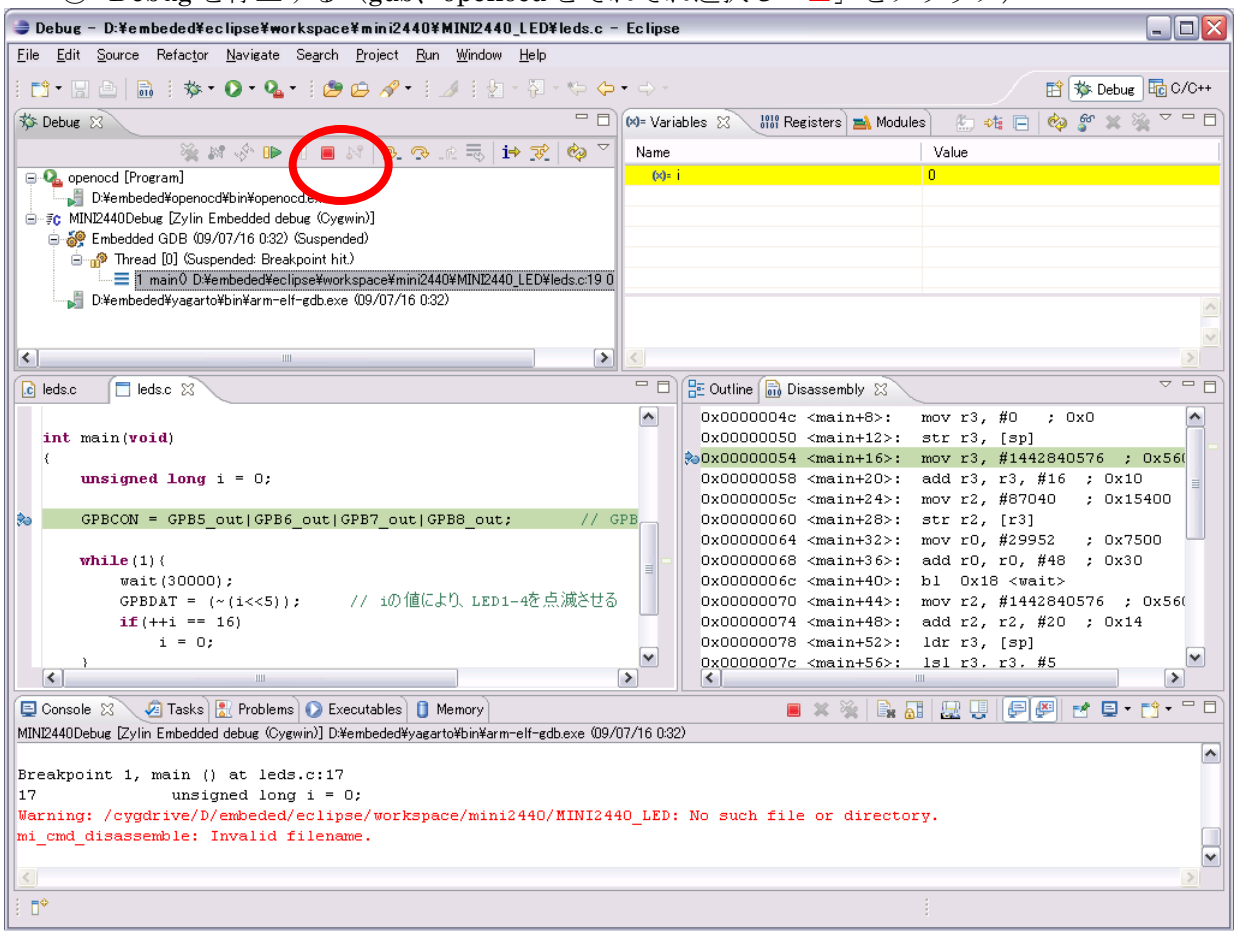


b)Debug 途中の ARM9 ボードの様子の1（一番右の LED ランプが点灯）



c)Debug 途中の ARM9 ボードの様子の2（右から 2 番目の LED ランプが点灯）

⑧ Debug を停止する（gdb、openocd をそれぞれ選択し「■」をクリック）



6.4.4 u-boot サンプルデバッグ

１．サンプルの導入

①プロジェクト「MINI2440_u-boot」を作る

作成方法は「4.4.3 LED サンプルデバッグ」の「①プロジェクト「MINI2440_LED」を作る」を参照

メモ：

a)u-boot のソースコードを Windows 上にコンパイルできないので、コンパイル後のファイルを提供

b)自分のソースをコンパイル場合、下記の 2 点修正が必要

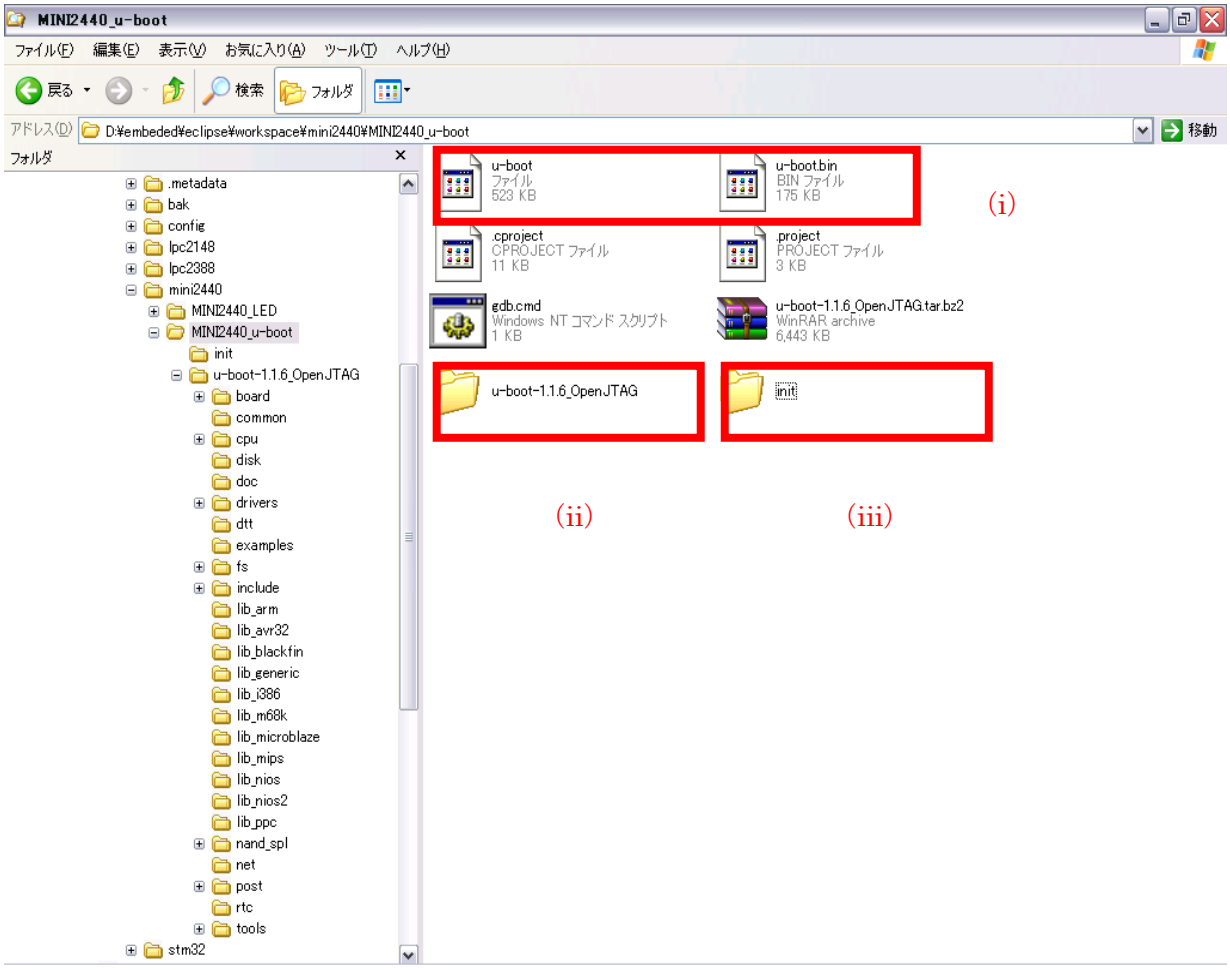
```
#define CONFIG_SKIP_LOWLEVEL_INIT 1 /* SDRAMの初期化を行わない */
#define CONFIG_SKIP_RELOCATE_UBOOT 1 /*実行アドレスをロードされるため、再配置必要なし*/
```

作成されたプロジェクトフォルダー構成：

（ア）コンパイル後のファイル：u-boot は elf ファイルとなる

（イ）デバッグ用の u-boot ソース

（ウ）「init」初期化用のファイル（u-boot は SDRAM に実行されるため、初期化必要）

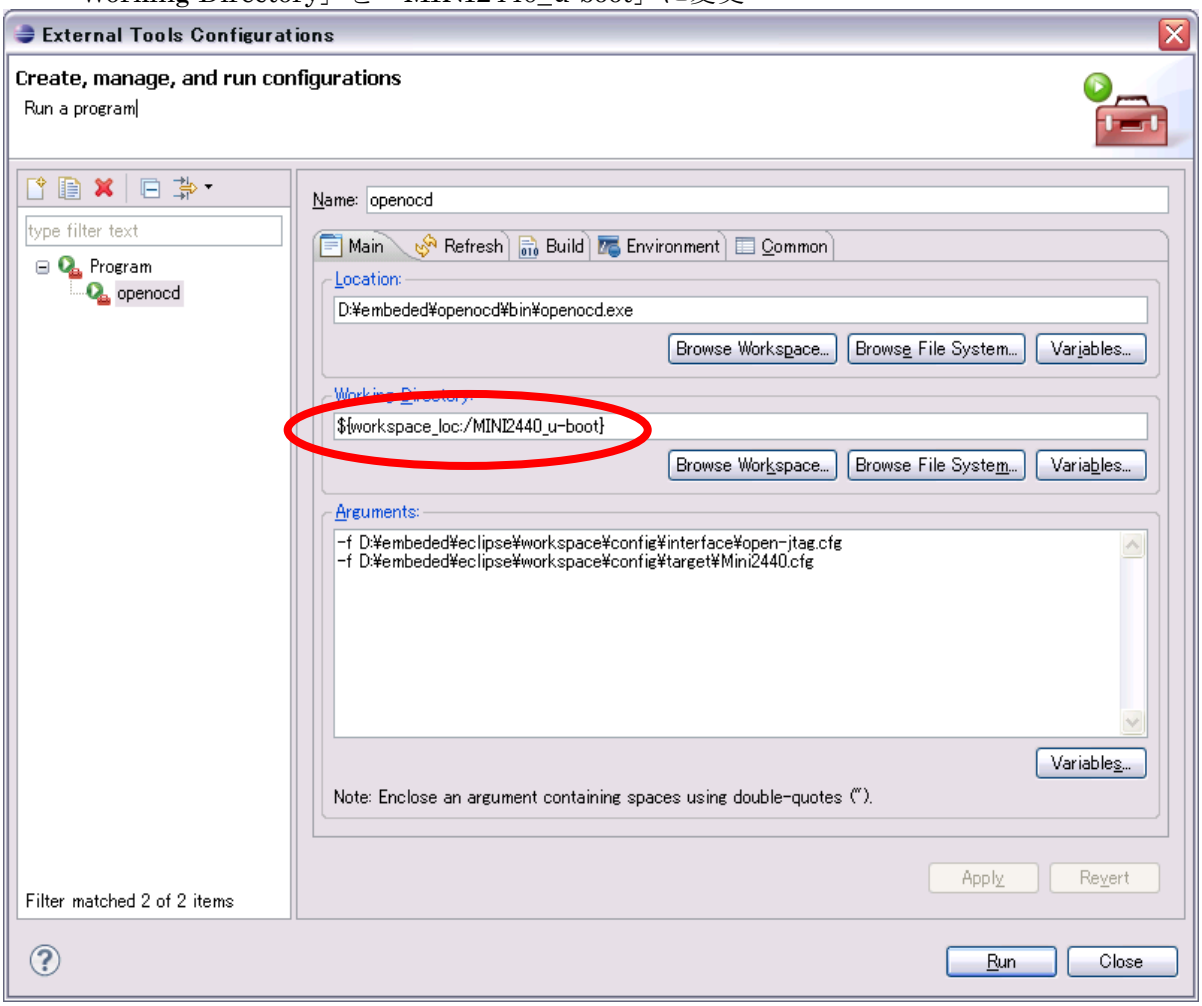


２．サンプル MINI2440_u-boot をデバッグ

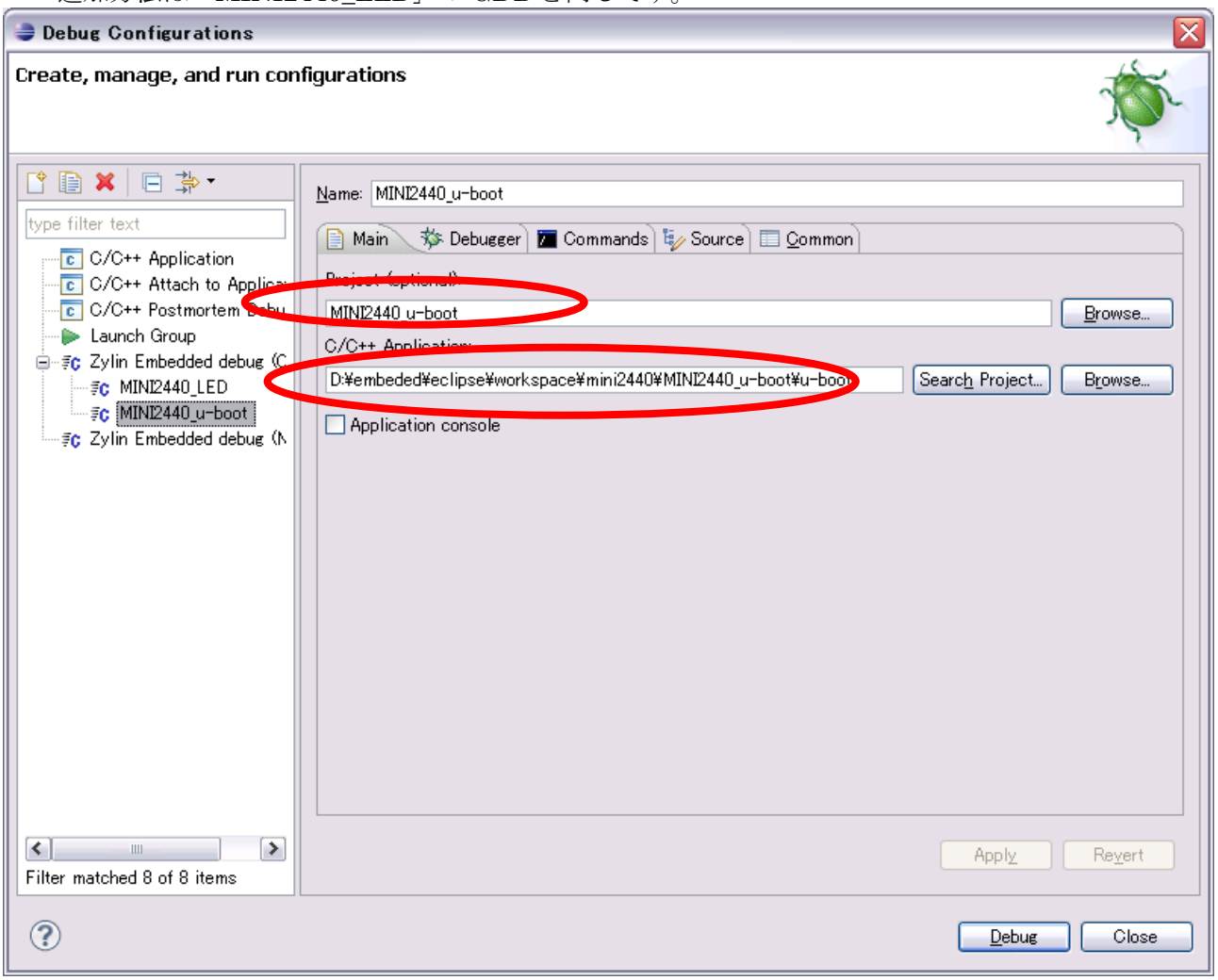
他の設定は MINI2440_LED と同じですので、違い設定のみを説明

① openocd 設定

「Working Directory」を「MINI2440_u-boot」に変更

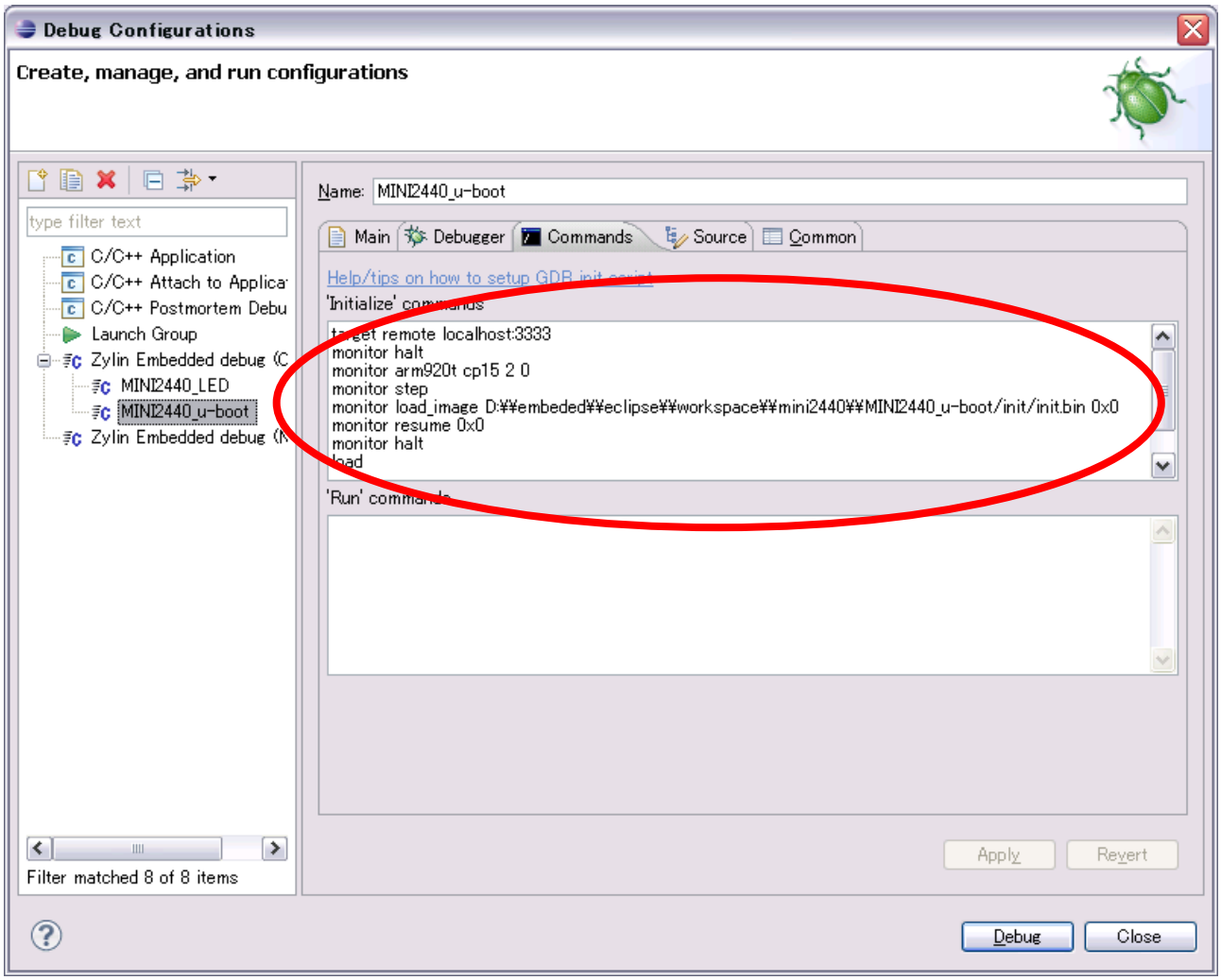


② MINI2440_u-boot 用の GDB を追加
＊追加方法は「MINI2440_LED」の GDB と同じです。



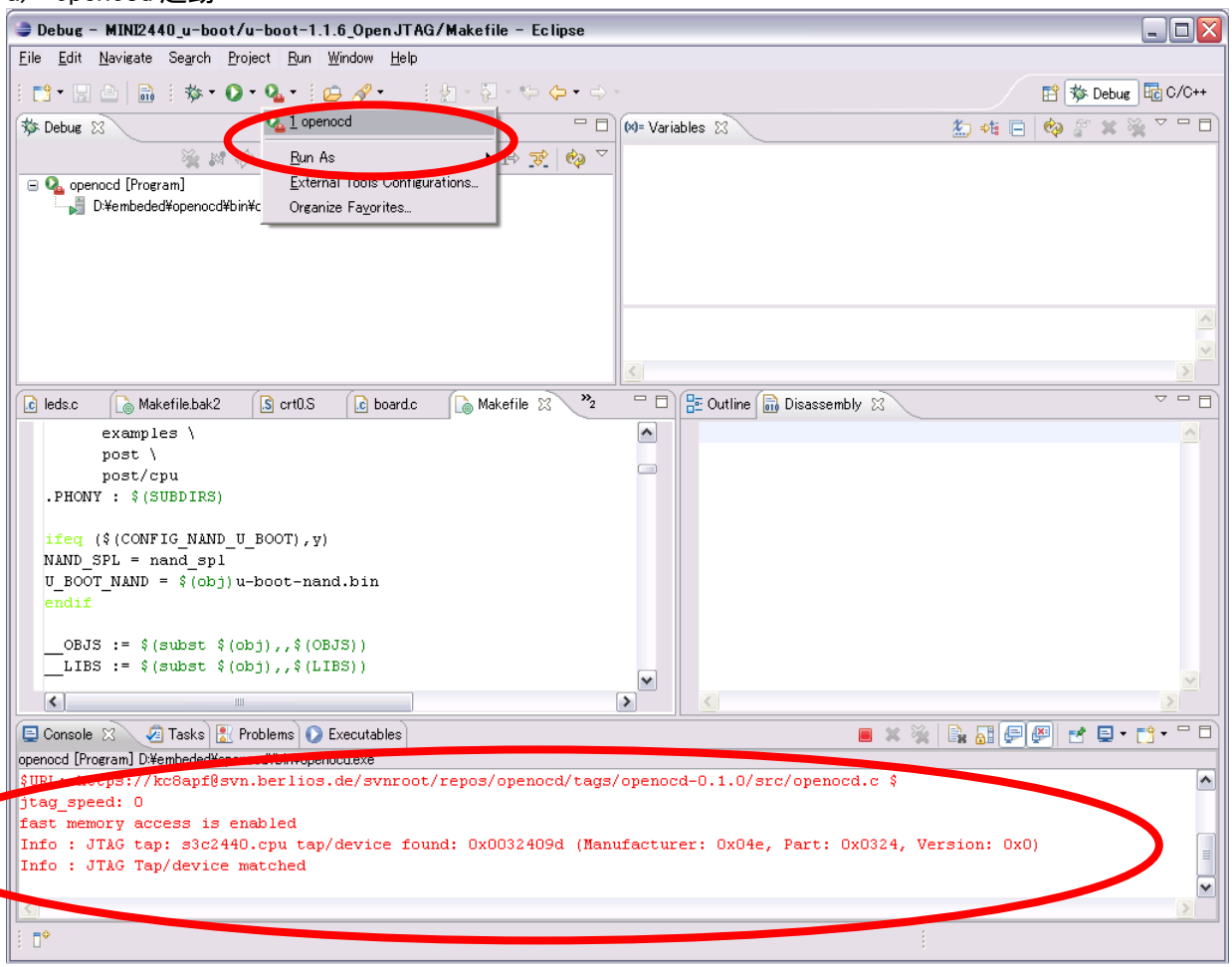
③デバッグ用のコマンドを設定(//はコメント内容となり、実際の設定に入れないでください。)

target remote localhost:3333 // ローカルポート「3333」と接続 (OpenOCD と接続)
monitor halt//ボードの実行を中断させる
monitor arm920t cp15 2 0 // MMU 機能をクローズ
monitor step//ステップで実行するように設定
monitor load_image D:¥¥embeded¥¥eclipse¥¥workspace¥¥mini2440¥¥MINI2440_u-boot/init/init.bin 0x0
//u-boot は SDRAM に実行されるため、初期化実行モジュールをロード(実際の init パスを入力)
monitor resume 0x0//初期化モジュールを実施
monitor halt//ボードの実行を中断させる
load//u-boot をロード
break start_armboot//「start_armboot」関数にブレークポイントを設定
continue// プログラムを実行させて、「start_armboot」にとまってステップでデバッグ可能



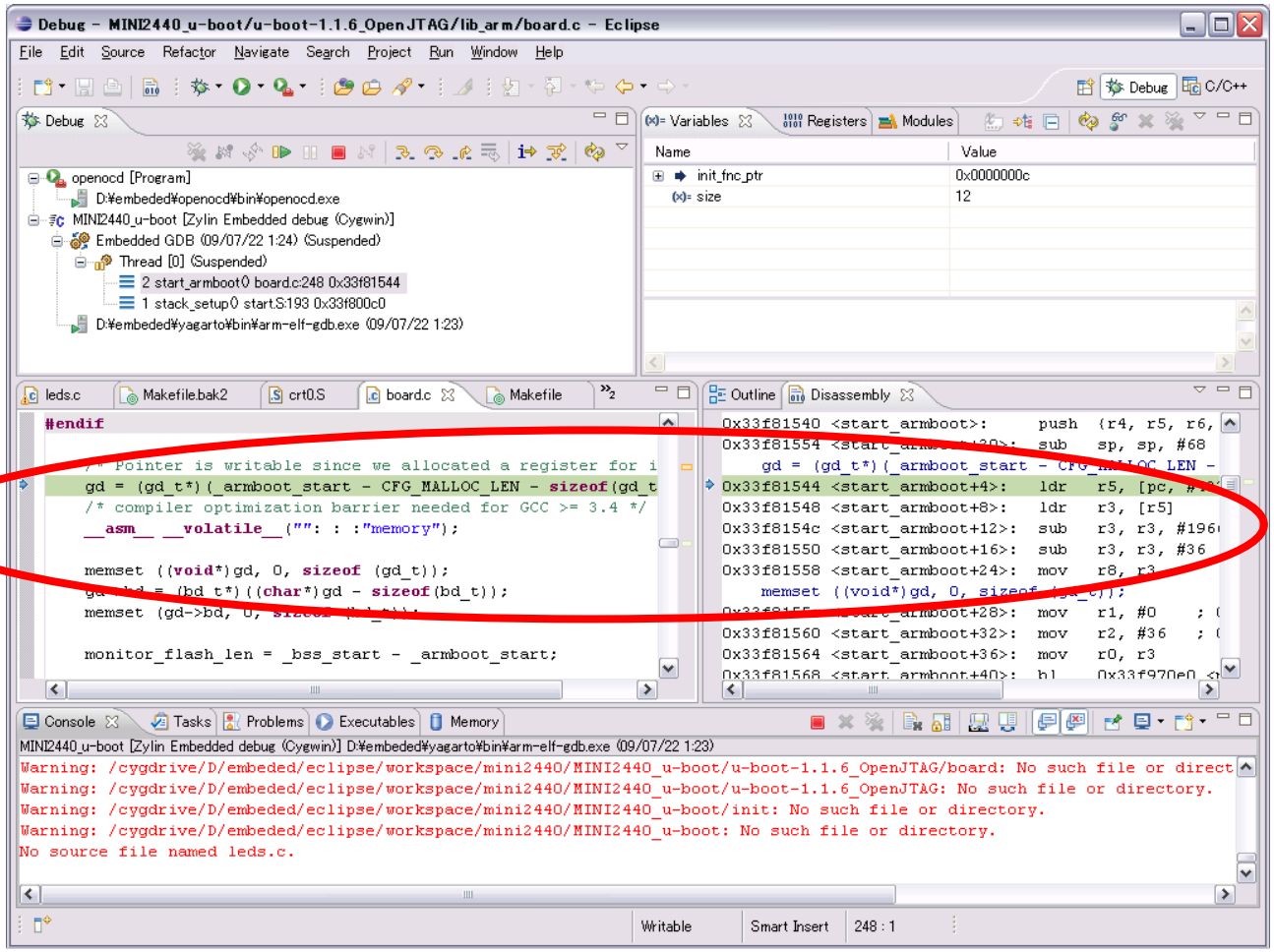
④デバッグ

a） openocd 起動

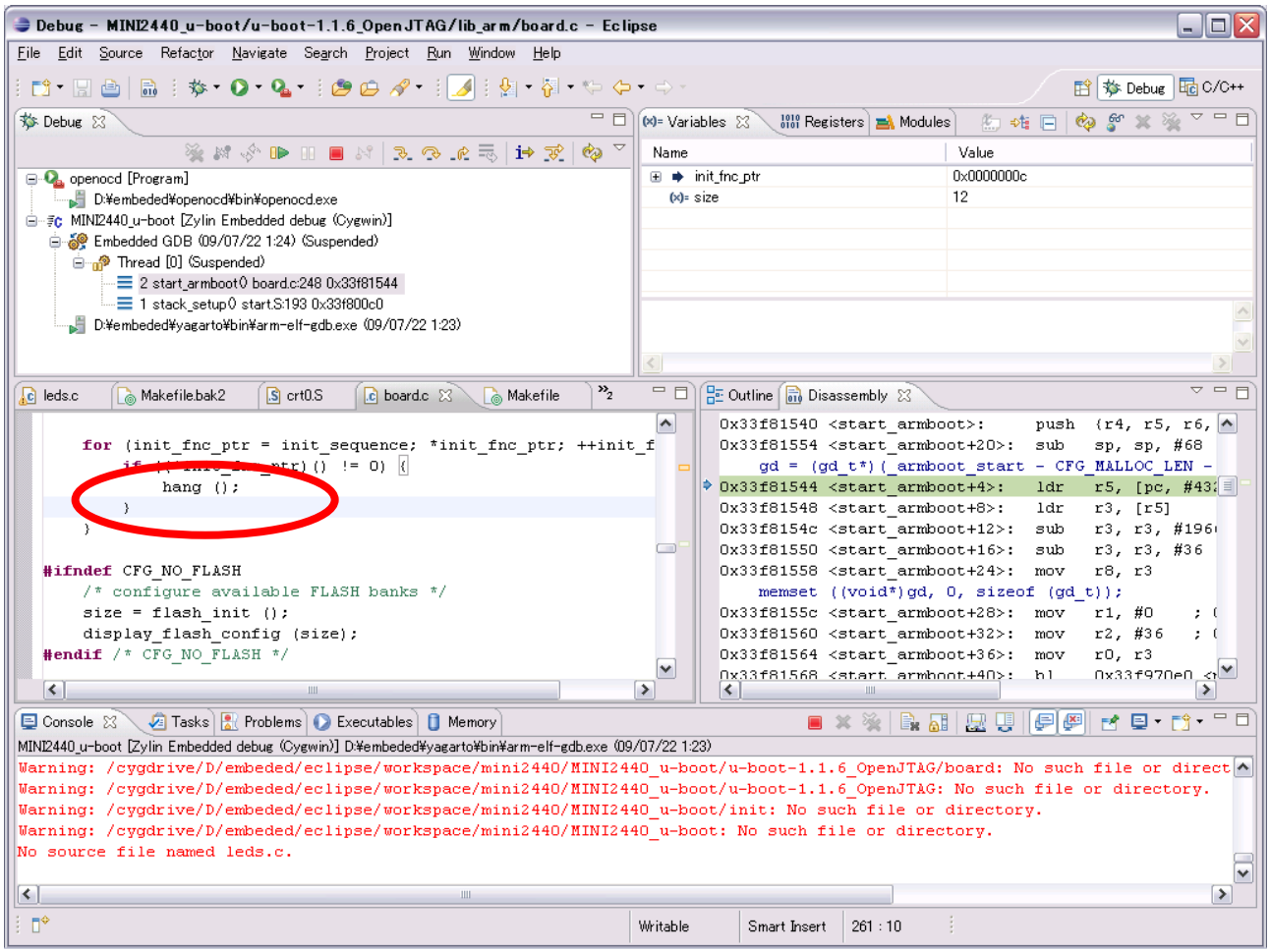


b） gdb 起動

関数「start_armboot」に止まり、ショットカットキー「F6」でステップでデバッグできる



c） ステップでデバッグ
コーディング「hang()」を実行したら、MINI2440 ボードからベルを鳴る



第七章 Linux 環境上の OpenJTAG の使用手順

7.1 ハードウエア、ソフトウエアインストール

もし OpenJTAG を USB からシリアルポート変換というツールのみとして使う場合、7.1.1 節のみを参照すれば OK です。

※Linux 関連リソースダウンロード URL：

http://www.dragonwake.com/download/open-jtag/Linux.zip

本マニュアルで使われる Linux ディストリビューションは Ubuntu となります。

仮に上記 URL からダウンロードしたファイルは「/tmp」に解凍します。

7.1.1 Linux で OpenJTAG の自動認識

■udev ルールを修正すれば、Ubuntu で OpenJTAG を認識後ドライバを自動ロードします。

Ubuntu7.10 の場合、下記圧縮ファイルの中「Linux¥install¥ 50-ftdi.rules」を「/etc/udev/rules.d/」にコピー

```
$sudo cp /tmp/Linux/install/50-ftdi.rules /etc/udev/rules.d/
```

udev ルールをすぐに有効するため、下記のコマンドを発行してください。（次回起動後、実行必要ない）

```
$sudo udevcontrol reload_rules
```

Ubuntu8.10 以降のバージョンであれば、上記作業が必要ありません。

OpenJTAG を PC の USB と接続すれば、Ubuntu は自動認識してドライバをロードします。「/dev」フォルダデバイス「ttyUSB0」を自動生成されます。古い Ubuntu バージョンの場合、ttyUSB1 を生成されたかも知れません。

```
$ ls /dev/ttyUSB* -l
crw-rw---- 1 root dialout 188, 0 2011-09-28 13:25 /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 1 2011-09-28 13:25 /dev/ttyUSB1
```

「/dev/ttyUSB1」があれば、「/dev/ttyUSB1」を普通のシリアルポートとして使えます、ttyUSB0 のみがある場合、/dev/ttyUSB0」を普通のシリアルポートとして使えます。

Ubuntu の中、 kermit あるいは minicom ツールでシリアルポートを操作できます。下記のコマンドでこの二つツールをインストールできます。

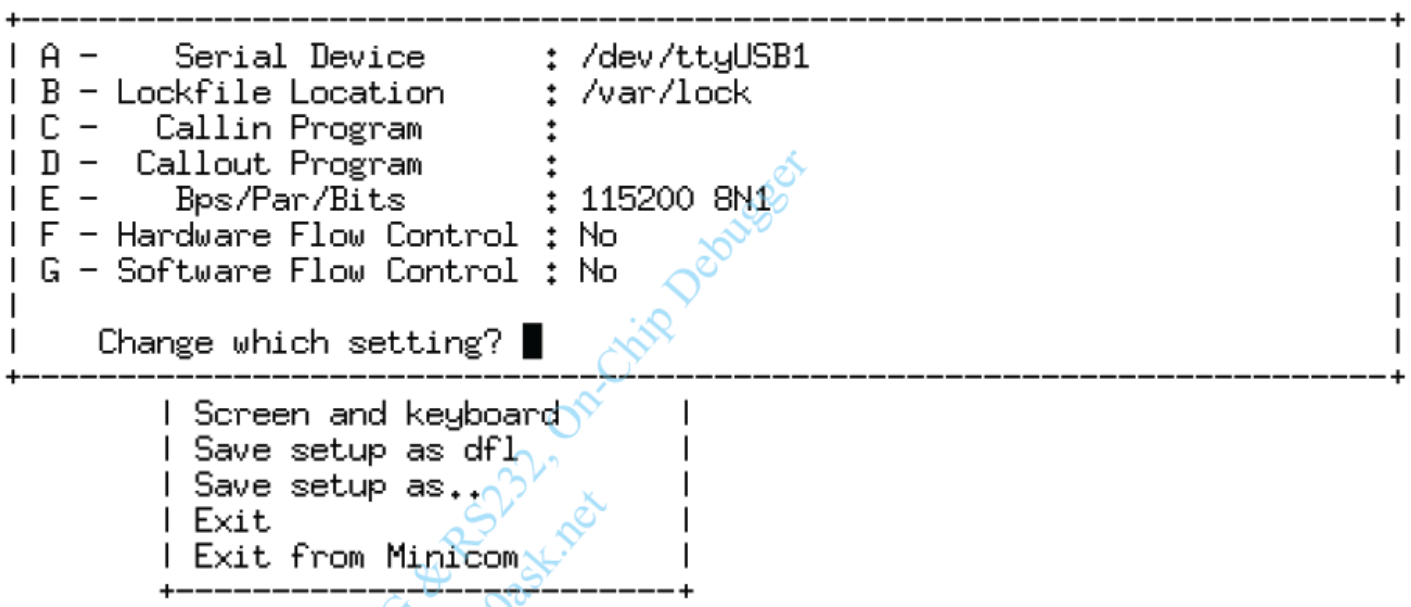
```
$ sudo apt-get install ckermit
$ sudo apt-get install minicom
```

kermit ツールを使用前、「/home/csun」（仮にユーザー名が csun）フォルダーに名前が「.kermrc」のファイルを作成します。内容は以下通りです。（/dev/ttyUSB1 がない場合、/dev/ttyUSB0 に変更してください。）

```
set line /dev/ttyUSB1
set speed 115200
set carrier-watch off
set handshake none
set flow-control none
robust
set file type bin
set file name lit
set rec pack 1000
set send pack 1000
set window 5
```

“$ sudo kermit -c”コマンドでシリアルポート操作画面を起動できます。シリアルポートを閉じたい場合、“Ctrl”と“¥”を同時押し、その後、“Ctrl”を押さなくて、“exit”を入力して Enter キ―を押します。

minicom を使う場合、使用前に“minicom -s”コマンドで設定画面に入ります、“Serial port setup”を選べ、下図のように設定を行いましょう。その後、“Save setup as dfl”を選択（/dev/ttyUSB1 がない場合、/dev/ttyUSB0 に変更）



minicom 設定画面

上記設定を行った後、直接 minicom コマンドを発行すれば、シリアルポート操作画面を起動できます。操作画面んを閉じたい場合、“Ctrl”と“A”を同時押し、その後、離して“X”を押します。

7.1.2 OpenOCD、GDB、クロスコンパイルチェイン、Eclipse のインストール

1.OpenOCD、GDB、oflash のインストール

Linux¥install フォルダの直下の openocd.tar.bz2 を/に解凍して、arm-linux-gdb、oflash を/usr/bin にコピーしてください。

そして、所属のユーザーを root に設定、実行可能の属性も追加：

```
$ sudo tar xjf /tmp/Linux/install/openocd.tar.bz2 -C /
$ sudo cp /tmp/Linux/install/arm-linux-gdb /usr/bin/
$ sudo cp /tmp/Linux/tools/oflash /usr/bin/
$ sudo chown root:root /usr/local/bin/openocd /usr/bin/arm-linux-gdb /usr/bin/oflash
$ sudo chmod +xs /usr/local/bin/openocd /usr/bin/arm-linux-gdb /usr/bin/oflash
```

注意：必ず/usr/bin/arm-linux-gdb を実行権限”x”、スーパーユーザー権限"s"を追加

2. クロスコンパイルチェインのインストール

もし、Ubuntu 上にクロスコンパイルチェインをインストールされない場合（”arm-linux-gcc -v”コマンドで確認、出力内容がない場合、インストールされないという事です。）

Linux/install/arm-linux-gcc-3.4.5-glibc-2.3.6.tar.bz2 をあるフォルダに解凍してから PATH 変数を設定。

例：

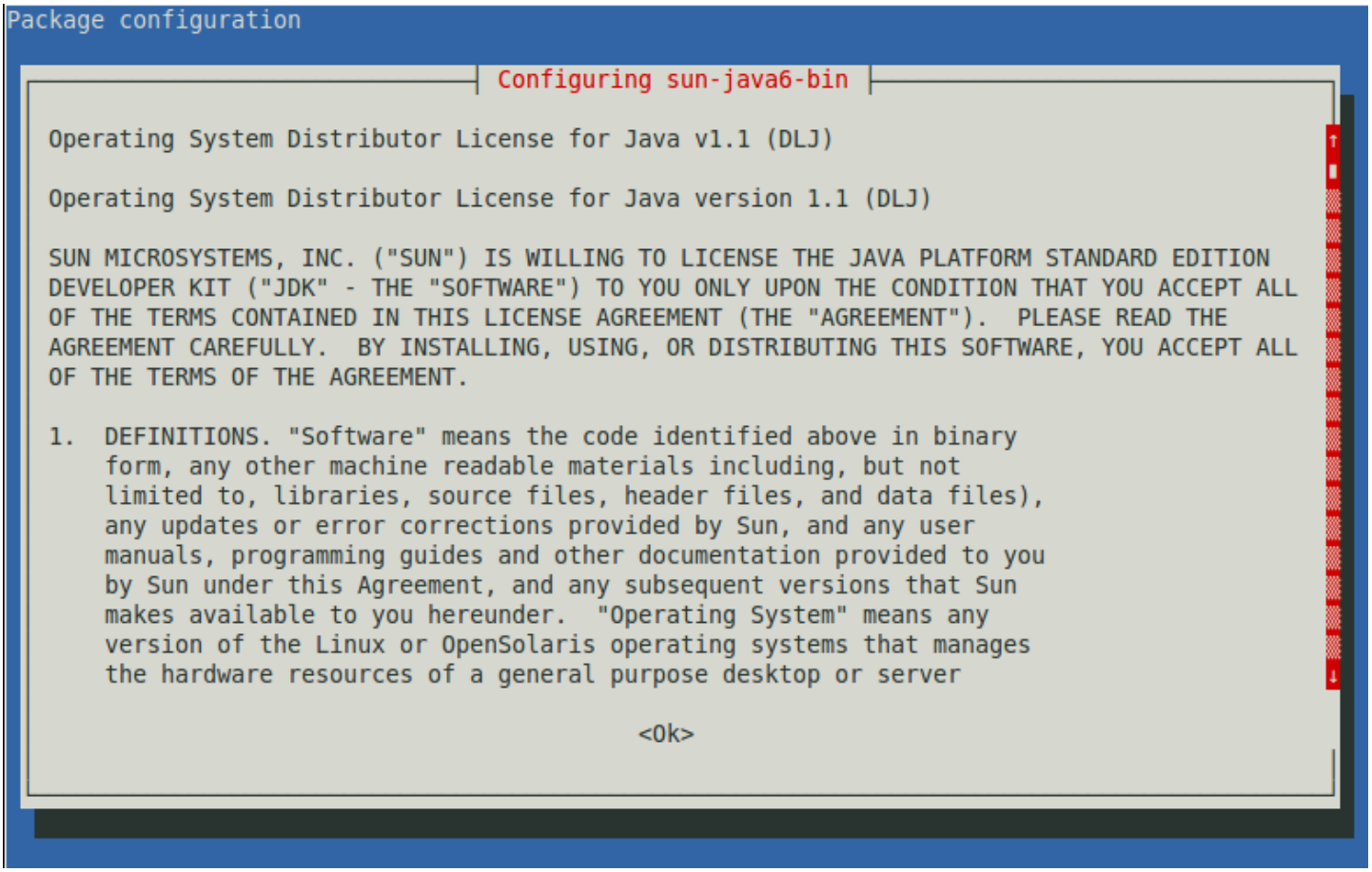
```
$ cp /tmp/Linux/install/arm-linux-gcc-3.4.5-glibc-2.3.6.tar.bz2 /work/tools
$ cd /work/tools
$ tar xjf arm-linux-gcc-3.4.5-glibc-2.3.6.tar.bz2
```

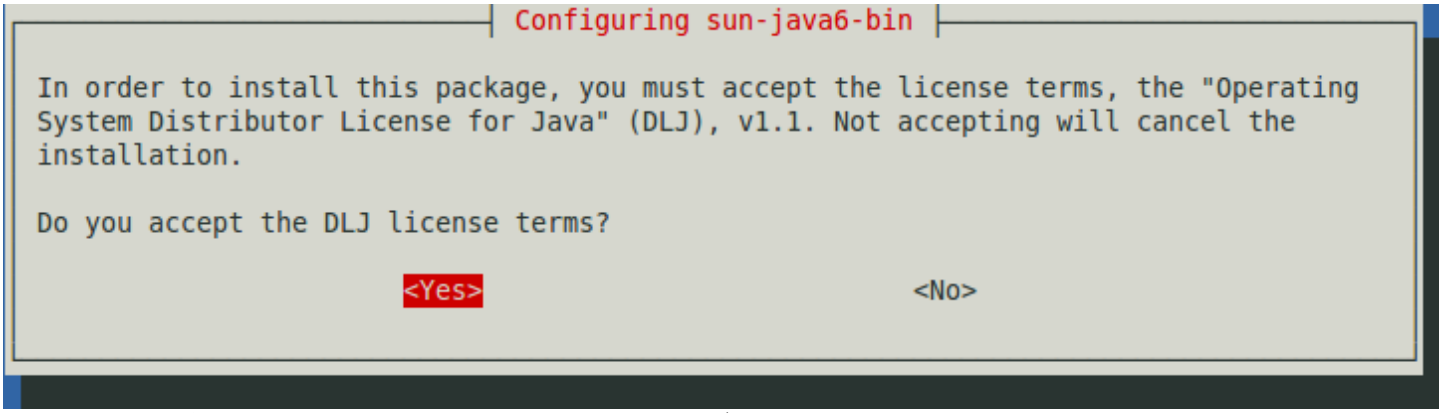
「/etc/environment」を編集して PATH 環境変数を修正しましょう。（赤字が追加）

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/work/tools/gcc-3.4.5-gl
```

ibc-2.3.6/bin"
次回起動後、直接クロスコンパイルチェインを使えます、すぐ使いたい場合、下記コマンドで発行します。
$ export PATH=$PATH:/work/tools/gcc-3.4.5-glibc-2.3.6/bin

2. Eclipse のインストール
①JAVA のインストール（以前の SUN、今の Oracle 社）
Ubuntu 上に既に GNU バージョンの Java をインストールされますが、
このバージョンの Java は機能上 SUN Java より弱いですので、Eclipse のあるプラグインは必ず SUN Java 上に使えますので、下記のコマンド SUN Java をインストールしましょう。
$sudo apt-get install sun-java6-jre
$sudo apt-get install sun-java6-jdk
注意:リモートツールで上記コマンドを実行しないでください。インストール途中で以下の二つ画面が出てきます、リモートツールで GUI 画面をサポートしないです。
上記一番のコマンドを実行する時、下記画面が出てきます、Ok、Yes を選択してください。





以下のコマンドで上記インストールされた sun-java6 をデフォルトの Java に設定しましょう。

```
$sudo update-alternatives --config java
book@book-desktop:/var/lib/dpkg/info$ sudo update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority   Status
------------------------------------------------------------
* 0            /usr/lib/jvm/java-6-openjdk/jre/bin/java   1061      auto mode
  1            /usr/lib/jvm/java-6-openjdk/jre/bin/java   1061      manual mode
  2            /usr/lib/jvm/java-6-sun/jre/bin/java       63        manual mode

Press enter to keep the current choice[*], or type selection number: 2
```

注意：Ubuntu 上に Java をインストールした事がない場合、下記の様なメッセージがでてきます。

```
There is only 1 program which provides java
(/usr/lib/jvm/java-6-sun/jre/bin/java). Nothing to configure.
```

②Eclipse のインストール

/tmp/Linux/install/eclipse-cpp-helios-SR1-linux-gtk.tar.gz を/opt に解凍します、そして、属性も下記のように変更してください。

```
$ cd /tmp/Linux/install
$ sudo tar xzf eclipse-cpp-helios-SR1-linux-gtk.tar.gz -C /opt
$ cd /opt
$ sudo chmod 775 eclipse -R
```

スクリプトファイル「/tmp/Linux/install/eclipse」を「/usr/bin」にコピーし、実行可能の属性も追加します。

```
$ sudo cp /tmp/Linux/install/eclipse /usr/bin
$ sudo chmod 775 /usr/bin/eclipse
```

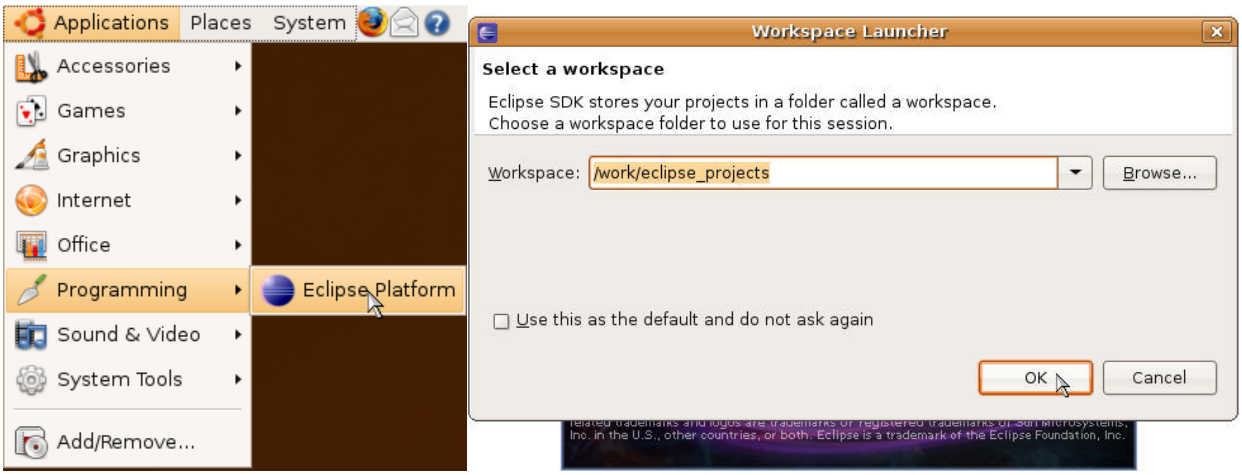
最後、メニューに Eclipse のショットカットも追加しましょう。

```
$ sudo cp /tmp/Linux/install/eclipse.desktop /usr/share/applications
$ sudo chmod +r /usr/share/applications/eclipse.desktop
```

インストール結果を確認

```
$ openocd -v
$ arm-linux-gdb -v
$ arm-linux-gcc -v
$ java -version
```

メニューから Eclipse を起動しましょう。



7.2 OpenJTAG の使用

環境を構築した後、使用方法は Windows と同じです、このマニュアルの Windows 部分をご参照ください。