本ドキュメントは Cypress (サイプレス) 製品に関する情報が記載されております。本ドキュメントには、「MB」から始まるシリーズ名，品名およびオーダ型格が記載されておりますが、これらはすべて「CY」から始まるシリーズ名，品名およびオーダ型格として、新規および既存のお客様に引き続き提供してまいります。

オーダ型格の調べ方について

1. www.cypress.com/pcnにアクセスしてください。
2. SEARCH PCNS フィールドに、オーダ型格などのキーワードを入力し、「Apply」をクリックしてください。
3. 該当するタイトル(Title)をクリックしてください。
4. 「Affected Parts List」ファイルを開いてください。
   当該ファイルに記載されている各種変更情報をご利用ください。

詳しいお問い合わせ先

Cypress 製品およびそのソリューションの詳細につきましては、お近くの営業所へお問い合わせください。

サイプレスについて

サイプレスは、世界で最も革新的な車載や産業機器、スマート家電、民生機器および医療機器製品向けに、最先端の組み込みシステム ソリューションを提供するリーディング カンパニーです。サイプレスのマイクロコントローラーや、アナログ IC、ワイヤレスおよび USB ベースのコネクティビティ ソリューション、高い信頼性と高性能を提供するメモリ製品は、各種機器メーカーの差異化製品の開発と早期市場参入を支援します。サイプレスは、ベストクラスのサポートと開発リソースをグローバルに提供することで、彼らが従来市場を破壊しまったく新しい製品カテゴリを歴史的なスピードで市場投入できるよう支援します。詳細はサイプレスのウェブサイト (japan.cypress.com) をご覧ください。

AN204421

FM3 GNU ツールチェーンを使用した開発環境構築方法

関連製品ファミリ: セクション 2 を参照

本書は、FM3 ファミリの開発環境を、Eclipse をベースとした GNU ツールチェーンで構築するための方法を説明します。

Contents

1 イントロダクション 2
1.1 概要 2
2 対象製品 2
2.1 JTAG インタフェース 5
2.2 J-Link 6
2.3 ARM-USB-TINY 7
3 コンパイラ 8
3.1 Yet another GNU ARM Tool Chain (YAGARTO) 8
3.2 YAGARTO ツールのダウンロード方法 8
3.3 YAGARTO ツールのインストール方法 9
4 ドライバ 12
4.1 LibUSB 12
4.2 LibUSB のインストール方法 12
5 デバッガ 14
5.1 Open On-Chip Debugger (OpenOCD) 14
5.2 LibUSB ドライバの適用 14
6 Java Runtime Environment (JRE) 18
6.1 JRE の確認方法 18
6.2 JRE のインストール方法 18
7 Eclipse プラットフォーム 19
7.1 Eclipse プラットフォームのダウンロード方法 19
7.2 Eclipse IDE の開始方法 23
8 C/C++開発ツールキット (CDT) 24
8.1 最新ソフトウェアのインストール方法 24
8.2 Eclipse ネットワークコンフィグレーション 26
8.3 Eclipse CDT プラグイン 27
9 Eclipse IDE の使用方法 31
9.1 C/C++パースペクティブ 31
9.2 C/C++プロジェクトの作成 33
9.3 プロジェクトのクリーニング 37
9.4 プロジェクトのビルド 40
9.5 メイクターゲットの作成 41
10 サンプル Eclipse プロジェクト 44
10.1 ファイルの追加 45
10.2 Includes フォルダへライブラリを追加 48
10.3 makefile 51
11 Flash メモリへの書込み 61
11.1 OpenOCD と Flash 書込み 61
12 Eclipse 外部ツールのセットアップ 64
12.1 外部ツール 64
12.2 外部ツールとして OpenOCD を使用 64
13 Eclipse CDT のデバッグパースペクティブ 69
13.1 OpenOCD を使用したプログラムの書込みとデバッグ 70
13.2 RAM でのデバッグ 77
14 Eclipse Embedded Systems Register View プラグイン 84
14.1 プラグインのインストール 84
14.2 Register View の使用 87
15 Eclipse の特徴 91
15.1 概要 91
15.2 Disassembly ビュー 91
15.3 CPU Register ビュー 93
15.4 Memory ビュー 94
15.5 ブレークポイントの使用 95
16 付録 97
16.1 用語集 97
16.2 リンク 98
17 追加情報 98
18 改訂履歴 99
セールス, ソリューションおよび法律情報 100

1 イントロダクション

1.1 概要

本書は、FM3 ファミリの開発環境を、Eclipse をベースとした GNU ツールチェーンで構築するための方法を説明します。ホストとターゲットのハードウェアには下記を使用します。

本書では、ICE に J-Link もしくは ARM-USB-TINY を使用するための方法を記載します。

ホストマシン OS	Windows7 (32 ビット)
ICE	J-Link / ARM-USB-TINY
ターゲットボード	SK-FM3-176PMC-ETHERNET V1.1
ターゲットマイコン	MB9BFD18T

2 対象製品

本アプリケーションノートに記載されている内容の対象製品は、下記の通りです。

(TYPE0)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A100A	MB9AF102NA, MB9AF104NA, MB9AF105NA MB9AF102RA, MB9AF104RA, MB9AF105RA
MB9B100A	MB9BF102NA, MB9BF104NA, MB9BF105NA, MB9BF106NA MB9BF102RA, MB9BF104RA, MB9BF105RA, MB9BF106RA
MB9B300B	MB9BF304NB, MB9BF305NB, MB9BF306NB MB9BF304RB, MB9BF305RB, MB9BF306RB
MB9B400A	MB9BF404NA, MB9BF405NA, MB9BF406NA MB9BF404RA, MB9BF405RA, MB9BF406RA
MB9B500B	MB9BF504NB, MB9BF505NB, MB9BF506NB MB9BF504RB, MB9BF505RB, MB9BF506RB

(TYPE1)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A110A	MB9AF111LA, MB9AF112LA, MB9AF114LA MB9AF111MA, MB9AF112MA, MB9AF114MA, MB9AF115MA, MB9AF116MA MB9AF111NA, MB9AF112NA, MB9AF114NA, MB9AF115NA, MB9AF116NA
MB9A310A	MB9AF311LA, MB9AF312LA, MB9AF314LA MB9AF311MA, MB9AF312MA, MB9AF314MA, MB9AF315MA, MB9AF316MA MB9AF311NA, MB9AF312NA, MB9AF314NA, MB9AF315NA, MB9AF316NA

(TYPE2)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9B110T	MB9BF116S, MB9BF117S, MB9BF118S MB9BF116T, MB9BF117T, MB9BF118T
MB9B210T	MB9BF216S, MB9BF217S, MB9BF218S MB9BF216T, MB9BF217T, MB9BF218T
MB9B310T	MB9BF316S, MB9BF317S, MB9BF318S MB9BF316T, MB9BF317T, MB9BF318T
MB9B410T	MB9BF416S, MB9BF417S, MB9BF418S MB9BF416T, MB9BF417T, MB9BF418T
MB9B510T	MB9BF516S, MB9BF517S, MB9BF518S MB9BF516T, MB9BF517T, MB9BF518T
MB9B610T	MB9BF616S, MB9BF617S, MB9BF618S MB9BF616T, MB9BF617T, MB9BF618T
MB9BD10T	MB9BFD16S, MB9BFD17S, MB9BFD18S MB9BFD16T, MB9BFD17T, MB9BFD18T

(TYPE3)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A130LA	MB9AF131KA, MB9AF132KA MB9AF131LA, MB9AF132LA

(TYPE4)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9B110R	MB9BF112N, MB9BF114N, MB9BF115N, MB9BF116N MB9BF112R, MB9BF114R, MB9BF115R, MB9BF116R
MB9B310R	MB9BF312N, MB9BF314N, MB9BF315N, MB9BF316N MB9BF312R, MB9BF314R, MB9BF315R, MB9BF316R
MB9B410R	MB9BF412N, MB9BF414N, MB9BF415N, MB9BF416N MB9BF412R, MB9BF414R, MB9BF415R, MB9BF416R
MB9B510R	MB9BF512N, MB9BF514N, MB9BF515N, MB9BF516N MB9BF512R, MB9BF514R, MB9BF515R, MB9BF516R

(TYPE5)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A110K	MB9AF111K, MB9AF112K
MB9A310K	MB9AF311K, MB9AF312K

(TYPE6)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A140NA	MB9AF141LA, MB9AF142LA, MB9AF144LA MB9AF141MA, MB9AF142MA, MB9AF144MA MB9AF141NA, MB9AF142NA, MB9AF144NA
MB9A340NA	MB9AF341LA, MB9AF342LA, MB9AF344LA MB9AF341MA, MB9AF342MA, MB9AF344MA MB9AF341NA, MB9AF342NA, MB9AF344NA
MB9AA40NA	MB9AFA41LA, MB9AFA42LA, MB9AFA44LA MB9AFA41MA, MB9AFA42MA, MB9AFA44MA MB9AFA41NA, MB9AFA42NA, MB9AFA44NA
MB9AB40NA	MB9AFB41LA, MB9AFB42LA, MB9AFB44LA MB9AFB41MA, MB9AFB42MA, MB9AFB44MA MB9AFB41NA, MB9AFB42NA, MB9AFB44NA

(TYPE7)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A130N	MB9AF131M, MB9AF132M MB9AF131N, MB9AF132N
MB9AA30N	MB9AFA31L, MB9AFA32L MB9AFA31M, MB9AFA32M MB9AFA31N, MB9AFA32N

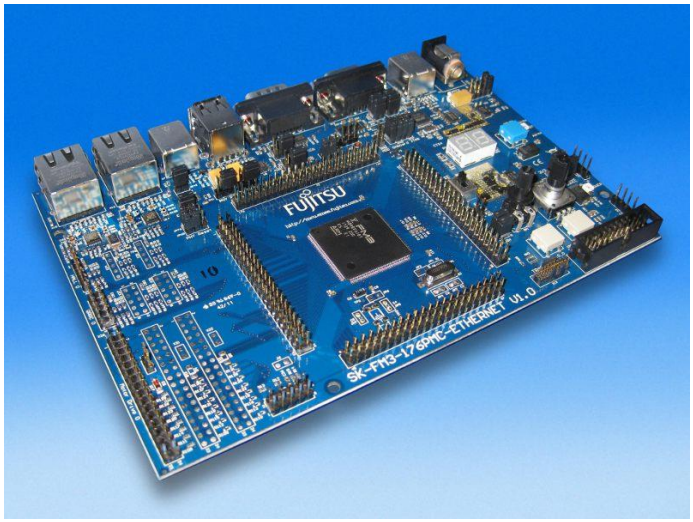
(TYPE8)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9A150R	MB9AF154M, MB9AF155M, MB9AF156M MB9AF154N, MB9AF155N, MB9AF156N MB9AF154R, MB9AF155R, MB9AF156R

(TYPE9)

シリーズ名	品種型格 (パッケージサフィックスは除く)
MB9B120M	MB9BF121K, MB9BF122K, MB9BF124K MB9BF121L, MB9BF122L, MB9BF124L MB9BF121M, MB9BF122M, MB9BF124M
MB9B320M	MB9BF321K, MB9BF322K, MB9BF324K MB9BF321L, MB9BF322L, MB9BF324L MB9BF321M, MB9BF322M, MB9BF324M
MB9B520M	MB9BF521K, MB9BF522K, MB9BF524K MB9BF521L, MB9BF522L, MB9BF524L MB9BF521M, MB9BF522M, MB9BF524M

図 1. Cypress スターターキット SK-FM3-176PMC-ETHERNET



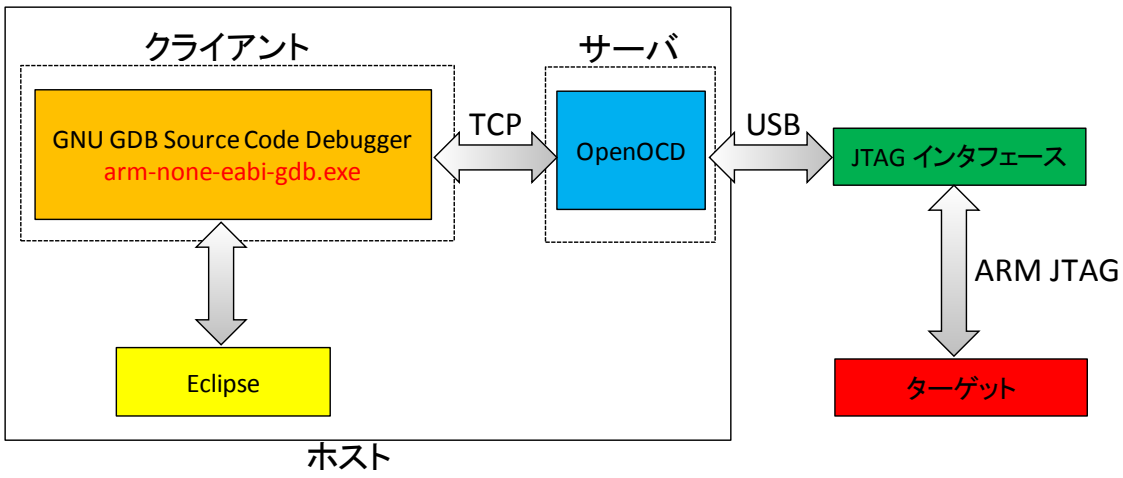
本書では、開発環境を構築するために下記のプログラムを使用します。

コンパイラ	YAGARTO
ドライバ	LibUSB
デバッガ	OpenOCD
統合開発環境	Eclipse + C/C++開発ツールセット(CDT)
その他	Java Runtime Environment(JRE)

2.1 JTAG インタフェース

MCU へのプログラムの書込みやデバッグには JTAG を使用します。JTAG を使用する際のホストとターゲットの関係を以下の図に示します。

図 2 JTAG を使用する際のホストとターゲットの関係



2.2 J-Link

J-Link には IAR 社製のものを使用します。

図 3 IAR 社製 J-Link



詳細は下記の URL をご参照ください。

http://www.iar.com/Global/Products/Hardware-Debug-probes/DS-J-Link-ARM-09.pdf

J-Link は下記の特徴があります。

- USB から電源供給
- J-Link GDB サーバが使用可能
- RAM / Flash へ書込み可能
- フラッシュブレークポイントが使用可能
- SWD / SWV が使用可能
- 電圧範囲: 1.2V～5V

2.3 ARM-USB-TINY

olimex 社製の ARM-USB-TINY を使用します。

図 4 olimex 社製 ARM-USB-TINY



詳細は下記の URL をご参照ください。

https://www.olimex.com/Products/ARM/JTAG/ARM-USB-TINY/

ARM-USB-TINY は下記の特徴があります。

- ARM マイコンのデバッグ可能 (OpenOCD がサポートしているものに限る)
- 高速 USB2.0 JTAG ドングルインタフェース
- ARM 標準の 2×10 ピン JTAG コネクタを使用可能
- 電圧範囲: 2V～5V
- OpenOCD をサポート

3 コンパイラ

3.1 Yet another GNU ARM Tool Chain (YAGARTO)

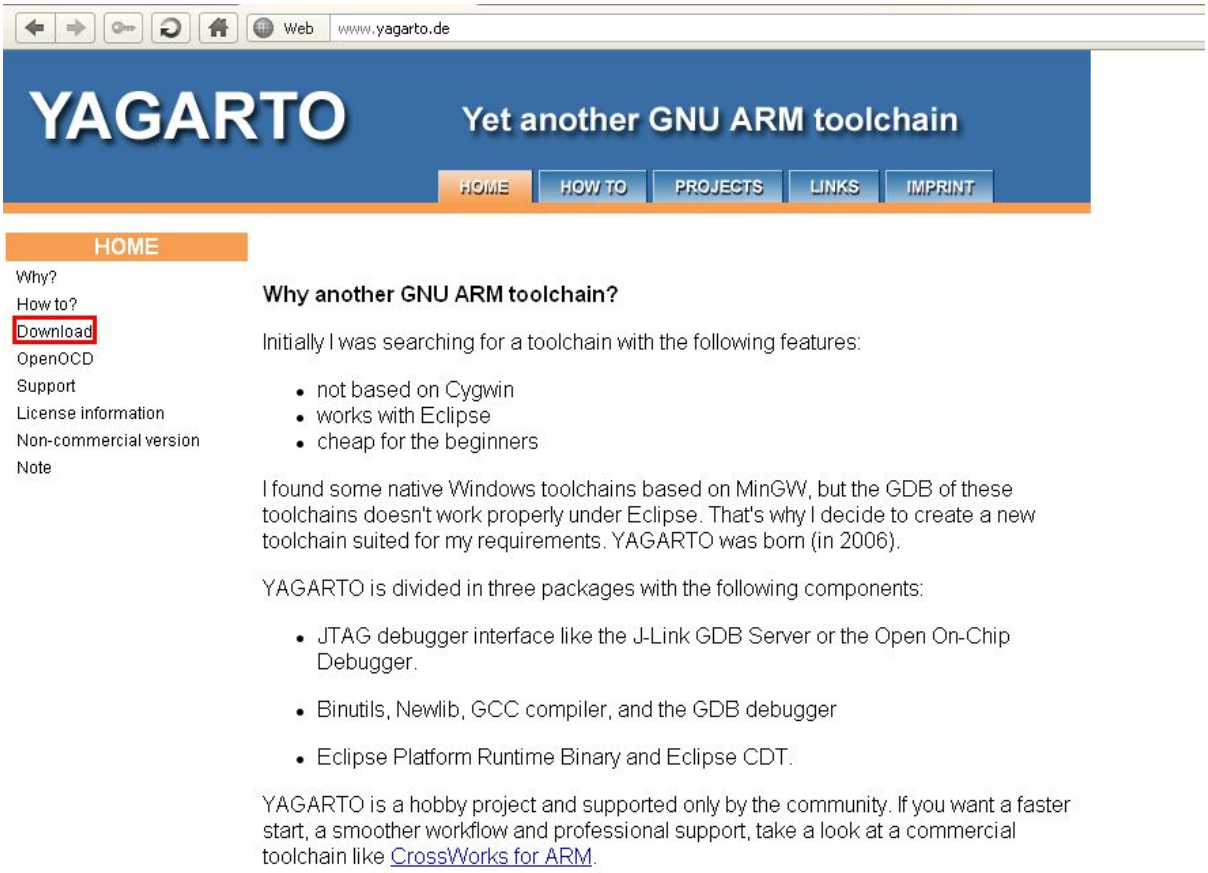
現在、数多くの GNU ARM コンパイラツールセットが、ビルドされた状態でウェブに公開されています。本書では Michael Fischer 社が開発した YAGARTO という ARM コンパイラツールを使用します。使用するバージョンは Windows (Intel) 向けにコンパイルされたものです。

ARM コンパイラ以外にも、YAGARTO プロジェクトは Eclipse CDT のプロジェクト作成に必要なツール (例えば make ユーティリティ) を提供しています。

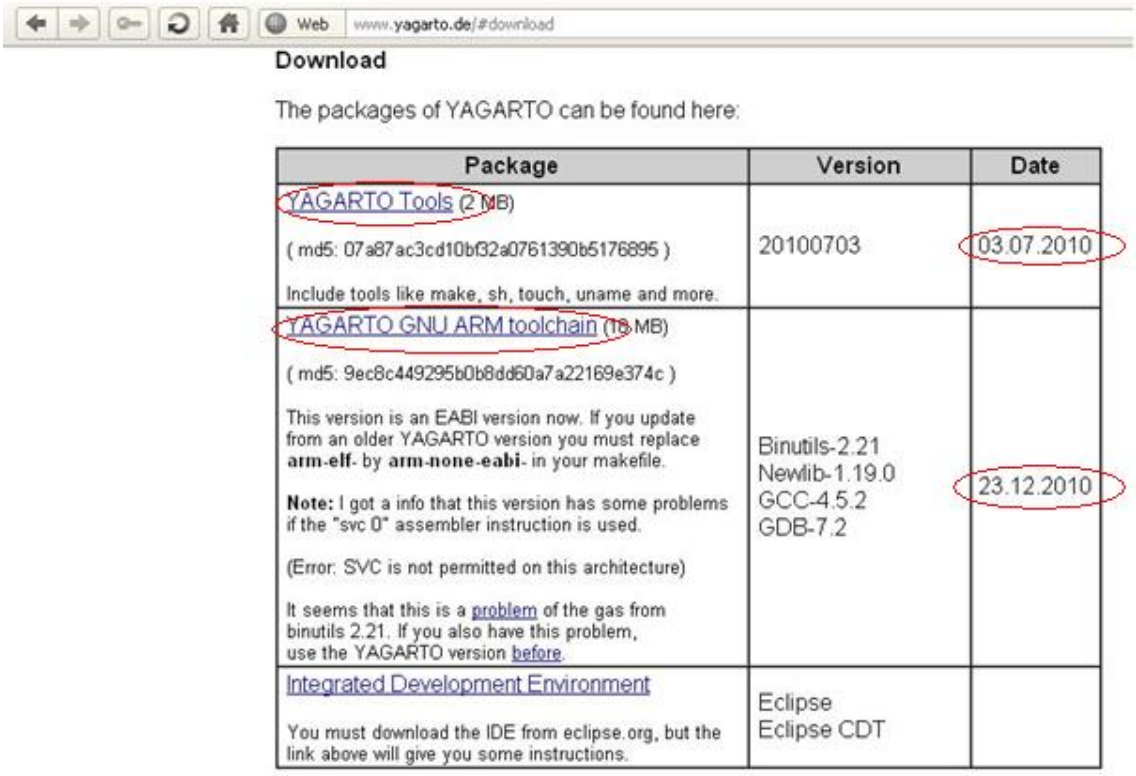
3.2 YAGARTO ツールのダウンロード方法

YAGARTO コンポーネントは下記のウェブページからダウンロードできます。

http://www.yagarto.de/



左側のメニューにある ”Download” をクリックします。



リンク先には推奨されている最新バージョンがアップされています。

ここでは最初の 2 つのパッケージのみをダウンロードします。3 つ目の ”Eclipse” と ”Eclipse CDT” は 8 章で別途説明します。

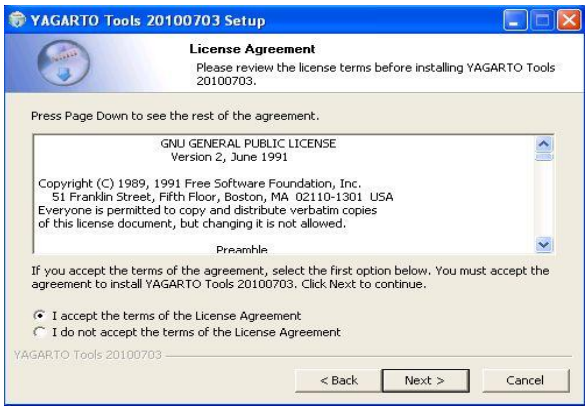
3.3 YAGARTO ツールのインストール方法

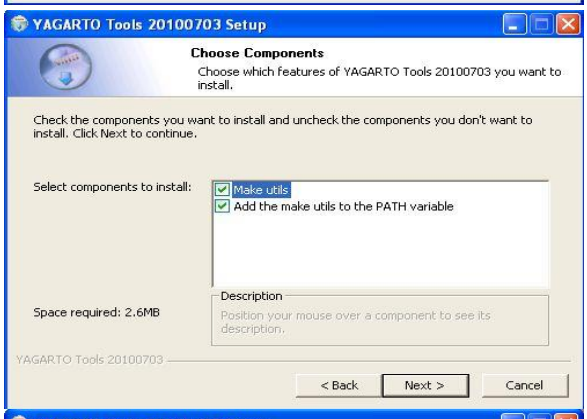
任意のフォルダ (ここでは ”Yagarto-Downloads”) にパッケージを保存後、これらのインストールを開始します。



“yagarto-tools-20100703-setup” (もしくはそれ以降の最新版) から make ユーティリティをインストールします。

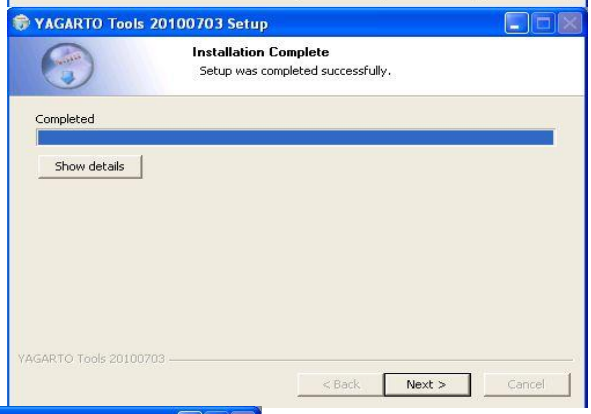

YAGARTO Tools 20100703 Setup
Welcome to the YAGARTO Tools 20100703 Setup Wizard
This wizard will guide you through the installation of YAGARTO Tools 20100703.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel


YAGARTO Tools 20100703 Setup
License Agreement
Please review the license terms before installing YAGARTO Tools 20100703.
Press Page Down to see the rest of the agreement.
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
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 Tools 20100703. Click Next to continue.
I accept the terms of the License Agreement
I do not accept the terms of the License Agreement
YAGARTO Tools 20100703
< Back
Next >
Cancel


YAGARTO Tools 20100703 Setup
Choose Components
Choose which features of YAGARTO Tools 20100703 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: 2.6MB
YAGARTO Tools 20100703
< Back
Next >
Cancel


YAGARTO Tools 20100703 Setup
Choose Install Location
Choose the folder in which to install YAGARTO Tools 20100703.
Setup will install YAGARTO Tools 20100703 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Yagarto\yagarto-tools-20100703
Browse...
Space required: 2.6MB
Space available: 3.4GB
YAGARTO Tools 20100703
< Back
Next >
Cancel

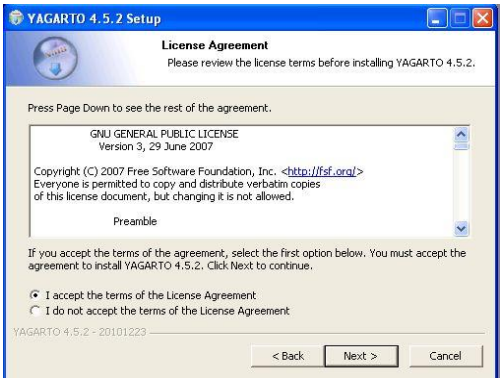

YAGARTO Tools 20100703 Setup
Choose Start Menu Folder
Choose a Start Menu folder for the YAGARTO Tools 20100703 shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
YAGARTO Tools
Autostart
Avira
Cygwin
ElsterFormular
Microsoft Office
OpenOffice.org 3.3
Packard Bell ImageWriter
Panda Security
Sandboxie
Spiele
Verwaltung
Do not create shortcuts
YAGARTO Tools 20100703
< Back
Install
Cancel

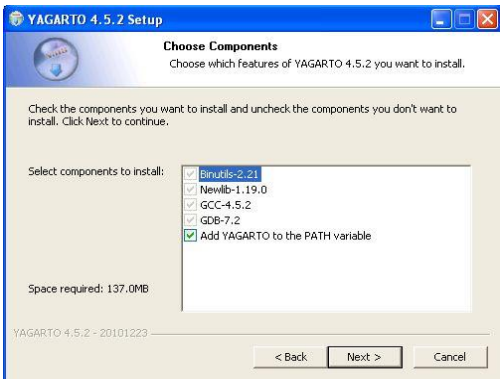

YAGARTO Tools 20100703 Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
YAGARTO Tools 20100703
< Back
Next >
Cancel

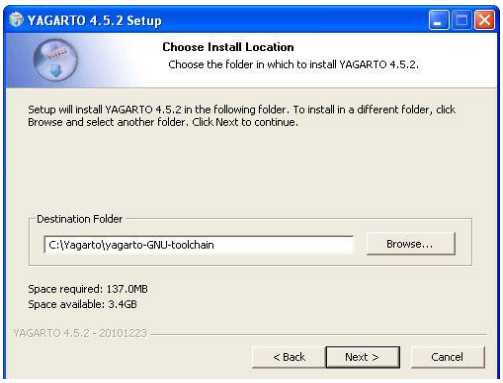

YAGARTO Tools 20100703 Setup
Completing the YAGARTO Tools 20100703 Setup Wizard
YAGARTO Tools 20100703 has been installed on your computer.
Click Finish to close this wizard.
Visit the YAGARTO site for the latest news.
< Back
Finish
Cancel

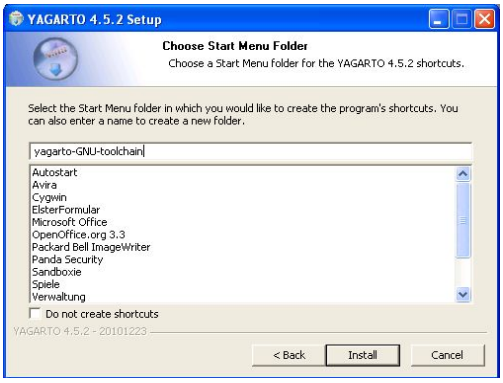
続いて、“yagarto-bu-2.21_gcc-4.5.2-c-c++_nl-1.19.0_gdb-7.2_eabi_20101223” もしくはそれ以降の最新版から ARM コンパイラをインストールします。

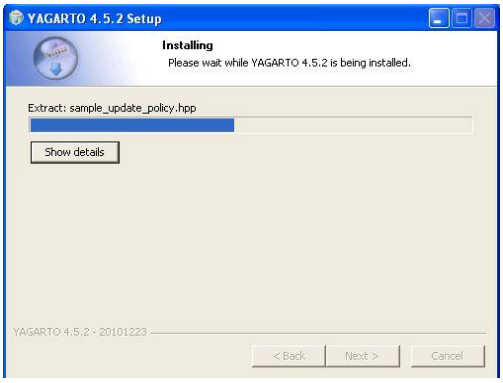


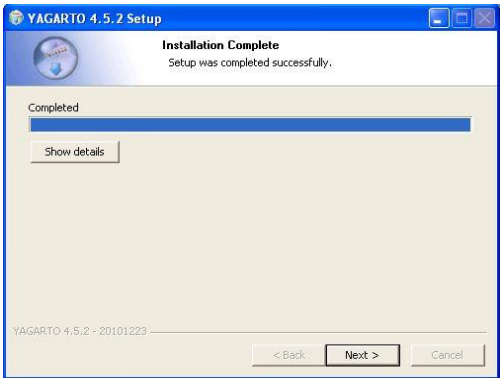














4 ドライバ

4.1 LibUSB

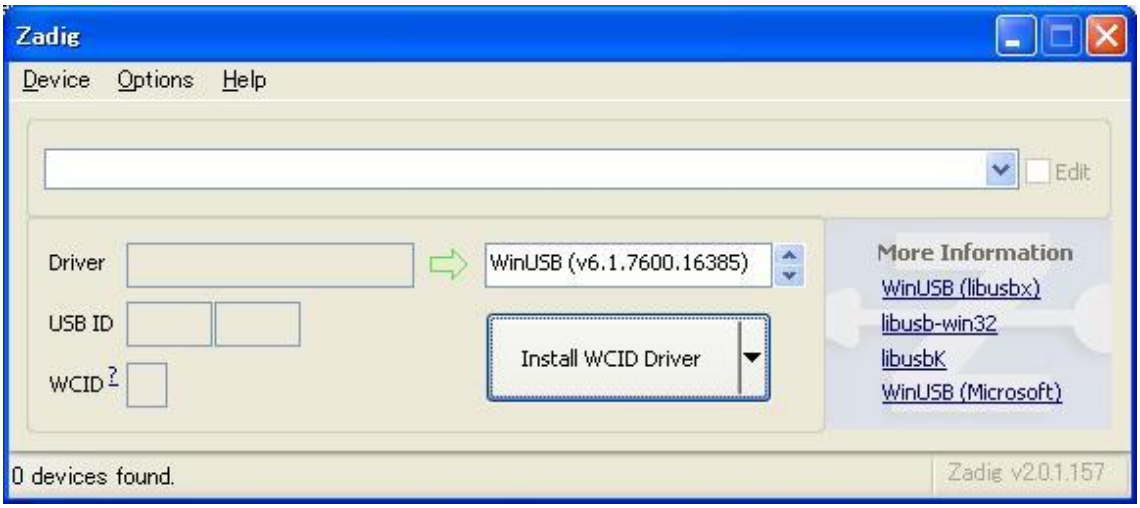
[備考] 本章では J-Link を例にドライバの設定方法を記載していますが、ARM-USB-TINY にも共通の内容です。

J-Link で OpenOCD を使用するためにドライバを設定します。本書では LibUSB というドライバを使用します。J-Link の正規のドライバは OpenOCD に対応していないため、LibUSB に入れ替える必要があるためです。入替えには Zadig というフリーツール (LGPL ライセンス) を使用します。Zadig にはあらかじめ LibUSB が含まれているため、LibUSB を個別でダウンロードする必要はありません。Zadig は下記のウェブページから入手できます。

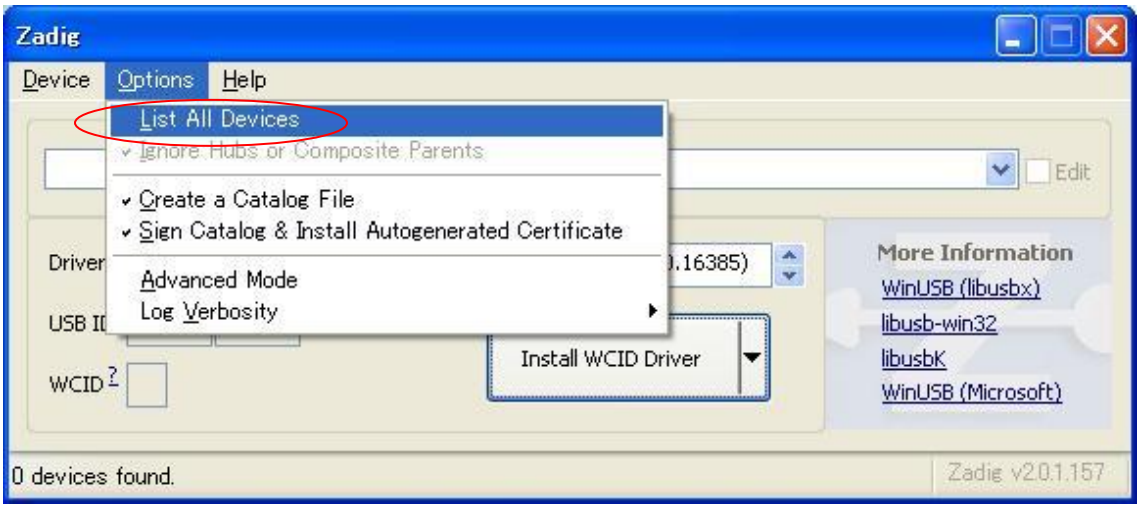
http://sourceforge.net/projects/libwdi/files/zadig/

4.2 LibUSB のインストール方法

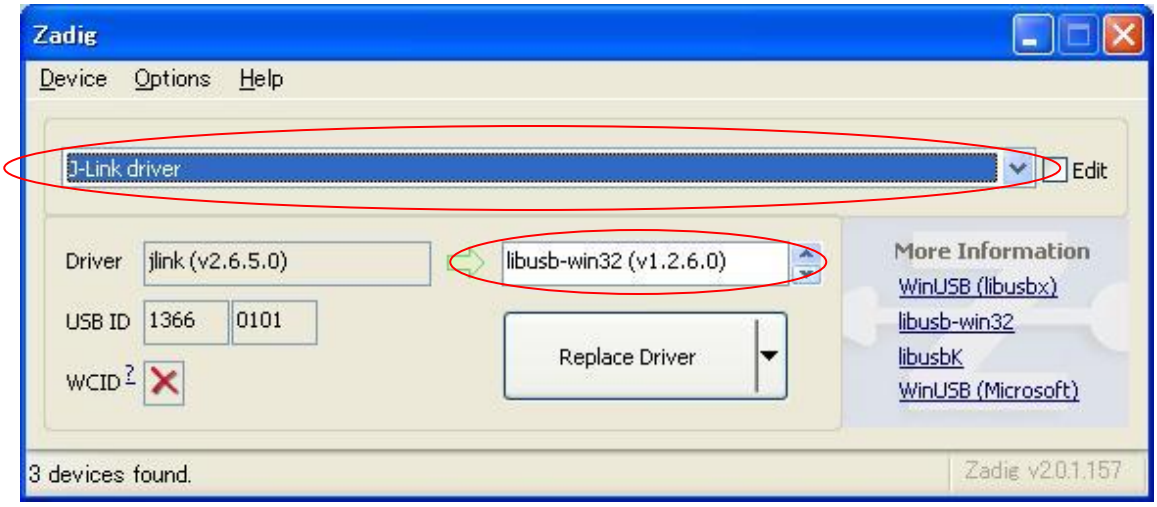
J-Link を PC と接続します。J-Link に正規のドライバが設定されていても、特に削除する必要はありません。Zadig を起動後、以下のようなウィンドウが表示されます。



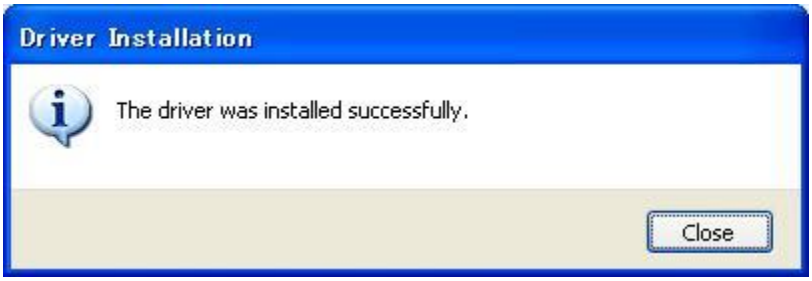
“Options” -> ”List All Devices” をクリックします。



プルダウンメニューから ”J-Link Driver” を選択し、Driver を ”libusb-win32 (v1.2.6.0)” に設定します。



“Replace Driver” をクリックすると、ドライバの入替えが開始されます。以下のメッセージが出れば完了です。



デバイスマネージャから、”libusb-win32 devices” に J-Link driver があることを確認します。



5 デバッガ

5.1 Open On-Chip Debugger (OpenOCD)

OpenOCD は、ARM コアに JTAG インタフェースを通してアクセスするためのオープンツールです。OpenOCD は本書で使用する J-Link, ARM-USB-TINY を含めて多くの JTAG ドングルをサポートしています。多くのドングルは、Future Technology Devices International 社の FT2232D チップを搭載した FTDI デバイスで構成されています。

本章では、OpenOCD の使用方法を説明します。

5.2 LibUSB ドライバの適用

5.2.1 LibUSB ドライバサポート版の OpenOCD インストール方法

LibUSB をサポートした Windows 向けの OpenOCD プログラムが下記のウェブページからダウンロードできます (FM3 では OpenOCD 0.5.0 以降を使用してください)。

http://openocd.sourceforge.net/

保存し、任意のフォルダ (例えば C:¥OpneOCD_LibUSB) に解凍します。

5.2.2 OpenOCD の動作確認

OpenOCD にはコンフィグレーションスクリプトファイルである ”openocd.cfg” が必要です。MB9BFD18T 用のサンプルを下記に示します (本ファイルは本書で使用するサンプルプロジェクトに含まれています)。

```
#interface jlink          } j-Link を使用する場合はこの行を有効にしてください

#interface ft2232
#ft2232_device_desc "Olimex OpenOCD JTAG TINY"   } ARM-USB-TINY を使用する場合はこれらの行を有効にしてください
#ft2232_layout olimex-jtag
#ft2232_vid_pid 0x15ba 0x0004

# Cortex-M3 with 1MB Flash and 64*2 kB RAM

if { [info exists CHIPNAME] } {
      set  _CHIPNAME $CHIPNAME
} else {
      set  _CHIPNAME mb9bfxx6
}

if { [info exists ENDIAN] } {
      set  _ENDIAN $ENDIAN
} else {
      set  _ENDIAN little
}

if { [info exists CPUTAPID ] } {
      set _CPUTAPID $CPUTAPID
} else {
      set _CPUTAPID 0x4ba00477
}

#delays on reset lines
jtag_nsrst_delay 100
jtag_ntrst_delay 100

# cortex-M3 reset configuration
# reset_config trst_only
reset_config trst_and_srst

jtag newtap $_CHIPNAME cpu -irlen 4 -ircapture 0x1 -irmask 0xf -expected-
id $_CPUTAPID

set _TARGETNAME $_CHIPNAME.cpu
target create $_TARGETNAME cortex_m3 -endian $_ENDIAN -chain-position
$_TARGETNAME

# MB9BFD18 has 64*2kB of RAM on its main system bus
$_TARGETNAME configure -work-area-phys 0x1FFF0008 -work-area-size 0x8000
```

```
# MB9BFD18 has 1MB of user-available FLASH
# flash bank mb9bf500 <base> <size> 0 0 <target#> <variant> <cclk>
[calc_checksum]

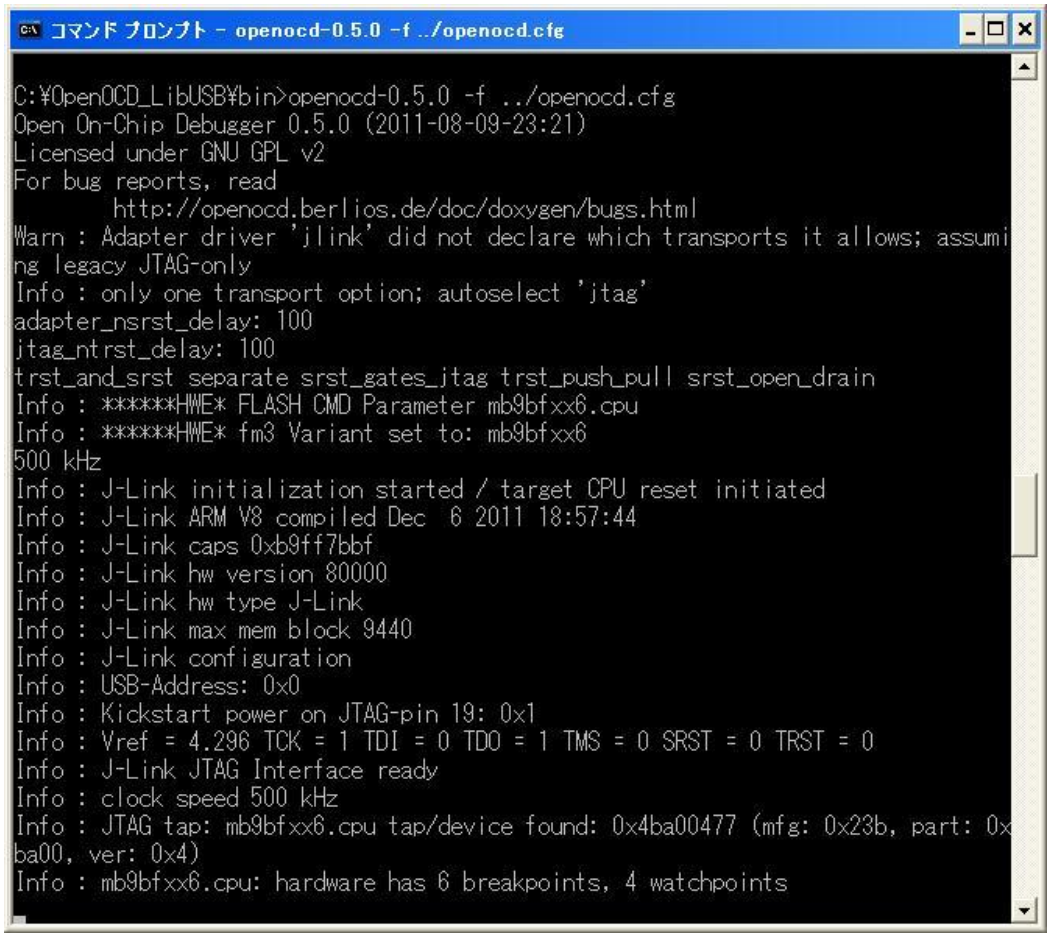
set _FLASHNAME $_CHIPNAME.flash
flash bank $_FLASHNAME fm3 0 0 0 0 $_TARGETNAME mb9bfxx6

# 4MHz / 6 = 666kHz, so use 500
jtag_khz 500
```

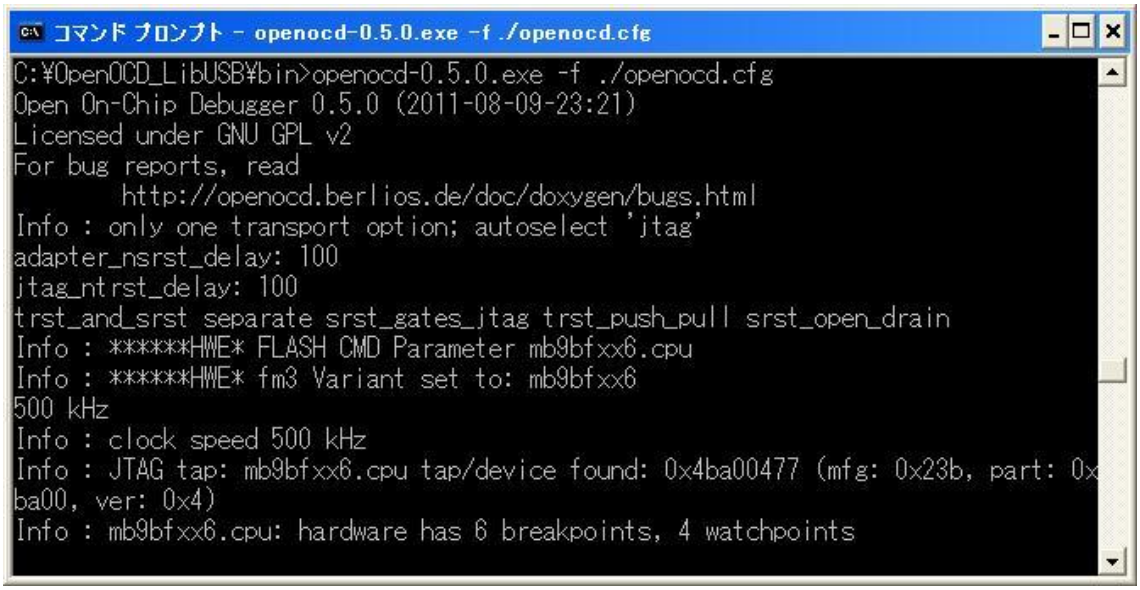
OpenOCD の動作確認をします。まず J-Link と評価ボードを接続します。次に、コマンドプロンプトから解凍先のフォルダへ移動し、-f の引数をつけて ”openocd-0.5.0” を実行します。-f にはコンフィグレーションファイルがあるフォルダのパスを設定してください。

```
>Openocd-0.5.0 -f <Your path to the Eclipse workspace project>/openocd.cfg
```

j-Link を使用した場合、以下の実行結果が出力されることを確認します。



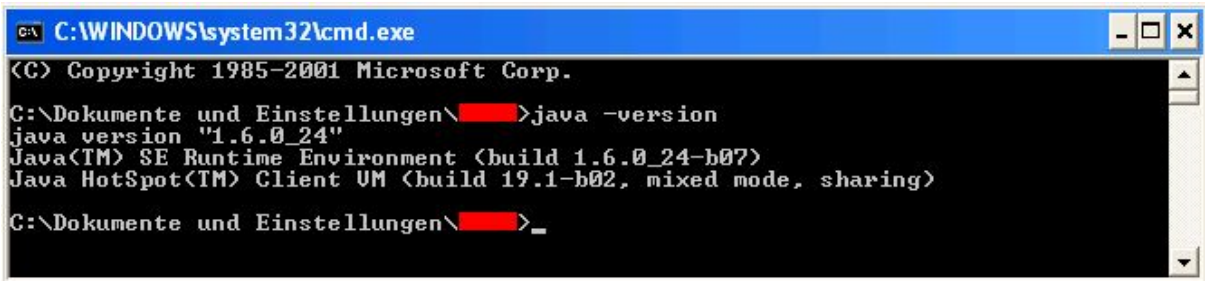
ARM-USB-TINY を使用した場合、以下の実行結果が出力されることを確認します。



6 Java Runtime Environment (JRE)

6.1 JRE の確認方法

Eclipse を動作させるには、Java 仮想マシンを入手しておく必要があります。既に Java が使用できるかをチェックするためには、DOS コンソールで `java -version` コマンドを使用します。

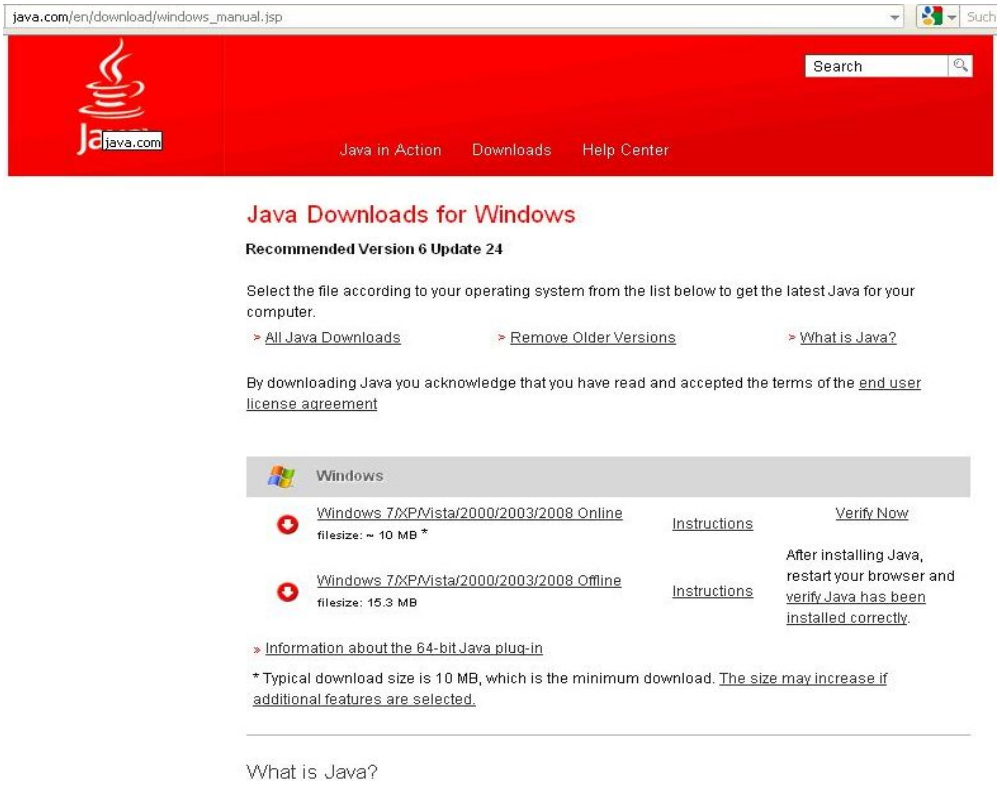


もしコマンドが認識できない場合は、JRE のインストールが必要です。

6.2 JRE のインストール方法

JRE は下記の URL からダウンロードできます。

http://java.com/



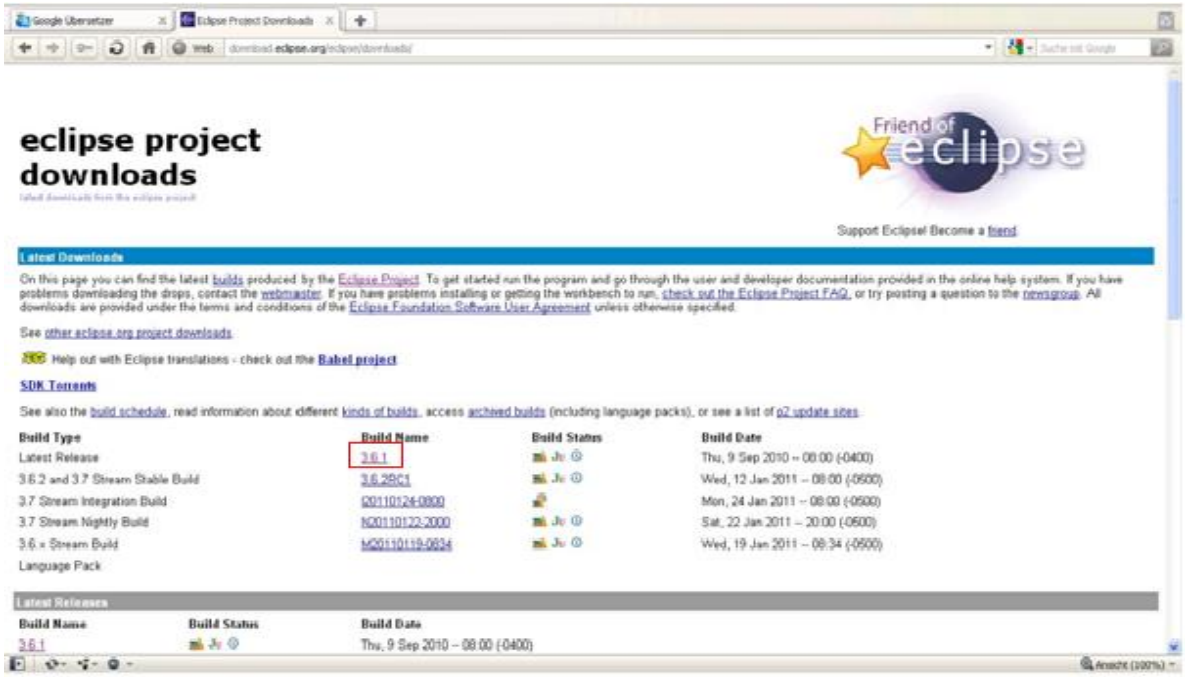
インストールファイルをダウンロードし、JRE のインストールを開始してください。

7 Eclipse プラットフォーム

7.1 Eclipse プラットフォームのダウンロード方法

Eclipse の最新バージョンは下記のウェブページからダウンロードできます。

http://download.eclipse.org/eclipse/downloads/



Helios 3.6.1 (もしくはそれ以降) は様々なパッケージが存在します。これらのパッケージは、ダウンロードウェブページ左側のメニューから入手できます (本書では Helios 3.6.1 を使用します)。

最低限の機能を持った Eclipse プラットフォームを使用するため、メニューから ”Platform Runtime Binary” を選択します。



Eclipse プラットフォームバイナリパッケージは、様々な OS 向けのものが入手できます。

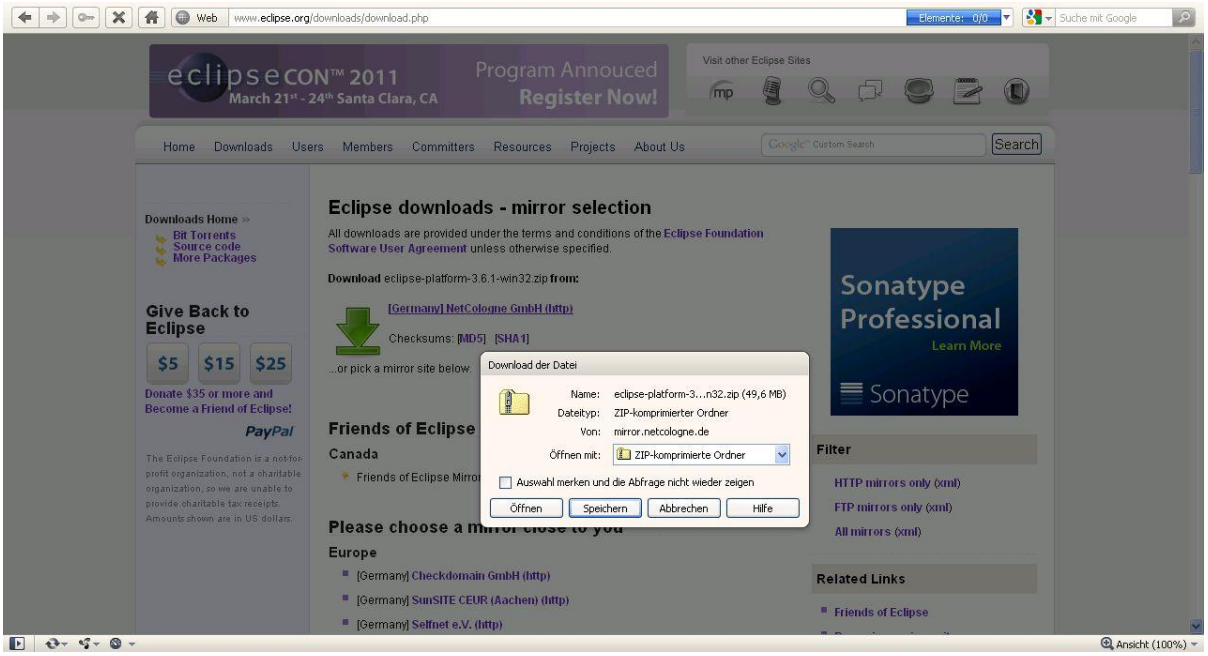
本書で使用するために十分な 32 ビット版 Windows 向けの Eclipse パッケージは、”Platform Runtime Binary” にある一番上の http をクリックすることでダウンロードできます。

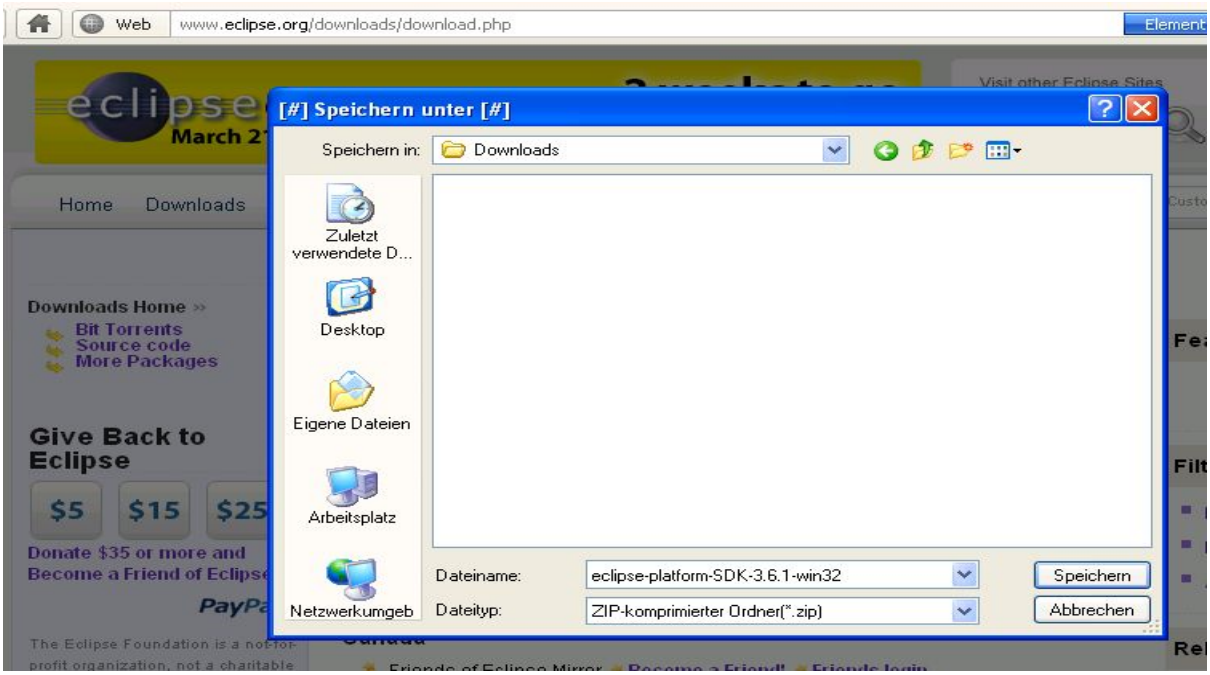


Platform Runtime Binary

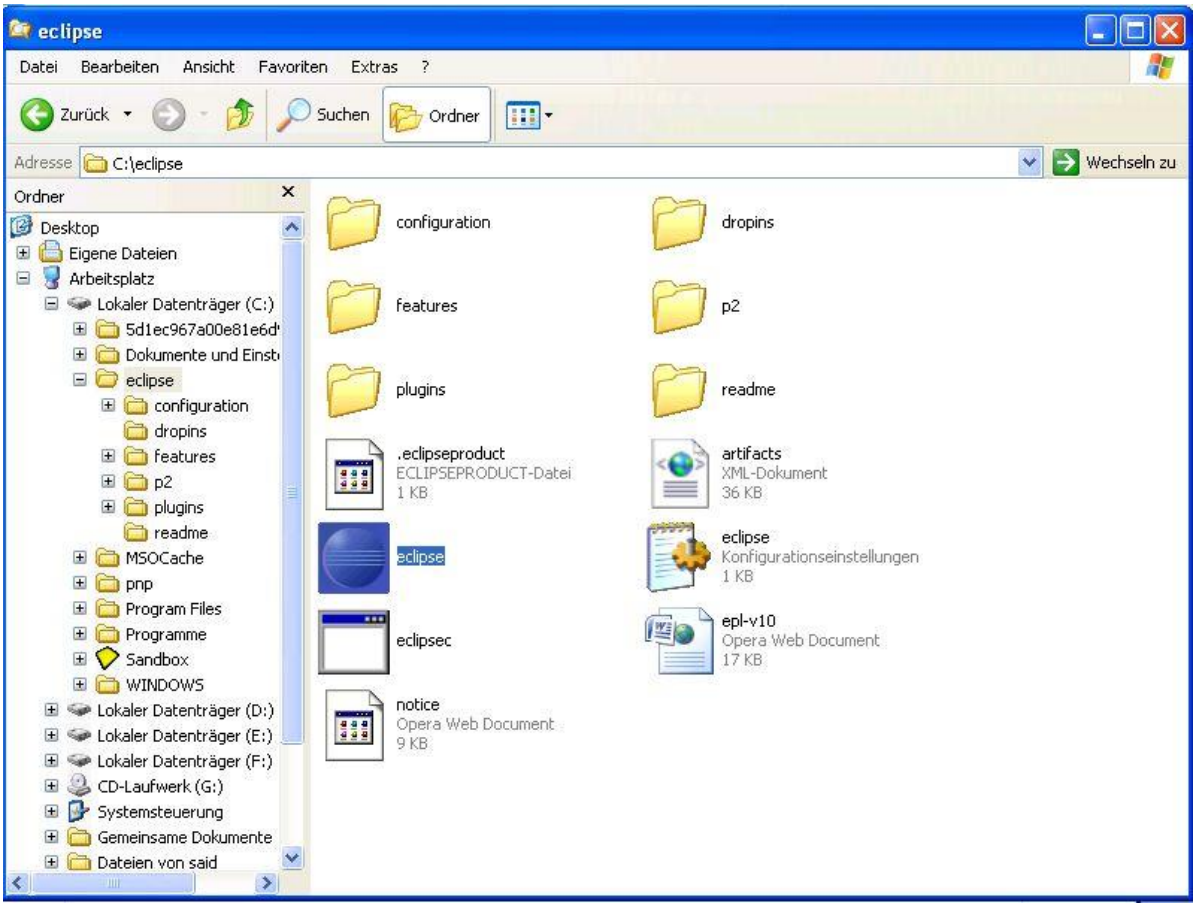
Status	Platform	Download	Size
	Windows (Supported Versions)	(http)	57 MB
	Windows (x86_64) (Supported Versions)	(http)	57 MB
	Linux (x86/GTK 2) (Supported Versions)	(http)	57 MB
	Linux (x86_64/GTK 2) (Supported Versions)	(http)	57 MB

Eclipse プラットフォームのバイナリは多くの http ミラーサイトから入手できます。それらから 1 つのミラーサイトを選択後、ダウンロードが開始されます。





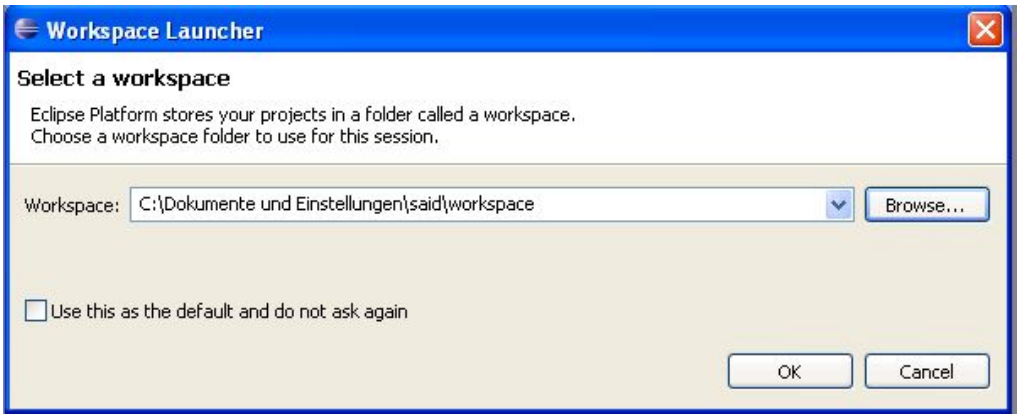
ダウンロード後、eclipse-platform-SDK-3.6.1-win32.zip を保存し、例えば C:¥に解凍します。



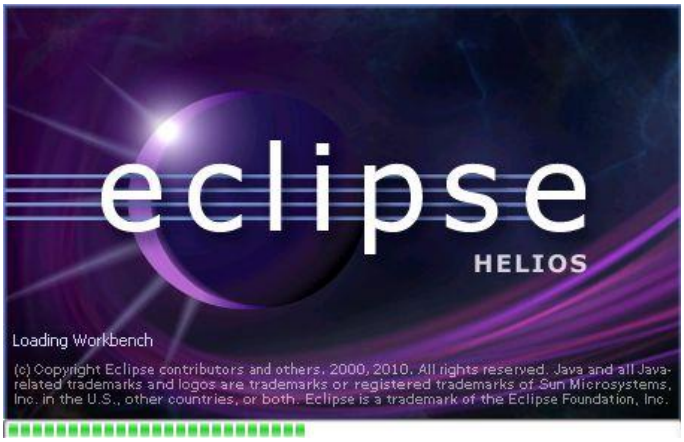
インストールファイルを実行し、Eclipse のインストールは完了です。

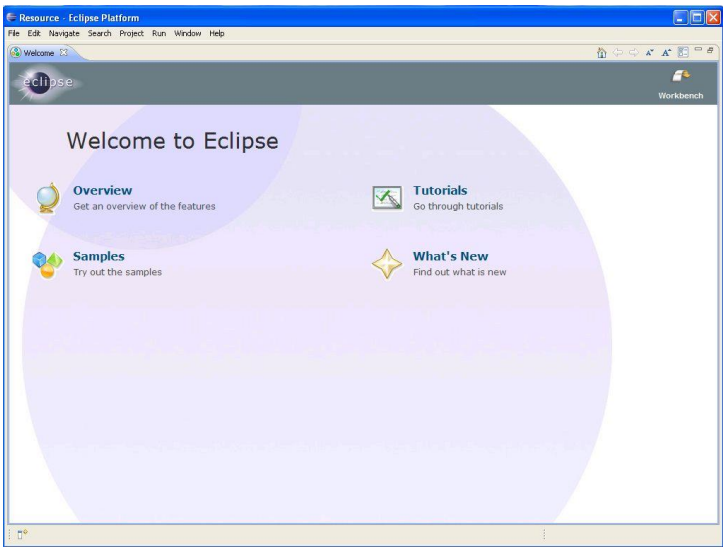
7.2 Eclipse IDE の開始方法

初めに、C:¥eclipse フォルダにある eclipse.exe を実行し、Eclipse のプロジェクトファイルを保存するワークスペースを指定します。



ワークスペースを指定すると、Eclipse がスタートします。



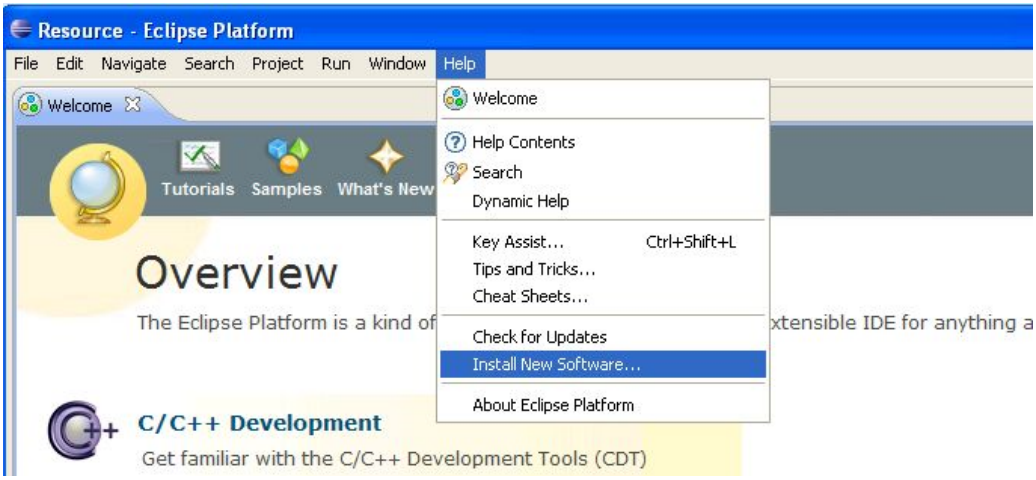


8 C/C++開発ツールキット (CDT)

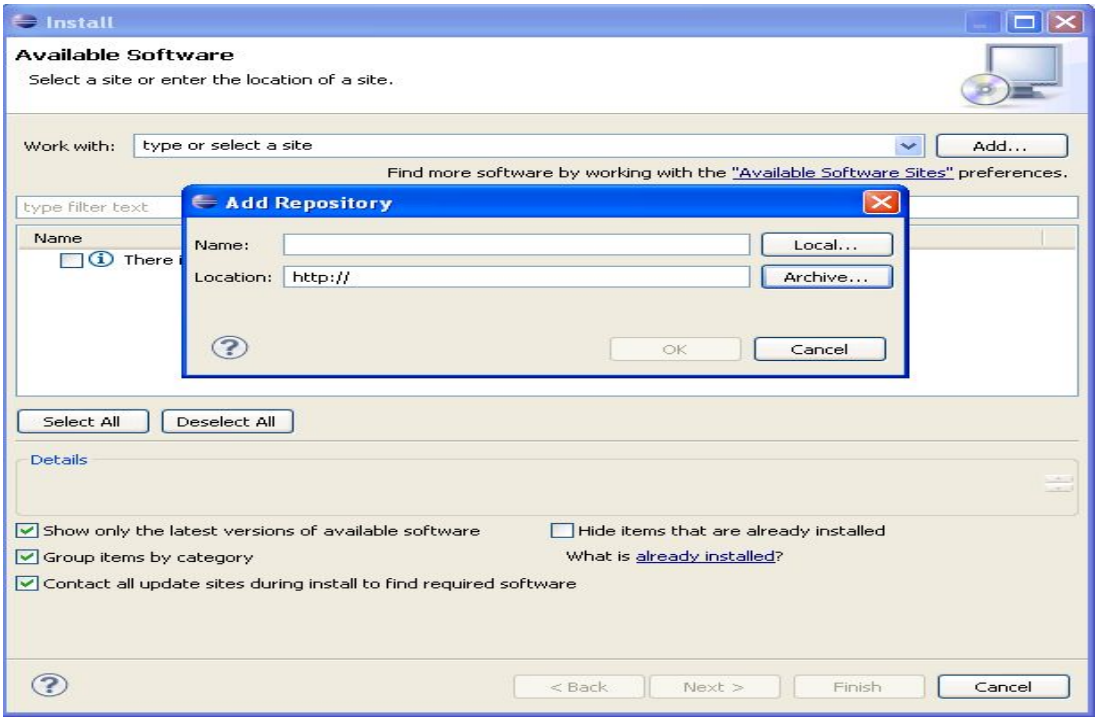
8.1 最新ソフトウェアのインストール方法

Eclipse で C/C++の開発をするためには、CDT パッケージを Eclipse へインポートする必要があります。CDT パッケージはプラグインのように入手できます。

Eclipse で最新のソフトウェアをインストールするためには、Eclipse を起動し以下のとおり ”Help” -> ”Install New Software menu…” をクリックします。

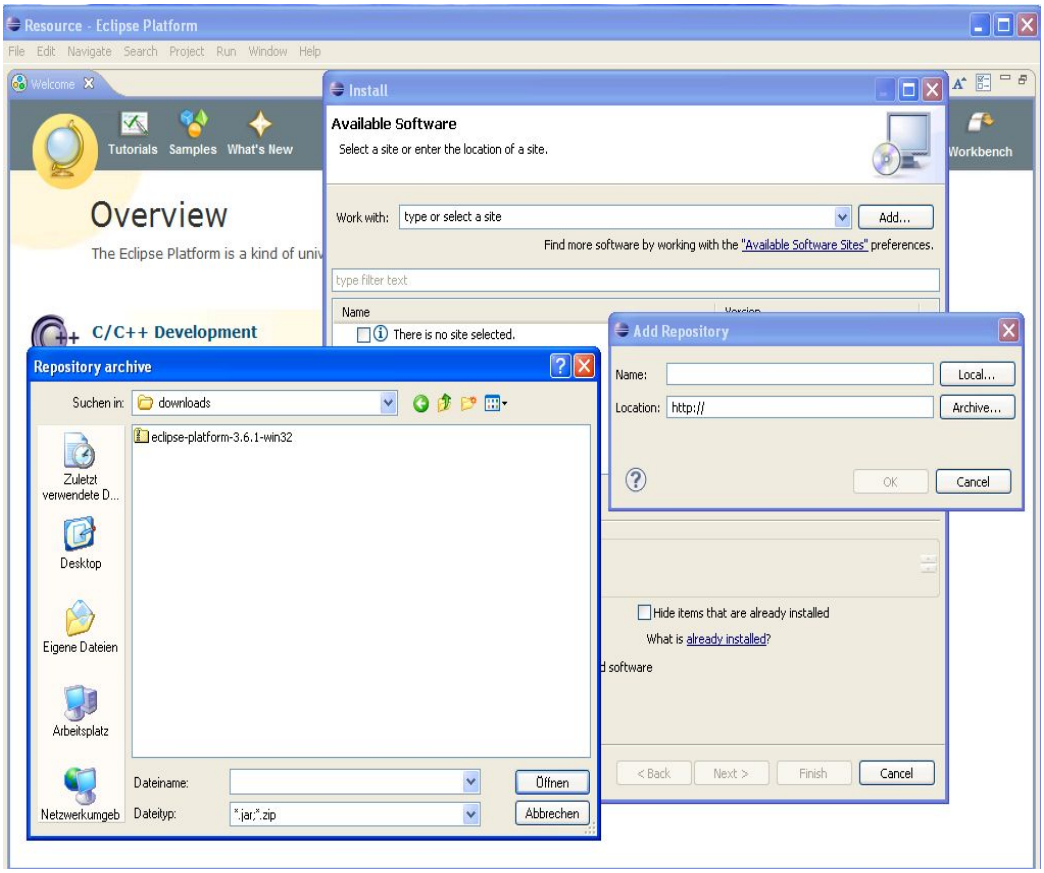


ユーザは、追加したいソフトウェアを選択できます。これは CDT プラグインだけでなくほかのパッケージを Eclipse へインストールする場合も同様です。”add” ボタンをクリック後、”Add Repository” ウィンドウが表示されます。

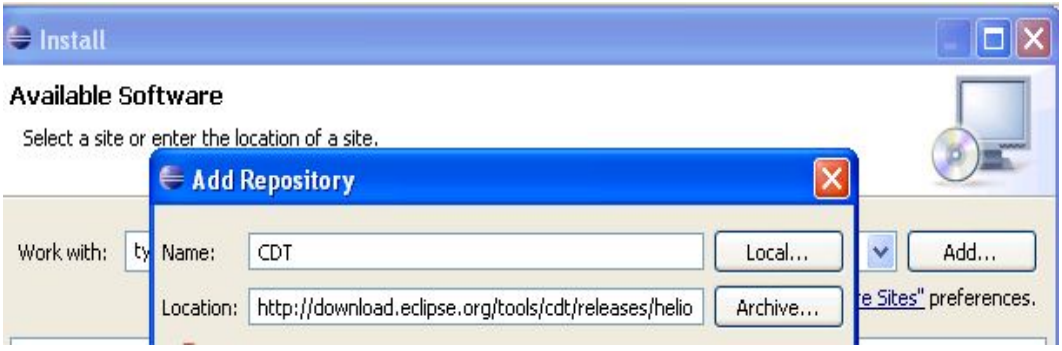


Eclipse には新しいプラグインをインストールする際に 2 つの方法があります。

プラグインを JAR ファイルや ZIP ファイルのようにローカルで持っている場合は、オフラインでインストールできます。



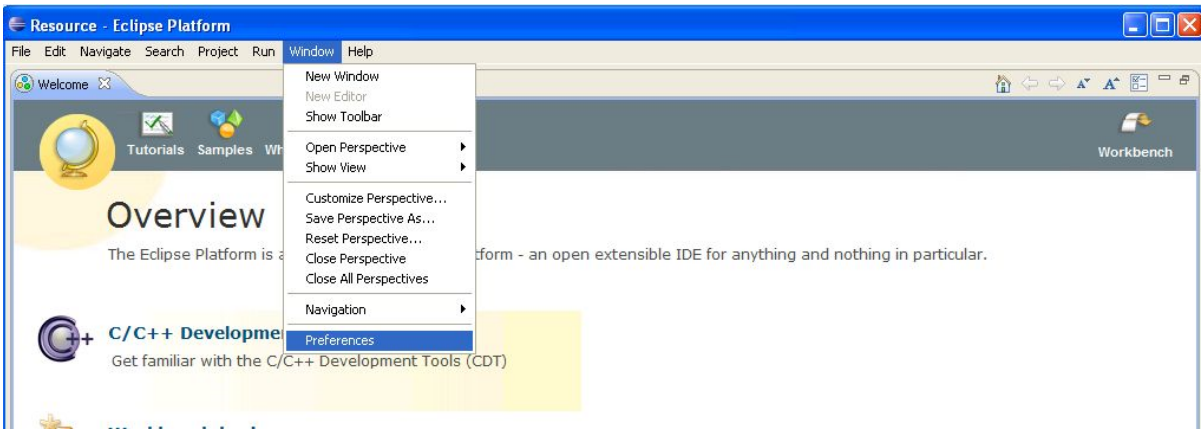
プラグインをウェブページから入手する場合は、オンラインでインストール (もしくはアップデート) します。



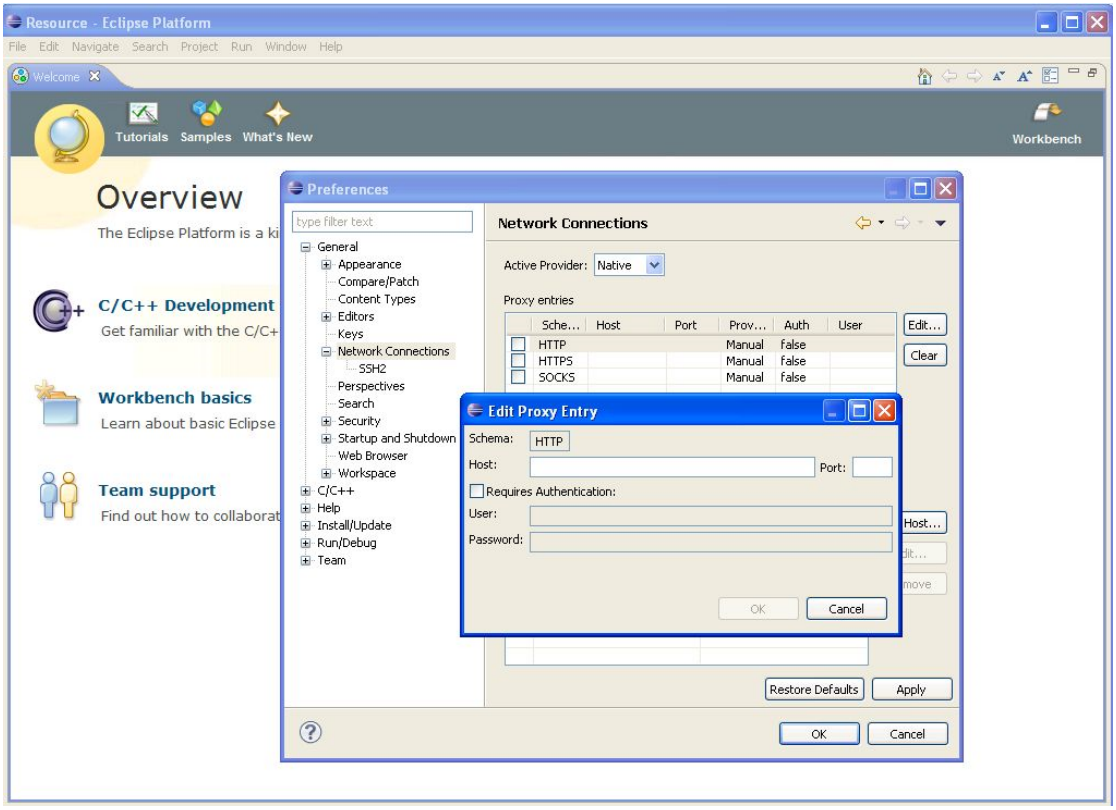
オンラインでのインストール方法が推奨されています。オンラインでインストールする場合は、インストール前にネットワーク環境を確認してください。

8.2 Eclipse ネットワークコンフィグレーション

“Window” -> ”Preferences” から、ネットワークのコンフィグレーションを設定できます。



“Network connections” を選択します。このフィールドから、Eclipse をインターネットへ接続するために必要な設定を編集できます。



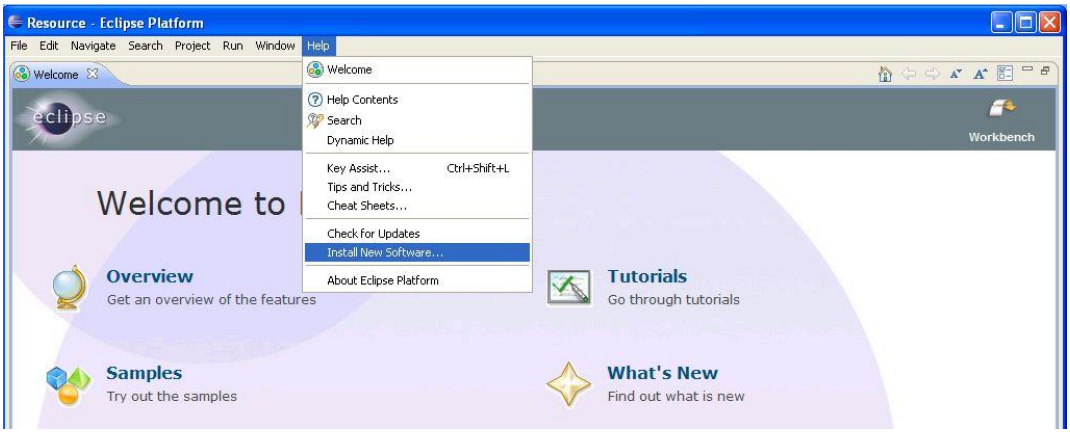
変更後、”Apply” ボタンをクリックして新しいネットワークコンフィグレーションを保存します。これで、CDT プラグインをオンラインでインストールできます。

8.3 Eclipse CDT プラグイン

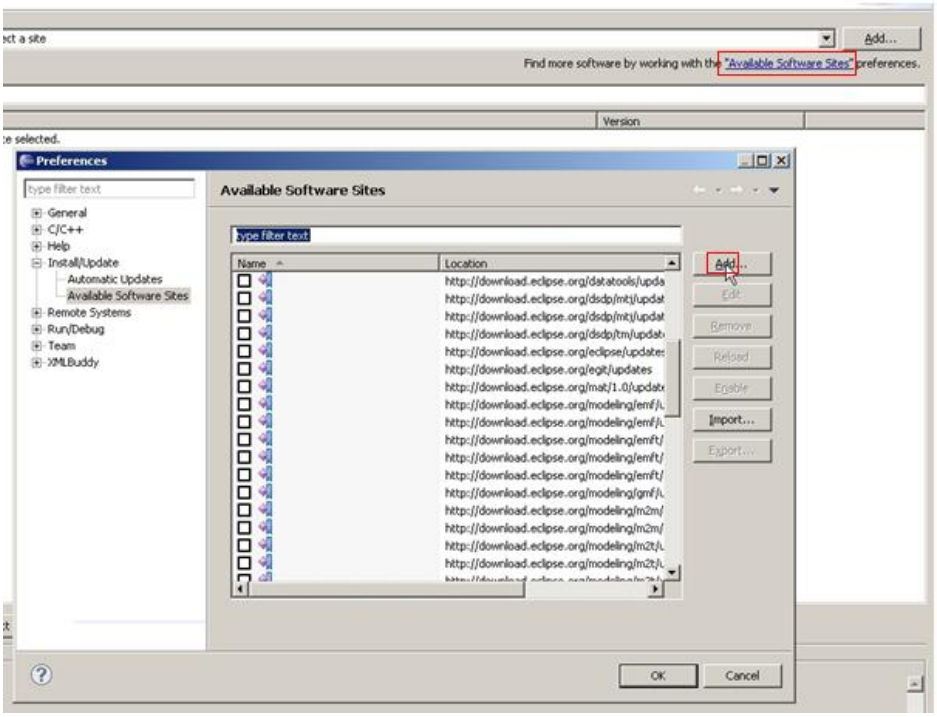
CDT プラグインには Standard と Zylin のバージョンが存在します。どちらか 1 つのバージョンをインストールしてください (本書では Standard を使用します)。

新しい CDT プラグインのインストール方法を下記に示します。

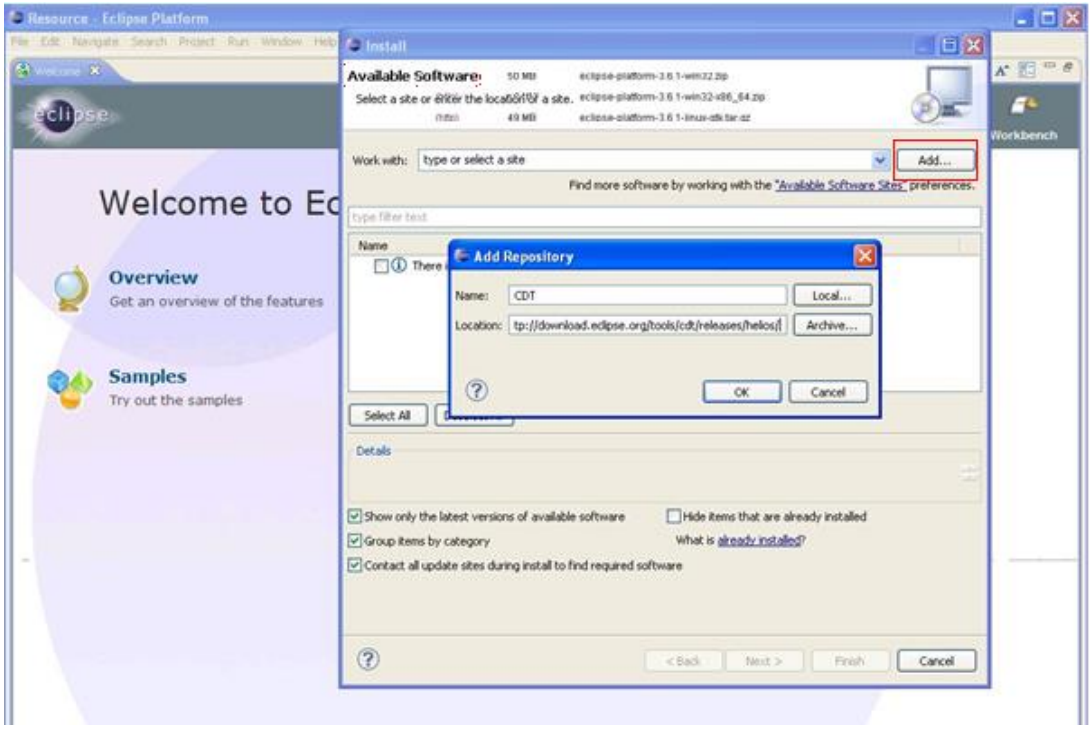
“Help” メニューから、”Install New Software…” をクリックします。



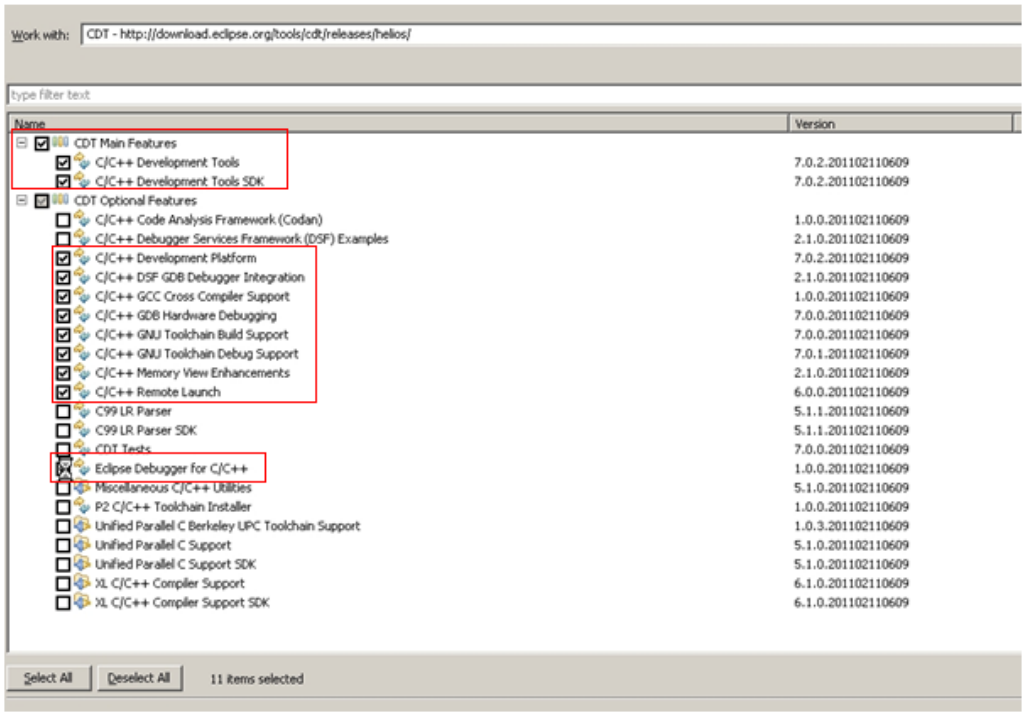
次のウィンドウ ”Available Software Sites” から、CDT のダウンロードミラーサイトを探します。ミラーサイトは http://download.eclipse.org/tools/cdt/releases/helios/ です。



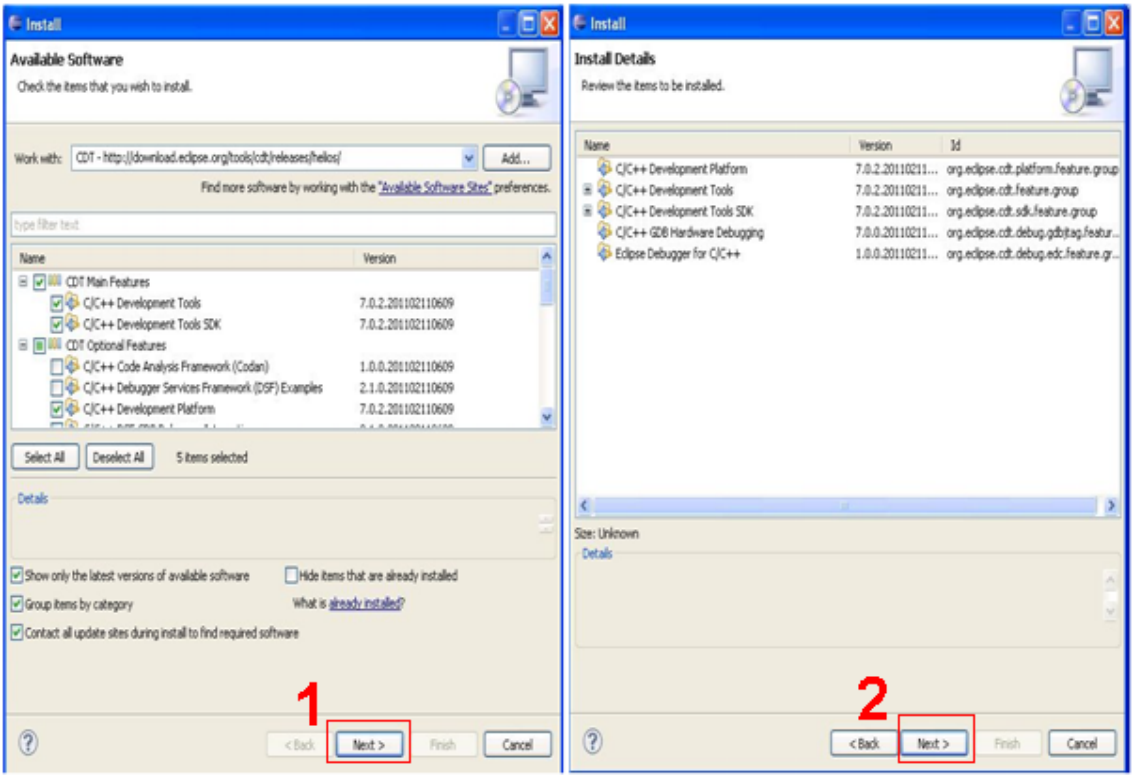
目的のミラーサイトがない場合、”Add” ボタンをクリックします。”Name” に CDT、”Location” に http://download.eclipse.org/tools/cdt/releases/helios/と入力してください。

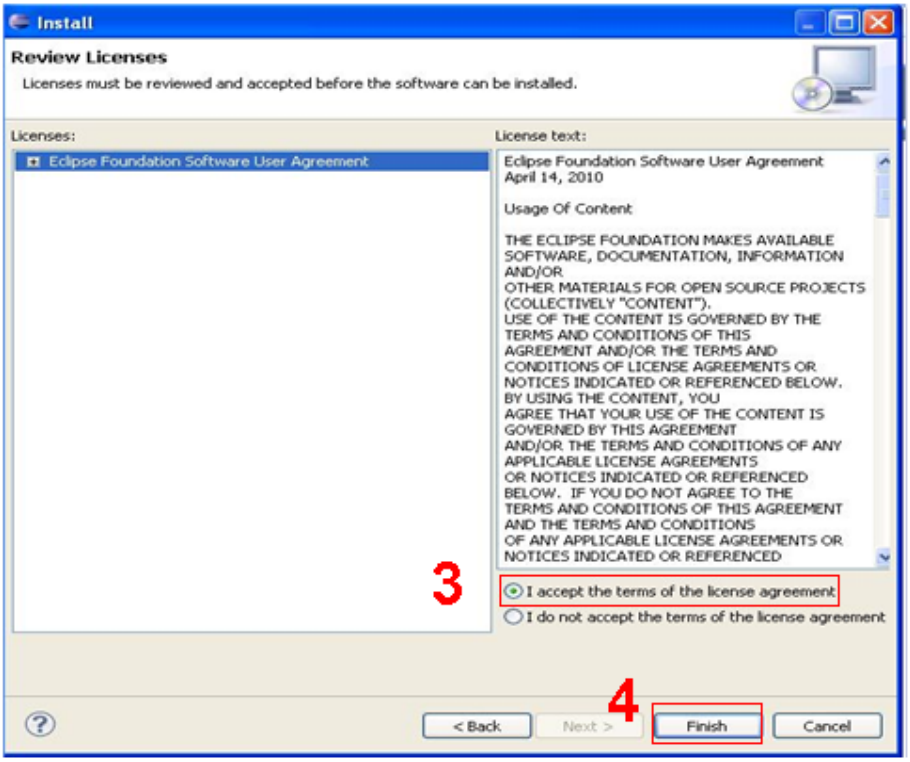


“OK” ボタンをクリックすると、以下のウィンドウが表示されます。”CDT Main Features” と ”CDT Optional Features” の両方のリストを開きます。

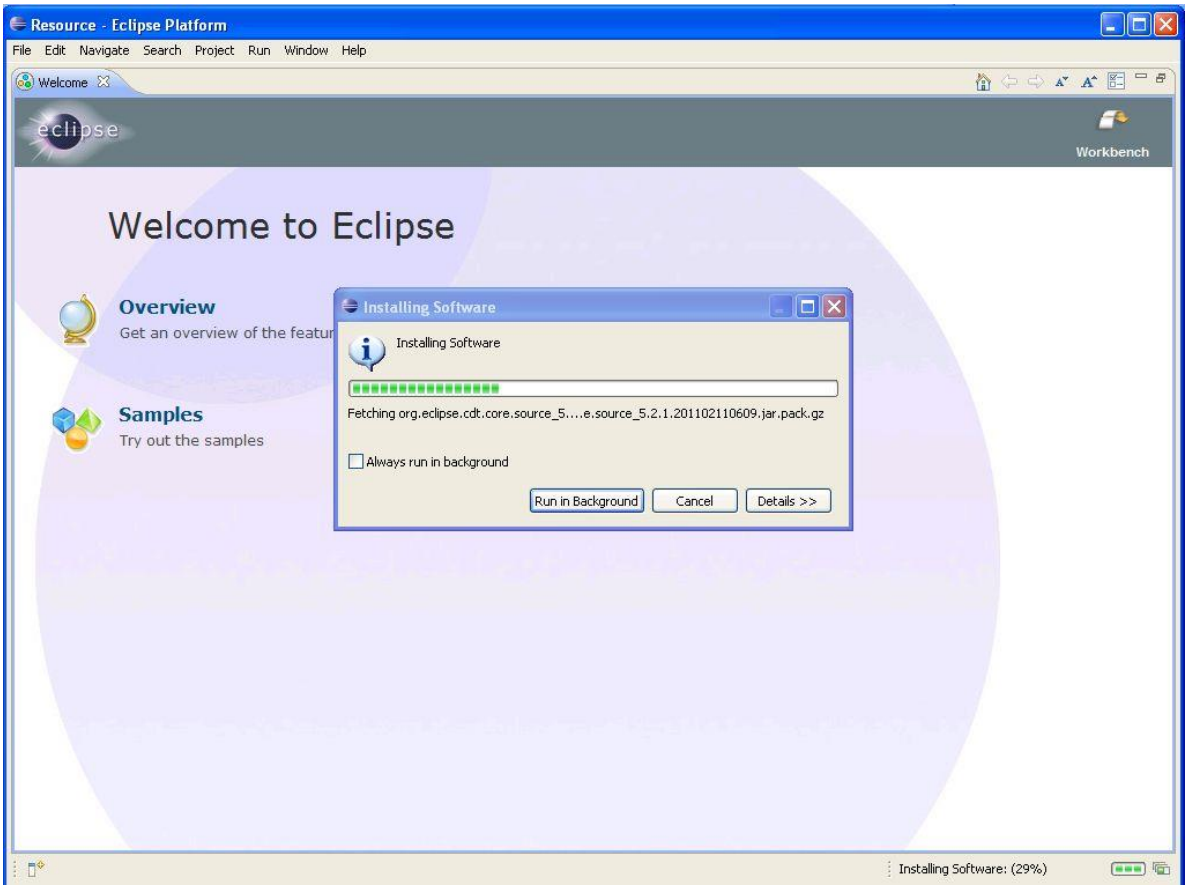


次に示すとおりのステップを実行してください。





Eclipse が CDT プラグインのインストールを開始します。



プラグインのインストールが終わったら、Eclipse IDE を再起動します。



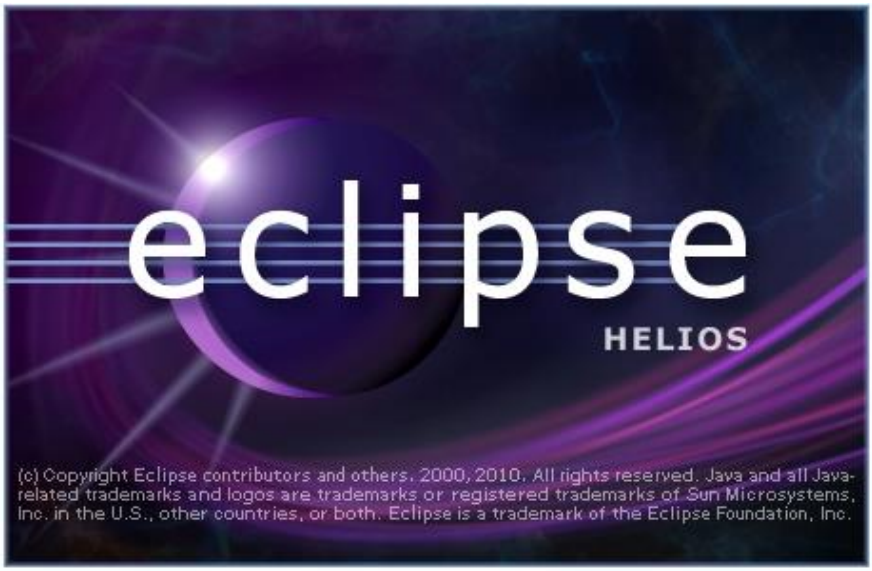
9 Eclipse IDE の使用方法

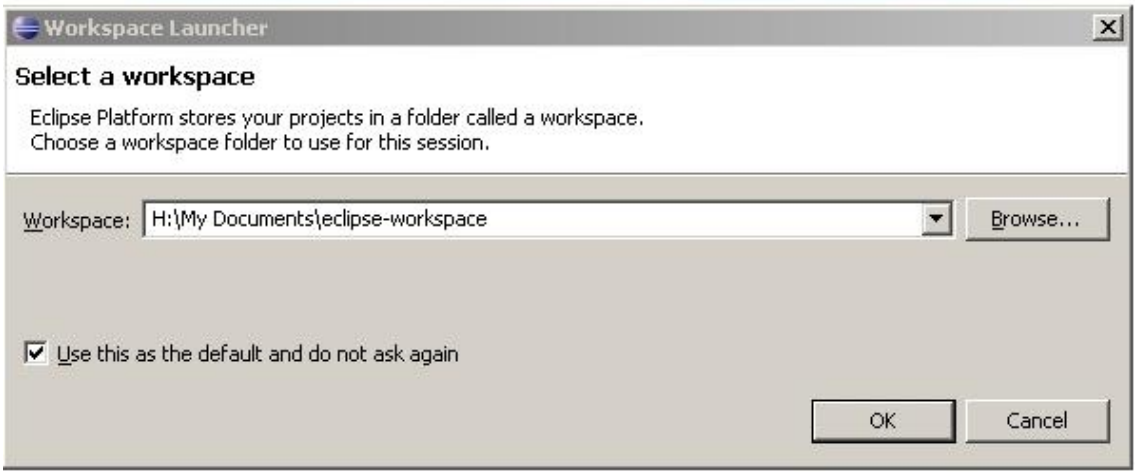
9.1 C/C++パースペクティブ

Eclipse IDE を起動します。

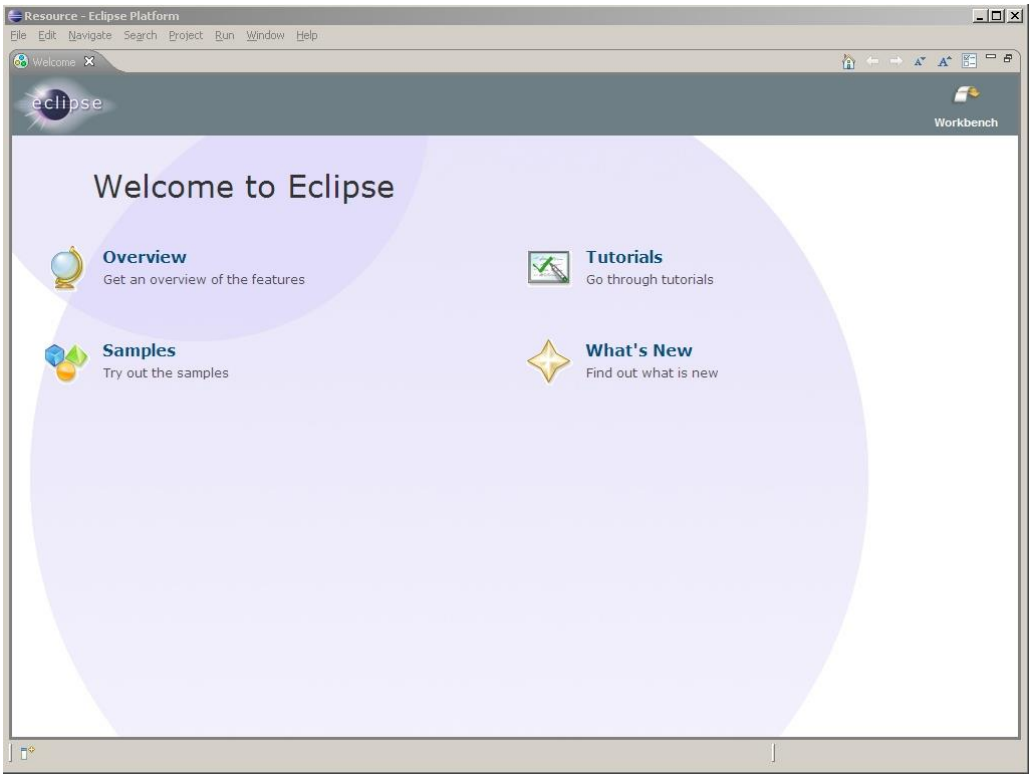


以下のような ”Workspace Launcher” ダイアログが表示されます。ここには、Eclipse CDT プロジェクトで使用するために指定した ”workspace” が設定されています (詳細は 7.2 章を参照)。



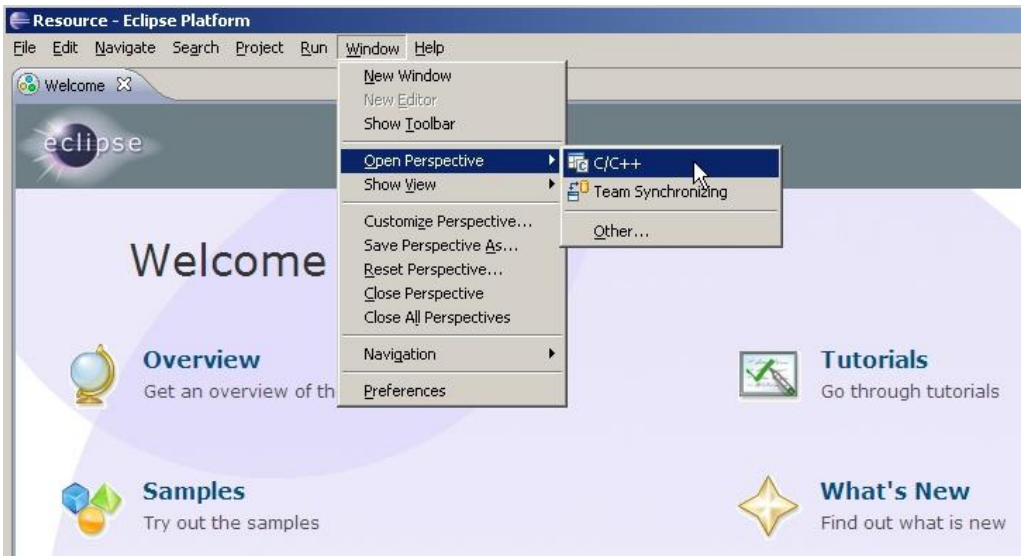


これで Eclipse が立ち上がり、以下の ”Welcome to Eclipse” ページが表示されます。



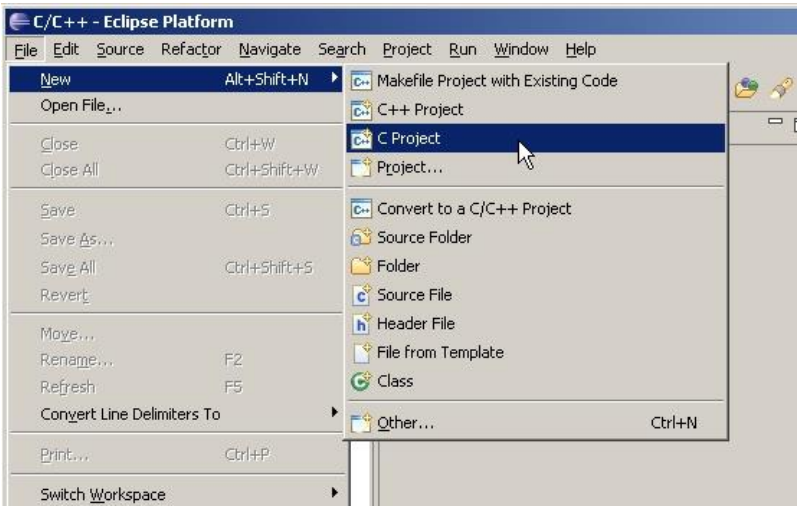
C/C++で開発する場合は、C/C++パースペクティブへ移動します。

“Window” -> ”Open Perspective” を選び、”C/C++” をクリックして C/C++パースペクティブを開きます。

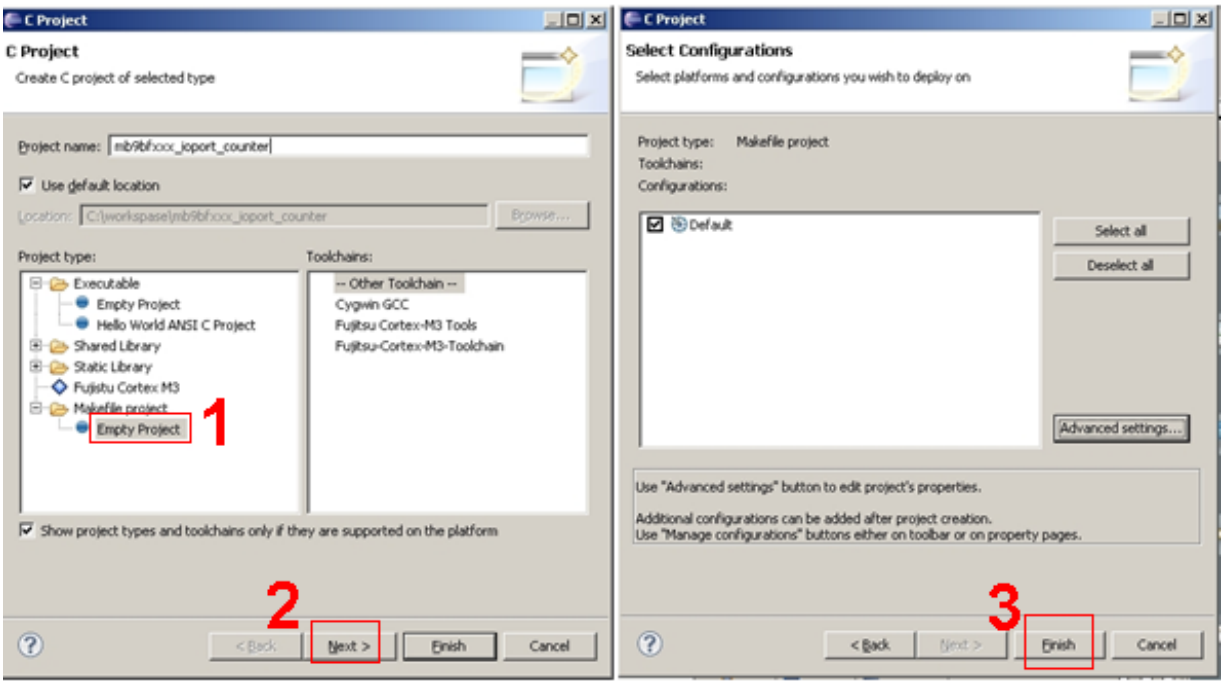


9.2 C/C++プロジェクトの作成

Eclipse C/C++パースペクティブから、ターゲット (ここでは FM3 ファミリ) のプロジェクトを作成できます。”File” -> ”New” -> ”C Project”を選択します。

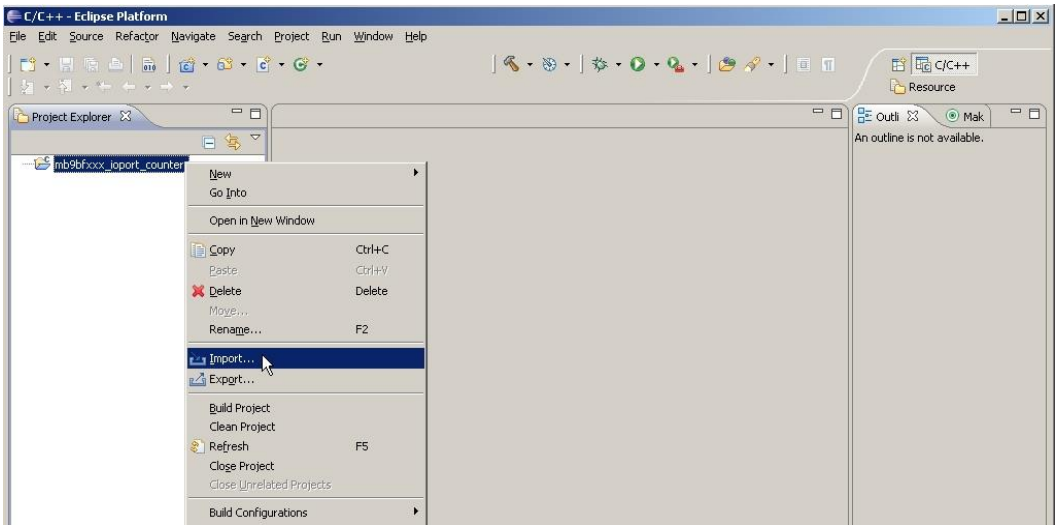


以下の左の図のとおり、“New Project wizard” で ”Makefile Project” の ”+” をクリックし ”Empty Project” を選びます。プロジェクト名 (例えば ”mb9bfxxx_ioport_counter”) を入力し ”Next” をクリックします。次に右の図のとおり ”Finish” を選択します。

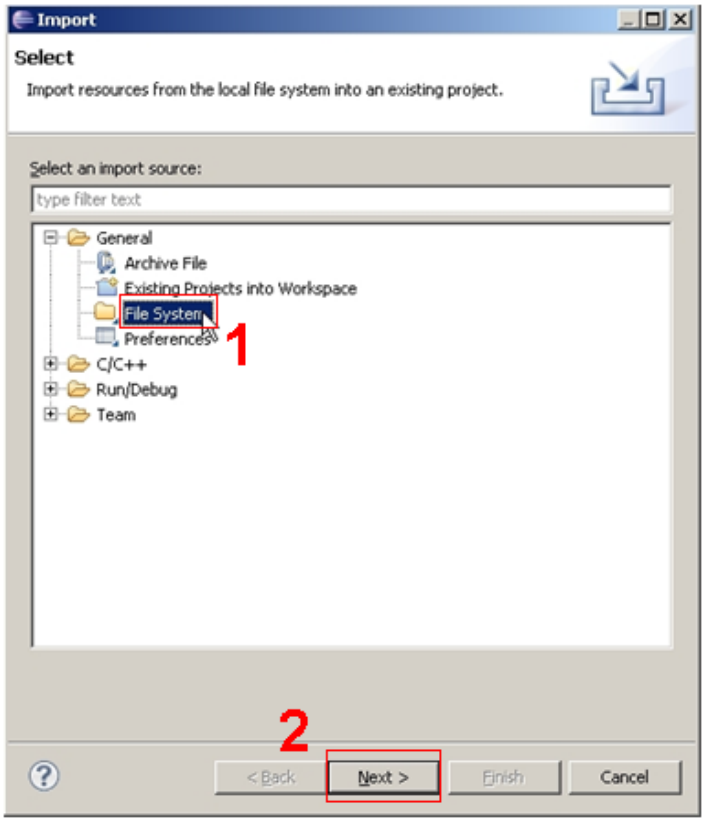


以下に示すとおり、左側に C/C++プロジェクトが表示されますが、まだソースファイルは含まれていません。大抵の場合、”File” -> ”New” -> ”Source File” を選びファイル名を入力後にコーディングを開始します。しかし本書では既存のソースファイルをインポートします。

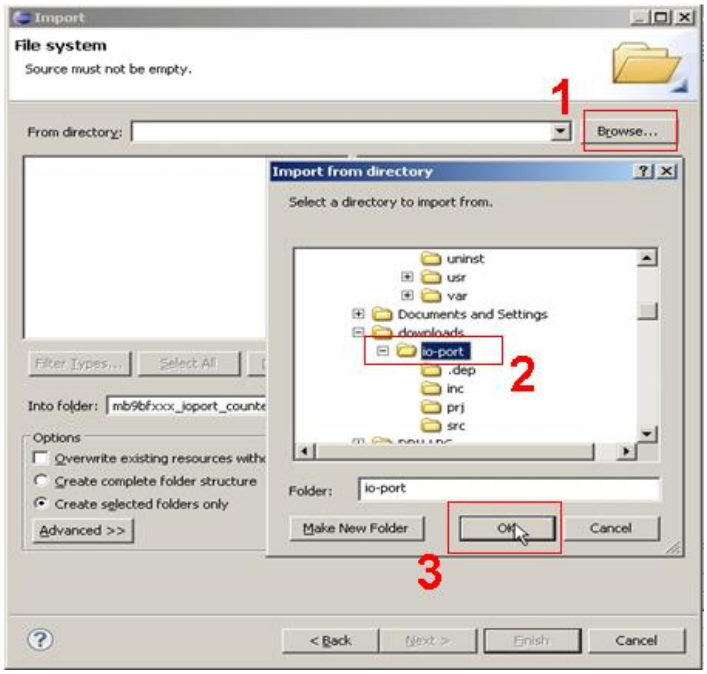
“File” -> ”Import…” をクリックし、ファイルのインポートウィンドウを表示させます。



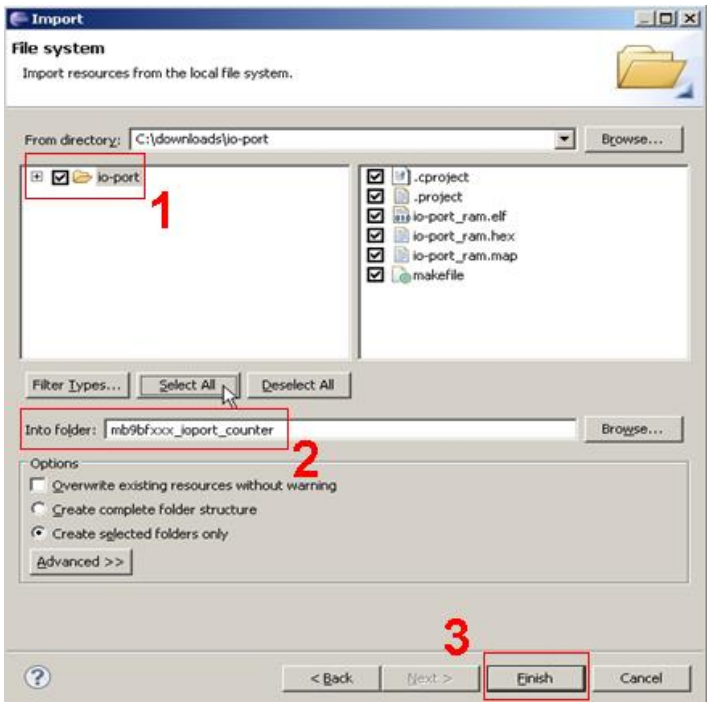
“File System” を選択し、”Next” をクリックします。



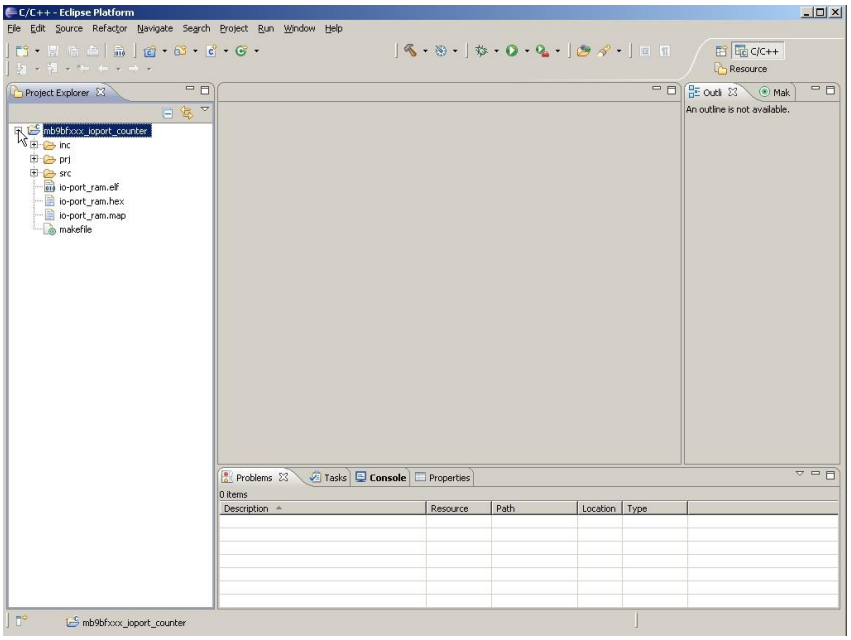
“From directory” テキストボックスへインポートしたいプロジェクトを入力するため、”Browse” ボタンを選択します。本書では io-port というテンプレートプロジェクトを使用します。本プロジェクトにはソフトウェアパッケージが含まれています。あらかじめ io-port プロジェクトを保存しておいてください (例えば C:¥downloads¥io-port)。



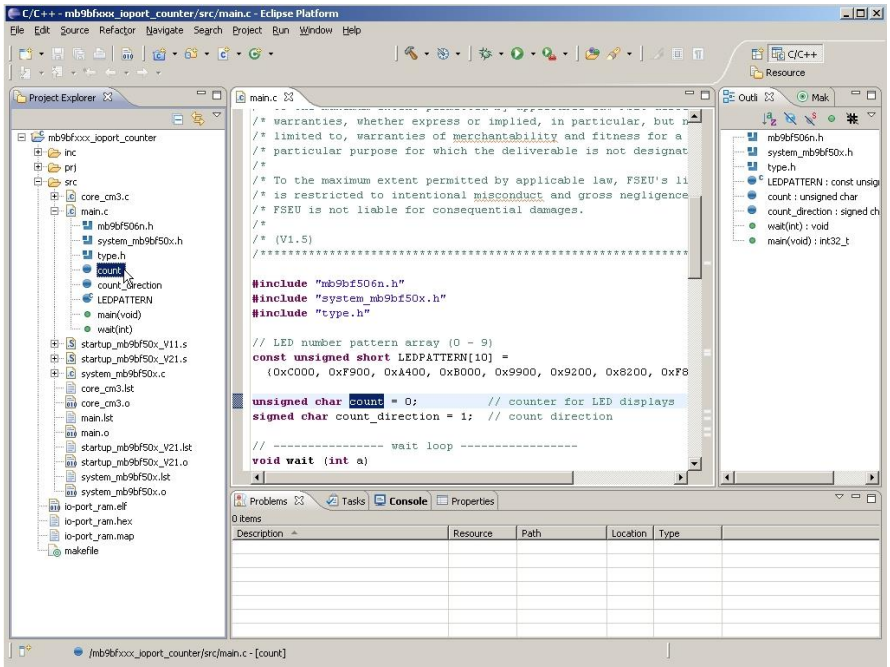
io-port フォルダを選択後、以下のとおり ”Select All” ボタンをクリックします。フォルダ内の全ファイルを使用するためです。”Finish” をクリックすると、ファイルのインポート処理が開始されます。



以下のように C/C++プロジェクトに mb9bfxxx_ioport_counter が表示されます。ここにはインポートしたファイルがすべて表示されます。プロジェクト名の ”+” をクリックすることでインポートファイルをツリーで開くことができます。

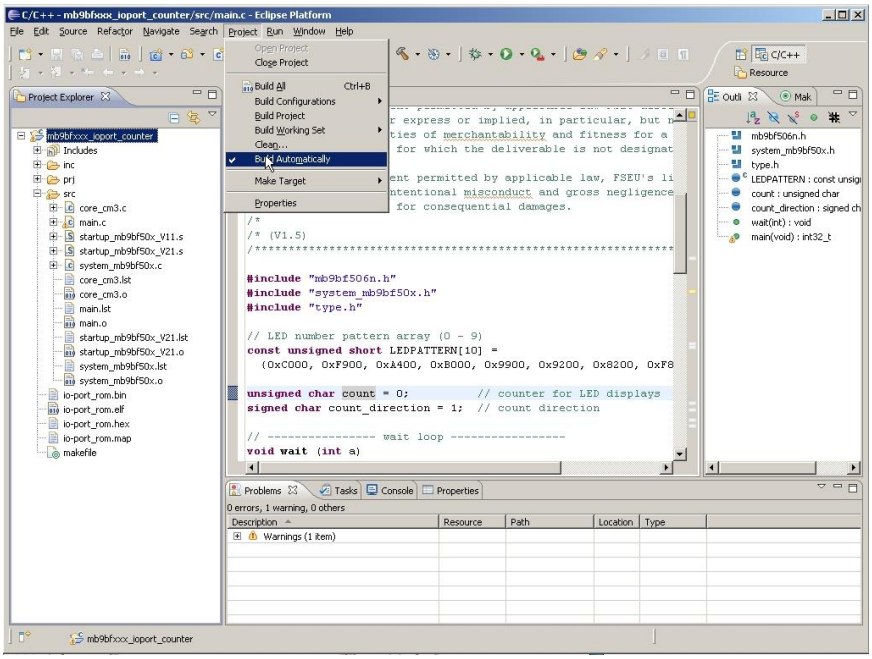


以下の図では、main.c ファイルをクリックして中央のエディタビューワに表示させています。”Project Explorer” では main.c の変数と関数が表示されます。例えば変数 count をクリックすることで、エディタビューワ内の count が定義された箇所へジャンプします。

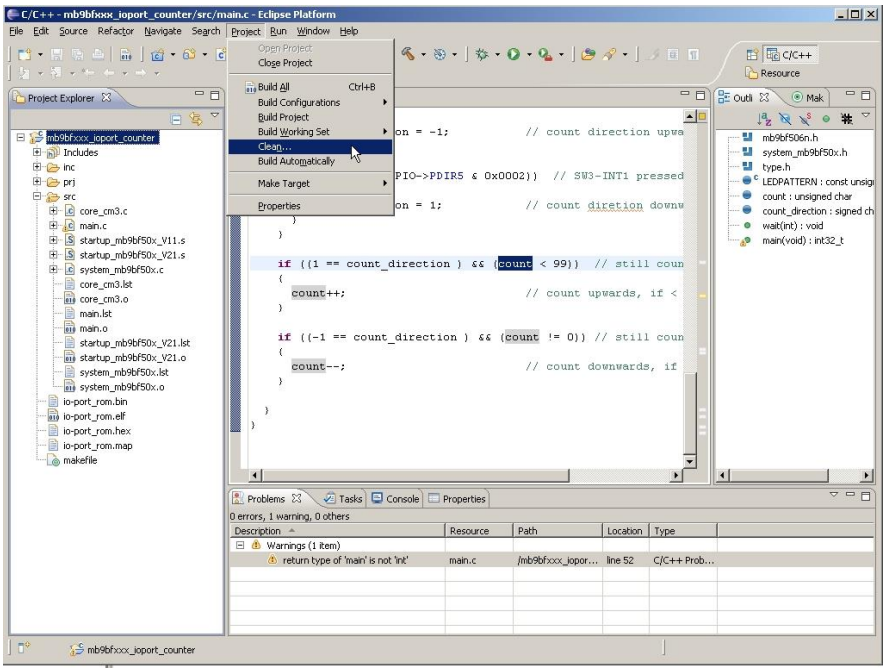


9.3 プロジェクトのクリーニング

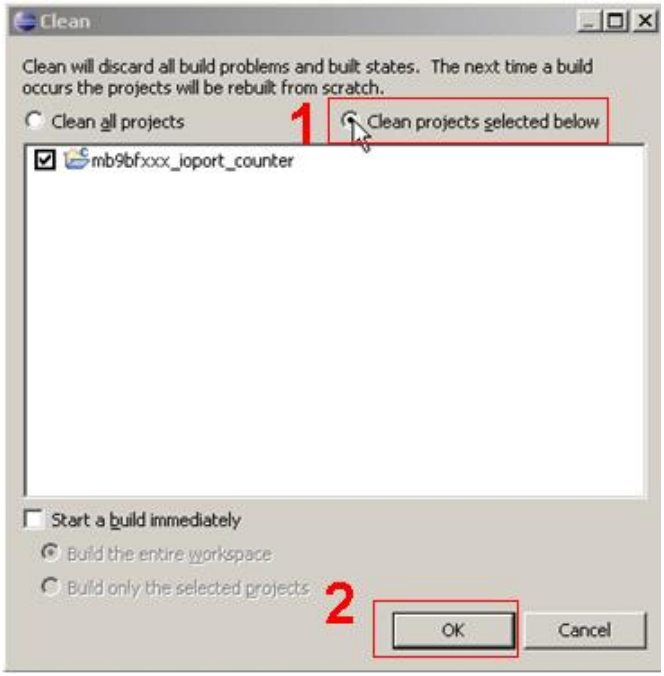
プロジェクトをコンパイルするためには、まず自動ビルドを無効にします。IDE の ”Project” メニューから ”Build Automatically” のチェックを外してください。



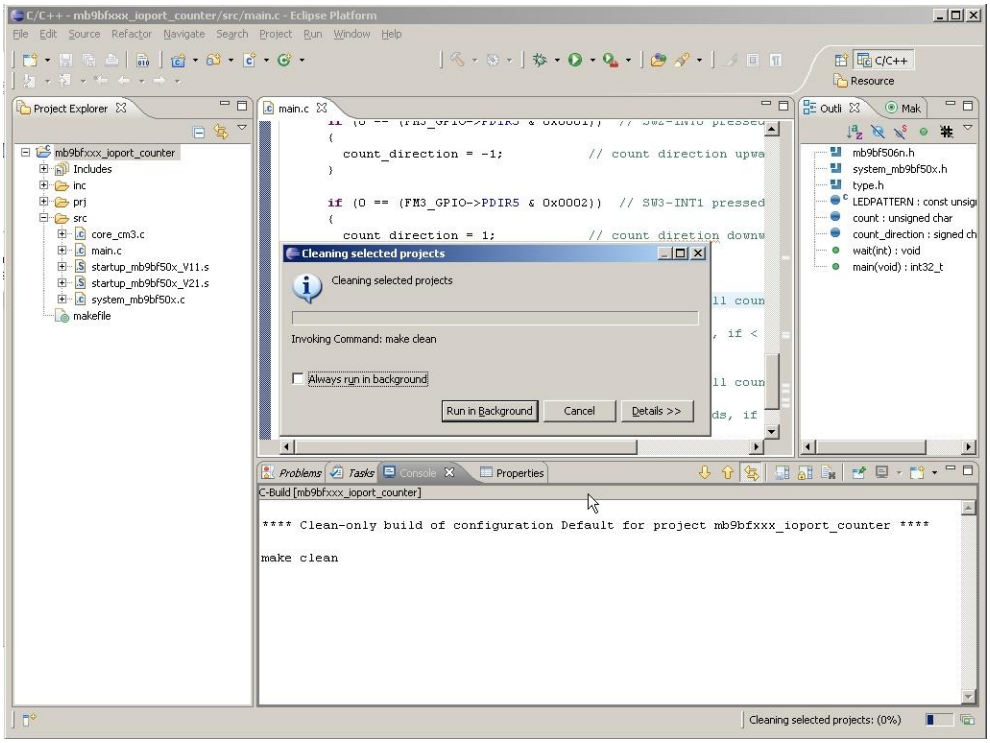
プロジェクトをクリーンします。Project Explorer から mb9bfxxx_ioport_counter を選択し、”Project” メニューから ”Clean…” をクリックします。



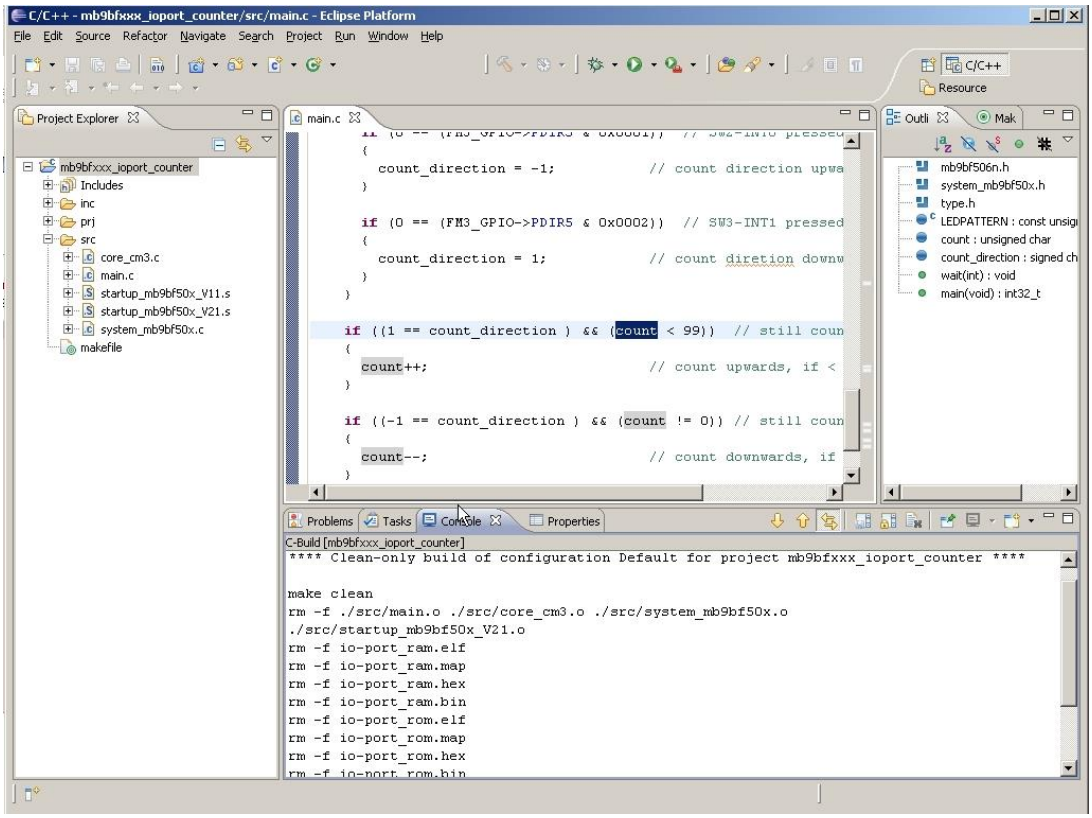
Clean ウィンドウから ”Clean all projects” のオプションを外し、目的のプロジェクトを選択します。”Start a build immediately” のオプションも外してください。



“OK” ボタンをクリックすることでコンフィグレーションが完了し、クリーン処理が始まります。



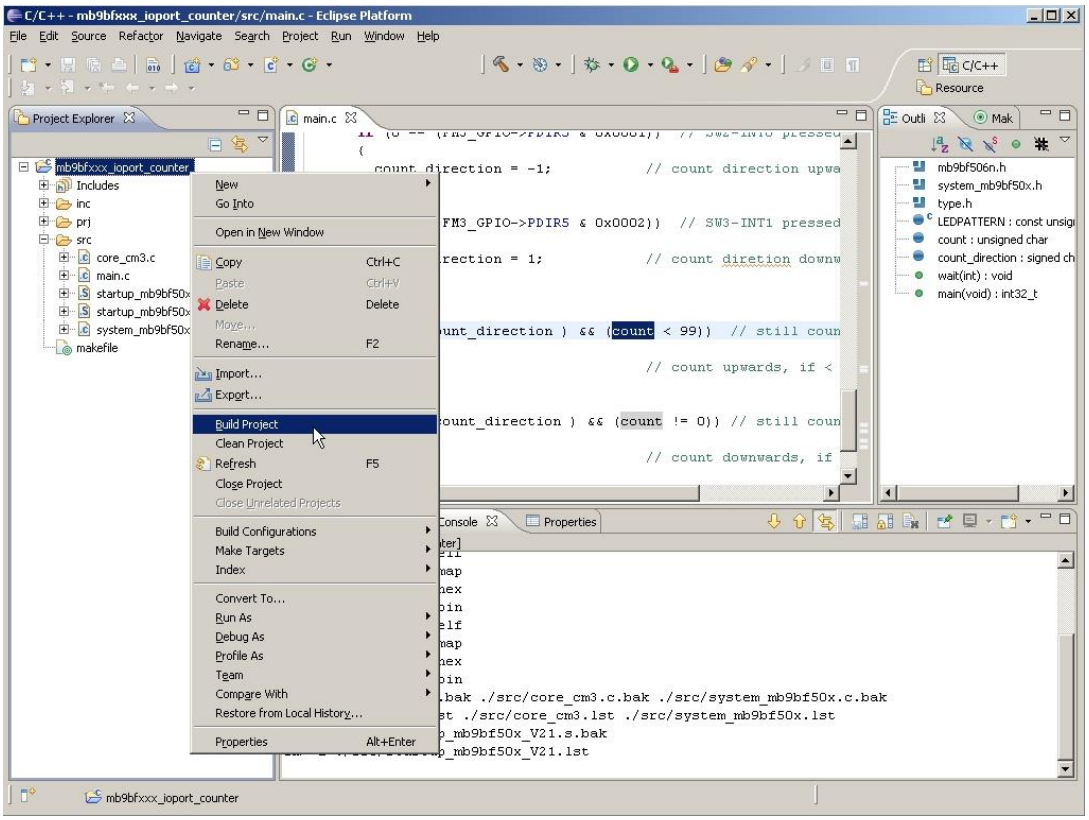
コンソールを開くことで、クリーン処理の結果を見ることができます。



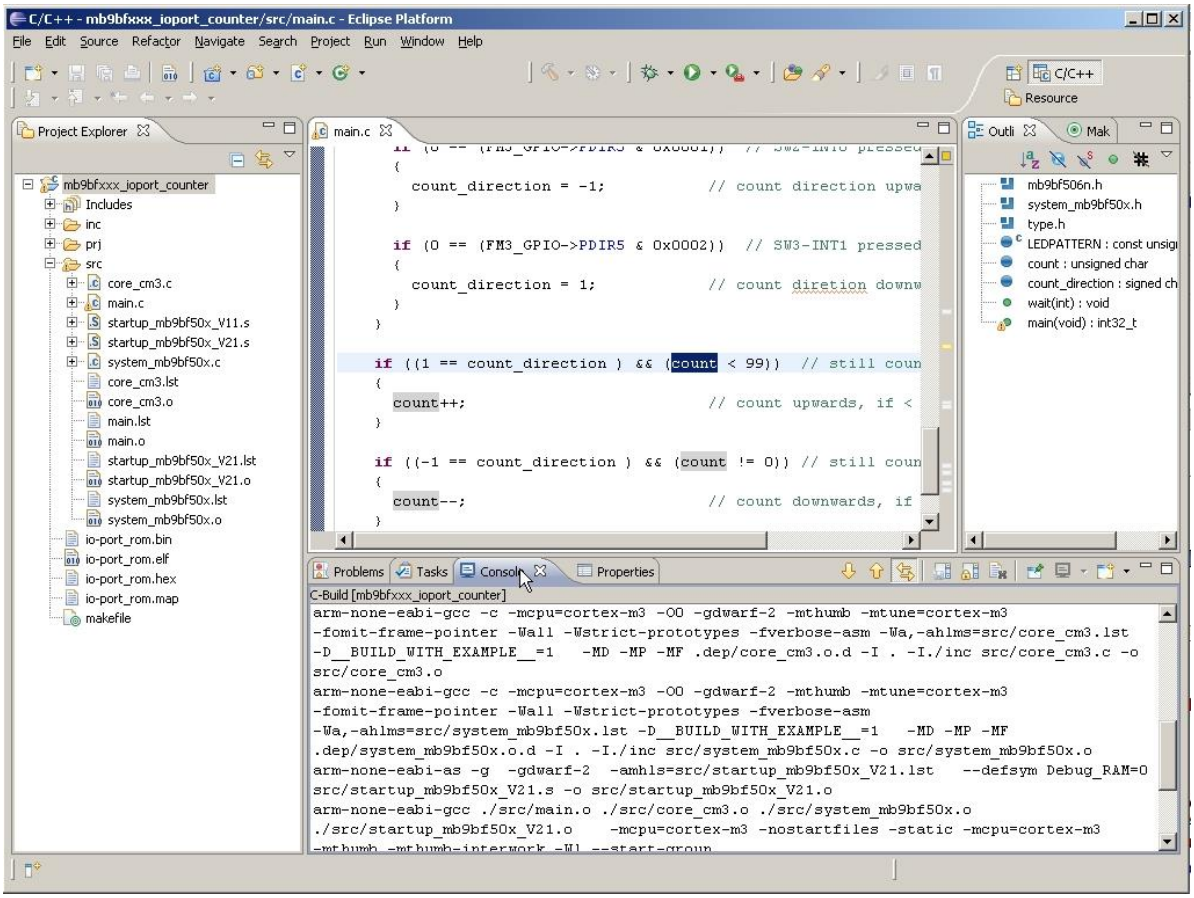
9.4 プロジェクトのビルド

[備考] 本書で使用するソフトウェアパッケージの makefile を使用する場合、すべてのパス (例えば OpenOCD へのパスなど) を確認し、インストールパスを個別に設定してください。

mb9bfxxx_ioport_counter プロジェクトを、既にインストールしている YAGARTO ツールチェーンでコンパイルできます。これには、Project Explorer から mb9bfxxx_ioport_counter を選択し、右クリックで表示されたリストから ”Build Project” を選択します。



以下のように、コンソールから結果を見ることができます。



Project Explorer に、プロジェクトのアウトプットファイル (*.bin、*.elf など) が生成されています。

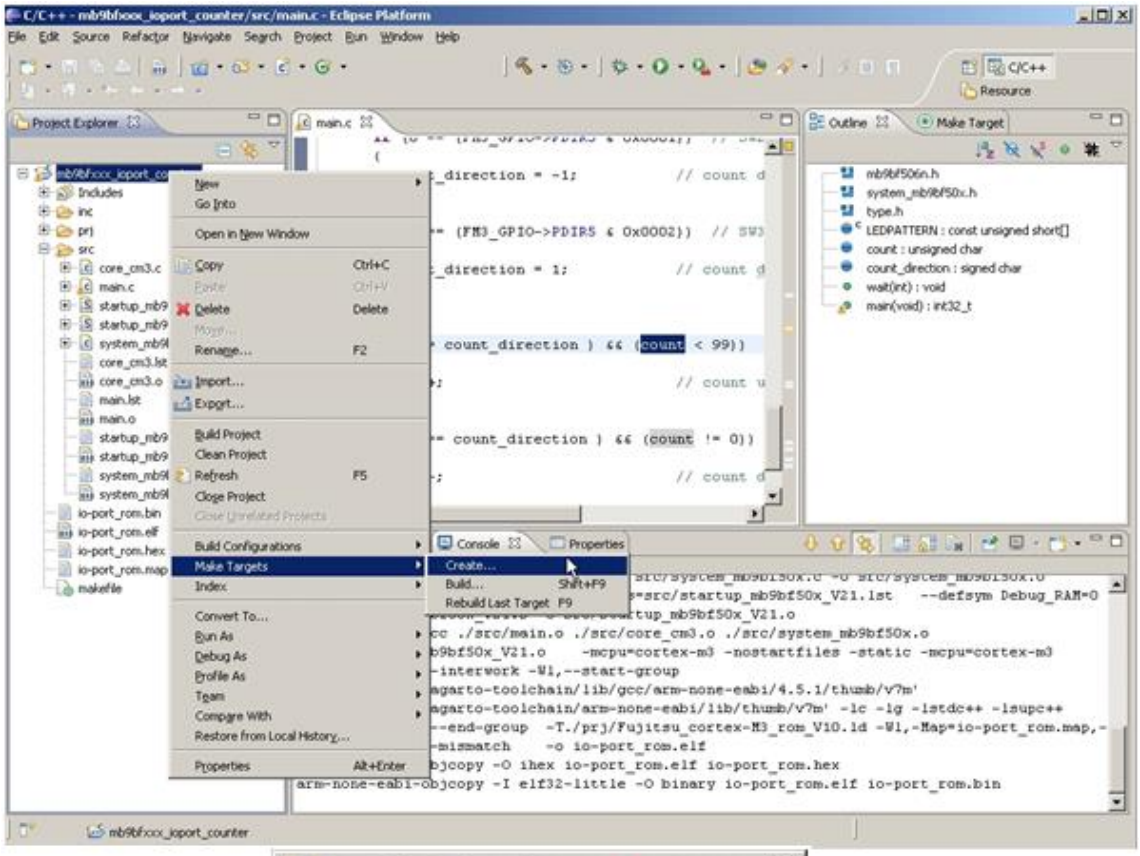
9.5 メイクターゲットの作成

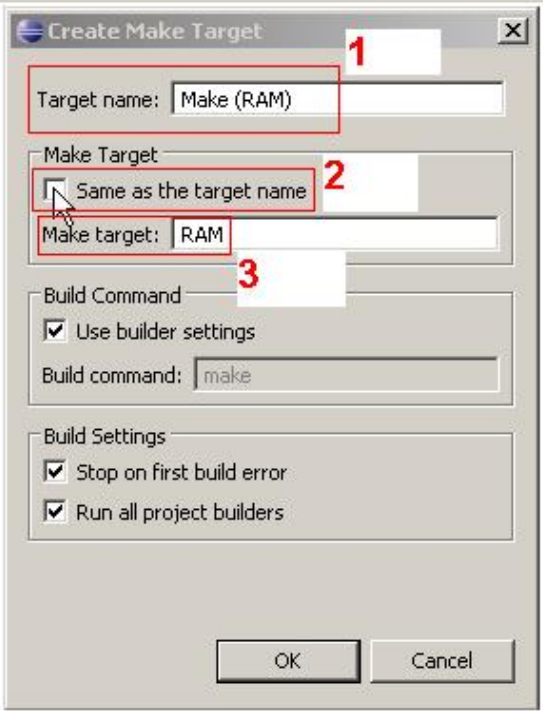
メイクターゲットは既に mb9bfxxx_ioport_counter プロジェクトに設定されています。新しいプロジェクトをセットアップした、もしくはターゲットを削除してしまった場合に本章の作成方法を参照してください。

mb9bfxxx_ioport_counter の makefile がビルドプロセスを構成します。makefile は、RAM / ROM 上でデバッグするためのファイルを生成します。また、Cypress Flash Programmer のようなツールを使用した Flash への書込みをするためのファイルも生成します。

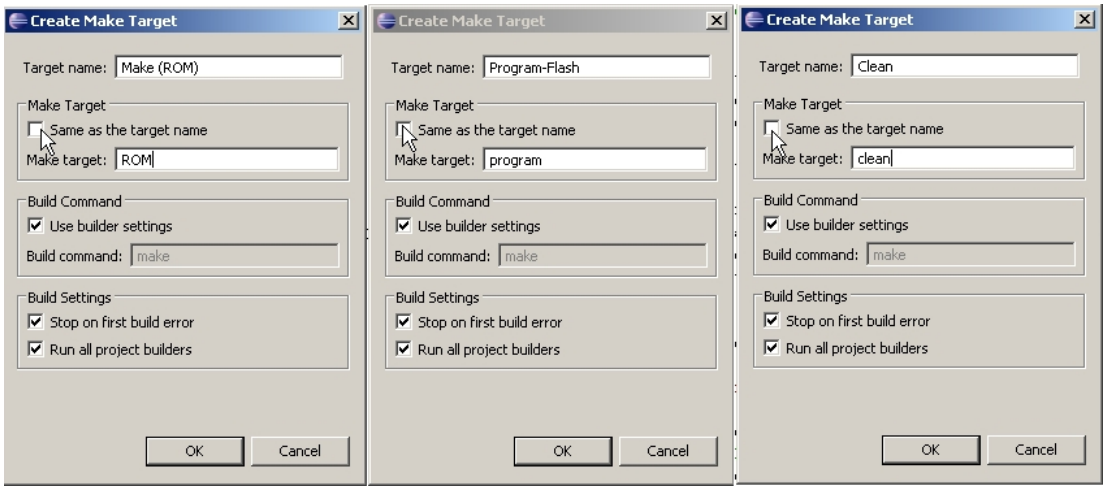
メイクターゲットは RAM と ROM (Flash) で別々に作成する必要があります。また本書ではクリーン処理も追加します。

メイクターゲットを作成するためには、Project Explorer から mb9bfxxx_ioport_counter を選択し、右クリックのメニューから ”Make Target” を選択します。

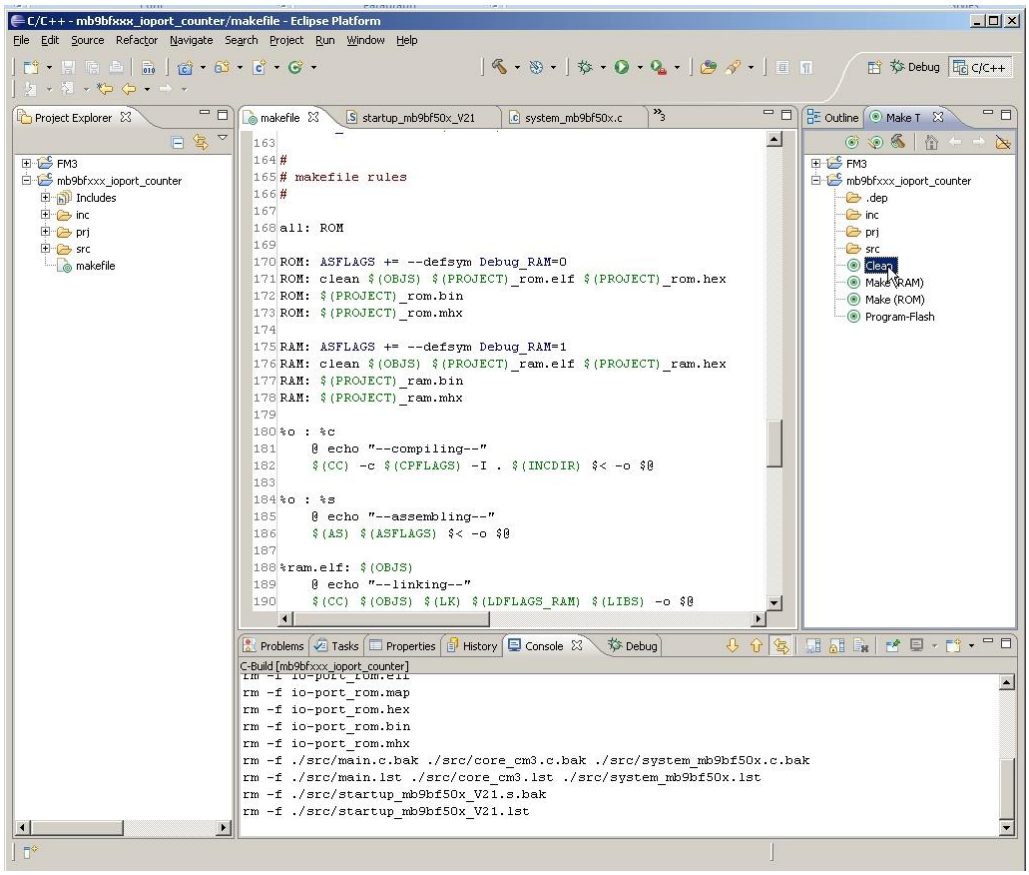

C/C++ - mb9bf5xx_ioport_counter/src/main.c - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
main.c
Outline
Make Target
New
Go Into
Open in New Window
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Move...
Rename... F2
Import...
Export...
Build Project
Clean Project
Refresh F5
Close Project
Close Unrelated Projects
Build Configurations
Make Targets
Index
Convert To...
Run As
Debug As
Profile As
Team
Compare With
Restore from Local History...
Properties Alt+Enter
Console
Properties
Create...
Build... Shift+F9
Rebuild Last Target F9
Includes
inc
prj
src
core_cm3.c
main.c
core_cm3.lst
core_cm3.o
main.lst
main.o
io-port_rom.bin
io-port_rom.elf
io-port_rom.hex
io-port_rom.map
makefile
mb9bf506n.h
system_mb9bf50x.h
type.h
LEDPATTERN : const unsigned short[]
count : unsigned char
count_direction : signed char
wait(int) : void
main(void) : int32_t
arm-none-eabi-objcopy -O ihex io-port_rom.elf io-port_rom.hex
arm-none-eabi-objcopy -I elf32-little -O binary io-port_rom.elf io-port_rom.bin


Create Make Target
1
Target name: Make (RAM)
Make Target
2
Same as the target name
Make target: RAM
3
Build Command
Use builder settings
Build command: make
Build Settings
Stop on first build error
Run all project builders
OK
Cancel

“Target name” に Make (RAM) と入力し、”Same as the target name” のチェックを外し、”Make target” に RAM と入力します。”OK” をクリックすることで ”Make (RAM)” を生成できます。同様に、”Make(ROM)”, ”Program-Flash”, ”Clean” を作成します。



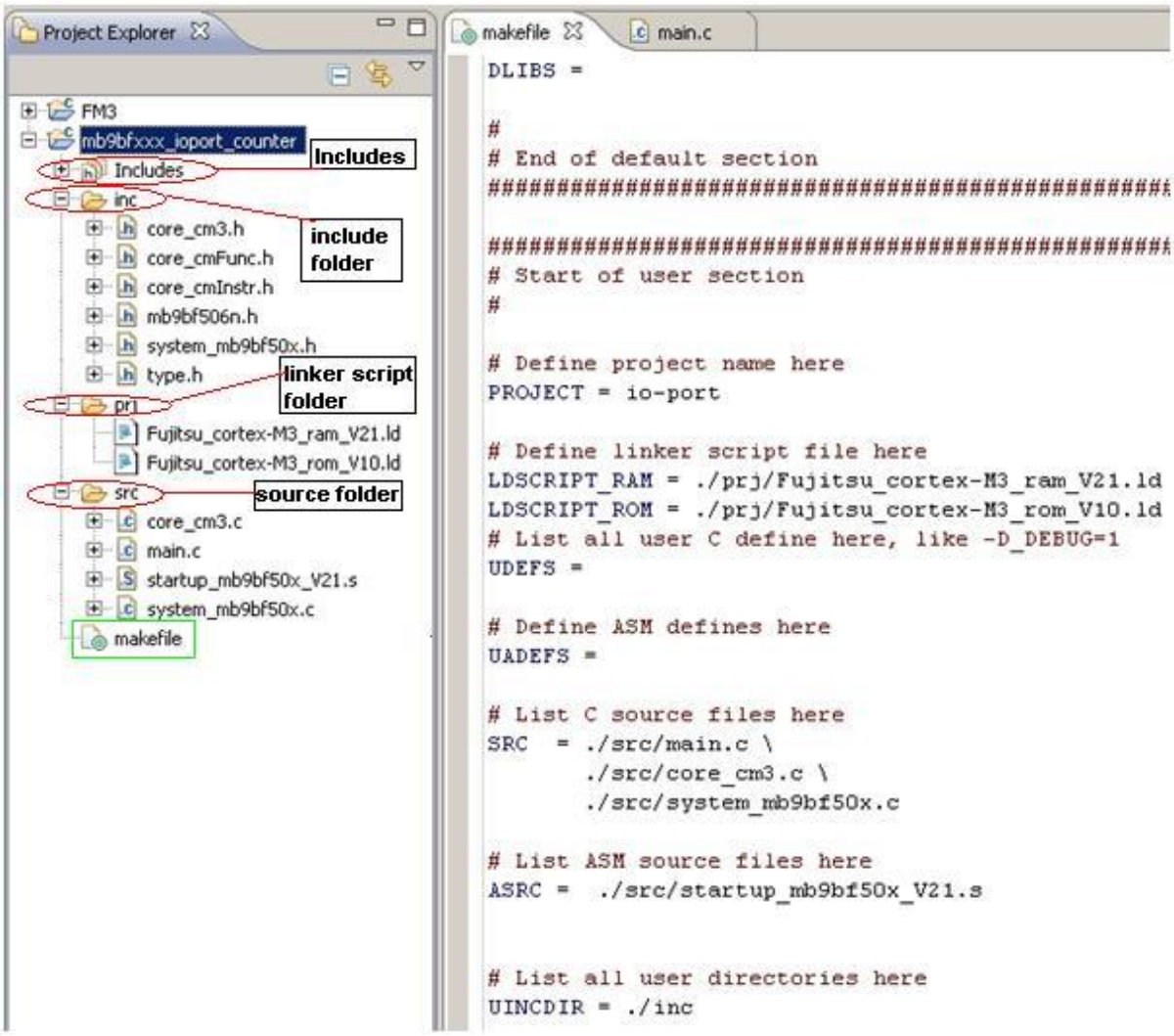
次の図に ”Make Target” ビューワが確認できます。ダブルクリックすることで、”Make (RAM)”, ”Make (ROM)”, ”Clean” のビルドプロセスを開始できます。



コンソールからプロセスの完了を確認できます。

10 サンプル Eclipse プロジェクト

本書で使用するプロジェクトは以下の構成になっています。



inc フォルダにすべてのプロジェクトで使用する I/O ヘッダファイルが含まれています。ここには、CMSIS とスタートアップのヘッダファイルも含まれています。prj フォルダにはリンカスクリプトファイル、src フォルダにはソースファイルが含まれています。

またサンプルプロジェクトには makefile も含まれています。

Includes フォルダにはビルドに必要な Yagarto ライブラリ (例えば stdint.h) が含まれています。

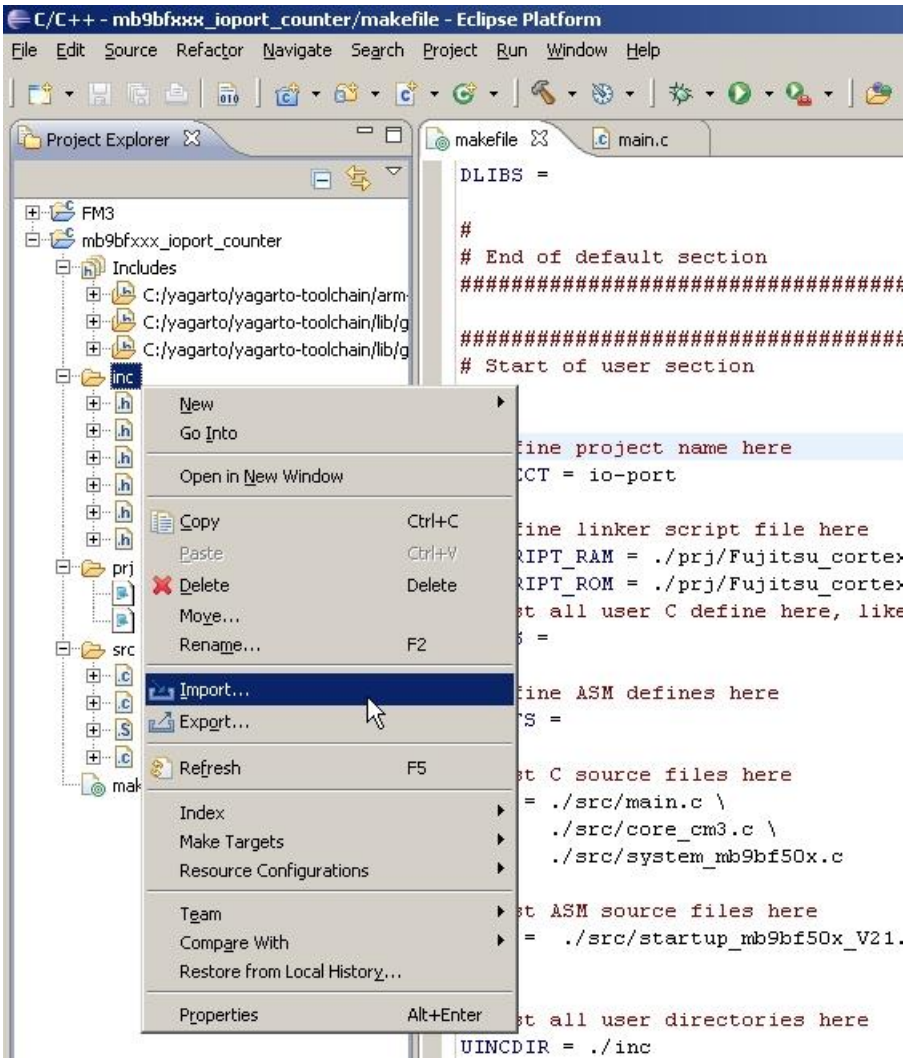
ほかのソースファイルは src フォルダに格納してください。

新しいヘッダファイルは inc フォルダもしくは、includes フォルダに格納できます。

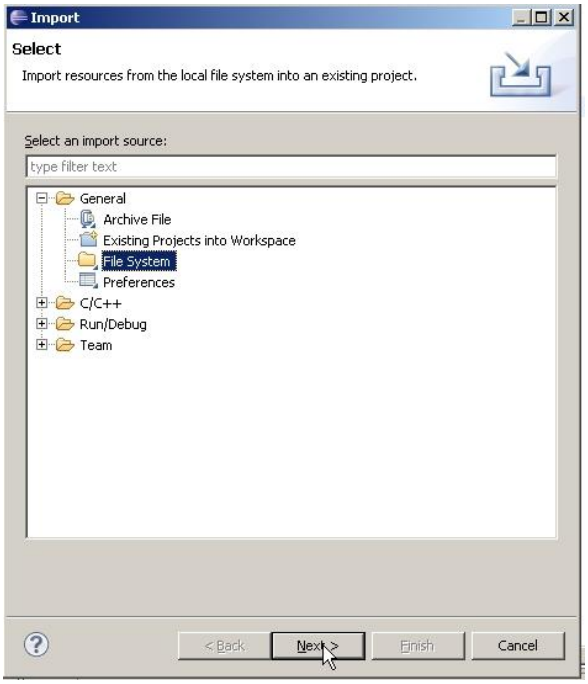
10.1 ファイルの追加

プロジェクトを開き、ファイルを追加したいフォルダを選択して右クリックします。

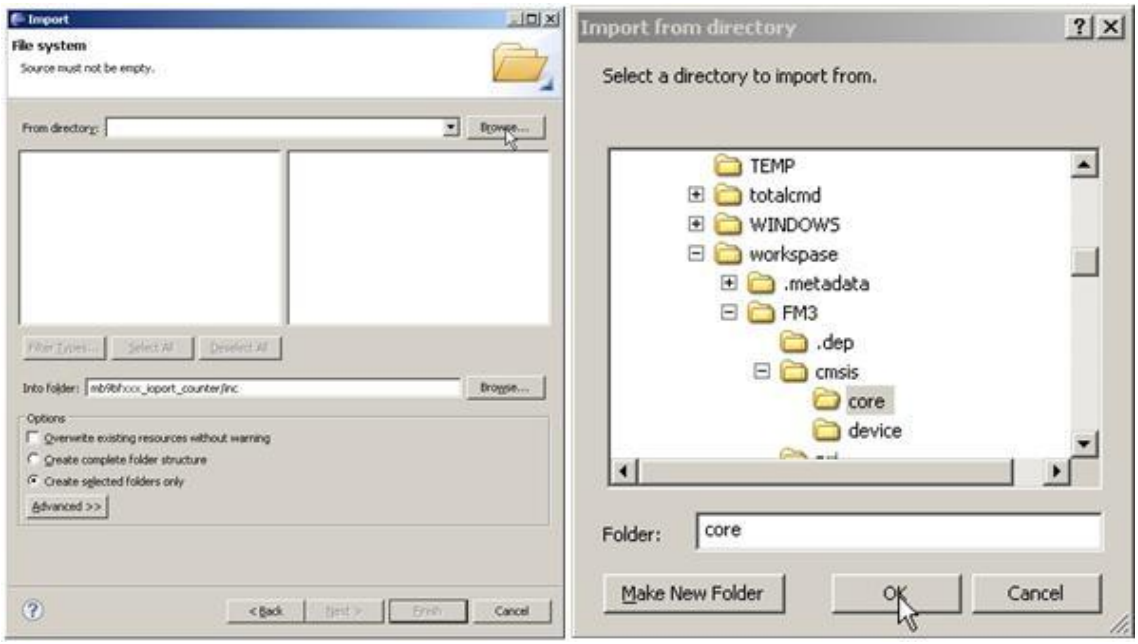
メニューから ”import” を選択してください。



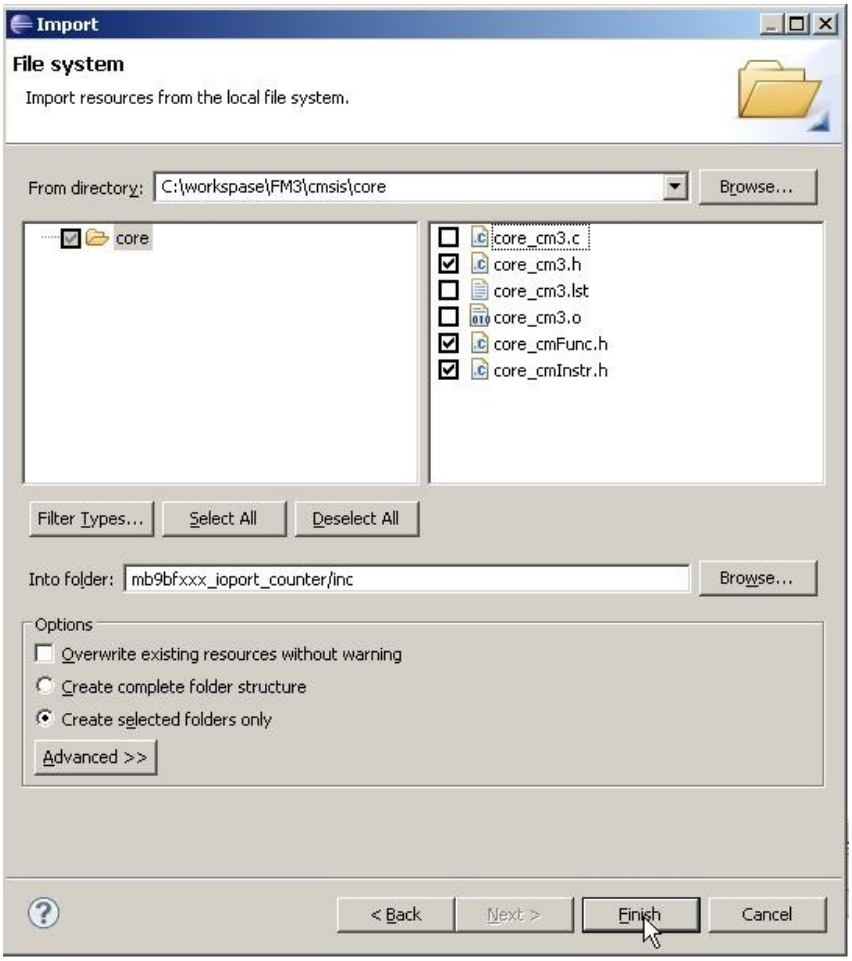
ファイルを選択し、”Next” ボタンをクリックしてください。



“Browse” ボタンをクリックし、追加するファイルを選択します。



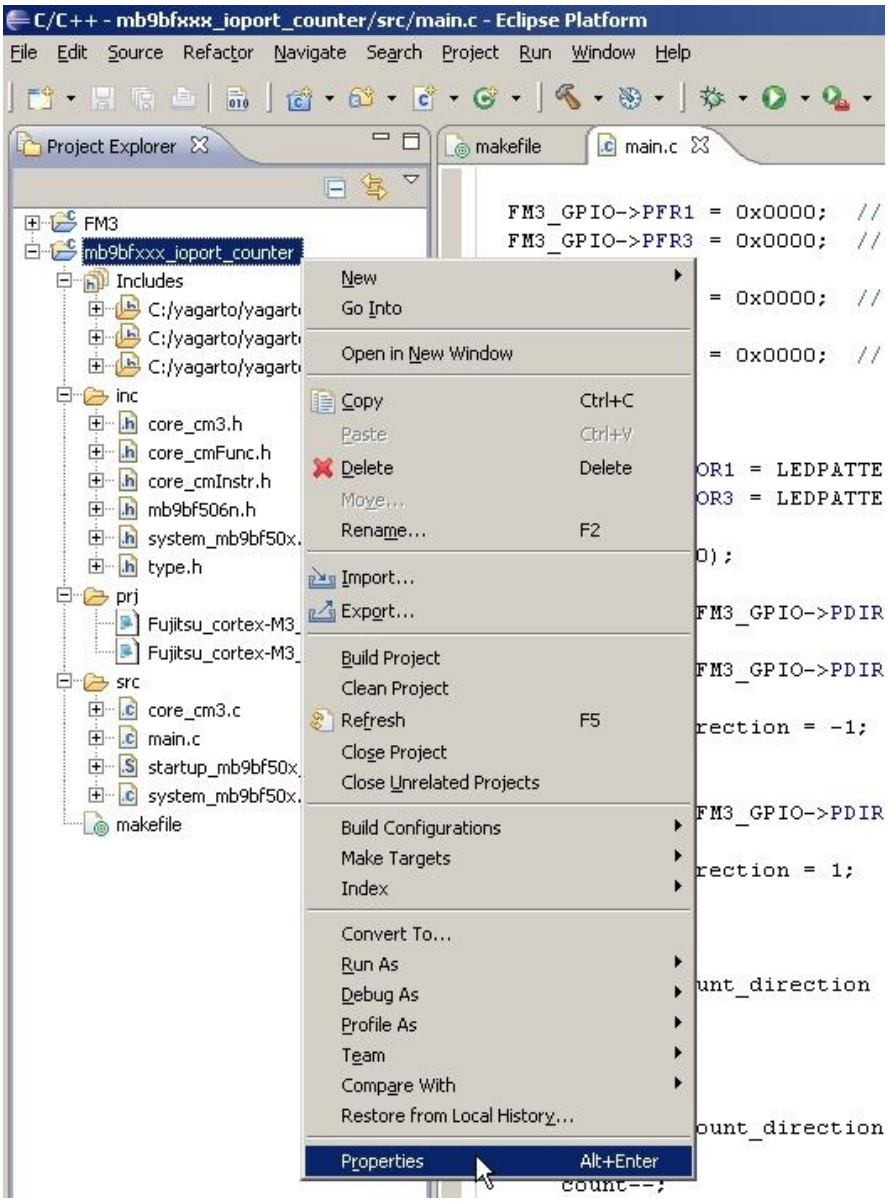
インポート後、リストにファイルが追加されていることを確認してください。



“Finish” ボタンをクリックすることで、選択したファイルを指定フォルダにインポートできます。

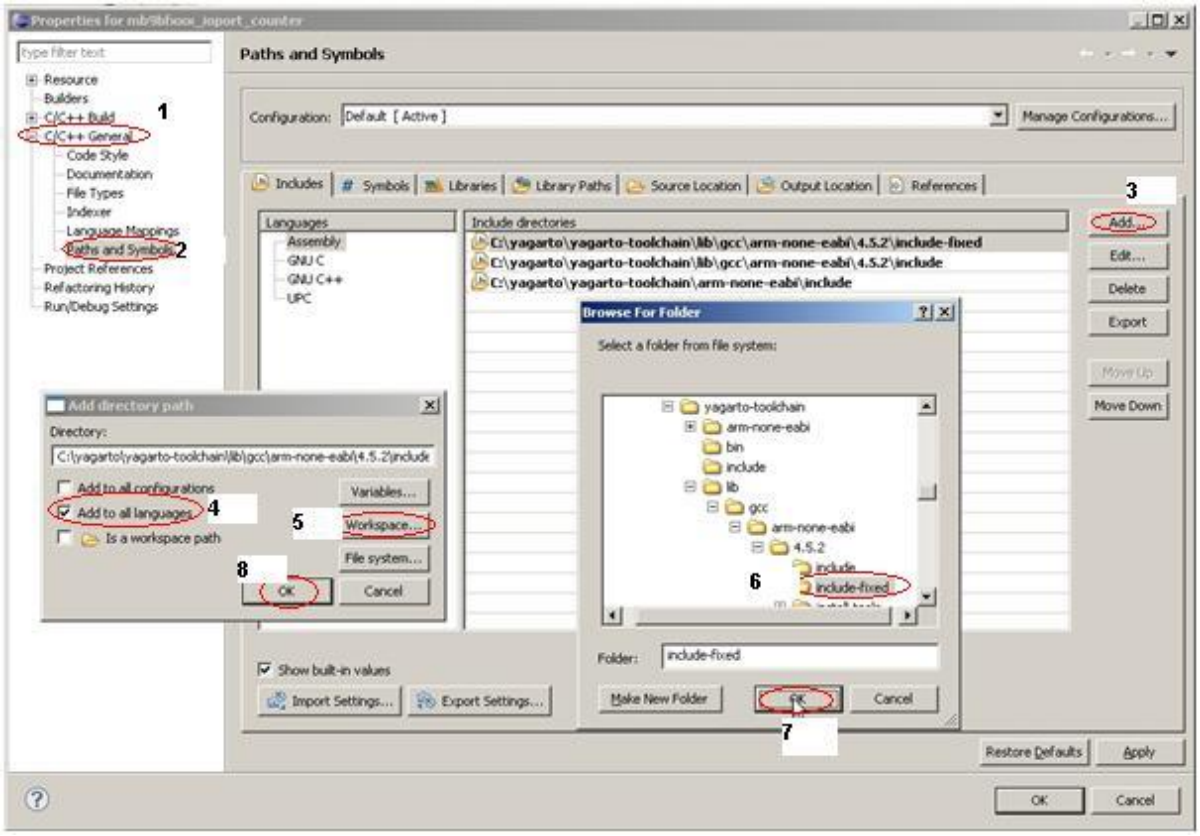
10.2 Includes フォルダへライブラリを追加

いくつかのライブラリヘッダ (例えば stdint.h) を、Yagarto インストールフォルダからインクルードする必要があります。プロジェクトを選択して右クリックし、”Proprieties” を選択します。ここで、選択したプロジェクトのコンフィグレーションを変更できます。

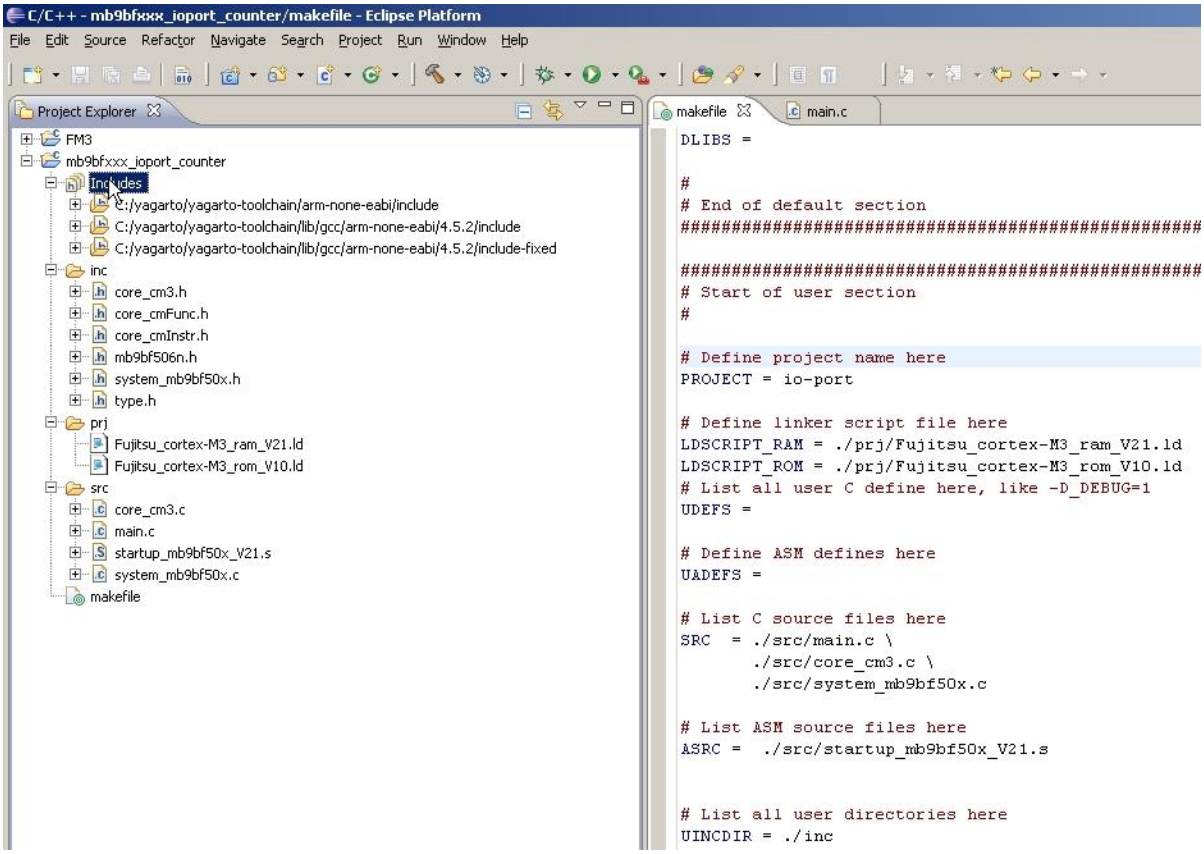


下記のとおり設定してください。

1. “C/C++ General” を選択
2. “Paths and Symbols” をダブルクリック
3. “Add” をクリック
4. “Add to all languages” にチェック
5. “File system” を選択しインクルードフォルダを指定
6. インクルードフォルダを選択
7. “browser” の ”OK” をクリック
8. “Add directory path” の ”OK” をクリック



新しいライブラリフォルダが Includes フォルダに追加されます。



10.3 makefile

makefile は GNU メイクツールに対する多くのインストラクションから構成されています。これらのインストラクションは、ユーザにとって必要な情報を設定してプロジェクトをメイクするために使用されます。makefile は本書で使用するソフトウェアパッケージに含まれています。

makefile のインストラクションの詳細を下記に説明します。本章ではインストラクションを意味ごとに複数のパートに分けています。

最初のパートでは、GUN ツールが必要とするコンパイラ (arm-none-eabi-gcc-.exe), アセンブラ (arm-none-eabi-as.exe), リンカ (arm-none-eabi-ld.exe), をプロジェクトに設定します。コンパイラ, アセンブラによって生成されたファイルはオブジェクトファイル (*.o) とよばれます。コンパイラとアセンブラに加え、GNU ツール (arm-none-eabi-objcopy.exe) からアウトプットファイル (*.elf), リンカから他フォーマット (例えば hex ファイル (*.hex) もしくはバイナリファイル (*.bin)), を生成するための設定が必要です。

```
TRGT = arm-none-eabi-
CC   = $(TRGT)gcc
AS   = $(TRGT)as
LD   = $(TRGT)ld -v
CP   = $(TRGT)objcopy
```

本章では、必要とされるすべての GNU ツールがインストールされており、3 章のインストール手順によって Yagarto が windows のパスに追加されていることが前提です。これらのツールは、Yagarto GNU ARM ツールチェーンのインストールフォルダの bin フォルダに格納されています。

次のパートは、GNU リンカが生成したアウトプットファイル (*.elf) からほかのフォーマットを生成するために必要な GNU Objcopy ツールのオプションです。

最初の行は Intel フォーマットである hex ファイル (*.hex) を生成します。2 行目はバイナリファイル (*.bin) を、最後の行では Motorola S-record の hex ファイル (*.mhx) を生成します。

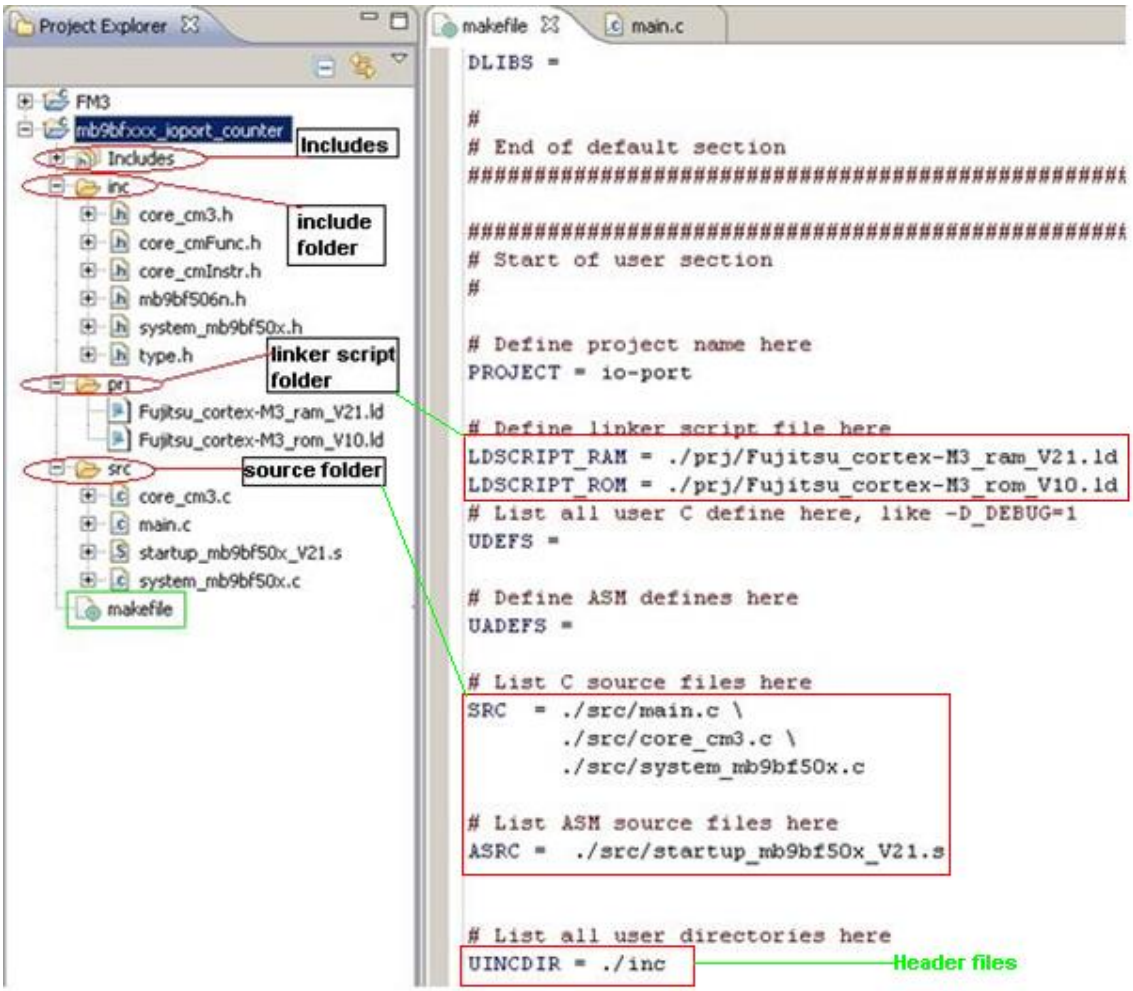
```
HEX  = $(CP) -O ihex
BIN  = $(CP) -I elf32-little -O binary
SREC = $(CP) -O srec
```

次の行ではすべてのプロジェクト名を定義します。定義した名前は、GNU リンカと GNU Objcopy ツールにより他フォーマットへコピーされたアウトプットファイルに使用されます。

```
# Define project name here
PROJECT = io-port
```

サンプル Eclipse プロジェクトは下記のプロジェクトフォルダを含んでいます。

- **inc** : すべてのインクルードファイルを格納
- **prj** : すべてのリンカスクリプトファイルを格納
- **src** : すべてのソースファイル (*.c と*.s) を格納



本フォルダ構成は下記のとおり定義できます。

```
# Define linker script file here
LDSCRIPT_RAM = ./prj/MB9BFD18_ram.ld
LDSCRIPT_ROM = ./prj/MB9BFD18_rom.ld

# List C source files here
SRC  = ./src/main.c ¥
       ./src/core_cm3.c ¥
       ./src/system_mb9bfd1x.c

# List ASM source files here
ASRC = ./src/ startup_mb9bfd1x.s

# List all user directories here
UINCDIR = ./inc
```

次のパートは使用しません。ユーザがほかの定義やライブラリモジュールの追加を意図的にする場合に、本パートを使用します。

```
#######################################################################
# Start of default and user defines
#

# List all default C defines here, like -D_DEBUG=1
DDEFS =

# List all default ASM defines here, like -D_DEBUG=1
DADEFS =

# List all default directories to look for include files here
DINCDIR =

# List the default directory to look for the libraries here
DLIBDIR =

# List all default libraries here
DLIBS =

# List all user C define here, like -D_DEBUG=1
UDEFS =

# Define all user ASM defines here
UADEFS =

# List the user directory to look for the libraries here
ULIBDIR =

# List all user libraries here
ULIBS =

#
# End of default and user defines
#######################################################################
```

次のパートでは、プロジェクトをビルドする際にコンパイラ, アセンブラ, リンカがオプションとして使用する追加の定義とロケーション (ヘッダファイルやライブラリモジュールの場所) の設定です。

- **INCDIR**: コンパイラフォルダオプション (例えば、C ヘッダは ”UINCDIR = ./inc”)
- **LIBDIR**: リンカライブラリフォルダオプション
- **DEFS**: コンパイラ定義オプション
- **ADEFS**: アセンブラ定義オプション
- **LIBS**: リンカライブラリオプション

本パートは変更する必要はありません。すべての定義は前述のパートで設定されているためです。

```
INCDIR  = $(patsubst %,-I%,$(DINCDIR) $(UINCDIR))
LIBDIR  = $(patsubst %,-L%,$(DLIBDIR) $(ULIBDIR))
DEFS    = $(DDEFS) $(UDEFS)
ADEFS   = $(DADEFS) $(UADEFS)
LIBS    = $(DLIBS) $(ULIBS)
```

次の行はオブジェクトファイルを確定させます。つまり、src フォルダに格納されているすべての C ファイルとアセンブラ (*.s) ファイルをオブジェクトファイル (*.o) にします。

```
OBJS    = $(SRC:.c=.o) $(ASRC:.s=.o)
```

次の行はコンパイラの最適化レベルを設定します。

```
# Define optimization level here
OPT = -O0
```

以下のインストラクションは、ターゲットの ARM プロセッサ(cortex-m3)を指定します。

コンパイラとアセンブラは、アセンブラコードを生成する際どのインストラクションセットを使用するかを本オプションを使用して決定します。

```
MCU   = cortex-m3
MCFLAGS = -mcpu=$(MCU)
```

GNU コンパイラによって使用されるすべてのオプションは、次のパートで設定します。

```
CPFLAGS                 = $(MCFLAGS)
CPFLAGS                 += $(OPT)
CPFLAGS                 += -gdwarf-2
CPFLAGS                 += -mthumb
CPFLAGS                 += -mapcs-frame
CPFLAGS                 += -msoft-float
CPFLAGS                 += -mno-sched-prolog
CPFLAGS                 += -fno-hosted
CPFLAGS                 += -mtune=cortex-m3
CPFLAGS                 += -mfix-cortex-m3-ldrd
CPFLAGS                 += -ffunction-sections
CPFLAGS                 += -fdata-sections
CPFLAGS                 += -fomit-frame-pointer
CPFLAGS                 += -Wall
CPFLAGS                 += -Wstrict-prototypes
CPFLAGS                 += -fverbose-asm
CPFLAGS                 += -Wa,-ahlms=$(<:.c=.lst)
CPFLAGS                 += $(DEFS)
```

C ソースファイルとインクルードされるヘッダファイルとの間の依存情報を生成するために、コンパイラフラグを使用可能に設定します。生成される情報は ”make clean” でプロジェクトをクリーンすると削除されます。

```
# Generate dependency information
CPFLAGS                 += -MD -MP -MF .dep/$(@F).d
```

以下の行は GNU アセンブラフラグです。

```
ASFLAGS             = $(MCFLAGS)
ASFLAGS             += -g
ASFLAGS             += -gdwarf-2
ASFLAGS             += -mthumb
ASFLAGS             += -amhls=$(<:.s=.lst)
ASFLAGS             += $(ADEFS)
```

次のパートはリンカフラグを決定します。

```
LK                  =  -static -mcpu=cortex-m3 -mthumb -mthumb-interwork
LK                  += -nostartfiles
LK                  += -Wl,--start-group
LK                  += -lc -lg -lstdc++ -lsupc++
LK                  += -lgcc -lm
LK                  += -Wl,--end-group
```

本 makefile は RAM / ROM のデバッグ用のアウトプットファイル (*.elf) を生成するビルドプロセスを管理します。なぜなら、それぞれのデバッグコンフィグレーションのリンカスクリプトファイルは別々に設定しなければならないためです。

次のインストラクションは **RAM** のリンカフラグです。

1. prj フォルダに格納されている RAM スクリプトファイル ”Fujitsu_cortex-M3_ram_V21.ld” を設定し、LDSRIPT_RAM 命令を与える
2. マップファイル (*.map) を生成
3. ライブラリフォルダを与える (設定されている場合)

```
LDFLAGS_RAM = -T$(LDSCRIPT_RAM)
LDFLAGS_RAM += -Wl,-Map=$(PROJECT)_ram.map,--cref,--no-warn-mismatch
LDFLAGS_RAM += $(LIBDIR)
```

次のインストラクションは **ROM** のリンカフラグです。

- prj フォルダに格納されている ROM スクリプトファイル ”Fujitsu_cortex-M3_rom_V10.ld” を設定し、LDSRIPT_ROM 命令を与える
- マップファイル (*.map) を生成
- ライブラリフォルダを与える (設定されている場合)

```
LDFLAGS_ROM = -T$(LDSCRIPT_ROM)
LDFLAGS_ROM += -Wl,-Map=$(PROJECT)_rom.map,--cref,--no-warn-mismatch
LDFLAGS_ROM += $(LIBDIR)
```

次のパートでは、**RAM** ターゲットを生成するためのメイクルールを定義します。RAM ターゲットをビルドすることで、すべてのオブジェクトファイル (*.o) とアウトプットファイル (*.els, *.bin, *.hex, *.mhx) が生成されます。

1. 最初の定義フラグはアセンブラ向けで変数 Debug_RAM に 1 を設定。本変数は startup_mb9bfd1x.s ファイルの if 文で実行され、RAM と ROM の初期化ルーチンを区別する
2. オブジェクトファイル ($ (OBJS)) のビルド開始前にターゲットをクリーンする
3. アウトプットファイル (*.elf) のビルドを開始
4. アウトプットファイル (*.hex) のビルドを開始
5. アウトプットファイル (*.bin) のビルドを開始
6. アウトプットファイル (*.mhx) のビルドを開始

```
RAM: ASFLAGS += --defsym Debug_RAM=1
RAM: clean $(OBJS) $(PROJECT)_ram.elf $(PROJECT)_ram.hex
RAM: $(PROJECT)_ram.bin
RAM: $(PROJECT)_ram.mhx
```

次の行では **ROM** ターゲット定義を記載しています。ROM ターゲットはメイクターゲットのデフォルトに設定されています。”make all” コマンドで ROM ターゲットのビルドプロセスが開始されます。

```
all: ROM
```

変数 Debug_RAM を 0 に設定します。ほかのインストラクションは RAM ターゲットと同様で、アウトプットファイルのみ ROM ベース (*_rom.elf, *_rom.hex など) になります。

```
ROM: ASFLAGS += --defsym Debug_RAM=0
ROM: clean $(OBJS) $(PROJECT)_rom.elf $(PROJECT)_rom.hex
ROM: $(PROJECT)_rom.bin
ROM: $(PROJECT)_rom.mhx
```

ビルドプロセスを開始することで、すべてのソースファイル (*.c, *.s) からオブジェクトファイル (*.o) を生成します。

(*.c) ファイルをコンパイルする際、GNU コンパイラ (CC=arm-none-eabi-gcc.exe) がコールされます。フラグ (CPFLAGS) とヘッダファイルの格納先がコンパイラにわたされます。

```
%o : %c
      @ echo "--compiling--"
      $(CC) -c $(CPFLAGS) -I . $(INCDIR) $< -o $@
```

次の行はアセンブラプロセスです。GNU アセンブラ (AS=arm-none-eabi-as.exe) はオブジェクトファイルを生成します。ASFLAGS は、既に定義した ROM もしくは RAM のビルドコンフィグレーションフラグです。

```
%o : %s
      @ echo "--assembling--"
      $(AS) $(ASFLAGS) $< -o $@
```

リンクプロセスでは、GNU コンパイラ (CC=arm-none-eabi-gcc.exe) がコンパイラとアセンブラによって生成されたすべてのオブジェクトファイル ($(OBJS)=*.o) からアウトプットファイル (*.elf) を生成します。

ROM ターゲットをビルドする際、GNU リンカは ROM リンカスクリプトファイルを特定するためにオプション $(LDFLAGS_ROM) (LDFLAGS_ROM = -T$(LDSCRIPT_ROM)) を使用します。

```
%rom.elf: $(OBJS)
      @ echo "--linking--"
      $(CC) $(OBJS) $(LK) $(LDFLAGS_ROM) $(LIBS) -o $@
```

RAM ターゲットをビルドする際、GNU リンカは RAM リンカスクリプトファイル ”LDSCRIPT_RAM = ./**prj/**MB9BFD18_ram.ld” を特定するためにオプション$(LDFLAGS_RAM) (LDFLAGS_RAM = -T$(LDSCRIPT_RAM)) を使用します。

```
%ram.elf: $(OBJS)
      @ echo "--linking--"
      $(CC) $(OBJS) $(LK) $(LDFLAGS_RAM) $(LIBS) -o $@
```

次のパートは、アウトプットファイル (*.elf) をほかのフォーマット (*.hex, *.bin, *.mhx) に変換します。

GNU ユーティリティ (CP=arm-none-eabi-objcopy.exe) はビルドプロセスでそれぞれのフォーマットを生成するために使用します。

GNU Objcopy ツールは、本 makefile の始めに HEX, BIN, SREC マクロからコールされます。Objcopy オプションはこれらのマクロを設定します。

```
%hex: %elf
      $(HEX) $< $@

%bin: %elf
      $(BIN) $< $@

%mhx: %elf
      $(SREC) $< $@
```

クリーンターゲットは clean ルールで管理されます。make clean コマンドはすべてのオブジェクトファイル (*.o), リストファイル (*.lst), アウトプットファイル (*.elf, *.hex, *.bin, *.mhx) を削除します。clean ルールは、RAM もしくは ROM のビルド時にいつもコールされます。

```
clean:
      -rm -f $(OBJS)
      -rm -f $(PROJECT)_ram.elf
      -rm -f $(PROJECT)_ram.map
      -rm -f $(PROJECT)_ram.hex
      -rm -f $(PROJECT)_ram.bin
      -rm -f $(PROJECT)_ram.mhx
      -rm -f $(PROJECT)_rom.elf
      -rm -f $(PROJECT)_rom.map
      -rm -f $(PROJECT)_rom.hex
      -rm -f $(PROJECT)_rom.bin
      -rm -f $(PROJECT)_rom.mhx
      -rm -f $(SRC:.c=.c.bak)
      -rm -f $(SRC:.c=.lst)
      -rm -f $(ASRC:.s=.s.bak)
      -rm -f $(ASRC:.s=.lst)
```

次のパートは OpenOCD を使用して内蔵 Flash に書き込む際に使用します。本パートは、J-Link GDB Server を使用してアウトプットファイル (*.elf) の書込みやデバッグをする場合は必要ありません。

初めのマクロは OpenOCD の場所を設定します。次のマクロは OpenOCD サーバ (openocd-0.5.0.exe) を設定します。OpenOCD サーバはスクリプトコンフィグレーションが必要です。プロジェクトフォルダ (./) 内のスクリプト (openocd.cfg) が使用されます。

```
# specify the directory where openocd executable and configuration files
reside
OPENOCD_DIR = <HERE YOUR PATH TO OPENOCD>/openocd-0.5.0/src

# specify OpenOCD executable
OPENOCD = $(OPENOCD_DIR)openocd-0.5.0.exe

# specify OpenOCD configuration file (pick the one for your device)
OPENOCD_CFG = -f ./openocd.cfg
```

次のパートは FM3 へプログラムを書き込むための OpenOCD コマンドです。

```
# specify OpenOCD flash programing commandos for FM3
OPENOCD_C += -c init
OPENOCD_C += -c jtag_khz 500
OPENOCD_C += -c reset init
OPENOCD_C += -c verify_ircapture disable
OPENOCD_C += -c halt
OPENOCD_C += -c poll
OPENOCD_C += -c 'FM3 mass_erase 0'
OPENOCD_C += -c 'flash write_image $(PROJECT)_rom.bin 0x0 bin'
OPENOCD_C += -c reset run
OPENOCD_C += -c shutdown
```

最後のパートは、ターゲットへの program (書込み) の実行です。

初めにサーバは割り当てられたコンフィグレーションスクリプト (openocd.cfg) をスタートさせます。スクリプト実行後、サーバは与えられたコマンドを実行します。書込みが完了すると、サーバはシャットダウンし Eclipse のコンソールは “Flash Programming Finished.” というメッセージを表示します。

```
# program the FM3 internal flash memory
program:
	@echo "Flash Programming with OpenOCD..."

	$(OPENOCD) $(OPENOCD_CFG) $(OPENOCD_C)
	@echo "Flash Programming Finished."
```

11 Flash メモリへの書込み

11.1 OpenOCD と Flash 書込み

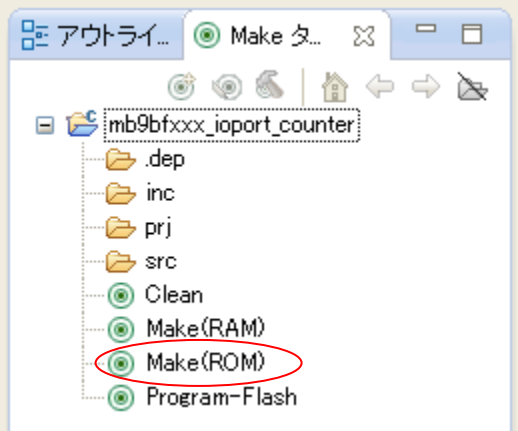
OpenOCD を使用して内蔵 Flash メモリにプログラムを書き込むために、Program-Flash を作成しました (9.5 章を参照)。

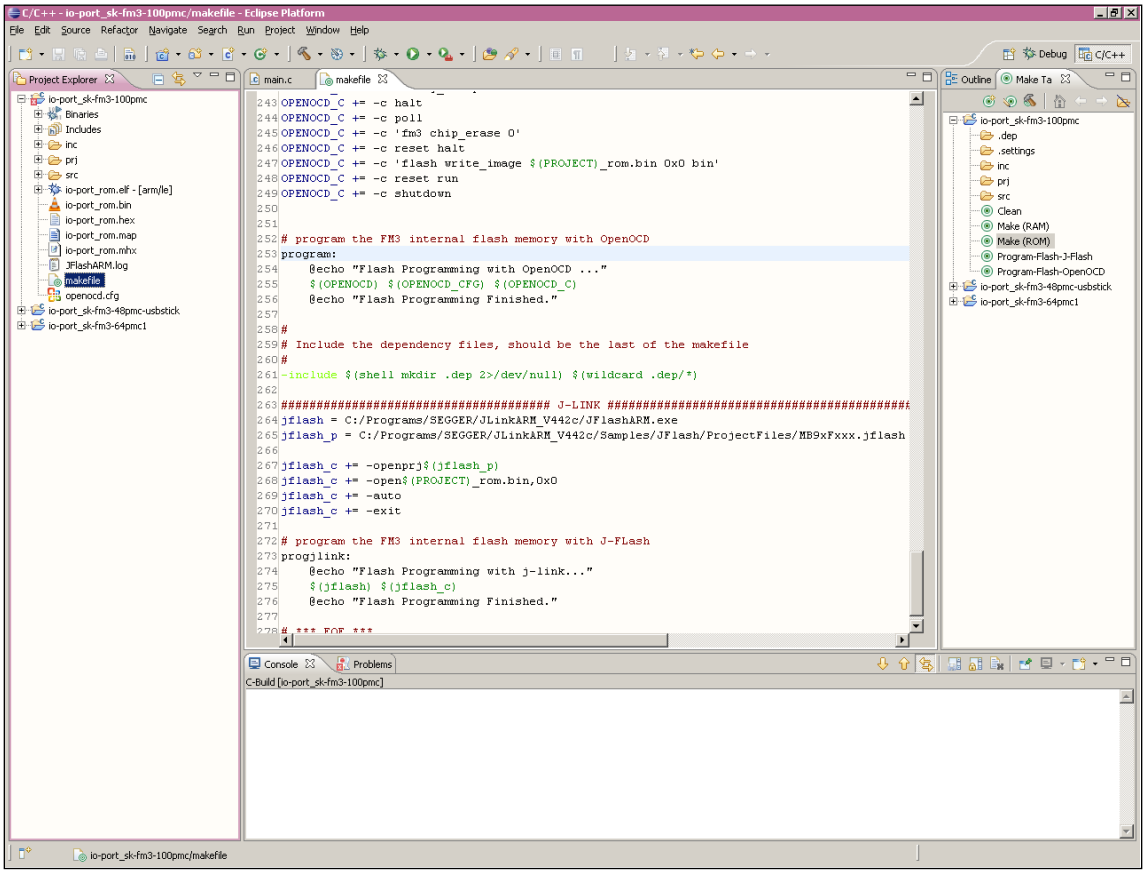
10.3 章では makefile で使用されるすべてのセクションの説明を記載しています。本 makefike で実行される最後のセクションは、内蔵 Flash に書き込むために Eclipse の “C/C++パースペクティブ” で使用されるメイクターゲットの ”Program-Flash” を管理します。

JTAG インタフェースから USB で評価ボード ”SK-FM3-176PMC-ETHERNET V1.1” と PC を接続してください。

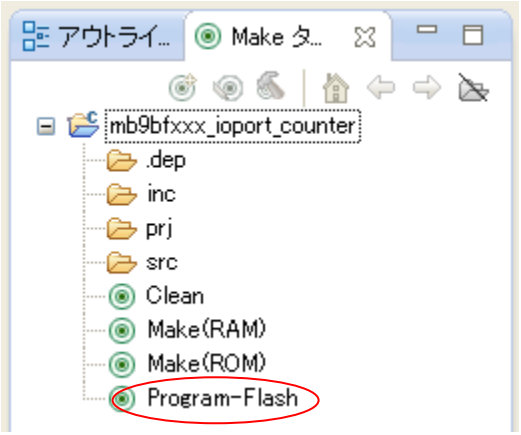
内蔵 Flash へプログラムを書き込むため、初めに Make(ROM) のビルドが必要です。バイナリファイル ”io-port_rom.bin” が生成されます (9.5 章を参照)。

Make(ROM) をクリックしてください。

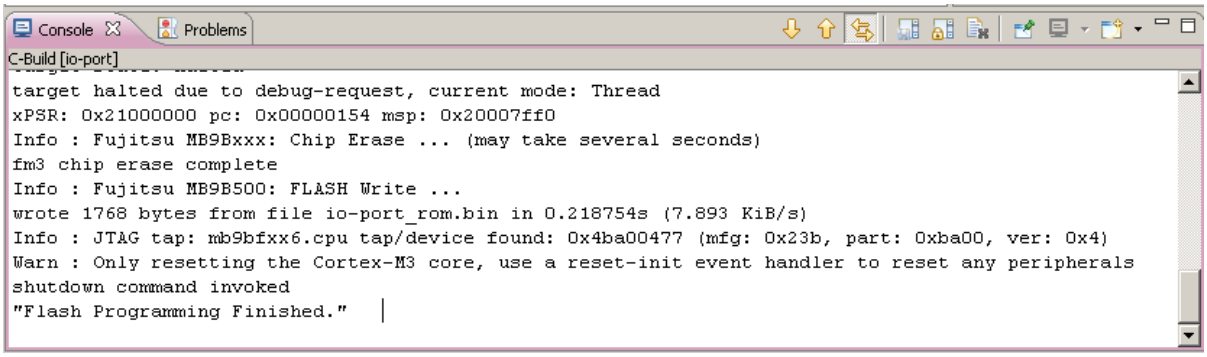




ビルド後、Program-Flash をビルドできるようになります。クリックすることで、OpenOCD から Flash への書込みが開始されます。



以下の図は、OpneOCD を使用した Flash 書込みで Eclipse のコンソールに表示されるメッセージを示しています。



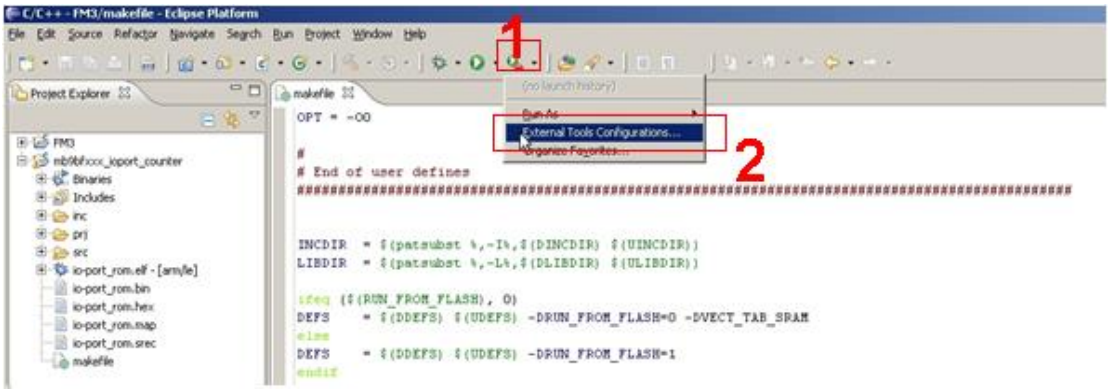
12 Eclipse 外部ツールのセットアップ

12.1 外部ツール

[備考] 本章で記載するコンフィグレーションは、5 章で使用したパスを使用しています。

コンフィグレーションをセットアップする際は、個人のインストールパスを使用してください。

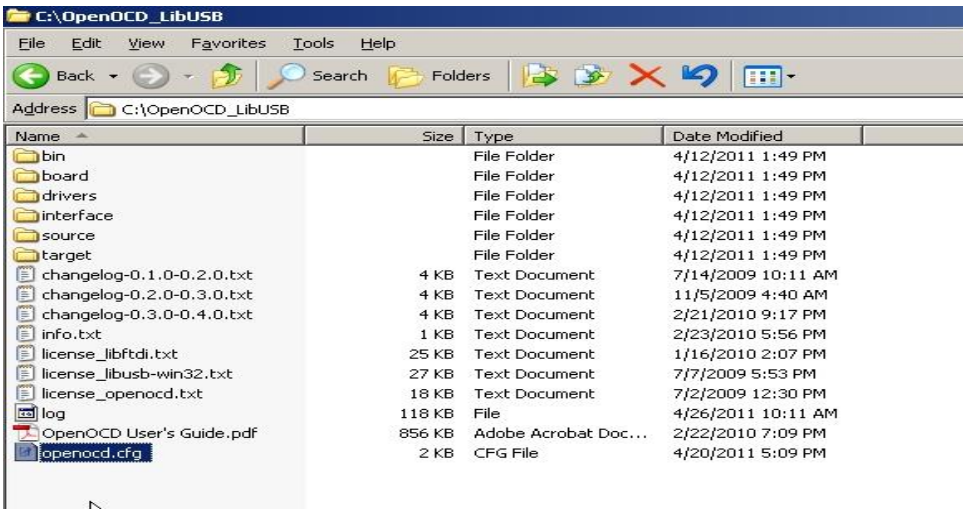
“Run” のプルダウンメニューから “External Tools Configurations…” をクリックすることでツールをインストールできます。



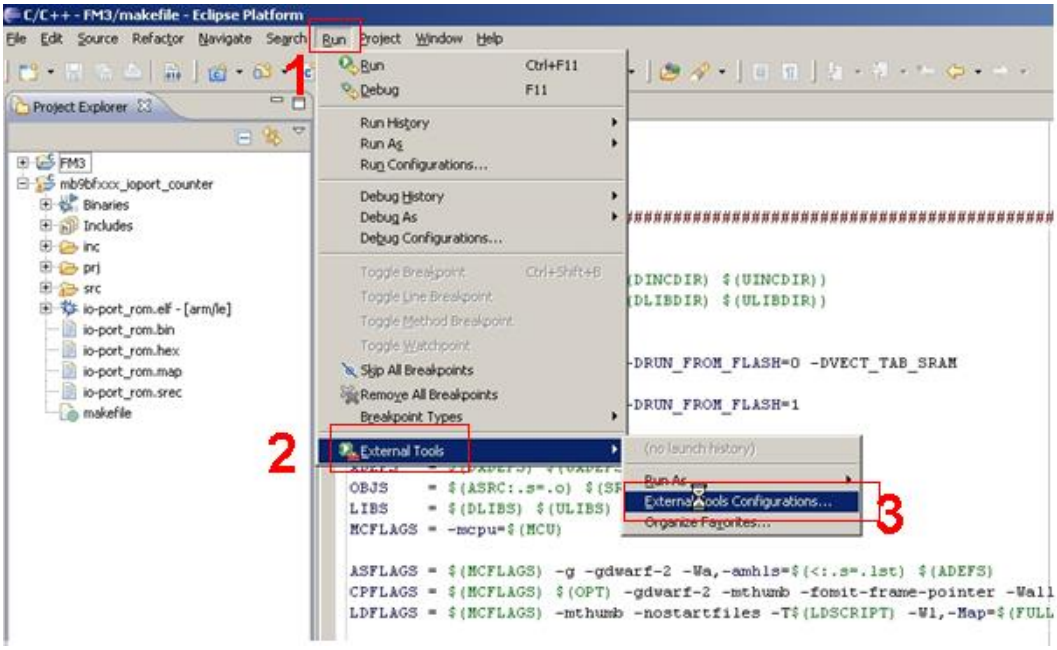
12.2 外部ツールとして OpenOCD を使用

JTAG インタフェースに J-Link を使用する場合、OpenOCD を外部ツールとして設定し J-Link で使用できるように設定します。

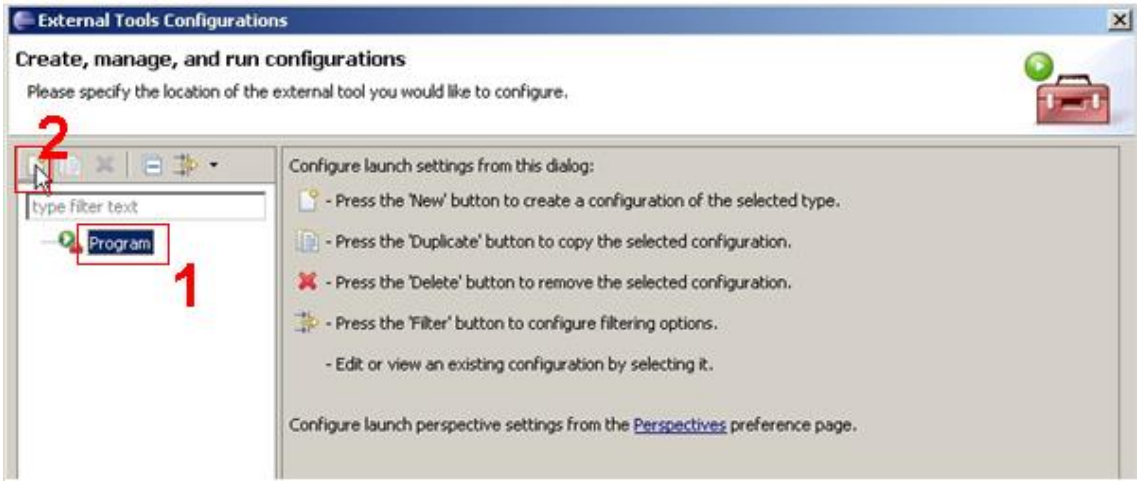
あらかじめ、コンフィグレーションファイルの openocd.cfg を OpenOCD_LibUSB フォルダ (C:¥OpenOCD_LibUSB) にコピーしておいてください。



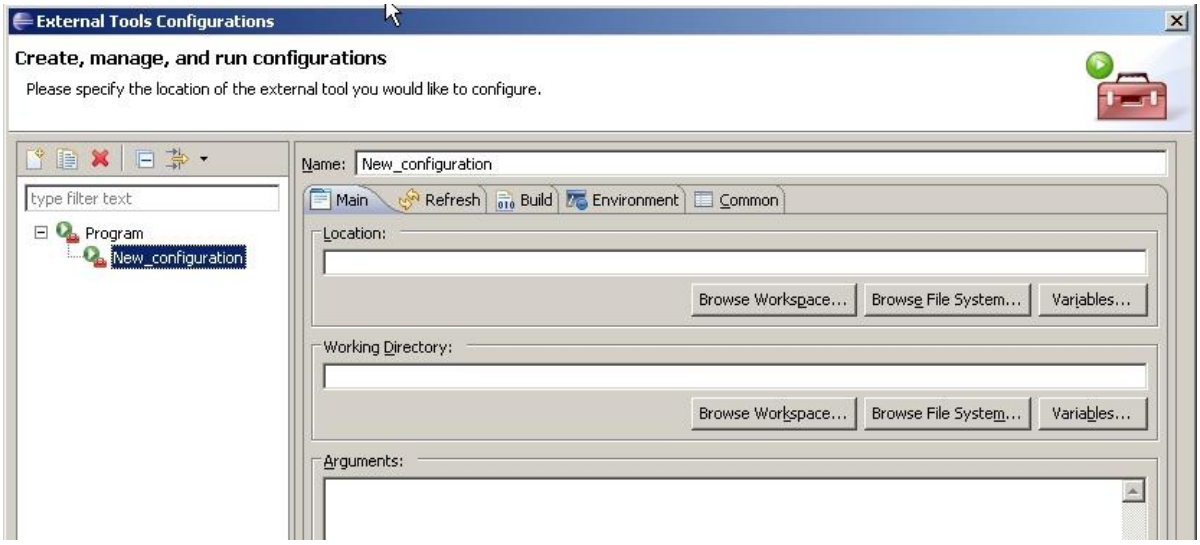
“Run” -> ”External Tools” -> ”External Tools Configurations…” の順にクリックしてください。



“External Tools Configurations” ウィンドウが表示されます。”Program” をクリックし、”New” ボタンから新しい外部ツールを作成します。



“Prigram” をダブルクリックしてください。



フォームに下記の内容を正確に記入していきます。

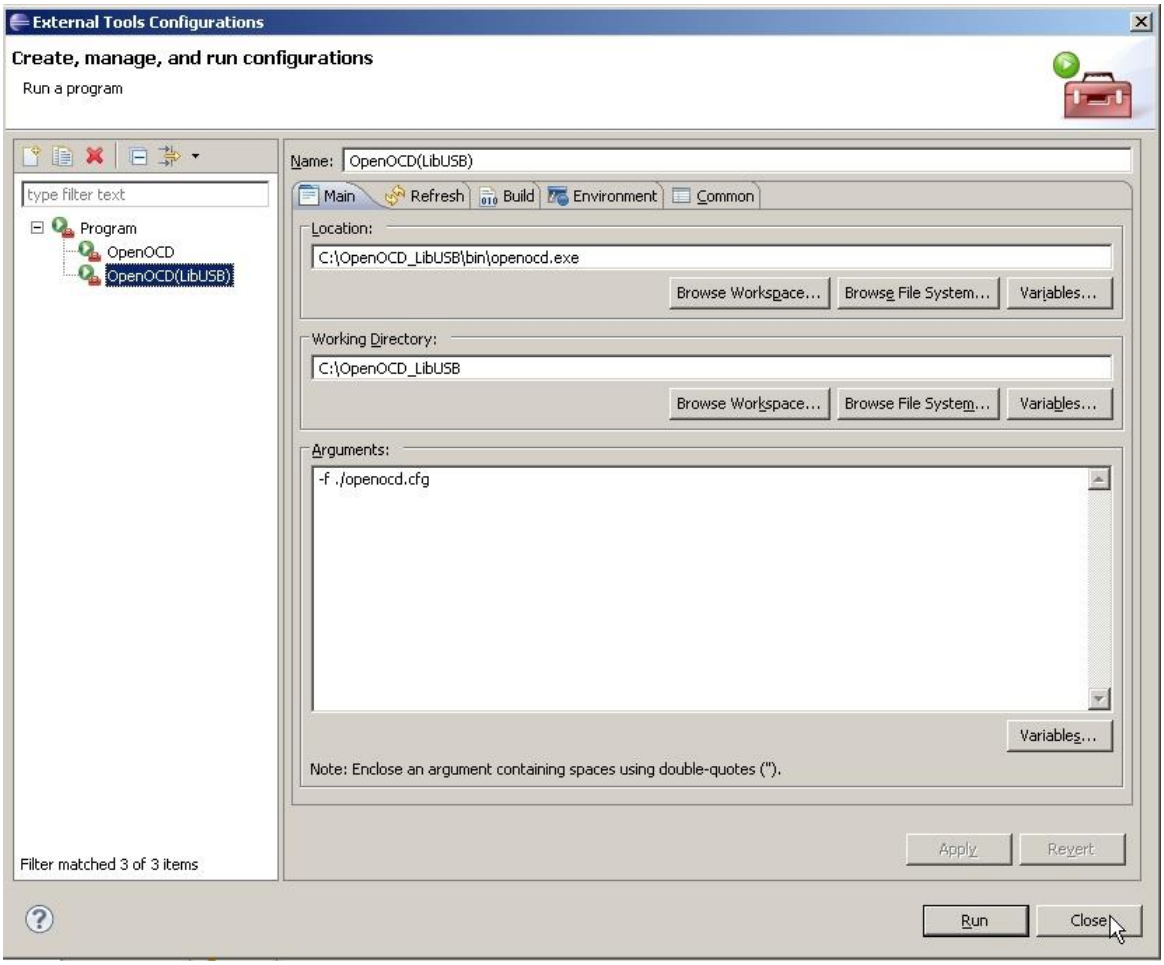
“Name” には OpenOCD と記入してください。

”Location” に、実行ファイルである openocd-0.5.0.exe を下記のとおり指定します。

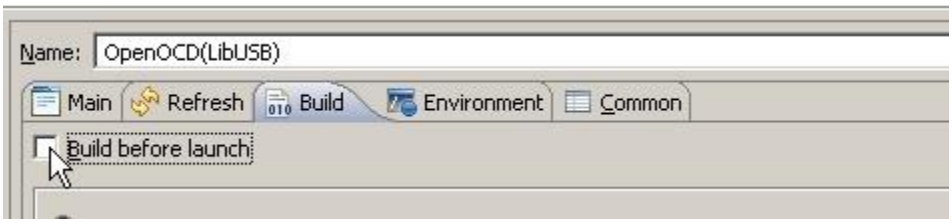
C:¥OpenOCD_LibUSB¥bin¥openocd-0.5.0.exe

“Working Directory” は、”Browse File System…” ボタンをクリックし C:¥OpenOCD_LibUSB を指定します。

Arguments には、引数 ”-f<your project path>¥openocd.cfg” を指定します。



“Build”タブ内の ”Build before launch” のチェックを外してください。

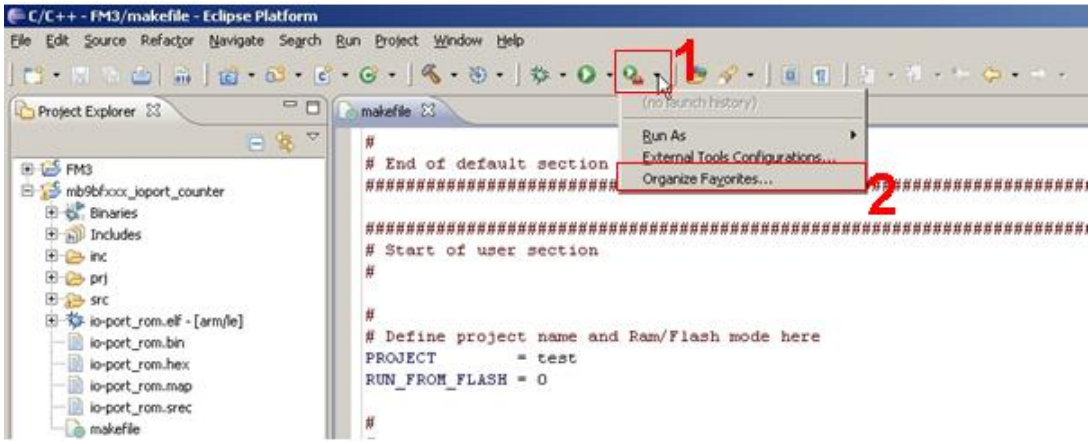


ほかのタブ (Refresh, Environment, Common) は変更する必要はありません。

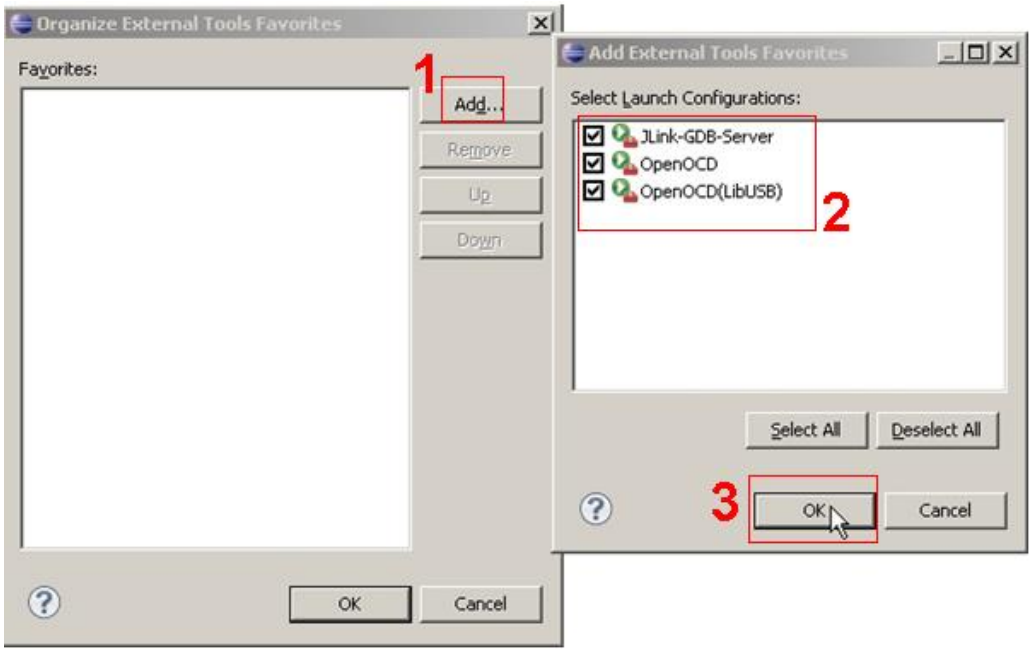
“Apply” と ”Close” をクリックし、OpenOCD の外部ツール設定を閉じます。

前述のセットアップを使用するためには、”Run” -> ”External Tools” -> ”External Tools Configurations…” から ”OpenOCD” を選択します。

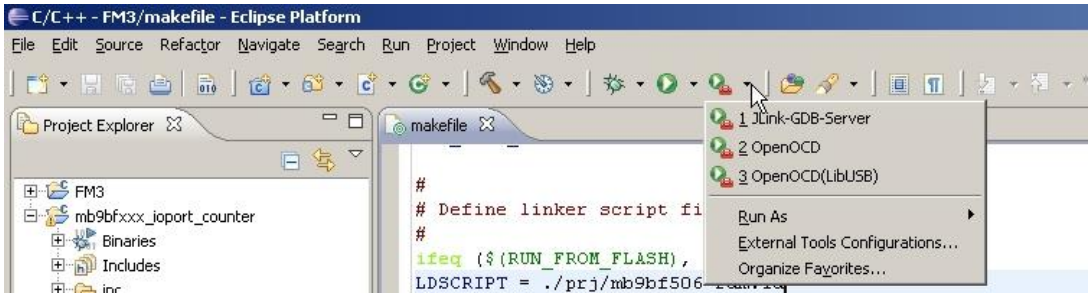
次に、メニューバーのコンフィグレーションウィンドウから、”Organize Favourites” をクリックします。



“Add” をクリックし、作成したコンフィグレーションを選択します。



“OK” をクリックして、コンフィグレーションを保存します。これでお気に入りの外部ツールを加えることができます。外部ツールはメニューバーから選択することで使用できます。

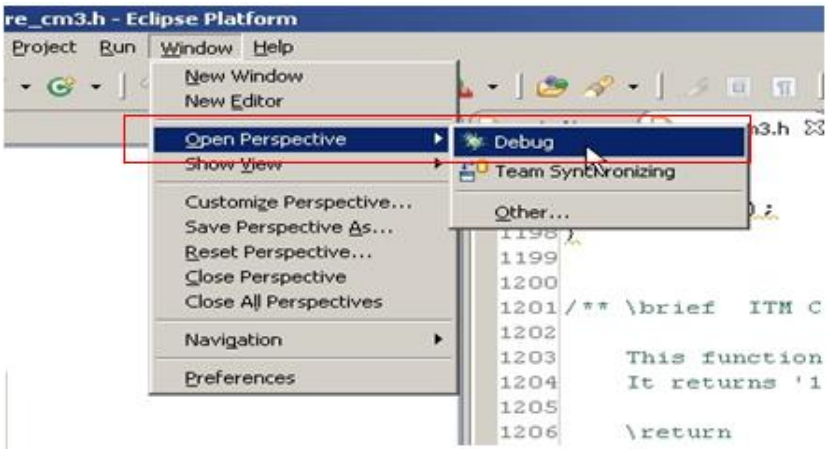


13 Eclipse CDT のデバッグパースペクティブ

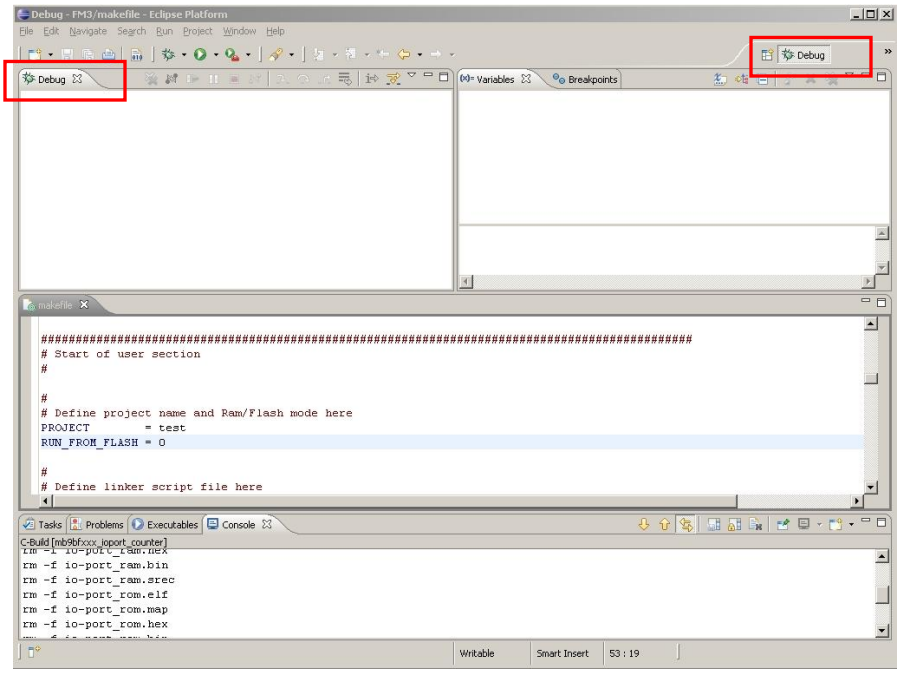
9 章でサンプルの FM3 プロジェクトを作成し、ビルドから作成したアウトプットファイル (*.bin, *.mhx, *.hex) を Flash に書き込むまでを説明しました。これらのアウトプットファイルには、Flash もしくは RAM でデバッグするために必要なデバッグ情報ファイル (*.elf) が含まれています。

デバッグを開始するためには、初めに Eclipse CDT の ”C/C++ Perspective” から ”Debug Perspective” に変更します。

メニューの ”Windows” から ”Open perspective” を選択し、”Debug” をクリックします。”Debug” は、”Other…” から探すこともできます。



クリック後、以下のウィンドウが表示されます。



13.1 OpenOCD を使用したプログラムの書込みとデバッグ

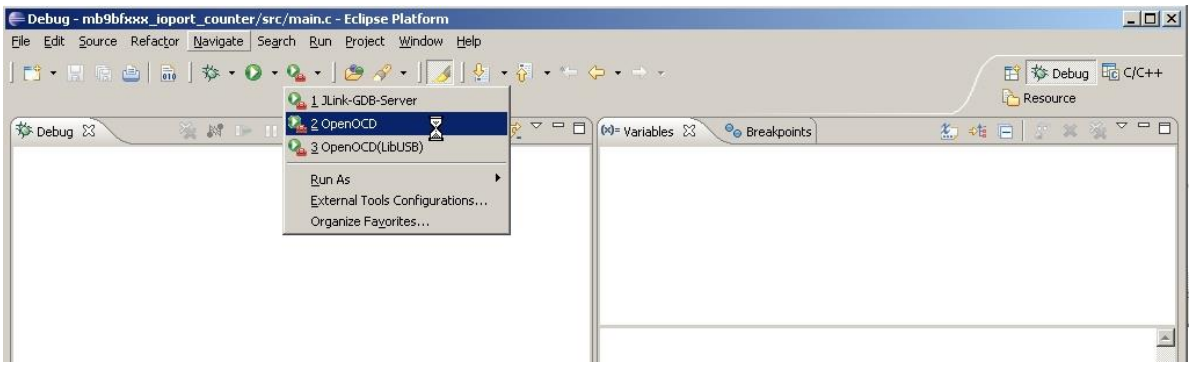
JTAG インタフェースから SK-FM3-176PMC-ETHERNET ボードを PC と USB 接続してください。J-Link と ARM-USB-TINY を接続した例をそれぞれ以下に示します。



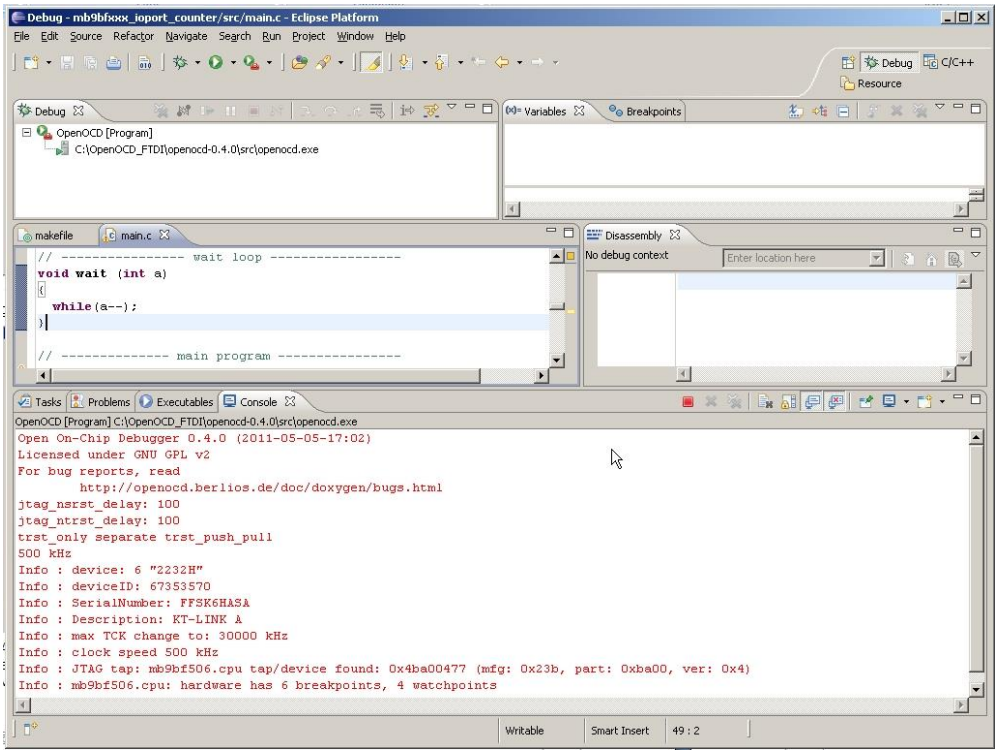
ICE に J-Link, ARM-USB-TINY のどちらを使用しても、下記で説明する操作は共通です。

OpenOCD を起動します。OpenOCD はデーモンとして実行します。これは、バックグラウンドでコマンドを待ち続けるような動作をすることを意味します。

以下のとおり、”OpenOCD” をクリックして外部ツールを起動します。



下のコンソールから、サーバが開始されたことを確認してください。

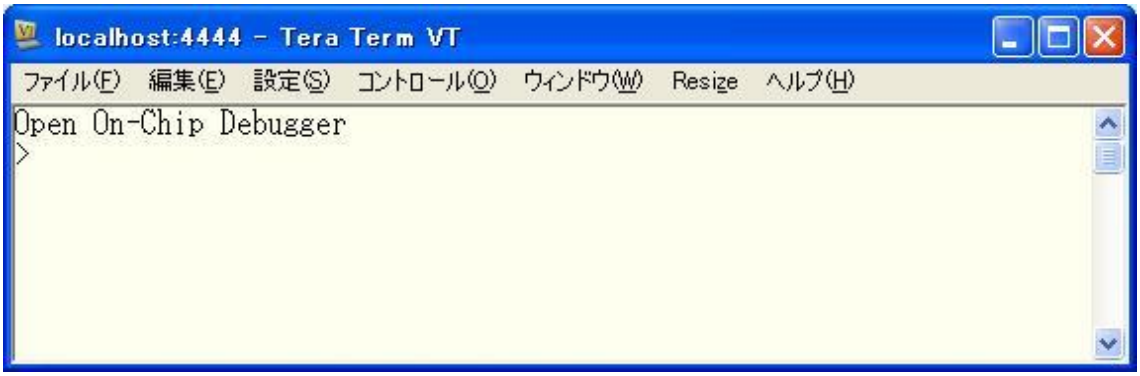


次に、マイコンを halt 状態にします。run 状態では GDB サーバと OpenOCD の間でエラーが発生するためです。

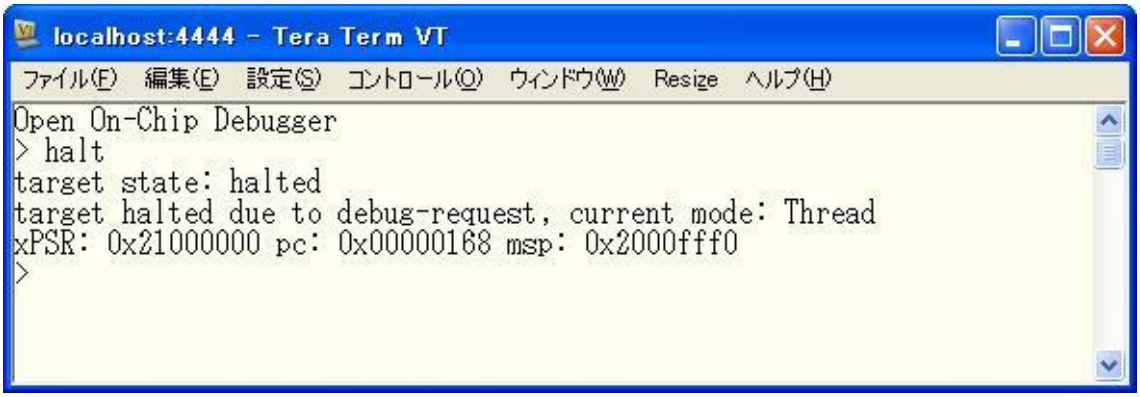
ターミナルエミュレータ (本書では Tera Term を使用) で以下のとおり接続してください。



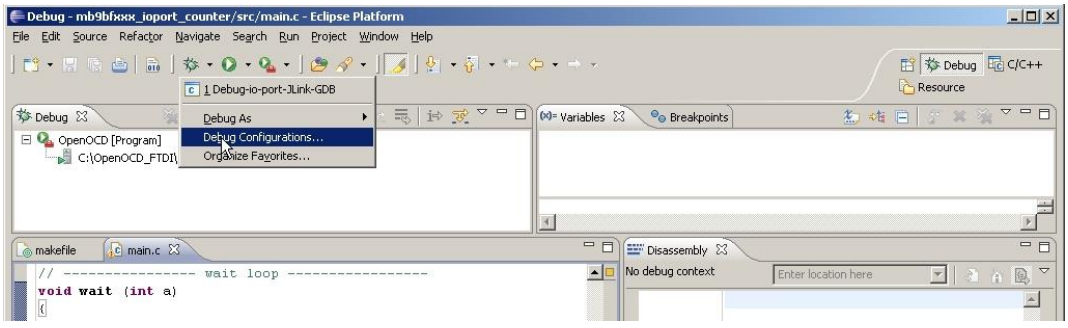
“Open On-Chip Debugger” と表示されたら接続が成功です。



“halt” コマンドを発行し、ターゲットが halt 状態になることを確認します。



続いて、新しく ”Debug Configuration” を作成します。以下のとおり、”Debug Configurations…” をクリックします。



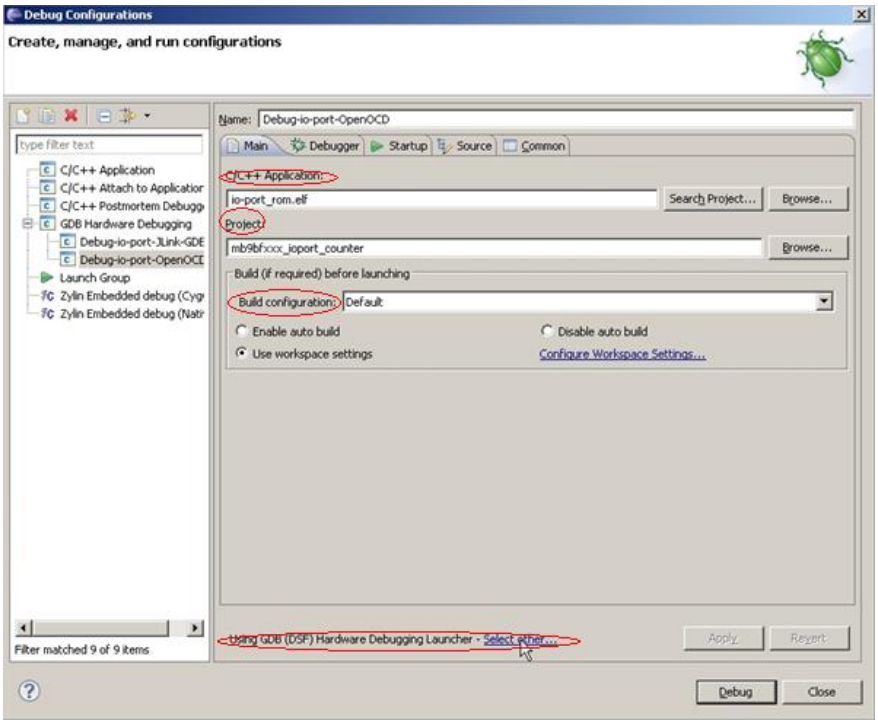
“GDB Hardware Debugging” を選択し、“New” をクリックしてください。

まずデバッグコンフィグレーションをリネームします。ほかのデバッグコンフィグレーションとの混同を避けるため、プロジェクト名 (io-port) に外部ツール名 (OpenOCD) をつけた名前を指定することが推奨されています。

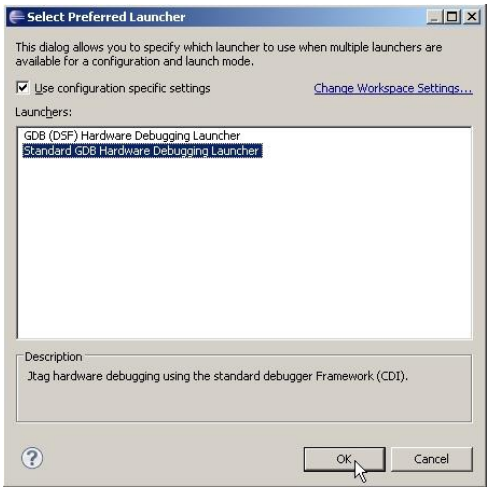
“Project” には、”Browse” ボタンを使って ”mb9bfxxx_ioport_counter” を探し、入力します。

“C/C++ Application” には、”Search Project…” ボタンを使いアプリケーションデバッガファイルである ”io-port-rom.elf” を指定します。

“Build configuration” には ”Use Active” を設定してください。

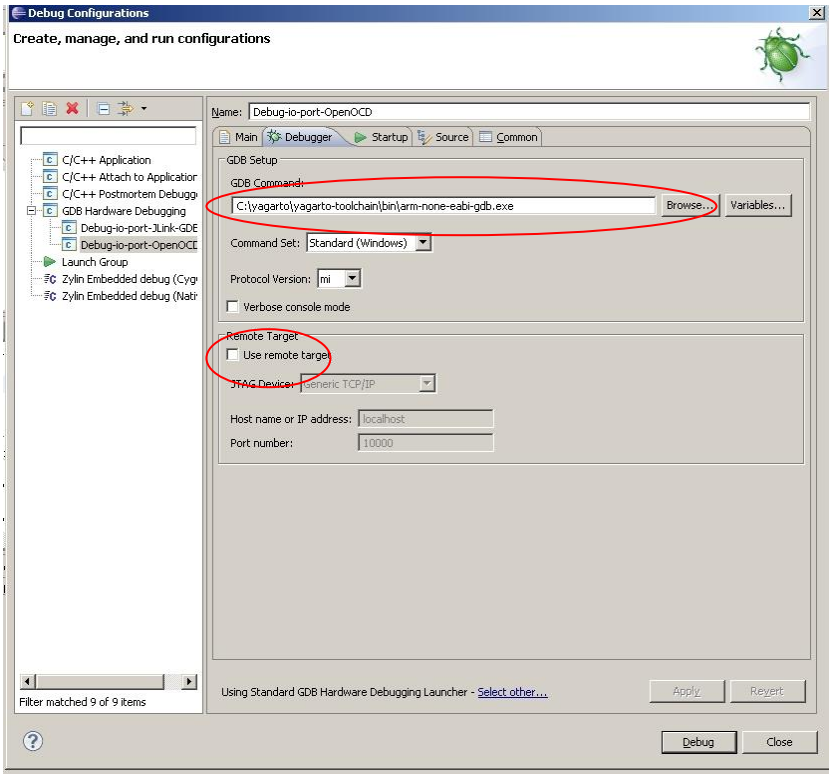


“Select other…” をクリックし、”GDB (DSF) Hardware Debugging launcher” から ”Standard GDB Hardware Debugging launcher” に切り替えます。切替え後、”OK” をクリックしてください。



次に、”Debugger” タブを選択します。”GDB Command” には、”Browse” ボタンを使って GDB デバッガである ”arm-none-eabi-gdb.exe” ファイルを、yagarto のインストールフォルダから探し指定します。

“Use remote target” のチェックを外します。



次に ”Startup” タブを選択します。

“Initialization Commands” に下記をコピーします。

```
# connect to the OpenOCD gdb server
target remote localhost:3333

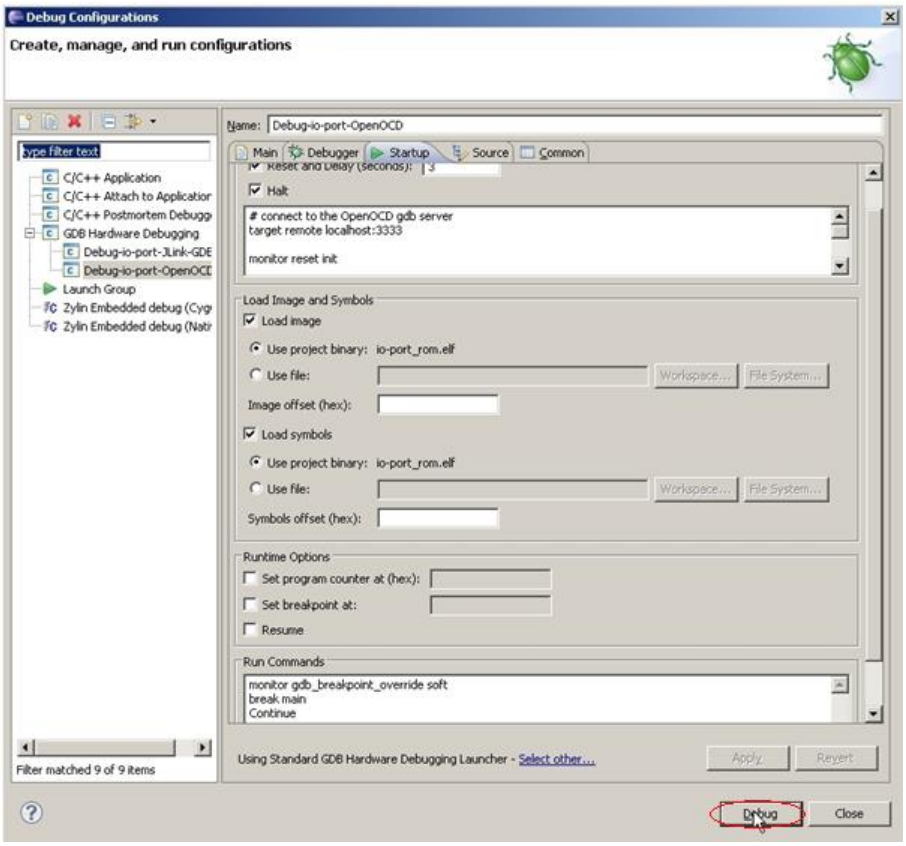
monitor reset init

monitor soft_reset_halt

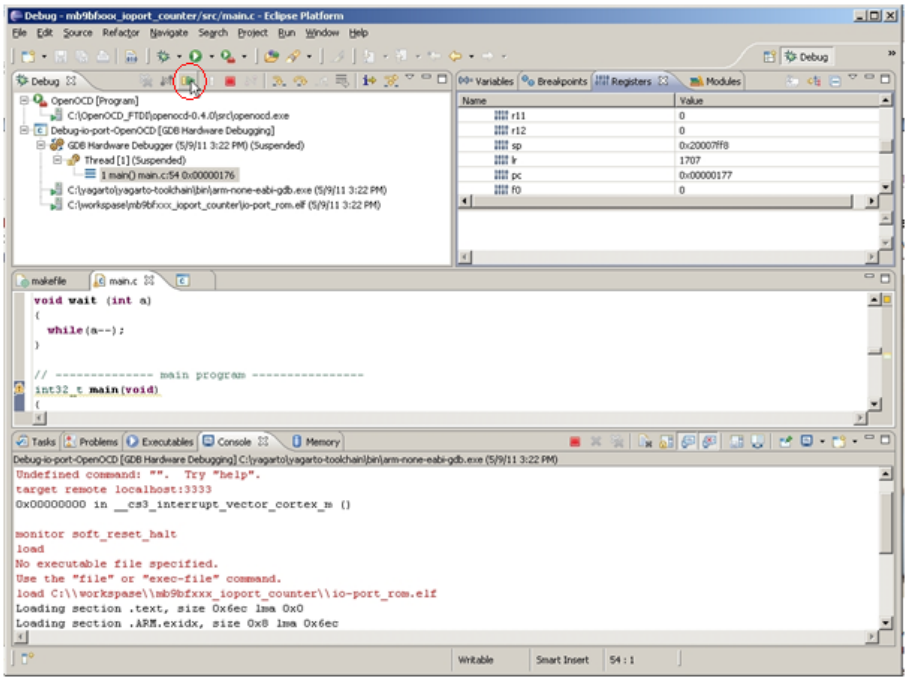
load
```

“Run Commands”に下記をコピーします。

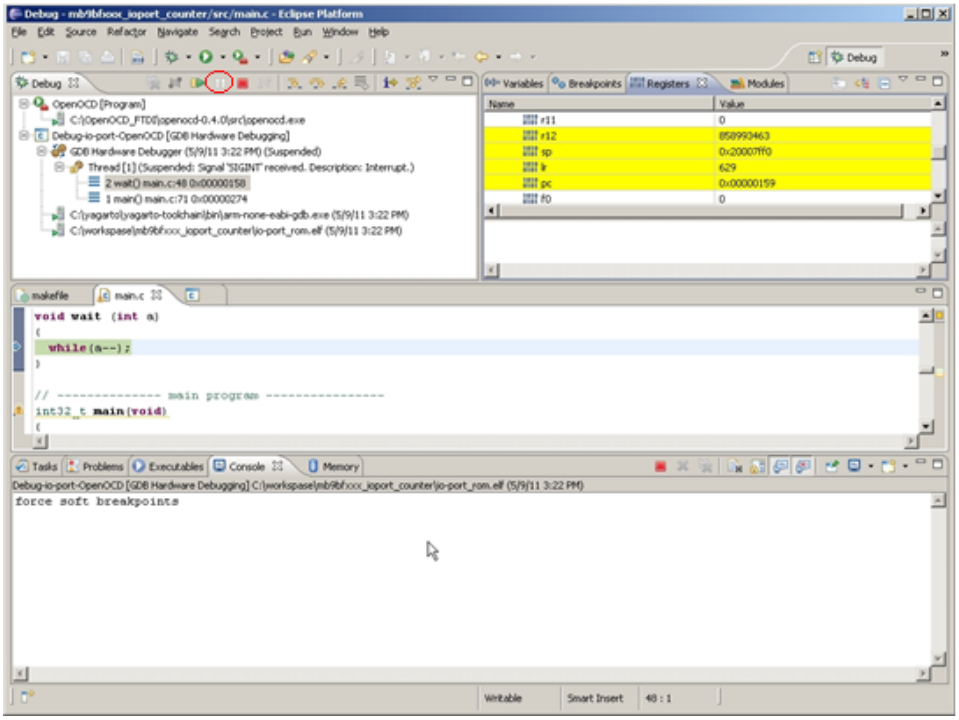
```
monitor gdb_breakpoint_override soft
break main
Continue
```



ほかはデフォルトのまま設定してください。”Debug” をクリックし、デバッグを開始します。



デバッグが成功すると、以下の図のとおりになります。再開には、”Resume” ボタンをクリックしてください。



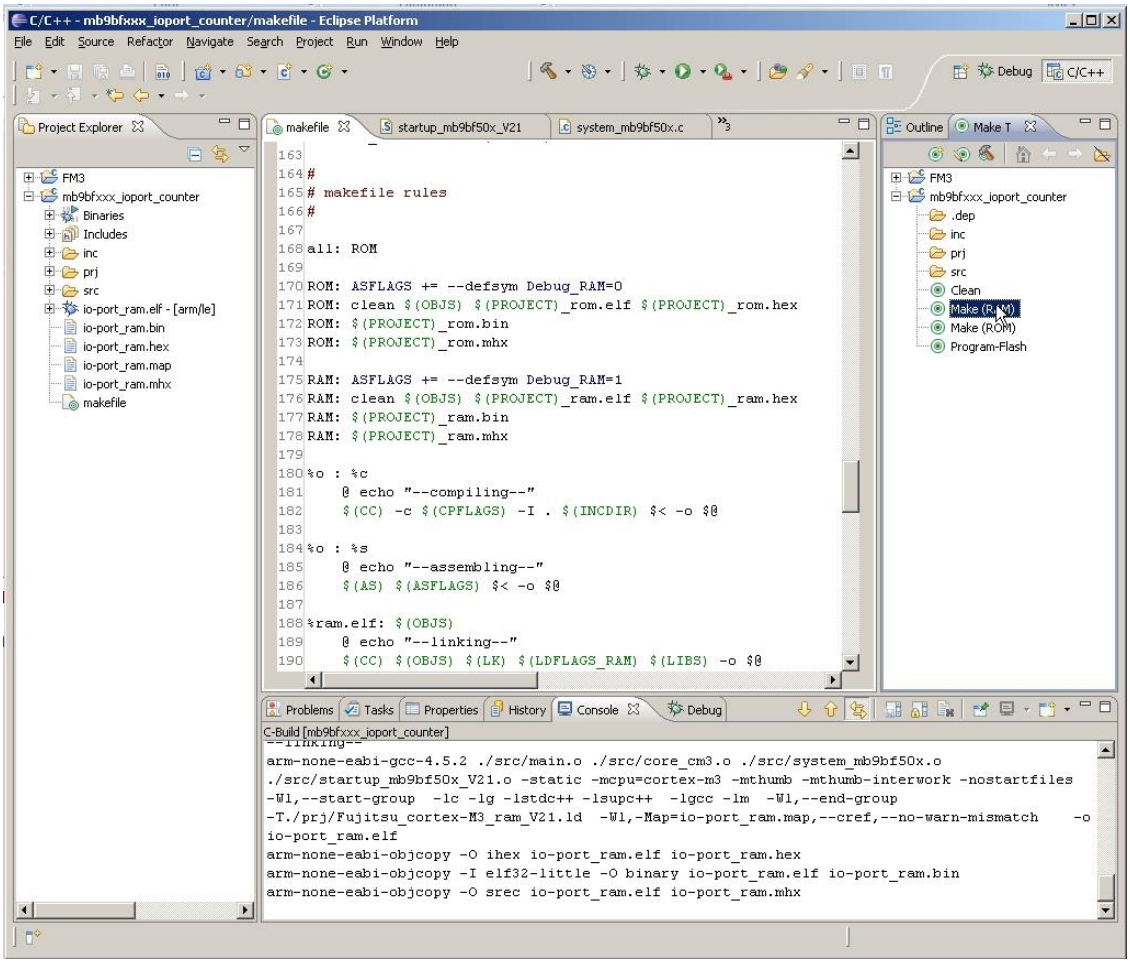
デバッグ開始後、”Suspend” ボタンをクリックすることでいつでも終了できます。

13.2 RAM でのデバッグ

Flash でのデバッグ方法を 13.1 章で説明しました。ほかにも、デバイスの RAM へリンク・ダウンロードしてデバッグすることも可能です。これには初めに RAM アプリケーションを生成する必要があります。”C/C++ Perspective” に戻ってください。



“C/C++” をクリックすると、IDE は C/C++開発パースペクティブに移ります。”Make (RAM)” をクリックすることで、RAM メイクターゲットをビルドし RAM アプリケーションが生成されます (コードとデータが RAM のメモリサイズを超えないよう注意してください)。

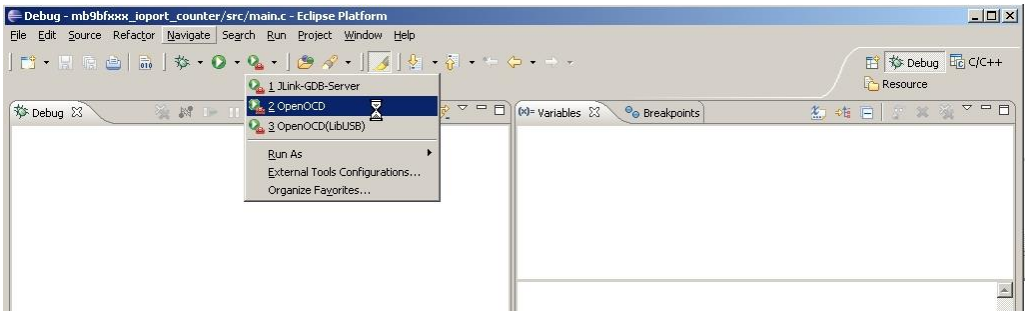


“Debug perspective” に戻り、RAM デバッグを開始します。

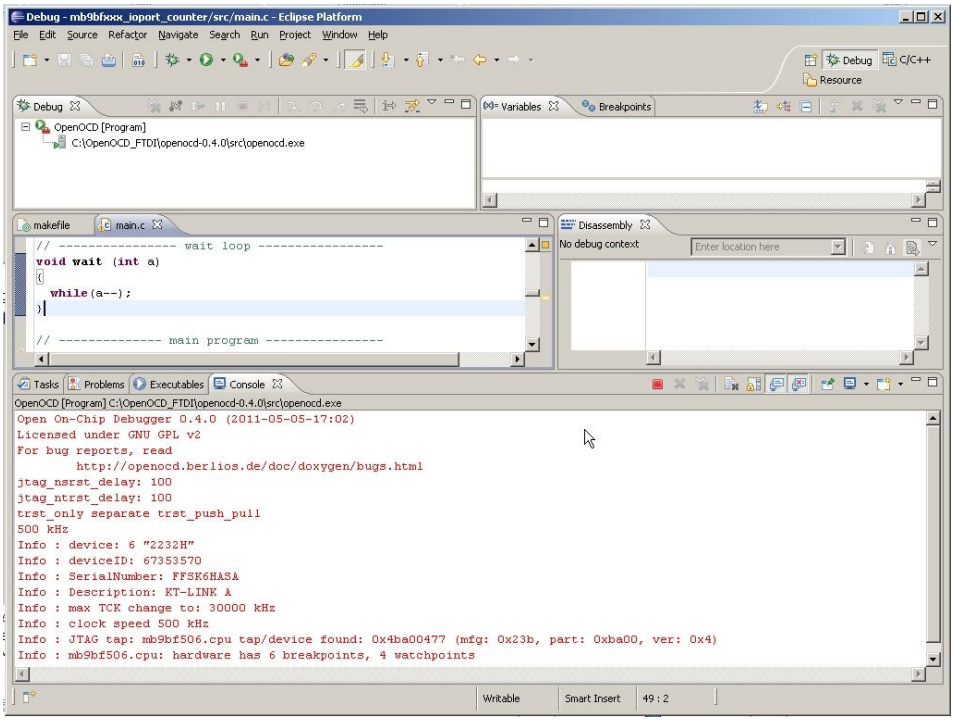


PC と SK-FM3-176PMC-ETHERNET ボードを再接続します。

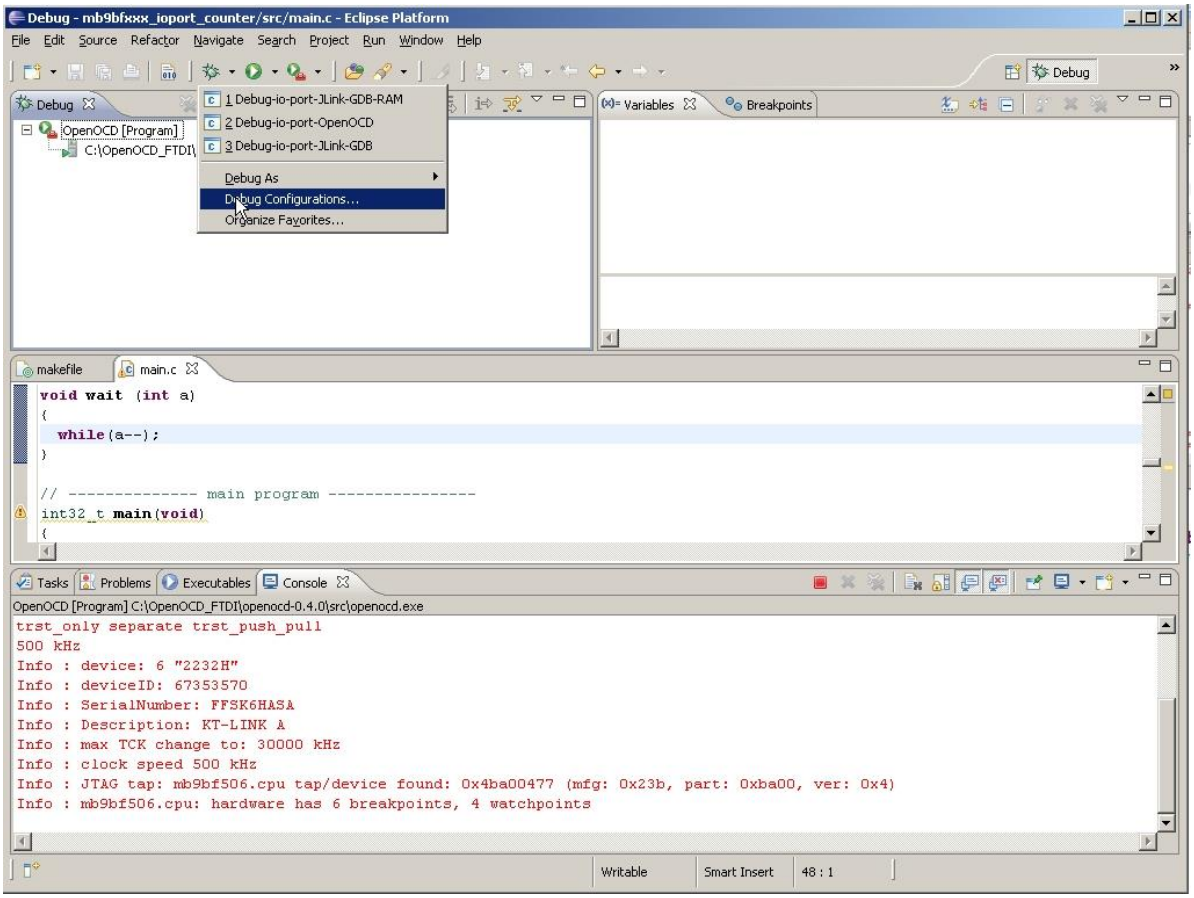
再接続後、OpenOCD を起動します。以下のとおり、”OpenOCD” をクリックして外部ツールを起動します。



下のコンソールから、サーバが開始されたことを確認してください。



新しく ”Debug Configuration” を作成します。以下のとおり、”Debug Configurations…” をクリックします。



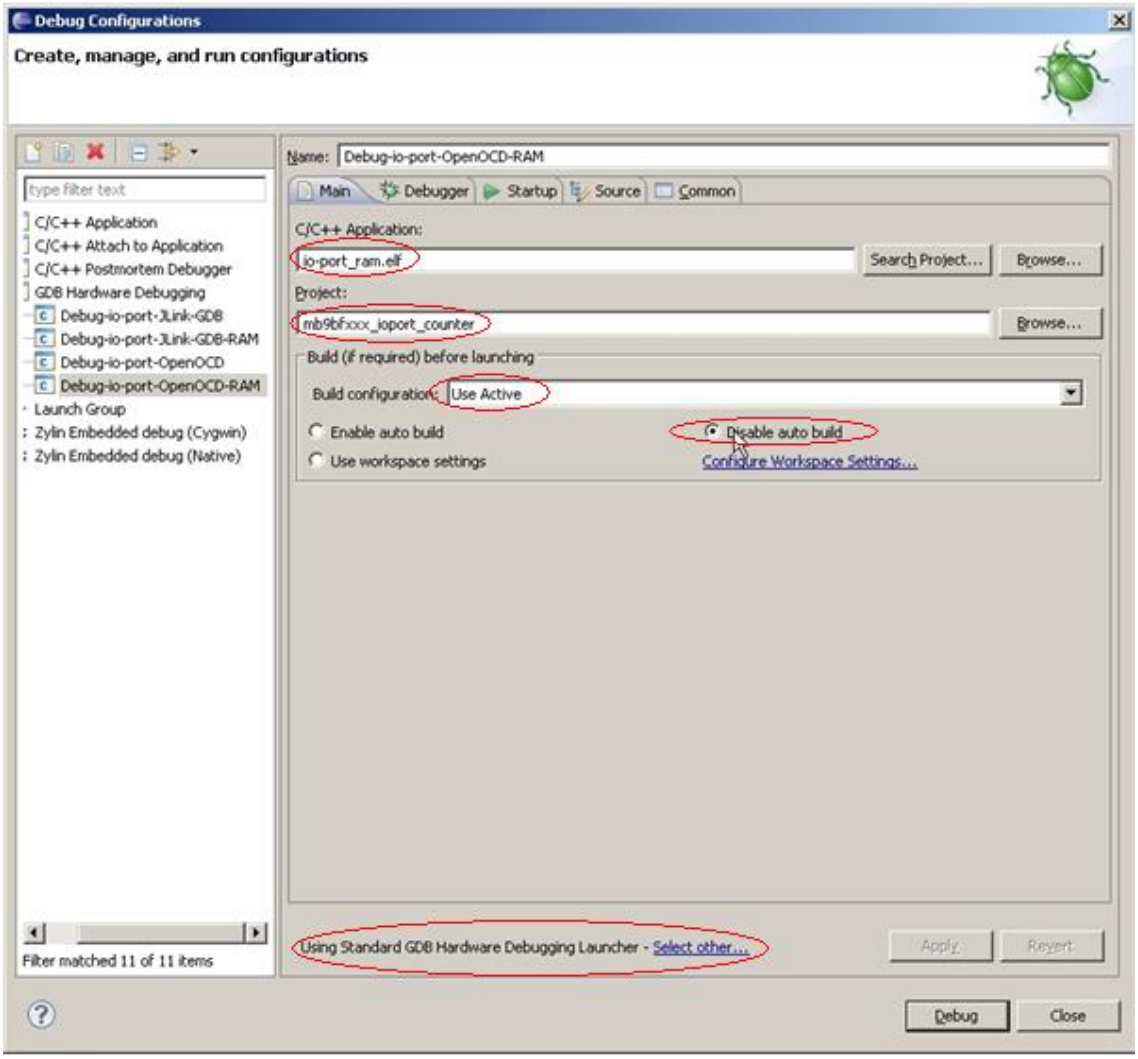
“GDB Hardware Debugging” を選択し、“New” をクリックします。

デバッグコンフィグレーションをリネームします。RAM デバッグと Flash デバッグを分けるため、サフィックスに ”_RAM” をつけて既に保存されているほかのデバッグコンフィグレーションとの混同を避けます。

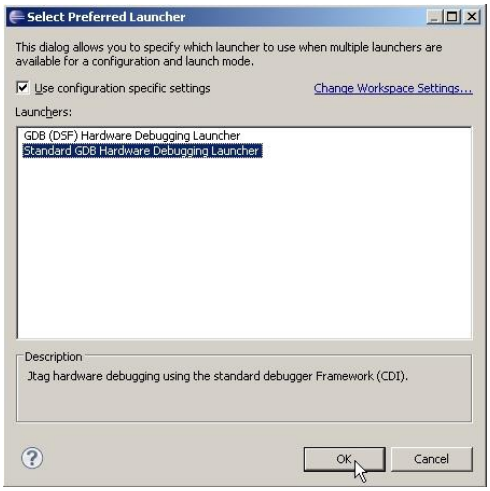
“Project” には、”Browse” ボタンを使って ”mb9bfxxx_ioport_counter” を探します。

“C/C++ Application” には、”Search Project…” ボタンを使ってアプリケーションデバッガファイルである ”io-port-ram.elf” を指定します。

“Build configuration” には ”Use Active” と、”Disable auto build” を設定してください。

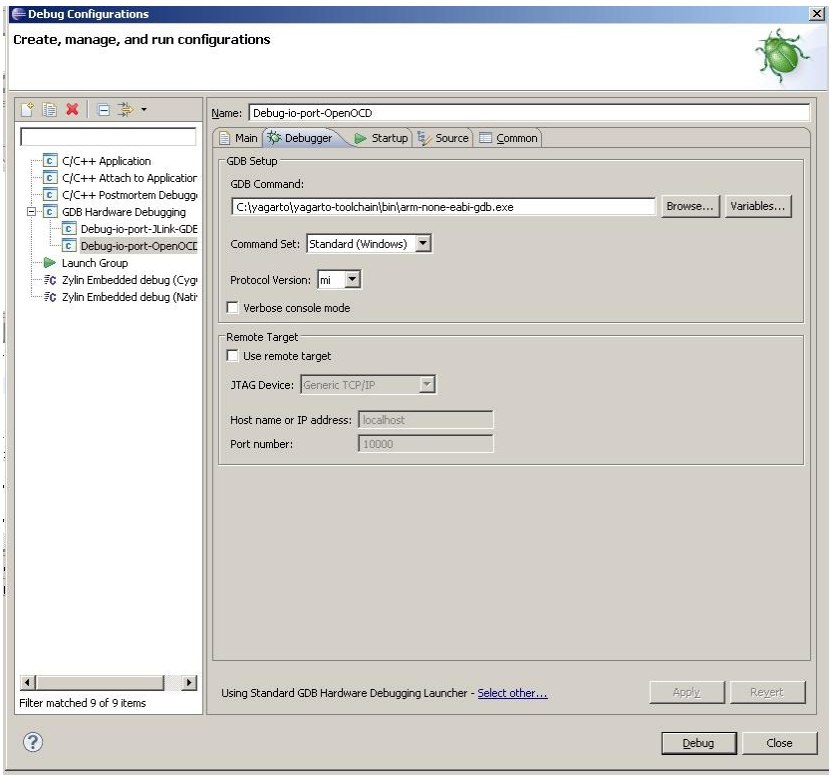


“Select other…” をクリックし、”GDB (DSF) Hardware Debugging launcher” から ”Standard GDB Hardware Debugging launcher” に切り替えます。切替え後、”OK” をクリックしてください。



次に、”Debugger” タブを選択します。”GDB Command” には、”Browse” ボタンを使って GDB デバッガである ”arm-none-eabi-gdb.exe” ファイルを、yagarto のインストールフォルダから探し指定します。

“Use remote target” のチェックを外します。



次に ”Startup” タブを選択します。

“Initialization Commands” に下記をコピーします。

```
# connect to the OpenOCD gdb server
target remote localhost:3333

monitor reset init
monitor reset halt
monitor soft_reset_halt

# Vector table placed in RAM
monitor mww 0xE000ED08 0x1fff0000

load
```

RAM のスタートアドレスを設定

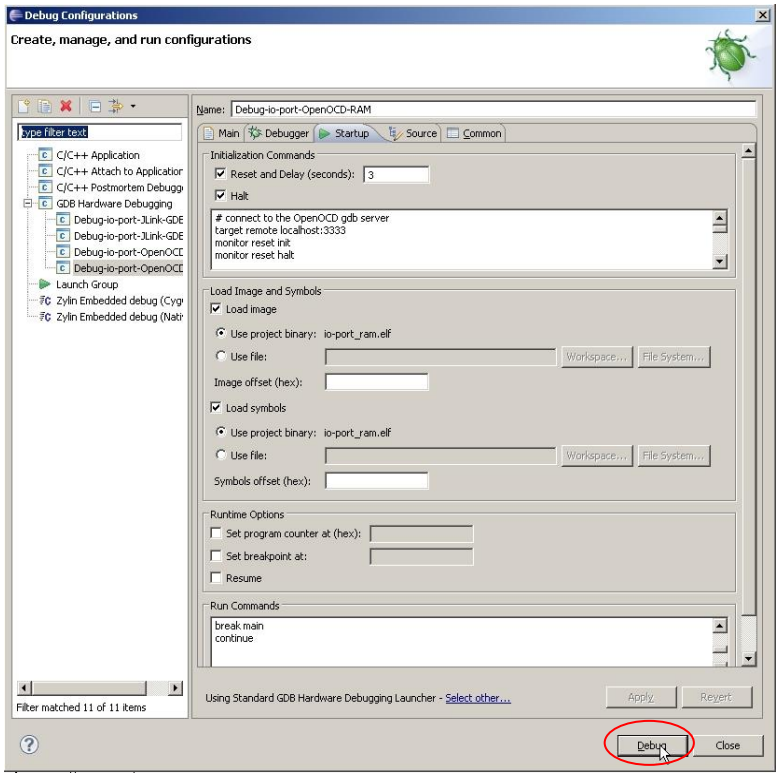
“Run Commands” に下記をコピーします。

```
break main
set $r13 = *(int*)0x1fffE000
set $pc = *(int*)0x1fff0004
continue
```

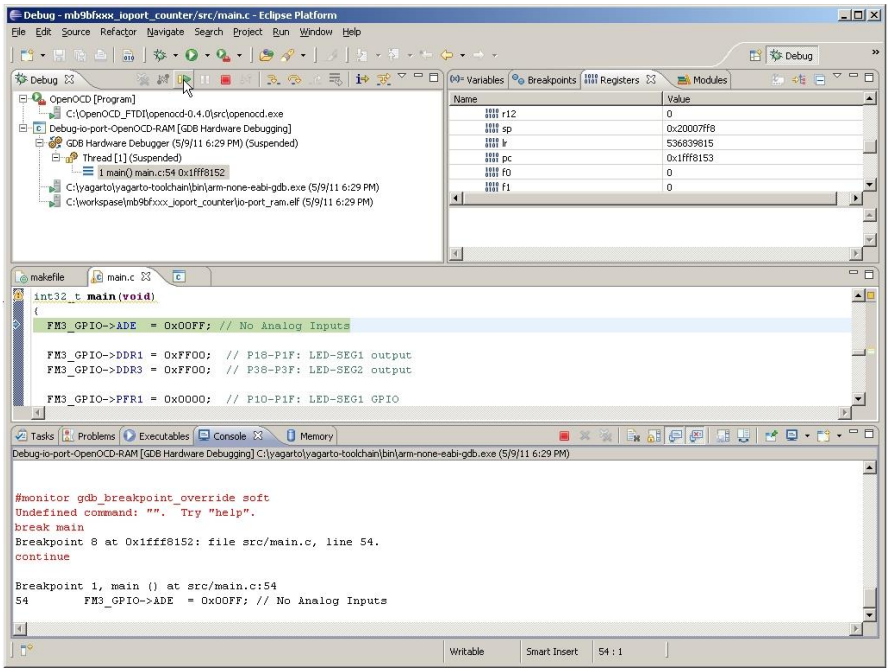
スタックポインタのアドレス

RAM のスタートアドレス +4 を設定

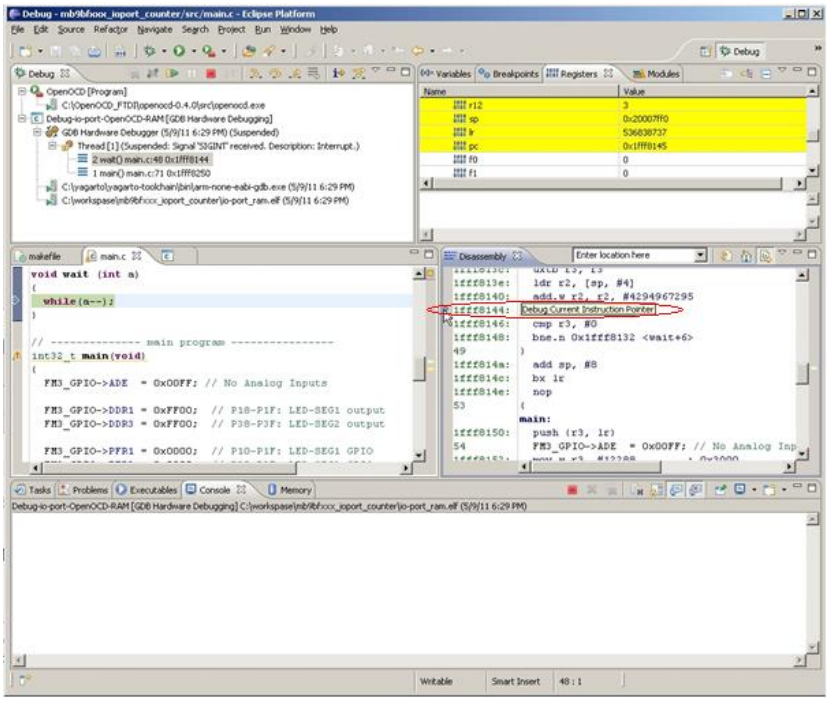
ほかはデフォルトのまま設定してください。”Debug” をクリックし、デバッグを開始します。



RAM デバッグが成功すると、以下の図のとおりになります。再開には、”Resume” ボタンをクリックしてください。



“Disassembly” から、現在のインストラクションを確認できます。”Window” メニューから ”Show View” を選択してください。

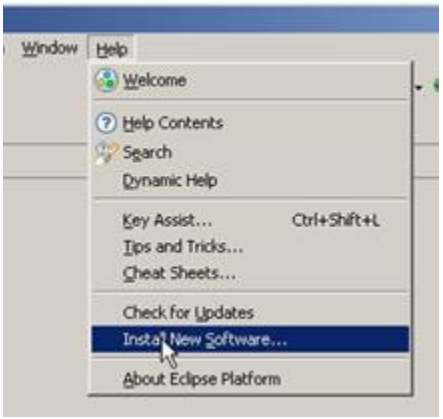


14 Eclipse Embedded Systems Register View プラグイン

Eclipse の ”EmbSysRegView” プラグインは FM3 のすべての資源のペリフェラルレジスタを表示し編集できます。

14.1 プラグインのインストール

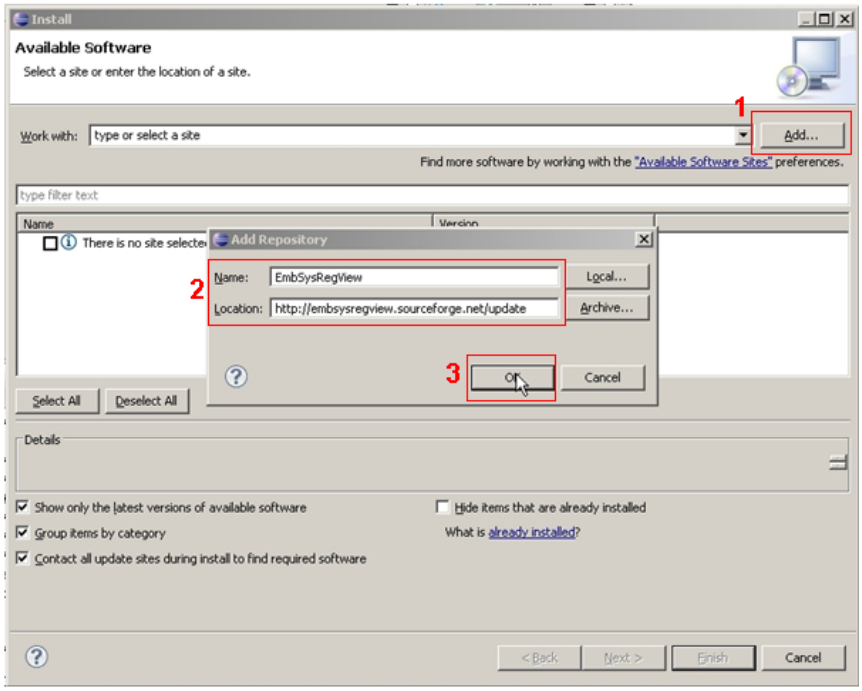
”EmbSysRegView” プラグインを Eclipse にインストールするために、メニューの ”Help” から ”Install New Software…” を選択します。



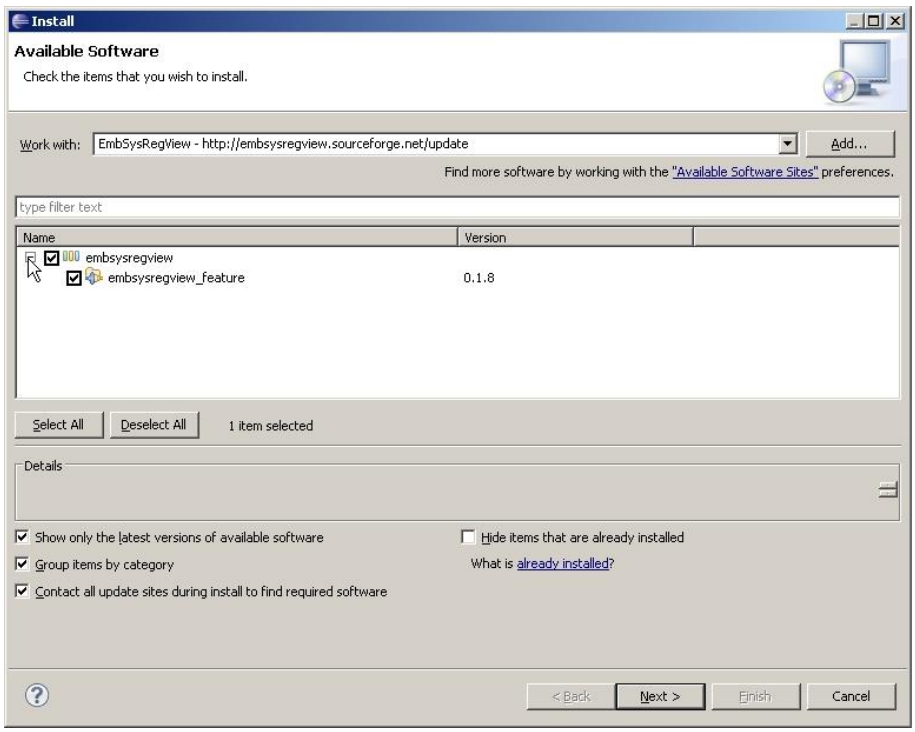
“Add” ボタンをクリックし、”Name” にはプラグインの名前を入力 (例えば EmbSysRegView) し、”Location” には下記の URL を指定してください。

http://embsysregview.sourceforge.net/update

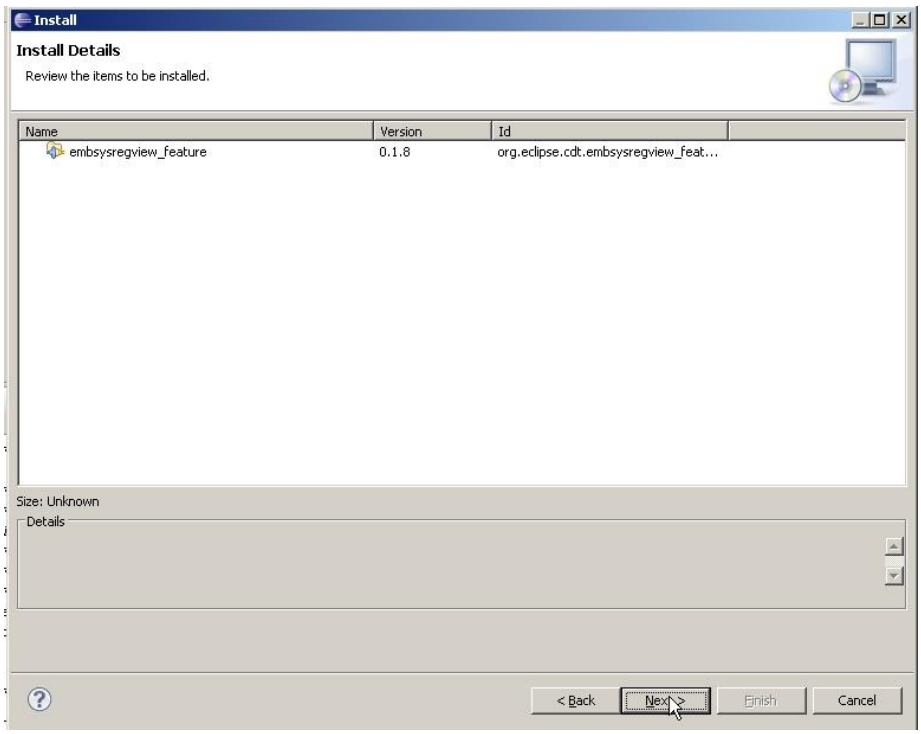
確認後、”OK” をクリックします。



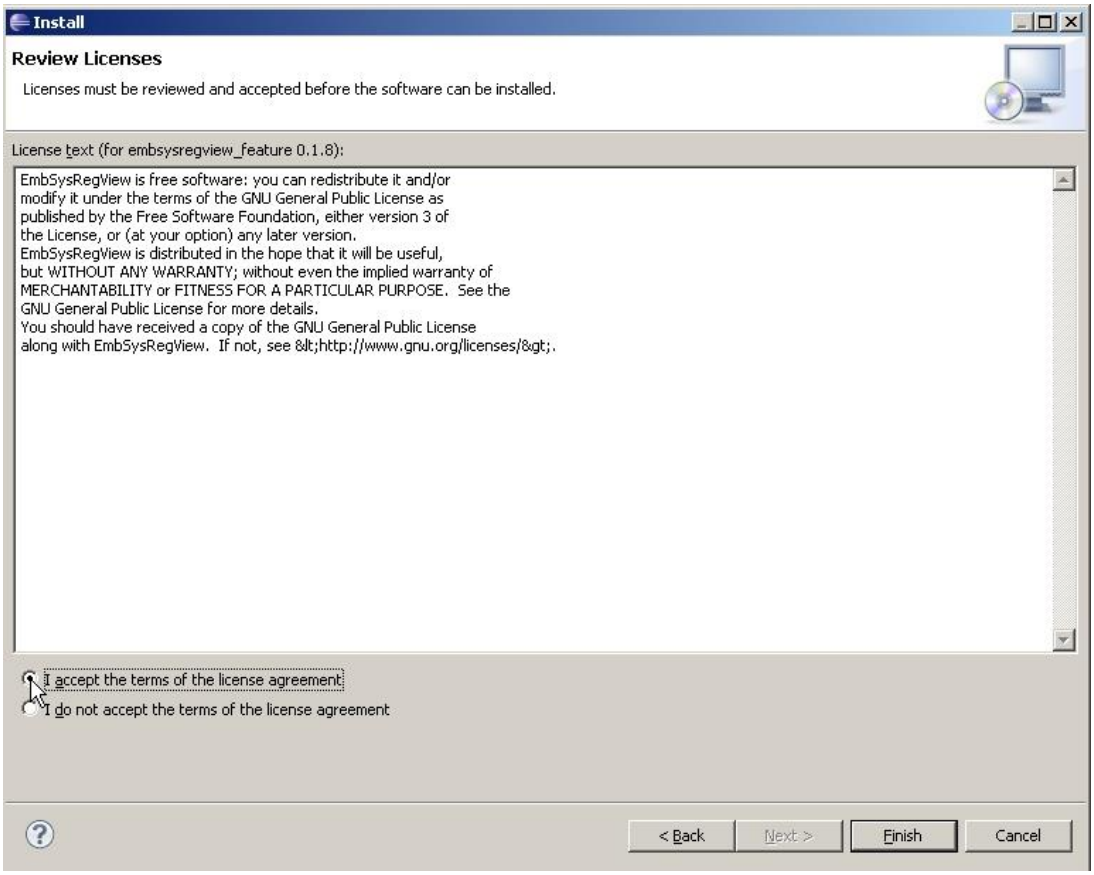
すべてのプラグインを選択して ”Next” をクリックします。



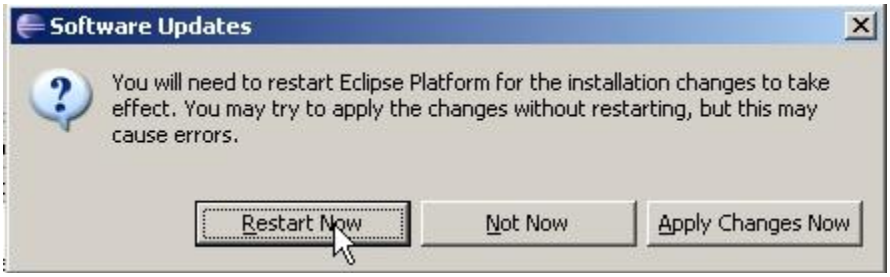
“Next” をクリックしてインストールの詳細を確認します。



ライセンスを確認し、”I accept the term of the license agreement” にチェックを入れ ”Finish” でウィンドウを閉じます。



Eclipse を再起動します。”Restart Now” をクリックしてください。

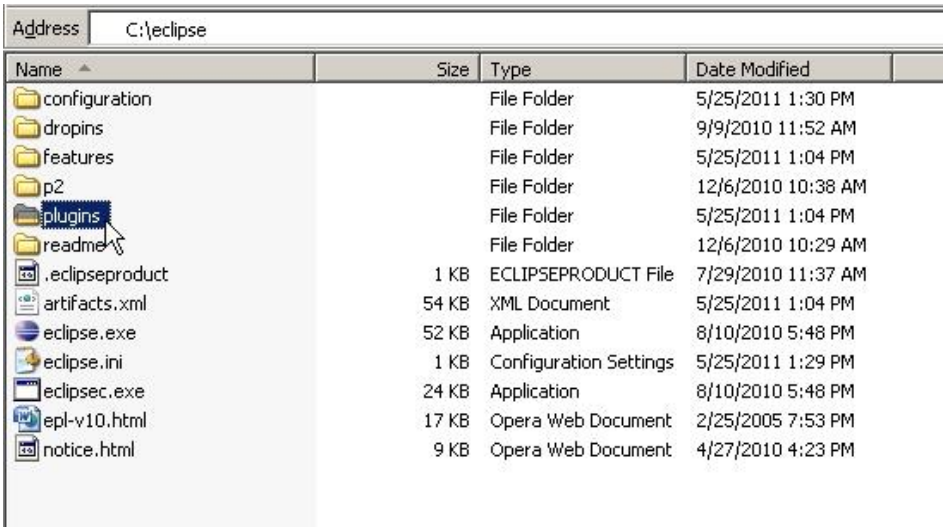


これで、”EmbSysRegView” のインストールが完了です。

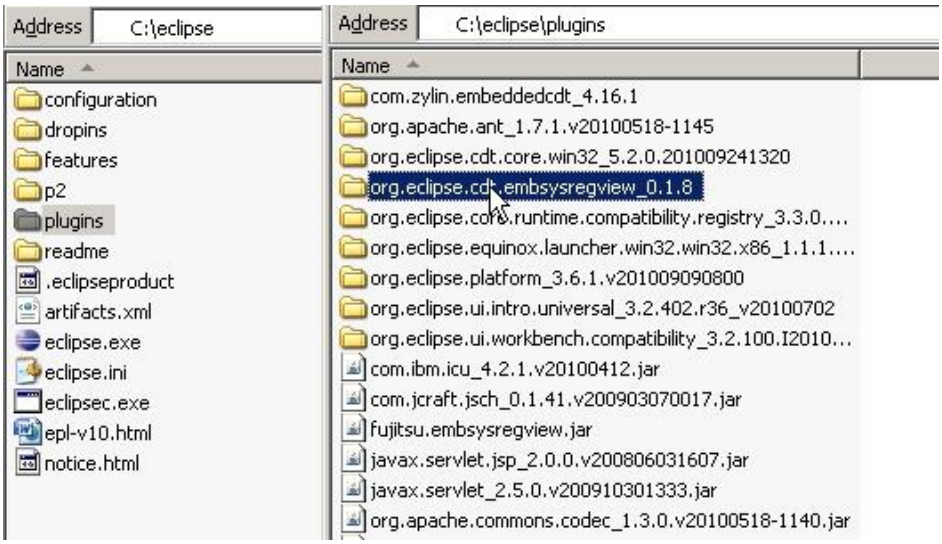
14.2 Register View の使用

FM3 のペリフェラルレジスタのビューをサポートするためには、FM3 用の xml ファイルを 2 つ使用する必要があります。本 xml ファイルは本書で使用するソフトウェアパッケージに含まれているので、Eclipse プラグインフォルダにコピーしてください。

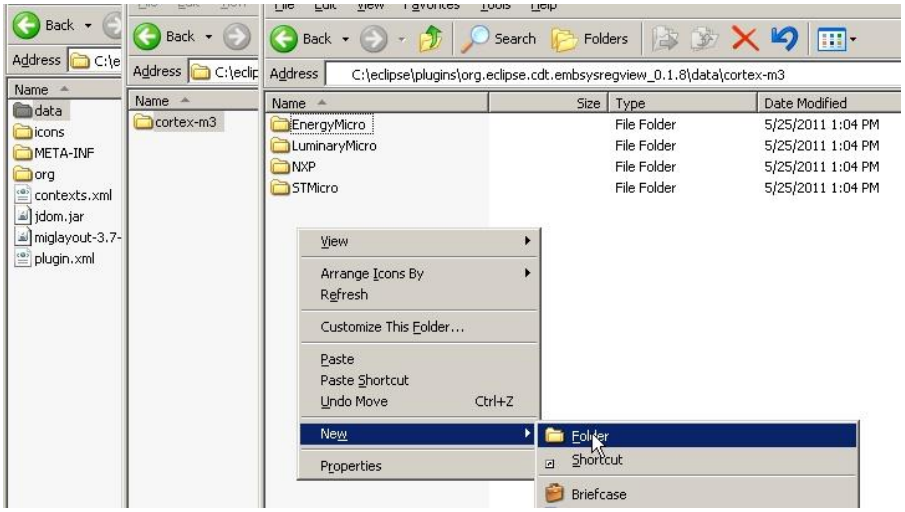
Eclipse のインストールフォルダは以下のような構造になっています。



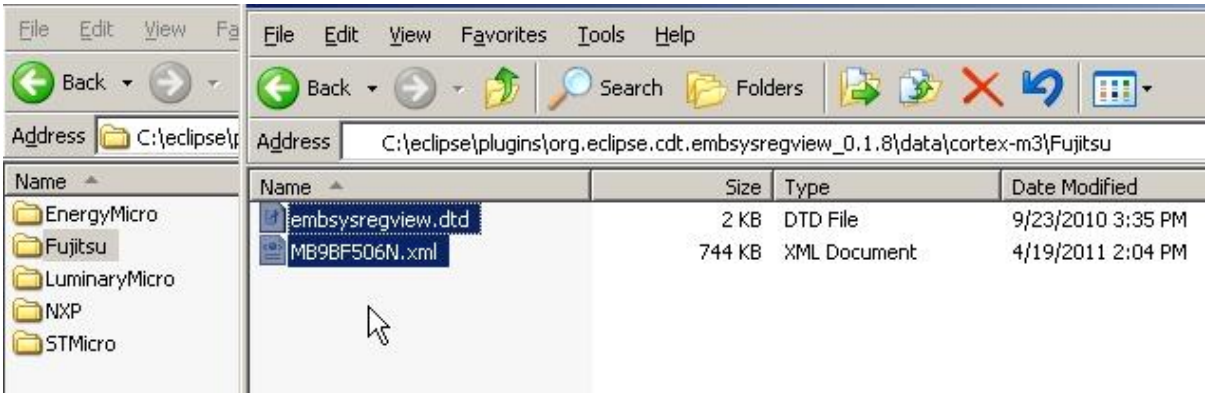
plugins フォルダを開き、インストールした “EmbSysRegView” プラグインのインストールファイルを探します。



フォルダを開き、¥data¥cortex-m3 のフォルダ内に新しいフォルダ (例えば Cypress) を作成します。

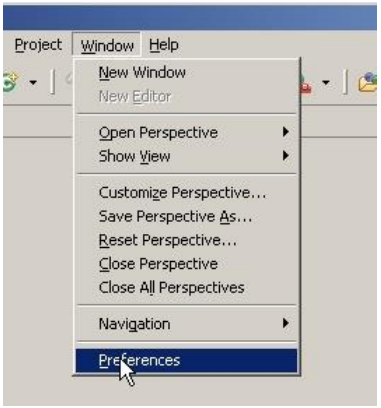


Cypress フォルダを作成後、”embsysregview.dtd” と ”MB9BF506N.xml” の両方のファイルを置きます。

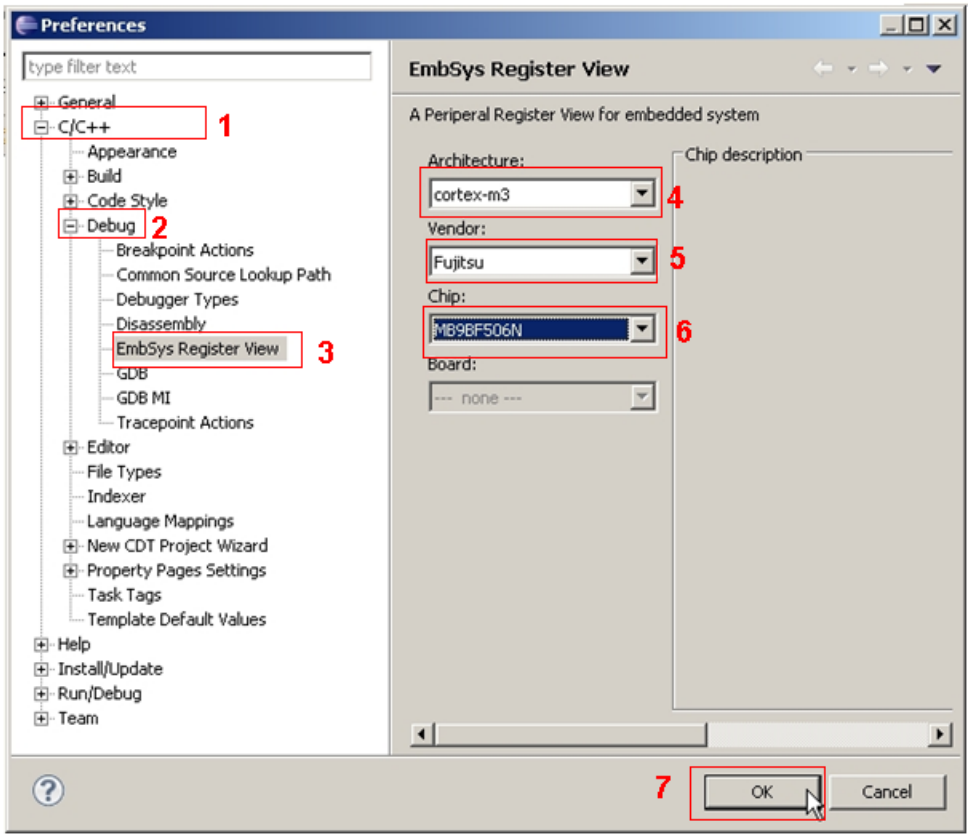


Eclipse IDE に戻り、インストールしたレジスタビューを使用します。

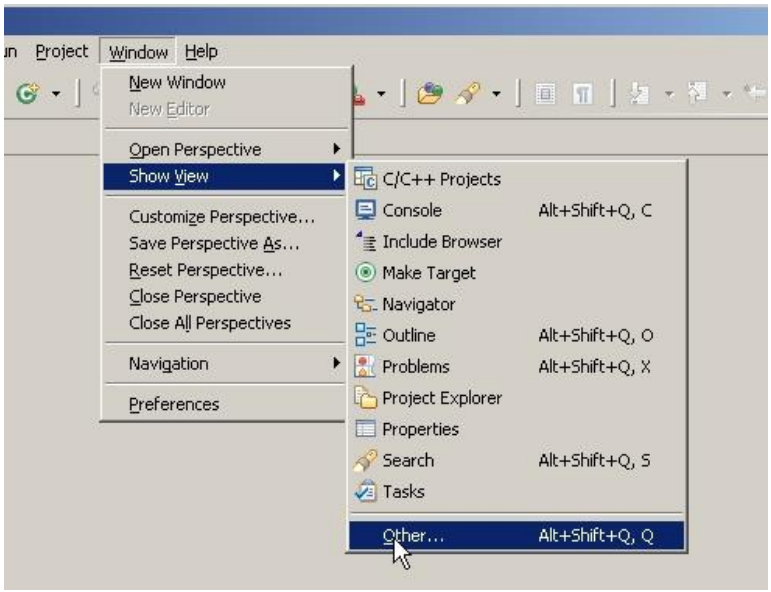
メニューの “Window” から ”Preferences” を開きます。



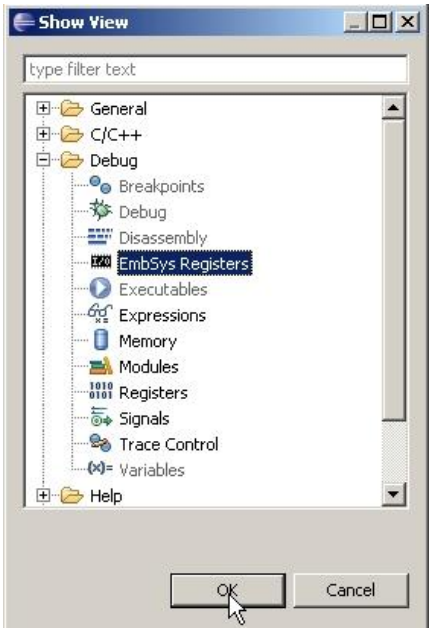
以下の図のように使用するデバイスを選択します。



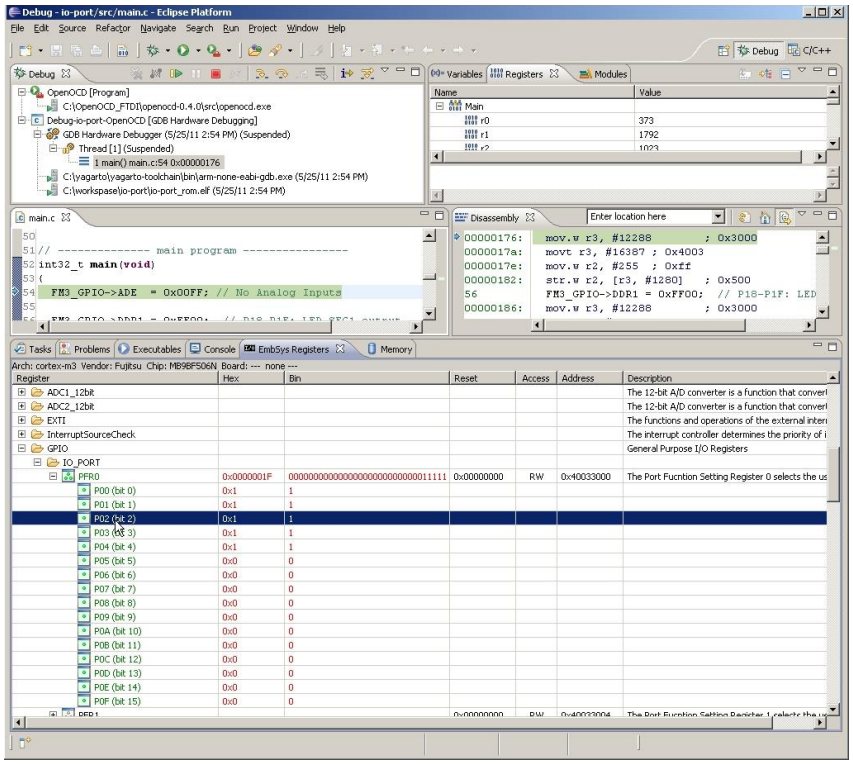
レジスタビューのコンフィグレーション確認後、ツールを使用します。CDT のデバッグパースペクティブ (詳細は 13 章を参照ください) からレジスタビューを開くには、”Window” -> ”Show View” -> ”Other…” を選択します。



“Debug” から ”EmbSys Registers” を選択し、”OK” をクリックしてください。



デバッグ中、ブレークポイントで実行をストップし選択したレジスタの値を確認できます。レジスタをダブルクリックすることで値のビューを開始できます。選択したレジスタは大文字になります。レジスタの値を変えると赤字になります。マウスポインタをレジスタに合わせると、レジスタの短い説明を見ることができます。



15 Eclipse の特徴

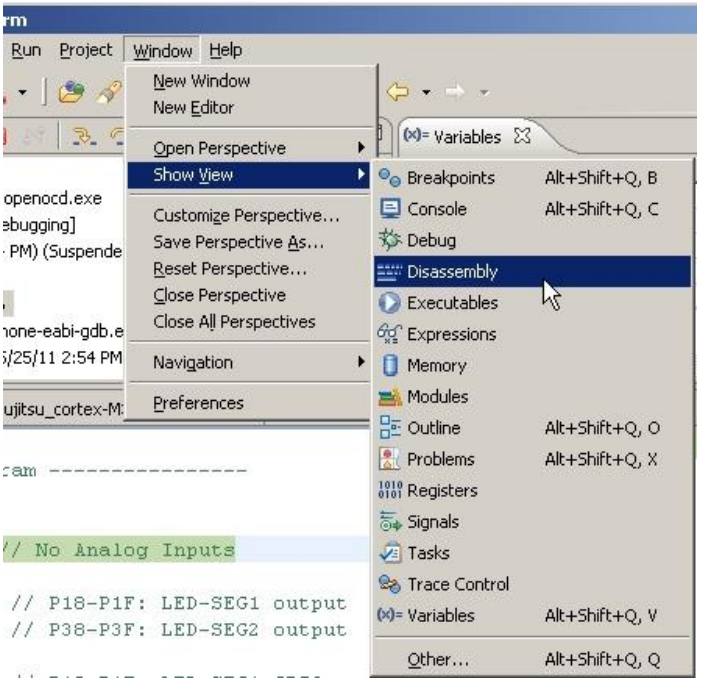
15.1 概要

Eclipse CDT は、ユーザが FM3 の組込みシステム開発をするにあたり多くのツールや特徴を持っています。

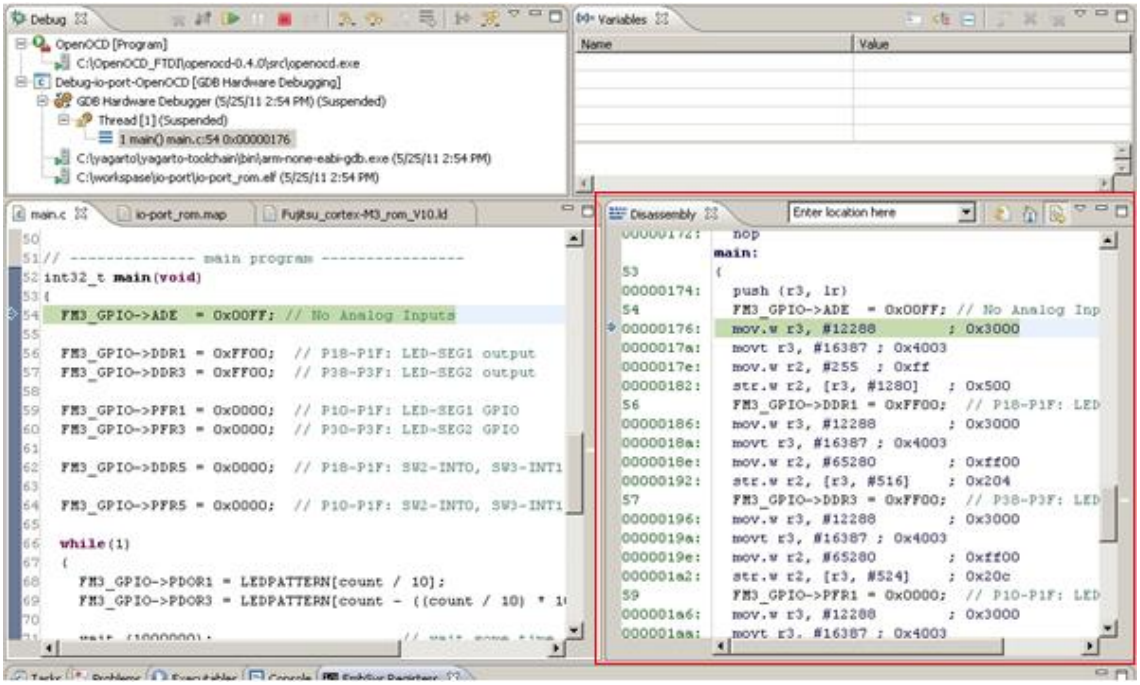
本章では、デバッグパースペクティブで使用できるツールを紹介します。

15.2 Disassembly ビュー

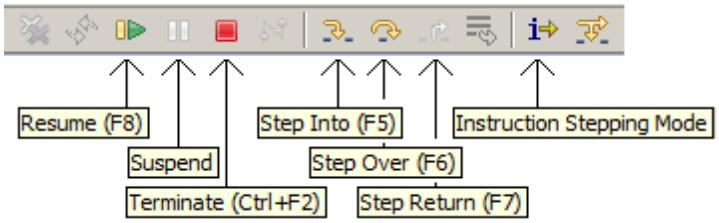
CDT デバッグパースペクティブ (詳細は 13 章を参照ください) の ”Disassembly” ビューを表示するためには、メニューの ”Window” -> ”Show View” -> ”Disassembly” を選択します。



ビューは、以下の図のとおり表示されます。



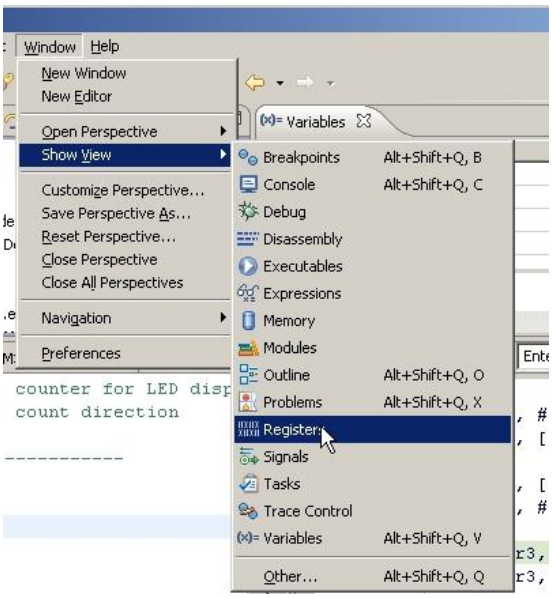
ユーザはいつでも ”Suspend” ボタンをクリックすることでブレークさせることができ、ブレーク時のインストラクションにポイントを当てます。プログラムを終了させる ”Terminate” ボタンと間違えないでください。



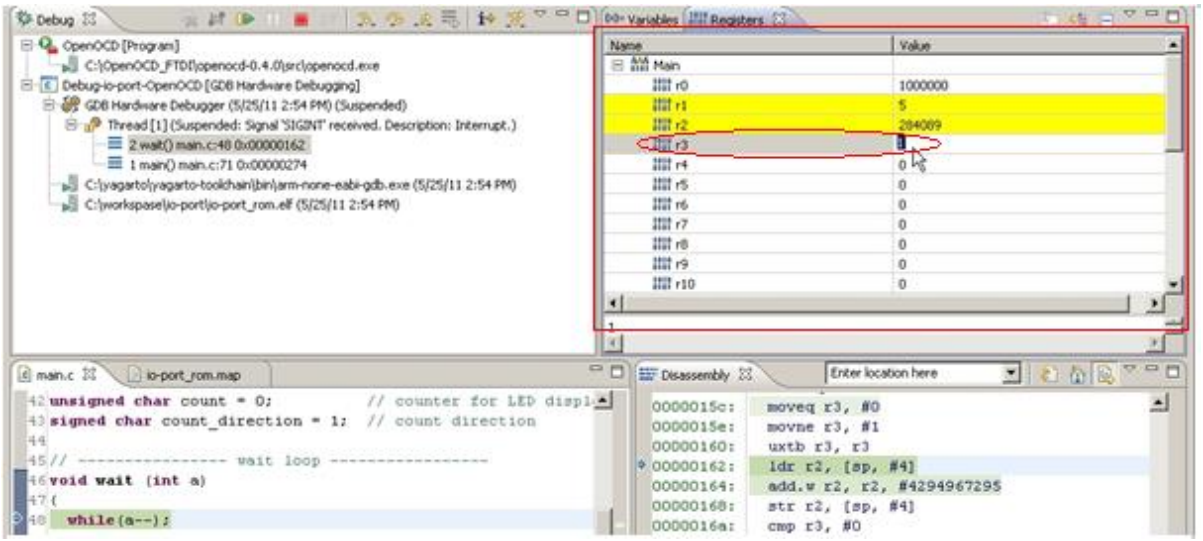
15.3 CPU Register ビュー

Eclipse CDT は、コアのレジスタにリードもしくはライトのアクセスができます。

“Window” -> ”Show View” -> ”Register” を選択してください。



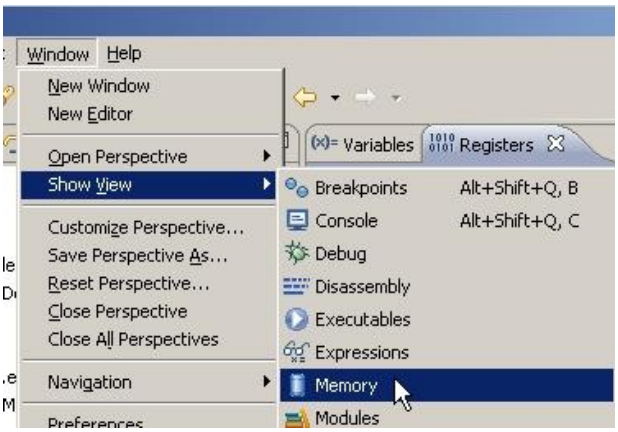
すべてのコアのレジスタとその値を表示します。コアのレジスタを見るには ”Main” ツリーを開いてください。



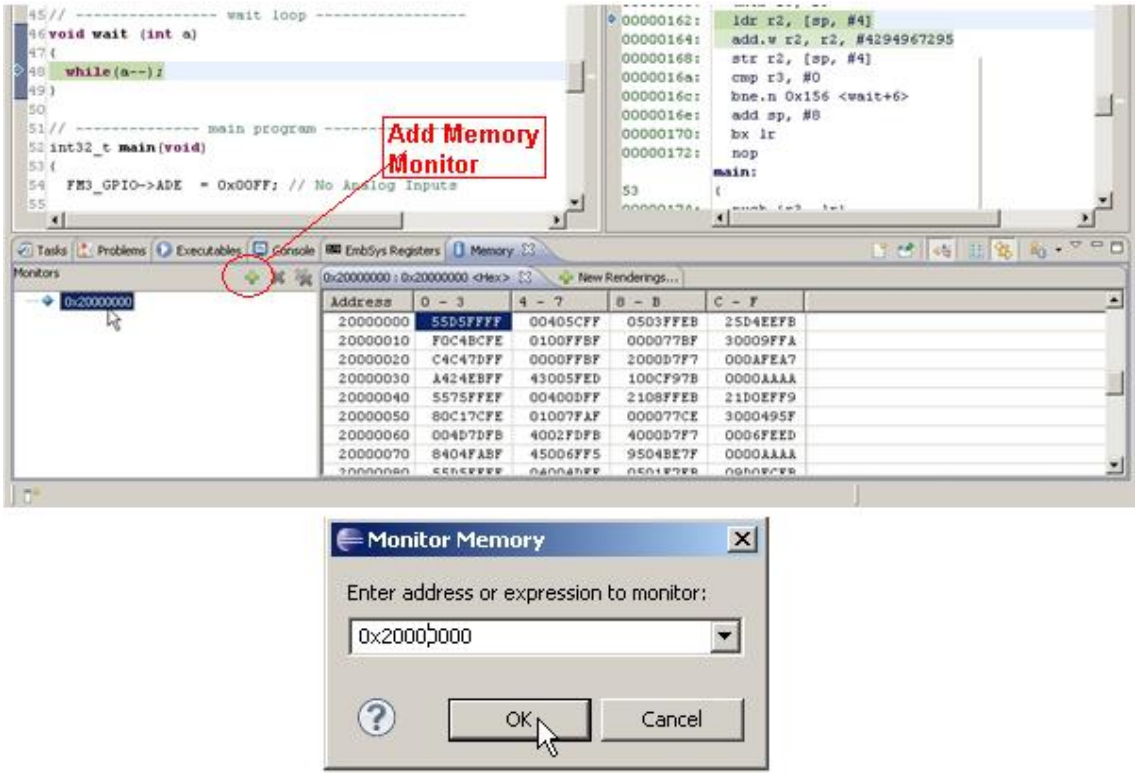
15.4 Memory ビュー

Eclupse のメモリモニタビューはデフォルトで設定されています。

“Winow” -> ”Show View” -> ”Memory” を選択してください。



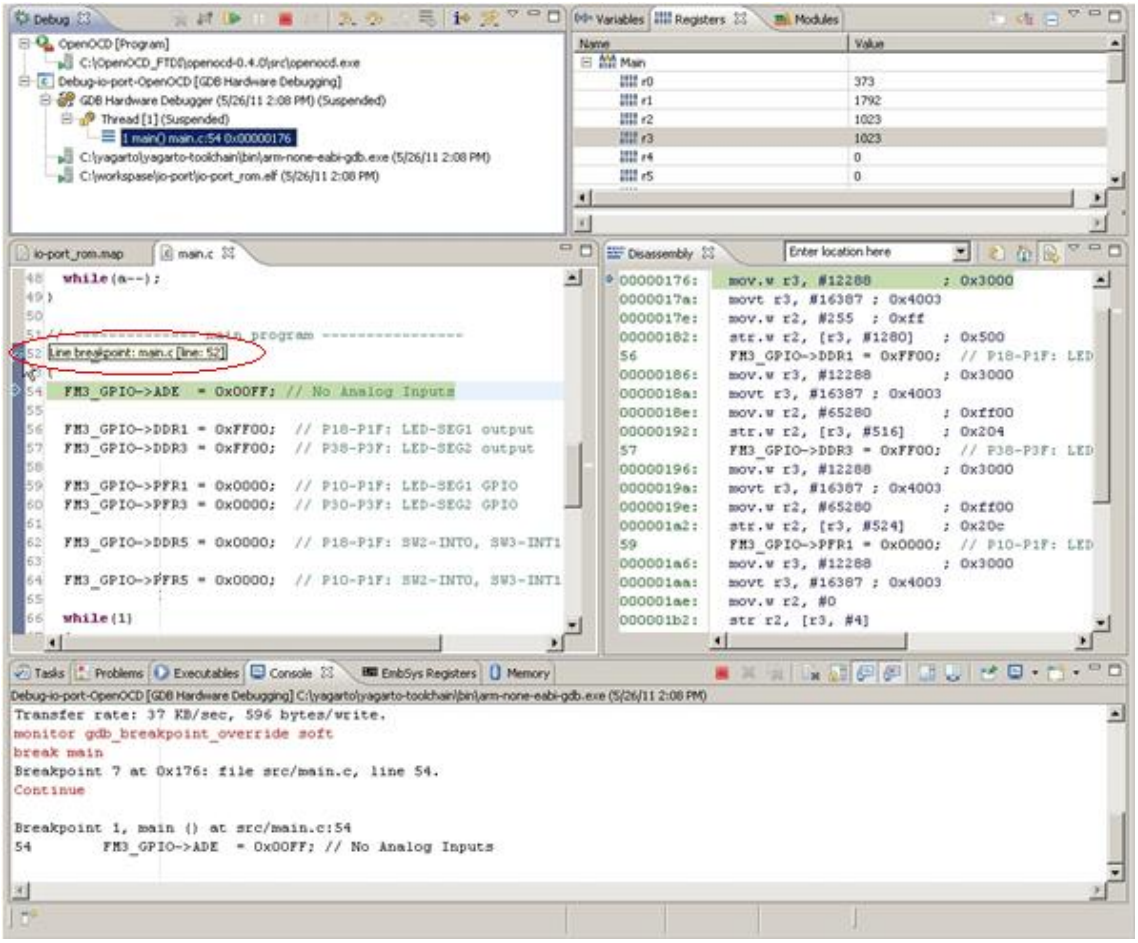
新しいメモリモニタを加えるには、モニタパネルの緑色のプラスボタンをクリックします。以下の図では 0x2000000 番地のメモリをモニタしています。



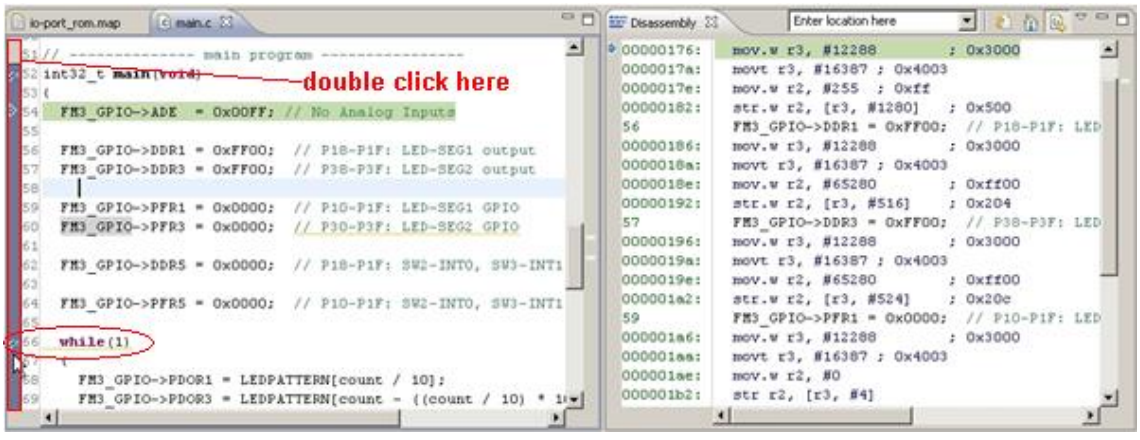
メモリアドレス (RAM や I/O 資源) は編集する場合、目的のアドレスをダブルクリックしてください。

15.5 ブレークポイントの使用

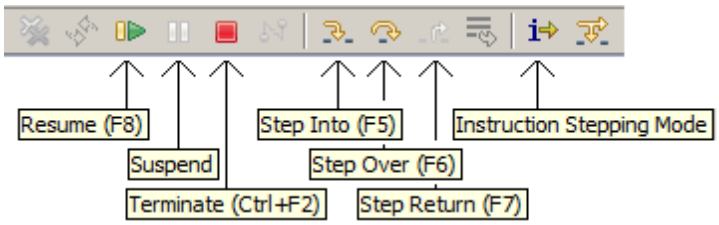
デバッグを開始後、デバッガはメイン関数にブレークポイントを設定します。



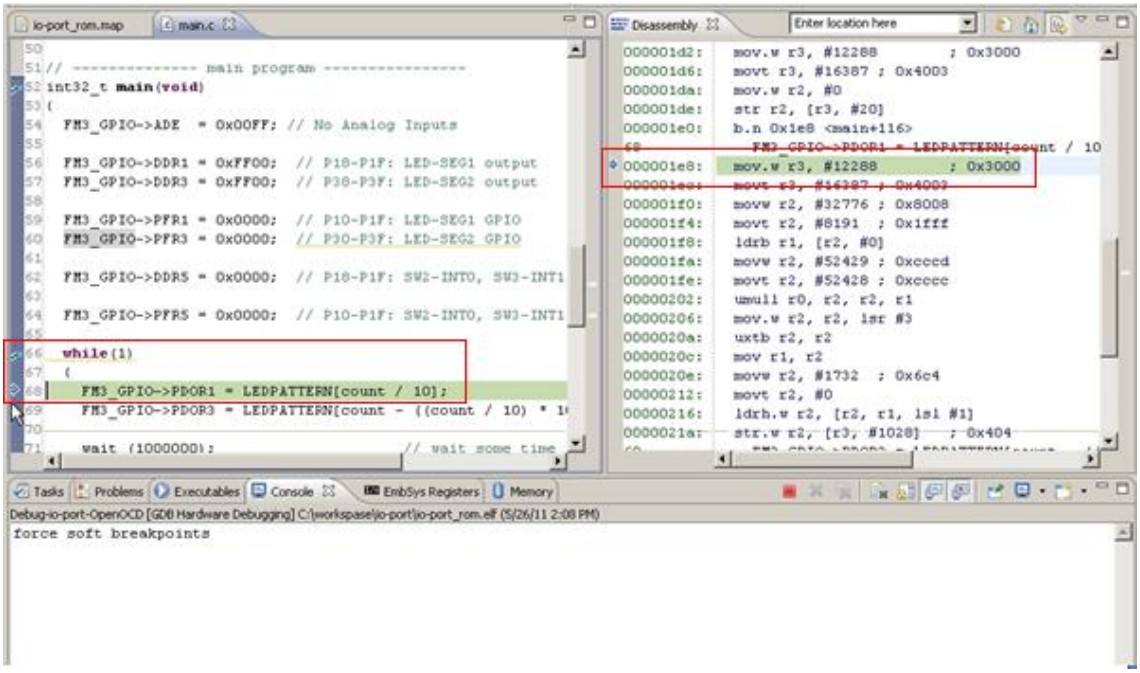
ほかのブレークポイントを設定するためには、左側のソースコードタブから行数をダブルクリックします。



デバッグを再開します。



ブレークポイントにヒットすると、以下の図のようになります。



16 付録

16.1 用語集

本書で使用する略語を下記にまとめます。

略語	意味	説明
*.bin (拡張子)	Binary Format File	プログラムデータのみが含まれているフォーマット
*.elf (拡張子)	Executable and Linkable Format	デバッグ情報 (シンボル, アドレス, モジュールなど) を含んだフォーマット
*.hex (拡張子)	Hexadecimal format file (Intel)	プログラムデータとアドレス情報を含んだフォーマット (Intel フォーマット)
*.mhx (拡張子)	Motorola Hexadecimal Format File	プログラムデータとアドレス情報を含んだフォーマット (Motorola S-Records フォーマット)
CDT	C/C++ Development Tooling	Eclipse のコンフィグレーションで使用されるツールチェーン
EABI	Embedded-Application Binary Interface	Linux 向けの、組込みソフトウェア向けインタフェースの標準フォーマット
FTDI	Future Technology Devices International Ltd.	社名, もしくは JTAG と USB の変換チップなどを意味する
JTAG	Joint Test Action Group	ハードウェア (本書では MCU) のテスト, デバッグで使用する規格であり、IEEE1149.1 として定められている
JRE	Java Runtime Environment	PC 上で Java アプレット (例えば Eclipse) を動作させるための仮想マシン
GDB	GNU Debugger	GNU ツールチェーンのデバッガ
GNU	“GNU’s not Unix”	GNU ツールチェーン、もしくはオープンソフトウェアを開発するプロジェクト名
LibUSB	Library for USB	オープンソースライブラリの USB ドライバであり、本書では Windows 向けにコンパイルしたものを使用している
None-EABI	None-Embedded-Application Binary Interface	Linux 向けでない、組込みソフトウェア向けインタフェースの標準フォーマット
OCD	On-Chip Debugger/Debugging	オンチップデバッグ向けのデバッガ。本書では JTAG プロトコルを使用
OpenOCD	Open Source On-Chip Debugger	オープンソースコードのデバッガ
YAGARTO	“Yet another GNU ARM tool chain”	Windows 向けに移植とコンパイルされた GNU ツールチェーン

16.2 リンク

16.2.1 ソフトウェア

Eclipse IDE:
http://download.eclipse.org/eclipse/downloads/

Yagarto Tool Chain:
www.yagarto.de

OpenOCD:
http://openocd.sourceforge.net/

LibUSB:
http://sourceforge.net/projects/libusb-win32/files/

Embedded System Register View Plug-In for Eclipse:
http://sourceforge.net/projects/embsysregview/

JRE:
http://java.com/

16.2.2 ハードウェア

IAR 社製　J-Link

http://www.iar.com/Global/Products/Hardware-Debug-probes/DS-J-Link-ARM-09.pdf

olimex 社製　ARM-USB-TINY

https://www.olimex.com/Products/ARM/JTAG/ARM-USB-TINY/

サイプレス社製　SK-FM3-176PMC-ETHERNET V1.1

http://www.cypress.com/SK-FM3-176PMC-ETHERNET

※URL は予告なく変更される可能性があります。

17 追加情報

サイプレスに関する情報は下記のウェブページから参照できます。

http://www.cypress.com

18 改訂履歴

文書名: AN204421 -FM3 GNU ツールチェーンを使用した開発環境構築方法

文書番号: 002-04422

版	ECN 番号	変更者	発行日	変更内容
**	-	KHAS	07/16/2015	スパンションアプリケーションノート AN706-00061-1v1-J をサイプレスとして登録したものです。
*A	5666837	KHAS	03/21/2017	これは英語版の 002-04421 Rev. *A を翻訳した日本語版です。

セールス, ソリューションおよび法律情報

ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、サイプレスのロケーション ページをご覧ください。

製品

ARM® Cortex® Microcontrollers	cypress.com/arm
車載用	cypress.com/automotive
クロック&バッファ	cypress.com/clocks
インターフェース	cypress.com/interface
IoT (モノのインターネット)	cypress.com/iot
メモリ	cypress.com/memory
マイクロコントローラ	cypress.com/mcu
PSoC	cypress.com/psoc
電源用 IC	cypress.com/pmic
タッチ センシング	cypress.com/touch
USB コントローラー	cypress.com/usb
ワイヤレス／RF	cypress.com/wireless

PSoC® ソリューション

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

サイプレス開発者コミュニティ

フォーラム | WICED IOT Forums | Projects | ビデオ | ブログ | トレーニング | Components

テクニカルサポート

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2013-2017. 本書面は、Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社（以下、「Cypress」という。）に帰属する財産である。本書面（本書面に含まれ又は言及されているあらゆるソフトウェア又はファームウェア（以下、「本ソフトウェア」という。）を含む）は、アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき、Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し、また、本段落で特に記載されているものを除き、Cypress の特許権、著作権、商標権又はその他の知的財産権のライセンスを一切許諾していない。本ソフトウェアにライセンス契約書が伴っておらず、かつ、あなたが Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意をしていない場合、Cypress は、あなたに対して、（1）本ソフトウェアの著作権に基づき、（a）ソースコード形式で提供されている本ソフトウェアについて、Cypress ハードウェア製品と共に用いるためにのみ、組織内部でのみ、本ソフトウェアの修正及び複製を行うこと、並びに（b）Cypress のハードウェア製品ユニットに用いるためにのみ、（直接又は再販売者及び販売代理店を介して間接のいずれかで）エンドユーザーに対して、バイナリーコード形式で本ソフトウェアを外部に配布すること、並びに（2）本ソフトウェア（Cypress により提供され、修正がなされていないもの）に抵触する Cypress の特許権のクレームに基づき、Cypress ハードウェア製品と共に用いるためにのみ、本ソフトウェアの作成、利用、配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス（サブライセンスの権利を除く）を付与する。本ソフトウェアのその他の使用、複製、修正、変換又はコンパイルを禁止する。

適用される法律により許される範囲内で、Cypress は、本書面又はいかなる本ソフトウェアに関しても、明示又は黙示をとわず、いかなる保証（商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない）も行わない。適用される法律により許される範囲内で、Cypress は、別途通知することなく、本書面を変更する権利を留保する。Cypress は、本書面に記載のあるいかなる製品又は回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報（あらゆるサンプルデザイン情報又はプログラムコードを含む）は、参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計し、プログラムし、かつテストすることは、本書面のユーザーの責任において行われるものとする。Cypress 製品は、兵器、兵器システム、原子力施設、生命維持装置若しくは生命維持システム、蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム、汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分として用いるため、又はシステムの不具合が人身傷害、死亡若しくは物的損害を生じさせることになるその他の使用（以下、「本目的外使用」という。）のためには、設計、意図又は承認されていない。重要な構成部分とは、装置又はシステムのその構成部分の不具合が、その装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できる、機器又はシステムのあらゆる構成部分をいう。Cypress 製品のあらゆる本目的外使用から生じ、若しくは本目的外使用に関連するいかなる請求、損害又はその他の責任についても、Cypress はその全部又は一部をとわず一切の責任を負わず、かつ、あなたは Cypress をそれら一切から免除するものとし、本書により免除する。あなたは、Cypress 製品の本目的外使用から生じ又は本目的外使用に関連するあらゆる請求、費用、損害及びその他の責任（人身傷害又は死亡に基づく請求を含む）から Cypress を免責補償する。

Cypress、Cypress のロゴ、Spansion、Spansion のロゴ及びこれらの組み合わせ、WICED、PSoC、CapsSense、EZ-USB、F-RAM、及び Traveo は、米国及びその他の国における Cypress の商標又は登録商標である。Cypress の商標のより完全なリストは、cypress.com を参照のこと。その他の名称及びブランドは、それぞれの権利者の財産として権利主張がなされている可能性がある。