eCosPro CDT (1.6.5) plug-in user's guide for Eclipse 2018-09 (4.9.0)

Alex Schuilenburg
Ross Younger
Jonathan Larmour

eCosPro CDT (1.6.5) plug-in user's guide for Eclipse 2018-09 (4.9.0)
by Alex Schuilenburg, Ross Younger, and Jonathan Larmour

Publication date 3 November 2020
Copyright © 2010-2013, 2016-2018 eCosCentric Limited

Legal Notice

Non-Commercial Use. This documentation accompanies eCosCentric's eCosPro CDT plug-in (the "*Software*"), both of which may **ONLY**[*] be used for Non-Commercial purposes (as defined in the eCosPro Non-Commercial Public License). Redistribution of the *Software* and all accompanying documentation, in all forms, is **NOT PERMITTED**.

[*] **Commercial Use.** Legitimate holders of an eCosPro License are subject to the terms and conditions of that license and may make Commercial Use of the *Software* and its accompanying documentation. Redistribution of the *Software* and all accompanying documentation, in all forms, remains **NOT PERMITTED**.

Disclaimer of Warranties and Limitation of Liability. UNLESS OTHERWISE SEPARATELY UNDERTAKEN BY eCosCentric Limited (the "*Licensor*"), TO THE EXTENT POSSIBLE, THE *Licensor* OFFERS THE *Software* AS-IS AND AS-AVAILABLE, AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE *Software*, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHER. THIS INCLUDES, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OR ABSENCE OF ERRORS, WHETHER OR NOT KNOWN OR DISCOVERABLE, OR THAT THE *Software* WILL MEET *Your* REQUIREMENTS. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE *Software* IS *Yours*. SHOULD THE *Software* PROVE DEFECTIVE AND You ASSUME ANY COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. WHERE DISCLAIMERS OF WARRANTIES ARE NOT ALLOWED IN FULL OR IN PART, THIS DISCLAIMER MAY NOT APPLY TO *You*.

TO THE EXTENT POSSIBLE, IN NO EVENT WILL THE *Licensor* BE LIABLE TO *You* ON ANY LEGAL THEORY (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE) OR OTHERWISE FOR ANY DIRECT, SPECIAL, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, EXEMPLARY, OR OTHER LOSSES, COSTS, EXPENSES, OR DAMAGES ARISING OUT OF USE OF THE *Software*, EVEN IF THE *Licensor* HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES, COSTS, EXPENSES, OR DAMAGES. WHERE A LIMITATION OF LIABILITY IS NOT ALLOWED IN FULL OR IN PART, THIS LIMITATION MAY NOT APPLY TO *You*.

The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Intellectual Property Rights. The intellectual property rights in the *Software* shall remain vested in the *Licensor*.

Trademarks

eCos®, eCosCentric® and eCosPro® are registered trademarks of eCosCentric Limited.

The Eclipse™ name and logo are the intellectual property of the Eclipse Foundation.

All other brand and product names, trademarks, and copyrights are the property of their respective owners.

Table of Contents

List of Figures

List of Examples

Introduction

[Eclipse](#) is commonly recognised as an integrated development environment (IDE) for computer programming which contains an extensible plug-in system.

Eclipse itself is a very small open-source framework and workbench. The IDE's functionality comes from sets of subsystems which are implemented in one or more plug-ins. Although Eclipse is widely used for developing Java applications, the [CDT Project](#) provides a set of plugins to create a fully functional C and C++ IDE, allowing developers to create, edit and debug projects based on C and C++. This distribution of Eclipse and CDT plug-ins is commonly known as Eclipse/CDT.

The eCosPro® CDT plug-in further enhances Eclipse/CDT allowing embedded application developers to create, develop and debug eCosPro applications and libraries for embedded hardware. An eCosPro application or library is typically an application or libary that runs on top of the eCos® and eCosPro Real-Time Operating System (RTOS), which in turn runs within embedded hardware such as the circuit board within washing machines, satellites, automobiles and so on.

Enhancements provided by the eCosPro CDT plug-in include the following:

eCos Configuration Projects

> These are distinct project types which contain at least one eCos configuration (files with the extension `.ecc`). The creation of these projects within the Eclipse framework allows developers to configure and build the eCos RTOS as well as eCosPro components provided with the RTOS.
>
> *eCos Configuration Projects* can be conveniently reused across multiple *eCos C/C++ Application Projects* and/or static library projects, and can also be shared via the Eclipse *Team Provider* mechanism which interfaces with various source control systems. With these projects it is possible to edit eCos configurations using the *[eCos Configuration Tool](#)* as well as manage and rebuild eCos libraries.

eCos C/C++ Application Projects

> These are Eclipse/CDT Application projects but are associated with a specific *eCos Configuration Project*.
>
> *eCos C/C++ Application Projects* will also be rebuilt automatically if any changes are made to their associated *eCos Configuration Project*.

eCos Hardware Debugging

> Downloading, programming and debugging of executable eCos C/C++ applications to remote target hardware using a [hardware debugger](#) is enhanced by this plug-in. *[OpenOCD](#)* and the Ronetix PEEDI are both supported, as well as other hardware debuggers. On specific supported targets, BDM may also be available.

eCos Remote Debugging

> Downloading and debugging of executable eCos applications to remote target hardware running RedBoot or GDB stubs is also enhanced by this plug-in.

eCosPro GNU Toolchains

> A GNU cross development toolchain provided by eCosCentric, or a *meta-toolchain* termed the *eCos toolchain*, may be selected for compiling and debugging the libraries of *eCos Configuration Projects* and executables of *eCos C/C++ Application Projects*. This eliminates the requirement of developers to select a specific toolchain for use by Eclipse/CDT.

Additional Features

> - Robust handling of remote target communication failures
>
> - The provision of information about eCos kernel threads while debugging (current with RedBoot or GDB Stubs only)
>
> - The facility to insert hardware-assisted breakpoints where supported by the GDB stub or hardware debugger

Note

The figures of screenshots within this guide are illustrations and may not reflect exactly what you may encounter when working through this guide. The exact contents, styles, colours, fonts and layout will vary according to your host operating system, the theme for your workspace, as well as the window manager you use on the host operating system.

Chapter 1. Initial Reading

This document describes the process of configuring Eclipse and CDT in order to develop eCos applications. It is not intended for this document to provide full documentation on using Eclipse/CDT itself, as that is better covered in the user guides and documentation that come with Eclipse and CDT. As a result, some familiarity with Eclipse and CDT is assumed.

More general user guides for the Eclipse workbench and for C/C++ development using CDT are accessible via the *Help*→*Help Contents* menu item within the Eclipse workbench, and also on-line at the Eclipse 2018-09 documentation site.

In particular, it is recommended to read the first documentation topic, the Workbench User Guide which will provide an overview of the basic structure of Eclipse, and important concepts and terms to be familiar with before reading further documentation. We then recommend reading the C/C++ Development User Guide which provides documentation specific to the CDT plug-in for Eclipse.

Notes

- The CDT documentation primarily caters for those developing native applications to run directly on the developer's computer. It does not fully deal with debugging of remote targets and certainly does not cover features specific to the eCosPro CDT plug-in for eCos development. Therefore in areas of overlap, the remaining documentation should be considered to supersede the CDT documentation.

- All installed eCosPro documentation is accessible under the *Help*→*Help Contents*→*eCosPro* branch of Eclipse Help.

Chapter 2. eCosPro CDT plug-in Macros

This chapter describes the eCosPro CDT plug-in Macros available to many of the plug-in's settings which are referenced throughout this documentation. Where these macros appear in a field value of a plug-in form, they are expanded to the values indicated below before they are passed to the underlying infrastructure. The macros and their default values are:

${eclipse_hostname}
This macro is the hostname on which Eclipse is currently running. It is often used as a parameter to the Ronetix PEEDI configuration script in order that the latter may correctly load the settings for accessing the target hardware from the Eclipse build host.

${eCosInstallDir}
This macro represents and has its value set to the installation directory where eCos sub-directories such as etc, include and lib are installed when the associated *eCos Configuration Project* is built.

This is the same value as the environment variables ${ECOS_INSTALL_DIR} for Linux or %ECOS_INSTALL_DIR% for Windows command shells.

${openocd_config_filename} Current default value: openocd.cfg

This is the name of the OpenOCD configuration file that may be used to initialise and access the target hardware through OpenOCD. This file is placed in the ${eCosInstallDir}/etc sub-directory upon creation of the eCosPro RTOS library.

${peedi_config_filename}
Current default value: peedi.cfg

This is the name of the PEEDI configuration file that may be used to initialise and access the target hardware through the Ronetix PEEDI. This file is placed in the ${eCosInstallDir}/etc sub-directory upon creation of the eCosPro RTOS library.

${peedi_gdb_port}
Current default value: **2000**

This is the GDB port for the Ronetix PEEDI which is normally defined within the peedi.cfg configuration file.

Note

The default values may change in later revisions of eCosPro and may also vary according to the eCosPro configuration.

Chapter 3. Quick Start / Walkthrough

This chapter shows how to create a simple eCos *"Hello World"* application project from scratch, using the included project template and a *default* eCos configuration on the target hardware.

1. Startup Eclipse.

   If this is your first time running Eclipse, you will be asked to select a workspace. This is where your projects, folders and files will be kept. Enter a suitable location, or leave the default location selected. You will then be presented with the *Welcome* screen which you can also return to at any time from the menu option: `Help`→`Welcome`

2. Open the *"C Project"* dialog window:

   Method 1:

   Select the New Project wizard for your project from the menu `File`→`New`→`Project` (Figure 3.1, "New Project from Menu") or toolbar (Figure 3.2, "New Project from Toolbar").

   ## Figure 3.1. New Project from Menu
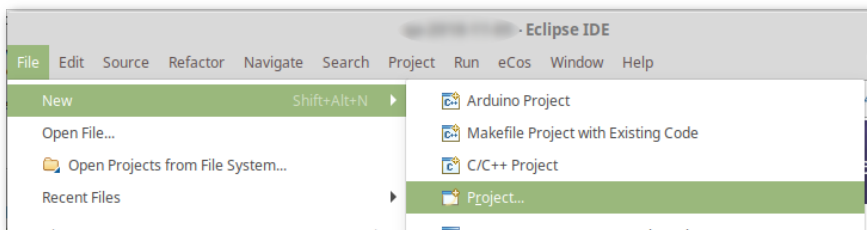
   ## Figure 3.2. New Project from Toolbar
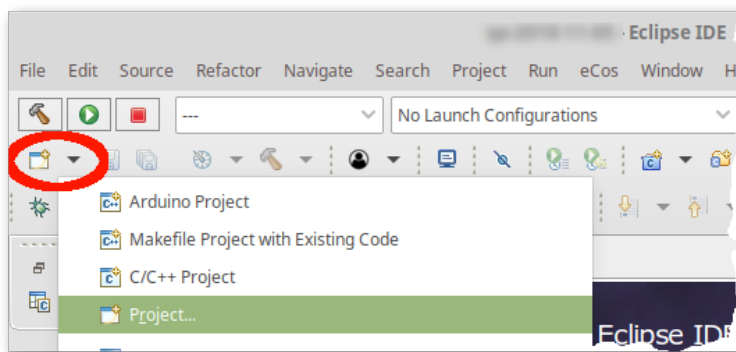
   > **Note**
   >
   > If you have run Eclipse before, and have the *C/C++ Perspective* active, your `File`→`New` sub-menu may include both the `C Project` and `C/C++ Project` options. In this case select `File`→`New`→`C Project`. This will skip the following *"New Project"* dialog and take you directly to the *"C Project"* dialog window illustrated in Figure 3.5, "C Project".

   Method 2:

   From the Welcome Screen, select either `Create a new C project` or `Create a New C++ project`, as highlighted below.

Figure 3.3. Welcome Screen
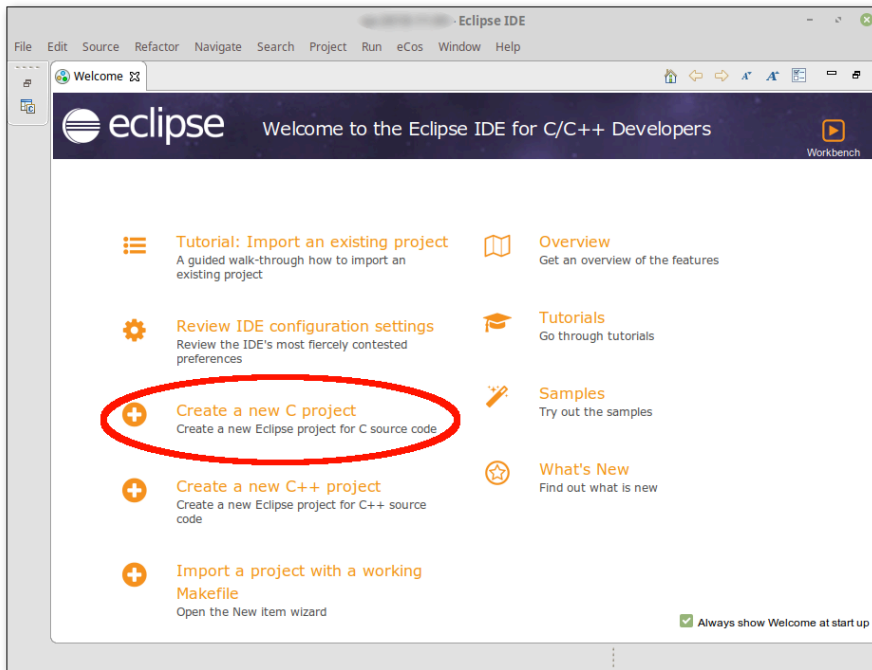
3. At the *"New Project"* dialog window, select: `C/C++` → `C Project` and select the `Next` button as illustrated in Figure 3.4, "New Project"

Figure 3.4. New Project
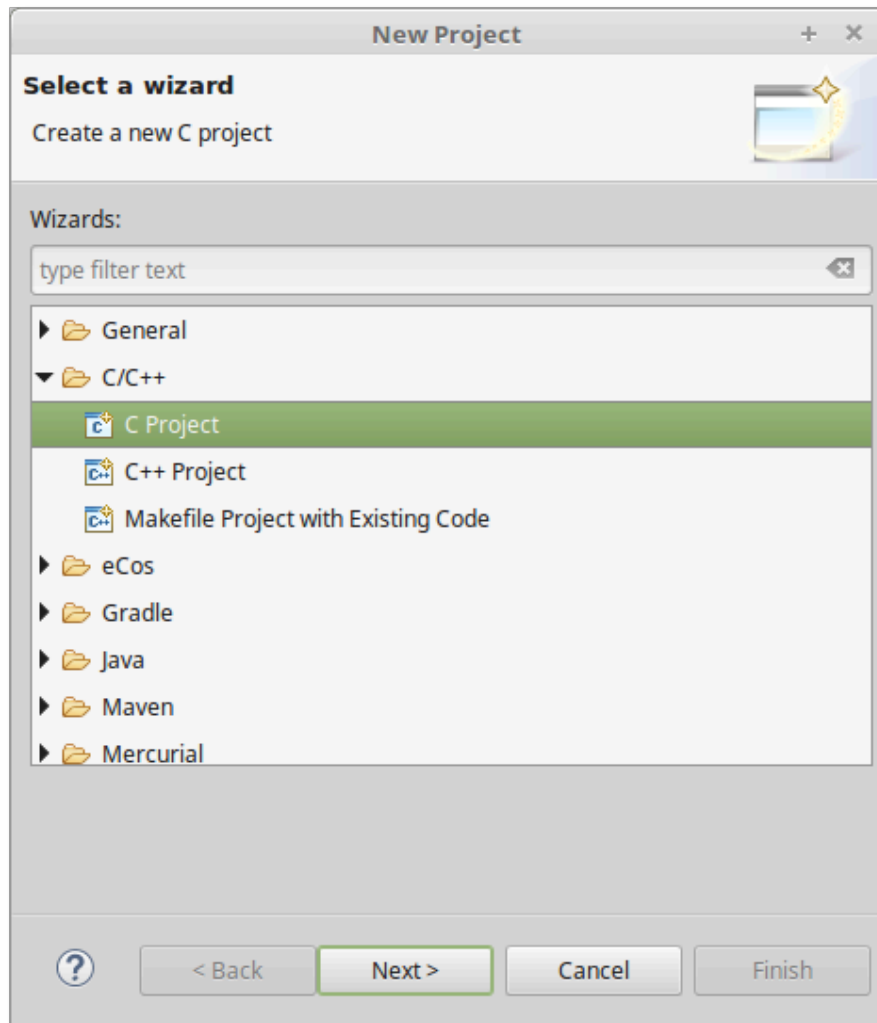
> **Tip**
>
> A C++ project may contain C code, but not vice versa.

4. A *"C Project"* dialog window will appear prompting you to "Create C project of selected type". In this example we will create an *eCos Managed Make Application Project* of subtype *Hello World Managed C Project*.

Select **eCos Managed Make Application**→`Hello World Managed C Project` as illustrated in .

Figure 3.5. C Project
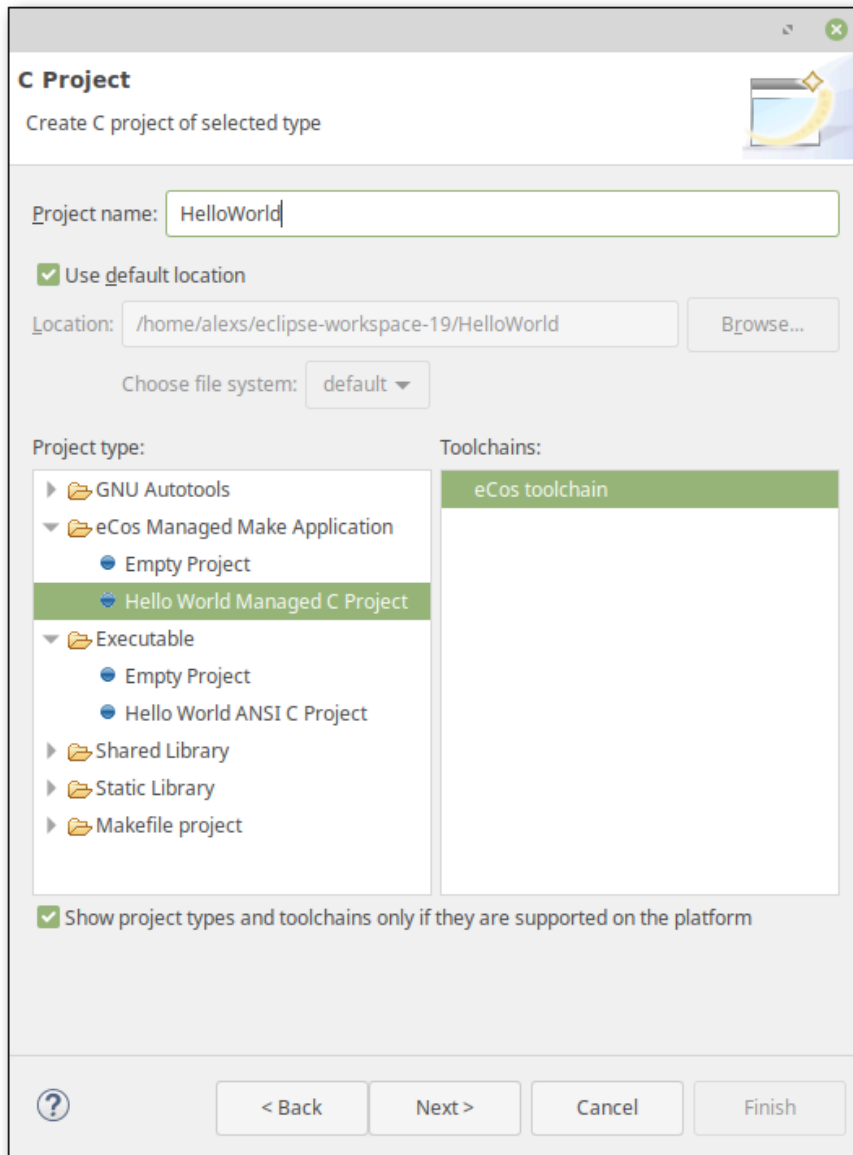
Normally the **Empty Project** subtype is used as the starting point for new projects, but the **Hello World Managed C Project** subtype is provided in order to create a project for demonstration purposes which already contains an example application. This application is a simple C program which, when run on the target, will output "Hello World" on the console. In every other respect it is the same as a normal eCos Managed Make Application Project, and only exists to easily demonstrate Eclipse/CDT.

5. Enter a name for your project next to the *Project name* option. **HelloWorld** was chosen in this example. If desired, customise its location.

> **Note**
>
> Spaces are not permitted in the name of a project or workspace nor are they allowed in the path to the workspace. This is because Eclipse uses the GNU make utility for which spaces have a specific meaning.

6. Check that the **eCos toolchain** is selected as illustrated in Figure 3.5, "C Project" and select the *Next* button.

The **"eCos toolchain"** entries are smart wrappers which automatically detect the correct GNU toolchain (gcc, gdb, …) to invoke based on the eCos configuration. If you later change the project to use a different eCos build, the appropriate toolchain will be invoked based on the settings within the associated eCos configuration.

7. The *Settings* page shown in Figure 3.6, "Hello World Settings" appears to provide a number of fields to customise the *Hello World* project. Fill these in appropriately, and press **Next**.

Note

This page does not appear if you selected an **Empty Project**.

Figure 3.6. Hello World Settings

8. The *Select Configurations* page appears as illustrated in Figure 3.7, "Select Configurations". It is not normally necessary to change anything here, so press **Next** again.

Figure 3.7. Select Configurations
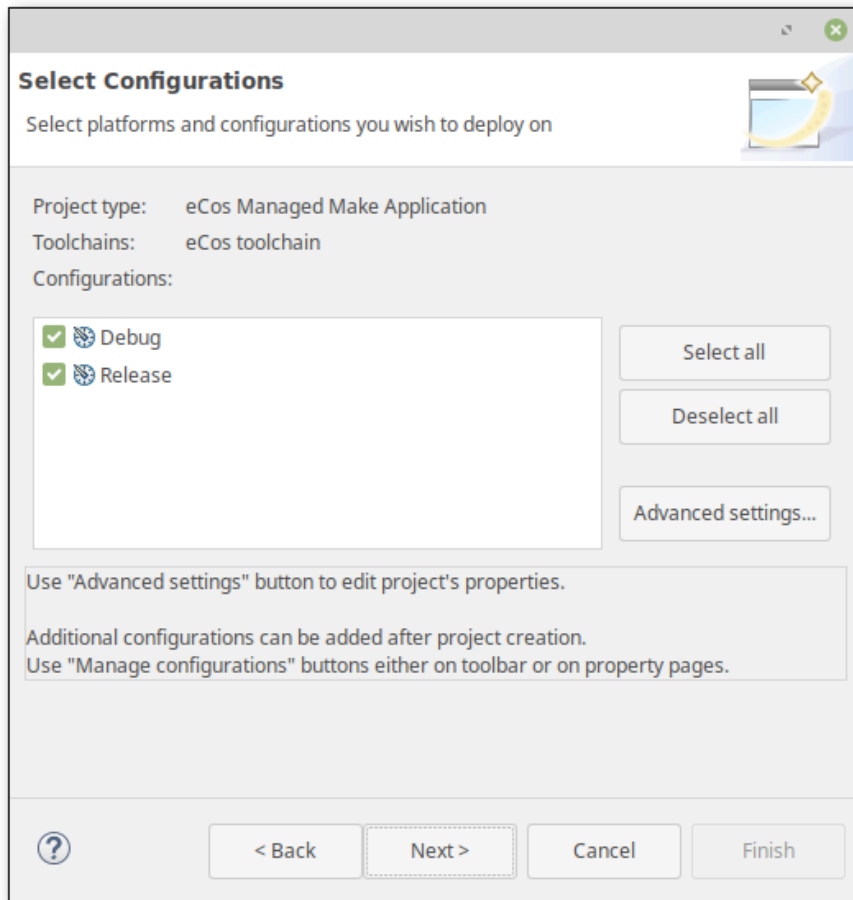
9. The *eCos Application Project configuration* page appears as illustrated in Figure 3.8, "eCos Application Project configuration". We are going to create a new eCos configuration, so select that option.

Figure 3.8. eCos Application Project configuration
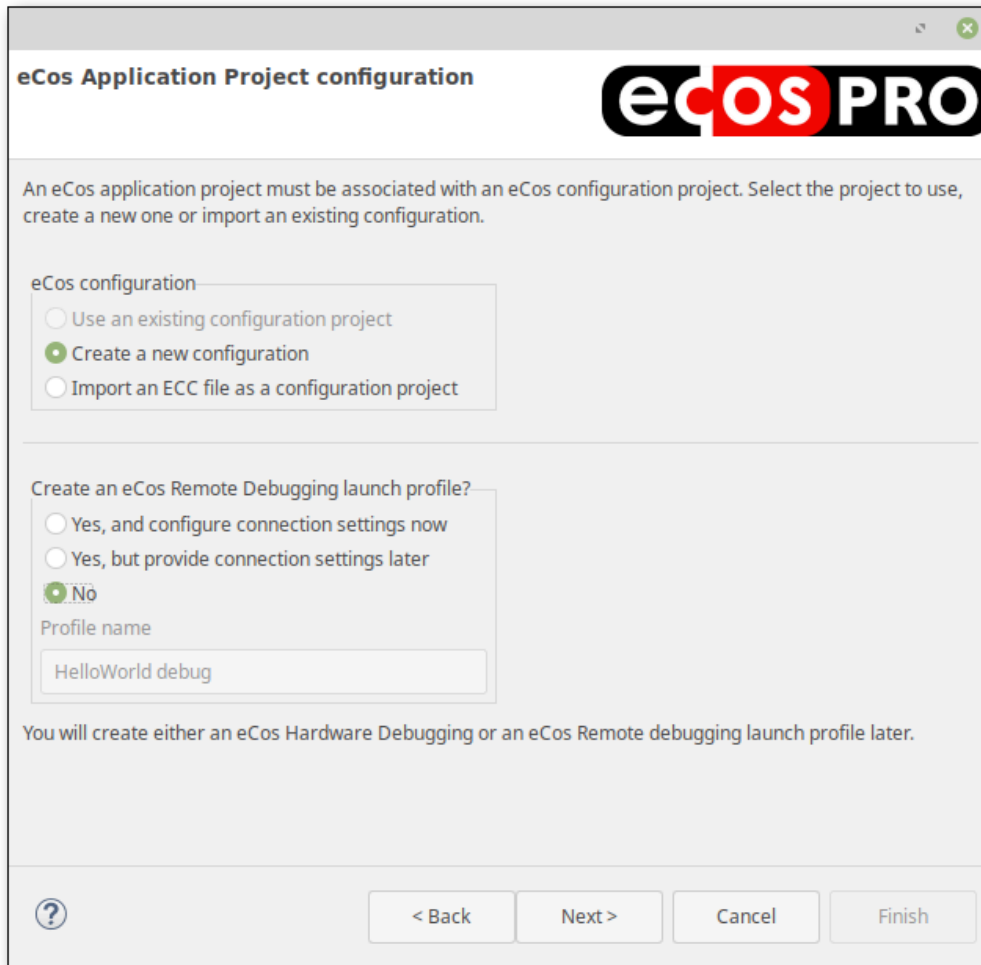
If you already have an eCos configuration for your target hardware in the form of an ECC file, you can instead import it as a configuration project. If you have already created or imported a configuration project, then the option to use an existing configuration project will no longer be greyed out, thereby allowing you to select the configuration project.

10. In order to create the ability to conveniently run this example on our target hardware we need to create a launch configuration. Launch configurations are used in Eclipse to describe how to run or debug the application, and for eCos applications, the most important aspect is that it provides information on how to connect to the target hardware (serial parameters, TCP, JTAG, …). In some instances, launch configurations are dependent on the eCos configuration so for now in the `Create an eCos Remote Debugging launch profile?` section select `No`. You will create a launch configuration later in this example. `No` is required if you are using *eCos Hardware Debugging*.

If you are familiar with your hardware and you know that you connect to your hardware through either a serial port or a TCP/IP port and know your target's IP address or DNS name, you may select `Yes, and configure connection settings now`. You must then provide the settings for Eclipse to communicate to your hardware. If you do not know the connection settings now, select `Yes, but provide configure connection settings later`. By choosing to provide connection settings later, we will be prompted for the settings when we first use the launch configuration to run or debug the application.

Notes

- Connection settings do not apply to projects for the Linux *synthetic* target; the prompt does not appear in that case.

- You can only provide options for *eCos Remote Debugging* if you choose to provide configuration settings now or later. If you intend to debug your application with a hardware debugger, you must select **No** so you can create the hardware debug configuration settings later.

Tip

- eCos Hardware Debugging launch configurations as well as eCos Remote Debugging launch configurations may be created by the `Run` → `Debug Configurations` menu option in the *"Debug Configurations"* dialog.

- The `eCos` → `Create Remote Debugging launch configuration` menu item can be used to create additional Remote Debugging launch configurations. There is currently no shortcut for creating *eCos Hardware Debugging* Debug Configurations

11. Press **Next**. After a few seconds, the wizard starts presenting the pages required to create an eCos configuration project as illustrated in Figure 3.9, "New eCos Configuration project".

Figure 3.9. New eCos Configuration project
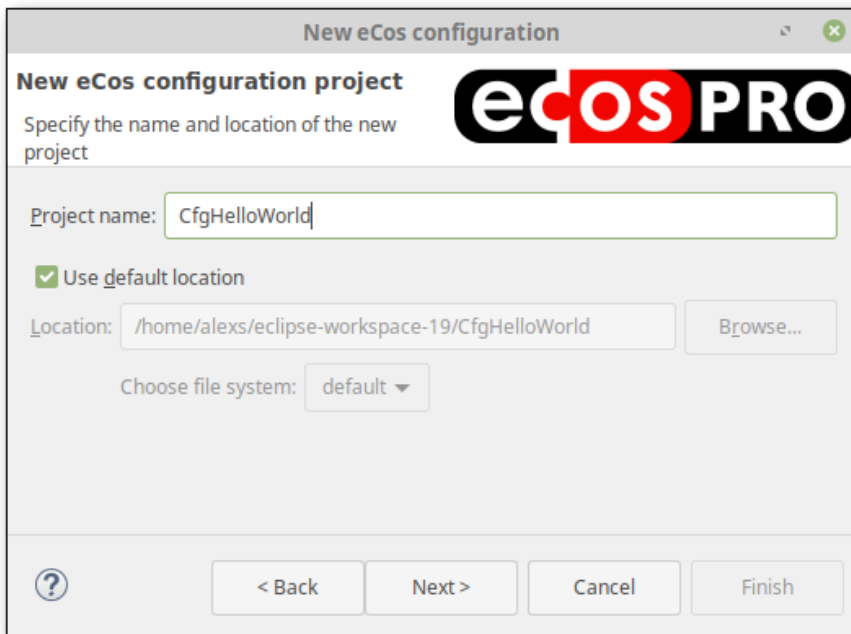
12. Specify a name for the configuration project. **CfgHelloWorld** was chosen in this example. Again, you may set a non-default location if required. Press **Next**.

13. If you have multiple eCosPro installations, the *Profile Selection* page appears as illustrated in Figure 3.10, "eCos profile selection". The drop-down box contains the list of detected profiles associated with eCosPro releases installed on your system. You can find out more information about profiles in the eCos and eCosPro User Guide. Select the appropriate release for your hardware and press **Next**.

Figure 3.10. eCos profile selection
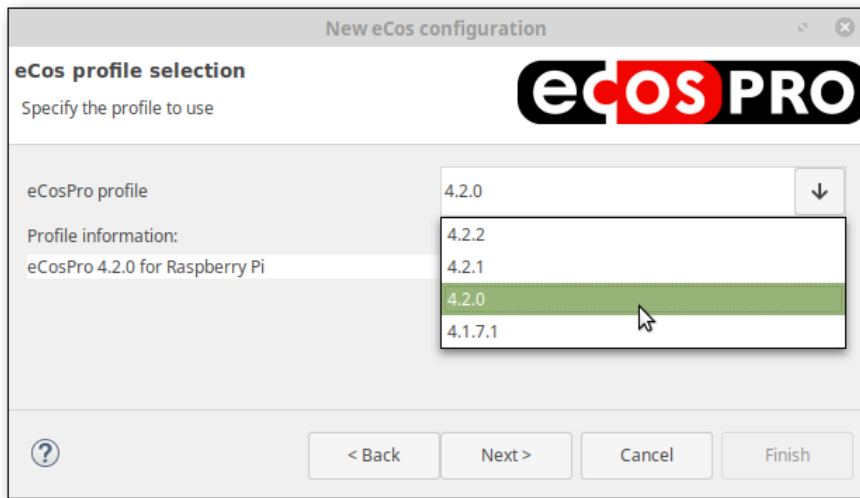
> ## Tip
>
> eCosPro releases prior to 3.1.0 will not appear in this list. If you have an older release installed which set the ECOS_REPOSITORY environment variable, that setting is used as a further profile. However if you do so, you must ensure you have also set your PATH environment variable to include the eCos host tools, and the relevant GNU toolchain for your hardware.

14. The *Target and Template Selection* page appears as illustrated in Figure 3.11, "eCos target and template selection". The default hardware target and template for the selected profile are preselected as defaults. In this example, we are using the stm3220g_eval, but you should select the relevant target for your hardware.

Figure 3.11. eCos target and template selection
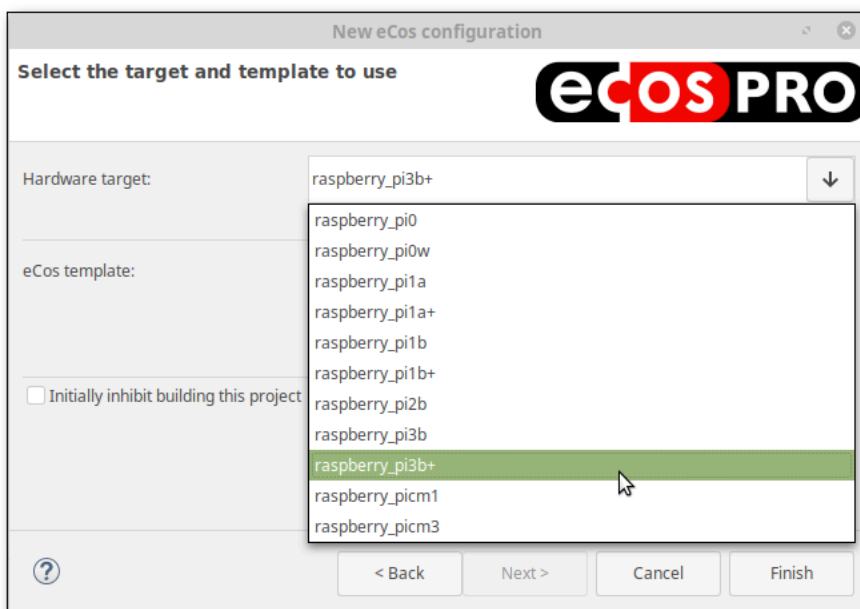
15. Note the option entitled *Initially inhibit building this project* in Figure 3.11, "eCos target and template selection". When checked, this option prevents the configuration from being built. This is useful if you want to make further changes to your new eCos configuration before building it. For the purposes of this example, keep the option unchecked.

16. Finally, press *Finish*. The two projects are created in your Eclipse workspace, with a project reference created within the application project to the eCos configuration project.

17. At this point, if you are running Eclipse for the first time, you will be prompted whether you wish to switch to the C/C++ perspective and whether you wish to switch automatically to this perspective in the future as illustrated in Figure 3.12, "Open Associated Perspective".

Figure 3.12. Open Associated Perspective
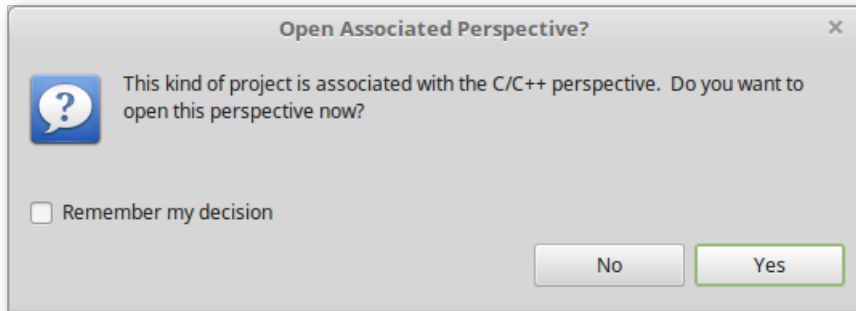
For further information on perspectives, see *Workbench User Guide* → **Concepts** → *Perspectives*.

Optionally check **Remember my decision** and select *Yes*.

> ### Note
>
> If you do not see this dialog because you have already run Eclipse and remembered this decision, you will not be prompted. Switch to the C/C++ perspective manually now if you have not already switched to that perspective.

18. At this point you should now be in the C/C++ perspective illustrated in Figure 3.13, "Build Progress". You may return to this perspective at any time by selecting the *C/C++* perspective button called out by the speach bubble within that figure.

As we left the **inhibit** option unchecked, Eclipse automatically begins to build the eCos configuration in the background. Progress reporting appears in the bottom-right corner of the workbench window. You can click on the progress bar circled in red in the figure to open up the indicated *Progress* view which has more detailed information, or even watch the build output on the *Console* view by selecting the *Tab* circled in blue.

Figure 3.13. Build Progress
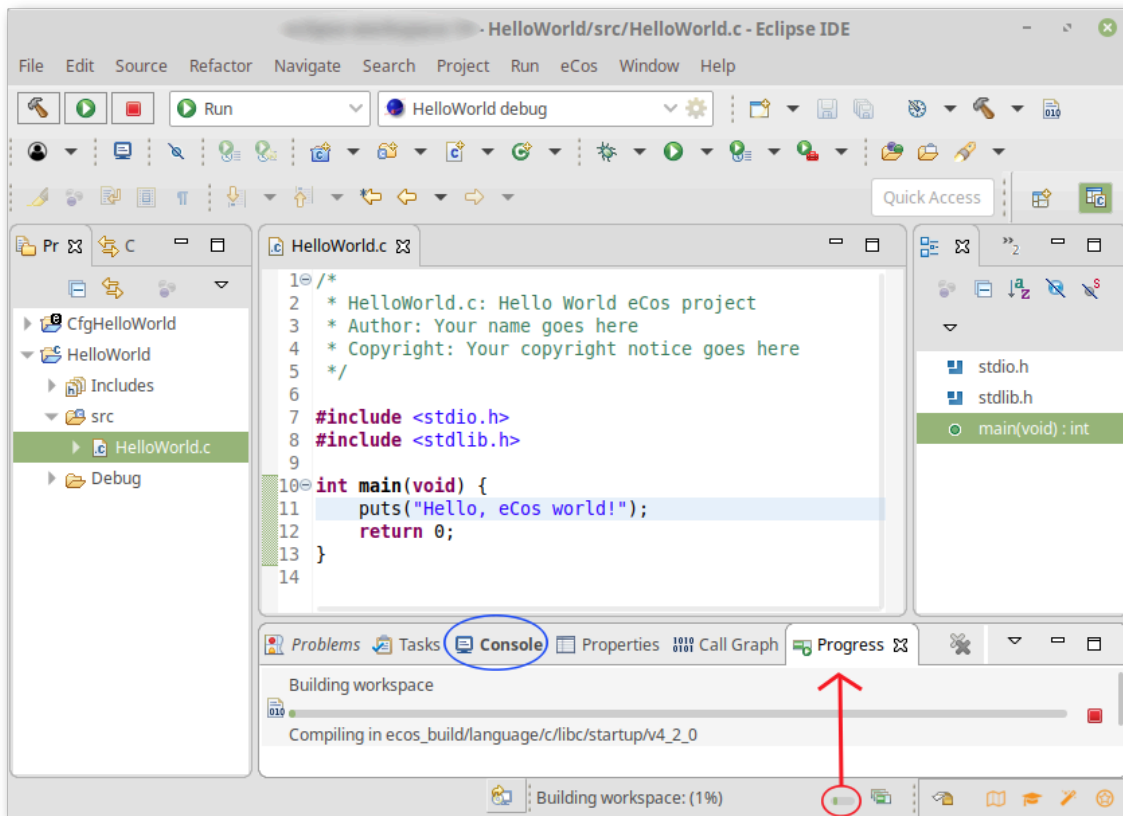
![Note]

Note

You may notice some initial errors appearing in the *Console* and *Problems* views, if you have them open. These notifications are benign and will clear when the configuration project has been built.

19. To manually build the application project select **Project** → *Build All* (**Ctrl+B**), as illustrated in Figure 3.14, "Build Project Menu".

Figure 3.14. Build Project Menu
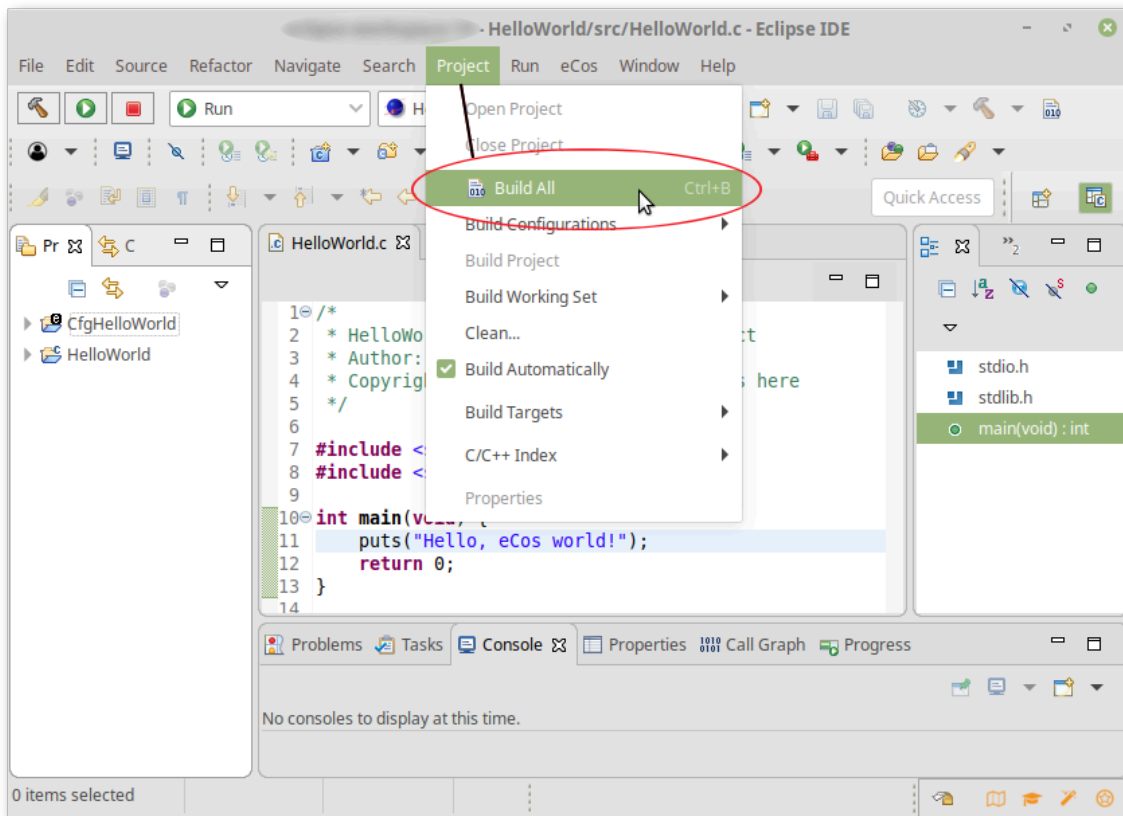
Alternatively right-click on the application project within the *Project Explorer* window and select `Build Project` as illustrated in Figure 3.15, "Application Build Project Menu":

Figure 3.15. Application Build Project Menu
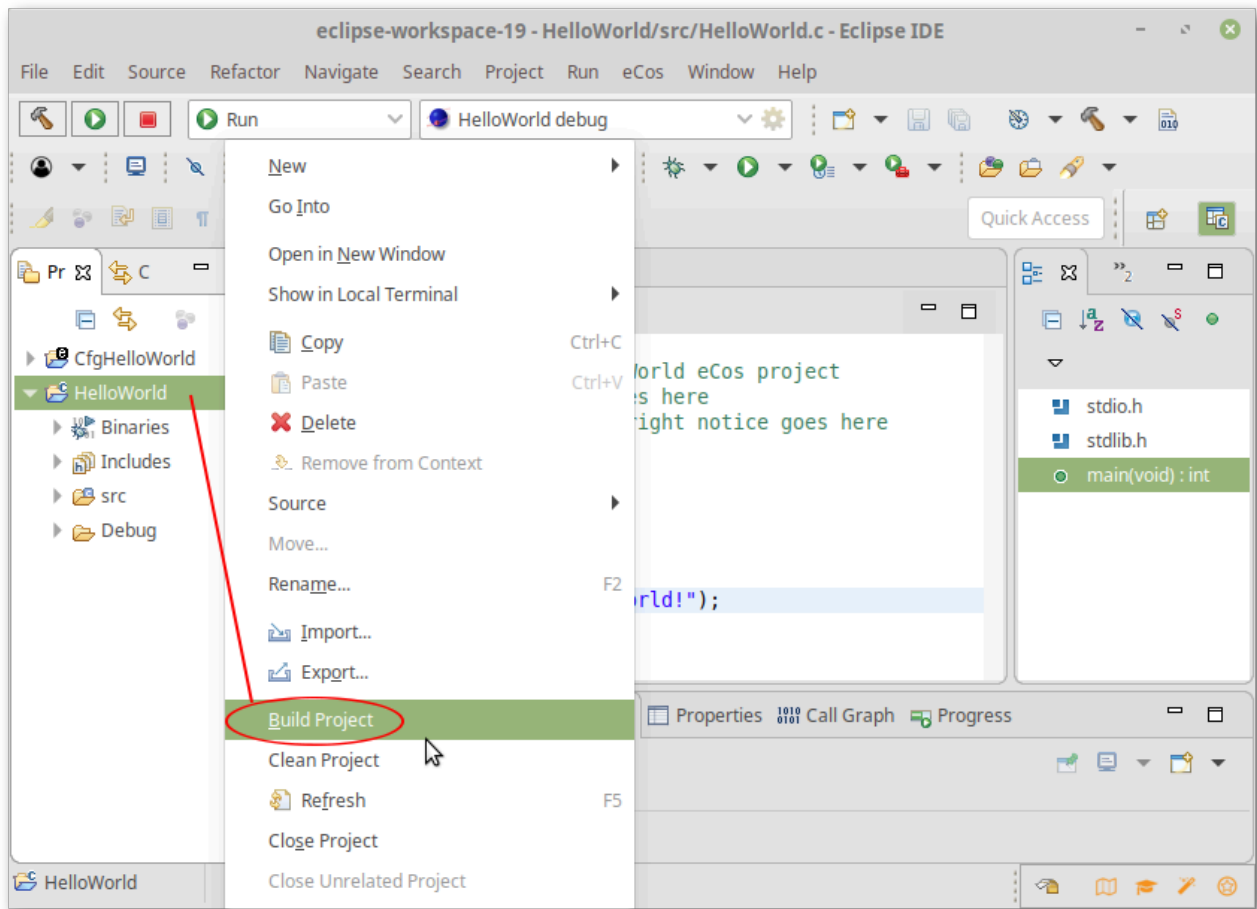
20. Once started, the build progress may be tracked as illustrated in Figure 3.13, "Build Progress". Wait a while for the build to complete; if the configuration build was still running, Eclipse schedules the application project build to happen next.

    The Eclipse CDT project builder is fully integrated with the Eclipse background builder so that both the application project as well as the associated eCos configuration project will be built in the background by default. This option may be deselected through the following steps: *Window* → *Preferences* and in the resultant popup: *C/C++* → **Build** → *Build Targets* deselect the *Build target in the background* option.

21. Your application project has now built. Browsing it in the Project Explorer view you will notice the source directory, Debug (containing builds in the Eclipse "Debug" build mode, which is the default) and autogenerated lists of Binaries and Includes. You can also browse the configuration project's build and install directories, the configuration file ecos.ecc and (via an Eclipse filesystem link) the source repository. All the usual Eclipse CDT functionality, such as the *Content Assist* feature which provides a list of available functions (*Edit* → *Content Assist* (**Ctrl+Space**)), function prototyping and descriptions as illustrated in Figure 3.16, "Project Explorer and Context Editor".

Figure 3.16. Project Explorer and Context Editor
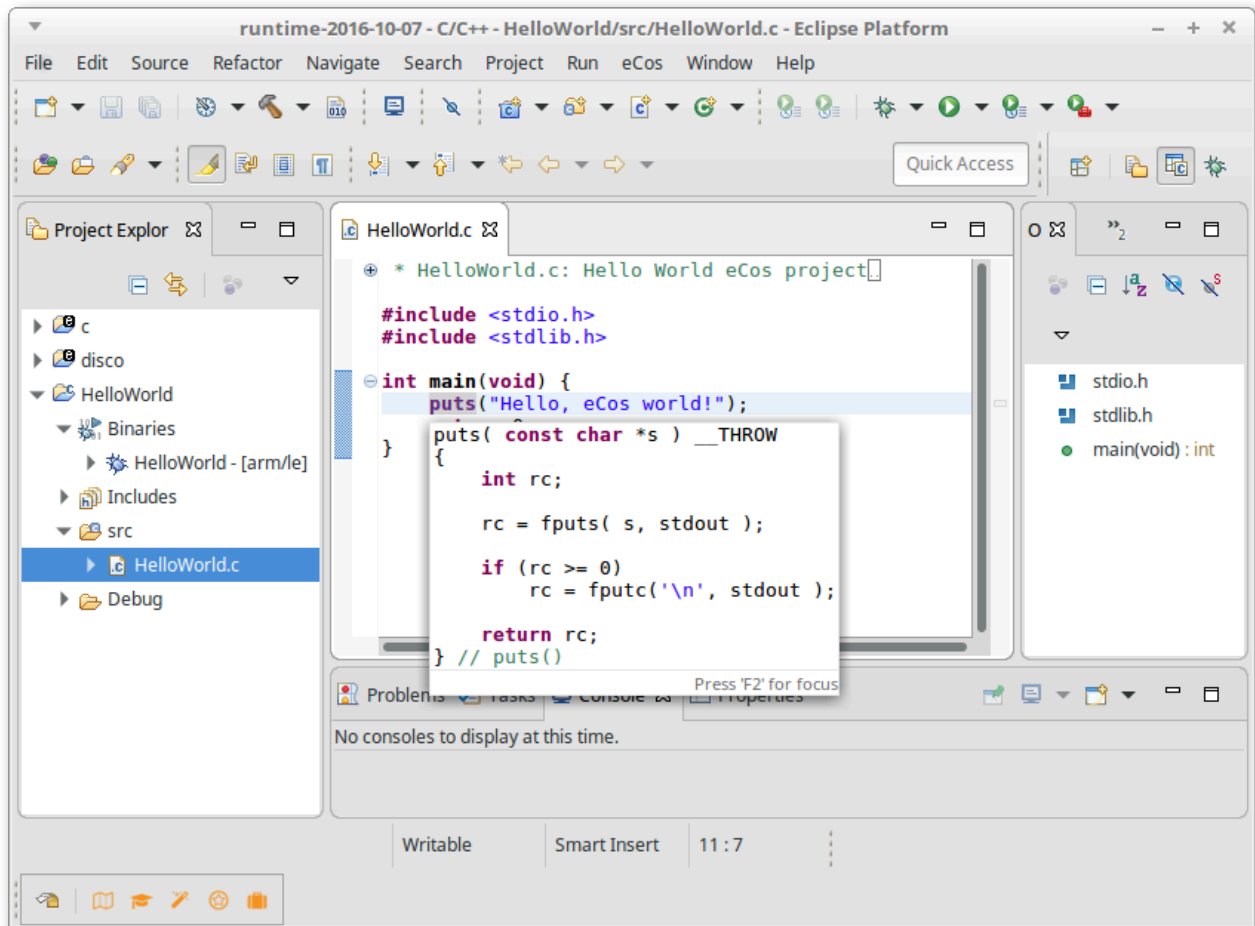
The next chapter "Debugging eCos applications" provides a walkthrough showing how to download and debug the application you have just built.

Chapter 4. Debugging eCos applications

eCos Launch Configurations

In the normal development cycle of eCos application development, once an application executable has been built, the application must be launched on the target hardware from where it is executed and debugged. This involves the creation of an *eCos launch configuration* which specifies to Eclipse and the eCosPro Eclipse/CDT plug-in how the binary executable can be placed on the target hardware and executed.

> **Tips**
>
> 1. When developing and debugging an eCos application, you are advised to disable optimizations and to enable debug assertions as described in Disable optimizations.
>
> 2. If you need to follow execution of a thread into the eCos kernel or any eCos/eCosPro packages or libraries, you are advised to link your application against an eCos configuration project that has been built with optimizations turned off. See Disable optimizations for further details.
>
> 3. The eCosPro New Project Wizard enables you to set up a `eCos Remote Debugging` launch configuration when creating the project. The wizard offers to take connection details for your target; if they are not provided at project creation time, you will be prompted for them when you first launch your project.

The eCosPro Eclipse/CDT plugin provides two types of *eCos launch configurations:*

1. eCos Remote Debugging

   The target hardware is running a bootstrap monitor such as RedBoot, or a debug monitor such as *GDB stubs*, which permit the remote download and debugging of the application on the target hardware through either a serial port or over a network using TCP/IP.

2. eCos Hardware Debugging

   The target hardware is connected to a hardware debugger, such as the Ronetix PEEDI or the *OpenOCD* Open On-Chip Debugger.

Method 1 for creating eCos Launch Configurations

To create a launch configuration, right mouse click on the application project within Eclipse's project explorer, select either `Debug As`→`eCos C/C++ Application (remote)` or `Debug As`→`eCos C/C++ Application (hardware)`, depending on whether you will use either eCos Remote Debugging or eCos Hardware Debugging respectively, as illustrated in Figure 4.1, "Creating eCos launch configurations".

- If no launch configurations of the corresponding type exist, the corresponding dialog illustrated in either Figure 4.5, "eCos Remote Debugging dialog" or Figure 4.6, "eCos Hardware Debugging dialog" will appear in order to allow you to create one. If the eCos Application Project contains more than one binary executable, you will also be asked to select the executable to be launched. Once created, the configuration is launched.

- If one launch configuration of the corresponding type already exists, it will be be launched immediately.

- If more than one launch configuration of the corresponding type already exists, you will be prompted to select a launch configuration which will be immediately launched.

Follow the corresponding instructions in either the section called "eCos Remote Debugging" or the section called "eCos Hardware Debugging" to walk-through the remaining process of debugging an example eCos application on your target.

Figure 4.1. Creating eCos launch configurations
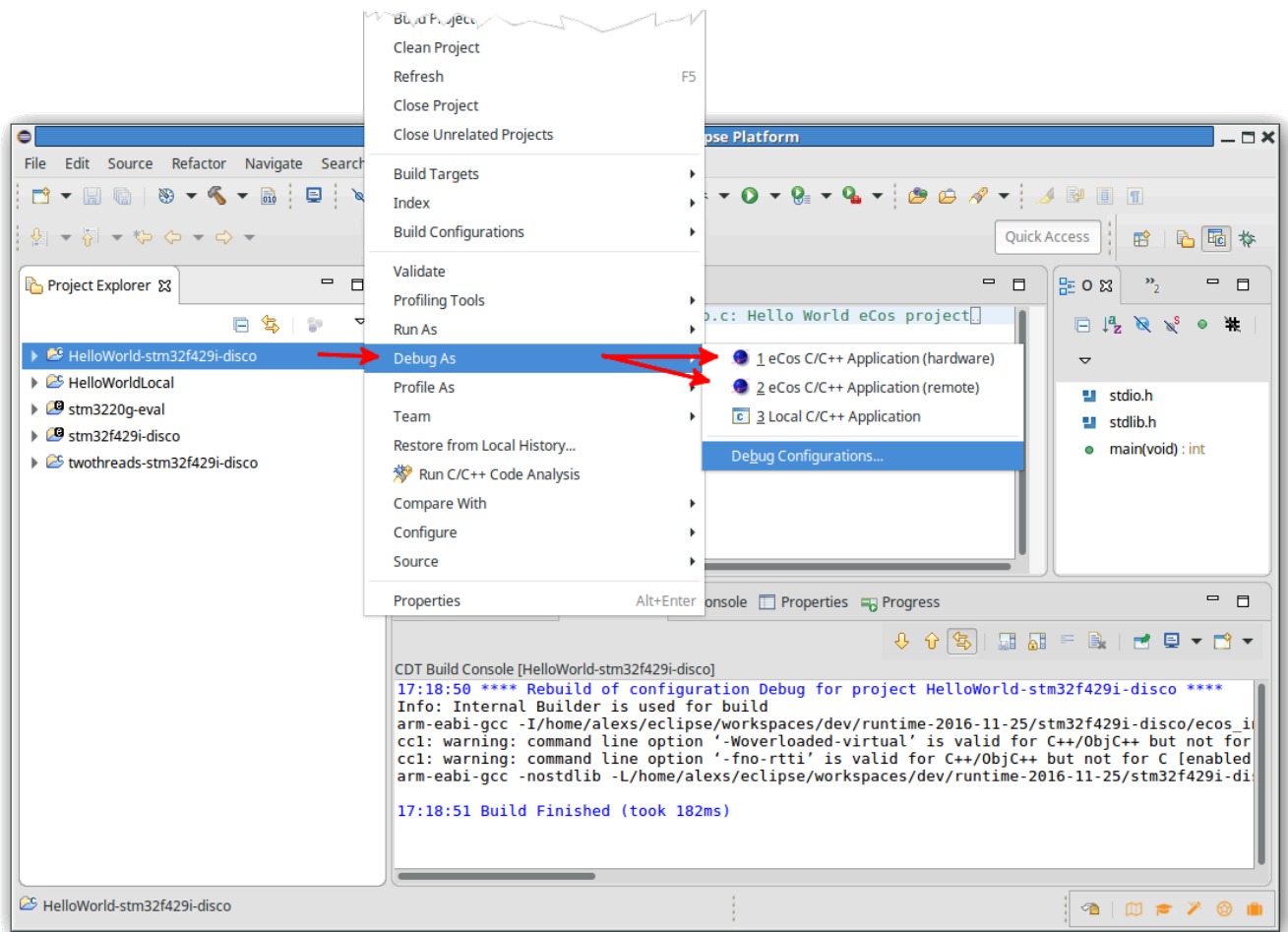
Method 2 for creating eCos Launch Configurations

If within the Project Explorer window you open up the eCos Application Project and right mouse click on the application binary followed by *Debug As* → *Debug Configurations* as illustrated in Figure 4.2, "Selecting eCos launch configuration types".

Figure 4.2. Selecting eCos launch configuration types
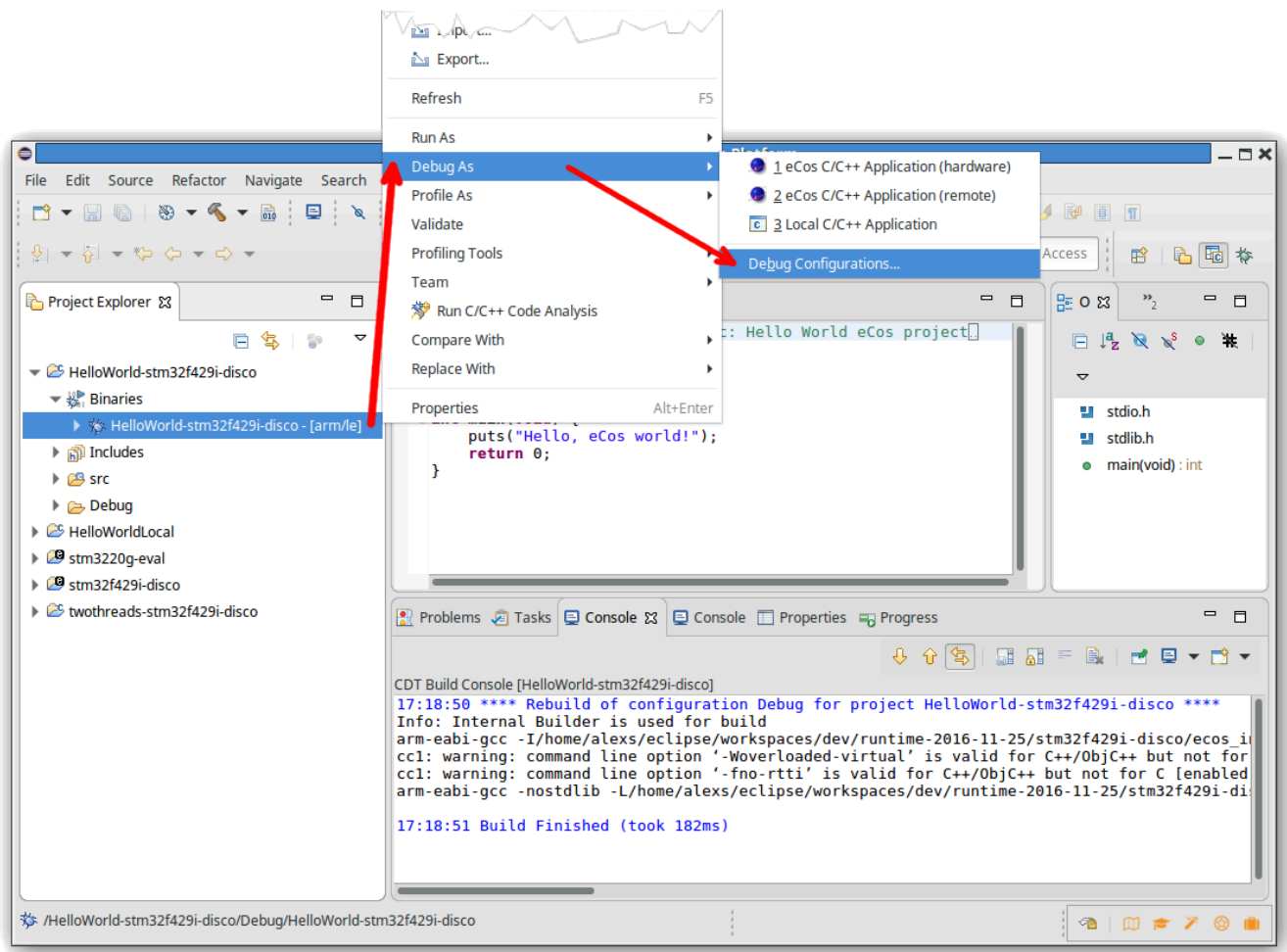
The dialog illustrated Figure 4.3, "eCos launch configurations" should appear.

Figure 4.3. eCos launch configurations
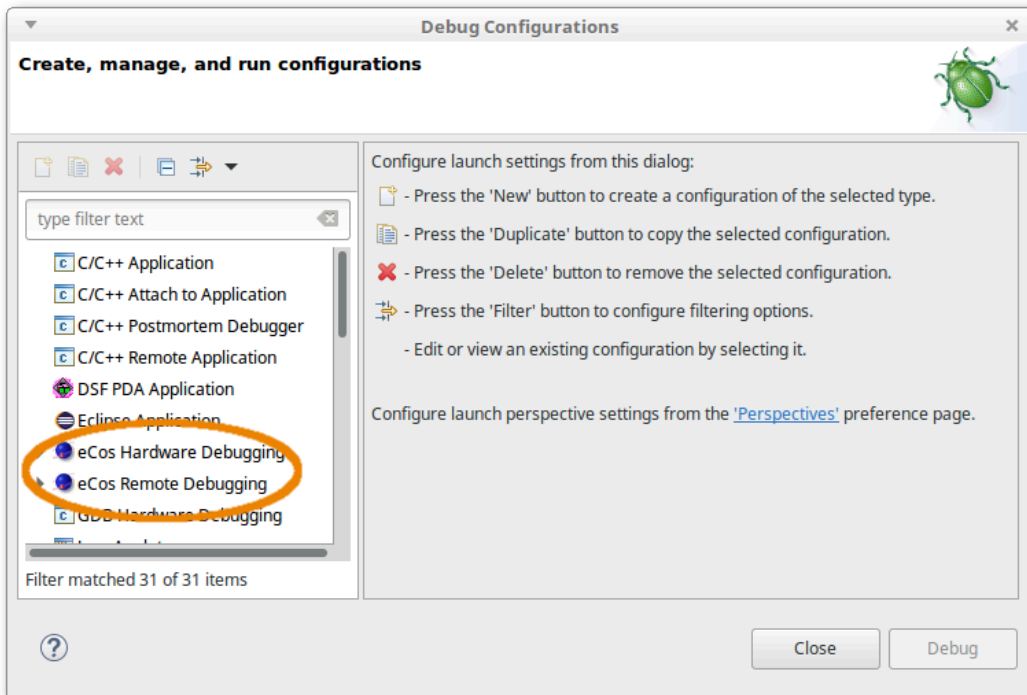
Now either:

1. Double click on the *eCos Remote Debugging* option illustrated in Figure 4.3, "eCos launch configurations". The right-hand side of the dialog will then be populated with the *eCos Remote Debugging* options as illustrated in Figure 4.5, "eCos Remote Debugging dialog"; or

2. Double click on the *eCos Hardware Debugging* option illustrated in Figure 4.3, "eCos launch configurations". The right-hand side of the dialog will be populated with the *eCos Hardware Debugging* options as illustrated in Figure 4.6, "eCos Hardware Debugging dialog".

Follow the instructions in either the section called "eCos Remote Debugging" or the section called "eCos Hardware Debugging", as appropriate, to walk-through the process of debugging an example eCos application on your target.

Note

If you do not select or have an eCos binary highlighted when you create any eCos launch configuration, you will get an error of the form illustrated in Example 4.1, "Executable Not Highlighted". This is because the context of the Project Explorer provides the project or application for which a launch configuration is to be created and it is only possible to create eCos launch configurations for eCos applications.

Example 4.1. Executable Not Highlighted

```
An error has occured. See error log for more details.
Project selected is not an Ecos Application project.
```

Method 3 for creating eCos Launch Configurations

Open the eCos Application's properties dialog window by highlighting the eCos application project within the *Project Explorer* window of the C/C++ perspective through the menu option *File → Properties* (**Alt+Enter**) or by *right-click → Properties*, and select *Run/Debug Settings*. Existing launches, if any, will appear in the left-hand panel. To create a new launch, select *New* and the dialog illustrated in Figure 4.4, "Select Configuration Type" will appear.

Figure 4.4. Select Configuration Type
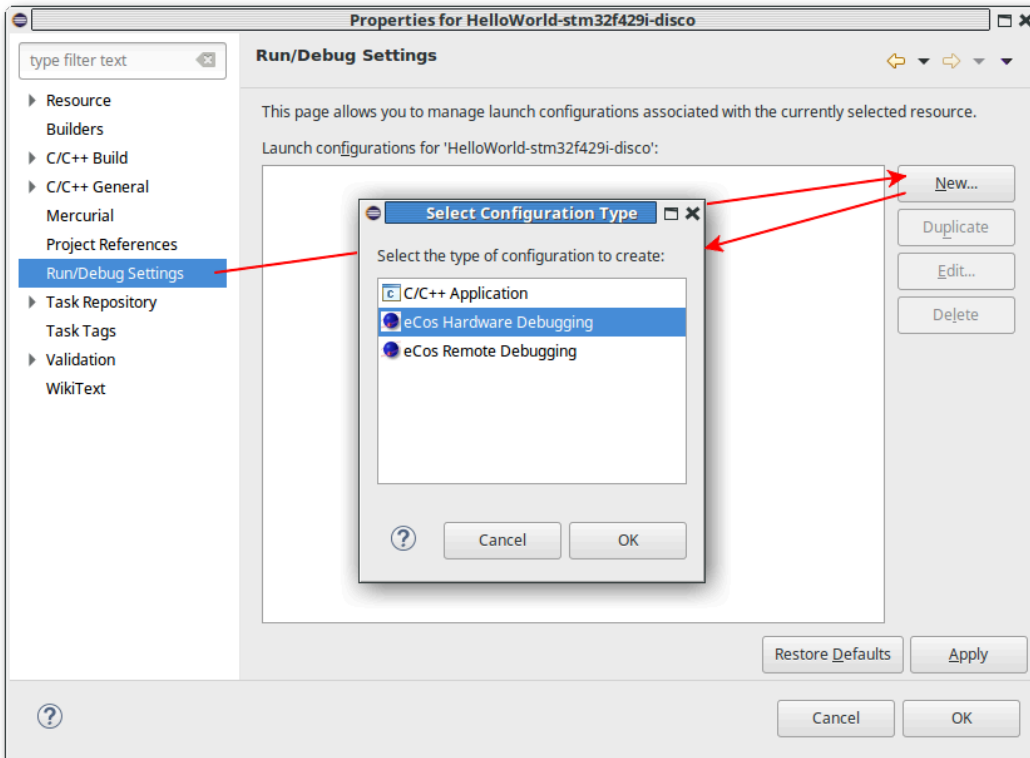
Select either *eCos Remote Debugging* or *eCos Hardware Debugging* followed by *OK* and follow the instructions provided in the corresponding section.

eCos Remote Debugging

eCos Remote Debugging requires the use of a bootstrap debugger such as RedBoot or *GDB stubs* which must be installed on your hardware before you can commence eCos Remote Debugging. Please refer to the target hardware documentation accompanying your eCosPro release and indexed from within the release documentation index page for detailed instructions on how to install either RedBoot or *GDB stubs* on your hardware.

eCos Remote Debugging launch configuration

These steps follow on from the Project Creation walkthrough, showing how one might go on to use an Eclipse debug session to launch a sample *Hello World* application on a target running a bootstrap monitor such as RedBoot.

At this point you should see a dialog similar to that illustrated in Figure 4.5, "eCos Remote Debugging dialog" below, although the exact appearance of the dialog may differ according to how you initiated the creation of the launch configuration. For example, the left-hand panel may not be present, or the buttons *Revert*, *Apply*, *Close* or *Debug* may either not all be present or be named differently.

Figure 4.5. eCos Remote Debugging dialog

![Debug Configurations dialog]
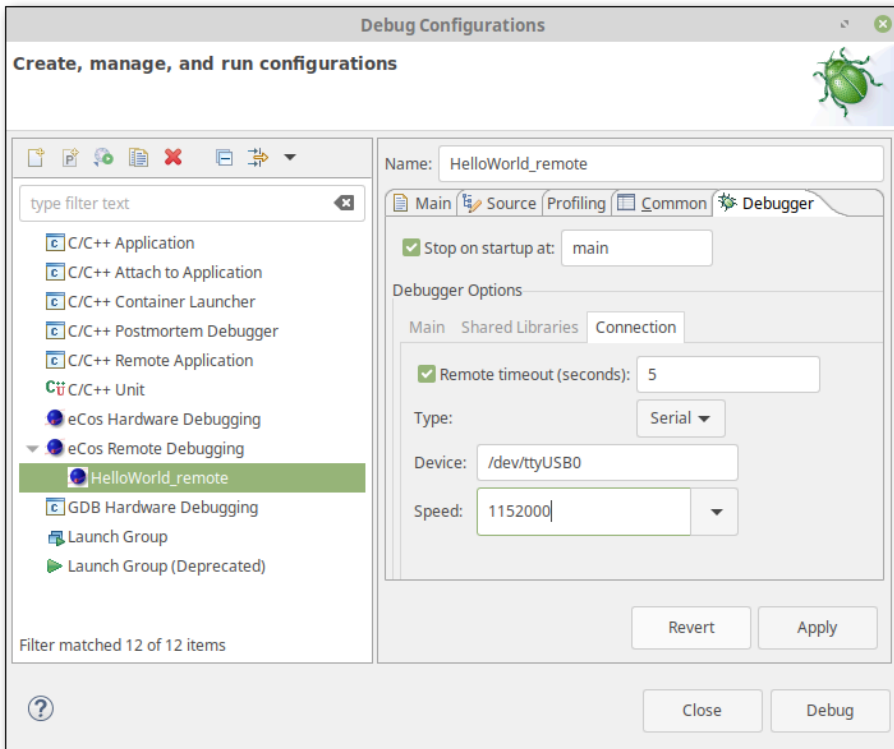
Notes:

- The *Remote timeout* field is active when checked with a non-zero integer value in the corresponding field. This field allows the developer to set the communication timeout between *gdb* and the target device, in place of the *gdb* default (normally 2 seconds). This field is useful in cases where, for example, the target device is physically remote and accessed through a where remote communication may be slow or delayed (e.g. satellite).

- Linux users must ensure they have read/write permissions to the serial port, if the connection to the target is a serial connection. Often this may be achieved by adding the user to the `dialup` group, or by creating a *udev* rule which modifies either the default user or group ownership, or the device permissions, to enable the serial device to be accessed by the user.

- The *Speed* field offers both a drop-down of common values, and may be modified to a custom value. No checks of the validity of the value are performed and the actual speed used by the host's device drivers is dependant on the speeds the host's device drivers are capable of being set to. In most cases, if an exact match is not found the closest baud rate the host's device driver is capable of is used.

- If `main()` is not the entry point for your own application, it is best to disable that option. If no user- supplied `main()` is provided, a default eCos one will be used, which will cause your application to be stopped by a breakpoint when it runs, even though that will have no significance to your application. If you are not using `main()`, consider removing the `CYGPKG_LIBC_STARTUP` package from your eCos configuration.

1. Choose an appropriate name if the default name is not suitable, select the *Debugger* tab and choose the connection type, modifying the settings as appropriate in order to access the target hardware. For example:

   - *If you are connecting via ethernet to RedBoot on the target:*

     Select `TCP` for the *Type* and specify the *IP address or host name* and *Port number* of RedBoot. The default Port number of RedBoot is `9000`.

   - *If you are connecting via serial to RedBoot or GDB stubs on the target:*

Select **Serial** for the *Type* and specify the *Device* and *Speed* (baud rate) used by RedBoot or *GDB stubs*. For Windows hosts, the device may be **COMn**, while for Linux platforms this should be the full path of your serial device (e.g. **/dev/ttyUSBn**).

2. Press *Apply* to save the configuration followed by either *Debug* or *OK*.

3. If you came to this dialog through the *Run/Debug Settings* option of the *properties* dialog for the eCos application project as described in the section called "Method 3 for creating eCos Launch Configurations", you will need to use one of the other methods described in the section called "Method 1 for creating eCos Launch Configurations" or the section called "Method 2 for creating eCos Launch Configurations" or to deploy the launch configuration as described in those sections.

4. Continue the debug walkthrough as described in the section called "Running and Debugging the application".

> **Note**
>
> Launching an *eCos Remote Debugging* session to a remote target, particularly when connected over a serial line, can be prone to spurious errors. Be sure to use a suitable *Remote timeout* and allow for sufficiently long time between resetting your target board, allowing debug monitors such as RedBoot to bootstrap the target fully, and starting a debug session from Eclipse.

eCos Hardware Debugging

To interactively debug an application built to execute from flash memory, or from RAM without a bootstrap monitor such as RedBoot or GDB stubs, you must normally use a hardware debugger such as OpenOCD or the Ronetix PEEDI (PEEDI). Most hardware debuggers present a GDB protocol interface and so may be used in the same way as RedBoot or GDB stubs on a target board.

eCos Hardware Debugging differences

The main differences between *eCos Remote Debugging* and *eCos Hardware Debugging* are:

- Hardware debuggers generally do not provide thread-level debugging information or require their own custom configuration to determine operating system specific thread information. Unless your hardware debugger is suitably configured, you will not be able to browse the list of active threads - your application will appear to consist of a single thread no matter how many it has and you cannot make a breakpoint thread-specific.

- Console output and output to the /dev/haldiag or /dev/ttydiag devices may be configured within an eCos configuration to be displayed in the Eclipse console, regardless whether the target hardware has any external I/O capability. This is a very useful facility and may be used by *all* targets running eCosPro applications which are launched using the *eCos Hardware Debugging* launcher. An example of how to create such an eCos configuration is described in the section called "eCos Configuration for eCos Hardware Debugging".

> **Warning**
>
> This form of output requires the use of an additional breakpoint which may not always be available. If no additional breakpoints are available, developers will be restricted to using the target's external devices (if any).

Should the target hardware have an external device to which output to the /dev/haldiag or /dev/ttydiag devices is sent, for example an RS232 serial device, this output may be captured and displayed in the Eclipse console. To acheive this, set the *Console connection* value within the *Debugger* tab to either **Serial** (configuring the host's serial port and speed as appropriate) or **TCP** (configuring the host name and TCP port number as appropriate) to read diagnostics output from either source. Leave the setting as **GDB** should eCos diagnostic output be configured to come through GDB.

> **Note**
>
> These configuration options are only available through eCosPro enhancements, the eCosPro CDT plug-in and eCosPro GNU Toolchains.

- Applications executing from flash memory cannot use ordinary ("soft") breakpoints; they can only use hardware breakpoints.

- Hardware breakpoints are set in essentially the same way as soft breakpoints, that is by right-clicking in the ruler of a source code window. To set a hardware breakpoint, first select *C/C++ Hardware Breakpoints* from the *Breakpoint Type* submenu, then select *Toggle Breakpoint* as before.

> ### Caution
>
> Most targets are limited in the number of hardware breakpoints they support, and such limits are not automatically reported to gdb or to Eclipse. If you attempt set too many, your debug session is likely to become confused. If you know the limit for your target, you can provide a limit in a *GDB command file* (see the section called "GDB command files") with the command shown in Example 4.2, "Set Hardware Breakpoint Limit".
>
> #### Example 4.2. Set Hardware Breakpoint Limit
>
> ```
> set remote hardware-watchpoint-limit N
> ```

> ### Tip
>
> If you run out of hardware breakpoints unexpectedly, look for and disable the option to *Automatically stop on main at startup* in the debugger config panel.

- Some hardware debuggers provide settings in their configuration files which allow developers to map requests for software breakpoints into hardware breakpoints. If you have trouble setting the type of breakpoint you expected, be sure to also check the settings in the hardware debugger's configuration file. For example, with the Ronetix PEEDI, it is configured with the CORE0_BREAKMODE directive which may be set to either soft or hard.

- Some hardware debuggers, like *OpenOCD*, may require that you reset and halt the target before downloading the executable to the target. Most recent eCosPro releases are provided with OpenOCD configuration files that result in OpenOCD performing these operations immediately on accepting a remote connection from *GDB*. However, if you are using a different hardware debugger and either your hardware debugger or your target require these operations, you must configure these options in the *eCos Hardware Debugging* launch configuration.

eCos Configuration for eCos Hardware Debugging

Continuing the steps of Chapter 3, *Quick Start / Walkthrough*, this section describes how to modify the eCos configuration such that output or diagnostics from the application being developed are directed to the Eclipse console through *GDB*.

The latter is a feature unique to eCosPro and eCosCentric-provided *GNU toolchains* which will work across all hardware debuggers, halting the hardware temporarily only when output occurs for the duration the output can be extracted from the target by *GDB*. Less intrusive methods, such as utilising the instrumentation trace macrocell (ITM) provided by certain ARM architectures, are also available. See the section called "Using ARM's Instrumentation Trace Macrocell for output" for an example.

1. If you have open the *Debug Configurations* dialog illustrated in Figure 4.2, "Selecting eCos launch configuration types", select *Close* to close the dialog and return to the C/C++ project explorer perspective (*Window* → *Perspective* → *Open Perspective* → *Other...* → *C/C++* → *OK* or through the *C/C++* perspective button illustrated by the speach bubble in Figure 3.13, "Build Progress").

2. Open up the eCos configuration project associated with the application you wish to debug within the C/C++ project explorer and look for the active eCos configuration file. This will have an .ecc extension and will normally be called ecos.ecc. Open this file for editing with the eCos configuration tool either by double clicking on this file, or by selecting the file and pressing F3, or by right-clicking the file and selecting *Open*.

3. If you have not already turned off compiler optimizations, do that now. See Disable optimizations for further details.

4. Find the CYG_HAL_STARTUP macro using the search dialog (see Disable optimizations for an example) and modify the startup type to one that is suitable for your target and hardware debugger (for example, *JTAG*), within the configuration

tree accepting the changes by pressing **Return**. Accept any conflict resultions which the eCos *configuration tool* may present to you.

> ### Note
>
> Your target may contain strict rules that inhibit the CYGFUN_HAL_DIAG_VIA_GDB_FILEIO macro from being enabled (see next step) unless startup type with an association to *GDB* is chosen.

5. Find the CYGFUN_HAL_DIAG_VIA_GDB_FILEIO macro using the search dialog and enable it within the configuration tree. This will likely result in a conflict which, when resolved, will cause the CYGFUN_HAL_GDB_FILEIO to be enabled. This is the correct resolution and expected behaviour. On some target platforms (for example the CortexM®) there may be additional hardware-specific resolutions (for example, CYGHWR_HAL_CORTEXM_DIAGNOSTICS_INTERFACE is set to *gdb_hwdebug*). Accept any conflict resultions which the eCos *configuration tool* may present to you.

6. Save your configuration, *File* → *Save* (**Ctrl+S**), and exit the configuration tool, *File* → *Exit* (**Alt+X**).

7. Your eCos configuration project will automatically be rebuilt if eCos configuration builds have not been inhibited for the project (*eCos* → *Inhibit configuration build disabled*) as well as your application project if automatic builds have been enabled (*Project* → *Build Automatically enabled*). Ensure both eCos and your application project are rebuilt if either of the above settings are not as indicated.

8. Create the eCos Hardware Debugging launch configuration as described in the section called "eCos Hardware Debugging launch configuration".

eCos Hardware Debugging launch configuration

The following steps continue on from Chapter 3, *Quick Start / Walkthrough* and the section called "eCos Configuration for eCos Hardware Debugging" to launch a sample *Hello World* application on a target using a hardware debugger.

eCos Hardware Debugging using OpenOCD

In this example we show you how to use *OpenOCD* to debug your application, but most of this walkthrough can be applied to other hardware debuggers as well. See the section called "eCos Hardware Debugging using the Ronetix PEEDI" for an example of how to use the Ronetix PEEDI.

1. Right mouse click on the application binary within Eclipse's project explorer, or select *Debug As* followed by *Debug Configurations* as illustrated in Figure 4.2, "Selecting eCos launch configuration types", to return to the the dialog window illustrated in Figure 4.3, "eCos launch configurations" and described in the beginning of this chapter.

2. Once the *"Debug Configurations"* dialog illustrated has been opened, double click on the *eCos Hardware Debugging* option illustrated in Figure 4.3, "eCos launch configurations". The right-hand side of the dialog will be populated with the *eCos Hardware Debugging* options as illustrated in Figure 4.6, "eCos Hardware Debugging dialog". Choose an appropriate name if the default name is not suitable.

3. Choose **OpenOCD** as your *JTAG Device* and ensure that *Use remote target* is checked and that the *GDB Connection String* is set as illustrated in Example 4.3, "GDB Connection to OpenOCD pipe".

Example 4.3. GDB Connection to OpenOCD pipe

```
| openocd -f ${eCosInstallDir}/etc/${openocd_config_filename} -c "gdb_port pipe"
```

If you are using a different hardware debugger, select the *Debugger* tab and choose the connection type, modifying the settings as appropriate in order to access the target hardware.

Figure 4.6. eCos Hardware Debugging dialog

![Notes icon]
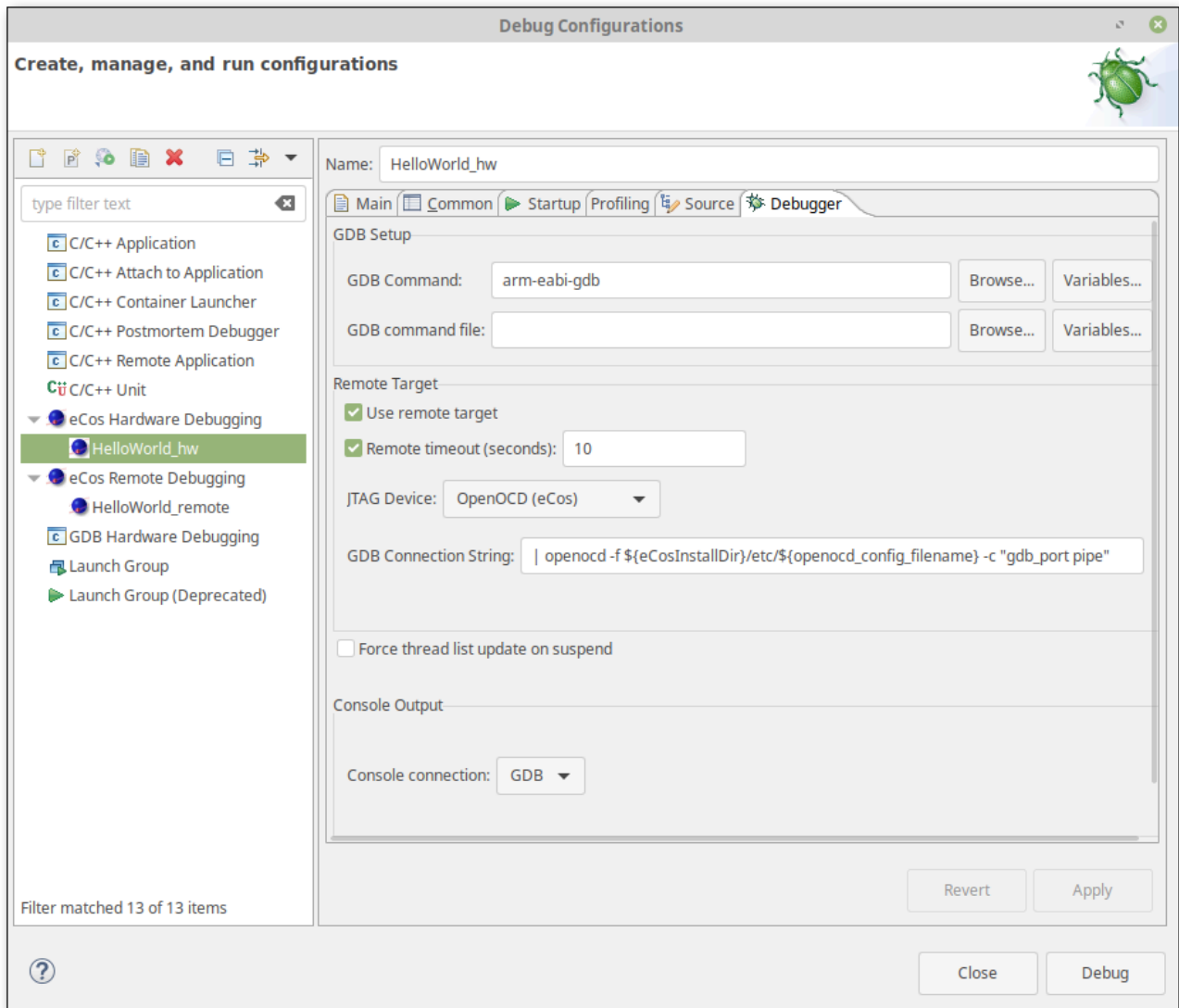
Notes

1. Modern eCosPro distributions have a build system that places usable or example hardware debugger configuration files in the `etc` subdirectory of the eCos installation tree during the build process of the eCos library, or after creating an eCos build tree and executing the command **make etc**. These files are specific to a target, eCosPro configuration and hardware debugger and use common names, permitting pre-defined pipe commands to be used as may be seen in the *OpenOCD* example above.

2. The default pre-defined pipe command for *OpenOCD* uses the `${eCosInstallDir}` and the `${openocd_config_filename}` Eclipse macros to select the appropriate configuration file for the target to provide portabilty and flexibility of the launch configuration.

3. The `Remote timeout` field is active when checked with a non-zero integer value in the corresponding field and has been set by default with a value suitable for starting up *OpenOCD* on slower/older PCs. The *gdb* default remote timeout of 2 seconds can be insufficient in some circumstances.

4. Should the eCos Configuration direct the eCos devices /dev/haldiag or /dev/ttydiag to external devices available on the target hardware, for example an RS232 serial device, to view this output within the Eclipse console set the `Console Connection` value within the `Debugger` tab to either **Serial** (configuring the host's serial port and speed as appropriate)

or **TCP** (configuring the host name and TCP port number as appropriate). Leave the setting as **GDB** should eCos diagnostic output be configured to come through GDB.

The default destination of the output depends on the hardware and specific platform port, as well as the default settings of the eCosPro configuration, but in many cases it will come from an RS232 serial device or ITM port (if available, in which case it may be read from a TCP port through *OpenOCD*).

5. Select *Apply* to save your launch profile followed by *Debug* and continue the debug walkthrough as described in the section called "Running and Debugging the application".

Running and Debugging the application

1. When launching a debug session for the first time, you will be prompted whether you wish to switch to the debug perspective as illustrated below in Figure 4.7, "Confirm Perspective Switch dialog".

Figure 4.7. Confirm Perspective Switch dialog
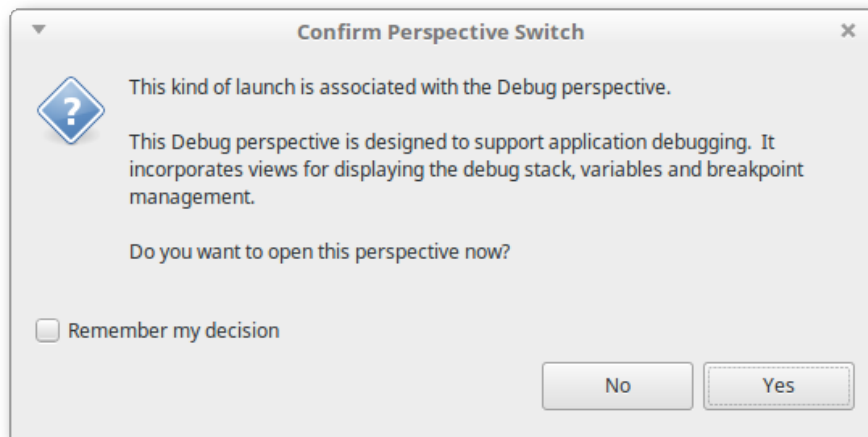
Optionally check *Remember my decision* if you wish to switch to the *Debug Perspective* each time you launch a debug session.

Select *Yes* to continue to have Eclipse open the Debug perspective illustrated in Figure 4.8, "Debug Download Execution".

Figure 4.8. Debug Download Execution
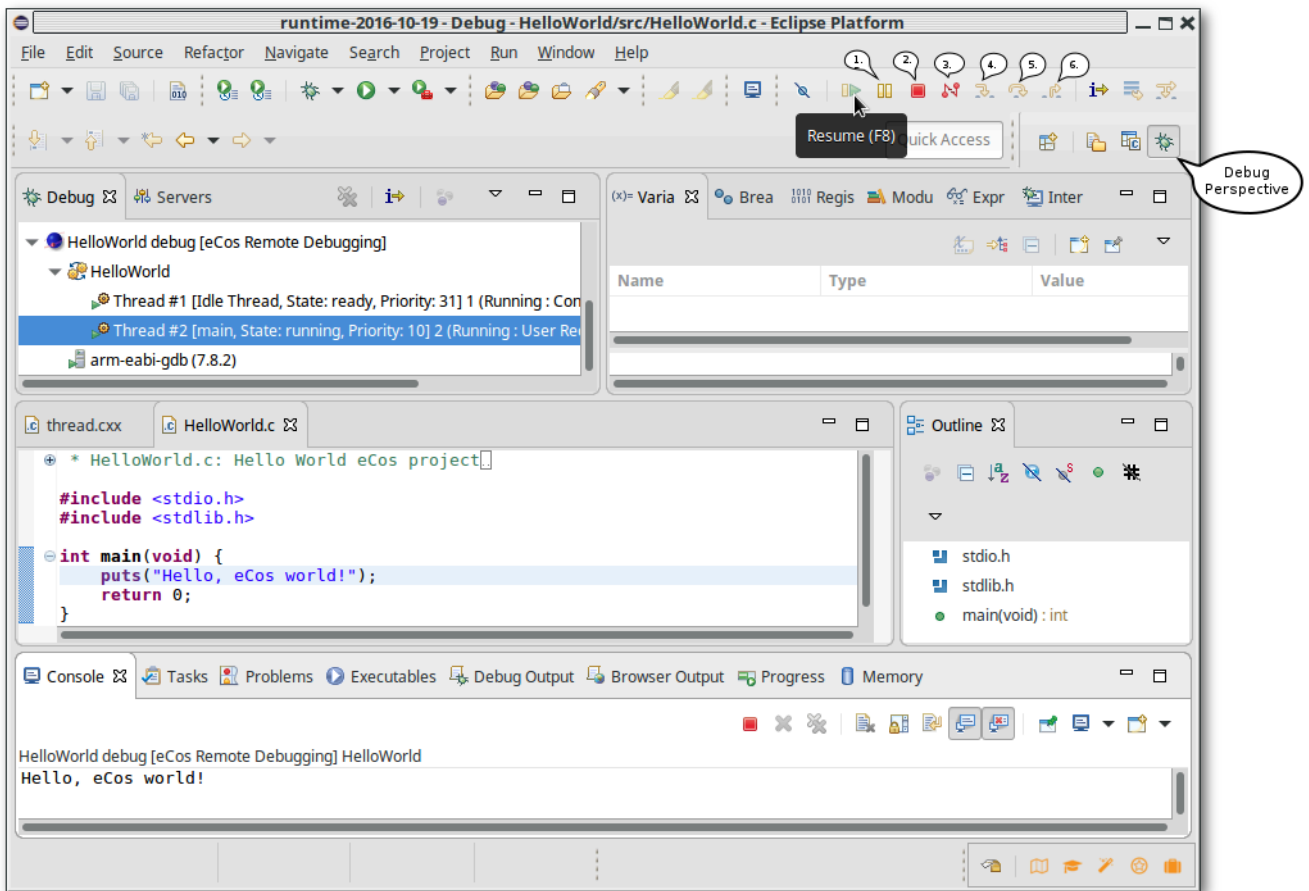
You may not see the exact contents of the previous figure until the download of your application has completed and execution started.

2. If the debug perspective is not opened due to some previous preference selection, open it now. You may open or return to the debug perspective at any time by selecting the `Bug` button called out by the speach bubble in Figure 4.8, "Debug Download Execution", or through the following menu selections: `Window` → `Perspective` → `Open Perspective` → `Other...` → `Debug` → `OK`. These menu options may vary slightly if the debug perspective has previously been selected. For example: `Window` → `Perspective` → `Open Perspective` → `Debug` → `OK`.

3. When a debug session is launched, the debug launcher will download the binary application to the target and commence execution, stopping at main() if you left that option enabled (or an alternative function such as cyg_user_start() if you changed the default function name within the launch configuration). Within the Debug perspective you may view the more detailed *Progress* tab by double-clicking on the Progress bar circled in red in Figure 4.9, "Debug Download Progress".

Figure 4.9. Debug Download Progress
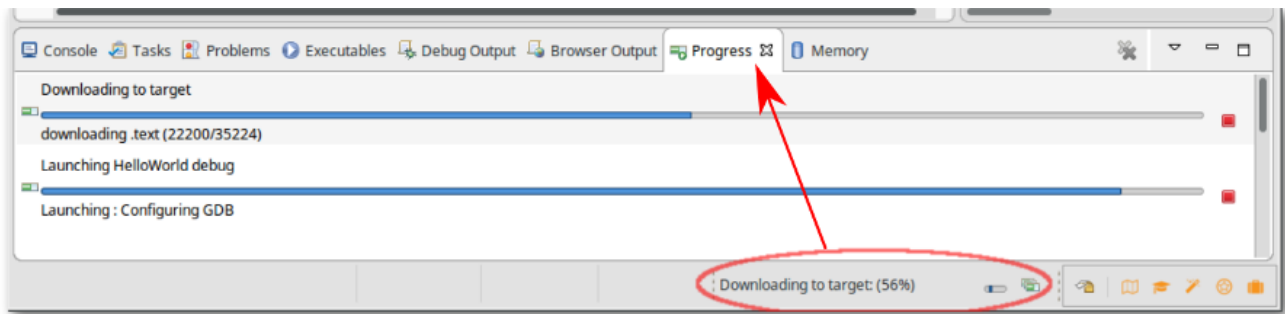
4. If you stopped at `main()`, press the **`Resume`** toolbar button as illustrated by the mouse cursor in Figure 4.8, "Debug Download Execution". Execution continues; on most hardware targets the output, your *Hello eCos World* message, is sent over the board's debug channel and is reported in the *Console* view illustrated in Figure 4.8, "Debug Download Execution". The remaining buttons, highlighted by the speech buttons are as follows:

1. *Pause* current execution.

2. *Stop* the current execution.

3. *Disconnect* from the remote target.

4. *Step into* the function, if any, otherwise execute the current instuction.

5. *Step over* the function, if any, otherwise execute the current instuction.

6. *Return* from the current function call.

5. For debugging your own application, you may set breakpoints and perform other debugger oprtations before resuming or stepping through your application. While your application is running, you may also *pause* the application to inspect variables, threads, the call stacks, memory and so on, or simply *halt* your debug session.

> **Note**
>
> Pausing your application through **`eCos Remote Debugging`** launch configurations is only possible when the debug monitor (such as RedBoot) is active. For example, when your application performs an output operation such a `printf()` through the debug monitor using a *gdb* debug channel.

Console Diagnostics Output

The **`Debug`** perspective includes the *Console tab* window, seen also within the **`C/C++`** perspective, in which a number of different consoles may be viewed including:

- GDB output (*[eCos Remote Debugging] xxx-gdb* or *[eCos Hardware Debugging] xxx-gdb* - any output produced by the underlying **xxx-gdb** command)

- GDB traces (*[eCos Remote Debugging] gdb traces* or *[eCos Hardware Debugging] gdb traces*)

- Application debug output (*Output via zzz*)

By default, the GDB output is displayed by the **`Debug`** perspective in the *Console tab*. This will include information such as the breakpoint information, tracepoints and so on. The eCosPro CDT plug-in provides a diagnostics console in which diagnostics output or output from the application running on the target will appear. You may choose the output which appears in the *Console tab* by pressing the **`Display Selected Console`** button highlighted by the **"A"** callout box and choosing the desired console from the dropdown menu illustrated in Figure 4.10, "Console Diagnostics Output".

Figure 4.10. Console Diagnostics Output
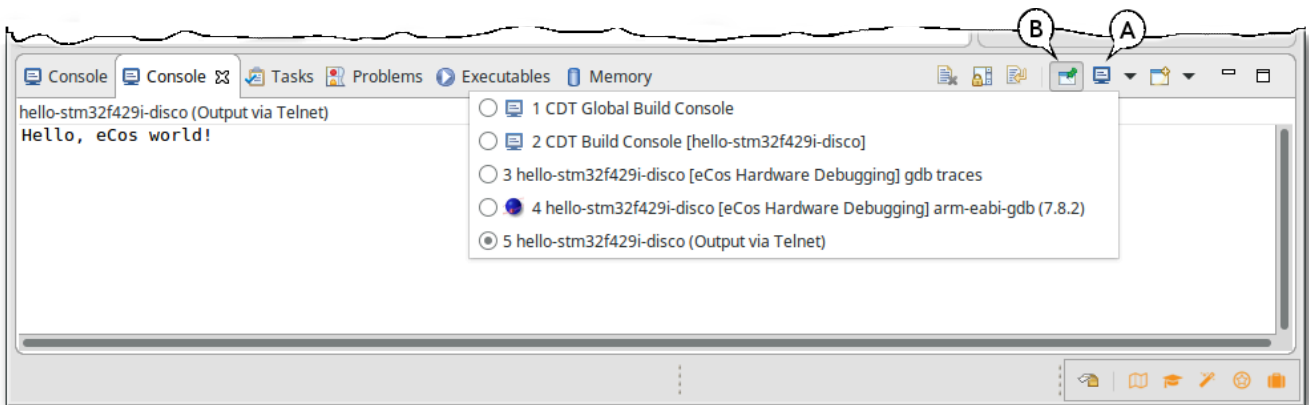
When the application produces any diagnostic output, the *Console tab* window will switch to the *Application debug output* from the *GDB output*, and similarly it will revert to displaying the *GDB output* when the underlying **xxx- gdb** command produces any information of it's own (for example, when a breakpoint is hit). The *Console tab* window may also be pinned to a selected console by first choosing the console using the `Display Selected Console` button highlighted by the **"A"** callout box, choosing the desired console in the resulting menu dropdown, and selecting the `Pin Console` button highlighted by the **"B"** callout box illustrated in Figure 4.10, "Console Diagnostics Output".

The diagnostic output is normally routed through **GDB** and either (i) RedBoot or GDB stubs; or (ii) from an external source such as a serial device, ITM port or TCP/IP socket. The source of the diagnostic output from the target application is defined by the eCos configuration and configured within the `eCos Hardware Debugging` launcher for Eclipse. An example of this is described in the section called "Using ARM's Instrumentation Trace Macrocell for output".

Using ARM's Instrumentation Trace Macrocell for output

This section provides an example how to create an eCosPro configuration that will direct diagnostic output through ARM's Instrumentation Trace Macrocell (ITM) on a Cortex-M target to *OpenOCD* running on the development host from where it may be picked up and displayed on the Eclipse console.

This facility is particularly useful where you may have timing-dependant operations or devices on your target hardware that need servicing and cannot afford the latency overhead required to provide diagnostics output through *GDB*, *Eclipse* or other third-party plug-ins. Diagnostic output is directed to an ITM channel from where it may be decoded on a host and presented to the developer within an Eclipse console.

Note

When large amounts of diagnostic data are produced, certain host or hardware debuggers may not be able to capture, or keep up with the display of, all the diagnostics information due to their slower speeds. In such circumstances the loss may range from a few characters through to large chunks of data.

Configuring eCosPro for ITM output

1. Within Eclipse, open the configuration project and select the active configuration file (`.ecc` extension) for editing using the eCos Configuration Tool by double-clicking on the filename.

2. Within the eCos Configuration Tool, from the default configuration for your target, open the `Find in configuration` dialog: `Edit`→`Find` (**Ctrl+F**)

3. Within the `Find in configuration` dialog, type **CYGHWR_HAL_CORTEXM_DIAGNOSTICS_INTERFACE** into the `Find what:` field, ensure `Search in:` is set to *Macro Names* and select `Find Next`.

4. That should take you to the option `Interface to use for HAL diagnostics`. Within the drop-down to the right, select **ITM** followed by **Return**.

5. Set the CYG_HAL_STARTUP startup type to *JTAG* as illustrated in the section called "eCos Configuration for eCos Hardware Debugging".

6. Save the configuration, *File → Save* (**Ctrl+S**), clean the project *Build → Clean*, build the library *Build → Library* (**F7**), open two terminal (shell) windows *Tools → Shell...* (and again *Tools → Shell...*), and exit the configuration tool, *File → Exit* (**Alt+X**).

   These operations build a new eCos library, but also open new *shell terminals* from which *OpenOCD* may be launched to accept *GDB* connections, and also a *parseitm* session that will allow the ITM diagnostics output to be read by Eclipse/CDT from a **TCP** port.

Using OpenOCD to capture eCosPro ITM output

OpenOCD configuration files provided by eCosCentric will have been preconfigured to capture ITM output to the default file tpui.out, so little is required of the developer in this regard. eCosCentric host tools also include an application, *parseitm*, which is used to extract from the tpui.out file the diagnostics output generated by a running eCos application, and pass this onto the standard output stream.

In this mode of development *OpenOCD* is run as a *GDB Server* on the devlopment host, providing a port for *Eclipse*, through *GDB*, to connect to so it can download, execute and allow debugging of the application on the target hardware. *parseitm* is used to extract the diagnostics output to a TCP stream which is read by *Eclipse*

1. Within the first terminal shell window opened, run the command illustrated in Example 4.4, "Running OpenOCD". You may change *ecos_install* to be either ${ECOS_INSTALL_DIR} (bash) or %ECOS_INSTALL_DIR% (Windows cmd) and the port number *9001* in accordance with your own system and requirements.

Example 4.4. Running OpenOCD

```
% openocd -f ecos_install/etc/openocd.cfg -c "gdb_port 9001"
Open On-Chip Debugger 0.9.0 (2016-06-27-01:12)
Licensed under GNU GPL v2
For bug reports, read
 http://openocd.org/doc/doxygen/bugs.html
Info : The selected transport took over low-level target control. The results might differ compared to plain JTA
adapter speed: 2000 kHz
adapter_nsrst_delay: 100
none separate
srst_only separate srst_nogate srst_open_drain connect_deassert_srst
Info : Unable to match requested speed 2000 kHz, using 1800 kHz
Info : Unable to match requested speed 2000 kHz, using 1800 kHz
Info : clock speed 1800 kHz
Info : STLINK v2 JTAG v23 API v2 SWIM v0 VID 0x0483 PID 0x3748
Info : using stlink api v2
Info : Target voltage: 2.892416
Info : stm32f4x.cpu: hardware has 6 breakpoints, 4 watchpoints
```

2. Within the second terminal shell window opened, run the command illustrated in Example 4.5, "Running parseitm". You may change *tpui.out* and the server port number *9002* in accordance with your own system configuration and requirements.

Example 4.5. Running parseitm

```
% parseitm -p 31 -f tpiu.out -s 9002
parseitm daemon started, listening on port 9002, pid=8204.
```

Example 4.6. Terminating a parseitm daemon

```
% telnet localhost 9002
Trying 127.0.0.1...
Connected to localhost.
```

```
Escape character is '^]'.
shutdown
Connection closed by foreign host.
```

Notes:

a. The above examples illustrates usage on a Linux system. **parseitm** will not run as a daemon or service on Windows.

b. To terminate a **parseitm** daemon, telnet to the **parseitm** port and type the command **shutdown** as illustrated in Example 4.6, "Terminating a parseitm daemon". On Windows, you can also use the keypress **Ctrl**+C within the terminal shell window.

Configuring eCosPro CDT plug-in for ITM output

1. Return to the *Debugger* tab of the dialog illustrated in Figure 4.6, "eCos Hardware Debugging dialog" as described in the section called "eCos Hardware Debugging launch configuration".

2. Within the *Remote Target* section illustrated in Figure 4.11, "Hardware Launcher - ITM output via TCP/IP port",

   • Check *Use remote target*, if it is not already selected

   • Select **Generic TCP/IP** within the *JTAG Device:* dropdown.

   • Enter **localhost** into the *Host name or IP address:* field.

   • Enter **9001** into the *Port number:* field.

3. Within the *Console Output* section illustrated in Figure 4.11, "Hardware Launcher - ITM output via TCP/IP port",

   • Select **TCP** within the *Console connection:* dropdown.

   • Enter **localhost** into the *Host name or IP address:* field.

   • Enter **9002** into the *Port number:* field.

4. Within the *Console Output* section illustrated in Figure 4.11, "Hardware Launcher - ITM output via TCP/IP port",

   • Select **TCP** within the *Console connection:* dropdown.

   • Enter **localhost** into the *Host name or IP address:* field.

   • Enter **9002** into the *Port number:* field.

5. Select the *Apply* to save the changes followed by *Debug* to begin a debug session. Refer to the section called "Running and Debugging the application" for more information on running and debugging the application.

Figure 4.11. Hardware Launcher - ITM output via TCP/IP port
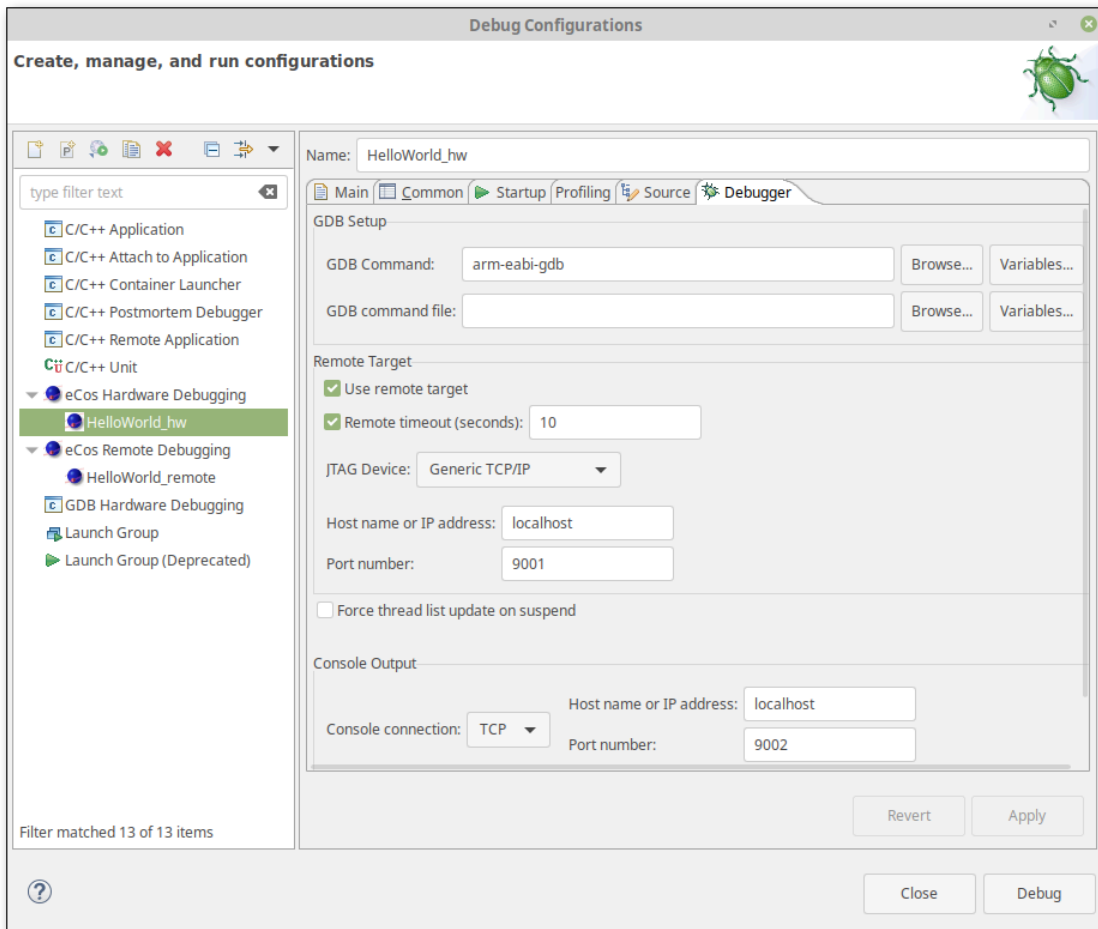
eCos Hardware Debugging using the Ronetix PEEDI

To use a Ronetix PEEDI hardware debugger to debug your eCos Application Project select **PEEDI** as the *JTAG Device* and the right-hand panel of the *eCos Hardware Debugging* launch configuration will change to allow you to enter the *Host name or IP address* and the *Port number* (the GDB port number) of the PEEDI as illustrated in Figure 4.12, "PEEDI Hardware Debugging launch configuration".

You may also specify the source of debug or informational messages which are to appear in the Console Output through the *Console connection* option. See the section called "eCos Configuration for eCos Hardware Debugging" for additional information on configuring eCosPro to provide these messages through the **GDB** connection of the PEEDI.

Figure 4.12. PEEDI Hardware Debugging launch configuration
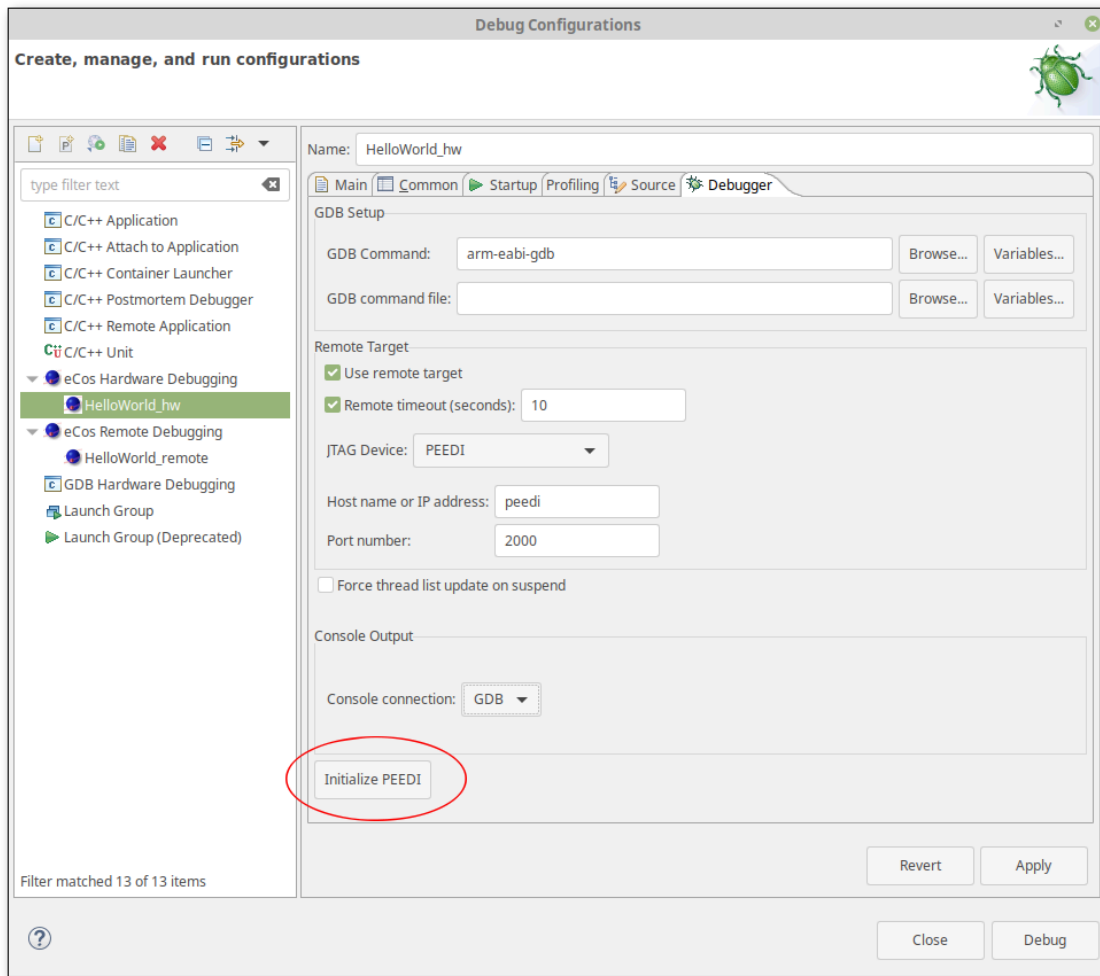
Once your PEEDI has been appropriately configured and initialised to debug your target hardware (see the section called "Configuration and Initialization") select *Apply* to save your launch profile followed by *Debug* to debug your application, or return to the debug walkthrough as described in the section called "Running and Debugging the application".

Licenses and Configuration files

The Ronetix PEEDI must of course be appropriately licensed and configured to debug your target hardware. If the Ronetix PEEDI is supported within the eCosPro distribution and you have not already configured your PEEDI, an example PEEDI configuration file is placed in the ${eCosInstallDir}/etc sub-directory by the eCosPro build system. This is similar to the method used for accessing example OpenOCD configuration files in the section called "eCos Hardware Debugging using OpenOCD".

PEEDI configuration files include a **[LICENSE]** section in which the license keys for the PEEDI and the target hardware are specified. As the licenses may also be located in a separate license file, the generic PEEDI configuration file provided within recent eCosPro distributions assumes that the license keys for the PEEDI are located in the file licenses.txt located within the PEEDI's EEPROM. The **[LICENSE]** section of the configuration file therefore contains:

Example 4.7. PEEDI configuration file including "licenses.txt" file

```
[LICENSE]
FILE = eep://licenses.txt
```

In this case the file licenses.txt on the EEPROM should contain something of the form:

Example 4.8. PEEDI "licenses.txt" file

```
[LICENSE]
KEY = ARM7_ARM9, 1111-2222-3333-4
KEY = UPDATE_29AUG2006, 5555-6666-7777-8
```

Please refer to your PEEDI documentation if you need instructions on how to create a `licenses.txt` file and store this on the PEEDI's EEPROM. Alternatively you may also create your own configuration file where the **[LICENSE]** section contains the relenvant license keys, using the configuration file provided within the eCosPro distribution as a template, and use the dialog below to specify the location of the new license file for storage into the PEEDI's EEPROM.

Configuration and Initialization

This remainder of section describes how this configuration file can be transferred to the PEEDI using the eCosPro CDT plug-in where it is stored and used by the PEEDI for sebsequent debug sessions. This operation only needs to be performed once when the PEEDI needs to be reconfigured to debug your hardware.

Press the additional button *Initialize PEEDI* circled in red in Figure 4.12, "PEEDI Hardware Debugging launch configuration". This will result in a new dialog, illustrated in Figure 4.13, "Hardware Debugger Device Initialization", appearing. This will allow you to set up your PEEDI Hardware debugger with the above-mentioned configuration file in a simple manner.

Figure 4.13. Hardware Debugger Device Initialization
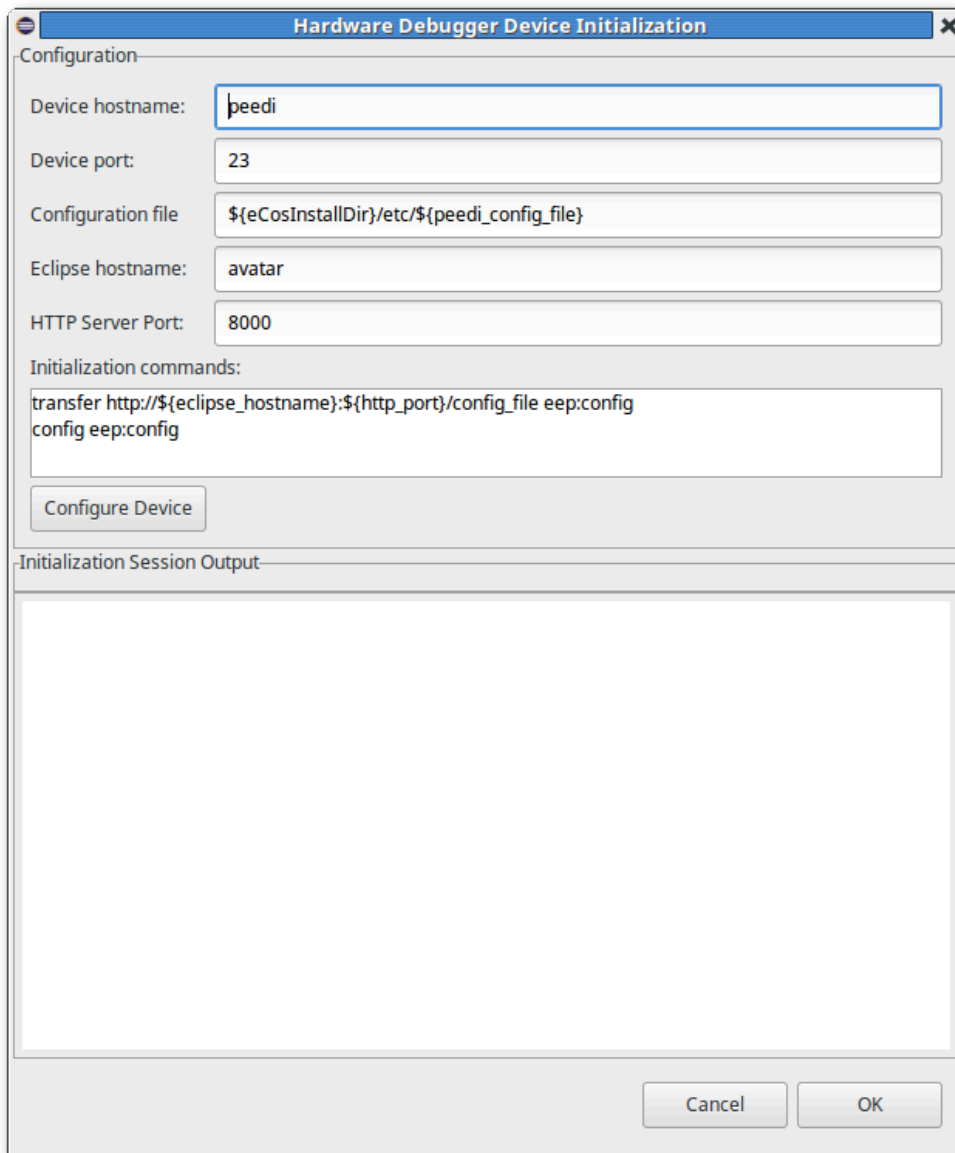
Modify the settings as required and press the `Configure Device` button when you are ready to configure your PEEDI. The following macros are available to you as an aid to simplify your settings and allow them to transfer easily to other devices:

`${eCosInstallDir}` See `${eCosInstallDir}` in Chapter 2, *eCosPro CDT plug-in Macros*.

`${eclipse_hostname}` This defaults to the DNS hostname of the host on which Eclipse is currently running and is the value of the field `Eclipse hostname` within the dialog.

`${http_port}` This defaults to 8000 and is the value of the field `HTTP Server Port` within the dialog.

`${peedi_config_file}` This currently expands as "`peedi.cfg`" by current eCosPro distributions which place a PEEDI configuration file of the same name into the `${eCosInstallDir}/etc` sub-directory. As the name may change depending on the configuration in future eCosPro distributions, use of the macro to name the file is recommended.

Pressing the `Configure Device` button within the resulting dialog will result in the eCosPro CDT plug-in to:

1. run a mini HTTP server that serves only the PEEDI configuration file specified; and

2. telnet to RedBoot on the PEEDI device; and

3. run a predefined set of commands that will download from the HTTP server the configuration file, store it into EEPROM and configure the PEEDI to use this configuration file.

The *Configuration* options of the dialog, with their default values, are as follows:

Device hostname	This is the host name of the PEEDI device. By default it is **peedi**.
Device port	This is the TCP port number of RedBoot on the PEEDI device. By default it is **23**.
Configuration file	This is the full pathname to physical location of the PEEDI configuration file that is to be served by the eCosPro CDT plug-in's mini HTTP server as /config_file. By default it is ${eCosInstallDir}/etc/${peedi_config_filename}, although you may also alter the field to point to your own PEEDI configuration file on your local filesystem.
Eclipse hostname	This is the host name of the development host running Eclipse and the eCosPro CDT plug-in mini HTTP server from which the PEEDI configuration file may be fetched. By default it is the name of your host, although the PEEDI configuration file may be located on any host which is reachable by the PEEDI hardware debugger. The eCosPro CDT plug-in macro ${eclipse_hostname} will be set to this value so that it can be used in the *Initialization commands* below.
HTTP Server Port	This is the port number on which the eCosPro CDT plug-in's mini HTTP server is to provide it's service. By default it is **8000**. The eCosPro CDT plug-in macro ${http_port} will be set to this value so that it can be used in the *Initialization commands* below.
Initialization commands	These are the set of commands, one per line, that are to be sent to *Device hostname*:*Device port* to initialise the PEEDI hardware debugger. The default is illustrated in Example 4.9, "PEEDI Initialization Commands".

Example 4.9. PEEDI Initialization Commands

```
transfer http://${eclipse_hostname}:${http_port}/config_file eep:config
config eep:config
```

The resulting conversation between the PEEDI and the eCosPro CDT plug-in during the initialization session when *Configure Device* is pressed may be viewed within the *Initialization Session Output* section.

Warning

You may need to permit incoming connections to the ${http_port} TCP port of the Eclipse host within your firewall or anti-virus configuration, and on Windows hosts you may also need to grant Eclipse permission to listen for incoming connections on that port.

Notes

1. This dialog may also be configured to do a variety of operations to initialize the PEEDI, including downloading the configuration file from a different host using an alternative protocol supported by the PEEDI.

2. Each line within *Initialization commands* is transmitted after the character ">" is seen.

GDB command files

If necessary, it is possible to override the normal launch behaviour by providing macros in a **"GDB command file"**, traditionally named .gdbinit and located in the same subdirectory as the executable. The path to the file is specified on the *Main* sub-tab of the *Debugger* panel of the eCos Remote debugging configuration dialog, as illustrated in Figure 4.14, "Specify

GDB command file", and in the `GDB Setup` section of the `Debugger` panel of the eCos Hardware debugging configuration dialog, as illustrated in Figure 4.11, "Hardware Launcher - ITM output via TCP/IP port".

Figure 4.14. Specify GDB command file
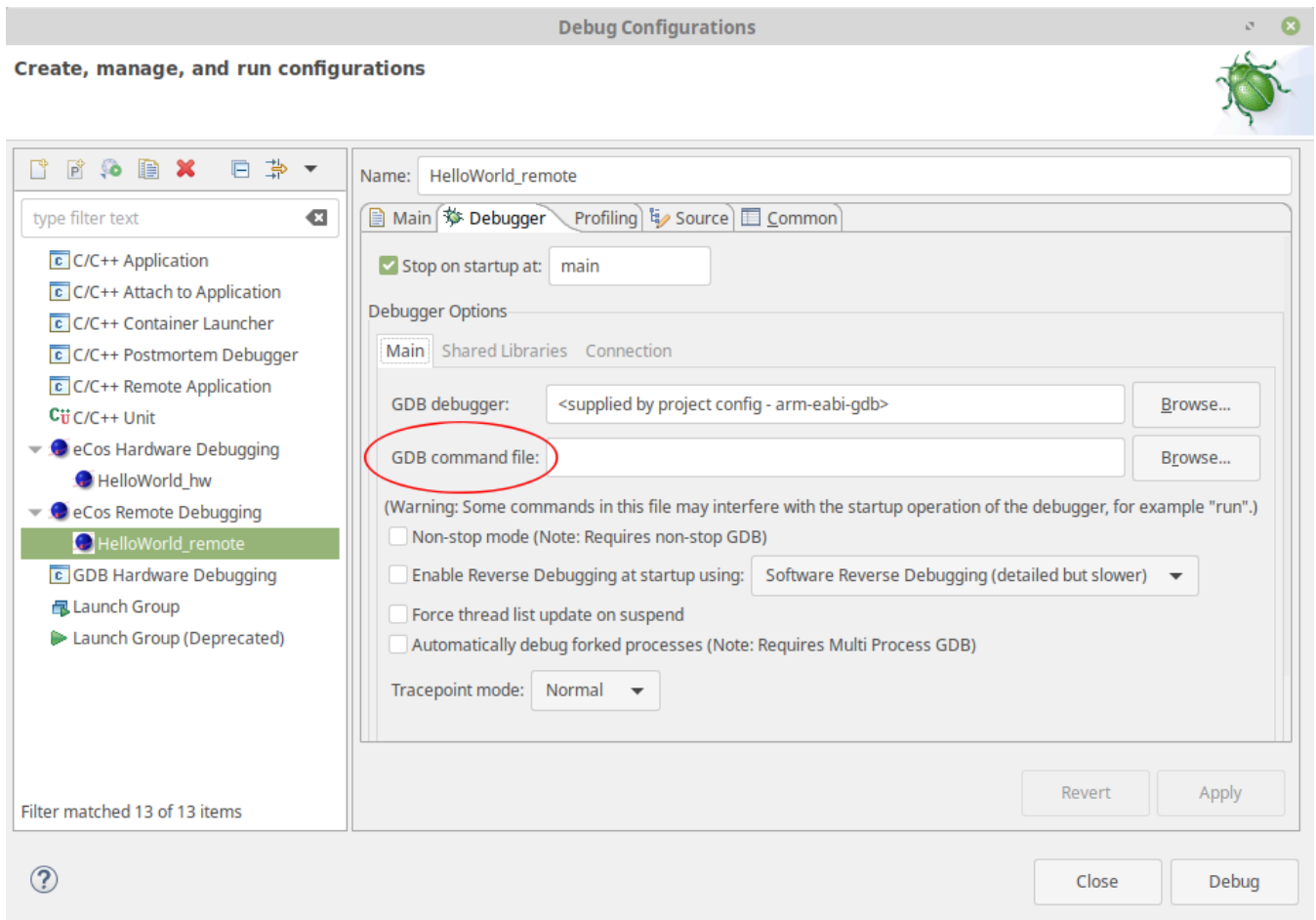
The following macros will, if present, be invoked during launch:

setup   This macro is invoked before connection to the target hardware is attempted. Within the macro, `$arg0` can be used to refer to the ELF executable file which is to be debugged and `$arg1` will provide the connection parameters. (The connection parameters are in the correct format for the commands "**file $arg0**" and "**target remote $arg1**".) For `eCos Hardware Debugging`, an additional `$arg2` argument is provided that is set to the `JTAG Device` type as illustrated in Figure 4.11, "Hardware Launcher - ITM output via TCP/IP port" (for example, "`Generic TCP/IP`", "`OpenOCD`", "`PEEDI`" and so on). This allows the developer to create GDB command files that cater for a variety of devices.

preload   This macro is invoked by CDT following connection to the target and before code download.

doload   This macro is invoked by CDT to download code to the target hardware.

> **Note**
>
> If the *doload* macro is present, the internal commands which would normally download the code onto the target are suppressed.

postload   This macro is invoked by CDT immediately following code download to the target hardware.

An example gdb command file ("gdbinit") that power-cycles a target that is to be debugged into a reset state before attempting to attach to the target is given in Example 4.10, "GDB Command File to reset hardware".

Example 4.10. GDB Command File to reset hardware

```
define setup
shell relay open target
shell sleep 2
shell relay close target
shell sleep 4
end
```

The above script runs the command **relay** on the host running Eclipse to open and close a relay switch controlling the power to the target. It opens the relay with the name `target`, effectively powering off the target, sleeps 2 seconds to allow on-board capacitors to fully discharge, closes the relay to power on the target and finally sleeps for 4 seconds to allow sufficient time for the target to bootstrap into a debug monitor and move to a state that will accept remote debug connections to be made either via a serial connections or via a network connection.

Note

Earlier version of Eclipse and eCosPro required a *preload* macro when debugging using a hardware debugger to **reset** and **halt** the target before downloading the application to the hardware. This is no longer the case for current versions of Eclipse and eCosPro. The majority of eCosCentric provided hardware debugger configuration files now will ensure the target is placed in a suitable state by the hardware debugger before or on accepting a remote debug connection from **gdb**, thereby allowing the application binary to be downloaded or programmed into the target by gdb's **load** command.

Warnings

1. *GDB* is launched with the **-nx** parameter which prohibits it from processing all `.gdbinit` files. If you need to include a `.gdbinit` file, **source** them from within your `GDB command file`.

2. While the *setup*, *preload*, *doload* and *postload* macros may be defined within other command files which are **source**'d from within this command file, either directly or indirectly, they will not be executed by the eCosPro CDT plug-in unless they are defined first within the `GDB command file`.

   ### Tip

   If you require these macros are defined elsewhere, create empty macros in the `GDB command file` and **source** the additional files after defining them. All previous user-supplied definitions will be overwritten by subsequent definitions.

3. Once defined in the `GDB command file`, do not undefine them ("**macro undef** *macro*"). Create an empty macro instead ("**macro define** *macro*").

Chapter 5. About application projects

eCos C/C++ Application Projects

eCos C/C++ Application Projects are Eclipse/CDT Application projects which are associated with a specific eCos Configuration Project. These projects are automatically rebuilt if any changes are made to their associated *eCos Configuration Project* and inherit certain `C/C++ Build` properties from this associated project such as the environment variable `ECOS_INSTALL_DIR`, which points to the `install` subdirectory of the eCos build, as well as the toolchain (compiler, linker and assembler) flags defined by the eCos configuration and toolchain executables defined by the eCosPro Profile of the associated *eCos Configuration Project*.

Application project options

When creating an eCos application project within Eclipse, you have a fundamental choice to make:

* *Managed make* projects are recommended for new users. All source code files within the project are implicitly considered to be part of the application and, in most cases, no further configuration is required. Compilation of these projects is fully managed by Eclipse.

  To use this sort of project for your application, create a C/C++ project of type *eCos Managed Make Application*.

* *Standard make* projects are for users who prefer to write a GNU makefile, or for situations where customised build steps are needed. The wizard offers to create a template makefile with eCos-specific build settings, which we strongly recommend; of course, you can customise as necessary.

  To use this sort of project for your application, create a project of type *Makefile project*. Then either the *eCos Makefile Project* subtype can be used to create a project with a template makefile to use as a starting point, or the *Empty Project* subtype can be used to create a completely empty project. In either case, ensure you select as your toolchain the *!eCos toolchain for standard make projects*.

  The eCos project creation wizard will automatically set important default properties for the project. For example, if you select a project in the Project Explorer and view its properties by right-clicking and selecting Properties, then within the C/C++ Build properties you will see an entry for Environment variables. While the wizard provides sensible default values, you may in future wish to modify variables such as `ECOS_INSTALL_DIR` which by default will point into the eCos Configuration Project you selected at project creation time; or the `PATH` which is used to search for the eCos host tools and GNU toolchain executables.

> **Note**
>
> If changing `ECOS_INSTALL_DIR` to point to a different eCos configuration, then you will probably want to remove or update the reference to the associated eCos Configuration Project. This reference is there to inform Eclipse when a project may need rebuilding because it has a dependent project which has been updated. The procedure for updating this reference is the same as that for Managed Make application projects.

You must also understand what the end result of your project will be. In most cases this will be an *executable*. In some circumstances it is useful to create a *library* project for later use by application projects. To create a managed make library project, use the *eCos Mangaged Make Library* project type, ensuring you also select the *eCos toolchain* as the toolchain.

> **Warning**
>
> A library project should be compiled with the same eCos configuration project which will be used to compile the application. If you use a different eCos configuration, the results will be difficult to predict.

To manage the building of a library yourself use a *Makefile project* in the same way as when creating a standard make application project. But in this case it will be up to you to construct a makefile which generates the necessary library. The name of the library must be assigned to the makefile variable `TARGET`, for example:

```
TARGET = libutils.a
```

No other project types are supported for use with eCos.

Managed-make application projects are created with what Eclipse calls two build *configurations*. The *Debug* configuration is the default; it has debugging symbols enabled and compiler optimization turned off. *Release* turns on optimization and omits debugging symbols; it is generally only used for performance testing and final deployment builds.

Application Project Properties

eCosPro C/C++ managed make application and library projects are created with, or inherit, properties that are defined by the associated *eCos Configuration Project*. These properties may be viewed or edited through the application properties dialog illustrated in Figure 5.1, "eCos Application Properties".

Figure 5.1. eCos Application Properties
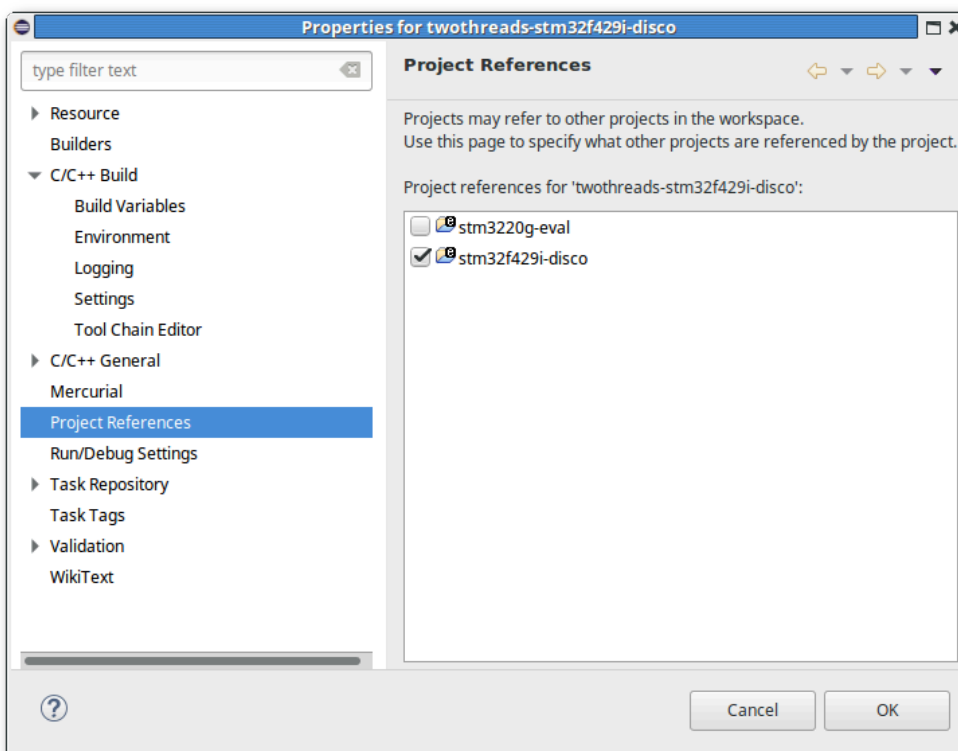
This dialog may be reached by highlighting the eCos application project within the `Project Explorer` window of the C/C++ perspective through the menu option `File → Properties` (**Alt+Enter**) or by `right-click → Properties`.

Project References

When the project references property is highlighted as illustrated in Figure 5.1, "eCos Application Properties", the right-hand panel will display a list of *eCos Configuration Projects* with which the application project may be referenced. This reference defines the eCos configuration, eCosPro profile (and hence also eCos toolchain) and build flags, as well as the eCos library with which the application project is compled and linked. An *eCos application project* may therefore only be associated with a single *eCos Configuration Project*. To change the associated *eCos Configuration Project*, deselect the selected configuration project, select the new configuration project and `OK`. An error will result if the application project is not associated with a single configuration project, otherwise the application's remaining properties will be updated to those appropriate to the associated *eCos Configuration Project*. The application will also be rebuilt against these new properties.

By changing the *eCos Configuration Project*, you can change either the target hardware of the application, or the eCos configuration (for example, between a debug or a release configuration).

C/C++ Build Environment

When the *Environment* item within the *C/C++ Build* properties branch is selected or highlighted, the user will see a list of environment variables appear in the panel on the right hand side, including their values and the source of these values (the **BUILD SYSTEM**, as described in the section called "eCos configuration projects").

CWD
: The current working directory under which the application project is to be built.

ECOS_INSTALL_DIR
: The installation directory where eCos sub-directories such as etc, include and lib are installed when the associated *eCos Configuration Project* is built.

PATH
: The search path for executables to be used in building the application which will include at its head the paths to the execuable directories of the eCos Host Tools and the GNU Toolchain for the eCosPro Profile with which the referenced *eCos Configuration Project* is associated.

PWD
: The present working directory under which the application project build is initially started. This is normally the same as the CWD.

C/C++ Build / Tool Settings

When the *Settings* item within the *C/C++ Build* properties branch is selected or highlighted, the panel on the right will include a *Tool Settings* tab in which the compiler, assembler and linker commands are defined as well as the flags that are passed to each when each of the application's source files are compiled or assembled, or when the application is linked with the eCos library resulting from the referenced *eCos Configuration Project* as illustrated in Figure 5.2, "C/C++ Build Properties / Tools Settings".

Figure 5.2. C/C++ Build Properties / Tools Settings
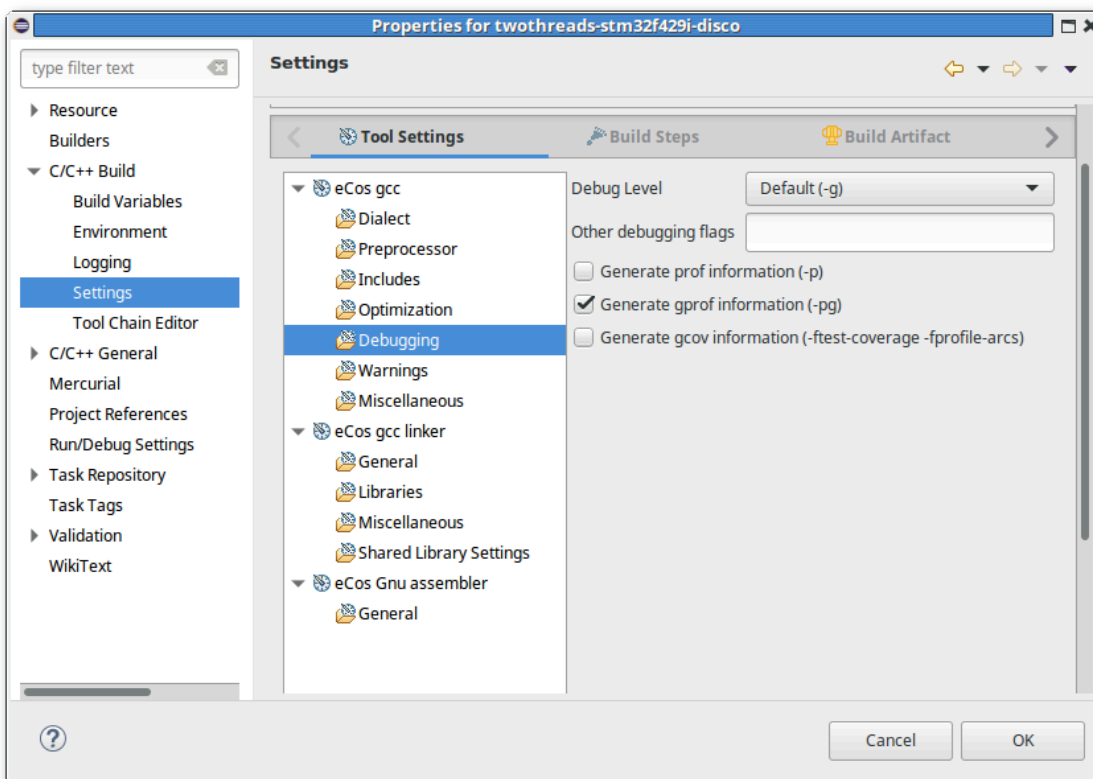
The defaults for these settings are inherited from the referenced *eCos Configuration Project*, but may be edited in accordance to the developer's requirements. For example, code optimisation may be enhanced, or the code may be generated to include **gprof** calls and information.

Chapter 6. About configuration projects

eCos configuration projects

eCos configuration projects are used to hold and build eCos configurations. One configuration may be shared across multiple application projects.

- These projects contain one or more `.ecc` files, build and install directories, as well as an Eclipse *link* to the source repository defined by the eCosPro profile.

    - If there are multiple `.ecc` files, only one may be selected as the active *eCos configuration file name*. This file name is set within the *eCos Configuration* property of the *eCos Configuration Project*. For details on how to change the active configuration file, refer to the section called "Using a different configuration file".

    - Providing the default behaviour has not been changed, double-clicking on a `.ecc` file opens it in the *eCos Configuration Tool*; when this tool exits, the plug-in checks if the `.ecc` has been changed and triggers a rebuild if so and it is the active *eCos configuration file name*.

    - While it is also possible to build eCos from within in the *eCos Configuration Tool*, it is not required as Eclipse will rebuild eCos as necessary whenever the configuration changes.

    - Right-clicking on a `.ecc` file will bring up an *"Open With"* menu item under which you can also choose to edit the eCos configuration using the built-in *Textual eCos Configuration Editor*; when this tool exits, the plug-in will again check for changes which may trigger a rebuild if this is also the *eCos configuration file name* as above.

    - Should there be any compilation failures in your eCos configuration, they are automatically highlighted and turned into Eclipse *problem markers* which you can double-click on to go directly to the error.

- eCos configuration projects are associated with an eCosPro profile which specifies the eCos source base for the project. For details on how to switch to a different eCosPro profile refer to the section called "Changing the configuration project's profile".

- Eclipse normally builds eCos configurations automatically in the background if the *Project* → *Build Automatically* option is selected. However, there is a per-project *Inhibit building this project* option which prevents a configuration from being built or rebuilt - if, for example, you create a new configuration but wish to customise it further before building.

    If set, any attempts to build the configuration project either manually or automatically will result in the error:

    Example 6.1. eCos Builds Inhibited Error

    ```
    Builds of project XXX are inhibited. To enable building, select the project
    and uncheck "Inhibit configuration build" in the eCos menu.
    ```

    The *Inhibit building this project* option may be changed through the project properties dialog when selecting the *eCos Configuration* item. See Figure 6.1, "eCos Configuration Properties" for an example of the *"eCos Configuration"* properties dialog.

- A wizard may be used in order to create eCos configuration projects. This wizard allows you to either import an existing configuration (`.ecc` file), or create an entirely new eCos configuration project. If importing an existing configuration, the `.ecc` file is copied into your project, and the original will not be modified. Bear in mind this also means any changes you make to the `.ecc` file within your configuration project will similarly not be reflected in the original `.ecc` file.

    > **Note**
    >
    > The wizard for importing an existing eCos configuration is listed as both a *New* wizard and as an *Import*. The functionality is the same in both cases: a new *eCos configuration project* is created and the `.ecc` file is copied into it.

It is also possible to create a configuration project or import an existing configuration at the same time as creating an application project.

Changing the eCos configuration

This can be done in three ways.

1. Editing the eCos configuration project

2. Changing the configuration project to use a different configuration file

3. Changing the application project to use a different configuration project

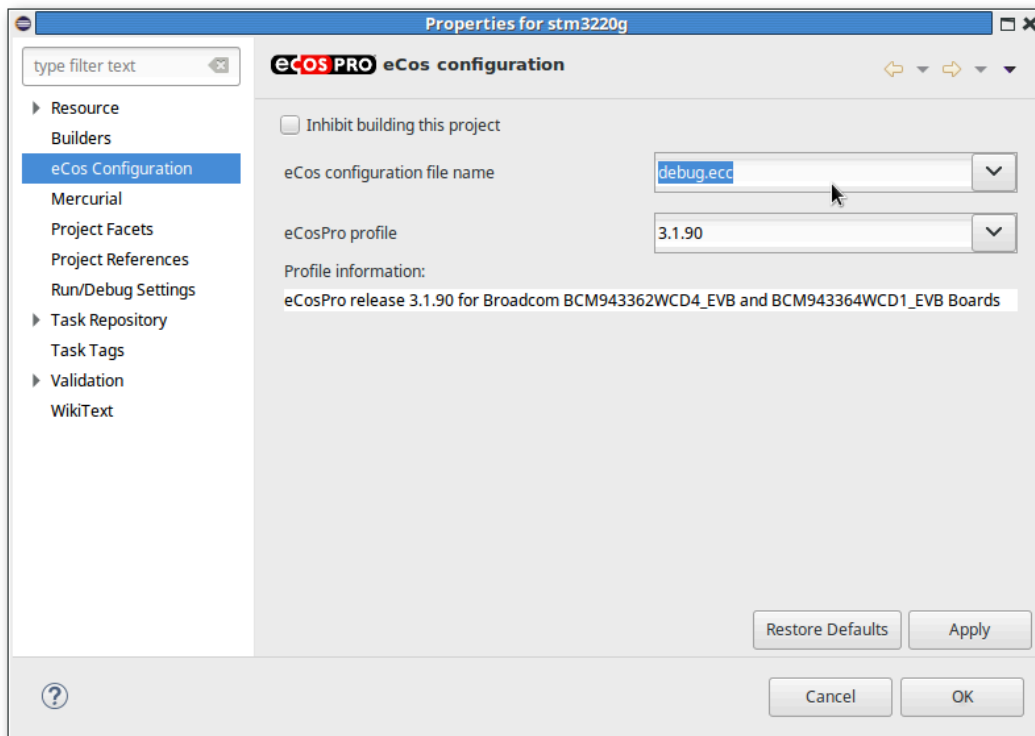# Editing the eCos configuration project

The configuration project contains an eCos configuration file which is named `ecos.ecc` by default. Double- clicking on this opens it up, by default, with the graphical *eCos Configuration Tool* (externally to Eclipse/CDT). When you close the eCos Configuration Tool, the eCosPro CDT plug-in checks to see whether the configuration has changed and, if so, triggers a rebuild (unless the *Inhibit building this project* project property is set).

> **Note**
>
> This per-project *inhibit* option is not set by default. If you set this option you must explicitly clear it before it is possible to (re)build eCos. It is set and cleared by first clicking on the project in the Project Explorer, and either selecting the *Inhibit configuration build* option from within the *eCos* drop-down menu; or by opening the project properties using `File` → `Properties` (**Alt+Enter**) or by `right-click` → `Properties`, then selecting *Inhibit building this project* in the *eCos Configuration* panel illustrated in Figure 6.1, "eCos Configuration Properties".

Figure 6.1. eCos Configuration Properties

> ⚠️ **Caution**
>
> Certain changes to an eCos configuration are more wide-ranging than others. Eclipse has no way of knowing of whether a configuration requires a clean build; this is due to inherent limitations in the eCos build system. We therefore recommend that, if you are unsure, you err on the side of caution and force a clean build (`Project →  Clean`) of both your configuration and application projects whenever you change your eCos configuration.

Using a different configuration file

Occasionally you may have more than one eCos configuration file (files with `.ecc` extensions) within your eCos configuration project. For example, you may have the files `release.ecc` and `debug.ecc` which contain identical eCos configurations apart from settings which effect debugging. The CYGBLD_GLOBAL_CFLAGS setting may contain the **-O2** flag in `release.ecc` and **-O0** in `debug.ecc`, which may also have and the CYGPKG_INFRA_DEBUG package enabled.

For example, to create a copy of an existing eCos configuration file, make minor modifications to it, and switch the *eCos Configuration Project* to use this file, you can use the following process:

1. Right-click on an existing eCos configuration file and select *Copy*.

2. Right-click again on the eCos configuration file and select *Paste*. Within the *Name Conflict* popup box, enter the name of the new configuration file (e.g. `debug.ecc`).

3. Right-click on *eCos Configuration Project* and select *Properties* followed by *eCos Configuration* in the resulting dialog box.

4. Select the appropriate `.ecc` file from the drop- down field next to `eCos configuration file name` followed by `Apply` and `OK` as illustrated in Figure 6.1, "eCos Configuration Properties".

5. Clean the eCos application project (`Project → Clean`) to be sure that it rebuilds correctly.

Using a different configuration project

Sometimes you may have a number of eCos configuration projects you may wish to choose between, perhaps with differing eCos packages, perhaps for different hardware, or perhaps even an alternative to the section called "Using a different configuration file" where you have one eCos Configuration Project for Release builds in Eclipse and a different one for Debug builds. In this case, you can use the following process to switch to an entirely different eCos configuration project:

1. Create the new configuration project, if you have not already done so.

2. Open up the project's properties dialog using `File → Properties` (**Alt+Enter**) or by `right-click → Properties`.

3. Click on the *Project References* panel.

4. Remove (uncheck) the link to the original configuration project and add a link to the new configuration project as illustrated illustrated in Figure 6.2, "Select eCos configuration file".

Figure 6.2. Select eCos configuration file
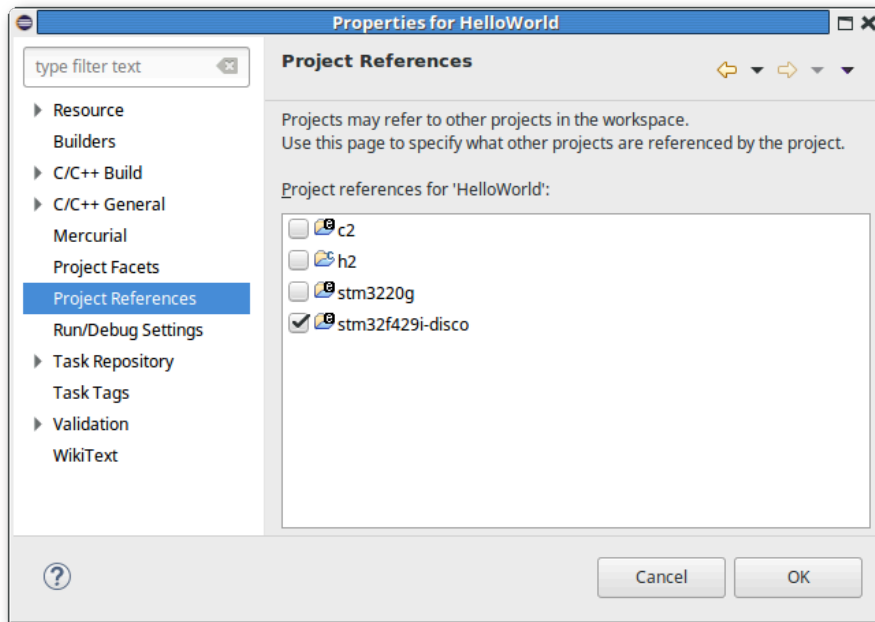
5. Clean your application, **Project**→*Clean*, to be sure that it rebuilds correctly.

Changing the configuration project's profile

![Caution icon] **Caution**

While the eCosPro Eclipse plug-in allows you to change the profile of an eCos configuration project, for example to allow you to make use of a newer version of the GNU tools, you must carefully consider the consequences before switching to profiles which contain different eCosPro repositories. `.ecc` eCos configuration files do not migrate easily between different eCos releases.

The eCosPro profile is initially set when you create a new eCos configuration project as detailed in the Quick Start / Walkthrough chapter but may also be changed in the *"Properties"* dialog box for the configuration project as illustrated in Figure 6.3, "Select eCosPro profile".

Figure 6.3. Select eCosPro profile
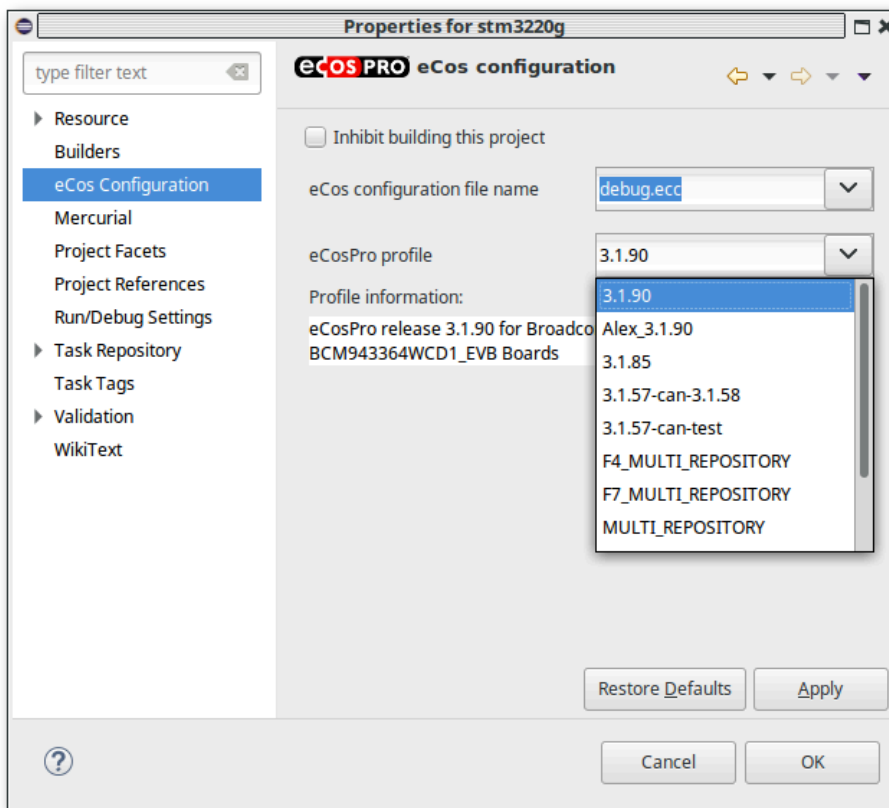
The above dialog window can be reached by *File → Properties* (**Alt+Enter**) or by *right-click → Properties* while the eCos Configuration project is selected within the *Project Explorer* window and selecting *eCos Configuration*.

> ## Warning
>
> A build error is normally expected when you switch to a different eCosPro profile that uses a different eCos repository. This is because eCos configurations contain version information and potentially also can include settings and packages that may not be known in another repository. For example, different eCosPro repositories may range from a simple update to an existing eCosPro release through to an eCosPro release for an entirely different architecture. Developers are therefore advised to follow the instructions in the Upgrading eCosPro Configurations to newer eCosPro releases section of the eCosPro Developer's Kit - Installation and Getting Started Guide to update .ecc files within an eCos configuration project.

Chapter 7. GProf Profile Plugin Support

The *Linux Tools GProf plugin* brings the profiling capabilities of the GNU profiler, **gprof**, to Eclipse, in a manner that is easy to use by developers with different levels of experience. eCos and eCosPro have the ability to generate the data used in the generation of **gprof** profile timing and call graphs (for additional information, please refer to the Profiling section of the eCos and eCosPro Reference Manual). The eCosPro CDT plug-in includes functionality to extract the data from an eCosPro application that is running, or has been halted on the target hardware, through the Eclipse GUI and provides as input this data to the *Linux Tools GProf plugin*.

This section describes how to bring all the above functionality together to permit you to graphically view and explore the target hardware application's profile and timing data on your development host. This will allow you to analyse and explore your code within the Eclipse GUI to determine not only the parts of your application which are slower in execution than expected, but also can help you find many other statistics through which potential bugs can discovered and resolved. This section also provides a walk-through of the installation of the *Linux Tools GProf plugin*, the configuration of eCosPro and creation and compilation of the eCosPro C/C++ Application Project such that profiling data is generated, through to the capture and display of the profiling data.

> **Note**
>
> GProf Profile support is not available on all target platforms. It requires an implementation of the profiling timer, there should be a hardware- specific implementation of `mcount` and it is not currently available for multi-core (SMP) systems. For further details, please refer to the Profiling documentation and the target-specific eCosPro documentation.

GProf Profile Plugin Installation

The current Eclipse distribution provided with the eCosPro Developer's Kit version 4.x and above includes the *Linux Tools GProf plugin* pre-installed. If you are using an earlier version of Eclipse or a distribution from a different source, you can follow the instructions in this section to install the *Linux Tools GProf plugin*.

Check if Linux Tools GProf plugin is installed

To check whether the *Linux Tools GProf plugin* is installed, within Eclipse select from the menu `Help`→ `About Eclipse IDE` and press the `Installation Details` button. Within the resulting dialog illustrated in Figure 7.1, "Eclipse Installation Details - GProf" look in the `Configuration contents:` section for "**GProf Integration**". If present, the plugin is installed.

Figure 7.1. Eclipse Installation Details - GProf
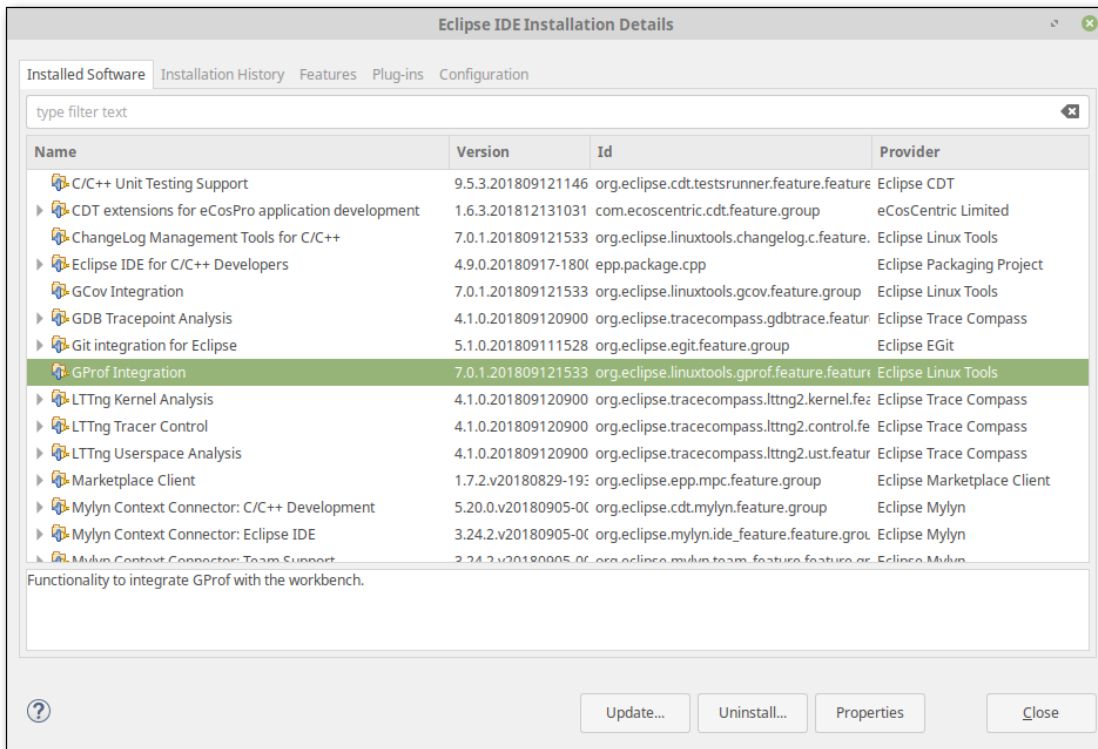
Install Linux Tools GProf plugin

To install the *Linux Tools GProf plugin* is installed you will need internet access. Within Eclipse select from the menu `Help` → `Install New Software` and within the `Work with:` field of the resulting dialog, enter the name of your Eclipse Installation and choose the download URL. An example illustration of the dialog is shown in Figure 7.2, "Eclipse Install GProf". For example, "`2018-09 - http://download.eclipse.org/releases/2018-09`".

Figure 7.2. Eclipse Install GProf
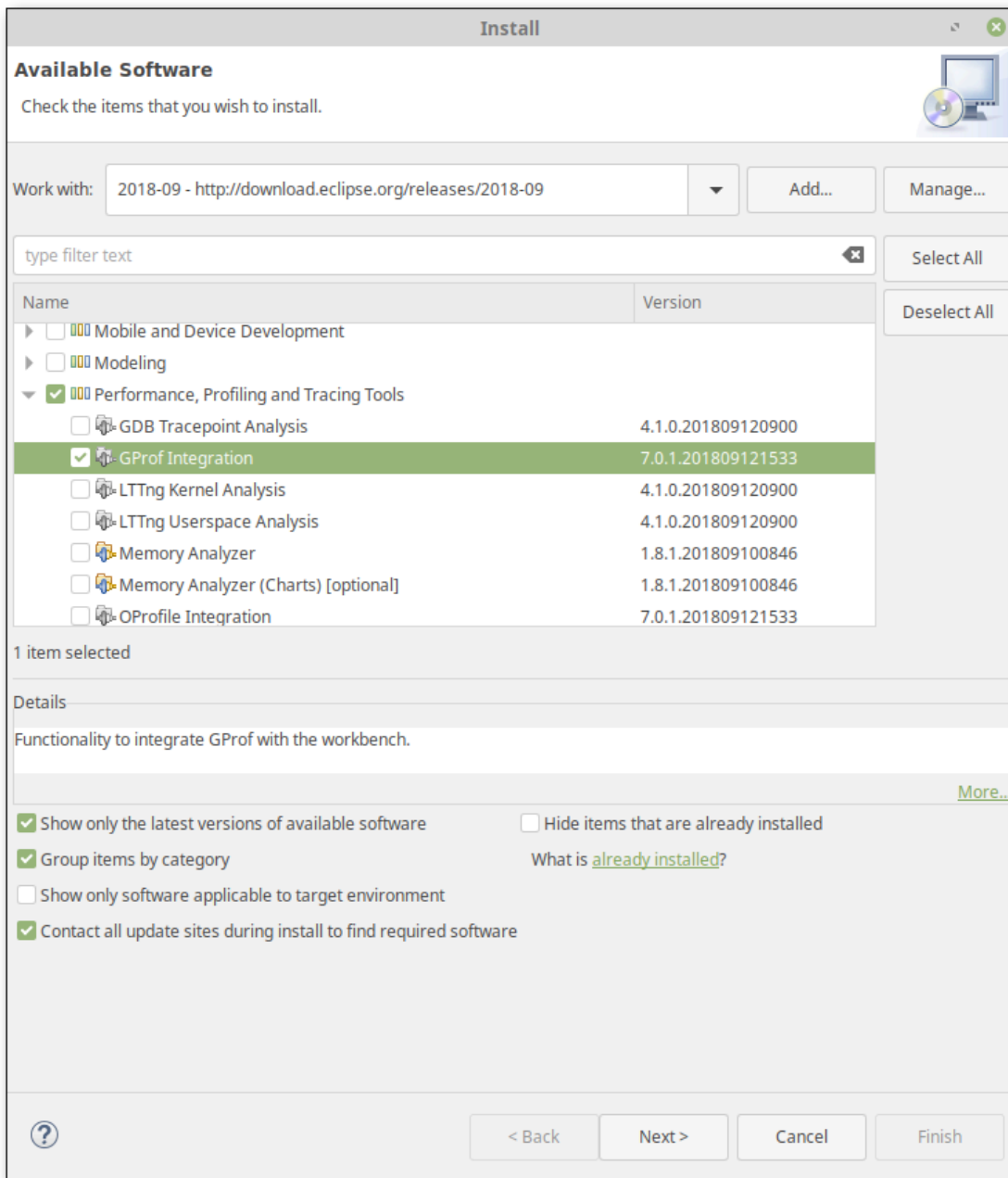
Select **Performance, Profiling and Tracing Tools** → *GProf Integration* and ensure the latter is checked. Select the **Next** button to confirm the install details followed by **Next** to confirm your acceptance of the license terms by selecting **I accept the terms of the license agreement**. Select the **Finish** button to confirm agreement and begin the installation process. On completion of the installation you will be required to restart Eclipse.

The eCosPro Runtime profile statistics package

Runtime support is required from the operating system in order to create and store the profiling data on the target platform and may easily be added by including the **Application profile support** package, also known as **CYGPKG_PROFILE_GPROF**, into your eCos configuration. This may be achieved through one of two methods:

1. the section called "Adding CYGPKG_PROFILE_GPROF with the eCos Configuration Tool"

2. the section called "Adding CYGPKG_PROFILE_GPROF with the command line"

Adding CYGPKG_PROFILE_GPROF with the eCos Configuration Tool

Open up your eCos configuration within the *eCos Configuration Tool* and select `Build → Packages` (**Ctrl+P**) and type **gprof** into the `Keywords` field as illustrated in Figure 7.3, "Configuration Tool Install GProf Package".

Figure 7.3. Configuration Tool Install GProf Package
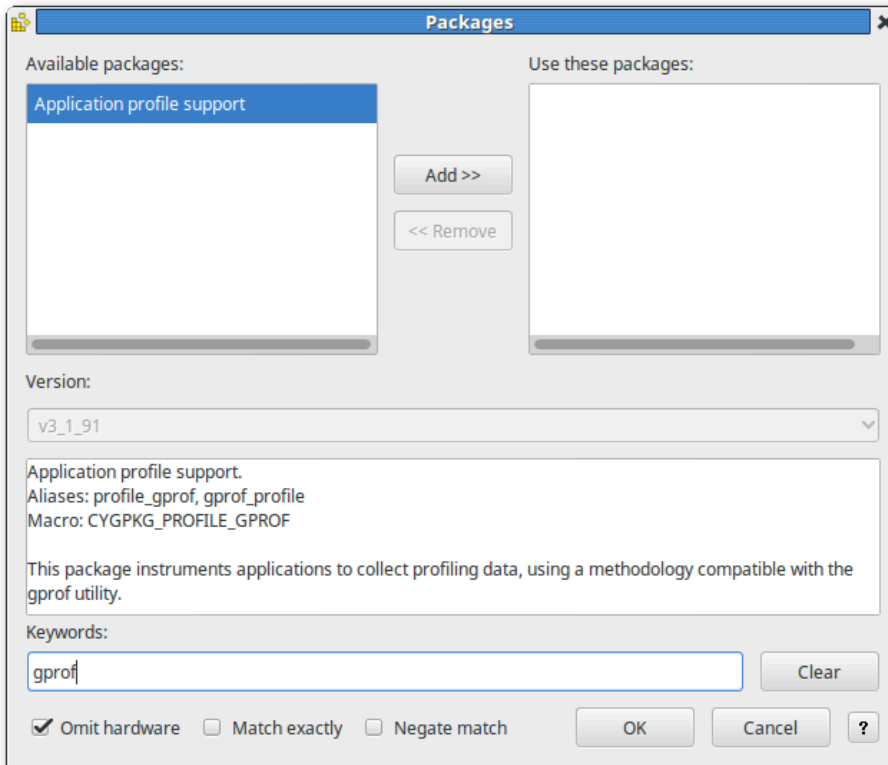
If already installed, the package **Application profile support** will appear in the right-hand column under `Use these packages`. If not installed, it will appear under the `Available Packages` column. In this case select the `Add` button to move the package to the right-hand column followed by selecting the `OK` button to accept the addition of the package. Finally select `File → Save` (**Ctrl+S**) to save your configuration followed optionally by `Build → Library` (**F7**) to rebuild the eCos library. The latter step is optional as Eclipse will rebuild the library as soon as you exit the *eCos Configuration Tool* if it detects a change in the active configuration file.

Adding CYGPKG_PROFILE_GPROF with the command line

Open a command shell with the appropriate environment variables set and, assuming your eCos configuration is named ecos.ecc, the commands illustrated in Example 7.1, "Adding CYGPKG_PROFILE_GPROF with the command line" will add the **CYGPKG_PROFILE_GPROF** package and rebuild the eCos library.

Example 7.1. Adding CYGPKG_PROFILE_GPROF with the command line

```
$ ecosconfig add CYGPKG_PROFILE_GPROF
$ ecosconfig tree
$ make
```

Enabling TFTP support for profiling data extraction

Normally the capture of profiling data from the target platform requires the temporary suspension of all code execution on the target platform while the data is extracted, either through the use of a GDB monitor or a hardware debugger.

However, if your eCos Configuration includes the FredBSD networking stack (the **CYGPKG_NET_FREEBSD_STACK** package) with the TFTP server option enabled (the default), the profiling data may be extracted from the target platform without temporarily suspending all code execution. This is achieved through the use of an additional low- priority eCos thread that provides a TFTP service (on port 69, the default) which allows the transfer of the profiling data from the target hardware to occur over TFTP. The eCos **CYGPKG_PROFILE_GPROF** package by default creates this thread when the **CYGPKG_NET_FREEBSD_STACK** package is enabled.

When capturing profiling data in this manner, the eCosPro CDT plug-in needs to be configured where the profiling data will be captured from, otherwise the default method through *GDB* is used. Open the debug launch configuration window you previously created to launch the binary in the section called "eCos Launch Configurations" and select the *Profiling* tab. This dialog is illustrated in Figure 7.4, "Profiling Data Capture through TFTP".

Figure 7.4. Profiling Data Capture through TFTP
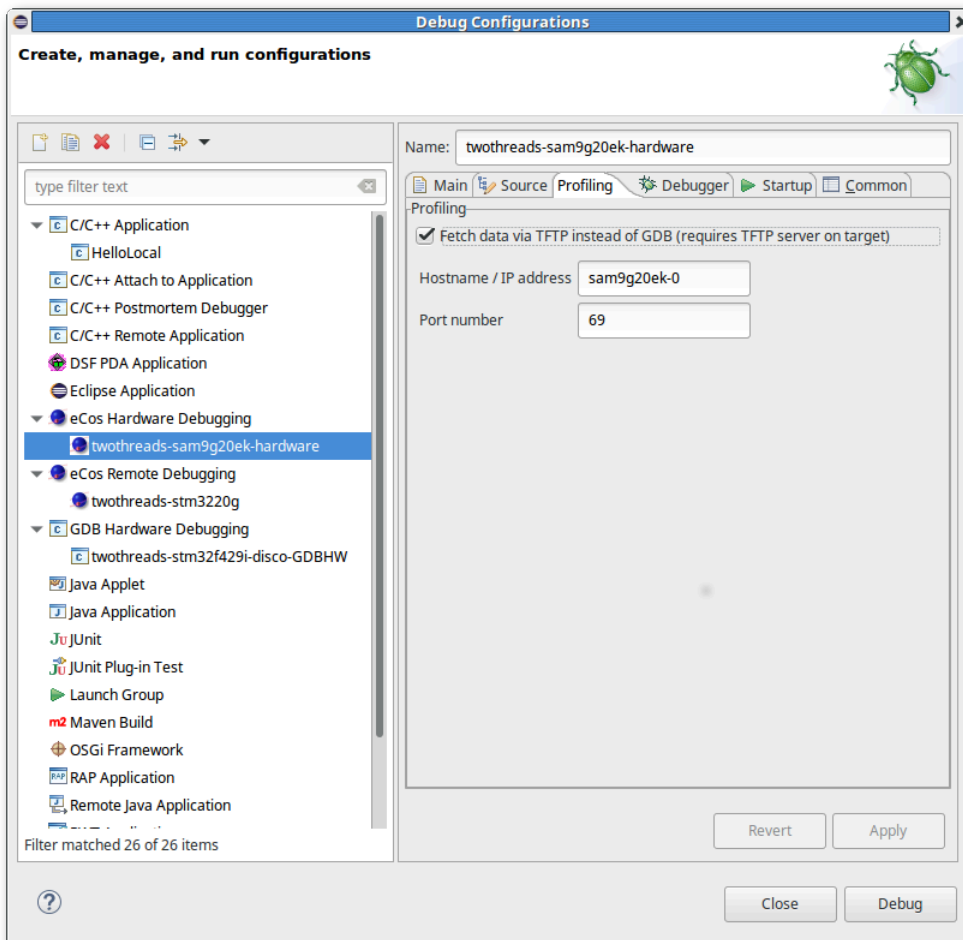
Check the *Fetch data via TFTP instad of GDB (requires TFTP server on target)* field and fill in appropriate values for the *Hostname / IP address* for the target platform's network address and *Port number*. The default port number set by the **CYGPKG_NET_FREEBSD_STACK** package for the TFTP server is **69**.

# Enabling profiling data generation within eCos and the eCosPro application

The generation of profiling data does not happen automatically when the **CYGPKG_NET_FREEBSD_STACK** package is added to eCos. The inclusion of this package within your eCos configuration just enables runtime support. Code has to be compiled with the **-pg** GNU compiler flag to make it capable of generating profiling data.

Compiling the application with the `-pg` GNU compiler flag

You may enable this flag project-wide by bringing up the project's properties dialog as described in the section called "Application Project Properties", and selecting *C/C++ Build* → *Settings*, selecting the *Tool Settings* tab in the right-hand panel and within tree of settings selecting either *eCos gcc* → *Debugging* or *eCos g++* → *Debugging* whether you wish profiling data to be generated for either C or C++ code, or both. Check the *Generate gprof information (-pg)* field from the right-hand panel, apply the changes and close the dialog by pressing the *Apply* and *OK* buttons respectively. This is illustrated in Figure 7.5, "Enable Application Profiling Data Generation".

Figure 7.5. Enable Application Profiling Data Generation
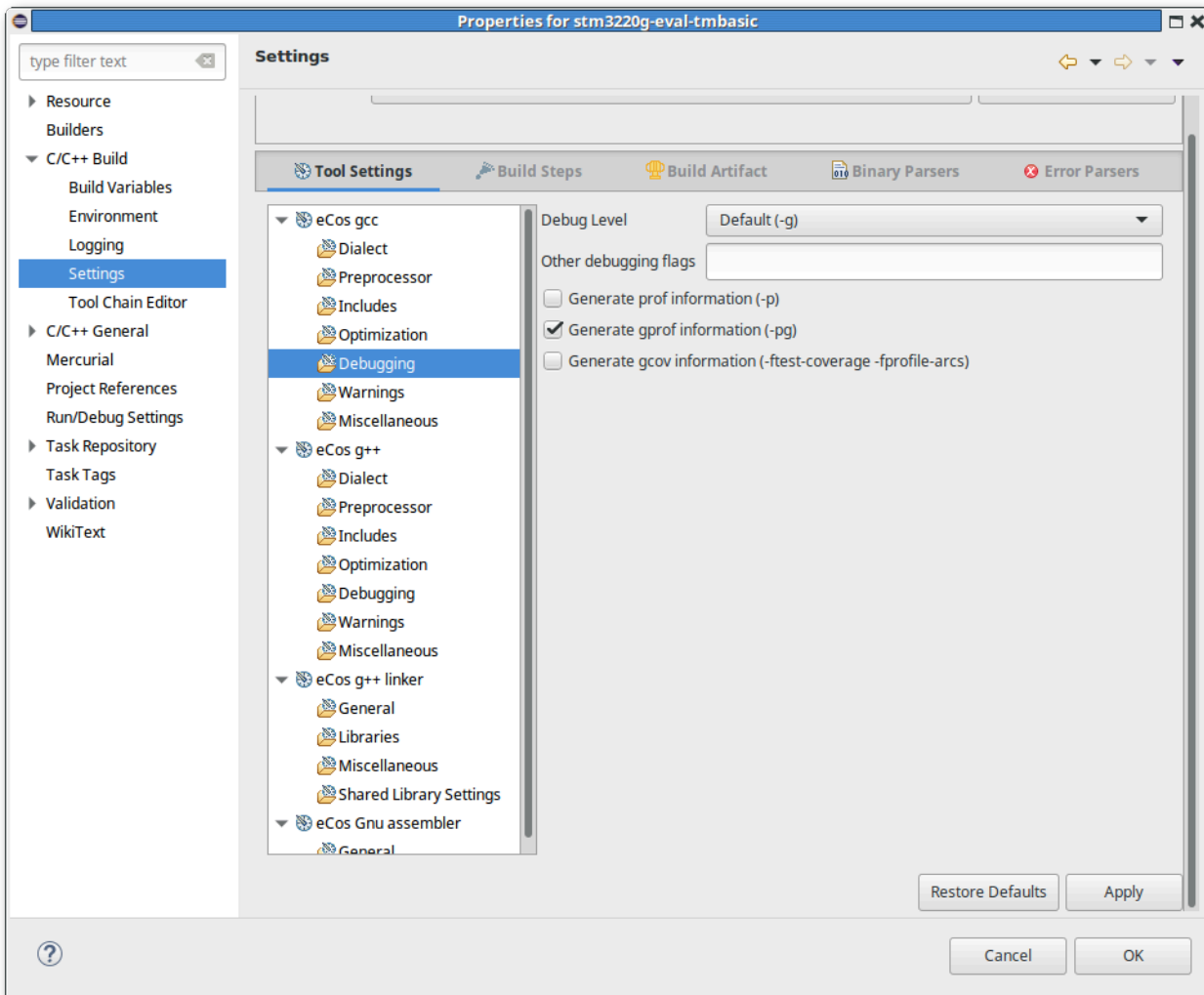
The `-pg` GNU compiler flag may also be enabled or disabled for individual source files by opening the properties dialog for each source file (similar to opening the application project's properties, but start by highlighting the source file within the *Project Explorer* window of the C/C++ perspective) and checking or unchecking the *Generate gprof information (-pg)* field.

Compiling eCos with the `-pg` GNU compiler flag

eCos and eCosPro functions may also be included for profiling data generation and analysis. This may be acheived by adding the `-pg` GNU compiler flag to the value of **CYGBLD_GLOBAL_CFLAGS** configuration macro. To add or remove the flag, open the eCos configuration in the eCos Configuraion Tool as described in the section called "Editing the eCos configuration project", and search using the *Find in configuration* dialog, reached through the menu options *Edit* → *Find* (**Ctrl+F**) for **CYGBLD_GLOBAL_CFLAGS** (as *Find what*) with *Search in* set to **Macro Names**. Once visible, left mouse click in the value field (or double-left click to bring up a *String Edit* dialog) and add or remove the `-pg` flag to/from the value as required.

Finally select **File** → *Save* (**Ctrl+S**) to save your configuration followed optionally by **Build** → *Library* (**F7**) to rebuild the eCos library.

Enabling and Disabling profiling data collection

Enabling and disabling the collection of profiling data is done programatically by the application through the following two functions: `profile_on` and `profile_off`.

Enable profiling data collection

The application must call the function `profile_on` to start the collection of profiling data. If the TFTP daemon is enabled, the call to `profile_on` must happen once the network is up and running, typically after the call to `init_all_network_interfaces`. This is because the TFTP daemon will be started within `profile_on`.

> **Note**
>
> `profile_on` may be invoked multiple times but each invocation will allocate a fresh profiling range, deleting the previous range.

A typical example is illustrated in Example 7.2, "Enable profiling data collection".

Example 7.2. Enable profiling data collection

```
#include <pkgconf/system.h>
#ifdef CYGPKG_NET
# include <network.h>
#endif
#ifdef CYGPKG_PROFILE_GPROF
# include <cyg/profile/profile.h>
#endif
…
int
main(int argc, char** argv)
{
    …
#ifdef CYGPKG_NET
    init_all_network_interfaces();
#endif
    …

#ifdef CYGPKG_PROFILE_GPROF
    {
        extern char _stext[], _etext[];
        profile_on(_stext, _etext, 16, 3500);
    }
#endif
    …
}
```

The `profile_on` function takes four arguments:

start address end address	These two arguments specify a range of addresses that are to be profiled in a contiguous section of memory. The eCos linker script export the symbols _stext and _etext on most targets and these correspond to the beginning and end of code. Profiling may be performed on a subset of code by specifying the start and end addresses of the code region on which profiling is to be performed.
bucket size	This is the bucket size which the `profile_on` divides the range of addresses into. It dynamically allocates a single array of 16-bit counters with one entry for each bucket. When the profiling timer interrupts, the interrupt handler will examine the program counter of the interrupted code and, assuming it is within the range of valid addresses, find the containing bucket and increment the appropriate counter.

Notes

- The number of 16-bit counters is determined by the range of addresses being profiled and by the bucket size. Smaller bucket sizes and large memory ranges will require more memory, so if the target hardware is low on memory this may be unacceptable. The memory requirements can be reduced by reducing the code range or increasing the bucket size. The latter increase will affect the accuracy of the results, making gprof more likely to report the wrong function and increasing the risk of a counter overflow.

- The bucket size must be a power of 2 and will be adjusted if necessary. This is done for the sake of run-time efficiency.

`time interval`   This specifies, in units of microseconds, the interval between profile timer interrupts. Increasing this value gives more accurate profiling results but will result in higher run-time overheads and a greater risk of a counter overflow. This value may be modified by the implementation because of hardware restrictions, so as a result the generated profile data contains the actual interval used.

Tip

Avoid using an interval that is a simple fraction of the system clock, typically 10000 microseconds, as use of such intervals runs the risk that the profiling timer will disproportionally sample code that runs only in response to the system clock.

Disabling profiling data collection

The collection of profiling data may be disabled by the application, using a call to `profile_off`. This will also reset any existing profile data. The function prototype is illustrated in Example 7.3, "profile_off prototype".

Example 7.3. profile_off prototype

```
void profile_off(void);
```

Extracting and Clearing the profiling data from the target

Extracting and clearing the profiling data from the target platform is a simple operation with eCosPro and the eCosPro CDT plug-in. When eCosPro is configured and built as described in the section called "Adding CYGPKG_PROFILE_GPROF with the eCos Configuration Tool", the series of macros required to extract and clear the profiling data from the target platform eCos are installed in a file within the `${eCosInstallDir}/etc` directory. When required, these macros are loaded by the eCosPro CDT plug-in into *GDB* and executed.

Also ensure that you have included the code described in the section called "Enable profiling data collection" into your application, configured the launch profile for TFTP if required as described in the section called "Enabling TFTP support for profiling data extraction", and start executing and debugging your application in the normal fashion.

Figure 7.6. Capture and Clear Profiling Data
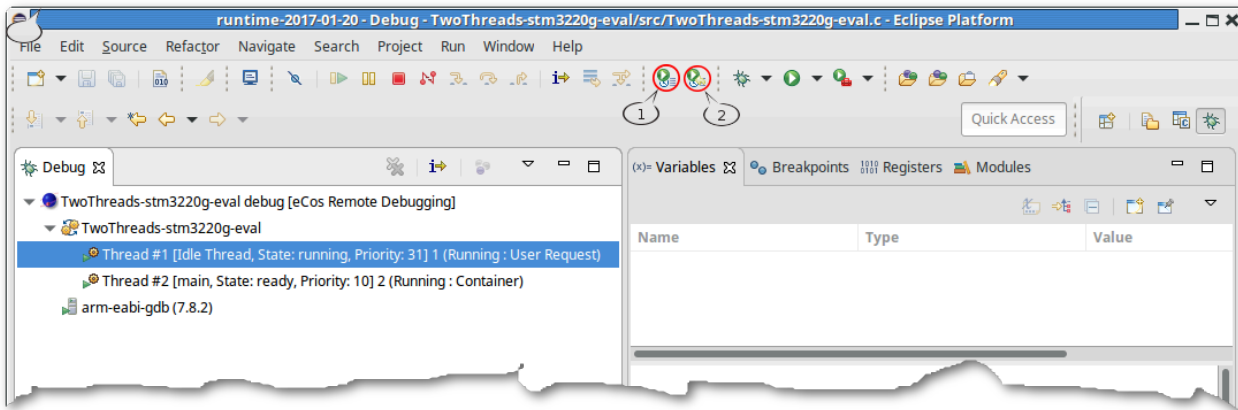
Extracting the profiling data from the target

To capture the profiling data, select the application's process or one of the application's threads within the `Debug` Window of the *Debug Perspective*, and press the `Take Profile Snapshot` button called out as *1* in Figure 7.6, "Capture and Clear Profiling Data".

This will extract the profiling data from the target, pausing and resuming execution if necessary, and save it to a file within the project explorer tree in the same directory as the binary executable. The file will have a `.gmon` extension with the date and time of the snapshot as the base name, allowing multiple snapshots to be taken and saved at different times.
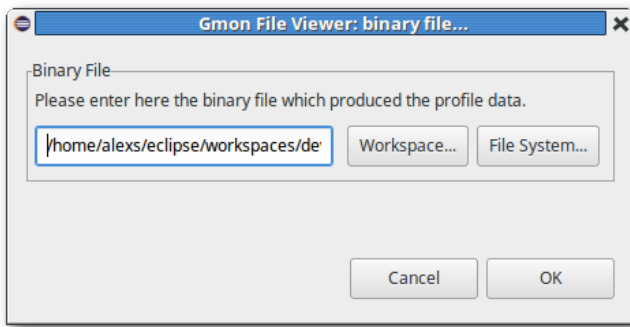
> **Note**
>
> If you are using a software monitor (e.g. *RedBoot*) to debug your application and not a hardware debugger, and if the application is not already paused, the snapshot may not immediately be taken but may be delayed until the software monitor is active (e.g. when performing diagnostics output). This is also dependant on the eCos configuration.

Clearing the profiling data from the target

To clear the profiling data, press the `Reset Profile Data` button called out as *2* in Figure 7.6, "Capture and Clear Profiling Data" when either the application's process or one of the application's threads within the `Debug` Window of the *Debug Perspective* is selected. Similar to the capturing of the data, this will pausie and resume execution of the application if necessary.

Display the profiling data using the GProf plugin

To display the profiling data in the *GProf plugin*, double click on the `.gmon` filename corresponding to the data capture you would like to display, or right click on the filename and select `Open` or `Open with` → `Gprof editor`. If you have multiple binaries, the popup illustrated in Figure 7.7, "Gmon File Viewer: select binary" will appear prompting you to select the binary used to generate the profiling data. If prompted, select the application binary used to generate the profile data.

Figure 7.7. Gmon File Viewer: select binary

Select the **gprof** tab as illustrated in Figure 7.8, "Gprof tab window".

Figure 7.8. Gprof tab window

The Gprof view shows how much execution time is consumed by each part of the application and also provides call graph infomation for each function. The buttons available are:

- "Show/Hide columns" button allows you to select which columns to display.

- "Export to CSV" button allows you to export the GProv result as a CSV text file.

- "Sorting" button allows you to choose the columns, their priority and their ordering by which the data is sorted.

- "Sort samples per file" button displays the GProf result sorted by file.

- "Sort samples per function" button displays the GProf result sorted by function.

- "Sort samples per line" button displays the GProf result sorted by line.

- "Display function call graph" button displays the GProf result as a call graph.

- "Switch sample/time" button allows you to switch the data between sample and time results.

- "Create chart" button allows you to create a chart (Bar/Vertical bar/Pie) from the data selected for the data of selected columns.

For further documentation, please refer to the GProf User Guide.

Chapter 8. Hints and Troubleshooting

Keyboard shortcuts

Pressing `Ctrl+Shift+L` within Eclipse will display a popup window with all the current keyboard bindings. If you press `Ctrl+Shift+L` again, you will be taken to the *Keys* preferences page from which you can change or set keyboard bindings for various *Commands* within Eclipse and its installed plug- ins.

Build Automatically

If this option is set and you change, add or delete a file within a project, the builder will automatically attempt to build that project in the background. This option is set by default and can be found in the *C/C++* perspective within the *Project* sub-menu.

Build Order

The order in which projects are built is important. For example, eCos configuration projects must be built before the eCos application projects or static libraries which refer to them are built. Normally the Eclipse Workbench will build all pre-requisites first but occassionally it may be desirable to build projects in a specific order. The Workbench allows users to explicitly define the order in which projects are built through the *Build Order* Window which can be reached through `Window` → `Preferences` → `General` → `WorkSpace` → `Build Order`. The build order is applied for both building the entire workspace or for a group of projects.

Clean Project

The *Clean...* option within the *Project* sub-menu will remove all artifacts created by Eclipse for all or just selected projects. If you have the *Build Automatically* option set, the artifacts of the chosen projects will be rebuilt automatically following a clean. If the option is off, clean will remove the artifacts and stop, allowing you invoke *Build* manually later.

The *Clean* option within a project context (i.e. right-click a project) will automatically clean the selected project and *all* the projects referenced. For example, if this option is selected in an eCos application project that also refers to a static library, all of the application project, eCos configuration project and static library project will be cleaned. If you also have the `Project` → `Build Automatically` option set, all those projects will be rebuilt after they have been cleaned.

Eclipse and the *eCos Configuration Tool*

As described in the "eCos configuration projects" section, the eCos Configuration Tool is used by Eclipse as a standalone tool to create and edit eCos configurations. However, when editing an eCos configuration with the *eCos Configuration Tool* in this manner, certain features of the *eCos Configuration Tool* are disabled which will tie the tool to the specific eCos configuration file that is being edited. This is known as the *Eclipse Lockdown Mode* of the *eCos Configuration Tool*. For example, you cannot *Save As* or *Open* other configuration projects, nor can you change *Profiles*, alter the eCos Repository or modify the *Path* to the host or GNU tools. This ensures that the Eclipse workspace as well as the eCos configuration and its elements remain consistent with the state Eclipse believes the eCos configuration project to be in.

When the *eCos Configuration Tool* is started normally from the command line, the Desktop or a Desktop menu, it is run in its normal *Default* mode with all its normal functionality available to you. However, you should not use the *eCos Configuration Tool* in this mode to edit or create eCos configurations, or build eCos configurations within an Eclipse Workspace as this may result in unexpected consequences and inconsistent behaviour. Always use Eclipse to start the *eCos Configuration Tool* when you wish to modify an eCos configuration file that is part of an Eclipse eCos Configuration Project.

Disable optimizations

Compiler optimization can result in code being re-ordered, particularly at assembly-level, to provide fast and efficient code, or just smaller code sizes. This makes the job of source- level debuggers to match assembly-level instructions with lines of source code at times difficult, if not impossible, as single stepping of source code is often performed at assembly-level. It is unsurprising then that your source-level debugger may jump around when executing a seemingly sequential set of source code operations, or sometimes the

source-level debugger may even fail to hit a breakpoint or stop at an apparently arbitrary location, when debugging optimized code.

Figure 8.1. Find CYGBLD_GLOBAL_CFLAGS in configuration
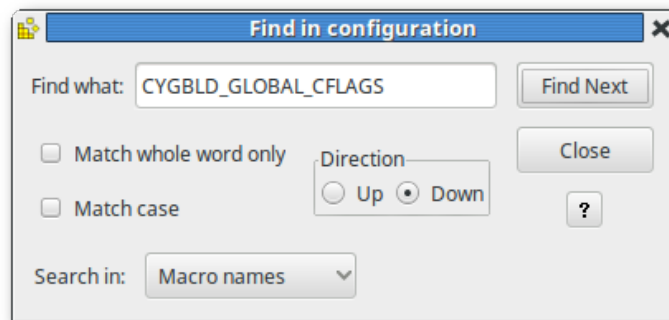
To follow the logical source-code progression of execution of an eCos application you must turn off optimizations for both your eCos application and the eCos library, or project, against which your application is linked. To do this, edit your active configuration from within the configuration project, and search for the CYGBLD_GLOBAL_CFLAGS macro using the search dialog, *Edit* → *Find* (**Ctrl+F**), illustrated in Figure 8.1, "Find CYGBLD_GLOBAL_CFLAGS in configuration" to take you to the generic *Global compiler flags* option and change the **-O2** flag to **-O0**. For additional debug support, developers are also encouraged to add the CYGPKG_INFRA *Infrastructure* package to their configuration: *Build* → *Packages* (**Ctrl+P**) (add CYGPKG_INFRA into the keywords field and ensure the *Infrastructure* package appears under *Use these pachages*, followed

by *OK* - See Figure 8.2, "Packages Dialog - Add CYGPKG_INFRA") and to enable the CYGPKG_INFRA_DEBUG *Asserts & Tracing* option within the configuration tree.

Figure 8.2. Packages Dialog - Add CYGPKG_INFRA
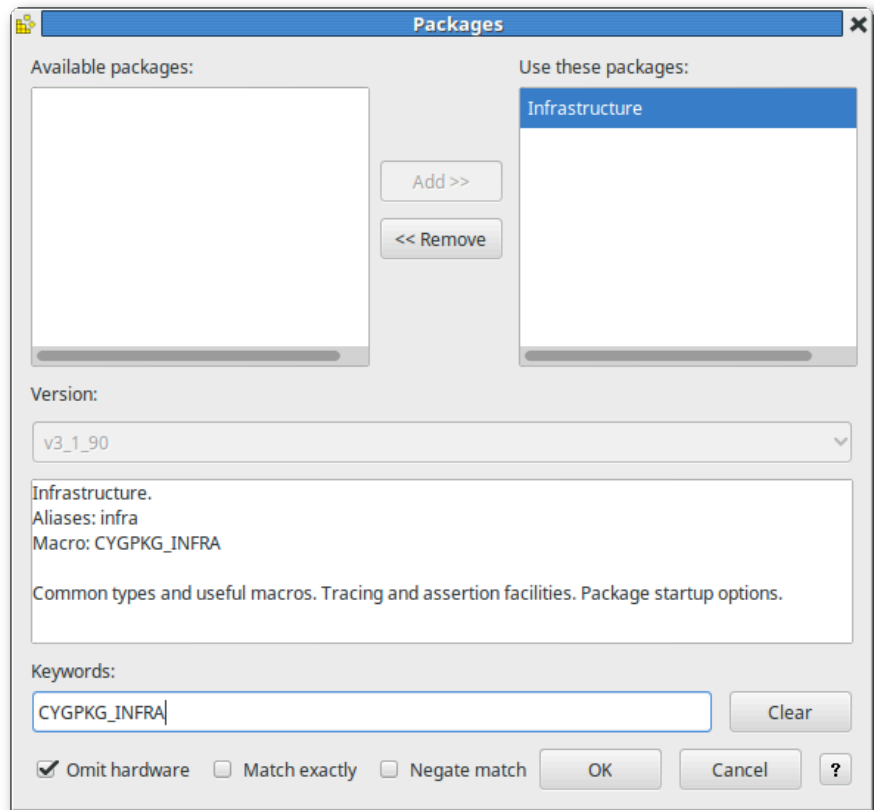
Error in final launch sequence, Failed to execute MI command

Slow host PCs or remote target communications can occasionally result in a *GDB* timeout. This in turn can result in errors such as "*Error in final launch sequence, Failed to execute MI command*" error within Eclipse/CDT, or debug sessions locking up and becoming unresponsive.

This issue can be resolved by extending the timeout used by GDB for remote communications. Use a GDB command file containing the line below:

```
set remotetimeout 30
```

• When using an *eCos Hardware Debug* configuration, within the "Debugging" tab, select the GDB command file within the "GDB Command File" section; or

• When using an *eCos Remote Debug* configuration, within the "Debugger" tab and under the "Debugger Options" in the "Main" tab, enter the GDB command file alongside the "GDB Command File" option.

You should now be able to debug without the error occurring.

Chapter 9. Upgrading Eclipse, eCosPro CDT plug-in and Eclipse Workspaces

There are three points to consider before you upgrade to a newer version of Eclipse:

1. Upgrading the Eclipse Plugins.

   This is the most important point to consider as Eclipse provides the framework for the IDE but the plug-ins provide the functionality of the IDE. You must ensure the plug-ins you use are compatible with the newer version of Eclipse, and if not whether newer versions of the plug-ins are available which are compaible with the newer version of Eclipse. If the plugins you use are are incompatible or do not have newer versions available which are incompatible you will need to either accept the loss of functionality which the upgrade make bring or forgo the upgrade until compatible versions become available.

2. Upgrading the Eclipse installation.

   Generally speaking, an upgrade of Eclipse consititutes a completely new installation of Eclipse rather than upgrades of individual components to an existing installation. Older installations of Eclipse may be run side-by-side with newer installations, although common workspaces may *not* be shared.

3. Upgrading the Eclipse workspace.

   Before performing any Eclipse upgrade, you *must* take a backup of your workspaces, or create a copy of the workspace for the new version of Eclipse. Once a workspace has been upgraded by a newer version of Eclipse, you will never be able to use it again with an older version of Ecplise.

Upgrading Eclipse

Upgrading Eclipse is generally a fairly simple task involving a new installation of Eclipse rather than the replacement or upgrade of an existing installation. Please refer to the installation instructions for the new Eclipse distribution you have been provided.

eCosCentric customers are provided with an Eclipse installer which allows the developer to create a new Eclipse installation, including the eCosPro CDT plug-in, for developing eCosPro applications, or installs the eCosPro CDT plug-in into an existing Eclipse installation provided by a third- party or downloaded and installed from the Eclipse website. Please refer to the installation instructions provided with your eCosPro distribution to install either a new Eclipse version, including the eCosPro CDT plug-in. These instructions include information on how to install the eCosPro CDT plug-in into third-party Eclipse installations.

Upgrading the eCosPro CDT plug-in

You may upgraded the eCosPro CDT plug-in within an Eclipse installation using the installer provided to you by eCosCentric. Alternatively, you can update online by following the instructions at: http://www.ecoscentric.com/eclipse/.

Upgrading the Eclipse Workspace

⚠️ **Important**

1. Before upgrading to a newer release of Eclipse from an older release it is extremely important to take a backup of the entire workspace directory. Once a workspace has been upgraded, it is no longer usable by older versions of Eclipse.

2. Do not copy or move the workspace directory because it may contain metadata with absolute pathnames, making them invalid if the workspace is copied or moved elsewhere.

If you previously started the older version of Eclipse using the `-data` argument to specify the name and location of the workspace directory, to upgrade the workspace you only need to start the newer Eclipse with the same `-data` argument.

If you did not provide the **-data** argument to Eclipse, the workspace chooser dialog allows you to choose the location of your workspace. Select *Browse* and navigate to the location of the workspace then select *OK*. Check the *Use this as the default and do not ask again* box for Eclipse to remember your decision. Finally select *OK* to accept the workspace. Eclipse will now upgrade your workspace.

Warning

Do not store your workspace inside the Eclipse install directory as it will make difficult to upgrade to a newer version of Eclipse.