

Contents

- 1 Introduction 5**
 - 1.1 Scope5**
 - 1.2 Audience5**
 - 1.3 Contact Information, Support.....5**
 - 1.4 Product Overview5**
 - 1.5 Document Organization5**
 - 1.6 Text Conventions6**
 - 1.7 Related Documents6**
 - 1.8 Document History6**
- 2 Installing the Development Environment..... 7**
- 3 Creating a project..... 11**
- 4 Using the debugger 18**
 - 4.1 How to use OOCDDLink18**
 - 4.2 How to use SAM-ICE24**
 - 4.3 How load and debug an application in the target’s RAM26**
 - 4.4 How debug an application in the target’s flash.....30**
- 5 Using U-Boot 32**
 - 5.1 How to flash an application34**
 - 5.2 How to start an application.....35**
 - 5.3 How to automatically start an application from flash in the boot sequence35**



1 Introduction

1.1 Scope

This guide will explain how to develop and extend your first application on the GE863-PRO³.

In the following sections, the term “host” will refer to the computer where the development environment is running, while we’ll refer to the GE863-PRO³ as the target.

1.2 Audience

This User Guide is intended for software developers who develop applications on the ARM processor of the module.

1.3 Contact Information, Support

Our aim is to make this guide as helpful as possible. Keep us informed of your comments and suggestions for improvements.

For general contact, technical support, report documentation errors and to order manuals, contact Telit’s Technical Support Center at:

TS-EMEA@telit.com or <http://www.telit.com/en/products/technical-support-center/contact.php>

Telit appreciates feedback from the users of our information.

1.4 Product Overview

The GE863-PRO³ module contains a fully featured GSM/GPRS communications section, compatible with the other Telit GSM/GPRS modules, but also incorporates a standalone ARM9 CPU and memories, dedicated to user applications.

This eliminates the need for an external host CPU in many applications, bringing true real-time and multi tasking capabilities to an embedded module.

1.5 Document Organization

This manual contains the following chapters:

- “Chapter 1 Introduction” provides a scope for this manual, target audience, technical contact information, and text conventions.



Development Environment User Guide

1vv0300775a Rev. 1 - 23/04/08

- “Chapter 2 Installing the Development Environment” describes the general information on how to install development environment that will be used.
- “Chapter 3, Creating a project” describes how to start a project in C or import an existing project, including examples, where relevant.
- “Chapter 4, Using the debugger” provides info on how to setup and use the debugging environment for OOCdLink and JTAG emulator SAM-ICE
- “Chapter 5, Using U-Boot” provides instructions on how to use the U-boot to start or flash the application. All U-boot commands definitions can be found in U-Boot Software User Guide.

1.6 Text Conventions

This section lists the paragraph and font styles used for the various types of information presented in this user guide.

Format	Content
Courier	Example commands and responses from U-Boot.

1.7 Related Documents

The following documents are related to this user guide:

- [1] TelitGE863PRO3 Bootloader Recovery Application Note 80000nt10012a
- [2] TelitGE863PRO3 U-Boot Software User Guide 1VV0300777
- [3] TelitGE863PRO3 EVK User Guide 1vv0300776
- [4] TelitGE863PRO3 Hardware User Guide 1vv0300773a
- [5] TelitGE863PRO3 Software User Guide
- [6] TelitGE863PRO3 Product Description 80285ST10036a

All documentation can be downloaded from Telit’s official web site www.telit.com if not otherwise indicated.

1.8 Document History

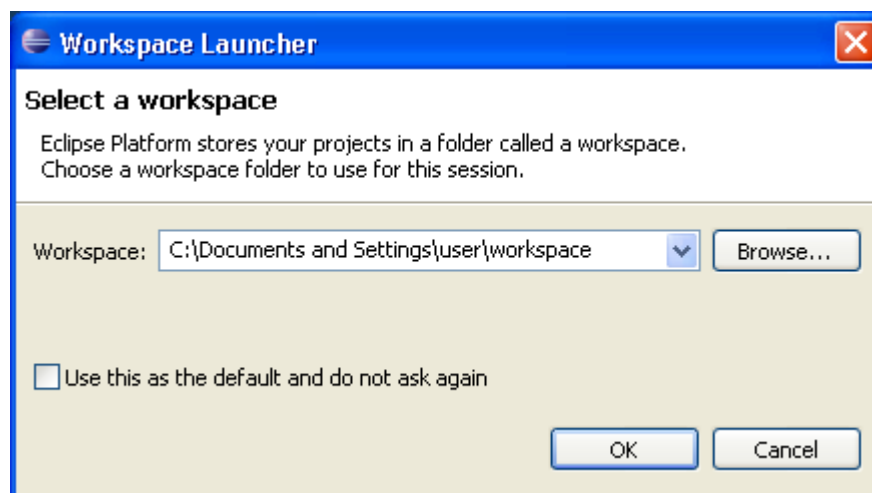
Revision	Date	Changes
ISSUE #0	25/01/07	First release
ISSUE #1	23/04/08	Added SAM-ICE is a JTAG emulator description in paragraph 4.2 Memory map paragraph moved to SW User Guide



2 Installing the Development Environment

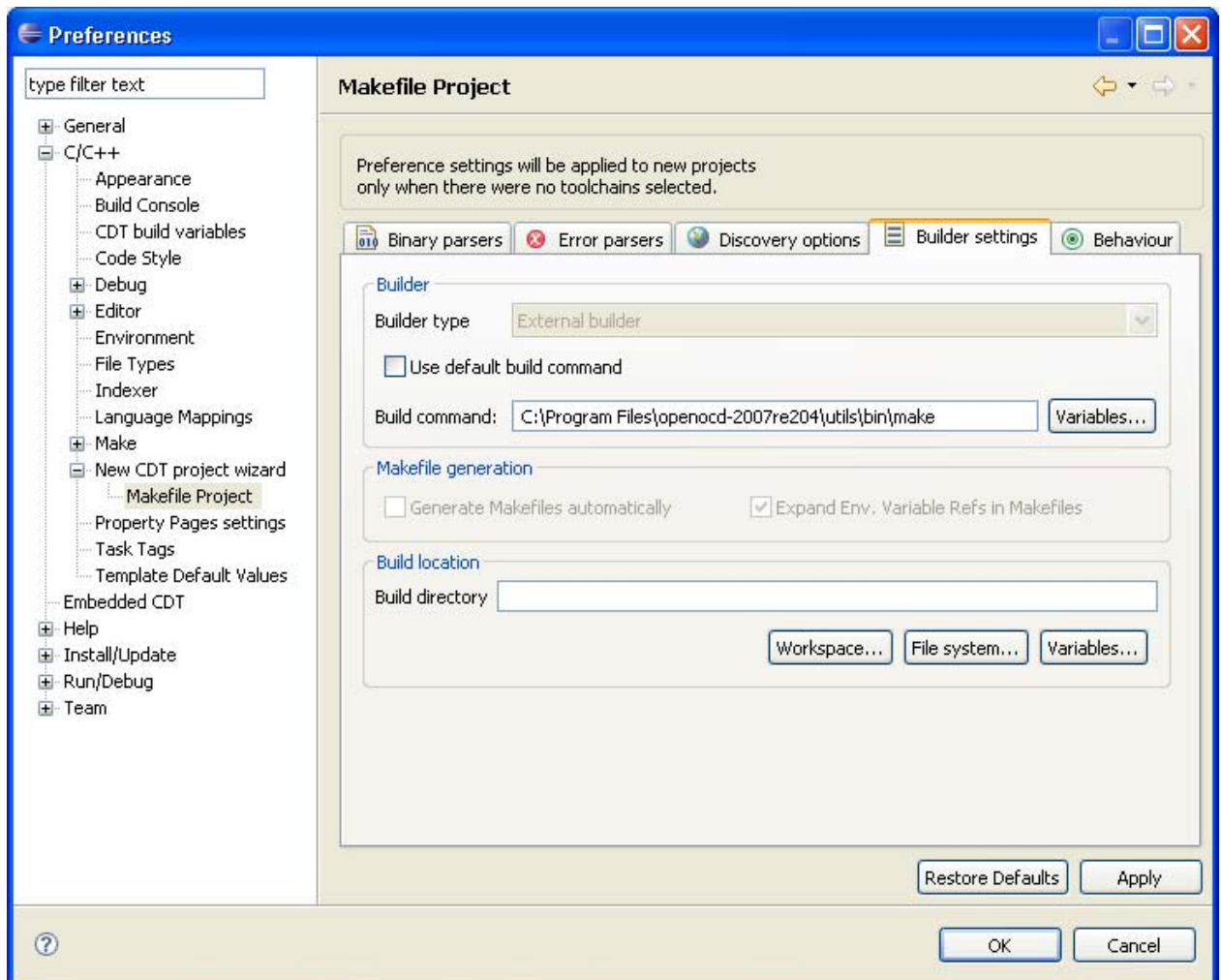
In order to set up the development environment the following steps should be taken:

- Go to <http://www.yagarto.de/> and click on the **Download** section.
- Download and install (with the default options) the following software packages, in the order shown below:
 - Yagarto GNU ARM toolchain.
 - Yagarto Integrated Development Environment.
 - Open On Chip Debugger.
- Open the IDE (**Start menu** → **All Programs** → **YAGARTO IDE** → **Eclipse Platform**) and choose a location in the filesystem for creating the workspace



- When the workspace has been created click on the **Go to the Workbench** icon.





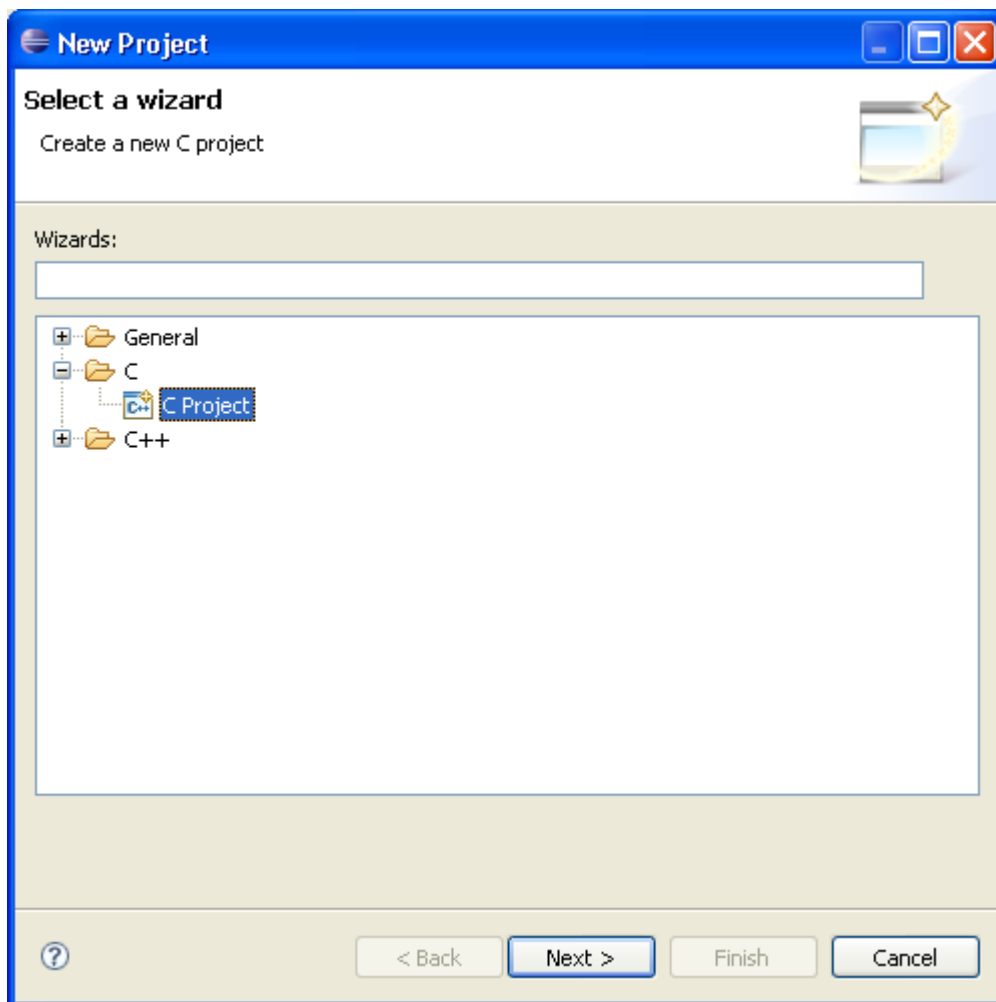
Now you are ready to develop your first application for the PRO³.



3 Creating a project

When you start creating a new project follow the steps below:

- Open the IDE (**Start menu** → **All Programs** → **YAGARTO IDE** → **Eclipse Platform**).
- Open the menu **File** → **New** → **Project**.
- Open the **C** folder and choose the **C Project** entry, then click **Next**.



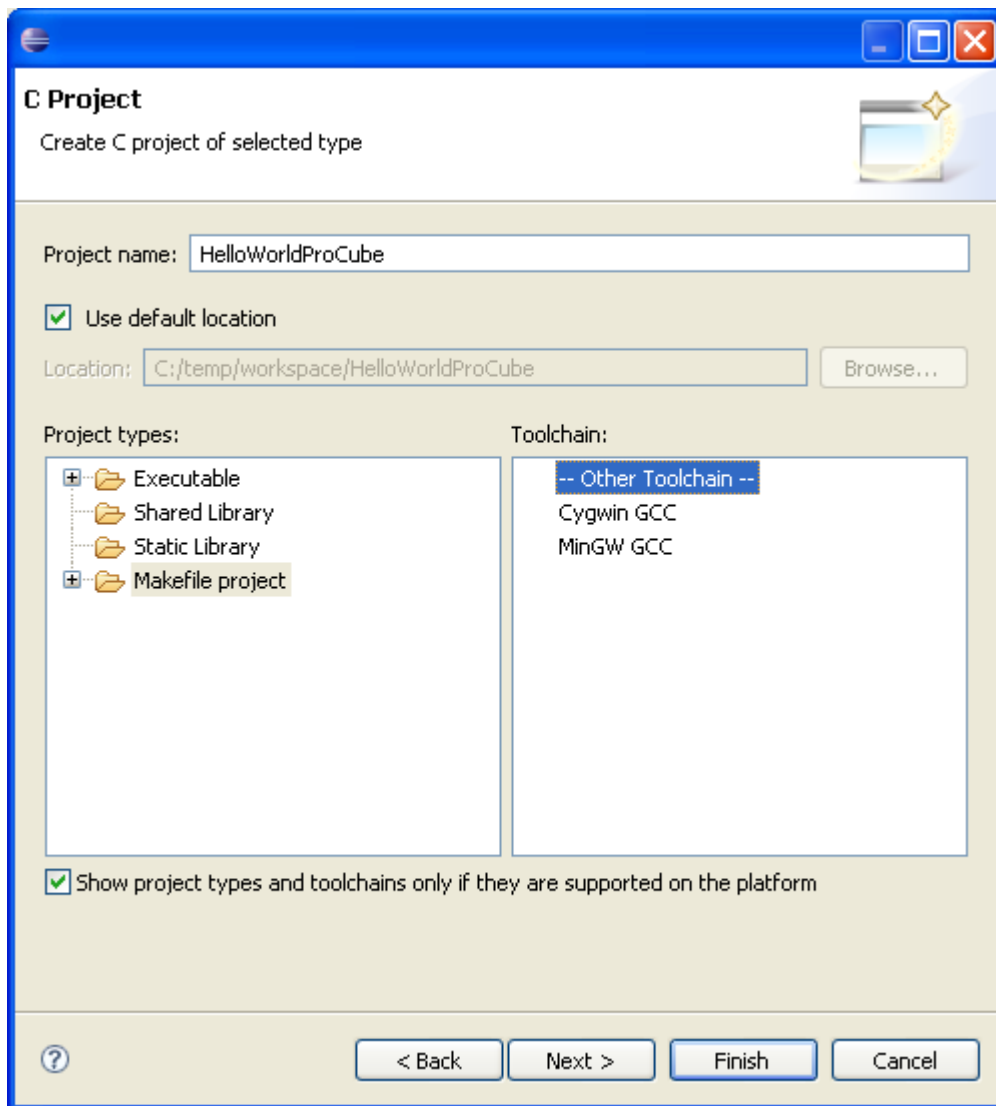
- Fill in the **Project Name** text field; in the **Project Types** area select the **Makefile Project** folder.



Development Environment User Guide

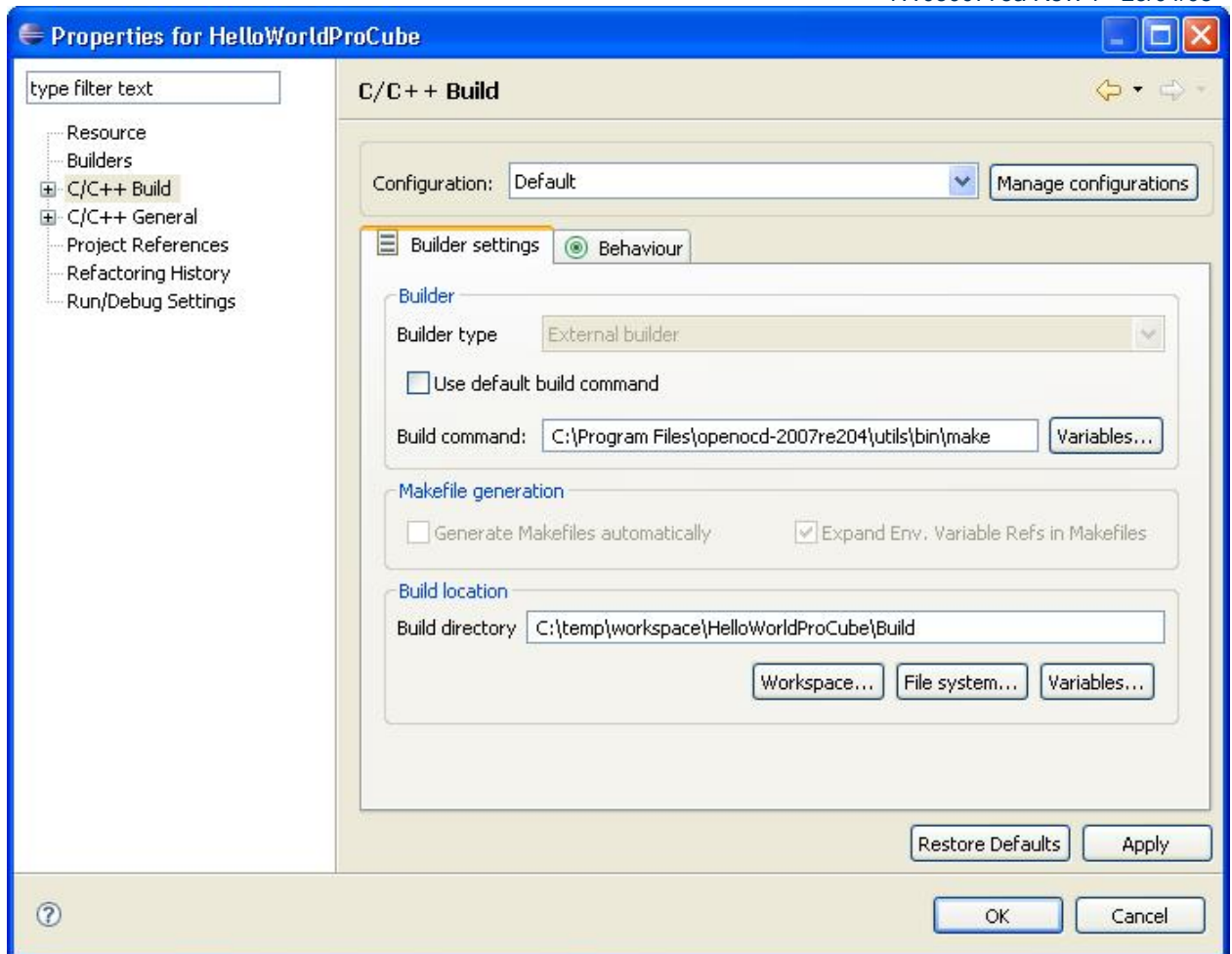
1vv0300775a Rev. 1 - 23/04/08

- In the **Toolchain** area choose the **Other Toolchain** entry and then click **Finish**. If the window entitled **Open Associated Perspective** comes up answer **Yes**. This will open the IDE's C/C++ perspective.



- Now you should see the workspace with the newly created project: it will be empty because currently there are no files.





- Go to **Project** → **Build All** and, if there are no errors, you will find the binaries in the directory previously inserted in the project preferences.



4 Using the debugger

To debug an application for the Pro³ you can use different hardware emulators (such as SAM-ICE, J-Link, Multi-Ice, Lauterbach...). The software configuration depends on the type of hardware you use. In this paragraph we'll explain how to setup the debugging environment using the OOCdLink emulator: the general steps are common, you only have to install the correct drivers and configure them for your hardware.

Regardless the emulator you use, certain files are required to configure your environment. In the host system you should create a directory called Ge863Pro3\Debug and copy in it the files provided in the Telit package:

- GE863-PRO3_ft2232.cfg
- GE863-PRO3_debug_commands.script
- GE863-PRO3_debug_commands_boot.script

(it is not mandatory to put the files in that directory, but if you choose another directory please remember to modify also the paths in the rest of this section)

The first file will serve to configure the gdb server while the other two offer two different possibilities to debug an application, as explained later.

4.1 How to use OOCdLink

To setup the debugging environment for using OOCdLink follow these steps:

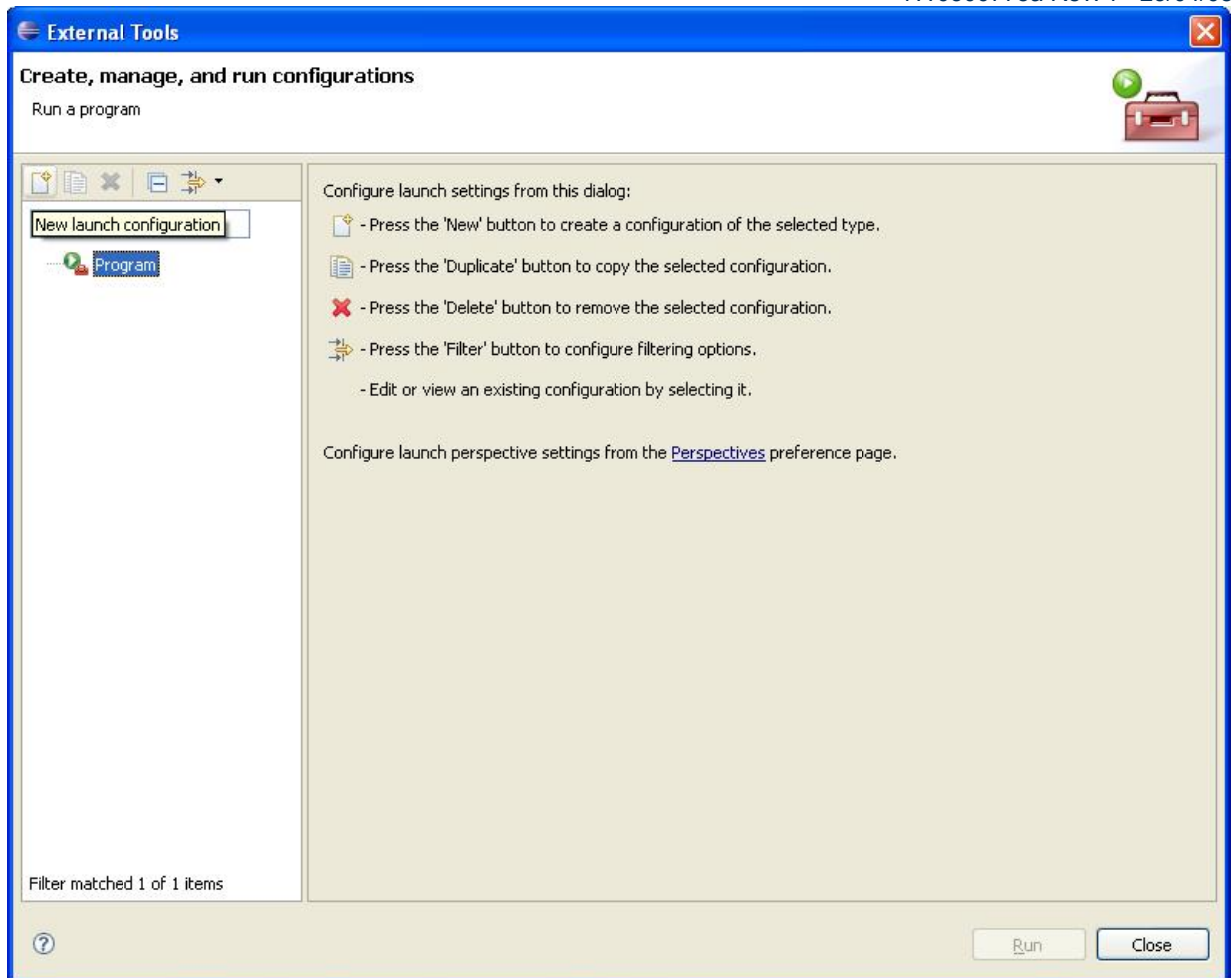
- Download the OOCdLink drivers (you can find them at the address http://www.ioernonline.de/dw/doku.php?id=projects:oo.cdlink:2_oo.cdlinks) and unzip the archive.
- Plug the USB connection cable of OOCdLink in the host system; you will be asked for the drivers: click on the **No, not this time** option button and then **Next**.





- Select the **Install from a list or specific location (Advanced)** option button and then click **Next**.





- Fill in the text field called **Name** with the value “Open On Chip Debugger”; select the tab **Main** and fill in the text field **Location** with the executable of the gdb server installed with the package Open On Chip Debugger. Fill in the text field **Working Directory** with the previously created directory (C:\Ge863Pro3\Debug); fill in the text box **Arguments** with the following line:

```
-f GE863-PRO3_ft2232.cfg
```

First Click on the **Apply** and then click on **Close**.

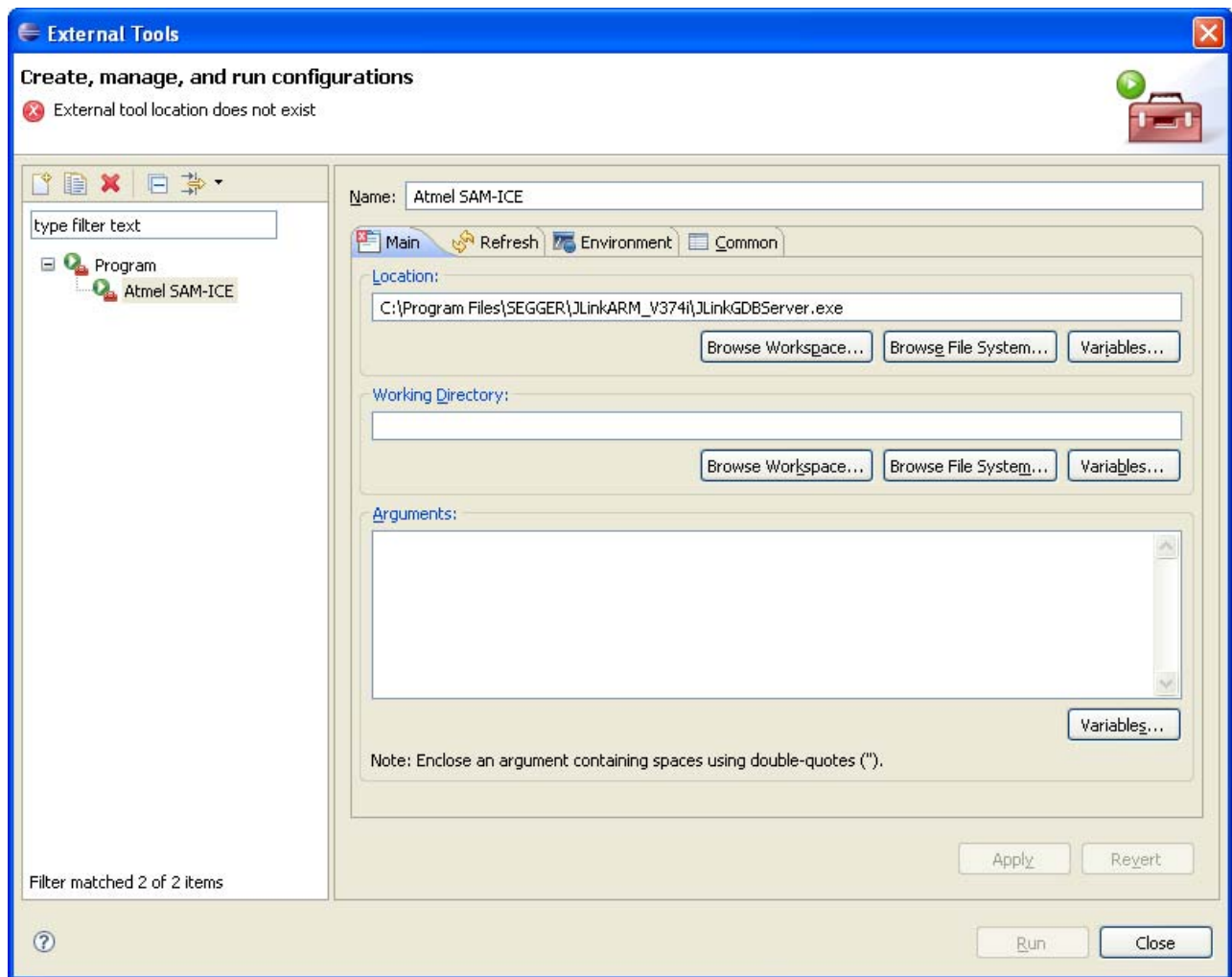
Note: the file GE863-PRO3_ft2232.cfg is necessary to instruct the gdb server to use the OOCLink: if you have another emulator, you need to modify that file in a propriety way.



Development Environment User Guide

1vv0300775a Rev. 1 - 23/04/08

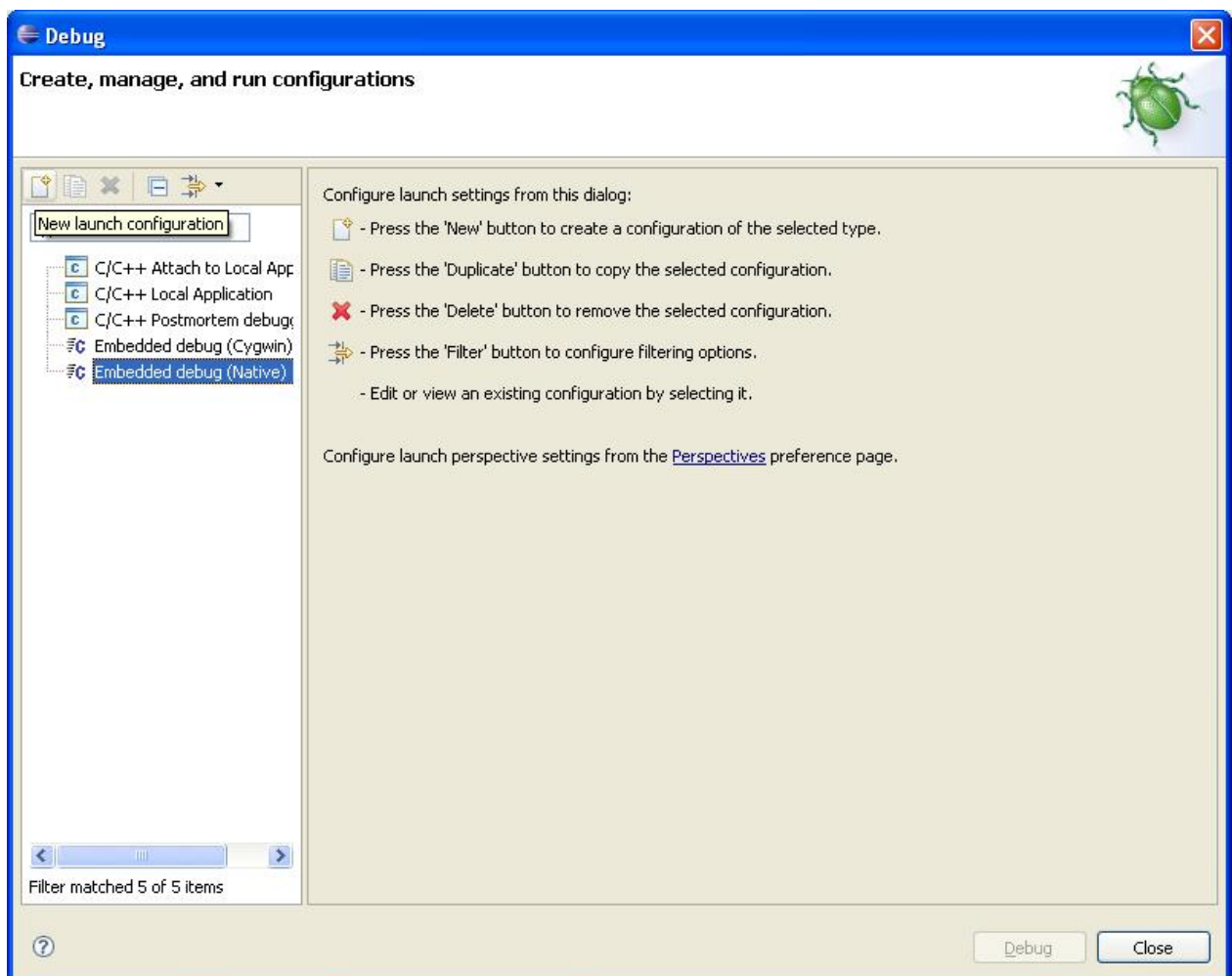
- Fill in the text field called **Name** with the value “Atmel SAM-ICE”; select the tab **Main** and fill in the text field **Location** with the executable of the gdb server installed with the J-Link ARM toolkit.
- Click on **Apply** and then click on **Close**.

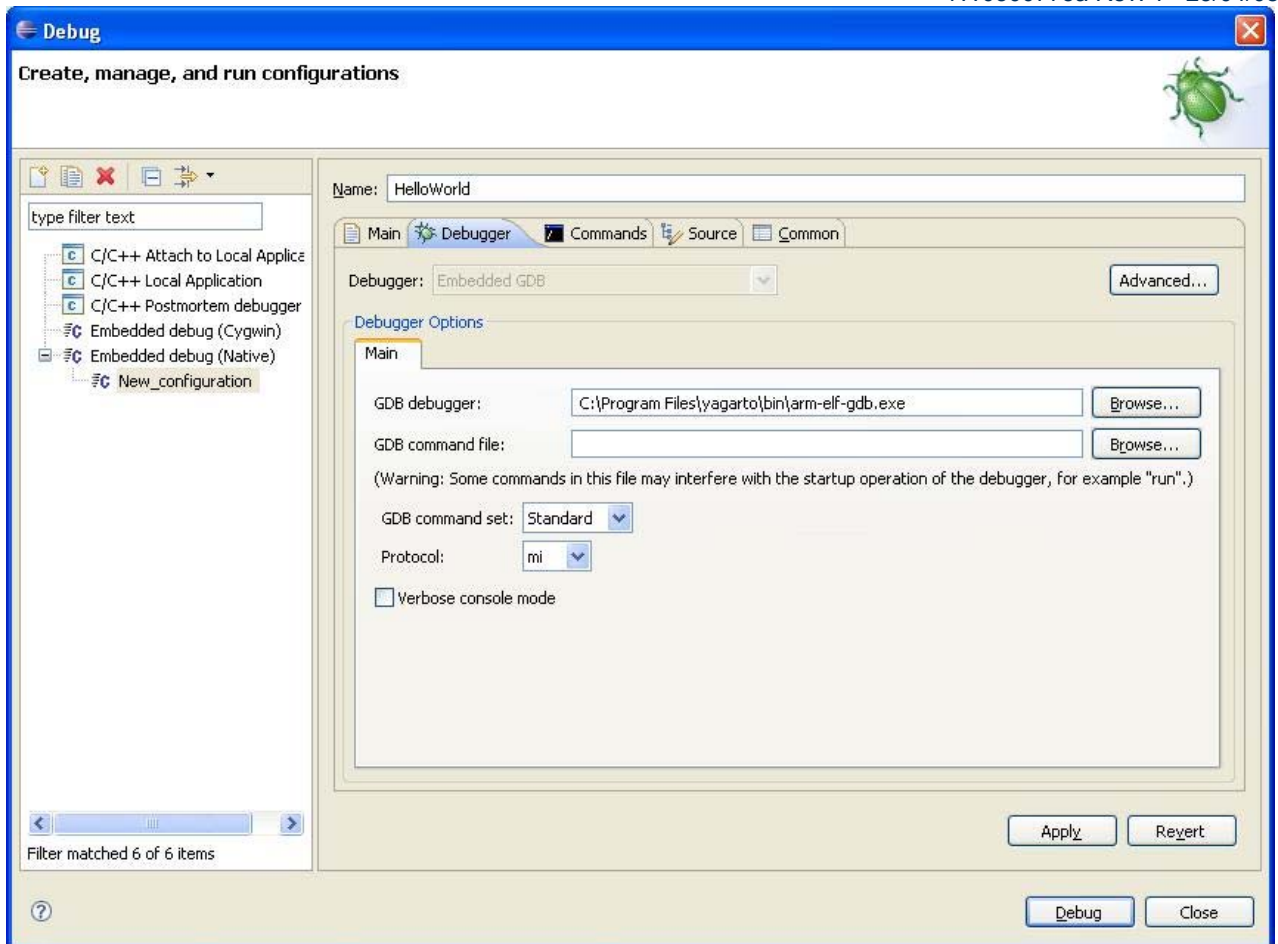


4.3 How load and debug an application in the target's RAM

The simplest method to debug an application is to load it in the target's RAM via JTAG and debug it from there. In this way you can debug your application without having to download it manually in the target every time you make a modification in the source code. Now we will instruct the IDE to run the gdb debugger with the correct parameters in order to debug the application built in the previous section.

- Go to the menu **Run** and click on **Open Debug Dialog**.
- On the left side of the window select the entry **Embedded debug (Native)** and click on the **New Launch Configuration** icon.

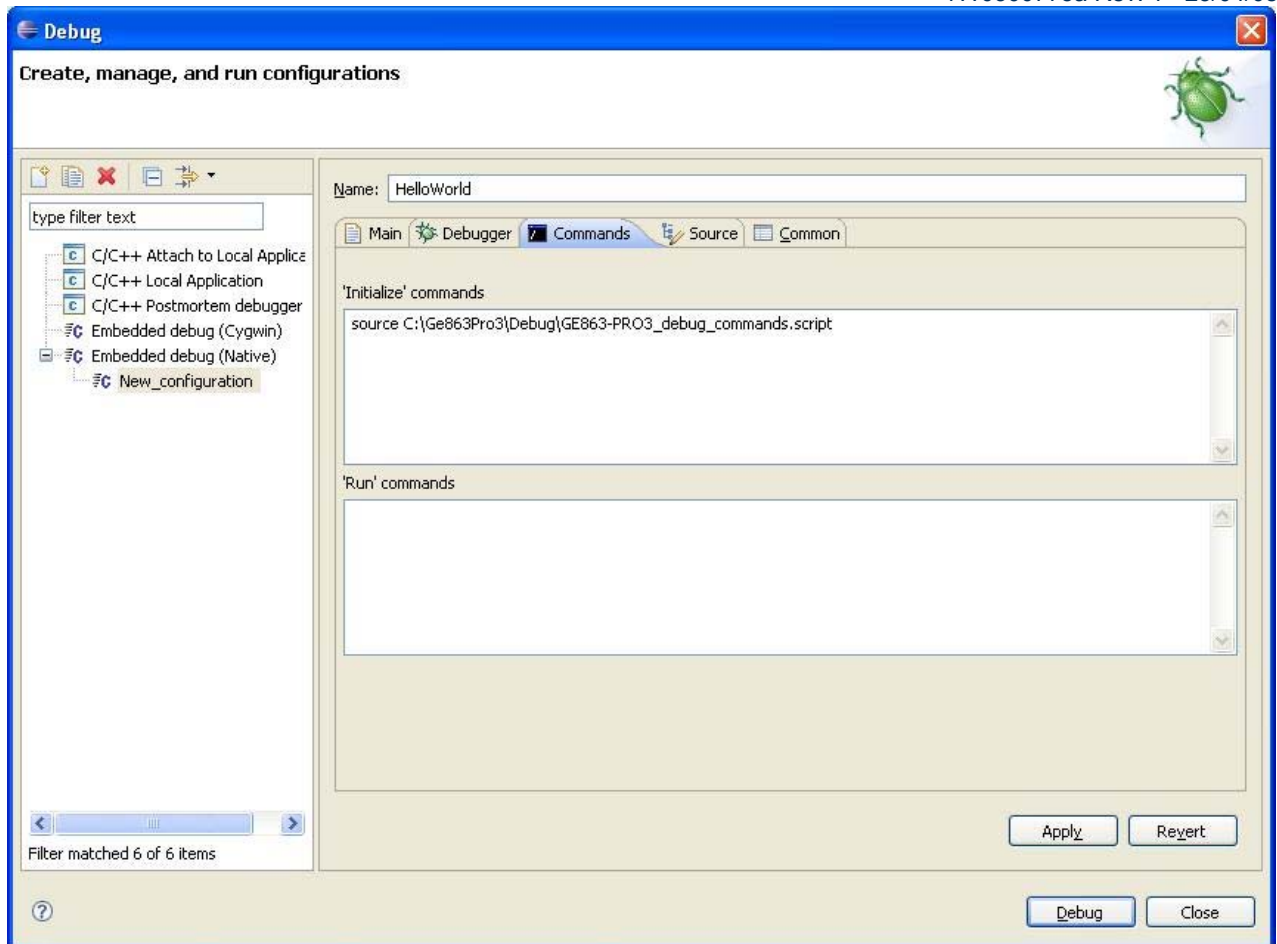




- Select the tab **Commands** and fill in the text field '**Initialize** commands with a line that will make gdb execute at startup the commands contained in the file GE863-PRO3_debug_commands.script.

First click on **Apply** and then click on **Close**.





Now that the debug environment is set up, you can start debugging your application. Connect OOCdLink to the JTAG port of the target. It is convenient to open the IDE's debug perspective, so you can follow the processes that are being performed in any moment. In order to do so, go to **Window** → **Open Perspective** → **Debug** and if you want to return to the previous perspective, just go to **Window** → **Open Perspective** → **C/C++**.

The gdb server you have previously configured in the **External Tools** dialog box must be started first. In order to do that you should select the menu entry **Run** → **External Tools** → **Open On Chip Debugger**. The related entry will appear in the Debug view.

The next step is launching gdb itself by going to **Run** → **Open Debug Dialog**. On the left side of the window select the icon **HelloWorld**, corresponding to the gdb configuration you just made, and click on **Debug** (from now on, you can start the debugger with this configuration also selecting the related entry from the menu **Run** → **Debug History**). Some additional entries should appear in the Debug view showing the status of the process you just started. Wait until gdb initializes the target and loads in it the application (it may take a few seconds). The target should halt the execution at the first line of your main() function: from there, you can debug your application setting breakpoints, stepping through the code and so on. Note that if there is any optimization enabled in the compiler options, debugging can have some difficulties since the program may not follow the execution flow that you are expecting.



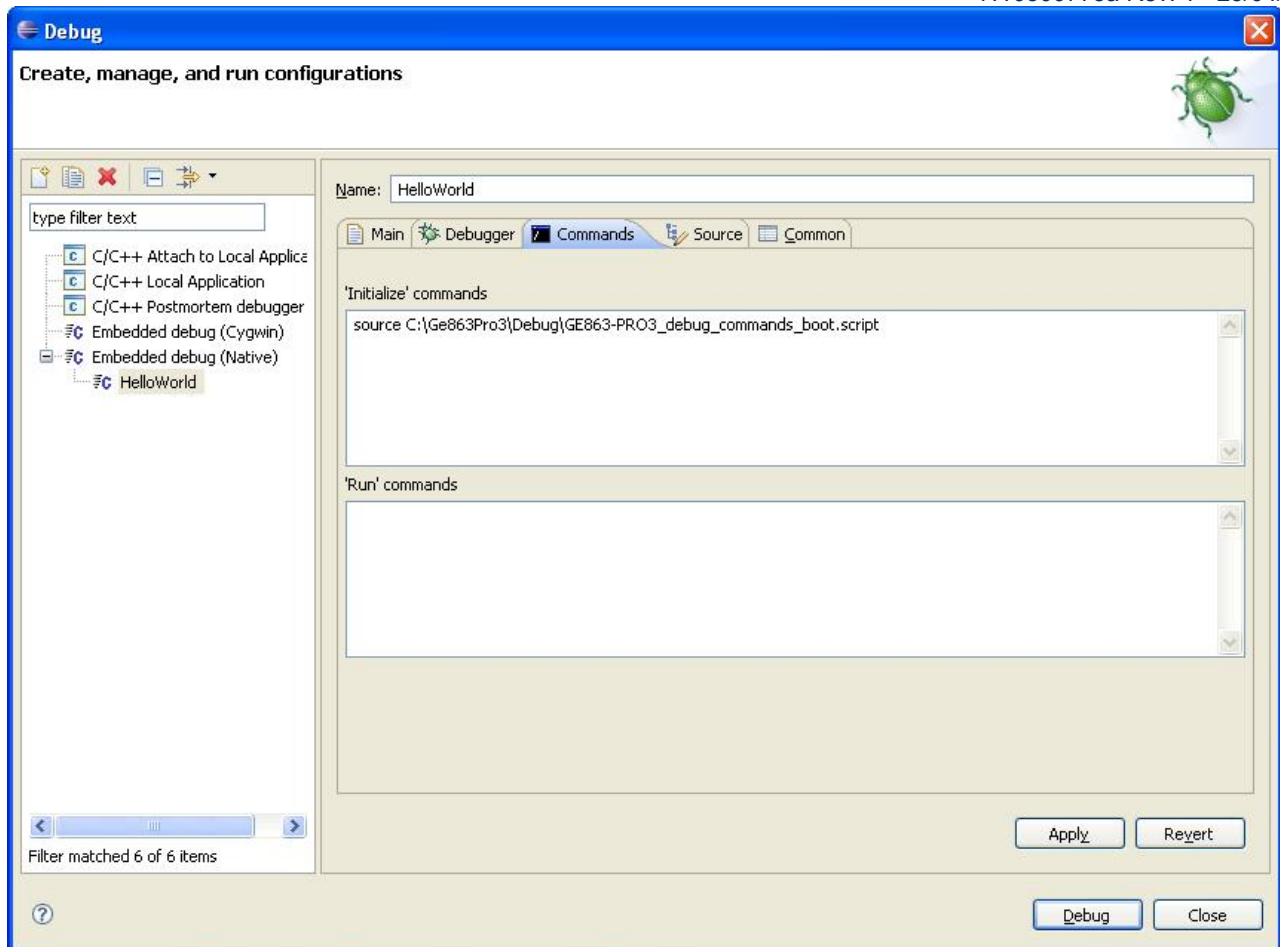
If you want to terminate the execution of any process you started from within the IDE, just select the related entry in the Debug view and click on **Run** → **Terminate**.

4.4 How debug an application in the target's flash

In the previous paragraph we described a debugging method in which the binary image of the application is downloaded through JTAG directly in the target's RAM and executed from there. Another way to debug your application is to transfer the image (the bin file created in the Build directory of your project) into your target's flash and instruct U-Boot to load it in RAM and execute it. You can debug the application once it is started by U-Boot. With this technique you can examine your application's behavior in the same conditions as when the target executes a normal boot sequence, but you have to download manually the image into the target memory every time you modify it. Read the next section for instructions on how to flash an application and instruct U-Boot to execute it automatically at startup.

To setup the debugger, you can either modify the configuration you made in the previous paragraph, or create a new one. The only difference is that the debugger must execute a different script file at the start-up (GE863-PRO3_debug_commands_boot.script). To use the new file, you have to change the text contained in the text field '**Initialize**' **commands** in the tab **Commands** of the Debug dialog box and click on **Apply**.





Now you can debug your application as described in the previous paragraph. First start the gdb server (if not already started) and then start the debugger with the new configuration. In this case, if you have an open terminal emulator in the host and connected to the target (for details see the next section), when the debugger is started you should see U-Boot output messages. When U-Boot starts your application, the execution is halted at the beginning of the main() function and from there you can begin to debug the application. It is recommended to configure U-Boot to load and execute automatically your application during the boot sequence. For more detailed instructions on how to do that please read the next section.



