Development Environment User Guide

For simple C applications in GE863-PRO3
1vv0300775a Rev. 2 - 10/09/08

Making machines talk.

DISCLAIMER

The information contained in this document is the proprietary information of Telit Communications S.p.A. and its affiliates (“TELIT”).
The contents are confidential and any disclosure to persons other than the officers, employees, agents or subcontractors of the owner or licensee of this document, without the prior written consent of Telit, is strictly prohibited.

Telit makes every effort to ensure the quality of the information it makes available. Notwithstanding the foregoing, Telit does not make any warranty as to the information contained herein, and does not accept any liability for any injury, loss or damage of any kind incurred by use of or reliance upon the information.

Telit disclaims any and all responsibility for the application of the devices characterized in this document, and notes that the application of the device must comply with the safety standards of the applicable country, and where applicable, with the relevant wiring rules.

Telit reserves the right to make modifications, additions and deletions to this document due to typographical errors, inaccurate information, or improvements to programs and/or equipment at any time and without notice. Such changes will, nevertheless be incorporated into new editions of this application note.

Applicable Products

Product	Part Number
GE863-PRO[3]	3990250691

Contents

1 Introduction

1.1 Scope

This guide will explain how to develop and extend your first application on the GE863-PRO^3.

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-PRO^3 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 Open Source Licenses

Telit Development Environment is made up of different Open Source Software licensed as follows.

1.4.1 Yagarto

- Yagarto GNU ARM toolchain: some of the software falls under the GNU General Public License (GPL).
- Integrated Development Environment: all of the software like the Eclipse Platform Runtime Binary, Eclipse CDT and the GDB embedded plugin falls unter the Eclipse Public License.
- Open On Chip Debugger: some of the software falls under the GNU General Public License (GPL). Some packages of the OpenOCD installer have their own version of a license.

1.4.2 OOCDLink

License information: http://www.joernonline.de/dw/doku.php?id=projects:oocdlink:2_oocdlinks.

1.5 Product Overview

The GE863-PRO^3 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.

1.6 Document Organization

This manual contains the following chapters:

- "Chapter 1, **Error! Reference source not found.**" provides a scope for this manual, target audience, technical contact information, and text conventions
- "Chapter **Error! Reference source not found.**, **Error! Reference source not found.**" describes the general information on how to install development environment that will be used.
- "Chapter 3, **Error! Reference source not found.**" describes how to start a project in C or import an existing project, including examples, where relevant.
- "Chapter 4, **Error! Reference source not found.**" provides info on how to setup and use the debugging environment for OOCDLink and JTAG emulator SAM-ICE
- "Chapter 5, **Error! Reference source not found.**" 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.

1.7 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.

1.8 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 1vv0300784

[6] TelitGE863PRO3 Product Description 80285ST10036a

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

1.9 Document History

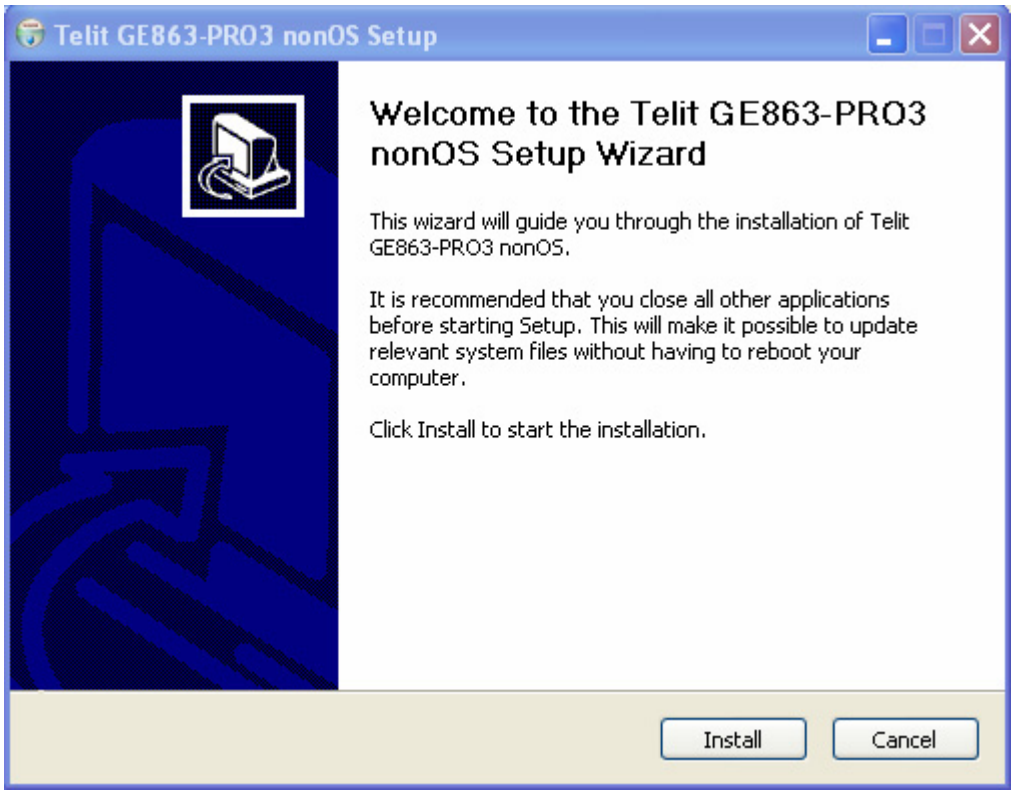
Revision	Date	Changes
ISSUE #0	25/01/07	First release
ISSUE #1	23/04/08	Added SAM-ICE is a JTAG emulator description in paragraph 4.2 Memory map paragraph moved to SW User Guide
ISSUE #2	10/09/08	New setup environment description

2 Installing the Development Environment

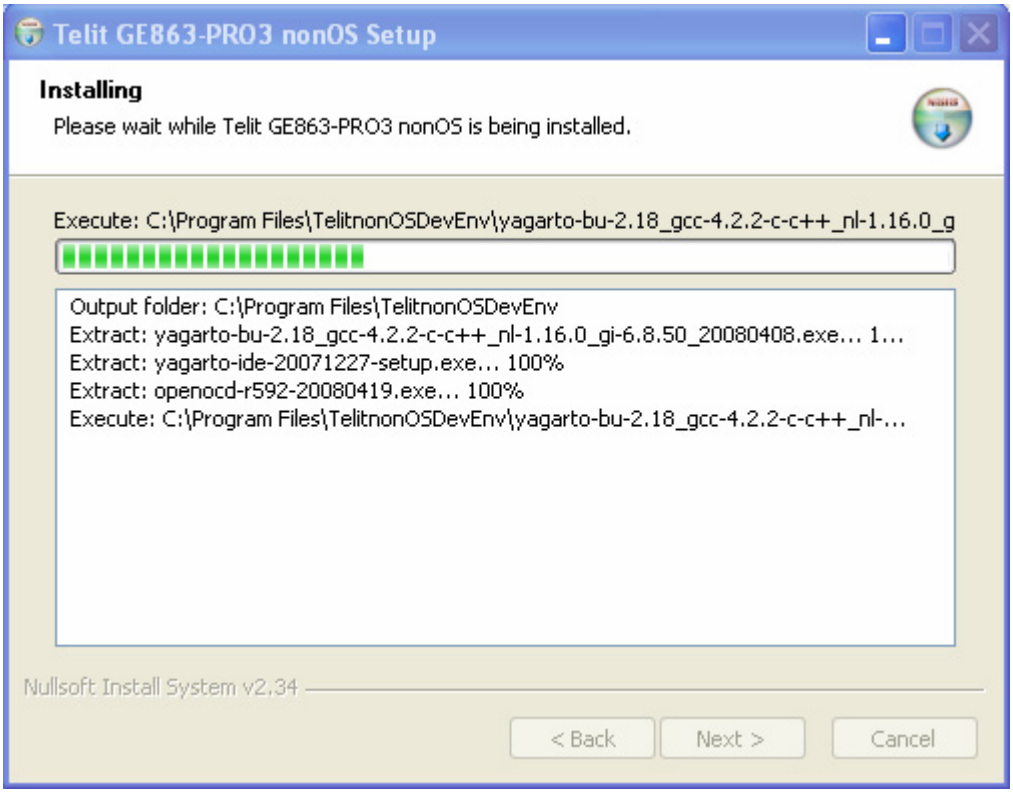
In order to set up the development environment the following steps should be taken:

- Go to http://www.telit.com/ → Download Zone,
- LOGIN with user and password provided by Telit.
- Click on **Software Tools_GSM/GPRS,**
- Click on **GE863-PRO³**
  Click on the **ZIP_DOWNLOAD** of the **Telit_GE863-PRO3_nonOSDevEnvironment.zip**
- Save it in a path like C:\Documents and Settings\user\Desktop.
- UnZip **Telit_GE863-PRO3_nonOSDevEnvironment.zip**, the unzip creates the file **TelitGE863PRO3nonOSDevEnvironment.exe**
- Run the installation of the development environment by clicking twice on the file:

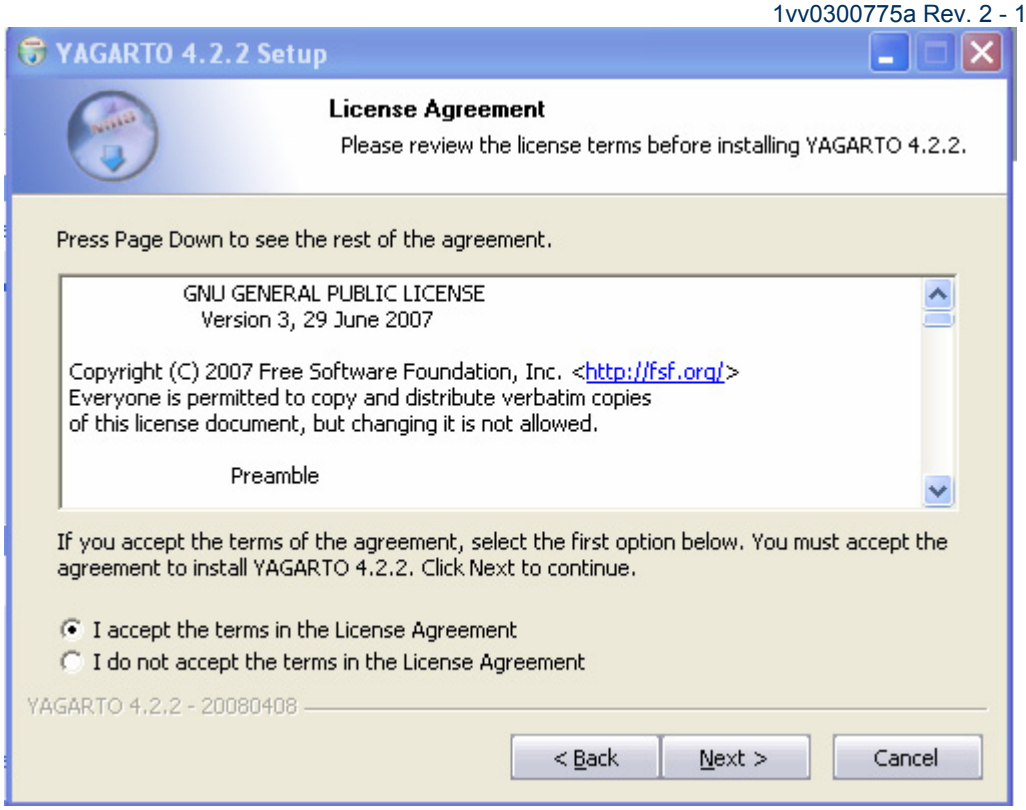
  - **TelitGE863PRO3nonOSDevEnvironment.exe:**



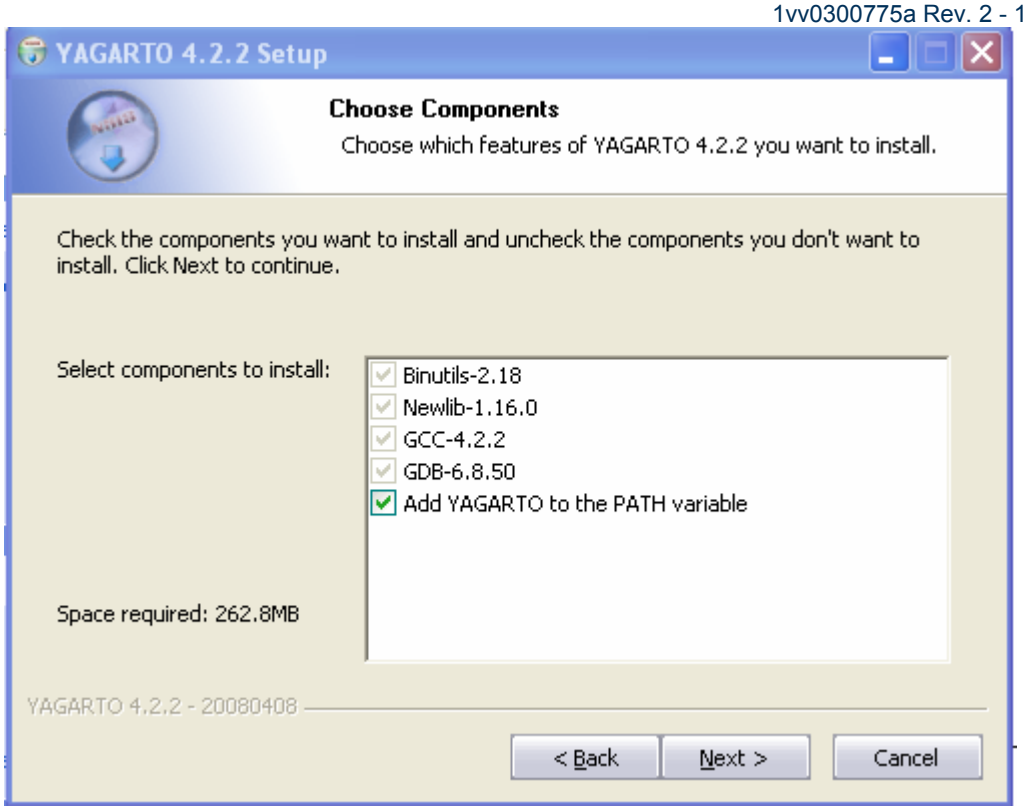
Click **Install**



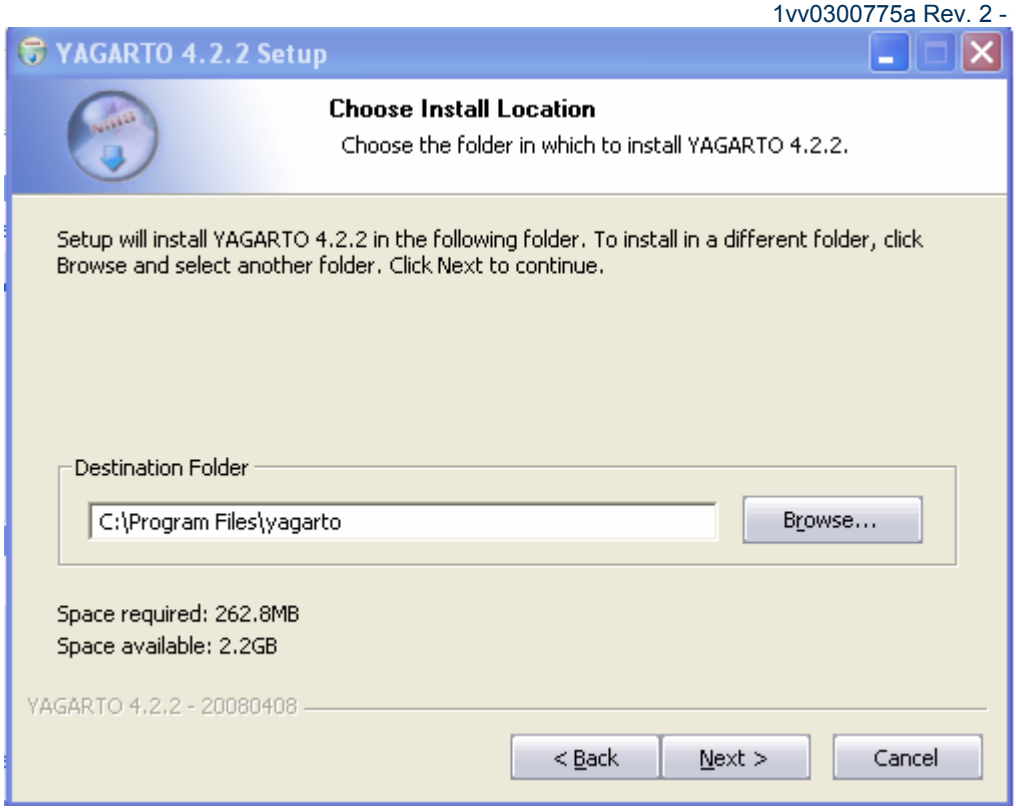
Click **Next**



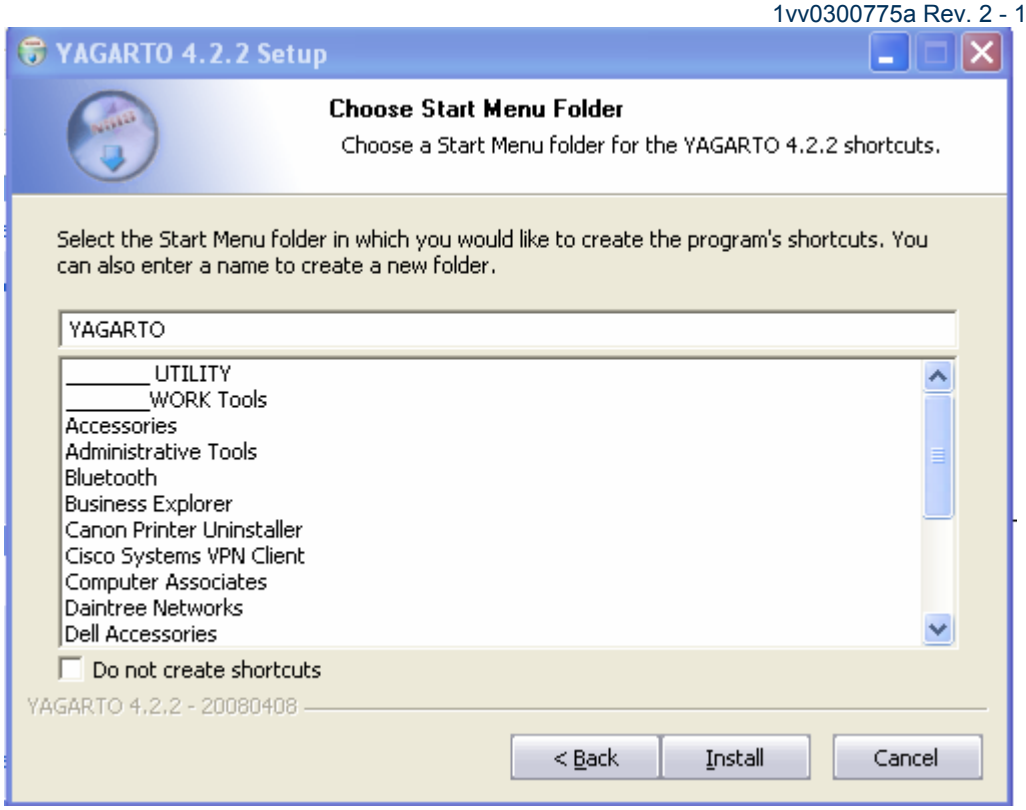
Carefully read the information and click: **I accept the terms in the License Agreement**, then click **Next.**



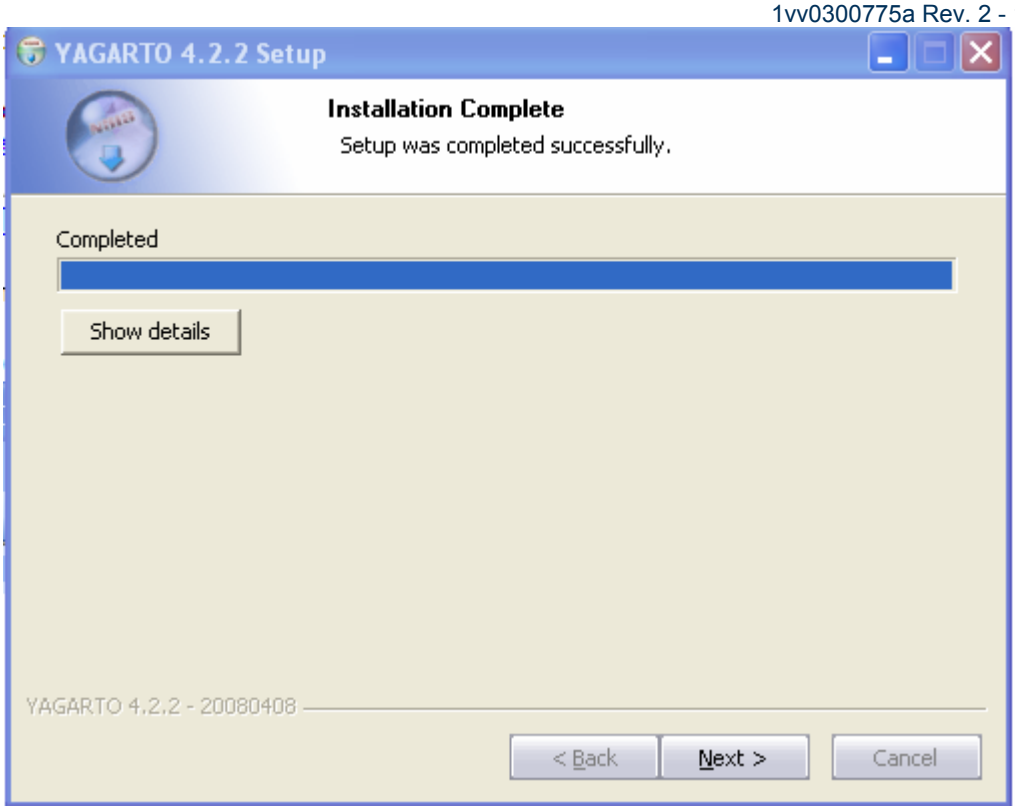
Click **Next**.



Click **Next**.



Click **Install.**

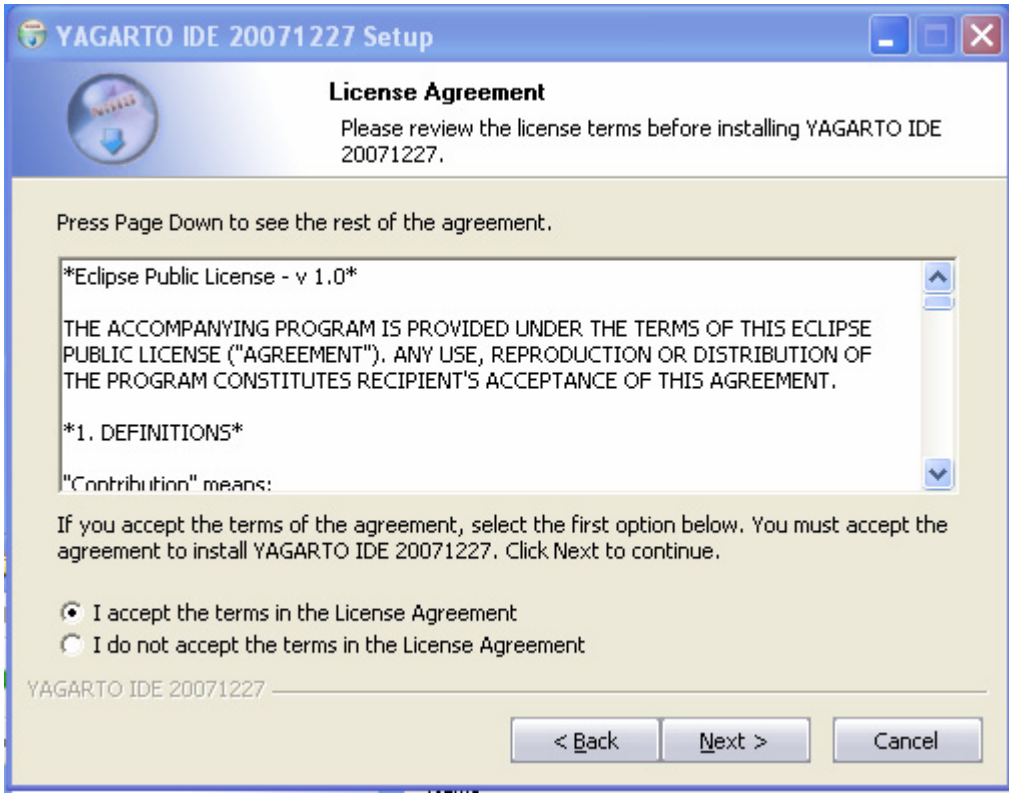

YAGARTO 4.2.2 Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
YAGARTO 4.2.2 - 20080408
< Back
Next >
Cancel

Click **Next.**

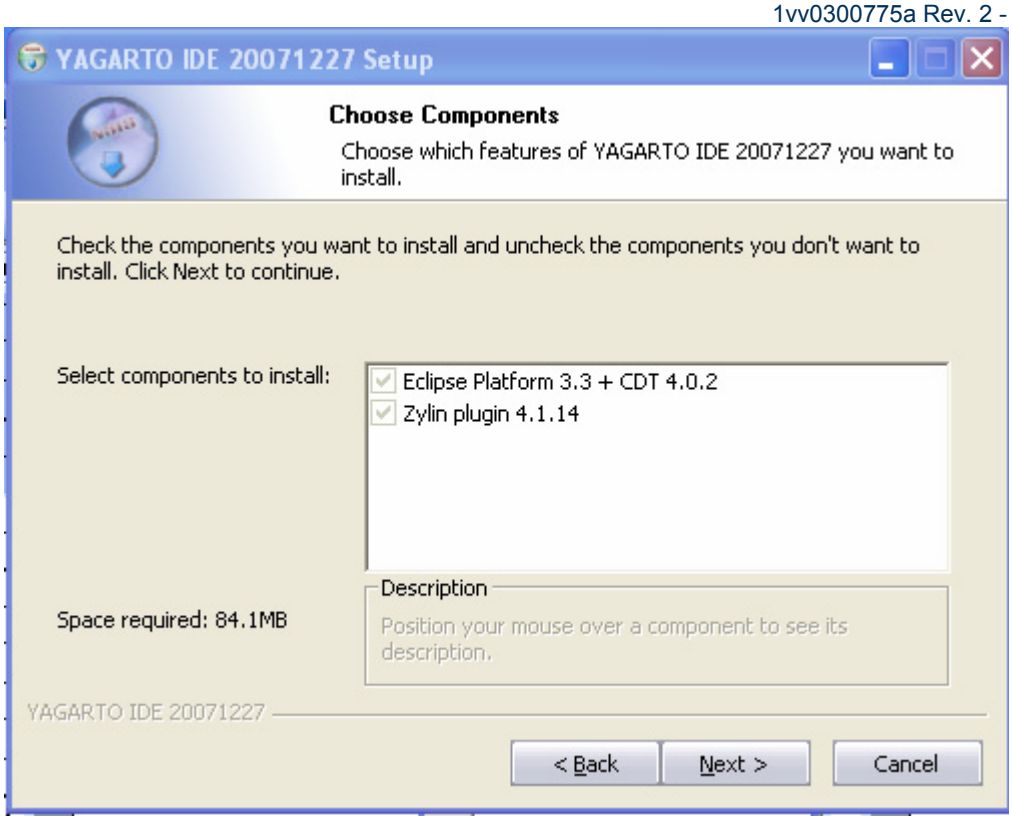
Click **Finish**.


YAGARTO IDE 20071227 Setup
Welcome to the YAGARTO IDE 20071227 Setup Wizard
This wizard will guide you through the installation of YAGARTO IDE 20071227.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel

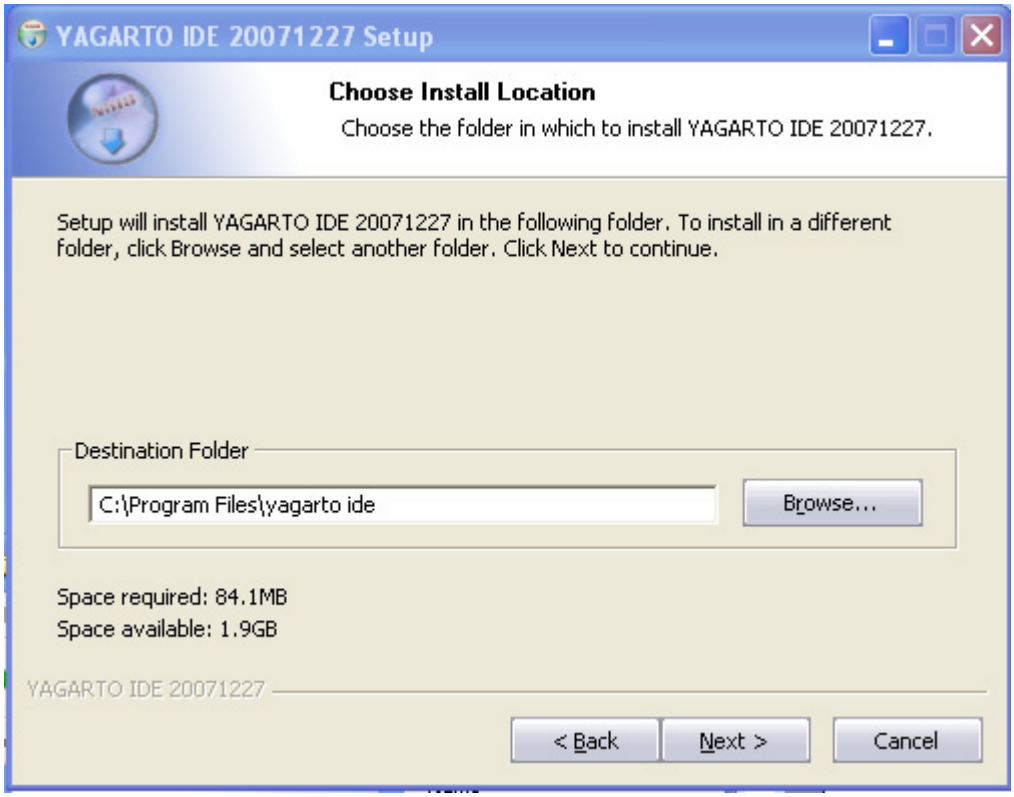
Click **Next**.



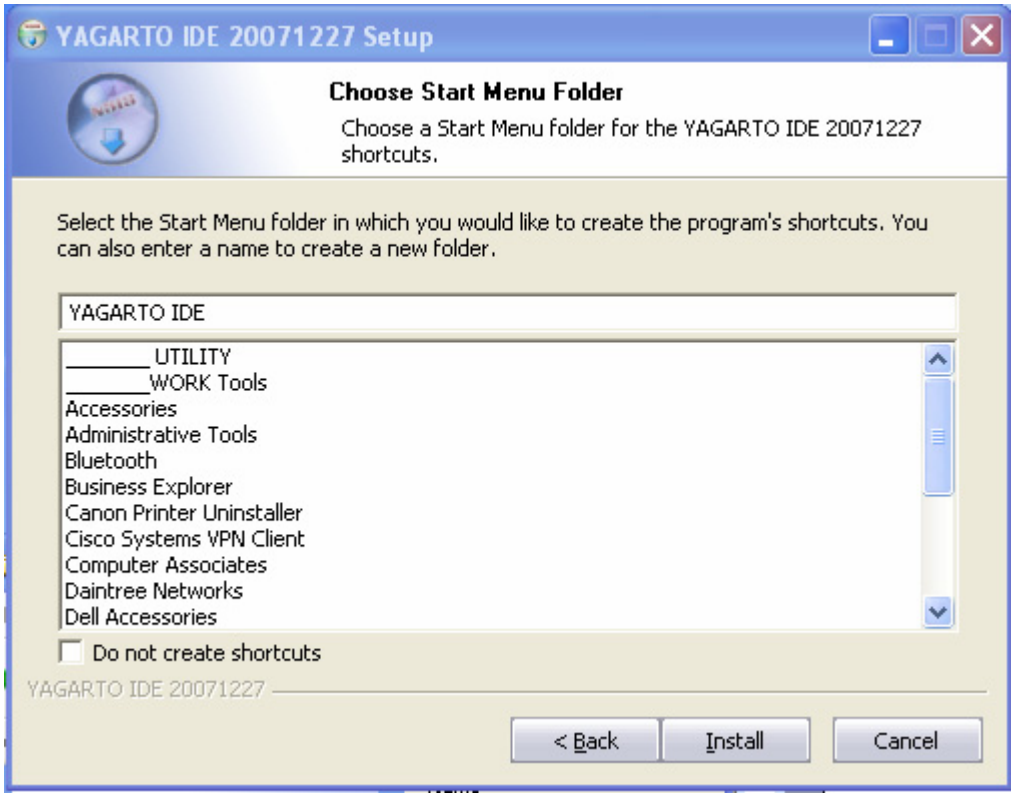
Carefully read the information and click: **I accept the terms in the License Agreement**, then click **Next.**



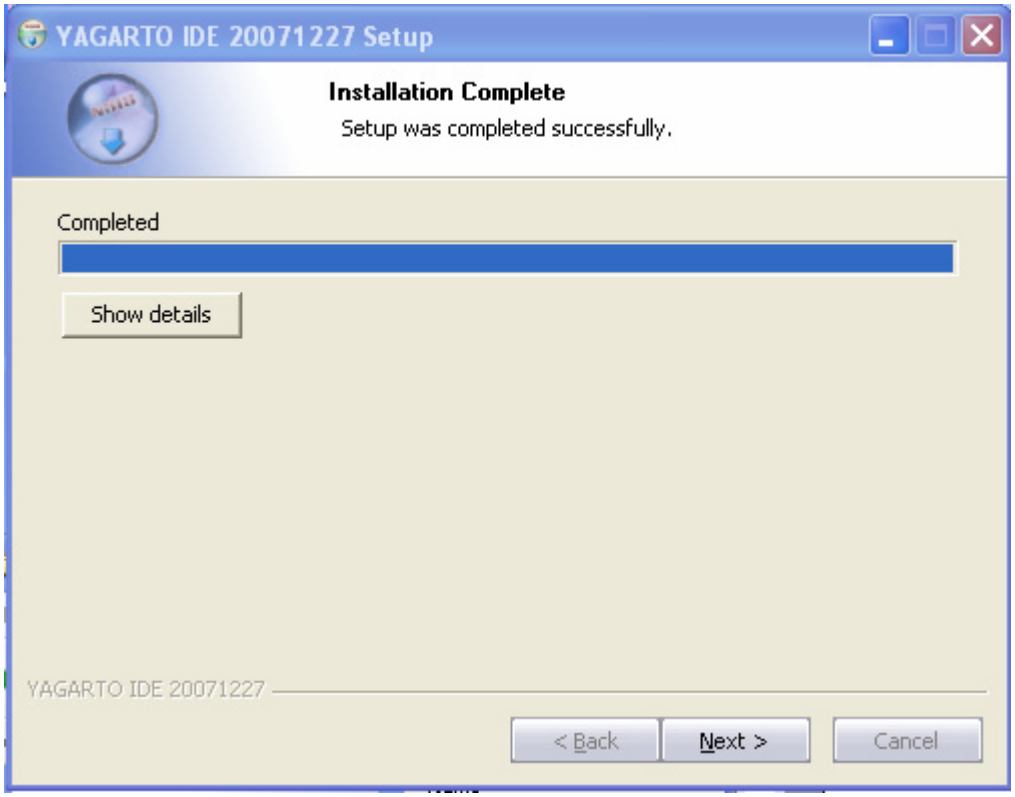
Click **Next**.



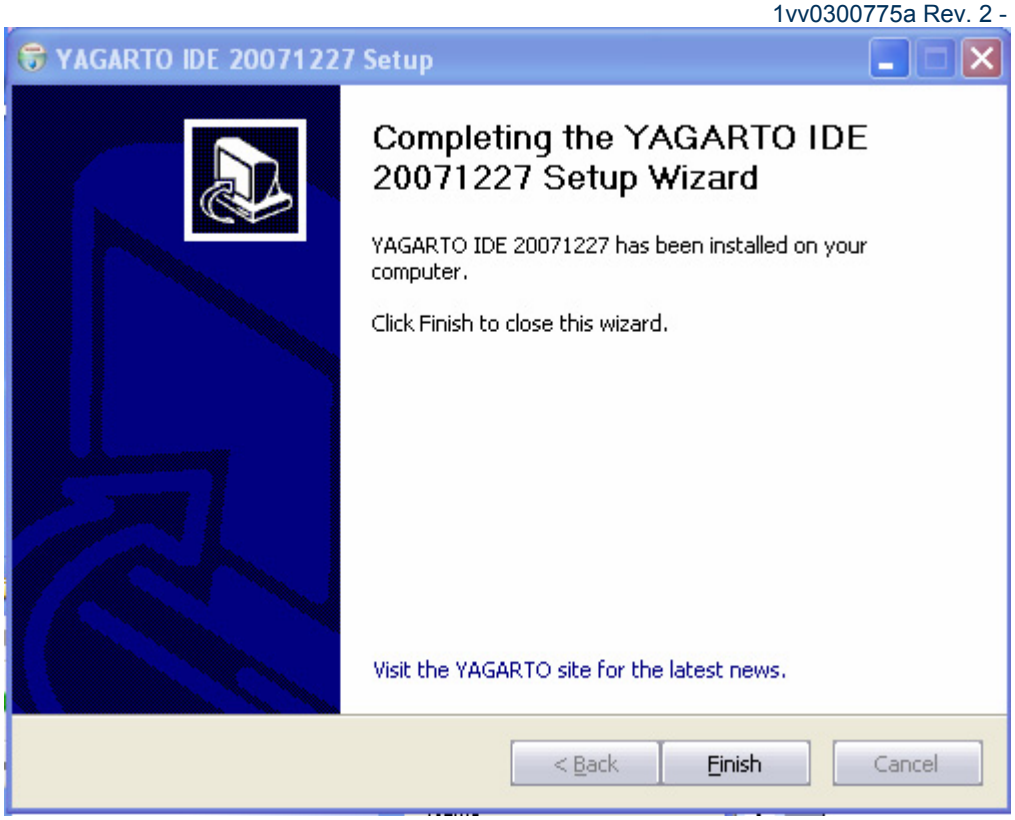
Click **Next**.



Click **Install**.

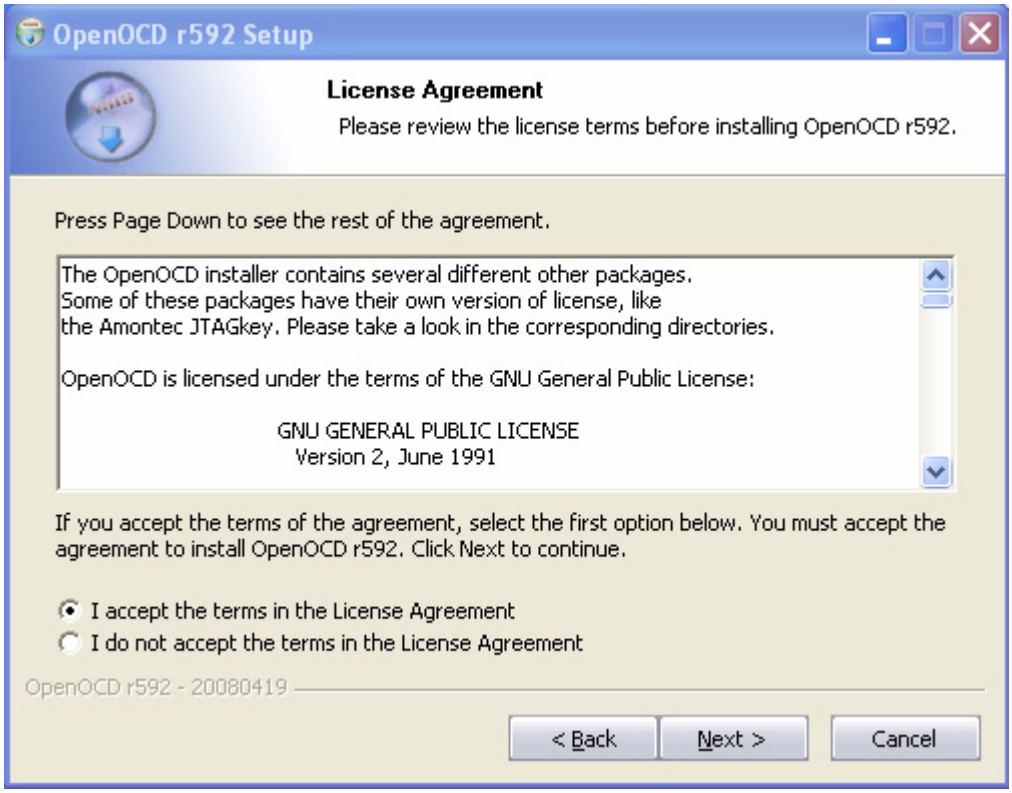


Click **Next**.

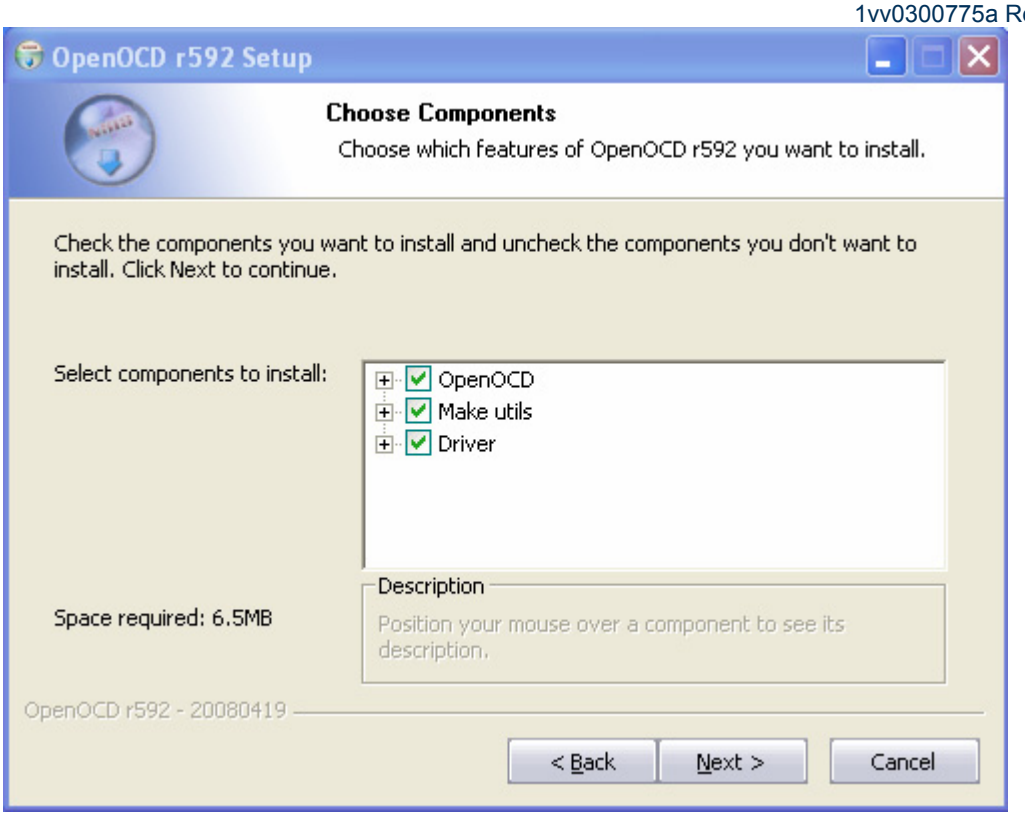


Click **Finish**.

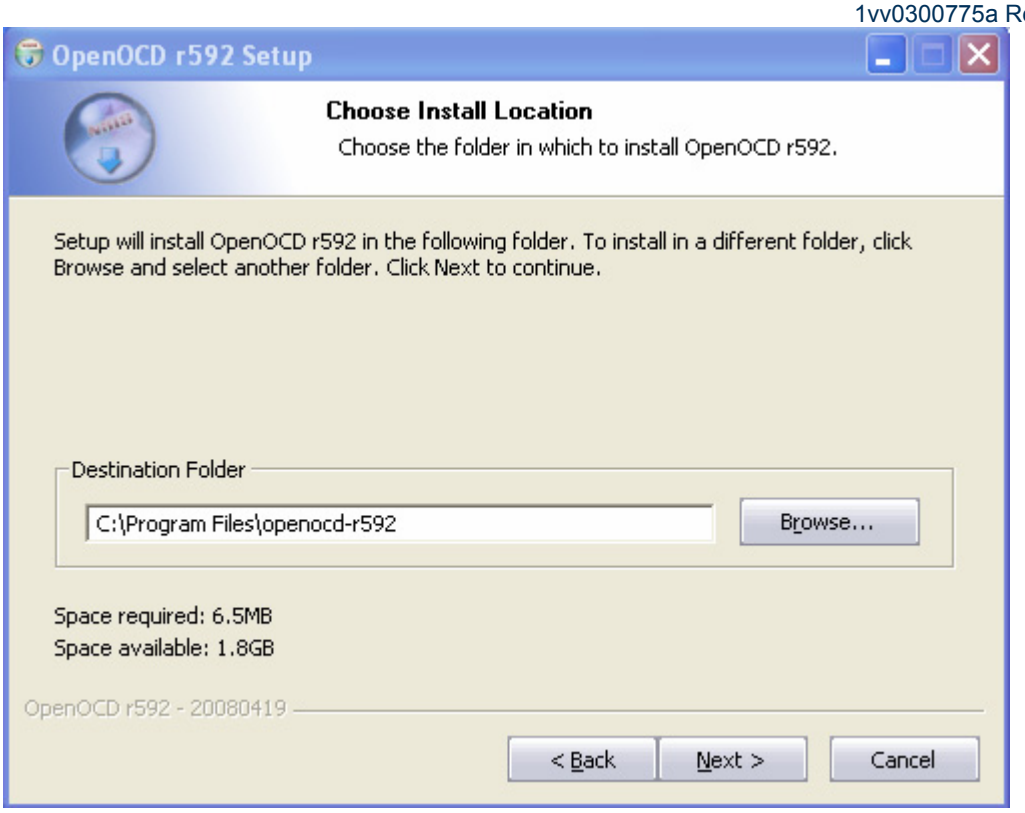
Click **Next**.



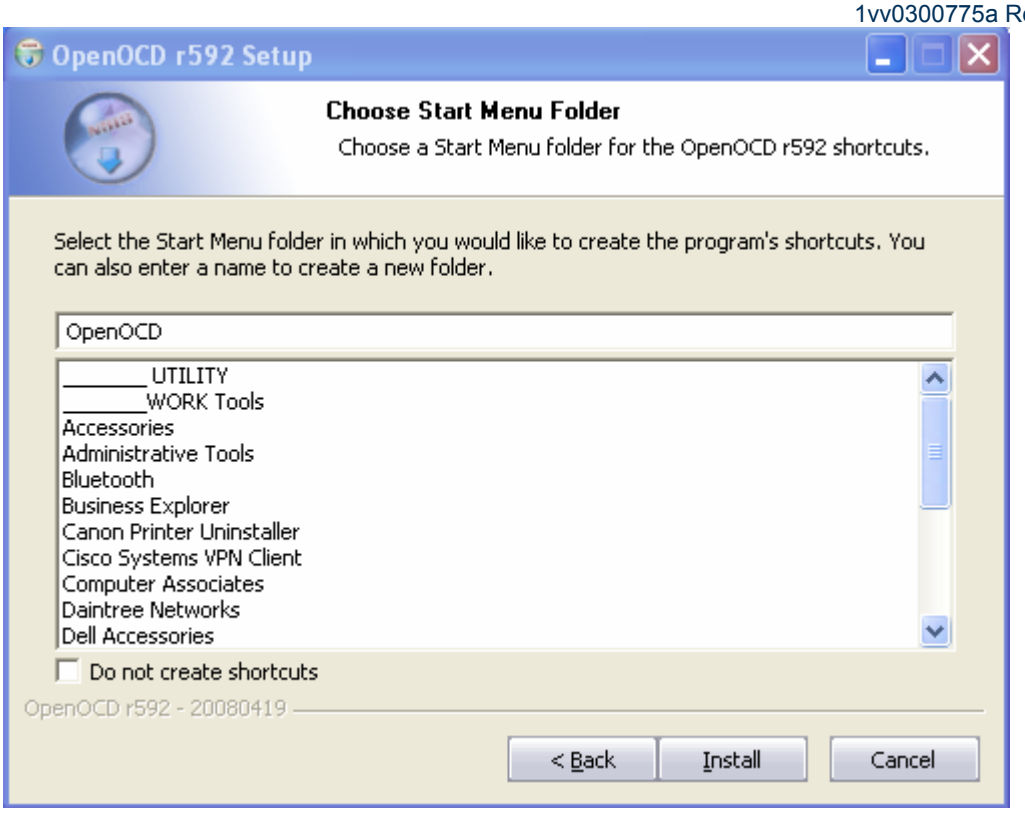
Carefully read the information and click: **I accept the terms in the License Agreement**, then click **Next.**



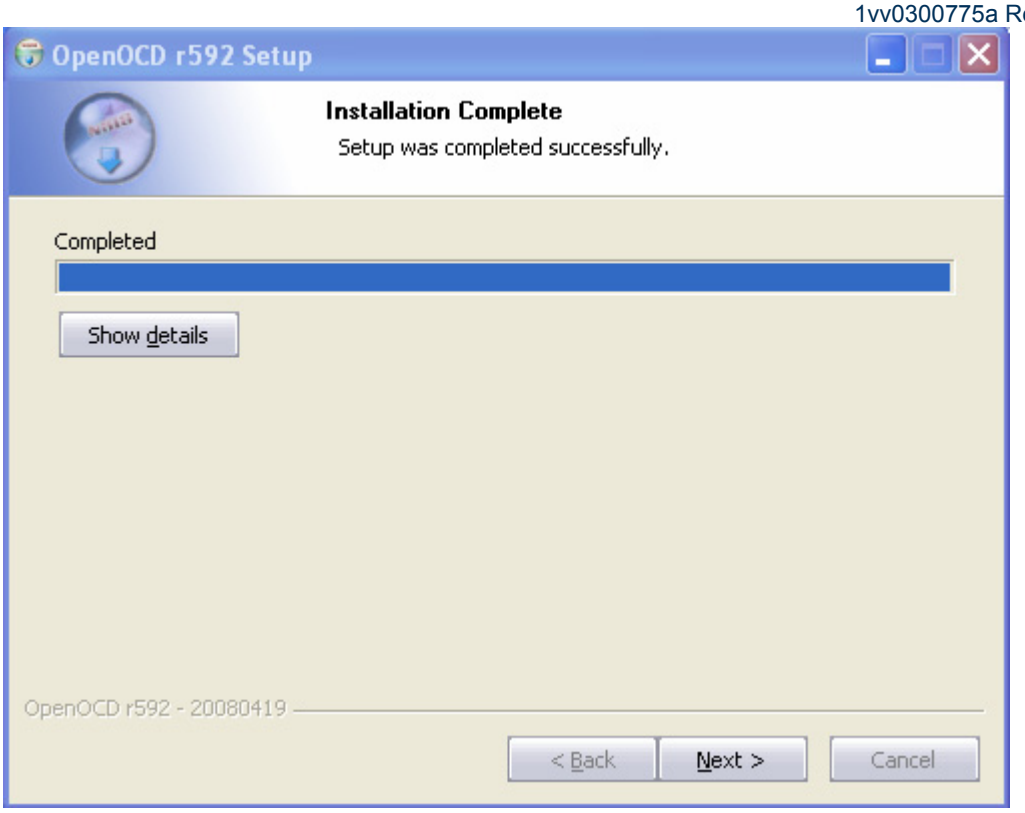
Click **Next**.



Click **Next**.

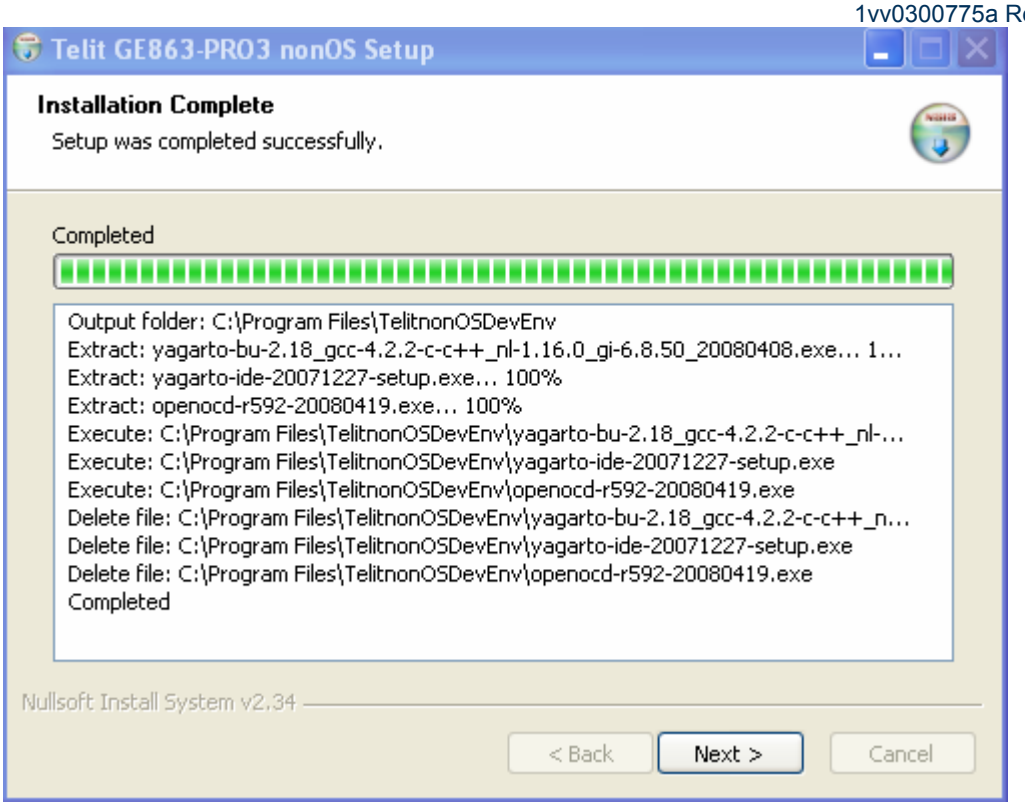


Click **Install**.

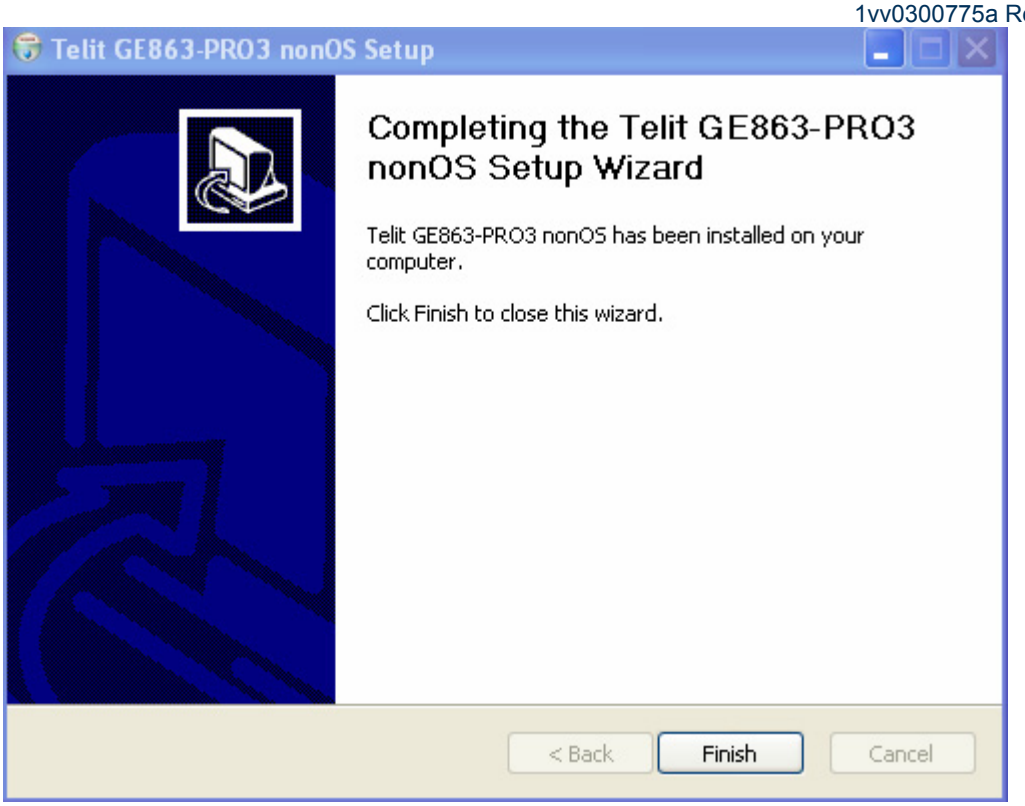


Click **Next**.

Click **Finish**.

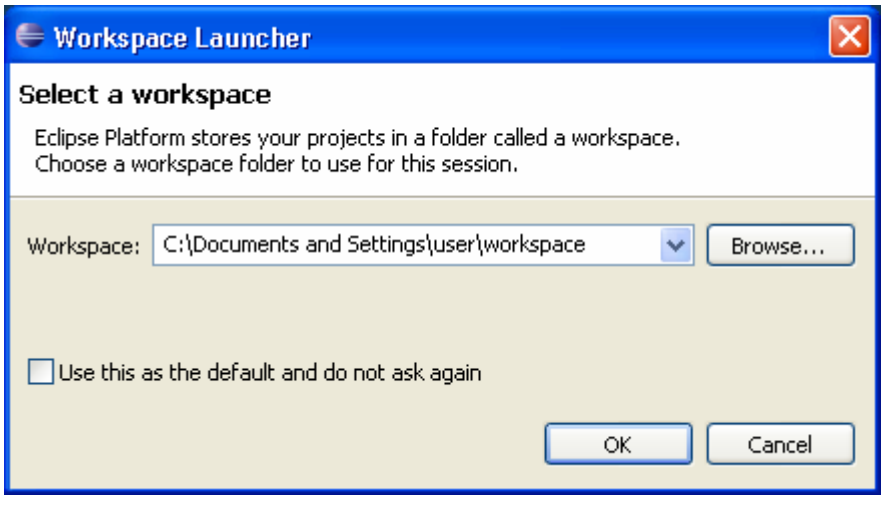


Click **Next**.

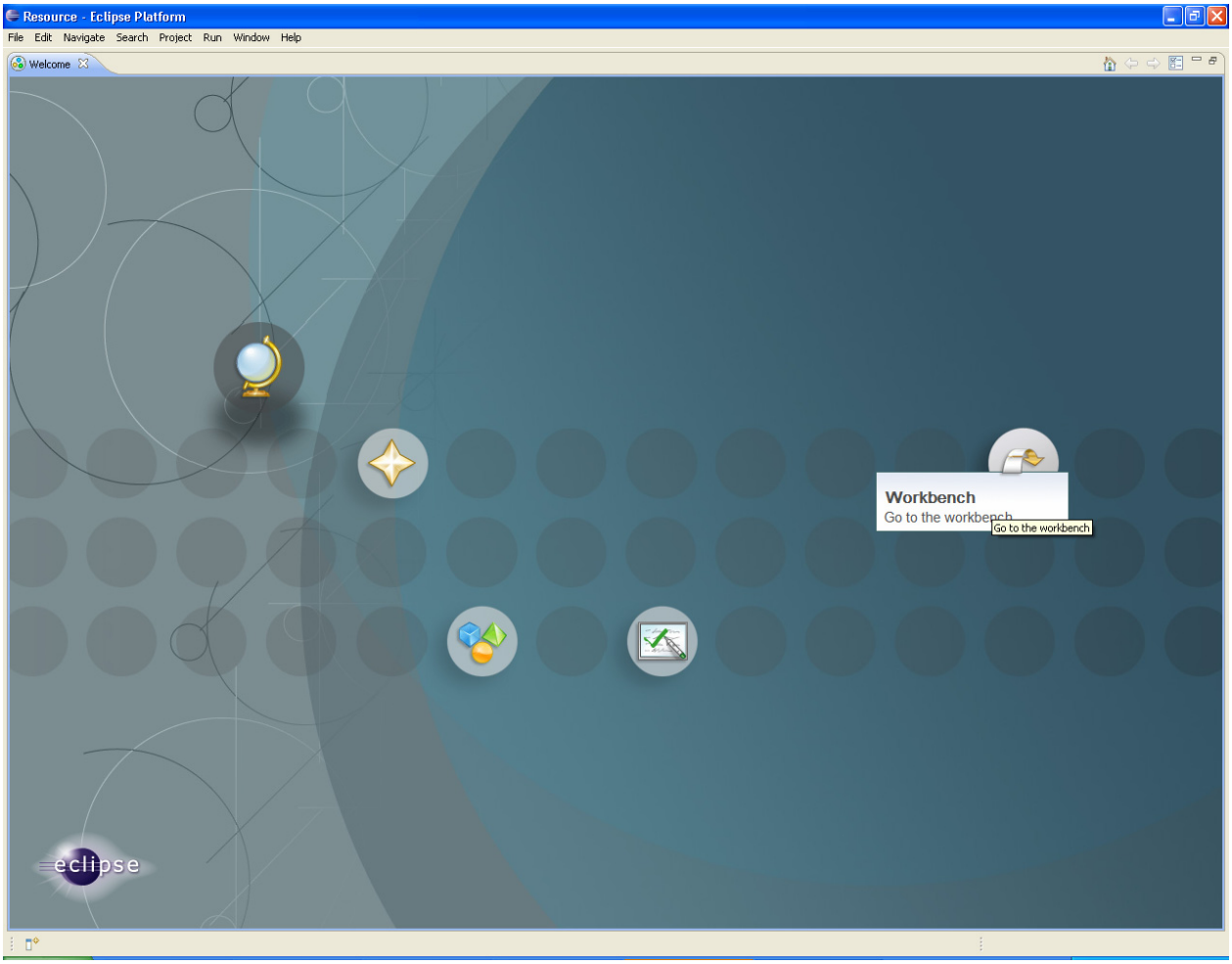


Click **Finish**.

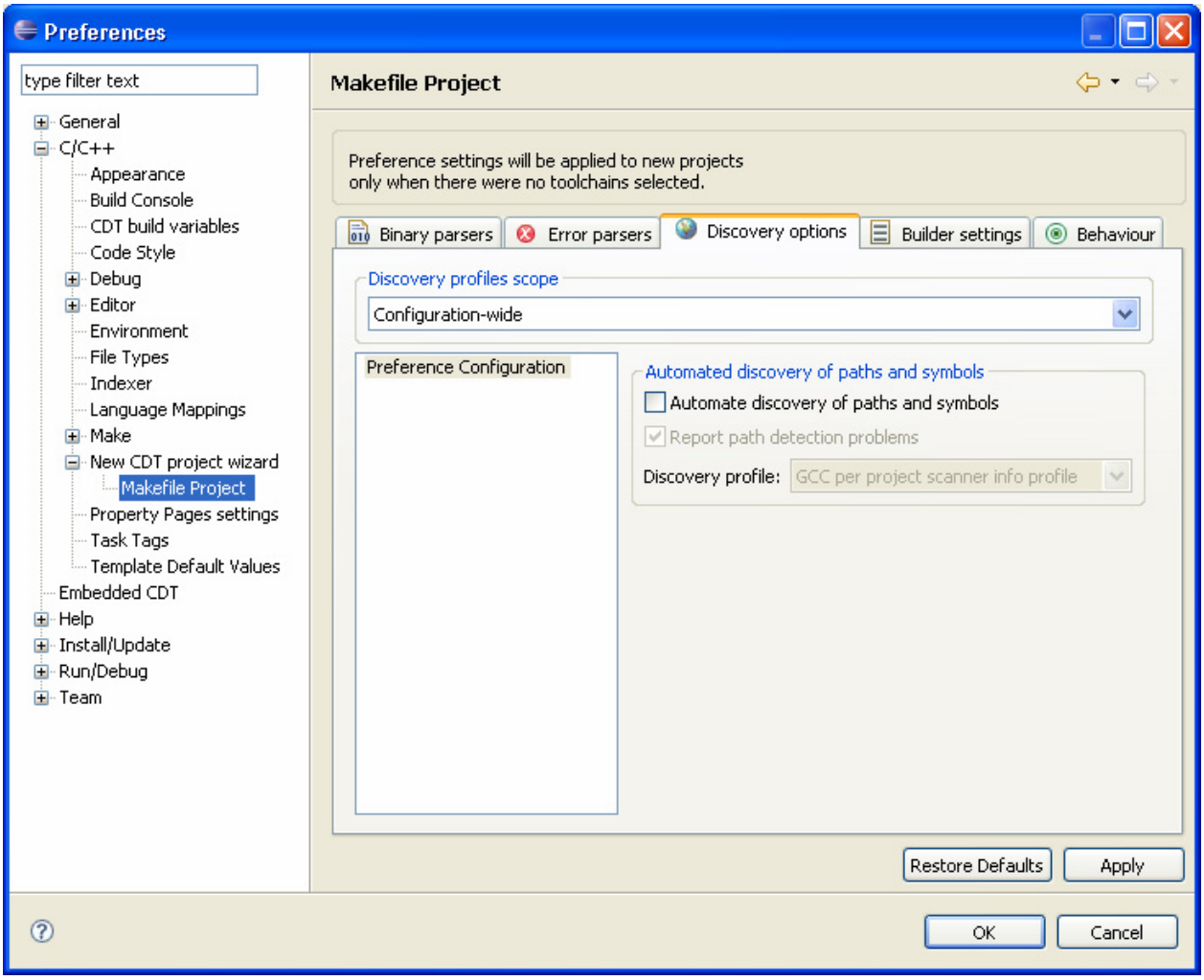
- Open the IDE (**Start menu → All Programs → YAGARTO IDE → Eclipse Platform 3.3**) and choose a location in the filesystem for creating the workspace



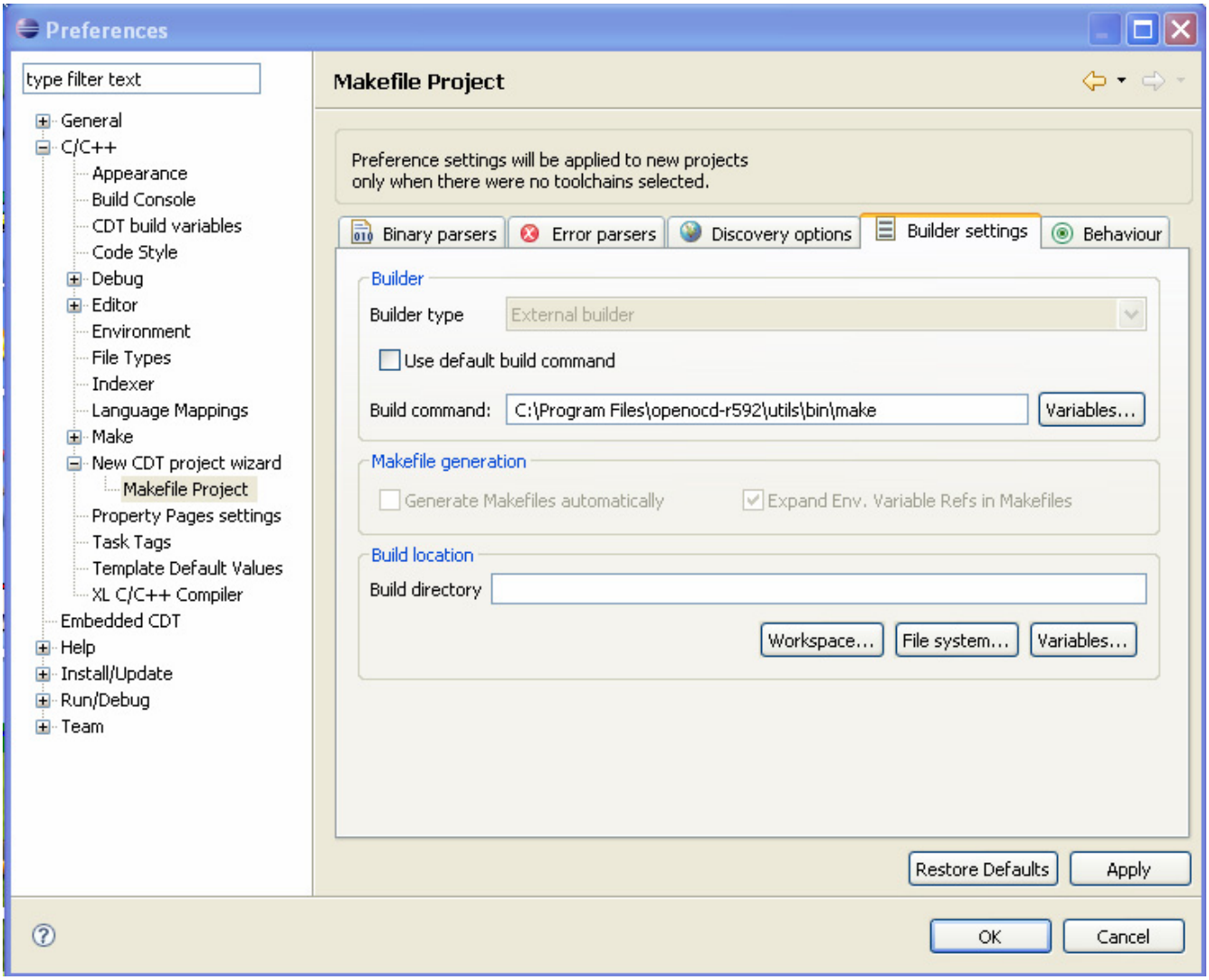
- When the workspace has been created click on the **Go to the Workbench** icon.



- Open the menu **Window** → **Preferences**.
- In the left column of the window expand the entry **C/C++**; inside it, expand the entry **New CDT Project Wizard** and select **Makefile Project**.
- In the right part of the window select the tab **Discovery Options** and uncheck the entry **Automate Discovery of Path and Symbols**. Click on the **Apply** button.



- Select the tab **Builder Settings** and uncheck the entry **Use Default Build Command**; in the text field called **Build Command** write the path of the make executable installed with the Yagarto GNU ARM toolchain (if you have not changed the default path, it should be something like C:\Program Files\openocd-r592\utils\bin\make). Afterwards click on the **Apply** button and then click **OK.**

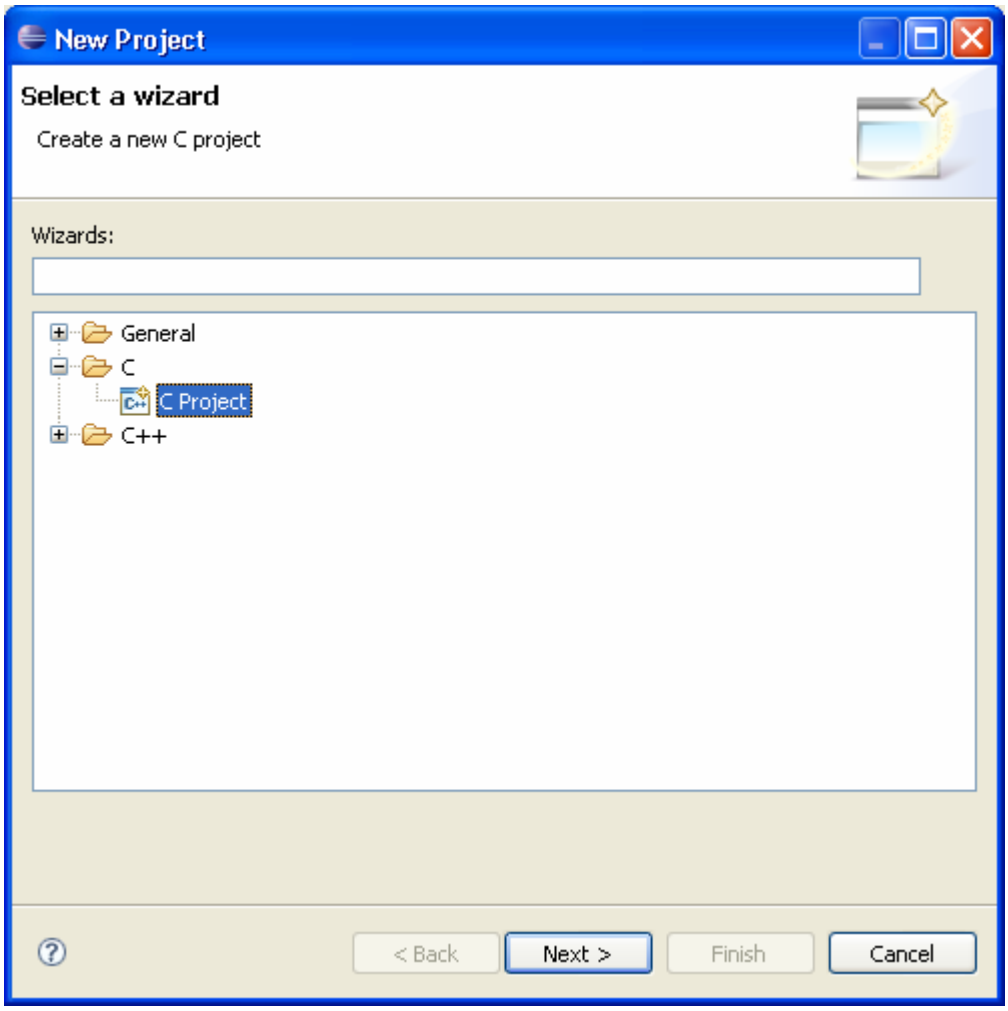


Now you are ready to develop your first application for the PRO^3.

3 Creating a project

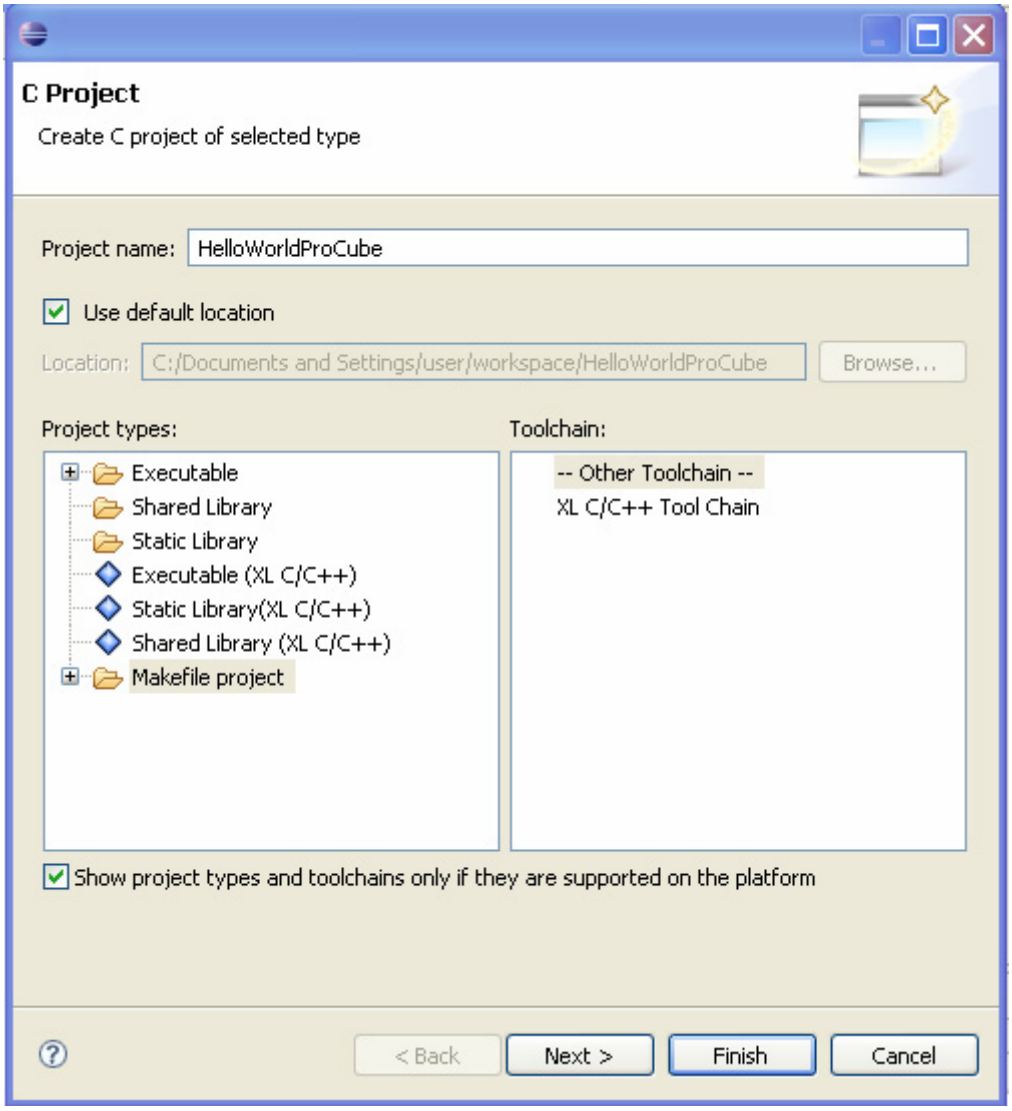
When you start creating a new project follow the steps below:

- 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**.

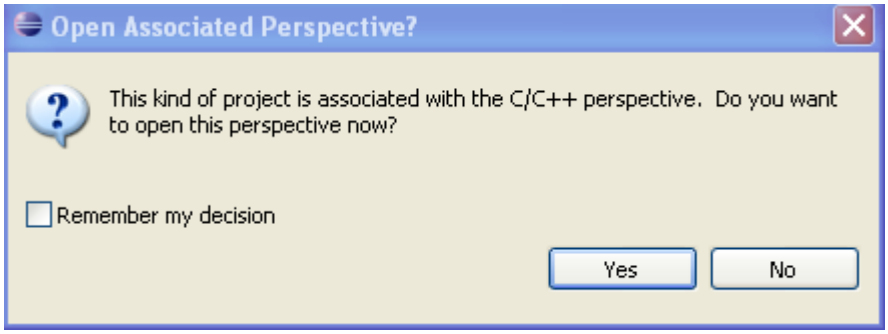


- Fill in the **Project Name** text field; in the **Project Types** area select the **Makefile Project** folder.

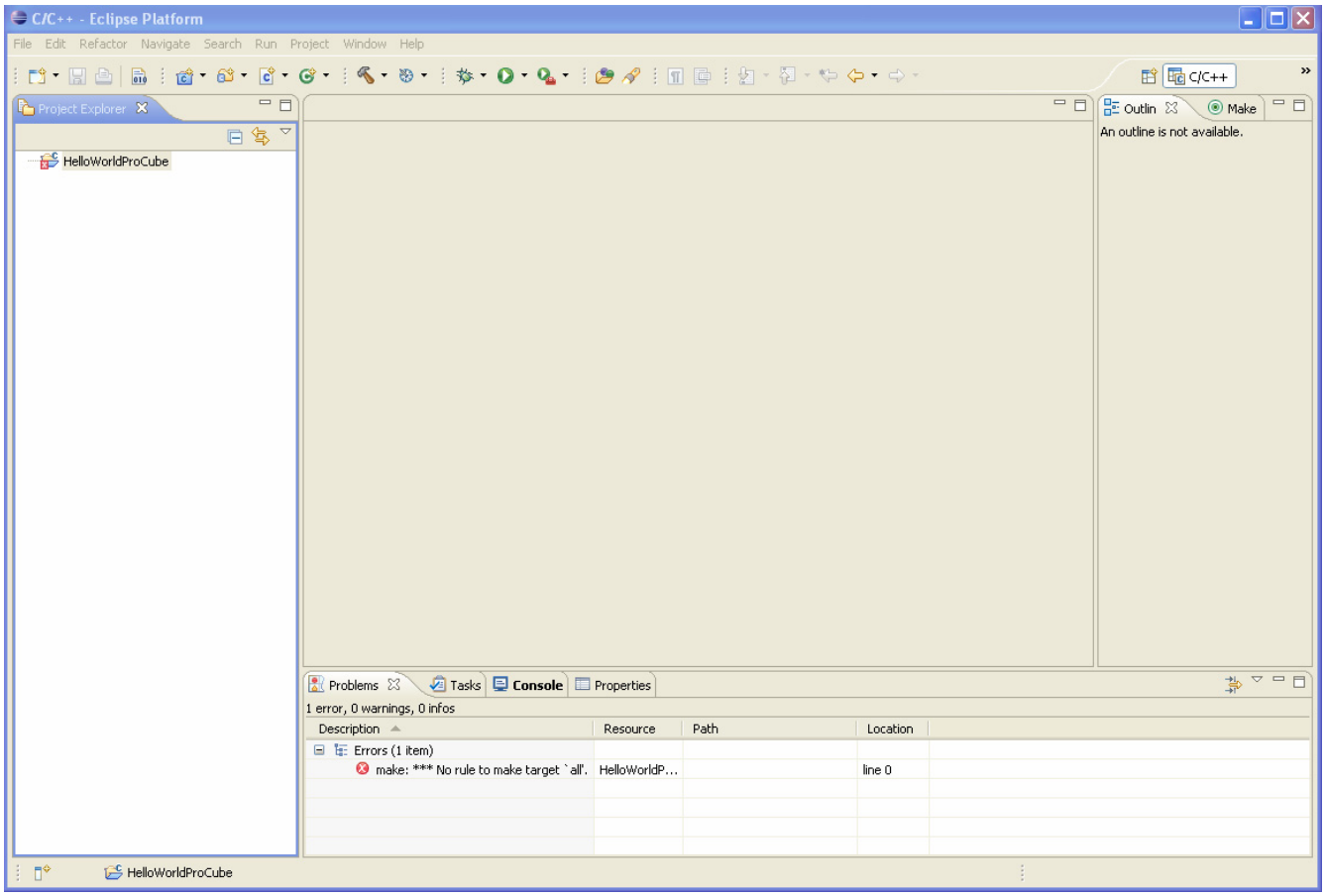
- In the **Toolchain** area choose the **Other Toolchain** entry and then click **Finish**.



- If the window entitled **Open Associated Perspective** comes up answer **Yes**. This will open the IDE's C/C++ perspective.



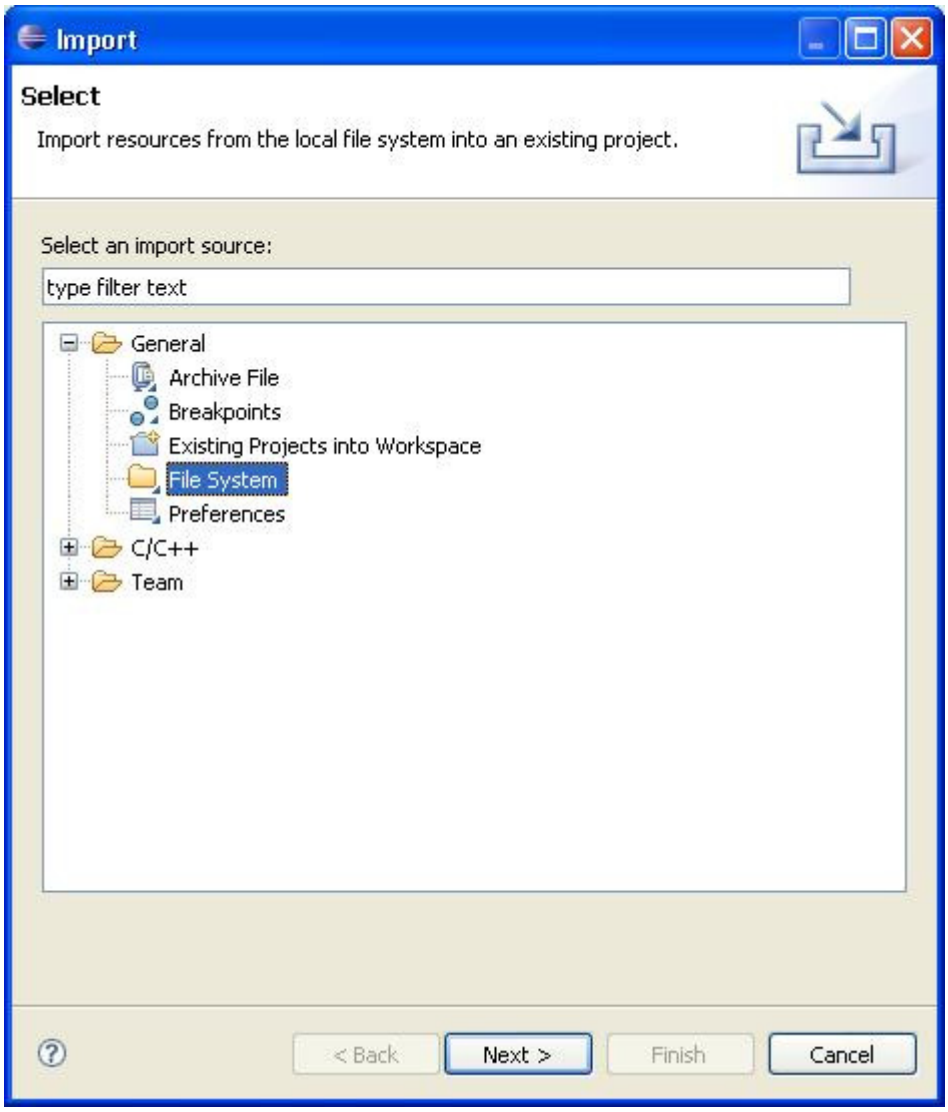
- Now you should see the workspace with the newly created project: it will be empty because currently there are no files. The message in the panel **Problems:** "*make: *** No rule to make target `all'.*" appears because at the moment in the project there are no files and it has not been yet configured the build directory.



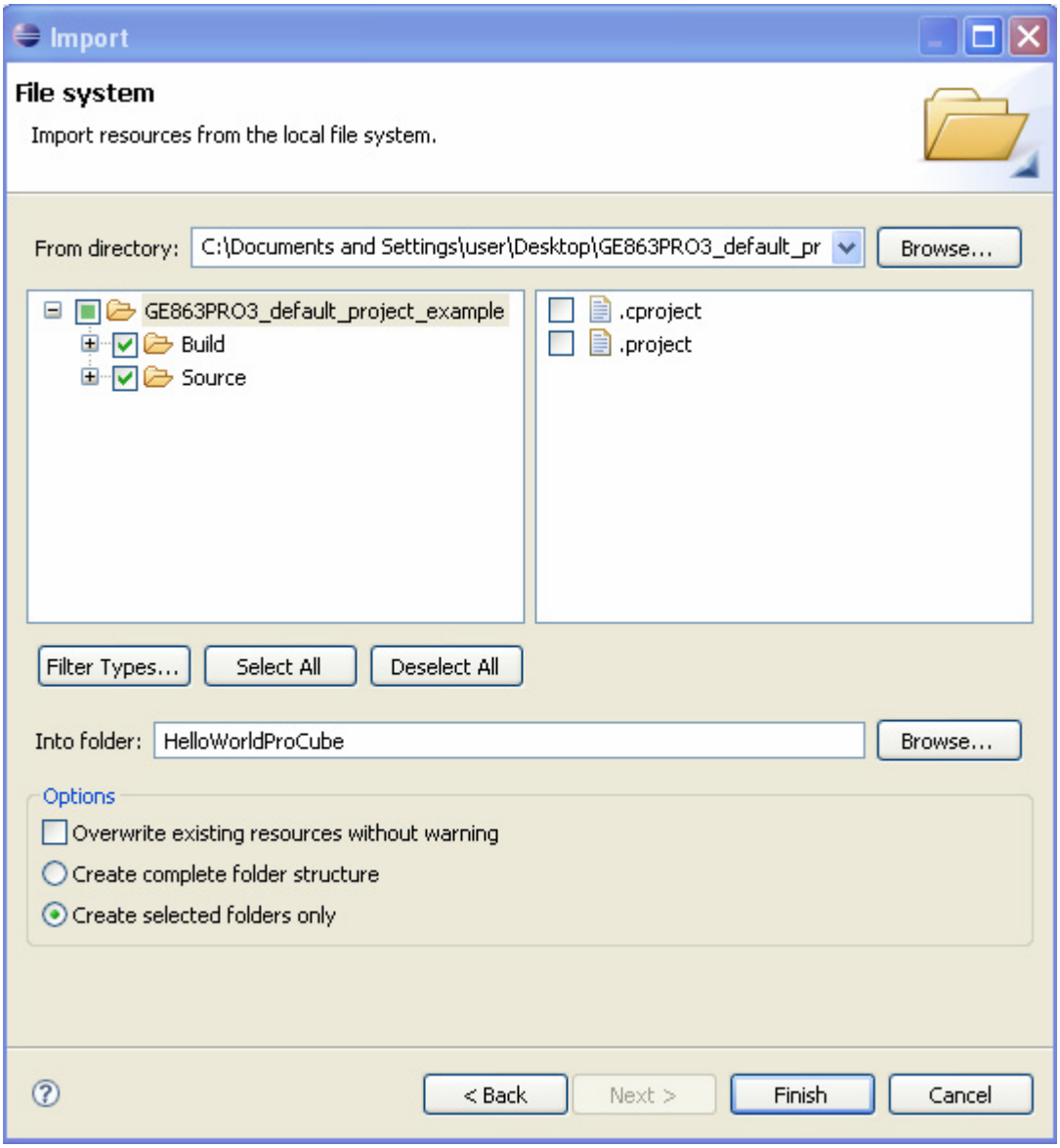
- To start developing an application for the PRO^3 you need to download the files contained in the directory **GE863PRO3_default_project_example** provided by Telit.
  To get them following next steps:
  - go to http://www.telit.com/ → Download Zone,
  - LOGIN with user and password provided by Telit.
  - Click on **Software Tools_GSM/GPRS,**
  - Click on **GE863-PRO³**
  - Click on the **ZIP_DOWNLOAD** of the **Telit_GE863PRO3_default_project_example.zip**
  - **Save** it in a path like C:\Documents and Settings\user\Desktop.
  - UnZip **Telit_GE863PRO3_default_project_example.zip** (the unzip creates the directory **GE863PRO3_deafult_project_example**)

- To start developing an application for the PRO^3 you need to import the files contained in the directory **GE863PRO3_deafult_project_example** provided by Telit. These files supply the drivers and configurations necessary to make the system work. Moreover in the main.c there is an application which can show you how to use the serial ports. To import the directory open the menu **File** → **Import**.

- Open the folder **General** and choose the entry **Filesystem** and then click on the **Next**.

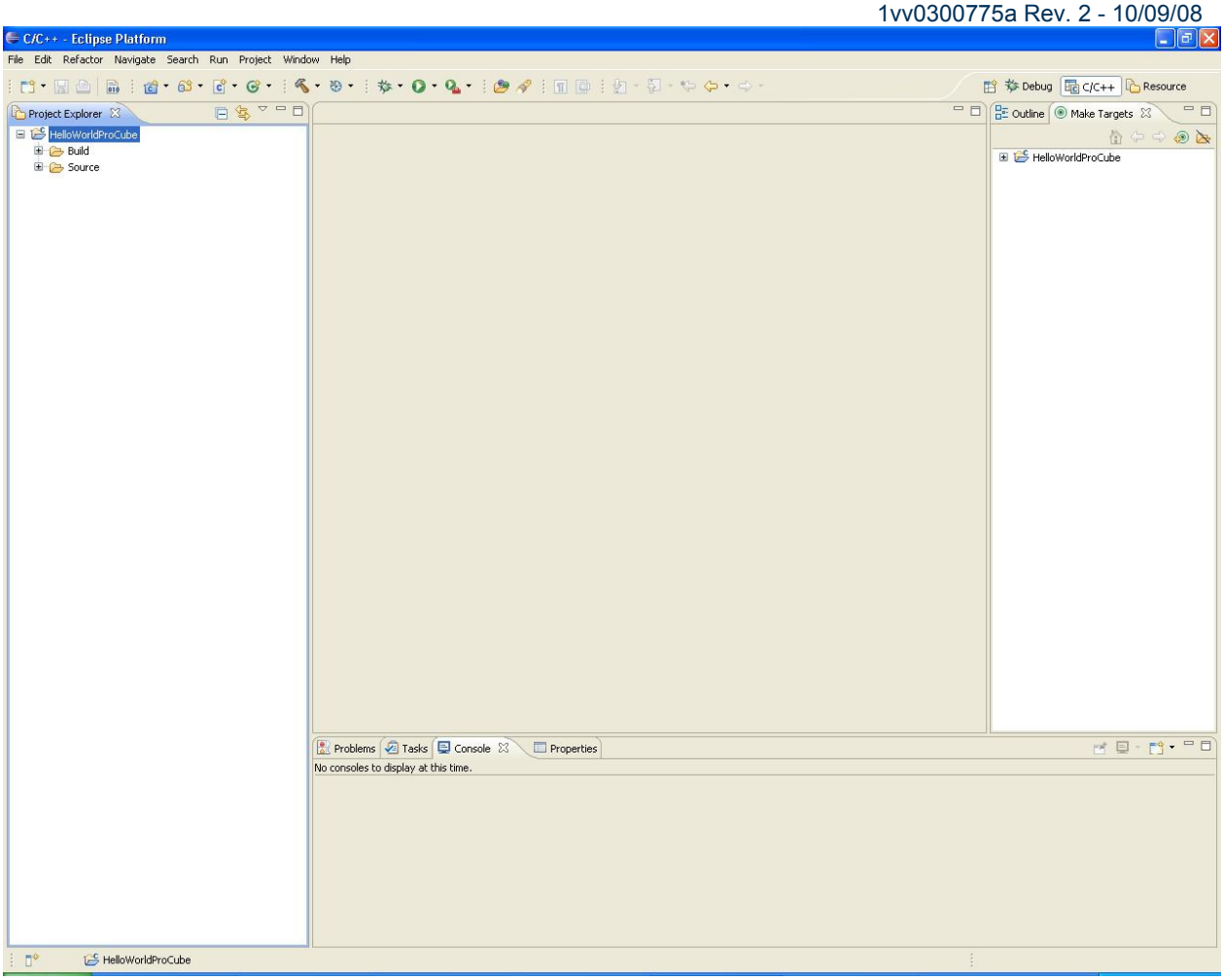


- Browse to the directory containing the files to be imported and check on the folder **GE863PRO3_deafult_project_example**, then click **Finish**.

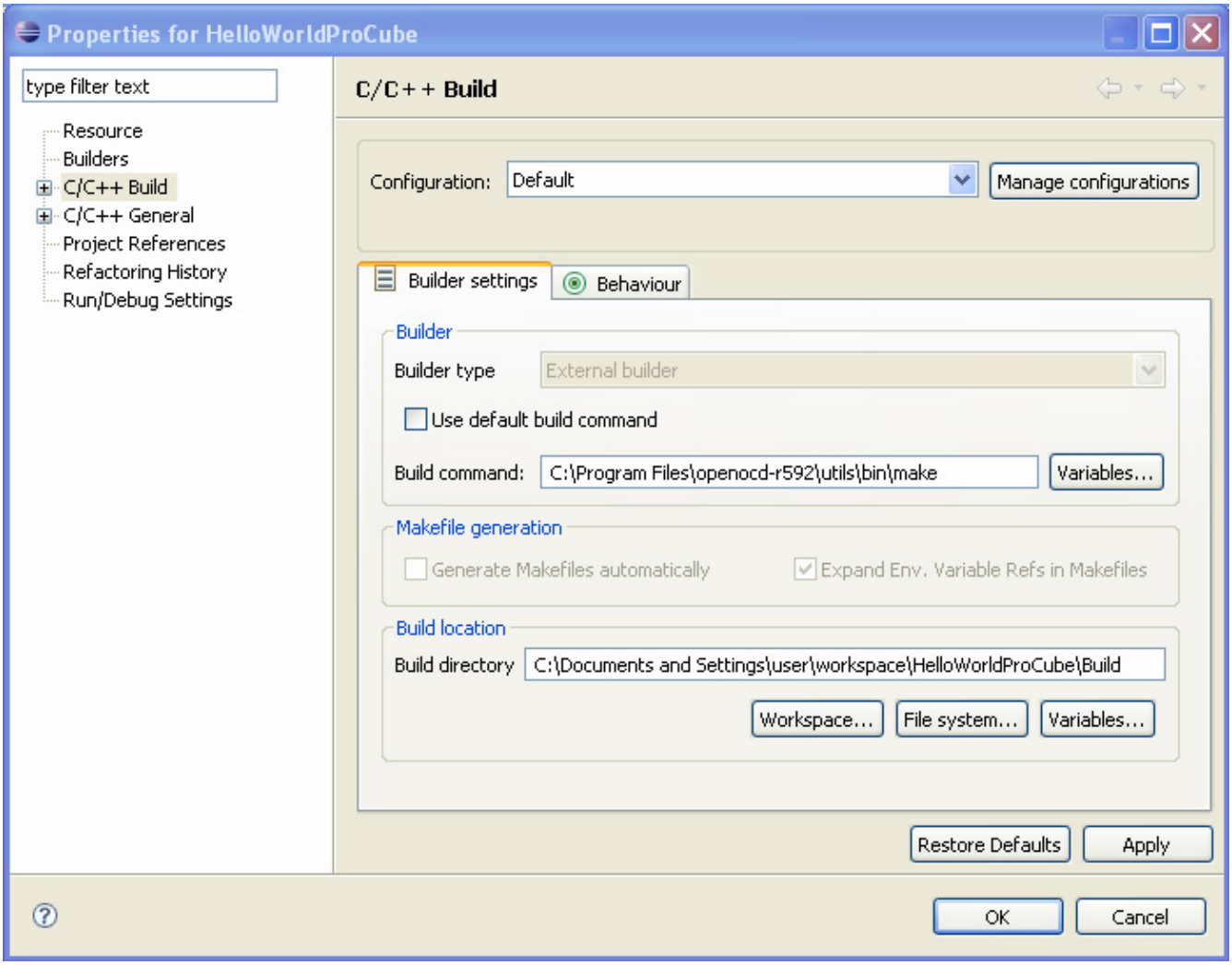


Click **Finish**

- Now you should see the workspace with the imported folders contained in the project.



- Before building the project it's necessary to configure the build directory. Select the project and open the menu **File** → **Properties**.
- In the left column choose the **C/C++ Build** entry and in the right part of the window select the tab **Builder Settings**. Fill in the **Build Directory** text field with the build directory of the HelloWorldProCube project, and then click **Ok**.



- 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 Pro^3 you can use a variety of hardware emulators (for example SAM-ICE, J-Link). The software configuration depends on the type of hardware you use, but the general framework is common: there is the gdb debugger, which accepts input commands from the user (generally through Eclipse's graphical user interface) and talks to a gdb server; this server listens for requests from gdb and manages the communication with the hardware emulator, by means of a driver.
Now we explain how to setup the debugging environment using two alternative emulators: OOCDLink, SAM-ICE and their toolkit.
Depending on the emulator you will use, you need some files which are used by the debugger to establish a connection to the gdb server and initialize the target to debug your application.
These files, which offer two distinct possibilities to debug an application, as will be explained later, contain a sequence of commands which must be executed by gdb to setup a debugging session.

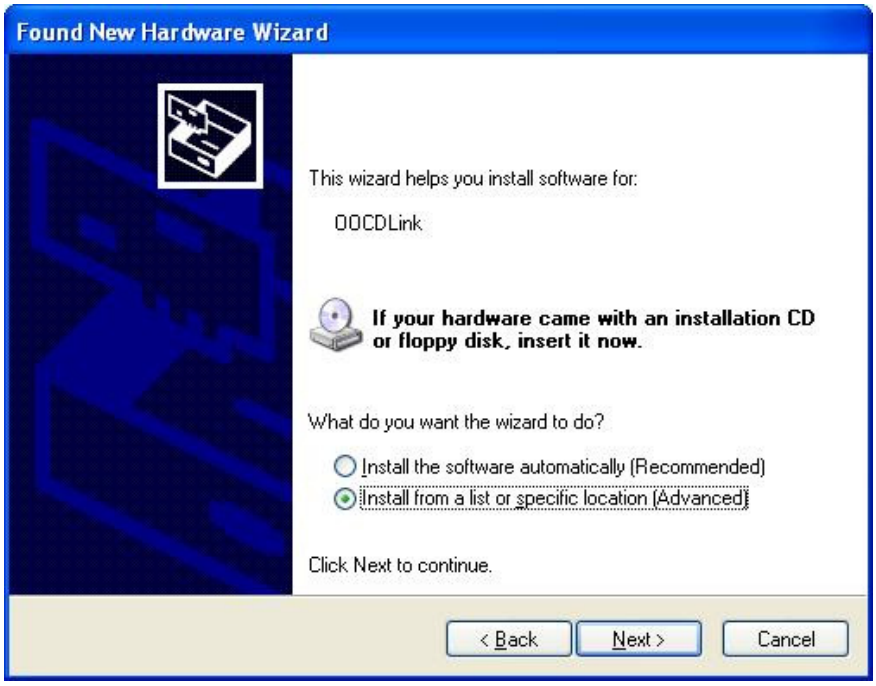
4.1 How to use OOCDLink

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

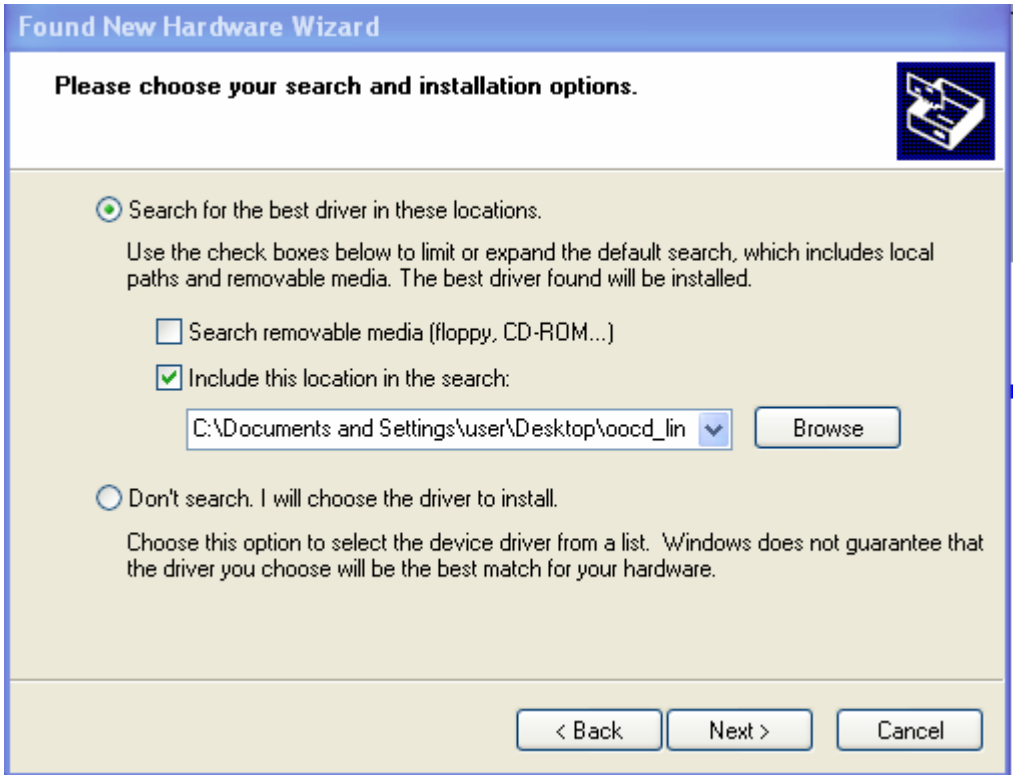
- Download the **OOCDLink drivers** provided by Telit.
  To get them following next steps:
  - go to http://www.telit.com/ → Download Zone,
  - LOGIN with user and password provided by Telit.
  - Click on **Software Tools_GSM/GPRS,**
  - Click on **GE863-PRO³**
  - Click on the **ZIP_DOWNLOAD** of the **oocd_link_driver.zip.**
  - **Save** it in a path like C:\Documents and Settings\user\Desktop.
  - UnZip **oocd_link_driver.zip** (the unzip creates the directory **oocd_link_treiber**)

- 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**.



- Check the **Include this location in the search** box and browse to the directory where you previously unzipped the drivers archive, then click **Next**.



- In the pop-up window entitled **Hardware Installation** click on **Continue Anyway**.

- Click **Finish** in the last window. Now the same wizard will restart asking you for other drivers: repeat the steps you have just followed.

– In the host system create a directory called Ge863Pro3\Debug and copy in it the files: GE863-PRO3_debug_commands.script, GE863-PRO3_debug_commands_boot.script and GE863-PRO3_ft2232.cfg provided in the Telit package **Telit_GE863-PRO³_nonOS_Environment_config_files.zip** (it is not mandatory to put the files in that directory, but if you choose another directory you will have to modify some paths referred to in the rest of this section).
To get those files following next steps:

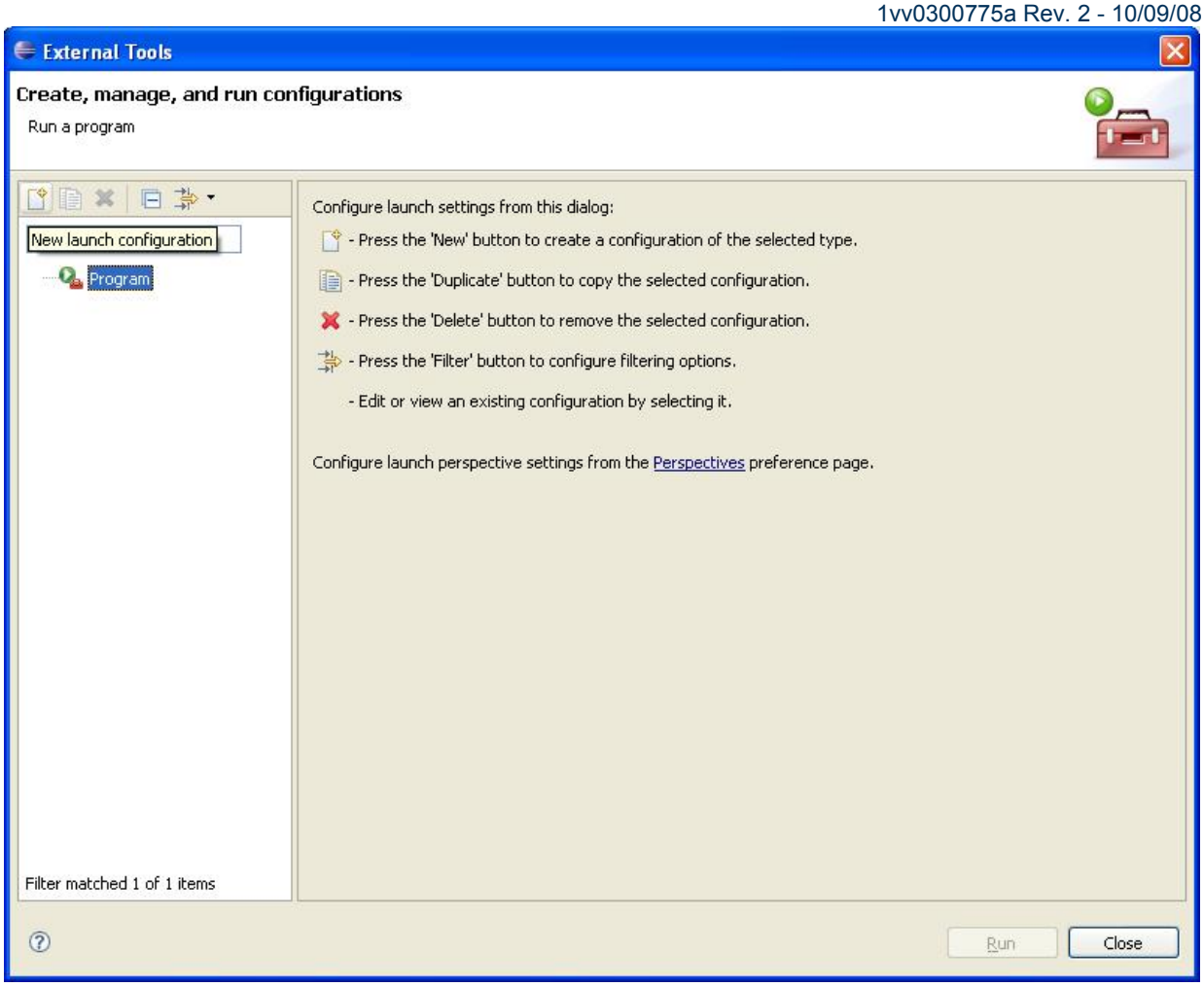
  - go to http://www.telit.com/ → Download Zone,
  - LOGIN with user and password provided by Telit.
  - Click on **Software Tools_GSM/GPRS,**
  - Click on **GE863-PRO³**
  - Click on the **ZIP_DOWNLOAD** of the: **Telit_GE863-PRO³_nonOS_Environment_config_files.zip**
  - **Save** it in a path like C:\Documents and Settings\user\Desktop.
  - **UnZip Telit_GE863-PRO³_nonOS_Environment_config_files.zip** (the unzip creates the directory **GE863-PRO3nonOS_Environment_config_files**)
  - Copy the files in **GE863-PRO3nonOS_Environment_config_files\OOCDLink**:
    - GE863-PRO3_ft2232.cfg
    - GE863-PRO3_debug_commands.script
    - GE863-PRO3_debug_commands_boot.script

    in the directory Ge863Pro3\Debug previus created.

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

- Now we have to configure the IDE in order to start the gdb server with the correct parameters; this server listens for requests from the gdb debugger and manages the communication with the drivers.
Open the IDE (**Start menu → All Programs → YAGARTO IDE → Eclipse Platform 3.3**) and go to the menu **Run → External Tools → Open External Tools Dialog**.

- On the left column of the window select the entry **Program** and click on the **New Launch Configuration** icon.

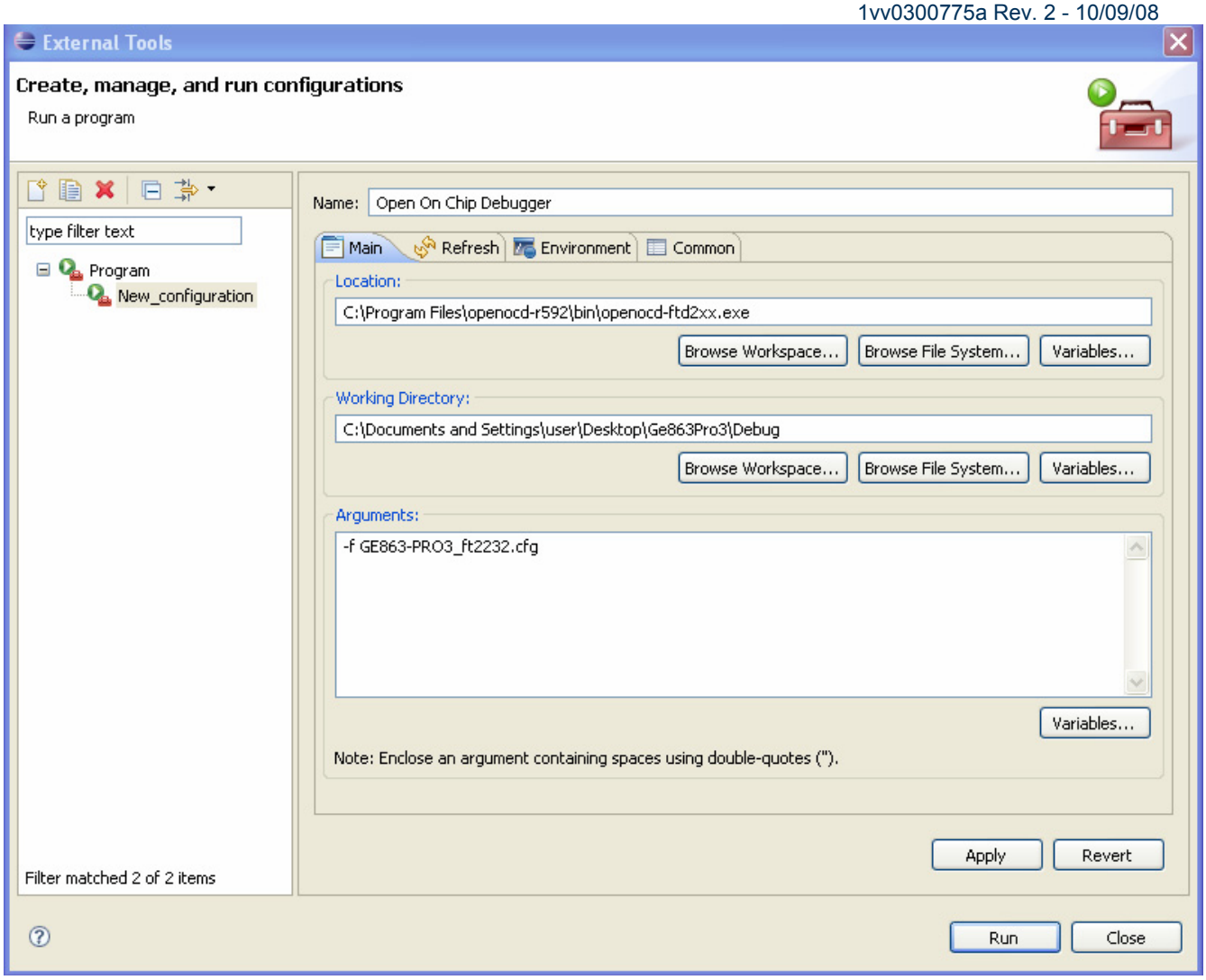


- Fill in the text field called **Name** with the value “Open On Chip Debugger“; select the tab **Main** and fill in the text field **Location** with the executable of the gdb server installed with the package Open On Chip Debugger. Fill in the text field **Working Directory** with the previously created directory (C:\Documents and Settings\user\Desktop\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 OOCDLink: if you have another emulator, you need to modify that file in a propriety way.



4.2 How to use SAM-ICE

SAM-ICE is a JTAG emulator from Segger that can be used to debug any AT91SAM 32-bit ARM-based microcontroller. To setup the debugging environment for using SAM-ICE follow these steps:

- In the host system create a directory called Ge863Pro3\Debug and copy in it the files: GE863-PRO3_debug_commands.script, GE863-PRO3_debug_commands_boot.script provided in the Telit package **Telit_GE863-PRO³_nonOS_Environment_config_files.zip** (it is not mandatory to put the files in that directory, but if you choose another directory you will have to modify some paths referred to in the rest of this section).
  To get those files following next steps:
  - go to http://www.telit.com/ → Download Zone,
  - LOGIN with user and password provided by Telit.
  - Click on **Software Tools_GSM/GPRS,**
  - Click on **GE863-PRO³**
  - Click on the **ZIP_DOWNLOAD** of the: **Telit_GE863-PRO³_nonOS_Environment_config_files.zip**
  - **Save** it in a path like C:\Documents and Settings\user\Desktop.
  - UnZip **Telit_GE863-PRO³_nonOS_Environment_config_files.zip** (the unzip creates the directory **GE863-PRO3nonOS_Environment_config_files**)
  - Copy the files in **GE863-PRO3nonOS_Environment_config_files\ SAM-ICE**:
    - GE863-PRO3_debug_commands.script
    - GE863-PRO3_debug_commands_boot.script

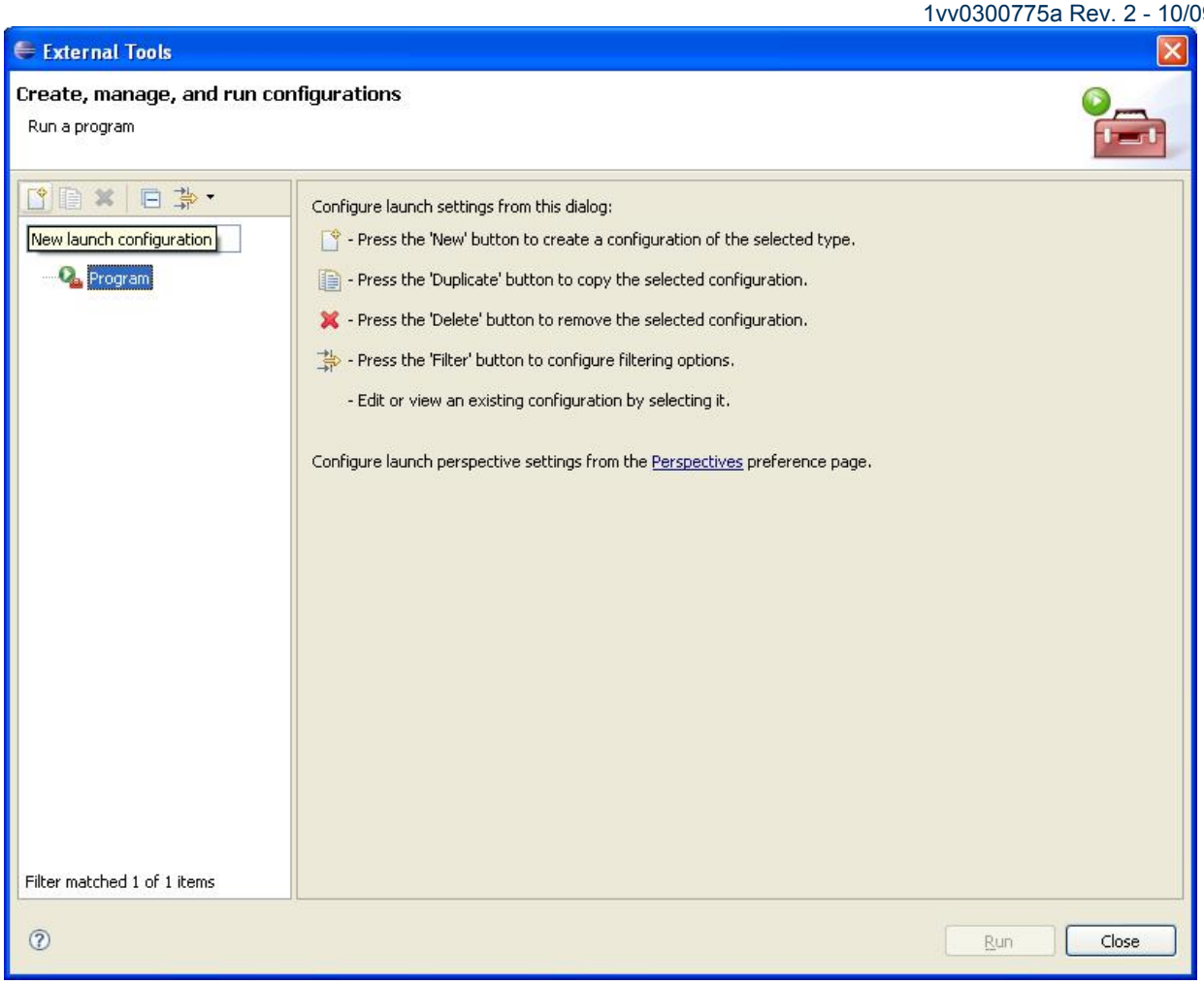
    in the directory Ge863Pro3\Debug previus created.

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

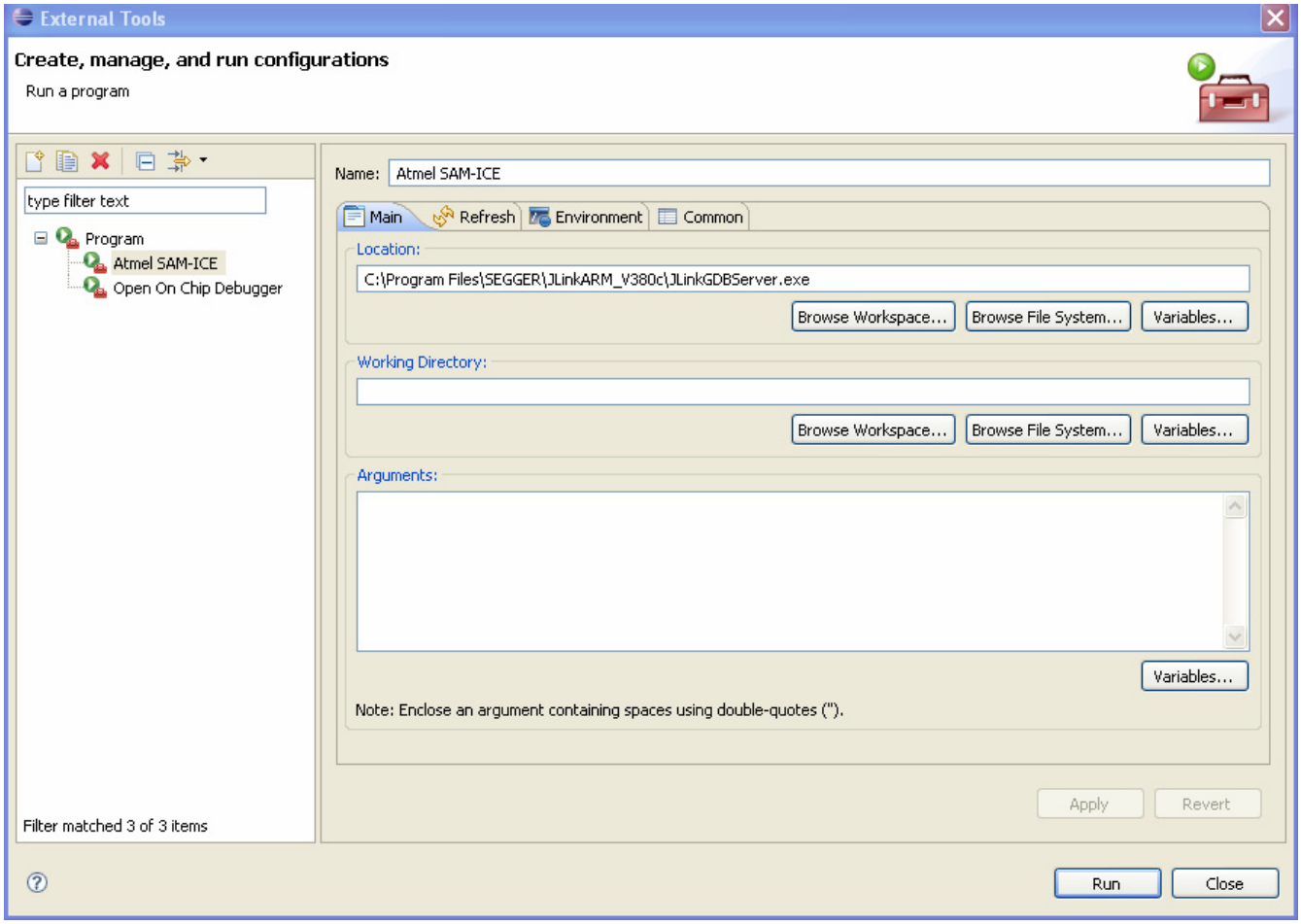
- Install the J-Link ARM toolkit provided with the emulator, or alternatively download and install the latest release from the Segger website; this toolkit contains the USB driver and some utilities, among which the gdb server which we will refer to in the rest of this paragraph.

- Plug the USB connection cable of SAM-ICE in the host system; the operating system should identify the new device and use the J-Link driver previously installed with the toolkit.

- Now we have to configure the IDE in order to start the gdb server. Open the IDE (**Start menu → All Programs → YAGARTO IDE → Eclipse Platform**) and go to the menu **Run → External Tools → Open External Tools Dialog**.
- On the left colum of the window select the entry **Program** and click on the **New Launch Configuration** icon.



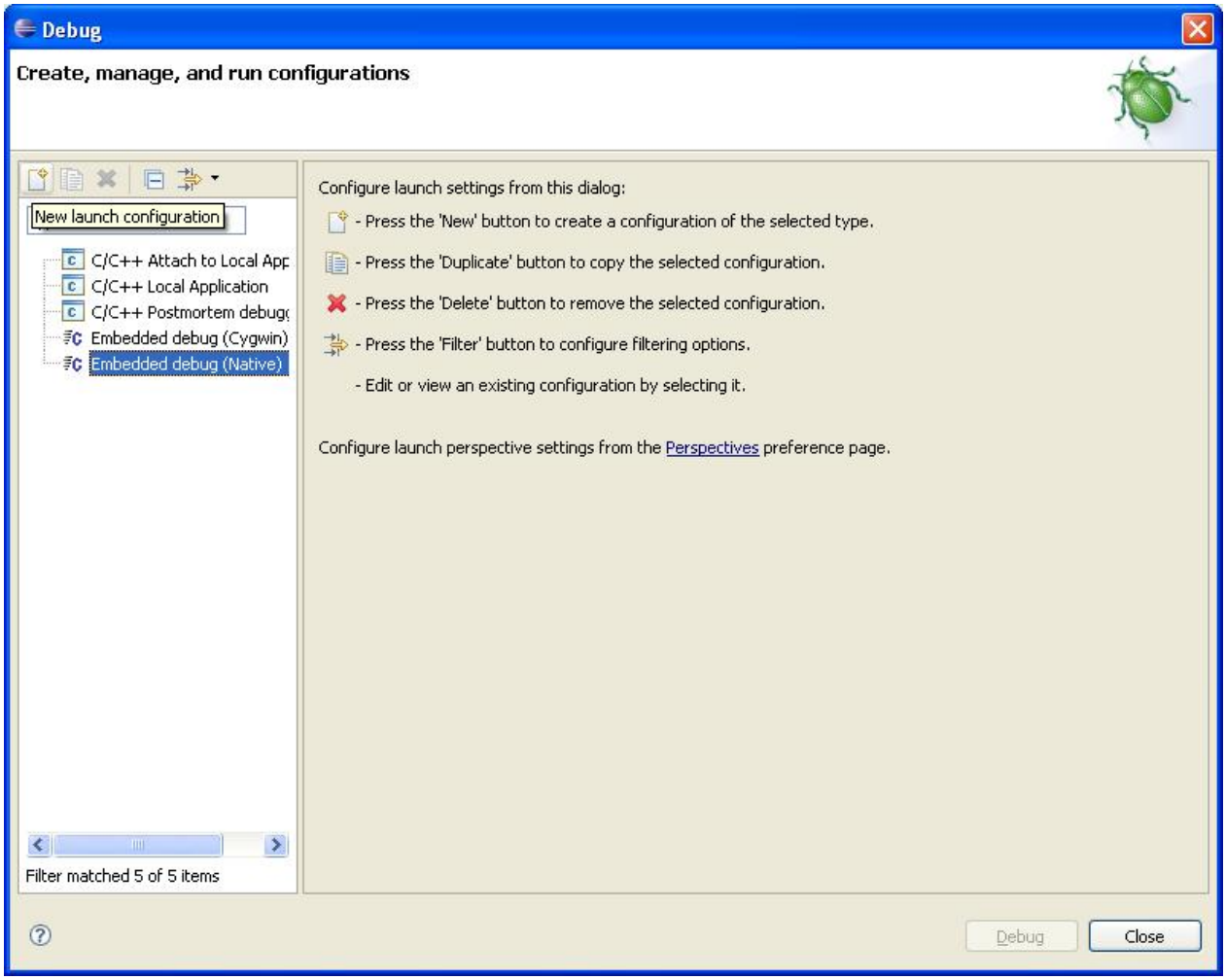
- Fill in the text field called **Name** with the value “Atmel SAM-ICE“; select the tab **Main** and fill in the text field **Location** with the executable of the gdb server installed with the J-Link ARM toolkit.
- Click on **Apply** and then click on **Close**.



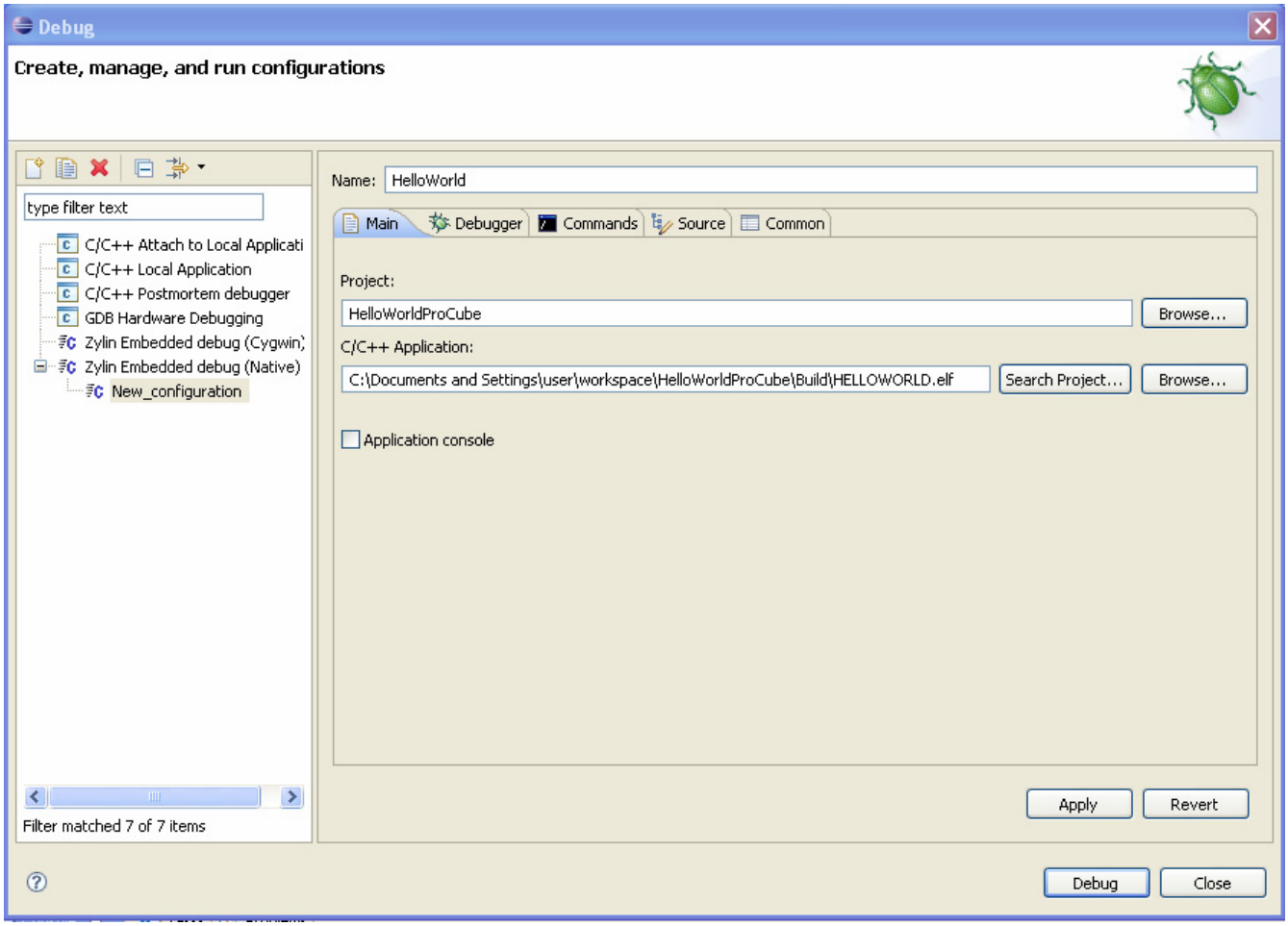
4.3 How load and debug an application in the target's RAM

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.

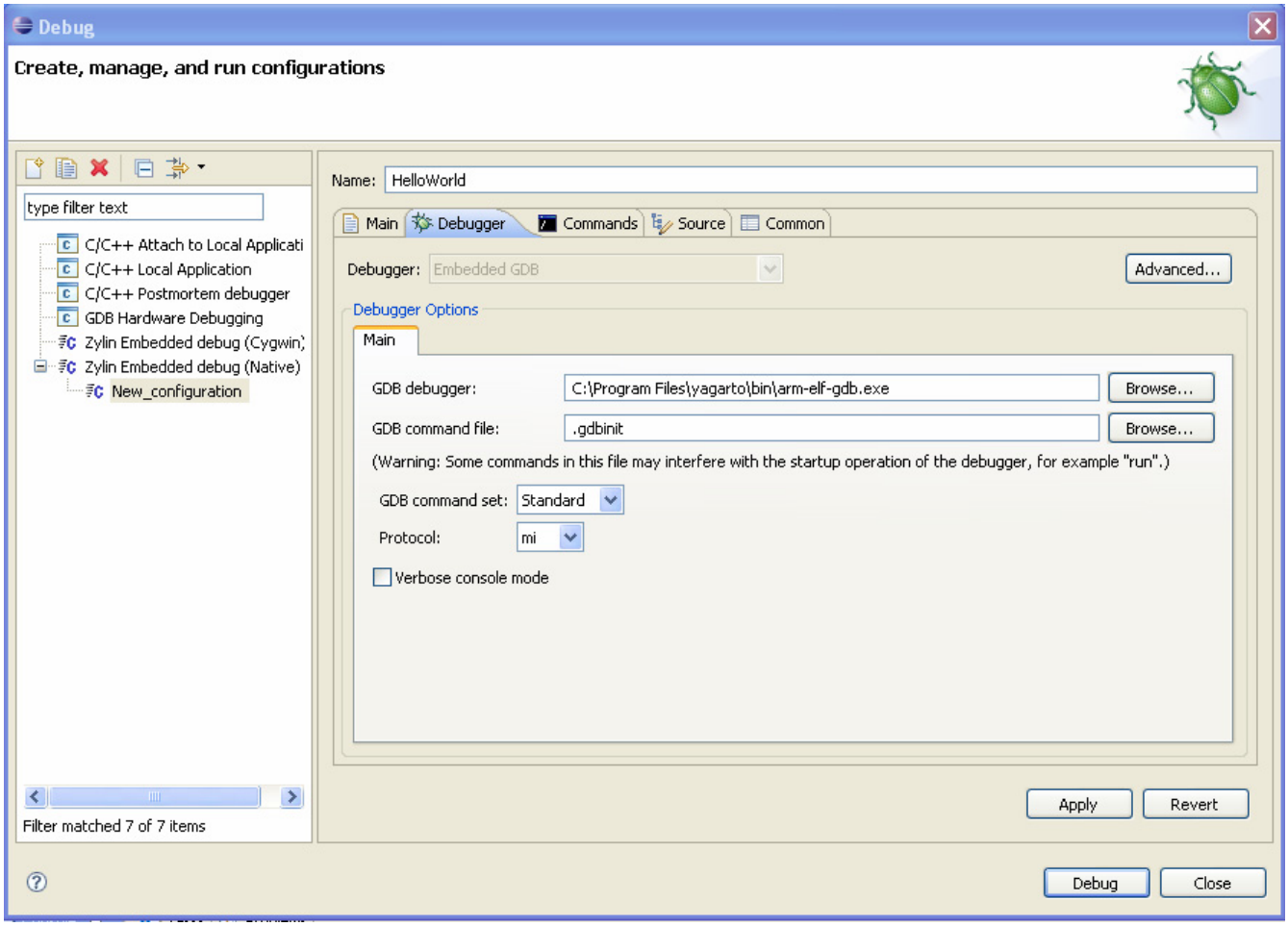
- Go to the menu **Run** and click on **Open Debug Dialog**.
- On the left side of the window select the entry **Embedded debug (Native)** and click on the **New Launch Configuration** icon.



- Fill in the text field called **Name** with the value “NonOS_lib“, select the tab **Main** and fill in the text field **Project** with the name of the project. Fill in the text field **C/C++ Application** with the path to the ELF file containing the application to debug and then click Apply.

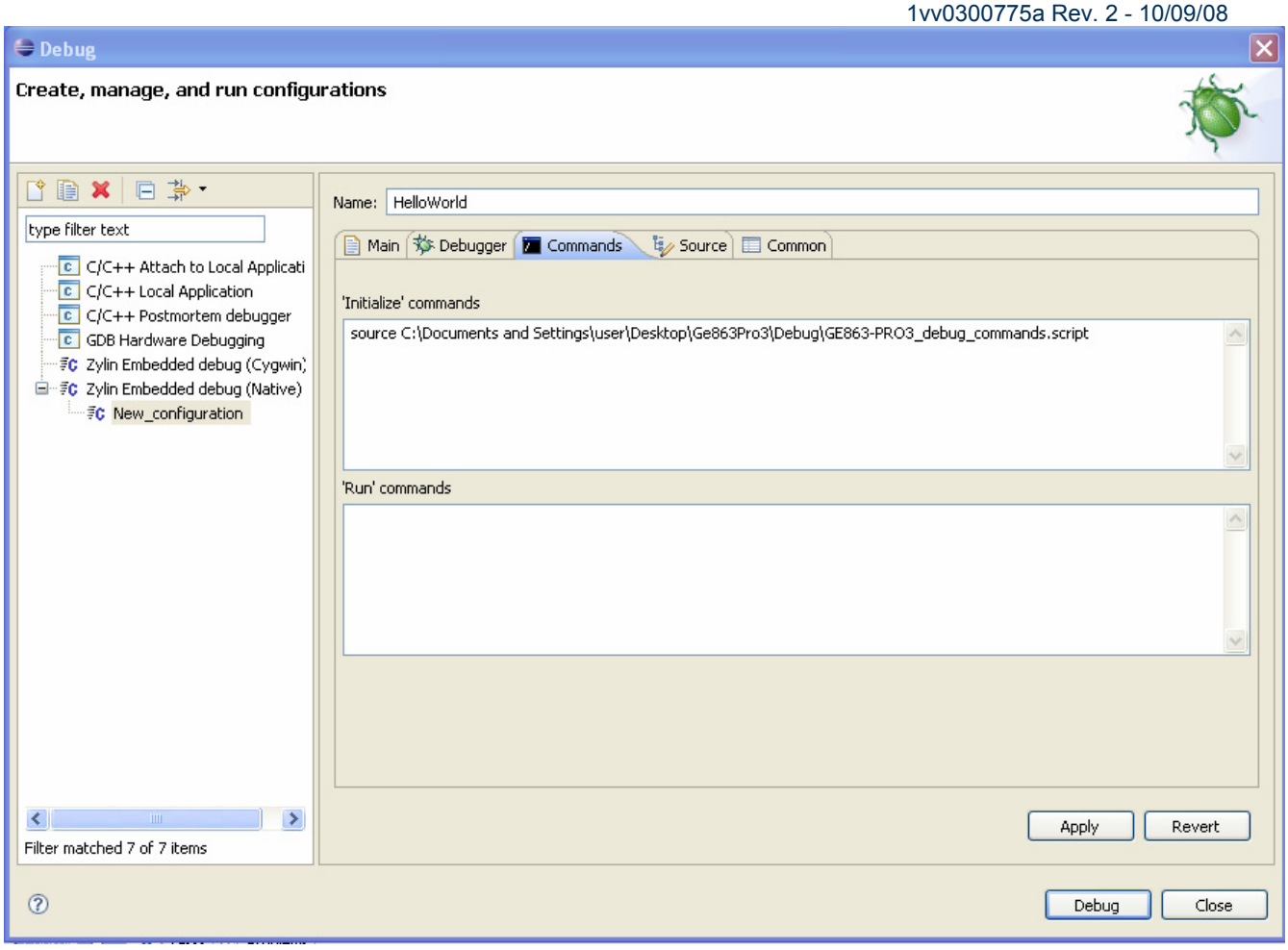


- Select the tab **Debugger**; fill in the text field **GDB Debugger** with the path to the gdb executable installed with Yagarto; delete any text contained in the text field **GDB command file**.



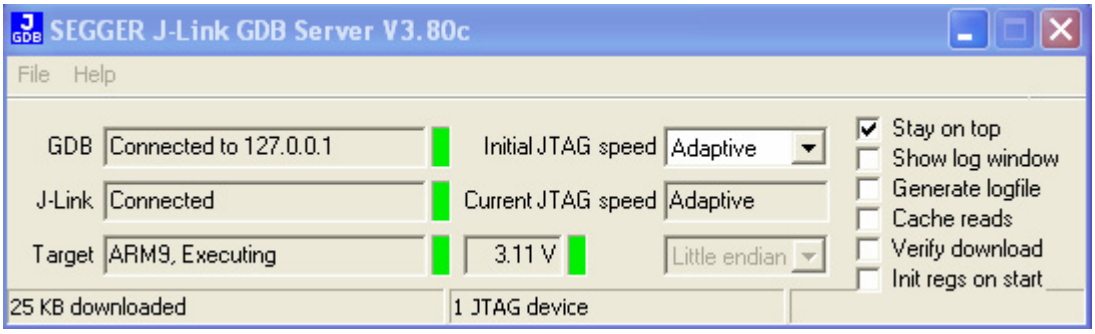
- Select the tab **Commands** and fill in the text field **'Initialize' commands** with a line that will make gdb execute at startup the commands contained in the file GE863-PRO3_debug_commands.script.

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



Now the debug environment is set up and you can start debugging your application.
Connect the emulator to your PC and to the JTAG port of the target.
It is convenient to open the IDE's debug perspective, so you can see which processes are running at any time. 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++**.
First, the gdb server you previously configured in the **External Tools** dialog box must be started: select the menu entry **Run → External Tools → Open External Tools Dialogs → Open On Chip Debugger** (or whatever menu entry you have, depending on the previously made configuration) → **Run**. You should see the related entry appear in the Debug view.

Note: if you chose to use the SAM-ICE emulator, when you start the relevant external tool a new window appears: on that window select **Adaptive** from the Initial JTAG speed drop-down list; also, if you want this window not to be always on top of other windows, uncheck the option **Stay on top**. These settings, along with all the other settings you may want to change, will remain stored in a configuration file, so there is no need to change them every time you launch this tool.

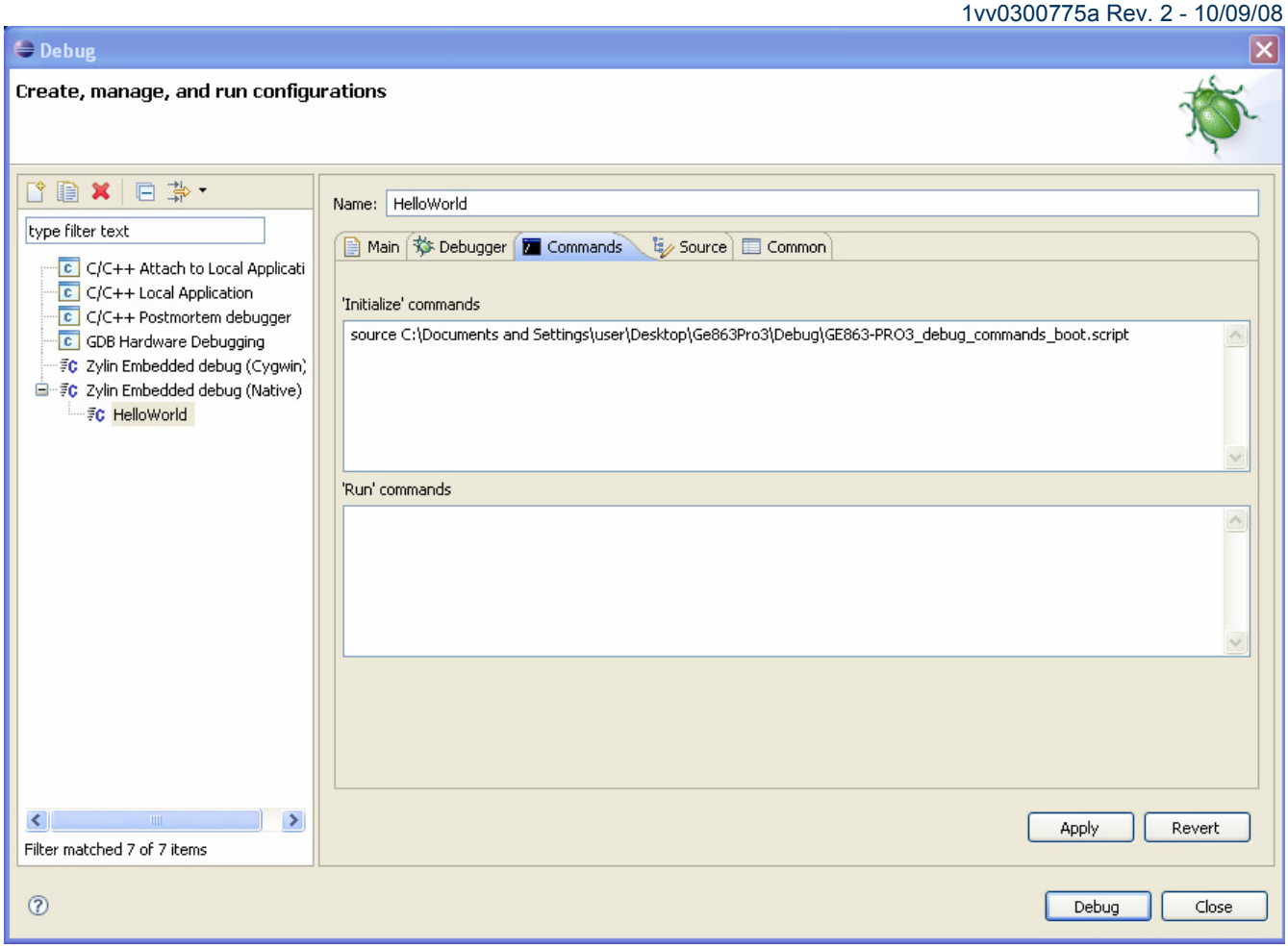


Now, gdb itself must be launched: in the Eclipse window go to **Run** → **Open Debug Dialog**. On the left side of the window select the icon **HelloWorld**, corresponding to the gdb configuration you just made, and click on **Debug** (from now on, you can start the debugger with this configuration also 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 be difficult because the program may not follow the execution flow you would expect.
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 **Open Debug Dialog** 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 a terminal emulator open 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.

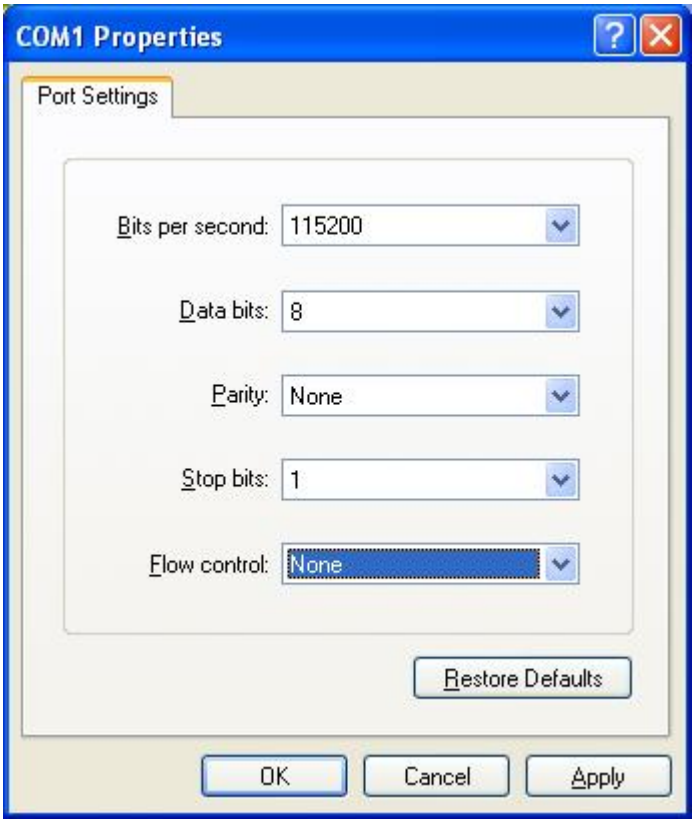
5 Using U-Boot

U-Boot is a popular bootloader that offers many features useful to manage an embedded system. You can load an operating system (or whatever application you want to deploy), but you can also transfer files from a remote system to the target and store these files in non volatile memory. In this chapter you will learn how to perform these tasks.
The user's interface of U-Boot is text-based and can be accessed through a terminal emulator. As an example, we will describe how to use the HyperTerminal application, which is included in the Windows operating system.

- Connect a serial port of the host system to the serial port of the target called AP DEBUG
- Go to the Windows menu **Start** → **All Programs** → **Accessories** → **Communications** → **HyperTerminal** to open the application.
- In the window named **Connection Description** enter a name for your connection (for example Ge863Pro3) and click on **OK**.

- In the window **Connect to** select the serial port of your host system (if your PC has only a serial port, it should be COM1) and than click on **OK**.

- In the next window select 115200 bits per second, 8 data bits, no parity bit, 1 stop bit and no flow control, then click on **OK**.



Now that you are ready to use U-Boot reset the target and you should see U-Boot's output messages. You can save the current configuration by clicking on **File** → **Save** and the next time you will want to use HyperTerminal, you can retrieve this configuration by clicking on the Windows menu item **Start** → **All Programs** → **Accessories** → **Communications** → **HyperTerminal** → **Ge863Pro3.ht** (in case you've saved the configuration with that name).

5.1 How to flash an application

In you want to flash an application in the target using U-Boot please follow the next steps:

- Load the application with one of the load commands below:
  - loady: for loading a file with the Ymodem protocol.
  - loadb: for loading a file with the Kermit protocol.

  Issue the chosen command with the RAM Address where you want to copy the file (possible values start from 0x20012000). U-Boot is now ready to accept incoming data.

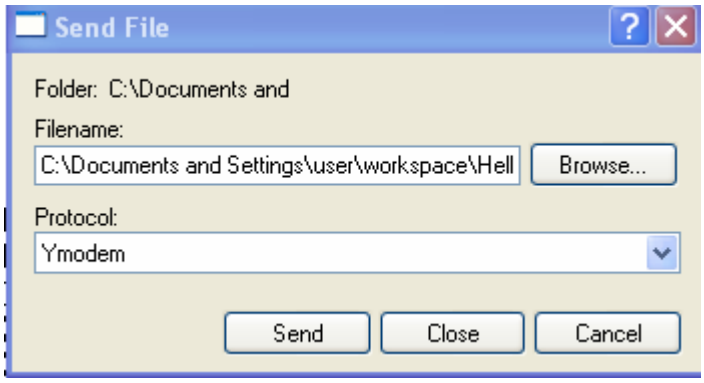
- Send the file from the host with the chosen protocol. If everything went on properly, at the end of the transfer the number of transferred bytes is displayed both in decimal and hexadecimal format in the U-Boot shell.

- Issue the cp.b command with the RAM address where you previously copied the file, the flash address where you want to store the file (possible values start from 0xD0021000) and the number of bytes to be copied.

Example: to transfer a file with size of SIZE bytes with the Ymodem protocol and store it in the flash at the virtual address 0xD0021000 (that corresponds to the 0x21000 offset from flash beginning) type the following command:

```
loady 0x20012000
```

Then click on the HyperTerminal's menu item **Transfer** → **Send file…**; in the window that appears browse to the file you want to transfer and select the protocol Ymodem.



Wait until the transfer is done, then type the following command to store the file in flash:


```
cp.b 0x20012000 0xD0021000 SIZE
```

5.2 How to start an application

Since the connection between the ARM processor and the flash memory is achieved through a serial interface, the target cannot execute the application directly from the flash. So, the binary image has to be loaded in RAM and executed from there. To launch an application in the target using U-Boot follow these steps:

- Load the application in RAM in one of the following two ways:
  - use the loady and loadb commands as explained in the previous paragraph;
  - if you have in flash a previously stored application, load it using the cp.b command specifying the flash address where the application is stored, the RAM address where it is to be copied and the number of bytes to be copied.
    Example:
    `cp.b 0xD0021000 0x20012000 SIZE`

- Issue the go command with the correct RAM address.
  Example:

  go `0x20012000`

After this command your application should start running.

5.3 How to automatically start an application from flash in the boot sequence

If you want the target to automatically start your application, U-Boot must be instructed to load the binary image from flash to RAM and execute it from there.

- In the U-Boot shell, change the environment variable bootcmd which contains the commands that must be automatically executed at startup. If we suppose to have copied an application of size SIZE bytes in the flash address 0xD0021000 we can use the following command:

  ```
  setenv bootcmd cp 0xD0021000 0x20012000 SIZE\;go 0x20012000
  ```

- Save the environment in the flash with the following command:
  `saveenv`

- When you reboot the target, U-Boot waits for a timeout (by default 3 seconds) to expire, and if during this interval you don't type anything in the shell, you will see your application running.

For further details on U-Boot usage you can refer to the home page of the project http://sourceforge.net/projects/u-boot.