Eclipse development with GNU Toolchain

Version 1.0

RONETIX

embedded development tools

Ronetix GmbH
Waidhausenstrasse 13/5
1140 Vienna
Austria
Tel: +43-720-500315
+43-1962-720 500315
Fax: +43-1- 8174 955 3464
Internet: www.ronetix.at
E-Mail info@ronetix.at

Acknowledgments:

Change log

December 2008	- First release

1 Introduction

This User Manual will show you how to install the GNU Toolchain and Eclipse, how to compile and debug a simple example using the Ronetix Evaluation board EB9263 with an Atmel AT91SAM9263 and PEEDI JTAG Emulator and Flash Programmer.

The necessary software components for an ARM cross development are:

- **GNU toolchain (compiler, linker, gdb)**
- **Eclipse IDE for C/C++ developers**
- **Java Runtime**

To enable a quick start in the ARM cross development Ronetix provides pre-built packages for Linux and Windows hosts.

The necessary files for a Linux host are:

- The GNU toolchain:
  http://download.ronetix.info/toolchains/arm/ronetix-gnutools-arm-elf-4.1.1-linux.tar.bz2

- The Eclipse IDE:
  http://download.ronetix.info/eclipse/eclipse-cpp-ganymede-SR1-linux-gtk.tar.gz

- A simple example:
  http://download.ronetix.info/examples/basic-dhrystone-project-1.4-pm9263.tar.bz2

2 PEEDI JTAG Emulator Installation

PEEDI (Powerful Embedded Ethernet Debug Interface) is an EmbeddedICE solution that enables you to debug software running on ARM processor cores via the JTAG port.
In order to debug you need to configure PEEDI JTAG Emulator. The configuration of PEEDI is common for both toolchains for Linux and Windows.

You can find detailed information about PEEDI in the PEEDI's User Manual:
http://download.ronetix.info/peedi/doc/peedi_rev.A_manual.pdf

- Connect PEEDI to a free port of your LAN switch/hub using the supplied UTP patch cable.
- Connect PEEDI to the target using a JTAG cable and if needs the one of the supplied JTAG adapters. The JTAG adapter must be on the PEEDI side of the JTAG cable:

- Connect PEEDI to a COM port of your PC using the RS232 cable. Start any kind of terminal emulation program (HyperTerminal) and set it to 115200 bauds, 8 data bits, no parity and no flow control.
- Restart PEEDI holding pressed both front panel buttons to enter RedBoot command line.
- Use **fconfig** command to set the network configuration and other parameters.

WARNING:

If PEEDI is set to get its network settings from a DHCP server and if the Ethernet cable is unplugged or there is no DHCP server on the Ethernet, it may take some time for PEEDI to boot. To avoid this, make sure PEEDI can reach a DHCP server or assign a static IP address.

- Restart PEEDI again for the changes to take effect

After PEEDI is up and running (this should take some seconds after reset), press and hold the green front panel button and PEEDI will start to display its IP address on the display.

Connect to PEEDI with telnet application using the IP address from the previous statement. If connected, you should see the PEEDI CLI prompt

3 Toolset installation on Linux

To install the pre-built from Ronetix GNU cross-development tools:

Download the GNU tools form here:

http://download.ronetix.info/toolchains/arm/ronetix-gnutools-arm-elf-4.1.1-linux.tar.bz2

or get it from the CD.

Uncompress the archive

```
Cd /
tar xvfj ronetix-gnutools-arm-elf-4.1.1-linux.tar.bz2
```

The toolchain will be installed in the /usr/cross/arm directory. If want to install the toolset in another directory make sure you have a symbolic link in the /usr/cross

1. Set a path to the /usr/cross/arm/bin: in the .bashrc file add the following:

```
export PATH=$PATH:/usr/cross/arm/bin
```

2. Test the toolchain installation:

```
[linbox]$ arm-elf-gcc -v
Using built-in specs.
Target: arm-elf
Configured with: /home/src/cross/gcc-4.1.1/configure --target=arm-elf --build i686-pc-linux-gnu --host i686-pc-linux-gnu --disable-nls --with-float=soft --prefix=/usr/cross/arm-elf --enable-interwork --enable-multilib --enable-languages=c,c++ --with-newlib --enable-win32-registry=ronetix-arm --with-gnu-as --with-gnu-ld --with-headers=/home/src/cross/newlib-1.14.0/newlib/libc/include
Thread model: single
gcc version 4.1.1
```

3. Installing java Runtime Environment (JRE)

Download it from here or get it from the CD:

http://download.ronetix.info/eclipse/jre-6u6-linux-i586.bin

```
cd /user/local
sh jre-6u6-linux-i586.bin
```

4. Installing Eclipse IDE

Download the Eclipse IDE from here or get it form the CD:

http://download.ronetix.info/eclipse/eclipse-cpp-ganymede-SR1-linux-gtk.tar.gz

```
cd /usr/local
tar xvfz eclipse-cpp-ganymede-SR1-linux-gtk.tar.gz
cd eclipse
ln -s ../jre1.6.0_06 jre
```

Set a path to the /usr/local/eclipse: in the .bashrc file add the following:

```
export PATH=$PATH:/usr/local/eclipse
```

5. Installing an example

Download the EB9263 example from here:

http://download.ronetix.info/examples/basic-dhrystone-project-1.4-pm9263.tar.bz2

or get it from the CD.

```
Cd
mkdir workspace
cd workspace
tar xvfj basic-dhrystone-project-1.4-pm9263.tar.bz2
```

At this point you should be able to build, debug and run applications on embedded ARM targets.

You can compile and debug the example manual, from the shell prompt or using Eclipse. The working with Eclipse is explained in “Section 4: Working with Eclipse” from this manual.

6. Compiling from the shell

```
Cd basic-dhrystone-project-1.4-pm9263/basic-dhrystone-project

[linbox]$ make
arm-elf-gcc -g -O0 -I../at91lib/boards/pm9263 -I../at91lib/peripherals
-I../at91lib/components -I../at91lib -Dat91sam9263 -D__ASSEMBLY__ -
Dsram -c -o obj/sram_board_cstartup.o
../at91lib/boards/pm9263/board_cstartup.S
arm-elf-gcc -Wall -ffunction-sections -g -O0 -I../at91lib/boards/pm9263
-I../at91lib/peripherals -I../at91lib/components -I../at91lib -
Dat91sam9263 -Dsram -c -o obj/sram_main.o main.c
arm-elf-gcc -Wall -ffunction-sections -g -O0 -I../at91lib/boards/pm9263
-I../at91lib/peripherals -I../at91lib/components -I../at91lib -
Dat91sam9263 -Dsram -c -o obj/sram_dhry_1.o dhry_1.c
arm-elf-gcc -Wall -ffunction-sections -g -O0 -I../at91lib/boards/pm9263
-I../at91lib/peripherals -I../at91lib/components -I../at91lib -
Dat91sam9263 -Dsram -c -o obj/sram_dhry_2.o dhry_2.c
arm-elf-gcc -Wall -ffunction-sections -g -O0 -I../at91lib/boards/pm9263
-I../at91lib/peripherals -I../at91lib/components -I../at91lib -
Dat91sam9263 -Dsram -c -o obj/sram_stdio.o ../at91lib/utility/stdio.c
arm-elf-gcc -Wall -ffunction-sections -g -O0 -I../at91lib/boards/pm9263
-I../at91lib/peripherals -I../at91lib/components -I../at91lib -
Dat91sam9263 -Dsram -c -o obj/sram_dbgu.o
../at91lib/peripherals/dbgu/dbgu.c
arm-elf-gcc -Wall -ffunction-sections -g -O0 -I../at91lib/boards/pm9263
-I../at91lib/peripherals -I../at91lib/components -I../at91lib -
Dat91sam9263 -Dsram -c -o obj/sram_pio.o
../at91lib/peripherals/pio/pio.c
arm-elf-gcc -Wall -ffunction-sections -g -O0 -I../at91lib/boards/pm9263
-I../at91lib/peripherals -I../at91lib/components -I../at91lib -
Dat91sam9263 -Dsram -c -o obj/sram_rtt.o
../at91lib/peripherals/rtt/rtt.c
arm-elf-gcc -Wall -ffunction-sections -g -O0 -I../at91lib/boards/pm9263
-I../at91lib/peripherals -I../at91lib/components -I../at91lib -
Dat91sam9263 -Dsram -c -o obj/sram_board_memories.o
../at91lib/boards/pm9263/board_memories.c
arm-elf-gcc -Wall -ffunction-sections -g -O0 -I../at91lib/boards/pm9263
-I../at91lib/peripherals -I../at91lib/components -I../at91lib -
Dat91sam9263 -Dsram -c -o obj/sram_board_lowlevel.o
../at91lib/boards/pm9263/board_lowlevel.c
arm-elf-gcc -g -O0 -nostartfiles -Wl,--gc-sections -
T"../at91lib/boards/pm9263/at91sam9263/sram.lds" -o bin/basic-
dhrystone-project-pm9263-at91sam9263-sram.elf obj/sram_board_cstartup.o
obj/sram_main.o obj/sram_dhry_1.o obj/sram_dhry_2.o obj/sram_stdio.o
obj/sram_dbgu.o obj/sram_pio.o obj/sram_rtt.o obj/sram_board_memories.o
obj/sram_board_lowlevel.o
```

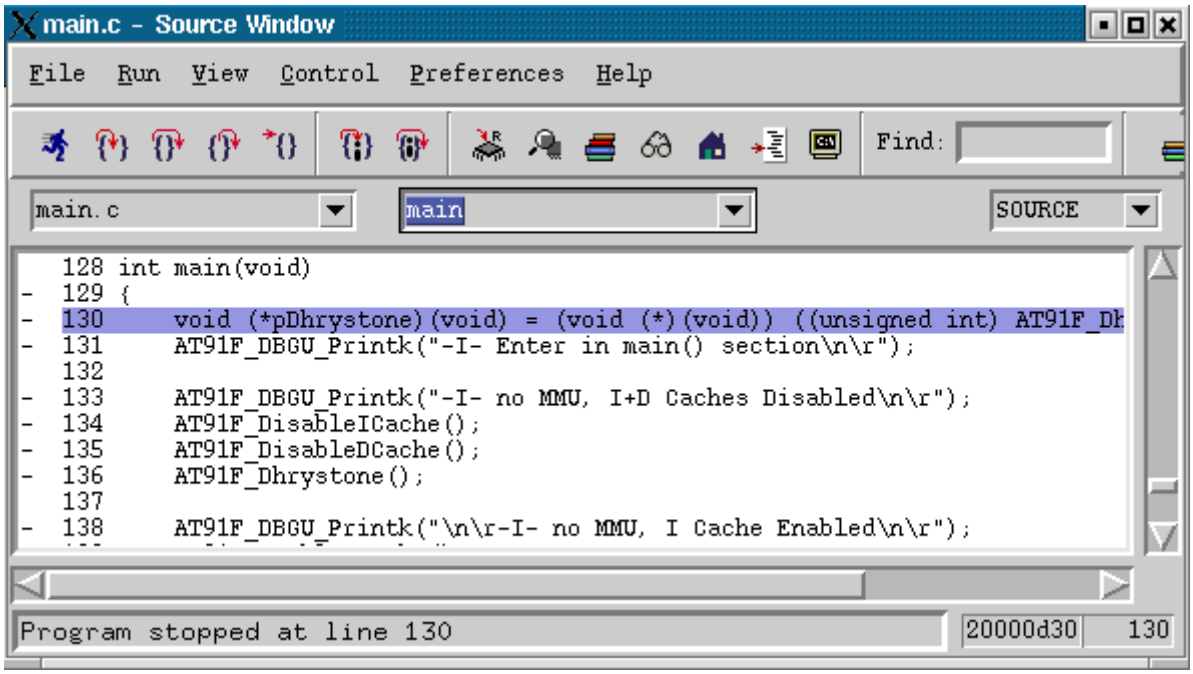
```
arm-elf-objcopy -O binary bin/basic-dhrystone-project-pm9263-
at91sam9263-sram.elf bin/basic-dhrystone-project-pm9263-at91sam9263-
sram.bin
arm-elf-size obj/sram_board_cstartup.o obj/sram_main.o
obj/sram_dhry_1.o obj/sram_dhry_2.o obj/sram_stdio.o obj/sram_dbgu.o
obj/sram_pio.o obj/sram_rtt.o obj/sram_board_memories.o
obj/sram_board_lowlevel.o bin/basic-dhrystone-project-pm9263-
at91sam9263-sram.elf
   text    data     bss     dec     hex filename
    304       0       0     304     130 obj/sram_board_cstartup.o
   1824       0       0    1824     720 obj/sram_main.o
   1536       0       4    1540     604 obj/sram_dhry_1.o
   1212       0       0    1212     4bc obj/sram_dhry_2.o
   3140    1028       0    4168    1048 obj/sram_stdio.o
    640       0       0     640     280 obj/sram_dbgu.o
   1408       0       0    1408     580 obj/sram_pio.o
    416       0       0     416     1a0 obj/sram_rtt.o
   1960       0       0    1960     7a8 obj/sram_board_memories.o
    640       0       0     640     280 obj/sram_board_lowlevel.o
  12116       0   10240   22356    5754 bin/basic-dhrystone-project-
pm9263-at91sam9263-sram.elf
```

7. Debugging with Insight

To simplify the debugging there is a file '`.gdbinit`' in the project directory which is read when you run gdb/insight. Adjust the '`~/workspace/basic-dhrystone-project-1.4-pm9263/basic-dhrystone-project/.gdbinit`' to match the IP address of PEEDI.

From a X-console start the Insight debugger:

```
[linbox] arm-elf-insight bin/at91sam9263-ek-pm9263-sram.elf
```



The following descriptions discuss the use of the default debugger toolbar buttons.

The **Run** button, do not use this button. Instead of this use the "**Continue**" button

During the debugging process, the **Run** button turns into the **Stop** button to interrupt the debugging.

The **Continue** button continues execution until a breakpoint, watchpoint or exception is encountered, or until execution completes.

The **Step** button steps to next executable line of source code. Also, the **Step** button steps into called functions.

The **Next** button steps to the next executable line of source code in the current file. Unlike the **Step** button, the **Next** button steps over called functions.

The **Finish** button finishes execution of a current frame. If clicked while in a function, it finishes the function and returns to the line that called the function.

Download Performance

By default, GDB versions previous than v6.5.50 (from 06.08.2006) download program code and data in small packets that are not necessarily a multiple of four bytes in length. This causes program download times to be slower than necessary, especially with ARM targets. There are two GDB internal variables that affect this. To improve GDB download performance, you should set the download write size to a binary value 16KB and the memory write packet size to a larger value to allow for packet overhead (+100 bytes is plenty). For example, to download 16KB at a time:

```
(gdb) set remote memory-write-packet-size fixed
(gdb) set remote memory-write-packet-size 16384
(gdb) set download-write-size 16128
```

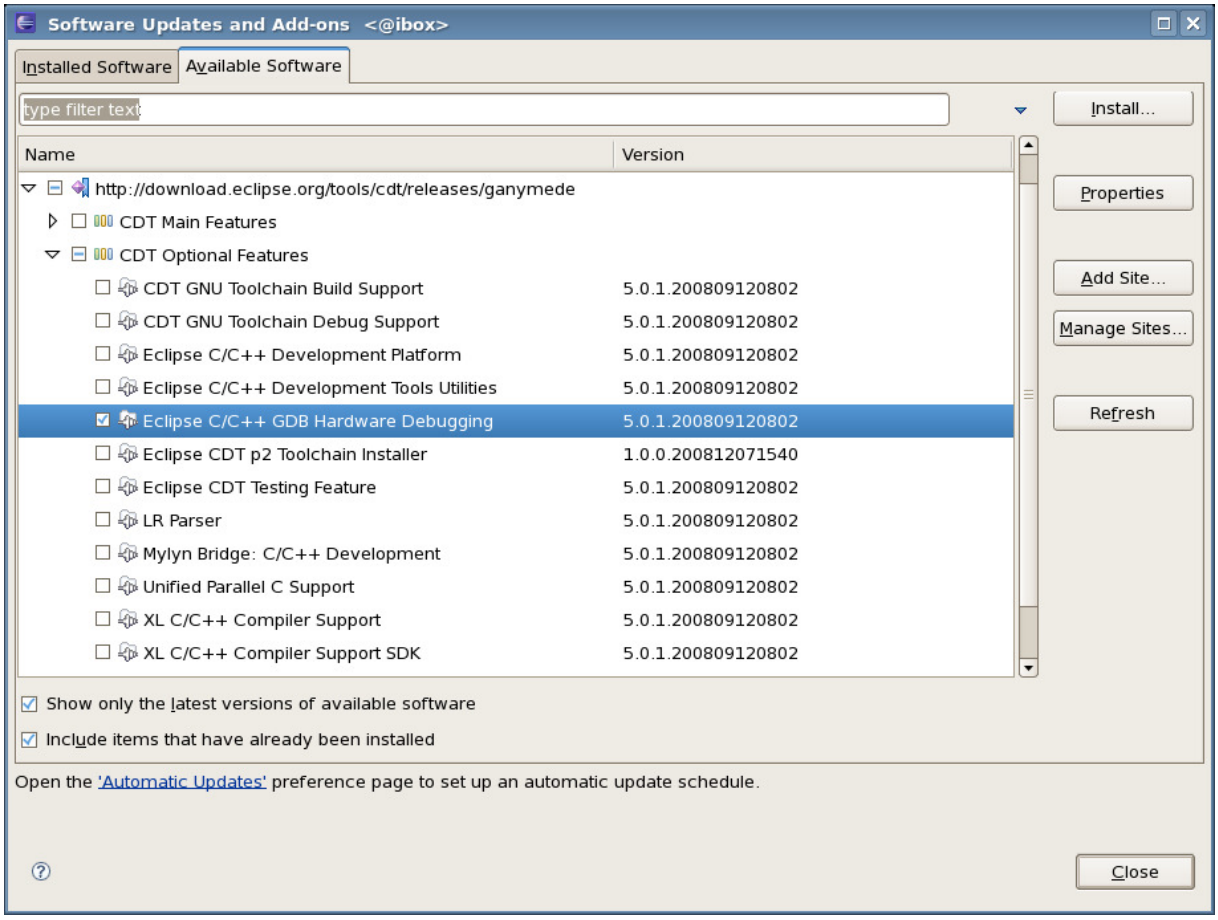
Note:

No error is returned to GDB in case of an invalid 4 byte memory read access during a period defined by ***GDB_READ_INGNORE_TIME*** *in the target configuration file. This is because GDB refuses to connect to the target if it gets an error during this connection sequence. GDB tries to read the stack frame during the connection sequence and this may lead to invalid memory access.*

4 Working with Eclipse

4.1 Installing the "Eclipse C/C++ Hardware Debugging" Add-on

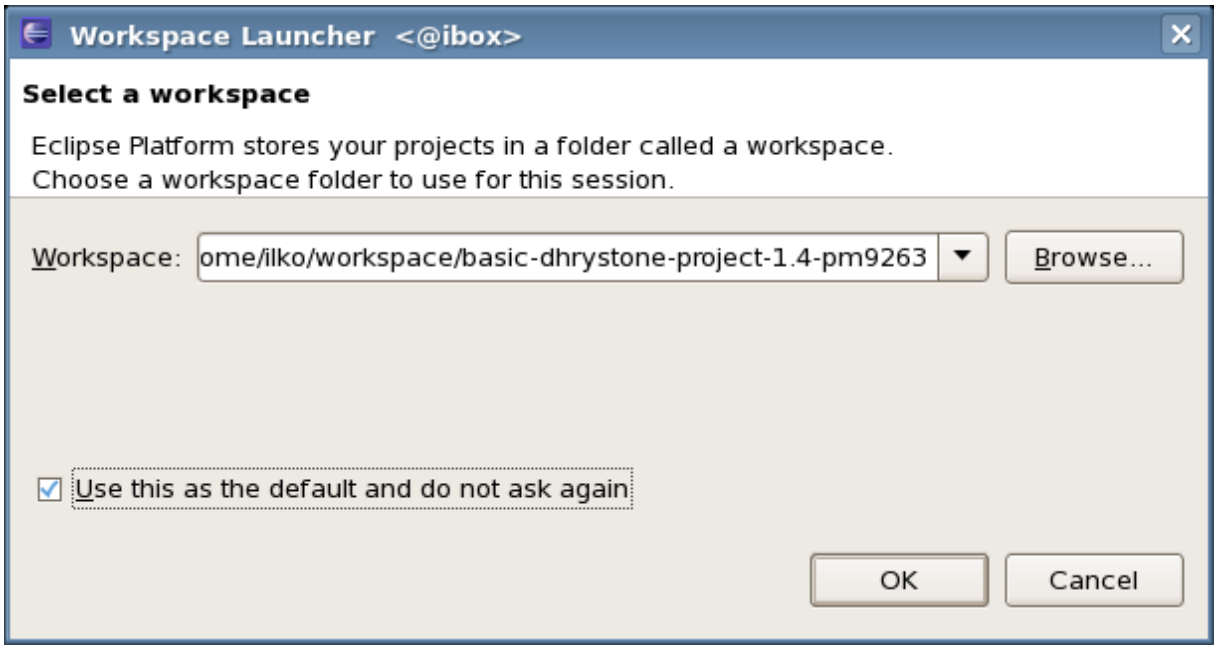
Start Eclipse and click "Help -> Software Updates -> Available Software" and install "Eclipse C/C++ hardware Debugging":



4.2 Adding a project

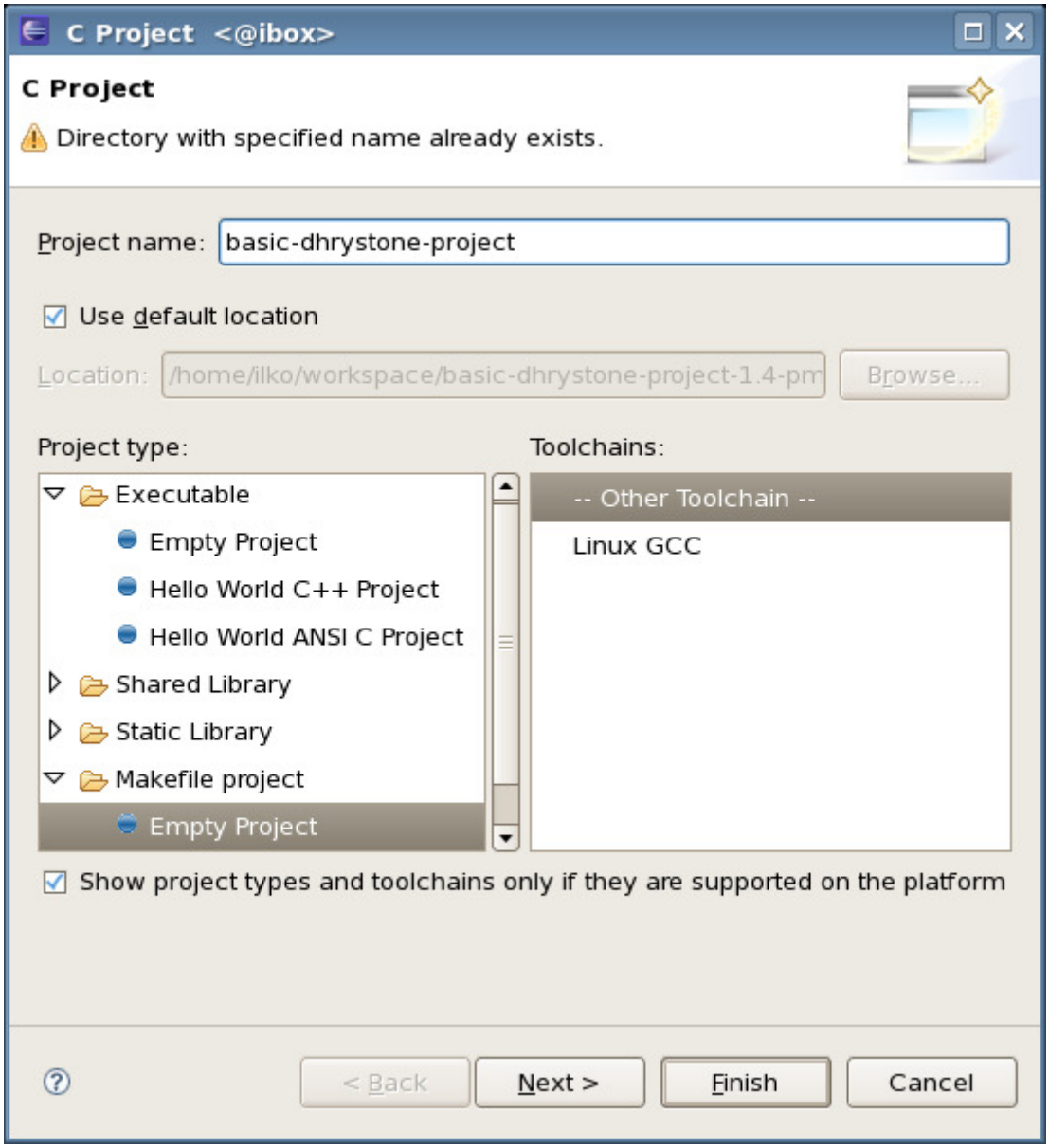
Eclipse (http://www.eclipse.org) is an open-source, Java based, powerful Integrated Development Environment (IDE). Adding the CDT plug-in (C/C++ Development Toolkit), you can edit and build C programs using the GNU compiler toolkit. A detailed C/C++ Development User Guide for Eclipse can be downloaded from http://www.eclipse.org/cdt.

Start the Eclipse and the "Workspace Launcher" dialog should appear where you need to point the workspace folder. In our case point "/home/ilko/workspace/basic-dhrystone-project-1.4-pm9263", this way eclipse will automatically include the project files when we create a new project. Next click OK.

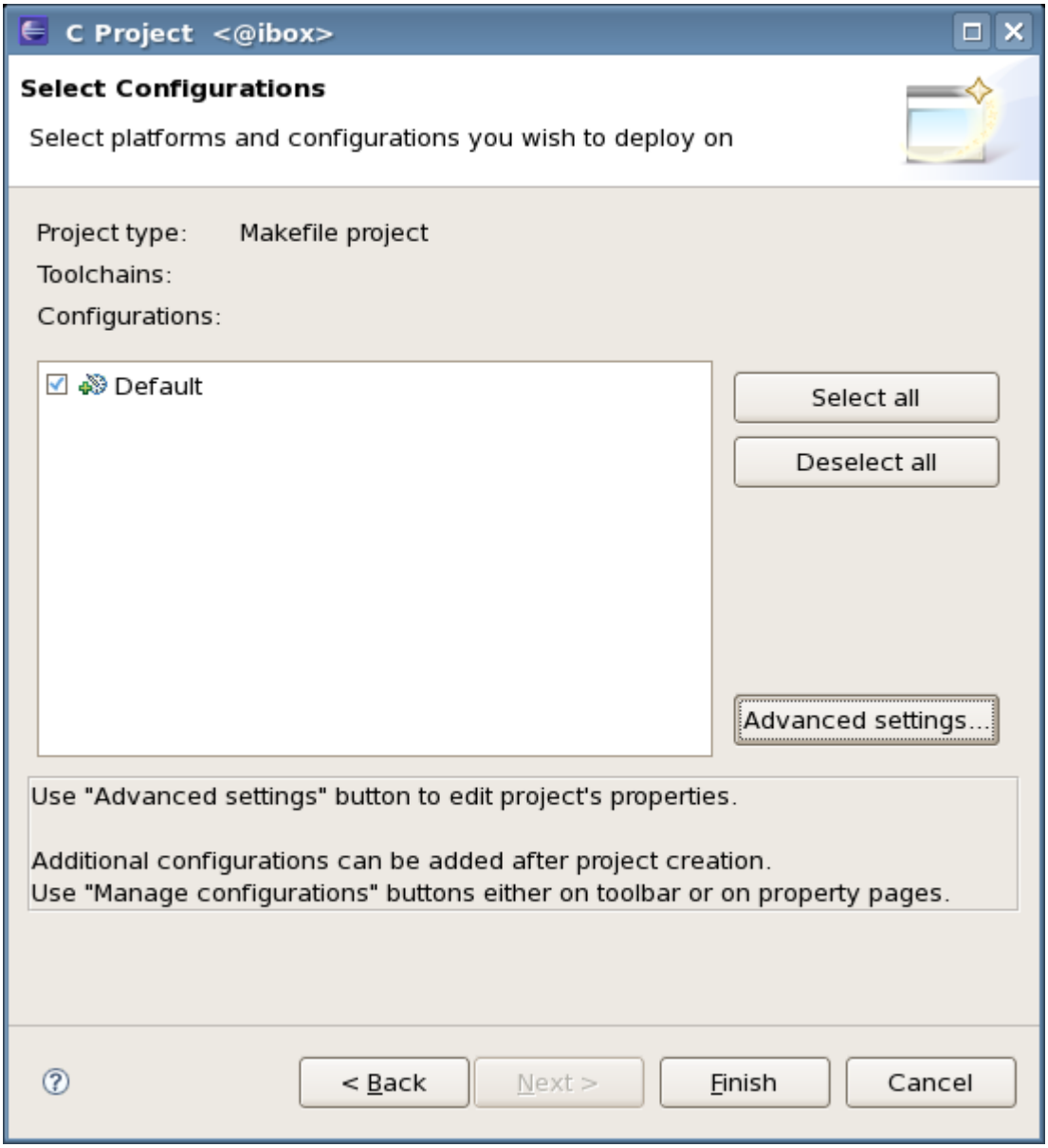

Workspace Launcher <@ibox>
Select a workspace
Eclipse Platform stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.
Workspace:
ome/ilko/workspace/basic-dhrystone-project-1.4-pm9263
Browse...
Use this as the default and do not ask again
OK
Cancel

Now you need to add as projects the basic-dhrystone-project-1.4-pm9263 example.

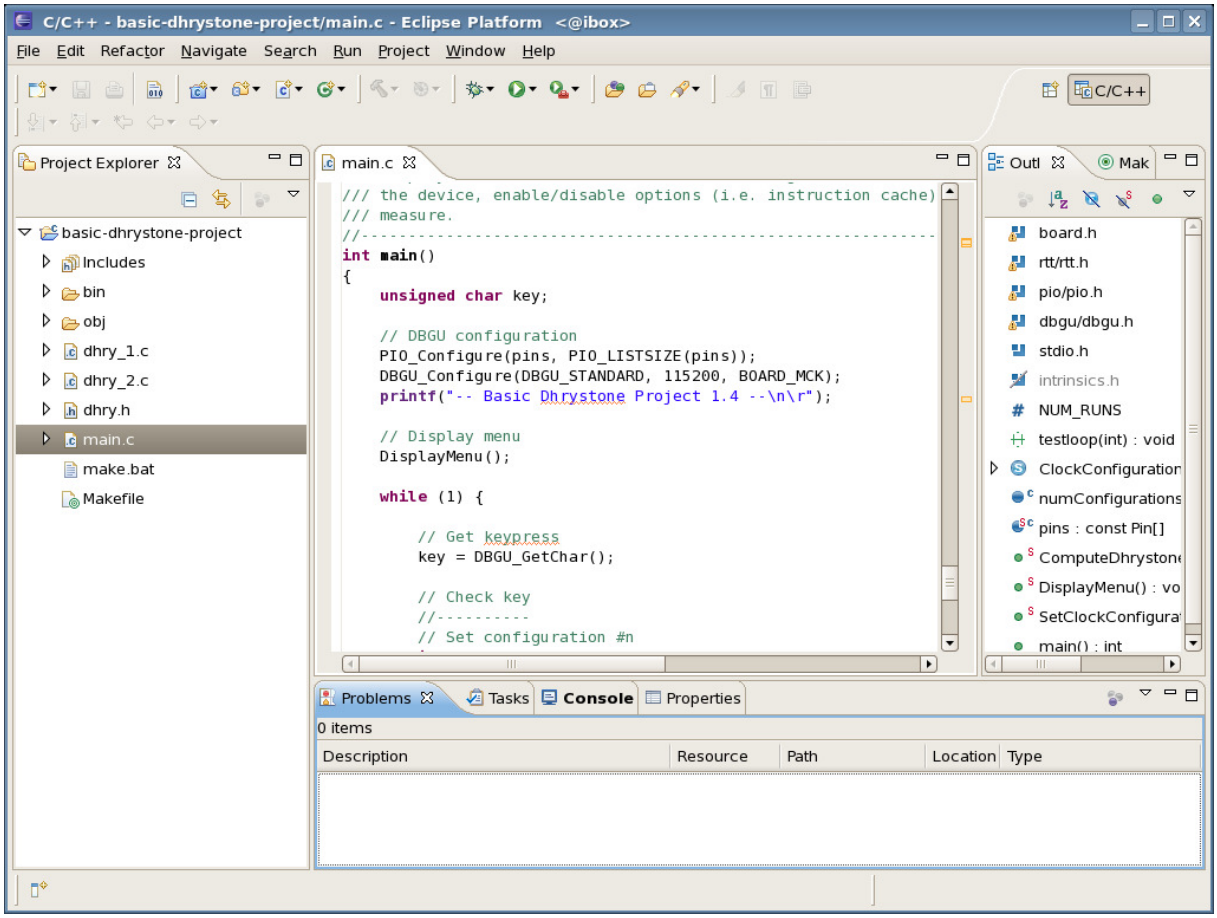
Click File->New->C Project. In the New Project dialog select “Makefile Project”, type “Project name: basic-dhrystone-project” and click Next:



Now click “Finish”:



Now the Eclipse window should look like (if it is necessary close the “Welcome screen”):

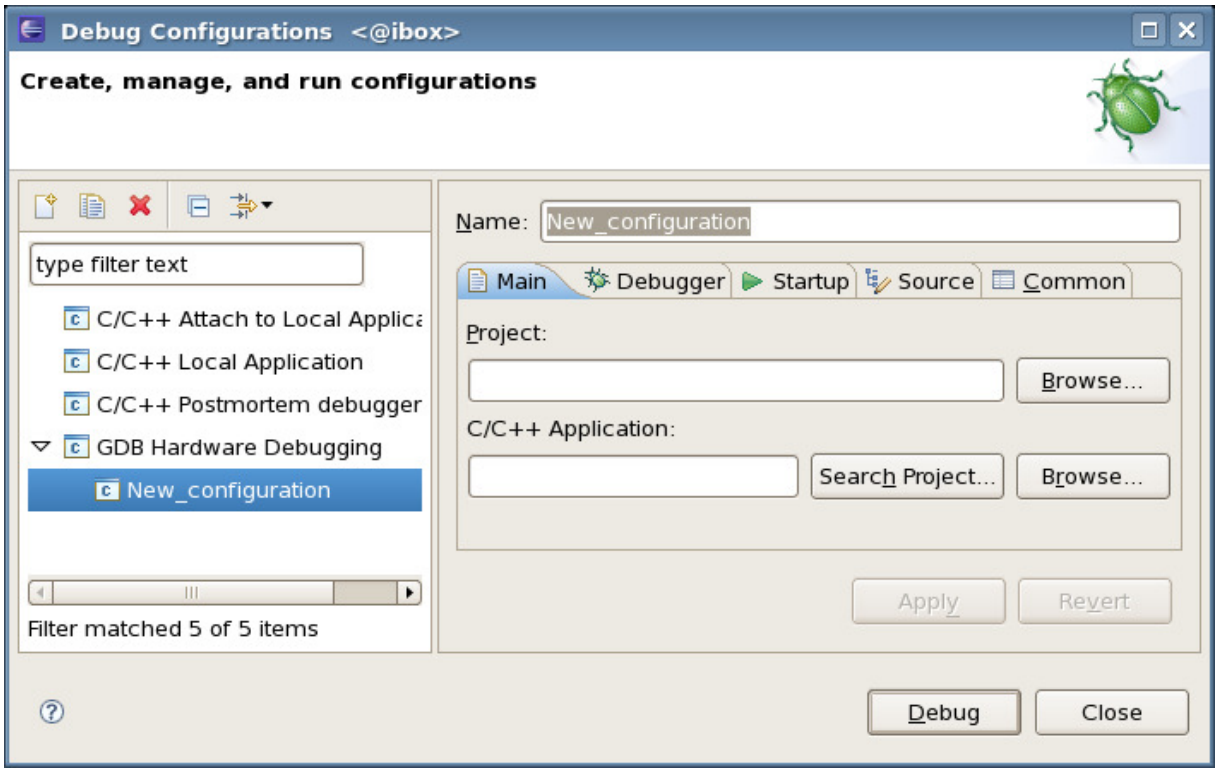


At this point your project should be compiled and liked to a single executable:

bin/basic-dhrystone-project-pm9263-at91sam9263-sram.elf

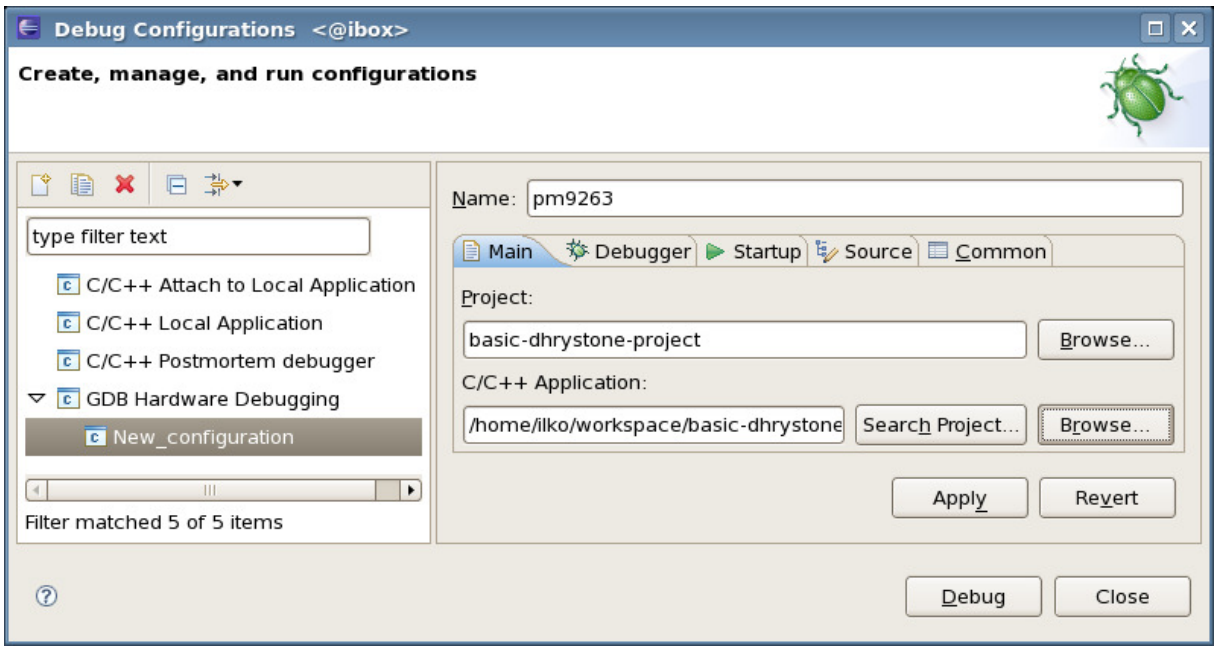
4.3 Configuring and working with the Eclipse built-in debugger

Click the “Run -> Debug configurations -> GDB Hardware Debugging”:



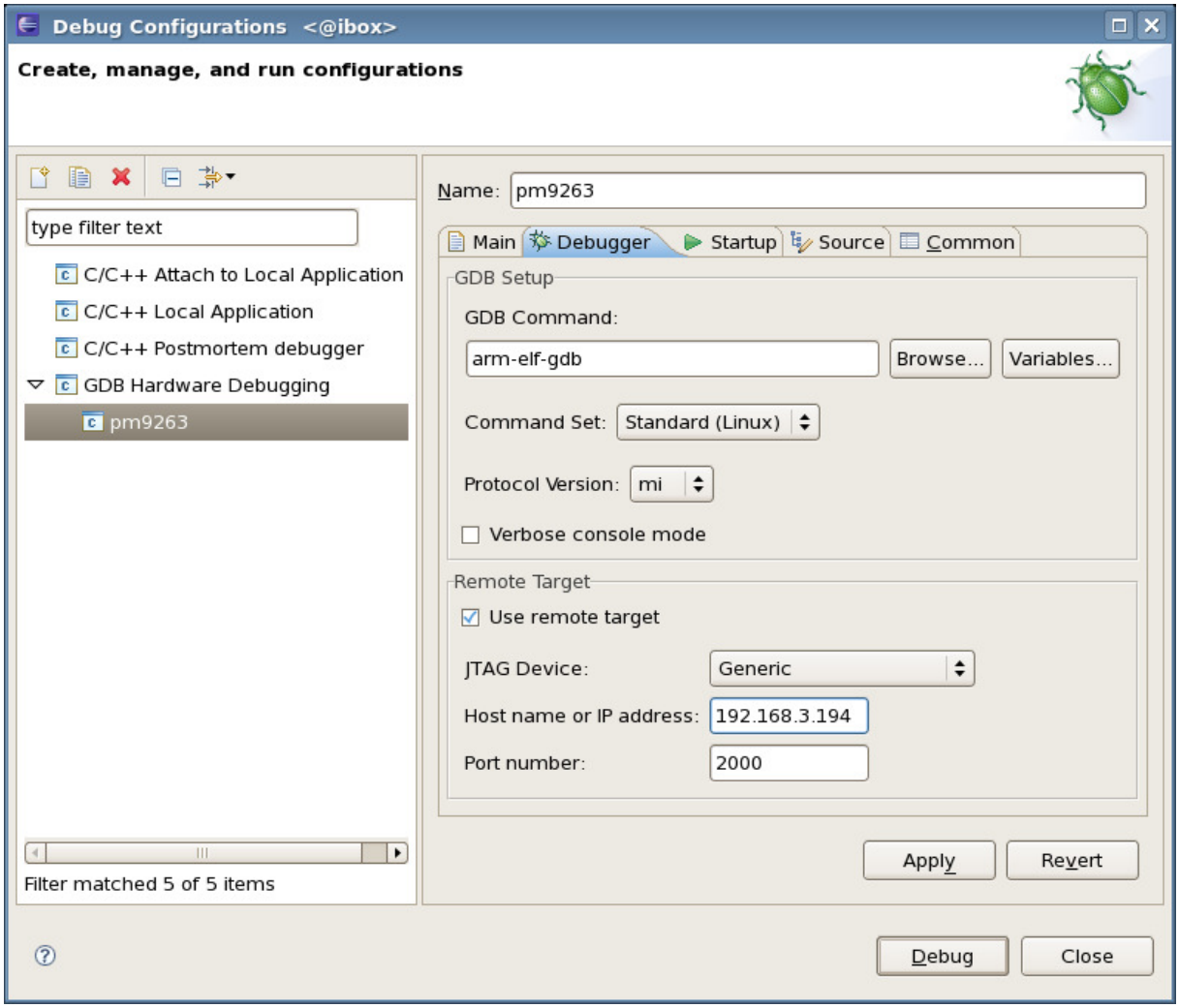
Type in the “Name” field: pm9263, in the “Project:” basic-dhrystone-project.

For the “C/C++ Application” browse and select: bin/basic-dhrystone-project-pm9263-at91sam9263-sram.elf

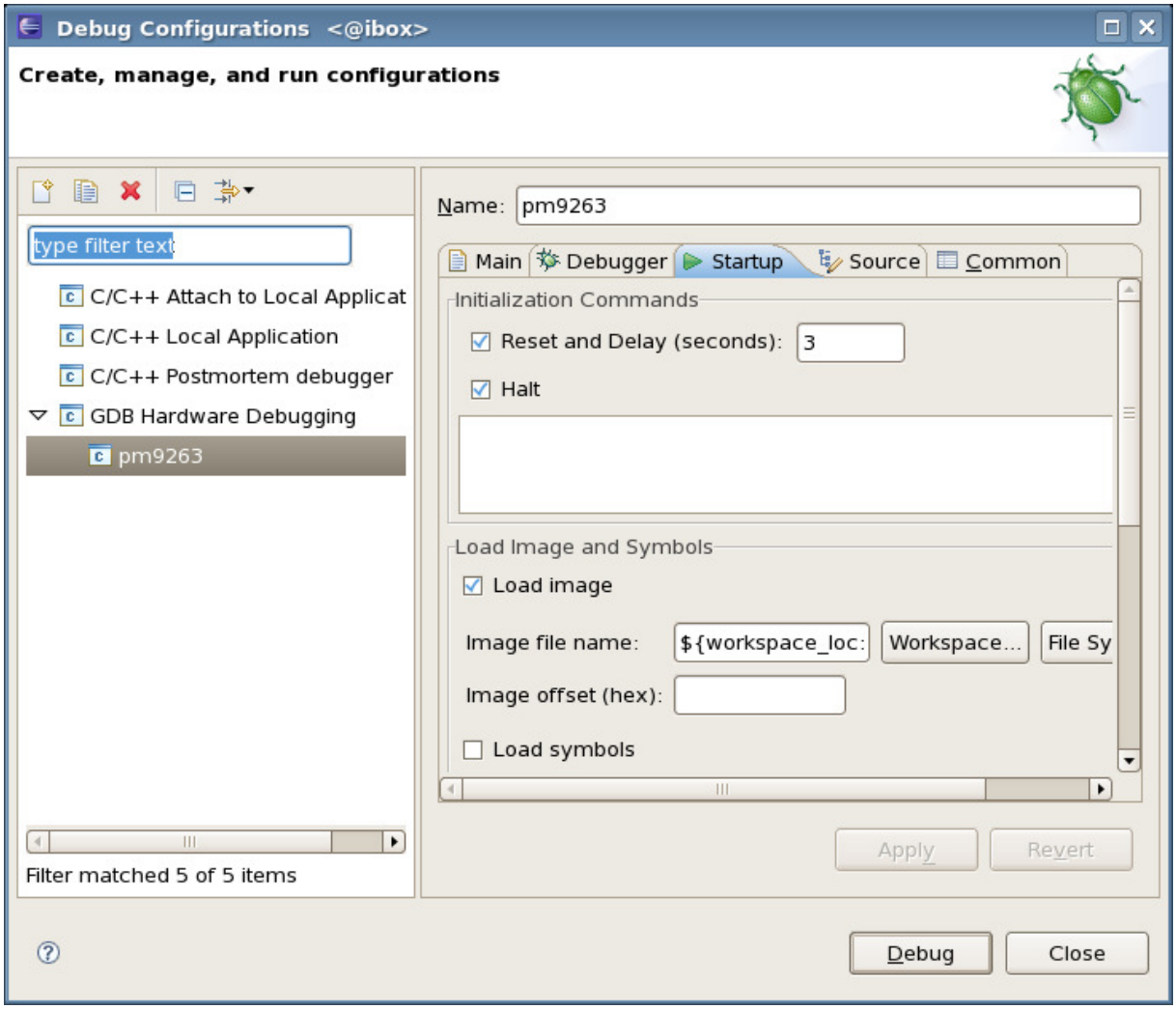


Now select the “Debugger” tab and type “GDB Command”: arm-elf-gdb

You should also enter the IP address of PEEDI (for example 192.168.3.194) and the port number (2000):



Now select the “Startup” tab and activate the “Load Image”. Click on the “Workspace” and browse until select: bin/basic-dhrystone-project-pm9263-at91sam9263-sram.elf:



Now click on the “Debug” button and the Eclipse debugger will start. You should also change to debug perspective. To start the application just type in the console window: “c”.