ColdFire cross development with GNU Toolchain and Eclipse

Version 1.0

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:

ColdFire is trademark of Freescale Ltd.
Windows, Win32, Windows CE are trademarks of Microsoft Corporation.
Ethernet is a trademark of XEROX.
All other trademarks are trademarks of their respective companies.

© 2005-2008 RONETIX
All rights reserved.

Change log

April 2007	- 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 Freescale Evaluation board MCF5282lite and PEEDI JTAG/BDM Emulator and Flash Programmer.

The necessary software components for a ColdFire cross development are:

- **GNU toolchain (compiler, linker, gdb)**
- **Eclipse IDE including Zylin CDT Plug-in for C/C++ development**
- **Java Runtime**

To enable a quick start in the ColdFire 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/coldfire/ronetix-m68k-elf.tar.bz2

- The Eclipse IDE: http://download.ronetix.info/eclipse/eclipse.tar.bz2
- A simple example: http://download.ronetix.info/examples/mcf5282lite_example.tar.gz

The necessary file for a Windows host is:
http://download.ronetix.info/toolchains/coldfire/ronetix-toolset-m68k.exe
This is a Windows installer which installs the GNU toolchain, Eclipse IDE + Zylin CDT and Java.

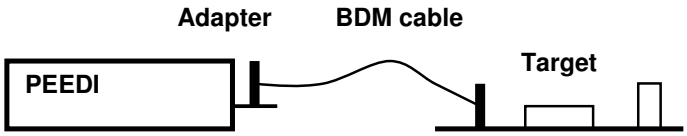
2 PEEDI ColdFire Emulator Installation

PEEDI (Powerful Embedded Ethernet Debug Interface) is a solution that enables you to debug software running on ColdFire processor cores via the BDM port.

In order to debug you need to configure PEEDI ColdFire 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 the supplied BDM adapter. The BDM adapter must be on the PEEDI side of the BDM 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/coldfire/ronetix-m68k-elf.tar.bz2

or get it from the CD.

Uncompress the archive:

```
Cd /
tar xvfj ronetix-linux-m68k-elf.tar.bz2
```

The toolchain will be installed in the /usr/cross/m68k-elf 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/m68k-elf/bin: in the .bashrc file add the following:

```
export PATH=$PATH:/usr/cross/m68k-elf/bin
```

2. Test the toolchain installation:

```
[linbox]$ m68k-elf-gcc -v
Using built-in specs.
Target: m68k-elf
Configured with: /home/src/cross/gcc-4.1.1/configure --target=m68k-elf
--build i686-pc-linux-gnu --host i686-pc-linux-gnu --disable-nls --
prefix=/usr/cross/m68k-elf --enable-interwork --enable-multilib --
enable-languages=c,c++ --with-newlib --enable-win32-registry=ronetix-
m68k --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 Eclipse IDE

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

http://download.ronetix.info/eclipse/eclipse.tar.bz2

The file eclipse.tar.bz2 includes:

- Eclipse SDK v3.2.1
- Embedded CDT v3.1 patched by Zylin
- Zylin plugin
- Java Virtual Machine v1.5.0_09

```
cd /usr/local
tar xvfj eclipse.tar.bz2
```

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

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

4. Installing an example

Download the MCF5282 example from here:

http://download.ronetix.info/examples/mcf5282lite_example.tar.gz

or get it from the CD.

```
Cd
mkdir work
cd work
tar xvfz mcf5282lite_example.tar.gz
```

At this point you should be able to build, debug and run applications on embedded ColdFire targets. You can compile and debug the example manual, from the shell prompt or using Eclipse. The working with Eclipse is explained in “Section 5: Working with Eclipse” from this manual.

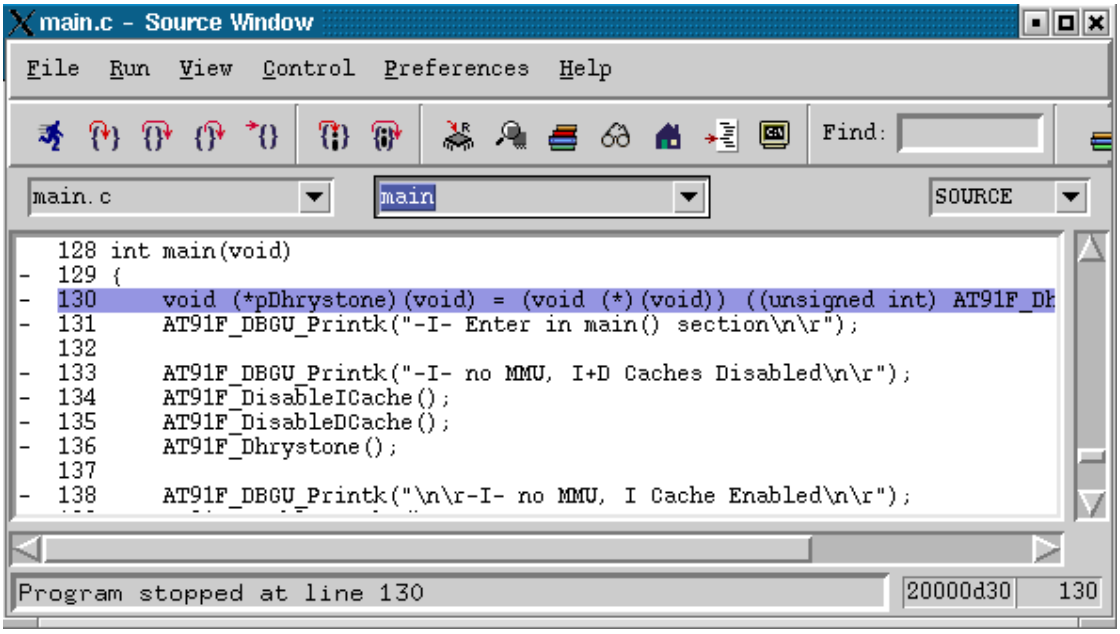
5. Compiling from the shell

```
[linbox]$ cd hello
[linbox]$ make
m68k-elf-gcc -x assembler-with-cpp -c -m528x -g -Wa,--register-prefix-
optional,-amhls=src/cf-crt0.lst   src/cf-crt0.S -o src/cf-crt0.o
m68k-elf-gcc -x assembler-with-cpp -c -m528x -g -Wa,--register-prefix-
optional,-amhls=src/cf-isv.lst   src/cf-isv.S -o src/cf-isv.o
m68k-elf-gcc -c -m528x -O0 -g -fomit-frame-pointer -fno-builtin -
ffreestanding -Wall -Wstrict-prototypes -fverbose-asm -Wa,-
ahlms=src/cf-crt1.lst   -MD -MP -MF .dep/cf-crt1.o.d -I . -I./. -I./src
-I./inc src/cf-crt1.c -o src/cf-crt1.o
m68k-elf-gcc -c -m528x -O0 -g -fomit-frame-pointer -fno-builtin -
ffreestanding -Wall -Wstrict-prototypes -fverbose-asm -Wa,-
ahlms=src/cf-isrs.lst   -MD -MP -MF .dep/cf-isrs.o.d -I . -I./. -I./src
-I./inc src/cf-isrs.c -o src/cf-isrs.o
m68k-elf-gcc -c -m528x -O0 -g -fomit-frame-pointer -fno-builtin -
ffreestanding -Wall -Wstrict-prototypes -fverbose-asm -Wa,-
ahlms=src/hello.lst   -MD -MP -MF .dep/hello.o.d -I . -I./. -I./src -
I./inc src/hello.c -o src/hello.o
m68k-elf-gcc ./src/cf-crt0.o ./src/cf-isv.o ./src/cf-crt1.o ./src/cf-
isrs.o ./src/hello.o -m528x -T"m5282evb-ram.ld" -fno-builtin -
ffreestanding -nostdinc -Wl,-Map=hello.map,--cref,--no-warn-mismatch
-o hello.elf
m68k-elf-objcopy -O binary hello.elf hello.bin
```

6. Debugging

From a X-console start the Insight debugger:

```
[linbox] m68k-elf-insight hello.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.

4 Toolset installation on Windows

Please download the GNU toolchain installer from our web site, launch it and follow the instructions.
http://download.ronetix.info/toolchains/coldfire/ronetix-toolset-m68k.exe

The setup will install:

By default, the installer provided by Ronetix installs everything necessary for developing. This includes:

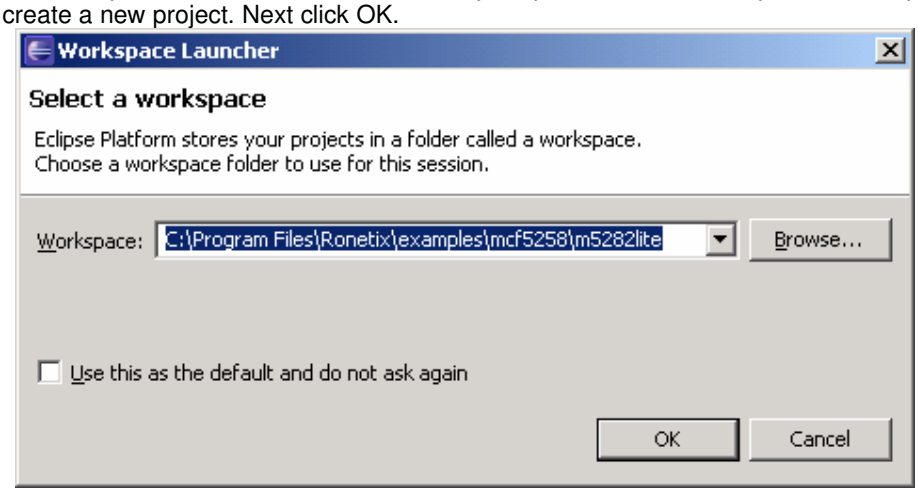
- m68k-elf GNU tools (compiler, linker, and debugger)
- Eclipse IDE v3.2.1 and Zylin CDT plug-in
- Java run-time environment for the Eclipse
- ready-to-build examples
- a free TFTP server

5 Working with Eclipse

5.1 Adding a project

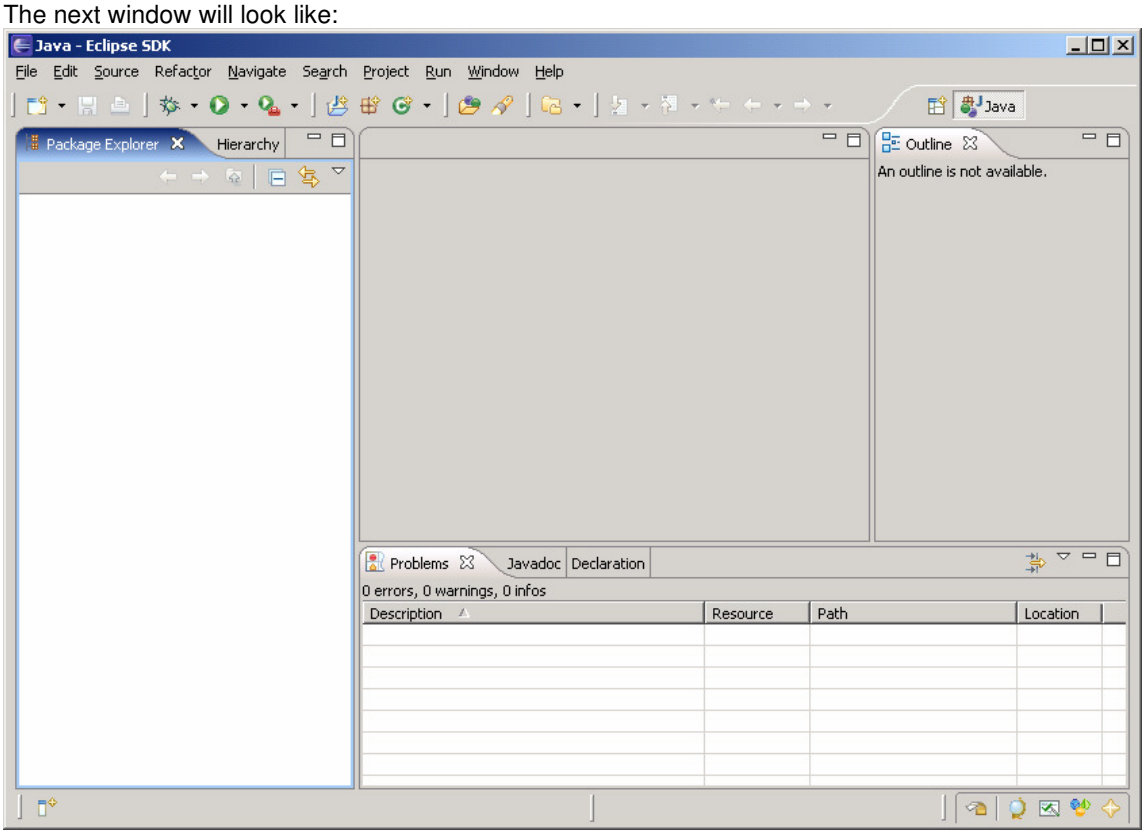
Eclipse (http://www.eclipse.org) is an open-source, Java based, powerful Integrated Development Environment (IDE). Adding the CDT plug-in (C/C++ Development Toolkit), you can edit and build C programs using the GNU compiler toolkit. A detailed C/C++ Development User Guide for Eclipse can be downloaded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

The Java run-time environment, Eclipse and the CDT plug-in are installed by the installer from Ronetix. Now start Eclipse IDE using the desktop icon, the “Workspace Launcher” dialog should appear, where you need to point the workspace folder. In our case point “C:\Program Files\Ronetix\examples\mcf5258\m5282lite”, this way eclipse will automatically include the project files when we create a new project. Next click OK.

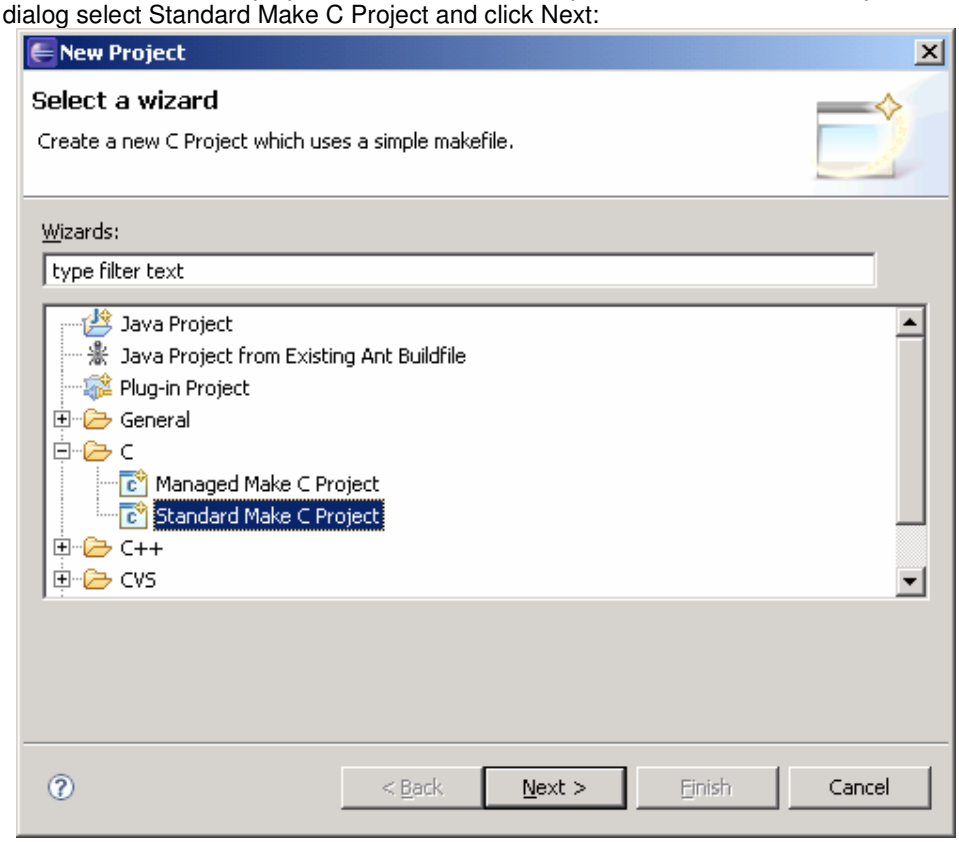


Now click at the "Workbench" icon at the right edge of the window.

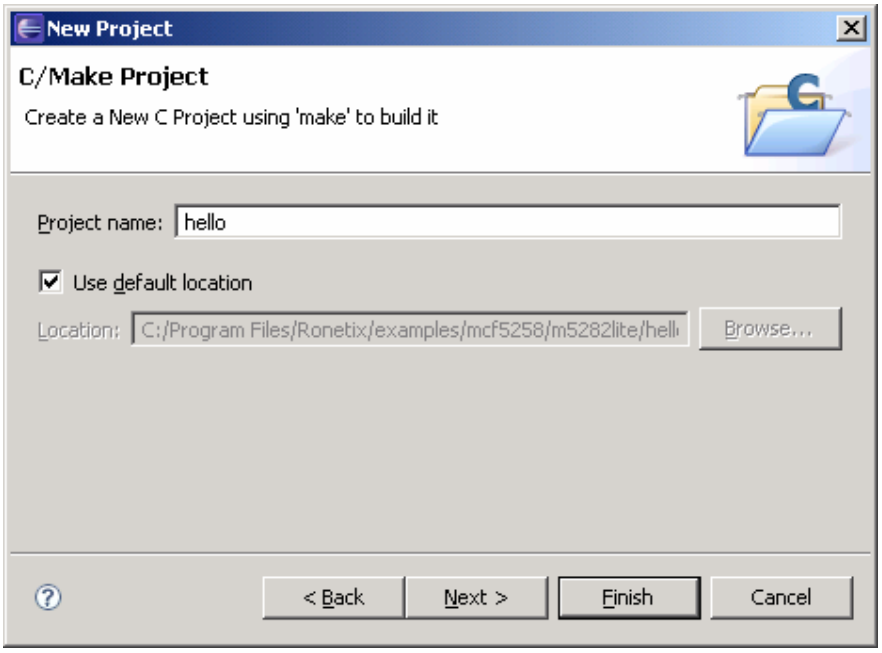
The next window will look like:



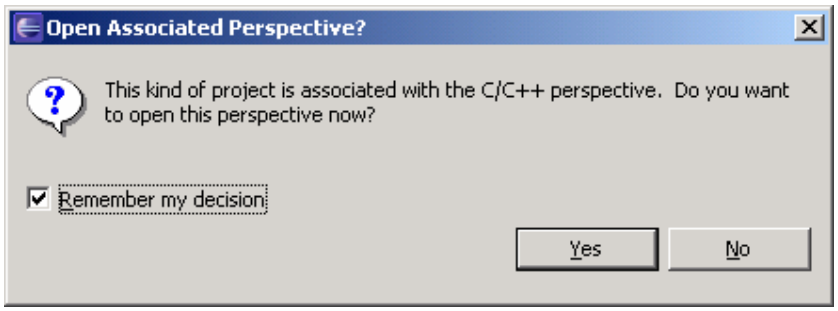
Now you need to add as a project the MCF5282 example. Click File->New->Project. In the New Project dialog select Standard Make C Project and click Next:



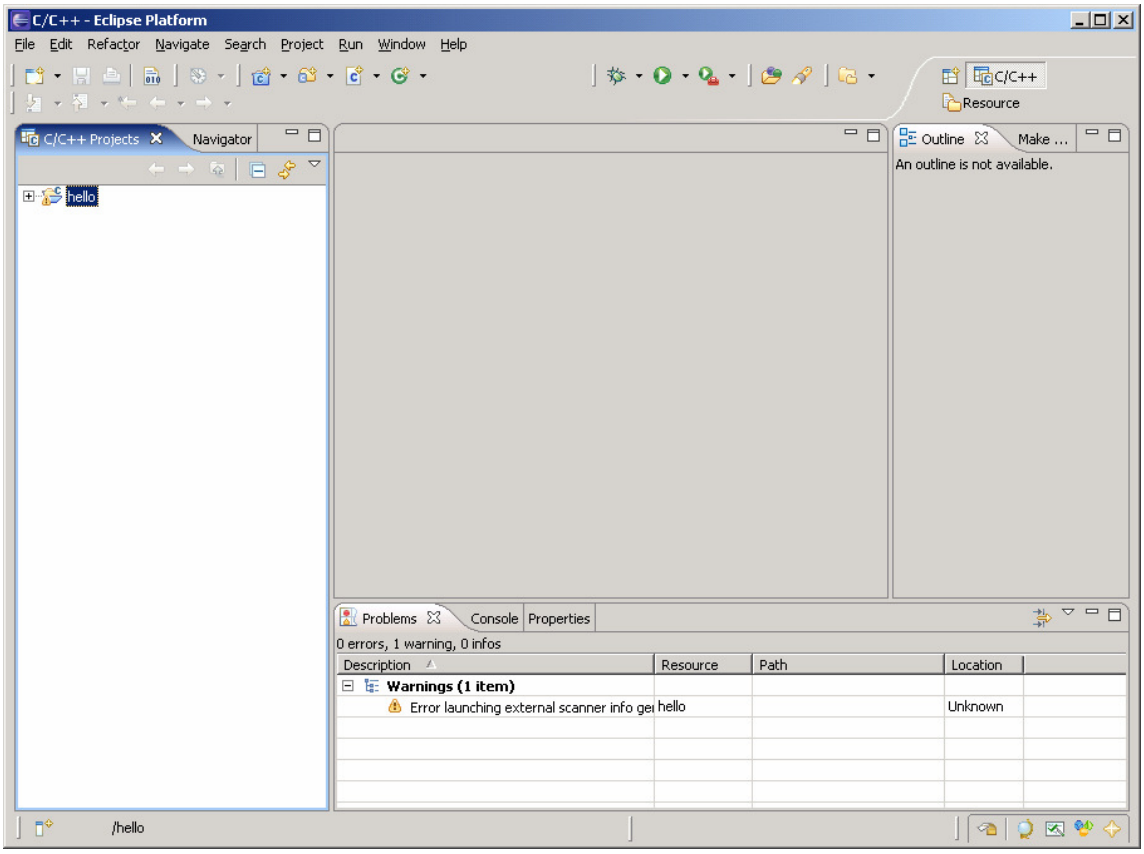
In the “New Project” dialog put “hello” for project name and click Finish. The hello already exists there so all of its files will be automatically imported in the new created Eclipse project.



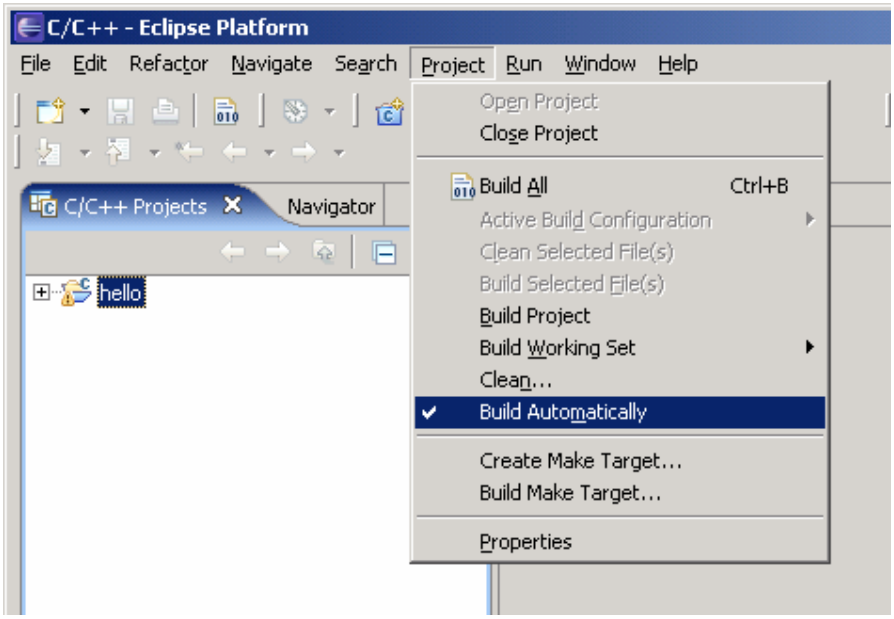
Now Eclipse will ask you to change the perspective to C/C++, check remember my decision and click YES:



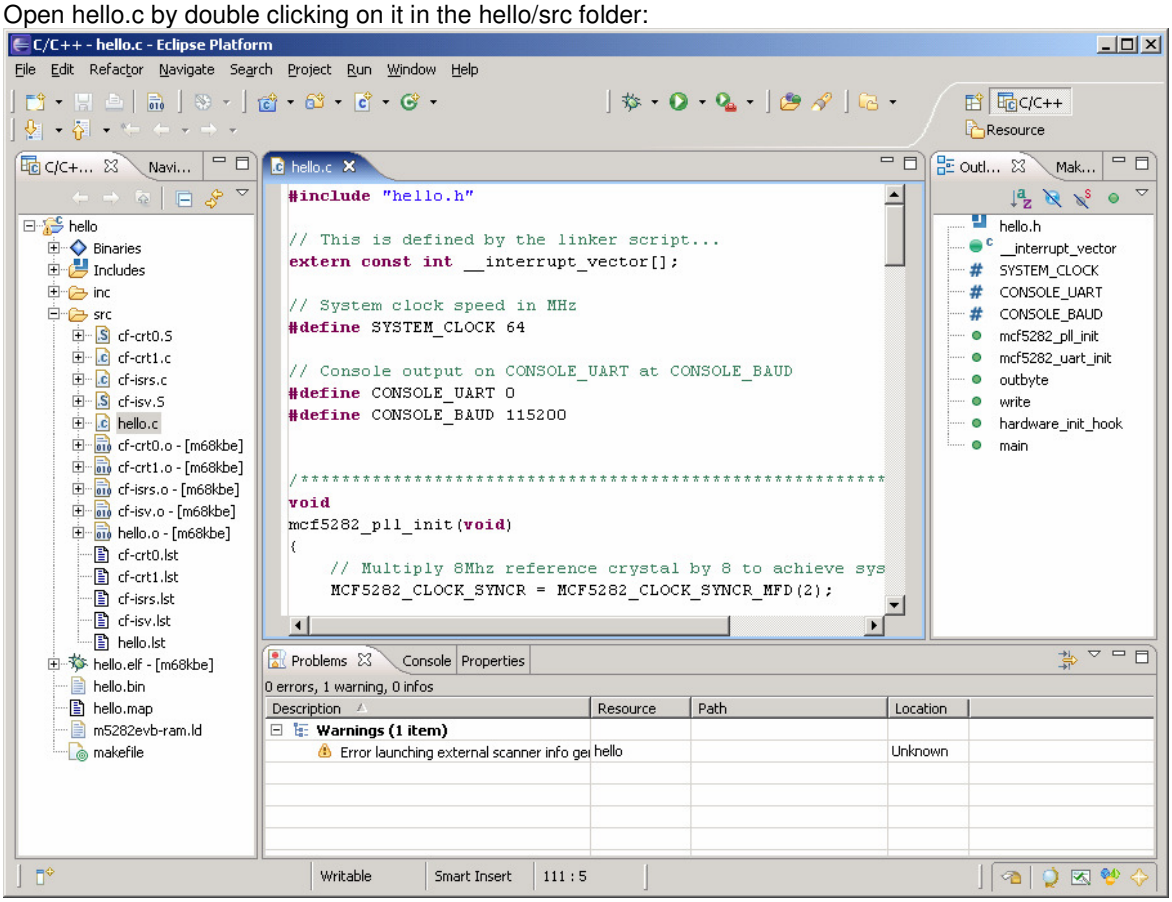
Now the Eclipse window should look like:



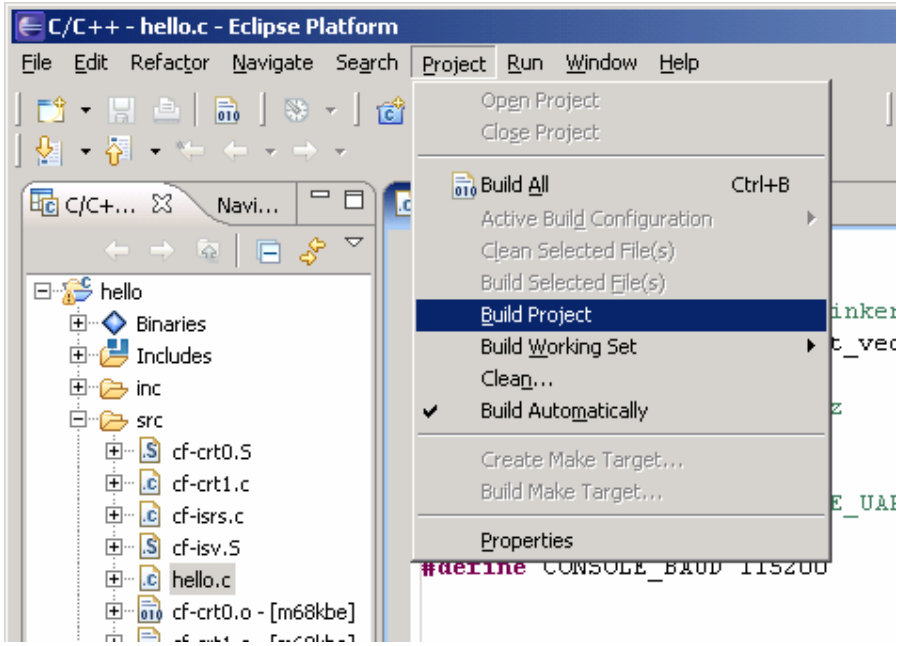
Use the "Project" menu and remove the checkmark at "Build Automatically":



Open hello.c by double clicking on it in the hello/src folder:



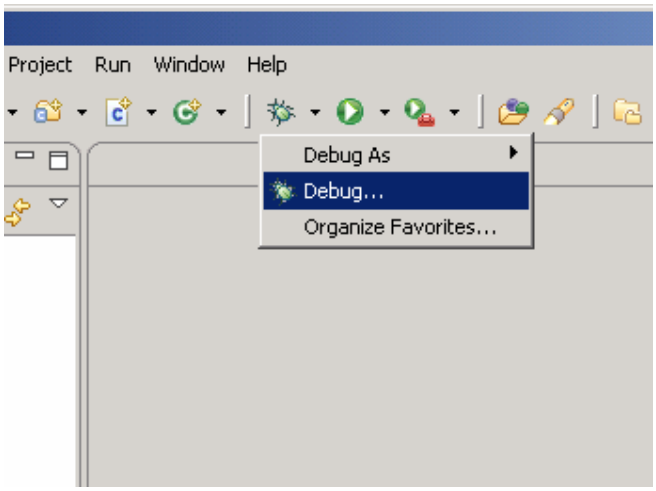
Now build the project:



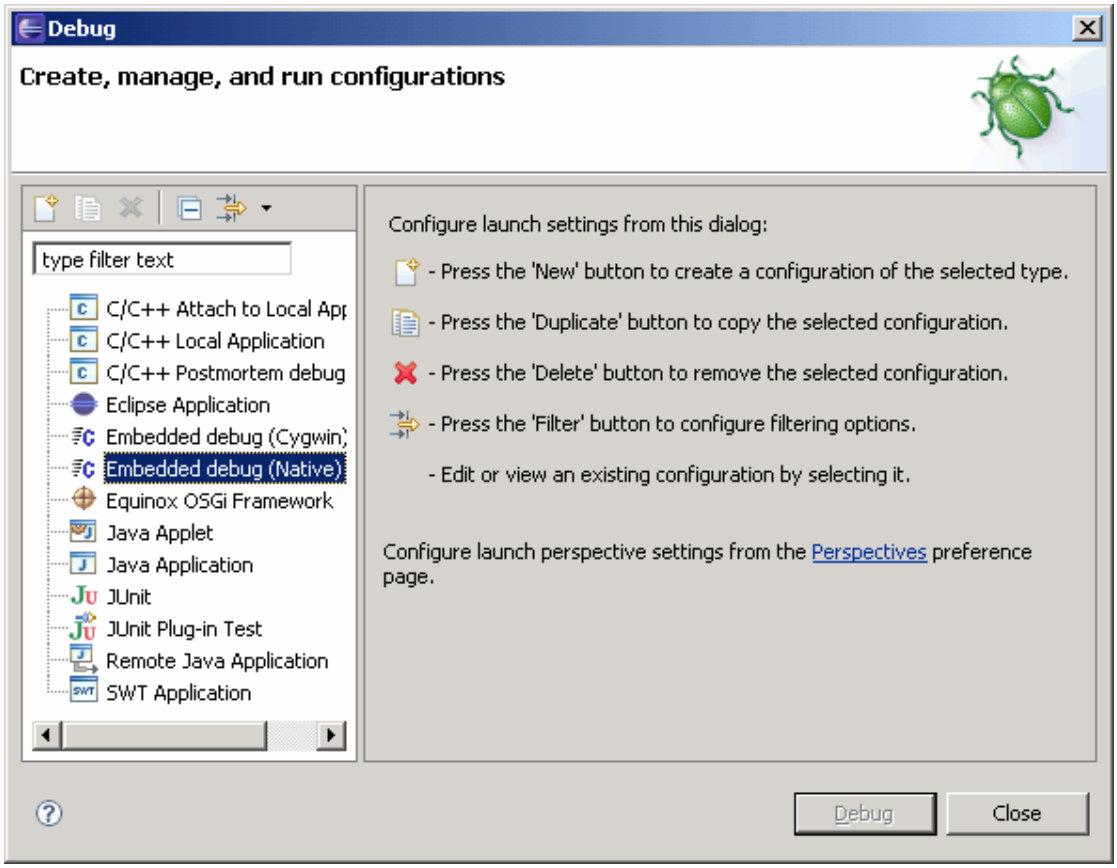
At this point your project should be compiled and liked to a single executable “hello.elf” file. You can check its presence.

5.2 Configuring and working with the Eclipse built-in debugger

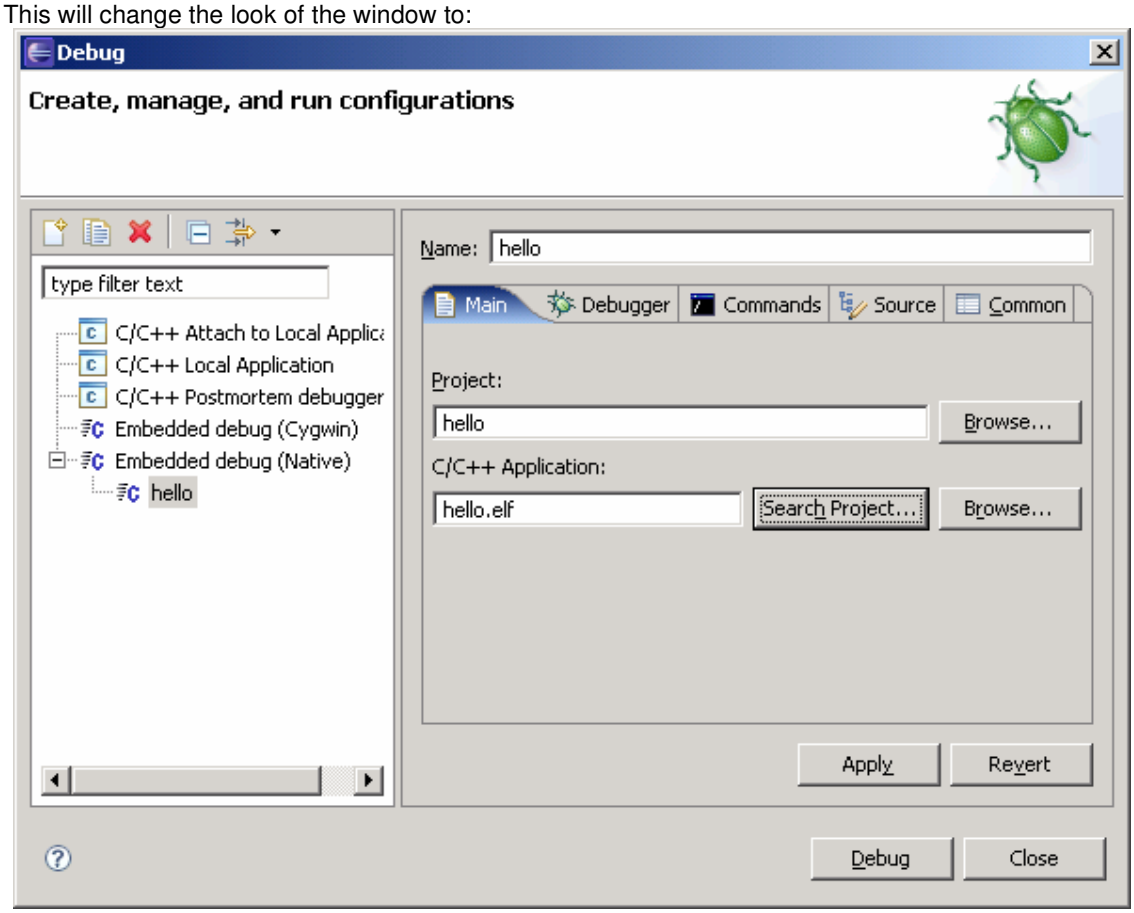
First click the arrow right to the “bug” button and next “Debug…”



This opens the Debug configuration dialog. Select Embedded debug (Native) and click the New button:

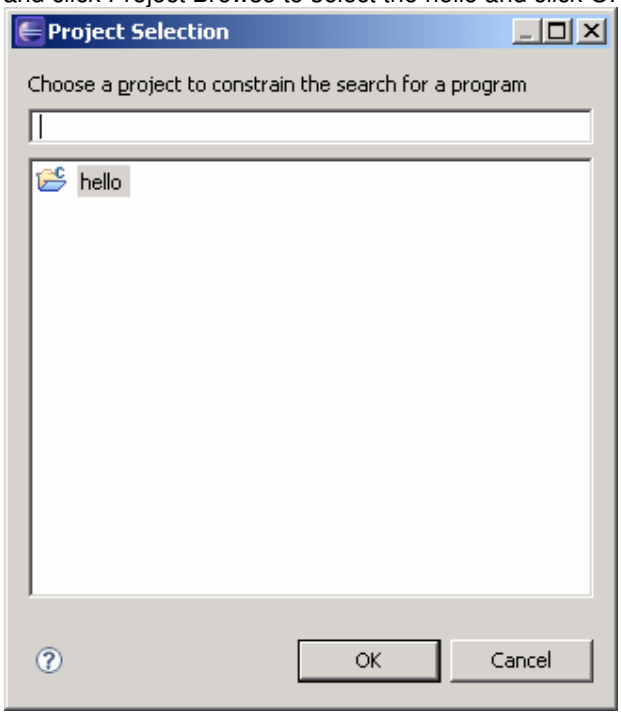


This will change the look of the window to:

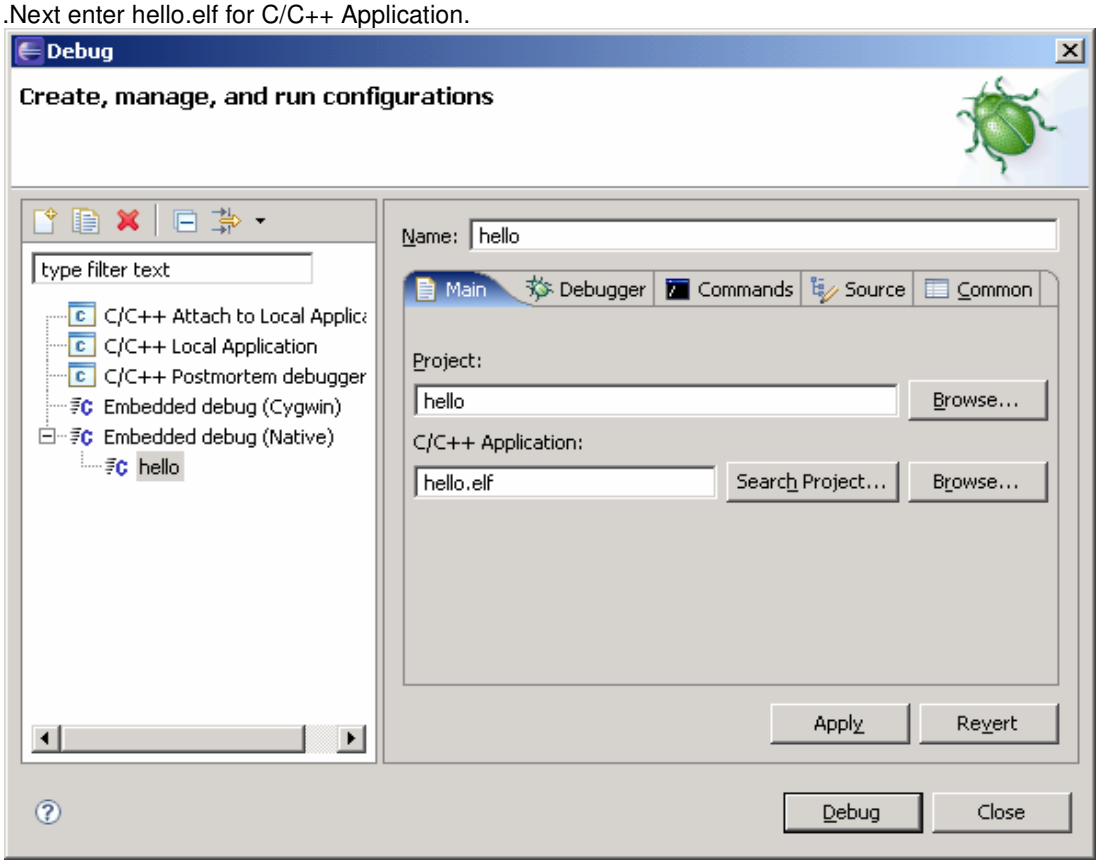


If your window does not look like the previous, follow the next two steps:

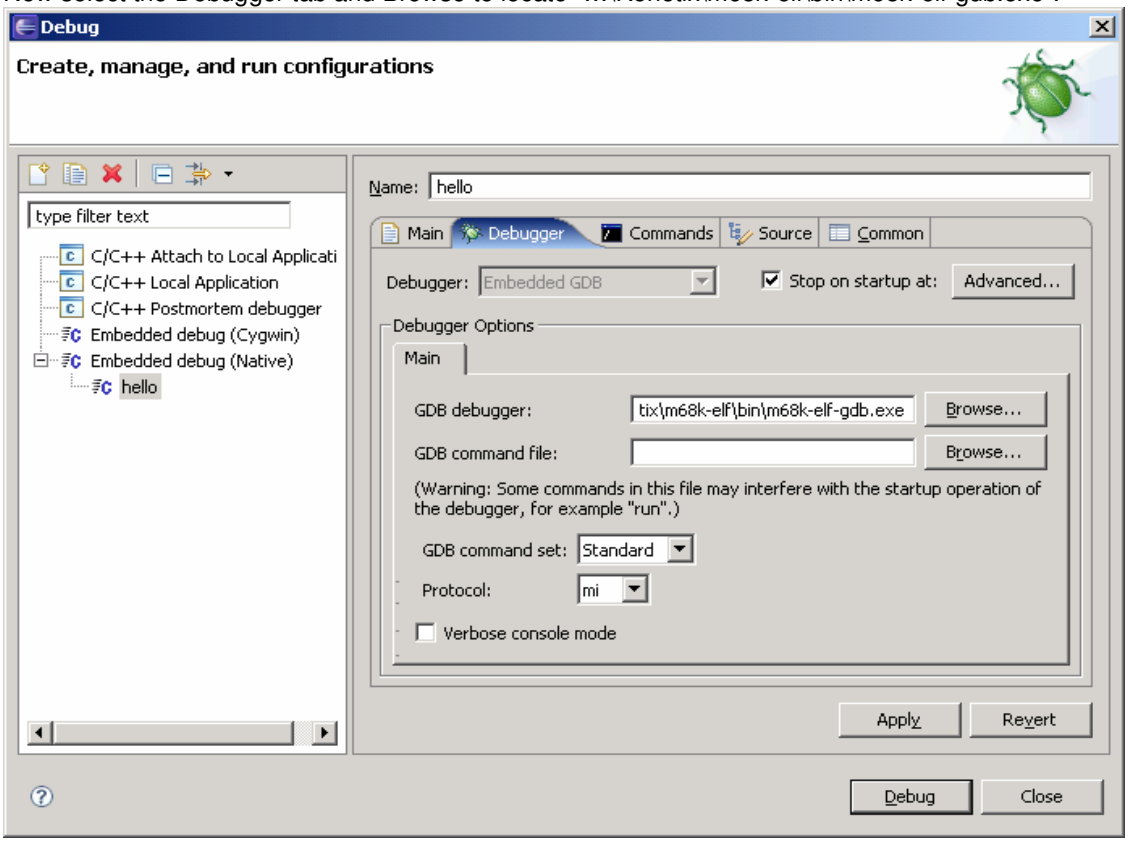
1. Enter hello for name and click Project Browse to select the hello and click OK:



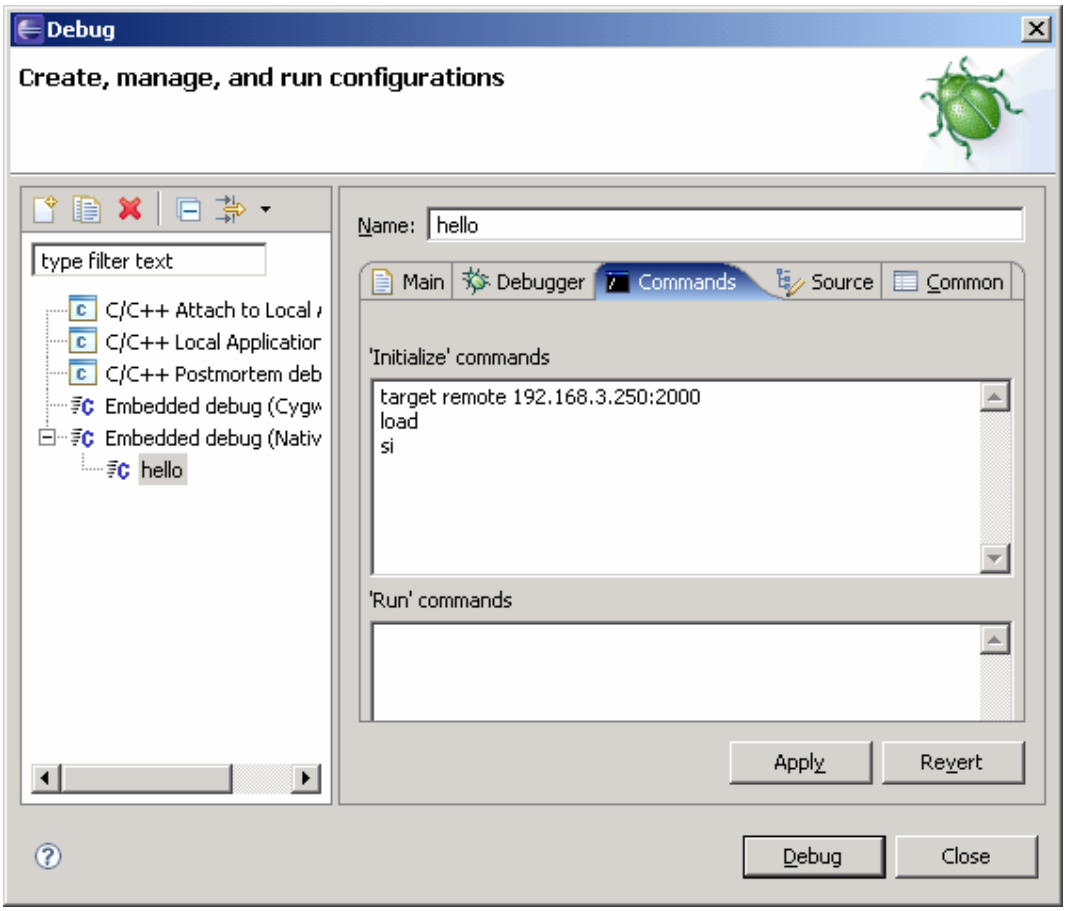
2 .Next enter hello.elf for C/C++ Application.



Now select the Debugger tab and Browse to locate "…\Ronetix\m68k-elf\bin\m68k-elf-gdb.exe":

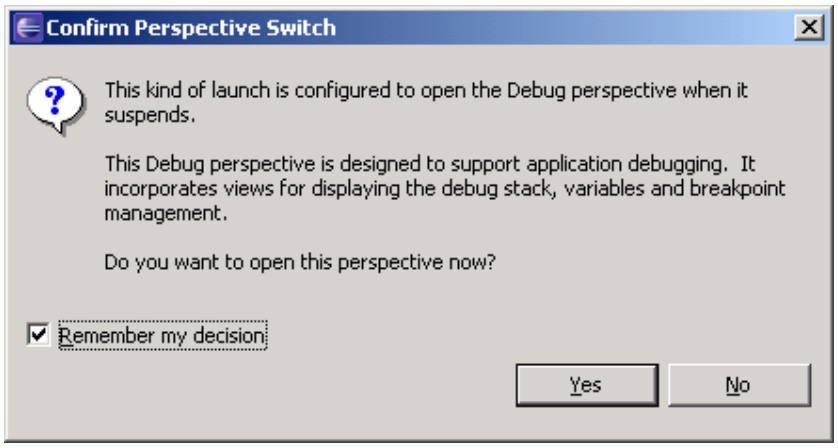


Now change to the Commands tab and enter these commands considering your PEEDI IP address:



These command will tell gdb:
- to connect to your PEEDI
- to load the project executable file to the target

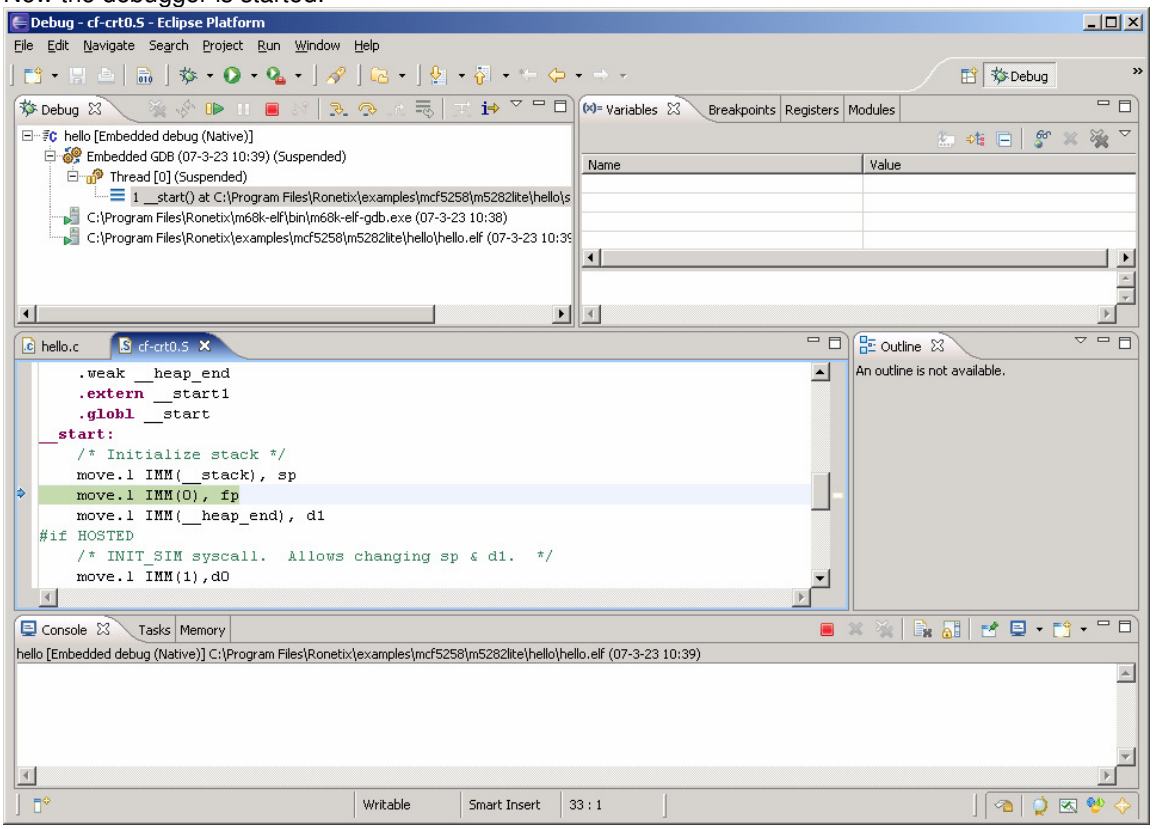
Next click Apply and Debug, this will start the debugger and Eclipse should ask you:



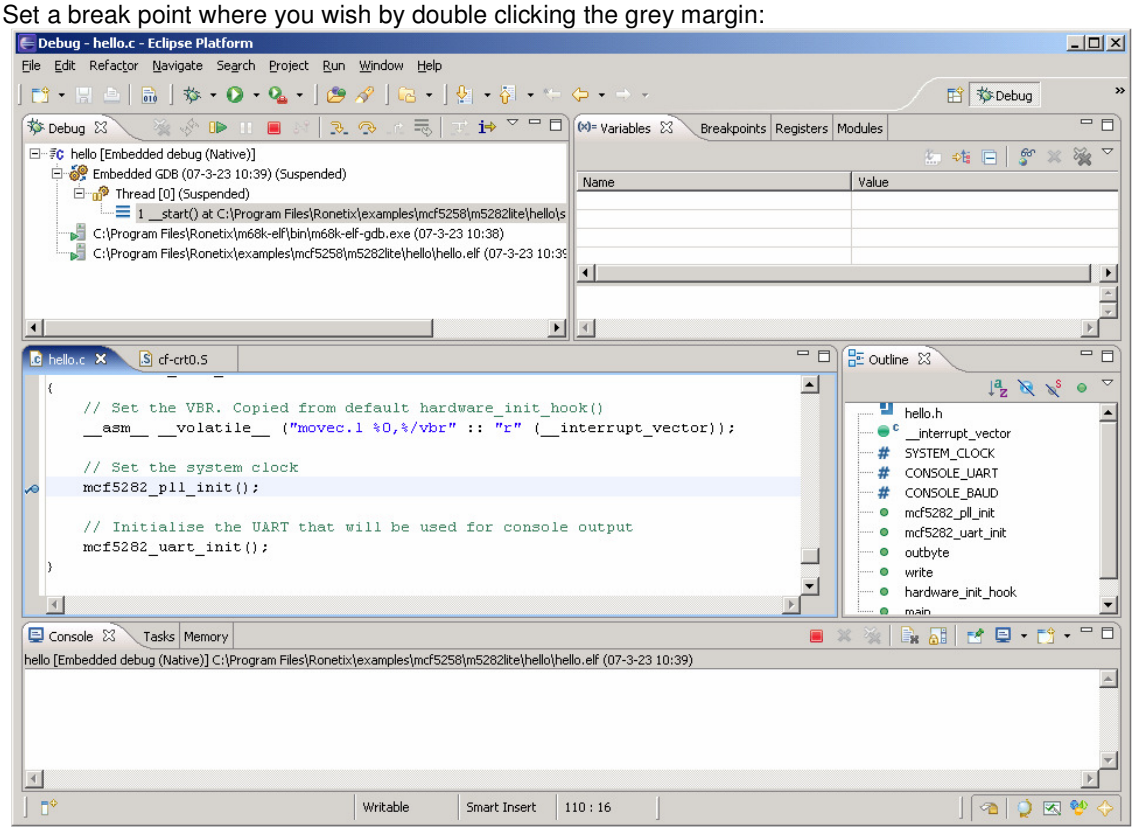
Check remember my decision and click YES.

If Eclipse doesn't automatically switch to the Debug perspective, you can switch manually using menu Window -> Open Perspective -> Debug

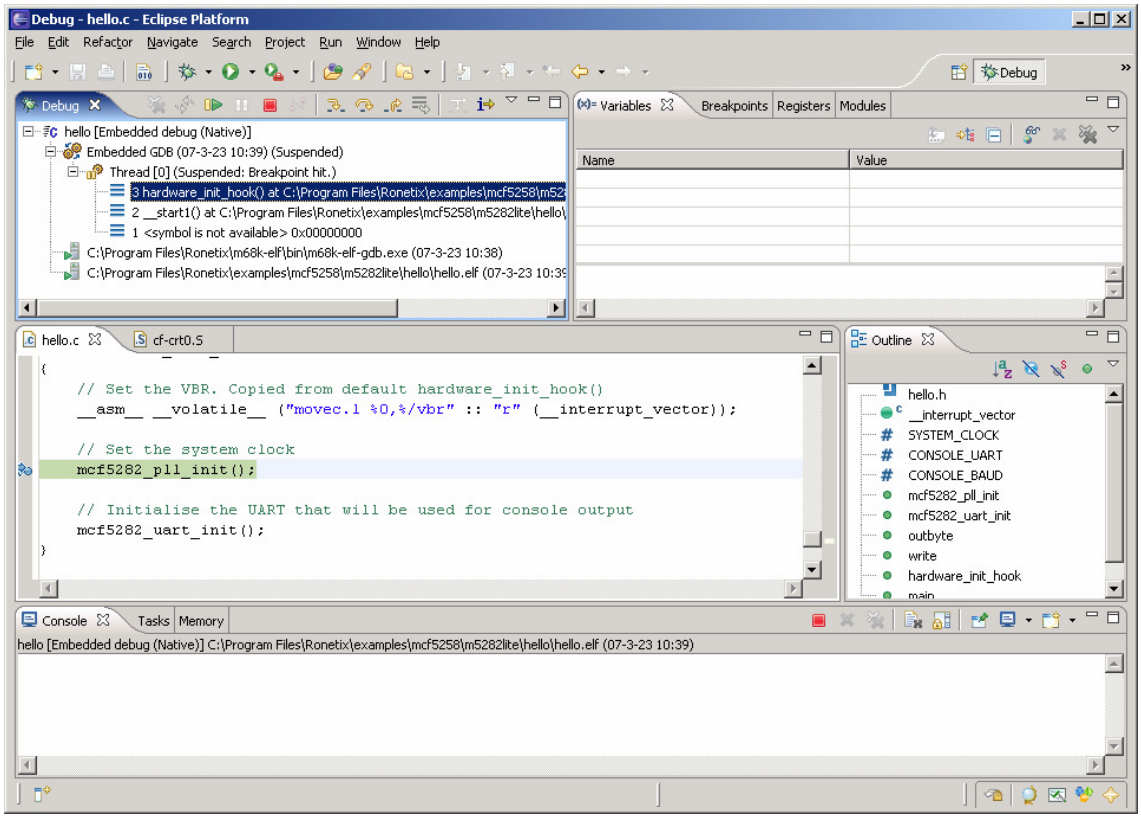
Now the debugger is started:



Set a break point where you wish by double clicking the grey margin:



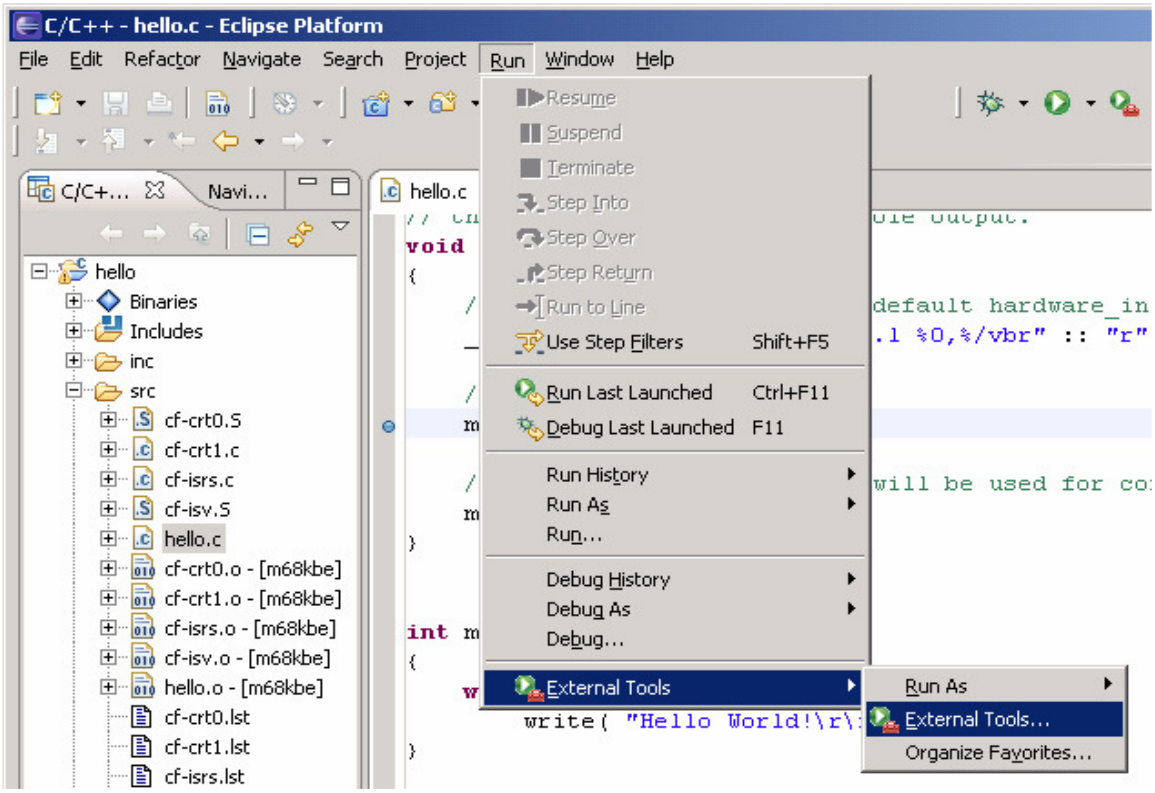
Next click the Resume botton. The application is started and shortly it should break were the break point is:



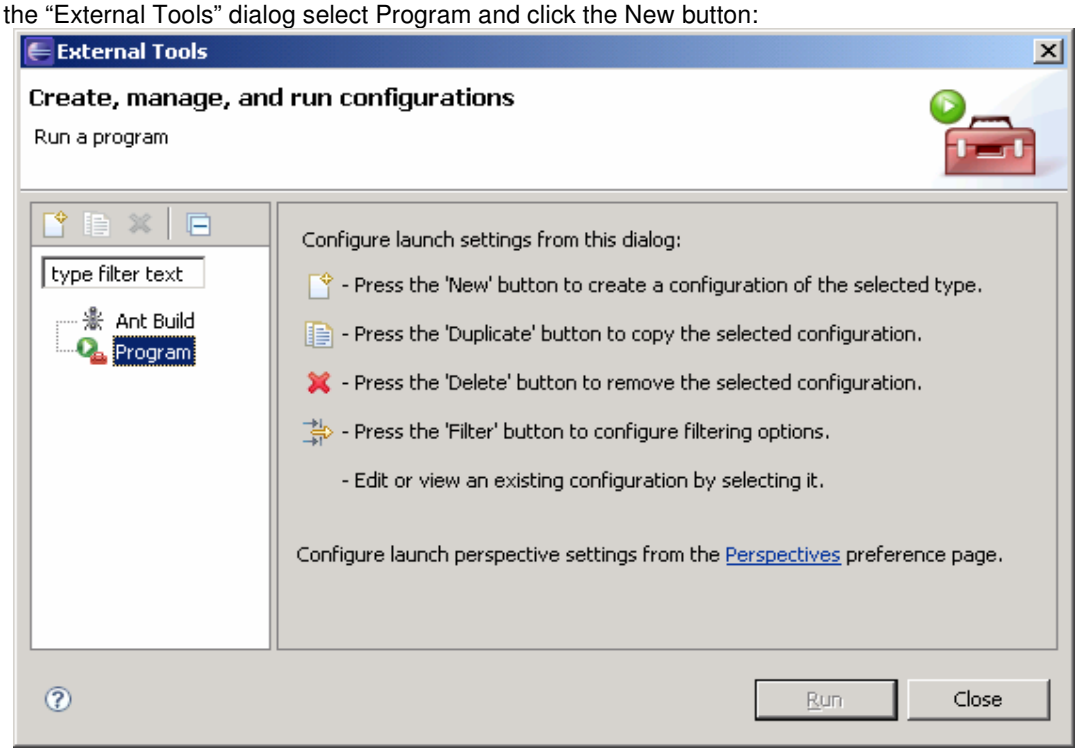
At this point you can continue debugging, i.e. step, put breaks, resume, etc.

5.3 Using Insight debugger as an Eclipse external tool

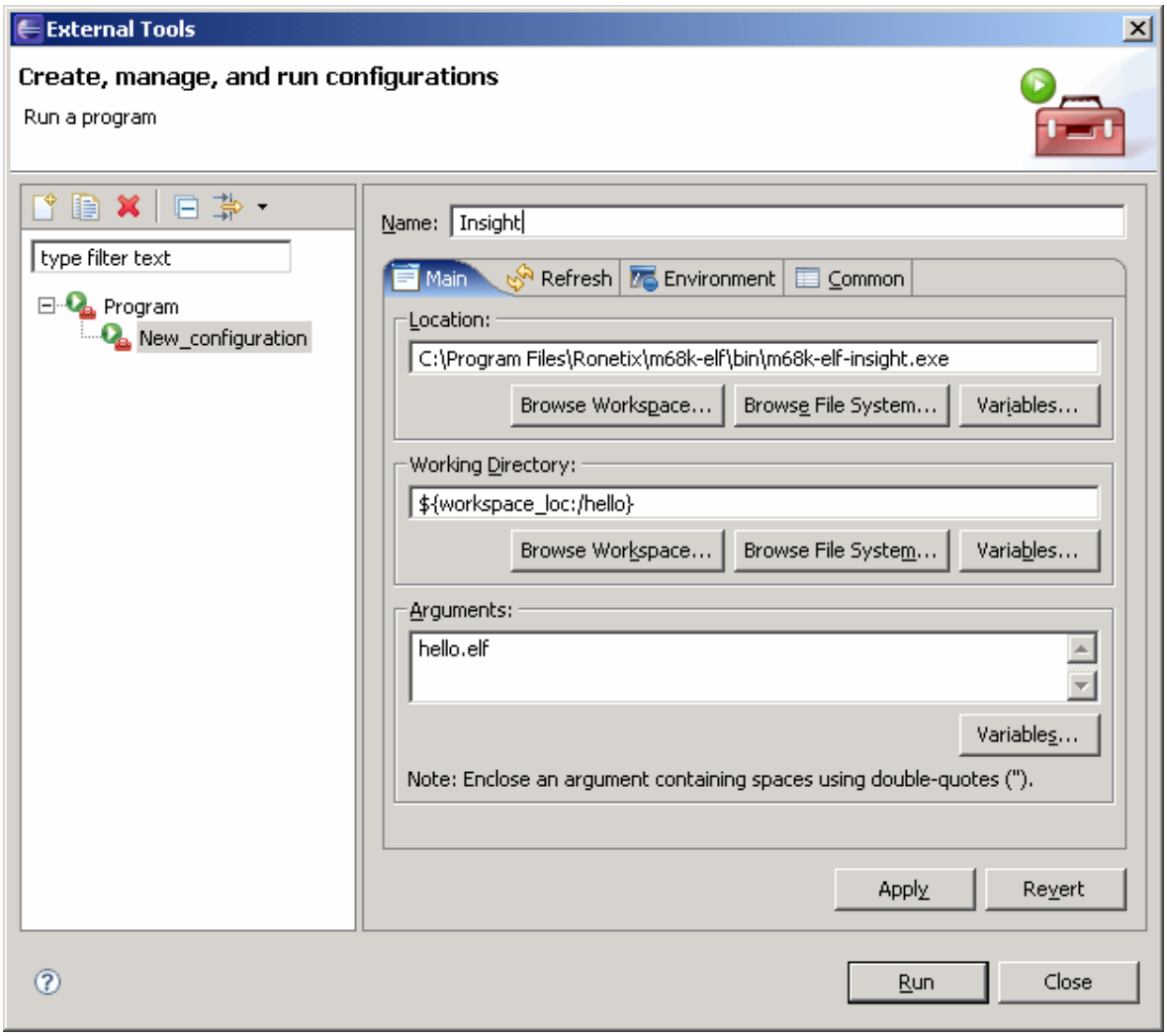
I you prefer you can use Insight for a debugger when developing with Eclipse IDE.
For this reason you need to add it as an external tool by clicking Run->External Tools->External Tool…



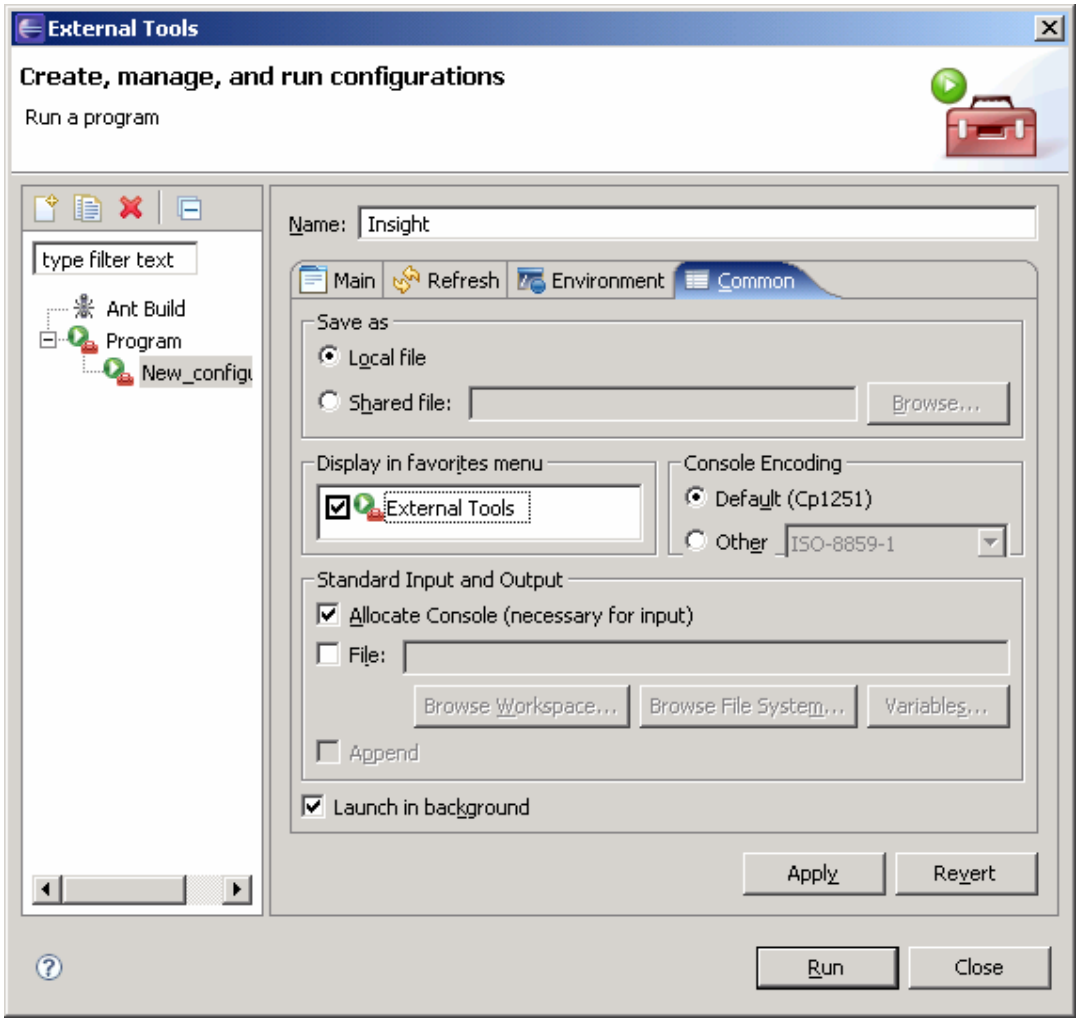
In the “External Tools” dialog select Program and click the New button:



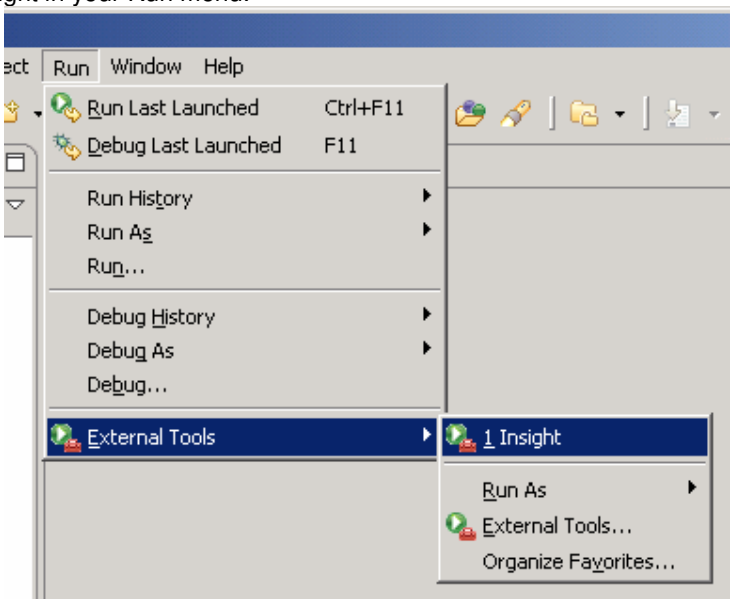
In the new dialog for name type “Insight”, for location click Browse File System and navigate to “…\Ronetix\m68k-elf\bin\m68k-elf-insight.exe”, for Working Directory click Browse Workspace and select hello and for Arguments type hello.elf.



Now click on Common, and in the “Display in favorites menu” select Insight to add it to the Run menu and click Apply and Close:



Now you have Insight in your Run menu.



You have just added the Insight.

You can follow the previous procedure to add any external tool you wish to use while developing. Now you can start Insight and debug the project.