OVP Debugging Applications with Eclipse User Guide

Imperas Software Limited
Imperas Buildings, North Weston,
Thame, Oxfordshire, OX9 2HA, UK
docs@imperas.com

Author:	Imperas Software Limited
Version:	1.3
Filename:	OVPsim_Debugging_Applications_with_Eclipse_User_Guide.doc
Project:	OVP Debugging Applications with Eclipse User Guide
Last Saved:	Monday, 13 January 2020
Keywords:	

Table of Contents

1 Preface

This document describes how to debug an application running on the OVP simulator using the Eclipse Helios 3.6.2 release Integrated Development Environment.

The example in this document demonstrates debugging of ARM and MIPS32 applications but the same approach is valid for applications running on processor models for other architectures supported by the OVP simulator.

1.1 Notation

`Code`	Code and command extracts

1.2 Related OVP Documents

- OVPsim Installation and Getting Started Guide
- OVPsim and CpuManager User Guide

2 Introduction

The *OVPsim and CpuManager User Guide* describes how platforms containing any number of processor models can be constructed. This document describes how to debug an application running on *one* processor in such a platform, while it is simulating using the OVPsim simulation environment. OVPsim supports single-processor debugging with Eclipse and the Gnu debugger (GDB) via the Remote Serial Protocol (RSP). Advanced multi-processor debug facilities are available in Imperas commercial products.

2.1 Prerequisites

Pre-built demos are provided, that will be described later, to allow the immediate debugging of an application using OVP and Eclipse Helios 3.6.2

To debug a simulated application with Eclipse you will need Eclipse with the C/C++ Development Tooling (CDT) plugins.

This walkthrough was produced with Eclipse Helios version 3.6.2.

It is recommended that Eclipse Helios version 3.6.2 has been installed using the Imperas installer, see section 3 The Imperas Eclipse Helios Distribution. Alternatively, Eclipse is available for free download from www.eclipse.org. The *Eclipse IDE for C/C++ developers* package should be used which includes the necessary plugins.

To build OVPsim platforms you require the OVP API files and libraries as well as a native compiler.

To build applications you need a cross-compiler and a version of the Gnu debugger for the chosen architecture.

The *OVPsim Installation and Getting Started Guide* provides a step-by-step guide to obtaining and installing the OVP files, a native compiler and cross compiler tools. It is strongly recommended that you follow that guide until you are able to build, compile and simulate a platform using OVPsim.

3 The Imperas Eclipse Helios Distribution

An installer *Imperas_Eclipse_Helios_3_6_2.<version>* is available from the OVP resources pages.

The Imperas Eclipse installation package provides an Eclipse installation that includes the required plugins for C/C++ application and OVPsim platform development and debugging.

The package installs, by default, into a directory Imperas_Eclipse.

An environment variable, IMPERAS_ECLIPSE_HOME, is used to point to the root of the installation. This is setup automatically on Windows and may be setup using the script, *setup.sh*, provided within the installation on Linux.

4 Imperas Eclipse Provided Debug Demos

A number of demonstrations are provided that include the required files to setup a defined project into a new Eclipse workspace.

The demos provided will include a script to start an Eclipse development and debug session. These scripts are identified with the starting prefix ***DEBUG_ECLIPSE_***

The script expects that the Imperas Eclipse package has been installed and is located by the environment variable IMPERAS_ECLIPSE_HOME. If it is not available the script will fail.

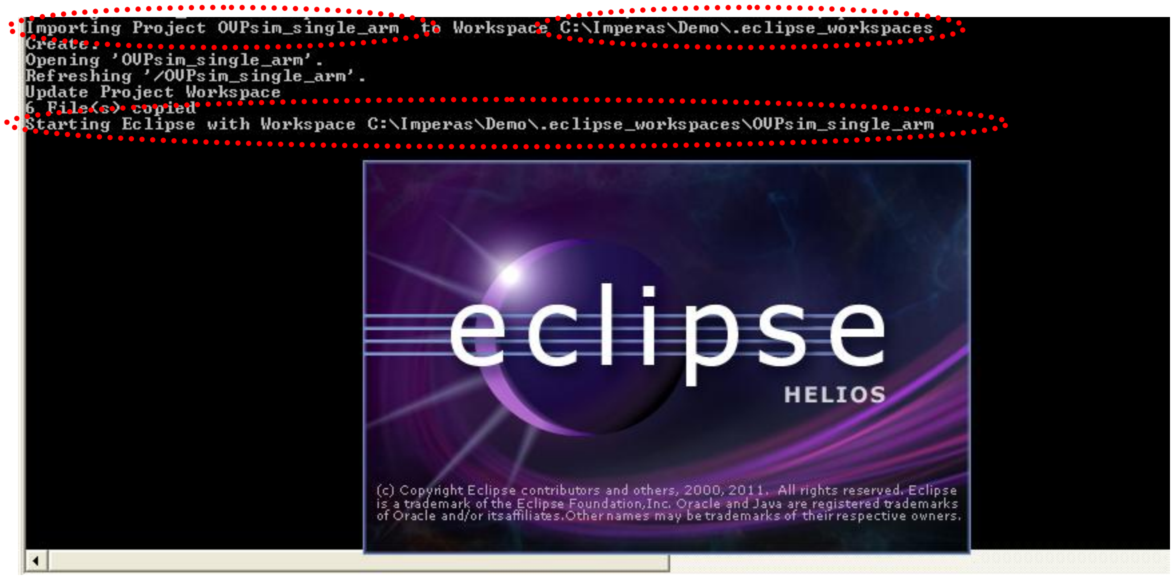
The GDB is expected to be found in one of the Cross Compiler libraries. If it is not available the script will fail.

4.1 Example Demonstration OVPsim_single_arm

4.1.1 Launching Eclipse

The first time we run the script DEBUG_ECLIPSE_dhrystone.bat we will create a new Eclipse workspace and in it a new project OVPsim_single_arm before invoking Eclipse.

You will see that a project is Imported into a default workspace location for this demonstration before Eclipse is invoked using this workspace.



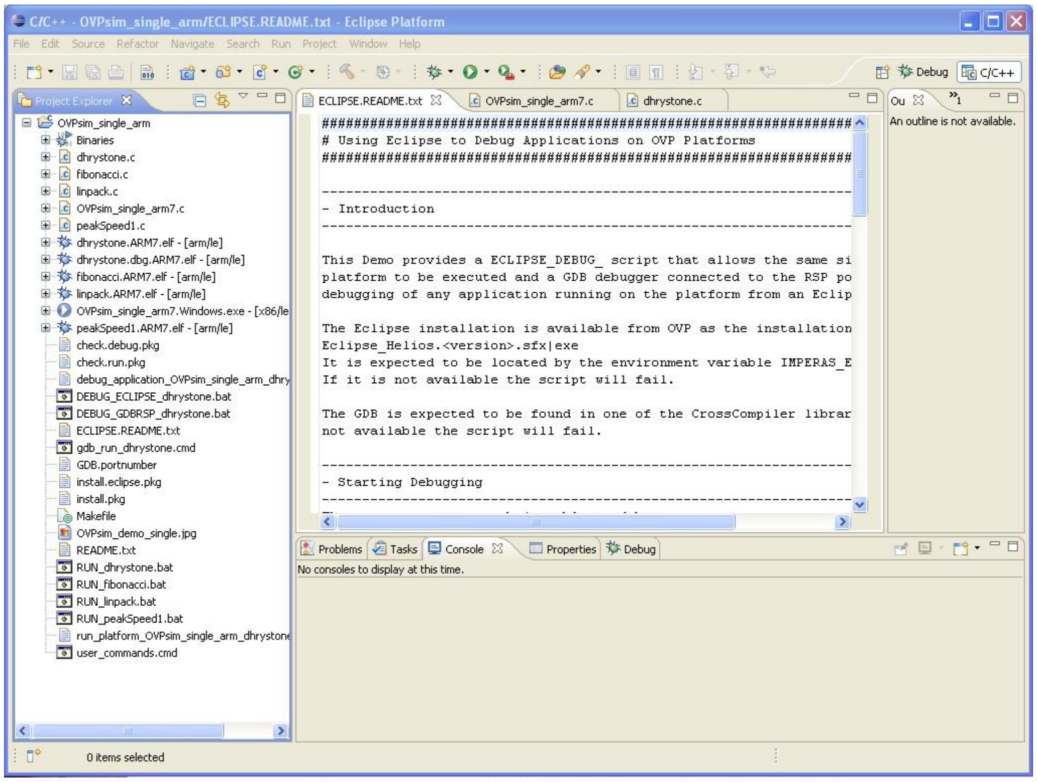
```
Importing Project OVPsim_single_arm  to Workspace C:\Imperas\Demo\.eclipse_workspaces
Create:
Opening 'OVPsim_single_arm'.
Refreshing '/OVPsim_single_arm'.
Update Project Workspace
6 File(s) copied
Starting Eclipse with Workspace C:\Imperas\Demo\.eclipse_workspaces\OVPsim_single_arm
```

eclipse
HELIOS
(c) Copyright Eclipse contributors and others, 2000, 2011. All rights reserved. Eclipse is a trademark of the Eclipse Foundation,Inc. Oracle and Java are registered trademarks of Oracle and/or its affiliates.Other names may be trademarks of their respective owners.

All subsequent runs of the script will only invoke Eclipse allowing the same workspace to remain.

When Eclipse has started a project is available with three files open

1. A README providing the same information provided in this section
2. The OVPsim platform definition
3. The Dhrystone benchmark application that will be used for this debugging demonstration.


C/C++ - OVPsim_single_arm/ECLIPSE.README.txt - Eclipse Platform
File Edit Source Refactor Navigate Search Run Project Window Help
Debug
C/C++
Project Explorer
OVPsim_single_arm
Binaries
dhrystone.c
fibonacci.c
linpack.c
OVPsim_single_arm7.c
peakSpeed1.c
dhrystone.ARM7.elf - [arm/le]
dhrystone.dbg.ARM7.elf - [arm/le]
fibonacci.ARM7.elf - [arm/le]
linpack.ARM7.elf - [arm/le]
OVPsim_single_arm7.Windows.exe - [x86/le
peakSpeed1.ARM7.elf - [arm/le]
check.debug.pkg
check.run.pkg
debug_application_OVPsim_single_arm_dhry
DEBUG_ECLIPSE_dhrystone.bat
DEBUG_GDBRSP_dhrystone.bat
ECLIPSE.README.txt
gdb_run_dhrystone.cmd
GDB.portnumber
install.eclipse.pkg
install.pkg
Makefile
OVPsim_demo_single.jpg
README.txt
RUN_dhrystone.bat
RUN_fibonacci.bat
RUN_linpack.bat
RUN_peakSpeed1.bat
run_platform_OVPsim_single_arm_dhrystone
user_commands.cmd
0 items selected
ECLIPSE.README.txt
OVPsim_single_arm7.c
dhrystone.c
Ou
An outline is not available.
Problems
Tasks
Console
Properties
Debug

```
####################################################################
# Using Eclipse to Debug Applications on OVP Platforms
####################################################################

--------------------------------------------------------------------
- Introduction
--------------------------------------------------------------------

This Demo provides a ECLIPSE_DEBUG_ script that allows the same si
platform to be executed and a GDB debugger connected to the RSP po
debugging of any application running on the platform from an Eclip

The Eclipse installation is available from OVP as the installation
Eclipse_Helios.<version>.sfx|exe
It is expected to be located by the environment variable IMPERAS_E
If it is not available the script will fail.

The GDB is expected to be found in one of the CrossCompiler librar
not available the script will fail.

--------------------------------------------------------------------
- Starting Debugging
--------------------------------------------------------------------
```

```
No consoles to display at this time.
```

4.1.2 Starting Debugging

There are two stages to being able to debug.
1) Starting the simulated platform.
2) Connecting the Eclipse Debugger

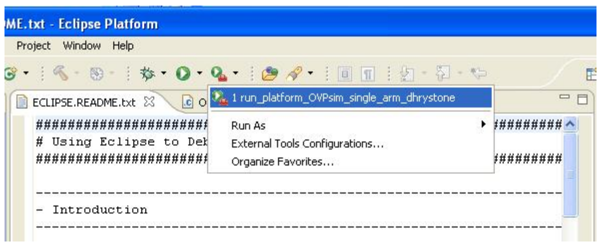
These are provided in the form of Eclipse 'launch's

4.1.2.1 Start the simulated platform

This is invoked as an 'External Tool' from the Menu "run->external tools" or the drop down menu button identified by the green run arrow with red toolbox icon

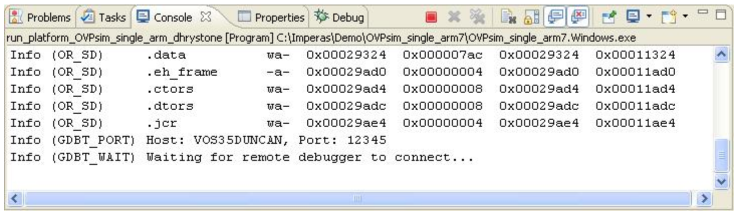
The 'launch' may already be visible in the quick launch section or will have to be selected from the "External Tools Configuration..." menu

The launch to invoke is "run_platform_OVPsim_single_arm_<application name>"


ME.txt - Eclipse Platform
Project Window Help
ECLIPSE.README.txt
1 run_platform_OVPsim_single_arm_dhrystone
Run As
External Tools Configurations...
Organize Favorites...

```
######################
# Using Eclipse to Deb
######################

------------------------------------------------------------------
- Introduction
------------------------------------------------------------------
```

The platform will now wait for a connection to be made to the specified port



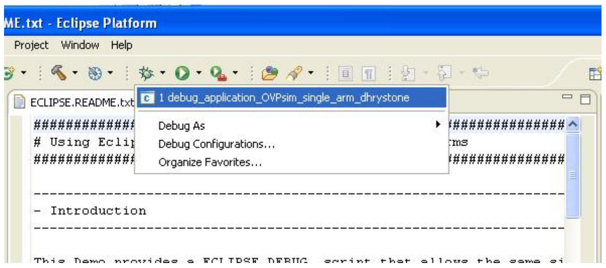
```
run_platform_OVPsim_single_arm_dhrystone [Program] C:\Imperas\Demo\OVPsim_single_arm7\OVPsim_single_arm7.Windows.exe
Info (OR_SD)      .data        wa-  0x00029324  0x000007ac  0x00029324  0x00011324
Info (OR_SD)      .eh_frame    -a-  0x00029ad0  0x00000004  0x00029ad0  0x00011ad0
Info (OR_SD)      .ctors       wa-  0x00029ad4  0x00000008  0x00029ad4  0x00011ad4
Info (OR_SD)      .dtors       wa-  0x00029adc  0x00000008  0x00029adc  0x00011adc
Info (OR_SD)      .jcr         wa-  0x00029ae4  0x00000004  0x00029ae4  0x00011ae4
Info (GDBT_PORT) Host: VOS35DUNCAN, Port: 12345
Info (GDBT_WAIT) Waiting for remote debugger to connect...
```

4.1.2.2 Connect the Eclipse Debugger

This is invoked as a 'Debug Configuration' from the Menu "run->Debug" or the drop down menu button identified by the same green run arrow icon

The 'launch' may already be visible in the quick launch section or will have to be selected from the "Debug Configuration..." menu

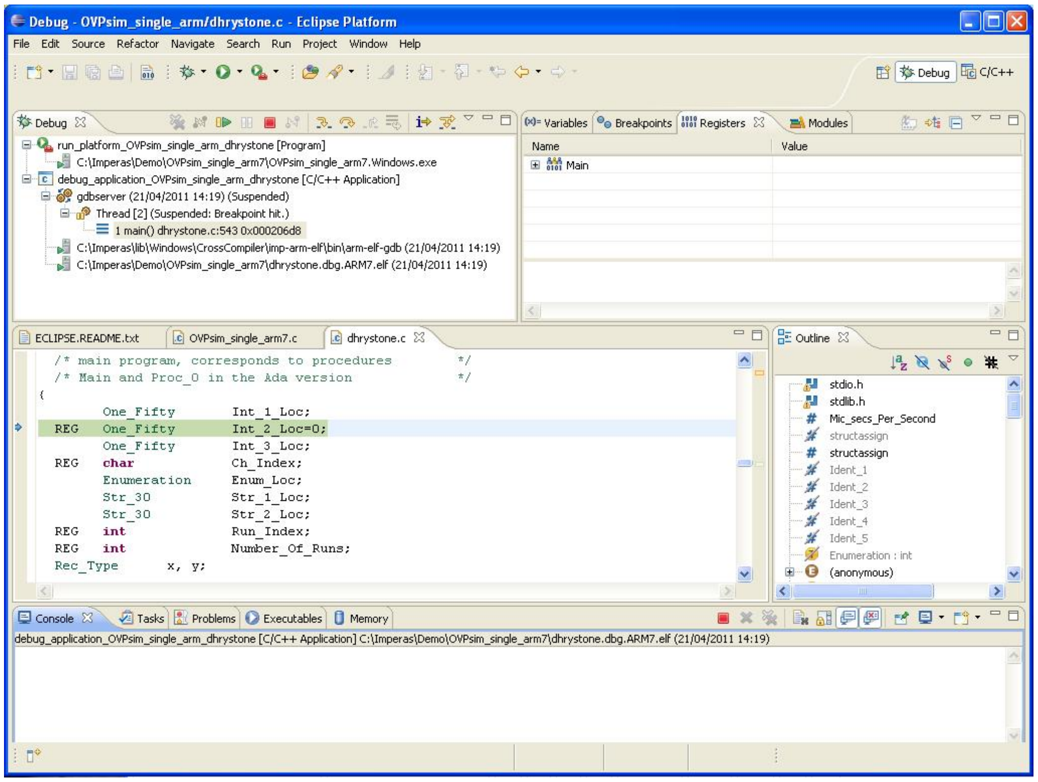
The launch to invoke is "debug_application_OVPsim_single_arm_<application name>



```
#############
# Using Eclip
#############

------------------------------------------------------------------
- Introduction
------------------------------------------------------------------
```

.

You should now be in the 'Debug' Perspective with control over application execution and visibility of all registers, variables etc.



```
run_platform_OVPsim_single_arm_dhrystone [Program]
    C:\Imperas\Demo\OVPsim_single_arm7\OVPsim_single_arm7.Windows.exe
debug_application_OVPsim_single_arm_dhrystone [C/C++ Application]
    gdbserver (21/04/2011 14:19) (Suspended)
        Thread [2] (Suspended: Breakpoint hit.)
            1 main() dhrystone.c:543 0x000206d8
    C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb (21/04/2011 14:19)
    C:\Imperas\Demo\OVPsim_single_arm7\dhrystone.dbg.ARM7.elf (21/04/2011 14:19)
```

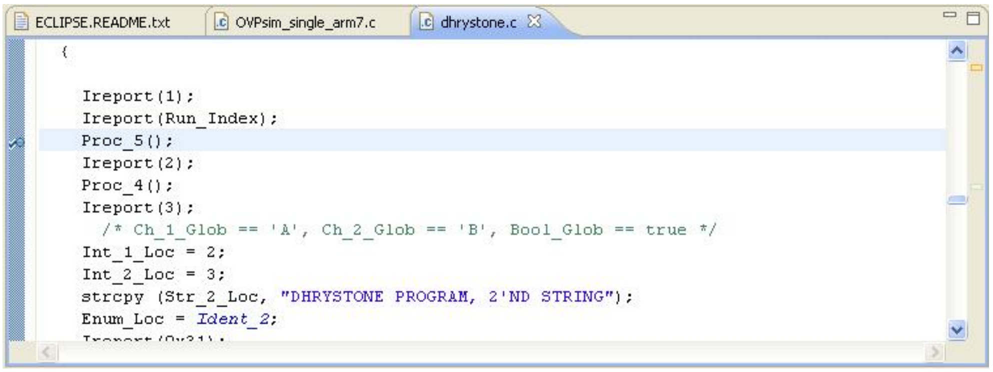
```
  /* main program, corresponds to procedures        */
  /* Main and Proc_0 in the Ada version             */
{
        One_Fifty       Int_1_Loc;
  REG   One_Fifty       Int_2_Loc=0;
        One_Fifty       Int_3_Loc;
  REG   char            Ch_Index;
        Enumeration     Enum_Loc;
        Str_30          Str_1_Loc;
        Str_30          Str_2_Loc;
  REG   int             Run_Index;
  REG   int             Number_Of_Runs;
  Rec_Type      x, y;
```

```
debug_application_OVPsim_single_arm_dhrystone [C/C++ Application] C:\Imperas\Demo\OVPsim_single_arm7\dhrystone.dbg.ARM7.elf (21/04/2011 14:19)
```

4.1.3 Example Debug Session for Dhrystone Benchmark application

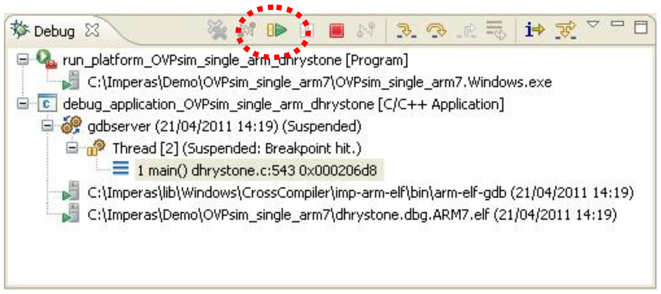
You should be familiar with the general debug control 'buttons' within Eclipse in order to work through the following

1. Set a breakpoint on the line containing the call to 'Proc_5();' by double clicking beside this line; this is found on line 608 (the line number may change slightly with versions of the application)



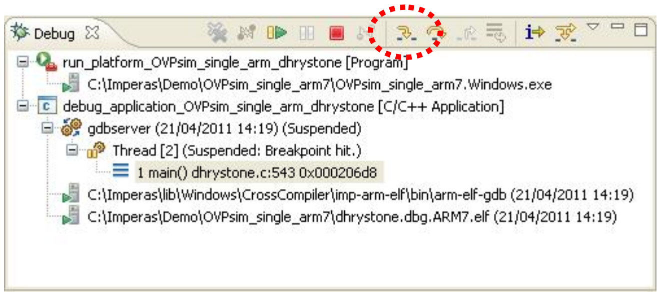
```
  {

    Ireport(1);
    Ireport(Run_Index);
    Proc_5();
    Ireport(2);
    Proc_4();
    Ireport(3);
      /* Ch_1_Glob == 'A', Ch_2_Glob == 'B', Bool_Glob == true */
    Int_1_Loc = 2;
    Int_2_Loc = 3;
    strcpy (Str_2_Loc, "DHRYSTONE PROGRAM, 2'ND STRING");
    Enum_Loc = Ident_2;
```

2. Continue the simulation using the green 'play' symbol



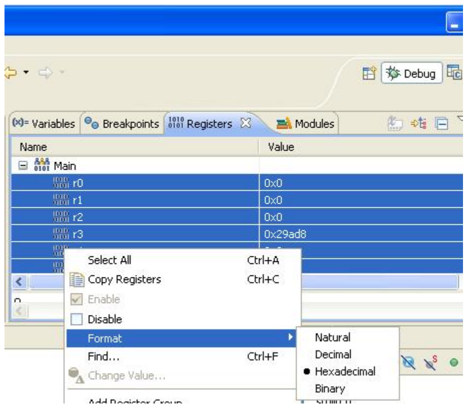
```
Debug
run_platform_OVPsim_single_arm_dhrystone [Program]
    C:\Imperas\Demo\OVPsim_single_arm7\OVPsim_single_arm7.Windows.exe
debug_application_OVPsim_single_arm_dhrystone [C/C++ Application]
    gdbserver (21/04/2011 14:19) (Suspended)
        Thread [2] (Suspended: Breakpoint hit.)
            1 main() dhrystone.c:543 0x000206d8
    C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb (21/04/2011 14:19)
    C:\Imperas\Demo\OVPsim_single_arm7\dhrystone.dbg.ARM7.elf (21/04/2011 14:19)
```

3. Step the application forward using the 'Step Into' button



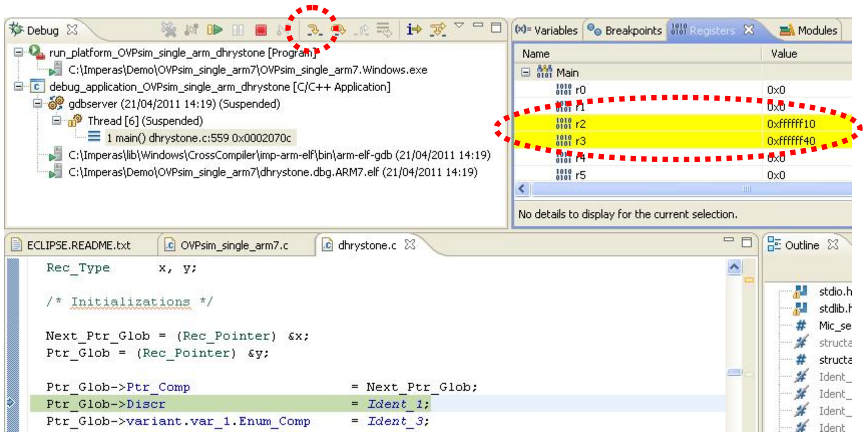
```
Debug
run_platform_OVPsim_single_arm_dhrystone [Program]
    C:\Imperas\Demo\OVPsim_single_arm7\OVPsim_single_arm7.Windows.exe
debug_application_OVPsim_single_arm_dhrystone [C/C++ Application]
    gdbserver (21/04/2011 14:19) (Suspended)
        Thread [2] (Suspended: Breakpoint hit.)
            1 main() dhrystone.c:543 0x000206d8
    C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb (21/04/2011 14:19)
    C:\Imperas\Demo\OVPsim_single_arm7\dhrystone.dbg.ARM7.elf (21/04/2011 14:19)
```

4. Open the register view; expand 'main', select registers and set the format to hexadecimal



Name	Value
Main	
r0	0x0
r1	0x0
r2	0x0
r3	0x29ad8

5. Step the application forward using the 'Step Into' button. This will show you the colored highlighting of the registers that change



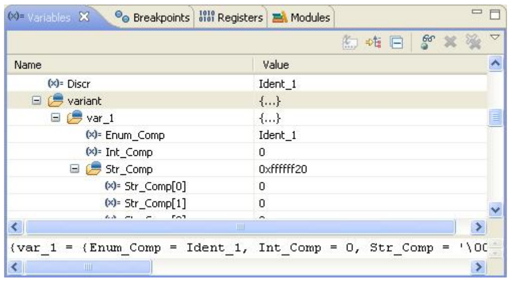
```
run_platform_OVPsim_single_arm_dhrystone [Program]
    C:\Imperas\Demo\OVPsim_single_arm7\OVPsim_single_arm7.Windows.exe
debug_application_OVPsim_single_arm_dhrystone [C/C++ Application]
    gdbserver (21/04/2011 14:19) (Suspended)
        Thread [6] (Suspended)
            1 main() dhrystone.c:559 0x0002070c
    C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb (21/04/2011 14:19)
    C:\Imperas\Demo\OVPsim_single_arm7\dhrystone.dbg.ARM7.elf (21/04/2011 14:19)
```

Name	Value
Main	
r0	0x0
r1	0x0
r2	0xffffff10
r3	0xffffff40
r4	0x0
r5	0x0

No details to display for the current selection.

```
Rec_Type      x, y;

/* Initializations */

Next_Ptr_Glob = (Rec_Pointer) &x;
Ptr_Glob = (Rec_Pointer) &y;

Ptr_Glob->Ptr_Comp                    = Next_Ptr_Glob;
Ptr_Glob->Discr                       = Ident_1;
Ptr_Glob->variant.var_1.Enum_Comp     = Ident_3;
```

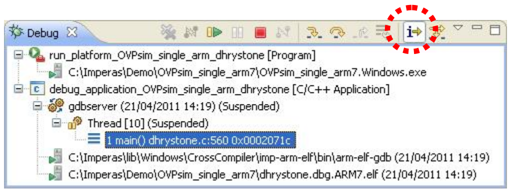
6. Open the Variables view; showing the current local variables



Name	Value
Discr	Ident_1
variant	{...}
var_1	{...}
Enum_Comp	Ident_1
Int_Comp	0
Str_Comp	0xffffff20
Str_Comp[0]	0
Str_Comp[1]	0

```
{var_1 = {Enum_Comp = Ident_1, Int_Comp = 0, Str_Comp = '\0C
```

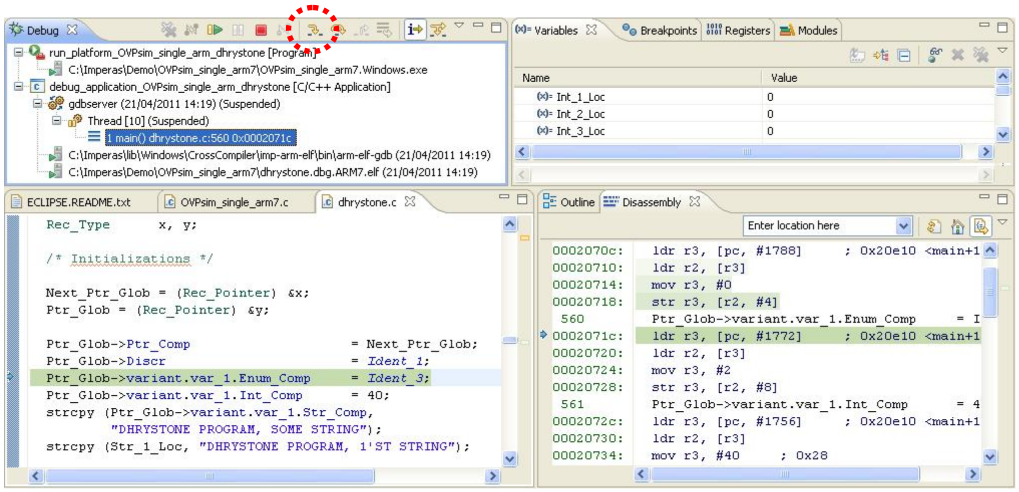
7. Select 'Instruction Stepping Mode' using the button.



```
run_platform_OVPsim_single_arm_dhrystone [Program]
    C:\Imperas\Demo\OVPsim_single_arm7\OVPsim_single_arm7.Windows.exe
debug_application_OVPsim_single_arm_dhrystone [C/C++ Application]
    gdbserver (21/04/2011 14:19) (Suspended)
        Thread [10] (Suspended)
            1 main() dhrystone.c:560 0x0002071c
    C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb (21/04/2011 14:19)
    C:\Imperas\Demo\OVPsim_single_arm7\dhrystone.dbg.ARM7.elf (21/04/2011 14:19)
```

.

8. Step the application forward using the 'Step Into' button this will open the disassembly view and highlight the instructions as they are stepped.



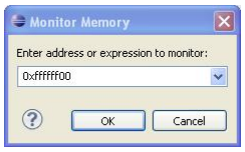
```
run_platform_OVPsim_single_arm_dhrystone [Program]
    C:\Imperas\Demo\OVPsim_single_arm7\OVPsim_single_arm7.Windows.exe
debug_application_OVPsim_single_arm_dhrystone [C/C++ Application]
    gdbserver (21/04/2011 14:19) (Suspended)
        Thread [10] (Suspended)
            1 main() dhrystone.c:560 0x0002071c
    C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb (21/04/2011 14:19)
    C:\Imperas\Demo\OVPsim_single_arm7\dhrystone.dbg.ARM7.elf (21/04/2011 14:19)
```

Name	Value
Int_1_Loc	0
Int_2_Loc	0
Int_3_Loc	0

```
Rec_Type      x, y;

/* Initializations */

Next_Ptr_Glob = (Rec_Pointer) &x;
Ptr_Glob = (Rec_Pointer) &y;

Ptr_Glob->Ptr_Comp                    = Next_Ptr_Glob;
Ptr_Glob->Discr                       = Ident_1;
Ptr_Glob->variant.var_1.Enum_Comp     = Ident_3;
Ptr_Glob->variant.var_1.Int_Comp      = 40;
strcpy (Ptr_Glob->variant.var_1.Str_Comp,
        "DHRYSTONE PROGRAM, SOME STRING");
strcpy (Str_1_Loc, "DHRYSTONE PROGRAM, 1'ST STRING");
```

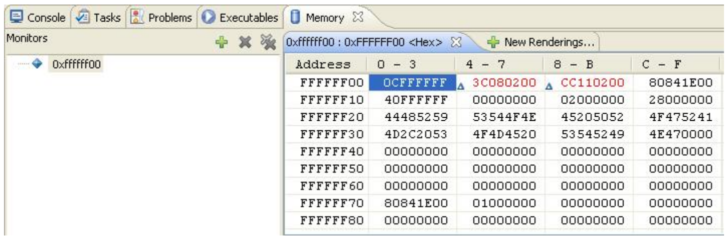
```
0002070c:   ldr r3, [pc, #1788]      ; 0x20e10 <main+1
00020710:   ldr r2, [r3]
00020714:   mov r3, #0
00020718:   str r3, [r2, #4]
 560        Ptr_Glob->variant.var_1.Enum_Comp     = I
0002071c:   ldr r3, [pc, #1772]      ; 0x20e10 <main+1
00020720:   ldr r2, [r3]
00020724:   mov r3, #2
00020728:   str r3, [r2, #8]
 561        Ptr_Glob->variant.var_1.Int_Comp      = 4
0002072c:   ldr r3, [pc, #1756]      ; 0x20e10 <main+1
00020730:   ldr r2, [r3]
00020734:   mov r3, #40     ; 0x28
```

9. In the 'Memory' window, select 'Add Memory Monitor' Use the value of the register 'SP' with bottom two bits as zeros to set the memory monitor base address so that it will show the program stack.



Enter address or expression to monitor:
0xffffff00

10. Move the application forward using the 'Step Return' button. The memory locations that are modified will be highlighted.



Address	0 - 3	4 - 7	8 - B	C - F
FFFFFF00	0CFFFFFF	3C080200	CC110200	80841E00
FFFFFF10	40FFFFFF	00000000	02000000	28000000
FFFFFF20	44485259	53544F4E	45205052	4F475241
FFFFFF30	4D2C2053	4F4D4520	53545249	4E470000
FFFFFF40	00000000	00000000	00000000	00000000
FFFFFF50	00000000	00000000	00000000	00000000
FFFFFF60	00000000	00000000	00000000	00000000
FFFFFF70	80841E00	01000000	00000000	00000000
FFFFFF80	00000000	00000000	00000000	00000000

11. To finish the simulation you may either:
    a. Use 'Resume' to continue to the end of the application
    b. Use 'Terminate' After either of these the debug perspective can be cleaned up with the 'Remove All terminated Launches' button

4.1.4 Re-building the Platform and Application

In order to re-build the platform file and the application source code you will require the following packages installed:

1. MSYS and MinGW environments (for use under Windows)
2. Processor Cross Compiler toolchain
3. OVPsim

With these packages installed re-building can be carried out.

This is done from the C/C++ perspective using the build button (a hammer icon) or using the menu item Project->Build All

5 A New Debugging Example

The following section provides information for the configuration of Eclipse Helios to allow the debugging of an application running on an OVP platform.

5.1 Command Line arguments for debugging a C Platform

If the platform has the Imperas Command Line Parser (CLP) included it makes available command line options to enable and control debugging of an application on the processor in a platform without the need to make any changes.

5.1.1 Specifying the debugger connection details

The debug port is enabled by specifying the argument --*port <port number>* on the command line. A specific port number may be specified or by setting *port number* to 0 the next available port is opened.

5.1.2 Nominating the debugged processor

In an OVPsim simulation only a single processor may be connected to a GDB debugger[1]. this requires that the processor is selected using the --*debugprocessor <processor name>*. In this case the processor name is the instance name in the platform, for example *platform/cpu0*

5.2 Creating a Debuggable C Platform

The following platform shows the modifications required to open a debug port for connection:

```
#define MODULE_DIR      "module"
#define MODULE_INSTANCE "u1"
#define CPU_INSTANCE    "cpu1"

int main(int argc, char ** argv) {

    int portNum;

    if(argc != 2) {
        opPrintf("usage: %s <port number>\n", argv[0]);
        return -1;
    }

    sscanf(argv[1], "%d", &portNum);

    optModuleP mi;

    opSessionInit(OP_VERSION);

    // enable debugging
    mi = opRootModuleNew(0, 0,
             OP_PARAMS (
```

[1] The Imperas Professional products allow the ability to attach a GDB debugger to any or all the processors defined in a platform. Imperas also provide alternative debugging solutions.

```
                 OP_PARAM_UNS32_SET(OP_FP_REMOTEDEBUGPORT,  portNum)
             )
         );

    // create the hardware definition by loading the module
    optModuleP u1 = opModuleNew(mi, MODULE_DIR, MODULE_INSTANCE, 0, 0);

     // run simulation
    opRootModuleSimulate(mi);

    // terminate simulation
    opSessionTerminate();

    return 0;
}
```

For a full explanation of OVPsim platform construction please see the documents *iGen Platform and Module Creation User Guide* and *Writing Platforms and Modules in C User Guide* . This section describes only those aspects of platform construction that relate to debugging.

5.2.1 Specifying the debugger connection details

The debugger connection is enabled by applying one of the debug parameters to the simulation root module.

The parameters that can be set are

Parameter	Description
OP_FP_REMOTEDEBUGPORT	Open a port and listen for a connection
OP_FP_GDBCONSOLE	Open a port and automatically connect a GDB debugger
OP_FP_GDBEGUI	Open a port and automatically connect a GDB debugger under the Eclipse GUI
OP_FP_MPDCONSOLE	Open a port and automatically connect the Imperas MPD debugger
OP_FP_MPDEGUI	Open a port and automatically connect the Imperas MPD debugger under the Eclipse GUI

Each of these parameters are applied in the same way to the root module creation as shown below

```
    mi = opRootModuleNew(0, 0,
             OP_PARAMS (
                 OP_PARAM_UNS32_SET(OP_FP_REMOTEDEBUGPORT, dbgPort)
             )
         );
```

Eclipse uses the GNU debugger for C/C++ debugging. GDB Remote Serial Protocol (RSP) debugging as supported by OVPsim uses standard operating system sockets on the host running OVPsim and GDB.

The `dbgPort` argument specifies the socket port on which to accept a debugger connection. The special value of zero allows OVPsim to choose any free port on the host, otherwise the specified port number is used. For the example platform the port number is given by the first command line argument.

5.2.2 Nominating the debugged processor

Any processor instance or a number of instances can be selected to be debugged using the *opProcessorDebug* API function:

In the following code we iterate over the processor instanced in a module until one with a matching name is found.

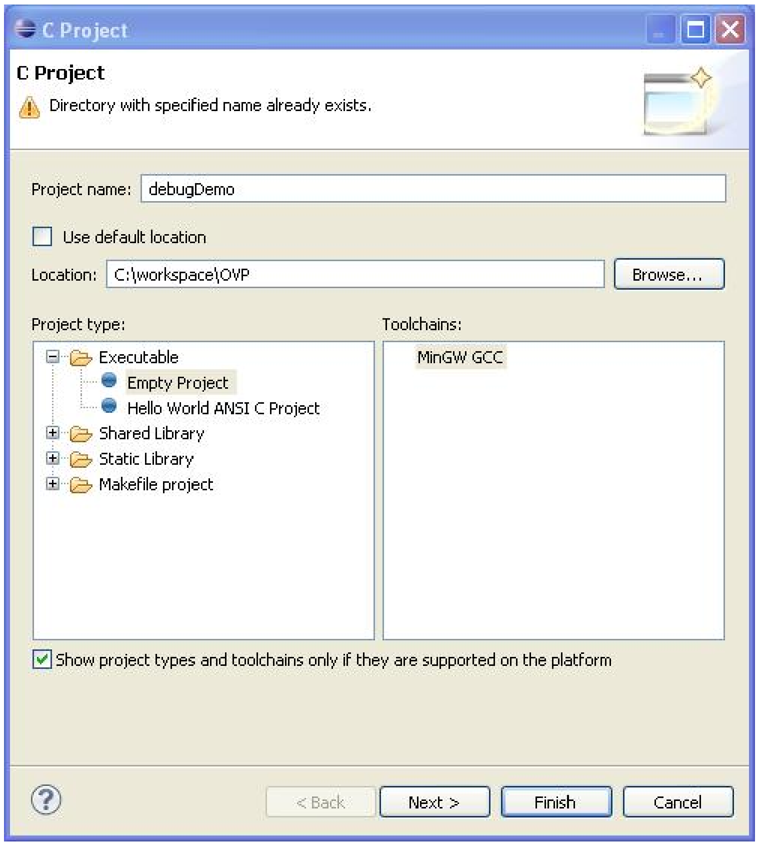
```
#define CPU_INSTANCE “cpu1”
…
    optProcessorP proc = NULL;
    while ((proc = opProcessorNext(u1, proc))) {
        opPrintf("found %s\n", opObjectName(proc));
        if (strcmp(CPU_INSTANCE, opObjectName(proc)) == 0) {
              opProcessorDebug(proc);
        }
    }
```

5.3 Preparing an Eclipse Project

This section shows all the steps needed to create an Eclipse project which allows the example platform and target application to be built and debugged in the Eclipse development environment.

5.3.1 Creating a New Project

Create a new C project by selecting “New” from the “File” menu, then selecting “C Project”. If this option is not present you may not have the C development plugins (CDT) for Eclipse – see section 2.1.


C Project

Directory with specified name already exists.

Project name: debugDemo

☐ Use default location

Location: C:\workspace\OVP

Project type:
- Executable
  - Empty Project
  - Hello World ANSI C Project
- Shared Library
- Static Library
- Makefile project

Toolchains: MinGW GCC

☑ Show project types and toolchains only if they are supported on the platform

Select “Makefile project” as we will use an external Makefile within the project. Specify a project name and optionally a location. Note that the project location path *must not* include spaces as Eclipse passes the project directory to an underlying Gnu debugger in a way that does not handle spaces correctly.

The default workspace and hence project location on Windows includes “Documents and Settings” so in this case you should uncheck the “Use default location” checkbox and specify a location which does not include spaces in the name. Click the “Finish” button to create the project.

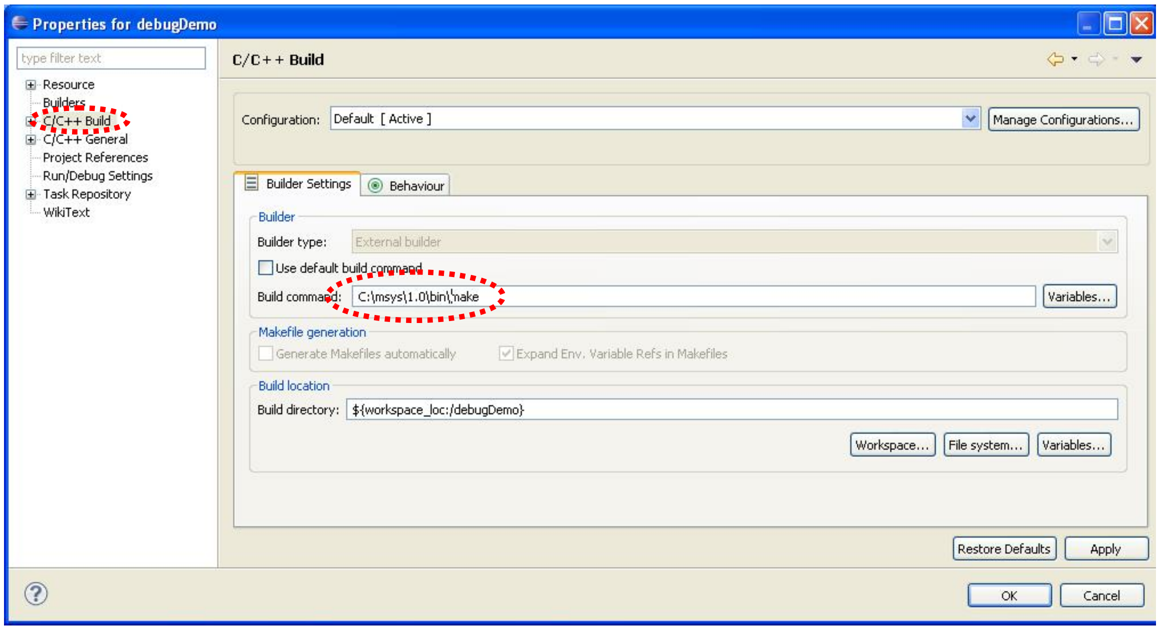
The new project contains no Makefile or source files so Eclipse will show an error message if it attempts to build at this point.

5.3.2 Configuring Project Build Commands

5.3.2.1 Using MINGW ‘make’

Change the project build configuration so that Eclipse can find the MinGW “make” tool used to build the project. This step may not be necessary if you already have a version of GNU “make” on your system path. Select “Properties” from the “Project” menu and select the “C/C++ Build” pane.

.



Configuration: Default [Active]

Builder type: External builder

☐ Use default build command

Build command: C:\msys\1.0\bin\make

☐ Generate Makefiles automatically ✓ Expand Env. Variable Refs in Makefiles

Build directory: ${workspace_loc:/debugDemo}

Uncheck the "Use default build command" checkbox and specify the full path of the MinGW/MSYS make executable on your system in the "Build command:" text box. In the example shown the default MSYS install location of "c:\msys\1.0\bin\make" is used.

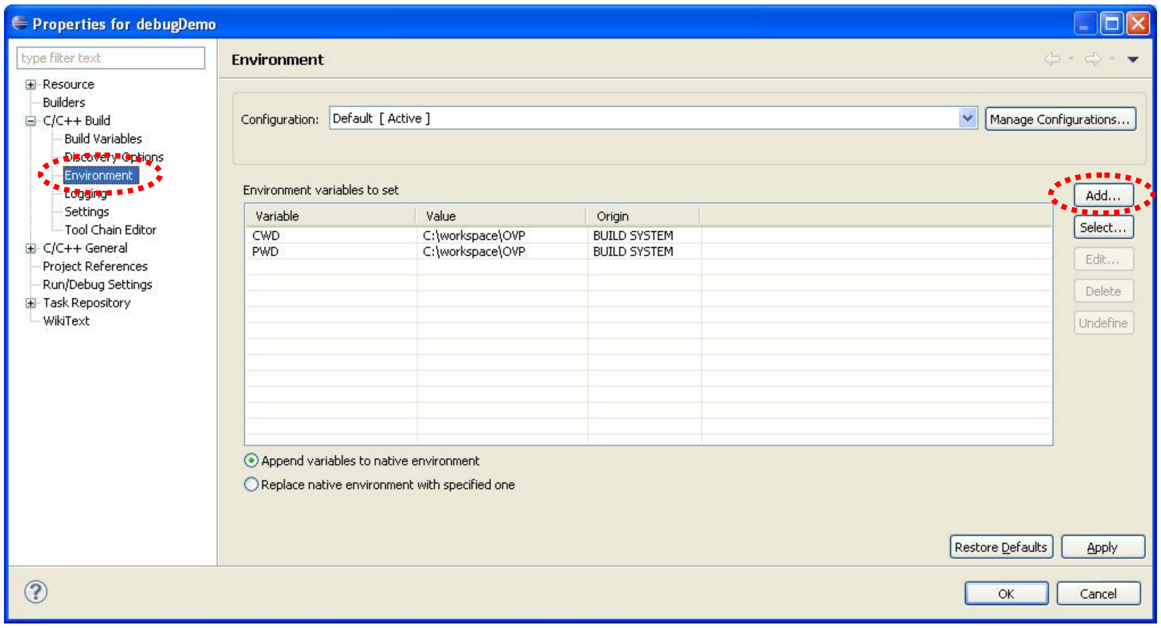
5.3.2.2 Adding MinGW tools onto PATH

It is recommended that your MinGW binary path is appended to the system path within your Windows environment. In the example the MinGW install location of "c:\msys\1.0\mingw\bin" is used.

However, it may be added into your Eclipse environment by adding a variable "PATH" which includes the current value of PATH (given by %PATH%) and the MinGW path separated by a semi-colon.

Add the MinGW native compilation tools to the path by selecting the "Environment" tab of the same dialog and pressing the "Add…" button.

.



Properties for debugDemo

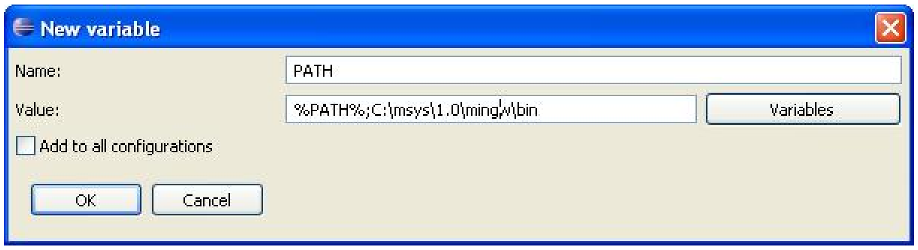
Environment

Configuration: Default [Active]

Environment variables to set

Variable	Value	Origin
CWD	C:\workspace\OVP	BUILD SYSTEM
PWD	C:\workspace\OVP	BUILD SYSTEM

(●) Append variables to native environment
( ) Replace native environment with specified one



New variable

Name: PATH

Value: %PATH%;C:\msys\1.0\mingw\bin

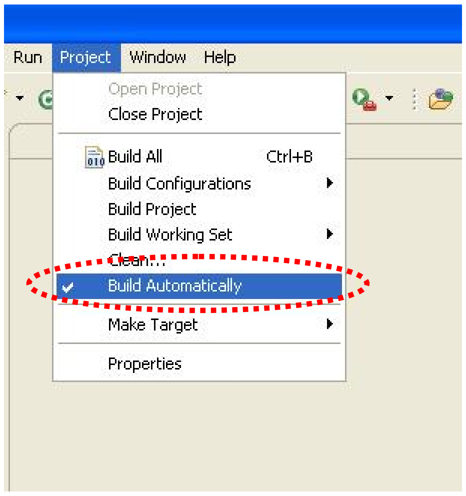
[ ] Add to all configurations

Close the properties dialog by pressing “OK”.

5.3.3 Deselect 'Autobuild' from Eclipse

By default, a new project in Eclipse will have autobuild selected. This can be useful, but when debugging cross compiled applications on a C platform this may cause the build process to be re-invoked between the launching of the simulation platform and the attaching of the debugger to allow application debug.

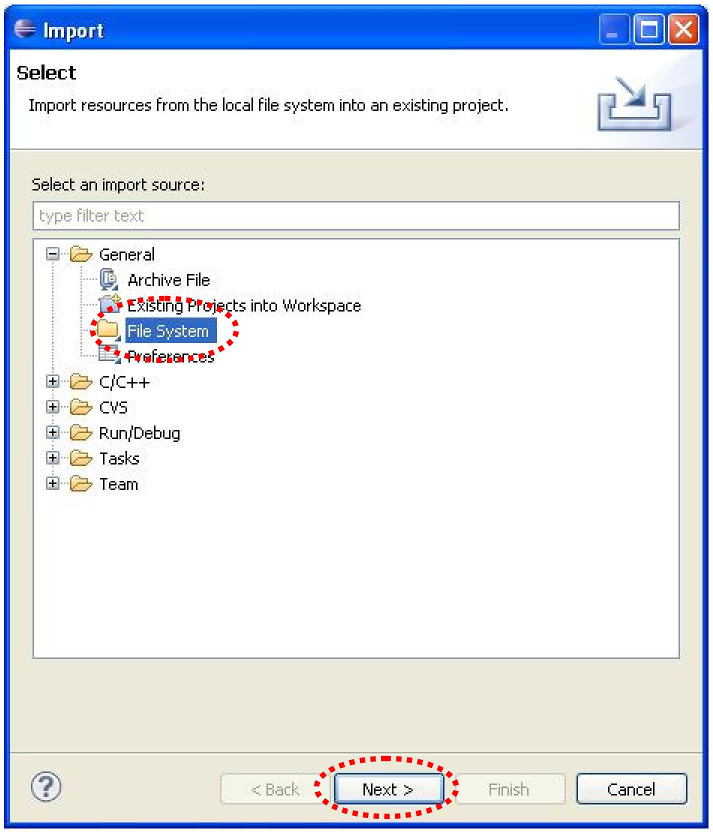
Select the "Project" menu and click on 'build automatically' to de-select it.



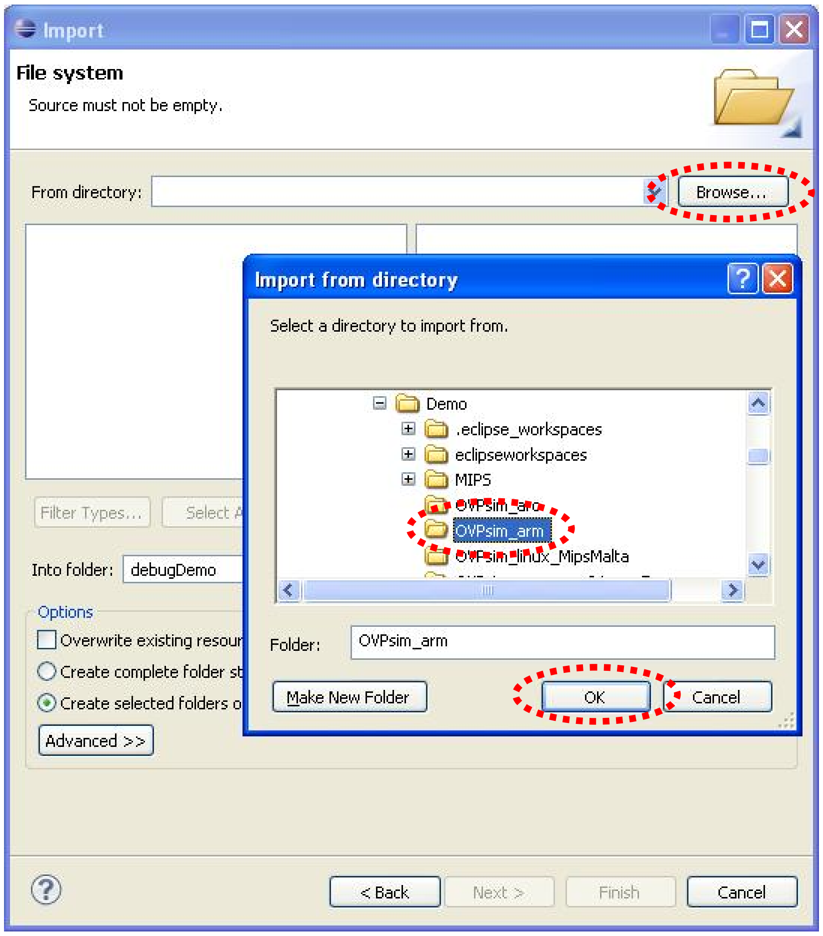
5.4 Importing a Demo to the Project

For the purposes of this document we will import the demo provided as part of an OVPsim installation for the ARM processor, that is, OVPsim_arm. We could equally choose the OVPsim_mips32 or other processor demo. The only difference is the Cross Compiler toolchain setting and the GDB debugger selected. These will be notified in the later sections.

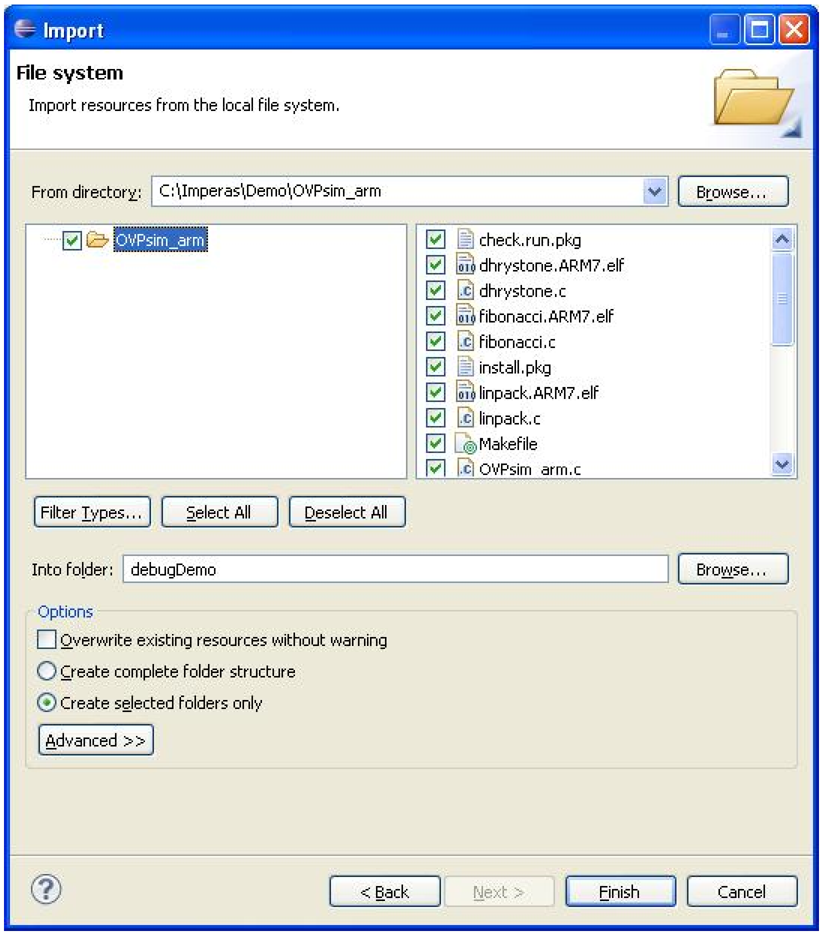
The files are imported to a project by selecting the File->Import menu and selecting 'File System' and followed by 'Next'



This brings up a further menu from which you may browse to the Imperas installation and the installed Demos. Select the demo directory that you wish to import. This may be any demo supplied as part of an OVP installation.



Once the directory containing the demo files has been selected, you will be prompted for which files you wish to import. In general this is ALL files, if you wish to exclude some files, please ensure that the source files and the exe and elf files are imported.



From directory: C:\Imperas\Demo\OVPsim_arm

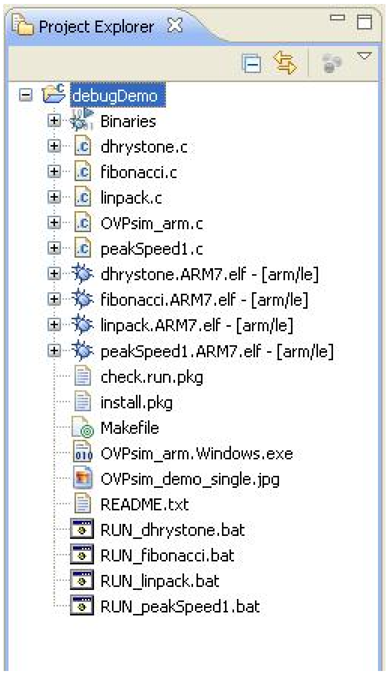
[✓] OVPsim_arm

[✓] check.run.pkg
[✓] dhrystone.ARM7.elf
[✓] dhrystone.c
[✓] fibonacci.ARM7.elf
[✓] fibonacci.c
[✓] install.pkg
[✓] linpack.ARM7.elf
[✓] linpack.c
[✓] Makefile
[✓] OVPsim_arm.c

Into folder: debugDemo

[ ] Overwrite existing resources without warning
( ) Create complete folder structure
(●) Create selected folders only

Once imported your project will contain all the files



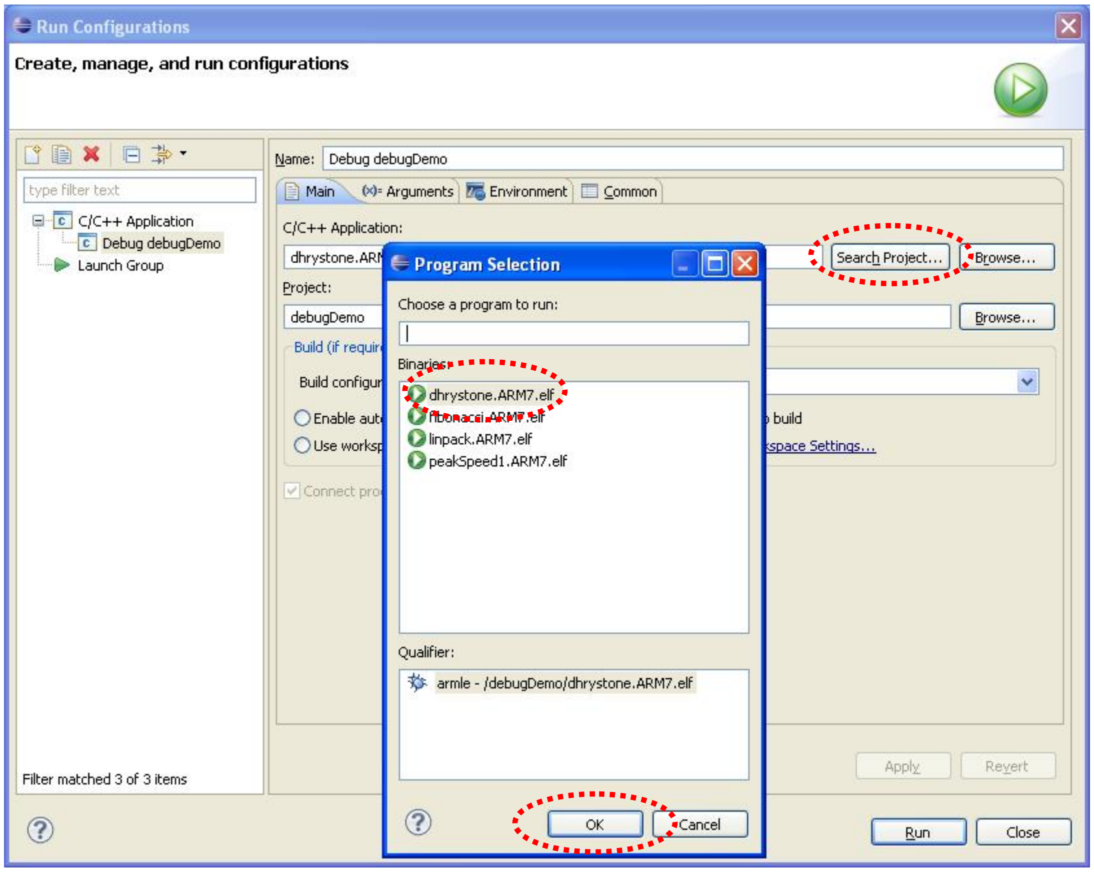
```
debugDemo
  Binaries
  dhrystone.c
  fibonacci.c
  linpack.c
  OVPsim_arm.c
  peakSpeed1.c
  dhrystone.ARM7.elf - [arm/le]
  fibonacci.ARM7.elf - [arm/le]
  linpack.ARM7.elf - [arm/le]
  peakSpeed1.ARM7.elf - [arm/le]
  check.run.pkg
  install.pkg
  Makefile
  OVPsim_arm.Windows.exe
  OVPsim_demo_single.jpg
  README.txt
  RUN_dhrystone.bat
  RUN_fibonacci.bat
  RUN_linpack.bat
  RUN_peakSpeed1.bat
```

5.5 Creating Launch Configurations

5.5.1 A Debug Configuration

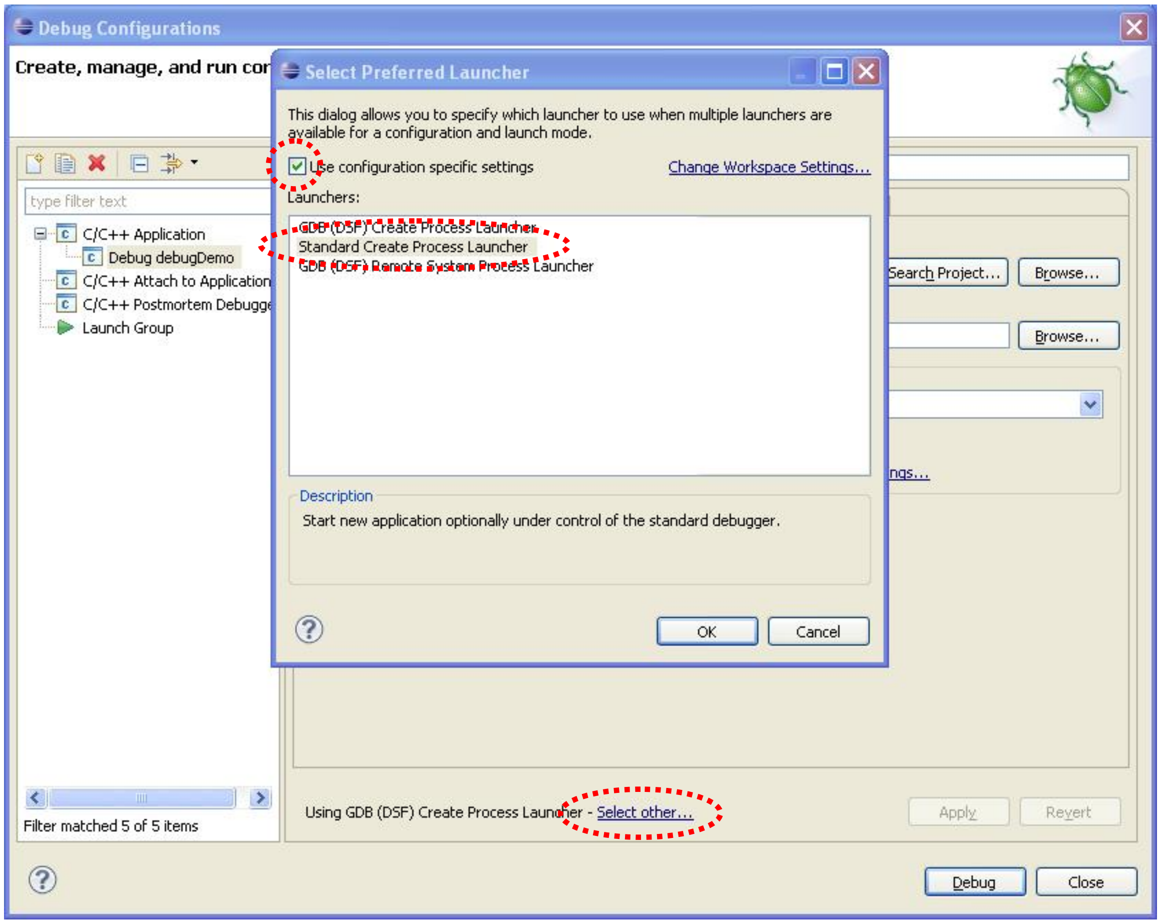
Create a debug configuration for the simulated application by selecting “Open Debug Dialog…” from the “Run” menu. Create a new “C/C++ Application” configuration.

Select the application executable as the C application file using the “search project” button



We now need to select the use of the standard debug launcher, not the DSF launcher. This is done by selecting the 'select other' option.

Select the 'Use configuration specific settings' and select the use of the 'standard Create Process Launcher'



This dialog allows you to specify which launcher to use when multiple launchers are available for a configuration and launch mode.

[✓] Use configuration specific settings

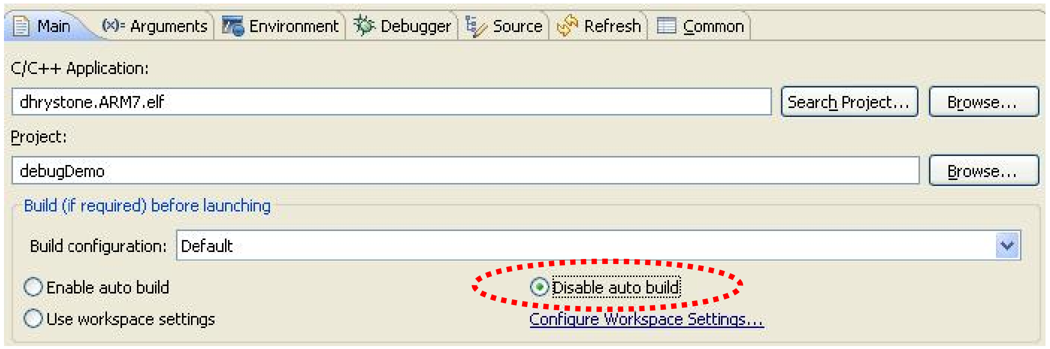
GDB (DSF) Create Process Launcher
Standard Create Process Launcher
GDB (DSF) Remote System Process Launcher

Start new application optionally under control of the standard debugger.

Using GDB (DSF) Create Process Launcher - Select other...

On the "Debugger" tab of this dialog select "gdbserver" as the debugger type and browse for the GDB executable for the simulated processor type.

It is recommended that the auto build feature is also disabled in this launcher.



C/C++ Application: dhrystone.ARM7.elf

Project: debugDemo

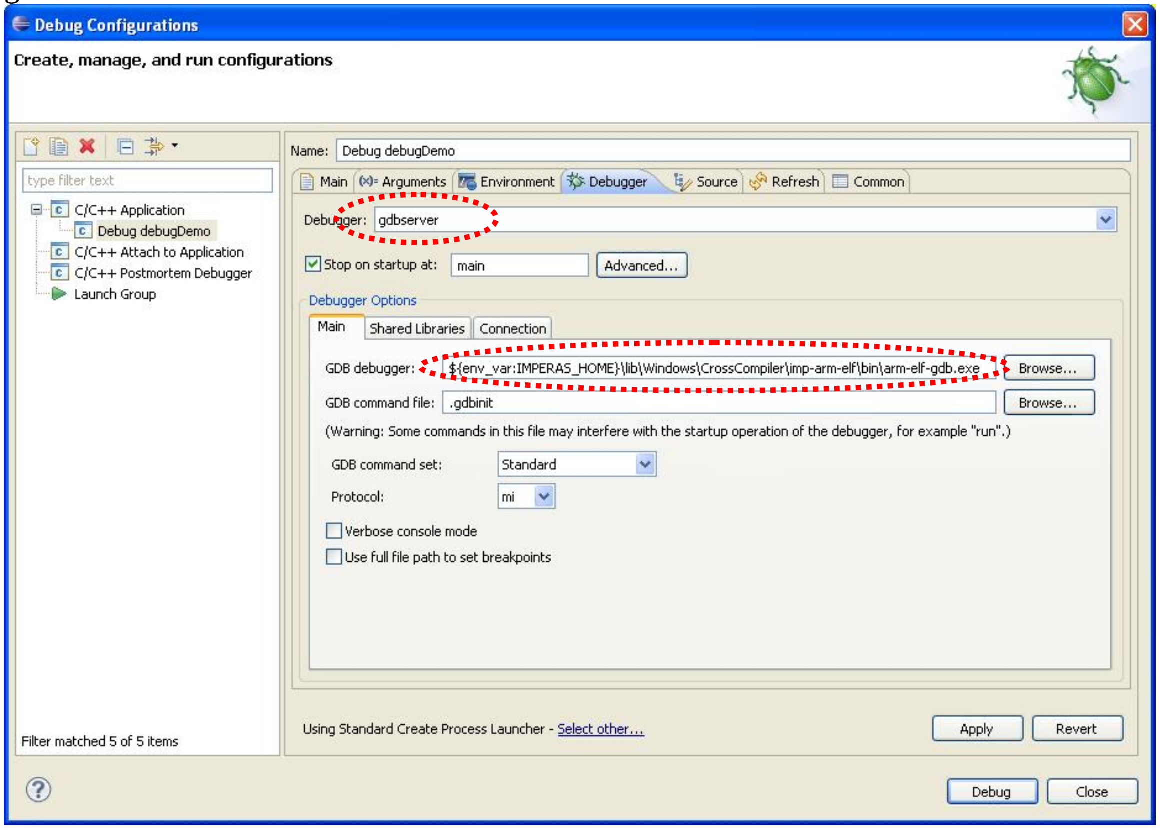
Build (if required) before launching

Build configuration: Default

○ Enable auto build [✓ ● Disable auto build]
○ Use workspace settings

Assuming you installed OVPsim and the toolchains to the default location “C:\Imperas”, the Gnu tools for simulated processors, including cross-compilers and the Gnu debugger can be found in “lib\Windows\CrossCompiler”.

The ARM GDB in this case is found at “C:\Imperas\lib\Windows\CrossCompiler\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb.exe”.



Name: Debug debugDemo

Debugger: gdbserver

[✓] Stop on startup at: main

GDB debugger: ${env_var:IMPERAS_HOME}\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb.exe

GDB command file: .gdbinit

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

GDB command set: Standard

Protocol: mi

[ ] Verbose console mode

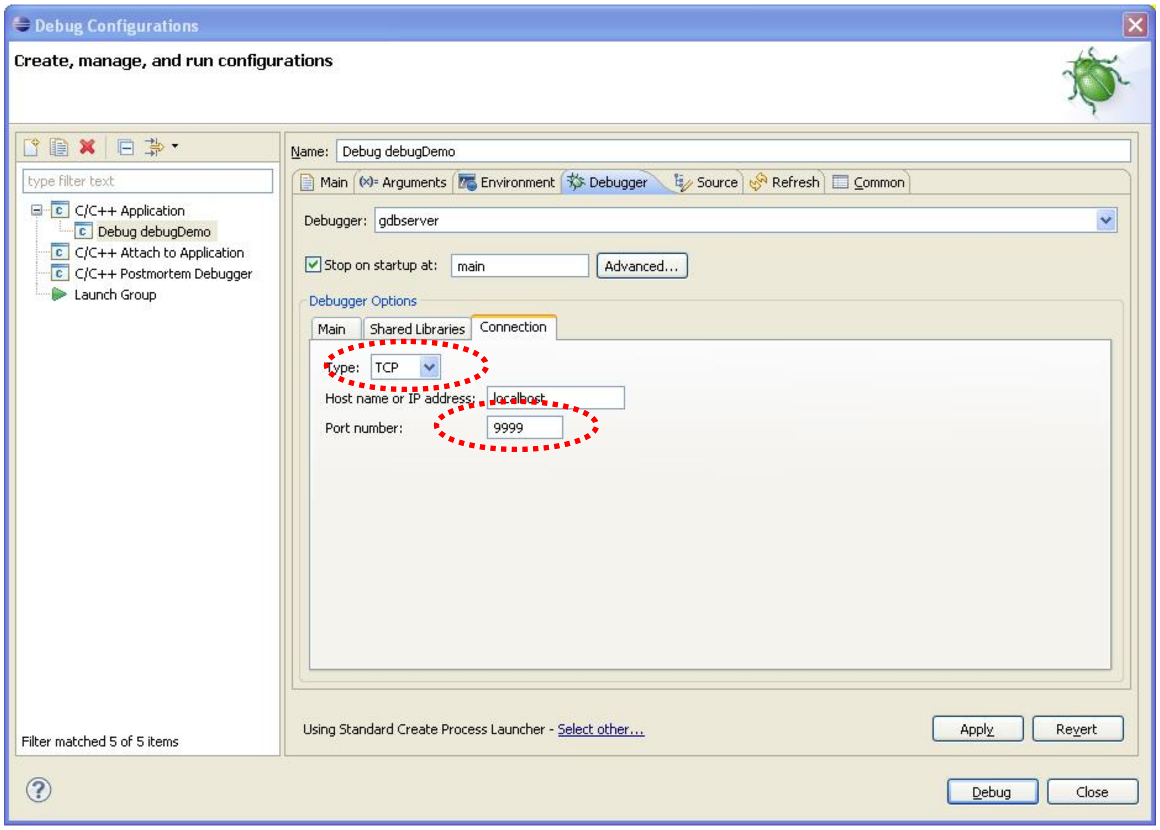
[ ] Use full file path to set breakpoints

Using Standard Create Process Launcher - Select other...

Instead of using the full path to the fixed location C:\Imperas it may instead be replaced with the environment variable IMPERAS_HOME.

In Eclipse the environment variable is accessed using the ${env_var:IMPERAS_HOME} construct, as shown.

Finally, specify the target connection settings:



Name: Debug debugDemo

Debugger: gdbserver

☑ Stop on startup at: main

Type: TCP

Host name or IP address: localhost

Port number: 9999

Select “TCP” as the connection.
Choose a port number, here we have initially chosen 9999. You may need to change this if there is a conflict on your host. The port number selected here must match the port number we select on the platform launch we will define next.

Press “Apply” then “Close” to save the debug configuration without starting it.

5.5.2 Creating a Shortcut to Launch the OVPsim Platform

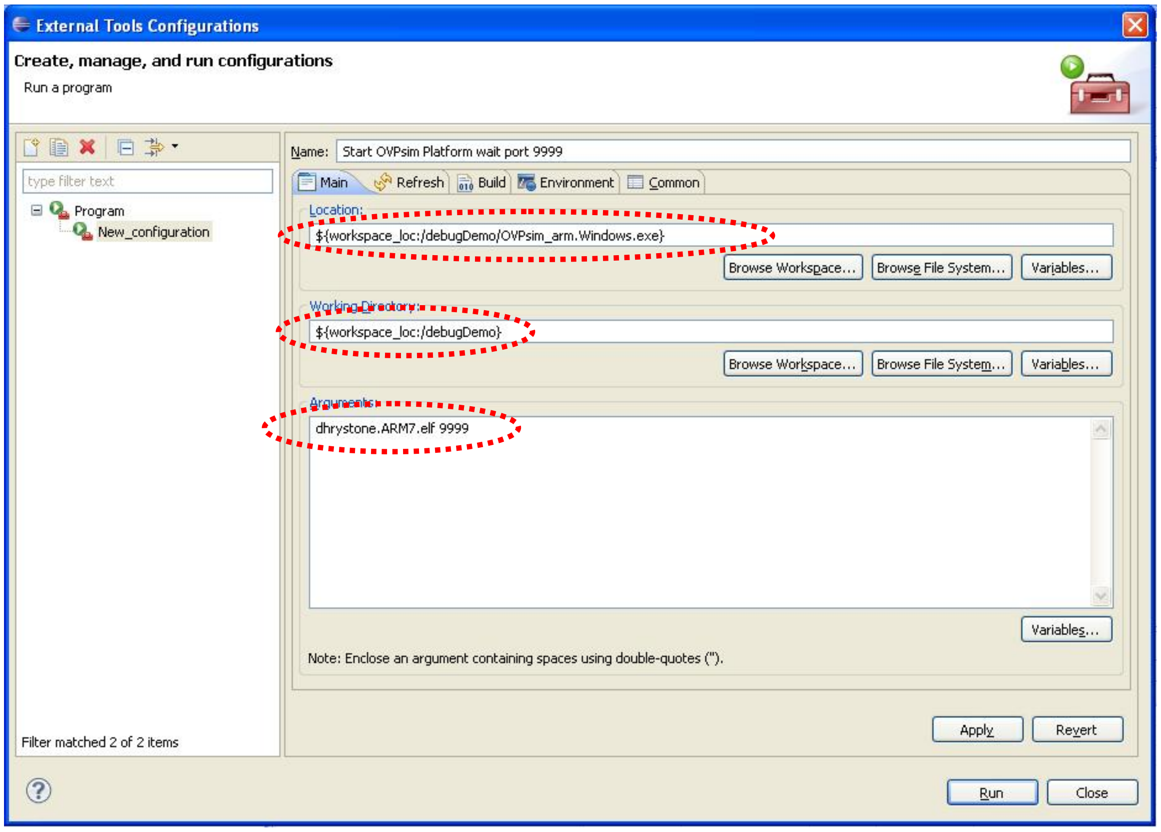
Before each debug session the OVPsim platform must be launched on the host machine. The Eclipse “External Tools” facility is a convenient way to do this. Select “Open External Tools Dialog…” from the “External Tools” menu within the “Run” menu.

Create a new tool configuration and give it a meaningful name. Specify the location for the platform executable by pressing the “Browse Workspace…” button and selecting the “platform.<IMPERAS_ARCH>.exe” file built earlier. Press the “Browse Workspace…” button for the working directory, and select the top-level project directory.

The arguments added into the arguments section must match the arguments that the executable platform expects.

In this example the platform takes two arguments; the first argument is the application elf file and the second argument is the port number on which it waits for a debugger connection.

Specify the port 9999 initially, matching the port we chose in the debug configuration in section 5.5.1.



Name: Start OVPsim Platform wait port 9999

Location: ${workspace_loc:/debugDemo/OVPsim_arm.Windows.exe}

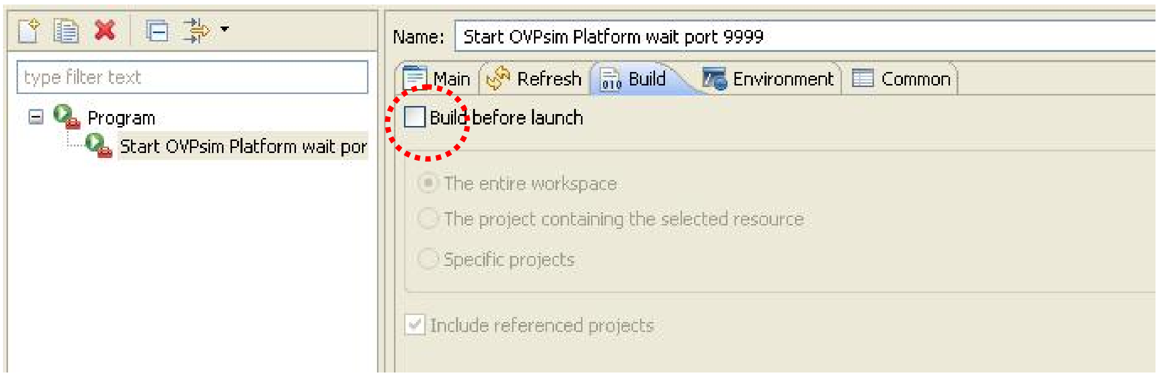
Working Directory: ${workspace_loc:/debugDemo}

Arguments: dhrystone.ARM7.elf 9999

Note: Enclose an argument containing spaces using double-quotes (").

Press “Apply”, then “Close” to save the external tool configuration without starting it.

NOTE: Depending upon the Demo you have selected, you may need to modify the platform C source file so that it is a “debuggable platform” as described in section 5.2, Creating a Debuggable C Platform, Ensure the ‘Build before launch’ checkbox is not selected. We want to ensure we have control over when the project is built, as we are using multiple launches.



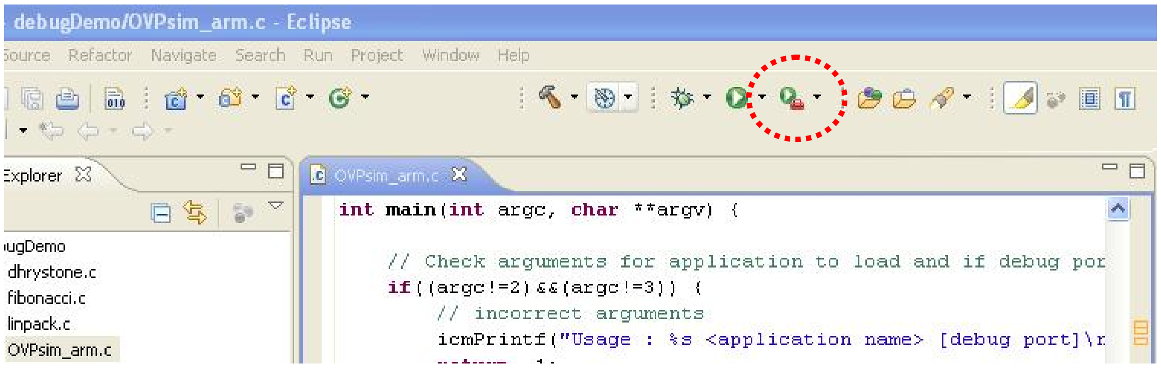
Name: Start OVPsim Platform wait port 9999

Build before launch [ ]

Include referenced projects [✓]

5.5.3 Testing the Debug Configuration

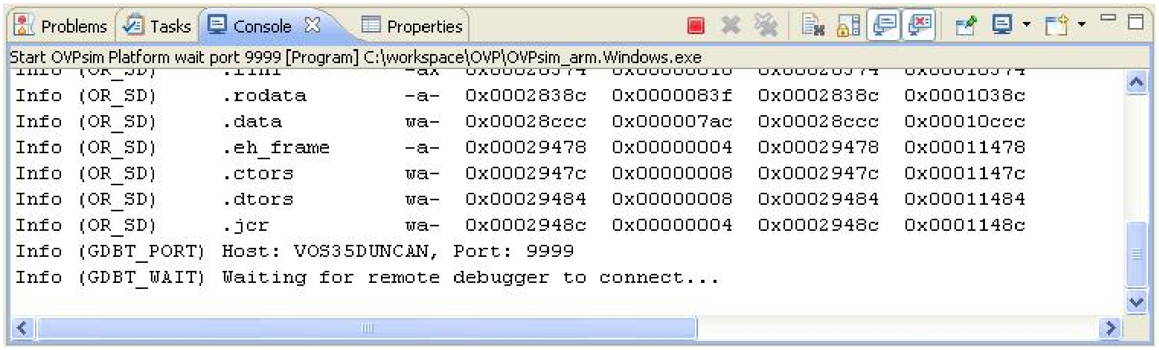
Earlier sections showed how to create a debug configuration for a simulated application and an external tool shortcut to launch an OVPsim platform. To test that we are ready to debug, first launch the OVPsim platform by running the external tool shortcut. The easiest way to do this is to press the external tool icon on the Eclipse toolbar.


debugDemo/OVPsim_arm.c - Eclipse
Source Refactor Navigate Search Run Project Window Help
Explorer
ugDemo
dhrystone.c
fibonacci.c
linpack.c
OVPsim_arm.c
OVPsim_arm.c

```
int main(int argc, char **argv) {

    // Check arguments for application to load and if debug por
    if((argc!=2)&&(argc!=3)) {
        // incorrect arguments
        icmPrintf("Usage : %s <application name> [debug port]\r
```

The first time you press this icon you will need to select the OVPsim platform shortcut from the list in the external tool dialog, then press the “Run” button. Subsequently it will automatically re-run the external tool you ran last.

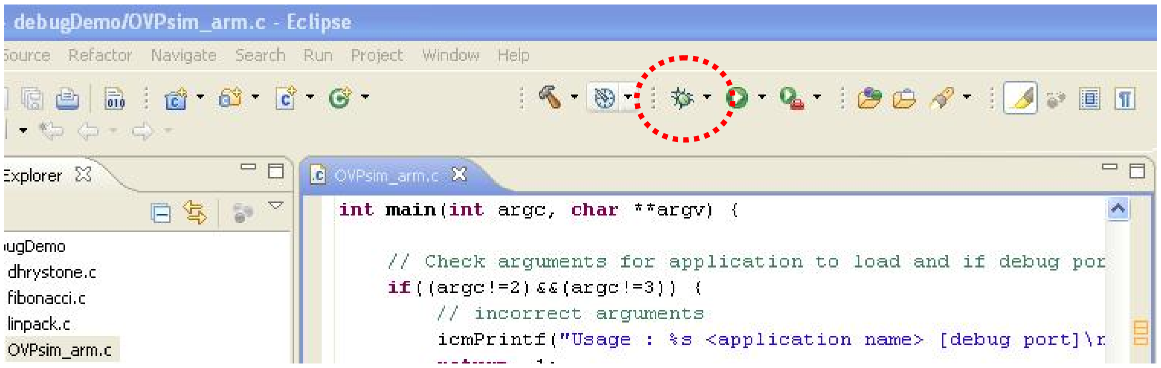
The Eclipse Console view will show the output from the OVPsim platform executable:



```
Start OVPsim Platform wait port 9999 [Program] C:\workspace\OVP\OVPsim_arm.Windows.exe
Info (OR_SD)     .rodata      -a-  0x0002838c  0x0000083f  0x0002838c  0x0001038c
Info (OR_SD)     .data        wa-  0x00028ccc  0x000007ac  0x00028ccc  0x00010ccc
Info (OR_SD)     .eh_frame    -a-  0x00029478  0x00000004  0x00029478  0x00011478
Info (OR_SD)     .ctors       wa-  0x0002947c  0x00000008  0x0002947c  0x0001147c
Info (OR_SD)     .dtors       wa-  0x00029484  0x00000008  0x00029484  0x00011484
Info (OR_SD)     .jcr         wa-  0x0002948c  0x00000004  0x0002948c  0x0001148c
Info (GDBT_PORT) Host: VOS35DUNCAN, Port: 9999
Info (GDBT_WAIT) Waiting for remote debugger to connect...
```

If port 9999 is already in use on your system the console will show an error message and you will need to modify both the debug configuration and the external tool configuration to select a new port number.

Once the OVPsim platform is running and waiting for a debugger to connect, launch the debug configuration we created earlier. Again, the easiest way to do this is to press the debug icon on the Eclipse toolbar.


debugDemo/OVPsim_arm.c - Eclipse
Source Refactor Navigate Search Run Project Window Help
Explorer
ugDemo
dhrystone.c
fibonacci.c
linpack.c
OVPsim_arm.c
OVPsim_arm.c

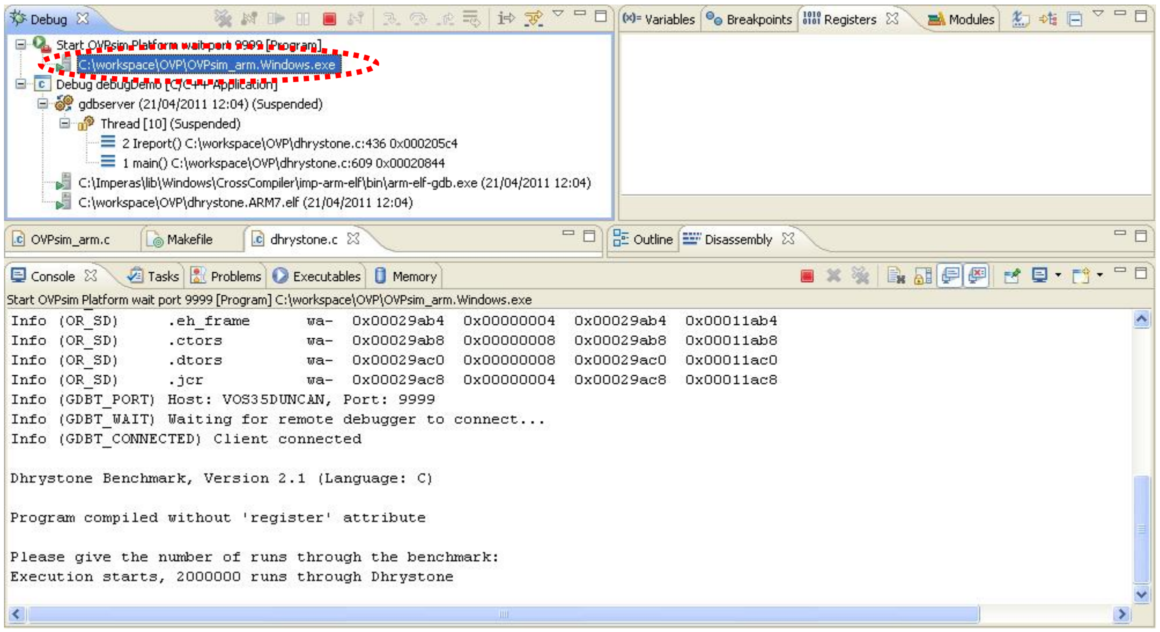
```
int main(int argc, char **argv) {

    // Check arguments for application to load and if debug por
    if((argc!=2)&&(argc!=3)) {
        // incorrect arguments
        icmPrintf("Usage : %s <application name> [debug port]\r
```

If debugging is configured and started correctly, Eclipse will switch (or prompt to switch) to the debug perspective, with the simulated application stopped at the entry to its main function.

5.5.4 Switching Console Output

The output from the console terminal window depends upon the selection made in the Debug window.

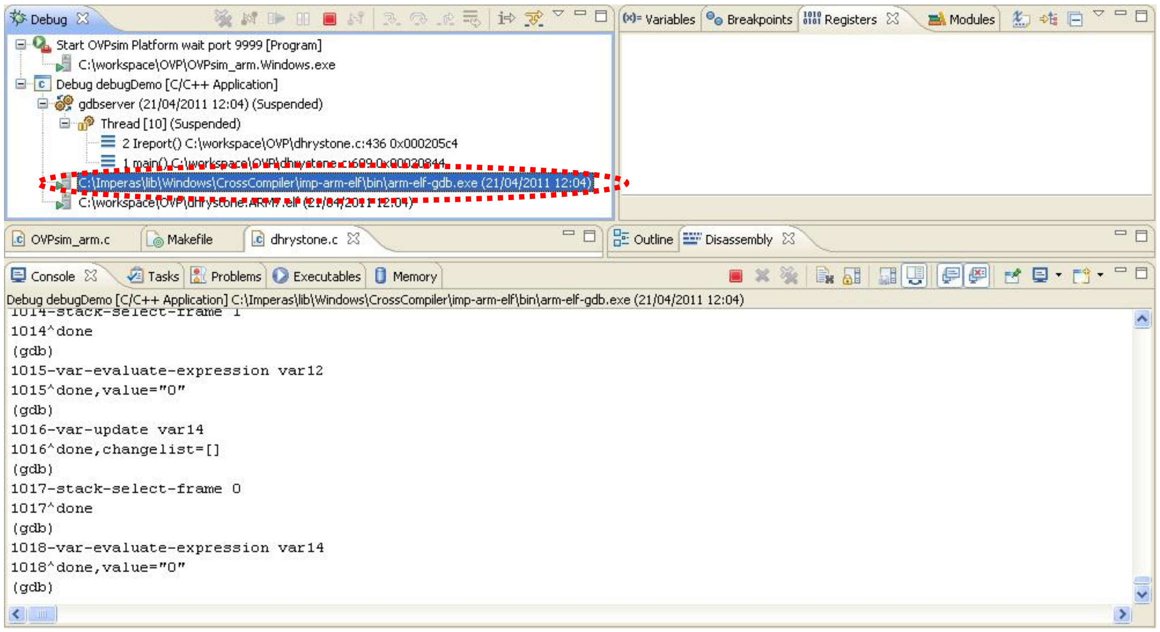
Here we have the output from the OVPsim platform executing after selecting the platform executable.



```
Start OVPsim Platform wait port 9999 [Program]
    C:\workspace\OVP\OVPsim_arm.Windows.exe
Debug debugDemo [C/C++ Application]
    gdbserver (21/04/2011 12:04) (Suspended)
        Thread [10] (Suspended)
            2 Ireport() C:\workspace\OVP\dhrystone.c:436 0x000205c4
            1 main() C:\workspace\OVP\dhrystone.c:609 0x00020844
    C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb.exe (21/04/2011 12:04)
    C:\workspace\OVP\dhrystone.ARM7.elf (21/04/2011 12:04)
```

```
Start OVPsim Platform wait port 9999 [Program] C:\workspace\OVP\OVPsim_arm.Windows.exe
Info (OR_SD)      .eh_frame      wa-  0x00029ab4  0x00000004  0x00029ab4  0x00011ab4
Info (OR_SD)      .ctors         wa-  0x00029ab8  0x00000008  0x00029ab8  0x00011ab8
Info (OR_SD)      .dtors         wa-  0x00029ac0  0x00000008  0x00029ac0  0x00011ac0
Info (OR_SD)      .jcr           wa-  0x00029ac8  0x00000004  0x00029ac8  0x00011ac8
Info (GDBT_PORT) Host: VOS35DUNCAN, Port: 9999
Info (GDBT_WAIT) Waiting for remote debugger to connect...
Info (GDBT_CONNECTED) Client connected

Dhrystone Benchmark, Version 2.1 (Language: C)

Program compiled without 'register' attribute

Please give the number of runs through the benchmark:
Execution starts, 2000000 runs through Dhrystone
```

In the next picture we see the GDB output, showing the commands passed between Eclipse and GDB with verbose selected, in the console after selecting the GDB debugger.

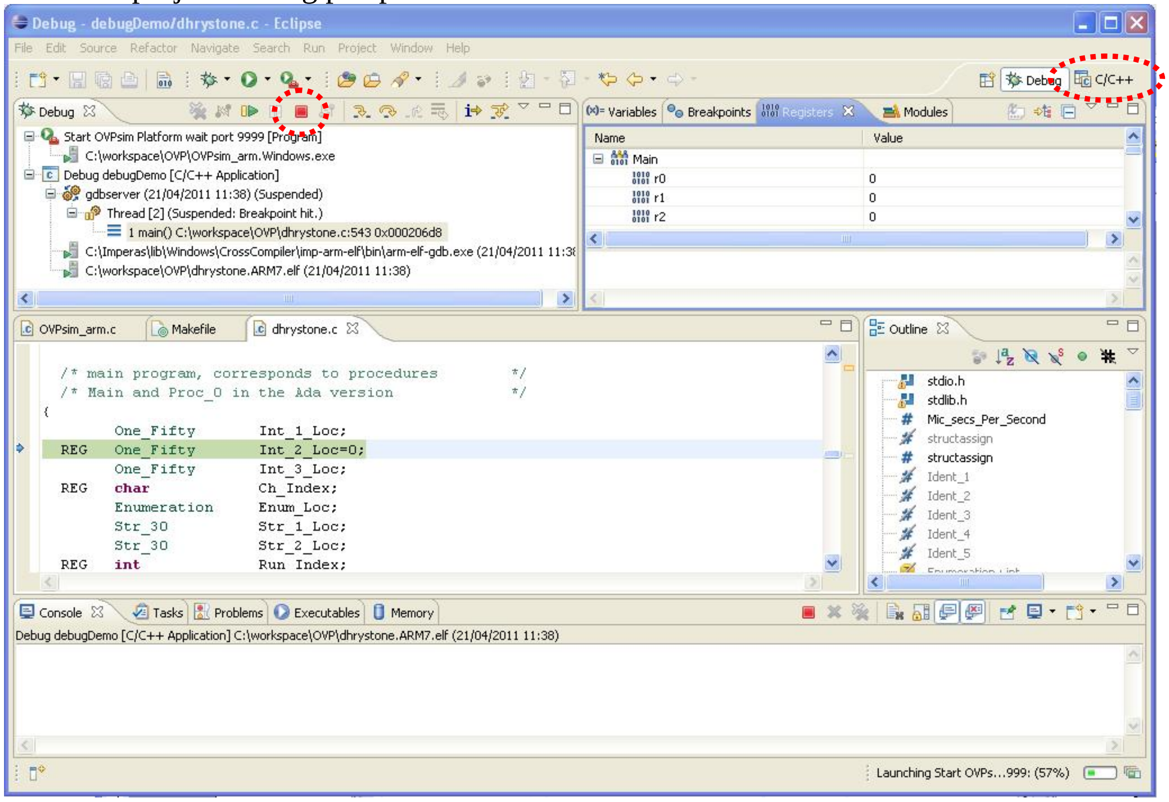


```
Start OVPsim Platform wait port 9999 [Program]
    C:\workspace\OVP\OVPsim_arm.Windows.exe
Debug debugDemo [C/C++ Application]
    gdbserver (21/04/2011 12:04) (Suspended)
        Thread [10] (Suspended)
            2 Ireport() C:\workspace\OVP\dhrystone.c:436 0x000205c4
            1 main() C:\workspace\OVP\dhrystone.c:609 0x00020844
    C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb.exe (21/04/2011 12:04)
    C:\workspace\OVP\dhrystone.ARM7.elf (21/04/2011 12:04)
```

```
Debug debugDemo [C/C++ Application] C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb.exe (21/04/2011 12:04)
1014-stack-select-frame 1
1014^done
(gdb)
1015-var-evaluate-expression var12
1015^done,value="0"
(gdb)
1016-var-update var14
1016^done,changelist=[]
(gdb)
1017-stack-select-frame 0
1017^done
(gdb)
1018-var-evaluate-expression var14
1018^done,value="0"
(gdb)
```

5.5.5 Terminating the Debug Session

Press the stop icon to stop debugging, then the “C/C++” perspective icon to return to the normal C project editing perspective.


Debug - debugDemo/dhrystone.c - Eclipse
File Edit Source Refactor Navigate Search Run Project Window Help
Debug
C/C++
Debug
Variables
Breakpoints
Registers
Modules
Name
Value
OVPsim_arm.c
Makefile
dhrystone.c
Outline
Console
Tasks
Problems
Executables
Memory
Launching Start OVPs...999: (57%)

```
Start OVPsim Platform wait port 9999 [Program]
    C:\workspace\OVP\OVPsim_arm.Windows.exe
Debug debugDemo [C/C++ Application]
    gdbserver (21/04/2011 11:38) (Suspended)
        Thread [2] (Suspended: Breakpoint hit.)
            1 main() C:\workspace\OVP\dhrystone.c:543 0x000206d8
    C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb.exe (21/04/2011 11:38
    C:\workspace\OVP\dhrystone.ARM7.elf (21/04/2011 11:38)
```

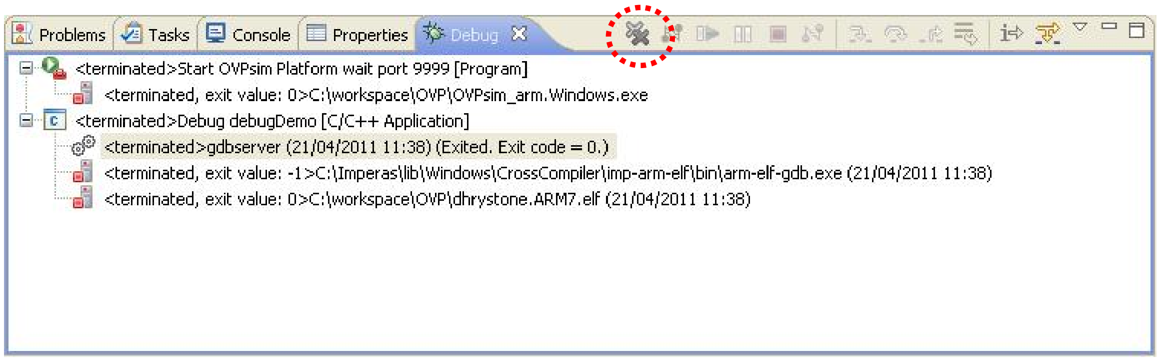
Name	Value
Main	
r0	0
r1	0
r2	0

```
  /* main program, corresponds to procedures        */
  /* Main and Proc_0 in the Ada version             */
{
        One_Fifty       Int_1_Loc;
  REG   One_Fifty       Int_2_Loc=0;
        One_Fifty       Int_3_Loc;
  REG   char            Ch_Index;
        Enumeration     Enum_Loc;
        Str_30          Str_1_Loc;
        Str_30          Str_2_Loc;
  REG   int             Run_Index;
```

```
stdio.h
stdlib.h
Mic_secs_Per_Second
structassign
structassign
Ident_1
Ident_2
Ident_3
Ident_4
Ident_5
```

```
Debug debugDemo [C/C++ Application] C:\workspace\OVP\dhrystone.ARM7.elf (21/04/2011 11:38)
```

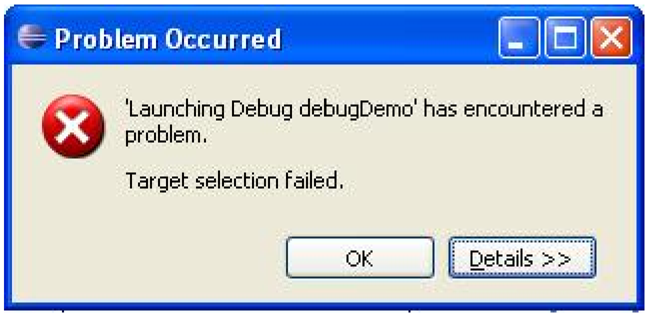
There will be a number of terminated processes shown in the debug session window that can be cleaned up using the icon shown.



```
<terminated>Start OVPsim Platform wait port 9999 [Program]
    <terminated, exit value: 0>C:\workspace\OVP\OVPsim_arm.Windows.exe
<terminated>Debug debugDemo [C/C++ Application]
    <terminated>gdbserver (21/04/2011 11:38) (Exited. Exit code = 0.)
    <terminated, exit value: -1>C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb.exe (21/04/2011 11:38)
    <terminated, exit value: 0>C:\workspace\OVP\dhrystone.ARM7.elf (21/04/2011 11:38)
```

If the debugger is unable to connect to the OVPsim platform you will see an error dialog.

.


Problem Occurred

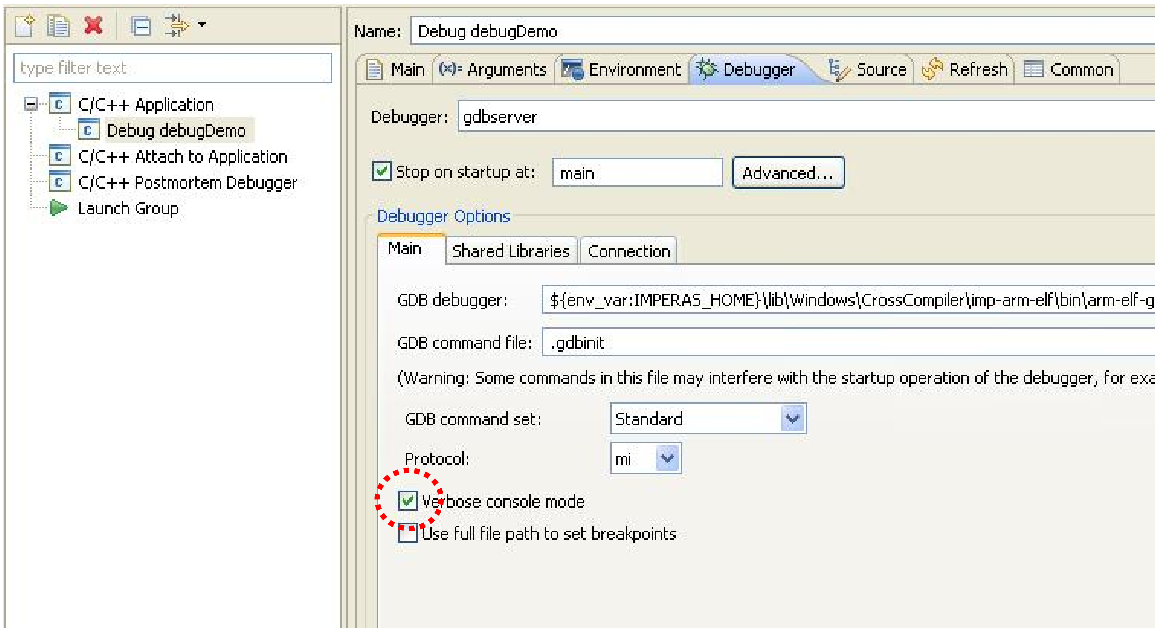
'Launching Debug debugDemo' has encountered a problem.

Target selection failed.

Check that the OVPsim platform is running, and that the port number it is waiting for (section 5.2.1) matches the port number in the debug configuration settings (section 5.5.1).

5.5.6 Verbose GDB Console Output

To enable further information from the interface between Eclipse and the GDB you may turn on a verbose output from the debugger tab of the GDB launcher.


Name: Debug debugDemo

Debugger: gdbserver

Stop on startup at: main

GDB debugger: ${env_var:IMPERAS_HOME}\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-g

GDB command file: .gdbinit

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for exa

GDB command set: Standard

Protocol: mi

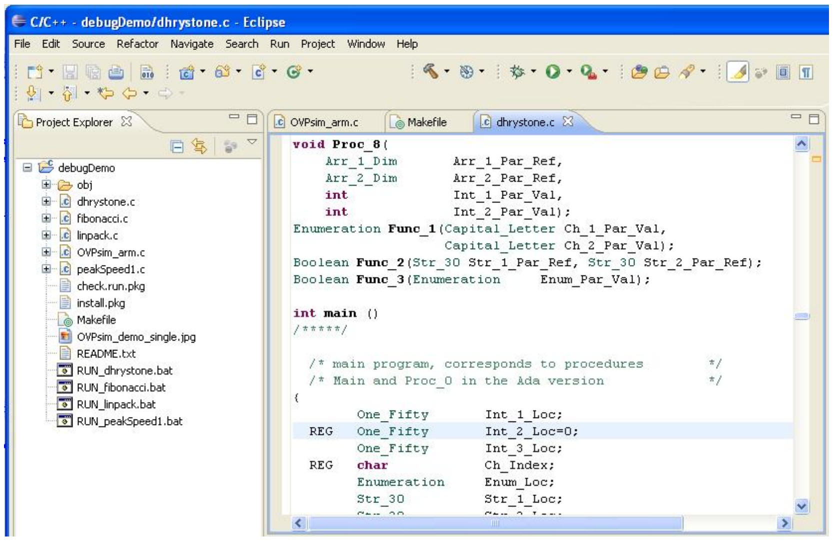
[✓] Verbose console mode

[ ] Use full file path to set breakpoints

5.6 *An Example Debug Session*

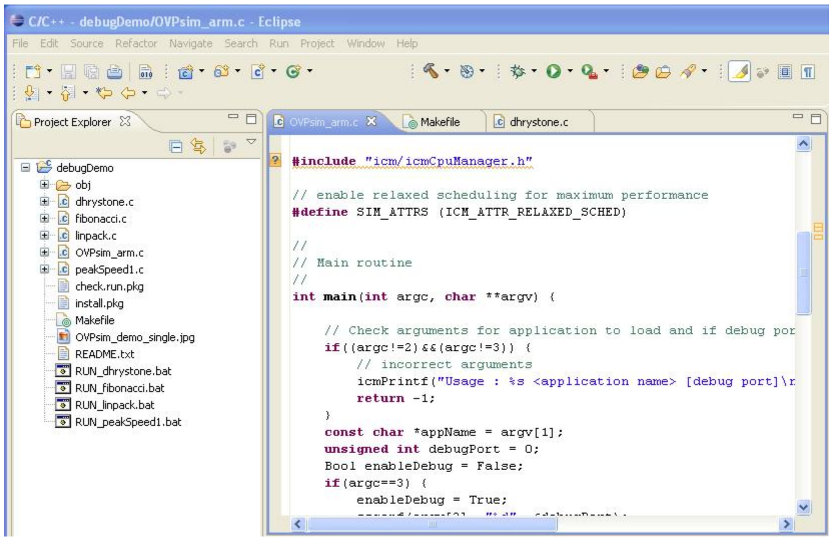
Section 5.3 showed how to prepare an Eclipse project to debug an application running on a processor in an OVPsim virtual platform. This section provides a quick overview of some common debugging tasks in Eclipse. For a full explanation of Eclipse C/C++ debugging please see the *C/C++ Development User Guide* available in the Eclipse help system or online at www.eclipse.org.

The application we will look at is the Dhrystone benchmark code.



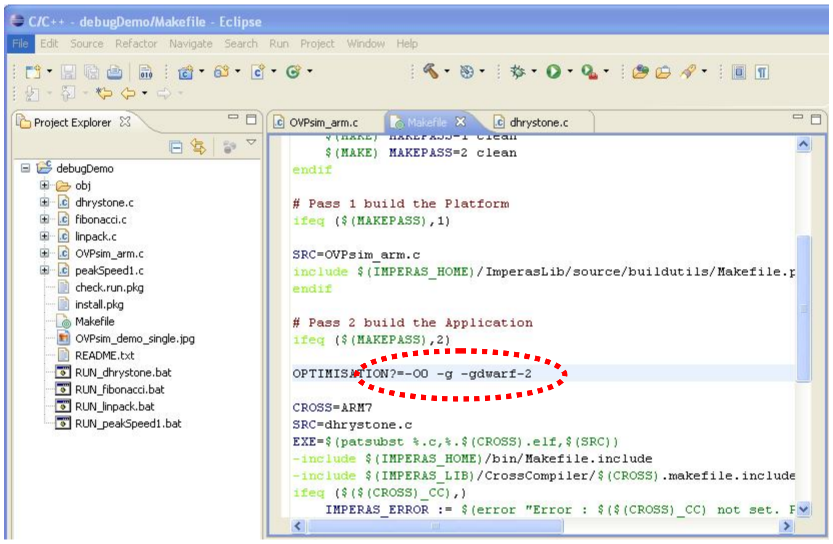
```
void Proc_8(
    Arr_1_Dim       Arr_1_Par_Ref,
    Arr_2_Dim       Arr_2_Par_Ref,
    int             Int_1_Par_Val,
    int             Int_2_Par_Val);
Enumeration Func_1(Capital_Letter Ch_1_Par_Val,
                   Capital_Letter Ch_2_Par_Val);
Boolean Func_2(Str_30 Str_1_Par_Ref, Str_30 Str_2_Par_Ref);
Boolean Func_3(Enumeration     Enum_Par_Val);

int main ()
/*****/

  /* main program, corresponds to procedures        */
  /* Main and Proc_0 in the Ada version             */
{
        One_Fifty       Int_1_Loc;
  REG   One_Fifty       Int_2_Loc=0;
        One_Fifty       Int_3_Loc;
  REG   char            Ch_Index;
        Enumeration     Enum_Loc;
        Str_30          Str_1_Loc;
```

With a single processor platform setup to receive a port number for opening the debug connection.



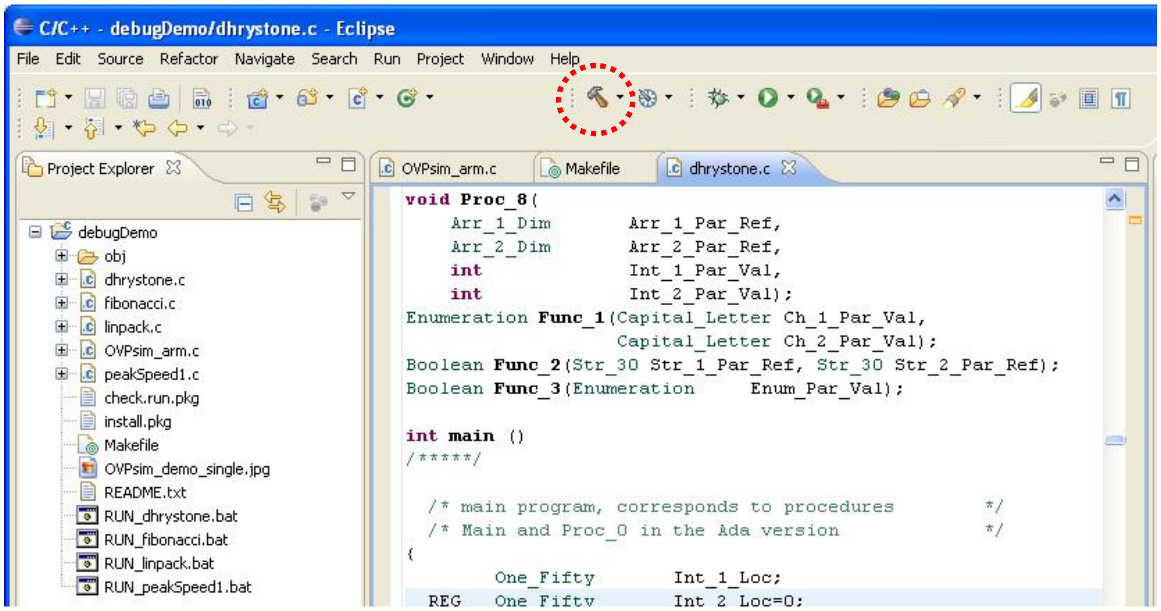
```
#include "icm/icmCpuManager.h"

// enable relaxed scheduling for maximum performance
#define SIM_ATTRS (ICM_ATTR_RELAXED_SCHED)

//
// Main routine
//
int main(int argc, char **argv) {

    // Check arguments for application to load and if debug por
    if((argc!=2)&&(argc!=3)) {
        // incorrect arguments
        icmPrintf("Usage : %s <application name> [debug port]\r
        return -1;
    }
    const char *appName = argv[1];
    unsigned int debugPort = 0;
    Bool enableDebug = False;
    if(argc==3) {
        enableDebug = True;
```

We also need to ensure that the external Makefile that is being used, is setup to build an application elf file containing debug information.



```
    $(MAKE) MAKEPASS=2 clean
endif

# Pass 1 build the Platform
ifeq ($(MAKEPASS),1)

SRC=OVPsim_arm.c
include $(IMPERAS_HOME)/ImperasLib/source/buildutils/Makefile.p
endif

# Pass 2 build the Application
ifeq ($(MAKEPASS),2)

OPTIMISATION?=-O0 -g -gdwarf-2

CROSS=ARM7
SRC=dhrystone.c
EXE=$(patsubst %.c,%.$(CROSS).elf,$(SRC))
-include $(IMPERAS_HOME)/bin/Makefile.include
-include $(IMPERAS_LIB)/CrossCompiler/$(CROSS).makefile.include
ifeq ($($(CROSS)_CC),)
    IMPERAS_ERROR := $(error "Error : $($(CROSS)_CC) not set. P
```

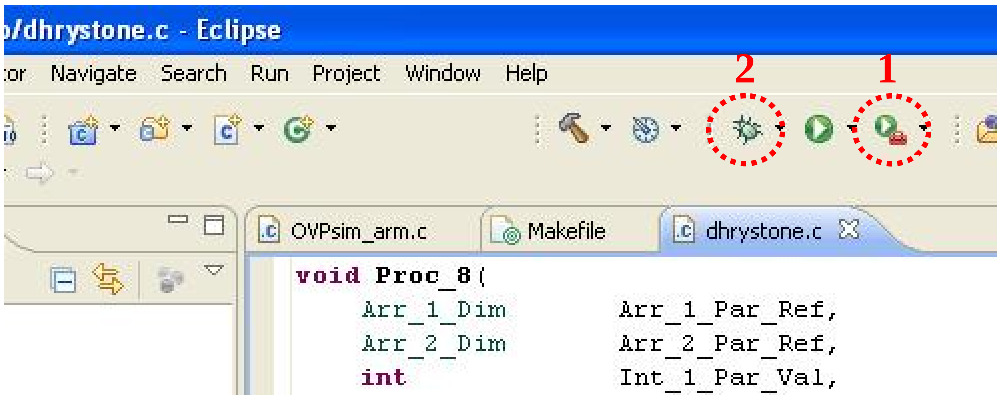
After any editing in the project of the application or platform, rebuild by pressing the build icon on the Eclipse toolbar. Any syntax errors will be reported in the console view and highlighted in the editor window.



```
void Proc_8(
    Arr_1_Dim       Arr_1_Par_Ref,
    Arr_2_Dim       Arr_2_Par_Ref,
    int             Int_1_Par_Val,
    int             Int_2_Par_Val);
Enumeration Func_1(Capital_Letter Ch_1_Par_Val,
                   Capital_Letter Ch_2_Par_Val);
Boolean Func_2(Str_30 Str_1_Par_Ref, Str_30 Str_2_Par_Ref);
Boolean Func_3(Enumeration     Enum_Par_Val);

int main ()
/*****/

  /* main program, corresponds to procedures        */
  /* Main and Proc_0 in the Ada version             */
{
        One_Fifty       Int_1_Loc;
  REG   One_Fifty       Int_2_Loc=0;
```

.

Start debugging by clicking the external tool toolbar button (1), then the debug toolbar button (2).

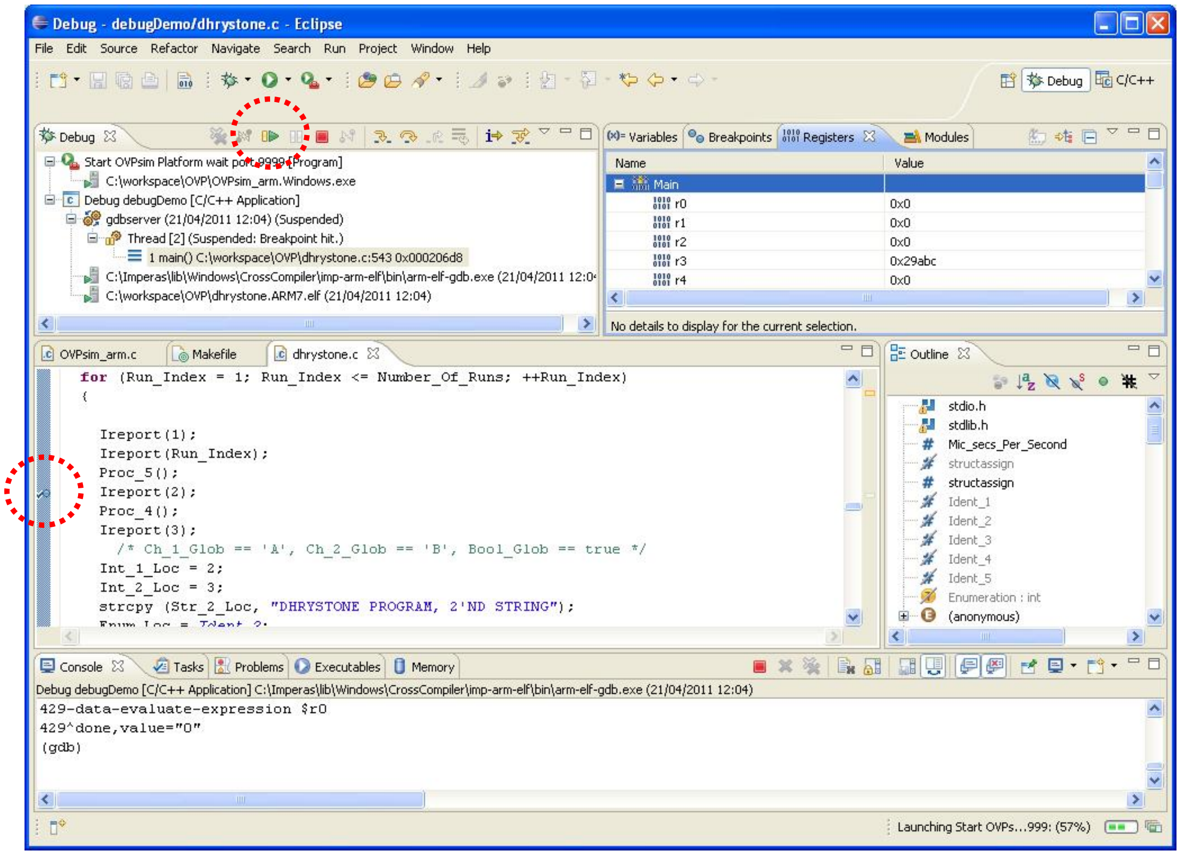


```
void Proc_8(
    Arr_1_Dim        Arr_1_Par_Ref,
    Arr_2_Dim        Arr_2_Par_Ref,
    int              Int_1_Par_Val,
```

Eclipse will switch to the debug perspective automatically.

Set a breakpoint on the call to `Ireport(2)` by double-clicking in the margin next to the call.

Run to the breakpoint by pressing the resume toolbar button.


Debug - debugDemo/dhrystone.c - Eclipse
File Edit Source Refactor Navigate Search Run Project Window Help
Debug C/C++
Debug
Variables Breakpoints Registers Modules
OVPsim_arm.c Makefile dhrystone.c
Outline
Console Tasks Problems Executables Memory

```
Start OVPsim Platform wait port 9999 [Program]
    C:\workspace\OVP\OVPsim_arm.Windows.exe
Debug debugDemo [C/C++ Application]
    gdbserver (21/04/2011 12:04) (Suspended)
        Thread [2] (Suspended: Breakpoint hit.)
            1 main() C:\workspace\OVP\dhrystone.c:543 0x000206d8
    C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb.exe (21/04/2011 12:0
    C:\workspace\OVP\dhrystone.ARM7.elf (21/04/2011 12:04)
```

Name	Value
Main	
r0	0x0
r1	0x0
r2	0x0
r3	0x29abc
r4	0x0

No details to display for the current selection.

```
  for (Run_Index = 1; Run_Index <= Number_Of_Runs; ++Run_Index)
  {

    Ireport(1);
    Ireport(Run_Index);
    Proc_5();
    Ireport(2);
    Proc_4();
    Ireport(3);
      /* Ch_1_Glob == 'A', Ch_2_Glob == 'B', Bool_Glob == true */
    Int_1_Loc = 2;
    Int_2_Loc = 3;
    strcpy (Str_2_Loc, "DHRYSTONE PROGRAM, 2'ND STRING");
```

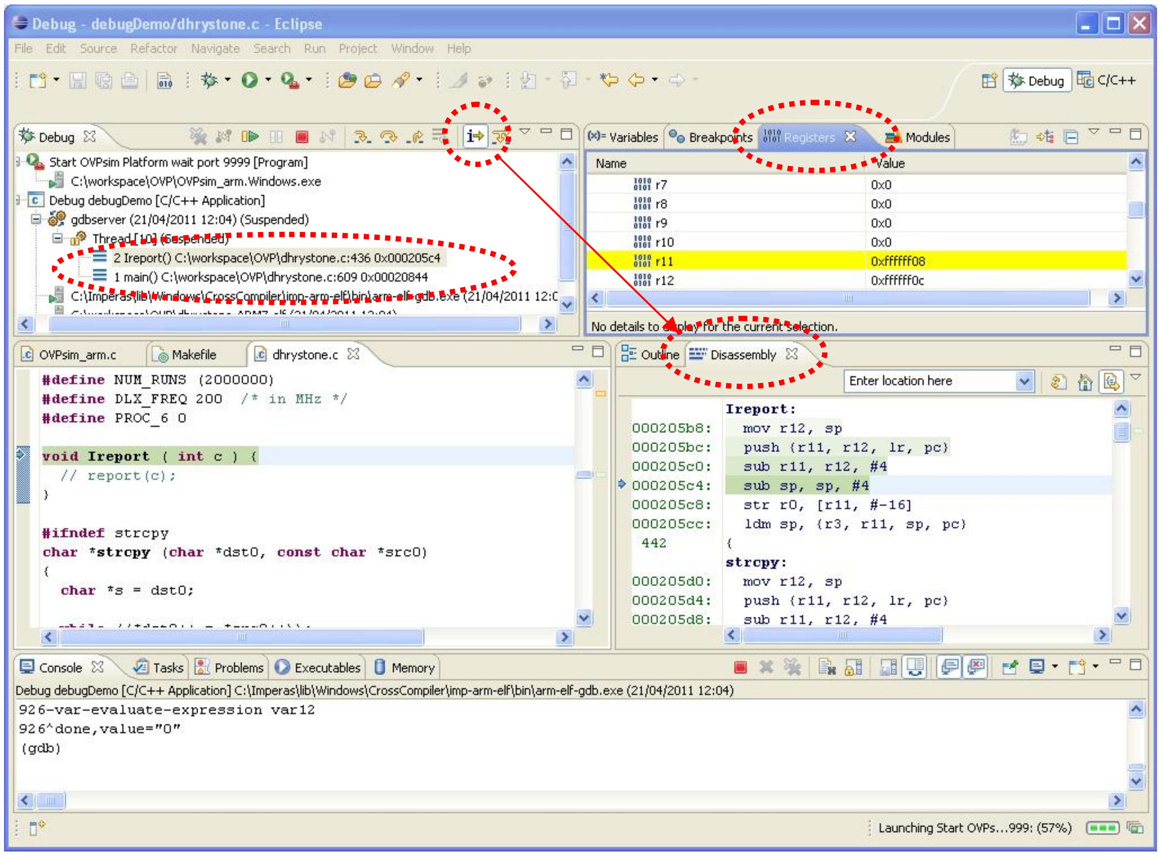
```
stdio.h
stdlib.h
Mic_secs_Per_Second
structassign
structassign
Ident_1
Ident_2
Ident_3
Ident_4
Ident_5
Enumeration : int
(anonymous)
```

Debug debugDemo [C/C++ Application] C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb.exe (21/04/2011 12:04)

```
429-data-evaluate-expression $r0
429^done,value="0"
(gdb)
```

Launching Start OVPs...999: (57%)

Show the processor registers by switching to the registers pane. Bring up the disassembly view by clicking the instruction stepping mode toolbar button. The debug view shows the call stack for the debugged application.



```
Start OVPsim Platform wait port 9999 [Program]
    C:\workspace\OVP\OVPsim_arm.Windows.exe
Debug debugDemo [C/C++ Application]
    gdbserver (21/04/2011 12:04) (Suspended)
        Thread [10] (Suspended)
            2 Ireport() C:\workspace\OVP\dhrystone.c:436 0x000205c4
            1 main() C:\workspace\OVP\dhrystone.c:609 0x00020844
    C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb.exe (21/04/2011 12:0
```

Name	Value
r7	0x0
r8	0x0
r9	0x0
r10	0x0
r11	0xffffff08
r12	0xffffff0c

No details to display for the current selection.

```
#define NUM_RUNS (2000000)
#define DLX_FREQ 200  /* in MHz */
#define PROC_6 0

void Ireport ( int c ) {
  // report(c);
}

#ifndef strcpy
char *strcpy (char *dst0, const char *src0)
{
  char *s = dst0;
```

```
          Ireport:
000205b8:   mov r12, sp
000205bc:   push {r11, r12, lr, pc}
000205c0:   sub r11, r12, #4
000205c4:   sub sp, sp, #4
000205c8:   str r0, [r11, #-16]
000205cc:   ldm sp, {r3, r11, sp, pc}
 442      {
          strcpy:
000205d0:   mov r12, sp
000205d4:   push {r11, r12, lr, pc}
000205d8:   sub r11, r12, #4
```

Debug debugDemo [C/C++ Application] C:\Imperas\lib\Windows\CrossCompiler\imp-arm-elf\bin\arm-elf-gdb.exe (21/04/2011 12:04)

```
926-var-evaluate-expression var12
926^done,value="0"
(gdb)
```

Launching Start OVPs...999: (57%)

As you step, you will see the registers that are modified being highlighted.

Finally, run to completion by double clicking on the breakpoint to remove it, then pressing the resume button again. Program output is shown in the Eclipse platform console view.

###

.