EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

2390 EMAC, Way
Carbondale, IL 62902
Phone: (618) 529-4525
Fax: (618) 457-0110
http://www.emacinc.com

Debugging Applications on EMAC ARM Open-Embedded

This document describes the process of debugging applications on EMAC's ARM Open Embedded build version 1.2 and higher using EMAC's Eclipse distribution version 3.3 and GDB. This release of EMAC OE uses the dropbear SSH server, which is not capable of using the Remote Services Explorer automated download and execute feature in Eclipse. Future versions of EMAC OE will include an OpenSSH server with SFTP enabled, allowing this functionality to work.

To begin, a valid executable will be needed for debugging. Also insure that the -g option is passed to GCC in the Makefile to ensure that debugging symbols are built into the executable. For demonstration the "hello" example project for the IPAC-9302 was modified as follows:

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/*
 *hello world
 */
int main()
{
        int i = 0;

   printf("hello IPAC9302\n");
   printf("This is a test\n");
   printf("Try it out!!\n");
   if (i)
        printf("i is true\n");
   else
        printf("i is false\n");
   return 0;
}
```

Listing 1: Modified Hello.c Code

After compiling the hello project, it will need to be transferred to the board for testing. To do this, first modify the TARGET_IP variable in the Makefile to match the IP address of the board. Next, use the upload make target to transfer the file to the board via FTP. By default, the executable will be transferred to /tmp on the board.

Next, open a terminal connection to the board either through a separate console or through the Terminal view in Eclipse. To access the terminal view through Eclipse, go to Window-> Show View-> Other-> Terminal-> Terminal. This will open a new view in the bottom of the window called terminal. Initialize a new SSH connection as shown below by clicking on the green connection settings button on the terminal view. Note that this process is described in detail in the "Linux Development Using Eclipse" manual.

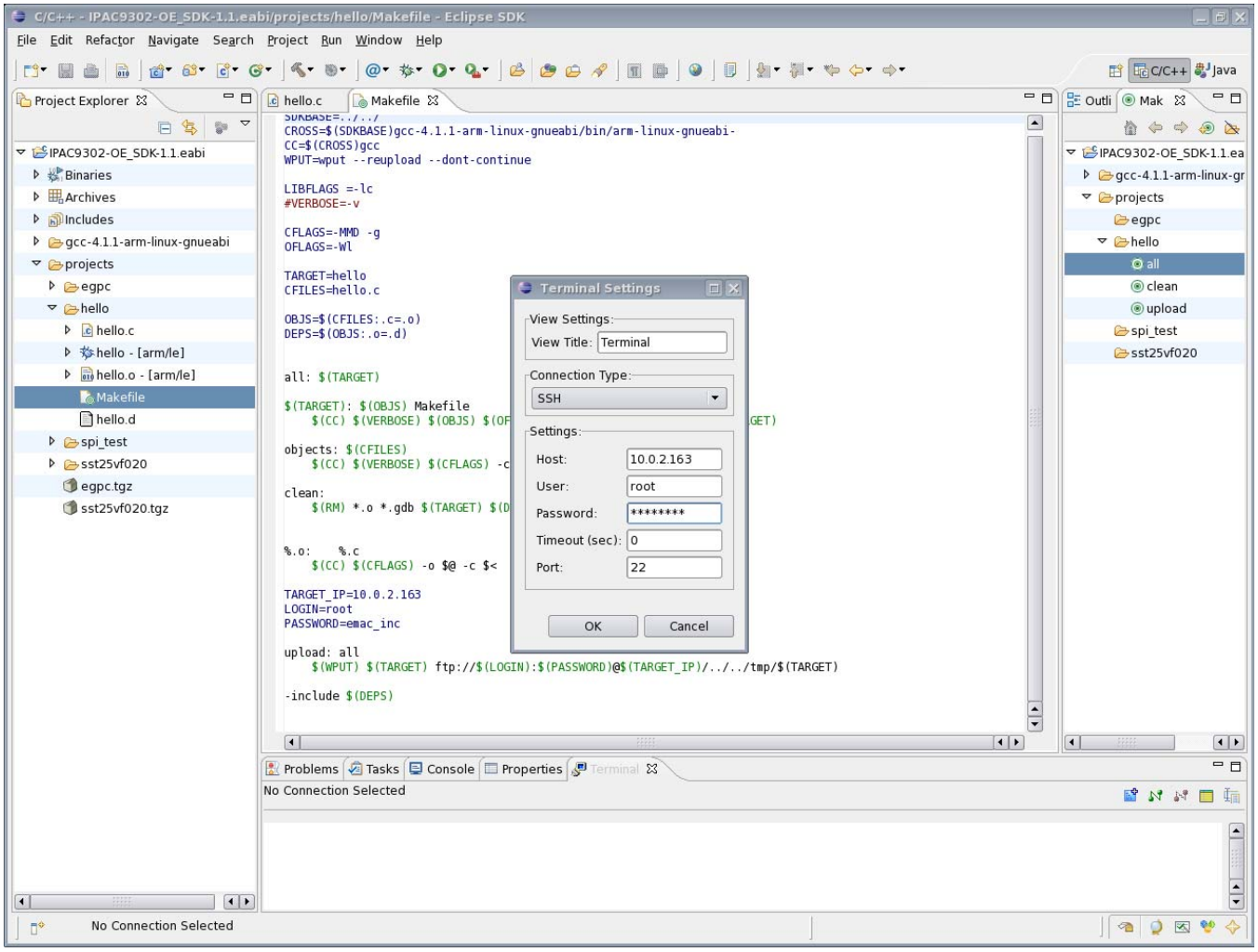


Figure 1: New Terminal Connection

Next, run the application to test that it is working correctly. Use the following commands to navigate to /tmp, make the application executable, and run the application:

```
root@IPAC9302:~# cd /tmp
root@IPAC9302:/var/volatile/tmp# chmod +x hello
root@IPAC9302:/var/volatile/tmp# ./hello
hello IPAC9302
This is a test
Try it out!!
i is false
```

Listing 2: Commands for Executing Application

After testing the application, start GDBserver on the board as shown below. Note that this is described in detail in the “Linux Development Using Eclipse” manual. The example below starts GDBserver listening on port 2828 from any IP address.

```
root@IPAC9302:/var/volatile/tmp# gdbserver :2828 ./hello
Process ./hello created; pid = 1043
Listening on port 2828
```

Listing 3: Starting GDBServer

Next, use Eclipse to connect to the running GDBserver application and debug the application.

1. Right click on the hello binary file in the Project Explorer view and select Debug As-> Open Debug Dialog...
2. Select C/C++ Local Application on the left and press the New button to create a new configuration.
3. On the Debugger tab, change Debugger to gdbserver Debugger.
4. Under Debugger Options, change the GDB debugger to the arm-linux-gnueabi-gdb application in the SDK. This is located under the gcc-<version>/bin directory in the SDK.

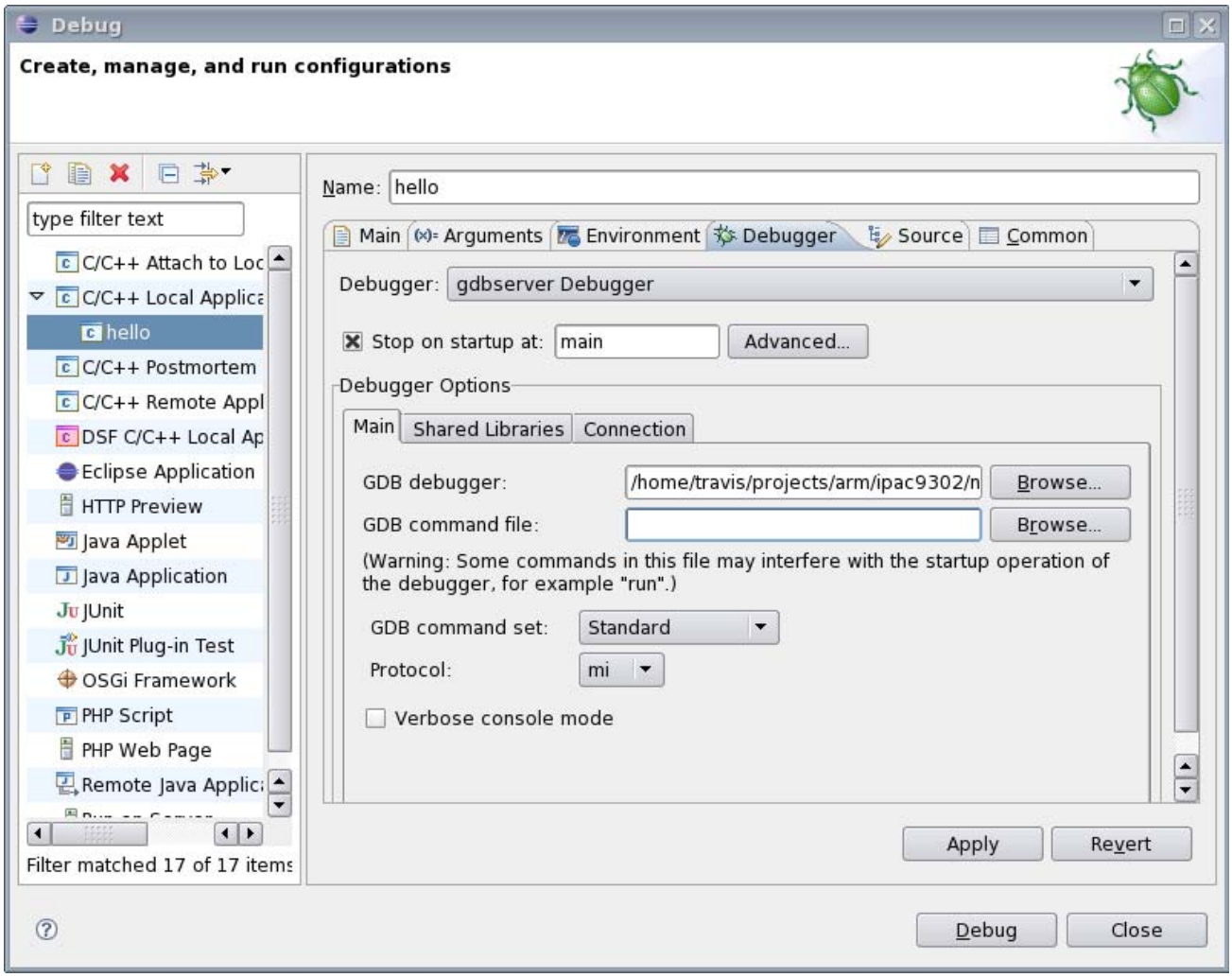


Figure 2: Debug Configuration Settings

5. Add a search path for the shared libraries in the SDK as shown below. This is located in the gcc-<version>/arm-linux-gnueabi/lib directory of the SDK.

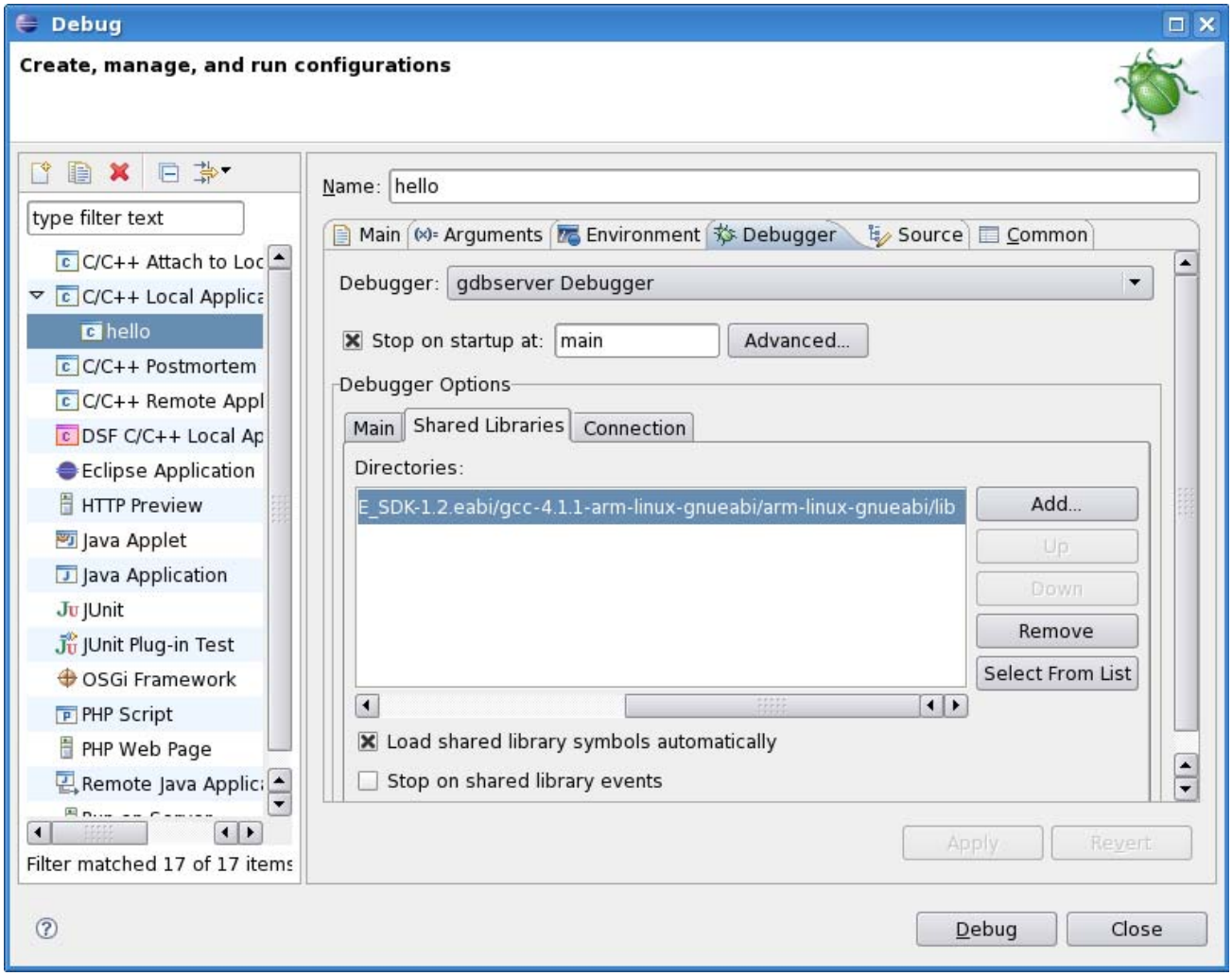


Figure 3: Shared Library Search Path

6. Set the Connection settings to match those of the board. Set the connection type to TCP and enter the IP address of the board and the port number of the running GDBserver as shown below.

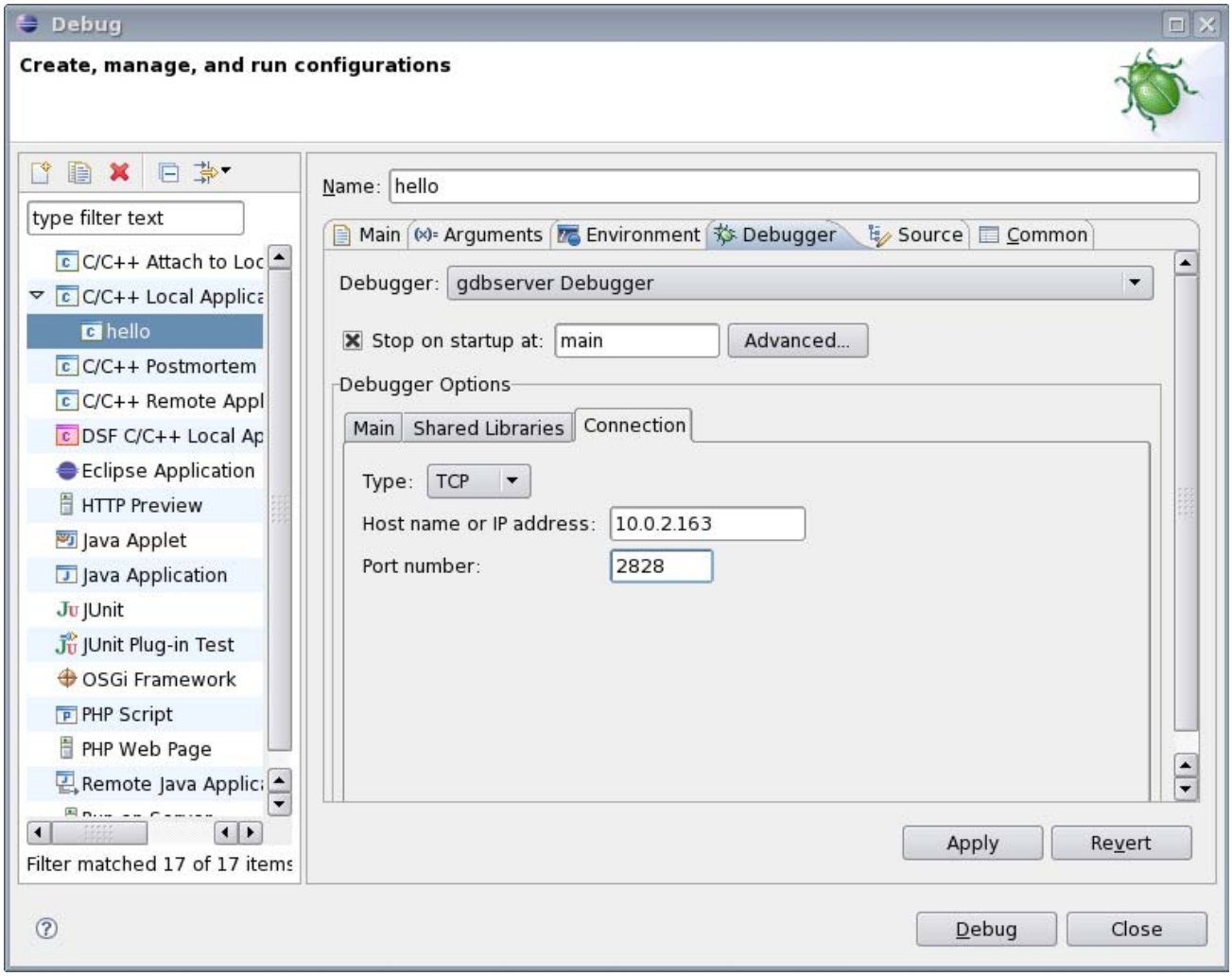
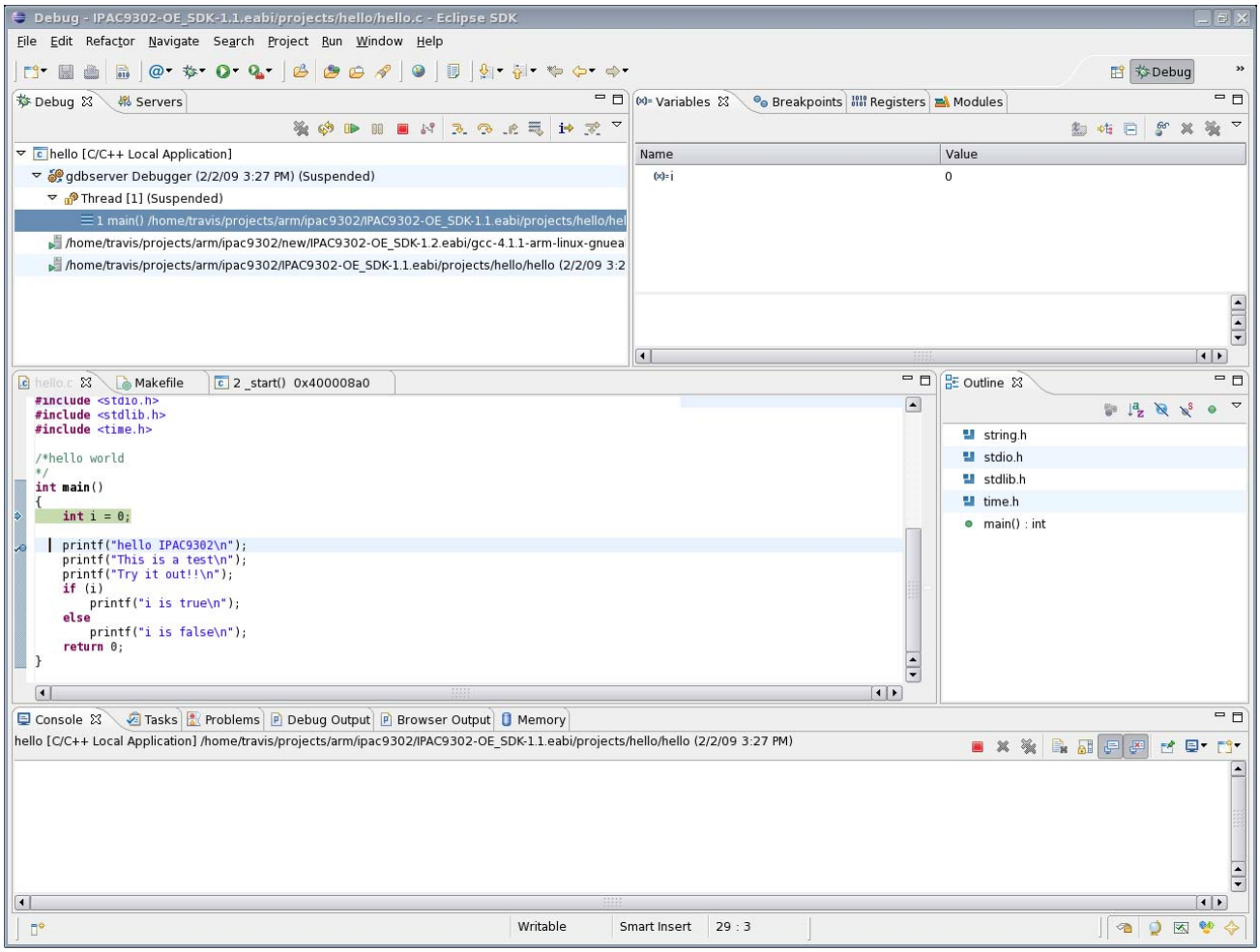


Figure 4: GDB Connection Settings

7. Click on Apply followed by Debug.
8. When prompted to switch to the Debug perspective, select Remember my decision and click Yes.
9. Eclipse will automatically open the source, highlighting the current position in the code. The different views in the Eclipse debug perspective provide many different tools for debugging and examining the application.
10. Set a breakpoint at a certain line in the code by double-clicking on the margin to the left of the line. A breakpoint is set on the first printf() call in the example below.



11. Press the “Resume” button on the Debug view at the top of the Window to continue to the next breakpoint. The instruction stepping mode can be used to step through the code instruction by instruction and display the application disassembly. Use the “Step” buttons to single step through the application.
12. Switch back to the C/C++ perspective using the arrows at the top right of the window.