Development Environment for the i.MX6

Application Note: AN105

December 2, 2019

Cimarron Systems, LLC

Table of Contents

List of Figures

Document**:** AN-191202-1
Document Status: Released
Revision: 3.0

Overview

The present application note describes the steps necessary to get started developing 'C' language software applications for the NXP®/Freescale i.MX6 platform from both the command line and utilizing the Eclipse C/C++ IDE. For our target, we will use a Toradex Apalis i.MX6 Computer on Module (CoM) installed on the company's Ixora Carrier Board.

i.MX6 Development Environment Context

Figure 1 shows a context diagram of the typical environment for application development using the i.MX6 platform, which includes: the Ubuntu Linux Host with keyboard, mouse, and display and an i.MX6 Apalis / Ixora with keyboard, mouse, display, and associated cabling.

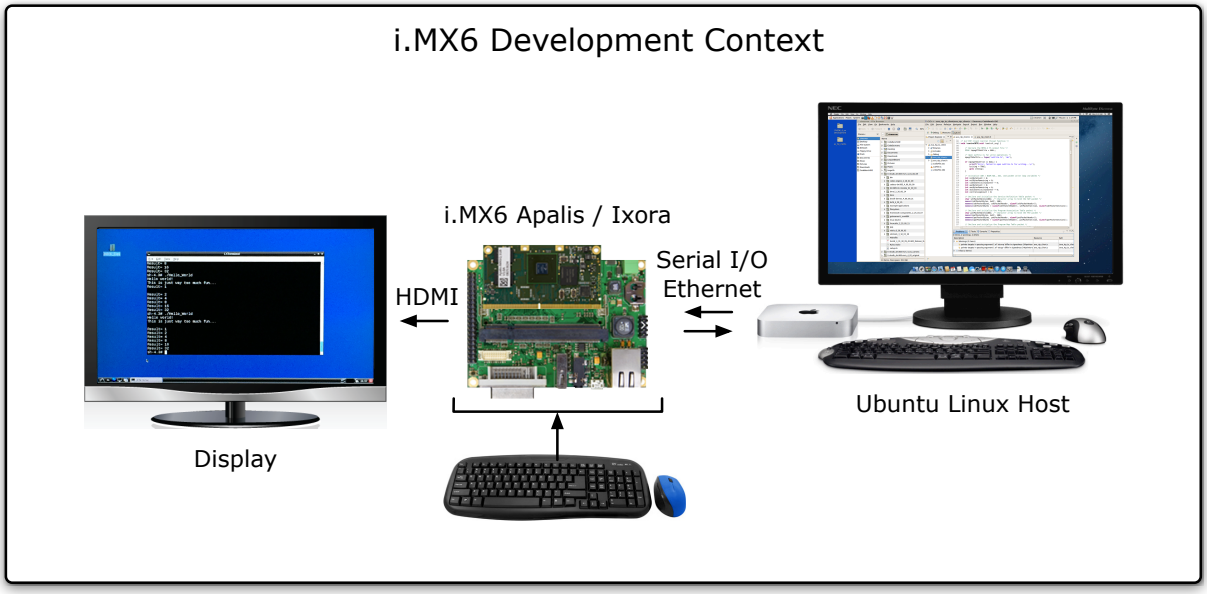

Figure 1: Development context for the i.MX6 platform.

Developing with the Command-line

The following steps will guide you through the set up process for developing using the Toradex Apalis i.MX6 CoM from the `bash` command-line (for more information, please see Toradex Apalis ARM Family).

Prerequisites: Your Host Computer (or VM) should be running the 64-bit version of Ubuntu Linux 16.04 LTS or later.

Step 1a: Download the Toradex SDK, found here, into the development directory of your choice. The SDK includes the cross development tool chain needed to build applications directly on your Linux Host as well the i.MX6 Apalis target root filesystem with necessary headers.

Step 1b: Open a terminal window, connect to your development directory, and install the SDK by executing the installer script as follows (this installs the SDK in the default location):

```
user@host:~$ ./angstrom-glibc-x86_64-armv7at2hf-neon-v2016.12-toolchain.sh
Angstrom SDK installer version nodistro.0
==========================================
Enter target directory for SDK (default: /usr/local/oecore-x86_64):
You are about to install the SDK to "/usr/local/oecore-x86_64". Proceed[Y/n]?
```


Evergreen CO 80439 • +1 (303) 674-9207

Step 1c: Source the development environment variables:

```
user@host:~$ . /usr/local/oecore-x86_64/environment-setup-armv7at2hf-neon-angstrom-linux-gnueabi
```

Note: You must execute this, Step 1c, each time you open a new terminal tab / window for application cross compilation.

Step 1d: Create a file named 'Hello_World.c' with the following contents:

```
#include <stdio.h>

int main(void){
    printf("Hello world!\n");
    return 0;
}
```

Step 1e: Cross compile 'Hello_World.c' by executing the following in the command-line:

```
user@host:~$ ${CC} -Wall Hello_World.c -o Hello_World
```

Step 1f: Determine the 'IP Address' of your target system by executing the following from the command-line:

```
root@apalis-imx6:~# ifconfig

eth0      Link encap:Ethernet  HWaddr 00:14:2D:49:E8:0A
          inet addr:192.168.2.3  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::214:2dff:fe49:e80a%lo/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:40238 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34742 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40585812 (38.7 MiB)  TX bytes:4115241 (3.9 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1%1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:3393 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3393 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:309829 (302.5 KiB)  TX bytes:309829 (302.5 KiB)

usb0      Link encap:Ethernet  HWaddr 00:14:2D:FF:FF:FF
          inet addr:192.168.11.1  Bcast:192.168.11.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Note: In the case shown above, the target system 'IP Address' of 'eth0' is: 'inet addr:192.168.2.3'.

Evergreen CO 80439 • +1 (303) 674-9207

Step 1g: From your host computer, copy the application to the target using the `'scp'` command. You may be prompted to continue during the first connection—if this is the case—just type `'yes'` and then `'enter'` (please see below).

```
user@host:~$ scp hello-world root@192.168.2.3:/home/root
The authenticity of host '192.168.2.3 (192.168.2.3)' can't be established.
ECDSA key fingerprint is SHA256:+Lf4hEVgmHPOR2pZgJRdrgN5WCsnEtQXEnYWyMXxswZ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.3' (ECDSA) to the list of known hosts.
```

Step 1h: From a terminal on the i.MX6 Apalis target, run the 'Hello_World' application.

```
root@apalis-imx6:~# ./Hello_World
Hello world!
```

Developing with the Eclipse C/C++ IDE

The following steps will guide you through the setup process for developing for the i.MX6 target platform using the Eclipse C/C++ IDE. Additionally, we will develop a 'Hello_World' application on our Linux Host, transfer it to the i.MX6 Apalis target for modification / debugging there, and run the final 'Hello_World' application.

Prerequisites: Your Host Computer (or VM) should be running the 64-bit version of Ubuntu Linux 14.04 LTS or later and, you must have executed Steps 1a — 1c above.

Step 2a: — DOWNLOAD AND INSTALL THE ECLIPSE C/C++ IDE — Download the Linux 64-bit version of the Eclipse IDE for C/C++ Developers at: https://www.eclipse.org/downloads/packages/release/Kepler/SR2. The downloaded package name will be: 'eclipse-cpp-kepler-SR2-linux-gtk-x86_64.tar.gz'.

Step 2b: In order to ensure that you have the latest Java JRE installed, execute the follow command from the terminal:

```
user@host:~$ sudo apt-get update
user@host:~$ sudo apt-get install default-jre
```

Step 2c: Install the Eclipse C/C++ IDE into a folder of your choice, for example, '../eclipse/', source the development environment variables, then start the IDE.

```
user@host:~/eclipse$ . /usr/local/oecore-x86_64/environment-setup-armv7at2hf-neon-
angstrom-linux-gnueabi
user@host:~/eclipse$ ./eclipse
```

Note: You must execute this, Step 2c, each time you open a new terminal tab / window for startup of the Eclipse IDE.

Step 2d: Once the Eclipse C/C++ IDE starts up, you will be asked to create a workspace directory. Create a directory in a location of your choice; for example, '../eclipse/workspace/'—all of your application development work product will be stored in this location.

Step 2e: — CREATE AN EMPTY 'C' PROJECT — First, create an 'empty' 'C' project, for example, a project named 'Hello_World' with the 'Cross GCC' option selected then click the 'Next >' button (please see the screenshot below).

Evergreen CO 80439 • +1 (303) 674-9207

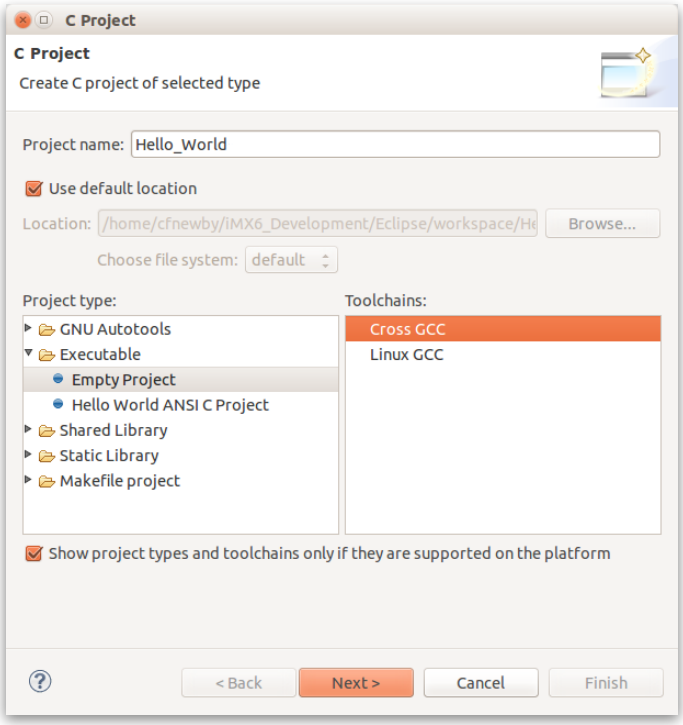

Figure 2: Create the 'Hello_World' project, part 1.

Step 2f: Next, select both the 'Debug' and 'Release' options then select the 'Next >' button (please see the screenshot below).

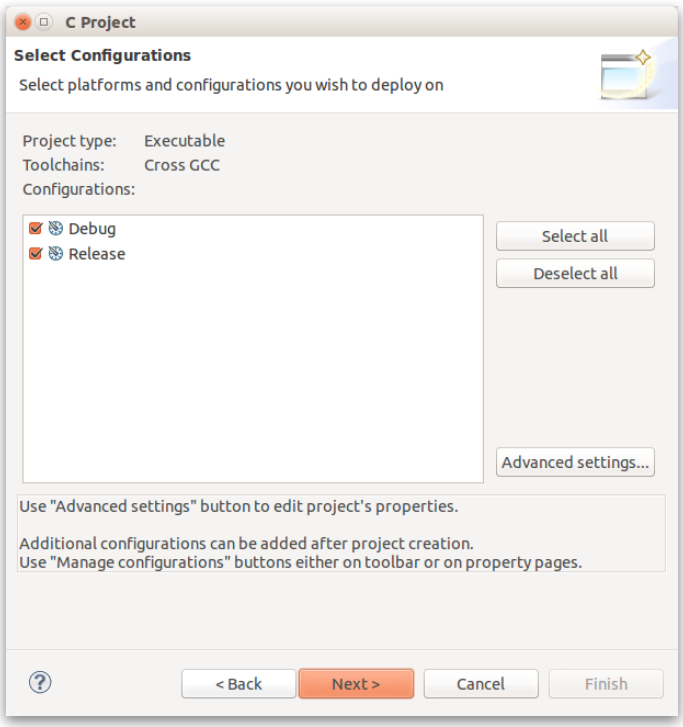

Figure 3: Create the 'Hello_World' project, part 2.

Step 2g: Finally, leave both the 'Prefix' and 'Path' lines blank then select the 'OK' or 'Finish' button (please

Evergreen CO 80439 • +1 (303) 674-9207

see the screenshot below).

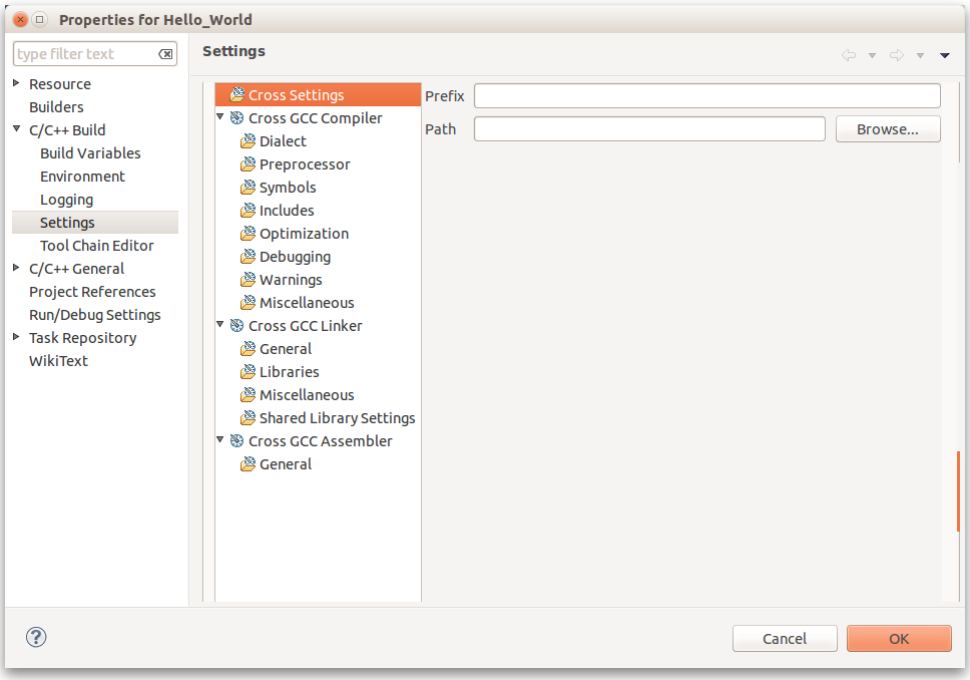

Figure 4: Create the 'Hello_World' project, part 3.

Step 2h: — SET THE 'HELLO_WORLD' PROJECT BUILD PARAMETERS — First, in order to set the project's compiler, linker, and assembler build parameters, from the 'Open Project' menu select 'Properties' (please see the screenshot below).

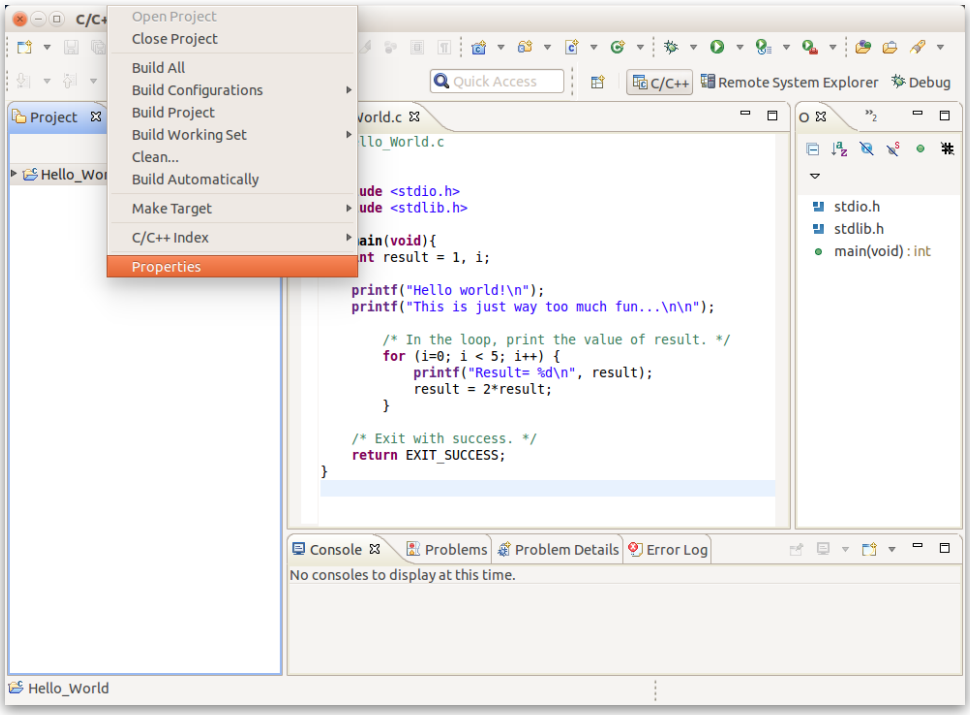

Figure 5: Setup the project build properties.

Evergreen CO 80439 • +1 (303) 674-9207

Step 2i: Next, select the 'C/C++ Build', then select 'Settings', then select 'Cross GCC Compiler', and finally, in the 'Command' box, enter '${CC}' (please see the screenshot below).

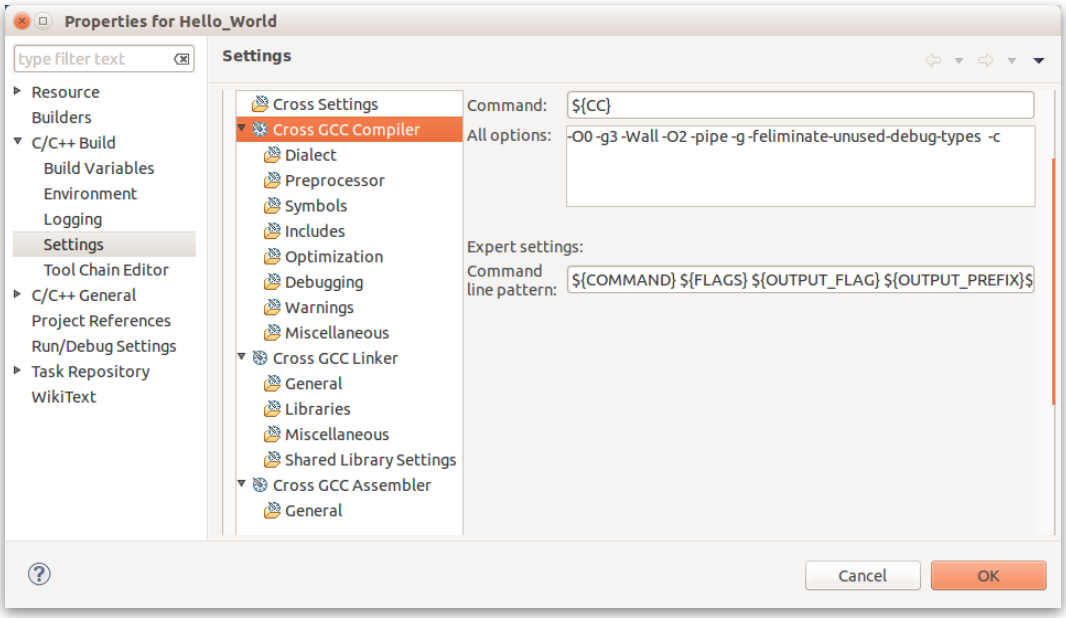

Figure 6: Setup the command variable to invoke the Cross Compiler.

Step 2j: Next, select the 'C/C++ Build', then select 'Settings', then select 'Miscellaneous', and finally, in the 'Other flags' box, enter '${CFLAGS} -c' (please see the screenshot below).

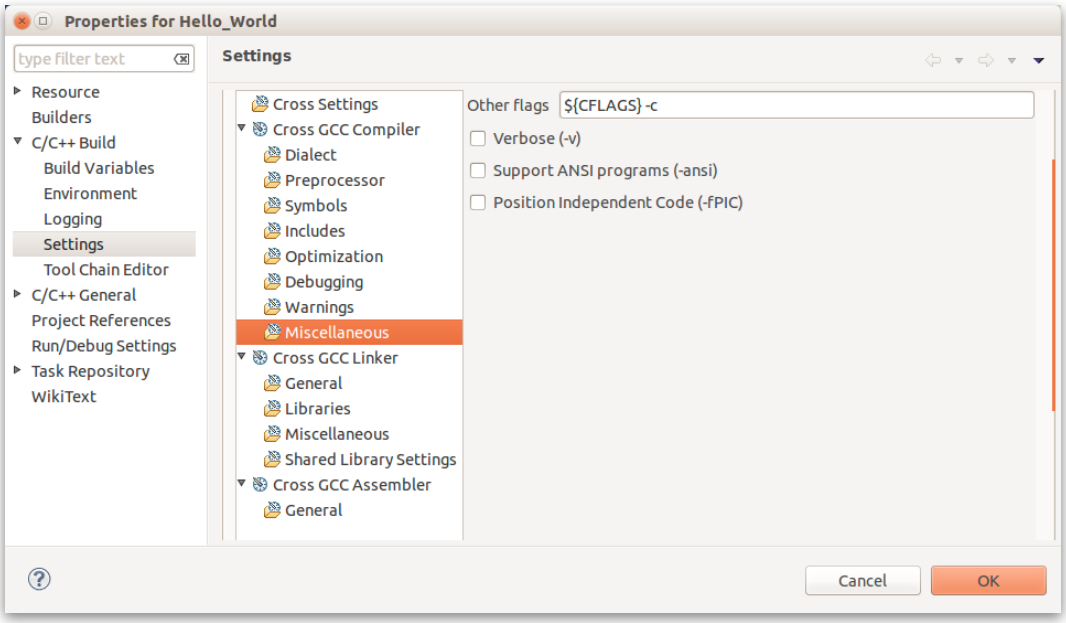

Figure 7: Setup the 'Other flags' to use variables from the Cross Compiler environment.

Evergreen CO 80439 • +1 (303) 674-9207

Step 2k: Next, select the 'C/C++ Build', then select 'Settings', then select 'Cross GCC Linker', and finally, in the 'Other flags' box, enter '${CXX}' (please see the screenshot below).

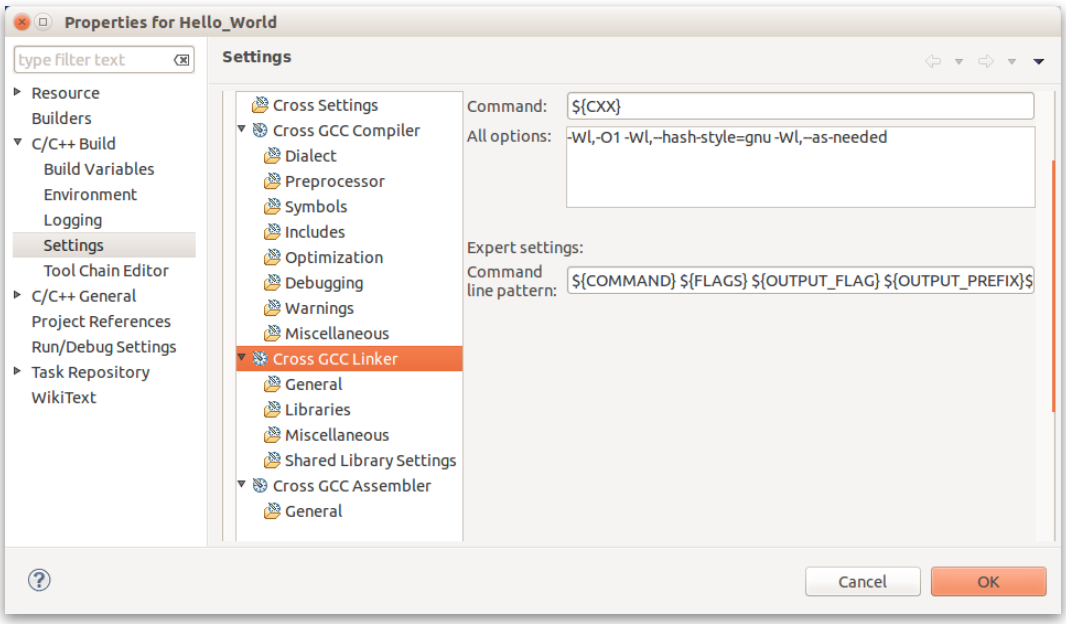

Figure 8: Setup the command variable to invoke the Linker.

Step 2l: Next, select the 'C/C++ Build', then select 'Settings', then select 'Cross GCC Linker', then select 'Miscellaneous', and finally, in the 'Linker flags' box, enter '${LDFLAGS}' (please see the screenshot below).

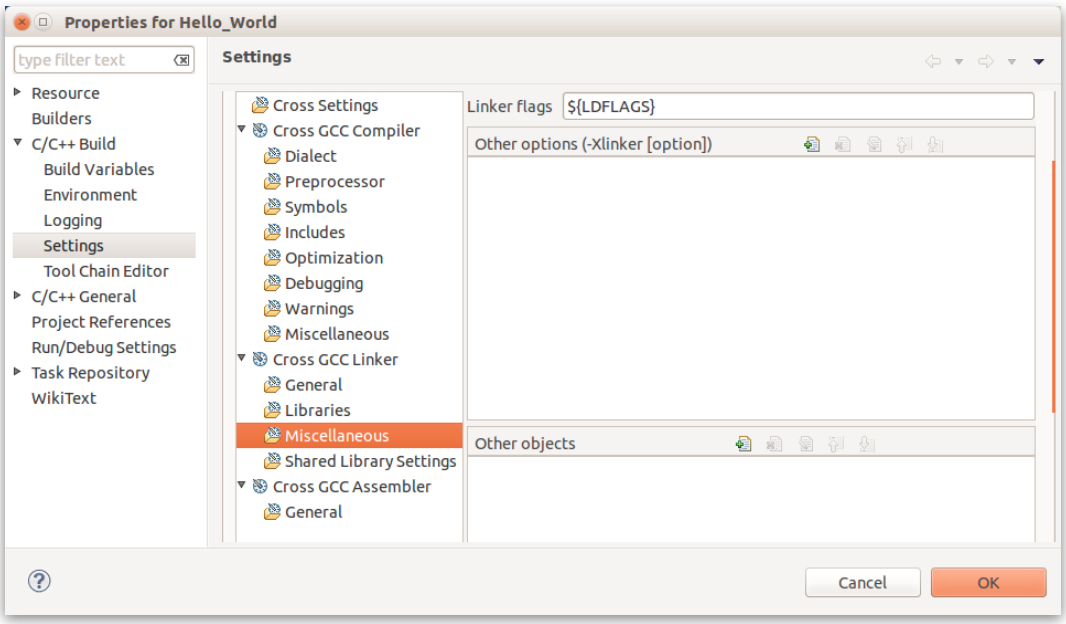

Figure 9: Setup the 'Linker flags' to use variables from the Cross Linker environment.

Evergreen CO 80439 • +1 (303) 674-9207

Step 2m: Next, select the 'C/C++ Build', then select 'Settings', then select 'Cross GCC Assembler', then in the 'Command' box, enter '${AS}', and finally, select the 'OK' button (please see the screenshot below).

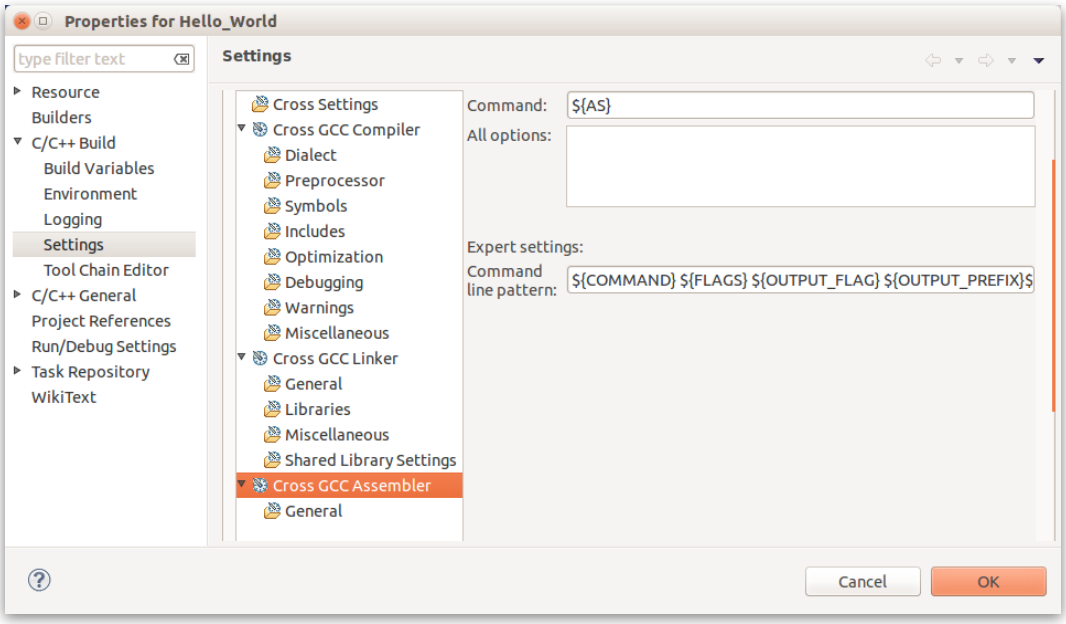

Figure 10: Setup the command variable to invoke the Assembler.

Step 2n: Finally, select the build the 'Hello_World'—you should see something similar to the screenshot below.

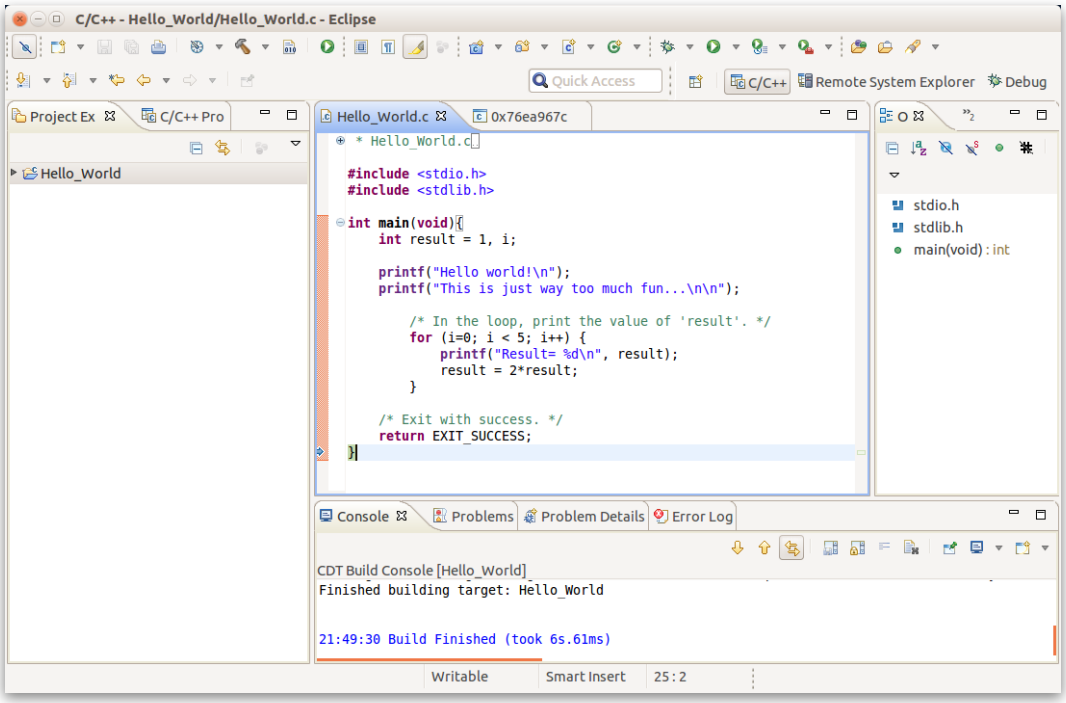

Figure 11: View after building the 'Hello_World' project.

Evergreen CO 80439 • +1 (303) 674-9207

Debugging with the Eclipse C/C++ IDE

The following steps will guide you through the setup process for using the using the Eclipse C/C++ IDE to remotely debug an application from the Ubuntu host while it runs on the i.MX6 target platform.

Prerequisites: Your Host Computer (or VM) should be running the 64-bit version of Ubuntu Linux 14.04 LTS or later and, you must have executed **Steps 2a — 2n** above.

Step 3a: — SETUP A REMOTE CONNECTION — First, open the **'Open Perspective'** list then, from the list, select **'Remote System Explorer'** (please see the screenshot below).

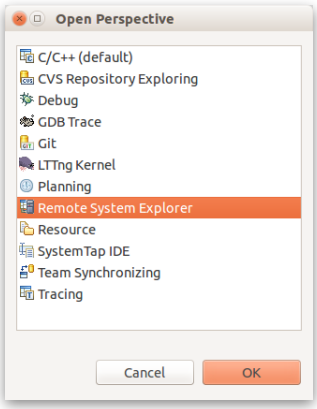

Figure 12: From the 'Open Perspective' list select 'Remote System Explorer'.

Step 3b: Next, on the left side in the **'Remote System'** tab, select the **'Define a connection to remote system'** button (please see the screenshot below).

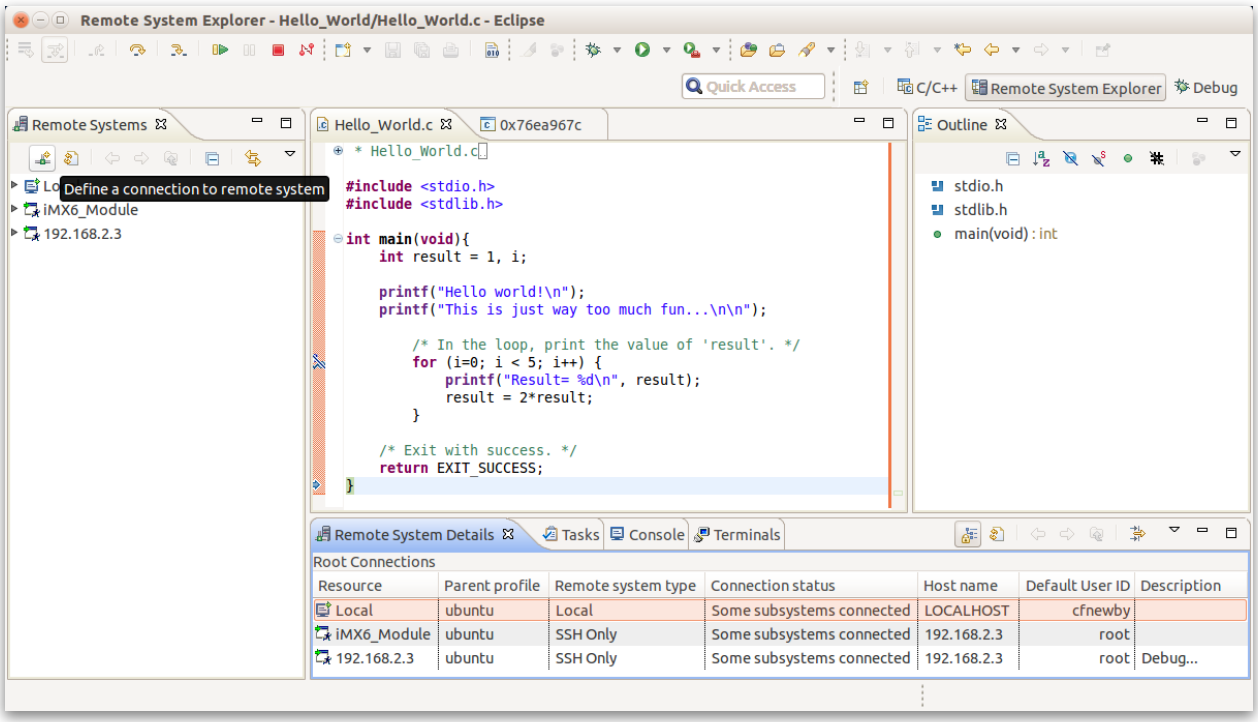

Figure 13: Select the 'Define a connection to remote system' button.

Evergreen CO 80439 • +1 (303) 674-9207

Step 3c: Next, from the 'Select Remote System Type' dialog window, select the 'SSH Only' (please see the screenshot below).

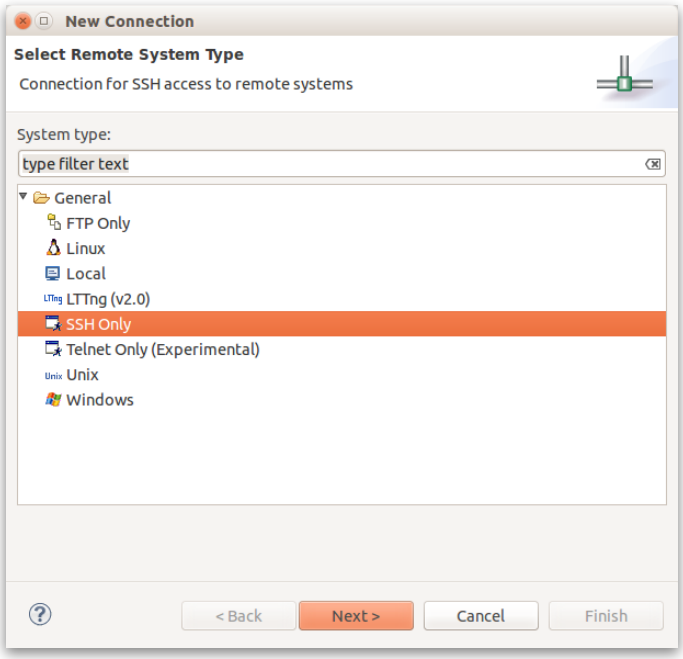

Figure 14: From the 'New Connection' dialog window, select 'SSH Only'.

Step 3d: Next, from the 'Remote SSH Only System Connection' dialog window, enter the appropriate information (please see the screenshot below).

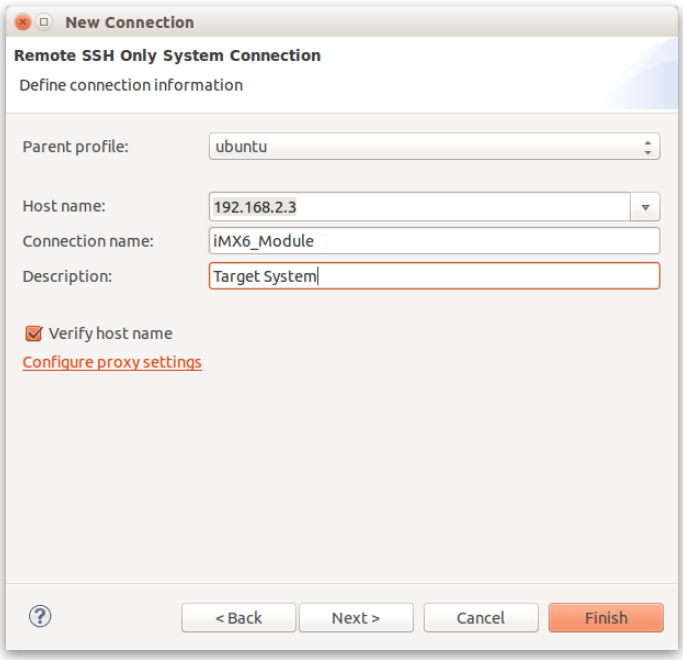

Figure 15: From the 'Remote SSH Only System Connection' dialog window, enter the appropriate information.

Evergreen CO 80439 • +1 (303) 674-9207

Step 3e: Next, from the 'Remote Systems' tab, select 'Connect' (please see screenshot below).

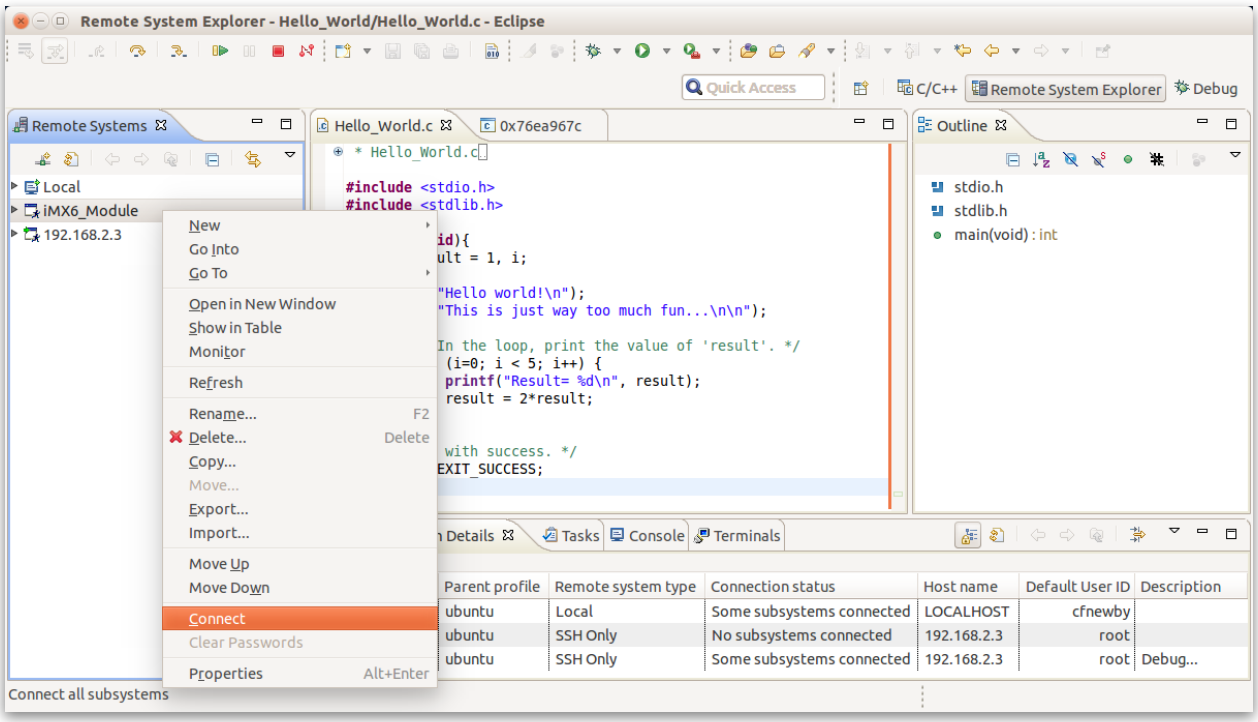

Figure 16: From the 'Remote Systems' tab, select 'Connect'.

Step 3f: Next, from the 'Enter Password' dialog window, enter 'root' on the 'User ID' line (please see screenshot below).

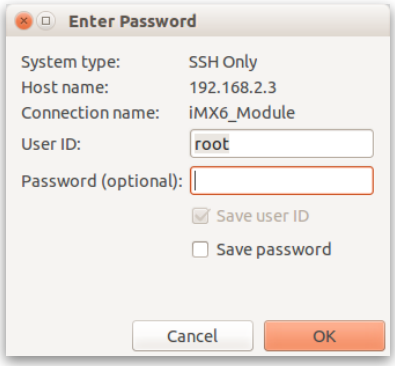

Figure 17: From the 'Enter Password' dialog window, enter 'root'.

Evergreen CO 80439 • +1 (303) 674-9207

Step 3g: Finally, explore the iMX6_Module's file system (please see screenshot below).

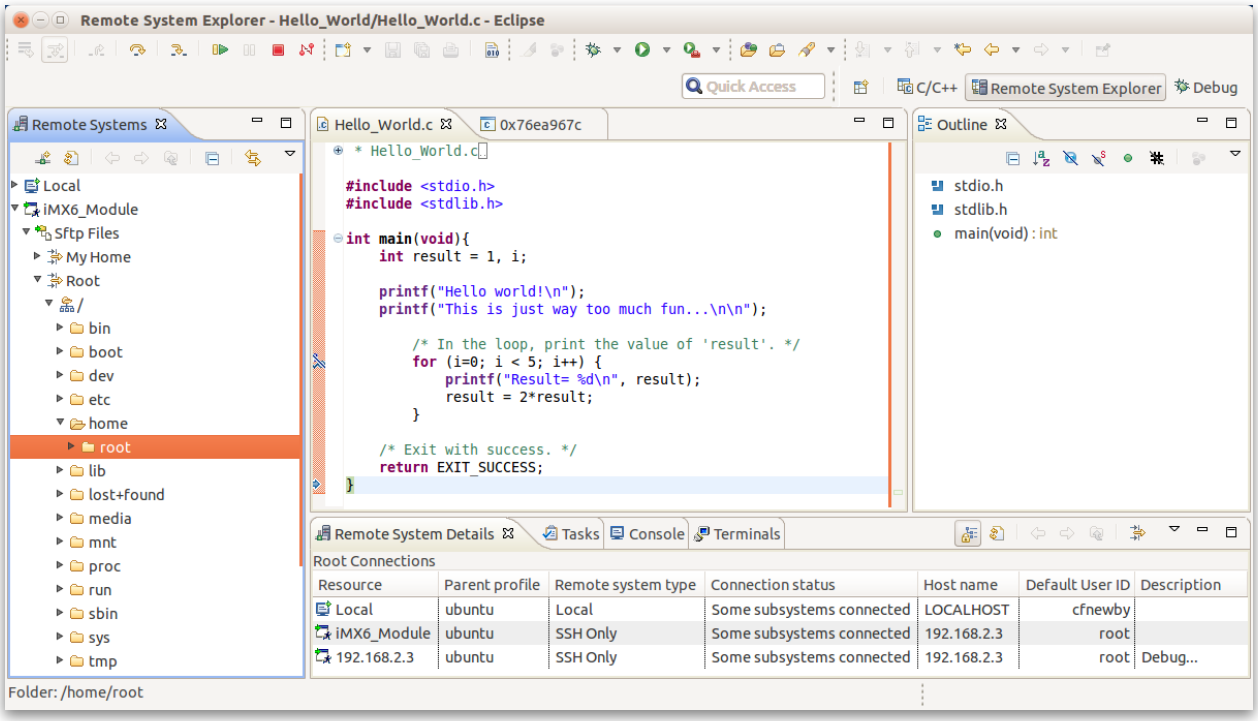

Figure 18: Explore the iMX6_Module's file system.

Step 3h: — SETUP THE DEBUG CONFIGURATION — First, from the 'Project Explorer View' menu, select the 'Hello_World' project, then 'Debug As', and, finally, 'Debug Configurations…' (please see screenshot below).

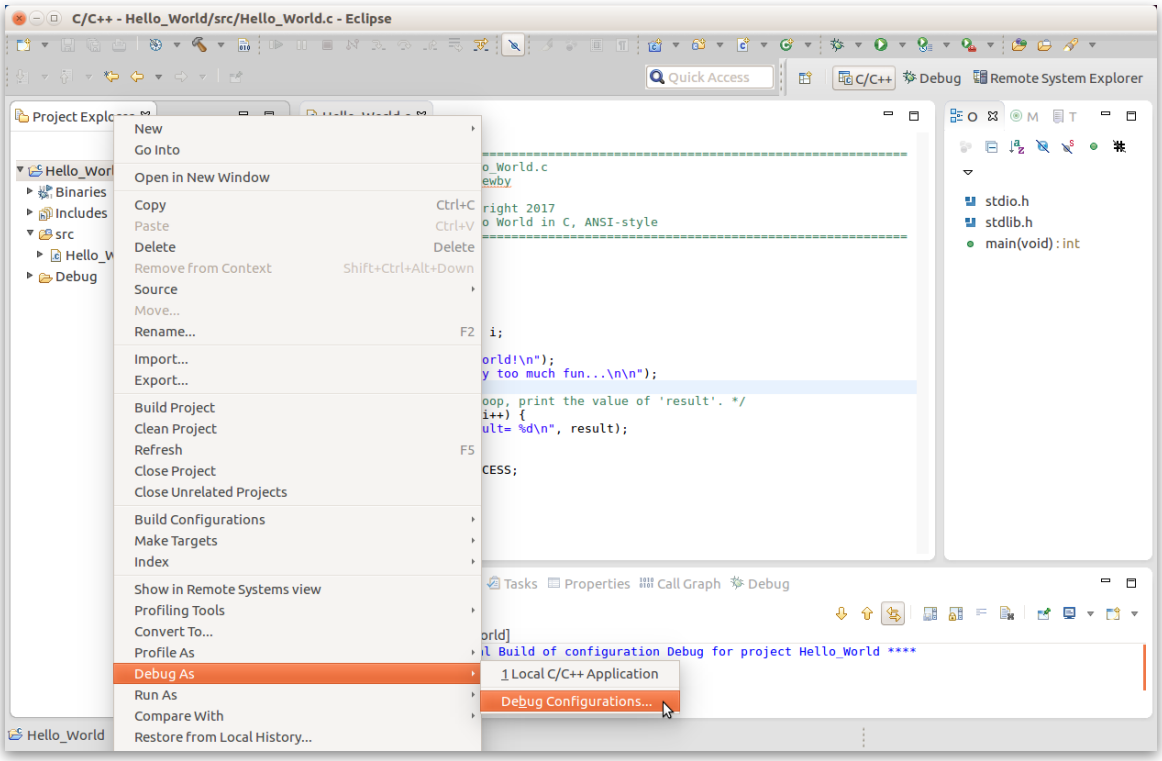

Figure 19: From the 'Project Explorer View', select 'Debug As', and 'Debug Configurations…'.

Evergreen CO 80439 • +1 (303) 674-9207

Step 3i: Next, configure the 'Debug Configurations' dialog window, configure the 'Main' tab as shown (please see screenshot below).

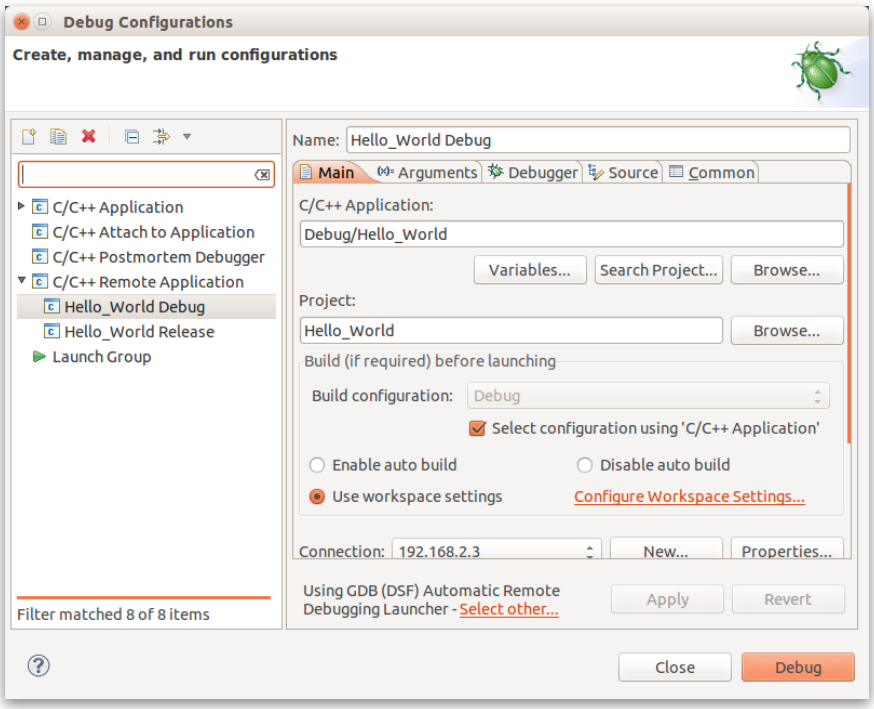

Figure 20: Configure the 'Debug Configurations' dialog window's 'Main' tab as shown.

Step 3j: Next, configure the 'Debug Configurations' dialog window so that, in the 'Debugger' tab, 'GDB debugger:' contains 'arm-angstrom-linux-gnueabi-gdb' (please see screenshot below).

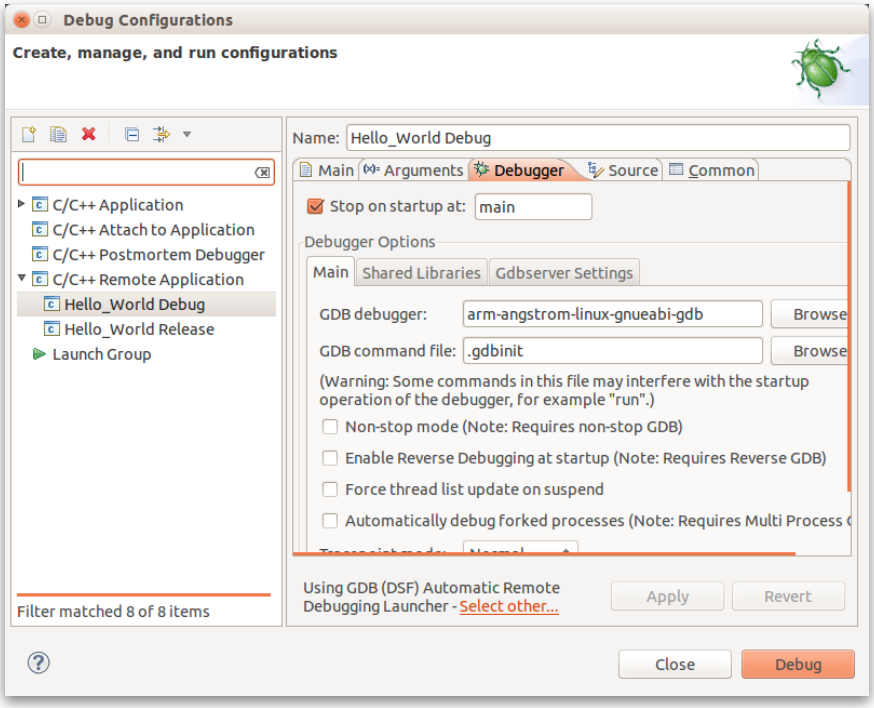

Figure 21: Configure the 'Debug Configurations' dialog window's 'Debugger' tab as shown.

Evergreen CO 80439 • +1 (303) 674-9207

Step 3k: Modify 'Hello_World.c' so that it reflects the following text:

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

int main(void){
       int result = 1, i;

   printf("Hello world!\n");
   printf("This is just way too much fun...\n\n");

       /* In the loop, print the value of 'result'. */
       for (i=0; i < 5; i++) {
              printf("Result= %d\n", result);
              result = 2*result;
       }

   /* Exit with success. */
   return EXIT_SUCCESS;
}
```

Step 3l: — RUN 'HELLO_WORLD DEBUG' — First, from the toolbar, select 'Hello_World Debug" from the Run Icon Menu (please see screenshot below).

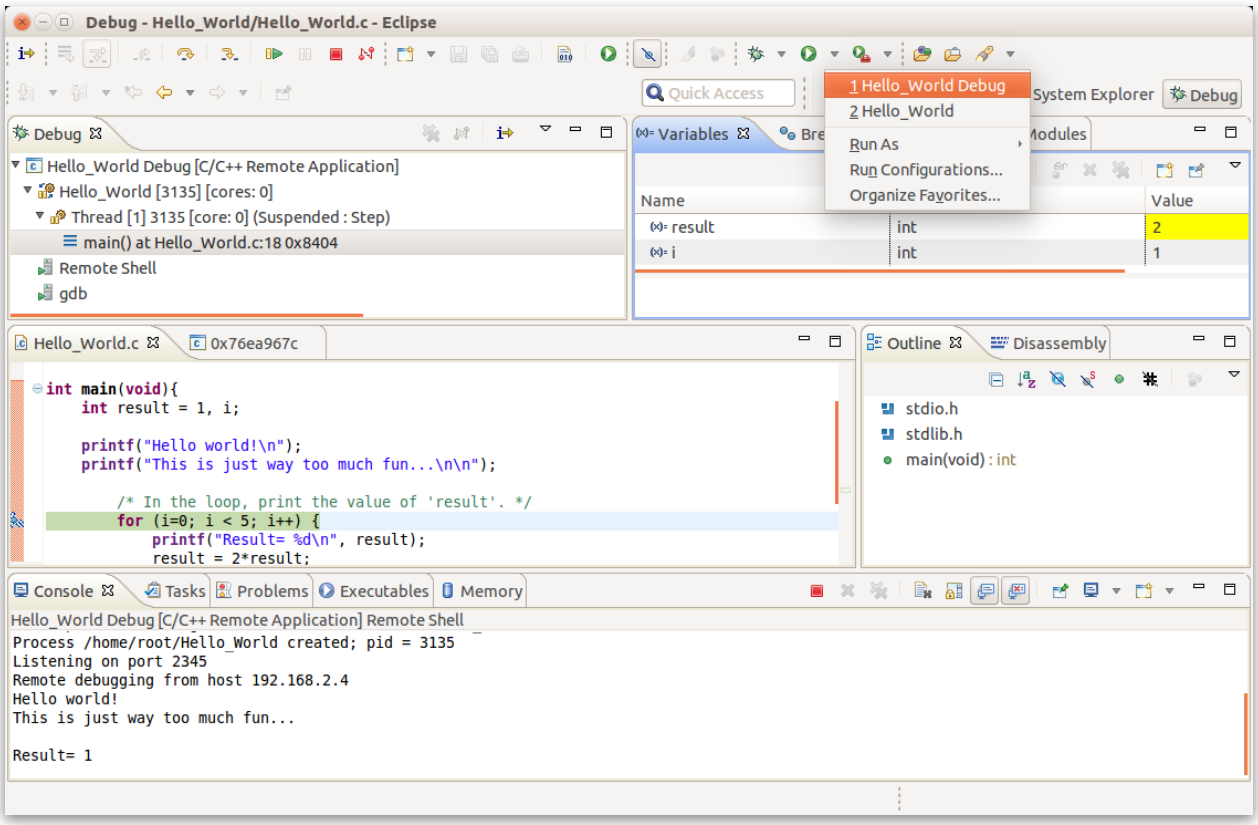

Figure 22: From the Run Icon, select 'Hello_World Debug'.

Evergreen CO 80439 • +1 (303) 674-9207

Step 3m: Next, set a breakpoint at the top of the 'for loop', then select the 'Step Over' button multiple times observing how variables 'result' and 'i' change as the application runs (please see screenshot below).

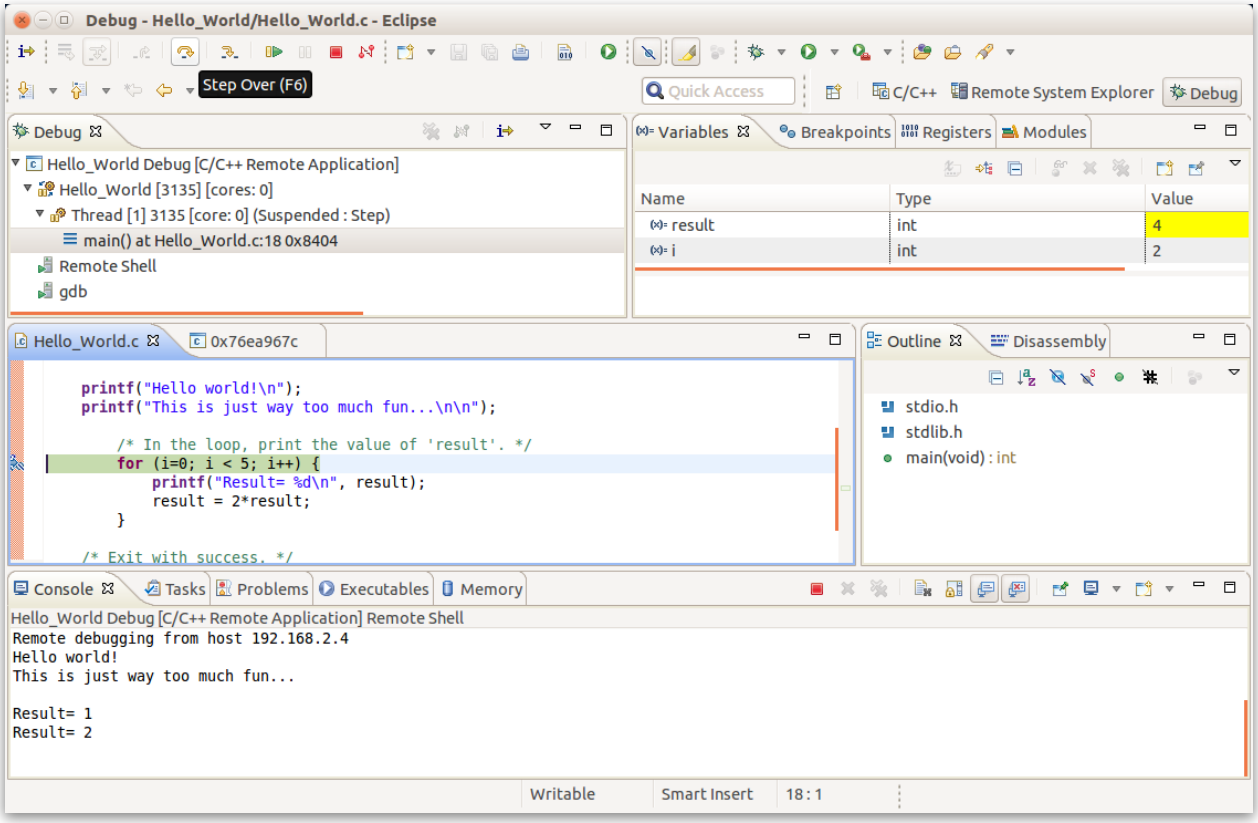

Figure 23: Set a breakpoint at top of 'for loop' then step through the application observing variables change.

Evergreen CO 80439 • +1 (303) 674-9207

Step 3n: Finally, step through to the end of the application then observe the state of each of the internal variables (please see screenshot below).

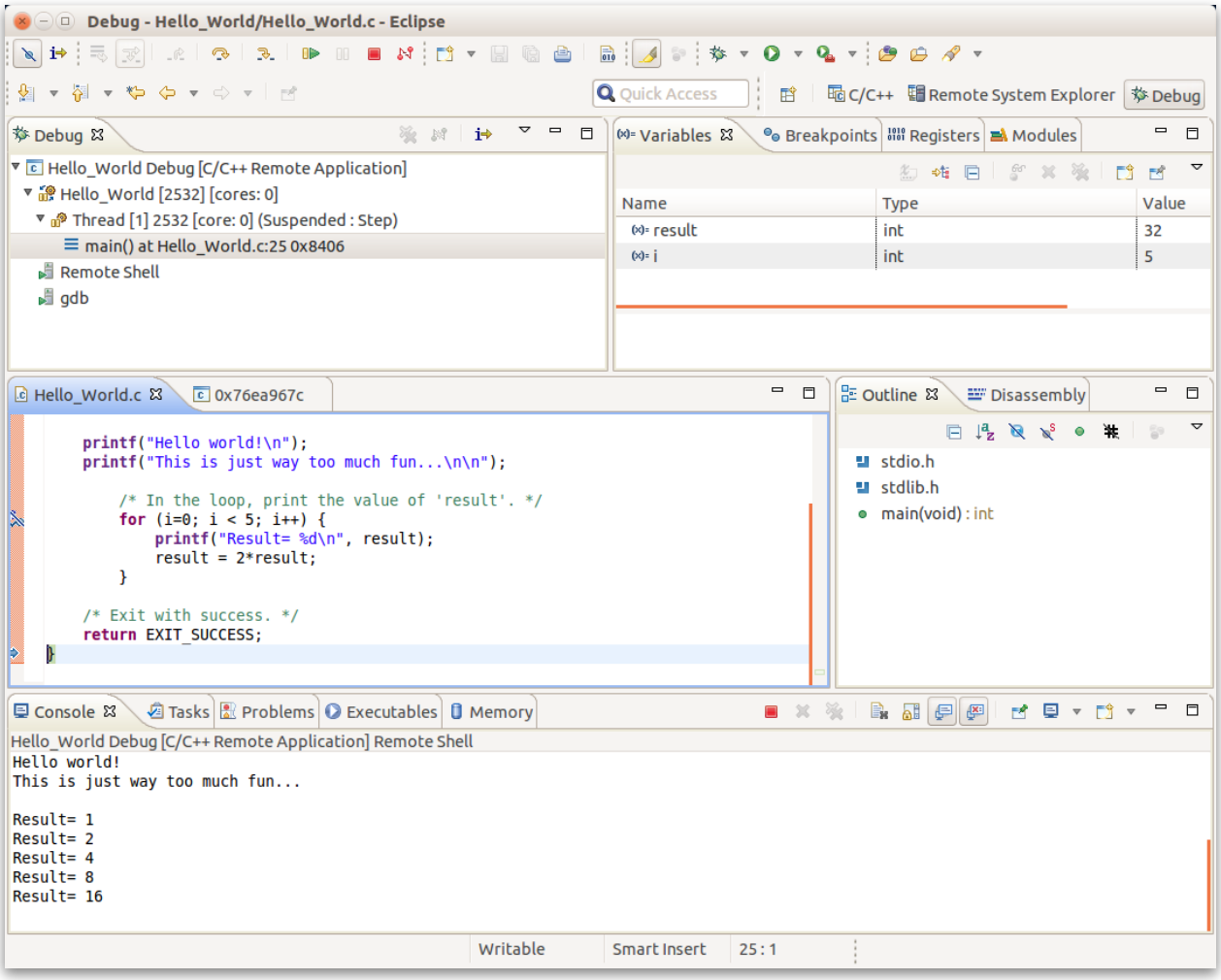

Figure 24: Step through to the end of the application then observe the state of the internal variables.

As a exercise, setup a '**Hello_World Release**' run configuration—using the procedures executed above—then run the release version of the application. Does the application behave as expected?

Evergreen CO 80439 • +1 (303) 674-9207

Additional Resources

In addition to those already described, listed below are a number of resources that may be helpful:

1. *C/C++ Development User Guide*
2. *GCC, the GNU Compiler Collection*
3. *Developing and Debugging C Programs*

For more information regarding this and other Cimarron Systems products, please contact us using the contact information below.

Contact Information:

Cimarron Systems, LLC
Evergreen, Colorado USA
telephone: +1 (303) 674-9207
email: info@cimarronsystems.com
website: https://www.cimarronsystems.com

Revision History:

Date	Version	Notes
9/25/2017	version 1.0	Initial version
7/15/2018	version 2.0	Minor updates
12/2/2019	version 3.0	Minor updates

Appendix A

Some Useful Linux Commands

In the tables below, you will find some useful Linux commands.

File Commands
ls — directory listing
ls -al — formatted directory listing with hidden files
cd 'dir' — connect to directory **'dir'**
cd — connect to home directory
pwd — show current directory
mkdir 'dir' — create directory **'dir'**
rm 'file' — delete **'file'**
rm -r 'dir' — delete **'dir'**
rm -f 'file' — force remove **'file'**
rm -rf 'dir' — force remove **'dir'***
cp 'file1' 'file2' — copy **'file1'** to **'file2'**
cp -r 'dir1' 'dir2' — copy **'dir1'** to **'dir2'**; create **'dir2'** if it does not exist
mv 'file1' 'file2' — rename or move **'file1'** to **'file2'** if **'file2'** is an existing directory, moves **'file1'** into directory **'file2'**
l**n -s 'file' 'link'** — create symbolic link **'link'** to **'file'**
touch 'file' — create or update **'file'**
cat > 'file' — put **'std in'** into **'file'**
more 'file' — output the contents of **'file'**
head 'file' —output the first 10 lines of **'file'**
tail 'file' —output the last 10 lines of **'file'**
tail -f 'file' —output the contents of **'file'** as it grows, starting with the last 10 lines
File Permissions
chmod 'octal' 'file' — change the permissions of **'file'** to **'octal'**, which can be found separately for **'user'**, **'group'**, and **'world'** by adding: • 4 — **read** (**r**) • 2 — **write** (**w**) • 1 — **execute** (**x**) Example: **chmod 777** - **read**, **write**, **execute** for all **chmod 755** - **rwx** for **'owner'**, **rx** for **'group'** and **'world'**. For more options, **man chmod**.

SSH Commands
ssh 'user@host' — connect to **'host'** as **'user'**
ssh -p 'user@host' — connect to **'host'** at **'port'** as **'user'**
ssh-copy-id 'user@host' — add your key to **'host'** for **'user'** to enable a keyed or password-less login
Search Commands
grep 'pattern' 'files' — search for **'pattern'** in **'files'**
grep -r 'pattern' 'dir' — search recursively for **'pattern'** in **'files'**
'command' | grep 'pattern' — search for **'pattern'** in the output of **'command'**
locate 'file' — find all instances of **'file'**
System Information
date — show the current date and time
cal — show this month's calendar
uptime — show current uptime
w — display who is online
whoami — login name
finger 'user' — display information about **'user'**
uname -a — show kernel information
cat /proc/cpuinfo — CPU information
cat /proc/meminfo — memory information
man 'command' — show the manual for **'command'**
df — show disk usage
du — show directory space usage
free — show memory and swap usage
whereis 'app' — show possible location of **'app'**
which 'app' — show which **'app'** will run by default

Evergreen CO 80439 • +1 (303) 674-9207

File / Directory Compression
tar cf 'file.tar' 'files' — create a tar named **file.tar** containing **'files'**
tar xf 'file.tar' — extract the files from **'file.tar'**
tar czf 'file.tar.gz' 'files' — create a tar with Gzip compression
tar xzf 'file.tar.gz' — extract a tar using Gzip
tar cjf 'file.tar.bz2' — create a tar with Bzip2 compression
Network Commands
ping 'host' — ping **'host'** then output results
whois 'domain' — get whois information for **'domain'**
dig 'domain' — get DNS information for 'domain'
dig -x 'host' — reverse lookup **'host'**
wget 'file' — download **'file'**
wget -c 'file' — continue a stopped download of **'file'**
Install from source: **./configure** **make** **make install** **dpkg -i 'pkg.deb'** — install a package (Debian) **rpm -Uvh 'pkg.rpm'** — install a package (RPM)