Install and Use the Sourcery G++ Lite Toolchain with Eclipse in Windows for Cross Development with ARM Single Board Computers

William Wang

ww2@andrew.cmu.edu

Class 2011 School of Computer Science Carnegie Mellon University

Table of Contents

1	Abstract		1
2	Downloa	ading and installing the Sourcery G++ Lite toolchain	1
3	Downloa	ading and installing the Eclipse IDE	2
	3.1 Crea	ting and Building an Example Project	5
	3.2 Mou	nting a Windows folder from an ARM Single Board Computer	
	3.2.1	Creating a Shared Folder Under Windows Vista	
	3.2.2	Creating a Shared Folder Under Windows XP	
	3.2.3	Terminal Access to the ARM SBC	
	3.2.4	Mounting the shared folder	
4	Appendi	x I: Remote debugging with gdbserver and Eclipse in Windows	
5	Appendi	x II: Internet Access While on Multiple Networks in Windows	28

1 Abstract

This tutorial is for those who wish to participate in cross development for ARM single board computers using a Windows machine. Windows is a good choice for those unfamiliar with Linux or prefer a graphical IDE.

2 Downloading and installing the Sourcery G++ Lite toolchain

Download Sourcery G++ Lite ARM toolchain from

<u>http://www.codesourcery.com/sgpp/lite/arm/portal/release858</u>. The Windows Installer is recommended for download. This tutorial will focus on the installer only.

Run the installation file and accept the agreement. Make changes to the default settings as necessary. If multiple users on a single Windows machine will be participating in development, it is recommended to change Modify PATH for current user to Modify PATH for all users. This is so that one can run the Sourcery G_{++} executables in any directory without having to specify a path. Figure 1 shows the screen which modifies this option.



Figure 1, Modifying the PATH

3 Downloading and installing the Eclipse IDE

Please go to http://java.sun.com/javase/downloads/index.jsp and download and install a JDK.

The Eclipse IDE is an open source graphical IDE, which is the recommended IDE for cross developemnt. Download it from <u>http://www.eclipse.org/downloads/</u>. Either the Eclipse IDE for C/C++ Developers or Eclipse Classic should be downloaded. If only C/C++ development is planned for use with Eclipse, the former should be installed. Otherwise, install the Classic version. Installation of Eclipse is as simple as extracting the eclipse folder to the desired location. Click eclipse.exe to start Eclipse. On the first run, Eclipse will ask for a location for its workspace. Choose a desired location and check the box to set the workspace as default. On future runs, Eclipse will no longer ask for a workspace location if the box is checked. A window like below in Figure 2 will be shown.



Figure 2, The Welcome Screen

Click the rightmost icon to reach the workbench which is shown below in Figure 3. The welcome screen will no longer appear in future executions of Eclipse and it will go directly to the workbench.

😂 C/C++ ·	- Eclipse																
<u>F</u> ile <u>E</u> dit	<u>S</u> ource R	efac <u>t</u> or <u>N</u> av	igate S	Se <u>a</u> rch <u>R</u> un	<u>P</u> roject	<u>W</u> indow	<u>H</u> elp										
1 -] &] - *> 4	• 🔶 •		<u>e</u> 🗸	6° - C	• @ •	«	r 🖲 🔻	☆・	0 - 9	} • [6 6	<i>«</i>	Π	1	c/C++	3
🔁 Project	t Explorer 🕺		- 0	Ŋ											🗄 Outli 🔅	() N	/lak 🗧 🗖
		= 🔄	<u>6</u> 9 ▽														<u>6</u> 9 ▽
															An outline is	not avail:	able.
				ి Probler	ms 🖾 🔪	🖉 Tasks	📮 Con	sole 🔲	Propertie	s						e e	· · · · ·
				Descriptio	n			v					Resource	e	Path		Location
				- senper										-			
				•													F
•	0 items se	lected													ſ	1	💌 💖 🔶

Figure 3, The workbench

If Eclipse IDE for C/C++ Developers is not the version installed, the CDT (C/C++ Development Tools) Plug-in must be installed. Go to Help->Install New Software.... As it shows in Figure 4, in the drop-down menu, choose Galileo. A list of categories will appear. Expand the Programming Languages category. Locate and check Eclipse C/C++ Development Tool. Click Next and then Finish to install the plug-in. Restart Eclipse if asked.

🌐 Install		
Available Check the it	Software .ems that you wish to install.	
Work with:	Galileo - http://download.eclipse.org/releases/galileo Find more software by working with the <u>'Available Software</u>	<u>A</u> dd <u>Sites'</u> preferences.
type filter te	dt	
Name	Version	*
▲ 000 P	rogramming Languages Dynamic Languages Toolkit - i 1.0.0.v20090613-1432-390A55V5L9K28LC_ Dynamic Languages Toolkit - I 1.0.0.v20090610-1638-7BE_EkMO1NTYtA Dynamic Languages Toolkit - ¹ 1.0.0.v20090610-1638-7DEB7sQSMRjetCf Dynamic Languages Toolkit - ¹ 1.0.0.v20090610-1638-4kE77g7TAS28NCF Eclipse C/C++ Development T 6.0.0.200906161748 Eclipse Java Development Too 3.5.0.v20090527-2000-7r88FEeFJePyvYeA33 Eclipse XML Editors and Tools 3.1.0.v200905240756-7H6FMVDxtkM-5OgP	E
🔽 Show only	the latest versions of available software $\square H$ ide items that are already installed	
☑ <u>G</u> roup iter	ns by category What is <u>already installed</u> ?	
© Contact al	l update sites during install to find required software < Back Einish	Cancel

Figure 4, Installing Plug-ins

Now Eclipse can be used for cross development for the ARM platform.

3.1 Creating and Building an Example Project.

To start a new C/C++ project, go to File->New->C Project or File->New->C++ Project. For project type, expand *Makefile project* and choose Empty Project. For the Toolchain, choose --Other Toolchain--. An example of this screen is shown in Figure 5. Choose a project name and click Finish.

C Project					
C Project Create C project of selected type					
Project name: Example Use default location Location: C:\Users\William\C\Example	Browse				
Project type: Executable Shared Library Static Library Makefile project Empty Project	Toolchains: Other Toolchain Cygwin GCC				
Show project types and toolchains only if they are supported on the platform (?) < Back					
Figure 5, Creating a new project					

The workbench should look similar to the one in Figure 6. If the Includes folder does not appear under

The workbench should look similar to the one in Figure 6. If the Includes folder does not appear under the Example project folder when expanded, do not worry. In general, the normal case is that Example should be empty. The Includes folder as shown in the figure is created automatically if Eclipse detects that Cygwin GCC libraries are installed on the computer.

6	● C/	C++ - Eclipse				- 1	Freedores	iner" its								-			. . . X
	<u>F</u> ile	<u>E</u> dit <u>S</u> ource	Refactor	<u>N</u> avigate	Se <u>a</u> rch	<u>R</u> un	<u>P</u> roject	<u>₩</u> indow	<u>H</u> elp										
	1 1	• 8 ≙ 8 • 7 • %) ⇔ • ⇔	~	ú	<u>â</u> • 6	ŝ - €	• 🕝 •	≪, •	• 🛞 •	☆ ▼	0 - 9	• [(9 🖨	<i></i> –	Π	1	🛅 C/C++	•
	P	roject Explorer	×													- 0	🗄 Outli 🛛	3 01	Mak 🕛 🗖
			Ξ	€ []															 €9 ▽
	4 [2	🐣 Example > 🗊 Includes															An outline	is not avail	able.
		-																	
1																			
						Prohlem	. 8	🖉 Tasks	🗉 Con	sole 🔲	Pronertie	,							
ł					0 iter	ms													
					Des	cription				*					Reso	urce	Path		Location
	•	III			+ -												1		۴
	□◆	😂 Exa	mple															1	🐟 💝 😓

Figure 6, Example Project

The tutorial example project will be to create a program that reads a number from the command line and prints the factorial of that number or -1 if the number is negative or would overflow a 32-bit integer. A total of 4 files will be created for this project. They are as follows:

- main.c The source file that will contain main()
- factorial.h The header file that will provide macro constants and function prototype for the factorial function
- factorial.c A source which will contain an implementation of the factorial function
- makefile The file needed to build the project, which will be later explained in greater detail.

To create a new file in the project, right click the Example project folder and go to New->File, New->Header File, or New->Source File (for makefiles, .h files, and .c files, respectively) depending on the desired file type. Figure 7 shows the creation of main.c. Click Finish to create the new file.

New Source	File	
Create a new s	ource file.	C
Source <u>F</u> older:	Example	Br <u>o</u> wse
Source File:	main.c]
Template:	Default C source template 🔹	Configure
?	<u> </u>	Cancel

Figure 7, Creating main.c

Figures 8, 9, 10 show what the source code for what main.c, factorial.h, and factorial.c should look like. Figure 8, main.c

🕼 factorial.c 🕼 factorial.h 🔂 main.c 🛛	
/*	*
* main.c	
*	
* Created on: Jul 22, 2009	
* Author: ww2	
•7	
<pre>#include <stdio.h></stdio.h></pre>	
<pre>#include <string.h></string.h></pre>	
#include "factorial.h"	
<pre>int min((no argo; char argo;)) (if(argc != 2) { printf("Usage: factorial n"); return -1; } printf("%d\n", factorial(atoi(argv[1]))); return 0; }</pre>	~
<	Þ



Figure 9, factorial.h





After writing these files, it should now be possible compile them into a program. This can be done in two ways. One way is to just call the compiler, arm-none-linux-gnueabi-gcc, directly. The other way is to use a makefile. However, for larger projects, calling the compiler directly becomes confusing and complicated. Makefiles significantly simplify the compiling process and thus will be the method explained in this tutorial.

make is a utility for automatically building executable programs and libraries by using files called makefiles which specify how to derive the target program from its dependencies. Figure 11 shows an example makefile to build the factorial program



Figure 11, makefile

In a makefile, one defines a set of targets, the set of dependencies for that target, and the commands that should be run to build the target. The three targets in the makefile illustrated in Figure 11 are *all, %.o,* and *clean.* Anything to the right of the colon is a dependency. Below the rule is the list of commands to build the target. This list of commands **must be TAB indented**. The makefile in Figure 11 only has one command per target, but there can be more.

One can also define macros like in the example. For instance, OBJECTS is a macro which lists the object files that are required to build the complete program. The example also shows use of some of the internal macros that *make* recognizes such as CC and CFLAGS and changes them. CC is a macro that contains a string for the C compiler. It normally defaults to a compiler called cc, but for programs to run on the ARM SBC, the compiler must be arm-none-linux-gnueabi-gcc. The makefile in Figure 11 has this change.

CFLAGS is a macro used to store the set of options or flags that the C compiler should use in compiling. The flags themselves will be discussed later. \$< is a macro that is defined to be the single dependency of the current running target.

The example also shows the use of inference rules. % is a wild card character that matches zero or more characters. The rule, %.o: %.c, infers that an object file can be built by its corresponding source file.

In Eclipse, *all* and *clean* are special targets. When building normally, Eclipse calls make on the *all* target. When cleaning up the project, Eclipse calls make on the *clean* target.

Out of the flags shown in the CFLAGS macro, only –static is necessary. The purpose of the –g flag is to allow for the code to be run through a debugger such as GDB and is thus recommended, but not necessary. –O2 is a flag for optimization purposes. The –static flag tells the compiler to not use dynamic linking to shared libraries. Normally, programs are compiled such that they make references to libraries located elsewhere. At runtime, the dynamic linker is called which then loads the relevant subroutines into the program. Part of the reason that this is done is to cut down the space taken up by the program. However, the ARM single board computer neither has the correct libraries nor has a usable dynamic linker because it does not have the space to store it. Therefore, programs must be statically linked or have the relevant subroutines loaded at compile time. Although this makes the program larger, it is an insignificant increase. The –o flag is for naming the output file. In Figure 11, the program will be named *factorial* when it is compiled. The –c flag tells the compiler to assemble the source files, but not to link them together.

Automatic building is the default setting, but should be turned off. Before building the program, click Project on the menu bar and uncheck Build Automatically as shown in Figure 12. If this is not done, the project will be built automatically even after cleaning. This could potentially cause confusion because cleaning implies that all the files should be removed, but building will cause them to reappear again. Also, sometimes building should be purposely held off until later because it could potentially take a long time for larger projects, slowing down the development process.



Figure 12, Uncheck Build Automatically

Now either click Build All in the same menu as shown in Figure 12 or right click Example and click Build Project. If everything was done correctly, then the Project Explorer should be similar to the one in Figure 13.



Figure 13, After building

It can be seen that the object files as well as the *factorial* program were created. Now this program can be loaded onto the ARM single board computer and it should be able to run. The next tutorial will teach how to connect the Windows machine with the ARM SBC.

3.2 Mounting a Windows folder from an ARM Single Board Computer

Once programs have been developed and written on Windows for the ARM SBC, they can be tested by being executed on the SBC. Therefore, the SBC needs some way to access the compile programs. A way to do this is to have Windows share a folder and have the SBC mount this folder as part of its file system. The steps are as follows.

3.2.1 Creating a Shared Folder Under Windows Vista

Example Example	۵	
		Explore
		Open
		7-Zip 🕨
		Convert to file format
		Open with RapidCRC
		Share
	2	TortoiseHG •
		Restore previous versions
	12	Combine supported files in Acrobat

Right click on the folder to be shared. In the menu click Share... as in Figure 14.

Figure 14, The right click menu

The window in Figure 15 will appear

choose people to she	re with	
eople must have a user ac	ount and password for this c	omputer to access files you have shared. To
hange this setting, use the:	<u>Network and Sharing Center</u> .	
		✓ Add
Name		Permission Level
🔏 William Wang		Owner 🔻

Figure 15, File Sharing

If only the folder creator will use the folder, then click Share to share the folder. Otherwise click the arrow on the drop down menu and choose other users and click Add to add them. Change their permission levels as necessary. There are three permission levels:

- Reader Users can display the contents of the folder, open files, display attributes, and run programs.
- Contributor Users have all the rights of Reader, plus the ability to create new folders and files within the shared folder and change or delete any files they have contributed to.
- Co-owner Users have all of the rights of Contributor, plus the ability to change or delete any files.

3.2.2 Creating a Shared Folder Under Windows XP

Right click on the folder to be shared. In the menu, click Sharing and Security... as in Figure 16.

Exa	mole
	Open
	Explore
	Search
	Sharing and Security
	Add to archive
	Add to "Example.rar"
	Compress and email
	Compress to "Example.rar" and email
	Send To
	Cut

Figure 16, The right click menu

The window in figure 17 will appear.

Example Properties 🔹 💽 🔀						
General Sharing Customize						
Local sharing and security To share this folder with other users of this computer only, drag it to the <u>Shared Documents</u> folder.						
To make this folder and its subfolders private so that only you have access, select the following check box.						
Network sharing and security To share this folder with both network users and other users of this computer, select the first check box below and type a share name.						
Share name: Example						
Allow network users to change my files						
Learn more about <u>sharing and security</u> .						
 Windows Firewall is configured to allow this folder to be shared with other computers on the network. <u>View your Windows Firewall settings</u> 						
OK Cancel Apply						

Figure 17, Folder Properties

Check the two boxes as shown in the figure and click Apply. Now this folder is shared.

3.2.3 Terminal Access to the ARM SBC

There are two ways to get a terminal window on the single board computer, through serial port or through telnet. A free client that can do both and more is PuTTY, which can be downloaded at http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. Putty.exe is only one that is needed. When PuTTY is opened, the window in Figure 18 will appear.

Reputity Configuration	×
Category:	
🖃 Session	Basic options for your PuTTY session
Logging ⊡ Terminal 	Specify the destination you want to connect to Host <u>Name (or IP address)</u> 22
Features	Connection type: <u>R</u> aw <u>T</u> elnet Rlogin <u>S</u> SH Serial
	Load, save or delete a stored session Sav <u>e</u> d Sessions
Colours ⊡ Connection Data Proxy Telnet Rlogin ⊕ SSH	Default Settings Load CMU UNIX Save Fish Save Home Delete
SSH Serial	Close <u>w</u> indow on exit: Only on clean exit
About	Open <u>C</u> ancel

Figure 18, PuTTY

PuTTY can save sessions so that the process of configuration does not have to be done every time. Just put a name in the text field under Saved Sessions and click Save. Click load from the list of sessions and the configuration will be loaded automatically or double click the desired session and a terminal window will open immediately.

3.2.3.1 Access ARM SBC via Serial Port

For connecting through the serial port, connect a null modem cable between one of the serial ports of the Windows computer and the serial port of the SBC. Then click Serial in the tree view and set the options as shown in Figure 19. These are the options needed for this particular kind of ARM single board computer.

😵 PuTTY Configuration	and a second second	
Category:		
Category: Session Cogging Terminal Keyboard Bell Features Window Appearance Behaviour Translation Selection Colours Colours Colours Froxy Telnet Rigin SSH	Options controlling le Select a serial line Serial line to connect to Configure the serial line <u>Speed (baud)</u> Data <u>b</u> its Stop bits Parity Elow control	COM1 115200 8 1 None • None •
<u>About</u>		en <u>C</u> ancel

Figure 19, Serial configuration

Then click Session in the tree view to return to the screen in Figure 18 and then click the Serial radio button. It should then look like Figure 20.

🕵 PuTTY Configuration	States a sector and the sector of the sector	X
Category:		
🖃 Session	Basic options for your PuTTY s	ession
Logging	Specify the destination you want to conn	ect to
Keyboard	Serial li <u>n</u> e	Speed
Bell	COM1	115200
Features	Connection type: ◎ <u>R</u> aw ◎ <u>I</u> elnet ◎ Rlogin ◎ <u>S</u> S	H 💿 Serial
Appearance Behaviour Translation Selection Colours Connection Data Proxy Telnet Rlogin SSH	Load, save or delete a stored session Sav <u>e</u> d Sessions Default Settings CMU UNIX Fish Home SBC	Load Sa⊻e Delete
Serial	Close <u>w</u> indow on exit: Only on the second seco	clean exit
About	<u>O</u> pen	<u>C</u> ancel

Figure 20, Serial session

Choose the correct serial line (e.g. COM1, COM2, etc), save the session if desired, and then click Open. The terminal window will show up like in Figure 21. When asked for a login, put root. There is no password.



Figure 21, The terminal window

If nothing shows up, press Enter and the login prompt should appear. The text in Figure 21 may not always appear

Note: If pressing Enter does not work, then it may be possible that another process or device is using the same serial line as the SBC. This is especially true of laptops which normally do not have serial ports. Please check the serial line properties. If there is something already using the line, either change the serial port that the SBC uses or move that process or device to a different line. This tutorial will not go in depth on this problem as this is not the focus and there are too many possibilities for the cause of this problem which may need different methods to fix.

3.2.3.2 Access ARM SBC via Telnet

For connecting using telnet, the SBC needs an IP address and needs to be on a network which the Windows computer can access. Assuming there is no other way to connect to the SBC, a router is necessary. The SBC is DHCP enabled and thus any router which has its DHCP server enabled can give it an IP address. Connect both the Windows computer and the SBC to the same computer. Wait for some time so that the computers can get their IP addresses. Find the computer in the router's DHCP client table. It will be normally the one with no name. Copy down the associated address and make sure the Windows computer can detect the SBC by using the ping command in the Command Prompt. An example is shown in Figure 22.



Figure 22, The ping command

If above results when pinging to the SBC, then the Windows computer should be able to connect. Then in PuTTY, click the Telnet radio button and type in the address, save if the session if desired, and click Open. See Figure 23 for an example.

🕵 PuTTY Configuration	an of Autor	X
Category:		
🖃 Session	Basic options for your PuTTY se	ssion
Logging	Specify the destination you want to conne	ect to
Keyboard	Host <u>N</u> ame (or IP address)	Port
Bell	10.10.101	23
Features ⊡ Window	Connection type: <u>R</u> aw	H 🔘 Serial
Appearance Behaviour Translation Selection	Load, save or delete a stored session Sav <u>e</u> d Sessions	
Colours ⊡ Connection Data Proxy Telnet Rlogin SSH	Default Settings CMU UNIX Fish Home SBC	Load Sa <u>v</u> e Delete
Serial	Close <u>w</u> indow on exit:	lean exit
About	Open	<u>C</u> ancel

Figure 23, Telnet configuration

The terminal will take some time to load and then the login prompt should appear similarly to the one for the serial connection. Again, the login name is root. There is no password.

3.2.4 Mounting the shared folder

Now that a shared folder (refer to step 1) has been created and the Windows machine can connect to the console on the ARM SBC, mounting is the final step to start running programs compiled on the Windows machine.

For mounting, the Common Internet Network Interface (CIFS) will be used. As the name suggests, networking will be required and thus the Windows machine needs an IP address that the SBC can access. This can be done simply by putting the Windows machine and the SBC on the same network. If connecting to a terminal was achieved through telnet, this is already done. If it was achieved through serial port, there are two ways to do it. One requires a crossover cable and is somewhat complicated. The other way is to follow the instructions for setting up the telnet connection. Connect both machines to the router and make sure they have connectivity. Now use either the DHCP table of the router or the ipconfig command in the Command Prompt to find the IP address of the Windows machine. An example run of this command is shown in Figure 24.



Figure 24, ipconfig

Copy down the IP address located in the network that the SBC is in. Now in the SBC, type the below command if on Vista:

mount -t cifs //<IP address>/<share name> -o username=<username>,password=<password> /mnt

Where <IP address> is the IP address of the Windows machine, <share name> is the path to the folder in Vista (e.g. C:\Users\Example, share name is Users/Example) or the share name chosen in Figure 17 in XP, <username> is the username of the user who wishes to access this folder, password> is the user's password. The password is in clear text. If this is not secure enough, then type in

mount -t cifs //<IP address>/<share name> -o username=<username> /mnt

The terminal will then prompt for a password in hidden text. If the username is not specified like below:

What this command does is that it mounts the folder in the /mnt directory. Thus the files can be accessed by going to that directory on the SBC. A different directory can be used if desired. The /mnt directory is meant for temporary mounts. To unmount the folder, use the below command:

umount /mnt

Now for an example, the factorial program created in the Sourcery G++ Lite toolchain tutorial will be placed into an example shared folder. The folder will be mounted by the SBC and the code will be executed. Then the folder will be unmounted.

First, place the factorial binary into the shared folder as shown in Figure 25.

to dear the P address local	of it the national Had the	WAR NEWS	BC, tare Ballation	
Computer 🕨	SW_Preload (C:) → Example		✓ 49 Search	٩
<u> </u>)			
🎍 Organize 👻 🏢 Views 👻	🖲 Burn	_	_	0
Favorite Links Documents Pictures Music Recently Changed Searches Public	Name Date modif	Type Size		
Folders 🔨				
1 item				1

Figure 25. factorial in the shared folder

Then, open a terminal window and enter in the mount command, change to the /mnt directory, execute the program, and *unmount*. Figure 26, shows an example terminal run from Telnet.

🛃 10.10.101 - PuTTY	
Debian GNU/Linux 4.0	*
GESBC login: root	
Last login: Sun Jul 26 18:52:47 2009 from 10.10.10.100 on pts/1	
Linux GESBC 2.6.30 #5 Thu Jul 2 17:42:45 EDT 2009 armv5tejl	
The programs included with the Debian GNU/Linux system are free software:	
the exact distribution terms for each program are described in the	
individual files in /usr/share/doc/*/copyright.	
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent	
permitted by applicable law.	
GESBC:~# mount -t cifs //10.10.10.100/Example -o username=ww2 /mnt	
Password:	
GESBC:~# cd /mnt	
GESBC:/mnt# ls	
factorial	
GESBC:/mnt# ./factorial 5	
GESBC:/mnt# cd	
GESBC:/# umount /mnt	
GESBC:/# cd mnt	
GESBC: MINT# IS	
GESBC:/mht#	
	Ŧ

Figure 26, Example run

There is an extra command used in this example called *ls. ls* is a command to list all the files and folders in a directory and is very similar to dir in a Windows Command Prompt. Is was used in this example to demonstrate that mounting and unmounting were successful. After mounting, the factorial program was listed. After unmounting, the directory became empty. One thing to note is that the program was called using ./factorial (i.e. the absolute path) rather than just simply factorial. Unlike Windows, Linux does NOT check for the program in the current directory. It only checks for programs in the directories listed in its PATH variable. Another thing to note is that unmounting cannot occur while still in the directory. Hence, this is why the directory was changed to the parent directory before calling *umount*. Now, one can run programs compiled on a Windows machine using a cross development tool on the ARM SBC.

4 Appendix I: Remote debugging with gdbserver and Eclipse in Windows

As stated above in section 3.1, if a program was compiled with the –g option, then it is possible debug it with a debugger such as *gdb* (GNU Debugger). To learn more about *gdb* and the various functions it provides, please go to <u>http://sourceware.org/gdb/current/onlinedocs/gdb_toc.html</u>. The Sourcery G++ Lite comes with its own version of *gdb* called arm-none-linux-gnueabi-gdb which can be used to debug programs using this toolchain. Unfortunately, as Windows does not run on an ARM processor architecture, it cannot natively run ARM code. To solve this problem, it is possible to have an ARM based computer (such as the SBC) serve the program through a program called *gdbserver* and do debugging remotely. Basically how this works is that a debugger remotely connects to the ARM computer and has the ARM computer perform the execution of the program for the debugger rather its host computer. Figure 27 shows an example run of *gdbserver*.



Figure 27, gdbserver

gdbserver takes at least 2 arguments. The first argument it takes is the location of *gdb*. This can be a combination of an IP address and a port as shown above, or a different port (such as a serial port). The second argument is the program to serve. Any arguments that follow are used by the program being debugged. As it shows, *gdbserver* waits and listens for a connection from an instance of *gdb*. One thing to note is that *gdbserver* cannot be stop through the means of pressing CTRL+C. *gdbserver* terminates either by ending a debugging session or using a command called *kill* through another terminal.

Eclipse can be used in tandem with *gdb* to provide a graphical debugging environment and can also interact with *gdbserver*. To set this up, in Eclipse, go to Run->Run Configurations (or Debug Configurations) to see the screen in Figure 28.

Run Configurations		X
Create, manage, and run configurat	ions	
Filter matched 2 of 2 items	Configure launch settings from this dialog: Press the 'New' button to create a configuration of the selected type. Press the 'Duplicate' button to copy the selected configuration. Press the 'Delete' button to remove the selected configuration. Press the 'Filter' button to configure filtering options. - Edit or view an existing configuration by selecting it. Configure launch perspective settings from the <u>Perspectives</u> preference page.	
?	ßun	Close

Figure 28, Run Configurations

Double-click C/C++ Application to create a new configuration. Give it a name. Note that the program has to be on both the ARM computer and the machine running Eclipse. This is because the symbol table and various other bits of debugging information are loaded from the local copy of the program. Thus, choose the project to debug under the Main tab. Then choose the program to debug. In the case that the program's file extension is not *.exe*, the program must be found through Browse... Otherwise, the program can be found using Search Project... Figure 29 shows how to set it up for the example created in section 3.1.

Run Configurations			×
Create, manage, and run configurat	ions		
Image: Second secon	Name: Example Configuration Image: Main Image: Common image:	(Search Project)	Browse
Filter matched 3 of 3 items		Apply	Revert
3		Run	Close

Figure 29, Configuring the Main tab

Now click the Debugger tab. Change the option in the dropdown menu to *gdbserver* Debugger. Choose a function to stop at if desired (if unchecked, the program will attempt to run to completion). Change the debugger to arm-none-linux-gnueabi-gdb. The Shared Libraries tab is for adding the shared libraries that gdb should use. The assumption is that no shared libraries are used and thus the Shared Libraries tab will be skipped. Move to the Connection tab. Change the connection type to the one being used. If using TCP, type in the IP address and port number of the computer where *gdbserver* is executing. Click Apply afterwards to save the configuration. Figure 30 shows an example of how the Main and Connection tabs under the Debugger tab should look.

Main Shared Libra	ries Connection	
GDB debugger:	C:\Program Files\CodeSourcery\Sourcery G++ Lite\bin\arm-none-linux-gnueabi-gdb.exe	<u>B</u> rowse
GDB command file:	.gdbinit	B <u>r</u> owse
(Warning: Some cor	nmands 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	
Main Shared Libra	aries Connection	
Туре: ТСР -		
Host name or IP add	dress: 10.10.10.101	
Port number:	47476	

Figure 30, Configuring the Debugger tab

If *gdbserver* is running, then it is now possible to start the remote debugging process. If in Debug Configurations, choose the correct debug configuration and click Debug. Otherwise, click the arrow right of the "**bug icon**" in the menu bar at the top and choose the correct debug configuration (refer to Figure 6). The message box in Figure 31 should pop up.

2	This kind of launch is configured to open the Debug perspective when it suspends.
•	This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.
	Do you want to open this perspective now?
<mark>√ R</mark> e	member my decision
	Ves No

Figure 31, Changing Perspective

Eclipse asks if a switch should be made to the Debug perspective. Debug perspective gives many useful tools such as a register viewer and a memory viewer. It is recommended to click Yes and check the box to remember the decision. If Yes is clicked, the screen will change to something similar to the one in Figure 32.

Debug - Example/main.c - Eclipse	fram Saura and			
<u>Eile E</u> dit <u>S</u> ource Refact <u>or</u> <u>N</u> avigate Se <u>a</u> rch <u>R</u> un <u>P</u> roject <u>W</u> indow <u>H</u> elp				
📑 • 🖃 👜 🐘 🕸 • O • 💁 • 🤔 🕫 🛷 • 📝 🐓 •	\$ \$ • • •		E 🕉	🖇 Debug 🔤 C/C + »
🏶 Debug 🖾 🔍 🖓	🗆 ⋈= Variables 🖾	🔏 Breakpoints 🐰 Regis	sters 🛋 Modules	- 0)
🍇 🚧 🤣 🕪 🗉 🔳 🕅 🥄 🖓 🔅 🖓	7		🆢 📲 📄	🍫 🗳 ≍ 💥 ▽
Example Configuration [C/C++ Application]	Name			Value
gdbserver Debugger (8/5/09 11:16 PM) (Suspended)	66 ⁹ argc			2
I hread [1] (Suspended: Breakpoint hit.)	💞 argv			0xbe81be14
Cryptogram Files/CodeSourcen/Sourcen/G++ Lite/bin/arm-pone-linux-gnueab	i-			
C:\Users\William\C\Example\factorial (8/5/09 11:16 PM)				
	•			
				<u> </u>
•	•			
R factorial h)	- 8)(🖳 Outling 🕞 Disasson	
t+				
* main.c		- All		
*			int main(int ar	gc, char * ^
* Created on: Jul 22, 2009			if(argc !=	2) (
* Author: ww2		=	0x00008210 <mai< p=""></mai<>	n>: cmp ≣
*/			0x00008218 <mai< td=""><td>n+8>: bne</td></mai<>	n+8>: bne
#include <stdio.h></stdio.h>			printf("Usage: fa
<pre>#include <string.h></string.h></pre>			0x0000823C <mai< td=""><td>n+44>: 1ar n+48>: b1</td></mai<>	n+44>: 1ar n+48>: b1
#include "factorial.h"			0x00008244 <mai< td=""><td>n+52>: mvn</td></mai<>	n+52>: mvn
			0x00008248 <mai< td=""><td>n+56>: pop</td></mai<>	n+56>: pop
int main(int argc, char *argv[]) {			0x0000824c <mai< td=""><td>n+60>: 1dr</td></mai<>	n+60>: 1dr
			0x00008250 <mai< td=""><td>n+64>: and</td></mai<>	n+64>: and
Console 23 Vall Tasks C Problems V Executables Viewory	1.16 0140	■ ~ %		
Example Configuration [C/C++ Application] C:\Users\william\C\Example\factorial (8/5/09 .	1110 P(V)			
				*
	Smort Incort	12.1		
Writable	Smanumsent	10.1		

Figure 32, Debug Perspective

As shown in Figure 32, other than the tool mentioned above, there is a disassembly of the code where one can choose to move to different lines or run up to specific lines of assembly. The variables that are in scope are also shown. In this configuration a breakpoint was set at main. More breakpoints can also be set as well by double clicking the space on the left of the code. In the upper left hand corner under

the Debug tab is another place that can be used to step through the code. To have the code step until the next breakpoint, either click the Pause/Play icon or press F8. Once the code has terminated, the screen should look similar to Figure 33.

Debug - Example/factorial.c - Eclipse		1 10.000		-	
<u>Eile E</u> dit <u>S</u> ource Refac <u>t</u> or <u>N</u> avigate Se <u>a</u> rch <u>R</u> un <u>P</u> roject <u>W</u> indov	v <u>H</u> elp				
📑 • 🗟 🖮 🔅 • 🔕 • 🤷 • 🍠 •	} • ÿ • ♥>	⇔ - ⇔			📑 🏇 Debug 💀 C/C + »
🏇 Debug 🛿	- 8)	🗱 Variables 🤷 Br	eakpoints 🛛	🗙 💥 🞯 -	a 🔪 🖪 🖨 🔄 ⁻ 🗖 🗖
🙀 🖉 🖉 🛤 🖩 🖬 🔍 🔍 👰 🙀	> 🕫 😽 📢		_		
c <terminated>Example Configuration [C/C++ Application]</terminated>					
© <terminated>gdbserver Debugger (8/5/09 11:16 PM) (Exited. Exit o</terminated>	ode = 0.)				
sterminated, exit value: 0>C:\Program Files\CodeSourcery\Source	8/5/09 11:16 PM				
	1				
h factorial.h 🗋 makefile 🖸 main.c 🖻 factorial.c 🛛				🗖 📴 Outline 🛽	Disassembly 🛛 🔽 🔤
/* * factorial c			-	<u> </u>	~
*					*
* Created on: Jul 22, 2009					
* Author: ww2 */				=	
#include "factorial.h"					
int factorial (int n) { int i, fact = 1;					
if (n < 0 n > MAX_FACTORIAL)					
return -1;					
for(1 = 2; 1 <= n; ++1)				•	<u></u>
			P		
Console XX Var Tasks X Problems V Executables	C) [-1 /0 /F (00 11-16 DM)		s % 🗔 🔝 🔛	
<terminated> Example Configuration (C/C++ Application) C:\Users\William\</terminated>	_\txample\ractor	Iai (6/5/09 11:10 PW)			A
					-
					F.
	Writable	Smart Insert	8:1		

Figure 33, Program Termination

If one looks at the terminal session, it should look similar to Figure 34.



Figure 34, gdbserver Termination

As shown in the figures, the printing to the console occurs on the ARM computer and not the console in Eclipse. Also shown is that gdbserver has terminated. To debug the program again, run gdbserver again.

One final note about the –g option. As stated before, this flag is needed for use with *gdb*. This is because this option tells the compiler to add extra information needed to debug effectively. However, also noted above is the fact that there needs to be two copies of the program, one on the machine running Eclipse and one on the ARM machine. *gdb* only needs to look at the local copy to get the debugging information and thus the copy on the ARM machine does not need it. If it is necessary to conserve space on the ARM machine, compile a copy of the program without the –g option and put that copy on the ARM machine. This copy will not have the debugging information and therefore a bit smaller.

5 Appendix II: Internet Access While on Multiple Networks in Windows

The mounting tutorial asks one to place the SBC on the same network as the Windows machine. This can be as simple as connecting the SBC to router that is currently being used by Windows machine. However, sometimes it is easier to have the Windows machine connect to two different networks at the same time, one that it normally is connected to and a network specifically for communication between the Windows machine and the SBC. Such is the case is when the Windows machines is connected wirelessly to a router for its Internet connection and the router is located relatively far away from the computer. It may be easier to get another router in the same area as the computer and connect the two on this router. However, this could cause a problem with the Windows' machine's Internet connection. When on multiple networks, Windows only tries to find Internet access on the network with the lowest metric or the network that Windows believes to be "closest." The problem is that Windows automatically assigns a lower metric to a wired network as compared to a wireless one and thus the Internet connection on the wireless network is not used.

To deal with this problem, metrics have to be manually set, specifically the network with the Internet connection should be set with a very small metric. To do this, go to the properties of the network device with an Internet connection. Click on TCP/IP (TCP/IPv4 on Vista) in the list and click Properties. Then in the General tab, click Advanced... At the bottom of the IP Settings tab, uncheck Automatic metric and put a small number as the metric (1 is a good choice). Figure 35 shows the window.

Advanced TCP/IP Settings	8 ×
IP Settings DNS WINS	
IP addresses	
IP address	Subnet mask
DHCP Enabled	
	Add <u>E</u> dit Remo <u>v</u> e
Default gateways:	
Gateway	Metric
	Add Edi <u>t</u> Re <u>m</u> ove
Automatic metric	
Interface metric: 1	
	OK Cancel

Figure 35, Setting the metric

After setting this metric, the computer should now have Internet access. In general, the metric should not need to be changed. It is recommended to check Automatic metric when this set up is not being used.