Sourcery G++ Lite

Power GNU/Linux

Sourcery G++ Lite 4.4-194

Getting Started

Sourcery G++ Lite: Power GNU/Linux: Sourcery G++ Lite 4.4-194: Getting Started

CodeSourcery, Inc.

Abstract

This guide explains how to install and build applications with Sourcery G++ Lite, CodeSourcery's customized, validated, and supported version of the GNU Toolchain. Sourcery G++ Lite includes everything you need for application development, including C and C++ compilers, assemblers, linkers, and libraries.

When you have finished reading this guide, you will know how to use Sourcery G++ from the command line.

Table of Contents

Preface iv
1. Intended Audience v
2. Organization v
3. Typographical Conventions v
1. Quick Start 1
1.1. Installation and Set-Up 2
1.2. Configuring Sourcery G++ Lite for the Target System 2
1.3. Building Your Program 2
1.4. Running and Debugging Your Program 2
2. Installation and Configuration 4
2.1. Terminology 5
2.2. System Requirements 5
2.3. Downloading an Installer 6
2.4. Installing Sourcery G++ Lite 6
2.5. Installing Sourcery G++ Lite Updates 9
2.6. Setting up the Environment 10
2.7. Uninstalling Sourcery G++ Lite 11
3. Sourcery G++ Lite for Power GNU/Linux 13
3.1. Included Components and Features 14
3.2. Library Configurations 14
3.3. Target Kernel Requirements 16
3.4. Using Sourcery G++ Lite on GNU/Linux Targets 16
3.5. Using GDB Server for Debugging 19
3.6. Object File Portability 20
3.7. Using OpenMP 20
3.8. Compiler Wrapper for Multiple Targets 20
4. Using Sourcery G++ from the Command Line 22
4.1. Building an Application 23
4.2. Running Applications on the Target System 23
4.3. Running Applications from GDB 24
5. Next Steps with Sourcery G++ 25
5.1. Sourcery G++ Knowledge Base 26
5.2. Manuals for GNU Toolchain Components 26
A. Sourcery G++ Lite Release Notes 27
A.1. Changes in Sourcery G++ Lite for Power GNU/Linux 28
B. Sourcery G++ Lite Licenses 34
B.1. Licenses for Sourcery G++ Lite Components 35
B.2. Sourcery G++ Software License Agreement 36
B.3. Attribution 39

Preface

This preface introduces the Sourcery G++ Lite Getting Started guide. It explains the structure of this guide and describes the documentation conventions used.

1. Intended Audience

This guide is written for people who will install and/or use Sourcery G++ Lite. This guide provides a step-by-step guide to installing Sourcery G++ Lite and to building simple applications. Parts of this document assume that you have some familiarity with using the command-line interface. If you are an administrator installing Sourcery G++ Lite on a UNIX-like system for all of your users to use, you should also be familiar with the package-management software (such as the Red Hat Package Manager) for your system.

2. Organization

This document is organized into the following chapters and appendices:

Chapter 1, "Quick Start"	This chapter includes a brief checklist to follow when installing and using Sourcery G++ Lite for the first time. You may use this chapter as an abbreviated guide to the rest of this manual.
Chapter 2, "Installation and Configuration"	This chapter describes how to download, install and configure Sourcery G++ Lite. This section describes the available installation options and explains how to set up your environment so that you can build applications.
Chapter 3, "Sourcery G++ Lite for Power GNU/Linux"	This chapter contains information about using Sourcery G++ Lite that is specific to Power GNU/Linux targets. You should read this chapter to learn how to best use Sourcery G++ Lite on your target system.
Chapter 4, "Using Sourcery G++ from the Command Line"	This chapter explains how to build applications with Sourcery G++ Lite using the command line. In the process of reading this chapter, you will build a simple application that you can use as a model for your own programs.
Chapter 5, "Next Steps with Sourcery G++"	This chapter describes where you can find additional documentation and information about using Sourcery G++ Lite and its components. It also provides information about Sourcery G++ subscriptions. CodeSourcery customers with Sourcery G++ subscriptions receive comprehensive support for Sourcery G++.
Appendix A, "Sourcery G++ Lite Release Notes"	This appendix contains information about changes in this release of Sourcery G++ Lite for Power GNU/Linux. You should read through these notes to learn about new features and bug fixes.
Appendix B, "Sourcery G++ Lite Licenses"	This appendix provides information about the software licenses that apply to Sourcery G++ Lite. Read this appendix to understand your legal rights and obligations as a user of Sourcery G++ Lite.

3. Typographical Conventions

The following typographical conventions are used in this guide:

`> command arg ...`	A command, typed by the user, and its output. The “>” character is the command prompt.
`command`	The name of a program, when used in a sentence, rather than in literal input or output.
`literal`	Text provided to or received from a computer program.
`placeholder`	Text that should be replaced with an appropriate value when typing a command.
`\`	At the end of a line in command or program examples, indicates that a long line of literal input or output continues onto the next line in the document.

Chapter 1
Quick Start

This chapter includes a brief checklist to follow when installing and using Sourcery G++ Lite for the first time. You may use this chapter as an abbreviated guide to the rest of this manual.

Sourcery G++ Lite for Power GNU/Linux is intended for developers working on embedded GNU/Linux applications. It may also be used for Linux kernel development and debugging, or to build a GNU/Linux distribution.

Follow the steps given in this chapter to install Sourcery G++ Lite and build and run your first application program. The checklist given here is not a tutorial and does not include detailed instructions for each step; however, it will help guide you to find the instructions and reference information you need to accomplish each step. Note that this checklist is also oriented towards application debugging rather than kernel debugging.

You can find additional details about the components, libraries, and other features included in this version of Sourcery G++ Lite in Chapter 3, “Sourcery G++ Lite for Power GNU/Linux”.

1.1. Installation and Set-Up

Install Sourcery G++ Lite on your host computer. You may download an installer package from the Sourcery G++ web site[1], or you may have received an installer on CD. The installer is an executable program that pops up a window on your computer and leads you through a series of dialogs to configure your installation. If the installation is successful, it will offer to launch the Getting Started guide. For more information about installing Sourcery G++ Lite, including host system requirements and tips to set up your environment after installation, refer to Chapter 2, “Installation and Configuration”.

1.2. Configuring Sourcery G++ Lite for the Target System

Identify your target libraries. Sourcery G++ Lite supports libraries optimized for different targets. Libraries are selected automatically by the linker, depending on the processor and other options you have specified. Refer to Section 3.2, “Library Configurations” for details.

Install runtime libraries on your target machine. In order to run programs built with the Sourcery G++ runtime libraries on target hardware, you must install these libraries, the Sourcery G++ dynamic linker, and other runtime support files -- collectively referred to as the *sysroot* -- on your GNU/Linux target system. Typically, this involves either using third-party tools to build a complete root filesystem including the Sourcery G++ sysroot, or using special commands when linking or running your program so it can find the sysroot installed in another location on the target. Refer to Section 3.4, “Using Sourcery G++ Lite on GNU/Linux Targets” for full discussion of these options.

1.3. Building Your Program

Build your program with Sourcery G++ command-line tools. Create a simple test program, and follow the directions in Chapter 4, “Using Sourcery G++ from the Command Line” to compile and link it using Sourcery G++ Lite.

1.4. Running and Debugging Your Program

The steps to run or debug your program depend on your target system and how it is configured. Choose the appropriate method for your target.

[1] http://www.codesourcery.com/gnu_toolchains/

Run your program on the Power GNU/Linux target. To run a program using the included Sourcery G++ libraries, you must install the sysroot on the target, as previously discussed. Copy the executable for your program to the target system. The method you use for launching your program depends on how you have installed the libraries and built your program. In some cases, you may need to invoke the Sourcery G++ dynamic linker explicitly. Refer to Section 3.4, “Using Sourcery G++ Lite on GNU/Linux Targets” for details.

Debug your program on the target using GDB server. You can use GDB server on a remote target to debug your program. When debugging a program that uses the included Sourcery G++ libraries, you must use the `gdbserver` executable included in the sysroot, and similar issues with respect to the dynamic linker as discussed previously apply. See Section 3.5, “Using GDB Server for Debugging” for detailed instructions. Once you have started GDB server on the target, you can connect to it from the debugger on your host system. Refer to Section 4.3, “Running Applications from GDB” for instructions on remote debugging from command-line GDB.

Chapter 2
Installation and Configuration

This chapter explains how to install Sourcery G++ Lite. You will learn how to:

1. Verify that you can install Sourcery G++ Lite on your system.
2. Download the appropriate Sourcery G++ Lite installer.
3. Install Sourcery G++ Lite.
4. Configure your environment so that you can use Sourcery G++ Lite.

2.1. Terminology

Throughout this document, the term *host system* refers to the system on which you run Sourcery G++ while the term *target system* refers to the system on which the code produced by Sourcery G++ runs. The target system for this version of Sourcery G++ is `powerpc-linux-gnu`.

If you are developing a workstation or server application to run on the same system that you are using to run Sourcery G++, then the host and target systems are the same. On the other hand, if you are developing an application for an embedded system, then the host and target systems are probably different.

2.2. System Requirements

2.2.1. Host Operating System Requirements

This version of Sourcery G++ supports the following host operating systems and architectures:

- Microsoft Windows NT 4, Windows 2000, Windows XP, Windows Vista, and Windows 7 systems using IA32, AMD64, and Intel 64 processors.
- GNU/Linux systems using IA32, AMD64, or Intel 64 processors, including Debian 3.1 (and later), Red Hat Enterprise Linux 3 (and later), and SuSE Enterprise Linux 8 (and later).

Sourcery G++ is built as a 32-bit application. Therefore, even when running on a 64-bit host system, Sourcery G++ requires 32-bit host libraries. If these libraries are not already installed on your system, you must install them before installing and using Sourcery G++ Lite. Consult your operating system documentation for more information about obtaining these libraries.

Installing on Ubuntu and Debian GNU/Linux Hosts

The Sourcery G++ graphical installer is incompatible with the `dash` shell, which is the default `/bin/sh` for recent releases of the Ubuntu and Debian GNU/Linux distributions. To install Sourcery G++ Lite on these systems, you must make `/bin/sh` a symbolic link to one of the supported shells: `bash`, `csh`, `tcsh`, `zsh`, or `ksh`.

For example, on Ubuntu systems, the recommended way to do this is:

```
> sudo dpkg-reconfigure -plow dash
Install as /bin/sh? No
```

This is a limitation of the installer and uninstaller only, not of the installed Sourcery G++ Lite toolchain.

2.2.2. Host Hardware Requirements

In order to install and use Sourcery G++ Lite, you must have at least 128MB of available memory.

The amount of disk space required for a complete Sourcery G++ Lite installation directory depends on the host operating system and the number of target libraries included. Typically, you should plan on at least 400MB.

In addition, the graphical installer requires a similar amount of temporary space during the installation process. On Microsoft Windows hosts, the installer uses the location specified by the `TEMP` environment variable for these temporary files. If there is not enough free space on that volume, the installer

prompts for an alternate location. On Linux hosts, the installer puts temporary files in the directory specified by the `IATEMPDIR` environment variable, or `/tmp` if that is not set.

2.2.3. Target System Requirements

See Chapter 3, “Sourcery G++ Lite for Power GNU/Linux” for requirements that apply to the target system.

2.3. Downloading an Installer

If you have received Sourcery G++ Lite on a CD, or other physical media, then you do not need to download an installer. You may skip ahead to Section 2.4, “Installing Sourcery G++ Lite”.

You can download Sourcery G++ Lite from the Sourcery G++ web site[1]. This free version of Sourcery G++, which is made available to the general public, does not include all the functionality of CodeSourcery's product releases. If you prefer, you may instead purchase or register for an evaluation of CodeSourcery's product toolchains at the Sourcery G++ Portal[2].

Once you have navigated to the appropriate web site, download the installer that corresponds to your host operating system. For Microsoft Windows systems, the Sourcery G++ installer is provided as an executable with the `.exe` extension. For GNU/Linux systems Sourcery G++ Lite is provided as an executable installer package with the `.bin` extension. If installing on a RPM-based GNU/Linux system you may download the `.rpm` file. You may also install from a compressed archive with the `.tar.bz2` extension.

On Microsoft Windows systems, save the installer to the desktop. On GNU/Linux systems, save the download package in your home directory.

2.4. Installing Sourcery G++ Lite

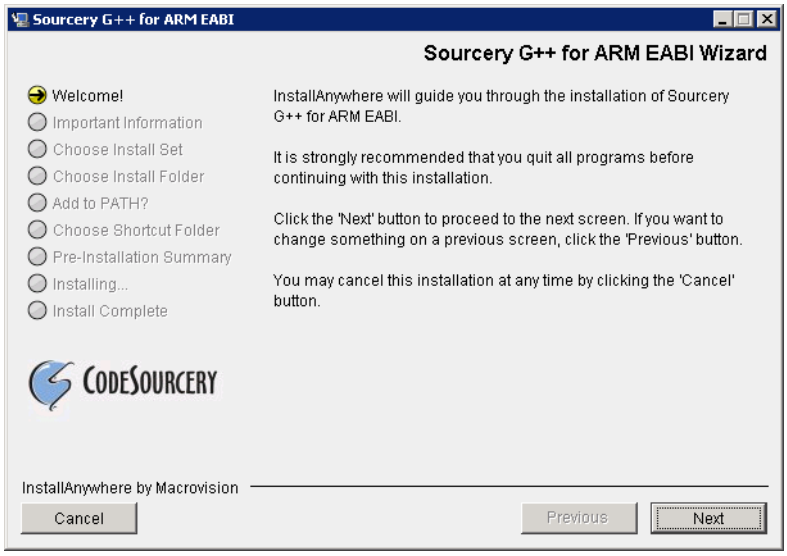
The method used to install Sourcery G++ Lite depends on your host system and the kind of installation package you have downloaded.

2.4.1. Using the Sourcery G++ Lite Installer on Microsoft Windows

If you have received Sourcery G++ Lite on CD, insert the CD in your computer. On most computers, the installer then starts automatically. If your computer has been configured not to automatically run CDs, open `My Computer`, and double click on the CD. If you downloaded Sourcery G++ Lite, double-click on the installer.

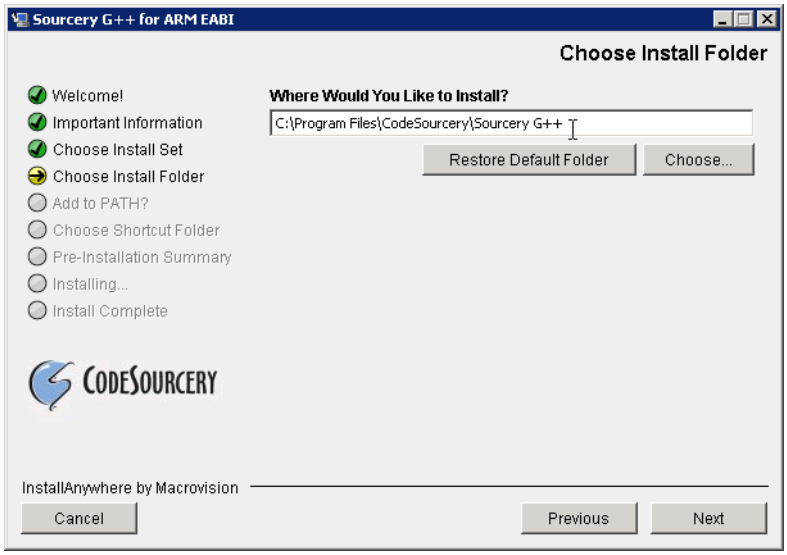
After the installer starts, follow the on-screen dialogs to install Sourcery G++ Lite. The installer is intended to be self-explanatory and on most pages the defaults are appropriate.

[1] http://www.codesourcery.com/gnu_toolchains/
[2] https://support.codesourcery.com/GNUToolchain/

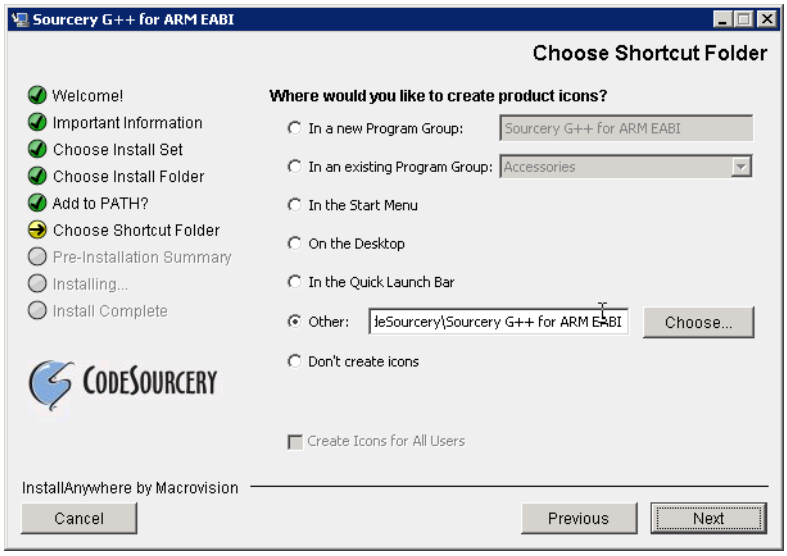


Running the Installer. The graphical installer guides you through the steps to install Sourcery G++ Lite.

You may want to change the install directory pathname and customize the shortcut installation.

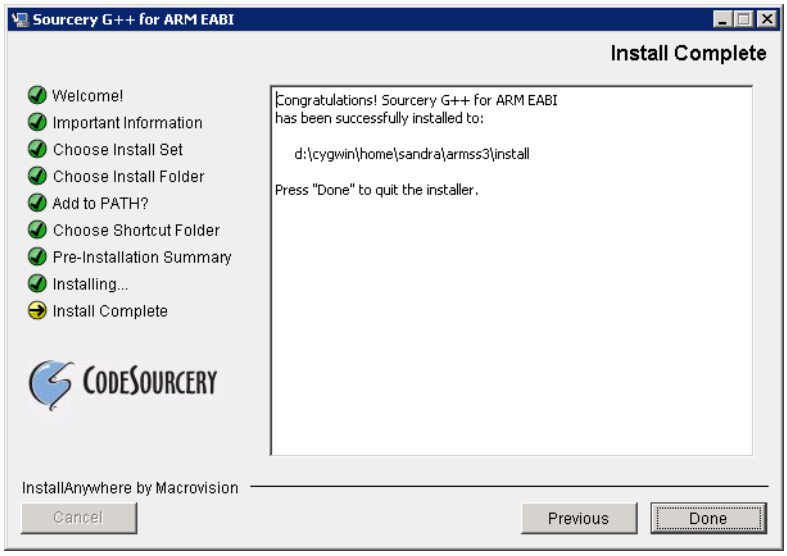


Choose Install Folder. Select the pathname to your install directory.



Choose Shortcut Folder. You can customize where the installer creates shortcuts for quick access to Sourcery G++ Lite.

When the installer has finished, it asks if you want to launch a viewer for the Getting Started guide. Finally, the installer displays a summary screen to confirm a successful install before it exits.



Install Complete. You should see a screen similar to this after a successful install.

If you prefer, you can run the installer in console mode rather than using the graphical interface. To do this, invoke the installer with the `-i console` command-line option. For example:

```
> /path/to/package.exe -i console
```

2.4.2. Using the Sourcery G++ Lite Installer on GNU/Linux Hosts

Start the graphical installer by invoking the executable shell script:

```
> /bin/sh ./path/to/package.bin
```

After the installer starts, follow the on-screen dialogs to install Sourcery G++ Lite. For additional details on running the installer, see the discussion and screen shots in the Microsoft Windows section above.

If you prefer, or if your host system does not run the X Window System, you can run the installer in console mode rather than using the graphical interface. To do this, invoke the installer with the `-i console` command-line option. For example:

```
> /bin/sh ./path/to/package.bin -i console
```

2.4.3. Installing Sourcery G++ Lite on RPM-based GNU/Linux Systems

On a RPM-based system you should use RPM to install the provided package. Execute the following command as root (administrator):

```
> rpm -ivh /path/to/package.rpm
```

To update an existing Sourcery G++ Lite installation, use:

```
> rpm -Uvh /path/to/package.rpm
```

2.4.4. Installing Sourcery G++ Lite from a Compressed Archive

You do not need to be a system administrator to install Sourcery G++ Lite from a compressed archive. You may install Sourcery G++ Lite using any user account and in any directory to which you have write access. This guide assumes that you have decided to install Sourcery G++ Lite in the `$HOME/CodeSourcery` subdirectory of your home directory and that the filename of the package you have downloaded is `/path/to/package.tar.bz2`. After installation the toolchain will be in `$HOME/CodeSourcery/sourceryg++-4.4`.

First, uncompress the package file:

```
> bunzip2 /path/to/package.tar.bz2
```

Next, create the directory in which you wish to install the package:

```
> mkdir -p $HOME/CodeSourcery
```

Change to the installation directory:

```
> cd $HOME/CodeSourcery
```

Unpack the package:

```
> tar xf /path/to/package.tar
```

2.5. Installing Sourcery G++ Lite Updates

If you have already installed an earlier version of Sourcery G++ Lite for Power GNU/Linux on your system, it is not necessary to uninstall it before using the installer to unpack a new version in the same location. The installer detects that it is performing an update in that case.

To update a previous RPM installation of Sourcery G++ Lite, use `rpm -U` instead of `rpm -i`, as described above.

If you are installing an update from a compressed archive, it is recommended that you remove any previous installation in the same location, or install in a different directory.

Note that the names of the Sourcery G++ commands for the Power GNU/Linux target all begin with `powerpc-linux-gnu`. This means that you can install Sourcery G++ for multiple target systems in the same directory without conflicts.

2.6. Setting up the Environment

As with the installation process itself, the steps required to set up your environment depend on your host operating system.

2.6.1. Setting up the Environment on Microsoft Windows Hosts

2.6.1.1. Setting the `PATH`

In order to use the Sourcery G++ tools from the command line, you should add them to your `PATH`. You may skip this step if you used the graphical installer, since the installer automatically adds Sourcery G++ to your `PATH`.

To set the `PATH` on a Microsoft Windows Vista system, use the following command in a `cmd.exe` shell:

```
> setx PATH "%PATH%;C:\Program Files\Sourcery G++\bin"
```

where `C:\Program Files\Sourcery G++` should be changed to the path of your Sourcery G++ Lite installation.

To set the `PATH` on a system running a Microsoft Windows version other than Vista, from the desktop bring up the `Start` menu and right click on `My Computer`. Select `Properties`, go to the `Advanced` tab, then click on the `Environment Variables` button. Select the `PATH` variable and click the `Edit`. Add the string `;C:\Program Files\Sourcery G++\bin` to the end, and click `OK`. Again, you must adjust the pathname to reflect your installation directory.

You can verify that your `PATH` is set up correctly by starting a new `cmd.exe` shell and running:

```
> powerpc-linux-gnu-g++ -v
```

Verify that the last line of the output contains: `Sourcery G++ Lite 4.4-194`.

2.6.1.2. Working with Cygwin

Sourcery G++ Lite does not require Cygwin or any other UNIX emulation environment. You can use Sourcery G++ directly from the Windows command shell. You can also use Sourcery G++ from within the Cygwin environment, if you prefer.

The Cygwin emulation environment translates Windows path names into UNIX path names. For example, the Cygwin path `/home/user/hello.c` corresponds to the Windows path `c:\cygwin\home\user\hello.c`. Because Sourcery G++ is not a Cygwin application, it does not, by default, recognize Cygwin paths.

If you are using Sourcery G++ from Cygwin, you should set the `CYGPATH` environment variable. If this environment variable is set, Sourcery G++ Lite automatically translates Cygwin path names into Windows path names. To set this environment variable, type the following command in a Cygwin shell:

```
> export CYGPATH=cygpath
```

To resolve Cygwin path names, Sourcery G++ relies on the `cygpath` utility provided with Cygwin. You must provide Sourcery G++ with the full path to `cygpath` if `cygpath` is not in your `PATH`. For example:

```
> export CYGPATH=c:/cygwin/bin/cygpath
```

directs Sourcery G++ Lite to use `c:/cygwin/bin/cygpath` as the path conversion utility. The value of `CYGPATH` must be an ordinary Windows path, not a Cygwin path.

2.6.2. Setting up the Environment on GNU/Linux Hosts

If you installed Sourcery G++ Lite using the graphical installer then you may skip this step. The installer does this setup for you.

Before using Sourcery G++ Lite you should add it to your `PATH`. The command you must use varies with the particular command shell that you are using. If you are using the C Shell (`csh` or `tcsh`), use the command:

```
> setenv PATH $HOME/CodeSourcery/Sourcery_G++/bin:$PATH
```

If you are using Bourne Shell (`sh`), the Korn Shell (`ksh`), or another shell, use:

```
> PATH=$HOME/CodeSourcery/Sourcery_G++/bin:$PATH
> export PATH
```

If you are not sure which shell you are using, try both commands. In both cases, if you have installed Sourcery G++ Lite in an alternate location, you must replace the directory above with `bin` subdirectory of the directory in which you installed Sourcery G++ Lite.

You may also wish to set the `MANPATH` environment variable so that you can access the Sourcery G++ manual pages, which provide additional information about using Sourcery G++. To set the `MANPATH` environment variable, follow the same steps shown above, replacing `PATH` with `MANPATH`, and `bin` with `share/doc/sourceryg++-powerpc-linux-gnu/man`.

You can test that your `PATH` is set up correctly by running the following command:

```
> powerpc-linux-gnu-g++ -v
```

Verify that the last line of the output contains: `Sourcery G++ Lite 4.4-194`.

2.7. Uninstalling Sourcery G++ Lite

The method used to uninstall Sourcery G++ Lite depends on the method you originally used to install it. If you have modified any files in the installation it is recommended that you back up these changes. The uninstall procedure may remove the files you have altered. In particular, the `powerpc-linux-gnu` directory located in the install directory will be removed entirely by the uninstaller.

2.7.1. Using the Sourcery G++ Lite Uninstaller on Microsoft Windows

You should use the provided uninstaller to remove a Sourcery G++ Lite installation originally created by the graphical installer. Start the graphical uninstaller by invoking the executable Uninstall execut-

able located in your installation directory, or use the uninstall shortcut created during installation. After the uninstaller starts, follow the on-screen dialogs to uninstall Sourcery G++ Lite.

You can run the uninstaller in console mode, rather than using the graphical interface, by invoking the Uninstall executable found in your Sourcery G++ Lite installation directory with the `-i console` command-line option.

To uninstall third-party drivers bundled with Sourcery G++ Lite, first disconnect the associated hardware device. Then use `Add or Remove Programs` (non-Vista) or `Uninstall a program` (Vista) to remove the drivers separately. Depending on the device, you may need to reboot your computer to complete the driver uninstall.

2.7.2. Using the Sourcery G++ Lite Uninstaller on GNU/Linux

You should use the provided uninstaller to remove a Sourcery G++ Lite installation originally created by the executable installer script. Start the graphical uninstaller by invoking the executable Uninstall shell script located in your installation directory. After the uninstaller starts, follow the on-screen dialogs to uninstall Sourcery G++ Lite.

You can run the uninstaller in console mode, rather than using the graphical interface, by invoking the Uninstall script with the `-i console` command-line option.

2.7.3. Uninstalling Sourcery G++ Lite on RPM-based GNU/Linux Systems

If you installed Sourcery G++ Lite from an RPM package, you should also use RPM to uninstall it. Execute the following command as root (administrator):

```
> rpm -e sourceryg++-powerpc-linux-gnu
```

2.7.4. Uninstalling a Compressed Archive Installation

If you installed Sourcery G++ Lite from a `.tar.bz2` file, you can uninstall it by manually deleting the installation directory created in the install procedure.

Chapter 3
Sourcery G++ Lite for Power GNU/Linux

This chapter contains information about features of Sourcery G++ Lite that are specific to Power GNU/Linux targets. You should read this chapter to learn how to best use Sourcery G++ Lite on your target system.

3.1. Included Components and Features

This section briefly lists the important components and features included in Sourcery G++ Lite for Power GNU/Linux, and tells you where you may find further information about these features.

Component	Version	Notes
GNU programming tools		
GNU Compiler Collection	4.4.1	Separate manual included.
GNU Binary Utilities	2.19.51	Includes assembler, linker, and other utilities. Separate manuals included.
Debugging support and simulators		
GNU Debugger	7.0.50	Separate manual included.
GDB Server	N/A	Included with GDB. See Section 3.5, “Using GDB Server for Debugging”.
Target libraries		
GNU C Library	2.11	Separate manual included.
Linux Kernel Headers	2.6.32	
OpenMP	N/A	
Other utilities		
GNU Make	N/A	Build support on Windows hosts.
GNU Core Utilities	N/A	Build support on Windows hosts.

3.2. Library Configurations

Sourcery G++ includes copies of run-time libraries that have been built with optimizations for different target architecture variants or other sets of build options. Each such set of libraries is referred to as a *multilib*. When you link a target application, Sourcery G++ selects the multilib matching the build options you have selected.

Each multilib corresponds to a *sysroot* directory that contains the files that should be installed on the target system. The sysroot contains the dynamic linker used to run your applications on the target as well as the libraries. Refer to Section 3.4, “Using Sourcery G++ Lite on GNU/Linux Targets” for instructions on how to install and use these support files on your target GNU/Linux system. You can find the sysroot directories provided with Sourcery G++ in the `powerpc-linux-gnu/libc` directory of your installation. In the tables below, the dynamic linker pathname is given relative to the corresponding sysroot.

3.2.1. Included Libraries

The following library configurations are available in Sourcery G++ Lite for Power GNU/Linux.

603 - GLIBC, 32-bit	
Command-line option(s):	default
Sysroot subdirectory:	`./`
Dynamic linker:	`lib/ld.so.1`
Notes:	This multilib is compatible with processor cores that implement hardware floating-point support, such as Freescale's e300c3, 603e, and G2 cores.

603 - Soft-Float, GLIBC, 32-bit	
Command-line option(s):	`-msoft-float`
Sysroot subdirectory:	`nof/`
Dynamic linker:	`lib/ld.so.1`
Notes:	This multilib is compatible with processor cores that do not implement hardware floating-point instructions, such as Freescale's 8XX and e300c2 cores.

e600 (Altivec) - GLIBC, 32-bit	
Command-line option(s):	`-te600`
Sysroot subdirectory:	`te600/`
Dynamic linker:	`lib/ld.so.1`
Notes:	This multilib is compatible with processor cores that implement AltiVec instructions, such as Freescale's e600 core.

e500v1 - GLIBC, 32-bit	
Command-line option(s):	`-te500v1`
Sysroot subdirectory:	`te500v1/`
Dynamic linker:	`lib/ld.so.1`
Notes:	This multilib is compatible with processor cores that implement SPE scalar and vector single-precision floating-point instructions.

e500v2 - GLIBC, 32-bit	
Command-line option(s):	`-te500v2`
Sysroot subdirectory:	`te500v2/`
Dynamic linker:	`lib/ld.so.1`
Notes:	This multilib is compatible with processor cores that implement SPE scalar and vector single-precision floating-point instructions and SPE double-precision floating-point instructions.

e500mc - GLIBC, 32-bit	
Command-line option(s):	`-te500mc`
Sysroot subdirectory:	`te500mc/`
Dynamic linker:	`lib/ld.so.1`
Notes:	This multilib is compatible with Freescale's QorIQ processor cores.

3.2.2. Library Selection

A given multilib may be compatible with additional processors and build options beyond those listed above. However, even if a particular set of command-line options produces code compatible with one of the provided multilibs, those options may not be sufficient to identify the intended library to the linker. For example, on some targets, specifying only a processor option on the command line may imply architecture features or floating-point support for compilation, but not for library selection. The details of the mapping from command-line options to multilibs are target-specific and quite complex. Therefore, it is recommended that your link command line include exactly the options listed in the tables above for your intended target multilib. In some cases, you may need to supply different options for linking than for compilation.

If you are uncertain which multilib is selected by a particular set of command-line options, GCC can tell you if you invoke it with the `-print-multi-directory` option in addition to your other build options. For example:

```
> powerpc-linux-gnu-gcc -print-multi-directory options...
```

The output of this command is a directory name for the multilib, which you can look up in the tables given previously.

3.3. Target Kernel Requirements

The GNU C library supplied with this version of Sourcery G++ Lite requires that Linux kernel version 2.6.10 or later be installed on the target in order to run applications.

3.4. Using Sourcery G++ Lite on GNU/Linux Targets

In order to run and debug programs produced by Sourcery G++ on a GNU/Linux target, you must install runtime support files on the target. You may also need to set appropriate build options so that your executables can find the correct dynamic linker and libraries at runtime.

The runtime support files, referred to as the *sysroot*, are found in the `powerpc-linux-gnu/libc` directory of your Sourcery G++ Lite installation. The sysroot consists of the contents of the `etc`, `lib`, `sbin`, and `usr` directories. There may be other directories in `powerpc-linux-gnu/libc` that contain additional sysroots customized for particular combinations of command-line compiler flags, or *multilibs*. Refer to Section 3.2, “Library Configurations” for a list of the included multilibs in this version of Sourcery G++ Lite, and the corresponding sysroot directory pathnames.

Note for Windows Host Users

The sysroots provided in Windows host packages for Sourcery G++ are not directly usable on the GNU/Linux target because of differences between the Windows and GNU/Linux file systems. Some files that are hard links, or copies, in the sysroot as installed on the Windows file system should be symbolic links on the GNU/Linux target. Additionally, some files in the sysroot which should be marked executable on the GNU/Linux target are not marked executable on Windows. If you intend to use the sysroot provided with Sourcery G++ on a Windows host system as the basis for your GNU/Linux target filesystem, you must correct these issues after copying the sysroot to the target.

You have these choices for installing the sysroot on the target:

- You can install the files in the filesystem root on the target (that is, installing the files directly in `/etc/`, `/lib/`, and so on). All applications on the target then automatically use the Sourcery G++ libraries. This method is primarily useful when you are building a GNU/Linux root filesystem from scratch. If your target board already has a GNU/Linux filesystem installed, overwriting the existing C library files is not recommended, as this may break other applications on your system, or cause it to fail to boot.

- You can install the sysroot in an alternate location and build your application with the `-rpath` and `--dynamic-linker` linker options to specify the sysroot location.

- You can install the sysroot in an alternate location and explicitly invoke your application through the dynamic linker to specify the sysroot location. If you are just getting started with Sourcery G++ Lite, this may be the easiest way to get your application running, but this method does not support use of the debugger.

Setting the environment variable `LD_LIBRARY_PATH` on the target is not sufficient, since executables produced by Sourcery G++ depend on the Sourcery G++ dynamic linker included in the sysroot as well as the Sourcery G++ runtime libraries.

3.4.1. Installing the Sysroot

If you are modifying an existing system, rather than creating a new system from scratch, you should place the sysroot files in a new directory, rather than in the root directory of your target system.

If you choose to overwrite your existing C library, you may not be able to boot your system. You should back up your existing system before overwriting the C library and ensure that you can restore the backup even with your system offline.

When running Sourcery G++ on a GNU/Linux host, you have the alternative of installing the sysroot on the target at the same pathname where it is installed on the host system. One way to accomplish this is to NFS-mount the installation directory on both machines in the same location, rather than to copy files.

In many cases, you do not need to copy all of the files in the sysroot. For example, the `usr/include` subdirectory contains files that are only needed if you will actually be running the compiler on your target system. You do not need these files for non-native compilers. You also do not need any `.o` or `.a` files; these are used by the compiler when linking programs, but are not needed to run programs. You should definitely copy all `.so` files and the executable files in `usr/bin` and `sbin`.

You need to install the sysroot(s) corresponding to the compiler options you are using for your applications. The tables in Section 3.2, “Library Configurations” tell you which sysroot directories correspond to which compiler options. If you are unsure what sysroot is being referenced when you build your program, you can identify the sysroot by adding `-v` to your compiler command-line options, and looking at the `--sysroot=` pathname in the compiler output.

3.4.2. Using Linker Options to Specify the Sysroot Location

If you have installed the sysroot on the target in a location other than the file system root, you can use the `-rpath` and `--dynamic-linker` linker options to specify the sysroot location.

If you are using Sourcery G++ from the command line, follow these steps:

1. First find the correct sysroot directory, dynamic linker, and library subdirectory for your selected multilib. Refer to Section 3.2, “Library Configurations”. In the following steps, *`sysroot`* is the absolute path to the sysroot directory on the target corresponding to your selected multilib. For

the default multilib, the dynamic linker path relative to the sysroot is `lib/ld.so.1`, and the library subdirectory is `lib`. This is used in the example below.

2. When invoking `powerpc-linux-gnu-gcc` to link your executable, include the command-line options:

```
-Wl,-rpath=sysroot/lib:sysroot/usr/lib \
-Wl,--dynamic-linker=sysroot/lib/ld.so.1
```

   where *`sysroot`* is the absolute path to the sysroot directory on the target corresponding to your selected multilib.

3. Copy the executable to the target and execute it normally.

Note that if you specify an incorrect path for `--dynamic-linker`, the common failure mode seen when running your application on the target is similar to

```
> ./factorial
./factorial: No such file or directory
```

or

```
> ./factorial
./factorial: bad ELF interpreter: No such file or directory
```

This can be quite confusing since it appears from the error message as if it is the `./factorial` executable that is missing rather than the dynamic linker it references.

3.4.3. Specifying the Sysroot Location at Runtime

You can invoke the Sourcery G++ dynamic linker on the target to run your application without having to compile it with specific linker options.

To do this, follow these steps:

1. Build your application on the host, without any additional linker options, and copy the executable to your target system.

2. Find the correct sysroot directory, dynamic linker, and library subdirectory for your selected multilib. Refer to Section 3.2, “Library Configurations”. In the following steps, *`sysroot`* is the absolute path to the sysroot directory on the target corresponding to your selected multilib. For the default multilib, the dynamic linker is `lib/ld.so.1`, and the library subdirectory is `lib`. This is used in the example below.

3. On the target system, invoke the dynamic linker with your executable as:

```
> sysroot/lib/ld.so.1 \
  --library-path sysroot/lib:sysroot/usr/lib \
  /path/to/your-executable
```

   where *`sysroot`* is the absolute path to the sysroot directory on the target corresponding to your selected multilib.

   Invoking the linker in this manner requires that you provide either an absolute pathname to your executable, or a relative pathname prefixed with `./`. Specifying only the name of a file in the current directory does not work.

3.5. Using GDB Server for Debugging

The GDB server utility provided with Sourcery G++ Lite can be used to debug a GNU/Linux application. While Sourcery G++ runs on your host system, `gdbserver` and the target application run on your target system. Even though Sourcery G++ and your application run on different systems, the debugging experience when using `gdbserver` is very similar to debugging a native application.

3.5.1. Running GDB Server

The GDB server executables are included in the sysroot in ABI-specific subdirectories of *`sysroot`*`/usr`. Use the executable from the sysroot and library subdirectory that match your program. See Section 3.2, “Library Configurations” for details.

You must copy the sysroot to your target system as described in Section 3.4.1, “Installing the Sysroot”. You must also copy the executable you want to debug to your target system.

If you have installed the sysroot in the root directory of the filesystem on the target, you can invoke `gdbserver` as:

```
> gdbserver :10000 program arg1 arg2 ...
```

where *`program`* is the path to the program you want to debug and *`arg1 arg2 ...`* are the arguments you want to pass to it. The `:10000` argument indicates that `gdbserver` should listen for connections from GDB on port 10000. You can use a different port, if you prefer.

If you have installed the sysroot in an alternate directory, invoking `gdbserver` becomes more complicated. You must build your application using the link-time options to specify the location of the sysroot, as described in Section 3.4.2, “Using Linker Options to Specify the Sysroot Location”. You must also invoke `gdbserver` itself using the dynamic linker provided in the Sourcery G++ sysroot, as described in Section 3.4.3, “Specifying the Sysroot Location at Runtime”. In other words, the command to invoke `gdbserver` in this case would be similar to:

```
> sysroot/lib/ld.so.1 \
  --library-path sysroot/lib:sysroot/usr/lib \
  sysroot/usr/lib/bin/gdbserver :10000 program arg1 arg2 ...
```

3.5.2. Connecting to GDB Server from the Debugger

You can connect to GDB server by using the following command from within GDB:

```
(gdb) target remote target:10000
```

where *`target`* is the host name or IP address of your target system.

When your program exits, `gdbserver` exits too. If you want to debug the program again, you must restart `gdbserver` on the target. Then, in GDB, reissue the `target` command shown above.

3.5.3. Setting the Sysroot in the Debugger

In order to debug shared libraries, GDB needs to map the pathnames of shared libraries on the target to the pathnames of equivalent files on the host system. Debugging of multi-threaded applications also depends on correctly locating copies of the libraries provided in the sysroot on the host system.

In some situations, the target pathnames are valid on the host system. Otherwise, you must tell GDB how to map target pathnames onto the equivalent host pathnames.

In the general case, there are two GDB commands required to set up the mapping:

```
(gdb) set sysroot-on-target target-pathname
(gdb) set sysroot host-pathname
```

This causes GDB to replace all instances of the *target-pathname* prefix in shared library pathnames reported by the target with *host-pathname* to get the location of the equivalent library on the host.

If you have installed the sysroot in the root filesystem on the target, you can omit the `set sysroot-on-target` command, and use only `set sysroot` to specify the location on the host system.

Refer to Section 3.4.1, "Installing the Sysroot" for more information about installing the sysroot on the target. Note that if you have installed a stripped copy of the provided libraries on the target, you should give GDB the location of an unstripped copy on the host.

3.6. Object File Portability

It is possible to create object files using Sourcery G++ for Power ELF or EABI that are link-compatible with the GNU C library provided with Sourcery G++ for Power GNU/Linux as well as with the CodeSourcery C Library or Newlib C Library provided with Power bare-metal toolchains. Currently this is only supported when compiling and linking for the e500mc (with `-te500mc`).

To use this feature, when compiling your files with the bare-metal Power ELF or EABI toolchain define the preprocessor constant `_AEABI_PORTABILITY_LEVEL` to 1 before including any system header files. For example, pass the option `-D_AEABI_PORTABILITY_LEVEL=1` on your compilation command line. No special options are required when linking the resulting object files. When building applications for Power ELF or EABI, files compiled with this definition may be linked freely with those compiled without it.

Files compiled in this manner may not use the functions `fgetpos` or `fsetpos`, or reference the type `fpos_t`. This is because Newlib assumes a representation for `fpos_t` that is not AEABI-compliant.

Note that object files are only portable from bare-metal toolchains to GNU/Linux, and not vice versa; object files compiled for Power GNU/Linux targets cannot be linked into Power ELF or EABI executables.

3.7. Using OpenMP

Sourcery G++ Lite for Power GNU/Linux includes the GNU OpenMP library (libgomp). This is an API that supports multi-platform shared-memory parallel programming.

To compile programs that use OpenMP features, use the `-fopenmp` command-line option. For more information about OpenMP, see `http://www.openmp.org/`.

3.8. Compiler Wrapper for Multiple Targets

Sourcery G++ Lite for ELF and GNU/Linux targets includes a compiler wrapper that enables you to select the Power target via a command-line argument instead of by invoking a different compiler executable.

Using the compiler wrapper from the command line is similar to using the compiler. Instead of:

```
> powerpc-linux-gnu-gcc arguments...
```

you can use the command:

```
> powerpc-unified-gcc --target=powerpc-linux-gnu arguments...
```

If you also have Sourcery G++ for Power ELF installed, you can use the wrapper to compile for that target by specifying the corresponding `--target` option:

```
> powerpc-unified-gcc --target=powerpc-elf arguments...
```

Note that the directory where `powerpc-elf-gcc` is installed must be present in your `PATH` environment variable. You can use the command:

```
> powerpc-unified-gcc --help
```

to determine what the supported values for the `--target` option are.

Please refer to Chapter 4, “Using Sourcery G++ from the Command Line” for more information on building applications with Sourcery G++.

Chapter 4
Using Sourcery G++ from the Command Line

This chapter demonstrates the use of Sourcery G++ Lite from the command line.

4.1. Building an Application

This chapter explains how to build an application with Sourcery G++ Lite using the command line. As elsewhere in this manual, this section assumes that your target system is powerpc-linux-gnu, as indicated by the `powerpc-linux-gnu` command prefix.

Using an editor (such as `notepad` on Microsoft Windows or `vi` on UNIX-like systems), create a file named `main.c` containing the following simple factorial program:

```
#include <stdio.h>

int factorial(int n) {
  if (n == 0)
    return 1;
  return n * factorial (n - 1);
}

int main () {
  int i;
  int n;
  for (i = 0; i < 10; ++i) {
    n = factorial (i);
    printf ("factorial(%d) = %d\n", i, n);
  }
  return 0;
}
```

Compile and link this program using the command:

```
> powerpc-linux-gnu-gcc -o factorial main.c
```

There should be no output from the compiler. (If you are building a C++ application, instead of a C application, replace `powerpc-linux-gnu-gcc` with `powerpc-linux-gnu-g++`.)

4.2. Running Applications on the Target System

You may need to install the Sourcery G++ runtime libraries and dynamic linker on the target system before you can run your application. Refer to Chapter 3, “Sourcery G++ Lite for Power GNU/Linux” for specific instructions.

To run your program on a GNU/Linux target system, use the command:

```
> factorial
```

You should see:

```
factorial(0) = 1
factorial(1) = 1
factorial(2) = 2
factorial(3) = 6
factorial(4) = 24
factorial(5) = 120
factorial(6) = 720
factorial(7) = 5040
```

```
factorial(8) = 40320
factorial(9) = 362880
```

4.3. Running Applications from GDB

You can run GDB, the GNU Debugger, on your host system to debug programs running remotely on a target board or system.

When starting GDB, give it the pathname to the program you want to debug as a command-line argument. For example, if you have built the factorial program as described in Section 4.1, “Building an Application”, enter:

```
> powerpc-linux-gnu-gdb factorial
```

While this section explains the alternatives for using GDB to run and debug application programs, explaining the use of the GDB command-line interface is beyond the scope of this document. Please refer to the GDB manual for further instructions.

4.3.1. Connecting to an External GDB Server

Sourcery G++ Lite includes a program called `gdbserver` that can be used to debug a program running on a remote Power GNU/Linux target. Follow the instructions in Chapter 3, “Sourcery G++ Lite for Power GNU/Linux” to install and run `gdbserver` on your target system.

From within GDB, you can connect to a running `gdbserver` or other debugging stub that uses the GDB remote protocol using:

```
(gdb) target remote host:port
```

where *`host`* is the host name or IP address of the machine the stub is running on, and *`port`* is the port number it is listening on for TCP connections.

Chapter 5
Next Steps with Sourcery G++

This chapter describes where you can find additional documentation and information about using Sourcery G++ Lite and its components.

5.1. Sourcery G++ Knowledge Base

The Sourcery G++ Knowledge Base is available to registered users at the Sourcery G++ Portal[1]. Here you can find solutions to common problems including installing Sourcery G++, making it work with specific targets, and interoperability with third-party libraries. There are also additional example programs and tips for making the most effective use of the toolchain and for solving problems commonly encountered during debugging. The Knowledge Base is updated frequently with additional entries based on inquiries and feedback from customers.

5.2. Manuals for GNU Toolchain Components

Sourcery G++ Lite includes the full user manuals for each of the GNU toolchain components, such as the compiler, linker, assembler, and debugger. Most of the manuals include tutorial material for new users as well as serving as a complete reference for command-line options, supported extensions, and the like.

When you install Sourcery G++ Lite, links to both the PDF and HTML versions of the manuals are created in the shortcuts folder you select. If you elected not to create shortcuts when installing Sourcery G++ Lite, the documentation can be found in the `share/doc/sourceryg++-powerpc-linux-gnu/` subdirectory of your installation directory.

In addition to the detailed reference manuals, Sourcery G++ Lite includes a Unix-style manual page for each toolchain component. You can view these by invoking the `man` command with the pathname of the file you want to view. For example, you can first go to the directory containing the man pages:

```
> cd $INSTALL/share/doc/sourceryg++-powerpc-linux-gnu/man/man1
```

Then you can invoke `man` as:

```
> man ./powerpc-linux-gnu-gcc.1
```

Alternatively, if you use `man` regularly, you'll probably find it more convenient to add the directory containing the Sourcery G++ man pages to your `MANPATH` environment variable. This should go in your `.profile` or equivalent shell startup file; see Section 2.6, “Setting up the Environment” for instructions. Then you can invoke `man` with just the command name rather than a pathname.

Finally, note that every command-line utility program included with Sourcery G++ Lite can be invoked with a `--help` option. This prints a brief description of the arguments and options to the program and exits without doing further processing.

[1] https://support.codesourcery.com/GNUToolchain/

Appendix A
Sourcery G++ Lite Release Notes

This appendix contains information about changes in this release of Sourcery G++ Lite for Power GNU/Linux. You should read through these notes to learn about new features and bug fixes.

A.1. Changes in Sourcery G++ Lite for Power GNU/Linux

This section documents Sourcery G++ Lite changes for each released revision.

A.1.1. Changes in Sourcery G++ Lite 4.4-194

Optimized E500 software floating-point functions. Sourcery G++ Lite now provides double-precision floating-point arithmetic libraries that are optimized for Freescale CPUs containing E500 cores, such as the 8540 and the 8548. To use the optimized libraries in your application, link with `-te500v1` or `-te500v2`.

Improved code generation for `if` statements. The compiler can now generate better code for `if` statements when the then and else clauses contain similar code.

IBM PowerPC 476 support. Sourcery G++ now includes support for IBM PowerPC 476 processors. To compile for these processors, use `-mcpu=476fp` for processors with hardware floating-point support and `-mcpu=476` for processors without hardware floating-point support.

Linker bug fix for `--section-start`. A linker bug that caused `--section-start` to fail to work as documented if the section is defined in multiple object files has been fixed.

Optimized math functions for E500 cores. Optimized implementations of `log10`, `atan`, `scalbn`, `modf`, `log`, `ceil`, `abs`, `floor`, and `atan2` have been added for Freescale CPUs containing E500 cores, such as the 8540 and the 8548.

GCC internal compiler error. A bug has been fixed that caused GCC to crash when compiling some C++ code using templates at `-O2` or `-O3`.

GCC internal compiler error with `optimize` attribute. A bug has been fixed that caused the compiler to crash when invoked with the `-O0` or `-O1` option on code using the `optimize` attribute to specify higher optimization levels for individual functions.

Internal compiler error fix. A bug has been fixed that caused the compiler to crash after issuing a warning `function called through a non-compatible type`. Such code has undefined behavior at runtime, but the compiler no longer crashes while processing it.

A.1.2. Changes in Sourcery G++ Lite 4.4-145

Debugging preprocessed source code. A compiler bug has been fixed that caused debug output to erroneously contain the name of the intermediate preprocessed file.

GDB update. The included version of GDB has been updated to 7.0.50.20100218. This update adds numerous bug fixes and new features, including improved C++ language support, automatic caching of stack memory, and Position Independent Executable (PIE) support.

Static constructor and destructor ordering fixes. The linker now correctly ensures that static destructors with priorities are executed after destructors without priorities. Another linker bug that caused incorrect static constructor and destructor ordering with partial linking involved has been fixed.

`-mblock-move-inline-limit` option added. The compiler now supports the `-mblock-move-inline-limit` option. This option enables you to specify the maximum size of block moves (such as calls to `memcpy` or structure copies) that should be inlined.

Code size with `-g`. A bug that caused binary code size regressions in GCC 4.4 when compiling with `-g` has been fixed.

Optimizer bug fix. A bug in GCC that caused internal compiler errors at `-O2` or above has been fixed. The bug also occurred at other optimization levels when the `-fpromote-loop-indices` command-line option was used.

Shorter function prologues with `-Os -fPIC`. GCC now generates shorter code sequences for function prologues when compiling with the options `-Os -fPIC`. The shorter code sequences enable U-Boot to be compiled successfully.

Improved use of `isel` instruction. When compiling for Freescale's E500 and QorIQ processors, the compiler now uses the `isel` instruction in sequences where the `mfcr` instruction was previously used. This change permits the compiler to better schedule instructions for improved performance.

EGLIBC version 2.11. Sourcery G++ Lite for Power GNU/Linux now includes EGLIBC version 2.11 library which is based on GNU C Library version 2.11. For more information about changes, see `http://www.eglibc.org/news#eglibc_2_11`.

Optimized library functions for E500 cores. Optimized implementations of `memset`, `bzero`, `strlen`, `strcmp`, and `strcpy` have been added for Freescale CPUs containing E500 cores, such as the 8540 and the 8548.

GDB asynchronous mode fix. GDB can now be used from the command line in asynchronous mode with remote targets. Previously, GDB did not accept user input while asynchronous commands (such as `continue &`) were running.

Frame manipulation bug fix. A bug in GDB has been fixed that caused frame manipulation commands to report an internal error in some cases when used on arbitrary stack frames specified by an address.

Read watchpoints bug fix. A GDB bug has been fixed that caused watchpoints set to trigger on memory reads to be silently ignored in some cases.

Improved disassembly of SPE instructions. GDB now supports disassembling SPE instructions when using the `disassemble` command.

Setting thread-specific breakpoints in GDB. A bug in GDB has been fixed that caused a syntax error for the `break *`*`expression`* `thread` *`threadnum`* command.

Improved assembler error checking. The assembler has been improved to perform additional checks for invalid inputs.

A.1.3. Changes in Sourcery G++ Lite 4.4-78

Incorrect symbol addresses bug fix. A bug in the linker that caused it to assign incorrect addresses to symbols has been fixed. The bug occurred when the input objects contained sections not explicitly mentioned in the linker script and was most likely to occur when building the Linux kernel.

`@`*`FILE`* fix. A bug has been fixed in the processing of `@`*`FILE`* command-line options by GCC, GDB, and other tools. The bug caused any options in *`FILE`* following a blank line to be ignored.

GDB interrupt handling bug fix. A bug in GDB has been fixed that caused it to sometimes fail to indicate that the target had stopped after being interrupted. The bug affected clients using GDB's MI front end.

ELF file corruption with `strip`. A bug that caused `strip` to corrupt unusual ELF files has been fixed.

GDB support for Cygwin pathnames. A bug in GDB's translation of Cygwin pathnames has been fixed.

GDB and programs linked with the `--gc-sections` linker option. GDB has been improved to better handle debug information found in programs and libraries linked with the `--gc-sections` option. GDB formerly selected the wrong debug information in some cases, resulting in incorrect behavior when stepping over a function or displaying local variables, for example.

GDB memory find bug fix. A bug in GDB's `find` command has been fixed. The bug caused searches on large memory areas to fail or report matches at incorrect addresses.

Static variables and `asm` statements bug fix. A bug in GCC that caused functions containing static variables and `asm` statements to be miscompiled at `-O2` or above has been fixed. The bug also occurred at other optimization levels when the `-fremove-local-statics` command-line option was used.

Optimizer bug fix. A bug in GCC that caused functions with complex loop nests to be miscompiled at `-O2` or above has been fixed. The bug also occurred at other optimization levels when the `-fpromote-loop-indices` command-line option was used.

GCC internal compiler error. A bug has been fixed that caused the compiler to crash when optimizing code that casts between structure types and the type of the first field.

ELF Program Headers. The linker now better diagnoses errors in the usage of `FILEHDR` and `PHDRS` keywords in `PHDRS` command of linker scripts. Refer to the linker manual for more information.

`gdbserver` bug fix. A bug has been fixed that caused `gdbserver` to crash when debugging programs using thread-local storage without other multi-threading features.

Preprocessor error handling. The preprocessor now treats failing to find a file referenced via `#include` as a fatal error.

`gdbserver` multi-threaded debugging fix. A bug has been fixed that prevented `gdbserver` from exiting after debugging a multi-threaded program.

A.1.4. Changes in Sourcery G++ Lite 4.4-16

Linux kernel headers update. Linux kernel header files have been updated to version 2.6.30.

Optimizer improvements. When optimizing for speed, the compiler now uses improved heuristics to limit certain types of optimizations that may adversely affect both code size and speed. This change also makes it possible to produce better code when optimizing for space rather than speed.

GDB `finish` internal error. A bug has been fixed that caused a GDB internal error when using the `finish` command. The bug occurred when debugging optimized code.

GDB update. The included version of GDB has been updated to 6.8.50.20090630. This update adds numerous bug fixes and new features, including support for multi-byte and wide character sets and improved C++ template support.

GDB and third-party compilers. Some bugs that caused GDB to crash when debugging programs compiled with third-party tools have been fixed. These bugs did not affect programs built with Sourcery G++.

Remote debugging hardware watchpoint bug fix. A GDB bug has been fixed that caused hardware watchpoint hits to be incorrectly reported in some cases.

GDB internal warning fix. A GDB bug has been fixed that caused warnings of the form `warning: (Internal error: pc` *`address`* `in read in psymtab, but not in symtab.)`.

Binutils update. The binutils package has been updated to version 2.19.51.20090709 from the FSF trunk. This update includes numerous bug fixes.

Installer fails during upgrade. The Sourcery G++ installer for Microsoft Windows hosts could fail during an upgrade while waiting for the previous version to be uninstalled. This bug has been fixed.

Uninstaller removed by upgrade. The uninstaller could be incorrectly deleted during an upgrade on Microsoft Windows hosts. This bug has been fixed.

Remote debugging connection auto-retry. The `target remote` command within GDB now uses a configurable auto-retry timeout when establishing TCP connections. This is useful in avoiding race conditions when the remote GDB stub or GDB server is launched simultaneously with GDB. The auto-retry behavior is enabled by default; refer to the GDB manual for details.

Extraneous linker error messages. A linker bug that caused extraneous error messages of the form `Dwarf Error: Offset (507) greater than or equal to .debug_str size (421).` has been corrected. This bug did not affect the correctness of output binaries.

Register variable corruption. A compiler bug has been fixed that caused incorrect code to be generated when the frame pointer or other special-use registers are used as explicit local register variables, introduced via the `asm` keyword on their declarations.

Startup code debugging fixes. Two GDB bugs have been fixed that caused errors when debugging startup code. One bug caused an internal error message; the other caused the error `Cannot find bounds of current function`.

`powerpc-linux-gnu-objcopy` bug fix. A bug has been fixed that caused `powerpc-linux-gnu-objcopy` to issue an error when generating output in the Intel HEX format and using `--change-section-lma` to change section addresses.

Linker script search path. The bug in the linker has been fixed that caused it not to follow its documented behavior for searching for linker scripts named with the `-T` option. Now scripts are looked up first in the current directory, then in library directories specified with `-L` command-line options, and finally in the default system linker script directory.

`tlbilx` encoding fix. An assembler bug that resulted in an incorrect encoding of the `tlbilx` instruction and its extended mnemonics `tlbilxlpid`, `tlbilxpid`, and `tlbilxva` has been fixed.

`-fremove-local-statics` optimization. The `-fremove-local-statics` optimization is now enabled by default at `-O2` and higher optimization levels.

Errors when inserting breakpoints. A GDB bug has been fixed that caused errors of the form `` ``*`function`*`' found in `*`filename`*` psymtab but not in symtab `` when setting a

breakpoint on *function*. This error commonly occurred when setting breakpoints on functions provided by the C library.

Install directory pathnames. Bugs in the install and uninstall scripts for Linux hosts that caused errors or incorrect behavior when the Sourcery G++ install directory pathname contains whitespace characters have been fixed.

Elimination of spurious warnings about `NULL`. The C++ compiler no longer issues spurious warnings about comparisons between pointers to members and `NULL`.

Vectorizer improvements. The compiler now generates improved code for accesses to static nested array variables (e.g. `static int foo[8][8];`).

EGLIBC version 2.10. Sourcery G++ Lite for Power GNU/Linux now includes EGLIBC version 2.10 library which is based on GNU C Library version 2.10. For more information about changes, see `http://www.eglibc.org/news#eglibc_2_10`.

Binutils update. The binutils package has been updated to version 2.19.51.20090205 from the FSF trunk. This update includes numerous bug fixes.

GDB `quit` error. A bug in GDB has been fixed that caused `quit` to report `Quitting: You can't do that without a process to debug.` when debugging a core dump file.

GCC version 4.4.1. Sourcery G++ Lite for Power GNU/Linux is now based on GCC version 4.4.1. For more information about changes from GCC version 4.3 that was included in previous releases, see `http://gcc.gnu.org/gcc-4.4/changes.html`.

Linker map address sorting. The map generated by the linker `-Map` option now lists symbols sorted by address.

GDB update. The included version of GDB has been updated to 6.8.50.20081022. This update includes numerous bug fixes.

A.1.5. Changes in Sourcery G++ Lite 4.3-74

A.1.6. Changes in Sourcery G++ Lite 4.3-71

GDB segment warning. Some compilers produce binaries including uninitialized data regions, such as the stack and heap. GDB incorrectly displayed the warning `Loadable segment "`*name*`" outside of ELF segments` for such binaries; the warning has now been fixed.

A.1.7. Changes in Sourcery G++ Lite 4.3-50

Printing casted values in GDB. A GDB bug that caused incorrect output for expressions containing casts, such as in the `print *(Type *)ptr` command, has been fixed.

Bug fix for objcopy/strip. An objcopy bug that corrupted COMDAT groups when creating new binaries has been fixed. This bug also affected `strip -g`.

Bug fix for assembly listing. A bug that caused the assembler to produce corrupted listings (via the `-a` option) on Windows hosts has been fixed.

Binutils support for DWARF Version 3. The `addr2line` command now supports binaries containing DWARF 3 debugging information. The `ld` command can display error messages with source locations for input files containing DWARF 3 debugging information.

GDB display of source. A bug has been fixed that prevented GDB from locating debug information in some cases. The debugger failed to display source code for or step into the affected functions.

Connecting to the target using a pipe. A bug in GDB's `target remote | program` command has been fixed. When launching the specified *`program`* failed, the bug caused GDB to crash, hang, or give a message `Error: No Error`.

Compiling dynamic libraries with `-Os`. A bug has been fixed that affected dynamic libraries compiled with `-Os`. The bug caused runtime errors such as segmentation faults in applications using the library, as a result of incorrect linker attributes on internal compiler-generated symbols in the library. You must rebuild both the affected shared libraries and the applications that use them to pick up this fix.

Linker script option syntax. GCC now accepts `-T` *`script`* (with whitespace before the *`script`*) as well as `-T`*`script`* (with no whitespace) to specify a linker script on the command line.

Errors after loading the debugged program. An intermittent GDB bug has been fixed. The bug could cause a GDB internal error after the `load` command.

Persistent remote server connections. A GDB bug has been fixed that caused the `target extended-remote` command to fail to tell the remote server to make the connection persistent across program invocations.

A.1.8. Changes in Sourcery G++ Lite 4.3-8

GDB update. The included version of GDB has been updated to 6.8.50.20080821. This update adds numerous bug fixes and new features, including support for decimal floating point, the new `find` command to search memory, the new `/m` (mixed source and assembly) option to the `disassemble` command, and the new `macro define` command to define C preprocessor macros interactively.

Remote debugging improvements. The `gdbserver` utility now supports a more efficient communications protocol that can reduce latency during remote debugging. The protocol optimizations are enabled automatically when `gdbserver` operates over a TCP connection. Refer to the GDB manual for more information.

Output files removed on error. When GCC encounters an error, it now consistently removes any incomplete output files that it may have created.

GCC version 4.3.2. Sourcery G++ Lite for Power GNU/Linux is now based on GCC version 4.3.2. For more information about changes from GCC version 4.2 that was included in previous releases, see `http://gcc.gnu.org/gcc-4.3/changes.html`.

OpenMP support. Support for the OpenMP application programming interface has been added. To compile programs that use OpenMP features, use the `-fopenmp` command-line option. For more information about OpenMP, see `http://www.openmp.org/`.

Bug fix for objdump on Windows. An objdump bug that caused the `-S` option not to work on Windows in some cases has been fixed.

A.1.9. Changes in Older Releases

For information about changes in older releases of Sourcery G++ Lite for Power GNU/Linux, please refer to the Getting Started guide packaged with those releases.

Appendix B
Sourcery G++ Lite Licenses

Sourcery G++ Lite contains software provided under a variety of licenses. Some components are “free” or “open source” software, while other components are proprietary. This appendix explains what licenses apply to your use of Sourcery G++ Lite. You should read this appendix to understand your legal rights and obligations as a user of Sourcery G++ Lite.

B.1. Licenses for Sourcery G++ Lite Components

The table below lists the major components of Sourcery G++ Lite for Power GNU/Linux and the license terms which apply to each of these components.

Some free or open-source components provide documentation or other files under terms different from those shown below. For definitive information about the license that applies to each component, consult the source package corresponding to this release of Sourcery G++ Lite. Sourcery G++ Lite may contain free or open-source components not included in the list below; for a definitive list, consult the source package corresponding to this release of Sourcery G++ Lite.

Component	**License**
GNU Compiler Collection	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
GNU Binary Utilities	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
GNU Debugger	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
GNU C Library	GNU Lesser General Public License 2.1 http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html
Linux Kernel Headers	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
GNU Make	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
GNU Core Utilities	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

The CodeSourcery License is available in Section B.2, “Sourcery G++ Software License Agreement”.

Important

Although some of the licenses that apply to Sourcery G++ Lite are “free software” or “open source software” licenses, none of these licenses impose any obligation on you to reveal the source code of applications you build with Sourcery G++ Lite. You can develop proprietary applications and libraries with Sourcery G++ Lite.

Sourcery G++ Lite may include some third party example programs and libraries in the `share/sourceryg++-powerpc-linux-gnu-examples` subdirectory. These examples are not covered by the Sourcery G++ Software License Agreement. To the extent permitted by law, these examples are provided by CodeSourcery as is with no warranty of any kind, including implied warranties of merchantability or fitness for a particular purpose. Your use of each example is governed by the license notice (if any) it contains.

B.2. Sourcery G++™ Software License Agreement

1. **Parties.** The parties to this Agreement are you, the licensee ("You" or "Licensee") and CodeSourcery. If You are not acting on behalf of Yourself as an individual, then "You" means Your company or organization.

2. **The Software.** The Software licensed under this Agreement consists of computer programs and documentation referred to as Sourcery G++™ Lite Edition (the "Software").

3. **Definitions.**

   3.1. **CodeSourcery Proprietary Components.** The components of the Software that are owned and/or licensed by CodeSourcery and are not subject to a "free software" or "open source" license, such as the GNU Public License. The CodeSourcery Proprietary Components of the Software include, without limitation, the Sourcery G++ Installer, any Sourcery G++ Eclipse plug-ins, and any Sourcery G++ Debug Sprite. For a complete list, refer to the *Getting Started Guide* included with the distribution.

   3.2. **Open Source Software Components.** The components of the Software that are subject to a "free software" or "open source" license, such as the GNU Public License.

   3.3. **Proprietary Rights.** All rights in and to copyrights, rights to register copyrights, trade secrets, inventions, patents, patent rights, trademarks, trademark rights, confidential and proprietary information protected under contract or otherwise under law, and other similar rights or interests in intellectual or industrial property.

   3.4. **Redistributable Components.** The CodeSourcery Proprietary Components that are intended to be incorporated or linked into Licensee object code developed with the Software. The Redistributable Components of the Software include, without limitation, the CSLIBC run-time library and the CodeSourcery Common Startup Code Sequence (CS3). For a complete list, refer to the *Getting Started Guide* included with the distribution.

4. **License Grant to Proprietary Components of the Software.** You are granted a non-exclusive, royalty-free license (a) to install and use the CodeSourcery Proprietary Components of the Software, (b) to transmit the CodeSourcery Proprietary Components over an internal computer network, (c) to copy the CodeSourcery Proprietary Components for Your internal use only, and (d) to distribute the Redistributable Component(s) in binary form only and only as part of Licensee object code developed with the Software that provides substantially different functionality than the Redistributable Component(s).

5. **Restrictions.** You may not: (i) copy or permit others to use the CodeSourcery Proprietary Components of the Software, except as expressly provided above; (ii) distribute the CodeSourcery Proprietary Components of the Software to any third party, except as expressly provided above; or (iii) reverse engineer, decompile, or disassemble the CodeSourcery Proprietary Components of the Software, except to the extent this restriction is expressly prohibited by applicable law.

6. **"Free Software" or "Open Source" License to Certain Components of the Software.** This Agreement does not limit Your rights under, or grant You rights that supersede, the license terms of any Open Source Software Component delivered to You by CodeSourcery. Sourcery G++ includes components provided under various different licenses. The *Getting Started Guide* provides an overview of which license applies to different components. Definitive licensing

information for each “free software” or “open source” component is available in the relevant source file.

7. **CodeSourcery Trademarks.** Notwithstanding any provision in a “free software” or “open source” license agreement applicable to a component of the Software that permits You to distribute such component to a third party in source or binary form, You may not use any CodeSourcery trademark, whether registered or unregistered, including without limitation, CodeSourcery™, Sourcery G++™, the CodeSourcery crystal ball logo, or the Sourcery G++ splash screen, or any confusingly similar mark, in connection with such distribution, and You may not recompile the Open Source Software Components with the `--with-pkgversion` or `--with-bugurl` configuration options that embed CodeSourcery trademarks in the resulting binary.

8. **Term and Termination.** This Agreement shall remain in effect unless terminated pursuant to this provision. CodeSourcery may terminate this Agreement upon seven (7) days written notice of a material breach of this Agreement if such breach is not cured; provided that the unauthorized use, copying, or distribution of the CodeSourcery Proprietary Components of the Software will be deemed a material breach that cannot be cured.

9. **Transfers.** You may not transfer any rights under this Agreement without the prior written consent of CodeSourcery, which consent shall not be unreasonably withheld. A condition to any transfer or assignment shall be that the recipient agrees to the terms of this Agreement. Any attempted transfer or assignment in violation of this provision shall be null and void.

10. **Ownership.** CodeSourcery owns and/or has licensed the CodeSourcery Proprietary Components of the Software and all intellectual property rights embodied therein, including copyrights and valuable trade secrets embodied in its design and coding methodology. The CodeSourcery Proprietary Components of the Software are protected by United States copyright laws and international treaty provisions. CodeSourcery also owns all rights, title and interest in and with respect to its trade names, domain names, trade dress, logos, trademarks, service marks, and other similar rights or interests in intellectual property. This Agreement provides You only a limited use license, and no ownership of any intellectual property.

11. **Warranty Disclaimer; Limitation of Liability.** CODESOURCERY AND ITS LICENSORS PROVIDE THE SOFTWARE “AS-IS” AND PROVIDED WITH ALL FAULTS. CODESOURCERY DOES NOT MAKE ANY WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. CODESOURCERY SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SYSTEM INTEGRATION, AND DATA ACCURACY. THERE IS NO WARRANTY OR GUARANTEE THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED, ERROR-FREE, OR VIRUS-FREE, OR THAT THE SOFTWARE WILL MEET ANY PARTICULAR CRITERIA OF PERFORMANCE, QUALITY, ACCURACY, PURPOSE, OR NEED. YOU ASSUME THE ENTIRE RISK OF SELECTION, INSTALLATION, AND USE OF THE SOFTWARE. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS AGREEMENT. NO USE OF THE SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

12. **Local Law.** If implied warranties may not be disclaimed under applicable law, then ANY IMPLIED WARRANTIES ARE LIMITED IN DURATION TO THE PERIOD REQUIRED BY APPLICABLE LAW.

13. **Limitation of Liability.** INDEPENDENT OF THE FORGOING PROVISIONS, IN NO EVENT AND UNDER NO LEGAL THEORY, INCLUDING WITHOUT LIMITATION, TORT, CONTRACT, OR STRICT PRODUCTS LIABILITY, SHALL CODESOURCERY BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCID-

ENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND, INCLUDING WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER MALFUNCTION, OR ANY OTHER KIND OF COMMERCIAL DAMAGE, EVEN IF CODESOURCERY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY TO THE EXTENT PROHIBITED BY APPLICABLE LAW. IN NO EVENT SHALL CODESOURCERY'S LIABILITY FOR ACTUAL DAMAGES FOR ANY CAUSE WHATSOEVER, AND REGARDLESS OF THE FORM OF ACTION, EXCEED THE AMOUNT PAID BY YOU IN FEES UNDER THIS AGREEMENT DURING THE PREVIOUS ONE YEAR PERIOD.

14. **Export Controls.** You agree to comply with all export laws and restrictions and regulations of the United States or foreign agencies or authorities, and not to export or re-export the Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary approvals. As applicable, each party shall obtain and bear all expenses relating to any necessary licenses and/or exemptions with respect to its own export of the Software from the U.S. Neither the Software nor the underlying information or technology may be electronically transmitted or otherwise exported or re-exported (i) into Cuba, Iran, Iraq, Libya, North Korea, Sudan, Syria or any other country subject to U.S. trade sanctions covering the Software, to individuals or entities controlled by such countries, or to nationals or residents of such countries other than nationals who are lawfully admitted permanent residents of countries not subject to such sanctions; or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals and Blocked Persons or the U.S. Commerce Department's Table of Denial Orders. By downloading or using the Software, Licensee agrees to the foregoing and represents and warrants that it complies with these conditions.

15. **U.S. Government End-Users.** The Software is a “commercial item,” as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of “commercial computer software” and “commercial computer software documentation,” as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire the Software with only those rights set forth herein.

16. **Licensee Outside The U.S.** If You are located outside the U.S., then the following provisions shall apply: (i) Les parties aux presentes confirment leur volonte que cette convention de meme que tous les documents y compris tout avis qui siy rattache, soient rediges en langue anglaise (translation: “The parties confirm that this Agreement and all related documentation is and will be in the English language.”); and (ii) You are responsible for complying with any local laws in your jurisdiction which might impact your right to import, export or use the Software, and You represent that You have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

17. **Severability.** If any provision of this Agreement is declared invalid or unenforceable, such provision shall be deemed modified to the extent necessary and possible to render it valid and enforceable. In any event, the unenforceability or invalidity of any provision shall not affect any other provision of this Agreement, and this Agreement shall continue in full force and effect, and be construed and enforced, as if such provision had not been included, or had been modified as above provided, as the case may be.

18. **Arbitration.** Except for actions to protect intellectual property rights and to enforce an arbitrator's decision hereunder, all disputes, controversies, or claims arising out of or relating to this Agreement or a breach thereof shall be submitted to and finally resolved by arbitration under the rules of the American Arbitration Association (“AAA”) then in effect. There shall be one arbitrator, and such arbitrator shall be chosen by mutual agreement of the parties in accordance with AAA rules. The arbitration shall take place in Granite Bay, California, and may be conducted

by telephone or online. The arbitrator shall apply the laws of the State of California, USA to all issues in dispute. The controversy or claim shall be arbitrated on an individual basis, and shall not be consolidated in any arbitration with any claim or controversy of any other party. The findings of the arbitrator shall be final and binding on the parties, and may be entered in any court of competent jurisdiction for enforcement. Enforcements of any award or judgment shall be governed by the United Nations Convention on the Recognition and Enforcement of Foreign Arbitral Awards. Should either party file an action contrary to this provision, the other party may recover attorney's fees and costs up to $1000.00.

19. **Jurisdiction And Venue.** The courts of Placer County in the State of California, USA and the nearest U.S. District Court shall be the exclusive jurisdiction and venue for all legal proceedings that are not arbitrated under this Agreement.

20. **Independent Contractors.** The relationship of the parties is that of independent contractor, and nothing herein shall be construed to create a partnership, joint venture, franchise, employment, or agency relationship between the parties. Licensee shall have no authority to enter into agreements of any kind on behalf of CodeSourcery and shall not have the power or authority to bind or obligate CodeSourcery in any manner to any third party.

21. **Force Majeure.** Neither CodeSourcery nor Licensee shall be liable for damages for any delay or failure of delivery arising out of causes beyond their reasonable control and without their fault or negligence, including, but not limited to, Acts of God, acts of civil or military authority, fires, riots, wars, embargoes, or communications failure.

22. **Miscellaneous.** This Agreement constitutes the entire understanding of the parties with respect to the subject matter of this Agreement and merges all prior communications, representations, and agreements. This Agreement may be modified only by a written agreement signed by the parties. If any provision of this Agreement is held to be unenforceable for any reason, such provision shall be reformed only to the extent necessary to make it enforceable. This Agreement shall be construed under the laws of the State of California, USA, excluding rules regarding conflicts of law. The application of the United Nations Convention of Contracts for the International Sale of Goods is expressly excluded. This license is written in English, and English is its controlling language.

B.3. Attribution

This version of Sourcery G++ Lite may include code based on work under the following copyright and permission notices:

B.3.1. Android Open Source Project

```
/*
 * Copyright (C) 2008 The Android Open Source Project
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *  * Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *  * Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
```

```
* COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
* INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
* OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
* AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*/
```