# Sourcery G++ Lite

## ARM uClinux

## Sourcery G++ Lite 2011.03-46

## Getting Started

# Sourcery G++ Lite: ARM uClinux: Sourcery G++ Lite 2011.03-46: Getting Started

CodeSourcery, Inc.


## Abstract

This guide explains how to install and build applications with Sourcery G++ Lite, CodeSourcery's customized and validated 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 ..... vi
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 ..... 9
2.7. Uninstalling Sourcery G++ Lite ..... 11
3. Sourcery G++ Lite for ARM uClinux ..... 13
3.1. Included Components and Features ..... 14
3.2. Library Configurations ..... 14
3.3. Using VFP Floating Point ..... 15
3.4. Fixed-Point Arithmetic ..... 16
3.5. ABI Compatibility ..... 17
3.6. Building uClinux Applications ..... 17
3.7. GDB Server ..... 17
4. Using Sourcery G++ from the Command Line ..... 19
4.1. Building an Application ..... 20
4.2. Running Applications on the Target System ..... 20
4.3. Running Applications from GDB ..... 21
5. Sourcery G++ Debug Sprite ..... 22
5.1. Probing for Debug Devices ..... 23
5.2. Invoking Sourcery G++ Debug Sprite ..... 23
5.3. Sourcery G++ Debug Sprite Options ..... 24
5.4. Remote Debug Interface Devices ..... 25
5.5. Actel FlashPro Devices ..... 25
5.6. Debugging a Remote Board ..... 26
5.7. Supported Board Files ..... 26
5.8. Board File Syntax ..... 27
6. Next Steps with Sourcery G++ ..... 30
6.1. Sourcery G++ Knowledge Base ..... 31
6.2. Example Programs ..... 31
6.3. Manuals for GNU Toolchain Components ..... 31
A. Sourcery G++ Lite Release Notes ..... 32
A.1. Changes in Sourcery G++ Lite for ARM uClinux ..... 33
B. Sourcery G++ Lite Licenses ..... 38
B.1. Licenses for Sourcery G++ Lite Components ..... 39
B.2. Sourcery G++ Software License Agreement ..... 40
B.3. Attribution ..... 43

# 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.

# 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 ARM uClinux" | This chapter contains information about using Sourcery G++ Lite that is specific to ARM uClinux 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, "Sourcery G++ Debug Sprite" | This chapter describes the use of the Sourcery G++ Debug Sprite for remote debugging. The Sprite is provided for debugging of the uClinux kernel on the target board. This chapter includes information about the debugging devices and boards supported by the Sprite for ARM uClinux. |
| Chapter 6, "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 ARM uClinux. 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 ARM uClinux is intended for developers working on embedded uClinux applications. It may also be used for uClinux kernel development and debugging, or to build a uClinux 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 ARM uClinux".

## 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. When the installation is complete, it offers 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. You must identify the multilib appropriate for your target in order to find the correct `gdbserver` executable to use for debugging your applications, as described in Section 3.7, "GDB Server".

## 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.

**Run your program on the target system.** Copy your program to the target system and run it from the command line.

**Debug your program on the target using GDB server.** You can debug a program on a remote ARM uClinux target using GDB server. Copy your program to the target system. Follow the instructions in Section 3.7, "GDB Server" to install and run `gdbserver` on your target system. Then, you can connect to GDB server from the debugger running on your host system. Refer to Section 4.3,

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

“Running Applications from GDB” for instructions on connecting to the target 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 `arm-uclinuxeabi`.

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 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 512MB 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. When you start the graphical installer, it checks whether there is sufficient disk space before beginning to install. Note that the graphical installer also requires additional temporary disk 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 ARM uClinux” 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. 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

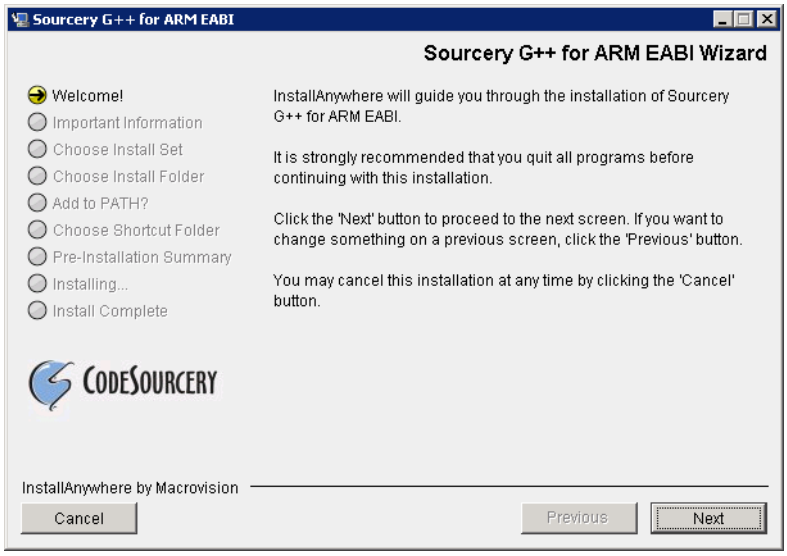
The method used to install Sourcery G++ Lite depends on your host system and the kind of installation package you have downloaded.

### 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.

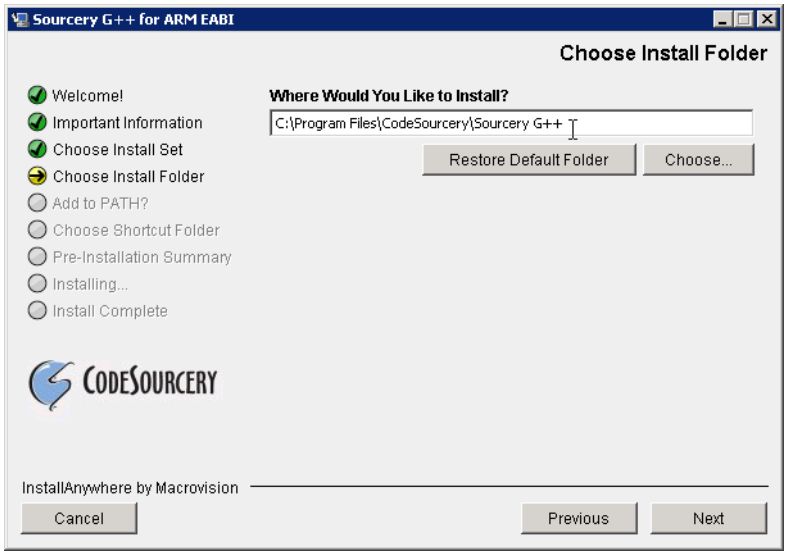
After the installer starts, follow the on-screen dialogs to install Sourcery G++ Lite. The installer is intended to be self-explanatory and on most pages the defaults are appropriate.

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

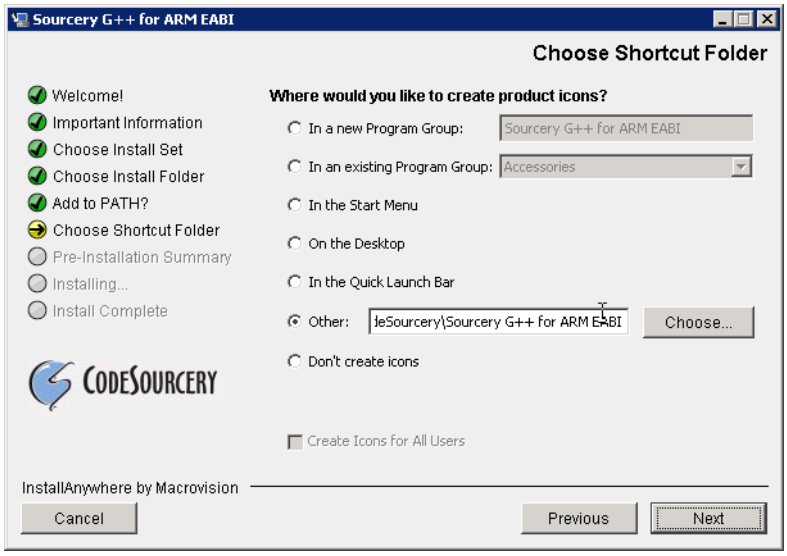



**Running the Installer.** The graphical installer guides you through the steps to install Sourcery G++ Lite.

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

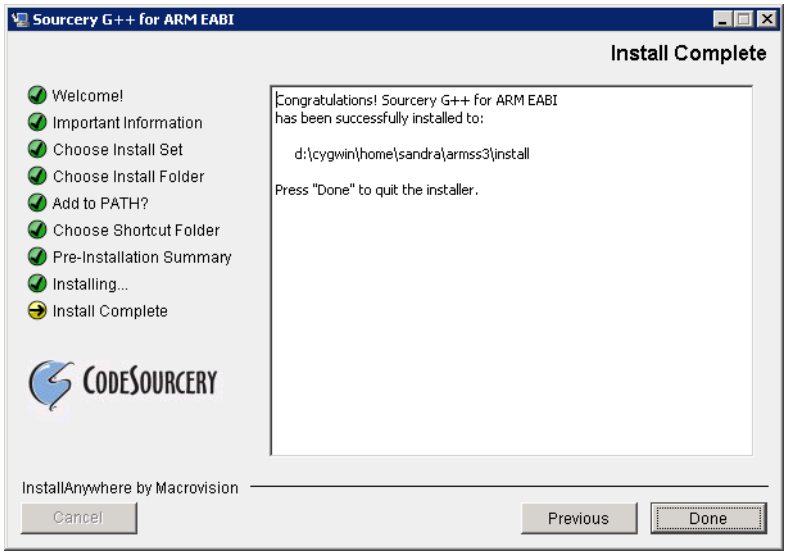



**Choose Install Folder.** Select the pathname to your install directory.




**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 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++-2011.03`.

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 ARM uClinux 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.

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 ARM uClinux target all begin with `arm-uclinuxeabi`. 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 Microsoft Windows 7, from the desktop bring up the Start menu and right click on Computer. Select Properties and click on Advanced system settings. 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. Be sure to adjust the pathname to reflect your actual installation directory.

To set the PATH on older versions of Microsoft Windows, 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:

```
> arm-uclinuxeabi-g++ -v
```

Verify that the last line of the output contains: Sourcery G++ Lite 2011.03-46.

### 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++-arm-uclinuxeabi/man`.

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

```
> arm-uclinuxeabi-g++ -v
```

Verify that the last line of the output contains: `Sourcery G++ Lite 2011.03-46`.

## 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 `arm-uclinuxeabi` 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 Uninstall executable 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 `Uninstall a program` (Vista and newer) or `Add or Remove Programs` (older versions of Windows) 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 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 ARM uClinux

This chapter contains information about features of Sourcery G++ Lite that are specific to ARM uClinux 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 ARM uClinux, and tells you where you may find further information about these features.

| Component | Version | Notes |
|---|---|---|
| **GNU programming tools** | | |
| GNU Compiler Collection | 4.5.2 | Separate manual included. |
| GNU Binary Utilities | 2.20.51 | Includes assembler, linker, and other utilities. Separate manuals included. |
| **Debugging support and simulators** | | |
| GNU Debugger | 7.2.50 | Separate manual included. |
| Sourcery G++ Debug Sprite for ARM | 2011.03-46 | Provided for kernel debugging only. See Chapter 5, “Sourcery G++ Debug Sprite”. |
| GDB Server | N/A | Included with GDB. See Section 3.7, “GDB Server”. |
| **Target libraries** | | |
| uClibc C Library | 0.9.30 | |
| Linux Kernel Headers | 2.6.38 | |
| **Other utilities** | | |
| ELF-to-FLT Conversion Utility | N/A | |
| 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++ Lite for ARM uClinux includes the following library configuration.

| **ARMv4T - Little-Endian, Soft-Float** | |
|---|---|
| Command-line option(s): | default |
| Sysroot subdirectory: | `./` |

| **ARMv6-M Thumb - Little-Endian, Soft-Float** | |
|---|---|
| Command-line option(s): | `-mthumb -march=armv6s-m` |
| Sysroot subdirectory: | `armv6-m/` |

| **ARMv7 Thumb-2 - Little-Endian, Soft-Float** | |
|---|---|
| Command-line option(s): | `-mthumb -march=armv7 -mfix-cortex-m3-ldrd` |
| Sysroot subdirectory: | `thumb2/` |

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 which contains the files that should be installed on the target system. You can find the sysroot directories provided with Sourcery G++ in the `arm-uclinuxeabi/libc` directory of your installation.

# 3.3. Using VFP Floating Point

## 3.3.1. Enabling Hardware Floating Point

GCC provides three basic options for compiling floating-point code:

- Software floating point emulation, which is the default. In this case, the compiler implements floating-point arithmetic by means of library calls.
- VFP hardware floating-point support using the soft-float ABI. This is selected by the `-mfloat-abi=softfp` option. When you select this variant, the compiler generates VFP floating-point instructions, but the resulting code uses the same call and return conventions as code compiled with software floating point.
- VFP hardware floating-point support using the VFP ABI, which is the VFP variant of the Procedure Call Standard for the ARM® Architecture (AAPCS). This ABI uses VFP registers to pass function arguments and return values, resulting in faster floating-point code. To use this variant, compile with `-mfloat-abi=hard`.

You can freely mix code compiled with either of the first two variants in the same program, as they both use the same soft-float ABI. However, code compiled with the VFP ABI is not link-compatible with either of the other two options. If you use the VFP ABI, you must use this option to compile your entire program, and link with libraries that have also been compiled with the VFP ABI. For example, you may need to use the VFP ABI in order to link your program with other code compiled by the ARM RealView® compiler, which uses this ABI.

Sourcery G++ Lite for ARM uClinux includes libraries built with software floating point, which are compatible with VFP code compiled using the soft-float ABI. While the compiler is capable of generating code using the VFP ABI, no compatible runtime libraries are provided for uClinux targets.

Note that, in addition to selecting hard/soft float and the ABI via the `-mfloat-abi` option, you can also compile for a particular FPU using the `-mfpu` option. For example, `-mfpu=neon` selects VFPv3 with NEON coprocessor extensions.

## 3.3.2. NEON SIMD Code

Sourcery G++ includes support for automatic generation of NEON SIMD vector code. Autovectorization is a compiler optimization in which loops involving normal integer or floating-point code are transformed to use NEON SIMD instructions to process several data elements at once.

To enable generation of NEON vector code, use the command-line options `-ftree-vectorize -mfpu=neon -mfloat-abi=softfp`. The `-mfpu=neon` option also enables generation of VFPv3 scalar floating-point code.

Sourcery G++ also includes support for manual generation of NEON SIMD code using C intrinsic functions. These intrinsics, the same as those supported by the ARM RealView® compiler, are defined in the `arm_neon.h` header and are documented in the 'ARM NEON Intrinsics' section of the GCC manual. The command-line options `-mfpu=neon -mfloat-abi=softfp` must be specified to use these intrinsics; `-ftree-vectorize` is not required.

### 3.3.3. Half-Precision Floating Point

Sourcery G++ for ARM uClinux includes support for half-precision (16-bit) floating point, including the new `__fp16` data type in C and C++, support for generating conversion instructions when compiling for processors that support them, and library functions for use in other cases.

To use half-precision floating point, you must explicitly enable it via the `-mfp16-format` command-line option to the compiler. For more information about `__fp16` representations and usage from C and C++, refer to the GCC manual.

## 3.4. Fixed-Point Arithmetic

Sourcery G++ for ARM uClinux includes experimental support for fixed-point arithmetic using a set of new data types, as described in the draft ISO/IEC technical report TR 18037. This support is provided for all ARM targets, and uses specialized instructions where available, e.g. saturating add and subtract operations on ARMv6T2 and above. Library functions are used for operations which are not natively supported on the target architecture.

This feature is a GNU extension, so is only available when the selected language standard includes GNU extensions (e.g. `-std=gnu90`, which is the default). Furthermore, only C is supported, not C++.

TR 18037 leaves up to the implementation the sizes of various quantities within the new data types it defines. For Sourcery G++ for ARM uClinux, these are, briefly:

- `short _Fract`: One sign bit, 7 fractional bits
- `_Fract`: One sign bit, 15 fractional bits
- `long _Fract`: One sign bit, 31 fractional bits
- `unsigned short _Fract`: 8 fractional bits
- `unsigned _Fract`: 16 fractional bits
- `unsigned long _Fract`: 32 fractional bits
- `short _Accum`: One sign bit, 7 fractional bits, 8 integral bits
- `_Accum`: One sign bit, 15 fractional bits, 16 integral bits
- `long _Accum`: One sign bit, 31 fractional bits, 32 integral bits
- `unsigned short _Accum`: 8 fractional bits, 8 integral bits
- `unsigned _Accum`: 16 fractional bits, 16 integral bits
- `unsigned long _Accum`: 32 fractional bits, 32 integral bits

These values (and various other useful constants) are also defined in the header file `stdfix.h` for use in your programs. Note that there is currently no support for the new standard-library functions described in TR 18037, nor for the pragmas controlling precision of operations.

Fixed-point extensions are not currently supported by GDB, nor are they compliant with the ARM EABI (which does not specify anything about fixed-point types at present). Code using fixed-point

types cannot be expected to interact properly (across ABI boundaries) with code generated by other compilers for the ARM architecture.

## 3.5. ABI Compatibility

The Application Binary Interface (ABI) for the ARM Architecture is a collection of standards, published by ARM Ltd. and other organizations. The ABI makes it possible to combine tools from different vendors, including Sourcery G++ and ARM RealView®.

Sourcery G++ implements the ABI as described in these documents, which are available from the ARM Information Center[1]:

- BSABI - ARM IHI 0036B (28 October 2009)
- BPABI - ARM IHI 0037B (28 October 2009)
- EHABI - ARM IHI 0038A (28 October 2009)
- CLIBABI - ARM IHI 0039B (4 November 2009)
- AADWARF - ARM IHI 0040A (28 October 2009)
- CPPABI - ARM IHI 0041C (5 October 2009)
- AAPCS - ARM IHI 0042D (16 October 2009)
- RTABI - ARM IHI 0043C (19 October 2009)
- AAELF - ARM IHI 0044D (28 October 2009)
- ABI Addenda - ARM IHI 0045C (4 November 2009)

Sourcery G++ currently produces DWARF version 2, rather than DWARF version 3 as specified in AADWARF.

## 3.6. Building uClinux Applications

When you use GCC to link a uClinux application, it creates two output files. The executable file, as specified by the `-o` command-line option, is a uClinux FLAT format binary (bFLT) file. This is the file you should copy to and run on your uClinux target. The second output file is an ELF-format file containing additional debug and symbol table information to allow you to debug your program with GDB, as described in Section 3.7, “GDB Server”. This file has a `.gdb` extension.

For example, if you specify the command

```
arm-uclinuxeabi-gcc foo.c -o bar
```

then `bar` is the FLAT-format executable and `bar.gdb` is the ELF-format file.

## 3.7. GDB Server

Sourcery G++ Lite contains a `gdbserver` for running on the target. The server executable is located in the *`sysroot`*`/usr/bin` directory of your installation, where *`sysroot`* is the pathname to the

[1] http://infocenter.arm.com

sysroot, as documented in Section 3.2, “Library Configurations”. You need to copy the appropriate `gdbserver` executable to your target system and then invoke it as

```
# gdbserver :port program
```

*`port`* can be any available TCP port; 5000 is a common choice. `gdbserver` waits for a connection from `gdb` and then commences serving requests for it. To connect to `gdbserver` from your host system, start `gdb`, but specify the special `.gdb` version of your program.

```
> arm-uclinuxeabi-gdb program.gdb
```

Then connect to the target system:

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

At this point you are able to debug as usual.

# 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 arm-uclinuxeabi, as indicated by the `arm-uclinuxeabi` 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:

```
> arm-uclinuxeabi-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 `arm-uclinuxeabi-gcc` with `arm-uclinuxeabi-g++`.)

# 4.2. Running Applications on the Target System

To run your program on a uClinux 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:

```
> arm-uclinuxeabi-gdb factorial.gdb
```

For uClinux you must specify the ELF binary, not the FLT binary that you load onto your target.

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 the Sourcery G++ Debug Sprite

The Sourcery G++ Debug Sprite is a program that runs on the host system to support hardware debugging devices. You can use the Debug Sprite to run and debug programs on a target board without an operating system, or to debug an operating system kernel. See Chapter 5, “Sourcery G++ Debug Sprite” for detailed information about the supported devices.

You can start the Sprite directly from within GDB:

```
(gdb) target remote | arm-uclinuxeabi-sprite arguments
```

Refer to Section 5.2, “Invoking Sourcery G++ Debug Sprite” for a full description of the Sprite arguments.

## 4.3.2. 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 ARM uClinux target. Follow the instructions in Chapter 3, “Sourcery G++ Lite for ARM uClinux” 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
# Sourcery G++ Debug Sprite

This chapter describes the use of the Sourcery G++ Debug Sprite for remote debugging. The Sprite is provided for debugging of the uClinux kernel on the target board. This chapter includes information about the debugging devices and boards supported by the Sprite for ARM uClinux.

Sourcery G++ Lite contains the Sourcery G++ Debug Sprite for ARM uClinux. This Sprite is provided to allow debugging of programs running on a bare board. You can use the Sprite to debug a program when there is no operating system on the board, or for debugging the operating system itself. If the board is running an operating system, and you wish to debug a program running on that OS, you should use the facilities provided by the OS itself (for instance, using `gdbserver`).

The Sprite acts as an interface between GDB and external debug devices and libraries. Refer to Section 5.2, "Invoking Sourcery G++ Debug Sprite" for information about the specific devices supported by this version of Sourcery G++ Lite.

**Note for uClinux users**

The Debug Sprite provided with Sourcery G++ Lite allows remote debugging of the uClinux kernel running on the target. For remote debugging of application programs, you should use `gdbserver` instead. See Chapter 3, "Sourcery G++ Lite for ARM uClinux" for details about how to install and run `gdbserver` on the target.

**Important**

The Sourcery G++ Debug Sprite is not part of the GNU Debugger and is not free or open-source software. You may use the Sourcery G++ Debug Sprite only with the GNU Debugger. You may not distribute the Sourcery G++ Debug Sprite to any third party.

# 5.1. Probing for Debug Devices

Before running the Sourcery G++ Debug Sprite for the first time, or when attaching new debug devices to your host system, it is helpful to verify that the Sourcery G++ Debug Sprite recognizes your debug hardware. From the command line, invoke the Sprite with the `-i` option:

```
> arm-uclinuxeabi-sprite -i
```

This prints out a list of supported device types. For devices that can be autodetected, it additionally probes for and prints out a list of attached devices. For instance:

```
CodeSourcery ARM Debug Sprite
    (Sourcery G++ Lite 2011.03-46)
armusb: [speed=<n:0-7>] ARMUSB (Stellaris) device
  armusb:///0B01000C - Stellaris Evaluation Board (0B01000C)
rdi: (rdi-library=<file>&rdi-config=<file>) RDI Device
  rdi:/// - RDI Device
```

This shows that ARMUSB and RDI devices are supported. The exact set of supported devices depends on your host system and the version of Sourcery G++ you have installed; refer to Section 5.2, "Invoking Sourcery G++ Debug Sprite" for complete information.

Note that it may take several seconds for the Debug Sprite to probe for all types of supported devices.

# 5.2. Invoking Sourcery G++ Debug Sprite

The Debug Sprite is invoked as follows:

```
> arm-uclinuxeabi-sprite [options] device-url board-file
```

The *`device-url`* specifies the debug device to use to communicate with the board. It follows the standard format:

*scheme*:*scheme-specific-part*[?*device-options*]

Most device URL schemes also follow the regular format:

*scheme*:[//*hostname*:[*port*]]/*path*[?*device-options*]

The meanings of *hostname*, *port*, *path* and *device-options* parts depend on the *scheme* and are described below. The following schemes are supported in Sourcery G++ Lite for ARM uC-linux:

`rdi` Use an RDI debugging device. Refer to Section 5.4, “Remote Debug Interface Devices”.

`flashpro` Use a FlashPro debugging device. Refer to Section 5.5, “Actel FlashPro Devices”.

The optional ?*device-options* portion is allowed in all schemes. These allow additional device-specific options of the form *name=value*. Multiple options are concatenated using `&`.

The *board-file* specifies an XML file that describes how to initialize the target board, as well as other properties of the board used by the debugger. If *board-file* refers to a file (via a relative or absolute pathname), it is read. Otherwise, *board-file* can be a board name, and the toolchain's board directory is searched for a matching file. See Section 5.7, “Supported Board Files” for the list of supported boards, or invoke the Sprite with the `-b` option to list the available board files. You can also write a custom board file; see Section 5.8, “Board File Syntax” for more information about the file format.

Both the *device-url* and *board-file* command-line arguments are required to correctly connect the Sprite to a target board.

# 5.3. Sourcery G++ Debug Sprite Options

The following command-line options are supported by the Sourcery G++ Debug Sprite:

`-b` Print a list of *board-file* files in the board config directory.

`-h` Print a list of options and their meanings. A list of *device-url* syntaxes is also shown.

`-i` Print a list of the accessible devices. If a *device-url* is also specified, only devices for that device type are scanned. Each supported device type is listed along with the options that can be appended to the *device-url*. For each discovered device, the *device-url* is printed along with a description of that device.

`-l` *[host]:port* Specify the host address and port number to listen for a GDB connection. If this option is not given, the Debug Sprite communicates with GDB using stdin and stdout. If you start the Sprite from within GDB using the `target remote | arm-uclinuxeabi-sprite ...` command, you do not need this option.

`-m` Listen for multiple sequential connections. Normally the Debug Sprite terminates after the first connection from GDB terminates. This option instead makes it listen for a subsequent connection. To terminate the Sprite, open a connection and send the string `END\n`.

`-q` Do not print any messages.

`-v` Print additional messages.

If any of `-b`, `-i` or `-h` are given, the Debug Sprite terminates after providing the information rather than waiting for a debugger connection.

# 5.4. Remote Debug Interface Devices

Remote Debug Interface (RDI) devices are supported. The RDI device URL accepts no hostname, port or path components, so the *`device-url`* is specified as follows:

```
rdi:[///][?device-options]
```

The following *`device-options`* are required:

`rdi-library=`*`library`* Specify the library (DLL or shared object) implementing the RDI target you wish to use.

`rdi-config=`*`configfile`* Specify a file containing configuration information for *`library`*. The format of this file is specific to the RDI library you are using, but tends to constitute a list of *`key=value`* pairs. Consult the documentation of your RDI library for details.

# 5.5. Actel FlashPro Devices

On Windows hosts, Sourcery G++ Lite supports FlashPro devices used with Actel Cortex-M1 development kits.

For FlashPro devices, the *`device-url`* has the following form:

```
flashpro:[//usb12345/][?jtagclock=rate]
```

The optional *`usb12345`* part indicates the ID of the FlashPro device to connect to, which is useful if you have more than one such device attached to your computer. If the ID is omitted, the Debug Sprite connects automatically to the first detected FlashPro device. You can enumerate the connected FlashPro devices by invoking the Sprite with the `-i` switch, as follows:

```
> arm-uclinuxeabi-sprite -i flashpro:
```

The `jtagclock` option allows the communication speed with the target board to be altered. The *`rate`* is specified in Hz and may range between 93750 and 4000000. The default is 93750, the slowest speed supported by the FlashPro device. Depending on your target board, you may be able to increase this rate, but beware that communication errors may occur above a certain threshold. If you encounter communication errors with a higher-than-default speed selected, try reducing the speed.

## 5.5.1. Installing FlashPro Windows drivers

Windows drivers for the FlashPro device are included with the FlashPro software provided by Actel. Refer to Actel's documentation for details on installing this software. You must use the Actel FlashPro software to configure the FPGA on your Cortex-M1 board, but it does not need to be running when using the Debug Sprite.

Once you have set up your board using the FlashPro software, you can check that it is recognized by the Sourcery G++ Debug Sprite by running the following command:

```
> arm-uclinuxeabi-sprite -i
flashpro: [jtagclock=<n:93750-4000000>] FlashPro device
  flashpro://usb12345/ - FlashPro Device
  ...
```

If output similar to the above does not appear, your FlashPro device is not working correctly. Contact CodeSourcery for further guidance in that case.

# 5.6. Debugging a Remote Board

You can run the Sourcery G++ Debug Sprite on a different machine from the one on which GDB is running. For example, if your board is connected to a machine in your lab, you can run the debugger on your laptop and connect to the remote board. The Sourcery G++ Debug Sprite must run on the machine that is connected to the target board. You must have Sourcery G++ installed on both machines.

To use this mode, you must start the Sprite with the `-l` option and specify the port on which you want it to listen. For example:

```
> arm-uclinuxeabi-sprite -l :10000 device-url board-file
```

starts the Sprite listening on port 10000.

When running GDB from the command line, use the following command to connect GDB to the remote Sprite:

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

where *`host`* is the name of the remote machine. After this, debugging is just as if you are debugging a target board connected to your host machine.

For more detailed instructions on using the Sourcery G++ Debug Sprite in this way, please refer to the Sourcery G++ Knowledge Base[1].

# 5.7. Supported Board Files

The Sourcery G++ Debug Sprite for ARM uClinux includes support for the following target boards. Specify the appropriate *`board-file`* as an argument when invoking the Sprite from the command line.

| **Board** | **Config** |
|---|---|
| Altera Cyclone III Cortex-M1 | `cycloneiii-cm1` |
| ARMulator (RDI) | `armulator` |
| Xilinx Cortex-A9 | `xilinxa9` |

[1] https://support.codesourcery.com/GNUToolchain/kbentry132

# 5.8. Board File Syntax

The *`board-file`* can be a user-written XML file to describe a non-standard board. The Sourcery G++ Debug Sprite searches for board files in the `arm-uclinuxeabi/lib/boards` directory in the installation. Refer to the files in that directory for examples.

The file's DTD is:

```
<!-- Board description files

     Copyright (c) 2007-2009 CodeSourcery, Inc.

     THIS FILE CONTAINS PROPRIETARY, CONFIDENTIAL, AND TRADE
     SECRET INFORMATION OF CODESOURCERY AND/OR ITS LICENSORS.

     You may not use or distribute this file without the express
     written permission of CodeSourcery or its authorized
     distributor.  This file is licensed only for use with
     Sourcery G++.  No other use is permitted.
     -->

<!ELEMENT board
 (properties?, feature?, initialize?, memory-map?)>

<!ELEMENT properties
 (description?, property*)>

<!ELEMENT initialize
 (write-register | write-memory | delay
  | wait-until-memory-equal | wait-until-memory-not-equal)* >
<!ELEMENT write-register EMPTY>
<!ATTLIST write-register
          address CDATA   #REQUIRED
                        value   CDATA   #REQUIRED
                        bits    CDATA   #IMPLIED>
<!ELEMENT write-memory EMPTY>
<!ATTLIST write-memory
          address CDATA   #REQUIRED
                        value   CDATA   #REQUIRED
                        bits    CDATA   #IMPLIED>
<!ELEMENT delay EMPTY>
<!ATTLIST delay
          time CDATA   #REQUIRED>
<!ELEMENT wait-until-memory-equal EMPTY>
<!ATTLIST wait-until-memory-equal
          address CDATA   #REQUIRED
                        value   CDATA   #REQUIRED
                        timeout CDATA   #IMPLIED
                        bits    CDATA   #IMPLIED>
<!ELEMENT wait-until-memory-not-equal EMPTY>
<!ATTLIST wait-until-memory-not-equal
          address CDATA   #REQUIRED
                        value   CDATA   #REQUIRED
```

```
                        timeout CDATA   #IMPLIED
                        bits    CDATA   #IMPLIED>

<!ELEMENT memory-map (memory-device)*>
<!ELEMENT memory-device (property*, description?, sectors*)>
<!ATTLIST memory-device
                        address CDATA   #REQUIRED
          size    CDATA   #REQUIRED
          type    CDATA   #REQUIRED
                        device  CDATA   #IMPLIED>

<!ELEMENT description (#PCDATA)>
<!ELEMENT property (#PCDATA)>
<!ATTLIST property name CDATA #REQUIRED>
<!ELEMENT sectors EMPTY>
<!ATTLIST sectors
 size CDATA #REQUIRED
 count CDATA #REQUIRED>

<!ENTITY % gdbtarget SYSTEM "gdb-target.dtd">
%gdbtarget;
```

All values can be provided in decimal, hex (with a `0x` prefix) or octal (with a `0` prefix). Addresses and memory sizes can use a `K`, `KB`, `M`, `MB`, `G` or `GB` suffix to denote a unit of memory. Times must use a `ms` or `us` suffix.

The following elements are available:

`<board>` This top-level element encapsulates the entire description of the board. It can contain `<properties>`, `<feature>`, `<initialize>` and `<memory-map>` elements.

`<properties>` The `<properties>` element specifies specific properties of the target system. This element can occur at most once. It can contain a `<description>` element.

It can also contain `<property>` elements with the following names:

`banked-regs` The `banked-regs` property specifies that the CPU of the target board has banked registers for different processor modes (supervisor, IRQ, etc.).

`has-vfp` The `has-vfp` property specifies that the CPU of the target board has VFP registers.

`system-v6-m` The `system-v6-m` property specifies that the CPU of the target board has ARMv6-M architecture system registers.

`system-v7-m` The `system-v7-m` property specifies that the CPU of the target board has ARMv7-M architecture system registers.

`core-family` The `core-family` property specifies the ARM family of the target. The body of the `<property>`

element may be one of `arm7`, `arm9`, `arm11`, and `cortex`.

`system-clock` This property specifies the target clock frequency (in Hertz) after reset. It is used to configure flash programming algorithms.

`<initialize>` The `<initialize>` element defines an initialization sequence for the board, which the Sprite performs before downloading a program. It can contain `<write-register>`, `<write-memory>` and `<delay>` elements.

`<feature>` This element is used to inform GDB about additional registers and peripherals available on the board. It is passed directly to GDB; see the GDB manual for further details.

`<memory-map>` This element describes the memory map of the target board. It is used by GDB to determine where software breakpoints may be used and when flash programming sequences must be used. This element can occur at most once. It can contain `<memory-device>` elements.

`<memory-device>` This element specifies a region of memory. It has four attributes: `address`, `size`, `type` and `device`. The `address` and `size` attributes specify the location of the memory device. The `type` attribute specifies that device as `ram`, `rom` or `flash`. The `device` attribute is required for `flash` regions; it specifies the flash device type. The `<memory-device>` element can contain a `<description>` element.

`<write-register>` This element writes a value to a control register. It has three attributes: `address`, `value` and `bits`. The `bits` attribute, specifying the bit width of the write operation, is optional; it defaults to 32.

`<write-memory>` This element writes a value to a memory location. It has three attributes: `address`, `value` and `bits`. The `bits` attribute is optional and defaults to 32. Bit widths of 8, 16 and 32 bits are supported. The address written to must be naturally aligned for the size of the write being done.

`<delay>` This element introduces a delay. It has one attribute, `time`, which specifies the number of milliseconds, or microseconds to delay by.

`<description>` This element encapsulates a human-readable description of its enclosing element.

`<property>` The `<property>` element allows additional name/value pairs to be specified. The property name is specified in a `name` attribute. The property value is the body of the `<property>` element.

# Chapter 6
# Next Steps with Sourcery G++

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

# 6.1. 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.

# 6.2. Example Programs

Sourcery G++ Lite includes some bundled example programs. You can find the source code for these examples in the `share/sourceryg++-arm-uclinuxeabi-examples` directory of your Sourcery G++ installation.

The subdirectories contain a number of small, target-independent test programs. You may find these programs useful as self-contained test cases when experimenting with configuring the correct compiler and debugger settings for your target, or when learning how to use the debugger or other features of the Sourcery G++ toolchain.

# 6.3. 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++-arm-uclinuxeabi/` 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++-arm-uclinuxeabi/man/man1
```

Then you can invoke `man` as:

```
> man ./arm-uclinuxeabi-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 ARM uClinux. You should read through these notes to learn about new features and bug fixes.

# A.1. Changes in Sourcery G++ Lite for ARM uClinux

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

## A.1.1. Changes in Sourcery G++ Lite 2011.03-46

**Variable Length Array (VLA) alignment bug.** A compiler bug that resulted in incorrectly aligned variable length arrays (VLA) in leaf functions has been fixed.

**Cortex-R5 support.** Sourcery G++ now includes support for ARM Cortex-R5 processors. To compile for these processors, use `-mcpu=cortex-r5`.

**Inline assembly and volatile fields.** A bug has been fixed that caused the compiler to incorrectly reject inline `asm` statements referring to volatile class/struct fields with errors such as `error: output number 1 not directly addressable`.

**Fixed-point arithmetic support.** Experimental compiler support has been added for fixed-point arithmetic on ARM, as described in the draft ISO/IEC technical report TR 18037. Specialized instructions defined in recent architecture versions for performing saturating arithmetic, etc. are used when available, but are not a prerequisite for using the new language features. See Section 3.4, "Fixed-Point Arithmetic" for further details.

**C++ constructor bug fix.** A compiler bug has been fixed that caused incorrect code for C++ constructors for some class hierarchies that use virtual inheritance and include empty classes. At runtime, the incorrect constructors resulted in memory corruption or other errors.

**Thumb debug information fix.** A compiler bug that resulted in incorrect debug information for Thumb code has been fixed. The incorrect information prevented single stepping through some code.

**Unaligned access support.** The compiler now generates more efficient code for accessing packed data structures and for copying small blocks of unaligned data when targeting architectures that permit unaligned word/halfword accesses. This feature can be controlled by the `-munaligned-access` and `-mno-unaligned-access` options, and is enabled by default for ARMv6 processors and above, except for ARMv6-M.

## A.1.2. Changes in Sourcery G++ Lite 2011.03-20

**GCC fixes for `-fstrict-volatile-bitfields`.** GCC now honors `-fstrict-volatile-bitfields` when a bitfield is not declared volatile initially, but an object including bit fields is cast to volatile. Also, a bug was fixed that caused incorrect code to be generated for some stores to volatile bit fields when `-fstrict-volatile-bitfields` is enabled.

**Compiler optimization improvements.** The compiler has been enhanced with a number of optimization improvements, including:

- Smaller and faster code for compound conditionals.
- Removal of superfluous sign and zero extensions.
- Improved code for multiply-and-accumulate operations on ARM.

**Internal compiler error with NEON intrinsics.** A compiler bug has been fixed that caused internal compiler errors when using certain NEON intrinsics.

**GCC version 4.5.2.** Sourcery G++ Lite for ARM uClinux is now based on GCC version 4.5.2.

**GCC code generation bug for casts to `volatile` types.** A compiler bug has been fixed that sometimes caused incorrect code for references to pointers to types with `volatile` casts.

**Incorrect optimization fix.** An optimizer bug that in rare cases caused incorrect code to be generated for complex AND and OR expressions containing redundant subexpressions has been fixed.

**Incorrect C++ warning fixed.** A bug in GCC has been fixed that caused spurious warnings about lambda expressions in C++ code that does not use them.

**GCC fixes for NEON in big-endian mode.** Several compiler bugs have been fixed that could lead to incorrect code when using NEON in big-endian mode. The problems only manifested when using the auto-vectorizer (enabled by default at the `-O3` optimization level) with the `-mvectorize-with-neon-quad` option.

**Incorrect code for built-in comparison functions.** A bug has been fixed that sometimes caused GCC's built-in comparison functions, such as `__builtin_isgreaterequal`, to incorrectly raise exceptions when invoked on unordered floating-point arguments.

**C++ exception handling.** A defect in the implementation of the EH-ABI specification has been fixed. The defect affected the catching of pointer types in code generated by the ARM RealView® compiler but using the Sourcery G++ runtime libraries. The fix also retains backward compatibility with existing GCC-compiled code.

**GCC bug where accesses to volatile structure fields are optimized away.** A bug has been fixed where accesses to volatile fields of a structure were sometimes incorrectly optimized away if the structure instance was defined as non-volatile.

**Internal compiler error fixes.** Two bugs have been fixed that caused compiler crashes in rare cases. The first bug involved code with multiple comparison operations, and the second one involved `char` to `int` conversion.

**Thumb-2 assembler validation fix.** The assembler now correctly rejects Thumb-2 `ADD`, `ADDS`, `SUB`, and `SUBS` instructions that have an invalid shift operand. Previously, invalid shift values were accepted and generated unpredictable instructions.

**Objdump fix for multiple input files.** The Objdump utility did not produce correct disassembly when processing multiple input files. This has been fixed.

**Linux kernel headers update.** Linux kernel header files have been updated to version 2.6.38.

### A.1.3. Changes in Sourcery G++ Lite 2010.09-58

**GCC fix for duplicated symbols.** A GCC optimizer bug that caused multiple definitions of local symbols has been fixed. Code affected by the bug was rejected by the assembler.

**NEON code generation fix.** A GCC bug has been fixed that resulted in an assembler error `VFP/Neon double precision register expected`.

**Static data size improvement at `-Os`.** When optimizing for size, the compiler no longer implicitly adds padding bytes to align static and local arrays on word boundaries. This fixes static data size regressions introduced since GCC 4.4. The additional alignment is still used when optimizing for speed.

**New `-fstrict-volatile-bitfields` option.** The compiler has a new option, `-fstrict-volatile-bitfields`, which forces access to a volatile structure member using the width that conforms to its type. This option is enabled by default to conform to the ARM EABI. Refer to the GCC manual for details.

**Internal compiler error fixes.** A bug has been fixed that caused the compiler to crash on code containing a `typedef` alias for `__builtin_va_list` with option `-femit-struct-debug-baseonly`. A second bug has been fixed that caused a crash when compiling code using C99 variable-length arrays. Additionally, a compiler crash on code using 64-bit integer multiplications with NEON vectorization enabled has also been fixed.

**NEON narrowing-move instructions.** The compiler now supports narrowing-move instructions when auto-vectorizing for NEON. Loops accessing arrays of `char` or `short` values are now more likely to be vectorized.

**Improved support for atomic memory builtins.** The compiler support for built-in atomic memory access operations on ARMv7 targets has been improved. These builtins are documented in the GCC manual.

**Linker debug information fix.** A bug in linker processing of debug information has been fixed. The bug sometimes prevented the Sourcery G++ debugger from displaying source code if the executable was linked with the `--gc-sections` option.

**Absolute branch bug fixes.** A bug that caused the assembler to crash on a branch to an absolute address has been fixed. Linker handling of the resulting relocations has also been improved. Previously this caused an invalid switch to ARM mode on ARMv7-M devices.

**`VMOV` instruction bug fix.** A bug that caused the assembler to incorrectly reject certain valid immediate operands for the `VMOV` instruction has been fixed.

### A.1.4. Changes in Sourcery G++ Lite 2010.09-21

**Changes to Sourcery G++ version numbering.** Sourcery G++ product and Lite toolchains now uniformly use a version numbering scheme of the form 2011.03-46. The major and minor parts of the version number, in this case 2011.03, identify the release branch, while the final component is a build number within the branch. There are also new preprocessor macros defined by the compiler for the version number components so that you may conditionalize code for Sourcery G++ or particular Sourcery G++ versions. Details are available in the Sourcery G++ Knowledge Base[1].

**GCC fix for reference to undefined label.** A bug in the optimizer that caused GCC to emit references to undefined labels has been fixed.

**Precision improvement with vectorization enabled.** The GCC auto-vectorizer no longer uses NEON floating-point instructions unless the `-funsafe-math-optimizations` option (implied by `-ffast-math`) is specified. This is because NEON hardware does not fully support the IEEE 754 standard for floating-point arithmetic. In particular, very small quantities may be flushed to zero.

**Alignment attributes.** A bug has been fixed that caused the compiler to ignore alignment attributes of C++ static member variables where the attribute was present on the definition, but not the declaration.

[1] https://support.codesourcery.com/GNUToolchain/kbentry1

**`naked` attribute semantics.** The `naked` function attribute now also implies the `noinline` and `noclone` attributes. This fixes bugs resulting from invalid optimizations of functions with this attribute.

**Stack corruption bug fix.** A bug in GCC has been fixed that caused stack corruption in functions with the `interrupt` attribute.

**GCC bug fix for push multiple instruction generation.** A bug has been fixed that caused GCC to generate incorrect push multiple instructions, causing an assembler warning `register range not in ascending order`.

**Thumb-2 internal compiler error fix.** A bug has been fixed that caused the compiler to crash when compiling Thumb-2 code using 64-bit integer arithmetic.

**Compiler optimization improvements.** The compiler has been enhanced with a number of optimization improvements, including:

- More efficient assignment for structures containing bitfields.
- Better code for initializing C++ arrays with explicit element initializers.
- Improved logic for eliminating/combining redundant comparisons in code with nested conditionals.
- Better selection of loop variables, resulting in fewer temporaries and more efficient register usage.
- More optimization of references to globals in position-independent code.
- Various Thumb code generation improvements.
- Better code when constant addresses are used as arguments to inline assembly statements.
- Better code for copying small constant strings.
- Improved tuning for Cortex-M4 processors.
- Cortex-A9 specific tuning for VFP and NEON instructions.
- Use of more NEON features.

**Preprocessor symbols for floating-point calling convention.** Built-in preprocessor symbols `__ARM_PCS` and `__ARM_PCS_VFP` are now defined to indicate the current floating-point calling convention.

**GCC version 4.5.1.** Sourcery G++ Lite for ARM uClinux is now based on GCC version 4.5.1. For more information about changes from GCC version 4.4 that was included in previous releases, see `http://gcc.gnu.org/gcc-4.5/changes.html`.

**New `-Wdouble-promotion` warning option.** The compiler has a new option, `-Wdouble-promotion`, which enables warnings about implicit promotions of `float` values to `double`. This option is useful when compiling code for processors (such as ARM Cortex-M4) that have hardware support for single-precision floating-point arithmetic only, where unintentional use of double precision results in dramatically slower code.

**Linker bug fix.** A bug that caused the linker error `relocation truncated to fit: R_ARM_THM_JUMP24` when linking some Thumb-2 applications has been fixed.

**Assembler PC-relative store fix.** A bug that caused the assembler to reject some valid PC-relative store instructions has been fixed. It now issues a warning instead for architectures where these instructions are deprecated.

**ARMv7-A linker bug fix.** A bug in the linker support for `--fix-cortex-a8`, which is enabled by default when linking ARMv7-A objects, has been fixed. Programs affected by the bug sometimes crashed with segmentation fault or illegal instruction errors.

**Smaller C++ programs with `-g`.** An assembler bug has been fixed that caused unnecessary references to exception-handling routines from C++ programs when debug information is enabled. For programs that do not otherwise use exceptions, this change results in smaller code size.

**Additional validation in the assembler.** The assembler now diagnoses an error, instead of producing an invalid object file, when directives such as `.hidden` are missing operands.

**Assembler PC-relative load fix.** An assembler bug that caused the assembler to reject some references to global symbols has been fixed. This bug affected Thumb instructions of the form `ldr r0, symbol`.

**Strip bug fix.** A bug in the `strip` and `objcopy` utilities, which resulted in stripped object files that the linker could not recognize, has been fixed.

**Binutils update.** The binutils package has been updated to version 2.20.51.20100809 from the FSF trunk. This update includes numerous bug fixes.

**Linux kernel headers update.** Linux kernel header files have been updated to version 2.6.35.2.

**Improved support for debugging RealView® C++ programs .** GDB has been enhanced to handle some debug information contained in binaries produced by the ARM RealView® compiler. Formerly, GDB sometimes crashed on programs which use C++ templates. Another bug has been fixed that caused GDB to fail to place breakpoints in binaries produced by the ARM RealView® compiler when the source file location for the breakpoint was specified as an absolute pathname.

**GDB update.** The included version of GDB has been updated to 7.2.50.20100908. This update adds numerous bug fixes and new features, including improved C++ language support, a new command to save breakpoints to a file, a new convenience variable `$_thread` that holds the number of the current thread, among many other improvements.

**GDB crash fix.** A bug has been fixed that caused GDB to crash on launch if the environment variable `CYGPATH` is set to a program that does not exist or cannot be executed.

### A.1.5. Changes in Older Releases

For information about changes in older releases of Sourcery G++ Lite for ARM uClinux, 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 ARM uClinux 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<br>http://www.gnu.org/licenses/gpl.html |
| GNU Binary Utilities | GNU General Public License 3.0<br>http://www.gnu.org/licenses/gpl.html |
| GNU Debugger | GNU General Public License 3.0<br>http://www.gnu.org/licenses/gpl.html |
| Sourcery G++ Debug Sprite for ARM | CodeSourcery License |
| uClibc C Library | GNU Lesser General Public License 2.1<br>http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html |
| Linux Kernel Headers | GNU General Public License 2.0<br>http://www.gnu.org/licenses/old-licenses/gpl-2.0.html |
| ELF-to-FLT Conversion Utility | GNU General Public License 2.0<br>http://www.gnu.org/licenses/old-licenses/gpl-2.0.html |
| GNU Make | GNU General Public License 2.0<br>http://www.gnu.org/licenses/old-licenses/gpl-2.0.html |
| GNU Core Utilities | GNU General Public License 2.0<br>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++-arm-uclinuxeabi-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, the CodeSourcery C Library (CSLIBC), 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, CSLIBC 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 You 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.

   5.1.

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, and, for components subject to the Eclipse Public License, contains information on how to obtain the source code. 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, INCIDENTAL, 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.
*/
```