Symbian ADT Sourcery G++ Lite

ARM SymbianOS

Symbian ADT Sourcery G++ Lite 4.4-172

Getting Started

Symbian ADT Sourcery G++ Lite: ARM SymbianOS: Symbian ADT Sourcery G++ Lite 4.4-172: Getting Started

CodeSourcery, Inc.

Abstract

This guide explains how to install and build applications with Symbian ADT Sourcery G++ Lite, CodeSourcery's customized, validated, and supported version of the GNU Toolchain. Symbian ADT 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

This preface introduces the Symbian ADT 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 Symbian ADT Sourcery G++ Lite. This guide provides a step-by-step guide to installing Symbian ADT 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 Symbian ADT 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 Symbian ADT 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, "Symbian ADT Sourcery G++ Lite for ARM SymbianOS"	This chapter contains information about using Symbian ADT Sourcery G++ Lite that is specific to ARM SymbianOS targets. You should read this chapter to learn how to best use Symbian ADT Sourcery G++ Lite on your target system.
Chapter 4, "Using Sourcery G++ from the Command Line"	This chapter explains how to build applications with Symbian ADT 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 applications running on a remote Symbian host or simulator. This chapter includes information about the debugging devices and boards supported by the Sprite for ARM SymbianOS.
Chapter 6, "Next Steps with Sourcery G++"	This chapter describes where you can find additional documentation and information about using Symbian ADT 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, "Symbian ADT Sourcery G++ Lite Release Notes"	This appendix contains information about changes in this release of Symbian ADT Sourcery G++ Lite for ARM SymbianOS. You should read through these notes to learn about new features and bug fixes.
Appendix B, "Symbian ADT Sourcery G++ Lite Licenses"	This appendix provides information about the software licenses that apply to Symbian ADT Sourcery G++ Lite. Read this appendix to understand your legal rights and obligations as a user of Symbian ADT 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 Symbian ADT Sourcery G++ Lite for the first time. You may use this chapter as an abbreviated guide to the rest of this manual.

Symbian ADT Sourcery G++ Lite for ARM SymbianOS is intended for developers working on embedded SymbianOS applications. It may also be used for SymbianOS kernel development and debugging, or to build a SymbianOS distribution.

Follow the steps given in this chapter to install Symbian ADT 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 Symbian ADT Sourcery G++ Lite in Chapter 3, "Symbian ADT Sourcery G++ Lite for ARM SymbianOS".

1.1. Installation and Set-Up

Install Symbian ADT Sourcery G++ Lite on your host computer. You may download an installer package from the Sourcery G++ web site[1], or you may have received an installer on CD. The installer is an executable program that pops up a window on your computer and leads you through a series of dialogs to configure your installation. If the installation is successful, it will offer to launch the Getting Started guide. For more information about installing Symbian ADT 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. 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 Symbian ADT Sourcery G++ Lite.

1.3. 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.

Debug your program in the QEMU platform emulator. The QEMU emulator provides an easy way to try out your program without requiring target hardware. Refer to Section 4.3, "Running Applications with QEMU Platform Emulator" for instructions on using QEMU.

Debug your program on the target using the Debug Sprite. You can use the Sourcery G++ Debug Sprite to load and execute your program on the target from the debugger. Refer to Section 4.4, "Running Applications from GDB" for instructions on using the Sprite from the GDB command line. Detailed reference material for the Sourcery G++ Debug Sprite, including information about supported debug devices, can be found in Chapter 5, "Sourcery G++ Debug Sprite".

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

Chapter 2
Installation and Configuration

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

1. Verify that you can install Symbian ADT Sourcery G++ Lite on your system.
2. Download the appropriate Symbian ADT Sourcery G++ Lite installer.
3. Install Symbian ADT Sourcery G++ Lite.
4. Configure your environment so that you can use Symbian ADT 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-none-symbianelf`.

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

2.2. System Requirements

2.2.1. Host Operating System Requirements

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

- Microsoft Windows NT 4, Windows 2000, Windows XP, and Windows Vista systems using IA32, AMD64, and EM64T processors.
- GNU/Linux systems using IA32, AMD64, or EM64T processors, including Debian 3.1 (and later), Red Hat Enterprise Linux 3 (and later), and SuSE Enterprise Linux 8 (and later).
- Apple Mac OS X 10.4 (Tiger) and 10.5 (Leopard).

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 Symbian ADT 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 Symbian ADT 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 Symbian ADT Sourcery G++ Lite toolchain.

2.2.2. Host Hardware Requirements

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

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

In addition, the graphical installer requires a similar amount of temporary space during the installation process. On Microsoft Windows hosts, the installer uses the location specified by the `TEMP` environment variable for these temporary files. If there is not enough free space on that volume, the installer prompts for an alternate location. On Linux hosts, the installer puts temporary files in the directory specified by the `IATEMPDIR` environment variable, or `/tmp` if that is not set. On Mac OS X hosts, the installer scans all mounted volumes with write permissions for one with sufficient free space to use for temporary files.

2.2.3. Target System Requirements

See Chapter 3, "Symbian ADT Sourcery G++ Lite for ARM SymbianOS" for requirements that apply to the target system.

2.3. Downloading an Installer

If you have received Symbian ADT 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 Symbian ADT Sourcery G++ Lite".

You can download Symbian ADT 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 Symbian ADT Sourcery G++ Lite is provided as an executable installer package with the `.bin` extension. For Mac OS X systems, Symbian ADT Sourcery G++ Lite is provided as a Mac OS X installer contained inside a compressed archive with the `.zip` 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. On Mac OS X systems, save the archive file in your home directory.

2.4. Installing Symbian ADT Sourcery G++ Lite

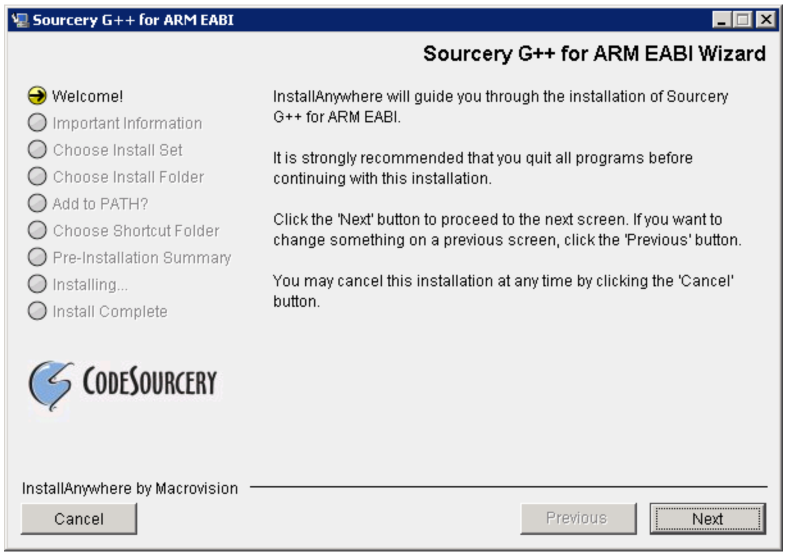
The method used to install Symbian ADT Sourcery G++ Lite depends on your host system and the kind of installation package you have downloaded.

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

If you have received Symbian ADT 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 Symbian ADT Sourcery G++ Lite, double-click on the installer.

After the installer starts, follow the on-screen dialogs to install Symbian ADT 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/


Sourcery G++ for ARM EABI

Sourcery G++ for ARM EABI Wizard

InstallAnywhere will guide you through the installation of Sourcery G++ for ARM EABI.

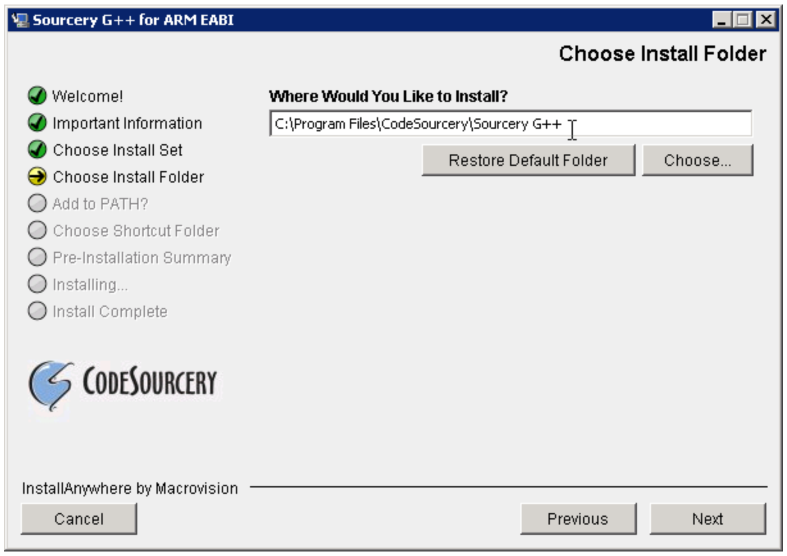
It is strongly recommended that you quit all programs before continuing with this installation.

Click the 'Next' button to proceed to the next screen. If you want to change something on a previous screen, click the 'Previous' button.

You may cancel this installation at any time by clicking the 'Cancel' button.

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

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

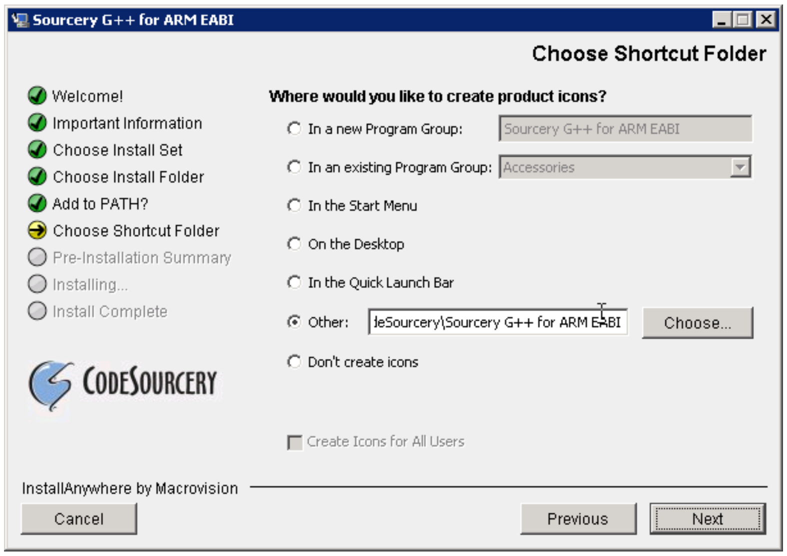

Sourcery G++ for ARM EABI

Choose Install Folder

Where Would You Like to Install?

C:\Program Files\CodeSourcery\Sourcery G++

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



Where would you like to create product icons?

( ) In a new Program Group: Sourcery G++ for ARM EABI

( ) In an existing Program Group: Accessories

( ) In the Start Menu

( ) On the Desktop

( ) In the Quick Launch Bar

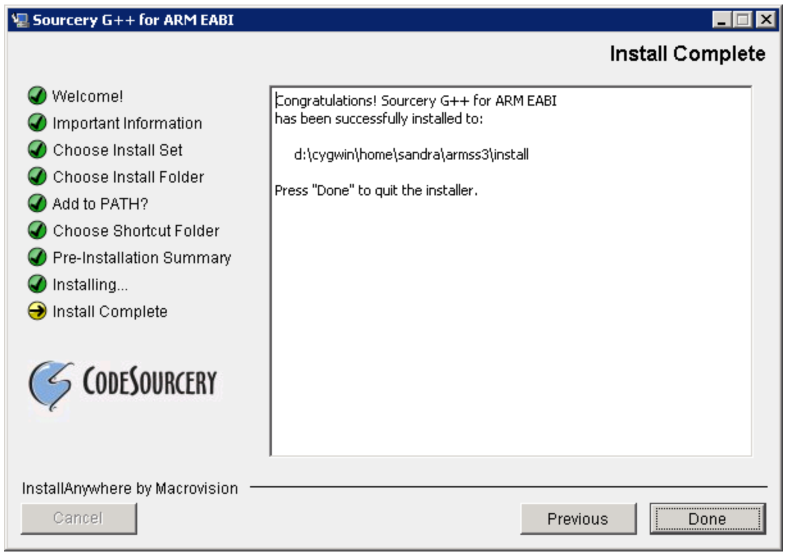
(•) Other: :eSourcery\Sourcery G++ for ARM EABI Choose...

( ) Don't create icons

[ ] Create Icons for All Users

Choose Shortcut Folder. You can customize where the installer creates shortcuts for quick access to Symbian ADT 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.



Congratulations! Sourcery G++ for ARM EABI
has been successfully installed to:

d:\cygwin\home\sandra\armss3\install

Press "Done" to quit the installer.

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 Symbian ADT 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 Symbian ADT 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. Using the Symbian ADT Sourcery G++ Lite Installer on Mac OS X

Double-click on the archive file (with `.zip` extension), and wait for the contents to uncompress. You can then start the graphical installer by double-clicking on the installer icon which appears in the same directory as the archive file.

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

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

You do not need to be a system administrator to install Symbian ADT Sourcery G++ Lite from a compressed archive. You may install Symbian ADT 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 Symbian ADT Sourcery G++ Lite in the `$HOME/CodeSourcery` subdirectory of your home directory and that the filename of the package you have downloaded is `/path/to/package.tar.bz2`. After installation the toolchain will be in `$HOME/CodeSourcery/sourceryg++-4.4`.

First, uncompress the package file:

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

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

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

Change to the installation directory:

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

Unpack the package:

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

2.5. Installing Symbian ADT Sourcery G++ Lite Updates

If you have already installed an earlier version of Symbian ADT Sourcery G++ Lite for ARM SymbianOS 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 SymbianOS target all begin with `arm-none-symbianelf`. 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 Symbian ADT Sourcery G++ Lite installation.

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

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

```
> arm-none-symbianelf-g++ -v
```

Verify that the last line of the output contains: `Symbian ADT Sourcery G++ Lite 4.4-172`.

2.6.1.2. Working with Cygwin

Symbian ADT 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, Symbian ADT 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 Symbian ADT 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 or Mac OS X Hosts

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

Before using Symbian ADT 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 Symbian ADT Sourcery G++ Lite in an alternate location, you must replace the directory above with `bin` subdirectory of the directory in which you installed Symbian ADT 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-none-symbianelf/man`.

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

```
> arm-none-symbianelf-g++ -v
```

Verify that the last line of the output contains: `Symbian ADT Sourcery G++ Lite 4.4-172`.

2.7. Uninstalling Symbian ADT Sourcery G++ Lite

The method used to uninstall Symbian ADT 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.

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

For Windows hosts other than Microsoft Windows Vista, select `Start`, then `Control Panel`. Select `Add or Remove Programs`. Scroll down and click on `Sourcery G++ for ARM SymbianOS`. Select `Change/Remove` and follow the on-screen dialogs to uninstall Symbian ADT Sourcery G++ Lite.

On Microsoft Windows Vista hosts, select `Start`, then `Settings` and finally `Control Panel`. Select the `Uninstall a program` task. Scroll down and double click on `Sourcery G++ for ARM SymbianOS`. Follow the on-screen dialogs to uninstall Symbian ADT 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 Symbian ADT Sourcery G++ Lite installation directory with the `-i console` command-line option.

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

2.7.2. Using the Symbian ADT Sourcery G++ Lite Uninstaller on GNU/Linux or Mac OS X

You should use the provided uninstaller to remove a Symbian ADT Sourcery G++ Lite installation originally created by the executable installer script. The `arm-none-symbianelf` directory located in the install directory will be removed entirely by the uninstaller. Please back up any changes you have made to this directory, such as modified linker scripts.

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 Symbian ADT 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 Symbian ADT 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
Symbian ADT Sourcery G++ Lite for ARM SymbianOS

This chapter contains information about features of Symbian ADT Sourcery G++ Lite that are specific to ARM SymbianOS targets. You should read this chapter to learn how to best use Symbian ADT Sourcery G++ Lite on your target system.

3.1. Included Components and Features

This section briefly lists the important components and features included in Symbian ADT Sourcery G++ Lite for ARM SymbianOS, and tells you where you may find further information about these features.

Component	**Version**	**Notes**
GNU programming tools		
GNU Compiler Collection	4.4.1	Separate manual included.
GNU Binary Utilities	2.19.51	Includes assembler, linker, and other utilities. Separate manuals included.
Debugging support and simulators		
GNU Debugger	6.8.50	Separate manual included.
Sourcery G++ Debug Sprite for ARM	4.4-172	See Chapter 5, "Sourcery G++ Debug Sprite".
QEMU Emulator	0.11.50	See Section 4.3, "Running Applications with QEMU Platform Emulator".
Target libraries		
Other utilities		
GNU Make	N/A	Build support on Windows hosts.
GNU Core Utilities	N/A	Build support on Windows hosts.

3.2. Library Configurations

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

3.2.1. Included Libraries

The following library configurations are available in Symbian ADT Sourcery G++ Lite for ARM SymbianOS.

ARMv5 - Little-Endian, Soft-Float	
Command-line option(s):	default

ARMv5 - Little-Endian, VFP	
Command-line option(s):	`-mfloat-abi=softfp`

3.2.2. Library Selection

A given multilib may be compatible with additional processors and build options beyond those listed above. However, even if a particular set of command-line options produces code compatible with one of the provided multilibs, those options may not be sufficient to identify the intended library to the linker. For example, on some targets, specifying only a processor option on the command line may imply architecture features or floating-point support for compilation, but not for library selection.

The details of the mapping from command-line options to multilibs are target-specific and quite complex. Therefore, it is recommended that your link command line include exactly the options listed in the tables above for your intended target multilib. In some cases, you may need to supply different options for linking than for compilation.

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

```
> arm-none-symbianelf-gcc -print-multi-directory options...
```

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

3.3. Building SymbianOS Programs

Building programs for SymbianOS requires you install additional software and follow the SymbianOS build procedure.

You must install the Symbian SDK[1]. For Linux and OSX hosts, you will have to install the SDK on a Windows machine and then make the file system visible on your Linux or OSX host. Alternatively, for Linux hosts, the GnuPoc[2] project provides patches. Set the environment variable `EPOCROOT` to the directory containing the `epoc32` directory of your Symbian SDK installation, and also ensure your `PATH` variable includes the `$EPOCROOT/epoc32/tools` directory. The following commands also make use of `epoclib` and `epocarch` variables for convenience. For instance, if you have installed the SDK at `/opt/symbian-sdk`, enter the following commands:

```
> export EPOCROOT=/opt/symbian-sdk/s60
> PATH=$EPOCROOT/epoc32/tools:$PATH
> epocinc=$EPOCROOT/epoc32/include
> epocarch=$EPOCROOT/epoc32/release/armv5
```

SymbianOS programs do not start at `main`, but at `E32Main`. Using an editor (such as `notepad` on Microsoft Windows or `vi` on UNIX-like systems), create a file named `main.cc` containing the following console program:

```
#include <e32base.h>
#include <e32cons.h>

_LIT (KTxtEPOC32EX, "EXAMPLES");
_LIT (KTxtExampleCode, "Symbian OS Example Code");
_LIT (KTxtOK, "ok [press any key]");

LOCAL_D CConsoleBase* console;

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

[1] http://developer.symbian.org/wiki/index.php/Symbian_C%2B%2B_Quick_Start
[2] http://gnupoc.sourceforge.net/

```
LOCAL_C void callExampleL () {
  console = Console::NewL
    (KTxtExampleCode,
     TSize (KConsFullScreen, KConsFullScreen));
  CleanupStack::PushL (console);

  _LIT (KHelloWorldText, "Hello world!\n");
  console->Printf (KHelloWorldText);
  for (int i = 0; i < 10; ++i) {
    int n = factorial (i);
    _LIT (KFactorialText, "factorial(%d) = %d\n");
    console->Printf (KFactorialText, i, n);
  }

  console->Printf (KTxtOK);
  console->Getch ();
  CleanupStack::PopAndDestroy ();
}

GLDEF_C TInt E32Main () {
  __UHEAP_MARK;
  CTrapCleanup *cleanup = CTrapCleanup::New ();
  TRAPD (error, callExampleL ());
  __ASSERT_ALWAYS (!error, User::Panic (KTxtEPOC32EX, error));
  delete cleanup;
  __UHEAP_MARKEND;
  return 0;
}
```

To compile your program use the following command:

```
> arm-none-symbianelf-g++ -march=armv5t -mthumb -mapcs -nostdinc \
  -D__MARM__ -D__MARM_ARMV5__ -D__MARM_THUMB__ \
  -D__MARM_INTERWORK__ -D__EABI__ -D__EXE__ \
  -D_DEBUG -D_UNICODE -D__SUPPORT_CPP_EXCEPTIONS__ \
  -D__GCCE__ -D__SYMBIAN32__ -D__EPOC32__ \
  -D__S60_50__ -D__S60_3X__ -D__SERIES60_3X__ \
  -D__PRODUCT_INCLUDE__=\"$epocinc/variant/symbian_os.hrh\" \
  -include $epocinc/gcce/gcce.h \
  -I $epocinc/libc -I $epocinc -I $epocinc/variant \
  -c -g -o main.o main.cc
```

You may see some warnings. These are from Symbian SDK header files, not Sourcery G++ files.

You can link your application with:

```
> arm-none-symbianelf-g++ -march=armv5t -mthumb -mapcs -nostdlib \
  -Wl,--target1-abs -Wl,--no-undefined \
  -Wl,-Ttext,0x8000 -Wl,-Tdata,0x400000 \
  -Wl,--default-symver -Wl,-soname,"factorial{000a0000}.exe" \
  -Wl,--entry,_E32Startup -Wl,-u,_E32Startup \
  $epocarch/udeb/eexe.lib \
  -shared -g -o factorial.sym main.o \
  -Wl,"-(" -Wl,$epocarch/udeb/usrt2_2.lib \
  -Wl,$epocarch/udeb/ecrt0.lib -Wl,"-)" \
  -Wl,$epocarch/lib/estlib.dso \
  -Wl,$epocarch/lib/euser.dso \
  -Wl,$epocarch/lib/dfpaeabi.dso \
  -Wl,$epocarch/lib/dfprvct2_2.dso \
  -Wl,$epocarch/lib/drtaeabi.dso \
  -Wl,$epocarch/lib/scppnwdl.dso \
  -Wl,$epocarch/lib/drtrvct2_2.dso \
  -lsupc++ -lgcc
```

This produces a `factorial.sym` file that can be used by `arm-none-symbianelf-gdb`.

To run the program on SymbianOS, you must convert this file to EPOC32 format using the `elf2e32` command. The `elf2e32` is part of the Symbian SDK and not part of Sourcery G++. If you are using a Linux or OSX host, and did not install GnuPoc, you must install Wine[3] and invoke `elf2e32` as:

```
> wine $EPOCROOT/epoc32/tools/elf2e32.exe other options
```

The following command creates `factorial.exe`:

```
> elf2e32 --sid=0x00000000 --version=10.0 --uid1=0x1000007a \
  --uid2=0xe8000075 --uid3=0x00000000 --vid=0x70000001 \
  --capability=none --fpu=softvfp --targettype=EXE \
  --output="factorial.exe" --elfinput="factorial.sym" \
  --linkas="factorial{000a0000}.exe" \
  --libpath="$epocarch/lib"
```

Refer to Section 4.3, “Running Applications with QEMU Platform Emulator” for instructions on running your program using QEMU. See Section 4.4, “Running Applications from GDB” for details of using arm-none-symbianelf-gdb to debug your program. Refer to Chapter 5, “Sourcery G++ Debug Sprite” for information on how to connect arm-none-symbianelf-gdb to your target.

3.4. SymbianOS Runtime Libraries

Symbian ADT Sourcery G++ Lite does not include C or C++ runtime libraries for SymbianOS. These are provided separately by Symbian.

3.5. 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] http://www.winehq.org/

3.6. Half-Precision Floating Point

Sourcery G++ for ARM SymbianOS 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. The included QEMU emulator also supports the hardware instructions when invoked with the `any` CPU specifier.

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.

3.7. 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[4]:

- BSABI - ARM IHI 0036B (10 October 2008)
- BPABI - ARM IHI 0037B (10 October 2008)
- EHABI - ARM IHI 0038A (10 October 2008)
- CLIBABI - ARM IHI 0039A (10 October 2008)
- AADWARF - ARM IHI 0040A (10 October 2008)
- CPPABI - ARM IHI 0041B (10 October 2008)
- AAPCS - ARM IHI 0042C (10 October 2008)
- RTABI - ARM IHI 0043B (10 October 2008)
- AAELF - ARM IHI 0044C (10 October 2008)
- ABI Addenda - ARM IHI 0045B (10 October 2008)

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

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

Chapter 4
Using Sourcery G++ from the Command Line

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

4.1. Building an Application

This chapter explains how to build an application with Symbian ADT Sourcery G++ Lite using the command line. As elsewhere in this manual, this section assumes that your target system is arm-none-symbianelf, as indicated by the `arm-none-symbianelf` command prefix.

Building programs for SymbianOS requires unique command-line arguments and build steps to integrate with the Symbian SDK; refer to Chapter 3, “Symbian ADT Sourcery G++ Lite for ARM SymbianOS” for details.

4.2. Running Applications on the Target System

Consult your target board documentation for instructions on loading programs onto the target, and running them. Alternatively, you can use the Sourcery G++ Debug Sprite from within GDB to download and run programs on the target via a supported hardware debugging device.

4.3. Running Applications with QEMU Platform Emulator

Symbian ADT Sourcery G++ Lite includes the QEMU emulator. This is a program which runs on your host computer and allows you to run and debug ARM SymbianOS applications without target hardware.

The QEMU emulator is invoked as follows:

```
> arm-none-symbianelf-qemu-system \
  -kernel kernel-image -M device-tree \
  [-serial serial-ports]
```

The *`kernel-image`* file is the SymbianOS kernel image you wish to use. The *`device-tree`* file describes the emulated devices that are to be provided.

The optional *`serial-ports`* specifies how virtual serial ports are made available. For example, a port is required for debugging. The following serial ports are available:

`vc`	Provide a virtual console.
`tcp:`*`hostname`*`:`*`portnum`*`[,`*`options`*`]`	Provide a TCP connection listening on *`hostname`*:*`portnum`*. You can optionally specify a comma-separated list of *`options`* keywords: `server` Listen for incoming requests. `nowait` Disable the Nagle wait delay heuristic. This improves the latency of interactive connections at the expense of bandwidth.

For instance, to start a SymbianOS emulation with the first virtual serial port connected to a virtual console and the second listening on port 5555 on the local host, use:

```
> arm-none-symbianelf-qemu-system \
  -kernel syborg_001.TextShell.IMG -M syborg.dtb \
  -serial vc -serial tcp:127.0.0.1:5555,server,nowait
```

Refer to Syborg documentation for information on creating the kernel image and device tree.

The version of QEMU included with Symbian ADT Sourcery G++ Lite for ARM SymbianOS is configured to run in platform emulation mode only, and other QEMU features not documented here are not supported in Symbian ADT Sourcery G++ Lite. For additional information about QEMU, visit the QEMU web site[1].

4.4. 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. GDB can also be used to run and debug programs with QEMU, a simulator that runs on your host 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-none-symbianelf-gdb factorial.sym
```

For SymbianOS you must specify the ELF binary, not the EPOC32 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.

4.4.1. 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-none-symbianelf-sprite arguments
```

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

4.4.2. Connecting to an External GDB Server

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.

[1] http://fabrice.bellard.free.fr/qemu

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 programs running under SymbianOS. This chapter includes information about the debugging devices and boards supported by the Sprite for ARM SymbianOS.

Symbian ADT Sourcery G++ Lite contains the Sourcery G++ Debug Sprite for ARM SymbianOS. This Sprite is provided to allow debugging of programs running on a SymbianOS target or simulator.

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 Symbian ADT Sourcery G++ Lite.

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-none-symbianelf-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
    (Symbian ADT Sourcery G++ Lite 4.4-172)
armusb: [speed=<n:0-7>] ARMUSB 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-none-symbianelf-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 Symbian ADT Sourcery G++ Lite for ARM SymbianOS:

`trk` Use SymbianOS TRK interface. Refer to Section 5.4, “SymbianOS TRK Interface”.

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 5.6, “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 5.7, “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-none-symbianelf-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. SymbianOS TRK Interface

The Debug Sprite supports debugging SymbianOS programs via the Target Resident Kernel (TRK) interface. The *`device-url`* is:

```
trk://host[:port][/target-program]
```

The *`host`* and *`port`* indicate the TRK server location. The *`target-program`* specifies the location of the program you wish to debug, on the remote target's file system.

In order to connect the Debug Sprite to the SymbianOS system, you must start the SymbianOS debug server on your target. Enter the following in your target console:

```
> trkconsole
```

For example, if you started a platform simulator with the `-serial tcp:127.0.0.1:5555,server,nowait` option (see Section 4.3, “Running Applications with QEMU Platform Emulator” for how to start the QEMU simulator) you can invoke the Debug Sprite on the same host with the following command from GDB:

```
(gdb) target extended-remote | \
    arm-none-symbianelf-sprite trk://127.0.0.1:5555 symbian
```

Then use the following commands to copy your program to the target, and start it:

```
(gdb) remote put program.exe "\\sys\\bin\\program.exe"
...
(gdb) set remote exec-file \sys\bin\program.exe
...
(gdb) run
```

The different quoting in the `remote put` and the `set remote exec-file` commands is deliberate. Normal SymbianOS kernels can only run programs from the `\sys\bin` directory.

5.5. 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-none-symbianelf-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].

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

5.6. Supported Board Files

The Sourcery G++ Debug Sprite for ARM SymbianOS 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**
Symbian ADT	`symbian`

5.7. 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-none-symbianelf/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 Symbian ADT 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. Manuals for GNU Toolchain Components

Symbian ADT 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 Symbian ADT 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 Symbian ADT Sourcery G++ Lite, the documentation can be found in the `share/doc/sourceryg++-arm-none-symbianelf/` subdirectory of your installation directory.

In addition to the detailed reference manuals, Symbian ADT 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-none-symbianelf/man/man1
```

Then you can invoke `man` as:

```
> man ./arm-none-symbianelf-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 Symbian ADT 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 Symbian ADT Sourcery G++ Lite Release Notes

This appendix contains information about changes in this release of Symbian ADT Sourcery G++ Lite for ARM SymbianOS. You should read through these notes to learn about new features and bug fixes.

A.1. Changes in Symbian ADT Sourcery G++ Lite for ARM SymbianOS

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

A.1.1. Changes in Symbian ADT Sourcery G++ Lite 4.4-172

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

A.1.2. Changes in Symbian ADT Sourcery G++ Lite 4.4-171

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

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

QEMU fails to start. Two bugs have been fixed that resulted in QEMU failing to start on Windows hosts due to unresolved symbols, and on Mac OS X hosts due to missing keymaps.

A.1.3. Changes in Symbian ADT Sourcery G++ Lite 4.4-164

Linker performance improvement. A bug in the linker that caused applications with many input files to link slowly has been fixed.

Assembler segmentation fault fix. A bug has been fixed that caused the assembler to crash when processing some data filling directives, such as `.fill 0, 0, 0`.

Improved support for debugging RealView® programs . GDB has been enhanced to handle some debug information contained in binaries produced by the ARM RealView® compiler. Formerly, GDB sometimes crashed on these programs and libraries.

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

A.1.4. Changes in Symbian ADT Sourcery G++ Lite 4.4-129

No significant changes. There are no significant changes for ARM SymbianOS in this release.

A.1.5. Changes in Symbian ADT Sourcery G++ Lite 4.4-106

Thumb-2 internal compiler error fix. A bug that caused an internal compiler error when building the QT library for Thumb-2 has been fixed.

Thumb-2 multiply fix. A bug that caused an invalid `muls` instruction to be generated in certain circumstances has been fixed. This affected code compiled for Thumb-2, and resulted in an error from the assembler.

A.1.6. Changes in Symbian ADT Sourcery G++ Lite 4.4-102

Initial release. This is the initial release for ARM SymbianOS.

Appendix B
Symbian ADT Sourcery G++ Lite Licenses

Symbian ADT 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 Symbian ADT Sourcery G++ Lite. You should read this appendix to understand your legal rights and obligations as a user of Symbian ADT Sourcery G++ Lite.

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

The table below lists the major components of Symbian ADT Sourcery G++ Lite for ARM SymbianOS 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 Symbian ADT Sourcery G++ Lite. Symbian ADT 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 Symbian ADT Sourcery G++ Lite.

Component	**License**
GNU Compiler Collection	GNU General Public License 3.0 [1]
GNU Binary Utilities	GNU General Public License 3.0 [2]
GNU Debugger	GNU General Public License 3.0 [3]
Sourcery G++ Debug Sprite for ARM	CodeSourcery License
QEMU Emulator	GNU General Public License 2.0 [4]
GNU Make	GNU General Public License 2.0 [5]
GNU Core Utilities	GNU General Public License 2.0 [6]

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

Important

Although some of the licenses that apply to Symbian ADT 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 Symbian ADT Sourcery G++ Lite. You can develop proprietary applications and libraries with Symbian ADT Sourcery G++ Lite.

Symbian ADT Sourcery G++ Lite may include some third party example programs and libraries in the `share/sourceryg++-arm-none-symbianelf-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.

[1] http://www.gnu.org/licenses/gpl.html
[2] http://www.gnu.org/licenses/gpl.html
[3] http://www.gnu.org/licenses/gpl.html
[4] http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
[5] http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
[6] http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

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

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

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

3. **Definitions.**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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