Sourcery CodeBench Lite

ARM GNU/Linux

Sourcery CodeBench Lite 2013.05-24

Getting Started

Sourcery CodeBench Lite: ARM GNU/Linux: Sourcery CodeBench Lite 2013.05-24: Getting Started

Mentor Graphics, Inc.

Abstract

This guide explains how to install and build applications with Sourcery CodeBench Lite, CodeSourcery's customized and validated version of the GNU Toolchain. Sourcery CodeBench 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 CodeBench from the command line.

Table of Contents

Preface iv
1. Intended Audience v
2. Organization v
3. Typographical Conventions v
1. Quick Start 1
1.1. Installation and Set-Up 2
1.2. Configuring Sourcery CodeBench 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 CodeBench Lite 6
2.5. Installing Sourcery CodeBench Lite Updates 9
2.6. Setting up the Environment 9
2.7. Customer Experience Improvement Program 11
2.8. Uninstalling Sourcery CodeBench Lite 12
3. Sourcery CodeBench Lite for ARM GNU/Linux 14
3.1. Included Components and Features 15
3.2. Library Configurations 15
3.3. Compiling for ARMv4T and ARMv5T Systems 16
3.4. Target Kernel Requirements 16
3.5. Target Dynamic Loader Requirements 16
3.6. Using Sourcery CodeBench Lite on GNU/Linux Targets 17
3.7. Using GDB Server for Debugging 19
3.8. GLIBC Backtrace Support 21
3.9. Using VFP Floating Point 21
3.10. Fixed-Point Arithmetic 22
3.11. ABI Compatibility 23
3.12. Object File Portability 24
4. Using Sourcery CodeBench from the Command Line 25
4.1. Building an Application 26
4.2. Running Applications on the Target System 26
4.3. Running Applications from GDB 27
4.4. Using the Compiler Cache 27
5. Next Steps with Sourcery CodeBench 29
5.1. Sourcery CodeBench Knowledge Base 30
5.2. Manuals for GNU Toolchain Components 30
A. Sourcery CodeBench Lite Release Notes 31
A.1. Changes in Sourcery CodeBench Lite for ARM GNU/Linux 32
B. Sourcery CodeBench Lite Licenses 38
B.1. Sourcery CodeBench Lite License Agreement 39
B.2. Licenses for Sourcery CodeBench Lite Components 49
B.3. Attribution 50

Preface

This preface introduces the Sourcery CodeBench 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 CodeBench Lite. This guide provides a step-by-step guide to installing Sourcery CodeBench 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 CodeBench 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 CodeBench 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 CodeBench Lite for ARM GNU/Linux"	This chapter contains information about using Sourcery CodeBench Lite that is specific to ARM GNU/Linux targets. You should read this chapter to learn how to best use Sourcery CodeBench Lite on your target system.
Chapter 4, "Using Sourcery CodeBench from the Command Line"	This chapter explains how to build applications with Sourcery CodeBench Lite using the command line. In the process of reading this chapter, you will build a simple application that you can use as a model for your own programs.
Chapter 5, "Next Steps with Sourcery CodeBench"	This chapter describes where you can find additional documentation and information about using Sourcery CodeBench Lite and its components. It also provides information about Sourcery CodeBench subscriptions. CodeSourcery customers with Sourcery CodeBench subscriptions receive comprehensive support for Sourcery CodeBench.
Appendix A, "Sourcery CodeBench Lite Release Notes"	This appendix contains information about changes in this release of Sourcery CodeBench Lite for ARM GNU/Linux. You should read through these notes to learn about new features and bug fixes.
Appendix B, "Sourcery CodeBench Lite Licenses"	This appendix provides information about the software licenses that apply to Sourcery CodeBench Lite. Read this appendix to understand your legal rights and obligations as a user of Sourcery CodeBench 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 CodeBench Lite for the first time. You may use this chapter as an abbreviated guide to the rest of this manual.

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

Follow the steps given in this chapter to install Sourcery CodeBench 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 CodeBench Lite in Chapter 3, "Sourcery CodeBench Lite for ARM GNU/Linux".

1.1. Installation and Set-Up

Install Sourcery CodeBench Lite on your host computer. You may download an installer package from the Sourcery CodeBench 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 CodeBench 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 CodeBench Lite for the Target System

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

Install runtime libraries on your target machine. In order to run programs built with the Sourcery CodeBench runtime libraries on target hardware, you must install these libraries, the Sourcery CodeBench dynamic linker, and other runtime support files -- collectively referred to as the *sysroot* -- on your GNU/Linux target system. Typically, this involves either using third-party tools to build a complete root filesystem including the Sourcery CodeBench sysroot, or using special commands when linking or running your program so it can find the sysroot installed in another location on the target. Refer to Section 3.6, "Using Sourcery CodeBench Lite on GNU/Linux Targets" for full discussion of these options.

1.3. Building Your Program

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

1.4. Running and Debugging Your Program

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

[1] http://go.mentor.com/codebench/

Run your program on the ARM GNU/Linux target. To run a program using the included Sourcery CodeBench libraries, you must install the sysroot on the target, as previously discussed. Copy the executable for your program to the target system. The method you use for launching your program depends on how you have installed the libraries and built your program. In some cases, you may need to invoke the Sourcery CodeBench dynamic linker explicitly. Refer to Section 3.6, "Using Sourcery CodeBench Lite on GNU/Linux Targets" for details.

Debug your program on the target using GDB server. You can use GDB server on a remote target to debug your program. When debugging a program that uses the included Sourcery CodeBench libraries, you must use the `gdbserver` executable included in the sysroot, and similar issues with respect to the dynamic linker as discussed previously apply. See Section 3.7, "Using GDB Server for Debugging" for detailed instructions. Once you have started GDB server on the target, you can connect to it from the debugger on your host system. Refer to Section 4.3, "Running Applications from GDB" for instructions on remote debugging from command-line GDB.

Chapter 2
Installation and Configuration

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

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

2.1. Terminology

Throughout this document, the term *host system* refers to the system on which you run Sourcery CodeBench while the term *target system* refers to the system on which the code produced by Sourcery CodeBench runs. The target system for this version of Sourcery CodeBench is `arm-none-linux-gnueabi`.

If you are developing a workstation or server application to run on the same system that you are using to run Sourcery CodeBench, 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 CodeBench supports the following host operating systems and architectures:

- Microsoft Windows XP (SP1), 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 5 (and later), Red Hat Enterprise Linux 5 (and later), SuSE Enterprise Linux 10 (and later), and Ubuntu 8.04 (and later).

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

Installing on Ubuntu and Debian GNU/Linux Hosts

The Sourcery CodeBench 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 CodeBench 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 CodeBench Lite toolchain.

2.2.2. Host Hardware Requirements

The amount of disk space required for a complete Sourcery CodeBench 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 CodeBench Lite for ARM GNU/Linux" for requirements that apply to the target system.

2.3. Downloading an Installer

If you have received Sourcery CodeBench 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 CodeBench Lite".

You can download Sourcery CodeBench Lite from the Sourcery CodeBench web site[1]. This free version of Sourcery CodeBench, 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 CodeBench 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 CodeBench installer is provided as an executable with the `.exe` extension. For GNU/Linux systems Sourcery CodeBench 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 CodeBench Lite

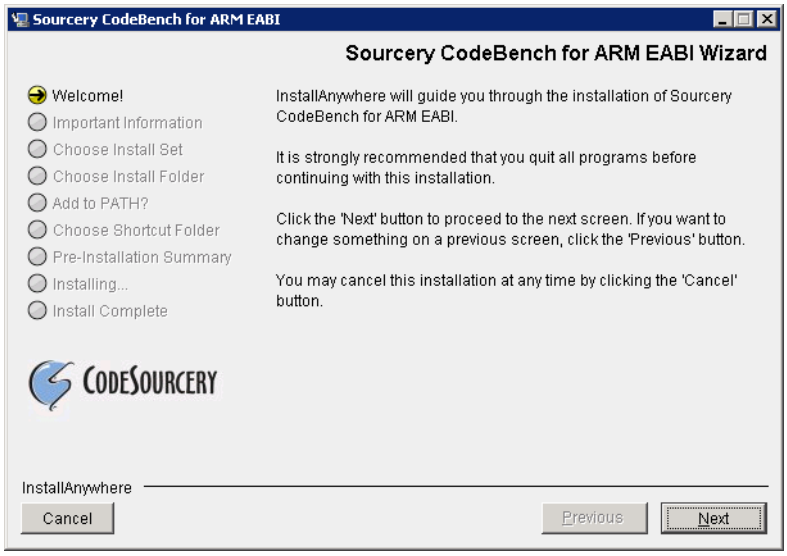
The method used to install Sourcery CodeBench Lite depends on your host system and the kind of installation package you have downloaded.

2.4.1. Using the Sourcery CodeBench Lite Installer on Microsoft Windows

If you have received Sourcery CodeBench 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 CodeBench Lite, double-click on the installer.

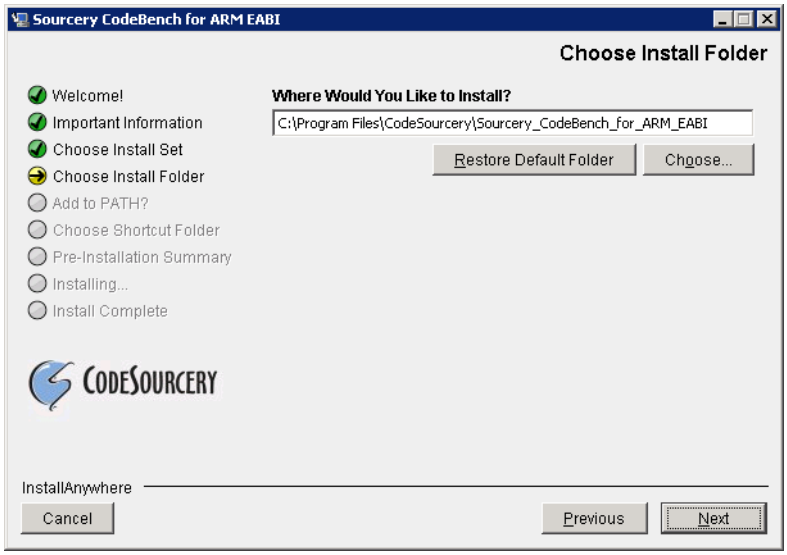
After the installer starts, follow the on-screen dialogs to install Sourcery CodeBench Lite. The installer is intended to be self-explanatory and on most pages the defaults are appropriate.

[1] http://go.mentor.com/codebench/
[2] https://sourcery.mentor.com/GNUToolchain/

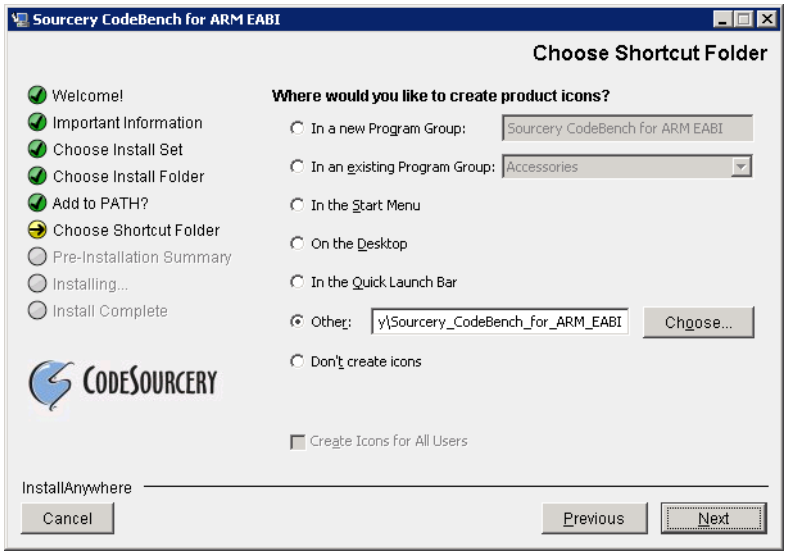


Running the Installer. The graphical installer guides you through the steps to install Sourcery CodeBench Lite.

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

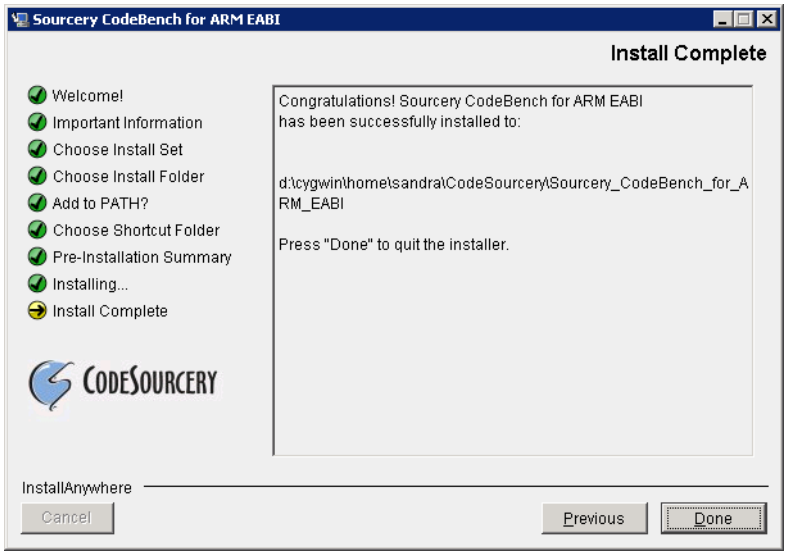


Choose Install Folder. Select the pathname to your install directory.



Choose Shortcut Folder. You can customize where the installer creates shortcuts for quick access to Sourcery CodeBench 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 CodeBench 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 CodeBench 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 CodeBench Lite from a Compressed Archive

You do not need to be a system administrator to install Sourcery CodeBench Lite from a compressed archive. You may install Sourcery CodeBench 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 CodeBench 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++-2013.05`.

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 CodeBench Lite Updates

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

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 CodeBench commands for the ARM GNU/Linux target all begin with `arm-none-linux-gnueabi`. This means that you can install Sourcery CodeBench 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

The graphical installer for Sourcery CodeBench Lite does this setup for you, however it may not take effect until you next log in.

In order to use the Sourcery CodeBench tools from the command line, you should add them to your PATH. In the instructions that follow, replace *installdir* with the full pathname of your Sourcery CodeBench Lite installation directory, including the drive letter.

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

```
> setx PATH "%PATH%;installdir\bin"
```

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 Edit. Add the string ;*installdir*\bin to the end, and click OK.

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 ;*installdir*\bin to the end, and click OK.

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

```
> arm-none-linux-gnueabi-gcc -v
```

Verify that the last line of the output contains: Sourcery CodeBench Lite 2013.05-24.

2.6.1.2. Working with Cygwin

Sourcery CodeBench Lite does not require Cygwin or any other UNIX emulation environment. You can use Sourcery CodeBench directly from the Windows command shell. You can also use Sourcery CodeBench 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 CodeBench is not a Cygwin application, it does not, by default, recognize Cygwin paths.

If you are using Sourcery CodeBench from Cygwin, you should set the CYGPATH environment variable. If this environment variable is set, Sourcery CodeBench 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 CodeBench relies on the cygpath utility provided with Cygwin. You must provide Sourcery CodeBench with the full path to cygpath if cygpath is not in your PATH. For example:

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

directs Sourcery CodeBench 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

The graphical installer for Sourcery CodeBench Lite does this setup for you, however it may not take effect until you next log in.

Before using Sourcery CodeBench 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 installdir/bin:$PATH
```

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

```
> PATH=installdir/bin:$PATH
> export PATH
```

If you are not sure which shell you are using, try both commands. In both cases, replace *`installdir`* with the full pathname of your Sourcery CodeBench Lite installation directory.

You may also wish to set the `MANPATH` environment variable so that you can access the Sourcery CodeBench manual pages, which provide additional information about using Sourcery CodeBench. To set the `MANPATH` environment variable, follow the same steps shown above, replacing `PATH` with `MANPATH`, and `bin` with `share/doc/arm-arm-none-linux-gnueabi/man`.

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

```
> arm-none-linux-gnueabi-gcc -v
```

Verify that the last line of the output contains: `Sourcery CodeBench Lite 2013.05-24`.

2.7. Customer Experience Improvement Program

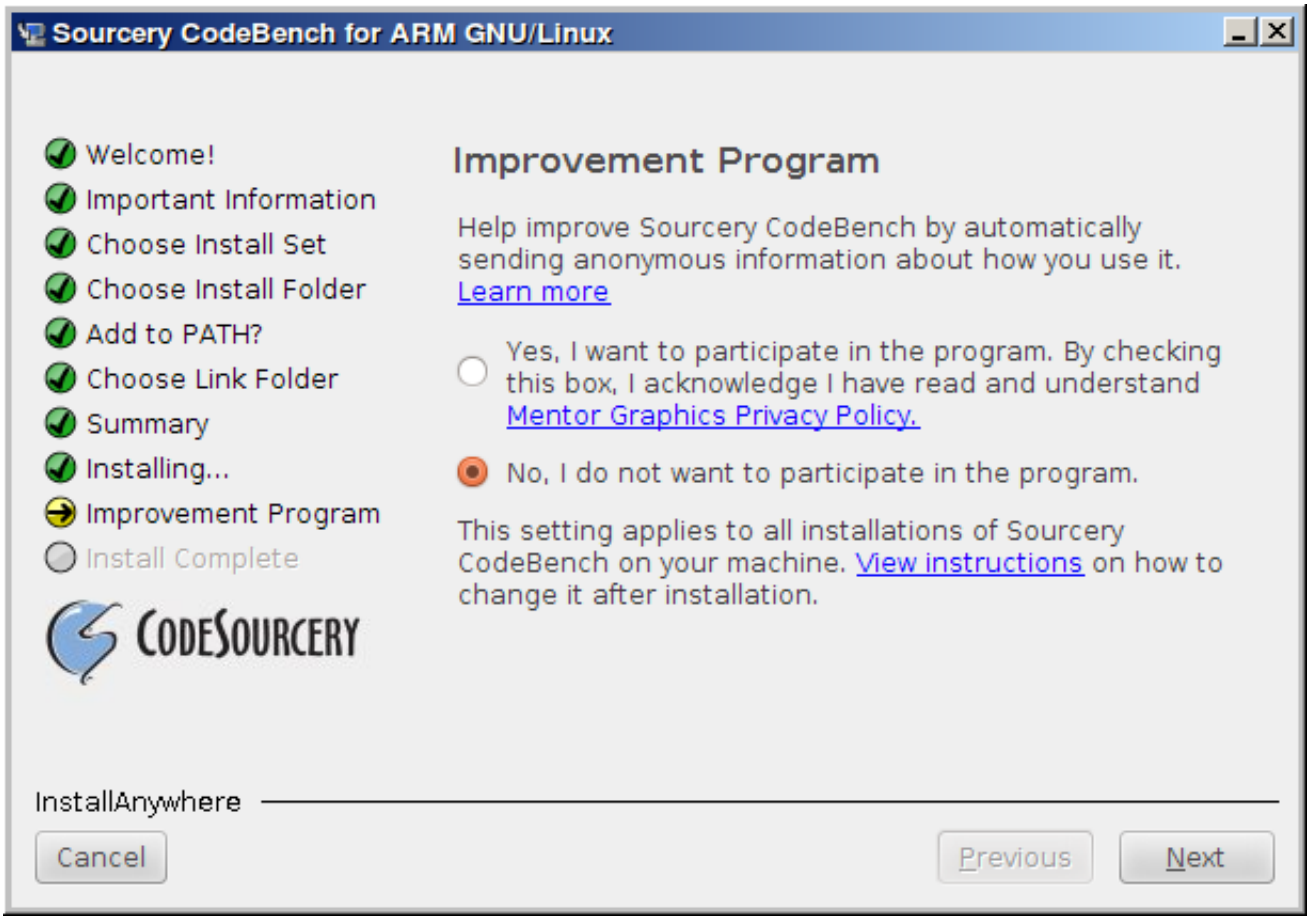
Opting into the Customer Experience Improvement Program (CEIP) permits Mentor Graphics to collect anonymous information about how you use Sourcery CodeBench Lite. For more information, please see the following web pages:

- CEIP Details[3].
- Privacy Policy[4].

You can opt in or out of the CEIP in any of the following ways.

- Check the box in the graphical installer.

[3] http://go.mentor.com/scbceip
[4] http://go.mentor.com/mentpp



Installer CEIP page.

This affects your personal opt-in settings only, and does not affect those of other users. If you have multiple instances of Sourcery CodeBench Lite installed, this setting applies to all of them.

- Set the configuration via the command line.

  To opt in:

  ```
  > arm-none-linux-gnueabi-cs -O cloud_mode=online
  ```

  Or, to opt out:

  ```
  > arm-none-linux-gnueabi-cs -O cloud_mode=offline
  ```

  This affects your personal opt-in settings for all installed instances of Sourcery CodeBench Lite.

  These commands are equivalent to editing `$HOME/.cs.conf`. It's also possible to edit this setting system-wide in `/opt/codesourcery/etc/cs.conf`, or per cache in *`cache_dir`*`/cs.conf`. For more information, see `man cs`.

2.8. Uninstalling Sourcery CodeBench Lite

The method used to uninstall Sourcery CodeBench 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-none-linux-gnueabi` directory located in the install directory will be removed entirely by the uninstaller.

2.8.1. Using the Sourcery CodeBench Lite Uninstaller on Microsoft Windows

You should use the provided uninstaller to remove a Sourcery CodeBench 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 CodeBench Lite.

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

To uninstall third-party drivers bundled with Sourcery CodeBench 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.

2.8.2. Using the Sourcery CodeBench Lite Uninstaller on GNU/Linux

You should use the provided uninstaller to remove a Sourcery CodeBench 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 CodeBench 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.

2.8.3. Uninstalling a Compressed Archive Installation

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

Chapter 3
Sourcery CodeBench Lite for ARM GNU/Linux

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

3.1. Included Components and Features

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

Component	Version	Notes
GNU programming tools		
GNU Compiler Collection	4.7.3	Separate manual included.
GNU Binary Utilities	2.23.52	Includes assembler, linker, and other utilities. Separate manuals included.
Debugging support and simulators		
GNU Debugger	7.4.50	Separate manual included.
GDB Server	N/A	Included with GDB. See Section 3.7, “Using GDB Server for Debugging”.
Target libraries		
GNU C Library	2.17	Separate manual included.
Linux Kernel Headers	3.8.2	
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 CodeBench Lite for ARM GNU/Linux includes the following library configuration.

ARMv5TE - Little-Endian, Soft-Float, GLIBC	
Command-line option(s):	default
Sysroot subdirectory:	`./`
Dynamic linker:	`lib/ld-linux.so.3`

ARMv4T - Little-Endian, Soft-Float, GLIBC	
Command-line option(s):	`-march=armv4t`
Sysroot subdirectory:	`armv4t/`
Dynamic linker:	`lib/ld-linux.so.3`
Notes:	This should also be used for ARMv5T cores such as the ARM1020T.

ARMv7-A Thumb-2 - Little-Endian, Soft-Float, GLIBC	
Command-line option(s):	`-mthumb -march=armv7-a`
Sysroot subdirectory:	`thumb2/`
Dynamic linker:	`lib/ld-linux.so.3`

Sourcery CodeBench 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 CodeBench 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. The sysroot contains the dynamic linker used to run your applications on the target as well as the libraries. Refer to Section 3.6, "Using Sourcery CodeBench Lite on GNU/Linux Targets" for instructions on how to install and use these support files on your target GNU/Linux system. You can find the sysroot directories provided with Sourcery CodeBench in the `arm-none-linux-gnueabi/libc` directory of your installation. In the tables below, the dynamic linker pathname is given relative to the corresponding sysroot.

3.3. Compiling for ARMv4T and ARMv5T Systems

By default Sourcery CodeBench generates Linux binaries that require an ARMv5TE or later CPU. To build applications or libraries capable of running on ARMv4T or early ARMv5 CPUs, use the `-march=armv4t` or `-march=armv5t` command-line options. These options also select libraries for ARMv4T processors; see Section 3.2, "Library Configurations" for details.

Code compiled for ARMv4T is ABI compatible with ARMv5 code. Code and binaries compiled for different architectures may be mixed freely.

3.4. Target Kernel Requirements

The GNU C library supplied with Sourcery CodeBench Lite uses the EABI-based kernel syscall interface. This means applications compiled with Sourcery CodeBench require at least a 2.6.16 kernel with EABI syscalls enabled.

To provide VFP and Advanced SIMD registers, `gdbserver` requires support from the Linux kernel. Linux 2.6.30 includes the necessary support; for older versions, visit the Sourcery CodeBench Knowledge Base[1].

3.5. Target Dynamic Loader Requirements

The compiler supplied in Sourcery CodeBench Lite emits TLS Descriptor sequences to access thread-local storage in position-independent code. This is a new TLS access model, with a specification at http://sourcery.mentor.com/public/publications/RFC-TLSDESC-ARM.txt. It improves the performance of shared objects and position-independent executables. This model requires dynamic loader support. The loader included with Sourcery CodeBench Lite includes the necessary support. Support for the older ARM EABI-specified access sequence is still provided and thus object files and executables built by EABI-compliant toolchains, including earlier versions of Sourcery CodeBench Lite, will continue to function. If you need to use an older dynamic loader that lacks TLS Descriptor support, you must compile all your code with `-mtls-dialect=gnu`. This option selects the previous TLS access method.

[1] https://sourcery.mentor.com/GNUToolchain/kbentry117

3.6. Using Sourcery CodeBench Lite on GNU/Linux Targets

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

The runtime support files, referred to as the *sysroot*, are found in the `arm-none-linux-gnueabi/libc` directory of your Sourcery CodeBench Lite installation. The sysroot consists of the contents of the `etc`, `lib`, `sbin`, and `usr` directories. There may be other directories in `arm-none-linux-gnueabi/libc` that contain additional sysroots customized for particular combinations of command-line compiler flags, or *multilibs*. Refer to Section 3.2, "Library Configurations" for a list of the included multilibs in this version of Sourcery CodeBench Lite, and the corresponding sysroot directory pathnames.

Note for Windows Host Users

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

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

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

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

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

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

3.6.1. Installing the Sysroot

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

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

The next step is to identify the correct sysroot subdirectory in the Sourcery CodeBench Lite install directory on your host system. The sysroot you copy to the target must be the one that corresponds to the linker options you are using to build your applications. The tables in Section 3.2, "Library Configurations" tell you which sysroot subdirectories correspond to which sets of command-line options. From the command line, you can identify the appropriate sysroot for your program by invoking the compiler with `-print-sysroot` added to your other build options. This causes GCC to print the host sysroot pathname and exit.

The mechanism you use for copying the sysroot to your target board depends on its hardware and software configuration. You may be able to use FTP or SSH with a server already running on your target. If your target board does not have networking configured, you may be able to copy files using an SD card or USB memory stick, or via a file transfer utility over a serial line. The instructions that come with your board may include specific suggestions.

When running Sourcery CodeBench on a GNU/Linux host, as an alternative to copying files to the target system, you may be able to NFS-mount the Sourcery CodeBench Lite installation directory from your host system on the target system. It is especially convenient for debugging if you can make the sysroot pathname on the target system be identical to that on the GNU/Linux host system; refer to Section 3.7.3, "Setting the Sysroot in the Debugger" for further discussion of this issue.

Otherwise, you must copy files from the appropriate sysroot subdirectory in the `arm-none-linux-gnueabi/libc` directory of your Sourcery CodeBench Lite install to the target system. In many cases, you do not need to copy all of the files in the sysroot. For example, the `usr/include` subdirectory contains files that are only needed if you will actually be running the compiler on your target system. You do not need these files for non-native compilers. You also do not need any `.o` or `.a` files; these are used by the compiler when linking programs, but are not needed to run programs. You should definitely copy all `.so` files and the executable files in `usr/bin` and `sbin`.

3.6.2. Using Linker Options to Specify the Sysroot Location

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

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

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

2. When invoking `arm-none-linux-gnueabi-gcc` to link your executable, include the command-line options:

   ```
   -Wl,-rpath=sysroot/lib:sysroot/usr/lib \
   -Wl,--dynamic-linker=sysroot/lib/ld-linux.so.3
   ```

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

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

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

or

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

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

3.6.3. Specifying the Sysroot Location at Runtime

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

To do this, follow these steps:

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

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

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

   ```
   > sysroot/lib/ld-linux.so.3 \
     --library-path sysroot/lib:sysroot/usr/lib \
     /path/to/your-executable
   ```

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

3.7. Using GDB Server for Debugging

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

3.7.1. Running GDB Server

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

You must copy the sysroot to your target system as described in Section 3.6.1, “Installing the Sysroot”. You must also copy the executable you want to debug to your target system.

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

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

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

If you have installed the sysroot in an alternate directory, invoking `gdbserver` becomes more complicated. You must build your application using the link-time options to specify the location of the sysroot, as described in Section 3.6.2, “Using Linker Options to Specify the Sysroot Location”. You must also invoke `gdbserver` itself using the dynamic linker provided in the Sourcery CodeBench sysroot, as described in Section 3.6.3, “Specifying the Sysroot Location at Runtime”. In other words, the command to invoke `gdbserver` in this case would be similar to:

```
> sysroot/lib/ld-linux.so.3 \
  --library-path sysroot/lib:sysroot/usr/lib \
  sysroot/usr/lib/bin/gdbserver :10000 \
  program arg1 arg2 ...
```

3.7.2. Connecting to GDB Server from the Debugger

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

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

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

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

3.7.3. Setting the Sysroot in the Debugger

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

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

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

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

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

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

Refer to Section 3.6.1, “Installing the Sysroot” for more information about installing the sysroot on the target. Note that if you have installed a stripped copy of the provided libraries on the target, you should give GDB the location of an unstripped copy on the host.

3.8. GLIBC Backtrace Support

Sourcery CodeBench supports the `backtrace` function from GLIBC. Backtracing is supported regardless of optimization, with or without a frame pointer, and in both ARM and Thumb modes.

In order to support backtracing, Sourcery CodeBench enables generation of unwind tables by default when compiling. These tables are used for any stack traversal, including `backtrace`, C++ exception handling, and POSIX thread cancellation. Where none of these are required, you can reduce application size by compiling with `-fno-unwind-tables`.

Some stand-alone programs, including bootloaders and the Linux kernel, cannot be built with unwind tables. To accommodate these programs, Sourcery CodeBench suppresses unwind tables for C code if the `-ffreestanding` option is used. Unwind tables are also suppressed if the `-mabi` option is provided, as this option is not generally used in user-space programs. To override this behavior, specify `-funwind-tables` on the `arm-none-linux-gnueabi-gcc` command line.

3.9. Using VFP Floating Point

3.9.1. 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 CodeBench Lite for ARM GNU/Linux 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 in Sourcery CodeBench Lite. However, VFP hard-float libraries built with both ABIs are available to Sourcery CodeBench Standard and Professional Edition subscribers.

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.

3.9.2. NEON SIMD Code

Sourcery CodeBench 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 CodeBench 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.

3.9.3. Half-Precision Floating Point

Sourcery CodeBench for ARM GNU/Linux 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.

3.10. Fixed-Point Arithmetic

Sourcery CodeBench for ARM GNU/Linux 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 CodeBench for ARM GNU/Linux, 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.

3.11. 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 CodeBench and ARM RealView®.

Sourcery CodeBench implements the ABI as described in these documents, which are available from the ARM Information Center[2]:

- 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 CodeBench currently produces DWARF version 2, rather than DWARF version 3 as specified in AADWARF.

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

3.12. Object File Portability

It is possible to create object files using Sourcery CodeBench for ARM EABI that are link-compatible with the GNU C library provided with Sourcery CodeBench for ARM GNU/Linux as well as with the CodeSourcery C Library or Newlib C Library provided with ARM bare-metal toolchains. These object files are additionally link-compatible with other ARM C Library ABI-compliant static linking environments and toolchains.

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

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

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

Chapter 4
Using Sourcery CodeBench from the Command Line

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

4.1. Building an Application

This chapter explains how to build an application with Sourcery CodeBench Lite using the command line. As elsewhere in this manual, this section assumes that your target system is arm-none-linux-gnueabi, as indicated by the `arm-none-linux-gnueabi` 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-none-linux-gnueabi-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-none-linux-gnueabi-gcc` with `arm-none-linux-gnueabi-g++`.)

4.2. Running Applications on the Target System

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

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

```
> factorial
```

You should see:

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

```
factorial(7) = 5040
factorial(8) = 40320
factorial(9) = 362880
```

4.3. Running Applications from GDB

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

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

```
> arm-none-linux-gnueabi-gdb factorial
```

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

4.3.1. Connecting to an External GDB Server

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

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

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

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

4.4. Using the Compiler Cache

Note on Availability

This feature is currently available on Linux hosts only.

Compiling source code can be quite slow, and frequently recompiling that code can be extremely inefficient. Sourcery CodeBench Lite includes a tool named `arm-none-linux-gnueabi-cs` that solves this problem via *caching*.

The caching tool intercepts compiler invocations, generates a unique signature from the source files, command-line parameters, and other environmental information, and serves the object file and warning messages directly from the cache. If the object is not currently cached then the real compiler is called, and the cache updated.

The first time you build with caching enabled you can expect the build to take 10-30% *longer*. The second time you build it might be 80% *faster*. The memory and CPU usage savings may also mean it is possible to use higher levels of build parallelism (e.g. `make -j`) and gain even more performance.

`arm-none-linux-gnueabi-cs` is based on the well-known open-source tool *ccache*. Refer to `man cs` for more information.

4.4.1. Invoking `arm-none-linux-gnueabi-cs`

There are two ways you can run the caching tool with command-line builds.

- Explicitly invoke `arm-none-linux-gnueabi-cs`. For example, like this:

  ```
  > arm-none-linux-gnueabi-cs arm-none-linux-gnueabi-gcc -c hello.c
  ```

  or like this:

  ```
  > make CC="arm-none-linux-gnueabi-cs arm-none-linux-gnueabi-gcc"
  ```

- Add *`installdir`*`/bin/cache` to the head of your `PATH`, and run your normal compile command:

  ```
  > export PATH=installdir/bin/cache:installdir/bin:$PATH
  > arm-none-linux-gnueabi-gcc -c hello.c
  ```

Chapter 5
Next Steps with Sourcery CodeBench

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

5.1. Sourcery CodeBench Knowledge Base

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

5.2. Manuals for GNU Toolchain Components

Sourcery CodeBench 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 CodeBench 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 CodeBench Lite, the documentation can be found in the `share/doc/arm-arm-none-linux-gnueabi/` subdirectory of your installation directory.

In addition to the detailed reference manuals, Sourcery CodeBench 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/arm-arm-none-linux-gnueabi/man/man1
```

Then you can invoke `man` as:

```
> man ./arm-none-linux-gnueabi-gcc.1
```

Alternatively, if you use `man` regularly, you'll probably find it more convenient to add the directory containing the Sourcery CodeBench 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 CodeBench 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://sourcery.mentor.com/GNUToolchain/

Appendix A
Sourcery CodeBench Lite Release Notes

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

A.1. Changes in Sourcery CodeBench Lite for ARM GNU/Linux

This section documents Sourcery CodeBench Lite changes for each released revision.

A.1.1. Changes in Sourcery CodeBench Lite 2013.05-24

GCC version 4.7.3. Sourcery CodeBench Lite for ARM GNU/Linux is now based on GCC version 4.7.3. This update incorporates numerous bug fixes. For more information, see `http://gcc.gnu.org/gcc-4.7/changes.html`.

Installer warnings fixed. A bug that caused Gtk warnings relating to `libappmenu.so` when running the installer on 64-bit Ubuntu GNU/Linux hosts has been fixed.

New GLIBC macro `issignaling`. A new `<math.h>` macro named `issignaling` to check for a signaling Not a Number (sNaN) has been added to GLIBC. Please see the manual for further information.

A.1.2. Changes in Sourcery CodeBench Lite 2013.05-5

Pointer comparison bug fixed. A bug in GCC that caused it to incorrectly optimize away a pointer comparison has been fixed.

Atomic operations. A bug has been fixed that caused several built-in atomic functions (e.g. `__sync_val_compare_and_swap`) to operate incorrectly with `char` or `short` arguments when compiled for architecture versions earlier than ARMv7.

Incorrect optimization bug fix. A compiler bug has been fixed that caused incorrect code to be generated for some comparisons unless optimization was suppressed with `-fno-forward-propagate`.

Performance regression fixed. A bug that introduced unnecessary instructions to zero-extend unsigned `char` or `short` values has been fixed.

Linker raw binary input crash fix. A bug that caused the linker to crash when linking binary inputs (`--format=binary`) while using `--gc-sections` has been fixed.

Linker assertion failure fix. A linker bug has been fixed that caused an assertion failure when linking unoptimized code using thread-local storage.

`ldralt` assembly bug fix. A bug that caused the assembly of `ldralt` instructions to sometimes produce the error message `selected processor does not support ARM mode` has been fixed.

Binutils update. The binutils package has been updated to version 2.23.52.20130219 from the FSF trunk. This update includes numerous bug fixes.

Installing multiple targets in one directory. Due to changes in the installer, Sourcery CodeBench Lite for ARM GNU/Linux can no longer be installed into a directory already containing an existing Sourcery CodeBench installation for a different target. The installer detects this situation and asks you to select a different directory.

Fix for installer upgrade problems. Sourcery CodeBench Lite for ARM GNU/Linux can now be installed into a directory already containing a previous version.

EGLIBC version 2.17. Sourcery CodeBench Lite for ARM GNU/Linux now includes EGLIBC version 2.17 library which is based on GNU C Library version 2.17. For more information about changes, see `http://www.eglibc.org/news#eglibc_2_17`.

Linux kernel headers update. Linux kernel header files have been updated to version 3.8.2.

Improved source line stepping. GDB and `gdbserver` now implement range stepping, which improves the performance of single stepping over a source line by reducing the number of control messages from GDB.

GDB hang fix. A bug that caused GDB to sometimes hang when setting a breakpoint has been fixed.

A.1.3. Changes in Sourcery CodeBench Lite 2012.09-64

Code size optimization improvement. When compiling with `-Os` or `-O2` and higher GCC no longer enables the `-funroll-loops` option by default. This change reduces the size of optimized code produced by Sourcery CodeBench and brings its behavior back into conformance with other GCC distributions. For more information about the tradeoffs between optimizing for size versus speed please see the Sourcery CodeBench Knowledge Base [1].

Loop optimization bug fix. A compiler bug that caused some forms of loop to be mis-optimized when using the `-fpromote-loop-indices` option (enabled by default when optimizing for speed) has been fixed.

Wrong-code bug fix. A bug in GCC's scheduler has been fixed that sometimes caused incorrect code to be generated.

Install to empty directory failure fixed. A bug that prevented installation of Sourcery CodeBench Lite into an existing empty directory has been fixed.

Misleading GDB warning fixed. A bug has been fixed that formerly caused GDB to display a warning about being unable to load symbols for ELF libraries that are contained in the Linux kernel itself.

Updated system requirements. The host operating system requirements for Sourcery CodeBench Lite have been updated. The minimum versions of GNU/Linux now supported are Red Hat Enterprise Linux 5, SuSE Enterprise Linux 10, Fedora Core 6, Ubuntu 8.04, and Debian 5, or later versions of these distributions running on 32-bit or 64-bit Intel or AMD CPUs.

A.1.4. Changes in Sourcery CodeBench Lite 2012.09-45

No significant changes. There are no significant changes for ARM GNU/Linux in this release.

A.1.5. Changes in Sourcery CodeBench Lite 2012.09-19

GCC version 4.7.2. Sourcery CodeBench Lite for ARM GNU/Linux is now based on GCC version 4.7.2. For more information about changes from GCC version 4.6 that was included in previous releases, see `http://gcc.gnu.org/gcc-4.7/changes.html`.

Select correct multilib for `-mcpu=cortex-a15`. A GCC bug has been fixed that caused an incorrect multilib to be selected when compiling with `-mcpu=cortex-a15`.

[1] https://support.codesourcery.com/GNUToolchain/kbentry77

Incorrect warnings for naked functions fixed. A bug in the C++ compiler has been fixed that caused it to incorrectly warn about missing return statements in functions with the `naked` attribute, which can only include inline assembly statements.

Linker script symbols. The linker now supports a new `HIDDEN` keyword to define symbols with object scope. Refer to the linker manual for details.

Binutils version 2.23. Sourcery CodeBench Lite for ARM GNU/Linux is now based on binutils version 2.23.

EGLIBC version 2.16. Sourcery CodeBench Lite for ARM GNU/Linux now includes EGLIBC version 2.16 library which is based on GNU C Library version 2.16. For more information about changes, see `http://www.eglibc.org/news#eglibc_2_16`.

Linux kernel headers update. Linux kernel header files have been updated to version 3.5.4.

GDB update. The included version of GDB has been updated to 7.4.50.20120716. This update adds numerous bug fixes and features. Refer to `http://www.gnu.org/software/gdb/news` for more information.

Removal of the Sourcery CodeBench Debug Sprite. The Sourcery CodeBench Debug Sprite has been removed.

A.1.6. Changes in Sourcery CodeBench Lite 2012.03-57

New Sourcery CodeBench Lite branding. Sourcery G++ has been renamed to Sourcery CodeBench. This change affects the names of the default installation directory and installer-created shortcuts, but no internal pathnames or tool names within the installation directory have been changed.

Fix for internal compiler error. A bug that caused GCC to report an internal compiler error in `push_minipool_fix` has been fixed.

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

Nondeterministic code generation bug fix. A GCC bug has been fixed that caused nondeterministic code generation for some input files when optimizing.

Fix for compiler hang. A bug that caused GCC to become stuck in an infinite loop in the optimizer has been fixed.

Internal compiler error. A GCC bug has been fixed that caused an internal compiler error when sign extending the result of an array subscript expression with an index greater than 255.

GCC version 4.6. Sourcery CodeBench Lite for ARM GNU/Linux is now based on GCC version 4.6. For more information about changes from GCC version 4.5 that was included in previous releases, see `http://gcc.gnu.org/gcc-4.6/changes.html`.

Fix for internal compiler error. A GCC bug has been fixed that caused an internal compiler error when using pointer casts in C++0x `constexpr` initialization expressions.

ARM VFP9-S errata workaround. A compiler workaround for ARM Errata Notice GENC-010704 (760019: Canceled FDIV or FSQRT can be executed twice) has been implemented.

Fix for bit-field optimization bug. A compiler bug that caused incorrect code to be generated for programs using bit-fields has been fixed.

GCC version 4.6.3. Sourcery CodeBench Lite for ARM GNU/Linux is now based on GCC version 4.6.3. For more information about issues that have been fixed since version 4.6.1, see `http://gcc.gnu.org/gcc-4.6/changes.html`.

Compiler crash fixed. A GCC bug that occasionally caused an internal compiler error during register allocation has been fixed.

Map file name demangling bug fix. GCC now properly passes the `--demangle` and `--no-demangle` options to the linker to control map file output. The default behavior on all hosts is now to demangle C++ names.

GCC stack usage improvement. GCC now generates better code for stack allocation in some cases when compiling with `-fno-strict-aliasing`.

ARM miscompilation fix. A bug has been fixed that caused miscompilation of some expressions involving the minimum or maximum idiom, such as `(a > 0) ? a : 0`.

Register allocation bug fix. A bug in the register allocator that caused incorrect code generation has been fixed.

Linker COMDAT group code size optimization. The linker supports a new option `-Wl,--shared-comdat` that causes it to resolve COMDAT group symbols to definitions provided by shared libraries when possible, discarding any duplicate definitions of the group in relocatable files included in the link. This optimization can produce smaller C++ executables and shared libraries at the expense of additional runtime overhead.

Linker `--gc-sections` option bug fix. A bug has been fixed that caused the linker to incorrectly remove the `.debug_types` section when using the `--gc-sections` option.

Linker `--gc-sections` bug fix. A linker bug that incorrectly caused undefined references to be diagnosed when the `--gc-sections` option is used has been fixed.

Binutils version 2.21. Sourcery CodeBench Lite for ARM GNU/Linux is now based on binutils version 2.21.

Assembler crash. The assembler now warns when there is line information for the `*ABS*` section, rather than crash. This can occur when the `.offset` directive is used incorrectly.

Installer failure during upgrade. Some recent releases were affected by an installer bug on Windows hosts that caused installing a newer version of Sourcery CodeBench Lite into the same directory to fail with the error `Sourcery CodeBench Lite for ARM GNU/Linux upgrade failed`. This bug has now been fixed, but the affected releases cannot be upgraded. As a workaround, uninstall the older release before installing the new version.

Freescale i.MX53 QSB support. Sourcery CodeBench Lite now supports the Freescale i.MX53 QSB board.

RPC library functions. GLIBC has been changed so that programs may again be built to use its RPC library functions. The ability to build programs using these functions had previously been disabled.

`zic` bug fix. A bug has been fixed in GLIBC's `zic` command (included in the target sysroot) that caused `too many transitions` errors on some valid input files.

EGLIBC version 2.15. Sourcery CodeBench Lite for ARM GNU/Linux now includes EGLIBC version 2.15 library which is based on GNU C Library version 2.15. For more information about changes, see `http://www.eglibc.org/news#eglibc_2_15`.

Linux kernel headers update. Linux kernel header files have been updated to version 2.6.39.

Linux kernel headers update. Linux kernel header files have been updated to version 3.0.1.

Linux kernel headers update. Linux kernel header files have been updated to version 3.2.10.

Fix for crash in GDB `maint print arch`. A bug in the GDB command `maint print arch` that sometimes caused GDB to crash has been fixed.

C++ debugging bug fix. A GDB bug has been fixed that caused GDB to fail to find `enum` constants in base classes when debugging C++ code.

Fix for crash in GDB. A memory corruption bug in GDB has been fixed that under very rare circumstances made it crash or exhibit other unpredictable behavior. On GNU/Linux hosts, this bug caused crashes with an error message similar to: `*** glibc detected *** arm-none-linux-gnueabi-gdb: free(): invalid next size (normal): 0x09466198 ***` followed by a backtrace.

GDB interrupt handling bug fix. A bug in GDB has been fixed that caused it to sometimes fail to interrupt lengthy single-step operations (as by a **Ctrl**+**C** when using GDB from the command line).

Fix debugger remote target interruption. A bug in GDB's handling of requests to interrupt execution on a remote target has been fixed that caused it to stop the target but not emit a stopped MI record.

Fix GDB crash during connection to debug agent. A bug has been fixed that caused GDB to crash while connecting to any debug agent through standard IO where the debug agent had detected an early error and terminated the communication.

GDB internal error fix. A bug has been fixed that caused GDB to produce messages of the form: `warning: (Internal error: pc 0x1000a0 in read in psymtab, but not in symtab.)` when taking the addresses of symbols from objects added with the `add-symbol-file` command.

Improved disassembler performance in the debugger. GDB's disassembler has been improved to use more efficient memory access on remote targets.

Fix GDB crash in debugging Thumb assembly routines. A bug in GDB has been fixed that caused a crash when debugging Thumb assembly routines that switch stacks by writing the stack pointer in the function prologue.

Debug Sprite option defaults. The Sourcery CodeBench Debug Sprite now uses default option values specified in board configuration files. Options included in the device URL override the default values.

Changes to host operating system requirements. The minimum required Microsoft Windows OS needed to run Sourcery CodeBench Lite is now Windows XP (SP1).

A.1.7. Changes in Older Releases

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

Appendix B
Sourcery CodeBench Lite Licenses

Sourcery CodeBench 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 CodeBench Lite. You should read this appendix to understand your legal rights and obligations as a user of Sourcery CodeBench Lite.

The Mentor Graphics License is available in Section B.1, “Sourcery CodeBench Lite License Agreement”.

B.1. Sourcery CodeBench Lite License Agreement

Sourcery CodeBench Lite for ARM GNU/Linux is licensed under the Mentor Graphics **Embedded Software and Hardware License Agreement.** If you have a separate signed or shrinkwrap agreement (as applicable) with Mentor Graphics related to your use of Sourcery CodeBench Lite, your order is subject to the terms of that agreement. If you do not, the following terms apply, unless otherwise specifically agreed to in writing by an authorized representative of Mentor Graphics. The terms of this Getting Started Guide supplement, but do not replace or amend, the terms of your separate agreement with Mentor Graphics. Accordingly, to the extent the following terms and conditions conflict with such separate agreement, the terms and conditions of the separate agreement shall control.

The latest version of the License Agreement is available on-line at http://www.mentor.com/terms_conditions/embedded_software_license.

B.1.1. Embedded Software and Hardware License Agreement

IMPORTANT INFORMATION

USE OF ALL PRODUCTS IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE PRODUCTS. USE OF PRODUCTS INDICATES CUSTOMER'S COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.

EMBEDDED SOFTWARE AND HARDWARE LICENSE AGREEMENT ("Agreement")

This is a legal agreement concerning the use of Products (as defined in Section 1) between the company acquiring the Products ("Customer"), and the Mentor Graphics entity that issued the corresponding quotation or, if no quotation was issued, the applicable local Mentor Graphics entity ("Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by Customer and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If Customer does not agree to these terms and conditions, promptly return or, in the case of Products received electronically, certify destruction of Products and all accompanying items within five days after receipt of such Products and receive a full refund of any license fee paid.

1. **Definitions.**

    1.1. "Customer's Product" means Customer's end-user product identified by a unique SKU (including any Related SKUs) in an applicable Addenda that is developed, manufactured, branded and shipped solely by Customer or an authorized manufacturer or subcontractor on behalf of Customer to end-users or consumers;

1.2. "Developer" means a unique user, as identified by a unique user identification number, with access to Embedded Software at an authorized Development Location. A unique user is an individual who works directly with the embedded software in source code form, or creates, modifies or compiles software that ultimately links to the Embedded Software in Object Code form and is embedded into Customer's Product at the point of manufacture;

1.3. "Development Location" means the location where Products may be used as authorized in the applicable Addenda;

1.4. "Development Tools" means the software that may be used by Customer for building, editing, compiling, debugging or prototyping Customer's Product;

1.5. "Embedded Software" means Software that is embeddable;

1.6. "End-User" means Customer's customer;

1.7. "Executable Code" means a compiled program translated into a machine-readable format that can be loaded into memory and run by a certain processor;

1.8. "Hardware" means a physically tangible electro-mechanical system or sub-system and associated documentation;

1.9. "Linkable Object Code" or "Object Code" means linkable code resulting from the translation, processing, or compiling of Source Code by a computer into machine-readable format;

1.10. "Mentor Embedded Linux" or "MEL" means Mentor Graphics' tools, source code, and recipes for building Linux systems;

1.11. "Open Source Software" or "OSS" means software subject to an open source license which requires as a condition for redistribution of such software, including modifications thereto, that the: (i) redistribution be in source code form or be made available in source code form; (ii) redistributed software be licensed to allow the making of derivative works; or (iii) redistribution be at no charge;

1.12. "Processor" means the specific microprocessor to be used with Software and implemented in Customer's Product;

1.13. "Products" means Software, Term-Licensed Products and/or Hardware;

1.14. "Proprietary Components" means the components of the Products that are owned and/or licensed by Mentor Graphics and are not subject to an Open Source Software license, as more fully set forth herein;

1.15. "Redistributable Components" means those components that are intended to be incorporated or linked into Customer's Linkable Object Code developed with the Software, as more fully set forth herein;

1.16. "Related SKU" means two or more Customer Products identified by logically-related SKUs, where there is no difference or change in the electrical hardware or software content between such Customer Products;

1.17. "Software" means software programs, Embedded Software and/or Development Tools, including any updates, modifications, revisions, copies, documentation and design data that are licensed under this Agreement;

1.18. "Source Code" means software in a form in which the program logic is readily understandable by a human being;

1.19. "Sourcery CodeBench Software" means Mentor Graphics' Development Tool for C/C++ embedded application development;

1.20. "Sourcery VSIPL++" is Software providing C++ classes and functions for writing embedded signal processing applications designed to run on one or more processors;

1.21. "Stock Keeping Unit" or "SKU" is a unique number or code used to identify each distinct product, item or service available for purchase;

1.22. "Subsidiary" means any corporation more than 50% owned by Customer, excluding Mentor Graphics competitors. Customer agrees to fulfill the obligations of such Subsidiary in the event of default. To the extent Mentor Graphics authorizes any Subsidiary's use of Products under this Agreement, Customer agrees to ensure such Subsidiary's compliance with the terms of this Agreement and will be liable for any breach by a Subsidiary; and

1.23. "Term-Licensed Products" means Products licensed to Customer for a limited time period ("Term").

2. **Orders, Fees and Payment.**

2.1. To the extent Customer (or if agreed by Mentor Graphics, Customer's appointed third party buying agent) places and Mentor Graphics accepts purchase orders pursuant to this Agreement ("Order(s)"), each Order will constitute a contract between Customer and Mentor Graphics, which shall be governed solely and exclusively by the terms and conditions of this Agreement and any applicable Addenda, whether or not these documents are referenced on the Order. Any additional or conflicting terms and conditions appearing on an Order will not be effective unless agreed in writing by an authorized representative of Customer and Mentor Graphics.

2.2. Amounts invoiced will be paid, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. All invoices will be sent electronically to Customer on the date stated on the invoice unless otherwise specified in an Addendum. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Prices do not include freight, insurance, customs duties, taxes or other similar charges, which Mentor Graphics will state separately in the applicable invoice(s). Unless timely provided with a valid certificate of exemption or other evidence that items are not taxable, Mentor Graphics will invoice Customer for all applicable taxes including, but not limited to, VAT, GST, sales tax, consumption tax and service tax. Customer will make all payments free and clear of, and without reduction for, any withholding or other taxes; any such taxes imposed on payments by Customer hereunder will be Customer's sole responsibility. If Customer appoints a third party to place purchase orders and/or make payments on Customer's behalf, Customer shall be liable for payment under Orders placed by such third party in the event of default.

2.3. All Products are delivered FCA factory (Incoterms 2010), freight prepaid and invoiced to Customer, except Software delivered electronically, which shall be deemed delivered when made available to Customer for download. Mentor Graphics' delivery of Software by electronic means is subject to Customer's provision of both a primary and an alternate e-mail address.

3. **Grant of License.**

3.1. The Products installed, downloaded, or otherwise acquired by Customer under this Agreement constitute or contain copyrighted, trade secret, proprietary and confidential information of Mentor Graphics or its licensors, who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to Customer, subject to payment of applicable license fees, a nontransferable, nonexclusive license to use Software as described in the applicable Addenda. The limited licenses granted under the applicable Addenda shall continue until the expiration date of Term-Licensed Products or termination in accordance with Section 12 below, whichever occurs first. Mentor Graphics does NOT grant Customer any right to (a) sublicense or (b) use Software beyond the scope of this Section without first signing a separate agreement or Addenda with Mentor Graphics for such purpose.

3.2. License Type. The license type shall be identified in the applicable Addenda.

3.2.1. Development License: During the Term, if any, Customer may modify, compile, assemble and convert the applicable Embedded Software Source Code into Linkable Object Code and/or Executable Code form by the number of Developers specified, for the Processor(s), Customer's Product(s) and at the Development Location(s) identified in the applicable Addenda.

3.2.2. End-User Product License: During the Term, if any, and unless otherwise specified in the applicable Addenda, Customer may incorporate or embed an Executable Code version of the Embedded Software into the specified number of copies of Customer's Product(s), using the Processor Unit(s), and at the Development Location(s) identified in the applicable Addenda. Customer may manufacture, brand and distribute such Customer's Product(s) worldwide to its End-Users.

3.2.3. Internal Tool License: During the Term, if any, Customer may use the Development Tools solely: (a) for internal business purposes and (b) on the specified number of computer work stations and sites. Development Tools are licensed on a per-seat or floating basis, as specified in the applicable Addenda, and shall not be distributed to others or delivered in Customer's Product(s) unless specifically authorized in an applicable Addenda.

3.2.4. Sourcery CodeBench Professional Edition License: During the Term specified in the applicable Addenda, Customer may (a) install and use the Proprietary Components of the Software (i) if the license is a node-locked license, by a single user who uses the Software on up to two machines provided that only one copy of the Software is in use at any one time, or (ii) if the license is a floating license, by the authorized number of concurrent users on one or more machines provided that only the authorized number of copies of the Software are in use at any one time, and (b) distribute the Redistributable Components of the Software in Executable Code form only and only as part of Customer's Object Code developed with the Software that provides substantially different functionality than the Redistributable Component(s) alone.

3.2.5. Sourcery CodeBench Standard Edition License: During the Term specified in the applicable Addenda, Customer may (a) install and use the Proprietary Components of the Software by a single user who uses the Software on up to two machines provided that only one copy of the Software is in use at any one time, and (b) distribute the Redistributable Component(s) of the Software in Executable Code form only and only as part of Customer's Object Code developed with the Software that provides substantially different functionality than the Redistributable Component(s) alone.

3.2.6. Sourcery CodeBench Personal Edition License: During the Term specified in the applicable Addenda, Customer may (a) install and use the Proprietary Components

of the Software by a single user who uses the Software on one machine, and (b) distribute the Redistributable Component(s) of the Software in Executable Code form only and only as part of Customer Object Code developed with the Software that provides substantially different functionality than the Redistributable Component(s) alone.

3.2.7. Sourcery CodeBench Academic Edition License: During the Term specified in the applicable Addenda, Customer may (a) install and use the Proprietary Components of the Software for non-commercial, academic purposes only by a single user who uses the Software on one machine, and (b) distribute the Redistributable Component(s) of the Software in Executable Code form only and only as part of Customer Object Code developed with the Software that provides substantially different functionality than the Redistributable Component(s) alone.

3.3. Mentor Graphics may from time to time, in its sole discretion, lend Products to Customer. For each loan, Mentor Graphics will identify in writing the quantity and description of Software loaned, the authorized location and the Term of the loan. Mentor Graphics will grant to Customer a temporary license to use the loaned Software solely for Customer's internal evaluation in a non-production environment. Customer shall return to Mentor Graphics or delete and destroy loaned Software on or before the expiration of the loan Term. Customer will sign a certification of such deletion or destruction if requested by Mentor Graphics.

4. **Beta Code.**

4.1. Portions or all of certain Products may contain code for experimental testing and evaluation ("Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to Customer a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. This grant and Customer's use of the Beta Code shall not be construed as marketing or offering to sell a license to the Beta Code, which Mentor Graphics may choose not to release commercially in any form.

4.2. If Mentor Graphics authorizes Customer to use the Beta Code, Customer agrees to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. Customer will contact Mentor Graphics periodically during Customer's use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of Customer's evaluation and testing, Customer will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.

4.3. Customer agrees to maintain Beta Code in confidence and shall restrict access to the Beta Code, including the methods and concepts utilized therein, solely to those employees and Customer location(s) authorized by Mentor Graphics to perform beta testing. Customer agrees that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on Customer's feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this Subsection 4.3 shall survive termination of this Agreement.

5. **Restrictions on Use.**

5.1. Customer may copy Software only as reasonably necessary to support the authorized use, including archival and backup purposes. Each copy must include all notices and legends

embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. Except where embedded in Executable Code form in Customer's Product, Customer shall maintain a record of the number and location of all copies of Software, including copies merged with other software and products, and shall make those records available to Mentor Graphics upon request. Customer shall not make Products available in any form to any person other than Customer's employees, authorized manufacturers or authorized contractors, excluding Mentor Graphics competitors, whose job performance requires access and who are under obligations of confidentiality. Customer shall take appropriate action to protect the confidentiality of Products and ensure that any person permitted access does not disclose or use Products except as permitted by this Agreement. Customer shall give Mentor Graphics immediate written notice of any unauthorized disclosure or use of the Products as soon as Customer learns or becomes aware of such unauthorized disclosure or use.

5.2. Customer acknowledges that the Products provided hereunder may contain Source Code which is proprietary and its confidentiality is of the highest importance and value to Mentor Graphics. Customer acknowledges that Mentor Graphics may be seriously harmed if such Source Code is disclosed in violation of this Agreement. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive any Source Code from Products that are not provided in Source Code form. Except as embedded in Executable Code in Customer's Product and distributed in the ordinary course of business, in no event shall Customer provide Products to Mentor Graphics competitors. Log files, data files, rule files and script files generated by or for the Software (collectively "Files") constitute and/or include confidential information of Mentor Graphics. Customer may share Files with third parties, excluding Mentor Graphics competitors, provided that the confidentiality of such Files is protected by written agreement at least as well as Customer protects other information of a similar nature or importance, but in any case with at least reasonable care. Under no circumstances shall Customer use Products or allow their use for the purpose of developing, enhancing or marketing any product that is in any way competitive with Products, or disclose to any third party the results of, or information pertaining to, any benchmark.

5.3. Customer may not assign this Agreement or the rights and duties under it, or relocate, sublicense or otherwise transfer the Products, whether by operation of law or otherwise ("Attempted Transfer"), without Mentor Graphics' prior written consent, which shall not be unreasonably withheld, and payment of Mentor Graphics' then-current applicable relocation and/or transfer fees. Any Attempted Transfer without Mentor Graphics' prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics' option, result in the immediate termination of the Agreement and/or the licenses granted under this Agreement. The terms of this Agreement, including without limitation the licensing and assignment provisions, shall be binding upon Customer's permitted successors in interest and assigns.

5.4. Notwithstanding any provision in an OSS license agreement applicable to a component of the Sourcery CodeBench Software that permits the redistribution of such component to a third party in Source Code or binary form, Customer may not use any Mentor Graphics trademark, whether registered or unregistered, in connection with such distribution, and may not recompile the Open Source Software components with the --with-pkgversion or --with-bugurl configuration options that embed Mentor Graphics' trademarks in the resulting binary.

5.5. The provisions of this Section 5 shall survive the termination of this Agreement.

6. **Support Services.**

   6.1. Except as described in Sections 6.2, 6.3 and 6.4 below, and unless otherwise specified in any applicable Addenda to this Agreement, to the extent Customer purchases support services, Mentor Graphics will provide Customer updates and technical support for the number of Developers at the Development Location(s) for which support is purchased in accordance with Mentor Graphics' then-current End-User Software Support Terms located at http://supportnet.mentor.com/about/legal/.

   6.2. To the extent Customer purchases support services for Sourcery CodeBench Software, support will be provided solely in accordance with the provisions of this Section 6.2. Mentor Graphics shall provide updates and technical support to Customer as described herein only on the condition that Customer uses the Executable Code form of the Sourcery CodeBench Software for internal use only and/or distributes the Redistributable Components in Executable Code form only (except as provided in a separate redistribution agreement with Mentor Graphics or as required by the applicable Open Source license). Any other distribution by Customer of the Sourcery CodeBench Software (or any component thereof) in any form, including distribution permitted by the applicable Open Source license, shall automatically terminate any remaining support term. Subject to the foregoing and the payment of support fees, Mentor Graphics will provide Customer updates and technical support for the number of Developers at the Development Location(s) for which support is purchased in accordance with Mentor Graphics' then-current Sourcery CodeBench Software Support Terms located at http://www.mentor.com/codebench-support-legal.

   6.3. To the extent Customer purchases support services for Sourcery VSIPL++, Mentor Graphics will provide Customer updates and technical support for the number of Developers at the Development Location(s) for which support is purchased solely in accordance with Mentor Graphics' then-current Sourcery VSIPL++ Support Terms located at. http://www.mentor.com/vsipl-support-legal.

   6.4. To the extent Customer purchases support services for Mentor Embedded Linux, Mentor Graphics will provide Customer updates and technical support for the number of Developers at the Development Location(s) for which support is purchased solely in accordance with Mentor Graphics' then-current Mentor Embedded Linux Support Terms located at http://www.mentor.com/mel-support-legal.

7. **Third Party and Open Source Software.** Products may contain Open Source Software or code distributed under a proprietary third party license agreement. Please see applicable Products documentation, including but not limited to license notice files, header files or source code for further details. Please see Section B.2.2, “Components” for additional rights and obligations governing your use and distribution of Open Source Software. Customer agrees that it shall not subject any Product provided by Mentor Graphics under this Agreement to any Open Source Software license that does not otherwise apply to such Product. In the event of conflict between the terms of this Agreement, any Addenda and an applicable OSS or proprietary third party agreement, the OSS or proprietary third party agreement will control solely with respect to the OSS or proprietary third party software component. The provisions of this Section 7 shall survive the termination of this Agreement.

8. **Limited Warranty.**

   8.1. Mentor Graphics warrants that during the warranty period its standard, generally supported Products, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual and/or specification. Mentor Graphics does not warrant that Products will meet Customer's requirements or that operation of Products will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after

delivery or upon installation, whichever first occurs. Customer must notify Mentor Graphics in writing of any nonconformity within the warranty period. For the avoidance of doubt, this warranty applies only to the initial shipment of Products under an Order and does not renew or reset, for example, with the delivery of (a) Software updates or (b) authorization codes. This warranty shall not be valid if Products have been subject to misuse, unauthorized modification or improper installation. MENTOR GRAPHICS' ENTIRE LIABILITY AND CUSTOMER'S EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF THE PRODUCTS TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF THE PRODUCTS THAT DO NOT MEET THIS LIMITED WARRANTY, PROVIDED CUSTOMER HAS OTHERWISE COMPLIED WITH THIS AGREEMENT. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; OR (B) PRODUCTS PROVIDED AT NO CHARGE, WHICH ARE PROVIDED "AS IS" UNLESS OTHERWISE AGREED IN WRITING.

8.2. THE WARRANTIES SET FORTH IN THIS SECTION 8 ARE EXCLUSIVE TO CUSTOMER AND DO NOT APPLY TO ANY END-USER. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, WITH RESPECT TO PRODUCTS OR OTHER MATERIAL PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

9. **LIMITATION OF LIABILITY.** EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, AND EXCEPT FOR EITHER PARTY'S BREACH OF ITS CONFIDENTIALITY OBLIGATIONS, CUSTOMER'S BREACH OF LICENSING TERMS OR CUSTOMER'S OBLIGATIONS UNDER SECTION 10, IN NO EVENT SHALL: (A) EITHER PARTY OR ITS RESPECTIVE LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF SUCH PARTY OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES; AND (B) EITHER PARTY OR ITS RESPECTIVE LICENSORS' LIABILITY UNDER THIS AGREEMENT, INCLUDING, FOR THE AVOIDANCE OF DOUBT, LIABILITY FOR ATTORNEYS' FEES OR COSTS, EXCEED THE GREATER OF THE FEES PAID OR OWING TO MENTOR GRAPHICS FOR THE PRODUCT OR SERVICE GIVING RISE TO THE CLAIM OR $500,000 (FIVE HUNDRED THOUSAND U.S. DOLLARS). NOTWITHSTANDING THE FOREGOING, IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 9 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

10. **Hazardous Applications.**

10.1. Customer agrees that Mentor Graphics has no control over Customer's testing or the specific applications and use that Customer will make of Products. Mentor Graphics Products are not specifically designed for use in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support systems, medical devices or other applications in which the failure of Mentor Graphics Products could lead to death, personal injury, or severe physical or environmental damage ("Hazardous Applications").

10.2. CUSTOMER ACKNOWLEDGES IT IS SOLELY RESPONSIBLE FOR TESTING PRODUCTS USED IN HAZARDOUS APPLICATIONS AND SHALL BE SOLELY

LIABLE FOR ANY DAMAGES RESULTING FROM SUCH USE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS SHALL BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF PRODUCTS IN ANY HAZARDOUS APPLICATIONS.

10.3. CUSTOMER AGREES TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE OR LIABILITY, INCLUDING REASONABLE ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH THE USE OF PRODUCTS AS DESCRIBED IN SECTION 10.1.

10.4. THE PROVISIONS OF THIS SECTION 10 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

11. **Infringement.**

11.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against Customer in the United States, Canada, Japan, or member state of the European Union which alleges that any standard, generally supported Product acquired by Customer hereunder infringes a patent or copyright or misappropriates a trade secret in such jurisdiction. Mentor Graphics will pay any costs and damages finally awarded against Customer that are attributable to the action. Customer understands and agrees that as conditions to Mentor Graphics' obligations under this section Customer must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to settle or defend the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

11.2. If a claim is made under Subsection 11.1 Mentor Graphics may, at its option and expense, and in addition to its obligations under Section 11.1, either (a) replace or modify the Product so that it becomes noninfringing; or (b) procure for Customer the right to continue using the Product. If Mentor Graphics determines that neither of those alternatives is financially practical or otherwise reasonably available, Mentor Graphics may require the return of the Product and refund to Customer any purchase price or license fee(s) paid.

11.3. Mentor Graphics has no liability to Customer if the claim is based upon: (a) the combination of the Product with any product not furnished by Mentor Graphics, where the Product itself is not infringing; (b) the modification of the Product other than by Mentor Graphics or as directed by Mentor Graphics, where the unmodified Product would not infringe; (c) the use of the infringing Product when Mentor Graphics has provided Customer with a current unaltered release of a non-infringing Product of substantially similar functionality in accordance with Subsection 11.2(a); (d) the use of the Product as part of an infringing process; (e) a product that Customer makes, uses, or sells, where the Product itself is not infringing; (f) any Product provided at no charge; (g) any software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; (h) Open Source Software, except to the extent that the infringement is directly caused by Mentor Graphics' modifications to such Open Source Software; or (i) infringement by Customer that is deemed willful. In the case of (i), Customer shall reimburse Mentor Graphics for its reasonable attorneys' fees and other costs related to the action.

11.4. THIS SECTION 11 IS SUBJECT TO SECTION 9 ABOVE AND STATES: (A) THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS AND (B) CUSTOMER'S SOLE AND EXCLUSIVE REMEDY, WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY PRODUCT PROVIDED UNDER THIS AGREEMENT.

12. **Termination and Effect of Termination.** If a Software license was provided for limited term use, such license will automatically terminate at the end of the authorized Term.

    12.1. **Termination for Breach.** This Agreement shall remain in effect until terminated in accordance with its terms. Mentor Graphics may terminate this Agreement and/or any licenses granted under this Agreement, and Customer will immediately discontinue use and distribution of Products, if Customer (a) commits any material breach of any provision of this Agreement and fails to cure such breach upon 30-days prior written notice; or (b) becomes insolvent, files a bankruptcy petition, institutes proceedings for liquidation or winding up or enters into an agreement to assign its assets for the benefit of creditors. Termination of this Agreement or any license granted hereunder will not affect Customer's obligation to pay for Products shipped or licenses granted prior to the termination, which amounts shall be payable immediately upon the date of termination. For the avoidance of doubt, nothing in this Section 12 shall be construed to prevent Mentor Graphics from seeking immediate injunctive relief in the event of any threatened or actual breach of Customer's obligations hereunder.

    12.2. **Effect of Termination.** Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination or expiration of the Term, Customer will discontinue use and/or distribution of Products, and shall return Hardware and either return to Mentor Graphics or destroy Software in Customer's possession, including all copies and documentation, and certify in writing to Mentor Graphics within ten business days of the termination date that Customer no longer possesses any of the affected Products or copies of Software in any form, except to the extent an Open Source Software license conflicts with this Section 12.2 and permits Customer's continued use of any Open Source Software portion or component of a Product. Upon termination for Customer's breach, an End-User may continue its use and/or distribution of Customer's Product so long as: (a) the End-User was licensed according to the terms of this Agreement, if applicable to such End-User, and (b) such End-User is not in breach of its agreement, if applicable, nor a party to Customer's breach.

13. **Export.** The Products provided hereunder are subject to regulation by local laws and United States government agencies, which prohibit export or diversion of certain products, information about the products, and direct or indirect products thereof, to certain countries and certain persons. Customer agrees that it will not export Products in any manner without first obtaining all necessary approval from appropriate local and United States government agencies. Customer acknowledges that the regulation of product export is in continuous modification by local governments and/or the United States Congress and administrative agencies. Customer agrees to complete all documents and to meet all requirements arising out of such modifications.

14. **U.S. Government License Rights.** Software was developed entirely at private expense. All Software is commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to US FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. Government or a U.S. Government subcontractor is subject solely to the terms and conditions set forth in this Agreement, except for provisions which are contrary to applicable mandatory federal laws.

15. **Third Party Beneficiary.** For any Products licensed under this Agreement and provided by Customer to End-Users, Mentor Graphics or the applicable licensor is a third party beneficiary of the agreement between Customer and End-User. Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, and other licensors may be third party beneficiaries of this Agreement with the right to enforce the obligations set forth herein.

16. **Review of License Usage.** Customer will monitor the access to and use of Software. With prior written notice, during Customer's normal business hours, and no more frequently than

once per calendar year, Mentor Graphics may engage an internationally recognized accounting firm to review Customer's software monitoring system, records, accounts and sublicensing documents deemed relevant by the internationally recognized accounting firm to confirm Customer's compliance with the terms of this Agreement or U.S. or other local export laws. Such review may include FlexNet (or successor product) report log files that Customer shall capture and provide at Mentor Graphics' request. Customer shall make records available in electronic format and shall fully cooperate with data gathering to support the license review. Mentor Graphics shall bear the expense of any such review unless a material non-compliance is revealed. Mentor Graphics shall treat as confidential information all Customer information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement. Such license review shall be at Mentor Graphics' expense unless it reveals a material underpayment of fees of five percent or more, in which case Customer shall reimburse Mentor Graphics for the costs of such license review. Customer shall promptly pay any such fees. If the license review reveals that Customer has made an overpayment, Mentor Graphics has the option to either provide the Customer with a refund or credit the amount overpaid to Customer's next payment. The provisions of this Section 16 shall survive the termination of this Agreement.

17. **Controlling Law, Jurisdiction and Dispute Resolution.** This Agreement shall be governed by and construed under the laws of the State of California, USA, excluding choice of law rules. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of the state and federal courts of California, USA. Nothing in this section shall restrict Mentor Graphics' right to bring an action (including for example a motion for injunctive relief) against Customer or its Subsidiary in the jurisdiction where Customer's or its Subsidiary's place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

18. **Severability.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.

19. **Miscellaneous.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements, including but not limited to any purchase order terms and conditions. This Agreement may only be modified in writing, signed by an authorized representative of each party. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.

Rev. 120305, Part No. 252061

B.2. Licenses for Sourcery CodeBench Lite Components

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

B.2.1. Mentor Graphics Proprietary Components

Components of the Software that are owned and/or licensed by Mentor Graphics and are not subject to a “free software” or “open source” license, such as the GNU Public License. The Mentor Graphics Proprietary Components of the Software include, without limitation, the Sourcery CodeBench Installer, any Sourcery CodeBench Eclipse plug-ins, the CodeSourcery C Library (CSLIBC), and any Sourcery CodeBench Debug Sprite.

B.2.2. 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 CodeBench Lite. Sourcery CodeBench 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 CodeBench Lite.

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

Important

Although some of the licenses that apply to Sourcery CodeBench 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 CodeBench Lite. You can develop proprietary applications and libraries with Sourcery CodeBench Lite.

Sourcery CodeBench Lite may include some third party example programs and libraries in the `share/sourceryg++-arm-none-linux-gnueabi-examples` subdirectory. These examples are not covered by the Sourcery CodeBench 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.3. Attribution

This version of Sourcery CodeBench 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.
*/
```