Sourcery CodeBench Lite

C6000 uClinux

Sourcery CodeBench Lite 4.5-124

Getting Started

Sourcery CodeBench Lite: C6000 uClinux: Sourcery CodeBench Lite 4.5-124: Getting Started

CodeSourcery, Inc.

Abstract

This guide explains how to install and build applications with Sourcery CodeBench Lite, Code-Sourcery'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

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 C6000 uClinux”	This chapter contains information about using Sourcery CodeBench Lite that is specific to C6000 uClinux 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, “Sourcery CodeBench Debug Sprite”	This chapter describes the use of the Sourcery CodeBench Debug Sprite for remote debugging. The Sprite is provided for debugging of the uClinux kernel on the target board. This chapter includes information about the debugging devices and boards supported by the Sprite for C6000 uClinux.
Chapter 6, “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 C6000 uClinux. 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 C6000 uClinux is intended for developers working on embedded uClinux applications. It may also be used for uClinux kernel development and debugging, or to build a uClinux distribution.

Follow the steps given in this chapter to install Sourcery 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 C6000 uClinux”.

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. You must identify the multilib appropriate for your target in order to find the correct `gdbserver` executable to use for debugging your applications, as described in Section 3.3, “GDB Server”.

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.

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

Debug your program on the target using GDB server. You can debug a program on a remote C6000 uClinux target using GDB server. Copy your program to the target system. Follow the instructions in Section 3.3, “GDB Server” to install and run `gdbserver` on your target system. Then, you can connect to GDB server from the debugger running on your host system. Refer to Section 4.3,

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

“Running Applications from GDB” for instructions on connecting to the target from command-line GDB.

Chapter 2
Installation and Configuration

This chapter explains how to install Sourcery 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 `c6x-uclinux`.

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

Sourcery 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

In order to install and use Sourcery CodeBench Lite, you must have at least 512MB of available memory.

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 C6000 uClinux” 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

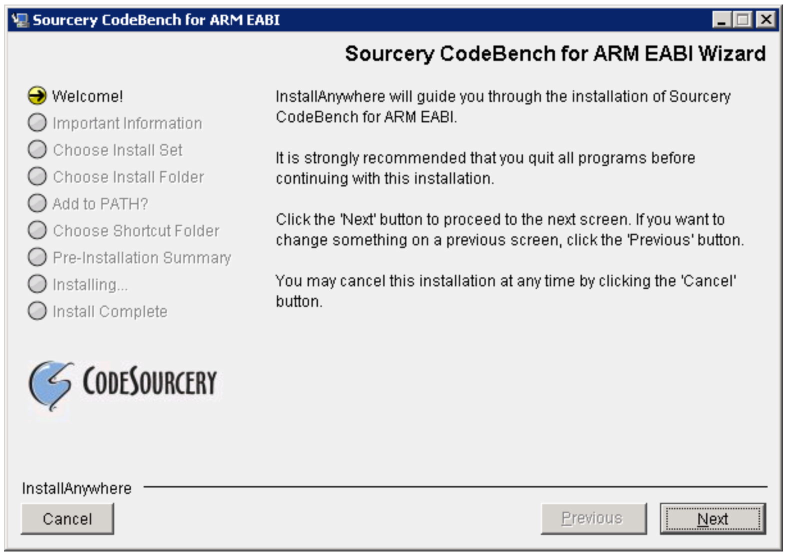
The method used to install Sourcery CodeBench Lite depends on your host system and the kind of installation package you have downloaded.

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



Sourcery CodeBench for ARM EABI Wizard

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

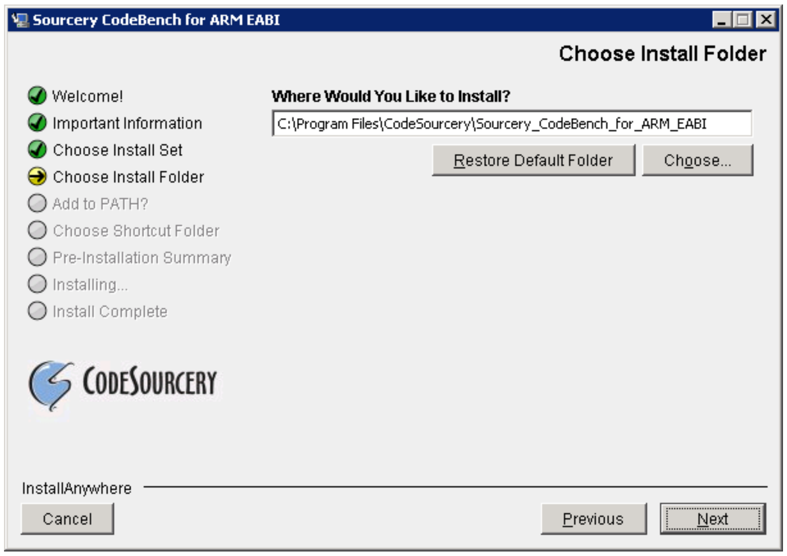
It is strongly recommended that you quit all programs before continuing with this installation.

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

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

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

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

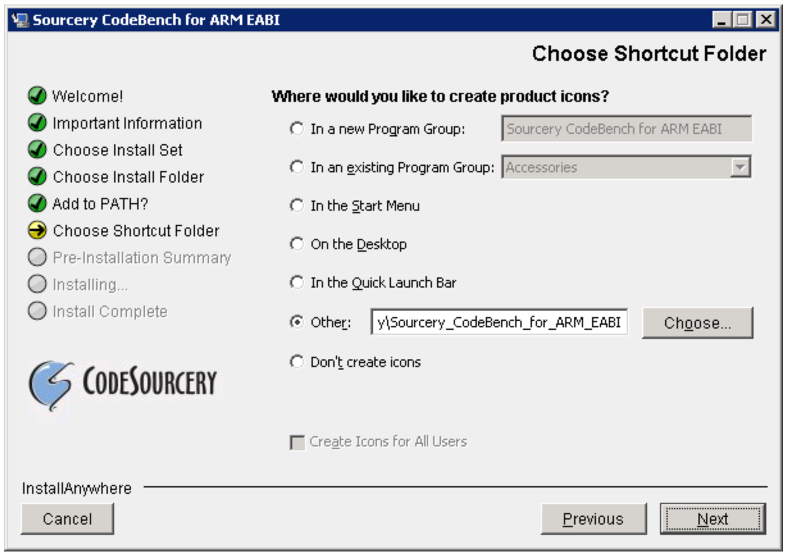


Choose Install Folder

Where Would You Like to Install?

C:\Program Files\CodeSourcery\Sourcery_CodeBench_for_ARM_EABI

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



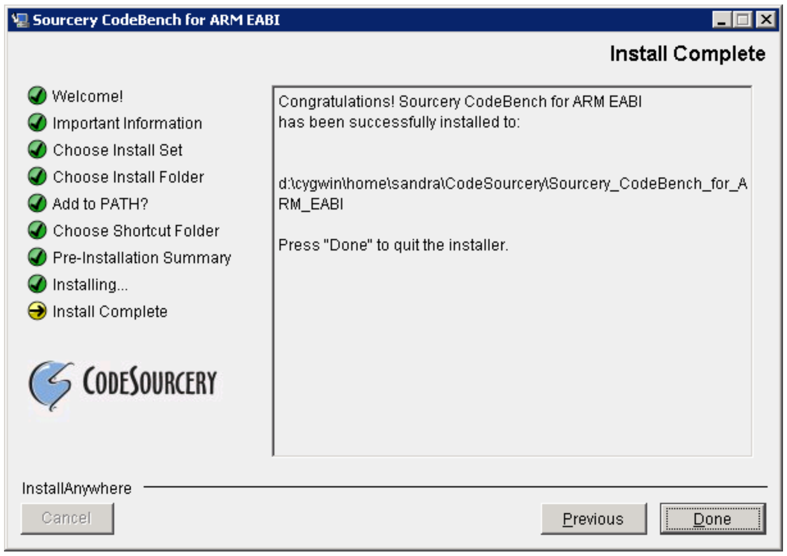
Where would you like to create product icons?

- In a new Program Group: Sourcery CodeBench for ARM EABI
- In an existing Program Group: Accessories
- In the Start Menu
- On the Desktop
- In the Quick Launch Bar
- [✓ Other: y\Sourcery_CodeBench_for_ARM_EABI] Choose...
- Don't create icons

Create Icons for All Users

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



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

d:\cygwin\home\sandra\CodeSourcery\Sourcery_CodeBench_for_ARM_EABI

Press "Done" to quit the installer.

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

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

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

2.4.2. Using the 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++-4.5`.

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 C6000 uClinux on your system, it is not necessary to uninstall it before using the installer to unpack a new version in the same location. The installer detects that it is performing an update in that case.

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

Note that the names of the Sourcery CodeBench commands for the C6000 uClinux target all begin with `c6x-uclinux`. 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

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

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:

```
> c6x-uclinux-g++ -v
```

Verify that the last line of the output contains: `Sourcery CodeBench Lite 4.5-124`.

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

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

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/sourceryg++-c6x-uclinux/man`.

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

```
> c6x-uclinux-g++ -v
```

Verify that the last line of the output contains: `Sourcery CodeBench Lite 4.5-124`.

2.7. 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 `c6x-uclinux` directory located in the install directory will be removed entirely by the uninstaller.

2.7.1. 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.

2.7.2. 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.

2.7.3. 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 C6000 uClinux

This chapter contains information about features of Sourcery CodeBench Lite that are specific to C6000 uClinux 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 C6000 uClinux, and tells you where you may find further information about these features.

Component	Version	Notes
GNU programming tools		
GNU Compiler Collection	4.5.1	Separate manual included.
GNU Binary Utilities	2.20.51	Includes assembler, linker, and other utilities. Separate manuals included.
Debugging support and simulators		
GNU Debugger	7.2.50	Separate manual included.
GDB Server	N/A	Included with GDB. See Section 3.3, “GDB Server”.
Target libraries		
uClibc C Library	0.9.32-20110309	
Linux Kernel Headers	2.6.38	
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 C6000 uClinux includes the following library configuration.

C64x+ - Little-Endian	
Command-line option(s):	default
Sysroot subdirectory:	`./`

C64x+ - Big-Endian	
Command-line option(s):	`-mbig-endian`
Sysroot subdirectory:	`be/`

C674x - Little-Endian	
Command-line option(s):	`-march=c674x`
Sysroot subdirectory:	`c674x/`

C674x - Big-Endian	
Command-line option(s):	`-mbig-endian -march=c674x`
Sysroot subdirectory:	`be/c674x/`

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. You can find the sysroot directories provided with Sourcery CodeBench in the `c6x-uclinux/libc` directory of your installation.

3.3. GDB Server

Sourcery CodeBench Lite contains a `gdbserver` for running on the target. The server executable is located in the *`sysroot`*`/usr/bin` directory of your installation, where *`sysroot`* is the pathname to the sysroot, as documented in Section 3.2, “Library Configurations”. You need to copy the appropriate `gdbserver` executable to your target system and then invoke it as

```
# gdbserver :port program
```

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

```
> c6x-uclinux-gdb program.gdb
```

Then connect to the target system:

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

At this point you are able to debug as usual.

Chapter 4
Using Sourcery 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 c6x-uclinux, as indicated by the `c6x-uclinux` 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:

```
> c6x-uclinux-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 `c6x-uclinux-gcc` with `c6x-uclinux-g++`.)

4.2. Running Applications on the Target System

To run your program on a uClinux target system, use the command:

```
> factorial
```

You should see:

```
factorial(0) = 1
factorial(1) = 1
factorial(2) = 2
factorial(3) = 6
factorial(4) = 24
factorial(5) = 120
factorial(6) = 720
factorial(7) = 5040
factorial(8) = 40320
factorial(9) = 362880
```

4.3. Running Applications from GDB

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

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

```
> c6x-uclinux-gdb factorial.gdb
```

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

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

4.3.1. Connecting to the Sourcery CodeBench Debug Sprite

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

You can start the Sprite directly from within GDB:

```
(gdb) target remote | c6x-uclinux-sprite arguments
```

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

4.3.2. 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 C6000 uClinux target. Follow the instructions in Chapter 3, “Sourcery CodeBench Lite for C6000 uClinux” to install and run `gdbserver` on your target system.

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

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

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

Chapter 5
Sourcery CodeBench Debug Sprite

This chapter describes the use of the Sourcery CodeBench Debug Sprite for remote debugging. The Sprite is provided for debugging of the uClinux kernel on the target board. This chapter includes information about the debugging devices and boards supported by the Sprite for C6000 uClinux.

Sourcery CodeBench Lite contains the Sourcery CodeBench Debug Sprite for C6000 uClinux. This Sprite is provided to allow debugging of programs running on a bare board. You can use the Sprite to debug a program when there is no operating system on the board, or for debugging the operating system itself. If the board is running an operating system, and you wish to debug a program running on that OS, you should use the facilities provided by the OS itself (for instance, using `gdbserver`).

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

Note for uClinux users

The Debug Sprite provided with Sourcery CodeBench Lite allows remote debugging of the uClinux kernel running on the target. For remote debugging of application programs, you should use `gdbserver` instead. See Chapter 3, “Sourcery CodeBench Lite for C6000 uClinux” for details about how to install and run `gdbserver` on the target.

Important

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

5.1. Probing for Debug Devices

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

```
> c6x-uclinux-sprite -i
```

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

```
CodeSourcery C6X Debug Sprite (Sourcery CodeBench Lite 4.5-124)
xds: [emulation=<name>&device=<name>] XDS device
  xds:// - XDS Device
```

This shows that Texas Instruments XDS devices are supported. It is not possible to autodetect these.

5.2. Invoking Sourcery CodeBench Debug Sprite

The Debug Sprite is invoked as follows:

```
> c6x-uclinux-sprite [options] device-url board-file
```

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

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

Most device URL schemes also follow the regular format:

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

The meanings of *`hostname`*, *`port`*, *`path`* and *`device-options`* parts depend on the *`scheme`* and are described below. The following schemes are supported in Sourcery CodeBench Lite for C6000 uClinux:

`xds`	Use a Texas Intruments XDS debugging device. Refer to Section 5.4, “Texas Instruments XDS Devices”.

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

The *`board-file`* specifies an XML file that describes how to initialize the target board, as well as other properties of the board used by the debugger. If *`board-file`* refers to a file (via a relative or absolute pathname), it is read. Otherwise, *`board-file`* can be a board name, and the toolchain's board directory is searched for a matching file. See Section 5.6, “Supported Board Files” for the list of supported boards, or invoke the Sprite with the `-b` option to list the available board files. You can also write a custom board file; see Section 5.7, “Board File Syntax” for more information about the file format.

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

5.3. Sourcery CodeBench Debug Sprite Options

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

`-b`	Print a list of *`board-file`* files in the board config directory.
`-h`	Print a list of options and their meanings. A list of *`device-url`* syntaxes is also shown.
`-i`	Print a list of the accessible devices. If a *`device-url`* is also specified, only devices for that device type are scanned. Each supported device type is listed along with the options that can be appended to the *`device-url`*. For each discovered device, the *`device-url`* is printed along with a description of that device.
`-l` *`[host]:port`*	Specify the host address and port number to listen for a GDB connection. If this option is not given, the Debug Sprite communicates with GDB using stdin and stdout. If you start the Sprite from within GDB using the `target remote | c6x-uclinux-sprite ...` command, you do not need this option.
`-m`	Listen for multiple sequential connections. Normally the Debug Sprite terminates after the first connection from GDB terminates. This option instead makes it listen for a subsequent connection. To terminate the Sprite, open a connection and send the string `END\n`.
`-q`	Do not print any messages.
`-v`	Print additional messages.

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

5.4. Texas Instruments XDS Devices

The Sourcery CodeBench Debug Sprite for C6000 supports Texas Instruments debug probes via the XDS software API. There may only be one XDS device connected, thus the device URL is simply `xds:`, and the *`hostname`*, *`port`* and *`path`* must be empty or omitted.

Two additional *`device-options`* are required:

`emulation=`*`name`*	This option specifies the driver emulation for your debug device. Suitable values for *`name`* are `xds100v2` and `ezdsp`.
`device=`*`name`*	This option specifies the type of device to connect to. For example, `device=`*`C64XP`*.

5.5. Debugging a Remote Board

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

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

```
> c6x-uclinux-sprite -l :10000 device-url board-file
```

starts the Sprite listening on port 10000.

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

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

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

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

5.6. Supported Board Files

The Sourcery CodeBench Debug Sprite for C6000 uClinux includes support for the following target boards. Specify the appropriate *`board-file`* as an argument when invoking the Sprite from the command line.

No board files are included with this release.

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

5.7. Board File Syntax

The *board-file* can be a user-written XML file to describe a non-standard board. The Sourcery CodeBench Debug Sprite searches for board files in the `c6x-uclinux/lib/boards` directory in the installation. Refer to the files in that directory for examples.

The file's DTD is:

```
<!-- Board description files

     Copyright (c) 2007-2009 CodeSourcery, Inc.

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

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

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

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

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

```
                        value   CDATA   #REQUIRED
                        timeout CDATA   #IMPLIED
                        bits    CDATA   #IMPLIED>

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

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

<!-- Definition of default option values for each debug interface -->
<!ELEMENT debuggerDefaults (debugInterface*)>
<!ELEMENT debugInterface (option*)>
<!ATTLIST debugInterface
 name CDATA #REQUIRED
>
<!ELEMENT option EMPTY>
<!ATTLIST option
 name CDATA #REQUIRED
 defaultValue CDATA #REQUIRED
>

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

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

The following elements are available:

`<board>`	This top-level element encapsulates the entire description of the board. It can contain `<properties>`, `<feature>`, `<initialize>` and `<memory-map>` elements.
`<properties>`	The `<properties>` element specifies specific properties of the target system. This element can occur at most once. It can contain a `<description>` element.
`<initialize>`	The `<initialize>` element defines an initialization sequence for the board, which the Sprite performs before downloading a program. It can contain `<write-register>`, `<write-memory>` and `<delay>` elements.

`<feature>`	This element is used to inform GDB about additional registers and peripherals available on the board. It is passed directly to GDB; see the GDB manual for further details.
`<memory-map>`	This element describes the memory map of the target board. It is used by GDB to determine where software breakpoints may be used and when flash programming sequences must be used. This element can occur at most once. It can contain `<memory-device>` elements.
`<memory-device>`	This element specifies a region of memory. It has four attributes: `address`, `size`, `type` and `device`. The `address` and `size` attributes specify the location of the memory device. The `type` attribute specifies that device as `ram`, `rom` or `flash`. The `device` attribute is required for `flash` regions; it specifies the flash device type. The `<memory-device>` element can contain a `<description>` element.
`<write-register>`	This element writes a value to a control register. It has three attributes: `address`, `value` and `bits`. The `bits` attribute, specifying the bit width of the write operation, is optional; it defaults to 32.
`<write-memory>`	This element writes a value to a memory location. It has three attributes: `address`, `value` and `bits`. The `bits` attribute is optional and defaults to 32. Bit widths of 8, 16 and 32 bits are supported. The address written to must be naturally aligned for the size of the write being done.
`<delay>`	This element introduces a delay. It has one attribute, `time`, which specifies the number of milliseconds, or microseconds to delay by.
`<description>`	This element encapsulates a human-readable description of its enclosing element.
`<property>`	The `<property>` element allows additional name/value pairs to be specified. The property name is specified in a `name` attribute. The property value is the body of the `<property>` element.

Chapter 6
Next Steps with Sourcery CodeBench

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

6.1. 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.

6.2. Example Programs

Sourcery CodeBench Lite includes some bundled example programs. You can find the source code for these examples in the `share/sourceryg++-c6x-uclinux-examples` directory of your Sourcery CodeBench installation.

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

6.3. 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/sourceryg++-c6x-uclinux/` 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/sourceryg++-c6x-uclinux/man/man1
```

Then you can invoke `man` as:

```
> man ./c6x-uclinux-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://support.codesourcery.com/GNUToolchain/

Appendix A Sourcery CodeBench Lite Release Notes

This appendix contains information about changes in this release of Sourcery CodeBench Lite for C6000 uClinux. You should read through these notes to learn about new features and bug fixes.

A.1. Changes in Sourcery CodeBench Lite for C6000 uClinux

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

A.1.1. Changes in Sourcery CodeBench Lite 4.5-124

Relocation overflow fix. Runtime support routines are now built with `-msdata=none` to avoid a relocation overflow when linking large executables or shared objects.

Floating point scheduling bug fix. A compiler bug has been fixed which could cause incorrect code to be generated during scheduling for floating point instructions on C674X.

C6X intrinsic functions. A selection of intrinsic functions to access specific C6X instructions has been added to the compiler.

C library header file fix. A bug in `ipc.h` has been corrected. The defect caused an incorrect structure layout on big-endian targets.

C66x support in `gdbserver`. Support for C66x processors has been added to `gdbserver`.

Device/options available as XML. The Sourcery CodeBench Debug Sprite can now generate XML output for devices and options when requested.

A.1.2. Changes in Sourcery CodeBench Lite 4.5-118

CodeSourcery Common Startup Code Sequence. Support for CS3, a unified startup scheme is included.

A.1.3. Changes in Sourcery CodeBench Lite 4.5-117

Fix for dynamic libraries with text relocs. A bug in the uClibc dynamic linker has been fixed that would prevent shared libraries with text relocs to be loaded correctly if they had dependencies on other libraries.

A.1.4. Changes in Sourcery CodeBench Lite 4.5-115

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.

Atomic builtin functions. The compiler now implements the atomic `__sync` family of builtin functions for `int` and `long` types.

Setting the ELF OSABI field. The linker has been changed to set the ELF OSABI field in executables and shared libraries as specified by the C6X ELF ABI.

Changes in object file attributes. The compiler and linker now behave differently when generating the PIC and PID object file attributes; Tag_ABI_PIC is generated only if `-fpic` or `-fPIC` is used. The new behaviour is incompatible with older versions of the toolchain.

Support for new relocations. The assembler and linker now support the new `R_C6000_PCR_H16` and `R_C6000_PCR_L16` relocations.

A.1.5. Changes in Sourcery G++ Lite 4.5-109

Alignment of `malloc` return values. A bug in uClibc has been fixed which caused `malloc` to return memory that was aligned to four bytes only, rather than eight.

A.1.6. Changes in Sourcery G++ Lite 4.5-108

Compiler crash with `-mlong-calls`. A bug that could cause the compiler to crash in certain situations with the `-mlong-calls` option has been fixed.

Incorrect code generation when scheduling. A compiler bug has been fixed which could cause incorrect code to be generated during scheduling.

Dynamic assignment of DSBT indices. The uClibc dynamic linker can now dynamically assign an index to a DSBT shared library that was compiled without a `--dsbt-index` option. This is not recommended in general as it generates private mappings of library text segments, requiring extra space and load time.

uClibc `clock_nanosleep` added. The function `clock_nanosleep` has been added to uClibc's librt. Due to lack of NPTL threading, it may not fully work in threaded cases.

A.1.7. Changes in Sourcery G++ Lite 4.5-106

Linker bug fix. A bug in the linker that caused incorrect output for C++ exception tables emitted by the TI compiler has been fixed.

New version of uClibc. A new version of uClibc, based on upstream git mainline, has been imported. This results in an ABI change; all programs and libraries built with earlier releases must be rebuilt.

Inferior calling support. The included version of GDB has been updated to provide inferior call support. A bug causing GDB to obtain the return address incorrectly has also been fixed.

A.1.8. Changes in Sourcery G++ Lite 4.5-104

C++ exception handling. Sourcery G++ Lite for C6000 uClinux now includes support for C++, including exception handling using the unwinding tables defined by the C6000 EABI.

Position independent code generation. The compiler no longer accepts the `-fpic` option without `-mdsbt`. Also, the code generator has been fixed not to emit `addkpc` instructions on C62X.

uClibc `ffsl` and `ffsll` functions. The uClibc library now includes implementations of the functions `ffsl` and `ffsll`.

Error reading FDPIC exec loadmap message. A bug has been fixed that caused gdb to fail with the error message "Error reading FDPIC exec loadmap" when the "set sysroot" command is used before connecting to the target.

Segmentation fault fixed. A bug causing gdb to segfault when the "set sysroot" command is used with no file to debug has been fixed.

A.1.9. Changes in Sourcery G++ Lite 4.5-97

No significant changes. There are no significant changes for C6000 uClinux in this release.

A.1.10. Changes in Sourcery G++ Lite 4.5-96

No significant changes. There are no significant changes for C6000 uClinux in this release.

A.1.11. Changes in Sourcery G++ Lite 4.5-94

No significant changes. There are no significant changes for C6000 uClinux in this release.

A.1.12. Changes in Sourcery G++ Lite 4.5-92

No significant changes. There are no significant changes for C6000 uClinux in this release.

A.1.13. Changes in Sourcery G++ Lite 4.5-88

No significant changes. There are no significant changes for C6000 uClinux in this release.

A.1.14. Changes in Sourcery G++ Lite 4.5-86

No significant changes. There are no significant changes for C6000 uClinux in this release.

A.1.15. Changes in Sourcery G++ Lite 4.5-78

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

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

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

- Smaller and faster code for compound conditionals.
- Improved filling of branch delay slots.
- Removal of superfluous sign and zero extensions.

New `-fstrict-volatile-bitfields` option. The compiler has a new option, `-fstrict-volatile-bitfields`, which forces access to a volatile structure member using the width that conforms to its type. Refer to the GCC manual for details.

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

- More efficient assignment for structures containing bitfields.
- Better code for initializing C++ arrays with explicit element initializers.
- Improved logic for eliminating/combining redundant comparisons in code with nested conditionals.
- Better selection of loop variables, resulting in fewer temporaries and more efficient register usage.
- Better code when constant addresses are used as arguments to inline assembly statements.
- Better code for copying small constant strings.

GCC version 4.5.1. Sourcery G++ Lite for C6000 uClinux is now based on GCC version 4.5.1. For more information about changes from GCC version 4.4 that was included in previous releases, see `http://gcc.gnu.org/gcc-4.5/changes.html`.

A.1.16. Changes in Sourcery G++ Lite 4.4-326

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

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

GDB and Ctrl+C on Windows . GDB no longer crashes when you press **Ctrl**+**C** twice during remote debugging to give up waiting for the target.

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

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

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

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

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

Multi-process mode for `gdbserver`. The `gdbserver` utility has a new command-line option, `--multi`, that allows you to use it to debug multiple program instances. Refer to the Debugger manual for more information.

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

GDB `qOffsets` crash fix. GDB no longer crashes when a remote stub provides load offsets for an unlinked object file.

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

Improved breakpoints in constructors and template functions. GDB now supports breakpoints on source code locations that have several code addresses associated with them. Setting a breakpoint on a constructor automatically associates the breakpoint with all constructor bodies generated by

GCC. If you set a breakpoint on a line of a templated function, GDB breaks at the indicated line in all instantiations of the templated function.

GDB `printf %p`. GDB's `printf` command now supports the `"%p"` format specifier.

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

GDB update. The included version of GDB has been updated to 6.6.20070821. This update includes numerous bug fixes.

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

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

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

Printing global variables in GDB. A GDB bug that caused errors in printing values of global variables in the debugger has been fixed. GDB was formerly computing addresses of such variables incorrectly; in some cases, this resulted in incorrect values being printed, while in others, it resulted in memory access errors in the remote `gdbserver`.

Improved debugging for optimized code. GDB's ability to print and change variables' values in optimized code is improved. GDB now tracks variable scopes more accurately, making better use of the detailed debugging information produced by Sourcery G++ compilers.

Improved handling of Windows paths in GDB. GDB now properly recognizes the names of source files that were passed to the compiler using an absolute path on Windows. You may refer to the file either by its base name (without any leading directory components), by the exact path passed to the compiler, or by its absolute path.

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

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

Robustness on Microsoft Windows. Defects that sometimes caused GDB to become non-responsive on Microsoft Windows have been eliminated.

Memory access errors when setting breakpoints. A GDB bug that caused spurious "Cannot access memory" errors has been fixed. The errors occurred when setting breakpoints after the program being debugged exited or was killed.

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

GDB update. The included version of GDB has been updated to 6.6.50.20070228. This update includes numerous bug fixes and improved support for C++ pointers to members.

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

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

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

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

Inlined function debugging fix. GDB now backtraces correctly when stopped at the first instruction of an inlined function. Earlier versions would sometimes encounter internal errors in this situation.

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

GDB support for user-defined prefixed commands. The GDB `define` and `document` commands, which allow you to add new commands to the GDB command-line interface, now support creating commands within an existing prefix such as `target`. Hooks for prefixed commands are also supported. Refer to the Debugger manual for more information.

GDB update. The included version of GDB has been updated to 6.7.20080107. This update includes numerous bug fixes.

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

GDB `info registers` crash fix. Executing `info registers` after executing `flushregs` no longer crashes GDB.

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

GDB search path bug fix. A bug in GDB has been fixed that formerly resulted in an internal error when setting `solib-search-path` or `solib-absolute-prefix` after establishing a connection to a remote target.

Debugging of inlined functions. GDB now supports inlined functions. GDB can include inlined functions in the stack trace; display inlined functions' arguments and local variables; and step into, over, and out of inlined functions.

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

`gdbserver` support for execution wrappers. `gdbserver` has a new command-line option, `--wrapper`, which specifies a wrapper for any programs run by `gdbserver`. The specified wrapper can prepare the system and environment for the new program.

Debugger access to out-of-bounds memory. GDB turns on `inaccessible-by-default` by default, disallowing access to memory outside the regions specified in a board configuration.

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

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

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

GDB update. The included version of GDB has been updated to 6.6.50.20070620. This update includes numerous bug fixes.

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

A.1.17. Changes in Sourcery G++ Lite 4.4-322

No significant changes. There are no significant changes for C6000 uClinux in this release.

A.1.18. Changes in Sourcery G++ Lite 4.4-320

No significant changes. There are no significant changes for C6000 uClinux in this release.

A.1.19. Changes in Sourcery G++ Lite 4.4-319

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

A.1.20. Changes in Sourcery G++ Lite 4.4-316

Initial release. This is the initial public release for C6000 uClinux.

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.

B.1. Licenses for Sourcery CodeBench Lite Components

The table below lists the major components of Sourcery CodeBench Lite for C6000 uClinux and the license terms which apply to each of these components.

Some free or open-source components provide documentation or other files under terms different from those shown below. For definitive information about the license that applies to each component, consult the source package corresponding to this release of Sourcery 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
uClibc C Library	GNU Lesser General Public License 2.1 http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html
Linux Kernel Headers	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
GNU Make	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
GNU Core Utilities	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

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

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++-c6x-uclinux-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.2. Sourcery CodeBench™ Software License Agreement

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

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

3. **Definitions.**

   3.1. **Mentor Graphics Proprietary Components.** The 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. For a complete list, refer to the *Getting Started Guide* included with the distribution.

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

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

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

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

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

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

ents subject to the Eclipse Public License, contains information on how to obtain the source code. Definitive licensing information for each “free software” or “open source” component is available in the relevant source file.

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

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

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

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

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

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

13. **Limitation of Liability.** INDEPENDENT OF THE FORGOING PROVISIONS, IN NO EVENT AND UNDER NO LEGAL THEORY, INCLUDING WITHOUT LIMITATION, TORT, CONTRACT, OR STRICT PRODUCTS LIABILITY, SHALL MENTOR GRAPHICS

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

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

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

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

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

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

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

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

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

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

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

B.3. Attribution

This version of Sourcery 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
```