Sourcery G++ Lite

ARM EABI

Sourcery G++ Lite 2011.03-42

Getting Started

Sourcery G++ Lite: ARM EABI: Sourcery G++ Lite 2011.03-42: Getting Started

CodeSourcery, Inc.

Abstract

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

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

Table of Contents

Preface v
- 1. Intended Audience vi
- 2. Organization vi
- 3. Typographical Conventions vii

1. Quick Start 1
- 1.1. Installation and Set-Up 2
- 1.2. Configuring Sourcery G++ Lite for the Target System 2
- 1.3. Building Your Program 2
- 1.4. Running and Debugging Your Program 2

2. Installation and Configuration 4
- 2.1. Terminology 5
- 2.2. System Requirements 5
- 2.3. Downloading an Installer 6
- 2.4. Installing Sourcery G++ Lite 6
- 2.5. Installing Sourcery G++ Lite Updates 9
- 2.6. Setting up the Environment 9
- 2.7. Uninstalling Sourcery G++ Lite 11

3. Sourcery G++ Lite for ARM EABI 13
- 3.1. Included Components and Features 14
- 3.2. Library Configurations 14
- 3.3. Using Flash Memory 15
- 3.4. Using VFP Floating Point 15
- 3.5. Fixed-Point Arithmetic 17
- 3.6. ABI Compatibility 17
- 3.7. ARM Profiling Implementation 18
- 3.8. Object File Portability 19

4. Using Sourcery G++ from the Command Line 20
- 4.1. Building an Application 21
- 4.2. Running Applications on the Target System 21
- 4.3. Running Applications in the Simulator 21
- 4.4. Running Applications from GDB 22

5. CS3™: The CodeSourcery Common Startup Code Sequence 24
- 5.1. Linker Scripts 25
- 5.2. Program Startup and Termination 27
- 5.3. Memory Layout 30
- 5.4. Interrupt Vectors and Handlers 32
- 5.5. Supported Boards for ARM EABI 33
- 5.6. Interrupt Vector Tables 35

6. Sourcery G++ Debug Sprite 37
- 6.1. Probing for Debug Devices 38
- 6.2. Debug Sprite Example 38
- 6.3. Invoking Sourcery G++ Debug Sprite 39
- 6.4. Sourcery G++ Debug Sprite Options 40
- 6.5. Remote Debug Interface Devices 40
- 6.6. Actel FlashPro Devices 41
- 6.7. Altera Devices 41
- 6.8. Debugging a Remote Board 42
- 6.9. Supported Board Files 43
- 6.10. Board File Syntax 43

7. Next Steps with Sourcery G++ 47
- 7.1. Sourcery G++ Knowledge Base 48

7.2. Example Programs 48
7.3. Manuals for GNU Toolchain Components 48
A. Sourcery G++ Lite Release Notes 49
A.1. Changes in Sourcery G++ Lite for ARM EABI 50
B. Sourcery G++ Lite Licenses 56
B.1. Licenses for Sourcery G++ Lite Components 57
B.2. Sourcery G++ Software License Agreement 58
B.3. Attribution 61

Preface

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

1. Intended Audience

This guide is written for people who will install and/or use Sourcery G++ Lite. This guide provides a step-by-step guide to installing Sourcery G++ Lite and to building simple applications. Parts of this document assume that you have some familiarity with using the command-line interface.

2. Organization

This document is organized into the following chapters and appendices:

Chapter 1, "Quick Start"	This chapter includes a brief checklist to follow when installing and using Sourcery G++ Lite for the first time. You may use this chapter as an abbreviated guide to the rest of this manual.
Chapter 2, "Installation and Configuration"	This chapter describes how to download, install and configure Sourcery G++ Lite. This section describes the available installation options and explains how to set up your environment so that you can build applications.
Chapter 3, "Sourcery G++ Lite for ARM EABI"	This chapter contains information about using Sourcery G++ Lite that is specific to ARM EABI targets. You should read this chapter to learn how to best use Sourcery G++ Lite on your target system.
Chapter 4, "Using Sourcery G++ from the Command Line"	This chapter explains how to build applications with Sourcery G++ Lite using the command line. In the process of reading this chapter, you will build a simple application that you can use as a model for your own programs.
Chapter 5, "CS3™: The CodeSourcery Common Startup Code Sequence"	CS3 is CodeSourcery's low-level board support library. This chapter documents the boards supported by Sourcery G++ Lite and the compiler and linker options you need to use with them. It also explains how you can use and modify CS3-provided definitions for memory maps, system startup code and interrupt vectors in your own code.
Chapter 6, "Sourcery G++ Debug Sprite"	This chapter describes the use of the Sourcery G++ Debug Sprite for remote debugging. The Sprite allows you to debug programs running on a bare board without an operating system. This chapter includes information about the debugging devices and boards supported by the Sprite for ARM EABI.
Chapter 7, "Next Steps with Sourcery G++"	This chapter describes where you can find additional documentation and information about using Sourcery G++ Lite and its components. It also provides information about Sourcery G++ subscriptions. CodeSourcery customers with Sourcery G++ subscriptions receive comprehensive support for Sourcery G++.
Appendix A, "Sourcery G++ Lite Release Notes"	This appendix contains information about changes in this release of Sourcery G++ Lite for ARM EABI. You should read through these notes to learn about new features and bug fixes.

Appendix B, "Sourcery G++ Lite Licenses"	This appendix provides information about the software licenses that apply to Sourcery G++ Lite. Read this appendix to understand your legal rights and obligations as a user of Sourcery G++ Lite.

3. Typographical Conventions

The following typographical conventions are used in this guide:

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

Chapter 1
Quick Start

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

Sourcery G++ Lite for ARM EABI is intended for developers working on embedded applications or firmware for boards without an operating system, or that run an RTOS or boot loader. This Sourcery G++ configuration is not intended for Linux or uClinux kernel or application development.

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

You can find additional details about the components, libraries, and other features included in this version of Sourcery G++ Lite in Chapter 3, "Sourcery G++ Lite for ARM EABI".

1.1. Installation and Set-Up

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

Install drivers for your debug device. If you plan to use the Sourcery G++ Debug Sprite, you may need to install drivers, libraries, or other software on your host system. Refer to Chapter 6, "Sourcery G++ Debug Sprite" for a list of supported devices and information about installing drivers and other device set-up. Sourcery G++ Lite also supports third-party debug devices that communicate via the GDB remote serial protocol. If you plan to use one of these devices, follow the manufacturer's directions to connect the device and install any required drivers or software.

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

Identify your target board. On bare-metal targets, you must explicitly specify a linker script for your target board on your link command line. Supported boards are listed in Chapter 5, "CS3™: The CodeSourcery Common Startup Code Sequence". You can also choose a simulator as your target board.

1.3. Building Your Program

Build your program with Sourcery G++ command-line tools. Create a simple test program, and follow the directions in Chapter 4, "Using Sourcery G++ from the Command Line" to compile and link it using Sourcery G++ Lite. On bare-metal targets, you must specify a linker script using the `-T` option on your link command line. Supported boards and linker scripts are listed in Chapter 5, "CS3™: The CodeSourcery Common Startup Code Sequence".

1.4. Running and Debugging Your Program

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

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

Run or debug your program in the simulator. Sourcery G++ Lite includes an instruction-set simulator, which provides an easy way to run or debug your program without requiring target hardware. The simulator can be run directly from the command line (see Section 4.3, “Running Applications in the Simulator”) or via the debugger (see Section 4.4, “Running Applications from GDB”).

Debug your program on the target using the Debug Sprite. You can use the Sourcery G++ Debug Sprite to load and execute your program on the target from the debugger. Refer to Section 4.4, “Running Applications from GDB” for instructions on using the Sprite from the GDB command line. Detailed reference material for the Sourcery G++ Debug Sprite, including information about supported debug devices, can be found in Chapter 6, “Sourcery G++ Debug Sprite”.

Debug your program on the target using a third-party debug device. Sourcery G++ supports debugging programs on the remote target using third-party debug devices that can communicate via the GDB remote serial protocol. For command-line GDB instructions, see Section 4.4, “Running Applications from GDB”.

Chapter 2
Installation and Configuration

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

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

2.1. Terminology

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

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

2.2. System Requirements

2.2.1. Host Operating System Requirements

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

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

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

Installing on Ubuntu and Debian GNU/Linux Hosts

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

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

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

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

2.2.2. Host Hardware Requirements

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

The amount of disk space required for a complete Sourcery G++ Lite installation directory depends on the host operating system and the number of target libraries included. When you start the graphical installer, it checks whether there is sufficient disk space before beginning to install. Note that the graphical installer also requires additional temporary disk space during the installation process. On Microsoft Windows hosts, the installer uses the location specified by the `TEMP` environment variable for these temporary files. If there is not enough free space on that volume, the installer prompts for an alternate location. On Linux hosts, the installer puts temporary files in the directory specified by the `IATEMPDIR` environment variable, or `/tmp` if that is not set.

2.2.3. Target System Requirements

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

2.3. Downloading an Installer

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

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

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

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

2.4. Installing Sourcery G++ Lite

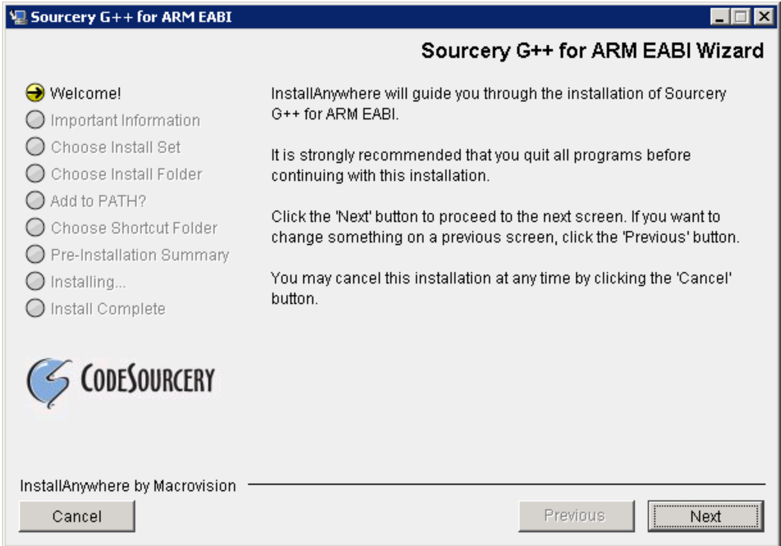
The method used to install Sourcery G++ Lite depends on your host system and the kind of installation package you have downloaded.

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

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

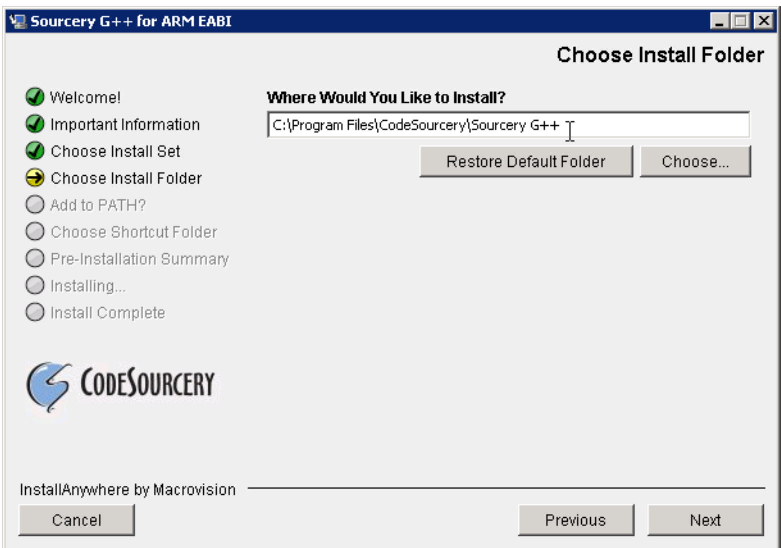
After the installer starts, follow the on-screen dialogs to install Sourcery G++ Lite. The installer is intended to be self-explanatory and on most pages the defaults are appropriate.

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

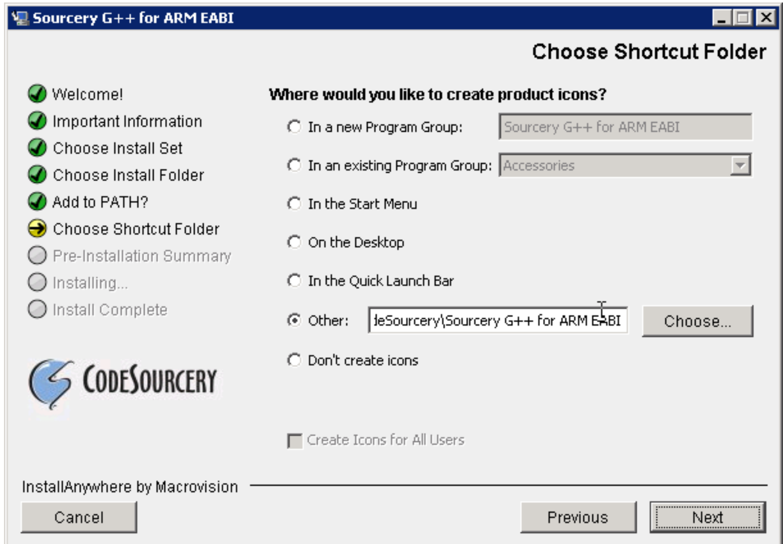


Running the Installer. The graphical installer guides you through the steps to install Sourcery G++ Lite.

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

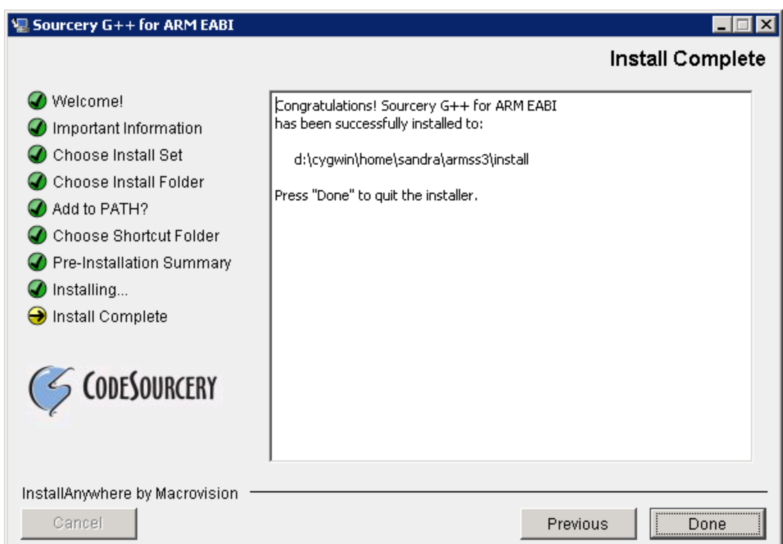


Choose Install Folder. Select the pathname to your install directory.



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

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



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

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

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

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

Start the graphical installer by invoking the executable shell script:

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

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

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

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

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

You do not need to be a system administrator to install Sourcery G++ Lite from a compressed archive. You may install Sourcery G++ Lite using any user account and in any directory to which you have write access. This guide assumes that you have decided to install Sourcery G++ Lite in the `$HOME/CodeSourcery` subdirectory of your home directory and that the filename of the package you have downloaded is `/path/to/package.tar.bz2`. After installation the toolchain will be in `$HOME/CodeSourcery/sourceryg++-2011.03`.

First, uncompress the package file:

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

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

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

Change to the installation directory:

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

Unpack the package:

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

2.5. Installing Sourcery G++ Lite Updates

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

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

Note that the names of the Sourcery G++ commands for the ARM EABI target all begin with `arm-none-eabi`. This means that you can install Sourcery G++ for multiple target systems in the same directory without conflicts.

2.6. Setting up the Environment

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

2.6.1. Setting up the Environment on Microsoft Windows Hosts

2.6.1.1. Setting the PATH

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

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

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

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

To set the `PATH` on a system running Microsoft Windows 7, from the desktop bring up the `Start` menu and right click on `Computer`. Select `Properties` and click on `Advanced system settings`. Go to the `Advanced` tab, then click on the `Environment Variables` button. Select the `PATH` variable and click the `Edit`. Add the string `;C:\Program Files\Sourcery G++\bin` to the end, and click `OK`. Be sure to adjust the pathname to reflect your actual installation directory.

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

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

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

Verify that the last line of the output contains: `Sourcery G++ Lite 2011.03-42`.

2.6.1.2. Working with Cygwin

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

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

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

```
> export CYGPATH=cygpath
```

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

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

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

2.6.2. Setting up the Environment on GNU/Linux Hosts

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

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

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

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

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

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

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

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

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

Verify that the last line of the output contains: `Sourcery G++ Lite 2011.03-42`.

2.7. Uninstalling Sourcery G++ Lite

The method used to uninstall Sourcery G++ Lite depends on the method you originally used to install it. If you have modified any files in the installation it is recommended that you back up these changes. The uninstall procedure may remove the files you have altered. In particular, the `arm-none-eabi` directory located in the install directory will be removed entirely by the uninstaller.

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

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

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

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

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

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

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

2.7.3. Uninstalling a Compressed Archive Installation

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

Chapter 3
Sourcery G++ Lite for ARM EABI

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

3.1. Included Components and Features

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

Component	Version	Notes
GNU programming tools		
GNU Compiler Collection	4.5.2	Separate manual included.
GNU Binary Utilities	2.20.51	Includes assembler, linker, and other utilities. Separate manuals included.
Debugging support and simulators		
GNU Debugger	7.2.50	Separate manual included.
Sourcery G++ Debug Sprite for ARM	2011.03-42	See Chapter 6, “Sourcery G++ Debug Sprite”.
GDB Simulator	N/A	See Section 4.3, “Running Applications in the Simulator”.
Target libraries		
CodeSourcery Common Startup Code Sequence	2011.03-42	See Chapter 5, “CS3™: The CodeSourcery Common Startup Code Sequence”.
Newlib C Library	1.18.0	Separate manuals included.
Other utilities		
GNU Make	N/A	Build support on Windows hosts.
GNU Core Utilities	N/A	Build support on Windows hosts.

3.2. Library Configurations

Sourcery G++ Lite for ARM EABI includes the following library configuration.

ARMv4 - Little-Endian, Soft-Float	
Command-line option(s):	default
Library subdirectory:	`./`

ARMv4 Thumb - Little-Endian, Soft-Float	
Command-line option(s):	`-mthumb`
Library subdirectory:	`thumb/`

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

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

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

Sourcery G++ Lite's library support includes linker scripts that pull in appropriate CS3 startup code, as well as the libraries themselves. You can find these linker scripts in multilib-specific subdirectories of the `arm-none-eabi/lib` directory of your Sourcery G++ install.

3.3. Using Flash Memory

Sourcery G++ Lite supports development and debugging of applications loaded into flash memory on ARM EABI targets. There are three steps involved:

1. You must use an appropriate linker script that identifies the ROM memory region on your target board, and locates the program text within that region. Refer to Chapter 5, “CS3™: The CodeSourcery Common Startup Code Sequence” for information about the boards supported by Sourcery G++.

2. Next, load your program into the flash memory on your target board. You must use third-party tools to program the flash memory.

3. Finally, when debugging a program in flash memory, GDB must be told about the ROM region so that it knows where it must use hardware breakpoints to control program execution. If you are using the Sourcery G++ Debug Sprite to debug your program, the Sprite does this automatically, using the memory map provided in the board configuration file. Otherwise, you must provide this information explicitly.

   When using GDB from the command line, you can mark the flash memory as read-only by using the command:

   ```
   (gdb) mem start end ro
   ```

   where *`start`* and *`end`* define the address range of the read-only memory region.

In addition to GDB's automatic use of hardware breakpoints in the ROM region, you can also set hardware breakpoints explicitly from the debugger. However, on many targets the number of available hardware breakpoints is very small. Furthermore, GDB also uses hardware breakpoints internally to implement commands such as `step`, `next`, and `finish`. Thus the number of breakpoints you can explicitly set in ROM may be fewer than the number of hardware breakpoints supported by the target system.

For example, ARM7TDMI cores support only one hardware breakpoint, which must also be used internally by the debugger if you set any software breakpoints in RAM. On ARM9 cores, there are two hardware breakpoints supported and one is consumed by the debugger if you set any software breakpoints.

3.4. Using VFP Floating Point

3.4.1. Enabling Hardware Floating Point

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

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

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

Sourcery G++ Lite for ARM EABI 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 G++ Lite. However, VFP hard-float libraries built with both ABIs are available to Sourcery G++ 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.

3.4.2. NEON SIMD Code

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

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

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

3.4.3. Half-Precision Floating Point

Sourcery G++ for ARM EABI 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.

3.5. Fixed-Point Arithmetic

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

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

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

3.6. ABI Compatibility

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

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

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

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

3.7. ARM Profiling Implementation

Profiling is enabled by means of the `-pg` compiler option. In this mode, the compiler inserts a call to `__gnu_mcount_nc` into every function prologue. However, no implementation of `__gnu_mcount_nc` is provided (to do so would be impossible without knowledge of the execution environment).

You must provide your own implementation of `__gnu_mcount_nc` . Here are the requirements:

- On exit, pop the top value from the stack, and place it in the `lr` register. The `sp` register should be adjusted accordingly. For example, this is how to write it as a stub function:

```
        .globl __gnu_mcount_nc
        .type __gnu_mcount_nc, %function
__gnu_mcount_nc:
        mov    ip, lr
        pop    { lr }
        bx     ip
```

- Preserve all other register state except for `r12` and the CPSR condition code bits. In particular all coprocessor state and registers `r0`-`r3` must be preserved.
- Record and count all occurrences of the function calls in the program. The caller can be determined from the `lr` value stored on the top of the stack (on entry to `__gnu_mcount_nc`), and the callee can be determined from the current value of the `lr` register (i.e. the caller of this function).

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

- Arrange for the data to be saved to a file named `gmon.out` when the program exits (via `atexit`). Refer to the `gprof` profiler manual for more information.

3.8. Object File Portability

It is possible to create object files using Sourcery G++ for ARM EABI that are link-compatible with the GNU C library provided with Sourcery G++ 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 G++ from the Command Line

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

4.1. Building an Application

This chapter explains how to build an application with Sourcery G++ Lite using the command line. As elsewhere in this manual, this section assumes that your target system is arm-none-eabi, as indicated by the `arm-none-eabi` 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-eabi-gcc -o factorial main.c -T script
```

Sourcery G++ requires that you specify a linker script with the `-T` option to build applications for bare-board targets. Linker errors like `undefined reference to `read'` are a symptom of failing to use an appropriate linker script. Default linker scripts are provided in `arm-none-eabi/lib`. Refer to Chapter 5, “CS3™: The CodeSourcery Common Startup Code Sequence” for information about the boards and linker scripts supported by Sourcery G++ Lite. You must also add the processor options for your board, as documented in that chapter, to your compile and link command lines.

There should be no output from the compiler. (If you are building a C++ application, instead of a C application, replace `arm-none-eabi-gcc` with `arm-none-eabi-g++`.)

4.2. Running Applications on the Target System

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

4.3. Running Applications in the Simulator

Sourcery G++ Lite includes a simulator that you can use on the host system to run programs compiled for the target system. Since you do not need target hardware, this is the easiest way to try out Sourcery G++.

To use the simulator run:

```
> arm-none-eabi-run factorial
```

You should see the expected output:

```
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
```

You can also use the simulator to execute target programs when debugging with GDB. See Section 4.4, “Running Applications from GDB” for more information.

The simulator supports the ARMv4 (StrongARM), ARMv4T (ARM7TDMI, ARM920, ARM9TDMI), ARMv5, and ARMv5TE (ARM926, Xscale) instruction sets. The arm-none-eabi-run simulator also includes support for Thumb instructions.

4.4. Running Applications from GDB

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

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-eabi-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.

4.4.1. Connecting to the GDB Simulator

GDB includes a simulator that allows you to debug ARM EABI applications without target hardware. To start and connect to the simulator from within GDB, use this command:

```
(gdb) target sim
```

4.4.2. Connecting to the Sourcery G++ Debug Sprite

The Sourcery G++ Debug Sprite is a program that runs on the host system to support hardware debugging devices. You can use the Debug Sprite to run and debug programs on a target board without an operating system, or to debug an operating system kernel. See Chapter 6, “Sourcery G++ Debug Sprite” for detailed information about the supported devices.

You can start the Sprite directly from within GDB:

```
(gdb) target remote | arm-none-eabi-sprite arguments
```

Refer to Section 6.3, “Invoking Sourcery G++ Debug Sprite” for a full description of the Sprite arguments.

4.4.3. Connecting to an External GDB Server

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

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

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

4.4.4. Loading and Running Applications

Connecting to a bare-metal target or simulator from GDB does not cause your program to be loaded into target memory. You must do this explicitly from GDB after you connect:

```
(gdb) load
```

Alternatively, you can use third-party tools to load your application into flash memory before starting GDB.

To begin execution of your application, you should generally use the `continue` command:

```
(gdb) continue
```

However, you should use `run` instead of `continue` to start your program if you used `target sim` to connect:

```
(gdb) run
```

Chapter 5
CS3™: The CodeSourcery Common Startup Code Sequence

CS3 is CodeSourcery's low-level board support library. This chapter documents the boards supported by Sourcery G++ Lite and the compiler and linker options you need to use with them. It also explains how you can use and modify CS3-provided definitions for memory maps, system startup code and interrupt vectors in your own code.

Many developers turn to the GNU toolchain for its cross-platform consistency: having a single system support so many different processors and boards helps to limit risk and keep learning curves gentle. Historically, however, the GNU toolchain has lacked a consistent set of conventions for processor- and board-level initialization, language run-time setup, and interrupt and trap handler definition.

The CodeSourcery Common Startup Code Sequence (CS3) addresses this problem. For each supported system, CS3 provides a set of linker scripts describing the system's memory map, and a board support library providing generic reset, startup, and interrupt handlers. These scripts and libraries all follow a standard set of conventions across a range of processors and boards.

In addition to providing linker support, CS3's functionality is fully integrated with the Sourcery G++ Debug Sprite. For each supported board, CS3 provides the board file containing the memory map and initialization sequence required for debugging applications on the board via the Sprite, as documented in Section 6.9, “Supported Board Files”.

This chapter is organized in two parts. The first part explains CS3 concepts:

- Section 5.1, “Linker Scripts” provides basic information you need to know in order to select an appropriate CS3-provided linker script for your ARM EABI board.
- CS3's program startup and termination model is discussed in Section 5.2, “Program Startup and Termination”.
- Section 5.3, “Memory Layout” discusses the mapping from program sections to memory regions. It also explains how you can refer to memory regions using CS3-provided symbolic names from C, assembly language, or the linker script, and customize placement of code or data in your program.
- Section 5.4, “Interrupt Vectors and Handlers” covers CS3's interrupt handling model, and discusses how you can customize the CS3-provided interrupt vector tables.

The second part provides details about the CS3 implementation for ARM EABI:

- Section 5.5, “Supported Boards for ARM EABI” lists the boards supported by CS3 for ARM EABI, and the available linker scripts for them.
- Section 5.6, “Interrupt Vector Tables” documents the details of the provided interrupt vectors for CS3-supported devices.

5.1. Linker Scripts

When you build programs for ARM EABI targets, you must use a linker script. The linker script serves several purposes:

- It determines the memory addresses for placement of code and data sections.
- It defines symbolic names for memory regions present on the board, which you can use programmatically within your code.
- It provides appropriate program startup and termination code, and causes the linker to pull in any low-level board support libraries that are required to run code on the target.
- It optionally provides a *hosting* library for basic I/O functionality.
- It provides a default interrupt vector appropriate for the target processor.

When invoking the Sourcery G++ linker from the command line, you must explicitly supply a linker script using the `-T` option; otherwise a link error results.

CS3 may provide multiple linker scripts for different configurations using the same board. For example, on some boards CS3 may support running the program from either RAM or ROM (flash). Some CS3 link configurations are also designed to co-exist with, or be run from, a boot monitor on the target board. Simulator targets typically require different startup code configurations than hardware targets. In CS3 terminology, each of these different configurations is referred to as a *profile*.

The remainder of this section discusses profile and hosting selection considerations in more detail. You can find the full list of supported boards and linker scripts included in this release of Sourcery G++ Lite in Section 5.5, “Supported Boards for ARM EABI”.

5.1.1. Program and Data Placement

Many boards have both RAM and ROM (flash) memory devices. CS3 provides distinct linker scripts to place the application either entirely in RAM, or to place code and read-only data in ROM.

Some boards have very small amounts of RAM memory. If you use large library functions (such as `printf` and `malloc`), you may overflow the available memory. You may need to use the ROM-based profile for such programs, so that the program itself is stored in ROM. You may be able to reduce the total amount of memory used by your program by replacing portions of the Sourcery G++ runtime library and/or startup code.

5.1.2. Hosting and Semihosting

CS3 is designed to support boards without an operating system. To allow functions like `open` and `write` to work without operating system support, a *semihosting* feature is supported, in conjunction with the debugger.

With semihosting enabled, these system calls are translated into equivalent function calls on your host system. You can only use these function calls while connected to the debugger; if you try to use them when disconnected from the debugger, you will get a hardware exception.

Semihosting requires support from the remote GDB debugging stub or agent, as well as the debugger itself. The Sourcery G++ Debug Sprite implements semihosting for all supported devices. Semihosting is also supported by the GDB Simulator included with Sourcery G++ Lite. However, semihosting may not be supported by debugging stubs provided by third parties. If you are using a debug device that communicates with GDB using the GDB remote protocol, check the documentation for your device to see whether semihosting is supported.

A good use of semihosting is to display debugging messages. For example, this program prints a message on the debugger console on the host:

```
#include <unistd.h>

int main () {
  write (STDERR_FILENO, "Hello, world!\n", 14);
  return 0;
}
```

The hosted CS3 linker scripts provide the semihosting support, and as such programs linked with them may only be run with the debugger. For production code, or programs where memory usage is tightly constrained, use the unhosted CS3 linker scripts instead. These scripts provide stub versions of the system calls, which return an appropriate error value in `errno`. If such a stub system call is required in the executable, the linker also produces a warning. Such a warning may indicate that you have left debugging code active, or that your program contains unused code.

As an alternative to semihosting via the debugger, some targets supported by CS3 can run a boot monitor that provides console I/O services and other basic system calls. CS3 can also provide hosting via these facilities; where a boot monitor is supported, this is noted in the board tables below. Unlike semihosting, hosting via the boot monitor can be used when running programs outside of the debugger.

5.1.3. Specifying a Linker Script

When using Sourcery G++ from the command line or from a `Makefile`, you must add `-T` *`script`* to your linking command, where *`script`* is the appropriate linker script. For example, to target ARMulator (RDI) boards, you could link with `-T armulator-ram-hosted.ld`.

5.2. Program Startup and Termination

This section documents CS3's model for target initialization prior to invoking the `main` function of your program, and aspects of program termination that are left unspecified in the C and C++ standards. It explains how you can customize or override the default behavior for your application.

CS3 divides the startup sequence into three phases:

- The *hard reset phase* (`__cs3_reset`) includes actions such as initializing the memory controller and setting up the memory map.
- The *assembly initialization phase* (`__cs3_start_asm`) prepares the stack to run C code, and jumps to the C initialization function.
- The *C initialization phase* (`__cs3_start_c`) is responsible for initializing the data areas, running constructors for statically-allocated objects, and calling `main`.

The hard reset and assembly initialization phases are necessarily written in assembly language; at reset, there may not yet be stack to hold compiler temporaries, or perhaps even any RAM accessible to hold the stack. These phases do the minimum necessary to prepare the environment for running simple C code. Then, the code for the final phase may be written in C; CS3 leaves as much as possible to be done at this point.

The CodeSourcery board support library provides default code for all three phases. The hard reset phase is implemented by board- and profile-specific code. The assembly initialization phase is implemented by profile-specific code. The C initialization phase is implemented by generic code.

5.2.1. The Hard Reset Phase

This phase, which begins at `__cs3_reset`, is responsible for initializing board-specific registers, such as memory base registers and DRAM controllers, or scanning memory to check the available size. It is written in assembler and ends with a jump to `__cs3_start_asm`, which is where the assembly initialization phase begins.

The hard reset code is in a section named `.cs3.reset`. CS3 linker scripts define `__cs3_reset` as an alias for a board- and profile-specific entry point. You may override the CS3-provided reset code by defining your own `__cs3_reset` entry point in the `.cs3.reset` section.

Program execution always begins at `__cs3_reset`, whether the program is started from the reset vector, the debugger, or a boot monitor. However, the `__cs3_reset` code linked into the application is typically non-empty only for ROM-based profiles. For example, in a RAM-based profile, resetting the memory controllers would overwrite the code being executed.

When using the Sourcery G++ Debug Sprite, the Sprite is responsible for carrying out the hard reset actions before the program is loaded onto the target. This is performed prior to execution of both RAM- and ROM-profile applications from the debugger. Thus, when debugging a ROM-profile application, hard reset is actually performed twice — once by the Sprite, and once by the application itself.

5.2.2. The Assembly Initialization Phase

This phase is responsible for initializing the stack pointer and creating an initial stack frame. The symbol `__cs3_start_asm` marks the entry point of the assembly initialization code. The assembly initialization phase ends with a call or jump to `__cs3_start_c`.

The assembly initialization phase is profile-specific. For example, while bare-board applications typically must initialize the stack themselves, CS3 also supports boot-monitor profiles where the stack is initialized by the boot monitor before it launches the application. Likewise, some simulators automatically initialize the stack pointer and initial stack frame on startup, while others require a supervisory operation on startup to determine the amount of available memory. Each of these scenarios requires different assembly initialization behavior.

Note that on bare-board targets setting the stack pointer explicitly in the assembly initialization phase is required even if the processor itself initializes the stack pointer automatically on reset. This is to support running programs from the debugger as well as from processor reset.

For backwards compatibility with previous versions of CS3, on RAM and ROM profiles the symbol `__cs3_start_asm` is actually an alias for a symbol named `_start`. However, referencing or defining `_start` directly is now deprecated.

The value of the symbol `__cs3_stack` provides the initial value of the stack pointer for profiles that must set it explicitly. The CodeSourcery linker scripts provide a default value for this symbol, which you may override by defining `__cs3_stack` yourself. See Section 5.3.3, “Heap and Stack Placement” for an example of a custom stack.

The initial stack frame is created for the use of ordinary C and C++ calling conventions. The stack should be initialized so that backtraces stop cleanly at this point; this might entail zeroing a dynamic link pointer, or providing hand-written DWARF call frame information.

The last action of the assembly initialization phase is to call the C function `__cs3_start_c`. This function never returns, and `__cs3_start_asm` need not be prepared to handle a return from it.

As with the hard reset code, the CodeSourcery board support library provides reasonable default assembly initialization code. However, you may provide your own code by providing a definition for `__cs3_start_asm`, either in an object file or a library.

5.2.3. The C Initialization Phase

Finally, C code can be executed. The C startup function is declared as follows:

```
void __cs3_start_c (void) __attribute__ ((noreturn));
```

This function performs the following steps:

- Initialize all `.data`-like sections by copying their contents. For example, ROM-profile linker scripts use this mechanism to initialize writable data in RAM from the read-only data program image.
- Clear all `.bss`-like sections.
- Run constructors for statically-allocated objects, recorded using whatever conventions are usual for C++ on the target architecture.

  CS3 reserves priorities from 0 to 100 for use by initialization code. You can handle tasks like enabling interrupts, initializing coprocessors, pointing control registers at interrupt vectors, and so on by defining constructors with appropriate priorities.
- Call `main` as appropriate.
- Call `exit`, if it is available.

As with the hard reset and assembly initialization code, the CodeSourcery board support library provides a reasonable definition for the `__cs3_start_c` function. You may override this by providing a definition for `__cs3_start_c`, either in an object file or in a library.

5.2.4. Arguments to `main`

The CodeSourcery-provided definition of `__cs3_start_c` can pass command-line arguments to `main` using the normal C `argc` and `argv` mechanism if the board support package provides corresponding definitions for `__cs3_argc` and `__cs3_argv`. For example:

```
int __cs3_argc;
char **__cs3_argv;
```

These variables should be initialized using a constructor function, which is run by `__cs3_start_c` after it initializes the data segment. Use the `constructor` attribute on the function definition:

```
__attribute__((constructor))
static void __cs3_init_args (void) {
   __cs3_argc = ...;
   __cs3_argv = ...;
}
```

The constructor function may have an arbitrary name; `__cs3_init_args` is used only for illustrative purposes here.

If definitions of `__cs3_argc` and `__cs3_argv` are not provided, then the default `__cs3_start_c` function invokes `main` with zero as the `argc` argument and a null pointer as `argv`.

5.2.5. Program Termination

A program running on an embedded system is usually designed never to exit — it runs until the system is powered down. The C and C++ standards leave it unspecified as to whether `exit` is called at program termination. If the program never exits, then there is no reason to include `exit`, facilities to run functions registered with `atexit`, or global destructors. This code would never be run and would therefore just waste space in the application.

The CS3 startup code, by itself, does not cause `exit` to be present in the application. It dynamically checks whether `exit` is present, and only calls it if it is. If you require `exit` to be present, either refer to it within your application, or add `-Wl,-u,exit` to the linking command line.

Similarly, code to register global destructors is only invoked when `atexit` is already in the executable; CS3, by itself, does not cause `atexit` to be present. If you require `atexit`, either refer to it within your application, or add `-Wl,-u,atexit` to the linking command line.

5.3. Memory Layout

Boards supported by CS3 can have multiple banks or regions of memory with different characteristics. This section describes how program sections are mapped onto memory regions, and how you can use these CS3 features to customize placement of your program's code or data in memory. CS3 also provides a uniform set of symbolic names for each region, allowing you to programmatically refer to each region's address range from C or assembly language as well as from the linker script.

5.3.1. Memory Regions and Program Sections

The regions that are available on a particular board are listed in the table for that board in Section 5.5, “Supported Boards for ARM EABI”, below. There are two kinds of regions: those documented as "Memory regions", which are general-purpose memory banks that can be used for program or data storage; and those documented as "Other regions", which typically correspond to memory-mapped control registers or other special-purpose storage.

CS3 supports boards that include both `ram` and `rom` memory regions. The `ram` region holds the `.data` and `.bss` sections, and the `.text` section in RAM profiles. In ROM profiles, the `rom` region holds the `.text` section and initialization values for the writable data sections.

In addition, all regions documented as "Memory regions" correspond to similarly-named program sections. For example, the linker script assigns the `.ram` section to the `ram` region.

More generally, for a memory region named *`R`*, CS3 linker scripts define a section named `.`*`R`*, which may contain initialized data or code. There is also a section named `.bss.`*`R`* for zero-initialized data (BSS), which is placed after the initialized data section for this region.

You can explicitly locate data or code in a section corresponding to a particular memory region using section attributes in your source C or C++ code. Section attributes are especially useful on code compiled for boards that include special memory banks, such as a fast on-chip cache memory, in addition to the default `ram` and/or `rom` regions. CS3's start-up code arranges for additional data-like sections to be initialized in the same way as the default `.data` section.

As an example to illustrate the attribute syntax, you can put a variable `v` in the `.ram` section using:

```
int v __attribute__ ((section (".ram")));
```

To declare a function `f` in this section, use:

```
int f (void) __attribute__ ((section (".ram"))) {...}
```

For more information about attribute syntax, see the GCC manual.

In addition to the `.`*`R`* and `.bss.`*`R`* sections, CS3 places a `.cs3.region-head.`*`R`* section at the beginning of each region *`R`*. Explicitly placing data in `.cs3.region-head.`*`R`* sections is discouraged, because CS3 itself may want to place items (like interrupt vector tables) at these locations. If there is a conflict, CS3 raises an error at link time.

Regions documented as "Other regions" in the tables in Section 5.5, “Supported Boards for ARM EABI” do not have corresponding program sections. Typically, these regions contain memory-mapped control and I/O registers and cannot be used for general data or program storage. If your program needs to manipulate data in these regions, you can use the CS3 memory map access interface declared in `cs3.h`, as described in Section 5.3.2, “Programmatic Access to the CS3 Memory Map”.

Memory maps for boards supported by Sourcery G++ Lite for ARM EABI are documented in XML files in the `arm-none-eabi/lib/boards/` subdirectory of your Sourcery G++ installation directory.

5.3.2. Programmatic Access to the CS3 Memory Map

CS3 makes C declarations describing the memory regions on the target board available to your program via the header file `cs3.h`, which you can find in the `arm-none-eabi/include` directory within your install.

For each region named *R*, `cs3.h` declares a byte array variable `__cs3_region_start_R` at the region's start address, and a `size_t` variable `__cs3_region_size_R` to represent the total size of the region. These symbols are defined by the linker script and so may also be referenced from assembly language. Note that all regions are aligned on eight-byte boundaries and sizes are also multiples of eight bytes.

For memory regions that can correspond to program sections (as described in Section 5.3.1, “Memory Regions and Program Sections”), there are additional symbols `__cs3_region_init_R` and `__cs3_region_init_size_R` that describe constant data used to initialize the region. During the C initialization phase (Section 5.2, “Program Startup and Termination”), this data is copied into the lower part of the memory region. The symbol `__cs3_region_zero_size_R` represents the size of the zero-initialized `.bss.R` section following the initialized data. Any of these identifiers may actually be defined as a preprocessor macro that expands to an expression of the appropriate type and value.

To perform the memory region initializations during startup, CS3 internally uses the array variable `__cs3_regions`, which contains descriptors for all of the writable (RAM) memory regions. These descriptors are also exposed in `cs3.h`; refer to the header file for details.

5.3.3. Heap and Stack Placement

CS3 linker scripts provide default placement of the heap and stack in the RAM region. However, you can override the defaults by providing your own definitions of the associated CS3 variables. For example, you may put the heap and/or stack in some other memory region.

Heap placement is controlled by defining the symbol `__cs3_heap_start` at the beginning of the heap, and either the symbol `__cs3_heap_end` or the pointer variable `__cs3_heap_limit` to mark the end of the heap. For example, this fragment of C code places the heap in a region named `extsram`:

```
#define HEAPSIZE ...  /* However big you want to make it. */
unsigned char __cs3_heap_start[HEAPSIZE]
    __attribute__ ((section (".bss.extsram"), aligned(8)));
unsigned char *__cs3_heap_limit = __cs3_heap_start + HEAPSIZE;
```

The default initial stack pointer for bare-metal profiles is given by the symbol `__cs3_stack`, and the stack grows downward from this address. Stack initialization is discussed in more detail in Section 5.2.2, “The Assembly Initialization Phase”.

You can find C declarations for the CS3 heap and stack symbols in the header file `cs3.h`.

The `cs3.h` header file also defines a macro for creating a custom stack. The custom stack is created as a block of RAM in the zero-initialized data section (BSS). The specified size must be a compile-time constant. To account for alignment, the final size of the stack may be a few bytes less than the requested size. The symbol `__cs3_stack` is initialized to point to the last extent of the stack block, and is 16-byte aligned. For example, the following fragment of C code creates a stack of 8192 bytes:

```
#include <cs3.h>

CS3_STACK(2 * 4096);
```

As indicated in Section 5.2.2, “The Assembly Initialization Phase”, there are cases where a boot monitor or simulator overrides a custom stack.

5.4. Interrupt Vectors and Handlers

CS3 provides standard handlers for interrupts, exceptions and traps, but also allows you to define your own handlers as needed. In this section, we use the term *interrupt* as a generic term for this entire class of events.

Different processors handle interrupts in various ways, but there are two general approaches:

- Some processors fetch an address from an array indexed by the interrupt number, and jump to that address. We call these *address vector* processors.
- Others multiply the interrupt number by some constant factor, add a base address, and jump directly to that address. Here, the interrupt vector consists of blocks of code, so we call these *code vector* processors.
- Still other processors use a more complicated descriptor mechanism for the interrupt table.

M-profile processors like the Cortex-M3 use the address vector model. Classic ARM processors (including ARM7/ARM9 as well as Cortex-A/R series processors) are technically code vector processors. However, each vector slot only holds a single instruction. CS3 emulates the address vector model on these processors by placing an indirect branch instruction in each slot of the real exception vector. The remainder of this section assumes that you have some understanding of the specific requirements for your target; refer to the architecture manuals if necessary.

5.4.1. ARM EABI Interrupt Vector Implementation

On address vector processors, the CS3 library provides an array of pointers to interrupt handlers named `__cs3_interrupt_vector_`*`form`*, where *`form`* identifies the particular processor variant the vector is appropriate for. Each entry in the vector holds a reference to a symbol named `__cs3_isr_`*`name`*, where *`name`* is the customary name of that interrupt on the processor, or a number if there is no consistently used name. You can find the interrupt vector details in Section 5.6, “Interrupt Vector Tables”. The particular vector used by a given CS3-supported board is documented in the tables in Section 5.5, “Supported Boards for ARM EABI”.

CS3 provides a reasonable default definition for each `__cs3_isr_`*`name`* handler. Many of these symbols are aliased to a common handler routine. If your program stops at a default interrupt handler, its name as shown in backtraces may therefore not correctly reflect which interrupt occurred.

To override an individual handler, provide your own definition for the appropriate `__cs3_isr_`*`name`* symbol. The definition need not be placed in any particular object file section.

To override the entire interrupt vector, you can define `__cs3_interrupt_vector_form`. You must place this definition in a section named `.cs3.interrupt_vector`. The linker script reports an error if the `.cs3.interrupt_vector` section is empty, to ensure that the definition of `__cs3_interrupt_vector_form` occupies the proper section.

You may define the vector in C with an array of pointers using the `section` attribute to place it in the appropriate section. For example, to override the interrupt vector on ARMulator (RDI) boards, make the following definition:

```
typedef void handler(void);
handler *__attribute__((section (".cs3.interrupt_vector")))
  __cs3_interrupt_vector_arm[] =
{ ... };
```

5.4.2. Writing Interrupt Handlers

Interrupt handlers typically require special call/return and register usage conventions that are target-specific and beyond the scope of this document. In many cases, normal C functions cannot be used as interrupt handlers. For example, the EABI requires that the stack be 8-byte aligned, but on some ARMv7-M processors, only 4-byte stack alignment is guaranteed when calling an interrupt vector. This can cause subtle runtime failures, usually when 8-byte types are used.

As an alternative to writing interrupt handlers in assembly language, on ARM targets they may be written in C using the `interrupt` attribute. This tells the compiler to generate appropriate function entry and exit sequences for an interrupt handler. For example, to override the `__cs3_isr_undef` handler, use the following definition:

```
void __attribute__ ((interrupt)) __cs3_isr_undef (void)
{
  ... custom handler code ...
}
```

On ARM targets, the `interrupt` attribute also takes an optional parameter to specify the type of interrupt. Refer to the GCC manual for more details about attribute syntax and usage.

5.5. Supported Boards for ARM EABI

CS3 provides support for the following boards on ARM EABI targets.

Altera Cyclone III Cortex-M1		
Processor name:	Cortex-M1	
Processor options:	`-mcpu=cortex-m1 -mthumb`	
Memory regions:	`itcm`, `ram` (SRAM), `rom` (Flash)	
Interrupt vector:	`__cs3_interrupt_vector_micro`	
Linker scripts:	RAM Hosted	`cycloneiii-cm1-ram-hosted.ld`
	RAM Unhosted	`cycloneiii-cm1-ram.ld`
	ROM Hosted	`cycloneiii-cm1-rom-hosted.ld`
	ROM Unhosted	`cycloneiii-cm1-rom.ld`

ARM M-profile Simulator		
Processor name:	Cortex-M3	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram`	
Interrupt vector:	`__cs3_interrupt_vector_micro`	
Linker scripts:	Simulator Hosted	`generic-m-hosted.ld`
	Simulator Unhosted	`generic-m.ld`

ARM Simulator		
Processor name:	unspecified	
Processor options:	none	
Memory regions:	`ram`	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	Simulator Hosted	`generic-hosted.ld`
	Simulator Unhosted	`generic.ld`

ARM Simulator (VFP)		
Processor name:	unspecified	
Processor options:	none	
Memory regions:	`ram`	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	Simulator Hosted	`generic-vfp-hosted.ld`
	Simulator Unhosted	`generic-vfp.ld`

ARMulator (RDI)		
Processor name:	unspecified	
Processor options:	none	
Memory regions:	`ram`	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	RAM Hosted	`armulator-ram-hosted.ld`
	RAM Unhosted	`armulator-ram.ld`

Xilinx Zynq-7000		
Processor name:	Cortex-A9	
Processor options:	`-mcpu=cortex-a9`	
Memory regions:	`ram` (256MB DDR SDRAM), `rom` (64MB NOR Flash Memory)	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	RAM Hosted	`zynq7000-ram-hosted.ld`
	RAM Unhosted	`zynq7000-ram.ld`
	ROM Hosted	`zynq7000-rom-hosted.ld`
	ROM Unhosted	`zynq7000-rom.ld`

5.6. Interrupt Vector Tables

5.6.1. `__cs3_interrupt_vector_arm`

The ARM interrupt vector table (`__cs3_interrupt_vector_arm`) contents are:

Number	Name	Meaning
0	`__cs3_reset`	Reset entry point
1	`__cs3_isr_undef`	Undefined Instruction
2	`__cs3_isr_swi`	Software Interrupt/Supervisor Call
3	`__cs3_isr_pabort`	Prefetch Abort
4	`__cs3_isr_dabort`	Data Abort
5	`__cs3_isr_reserved`	
6	`__cs3_isr_irq`	External Interrupt (IRQ)
7	`__cs3_isr_fiq`	Fast Interrupt (FIQ)

5.6.2. `__cs3_interrupt_vector_micro`

The Microcontroller Profile interrupt vector table (`__cs3_interrupt_vector_micro`) contents are:

Number	Name	Meaning
0	`__cs3_stack`	Initial stack pointer
1	`__cs3_reset`	Reset entry point
2	`__cs3_isr_nmi`	Non Maskable Interrupt
3	`__cs3_isr_hard_fault`	Hardware fault
4	`__cs3_isr_mpu_fault`	MPU fault
5	`__cs3_isr_bus_fault`	Bus fault
6	`__cs3_isr_usage_fault`	Usage fault
7..10	`__cs3_isr_reserved_7..10`	Reserved for future use
11	`__cs3_isr_svcall`	System Vector Call
12	`__cs3_isr_debug`	Debug interrupt

Number	Name	Meaning
13	`__cs3_isr_reserved_13`	Reserved for future use
14	`__cs3_isr_pendsv`	
15	`__cs3_isr_systick`	System Ticker
16..47	`__cs3_isr_external_0..31`	External interrupt

Chapter 6
Sourcery G++ Debug Sprite

This chapter describes the use of the Sourcery G++ Debug Sprite for remote debugging. The Sprite allows you to debug programs running on a bare board without an operating system. This chapter includes information about the debugging devices and boards supported by the Sprite for ARM EABI.

Sourcery G++ Lite contains the Sourcery G++ Debug Sprite for ARM EABI. 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 6.3, "Invoking Sourcery G++ Debug Sprite" for information about the specific devices supported by this version of Sourcery G++ Lite.

Important

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

6.1. Probing for Debug Devices

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

```
> arm-none-eabi-sprite -i
```

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

```
CodeSourcery ARM Debug Sprite
    (Sourcery G++ Lite 2011.03-42)
armusb: [speed=<n:0-7>] ARMUSB (Stellaris) device
  armusb:///0B01000C - Stellaris Evaluation Board (0B01000C)
rdi: (rdi-library=<file>&rdi-config=<file>) RDI Device
  rdi:/// - RDI Device
```

This shows that ARMUSB and RDI devices are supported. The exact set of supported devices depends on your host system and the version of Sourcery G++ you have installed; refer to Section 6.3, "Invoking Sourcery G++ Debug Sprite" for complete information.

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

6.2. Debug Sprite Example

Start by compiling and linking a simple test program for your target board, following the instructions in Chapter 4, "Using Sourcery G++ from the Command Line". Use the `-g` option to tell the compiler to generate debugging information.

To build the `factorial` program to run on the ARMulator simulator, which can communicate with the Sprite via the RDI protocol, use:

```
> arm-none-eabi-gcc -g -Tarmulator-ram-hosted.ld main.c \
    -o factorial
```

Next start the debugger on your host system:

```
> arm-none-eabi-gdb factorial
```

The command for connecting GDB to the board depends on the debug device you are using; this is described in more detail in Section 6.3, “Invoking Sourcery G++ Debug Sprite”. If you are connecting via RDI, you must specify the full path to the RDI library file and configuration file for that library. Use quotes to escape the Sprite argument syntax from the shell. For example, use a command like this to connect to the ARMulator:

```
(gdb) target remote | arm-none-eabi-sprite \
  "rdi:///?rdi-library=library&rdi-config=config" armulator
```

The Sprite prints some status messages as it connects to your debug device and target board. If the connection is successful, you should see output similar to:

```
arm-none-eabi-sprite:Target reset
0x00008936 in ?? ()
(gdb)
```

Next, use GDB to load your program onto the target board.

```
(gdb) load
```

At this point you can use GDB to control the execution of your program as required. For example:

```
(gdb) break main
(gdb) continue
```

6.3. Invoking Sourcery G++ Debug Sprite

The Debug Sprite is invoked as follows:

```
> arm-none-eabi-sprite [options] device-url board-file
```

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

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

Most device URL schemes also follow the regular format:

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

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

`rdi` Use an RDI debugging device. Refer to Section 6.5, “Remote Debug Interface Devices”.

`flashpro` Use a FlashPro debugging device. Refer to Section 6.6, “Actel FlashPro Devices”.

`altera` Use an Altera FPGA. Refer to Section 6.7, “Altera 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 6.9, “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 6.10, “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.

6.4. Sourcery G++ Debug Sprite Options

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

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

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

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

`-l` *`[host]:port`*
: Specify the host address and port number to listen for a GDB connection. If this option is not given, the Debug Sprite communicates with GDB using stdin and stdout. If you start the Sprite from within GDB using the `target remote | arm-none-eabi-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.

6.5. Remote Debug Interface Devices

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

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

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

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

`rdi-config=`*`configfile`*
: Specify a file containing configuration information for *`library`*. The format of this file is specific to the RDI library you are using,

but tends to constitute a list of *key*=*value* pairs. Consult the documentation of your RDI library for details.

6.6. Actel FlashPro Devices

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

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

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

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

```
> arm-none-eabi-sprite -i flashpro:
```

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

6.6.1. Installing FlashPro Windows drivers

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

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

```
> arm-none-eabi-sprite -i
flashpro: [jtagclock=<n:93750-4000000>] FlashPro device
  flashpro://usb12345/ - FlashPro Device
  ...
```

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

6.7. Altera Devices

The Debug Sprite can be used to debug applications running on a Cortex-M1 core embedded in an Altera FPGA supporting the System-Level Debug (SLD) architecture. Currently, the Sprite supports the Cyclone III FPGA Starter board on Microsoft Windows hosts.

The Debug Sprite accepts two forms of the *device-url* for Altera devices. For the common case where you have only one Altera Cortex-M1 device configured, you can use simply:

```
altera://
```

The full form of the *device-url* is:

```
altera://usbX/hubY/nodeZ
```

where *X*, *Y*, and *Z* are non-negative integers. The SLD architecture forms a hierarchy; there may be multiple USB Blaster devices (numbered by *X*), multiple Altera FPGAs (numbered by *Y*) per USB Blaster, and multiple nodes (numbered by *Z*) per FPGA.

The Debug Sprite can autodetect connected Altera Cortex-M1 devices. Invoking the Sprite with the `-i` option, as described in Section 6.1, “Probing for Debug Devices”, displays the *device-url* for each detected device:

```
> arm-none-eabi-sprite -i
...
altera: Altera SLD Hub device
  altera://usb0/hub0/node1 - Altera Cortex-M Device
```

6.7.1. Setting Up the Altera Device

Follow these steps for initial installation and set up of the Altera device.

1. Install Quartus II Web Edition (or any equivalent), available from Altera.
2. Install drivers for USB Blaster, also available from Altera.
3. Install Sourcery G++ Lite for ARM EABI. See Chapter 2, “Installation and Configuration”.
4. Connect the board and the host computer with a USB cable.
5. Turn on the board.
6. Use Quartus II to download a `.sof` file including a Cortex-M1 core to the FPGA.
7. Use `arm-none-eabi-sprite -i` to verify that the Sprite can detect the installed Cortex-M1 core.

6.7.2. Hardware Breakpoints

The Cortex-M1 core only permits hardware breakpoints to be set in the first 512MB of its address space. Because both external SRAM and flash memory are located at higher addresses, you cannot set hardware breakpoints in these memory regions.

6.8. Debugging a Remote Board

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

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

```
> arm-none-eabi-sprite -l :10000 device-url board-file
```

starts the Sprite listening on port 10000.

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

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

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

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

6.9. Supported Board Files

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

Board	Config
Altera Cyclone III Cortex-M1	`cycloneiii-cm1`
ARMulator (RDI)	`armulator`
Xilinx Zynq-7000	`zynq7000`

6.10. Board File Syntax

The *board-file* can be a user-written XML file to describe a non-standard board. The Sourcery G++ Debug Sprite searches for board files in the `arm-none-eabi/lib/boards` directory in the installation. Refer to the files in that directory for examples.

The file's DTD is:

```
<!-- Board description files

     Copyright (c) 2007-2009 CodeSourcery, Inc.

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

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

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

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

<!ELEMENT initialize
```

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

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

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

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

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

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

The following elements are available:

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

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

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

- `banked-regs` The `banked-regs` property specifies that the CPU of the target board has banked registers for different processor modes (supervisor, IRQ, etc.).
- `has-vfp` The `has-vfp` property specifies that the CPU of the target board has VFP registers.
- `system-v6-m` The `system-v6-m` property specifies that the CPU of the target board has ARMv6-M architecture system registers.
- `system-v7-m` The `system-v7-m` property specifies that the CPU of the target board has ARMv7-M architecture system registers.
- `core-family` The `core-family` property specifies the ARM family of the target. The body of the `<property>` element may be one of `arm7`, `arm9`, `arm11`, and `cortex`.
- `system-clock` This property specifies the target clock frequency (in Hertz) after reset. It is used to configure flash programming algorithms.

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

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

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

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

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

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

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

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

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

Chapter 7
Next Steps with Sourcery G++

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

7.1. Sourcery G++ Knowledge Base

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

7.2. Example Programs

Sourcery G++ Lite includes some bundled example programs. You can find the source code for these examples in the `share/sourceryg++-arm-none-eabi-examples` directory of your Sourcery G++ installation.

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

7.3. Manuals for GNU Toolchain Components

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

When you install Sourcery G++ Lite, links to both the PDF and HTML versions of the manuals are created in the shortcuts folder you select. If you elected not to create shortcuts when installing Sourcery G++ Lite, the documentation can be found in the `share/doc/sourceryg++-arm-none-eabi/` subdirectory of your installation directory.

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

```
> cd $INSTALL/share/doc/sourceryg++-arm-none-eabi/man/man1
```

Then you can invoke `man` as:

```
> man ./arm-none-eabi-gcc.1
```

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

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

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

Appendix A
Sourcery G++ Lite Release Notes

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

A.1. Changes in Sourcery G++ Lite for ARM EABI

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

A.1.1. Changes in Sourcery G++ Lite 2011.03-42

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

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

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

Fixed-point arithmetic support. Experimental compiler support has been added for fixed-point arithmetic on ARM, as described in the draft ISO/IEC technical report TR 18037. Specialized instructions defined in recent architecture versions for performing saturating arithmetic, etc. are used when available, but are not a prerequisite for using the new language features. See Section 3.5, “Fixed-Point Arithmetic” for further details.

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

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

Internal compiler error with pointer casting. A compiler bug has been fixed that caused internal compiler errors when accessing double-word memory locations with casted pointers under ARM mode.

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

Internal compiler error under Thumb mode. A compiler bug has been fixed that caused internal compiler errors when generating Thumb code.

Xilinx Zynq-7000 board support. Sourcery G++ Lite now includes CS3 board support for the Xilinx Zynq-7000.

Debugging M-profile targets with third-party GDB stubs. A bug in GDB has been fixed that caused the error `Remote 'g' packet reply is too long` with ARMv6-M and ARMv7-M targets. The error was reported when connecting to some third-party GDB stubs, including SEGGER GDB Server and OpenOCD, but did not affect the Sourcery G++ Debug Sprite.

A.1.2. Changes in Sourcery G++ Lite 2011.03-13

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

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

- Smaller and faster code for compound conditionals.
- Removal of superfluous sign and zero extensions.
- Improved code for multiply-and-accumulate operations on ARM.
- Faster code when tuning for Cortex-M series processors.

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

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

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

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

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

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

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

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

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

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

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

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

CS3 interrupt handlers. CS3 now provides separate ISR functions for each core processor exception, rather than aliasing them all to a single interrupt handler. This provides more useful backtrace information in the debugger.

A.1.3. Changes in Sourcery G++ Lite 2010.09-51

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

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

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

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

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

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

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

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

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

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

Debugger warnings quieted. GDB no longer prints `RMT ERROR` diagnostics on connection to the Sourcery G++ Debug Sprite. In spite of the alarming appearance of the messages, they were not actually indicative of a serious problem.

A.1.4. Changes in Sourcery G++ Lite 2010.09-22

Changes to Sourcery G++ version numbering. Sourcery G++ product and Lite toolchains now uniformly use a version numbering scheme of the form 2011.03-42. The major and minor parts of the version number, in this case 2011.03, identify the release branch, while the final component is a build number within the branch. There are also new preprocessor macros defined by the compiler for the version number components so that you may conditionalize code for Sourcery G++ or particular Sourcery G++ versions. Details are available in the Sourcery G++ Knowledge Base[1].

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

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

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

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

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

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

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

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

- More efficient assignment for structures containing bitfields.
- Better code for initializing C++ arrays with explicit element initializers.
- Improved logic for eliminating/combining redundant comparisons in code with nested conditionals.
- Better selection of loop variables, resulting in fewer temporaries and more efficient register usage.
- More optimization of references to globals in position-independent code.
- Various Thumb code generation improvements.

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

- Better code when constant addresses are used as arguments to inline assembly statements.
- Better code for copying small constant strings.
- Improved tuning for Cortex-M4 processors.
- Cortex-A9 specific tuning for VFP and NEON instructions.
- Use of more NEON features.

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

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

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

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

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

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

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

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

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

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

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

Additional alignment in CS3-defined linker scripts. Sourcery G++ now ensures 8-byte alignment at additional points in CS3-defined linker scripts. Previously, placing a symbol in certain sections broke the initialization of the `.data` and/or `.bss` sections.

Newlib update. The Newlib package has been updated to version 1.18.0, with additions from the community CVS trunk as of 2010-08-12. This update provides additional wide-character functions, along with other bug fixes and enhancements.

`malloc` fix. A bug that sometimes caused `free` to dereference an invalid address has been fixed. The bug was caused by incorrect handling within `malloc` of calls to `sbrk` from outside of `malloc`.

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

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

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

Debug Sprite abnormal termination bug fix. The Sourcery G++ Debug Sprite no longer terminates abnormally if GDB is killed while the target is waiting for semihosted I/O to complete. The bug was only triggered when running GDB on a Windows host.

Semihosting support for `gettimeofday`. The Sourcery G++ Debug Sprite now provides a semihosted implementation of the `gettimeofday` C library function.

A.1.5. Changes in Older Releases

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

Appendix B
Sourcery G++ Lite Licenses

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

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

The table below lists the major components of Sourcery G++ Lite for ARM EABI and the license terms which apply to each of these components.

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

Component	License
GNU Compiler Collection	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
GNU Binary Utilities	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
GNU Debugger	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
Sourcery G++ Debug Sprite for ARM	CodeSourcery License
CodeSourcery Common Startup Code Sequence	CodeSourcery License
Newlib C Library	BSD License. For the text of the license and a complete list of copyright holders, see Section B.3.2, “Newlib”.
GNU Make	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
GNU Core Utilities	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

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

Important

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

Sourcery G++ Lite may include some third party example programs and libraries in the `share/sourceryg++-arm-none-eabi-examples` subdirectory. These examples are not covered by the Sourcery G++ Software License Agreement. To the extent permitted by law, these examples are provided by CodeSourcery as is with no warranty of any kind, including implied warranties of merchantability or fitness for a particular purpose. Your use of each example is governed by the license notice (if any) it contains.

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

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

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

3. **Definitions.**

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

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

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

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

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

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

    5.1.

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

provides an overview of which license applies to different components, and, for components subject to the Eclipse Public License, contains information on how to obtain the source code. Definitive licensing information for each “free software” or “open source” component is available in the relevant source file.

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

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

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

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

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

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

13. **Limitation of Liability.** INDEPENDENT OF THE FORGOING PROVISIONS, IN NO EVENT AND UNDER NO LEGAL THEORY, INCLUDING WITHOUT LIMITATION,

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

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

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

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

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

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

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

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

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

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

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

B.3. Attribution

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

B.3.1. Android Open Source Project

```
/*
 * Copyright (C) 2008 The Android Open Source Project
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *  * Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *  * Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
```

```
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
 * COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
 * OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
 * AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */
```

B.3.2. Newlib

```
The newlib subdirectory is a collection of software from several sources.

Each file may have its own copyright/license that is embedded in the source
file.  Unless otherwise noted in the body of the source file(s), the following copyright
notices will apply to the contents of the newlib subdirectory:

(1) Red Hat Incorporated

Copyright (c) 1994-2007  Red Hat, Inc. All rights reserved.

This copyrighted material is made available to anyone wishing to use,
modify, copy, or redistribute it subject to the terms and conditions
of the BSD License.   This program is distributed in the hope that
it will be useful, but WITHOUT ANY WARRANTY expressed or implied,
including the implied warranties of MERCHANTABILITY or FITNESS FOR
A PARTICULAR PURPOSE.  A copy of this license is available at
http://www.opensource.org/licenses. Any Red Hat trademarks that are
incorporated in the source code or documentation are not subject to
the BSD License and may only be used or replicated with the express
permission of Red Hat, Inc.

(2) University of California, Berkeley

Copyright (c) 1981-2000 The Regents of the University of California.
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.
    * Neither the name of the University nor the names of its contributors
      may be used to endorse or promote products derived from this software
      without specific prior written permission.

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.

(3) David M. Gay (AT&T 1991, Lucent 1998)

The author of this software is David M. Gay.

Copyright (c) 1991 by AT&T.

Permission to use, copy, modify, and distribute this software for any
```

```
purpose without fee is hereby granted, provided that this entire notice
is included in all copies of any software which is or includes a copy
or modification of this software and in all copies of the supporting
documentation for such software.

THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED
WARRANTY.  IN PARTICULAR, NEITHER THE AUTHOR NOR AT&T MAKES ANY
REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY
OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

-------------------------------------------------------------------

The author of this software is David M. Gay.

Copyright (C) 1998-2001 by Lucent Technologies
All Rights Reserved

Permission to use, copy, modify, and distribute this software and
its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that the copyright notice and this
permission notice and warranty disclaimer appear in supporting
documentation, and that the name of Lucent or any of its entities
not be used in advertising or publicity pertaining to
distribution of the software without specific, written prior
permission.

LUCENT DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE,
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS.
IN NO EVENT SHALL LUCENT OR ANY OF ITS ENTITIES BE LIABLE FOR ANY
SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER
IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.


(4) Advanced Micro Devices

Copyright 1989, 1990 Advanced Micro Devices, Inc.

This software is the property of Advanced Micro Devices, Inc  (AMD)  which
specifically  grants the user the right to modify, use and distribute this
software provided this notice is not removed or altered.  All other rights
are reserved by AMD.

AMD MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS
SOFTWARE.  IN NO EVENT SHALL AMD BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL
DAMAGES IN CONNECTION WITH OR ARISING FROM THE FURNISHING, PERFORMANCE, OR
USE OF THIS SOFTWARE.

So that all may benefit from your experience, please report  any  problems
or  suggestions about this software to the 29K Technical Support Center at
800-29-29-AMD (800-292-9263) in the USA, or 0800-89-1131  in  the  UK,  or
0031-11-1129 in Japan, toll free.  The direct dial number is 512-462-4118.

Advanced Micro Devices, Inc.
29K Support Products
Mail Stop 573
5900 E. Ben White Blvd.
Austin, TX 78741
800-292-9263

(5) C.W. Sandmann

Copyright (C) 1993 C.W. Sandmann

This file may be freely distributed as long as the author's name remains.

(6) Eric Backus

(C) Copyright 1992 Eric Backus
```

```
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE AUTHOR 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.

(12) SuperH, Inc.

Copyright 2002 SuperH, Inc. All rights reserved

This software is the property of SuperH, Inc (SuperH) which specifically
grants the user the right to modify, use and distribute this software
provided this notice is not removed or altered.  All other rights are
reserved by SuperH.

SUPERH MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO
THIS SOFTWARE.  IN NO EVENT SHALL SUPERH BE LIABLE FOR INDIRECT, SPECIAL,
INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING FROM
THE FURNISHING, PERFORMANCE, OR USE OF THIS SOFTWARE.

So that all may benefit from your experience, please report any problems
or suggestions about this software to the SuperH Support Center via
e-mail at softwaresupport@superh.com .

SuperH, Inc.
405 River Oaks Parkway
San Jose
CA 95134
USA

(13) Royal Institute of Technology

Copyright (c) 1999 Kungliga Tekniska Högskolan
(Royal Institute of Technology, Stockholm, Sweden).
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. 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.

3. Neither the name of KTH nor the names of its contributors may be
   used to endorse or promote products derived from this software without
   specific prior written permission.

THIS SOFTWARE IS PROVIDED BY KTH AND ITS 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 KTH OR ITS 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.

(14) Alexey Zelkin

Copyright (c) 2000, 2001 Alexey Zelkin <phantom@FreeBSD.org>
All rights reserved.

Redistribution and use in source and binary forms, with or without
```

```
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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 AUTHOR 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 AUTHOR 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.

(15) Andrey A. Chernov

Copyright (C) 1997 by Andrey A. Chernov, Moscow, Russia.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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 AUTHOR ``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 REGENTS 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.

(16) FreeBSD

Copyright (c) 1997-2002 FreeBSD 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:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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 AUTHOR 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 AUTHOR 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.
```

(17) S. L. Moshier

Author: S. L. Moshier.

Copyright (c) 1984,2000 S.L. Moshier

Permission to use, copy, modify, and distribute this software for any
purpose without fee is hereby granted, provided that this entire notice
is included in all copies of any software which is or includes a copy
or modification of this software and in all copies of the supporting
documentation for such software.

THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED
WARRANTY. IN PARTICULAR, THE AUTHOR MAKES NO REPRESENTATION
OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS
SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

(18) Citrus Project

Copyright (c)1999 Citrus 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:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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 AUTHOR 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 AUTHOR 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.

(19) Todd C. Miller

Copyright (c) 1998 Todd C. Miller <Todd.Miller@courtesan.com>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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.
3. The name of the author may not be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED ``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 AUTHOR 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.

(20) DJ Delorie (i386)
Copyright (C) 1991 DJ Delorie

All rights reserved.

Redistribution and use in source and binary forms is permitted
provided that the above copyright notice and following paragraph are
duplicated in all such forms.

This file is distributed WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

(21) Free Software Foundation LGPL License (*-linux* targets only)

Copyright (C) 1990-1999, 2000, 2001   Free Software Foundation, Inc.
This file is part of the GNU C Library.
Contributed by Mark Kettenis <kettenis@phys.uva.nl>, 1997.

The GNU C Library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

The GNU C Library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with the GNU C Library; if not, write to the Free
Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
02110-1301 USA.

(22) Xavier Leroy LGPL License (i[3456]86-*-linux* targets only)

Copyright (C) 1996 Xavier Leroy (Xavier.Leroy@inria.fr)

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU Library General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU Library General Public License for more details.

(23) Intel (i960)

Copyright (c) 1993 Intel Corporation

Intel hereby grants you permission to copy, modify, and distribute this
software and its documentation. Intel grants this permission provided
that the above copyright notice appears in all copies and that both the
copyright notice and this permission notice appear in supporting
documentation. In addition, Intel grants this permission provided that
you prominently mark as "not part of the original" any modifications
made to this software or documentation, and that the name of Intel
Corporation not be used in advertising or publicity pertaining to
distribution of the software or the documentation without specific,
written prior permission.

Intel Corporation provides this AS IS, WITHOUT ANY WARRANTY, EXPRESS OR
IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY
OR FITNESS FOR A PARTICULAR PURPOSE. Intel makes no guarantee or
representations regarding the use of, or the results of the use of,
the software and documentation in terms of correctness, accuracy,
reliability, currentness, or otherwise; and you rely on the software,
documentation and results solely at your own risk.

IN NO EVENT SHALL INTEL BE LIABLE FOR ANY LOSS OF USE, LOSS OF BUSINESS,
LOSS OF PROFITS, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES
OF ANY KIND. IN NO EVENT SHALL INTEL'S TOTAL LIABILITY EXCEED THE SUM
PAID TO INTEL FOR THE PRODUCT LICENSED HEREUNDER.

```
(24) Hewlett-Packard  (hppa targets only)

(c) Copyright 1986 HEWLETT-PACKARD COMPANY

To anyone who acknowledges that this file is provided "AS IS"
without any express or implied warranty:
    permission to use, copy, modify, and distribute this file
for any purpose is hereby granted without fee, provided that
the above copyright notice and this notice appears in all
copies, and that the name of Hewlett-Packard Company not be
used in advertising or publicity pertaining to distribution
of the software without specific, written prior permission.
Hewlett-Packard Company makes no representations about the
suitability of this software for any purpose.

(25) Henry Spencer (only *-linux targets)

Copyright 1992, 1993, 1994 Henry Spencer.  All rights reserved.
This software is not subject to any license of the American Telephone
and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on
any computer system, and to alter it and redistribute it, subject
to the following restrictions:

1. The author is not responsible for the consequences of use of this
   software, no matter how awful, even if they arise from flaws in it.

2. The origin of this software must not be misrepresented, either by
   explicit claim or by omission.  Since few users ever read sources,
   credits must appear in the documentation.

3. Altered versions must be plainly marked as such, and must not be
   misrepresented as being the original software.  Since few users
   ever read sources, credits must appear in the documentation.

4. This notice may not be removed or altered.

(26) Mike Barcroft

Copyright (c) 2001 Mike Barcroft <mike@FreeBSD.org>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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 AUTHOR 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 AUTHOR 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.

(27) Konstantin Chuguev (--enable-newlib-iconv)

Copyright (c) 1999, 2000
   Konstantin Chuguev.  All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
```

```
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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 AUTHOR 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 AUTHOR 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.

   iconv (Charset Conversion Library) v2.0

(28) Artem Bityuckiy (--enable-newlib-iconv)

Copyright (c) 2003, Artem B. Bityuckiy, SoftMine Corporation.
Rights transferred to Franklin Electronic Publishers.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. 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 AUTHOR 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 AUTHOR 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.

(29) IBM, Sony, Toshiba (only spu-* targets)

  (C) Copyright 2001,2006,
  International Business Machines Corporation,
  Sony Computer Entertainment, Incorporated,
  Toshiba Corporation,

  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.
    * Neither the names of the copyright holders nor the names of their
      contributors may be used to endorse or promote products derived from this
      software without specific prior written permission.

  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.

(30) - Alex Tatmanjants (targets using libc/posix)

Copyright (c) 1995 Alex Tatmanjants <alex@elvisti.kiev.ua>
at Electronni Visti IA, Kiev, Ukraine.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. 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 AUTHOR ``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 AUTHOR 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.

(31) - M. Warner Losh (targets using libc/posix)

Copyright (c) 1998, M. Warner Losh <imp@freebsd.org>
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. 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 AUTHOR 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 AUTHOR 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.

(32) - Andrey A. Chernov (targets using libc/posix)

Copyright (C) 1996 by Andrey A. Chernov, Moscow, Russia.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. 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 AUTHOR ``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 REGENTS 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.

(33) - Daniel Eischen (targets using libc/posix)

  Copyright (c) 2001 Daniel Eischen <deischen@FreeBSD.org>.
  All rights reserved.

  Redistribution and use in source and binary forms, with or without
  modification, are permitted provided that the following conditions
  are met:
  1. Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
  2. 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 AUTHOR 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 REGENTS 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.


(34) - Jon Beniston (only lm32-* targets)

 Contributed by Jon Beniston <jon@beniston.com>

 Redistribution and use in source and binary forms, with or without
 modification, are permitted provided that the following conditions
 are met:
 1. Redistributions of source code must retain the above copyright
 notice, this list of conditions and the following disclaimer.
 2. 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 AUTHOR 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 AUTHOR 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.


(35) - ARM Ltd (arm and thumb variant targets only)

 Copyright (c) 2009 ARM Ltd
 All rights reserved.
```

```
 Redistribution and use in source and binary forms, with or without
 modification, are permitted provided that the following conditions
 are met:
 1. Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
 2. 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.
 3. The name of the company may not be used to endorse or promote
    products derived from this software without specific prior written
    permission.

 THIS SOFTWARE IS PROVIDED BY ARM LTD ``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 ARM LTD 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.


(36) - CodeSourcery, Inc.

Copyright (c) 2009 CodeSourcery, Inc.
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.
    * Neither the name of CodeSourcery nor the
      names of its contributors may be used to endorse or promote products
      derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY CODESOURCERY, INC. ``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 CODESOURCERY 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.

(37) MIPS Technologies, Inc
/*
 * Copyright (c) 2009 MIPS Technologies, Inc.
 *
 * 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.
 *      * Neither the name of MIPS Technologies Inc. nor the names of its
 *        contributors may be used to endorse or promote products derived
 *        from this software without specific prior written permission.
 *
```