Sourcery G++ Lite

ARM SymbianOS

Sourcery G++ Lite 2009q1-162

Getting Started

Sourcery G++ Lite: ARM SymbianOS: Sourcery G++ Lite 2009q1-162: Getting Started

CodeSourcery, Inc.

Abstract

This guide explains how to install and build applications with Sourcery G++ Lite, CodeSourcery's customized, validated, and supported 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

This preface introduces *Getting Started With Sourcery G++ Lite*. It explains the structure of this guide and lists other sources of information that relate to Sourcery G++ Lite.

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, *Sourcery G++ Lite Licenses*	This chapter provides information about the software licenses that apply to Sourcery G++ Lite. Read this chapter to understand your legal rights and obligations as a user of Sourcery G++ Lite.
Chapter 2, *Sourcery G++ Subscriptions*	This chapter provides information about Sourcery G++ subscriptions. CodeSourcery customers with Sourcery G++ subscriptions receive comprehensive support for Sourcery G++. Read this chapter to find out how to obtain and use a Sourcery G++ subscription.
Chapter 3, *Sourcery G++ Lite for ARM SymbianOS*	This chapter provides information about this release of Sourcery G++ Lite including any special installation instructions, recent improvements, or other similar information. You should read this chapter before building applications with Sourcery G++ Lite.
Chapter 4, *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 5, *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 6, *Next Steps with Sourcery G++*	This chapter describes where you can find additional documentation and information about using Sourcery G++ Lite and its components.

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
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 chapter explains what licenses apply to your use of Sourcery G++ Lite. You should read this chapter to understand your legal rights and obligations as a user of Sourcery G++ Lite.

1.1. Licenses for Sourcery G++ Lite Components

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

Some free or open-source components provide documentation or other files under terms different from those shown below. For definitive information about the license that applies to each component, consult the source package corresponding to this release of 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 Binary Utilities	GNU General Public License 3.0 [1]
GNU Compiler Collection	GNU General Public License 3.0 [2]
GNU Make	GNU General Public License 2.0 [3]
GNU Core Utilities	GNU General Public License 2.0 [4]

The CodeSourcery License is available in Section 1.2, “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.

1.2. Sourcery G++™ Software License Agreement

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

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

3. **Definitions.**

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

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

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.

4. **License Grant to Proprietary Components of the Software.** You are granted a non-exclusive, royalty-free license to install and use the CodeSourcery Proprietary Components of the Software, transmit the CodeSourcery Proprietary Components over an internal computer network, and/or copy the CodeSourcery Proprietary Components for Your internal use only.

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; or (iii) reverse engineer, decompile, or disassemble the CodeSourcery Proprietary Components of the Software, except to the extent this restriction is expressly prohibited by applicable law.

6. **"Free Software" or "Open Source" License to Certain Components of the Software.** This Agreement does not limit Your rights under, or grant You rights that supersede, the license terms of any Open Source Software Component delivered to You by CodeSourcery. Sourcery G++ includes components provided under various different licenses. The *Getting Started Guide* provides an overview of which license applies to different components. Definitive licensing information for each "free software" or "open source" component is available in the relevant source file.

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

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

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

10. **Ownership.** CodeSourcery owns and/or has licensed the CodeSourcery Proprietary Components of the Software and all intellectual property rights embodied therein, including copyrights and valuable trade secrets embodied in its design and coding methodology. The CodeSourcery Proprietary Components of the Software are protected by United States copyright laws and international treaty provisions. CodeSourcery also owns all rights, title and interest in and with respect to its trade names, domain names, trade dress, logos, trademarks, service marks, and

other similar rights or interests in intellectual property. This Agreement provides You only a limited use license, and no ownership of any intellectual property.

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

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

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

Chapter 2
Sourcery G++ Subscriptions

CodeSourcery provides support contracts for Sourcery G++. This chapter describes these contracts and explains how CodeSourcery customers can access their support accounts.

2.1. About Sourcery G++ Subscriptions

CodeSourcery offers Sourcery G++ subscriptions. Professional Edition subscriptions provide unlimited support, with no per-incident fees. CodeSourcery's support covers questions about installing and using Sourcery G++, the C and C++ programming languages, and all other topics relating to Sourcery G++. CodeSourcery provides updated versions of Sourcery G++ to resolve critical problems. Personal Edition subscriptions do not include support, but do include free upgrades as long as the subscription remains active.

CodeSourcery's support is provided by the same engineers who build Sourcery G++. A Sourcery G++ subscription is like having a team of compiler engineers and programming language experts available as consultants!

Subscription editions of Sourcery G++ also include many additional features not included in the free Lite editions:

- **Sourcery G++ IDE.** The Sourcery G++ IDE, based on Eclipse, provides a fully visual environment for developing applications, including an automated project builder, syntax-highlighting editor, and a graphical debugging interface. The debugger provides features especially useful to embedded systems programmers, including the ability to step through code at both the source and assembly level, view registers, and examine stack traces. CodeSourcery's enhancements to Eclipse include improved support for hardware debugging via JTAG or ICE units and complete integration with the rest of Sourcery G++.

- **Debug Sprites.** Sourcery G++ Debug Sprites provide hardware debugging support using JTAG and ICE devices. On some systems, Sourcery G++ Sprites can automatically program flash memory and display control registers. And the board initialization performed by each Sprite can be customized with simple XML-based configuration files to insert delays and write to particular memory addresses. Debug Sprites included in Lite editions of Sourcery G++ include only a subset of the functionality of the Sprites in the subscription editions.

- **QEMU Instruction Set Simulator.** The QEMU instruction set simulator can be used to run — and debug — programs even without target hardware. Most bare-metal configurations of Sourcery G++ include QEMU and linker scripts targeting the simulator. Configurations of Sourcery G++ for GNU/Linux targets include a user-space QEMU emulator that runs on Linux hosts.

- **Sysroot Utilities.** Subscription editions of Sourcery G++ include a set of sysroot utilities for GNU/Linux targets. These utilities simplify use of the Sourcery G++ dynamic linker and shared libraries on the target and also support remote debugging with **gdbserver**.

- **CS3.** CS3 provides a uniform, cross-platform approach to board initialization and interrupt handling on ARM EABI, ColdFire ELF, fido ELF, and Stellaris EABI platforms.

- **GNU/Linux Prelinker.** For select GNU/Linux target systems, Sourcery G++ includes the GNU/Linux prelinker. The prelinker is a postprocessor for GNU/Linux applications which can dramatically reduce application launch time. CodeSourcery has modified the prelinker to operate on non-GNU/Linux host systems, including Microsoft Windows.

- **Library Reduction Utility.** Sourcery G++ also includes a Library Reduction Utility for GNU/Linux targets. This utility allows the GNU C Library to be relinked to include only those functions used by a given collection of binaries.

- **Additional Libraries.** For some platforms, additional run-time libraries optimized for particular CPUs are available. Pre-built binary versions of the libraries with debug information are also available to subscribers.

If you would like more information about Sourcery G++ subscriptions, including a price quote or information about evaluating Sourcery G++, please send email to <sales@codesourcery.com>.

2.2. Accessing your Sourcery G++ Subscription Account

If you have a Sourcery G++ subscription, you may access your account by visiting the Sourcery G++ Portal[1]. If you have a support account, but are unable to log in, send email to <support@codesourcery.com>.

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

Chapter 3
Sourcery G++ Lite for ARM SymbianOS

This chapter contains information about using Sourcery G++ Lite on your target system. This chapter also contains information about changes in this release of Sourcery G++ Lite. You should read this chapter to learn how to best use Sourcery G++ Lite on your target system.

3.1. Library Configurations

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

Sourcery G++ Lite includes linker scripts as well as runtime libraries for each multilib. You can find these files in multilib-specific subdirectories of the `arm-none-symbianelf/lib` directory of your Sourcery G++ install.

3.1.1. Included Libraries

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

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

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

3.1.2. Library Selection

A given multilib may be compatible with additional processors and build options beyond those listed above. However, even if a particular set of command-line options produces code compatible with one of the provided multilibs, those options may not be sufficient to identify the intended library to the linker. For example, on some targets, specifying only a processor option on the command line may imply architecture features or floating-point support for compilation, but not for library selection. The details of the mapping from command-line options to multilibs are target-specific and quite complex. Therefore, it is recommended that your link command line include exactly the options listed in the tables above for your intended target multilib. In some cases, you may need to supply different options for linking than for compilation.

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

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

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

3.2. Using Sourcery G++ Lite for ARM SymbianOS

3.2.1. SymbianOS Runtime Libraries

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

3.2.2. 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.

3.2.3. Half-Precision Floating Point

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

3.2.3.1. Representations

ARM supports two incompatible representations for half-precision floating-point values. You must choose one of the representations and use it consistently in your program. The linker gives an error if objects compiled or assembled with different half-precision float attributes are combined in the same executable.

Compiling or assembling with `-mfp16-format=ieee` selects the representation defined in the *IEEE 754-2008* standard, with 1 sign bit, 5 exponent bits, and 10 significand bits (11 bits of significand precision, approximately 3 decimal digits). This format is capable of representing normalized values in the range of 2^{-14} to 65504. It includes support for infinities and NaNs, following the usual IEEE 754 rules.

ARM also supports an alternative half-precision representation, which you can select with `-mfp16-format=alternative`. This format does not include support for infinities and NaNs. Instead, the range of exponent values is extended, so that this format can represent normalized values in the range of 2^{-14} to 131008.

The default for this option is `-mfp16-format=none`, which disables support for half-precision floats.

3.2.3.2. C and C++ Usage

When you compile with `-mfp16-format=ieee` or `-mfp16-format=alternative`, GCC defines the `__fp16` data type to represent half-precision float values. Objects of this type have a size of 2 bytes and a natural alignment of 2 bytes.

The `__fp16` type is a storage format only. For purposes of arithmetic and other operations, `__fp16` values are automatically promoted to `float`. In addition, you cannot declare a function with a return value or parameters of type `__fp16`.

Note that conversions from `double` to `__fp16` involve an intermediate conversion to `float`. Because of rounding, this can sometimes produce a different result than a direct conversion.

3.2.3.3. Hardware and Library Support

ARM provides hardware support for conversions between `__fp16` and `float` values as an extension to VFP and NEON (Advanced SIMD). GCC generates code using the instructions provided by this extension if you compile with the options `-mfpu=neon-fp16 -mfloat-abi=softfp`, in addition to the `-mfp16-format` option to select a half-precision format.

In other cases, conversions between `__fp16` and `float` values are implemented as library calls.

3.2.4. 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 (10 October 2008)
- BPABI - ARM IHI 0037B (10 October 2008)
- EHABI - ARM IHI 0038A (10 October 2008)
- CLIBABI - ARM IHI 0039A (10 October 2008)
- AADWARF - ARM IHI 0040A (10 October 2008)
- CPPABI - ARM IHI 0041B (10 October 2008)
- AAPCS - ARM IHI 0042C (10 October 2008)
- RTABI - ARM IHI 0043B (10 October 2008)
- AAELF - ARM IHI 0044C (10 October 2008)
- ABI Addenda - ARM IHI 0045B (10 October 2008)

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

3.3. Sourcery G++ Lite Release Notes

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

3.3.1. Changes in Sourcery G++ Lite 2009q1-162

Incorrect placement of linker-generated functions. A bug that caused some linker-generated functions (including stubs to support interworking from ARM mode to Thumb mode and stubs to avoid processor errata) to be placed in data sections has been fixed.

New option for automatically generating IT blocks. The assembler now allows use of conditional Thumb-2 instructions without requiring explicit IT instructions. Use the `-mauto-it` command-line option to enable this automatic generation of IT instructions.

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

Reduced compilation time. Compilation and build times when using Sourcery G++ Lite are now slightly faster. This performance improvement is the result of building the compilers and other host tools with a recent version of Sourcery G++, rather than an older GCC version.

Assembler bug fix. A bug in the assembler that caused duplicate and missing mapping symbols has been fixed. The bug caused incorrect **objdump** output and incorrect byte-swapping for BE8 configurations.

Stack backtracing and C++ exception handling. Improvements have been made to the linker in support of C++ runtime exception handling and stack backtracing. A problem that caused crashes during the backtrace of C routines that were not compiled with the `-fexceptions` option has been fixed. In addition, the linker generates more compact stack unwinding tables which can lead to smaller executables.

Assembler floating-point format. The assembler now defaults to VFP format for floating-point numbers. It previously defaulted to the legacy FPA format if no `-mcpu` or `-march` option was specified, or if a CPU with no floating-point unit was specified. This bug resulted in incorrect behavior of the `.double` and `.dcb.d` directives.

Incorrect linker-generated functions. A bug that caused some linker-generated functions (such as stubs to support interworking from ARM mode to Thumb mode) to contain only `nop` instructions instead of correct code sequences has been fixed.

Assembler diagnostics for invalid instructions. The assembler now issues diagnostics for invalid `ADR` and `ADRL` instructions. Formerly, these invalid instructions were silently mis-assembled. This assembler bug did not affect correct code.

Disassembler bug fix. A bug has been fixed that caused incorrect disassembly of some object files with multiple sections whose symbol tables included symbols in the middle of functions. These typically resulted from hand-written assembly.

Linker crash with very large applications. A linker bug that caused a crash when linking very large applications with the `--fix-cortex-a8` command-line option has been fixed.

arm-none-symbianelf-objcopy bug fix. A bug has been fixed that caused **arm-none-symbianelf-objcopy** to issue an error when generating output in the Intel HEX format and using `--change-section-lma` to change section addresses.

Linker script search path. The bug in the linker has been fixed that caused it not to follow its documented behavior for searching for linker scripts named with the `-T` option. Now scripts are looked up first in the current directory, then in library directories specified with `-L` command-line options, and finally in the default system linker script directory.

Cortex-A8 erratum workaround enabled for ARMv7-A. The workaround for the erratum in Cortex-A8 processors mentioned below is now enabled by default if you are targeting the ARMv7-A architecture profile. The workaround can be disabled by passing the `--no-fix-cortex-a8` option to the linker.

Internal compiler error when optimizing. A bug has been fixed that caused `internal compiler error: in build2_stat` when compiling.

Erratum workaround for Cortex-A8 processors. The linker now implements a workaround for an erratum in Cortex-A8 processors. If you are targeting an affected part and wish to use the workaround, pass the `--fix-cortex-a8` option to the linker. Please contact ARM for further details of the erratum.

Maximum code alignment increased. The maximum allowed code alignment has been increased from 32 to 64 bytes. This change affects the `.p2align` and `.align` directives in GAS and the `-falign-functions` GCC option.

Corruption of block-scope variables. A compiler optimization bug that sometimes caused corruption of stack-allocated variables has been fixed. The bug affected variables declared in a local block scope in functions containing multiple non-overlapping lexical block scopes, a technique commonly used by programmers to reduce stack frame size. In some rare cases, other optimizations performed by the compiler were ignoring the local extent of such block-scope variables.

3.3.2. Changes in Sourcery G++ Lite 2009q1-115

Incorrect code when using `-falign-labels`. A bug that caused the compiler to generate incorrect code for `switch` statements when the `-falign-labels` option is used has been fixed.

Loop optimization improvements. A new option, `-fpromote-loop-indices`, has been added to the compiler. Specifying this option enables an optimization that improves the performance of loops with index variables of integer types narrower than the target machine word size, such as `char` or `short`. This optimization also applies to `int` on 64-bit targets.

`DMB`, `DSB`, and `ISB` instructions on ARMv6-M. The assembler now accepts the `DMB`, `DSB`, and `ISB` instructions on ARMv6-M CPUs, including Cortex-M0 and Cortex-M1. These instructions were incorrectly rejected on these CPUs in previous releases.

Extraneous linker error messages. A linker bug that caused extraneous error messages of the form `Dwarf Error: Offset (507) greater than or equal to .debug_str size (421).` has been corrected. This bug did not affect the correctness of output binaries.

Assembler marking of data. Data generated using the assembler directives `.ascii`, `.asciz`, `.dc.d`, `.dc.s`, `.dc.x`, `.dcb`, `.dcb.b`, `.dcb.d`, `.dcb.l`, `.dcb.s`, `.dcb.w`, `.dcb.x`, `.ds`, `.ds.b`, `.ds.d`, `.ds.l`, `.ds.p`, `.ds.s`, `.ds.w`, `.ds.x`, `.double`, `.fill`, `.float`, `.incbin`, `.single`, `.space`, `.skip`, `.string`, `.string8`, `.string16`, `.string32`, `.string64`, and `.zero` is now correctly marked by the assembler as data rather than code. This fixes incorrect byte-swapping of such data when linking for BE8 configurations.

Improved vectorization. Automatic vectorization for NEON now uses the fused multiply-add (`VMLA`) and fused multiply-subtract (`VMLS`) instructions. These fused instructions are faster than the equivalent two-instruction sequence consisting of a multiply followed by an add or subtract.

Out-of-bounds accesses to stack arrays. A bug has been fixed that caused internal compiler errors when some code involving out-of-bounds accesses to stack-allocated arrays was compiled with the `-mthumb` option. Such code is not valid C; although it is now accepted by the compiler and no diagnostic is issued, it has undefined behavior if executed.

GCC version 4.3.3. Sourcery G++ Lite for ARM SymbianOS is now based on GCC version 4.3.3. This is a bug fix update to GCC. For more information about changes from GCC version 4.3.2 that was included in previous releases, see `http://gcc.gnu.org/gcc-4.3/changes.html`.

Improved NOP generation for Thumb-2 cores. The assembler now generates Thumb-2/ARMv6K architectural NOP instructions when alignment padding is required in code sections.

Internal compiler error with `-O3` or `-fpredictive-commoning`. A bug has been fixed that caused internal compiler errors when compiling some code with `-O3` or `-fpredictive-commoning`.

C++ named operators bug fix. A bug has been fixed that caused the compiler to crash in some cases when the C++ operators `and_eq`, `bitand`, `bitor`, `compl`, `not_eq`, `or_eq` and `xor_eq` were used in contexts where the preprocessor converts their names to strings.

Debug information for anonymous structure types. A GCC bug in the generation of debug information for anonymous structure types in C++ code has been fixed. The bug caused printing the type information for such structures in the debugger (via the **ptype** command) to fail with an error message.

Linker errors on non-ELF input. A bug has been fixed that caused internal errors from the linker when linking non-ELF input files (with the `-b` or `--format` linker options).

Undefined weak references in shared libraries. A linker bug has been fixed affecting calls from Thumb code in shared libraries to functions that are undefined weak references when the shared library is linked. Such calls executed as nops whether or not the functions were defined at run time.

Improved code generation. The compiler has been improved to generate better code for an integer multiplication whose result feeds into an addition.

Installer fails during upgrade. The Sourcery G++ installer for Microsoft Windows hosts could fail during an upgrade while waiting for the previous version to be uninstalled. This bug has been fixed.

Performance improvements. Tuning parameters for ARM code generation have been adjusted to improve performance of the generated code.

Uninstaller removed by upgrade. The uninstaller could be incorrectly deleted during an upgrade on Microsoft Windows hosts. This bug has been fixed.

`CMP` Thumb-2 instruction. The assembler no longer issues an error about `CMP` instructions in which the second argument is the stack pointer (`r13`), as these are valid instructions. However, use of the stack pointer in this context is deprecated in the current ARM architecture specification and the assembler now warns about the deprecated use.

Thumb half-precision floating point bug fix. A compiler bug has been fixed that formerly caused incorrect code to be generated in Thumb mode for functions using half-precision floating-point constants. The bug did not affect Thumb-2 code.

Improved code generation. The compiler has been improved to generate better code for integer multiplication by certain constants.

Thumb-2 `switch` code generation bug fix. A bug has been fixed that caused incorrect Thumb-2 code to be generated for some `switch` statements.

Internal compiler errors when optimizing. A defect that occasionally caused internal compiler errors when partial redundancy elimination (PRE) optimization was enabled has been corrected.

Install directory pathnames. Bugs in the install and uninstall scripts for Linux hosts that caused errors or incorrect behavior when the Sourcery G++ install directory pathname contains whitespace characters have been fixed.

Internal compiler error with large NEON types. A bug has been fixed that caused internal compiler errors when compiling code using NEON types at least 32 bytes wide.

Temporary files on Microsoft Windows. On Microsoft Windows hosts, Sourcery G++ Lite now uses the standard Windows algorithm to choose the directory in which to place temporary files. This

change eliminates a crash that occurred if none of the `TEMP`, `TMP`, or `TMPDIR` variables were set to a suitable directory.

Vectorized shift fix. A bug has been fixed that caused incorrect code for loops containing a right shift by a constant. The bug affected code compiled with `-mfpu=neon` and loop vectorization enabled with `-O3` or `-ftree-vectorize`.

Incorrect code for nested functions. A bug in GCC that caused the compiler to generate incorrect code for nested functions has been fixed. The bug resulted in incorrect stack alignments in the affected functions.

Binutils update. The binutils package has been updated to version 2.19.51.20090205 from the FSF trunk. This update includes numerous bug fixes.

ARM build attributes conformance improvements. Several ARM EABI 2.07 conformance issues relating to the handling of build attributes in the assembler and linker have been fixed. All build attribute types are now recognized, and can now be declared by name, in addition to by number. Support for merging attributes in the linker has been improved, and the linking of incompatible objects is now detected and rejected in more cases.

Internal compiler error with `-fremove-local-statics`. An internal compiler error that occurred when using the `-fremove-local-statics` option has been fixed. The error occurred when compiling code with function-local `static` array or structure variables.

Linker crash on incompatible input files. Some third-party compilers, including ARM RealView® 4.0, produce a build attribute marking output files that are not compatible with the ABI for the ARM Architecture. This attribute sometimes caused the linker to crash. The linker now correctly issues an error message.

3.3.3. Changes in Sourcery G++ Lite 2008q3-67

Bug fix for assembly listing. A bug that caused the assembler to produce corrupted listings (via the `-a` option) on Windows hosts has been fixed.

Optimizer bug fix. A bug that caused an `unrecognizable insn` internal compiler error when compiling at optimization levels above `-O0` has been fixed.

VFP compiler fix. A compiler bug that resulted in `internal compiler error: output_operand: invalid expression as operand` when generating VFP code has been fixed.

Misaligned NEON memory accesses. A bug has been fixed that caused the compiler to use aligned NEON load/store instructions to access misaligned data when autovectorizing certain loops. The bug affected code compiled with `-mfpu=neon` and loop vectorization enabled with `-O3` or `-ftree-vectorize`.

3.3.4. Changes in Sourcery G++ Lite 2008q3-40

Definition of `va_list`. In order to conform to the ABI for the ARM Architecture, the definition of the type of `va_list` (defined in `stdarg.h`) has been changed. This change impacts only the mangled names of C++ entities. For example, the mangled name of a C++ function taking an argument of type `va_list`, or `va_list *`, or another type involving `va_list` has changed. Since this is an incompatible change, you must recompile and relink any modules defining or using affected `va_list`-typed entities.

Thumb-2 assembler fixes. The Thumb-2 encodings of `QADD`, `QDADD`, `QSUB`, and `QDSUB` have been corrected. Previous versions of the assembler generated incorrect object files for these instructions. The assembler now accepts the `ORN`, `QASX`, `QSAX`, `RRX`, `SHASX`, `SHSAX`, `SSAX`, `USAX`, `UHASX`, `UQSAX`, and `USAX` mnemonics. The assembler now detects and issues errors for invalid uses of register 13 (the stack pointer) and register 15 (the program counter) in many instructions.

Bug fix for objcopy/strip. An objcopy bug that corrupted COMDAT groups when creating new binaries has been fixed. This bug also affected **strip -g**.

Binutils support for DWARF Version 3. The **addr2line** command now supports binaries containing DWARF 3 debugging information. The **ld** command can display error messages with source locations for input files containing DWARF 3 debugging information.

NEON improvements. Several improvements and bug fixes have been made to the NEON Advanced SIMD Extension support in GCC. A problem that caused the autovectorizer to fail in some circumstances has been fixed. Also, many of the intrinsics available via the `arm_neon.h` header file now have improved error checking for out-of-bounds arguments, and the `vget_lane` intrinsics that return signed values now produce improved code.

NEON compiler fix. A compiler bug that resulted in incorrect NEON code being generated has been fixed. Typically the incorrect code occurred when NEON intrinsics were used inside small `if` statements.

Mixed-case NEON register aliases. An assembler bug that prevented NEON register aliases from being created with mixed-case names using the `.dn` and `.qn` directives has been fixed. Previously only aliases created with all-lowercase or all-uppercase names worked correctly.

Inline functions declared with `dllexport`. The compiler now always emits an out-of-line copy of inline functions declared with the `__declspec(dllexport)` specifier. This allows such functions to be referenced from outside the DLL, just like non-inline functions.

Janus 2CC support. GCC now includes a work-around for a hardware bug in Avalent Janus 2CC cores. To compile and link for these cores, use the `-mfix-janus-2cc` compiler option. If you are using the linker directly use the `--fix-janus-2cc` linker option.

ARM exception handling bug fix. A bug in the runtime library has been fixed that formerly caused throwing an unexpected exception in C++ to crash instead of calling the unexpected exception handler. The bug only affected C++ code compiled by non-GNU compilers such as ARM RealView®.

Mangling of NEON type names. A bug in the algorithm used by the C++ compiler for mangling the names of NEON types, such as `int8x16_t`, has been fixed. These mangled names are used internally in object files to encode type information in addition to the programmer-visible names of the C++ variables and functions. The new mangled name encoding is more compact and conforms to the ARM C++ ABI.

Half-precision floating point. Sourcery G++ now includes support for half-precision floating point via the `__fp16` type in C and C++. The compiler can generate code using either hardware support or library routines. For more information, see Section 3.2.3, “Half-Precision Floating Point”.

3.3.5. Changes in Sourcery G++ Lite 2008q3-10

Uppercase operands to IT instructions. The assembler now accepts both uppercase and lowercase operands for the `IT` family of instructions.

NEON autovectorizer fix. A compiler bug that caused generation of bad `VLD1` instructions has been fixed. The bug affected code compiled with `-mfpu=neon -ftree-vectorize`.

Output files removed on error. When GCC encounters an error, it now consistently removes any incomplete output files that it may have created.

ARMv7 offset out of range errors. An assembler bug that resulted in `offset out of range` errors when compiling for ARMv7 processors has been fixed.

Symbian binary relocation. The linker now correctly generates relocations for writable data as data-relative rather than text-relative. The former behavior caused runtime failures accessing writable data.

Thumb-2 `MUL` encoding. In Thumb-2 mode, the assembler now encodes `MUL` as a 16-bit instruction (rather than as a 32-bit instruction) when possible. This fix results in smaller code, with no loss of performance.

ARM C++ ABI utility functions. Vector utility functions required by the ARM C++ ABI no longer crash when passed null pointers. The affected functions are `__aeabi_vec_dtor_cookie`, `__aeabi_vec_delete`, `__aeabi_vec_delete3`, and `__aeabi_vec_delete3_nodtor`. These functions are not intended for use by application programmers; they are only called by compiler-generated code. They are not presently used by the GNU C++ compiler, but are used by some other compilers, including ARM's RealView® compiler.

GCC version 4.3.2. Sourcery G++ Lite for ARM SymbianOS is now based on GCC version 4.3.2. For more information about changes from GCC version 4.2 that was included in previous releases, see `http://gcc.gnu.org/gcc-4.3/changes.html`.

Smaller Thumb-2 code. When optimizing for size (i.e., when `-Os` is in use), GCC now generates the 16-bit `MULS` Thumb-2 multiply instruction instead of the 32-bit `MUL` instruction.

Thumb-2 `RBIT` encoding. An assembler bug that resulted in incorrect encoding of the Thumb-2 `RBIT` instruction has been fixed.

Marvell Feroceon compiler bug fix. A bug that caused an internal compiler error when optimizing for Marvell Feroceon CPUs has been fixed.

Misaligned accesses to packed structures fix. A bug that caused GCC to generate misaligned accesses to packed structures has been fixed.

Bug fix for objdump on Windows. An objdump bug that caused the `-S` option not to work on Windows in some cases has been fixed.

3.3.6. Changes in Sourcery G++ Lite 2008q1-126

Disassembler bug fix. A bug in the disassembler has been fixed that formerly caused **objdump** to crash when processing raw binary files, or other executables with an empty symbol table.

NEON assembler symbols. An assembler bug that caused spurious undefined symbols to be generated has been fixed. The `mov d0, d1` instruction would incorrectly cause an undefined symbol `d1` to be created.

3.3.7. Changes in Sourcery G++ Lite 2008q1-102

ARM Cortex-A9 processor support. The compiler can now generate code optimized for the ARM Cortex-A9 processor. This is enabled by the the `-mcpu=cortex-a9` command-line option.

MOVW and MOVT relocations. A linker error that resulted in incorrect offsets when processing relocations on `MOVW` and `MOVT` instructions referencing mergeable string sections has been fixed.

Improved argument-passing code. The compiler can now generate more efficient code for certain functions whose arguments must be sign-extended to conform with language or ABI conventions. The required conversion was formerly being performed both in the called function and at all call sites; now the redundant conversion has been eliminated for functions that can only be called within the compilation unit where they are defined.

Linker error allocating ELF segments. A bug where the linker produces an incorrect error message with segments at the top of the address space has been fixed.

GCC stack size limit increased. On Windows hosts, the maximum stack size for the GCC executable has been increased. This means that more complex programs can be compiled.

Invalid object file after strip. A bug in the assembler has been fixed that formerly caused `.set` *`symbol expression`* constructs to emit *`symbol`* in the wrong section. This in turn caused inconsistent behavior after stripping the symbol table.

GCC update. The GCC package has been updated to version 4.2.3. This version includes numerous bug fixes since GCC 4.2.

License checking on Linux. Sourcery G++'s license-checking logic now includes a workaround for a kernel bug present in some versions of Linux. This bug formerly caused failures with an error message from the `cs-license` component.

Cortex-R4F and VFPv3-D16. Sourcery G++ now supports the ARM Cortex-R4F CPU and the VFPv3-D16 floating-point coprocessor. These can be selected with `-mcpu=cortex-r4f` and `-mfpu=vfpv3-d16`, respectively.

Invalid `dllimport` and `dllexport` attributes. GCC now correctly diagnoses some invalid uses of `dllimport` and `dllexport` on `typedef` declarations.

Size optimization bug. A code generation bug that caused corruption of function arguments when compiling with `-Os` has been fixed. The corruption occurred as part of the sibling call optimization.

UNC pathname bug fix. A bug has been fixed that caused linker errors on Windows hosts when running a Sourcery G++ toolchain installed in a UNC path (`\\`*`host`*`\`*`directory`*).

Linker crash on invalid input files. Some older versions of GCC generated object files with invalid mergeable string sections when compiling with `-fmerge-all-constants`. This bug was fixed in Sourcery G++ as of version 4.1-43. However, since system libraries included with some GNU/Linux distributions were affected by this bug, the linker has now been changed to accept object files with such invalid sections, rather than crash or produce an error message.

Binutils update. The binutils package has been updated to version 2.18.50.20080215 from the FSF trunk. This update includes numerous bug fixes.

3.3.8. Changes in Sourcery G++ Lite 2007q3-66

Overlapping operands for long multiply instructions. An incorrect assembler warning has been removed in the case of overlapping source and destination operands for `UMULL`, `SMULL`, `UMLAL` and `SMLAL` instructions on ARMv6 processors.

Linker crash on invalid input files. Some older versions of Sourcery G++ Lite generated object files with invalid mergeable string sections when compiling with `-fmerge-all-constants`. This bug was fixed as of version 4.1-43. The linker has additionally been fixed to detect and issue an error message for such invalid input files, instead of crashing.

C++ library ABI fix. GCC 4.2.1's `std::type_info` was not fully compatible with earlier versions. The ordering of four virtual functions has been fixed in this update.

Read-only variables. The C++ compiler now places variables whose types are instantiations of template classes in a read-only data section if they are declared `const` and initialized with a constant value. This changes reduces the RAM usage of affected applications.

3.3.9. Changes in Sourcery G++ Lite 2007q3-52

Preprocessing assembly code. The compiler driver passes `-I` options to the assembler, so that `#include` directives (processed by the preprocessor) and `.include` directives (processed by the assembler) use the same search path.

C++ class debug information. The flag `-femit-class-debug-always` is now disabled by default. The flag produces duplicate C++ class debug information as a work-around for older debuggers.

Dynamically-initialized `const` variables. Dynamically-initialized namespace-scope C++ variables are no longer placed in read-only data sections, even when marked `const`. These variables must be modified at startup, so they cannot be placed in ROM, even though their values cannot change once initialized.

Register allocation bug fix. A register allocation bug has been fixed. Under rare circumstances, the bug caused incorrect code generation.

iWMMXt bug fix. A GCC bug affecting code generation for iWMMXt processors has been fixed. The bug caused internal compiler errors when compiling some functions with large stack frames.

Default linker script. GCC no longer uses the simulator linker script by default. To avoid a link failure, you must specify a linker script explicitly with the `-T` command-line option, or via the `Properties` item on the `Project` menu in the Sourcery G++ IDE.

Volatile postincrement and postdecrement bug fix. A code generation bug that caused postincrement or postdecrement of a volatile object to reread the modified value from that object in some contexts has been fixed. The bug affected code performing a comparison of the postincrement or postdecrement expression with a constant, or that was optimized to comparison with a constant.

Widening multiply instructions for ARMv6 and later. GCC now makes use of the 32-to-64-bit widening multiply instructions (`umull`, `smull`, `umlal`, and `smlal`) when generating code for ARMv6 and later. A bug had caused these instructions to be used for ARMv3 to ARMv5 only.

Stricter check for anonymous unions. G++ now issues an error about invalid code that uses the same name for a member of an anonymous union and an entity in the surrounding namespace. For example, you will now get an error about code like:

```
int i;
static union { int i; };
```

because both the global variable and the anonymous union member are named `i`. To make this code valid you must change one of the declarations to use a different name.

Assembler code file name suffixes. GCC now recognizes `.sx` as well as `.S` as a file name suffix indicating assembler code which must be preprocessed. The alternate suffix may be useful in conjunction with other program development tools on Windows that do not distinguish case on filenames and treat `.S` the same as `.s`, which GCC uses to indicate assembler code without preprocessing.

GCC update. The GCC package has been updated to version 4.2.1. This version includes numerous bug fixes since GCC 4.2.

Handling of `dllimport` member functions within `notshared` classes. G++ now assigns default ELF visibility to member functions marked `dllimport`, even if those member functions appear within `notshared` classes. As a result, it is now possible to use a `notshared` class in a DLL, but still define a member function marked `dllimport` in another DLL.

Smaller code for C++ destructors. G++ now generates more compact code to handle the destruction of C++ objects declared at namespace scope or declared within a function scope using the `static` keyword.

Binutils update. The binutils package has been updated to the 2007-08-19 version of the pre-2.18 FSF trunk. This contains many new improvements and bug fixes. For more information, refer to the manuals for the individual utilities, and to the binutils web site at `http://www.gnu.org/software/binutils/`.

Debugging information fix. GCC no longer generates invalid debugging information for sections with no contents. The invalid debugging information caused the GNU/Linux prelinker to crash.

Calls to undefined weak symbols. The linker now implements semantics that comply to the ARM EABI for `R_ARM_CALL` and `T_ARM_THM_CALL` relocations against undefined weak symbols. These now result in a jump to the next instruction.

Assembler skipping \ characters. A bug is fixed where the assembler would skip \ characters when they appeared at certain positions in the input file. This bug primarily affected assembler macros.

Improved diagnostics for region overflow. The linker will now give more helpful diagnostics when the object files being linked are too big for one of the memory regions defined in the linker script.

Spurious compiler warnings eliminated. GCC no longer emits warnings when linker-specific command-line options are provided in combination with modes that do not perform linking, such as with the `-c` flag.

Call shortening bug fix. GCC no longer overrides `__attribute__((long_call))` on calls to locally-defined functions when the function is weak, or when it is in a different section from the caller.

Binutils update. The binutils package has been updated from version 2.17 to the pre-2.18 FSF trunk. This is a significant update with many improvements and bug fixes.

Changes to the assembler (**as**) include:

- On MIPS targets, support for additional processors and the SmartMIPS and DSP Release 2 extensions has been added.

New linker (**ld**) features include:

- A new command-line option `--default-script` has been added to give more precise control over linker script processing.
- There are new command-line options `-Bsymbolic-functions`, `--dynamic-list`, `--dynamic-list-cpp-new`, and `--dynamic-list-data` to control symbols that should be dynamically linked.
- The new `--print-gc-sections` option lists sections removed by garbage collection.

Other changes include:

- The **objcopy** utility has a new `--extract-symbol` option to extract only symbol table information from the input file.
- The **gprof** utility now allows input files to have histogram records for several memory ranges, provided those ranges are disjoint.

For more information, refer to the manuals for the individual utilities, and the binutils web site at `http://www.gnu.org/software/binutils/`.

Forced alignment of array variables. A new option `-falign-arrays` has been added to the compiler. Specifying this option sets the minimum alignment for array variables to be the largest power of two less than or equal to their total storage size, or the biggest alignment used on the machine, whichever is smaller. This option may be helpful when compiling legacy code that uses type punning on arrays that does not strictly conform to the C standard.

ARM EABI compliance. Objects produced by Sourcery G++ are now marked as ARM ELF version 5 rather than ARM ELF version 4. This reflects compliance with recent revisions of the ARM EABI. Sourcery G++ still accepts objects marked with version 4.

Smaller C++ applications. The C++ runtime library has been modified so that using namespace-scope objects with destructors does not pull in unnecessary support functions. Therefore, statically linked C++ applications compiled with `-fno-exceptions` are substantially smaller.

ARMv6-M floating-point bug fix. A bug affecting conversion of wider floating-point types to subnormal `float` values on ARMv6-M processors has been fixed.

3.3.10. Changes in Sourcery G++ Lite 2007q1-21

NEON coprocessor system registers. The assembler now accepts the `MVFR0` and `MVFR1` coprocessor registers in `fmrx` and `fmxr` instructions.

Disabling diagnostics for use of system header and library directories. The warnings for use of options such as `-I/usr/include` when cross compiling can be disabled with a new option `-Wno-poison-system-directories`. This option is intended for use in chroot environments when such directories contain the correct headers and libraries for the target system rather than the host.

Thumb-2 doubleword writeback addressing modes. An assembler bug that caused writeback addressing modes for `ldrd` and `strd` to be incorrectly encoded has been fixed.

Thumb-2 shift instruction aliases. The assembler now accepts `mov` with shifted operands as an alias for Thumb-2 shift instructions. For example `mov r0, r1, lsl r2` is encoded as `lsl r0, r1, r2`.

EABI object attribute merging. The linker now properly merges EABI object attributes into its output file.

Thumb-2 exception return instructions. An assembler bug that caused `subs pc, lr, #const` and `movs pc, lr` to be incorrectly encoded has been fixed.

Tag_ABI_PCS_wchar_t object attributes. Objects generated with `-fshort-wchar` are now given the correct `Tag_ABI_PCS_wchar_t` EABI object attribute annotations.

Uppercase special register names. The assembler now accepts both uppercase and lowercase special register names when assembling `msr` and `mrs` instructions for the Microcontroller profile of the ARM Architecture.

3.3.11. Changes in Sourcery G++ Lite 2007q1-10

Disassembly of overlapping sections. A bug in the disassembler that caused code to be displayed as data (and vice-versa) in files with overlapping sections has been fixed. This mainly affects the **objdump** utility.

Marvell Feroceon support. Sourcery G++ Lite now generates code optimized for Marvell Feroceon CPUs when the `mcpu=marvell-f` option is specified. This option also selects runtime libraries optimized for this processor.

Fix --gc-sections and C++ exceptions. A bug in the `--gc-sections` linker option has been fixed. Previously this would incorrectly remove unwinding tables, breaking C++ applications that use exceptions.

Installer hangs while refreshing environment. The Sourcery G++ installer for Microsoft Windows now updates the `PATH` environment variable without waiting for open applications to acknowledge the update. This change prevents open applications from blocking the installer's progress.

Improved assembler diagnostics for 8-bit offsets. The assembler now correctly diagnoses out-of-range offsets to instructions such as `LDRD` as 8-bit rather than half-word offsets.

Less disk space required for installation. Sourcery G++ Lite packages are smaller because multiple copies of files have been replaced with hard and/or symbolic links when possible. Both the size of the installer images and the amount of disk space required for an installed package have been reduced.

Thumb register corruption fix. A bug in the compiler that could cause register corruption in Thumb mode has been fixed. The compiler was formerly emitting code to restore registers on function return that was not interrupt safe.

__aeabi_lcmp. An error in the libgcc implementation of `__aeabi_lcmp` that caused incorrect results to be returned has been fixed. This is a support routine defined by the ARM EABI. GCC does not normally use this routine directly, however it may be used by third-party code.

The \@ assembler pseudo-variable. A bug in the assembler that caused uses of the `\@` pseudo-variable to be mis-parsed as comments has been fixed.

Assembly of SRS instructions. An assembler bug that resulted in incorrect encoding of the Thumb-2 `SRS` instruction has been fixed. In addition the assembler supports explicit specification of the base register, as accepted by other ARM toolchains.

Symbols defined in linker scripts. A bug is fixed that caused the linker to crash in some circumstances when a linker script defined a symbol in an output section. Typically usage is where the script contained a `__DATA_LOAD = LOADADDR(.data)` statement in the `.data` section.

Crash when generating vector code. A bug that sometimes caused the compiler to crash when invoked with the `-ftree-vectorize` option has been fixed.

VFP disassembly crash. A bug that caused crashes when disassembling some forms of the VFP `fmrx` and `fmxr` instructions has been fixed.

ARM NEON store intrinsics bug fix. A compiler bug that incorrectly caused calls to ARM NEON store intrinsics (such as `vst1_u8`) to be optimized away has been fixed.

Improvements to ARM NEON support. The ARM NEON support in GCC has been enhanced to comply with new rules for containerized vector types specified in the ARM procedure call standard. Additionally, the compiler now rejects implicit conversions between NEON polynomial vector types and NEON integer vector types of the same layout.

Propagation of Thumb symbol attributes. Symbols referring to Thumb functions on ARM targets now have their Thumb attribute correctly propagated to any aliases defined with `.set` or `.symver`.

Complex numbers bug fix. A bug that could lead to incorrect code generation for code using complex numbers has been fixed.

Use of system header and library directories diagnosed. The compiler and linker now diagnose the incorrect use of native system header and library directories for cross-compilation. This typically arises from options such as `-I/usr/X11R6/include` hard-coded in build scripts written without a view to cross-compilation.

Linking of non-ELF images. A linker bug that could cause a crash when linking non-ELF objects for ARM targets has been fixed.

Initialization priorities. The `constructor` and `destructor` function attributes now accept an optional priority argument. Constructors with small priorities are run before those with larger priorities; the opposite is true for destructors. For example:

```
void f __attribute__((constructor(500)));
void f() {
  /* Perform initialization.  */
}
```

defines a function `f` with priority 500. This function will be run before constructors with larger priorities. Constructor and destructors with no explicit priority argument have priority 65535, the maximum permitted value.

Thumb-2 IT block code generation error fixed. A bug in Thumb-2 code generation has been fixed. This bug would result in missing IT instructions, causing the assembler to reject the code.

ARM Cortex-R4 performance improvements. Sourcery G++ Lite now generates faster code when compiling for the ARM Cortex-R4 processor by scheduling instructions for the processor's pipelines. To generate code for this processor, use the `-mcpu=cortex-r4` command-line option.

Invalid load instructions. A bug in the compiler which caused it to generate invalid assembly (e.g. `ldrd r0, [#0, r2]`) has been fixed.

VFPv3/NEON debug information. A bug in the compiler which caused it to generate incorrect debug information for code using VFPv3/NEON registers has been fixed. The debugger is now able unable to locate and display values held in these registers.

iWMMXt compiler errors. A compiler bug that caused invalid assembly when generating iWMMXt code has been fixed.

ARMv6-M system instructions. An assembler bug that caused some ARMv6-M system instructions to be incorrectly rejected has been fixed. The affected instructions are `msr`, `mrs`, `yield`, `wfi`, `wfe` and `sev`.

Assembling Thumb store-multiple instructions. The assembler now issues an error message instead of crashing on load/store multiple instructions that incorrectly use Thumb-2 addressing modes (e.g., `ldmdb`) in legacy Thumb syntax mode. If you want to use these address modes, you should use unified syntax mode instead.

Thumb-2 stack decrement misassembly. An assembler bug that resulted in incorrect encoding of the 32-bit Thumb-2 form of the `sub sp, sp, #const` instruction has been fixed. Previously this was misassembled as `subs`.

Naked functions. Functions marked with `__attribute__((naked))` no longer contain prologue and epilogue code. Please refer to the GCC manual for the proper use of this attribute.

Assembly of Thumb-2 load/store multiple instructions. The Thumb-2 `ldm` and `stm` assembly mnemonics are now assembled to `ldr` and `str` instructions when a single register is transferred, as specified in the Thumb-2 Architecture Supplement.

Conditional Thumb-2 branch instructions. A linker bug that could cause objects involving conditional Thumb-2 branch instructions to be incorrectly rejected has been fixed.

Fix addr2line defect. The binary utility **addr2line** now operates correctly on 64-bit targets with DWARF2 debug information.

Thumb-2 assembler infinite loop. An assembler bug that would cause it to enter an infinite loop when processing some Thumb-2 assembly has been fixed.

Assembler warnings about overlapping multiplication operands. The assembler no longer warns about overlapping `Rd` and `Rm` operands when assembling `mul` and `mla` instructions for the ARM architecture version six or above.

Alignment bug fix. A bug has been fixed that formerly caused incorrect code to be generated in some situations for copying structure arguments being passed by value. The incorrect code caused alignment errors on stack accesses on some targets.

ARM Cortex-A8 performance improvements. Sourcery G++ Lite now generates faster code when compiling for the ARM Cortex-A8 processor by scheduling instructions for the processor's dual-issue pipelines. To generate code for this processor, use the `-mcpu=cortex-a8` command-line option.

GCC version 4.2. Sourcery G++ Lite for ARM SymbianOS is now based on GCC version 4.2. For more information about changes from GCC version 4.1 that was included in previous releases, see `http://gcc.gnu.org/gcc-4.2/changes.html`.

Improve handling of corrupt debug information. The binary utility **readelf** now copes more gracefully with corrupted DWARF 2 information.

Smaller C++ programs. Rarely-used functions in the C++ runtime library have been isolated into separate object files so that they will not be included unless needed. As a result, most statically linked C++ programs are smaller.

3.3.12. Changes in Sourcery G++ Lite 4.1-37

Preserve volatile accesses. Reads from volatile memory are no longer incorrectly optimized away at higher optimization levels.

3.3.13. Changes in Sourcery G++ Lite 4.1-34

Implicit conversions between generic vector types. Implicit conversions between generic vector types are now only permitted when the two vectors in question have the same number of elements and compatible element types. (Note that the restriction involves *compatible* element types, not implicitly-convertible element types: thus, a vector type with element type `int` may not be implicitly converted to a vector type with element type `unsigned int`.) This restriction, which is in line with specifications for SIMD architectures such as AltiVec, may be relaxed using the flag `-flax-vector-conversions`. This flag is intended only as a compatibility measure and should not be used for new code.

`type_info` comparison fix. Comparison of `type_info` objects now uses pointer comparison where possible.

C++ forced unwinding fixes. Some bugs relating to forced unwinding through C++ code have been fixed.

Support for additional Stellaris boards. Linker scripts are provided for the 6xx and 8xx series Stellaris boards.

Linux support for USB Debug Sprite. A new driver is included to allow the Sourcery G++ Lite USB Debug Sprite to run on Linux hosts. See Chapter 3, *Sourcery G++ Lite for ARM SymbianOS* for additional information.

3.3.14. Changes in Sourcery G++ Lite 4.1-33

Linker scripts. A bug is fixed where an erroneous linker script would cause a linker crash. An error message is now produced.

Newlib memory use improvements. The memory overhead of linking with newlib is reduced. Applications that use only a minimal set of library features may now require significantly less memory.

3.3.15. Changes in Sourcery G++ Lite 4.1-31

Compiler alias analysis. The type-based alias analysis performed by the compiler when compiling with `-O2` or with `-fstrict-aliasing` is now more conservative. The more aggressive analysis used in previous versions sometimes resulted in incorrect code generation.

Fully relocatable preprocessor. When cross-compiling, the default preprocessor search path includes only the directories present in the installed toolchain. This speeds up the preprocessor and prevents the unintentional use of unrelated files and directories on the machine where it is installed.

3.3.16. Changes in Sourcery G++ Lite 4.1-29

Support for new-style symbol hashing. Support has been added in binutils and the prelinker for new-style (also known as `DT_GNU_HASH`) symbol hashing. This can dramatically speed up symbol

resolution time and is particularly applicable in environments where full prelinking is not possible (for example where shared libraries are dynamically opened at runtime). The new-style hashing may be enabled by passing `--hash-style=gnu` to the linker.

Prelinker update. The prelinker has been updated to the current upstream sources and some bugs affecting operation have been fixed.

3.3.17. Changes in Sourcery G++ Lite 4.1-28

Improved support for ROM debugging. GDB now determines ROM regions automatically from the memory map included in target configuration files. This information is used to determine when hardware breakpoints should automatically be used (for instance the **step**, **next** and **finish** commands). Separate ROM configurations have been removed from the Eclipse debugger menu. The Eclipse GUI has been extended to provide improved support for debugging programs in ROM, when a memory map is not automatically available.

3.3.18. Changes in Sourcery G++ Lite 4.1-27

Rename Windows executables. The Windows host tools **make.exe** and **rm.exe** are now named **cs-make.exe** and **cs-rm.exe**. This change avoids conflicts with tools provided by other distributors.

iWMMXt bug fixes. Some bugs involving incorrect code generation and internal compiler errors when generating iWMMXt code have been fixed.

Cortex-M3 startup code. The ARMv7-M startup code (`armv7m-crt0.o`) incorrectly contained ARM code. This has been replaced with Thumb-2 code.

ARM EABI coverage testing support. Coverage testing using GCOV is now supported for the ARM EABI target. Please refer to the *GNU C Compiler Manual (HTML)* for more information on coverage testing.

3.3.19. Changes in Sourcery G++ Lite 4.1-23

Windows debugging fix. In recent releases of Sourcery G++ Lite, the GDB **target remote |** command would hang on Windows. This affected both command line and Eclipse debugging when using the Sourcery G++ Lite Debug Sprite.

Stellaris USB Debug Sprite improvements. The former USB Debug Stub, **armswd**, is now known as the USB Debug Sprite, and has been renamed to **arm-stellaris-eabi-sprite**. In addition, its initialization sequence has been updated to recognize the r1p1 release of the Cortex-M3 processor.

Incompatible changes to Stellaris linker scripts. Sourcery G++ Lite now supports linking executables to run from RAM as well as ROM. As part of this change, there are now separate RAM and ROM versions of the linker scripts for each supported board, and the former ROM-based versions have been renamed. For example, if you were formerly linking with `-T lm3s10x.ld`, you should now use `-T lm3s10x-rom.ld` to get the same behavior.

3.3.20. Changes in Sourcery G++ Lite 4.1-21

Eclipse debuggers. Eclipse configurations for debugging arm-none-eabi applications using the GDB simulator and remote debug stubs have been added.

iWMMXt2 support. The assembler and disassembler now support iWMMXt2 instructions.

NEON intrinsics support. GCC now supports NEON intrinsics defined in the `arm_neon.h` header. These are the same intrinsics supported by the ARM RealView® compiler and are documented in the 'ARM NEON Intrinsics' section of the GCC manual.

3.3.21. Changes in Sourcery G++ Lite 4.1-19

ARMv4t linux multilib. Linux configurations now support ARMv4t CPUs.

Linker scripts. Several problems with the linker scripts for bare-metal targets have been fixed.

3.3.22. Changes in Sourcery G++ Lite 4.1-18

Binutils update. The binutils in this release is based on the final binutils 2.17 release.

GDB update. The included version of GDB has been upgraded to 6.5.50.20060822. This includes numerous bug fixes from the previous version.

GDB support for flash memory. The GDB **load** command can now write to flash memory, if the remote debugging stub contains appropriate support.

Compiler support for NEON. Initial GCC support for autovectorization and generation of NEON SIMD instructions has been added.

Bare metal Cortex-M3 configurations. Bare metal configurations now support generating images for use on ARMv7-M devices (eg. Cortex-M3).

iWMMXt support in GLIBC. GLIBC's `setjmp` and `longjmp` now support saving and restoring iWMMXt registers on hardware with those registers. This requires a kernel reporting `iwmmxt` in the `Features` entry in `/proc/cpuinfo`.

iWMMXt exception handling support. Exception handling now restores the values of iWMMXt registers correctly.

Corrected IPC functions. A bug in GLIBC's `msgctl`, `semctl`, and `shmctl` functions has been corrected.

3.3.23. Changes in Sourcery G++ Lite 4.1-16

GCC update. This release is based on GCC 4.1.1.

Fully relocatable compiler. The compiler now searches for its components only in the directory where it has been installed, and no longer also searches pathnames matching the directory where it was configured. This speeds up the compiler and prevents problems with unintentionally finding unrelated files or directories on the machine where it has been installed.

Stack permission marking for ARM GNU/Linux. Non-executable stacks can provide increased security against some forms of buffer overflow attacks. The tools involved must coordinate the annotation of required stack permissions, either executable, or non-executable. For ARM GNU/Linux targets the compiler now outputs annotations indicating the required stack permissions.

3.3.24. Changes in Sourcery G++ Lite 4.1-15

Stabs debugging information support. Using the Stabs debugging format (available with `-gstabs` or `-gstabs+`) now works in conjunction with `-mthumb`. CodeSourcery recommends the default DWARF debugging format (available with `-g`) as DWARF is a more comprehensive debugging format.

3.3.25. Changes in Sourcery G++ Lite 4.1-13

Stellaris linker scripts in IDE. Linker scripts may now be selected via a drop-down menu in Eclipse.

Stellaris linker scripts for 3xx series CPUs. The linker scripts for 3xx Series CPUs now place the ISR vector at address zero, as required by all Cortex-M3 cores.

Stellaris USB Debug Sprite improvements. Bug fixes and new features include:

- A bug that caused the stub not to correctly update the program counter and other register values was fixed. As a result of this fix, it is now possible to run programs residing in SRAM using the `continue` command from GDB.
- The stub no longer prints status messages via GDB console output when invoked with the `-q` command-line option.
- The stub's initialization sequence was updated to recognize revision C Cortex-M3 hardware.

3.3.26. Changes in Sourcery G++ Lite 4.1-9

Stellaris USB Debug Sprite improvements. Program images exceeding 4K can now be uploaded to flash memory.

Additional Stellaris boards supported. The Stellaris 301, 310, 315, and 316 CPUs are now supported. Linker scripts have been added for these boards.

3.3.27. Changes in Sourcery G++ Lite 4.1-8

Stellaris USB Debug Sprite improvements. Several bug fixes and enhancements were made to the USB Debug Stub. In particular:

- Bugs in the implementation of `open`, `read`, and `lseek` were fixed.
- Support was added for `isatty`, `rename`, `unlink`, and `system`.
- Memory reads that span 4K block boundaries now work correctly.

3.3.28. Changes in Sourcery G++ Lite 4.1-4

Runtime libraries. Support for ARMv7 including Cortex-M3 and pure Thumb-2.

Assembler. Support for NEON and VFPv3, including unified NEON/VFP syntax.

3.3.29. Changes in Sourcery G++ Lite 4.1-1

Initial release. This release is based on GCC 4.1.0.

Chapter 4
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.

4.1. Terminology

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

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

4.2. System Requirements

4.2.1. Host Operating System Requirements

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

- Microsoft Windows NT 4, Windows 2000, Windows XP, and Windows Vista systems using IA32, AMD64, and EM64T processors.
- GNU/Linux systems using IA32, AMD64, or EM64T processors, including Debian 3.0 (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.

4.2.2. Host Hardware Requirements

In order to install and use Sourcery G++ Lite, you must have at least 128MB 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. Typically, you should plan on at least 400MB.

In addition, the graphical installer requires a similar amount of temporary space during the installation process. On Microsoft Windows hosts, the installer uses the location specified by the `TEMP` environment variable for these temporary files. If there is not enough free space on that volume, the installer

prompts for an alternate location. On Linux hosts, the installer puts temporary files in the directory specified by the `IATEMPDIR` environment variable, or `/tmp` if that is not set.

4.2.3. Target System Requirements

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

4.3. 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 4.4, “Installing Sourcery G++ Lite”.

If you have a Sourcery G++ subscription (or evaluation), then you can log into the Sourcery G++ Portal[1] to download your Sourcery G++ toolchain(s). CodeSourcery also makes some toolchains available to the general public from the Sourcery G++ web site[2]. These publicly available toolchains do not include all the functionality of CodeSourcery's product releases.

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.

4.4. 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.

4.4.1. 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.

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
```

4.4.2. 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
```

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

After the installer starts, follow the on-screen dialogs to install Sourcery G++ Lite.

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
```

4.4.3. 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++-2009q1`.

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
```

4.5. Installing Sourcery G++ Lite Updates

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

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

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

4.6. 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.

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

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

On Microsoft Windows Vista hosts, select `Start`, then `Settings` and finally `Control Panel`. Select the `Uninstall a program` task. Scroll down and double click on `Sourcery G++ for ARM SymbianOS`. Follow the on-screen dialogs to uninstall 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 `Add or Remove Programs` (non-Vista) or `Uninstall a program` (Vista) to remove the drivers separately. Depending on the device, you may need to reboot your computer to complete the driver uninstall.

4.6.2. 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. The `arm-none-symbianelf` directory located in the install directory will be removed entirely by the uninstaller. Please back up any changes you have made to this directory, such as modified linker scripts.

Start the graphical uninstaller by invoking the executable Uninstall shell script located in your installation directory. After the uninstaller starts, follow the on-screen dialogs to uninstall 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.

4.6.3. 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.

4.7. Setting up the Environment

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

4.7.1. Setting up the Environment on Microsoft Windows Hosts

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

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

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

Verify that the last line of the output contains: `Sourcery G++ Lite 2009q1-162`.

4.7.1.2. 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.

4.7.2. 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-symbianelf/man`.

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

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

and verifying that you receive the message:

```
arm-none-symbianelf-g++: no input files
```

Chapter 5
Using Sourcery G++ from the Command Line

This chapter demonstrates the use of Sourcery G++ Lite from the command line. This chapter assumes you have installed Sourcery G++ Lite as described in Chapter 4, *Installation and Configuration*.

5.1. 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-symbianelf, as indicated by the **arm-none-symbianelf** 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-symbianelf-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-symbianelf/lib`.

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

5.2. Running Applications on the Target System

Consult your target board documentation for instructions on loading programs onto the target, and running them.

5.3. Running Applications from GDB

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

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.

5.3.1. 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.

5.3.2. 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
```

Chapter 6
Next Steps with Sourcery G++

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

6.1. Sourcery G++ Knowledge Base

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

For more information on CodeSourcery support, see Chapter 2, *Sourcery G++ Subscriptions*.

6.2. 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-symbianelf/` 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-symbianelf/man/man1
```

Then you can invoke **man** as:

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

Alternatively, if you use **man** regularly, you'll probably find it more convenient to add the directory containing the Sourcery G++ man pages to your `MANPATH` environment variable. This should go in your `.profile` or equivalent shell startup file; see Section 4.7, “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/