Sourcery G++

ARM EABI

Sourcery G++ 2011.02-2

Getting Started

Sourcery G++: ARM EABI: Sourcery G++ 2011.02-2: Getting Started

CodeSourcery, Inc.

Abstract

This guide explains how to install and build applications with Sourcery G++, CodeSourcery's customized, validated, and supported version of the GNU Toolchain. Sourcery G++ 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++ both from the IDE and from the command line.

Table of Contents

Preface v
1. Intended Audience vi
2. Organization vi
3. Typographical Conventions vii
1. Quick Start 1
1.1. Installation and Set-Up 2
1.2. Configuring Sourcery G++ for the Target System 2
1.3. Building Your Program 3
1.4. Running and Debugging Your Program 3
2. Installation and Configuration 4
2.1. Terminology 5
2.2. System Requirements 5
2.3. Registering with the Sourcery G++ Portal 6
2.4. Downloading an Installer 6
2.5. Installing Sourcery G++ 7
2.6. Installing Sourcery G++ Updates 10
2.7. Setting up the Environment 10
2.8. License Keys 12
2.9. Installing Add-Ons 17
2.10. Uninstalling Sourcery G++ 18
3. Sourcery G++ for ARM EABI 20
3.1. Included Components and Features 21
3.2. Library Configurations 21
3.3. CodeSourcery C Library 24
3.4. Using Sourcery G++ with Kinetis Boards 24
3.5. Using Sourcery G++ with Stellaris Boards 25
3.6. Using Sourcery G++ with STM32 Boards 26
3.7. Peripheral Register Browsing 26
3.8. Using Flash Memory 28
3.9. Using VFP Floating Point 29
3.10. ABI Compatibility 30
3.11. ARM Profiling Implementation 31
3.12. Object File Portability 31
4. Using the Sourcery G++ IDE 33
4.1. Overview 34
4.2. Building Applications 34
4.3. Debugging Applications 41
4.4. Advanced IDE Features 52
5. Using Sourcery G++ from the Command Line 65
5.1. Building an Application 66
5.2. Running Applications on the Target System 66
5.3. Running Applications from GDB 66
6. CS3™: The CodeSourcery Common Startup Code Sequence 69
6.1. Linker Scripts 70
6.2. Program Startup and Termination 72
6.3. Memory Layout 75
6.4. Interrupt Vectors and Handlers 77
6.5. Supported Boards for ARM EABI 78
6.6. Interrupt Vector Tables 148
7. Sourcery G++ Debug Sprite 163
7.1. Probing for Debug Devices 164

7.2. Debug Sprite Example 164
7.3. Invoking Sourcery G++ Debug Sprite 165
7.4. Sourcery G++ Debug Sprite Options 166
7.5. ARMUSB (Stellaris) Devices 167
7.6. Remote Debug Interface Devices 169
7.7. Actel FlashPro Devices 169
7.8. Keil ULINK2 Devices 170
7.9. Altera Devices 172
7.10. SEGGER J-Link Devices 173
7.11. P&E Devices 174
7.12. Debugging a Remote Board 175
7.13. Supported Board Files 176
7.14. Board File Syntax 181
8. Next Steps with Sourcery G++ 185
8.1. Sourcery G++ Support 186
8.2. Sourcery G++ Knowledge Base 186
8.3. Example Programs 186
8.4. Manuals for GNU Toolchain Components 186
8.5. Help for the Sourcery G++ IDE 187
A. Sourcery G++ Release Notes 189
A.1. Changes in Sourcery G++ for ARM EABI 190
B. Sourcery G++ Licenses 197
B.1. Licenses for Sourcery G++ Components 198
B.2. Sourcery G++ Software License Agreement 199
B.3. Attribution 207

Preface

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

1. Intended Audience

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

2. Organization

This document is organized into the following chapters and appendices:

Chapter 1, “Quick Start”	This chapter includes a brief checklist to follow when installing and using Sourcery G++ for the first time. You may use this chapter as an abbreviated guide to the rest of this manual.
Chapter 2, “Installation and Configuration”	This chapter describes how to download, install and configure Sourcery G++. This section describes the available installation options and explains how to set up your environment so that you can build applications.
Chapter 3, “Sourcery G++ for ARM EABI”	This chapter contains information about using Sourcery G++ that is specific to ARM EABI targets. You should read this chapter to learn how to best use Sourcery G++ on your target system.
Chapter 4, “Using the Sourcery G++ IDE”	This chapter explains how to use the Sourcery G++ IDE, which is based on Eclipse. The IDE provides a fully-integrated programming environment that allows you to edit, build, and debug programs.
Chapter 5, “Using Sourcery G++ from the Command Line”	This chapter explains how to build applications with Sourcery G++ 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, “CS3™: The CodeSourcery Common Startup Code Sequence”	CS3 is CodeSourcery's low-level board support library. This chapter documents the boards supported by Sourcery G++ and the compiler and linker options you need to use with them. It also explains how you can use and modify CS3-provided definitions for memory maps, system startup code and interrupt vectors in your own code.
Chapter 7, “Sourcery G++ Debug Sprite”	This chapter describes the use of the Sourcery G++ Debug Sprite for remote debugging. The Sprite allows you to debug programs running on a bare board without an operating system. This chapter includes information about the debugging devices and boards supported by the Sprite for ARM EABI.
Chapter 8, “Next Steps with Sourcery G++”	This chapter describes where you can find additional documentation and information about using Sourcery G++ and its components. It also provides information about Sourcery G++ subscriptions. CodeSourcery customers with Sourcery G++ subscriptions receive comprehensive support for Sourcery G++.

Appendix A, “Sourcery G++ Release Notes”	This appendix contains information about changes in this release of Sourcery G++ for ARM EABI. You should read through these notes to learn about new features and bug fixes.
Appendix B, “Sourcery G++ Licenses”	This appendix provides information about the software licenses that apply to Sourcery G++. Read this appendix to understand your legal rights and obligations as a user of Sourcery G++.

3. Typographical Conventions

The following typographical conventions are used in this guide:

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

Chapter 1
Quick Start

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

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

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

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

1.1. Installation and Set-Up

Register with the Sourcery G++ Support Portal. You must register with the Sourcery G++ Portal[1] before you can use this subscription version of Sourcery G++. Free evaluations are also available from the Portal. Registering with the Portal also gives you access to the latest software updates, the Sourcery G++ Knowledge Base, and support for subscription customers.

Install Sourcery G++ on your host computer. You may download an installer package from the Sourcery G++ Support Portal, or you may have received an installer on CD. The installer is an executable program that pops up a window on your computer and leads you through a series of dialogs to configure your installation. If the installation is successful, it will offer to launch the Sourcery G++ IDE and the Getting Started guide. For more information about installing Sourcery G++, including host system requirements and tips to set up your environment after installation, refer to Chapter 2, “Installation and Configuration”.

Install drivers for your debug device. If you plan to use the Sourcery G++ Debug Sprite, you may need to install drivers, libraries, or other software on your host system. Refer to Chapter 7, “Sourcery G++ Debug Sprite” for a list of supported devices and information about installing drivers and other device set-up. Sourcery G++ also supports third-party debug devices that communicate via the GDB remote serial protocol. If you plan to use one of these devices, follow the manufacturer's directions to connect the device and install any required drivers or software.

Install a license key. License keys are managed from the Sourcery G++ IDE. When you start the IDE for the first time, it pops up a dialog box to guide you through the steps to download and/or install your license key. For detailed help, refer to Section 2.8, “License Keys”.

1.2. Configuring Sourcery G++ for the Target System

Identify your target board. On bare-metal targets, you must either select an appropriate board when using the Sourcery G++ IDE to build your application, or explicitly specify a linker script for your target board on your link command line. Supported boards are listed in Chapter 6, “CS3™: The CodeSourcery Common Startup Code Sequence”. If CS3 does not support your board directly, you may use the Sourcery G++ Board Builder to create a custom board definition and linker scripts. Choose QEMU as your target board if you want to run your program in the QEMU simulator instead of on target hardware.

Identify your target libraries. Sourcery G++ supports libraries optimized for different targets. Libraries are selected automatically by the linker, depending on the processor and other options you

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

have specified. Refer to Section 3.2, “Library Configurations” for details. Some libraries are not included in the base Sourcery G++ installation, but are available as add-ons for Standard and Professional Edition subscribers. If you get a linker error when building your application, most likely the cause is that you have not installed the appropriate add-on libraries. Refer to Section 2.9, “Installing Add-Ons” for instructions on installing add-ons.

1.3. Building Your Program

Build your program using the Sourcery G++ IDE. In the Sourcery G++ IDE, create a new managed build C project. Use the dialog in the C Project wizard to configure the board and other target properties for building your project. Using the editor in the IDE, create a file containing a simple test program; save the file and build the project. For a tutorial introduction to the Sourcery G++ IDE that covers these steps, refer to Section 4.2, “Building Applications”.

Build your program with Sourcery G++ command-line tools. If you prefer, you can build your program from the command line instead of using the IDE. Create a simple test program, and follow the directions in Chapter 5, “Using Sourcery G++ from the Command Line” to compile and link it using Sourcery G++. On bare-metal targets, you must specify a linker script using the `-T` option on your link command line. Supported boards and linker scripts are listed in Chapter 6, “CS3™: The CodeSourcery Common Startup Code Sequence”.

1.4. Running and Debugging Your Program

The steps to run or debug your program depend on your target system and how it is configured. Choose the appropriate method for your target. For general information and a tutorial on using the Sourcery G++ IDE for debugging, see Section 4.3, “Debugging Applications”.

Debug your program in the QEMU emulator. The QEMU emulator provides an easy way to try out your program without requiring target hardware. QEMU support is integrated with the debugger in the Sourcery G++ IDE. Refer to Section 4.3, “Debugging Applications” for help with using the debugger. You can also run your program in QEMU from command-line GDB, as described in Section 5.3.1, “Connecting to the QEMU Emulator”.

Debug your program on the target using the Debug Sprite. You can use the Sourcery G++ Debug Sprite to load and execute your program on the target from the debugger. If you have built your program for a ROM memory profile, flash programming is handled automatically by the Sprite. For instructions on using the Sourcery G++ IDE to debug your program, refer to Section 4.3, “Debugging Applications”. The IDE also supports using the Sprite to run your program without debugger control; see Section 4.4.8, “Using Run Launches”. Refer to Section 5.3, “Running Applications from GDB” for instructions on using the Sprite from the GDB command line. Detailed reference material for the Sourcery G++ Debug Sprite, including information about supported debug devices, can be found in Chapter 7, “Sourcery G++ Debug Sprite”.

Debug your program on the target using a third-party debug device. Sourcery G++ supports debugging programs on the remote target using third-party debug devices that can communicate via the GDB remote serial protocol. In the Sourcery G++ IDE, select the External Embedded Server debugging mode; see Section 4.3, “Debugging Applications” for detailed instructions. For command-line GDB instructions, see Section 5.3, “Running Applications from GDB”.

Chapter 2
Installation and Configuration

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

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

2.1. Terminology

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

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

2.2. System Requirements

2.2.1. Host Operating System Requirements

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

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

Sourcery G++ is built as a 32-bit application. Therefore, even when running on a 64-bit host system, Sourcery G++ requires 32-bit host libraries. If these libraries are not already installed on your system, you must install them before installing and using Sourcery G++. 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++ 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++ toolchain.

2.2.2. Host Hardware Requirements

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

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

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

2.2.3. Target System Requirements

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

2.3. Registering with the Sourcery G++ Portal

If you do not already have a Sourcery G++ Portal account, you must register for one now. You must have an active Sourcery G++ subscription to download an installer or generate a license key. Evaluation subscriptions are available at no charge and also give you access to support from CodeSourcery.

If you purchased Sourcery G++ directly from CodeSourcery, you already have an account, and you may skip ahead to the next section. However, if you received Sourcery G++ with a hardware development kit or from a distributor, you probably do not have an account.

To register for an account, visit the Sourcery G++ Portal[1]. Click on the link to register for an evaluation subscription. Follow the instructions on the web site to create your account. Then, once your account is active, click the button to request an evaluation subscription.

You should request an evaluation version of Sourcery G++ that matches the version you received with your development kit. Select the host system where you will install Sourcery G++, and ARM EABI as the target system where you will run applications. Then click the `Request Evaluation` button.

If there are newer versions of Sourcery G++ than the one provided with your development kit, they will be visible through the Sourcery G++ Portal once your evaluation subscription is active. CodeSourcery recommends that you first work with the version of Sourcery G++ that came with your development kit, since CodeSourcery and the manufacturer have tested that particular combination of hardware and software. However, you may also wish to experiment with newer versions.

2.4. Downloading an Installer

If you have received Sourcery G++ on a CD, or other physical media, then you do not need to download an installer. You may skip ahead to Section 2.5, “Installing Sourcery G++”.

Log into the Sourcery G++ Portal[2] to download your Sourcery G++ toolchain(s). This version of Sourcery G++ requires a valid subscription or evaluation. CodeSourcery also makes some toolchains available to the general public from the Sourcery G++ web site[3]. 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++ is provided as an executable installer package with the `.bin` extension.

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

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

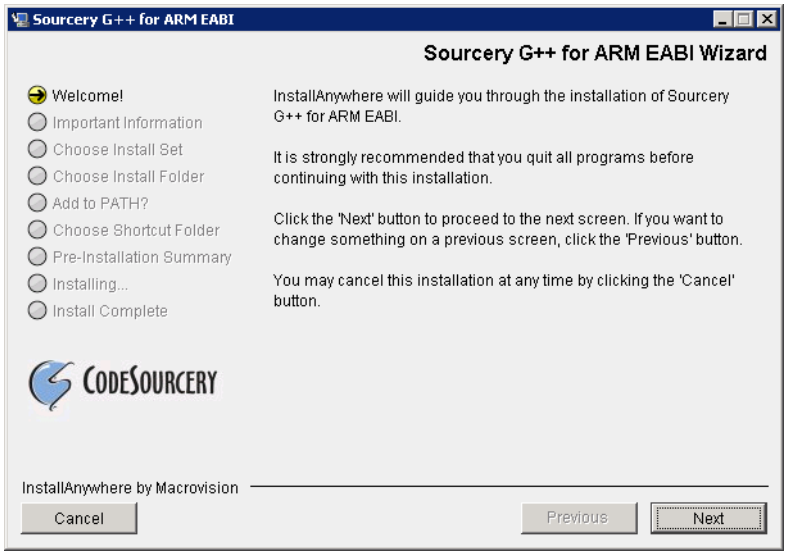
2.5. Installing Sourcery G++

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

2.5.1. Using the Sourcery G++ Installer on Microsoft Windows

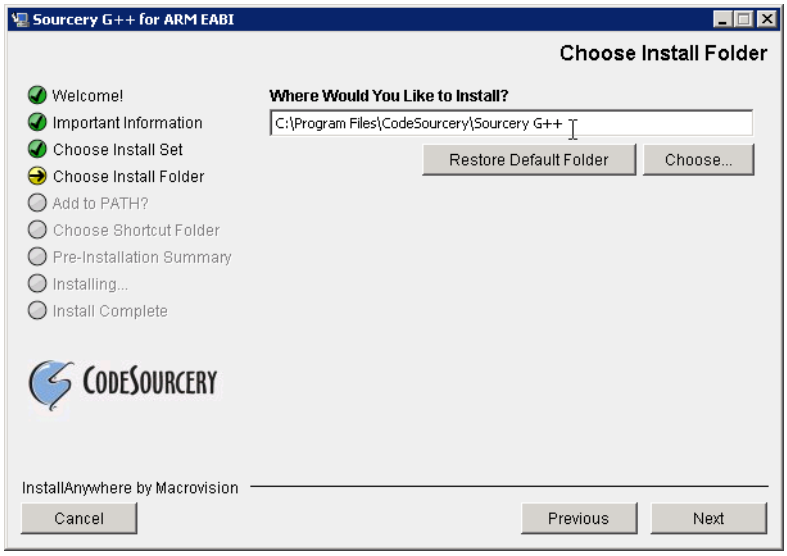
If you have received Sourcery G++ on CD, insert the CD in your computer. On most computers, the installer then starts automatically. If your computer has been configured not to automatically run CDs, open `My Computer`, and double click on the CD. If you downloaded Sourcery G++, double-click on the installer.

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

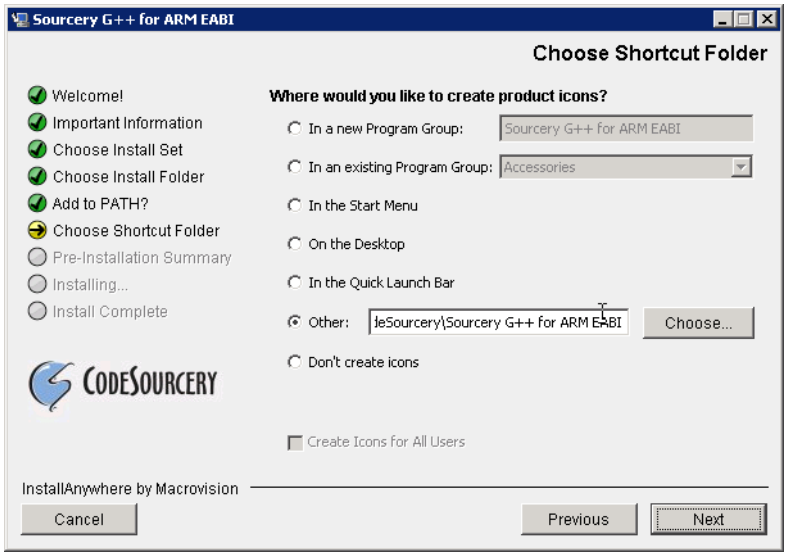


Running the Installer. The graphical installer guides you through the steps to install Sourcery G++.

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

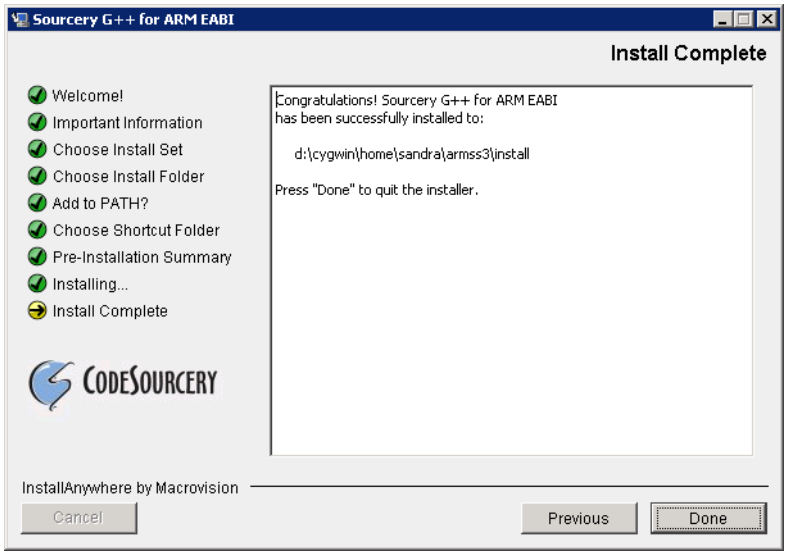


Choose Install Folder. Select the pathname to your install directory.



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

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



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

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

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

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

Start the graphical installer by invoking the executable shell script:

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

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

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

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

2.5.3. Installing the Java Runtime Environment

Sourcery G++ for ARM EABI includes the Sourcery G++ IDE, based on the Eclipse Integrated Development Environment. Eclipse is a Java application and requires the Java Runtime Environment (JRE).

If you use the graphical installer, the JRE is included when you install the Sourcery G++ IDE. Otherwise you must install the JRE separately if you wish to use the Sourcery G++ IDE. The Java Runtime Environment is available at no charge from Sun Microsystems Java website[4]. You may download either the Java Runtime Environment (JRE) or the Java Development Kit (JDK). (The JDK includes the JRE.)

[4] http://java.sun.com/j2se/

2.6. Installing Sourcery G++ Updates

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

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

2.7. Setting up the Environment

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

2.7.1. Setting up the Environment on Microsoft Windows Hosts

2.7.1.1. Setting the `PATH`

In order to use the Sourcery G++ tools from the command line, you should add them to your `PATH`. If you plan to use only the Sourcery G++ IDE, it is not necessary to adjust your `PATH`, and you may skip this step. You may also 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++ installation.

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

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

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

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

Verify that the last line of the output contains: `Sourcery G++ 2011.02-2`.

2.7.1.2. Working with Cygwin

Sourcery G++ does not require Cygwin or any other UNIX emulation environment. You can use Sourcery G++ directly from the Eclipse IDE or 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++ 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++ 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.

When you run GDB from a Cygwin shell instead of a bare Windows console, Cygwin's terminal input handling conflicts with some features used by GDB, such as **Ctrl**+**C**, arrow key support, and automatic page breaks. To work around these issues, Sourcery G++ includes a Cygwin wrapper for GDB. When you use a Cygwin console, xterm, or SSH session, run `arm-none-eabi-cyggdb` instead of `arm-none-eabi-gdb`.

2.7.2. Setting up the Environment on GNU/Linux Hosts

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

Before using Sourcery G++ 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++ in an alternate location, you must replace the directory above with `bin` subdirectory of the directory in which you installed Sourcery G++.

You may also wish to set the `MANPATH` environment variable so that you can access the Sourcery G++ manual pages, which provide additional information about using Sourcery G++. To set the

MANPATH environment variable, follow the same steps shown above, replacing PATH with MANPATH, and bin with share/doc/sourceryg++-arm-none-eabi/man.

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

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

Verify that the last line of the output contains: Sourcery G++ 2011.02-2.

2.8. License Keys

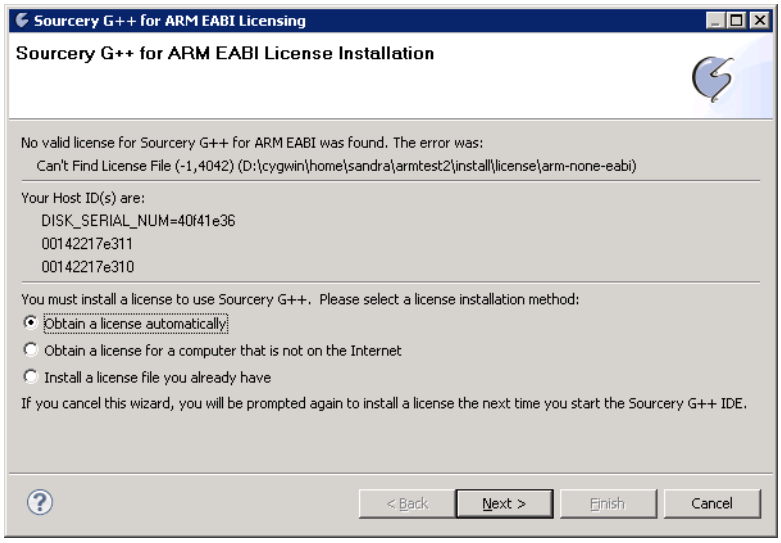
Sourcery G++ requires a license key. Each license key is associated with a particular computer, and allows you to use Sourcery G++ on that computer under the terms of your subscription and license agreement.

License Keys and the GPL

A license key is required to use the version of the GNU Compiler Collection included in Sourcery G++. Because GCC is made available under the terms of the GPL, all of the code linked into GCC is also covered by the GPL, including the code that checks for a license key. The GPL permits you to recompile the source code to remove the requirement that a license key be present. However, CodeSourcery's support covers only the original, validated GCC binaries provided with Sourcery G++.

2.8.1. Using the Licensing Wizard

When you start the Sourcery G++ IDE for the first time, it launches the Sourcery G++ Licensing wizard to guide you through the steps to install your license key.



Licensing Wizard. Use the Licensing wizard to install your license key.

The wizard displays the host ID (or host IDs) for the computer on which you have installed Sourcery G++. If your computer has more than one host ID, you must choose one of them for generating your license key.

Host IDs are either 12-digit hexadecimal numbers based on network interface addresses, or hard drive serial numbers prefixed with the text DISK_SERIAL_NUM=. You should choose a host ID

associated with a device that is unlikely to be removed from your computer. If Sourcery G++ can determine your hard drive serial number, this is usually the best choice for your host ID, since network interface addresses may change if you change your networking configuration, as on a laptop that you use with a dock or removable network card.

Computers without a Host ID

If Sourcery G++ cannot determine your network interface address or hard drive serial number, it reports that there is no host ID for your computer.

On GNU/Linux systems, the network interface address is determined by looking for the `eth0` interface. Therefore, if your network interface has a different name, the wizard is unable to determine your host ID.

If the Sourcery G++ Licensing wizard cannot determine your host ID, please contact `<support@codesourcery.com>` for assistance.

To continue with the Licensing wizard, you must select a license installation method.

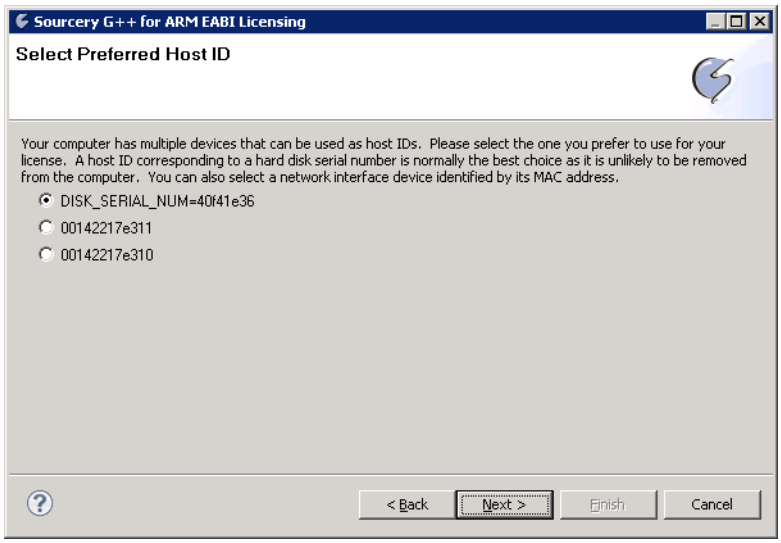
Obtain a license automatically. If you select this option, the Licensing wizard contacts the Sourcery G++ Portal to download the license key for your host. This is the easiest way to install the license for a new Sourcery G++ subscription. Refer to Section 2.8.2, "Obtaining a License Automatically" for instructions on how to continue.

Obtain a license for a computer that is not on the Internet. If your host system is not on the Internet, or if access to the Sourcery G++ Portal is blocked by a firewall, you can download a license key manually from the Portal using another computer. Refer to Section 2.8.4, "Manually Downloading Your License Key" for detailed instructions.

Install a license file you already have. You can use this method if you already have a valid license key file for your Sourcery G++ product. If you choose this installation method, refer to Section 2.8.5, "Installing a License File".

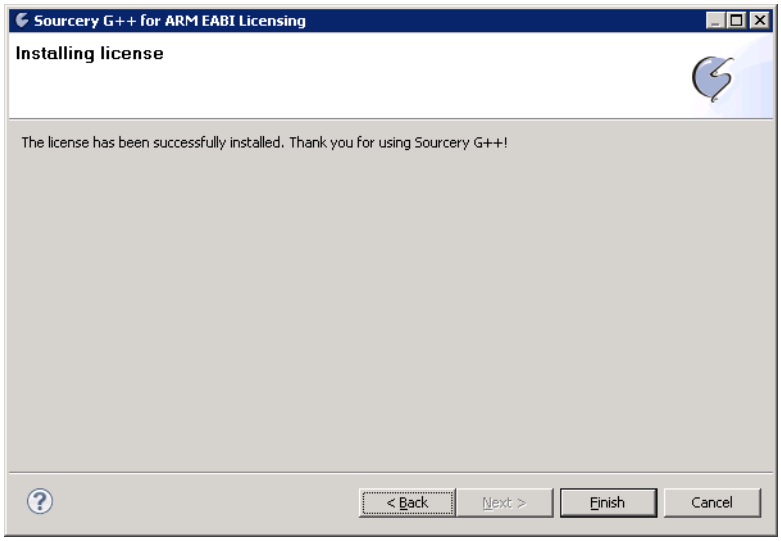
2.8.2. Obtaining a License Automatically

As mentioned in Section 2.8.1, "Using the Licensing Wizard", if more than one host ID was found for your host computer, you must choose one to associate with your license. If the Licensing wizard has found only one host ID, it uses that ID and skips the selection screen.



Select Preferred Host ID. Select the host ID to associate with your license.

The next step is to supply your login information for the Sourcery G++ Portal. When you click on `Next`, the Licensing wizard downloads and installs your license.



License Install Complete. You should see a screen similar to this when your license key has been installed successfully.

2.8.3. Configuring a Proxy Server

If the Licensing wizard is unable to connect to the Sourcery G++ Portal, this may mean access to the Portal is blocked by a firewall or VPN that requires the use of a proxy server for access to external web sites.

To set up your proxy configuration, first cancel the Licensing wizard. Then select `Window` → `Preferences` from the top menu, and then the `General` → `Network Connections` panel of the dialog. Contact your local system administrator for assistance with the proxy settings; they are typically the same as required for your regular web browser.

After you save your proxy settings, restart the Licensing wizard by selecting `Help` → `Sourcery G++ for ARM EABI Licensing` from the top menu bar.

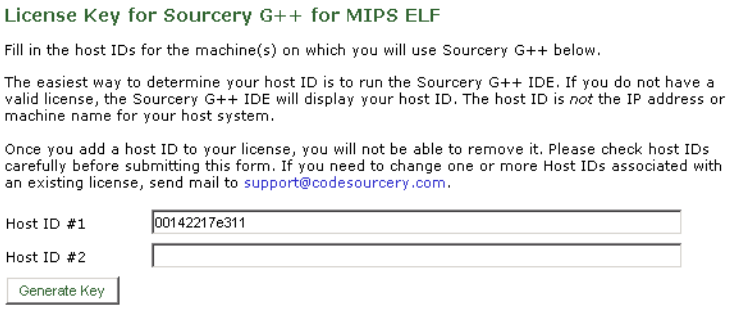
If you cannot use a proxy server to fetch your license key, you may still be able to download and install a license key manually, as described in the following sections. Otherwise, contact `<support@codesourcery.com>` for assistance.

2.8.4. Manually Downloading Your License Key

If you cannot use the automatic license download option because your host system is not on the Internet or is behind a firewall that blocks access to the Sourcery G++ Portal, you can obtain a license key file manually using another computer to access the Portal. Choosing the `Obtain a license for a computer that is not on the Internet` option in the Licensing wizard brings up a wizard page summarizing the steps to follow, which are discussed in more detail in this section.

To generate your license key, first log in to the Sourcery G++ Portal[5]. If you do not yet have an account, you must register for one at this time.

After you log in, click on the ID of the Sourcery G++ product configuration for which you need a license key. On the next screen, click on the `Generate Key` button. The Sourcery G++ Portal then displays a form:



License Key Creation. Provide your Host ID to the Sourcery G++ Portal to generate a license key.

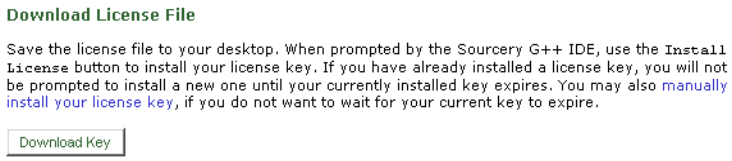
Caution

You must use one of the host ID values displayed by the Sourcery G++ IDE's Licensing wizard. If you enter an incorrect host ID, you will not be able to correct the value. Instead, you must contact CodeSourcery for assistance in regenerating your license.

Enter your host ID in the box. After checking the value, click the `Generate Key` button.

After generating your key, the Sourcery G++ Portal provides a link that you can use to download your license file:

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



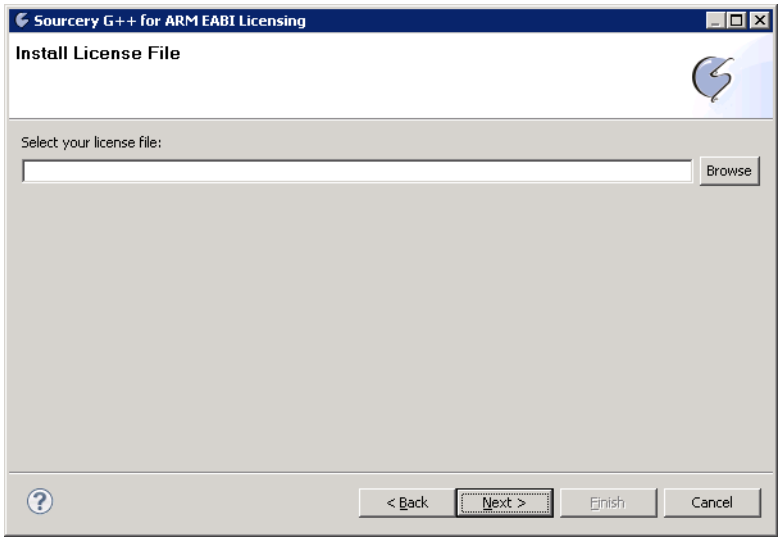
License Download. Use the button to download your license file.

Click on the `Download Key` button and save the file on your desktop or in another convenient location. Then, if you are installing Sourcery G++ on a different computer, copy the license file to that computer.

You must install your license file before you can use Sourcery G++. If you chose the `Obtain a license for a computer that is not on the Internet` option in the Licensing wizard, click on the `Next` button to continue with the `Install a license file you already have` page, as documented in the next section.

2.8.5. Installing a License File

If you choose the `Install a license file you already have` option in the Sourcery G++ Licensing wizard, the next screen allows you to select the file containing your license key. Click `Next` to install your license.



Install License File. Enter the filename to install a license from a file you already have.

2.8.6. Viewing or Reinstalling Your License Key

The Sourcery G++ Licensing wizard is normally started automatically when you run the IDE without a valid license key installed. You may also start the wizard explicitly by selecting `Help` → `Sourcery G++ for ARM EABI Licensing` from the top menu bar.

If you start the Licensing wizard when you already have a valid license key installed, the wizard displays information about your current license on the first page. You can also request or install a new license key by invoking the Licensing wizard in this way. For example, this allows you to replace

a temporary evaluation license key with a permanent one after you purchase a Sourcery G++ subscription.

2.9. Installing Add-Ons

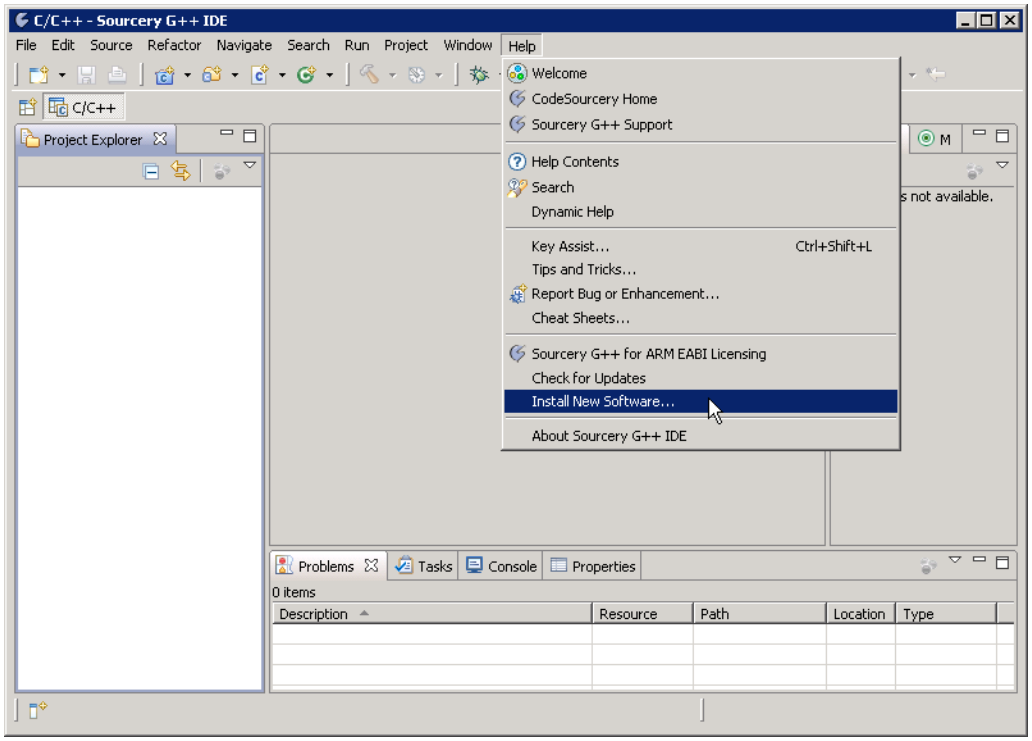
Standard and Professional Edition subscribers may download and install add-ons that provide additional features and functionality for Sourcery G++, including:

- Run-time libraries optimized for additional target configurations, as listed in Section 3.2, “Library Configurations”.
- Source code for the run-time libraries packaged for the debugger. While library source code is included in the freely available Sourcery G++ source package as well, the add-on package arranges the files for the debugger to find them and includes additional source files generated automatically during the build process. Note that source code for the proprietary CodeSourcery C Library (CSLIBC) is not included.

Add-ons are managed from the Sourcery G++ IDE. To browse for and install available add-ons for your version of Sourcery G++, start the IDE and select `Help` → `Install New Software...`.

Note about Screen Shots

The screen shots included in this section are provided for illustrative purposes only. The list of available add-ons for your version of Sourcery G++ will be different than those shown in these examples.

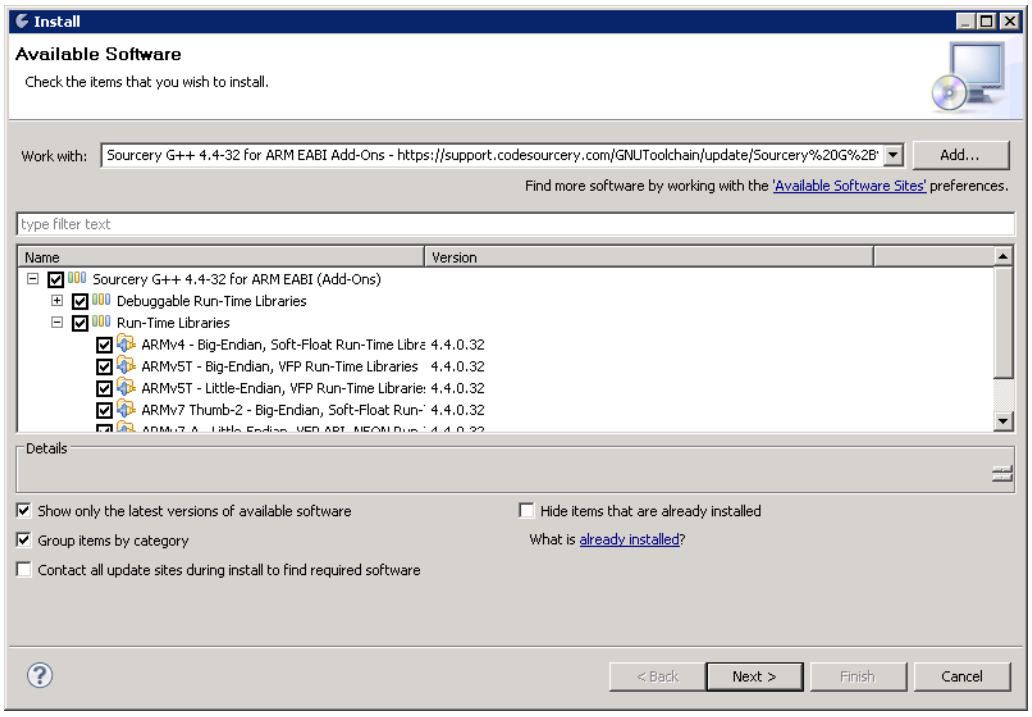


Install New Software. Add-on configurations are managed from the Sourcery G++ IDE.

In the `Work with:` drop-down, choose the `Sourcery G++` update site for your target.

At this point, the IDE prompts you to enter your username and password for the Sourcery G++ Portal, as you did when installing your license key. See Section 2.3, “Registering with the Sourcery G++ Portal”. If your computer is behind a firewall and you have problems accessing the Portal from the Sourcery G++ IDE, you may need to check your proxy settings. Refer to Section 2.8.3, “Configuring a Proxy Server” for instructions.

Scanning the update site populates the list of available add-ons. Click the checkboxes for the ones you want to install; selecting the top-level list entry selects all available Sourcery G++ add-ons.



Available Software. Choose the `Sourcery G++` update site, then check the boxes for the add-ons you want to install.

Click `Next` to continue and follow the prompts in the remaining pages of the wizard to download the add-ons. Then restart the Sourcery G++ IDE when prompted by the dialog box. At that point, installation of the add-ons is complete.

2.10. Uninstalling Sourcery G++

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

2.10.1. Using the Sourcery G++ Uninstaller on Microsoft Windows

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

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

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

2.10.2. Using the Sourcery G++ Uninstaller on GNU/Linux

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

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.

Chapter 3
Sourcery G++ for ARM EABI

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

3.1. Included Components and Features

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

Component	Version	Notes
GNU programming tools		
GNU Compiler Collection	4.5.1	Separate manual included.
GNU Binary Utilities	2.20.51	Includes assembler, linker, and other utilities. Separate manuals included.
Sourcery G++ IDE		
Eclipse IDE	Helios	See Chapter 4, “Using the Sourcery G++ IDE”. Additional documentation is available through the Eclipse online help facility.
Eclipse C/C++ Development Tools	7.0M7	
Sourcery G++ Eclipse Plugin(s)	2011.02-2	
Sourcery G++ IDE Launcher	2011.02-2	
Sourcery G++ Board Builder	2011.02-2	See Section 4.4.1, “Using the Sourcery G++ Board Builder”.
Debugging support and simulators		
GNU Debugger	7.2.50	Separate manual included.
Sourcery G++ Cygwin GDB Wrapper	2011.02-2	See Section 2.7.1.2, “Working with Cygwin”.
Sourcery G++ Debug Sprite for ARM	2011.02-2	See Chapter 7, “Sourcery G++ Debug Sprite”.
QEMU Emulator	0.11.50	See Section 5.3.1, “Connecting to the QEMU Emulator”.
Target libraries		
CodeSourcery Common Startup Code Sequence	2011.02-2	See Chapter 6, “CS3™: The CodeSourcery Common Startup Code Sequence”.
CodeSourcery C Library	2011.02-2	Separate manual included.
Library Debug Information	N/A	Available as an add-on for Standard and Professional Edition subscribers. See Section 2.9, “Installing Add-Ons”.
Other utilities		
GNU Make	N/A	Build support on Windows hosts.
GNU Core Utilities	N/A	Build support on Windows hosts.

3.2. Library Configurations

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

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

In addition to the libraries included in the base installation package, this version of Sourcery G++ supports add-on multilibs which can be downloaded separately from the Sourcery G++ Portal[1]. These additional multilibs are only available to Standard and Professional Edition subscribers. Refer to Section 2.9, "Installing Add-Ons" for information about how to download and install add-ons.

Important

If you attempt to build your application with options that correspond to one of the add-on multilibs, and you do not have that multilib installed, you will get a link error. Sourcery G++ does not automatically select another multilib from those that are available. You must either install the add-on or change your project build options to select a different multilib.

3.2.1. Base Library Configurations

The following library configurations are available in the base installation of Sourcery G++ for ARM EABI.

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

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

ARMv5TE - Little-Endian, Soft-Float	
Command-line option(s):	`-march=armv5te`
Library subdirectory:	`armv5te/`

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

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

3.2.2. Add-On Libraries

The following additional library configurations are available as add-on subpackages.

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

ARMv4 - Big-Endian, Soft-Float	
Command-line option(s):	`-mbig-endian`
Library subdirectory:	`be/`

ARMv5TE - Little-Endian, VFP	
Command-line option(s):	`-march=armv5te -mfloat-abi=softfp`
Library subdirectory:	`vfp/`

ARMv5TE - Big-Endian, VFP	
Command-line option(s):	`-march=armv5te -mfloat-abi=softfp -mbig-endian`
Library subdirectory:	`vfp-be/`

ARMv7-A - Little-Endian, VFP, NEON	
Command-line option(s):	`-march=armv7-a -mfloat-abi=softfp -mfpu=neon`
Library subdirectory:	`armv7-a-neon/`

ARMv7-A - Little-Endian, VFP ABI, NEON	
Command-line option(s):	`-march=armv7-a -mfloat-abi=hard -mfpu=neon`
Library subdirectory:	`armv7-a-hard/`
Notes:	This multilib uses an ABI that is not link-compatible with any other Sourcery G++ multilib. It is useful for some floating-point intensive applications and to interoperate with code produced by certain ARM RealView configurations.

ARMv7 Thumb-2 - Big-Endian, Soft-Float	
Command-line option(s):	`-mthumb -march=armv7 -mbig-endian`
Library subdirectory:	`thumb2-be/`

3.2.3. 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. In some cases, you may need to supply different options for linking than for compilation to select the appropriate multilib.

When you use the Sourcery G++ IDE to build your application, the target options that you set from the C/C++ Project wizard or the `Target` tab of the project properties dialog apply to both compilation and linking. (The specific options for ARM EABI targets are documented in Section 4.2.1, “Setting Up an Example Project”.) This dialog also includes an option to display the multilib selected by the other build options, for informational purposes. This option is read-only; you cannot use it to select a different multilib explicitly. If you need to adjust the compiler and linker command-line options separately to select some other multilib, see Section 4.2.5, “Customizing Build Actions”.

When invoked from the command line, GCC can tell you which multilib corresponds to a given set of link options if you add the `-print-multi-directory` option to your other command-line options. For example:

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

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

3.3. CodeSourcery C Library

Sourcery G++ for ARM EABI includes the CodeSourcery C Library (CSLIBC). This library is specially optimized for smaller code size on embedded targets. On ARM targets, commonly-used functions such as `memcpy` have been hand-optimized for better performance.

CSLIBC is an exclusive feature of CodeSourcery's subscription (non-Lite) toolchains and is licensed as a Redistributable Component. Refer to Appendix B, “Sourcery G++ Licenses” for more information about Sourcery G++ licenses.

3.4. Using Sourcery G++ with Kinetis Boards

This section covers special features of Sourcery G++ to support debugging applications on Kinetis boards.

3.4.1. Using the Sourcery G++ Debug Sprite with Kinetis Boards

The Sourcery G++ Debug Sprite allows you to run and debug applications on a Freescale Tower System board using its built-in P&E OSJTAG interface; you do not need a separate ICE device. The Sprite also supports P&E Cyclone MAX devices with Kinetis targets. For instructions on connecting your board and installing the P&E drivers, refer to Section 7.11, “P&E Devices”.

You can also use the Sourcery G++ Debug Sprite with SEGGER J-Link devices to debug applications on Kinetis boards. Refer to Section 7.10, “SEGGER J-Link Devices” for set-up instructions.

The Debug Sprite is fully integrated with the Sourcery G++ IDE. You can learn how to use the IDE debugger by following the tutorial provided in Section 4.3, “Debugging Applications”. For command-line use, see the Debug Sprite reference material in Chapter 7, “Sourcery G++ Debug Sprite”.

Peripheral register browsing is supported via the Debug Sprite on Kinetis targets. See Section 3.7, “Peripheral Register Browsing” for more information about this feature.

The Debug Sprite disables the watchdog timer so that it won't trigger while you are debugging your application. When you design a production application, be sure to disable or service the watchdog timer as you see fit.

3.4.2. Using the Kinetis Example Programs

Sourcery G++ includes a set of example programs and libraries for using the peripheral devices on Kinetis boards. These examples have been bundled so that you can import them directly into the Sourcery G++ IDE.

Select `File` → `Import...` from the top menu, then expand `Sourcery G++` and choose `Kinetis Project`. This brings up the Kinetis Import wizard. To import all the example programs for your board, select the file `share/sourceryg++-arm-none-eabi-examples/Kinetis/`

boards/*board*/*board*-examples.sgxw within your Sourcery G++ installation directory, where *board* matches your board. For example, if your board is a Freescale TWR-K60N512 Development Kit, choose boards/TWR-K60N512/TWR-K60N512-examples.sgxw. You may also import a single example program by selecting boards/*board*/*example*/*example*.sgxx, where *example* is the example program to import. Then click Finish.

The Import wizard copies the Kinetis example files into your workspace, creating separate projects for each sample program as well as for the libraries and shared header files. If you import a single example, required libraries are also imported automatically. The sample programs are pre-configured for your selected board. You may build, run, and modify these applications from the IDE as you would any other project.

3.5. Using Sourcery G++ with Stellaris Boards

This section covers special features of Sourcery G++ to support debugging applications on Stellaris boards.

3.5.1. Using the Sourcery G++ Debug Sprite with Stellaris Boards

The Sourcery G++ Debug Sprite allows you to run and debug applications on a Stellaris board using its built-in FTDI ARMUSB interface; you do not need a separate ICE device. Most Stellaris evaluation kit boards also include an In-Circuit Debug Interface (ICDI) feature that allows them to be used to debug production systems containing other Stellaris microcontrollers.

For instructions on connecting your board and installing ARMUSB drivers on your host system, refer to Section 7.5.1, “ARMUSB Configuration and Drivers”.

The Debug Sprite is fully integrated with the Sourcery G++ IDE. You can learn how to use the IDE debugger by following the tutorial provided in Section 4.3, “Debugging Applications”. For command-line use, see the Debug Sprite reference material in Chapter 7, “Sourcery G++ Debug Sprite”.

The Debug Sprite supports peripheral register browsing on Stellaris targets. See Section 3.7, “Peripheral Register Browsing” for more information about this feature.

3.5.2. Using StellarisWare with Sourcery G++

StellarisWare is a software package that includes libraries such as the Stellaris Peripheral Driver Library along with a set of sample programs that demonstrate the use of these libraries. StellarisWare is provided by Luminary Micro and is not part of Sourcery G++, but it is bundled in such a way that you can import StellarisWare projects directly into the Sourcery G++ IDE.

Sourcery G++ includes a copy of StellarisWare, located in share/sourceryg++-arm-none-eabi-examples/StellarisWare. If you need a different release of StellarisWare, packages are also included on a CD with Stellaris development kits and available for download from the Luminary Micro Software Updates[2] web site. To use a different release of StellarisWare, first unpack it on your local machine. The downloads are packaged as self-extracting archives for Windows hosts; if you are using a Linux host, you can unpack the .exe file with unzip.

Then start the Sourcery G++ IDE. Select File → Import... from the top menu, then expand Sourcery G++ and choose StellarisWare Project. This brings up the StellarisWare Import wizard. To import all the example programs for your board, select the file boards/*board*/*board*.sgxw within your StellarisWare installation directory, where *board* matches

[2] http://www.luminarymicro.com/products/software_updates.html

your board. For example, if your board is a DK-LM3S9B96 Development Kit, choose `boards/dk-lm3s9b96/dk-lm3s9b96.sgxw`. You may also import a single example program by selecting `boards/`*`board/example/example`*`.sgxx`, where *`example`* is the example program to import. Then click `Finish`.

The Import wizard copies the StellarisWare files into your workspace, creating separate projects for each sample program as well as for the libraries and shared header files. If you import a single example, required libraries are also imported automatically. The sample programs are pre-configured for your selected board. Each project also includes an automatically-generated debug launch configuration using your board's ARMUSB debug device. You may build, run, and modify these applications from the IDE as you would any other project.

3.6. Using Sourcery G++ with STM32 Boards

Sourcery G++ includes special features to support application development and debugging on STMicroelectronics STM32 boards. Refer to Chapter 6, "CS3™: The CodeSourcery Common Startup Code Sequence" for specific boards supported by CS3.

STM32 boards are supported by the Sourcery G++ Debug Sprite using third-party JTAG probes. When used with the Sprite, CS3 provides access to peripheral registers in the debugger. See Section 3.7, "Peripheral Register Browsing" for more information about using this feature.

Sourcery G++ also includes a set of example programs and libraries for using the peripheral devices on STM32 boards. These examples are provided by STMicroelectronics as part of the STM32 Standard Peripheral Library[3] but have been bundled so that you can import them directly into the Sourcery G++ IDE.

Select `File` → `Import...` from the top menu, then expand `Sourcery G++` and choose `STM32 Project`. This brings up the STM32 Import wizard. To import all the example programs for your board, select the file `share/sourceryg++-arm-none-eabi-examples/stm32/`*`board/board`*`.sgxw` within your Sourcery G++ installation directory, where *`board`* matches your board. For example, if your board is a STM3210E-EVAL, choose `STM3210E-EVAL/STM3210E-EVAL.sgxw`. You may also import a single example program or library by selecting a `.sgxx` file within the directory for your board. Then click `Finish`.

The Import wizard copies the STM32 example files into your workspace, creating separate projects for each sample program as well as for the libraries and shared header files. If you import a single example, required libraries are also imported automatically. The sample programs are pre-configured for your selected board. Each project also includes an automatically-generated debug launch configuration, pre-configured for a SEGGER J-Link debug device. You may build, run, and modify these applications from the IDE as you would any other project.

3.7. Peripheral Register Browsing

When you use the Sourcery G++ Debug Sprite to debug a program running on a supported board, you can browse and modify the memory-mapped registers on the board as well as the core ARM registers. Support for this feature is provided on a per-board basis in the Sprite's board configuration files. For example, Sourcery G++ includes register browsing data for all Stellaris boards otherwise supported by the Sprite. You can find a complete list of boards that support this feature in Section 7.13, "Supported Board Files".

[3] http://www.st.com/mcu/inchtml.php?fdir=pages&fnam=stm32lib

Enhanced register browsing functionality is available both when using the Sourcery G++ IDE, and when using the command-line debugger. Using the IDE is the recommended method.

In the Sourcery G++ IDE, the list of registers is automatically shown in the `Registers` tab, typically located at the right top of the Sourcery G++ IDE window. The register tab includes two top-level items: `Main` and `$io`. The former contains the core ARM registers and the latter contains memory-mapped registers. The memory-mapped registers are arranged in a tree according to the functional unit they belong to, and register groups inside that functional unit. For example, on Stellaris boards the `UART` group contains the `UART0` group which in turn contains the `UARTDR` register. Many registers further contain individual fields. The fields are shown as children of the containing register.

Most registers are displayed in hexadecimal format by default. To view a register in decimal format, right click on the register, select the `Format` menu item, and then select `Decimal`.

Values of registers can be modified by right clicking on a register, and selecting the `Change Value` command. For registers with fields, you can modify the values of the fields, and the register itself cannot be modified.

Every time program execution is suspended—such as after stepping to the next statement, or stopping on a breakpoint—the new values of all visible registers are read from the processor. Registers that have changed value are highlighted in yellow. Reading the values of all registers can take considerable time. To speed up debugging, close the register groups you are not interested in. For example, if you close the `$io` group, no memory-mapped registers are read when you step through the program. When you open a group again, new values are read automatically.

Several registers are *read-sensitive*—reads from them have side effects. For example, interrupt flags are often cleared on read. The Sourcery G++ IDE reads such registers only by your explicit request. Initially, no values are shown for such registers at all. To read and display the value, right click on the register and select the `Fetch Value` command. The value of the register is not automatically updated as you step though the program. You should use the `Fetch Value` each time you want to read and display the current value of a read-sensitive register.

On many embedded devices, most peripherals are initially powered off. In this state, their registers cannot be read by the debugger, and the values appear blank or zero in the registers view. Your application should turn on the peripherals it uses, which makes the corresponding register values available in the debugger. You may also turn peripherals on or off from the debugger by adjusting the control registers. For example, on Stellaris boards peripherals other than `SYSCTL` (system control) are initially powered off. You can turn on the `UART0` peripheral by setting the corresponding bit in the `RCGC1` register in the `SYSCTL` group.

When debugging from the GDB command line, the values of the core registers can be listed using the following command:

```
(gdb) info registers
```

The value of an individual register can be printed using a command of the form:

```
(gdb) print $register-name
```

A register value can be modified using a command of the form:

```
(gdb) set variable $register-name = new-value
```

The values of all memory-mapped registers can be printed using the following command:

```
(gdb) print $io
```

The values are printed as a C structure, where top-level fields are register groups, containing registers and further groups.

Warning

This command reads and displays all registers, including read-sensitive registers.

Because the above command reads read-sensitive registers, and because the amount of output is large, it usually better to print values of individual registers. First, you need to find the register you're interested in. If you issue the command:

```
(gdb) ptype $io
```

you'll see the list of top-level register groups and registers. Each top-level register group can be examined using a command of the form:

```
(gdb) ptype $io.group-name
```

You can examine the entire hierarchy of memory-mapped registers and locate the register you're looking for by applying the `ptype` command recursively. After that, the value of the register can be printed using a command of the form:

```
(gdb) print $io.register-name
```

For example:

```
(gdb) print $io.UART.UART0.UARTDR
```

The value of a register can be modified using the following command:

```
(gdb) set variable $io.register-name = new-value
```

3.8. Using Flash Memory

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

1. You must use an appropriate linker script that identifies the ROM memory region on your target board, and locates the program text within that region. Refer to Chapter 6, “CS3™: The CodeSourcery Common Startup Code Sequence” for information about the boards supported by Sourcery G++.

2. Next, load your program into the flash memory on your target board. When you use the Sourcery G++ Debug Sprite to debug your program, flash programming happens transparently as part of the normal debugger operation of loading your program onto the target. However, this is not supported if you are using some other debug stub to connect to your target board. You must instead use third-party tools to program the flash memory.

   Since the Sourcery G++ IDE normally loads the program onto the target automatically as part of its debugger startup operations, you must disable this if you want to debug a program that you have already flashed onto the target using an external tool. Uncheck the `Automatically download program` box on the `Startup` subtab in the debugger dialog. See Section 4.3.3.1, “Debugger Startup”.

3. Finally, when debugging a program in flash memory, GDB must be told about the ROM region so that it knows where it must use hardware breakpoints to control program execution. If you are

using the Sourcery G++ Debug Sprite to debug your program, the Sprite does this automatically, using the memory map provided in the board configuration file. Otherwise, you must provide this information explicitly.

In the Sourcery G++ IDE, you can specify the read-only memory region on the `Memory Map` subtab in the debugger configuration dialog. Refer to Section 4.3.3.2, "Configuring the Memory Map" for more information.

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

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

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

Although GDB automatically attempts to use hardware breakpoints on code locations in the read-only memory region, on many targets the number of available hardware breakpoints is very small. Furthermore, GDB also uses hardware breakpoints internally to implement commands such as `step`, `next`, and `finish`. Thus the number of breakpoints you can explicitly set in ROM may be fewer than the number supported by the target system.

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

3.9. Using VFP Floating Point

3.9.1. Enabling Hardware Floating Point

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

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

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

Sourcery G++ for ARM EABI includes libraries built with software floating point, which are compatible with VFP code compiled using the soft-float ABI. VFP hard-float libraries built with both the soft-float and VFP ABIs are available as add-ons to Sourcery G++ Standard and Professional

Edition subscribers. Selecting the VFP ABI for compilation is only useful if you have appropriate runtime libraries installed. Currently, Sourcery G++ provides VFP ABI libraries for ARMv7-A targets in little-endian mode only. Refer to Section 3.2, “Library Configurations” for a complete list of available libraries for ARM EABI.

Note that, in addition to selecting hard/soft float and the ABI via the `-mfloat-abi` option, you can also compile for a particular FPU using the `-mfpu` option. For example, `-mfpu=neon` selects VFPv3 with NEON coprocessor extensions.

3.9.2. NEON SIMD Code

Sourcery 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.

3.9.3. Half-Precision Floating Point

Sourcery G++ for ARM EABI includes support for half-precision (16-bit) floating point, including the new `__fp16` data type in C and C++, support for generating conversion instructions when compiling for processors that support them, and library functions for use in other cases. The included QEMU emulator also supports the hardware instructions when invoked with the `any` CPU specifier.

To use half-precision floating point, you must explicitly enable it via the `-mfp16-format` command-line option to the compiler. For more information about `__fp16` representations and usage from C and C++, refer to the GCC manual.

3.10. ABI Compatibility

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

Sourcery G++ implements the ABI as described in these documents, which are available from the ARM Information Center[4]:

- BSABI - ARM IHI 0036B (28 October 2009)
- BPABI - ARM IHI 0037B (28 October 2009)
- EHABI - ARM IHI 0038A (28 October 2009)
- CLIBABI - ARM IHI 0039B (4 November 2009)

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

- AADWARF - ARM IHI 0040A (28 October 2009)
- CPPABI - ARM IHI 0041C (5 October 2009)
- AAPCS - ARM IHI 0042D (16 October 2009)
- RTABI - ARM IHI 0043C (19 October 2009)
- AAELF - ARM IHI 0044D (28 October 2009)
- ABI Addenda - ARM IHI 0045C (4 November 2009)

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

3.11. ARM Profiling Implementation

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

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

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

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

- Preserve all other register state except for `r12` and the CPSR condition code bits. In particular all coprocessor state and registers `r0`-`r3` must be preserved.
- Record and count all occurrences of the function calls in the program. The caller can be determined from the `lr` value stored on the top of the stack (on entry to `__gnu_mcount_nc`), and the callee can be determined from the current value of the `lr` register (i.e. the caller of this function).
- Arrange for the data to be saved to a file named `gmon.out` when the program exits (via `atexit`). Refer to the `gprof` profiler manual for more information.

3.12. Object File Portability

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

To use this feature, when compiling your files with the bare-metal ARM EABI toolchain define the preprocessor constant `_AEABI_PORTABILITY_LEVEL` to 1 before including any system header

files. For example, pass the option `-D_AEABI_PORTABILITY_LEVEL=1` on your compilation command line. No special options are required when linking the resulting object files. When building applications for ARM EABI, files compiled with this definition may be linked freely with those compiled without it.

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

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

Chapter 4
Using the Sourcery G++ IDE

This chapter explains how to use the Sourcery G++ IDE to build a C or C++ application. This chapter assumes you have installed Sourcery G++ as described in Chapter 2, “Installation and Configuration”. If you prefer to use the command line to build your applications, you may refer to Chapter 5, “Using Sourcery G++ from the Command Line” instead.

4.1. Overview

This chapter explains how to create, modify, and debug a program using the Sourcery G++ IDE. After working through the example program in this chapter, you can use the same techniques to create your own programs.

To start the Sourcery G++ IDE, use the launcher program which can be found in the `bin/` subdirectory of your Sourcery G++ installation. On Windows hosts, the launcher is called `sourcerygxx-ide.exe`. On Linux hosts, the launcher is called `sourcerygxx-ide`. Alternatively, if you installed Sourcery G++ with the graphical installer and specified a shortcut location, you can run the IDE using that shortcut.

When you start the IDE for the first time, it prompts you to select a *workspace* directory; this is where your program source files, compiled binaries, and other files managed by the IDE will be stored. Next, it starts the Sourcery G++ Licensing wizard to guide you through installation of your license key. Refer to Section 2.8, “License Keys” for more information. The IDE also displays a welcome screen with links you can click on to get more information about using Sourcery G++. When you are ready to begin using Sourcery G++, first close the welcome screen tab. You can return to the welcome screen again later, if you wish, from the `Help` menu.

Learning More About Eclipse

The Sourcery G++ IDE is based on Eclipse. While this chapter explains how to accomplish basic tasks using the Sourcery G++ IDE, it is not a comprehensive reference manual. If you want to learn more about Eclipse, use the online help from the welcome screen or as described in Chapter 8, “Next Steps with Sourcery G++”. You can also visit the Eclipse web site[1] for additional tutorials and documentation.

Note About Screen Shots

The screen shots included in this chapter are provided for illustrative purposes only. You may see slightly different sets of menu options when you run the IDE, corresponding to the specific features included in your version of Sourcery G++. Refer to the text for information about the particular targets and debugging options supported by Sourcery G++ for ARM EABI.

The remainder of this chapter is divided into three sections. The first guides you through the process of creating and building an example program; the second section shows how to debug and run the program once it has been built. The final section covers advanced features of the Sourcery G++ IDE.

4.2. Building Applications

In the Sourcery G++ IDE, every program is a *project*. The project contains all of the source files required to build the program. So, the first step is to create a project.

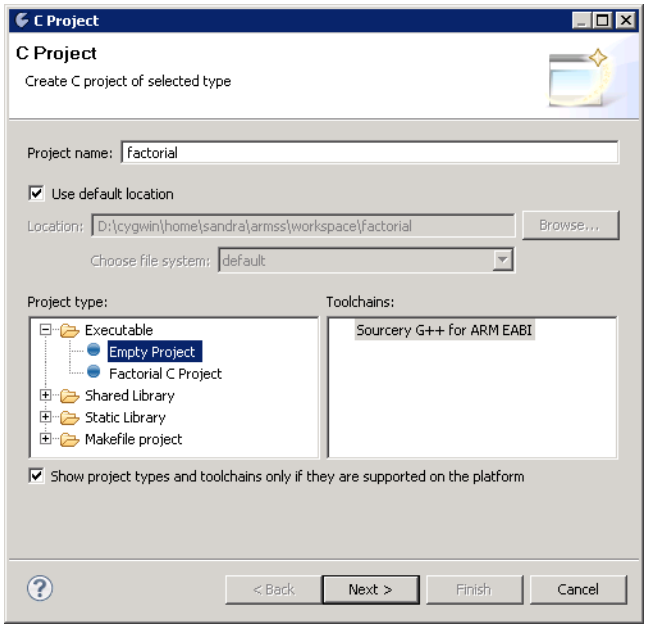
Normally, the IDE manages building your project for you. This is convenient if you intend to do all of your development from within the IDE. However, if you are working with code that has previously been built with `make`, you may wish to use a Makefile project instead. The following several sections explain how to create and work with a project using the IDE's managed build support. For instructions on creating a Makefile project, refer to Section 4.4.2, “Makefile Projects”.

[1] http://www.eclipse.org

4.2.1. Setting Up an Example Project

Create a new project by selecting `File` → `New` → `C Project`. This opens the new project wizard. (If you want to build a C++ application, select `C++ Project` instead.) Then click `Next`.

Give the project the name `factorial`. From the `Project types` menu, expand `Executable`. Select `Empty Project` to begin a new empty project. (Selecting `Factorial C Project` creates a project pre-populated with the example used in this tutorial.) On the `Toolchain` menu ensure that `Sourcery G++ for ARM EABI` is selected. Then click the `Next` button.



Creating an Executable Project. Use the C Project wizard to create a new empty Executable project using the Sourcery G++ toolchain for your target.

The next page of the wizard allows you to customize the project build settings to match your target board or processor. For ELF and EABI targets, you must choose a target board before you can build your application. This is necessary to specify an appropriate memory map, system startup code, and hosting (I/O) support for your target. Refer to Chapter 6, "CS3™: The CodeSourcery Common Startup Code Sequence" for detailed information about the boards supported by Sourcery G++. You can add your own CS3 support for custom boards using the Sourcery G++ Board Builder; see Section 4.4.1, "Using the Sourcery G++ Board Builder" for details. If you have no target board, you can choose a simulator as your target instead.

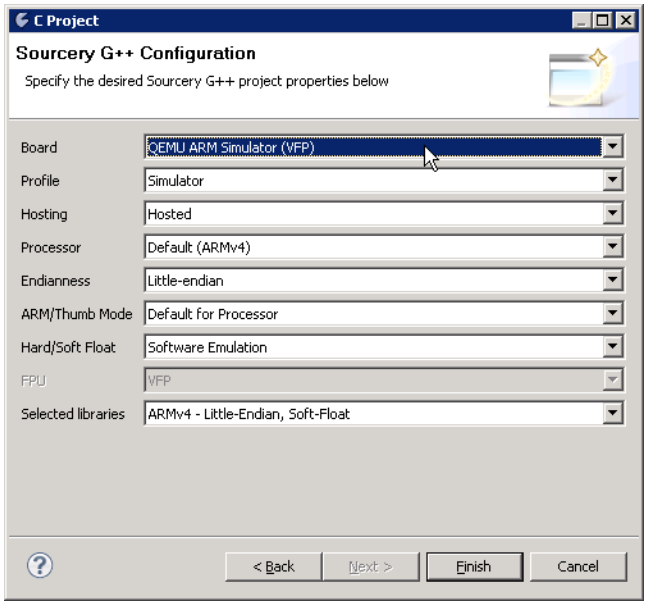
In most cases, selecting a board automatically chooses the build settings for the processor on that board, but some boards (such as simulator targets) are "generic" and can be used with different processors. In this case you should choose the processor you wish to target.

The following properties can be set on this page:

- `Board`: Select the target board. This controls the memory map and initialization code, and may also select a processor. You must select a value for this option.
- `Profile`: Select the linker profile or memory region where the program will be loaded.
- `Hosting`: Select the hosting mode for the program.

- `Processor`: Select the target processor. This controls code generation and library selection.
- `Endianness`: Choose little-endian or big-endian mode.
- `ARM/Thumb Mode`: Select whether to generate ARM or Thumb code.
- `Hard/Soft Float`: Select hardware or software floating-point support and ABI.
- `FPU`: Choose floating-point unit for code generation.

The project properties dialog also includes an option that displays the name of the multilib selected by your other project build settings. This option is informational only; you cannot use it to explicitly select a different multilib. This option additionally indicates when you have selected an add-on multilib that is not installed. In this case you must either install the add-on or change your build options to select a different set of libraries before you can link your project. Refer to Section 3.2, "Library Configurations" for further discussion of how libraries are selected.

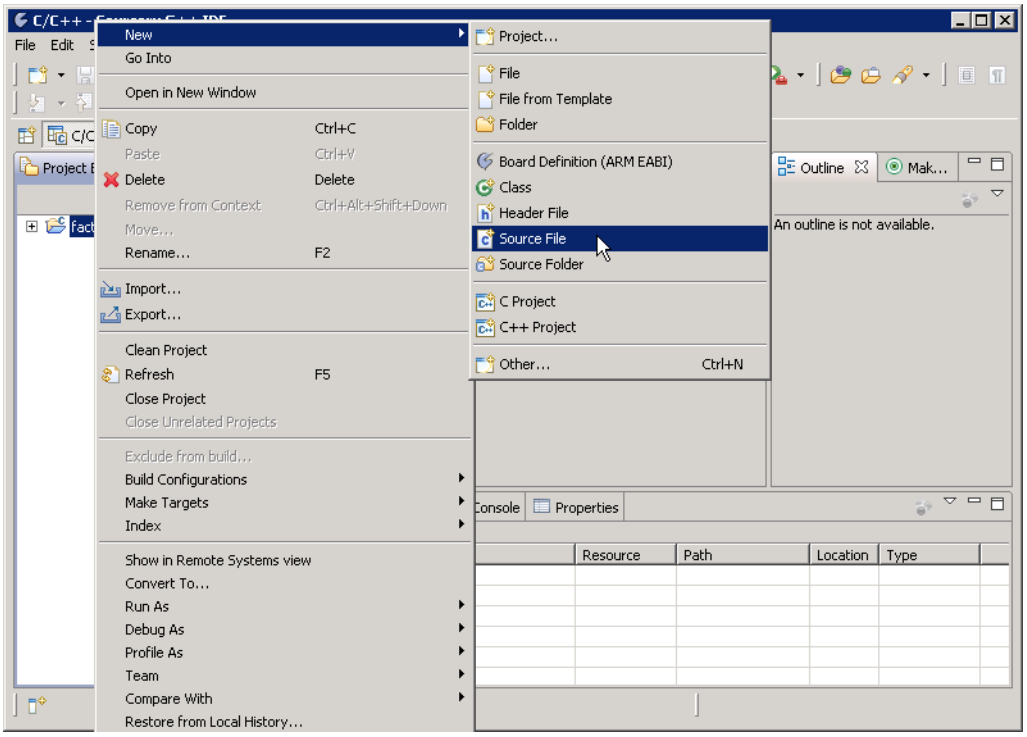


Setting Build Properties during Project Creation. You can choose common build properties for your project from the new project wizard. The exact set of properties available depends on the target system.

Select `Finish` to create the project. If you are asked whether to open a new perspective, click the `Yes` button.

4.2.2. Writing Source Code

At this point, the project exists, but there is no associated source code. So, the next step is to create the main program. Right-click on the `factorial` project, and select `New` → `Source File`. Give the new file the name `main.c` and click the `Finish` button.



Adding a Source File. Right-click on the project name to add a new source file.

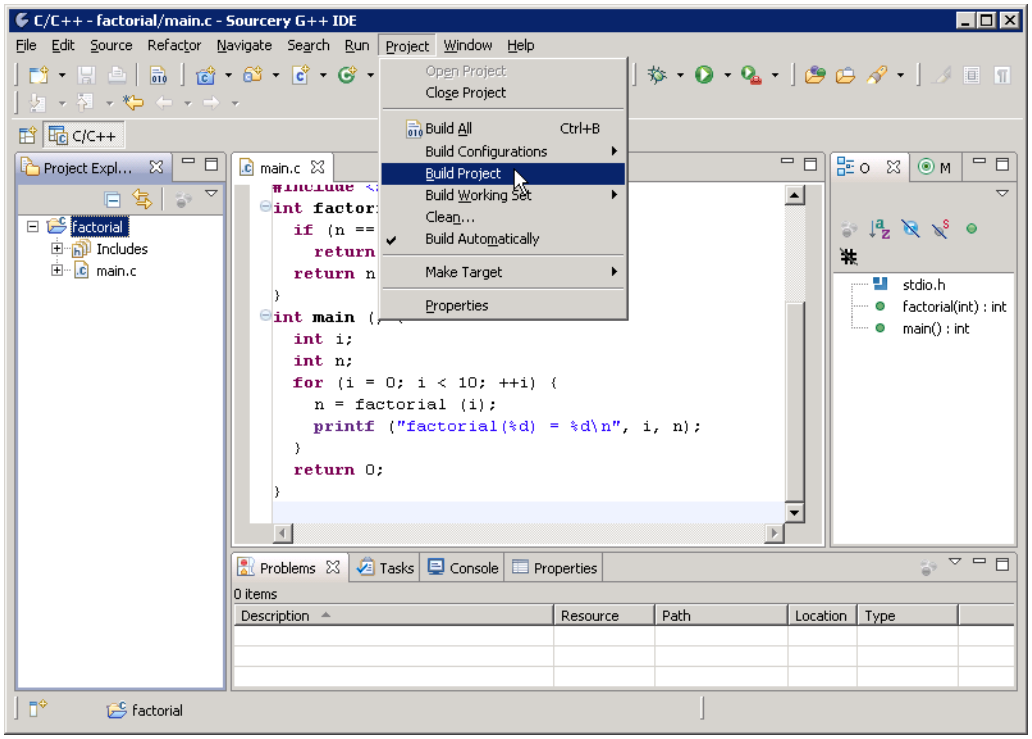
The Sourcery G++ IDE now displays an editing window for you to use to create the program. Type (or cut-and-paste) the following program into the editor:

```
#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;
}
```

When you are done, save the file with `File` → `Save` (**Ctrl+S**).

After you save the file, build your project by selecting it in the `Project Explorer` pane on the left, then going to the `Project` menu and selecting `Build Project`. The output of the commands run by the IDE is displayed in the `Console` tab in the lower pane. If the project build is successful,

the IDE prints statistics about the code size of the `factorial` executable at the bottom of the console.



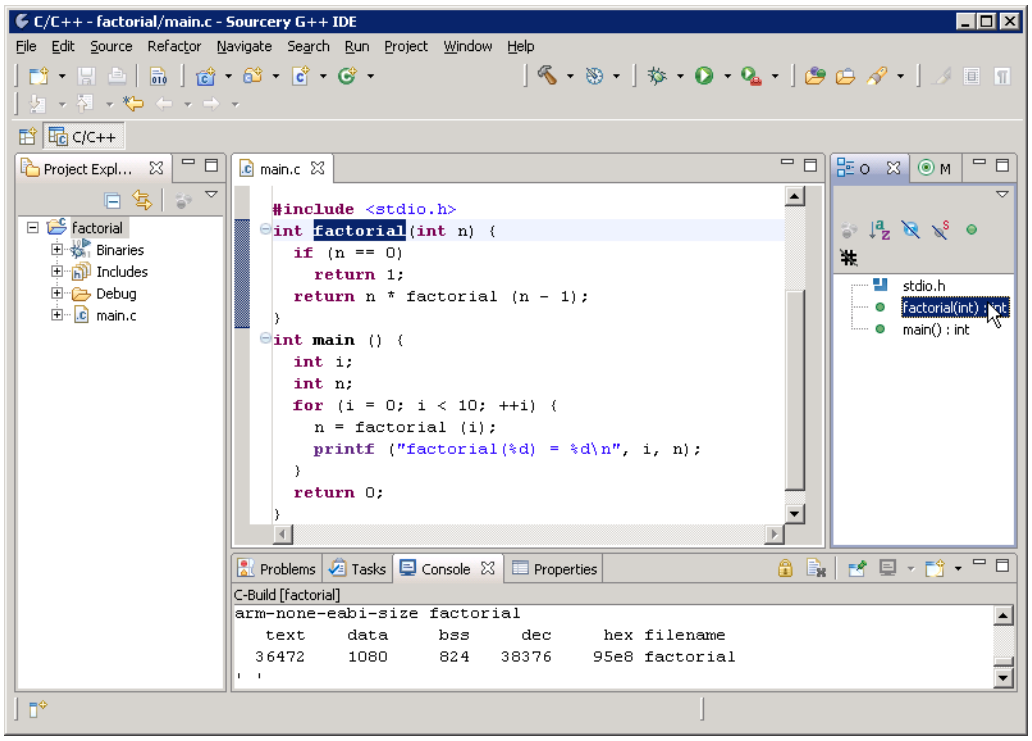
Building the Project. Use `Build Project` to build the project after saving it.

Building Automatically

If you want the IDE to build your project to automatically when you add or save files, in addition to selecting `Build Automatically` from the `Project` menu, you also need to enable the setting on a per-project basis. From the `Project` menu, select `Properties`, then `C/C++ Build`. Open the `Behaviour` tab and check `Build on resource save (Auto build)`.

4.2.3. Using Cross-Reference Information

Whenever it rebuilds your project, the Sourcery G++ IDE also computes cross-reference information. You can see some of this information in the `Outline` pane. In particular, each of the two functions in the program (`factorial` and `main`) are shown in the `Outline` pane. When you click on name of a function or variable in the `Outline` pane, the IDE repositions the cursor to show you that entity.



Using the Outline. Click a function name in the `Outline` pane to jump to it in the editor.

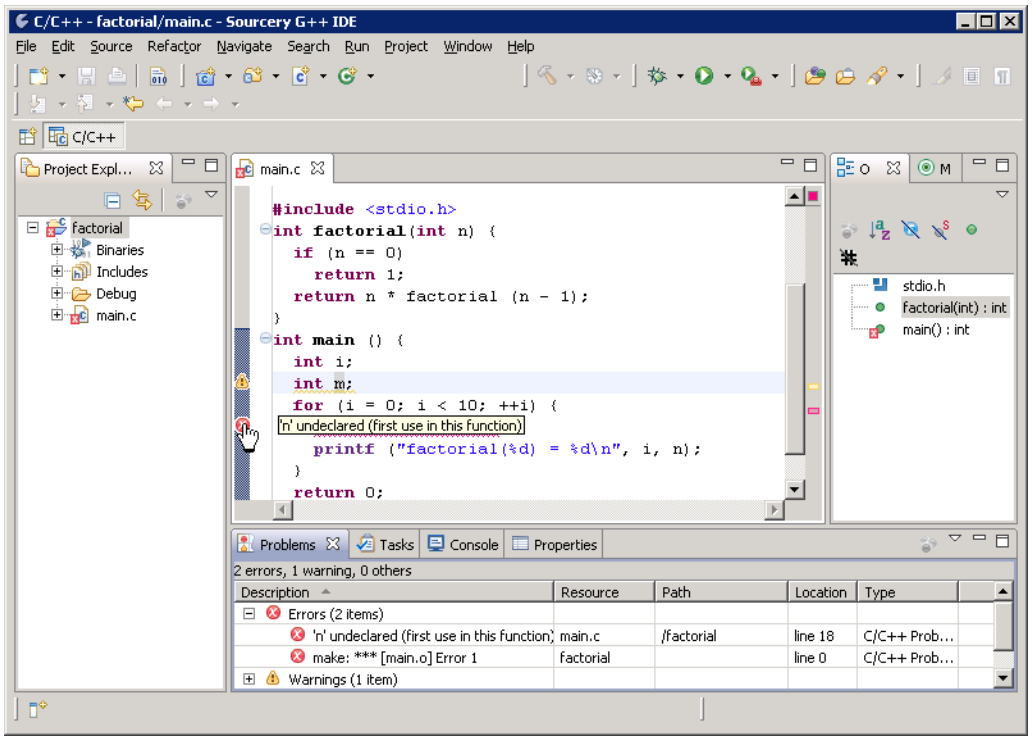
You can also use the cross-reference information to jump from a reference to a function or variable to its definition. For example, find the line in `main` that calls `factorial` and place the cursor over the name `factorial`. This pops up a small box showing the definition of the function. You can right-click and select `Open Declaration` (**F3**) to jump to the definition of `factorial` in the editor. The cross-reference functionality works even if the function call is in a different file from the declaration of the function.

4.2.4. Dealing with Errors

If you pasted the sample application into the IDE, the program probably compiled correctly the first time. But, of course, that rarely happens when writing a large program from scratch. To see how the Sourcery G++ IDE deals with errors, you can intentionally introduce an error.

Change the declaration of `n` in `main` to declare `m`, instead of `n`, and save the file. This change makes the program invalid because there are references to `n` in the function, but no declaration. In addition, the new variable `m` is not serving any useful purpose (since there are no references to it). Sourcery G++ informs you of both issues by flagging the problematic lines of source code.

The IDE places a circular red symbol next to lines that cause errors and a triangular yellow symbol on lines that cause warnings. There are several ways to get more detailed information about the problems. One way is to click on the `Problems` pane at the bottom of the IDE. This pane shows the error and/or warning messages issued by the compiler. Also, when you place the cursor over the error indicators, the IDE displays the error message.



Viewing Errors. Place the cursor over a warning or error indicator to see the cause of the problem.

Correct the error by changing `m` back to `n`, and then rebuild the project. The IDE removes the error indicators and the `Problems` pane is cleared, indicating a successful build.

4.2.5. Customizing Build Actions

If you wish, you can customize the actions the Sourcery G++ IDE uses to build your project. To do this, pull up the project properties dialog by right-clicking on your project name in the `Project Explorer` pane, and selecting `Properties`. Then expand `C/C++ Build` and select `Settings`.

You can adjust the options for invoking the compiler, assembler, and linker from the `Tool Settings` tab. The project properties initially set in the C or C++ Project wizards (Section 4.2.1, “Setting Up an Example Project”) can be adjusted by selecting the `Target` category. The other categories allow you to configure other options specific to each tool. There are dialog boxes for the most common types of options, or you can specify arbitrary command-line options by selecting the `Miscellaneous` category for each tool.

For example, if you wish to build for a processor that is supported by the compiler but not listed in the `Processor` option in the `Target` category, you can select `Other` as the `Processor`, and then enter the appropriate command-line options directly in the `Miscellaneous` categories for the compiler, assembler, and linker. Similarly, to use a custom linker script, select `Other` from the `Board` option, then provide the appropriate `-T` option in the `Miscellaneous` panel for the linker.

Use the Objcopy tool if you want to automatically translate the output file from the linker into another format, such as Intel HEX or S-record. You can select from several common file formats or use `Other` to specify any other format known to Objcopy on ARM EABI targets. The Objcopy step in the build process is optional, and is only run if you explicitly enable it in your project properties.

You can also specify additional pre-build and post-build steps from the `Build steps` tab. For example, to automatically produce a disassembly listing of the executable after linking, enter the command

```
${cs_target}-objdump -ldr ${BuildArtifactFileName} \
    > ${BuildArtifactFileName}-asm.txt
```

in the `Command` field for `Post-build steps`.

In this example, `cs_target` and `BuildArtifactFileName` are predefined *build macros* that expand to the name of the target prefix for Sourcery G++ commands (`arm-none-eabi`, in the case of ARM EABI), and the output filename from the build process, respectively. You can browse the list of available build macros and define new ones by selecting `Variables` in the left-hand pane of the project properties dialog.

4.3. Debugging Applications

4.3.1. Starting the Debugger

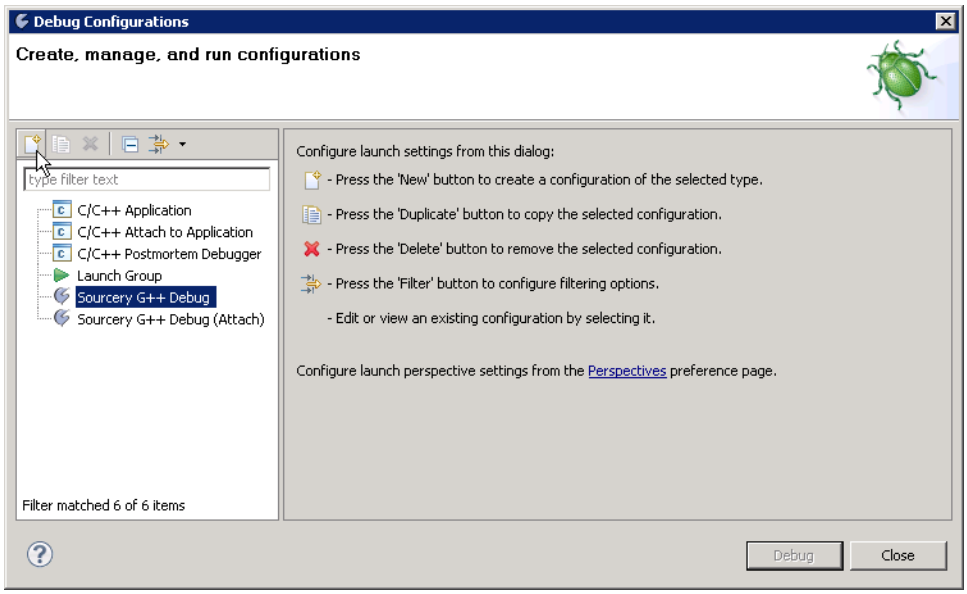
After you build your application, select it in the left-hand `Project Explorer` pane. Then choose `Run` → `Debug Configurations...` from the menu bar. This opens the dialog for creating and editing debug launch configurations.

The pane on the left lists the available debug launch configuration types. Sourcery G++ for ARM EABI provides these custom launch types:

Sourcery G++ Debug. Use this launch configuration type to run an application on the target. This type of launch initializes the target and loads your application onto it. This is the most typical launch type for application program debugging.

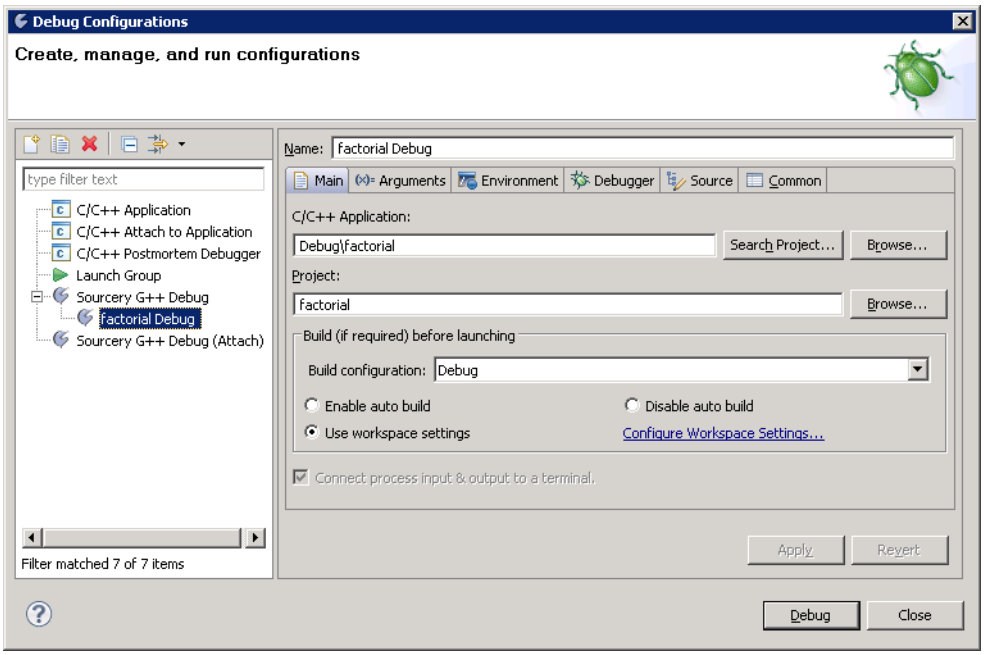
Sourcery G++ Debug (Attach). Use this launch configuration type to attach the debugger to an application which has already been started on the target.

Select the `Sourcery G++ Debug` launch type in the left-hand pane. Then, click the `New` icon positioned towards the upper left of the window.



Creating a Debug Configuration. Select `Sourcery G++ Debug` and click the `New` icon to create to create a new debug configuration.

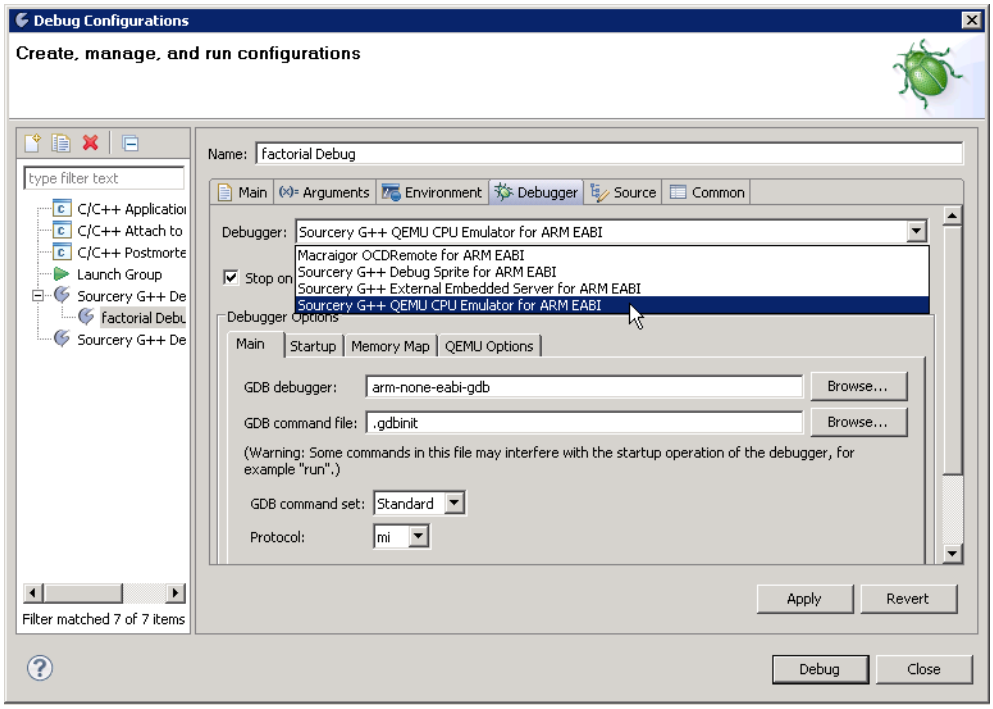
When you create the launch configuration, a new pane appears on the right. On the `Main` tab, use the `Browse...` button to select your project, if it is not already selected. Then, use the `Search Project...` or `Browse...` buttons to select your application.



Selecting a Program. Use the `Search Project...` button to locate your program.

Next, switch to the `Debugger` tab and select the debugging mode you want to use. The different debugging modes are discussed in detail below; the choices depend on which debug launch config-

uration type you have selected. Some debugging modes require you to configure additional options. When you have made any necessary adjustments, click the `Debug` button to start the debugger.



Selecting a Debugger. Use the drop-down menu to pick the debugger that you want to use.

You do not need to repeat the debugger selection process the next time you launch the debugger. Instead, you can select `Run` → `Debug` to start the debugger using the settings you have selected.

4.3.2. Debugging Modes for Embedded Targets

The following debugging modes are available for bare-metal targets. Except as otherwise noted, these debugging modes are available in both Sourcery G++ Debug and Sourcery G++ Debug (Attach) launch configurations.

Sourcery G++ Debug Sprite for ARM. The Sourcery G++ Debug Sprite for ARM is designed to debug ARM hardware connected to your host system using a supported debugging device, as described in Chapter 7, “Sourcery G++ Debug Sprite”. You can select the device you are connecting with, the type of target board you have, and any device-specific options using the `ARM Settings` subtab of the `Debugger` tab.

Macraigor OCDRemote. OCDRemote is a utility provided by Macraigor Systems that supports various JTAG/BDM debugging devices. The Sourcery G++ IDE launches OCDRemote automatically when you begin to debug, using the options you provide when you select this debugging mode.

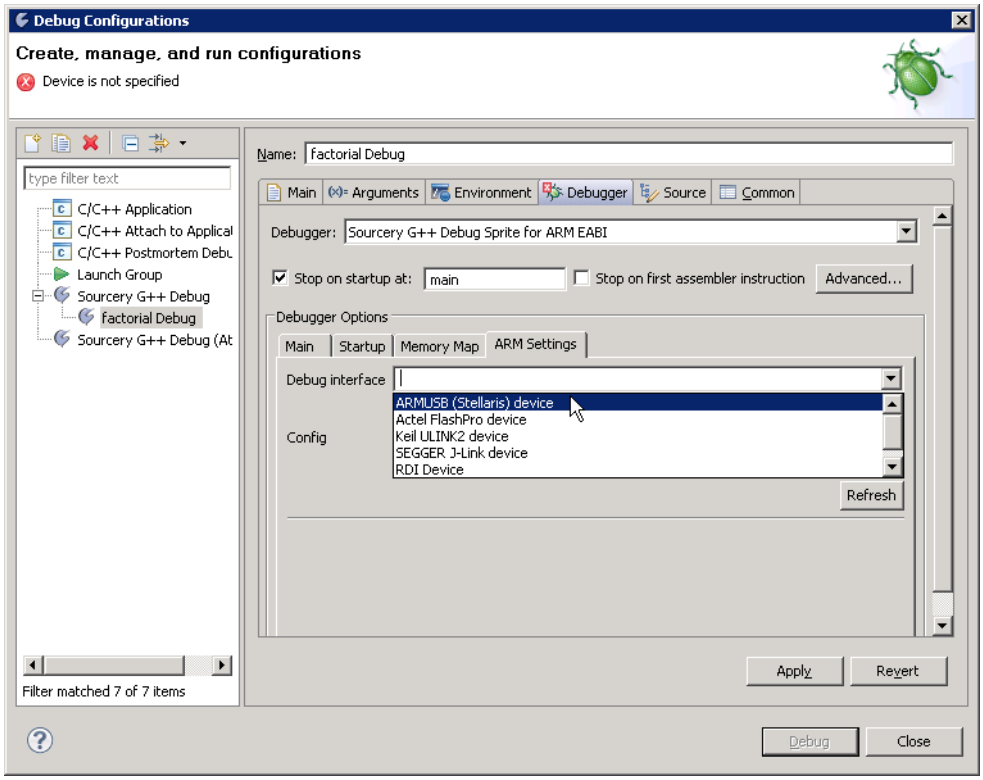
Sourcery G++ External Embedded Server. The External Embedded Server mode is designed for use with target systems that have no operating system support for debugging. In the External Embedded Server mode, Sourcery G++ connects to a “GDB stub” that controls execution on the target system. You must start the stub manually.

Sourcery G++ QEMU CPU Emulator. The Sourcery G++ QEMU CPU Emulator debugger uses the QEMU instruction-set simulator provided with Sourcery G++. You do not need target hardware in order to use it. This is the easiest way to try out Sourcery G++. You can use this mode with Sourcery G++ Debug launch configurations; it is not supported by Attach configurations.

Detailed information about using each of these debugging modes is provided below.

4.3.2.1. Sourcery G++ Debug Sprite for ARM

Settings specific to the Debug Sprite for ARM can be found on the `ARM Settings` subtab of the `Debugger` tab. On that subtab, you can specify which device to use, which device initialization file to use, and, depending on the device, additional options relevant to that device.



ARM Settings. Use the `ARM Settings` subtab to configure the Sourcery G++ Debug Sprite for ARM EABI.

The `Device` field allows you to select which debug device to use. When you select the Sprite debugger in the IDE, the Sprite probes for supported devices connected to your system. (This may take a few seconds.) You can then select one of the devices found on your system. Refer to Chapter 7, “Sourcery G++ Debug Sprite” for more information about the devices supported by this version of Sourcery G++, and for troubleshooting help if a device you have connected is not listed by the IDE.

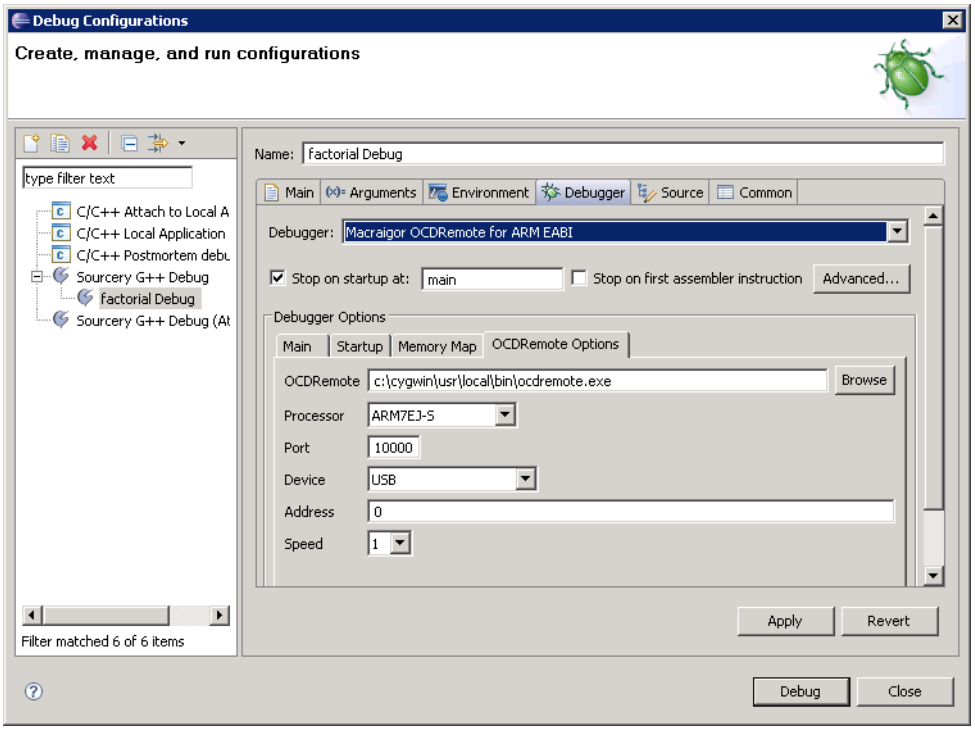
The `Config` field allows you to specify a device initialization file that is used when connecting to the device. This initialization file should always be specified. You can either select from a list of predefined initialization files, or specify a custom initialization file by clicking the `Browse` button and selecting a file. By default, the IDE uses the initialization file for the board you previously selected in the project properties.

A debug device may have additional parameters that can be specified on the `ARM Settings` subtab. The parameters that apply to specific debug devices are documented in Chapter 7, “Sourcery G++ Debug Sprite”. Some parameters are always necessary and others are optional. To set the value for an optional parameter, first enable the parameter by selecting the corresponding checkbox, and then set its value.

Refer to Section 4.3.3, “Tuning Debugger Behavior” for additional options you can set for this debugging mode.

4.3.2.2. Macraigor OCDRemote

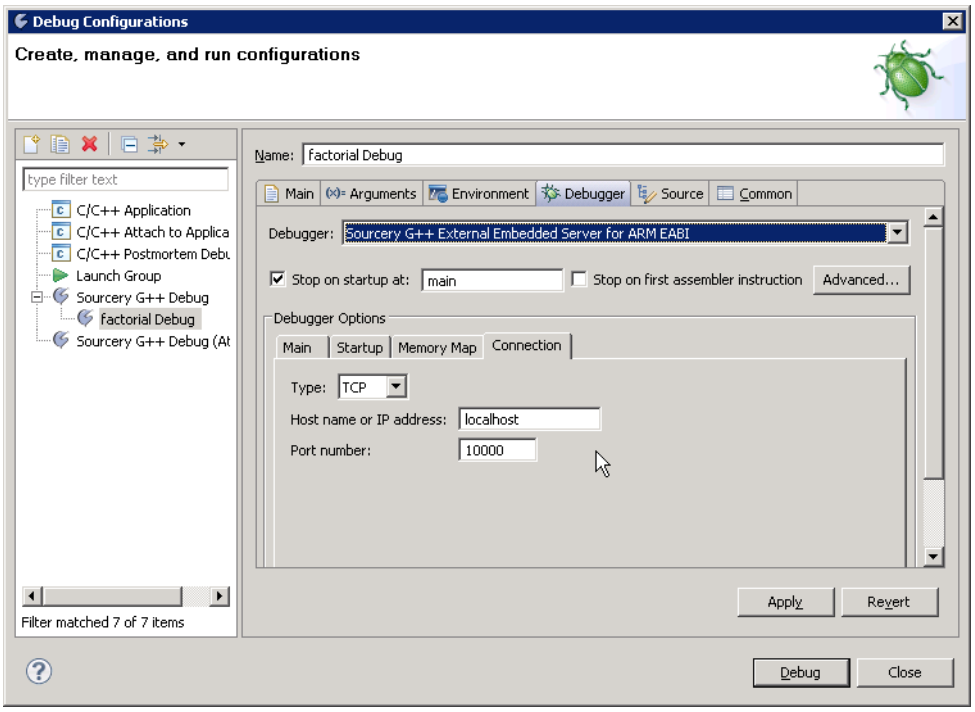
When configuring the OCDRemote debugger option, you can click on the `OCDRemote Options` subtab to specify options that are passed to Macraigor OCDRemote. Refer to Macraigor's documentation for more information on the OCDRemote configuration parameters.



OCDRemote Options. Use the `OCDRemote Options` subtab to configure OCDRemote.

4.3.2.3. Sourcery G++ External Embedded Server

When using the External Embedded Server mode, you specify how the debugger connects to the target on the `Connection` subtab of the `Debugger` tab. The default connection is set to TCP connection to localhost, port 10000. You can adjust the host and port number, or you can select a serial line connection.



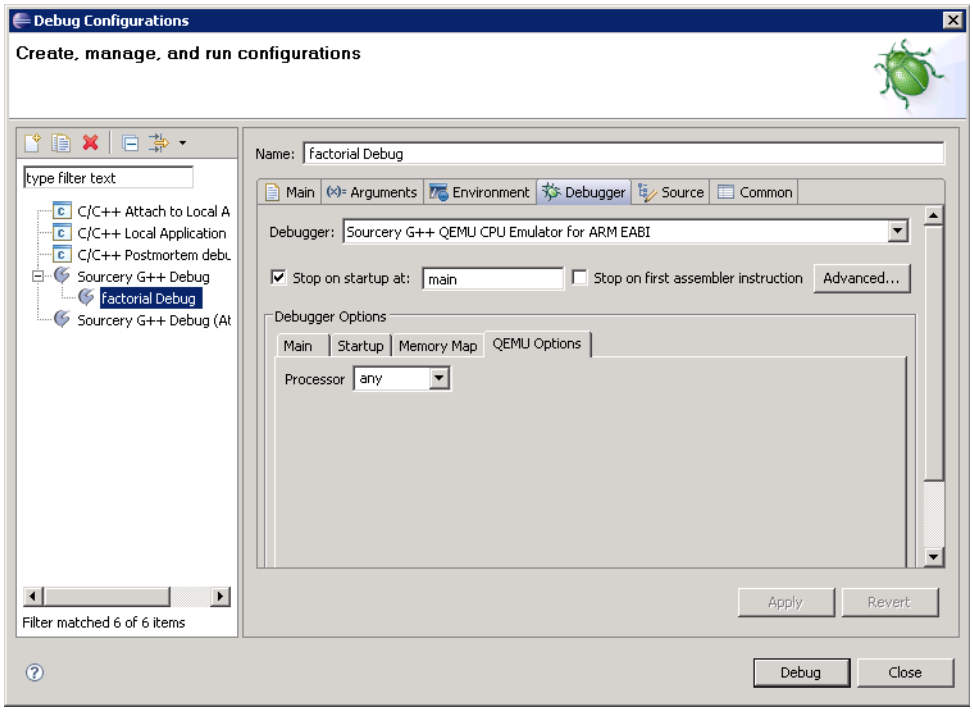
Embedded Server Connection. Use the `Connection` subtab to configure the External Embedded Server debugger.

Refer to Section 4.3.3, “Tuning Debugger Behavior” for additional options you can set for this debugging mode.

4.3.2.4. Sourcery G++ QEMU CPU Emulator

In order to use QEMU as a debugging target, you must select QEMU as your target board in your project properties. You must also select a processor and other options for your project C/C++ build settings that are compatible with the emulations supported by QEMU. See Section 4.2.1, “Setting Up an Example Project” for information about setting your project properties.

When configuring the QEMU debugger option, you can click on the `QEMU Options` subtab to select a processor for QEMU to emulate. The default value, `any`, allows QEMU to execute code compiled for any ARM processor.

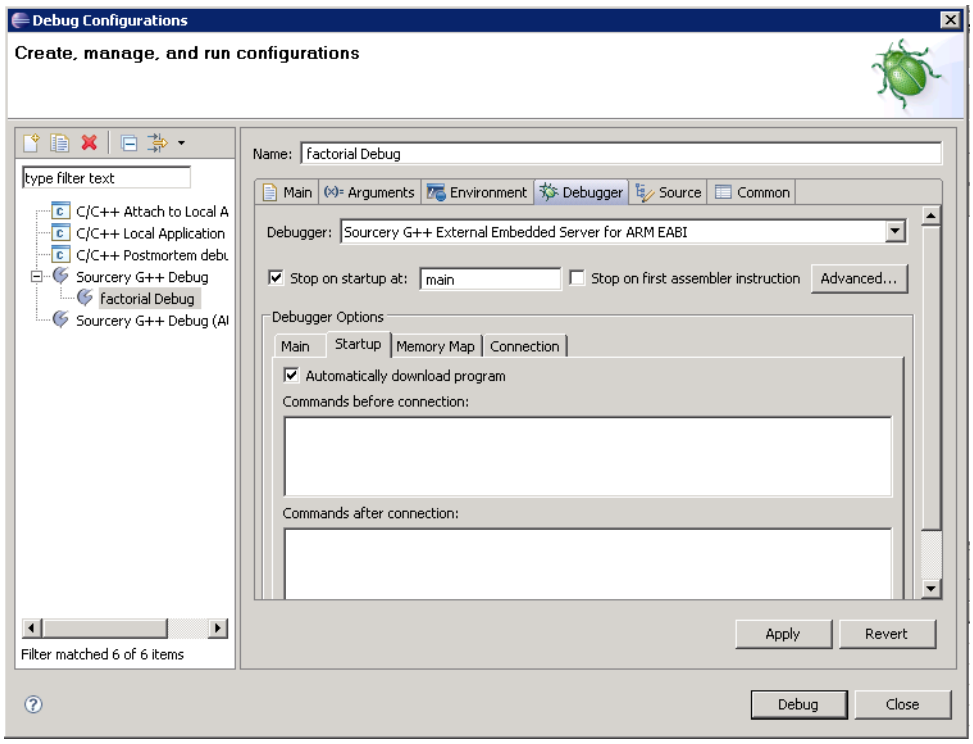


QEMU Options Tab. Use the `QEMU Options` subtab to configure the QEMU CPU emulator.

4.3.3. Tuning Debugger Behavior

4.3.3.1. Debugger Startup

Debugger startup consists of initialization or connection to a target, optional loading of the application (for non-native targets) and running the program. You can customize the startup process on the `Startup` subtab of the debugger dialog.



Customizing Debugger Startup. Use the `Startup` subtab to customize debugger startup actions.

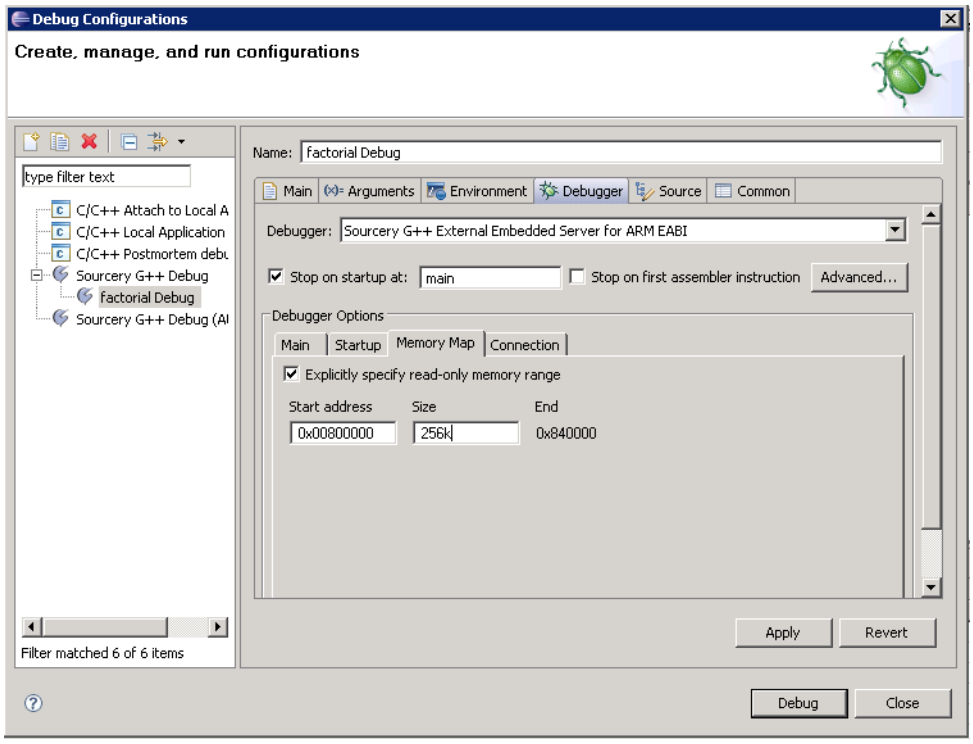
The `Automatically download program` checkbox controls whether the IDE loads the program to the target during startup. For bare-metal targets such as ARM EABI, loading automatically is the default and recommended setting, as it ensures that the binary being debugged on the target is always up-to-date. However, you can disable this behavior if you want to debug a program that is already loaded onto the target. For example, you may need to do this if the Sourcery G++ Debug Sprite does not support the flash memory on your target board or you are not using the Sprite, and instead you use a third-party tool to load your program into flash.

The `Commands before connection` and `Commands after connection` text boxes allow you to specify debugger commands to be sent before and after connection. Each line in these text boxes is interpreted as a single GDB command.

4.3.3.2. Configuring the Memory Map

If your target has a read-only memory region, such as flash memory, you should supply a memory map. GDB uses this information to determine where it must use hardware breakpoints. GDB also flags writes to memory in the read-only region as errors. Refer to Section 3.8, “Using Flash Memory” for more details about debugging programs that reside in read-only memory.

When debugging with the Sourcery G++ Debug Sprite, the memory map for the target is typically provided in the board configuration file passed to the Sprite, rather than in the IDE. If you choose to set a read-only memory region in the IDE, this overrides any memory map set in the board configuration file.



Setting a Read-Only Memory Region. Use the `Memory Map` subtab to set a read-only memory region.

To specify a read-only memory range from the Sourcery G++ IDE, you can use the `Memory Map` subtab in the `Debugger` dialog. The `Explicitly specify read-only memory range` checkbox enables this option. The memory range is specified using the `Start address` and `Size` input fields. Both fields accept either decimal or hexadecimal (with the `0x` prefix) values. When entering decimal values, you can use the `K` and `M` suffixes to specify values in kilobytes (1024 bytes) and megabytes (1024 kilobytes). It is not possible to specify more than one memory range, or designate a memory range as a specific flash chip.

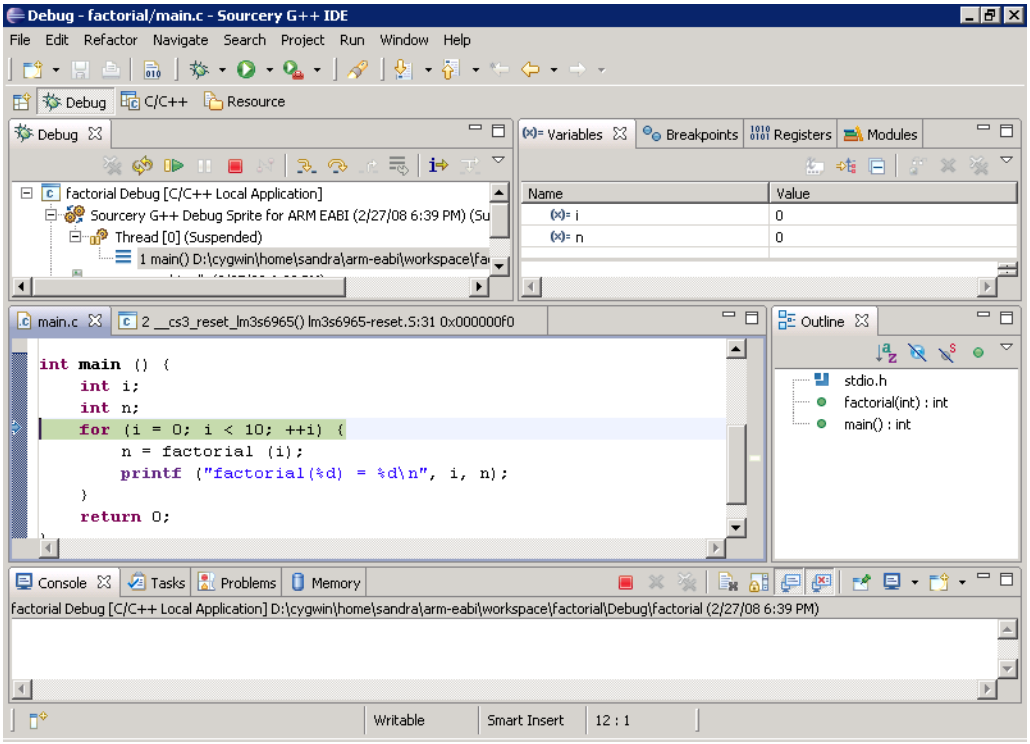
Once you have identified a read-only memory region, the debugger uses hardware breakpoints in this region automatically, and you can debug your program just as you would if it were in RAM.

4.3.3.3. Troubleshooting

When your application is large, or the debugging device is relatively slow, you may encounter timeout errors when starting debugging. In that case, you should increase the timeout settings. Select the `Preferences` item in the `Window` menu, and in the dialog that appears select `C/C++`, `Debug`, `GDB MI`. Increase the values in the `Debugger Timeout` and the `Launch Timeout` fields until your application starts without errors.

4.3.4. Controlling Execution

When you start the debugger, if you are asked whether to switch from the C/C++ perspective to the debug perspective, click the `Yes` button. Instead of showing panes that help you to develop your application, the IDE now shows panes that help you to debug your application.

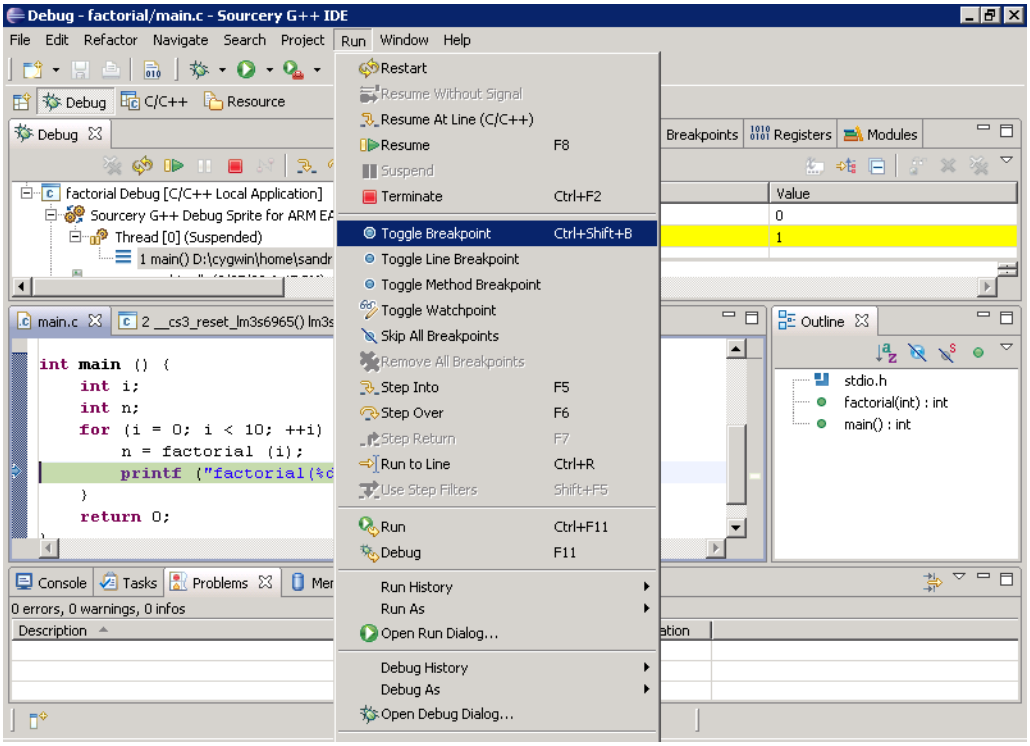


Debug Perspective. The debug perspective displays the stack, local variables, and the current location.

The debugger automatically stops on the first line of `main`. The currently active source line is highlighted. The pane at the upper left shows the application threads and the stack associated with each thread. The pane at the upper right shows the values of local variables. (At this point, `i` and `n` have not yet been initialized, so their values are indeterminate.)

Use `Run` → `Step Over` (**F6**) to advance by a single line. Because the program has changed the value of `i`, the IDE highlights the value in the variable pane.

By looking at the code, you can see that the program calls `factorial` and then calls `printf` to print out the resulting value. You can set a breakpoint right before the call to `printf` by clicking anywhere on that line, and then using `Run` → `Toggle Breakpoint` (**Ctrl**+**Shift**+**B**).



Setting a Breakpoint. Set a breakpoint by highlighting the line where you want to stop and then using the `Run` menu.

After setting the breakpoint, use `Run` → `Step Into` (**F5**) to step into the body of `factorial`.

The IDE no longer displays the value of `i` because there is no local variable `i` within `factorial`. If you wish to see the value of `i` (from `main`), select the stack frame for `main` in the pane at the upper left. The IDE displays the variables for whichever frame is presently selected.

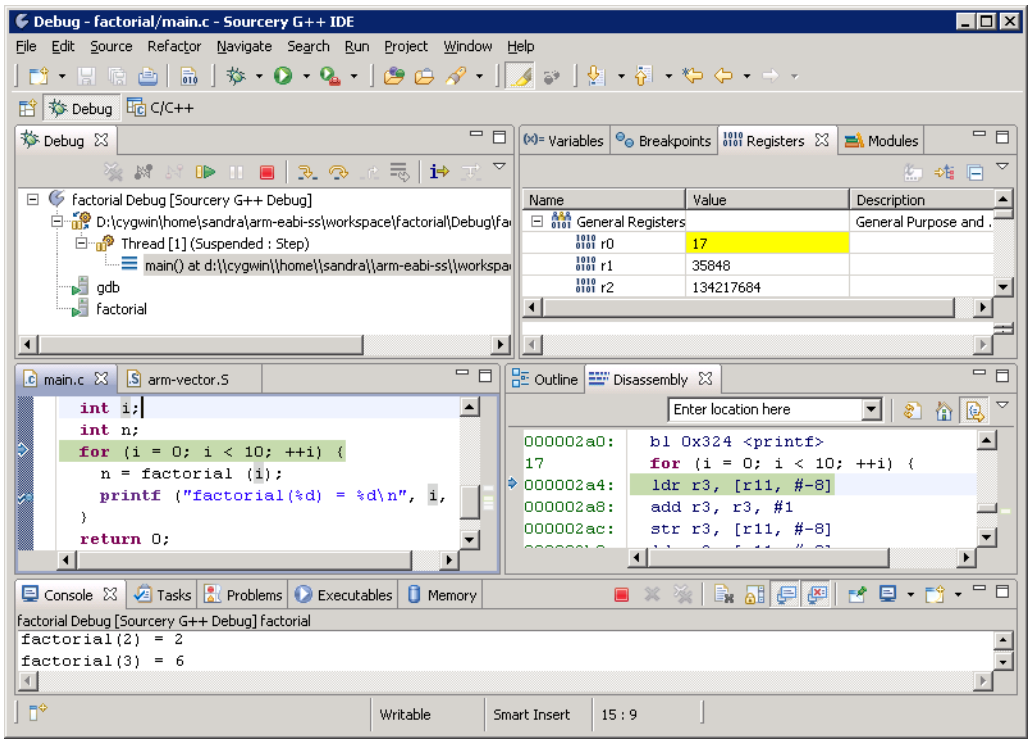
Now, proceed to the breakpoint by using `Run` → `Resume` (**F8**). The variable `n` now has the value `1` because the factorial of zero is one. Step over the call to `printf` to print the value in the console.

4.3.5. Low-Level Debugging

You may sometimes need to debug at the machine level, rather than at the source code level. For example, if you are working with an assembly code device driver, you may wish to see the values stored in machine registers and step through the code instruction by instruction.

To view machine registers, click on the `Registers` tab, and expand the `General Registers` register group. When the values of registers change, the registers are highlighted in the IDE.

To see the instructions being executed, use `Window` → `Show View` → `Disassembly`. You can set breakpoints on particular machine instructions in disassembly view in the same way that you can set breakpoints on source lines in the source code editor views in the debugger.



Low-Level Debugging. The Sourcery G++ IDE can display machine registers and assembly code.

The `Step Over` and `Step Into` commands normally operate at the source level, advancing program execution to the next C or C++ statement. To step by machine instructions instead, click on the `Instruction Stepping Mode` button on the `Debug` toolbar. This is the button whose icon has the letter "i". Click this button again to return to stepping by source lines.

4.4. Advanced IDE Features

This section covers several advanced features of the Sourcery G++ IDE, including using the Sourcery G++ Board Builder to extend CS3's board support with custom board definitions. It also covers creating Makefile projects, building managed build projects from the command line, importing source code and already-compiled programs into the IDE, and using other programming tools provided by the IDE.

4.4.1. Using the Sourcery G++ Board Builder

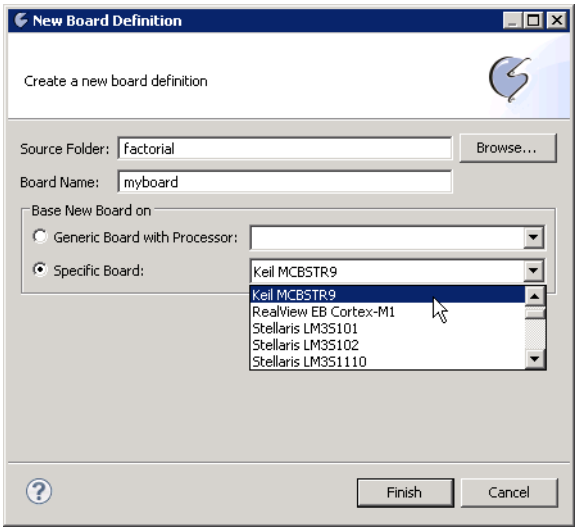
The Sourcery G++ Board Builder is an advanced IDE feature that allows you to extend the CS3 board support library with additional definitions for custom boards. This section assumes you have some familiarity with CS3 concepts and terminology; refer to Chapter 6, “CS3™: The CodeSourcery Common Startup Code Sequence”.

The Board Builder creates *board definition* files, which contain the memory map and initialization sequence for each board. Board definition files have a `.cs3` extension. Board definitions are contained in projects, but are not used to build the project until you explicitly select the board as the target in the project properties. For example, this allows you to set up a project as a library containing multiple board definitions that you can import as needed into other application projects.

To simplify data entry and configuration for a new board, the Board Builder allows you to specify a similar *base board* from those already provided by CS3 for ARM EABI targets. The processor, interrupt scheme, memory map, reset sequence, and other properties of the base board are copied into your new board definition, so that you need only change the properties that are different for your new custom board. If there is no appropriate base board available, you can also create a new board definition based on a specified processor.

To begin using the Board Builder, first create a new C or C++ project as described in Section 4.2.1, "Setting Up an Example Project". On the target properties page of the C/C++ Project wizard, you can select the base board on the `Board` option and configure other options such as the memory location and hosting now; those selections will remain in place after your new custom board is added to the project. If there is no appropriate base board listed, you can also select `Other` on the `Board` option to indicate that you will fill in your own target board and other properties later.

Right-click on your project in the `Project Explorer` tab, then choose `New` → `Board Definition` from the menu. This opens the New Board Definition Wizard.



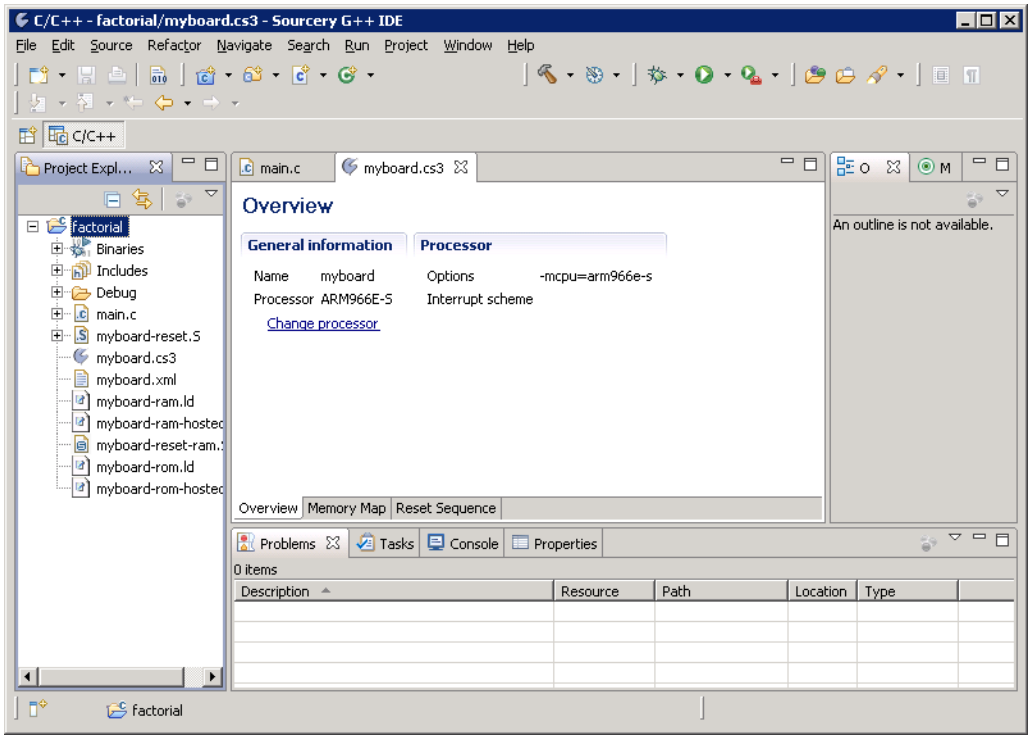
New Board Definition Wizard. You can create a board definition by copying from a base board, or by creating a board based on a particular processor.

By default, board definitions are created in the usual source folder for the selected project, but you can choose another source folder if you wish. Fill in the board name; this must be a valid C identifier and so not contain spaces or other special characters. Finally, choose the base board for your new board definition, or specify that you want to create a new board based on a particular processor instead.

Clicking `Finish` pops up a dialog asking whether you want to switch your project to use the new board. Note that even if you answer `Yes`, you may still need to adjust your project build properties manually to use the correct hosting and memory regions for your new board. You can do this from the project properties dialog, as described in Section 4.2.5, "Customizing Build Actions". Note that your newly-created board definition is now available in the choices for the `Board` option in this dialog, and you may select it as you would any other listed board.

When you click `Finish` in the New Board Wizard, this also opens the new board definition in the board editor. The board editor uses tabs to group the various properties of your board definition.

The `Overview` tab displays general information about the board, including its processor and interrupt handling scheme. You can change the processor for a previously-created board definition by clicking the link on this page. The other information displayed is not editable.



Board Definition Overview. The `Overview` tab shows properties of the board.

The `Memory Map` tab displays a table of the memory regions on the board. You may edit any of the fields in the table by clicking on them. Use the buttons on the right side of the pane to add or delete memory regions from the table.

- The `Name` of a memory region can be an arbitrary identifier, but some names, such as `ram` and `rom`, have special meanings to CS3, depending on the supported profiles. Refer to Section 6.3.1, “Memory Regions and Program Sections” for details.

- The `Start` and `Size` values can be provided in decimal, hex (with a `0x` prefix) or octal (with a `0` prefix). You can use a `K`, `KB`, `M`, `MB`, `G` or `GB` suffix to denote a unit of memory.

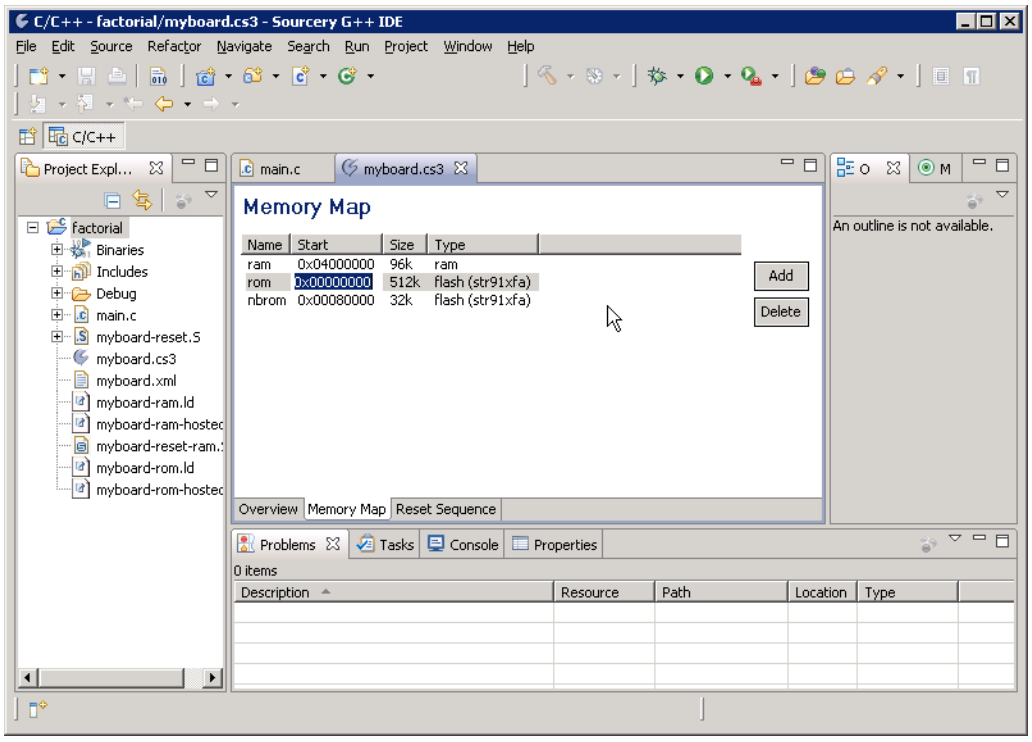
- The `Type` field identifies the properties of the memory region. It can have one of the following values:

  `ram` Memory that is both readable and writable.

  `rom` Read-only memory with contents fixed at manufacture or writable only by external tools.

  `flash` Read-only memory that can be flashed. There may be multiple `flash` types listed. The Sourcery G++ Debug Sprite can program such regions automatically. The Sprite uses the flash type to select the appropriate flash programming method. Refer to Section 7.14, “Board File Syntax” for more information about flash support in the Sourcery G++ Debug Sprite.

`io` Device memory which cannot be used for program storage.



Board Definition Memory Map. The `Memory Map` tab allows you to edit the location, size, and type of the memory regions on your custom board.

The `Reset Sequence` tab allows you to define the sequence of control register and memory writes necessary to perform the hard reset phase of CS3 initialization, as described in Section 6.2, “Program Startup and Termination”. Consult the documentation for your board for the exact sequence of actions that is required.

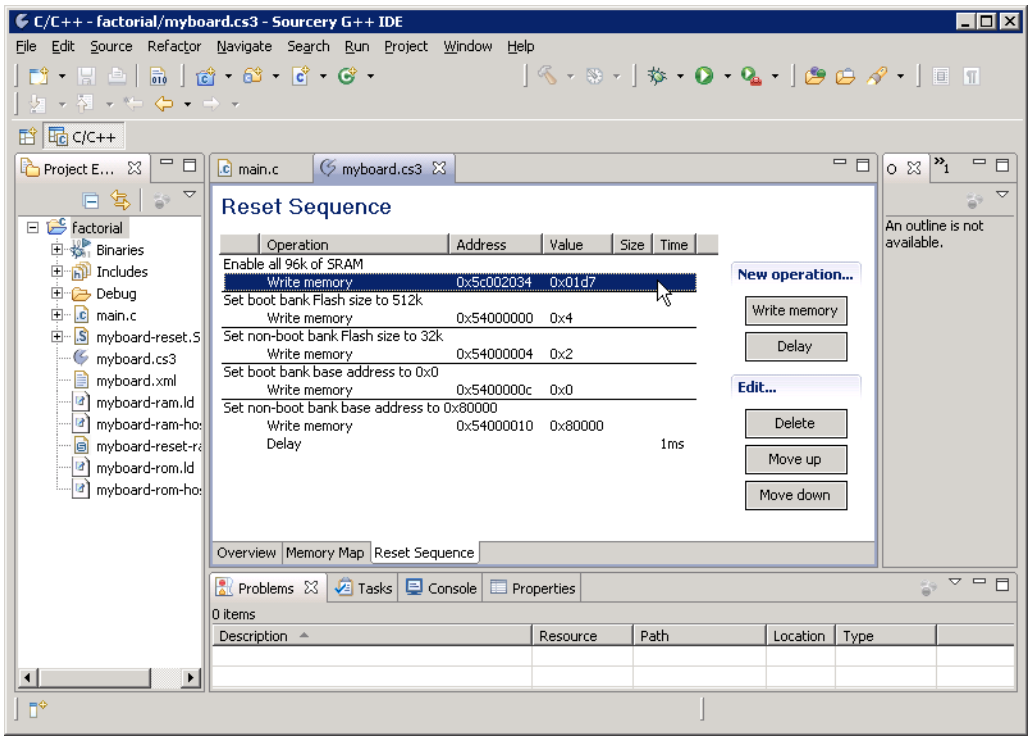
Buttons for adding, reordering, or deleting operations are listed on the right-hand side of this tab. To add an operation to an existing reset sequence, you must first select an element of that sequence; the new operation is added after the one selected.

The supported reset operations are:

- `Write memory` writes a value to memory at the given address. The `Size` field can be used to specify the bit width of the operation; it defaults to 32 if omitted. The `Time` field is not used for this action.
- The `Delay` element introduces a delay to allow a previous initialization to complete. This element uses only the `Time` field.

The conventions for the numeric fields in the table are the same as those given above for the `Memory Map` tab. There are also these additional rules for syntax on specific value types:

- The name of a memory region may be used as an address value, and you can also use address expressions of the form *`name+offset`*, such as `mbo+0xa0680`.
- Time values must include a `s`, `ms`, or `us` suffix.



Board Definition Reset Sequence. The `Reset Sequence` tab allows you to specify the series of operations, such as memory and control register writes, to initialize your board.

When you have finished editing your board definition, select `File` → `Save` (**Ctrl+S**). Saving the board definition causes the Board Builder to regenerate linker scripts and startup code files which can be used when building your project. A board file for use with the Sourcery G++ Debug Sprite is also generated. These files are installed in the same source folder in your project where the `.cs3` file was created.

When you save a board definition, the Sourcery G++ Board Builder detects if you have made manual modifications to previous versions of its generated files rather than simply overwriting them. You can choose either to overwrite the modified files or disable automatic generation for that board.

If you copy a `.cs3` file directly into your project, note that the generated files required to use the board in your project are not created until the next time you open the project in the IDE. Alternatively, if the project is already open, you can right-click on the source directory in the `Project Explorer` pane and select `Refresh`. Building the project does not trigger regeneration of the files from the board definition.

Any errors that occur when building the linker scripts and other generated files from the board definition are noted in the `Problems` tab in the lower pane of the IDE. For example, you may get errors if the memory map in your board definition does not include all the expected regions for the memory profiles supported by CS3 on ARM EABI targets.

4.4.2. Makefile Projects

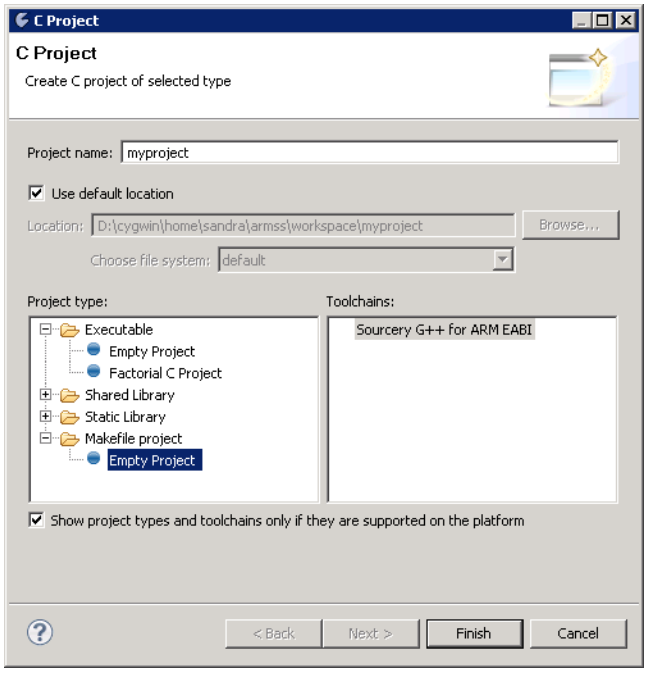
This section explains how to use the advanced Makefile project mode, instead of the simpler managed build mode described in Section 4.2, “Building Applications”.

Caution

Building a Makefile project requires that you manually maintain information about how your program is built. If you use this mode, you need to be familiar with the `make` utility.

If you want to import an existing project for use with the Sourcery G++ IDE, and that project uses `make`, or some similar command-line tool to manage the build process, you should use a Makefile project, instead of the IDE's built-in project types. In Makefile project mode, the IDE invokes `make` (or an alternative program that you specify) to build your program. If you add new files to your project, you have to manually update the `Makefile` for your project.

To create a new Makefile project in the Sourcery G++ IDE, open the C or C++ Project wizard as described in Section 4.2.1, “Setting Up an Example Project”. This time, however, expand `Makefile project` under `Project type:`, and choose `Empty Project`. Under `Toolchain`, select `Sourcery G++ for ARM EABI`. Then click `Finish`.



Creating a Makefile Project. Use the C or C++ Project wizard to create a new empty Makefile project.

Before you can build your project, you must add a `Makefile` to it as well as your C or C++ source files. You can create new files by right-clicking on the project name in the left-hand pane and selecting `New`. To import existing files into your project, right-click on the project name and select `Import...`. In the Import Wizard, expand `General` and select `File System`. Then continue to the next page to select the files to import.

Alternatively, you can import the entire set of sources for a Makefile-based software package into the Sourcery G++ IDE from an existing directory in one step when you create the project. To do this, uncheck the `Use default location` box in the new project wizard and choose the appropriate `Location` to your files.

You may also have to adjust your `Makefile` to use Sourcery G++. For example, you might need to set the `CC` variable in your `Makefile` to `arm-none-eabi-gcc`.

4.4.3. Building IDE Projects from the Command Line

If you need to be able to do batch-mode builds of your project — for example, for nightly builds or automated product packaging scripts — you can invoke the Sourcery G++ IDE builder from the command line. This is an alternative to converting your managed build project to a Makefile project so that it can be built outside the IDE.

The basic recipe is

```
> sourcerygxx-ide -data workspace -cleanBuild project
```

where *`workspace`* is the pathname of your workspace and *`project`* is the name of the project to build. You can also specify the literal value `all` to build all the projects in the workspace.

The above command does a full rebuild of the project, the equivalent of cleaning the project and then building it. To do an incremental build without cleaning, use `-build` rather than `-cleanBuild`:

```
> sourcerygxx-ide -data workspace -build project
```

You can also import projects into your workspace using the `-import` option:

```
> sourcerygxx-ide -data workspace -import project-uri
```

Here *`project-uri`* is the pathname or URI of the project to import. The effect of this option is similar to using the Import wizard's `Existing Projects into Workspace` feature.

The `-importAll` option is similar to `-import`, but imports all projects from a workspace or other directory tree into your active workspace. This option also takes a pathname or URI argument to specify the root of the directory tree.

The `-import` and `-importAll` options can be repeated and used in combination with `-build` or `-cleanBuild`. For example, you can import multiple projects into a new temporary workspace and use `-cleanBuild all` to build all of them with a single command.

Use the `-refresh` option if you have made changes to your project files outside the IDE, such as adding or deleting files. This forces the IDE to refresh its internal workspace state before starting to build.

Output from the command-line builder, which would be directed to a console window when building in the interactive IDE, is sent to standard output. If there are errors in the build command itself (such as specifying a project that doesn't exist), they are logged to standard error. The `sourcerygxx-ide` command returns zero on success and non-zero if the build fails.

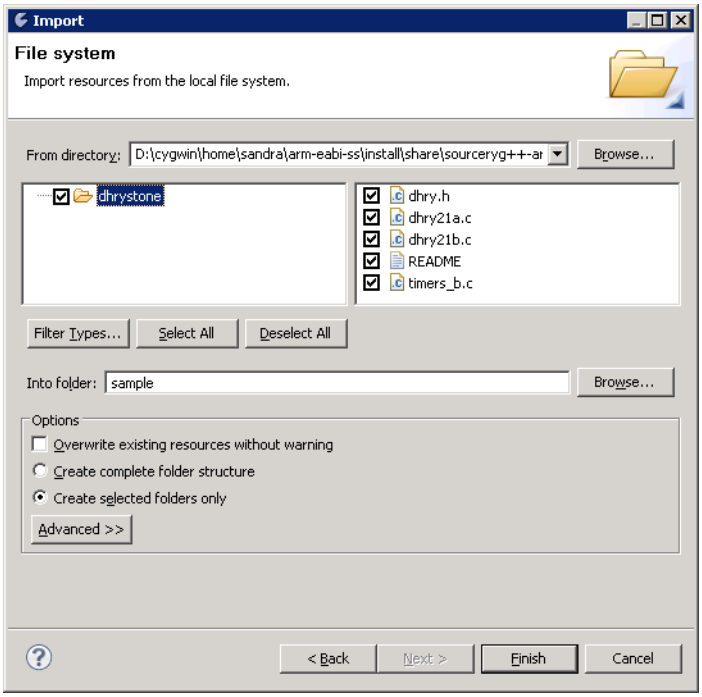
Note that only the active configuration of the selected projects (e.g., either the Debug or Release configuration) is built. It is not possible to select the configuration to build on the command line.

4.4.4. Importing Code into the IDE

If you have an already-written program, you can import it into the IDE as a new managed build project using your existing source files. This section shows you how to do this, using the Dhrystone application program bundled with Sourcery G++ as an example.

Start by following the steps described in Section 4.2, “Building Applications” to create a new Executable project. Call the new project `sample`.

The next step is to import the source code. Right-click on the `sample` project in the `Project Explorer` tab, and select `Import...`. This opens the import wizard. Select `General` → `File System`, and click `Next`. Click on `Browse...` beside the `From directory:` edit box at the top of the page. Navigate to the Sourcery G++ install directory and then to `share/sourceryg++-arm-none-eabi-examples/dhrystone`, and click `Ok`. Click the checkboxes to select all the files in this directory, then click `Finish`.



Importing Source Files. First choose the source directory, then select the files to import.

Now follow the instructions in Section 4.2, “Building Applications” to build the project. Your sample program is now ready to execute or debug. Please refer to Section 4.3, “Debugging Applications” for instructions on how to debug the target application.

Note that if you wish to run the Dhrystone program for benchmarking purposes, you should build a Release configuration rather than the default Debug configuration, and adjust the optimization options in the `Build Settings` section of the project properties dialog. Refer to the `README` file included with this example for the correct build options for benchmarking.

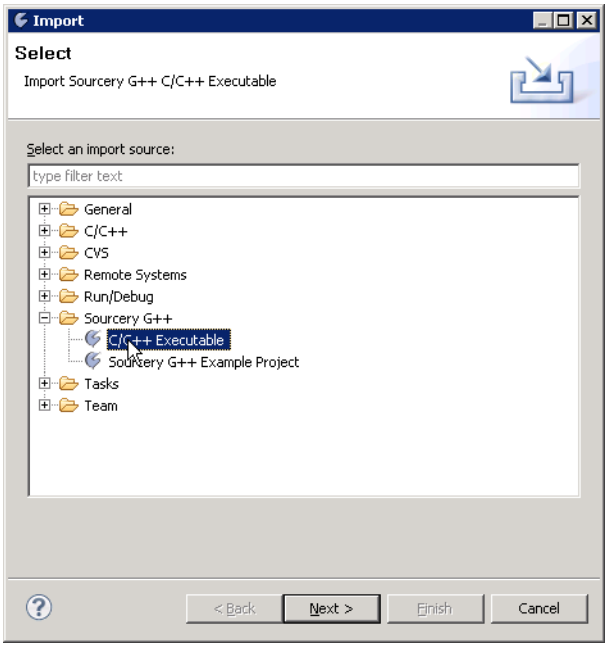
While this section has shown how to import an entire program into a new project, you can also use the import wizard to import individual files rather than a complete program, or to add files to an existing project. The import wizard copies the files into the project from the file system rather than linking to their original locations.

4.4.5. Importing an Executable into the Sourcery G++ IDE

You can use the Sourcery G++ IDE to debug an executable program you have built outside the IDE. You may need to do this to debug a program that requires a build procedure that cannot be simply expressed in a `Makefile`, a program built with a non-Sourcery G++ compiler, or a binary provided by a third party for which you do not have complete source code.

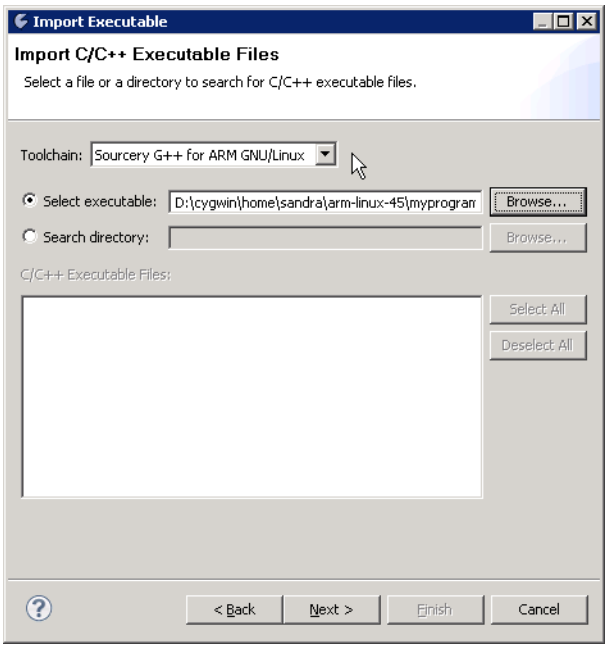
For best results in the debugger, you should build your application with debugging enabled (via the `-g` option) if possible prior to importing it. The IDE can use the debugging information in the imported executable to find and display the source code for the program as you debug it, even though the source files are not part of the project.

To import a program, select `File` → `Import...`. This opens the Import Wizard. Expand `Sourcery G++` and select `C/C++ Executable`. Then click `Next`.



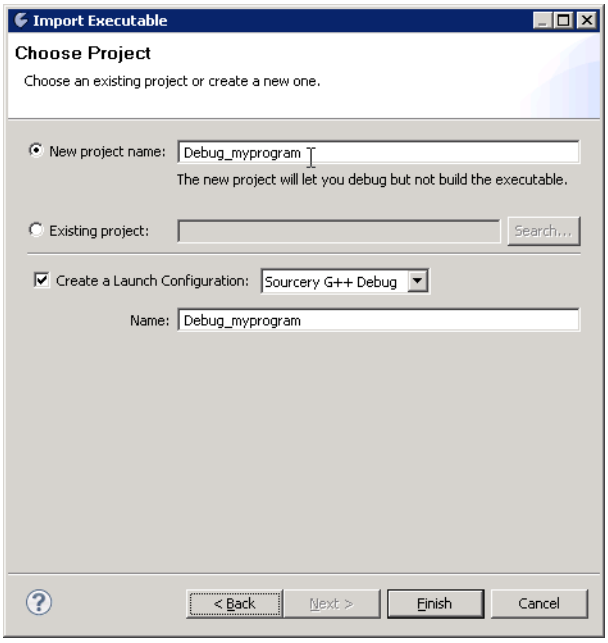
Import Wizard. Select `C/C++ Executable`.

On the next screen, use the `Browse` button to select the program you wish to debug. If you have more than one Sourcery G++ toolchain installed, use the `Toolchain` drop-down to select the target for your executable. Then click `Next`.



Choose Executable to Import. Import the executable file you wish to debug.

Finally, choose a name for the project. If you leave the `Create a Launch Configuration` box checked, the launch configuration dialog automatically pops up when you click `Finish`. At this point, you can continue to set up the debugger options and launch the program as described in Section 4.3, “Debugging Applications”.

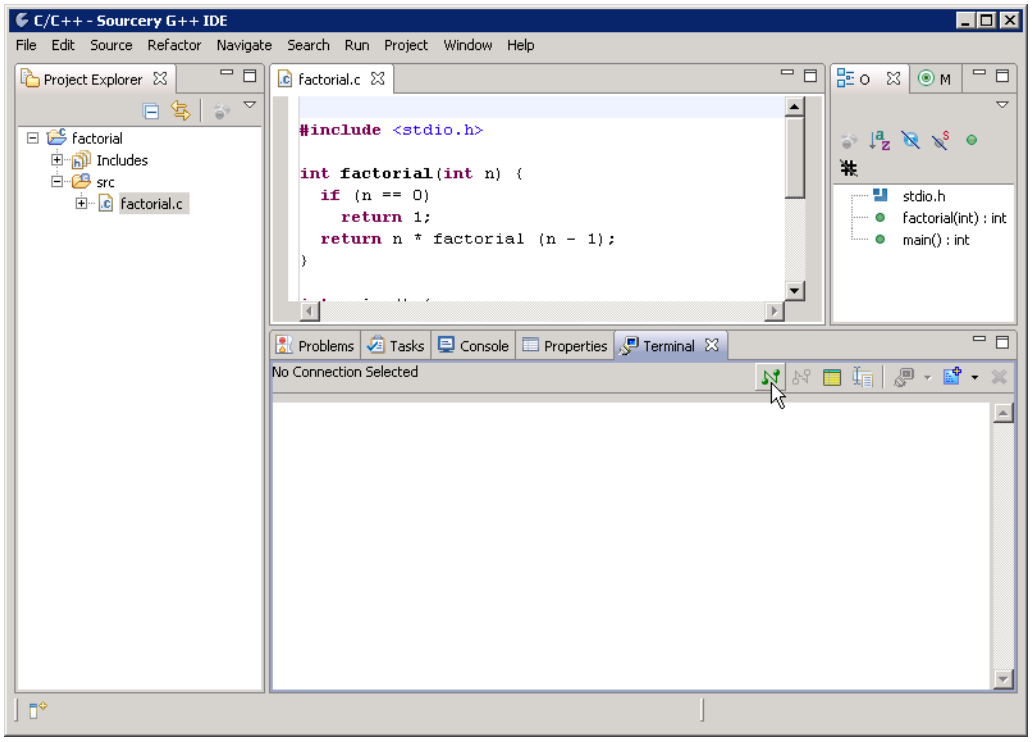


Choose Project for Import. Give the new project a name.

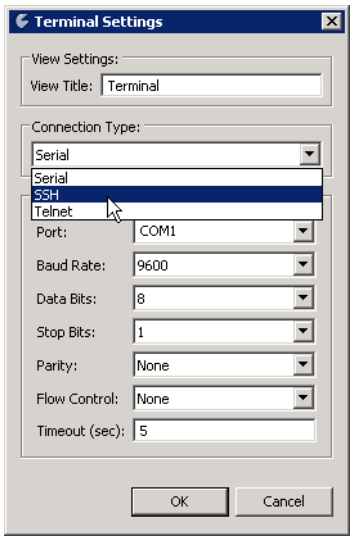
4.4.6. Using the Terminal Emulator

The Sourcery G++ IDE includes a built-in terminal emulator which supports serial, Telnet, and SSH connections. For example, you can use the terminal to connect to a serial console or UART driver on your target board.

To open a terminal connection, first select `Window` → `Show View` → `Other...` from the top menu. Then choose `Terminal` → `Terminal` in the dialog box. This opens a new tab for the `Terminal` view. Click on the new connection icon in the toolbar for this tab, and use the pop-up dialog to configure the connection.



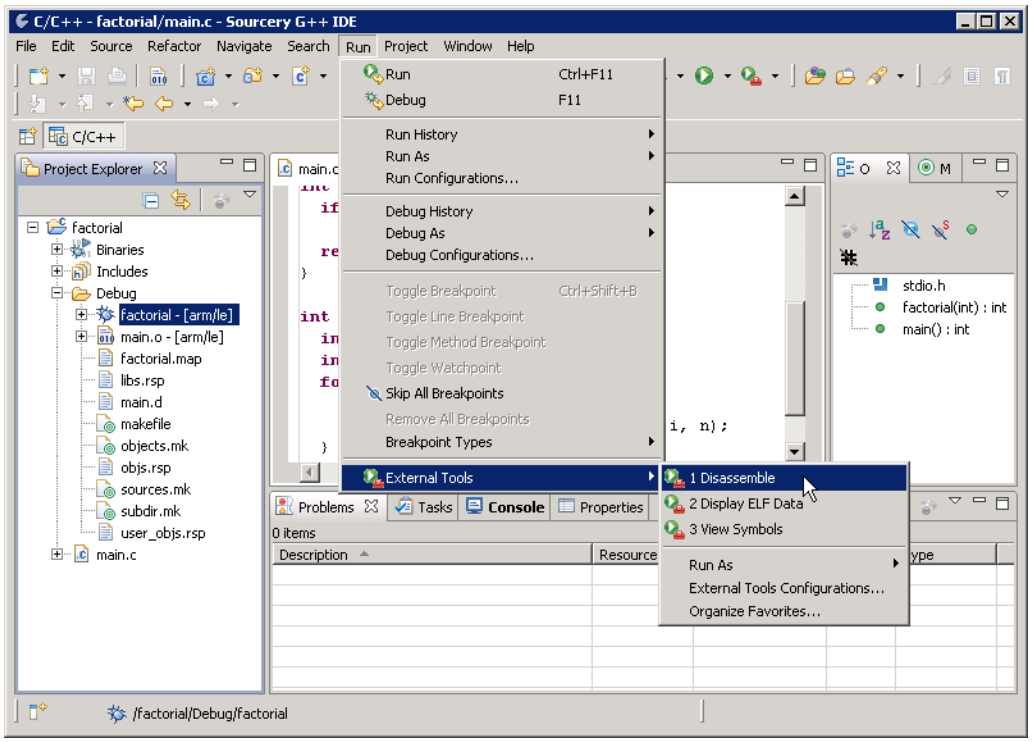
Opening a Terminal. Click on the new connection icon in the `Terminal` tab.



Terminal Settings. Configure the terminal properties in the pop-up dialog.

4.4.7. Using External Tools

The Sourcery G++ IDE includes some predefined External Tool launches to run common informational queries using the GNU Binary Utilities. To use these tools, first select the object file or executable in the `Project Explorer` tab. Then, choose `Run` → `External Tools` from the top menu. The list of available tools includes `Disassemble` (which runs the `objdump -d` command), `Display ELF Data` (`readelf`), and `View Symbols` (`nm`).



External Tools. The Sourcery G++ IDE provides built-in tools for displaying properties of compiled files.

You can modify the command-line arguments passed to these tools or add your own custom External Tool launches by selecting `External Tools Configurations...` from this submenu. To create a new launch, select `Program` in the left-hand pane of the dialog and click the `New` icon. Use the predefined External Tool launches as a guide to filling in the properties of your new launch.

4.4.8. Using Run Launches

On ARM EABI targets, the Sourcery G++ IDE supports loading and running a program on the target board without debugger control. In this mode, the IDE initializes the target and loads the program (including programming flash, if necessary) using the Sourcery G++ Debug Sprite. However, instead of passing control over the application to the interactive debugger, the Sprite detaches after starting the program from reset, leaving the target running freely.

To run programs in this way, use the `Run` → `Run Configurations...` menu item. Create a new `Sourcery G++ Debug` launch, or select one that you have previously created for debugging, as described in Section 4.3.1, "Starting the Debugger". Click the `Run` button in the launch configuration dialog to start your program. You can re-launch the same configuration again, without going through the configuration dialog, by selecting `Run` → `Run` from the top menu.

Because your application runs outside the control of the debugger when launched in this way, it cannot make use of semihosted I/O features, which depend on debugger support. You should link your application with an unhosted CS3 profile. For further discussion of CS3 semihosting, see Section 6.1.2, "Hosting and Semihosting".

4.4.9. Using Eclipse Plugins in the Sourcery G++ IDE

Eclipse plugins provide many additional tools and utilities to extend the functionality of the Sourcery G++ IDE. Plugins that are bundled with the IDE include:

CVS. This plugin provides direct integration with the CVS version control system. You can check out a project from a CVS repository into your workspace by choosing `File` → `Import...` from the top menu and then `CVS` → `Projects from CVS` in the Import Wizard. Additional CVS operations are available by right-clicking on the project in the `Project Explorer` tab and selecting the `Team` submenu. For additional documentation, visit the CVS plugin web site[2].

Mylyn. This plugin provides task management features for Eclipse, including integration with external bug trackers as well as personal to-do lists. Mylyn also provides a task-focused way to organize your workspace, so that you can easily switch an entire set of active views and resources when you switch tasks. Visit the Mylyn project web site[3] for tutorials and documentation.

In addition to these pre-installed plugins, there are many others available that you can install into the Sourcery G++ IDE yourself. You can find a directory of available plugins, sorted by category, on the Eclipse Plugin Central[4] web site.

[2] http://www.eclipse.org/eclipse/platform-cvs/
[3] http://www.eclipse.org/mylyn/
[4] http://www.eclipseplugincentral.com/

Chapter 5
Using Sourcery G++ from the Command Line

This chapter demonstrates the use of Sourcery G++ from the command line. If you prefer to use an integrated development environment to build your applications, you may refer to Chapter 4, “Using the Sourcery G++ IDE” instead.

5.1. Building an Application

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

Using an editor (such as `notepad` on Microsoft Windows or `vi` on UNIX-like systems), create a file named `main.c` containing the following simple factorial program:

```
#include <stdio.h>

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

int main () {
  int i;
  int n;
  for (i = 0; i < 10; ++i) {
    n = factorial (i);
    printf ("factorial(%d) = %d\n", i, n);
  }
  return 0;
}
```

Compile and link this program using the command:

```
> arm-none-eabi-gcc -o factorial main.c -T script
```

Sourcery G++ requires that you specify a linker script with the `-T` option to build applications for bare-board targets. Linker errors like `undefined reference to `read'` are a symptom of failing to use an appropriate linker script. Default linker scripts are provided in `arm-none-eabi/lib`. Refer to Chapter 6, “CS3™: The CodeSourcery Common Startup Code Sequence” for information about the boards and linker scripts supported by Sourcery G++. You must also add the processor options for your board, as documented in that chapter, to your compile and link command lines.

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

5.2. Running Applications on the Target System

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

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. GDB can also be used to run and debug programs with QEMU, a simulator that runs on your host system.

When starting GDB, give it the pathname to the program you want to debug as a command-line argument. For example, if you have built the factorial program as described in Section 5.1, “Building an Application”, enter:

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

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

5.3.1. Connecting to the QEMU Emulator

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

To start and connect to the emulator from within GDB, use this command:

```
(gdb) target qemu
```

This starts QEMU with the appropriate options to emulate a bare-board target and accept the connection from GDB.

You can optionally pass an argument to specify the CPU that QEMU should emulate:

```
(gdb) target qemu cpu
```

The default value, `any`, allows QEMU to execute code compiled for any ARM processor. Additional supported CPU emulations include `arm926`, `cortex-a8`, and `cortex-m3`.

In order to use QEMU as a debugging target, you must build your program with a QEMU linker script. Refer to Section 6.5, “Supported Boards for ARM EABI” for details. You must also compile your code with options that are consistent with the processor you specify when invoking QEMU.

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

5.3.2. Connecting to the Sourcery G++ Debug Sprite

The Sourcery G++ Debug Sprite is a program that runs on the host system to support hardware debugging devices. You can use the Debug Sprite to run and debug programs on a target board without an operating system, or to debug an operating system kernel. See Chapter 7, “Sourcery G++ Debug Sprite” for detailed information about the supported devices.

You can start the Sprite directly from within GDB:

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

Refer to Section 7.3, “Invoking Sourcery G++ Debug Sprite” for a full description of the Sprite arguments.

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

5.3.3. Connecting to an External GDB Server

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

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

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

5.3.4. Loading and Running Applications

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

```
(gdb) load
```

If you are using the Sourcery G++ Debug Sprite and have built your application to run from flash memory, flash programming is performed transparently by the Sprite when you issue the `load` command from GDB. 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
CS3™: The CodeSourcery Common Startup Code Sequence

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

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

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

In addition to providing linker support, CS3's functionality is fully integrated with the Sourcery G++ Debug Sprite. For each supported board, CS3 provides the board file containing the memory map and initialization sequence required for debugging applications on the board via the Sprite, as documented in Section 7.13, “Supported Board Files”.

CS3 is also the foundation of the Sourcery G++ Board Builder. This feature of the Sourcery G++ IDE allows you to use the power of CS3 to extend the board library included with Sourcery G++ to include your own custom board definitions. You can find more information about using the Board Builder in Section 4.4.1, “Using the Sourcery G++ Board Builder”.

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

- Section 6.1, “Linker Scripts” provides basic information you need to know in order to select an appropriate CS3-provided linker script for your ARM EABI board.
- CS3's program startup and termination model is discussed in Section 6.2, “Program Startup and Termination”.
- Section 6.3, “Memory Layout” discusses the mapping from program sections to memory regions. It also explains how you can refer to memory regions using CS3-provided symbolic names from C, assembly language, or the linker script, and customize placement of code or data in your program.
- Section 6.4, “Interrupt Vectors and Handlers” covers CS3's interrupt handling model, and discusses how you can customize the CS3-provided interrupt vector tables.

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

- Section 6.5, “Supported Boards for ARM EABI” lists the boards supported by CS3 for ARM EABI, and the available linker scripts for them.
- Section 6.6, “Interrupt Vector Tables” documents the details of the provided interrupt vectors for CS3-supported devices.

6.1. Linker Scripts

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

- It determines the memory addresses for placement of code and data sections.
- It defines symbolic names for memory regions present on the board, which you can use programmatically within your code.
- It provides appropriate program startup and termination code, and causes the linker to pull in any low-level board support libraries that are required to run code on the target.

- It optionally provides a *hosting* library for basic I/O functionality.
- It provides a default interrupt vector appropriate for the target processor.

When you use the Sourcery G++ IDE to build your program, the appropriate linker script is used automatically based on your settings for the board and other attributes on the `Project` → `Properties` dialog. When invoking the Sourcery G++ linker from the command line, you must explicitly supply a linker script using the `-T` option; otherwise a link error results.

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

The remainder of this section discusses profile and hosting selection considerations in more detail. You can find the full list of supported boards and linker scripts included in this release of Sourcery G++ in Section 6.5, “Supported Boards for ARM EABI”.

6.1.1. Program and Data Placement

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

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

Flash programming for ROM-based profiles is integrated with the Sourcery G++ Debug Sprite. When you debug a program on a supported board using the Sprite, flash programming is performed automatically as part of loading your program onto the target.

6.1.2. Hosting and Semihosting

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

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

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

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

```
#include <unistd.h>

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

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

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

6.1.3. Specifying a Linker Script

The Sourcery G++ IDE chooses an appropriate linker script for managed build projects based on the board and other settings you have selected for your project properties. These are set in the C/C++ Project wizard when you create your project, and can also be adjusted from the project properties dialog. For instructions, refer to Section 4.2.5, “Customizing Build Actions”.

When using Sourcery G++ from the command line or from a `Makefile`, you must add `-T` *`script`* to your linking command, where *`script`* is the appropriate linker script. For example, to target Actel CoreMP7 Cortex-M1 boards, you could link with `-T coremp7-cm1-ram-hosted.ld`.

6.2. Program Startup and Termination

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

CS3 divides the startup sequence into three phases:

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

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

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

6.2.1. The Hard Reset Phase

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

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

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

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

6.2.2. The Assembly Initialization Phase

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

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

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

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

The value of the symbol `__cs3_stack` provides the initial value of the stack pointer for profiles that must set it explicitly. The CodeSourcery linker scripts provide a default value for this symbol, which you may override by defining `__cs3_stack` yourself.

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

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

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

6.2.3. The C Initialization Phase

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

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

This function performs the following steps:

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

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

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

6.2.4. Arguments to `main`

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

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

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

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

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

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

6.2.5. Program Termination

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

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

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

6.3. Memory Layout

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

6.3.1. Memory Regions and Program Sections

The regions that are available on a particular board are listed in the table for that board in Section 6.5, “Supported Boards for ARM EABI”, below. There are two kinds of regions: those documented as "Memory regions", which are general-purpose memory banks that can be used for program or data storage; and those documented as "Other regions", which typically correspond to memory-mapped control registers or other special-purpose storage.

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

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

More generally, for a memory region named *R*, CS3 linker scripts define a section named `.`*R*, which may contain initialized data or code. There is also a section named `.bss.`*R* for zero-initialized data (BSS), which is placed after the initialized data section for this region. When you use the Sourcery G++ Board Builder to create a custom board definition, the generated linker script provides exactly the same mapping from program sections to memory regions for regions you create yourself using the Memory Map editor, as for the memory regions in CS3's predefined linker scripts.

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

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

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

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

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

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

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

Regions documented as "Other regions" in the tables in Section 6.5, “Supported Boards for ARM EABI” do not have corresponding program sections. Typically, these regions contain memory-mapped control and I/O registers and cannot be used for general data or program storage. If your program needs to manipulate data in these regions, you can use the CS3 memory map access interface declared in `cs3.h`, as described in Section 6.3.2, “Programmatic Access to the CS3 Memory Map”.

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

6.3.2. Programmatic Access to the CS3 Memory Map

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

For each region named *R*, `cs3.h` declares a byte array variable `__cs3_region_start_`*R* at the region's start address, and a `size_t` variable `__cs3_region_size_`*R* to represent the total size of the region. These symbols are defined by the linker script and so may also be referenced from assembly language. Note that all regions are aligned on eight-byte boundaries and sizes are also multiples of eight bytes.

For memory regions that can correspond to program sections (as described in Section 6.3.1, “Memory Regions and Program Sections”), there are additional symbols `__cs3_region_init_`*R* and `__cs3_region_init_size_`*R* that describe constant data used to initialize the region. During the C initialization phase (Section 6.2, “Program Startup and Termination”), this data is copied into the lower part of the memory region. The symbol `__cs3_region_zero_size_`*R* represents the size of the zero-initialized `.bss.`*R* section following the initialized data. Any of these identifiers may actually be defined as a preprocessor macro that expands to an expression of the appropriate type and value.

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

6.3.3. Heap and Stack Placement

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

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

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

The default initial stack pointer for bare-metal profiles is given by the symbol `__cs3_stack`. Stack initialization is discussed in more detail in Section 6.2.2, “The Assembly Initialization Phase”.

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

6.4. Interrupt Vectors and Handlers

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

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

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

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

6.4.1. ARM EABI Interrupt Vector Implementation

On address vector processors, the CS3 library provides an array of pointers to interrupt handlers named `__cs3_interrupt_vector_`*`form`*, where *`form`* identifies the particular processor variant the vector is appropriate for. Each entry in the vector holds a reference to a symbol named `__cs3_isr_`*`name`*, where *`name`* is the customary name of that interrupt on the processor, or a number if there is no consistently used name. You can find the interrupt vector details in Section 6.6, “Interrupt Vector Tables”. The particular vector used by a given CS3-supported board is documented in the tables in Section 6.5, “Supported Boards for ARM EABI”.

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

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

To override the entire interrupt vector, you can define `__cs3_interrupt_vector_`*`form`*. You must place this definition in a section named `.cs3.interrupt_vector`. The linker script reports an error if the `.cs3.interrupt_vector` section is empty, to ensure that the definition of `__cs3_interrupt_vector_`*`form`* occupies the proper section.

You may define the vector in C with an array of pointers using the `section` attribute to place it in the appropriate section. For example, to override the interrupt vector on Actel CoreMP7 Cortex-M1 boards, make the following definition:

```
typedef void handler(void);
handler *__attribute__((section (".cs3.interrupt_vector")))
  __cs3_interrupt_vector_micro[] =
{ ... };
```

6.4.2. Writing Interrupt Handlers

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

As an alternative to writing interrupt handlers in assembly language, on ARM targets they may be written in C using the `interrupt` attribute. This tells the compiler to generate appropriate function entry and exit sequences for an interrupt handler. For example, to override the `__cs3_isr_nmi` handler, use the following definition:

```
void __attribute__ ((interrupt)) __cs3_isr_nmi (void)
{
  ... custom handler code ...
}
```

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

6.5. Supported Boards for ARM EABI

CS3 provides support for the following boards on ARM EABI targets. Note that you can use the Board Builder in the Sourcery G++ IDE to define additional custom boards using CS3. See Section 4.4.1, “Using the Sourcery G++ Board Builder” for additional information.

Actel CoreMP7 Cortex-M1		
Processor name:	Cortex-M1	
Processor options:	`-mcpu=cortex-m1 -mthumb`	
Memory regions:	`ram` (SRAM), `rom` (Flash)	
Interrupt vector:	`__cs3_interrupt_vector_micro`	
Linker scripts:	RAM Hosted	`coremp7-cm1-ram-hosted.ld`
	RAM Unhosted	`coremp7-cm1-ram.ld`
	ROM Hosted	`coremp7-cm1-rom-hosted.ld`
	ROM Unhosted	`coremp7-cm1-rom.ld`

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

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

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

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

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

Atmel AT91SAM7S		
Processor name:	ARM7TDMI	
Processor options:	`-mcpu=arm7tdmi`	
Memory regions:	`rom` (Flash), `ram` (SRAM)	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	RAM Hosted	`at91sam7s-ek-ram-hosted.ld`
	RAM Unhosted	`at91sam7s-ek-ram.ld`
	ROM Hosted	`at91sam7s-ek-rom-hosted.ld`
	ROM Unhosted	`at91sam7s-ek-rom.ld`

Energy Micro EFM32-G2XX-DK		
Processor name:	EFM32G290F128	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`rom` (128K Flash ROM), `ram` (16K RAM)	
Interrupt vector:	`__cs3_interrupt_vector_efm32g`	
Linker scripts:	RAM Hosted	`efm32-g2xx-dk-ram-hosted.ld`
	RAM Unhosted	`efm32-g2xx-dk-ram.ld`
	ROM Hosted	`efm32-g2xx-dk-rom-hosted.ld`
	ROM Unhosted	`efm32-g2xx-dk-rom.ld`

Energy Micro EFM32-G8XX-DK		
Processor name:	EFM32G890F128	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`rom` (128K Flash ROM), `ram` (16K RAM)	
Interrupt vector:	`__cs3_interrupt_vector_efm32g`	
Linker scripts:	RAM Hosted	`efm32-g8xx-dk-ram-hosted.ld`
	RAM Unhosted	`efm32-g8xx-dk-ram.ld`
	ROM Hosted	`efm32-g8xx-dk-rom-hosted.ld`
	ROM Unhosted	`efm32-g8xx-dk-rom.ld`

Energy Micro EFM32-G8XX-STK		
Processor name:	EFM32G890F128	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`rom` (128K Flash ROM), `ram` (16K RAM)	
Interrupt vector:	`__cs3_interrupt_vector_efm32g`	
Linker scripts:	RAM Hosted	`efm32-g8xx-stk-ram-hosted.ld`
	RAM Unhosted	`efm32-g8xx-stk-ram.ld`
	ROM Hosted	`efm32-g8xx-stk-rom-hosted.ld`
	ROM Unhosted	`efm32-g8xx-stk-rom.ld`

Freescale i.MX233 (with Mobile DDR)		
Processor name:	Freescale i.MX23	
Processor options:	`-mcpu=arm926ej-s`	
Memory regions:	`ram` (128MB SDRAM (Mobile DDR)), `ocram` (On-Chip RAM)	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	RAM Hosted	`imx233mddr-ram-hosted.ld`
	RAM Unhosted	`imx233mddr-ram.ld`
	On-Chip RAM Hosted	`imx233mddr-ocram-hosted.ld`
	On-Chip RAM Unhosted	`imx233mddr-ocram.ld`

Freescale i.MX233 EVK		
Processor name:	Freescale i.MX23	
Processor options:	`-mcpu=arm926ej-s`	
Memory regions:	`ram` (128MB SDRAM (DDR1)), `ocram` (On-Chip RAM)	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	RAM Hosted	`imx233evk-ram-hosted.ld`
	RAM Unhosted	`imx233evk-ram.ld`
	On-Chip RAM Hosted	`imx233evk-ocram-hosted.ld`
	On-Chip RAM Unhosted	`imx233evk-ocram.ld`

Freescale i.MX31 ADS		
Processor name:	ARM1136JF-S	
Processor options:	`-mcpu=arm1136jf-s`	
Memory regions:	`ram` (128MB SDRAM), `rom` (32MB NOR Flash), `internalram` (16K Internal RAM)	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	RAM Hosted	`imx31-ram-hosted.ld`
	RAM Unhosted	`imx31-ram.ld`
	ROM Hosted	`imx31-rom-hosted.ld`
	ROM Unhosted	`imx31-rom.ld`

Freescale TWR-K40X256		
Processor name:	Freescale MK40X256Vxx100	
Processor options:	`-mcpu=cortex-m4 -mthumb`	
Memory regions:	`rom` (256 KBytes Program Flash), `flexnvm` (256 KBytes FlexNVM), `flexram` (4 KBytes FlexRAM), `sram_l` (32 KBytes Internal SRAM_L), `ram` (32 KBytes Internal SRAM_U)	
Interrupt vector:	`__cs3_interrupt_vector_kinetis`	
Linker scripts:	RAM Hosted	`twr-k40x256-ram-hosted.ld`
	RAM Unhosted	`twr-k40x256-ram.ld`
	ROM Hosted	`twr-k40x256-rom-hosted.ld`
	ROM Unhosted	`twr-k40x256-rom.ld`

Freescale TWR-K60N512		
Processor name:	Freescale MK60N512Vxx100	
Processor options:	`-mcpu=cortex-m4 -mthumb`	
Memory regions:	`rom` (512 KBytes Program Flash), `sram_l` (64 KBytes Internal SRAM_L), `ram` (64 KBytes Internal SRAM_U)	
Interrupt vector:	`__cs3_interrupt_vector_kinetis`	
Linker scripts:	RAM Hosted	`twr-k60n512-ram-hosted.ld`
	RAM Unhosted	`twr-k60n512-ram.ld`
	ROM Hosted	`twr-k60n512-rom-hosted.ld`
	ROM Unhosted	`twr-k60n512-rom.ld`

Keil MCB1760		
Processor name:	NXP LPC1768	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`rom` (512 KBytes Internal Flash), `ram` (32 KBytes Internal SRAM), `ahbsram0` (16 KBytes Internal SRAM), `ahbsram1` (16 KBytes Internal SRAM)	
Interrupt vector:	`__cs3_interrupt_vector_lpc17xx`	
Linker scripts:	RAM Hosted	`mcb1760-ram-hosted.ld`
	RAM Unhosted	`mcb1760-ram.ld`
	ROM Hosted	`mcb1760-rom-hosted.ld`
	ROM Unhosted	`mcb1760-rom.ld`

Keil MCB2100		
Processor name:	NXP LPC21xx	
Processor options:	`-mcpu=arm7tdmi-s`	
Memory regions:	`rom` (256 KBytes Internal Flash), `ram` (16 KBytes Internal SRAM)	
Interrupt vector:	`__cs3_interrupt_vector_lpc21xx`	
Linker scripts:	RAM Hosted	`mcb2100-ram-hosted.ld`
	RAM Unhosted	`mcb2100-ram.ld`
	ROM Hosted	`mcb2100-rom-hosted.ld`
	ROM Unhosted	`mcb2100-rom.ld`

Keil MCB2130		
Processor name:	NXP LPC21xx	
Processor options:	`-mcpu=arm7tdmi-s`	
Memory regions:	`rom` (512 KBytes Internal Flash), `ram` (32 KBytes Internal SRAM)	
Interrupt vector:	`__cs3_interrupt_vector_lpc21xx`	
Linker scripts:	RAM Hosted	`mcb2130-ram-hosted.ld`
	RAM Unhosted	`mcb2130-ram.ld`
	ROM Hosted	`mcb2130-rom-hosted.ld`
	ROM Unhosted	`mcb2130-rom.ld`

Keil MCB2140		
Processor name:	NXP LPC21xx	
Processor options:	`-mcpu=arm7tdmi-s`	
Memory regions:	`rom` (512 KBytes Internal Flash), `ram` (32 KBytes Internal SRAM)	
Interrupt vector:	`__cs3_interrupt_vector_lpc21xx`	
Linker scripts:	RAM Hosted	`mcb2140-ram-hosted.ld`
	RAM Unhosted	`mcb2140-ram.ld`
	ROM Hosted	`mcb2140-rom-hosted.ld`
	ROM Unhosted	`mcb2140-rom.ld`

Keil MCB2470		
Processor name:	NXP LPC21xx	
Processor options:	`-mcpu=arm7tdmi-s`	
Memory regions:	`rom` (512 KBytes Internal Flash), `ram` (64 KBytes Internal SRAM), `extnor` (External NOR Flash), `extsdram` (32 MBytes External SDRAM)	
Other regions:	`extnand` (External NAND Flash)	
Interrupt vector:	`__cs3_interrupt_vector_lpc21xx`	
Linker scripts:	RAM Hosted	`mcb2470-ram-hosted.ld`
	RAM Unhosted	`mcb2470-ram.ld`
	ROM Hosted	`mcb2470-rom-hosted.ld`
	ROM Unhosted	`mcb2470-rom.ld`

Keil MCBSTM32		
Processor name:	STM32F103RB	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (Internal SRAM), `rom` (Internal Flash), `option_bytes_rom` (Option Bytes)	
Interrupt vector:	`__cs3_interrupt_vector_stm32f10`	
Linker scripts:	RAM Hosted	`mcbstm32-ram-hosted.ld`
	RAM Unhosted	`mcbstm32-ram.ld`
	ROM Hosted	`mcbstm32-rom-hosted.ld`
	ROM Unhosted	`mcbstm32-rom.ld`

Keil MCBSTR7 (flash boot)		
Processor name:	ARM7TDMI	
Processor options:	`-mcpu=arm7tdmi`	
Memory regions:	`ram` (64 kBytes Internal SRAM), `rom` (256 kBytes Internal Flash), `datarom` (16 kBytes Internal Flash)	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	RAM Hosted	`str710-flashboot-ram-hosted.ld`
	RAM Unhosted	`str710-flashboot-ram.ld`
	ROM Hosted	`str710-flashboot-rom-hosted.ld`
	ROM Unhosted	`str710-flashboot-rom.ld`

Keil MCBSTR7 (ram boot)		
Processor name:	ARM7TDMI	
Processor options:	`-mcpu=arm7tdmi`	
Memory regions:	`ram` (64 kBytes Internal SRAM), `rom` (256 kBytes Internal Flash), `datarom` (16 kBytes Internal Flash)	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	RAM Hosted	`str710-ramboot-ram-hosted.ld`
	RAM Unhosted	`str710-ramboot-ram.ld`
	ROM Hosted	`str710-ramboot-rom-hosted.ld`
	ROM Unhosted	`str710-ramboot-rom.ld`

Keil MCBSTR9		
Processor name:	ARM966E-S	
Processor options:	`-mcpu=arm966e-s`	
Memory regions:	`ram` (96 kBytes Internal SRAM), `rom` (512 kBytes Internal Flash), `nbrom` (32 kBytes Internal Flash)	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	RAM Hosted	`str91x-ram-hosted.ld`
	RAM Unhosted	`str91x-ram.ld`
	ROM Hosted	`str91x-rom-hosted.ld`
	ROM Unhosted	`str91x-rom.ld`

Keil Microcontroller Prototyping System (Cortex-M0)		
Processor name:	Cortex-M0	
Processor options:	`-mcpu=cortex-m0 -mthumb`	
Memory regions:	`rom` (64MB NOR Flash), `ssram1` (4MB SRAM (SSRAM1)), `ram` (4MB SRAM (SSRAM0))	
Interrupt vector:	`__cs3_interrupt_vector_micro`	
Linker scripts:	ROM Hosted	`mps-cm0-rom-hosted.ld`
	ROM Unhosted	`mps-cm0-rom.ld`
	RAM Hosted	`mps-cm0-ram-hosted.ld`
	RAM Unhosted	`mps-cm0-ram.ld`
	SSRAM1 Hosted	`mps-cm0-ssram1-hosted.ld`
	SSRAM1 Unhosted	`mps-cm0-ssram1.ld`

Keil Microcontroller Prototyping System (Cortex-M1)		
Processor name:	Cortex-M1	
Processor options:	`-mcpu=cortex-m1 -mthumb`	
Memory regions:	`rom` (64MB NOR Flash), `ssram1` (4MB SRAM (SSRAM1)), `ram` (4MB SRAM (SSRAM0))	
Interrupt vector:	`__cs3_interrupt_vector_micro`	
Linker scripts:	ROM Hosted	`mps-cm1-rom-hosted.ld`
	ROM Unhosted	`mps-cm1-rom.ld`
	RAM Hosted	`mps-cm1-ram-hosted.ld`
	RAM Unhosted	`mps-cm1-ram.ld`
	SSRAM1 Hosted	`mps-cm1-ssram1-hosted.ld`
	SSRAM1 Unhosted	`mps-cm1-ssram1.ld`

Keil Microcontroller Prototyping System (Cortex-M3)		
Processor name:	Cortex-M3	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`rom` (64MB NOR Flash), `ssram1` (4MB SRAM (SSRAM1)), `ram` (4MB SRAM (SSRAM0))	
Interrupt vector:	`__cs3_interrupt_vector_micro`	
Linker scripts:	ROM Hosted	`mps-cm3-rom-hosted.ld`
	ROM Unhosted	`mps-cm3-rom.ld`
	RAM Hosted	`mps-cm3-ram-hosted.ld`
	RAM Unhosted	`mps-cm3-ram.ld`
	SSRAM1 Hosted	`mps-cm3-ssram1-hosted.ld`
	SSRAM1 Unhosted	`mps-cm3-ssram1.ld`

Keil Microcontroller Prototyping System (Cortex-M4)		
Processor name:	Cortex-M4	
Processor options:	`-mcpu=cortex-m4 -mthumb`	
Memory regions:	`rom` (64MB NOR Flash), `ssram1` (4MB SRAM (SSRAM1)), `ram` (4MB SRAM (SSRAM0))	
Interrupt vector:	`__cs3_interrupt_vector_micro`	
Linker scripts:	ROM Hosted	`mps-cm4-rom-hosted.ld`
	ROM Unhosted	`mps-cm4-rom.ld`
	RAM Hosted	`mps-cm4-ram-hosted.ld`
	RAM Unhosted	`mps-cm4-ram.ld`
	SSRAM1 Hosted	`mps-cm4-ssram1-hosted.ld`
	SSRAM1 Unhosted	`mps-cm4-ssram1.ld`

PHYTEC phyCore-LPC3250		
Processor name:	ARM926EJ-S with VFP	
Processor options:	`-mcpu=arm926ej-s`	
Memory regions:	`extram` (64MB external SDRAM), `extrom` (2MB external NOR Flash), `ram` (256K Internal RAM at reset with default Boot Map control register settings), `remappedram` (256K Internal RAM after remapping by setting bit0 = 1 in the Boot Map control register @ 0x40004014)	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	RAM Hosted	`phycore-lpc3250-ram-hosted.ld`
	RAM Unhosted	`phycore-lpc3250-ram.ld`

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

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

QEMU Stellaris Simulator		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram`	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	Simulator Hosted	`qemu-luminary-hosted.ld`
	Simulator Unhosted	`qemu-luminary.ld`

RealView EB Cortex-M1		
Processor name:	Cortex-M1	
Processor options:	`-mcpu=cortex-m1 -mthumb`	
Memory regions:	`ram` (2Mb RAM), `rom` (2Mb ROM)	
Interrupt vector:	`__cs3_interrupt_vector_micro`	
Linker scripts:	RAM Hosted	`realview-cm1-ram-hosted.ld`
	RAM Unhosted	`realview-cm1-ram.ld`
	ROM Hosted	`realview-cm1-rom-hosted.ld`
	ROM Unhosted	`realview-cm1-rom.ld`

Stellaris LM3S101		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (2K RAM), `rom` (8K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s101-ram-hosted.ld`
	RAM Unhosted	`lm3s101-ram.ld`
	ROM Hosted	`lm3s101-rom-hosted.ld`
	ROM Unhosted	`lm3s101-rom.ld`

Stellaris LM3S102		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (2K RAM), `rom` (8K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s102-ram-hosted.ld`
	RAM Unhosted	`lm3s102-ram.ld`
	ROM Hosted	`lm3s102-rom-hosted.ld`
	ROM Unhosted	`lm3s102-rom.ld`

Stellaris LM3S1110		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1110-ram-hosted.ld`
	RAM Unhosted	`lm3s1110-ram.ld`
	ROM Hosted	`lm3s1110-rom-hosted.ld`
	ROM Unhosted	`lm3s1110-rom.ld`

Stellaris LM3S1133		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1133-ram-hosted.ld`
	RAM Unhosted	`lm3s1133-ram.ld`
	ROM Hosted	`lm3s1133-rom-hosted.ld`
	ROM Unhosted	`lm3s1133-rom.ld`

Stellaris LM3S1138		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1138-ram-hosted.ld`
	RAM Unhosted	`lm3s1138-ram.ld`
	ROM Hosted	`lm3s1138-rom-hosted.ld`
	ROM Unhosted	`lm3s1138-rom.ld`

Stellaris LM3S1150		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1150-ram-hosted.ld`
	RAM Unhosted	`lm3s1150-ram.ld`
	ROM Hosted	`lm3s1150-rom-hosted.ld`
	ROM Unhosted	`lm3s1150-rom.ld`

Stellaris LM3S1162		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1162-ram-hosted.ld`
	RAM Unhosted	`lm3s1162-ram.ld`
	ROM Hosted	`lm3s1162-rom-hosted.ld`
	ROM Unhosted	`lm3s1162-rom.ld`

Stellaris LM3S1165		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1165-ram-hosted.ld`
	RAM Unhosted	`lm3s1165-ram.ld`
	ROM Hosted	`lm3s1165-rom-hosted.ld`
	ROM Unhosted	`lm3s1165-rom.ld`

Stellaris LM3S1332		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1332-ram-hosted.ld`
	RAM Unhosted	`lm3s1332-ram.ld`
	ROM Hosted	`lm3s1332-rom-hosted.ld`
	ROM Unhosted	`lm3s1332-rom.ld`

Stellaris LM3S1435		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1435-ram-hosted.ld`
	RAM Unhosted	`lm3s1435-ram.ld`
	ROM Hosted	`lm3s1435-rom-hosted.ld`
	ROM Unhosted	`lm3s1435-rom.ld`

Stellaris LM3S1439		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1439-ram-hosted.ld`
	RAM Unhosted	`lm3s1439-ram.ld`
	ROM Hosted	`lm3s1439-rom-hosted.ld`
	ROM Unhosted	`lm3s1439-rom.ld`

Stellaris LM3S1512		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1512-ram-hosted.ld`
	RAM Unhosted	`lm3s1512-ram.ld`
	ROM Hosted	`lm3s1512-rom-hosted.ld`
	ROM Unhosted	`lm3s1512-rom.ld`

Stellaris LM3S1538		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1538-ram-hosted.ld`
	RAM Unhosted	`lm3s1538-ram.ld`
	ROM Hosted	`lm3s1538-rom-hosted.ld`
	ROM Unhosted	`lm3s1538-rom.ld`

Stellaris LM3S1601		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1601-ram-hosted.ld`
	RAM Unhosted	`lm3s1601-ram.ld`
	ROM Hosted	`lm3s1601-rom-hosted.ld`
	ROM Unhosted	`lm3s1601-rom.ld`

Stellaris LM3S1607		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s1607-ram-hosted.ld`
	RAM Unhosted	`lm3s1607-ram.ld`
	ROM Hosted	`lm3s1607-rom-hosted.ld`
	ROM Unhosted	`lm3s1607-rom.ld`

Stellaris LM3S1608		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1608-ram-hosted.ld`
	RAM Unhosted	`lm3s1608-ram.ld`
	ROM Hosted	`lm3s1608-rom-hosted.ld`
	ROM Unhosted	`lm3s1608-rom.ld`

Stellaris LM3S1620		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1620-ram-hosted.ld`
	RAM Unhosted	`lm3s1620-ram.ld`
	ROM Hosted	`lm3s1620-rom-hosted.ld`
	ROM Unhosted	`lm3s1620-rom.ld`

Stellaris LM3S1621		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1621-ram-hosted.ld`
	RAM Unhosted	`lm3s1621-ram.ld`
	ROM Hosted	`lm3s1621-rom-hosted.ld`
	ROM Unhosted	`lm3s1621-rom.ld`

Stellaris LM3S1625		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s1625-ram-hosted.ld`
	RAM Unhosted	`lm3s1625-ram.ld`
	ROM Hosted	`lm3s1625-rom-hosted.ld`
	ROM Unhosted	`lm3s1625-rom.ld`

Stellaris LM3S1626		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s1626-ram-hosted.ld`
	RAM Unhosted	`lm3s1626-ram.ld`
	ROM Hosted	`lm3s1626-rom-hosted.ld`
	ROM Unhosted	`lm3s1626-rom.ld`

Stellaris LM3S1627		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s1627-ram-hosted.ld`
	RAM Unhosted	`lm3s1627-ram.ld`
	ROM Hosted	`lm3s1627-rom-hosted.ld`
	ROM Unhosted	`lm3s1627-rom.ld`

Stellaris LM3S1635		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1635-ram-hosted.ld`
	RAM Unhosted	`lm3s1635-ram.ld`
	ROM Hosted	`lm3s1635-rom-hosted.ld`
	ROM Unhosted	`lm3s1635-rom.ld`

Stellaris LM3S1637		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1637-ram-hosted.ld`
	RAM Unhosted	`lm3s1637-ram.ld`
	ROM Hosted	`lm3s1637-rom-hosted.ld`
	ROM Unhosted	`lm3s1637-rom.ld`

Stellaris LM3S1651		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1651-ram-hosted.ld`
	RAM Unhosted	`lm3s1651-ram.ld`
	ROM Hosted	`lm3s1651-rom-hosted.ld`
	ROM Unhosted	`lm3s1651-rom.ld`

Stellaris LM3S1751		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1751-ram-hosted.ld`
	RAM Unhosted	`lm3s1751-ram.ld`
	ROM Hosted	`lm3s1751-rom-hosted.ld`
	ROM Unhosted	`lm3s1751-rom.ld`

Stellaris LM3S1776		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s1776-ram-hosted.ld`
	RAM Unhosted	`lm3s1776-ram.ld`
	ROM Hosted	`lm3s1776-rom-hosted.ld`
	ROM Unhosted	`lm3s1776-rom.ld`

Stellaris LM3S1811		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1811-ram-hosted.ld`
	RAM Unhosted	`lm3s1811-ram.ld`
	ROM Hosted	`lm3s1811-rom-hosted.ld`
	ROM Unhosted	`lm3s1811-rom.ld`

Stellaris LM3S1816		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1816-ram-hosted.ld`
	RAM Unhosted	`lm3s1816-ram.ld`
	ROM Hosted	`lm3s1816-rom-hosted.ld`
	ROM Unhosted	`lm3s1816-rom.ld`

Stellaris LM3S1850		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1850-ram-hosted.ld`
	RAM Unhosted	`lm3s1850-ram.ld`
	ROM Hosted	`lm3s1850-rom-hosted.ld`
	ROM Unhosted	`lm3s1850-rom.ld`

Stellaris LM3S1911		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1911-ram-hosted.ld`
	RAM Unhosted	`lm3s1911-ram.ld`
	ROM Hosted	`lm3s1911-rom-hosted.ld`
	ROM Unhosted	`lm3s1911-rom.ld`

Stellaris LM3S1918		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1918-ram-hosted.ld`
	RAM Unhosted	`lm3s1918-ram.ld`
	ROM Hosted	`lm3s1918-rom-hosted.ld`
	ROM Unhosted	`lm3s1918-rom.ld`

Stellaris LM3S1937		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1937-ram-hosted.ld`
	RAM Unhosted	`lm3s1937-ram.ld`
	ROM Hosted	`lm3s1937-rom-hosted.ld`
	ROM Unhosted	`lm3s1937-rom.ld`

Stellaris LM3S1958		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1958-ram-hosted.ld`
	RAM Unhosted	`lm3s1958-ram.ld`
	ROM Hosted	`lm3s1958-rom-hosted.ld`
	ROM Unhosted	`lm3s1958-rom.ld`

Stellaris LM3S1960		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1960-ram-hosted.ld`
	RAM Unhosted	`lm3s1960-ram.ld`
	ROM Hosted	`lm3s1960-rom-hosted.ld`
	ROM Unhosted	`lm3s1960-rom.ld`

Stellaris LM3S1968		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s1968-ram-hosted.ld`
	RAM Unhosted	`lm3s1968-ram.ld`
	ROM Hosted	`lm3s1968-rom-hosted.ld`
	ROM Unhosted	`lm3s1968-rom.ld`

Stellaris LM3S1B21		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (96K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1b21-ram-hosted.ld`
	RAM Unhosted	`lm3s1b21-ram.ld`
	ROM Hosted	`lm3s1b21-rom-hosted.ld`
	ROM Unhosted	`lm3s1b21-rom.ld`

Stellaris LM3S1J11		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (20K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1j11-ram-hosted.ld`
	RAM Unhosted	`lm3s1j11-ram.ld`
	ROM Hosted	`lm3s1j11-rom-hosted.ld`
	ROM Unhosted	`lm3s1j11-rom.ld`

Stellaris LM3S1J16		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (20K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1j16-ram-hosted.ld`
	RAM Unhosted	`lm3s1j16-ram.ld`
	ROM Hosted	`lm3s1j16-rom-hosted.ld`
	ROM Unhosted	`lm3s1j16-rom.ld`

Stellaris LM3S1N11		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (12K RAM), `rom` (64K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1n11-ram-hosted.ld`
	RAM Unhosted	`lm3s1n11-ram.ld`
	ROM Hosted	`lm3s1n11-rom-hosted.ld`
	ROM Unhosted	`lm3s1n11-rom.ld`

Stellaris LM3S1N16		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (12K RAM), `rom` (64K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1n16-ram-hosted.ld`
	RAM Unhosted	`lm3s1n16-ram.ld`
	ROM Hosted	`lm3s1n16-rom-hosted.ld`
	ROM Unhosted	`lm3s1n16-rom.ld`

Stellaris LM3S1P51		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (24K RAM), `rom` (64K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1p51-ram-hosted.ld`
	RAM Unhosted	`lm3s1p51-ram.ld`
	ROM Hosted	`lm3s1p51-rom-hosted.ld`
	ROM Unhosted	`lm3s1p51-rom.ld`

Stellaris LM3S1R21		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (48K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1r21-ram-hosted.ld`
	RAM Unhosted	`lm3s1r21-ram.ld`
	ROM Hosted	`lm3s1r21-rom-hosted.ld`
	ROM Unhosted	`lm3s1r21-rom.ld`

Stellaris LM3S1R26		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (48K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1r26-ram-hosted.ld`
	RAM Unhosted	`lm3s1r26-ram.ld`
	ROM Hosted	`lm3s1r26-rom-hosted.ld`
	ROM Unhosted	`lm3s1r26-rom.ld`

Stellaris LM3S1W16		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1w16-ram-hosted.ld`
	RAM Unhosted	`lm3s1w16-ram.ld`
	ROM Hosted	`lm3s1w16-rom-hosted.ld`
	ROM Unhosted	`lm3s1w16-rom.ld`

Stellaris LM3S1Z16		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (6K RAM), `rom` (16K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s1z16-ram-hosted.ld`
	RAM Unhosted	`lm3s1z16-ram.ld`
	ROM Hosted	`lm3s1z16-rom-hosted.ld`
	ROM Unhosted	`lm3s1z16-rom.ld`

Stellaris LM3S2110		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2110-ram-hosted.ld`
	RAM Unhosted	`lm3s2110-ram.ld`
	ROM Hosted	`lm3s2110-rom-hosted.ld`
	ROM Unhosted	`lm3s2110-rom.ld`

Stellaris LM3S2139		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2139-ram-hosted.ld`
	RAM Unhosted	`lm3s2139-ram.ld`
	ROM Hosted	`lm3s2139-rom-hosted.ld`
	ROM Unhosted	`lm3s2139-rom.ld`

Stellaris LM3S2276		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (64K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s2276-ram-hosted.ld`
	RAM Unhosted	`lm3s2276-ram.ld`
	ROM Hosted	`lm3s2276-rom-hosted.ld`
	ROM Unhosted	`lm3s2276-rom.ld`

Stellaris LM3S2410		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2410-ram-hosted.ld`
	RAM Unhosted	`lm3s2410-ram.ld`
	ROM Hosted	`lm3s2410-rom-hosted.ld`
	ROM Unhosted	`lm3s2410-rom.ld`

Stellaris LM3S2412		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2412-ram-hosted.ld`
	RAM Unhosted	`lm3s2412-ram.ld`
	ROM Hosted	`lm3s2412-rom-hosted.ld`
	ROM Unhosted	`lm3s2412-rom.ld`

Stellaris LM3S2432		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2432-ram-hosted.ld`
	RAM Unhosted	`lm3s2432-ram.ld`
	ROM Hosted	`lm3s2432-rom-hosted.ld`
	ROM Unhosted	`lm3s2432-rom.ld`

Stellaris LM3S2533		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2533-ram-hosted.ld`
	RAM Unhosted	`lm3s2533-ram.ld`
	ROM Hosted	`lm3s2533-rom-hosted.ld`
	ROM Unhosted	`lm3s2533-rom.ld`

Stellaris LM3S2601		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2601-ram-hosted.ld`
	RAM Unhosted	`lm3s2601-ram.ld`
	ROM Hosted	`lm3s2601-rom-hosted.ld`
	ROM Unhosted	`lm3s2601-rom.ld`

Stellaris LM3S2608		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2608-ram-hosted.ld`
	RAM Unhosted	`lm3s2608-ram.ld`
	ROM Hosted	`lm3s2608-rom-hosted.ld`
	ROM Unhosted	`lm3s2608-rom.ld`

Stellaris LM3S2616		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s2616-ram-hosted.ld`
	RAM Unhosted	`lm3s2616-ram.ld`
	ROM Hosted	`lm3s2616-rom-hosted.ld`
	ROM Unhosted	`lm3s2616-rom.ld`

Stellaris LM3S2620		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2620-ram-hosted.ld`
	RAM Unhosted	`lm3s2620-ram.ld`
	ROM Hosted	`lm3s2620-rom-hosted.ld`
	ROM Unhosted	`lm3s2620-rom.ld`

Stellaris LM3S2637		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2637-ram-hosted.ld`
	RAM Unhosted	`lm3s2637-ram.ld`
	ROM Hosted	`lm3s2637-rom-hosted.ld`
	ROM Unhosted	`lm3s2637-rom.ld`

Stellaris LM3S2651		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2651-ram-hosted.ld`
	RAM Unhosted	`lm3s2651-ram.ld`
	ROM Hosted	`lm3s2651-rom-hosted.ld`
	ROM Unhosted	`lm3s2651-rom.ld`

Stellaris LM3S2671		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s2671-ram-hosted.ld`
	RAM Unhosted	`lm3s2671-ram.ld`
	ROM Hosted	`lm3s2671-rom-hosted.ld`
	ROM Unhosted	`lm3s2671-rom.ld`

Stellaris LM3S2678		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s2678-ram-hosted.ld`
	RAM Unhosted	`lm3s2678-ram.ld`
	ROM Hosted	`lm3s2678-rom-hosted.ld`
	ROM Unhosted	`lm3s2678-rom.ld`

Stellaris LM3S2730		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2730-ram-hosted.ld`
	RAM Unhosted	`lm3s2730-ram.ld`
	ROM Hosted	`lm3s2730-rom-hosted.ld`
	ROM Unhosted	`lm3s2730-rom.ld`

Stellaris LM3S2739		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2739-ram-hosted.ld`
	RAM Unhosted	`lm3s2739-ram.ld`
	ROM Hosted	`lm3s2739-rom-hosted.ld`
	ROM Unhosted	`lm3s2739-rom.ld`

Stellaris LM3S2776		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s2776-ram-hosted.ld`
	RAM Unhosted	`lm3s2776-ram.ld`
	ROM Hosted	`lm3s2776-rom-hosted.ld`
	ROM Unhosted	`lm3s2776-rom.ld`

Stellaris LM3S2793		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s2793-ram-hosted.ld`
	RAM Unhosted	`lm3s2793-ram.ld`
	ROM Hosted	`lm3s2793-rom-hosted.ld`
	ROM Unhosted	`lm3s2793-rom.ld`

Stellaris LM3S2911		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2911-ram-hosted.ld`
	RAM Unhosted	`lm3s2911-ram.ld`
	ROM Hosted	`lm3s2911-rom-hosted.ld`
	ROM Unhosted	`lm3s2911-rom.ld`

Stellaris LM3S2918		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2918-ram-hosted.ld`
	RAM Unhosted	`lm3s2918-ram.ld`
	ROM Hosted	`lm3s2918-rom-hosted.ld`
	ROM Unhosted	`lm3s2918-rom.ld`

Stellaris LM3S2939		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2939-ram-hosted.ld`
	RAM Unhosted	`lm3s2939-ram.ld`
	ROM Hosted	`lm3s2939-rom-hosted.ld`
	ROM Unhosted	`lm3s2939-rom.ld`

Stellaris LM3S2948		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2948-ram-hosted.ld`
	RAM Unhosted	`lm3s2948-ram.ld`
	ROM Hosted	`lm3s2948-rom-hosted.ld`
	ROM Unhosted	`lm3s2948-rom.ld`

Stellaris LM3S2950		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2950-ram-hosted.ld`
	RAM Unhosted	`lm3s2950-ram.ld`
	ROM Hosted	`lm3s2950-rom-hosted.ld`
	ROM Unhosted	`lm3s2950-rom.ld`

Stellaris LM3S2965		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s2965-ram-hosted.ld`
	RAM Unhosted	`lm3s2965-ram.ld`
	ROM Hosted	`lm3s2965-rom-hosted.ld`
	ROM Unhosted	`lm3s2965-rom.ld`

Stellaris LM3S2B93		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (96K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s2b93-ram-hosted.ld`
	RAM Unhosted	`lm3s2b93-ram.ld`
	ROM Hosted	`lm3s2b93-rom-hosted.ld`
	ROM Unhosted	`lm3s2b93-rom.ld`

Stellaris LM3S300		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (4K RAM), `rom` (16K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s300-ram-hosted.ld`
	RAM Unhosted	`lm3s300-ram.ld`
	ROM Hosted	`lm3s300-rom-hosted.ld`
	ROM Unhosted	`lm3s300-rom.ld`

Stellaris LM3S301		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (2K RAM), `rom` (16K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s301-ram-hosted.ld`
	RAM Unhosted	`lm3s301-ram.ld`
	ROM Hosted	`lm3s301-rom-hosted.ld`
	ROM Unhosted	`lm3s301-rom.ld`

Stellaris LM3S308		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (4K RAM), `rom` (16K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s308-ram-hosted.ld`
	RAM Unhosted	`lm3s308-ram.ld`
	ROM Hosted	`lm3s308-rom-hosted.ld`
	ROM Unhosted	`lm3s308-rom.ld`

Stellaris LM3S310		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (4K RAM), `rom` (16K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s310-ram-hosted.ld`
	RAM Unhosted	`lm3s310-ram.ld`
	ROM Hosted	`lm3s310-rom-hosted.ld`
	ROM Unhosted	`lm3s310-rom.ld`

Stellaris LM3S315		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (4K RAM), `rom` (16K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s315-ram-hosted.ld`
	RAM Unhosted	`lm3s315-ram.ld`
	ROM Hosted	`lm3s315-rom-hosted.ld`
	ROM Unhosted	`lm3s315-rom.ld`

Stellaris LM3S316		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (4K RAM), `rom` (16K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s316-ram-hosted.ld`
	RAM Unhosted	`lm3s316-ram.ld`
	ROM Hosted	`lm3s316-rom-hosted.ld`
	ROM Unhosted	`lm3s316-rom.ld`

Stellaris LM3S317		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (4K RAM), `rom` (16K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s317-ram-hosted.ld`
	RAM Unhosted	`lm3s317-ram.ld`
	ROM Hosted	`lm3s317-rom-hosted.ld`
	ROM Unhosted	`lm3s317-rom.ld`

Stellaris LM3S328		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (4K RAM), `rom` (16K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s328-ram-hosted.ld`
	RAM Unhosted	`lm3s328-ram.ld`
	ROM Hosted	`lm3s328-rom-hosted.ld`
	ROM Unhosted	`lm3s328-rom.ld`

Stellaris LM3S3634		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s3634-ram-hosted.ld`
	RAM Unhosted	`lm3s3634-ram.ld`
	ROM Hosted	`lm3s3634-rom-hosted.ld`
	ROM Unhosted	`lm3s3634-rom.ld`

Stellaris LM3S3651		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s3651-ram-hosted.ld`
	RAM Unhosted	`lm3s3651-ram.ld`
	ROM Hosted	`lm3s3651-rom-hosted.ld`
	ROM Unhosted	`lm3s3651-rom.ld`

Stellaris LM3S3739		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s3739-ram-hosted.ld`
	RAM Unhosted	`lm3s3739-ram.ld`
	ROM Hosted	`lm3s3739-rom-hosted.ld`
	ROM Unhosted	`lm3s3739-rom.ld`

Stellaris LM3S3748		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s3748-ram-hosted.ld`
	RAM Unhosted	`lm3s3748-ram.ld`
	ROM Hosted	`lm3s3748-rom-hosted.ld`
	ROM Unhosted	`lm3s3748-rom.ld`

Stellaris LM3S3749		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s3749-ram-hosted.ld`
	RAM Unhosted	`lm3s3749-ram.ld`
	ROM Hosted	`lm3s3749-rom-hosted.ld`
	ROM Unhosted	`lm3s3749-rom.ld`

Stellaris LM3S3826		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s3826-ram-hosted.ld`
	RAM Unhosted	`lm3s3826-ram.ld`
	ROM Hosted	`lm3s3826-rom-hosted.ld`
	ROM Unhosted	`lm3s3826-rom.ld`

Stellaris LM3S3J26		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (20K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s3j26-ram-hosted.ld`
	RAM Unhosted	`lm3s3j26-ram.ld`
	ROM Hosted	`lm3s3j26-rom-hosted.ld`
	ROM Unhosted	`lm3s3j26-rom.ld`

Stellaris LM3S3N26		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (12K RAM), `rom` (64K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s3n26-ram-hosted.ld`
	RAM Unhosted	`lm3s3n26-ram.ld`
	ROM Hosted	`lm3s3n26-rom-hosted.ld`
	ROM Unhosted	`lm3s3n26-rom.ld`

Stellaris LM3S3W26		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s3w26-ram-hosted.ld`
	RAM Unhosted	`lm3s3w26-ram.ld`
	ROM Hosted	`lm3s3w26-rom-hosted.ld`
	ROM Unhosted	`lm3s3w26-rom.ld`

Stellaris LM3S3Z26		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (6K RAM), `rom` (16K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s3z26-ram-hosted.ld`
	RAM Unhosted	`lm3s3z26-ram.ld`
	ROM Hosted	`lm3s3z26-rom-hosted.ld`
	ROM Unhosted	`lm3s3z26-rom.ld`

Stellaris LM3S5632		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s5632-ram-hosted.ld`
	RAM Unhosted	`lm3s5632-ram.ld`
	ROM Hosted	`lm3s5632-rom-hosted.ld`
	ROM Unhosted	`lm3s5632-rom.ld`

Stellaris LM3S5651		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5651-ram-hosted.ld`
	RAM Unhosted	`lm3s5651-ram.ld`
	ROM Hosted	`lm3s5651-rom-hosted.ld`
	ROM Unhosted	`lm3s5651-rom.ld`

Stellaris LM3S5652		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s5652-ram-hosted.ld`
	RAM Unhosted	`lm3s5652-ram.ld`
	ROM Hosted	`lm3s5652-rom-hosted.ld`
	ROM Unhosted	`lm3s5652-rom.ld`

Stellaris LM3S5656		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5656-ram-hosted.ld`
	RAM Unhosted	`lm3s5656-ram.ld`
	ROM Hosted	`lm3s5656-rom-hosted.ld`
	ROM Unhosted	`lm3s5656-rom.ld`

Stellaris LM3S5662		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s5662-ram-hosted.ld`
	RAM Unhosted	`lm3s5662-ram.ld`
	ROM Hosted	`lm3s5662-rom-hosted.ld`
	ROM Unhosted	`lm3s5662-rom.ld`

Stellaris LM3S5732		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s5732-ram-hosted.ld`
	RAM Unhosted	`lm3s5732-ram.ld`
	ROM Hosted	`lm3s5732-rom-hosted.ld`
	ROM Unhosted	`lm3s5732-rom.ld`

Stellaris LM3S5737		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s5737-ram-hosted.ld`
	RAM Unhosted	`lm3s5737-ram.ld`
	ROM Hosted	`lm3s5737-rom-hosted.ld`
	ROM Unhosted	`lm3s5737-rom.ld`

Stellaris LM3S5739		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s5739-ram-hosted.ld`
	RAM Unhosted	`lm3s5739-ram.ld`
	ROM Hosted	`lm3s5739-rom-hosted.ld`
	ROM Unhosted	`lm3s5739-rom.ld`

Stellaris LM3S5747		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s5747-ram-hosted.ld`
	RAM Unhosted	`lm3s5747-ram.ld`
	ROM Hosted	`lm3s5747-rom-hosted.ld`
	ROM Unhosted	`lm3s5747-rom.ld`

Stellaris LM3S5749		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s5749-ram-hosted.ld`
	RAM Unhosted	`lm3s5749-ram.ld`
	ROM Hosted	`lm3s5749-rom-hosted.ld`
	ROM Unhosted	`lm3s5749-rom.ld`

Stellaris LM3S5752		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s5752-ram-hosted.ld`
	RAM Unhosted	`lm3s5752-ram.ld`
	ROM Hosted	`lm3s5752-rom-hosted.ld`
	ROM Unhosted	`lm3s5752-rom.ld`

Stellaris LM3S5762		
Processor name:	Stellaris DustDevil	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (16K Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris`	
Linker scripts:	RAM Hosted	`lm3s5762-ram-hosted.ld`
	RAM Unhosted	`lm3s5762-ram.ld`
	ROM Hosted	`lm3s5762-rom-hosted.ld`
	ROM Unhosted	`lm3s5762-rom.ld`

Stellaris LM3S5791		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5791-ram-hosted.ld`
	RAM Unhosted	`lm3s5791-ram.ld`
	ROM Hosted	`lm3s5791-rom-hosted.ld`
	ROM Unhosted	`lm3s5791-rom.ld`

Stellaris LM3S5951		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5951-ram-hosted.ld`
	RAM Unhosted	`lm3s5951-ram.ld`
	ROM Hosted	`lm3s5951-rom-hosted.ld`
	ROM Unhosted	`lm3s5951-rom.ld`

Stellaris LM3S5956		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (64K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5956-ram-hosted.ld`
	RAM Unhosted	`lm3s5956-ram.ld`
	ROM Hosted	`lm3s5956-rom-hosted.ld`
	ROM Unhosted	`lm3s5956-rom.ld`

Stellaris LM3S5B91		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (96K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5b91-ram-hosted.ld`
	RAM Unhosted	`lm3s5b91-ram.ld`
	ROM Hosted	`lm3s5b91-rom-hosted.ld`
	ROM Unhosted	`lm3s5b91-rom.ld`

Stellaris LM3S5K31		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (24K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5k31-ram-hosted.ld`
	RAM Unhosted	`lm3s5k31-ram.ld`
	ROM Hosted	`lm3s5k31-rom-hosted.ld`
	ROM Unhosted	`lm3s5k31-rom.ld`

Stellaris LM3S5K36		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (24K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5k36-ram-hosted.ld`
	RAM Unhosted	`lm3s5k36-ram.ld`
	ROM Hosted	`lm3s5k36-rom-hosted.ld`
	ROM Unhosted	`lm3s5k36-rom.ld`

Stellaris LM3S5P31		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (24K RAM), `rom` (64K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5p31-ram-hosted.ld`
	RAM Unhosted	`lm3s5p31-ram.ld`
	ROM Hosted	`lm3s5p31-rom-hosted.ld`
	ROM Unhosted	`lm3s5p31-rom.ld`

Stellaris LM3S5P36		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (24K RAM), `rom` (64K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5p36-ram-hosted.ld`
	RAM Unhosted	`lm3s5p36-ram.ld`
	ROM Hosted	`lm3s5p36-rom-hosted.ld`
	ROM Unhosted	`lm3s5p36-rom.ld`

Stellaris LM3S5P51		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (24K RAM), `rom` (64K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5p51-ram-hosted.ld`
	RAM Unhosted	`lm3s5p51-ram.ld`
	ROM Hosted	`lm3s5p51-rom-hosted.ld`
	ROM Unhosted	`lm3s5p51-rom.ld`

Stellaris LM3S5P56		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (24K RAM), `rom` (64K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5p56-ram-hosted.ld`
	RAM Unhosted	`lm3s5p56-ram.ld`
	ROM Hosted	`lm3s5p56-rom-hosted.ld`
	ROM Unhosted	`lm3s5p56-rom.ld`

Stellaris LM3S5R31		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (48K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5r31-ram-hosted.ld`
	RAM Unhosted	`lm3s5r31-ram.ld`
	ROM Hosted	`lm3s5r31-rom-hosted.ld`
	ROM Unhosted	`lm3s5r31-rom.ld`

Stellaris LM3S5R36		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (48K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5r36-ram-hosted.ld`
	RAM Unhosted	`lm3s5r36-ram.ld`
	ROM Hosted	`lm3s5r36-rom-hosted.ld`
	ROM Unhosted	`lm3s5r36-rom.ld`

Stellaris LM3S5T36		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (12K RAM), `rom` (32K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5t36-ram-hosted.ld`
	RAM Unhosted	`lm3s5t36-ram.ld`
	ROM Hosted	`lm3s5t36-rom-hosted.ld`
	ROM Unhosted	`lm3s5t36-rom.ld`

Stellaris LM3S5Y36		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (16K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s5y36-ram-hosted.ld`
	RAM Unhosted	`lm3s5y36-ram.ld`
	ROM Hosted	`lm3s5y36-rom-hosted.ld`
	ROM Unhosted	`lm3s5y36-rom.ld`

Stellaris LM3S600		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s600-ram-hosted.ld`
	RAM Unhosted	`lm3s600-ram.ld`
	ROM Hosted	`lm3s600-rom-hosted.ld`
	ROM Unhosted	`lm3s600-rom.ld`

Stellaris LM3S601		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s601-ram-hosted.ld`
	RAM Unhosted	`lm3s601-ram.ld`
	ROM Hosted	`lm3s601-rom-hosted.ld`
	ROM Unhosted	`lm3s601-rom.ld`

Stellaris LM3S608		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s608-ram-hosted.ld`
	RAM Unhosted	`lm3s608-ram.ld`
	ROM Hosted	`lm3s608-rom-hosted.ld`
	ROM Unhosted	`lm3s608-rom.ld`

Stellaris LM3S610		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s610-ram-hosted.ld`
	RAM Unhosted	`lm3s610-ram.ld`
	ROM Hosted	`lm3s610-rom-hosted.ld`
	ROM Unhosted	`lm3s610-rom.ld`

Stellaris LM3S6100		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6100-ram-hosted.ld`
	RAM Unhosted	`lm3s6100-ram.ld`
	ROM Hosted	`lm3s6100-rom-hosted.ld`
	ROM Unhosted	`lm3s6100-rom.ld`

Stellaris LM3S611		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s611-ram-hosted.ld`
	RAM Unhosted	`lm3s611-ram.ld`
	ROM Hosted	`lm3s611-rom-hosted.ld`
	ROM Unhosted	`lm3s611-rom.ld`

Stellaris LM3S6110		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (16K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6110-ram-hosted.ld`
	RAM Unhosted	`lm3s6110-ram.ld`
	ROM Hosted	`lm3s6110-rom-hosted.ld`
	ROM Unhosted	`lm3s6110-rom.ld`

Stellaris LM3S612		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s612-ram-hosted.ld`
	RAM Unhosted	`lm3s612-ram.ld`
	ROM Hosted	`lm3s612-rom-hosted.ld`
	ROM Unhosted	`lm3s612-rom.ld`

Stellaris LM3S613		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s613-ram-hosted.ld`
	RAM Unhosted	`lm3s613-ram.ld`
	ROM Hosted	`lm3s613-rom-hosted.ld`
	ROM Unhosted	`lm3s613-rom.ld`

Stellaris LM3S615		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s615-ram-hosted.ld`
	RAM Unhosted	`lm3s615-ram.ld`
	ROM Hosted	`lm3s615-rom-hosted.ld`
	ROM Unhosted	`lm3s615-rom.ld`

Stellaris LM3S617		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s617-ram-hosted.ld`
	RAM Unhosted	`lm3s617-ram.ld`
	ROM Hosted	`lm3s617-rom-hosted.ld`
	ROM Unhosted	`lm3s617-rom.ld`

Stellaris LM3S618		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s618-ram-hosted.ld`
	RAM Unhosted	`lm3s618-ram.ld`
	ROM Hosted	`lm3s618-rom-hosted.ld`
	ROM Unhosted	`lm3s618-rom.ld`

Stellaris LM3S628		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (32K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s628-ram-hosted.ld`
	RAM Unhosted	`lm3s628-ram.ld`
	ROM Hosted	`lm3s628-rom-hosted.ld`
	ROM Unhosted	`lm3s628-rom.ld`

Stellaris LM3S6420		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6420-ram-hosted.ld`
	RAM Unhosted	`lm3s6420-ram.ld`
	ROM Hosted	`lm3s6420-rom-hosted.ld`
	ROM Unhosted	`lm3s6420-rom.ld`

Stellaris LM3S6422		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6422-ram-hosted.ld`
	RAM Unhosted	`lm3s6422-ram.ld`
	ROM Hosted	`lm3s6422-rom-hosted.ld`
	ROM Unhosted	`lm3s6422-rom.ld`

Stellaris LM3S6432		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6432-ram-hosted.ld`
	RAM Unhosted	`lm3s6432-ram.ld`
	ROM Hosted	`lm3s6432-rom-hosted.ld`
	ROM Unhosted	`lm3s6432-rom.ld`

Stellaris LM3S6537		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6537-ram-hosted.ld`
	RAM Unhosted	`lm3s6537-ram.ld`
	ROM Hosted	`lm3s6537-rom-hosted.ld`
	ROM Unhosted	`lm3s6537-rom.ld`

Stellaris LM3S6610		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6610-ram-hosted.ld`
	RAM Unhosted	`lm3s6610-ram.ld`
	ROM Hosted	`lm3s6610-rom-hosted.ld`
	ROM Unhosted	`lm3s6610-rom.ld`

Stellaris LM3S6611		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6611-ram-hosted.ld`
	RAM Unhosted	`lm3s6611-ram.ld`
	ROM Hosted	`lm3s6611-rom-hosted.ld`
	ROM Unhosted	`lm3s6611-rom.ld`

Stellaris LM3S6618		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6618-ram-hosted.ld`
	RAM Unhosted	`lm3s6618-ram.ld`
	ROM Hosted	`lm3s6618-rom-hosted.ld`
	ROM Unhosted	`lm3s6618-rom.ld`

Stellaris LM3S6633		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6633-ram-hosted.ld`
	RAM Unhosted	`lm3s6633-ram.ld`
	ROM Hosted	`lm3s6633-rom-hosted.ld`
	ROM Unhosted	`lm3s6633-rom.ld`

Stellaris LM3S6637		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6637-ram-hosted.ld`
	RAM Unhosted	`lm3s6637-ram.ld`
	ROM Hosted	`lm3s6637-rom-hosted.ld`
	ROM Unhosted	`lm3s6637-rom.ld`

Stellaris LM3S6730		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6730-ram-hosted.ld`
	RAM Unhosted	`lm3s6730-ram.ld`
	ROM Hosted	`lm3s6730-rom-hosted.ld`
	ROM Unhosted	`lm3s6730-rom.ld`

Stellaris LM3S6753		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6753-ram-hosted.ld`
	RAM Unhosted	`lm3s6753-ram.ld`
	ROM Hosted	`lm3s6753-rom-hosted.ld`
	ROM Unhosted	`lm3s6753-rom.ld`

Stellaris LM3S6911		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6911-ram-hosted.ld`
	RAM Unhosted	`lm3s6911-ram.ld`
	ROM Hosted	`lm3s6911-rom-hosted.ld`
	ROM Unhosted	`lm3s6911-rom.ld`

Stellaris LM3S6918		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6918-ram-hosted.ld`
	RAM Unhosted	`lm3s6918-ram.ld`
	ROM Hosted	`lm3s6918-rom-hosted.ld`
	ROM Unhosted	`lm3s6918-rom.ld`

Stellaris LM3S6938		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6938-ram-hosted.ld`
	RAM Unhosted	`lm3s6938-ram.ld`
	ROM Hosted	`lm3s6938-rom-hosted.ld`
	ROM Unhosted	`lm3s6938-rom.ld`

Stellaris LM3S6950		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6950-ram-hosted.ld`
	RAM Unhosted	`lm3s6950-ram.ld`
	ROM Hosted	`lm3s6950-rom-hosted.ld`
	ROM Unhosted	`lm3s6950-rom.ld`

Stellaris LM3S6952		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6952-ram-hosted.ld`
	RAM Unhosted	`lm3s6952-ram.ld`
	ROM Hosted	`lm3s6952-rom-hosted.ld`
	ROM Unhosted	`lm3s6952-rom.ld`

Stellaris LM3S6965		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s6965-ram-hosted.ld`
	RAM Unhosted	`lm3s6965-ram.ld`
	ROM Hosted	`lm3s6965-rom-hosted.ld`
	ROM Unhosted	`lm3s6965-rom.ld`

Stellaris LM3S800		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s800-ram-hosted.ld`
	RAM Unhosted	`lm3s800-ram.ld`
	ROM Hosted	`lm3s800-rom-hosted.ld`
	ROM Unhosted	`lm3s800-rom.ld`

Stellaris LM3S801		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s801-ram-hosted.ld`
	RAM Unhosted	`lm3s801-ram.ld`
	ROM Hosted	`lm3s801-rom-hosted.ld`
	ROM Unhosted	`lm3s801-rom.ld`

Stellaris LM3S808		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s808-ram-hosted.ld`
	RAM Unhosted	`lm3s808-ram.ld`
	ROM Hosted	`lm3s808-rom-hosted.ld`
	ROM Unhosted	`lm3s808-rom.ld`

Stellaris LM3S811		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s811-ram-hosted.ld`
	RAM Unhosted	`lm3s811-ram.ld`
	ROM Hosted	`lm3s811-rom-hosted.ld`
	ROM Unhosted	`lm3s811-rom.ld`

Stellaris LM3S812		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s812-ram-hosted.ld`
	RAM Unhosted	`lm3s812-ram.ld`
	ROM Hosted	`lm3s812-rom-hosted.ld`
	ROM Unhosted	`lm3s812-rom.ld`

Stellaris LM3S815		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s815-ram-hosted.ld`
	RAM Unhosted	`lm3s815-ram.ld`
	ROM Hosted	`lm3s815-rom-hosted.ld`
	ROM Unhosted	`lm3s815-rom.ld`

Stellaris LM3S817		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s817-ram-hosted.ld`
	RAM Unhosted	`lm3s817-ram.ld`
	ROM Hosted	`lm3s817-rom-hosted.ld`
	ROM Unhosted	`lm3s817-rom.ld`

Stellaris LM3S818		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s818-ram-hosted.ld`
	RAM Unhosted	`lm3s818-ram.ld`
	ROM Hosted	`lm3s818-rom-hosted.ld`
	ROM Unhosted	`lm3s818-rom.ld`

Stellaris LM3S828		
Processor name:	Stellaris Sandstorm	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (8K RAM), `rom` (64K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_sandstorm`	
Linker scripts:	RAM Hosted	`lm3s828-ram-hosted.ld`
	RAM Unhosted	`lm3s828-ram.ld`
	ROM Hosted	`lm3s828-rom-hosted.ld`
	ROM Unhosted	`lm3s828-rom.ld`

Stellaris LM3S8530		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8530-ram-hosted.ld`
	RAM Unhosted	`lm3s8530-ram.ld`
	ROM Hosted	`lm3s8530-rom-hosted.ld`
	ROM Unhosted	`lm3s8530-rom.ld`

Stellaris LM3S8538		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (96K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8538-ram-hosted.ld`
	RAM Unhosted	`lm3s8538-ram.ld`
	ROM Hosted	`lm3s8538-rom-hosted.ld`
	ROM Unhosted	`lm3s8538-rom.ld`

Stellaris LM3S8630		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (32K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8630-ram-hosted.ld`
	RAM Unhosted	`lm3s8630-ram.ld`
	ROM Hosted	`lm3s8630-rom-hosted.ld`
	ROM Unhosted	`lm3s8630-rom.ld`

Stellaris LM3S8730		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8730-ram-hosted.ld`
	RAM Unhosted	`lm3s8730-ram.ld`
	ROM Hosted	`lm3s8730-rom-hosted.ld`
	ROM Unhosted	`lm3s8730-rom.ld`

Stellaris LM3S8733		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8733-ram-hosted.ld`
	RAM Unhosted	`lm3s8733-ram.ld`
	ROM Hosted	`lm3s8733-rom-hosted.ld`
	ROM Unhosted	`lm3s8733-rom.ld`

Stellaris LM3S8738		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8738-ram-hosted.ld`
	RAM Unhosted	`lm3s8738-ram.ld`
	ROM Hosted	`lm3s8738-rom-hosted.ld`
	ROM Unhosted	`lm3s8738-rom.ld`

Stellaris LM3S8930		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8930-ram-hosted.ld`
	RAM Unhosted	`lm3s8930-ram.ld`
	ROM Hosted	`lm3s8930-rom-hosted.ld`
	ROM Unhosted	`lm3s8930-rom.ld`

Stellaris LM3S8933		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8933-ram-hosted.ld`
	RAM Unhosted	`lm3s8933-ram.ld`
	ROM Hosted	`lm3s8933-rom-hosted.ld`
	ROM Unhosted	`lm3s8933-rom.ld`

Stellaris LM3S8938		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8938-ram-hosted.ld`
	RAM Unhosted	`lm3s8938-ram.ld`
	ROM Hosted	`lm3s8938-rom-hosted.ld`
	ROM Unhosted	`lm3s8938-rom.ld`

Stellaris LM3S8962		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8962-ram-hosted.ld`
	RAM Unhosted	`lm3s8962-ram.ld`
	ROM Hosted	`lm3s8962-rom-hosted.ld`
	ROM Unhosted	`lm3s8962-rom.ld`

Stellaris LM3S8970		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8970-ram-hosted.ld`
	RAM Unhosted	`lm3s8970-ram.ld`
	ROM Hosted	`lm3s8970-rom-hosted.ld`
	ROM Unhosted	`lm3s8970-rom.ld`

Stellaris LM3S8971		
Processor name:	Stellaris Fury	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_fury`	
Linker scripts:	RAM Hosted	`lm3s8971-ram-hosted.ld`
	RAM Unhosted	`lm3s8971-ram.ld`
	ROM Hosted	`lm3s8971-rom-hosted.ld`
	ROM Unhosted	`lm3s8971-rom.ld`

Stellaris LM3S9781		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s9781-ram-hosted.ld`
	RAM Unhosted	`lm3s9781-ram.ld`
	ROM Hosted	`lm3s9781-rom-hosted.ld`
	ROM Unhosted	`lm3s9781-rom.ld`

Stellaris LM3S9790		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s9790-ram-hosted.ld`
	RAM Unhosted	`lm3s9790-ram.ld`
	ROM Hosted	`lm3s9790-rom-hosted.ld`
	ROM Unhosted	`lm3s9790-rom.ld`

Stellaris LM3S9792		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s9792-ram-hosted.ld`
	RAM Unhosted	`lm3s9792-ram.ld`
	ROM Hosted	`lm3s9792-rom-hosted.ld`
	ROM Unhosted	`lm3s9792-rom.ld`

Stellaris LM3S9997		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (64K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s9997-ram-hosted.ld`
	RAM Unhosted	`lm3s9997-ram.ld`
	ROM Hosted	`lm3s9997-rom-hosted.ld`
	ROM Unhosted	`lm3s9997-rom.ld`

Stellaris LM3S9B81		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (96K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s9b81-ram-hosted.ld`
	RAM Unhosted	`lm3s9b81-ram.ld`
	ROM Hosted	`lm3s9b81-rom-hosted.ld`
	ROM Unhosted	`lm3s9b81-rom.ld`

Stellaris LM3S9B90		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (96K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s9b90-ram-hosted.ld`
	RAM Unhosted	`lm3s9b90-ram.ld`
	ROM Hosted	`lm3s9b90-rom-hosted.ld`
	ROM Unhosted	`lm3s9b90-rom.ld`

Stellaris LM3S9B92		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (96K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s9b92-ram-hosted.ld`
	RAM Unhosted	`lm3s9b92-ram.ld`
	ROM Hosted	`lm3s9b92-rom-hosted.ld`
	ROM Unhosted	`lm3s9b92-rom.ld`

Stellaris LM3S9B95		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (96K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s9b95-ram-hosted.ld`
	RAM Unhosted	`lm3s9b95-ram.ld`
	ROM Hosted	`lm3s9b95-rom-hosted.ld`
	ROM Unhosted	`lm3s9b95-rom.ld`

Stellaris LM3S9B96		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (96K RAM), `rom` (256K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s9b96-ram-hosted.ld`
	RAM Unhosted	`lm3s9b96-ram.ld`
	ROM Hosted	`lm3s9b96-rom-hosted.ld`
	ROM Unhosted	`lm3s9b96-rom.ld`

Stellaris LM3S9L97		
Processor name:	Stellaris Tempest	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (48K RAM), `rom` (128K Flash ROM), `boot` (Boot ROM)	
Interrupt vector:	`__cs3_interrupt_vector_stellaris_tempest`	
Linker scripts:	RAM Hosted	`lm3s9l97-ram-hosted.ld`
	RAM Unhosted	`lm3s9l97-ram.ld`
	ROM Hosted	`lm3s9l97-rom-hosted.ld`
	ROM Unhosted	`lm3s9l97-rom.ld`

STMicroelectronics STM3210B-EVAL		
Processor name:	STM32F103RB	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (Internal SRAM), `rom` (Internal Flash), `option_bytes_rom` (Option Bytes)	
Interrupt vector:	`__cs3_interrupt_vector_stm32f10`	
Linker scripts:	RAM Hosted	`stm3210b-eval-ram-hosted.ld`
	RAM Unhosted	`stm3210b-eval-ram.ld`
	ROM Hosted	`stm3210b-eval-rom-hosted.ld`
	ROM Unhosted	`stm3210b-eval-rom.ld`

STMicroelectronics STM3210C-EVAL		
Processor name:	STM32F107VC	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (Internal SRAM), `rom` (Internal Flash), `option_bytes_rom` (Option Bytes)	
Interrupt vector:	`__cs3_interrupt_vector_stm32f10cl`	
Linker scripts:	RAM Hosted	`stm3210c-eval-ram-hosted.ld`
	RAM Unhosted	`stm3210c-eval-ram.ld`
	ROM Hosted	`stm3210c-eval-rom-hosted.ld`
	ROM Unhosted	`stm3210c-eval-rom.ld`

STMicroelectronics STM3210E-EVAL		
Processor name:	STM32F103ZE	
Processor options:	`-mcpu=cortex-m3 -mthumb`	
Memory regions:	`ram` (Internal SRAM), `rom` (Internal Flash), `option_bytes_rom` (Option Bytes), `extnor` (External NOR Flash), `extsram` (External SRAM), `extnand` (External NAND Flash)	
Interrupt vector:	`__cs3_interrupt_vector_stm32f10`	
Linker scripts:	RAM Hosted	`stm3210e-eval-ram-hosted.ld`
	RAM Unhosted	`stm3210e-eval-ram.ld`
	ROM Hosted	`stm3210e-eval-rom-hosted.ld`
	ROM Unhosted	`stm3210e-eval-rom.ld`

Xilinx Cortex-A9		
Processor name:	Cortex-A9	
Processor options:	`-mcpu=cortex-a9`	
Memory regions:	`ram` (256MB DDR SDRAM), `rom` (64MB NOR Flash Memory)	
Interrupt vector:	`__cs3_interrupt_vector_arm`	
Linker scripts:	RAM Hosted	`xilinxa9-ram-hosted.ld`
	RAM Unhosted	`xilinxa9-ram.ld`
	ROM Hosted	`xilinxa9-rom-hosted.ld`
	ROM Unhosted	`xilinxa9-rom.ld`

6.6. Interrupt Vector Tables

6.6.1. `__cs3_interrupt_vector_arm`

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

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

6.6.2. `__cs3_interrupt_vector_efm32g`

The efm32g interrupt vector table (`__cs3_interrupt_vector_efm32g`) contents are:

Number	**Name**	**Meaning**
0	`__cs3_stack`	Initial stack pointer
1	`__cs3_reset`	Reset entry point
2	`__cs3_isr_nmi`	Non Maskable Interrupt
3	`__cs3_isr_hard_fault`	Hardware fault
4	`__cs3_isr_mpu_fault`	MPU fault
5	`__cs3_isr_bus_fault`	Bus fault
6	`__cs3_isr_usage_fault`	Usage fault
7..10	`__cs3_isr_reserved_7..10`	Reserved for future use
11	`__cs3_isr_svcall`	System Vector Call
12	`__cs3_isr_debug`	Debug interrupt
13	`__cs3_isr_reserved_13`	Reserved for future use

Number	Name	Meaning
14	`__cs3_isr_pendsv`	
15	`__cs3_isr_systick`	System Ticker
16	`__cs3_isr_dma`	DMA interrupt
17	`__cs3_isr_GPIO_EVEN`	GPIO_EVEN interrupt
18	`__cs3_isr_TIMER0`	TIMER0 interrupt
19	`__cs3_isr_USART0_RX`	USART0_RX interrupt
20	`__cs3_isr_USART0_TX`	USART0_TX interrupt
21	`__cs3_isr_ACMP0`	ACMP0/ACMP1 interrupt
22	`__cs3_isr_ADC0`	ADC0 interrupt
23	`__cs3_isr_DAC0`	DAC0 interrupt
24	`__cs3_isr_I2C0`	I2C0 interrupt
25	`__cs3_isr_GPIO_ODD`	GPIO_ODD interrupt
26..27	`__cs3_isr_TIMER1..TIMER2`	TIMER1 interrupt
28	`__cs3_isr_USART1_RX`	USART1_RX interrupt
29	`__cs3_isr_USART1_TX`	USART1_TX interrupt
30	`__cs3_isr_USART2_RX`	USART2_RX interrupt
31	`__cs3_isr_USART2_TX`	USART2_TX interrupt
32	`__cs3_isr_UART0_RX`	UART0_RX interrupt
33	`__cs3_isr_UART0_TX`	UART0_TX interrupt
34..35	`__cs3_isr_LEUART0..LEUART1`	LEUART0 interrupt
36	`__cs3_isr_LETIMER0`	LETIMER0 interrupt
37..39	`__cs3_isr_PCNT0..PCNT2`	PCNT0 interrupt
40	`__cs3_isr_RTC`	RTC interrupt
41	`__cs3_isr_CMU`	CMU interrupt
42	`__cs3_isr_VCMP`	VCMP interrupt
43	`__cs3_isr_LCD`	LCD interrupt
44	`__cs3_isr_MSC`	MSC interrupt
45	`__cs3_isr_AES`	AES interrupt

6.6.3. `__cs3_interrupt_vector_kinetis`

The kinetis interrupt vector table (`__cs3_interrupt_vector_kinetis`) contents are:

Number	Name	Meaning
0	`__cs3_stack`	Initial stack pointer
1	`__cs3_reset`	Reset entry point
2	`__cs3_isr_nmi`	Non Maskable Interrupt
3	`__cs3_isr_hard_fault`	Hardware fault
4	`__cs3_isr_mpu_fault`	MPU fault
5	`__cs3_isr_bus_fault`	Bus fault

Number	Name	Meaning
6	`__cs3_isr_usage_fault`	Usage fault
7..10	`__cs3_isr_reserved_7..10`	Reserved for future use
11	`__cs3_isr_svcall`	System Vector Call
12	`__cs3_isr_debug`	Debug interrupt
13	`__cs3_isr_reserved_13`	Reserved for future use
14	`__cs3_isr_pendsv`	
15	`__cs3_isr_systick`	System Ticker
16..31	`__cs3_isr_dma0..dma15`	DMA Channel 0 transfer complete
32	`__cs3_isr_dma_error`	DMA Error Interrupt
33	`__cs3_isr_mcm`	MCM
34	`__cs3_isr_flash_command_complete`	Flash Command Complete
35	`__cs3_isr_flash_read_collision`	Flash Read Collision
36	`__cs3_isr_mode_controller`	Mode Controller
37	`__cs3_isr_llwu`	Low Leakage Wakeup
38	`__cs3_isr_wdog`	WDOG
39	`__cs3_isr_rngb`	Random Number Generator
40..41	`__cs3_isr_i2c0..i2c1`	I2C0
42..44	`__cs3_isr_spi0..spi2`	SPI0
45	`__cs3_isr_can0_ored_message_buffer`	CAN0 ORed Message Buffer
46	`__cs3_isr_can0_buf_off`	CAN0 Bus Off
47	`__cs3_isr_can0_error`	CAN0 Error
48	`__cs3_isr_can0_transmit_warning`	CAN0 Transmit Warning
49	`__cs3_isr_can0_receive_warning`	CAN0 Receive Warning
50	`__cs3_isr_can0_wake_up`	CAN0 Wake Up
51	`__cs3_isr_can0_imeu`	CAN0 IMEU
52	`__cs3_isr_can0_lost_receive`	CAN0 Lost Receive
53	`__cs3_isr_can1_ored_message_buffer`	CAN1 ORed Message Buffer
54	`__cs3_isr_can1_buf_off`	CAN1 Bus Off
55	`__cs3_isr_can1_error`	CAN1 Error
56	`__cs3_isr_can1_transmit_warning`	CAN1 Transmit Warning
57	`__cs3_isr_can1_receive_warning`	CAN1 Receive Warning
58	`__cs3_isr_can1_wake_up`	CAN1 Wake Up
59	`__cs3_isr_can1_imeu`	CAN1 IMEU
60	`__cs3_isr_can1_lost_receive`	CAN1 Lost Receive
61	`__cs3_isr_uart0_status_sources`	UART0 Status Sources
62	`__cs3_isr_uart0_error_sources`	UART0 Error Sources
63	`__cs3_isr_uart1_status_sources`	UART1 Status Sources
64	`__cs3_isr_uart1_error_sources`	UART1 Error Sources

Number	Name	Meaning
65	`__cs3_isr_uart2_status_sources`	UART2 Status Sources
66	`__cs3_isr_uart2_error_sources`	UART2 Error Sources
67	`__cs3_isr_uart3_status_sources`	UART3 Status Sources
68	`__cs3_isr_uart3_error_sources`	UART3 Error Sources
69	`__cs3_isr_uart4_status_sources`	UART4 Status Sources
70	`__cs3_isr_uart4_error_sources`	UART4 Error Sources
71	`__cs3_isr_uart5_status_sources`	UART5 Status Sources
72	`__cs3_isr_uart5_error_sources`	UART5 Error Sources
73..74	`__cs3_isr_adc0..adc1`	ADC0
75..77	`__cs3_isr_cmp0..cmp2`	CMP0
78..80	`__cs3_isr_ftm0..ftm2`	FTM0
81	`__cs3_isr_cmt`	CMT
82	`__cs3_isr_rtc`	RTC
83	`__cs3_isr_reserved_83`	Reserved for future use
84..87	`__cs3_isr_pit0..pit3`	PIT Channel 0
88	`__cs3_isr_pdb`	PDB
89	`__cs3_isr_usb_otg`	USB OTG
90	`__cs3_isr_usb_charger_detect`	USB Charger Detect
91	`__cs3_isr_ethernet_mac_timer`	Ethernet MAC Timer
92	`__cs3_isr_ethernet_mac_transmit`	Ethernet MAC Transmit
93	`__cs3_isr_ethernet_mac_receive`	Ethernet MAC Receive
94	`__cs3_isr_ethernet_mac_error`	Ethernet MAC Error
95	`__cs3_isr_i2s`	I2S
96	`__cs3_isr_sdhc`	SDHC
97..98	`__cs3_isr_dac0..dac1`	DAC0
99	`__cs3_isr_tsi`	TSI
100	`__cs3_isr_mcg`	MCG
101	`__cs3_isr_low_power_timer`	Low Power Timer
102	`__cs3_isr_reserved_102`	Reserved for future use
103	`__cs3_isr_port_control_module_a`	Port Control Module a
104	`__cs3_isr_port_control_module_b`	Port Control Module b
105	`__cs3_isr_port_control_module_c`	Port Control Module c
106	`__cs3_isr_port_control_module_d`	Port Control Module d
107	`__cs3_isr_port_control_module_e`	Port Control Module e

6.6.4. `__cs3_interrupt_vector_lpc17xx`

The NXP LPC17xx interrupt vector table (`__cs3_interrupt_vector_lpc17xx`) contents are:

Number	Name	Meaning
0	`__cs3_stack`	Initial stack pointer
1	`__cs3_reset`	Reset entry point
2	`__cs3_isr_nmi`	Non Maskable Interrupt
3	`__cs3_isr_hard_fault`	Hardware fault
4	`__cs3_isr_mpu_fault`	MPU fault
5	`__cs3_isr_bus_fault`	Bus fault
6	`__cs3_isr_usage_fault`	Usage fault
7	`__cs3_lpc17xx_checksum`	Interrupt vector checksum
8..10	`__cs3_isr_reserved_8..10`	Reserved for future use
11	`__cs3_isr_svcall`	System Vector Call
12	`__cs3_isr_debug`	Debug interrupt
13	`__cs3_isr_reserved_13`	Reserved for future use
14	`__cs3_isr_pendsv`	
15	`__cs3_isr_systick`	System Ticker
16	`__cs3_isr_watchdog`	Watchdog interrupt
17..20	`__cs3_isr_timer0..timer3`	Timer 0 interrupt
21..24	`__cs3_isr_uart0..uart3`	UART0 interrupt
25	`__cs3_isr_pwm1`	PWM1 interrupt
26..28	`__cs3_isr_i2c0..i2c2`	I2C0 interrupt
29	`__cs3_isr_spi`	SPI interrupt
30..31	`__cs3_isr_ssp0..ssp1`	SSP0 interrupt
32	`__cs3_isr_pll0`	PLL0 interrupt
33	`__cs3_isr_rtc`	RTC interrupt
34..37	`__cs3_isr_external0..external3`	External interrupt 0
38	`__cs3_isr_adc`	ADC interrupt
39	`__cs3_isr_bod`	Brown out detect interrupt
40	`__cs3_isr_usb`	USB interrupt
41	`__cs3_isr_can`	CAN interrupt
42	`__cs3_isr_gpdma`	GPDMA interrupt
43	`__cs3_isr_i2s`	I2S interrupt
44	`__cs3_isr_ethernet`	Ethernet interrupt
45	`__cs3_isr_ritint`	RIT interrupt
46	`__cs3_isr_motor_control_pwm`	Motor control PWM interrupt
47	`__cs3_isr_quadrature_encoder`	Quadrature encoder interrupt
48	`__cs3_isr_pll1`	PLL1
49	`__cs3_isr_usb_activity`	USB activity interrupt
50	`__cs3_isr_can_activity`	CAN activity interrupt

6.6.5. `__cs3_interrupt_vector_lpc21xx`

The NXP LPC21xx interrupt vector table (`__cs3_interrupt_vector_lpc21xx`) contents are:

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

6.6.6. `__cs3_interrupt_vector_micro`

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

Number	Name	Meaning
0	`__cs3_stack`	Initial stack pointer
1	`__cs3_reset`	Reset entry point
2	`__cs3_isr_nmi`	Non Maskable Interrupt
3	`__cs3_isr_hard_fault`	Hardware fault
4	`__cs3_isr_mpu_fault`	MPU fault
5	`__cs3_isr_bus_fault`	Bus fault
6	`__cs3_isr_usage_fault`	Usage fault
7..10	`__cs3_isr_reserved_7..10`	Reserved for future use
11	`__cs3_isr_svcall`	System Vector Call
12	`__cs3_isr_debug`	Debug interrupt
13	`__cs3_isr_reserved_13`	Reserved for future use
14	`__cs3_isr_pendsv`	
15	`__cs3_isr_systick`	System Ticker
16..47	`__cs3_isr_external_0..31`	External interrupt

6.6.7. `__cs3_interrupt_vector_stellaris`

The Stellaris DustDevil interrupt vector table (`__cs3_interrupt_vector_stellaris`) contents are:

Number	Name	Meaning
0	`__cs3_stack`	Initial stack pointer
1	`__cs3_reset`	Reset entry point

Number	Name	Meaning
2	`__cs3_isr_nmi`	Non Maskable Interrupt
3	`__cs3_isr_hard_fault`	Hardware fault
4	`__cs3_isr_mpu_fault`	MPU fault
5	`__cs3_isr_bus_fault`	Bus fault
6	`__cs3_isr_usage_fault`	Usage fault
7..10	`__cs3_isr_reserved_7..10`	Reserved for future use
11	`__cs3_isr_svcall`	System Vector Call
12	`__cs3_isr_debug`	Debug interrupt
13	`__cs3_isr_reserved_13`	Reserved for future use
14	`__cs3_isr_pendsv`	
15	`__cs3_isr_systick`	System Ticker
16	`__cs3_isr_gpio_a`	General Purpose IO
17	`__cs3_isr_gpio_b`	General Purpose IO
18	`__cs3_isr_gpio_c`	General Purpose IO
19	`__cs3_isr_gpio_d`	General Purpose IO
20	`__cs3_isr_gpio_e`	General Purpose IO
21..22	`__cs3_isr_uart0..uart1`	UART
23	`__cs3_isr_ssi0`	SSI
24	`__cs3_isr_i2c0`	I2C
25	`__cs3_isr_pwm_fault`	Pulse Width Modulation
26..28	`__cs3_isr_pwm0..pwm2`	Pulse Width Modulation
29	`__cs3_isr_qei0`	QEI
30..33	`__cs3_isr_adc0..adc3`	Analog to Digital
34	`__cs3_isr_watchdog`	Watchdog Timeout
35	`__cs3_isr_timer0a`	Timer
36	`__cs3_isr_timer0b`	Timer
37	`__cs3_isr_timer1a`	Timer
38	`__cs3_isr_timer1b`	Timer
39	`__cs3_isr_timer2a`	Timer
40	`__cs3_isr_timer2b`	Timer
41..43	`__cs3_isr_comp0..comp2`	Comparator
44	`__cs3_isr_sysctl`	System Control
45	`__cs3_isr_flashctl`	Flash Control
46	`__cs3_isr_gpio_f`	General Purpose IO
47	`__cs3_isr_gpio_g`	General Purpose IO
48	`__cs3_isr_gpio_h`	General Purpose IO
49	`__cs3_isr_uart2`	UART
50	`__cs3_isr_ssi1`	SSI

Number	Name	Meaning
51	`__cs3_isr_timer3a`	Timer
52	`__cs3_isr_timer3b`	Timer
53	`__cs3_isr_i2c1`	I2C
54	`__cs3_isr_qei1`	QEI
55..57	`__cs3_isr_can0..can2`	CAN
58	`__cs3_isr_ethernet0`	Ethernet
59	`__cs3_isr_hibernate`	Hibernate
60	`__cs3_isr_usb0`	USB Controller
61	`__cs3_isr_pwm3`	Pulse Width Modulation
62	`__cs3_isr_udma`	uDMA Controller
63	`__cs3_isr_udmaerr`	uDMA Error

6.6.8. `__cs3_interrupt_vector_stellaris_fury`

The Stellaris Fury interrupt vector table (`__cs3_interrupt_vector_stellaris_fury`) contents are:

Number	Name	Meaning
0	`__cs3_stack`	Initial stack pointer
1	`__cs3_reset`	Reset entry point
2	`__cs3_isr_nmi`	Non Maskable Interrupt
3	`__cs3_isr_hard_fault`	Hardware fault
4	`__cs3_isr_mpu_fault`	MPU fault
5	`__cs3_isr_bus_fault`	Bus fault
6	`__cs3_isr_usage_fault`	Usage fault
7..10	`__cs3_isr_reserved_7..10`	Reserved for future use
11	`__cs3_isr_svcall`	System Vector Call
12	`__cs3_isr_debug`	Debug interrupt
13	`__cs3_isr_reserved_13`	Reserved for future use
14	`__cs3_isr_pendsv`	
15	`__cs3_isr_systick`	System Ticker
16	`__cs3_isr_gpio_a`	General Purpose IO
17	`__cs3_isr_gpio_b`	General Purpose IO
18	`__cs3_isr_gpio_c`	General Purpose IO
19	`__cs3_isr_gpio_d`	General Purpose IO
20	`__cs3_isr_gpio_e`	General Purpose IO
21..22	`__cs3_isr_uart0..uart1`	UART
23	`__cs3_isr_ssi0`	SSI
24	`__cs3_isr_i2c0`	I2C
25	`__cs3_isr_pwm_fault`	Pulse Width Modulation

Number	Name	Meaning
26..28	`__cs3_isr_pwm0..pwm2`	Pulse Width Modulation
29	`__cs3_isr_qei0`	QEI
30..33	`__cs3_isr_adc0..adc3`	Analog to Digital
34	`__cs3_isr_watchdog`	Watchdog Timeout
35	`__cs3_isr_timer0a`	Timer
36	`__cs3_isr_timer0b`	Timer
37	`__cs3_isr_timer1a`	Timer
38	`__cs3_isr_timer1b`	Timer
39	`__cs3_isr_timer2a`	Timer
40	`__cs3_isr_timer2b`	Timer
41..43	`__cs3_isr_comp0..comp2`	Comparator
44	`__cs3_isr_sysctl`	System Control
45	`__cs3_isr_flashctl`	Flash Control
46	`__cs3_isr_gpio_f`	General Purpose IO
47	`__cs3_isr_gpio_g`	General Purpose IO
48	`__cs3_isr_gpio_h`	General Purpose IO
49	`__cs3_isr_uart2`	UART
50	`__cs3_isr_ssi1`	SSI
51	`__cs3_isr_timer3a`	Timer
52	`__cs3_isr_timer3b`	Timer
53	`__cs3_isr_i2c1`	I2C
54	`__cs3_isr_qei1`	QEI
55..57	`__cs3_isr_can0..can2`	CAN
58	`__cs3_isr_ethernet0`	Ethernet
59	`__cs3_isr_hibernate`	Hibernate

6.6.9. `__cs3_interrupt_vector_stellaris_sandstorm`

The Stellaris Sandstorm interrupt vector table (`__cs3_interrupt_vector_stellaris_sandstorm`) contents are:

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

Number	Name	Meaning
11	`__cs3_isr_svcall`	System Vector Call
12	`__cs3_isr_debug`	Debug interrupt
13	`__cs3_isr_reserved_13`	Reserved for future use
14	`__cs3_isr_pendsv`	
15	`__cs3_isr_systick`	System Ticker
16	`__cs3_isr_gpio_a`	General Purpose IO
17	`__cs3_isr_gpio_b`	General Purpose IO
18	`__cs3_isr_gpio_c`	General Purpose IO
19	`__cs3_isr_gpio_d`	General Purpose IO
20	`__cs3_isr_gpio_e`	General Purpose IO
21..22	`__cs3_isr_uart0..uart1`	UART
23	`__cs3_isr_ssi0`	SSI
24	`__cs3_isr_i2c0`	I2C
25	`__cs3_isr_pwm_fault`	Pulse Width Modulation
26..28	`__cs3_isr_pwm0..pwm2`	Pulse Width Modulation
29	`__cs3_isr_qei0`	QEI
30..33	`__cs3_isr_adc0..adc3`	Analog to Digital
34	`__cs3_isr_watchdog`	Watchdog Timeout
35	`__cs3_isr_timer0a`	Timer
36	`__cs3_isr_timer0b`	Timer
37	`__cs3_isr_timer1a`	Timer
38	`__cs3_isr_timer1b`	Timer
39	`__cs3_isr_timer2a`	Timer
40	`__cs3_isr_timer2b`	Timer
41..43	`__cs3_isr_comp0..comp2`	Comparator
44	`__cs3_isr_sysctl`	System Control
45	`__cs3_isr_flashctl`	Flash Control

6.6.10. `__cs3_interrupt_vector_stellaris_tempest`

The Stellaris Tempest interrupt vector table (`__cs3_interrupt_vector_stellaris_tempest`) contents are:

Number	Name	Meaning
0	`__cs3_stack`	Initial stack pointer
1	`__cs3_reset`	Reset entry point
2	`__cs3_isr_nmi`	Non Maskable Interrupt
3	`__cs3_isr_hard_fault`	Hardware fault
4	`__cs3_isr_mpu_fault`	MPU fault
5	`__cs3_isr_bus_fault`	Bus fault

Number	Name	Meaning
6	`__cs3_isr_usage_fault`	Usage fault
7..10	`__cs3_isr_reserved_7..10`	Reserved for future use
11	`__cs3_isr_svcall`	System Vector Call
12	`__cs3_isr_debug`	Debug interrupt
13	`__cs3_isr_reserved_13`	Reserved for future use
14	`__cs3_isr_pendsv`	
15	`__cs3_isr_systick`	System Ticker
16	`__cs3_isr_gpio_a`	General Purpose IO
17	`__cs3_isr_gpio_b`	General Purpose IO
18	`__cs3_isr_gpio_c`	General Purpose IO
19	`__cs3_isr_gpio_d`	General Purpose IO
20	`__cs3_isr_gpio_e`	General Purpose IO
21..22	`__cs3_isr_uart0..uart1`	UART
23	`__cs3_isr_ssi0`	SSI
24	`__cs3_isr_i2c0`	I2C
25	`__cs3_isr_pwm_fault`	Pulse Width Modulation
26..28	`__cs3_isr_pwm0..pwm2`	Pulse Width Modulation
29	`__cs3_isr_qei0`	QEI
30..33	`__cs3_isr_adc0..adc3`	Analog to Digital
34	`__cs3_isr_watchdog`	Watchdog Timeout
35	`__cs3_isr_timer0a`	Timer
36	`__cs3_isr_timer0b`	Timer
37	`__cs3_isr_timer1a`	Timer
38	`__cs3_isr_timer1b`	Timer
39	`__cs3_isr_timer2a`	Timer
40	`__cs3_isr_timer2b`	Timer
41..43	`__cs3_isr_comp0..comp2`	Comparator
44	`__cs3_isr_sysctl`	System Control
45	`__cs3_isr_flashctl`	Flash Control
46	`__cs3_isr_gpio_f`	General Purpose IO
47	`__cs3_isr_gpio_g`	General Purpose IO
48	`__cs3_isr_gpio_h`	General Purpose IO
49	`__cs3_isr_uart2`	UART
50	`__cs3_isr_ssi1`	SSI
51	`__cs3_isr_timer3a`	Timer
52	`__cs3_isr_timer3b`	Timer
53	`__cs3_isr_i2c1`	I2C
54	`__cs3_isr_qei1`	QEI

Number	Name	Meaning
55..57	`__cs3_isr_can0..can2`	CAN
58	`__cs3_isr_ethernet0`	Ethernet
59	`__cs3_isr_hibernate`	Hibernate
60	`__cs3_isr_usb0`	USB Controller
61	`__cs3_isr_pwm3`	Pulse Width Modulation
62	`__cs3_isr_udma`	uDMA Controller
63	`__cs3_isr_udmaerr`	uDMA Error
64..67	`__cs3_isr_adc1ss0..adc1ss3`	Analog-to-Digital Sample Sequence
68	`__cs3_isr_i2s0`	Inter-Integrated Circuit Sound
69	`__cs3_isr_epi0`	External Peripheral Interface
70	`__cs3_isr_gpio_j`	General Purpose IO

6.6.11. `__cs3_interrupt_vector_stm32f10`

The STM32F10xxx interrupt vector table (`__cs3_interrupt_vector_stm32f10`) contents are:

Number	Name	Meaning
0	`__cs3_stack`	Initial stack pointer
1	`__cs3_reset`	Reset entry point
2	`__cs3_isr_nmi`	Non Maskable Interrupt
3	`__cs3_isr_hard_fault`	Hardware fault
4	`__cs3_isr_mpu_fault`	MPU fault
5	`__cs3_isr_bus_fault`	Bus fault
6	`__cs3_isr_usage_fault`	Usage fault
7..10	`__cs3_isr_reserved_7..10`	Reserved for future use
11	`__cs3_isr_svcall`	System Vector Call
12	`__cs3_isr_debug`	Debug interrupt
13	`__cs3_isr_reserved_13`	Reserved for future use
14	`__cs3_isr_pendsv`	
15	`__cs3_isr_systick`	System Ticker
16	`__cs3_isr_wwdg`	Window watchdog interrupt
17	`__cs3_isr_pvd`	PVD through EXTI Line detection interrupt
18	`__cs3_isr_tamper`	Tamper interrupt
19	`__cs3_isr_rtc`	RTC global interrupt
20	`__cs3_isr_flash`	Flash global interrupt
21	`__cs3_isr_rcc`	RCC global interrupt
22..26	`__cs3_isr_exti0..exti4`	EXTI Line0 interrupt
27..33	`__cs3_isr_dma1_channel1..channel7`	DMA1 Channel1 global interrupt

Number	Name	Meaning
34	`__cs3_isr_adc1_2`	ADC1 and ADC2 global interrupt
35	`__cs3_isr_usb_hp_can_tx`	USB High Priority or CAN TX interrupts
36	`__cs3_isr_usb_lp_can_rx0`	USB Low Priority or CAN RX0 interrupts
37	`__cs3_isr_can_rx1`	CAN RX1 interrupt
38	`__cs3_isr_can_sce`	CAN SCE interrupt
39	`__cs3_isr_exti9_5`	EXTI Line[9:5] interrupts
40	`__cs3_isr_tim1_brk`	TIM1 Break interrupt
41	`__cs3_isr_tim1_up`	TIM1 Update interrupt
42	`__cs3_isr_tim1_trg_com`	TIM1 Trigger and Commutation interrupts
43	`__cs3_isr_tim1_cc`	TIM1 Capture Compare interrupt
44..46	`__cs3_isr_tim2..tim4`	TIM2 global interrupt
47	`__cs3_isr_i2c1_ev`	I2C1 event interrupt
48	`__cs3_isr_i2c1_er`	I2C1 error interrupt
49	`__cs3_isr_i2c2_ev`	I2C2 event interrupt
50	`__cs3_isr_i2c2_er`	I2C2 error interrupt
51..52	`__cs3_isr_spi1..spi2`	SPI1 global interrupt
53..55	`__cs3_isr_usart1..usart3`	USART1 global interrupt
56	`__cs3_isr_exti15_10`	EXTI Line[15:10] interrupts
57	`__cs3_isr_rtcalarm`	RTC alarm through EXTI line interrupt
58	`__cs3_isr_usbwakeup`	USB wakeup from suspend through EXTI line interrupt
59	`__cs3_isr_tim8_brk`	TIM8 Break interrupt
60	`__cs3_isr_tim8_up`	TIM8 Update interrupt
61	`__cs3_isr_tim8_trg_com`	TIM8 Trigger and Commutation interrupts
62	`__cs3_isr_tim8_cc`	TIM8 Capture Compare interrupt
63	`__cs3_isr_adc3`	ADC3 global interrupt
64	`__cs3_isr_fsmc`	FSMC global interrupt
65	`__cs3_isr_sdio`	SDIO global interrupt
66	`__cs3_isr_tim5`	TIM5 global interrupt
67	`__cs3_isr_spi3`	SPI3 global interrupt
68..69	`__cs3_isr_uart4..uart5`	UART4 global interrupt
70..71	`__cs3_isr_tim6..tim7`	TIM6 global interrupt
72..74	`__cs3_isr_dma2_channel1..channel3`	DMA2 Channel1 global interrupt
75	`__cs3_isr_dma2_channel4_5`	DMA2 Channel4 and DMA2 Channel5 global interrupts

6.6.12. `__cs3_interrupt_vector_stm32f10cl`

The STM32F10xxx connectivity line interrupt vector table (`__cs3_interrupt_vector_stm32f10cl`) contents are:

Number	Name	Meaning
0	`__cs3_stack`	Initial stack pointer
1	`__cs3_reset`	Reset entry point
2	`__cs3_isr_nmi`	Non Maskable Interrupt
3	`__cs3_isr_hard_fault`	Hardware fault
4	`__cs3_isr_mpu_fault`	MPU fault
5	`__cs3_isr_bus_fault`	Bus fault
6	`__cs3_isr_usage_fault`	Usage fault
7..10	`__cs3_isr_reserved_7..10`	Reserved for future use
11	`__cs3_isr_svcall`	System Vector Call
12	`__cs3_isr_debug`	Debug interrupt
13	`__cs3_isr_reserved_13`	Reserved for future use
14	`__cs3_isr_pendsv`	
15	`__cs3_isr_systick`	System Ticker
16	`__cs3_isr_wwdg`	Window Watchdog interrupt
17	`__cs3_isr_pvd`	PVD through EXTI Line detection interrupt
18	`__cs3_isr_tamper`	Tamper interrupt
19	`__cs3_isr_rtc`	RTC global interrupt
20	`__cs3_isr_flash`	Flash global interrupt
21	`__cs3_isr_rcc`	RCC global interrupt
22..26	`__cs3_isr_exti0..exti4`	EXTI Line0 interrupt
27..33	`__cs3_isr_dma1_channel1..channel7`	DMA1 Channel1 global interrupt
34	`__cs3_isr_adc1_2`	ADC1 and ADC2 global interrupt
35	`__cs3_isr_can1_tx`	CAN1 TX interrupts
36..37	`__cs3_isr_can1_rx0..rx1`	CAN1 RX0 interrupts
38	`__cs3_isr_can1_sce`	CAN1 SCE interrupt
39	`__cs3_isr_exti9_5`	EXTI Line[9:5] interrupts
40	`__cs3_isr_tim1_brk`	TIM1 Break interrupt
41	`__cs3_isr_tim1_up`	TIM1 Update interrupt
42	`__cs3_isr_tim1_trg_com`	TIM1 Trigger and Commutation interrupts
43	`__cs3_isr_tim1_cc`	TIM1 Capture Compare interrupt
44..46	`__cs3_isr_tim2..tim4`	TIM2 global interrupt
47	`__cs3_isr_i2c1_ev`	I2C1 event interrupt
48	`__cs3_isr_i2c1_er`	I2C1 error interrupt
49	`__cs3_isr_i2c2_ev`	I2C2 event interrupt

Number	Name	Meaning
50	`__cs3_isr_i2c2_er`	I2C2 error interrupt
51..52	`__cs3_isr_spi1..spi2`	SPI1 global interrupt
53..55	`__cs3_isr_usart1..usart3`	USART1 global interrupt
56	`__cs3_isr_exti15_10`	EXTI Line[15:10] interrupts
57	`__cs3_isr_rtcalarm`	RTC alarm through EXTI line interrupt
58	`__cs3_isr_otg_fs_wkup`	USB On-The-Go FS Wakeup through EXTI line interrupt
59..65	`__cs3_isr_reserved_59..65`	Reserved for future use
66	`__cs3_isr_tim5`	TIM5 global interrupt
67	`__cs3_isr_spi3`	SPI3 global interrupt
68..69	`__cs3_isr_uart4..uart5`	UART4 global interrupt
70..71	`__cs3_isr_tim6..tim7`	TIM6 global interrupt
72..76	`__cs3_isr_dma2_channel1..channel5`	DMA2 Channel1 global interrupt
77	`__cs3_isr_eth`	Ethernet global interrupt
78	`__cs3_isr_eth_wkup`	Ethernet Wakeup through EXTI line interrupt
79	`__cs3_isr_can2_tx`	CAN2 TX interrupts
80..81	`__cs3_isr_can2_rx0..rx1`	CAN2 RX0 interrupts
82	`__cs3_isr_can2_sce`	CAN2 SCE interrupt
83	`__cs3_isr_otg_fs`	USB On The Go FS global interrupt

Chapter 7
Sourcery G++ Debug Sprite

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

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

The Sprite acts as an interface between GDB and external debug devices and libraries. Refer to Section 7.3, "Invoking Sourcery G++ Debug Sprite" for information about the specific devices supported by this version of Sourcery G++.

The Sourcery G++ Debug Sprite also supports programming of flash memory on the target. When used with an appropriate linker script and board configuration, flash programming is automatic when you load your program in the debugger.

Important

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

7.1. Probing for Debug Devices

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

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

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

```
CodeSourcery ARM Debug Sprite
    (Sourcery G++ 2011.02-2)
armusb: [speed=<n:0-7>] ARMUSB (Stellaris) device
  armusb:///0B01000C - Stellaris Evaluation Board (0B01000C)
rdi: (rdi-library=<file>&rdi-config=<file>) RDI Device
  rdi:/// - RDI Device
```

This shows that ARMUSB and RDI devices are supported. The exact set of supported devices depends on your host system and the version of Sourcery G++ you have installed; refer to Section 7.3, "Invoking Sourcery G++ Debug Sprite" for complete information.

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

7.2. Debug Sprite Example

If you are using the Sourcery G++ IDE, refer to Section 4.3, "Debugging Applications" for basic instructions on how to build and debug your program. This section explains how to use the Sourcery G++ Debug Sprite from the command line, or as a remote debug agent.

Start by compiling and linking a simple test program for your target board, following the instructions in Chapter 5, "Using Sourcery G++ from the Command Line". Use the `-g` option to tell the compiler to generate debugging information.

For example, use this command to build the `factorial` program to run from RAM on a Stellaris LM3S2965 board:

```
> arm-none-eabi-gcc -g -mcpu=cortex-m3 -mthumb \
    -Tlm3s2965-ram-hosted.ld main.c -o factorial
```

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

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

Next start the debugger on your host system:

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

The command for connecting GDB to the board depends on the debug device you are using; this is described in more detail in Section 7.3, “Invoking Sourcery G++ Debug Sprite”. If you are using an ARMUSB debug device to connect to a Stellaris LM3S2965 board, use:

```
(gdb) target remote | arm-none-eabi-sprite \
  armusb:///?speed=2 lm3s2965
```

If you are connecting via RDI, you must specify the full path to the RDI library file and configuration file for that library. Use quotes to escape the Sprite argument syntax from the shell. For example, use a command like this to connect to the ARMulator:

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

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

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

Next, use GDB to load your program onto the target board. If your target board includes a flash memory region and you have linked your program to reside in flash by providing an appropriate linker script, flash programming happens automatically when you issue the GDB `load` command.

```
(gdb) load
```

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

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

7.3. Invoking Sourcery G++ Debug Sprite

The Debug Sprite is invoked as follows:

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

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

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

Most device URL schemes also follow the regular format:

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

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

`armusb` Use an ARMUSB (Stellaris) debugging device. Refer to Section 7.5, "ARMUSB (Stellaris) Devices".

`rdi` Use an RDI debugging device. Refer to Section 7.6, "Remote Debug Interface Devices".

`flashpro` Use a FlashPro debugging device. Refer to Section 7.7, "Actel FlashPro Devices".

`ulink` Use a Keil ULINK2 debugging device. Refer to Section 7.8, "Keil ULINK2 Devices".

`altera` Use an Altera FPGA. Refer to Section 7.9, "Altera Devices".

`jlink` Use a SEGGER J-Link. Refer to Section 7.10, "SEGGER J-Link Devices".

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

The *board-file* specifies an XML file that describes how to initialize the target board, as well as other properties of the board used by the debugger. If *board-file* refers to a file (via a relative or absolute pathname), it is read. Otherwise, *board-file* can be a board name, and the toolchain's board directory is searched for a matching file. See Section 7.13, "Supported Board Files" for the list of supported boards, or invoke the Sprite with the `-b` option to list the available board files. If you are using the Sourcery G++ Board Builder to generate a board definition for your board, it produces an XML file for use as a Sprite *board-file*. You can also write a custom board file; see Section 7.14, "Board File Syntax" for more information about the file format.

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

7.4. Sourcery G++ Debug Sprite Options

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

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

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

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

`-l` *[host]:port* Specify the host address and port number to listen for a GDB connection. If this option is not given, the Debug Sprite communicates with GDB using stdin and stdout. If you start the Sprite from within GDB using the `target`

`remote | arm-none-eabi-sprite ...` command, you do not need this option.

`-m` Listen for multiple sequential connections. Normally the Debug Sprite terminates after the first connection from GDB terminates. This option instead makes it listen for a subsequent connection. To terminate the Sprite, open a connection and send the string `END\n`.

`-q` Do not print any messages.

`-v` Print additional messages.

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

7.5. ARMUSB (Stellaris) Devices

The Sourcery G++ Debug Sprite supports Stellaris devices equipped with an FTDI ARMUSB debug interface.

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

```
armusb:///
```

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

```
armusb:///[path][?device-options]
```

The *`path`* values are serial numbers reported by the connected boards. These numbers may not be unique; for example, some early-model Stellaris boards all report the same serial number. In this case, it is not possible to use the *`device-url`* to select among them.

The Debug Sprite can autodetect connected ARMUSB devices. Invoking the Sprite with the `-i` option, as described in Section 7.1, “Probing for Debug Devices”, displays the *`device-url`* for each detected device:

```
> arm-none-eabi-sprite -i
...
armusb: [speed=<n:0-7>] ARMUSB (Stellaris) device
  armusb:///0B01000C - Stellaris Evaluation Board (0B01000C)
  armusb:///051100E2 - Stellaris Evaluation Board (051100E2)
```

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

`speed=`*`speed`* Specify the speed of the connection, from `0` (fastest, default) to `7` (slowest). Depending on the CPU speed of the target board, lower values may lead to unreliable communication with the target. It is recommended to use slower speeds in that case.

7.5.1. ARMUSB Configuration and Drivers

This section explains how to set up your host computer so that the debugger can connect to the Stellaris board.

7.5.1.1. Configuration on Microsoft Windows Hosts

To communicate with your Stellaris board, you must install the Luminary Micro FTDI drivers.

If you installed Sourcery G++ from a CD distributed with your Stellaris board, the drivers are located in the directory `Tools\FTDI` on the CD. You can also download the latest Luminary Micro FTDI drivers from the Luminary Micro Software Updates[1] page on the internet.

To install the drivers, first unpack them on your local machine. Then, plug in your Stellaris board using the USB connector. Windows should detect the device automatically and start the Found New Hardware Wizard to guide you through the driver install. Tell the wizard to install the drivers from the folder where you unpacked them, rather than by searching the internet. You must go through the wizard twice to install the complete set of drivers.

7.5.1.2. Configuration on GNU/Linux Hosts

The Sourcery G++ Debug Sprite includes an open-source USB driver for GNU/Linux systems. You do not need to install a separate driver.

On some Linux systems, it may be necessary only to connect the USB cable from the target board to your computer in order to run the Sourcery G++ Debug Sprite. On other systems, you may need to perform either or both of the following actions to configure the USB device:

1. If the ftdi_sio kernel module is being loaded automatically when the device is connected, you must unload it again. As root, do `rmmod ftdi_sio`.

2. Make sure that you have privilege to access the USB device. Rather than running the Sourcery G++ Debug Sprite as root, the preferred solution on modern Linux distributions is to add a udev configuration file for the device. As root, create a file named `/etc/udev/rules.d/stellaris.rules` that contains the following line:

   ```
   SUBSYSTEM=="usb", ATTR{manufacturer}=="LMI", OWNER="name"
   ```

   where *name* is your normal login name. Then reconnect the device to cause udev to read the new configuration file.

   If you have multiple Stellaris USB devices attached to the same host that need to be managed with different permissions, consult the udev documentation supplied with your Linux distribution for information on how to match on the serial number, product identifier, or other fields.

7.5.2. Using a Stellaris Board to Debug Production Systems

The In-Circuit Debug Interface (ICDI) provided on most Stellaris evaluation kit boards allows them to be used in a pass-through mode to debug production systems containing Stellaris microcontrollers. You do not need to purchase a separate ICE unit or any additional software to debug a production system.

To use the Stellaris evaluation kit board in this way, you must provide a standard 20-pin JTAG header connected to the Stellaris microcontroller on your production system. Then, use the 20-pin JTAG cable provided with the evaluation kit to connect your production system to the evaluation kit board. Finally, connect the evaluation board to the USB port on your workstation, just as you would to debug applications running on the evaluation board itself.

[1] http://www.luminarymicro.com/products/software_updates.html

The Stellaris chip on the evaluation board is automatically disabled when the JTAG header on the evaluation board is connected to a production system. In this pass-through configuration, the Sourcery G++ Debug Sprite automatically connects directly to the production system instead. Since the production system may require a different initialization sequence than the evaluation kit, when invoking the Debug Sprite you should specify a configuration file that matches the production system rather than the evaluation board.

7.5.3. Troubleshooting

Some Stellaris boards have a `USB_OFF` jumper. You must remove this jumper so that the Debug Sprite can connect to the board via USB.

If you power-cycle the Stellaris board, unplug it from your computer, or use the reset button on the board while you are debugging, the debugger will not notice the reset, and debugging operations will stop working. At this point, you should exit and re-start from the debugger. If you are using the Eclipse IDE, choose `Terminate` from the `Run` menu.

7.6. Remote Debug Interface Devices

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

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

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

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

`rdi-config=`*`configfile`* Specify a file containing configuration information for *`library`*. The format of this file is specific to the RDI library you are using, but tends to constitute a list of *`key=value`* pairs. Consult the documentation of your RDI library for details.

7.7. Actel FlashPro Devices

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

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

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

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

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

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

you encounter communication errors with a higher-than-default speed selected, try reducing the speed.

7.7.1. Installing FlashPro Windows drivers

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

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

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

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

7.8. Keil ULINK2 Devices

Keil ULINK2 devices are supported on Windows hosts. For Cortex-M targets (e.g. Cortex-M1, Cortex-M3) the ULINK2 device partitions the *`device-url`* as follows:

```
ulink://cm/?opts=file
```

For older ARM targets (e.g. ARM7TDMI, ARM9), use the following instead:

```
ulink://arm/?opts=file[?device-options]
```

The `opts` option is mandatory, and is used to specify an options file. See Section 7.8.1, “Configuring ULINK2 Options” for instructions on creating this file.

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

`semihosting=`*`setting`*
: This option is for ARM7 and ARM9 targets. On these targets, the Debug Sprite implements semihosting using the `SWI` (also known as `SVC`) instruction. If your application uses the `SWI` instruction for other purposes and does not use semihosting, you can improve the performance of your application by disabling the semihosting support in the Sprite.

  Specify `semihosting=0` to disable semihosting support in the Debug Sprite. The default is to enable semihosting, equivalent to `semihosting=1`.

  If you choose to disable semihosting support in the Debug Sprite, you should link your application with an unhosted linker script. Refer to Chapter 6, “CS3™: The CodeSourcery Common Startup Code Sequence” for details.

7.8.1. Configuring ULINK2 Options

Before you can use the ULINK2 device for debugging, you must configure various JTAG and flash properties for your board. A graphical user interface is provided to assist you with this configuration.

When invoking the Debug Sprite from the command line, you must perform this configuration step first, and save the settings in an options file that you pass to the Sprite for debugging.

If you are using the Sourcery G++ IDE, there are buttons on the `ARM Settings` subtab that bring up the configuration dialogs described in this section. You do not need to run the Debug Sprite from the command line to create an options file, or specify an options file explicitly in the IDE.

Start by setting the JTAG properties. In the Sourcery G++ IDE, click on the corresponding button. From the command line, this dialog is opened by invoking the Sourcery G++ Debug Sprite with a `config` option. For Cortex-M targets, use:

```
> arm-none-eabi-sprite "ulink://cm/?opts=file&config=debug"
```

For older ARM targets, use:

```
> arm-none-eabi-sprite "ulink://arm/?opts=file&config=debug"
```

Replace *`file`* with the name of the options file to create. Note that you need the double quotes to prevent the shell from treating `&` as a special character.

This opens a dialog box on your desktop, which allows you to configure various debugging options.

- You can change the communication speed with your target board. Beware that rates above 1MHz may not work reliably.
- The `Cache Code` and `Cache Memory` options in the `Cache Options` group may speed up successive memory read operations from your board. It is safe to leave these two options checked.
- The `Verify Code Download` and `Download to Flash` options in the `Download Options` group should be left checked.
- The `Use Reset at Startup` option should be left checked.
- The `JTAG Device Chain` group can be used to override the default processor core detected by ULINK2. Leaving it set to `Automatic Detection` is usually the right thing to do, unless you have special requirements.

Clicking `OK` saves the configured options and closes the dialog.

The Sprite provides transparent support for programming CFI flash devices. To program other types of flash devices with ULINK2, you must configure a separate set of options for this. In the Sourcery G++ IDE, use the `ULINK Flash properties` button. From the command line, invoke the Sprite again with the appropriate `config` option. For Cortex-M targets, use:

```
> arm-none-eabi-sprite "ulink://cm/?opts=file&config=flash"
```

For older ARM targets, use:

```
> arm-none-eabi-sprite "ulink://arm/?opts=file&config=flash"
```

Use the same *`file`* that you previously used to store the JTAG debug configuration information.

This opens another dialog box on your desktop. There are several options which you can configure:

- In the `Download Function` group, leave the radio buttons for erase behavior set to `Erase Sectors`. Check the `Program` and `Verify` options, but leave the `Reset and Run` option unchecked.
- Add the flash device suitable for your target board by clicking `Add`, then choosing your device from the pop-up list.
- In the `RAM for Algorithm` group, choose a suitable RAM area for use as scratch space during flash programming.

Click `OK` to save your flash programming options.

7.8.2. ULINK2 Target Boards

ULINK2 supports many target boards and processors; refer to Keil documentation for a full list. The Sourcery G++ Debug Sprite can be used with any target supported by the ULINK2 drivers as long as an appropriate *`board-file`* is used. See Section 7.13, “Supported Board Files” for a list of boards supported out-of-the-box by Sourcery G++. You can use the Board Builder in the Sourcery G++ IDE to generate a *`board-file`* for other targets. For instructions, refer to Section 4.4.1, “Using the Sourcery G++ Board Builder”.

7.8.3. Installing ULINK2 Windows Drivers

No special driver is needed for ULINK2 devices.

7.9. Altera Devices

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

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

```
altera://
```

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

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

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

The Debug Sprite can autodetect connected Altera Cortex-M1 devices. Invoking the Sprite with the `-i` option, as described in Section 7.1, “Probing for Debug Devices”, displays the *`device-url`* for each detected device:

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

7.9.1. Setting Up the Altera Device

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

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

7.9.2. Hardware Breakpoints

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

7.10. SEGGER J-Link Devices

The Debug Sprite supports ARM7TDMI, ARM9, Cortex-M3, Cortex-M4, Cortex-R4, Cortex-A5, and Cortex-A8 cores via SEGGER J-Link on both Microsoft Windows and GNU/Linux hosts.

There are several forms of the *`device-url`* for J-Link devices. If you have only one J-Link device connected, you can use:

```
jlink:///
```

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

```
jlink://[address]/[serial][?device-options]
```

To connect multiple J-Link devices to one host system, each J-Link must be configured with a different USB address. To configure the USB address, see "Connecting multiple J-Links / J-Traces to your PC" in the J-Link User Guide.

The optional *`address`* is the configured USB address of the J-Link device, e.g. `usb1`. The optional *`serial`* is the serial number of the device. You may identify a device by either address or serial number.

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

`settings-file=`*`file`* This option allows you to specify a file to save J-Link settings. You can use the J-Link control panel to modify additional J-Link settings, including the JTAG speed and a target-specific script. Your changes will be saved in the settings file.

`semihosting=`*`setting`* Semihosting is implemented using the `SVC` (also known as `SWI`) instruction on classic ARM targets (including ARM7, ARM9, Cortex-

A, and Cortex-R). If your application uses the `SWI` instruction for other purposes and does not use semihosting, you can improve the performance of your application by disabling the semihosting support in the Sprite.

Specify `semihosting=0` to disable `SVC` semihosting support in the Debug Sprite. The default is to enable semihosting, equivalent to `semihosting=1`.

If you choose to disable semihosting support in the Debug Sprite, you should link your application with an unhosted linker script. Refer to Chapter 6, "CS3™: The CodeSourcery Common Startup Code Sequence" for details.

Semihosting on Thumb-only devices (including Cortex-M3 and Cortex-M4) uses the `BKPT` instruction which is not affected by this option.

Flash programming is supported on CFI-compliant devices and those devices listed in Section 6.3 of J-Link ARM User Guide as of J-Link ARM V4.20.

The Debug Sprite does not require or utilize additional software components from SEGGER, such as J-Link ARM FlashBP or J-Link GDB Server.

Connecting to a Cortex-A8 device may require an additional script to configure or reset the target. Sample scripts are installed by the J-Link driver installer. Some devices also require a hardware adaptor. Contact CodeSourcery or SEGGER for more information about particular devices.

7.10.1. Configuration on Microsoft Windows Hosts

You must install drivers for the SEGGER J-Link device before you can use it to debug your program. These drivers are bundled with Sourcery G++ for your convenience. You can find them at `libexec/arm-none-eabi-post-install/sprite-drivers/Setup_JLinkARM_`*`version`*`.exe` in your Sourcery G++ installation directory. Simply execute the file and follow instructions from the installer.

7.10.2. Configuration on GNU/Linux Hosts

On GNU/Linux systems, no additional drivers are necessary to use J-Link devices. The Sourcery G++ Debug Sprite is bundled with an open-source USB driver as well as the proprietary SEGGER shared library.

Make sure that you have privilege to access the USB device. Rather than running the Sourcery G++ Debug Sprite as root, the preferred solution on modern Linux distributions is to copy `libexec/arm-none-eabi-post-install/sprite-drivers/45-jlink.rules` to `/etc/udev/rules.d/`. You need to reboot your system for the changes to go into effect.

7.11. P&E Devices

The Sourcery G++ Debug Sprite supports Freescale Kinetis evaluation boards with debug devices from P&E Microcomputer Systems. The Tower microcontroller modules, such as the TWR-K40X256 and TWR-K60N512, include an on-board OSJTAG device which uses the P&E USB Multilink drivers. Sourcery G++ for ARM EABI also supports P&E Cyclone MAX devices with Kinetis targets.

The *`device-url`* for P&E devices takes the following form:

```
pe:[//type[/number]]
```

The *type* specifies the device type. The interpretation of *number* depends on the device type.

The following *type* keywords are supported in Sourcery G++ for ARM EABI:

`USBMultilink`	The *number* is used to distinguish between multiple connected devices; it ranges between 0 and 8, with 0 being the first detected device.
`CycloneProMaxUSB`	*number* ranges between 0 and 18, with 0 being the first detected device.
`CycloneProMaxSerial`	On Linux hosts, *number* ranges between 0 and 7, with 0 corresponding to `/dev/ttyS0`. On Windows hosts, *number* ranges between 1 and 8, with 1 corresponding to `COM1`.
`CycloneProMaxEthernet`	*number* specifies the IP address or host name of the P&E device.

The abbreviated *device-url* form `pe:` is equivalent to `pe://USBMultilink/0`.

7.11.1. Configuration on Microsoft Windows Hosts

You must install drivers for the P&E device before you can use it to debug your program. These drivers are bundled with Sourcery G++ for your convenience. You can find them at `libexec/arm-none-eabi-post-install/sprite-drivers/drivers_osbdm_install.exe` in your Sourcery G++ installation directory. Simply execute the file and follow instructions from the installer.

7.11.2. Configuration on GNU/Linux Hosts

On GNU/Linux systems, no additional drivers are necessary to use P&E devices. The Sourcery G++ Debug Sprite is bundled with the proprietary P&E shared library.

Make sure that you have privilege to access the USB device. Rather than running the Sourcery G++ Debug Sprite as root, the preferred solution on modern Linux distributions is to copy `libexec/arm-none-eabi-post-install/sprite-drivers/45-pe.rules` to `/etc/udev/rules.d/`. You need to reboot your system for the changes to go into effect.

7.12. Debugging a Remote Board

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

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

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

starts the Sprite listening on port 10000.

If you are using the Sourcery G++ IDE, you can use the External Embedded Server debugger to connect to the remote Sprite; Section 4.3.2.3, “Sourcery G++ External Embedded Server”. When running GDB from the command line, use the following command to connect GDB to the remote Sprite:

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

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

For more detailed instructions on using the Sourcery G++ Debug Sprite in this way, please refer to the Sourcery G++ Knowledge Base[2].

7.13. Supported Board Files

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

Board	Config	Register Browsing
Actel CoreMP7 Cortex-M1	coremp7-cm1	
Altera Cyclone III Cortex-M1	cycloneiii-cm1	
ARMulator (RDI)	armulator	
Atmel AT91SAM7S	at91sam7s-ek	
Energy Micro EFM32-G2XX-DK	efm32-g2xx-dk	yes
Energy Micro EFM32-G8XX-DK	efm32-g8xx-dk	yes
Energy Micro EFM32-G8XX-STK	efm32-g8xx-stk	yes
Freescale i.MX233 (with Mobile DDR)	imx233mddr	yes
Freescale i.MX233 EVK	imx233evk	yes
Freescale i.MX31 ADS	imx31	
Freescale TWR-K40X256	twr-k40x256	
Freescale TWR-K60N512	twr-k60n512	yes
Keil MCB1760	mcb1760	
Keil MCB2100	mcb2100	
Keil MCB2130	mcb2130	
Keil MCB2140	mcb2140	
Keil MCB2470	mcb2470	
Keil MCBSTM32	mcbstm32	yes
Keil MCBSTR7 (flash boot)	str710-flashboot	
Keil MCBSTR7 (ram boot)	str710-ramboot	
Keil MCBSTR9	str91x	
Keil Microcontroller Prototyping System (Cortex-M0)	mps-cm0	
Keil Microcontroller Prototyping System (Cortex-M1)	mps-cm1	

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

Board	Config	Register Browsing
Keil Microcontroller Prototyping System (Cortex-M3)	`mps-cm3`	
Keil Microcontroller Prototyping System (Cortex-M4)	`mps-cm4`	
PHYTEC phyCore-LPC3250	`phycore-lpc3250`	
RealView EB Cortex-M1	`realview-cm1`	
Stellaris LM3S101	`lm3s101`	yes
Stellaris LM3S102	`lm3s102`	yes
Stellaris LM3S1110	`lm3s1110`	yes
Stellaris LM3S1133	`lm3s1133`	yes
Stellaris LM3S1138	`lm3s1138`	yes
Stellaris LM3S1150	`lm3s1150`	yes
Stellaris LM3S1162	`lm3s1162`	yes
Stellaris LM3S1165	`lm3s1165`	yes
Stellaris LM3S1332	`lm3s1332`	yes
Stellaris LM3S1435	`lm3s1435`	yes
Stellaris LM3S1439	`lm3s1439`	yes
Stellaris LM3S1512	`lm3s1512`	yes
Stellaris LM3S1538	`lm3s1538`	yes
Stellaris LM3S1601	`lm3s1601`	yes
Stellaris LM3S1607	`lm3s1607`	yes
Stellaris LM3S1608	`lm3s1608`	yes
Stellaris LM3S1620	`lm3s1620`	yes
Stellaris LM3S1621	`lm3s1621`	yes
Stellaris LM3S1625	`lm3s1625`	yes
Stellaris LM3S1626	`lm3s1626`	yes
Stellaris LM3S1627	`lm3s1627`	yes
Stellaris LM3S1635	`lm3s1635`	yes
Stellaris LM3S1637	`lm3s1637`	yes
Stellaris LM3S1651	`lm3s1651`	yes
Stellaris LM3S1751	`lm3s1751`	yes
Stellaris LM3S1776	`lm3s1776`	yes
Stellaris LM3S1811	`lm3s1811`	yes
Stellaris LM3S1816	`lm3s1816`	yes
Stellaris LM3S1850	`lm3s1850`	yes
Stellaris LM3S1911	`lm3s1911`	yes
Stellaris LM3S1918	`lm3s1918`	yes
Stellaris LM3S1937	`lm3s1937`	yes
Stellaris LM3S1958	`lm3s1958`	yes
Stellaris LM3S1960	`lm3s1960`	yes

Board	Config	Register Browsing
Stellaris LM3S1968	`lm3s1968`	yes
Stellaris LM3S1B21	`lm3s1b21`	yes
Stellaris LM3S1J11	`lm3s1j11`	yes
Stellaris LM3S1J16	`lm3s1j16`	yes
Stellaris LM3S1N11	`lm3s1n11`	yes
Stellaris LM3S1N16	`lm3s1n16`	yes
Stellaris LM3S1P51	`lm3s1p51`	yes
Stellaris LM3S1R21	`lm3s1r21`	yes
Stellaris LM3S1R26	`lm3s1r26`	yes
Stellaris LM3S1W16	`lm3s1w16`	yes
Stellaris LM3S1Z16	`lm3s1z16`	yes
Stellaris LM3S2110	`lm3s2110`	yes
Stellaris LM3S2139	`lm3s2139`	yes
Stellaris LM3S2276	`lm3s2276`	yes
Stellaris LM3S2410	`lm3s2410`	yes
Stellaris LM3S2412	`lm3s2412`	yes
Stellaris LM3S2432	`lm3s2432`	yes
Stellaris LM3S2533	`lm3s2533`	yes
Stellaris LM3S2601	`lm3s2601`	yes
Stellaris LM3S2608	`lm3s2608`	yes
Stellaris LM3S2616	`lm3s2616`	yes
Stellaris LM3S2620	`lm3s2620`	yes
Stellaris LM3S2637	`lm3s2637`	yes
Stellaris LM3S2651	`lm3s2651`	yes
Stellaris LM3S2671	`lm3s2671`	yes
Stellaris LM3S2678	`lm3s2678`	yes
Stellaris LM3S2730	`lm3s2730`	yes
Stellaris LM3S2739	`lm3s2739`	yes
Stellaris LM3S2776	`lm3s2776`	yes
Stellaris LM3S2793	`lm3s2793`	yes
Stellaris LM3S2911	`lm3s2911`	yes
Stellaris LM3S2918	`lm3s2918`	yes
Stellaris LM3S2939	`lm3s2939`	yes
Stellaris LM3S2948	`lm3s2948`	yes
Stellaris LM3S2950	`lm3s2950`	yes
Stellaris LM3S2965	`lm3s2965`	yes
Stellaris LM3S2B93	`lm3s2b93`	yes
Stellaris LM3S300	`lm3s300`	yes

Board	Config	Register Browsing
Stellaris LM3S301	`lm3s301`	yes
Stellaris LM3S308	`lm3s308`	yes
Stellaris LM3S310	`lm3s310`	yes
Stellaris LM3S315	`lm3s315`	yes
Stellaris LM3S316	`lm3s316`	yes
Stellaris LM3S317	`lm3s317`	yes
Stellaris LM3S328	`lm3s328`	yes
Stellaris LM3S3634	`lm3s3634`	yes
Stellaris LM3S3651	`lm3s3651`	yes
Stellaris LM3S3739	`lm3s3739`	yes
Stellaris LM3S3748	`lm3s3748`	yes
Stellaris LM3S3749	`lm3s3749`	yes
Stellaris LM3S3826	`lm3s3826`	yes
Stellaris LM3S3J26	`lm3s3j26`	yes
Stellaris LM3S3N26	`lm3s3n26`	yes
Stellaris LM3S3W26	`lm3s3w26`	yes
Stellaris LM3S3Z26	`lm3s3z26`	yes
Stellaris LM3S5632	`lm3s5632`	yes
Stellaris LM3S5651	`lm3s5651`	yes
Stellaris LM3S5652	`lm3s5652`	yes
Stellaris LM3S5656	`lm3s5656`	yes
Stellaris LM3S5662	`lm3s5662`	yes
Stellaris LM3S5732	`lm3s5732`	yes
Stellaris LM3S5737	`lm3s5737`	yes
Stellaris LM3S5739	`lm3s5739`	yes
Stellaris LM3S5747	`lm3s5747`	yes
Stellaris LM3S5749	`lm3s5749`	yes
Stellaris LM3S5752	`lm3s5752`	yes
Stellaris LM3S5762	`lm3s5762`	yes
Stellaris LM3S5791	`lm3s5791`	yes
Stellaris LM3S5951	`lm3s5951`	yes
Stellaris LM3S5956	`lm3s5956`	yes
Stellaris LM3S5B91	`lm3s5b91`	yes
Stellaris LM3S5K31	`lm3s5k31`	yes
Stellaris LM3S5K36	`lm3s5k36`	yes
Stellaris LM3S5P31	`lm3s5p31`	yes
Stellaris LM3S5P36	`lm3s5p36`	yes
Stellaris LM3S5P51	`lm3s5p51`	yes

Board	Config	Register Browsing
Stellaris LM3S5P56	`lm3s5p56`	yes
Stellaris LM3S5R31	`lm3s5r31`	yes
Stellaris LM3S5R36	`lm3s5r36`	yes
Stellaris LM3S5T36	`lm3s5t36`	yes
Stellaris LM3S5Y36	`lm3s5y36`	yes
Stellaris LM3S600	`lm3s600`	yes
Stellaris LM3S601	`lm3s601`	yes
Stellaris LM3S608	`lm3s608`	yes
Stellaris LM3S610	`lm3s610`	yes
Stellaris LM3S6100	`lm3s6100`	yes
Stellaris LM3S611	`lm3s611`	yes
Stellaris LM3S6110	`lm3s6110`	yes
Stellaris LM3S612	`lm3s612`	yes
Stellaris LM3S613	`lm3s613`	yes
Stellaris LM3S615	`lm3s615`	yes
Stellaris LM3S617	`lm3s617`	yes
Stellaris LM3S618	`lm3s618`	yes
Stellaris LM3S628	`lm3s628`	yes
Stellaris LM3S6420	`lm3s6420`	yes
Stellaris LM3S6422	`lm3s6422`	yes
Stellaris LM3S6432	`lm3s6432`	yes
Stellaris LM3S6537	`lm3s6537`	yes
Stellaris LM3S6610	`lm3s6610`	yes
Stellaris LM3S6611	`lm3s6611`	yes
Stellaris LM3S6618	`lm3s6618`	yes
Stellaris LM3S6633	`lm3s6633`	yes
Stellaris LM3S6637	`lm3s6637`	yes
Stellaris LM3S6730	`lm3s6730`	yes
Stellaris LM3S6753	`lm3s6753`	yes
Stellaris LM3S6911	`lm3s6911`	yes
Stellaris LM3S6918	`lm3s6918`	yes
Stellaris LM3S6938	`lm3s6938`	yes
Stellaris LM3S6950	`lm3s6950`	yes
Stellaris LM3S6952	`lm3s6952`	yes
Stellaris LM3S6965	`lm3s6965`	yes
Stellaris LM3S800	`lm3s800`	yes
Stellaris LM3S801	`lm3s801`	yes
Stellaris LM3S808	`lm3s808`	yes

Board	Config	Register Browsing
Stellaris LM3S811	`lm3s811`	yes
Stellaris LM3S812	`lm3s812`	yes
Stellaris LM3S815	`lm3s815`	yes
Stellaris LM3S817	`lm3s817`	yes
Stellaris LM3S818	`lm3s818`	yes
Stellaris LM3S828	`lm3s828`	yes
Stellaris LM3S8530	`lm3s8530`	yes
Stellaris LM3S8538	`lm3s8538`	yes
Stellaris LM3S8630	`lm3s8630`	yes
Stellaris LM3S8730	`lm3s8730`	yes
Stellaris LM3S8733	`lm3s8733`	yes
Stellaris LM3S8738	`lm3s8738`	yes
Stellaris LM3S8930	`lm3s8930`	yes
Stellaris LM3S8933	`lm3s8933`	yes
Stellaris LM3S8938	`lm3s8938`	yes
Stellaris LM3S8962	`lm3s8962`	yes
Stellaris LM3S8970	`lm3s8970`	yes
Stellaris LM3S8971	`lm3s8971`	yes
Stellaris LM3S9781	`lm3s9781`	yes
Stellaris LM3S9790	`lm3s9790`	yes
Stellaris LM3S9792	`lm3s9792`	yes
Stellaris LM3S9997	`lm3s9997`	yes
Stellaris LM3S9B81	`lm3s9b81`	yes
Stellaris LM3S9B90	`lm3s9b90`	yes
Stellaris LM3S9B92	`lm3s9b92`	yes
Stellaris LM3S9B95	`lm3s9b95`	yes
Stellaris LM3S9B96	`lm3s9b96`	yes
Stellaris LM3S9L97	`lm3s9l97`	yes
STMicroelectronics STM3210B-EVAL	`stm3210b-eval`	yes
STMicroelectronics STM3210C-EVAL	`stm3210c-eval`	yes
STMicroelectronics STM3210E-EVAL	`stm3210e-eval`	yes
Xilinx Cortex-A9	`xilinxa9`	

7.14. Board File Syntax

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

The file's DTD is:

```
<!-- Board description files

     Copyright (c) 2007-2009 CodeSourcery, Inc.

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

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

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

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

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

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

```
                        device  CDATA   #IMPLIED>

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

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

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

The following elements are available:

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

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

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

`banked-regs` The `banked-regs` property specifies that the CPU of the target board has banked registers for different processor modes (supervisor, IRQ, etc.).

`has-vfp` The `has-vfp` property specifies that the CPU of the target board has VFP registers.

`system-v6-m` The `system-v6-m` property specifies that the CPU of the target board has ARMv6-M architecture system registers.

`system-v7-m` The `system-v7-m` property specifies that the CPU of the target board has ARMv7-M architecture system registers.

`core-family` The `core-family` property specifies the ARM family of the target. The body of the `<property>` element may be one of `arm7`, `arm9`, `arm11`, and `cortex`.

`system-clock` This property specifies the target clock frequency (in Hertz) after reset. It is used to configure flash programming algorithms.

`<initialize>` The `<initialize>` element defines an initialization sequence for the board, which the Sprite performs before downloading a program. It can

contain <write-register>, <write-memory> and <delay> elements.

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

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

<memory-device> This element specifies a region of memory. It has four attributes: address, size, type and device. The address and size attributes specify the location of the memory device. The type attribute specifies that device as ram, rom or flash. The device attribute is required for flash regions; it specifies the flash device type. Supported flash device types include at91sam7sxxx, cfi, lpc21xx, stellaris, stm32f10xxx, str91xfa, and ulink; not all flash devices are supported by all debugging devices. Additional flash device types are supported if you are using the Sourcery G++ Debug Sprite with SEGGER J-Link. For more information, refer to Section 7.10, "SEGGER J-Link Devices".. The <memory-device> element can contain a <description> element.

It can also contain the following named <property> element for additional flash-specific information:

programaddress This numeric property is used for ulink flash devices. It specifies an alias for the flash region in the memory map that should be used for programming the flash device, independently of the address the CPU uses.

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

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

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

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

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

Chapter 8
Next Steps with Sourcery G++

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

8.1. Sourcery G++ Support

If you have a Sourcery G++ subscription, you may manage your account by visiting the Sourcery G++ Portal[1]. The Portal gives you access to technical support, the latest software updates, and the Sourcery G++ Knowledge Base.

If you have a support account, but are unable to log in to the Portal, send email to `<support@codesourcery.com>` for assistance.

8.2. Sourcery G++ Knowledge Base

The Sourcery G++ Knowledge Base is available to registered users at the Sourcery G++ Portal[2]. 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.

8.3. Example Programs

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

The `StellarisWare` subdirectory contains examples for TI Stellaris boards that illustrate target-specific features such as peripheral I/O. The example programs and libraries are packaged for easy import into the Sourcery G++ IDE; refer to Section 3.5.2, “Using StellarisWare with Sourcery G++” for more details.

The `stm32` subdirectory similarly contains sample programs and libraries to illustrate the use of peripheral devices on STMicroelectronics STM32 boards. These examples are packaged for easy import into the Sourcery G++ IDE, as described in Section 3.6, “Using Sourcery G++ with STM32 Boards”.

The `Kinetis` subdirectory likewise contains sample programs and libraries specific to Freescale Kinetis targets. Refer to Section 3.4, “Using Sourcery G++ with Kinetis Boards” for more information and instructions for importing the examples into the Sourcery G++ IDE.

The remaining subdirectories contain a number of small, target-independent test programs. You may find these programs useful as self-contained test cases when experimenting with configuring the correct compiler and debugger settings for your target, or when learning how to use the debugger or other features of the Sourcery G++ toolchain. You can import these examples into the Sourcery G++ IDE using the procedure outlined in Section 4.4.4, “Importing Code into the IDE”.

8.4. Manuals for GNU Toolchain Components

Sourcery G++ 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.

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

When you install Sourcery G++, 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++, the documentation can be found in the `share/doc/sourceryg++-arm-none-eabi/` subdirectory of your installation directory.

You can also access the manuals from the `Help` menu in the Sourcery G++ IDE. See Section 8.5, "Help for the Sourcery G++ IDE", below.

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

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

Then you can invoke `man` as:

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

Alternatively, if you use `man` regularly, you'll probably find it more convenient to add the directory containing the Sourcery G++ man pages to your `MANPATH` environment variable. This should go in your `.profile` or equivalent shell startup file; see Section 2.7, "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++ 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.

8.5. Help for the Sourcery G++ IDE

The Sourcery G++ IDE, which is based on Eclipse and its C/C++ Development Toolkit, includes an extensive online help facility. To access this information, select `Help Contents` from the `Help` menu in the IDE toolbar.

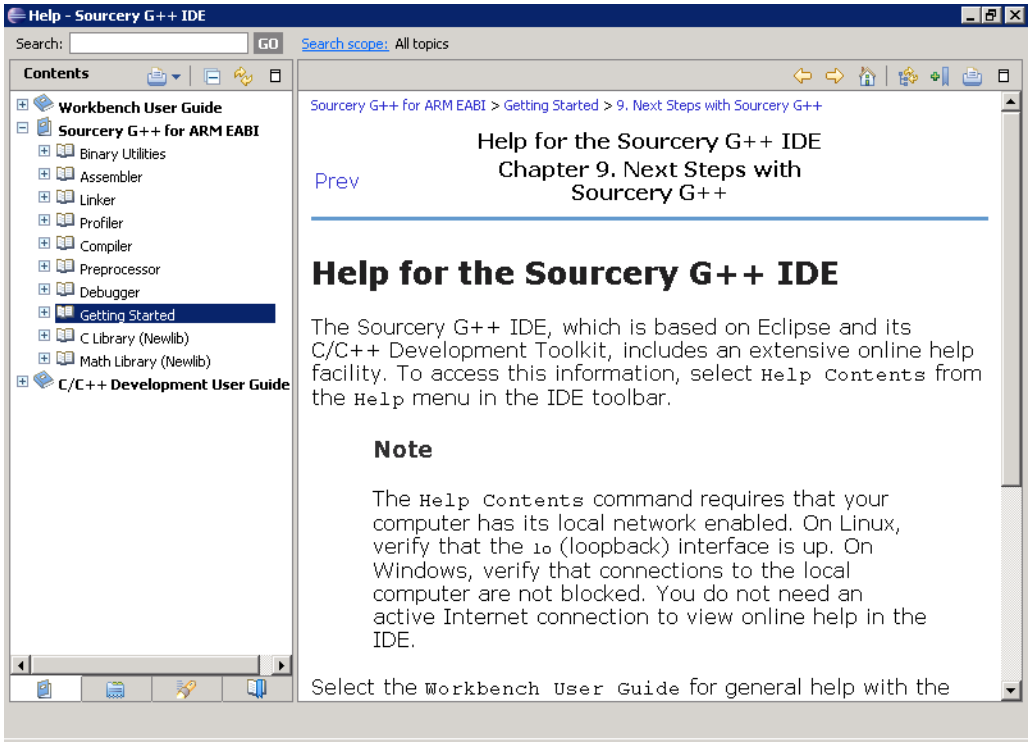
Note

The `Help Contents` command requires that your computer has its local network enabled. On Linux, verify that the `lo` (loopback) interface is up. On Windows, verify that connections to the local computer are not blocked. You do not need an active Internet connection to view online help in the IDE.

Select the `Workbench User Guide` for general help with the Eclipse IDE. Topics discussed in this manual include using the editor, file operations, and managing views and bookmarks.

Select the `C/C++ Development Toolkit User Guide` for assistance with using project templates, importing existing projects into the IDE, using Makefile mode, using and customizing features of the editor specific to C and C++ code, and using the debugger GUI.

Select `Sourcery G++ for ARM EABI` to view the Getting Started Guide and manuals for GNU toolchain components included with this version of Sourcery G++.



IDE Help. Sourcery G++ manuals are available in the IDE from the `Help Contents` menu command.

Appendix A
Sourcery G++ Release Notes

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

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

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

A.1.1. Changes in Sourcery G++ 2011.02-2

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

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

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

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

License manager bug fix for Windows hosts. A bug in the license manager has been fixed that sometimes caused a confusing `FLEXnet License Finder` dialog to pop up on the Windows desktop when running licensed tools, such as the compiler, without first installing a Sourcery G++ license. Sourcery G++ does not use this method to locate a license. Instead, you should use the Licensing wizard from the Sourcery G++ IDE to install a license file.

Floating license checkout failure fixed. A bug that caused floating license checkouts on GNU/Linux hosts to fail with a `This platform not authorized by license.` error has been fixed. To correct the error update both the license server manager and the CodeSourcery vendor daemon to the latest version. For more information about installing a floating license server please refer to the Sourcery G++ Knowledge Base[1].

Example programs for Kinetis. Sourcery G++ now includes a set of example programs for Freescale Kinetis targets. The examples have been bundled for easy import into the Sourcery G++ IDE. For more information, refer to Section 3.4, “Using Sourcery G++ with Kinetis Boards”.

Importing Sourcery G++ Example Projects. A bug in the Sourcery G++ IDE that caused selecting `Sourcery G++ Example Project` in the `Import` dialog to raise an error dialog box has been fixed.

Support for P&E devices. The Sourcery G++ Debug Sprite now supports Freescale Kinetis targets equipped with an OSJTAG device from P&E Microcomputer Systems. For more information, refer to Section 7.11, “P&E Devices”.

A.1.2. Changes in Sourcery G++ 2010.09-66

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

Add-on installation fix. A bug in the Sourcery G++ IDE that prevented it from installing add-ons from the Sourcery G++ Portal[2] has been fixed.

[1] https://support.codesourcery.com/GNUToolchain/kbentry198
[2] https://support.codesourcery.com/GNUToolchain/

Incorrect Board Builder diagnostic fixed. A bug in the Sourcery G++ IDE has been fixed that caused a pop-up dialog incorrectly asserting that files generated by the Board Builder were manually modified. The bug was specific to Windows hosts.

IDE static library projects. Bugs in the Sourcery G++ IDE have been fixed that caused the new project wizard to fail to create new static library projects, or other incorrect behavior when setting the project properties.

A.1.3. Changes in Sourcery G++ 2010.09-47

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

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

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

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

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

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

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

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

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

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

Extraneous editor window in IDE debugger. A bug in the Sourcery G++ IDE has been fixed that caused the debugger to open an editor window at the program start address when connecting to the target.

Map file name. The Sourcery G++ IDE now names map files with the `.map` extension replacing any extension on the executable file name, instead of simply appending to it.

IDE out-of-memory error fix. A bug has been fixed that caused the Sourcery G++ IDE to start with a low heap size. The bug could lead to `OutOfMemoryError` failures in the IDE.

GDB termination from IDE debugger. Bugs have been fixed that caused the Sourcery G++ IDE to fail to terminate GDB when the application being debugged exits, or when launching the debug session fails for any reason.

New IDE command-line build option. The IDE command-line builder now supports the `-refresh` option, which forces a workspace refresh before the build. Use this option if you have made changes to the project outside the IDE, such as adding or deleting files.

Terminate and Relaunch fix. The bug in the Sourcery G++ IDE that caused the `Terminate and Relaunch` command to not actually relaunch has been fixed.

Support for Freescale Kinetis devices. Sourcery G++ now includes CS3 support for Freescale Kinetis devices. Board support packages for the Freescale TWR-K40X256 and TWR-K60N512 are provided, and other boards based on Kinetis devices are supported via the Sourcery G++ Board Builder.

New manual for CodeSourcery C Library. The CodeSourcery C Library now includes its own manual, replacing the Newlib C Library and Math Library manuals formerly distributed with Sourcery G++.

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

Faster flash programming on Stellaris. The Sourcery G++ Debug Sprite can now program flash memory on Stellaris Sandstorm, Fury, and DustDevil devices significantly faster.

J-Link driver update and new CPU support. The J-Link driver included in Sourcery G++ has been updated to version 4.20h. The Sourcery G++ Debug Sprite can now use a J-Link to debug Cortex-M4, Cortex-R4, Cortex-A5, and Cortex-A8 cores.

A.1.4. Changes in Sourcery G++ 2010.09-15

Changes to Sourcery G++ version numbering. Sourcery G++ product and Lite toolchains now uniformly use a version numbering scheme of the form 2011.02-2. The major and minor parts of the version number, in this case 2011.02, identify the release branch, while the final component is a build number within the branch. There are also new preprocessor macros defined by the compiler for the version number components so that you may conditionalize code for Sourcery G++ or particular Sourcery G++ versions. Details are available in the Sourcery G++ Knowledge Base[3].

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

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

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

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

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

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

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

- More efficient assignment for structures containing bitfields.
- Better code for initializing C++ arrays with explicit element initializers.
- Improved logic for eliminating/combining redundant comparisons in code with nested conditionals.
- Better selection of loop variables, resulting in fewer temporaries and more efficient register usage.
- More optimization of references to globals in position-independent code.
- Various Thumb code generation improvements.
- Better code when constant addresses are used as arguments to inline assembly statements.
- Better code for copying small constant strings.
- Improved tuning for Cortex-M4 processors.
- Cortex-A9 specific tuning for VFP and NEON instructions.
- Use of more NEON features.

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

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

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

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

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

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

License files with non-ASCII characters. A bug in the Sourcery G++ IDE that resulted in license installation failures on Windows hosts has been fixed.

IDE displays selected multilib name. The project properties dialog in the Sourcery G++ IDE now identifies the multilib selected by the project build options. Additionally, when the selected multilib is an add-on library that is not installed, this is flagged in the project properties dialog. For more information about using the libraries included with Sourcery G++, refer to Section 3.2, "Library Configurations".

Fix for unnecessary project relinking. A bug in the Sourcery G++ IDE that caused project relink after every IDE restart has been fixed.

Importing an executable into the IDE. The Sourcery G++ IDE now includes an improved wizard for importing an existing executable. The new wizard correctly associates a Sourcery G++ toolchain with the program, and creates a Sourcery G++ debug launch by default. See Section 4.4.5, "Importing an Executable into the Sourcery G++ IDE" for details.

IDE post-build binary size display bug fixes. Two bugs in the post-build step in the Sourcery G++ IDE that displays the size of the binary have been fixed. One bug caused the command to fail when the build artifact pathname contains spaces. The second bug resulted in a command syntax error when building static libraries.

Sourcery G++ IDE update. The Sourcery G++ IDE has been updated to version 3.6 (Helios) of the Eclipse Platform and version 7.0 of the Eclipse C/C++ Development Tools (CDT). This update includes many bug fixes and usability improvements. In addition, the internal change to the new DSF debugger framework has resulted in a number of (mostly minor) changes to the debugger user interface. The CDT documentation[4] includes a detailed list of changes, but note that not all new features are applicable to Sourcery G++ for ARM EABI.

IDE memory view improvement. The IDE `Memory` window now better handles display and scrolling of memory contents at the lower end of a memory region. Previously, it sometimes reported bytes within the valid memory range as unavailable.

IDE external tools improvements. The predefined External Tool launches for GNU Binary Utilities queries now save output to a file which is automatically opened in an editor view, instead of writing to a console. This change only affects newly-created workspaces; to get the new behavior in an existing workspace, first delete the previous External Tool launch definitions, then restart the Sourcery G++ IDE.

IDE build errors for simulator projects. A bug in the Sourcery G++ IDE has been fixed that caused projects for simulator targets (i.e., QEMU) created with older versions of Sourcery G++ to fail to build when loaded or imported into a newer version of the IDE.

Support for NXP LPC17xx devices. Sourcery G++ now includes CS3 support for NXP LPC17xx devices. A board support package for the Keil MCB1760 is provided, and other boards based on these devices are supported via the Sourcery G++ Board Builder.

Support for Keil Microcontroller Prototyping System. Sourcery G++ now includes support for the MPS. The Cortex-M0, Cortex-M1, Cortex-M3, and Cortex-M4 images are all supported. Programs may execute from flash or SSRAM1.

[4] http://wiki.eclipse.org/CDT/User/NewIn70

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

Thumb support in Board Builder reset sequences. The Board Builder in the Sourcery G++ IDE now supports both ARM and Thumb in delay code used in reset sequences. Previously, Thumb was not supported.

Library footprint improvements. The Sourcery G++ libraries have been improved to reduce program size by simplifying the actions that are performed in some situations where the C and/or C++ standards do not specify behavior. Specifically:

- In freestanding (unhosted) configurations, `abort` now calls `__builtin_trap`, resulting in an immediate program crash. In previous releases, `abort` caused `SIGABRT` to be raised. The behavior in hosted configurations has not been changed because the ISO C standard requires that `SIGABRT` be raised in a hosted configuration.

- When a C++ pure virtual function is called, the internal `__cxa_pure_virtual` function now calls `__builtin_trap`. In previous releases, `__cxa_pure_virtual` printed a message before calling `std::terminate`.

- Recursive initialization of a C++ function-local `static` variable now results in a call to `__builtin_trap` instead of throwing an exception.

CSLIBC update. The CodeSourcery C Library has been updated to incorporate changes from Newlib 1.18.0. This update provides additional wide-character functions, along with other bug fixes and enhancements.

Improved support for debugging RealView® C++ programs . GDB has been enhanced to handle some debug information contained in binaries produced by the ARM RealView® compiler. Formerly, GDB sometimes crashed on programs which use C++ templates.

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

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

IDE debugger assignment to I/O registers. A bug has been fixed that caused a GDB crash when writing values to read-sensitive I/O registers that have not previously been read.

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

J-Link support on GNU/Linux hosts. The Sourcery G++ Debug Sprite now supports J-Link devices on GNU/Linux hosts as well as Microsoft Windows. For more information, refer to Section 7.10, “SEGGER J-Link Devices”.

Faster startup for J-Link debugging. The Sourcery G++ Debug Sprite now launches significantly faster when used with SEGGER J-Link devices and boards with external CFI-compatible flash devices, such as STMicroelectronics STM32 evaluation boards.

Updated ULINK2 support. The ULINK2 drivers, firmware, and flash algorithms included with Sourcery G++ have been updated to V4.11. The new drivers support debugging on additional devices, including NXP LPC11xx, NXP LPC13xx, Energy Micro EFM32, and Cortex-M4 processors. For more information on using the Sourcery G++ Debug Sprite with ULINK2 devices, see Section 7.8, "Keil ULINK2 Devices".

J-Link driver update. The J-Link driver included in Sourcery G++ has been updated to version 4.14i.

Debug Sprite fix for Stellaris targets at low system clocks. The Sourcery G++ Debug Sprite now accommodates Stellaris devices running at low system clocks. Previously, the Sprite sometimes reported spurious traps in this situation.

ULINK2 flash programming. The Sourcery G++ Debug Sprite now supports programming CFI-compatible flash devices with ULINK2 probes.

J-Link watchpoint support. The Sourcery G++ Debug Sprite now supports watchpoints with SEGGER J-Link devices.

Debug Sprite interrupt behavior fix. A bug in the ARMUSB and J-Link support in the Sourcery G++ Debug Sprite has been fixed. The bug sometimes caused the Sprite to ignore requests from the debugger to interrupt a running program on the target.

StellarisWare update. Sourcery G++ includes a more recent version of the bundled Stellarisware libraries (release 6459).

Dhrystone example program. The Dhrystone 2.1 benchmark has been added to the set of example programs bundled with Sourcery G++. Refer to Section 8.3, "Example Programs" for more information about bundled examples.

A.1.5. Changes in Older Releases

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

Appendix B
Sourcery G++ Licenses

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

B.1. Licenses for Sourcery G++ Components

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

Some free or open-source components provide documentation or other files under terms different from those shown below. For definitive information about the license that applies to each component, consult the source package corresponding to this release of Sourcery G++. Sourcery G++ 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++.

Component	**License**
GNU Compiler Collection	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
GNU Binary Utilities	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
Eclipse IDE	Eclipse Public License 1.0 http://www.eclipse.org/org/documents/epl-v10.php
Eclipse C/C++ Development Tools	Eclipse Public License 1.0 http://www.eclipse.org/org/documents/epl-v10.php
Sourcery G++ Eclipse Plugin(s)	CodeSourcery License and Eclipse Public License 1.0 http://www.eclipse.org/org/documents/epl-v10.php
Sourcery G++ IDE Launcher	CodeSourcery License
GNU Debugger	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
Sourcery G++ Cygwin GDB Wrapper	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
Sourcery G++ Debug Sprite for ARM	CodeSourcery License
QEMU Emulator	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
CodeSourcery Common Startup Code Sequence	CodeSourcery License
CodeSourcery C Library	CodeSourcery License
GNU Make	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
GNU Core Utilities	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

The CodeSourcery License refers to the license agreement under which you licensed Sourcery G++ from CodeSourcery, Inc. or its authorized distributor or reseller, including without limitation the Sourcery G++™ Software License Agreement (see Section B.2, “Sourcery G++ Software License Agreement” below).

Important

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

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

The Sourcery G++ IDE contains components licensed under the Eclipse Public License. In order to obtain the source code for these components, file a request for the source code in the Sourcery G++ Portal[1] specifying the Sourcery G++ version number, the target CPU, the target OS, and the host OS.

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

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

2. **The Software.** The Software licensed under this Agreement consists of computer programs and documentation referred to as Sourcery G++™ Professional Edition, Sourcery G++™ Standard Edition, Sourcery G++™ Personal Edition, Sourcery G++™ Academic Edition, or Sourcery G++™ for Evaluation (the "Software") provided to You, including any Updates thereto.

3. **Definitions.**

    3.1. **Effective Date.** The date on which CodeSourcery gives You access to CodeSourcery's electronic support system.

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

    3.3. **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.

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

    3.5. **Redistributable Components.** The CodeSourcery Proprietary Components that are intended to be incorporated or linked into Licensee object code developed with the Software. The Redistributable Components of the Software include, without limitation,

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

CSLIBC and the CodeSourcery Common Startup Code Sequence (CS3). For a complete list, refer to the *Getting Started Guide* included with the distribution.

4. **License Grant to Proprietary Components of the Software.**

   4.1. **Sourcery G++ Professional Edition.** Subject to the terms and conditions hereof, CodeSourcery hereby grants to Licensee of Sourcery G++ Professional Edition a perpetual, non-exclusive license under the Proprietary Rights of CodeSourcery and its licensors (a) to install and use the CodeSourcery Proprietary Components of the Software (i) if the license is a node-locked license, by a single user who uses the Software on up to two machines provided that only one copy of the Software is in use at any one time, or (ii) if the license is a floating license, by the authorized number of concurrent users on one or more machines provided that only the authorized number of copies of the Software are in use at any one time, and (b) to distribute the Redistributable Component(s) of the Software in binary form only and only as part of Licensee object code developed with the Software that provides substantially different functionality than the Redistributable Component(s).

   4.2. **Sourcery G++ Standard Edition.** Subject to the terms and conditions hereof, CodeSourcery hereby grants to Licensee of Sourcery G++ Standard Edition a perpetual, non-exclusive license under the Proprietary Rights of CodeSourcery and its licensors (a) to install and use the CodeSourcery Proprietary Components of the Software by a single user who uses the Software up to two machines provided that only one copy of the Software is in use at any one time, and (b) to distribute the Redistributable Component(s) of the Software in binary form only and only as part of Licensee object code developed with the Software that provides substantially different functionality than the Redistributable Component(s).

   4.3. **Sourcery G++ Personal Edition.** Subject to the terms and conditions hereof, CodeSourcery hereby grants to Licensee of Sourcery G++ Personal Edition a perpetual, non-exclusive license under the Proprietary Rights of CodeSourcery and its licensors (a) to install and use the CodeSourcery Proprietary Components of the Software by a single user who uses the Software on one machine, and (b) to distribute the Redistributable Component(s) of the Software in binary form only and only as part of Licensee object code developed with the Software that provides substantially different functionality than the Redistributable Component(s).

   4.4. **Sourcery G++ Academic Edition.** Subject to the terms and conditions hereof, CodeSourcery hereby grants to Licensee of Sourcery G++ Academic Edition a perpetual, non-exclusive license under the Proprietary Rights of CodeSourcery and its licensors (a) to install and use the CodeSourcery Proprietary Components of the Software, for non-commercial, academic purposes only, by a single user who uses the Software on one machine, and (b) to distribute the Redistributable Component(s) of the Software in binary form only and only as part of Licensee object code developed with the Software that provides substantially different functionality than the Redistributable Component(s).

   4.5. **Sourcery G++ for Evaluation.** Subject to the terms and conditions hereof, CodeSourcery hereby grants to Licensee of Sourcery G++ for Evaluation a limited, non-exclusive license under the Proprietary Rights of CodeSourcery and its licensors to install and use the CodeSourcery Proprietary Components of the Software, for evaluation purposes only, by a single user who uses the Software on one machine during the term of this Agreement.

5. **Restrictions.** You may not: (i) copy or permit others to use the CodeSourcery Proprietary Components of the Software, except as expressly provided above; (ii) distribute the CodeSourcery

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

5.1. **ARM Keil ULINK2 Drivers.** Sourcery G++ may include ULINK2 drivers from ARM, Ltd. If these drivers are included, the following additional terms and conditions apply:

a. You may use the ULINK2 drivers only in conjunction with a compatible ARM Keil ULINK2 hardware unit manufactured by or under license from ARM and purchased from CodeSourcery, ARM, or a distributor authorized by ARM.

b. You may use the ULINK2 drivers only to connect to the GNU Debugger included in Sourcery G++.

c. The ULINK2 drivers are not supported by ARM, Ltd.; You should contact CodeSourcery for any support regarding the ULINK2 drivers.

d. You may not redistribute or transfer the ULINK2 drivers.

e. You may not translate, adapt, arrange or otherwise alter the object code of the ULINK2 drivers (including without limitation copying, adapting or reverse compiling the object code of the ULINK2 drivers for the purpose of error correction) except as allowed by applicable law.

f. You may not remove or obstruct any notice or marker incorporated into the ULINK2 drivers to protect ARM's or third parties' intellectual property or Proprietary Rights.

g. The ULINK2 drivers are licensed, not sold; all right, title and interest therein is reserved to CodeSourcery or its licensors, and You acquire no right, title or interest therein.

5.2. **SEGGER J-Link™ Devices.** Sourcery G++ includes proprietary software from SEGGER Microcontroller GmbH & Co.KG that allows the use of SEGGER J-Link debug devices with the Sourcery G++ Debug Sprite. You may use software from SEGGER only under the SEGGER J-Link software terms of use [2] and license agreement[3].

5.3. **Sourcery G++ Debug Sprite with ARM SWD.** The Sourcery G++ Debug Sprite for ARM and Stellaris processors includes software for ARM SWD support from ARM, Ltd. You may use the ARM SWD software only in conjunction with a Cortex-M1 or Cortex-M3 microprocessor manufactured under license from ARM.

5.4. **Sourcery G++ Debug Sprite for P&E Devices.** This software application may include P&E NGS Drivers version 120210 third-party software, which is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. P&E NGS Drivers version 120210 may be subject to the following copyrights:

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

[2] http://segger.com/jlink_arm_software_terms_of_use.html
[3] http://segger.com/pub/jlink/license_agreement.txt

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

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

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

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

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

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

8. **Term.**

   8.1. **Sourcery G++ Professional, Standard, Personal, and Academic Edition Subscriptions.** For Licensee of Sourcery G++ Professional, Standard, Personal, or Academic Edition, this Agreement shall have a subscription term of one (1) year unless this Agreement is terminated pursuant to Section 12.

   8.2. **Sourcery G++ for Evaluation.** For Licensee of Sourcery G++ for Evaluation, this Agreement shall have an evaluation term of thirty (30) days unless this Agreement is terminated pursuant to Section 12.

8.3. **Renewals.** For Licensee of Sourcery G++ Professional, Standard, Personal, or Academic Edition, You may renew this Agreement for successive one (1) year subscription terms by paying the maintenance fee on or before the anniversary of the Effective Date of this Agreement.

8.4. **Lapsed Subscription.** If You desire to reinstate an Agreement that has been expired for more than thirty (30) days, You must pay the renewal fee for the any years during which Your subscription lapsed and the full license fee for the first year in which Your subscription is reinstated.

9. **Updates.** During the term of this Agreement, You may download, free of charge, any new version(s), update(s), or upgrade(s) ("Updates") to the Software that CodeSourcery makes available at such times as may be determined by CodeSourcery in its sole discretion.

10. **Technical Support.** CodeSourcery shall provide technical support only to Licensee of Sourcery G++ Professional or Standard Edition.

10.1. **Support Term.** CodeSourcery shall provide technical support to Licensee pursuant to this Section 10 during the subscription term of this Agreement.

10.2. **Scope of Support.** CodeSourcery shall assist Licensee in installing and using the Software in binary form. If Licensee reports a defect in the Software, CodeSourcery shall suggest a work-around, and, if Licensee is a Professional Edition subscriber, correct the defect, subject to the limitations set forth below. CodeSourcery shall impose no limit on the number of support requests made by Licensee. Licensee may submit support requests only on its own behalf and not on behalf of any other party, including a licensee of any other edition of Sourcery G++, such as Sourcery G++ Personal, Academic or Lite Edition. If Licensee is a member of a development team, all members of the team must be Licensees of the same edition of Sourcery G++ (either Professional or Standard Edition). Although CodeSourcery will make source code for the Open Source Components available to Licensee, CodeSourcery's support does not cover rebuilding the toolchains from the source packages, correcting defects in any such rebuilt toolchains, or answering any questions arising from Licensee's use of source packages.

10.3. **Professional Edition Update Releases.** For Licensee of Sourcery G++ for Professional Edition only, CodeSourcery shall use commercially reasonable efforts to provide an Update incorporating the correction of any defect reported by Licensee for which no satisfactory work-around exists.

10.4. **Electronic Support System.** Licensee shall make all support requests via CodeSourcery's electronic support system, and CodeSourcery shall respond via the same electronic support system. CodeSourcery will not accept support requests by telephone or other means.

10.5. **Response Time.** CodeSourcery's electronic support system will provide Licensee with an immediate acknowledgment of the support request (including a unique tracking number) by electronic mail. Support requests from Licensee of Sourcery G++ Professional Edition shall receive priority for response and resolution. CodeSourcery shall respond to all Sourcery G++ Professional Edition support requests within one business day and Sourcery G++ Standard Edition support requests within three business days except in extraordinary circumstances.

10.6. **No Guarantee of Resolution.** CodeSourcery does not guarantee that it will be able to resolve all support requests. Without limitation, CodeSourcery may, in its sole discretion, determine that a defect in the Software is too difficult to correct, or that any correc-

tion would likely risk the introduction of additional defects, or that the defect is not likely to be encountered often enough to be worthy of correction, or that the defect is insufficiently severe to be worthy of correction.

10.7. **Support for Previous Versions.** After the release of an Update, CodeSourcery shall provide support for previous version(s) of the Software for a period of six (6) months. CodeSourcery will have no obligation to provide support after this period. Long-term support for an older version of the Software is available to Licensee of Sourcery G++ Professional Edition for an additional fee.

10.8. **Test Cases.** In many cases, CodeSourcery will require access to Licensee's source code in order to resolve Licensee's support request. CodeSourcery may, in its sole discretion, create regression tests distilled from Licensee's source code for use in testing changes to the Software. CodeSourcery shall use commercially reasonable efforts to disguise the origin of the source code, to eliminate non-essential aspects of the source code, and to otherwise protect the confidentiality of Licensee's source code. These regression tests may be made available to other CodeSourcery licensees or to the general public.

11. **Fees.**

11.1. **Fees.** The licenses and rights granted in this Agreement are subject to Licensee's payment of all fees owed to CodeSourcery.

11.2. **Taxes.** All fees are exclusive of sales or use taxes and any levy imposed on the transportation or use of the Software. Licensee shall pay all such charges either as levied by taxing authorities or as invoiced by CodeSourcery.

11.3. **Non-refundability.** All payments made to CodeSourcery are non-refundable.

12. **Termination.**

12.1. **Grounds for Termination.**

12.1.1. **Termination for Material Breach.** 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.

12.1.2. **Termination of Updates and Technical Support for Unauthorized Redistribution of Binaries .** CodeSourcery shall provide Updates and Technical Support to Licensee, pursuant to Sections 9 and 10, respectively, only on the condition that Licensee uses the Software for internal use and/or distributes the Software in binary form pursuant to a separate redistribution agreement with CodeSourcery. Any other distribution by Licensee of the Software in binary form, including distribution permitted by the applicable "free software" or "open source" license, shall automatically terminate this Agreement and shall void the remainder of Licensee's subscription term.

12.2. **Effect of Termination.**

12.2.1. **Surviving Obligations.** The following obligations shall survive the termination or expiration of this Agreement: (i) any and all warranty disclaimers or limitations of liability herein, and (ii) any covenant granted herein for the purpose of determining ownership of, or protecting, the Proprietary Rights of

either party, including the CodeSourcery trademarks as set forth in Section 7, or any remedy for breach thereof.

12.2.2. **Surviving Rights.** After the expiration of this Agreement, Licensee of Sourcery G++ Professional, Standard, Personal, or Academic Edition may continue to use the Software, including the CodeSourcery Proprietary Components, pursuant to Section 4 and 5, but CodeSourcery shall provide no further Updates or Technical Support to Licensee. However, if this agreement is terminated pursuant to Section 12.1, Licensee's rights pursuant to Section 4 shall terminate immediately and Licensee shall destroy all copies of the CodeSourcery Proprietary Components of the Software.

13. **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.

14. **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.

15. **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.

16. **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.

17. **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 WHAT-

SOEVER, AND REGARDLESS OF THE FORM OF ACTION, EXCEED THE AMOUNT PAID BY YOU IN FEES UNDER THIS AGREEMENT DURING THE PREVIOUS ONE YEAR PERIOD.

18. **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.

19. **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.

20. **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.

21. **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.

22. **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.

23. **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.

24. **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.

25. **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.

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

B.3. Attribution

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

B.3.1. Android Open Source Project

```
/*
 * Copyright (C) 2008 The Android Open Source Project
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *  * Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *  * Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
 * COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
 * OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
 * AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
```

```
 * SUCH DAMAGE.
 */
```

B.3.2. Newlib

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

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

(1) Red Hat Incorporated

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

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

(2) University of California, Berkeley

Copyright (c) 1981-2000 The Regents of the University of California.
All rights reserved.

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

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright notice,
      this list of conditions and the following disclaimer in the documentation
      and/or other materials provided with the distribution.
    * Neither the name of the University nor the names of its contributors
      may be used to endorse or promote products derived from this software
      without specific prior written permission.

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

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

The author of this software is David M. Gay.

Copyright (c) 1991 by AT&T.

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

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

```
--------------------------------------------------------------------

The author of this software is David M. Gay.

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

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

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


(4) Advanced Micro Devices

Copyright 1989, 1990 Advanced Micro Devices, Inc.

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

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

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

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

(5) C.W. Sandmann

Copyright (C) 1993 C.W. Sandmann

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

(6) Eric Backus

(C) Copyright 1992 Eric Backus

This software may be used freely so long as this copyright notice is
left intact.  There is no warrantee on this software.

(7) Sun Microsystems

Copyright (C) 1993 by Sun Microsystems, Inc. All rights reserved.

Developed at SunPro, a Sun Microsystems, Inc. business.
Permission to use, copy, modify, and distribute this
```

```
software is freely granted, provided that this notice is preserved.

(8) Hewlett Packard

(c) Copyright 1986 HEWLETT-PACKARD COMPANY

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

(9) Hans-Peter Nilsson

Copyright (C) 2001 Hans-Peter Nilsson

Permission to use, copy, modify, and distribute this software is
freely granted, provided that the above copyright notice, this notice
and the following disclaimer are preserved with no changes.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.

(10) Stephane Carrez (m68hc11-elf/m68hc12-elf targets only)

Copyright (C) 1999, 2000, 2001, 2002 Stephane Carrez (stcarrez@nerim.fr)

The authors hereby grant permission to use, copy, modify, distribute,
and license this software and its documentation for any purpose, provided
that existing copyright notices are retained in all copies and that this
notice is included verbatim in any distributions. No written agreement,
license, or royalty fee is required for any of the authorized uses.
Modifications to this software may be copyrighted by their authors
and need not follow the licensing terms described here, provided that
the new terms are clearly indicated on the first page of each file where
they apply.

(11) Christopher G. Demetriou

Copyright (c) 2001 Christopher G. Demetriou
All rights reserved.

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

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

(12) SuperH, Inc.
```

```
Copyright 2002 SuperH, Inc. All rights reserved

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

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

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

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

(13) Royal Institute of Technology

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

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

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

2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.

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

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

(14) Alexey Zelkin

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

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

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
```

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

(15) Andrey A. Chernov

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

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

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

(16) FreeBSD

Copyright (c) 1997-2002 FreeBSD Project.
All rights reserved.

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

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

(17) S. L. Moshier

Author:  S. L. Moshier.

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

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

documentation for such software.

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

(18) Citrus Project

Copyright (c)1999 Citrus Project,
All rights reserved.

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

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

(19) Todd C. Miller

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

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

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

(20) DJ Delorie (i386)
Copyright (C) 1991 DJ Delorie
All rights reserved.

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

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

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

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

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

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

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

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

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

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

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

(23) Intel (i960)

Copyright (c) 1993 Intel Corporation

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

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

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

(24) Hewlett-Packard  (hppa targets only)

(c) Copyright 1986 HEWLETT-PACKARD COMPANY

To anyone who acknowledges that this file is provided "AS IS"
without any express or implied warranty:
    permission to use, copy, modify, and distribute this file
for any purpose is hereby granted without fee, provided that
the above copyright notice and this notice appears in all
copies, and that the name of Hewlett-Packard Company not be
```

```
used in advertising or publicity pertaining to distribution
of the software without specific, written prior permission.
Hewlett-Packard Company makes no representations about the
suitability of this software for any purpose.

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

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

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

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

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

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

4. This notice may not be removed or altered.

(26) Mike Barcroft

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

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

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

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

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

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

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
```

```
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

   iconv (Charset Conversion Library) v2.0

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

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

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

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

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

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

  All rights reserved.

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

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * Neither the names of the copyright holders nor the names of their
      contributors may be used to endorse or promote products derived from this
      software without specific prior written permission.

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

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

  Copyright (c) 1995 Alex Tatmanjants <alex@elvisti.kiev.ua>
```

```
   at Electronni Visti IA, Kiev, Ukraine.
    All rights reserved.

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

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

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

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

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

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

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

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

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

  THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND
  ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
  ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
  FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
  OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
```

```
  LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
  OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
  SUCH DAMAGE.

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

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

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

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


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

 Contributed by Jon Beniston <jon@beniston.com>

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

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


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

 Copyright (c) 2009 ARM Ltd
 All rights reserved.

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

```
   permission.

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


(36) - CodeSourcery, Inc.

Copyright (c) 2009 CodeSourcery, Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * Neither the name of CodeSourcery nor the
      names of its contributors may be used to endorse or promote products
      derived from this software without specific prior written permission.

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

(37) MIPS Technologies, Inc
/*
 * Copyright (c) 2009 MIPS Technologies, Inc.
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 *      * Redistributions of source code must retain the above copyright
 *        notice, this list of conditions and the following disclaimer.
 *      * Redistributions in binary form must reproduce the above
 *      copyright
 *        notice, this list of conditions and the following disclaimer
 *        in the documentation and/or other materials provided with
 *        the distribution.
 *      * Neither the name of MIPS Technologies Inc. nor the names of its
 *        contributors may be used to endorse or promote products derived
 *        from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
```

```
* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/
```