StellarisWare Quickstart Guide

Sourcery G++ Application Note AN004

March 5, 2011

Abstract

This application note provides a step-by-step introduction to building and running StellarisWare with Sourcery G++. It covers the complete process beginning with Sourcery G++ installation.

1. Introduction

Sourcery G++ provides a full-featured programming and debugging environment for TI Stellaris boards. This application note provides a quick introduction to installing and using Sourcery G++ with a focus on getting the StellarisWare examples running on your board. This is a set of programs and libraries that demonstrate the use of peripherals on the Stellaris board.

Most of the material in this application note is covered in greater detail in the *Getting Started* guide included with Sourcery G++, which is a more general introduction to using the product and also covers many other features. However, you may find a more focused checklist useful to get up and running quickly.

2. Requirements

This application note assumes you have a TI Stellaris evaluation board equipped with a built-in ICDI USB debug port and will be installing Sourcery G++ for Stellaris EABI on a Windows host. Most of the instructions are also applicable, with minor changes, if you are using Sourcery G++ for ARM EABI, or installing on a GNU/Linux host system rather than Windows.

Stellaris EABI or ARM EABI?

CodeSourcery sells two different products that support TI Stellaris boards: Sourcery G++ for Stellaris EABI and Sourcery G++ for ARM EABI. The Stellaris EABI toolchain supports *only* TI Stellaris targets, while the ARM EABI product also supports many other targets and includes a simulator as well as support for third-party JTAG devices. For more information about the differences between the two products, refer to the Sourcery G++ Knowledge Base[1].

The instructions in this application note are applicable to either product. However, be aware that the Stellaris EABI and ARM EABI toolchains are sold and licensed separately; you must install the toolchain that matches your subscription.

3. Installing Sourcery G++

In this section, we'll cover the steps required to install Sourcery G++ on your computer and prepare it for use. This includes obtaining and installing a license for your product.

3.1. Register with the Sourcery G++ Support Portal

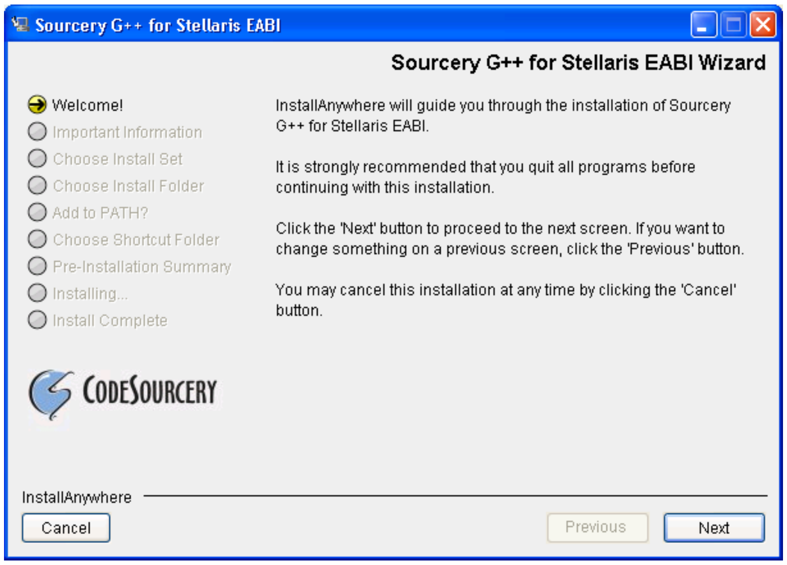
Before you can use Sourcery G++, you must register with the Sourcery G++ Portal[2] and either purchase a subscription or request a free 30-day evaluation. This is necessary to activate your product and obtain a license key.

Once you have registered for Sourcery G++, you can download an installer package from the Support Portal. Even if you received an installer on CD, obtaining the most recent version from the Portal instead is recommended, since newer versions may include support for additional Stellaris boards, as well as improvements to the IDE and general bug fixes.

[1] https://support.codesourcery.com/GNUToolchain/kbentry20
[2] https://support.codesourcery.com/GNUToolchain/

3.2. Install Sourcery G++ on your host computer

The Sourcery G++ 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. You can accept the defaults for a standard install, although you may want to customize the install directory or shortcut configuration. When the installation is complete, it offers to launch the Sourcery G++ IDE and the *Getting Started* guide.


Sourcery G++ for Stellaris EABI

Sourcery G++ for Stellaris EABI Wizard

InstallAnywhere will guide you through the installation of Sourcery G++ for Stellaris EABI.

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

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

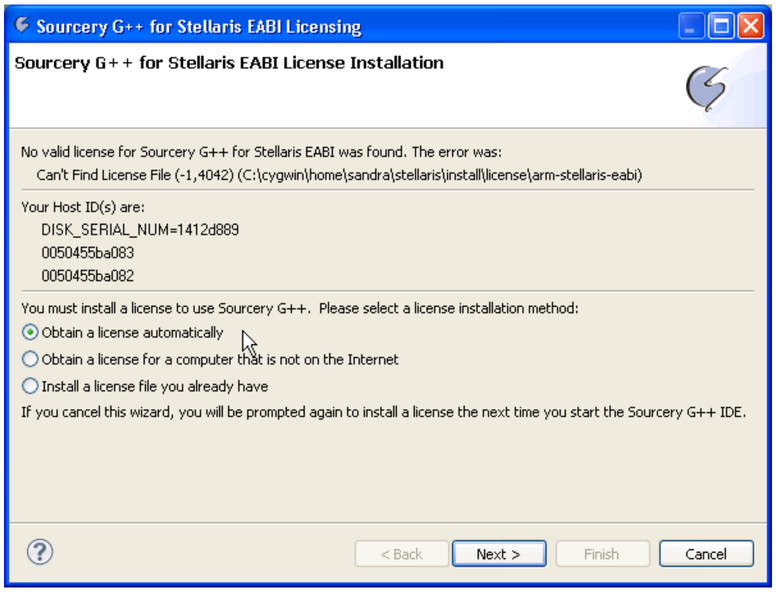
You may cancel this installation at any time by clicking the 'Cancel' button.

Installing Sourcery G++. Follow the step-by-step installer dialogs.

3.3. Install a License Key

When the Sourcery G++ IDE starts, it pops up a dialog asking you to select a workspace directory, which is used as the default location for all your project source and object files as well as for project configuration files. You can use the default or select some other directory, as you prefer.

Next the IDE notices that you do not have a license installed yet, and pops up the Licensing wizard. Choose the `Obtain a license automatically` option. If your computer has multiple host IDs, choose the default. Then enter your username and password for the Sourcery G++ Support Portal.


Sourcery G++ for Stellaris EABI Licensing
Sourcery G++ for Stellaris EABI License Installation

No valid license for Sourcery G++ for Stellaris EABI was found. The error was:
Can't Find License File (-1,4042) (C:\cygwin\home\sandra\stellaris\install\license\arm-stellaris-eabi)

Your Host ID(s) are:
DISK_SERIAL_NUM=1412d889
0050455ba083
0050455ba082

You must install a license to use Sourcery G++. Please select a license installation method:
[✓ Obtain a license automatically]
Obtain a license for a computer that is not on the Internet
Install a license file you already have

If you cancel this wizard, you will be prompted again to install a license the next time you start the Sourcery G++ IDE.

Sourcery G++ Licensing wizard. Choose the `Obtain a license automatically` option to get a license using your Sourcery G++ Support Portal login.

The Licensing wizard uses your Portal login to verify that you have a valid subscription or evaluation for the Sourcery G++ product you are using, and downloads and installs a license key.

Problems?

Refer to the *Getting Started* guide if you need to get a license for a computer that is not on the Internet or is behind a firewall, or for help with other methods of obtaining and installing your license key.

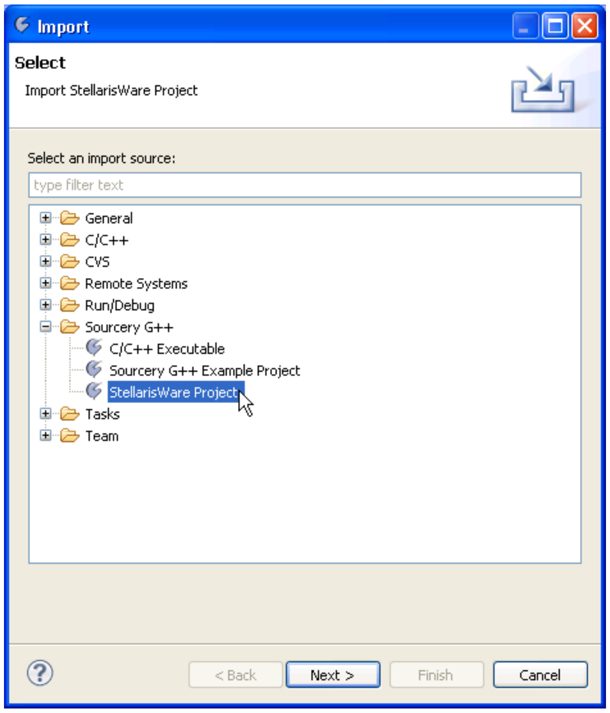
4. Getting StellarisWare Running

Now Sourcery G++ is installed and ready for use. To start, close the welcome screen tab to bring up the C/C++ perspective in the Sourcery G++ IDE. The left-hand pane is the Project Explorer, which lets you navigate around in the projects and files in your workspace. The larger pane to the right is used for editing source code and the lower pane displays console output and build status messages. The IDE has other perspectives, too, that are customized for other tasks; later we'll use the Debug perspective.

4.1. Import and Build StellarisWare

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 TI and is bundled with Sourcery G++ in such a way that you can import StellarisWare projects directly into 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. Use the `Choose` button to open the file selection dialog.



Import wizard. Choose the `StellarisWare Project` option.

The examples directory is organized by board, and then by project. For this application note, we'll use the `blinky` example on a EK-LM3S9B92 Development Kit; so navigate to the `ek-lm3s9b92/blinky` directory and open the file `blinky.sgxx`. Then click `Finish` to import the example.

StellarisWare projects build automatically when you import them into your workspace. Note that importing a StellarisWare project also automatically imports any libraries and shared header files required to build it. Output from the builder appears in the `Console` tab at the bottom of the screen.

4.2. Configure the Stellaris ARMUSB Debug Device

Before you can run code on your Stellaris board using Sourcery G++, you must install the FTDI drivers for the ARMUSB debug interface. The drivers are located in the directory `Tools\FTDI` on the CD provided with your Stellaris board. You can also obtain the latest Stellaris FTDI drivers from TI's web site[3].

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.

Using a Linux Host?

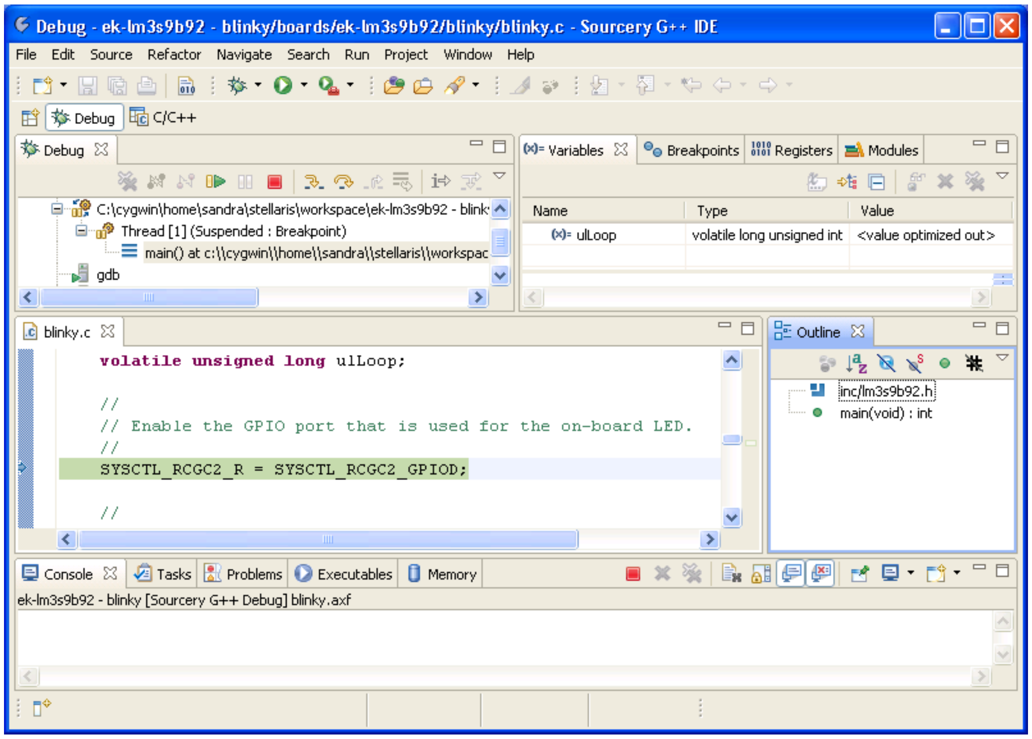
It is not necessary to install drivers to use the Stellaris USB debug interface on a Linux host system, but other setup may be required. Refer to the Debug Sprite chapter of the *Getting Started* guide for details.

[3] http://focus.ti.com/docs/toolsw/folders/print/lm_ftdi_driver.html

4.3. Run an Example Program from the Debugger

Each imported StellarisWare project includes a predefined debug launch configuration that uses the Sourcery G++ Debug Sprite to control the board via its built-in ICDI ARMUSB debug interface.

To run the `blinky` example from the debugger, click on the project name in the Project Explorer view in the left-hand pane. Then select `Run` → `Debug` from the top menu. The IDE then pops up a dialog asking you to confirm that you want to switch to the Debug perspective. Click on the `Yes` button to continue.


Debug - ek-lm3s9b92 - blinky/boards/ek-lm3s9b92/blinky/blinky.c - Sourcery G++ IDE
File Edit Source Refactor Navigate Search Run Project Window Help
Debug
C/C++
Debug
Variables
Breakpoints
Registers
Modules
blinky.c
Outline
Console
Tasks
Problems
Executables
Memory

```
C:\cygwin\home\sandra\stellaris\workspace\ek-lm3s9b92 - blink
  Thread [1] (Suspended : Breakpoint)
    main() at c:\\cygwin\\home\\sandra\\stellaris\\workspac
gdb
```

Name	Type	Value
ulLoop	volatile long unsigned int	<value optimized out>

```
volatile unsigned long ulLoop;

//
// Enable the GPIO port that is used for the on-board LED.
//
SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOD;

//
```

```
inc/lm3s9b92.h
main(void) : int
```

```
ek-lm3s9b92 - blinky [Sourcery G++ Debug] blinky.axf
```

Debug perspective. When you start the debugger, your program is initially stopped at the beginning of `main`.

In the Debug perspective, the IDE gives you a new set of views and panes. The Debug tab at the upper left shows the call stack and has a row of tool buttons to control debugger actions. Below this is a pane that displays the source code for the function currently being executed, The upper right pane holds views that let you examine registers and variables.

> **Tip**
>
> There are additional views, such as disassembly and memory browsing, available if you select `Window` → `Show View` from the top menu.

By default, when you start the debugger, it stops your program at the beginning of its `main` function. Take some time now to locate and try the important debugger controls:

- Use the `Step Into` or `Step Over` tool buttons to single-step through the program execution.
- You can toggle breakpoints by right-clicking in the margin of the code view next to the line where you want the program to stop.

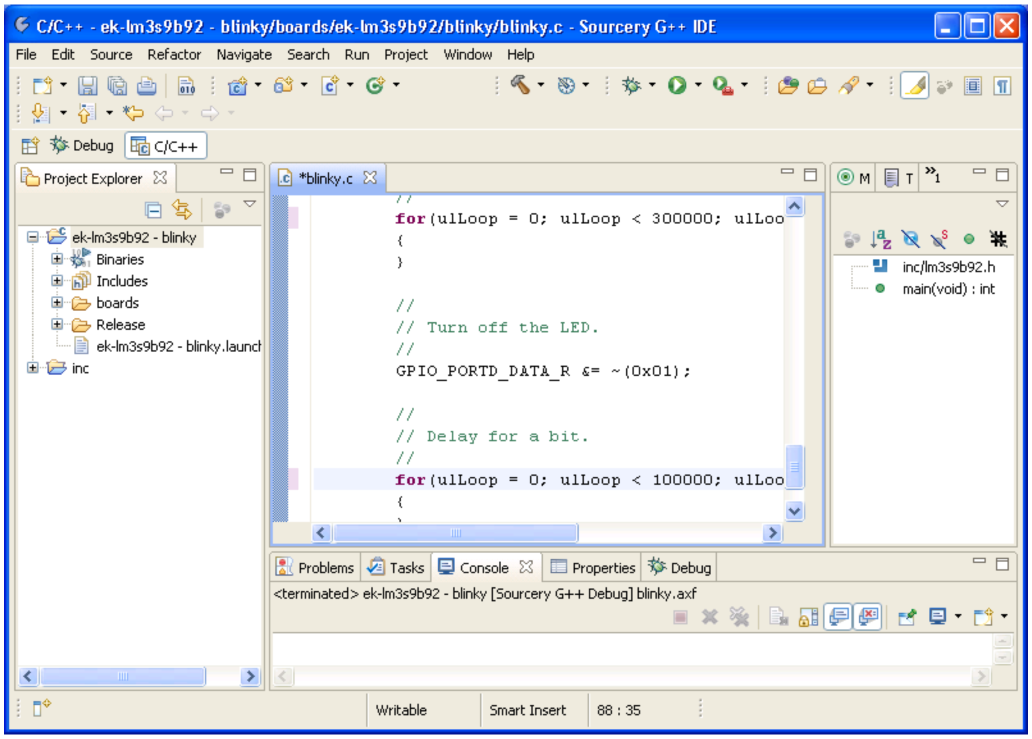
- Use the green `Resume` button to let the program run freely.
- The yellow `Suspend` button stops the running program.
- Use the red `Terminate` button to terminate the debug session.

4.4. Modifying the Example

The `blinky` example is a very simple program that flashes the LED on the Stellaris board on and off. To demonstrate some additional features of the Sourcery G++ IDE, let's modify the program slightly to change the parameters that control the blink timing.

In the Debug perspective, make sure you have terminated your previous debug session. Then switch back to the C/C++ perspective by clicking the button above the Debug tab.

The `blinky.c` source file should already be open in the central editor pane. Find the two busy-wait loops and change the limits `200000` to `300000` in the first loop and `100000` in the second one.



```
//
for(ulLoop = 0; ulLoop < 300000; ulLoo
{
}

//
// Turn off the LED.
//
GPIO_PORTD_DATA_R &= ~(0x01);

//
// Delay for a bit.
//
for(ulLoop = 0; ulLoop < 100000; ulLoo
{
```

```
<terminated> ek-lm3s9b92 - blinky [Sourcery G++ Debug] blinky.axf
```

C/C++ perspective. Use the editor to modify `blinky.c`.

Save the file by using `File` → `Save` from the top menu, or just using the **Ctrl**+**S** keyboard shortcut. Then rebuild the modified program by right-clicking on the `ek-lm3s9b92 - blinky` project in the Project Explorer tab, and selecting `Build Project` from the menu.

Tip

You can rebuild *all* the projects in your workspace using the **Ctrl**+**B** keyboard shortcut.

Now relaunch the debugger, following the same instructions given in the previous section. When you let the program run freely, you should see the LED on the board blinking in a long-short pattern now, instead of the previous pattern of uniform on/off blinks.

5. Next Steps

Now you have Sourcery G++ installed on your host system and have set up your Stellaris board to work with the provided StellarisWare examples. Next you can explore more features of the Sourcery G++ IDE to help you develop and debug your own applications.

The *Getting Started* guide included with your Sourcery G++ installation includes a tutorial that shows you how to create a new C or C++ project from scratch and set up a debug configuration for it. Some additional advanced topics covered in the documentation include:

- Using CS3 and the Sourcery G++ Board Builder to customize the memory map and startup code for your application.
- Using the peripheral register browser from the debugger.
- Using the Stellaris ICDI pass-through feature to debug a production system containing a Stellaris microcontroller.
- For ARM EABI versions of Sourcery G++, information about using the QEMU simulator and additional debug devices such as the SEGGER J-Link.