SiFive Freedom Studio Quick Start Guide

May 22, 2017

SiFive Freedom Studio Quick Start Guide

Release Information

Version	Date	Changes
v1.0	Initial Release	N/A

Contents

SiFive Freedom Studio Quick Start Guide **i**

1 Introduction **1**

1.1 Freedom Studio Introduction 1

1.2 Product Overview 1

1.2.1 Eclipse 1

1.2.2 RISC-V GCC 2

1.2.3 OpenOCD 2

2 Freedom Studio Set Up **3**

2.0.1 Download and Install 3

2.0.2 Hardware Setup 3

2.0.3 Freedom Studio Contents 3

3 Freedom Studio Environment **5**

3.1 Workspace 5

3.2 Perspectives 5

3.2.1 C/C++ Perspective 6

3.2.2 Debug Perspective 7

4 Getting Started **9**

4.1 Import the Bundled Examples 9

4.1.1 Bundled Examples Step by Step 9

4.2 Import Freedom-E-SDK Examples 11

4.2.1 Freedom-E-SDK Examples Step by Step 11

4.3 Creating a New Project From Scratch 12

4.4 Debug 13

5 Troubleshooting **15**

5.1 Build Issues 15

5.2 USB Permission Issues 15

5.3 Mac OS FTDI Driver Issues 15

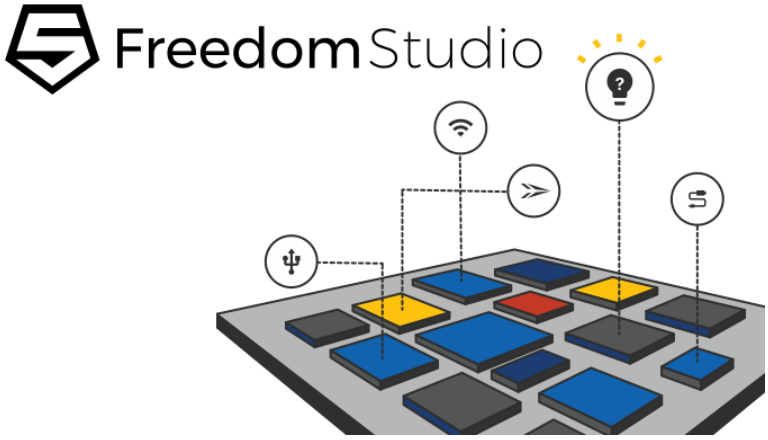
List of Figures

3.1 C/C++ Perspective 6
3.2 Debug Perspective 7

4.1 Import Bundled Examples 10
4.2 Import Freedom-E-SDK Examples 12

Chapter 1

Introduction



1.1 Freedom Studio Introduction

Freedom Studio is an integrated developers environment for SiFive RISC-V based products which can be used to write and debug software targeting SiFive based processors. Freedom Studiois based on the industry standard Eclipse platform and is bundled with a pre-built RISC-V GCC Toolchain, OpenOCD, example programs, and documentation.

1.2 Product Overview

This section will describe the individual component's used in Freedom Studio release BETA.

1.2.1 Eclipse

The major versions of the Eclipse plugins are as follows:

- Eclipse - Neon.2 (4.6.2)
- CDT - 9.2.1201704050430
- egit - 4.4.1.201607150455

1.2.2 RISC-V GCC

RISC-V GCC was built from source using the following repository and commit hash:

- Repository: `https://github.com/riscv/riscv-gcc`
- Commit hash: 4d4068e468d989c1b104df4e97cb46e63706e906

This build of GCC is able to target RV32 and RV64 based processors.

1.2.3 OpenOCD

Freedom Studio uses OpenOCD for hardware debug. OpenOCD was built from source using the following repository and commit hash:

- Repository: `https://github.com/riscv/riscv-openocd`
- Commit hash: 95a2eb157ab0f1569faf17ecb666b99532755136

Chapter 2

Freedom Studio Set Up

2.0.1 Download and Install

Freedom Studio can be downloaded from the SiFive website at the following address: `https://www.sifive.com/products/tools/`

Unzip the download to a directory on your computer. Inside the unzipped directory, there will be a launcher in the root directory called "Freedom Studio". To launch the Freedom Studio environment, either double-click "Freedom Studio" or run it from the command line.

2.0.2 Hardware Setup

This Quick Start Guide assumes that you have already completed the Getting Started guide for your particular hardware, including the OpenOCD setup. If you have not, please refer to the Getting Started guide for your particular hardware and ensure that you are able to connect a serial terminal to your board and upload applications.

Hardware documentation can be found in the "FreedomStudio/SiFive/Documentation" directory and also online at: `https://www.sifive.com/documentation/`.

2.0.3 Freedom Studio Contents

The Freedom Studio directory contents are as follows:

- FreedomStudio - Root directory
  - bin - Directory containing symlinks to all binaries
  - eclipse - Directory containing elipse environment
  - SiFive - SiFive files
    - Documentation - documentation delivered with Freedom Studio.
    - Examples - Zip files containing example projects
    - Misc - Directory containing miscellaneous files such as OpenOCD config files and Linux OpenOCD udev rules
    - OpenOCD - Directory containing the OpenOCD build described in Section 1.2
    - risv64 - Directory containing the RISC-V GCC build described in Section 1.2

Chapter 3

Freedom Studio Environment

3.1 Workspace

Eclipse uses workspaces to group together a set of related projects. Eclipse workspaces allow for a lot of flexibility in how one organizes their projects. For example, it is possible to have a workspace which Contains only a single project. It is also possible to have a workspace which contain multiple related projects such as a library project and an application which depends on that library.

Switching between workspaces is accomplished by selecting ***File – Switch Workspace***.

3.2 Perspectives

Eclipse uses perspectives to group certain windows together which are used for the same tasks. Freedom Studio currently has 3 main perspectives: C/C++ , Debug, and Git. From Eclipse, you can change perspectives by clicking ***Window – Perspectives — Open Perspective***.

Perspectives are fully user customizable and persistent to a workspace. The sections below describe the default perspectives.

3.2.1 C/C++ Perspective

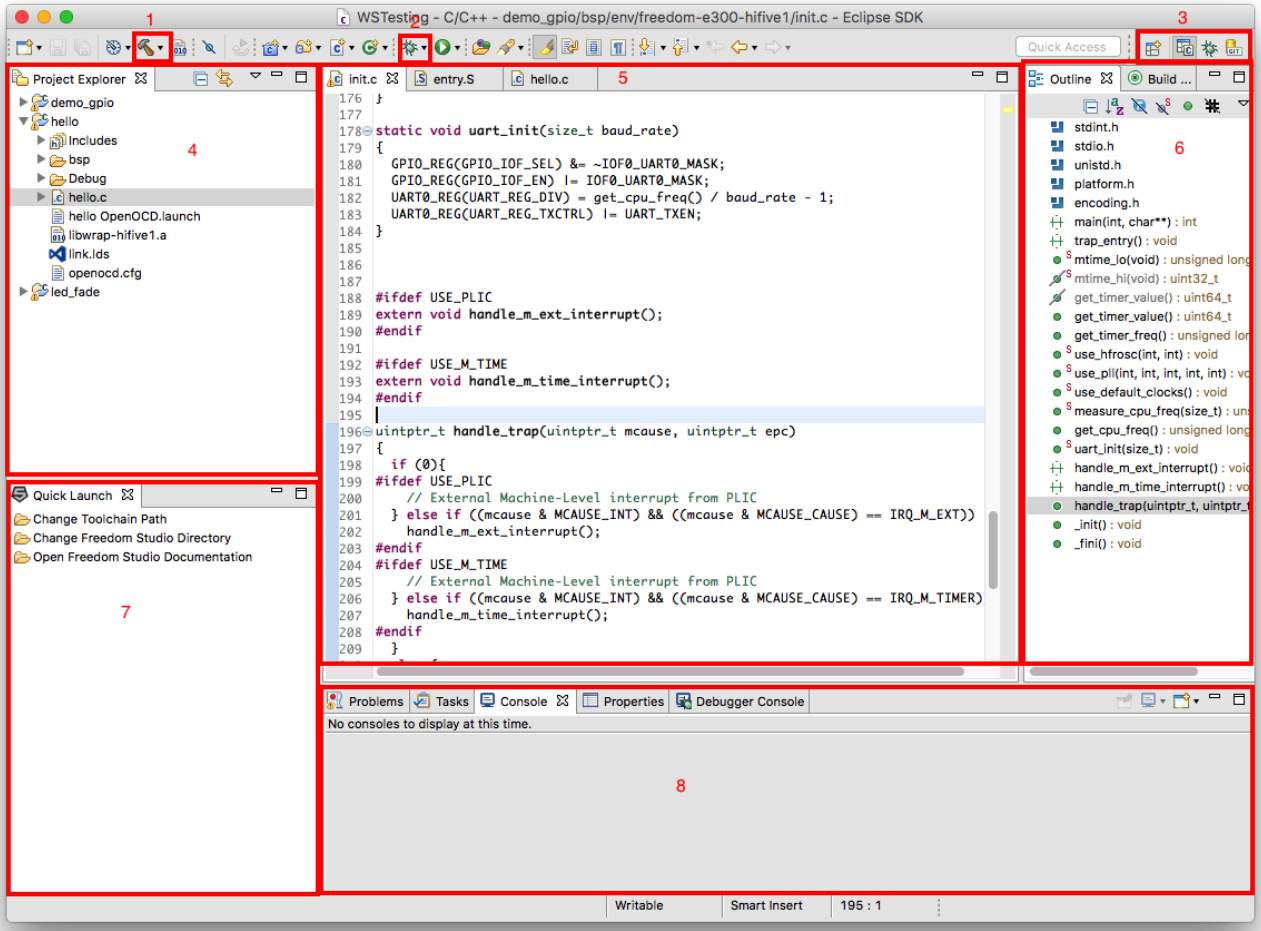


Figure 3.1: C/C++ Perspective

1. Build Toolbar Button.
2. Debug Toolbar Button. The down arrow next to the bug lets you pick a particular configuration.
3. Perspective toolbar - allows you to quickly change perspectives.
4. Project Explorer - Displays all projects in the workspace.
5. Editor - Main view for editing source files.
6. Outline View - Displays the source code Outline of the active editor file.
7. SiFive QuickLaunch - Contains useful quick actions

8. Console - Displays useful information from project builds.

3.2.2 Debug Perspective

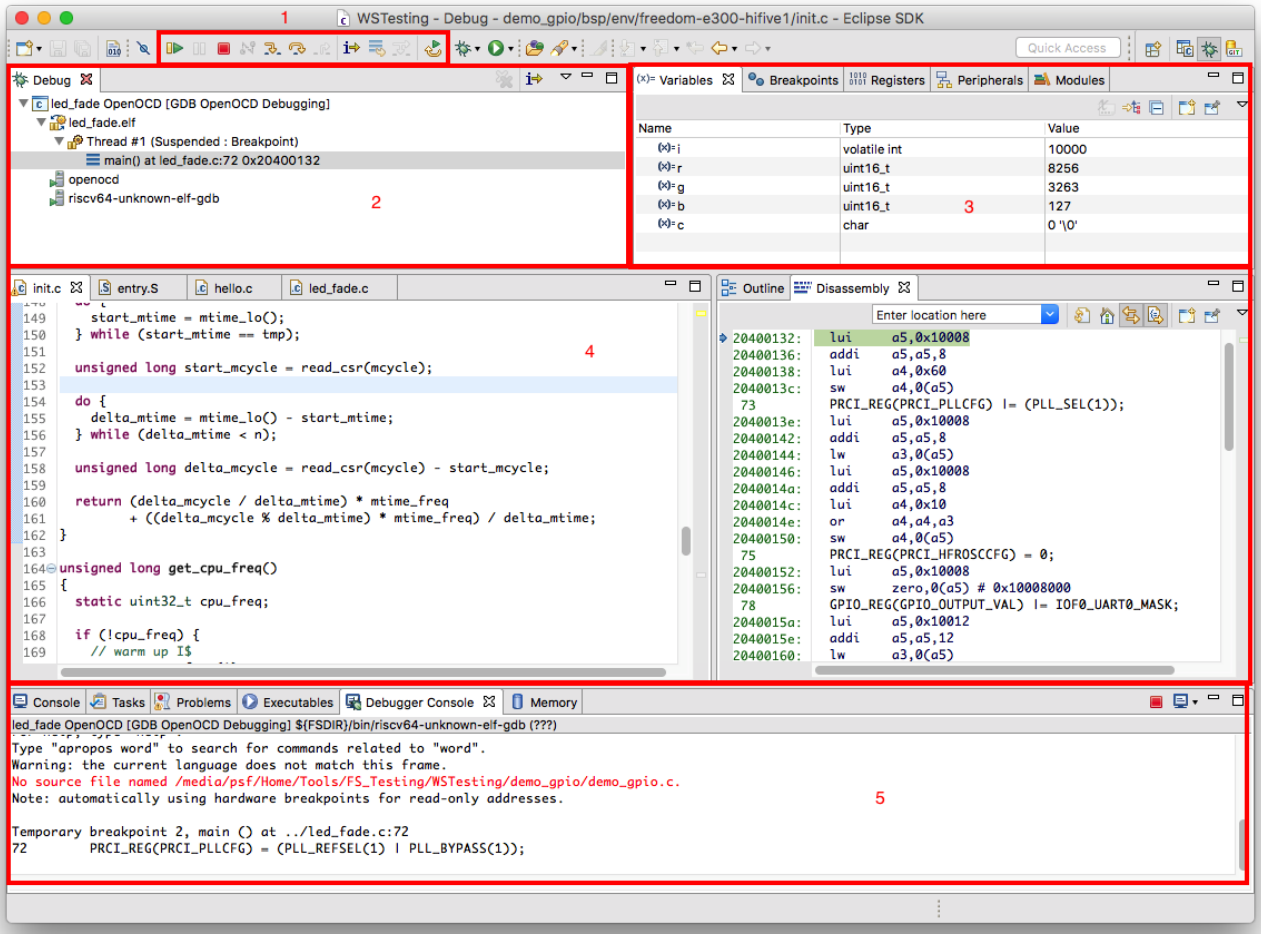


Figure 3.2: Debug Perspective

1. Runtime Controls - (from left to right) Start, Halt, Stop, Step Into, Step Over, Step Out, Step Instruction, Restart.
2. Runtime Context - Displays current context information on a per-thread basis.
3. Debug Views - Variable, Core Registers, Breakpoints.
4. Source View - Displays current PC in source file and in dissasembly.
5. Consoles - Displays GDB and OpenOCD console output.

Chapter 4

Getting Started

4.1 Import the Bundled Examples

Freedom Studio bundles several examples for each SiFive development platform which are suitable starting a new project. These are the same examples from the Freedom-E-SDK but packaged to where each example is self-contained and not reliant on other projects. Specifically this means that relevant files in the *bsp* directory have been copied into the project folder as well as *libwrap.a*, and Include Paths have been changed accordingly.

4.1.1 Bundled Examples Step by Step

After opening Freedom Studio and selecting a workspace (any location is OK), the procedure is as follows:

- From Freedom Studio click ***File – Import***.
- ***General – Existing Projects into Workspace***.
- ***Select Archive File***.
- Browse to your *FreedomStudio/SiFive/Examples* directory and select the zip file for your platform.
- Select as many of the examples as you like, and click ***Finish***.

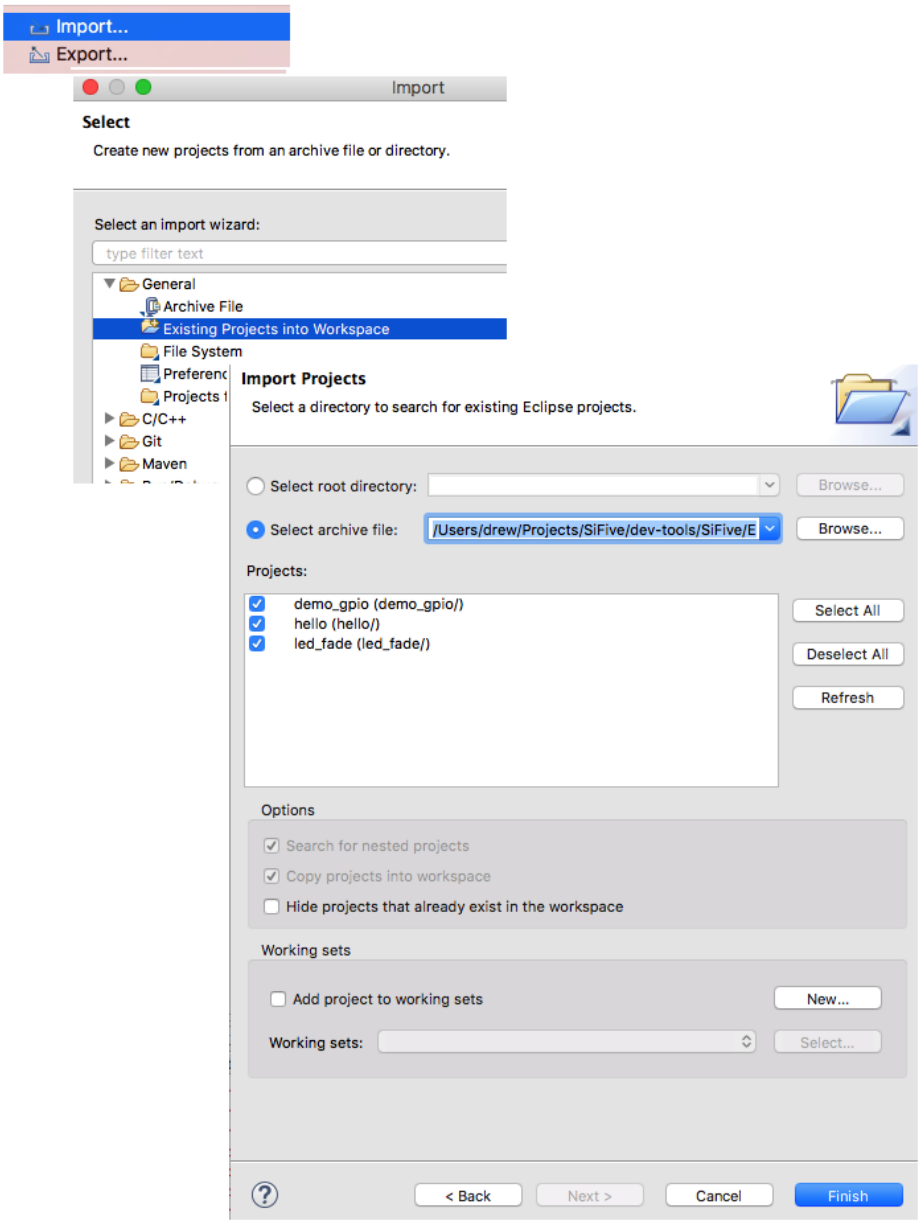


Figure 4.1: Import Bundled Examples

The projects should now be copied into your workspace. Pressing the Build (hammer) icon will build the currently selected project or Control+B (Command+B on macOS) will build the entire workspace.

If switching to a new platform (for example from E31FPGA to E51FPGA), it is reccomended that you also switch to a new workspace.

4.2 Import Freedom-E-SDK Examples

Freedom-E-SDK also includes Freedom Studio example projects in the *Freedom-E-SDK/FreedomStudio* directory. These projects are set up to link to the files in the rest of the Freedom-E-SDK so as the same source files can be built from both Freedom Studio and the classic Freedom-E-SDK makefiles.

Freedom-E-SDK is maintained on github and will therefore always have the latest versions of the example projects compared the bundled examples which are only updated on new releases of Freedom Studio.

4.2.1 Freedom-E-SDK Examples Step by Step

After opening Freedom Studio and selecting a workspace (any location is OK), the procedure is as follows:

- From Freedom Studio click ***File – Import***.
- ***General – Existing Projects into Workspace***.
- ***Select Root Directory***.
- Browse to your *freedom-e-sdk/FreedomStudio* directory and select the folder which matches your platform.
- Select the libwrap project, and as many of the other projects as you like.
- Make sure that “Copy projects into workspace” is not checked, and click ***Finish***.

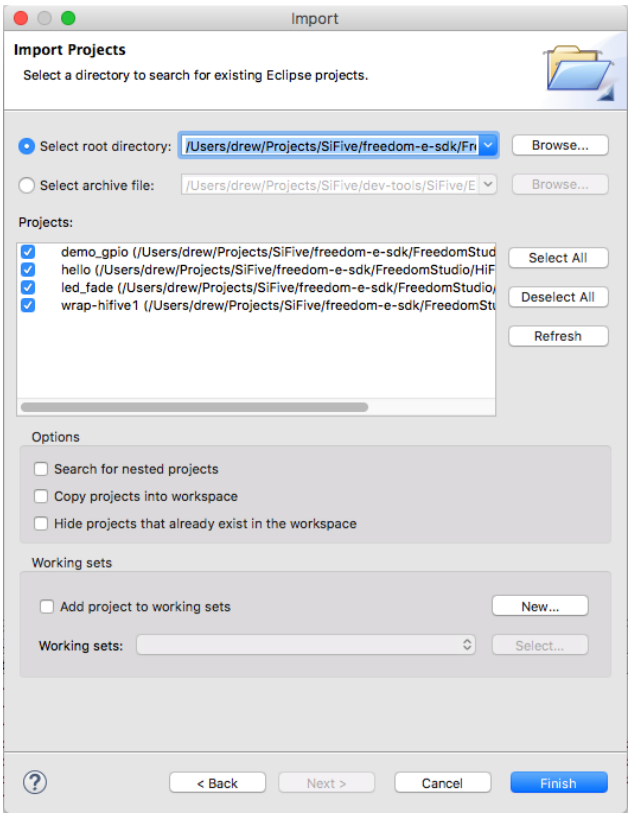


Figure 4.2: Import Freedom-E-SDK Examples

This should result in the selected projects being imported into your workspace. Note that the actual source files for the project remain in the Freedom-E-SDK folder, and any modifications you make to them in Freedom Studio will modify the original Freedom-E-SDK files.

4.3 Creating a New Project From Scratch

It is also possible to create a new project from scratch. This will not be covered in detail here in the Quick Start Guide but the general procedure is as follows:

- After opening Freedom Studio, select ***File – New***.
- Select any of the following Project Types: C, C++, C/C++.
- Under *Project Type* Select RISC-V Embedded Application. If RISC-V is not an option, then un-check the checkbox “Show project types and toolchains only if they are supported on the platform”.
- Give the project a name and select ***Finish***.

From here you will have an empty project which you can begin development. Toolchain settings can be found by right-clicking on the project and selecting ***Properties – C/C++ Build – Settings***.

If switching to a new platform (for example from E31FPGA to E51FPGA), it is required that you also switch to a new workspace.

4.4 Debug

All of the examples include launch files which tell Freedom Studio how to start a debug connection with the target platform. All launch configurations are set to load the program into the address to which they were linked (defined in the linker file), and to halt execution at the *main* function.

To launch a Debug session, click ***Run – Debug Configurations*** and then ***GDB OpenOCD*** followed by the desired program. By default this opens the Debug perspective which contains lots of useful information for debugging. If a particular window of interest is not in the view, click ***Window – Show View*** for the full list of windows.

Some useful debug controls are described in Figure 3.2.

Chapter 5

Troubleshooting

5.1 Build Issues

The current version of Freedom Studio is set to key off of the installation directory to set the toolchain paths correctly. In the current version of Freedom Studio this behavior is not robust and there are many instances when the path will not be set properly by default.

The process to fix is as follows:

- In the C/C++ perspective locate the SiFive Quick Launch View (usually in the bottom-left of environment).
- Click ***Change Toolchain Path*** and select the *FreedomStudio/bin* directory.
- Then click ***Change Freedom Studio Directory*** and select the *FreedomStudio* directory.

5.2 USB Permission Issues

By default, some Linux distributions do not give users permissions to access USB devices. The HiFive1 and FPGA getting started guides describe the process to grant your user the correct permissions. For your convenience the *99-openocd.rules* file is included with Freedom Studio in the *FreedomStudio/SiFive/Misc* directory.

5.3 Mac OS FTDI Driver Issues

There is an issue with Apple's FTDI driver which is used by OpenOCD. This can be fixed from a console using the procedure below:

- Open *Applications/Utilities/Terminal*
- Paste in the following command:
  `sudo kextunload -p -b com.apple.driver.AppleUSBFTDI`
- Paste in the following command:
  `sudo kextutil -b com.apple.driver.AppleUSBFTDI -p AppleUSBEFTDI-6010-1s`

After logging out and logging back into your Mac, it may be necessary to issue the commands above again. It is also possible to add the above commands to your user's *~/.bash_profile*. By doing so, the above commands will be issued every time your user logs in.