SiFive Embedded Software Quick Start Guide

Version 1.2

SiFive Embedded Software Quick Start Guide

Proprietary Notice

Information in this document is provided “as is,” with all faults.

SiFive expressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchantability, fitness for a particular purpose and non-infringement.

SiFive does not assume any liability rising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

SiFive reserves the right to make changes without further notice to any products herein.

Release Information

Version	Date	Changes
1.2	July 9, 2021	• Updated with note about the compact code model
1.1	June 29, 2021	• Updated to describe `CC_CHOICE=clang` option
1.0	February 5, 2021	• Initial release

Contents

1

SiFive Embedded Software Quick Start Guide

1.1 Introduction

All designs generated by SiFive are delivered in a Core IP deliverable that contains the Verilog RTL, simulation testbench, documentation, custom board support package (BSP), and several software examples. This document describes how to begin software development on any SiFive custom core.

1.1.1 Tarball Contents

The IP tarball has a `sifive_coreip` root directory with the following contents:

- Verilog
- Testbench
- Documentation
- Software repo named `freedom-e-sdk` containing a custom BSP that includes:
    - Unique header files, linker scripts, makefiles, and debug configuration files
- Software repository with several examples
- FPGA bitstream for the Arty A7-100T or Xilinx VCU118 FPGA development boards

To extract the IP tarball, simply download the tarball to your workspace and type:

```
% tar xvf ipdelivery.tar.gz
```

1.1.2 Software Examples

Many software examples are pre-compiled and included in the IP deliverable tarball package in a `.hex` file format, located in the `freedom-e-sdk/software/<software-example>/release` path. The `.hex` file can be run in RTL simulation without having to build any software.

The `release` build option has optimizations for size and speed and is typically the build option used for simulation, while the `debug` build option uses fewer optimizations and includes debug symbols.

If your RV64 package does not contain any pre-built hex files, check to see if the software examples require the compact code model option, described in the following SiFive Knowledge Base article: https://sifive.atlassian.net/servicedesk/customer/portal/47/article/1915585520.

Boot Flow

All of the software examples within the `freedom-e-sdk` repo follow the same boot path, provided by the freedom-metal library.

- Entry point is `_enter`, located in `freedom-e-sdk/freedom-metal/src/entry.S`
- Entry calls `_start` in `freedom-e-sdk/freedom-metal/gloss/crt0.s`
- Then, `main()` is called, unless:
    - There is a C-function named `secondary_main()`, which overrides the `secondary_main` in `crt0.s`.

This provides a method on multi-hart designs to redirect different harts to different parts of the software application. Refer to the `freedom-e-sdk/software/multicore-hello` example for more detail.

RTL Simulation

To run a pre-built `.hex` file using Synopsys VCS for example, simply navigate to the IP tarball root folder and type:

```
% make hello.vcs.out
```

Consult the User Guide for more information on RTL simulation options, including Cadence Xcelium, SystemC (Verilator), and others.

FPGA Emulation

To run on an FPGA evaluation platform, like the Arty A7-100T (https://www.xilinx.com/products/boards-and-kits/1-w51quh.html) or the Xilinx VCU118 platform, an `.elf` file is needed which requires the proper software tools to build the examples.

1.2 Tools Setup

Users have the option to select command line build and debug, or use the Freedom Studio Eclipse integrated debug environment (IDE) for software development.

1.2.1 Command Line

To build from the command line, add the following tools to your `$PATH`, which are available at https://www.sifive.com/software.

- GNU Embedded Toolsuite (GCC and LLVM Compilers, Linker, Debugger)
- OpenOCD (Remote GNU Debugger Server for Olimex debug hardware)

To program the Arty A7-100T FPGA platform from the command line, download the `xc3sprog` utility, available at https://github.com/sifive/freedom-tools/releases. This also should be added to your system `$PATH`.

1.2.2 Flash the Bitfile to the FPGA board

To program the Arty A7-100T board from the command line:

```
% xc3sprog -c nexys4 <tarball-path>/arty_a7_100t-sifive/design-arty.bit
```

To program the `.mcs` file, use Freedom Studio or Vivado.

Command Line Build

Navigate to the `freedom-e-sdk` path in your IP tarball to build your software example:

```
% cd /path/to/sifive_coreip/freedom-e-sdk
% make PROGRAM=hello TARGET=design-arty CONFIGURATION=release software
```

The `PROGRAM` parameter specifies the software example and the `TARGET` parameter determines which BSP is used. For RTL simulation, use `CONFIGURATION=release` to optimize for size (using compiler option -Os), and `TARGET=design-rtl`. To debug an application on the Arty A7-100T FPGA board, use `CONFIGURATION=debug` (using compiler options -O0 and -g), and `TARGET=design-arty`. The `release.mk` and `debug.mk` files can be edited directly to change the compiler optimizations.

Note

When building software examples containing vector instructions on a vector-enabled core, it is recommended to add `CC_CHOICE=clang` on the command line.

Type `make help` to see all build and debug options.

To generate a new software project in a different workspace:

```
% make PROGRAM=return-pass TARGET=design-arty \
STANDALONE_DEST=/path/to/desired/location standalone
```

Command Line Debug

To upload and begin debugging a software example on the command line:

```
% make PROGRAM=hello TARGET=design-arty upload
% make PROGRAM=hello TARGET=design-arty debug
```

This will take you to a gdb command prompt that allows full debug control.
Some example GDB commands:

```
(gdb) load
(gdb) b main
(gdb) cont
(gdb) list
(gdb) n
(gdb) s
```

1.2.3 Freedom Studio

The Freedom Studio integrated debug environment (IDE) contains all the tools needed to compile and debug new software examples. Freedom Studio is also available at https://www.sifive.com/software. To launch Freedom Studio:

- **Windows**: Unzip the package and launch FreedomStudio from the root folder. No installer is needed. Make sure there are no spaces in your pathname.
- **macOS**: Double click the `*.tar.gz` file and optionally move `FreedomStudio` to the Applications folder.
- **Linux**: Extract the package and launch the FreedomStudio binary.

There is a built-in wizard that walks through programming a bit file, creating a software example, and launching a debug session. To select this option, use the **Create a new IP Project from IP Deliverable** from the **SiFive Tools** menu. This process will import the compressed IP tarball into the Freedom Studio workspace and walk through all of the steps automatically. The next sections describe each of these options individually.

Flash the Bitfile to the FPGA board

To flash an `.mcs` or `.bit` file to the Arty A7-100T FPGA board, navigate to the **SiFive Tools** menu and select **Program FPGA image to target…**, or select the square chip icon with the letter "A" from the toolbar.

Browse for the file in your IP tarball, located in the `arty_a7_100t-sifive` path. The `.bit` file programs directly to the on-board FPGA, but needs to be reprogrammed if power is removed from the board. This is a good option if multiple different designs are being evaluated on the same board.

The `.mcs` file programming takes longer than the `.bit` file since it gets programmed to flash, but it will get re-imaged to the FPGA from the on-board flash after every power-on reset of the Arty A7-100T board. Programming the `.mcs` is useful when there is only a single design being evaluated, as it retains its state in the on-board flash.

Note

To program a bit file to the VCU118 FPGA board, use Vivado Lab Edition, which is available as a free download. See the SiFive VCU118 FPGA Getting Started Guide for more details.

Create New Freedom Studio Project

Freedom Studio ships with a bundled version of `freedom-e-sdk` but this only contains generic BSP options, which are not meant for creating software examples with custom cores.

To generate a software project for a custom core, launch Freedom Studio and select **Create a New Freedom E SDK Project** from the **SiFive Tools** menu, or select the SiFive logo from the toolbar. Select the `freedom-e-sdk` path within your IP tarball so Freedom Studio will present the proper BSP options for your design. Select the debug connection and select **Finish** to begin building your project.

Freedom Studio Debugging

The output files (`.elf`, `.hex`, `.lst`, `.map`) are located in the `release` or `debug` path under the `src` path in the project workspace.

The **Debug Configurations…** option can be found by right clicking the project and selecting **Debug As…**, or selecting the down arrow next to the bug icon on the toolbar. All of the debug configuration fields will be pre-populated and do not need to be manually configured.

Freedom Studio Manual

For more information, consult the Freedom Studio manual, available from the **Help** menu within the IDE. Also refer to the Application Notes section of https://www.sifive.com/documentation, as well as the Manual and User Guide for the custom core you are working with.