SiFive HiFive1 Rev B Getting Started Guide

Version 1.0

SiFive HiFive1 Rev B Getting Started Guide

Proprietary Notice

Release Information

Version	Date	Changes
1.0	March 19, 2019	Initial Release

Contents

Chapter 1

HiFive1 Rev B Diagram

1.1 HiFive1 Rev B Components

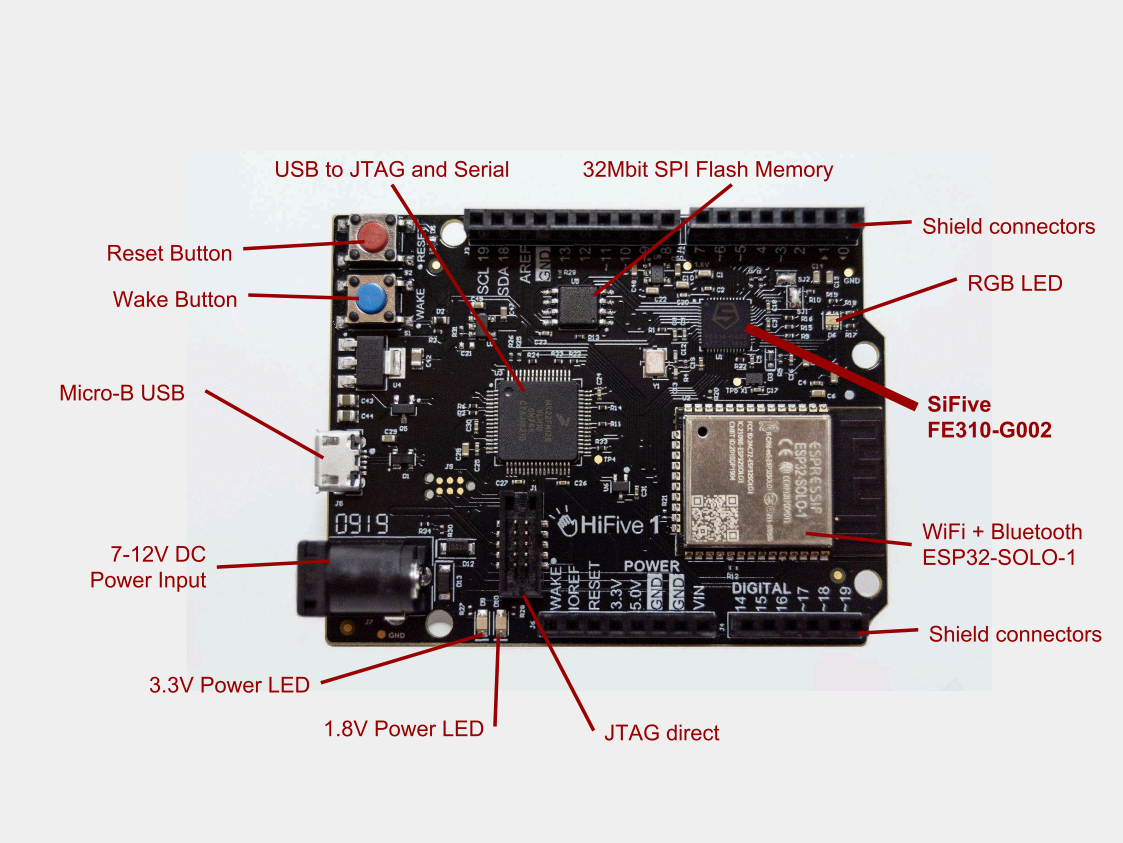

Figure 1: HiFive1 Rev B Board

Figure 1 shows the HiFive1 Rev B with the components which are described in this document.

1.2 HiFive1 Rev B Schematics

Schematics for the HiFive1 Rev B board can be found on SiFive's web site.

- Documentation:

  `https://www.sifive.com/documentation`

Chapter 2

Required Hardware

Using the HiFive1 Rev B requires the following hardware.

2.1 HiFive1 Rev B Board

SiFive's HiFive1 Rev B is a development board for the FE310-G002, a microcontroller with an E31 RISC-V RV32IMAC CPU.

2.2 USB Cable

A standard USB Type A Male to Micro-B Male cable can be used to connect a host system to the HiFive1 Rev B. USB connection is used for power and communication.

- USB cable example:

  `http://store.digilentinc.com/usb-a-to-micro-b-cable/`

Chapter 3

Hardware Features

3.1 SiFive FE310-G002 SoC

The HiFive1 Rev B enables evaluation of the SiFive FE310-G002 System on Chip features. More information on this SiFive FE310-G002 chip can be found at:

- Documentation:

  `https://www.sifive.com/documentation`

3.2 SPI Flash

The HiFive1 Rev B is populated with 32Mbit of Flash memory connected to the SPI interface.

- Datasheet:

  `http://www.issi.com/WW/pdf/25LP-WP032D.pdf`

3.3 USB to JTAG and Serial Ports

The HiFive1 Rev B is populated with a Segger J-Link OB module which bridges USB to JTAG and two serial ports used for FE310-G002 console, FE310-G002 JTAG, and ESP32-SOLO-1 configuration.

`https://www.segger.com/products/debug-probes/j-link/models/j-link-ob/`

- JTAG used for FE310-G002 debug
- Serial 0 used for FE310-G002 console
- Serial 1 used for Espressif Systems ESP32-SOLO-1

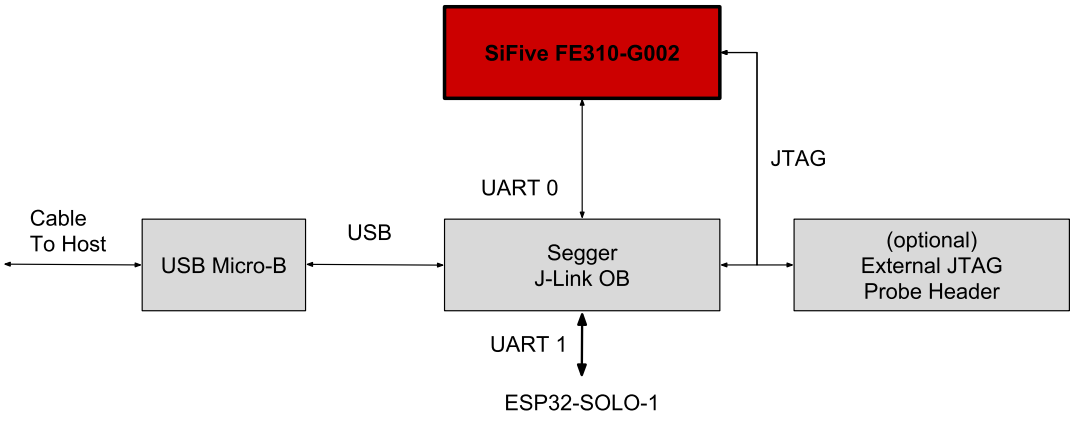

Figure 2: J-Link OB connectivity

Figure 2 shows the connectivity between SiFive FE310-G002 SoC, J-Link OB, and Wireless Connectivitiy on HiFive1 Rev B

3.4 I/O expansion connectors

The HiFive1 Rev B features I/O expansion connectors and the pinout was derived from Arduino®. See the pinout in (Figure 5). Note that:

- Arduino® adapter compatibility has not been verified.
- No analog signal I/O capability and AREF pin is not connected.
- Technical support requests must be directed to SiFive.

3.5 Wireless

The HiFive1 Rev B features an Espressif Systems ESP32-SOLO-1 module for wireless standard Bluetooth, Bluetooth LE, and Wi-Fi 802.11n. The ESP32-SOLO-1 module contains an SoC, flash memory, precision discrete components and a PCB antenna. Note that the flash memory for the ESP32-SOLO-1 module is a separate flash memory from the SPI flash dedicated to the FE310-G002. The serial interface is used to configure the Espressif Systems

ESP32-SOLO-1 flash but note that the ESP32-SOLO-1 is delivered flashed with esp32-at firmware from:

https://github.com/espressif/esp32-at

The ESP32-SOLO-1 radios are disabled on boot by firmware to reduce power consumption.

Note that drivers are expected in the late Q2, 2019 timeframe.

- Datasheet:

https://www.espressif.com/sites/default/files/documentation/esp32-solo-1_datasheet_en.pdf

The Espressif Systems ESP32-SOLO-1 has multiple communication interfaces. It has wireless interfaces 802.11n and Bluetooth. It also has wired SPI and serial. The SPI interface is used for as primary data path between wireless interfaces and the application running on FE310-G002.

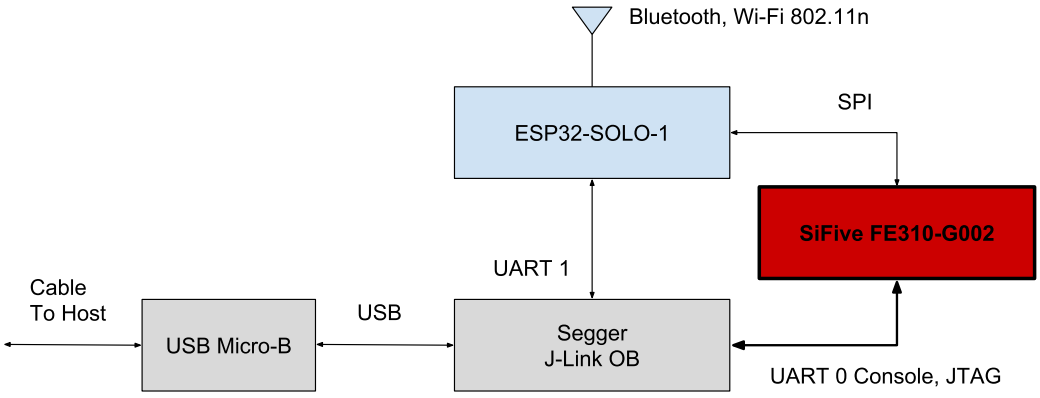

Figure 3: Espressif Systems ESP32-SOLO-1 connectivity

Figure 3 shows the connectivity between SiFive FE310-G002 SoC, Wireless Connectivitiy, and J-Link OB on HiFive1 Rev B.

Chapter 4

Hardware Options

4.1 Power Supply

One option for HiFive1 Rev B power input is USB via host system or USB power supply through the USB Type A to Micro-B cable. This power input method is convenient because the USB interface also features communication capability.

Another option for HiFive1 Rev B power is via power jack J7 where an external 7-12V DC supply or battery can be connected. Note the center positive polarity of the input power jack J7.

Another option for HiFive1 Rev B power is via shield connector VIN pin. The input range is 7-12V DC.

4.2 Compatible Shields

Shields are devices which are designed to fit on the I/O headers on devices which match the Arduino® form factor. They provide additional functionality. Generally, shields which communicate with SPI, UART, and digital I/O should be easy to use with the HiFive1 Rev B, but their supporting software library may need minor tweaks to recognize the HiFive1 Rev B.

Note that HiFive1 Rev B does not feature direct analog signal input capability. Analog inputs will need an adapter such as a shield that supports this capability.

Note that shield IOREF signal for HiFive1 Rev B is fixed at 3.3V.

The shield manufacturer typically provides software libraries.

4.3 External JTAG probe header

The HiFive1 Rev B is populated with a 2x5, 0.05" pitch connector for use with an optional external JTAG probe. This option would replace the on board Segger J-Link OB JTAG function with an external JTAG debugger.

Chapter 5

Board Setup

5.1 Connecting the USB Interface

Connect the USB Type A to Micro-B cable between the USB Micro-B port of the HiFive1 Rev B and the USB Type A of the host machine. This interface provides serial console access to the HiFive1 Rev B, a power source for the board, and is a mechanism to program and debug the FE310-G002

When the USB cable is connected you should see the green power indication LEDs D10 and D9 light up. This indicates that the main 5V supply is active, 3.3V supply is active, and the 1.8V supply is active.

Chapter 6

Debugger and Console

The HiFive1 Rev B comes programmed with a simple bootloader and a demo software program which prints to the console and cycles through the RGB LED in a rainbow pattern. You can input “y” on the console to indidcate that the LEDs are changing and receive a “PASS” message.

This default program will be overwritten in the SPI Flash when you program a new program onto the board but the bootloader code will not be modified. Chapter 7 provides more detail about generating a new program.

The console, program, and debug functions on are implemented with a combination of debugger hardware and software. This combination of debug hardware and software is collectively refered to as a “debugger”. The HiFive1 Rev B uses the Segger J-Link OB debugger. The Segger J-Link OB all in one debugger solution includes GDB server software.

The J-Link OB debugger has the capability to transfer files via the USB mass storage feature. When connected to a host system the HiFive1 Rev B advertises a USB mass storage device and programming files can be conveniently flashed to the HiFive1 Rev B with graphical drag and drop operation.

An optional external debugger can be used but this configuration is beyond the scope of this document.

6.1 Segger J-Link OB Debugger Configuration

The only hardware connection required is the USB Type A Male to Micro-B Male cable from the host machine to the HiFive1 Rev B

The Freedom Studio development download bundles debugger software such that there is no need to configure the software separately. If Freedom Studio detects that J-Link OB software is installed it will attempt to use this install otherwise it will use the bundled software. If you’re using Freedom Studio you can skip this section.

The Freedom E SDK development environment requires J-Link OB software to be installed separately on the host machine software for programming and debugging on the HiFive1 Rev B.

The J-Link software that is installed is a function of the host machine Operating System.

- Select the host OS type and download documentation pack from:

https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack

6.1.1 Ubuntu install example

Download the JLink_Linux_V644b_x86_64.deb file from https://www.segger.com/downloads/jlink/JLink_Linux_x86_64.deb

After downloading the file, run the command below to install the software:

```
> sudo dpkg -i ~/Downloads/JLink_Linux_V644b_x86_64.deb
```

Symbolic links to executables are installed in /usr/bin directory.

6.1.2 Windows install example

Download and run the JLink_Windows.exe file from https://www.segger.com/downloads/jlink/JLink_Windows.exe

6.2 Console Configuration

After configuring the J-Link OB Debugger and connecting the HiFive1 Rev B to the host machine with a USB Type A Male to Micro-B Male cable, the HiFive1 Rev B console can be accessed from the host machine.

6.2.1 Ubuntu Console Example

The debugger will present two ttyACM devices and a USB storage device to the operating system. The operating system will label these devices something like /dev/ttyACM0, /dev/ttyACM1, and /dev/sdb.

Using a terminal emulator such as GNU screen on Linux, open a console connection from the host machine to the HiFive1 Rev B.

Set the following parameters:

Speed	115200
Parity	None
Data bits	8
Stop bits	1
Hardware Flow	None

For example, on Linux using GNU Screen:

```
> sudo screen /dev/ttyACM0 115200
```

You can use *Ctrl-a k* sequence to “kill” (exit) the running screen session.

Depending on the host setup, you may need additional drivers or permissions to access the USB port.

Below is an example of the steps you may need to follow if you are running on Ubuntu-style Linux to access a USB port based console without `sudo` permissions:

1. With the board’s debug interface connected, verify the HiFive1 Rev B shows up with the `lsusb` command:

   ```
   > lsusb
     ...
   > Bus XXX Device XXX: ID 1366:1051 SEGGER
     ...
   ```

2. Set the udev rules to allow the device to be accessed by the `plugdev` group:

   ```
   > sudo vi /etc/udev/rules.d/99-jlink.rules
   ```

   Add the below “SUBSYSTEM==…” line directly above the last line of the file “LABEL="jlink_rules_end"” such that it looks like:

   ```
   SUBSYSTEM=="tty", ATTRS{idVendor}=="1366", ATTRS{idProduct}=="1051",
MODE="664", GROUP="plugdev"
   LABEL="jlink_rules_end"
   ```

3. See if the board shows up as a serial device belonging to the `plugdev` group:

   ```
   > ls /dev/ttyACM*
     /dev/ttyACM0  /dev/ttyACM1
   ```

   (If you have other serial devices or multiple boards attached, you may have more devices listed). For serial communication with the console UART, you will always want to select the first number of the two, in this example `/dev/ttyACM0`.

   ```
   > ls -l /dev/ttyACM0
     crw-rw-r-- 1 root plugdev 166, 0 Mar 19 20:30 /dev/ttyACM0
   ```

4. Add the user name to the `plugdev` group to eliminate need to sudo for access to the device. You can use the `whoami` command to determine the user name.

```
> whoami
  <user_name>
> sudo usermod -a -G plugdev <user_name>
```

5. Log out and log back in, then check that you're now a member of the plugdev group:

```
> groups
  ... plugdev ...
```

Now you should be able to access the serial console (UART) and debug interface without sudo permissions.

6.3 Console I/O

If you have the serial console setup correctly, below is what you will see on the console on a factory programmed HiFive1 Rev B and note that you may need to hit the 'Reset' button to restart the program and see the output:

```
                SIFIVE, INC.

          55555555555555555555555
        5555                    5555
       5555                      5555
      5555                        5555
     5555       555555555555555555555
    5555       5555555555555555555555
   5555                               5555
  5555                                 5555
 5555                                   5555
555555555555555555555555555           55555
 55555           555555555           55555
   55555           55555           55555
     55555           5           55555
       55555                   55555
         55555               55555
           55555           55555
             55555       55555
               55555   55555
                 555555555
                   55555
                     5

              'led_fade' Demo

5555555555555555555555555555555555555555555555555
5555555 Are the LEDs Changing? [y/n]  555555555
5555555555555555555555555555555555555555555555555
y
PASS
```

6.4 Drag and Drop Flashing Operation

The HiFive1 Rev B appears as a flash drive to the OS. Using an OS GUI a .hex file can be transferred to the HiFive1 Rev B flash memory with drag and drop operation.

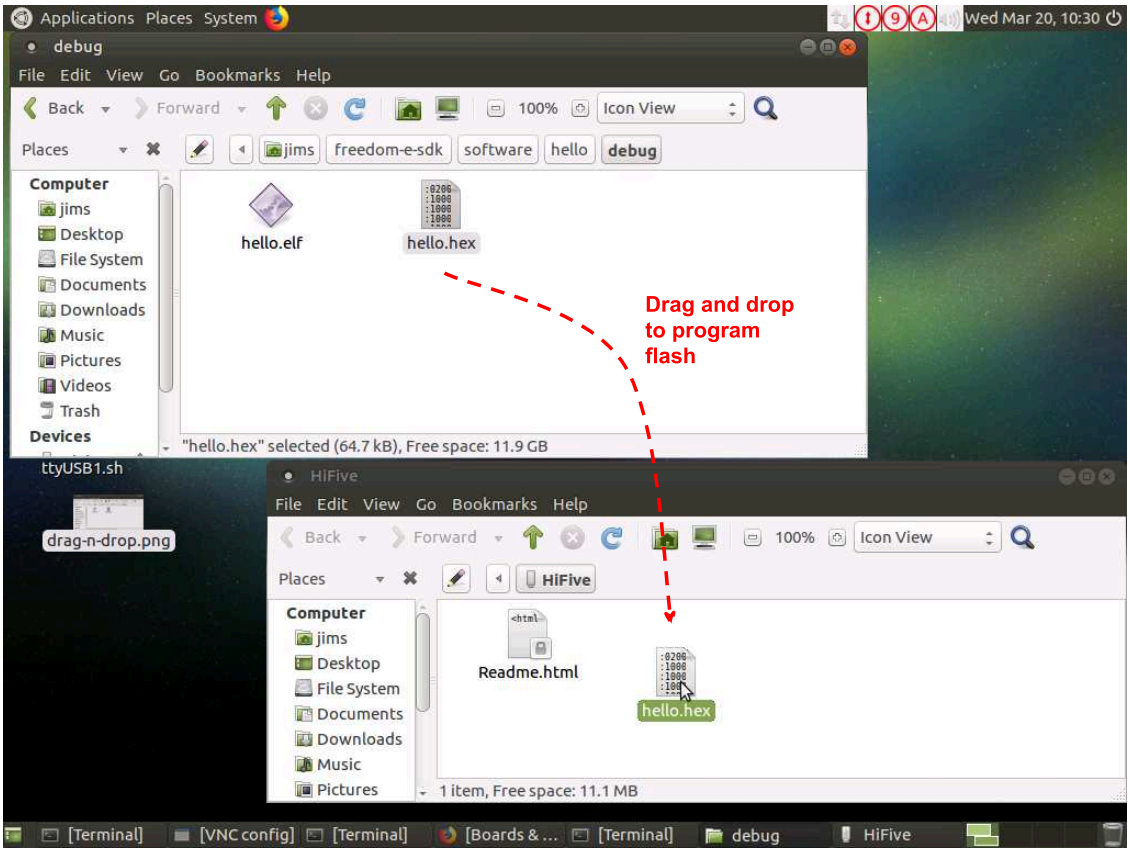

Figure 4: HiFive1 Rev B Drag and Drop Flash Memory Feature

Figure 4 shows a flashing operation via Linux GUI.

Chapter 7

SiFive Software Development Flow

The FE310-G002's boot code contains a jump to the external SPI Flash memory on the HiFive1 Rev B, at address 0x20010000. You can change the program which the FE310-G002 executes by using the debug/programming interface to flash a new compiled program into the external SPI Flash.

SiFive supports multiple software development flows.

- Freedom Studio (Section 7.2) is an Eclipse-Based IDE which bundles everything you need for development into one download.
- Freedom E SDK (Section 7.3) enables you to compile source code with command line tools.

These software development flows will install the same set of tools, but the versions, install paths, and associated software libraries and examples are different for each development flow.

Note that there is no support for HiFive1 Rev B from Arduino® IDE.

7.1 Supported Development Platforms

Freedom Studio IDE is supported on Linux, macOS, and Windows.

Freedom E SDK is supported only on Linux and macOS.

7.2 Software Development Using Freedom Studio IDE

SiFive recommends software development for the HiFive1 Rev B with the Eclipse-based Freedom Studio IDE. Freedom Studio IDE is supported for Windows, macOS, and Linux operating systems. When using this method of software development, the precompiled tools and drivers are automatically installed, you do not need to download or install them seperately to get tools and example code.

- You can obtain Freedom Studio IDE from the SiFive website:

```
https://www.sifive.com/boards
```

- More information on how to use Freedom Studio IDE can be found in the manual at:

```
https://www.sifive.com/documentation
```

7.3 Software Development Using Freedom E SDK Command Line Tools

Freedom-E-SDK is a public github repository, maintained by SiFive Inc, that makes it easy to get started developing software for SiFive's Freedom and RISC-V Core IP Platforms. The SDK supports a wide array of SiFive Core Complexes, SoCs and Emulation environments. This section describes how to setup the toolchain and configure the SDK. The section also walks through building an example program and executing it in the RTL testbench included in a SiFive Core IP devlierable. In addition, the section will walk through how to import custom BSPs and build a program using the custom BSP target.

This section assumes the SDK will be used on a Linux or Linux like environments.

7.3.1 Setting Up Freedom E SDK

Prerequisites

To use this SDK, you will need the following software available on the host machine:

```
GNU Make
Git
```

Toolchain Prerequisites

To build programs and debug, you will need the following toolchains installed on the host machine:

```
RISC-V GNU Toolchain
Segger J-Link OB debugger
```

Toolchains can be found on the SiFive Website, https://www.sifive.com/boards. The pre-built tools have been carefully packaged to support both RISC-V 32-bit & 64-bit ISAs and work on Linux, macOS, and Windows hosts.

Download the toolchain for the platform, and untar it to the desired location. Then, set the RISCV_PATH environment variable so the Freedom-E-SDK can locate the tools. For example,

```
> cp riscv64-unknown-elf-gcc-<date>-<platform>.tar.gz <desired untar dir>
> cd <desired untar dir>
> tar -xvf riscv64-unknown-elf-gcc-<date>-<platform>.tar.gz
> export RISCV_PATH=<desired untar dir>/riscv64-unknown-elf-gcc-<date>-<version>
```

See section Section 6.1 for installation of Segger J-Link OB debugger

Cloning the Freedom-E-SDK Repository

The Freedom-E-SDK repository can be cloned by running the following commands:

```
> git clone --recursive https://github.com/sifive/freedom-e-sdk.git
> cd freedom-e-sdk
```

The --recursive option is required to clone the git submodules included in the repository. If at first you omit the --recursive option, you can achieve the same effect by updating submodules using the command:

```
> git submodule update --init --recursive
```

For more info: https://github.com/sifive/freedom-e-sdk

7.3.2 Freedom Metal

Freedom Metal is a library developed by SiFive for writing portable software for all of SiFive's RISC-V IP, RISC-V FPGA evaluation images, and development boards. Programs written against the Freedom Metal API are intended to build and run for all SiFive RISC-V targets. This makes Freedom Metal suitable for writing portable tests, bare metal application programming, and as a hardware abstraction layer for porting operating systems to RISC-V.

Consumers of Freedom-E-SDK should also be aware that we are still making refinements to the API provided by Freedom Metal. As such, the Freedom Metal API should be considered in beta until we tag a stable release of Freedom E SDK.

- Freedom Metal repository:

  `https://github.com/sifive/freedom-metal`

- Freedom Metal API Documentation:

  `https://sifive.github.io/freedom-metal-docs/index.html`

7.3.3 Freedom E SDK Standard BSP

The Freedom Metal Compatibility Library layer uses the board support package files to provide the hardware abstraction layer. These bsp files can be found under the bsp folder in Freedom-E-SDK and are encapsulated entirely within each target directory.

The HiFive1 Rev B support files for Freedom Metal are entirely within bsp/sifive-hifive1b/ and consist of the following:

design.dts

The DeviceTree description of the target. This file is used to parameterize the Freedom Metal library to the target device. It is included as a reference so that users of Freedom Metal are aware of what features and peripherals are available on the target.

metal.h

The Freedom Metal machine header which is used internally to Freedom Metal to instantiate structures to support the target device.

metal.lds

The linker script for the target device.

settings.mk

Used to set -march and -mabi arguments to the RISC-V GNU Toolchain.

7.3.4 Example Programs

Some example programs can be found in the software directory:

hello

Prints "Hello, World!" to stdout, if a serial device is present on the target.

return-pass

Returns status code 0 indicating program success.

return-fail

Returns status code 1 indicating program failure.

example-itim

Demonstrates how to statically link application code into the Instruction Tightly Integrated Memory (ITIM) if an ITIM is present on the target.

software-interrupt

Demonstrates how to register a handler for and trigger a software interrupt.

timer-interrupt

Demonstrates how to register a handler for and trigger a timer interrupt.

local-interrupt

Demonstrates how to register a handler for and trigger a local interrupt.

example-pmp

Demonstrates how to configure a Physical Memory Protection (PMP) region.

example-spi

Demonstrates how to configure, read, and write to SPI bus.

7.3.5 Updating the SDK

To update the SDK to the latest version:

```
> git pull origin master
> git submodule update --init --recursive
```

7.3.6 Using the Freedom E SDK

Building an Example

To compile a bare-metal RISC-V program:

```
> make BSP=metal [PROGRAM=hello] [TARGET=sifive-hifive1-revb] software
```

The square brackets in the above command indicate optional parameters for the make invocation. The default values of these parameters tell the build script to build the hello example for the sifive-hifive1 target. If, for example, you wished to build the timer-interrupt example, you would instead run the command

```
> make BSP=metal PROGRAM=timer-interrupt TARGET=sifive-hifive1-revb software
```

Uploading to the Target Board

```
> make BSP=metal [PROGRAM=hello] [TARGET=sifive-hifive1-revb] upload
```

Debugging a Target Program

```
> make BSP=metal [PROGRAM=hello] [TARGET=sifive-hifive1-revb] debug
```

Cleaning a Target Program Build Directory

```
> make BSP=metal [PROGRAM=hello] [TARGET=sifive-hifive1-revb] clean
```

Operations can be sequenced: clean then build then upload

```
    > make BSP=metal [PROGRAM=hello] [TARGET=sifive-hifive1-revb] clean software
upload
```

Debugging tips. View program addresses, sections, opcodes, and source

```
    > $RISCV_PATH/bin/riscv64-unknown-elf-objdump -dS software/hello/debug/hello.elf
|less

    > $RISCV_PATH/bin/riscv64-unknown-elf-readelf -a software/hello/debug/hello.elf
|less
```

Create a Standalone Project

You can export a program to a standalone project directory using the standalone target. The resulting project will be locked to a specific TARGET.

STANDALONE_DEST is a required argument to provide the desired project location.

```
    > make standalone BSP=metal [PROGRAM=hello] [TARGET=sifive-hifive1-revb]
STANDALONE_DEST=<desired standalone dir>
```

Once the standalone project is created, it can be compiled simply by typing make.

```
    > cd <desired standalone dir>
    > make
```

For more make options run:

```
    > make help
```

Chapter 8

Zephyr RTOS

For support of the Zephyr Real Time Operating System for HiFive1 Rev B, visit the following url:

https://github.com/sifive/riscv-zephyr/tree/hifive1-revb

Chapter 9

For More Information

Additional information, the latest version of this guide, and supporting files can be found at https://www.sifive.com.

Questions are best answered on the SiFive Forums at http://forums.sifive.com.

More information about RISC-V in general is available at http://riscv.org.

SiFive thoughts, ideas, and news at https://www.sifive.com/blog/.

Webinars at https://info.sifive.com/risc-v-webinar.

9.1 Debug Reference Info:

https://gnu-mcu-eclipse.github.io/debug/jlink/install/.

https://www.segger.com/downloads/jlink/UM08001.

https://www.gnu.org/software/gdb/documentation/.

https://github.com/riscv/riscv-debug-spec.

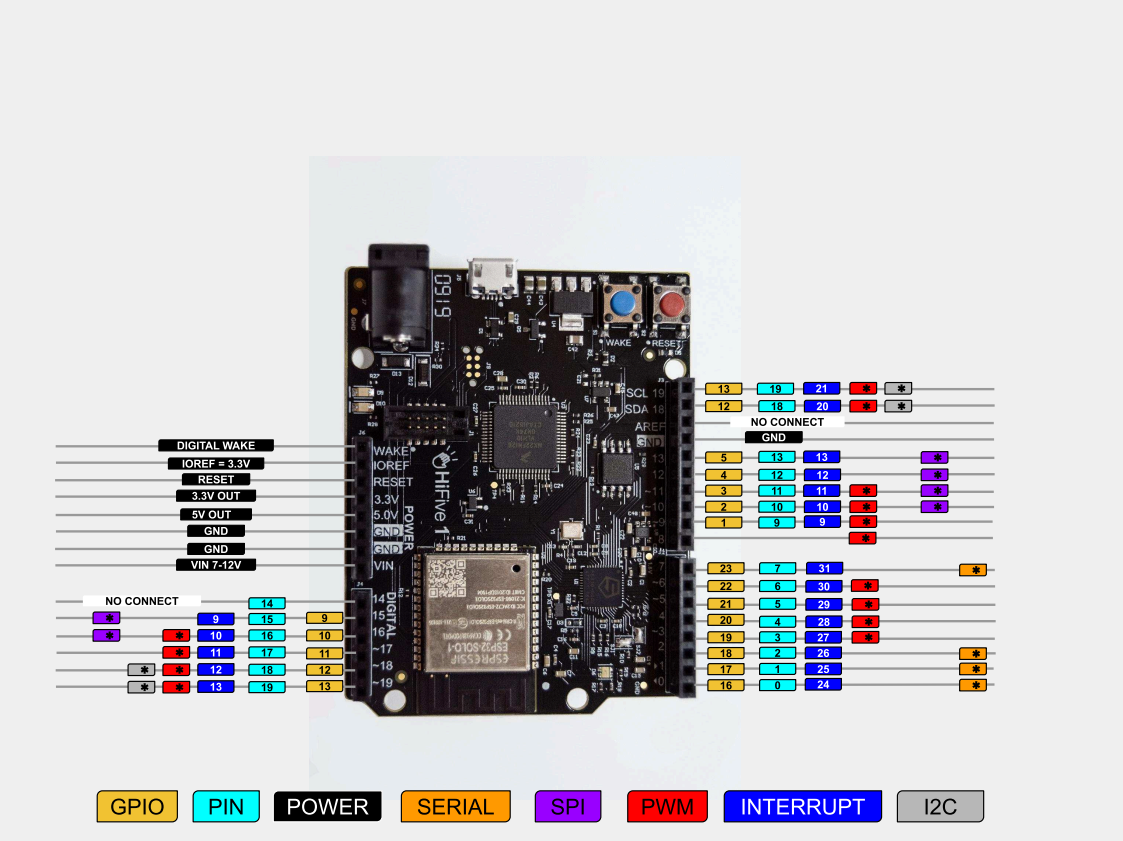

Figure 5: HiFive1 Rev B Pinout

Figure 5 shows the HiFive1 Rev B shield pinout pin labeling and muxing.