

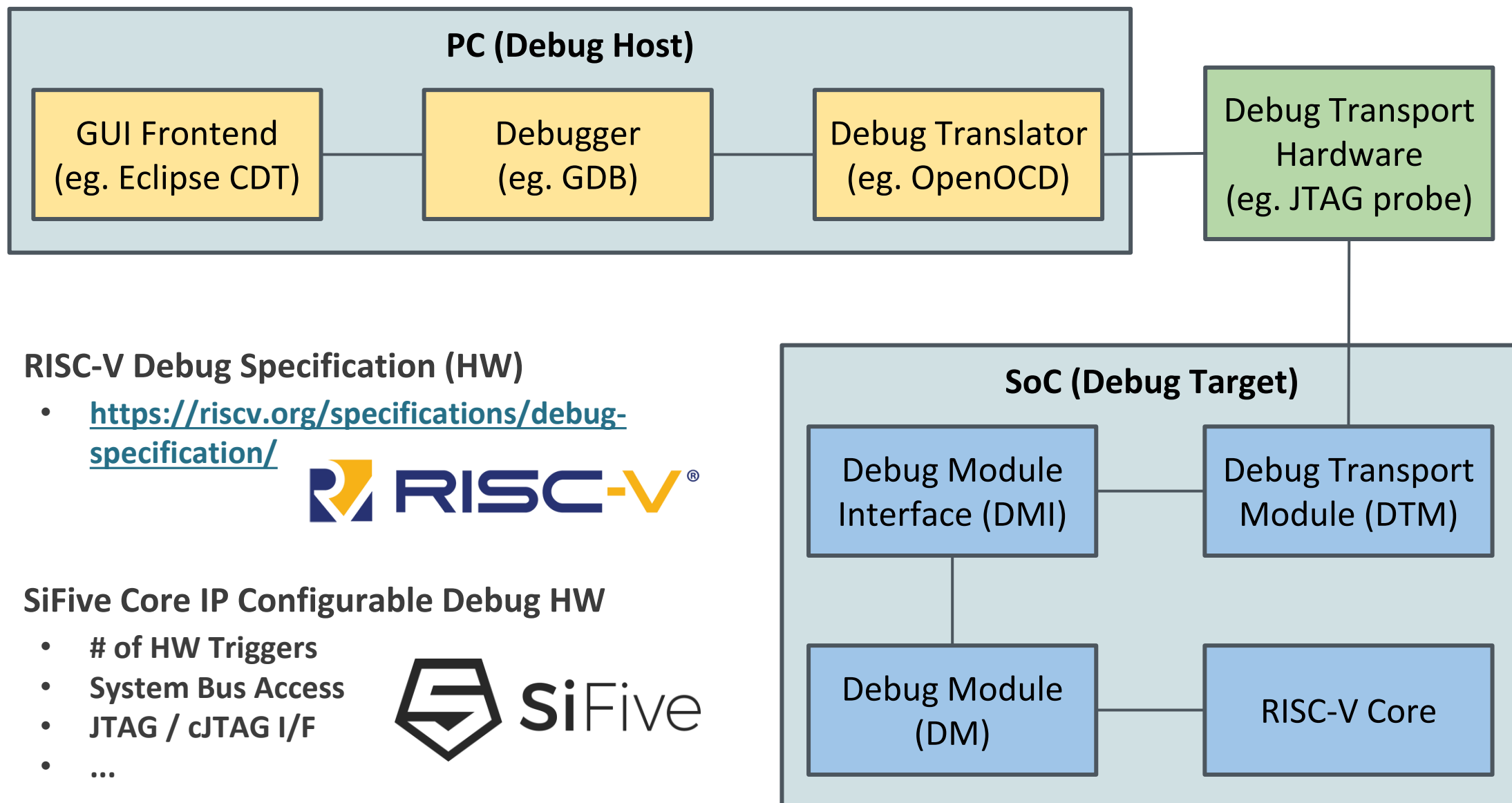


SiFive Tech Symposium 2019

RISC-V Core IP Debug SW Stack

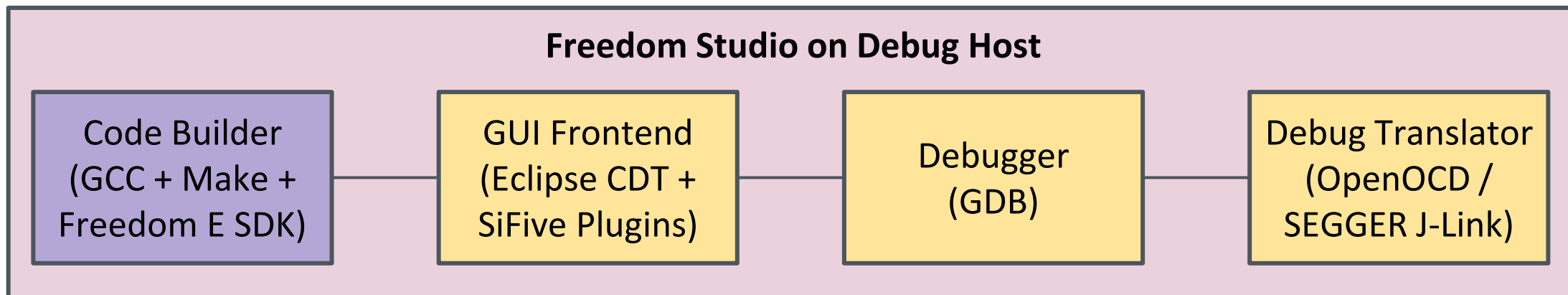


RISC-V Core IP Debug SW+HW Stack





Freedom Studio - A Fully Integrated Developer Environment



What is Freedom Studio

- IDE for Bare Metal embedded development & debugging on SiFive RISC-V Platforms and Core-IP
- Distributions available for Windows, MacOS and Linux
- Based on Open Source technology like Eclipse CDT
- Includes SiFive extensions
 - Toolchain integration
 - Debugger interfaces
 - Project management
- Bundled with all needed prebuilt packages
 - Java Runtime Environment
 - RISC-V OpenOCD
 - SEGGER J-Link SW
 - GNU RISC-V Newlib Toolchain (Binutils, GCC, Newlib, GDB)
 - Freedom E SDK (BSP's, Freedom Metal HAL, example programs)



FreedomStudio





Freedom Tools - Open Source Repository for SiFive's RISC-V Tools

What is Freedom Tools

- **Public repository with open source tools and scripts for building these tools**
 - <https://github.com/sifive/freedom-tools>
 - GNU RISC-V Newlib Toolchain (Binutils, GCC, Newlib (+Nano), GDB)
 - RISC-V OpenOCD
- **The toolchain setup is a riscv64-unknown-elf-gcc Newlib toolchain for Bare Metal embedded development**
 - Supported multilibs: rv32e, rv32eac, rv32em, rv32emac, rv32i, rv32iac, rv32im, rv32imac, rv32imaafc, rv32imafdc, rv64imac, rv64imaafc, rv64imafdc
- **We use these scripts to provide prebuilt binary distributions for Windows, MacOS and Linux**
- **We provide these prebuilt binary distributions as part of Freedom Studio and as standalone on the website**



Other SW packages bundled as part of the Freedom Studio distributions

- **SEGGER J-Link SW**
 - <https://www.segger.com/downloads/jlink/>
- **Java Runtime Environment**
 - <https://www.oracle.com/technetwork/java/javase/downloads/index.html>





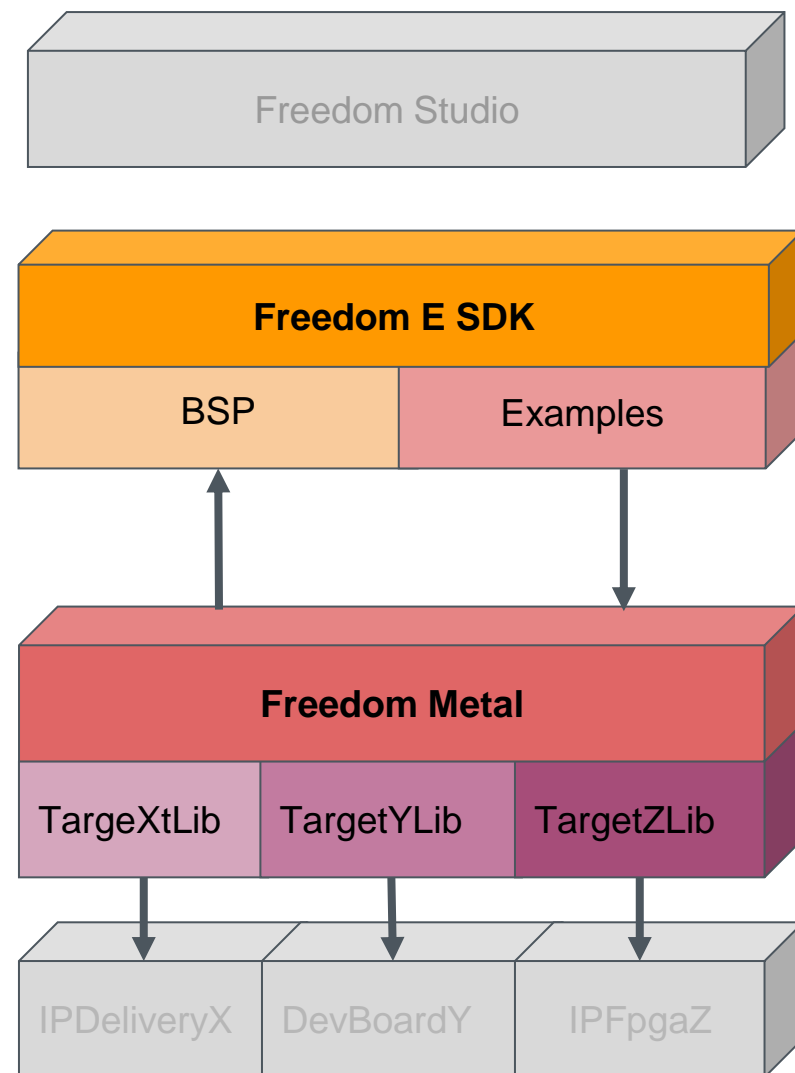
Freedom E SDK & Metal - A Bare Metal SW Stack for SiFive Devices

What is Freedom E SDK

- Embedded development kit providing a command line driven workflow with Examples and Utilities including BSP's for
 - Standard Core IP Deliverables
 - Standard Core FPGA Deliverables
 - SiFive Development Boards
- Examples uses Freedom Metal to provide portability
- Open source repository
 - <https://github.com/sifive/freedom-e-sdk>

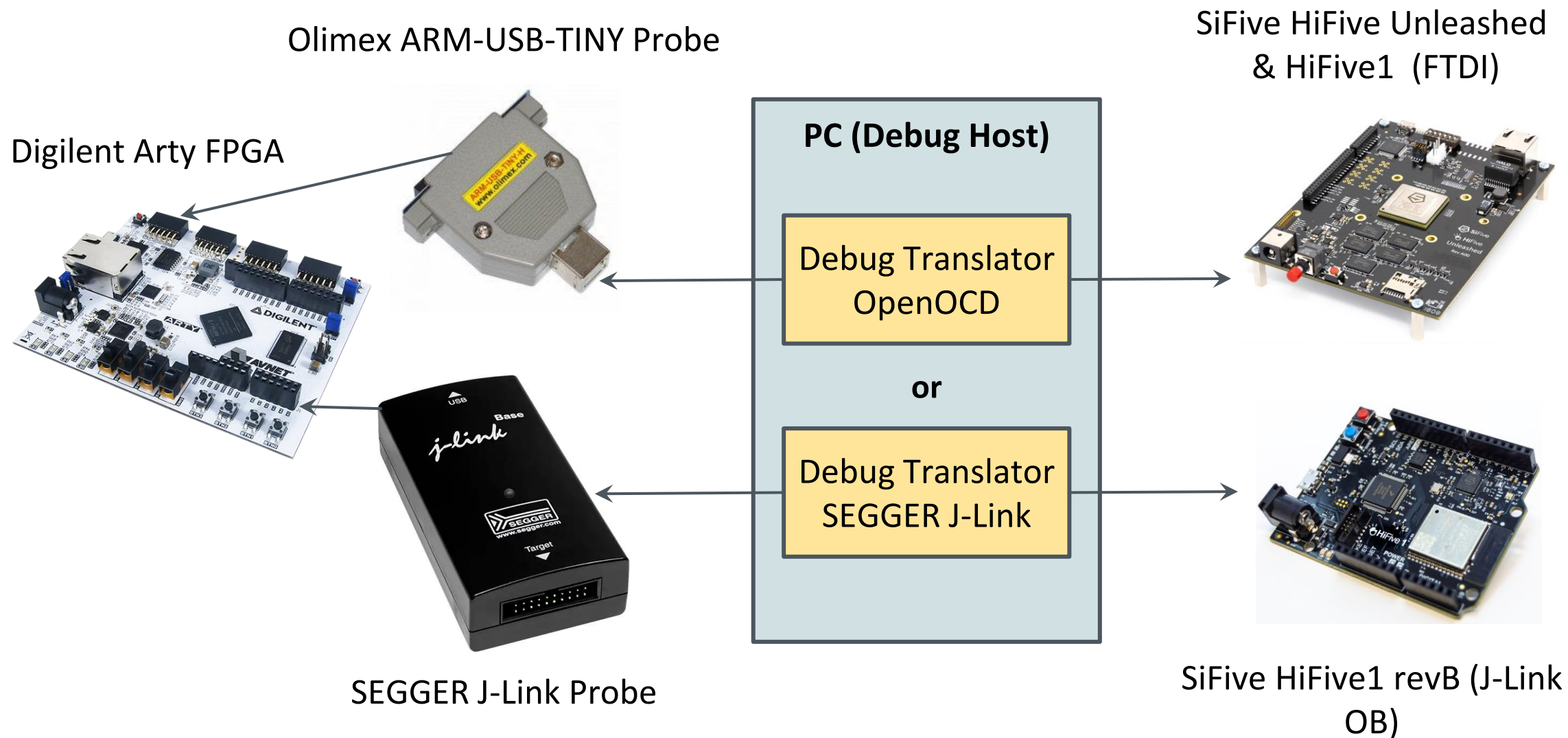
What is Freedom Metal

- Library for writing Portable, Bare Metal SW for all SiFive devices
 - A Bare Metal C application environment
 - An API for controlling CPU features and peripherals
 - The ability to retarget to any SiFive RISC-V product
 - A RISC-V hardware abstraction layer (HAL)
- Uses BSP's to provide target adaptation
- Open source repository
 - <https://github.com/sifive/freedom-metal>





Supported Debug Transport Hardware - JTAG Probes





How do you get it?

<https://www.sifive.com/boards>

Prebuilt RISC-V GCC Toolchain

Save time by using one of our prebuilt toolchains which contain all the tools necessary to compile and debug programs on SiFive products. Our toolchain distributions have been carefully packaged to support both 32-bit & 64-bit ISAs.

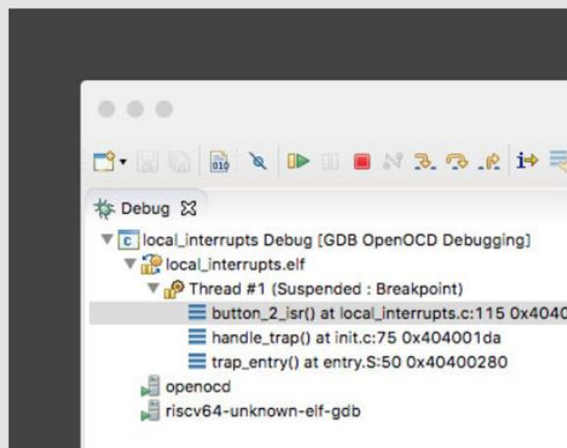
GNU Embedded Toolchain — v2019.02.0

OpenOCD — v2019.02.0

<https://www.sifive.com/boards>

Freedom Studio

Freedom Studio is the fastest way to get started programming with your SiFive hardware. Freedom Studio is built on top of the popular Eclipse IDE and packaged with a prebuilt toolchain and example projects from the Freedom E SDK. Freedom Studio is compatible with all SiFive RISC-V development boards.



Freedom Studio

Windows

macOS

Linux

Windows

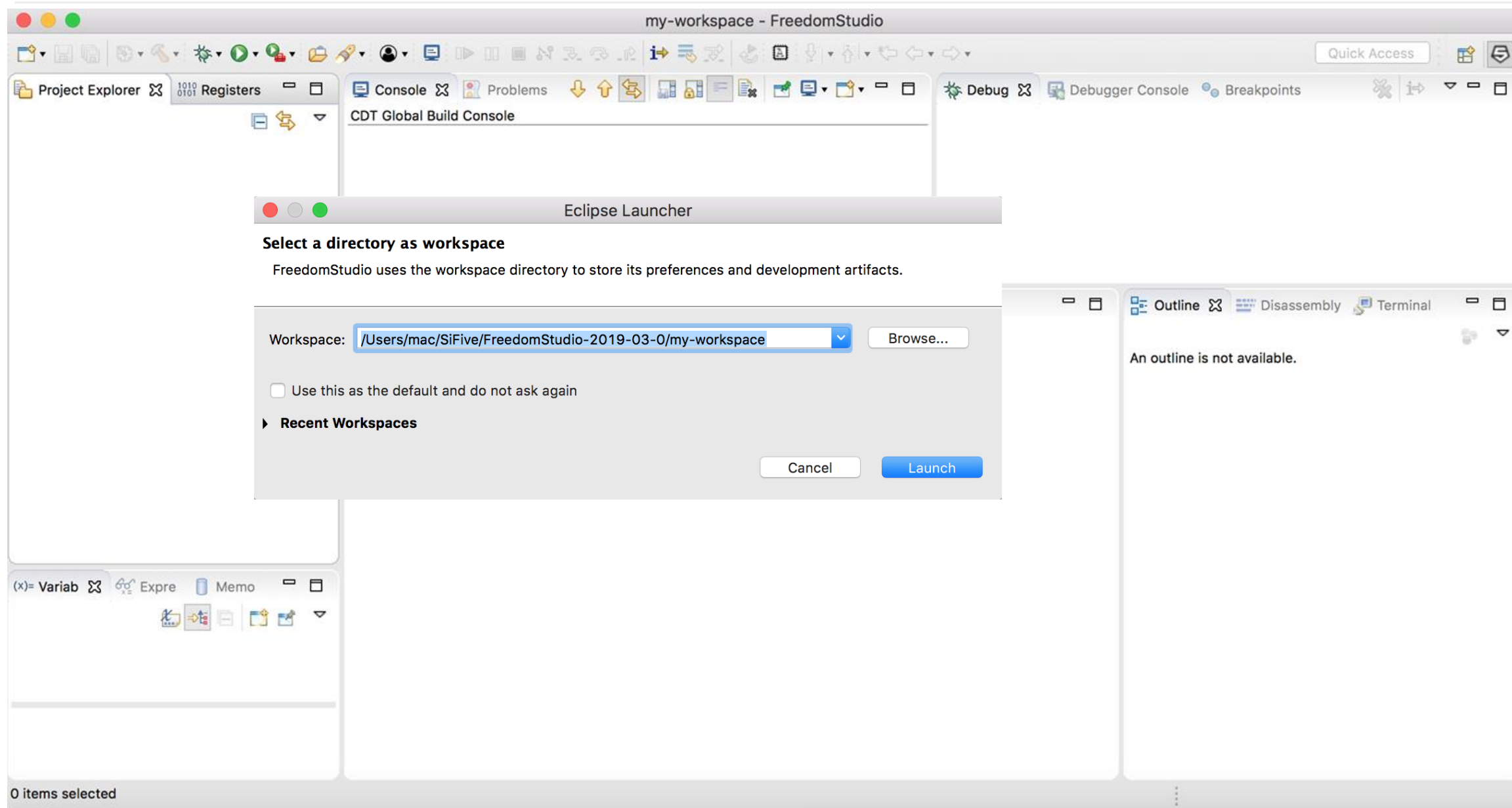
macOS

CentOS

Ubuntu



How does it look?





Freedom E SDK New Project Wizard

FreedomStudio

File Edit Navigate

New

Open File

C Project

Create C project of selected type

Project name: my-project

☒ Use default location

Location: /Users/mac/SiFive/FreedomStudio-2019-

Choose file system: default

Project type:

GNU Autotools

Executable

Shared Library

Static Library

Makefile project

Empty Project

Freedom E SDK Project

☒ Show project types and toolchains only if they are

Create a freedom-e-sdk standalone project for a specified target and example program.

Select Target

coreip-e31-arty

The SiFive E31 Standard Core is the world's most deployed RISC-V core. Co-designed alongside the RISC-V ISA, the E31 takes maximum advantage of the RISC-V ISA, resulting in a power-efficient core that delivers the high performance needed for tomorrow's smart IoT, storage, and industrial applications.

This FPGA core target is ideal for m

Select Example Program

☒ hello

example-itim

software-interrupt

timer-interrupt

local-interrupt

return-fail

return-pass

example-pmp

example-spi

empty

sifive-welcome

dhrystone

☒ Create an OpenOCD debug launch for this

Create a freedom-e-sdk debug launch

Debug Launch

Specify whether or not you want to create a debug launch with this new project. If you choose to create a debug launch, select the type of launch to create. After the project is created and built the Debug Launch Configuration dialog will be opened so that you can review the debug configuration and, optionally, launch the debugger.

☒ Create a debug launch configuration for this project

Debug launch type:

OpenOCD

J-Link

?

< Back

?

< Back

?

< Back

Next >

Cancel

Finish



Project Explorer and Console after project is created

Freedom-E-SDK:

```
make standalone
STANDALONE_DEST=/Users
/mac/SiFive/FreedomStudio-2019-03-0/my-
workspace/my-project
TARGET=coreip-e31-arty
PROGRAM=sifive-welcome
RISCV_PATH=/Users/mac/
SiFive/FreedomStudio-
2019-03-
0/FreedomStudio.app/Co
ntents/Eclipse/SiFive/
toolchain/riscv64-
unknown-elf-gcc-8.2.0-
2019.02.0
```

Standalone-Project:

```
make all
CONFIGURATION=debug
```

The screenshot displays the SiFive FreedomStudio IDE interface. The **Project Explorer** on the left shows a project structure for 'my-project'. The 'src' directory is expanded, showing 'debug' and 'sifive-welcome.elf - [none/le]'. Other files include 'sifive-welcome.hex', 'sifive-welcome.lst', 'sifive-welcome.map', 'sifive-welcome.c', 'Makefile', and 'README.md'. The 'bsp' directory is also expanded, showing various build files like 'design.dts', 'design.reglist', 'metal.default.lds', 'metal.h', 'metal.ramrodata.lds', 'metal.scratchpad.lds', 'openocd.cfg', 'README.md', 'settings.mk', 'freedom-metal', 'debug.mk', 'Makefile', and 'release.mk'.

The **Console** on the right shows the output of the build process. It starts with 'CDT Global Build Console' and lists various compiler flags and commands. The build process includes compiling 'sifive-welcome.c' into 'sifive-welcome.elf' and linking it with various libraries. The final output is 'sifive-welcome.hex'. The console message at the bottom indicates '21:12:44 Build Finished (took 7s.755ms)'.

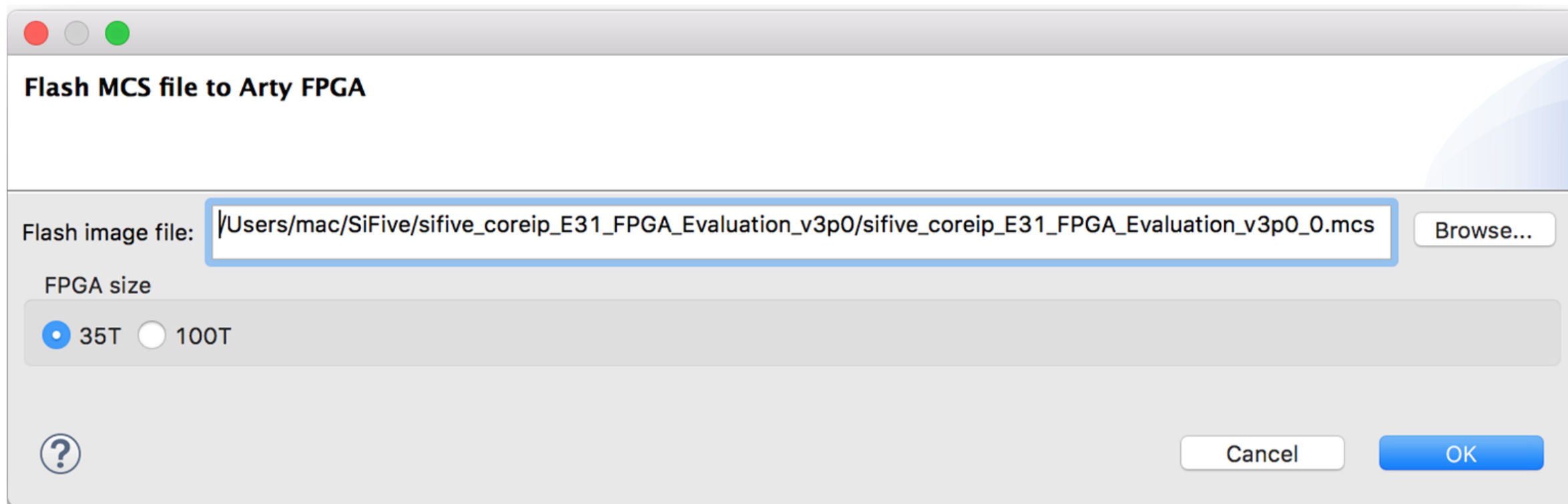
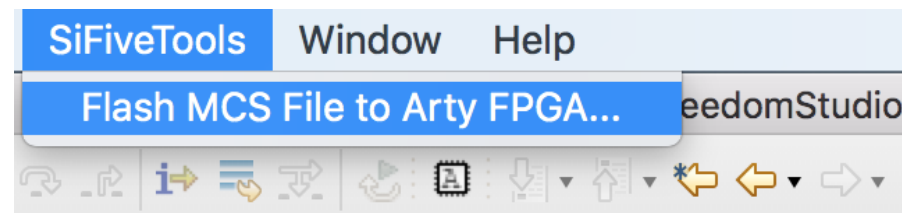
```
CDT Global Build Console
riscv64-unknown-elf-gcc -march=rv32imac -mabi=ilp32 -mmodel=medany -
ffunction-sections -fdata-sections -I/Users/mac/SiFive/
FreedomStudio-2019-03-0/my-workspace/my-project/bsp/install/include -O0 -g
-Wl,--gc-sections -Wl,-Map,sifive-welcome.map -nostartfiles -nostdlib -L/
Users/mac/SiFive/FreedomStudio-2019-03-0/my-workspace/my-project/bsp/
install/lib/debug/ -T/Users/mac/SiFive/FreedomStudio-2019-03-0/my-
workspace/my-project/bsp/metal.default.lds sifive-welcome.c -Wl,--start-
group -lc -lgcc -lmatal -lmatal-gloss -Wl,--end-group -o sifive-welcome
mv /Users/mac/SiFive/FreedomStudio-2019-03-0/my-workspace/my-project/src/
sifive-welcome.map /Users/mac/SiFive/FreedomStudio-2019-03-0/my-workspace/
my-project/src/debug/
mv /Users/mac/SiFive/FreedomStudio-2019-03-0/my-workspace/my-project/src/
sifive-welcome /Users/mac/SiFive/FreedomStudio-2019-03-0/my-workspace/my-
project/src/debug/sifive-welcome.elf
touch -c /Users/mac/SiFive/FreedomStudio-2019-03-0/my-workspace/my-
project/src/debug/sifive-welcome.elf
riscv64-unknown-elf-objdump --source --all-headers --demangle --line-
numbers --wide /Users/mac/SiFive/FreedomStudio-2019-03-0/my-workspace/my-
project/src/debug/sifive-welcome.elf > /Users/mac/SiFive/
FreedomStudio-2019-03-0/my-workspace/my-project/src/debug/sifive-
welcome.lst
riscv64-unknown-elf-size /Users/mac/SiFive/FreedomStudio-2019-03-0/my-
workspace/my-project/src/debug/sifive-welcome.elf
      text    data     bss     dec      hex filename
    19761    4200    61340    85301    14d35 /Users/mac/SiFive/
FreedomStudio-2019-03-0/my-workspace/my-project/src/debug/sifive-
welcome.elf
riscv64-unknown-elf-objcopy -O ihex /Users/mac/SiFive/
FreedomStudio-2019-03-0/my-workspace/my-project/src/debug/sifive-
welcome.elf /Users/mac/SiFive/FreedomStudio-2019-03-0/my-workspace/my-
project/src/debug/sifive-welcome.hex

21:12:44 Build Finished (took 7s.755ms)
```



Flashing Arty FPGA Image

- **Writes driver .bit file to FPGA via Xc3sprog (Digilent)**
 - E31 image in FPGA RAM (not Flash)
 - E31 image allows OpenOCD to access Flash
- **Writes MCS (ihex) file to Flash via OpenOCD (Olimex)**





Debug configurations

Debug Configurations

Create, manage, and run configurations

Name: openocd my-project

Project: my-project

C/C++ Application: src/debug/sifive-welcome.elf

Build (if required) before launching

Build Configuration: Use Active

Enable auto build

Disable auto build

Use workspace settings

Configure Workspace Settings...

Revert

Apply

Close

Debug

Project Explorer

my-project

Binaries

Debug As

Debug Configurations...

Organize Favorites...



Target Device information and configuration

Target Device Tree

Device Tree File

Actual Path

Selected cpu

Target Device Info

```
Number of cpus on target: 1
Selected cpu for this launch: cpu@0
Information being used from DTS file:
  Configure GDB (unless overridden):
    set arch riscv:rv32
    set remote hardware-breakpoint-limit 4
```

Register List

Register List File

Actual Path

Hardware Breakpoints

Hardware Breakpoint Limit

When unspecified here, will use a value of '4' (from [dts file](#)).
See also the [global](#) and [workspace](#) preferences, and the [project](#) properties.



Debugger setup - GDB client and OpenOCD server

OpenOCD Setup

☒ Start OpenOCD locally

Executable path:

Actual executable:
(to change it use the [global](#) or [workspace](#) preferences pages or the [project](#) properties page)

GDB port:

Telnet port:

Tcl port:

Config options:

☒ Allocate console for OpenOCD

GDB Client Setup

☒ Start GDB session

Executable name:

Actual executable:

Other options:

Commands:

Remote Target

Host name or IP address:

Port number:

☐ Force thread list update on suspend



Start debugging

my-workspace - my-project/src/sifive-welcome.c - FreedomStudio

Project Explorer

- my-project
 - Binaries
 - Archives
 - Includes
 - freedom-metal/src
 - src
 - debug
 - sifive-welcome.elf - [none/le]
 - sifive-welcome.hex
 - sifive-welcome.lst
 - sifive-welcome.map
 - release

Registers

Console

```
openocd my-project [SiFive GDB OpenOCD Debugging] openocd
(3289) hpmcounter24h (/32)
(3290) hpmcounter25h (/32)
(3291) hpmcounter26h (/32)
(3292) hpmcounter27h (/32)
(3293) hpmcounter28h (/32)
(3294) hpmcounter29h (/32)
(3295) hpmcounter30h (/32)
(3296) hpmcounter31h (/32)
(3922) mvendorid (/32)
```

Problems

Debug

Debugger Console

Breakpoints

openocd my-project [SiFive GDB OpenOCD Debugging]

- sifive-welcome.elf
 - Thread #1 (Suspended : Breakpoint)
 - main() at sifive-welcome.c:87 0x40400382
 - openocd
 - riscv64-unknown-elf-gdb

Variables

Name	Type	Value
(x)=rc	int	0x0
led0_red	struct metal_led *	0x4040436a
vtable	const struct met...	0x87aabfd9
led0_green	struct metal_led *	0x1
led0_blue	struct metal_led *	0x0

Name : led0_red

Details: 0x4040436a <__libc_init_array+108>
Default: 0x4040436a <__libc_init_array+108>
Decimal: 1077953386
Hex: 0x4040436a
Binary: 1000000010000000100001101101010
Octal: 010020041552

Expression

Expression	Type	Value
led0_red	struct metal_led *	0x4040436a
vtable	const struct metal_led_vtable *	0x87aabfd9

Name : led0_red

Details: 0x4040436a <__libc_init_array+108>
Default: 0x4040436a <__libc_init_array+108>
Decimal: 1077953386
Hex: 0x4040436a
Binary: 1000000010000000100001101101010

timer_isr(int, void*) : void
wait_for_timer(struct metal_led*)
main(void) : int

sifive-welcome.c

```
int main (void)
{
    int rc, up_cnt, dn_cnt;
    struct metal_led *led0_red, *led0_green, *led0_blue;

    // This demo will
    led0_red = metal_l
    led0_green = metal
    led0_blue = metal
    if ((led0_red == N
        printf("At lea
    return 1;
}

// Enable each LED
metal_led_enable(l
metal_led_enable(l
metal_led_enable(led0_blue);

// All Off
```



The screenshot shows the 'Export Memory' dialog box in a debugger. The dialog has the following fields and values:

- Format:** Plain Text (selected from a dropdown menu)
- Start address:** 0x80000000
- End address:** 0x80000010
- Length:** 16
- File name:** /Users/mac/SiFive/my-memory-export.hex

Buttons at the bottom include a question mark icon, 'Cancel', and 'OK'. The background shows a memory dump table with columns for Address, 0-3, 4-7, 8-B, and C-F. The first row of data shows the address 0x80000000 with a value of FFFFFFFA8 in the 0-3 column.



```
zero
ra
sp
gp
tp
t0
t1
t2
fp
s1
a0
a1
a2
a3
a4
a5
a6
a7
s2
...
```

Launch Terminal

Choose terminal: Serial Terminal

Settings

Serial port: /dev/cu.usbserial-210319A28D950

Baud rate: 115200

Data size: 8

Parity: None

Stop bits: 1

Encoding: Default (ISO-8859-1)

Cancel OK

19A28D951

FIVE, INC.

5555555555555555

5555

5555

5555

555555555555555555

555555555555555555

5555

5555

5555

55555555 55555

555555 55555

5555 55555

5 55555

55555

55555

55555

55555

555555

5555

5



Viewing disassembly

```
sifive-welcome.c
    return 4;
}
cpu_intr = metal_cpu_interrupt_controller(cpu0);
if (cpu_intr == NULL) {
    printf("CPU interrupt controller is null.\n");
    return 3;
}
metal_interrupt_init(cpu_intr);

// display welcome banner
display_banner();

// Setup Timer and its interrupt so we can toggle LEDs on 1s cadence
tmr_intr = metal_cpu_timer_interrupt_controller(cpu0);
if (tmr_intr == NULL) {
    printf("TIMER interrupt controller is null.\n");
    return 4;
}
metal_interrupt_init(tmr_intr);
tmr_id = metal_cpu_timer_get_interrupt_id(cpu0);
rc = metal_interrupt_register_handler(tmr_intr, tmr_id, timer_isr, cpu0);
if (rc < 0) {
    printf("TIMER interrupt handler registration failed\n");
    return (rc * -1);
}

// Lastly CPU interrupt
if (metal_interrupt_enable(cpu_intr, 0) == -1) {
    printf("CPU interrupt enable failed\n");
    return 6;
}
```

```
Outline Disassembly Terminal 1
Enter location here
114      if (cpu_intr == NULL) {
40400478:  li      a5,3
4040047a:  j       0x4040057a <main+512>
116      return 3;
4040047c:  auipc   a5,0x3fc01
40400480:  addi    a5,a5,-992 # 0x8000109c <cpu_intr>
40400484:  lw      a5,0(a5)
40400486:  mv      a0,a5
40400488:  jal     ra,0x40402132 <metal_interrupt_init>
119
4040048c:  jal     0x40400180 <display_banner>
122
4040048e:  auipc   a5,0x3fc01
40400492:  addi    a5,a5,-1006 # 0x800010a0 <cpu0>
40400496:  lw      a5,0(a5)
40400498:  mv      a0,a5
4040049a:  jal     ra,0x40402092 <metal_cpu_timer_interrupt_controll
4040049e:  mv      a4,a0
404004a0:  auipc   a5,0x3fc01
404004a4:  addi    a5,a5,-1032 # 0x80001098 <tmr_intr>
404004a8:  sw      a4,0(a5)
123      // Setup Timer and its interrupt so we can toggle LEDs
404004aa:  auipc   a5,0x3fc01
404004ae:  addi    a5,a5,-1042 # 0x80001098 <tmr_intr>
404004b2:  lw      a5,0(a5)
404004b4:  bnez    a5,0x404004c4 <main+330>
124      tmr_intr = metal_cpu_timer_interrupt_controller(cpu0);
404004b6:  auipc   a0,0x4
404004ba:  addi    a0,a0,1430 # 0x40404a4c
404004be:  jal     0x4040060e <puts>
```



Setting breakpoints

- Types: temporary, regular, hardware, conditional, dynamic printf, data watchpoint

The screenshot shows an IDE with a C file named `sifive-welcome.c` open. The code contains several `printf` statements and a `while` loop. A breakpoint is set at line 147, which is a `printf` statement. The `Breakpoints` panel on the right shows a list of breakpoints, including a temporary breakpoint for the `main` function, a regular breakpoint at line 121, a hardware breakpoint at line 150, a conditional breakpoint at line 153, and a dynamic printf breakpoint at line 147. The `Properties for C/C++ Line Dynamic Printf` dialog is open, showing the `Common` tab. The dialog fields are as follows:

Common	
Class:	C/C++ Line Dynamic Printf
File:	/Users/mac/SiFive/FreedomStudio-2019-03-0/my-workspace/my-project/src/sifive-welcome.c
Line number:	147
Enabled:	<input checked="" type="checkbox"/>
Condition:	
Ignore count:	0
printf:	"Hit line %d of /Users/mac/SiFive/FreedomStudio-2019-03-0/my-workspace/my-project/src/si



Debugger interfaces

- Debugger Console is a direct interface to the GDB command shell
- GDB “monitor” command feeds directly to OpenOCD with output in Console
- Debug View shows the state of the debugging session

The screenshot displays the SiFive FreedomStudio IDE interface during a debugging session. The top-left pane shows the **Console** output, which includes OpenOCD startup messages and memory dump results for registers `hpmcounter31h`, `mvendorid`, `marchid`, `mimpid`, `mhartid`, and `priv`. The top-right pane shows the **Debugger Console** with the `info breakpoints` command output, listing two breakpoints: one for `breakpoint` at address `0x4040048e` (hit 1 time) and one for `dprintf` at address `0x4040056c` (hit 4 times). The bottom-left pane shows the source code for `sifive-welcome.c`, with a `while (1) {` loop containing calls to `wait_for_timer` for RED, Green, and Blue LEDs. The bottom-right pane shows the **Debug View**, which displays the call stack for the current thread. The top entry is `wait_for_timer() at sifive-welcome.c:73 0x4040034e`, followed by `main() at sifive-welcome.c:151 0x40400578`. Below the call stack, the loaded modules `openocd` and `riscv64-unknown-elf-gdb` are listed.

```
openocd my-project [SiFive GDB OpenOCD Debugging] openocd
(3296) hpmcounter31h (/32)
(3922) mvendorid (/32)
(3923) marchid (/32)
(3924) mimpid (/32)
(3925) mhartid (/32)
(4161) priv (/8)
0x80000000: ffffffff
0x80000000: ffffffff

info breakpoints
Num      Type           Disp Enb Address      What
2        breakpoint      keep y   0x4040048e in main at sifive-welcome.c:122
          breakpoint already hit 1 time
6        dprintf        keep y   0x4040056c in main at sifive-welcome.c:148
          breakpoint already hit 4 times
          printf "Hit line %d of /Users/mac/SiFive/FreedomStudio-2019-03-0/my-wor
monitor mdw 0x80000000

// Red -> Green -> Blue, repeat
while (1) {

    // Turn on RED
    wait_for_timer(led0_red);

    // Turn on Green
    wait_for_timer(led0_green);

    // Turn on Blue
    wait_for_timer(led0_blue);

}

openocd my-project [SiFive GDB OpenOCD Debugging]
  sifive-welcome.elf
    Thread #1 (Suspended : Signal : SIGINT:Interrupt)
      wait_for_timer() at sifive-welcome.c:73 0x4040034e
      main() at sifive-welcome.c:151 0x40400578
    openocd
    riscv64-unknown-elf-gdb
```



Using other versions of Freedom-E-SDK than the bundled one

The image shows two screenshots of the Freedom Studio Preferences dialog, illustrating how to configure the Freedom E SDK Path.

Top Screenshot: Global Freedom E SDK Path

- Left Panel:** A tree view with categories: General, C/C++, and Freedom Studio. Under Freedom Studio, the following options are listed: Global Freedom E SDK Path (highlighted), Global HW Breakpoint Limit, Global OpenOCD Path, Global Register List File, Global SEGGER J-Link Path, and Global Toolchain Path.
- Right Panel:** Titled "Global Freedom E SDK Path". It contains a text box with the path: `/Users/mac/SiFive/FreedomStudio-2019-03-0/FreedomStudio.app/Contents/Eclipse/SiFive/freedom-e-sdk`. A "Browse..." button is to the right. Below the text box is a "Restore Defaults" button and an "Apply" button.

Bottom Screenshot: Workspace Freedom E SDK Path

- Left Panel:** Similar to the top screenshot, but the "Workspace Freedom E SDK Path" option is highlighted under the Freedom Studio category.
- Right Panel:** Titled "Workspace Freedom E SDK Path". It contains a text box with the path: `/Users/mac/SiFive/FreedomStudio-2019-03-0/my-freedom-e-sdk-clone`. A "Browse..." button is to the right. Below the text box are "Restore Defaults" and "Apply" buttons. At the bottom right, there is a blue "Apply and Close" button.

Instructions (highlighted in a red box):

1. Point to another SDK with BSP's from SiFive Core Designer
2. Point to newer SDK with updated BSP's, Examples or Freedom Metal
3. ...



Using other tools than the bundled ones

The screenshot shows the Eclipse IDE Preferences dialog with the 'Global Toolchain Path' and 'Global OpenOCD Path' sections. The 'Global Toolchain Path' section is selected in the left sidebar. The 'Global OpenOCD Path' section is also visible below it. A red box highlights a list of instructions for using other tools.

Global Toolchain Path

Select the root directory of the globally preferred Newlib riscv64-unknown-elf toolchain. The chosen directory must contain a 'bin' directory with an executable compiler. Leave blank to have no preference. This global setting can be overridden by the workspace preference, or the project property setting.

Toolchain Path:

Global OpenOCD Path

Configure the location of the OpenOCD Executable (in 'bin' directory). The values are stored within Eclipse. Unless redefined more specifically, they are used for all projects in all workspaces.

After installing OpenOCD updates, restart Eclipse for the defaults to be re-evaluated and use the Restore Defaults button to configure the new location.

OpenOCD Path:

Instructions for using other tools:

1. Point to tools with different configurations like multilibs, cmodel, ...
2. Point to newer tools with new features or bug fixes
3. ...



Accessing bundled documentation

sifive-welcome.c

Browse Tools Docs

Tools documentation paths:

- /Users/mac/SiFive/FreedomStudio-2019-03-0/FreedomStudio.app/Contents/Eclipse/doc/html
- /Users/mac/SiFive/FreedomStudio-2019-03-0/FreedomStudio.app/Contents/Eclipse/doc/html/other
- /Users/mac/SiFive/FreedomStudio-2019-03-0/FreedomStudio.app/Contents/Eclipse/SiFive/toolchain/riscv64-unknown-elf-gcc-8.2.0-2019.02.0/share/doc
- /Users/mac/SiFive/FreedomStudio-2019-03-0/FreedomStudio.app/Contents/Eclipse/SiFive/openocd/riscv-openocd-0.10.0-2019.02.0/share/doc

Tools documentation links:

• /freedom/studio/	Freedom Studio User Manual
• /sifive/devkit/	SiFive DevKit Documentation
• /annotate/	GDB's Obsolete Annotations
• /as/	Using as
• /bfd/	Using bfd
• /binutils/	GNU Binary Utilities
• /cpp/	The C Preprocessor
• /gcc/	Using the GNU Compiler Collection (GCC)
• /gdb/	Debugging with GDB
• /gmp/	GNU MP 6.1.0
• /gprof/	GNU gprof
• /ld/	LD
• /mpc/	GNU MPC 1.0.3
• /mpfr/	GNU MPFR 3.1.4
• /newlib/libc/	The Red Hat newlib C Library
• /newlib/libm/	LIBM
• /porting/	Embed with GNU
• /stabs/	STABS
• /openocd/openocd/	OpenOCD User's Guide

Help

Search

Help Contents

Search

Show Contextual Help

Browse Tools Docs

Show Active Keybindings...

Tips and Tricks...

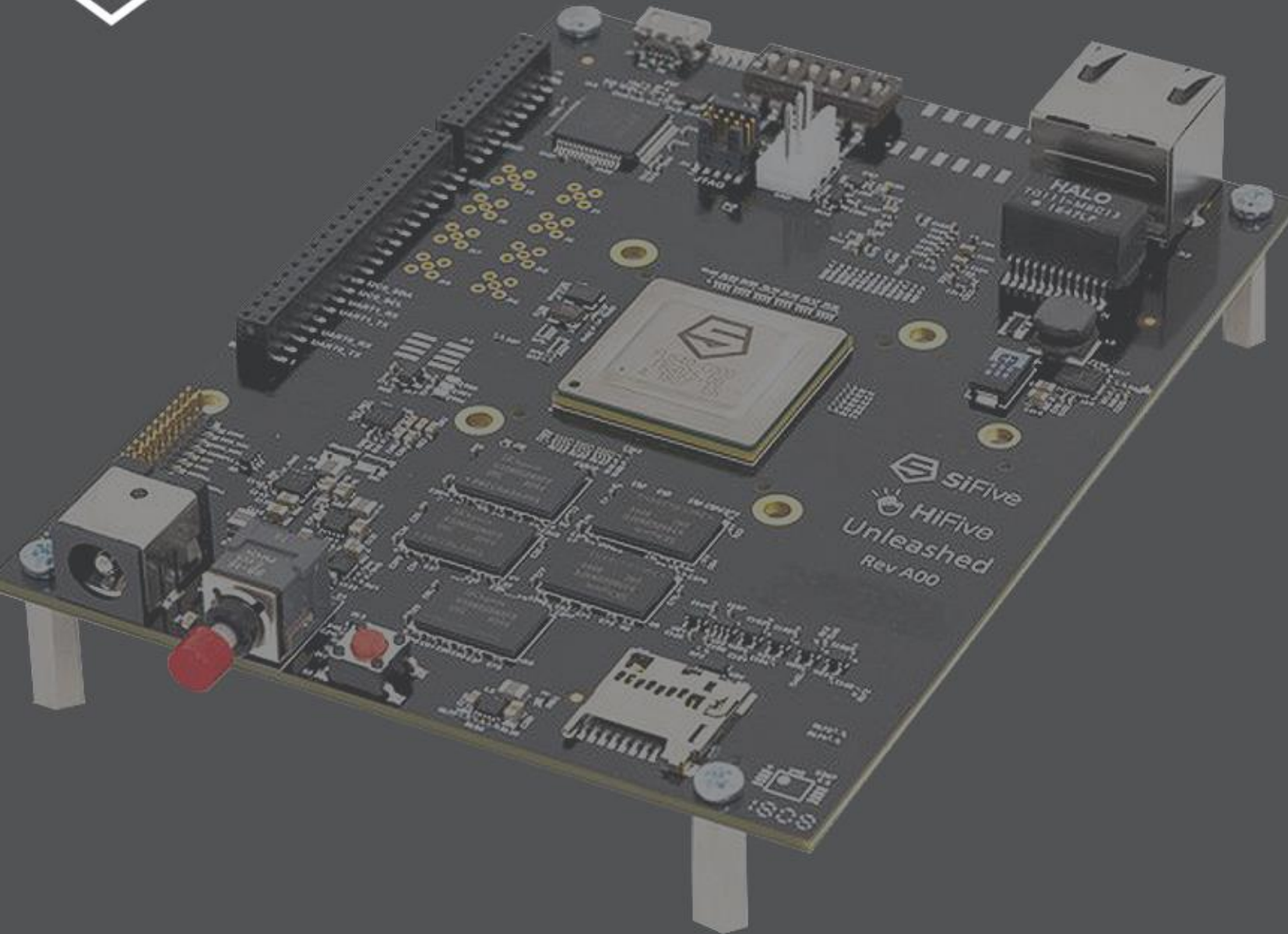
Cheat Sheets...

Eclipse User Storage

Check for Updates

Install New Software...

Eclipse Marketplace...



Silicon verified. Market proven.

The most advanced configurable core IP and silicon solutions from the inventors of RISC-V.

Microcontrollers ■ Embedded ■ Linux ■ Multicore

■ Networking ■ Storage ■ Computing ■ AI
■ Industrial ■ IoT ■ Consumer ■ Automotive

www.sifive.com