

HiFive1 revb 환경 구축, 컴파일, 테스트, 그리고 타이밍

<https://www.sifive.com/software> 를 통해 Freedom Studio를 설치한다.



<https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack>를 통해 SEGGER J-Link 윈도우 드라이버를 설치한다.

	Version	Date	File size	
J-Link Software and Documentation pack for Windows Installing the software will automatically install the J-Link USB drivers and offers to update applications which use the J-Link DLL. Multiple versions of the J-Link software can be installed on the same PC without problems; they will co-exist in different directories.	V6.92	[2020-12-18]	50,642 KB	DOWNLOAD

Freedom Studio의 E SDK Project를 통해 Hello World 프로젝트를 새롭게 생성한다.

Create a Freedom E SDK Project

Use this Freedom E SDK

C:\Users\Winfo\FreedomStudio-4.7.2.2020-11-3-x86_64-w64-mingw32\SiFive\Freedom-e-sdk-master

⚠ Do not use the bundled freedom-e-sdk to build example projects for an IP deliverable package ([more info in the KB](#))!

Select Target

sifive-hifive1-revb

☐ Use the BSP from the SDK when building this project

☐ Use the Metal Library from the SDK when building this project

Open S5KB

Select Example Program

hello

hello

A simple "Hello, World!" example to demonstrate printf and build environment.

Options

Project name sifive_hifive1_revb_hello_1

(you can change the project location on the next page)


























☒ Build the project

☒ Create a debug launch configuration for J-Link

Watch

< Back Next > Finish Cancel

프로그램을 컴파일하고 난 이후 hello.elf를 SiFive J-Link Launch를 통해 실행해 주게 되면 프로그램이 디바이스 상으로 업로드된다. 여기서 J-Link USB는 Freedom Studio에서 자동 설치가 안되기 때문에 수동으로 설치해 주어야 한다.

- ▼  sifive_hifive1_revb_hello_1
 - >  Build Targets
 - >  Binaries
 - >  Archives
 - >  Includes
 - >  freedom-metal/src
 - ▼  src
 - ▼  debug
 - >  hello.elf - [none/le]
 -  hello.hex
 -  hello.map
 - >  hello.c
 -  LICENSE
 -  LICENSE.Apache2
 -  LICENSE.MIT
 -  Makefile
 -  README.md
 - >  bsp
 - >  freedom-metal
 - >  scripts
 -  debug.mk
 -  freedom-e-sdk.mk
 -  Makefile
 -  release.mk
 -  requirements.txt

타이밍을 측정하기 위해서는 내부 카운터를 활용해야 하며 이에 대한 코드는 아래와 같다. 카운터는 실시간 카운터이기 때문에 디버그를 통해 카운터값을 가지고 오면 정확한 값을 확인할 수 없다. 따라서 printf문을 통해 확인하는 것이 정확하다. 여기서 minstret은 명령어의 수를 의미하고 mcycle은 클럭 사이클을 의미한다.

```
/* Copyright 2019 SiFive, Inc */
/* SPDX-License-Identifier: Apache-2.0 */

#include <stdio.h>
#include <metal/cpu.h>
#include <metal/hpm.h>
#include <stdint.h>
#include <stdio.h>
#include <time.h>

int main() {
    struct metal_cpu *cpu;

    /* Get CPU device handle. */
    cpu = metal_cpu_get(metal_cpu_get_current_hartid());

    /* Enable module */
    if (metal_hpm_init(cpu) != 0) {
        return 1;
    }
}
```

```
unsigned long hi = 0, lo = 0;
unsigned long hi2 = 0, lo2 = 0;
unsigned long long num = 0;

asm volatile ("csrr %0, minstret" : "=r"(lo2));
asm volatile ("csrr %0, mcycle" : "=r"(lo));

for (int i = 0; i < 10000; i++) {
    num++;
}

asm volatile ("csrr %0, mcycle" : "=r"(hi));
asm volatile ("csrr %0, minstret" : "=r"(hi2));
printf("%d\n", hi - lo);
printf("%d\n", hi2 - lo2);
```

```
}
```