**임베디드 시스템 프로젝트 보고서**

**2015253039 권진우**

**보고서 목차**

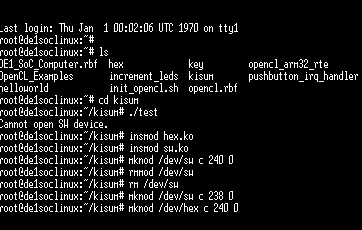
* 프로젝트 제목
* 프로젝트 설명
* 프로젝트 수행 과정
* 프로젝트 주요 소스
* 프로젝트 Audio I/O register 특징 조사/설명
* 프로젝트 동작 + 기타사항
* 프로젝트 Reference

**제목** : **HEX display, Switch, Audio를 사용한 피아노 및 간단한 음악 출력하기**

**설명** : 이번 프로젝트는 스위치와 HEX LED 그리고 AUDIO I/O를 사용하여서 Switch를 사용하여 출력할 음계를 선택하고 HEX LED를 사용하여서 현재 & 지난 음계를 출력해주며 AUDIO I/O를 사용하여서 음계의 주파수에 맞는 소리를 출력하도록 하였습니다. 또한 응용 프로그램 내에 간단한 노래 음계를 넣어서 가장 좌측 스위치를 누르면 노래가 나오도록 만들었습니다.

**수행 과정**

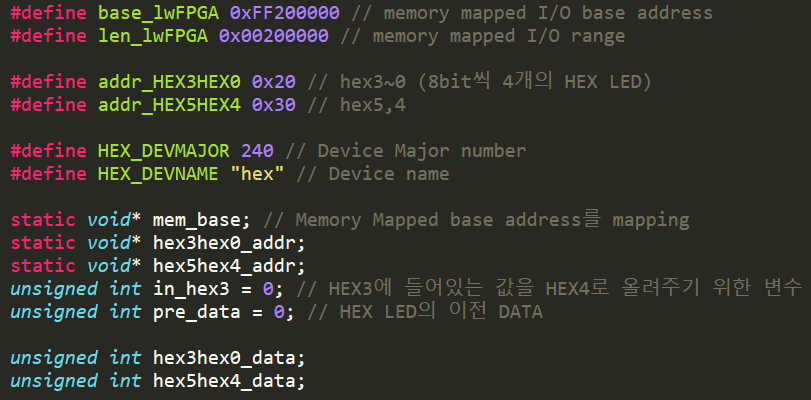
1. **프로젝트 구상** : AUDIO를 사용한 I/O프로젝트를 하고 싶었으며 Switch를 사용하여서 음계를 적용하여 음을 출력하는 피아노 프로젝트를 수행하고 싶었습니다.
2. **I/O Driver 작성** : Switch와 HEX LED를 위한 Driver를 작성하여서 원하는 동작을 수행하도록 sw, hex driver를 만들어 Runtime동안 설치하였습니다.
3. **Driver 설치** 과정은 Driver.c 파일 작성(init, exit, read, write()), Makefile을 통해 .ko파일 생성, .ko파일을 사용하여 insmod 명령어로 커널 모듈 설치, mknod 명령어로 I/O를 위한 File을 생성합니다.
4. **응용프로그램에서 Audio 출력**을 위해서 mmap 함수를 사용하고 pointer를 사용하여memory mapped된 audio 주소를 사용하여 sin 함수를 적절히 이용해 적절한 음계에 대한 주파수를 만들어준 후 While문 내에서 loop를 돌며 Audio가 출력되도록 작성
5. **실행 파일을 만들어서 Test** 수행

<- hex.ko, sw.kokernel module 설치

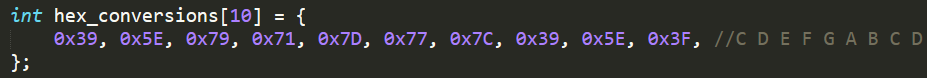
**주요 소스 설명(driver, user\_app\_program)**

**HEX\_driver.c의 주요 코드**

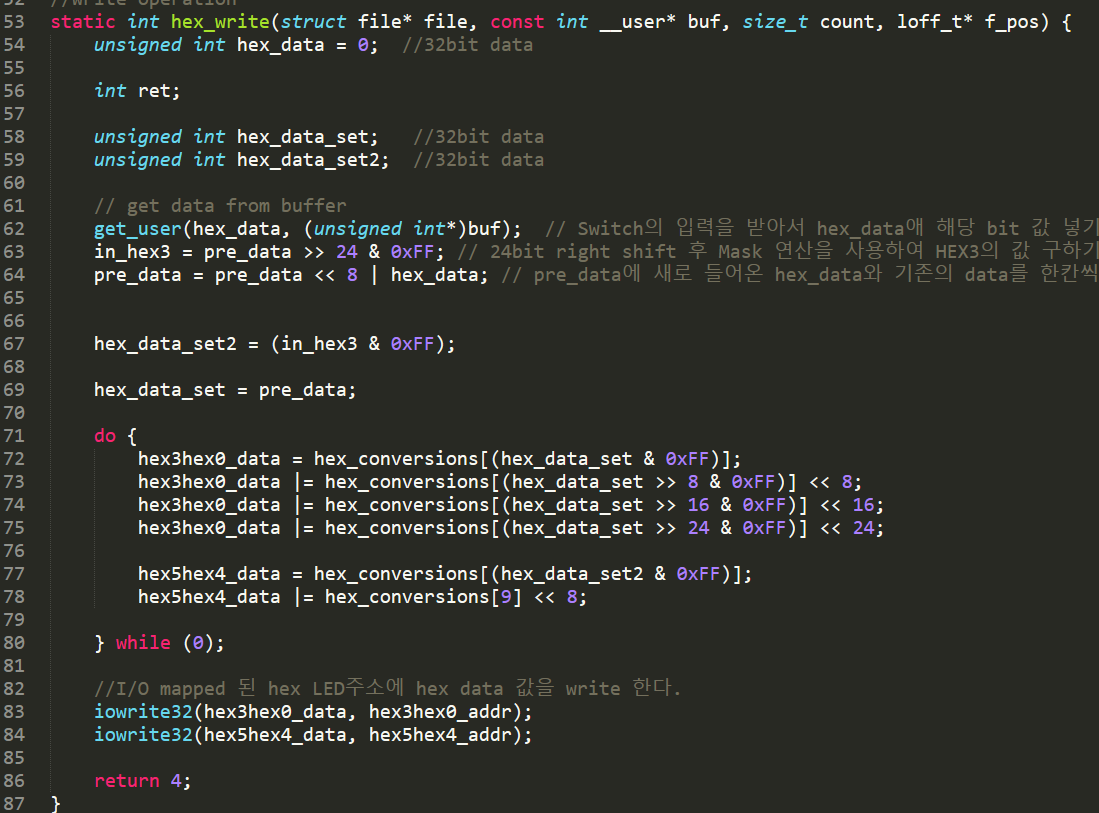
HEX LED Driver module을 설치하기 위해서 작성되어야 하는 Driver 파일입니다.



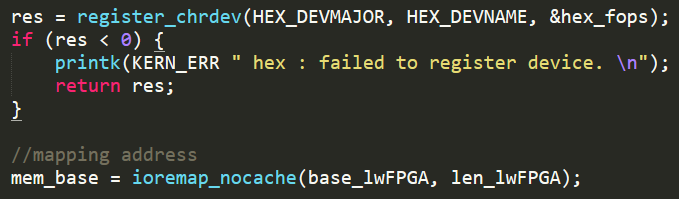
->HEX LED I/O 장치를 다루기 위해서 먼저 init함수에서 사용할 Memory Mapped I/O address에액세스하기 위해 I/O Mapped 된 주소 값의 시작위치와 범위를 define하고 HEX LED의 상대주소를 define 합니다.또한 I/O device의 Major number와 name을 define하고 필요한 변수를 전역으로 선언하였습니다.



음계 코드를 HEX LED에 출력하기 위해서 각 Index에 대해서 C~B까지의 코드를 HEX LED에 맞게 변형하는 변수를 만들었습니다.



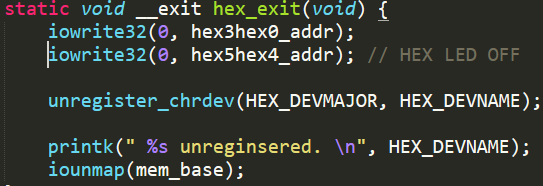
->HEX LED Driver를 통해서는 write만 수행하여 출력만 할 것이기 때문에write함수만을 사용한다. get\_user커널 함수를 통해 Switch의 값을 받아서 HEX LED에 적절히 출력되도록 iowrite32()를 사용해 HEX LED 주소에 data값을 write 합니다.



->Register\_chrdev()를 통해서 문자 드라이버 디바이스인 HEX LED driver를 kernel에 등록하고mem\_base에 Physical I/O memory address를 virtual address로 mapping한다.



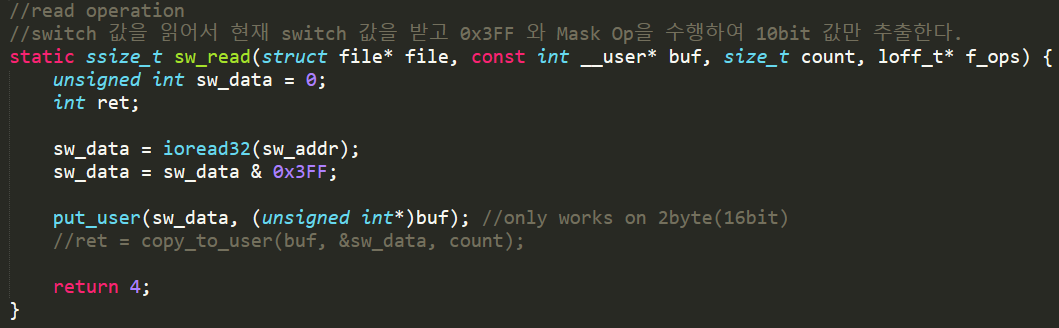
->HEX LED의 주소를 mapping



->driver 해제 시 iowrite()로 LED를 점멸하고 iounmap을 사용하여 virtual address로 mapping 하였던 주소를 해제한다.또한 unregister\_chrdev()를 사용하여 드라이버를 해제함으로써 드라이버 해제가 이루어진다.

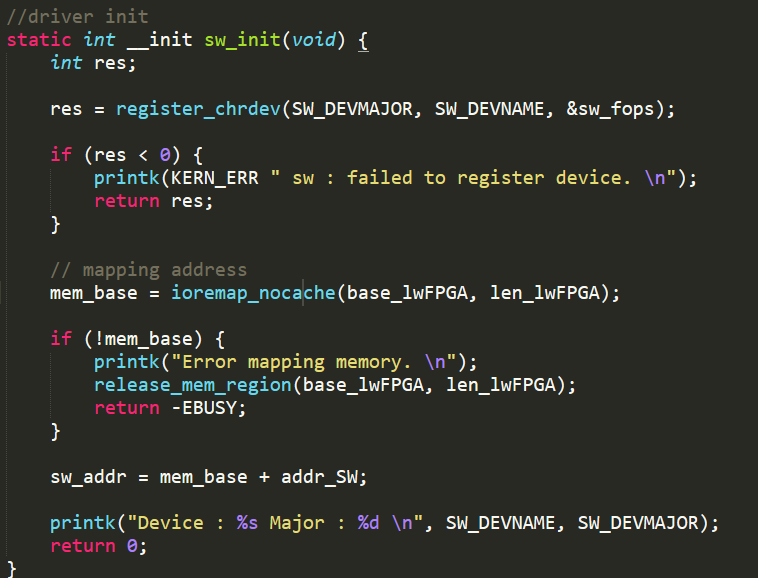
**Switch\_driver.c의 주요 코드**

Switch를 다루기 위한 Driver 작성 입니다.



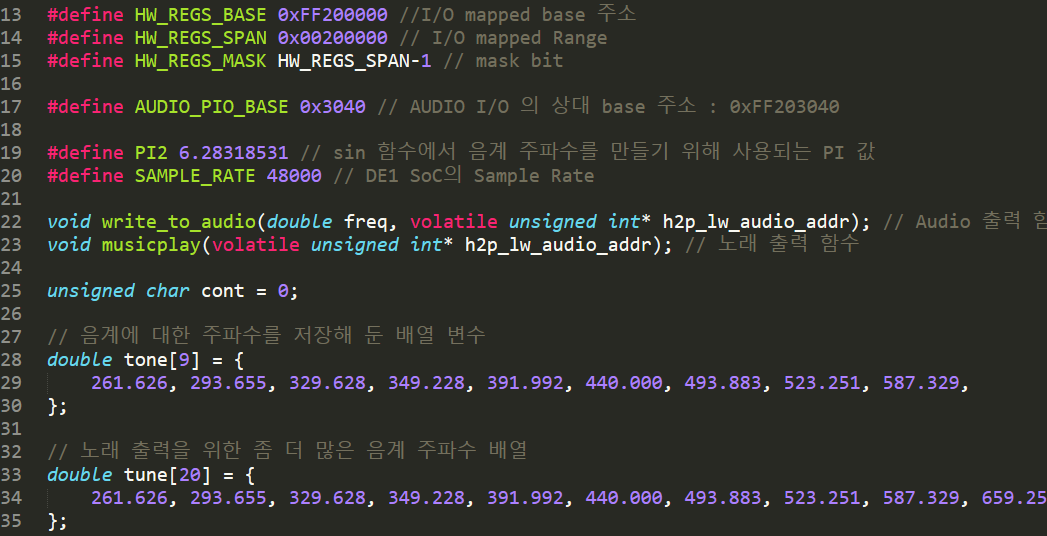
->sw\_addr는 switch의 data를 나타내는 register의 주소이며 sw\_data 변수에 switch 값을 넣고 다시 put\_user를 사용해서 switch data를 돌려줍니다.

Init 함수

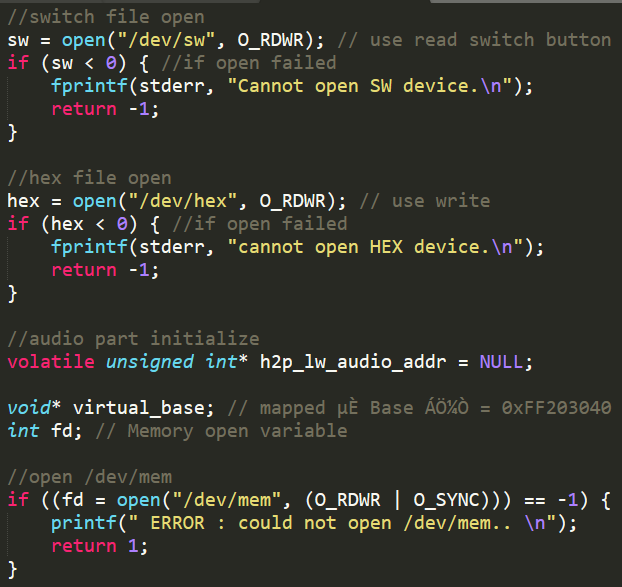


->init함수로써 memory Mapping, 문자 디바이스 Driver 등록,register 초기화 등의 작업을 수행한다.

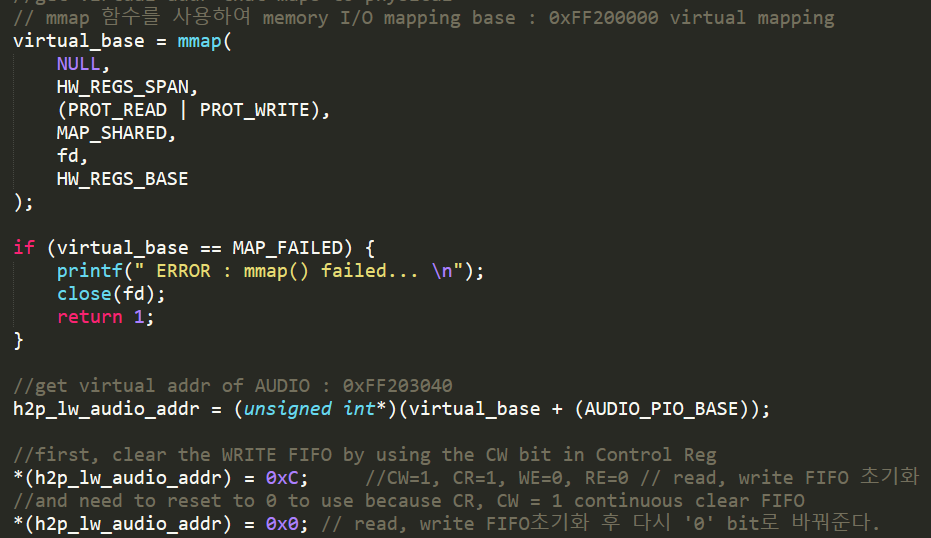
**응용프로그램 test\_audio.c주요 소스 코드**



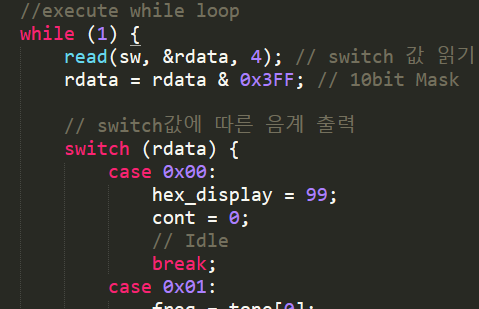
->음계 주파수 배열 선언과 함수 프로토타입 선언, Define을 사용해서 PI, DE1 보드의 Sample rate 등을 선언해주었습니다.



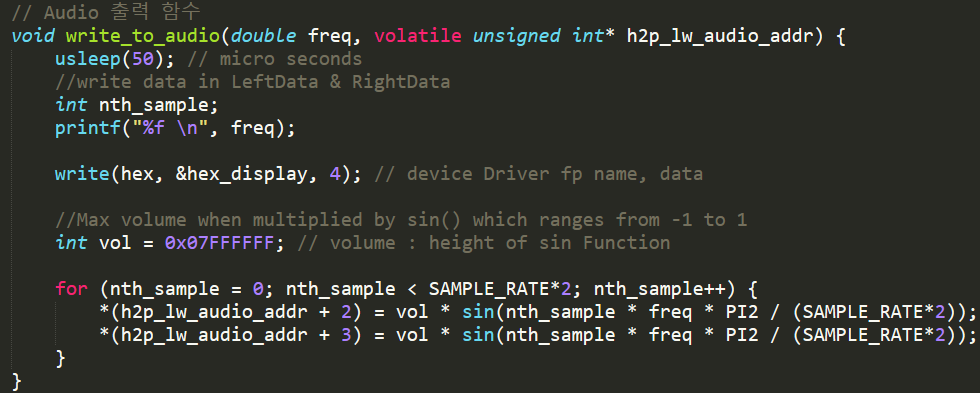
->Switch, hex Device file open & memory open



->virtual\_base에 Physical I/O memory address mapping & Audio control register 주소를 받아서 \*를 사용해서 Control Register에 값을 변경한다. 0xC 후 0x0을 하는 이유는 CW, CR bit를 ‘1’로 만듦으로써 Left, Right FIFO space를 모두 clear 시키게 되고 bit가 ‘1’이면 계속해서 clear를 수행하기 때문에 최초 1번 수행 후 0x0을 넣어서 bit를 ‘0’으로 reset 시켜준다.



while문을 사용하여 Switch 음계를 지속적으로 출력한다.



->Audio 출력을 담당하는 함수이다.

Sin 함수를 사용하여 음계 주파수를 만들어 내고 원리는 다음과 같습니다.

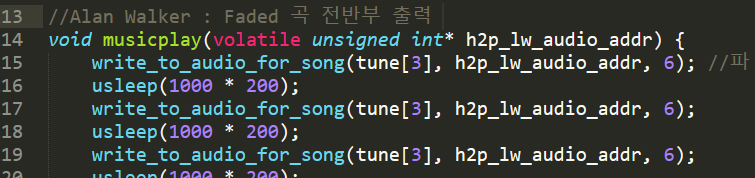
sin 함수를 사용해서 음계 주파수를 만들어내고 공식은 sin(n \* freq \* PI2 / SAMPLE\_RATE)입니다.

sin 함수의 주기는 2PI 이며음계'C'의 경우 262의 Hz를 가지는데 1초에 262개의 sin 파형이 그려져야 합니다.

DE1의 Sample rate는 48000Hz이므로 따라서, 'C'출력을 위해서는 frequency\*PI2/Sample\_Rate

1초에 48000번 Sampling을 수행하며 이 때, sin 함수 262개의 주기가 들어가야 한다.

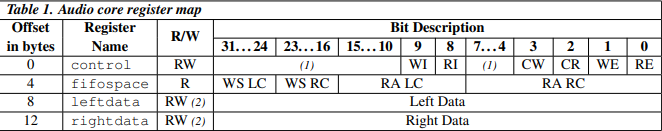
따라서 262\*PI\*2/48000공식을 사용하여 48000번 Sampling 을 하고 나면 1초에 262번의 sin 주기가 진행된다.



->노래 출력 함수

**Audio I/O register 특징 조사/설명**

﻿



\* 0xFF203040 의 주소에 mapping 되어있는 Audio I/O 관련 register

﻿ ﻿

- control register의 각 bit의 의미는 다음과 같습니다.

**RE** : read interrupt enable

**WE** : write interrupt enable

**CR** : Clear both left, right read FIFOs <- read FIFO에 들어있는 모든 값을 clear 시킨다.

**CW** : Clear both left, right write FIFOs <- write FIFO에 들어있는 모든 값을 clear 시킨다.

**RI** : Indicates that a read interrupt is pending

**WI** : Indicates that a write interrupt is pending

﻿

\* CR, CW의 경우 '1'로 set 된 경우 계속해서 FIFO를 Clear시키기 때문에 초기화 할 때만 '1'로 set 하였다가 다시 '0'으로 reset 하는 과정이 필요합니다.

\* 또한 주의할 점으로는 write FIFO에 값을 넣을 때, left와 write FIFO에 모두 값을 넣어주어야 Audio CODEC으로 값이 전달되게 됩니다. 즉, left나 write 중 하나에만 넣으면 아무런 동작도 일어나지 않습니다.

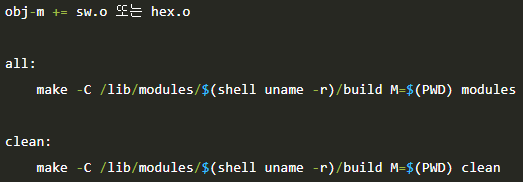
﻿

**fifospace** register의 WSLC(8bit) WSRC(8bit)는 write FIFOs에 남은 공간(빈 공간)을 나타내며 RALC, RARC는 read FIFOs에 data가 들어있는 양(공간)을 나타냅니다. 범위는 0~128까지이며 FIFO가 모두 비어있는 상태일 때 WSLC = WSRC = 128, RALC = RARC = 0 이 됩니다.

interrupt bit인 RE, WE를 '1'로 사용할 때 read FIFO가 75%(96/128) 이상 filled 되면 interrupt가 발생하고 write FIFO가 75%이상 비어있을 때 interrupt가 발생합니다.

﻿

**\* 작성된 Makefile**

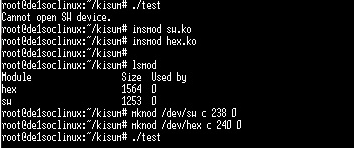
****

**동작 이미지**

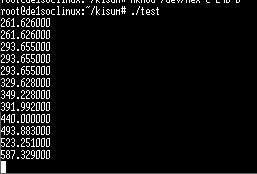
**1.** /kisum directory에 작성된 sw.c, hex.c, test.c(app\_program)과 Makefile

C:\Users\jinwoo\Desktop\임베 프젝\test 실행사진ls.jpg

**2.** Runtime 동안에 Driver를 Kernel Module로 설치하고 사용자모드의 프로그램에서 특수한 파일을 통한 Driver 접근을 위해 sw, hex Device File을 생성한다.



**3.** 스위치가 '1'이 된 음계의 주파수를 소리와 모니터를 통해 출력해주며 계속 '1'로 스위치를 유지시키면 지속적으로 해당 음계가 출력이 된다.



**4.** 솔파미레도(sw4, sw3, sw2, sw1, sw0)을 차례대로 조작하였을 때 다음과 같이 음계의 코드를 출력한다. 1칸씩 좌측으로 shift 되는 형식으로 출력된다. ('도' ~ 2옥타브 '레' 까지 출력가능)



**5.** 가장 좌측 switch(sw9)를 '1'로 set하면 user program에서 작성해둔 Alan Walker의 faded노래가 출력되게 됩니다.

(동영상 파일 별도 첨부)

***동작되는 동영상파일은 별도로 첨부되었습니다.***

***\* math.h는 Floating Point 연산이 사용되는 라이브러리이므로 kernel 프로그래밍에서는 사용할 수 없으며, 사용자 프로그램에서 math.h 라이브러리를 include 했다면 GNU 컴파일 시 '-lm' 명령어를 추가해주어야 한다. ex. cc -o test test.c -lm***

**프로젝트 Reference**

<http://csys.yonsei.ac.kr/lect/embed/DE1-SoC_Computer.pdf>

(DE1 SoC의 Manual)

<https://www.cl.cam.ac.uk/teaching/1920/ECAD+Arch/files/docs/Audio_core.pdf>

(DE1 SoC audio core manual)

<http://www-ug.eecg.toronto.edu/msl/nios_devices/dev_audio.html>

(audio core의 구조도 & Memory Mapped I/O Audio의 control register bit, fifospace register 설명)

<https://software.intel.com/content/www/us/en/develop/topics/fpga-academic/learn/workshops.html>

(Intel FPGA Learn tutorial)

+ GIT

읽어주셔서 감사합니다.