임베디드시스템 설계 및 실험 보고서

[002 분반 - 2 조 - 4 주차]

조원	202055531 김후겸 202055584 이태경 202155540 김채현 202255535 김진우
실험날짜	2024-10-02

1. 실험 주제

- 폴링 방식을 이용한 릴레이 모듈 제어

2. 실험 목적

- 스케터 파일의 이해 및 플래시 프로그래밍
- 릴레이 모듈의 이해 및 임베디드 펌웨어를 통한 동작
- 센싱에서 폴링 방식의 이해

3. 세부 목표

- Datasheet 및 Reference Manual 을 참고하여 해당 레지스터 및 주소에 대한 설정 이해
- Scatter File 의 RAM 과 ROM 주소 설정
- GPIO(general-purpose input/output)를 사용하여 릴레이 모듈 제어

4. 실험 장비

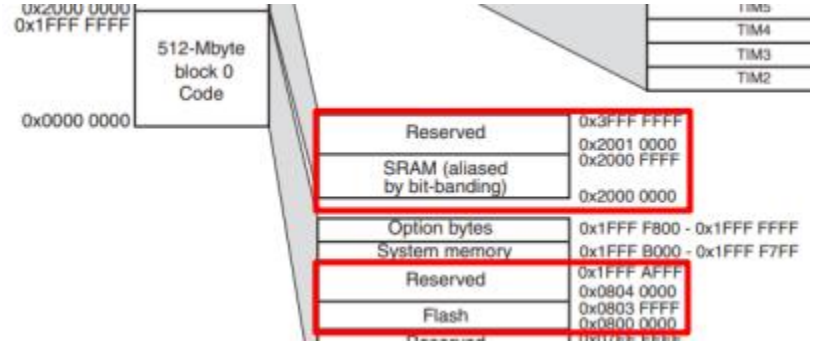
- STM32F107VCT6
- IAR Embedded Workbench (EW)

5. 실험 과정(코드설명)

1) 프로젝트 생성 및 파일 구조 설정

3 주차와 동일

2) ROM 과 RAM 주소 설정



A. ROM 크기 0x80000↵

B. RAM 크기 0x8000↵

[그림 1] - RAM 과 ROM 의 시작 주소 / 주어진 ROM 과 RAM 의 크기

-ROM 의 주소:0x0800 0000 ~ 0x0808 0000

-RAM 의 주소:0x2000 0000 ~ 0x2000 8000

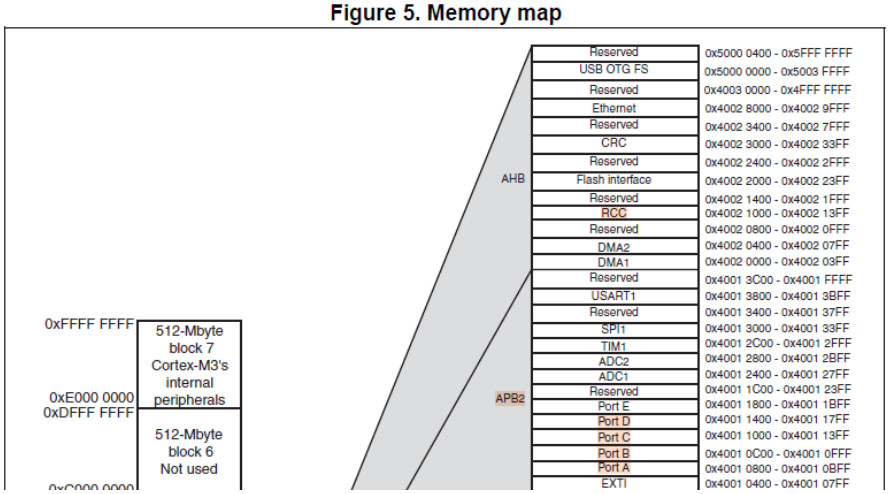
```
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08000000 // TODO
define symbol __ICFEDIT_region_ROM_end__   = 0x08080000 // TODO
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000 // TODO
define symbol __ICFEDIT_region_RAM_end__   = 0x20008000 // TODO
```

[그림 2] - myicf.cf 파일 일부

ROM 과 RAM 의 주소를 수정한 myicf.icf 파일을 project -> Options -> Linker -> Config 로 이동한 뒤, Override default 를 클릭하여 myicf.icf 가 있는 위치로 이동

3) RCC 주소, KEY(1,2,3), 사용하지 않는 GPIO 포트 핀(PC8, PC9) 번호 확인

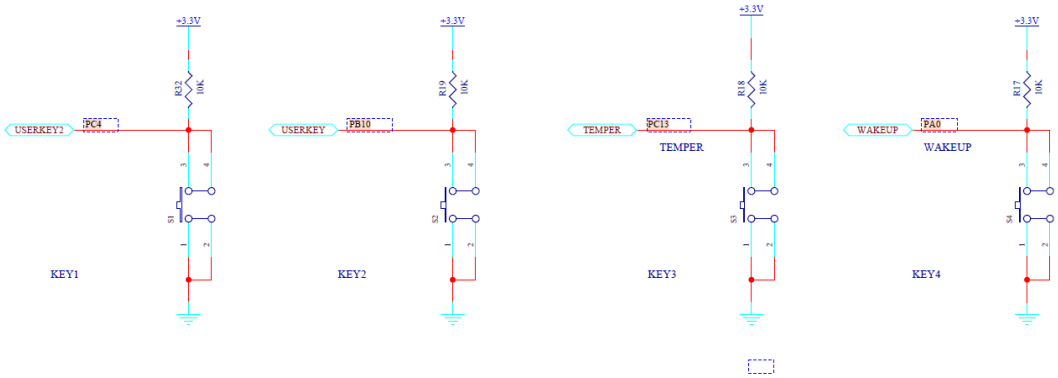
Figure 5. Memory map



[그림 3] - Datasheet

[MemoryMap, baseaddress 확인] : [그림 4]에서 KEY1,2,3 을 위해 Port B(10),C(4,13)가 사용됨을 알 수 있음

- Port B:0x4001 0C00
- Port C:0x4001 1000
- RCC:0x4002 1000



[그림 4] - Schematic

4) main 함수 작성하기

```
#define RCC_APB2ENR (*(volatile unsigned int *)0x40021018)

#define GPIOA_CRL (*(volatile unsigned int *)0x40010800)
#define GPIOA_IDR (*(volatile unsigned int *)0x40010808)

#define GPIOB_CRH (*(volatile unsigned int *)0x40010C04)
#define GPIOB_IDR (*(volatile unsigned int *)0x40010C08)

#define GPIOC_CRL (*(volatile unsigned int *)0x40011000)
#define GPIOC_CRH (*(volatile unsigned int *)0x40011004)
#define GPIOC_IDR (*(volatile unsigned int *)0x40011008)
#define GPIOC_BSRR (*(volatile unsigned int *)0x40011010)

#define GPIOD_CRL (*(volatile unsigned int *)0x40011400)
#define GPIOD_BSRR (*(volatile unsigned int *)0x40011410)
```

[그림 5] - main.c 의 매크로 정의

3 주차와 동일한 방식으로 APB2 peripheral clock enable register 와 GPIO register 들의 주소를 상수로 정의(0..7=>register low, 8..15=>register high)

-RCC_APB2ENR: APB2 버스에 연결된 장치들에 대한 클럭을 활성화하는 레지스

-GPIOx_CRL/CRH: GPIO 포트의 핀 모드를 설정하는 레지스터

-GPIOx_IDR: 포트의 입력 상태를 읽는 레지스터

-GPIOx_BSRR: 포트의 특정 핀을 설정(set) 또는 리셋(reset)하는 데 사용

```
void delay() {
  int i;
  for(i=0; i<10000000; i++) {}
}
```

[그림 5] - main.c 의 delay() 함수

이 함수를 활용하여 약간의 지연을 발생시킴

```
int main(void)
{

    // Key 1 PC4
    GPIOC_CRL &= 0xFFF0FFFF;
    GPIOC_CRL |= 0x00080000;
    // Key 2 PB10
    GPIOB_CRH &= 0xFFFFF0FF;
    GPIOB_CRH |= 0x00000800;
    // Key 3 PC13
    GPIOC_CRH &= 0xFF0FFFFF;
    GPIOC_CRH |= 0x00800000;


    // clock
    // PORT A, B, C, D ON
    RCC_APB2ENR |= 0x3C;

    // Relay Module
    // PC8, PC9
    GPIOC_CRH &= 0xFFFFFF00;
    GPIOC_CRH |= 0x00000033;
```

[그림 5] - main.c 의 main() 함수

사용하려는 Port 의 pin 들의 값을 0 으로 초기화 한 후 설정

In input mode (MODE[1:0]=00):
00: Analog mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved
In output mode (MODE[1:0] > 00):
00: General purpose output push-pull
01: General purpose output Open-drain
10: Alternate function output Push-pull
11: Alternate function output Open-drain

[그림 5] - main.c 의 main() 함수

-0x8(0b1000): Input with pull-up / pull-down

-0x3(0b0011): Output mode

Bit 6 **IOPEEN**: I/O port E clock enable
Set and cleared by software.
0: I/O port E clock disabled
1: I/O port E clock enabled

Bit 5 **IOPDEN**: I/O port D clock enable
Set and cleared by software.
0: I/O port D clock disabled
1: I/O port D clock enabled

Bit 4 **IOPCEN**: I/O port C clock enable
Set and cleared by software.
0: I/O port C clock disabled
1:I/O port C clock enabled

Bit 3 **IOPBEN**: I/O port B clock enable
Set and cleared by software.
0: I/O port B clock disabled
1:I/O port B clock enabled

Bit 2 **IOPAEN**: I/O port A clock enable
Set and cleared by software.
0: I/O port A clock disabled
1:I/O port A clock enabled

[그림 5] - reference manual 의 APB2 peripheral clock enable register (RCC_APB2ENR)

[그림 5]를 참고하여 Port A,B,C,D 는 0b111100(0x3x)의 값을 RCC_APB2ENR 에 넣음으로써 clock enable 설정

```
while (1)
{
    if ((GPIOC_IDR & 0x10) == 0) // Key1
    {
        GPIOC_BSRR |= 0x300; // PC8 set, PC9 set
        delay();
        GPIOC_BSRR |= 0x3000000; // PC8 reset, PC9 reset
    }

    else if ((GPIOB_IDR & 0x0400) == 0) // Key2
    {
        GPIOC_BSRR |= 0x2000100; // PC8 set, PC9 reset
        delay();
        GPIOC_BSRR |= 0x3000000; // PC8 reset, PC9 reset
    }

    else if ((GPIOC_IDR & 0x2000) == 0) // key 3
    {
        GPIOC_BSRR |= 0x1000200; // PC8 reset, PC9 set
        delay();
        GPIOC_BSRR |= 0x3000000; // PC8 reset, PC9 reset
    }
}
```

[그림 5] - main.c 의 main() 함수 중 반복문

<미션>은 아래와 같다.

[버튼 1(KEY1) : 1 번 및 2 번 모터 1 초 회전 후 정지

버튼 2(KEY2) : 1 번 모터 1 초 회전 후 정지

버튼 3(KEY3) : 2 번 모터 1 초 회전 후 정지]

이를 수행하기 위해

Key1 이 입력되면 PC8 과 PC9 모두 set 으로 설정 -> 일정 시간 지난 뒤 reset

Key2 가 입력되면 PC8 은 set, PC9 는 reset 으로 설정 -> 시간 지난 뒤 둘 다 reset

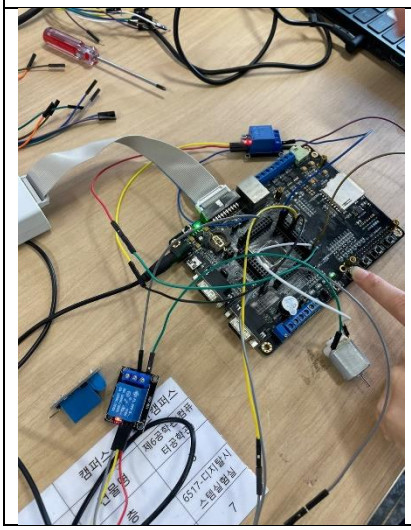
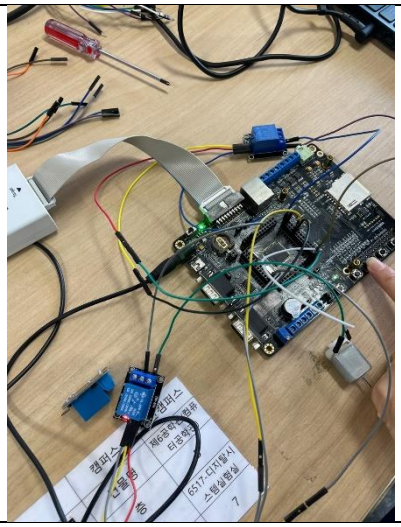
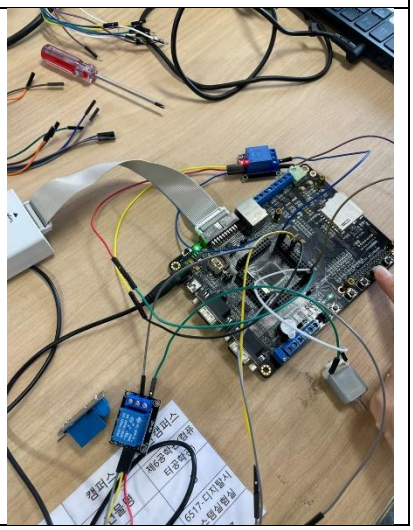
Key3 이 입력되면 PC8 은 reset, PC9 는 set 으로 설정 -> 일정 시간 지난 뒤 둘 다 reset

하도록 코드를 작성하였다

5) 회로 연결하기

PC8, 9 포트를 사용하는 것으로 코드 작성을 했기 때문에, PC8, 9 에 연결을 해준다.

5. 실험 결과

KEY1	KEY2	KEY3
		

6. 분석 및 결론

- 4 주차 실험은 scatter file 을 통한 RAM 과 ROM 의 주소 설정과 relay module 을 제어하는 것이었다. 이번 실험을 통해 scatter file 을 설정하여 원하는 위치에 원하는 크기만큼 할당 받을 수 있게 되었다. 또한, relay 모듈을 직접 연결하고 제어하는 방법을 깨달을 수 있었다. 폴링을 사용하는 방식과 인터럽트를 사용하는 방식의 장단점을 알아볼 수 있는 시간이었다.

- 회로를 연결할 때 모터의 한쪽에 GND 를 연결해야 한다는 점을 인지하지 못했고, 디버깅도 되지 않아 작동하지 않은 이유가 코드때문인지 회로때문인지 몰랐지만, 처음부터 천천히 생각을 해서 코드와 회로를 정리한 후 실행하니 성공했다.

7. 참고자료

- stm32_Datasheet.pdf
- stm32_ReferenceManual.pdf
- STM32107VCT6_Schematic.pdf