임베디드시스템
실험 및 설계
3주차 실험보고서

2조 노윤정, 박건우, 이동근, 조영진

1. 실험목표

1) 임베디드 시스템 설계의 기본 원리 습득

2) 디버깅 툴 사용방법 습득 및 레지스터 제어를 통한 임베디드 펌웨어 개발

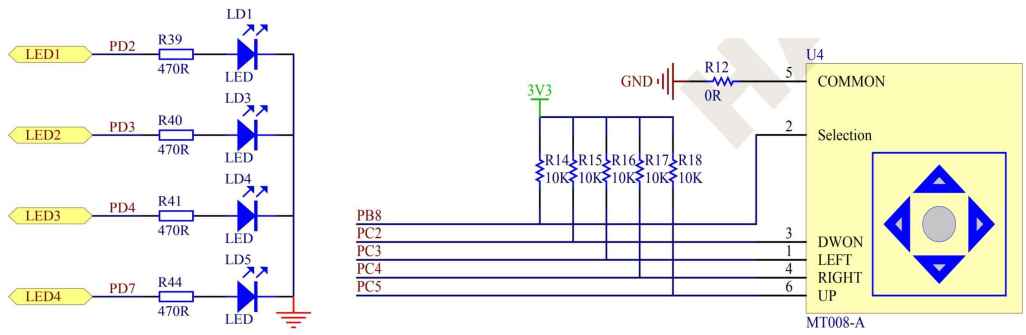
3) Datasheet 및 Reference Manual을 참고해 해당 레지스터 및 주소에 대한 설정 이해

4) GPIO(general-purpose input/output)를 사용하여 조이스틱과 LED 제어

2. 실험과정

1) 사용할 GPIO 목록

LED		Joystick	
1	Port D, pin 2	Down	Port C, pin 2
2	Port D, pin 3	Left	Port C, pin 3
3	Port D, pin 4	Right	Port C, pin 4
4	Port D, pin 7	Up	Port C, pin 5



2) 사용할 GPIO register address 정의

- RCC_APB2ENR: peripheral clock enable register
- CRL: Port configuration register low
- BSRR: Port bit set/reset register
- BRR: Port bit reset register
- IDR: Port input data register

Base Address		Register Offset	
RCC	0x4002 1000	RCC_ABP2ENR	0x18
Port C	0x4001 1000	CRL	0x00
Port D	0x4001 1400	BSRR	0x10
		BRR	0x14
		IDR	0x08

- Register Address: (register address) = (base address) + (register offset)

```
3   #define RCC_BASE_ADDR 0x40021000
4   #define RCC_OFFSET 0x18
5   #define RCC_ADDR (*(volatile unsigned int*)(RCC_BASE_ADDR + RCC_OFFSET))
6
7   #define GPIOD_BASE_ADDR 0x40011400
8   #define GPIOD_CRL *(volatile unsigned int*) (GPIOD_BASE_ADDR + 0x00)
9   #define GPIOD_CRH *(volatile unsigned int*) (GPIOD_BASE_ADDR + 0x4)
10  #define GPIOD_BSRR *(volatile unsigned int*) (GPIOD_BASE_ADDR +0x10)
11  #define GPIOD_BRR *(volatile unsigned int*) (GPIOD_BASE_ADDR + 0x14)
12
13  #define GPIOC_BASE_ADDR 0x40011000
14  #define GPIOC_CRL *(volatile unsigned int*) (GPIOC_BASE_ADDR + 0x00)
15  #define GPIOC_IDR *(volatile unsigned int*) (GPIOC_BASE_ADDR + 0x08)
```

3) RCC를 사용하여 GPIO에 clock 인가 (peripheral clock enable)

- GPIO clock 인가는 RCC_APB2ENR에서 사용할 Port에 대응하는 bit 값을 설정
- 사용할 Port C, D는 RCC_APB2ENR의 0~31 bit 중의 4, 5번째에 대응
- 0b00...0011 0000 = 0x30

```
17  int main(void) {
18      RCC_ADDR |= 0x30;
```

4) 사용하려는 GPIO Port, Pin의 input/output mode 설정 (Port Configuration)

- 각 Port의 0~7번 pin의 모드는 Port Configuration Low Register로 설정
- 사용할 pin에 해당하는 자리의 bit에 값을 대입해 초기화 및 mode 설정

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Port C (pin 2~5)

- pin 2~5 초기화: 0xFF0000FF = ~0x00FFFF00
- input mode, pull-up/down mode: 0b1000 (0x8)
- pin 2~5 mode 설정: 0x00888800

Port D (pin 2~4, 7)

- pin 2~4, 7 초기화: 0x0FF000FF = ~0xF00FFF00
- output mode, general push-pull mode: 0b0011 (0x3)
- pin 2~4, 7 mode 설정: 0x30033300

```
24      GPIOC_CRL &= ~0x00FFFF00;
25      GPIOC_CRL |= 0x00888800;
26
27      GPIOD_CRL &= ~0xF00FFF00;
28      GPIOD_CRL |= 0x30033300;
```

5) GPIO Port, Pin Register의 입출력 제어

Port C (조이스틱) 입력 제어

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

- Down(pin 2), Left(pin 3) Right(pin 4), Up(pin 5) 동작에 해당하는 값 설정

Joystick	Pin Number	IDR value
Down	2	0x4
Left	3	0x8
Right	4	0x10
Up	5	0x20

Port D (LED) 출력 제어

- BRR(Port bit reset register) 이용해 LED Off
- bit 값으로 1을 주면 Off, 0을 주면 상태 유지

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

-BSRR(Port bit set reset register) 이용해 LED On

- bit 값으로 1을 주면 On, 0을 주면 상태 유지

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

- LED1(pin 2), LED2(pin 3), LED3(pin 4), LED4(pin 7)에 해당하는 bit 값 설정

Joystick	LED	pin	Register value
Down/Right	1, 2	2, 3	0xC
Up/Left	3, 4	4, 7	0x90

- 조이스틱 입력 전체 초기화 및 LED 출력 전체 Off

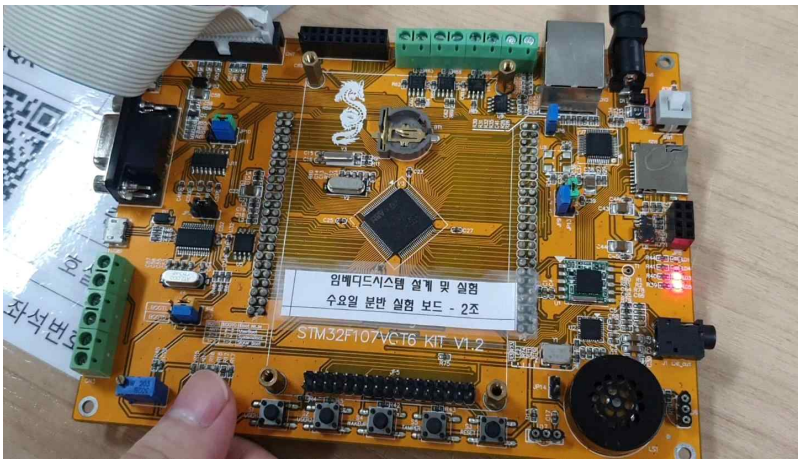
```
34    GPIOC_IDR &= 0x00000000;
35    GPIOD_BRR |= 0x9C;
```

- 조이스틱 입력(GPIOC_IDR) 읽어 LED 출력하는 반복문
- 입력값이 원하는 값과 일치하는지 '&' 연산자로 검사하는 조건문
- 조이스틱 입력은 VCC와 연결되어 있으므로 '!' 연산자를 사용해 반대로 설정
- LED를 켤 때는 BSRR, 끌 때는 BRR에 해당하는 pin의 값 대입

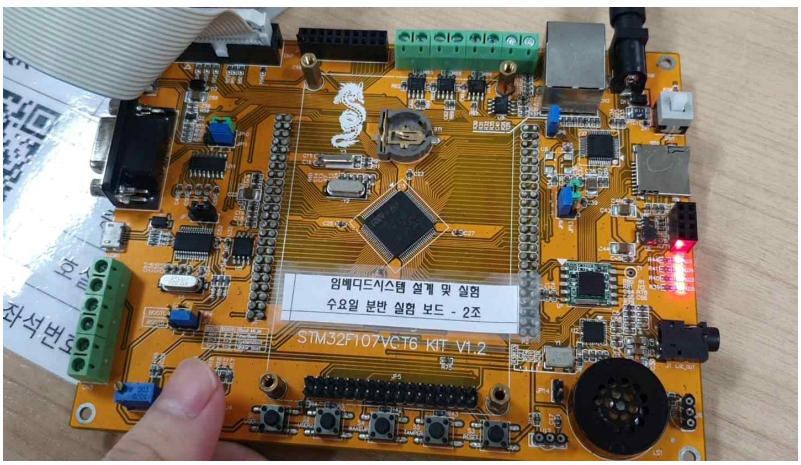
```
37      while(1) {
38        if(!(GPIOC_IDR&0x4)) GPIOD_BSRR |= 0xC;
39        else if(!(GPIOC_IDR&0x8)) GPIOD_BRR |= 0x90;
40        else if(!(GPIOC_IDR&0x10)) GPIOD_BRR |= 0xC;
41        else if(!(GPIOC_IDR&0x20)) GPIOD_BSRR |= 0x90;
42      }
43      return 0;
```

3. 실험결과

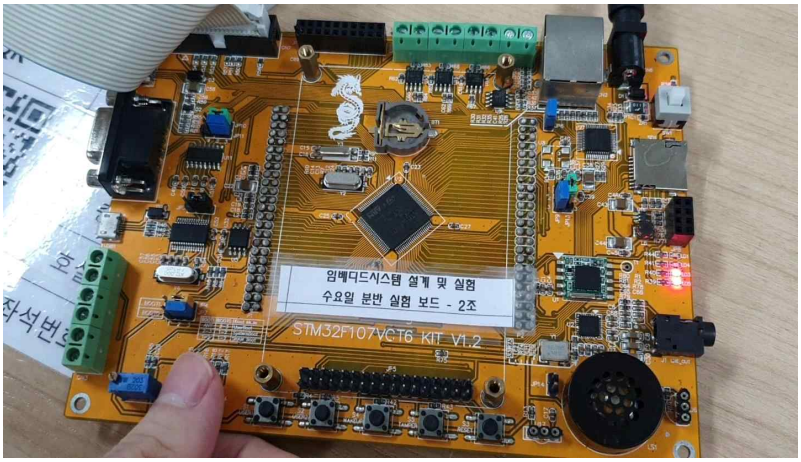
1) 조이스틱을 아래로(Down) 움직이면 LED 1, 2가 켜진다.



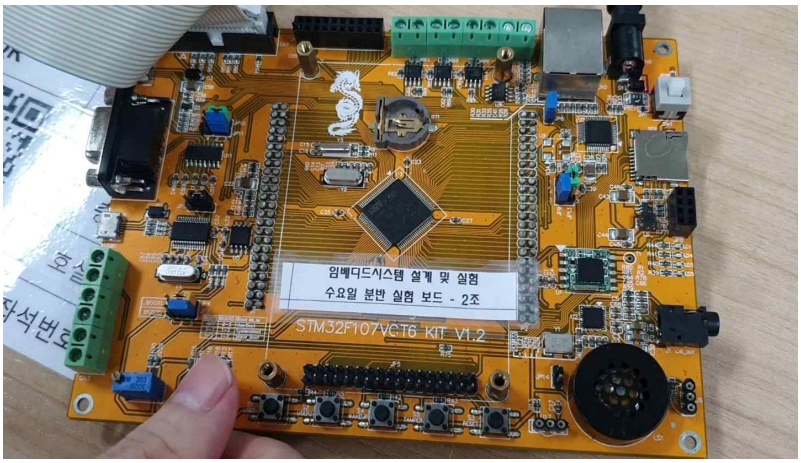
2) 조이스틱을 위로(Up) 움직이면 LED 3, 4가 켜진다.



3) 조이스틱을 왼쪽으로(Left) 움직이면 LED 3, 4가 꺼진다.



4) 조이스틱을 오른쪽(Right)으로 움직이면 LED 1, 2가 꺼진다.



4. 고찰

소스코드를 작성하며 레지스터에 값을 대입할 때 부적절한 대입 연산자를 사용한 채로 컴파일했다. 레지스터를 제어하는 소스코드에서 모든 연산은 비트 연산으로 이루어지기 때문에 그에 맞춰 비트 연산자를 사용해야 한다. 그러나 이를 알아차리지 못하고 '=' 연산자를 사용해 디버깅이 끝까지 진행되지 못하고 중단되었다. 대입 연산자를 모두 비트 연산자인 '|='나 '&=' 연산자로 수정하고 난 뒤에야 소스코드가 정상적으로 실행되었다. 보드를 코딩할 때 수행하는 연산의 종류를 정확히 파악하지 못한 실수였다.

조이스틱의 입력값을 검사할 때 조이스틱을 아무리 움직여도 조건문의 결과가 예상했던 참(1)으로 나오지 않았다. 조이스틱에는 전압이 인가되기 때문에 입력 기본값이 1로 유지되며, 동작을 취할 때 새로운 입력값으로 0이 들어온다. 스키메틱에서 이를 확인하지 못한 채 조건문을 반대로 작성해버려 입력이 인식되지 않

은 것이다. 오류를 발견하고 조건문 전체에 '!' 연산자를 이용해 식의 결과를 반대로 바꿔주자 조이스틱의 입력이 정상적으로 인식되고 올바른 결과가 나왔다.

3주차 실험을 전체적으로 되돌아보면 어렵기보다는 사소한 부분을 제대로 확인하지 못해 시행착오를 겪었다. 연산자 종류나 스키메틱의 전압 인가 여부와 같이 기본적인 요소를 챙기지 못한 부주의로 인해 실험이 수월히 진행되지 못했던 것 같다. 이를 반성하고 앞으로의 실험에서는 기본적인 것들을 꼼꼼히 확인해 같은 문제가 반복되지 않도록 하겠다.