

# 임베디드시스템 설계 및 실험 보고서

[002 분반 - 2 조 - 3 주차]



조원	202055531 김후겸 202055584 이태경 202155540 김채현 202255535 김진우
실험날짜	2024-09-25

## 1. 실험 주제

- GPIO 조작

## 2. 실험 목적

- 임베디드 시스템의 기본 원리 습득
- 레지스터와 주소 제어를 통한 임베디드 펌웨어 개발 이해

## 3. 세부 목표

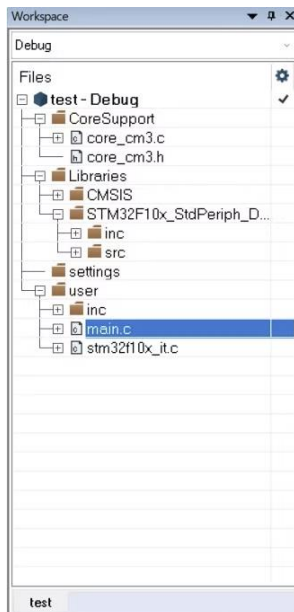
- 개발 환경 구축
- IAR Embedded Workbench 에서 프로젝트 생성 및 설정
- Datasheet 및 Reference Manual 을 참고하여 해당 레지스터 및 주소에 대한 설정 이해
- GPIO(general-purpose input/output)를 사용하여 LED 제어

## 4. 실험 장비

- STM32F107VCT6
- IAR Embedded Workbench (EW)

## 5. 실험 과정(코드설명)

### 1) 프로젝트 생성 및 파일 구조 설정



[그림 1]

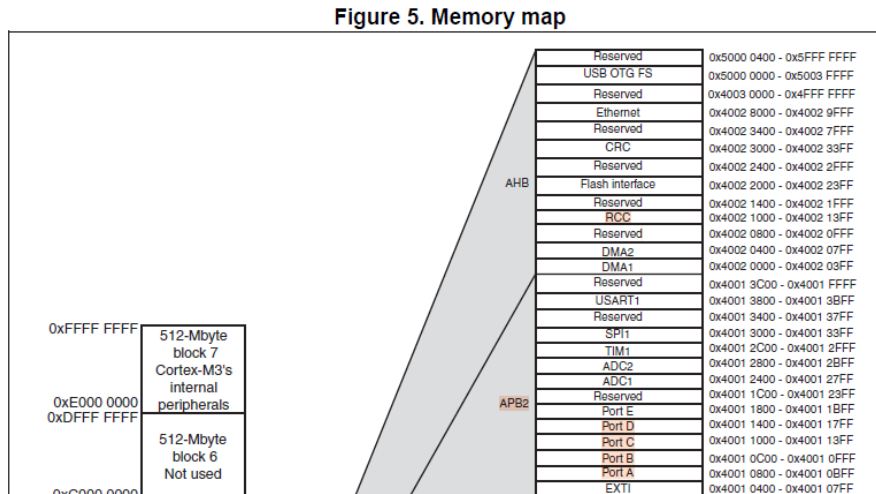
[그림 1]과 같이 폴더 구조를 만들고 각종 설정을 한다

추가적인 주의사항은 다음과 같다.

동작중 케이블을 뽑지 말아야하고, 보드는 전원으로 usbport 나 어댑터(5V, 1A)를 사용해야 한다.

디버깅 모드 중 보드에 전원을 끄거나 연결된 케이블을 분리하지 말아야 한다.

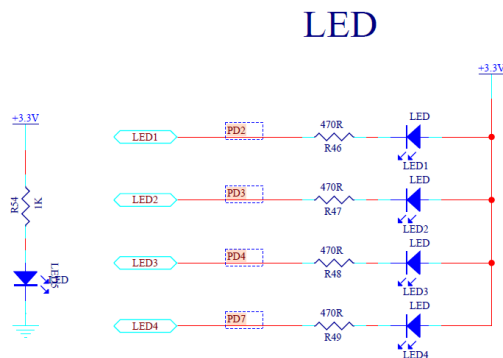
## 2) RCC 주소, LED, KEY 의 GPIO 포트 핀 번호 확인



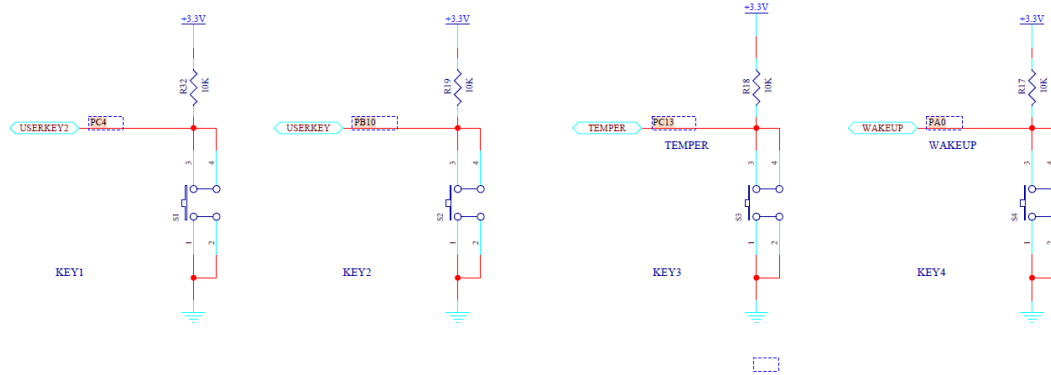
[그림 2] - Datasheet

[MemoryMap, baseaddress 확인] : [그림 3,4]에서 Port A,B,C,D 가 사용됨을 알 수 있음

- Port A:0x4001 0800
- Port B:0x4001 0C00
- Port C:0x4001 1000
- Port D:0x4001 1400
- RCC:0x4002 1000



[그림 3] - Schematic : LED 의 경우 모두 Port D 를 사용한다



## Button

[그림 4] - Schematic

### 3) 각종 offset 확인

Volatile 선언된 변수는 컴파일러가 해당 volatile 변수를 최적화에서 제외하게 한다. 즉, Volatile 변수를 참조하면 레지스터의 로드된 값을 사용하지 않고 메모리를 참조한다.

#### 9.2.1 Port configuration register low (GPIOx\_CRL) (x=A..G)

Address offset: 0x00

Reset value: 0x4444 4444

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

[그림 5] - Reference

#### 9.2.2 Port configuration register high (GPIOx\_CRH) (x=A..G)

Address offset: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[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
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

[그림 6] - Reference

[그림 5]와 [그림 6]을 보면 포트 번호를 지정해주기 위한 offset 이 나와있다. 포트번호 0~7 번까지 [그림 5]의 Port configuration register low 의 사용하며 8~15 번까지는 [그림 6]의 Port configuration register high 을 사용한다. 따라서 KEY1, KEY4 는 offset 0x00 을 KEY2, KEY3 은 offset 0x04 을 사용하여 값을 지정해준다.([그림 7] 참고)

```
#define GPIOA_CRL (*(volatile unsigned int *)0x40010800)
#define GPIOB_CRH (*(volatile unsigned int *)0x40010C04)
#define GPIOC_CRL (*(volatile unsigned int *)0x40011000)
#define GPIOC_CRH (*(volatile unsigned int *)0x40011004)
#define GPIOD_CRL (*(volatile unsigned int *)0x40011400)
```

[그림 7] : C 의 경우 C4, C13 이 사용되기 때문에 low, high 둘 다 필요하다

### 9.2.3 Port input data register (GPIOx\_IDR) (x=A..G)

Address offset: 0x08h

Reset value: 0x0000 XXXX

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

[그림 8] : input data register 의 offset 은 0x08 이다. 따라서 각 KEY 들이 사용하는 Port 의 시작 주소에 0x08 을 더해서 IDR 를 선언한다.([그림 9]참고)

```
#define GPIOA_IDR (*(volatile unsigned int *) 0x40010808)
#define GPIOB_IDR (*(volatile unsigned int *)0x40010C08)
#define GPIOC_IDR (*(volatile unsigned int *)0x40011008)
```

[그림 9] : Port D 의 경우는 LED 인데, LED 는 input 이 아니기 때문에 선언할 필요가 없다.

### 9.2.5 Port bit set/reset register (GPIOx\_BSRR) (x=A..G)

Address offset: 0x10

Reset value: 0x0000 0000

[그림 10]

### 9.2.6 Port bit reset register (GPIOx\_BRR) (x=A..G)

Address offset: 0x14

Reset value: 0x0000 0000

[그림 11] : BRR 이 LED 를 킬 때 사용

BSRR 과 BRR 은 LED 를 끄고 켤 때 사용되기 때문에 Port D 에 관해서 설정을 하고, 주어진 offset 만큼 더해서 선언한다.([그림 12] 참고)

```
#define GPIOD_BSRR (*(volatile unsigned int *)0x40011410)
#define GPIOD_BRR (*(volatile unsigned int *)0x40011414)
```

[그림 12]

### 3) main 함수 작성하기

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

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

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

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

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

Bit 1 Reserved, must be kept at reset value.

Bit 0 **AFIOEN**: Alternate function IO clock enable  
Set and cleared by software.  
0: Alternate Function IO clock disabled  
1: Alternate Function IO clock enabled

[그림 13] - Reference

우리가 제어할 레지스터의 base 주소에 clock enable register 의 offset 값을 더해줌으로서 레지스터를 활성화 가능하다.

우리가 제어할 레지스터는 RCC 에 있으므로, RCC base address + clock enable offset 으로 레지스터에 접근할 수 있다.

우리는 Port A, B, C, D 가 필요하기 때문에 2, 3, 4, 5 번째 bit 에 1 을 인가해주면 포트들이 활성화된다.

[그림 10]에서 보듯 bit 2, 3, 4, 5 를 1 로 변경해야 하고, 이를 16 진수로 나타내면 0x3C 이다.

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

[그림 14]

[그림 5]와 [그림 6]에서 보듯 configuration register 의 reset value 가 0x44444444 이므로 각 KEY 들과 LED reset value 값으로 초기화하고, 포트번호를 지정해준다([그림 15]참고)

```
// Key 1 PC4
GPIOC_CRL &= 0xFFF0FFFF;
GPIOC_CRL |= 0x00080000;
// Key 2 PB10
GPIOB_CRH &= 0xFFFF00FF;
GPIOB_CRH |= 0x00000800;
// Key 3 PC13
GPIOC_CRH &= 0xFF0FFFFF; // LED
GPIOC_CRH |= 0x00800000; // PD2, PD3, PD4, PD7
// Key 4 PA0
GPIOA_CRL &= 0xFFFFFFF0; GPIOD_CRL &= 0xFF0000FF;
GPIOA_CRL |= 0x00000008; GPIOD_CRL |= 0x30033300;
GPIOA_CRL |= 0x00000008; GPIOD_BSRR |= 0x9C; // LED1,2,3,4 OFF
```

[그림 15]

### While loop 로직 설명





- 반복문 내에서 조건에 따라 LED 가 켜지고 꺼진다. BRR 혹은 BSRR 의 값을 적절히 수정하여 LED 를 켜다 끈다.

- 각 key 들이 해당하는 핀 번호에 맞는 값이 들어오면 인식해서 해당하는 LED 번호를 켜도록 하였다.

```
while(1){
if((GPIOC_IDR & 0x10) == 0){ //Key1
// GPIOD_BSRR |= 0xC0000; // PD2, PD3 ON
GPIOD_BRR |= 0x84; // LED1,4 ON
}
else if((GPIOB_IDR & 0x0400) == 0){ // Key2
GPIOD_BSRR |= 0x84; // LED1,4 OFF
}
else if((GPIOC_IDR & 0x2000) == 0){ //key 3
// GPIOD_BSRR |= 0x900000; // PD4, PD7 ON
GPIOD_BRR |= 0x18; // LED2,3 ON
}
else if((GPIOA_IDR & 0x01) == 0){ //key 4
GPIOD_BSRR |= 0x18; // LED2,3 OFF
}
}
```

[그림 16]

## 5. 실험 결과

KEY1 LED1,4 ON	KEY2 LED1,4 OFF	KEY3 LED2,3 ON	KEY1 LED2,3 OFF
			

## 6. 분석 및 결론

- 3 주차 실험을 통해 개발 환경을 구축하는 방법, IAR Embedded Workbench 의 사용법, Datasheet 와 Reference Manual 로부터 필요한 정보를 찾는 방법 등에 대해 알게 되었다. Datasheet 와 Reference Manual 을 읽고 분석하는 것이 처음에는 어려웠고 코드를 짜는 것에 있어 많은 어려움이 있었지만, 몇 번 시도하고 방법을 이해하고 난 후에 성공하게 되었다. 또한 버튼을 통해 여러 개의 LED 를 제어하는 코드를 직접 구현하고 실제로 보드에 포팅 후 동작할 때는 신기하고 성취감을 느꼈다. 비록 다른 조에 비해 시간이 오래 걸리긴 했지만, 조원들과 함께 의견을 나누고 문제점을 해결해 나가며 [임베디드시스템 설계 및 실험]이라는 과목의 기본을 확실히 이해한 것 같다.

- 임베디드 시스템에서는 레지스터와 메모리 매핑 주소를 사용하여 데이터를 조작한다. 이때 각 장치나 포트는 고유한 메모리 주소를 가지고 있으며, 레지스터를 통해 데이터를 주고받는다. 따라서 해당 보드에서 레지스터와 주소가 어떻게 구현되어 있는지 명확히 이해하는 것이 필요하며, Reference Manual 및 Datasheet 에 필요한 대부분의 정보가 기재 되어있기 때문에 앞으로 이들을 자세히 읽는 것이 실험의 성공 여부에 큰 영향을 미칠 것 같다.

## 7. 참고자료

- stm32\_Datasheet.pdf
- stm32\_ReferenceManual.pdf
- STM32107VCT6\_Schematic.pdf