임베디드 시스템 설계 및 실험
화요일 분반 4주차 실험 결과 보고서

조 : 6조

조원 : 이주승 김선규 이동현 이지현 최세희

실험 목표

1. 스캐터 파일의 이해 및 플래시 프로그래밍
2. 릴레이 모듈의 이해 및 임베디드 펌웨어를 통한 동작
3. 폴링 방식의 이해

세부 목표

1. Datasheet 및 Reference Manual을 참고하여 해당 레지스터 및 주소에 대한 설정 이해
2. 스캐터 파일을 통해 플래시 메모리에 프로그램 다운로드
3. 플래시 메모리에 올려진 프로그램 정상적인 동작 확인

실험 결과

1. 레지스터 선언 및 설정

```c
#include "stm32f10x.h"

#define RCC_APB2_ENR *(volatile unsigned int *)0x40021018
#define GPIOC_CRL *(volatile unsigned int *)0x40011000
#define GPIOE_CRL *(volatile unsigned int *)0x40011800
#define GPIOE_BSRR *(volatile unsigned int *)0x40011810
#define GPIOC_IDR *(volatile unsigned int *)0x40011008
```

```c
int main() {
  RCC_APB2_ENR = 0x00000050; // IO Port C, E Enable

  GPIOC_CRL = 0x44444444; // GPIO C Reset
  GPIOE_CRL = 0x44444444; // GPIO E Reset

  GPIOC_CRL = 0x44844844; // PC2, PC5 Set
  GPIOE_CRL = 0x44444344; // PE2 Set
  GPIOE_BSRR = 0x00000000; // Port bit Reset
```

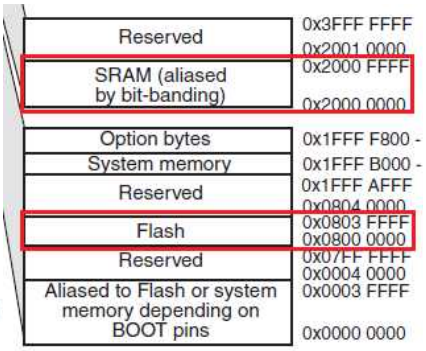
- 3주차에서의 레지스터 및 주소 설정 방식과 동일하게 각 base 주소에 offset을 더한 주소를 구하여 선언해준 뒤, 사용할 GPIO Port를 Enable 상태로 초기화한다.

- Schematic에서 사용할 핀의 번호를 확인한다. <GPIO C – 2, 5번 / GPIO E – 2번> 이므로 각각 CRL 레지스터를 사용해 값을 설정해준다.

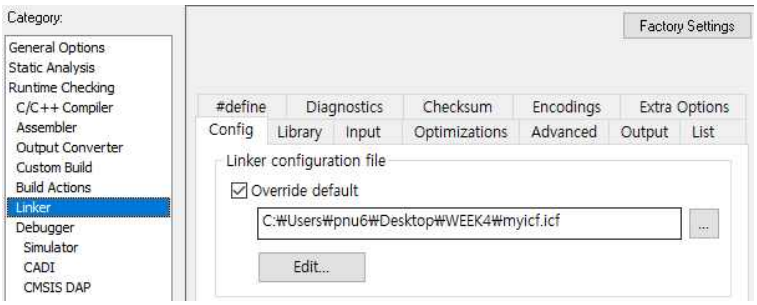
- Output으로 사용될 GPIO E는 BSRR 레지스터를 사용해 각 비트에 Set과 Reset을 설정하여야 하므로 선언한 GPIOE_BSRR에 초깃값을 설정해준다.

2. 스캐터 파일 설정

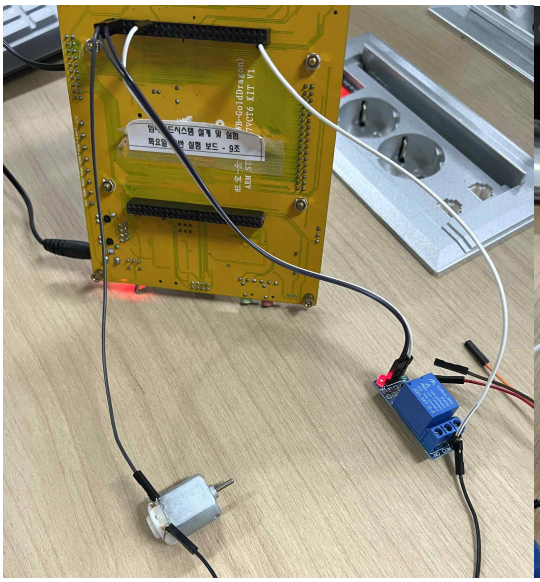
```
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08000000; // TODO
define symbol __ICFEDIT_region_ROM_end__   = 0x8080000; // TODO
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000; // TODO
define symbol __ICFEDIT_region_RAM_end__   = 0x20008000; // TODO
```

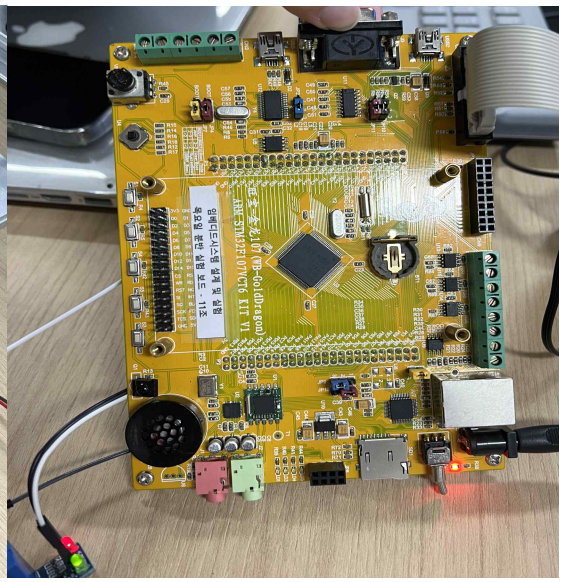


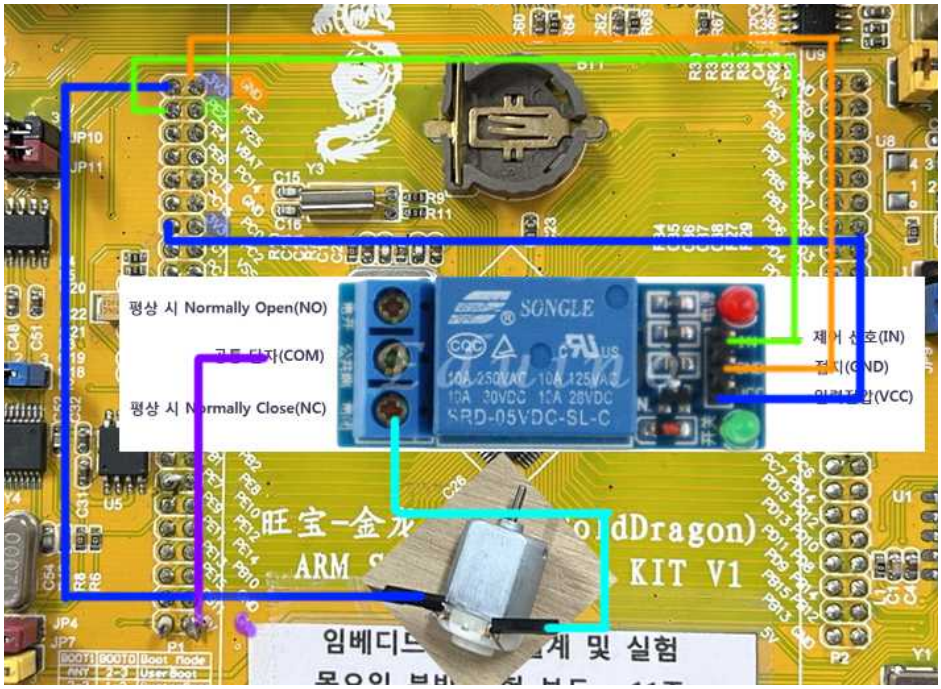
- Datasheet를 참고해 플래시 메모리(ROM)과 RAM의 주소를 설정한다. 각각의 시작주소에서 지정된 크기(ROM-0x80000, ram-0x8000)만큼 더한 값을 end 주소에 작성한다.



3. 회로 구성







=> 보드에는 사진과 같이 회로를 구성하였다. 릴레이 모듈에는 3.3v 전원을 인가하고, 제어신호(IN)에는 보드의 Output으로 나올 값을 넣어준다. (PE2 핀에 해당) 평소에는 close인데, high 신호가 들어오면 open 되는 NC와 모터를 연결시킨다.

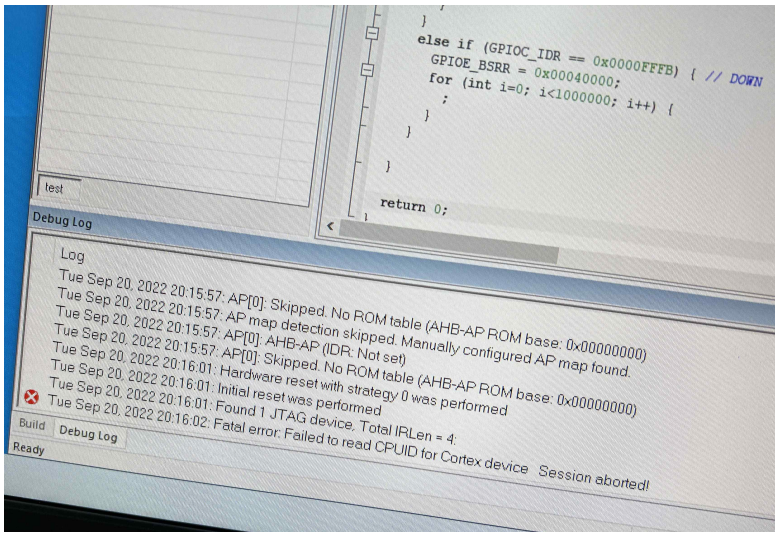
4. 동작

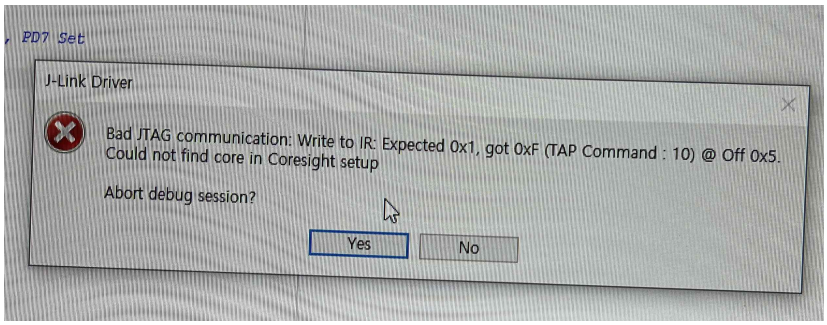
```
while (1) {

  if (GPIOC_IDR == 0x0000FFDF) { // UP
    GPIOE_BSRR = 0x00000004;
    for (int i=0; i<1000000; i++) {
      ;
    }
  }

  else if (GPIOC_IDR == 0x0000FFFB) { // DOWN
    GPIOE_BSRR = 0x00040000;
    for (int i=0; i<1000000; i++) {
      ;
    }
  }
```

- **조이스틱 UP: 모터 회전(전진)**
  GPIOC_IDR로 GPIO C의 Input 값을 확인할 수 있다. Pull-up 저항을 사용하고 있으므로 초깃값은 0x0000FFFF로 설정한다. Input이 들어온 Bit는 0이 조이스틱 UP에 해당하는 PC5가 Input으로 들어왔다면 5번 Bit가 0이 될 것이므로 GPIOC_IDR의 값은 0x0000FFDF이다.
  이 조건을 만족할 때 모터가 회전해야 하므로, Output에 해당하는 GPIOE_BSRR의 값을 설정해준다. 릴레이 모듈의 제어신호(IN)에 보드의 Output 값(PE2)이 들어간다. PE2를 켜야 하므로 2번 Bit에 1을 할당하고, 16진수로 바꾼 0x00000004로 값을 설정한다.

- **조이스틱 DOWN: 모터 정지**
  조이스틱 DOWN에 해당하는 PC2가 Input으로 들어오면 2번 Bit가 0이므로, GPIOC_IDR의 값은 0x0000FFFB이다. 이때는 모터가 정지해야 하므로, Output에 해당하는 GPIOE_BSRR에서 18번 Bit에 1을 할당해서 PE2를 꺼야 한다. 이전과 동일한 방법으로 값을 0x00040000로 설정한다.

한계 / 어려웠던 점





- 잘못 짠 코드를 올리고 다시 디버깅하였을 때 사진과 같은 오류 창이 나왔다. 이전에 잘못 짠 코드가 레지스터에 남아있는 경우가 있어 Project-Download-Erase memory를 한 뒤 다시 디버깅을 해 봤지만 문제를 해결할 수 없었고, 점퍼선을 다시 끼우고 돌려보니 제대로 작동했다. 가끔 이런 문제가 생긴다고 하셨다.
- 릴레이 모듈의 문제로 모터가 반대로 돌아가는 오류가 있었다.