**4주차 결과보고서**



10조 (001분반)

과목명:임베디드시스템설계및실험

담당 교수 : 정상화 교수님

담당 조교 : 최호진 조교님

조원 : 201924603 하규승(조장)

201727102 강준혁

201924525 이광훈

202023139 박지원

제출 날짜 : 2023.10.01

**목차**

목표

1. 스캐터 파일의 이해 및 플래시 프로그래밍

2. 릴레이 모듈의 이해 및 임베디드 펌웨어를 통한 동작

3. 폴링 방식의 이해

세부 실험 내용

1. Datasheet 및 Reference Manual을 참고하여 해당 레지스터 및 주소에 대한 설정 이해

2. 스캐터 파일을 통해 플래시 메모리에 프로그램 다운 로드

3. 플래시 메모리에 올려진 프로그램 정상적인 동작 확 인

**목표**

1. **스캐터 파일의 이해 및 플래시 프로그래밍**

스캐터 파일(Scatter File)은 실행될 바이너리 이미지가 메모리에 로드될 때, 그 바이너리 이미지의 어떤 영역이 어느 주소에 어느 크기만큼 배치되어야 할지 작성한 파일입니다. 이러한 스캐터 파일은 바이너리의 여러 부분을 각각의 메모리 영역에 로드해야 될 때 필요하며, 특히 자주 사용되거나 빠른 실행을 요구하는 코드영역을 접근 시간이 빠른 메모리에 우선 배치하도록 설정할 수 있습니다. 이 스캐터 파일은 다음과 같이 구성됩니다:

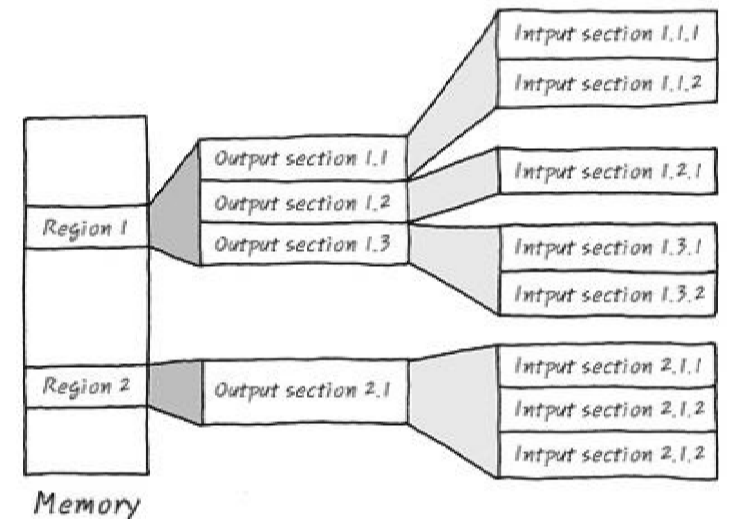


그림 1. Scatter File의 구조

1. Input Section
2. RO (Read Only) : 읽기만 가능하고 수정이 불가한 symbol(ex) const형, 소스 코드)
3. RW (Read Write) : 읽기/쓰기가 모두 가능한 symbol(ex) 초기화 값이 있는 전역 변수)
4. ZI (Zero Initialized) : 0으로 초기화된 symbol(ex) 초기화 값이 없는 전역 변수)
5. Output Section
6. Input Section 들 중에 같은 속성값을 갖는 것들을 묶어 놓은 것
7. Region

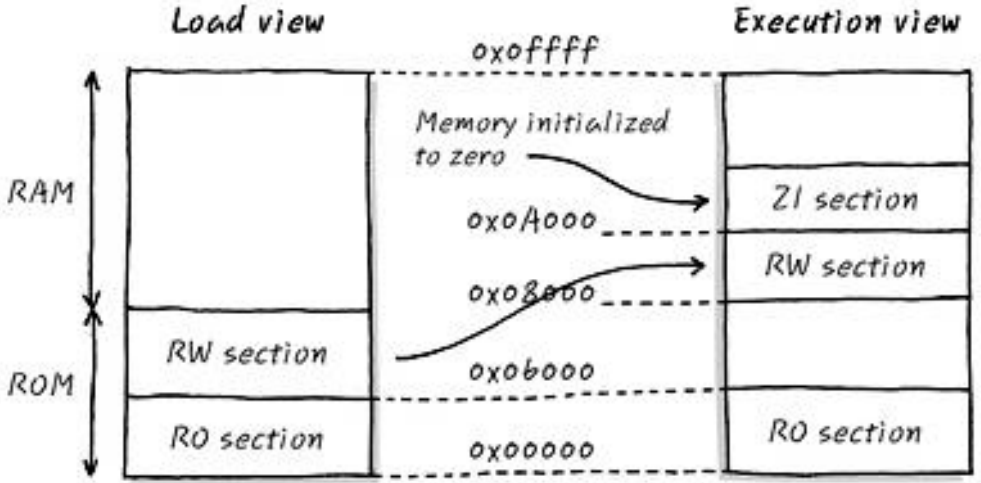


그림 2. Program의 Load/Execution View

위와 같이 구성된 Scatter File을 통해, Binary Image는 지정된 주소의 ROM에 로드되어 대기 중인 프로그램이 됩니다. 프로그램이 실행되게 되면 RO Section은 그대로 ROM에 남고, RW와 ZI Section은 지정된 주소의 RAM으로 이동하여 실행되게 됩니다.

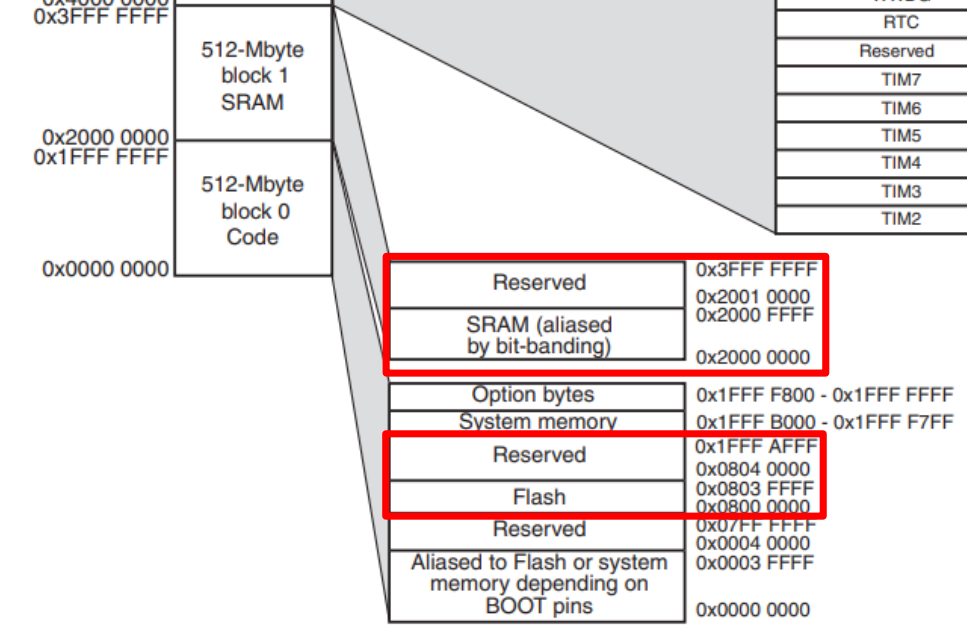


그림 3. RAM / ROM의 Memory Mapping

이번 실험에서는 RAM과 ROM 영역의 크기를 위의 남는 공간인 Reserved 공간을 활용하여 실험 조건을 충족하는 방향으로 늘리거나 줄이게 됩니다.

**2. 릴레이 모듈의 이해 및 임베디드 펌웨어를 통한 동작**

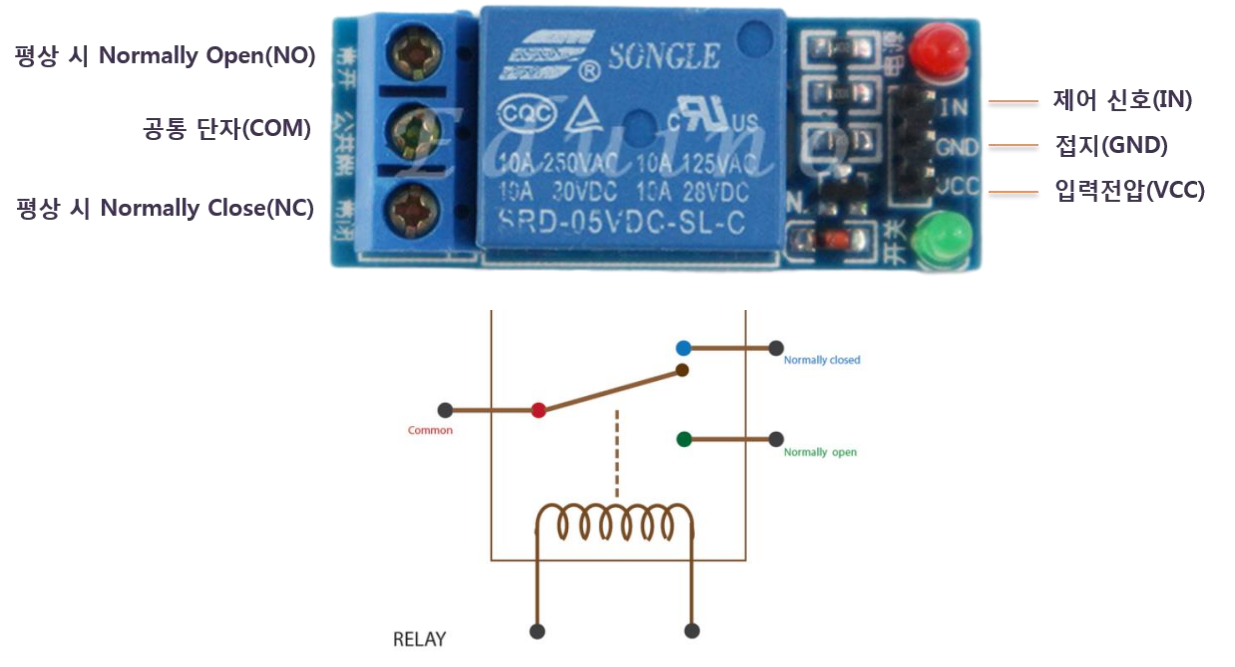


그림 4. Relay의 내부 회로도

릴레이는 입력되는 신호에 따라 작동하여 회로를 개폐하는 장치입니다. 릴레이는 내부에 코일로 이루어진 전자석이 있으며 이 전자석에 전류가 흐르게 되면 전자기 유도현상에 의해 자석이 되어 옆의 철을 끌어당기게 되어 스위치가 ON 됩니다. 이후 전류가 끊어지면 다시 옆의 철이 원래 자리로 돌아가게 되어 스위치가 OFF 됩니다.

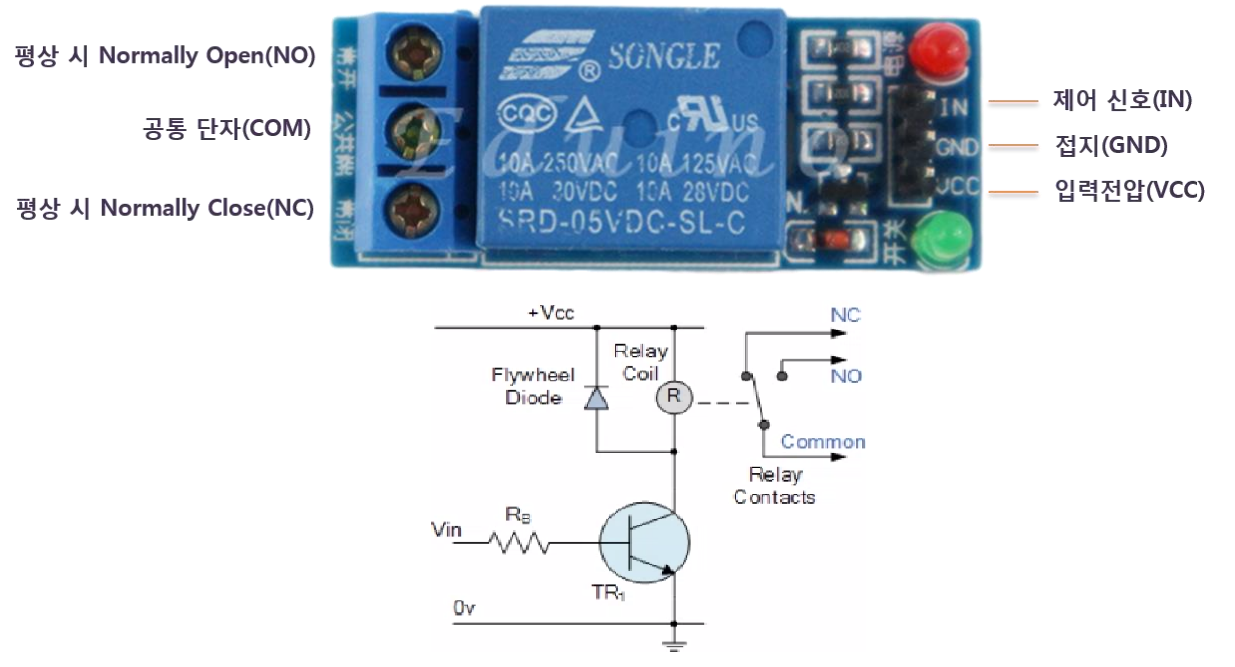


그림 4. Relay Module의 외관과 내부 회로도

릴레이 모듈은 릴레이를 이용해 구성되었으며, Vcc - GND를 통해 3.3V의 전원이 인가되어야 합니다. 릴레이의 상태는 IN에 따라 결정되며, 평소(IN이 Low)에는 NC와 COM이 연결된 상태, In이 High 일 땐 NO와 COM이 연결된 상태가 됩니다.

**3. 폴링 방식의 이해**

CPU와 입/출력 장치 사이에는 통신이 필요하고, 주된 통신 방법이 2가지가 있는데, 크게 인터럽트(Interrupt)와 폴링(Polling) 방식으로 나뉩니다.

일단 각 방식에 관해 설명하자면, Interrupt 방식은 **H/W가 CPU에** 예외 상황이 발생하여 처리가 필요하다고 알려주는 방식입니다. 반대로 Polling 방식은 **CPU가 H/W를** 주기적으로 감시하면서 예외 상황 발생 시 처리하는 방식입니다.

각 방식에 대해 차이점은 다음과 같습니다. Interrupt는 CPU 외부에서 신호가 들어오는 방식이고, H/W로부터 신호가 들어오게 되면 CPU는 진행 중인 작업을 잠시 멈추고 Interrupt 처리 루틴을 실행하여 신호를 처리하게 됩니다. 반대로 Polling은 CPU가 H/W의 변화를 주기적으로 체크해야 하므로 CPU가 다른 일을 하는 도중에 신호를 읽을 수 없고, 이에 따라 문제가 생기더라도 CPU가 일단 작업을 시작했다면, 그 작업이 끝난 이후에야 오류를 발견하게 됩니다.

이러한 Interrupt 방식과 Polling 방식의 차이점으로 인해, Polling 방식은 오류가 일어난 시점에서 즉시 반응이 힘들고, 계속 해당 H/W에 대해 상태를 체크해야 하므로 레지스터를 계속 확인해 봐야 하므로 CPU의 점유율이 높다는 단점이 있습니다. 하지만 단점만 있는 건 아니고, Interrupt 방식에 비해 Polling 방식이 좀 더 구현이 쉽다는 장점이 있습니다.

**세부 실험 내용**

**1. 스캐터 파일의 이해 및 플래시 프로그래밍**

스

**2. 릴레이 모듈의 이해 및 임베디드 펌웨어를 통한 동작**

베

**3. 폴링 방식의 이해**

임