



Pusan
National
University



September 20, 2022

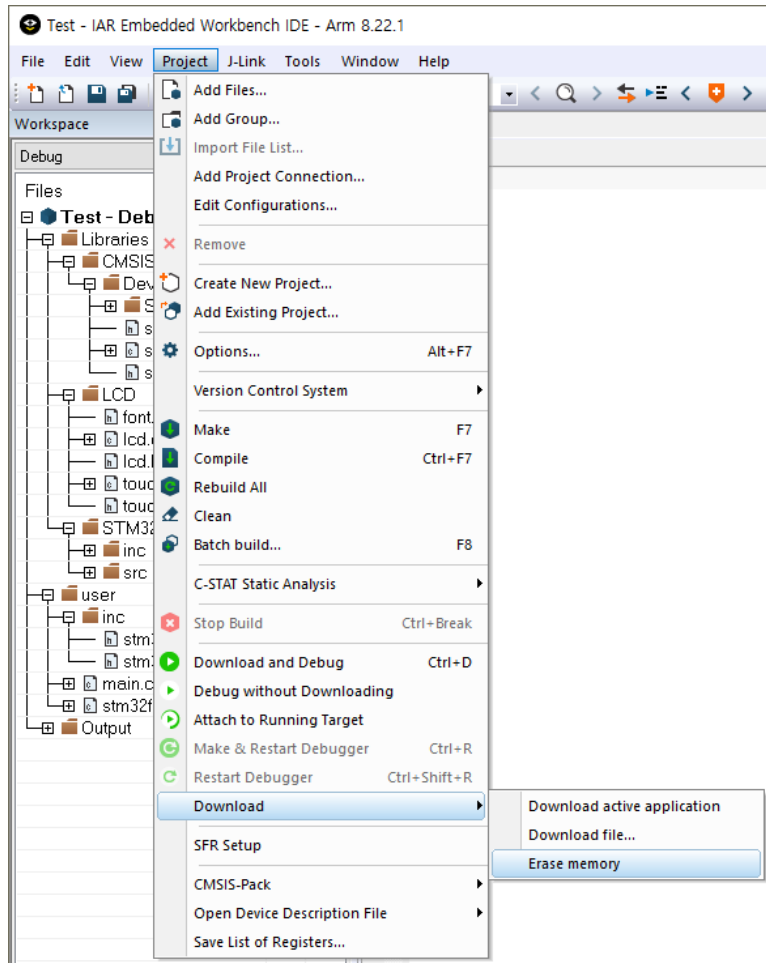
조교
이시현

shyeon0528@gmail.com

임베디드 시스템 설계 및 실험

화요일 분반

4주차



1. 이전 코드가 동작하는 것 같을 때

간혹 이전에 잘못 짠 코드를 올린 것이 그대로 레지스터에 남아서 원하는 대로 동작이 안 될 경우가 있습니다.

왼쪽과 같이 erase memory를 한 뒤 작성한 코드를 올리시기 바랍니다.

2. 터미널로 디버깅하는 방법

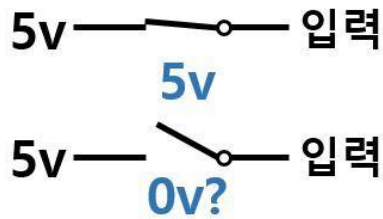
→ 디버깅 모드 진입 – View – Terminal I/O, printf로 디버깅

Contents

4주차 실험 내용

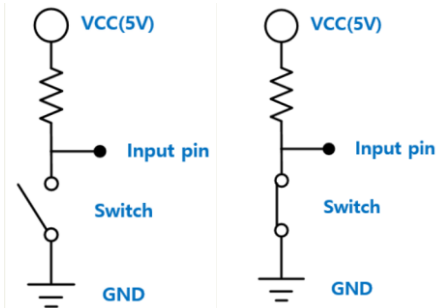
- 스케터 파일의 이해 및 플래시 프로그래밍
- 릴레이 모듈의 이해 및 임베디드 펌웨어를 통한 동작
- 센싱에서 폴링 방식의 이해

• 플로팅 (Floating)



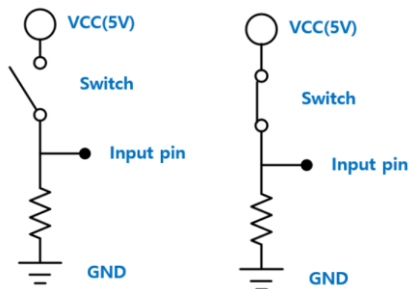
- 전압을 High / Low 로 보기 힘든 상태
- 아주 작은 노이즈만으로도 High와 low 사이를 빠르게 이동하여 오동작 유발
- 따라서 풀업 저항 또는 풀다운 저항을 사용

• Pull Up



- VCC에 저항을 연결하는 방법
- 스위치 OFF 시 input에는 High 신호
- 스위치 ON 시 input에는 Low 신호

• Pull Down



- GND에 저항을 연결하는 방법
- 스위치 OFF 시 input에는 Low 신호
- 스위치 ON 시 input에는 High 신호

Scatter File 이란?

컴퓨터인터넷IT용어대사전

분산 적재

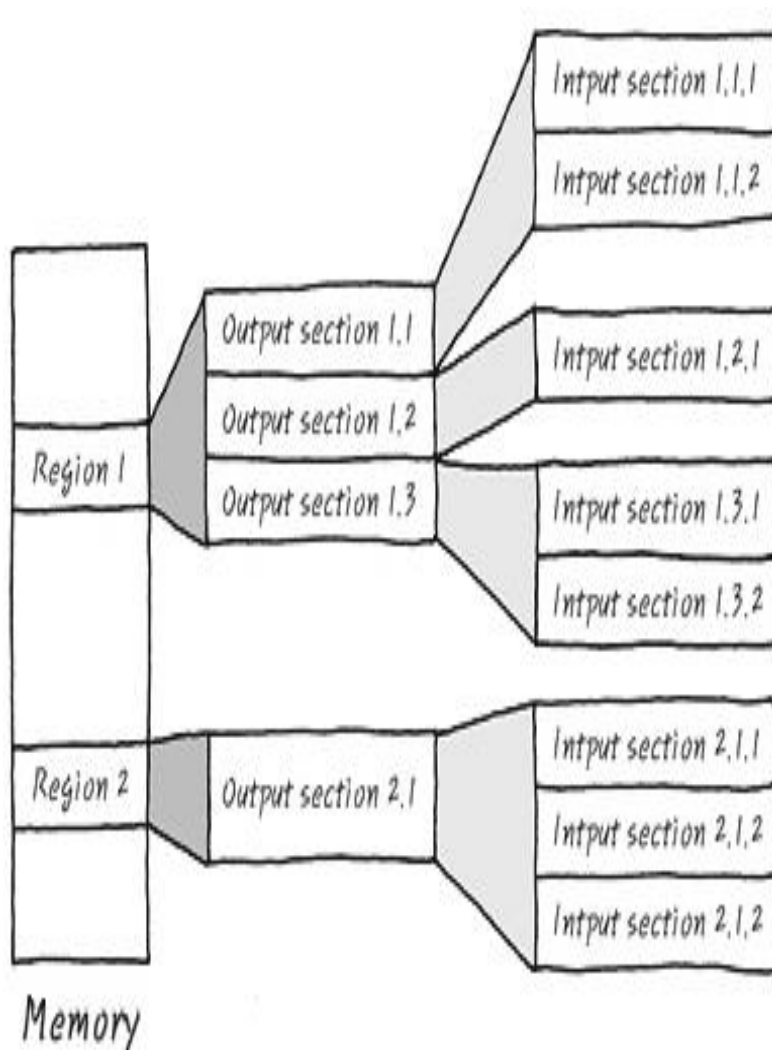
[scatter loading]

꺼내기의 한 형식으로 판독 모듈의 제어 섹션을 주기억 장치 가운데 각각의 장소에 적재하는 것.

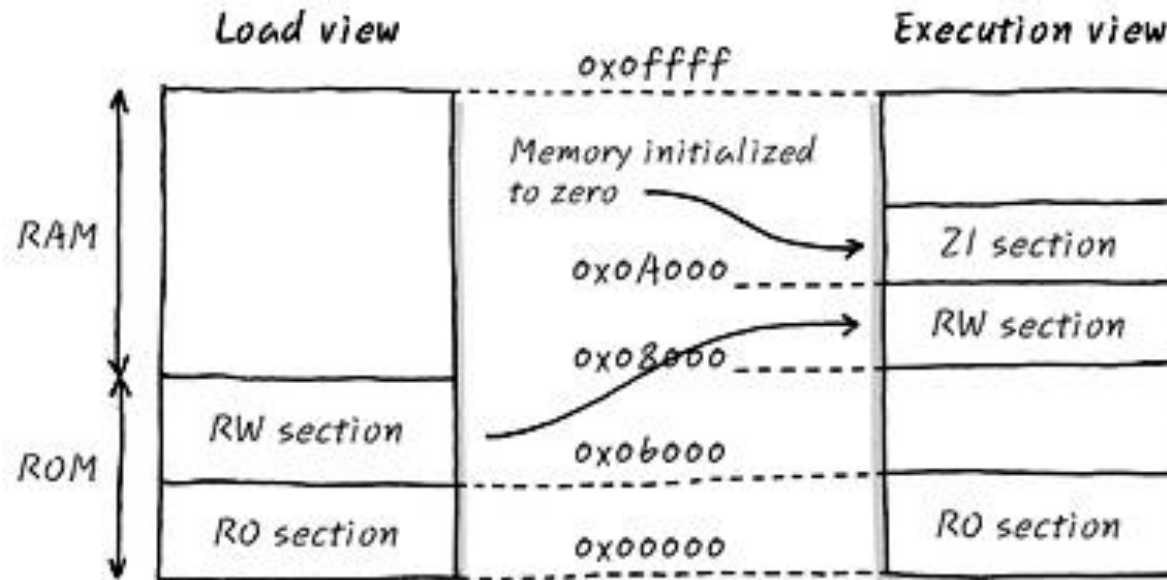
실행시킬 바이너리 이미지가 메모리에 로드될 때,
바이너리 이미지의 어떤 영역이 어느 주소에 어느
크기만큼 배치되어야 할 지 작성한 파일.

Scatter File이 필요한 이유?

- 1) 바이너리의 여러 부분을 각각 별개의 메모리 영역에 로드해야 될 때
- 2) 자주 사용되거나 빠른 실행을 요구하는 코드영역을 접근 시간이 빠른 메모리에 우선 배치하도록 설정할 수 있음.

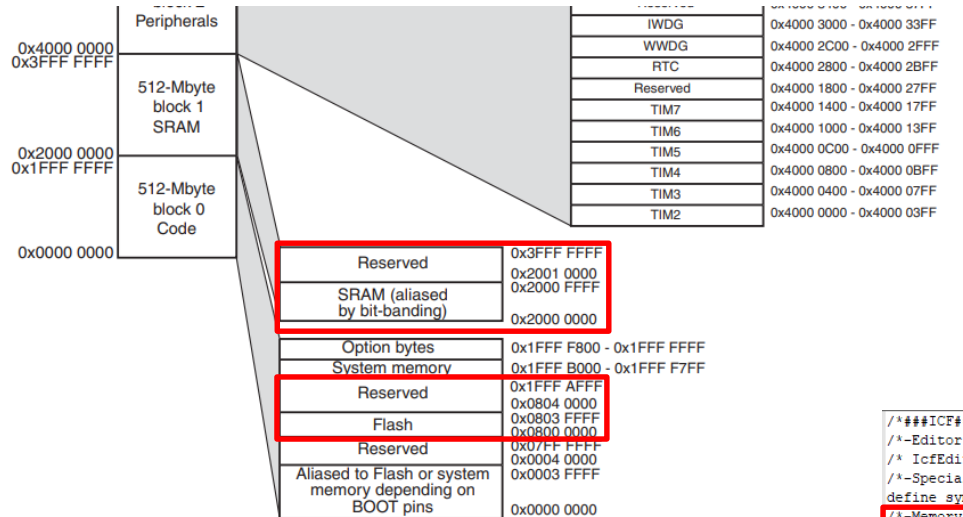


- **Input Section**
 - RO (code, constant data)
 - RW (global data)
 - ZI(zero initialized)
 - 중 하나의 속성을 갖는 집합
- **Output Section**
 - Input section들 중에 같은 속성을 갖는 것들을 묶어 놓은 것
- **Region**
 - Output section을 묶어 놓은 것



- Load view : flash에 실행 image가 담겨 있을 때의 형태
- Execution view : flash에 실행 image가 실행 될 때의 형태

IAR EW는 .icf 파일을 스캐터 파일로 이용



```

/****ICF**** Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file-*/
/* IcfEditorFile="$TOOLKIT_DIR$\config\ide\IcfEditor\cortex_vl_0.xml" */
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = // TODO
define symbol __ICFEDIT_region_ROM_end__ = // TODO
define symbol __ICFEDIT_region_RAM_start__ = // TODO
define symbol __ICFEDIT_region_RAM_end__ = // TODO
/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x1000;
define symbol __ICFEDIT_size_heap__ = 0x1000;
/**** End of ICF editor section. ****ICF****/

define memory mem with size = 4G;
define region ROM_region = mem:[from __ICFEDIT_region_ROM_start__ to __ICFEDIT_region_ROM_end__];
define region RAM_region = mem:[from __ICFEDIT_region_RAM_start__ to __ICFEDIT_region_RAM_end__];

define block CSTACK with alignment = 8, size = __ICFEDIT_size_cstack__ { };
define block HEAP with alignment = 8, size = __ICFEDIT_size_heap__ { };

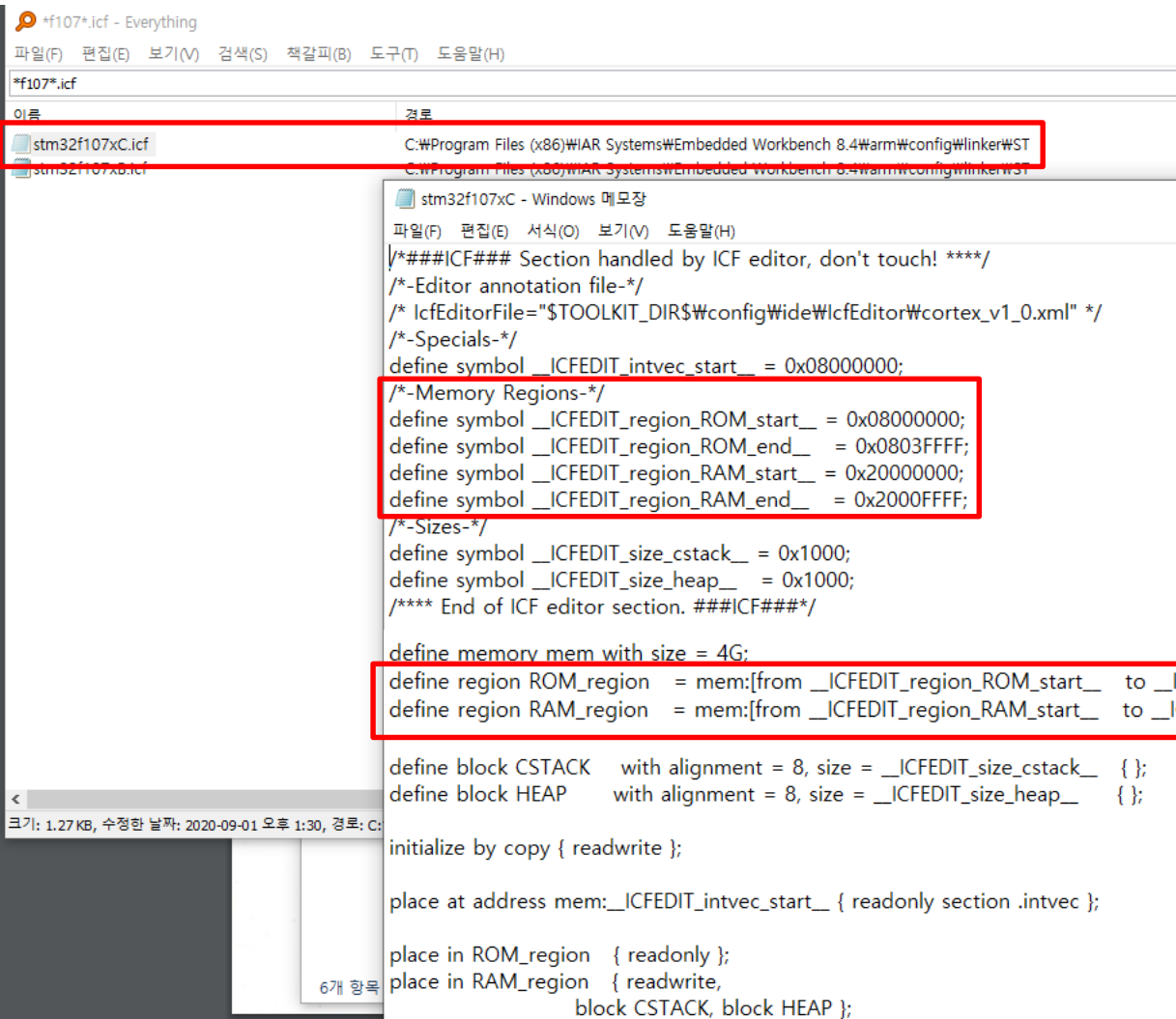
initialize by copy { readwrite };
do not initialize { section .noinit };

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };

place in ROM_region { readonly };
place in RAM_region { readwrite,
block CSTACK, block HEAP };
    
```

원하는 만큼 메모리 영역을 할당 가능

우리 보드 모델명인 "STM32F107VCT6" 에 대한 스캐터 파일을 확인할 수 있음

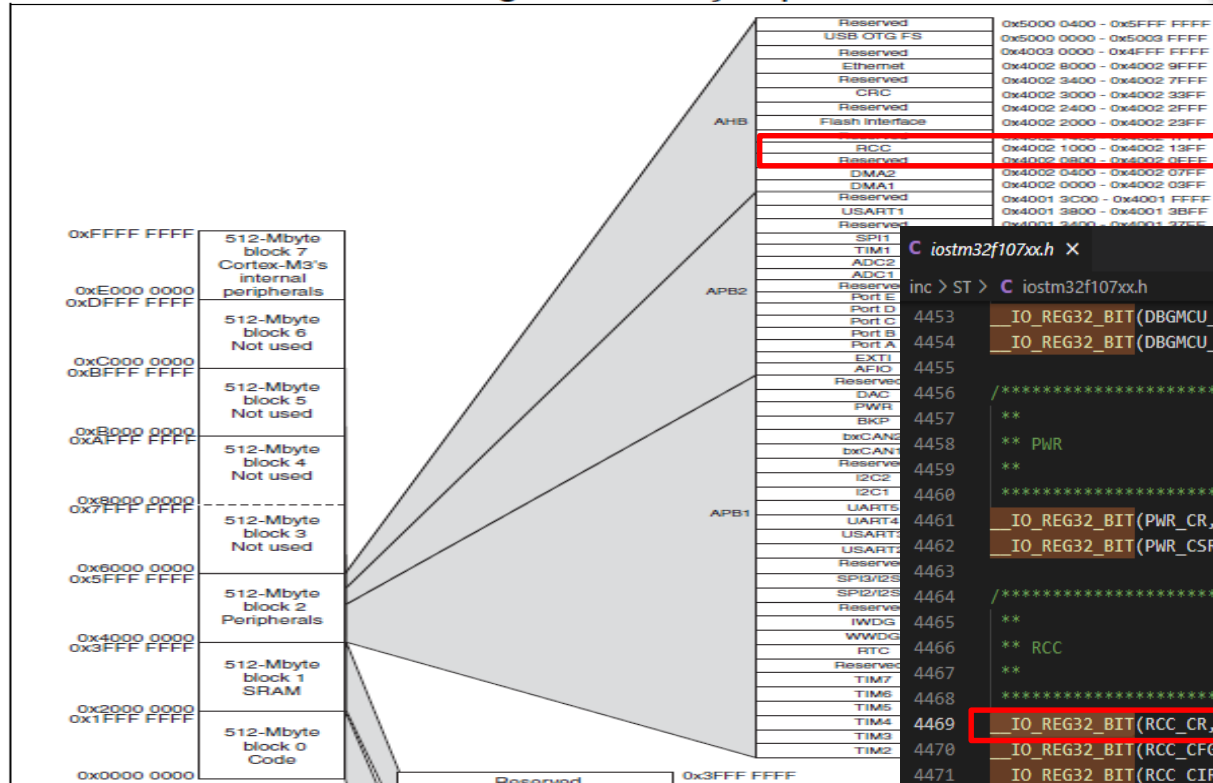


Features

- Core: ARM® 32-bit Cortex®-M3 CPU
 - 72 MHz maximum frequency, 1.25 DMIPS/MHz (Dhrystone 2.1) performance at 0 wait state memory access
 - Single-cycle multiplication and hardware division
- Memories
 - 64 to 256 Kbytes of Flash memory
 - 64 Kbytes of general-purpose SRAM
- Clock, reset and supply management
 - 2.0 to 3.6 V application supply and I/Os
 - POR, PDR, and programmable voltage detector (PVD)
 - 3-to-25 MHz crystal oscillator
 - Internal 8 MHz factory-trimmed RC
 - Internal 40 kHz RC with calibration
 - 32 kHz oscillator for RTC with calibration
- Low power

The memory map is shown in *Figure 5*.

Figure 5. Memory map



C iostm32f107xx.h X

```
inc > ST > C iostm32f107xx.h
```

```
4453 IO_REG32_BIT(DBGMCU_IDCODE, 0xE0042000, __READ, __dbgmcu_idcode_bits);
4454 IO_REG32_BIT(DBGMCU_CR, 0xE0042004, __READ_WRITE, __dbgmcu_cr_bits);
```

```

4456 /*****
4457 **
4458 ** PWR
4459 **
4460 *****/

```

```
4461 IO_REG32_BIT(PWR_CR, 0x40007000, __READ_WRITE, __pwr_cr_bits);
4462 IO_REG32_BIT(PWR_CSR, 0x40007004, __READ_WRITE, __pwr_csr_bits);
```

```

4464 /*****
4465 **
4466 ** RCC
4467 **
4468 *****/

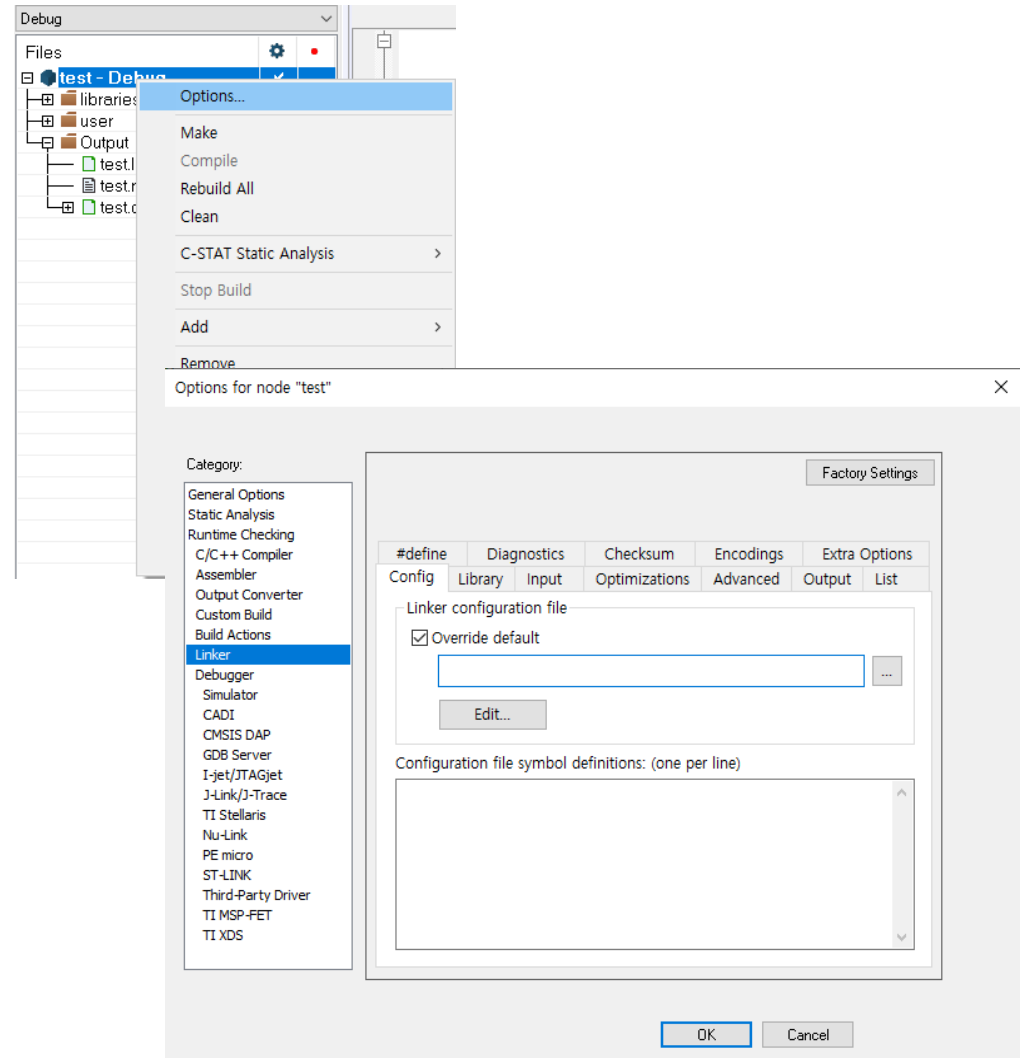
```

```
4469 IO_REG32_BIT(RCC_CR, 0x40021000, _READ_WRITE, _rcc_cr_bits);
4470 IO_REG32_BIT(RCC_CFGR, 0x40021004, _READ_WRITE, _rcc_cfgr_bits);
4471 IO_REG32_BIT(RCC_CIR, 0x40021008, _READ_WRITE, _rcc_cir_bits);
4472 IO_REG32_BIT(RCC_APB2RSTR, 0x4002100C, _READ_WRITE, _rcc_apb2rstr_bits);
4473 IO_REG32_BIT(RCC_APB1RSTR, 0x40021010, _READ_WRITE, _rcc_apb1rstr_bits);
4474 IO_REG32_BIT(RCC_AHBENR, 0x40021014, _READ_WRITE, _rcc_ahbenr_bits);
4475 IO_REG32_BIT(RCC_APB2ENR, 0x40021018, _READ_WRITE, _rcc_apb2enr_bits);
4476 IO_REG32_BIT(RCC_APB1ENR, 0x4002101C, _READ_WRITE, _rcc_apb1enr_bits);
4477 IO_REG32_BIT(RCC_BDCR, 0x40021020, _READ_WRITE, _rcc_bdcr_bits);
4478 IO_REG32_BIT(RCC_CSR, 0x40021024, _READ_WRITE, _rcc_csr_bits);
4479 IO_REG32_BIT(RCC_AHBRSTR, 0x40021028, _READ_WRITE, _rcc_ahbrstr_bits);
4480 IO_REG32_BIT(RCC_CFGR2, 0x4002102C, _READ_WRITE, _rcc_cfg2_bits);
```

- 작성한 *.icf 파일
- project 오른쪽 클릭 – Options..
- Linker – Config – Override default
- ... 을 눌러 업로드

```

0x800'02ae: 0xb538 PUSH {R3-R5, LR}
*GCC_APB2ENR |= 0x8; // port B enable
0x800'02b0: 0x482d LDR N R0, [PC, #0xb4] ...
0x800'02b2: 0x6801 LDR R1, [R0]
0x800'02b4: 0xf051 0x0108 ORRS.W R1, R1, #8
0x800'02b8: 0x6001 STR R1, [R0]
*GCC_APB2ENR |= 0x10; // port C enable
0x800'02ba: 0x6801 LDR R1, [R0]
0x800'02bc: 0xf051 0x0110 ORRS.W R1, R1, #16 ...
0x800'02c0: 0x6001 STR R1, [R0]
*GCC_APB2ENR |= 0x20; // port D enable
0x800'02c2: 0x6801 LDR R1, [R0]
0x800'02c4: 0xf051 0x0120 ORRS.W R1, R1, #32 ...
0x800'02c8: 0x6001 STR R1, [R0]
*GPIOB_cfg_reg_high &= ~0xf; // joystick input mode
0x800'02ca: 0x482d LDR N R0, [PC, #0xa0] ...
0x800'02cc: 0x6801 LDR R1, [R0]
0x800'02ce: 0x0909 ISRS R1, R1, #4
0x800'02d0: 0x0109 ISLS R1, R1, #4
0x800'02d2: 0x6001 STR R1, [R0]
*GPIOB_cfg_reg_high |= 0x8;
0x800'02d4: 0x6801 LDR R1, [R0]
0x800'02d6: 0xf051 0x0108 ORRS.W R1, R1, #8
0x800'02da: 0x6001 STR R1, [R0]
*GPIOC_cfg_reg_high &= ~0xff; // reset GPIOC pin 8, 9
0x800'02dc: 0x482d LDR N R0, [PC, #0x90] ...
0x800'02de: 0x6801 LDR R1, [R0]
0x800'02e0: 0x0a09 ISRS R1, R1, #8
0x800'02e2: 0x0209 ISLS R1, R1, #8
0x800'02e4: 0x6001 STR R1, [R0]
*GPIOC_cfg_reg_high |= 0x33; // output mode (0011)
    
```

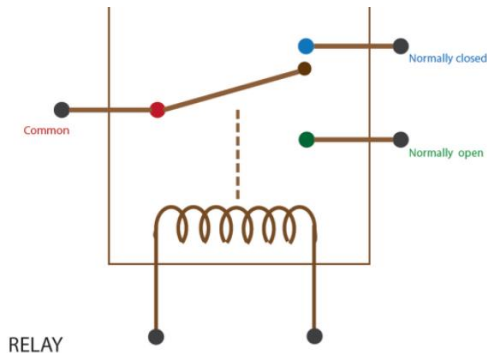
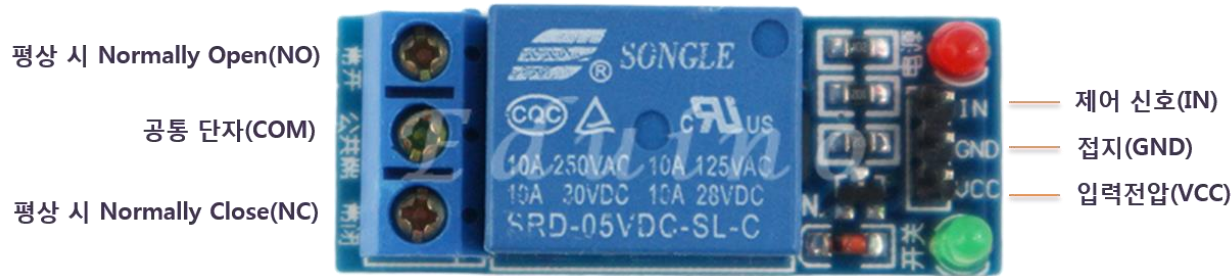


Interrupt

- Hardware의 변화를 감지해 외부로부터 전기신호 입력을 CPU가 알아채는 방법
- CPU 마다 다른 방식으로 동작
- 진행 중인 일을 잠시 멈추고 인터럽트 처리 루틴을 실행하여 신호를 처리함

Polling

- Hardware의 변화를 지속적으로 읽어들이며 변화를 알아채는 방법
- 신호를 판단하기 위해 지속적으로 확인해야 함
- 다른 일을 하는 중에 신호를 읽을 수 없음



Relay Module

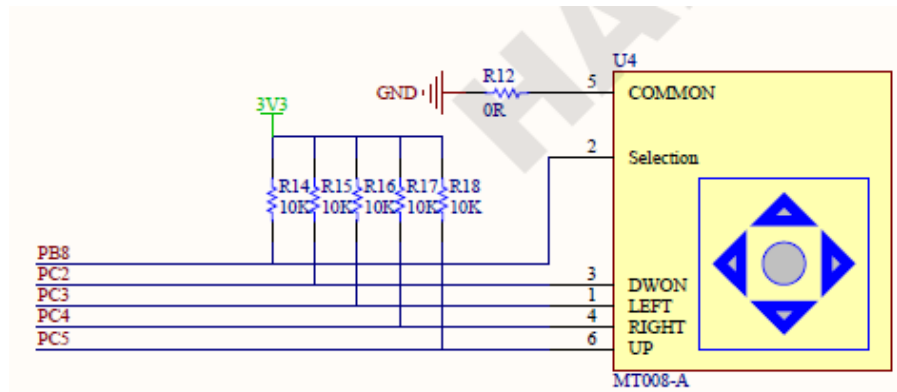
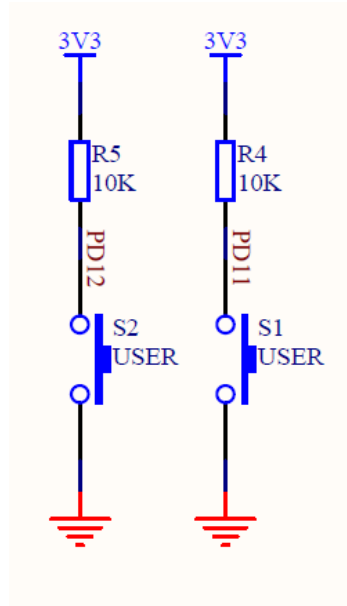
- : 릴레이를 제어하는 모듈
- : 전자기유도원리를 이용하여 스위치 역할
- : 릴레이에 신호를 가하면 출력 상태(ON/OFF)가 변경된다

릴레이 모듈에 **3.3v** 전원 인가해서 사용 (5V 는 작동 안 할 수 있음)

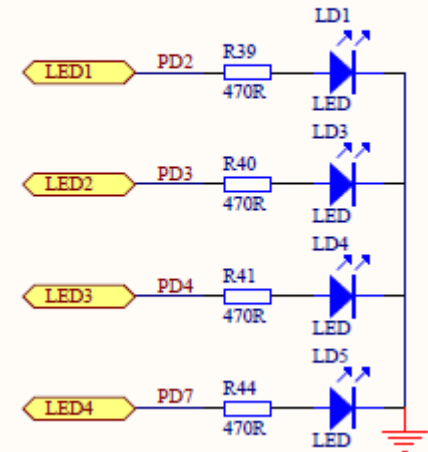
COM은 제어 신호(IN)에 따라 NO 또는 NC로 붙는다

NO: 평소에 open, high 신호가 들어오면 close

NC: 평소에 close, high 신호가 들어오면 open



LED



4주차 실험 릴레이 모듈 사용을 위한 필수 GPIO : PC5, PC2, PD11

릴레이 모듈의 Input(보드 output) 핀 : 조별 선택

- 실험 장비들을 연결 및 분리할 때 반드시 모든 전원을 끄고 연결해주세요.
- 장비사용시 충격이 가해지지 않도록 주의해주세요.
- 자리는 항상 깔끔하게 유지하고 반드시 정리 후 퇴실해주세요.
- 실험 **소스 코드와 프로젝트 폴더**는 **백업** 후 반드시 **삭제**해주세요.
- 장비 관리, 뒷정리가 제대로 되지 않을 경우 해당 조에게 감점이 주어집니다.

- **동작 중 케이블 절대 뽑지말것**
- **보드는 전원으로 USBPort나 어댑터(5V,1A)를 사용할것 (5V 5A 어댑터(비슷하게 생김)와 혼동하지 말 것, 사용시 보드가 타버림 -> 감점)**
- **디버깅 모드 중에 보드 전원을 끄거나 연결 케이블을 분리하지 말 것!!!**

- **-> 지켜지지 않을 시 해당 조 감점**

미션 ! 별도 미션지 참고

실험 검사

오늘 검사 받을 수 있는 조는 오늘 받고 퇴실.

이번 주 실험 결과 보고서 및 소스 코드

- A. 이론부터 실습까지 전반적인 내용을 포함하도록 작성 (실험 과정 사진 찍으시면 좋아요)
- B. 다음 실험시간 전까지 PLATO 제출 (보고서, 동작 영상)
- C. 소스 코드는 직접 작성 및 수정한 파일만 제출

예비 발표 조는 발표 자료(영상) 만들어서
일요일 24시까지 조교 이메일로 제출

나가실 때, 만드신 코드 및 프로젝트 폴더는 모두 백업하시고 삭제해주세요.
다른 분반 파일은 만지지 마시고 조교에게 알려주세요.
자리 정리정돈 안 되어 있으면 **감점**합니다!!!