임베디드 시스템 설계 및 실험
월요일 분반

4주차

September 27, 2021

조교
최호진
tiger981228@pusan.ac.k

공지

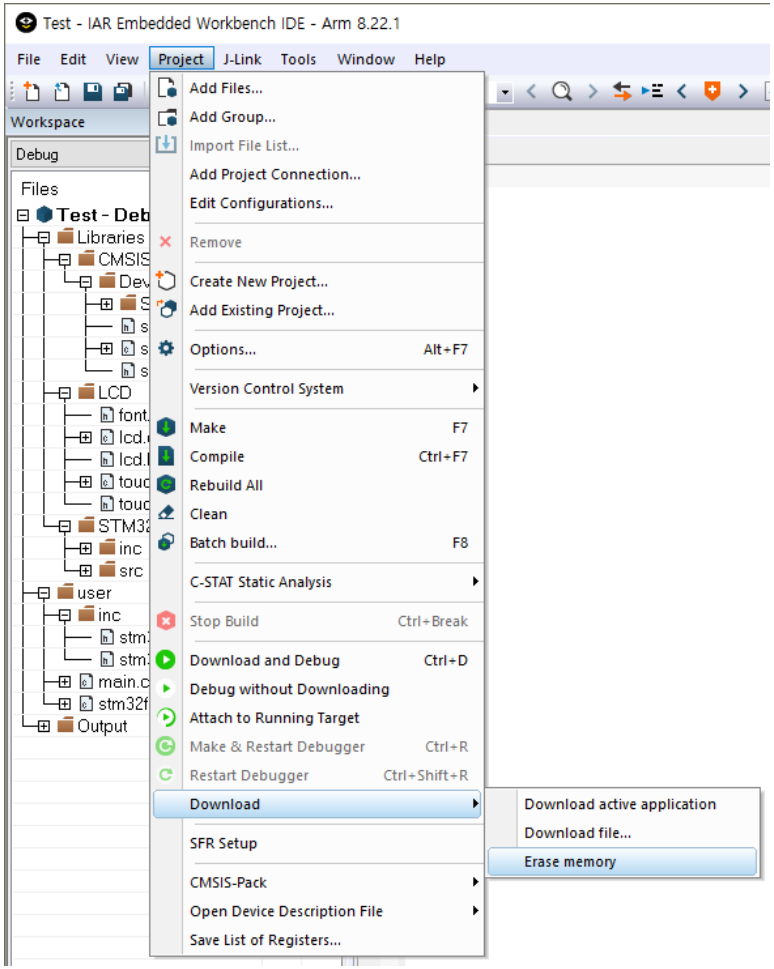
예비 발표

- 해당 주차 실습 내용 및 이론적 배경 등을 약 10분 발표
- 발표 자료는 수업 하루 전(일요일) 23:59 까지 PLATO '예비 발표 자료실'에 PDF 업로드 (늦으면 감점)

발표 일정

9월 25일	10월 13일	10월 16일	10월 30일	11월 6일	11월 13일	11월 20일
8조, 11조	2조, 9조	5조, 10조	4조, 6조	7조	3조	1조

공지

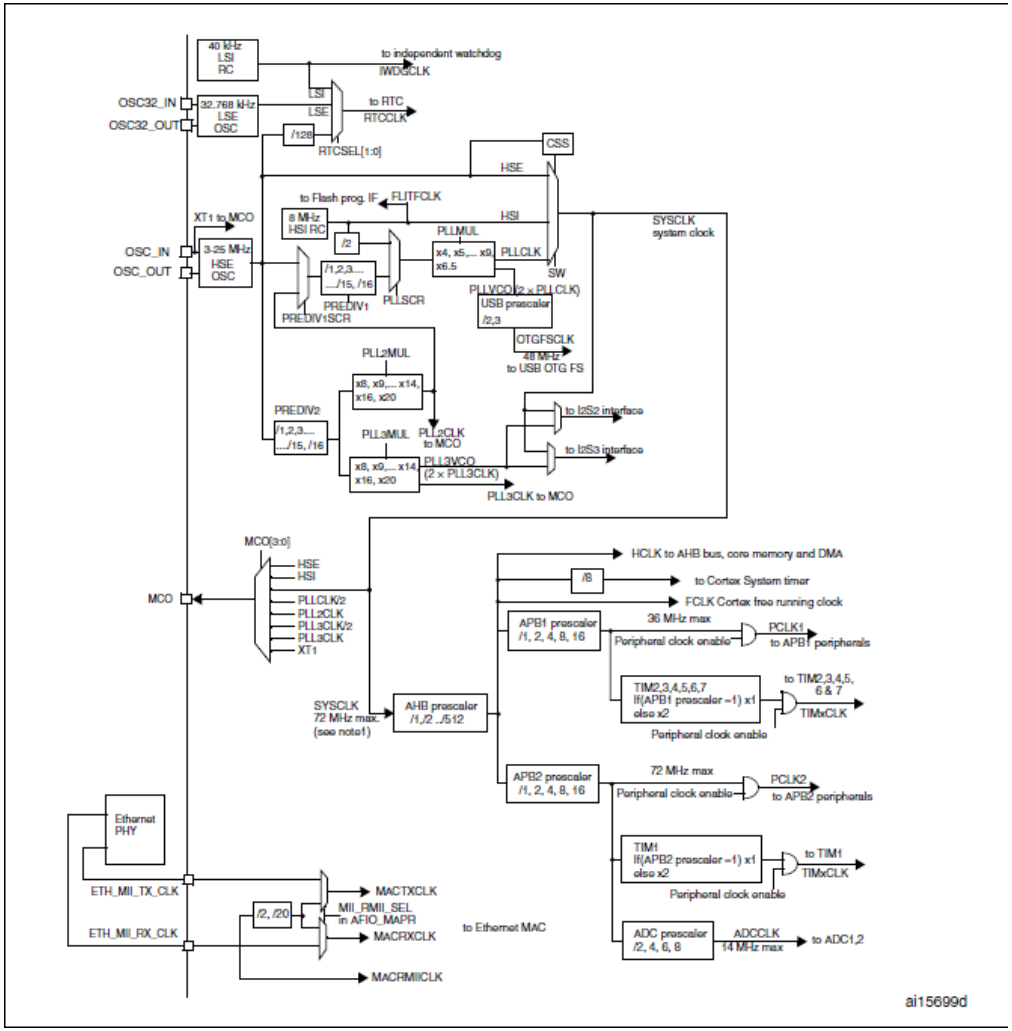


간혹 이전에 잘못 짠 코드를 올린 것이 그대로 레지스터에 남아서 원하는 대로 동작이 안 될 경우가 있습니다.

왼쪽과 같이 memory를 지운 뒤 작성한 코드를 올리시기 바랍니다.

Clock Tree

- Reference Manual 126p
- FCLK : CPU에 사용
- HCLK : AHB Bus에 사용
  - 고속 입출력 장치에 사용
- PCLK : APB Bus에 사용
  - 저속 입출력 장치에 사용



Clock 설정

- 내부 clock (HIS clock)
- 외부 clock (HSE clock)
- 내부와 외부 clock 선택 후 PLL을 통해 주파수 조정
  - PLL : Phase Locked Loop

MCO

- Microcontroller Clock Output
- STM 내부에서 사용되는 clock을 외부로 출력
- MCO 핀으로 출력할 때 GPIO 최대 속도를 넘으면 안됨

6주차 예비 발표 준비 내용

- Clock의 개념
  - HIS Clock 및 HSE Clock
  - Clock Tree에 대해 자세하게
  - PLL이 무엇인지
  - MCO가 무엇인지
- UART/USART의 개념
  - UART/USART 송수신 프로토콜
  - Data Frame (Start Bit, Data Bit, Parity Bit, Stop Bit, Baud Rate)에 대해 자세하게

INS LAB
IoT NETWORK & SYSTEMS LAB

Contents

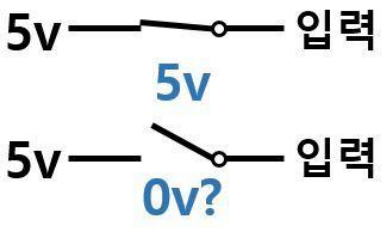
4주차 실험 내용

실험 목적

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

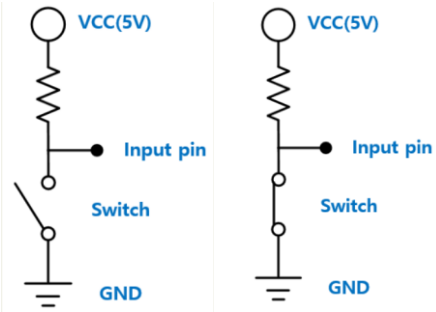
Floating / Pull Up / Pull Down

- **플로팅 (Floating)**



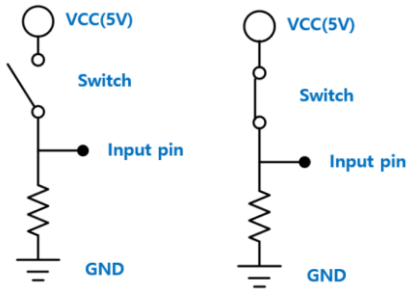
- 전압을 High / Low 로 보기 힘든 상태
- 아주 작은 노이즈만으로도 High와 low 사이를 빠르게 이동하여 오동작 유발
- 따라서 풀업 저항 또는 풀다운 저항을 사용

- **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.

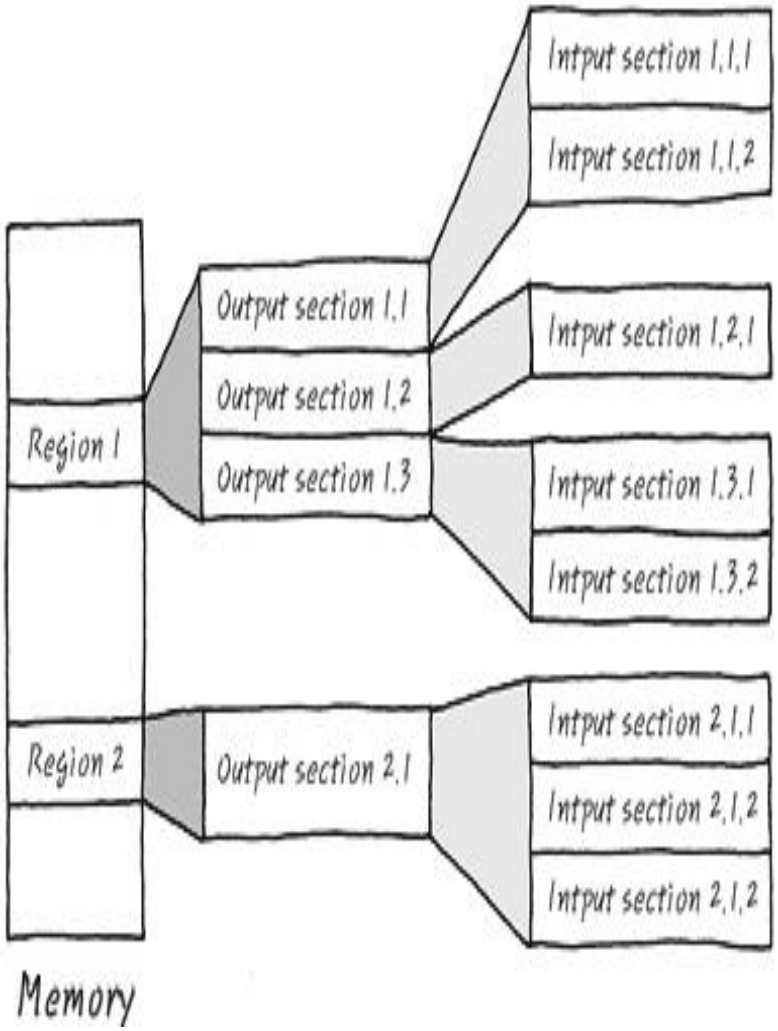
실행시킬 바이너리 이미지가 메모리에 로드될 때, 바이너리 이미지의 어떤 영역이 어느 주소에 어느 크기만큼 배치되야 할 지 작성한 파일.

Scatter File이 필요한 이유?

1) 바이너리의 여러 부분을 각각 별개의 메모리 영역에 로드 해야 될 때

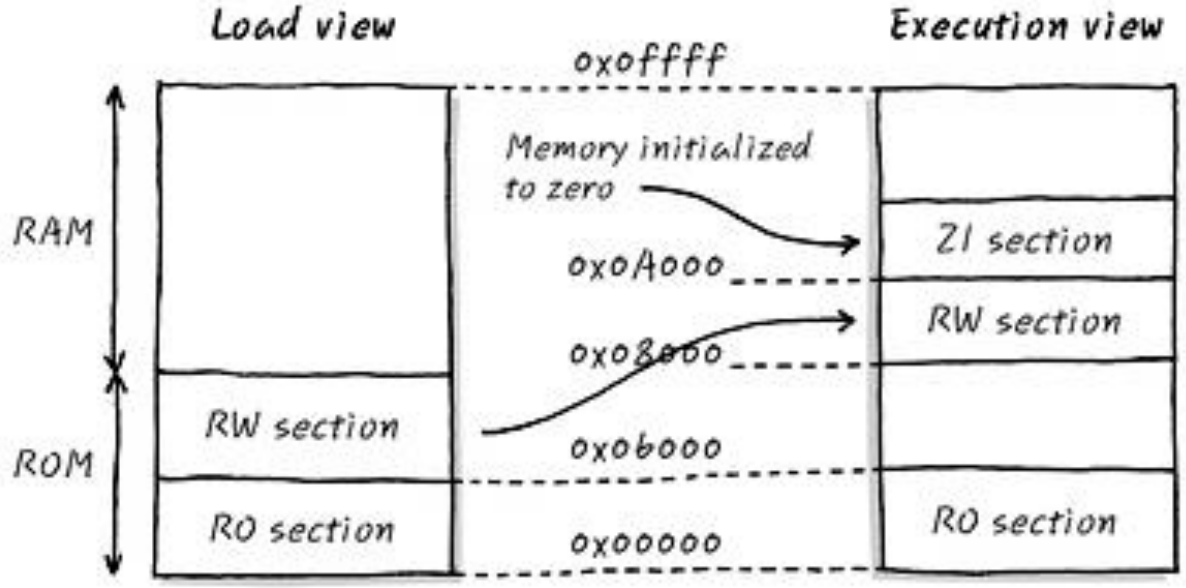
2) 자주 사용되거나 빠른 실행을 요구하는 코드영역을 접근 시간이 빠른 메모리에 우선 배치하도록 설정할 수 있음.

스캐터 파일 코드 분석



- **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

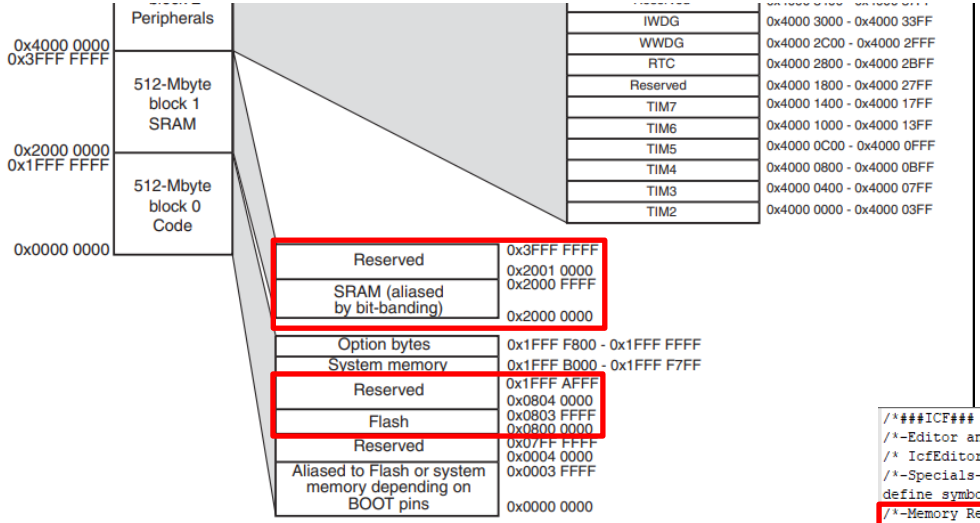
스캐터 파일 코드 분석



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

스캐터 파일 업로드

IAR EW는 .icf 파일을 스캐터 파일로 이용



원하는 만큼 메모리 영역을 할당 가능

```
/*###ICF### Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file-*/
/* IcfEditorFile="$TOOLKIT_DIR$\config\ide\IcfEditor\cortex_v1_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 };
```

STM32보드 스캐터 파일 확인

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

f107.icf - Everything

파일(F) 편집(E) 보기(V) 검색(S) 책갈피(B) 도구(T) 도움말(H)

f107.icf

이름 경로

stm32f107xC.icf C:₩Program Files (x86)₩IAR Systems₩Embedded Workbench 8.4₩arm₩config₩linker₩ST

stm32f107xB.icf C:₩Program Files (x86)₩IAR Systems₩Embedded Workbench 8.4₩arm₩config₩linker₩ST

크기: 1.27 KB, 수정한 날짜: 2020-09-01 오후 1:30, 경로: C:

6개 항목

stm32f107xC - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
/*###ICF### Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file-*/
/* IcfEditorFile="$TOOLKIT_DIR$₩config₩ide₩IcfEditor₩cortex_v1_0.xml" */
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__   = 0x0803FFFF;
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;
define symbol __ICFEDIT_region_RAM_end__   = 0x2000FFFF;
/*-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 };

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

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

Features

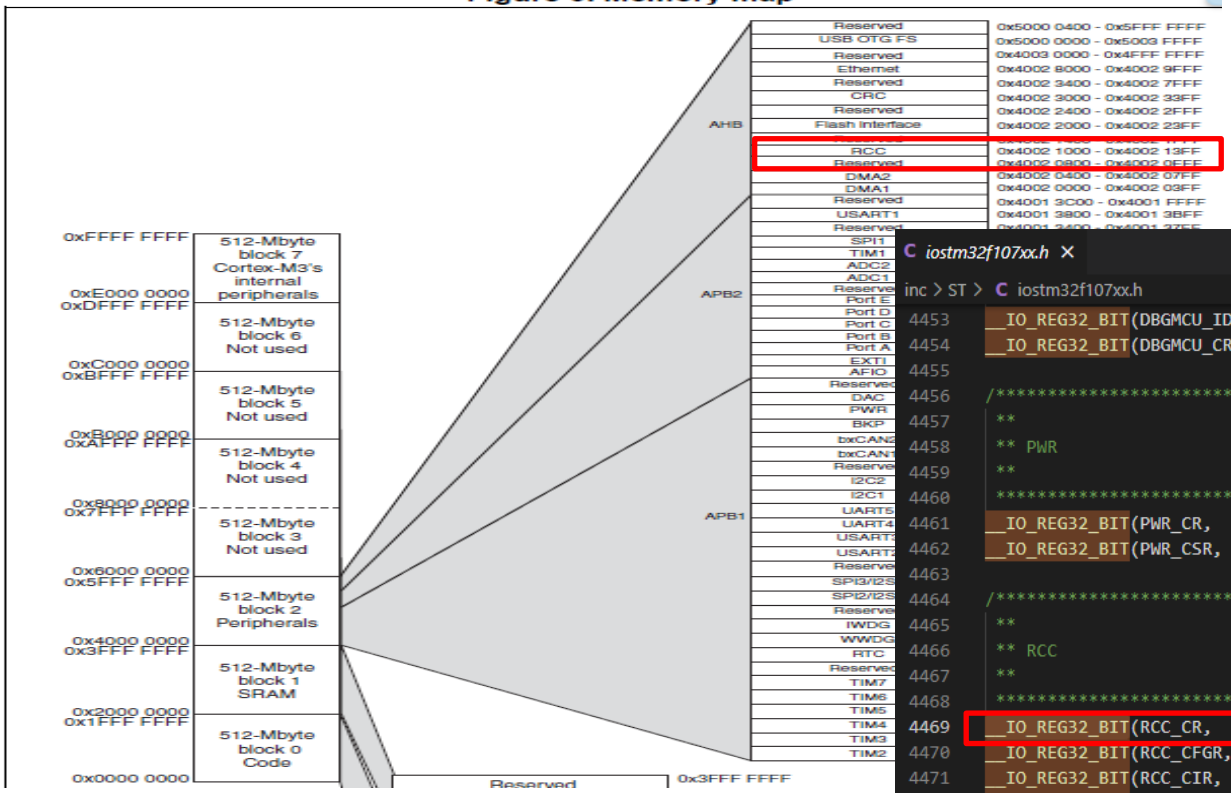
- Core: ARM® 32-bit Cortex®-M3 CPU
  - 72 MHz maximum frequency, 1.25 DMIPS/MHz (Dhrystone 2.1) performance at 0 wait state memory access
  - Single-cycle multiplication and hardware division
- Memories
  - 64 to 256 Kbytes of Flash memory
  - 64 Kbytes of general-purpose SRAM
- Clock, reset and supply 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

STM32보드 메모리 맵핑 확인

Memory mapping

The memory map is shown in *Figure 5.*

Figure 5. Memory map



iostm32f107xx.h

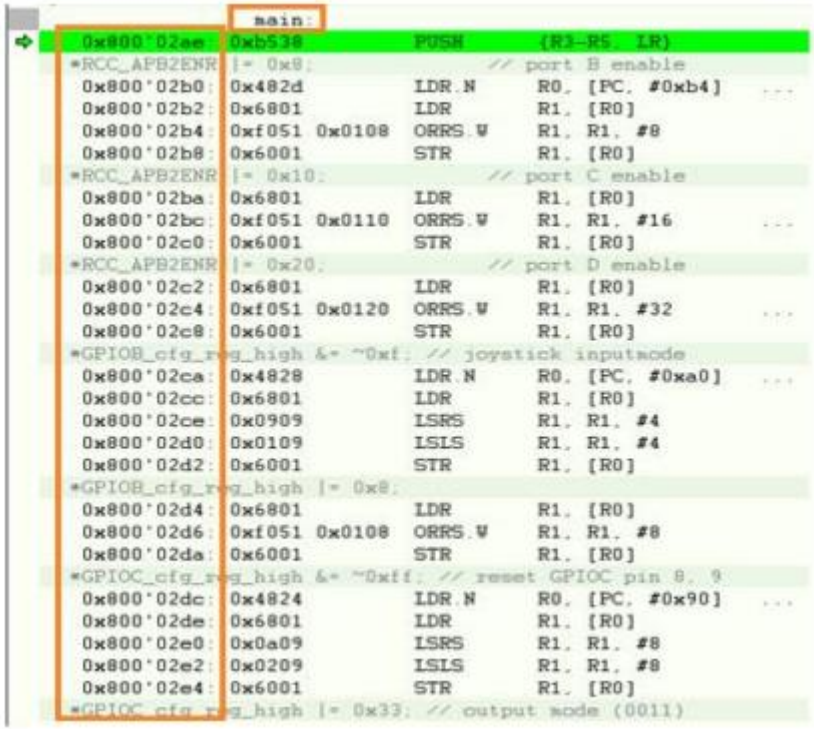
inc > ST > iostm32f107xx.h

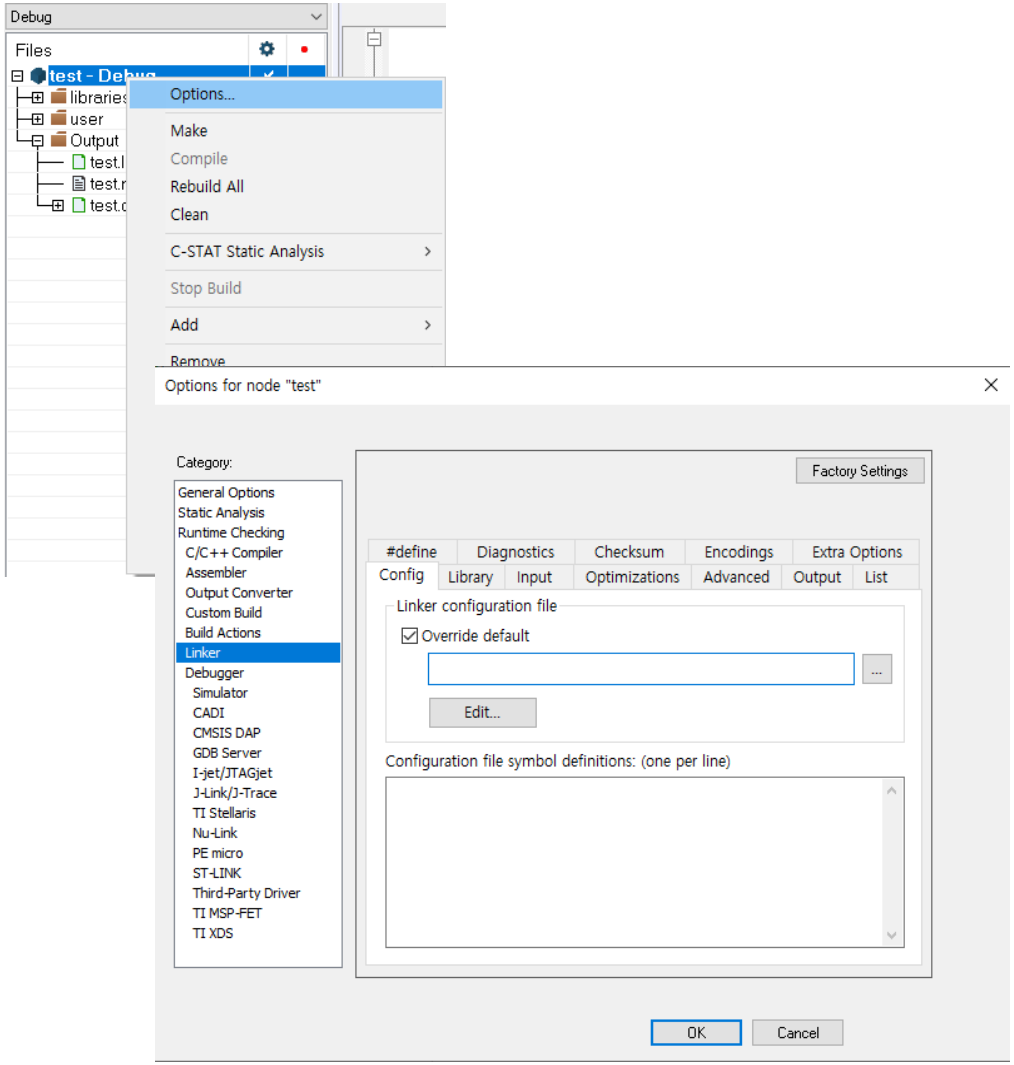
```
4453    __IO_REG32_BIT(DBGMCU_IDCODE,     0xE0042000,__READ       ,__dbgmcu_idcode_bits);
4454    __IO_REG32_BIT(DBGMCU_CR,         0xE0042004,__READ_WRITE ,__dbgmcu_cr_bits);
4455
4456    /***************************************************************************
4457     **
4458     ** PWR
4459     **
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);
4463
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_cfgr2_bits);
4481
4482    /***************************************************************************
```

스캐터 파일 업로드

- 작성한 *.icf 파일

- project 오른쪽 클릭 – Options..
- Linker – Config – Override default
- … 을 눌러 업로드





Interrupt vs Polling

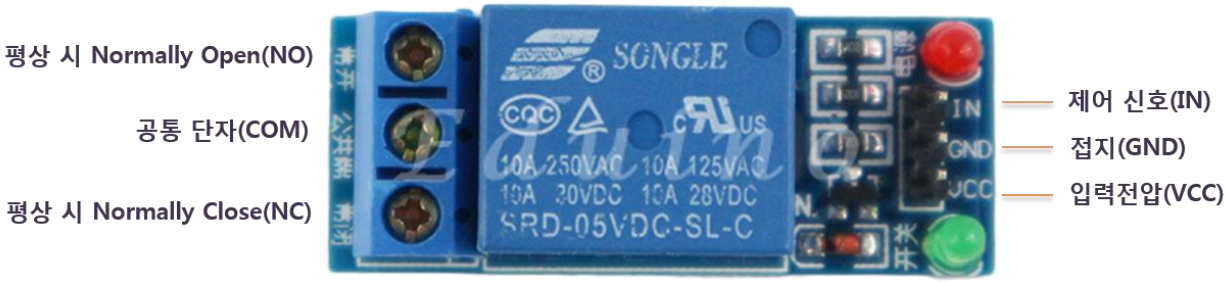
Interrupt

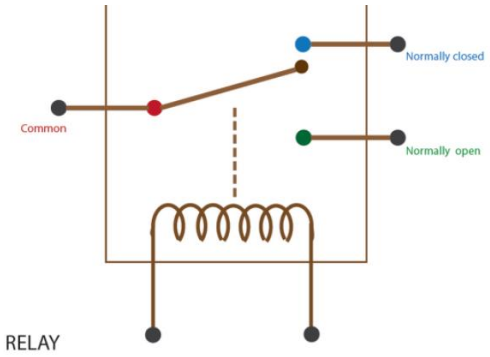
- Hardware의 변화를 감지해 외부로부터 전기신호 입력을 CPU가 알아채는 방법
- CPU 마다 다른 방식으로 동작
- 진행 중인 일을 잠시 멈추고 인터럽트 처리 루틴을 실행하여 신호를 처리함

Polling

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

릴레이 모듈





Relay Module

: 릴레이를 제어하는 모듈

: 전자기유도원리를 이용하여 스위치 역할

: 릴레이에 신호를 가하면 출력 상태(ON/OFF)가 변경된다

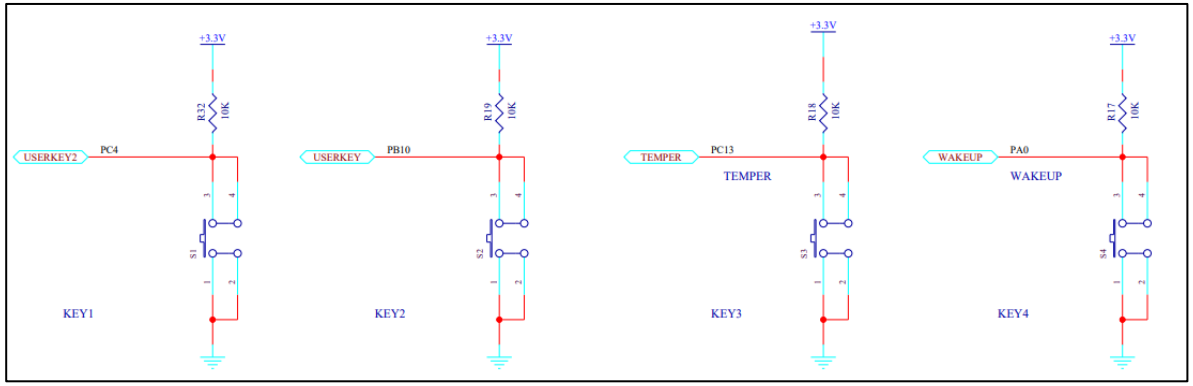
릴레이 모듈에 3.3v 전원 인가해서 사용 (5V 는 작동 안 할 수 있음)

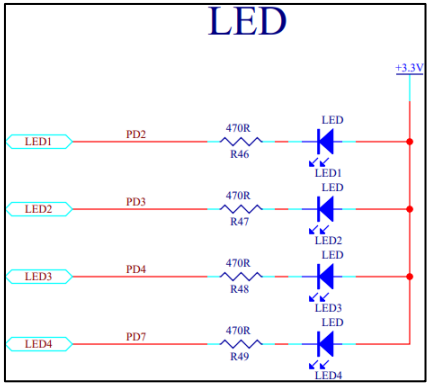
COM은 제어 신호(IN)에 따라 NO 또는 NC로 붙는다

NO: 평소에 open, high 신호가 들어오면 close

NC: 평소에 close, high 신호가 들어오면 open

LED, 버튼 회로도





4주차 실험 릴레이 모듈 사용을 위한 필수 GPIO : PC4, PB10, PC13, PA0

릴레이 모듈의 Input(보드 output) 핀 : 조별 선택

실험 주의사항

- 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 폴더는 백업 후 반드시 삭제해주세요.
- 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

릴레이 모듈과 연결된 pin 을 set 한 뒤 delay() 함수 호출하고 다시 reset 하는 방식으로 구현

```
void delay() {
  int i;
  for (i = 0; i < 10000000; i++) {}
}
```

실험 검사

오늘 검사 받을 수 있는 조는 오늘 받고 못 받는 조는 따로 미션 수행 후 다음 주 수업 시작할 때 검사

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

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