**REPORT**



**제목 : 3주차 실험결과 보고서**

**수강과목 : 임베디드 시스템 설계 및 실험**

**조 원 : 201524414 기호영**

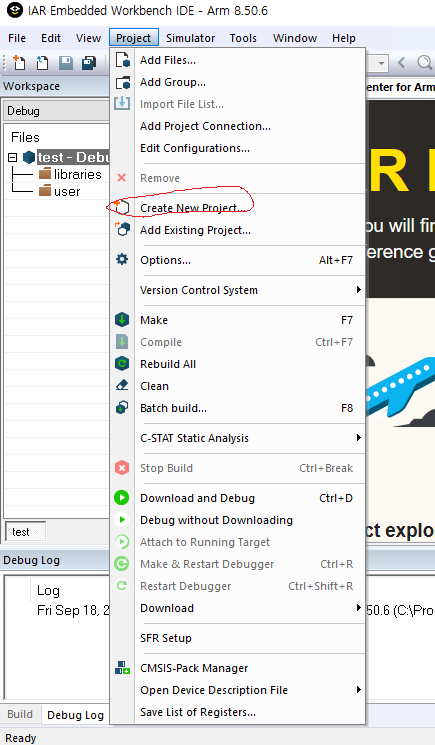
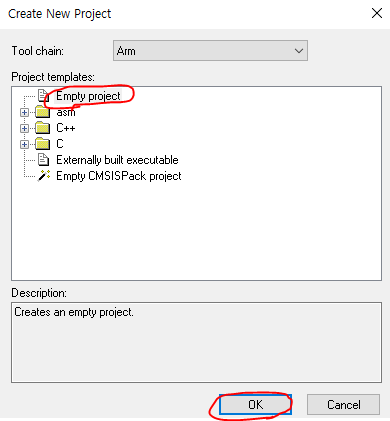
**201524437 김요한**

**201624410 권선근**

**201824461 남지원**

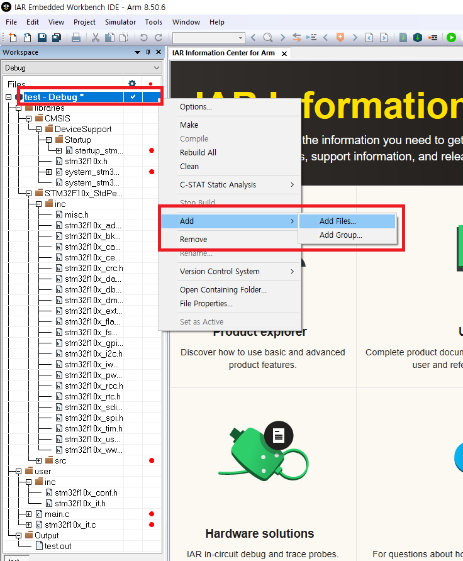
**제출일자 : 2019.09.18**

1. **실험 목표**
   1. 임베디드 시스템의 기본 원리 습득
   2. 레지스터와 주소 제어를 통한 임베디드 펌웨어 개발 이해
2. **실험 환경**
3. JTAG & J-LINK  
    JTAG는 Joint Test Action Group의 약자로 Embedded system 개발 시에 이용되는 Debugging 장비다. CPU의 기계어 코드를 실행하지 않고 MCU 내부의 플래시 메모리나 임베디드 장치에서 CPU의 외부 플래시 메모리에 코드를 쓰거나 읽을 수 있다. J-LINK는 JTAG가 발전한 형태로 보다 많은 프로세스에서 디버깅이 가능하다. 특히 SWD(Serial wire debug) 인터페이스를 통해 Serial 통신이 가능하기 때문에 더 적은 I/O핀으로 더 많은 입출력이 가능하게 발전했다. 이번 실험에서는 J-LINK를 사용한다.
4. STM32F107VCT6 DEVELOPMENT BOARD  
    이번 실험에 사용되는 임베디드 보드. 다양한 기능과 장치가 있지만, 이번 실험에서 사용할 것은 key input과, joystick input key, 프로그래밍 가능한 LED 조명을 사용할 것이다.
5. IAR EW for Arm  
   IAR Embedded Workbench. 어셈블러, C 그리고 C++을 이용하여 임베디드 애플리케이션을 빌드하고 디버깅 하는 개발 툴 세트.
6. **실험 방법**
   1. 보드연결
      1. 보드와 J-LINK연결
      2. J-link 본체의 전원을 on해서 Status LED 점등 확인
      3. 보드의 전원을 on해서 Target LED 점등 확인
   2. 환경설정
      1. 제공된 “3주차 실험제공파일”을 프로젝트 폴더에 압축을 푼다.
      2. IAR Embedded Workbench IDE 실행. Project – Create New Project - Empty project – OK - 생성한 프로젝트 폴더에 원하는 프로젝트 명으로 저장

<그림 1. 프로젝트 생성>

* + 1. 프로젝트 오른쪽 클릭 – Add – Add Group으로 프로젝트 폴더의 구조와 같게 생성. 프로젝트 오른쪽 클릭 – Add – Add Files으로 .c, .h, .s 파일 모두 추가.

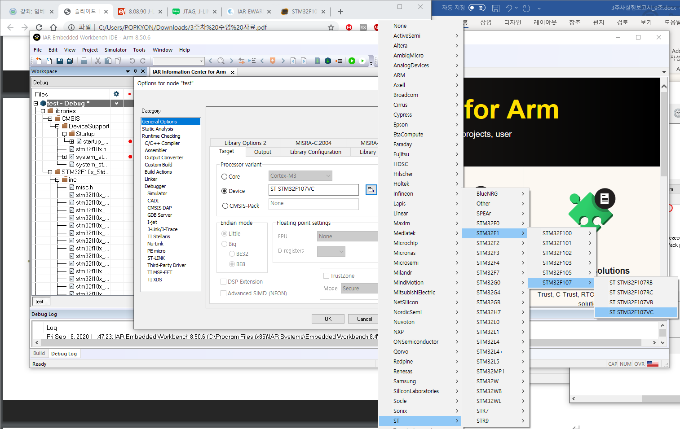
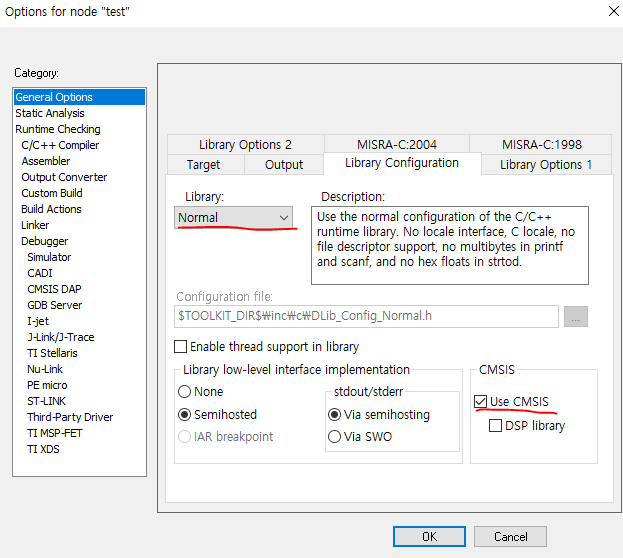


<그림 2. 프로젝트 구조 설정>

* + 1. 프로젝트 옵션 설정: 프로젝트 오른쪽 클릭 – Options
       1. General Options:

Target 탭 – Device: ST – STM32F1 – STM32F107 – ST STM32F107VC 선택

Library Configuration 탭: Library- Normal 선택, CMSIS - Use CMSIS 체크

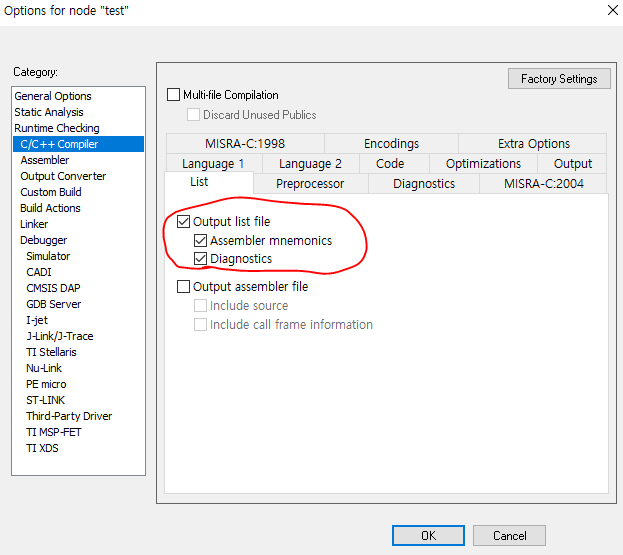
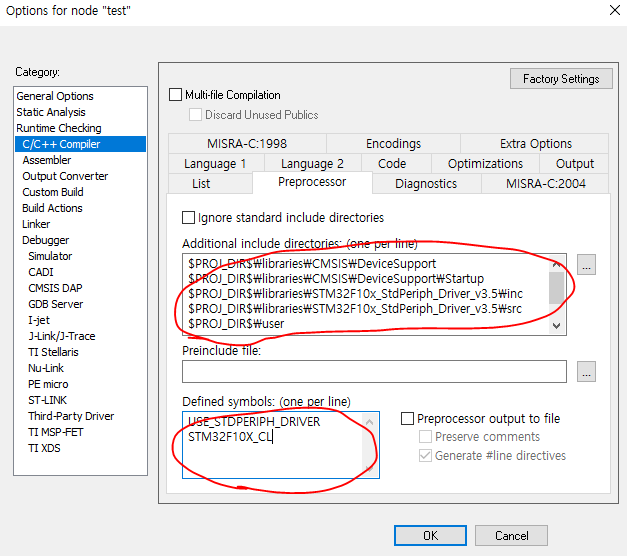
<그림 3. 프로젝트 옵션 설정(General Options)>

* + - 1. Runtime Checking

C/C++ Compiler:

List 탭: Output list file - Assembler mnemonics, Diagnostics 체크

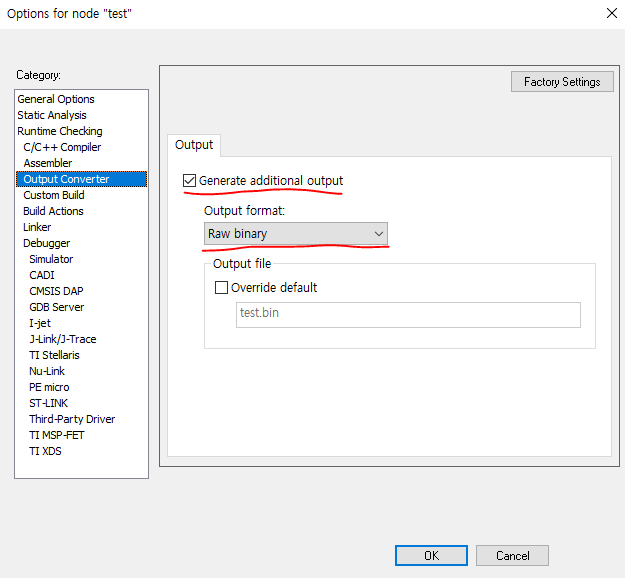
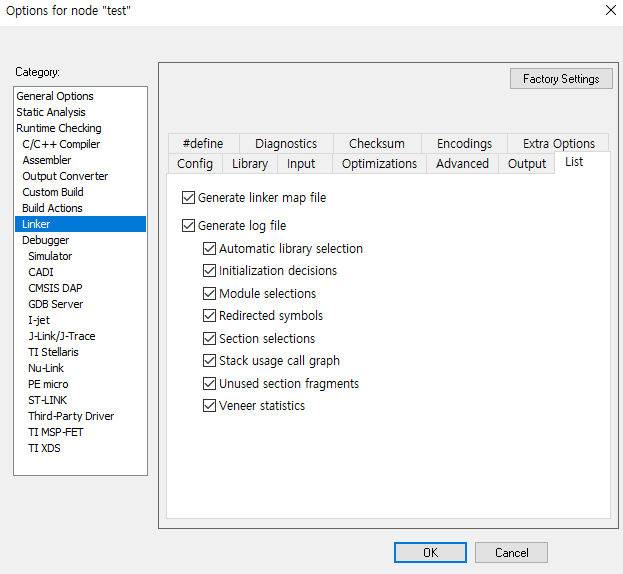
Preprocessor 탭: Additional include directories: (one per line), Defined symbols: (one per line) 입력

<그림 4. 프로젝트 옵션 설정(Runtime Checking – C/C++ Compiler)>

Output Converter: Generate additional output 체크, Output format – Raw binary

Linker- List 탭: 모두 체크

<그림 5. 프로젝트 옵션 설정(Runtime Checking)>

* + 1. 보드 포팅

options - Runtime checking – Debugger:

Setup 탭: Driver – J-link/J-Trace 선택

Download 탭: Verify download, Use flash loader(s) 체크

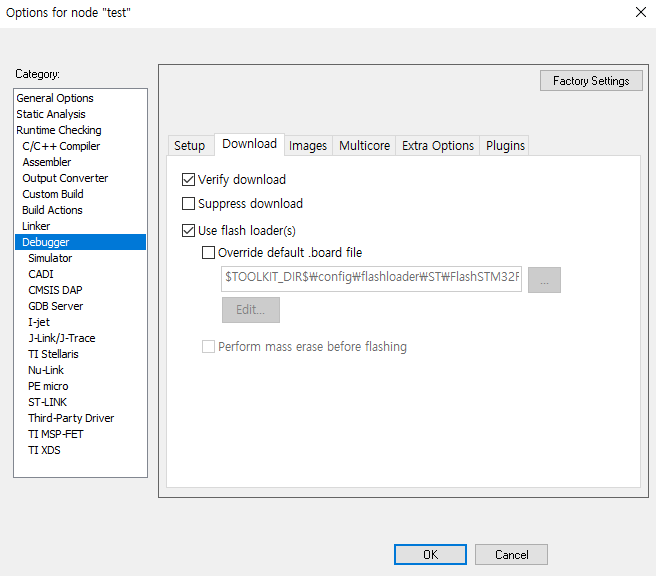
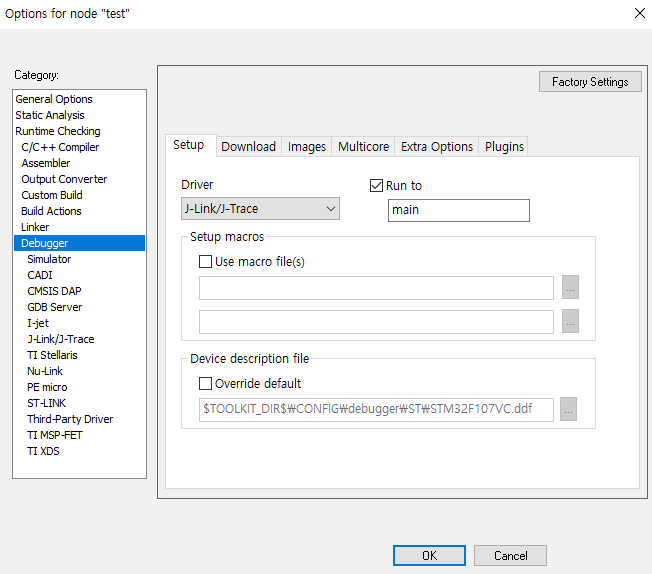


그림 6. 프로젝트 옵션 설정1

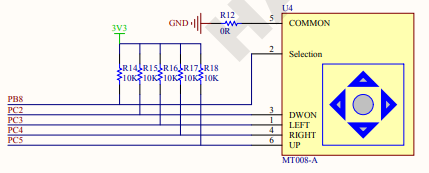
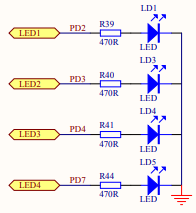
1. 컴파일
2. user/main.c에 코드 작성
3. F7(Make)로 컴파일. (첫 컴파일 시 저장하는 창이 뜸 - 프로젝트 폴더에 프로젝트 명으로 저장)
4. Download and Debug (동의 묻는 창 뜨면 Accept)
5. **실험결과**
   1. 전체 소스코드

|  |
| --- |
| #include "stm32f10x.h"  #define RCC\_APB2\_ENR \*(volatile unsigned int \*)0x40021018  #define PORTB \*(volatile unsigned int \*)0x40010C04  #define PORTB\_set \*(volatile unsigned int \*)0x40010C10  #define PORTB\_data \*(volatile unsigned int \*)0x40010C08  #define PORTC \*(volatile unsigned int \*)0x40011000  #define PORTC\_set \*(volatile unsigned int \*)0x40011010  #define PORTC\_data \*(volatile unsigned int \*)0x40011008  #define PORTD \*(volatile unsigned int \*)0x40011400  #define SET\_LED \*(volatile unsigned int \*)0x40011410  #define RESET\_LED \*(volatile unsigned int \*)0x40011414  int main(void){  RCC\_APB2\_ENR=0x38;  PORTB &= 0x00000000;  PORTB |= 0x00000004;  PORTB\_set = 0x0;    PORTC &= 0x00000000;  PORTC |= 0x00444400;  PORTC\_set = 0x3C;  PORTD &= 0x00000000;  PORTD |= 0x30033300;  SET\_LED &= 0x00000000;  while(1){  if (~PORTC\_data & 0x20) {  SET\_LED |= 0x1C;  RESET\_LED |= 0x1C;  }  else if(~PORTC\_data & 0x4){  SET\_LED |= 0x98;  RESET\_LED |= 0x98;  }  else if(~PORTC\_data & 0x10) {  SET\_LED |= 0x8C;  RESET\_LED |= 0x8C;  }  else if (~PORTC\_data & 0x8) {  SET\_LED |= 0x94;  RESET\_LED |= 0x94;  }  else if (~PORTB\_data & 0x100) {  SET\_LED |= 0x9C;  RESET\_LED |= 0x9C;  }  }  return 0;  } |

* 1. 소스코드 상세
     1. Memory Address

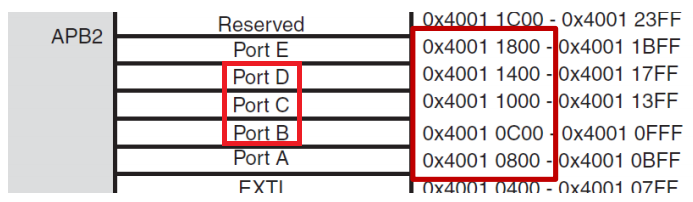
|  |
| --- |
| #define RCC\_APB2\_ENR \*(volatile unsigned int \*)0x40021018  #define PORTB \*(volatile unsigned int \*)0x40010C04  #define PORTB\_set \*(volatile unsigned int \*)0x40010C10  #define PORTB\_data \*(volatile unsigned int \*)0x40010C08  #define PORTC \*(volatile unsigned int \*)0x40011000  #define PORTC\_set \*(volatile unsigned int \*)0x40011010  #define PORTC\_data \*(volatile unsigned int \*)0x40011008  #define PORTD \*(volatile unsigned int \*)0x40011400  #define SET\_LED \*(volatile unsigned int \*)0x40011410  #define RESET\_LED \*(volatile unsigned int \*)0x40011414 |

* + - 1. 조이스틱 & LED

<그림 7. schematic GPIO Port>

이번에 사용하는 임베디드 보드에는 입출력 포트인 GPIO Port가 A에서 G까지 있는데 조이스틱과 LED는 PB, PC, PD와 연결되어 있다. 따라서 이번 실험에서는 B, C, D 포트를 사용한다.

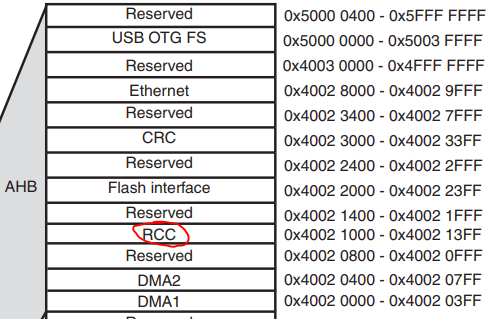


<그림 8. GPIO Port Address>

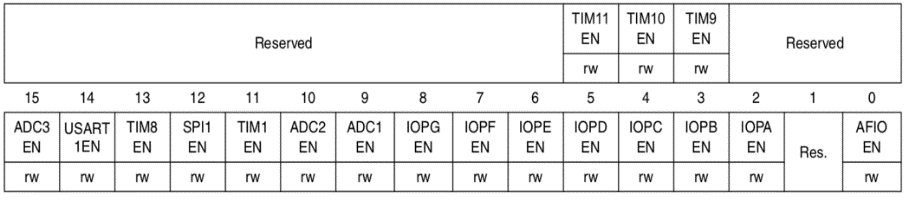
위의 그림에서 보이듯이 각각의 Port는 고유의 Base address를 가진다. 이 Ba se address에 offset을 더해서 원하는 Register를 조작하게 되는 것이다. 그리고 위의 소스 코드에서 define한 값들은 모두 이번에 사용하게 될 Register들의 memory address 값이다. 하지만 GPIO Port를 사용하기위해서 Clock이 할당되어야 한다. 위의 그림에서 base address를 알 수 있다.

**Port B: 0x40010C04, Port C: 0x40011000, Port D: 0x40011400**

* + - 1. GPIO Port 사용







<그림 9. GPIO Port Enable>

Clock을 enable 하려면 RCC 레지스터에서 APB2포트를 enable 시켜야 한다. 따라서 RCC Register의 base address 0x40021000에서 RCC\_APB2ENR의 address인 0x18을 더해 줘야한다. 따라서 Clock을 enable 하려면 0x40021018의 값을 바꾸어야 한다.

이번에 우리가 사용할 포트는 B, C, D이므로, 그림처럼 3, 4, 5번째 bit에 1을 주어야 한다. 따라서 [0000 0011 1000=0x38] 로 Bit Mask 하게 되면 포트를 사용할 수 있게 된다.

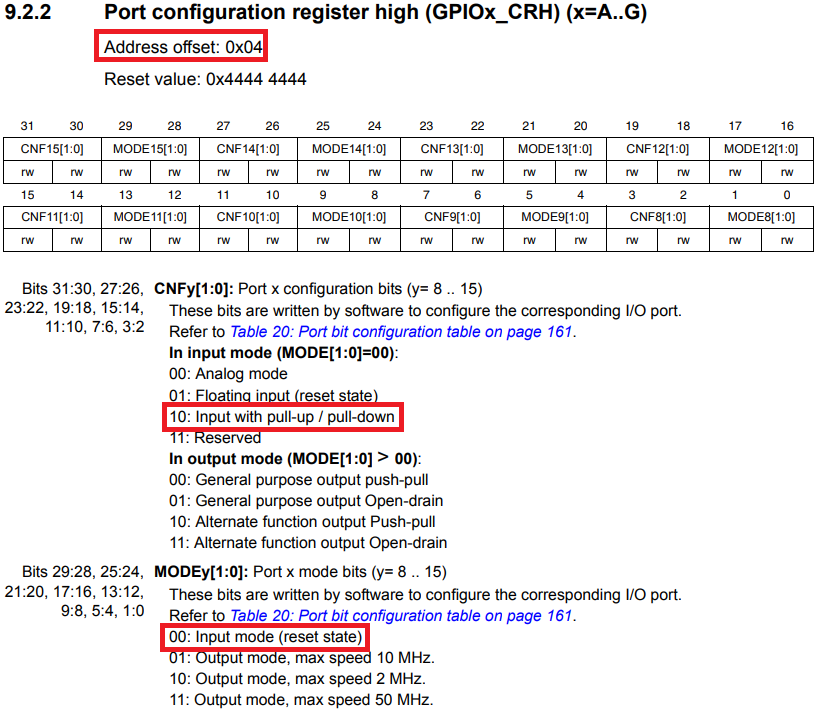
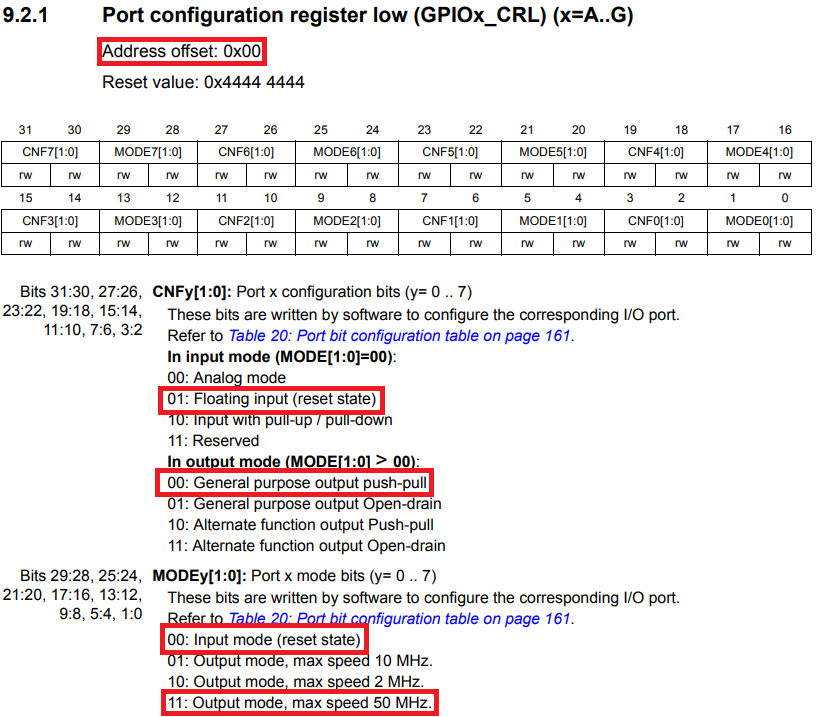
각 Port의 입출력 모드를 설정해 주어야 하는데 조이스틱은 입력이고 LED는 출력이므로, Port B~C는 입력 모드로 Port D는 출력모드로 설정해야 한다.

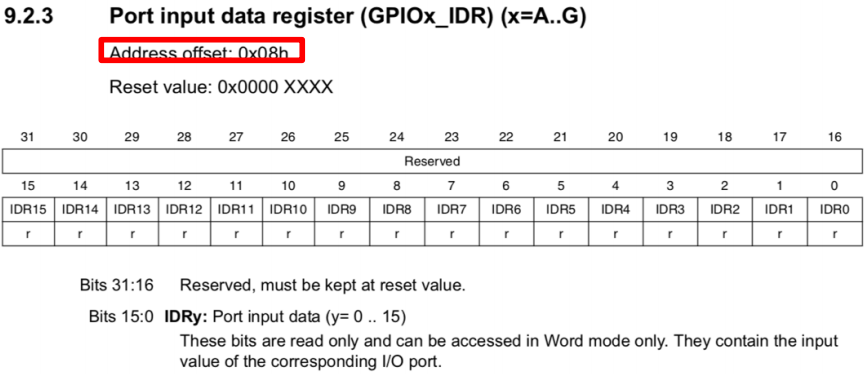
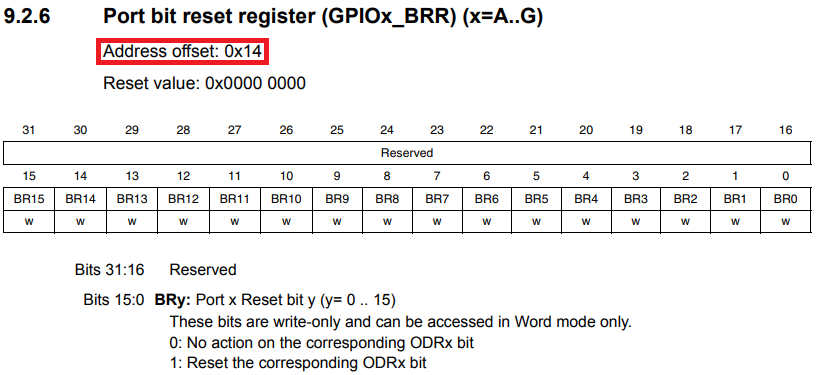
**RCC\_APB2\_ENR: 0x40021018**

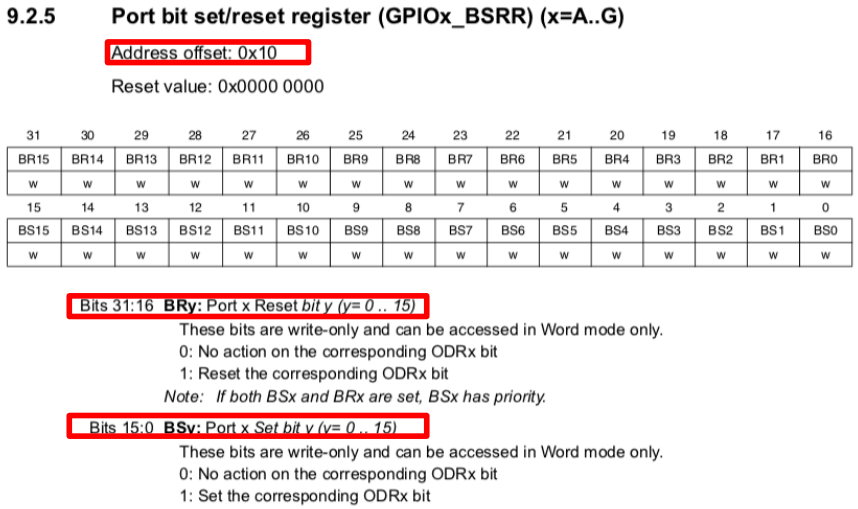
* + 1. Main
       1. Init

|  |
| --- |
| int main(void){  RCC\_APB2\_ENR=0x38;  PORTB &= 0x00000000;  PORTB |= 0x00000004;  PORTB\_set = 0x0;    PORTC &= 0x00000000;  PORTC |= 0x00444400;  PORTC\_set = 0x3C;  PORTD &= 0x00000000;  PORTD |= 0x30033300;  SET\_LED &= 0x00000000;  while(1){  …  }  return 0;  } |

register는 사용하기 위해 각 Port들의 clock을 enable하여야 한다. 따라서 이번 실험에 사용할 Port가 어떤 register를 사용하는지 파악할 필요가 있다.







<그림 10. GPIO Port Enable>

먼저 Port는 위치에 따라 low(0~7)와 high(8~15)로 나눠서 사용된다. <그림 7>에서 LED는 PD 2, 3, 4, 7을 사용하고, 조이스틱은 PB 8, PC 2, 3, 4, 5를 사용하는 것을 알 수 있다. 따라서 이번 실험에서는 PC와 PD는 CRL(Clock Register Low)을, PB는 CRH(Clock Register High)를 사용하여 Port를 Enable하여야 한다. 이때 입출력을 설정할 수가 있다.

또한 나머지 레지스터에 대해 간단히 설명하자면 IDR(Input Data Register)은 조이스틱 입력을 확인할 때 사용하고 BRR(Bit Reset Register), BSRR(Bit Set Reset Register)은 LED제어를 위해 Set하거나 Reset하기위해 사용한다.

먼저 조이스틱의 경우 PC의 Base address(0x40011000)에 CRL의 address offset(0x00)을 더한 register address 0x40011400와 PB의 Base address(0x40010C00)에 CRH의 address offset(0x04)을 더한 register address 0x40010C04를 사용한다. 먼저 해당 register의 값을 &= 0x0 연산으로 초기화 시킨다. 그 다음 PC 2, 3, 4, 5의 CNF에 01(Floating input)와 00(Input mode) 합쳐서 0x0100 = 0x4를 입력해야만 조이스틱이 정상 작동된다. PB 8, 즉 조이스틱 셀렉트 또한 0x0100 = 0x4를 입력해야만 작동된다. 따라서

**0x40010C00 &= 0x00000000, 0x40010C04 &= 0x00000000**

**0x40010C00 |= 0x00444400, 0x40010C04 |= 0x00000004**

다음으로 LED의 경우 PD의 Base address(0x40011400)에 CRL의 address offset(0x00)을 더한 register address 0x40011400을 사용한다. PD 2, 3, 4, 7의 CNF에 00(push-pull)와 11(50MHZ) 합쳐서 0x0011 = 0x3를 입력해야만 LED가 정상 작동된다. 따라서

**0x40011400 |= 0x30033300**

각 Port의 reset을 위해 PB, PC의, PD의 Base address에 BSRR의 address offset(0x10)을 더한 register address를 초기화해야 한다.

0b00111100=0x3C

**PORTB\_set = 0x0, PORTC\_set = 0x3C. SET\_LED &= 0x00000000;**

사용하는 Port들을 enable 하였으므로 조작하는 부분을 작성했다.

1. Loop

|  |
| --- |
| …  while(1){  if (~PORTC\_data & 0x20) {  SET\_LED |= 0x1C;  RESET\_LED |= 0x1C; //off  }  else if(~PORTC\_data & 0x4){  SET\_LED |= 0x98;  RESET\_LED |= 0x98; //off  }  else if(~PORTC\_data & 0x10) {  SET\_LED |= 0x8C;  RESET\_LED |= 0x8C; //off  }  else if (~PORTC\_data & 0x8) {  SET\_LED |= 0x94;  RESET\_LED |= 0x94; //off  }  else if (~PORTB\_data & 0x100) {  SET\_LED |= 0x9C;  RESET\_LED |= 0x9C;  }  }  … |

조이스틱의 입력을 확인하기위해 IDR(+0x8)을 사용한다. 예를 들어 UP으로 조작할 때 PC5를 이용한다. 따라서 입력을 확인하려면 0b00100000 = 0x20인지 확인해야한다.

하지만 실제로 코드를 돌려보면 실행이 되지 않는다. 그 이유는 조이스틱은 default로 입력을 주지 않았을 때 1이 저장되고 조이스틱으로 입력을 했을 때 해당 위치에 0의 값이 저장된다. 따라서 확인을 할 PORTC\_\_data(0x40011008) 값에 ~(not)을 붙여서 확인을 해주어야 한다.

**if (~(0x40011008) & 0x20) { … }**

입력을 확인하였다면 이제 LED로 출력을 시켜주어야 한다. Output Data Register에 값을 쓰려면 BSRR(+0x10)을 사용해야 하는데 BSRR의 상위 16bit(16~31)는 Reset을 위한 bit이고 하위 16bit(0~15)는 Set을 위한 bit임을 알 수 있다. Up의 경우 LED1,2,3을 on 해야 하므로 PD2,3,4를 Set한다. 따라서 0b00011100 = 0x1C을 mask한다.

**0x40011410 |= 0x1C**

입력이 끝난 후에는 LED가 꺼져야 하므로 PORTD를 reset 시켜야 한다. 따라서 Set된 부분을 BRR(+0x14)로 reset한다. UP의 경우 PD2,3,4를 reset해야 하므로 0x1c를 mask한다.

**0x40011414 |= 0x1C**

|  |
| --- |
| if (~PORTC\_data & 0x20) {  SET\_LED |= 0x1C  RESET\_LED |= 0x1C;  } |
| Joystick 설정: 0x20 = 0010 0000 = PC5 = UP  LED 설정: 0x1C = 0001 1100 = PD2,3,4 = LED 1번, 2번, 3번 |
| else if(~PORTC\_data & 0x4){  SET\_LED |= 0x98  RESET\_LED |= 0x98;  } |
| Joystick 설정: 0x4 = 0000 0100 = PC2 = DOWN  LED 설정: 0x98 = 1001 1000 = PD3,4,7 = LED 2번, 3번, 4번 |
| else if(~PORTC\_data & 0x10) {  SET\_LED |= 0x8C;  RESET\_LED |= 0x8C;  } |
| Joystick 설정: 0x10 = 0001 0000 = PC4 = LEFT  LED 설정: 0x8C = 1000 1100 = PD3,4,7 = LED 1번, 3번, 4번 |
| else if (~PORTC\_data & 0x8) {  SET\_LED |= 0x94;  RESET\_LED |= 0x94;  } |
| Joystick 설정: 0x8 = 0000 1000 = PC3 = RIGHT  LED 설정: 0x94 = 1001 0100 = PD2,4,7 = LED 1번, 2번, 4번 |
| else if (~PORTB\_data & 0x100) {  SET\_LED |= 0x9C;  RESET\_LED |= 0x9C;  } |
| Joystick 설정: 0x100 = 0001 0000 0000 = PB8 = SELECT  LED 설정: 0x9C = 1001 1100 = PD2,3,4,7 = LED 1번, 2번, 3번, 4번 |

* 1. 보드 작동

|  |  |
| --- | --- |
|  |  |
| seletion | Down |
|  |  |
| Left | Right |
|  |  |
| UP |

* 1. 오실로스코프 작동

1. **결론**
   1. J-LINK와 IAR EW를 이용하여 임베디드 시스템 디버깅을 이해하고 제어한다.
   2. STM32F107VCT6 보드의 Memory Map을 이해할 수 있다.
   3. 레지스터를 사용하기 위해서 clock, Mode 설정을 할 수 있다.
   4. LED와 같은 모듈들을 GPIO를 통해서 제어할 수 있으며, GPIO PORT에는 레지스터와 주소영역대가 정해져 있다.