2021년 2학기 임베디드 시스템 설계 및 실험

- 3주차 보고서 -

004분반 9조

개요

이번 실험에서는 레지스터와 주소 제어를 통한 GPIO 제어 방법을 알아보았다.

목표

- 임베디드 시스템의 기본 원리 습득
- 레지스터와 주소 제어를 통한 임베디드 펌웨어 개발 이해

세부실험내용

- C핀(조이스틱), D핀(LED)을 사용하기 위해서 RCC 레지스터에 값을 인가한다.
- C핀, D핀에서 어떤 핀을 사용할지, 어떤 클럭을 사용할지, 어떤 스피드로 사용할지 각각 설정한다.
- C핀에서 입력받은 값을 활용하여 해당하는 LED 핀에 신호를 인가한다.

개념

기본적으로 필요한 레지스터에 원하는 값(핀 번호, 특정 상수값 등)을 인가하는 방식으로 진행한다.

- RCC_APB2ENR : 원하는 포트에 클럭을 부여하기 사용하는 레지스터
- GPIOx_CRL(Port configuration register) : 어떤 핀을 사용할 것인지, 어떤 모드를 사용할 건지, 어떤 속도로 사용할 건지 설정하는 레지스터
- GPIOx_IDR (Port input data register) : 입력 값을 읽어들일 때 사용하는 레지스터
- GPIOx_ODR(Port output data register) : 출력 값을 인가할 때 사용하는 레지스터

실험과정

- 세팅

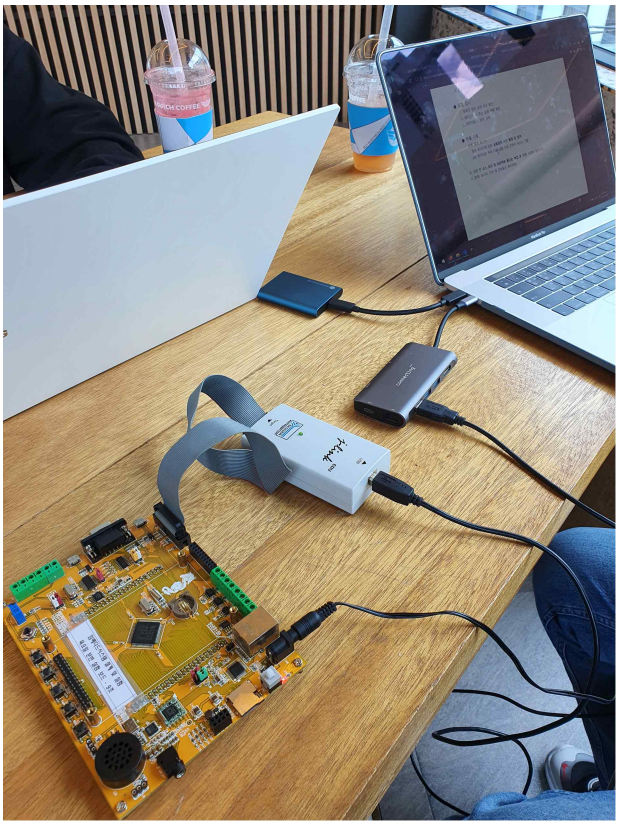

그림 1) '보드 – J-Link – PC' 순으로 연결한다.

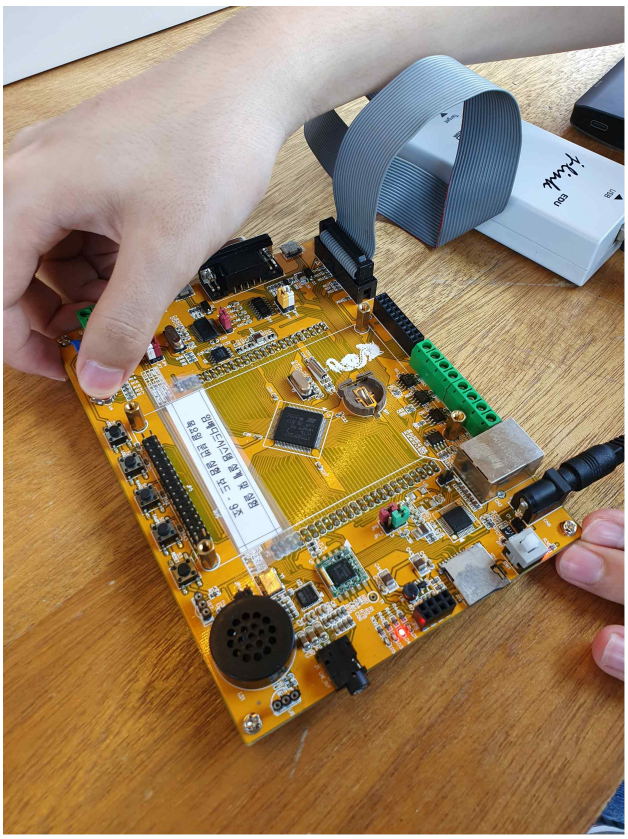

그림 2) IAR IDE를 이용하여 컴파일된 프로그램을 보드에 올린 뒤, 작동을 확인한다.

- 코드

```c
#include "stm32f10x.h"

int main(void) {
  // GPIO 설정을 위한 InitTypeDef 변수 선언
  GPIO_InitTypeDef GPIO_InitInput;
  GPIO_InitTypeDef GPIO_InitOutput;

  // Input(조이스틱) 설정
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
  // Port C의 2, 3, 4, 5번 핀에 대하여 설정
  GPIO_InitInput.GPIO_Pin = GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5;
  // Pull-Up 모드로 설정(스틱을 꺾으면 Low, 가만히 두면 High)
  GPIO_InitInput.GPIO_Mode = GPIO_Mode_IPU;

  // OutPut(LED) 설정
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
  // Port D의 2, 3, 4, 7번 핀에 대하여 설정
  GPIO_InitOutput.GPIO_Pin = GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_7;
  // Push-Pull 모드로 설정(High가 인가되면 1, Low가 인가되면 0)
  GPIO_InitOutput.GPIO_Mode = GPIO_Mode_Out_PP;
  // 저전력 모드인 2MHz로 설정
  GPIO_InitOutput.GPIO_Speed = GPIO_Speed_2MHz;

  // 위에서 지정한 설정대로 각 포트를 초기화
  GPIO_Init(GPIOC, &GPIO_InitInput);
  GPIO_Init(GPIOD, &GPIO_InitOutput);

  // 초기화 이후 아래 동작을 계속 반복
  while (1) {
    uint16_t ledPin;  // 점등할 LED의 핀 번호를 저장할 변수
    uint16_t inputPin = GPIO_ReadInputData(GPIOC); // 입력된 조이스틱의 핀 값

    // 원하는 핀 번호를 얻기 위해 AND 연산과 XOR 연산 사용
    inputPin &= (GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5);
    inputPin ^= (GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5);

    // 입력된 조이스틱의 방향에 따라 점등할 LED 결정
    switch (inputPin) {
    case GPIO_Pin_2:
      ledPin = GPIO_Pin_3;
      break;
    case GPIO_Pin_3:
      ledPin = GPIO_Pin_4;
      break;
    case GPIO_Pin_4:
      ledPin = GPIO_Pin_7;
      break;
    case GPIO_Pin_5:
      ledPin = GPIO_Pin_2;
      break;
    default:
      // 조이스틱의 입력이 없을 경우 점등하지 않음
      ledPin = (uint16_t)0x0000;
      break;
    }

    // 위에서 결정된 LED만 점등
    GPIO_Write(GPIOD, ledPin);
  }
  return 0;
}
```

결론

이번 실습에서는 편의를 위해 직접 레지스터 상수값을 사용하는 대신 stm32f10x.h 헤더파일의 함수를 사용하였다.

또한 본 코드의 while문 안에서 처음에 inputPin을 초기화하였을 때 레지스터의 값 0과 1이 반대로 설정되어서 원하는 핀 번호를 얻기 위해 핀 값과 초기화된 값을 AND연산으로 모든 핀 값을 0으로 만든 후 XOR연산으로 더해주었다.

코드의 가독성이나 길이, 실습의 편리 등을 위해서라도 헤더파일의 사용은 필수적이라고 느꼈다.