임베디드 시스템 설계 및 실험 **3** 주차 실험 보고서

월요일 분반 2 조

조원**:** 정경호(201524566)**,** 이재욱(201724539), 정현모(201724579), 이해민(201814331)

실험 결과

1. **RCC ENABLE**

A. RCC 초기주소 0x40021000 에서 clock enable register 의 offset 0x18 을 더해준 주소 0x40021018 에 저장된 값을 포트 B, C, D 를 활성화시켜주도록 바꿔준다. 그 값은 0b00111000 으로 0x38 이다. 해당하는 위치의 값만 바꿔 주기 위해 |= OR 연산을 사용한다

2. **사용 포트 ENABLE**

A. 각 포트들은 사용 전에 ENABLE 을 시켜주어야 한다. 각 포트의 사용 여부는 GPIO register 에서 확인하고 조절할 수 있다. GPIO 레지스터의 기본값은 0x4444 4444 로, Input mode (Floating input)로 되어있다. 포트 b 와 c 는 Input mode(Input with pull-up / pull-down)으로, 포트 d 는 output mode 50MHz(General purpose output push-pull)로 지정해 주었다.

B. GPIO register 를 조작할 때 주의할 점은 기본 reset 값이 0x0000 이 아닌 0x4444 라는 점이다. 0x4444 가 아니더라도 이전 과정에서 GPIO register 를 수정해주었을 가능성 이 있으므로 우리가 사용할 비트만 비워 놓고 나머지는 그대로 유지해 주어야 한다. 우 리는 해당 문제를 사용하지 않는 비트는 1, 사용할 비트는 0 으로 채워서 AND 연산을 통해 해결하였고, 우리가 넣고 싶은 `값을 사용하지 않는 비트는 0, 사용할 비트는 1 을 채워서 OR 연산을 통해 GPIO register 를 제어하였다.

3. 조이스틱 UP, DOWN, LEFT, RIGHT 제어 및 LED 점등

A. UP 의 경우, 조이스틱의 상하좌우를 담당하는 port C 의 초기 주소 0x40011000 에서 Input data 가 저장되는 register(IDR)의 offset 0x08 을 더한 주소인 0x40011008 에서 조이스틱의 UP 의 값이 0 인지 체크해야 한다. UP 의 경우엔 5 번째 주소를 참조하므로, (1 << 5) shift 연산으로 0b00010000 값을 만들어준 뒤, ~(NOT)연산으로 input 값을 반전시키고, &(AND)연산으로 0 인지를 체크한다.

B. 조건문 if 를 통해 0 인 것이 확인되면, OUTPUT 을 담당한 port D 의 초기 주소 0x40011400 에서 set/reset register(BSRR)의 offset 0x10 을 더한 주소 0x40011410 에서 4 번, 7 번 LED 만 set 값을 1 로 바꿔 놓아야 한다. 따라서 0b10010000 = 0x90 을 0x40011410 에 |=연산으로 넣어준다.

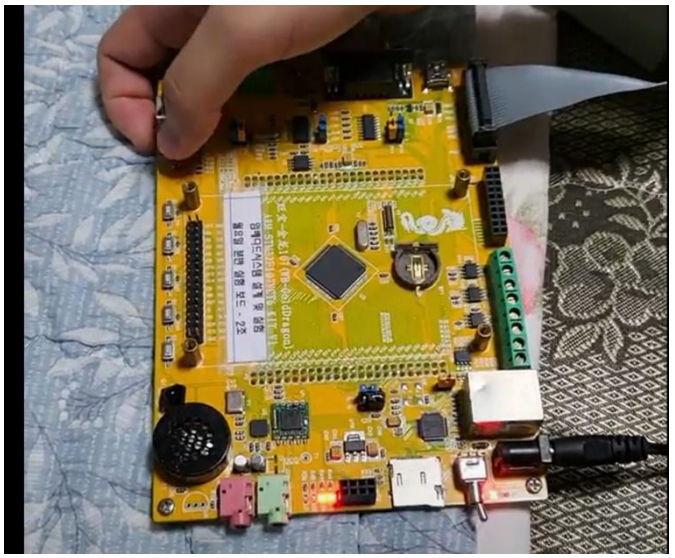

그림 1 : 조이스틱 UP 을 통해 상단 LED2 개가 켜진모습

C. LEFT 의 경우에도, 3 번째 값을 참조하므로 (1 << 3)과 &연산을 해주고, 2 번, 3 번 LED 를 켜주어야 하므로 0b00001100 = 0x0c 로 설정한다.

D. DOWN 과 RIGHT 의 경우, 각각 2 번째, 4 번째 주소를 참조하며, BSRR 에서 reset 부분을 변경해야 한다.
reset 부분은 set 의 위치보다 4byte 상위에 있고, DOWN 의 경우, 2 번 3 번 LED 를 꺼야 하므로 0x90 를 4 번 lshift 한 0x00900000 으로 설정하고, RIGHT 는 0x0c 를 옮긴 0x000c0000 을 연산해준다.

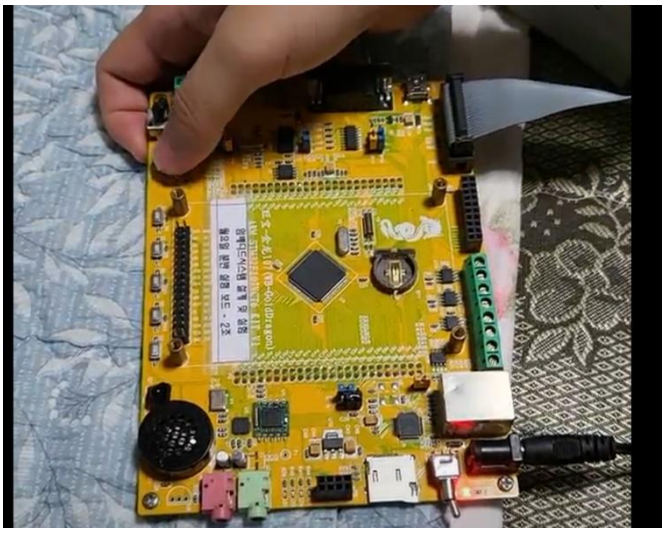

그림 2 조이스틱 DOWN 을 통해 다시 LED 를 끈 모습

1. 조이스틱 SELECTION 제어 및 LED 점등

a. SELECTION 버튼을 제어하기 위해 연관 포트인 PB8 을 제어해야 했다. 우리는 PB8 에서 Input 이 들어올 때 값의 변화를 알아보기 위해 GPIO_IDR 를 사용했다. IDR 의 offset 은 0x08 로 포트 B 의 주소 값인 0x40010c00 에서 8 을 더해 IDR 를 확인할 수 있었다. 포트 B 의 IDR 값 중 8 번포트의 값이 궁금했으므로 0x40010c08 주소 값이 가리키는 레지스터에서 8 번 비트를 확인해야 했다. 확인을 위해 0x1 값을 << (shift operation)을 8 번 해주어서 8 번 비트가 True 인지 체크하고 True 값이라면 LED 를 제어 할 수 있게끔 if 문을 만들었다.

b. SELECTION Input 이 들어오면 모든 LED 가 반대가 되어야 한다. 따라서 UP, DOWN, LEFT, RIGHT 에 썼던 BSRR 대신 ODR 을 사용한다면 더 직관적이게 동작할 수 있겠다고 생각했다. LED 의 ODR 주소 값인 0x4001140C 를 찾아가서 우리가 사용하는 LED 포트인 PD2, PD3, PD4, PD7 을 xor 연산으로 반전시켜주었다. 다른 비트들은 0 으로 채워서 우리의 XOR 연산에 영향을 받지 않도록 코드를 작성하였다.

c. 하지만 이 경우 IDR 에서 CLOCK 만큼 True 문이 돌고 있으므로 랜덤으로 동작을 하지 않으려면 토글 방식으로 제작해줄 필요가 있었다. 우리는 간단한 flag 변수로 해당 문제를 해결했다. flag 는 Input 값이 True 인지 False 인지를 저장하는 변수이다. 한번 Input 이 True 가 되었다면 Input 이 False 로 바뀔 때까지 XOR 연산을 해줄 필요가 없다. 따라서 Input 이 True 라면 한번 XOR 연산을 해주고 False 가 되면 flag 를 다시 리셋 해주고, Input 이 True 라면 다시한번 XOR 연산을 해주는 방식으로 제작하였다.

그림 3 조이스틱 SELECTION 으로 꺼진 LED 를 모두 반전시킨 모습

소스코드

```
#include "stm32f10x.h"
int main(void)
{
  *((volatile unsigned int *)0x40021018) |= 0x38; // RCC clock enable B, C, D

  *((volatile unsigned int *)0x40010c04) &= 0xfffffff0; // port b init
  *((volatile unsigned int *)0x40010c04) |= 0x8; // port b set to input

  *((volatile unsigned int *)0x40011000) &= 0xff0000ff; // port c init
  *((volatile unsigned int *)0x40011000) |= 0x00888800; // port c set to input

  *((volatile unsigned int *)0x40011400) &= 0x0ff000ff; // port d init
  *((volatile unsigned int *)0x40011400) |= 0x30033300; //port d set to output

  unsigned int flag = 0;
  while(1)
  {
    if ((~(*((volatile unsigned int *)0x40010c08)) & (1 << 8))) // port b 8 IDR CENTER
    {
      if (!flag) // if flag is not reversed(0)
      {
        *((volatile unsigned int *)0x4001140C) ^= 0x0000009c; // port d 2 3 4 7 REVERSE
        flag = 1; // make flag reversed(1)
      }
    }
    else
    {
      if (flag) // if flag is reversed(1)
      {
        flag = 0; // make flag not reversed(0)
      }
    }

    if ((~(*((volatile unsigned int *)0x40011008)) & (1 << 5))) // port c 5 IDR UP
    {
      *((volatile unsigned int *)0x40011410) |= 0x00000090; // port d 4 7 ON
    }

    if ((~(*((volatile unsigned int *)0x40011008)) & (1 << 3))) // port c 3 IDR LEFT
    {
      *((volatile unsigned int *)0x40011410) |= 0x0000000c; // port d 2 3 ON
    }

    if ((~(*((volatile unsigned int *)0x40011008)) & (1 << 4))) // port c 4 IDR RIGHT
    {
      *((volatile unsigned int *)0x40011410) |= 0x000c0000; // port d 2 3 OFF
    }

    if ((~(*((volatile unsigned int *)0x40011008)) & (1 << 2))) // port c 2 IDR DOWN
    {
      *((volatile unsigned int *)0x40011410) |= 0x00900000; // port d 4 7 OFF
    }
  }
}
```