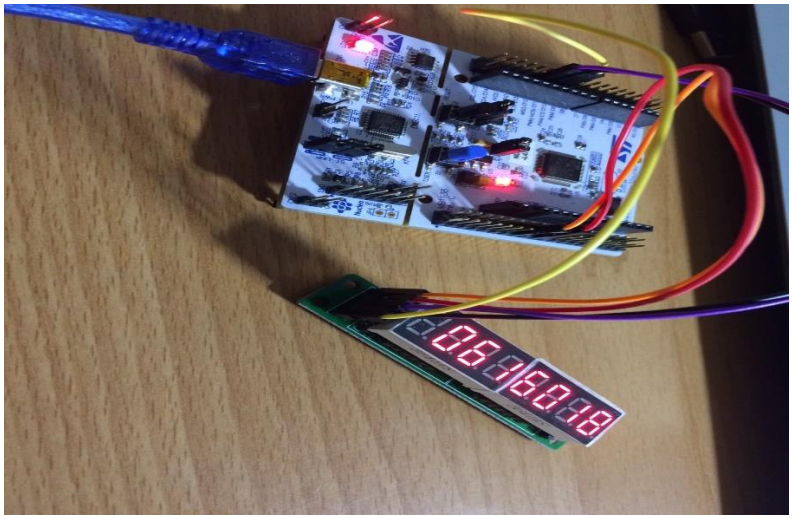
# 微處理機 Lab5

學號 ： 0616018
姓名 ： 林哲宇

## Lab5.1: Max7219 displayer

這題要把 Lab4 的第二題改寫成 C 的版本。也就是說，要從 C 中呼叫之前寫的 asm。

線路接好大概長這樣，其實和 Lab4 一模一樣。把 max 接給電源、接地，以及 GPIOA 的 pin5, 6, 7。



首先要把.text 中的 main 刪掉，改成要被呼叫的三個函數

```
4  .text
5      .global max7219_init
6      .global max7219_send
7      .global gpio_init
8      .equ RCC_AHB2ENR, 0x4002104C
9      .equ GPIOA_MODER, 0x48000000
10     .equ GPIOA_OSPEEDR, 0x48000008
11     .equ GPIOA_ODR, 0x48000014
12     .equ MAX7219_DIN, 0b00100000
13     .equ MAX7219_CS, 0b01000000
14     .equ MAX7219_CLK, 0b10000000
15     .equ BSRR, 0x48000018
16     .equ BRR, 0x48000028
```

Max7219_init, max7219_send, gpio_init 三個函式基本上跟 Lab4 大同小異。

```
18 gpio_init:
19     push {r0, r1, r2, lr}
20     //TODO: Initialize three GPIO pins as output for max7219 DIN, CS and CLK
21     movs r0, 0b01
22     ldr r1, =RCC_AHB2ENR
23     str r0, [r1]
24 
25     ldr r1, =GPIOA_MODER
26     ldr r0, [r1]
27     and r0, 0b11111111111111110000001111111111
28     orr r0, 0b00000000000000000101010000000000
29     str r0, [r1]
30 
31     movs r0, 0b1010100000000000
32     ldr r1, =GPIOA_OSPEEDR
33     str r0, [r1]
34 
35     movs r0, 0
36     ldr r1, =GPIOA_ODR
37     str r0, [r1]
38     pop {r0, r1, r2, lr}
39     bx lr
40 
41 max7219_send:
42     push {r0, r1, r2, r3, r4, r5, r6, r7, lr}
43     ldr r2, =MAX7219_DIN
44     ldr r3, =MAX7219_CS
45     ldr r4, =MAX7219_CLK
46     ldr r5, =BSRR
47     ldr r6, =BRR
48     //input parameter: r0 is ADDRESS , r1 is DATA
49     //TODO: Use this function to send a message to max7219
50     lsl r0, r0, 0x8
51     adds r0, r0, r1
52     movs r7, 0b1000000000000000
53     bl Loop16
54     pop {r0, r1, r2, r3, r4, r5, r6, r7, lr}
55     bx lr
```

```
Loop16:
    // set the clock to 0
    str r4, [r6]
    // check r7 bit is 0 or 1 and then set DIN
    tst r0, r7
    beq Clear
    str r2, [r5]
    b Done
    Clear:
        str r2, [r6]
    Done:
        // set the clock to 1
        lsr r7, r7, 1
        str r4, [r5]
        // check if all 16 bits are in and set CS from 0 to 1
        cmp r7, 0
        bne Loop16
        str r3, [r6]
        str r3, [r5]
    bx lr
max7219_init:
    //TODO: Initialize max7219 registers
    push {r0, r1, r2, r3, r4, r5, r6, r7, lr}
    // Decode Mode
    movs r0, 0b1001
    movs r1, 0xff
    bl max7219_send
    // Intensity
    movs r0, 0b1010
    movs r1, 0b1111
    bl max7219_send
    // Scan Limit
    movs r0, 0b1011
    movs r1, 0b110
    bl max7219_send
    // Shutdown
    movs r0, 0b1100
    movs r1, 1
    bl max7219_send
    // Display Test
    movs r0, 0b1111
    movs r1, 0
```

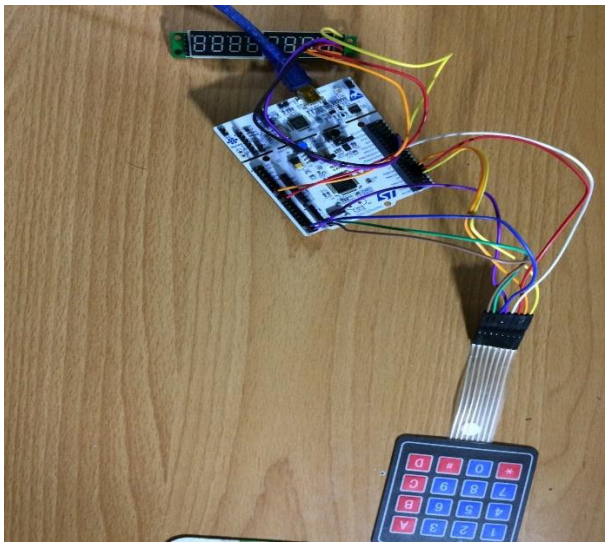
最後是 C 的部分，main 中就是簡單的 call 那三個用 arm 寫的 function，display 就是將 data 和 address 傳至 max7219_send，並且將結果印在板子上。

```
//These functions inside the asm file
extern void GPIO_init();
extern void max7219_init();
extern void max7219_send(unsigned char address, unsigned char data);

/* TODO: Show data on 7-seg via max7219_send
int display(int data, int num_digs)
{
    int i;
    for (i = 1; i <= num_digs; i++)
    {
        if(data >= 0){
            max7219_send(i , data % 10);
            data /= 10;
        }
        else max7219_send(i , 15);
    }
    for (i = num_digs ; i <= 8; i++)
        max7219_send(i, 15);
    return (data > 99999999) ? -1 : 0;
}

int main()
{
    int student_id = 616018;
    GPIO_init();
    max7219_init();
    display(student_id, 8);
}
```

## Lab5.2: KeypadScanning

這題多了一個新玩具—keypad，基本上原理就是透過掃 column 和 row 去偵測使用者按下了什麼。

我把 input 接給 GPIOB 的 pin3, 4, 5, 6，把 output 接給 GPIOC 的 pin0, 1, 2, 3，其他的部分跟第一題一樣。



程式碼部分比較不一樣的部分就是 keypad 的 GPIOB, GPIOC 需要初始化，另外就是需要持續偵測使用者是否有按下按鈕。

```
19 void keypad_init()
20 {
21     RCC->AHB2ENR   |= 0b110;
22
23     GPIOC->MODER   &= 0b11111111111111111111111100000000;
24     GPIOC->MODER   |= 0b00000000000000000000000001010101;
25     GPIOC->PUPDR   &= 0b11111111111111111111111100000000;
26     GPIOC->PUPDR   |= 0b00000000000000000000000001010101;
27     GPIOC->OSPEEDR &= 0b11111111111111111111111100000000;
28     GPIOC->OSPEEDR |= 0b00000000000000000000000001010101;
29     GPIOC->ODR     |= 0b00000000000000000000000000001111;
30
31     GPIOB->MODER   &= 0b11111111111111111100000000111111;
32     GPIOB->PUPDR   &= 0b11111111111111111100000000111111;
33     GPIOB->PUPDR   |= 0b00000000000000000001010100000000;
34     GPIOB->OSPEEDR &= 0b11111111111111111111111100000000;
35     GPIOB->OSPEEDR |= 0b00000000000000000000000001010101;
36 }
```

監測使用者輸入的部分是用 Output 去掃描 input，如果 input 讀出來是 1 就 return。要將四個 column 都掃描一次。

```
signed char keypad_scan()
{
    int i, j;
    for(i=0; i<=3; i++){
        for(j=0; j<=3; j++){
            GPIOC->BRR = x_pin[j];
        }
        GPIOC->BSRR = x_pin[i];
        for(j=0; j<=3; j++){
            if(GPIOB->IDR & y_pin[j]) return table[i][j];
        }
    }
    return -1;
}
```

基本上就是這樣，還有其他細節，例如要處理當 input 有兩位數的情形等等。

## Lab5.3: multi buttons 處理多按鍵

這題比較麻煩一點，因為跟前兩題不太一樣，這題還有牽扯到一些硬體的問題。

首先，第三題跟第二題的差別就在於會有兩個按鍵同時按下去，因此寫法要稍微改一下。要記錄第一個偵測到的按紐，之後再偵測一次。

```
void multi(){
    int first=-1,second=-1,r,c;
    while(1){
        int count=0;
        int keypad_row=0,col=0;
        char key_val;

        for(col=0;col<4;col++){
            for(keypad_row=0;keypad_row<4;keypad_row++){


                if(!is_pressed){
                    display_clr(2);
                    continue;
                }
                key_val=keypad_value[keypad_row][col];
                count++;
                if(count==2){
                    second=key_val;
                    display(second+first,second+first>=10?2:1);
                    keypad_scan2(r,c);
                    }


                }


            }
        }
```

第二就是電壓不足 threshole 的問題。如果第一個按鍵被放開，就會離開迴圈，回第一個地方。

## 心得：

這次作業有兩個新觀念，第一個是要在 C 中嵌入 asm，還有就是 keypad 的使用。前者沒什麼大問題，簡單易懂。然而，keypad 需要時間理解。在處理第三題時，要怎麼處理兩個按鍵同時按的情況又會有一些問題。總之這次作業每個坑跳好跳滿，但是還是挺有趣

的，期待下次作業的內容。