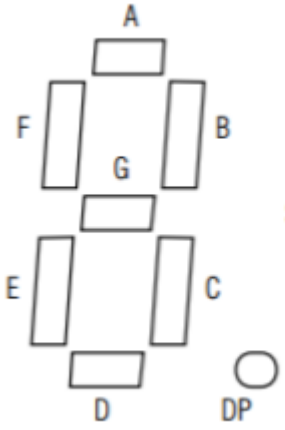
Lab4 7-Seg LED

1. Lab objectives 實驗目的

- Understand the principle of using MAX7219.
- Design the program of 7-Seg LED.
- 了解 MAX7219 使用原理
- 設計 7-Seg LED 程式

2. Lab theory 實驗原理



STANDARD 7-SEGMENT LED

Table 1. Serial-Data Format (16 Bits)

DP A B C D E F G

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

Str 進去 Max7219(pa5=Din, pa6=LS, pa7=CLK)的 data 格式如上：
D7~D0 是塞進去的 data，D11~D8 是拿來定址的，例如：

- 0x01,…,0x09 是第 1,…,第 8 顆 7-seg
- 0x09: Decode，D<x>是 1 就代表第<x+1>顆 7-seg 是 Decode Mode。
- 在 Decode Mode 下只看 D3~D0 的值，只有 16 種顯示可用 (0~9,-,e,h,l,p,<blank>)，
  不在 Decode Mode 時比較自由，你可以透過定址 + data，來決定哪一顆的哪一段 LED 亮。
  - ✧ 例如 Decode Mode 時把 0x037b (=0000 0011 0111 1011) 透過 send function Str 進去 pa5，可以讓第 4 顆 7-seg 亮字母 E。

- ✧ 不在 Decode Mode 時把 0x037b (=0000 0011 0111 1011) 透過 send function Str 進去 pa5，可以讓第 4 顆 7-seg 亮數字 9。

- 0x0A: Intensity，亮度 0~15(所以只有 D3~D0 的值有用)
- 0x0B: Scan-Limit 顯示幾個 7-seg (1~8，所以只有 D2~D0 的值有用)
- 0x0C: Shutdown，只有 D0 值有用。1 為正常，0 為 Shutdown mode 所有 7-seg LED 會關掉，是一種省電模式
- 0x0F: Display Test，只有 D0 值有用，0 為正常，1 為 Test mode，會讓所有燈全亮
- Send function 要模擬 clk，把 data 一步步的傳進去 Max7219

見 lab4_note 課程講義。

3. Steps 實驗步驟

3.1. Lab4.1: Max7219 與 7-Seg LED 練習 without decode mode

將 stm32 的 3.3V 接到 7-Seg LED 板的 VCC，GND 接到 GND，並選擇三 GPIO 接腳分別接到 DIN、CS 和 CLK。並利用 GPIO 控制 Max7219 並在 7-Seg LED 上的第一位依序顯示 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F (時間間隔 1 秒)，。

了解上面如何運作後就簡單了，只要設定都對就沒問題了。

- 把 GPIO 開好(這次用 pa5, pa6, pa7)

```
GPIO_init:
    //TODO: Initialize three GPIO pins as output for max7219 DIN, CS and CLK
    movs    r0, #0x1 //0000000(0001) :active A block
    ldr     r1, =RCC_AHB2ENR
    str     r0, [r1]

    movs    r0, #0x5400 //00005400 = 0000(0101 0100 0000 0000)
    ldr     r1, =GPIOA_MODER
    ldr     r2, [r1]
    //mask: keep from modifying the origin setting
            //FFFF03FF = FFFF(0000 0011 1111 1111)
    and     r2, #0xFFFF03FF //mask
    orrs    r2, r2, r0
    str     r2, [r1]

    movs    r0, #0xA800 //0000(1010 1000 0000 0000)
    //(10):high speed
    ldr     r1, =GPIOA_OSPEEDR
    strh    r0, [r1]


    BX LR
```

- 把 Max7219 設定好
  - ✧ DECODE：00，全部不 decode
  - ✧ DISPLAY：0，不 test
  - ✧ SCAN_LIMIT：0，第一題只要 1 位，顯示不必要的會因為殘留的 data 導致顯示亂碼
  - ✧ INTENSITY：10，隨便一個值
  - ✧ SHUNDOWN：0，不 shutdown

```
max7219_init:
    // Din is connneted to
    // operate pa7 to set D
    //TODO: Initialize max7
    push {r0, r1, r2, lr}
    // In DECODE_MODE: 依照表
    // Not: 將D0~D7直接顯示在7-
    ldr r0, =DECODE_MODE //
    ldr r1, =0x00 //no deco
    BL MAX7219Send
    ldr r0, =DISPLAY_TEST /
    ldr r1, =0x0
    //0:normal operation
    //1:test
    BL MAX7219Send
    ldr r0, =SCAN_LIMIT //s
    ldr r1, =0x00 //0x00~0x
    BL MAX7219Send
    ldr r0, =INTENSITY //br
    ldr r1, =0xA
    BL MAX7219Send
    ldr r0, =SHUTDOWN
    ldr r1, =0x1
    //0:shutdown
    //1:operation
    BL MAX7219Send
    pop {r0, r1, r2, lr}
    BX LR
```

```
.data
    arr: .byte 0x7e, 0x30, 0x6d, 0x79, 0x33, 0x5b, 0x5f, 0x70, 0x7f, 0x7b, 0x77, 0x1f, 0x4e, 0x3d, 0x4f, 0x47
    //0,1,2,3,4,5,6,7,8,9,A,b,C,d,E,F
```

- 基本上就是把已經寫好的值一個個塞好
- 注意在 BL 過的函式內跳別的 BL 時，一定要先把以後需要的值(特別是 lr) push 進 stack，之後回來再 pop。

```
Display0toF:
    //TODO: Display 0 to F at first
    ldr r2, =arr
    mov r3, #0

    Loop:
    ldrb r1, [r2, r3]
    ldr r0, =DIGIT_0
    push {r2, r3, lr}
    BL MAX7219Send
    BL delay
    pop {r2, r3, lr}
    add r3, r3, #1
    cmp r3, #16
    bne Loop

    BX LR
```

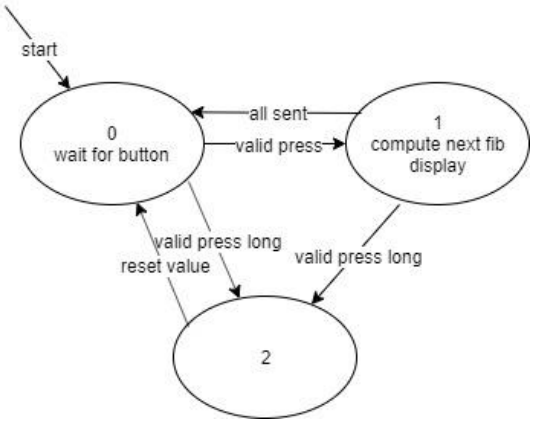
3.2 Lab4.2: Max7219 與 7-Seg LED 練習

(Practice of Max7219 and 7-Seg LED) — use code B decode mode (30%)

- 要以 decode B mode 顯示學號，得將每個 bit 開啟 decode B mode，做法是在 send decode mode 的 address 時，data 設為 FF(0b11111111)，除此之外，為了要同時印出 7 個數字，要將 scan limit 設為 0x6。
- 將學號以連續的方式存在 data 區段，當讀入後直接以其值當作 data，配合要印的 address 送至 Max7219，執行七次後表示七個數字皆完成。

3-3 設計一程式偵測實驗板上的 User button，當 User button 按 N 次時，7-Seg LED 上會顯示 fib(N) 的值。User button 長按 1 秒則將數值歸零。
fib(0) = 0、fib(1) = 1、fib(2) = 1 、…
若 fib(N) ≥ 100000000 則顯示-1。(超過 8 位)

- State:0:等按鈕；1:算下一個數字：2:重置




```
mov r0, #0x01
mov r1, #0
push {r0, r1, lr}
BL MAX7219Send
pop {r0, r1, lr}
add r0, r0, #1
mov r1, #15
L:
    push {r0, r1, lr}
    BL MAX7219Send
    pop {r0, r1, lr}
    add r0, r0, #1
    cmp r0, #0x09
    bne L
```

- 一開始先開好 SCAN_LIMIT=7，並顯示空白跟初始值。如果一開始 SCAN_LIMIT=0 還要一直回去調，很麻煩。

- Debounce 長度用下圖設定感覺效果挺不錯，不會有按鈕遲鈍的問題。

```
do_pushed:
//debounce delay
    LDR   R0, =150
    L3: LDR R5, =250
    L4: SUBS R5, #1
    BNE   L4
    SUBS  R0, #1
    BNE   L3
```

長按的功能：

- 把之前過濾後續按按鈕的程式稍微修改，只要一直按的話就會一直累加 r3，當 r3 超過一定值之後就會跳離，並修改 state 狀態。

```
mov   r3 ,#0
mov   r4, #1
lsl   r4, #18
pressing:
    add   r3, #1
    cmp   r3, r4
    beq   reset
    ldr   r2, =GPIOC_IDR
    ldr   r0, [r2]
    ands  r0, r0, r1
    beq   pressing
    bx lr
reset:
    ldr r0, =state//#modi
    mov r1, #2
    str r1, [r0]
out: bx lr
```

- 因為長按視為重置，所以把後續過濾長按的功能移到最前面，如果 state=2 代表是從 init 跳來的，所以要執行過濾長按的程式部分。

```
Fibo:

    mov r0, #0x01
    mov r1, #0
    push {r0, r1, lr}
    BL MAX7219Send
    pop {r0, r1, lr}
    add r0, r0, #1
    mov r1, #15
    L:
        push {r0, r1, lr}
        BL MAX7219Send
        pop {r0, r1, lr}
        add r0, r0, #1
        cmp r0, #0x09
        bne L
    //eliminate pressing
    ldr r0, =state
    ldr r1, [r0]
    cmp r1 ,#2
    bne init //for first pass
    movs  r1, #1
    lsl   r1, #13
    ldr   r2, =GPIOC_IDR
    Keep_pressing:
        ldr   r0, [r2]
        ands  r0, r0, r1
        beq   Keep_pressing
    init:
```

其他要注意的：
要用 loop 把要改的 digit 一個個給 Max7219。

4. Feedback 實驗心得或建議

- Too many hardware breakpoints

這次漸漸發現設斷點很好用，即使是斷點的狀態，按按鈕也是有反應的。不過要注意斷點數有上限。

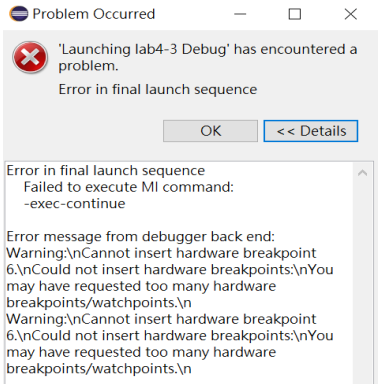


Table 2. Register Address Map

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xXA
Scan Limit	X	1	0	1	1	0xXB
Shutdown	X	1	1	0	0	0xXC
Display Test	X	1	1	1	1	0xXF

- 給出的位址超出 01~08 範圍，發生全亮，猜測可能是不小心塞給了 test mode(D11~D8=0x0F) 的 D0 = 1，因為當時寫的是 loop address 0~15 去塞，要注意 address 0x00 是 non op，address 0x01~ 0x09 才是 7-seg 的位址，
- 以及 address offset 和 decode mode 不要搞混，一個是 address，一個是 data。

Table 5. Code B Font

7-SEGMENT CHARACTER	REGISTER DATA						ON SEGMENTS = 1							
	D7*	D6–D4	D3	D2	D1	D0	DP*	A	B	C	D	E	F	G
0		X	0	0	0	0		1	1	1	1	1	1	0
1		X	0	0	0	1		0	1	1	0	0	0	0
2		X	0	0	1	0		1	1	0	1	1	0	1
3		X	0	0	1	1		1	1	1	1	0	0	1
4		X	0	1	0	0		0	1	1	0	0	1	1
5		X	0	1	0	1		1	0	1	1	0	1	1
6		X	0	1	1	0		1	0	1	1	1	1	1
7		X	0	1	1	1		1	1	1	0	0	0	0
8		X	1	0	0	0		1	1	1	1	1	1	1
9		X	1	0	0	1		1	1	1	1	0	1	1
—		X	1	0	1	0		0	0	0	0	0	0	1
E		X	1	0	1	1		1	0	0	1	1	1	1
H		X	1	1	0	0		0	1	1	0	1	1	1
L		X	1	1	0	1		0	0	0	1	1	1	0
P		X	1	1	1	0		1	1	0	0	1	1	1
blank		X	1	1	1	1		0	0	0	0	0	0	0

*The decimal point is set by bit D7 = 1