實驗名稱：實驗五 MAX7219 與 7-Seg LED

實驗目的：了解MAX7219使用原理、設計7-Seg LED程式

實驗步驟、結果與問題回答：

5.1 Max7219 與 7-Seg LED 的練習—without code B decode mode

將 stm32 的 3.3V 接到 7-Seg LED 板的 VCC，GND 接到 GND，並將 PA0、PA1、PA4 三個 GPIO 接腳分別接到 DIN、CS 和 CLK。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)。

```
36 main:
37     BL GPIO_init
38     BL max7219_init
39     loop:
40         BL Display0toF
41         B loop
```

一開始先用 GPIO_init 初始化各個接腳，再用 max7219_init 初始化 7-Seg LED 的顯示設定。

```
43 GPIO_init:
44     //TODO: Initialize three GPIO pi
45     // enable AHB2 clock
46     ldr r0, =RCC_AHB2ENR
47     movs r1, #0b1 // enable port A
48     str r1, [r0]
49
50     // set PA0, 1, 4 as output mode
51     ldr r0, =GPIOA_MODER
52     ldr r1, [r0]
53     ldr r2, =clear
54     and r1, r2
55     ldr r2, =moder
56     orrs r1, r1, r2
57     str r1, [r0]

59     // set PA0, 1, 4 as high speed mode
60     ldr r0, =GPIOA_SPEEDR
61     ldr r1, [r0]
62     ldr r2, =clear
63     and r1, r2
64     ldr r2, =speed
65     orrs r1, r1, r2
66     str r1, [r0]
67
68     // set PA0, 1, 4 as pull-down output
69     ldr r0, =GPIOA_PUPDR
70     ldr r1, [r0]
71     ldr r2, =clear
72     and r1, r2
73     ldr r2, =pull
74     orrs r1, r1, r2
75     str r1, [r0]
76     BX LR
```

```
127 max7219_init:
128     //TODO: Initialize max7219 registers
129     push {lr}
130     ldr r0, =#DECODE_MODE
131     ldr r1, =#0x0 // not using decode mode
132     BL MAX7219Send
133     ldr r0, =#DISPLAY_TEST
134     ldr r1, =#0x0
135     BL MAX7219Send
136     ldr r0, =#SCAN_LIMIT
137     ldr r1, =0x0 // digit 0
138     BL MAX7219Send
139     ldr r0, =#INTENSITY
140     ldr r1, =#0xA
141     BL MAX7219Send
142     ldr r0, =#SHUTDOWN
143     ldr r1, =#0x1
144     BL MAX7219Send
145     pop {pc}
```

之後就重複執行 Display0toF，每秒顯示一個數字。

```
78 Display0toF:
79     //TODO: Display 0 to F at first digit on 7-SEG LED. Display one per second.
80     push {lr}
81     movs r2, #0 // r2 = i
82     ldr r3, =arr // get the beginning of the array
83     display_loop:
84         adds r0, r3, r2
85         ldrb r1, [r0] // put the 7 segment data into r1
86         movs r0, 0x1 // set r0 to 1 (setting digit 0)
87         bl MAX7219Send
88         bl Delay
89         adds r2, #1
90         cmp r2, #16 // loop until 0~F are all displayed
91         bne display_loop
92     pop {pc}
```

其中 MAX7219Send 用來將要顯示的資料傳給 7-Seg LED，ADDRESS 放在 r0、DATA 放在 r1。

```
94 MAX7219Send:
95     //input parameter: r0 is ADDRESS , r1 is DATA
96     //TODO: Use this function to send a message to max7219
97     push {r2, r3, lr}
98     lsl r0, r0, #8
99     add r0, r0, r1 // r0 contains address and data now
100
101    ldr r1, =#GPIOA_BASE
102    ldr r2, =#LOAD
103    ldr r3, =#DATA
104    ldr r4, =#CLOCK
105    ldr r5, =#GPIO_BSRR_OFFSET
106    ldr r6, =#GPIO_BRR_OFFSET
107    mov r7, #16 // r7 = i
108 max7219send_loop:
109    mov r8, #1
110    sub r9, r7, #1
111    lsl r8, r8, r9 // r8 = mask
112    str r4, [r1,r6] // HAL_GPIO_WritePin(GPIOA, CLOCK, 0); // clk 0
113    tst r0, r8
114    beq bit_not_set // bit not set
115    str r3, [r1,r5] // data 1
116    b if_done
117 bit_not_set:
118     str r3, [r1,r6] // data 0
119 if_done:
120     str r4, [r1,r5] // clk 1
121     subs r7, r7, #1
122     bgt max7219send_loop
123     str r2, [r1,r5] // load 1
124     str r2, [r1,r6] // load 0
125     pop {r2, r3, pc}
```

Delay 用來製造每個數字間 1 秒的間隔。

```
147 Delay:
148     //TODO: Write a delay 1sec function
149     push {r0, lr}
150     ldr r0, =T
151     delay_loop:
152         subs r0, #1
153         bne delay_loop
154     pop {r0, pc}
```

因為結果是動態的，所以就等 DEMO 時再展示。

5.2 Max7219 與 7-Seg LED 的練習—use code B decode mode

利用 GPIO 控制 Max7219 並在 7-Seg LED 上顯示自己的學號(使用 decode mode)。

一開始一樣先用 GPIO_init 和 max7219_init 進行初始化。因為學號在 data 裡宣告的時候是用 word 的形式，所以需要先把它用十進位一位一位拆開才能傳給 7-Seg LED 顯示。

```
186 main:
187     BL GPIO_init
188     BL max7219_init
189     //TODO: display your student id on 7-Seg LED
190     movs r0, #7 // r0 = i  ←從第 7 位(decimal)開始處理
191     ldr r2, =student_id
192     ldr r3, [r2] // r3 = student_id  ←學號存在 r3 裡
193     loop:
194         movs r2, #1
195         movs r6, #10
196         subs r4, r0, #1
197         beq cmp_loop
198         mul_loop: // r2 decimal shift  ←決定現在要處理第幾位(decimal)
199             mul r2, r6
200             subs r4, #1
201             bne mul_loop
202         cmp_loop: // determine the value of the digit
203             mul r5, r2, r4
204             cmp r5, r3
205             it le
206             addle r4, #1
207             ble cmp_loop  ←決定這一位的數字
208         subs r1, r4, #1
209         mul r5, r2, r1
210         subs r3, r5
211         bl MAX7219Send
212         subs r0, #1 // to the next digit
213         beq Program_end
214         b loop
```

因為學號有 7 位，所以 loop 總共會進行 7 次，一次設定 7-Seg LED 上的一位數，然後就進 Program_end 停下來。將要顯示的數字傳給 7-Seg LED 的 MAX7219Send 和上一個實驗一樣，所以就不再貼一次。

最後顯示的結果如下圖：

5.3 Max7219 與 7-Seg LED 的練習—顯示 Fibonacci 數

設計一組語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。例如fib(0) = 0、fib(1) = 1、fib(2) = 1⋯ 若fib(N) ≥ 100000000則顯示-1。

```
main:
    BL GPIO_init
    BL max7219_init
    movs r3, #0 // initialize N
    movs r4, #0 // initialize fib
    movs r0, #0x1
    b display_loop // initialize the 7 segment
    loop:
        ldr r1, [r2] // get the state of button
        movs r0, #1
        lsl r0, #13
        ands r1, r0 // check if the button is pressed
        bne loop
        bl debounce
        bl fib // N in r3, fib in r4
        cmp r4, #-1
        beq display_overflow
        movs r1, #0
        movs r6, #1
        movs r7, #10
        check_digit:
            mul r6, r7
            cmp r4, r6
            it ge
            addge r1, #1
            bge check_digit
        ldr r0, =#SCAN_LIMIT
        BL MAX7219Send
        adds r0, r1, #1
        display_loop:
            movs r6, #1
            movs r7, #10
            subs r5, r0, #1
            beq cmp_loop
            mul_loop: // r6 decimal shift
                mul r6, r7
                subs r5, #1
                bne mul_loop
            cmp_loop: // determine the value of the digit
                mul r8, r5, r6
                cmp r8, r4
                it le
                addle r5, #1
                ble cmp_loop
            subs r1, r5, #1
            mul r8, r1, r6
            subs r4, r8
            bl MAX7219Send
            subs r0, #1 // to the next digit
            beq loop
            b display_loop
```

←分別用 r3 跟 r4 存 N 和 fib (N)

←初始化 7-Seg LED 所以先顯示一次

←檢查 button 有沒有被按下

←debounce 並檢查有沒有按到 1 秒

←算出改變過後的 N 的 fib(N)

←處理如果 fib(N)超過 1000000 的狀況

←依照 fib(N)的位數調整 7-Seg LED 顯示的位數

display_loop 的部分和上一個實驗的 loop 的部分是一樣的，就不再重複解釋。

debounce 用來處理 button 被按到之後的反應。

```
489 debounce: //~25 cycle
490     push {r0, r2, r4, lr}
491     ldr r0, =debounce_num  ←將扣完需要 1 秒定值存進 r0
492     debounce_loop:
493         ldr r1, [r2] // get the state of button
494         movs r4, #1
495         lsl r4, #13
496         ands r1, r4 // check if the button is pressed
                                                    ←檢查 button 有沒有持續被按下
497         bne release  ←處理當 button 被放開時的動作
498         cmp r0, #0
499         beq debounce_loop // pressed more than 1s
                                                    ←當 r0=0 代表 button 已經被按超過一秒
500         subs r0, #1 // pressed less than 1s ←button 還被按不到 1 秒的話 r0 減 1
501         b debounce_loop
502
503 release:
504     ldr r1, =debounce_limit
505     cmp r0, r1
506     it gt
507     popgt {r0, r2, r4, r5}
                                                    ←如果 button 被按的時間少於 debounce_limit 則視為 bouncing，不做反應
508     bgt loop // bouncing
509     cmp r0, #0
510     it eq // pressed for 1s
511     moveq r3, #0
                                                    ←當 button 被按超過一秒就將 N 歸零
512     it ne // pressed less than 1s
513     addne r3, #1
                                                    ←被按的時間大於 debounce_limit 但少於 1 秒就將 N 加 1
514     pop {r0, r2, r4, pc}
```

display_overflow 直接進行在 7-Seg LED 上顯示-1 需要的動作，然後回到等 button 被按的地方。

```
554 display_overflow:
555     ldr r0, =#SCAN_LIMIT
556     ldr r1, =0x1
557     BL MAX7219Send
                                                    ←7-Seg LED 顯示兩位數
558     movs r0, #0x2
559     movs r1, #0xA
560     BL MAX7219Send
                                                    ←7-Seg LED 第 2 位顯示-
561     movs r0, #0x1
562     movs r1, #0x1
563     BL MAX7219Send
                                                    ←7-Seg LED 第 1 位顯示 1
564     b loop
```

fib 和 lab 3 時一樣，將 r3 中的 N 對應到的 fib(N)存進 r4。

```
516 fib:
517     push {r0, r1, r2, r3, lr}
518     cmp r3, #0
519     it eq
520     moveq r4, #0                    ←fib(0) = 0
521     it eq
522     popeq {r0, r1, r2, r3, pc}
523     cmp r3, #1
524     it eq
525     moveq r4, #1                    ←fib(1) = 1
526     it eq
527     popeq {r0, r1, r2, r3, pc}
528     cmp r3, #2
529     it eq
530     moveq r4, #1                    ←fib(2) = 1
531     it eq
532     popeq {r0, r1, r2, r3, pc}
533     sub r3, #2
534     movs r1, #1
535     movs r2, #1
536     fib_loop:
537         ldr r5, =overflow
538         adds r0, r1, r2             ←計算 fib(N) = fib(N-2) + fib(N-1)
539         cmp r0, r5                    如果結果超過 100000000 的話 r4 裡存-1
540         it ge
541         movge r4, #-1
542         it ge
543         popge {r0, r1, r2, r3, pc}
544         sub r3, #1
545         cmp r3, #0
546         it eq                       ←檢查算到 r3 裡的 N 了沒
547         moveq r4, r0
548         it eq
549         popeq {r0, r1, r2, r3, pc}
550         movs r1, r2
551         movs r2, r0
552         b fib_loop
```

這個實驗的結果也是動態的，所以同樣等 DEMO 時再展示。

心得討論與應用聯想：

這次的實驗卡最久的地方是第一個實驗的時候。一開始接線有點搞錯，7-Seg LED 詳細是怎麼運作的又不太清楚，而且原本 7-Seg LED 一接上電就會全亮，弄一弄又不亮了，一度以為是接觸不良還調整了很久，結果根本只是因為灌了沒寫好的程式。後面的兩個程式寫起來就順利不少，感覺他們都是建構在各自前一個實驗的基礎之上的，所以寫起來比較像是在增加功能的感覺，最後第三個實驗甚至把 lab3 的 fibonacci 撿回來用了。據老師說這是最後一個要用 assembly 寫的作業了，雖然還是對它應該還是稱不上很熟練，但是至少算是有基本的認識，弄懂它的運作邏輯了吧。