實驗五 7-Seg LED

0410137劉家麟 0416324 胡安鳳

# 實驗目的

* 了解MAX7219使用原理
* 設計7-Seg LED程式

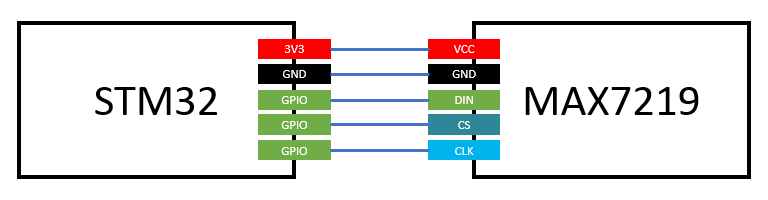
# 實驗原理

請參考上課lab5\_note講義。

# 實驗步驟

## Lab5.1: Max7219與7-Seg LED練習—without code B decode mode

將stm32的3.3V接到7-Seg LED板的VCC，GND接到GND，並選擇三個GPIO接腳分別接到DIN、CS和CLK。



完成以下程式碼，並利用GPIO控制Max7219並在7-Seg LED上顯的第一位依序顯示0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F (時間間隔1秒)，範例影片如下：

<https://goo.gl/ZDZcdl>

Note:由於decode mode無法顯示AbCdF等字，因此請將decode mode關掉。(參考lab5\_note講義的table 6)

Connect 3.3V and GND pin on STM32 to VCC and GND port on MAX7219. Choose three GIPO ports on STM32 for DIN, CS and CLK on MAX7219.

Complete the code giving below and display 0, 1, 2, 3…, 9, A, b, C, d, E, F to the first digit of 7-Seg LED at 1 second interval. Example video link is giving above.

Note: Due to the fact that decode mode is unable to display alphabets, please disable decode mode(ref: lab5\_note table 6).

|  |
| --- |
| .syntax unified  .cpu cortex-m4  .thumb  .data  arr: .byte 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0 //TODO: put 0 to F 7-Seg LED pattern here  .text  .global main  main:  BL GPIO\_init  BL max7219\_init  loop:  BL Display0toF  B loop  GPIO\_init:  //TODO: Initialize three GPIO pins as output for max7219 DIN, CS and CLK  BX LR  Display0toF:  //TODO: Display 0 to F at first digit on 7-SEG LED. Display one per second.  BX LR  MAX7219Send:  //input parameter: r0 is ADDRESS , r1 is DATA  //TODO: Use this function to send a message to max7219  BX LR  max7219\_init:  //TODO: Initialize max7219 registers  BX LR  Delay:  //TODO: Write a delay 1sec function  BX LR |

1. **GPIO\_init**Set GPIO here, just like LAB4.

|  |
| --- |
| RCC\_AHB2ENR -> enable GPIOA = 0x1  GPIOA\_MODER -> set PA5, 6, 7 as output mode  =(GPIOA\_MODER & 0xFFFF03FF) | 0b0101010000000000  GPIOA\_OSPEEDER -> set PA5, 6, 7 as high speed  =(GPIOA\_OSPEEDER & 0xFFFF03FF) | 0b1010100000000000 |

1. **MAX7219\_init**Set decode mode (whether decode or not), intensity (brightness of 7-segment), scan limit (how many digit to display), shut down, and display test.

|  |
| --- |
| DECODE(0xX9) -> No Decode(0x0) = 0x1900  DISPLAY\_TEST(0xXF) -> normal operation(0x0) = 0x1F00  INTENSITY(0xXA) -> 21/32(0xA) = 0x1A0A  SCAN\_LIMIT(0xXB) -> only light up digit 0(0x0) = 0x1B00  SHUT\_DOWN(0xXC) -> normal operation(0x1) = 0x1C01 |

1. **MAX7219Send**We use r0 as higher 8 bits (D15-D8), and r1 as lower 8 bits (D7-D0).  
   Because we use some registers to represent DIN, CS, CLK, BSRR, BRR and the index, we push r0-r7 and LR to stack first and pop them up at last preventing registers from being overwritten.

|  |
| --- |
| **MAX7219Send:**  //input parameter: r0 is ADDRESS , r1 is DATA  //**TODO**: Use this function to send a message to max7219  push {r0, r1, r2, r3, r4, r5, r6, r7, LR}  lsl r0, 8 //move to D15-D8  add r0, r1 //r0 == din  ldr r1, =DIN  ldr r2, =CS  ldr r3, =CLK  ldr r4, =GPIOA\_BSRR //-> 1  ldr r5, =GPIOA\_BRR //-> 0  ldr r6, =0xF //now sending (r6)-th bit  **send\_loop:**  mov r7, 1  lsl r7, r6  str r3, [r5] //CLK -> 0  tst r0, r7 //same as ANDS but discard the result (just update condition flags)  beq bit\_not\_set //the sending bit(r0) != 1  str r1, [r4] //din -> 1  b if\_done  **bit\_not\_set:** //send clear bit  str r1, [r5] //din -> 0  **if\_done:**  str r3, [r4] //CLK -> 1  subs r6, 0x1  bge send\_loop  str r2, [r5] //CS -> 0  str r2, [r4] //CS -> 1  pop {r0, r1, r2, r3, r4, r5, r6, r7, PC}  BX LR |

1. **Delay**Same as LAB 4. Initial r0 as one\_sec and keep subtracting it until r0 = 0.

|  |
| --- |
| **Delay:**  //**TODO**: Write a delay 1sec function  beq delay\_end  subs r0, 0x4  b Delay  **delay\_end:**  bx lr |

1. **Display0toF**Load the array we’d like to display, and use r0 to save the array index. Because we only display digit 0, the r0 which saving higher 8 bits is 1. And the lower 8 bits load from array. Do the loop and each delay 1 sec to display from 0 to F.

|  |
| --- |
| **Display0toF:**  //**TODO**: Display 0 to F at first digit on 7-SEG LED Display one per second  mov r2, 0x0  ldr r3, =arr **display\_loop:**  mov r0, 0x1  ldrb r1, [r3,r2]  bl MAX7219Send   ldr r0, =one\_sec  bl Delay   add r2, 1  cmp r2, 0x10  bne display\_loop  b Display0toF |

1. **Demo Video**  
   <https://youtu.be/Nk4KFui3HAE>

## Lab5.2: Max7219與7-Seg LED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，例如學號為1234567則顯示下圖：



完成以下程式碼，將放在student\_id array 裡的學號顯示到7-seg LED上。

Note: 請使用decode mode

Using GPIO output to display your student ID on 7-Seg LED. Picture above is showing the case that your student ID is 1234567.

Complete the code giving below. Put your student ID in **student\_id array** and display it to 7-Seg LED.

Note: Please enable decode mode.

|  |
| --- |
| .syntax unified  .cpu cortex-m4  .thumb  .data  student\_id: .byte 1, 2, 3,4, 5, 6, 7 //TODO: put your student id here  .text  .global main  main:  BL GPIO\_init  BL max7219\_init  //TODO: display your student id on 7-Seg LED  Program\_end:  B Program\_end    GPIO\_init:  //TODO: Initialize three GPIO pins as output for max7219 DIN, CS and CLK  BX LR  MAX7219Send:  //input parameter: r0 is ADDRESS , r1 is DATA  //TODO: Use this function to send a message to max7219  BX LR  max7219\_init:  //TODO: Initial max7219 registers.  BX LR |

1. **GPIO\_init**Same as LAB 5.1.

|  |
| --- |
| RCC\_AHB2ENR -> enable GPIOA = 0x1  GPIOA\_MODER -> set PA5, 6, 7 as output mode  =(GPIOA\_MODER & 0xFFFF03FF) | 0b0101010000000000  GPIOA\_OSPEEDER -> set PA5, 6, 7 as high speed  =(GPIOA\_OSPEEDER & 0xFFFF03FF) | 0b1010100000000000 |

1. **MAX7219\_init**Set decode mode (whether decode or not), intensity (brightness of 7-segment), scan limit (how many digit to display), shut down, and display test.

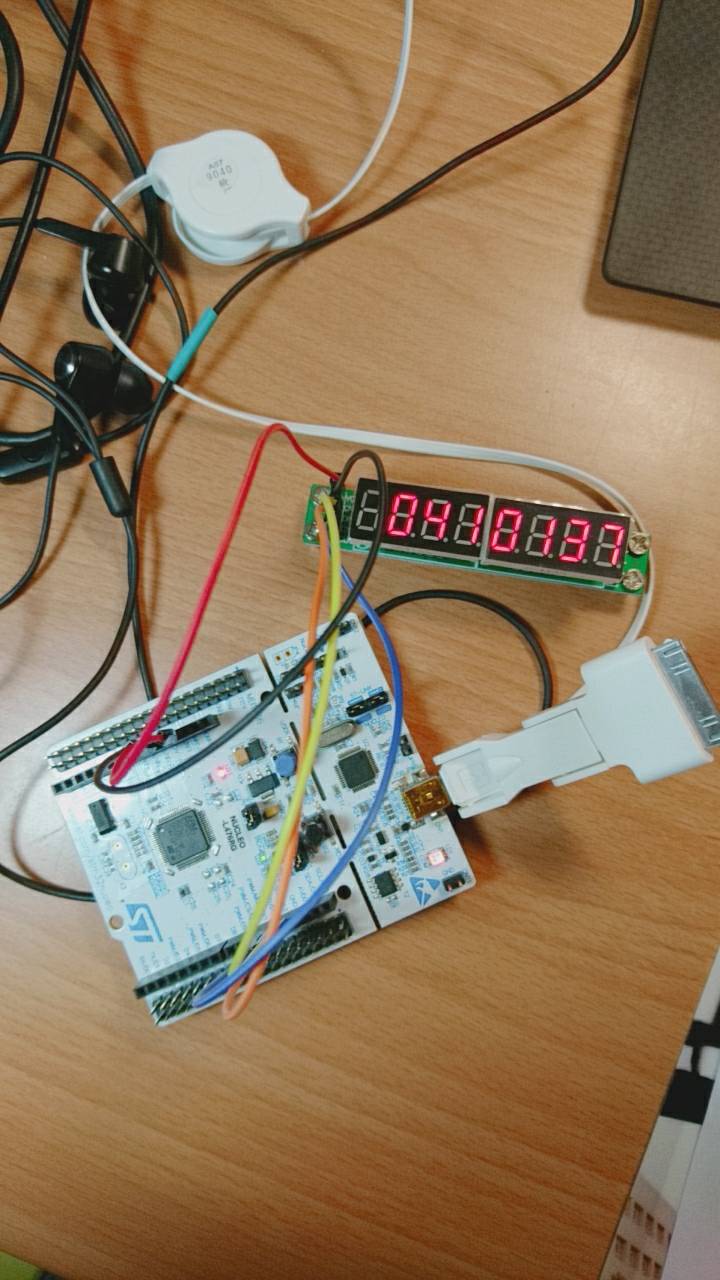
|  |
| --- |
| DECODE(0xX9) -> Code B decode for digit 0-7(0xFF) = 0x19FF  DISPLAY\_TEST(0xXF) -> normal operation(0x0) = 0x1F00  INTENSITY(0xXA) -> 21/32(0xA) = 0x1A0A  SCAN\_LIMIT(0xXB) -> only light up digit 0(0x0) = 0x1B00  SHUT\_DOWN(0xXC) -> normal operation(0x1) = 0x1C01 |

1. **MAX7219Send**Same as LAB 5.1.   
   We use r0 as higher 8 bits (D15-D8), and r1 as lower 8 bits (D7-D0).  
   Because we use some registers to represent DIN, CS, CLK, BSRR, BRR and the index, we push r0-r7 and LR to stack first and pop them up at last preventing registers from being overwritten.

|  |
| --- |
| **MAX7219Send:**  //input parameter: r0 is ADDRESS , r1 is DATA  //**TODO**: Use this function to send a message to max7219  push {r0, r1, r2, r3, r4, r5, r6, r7, LR}  lsl r0, 8 //move to D15-D8  add r0, r1 //r0 == din  ldr r1, =DIN  ldr r2, =CS  ldr r3, =CLK  ldr r4, =GPIOA\_BSRR //-> 1  ldr r5, =GPIOA\_BRR //-> 0  ldr r6, =0xF //now sending (r6)-th bit  **send\_loop:**  mov r7, 1  lsl r7, r6  str r3, [r5] //CLK -> 0  tst r0, r7 //same as ANDS but discard the result (just update condition flags)  beq bit\_not\_set //the sending bit(r0) != 1  str r1, [r4] //din -> 1  b if\_done  **bit\_not\_set:** //send clear bit  str r1, [r5] //din -> 0  **if\_done:**  str r3, [r4] //CLK -> 1  subs r6, 0x1  bge send\_loop  str r2, [r5] //CS -> 0  str r2, [r4] //CS -> 1  pop {r0, r1, r2, r3, r4, r5, r6, r7, PC}  BX LR |

1. **Display\_student\_id**We’ve used a byte array to save the student ID, so just load the array and display them on different digits.  
   R0 is the index of display digit.  
   R2 represents for student\_id array index.  
   Digit 7 is to display the first number in student ID array whose index is 0, so we add the index of student ID array while subtracting the index of display digit.  
   The frequency is 100MHz, which is quite high for our eyes, and there’s no delay between loops, so we can see these 7 digits at the same time.

|  |
| --- |
| **Display\_student\_ID:**  mov r0, 0x8 //init digit = 8  mov r2, 0x0 //arr index  ldr r3, =student\_id1 **display\_loop:**  subs r0, r0, 1 //digit -1  ldrb r1, [r3,r2] //student\_id1[r2]  bl MAX7219Send  adds r2, r2, 1 //arr index +1  cmp r0, 1 //digit == 1  bne display\_loop  b Display\_student\_ID |

1. **Demo**

## Lab5.3 Max7219與7-SEG LED練習—顯示Fibonacci數

請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。

fib(0) = 0、fib(1) = 1、fib(2) = 1 、…

若fib(N) 100000000則顯示-1。

範例影片如下：

<https://goo.gl/6DF6eY>

Note: 請記得處理User button開關彈跳的問題。

Design a program to detect user button on STM32 pressed. When user button is pressed N times, display fib(N) on 7-Seg LED. When user button is held down for 1 second, set displayed number to 0. Example video link is given above.

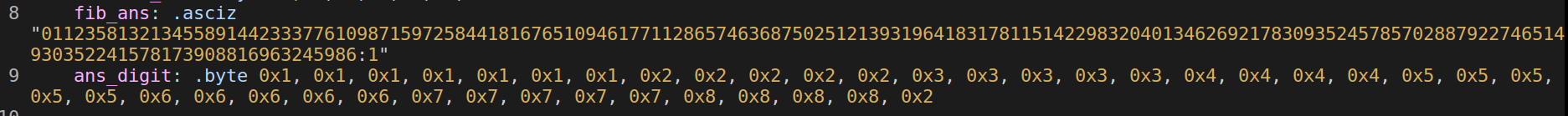
fib(0) = 0, fib(1) = 1, fib(2) = 1, ……

if fib(N) 100000000 then display -1.

Note: Please remember to deal with the bouncing problem.

1.Workflow and abstraction of this problem

step(1)

build the fibonacci table (The mod is not defined in ARM, which will be quite hard to acquire each digit of fibonacci number)

(2)

Use triple pointer, ptr1 to point the current position at fib\_ans for reading the data from table, ptr2 to point the current digit of the answer, which , in short is ans\_digit, and finally, the ptr3 that iterates again and again in each max7219 sequence.

(3)

From 2 , e.g. the fibonacci number is now 987, than the ptr1 will point to 9

9 8 7

ptr1 ar 9, ptr3

ptr2=3 symbolizes the digit of current fibonacci number, for instance, the digit now is 3

and ptr 3 will now at position of 9 as well.

(4)

Once the max7219 has successfully send the data to the display, the ptr3 will increase to next position, which, to be more detailed , will get the data of next position , say 8.

Than 8 will be sent to the max7219, the same is true for all the fibonacci number in this question.

(5)

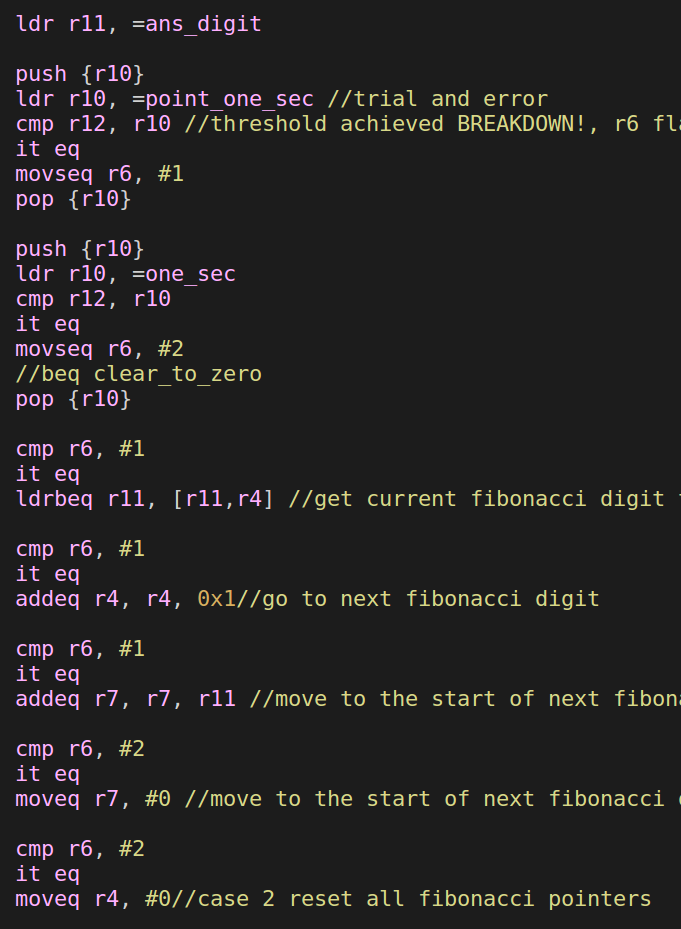
Suppose we press the use button, the ptr2 will move to the next one, to get the digit of next fibonacci number, what’s more, the position of the fobonacci pointer will be moved to the position: ptr1=ptr1+ptr2

e.g. 9 8 7 1 5 9 7

ptr3------→ptr3 (move the step of digit of current fibonacci number, likewise 3)

(6)

The final position , :1 which in ascii - ‘0’ will be turned to -1, where the fibonacci is now at #32th fibonacci number.



(7)

In the circumstance when user button is not pressed , step 2,3,4 will be iterated again and again.

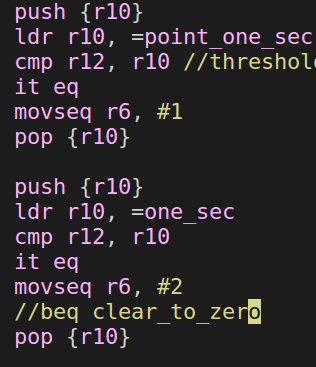
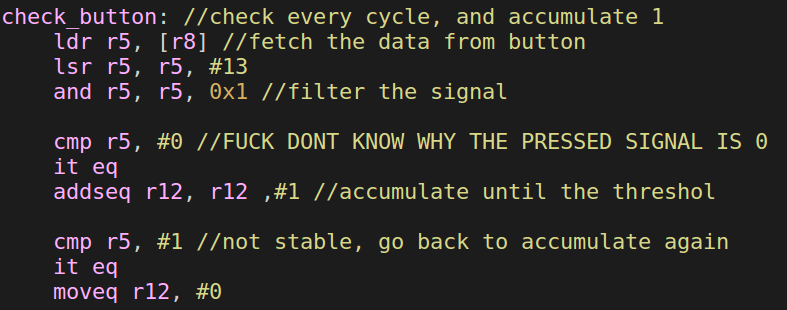
2.Button debounce logic will be described as follows.

(1) Once the signal is detected as 0, accumulate the counter.

(2) Once the signal reaches 1 again, reset the counter

(3) As long as the counter reach 1000, the button is confirmed as triggered, moreover, the button will be considered to be LONG PRESS as the counter reaches 10000(or some amount much higher than the normal detection of short press.

(4) In conclusion there will be two type of flag, one for short press and the other for long press.



3.Demo

<https://www.youtube.com/watch?v=_9CbzX6SYX0>

Acquired knowledge and thoughts

Compared to the last lab with only LED to light, this one is much more complicated ans sophisticated. We have to fully comprehend the structure and procedure of how to place the right number on the right position in the right moment, which is extremely tough along with the debounce.

Despite the tiredness, I really learned a lot of how microprocessor works.