Lab5 7-Seg LED

0516059 劉嘉豪
051632 朱蝶

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

1. Theory 實驗原理

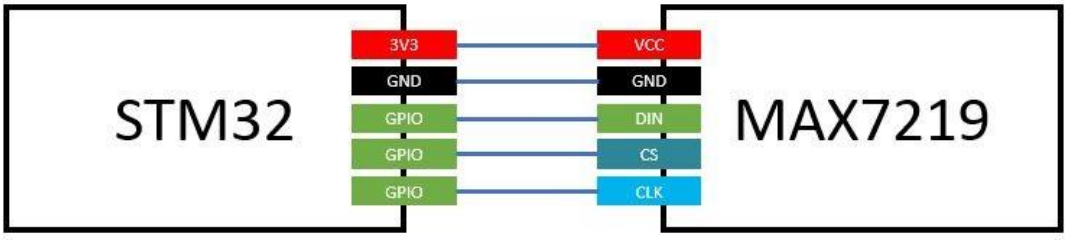
Please check the course material of lab5_note

請參考上課lab5_note講義。

2. Steps 實驗步驟

2.1. Lab5.1: Max7219與7-Seg LED練習(Practice of Max7219 and 7-Seg LED)—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
```

First: GPIO_INIT

如同之前的LAB一樣，先把要用的GPIO_INTI做好，這次使用的是PA5,PA6,PA7，因此把MODER的那三個PORT的部分改成01(output mode)，Speeder的部分則設成10。

```
GPIO_init:
    //enable GPIO port A
    ldr r0, =RCC_AHB2ENR
    mov r1, 0b1
    str r1, [r0]

    //enable GPIO PA7,6,5 for output mode=01
    ldr r0, =GPIOA_MODER
    ldr r1, =0xABFF57FF//0xFFFF 01 01 01 (765) 11 FF
    str r1, [r0]

    //GPIOA_OTYPER: push-pull (reset state)

    //default low speed, set to high speed=10
    ldr r0, =GPIOA_OSPEEDER
    ldr r1, =0x0000A800 //1010 10(765)00 00
    str r1, [r0]
```

Second: MAX7219_init:

在此處把MAX7219_init設定好，由於這題需要印出abc等字，因此把decode的關掉(後8個byte設成0)，Display_test也設定成0(關掉，不需要全亮)，Intensity設定成A(差不多亮就可以了)，SCAN_LIMIT設定成0，因為我們只需要一個位數(0代表一個位數)，Shut_down則關掉(0才是shut down mode)

```
  .equ    DECODE,         0x19 //decode control
  .equ    INTENSITY,      0x1A //brightness
  .equ    SCAN_LIMIT,     0x1B //how many digits to display
  .equ    SHUT_DOWN,      0x1C //shut down -- we did't use this
  .equ    DISPLAY_TEST,   0x1F //display test -- we did' use this

max7219_init:
    //TODO: Initialize max7219 registers
    push {r0, r1, LR}
    ldr r0, =DECODE
    ldr r1, =0x0 //NO DECODE
    bl MAX7219Send

    ldr r0, =DISPLAY_TEST
    ldr r1, =0x0 //normal operation
    bl MAX7219Send

    ldr r0, =INTENSITY
    ldr r1, =0xA // 21/32 (brightness)
    bl MAX7219Send

    ldr r0, =SCAN_LIMIT
    ldr r1, =0x0 //only light up digit 0
    bl MAX7219Send

    ldr r0, =SHUT_DOWN
    ldr r1, =0x1 //normal operation
    bl MAX7219Send
```

Third: MAX7219Send:

我們使用r0當作前8個bit，r1當作後8個bit，我們在這個部份把DIN,CS,CLK等值照著老師講義上的步驟設定好，詳細寫在下方的註解中

```
MAX7219Send:
    lsl r0, r0, #8 //把R0變成高8位
    add r0, r0, r1 //R0+R1變成DIN應該要有的值, 存在R0
    ldr r1, =DIN
    ldr r2, =CS
    ldr r3, =CLK
    ldr r4, =GPIOA_BSRR //-> 1
    ldr r5, =GPIOA_BRR //-> 0
    ldr r6, =0xF

send_loop:
    mov r7, #1
    lsl r7, r7, r6  // 用來測試 R0 的BYTE 從第16檢視到第一個
    str r3, [r5]  // 先把CLK設定成0
    tst r0, r7    // 看看R0的第R7個BYTE是否為一
    beq bit_not_set //如果是一, 則不進去
    str r1, [r4]    //把DIN設定成1
    b if_done

bit_not_set:
    str r1, [r5] //因為R0就是DIN的值, 因此R0的BYTE如果不為1, 則為0, 把DIN設定成0

if_done:
    str r3, [r4]    //CLK設定成1
    sub r6, r6, #1  //檢視R0的前一個位數
    cmp r6, 0       //是0的話就代表DIN的值都設定完了
    bge send_loop   //還沒設定玩就跳回去繼續設定
    str r2, [r5]    //若都設定玩則把CS設定成0
    str r2, [r4]    //再把CS設定成1
    bx lr
```

Fourth:Delay

Delay的時間須為1秒，這個delay loop有4個cycle(cmp為兩個 1+2+1)而一個cycle需要0.25us，4個就需要1us因此r9設定為100萬，才能使delay最接近1秒。

```
Delay:
    sub r9, r9, #1 //r9 為1000000
    cmp r9, #0
    bne Delay
    BX LR
```

Fifth:Display0toF

先把ARR設定好，因為我們使用的不是decodemode因此在arr當中的數字須根據7段顯示器分別的要求去設定，之後Load到r8，詳細的步驟在下方的註解當中

```
arr: .byte 0x7E, 0x30, 0x6D, 0x79, 0x33, 0x5B, 0x5F, 0x70, 0x7F, 0x7B,
0x77, 0x1F, 0x4E, 0x3D, 0x4F, 0x47
```

```
Display0toF:
    ldr r8, =arr // arr裡面放的東西為依序的0到F
    mov r10, #0  // R10用來進算現在該顯示ARR裡面的第幾個
    loop:
        mov r0, #1  // R0設為1表示 現在設定的是七段顯示器的第一個輸出
        ldrb r1, [r8,r10]    //R1則為R8(ARR的位址)+COUNTER(R10)的值
        bl MAX7219Send  //傳進去設定
        ldr r9, =one_sec
        bl Delay        //DELAY 1秒
        add r10, r10, #1 //R10+1 讓下次顯示的數字為ARR裡面的下一個
        cmp r10, #16    //若R10已經執行到第16次, 則把R10歸零, 讓他繼續跑下去
        beq Display0toF
        b loop
```

2.2. Lab5.2: Max7219與7-Seg LED練習(Practice of Max7219 and 7-Seg LED)—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
```

First: GPIO_INIT

同上一題

Second: MAX7219_init:

由於這題要求是要使用decode mode因此在init時把decode的後8位數設定成ff，而使用的位數則需要使用7個，因此SCAN_LIMIT的後8個byte設定成6(代表7個)，其他部分同上一題

```
ldr r0, =DECODE
ldr r1, =0xFF
bl MAX7219Send

ldr r0, =SCAN_LIMIT
ldr r1, =0x6
bl MAX7219Send
```

Third:MAX7219Send:

同上一題

Fourth: Display_student

先把arr設定成我的學號(0516059)，因為我們使用的是decode mode因此在arr當中直接輸入數字即可，且因為有7個數字，因 此需要用到7個顯示器，詳細的步驟在下

面註解當中

```
.data
    arr: .byte 0x0, 0x5, 0x1, 0x6, 0x0, 0x5, 0x9

Display0toF:
    ldr r8, =arr //把arr的位址load到r8
    mov r10, #0  //r10為計算現在在學號的第幾個數字
    mov r11, #7  //r11為要使用7段顯示器的第幾個數字
    loop:
        mov r0, r11  //把r11mov到r0 r0為等等傳進max7219send的高8個byte
        ldrb r1, [r8,r10] //r1則設定為arr的位址+counter(r10) 為後8個byte
        bl MAX7219Send
        add r10, r10, #1 //每執行完一次則把r10+1, 讓下次做的是下一個學號數字
        sub r11, r11, #1 //R11+1,下個要顯示的數字要在下一個7段顯示器內
        cmp r11, #0     //若r11=0則代表輸入完畢
        beq Display0toF
        b loop
```

3.1. Lab5.3 Max7219與7-SEG LED練習(Practice of Max7219 and 7-Seg LED)—顯示Fibonacci數(Show the Fibonacci number)

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.

First: GPIO_INIT

這題需要使用按紐，因此要用PC13這個PORT，同時把GPIOC_IDR load到r8

```
ldr r0, =GPIOC_MODER
ldr r1, [r0]
//clear pc13 to zero
and r1, r1, 0xf3ffffff
str r1, [r0]

ldr r8, =GPIOC_IDR
```

其餘同上一題

Second: MAX7219_init:

由於這題所需要的位數隨著按按鈕的次數不同而改變，因此scan_limit最初先設為0，
其餘同上一題

```
    ldr r0, =SCAN_LIMIT
    ldr r1, =0x0
    bl MAX7219Send
```

Third:MAX7219Send:

同上一題

Fourth: Display_fib:

在印出fib數列的時候我們分成兩個部分討論，先計算出每個位數分別有幾個數字(例:只有一個位數的有0 1 1 2 3 5 8 有7個，兩個位數的有13 21 34 55 89 有5個等等)，之後按照按過按鈕的次數來決定下一個顯示的數字應該使用幾個位數，算出來後存到r1，送到max7219send中(下圖左為部分程式碼，r12為按按鈕的計數器從0開始算)

得到需要用到幾個位數之後便進入下個loop，來看要印出哪個數字，由於在宣告arr時只能使用byte(word嘗試過了但似乎沒辦法)，因此大於256的數字沒辦法存在arr當中，所以在r12小於13時都可以從arr當中load出數字，但第14個數字之後則不能，於是我們將每個數字都列出來(如下圖右)，根據r12的數字來決定是哪個數字要顯示在顯示器上，取出來的數字我們放在r11裡(我們對於-1這個數字在後面有特殊的處理)

因此在這邊的處理我們得到r12為按幾次按鍵的計數器，r11為顯示器上應顯示的值

```
.data
    arr: .byte 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233
```

```
    cmp r12, #20
    it le
    ldrle r1, =0x3

    cmp r12, #16
    it le
    ldrle r1, =0x2

    cmp r12, #11
    it le
    ldrle r1, =0x1

    cmp r12, #6
    it le
    ldrle r1, =0x0

    cmp r12, #40
    it eq
    ldreq r1, =0x1

    ldr r0, =SCAN_LIMIT
    bl MAX7219Send
```

```
    cmp r12, #13
    bgt more_than
    ldr r11, =arr
    ldrb r11, [r11, r12]
    b check_button
    more_than:
        cmp r12, #14
        it eq
        ldreq r11, =377
        cmp r12, #15
        it eq
        ldreq r11, =610
        cmp r12, #16
        it eq
        ldreq r11, =987
        cmp r12, #17
        it eq
        ldreq r11, =1597
        cmp r12, #18
        it eq
        ldreq r11, =2584
        cmp r12, #19
        it eq
```

Fifth:check_button

在存完r12及r11之後 每次都進入check_button來確認按鈕有沒有被按，流程在註解中

```
check_button:
    ldr r5, [r8]        //r8為GPIOC_IDR load到r5
    lsr r5, r5, #13
    and r5, r5, 0x1     //看看GPIOC_IDR是不是一(有沒有被按按鈕)

    cmp r5, #0          //R5是0的話則R10+1 (R10為去彈跳的計數器,
                        超過設定的值就算是按下按鈕了)
    it eq
    addseq r10, r10, #1

    cmp r5, #1          //R5若是1則R10歸零(R5=0代表沒按或是還在不穩定階段)
    it eq
    moveq r10, #0

    ldr r9, =one_sec    //R9=one_sec
    cmp r10, r9         //若r10=one_sec則代表按鈕已被按下
    it eq
    movseq r6, #1       //r6設成1

    ldr r9, =long       //r9=long
    cmp r10, r9         //若r10=long則代表按鈕被長按
    it eq
    movseq r6, #2       //r6設成2
    //beq clear_to_zero

    cmp r6, #1          //若按鈕被按下則r12+1(r12為按按鈕的計數器)
    it eq               //之後回去display_fib重新檢視下個數字需要多少位數
    addseq r12, r12, #1
    beq Display_fib

    cmp r6, #2          //若按鈕為長按則把r12設成0(因此會輸出0)
    it eq
    movseq r12, #0

    mov r6, #0          //若都沒被按或是還在不穩定狀態則進入check_end
    b check_end
```

Sixth: check_end

```
    check_end:
    mov r2, r1      //此處的r1是顯示器需要用幾個數字
    adds r2, r2, #1 //由於r1是0-7因此r2要+1變成1-8
    mov r0, #1      //r0=1
    b light

light:
    cmp r12, #40    //若r12=40則跳到minus_one的loop
    beq minus_one
    mov r3, #10     //r3=10
    cmp r0, r2      //若r0跟r2相同代表所有位數都顯示完畢
    bgt Display_fib
    udiv r5, r11, r3    //r5=r11/10 (r11為應該顯示的數字) r5為r11除以10的商數
    mul r4, r5, r3      //r4=r5*10
    subs r1, r11, r4    //r1=r11-r4 (r1為r11除以10的餘數, 也就是要印出的數字)
    udiv r11, r11, r3   //把r11除以10, 作為下次近來運算的r11 (例 123->12)
    bl MAX7219Send      //把r0(第幾個位數)及r1(輸出的值)傳入讓顯示器運作
    adds r0, r0, #1     //r0+1 (下次進來時顯示的會是下一個顯示器)(從右至左)
    b light
minus_one:
    mov r0, #2          //若近到minus_one代表已經按了40次, 輸出的值應為-1
    mov r1, 0xA         //但我們是用mod的方式來處理數字
    bl MAX7219Send      //因此沒辦法印出負號, 所以我們把-1特別提出來寫
    mov r0, #1
    mov r1, 0x1
    bl MAX7219Send
    b Display_fib
```

Conclusion

第一次用七段顯示器覺得很好玩，其實一二題都不難，但是第三題debug了很久，一部分是在思考怎麼樣才能夠輸出一個費波納數，最後決定建一個array一個一個傳而不是每次都要計算。另一部份是在熟悉每個mode轉換以及哪個先做、哪個後做，要跳回哪裡，總結就是這次lab比之前都更加複雜，需要更多時間。