



Lab4 7-Seg LED 實驗四 7-Seg LED

Group7 0616231 彭世丞

4-1. Print Number and Words (without code B decode mode)

1. 實驗目的

這個實驗想要先讓我們基礎的了解怎麼在實作控制在7-Seg LED。
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)

2. 實驗過程

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。



我選擇 PA5,6,7 分別對應到DIN, CS, CLK

又因為本題不能使用code B decode mode，我先用了一個陣列儲存0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F 以 D0~D7 表示所得的值。

Digit	BCD				7-Segments							abcdefg	gfedcba	Display	Reference
	D	C	B	A	a	b	c	d	e	f	g				
0	0	0	0	0	1	1	1	1	1	1	0	0x7E	0x3F	0	
1	0	0	0	1	0	1	1	0	0	0	0	0x30	0x06	1	
2	0	0	1	0	1	1	0	1	1	0	1	0x6D	0x5B	2	
3	0	0	1	1	1	1	1	1	0	0	1	0x79	0x4F	3	
4	0	1	0	0	0	1	1	0	0	1	1	0x33	0x66	4	
5	0	1	0	1	1	0	1	1	0	1	1	0x5B	0x6D	5	
6	0	1	1	0	1	0	1	1	1	1	1	0x5F	0x7D	6	
7	0	1	1	1	1	1	1	0	0	0	0	0x70	0x07	7	
8	1	0	0	0	1	1	1	1	1	1	1	0x7F	0x7F	8	
9	1	0	0	1	1	1	1	1	0	1	1	0x7B	0x6F	9	
A	X	X	X	X	1	1	1	0	1	1	1	0x77	0x77	A	
b	X	X	X	X	0	0	1	1	1	1	1	0x1F	0x7C	b	
C	X	X	X	X	1	0	0	1	1	1	0	0x4E	0x39	C	
d	X	X	X	X	0	1	1	1	1	0	1	0x3D	0x5E	d	
E	X	X	X	X	1	0	0	1	1	1	1	0x4F	0x79	E	
F	X	X	X	X	1	0	0	0	1	1	1	0x47	0x71	F	



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.

```
.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, 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
```

以下是我的 code 區段剖析：

➤ .data

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

➤ .text

除了之前作業設定的那些，我多加了以下東西

```
.equ    GPIOA_BSRR,      0X48000018    //SET THAT BIT TO 1
.equ    GPIOA_BRR,       0X48000028    //SET THAT BIT TO 0
.equ    DIN,             0b100000     //PA5
.equ    CS,              0b1000000    //PA6
.equ    CLK,             0b10000000   //PA7
```



.equ	DECODE,	0x9
.equ	INTENSITY,	0xA
.equ	SCAN_LIMIT,	0xB
.equ	SHUT_DOWN,	0xC
.equ	DISPLAY_TEST,	0xF

➤ max7219_init

DECODE	+	No decode for digits 7-0	= 0x900
INTENSITY	+	MAX7219(21/32)	= 0xA00
SCAN_LIMIT	+	Display digit 0 only	= 0xB00
SHOUDOWN	+	Normal Operation	= 0xC01
DISPLAY_TEST	+	Normal Operation	= 0xF00

➤ MAX7219Send

分為兩部分，(1)初始化registers，(2)傳16bits的loop

整個MAX7219Send我用push和pop包起來，所有MAX7219Send有用到的registers我都先存到stack裡頭，MAX7219Send結束後再全部pop出來。如此便能避免改變到原本registers存的值。

想法如下：

```
confirm din's 12 bits data and put it in x
default all value
for ( int i = 16; i > 0; i -- ){
    set CLK to 0
    check if current bit in x is 0 or 1
    if ( x == 0 )          set DIN to 1
    else if ( x == 1 )     set DIN to 0

    set CLK to 1
}
set CS 0 then 1
```

過程中，遇到的問題：

- 本來很煩惱不知道要怎麼儲存每一次傳出去的DIN值，結果後來發現根本不用存，真是好家在。
- 如何在寫每個function的時候考慮到不更動原來的register值是一件很苦惱的事情，後來想到可以在進function的時候push進所有會用到的register，離開function的時候再pop出那些register，便能夠解決不小心改變register值的問題了



4-2. Print Student ID in 7-Seg LED (with code B decode mode)

1. 實驗目的

這個實驗想要實現多個digits的展示。
如下圖所示



2. 實驗過程

Complete the code giving below. Put my student ID in `student_id` array and display it to 7-Seg LED..

```
.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.
```



過程中，遇到的問題：

- 這一題基本上前一題做出來的話就不會有甚麼太大的困難。
- 記得如果要填Digit 1的話HEX_CODE是0x02，不是0x01。然後要小心數字的順序，一不小心的話可能會反著印。

4-3. Show the Fibonacci number (with code B decode mode)

1. 實驗目的

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。

【Note: 請記得處理 User button 開關彈跳的問題。】

2. 實驗過程

基本想法：我用兩個陣列，一個陣列去存fib的所有值、一個去存現在這個fib number 的位數。如下，

```
fib_array: .asciz "01123581321345589144233377610987159.....  
num_digit: .byte 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x2, .....
```

以下是我的 code 各區段剖析(主要增加的區段)：

➤ Main:

大致上就是先load出現在這個fib number的位數，然後設定好SCAN_LIMIT
接著就是跑迴圈，照著該fib number由低位到高位send 到 LED
迴圈中每次都會跳到check_bottom去判斷有沒有按

```
BL    GPIO_init  
BL    max7219_init  
mov r4, 0x0 //***** # of current number *****  
mov r5, 0x0 //***** current number's "Start Position" in fib_array *****  
start_from_find_digit:  
ldr r2, =fib_array  
ldr r3, =num_digit  
ldrb r11, [r3, r4] //r11 now = the digit of current number  
subs r1, r11, 1  
ldr r0, =SCAN_LIMIT  
bl MAX7219Send    //set digit finish  
mov r0, 0x0  
important_loop:  
adds r0, r0, #1    //r0 = position in LED
```



```
subs r11, r11, #1
adds r11, r11, r5
ldrb r1, [r2, r11]
subs r11, r11, r5 //maintain r11
sub r1, r1, #48 //r1 = char of current number digit
bl check_bottom
bl MAX7219Send
cmp r11, 0
bne important_loop
b start_from_find_digit
```

➤ Check_bottom:

想法如下：

```
Read value from GPIOC_IDR, put value in x
Check whether x is 0 or 1
If ( x == 0 )      y+=1
Else if ( x == 1 )  y=0
If ( y == 100 )      move to next fib number
Else if ( y == 10000 ) restart all pointers
Else if ( now is fib(N) and N greater than 39 )    N = 40, set fib(N) to -1
```

過程中，遇到的問題：

- 這一題最難的應該就是register實在是太多了，要小心不要弄混。
- 如何設定按一下和長按的時間也是一門藝術阿！按一下的時間設太大的要按很久、而長按如果設太短的話很容易會跟按一下的重疊，造成無可預期的結果。或許在長按過後設Delay是個好選擇(?)
- 如何能在組合語言中活用if,then,else能讓你的code省下很多行，是值得去熟練的好工具(it eq.....)。

實驗心得與結語

這次lab4的三個小題讓我學會了怎麼靈活的對7-Seg LED下達指令和傳送資料，對於組合語言的整體寫法也更有sense了。(像在function的前後push pop registers...等等)

我現在也有自己一套方式去選用register和寫function了，這次的lab相較於之前感覺比較沒有手足無措的地方，只要對7-Seg LED燈的傳輸觀念沒有錯誤，基本上不難寫出來。