**微算機實驗報告格式**

**姓姓名：陳達軒**

**學號：0610837**

**上課時間：**

**2018 Fall 2EF**



Lab # 5

**ㄧ、實驗目的：**

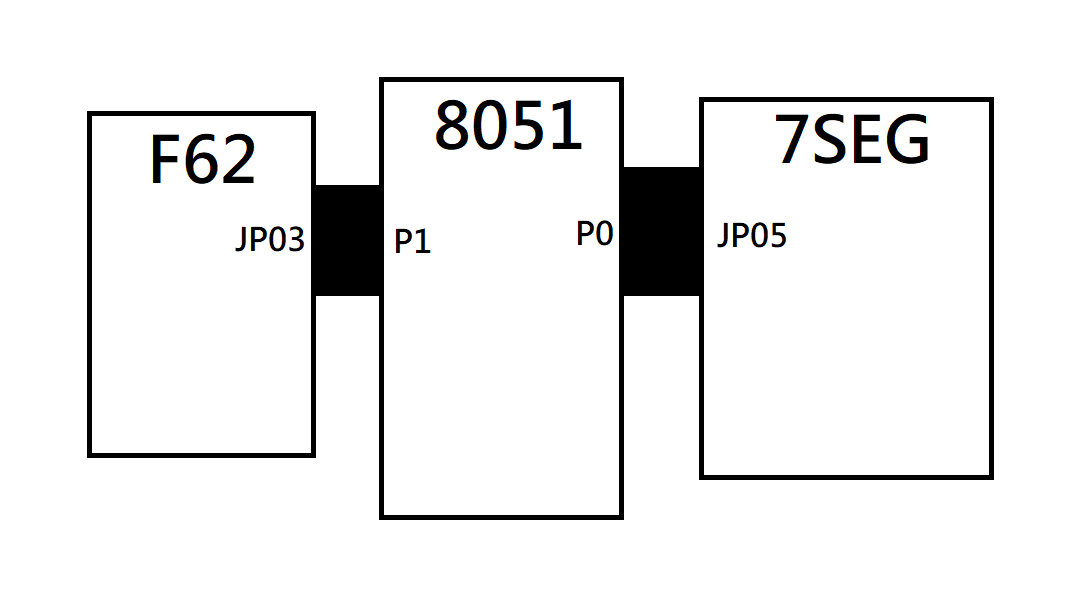
但請以自己對本次實驗的認知來加以說明。

學習鍵盤掃描過程，學習更多七段顯示器的應用，如何技巧性使用delay，使得程式能夠在理想狀態下執行。

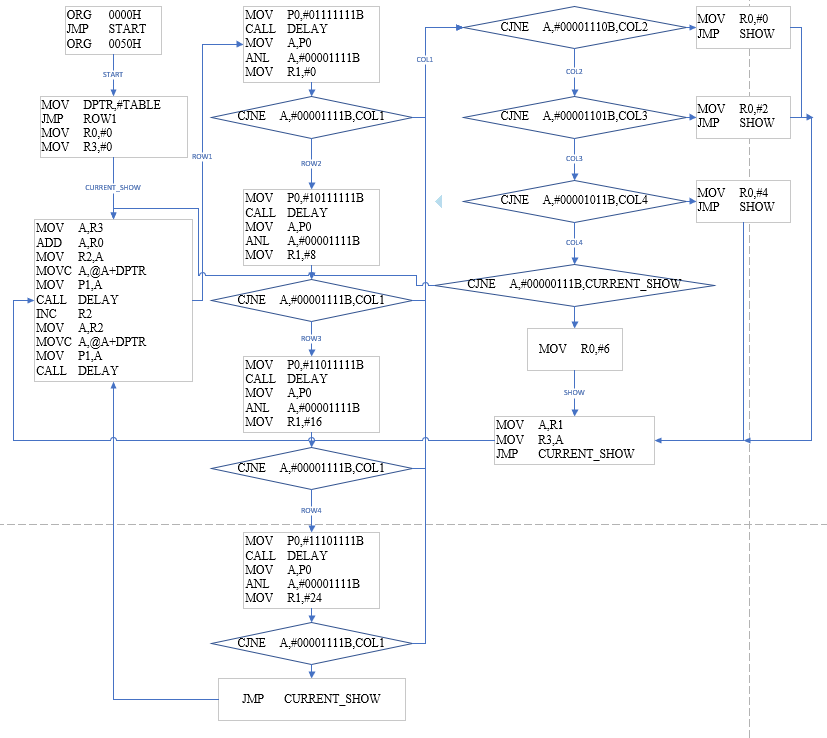
**二、硬體架構：**

請儘量自己畫圖不管是手繪或者是以電腦繪圖(Word or PoworPoint)。

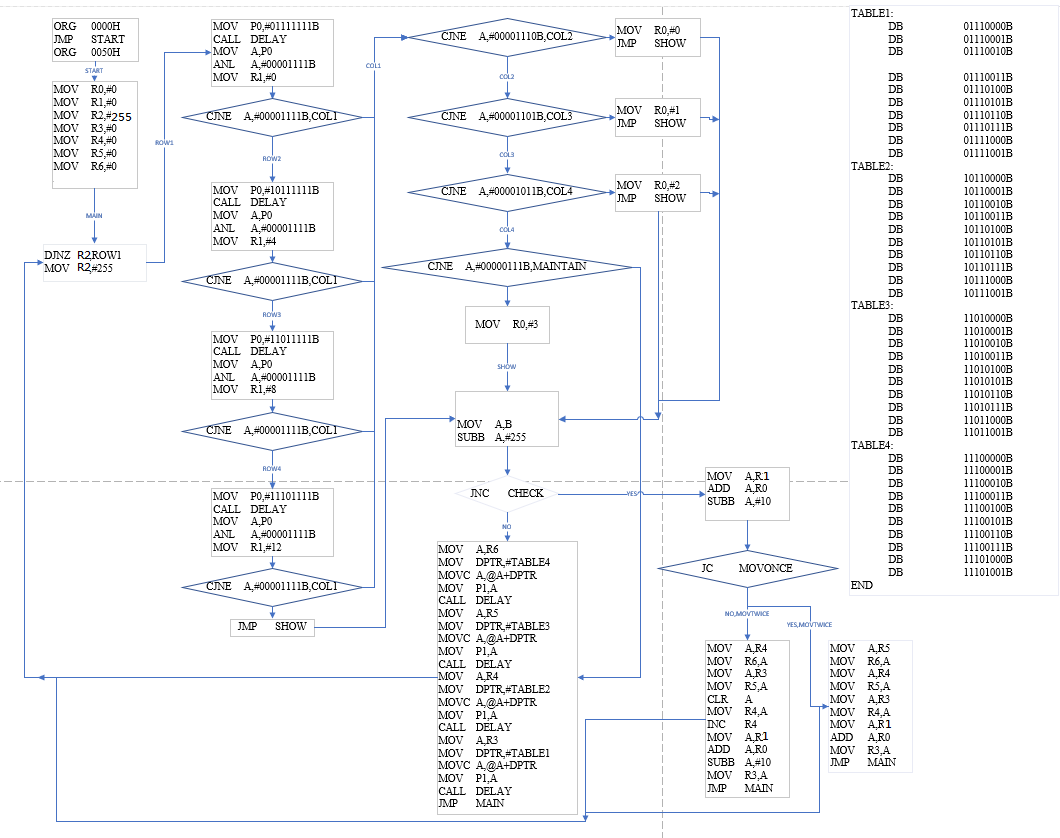
四個七段顯示器、F62鍵盤4\*4、8051 ICE Adapter



**三、程式流程圖：**

基礎題：

進階題：

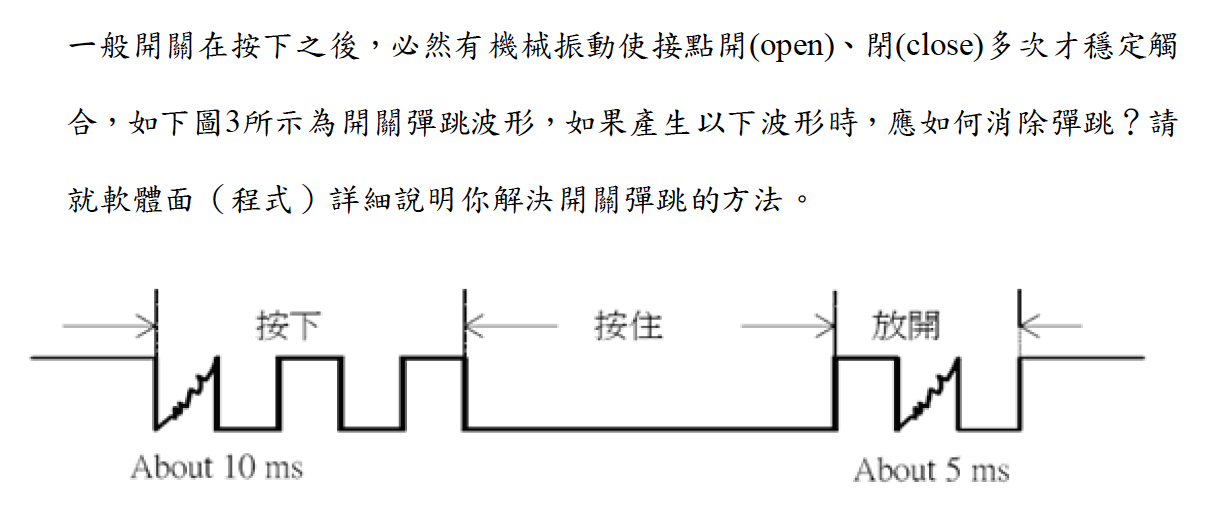


**四、問題與討論：**

Note:

1.回答助教在實驗講義中所提的問題。

2.自己發現的疑問以及如何解決的方法。

****

使用一個delay跟在讀取之後，不讓程式快速的重複讀取，讓delay的時間超過10ms，使得在彈跳波型持續的時候，CPU是處於delay而不運算的狀態，進而使得按鈕能穩定觸發。這次的實驗，做到一半曾出現十位數消失，個位數還在的狀況，根據延遲長短，當延遲長十位數會消失，但是當延遲較短時十位數又會出現，可能是因為程式讓4顆七段顯示器停留在個位數的部分時進行的運算量比較多，為讓四顆七段顯示器都同時有相同的亮度，必須把程式設計成每一個位數停留時間相同。做的時候還發現彈跳波型似乎也某些時候也存在於沒按下的狀況，如果不考慮讓程式自己排除一部分一瞬間亂跳的狀況，似乎會讓程式運作出現問題導致結果不同。

**五、心得：**

Note:

1.對於實驗內容的心得感想

2.對本課程的建議。

雖然這幾周重複不斷的使用掃描，但是經過這次的進階題才學會delay其實不是那麼方便的東西，在不斷的累積下也可能讓CPU慢下來，delay雖然重要，但是必須要有技巧性的使用，除了延遲顯示以外，還可以讓讀取時間更人性化，而鍵盤掃描也讓我體會到了CPU如果不考慮Interrupt將會處於非常浪費運算的狀態，必須時時刻刻進行掃描。

**六、程式碼與註解：**

基礎題：

ORG 0000H

JMP START

ORG 0050H

START:

MOV DPTR,#TABLE

JMP ROW1

MOV R0,#0

MOV R3,#0

CURRENT\_SHOW:

MOV A,R3 //R3為row

ADD A,R0 //R0為col

MOV R2,A //放到R2

//顯示十位數

MOVC A,@A+DPTR

MOV P1,A

CALL DELAY

//顯示個位數

INC R2

MOV A,R2

MOVC A,@A+DPTR

MOV P1,A

CALL DELAY

ROW1:

MOV P0,#01111111B

CALL DELAY

MOV A,P0

ANL A,#00001111B

MOV R1,#0 //一個row需要8個位置儲存 8\*0

CJNE A,#00001111B,COL1 // A不等於00001111B，表示有col按下

ROW2:

MOV P0,#10111111B

CALL DELAY

MOV A,P0

ANL A,#00001111B

MOV R1,#8 //第二個row 8\*1

CJNE A,#00001111B,COL1

ROW3:

MOV P0,#11011111B

CALL DELAY

MOV A,P0

ANL A,#00001111B

MOV R1,#16 //第二個row 8\*2

CJNE A,#00001111B,COL1

ROW4:

MOV P0,#11101111B

CALL DELAY

MOV A,P0

ANL A,#00001111B

MOV R1,#24 // 第三個row 8\*3

CJNE A,#00001111B,COL1

JMP CURRENT\_SHOW //沒有數字被按下

COL1:

CJNE A,#00001110B,COL2

MOV R0,#0 //0\*2

JMP SHOW

COL2:

CJNE A,#00001101B,COL3

MOV R0,#2 //1\*2

JMP SHOW

COL3:

CJNE A,#00001011B,COL4

MOV R0,#4 //2\*2

JMP SHOW

COL4:

CJNE A,#00000111B,CURRENT\_SHOW //沒有數字被按下 不更新R1到R3也不更新R0

MOV R0,#6 //3\*2

SHOW:

MOV A,R1

MOV R3,A //更新R1到R3

JMP CURRENT\_SHOW

;--------------------------

DELAY:

MOV R5,#255

DELAY1:

DJNZ R5,DELAY1

RET

;--------------------------

TABLE:

DB 11110000B,01110000B //十位數不顯示+個位數=0

DB 11110000B,01110001B //十位數不顯示+個位數=1

DB 11110000B,01110010B //十位數不顯示+個位數=2

DB 11110000B,01110011B //十位數不顯示+個位數=3

DB 11110000B,01110100B //...

DB 11110000B,01110101B

DB 11110000B,01110110B

DB 11110000B,01110111B

DB 11110000B,01111000B

DB 11110000B,01111001B

DB 10110001B,01110000B //十位數 =1+個位數=0

DB 10110001B,01110001B //...

DB 10110001B,01110010B

DB 10110001B,01110011B

DB 10110001B,01110100B

DB 10110001B,01110101B

END

進階題：

ORG 0000H

JMP START

ORG 0050H

START:

MOV R0,#0 //歸零

MOV R1,#0

MOV R2,#255 //將B當作counter設為255

MOV R3,#0 //儲存第一格

MOV R4,#0 //儲存第二格

MOV R5,#0 //儲存第三格

MOV R6,#0 //儲存第四格

MAIN:

DJNZ R2,ROW1 //開始計算讀取次數

MOV R2,#255 //將B重設回255

ROW1:

MOV P0,#01111111B //看第一row

CALL DELAY //Delay消除彈跳

MOV A,P0

ANL A,#00001111B //看有無被按下

MOV R1,#0 //一行差4 4\*0

CJNE A,#00001111B,COL1 //如果A不等於00001111B，表示有col按下

ROW2:

MOV P0,#10111111B

CALL DELAY

MOV A,P0

ANL A,#00001111B

MOV R1,#4 //一行差4 4\*1

CJNE A,#00001111B,COL1

ROW3:

MOV P0,#11011111B

CALL DELAY

MOV A,P0

ANL A,#00001111B

MOV R1,#8 //一行差4 4\*2

CJNE A,#00001111B,COL1

ROW4:

MOV P0,#11101111B

CALL DELAY

MOV A,P0

ANL A,#00001111B

MOV R1,#12 //一行差4 4\*3

CJNE A,#00001111B,COL1

JMP MAINTAIN

COL1:

CJNE A,#00001110B,COL2

MOV R0,#0 //0

JMP SHOW

COL2:

CJNE A,#00001101B,COL3

MOV R0,#1 //1

JMP SHOW

COL3:

CJNE A,#00001011B,COL4

MOV R0,#2 //2

JMP SHOW

COL4:

CJNE A,#00000111B,MAINTAIN //沒有數字被按下，不更新

MOV R0,#3 //3

SHOW:

MOV A,R2 //查看R2是不是到255

SUBB A,#255

JNC CHECK //相減沒借位表示有255跳到更新數字

MAINTAIN: //維持原本的顯示

MOV A,R6

MOV DPTR,#TABLE4

MOVC A,@A+DPTR

MOV P1,A

CALL DELAY

MOV A,R5

MOV DPTR,#TABLE3

MOVC A,@A+DPTR

MOV P1,A

CALL DELAY

MOV A,R4

MOV DPTR,#TABLE2

MOVC A,@A+DPTR

MOV P1,A

CALL DELAY

MOV A,R3

MOV DPTR,#TABLE1

MOVC A,@A+DPTR

MOV P1,A

CALL DELAY

JMP MAIN

CHECK: //查看新的數字是不是十位數

MOV A,R1

ADD A,R0

SUBB A,#10

JC MOVONCE

MOVTWICE: //十位數移動兩次

MOV A,R4

MOV R6,A

MOV A,R3

MOV R5,A

CLR A

MOV R4,A

INC R4

MOV A,R1

ADD A,R0

SUBB A,#10

MOV R3,A

JMP MAIN

MOVONCE: //個位數移動一次

MOV A,R5

MOV R6,A

MOV A,R4

MOV R5,A

MOV A,R3

MOV R4,A

MOV A,R1

ADD A,R0

MOV R3,A

JMP MAIN

;--------------------------

DELAY:

MOV R7,#255

DELAY1:

DJNZ R7,DELAY1

RET

;--------------------------

TABLE1: //第一格的表格

DB 01110000B

DB 01110001B

DB 01110010B

DB 01110011B

DB 01110100B

DB 01110101B

DB 01110110B

DB 01110111B

DB 01111000B

DB 01111001B

TABLE2: //第二格的表格

DB 10110000B

DB 10110001B

DB 10110010B

DB 10110011B

DB 10110100B

DB 10110101B

DB 10110110B

DB 10110111B

DB 10111000B

DB 10111001B

TABLE3: //第三格的表格

DB 11010000B

DB 11010001B

DB 11010010B

DB 11010011B

DB 11010100B

DB 11010101B

DB 11010110B

DB 11010111B

DB 11011000B

DB 11011001B

TABLE4: //第四格的表格

DB 11100000B

DB 11100001B

DB 11100010B

DB 11100011B

DB 11100100B

DB 11100101B

DB 11100110B

DB 11100111B

DB 11101000B

DB 11101001B

END