微算機實驗報告

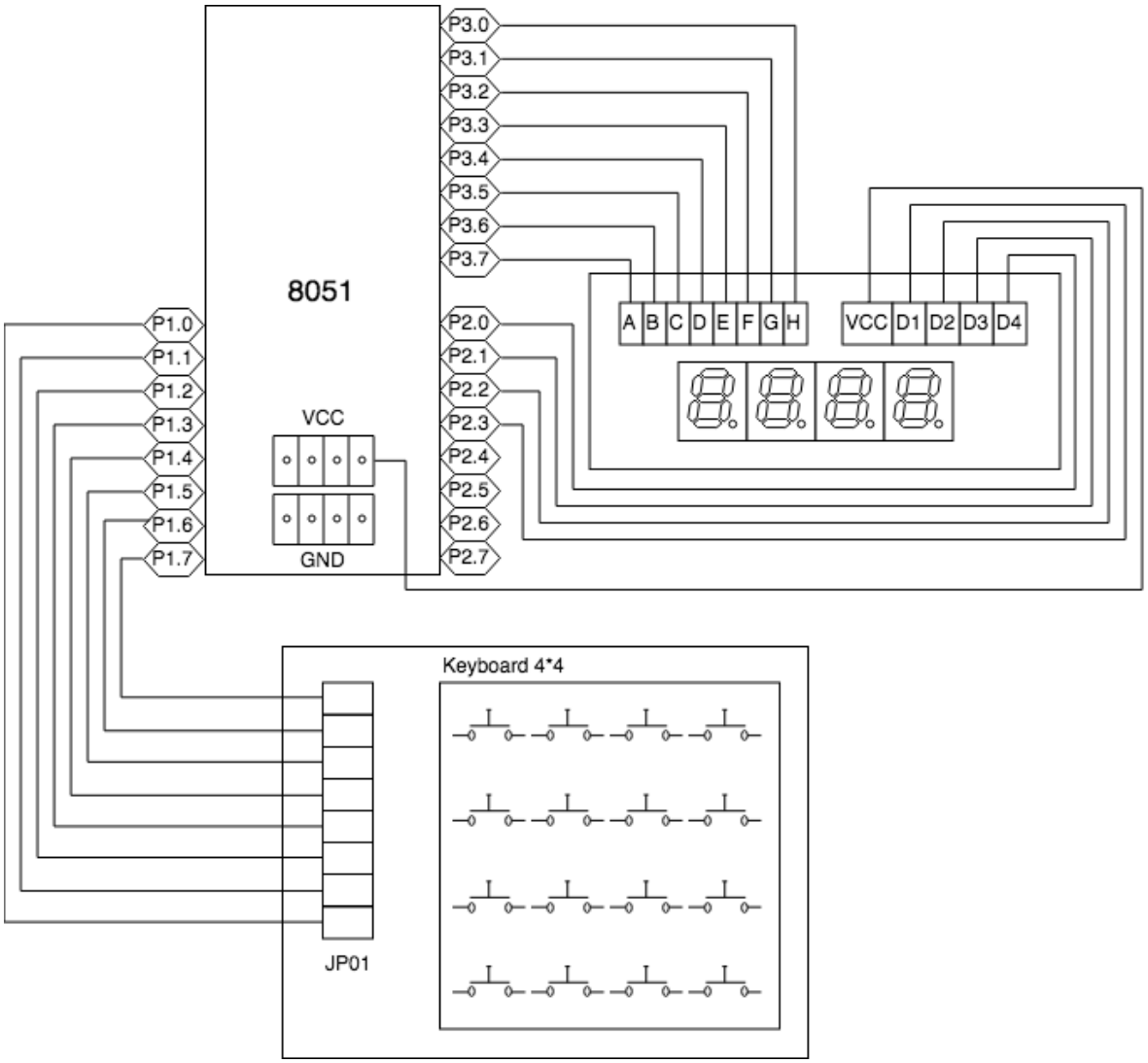
Lab #7

姓名：楊哲睿
系級：電機 10
學號：0610780
上課時間：4EF、4IJ

一、實驗目的：

- 瞭解鍵盤掃描並結合應用其他實驗板。

二、硬體架構：



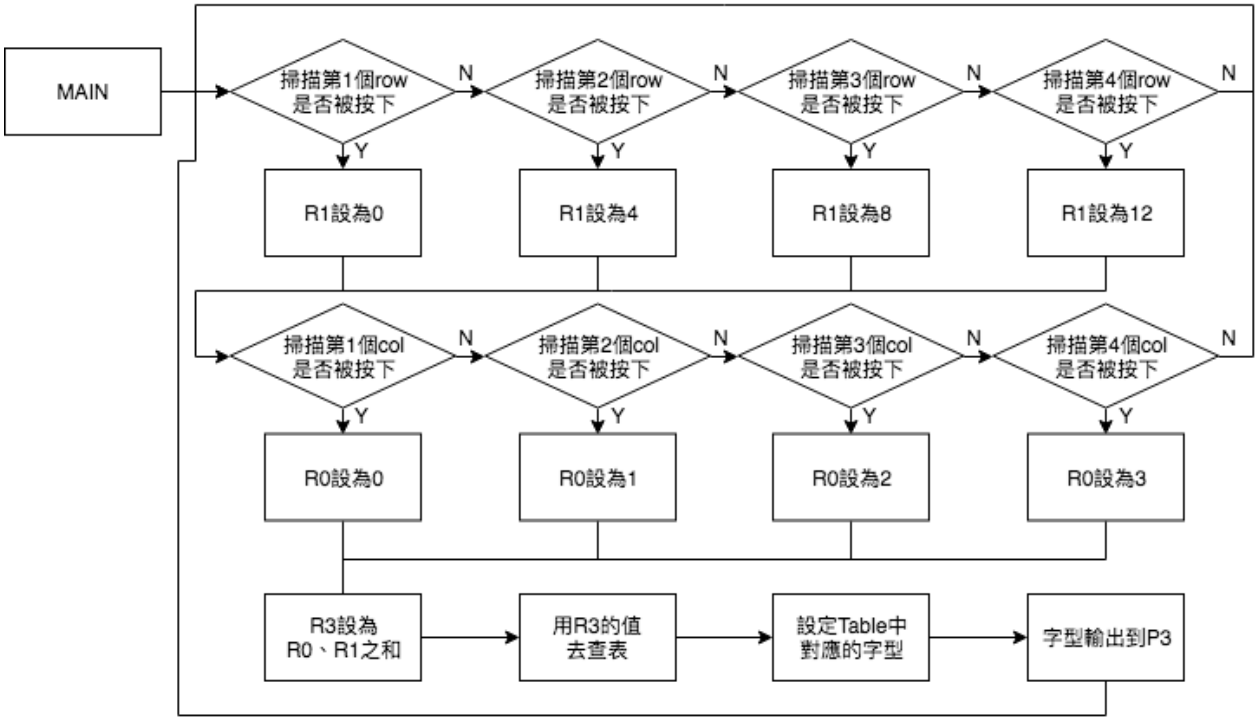
三、程式流程圖：

- 基本題

鍵盤由左至右，上至下分別設為下列數值：

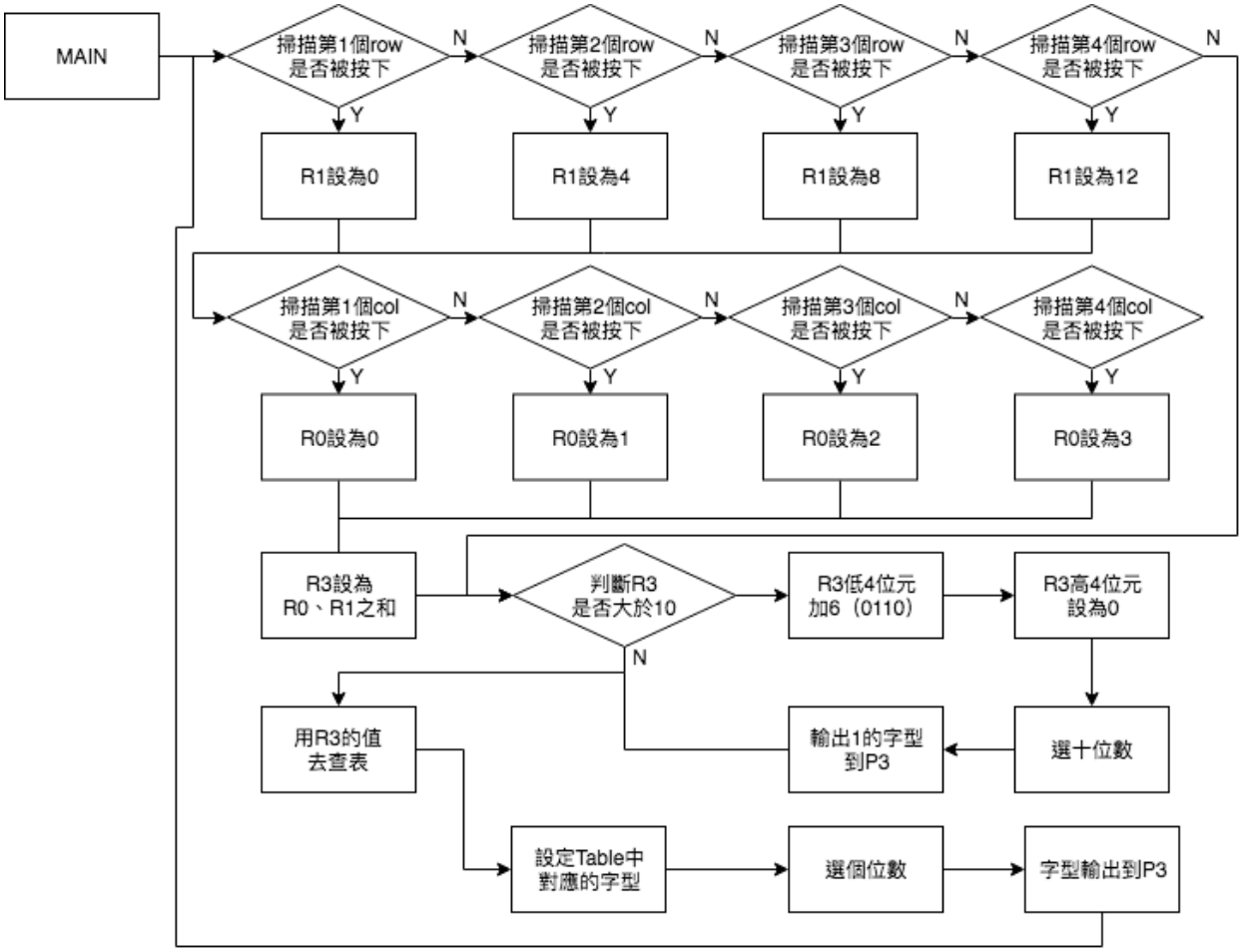
0	1	2	3
4	5	6	7
8	9	x	x
x	x	x	x

當鍵盤被按下去時，將其數值顯示在四顆七節顯示器的最右邊。



- 進階題

  除了 0~9 數字顯示外，能延伸至 10~15 的數字，即每個鍵盤按下去都有對應的數字。



四、問題與討論：

- 一般開關在按下之後，必然有機械振動使接點開(open)、閉(close)多次才穩定觸合，如下圖 3 所示為開關彈跳波形，如果產生以下波形時，應如何消除彈跳？請就軟體面（程式)詳細說明你解決開關彈跳的方法。

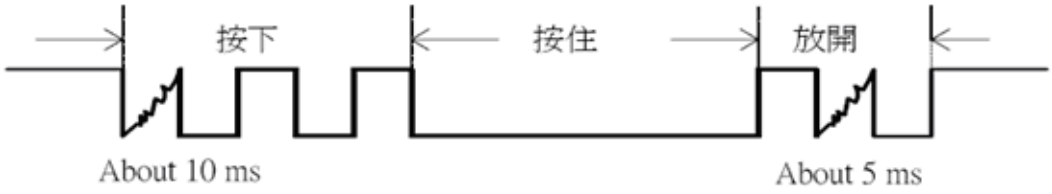


圖 3　開關彈跳波形

就軟體面解決的話我有想到兩個方法：

第一個就是在課本上的示範程式中所使用的，當要掃描按鍵是否有被按下時，先延遲一段時間（ACALL DELAY），等到彈跳消失後，再來讀取鍵盤輸入，如此便能避免掉彈跳時不穩定的情形。

另一個方法是可以設計一個計數器，當輸入改變時，便將計數器加一，到一定數值時才開始判定是按下開關或是放開開關，其實也是類似 DELAY 的效果，執行完動作再將計數器歸零，這樣也能改善彈跳的不穩定所帶來的影響。

五、程式碼與註解：

- 基本題

鍵盤由左至右，上至下分別設為下列數值：

0	1	2	3
4	5	6	7
8	9	x	x
x	x	x	x

當鍵盤被按下去時，將其數值顯示在四顆七節顯示器的最右邊。

```
    ORG 00H
    AJMP MAIN
    ORG 50H
MAIN:
    MOV DPTR, #NUMTABLE
ROW1:
    MOV P1, #7FH          ;掃描第一個 row(s01~s04)
```

```
    CALL DELAY
    MOV A, P1
    ANL A, #0FH           ;將較高的 4-bit 清為 0
    MOV R1, #0
    CJNE A, #0FH, COL1    ;確認 s01~s04 是否有被按下
                          ;有則至掃描 col 的地方
ROW2:                     ;沒有則往下掃第二個 row(s05~s08)
    MOV P1, #0BFH         ;掃描第二個 row(s05~s08)
    CALL DELAY
    MOV A, P1
    ANL A, #0FH           ;將較高的 4-bit 清為 0
    MOV R1, #4
    CJNE A, #0FH, COL1    ;確認 s05~s08 是否有被按下
                          ;有則至掃描 col 的地方
ROW3:                     ;沒有則往下掃第三個 row(s09~s12)
    MOV P1, #0DFH         ;掃描第三個 row(s09~s12)
    CALL DELAY
    MOV A, P1
    ANL A, #0FH           ;將較高的 4-bit 清為 0
    MOV R1, #8
    CJNE A, #0FH, COL1    ;確認 s09~s12 是否有被按下
                          ;有則至掃描 col 的地方
ROW4:                     ;沒有則往下掃第四個 row(s13~s16)
    MOV P1, #0EFH         ;掃描第四個 row(s13~s16)
    CALL DELAY
    MOV A, P1
    ANL A, #0FH           ;將較高的 4-bit 清為 0
    MOV R1, #12
    CJNE A, #0FH, COL1    ;確認 s13~s16 是否有被按下
                          ;有則至掃描 col 的地方
    JMP ROW1              ;沒有則從頭掃描
COL1:
    CJNE A, #0EH, COL2    ;0EH=00001110B
    MOV R0, #0
    JMP SHOW
COL2:
    CJNE A, #0DH, COL3    ;0DH=00001101B
    MOV R0, #1
```

```
    JMP SHOW
COL3:
    CJNE A, #0BH, COL4    ;0BH=00001011B
    MOV R0, #2
    JMP SHOW
COL4:
    CJNE A, #07H, ROW1    ;07H=00000111B
    MOV R0, #3
SHOW:
    MOV A, R1
    ADD A, R0
    MOVC A, @A+DPTR
OVER:
    MOV P2, #0DH
    MOV P3, A
    JMP ROW1

DELAY:
    MOV R5, #100
DELAY1:
    MOV R6, #150
DELAY2:
    DJNZ R6, DELAY2
    DJNZ R5, DELAY1
    RET

NUMTABLE:
    DB 0C0H ;0
    DB 0F9H ;1
    DB 0A4H ;2
    DB 0B0H ;3
    DB 099H ;4
    DB 092H ;5
    DB 082H ;6
    DB 0F8H ;7
    DB 080H ;8
    DB 090H ;9
    DB 088H ;A
```

```
    DB 083H ;B
    DB 0C6H ;C
    DB 0A1H ;D
    DB 086H ;E
    DB 08EH ;F

END
```

- 進階題

除了 0~9 數字顯示外，能延伸至 10~15 的數字，即每個鍵盤按下去都有對應的數字

```
    ORG 00H
    AJMP MAIN
    ORG 50H
MAIN:
    MOV DPTR, #NUMTABLE
ROW1:
    MOV P1, #7FH          ;掃描第一個 row(s01~s04)
    CALL DELAY
    MOV A, P1
    ANL A, #0FH           ;將較高的 4-bit 清為 0
    MOV R1, #0
    CJNE A, #0FH, COL1    ;確認 s01~s04 是否有被按下
                          ;有則至掃描 col 的地方
ROW2:                     ;沒有則往下掃第二個 row(s05~s08)
    MOV P1, #0BFH         ;掃描第二個 row(s05~s08)
    CALL DELAY
    MOV A, P1
    ANL A, #0FH           ;將較高的 4-bit 清為 0
    MOV R1, #4
    CJNE A, #0FH, COL1    ;確認 s05~s08 是否有被按下
                          ;有則至掃描 col 的地方
ROW3:                     ;沒有則往下掃第三個 row(s09~s12)
    MOV P1, #0DFH         ;掃描第三個 row(s09~s12)
    CALL DELAY
    MOV A, P1
    ANL A, #0FH           ;將較高的 4-bit 清為 0
```

```
    MOV R1, #8
    CJNE A, #0FH, COL1     ;確認 s09~s12 是否有被按下
                           ;有則至掃描 col 的地方
ROW4:                      ;沒有則往下掃第四個 row(s13~s16)
    MOV P1, #0EFH          ;掃描第四個 row(s13~s16)
    CALL DELAY
    MOV A, P1
    ANL A, #0FH                ;將較高的 4-bit 清為 0
    MOV R1, #12
    CJNE A, #0FH, COL1         ;確認 s13~s16 是否有被按下
                               ;有則至掃描 col 的地方
COL1:
    CJNE A, #0EH, COL2         ;0EH=00001110B
    MOV R0, #0
    JMP SHOW
COL2:
    CJNE A, #0DH, COL3         ;0DH=00001101B
    MOV R0, #1
    JMP SHOW
COL3:
    CJNE A, #0BH, COL4         ;0BH=00001011B
    MOV R0, #2
    JMP SHOW
COL4:
    CJNE A, #07H, NOCHANGE     ;07H=00000111B
    MOV R0, #3
SHOW:
    MOV A, R1
    ADD A, R0
    MOV R3, A
NOCHANGE:
    MOV A, R3
    SETB C
    CJNE R3, #10, OVER1        ;R3>10 時，C 被設為 0
OVER1:                         ;顯示 0~9（一位數）
    JNC OVER2
    MOVC A, @A+DPTR
    MOV P2, #0DH
```

```
    MOV P3, A
    CALL DELAY
    JMP ROW1
OVER2:                          ;顯示 10~15（兩位數）
    ADD A, #6
    ANL A, #0FH                 ;取出個位數
    MOVC A, @A+DPTR
    MOV P2, #0DH                ;個位
    MOV P3, A
    CALL DELAY
    MOV P3, #0FFH               ;避免殘影
    MOV P2, #0BH                ;十位
    MOV P3, #0F9H               ;顯示十位數 1
    JMP ROW1

DELAY:
    MOV R5, #100
DELAY1:
    MOV R6, #150
DELAY2:
    DJNZ R6, DELAY2
    DJNZ R5, DELAY1
    RET

NUMTABLE:
    DB 0C0H ;0
    DB 0F9H ;1
    DB 0A4H ;2
    DB 0B0H ;3
    DB 099H ;4
    DB 092H ;5
    DB 082H ;6
    DB 0F8H ;7
    DB 080H ;8
    DB 090H ;9

END
```

六、心得：

這次的實驗，原本在判斷被按下的按鍵代表的數字是一位數還是兩位數卡住了好長一段時間，後來回去翻翻課本，想到了老師上課時曾提到過 CJNE 指令會改變 C 旗標這件事情，搭配 JZ 或 JNZ 使用，發現這個拿來用在判斷某個變數大於或小於某個數值時真的十分管用，程式也簡化許多。

另外一個困擾我許久的事情是當我按下按鍵後，如果沒有持續按著就無法顯示數字，後來想到是要多設計一個情況：當沒有按鍵被按下時，我們要保留前一次被按下的數字，如此一來便不會在沒有按著按鍵的情況下無法正確的顯示數字了。