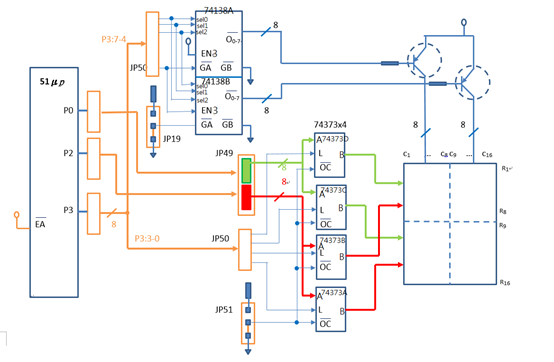
Lab4 Report

組員: 0616318蔡承恩 0616098黃秉茂 0616041 陳俊惟 0616329 李京叡

實驗目的:

學習程式與電路板的關係，利用組合語言控制16\*16 dot-matrix

設計圖:



74138 function table:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | EN3 | sel2 | sel1 | sel0 |  |  |  |  |  |  |  |  |
| L | L | H | 0 | 0 | 0 | L | H | H | H | H | H | H | H |
| 0 | 0 | 1 | H | L | H | H | H | H | H | H |
| 0 | 1 | 0 | H | H | L | H | H | H | H | H |
| 0 | 1 | 1 | H | H | H | L | H | H | H | H |
| 1 | 0 | 0 | H | H | H | H | L | H | H | H |
| 1 | 0 | 1 | H | H | H | H | H | L | H | H |
| 1 | 1 | 0 | H | H | H | H | H | H | L | H |
| 1 | 1 | 1 | H | H | H | H | H | H | H | L |
| otherwise | | | X | | | H | | | | | | | |

74138為一個decoder，實驗中使用2個74138: 74138A、74138B，分別控制左右班部分Dot-matrix Led欄位的開/關，P3[7]接到74138A的，和74138B的EN3，所以P3[7]可以控制dot-matrix led左右半部的開/關，當P3[7]=0時，74138A輸出為sel decode之後的值，74138B輸出全為0；當P3[7]=1時，74138A輸出全為0，74138B輸出為sel decode之後的值，因此P3[7]可以用來選擇dot-matrix的左右半部開關。P3[6:4]分別接到兩個74138的sel2, sel1, sel0，控制decoder輸出的數字。

74373 function table:

|  |  |  |
| --- | --- | --- |
|  | L | B |
| L | 0 | Varying as A |
| 1 | Status of A at L to H transition of L |
| H | X | Hi-Z |

74373是一個latch，實驗中使用到了4個74373，由下至上功用依序為:

74373A : 控制dot-matrix led下半部紅色的開/關

74373B : 控制dot-matrix led下半部綠色的開/關

74373C : 控制dot-matrix led上半部紅色的開/關

74373D : 控制dot-matrix led上半部綠色的開/關

P3[3], P3[2], P3[1], P3[0]分別為74373A~D的輸入L，P0接到74373D、74373C的輸入A，因此P0可以控制dot-matrix led綠色各列的開/關；P2接到74373B、74373A的輸入A，因此P2可以控制dot-matrix led紅色各列的開/關。當一個led所在的欄的值為1且所在的列的值為0時，該led即為開，否則為關。

設立環境:

依照LAB1的只是建立環境:

點選Option for Target: "Target-test"

在Options for Target 'Target-test'中操作如下:

Device中Database選擇為Megawin Device Database, MPC82G516

Target中勾選"Use On-chip ROM(0x0-0xFFFF)”和"Use On-chip XRAM(0x0-0x3FF)

Output 中勾選(Enabled) Debug information

C51中的code Operation的level 選擇 "0: Constant folding"

Debug選擇右側Use，選擇"Megawin On-Chip-Debug Driver”

點選setting 將cache options的3個選項全勾

CODE:

org 0

mov SP, #50H

mov P3, #0

;call delay

mov P0, #0FFH

mov P2, #0FFH

mov P3, #5H

mov P3, #0

start:

mov R6, #250

green\_2:

mov P3, #0

mov P0, #0H

mov A, #1H

mov P3, A

call delay2 ; col1 done

anl P3, #0F0H ; ==XXX==

mov P0, #7aH

add A, #10H

mov P3, A ; A:= ???

call delay2 ; col2 done

mov R7, #4

g2\_loop:

anl P3, #0F0H ; ==XXX==

mov P0, #4AH

add A, #10H

mov P3, A ; A:= ???

call delay2 ; col3-6 done in sequence

djnz R7, g2\_loop

anl P3, #0F0H ; ==XXX==

mov P0, #4EH

add A, #10H

mov P3, A

call delay2 ; col7 done

anl P3, #0F0H ; ==XXX==

mov P0, #0H

add A, #10H ; A:= ???

mov P3, A

call delay2 ; col8 done

djnz R6, green\_2

anl P3, #0F0H ; ==AA==

mov P0, #0FFH ; ==AA==

mov P3, A ; ==AA==

call delay2

redd:

mov R6, #250

red\_2:

mov P3, #0

mov P2, #0H

mov A, #4H

mov P3, A

call delay2 ; col1 done

anl P3, #0F0H ; ==XXX==

mov P2, #7AH

add A, #10H ; A:= ???

mov P3, A

call delay2 ; col2 done

mov R7, #4

r2\_loop:

anl P3, #0F0H ; ==XXX==

mov P2, #4AH

add A, #10H ; A:= ???

mov P3, A

call delay2 ; col3-6 done

djnz R7, r2\_loop

anl P3, #0F0H ; ==XXX==

mov P2, #4EH

add A, #10H ; A:= ???

mov P3, A

call delay2 ; col7 done

anl P3, #0F0H ; ==XXX==

mov P2, #0H

add A, #10H ; A:= ???

mov P3, A

call delay2 ; col8 done

djnz R6, red\_2

anl P3, #0F0H ; ==BB==

mov P2, #0FFH ; ==BB==

mov P3, A ; ==BB==

call delay2

mov R6, #250

yellow\_2:

mov P0, #0H

mov P2, #0H

mov A, #5H

mov P3, A

call delay2 ; col1 done

anl P3, #0F0H ; ==XXX==

mov P0, #7AH

mov P2, #7AH

add A, #10H ; A:= ???

mov P3, A

call delay2 ; col2 done

mov R7, #4

y2\_loop:

anl P3, #0F0H ; ==XXX==

mov P0, #4AH

mov P2, #4AH

add A, #10H ; A:= ???

mov P3, A

call delay2 ; col3-6 done

djnz R7, y2\_loop

anl P3, #0F0H ; ==XXX==

mov P0, #4EH

mov P2, #4EH

add A, #10H ; A:= ???

mov P3, A

call delay2 ; col7 done

anl P3, #0F0H ; ==XXX==

mov P0, #0H

mov P2, #0H

add A, #10H ; A:= ???

mov P3, A

call delay2 ; col8 done

djnz R6, yellow\_2

anl P3, #0F0H ; ==CC==

mov P0, #0FFH ; ==CC==

mov P2, #0FFH ; ==CC==

mov P3, A ; ==CC==

call delay2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame0:

mov P0, #0FEH

mov P2, #0H

add A, #00H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame0g:

add A, #10H

mov P3, A

call delay

djnz r7, frame0g

djnz r6, frame0

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame1:

mov P0, #0FDH

mov P2, #0H

add A, #10H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame1g:

add A, #10H

mov P3, A

call delay

djnz r7, frame1g

djnz r6, frame1

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame2:

mov P0, #0FBH

mov P2, #0H

add A, #20H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame2g:

add A, #10H

mov P3, A

call delay

djnz r7, frame2g

djnz r6, frame2

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame3:

mov P0, #0F7H

mov P2, #0H

add A, #30H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame3g:

add A, #10H

mov P3, A

call delay

djnz r7, frame3g

djnz r6, frame3

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame4:

mov P0, #0EFH

mov P2, #0H

add A, #40H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame4g:

add A, #10H

mov P3, A

call delay

djnz r7, frame4g

djnz r6, frame4

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame5:

mov P0, #0DFH

mov P2, #0H

add A, #50H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame5g:

add A, #10H

mov P3, A

call delay

djnz r7, frame5g

djnz r6, frame5

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame6:

mov P0, #0BFH

mov P2, #0H

add A, #60H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame6g:

add A, #10H

mov P3, A

call delay

djnz r7, frame6g

djnz r6, frame6

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame7:

mov P0, #07FH

mov P2, #0H

add A, #70H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame7g:

add A, #10H

mov P3, A

call delay

djnz r7, frame7g

djnz r6, frame7

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame8:

mov P0, #0BFH

mov P2, #0H

add A, #60H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame8g:

add A, #10H

mov P3, A

call delay

djnz r7, frame8g

djnz r6, frame8

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame9:

mov P0, #0DFH

mov P2, #0H

add A, #50H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame9g:

add A, #10H

mov P3, A

call delay

djnz r7, frame9g

djnz r6, frame9

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame10:

mov P0, #0EFH

mov P2, #0H

add A, #40H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame10g:

add A, #10H

mov P3, A

call delay

djnz r7, frame10g

djnz r6, frame10

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame11:

mov P0, #0F7H

mov P2, #0H

add A, #30H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame11g:

add A, #10H

mov P3, A

call delay

djnz r7, frame11g

djnz r6, frame11

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame12:

mov P0, #0FBH

mov P2, #0H

add A, #20H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame12g:

add A, #10H

mov P3, A

call delay

djnz r7, frame12g

djnz r6, frame12

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

;;;;;;;;;;;;;;;;;;;;;;;

mov A, #5

mov R6, #8

mov R7, #8

frame13:

mov P0, #0FDH

mov P2, #0H

add A, #10H

mov P3, A

call delay1

mov A, #5

mov P2, #0FFH;

frame13g:

add A, #10H

mov P3, A

call delay

djnz r7, frame13g

djnz r6, frame13

mov A, #5

mov R7, #8

;;;;;;;;;;;;;;;;;;;;;;;

call delay

jmp start

delay: push 2

push 3

mov R2, #2

dd: mov R3, #64

djnz R3, $

djnz R2, dd

pop 3

pop 2

ret

delay1: push 2

push 3

mov R2, #20

dd12: mov R3, #250

djnz R3, $

djnz R2, dd12

pop 3

pop 2

ret

delay2: push 2

push 3

mov R2, #2

dd3: mov R3, #250

djnz R3, $

djnz R2, dd3

pop 3

pop 2

ret

end

CODE解釋:

初始: SP指向50H做為stack的起始位置

R2、3 : 作為delay計時的暫存器

R6 : 延長green、red、yellow出現”2”的時間，以及frame[i]的時間

R7 : 1.一個frame中綠色需經過8個column，將R7初始設為8，讓綠色依照col向右移動(時間過 快，眼睛上感覺會是一次閃八個column)

CODE分2大部分:

1. 顯示數字2:

綠色:

將P2設為#0FFH控制紅色不亮，依據數字的形狀，控制P0，顯示綠色為底黑色數字的形狀。

P3初始為#1H，只讓綠色上半部顯示。將P0決定好參數後，之後每次P3加#10H，換 到下一行做處理，一共做8行。產生綠色為底的數字2

紅色:

P0設為#0FFH控制綠色不亮，依據數字的形狀，控制P2，顯示紅色為底黑色數字的形狀。

P3初始為#4H，只讓紅色上半部顯示。將P2決定好參數後，之後每次P3加#10H，換到下一行做處理，一共做8行。產生紅色為底的數字2。

黃色:

控制P0、P2，顯示黃色為底黑色數字的形狀。

P3初始為#5H，讓紅色綠色同時出現在上半部，顯示成為黃色。控制P0及P2，決定好參數後，P3加#10H，換到下一行處理，產生黃色為底的數字2。

2.顯示phase4

共有13個phase，每一個phase除了要產生1行的紅色外，也要產生一列的綠色，在交叉點因為紅色綠色同時顯示產生黃色。

依照phase的排列，對應給他不同的col顯示，在一個的phase中只會有一行紅色的顯示。在其他的col時，將P2設為#0FFH，迫使他不會亮。

而每一次的phase裡，我們都需要先設定好綠色該顯示的位置，再從col1開始向右顯示，使P3相加#10H，直到到達column8。

而P3[3:0]為#5H，控制綠色紅色在上排都可以亮。

\* difficulties encountered and resolving measures:

最一開始直接把lab2.3的code複製貼上，以為這樣就可以完成前3個frame(綠2 -> 紅2 -> 黃2 )了，結果我們都輸出值卻是都在dot-matrix的最右邊那一排，完全沒有出現任何2，大部分的組別也出現跟我們相同的狀況，發現顯示怪怪的，去檢查一下code認為應該總要有個東西對才對，所以懷疑是接線的問題，但卻不是硬體上的問題。後來經過老師提點，才發現環境的設定跟之前的不一樣，所以我們選擇重新建立一個新project，先把環境參數的設定先弄好了才開始把講義的code複製貼上。結果終於如講義上所言出現循環的3個frame(綠2 -> 紅2 -> 黃2 )。

但除了講義上說的左上角出現3個顏色的2外，我們的dot-matrix的左下角竟然是滿版的紅色，我們便先懷疑是接線的問題，重插線和看講義上硬體的接線圖後，確認我們的線並沒有接錯並再次檢查code是否有問題，之後把code刪刪改改後把code恢復原狀，然後莫名其妙就好了，所以懷疑是原本硬體上的預設值、初始值有誤或是code有小瑕疵。

接著在phase中，我們一致認為在同一個frame中，應該要先輸出紅線，再來是輸出綠線，然後就換下一個frame。輸出的順序的確也如我們的預期，不過時間差、時間間隔卻大有問題，紅線太快而綠線跑得太慢，以至於綠線看到的不是線而是綠點從最左邊慢慢地往右移，最寫依據不同的情況寫了新的delay才完成老師要求的phase4(紅線與綠線移動的部分)。

\* phenomena observed and explanations:

Code上發現:

按照lab2.3給的code可以執行，而結果會是出現綠底的2 -> 紅底的2 -> 黃底的2 之後我們要靠著自己謝phase4(紅色線從最左邊往最右邊移動再回到最左邊，以及綠色同時從最上排移到最下排在移回最上排)，透過code，我們可以知道R0 控制綠色column的哪幾格要亮、R2控制紅色column的哪幾格要亮、R3控制區塊和第幾個column。

其實整個綠底的2並不是同時閃，而是先用R0設定綠色column的哪幾格要亮，顯示玩綠底的2的第1個column後，透過P3移到下一個column，再用P0決定綠色column的哪幾格要亮，用P3來向右移，而P0是主要輸出，以此類推，便能顯示綠底的2。之所以能同時顯示是因為instruction執行的速度很快，因為顯示的時間差的極短所以我們會認為是同時顯示。而紅底和黃底也是以此類推，只是紅底不是改P0而是P2，而黃底改的不只是P0還有P2。

至於phase4，我們則是設定一個一個frame，每個frame再去設定他的P0

、P2和P3已達到紅色線從最左邊往最右邊移動再回到最左邊，以及綠色同時從最上排移到最下排在移回最上排

重設參數問題:

===AAA===：green\_label的最後一步，也是設定進入red\_label的參數(P0和P3)。如果少了 ===AAA===的code，紅底的2的frame會變成黃底和中間是綠色的2，也就是紅底的2和全版綠色的結果，因為參數是錯的

===BBB===：red\_label的最後一步，也是設定進入yellow\_label的參數(P2和P3)。就算少了 ===BBB===的code，也不會造成影響，因為參數是對的

===CCC===：yellow\_label的最後一步，也是設定回到green\_label的參數(P0、P2和P3)。如果少了 ===CCC===的code，綠底的2的frame會變成黃底和中間是洪色的2，也就是綠底的2和全版紅色的結果，因為參數是錯的

===XXX===：保留P3的前4bit和把P3的後4bit歸0。就算少了 ===XXX===的code，也不會造成影響，因為參數是對的

改delay就能影響程式執行的差異(如整體速度，間隔時間，閃爍程度，時間暫留等)

在code中，我們在16\*16的dot-matrix僅用8\*8的空間，若是要右半邊能夠閃爍(亮)，需調整P3[7]的數字，若是

self-evaluation

\* things learnt

學會最基本的流程，了解如何用code去控制dot-matrix，還有要怎麼利用delay和視覺暫留以達到同時顯示綠色的row，而是無法整個row同時亮的，一定是一個一個輛配合時間暫留才看起來會是一整排一起亮，不論是哪一區塊的8X8甚至是8X16、16X8皆是如此，因為一次只能亮一排column。所以要亮一排row的話，必定是靠著很快地亮起不同column的同個row的位置，依靠視覺暫留才能看起來是整個row同時亮

因為JP49只是控制上下區塊的顏色配置(綠、紅、黃、黑)以及該coloumn要如何顯示，而JP50是決定亮哪個區塊的第幾個column，所以其實可以靠著接線將JP49和JP50連到switch的區塊，就可以單靠硬體決定dot-matrix的哪行該顯示或亮了

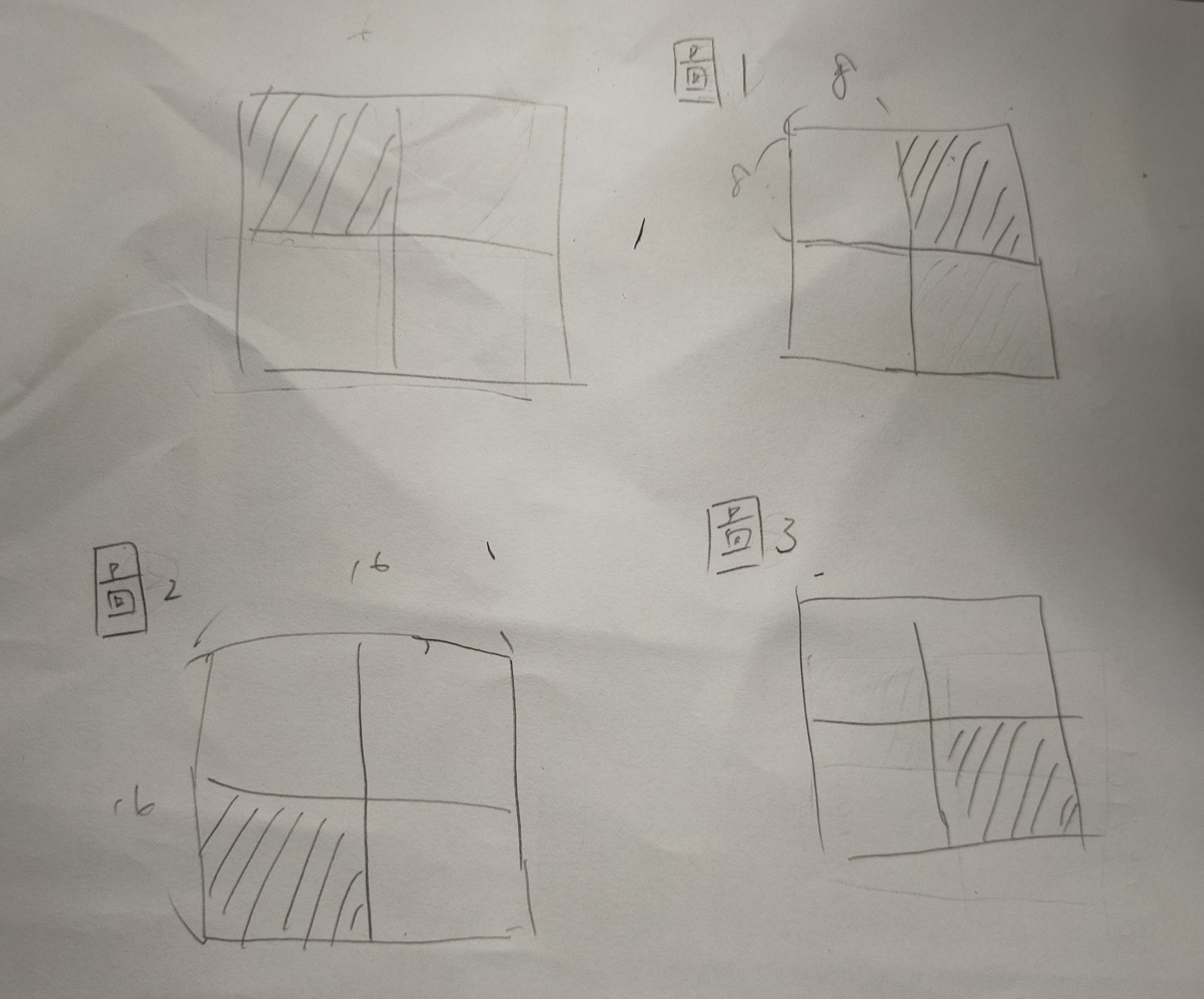
理所當然的，依據相同的方法，我們也可以靠著接線將JP49和JP50連到switch的區塊，單靠硬體決定dot-matrix及其輸出值，所以理論上是可以只靠硬體和接線和switch區塊是顯示2的。但因為人的接線速度有限以及視覺暫留的限制時間很短暫，所以考慮時間性問題的話，我們是無法只透過硬體的接線觀察出2的

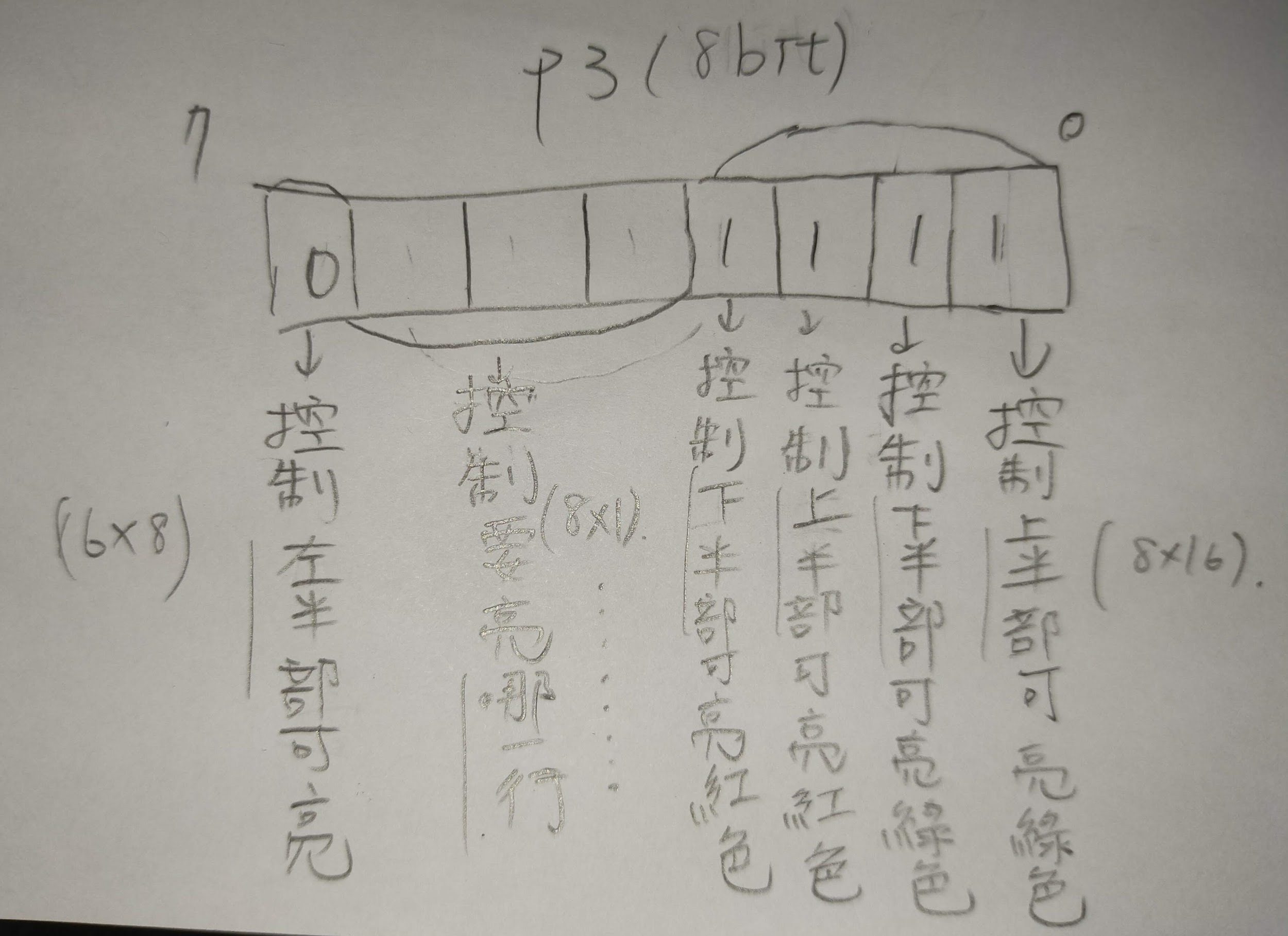
\* retrospections raised to oneself

在開始用code前記得hardware已經都準備好了，還有環境的參數在建置project之前要設定好，參數錯誤的話寫進去8051的code會有問題。要debug的時候不要一次改太多東西，一次變動一點就好比較好知道bug發生在哪。要動軟體前先確定硬體是對的(如線是否接對，電源是否有接上)，還有delay的時間要約略估計好不然會導致燈看起來不亮、延遲或不連續而有瑕疵。記得要保存值的時候用push-pop

Designated Assignment

在圖B中，左上角是原本8x8的顯示區塊，要控制顯示的區塊則與P3有關，P3有8個bit，P3[0:7]，說明如圖A。若要更改顯示區塊位置為圖B右上角圖1，則必須將P3[7]從原本的0改為1 ; 若要改變顯示區塊位置為圖B左下角圖2，則須將P3[3:0]改成1010 ; 若要更改顯示區塊位置為圖B右下角圖3，則需前兩者都執行。(圖A、B如下)

圖A



圖B