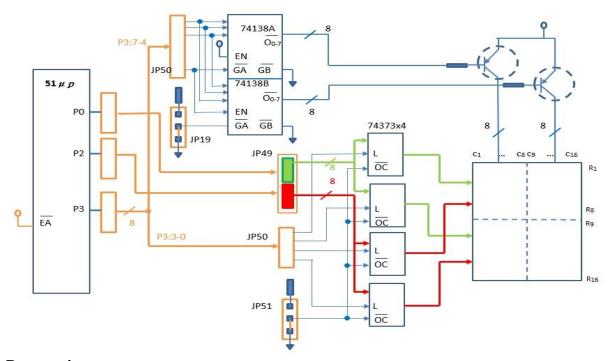
Microprocessor Lab 2.3 Report

0416106 彭敬樺 0416109 周才錢

Subject and Goal:

This lab is about using μ -Vision 51IDE residing on MegaWin82G516 to:

- The access of every individual LED-dot for ON/OFF and color control when operating with the 16x16 tri-colored dot-matrix LED module.
- Organize display patterns in static or dynamic form as required.



Preparations:

- Power cable and required connection from the output to the led input is established.
 The color of LED-dot is controlled by the output of port P0 and P2 that is connected to JP49.
- Check the correctness and check if there are any defective on the board by activating all the 16x16 LED using static/dynamic pattern display.
- Knowing the operational limit of the 16x16 LED module that can only operate interchangeably from left and right part of the LED module to be controlled at the same time. However, this is not a big problem due to the inability for human eye to perceive small delay difference between left and right and the latency for the input to be implemented is quite small.

Operating Procedure:

- Jumper-wiring for circuit setup
- Check the 16x16 LED module to see if it's working or not by running code to turn all the light on.

- Code preparation
- Task execution:
 - Start IDE51 emulation,
 - Start execution and troubleshooting if necessary.

Code Preparation:

```
org 0
                                                     anl P3, #0F0H ; ==XXX==
mov SP, #50H
                                                     mov P0, #0H
                                                     add A, #10H; A:=???
   mov P3, #0
   ;call delay
                                                     mov P3, A
   mov P0, #0FFH
                                                     call delay
                                                                  ; col8 done
   mov P2, #0FFH
                                                     djnz R6, green_2
   mov P3, #5H
                                                     anl P3, #0F0H ; ==AA==
   mov P3, #0
                                                     mov P0, #0FFH ; ==AA==
                                                     mov P3, A
                                                                  ==AA==
start:
   mov R6, #250
                                                     call delay
green_2:
   mov P3, #0
                                                 redd:
   mov P0, #0H
                                                     mov R6, #250
   mov A, #1H
                                                 red 2:
   mov P3, A
                                                     mov P3, #0
                                                     mov P2, #0H
   call delay
                ; coll done
   anl P3, #0F0H ; ==XXX==
                                                     mov A, #4H
                                                     mov P3, A
   mov P0, #7aH
   add A, #10H
                                                               ; col1 done
                                                     call delay
   mov P3, A
               ; A := ???
                                                     anl P3, #0F0H ; ==XXX==
                                                     mov P2, #7AH
   call delay
                ; col2 done
                                                     add A, #10H; A:=???
   mov R7, #4
                                                     mov P3, A
g2_loop:
                                                     call delay
                                                                 ; col2 done
   anl P3, #0F0H ; ==XXX==
                                                     mov R7, #4
   mov P0, #4AH
                                                 r2_loop:
   add A, #10H
                                                     anl P3, #0F0H ; ==XXX==
   mov P3, A
               ; A:= ???
                                                     mov P2, #4AH
   call delay
              ; col3-6 done in sequence
                                                     add A, #10H ; A:=???
   djnz R7, g2_loop
                                                     mov P3, A
   anl P3, #0F0H ; ==XXX==
                                                     call delay
                                                                  ; col3-6 done
   mov P0, #4EH
                                                     djnz R7, r2 loop
   add A, #10H
   mov P3, A
                                                     anl P3, #0F0H ; ==XXX==
   call delay
               ; col7 done
                                                     mov P2, #4EH
```

```
mov P3, A
   add A, #10H ; A:= ???
   mov P3, A
                                                     call delay
                                                                  ; col3-6 done
   call delay
                : col7 done
                                                     djnz R7, y2_loop
   anl P3, #0F0H ; ==XXX==
                                                     anl P3, #0F0H ; ==XXX==
   mov P2, #0H
                                                     mov P0, #4EH
   add A, #10H
                  ; A:= ???
                                                     mov P2, #4EH
   mov P3, A
                                                     add A, #10H
                                                                  A := ???
   call delay
                 ; col8 done
                                                     mov P3. A
   djnz R6, red_2
                                                     call delay
                                                                  ; col7 done
   anl P3, #0F0H ; ==BB==
                                                     anl P3, #0F0H ; ==XXX==
   mov P2, #0FFH ; ==BB==
                                                     mov P0, #0H
   mov P3, A
                ; ==BB==
                                                     mov P2, #0H
   call delay
                                                     add A, #10H
                                                                  A := ???
   mov R6, #250
                                                     mov P3, A
yellow_2:
                                                     call delay
                                                                  : col8 done
                                                     djnz R6, yellow_2
   mov P0, #0H
   mov P2, #0H
                                                     anl P3, #0F0H ; ==CC==
   mov A, #5H
                                                     mov P0, #0FFH ; ==CC==
   mov P3. A
                                                     mov P2, #0FFH ; ==CC==
   call delay
               ; col1 done
                                                     mov P3, A
                                                                   ; ==CC==
   anl P3, #0F0H ; ==XXX==
                                                     call delay
   mov P0, #7AH
                                                     jmp start
   mov P2, #7AH
                                                 delay: push 2
   add A, #10H; A:=???
                                                     push 3
   mov P3, A
                                                     mov R2, #2
   call delay
                ; col2 done
                                                 dd1: mov R3, #250
   mov R7, #4
                                                     djnz R3,$
y2_loop:
                                                     djnz R2, dd1
   anl P3, #0F0H ; ==XXX==
                                                     pop 3
   mov P0, #4AH
                                                     pop 2
   mov P2, #4AH
                                                     ret
   add A, #10H ; A:= ???
                                                     end
```

Observation:

- The code is running well, however there are some issue for some of the existing boards (including the one used by our group) that cause the initialization at the beginning of the code being overrun by another signal in the middle of the operation. The way to solved this is to regularly initialize P0, P2, P3 to the wanted value
- When the ==AA== code line is omitted, the supposedly blank off LED area that formed the '2' character turn into yellowish color due to lack of re-initialization of P0 and P3 before redoing the loop.

- When the ==BB== code line is omitted, there are no change that can be observed, the
 reason is due to the coincidence where the last value suited the need for the initial value
 for the next iteration.
- When the ==CC== code line is omitted, the green phase for the LED module change into orange phase, which let the entire code has 2 orange phase and 1 red phase. The new orange phase is the result of failing the initialization and let the green color combined with red, which represent orange / yellow color. This also affected the supposedly blank portion of the LED when failure for initialization happen into red color.
- For the code presented above, we observe no change whatsoever in the behavior of the LED module.
- For every value moved into P3 port, which control the on/off of the LED, we can handle the power for the right half of the 16x16 LED module.
- The part of code that is quite similar can be moved to a function that is ready to be called by using the parameter that has the value of pointer to the wanted initial argument that slightly differ for the wanted color phase showed.

Comprehensive evaluation:

- It is possible to let any combination part of the LED module to look as if being turned on simultaneously. However, this is actually implemented by interchangeably controlling and handling the power for left and right part of the LED module in a very fast way without being noticed by the user.
- To turn on one column of the LED module by manual jumper wiring, we can let the corresponding signal that value wanted to be sent into the corresponding pin. The area of J45 can be set to the wanted value while connected to the JP50, which control the LED on/off
- We can displayed the decimal '2' on the upper right 8x8 LED module by manual wiring plus the code used in driving the LED in upper left of the LED module, by using the same method of connecting J45 wire to the JP50 and set the wanted value to moved the activation the right side.

Designated Assignment:

K = mod(19,3) == 1 (lower left 8x8 LED module)

```
org 0
                                       call delay
                                                                            add A, #10H
mov SP, #50H
                                       anl P3, #0F0H
                                                                            mov P3, A
                                       mov P0, #7aH
   mov P3, #0
                                                                            call delay
                                       add A, #10H
                                                                            anl P3, #0F0H
   ;call delay
                                                                            mov P0, #0H
   mov P0, #0FFH
                                       mov P3, A
   mov P2, #0FFH
                                       call delay
                                                                            add A, #10H
   mov P3, #5H
                                                                            mov P3, A
   mov P3, #0
                                       mov R7, #4
                                                                            call delay
                                                                            djnz R6, green_2
                                   g2_loop:
                                       anl P3, #0F0H
                                                                            mov P0, #0FFH
start:
                                       mov P0, #4AH
                                                                            mov P2, #0FFH
greenn:
                                                                            mov P3, #5H
   mov R6, #250
                                       add A, #10H
                                                                            mov P3, #0
                                       mov P3, A
green_2:
   mov P3, #0
                                       call delay
                                                                            call delay
   mov P0, #0H
                                       djnz R7, g2_loop
   mov A, #2H //#1H
                                       anl P3, #0F0H
                                                                        redd:
   mov P3, A
                                       mov P0, #4EH
                                                                            mov R6, #250
```

red_2:	mov P0, #0FFH	mov P0, #4EH
mov P3, #0	mov P2, #0FFH	mov P2, #4EH
mov P2, #0H	mov P3, #5H	add A, #10H
mov A, #08H //#4H	mov P3, #0	mov P3, A
mov P3, A	call delay	call delay
call delay		anl P3, #0F0H
anl P3, #0F0H	yelloww:	mov P0, #0H
mov P2, #7AH	mov R6, #250	mov P2, #0H
add A, #10H	yellow_2:	add A, #10H
mov P3, A	mov P0, #0H	mov P3, A
call delay	mov P2, #0H	call delay
	mov A, #0AH //#5H	djnz R6, yellow_2
mov R7, #4	mov P3, A	mov P0, #0FFH
r2_loop:	call delay	mov P2, #0FFH
anl P3, #0F0H	anl P3, #0F0H	mov P3, #5H
mov P2, #4AH	mov P0, #7AH	mov P3, #0
add A, #10H	mov P2, #7AH	call delay
mov P3, A	add A, #10H	jmp start
call delay	mov P3, A	
djnz R7, r2_loop	call delay	delay: push 2
anl P3, #0F0H		push 3
mov P2, #4EH	mov R7, #4	mov R2, #2
add A, #10H	y2_loop:	dd1: mov R3, #250
mov P3, A	anl P3, #0F0H	djnz R3,\$
call delay	mov P0, #4AH	djnz R2, dd1
anl P3, #0F0H	mov P2, #4AH	pop 3
mov P2, #0H	add A, #10H	pop 2
add A, #10H	mov P3, A	ret
mov P3, A	call delay	end
call delay	djnz R7, y2_loop	
djnz R6, red_2	anl P3, #0F0H	

Additional Testing:

The code below will be able to drive all 16x16 LED module to present the '2' character in each 8x8 LED module area

```
org 0
                                       call foo_red2
                                                                               djnz R2, dd1
mov SP, #50H
                                         dinz R6, red 2
                                                                               pop 3
   mov P3, #0
                                         anl P3, #0F0H
                                                                               pop 2
   ;call delay
                                         mov P2, #0FFH
                                                                               ret
   mov P0, #0FFH
                                         mov P3, A
                                         call delay
   mov P2, #0FFH
                                                                           foo_green1:
   mov P3, #5H
                                                                            mov P3, #0
   mov P3, #0
                                     yelloww:
                                                                               mov P0, #0H
                                         mov R6, #250
                                                                               mov A, #3H //#1H
                                     yellow_2:
                                                                               mov P3, A
start:
                                         call foo_yellow1
                                                                               call delay
greenn:
   mov R6, #250
                                       call foo_yellow2
                                                                               anl P3, #0F0H
                                         djnz R6, yellow_2
                                                                               mov P0, #7aH
green_2:
   call foo_green1
                                         anl P3, #0F0H
                                                                               add A, #10H
                                         mov P0, #0FFH
                                                                               mov P3, A
 call foo_green2
   djnz R6, green_2
                                         mov P2, #0FFH
                                                                               call delay
   anl P3, #0F0H
                                         mov P3, A
   mov P0, #0FFH
                                         call delay
                                                                               mov R7, #4
   mov P3. A
                                                                           g2_loop1:
                                         jmp start
   call delay
                                                                               anl P3, #0F0H
                                     delay: push 2
                                                                               mov P0, #4AH
redd:
                                         push 3
                                                                               add A, #10H
   mov R6, #250
                                         mov R2, #1 //#2
                                                                               mov P3, A
                                                                               call delay
                                     dd1: mov R3, #250
red 2:
                                         djnz R3,$
                                                                               djnz R7, g2_loop1
   call foo_red1
```

anl P3, #0F0H	djnz R7, r2_loop1	add A, #10H
mov P0, #4EH	anl P3, #0F0H	mov P3, A
add A, #10H	mov P2, #4EH	call delay
mov P3, A	add A, #10H	djnz R7, y2_loop1
call delay	mov P3, A	anl P3, #0F0H
anl P3, #0F0H	call delay	mov P0, #4EH
mov P0, #0H	anl P3, #0F0H	mov P2, #4EH
add A, #10H	mov P2, #0H	add A, #10H
mov P3, A	add A, #10H	mov P3, A
call delay	mov P3, A	call delay
ret	call delay	anl P3, #0F0H
	ret	mov P0, #0H
foo_green2:		mov P2, #0H
mov P3, #0	foo_red2:	add A, #10H
mov P0, #0H	mov P3, #0	mov P3, A
mov A, #083H //#1H	mov P2, #0H	call delay
mov P3, A	mov A, #08CH //#4H	ret
call delay	mov P3, A	
anl P3, #0F0H	call delay	foo_yellow2:
mov P0, #7aH	anl P3, #0F0H	mov P0, #0H
add A, #10H	mov P2, #7AH	mov P2, #0H
mov P3, A	add A, #10H	mov A, #08FH //#5H
call delay	mov P3, A	mov P3, A
	call delay	call delay
mov R7, #4		anl P3, #0F0H
g2_loop2:	mov R7, #4	mov P0, #7AH
anl P3, #0F0H	r2_loop2:	mov P2, #7AH
mov P0, #4AH	anl P3, #0F0H	add A, #10H
add A, #10H	mov P2, #4AH	mov P3, A
mov P3, A	add A, #10H	call delay
call delay	mov P3, A	cuii uciuy
2	,	may D7 #4
djnz R7, g2_loop2	call delay	mov R7, #4
anl P3, #0F0H	djnz R7, r2_loop2	y2_loop2:
mov P0, #4EH	anl P3, #0F0H	anl P3, #0F0H
add A, #10H	mov P2, #4EH	mov P0, #4AH
mov P3, A	add A, #10H	mov P2, #4AH
call delay	mov P3, A	add A, #10H
anl P3, #0F0H	call delay	mov P3, A
mov P0, #0H	anl P3, #0F0H	call delay
add A, #10H	mov P2, #0H	djnz R7, y2_loop2
mov P3, A	add A, #10H	anl P3, #0F0H
call delay	mov P3, A	mov P0, #4EH
ret	call delay	mov P2, #4EH
	ret	add A, #10H
foo_red1:		mov P3, A
mov P3, #0	foo_yellow1:	call delay
mov P2, #0H	mov P0, #0H	anl P3, #0F0H
mov A, #0CH //#4H	mov P2, #0H	mov P0, #0H
mov P3, A	mov A, #0FH //#5H	mov P2, #0H
call delay	mov P3, A	add A, #10H
· ·		
anl P3, #0F0H	call delay	mov P3, A
mov P2, #7AH	anl P3, #0F0H	call delay
add A, #10H	mov P0, #7AH	ret
mov P3, A	mov P2, #7AH	end
call delay	add A, #10H	
	mov P3, A	
mov R7, #4	call delay	
r2_loop1:		
_ ·	mov P7 #4	
anl P3, #0F0H	mov R7, #4	
mov P2, #4AH	y2_loop1:	
add A, #10H	anl P3, #0F0H	
mov P3, A	mov P0, #4AH	
call delay	mov P2, #4AH	