**Microprocessor Lab-work #2.3**

**dot-matrix LEDs**  100-11-14

[1] **Subject and goals**

(a) How to access every individual LED-dot for ON/OFF and color control when operating with the 16x16 tri-colored dot-matrix LED module.

(b) Coding capacity of organizing display patterns in static or dynamic form as required.

[2] **Preparations**

(a) **Refer to the ckt schematic diagram**:

(a.1) how the 16x16 LED module may operate (the inputs for column/row/color selection)?

(a.2) functions of TTLs 74138, 74244 and 74373?

(a.3) data path from 51CPU to the 16x16 LED modules?

(b) **Datasheets reading**:

(b.1) 74138 and 74373

(c) **Readiness evaluation:**

16x16 LED module physically consists of four 8x8 tri-colored dot-matrix LED components, interconnected in such a way that any of the 4 operating modes are allowed --

\*\* operation on any single 8x8 LED component (one in four),

\*\* operation on any two LED components in 8x16 dot format (i.e., selection between upper/lower half of the 16x16 module),

\*\* operation on any two LED components in 16x8 dot format (i.e., selection between left/right half of the 16x16 module),

\*\* operation on the entire module in 16x16 dot format.

Can you or can you not

(c.1) handle the jumper-wiring of the 16x16 LED module for any of the 4 operating modes?

(c.2) write the codes for any static/dynamic pattern display after step (c.1) is done?

(c.3) describe the operational limits of the 16x16 LED module imposed by the circuitry?

(c.4) check the discrete LED module to see if it’s working or not by manual wiring the circuitry? (e.g., turn on an 8x1 or a 16x1 LED column in the left-half of the 16x16 module in RED manually? In Green? In Orange?)

(c.5) do the same as in (c.4), but for the right-half of the 16x16 module?

[3] **Lab-exercise for all: simple demonstration on one single 8x8 LED component**

(a)**Operating Procedure**

(a.1) **jumper-wiring for ckt setup**

for using upper-half of 16x16 LED module (8\*16 unit)

JP50 74138x2

8

**51μ*p*** EN

G2A G2B

P3 c1 … c8 c9 … c16

8 74244x2 R1

EN

P0 G2A G2B 8

P19 1G 2G R8

8 R9

EA P2 JP51

1G 2G

JP49 R16

(a.2) **jumper-wiring for ckt setup**

using entire 16x16 module 74138A 8

P3:7-4 O0-7

EN

JP50 GA GB

**51μ*p*** 74138B

O0-7

P0 EN 8

GA GB 74373x4 8 8

JP19

P2 JP49 L c1 … c8 c9 … c16

8 OC R1

8

P3 L R8

EA 8 OC R9

P3:3-0 JP50

L

OC

R16

JP51 L

OC

(a.3) **code preparation**:

\*\* edit the following sample 51 assembly codes

\*\* get the code ready for execution under IDE51 emulator.

; the sample code is intended for displaying a decimal “2” on the upper-left 8x8 LED component in green, red and yellow color

; alternately, with the 16x16 LED module wired for the 16\*16 operation as shown in (a.2) .

; the sample code guarantees neither syntax error free nor bug free; fix all errors encountered.

; code testing procedure (strongly recommended)

; (1) first run the green\_2 loop alone, making sure that you see a “2” in green appear on the designated 8x8 LED

; (2) then run the red\_2 loop alone to see what should appear.

; (3) finally run the entire code altogether.

; P0: GREEN [why????]

; P2: RED [why????]

; P3[7] 138A-GA P3[3] Latch: Red for row9-16

; P3[6] 138-22 P3[2] Latch: Red for row1-8

; P3[5] 138-21 P3[1] Latch: Green for row9-16

; P3[4] 138-20 P3[0] Latch: Green for row1-8

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 delay ; col1 done

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

mov P0, #7aH

add A, #10H

mov P3, A ; A:= ???

call delay ; col2 done

mov R7, #4

g2\_loop:

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

mov P0, #4AH

add A, #10H

mov P3, A ; A:= ???

call delay ; col3-6 done in sequence

djnz R7, g2\_loop

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

mov P0, #4EH

add A, #10H

mov P3, A

call delay ; col7 done

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

mov P0, #0H

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

mov P3, A

call delay ; col8 done

djnz R6, green\_2

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

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

mov P3, A ; ==AA==

call delay

redd:

mov R6, #250

red\_2:

mov P3, #0

mov P2, #0H

mov A, #4H

mov P3, A

call delay ; col1 done

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

mov P2, #7AH

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

mov P3, A

call delay ; col2 done

mov R7, #4

r2\_loop:

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

mov P2, #4AH

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

mov P3, A

call delay ; col3-6 done

djnz R7, r2\_loop

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

mov P2, #4EH

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

mov P3, A

call delay ; col7 done

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

mov P2, #0H

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

mov P3, A

call delay ; col8 done

djnz R6, red\_2

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

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

mov P3, A ; ==BB==

call delay

mov R6, #250

yellow\_2:

mov P0, #0H

mov P2, #0H

mov A, #5H

mov P3, A

call delay ; col1 done

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

mov P0, #7AH

mov P2, #7AH

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

mov P3, A

call delay ; 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 delay ; 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 delay ; col7 done

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

mov P0, #0H

mov P2, #0H

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

mov P3, A

call delay ; col8 done

djnz R6, yellow\_2

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

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

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

mov P3, A ; ==CC==

call delay

jmp start

delay: push 2

push 3

mov R2, #2

dd1: mov R3, #250

djnz R3, $

djnz R2, dd1

pop 3

pop 2

ret

end

\*\* The sample codes guarantee neither syntax err-free nor runtime err-free. Fix all syntax errs due to typo or whatever causes.

(a.4) **task execution:**

\*\* **using ckt wiring for the 16x16 module operation as in (a.2)**

\*\* start IDE51 emulation for the code prepared at (a.3)

\*\* start trouble-shooting if necessary

(b) Observations

(b.1) Is the code running well? Why or why not?

(b.2) What may happen to the display if the instructions marked by **==AA==** being omitted? Why so?

And the omission of code lines marked by **==BB==**? And those by **==CC==**?

(b.3) Will the omission of the instruction marked by **==XXX==** cause any undesired effect to the display? Why so or why not so?

(b.4) Can you modify the code so as to make it shorter, quicker, in better code structure, or produce a more steady display?

(b.4) In which step(s) of the code, the 74138 handling the power for the right half of the 16x16 LED is shut down? What may be the side effect of this particular step?

(b.5) Using the circuit wiring to drive only the upper-half of the 16x16 module, as given in step [3](a.1), is it possible to run the same codes without any modification for the same display sequence on the upper-left 8x8 LED module?

[4] **Comprehension evaluation:**

(a) Is it possible to turn on simultaneously one row of LEDs on the upper-left 8x8 LED component? Why or why not?

What about simultaneously turning on one row of LED on any of the rest three 8x8 modules? The upper 8x16 module? The lower 8x16 module? The left 16x8 module? The right 16x8 module?

(b) Can you turn on one column of LEDs on the upper-left 8x8 component by wiring jumpers manually? How?

(c) Using exactly the same code, is it possible to get the decimal “2” displayed on the upper-right 8x8 LED by simply rewiring the jumpers? Why or why not?

[5] **Designated Assignment**

Let **k**=mod(ID, 3), where ID is the table # of yours in the laboratory. Please write a simple 51 assembly code to fulfill the assignment given in the question Q(**k**).

The coding requirements concern the driving of 16x16 dot-matrix LEDs on the ckt board, wired as in [3](a.2).

Q(0) Modify the codes so that the same display sequence appears on the upper-right 8x8 module.

Q(1) Modify the codes so that the same display sequence appears on the lower-left 8x8 module.

Q(2) Modify the codes so that the same display sequence appears on the lower-right 8x8 module.