Microprocessor Lab-work #2.1uvision

Use of discrete LEDs

100-11-14

[1] **Subject and goals**

(a) The access of every individual LED for ON/OFF control in the 2 sets of discrete LED modules.

(b) Organized display patterns in static or dynamic form can be achieved as required.

[2] **Preparations**

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

(a.1) how ON/OFF control of the LED module is to be done?

(a.2) functions of TTL 74244 and its role in the ckt?

(a.3) functions of the *array resistors* RN4 and RN6, and their roles in the ckt?

(a.4) data path from 51CPU to the discrete LED modules?

(b) **Datasheets reading**:

(b.1) TTL 74244

(c) **Readiness-evaluation:**

Can you or can you not

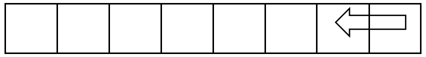
(c.1) check the discrete LED module to see if it’s working or not by manual wiring the circuitry?

(c.2) carry out trouble shooting along the path way when the lab-work isn’t going as expected? How will you do that?

[3] **Lab-work for all:**

The task here is to use the two discrete LDE modules for dynamic display patterns as graphically depicted below.

phase 1

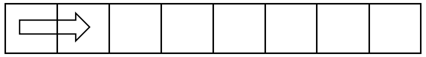


module1 (right to left flickering)

module2 (ON-OFF switching)

ON ⟺ OFF

phase 2

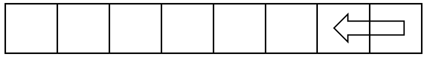


module1 (left to right flickering)

module2 (ON-OFF switching)

OFF	**ON**	**OFF**	**ON**

phase 3

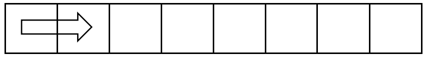


module1 (right to left flickering)

module2 (ON-OFF switching)

OFF ⟺ ON

phase 4



module1 (left to right flickering)

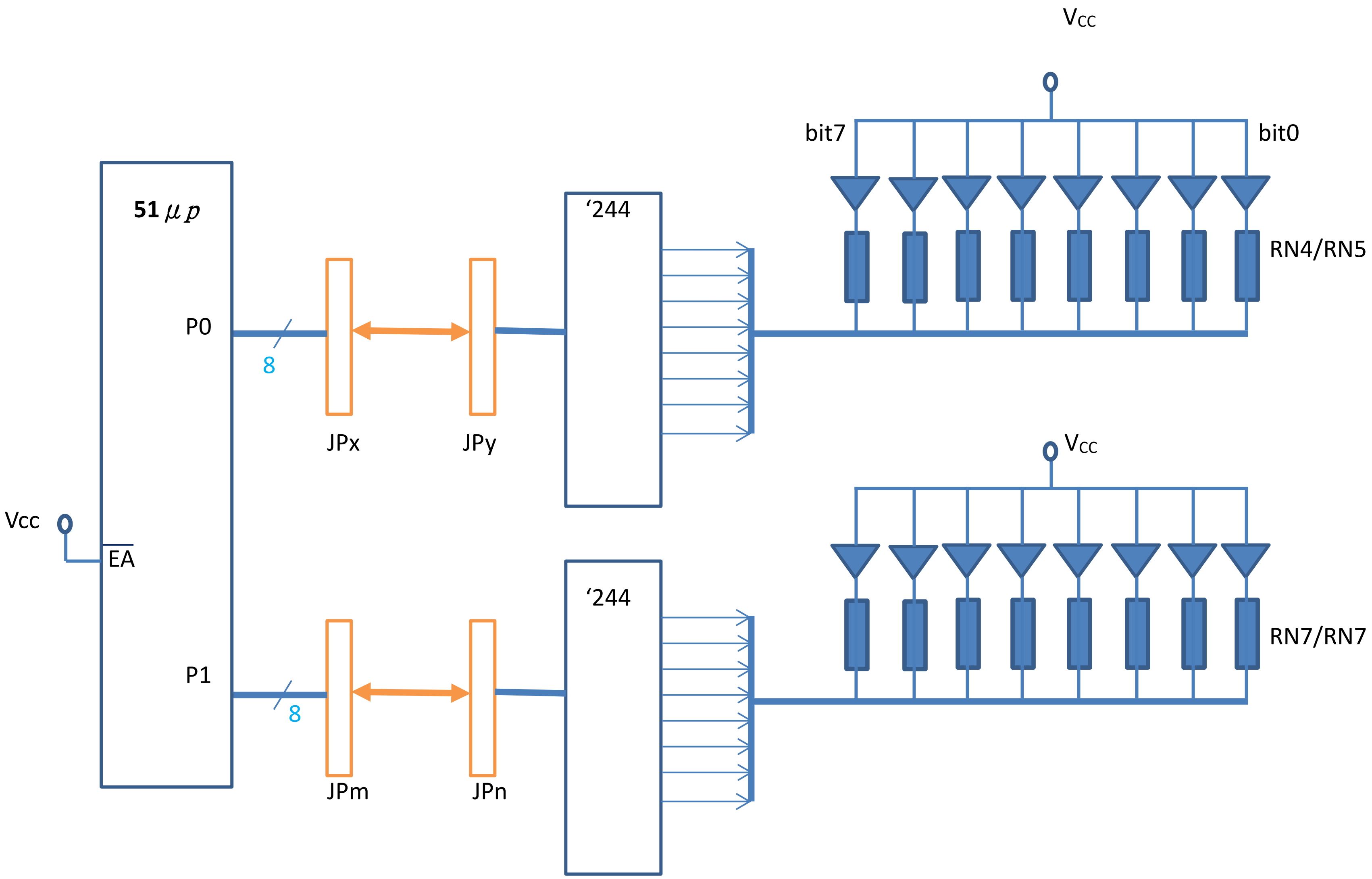
module2 (ON-OFF switching)

ON	**OFF**	**ON**	**OFF**

(a) **Operating Procedure**

(a.1) jumper-wiring for ckt setup

Refer to the schematic circuit diagram, do all jumper-wiring necessary for setting up the circuitry as required below.


Vcc
bit7
bit0
51 μp
'244
RN4/RN5
P0
8
JPx
JPy
Vcc
Vcc
EA
'244
RN7/RN7
P1
8
JPm
JPn

(a.2) code preparation:

** edit the following sample 51 assembly code under µVsion51.

```
    org     0
    mov     sp, #50H
    clr     c
    mov     a, #0feH
    mov     R7, a

    mov     a, #0fH
mk1:
    cpl     a
    mov     r6, a
    mov     p1, A
    mov     a, r7
    mov     p0, a
    call    delay
    rlc     a
    mov     r7, a
    mov     a, r6
    jc      mk1

    mov     a, #0ccH
mk2:
    cpl     a
    mov     r6, a
    mov     p1, a
    mov     a, r7
    mov     p0, a
    call    delay
    rrc     a
    mov     r7,a
    mov     a, r6
    jc      mk2

    mov     a, #0f0H
mk3:
    cpl     a
    mov     r6, a
    mov     p0, a
    mov     a, r7
    mov     p1, a
    call    delay
    rlc     a
    mov     r7, a
    mov     a, r6
    jc      mk3

    mov     a, #0ccH
mk4:
    cpl     a           ; XXX
    mov     r6, a
    mov     p0, a
    mov     a, r7
    mov     p1, a
    call    delay
    rrc     a
    mov     r7,a
    mov     a, r6
    jc      mk4

    mov     a, #0fH
    jmp     mk1
delay:
    push    5
            ; push R5???
    push    6
    push    7
    mov     r5, #2
  dd1:
    mov     r6, #200
  dd2:
    mov     r7, #250
    djnz    r7, $
    djnz    r6, dd2
    djnz    r5, dd1
    pop     7
    pop     6
    pop     5
    ret
```

```
end
```

(a.3) task execution:

** start IDE51 emulation,

** start execution and trouble-shooting if necessary.

(b) **Observations**

(b.1) Through the display of IDE51 in emulation mode, get yourself acquainted with the machine codes of instructions in the sample program.

(b.2) Is the code running well? If not, congratulate you that you have a chance for getting more experience in trouble-shooting. If so, also congratulate you that you may call it a day

(b.3) *Any possibility of making the codes more concise?*

[4] **Comprehension evaluation**

(a) Can you identify the stack status (where SP is pointing to, contents of the stack, etc.) at any instance during the task execution?

(b) When seeing a specific patterns appearing on the two LED modules, can you tell exactly which instruction line (or some instruction lines) is (are) possibly being executed? And the contents of associated registers?

(c) For the code line marked with **;XXX**, how would the display pattern sequence changed if it is removed?