Microcontroller System Laboratory

Experiment 2 - Part 1

Utkarsh Patel 18EC35034

Objective

***Real time minute second clock*:** Design a digital clock using the 7-segment display modules. The clock normally displays the time in mm:ss format. It updates time automatically using the timer interrupt of the microcontroller.

Circuit Diagram

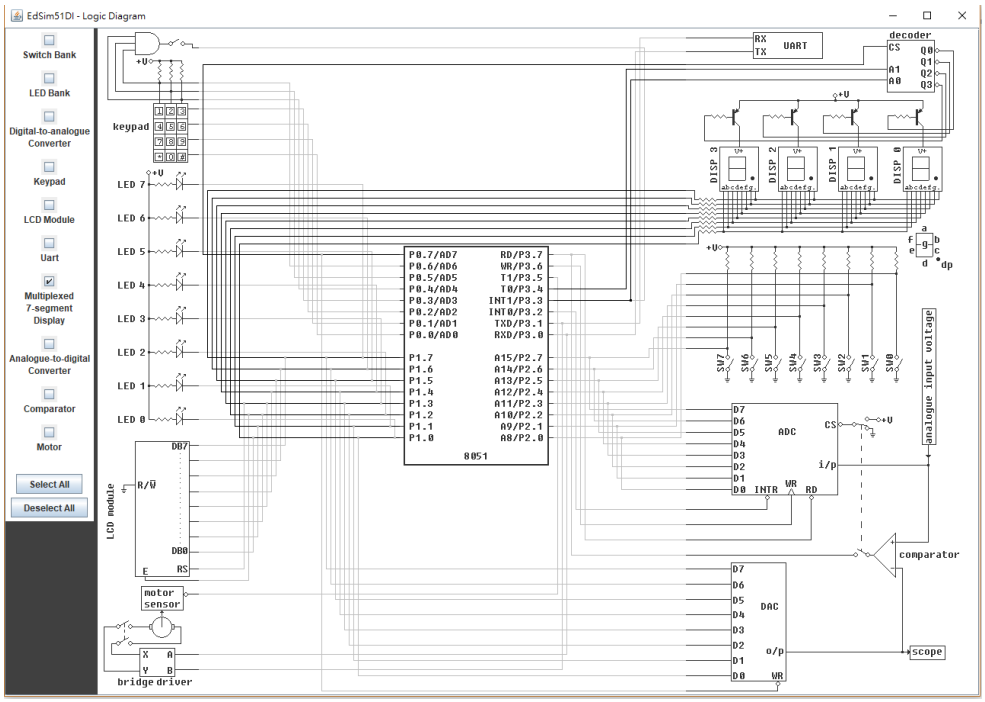

Fig. 1. Circuit diagram of connections in Edsim simulator

Code

```
; Author: Utkarsh Patel (18EC35034)
;
; Specification:
; Design a digital clock using the 7 segment display modules.
; The clock normally displays the time in mm-ss format.
; It updates time automatically using the timer interrupt of the microcontroller.
;
; Run this code with update frequency = 100

mov 30H,#00                ; temporary location to store m1
mov 31H,#00                ; temporary location to store m0
mov 32H,#00                ; temporary location to store s1
mov 33H,#00                ; temporary location to store s0
mov TMOD,#00H              ; setting TMOD
```

```
start:
    inc 33H                 ; incremeting s0
    mov A, 33H
    cjne A, #0AH, X         ; checking whether s0 > 9, if true proceed, else continue
    mov 33H, #00            ; s0 = 0

    inc 32H                 ; incrementing s1
    mov A, 32H
    cjne A, #06, X          ; checking whether s1 > 5, if true proceed, else continue
    mov 32H,#00             ; s1 = 0

    inc 31H                 ; incrementing m0
    mov A, 31H
    cjne A, #0AH, X         ; checking whether m0 > 9, if true proceed, else continue
    mov 31H,#00             ; m0 = 0

    inc 30H                 ; incrementing m1
    mov A, 30H
    cjne A, #06, X          ; checking whether m1 > 5, if true proceed, else continue
    MOV 30H,#00;

    X:                      ; display time on 7-seg display
        clr TF1
        acall displayClock

    jmp start

displayClock: ; displaying clock on multiplexed 7-seg display
    ; displaying s0 on disp0
    mov A, 33H
    clr P3.4
    clr P3.3
    acall displayDigit

    ; displaying s1 on disp1
    mov A, 32H
    setb P3.3
    acall displayDigit

    ; displaying m0 on disp2
    mov A, 31H
    clr P3.3
    setb P3.4
    acall displayDigit

    ; displaying m1 on disp3
    mov A, 30H
```

```
    setb P3.3
    acall displayDigit

    ret

displayDigit: ; display current digit on 7-segment display
    add A, #70H
    mov R1, A
    mov P1, @R1
    acall delay
    ret

delay: ; creating a delay of 0.25 sec
    mov R0, #125
    djnz R0, $
    ret
```

Simulation

Visit https://drive.google.com/file/d/1rK7lC2zrx1T9eMDJ7HiLbctcGhlF0FGO/view?usp=sharing to see the simulation of Part 1.

Discussion

- In this part, a real time minute second clock is implemented in Edsim simulator, which displays the time in mm:ss format.
- The code starts with assigning temporary locations to variable needed to simulate the clock.
- The initial state of the clock is set to 00:00, which can be modified by changing the initialization in the code.
- The first step in the design is the delay module. As the system clock frequency is set to 12 MHz, one DJNZ instruction takes 2 us. Repeating this process 125 times and setting update frequency to 100, the total delay will be 0.25 sec.
- The delay module is then used in displayDigit module, which is used to display current digit in the clock on the corresponding 7-segment display.
- The displayDigit module is then used in displayClock module. As the requirement to represent the time in mm:ss $(m_1 m_0 : s_1 s_0)$ format, we use display #0 for displaying s_0, display #1 for s_1, display #2 for m_0 and display #3 for m_1.
- As the 7-segment displays are multiplexed, all of the displays can't be used simultaneously. Hence, we use display #0, display #1, display #2 and display #3 in round-robin method.
- As the clock has to be real-time, the required delay would be 0.25 sec for each display. This is why the total delay of the delay module is set to 0.25 sec.
- As all the helper modules have been created, we need a method to update the temporary variables declared for storing values for m_1, m_0, s_1 and s_0.
- In the start module, we try incrementing value stored in s_0. If $s_0 \leq 9$, we jump to submodule X which calls the displayClock module. Else, we set s_0 to 0, and proceed to increment s_1.
- Similar process is followed to check overflow condition of s_1, m_0 and m_1.