Microcontroller System Laboratory

Experiment 2 – Part 2 & 3

Utkarsh Patel 18EC35034

Part 2 – Stop Watch

Objective

In stopwatch mode, there are two more buttons namely, ***start*** and ***stop.*** On pressing the start button, the stopwatch sets to zero and continue counting time and then on pressing the stop button it display the counted duration.

Circuit Diagram

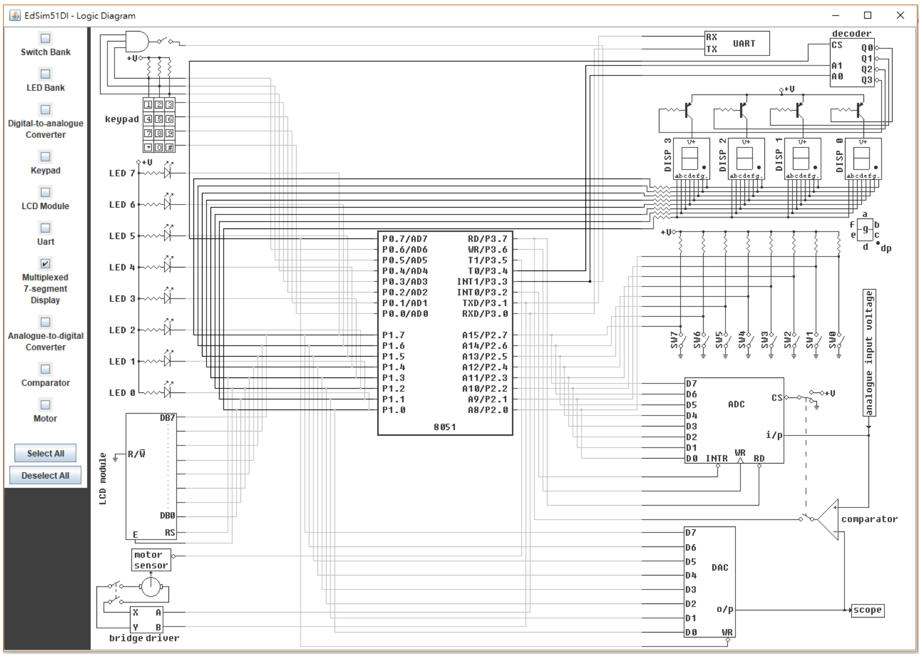

Fig. 1. Circuit diagram

Code

```
; Instructions:
; * Press P2.0 to start stop-watch
; * Run this code with update frequency 100

org 0000H
mov 70H, #11000000B          ; Codes for digits stored from 70H
mov 71H, #11111001B
mov 72H, #10100100B
mov 73H, #10110000B
mov 74H, #10011001B
mov 75H, #10010010B
mov 76H, #10000010B
mov 77H, #11111000B
mov 78H, #10000000B
mov 79H, #10010000B

mov 40H, #00        ; temporary loc to store m1
mov 41H, #00        ; temporary loc to store m0
mov 42H, #00        ; temporary loc to store s1
mov 43H, #00        ; temporary loc to store s0
mov TMOD, #00H              ; setting TMOD

start:
  inc 43H                 ; increment s0
  mov A, 43H
  cjne A, #0AH, Y         ; check s0 > 9
  mov 43H, #00            ; set s0 to 0

  inc 42H                 ; increment s1
  mov A, 42H
  cjne A, #06, Y          ; check s1 > 5
  mov 42H, #00            ; set s1 = 0

  inc 41H                 ; increment m0
  mov A, 41H
  cjne A, #0AH, Y         ; check m0 > 9
  mov 41H, #00            ; set m0 = 0

  inc 40H                 ; increment m1
  mov A, 40H
  cjne A, #06, Y          ; check m1 > 5
  mov 40H, #00            ; set m1 = 0

  Y:
    clr TF1
    jb P2.0, resetStopWatch ; if P2.0 is 0, always goto resetStopWatch module
    acall displayStopWatch  ; else no need to reset
  jmp start

displayStopWatch: ; displaying clock on multiplexed 7-seg display
  ; displaying s0 on disp0
  mov A, 40H
  setb P3.4
  setb P3.3
  acall displayDigit

  ; displaying s1 on disp1
  mov A, 41H
  clr P3.3
  acall displayDigit

  ; displaying m0 on disp2
  mov A, 42H
  setb P3.3
  clr P3.4
  acall displayDigit

  ; displaying m1 on disp3
  mov A, 43H
  clr P3.3
  acall displayDigit

  ret

resetStopWatch:
  mov 40H, #00
  mov 41H, #00
  mov 42H, #00
  mov 43H, #00
  acall displayStopWatch
  jmp start

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/1ZM_NCkbMu459QPb7JX2Fwdfw0-EJFs63/view?usp=sharing to see the simulation of Part 1.

Discussion

- In this part, a stop-watch is implemented using edsim simulator in mm:ss format.
- To start the stop-watch, P2.0 must be set to logic high, and to stop, it must be set to logic low via button available in edsim simulator
- The code starts with assigning temporary RAM locations to store the state of stop-watch in mm:ss format.
- 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 displayStopWatch 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 displayStopWatch module. Else, we set s_0 to 0, and proceed to increment s_1.
- As P2.0 is used for start/stop button, while P2.0 is set to logic low level, the stop-watch must not start.
- As in start module, the values of variables representing the state of stop-watch is incremented all the time, a resetStopWatch module is required to always reset the stop-watch as long as P2.0 remains at logic-low level.

Part 3 – Real-time minute-second clock & Stop-watch using mode switch

Objective

On pressing the mode switch, the display changes to stopwatch mode in mm-ss format. On pressing the mode button once more, the display returns to show clock time. It should be noted that in the stopwatch mode, both normal clock and stopwatch clock get updated with timer interrupt. This ensures that the normal time also gets updated during the run of stopwatch.

Circuit Diagram

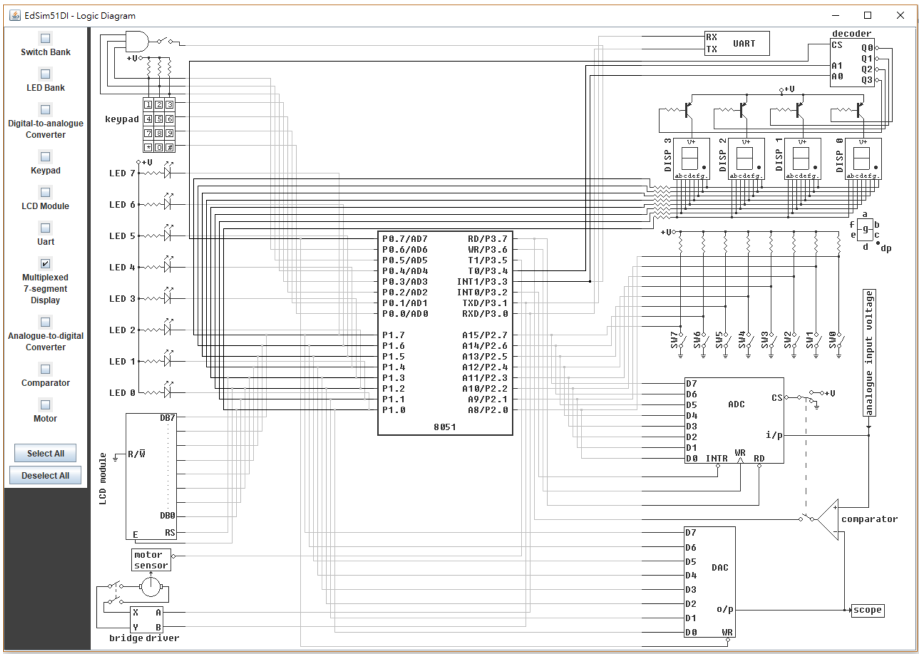

Fig. 2. Circuit diagram

Code

```
; Instructions:
; * Default mode is clock mode
; * Press P2.7 to switch to stop-watch mode and release it to switch to clock mode
; * Press P2.0 to start stop-watch and release it to reset stop-watch
; * Run this code with update frequency 100

org 0000H
mov 70H, #11000000B          ; Codes for digits stored from 70H
mov 71H, #11111001B
mov 72H, #10100100B
mov 73H, #10110000B
mov 74H, #10011001B
mov 75H, #10010010B
mov 76H, #10000010B
mov 77H, #11111000B
mov 78H, #10000000B
mov 79H, #10010000B

; temporary location of storing clock state
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

; temporary location for storing stop-watch mode
mov 40H, #00                 ; temporary loc to store m1
mov 41H, #00                 ; temporary loc to store m0
mov 42H, #00                 ; temporary loc to store s1
mov 43H, #00                 ; temporary loc to store s0
mov TMOD, #00H               ; setting TMOD

start:
  acall changeClockState
  jmp start

changeClockState:          ; changing state of the clock
  inc 33H                  ; increment s0
  mov A, 33H
  cjne A, #0AH, X          ; check s0 > 9
  mov 33H, #00             ; set s0 to 0

  inc 32H                  ; increment s1
  mov A, 32H
  cjne A, #06, X           ; check s1 > 5
  mov 32H, #00             ; set s1 = 0

  inc 31H                  ; increment m0
  mov A, 31H
  cjne A, #0AH, X          ; check m0 > 9
  mov 31H, #00             ; set m0 = 0

  inc 30H                  ; increment m1
  mov A, 30H
  cjne A, #06, X           ; check m1 > 5
  mov 30H, #00             ; set m1 = 0
  acall X

changeStopWatchState:      ; changing stop-watch state
  inc 43H                  ; increment s0
  mov A, 43H
  cjne A, #0AH, Y          ; check s0 > 9
  mov 43H, #00             ; set s0 to 0

  inc 42H                  ; increment s1
  mov A, 42H
  cjne A, #06, Y           ; check s1 > 5
  mov 42H, #00             ; set s1 = 0
  inc 41H                 ; increment m0
  mov A, 41H
  cjne A, #0AH, Y         ; check m0 > 9
  mov 41H, #00            ; set m0 = 0

  inc 40H                 ; increment m1
  mov A, 40H
  cjne A, #06, Y          ; check m1 > 5
  mov 40H, #00            ; set m1 = 0
  acall Y
X:
  clr TF1
  jb P2.7, displayClock           ; if P2.7 is zero, call displayClock module
  jnb P2.7, changeStopWatchState  ; else           , call changeStopWatchState mod
ule

Y:
  clr TF1
  jb P2.7, displayClock     ; if P2.7 is zero, display state of clock on 7-seg
  jb P2.0, resetStopWatch   ; if P2.0, reset the stop-watch
  jnb P2.7, displayStopWatch; if P2.7 is one, display state of stop-watch on 7-seg

displayClock: ; displaying clock on multiplexed 7-seg display
  ; displaying s0 on disp0
  mov A, 30H
  setb P3.4
  setb P3.3
  acall displayDigit

  ; displaying s1 on disp1
  mov A, 31H
  clr P3.3
  acall displayDigit

  ; displaying m0 on disp2
  mov A, 32H
  setb P3.3
  clr P3.4
  acall displayDigit

  ; displaying m1 on disp3
  mov A, 33H
  clr P3.3
  acall displayDigit

  ret

displayStopWatch: ; displaying stop-watch on multiplexed 7-seg display
```

```
  ; displaying s0 on disp0
  mov A, 40H
  setb P3.4
  setb P3.3
  acall displayDigit

  ; displaying s1 on disp1
  mov A, 41H
  clr P3.3
  acall displayDigit

  ; displaying m0 on disp2
  mov A, 42H
  setb P3.3
  clr P3.4
  acall displayDigit

  ; displaying m1 on disp3
  mov A, 43H
  clr P3.3
  acall displayDigit

  ret

resetStopWatch:    ; reset stop-watch
  mov 40H, #00
  mov 41H, #00
  mov 42H, #00
  mov 43H, #00
  acall displayStopWatch

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/1mh0I7k6NpMX4XhxMPziE3SL3atcJevHS/view?usp=sharing to see the simulation of Part 2.

Discussion

- In this part, a hybrid clock supporting both the real-time clock and stop-watch functionalityies is implemented using edsim simulator.
- The implemetation of real-time clock is already discussed in Part 1 and that of stop-watch in Part 2. Here, the switching between the two modes and their integration is discussed.
- As can be observed from Part 1 and Part 2, both the real-time clock and stop-watch use some common modules like delay module and displayDigit module.
- The displayClock module and displayStopWatch module are also closely related with the differences in the RAM address for storing the state.
- P2.7 is used to switch from one mode to other.
- If P2.7 is set to logic low, clock mode is enabled (which is the default mode) and if P2.7 is set to logic high, stop-watch mode will be enabled.
- As the state of real-time clock has to be updated even if stop-watch mode is chosen, the changeClockState module is always called in the start module. Notice, this is not the case with changeStopWatchState module.
- When it comes to display the state of hybrid clock on 7-segment displays, we need to be sure in which mode the hybrid clock is operating.
- This is checked in X module for clock mode, and in Y module for stop-watch mode.
- Once the state of the hybrid clock is verified, display-digit module for that specific mode is called as observed in Part 1 and Part 2.