Microcontroller System Laboratory

Experiment 3

Utkarsh Patel 18EC35034

Part 1 – Displaying name on LCD display unit

Objective

Display all characters of your names (only name) on the LCD display unit

Circuit Diagram

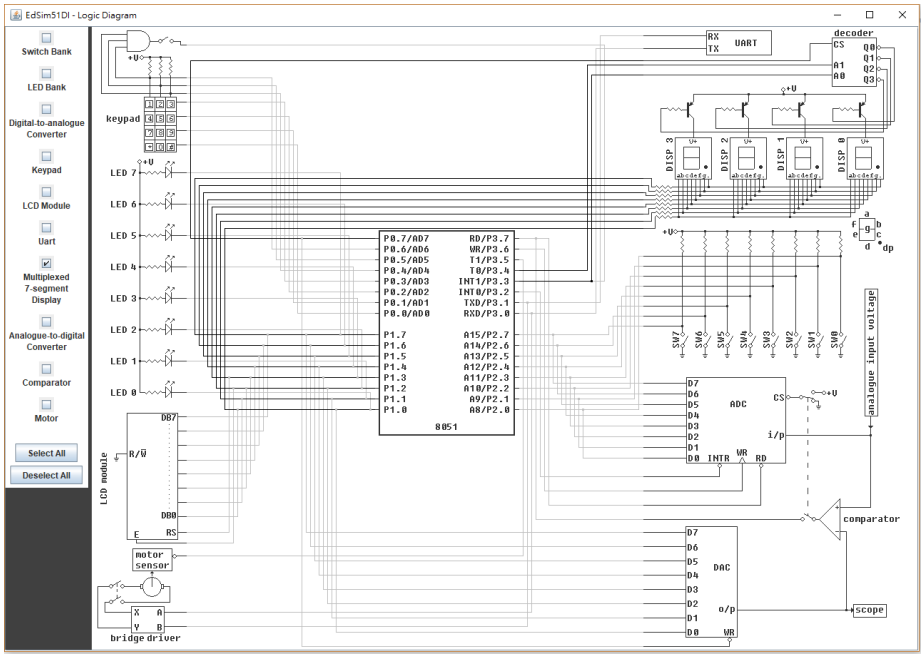

Fig. 1. Circuit diagram

Code

```
; Run this code with update frequency 100

org 0000H

MOV 30H, #'U'
MOV 31H, #'T'
MOV 32H, #'K'
MOV 33H, #'A'
MOV 34H, #'R'
MOV 35H, #'S'
MOV 36H, #'H'
MOV 37H, #0 ; end of data marker

init:
  call configureFor4BitOperation
```

```
  call incrementCursorMode
  call displayOnCursonOnBlinkingOn
  call main


configureFor4BitOperation:
  ; for configuring 4-bit operation in LCD

  clr P1.3     ; instruction flow mode

  clr P1.7     ; |
  clr P1.6     ; |
  setb P1.5    ; |
  clr P1.4     ; | high nibble set

  setb P1.2    ; |
  clr P1.2     ; | negative edge

  call delay

  setb P1.2    ; |
  clr P1.2     ; | negative edge

  setb P1.7

  setb P1.2    ; |
  clr P1.2     ; | negative edge

  call delay

  ret

incrementCursorMode:
  ; for displaying next character on adjacent display
  ; Code = 0x06 = 0000 0110

  clr P1.3   ; instruction mode on

  clr P1.7   ; |
  clr P1.6   ; |
  clr P1.5   ; |
  clr P1.4   ; | high nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  setb P1.6  ; |
  setb P1.5  ; | low nibble set
```

```
  setb P1.2  ; |
  clr P1.2   ; | negative edge

  call delay

  ret

displayOnCursonOnBlinkingOn:
  ; turning on the display and cursor and choosing blinking
  ; Code = 0x0F = 0000 1111

  clr P1.3   ; instruction mode on

  clr P1.7   ; |
  clr P1.6   ; |
  clr P1.5   ; |
  clr P1.4   ; | high nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  setb P1.7  ; |
  setb P1.6  ; |
  setb P1.5  ; |
  setb P1.4  ; | low nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  call delay

  ret


main:
  SETB P1.3     ; clear RS - indicates that data is being sent to module
  MOV R1, #30H  ; data to be sent to LCD is stored in 8051 RAM, starting at location 30H
  acall loop

loop:
  MOV A, @R1          ; move data pointed to by R1 to A
  JZ finish           ; if A is 0, then end of data has been reached - jump out of loop
  CALL sendCharacter  ; send data in A to LCD module
  INC R1              ; point to next piece of data
  JMP loop            ; repeat

finish:
```

```
  JMP $

sendCharacter:
  MOV C, ACC.7    ; |
  MOV P1.7, C     ; |
  MOV C, ACC.6    ; |
  MOV P1.6, C     ; |
  MOV C, ACC.5    ; |
  MOV P1.5, C     ; |
  MOV C, ACC.4    ; |
  MOV P1.4, C     ; | high nibble set

  SETB P1.2       ; |
  CLR P1.2        ; | negative edge on E

  MOV C, ACC.3    ; |
  MOV P1.7, C     ; |
  MOV C, ACC.2    ; |
  MOV P1.6, C     ; |
  MOV C, ACC.1    ; |
  MOV P1.5, C     ; |
  MOV C, ACC.0    ; |
  MOV P1.4, C     ; | low nibble set

  SETB P1.2       ; |
  CLR P1.2        ; | negative edge on E

  CALL delay      ; wait for BF to clear

delay:
  MOV R0, #50
  DJNZ R0, $
  RET
```

Discussion

- The LCD module consists of 16 rows and 2 columns of 5x8 dot matrices.
- Name of the pins and their corresponding functions are as follows:

Pin Name	Function
VSS	Must be grounded
VCC	5V DC power supply
RS	Register Selection
R/W	Read/write
E	Enable
DB[7:0]	Data

- From the circuit diagram, it can be observed that P1.3 is the RS of the LCD display, and P1.2 is the E of the LCD display.
- To execute any set of command, a negative edge has to be generated by E, i.e., set P1.2 to logic high and then to logic low and add some delay
- There are two types of register modes:
    - **Command mode**: Indicates flow of instruction to the LCD module, RS (P1.3) must be set to logic low to select this register mode
    - **Data mode**: Indicates flow of data to the LCD module, RS (P1.3) must be set to logic high to select this register mode
- Every operation linked with LCD display has a unique hexadecimal code. Some of them are:

Code (in hexadecimal)	Operation
0F	LCD ON, cursor ON, blinking ON
01	Clear screen
02	Return home
04	Decrement cursor
06	Increment cursor
0E	Display ON, cursor OFF
80	Force cursor to the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
38	Use 2 lines and 5x7 matrix
83	Cursor line 1 position 3
3C	Activate second line
08	Display OFF, cursor OFF
C1	Jump to second line, position 1
C2	Jump to second line, position 2
0C	Display ON, cursor OFF

- To perform any operation with code, say 0x75 = 01110101B, we divide the instruction into high nibble set (0111 in this case) and low nibble set (0101 in this case). Then we do:

```
CLR P1.7     ;  0|
SETB P1.6    ;  1|
SETB P1.5    ;  1|
SETB P1.4    ;  1| high nibble set
SETB P1.2    ;   |
CLR P1.2     ;   | negative edge on E
CLR P1.7     ;  0|
SETB P1.6    ;  1|
CLR P1.5     ;  0|
SETB P1.4    ;  1| low nibble set
SETB P1.2    ;   |
CLR P1.2     ;   | negative edge on E
CALL delay
```

- The characters of my name are stored in RAM in from 0x30 to 0x37
- sendCharacter module is used for displaying the current character on the LCD display

Part 2 – Real time clock on LCD display unit

Objective

Design a digital clock and display its output on LCD display unit. The clock should show the time in mm-ss format. It updates time automatically using the timer interrupt of the microcontroller.

Circuit Diagram

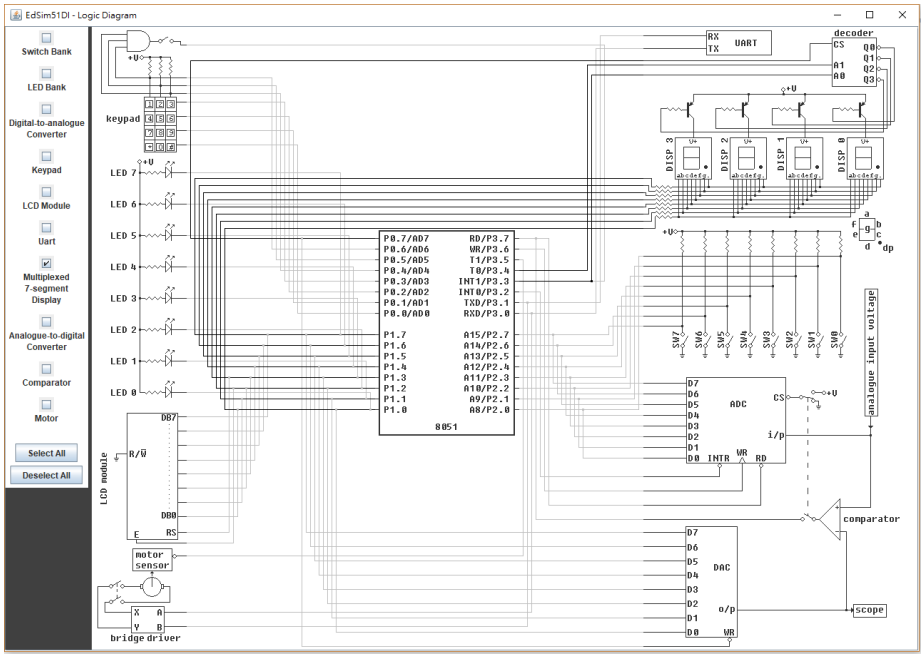

Fig. 1. Circuit diagram

Code

```
; Run this code with update frequency 100

org 0000H

mov 30H, #030H
mov 31H, #030H
mov 32H, #030H
mov 33H, #030H

init:
  call configureFor4BitOperation
  call incrementCursorMode
  call displayOnCursonOnBlinkingOn
  call main


configureFor4BitOperation:
  ; for configuring 4-bit operation in LCD

  clr P1.3     ; instruction flow mode
```

```
  clr P1.7     ; |
  clr P1.6     ; |
  setb P1.5    ; |
  clr P1.4     ; | high nibble set

  setb P1.2    ; |
  clr P1.2     ; | negative edge

  call delay

  setb P1.2    ; |
  clr P1.2     ; | negative edge

  setb P1.7

  setb P1.2    ; |
  clr P1.2     ; | negative edge

  call delay
  ret

incrementCursorMode:
  ; for displaying next character on adjacent display
  ; Code = 0x06 = 0000 0110

  clr P1.3   ; instruction mode on

  clr P1.7   ; |
  clr P1.6   ; |
  clr P1.5   ; |
  clr P1.4   ; | high nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  setb P1.6  ; |
  setb P1.5  ; | low nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge
  call delay
  ret

displayOnCursonOnBlinkingOn:
  ; turning on the display and cursor and choosing blinking
  ; Code = 0x0F = 0000 1111

  clr P1.3   ; instruction mode on
```

```
  clr P1.7   ; |
  clr P1.6   ; |
  clr P1.5   ; |
  clr P1.4   ; | high nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  setb P1.7  ; |
  setb P1.6  ; |
  setb P1.5  ; |
  setb P1.4  ; | low nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  call delay
  ret

main:
  call delayLarge
  call displayClock
  call delayLarge
  call updateClock
  jmp main

displayClock:
  ; display clock on the LCD display

  setb P1.3 ; data mode on
  mov A, 30H
  call sendCharacter
  mov A, 31H
  call sendCharacter
  mov A, 32H
  call sendCharacter
  mov A, 33H
  call sendCharacter
  ret

updateClock:
  ; updating value of the clock

  inc 33H
  mov A, 33H
  cjne A, #03AH, finishUpdate
  mov 33H, #030H
```

```
  inc 32H
  mov A, 32H
  cjne A, #036H, finishUpdate
  mov 32H, #030H

  inc 31H
  mov A, 31H
  cjne A, #03AH, finishUpdate
  mov 31H, #030H

  inc 30H
  mov A, 30H
  cjne A, #036H, finishUpdate
  mov 30H, #030H
  ret

finishUpdate:
  ; finsh updating clock

  call delay
  jmp resetDisplay

resetDisplay:
  ; reset LCD display
  ; Code = 0x01 = 0000 0001

  clr P1.3   ; instruction mode on

  clr P1.7   ; |
  clr P1.6   ; |
  clr P1.5   ; |
  clr P1.4   ; | high nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  clr P1.7   ; |
  clr P1.6   ; |
  clr P1.5   ; |
  setb P1.4  ; | low nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  call delay
  ret
```

```
sendCharacter:
  ; diplsay current time on LCD display

  mov C, ACC.7
  mov P1.7, C    ; |
  mov C, ACC.6
  mov P1.6, C    ; |
  mov C, ACC.5
  mov P1.5, C    ; |
  mov C, ACC.4
  mov P1.4, C    ; | high nibble set

  setb P1.2      ; |
  clr P1.2       ; | negative edge

  mov C, ACC.3
  mov P1.7, C    ; |
  mov C, ACC.2
  mov P1.6, C    ; |
  mov C, ACC.1
  mov P1.5, C    ; |
  mov C, ACC.0
  mov P1.4, C    ; | low nibble set

  setb P1.2      ; |
  clr P1.2       ; | negative edge

  call delay
  ret

delay:
  ; delay for internal LCD operation
  mov R0, #50
  DJNZ R0, $
  ret

delayLarge:
  ; delay for displaying
  mov R0, #150
  DJNZ R0, $
  ret
```

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 delay module is then used in sendCharacter module, which is used to display current digit in the LCD display unit.
- The sendCharacter module is then used in displayClock module for displaying the current clock state on LCD.
- The updateClock module updates the state of the clock.
- The resetDisplay module is used to rest the LCD display so as to make room for the next clock state.
- The LCD module consists of 16 rows and 2 columns of 5x8 dot matrices.
- Name of the pins and their corresponding functions are as follows:

Pin Name	Function
VSS	Must be grounded
VCC	5V DC power supply
RS	Register Selection
R/W	Read/write
E	Enable
DB[7:0]	Data

- From the circuit diagram, it can be observed that P1.3 is the RS of the LCD display, and P1.2 is the E of the LCD display.
- To execute any set of command, a negative edge has to be generated by E, i.e., set P1.2 to logic high and then to logic low and add some delay
- There are two types of register modes:
    - **Command mode**: Indicates flow of instruction to the LCD module, RS (P1.3) must be set to logic low to select this register mode
    - **Data mode**: Indicates flow of data to the LCD module, RS (P1.3) must be set to logic high to select this register mode
- Every operation linked with LCD display has a unique hexadecimal code. Some of them are:

Code (in hexadecimal)	Operation
0F	LCD ON, cursor ON, blinking ON
01	Clear screen
02	Return home
04	Decrement cursor
06	Increment cursor
0E	Display ON, cursor OFF
80	Force cursor to the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
38	Use 2 lines and 5x7 matrix
83	Cursor line 1 position 3
3C	Activate second line
08	Display OFF, cursor OFF
C1	Jump to second line, position 1
C2	Jump to second line, position 2
0C	Display ON, cursor OFF

- To perform any operation with code, say 0x75 = 01110101B, we divide the instruction into high nibble set (0111 in this case) and low nibble set (0101 in this case). Then we do:

```
CLR P1.7     ;  0|
SETB P1.6    ;  1|
SETB P1.5    ;  1|
SETB P1.4    ;  1| high nibble set

SETB P1.2    ;   |
CLR P1.2     ;   | negative edge on E

CLR P1.7     ;  0|
SETB P1.6    ;  1|
CLR P1.5     ;  0|
SETB P1.4    ;  1| low nibble set

SETB P1.2    ;   |
CLR P1.2     ;   | negative edge on E

CALL delay
```

Part 3 – Display your name and *Real-time clock output* on display unit

Objective

In normal mode, display your name. On pressing the mode switch, the display to clock mode On pressing the mode button once more, the display returns to normal mode (i.e., display your name). It should be noted that the real time clock should get updated during the normal mode.

Circuit Diagram

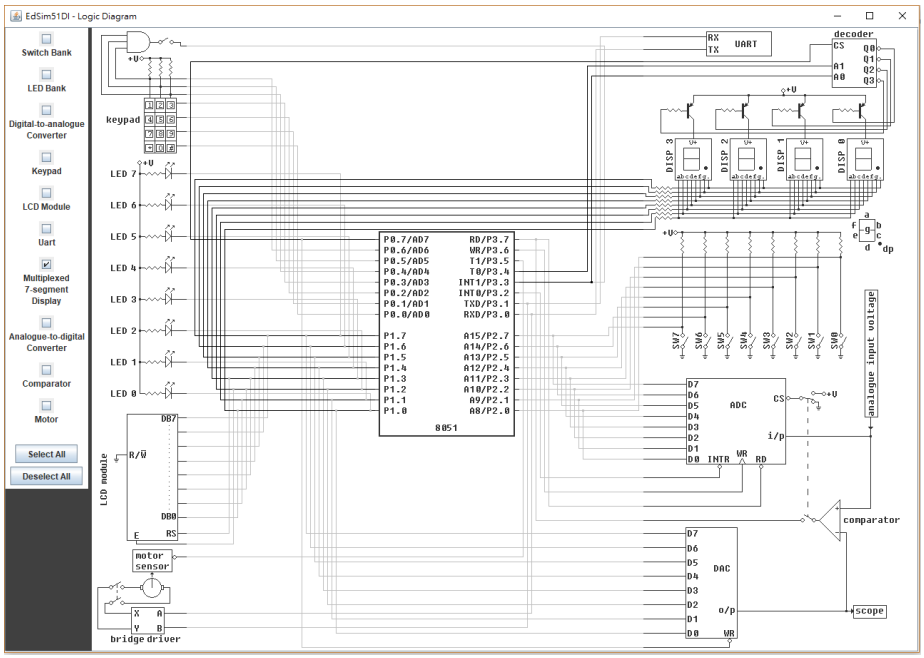

Fig. 1. Circuit diagram

Code

```
; Instruction
; * Displaying name is the default mode
; * Press P2.7 to switch to real-time clock mode
; * Un-press P2.7 to switch to display name mode
;
; Run this code with update frequency 100

org 0000H

mov 70H, #030H
mov 71H, #030H
mov 72H, #030H
mov 73H, #030H

MOV 30H, #'U'
MOV 31H, #'T'
MOV 32H, #'K'
MOV 33H, #'A'
MOV 34H, #'R'
MOV 35H, #'S'
```

```
MOV 36H, #'H'

init:
  call configureFor4BitOperation
  call incrementCursorMode
  call displayOnCursonOnBlinkingOn
  call main


configureFor4BitOperation:
  ; for configuring 4-bit operation in LCD

  clr P1.3     ; instruction flow mode

  clr P1.7     ; |
  clr P1.6     ; |
  setb P1.5    ; |
  clr P1.4     ; | high nibble set

  setb P1.2    ; |
  clr P1.2     ; | negative edge

  call delay

  setb P1.2    ; |
  clr P1.2     ; | negative edge

  setb P1.7

  setb P1.2    ; |
  clr P1.2     ; | negative edge

  call delay

  ret

incrementCursorMode:
  ; for displaying next character on adjacent display
  ; Code = 0x06 = 0000 0110

  clr P1.3   ; instruction mode on

  clr P1.7   ; |
  clr P1.6   ; |
  clr P1.5   ; |
  clr P1.4   ; | high nibble set

  setb P1.2  ; |
```

```
  clr P1.2   ; | negative edge

  setb P1.6  ; |
  setb P1.5  ; | low nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  call delay

  ret

displayOnCursonOnBlinkingOn:
  ; turning on the display and cursor and choosing blinking
  ; Code = 0x0F = 0000 1111

  clr P1.3   ; instruction mode on

  clr P1.7   ; |
  clr P1.6   ; |
  clr P1.5   ; |
  clr P1.4   ; | high nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  setb P1.7  ; |
  setb P1.6  ; |
  setb P1.5  ; |
  setb P1.4  ; | low nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  call delay

  ret


main:
        jnb P2.7, displayClock  ; if P2.7 is 1, diplay clock
  jb P2.7, displayName    ; else diplay name

main2:
  call updateClock
  jmp main

displayClock:
```

```
  ; display clock on LCD

  setb P1.3   ; data mode on
  mov A, 70H
  call sendCharacter
  mov A, 71H
  call sendCharacter
  mov A, 72H
  call sendCharacter
  mov A, 73H
  call sendCharacter
  call delayLarge
  call resetDisplay
  jmp main2

displayName:
  ; display name on LCD

  setb P1.3   ; data mode on
  mov A, 30H
  call sendCharacter
  mov A, 31H
  call sendCharacter
  mov A, 32H
  call sendCharacter
  mov A, 33H
  call sendCharacter
  mov A, 34H
  call sendCharacter
  mov A, 35H
  call sendCharacter
  mov A, 36H
  call sendCharacter
  call delayLarge
  call resetDisplay
  jmp main2

loop:
  MOV A, @R1          ; move data pointed to by R1 to A
  JZ finish           ; if A is 0, then end of data has been reached - jump out of loop
  CALL sendCharacter  ; send data in A to LCD module
  INC R1              ; point to next piece of data
  JMP loop            ; repeat

finish:
  jmp main2

updateClock:
```

```
  ; update clock state

  inc 73H
  mov A, 73H
  cjne A, #03AH, finishUpdate
  mov 73H, #030H

  inc 72H
  mov A, 72H
  cjne A, #036H, finishUpdate
  mov 72H, #030H

  inc 71H
  mov A, 71H
  cjne A, #03AH, finishUpdate
  mov 71H, #030H

  inc 70H
  mov A, 70H
  cjne A, #036H, finishUpdate
  mov 70H, #030H
  call finishUpdate

finishUpdate:
  call delayLarge
  jmp main

resetDisplay:
  ; reset LCD display
  ; Code = 0x01 = 0000 0001

  clr P1.3   ; instruction mode on

  clr P1.7   ; |
  clr P1.6   ; |
  clr P1.5   ; |
  clr P1.4   ; | high nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge

  clr P1.7   ; |
  clr P1.6   ; |
  clr P1.5   ; |
  setb P1.4  ; | low nibble set

  setb P1.2  ; |
  clr P1.2   ; | negative edge
```

```
  call delay
  ret

sendCharacter:
  ; display current character on the display

  mov C, ACC.7
  mov P1.7, C    ; |
  mov C, ACC.6
  mov P1.6, C    ; |
  mov C, ACC.5
  mov P1.5, C    ; |
  mov C, ACC.4
  mov P1.4, C    ; | high nibble set

  setb P1.2      ; |
  clr P1.2       ; | negative edge

  mov C, ACC.3
  mov P1.7, C    ; |
  mov C, ACC.2
  mov P1.6, C    ; |
  mov C, ACC.1
  mov P1.5, C    ; |
  mov C, ACC.0
  mov P1.4, C    ; | low nibble set

  setb P1.2      ; |
  clr P1.2       ; | negative edge

  call delay
  ret

delay:
  mov R0, #50
  DJNZ R0, $
  ret

delayLarge:
  mov R0, #150
  DJNZ R0, $
  ret
```

Discussion

- In this part, a hybrid display supporting displaying name and displaying a real time clock on LCD display is implemented in edsim simulator.
- P2.7 is used to switch from one mode to other.
- If P2.7 is set to logic low, display name mode is enabled (which is the default mode) and if P2.7 is set to logic high, real-time clock mode will be enabled.
- As the state of real-time clock has to be updated even if display name mode is chosen, the updateClock module is always called in the main module.
- The code starts with assigning temporary locations to variable needed to simulate the clock and storing the characters of the name.
- The initial state of the clock is set to 00:00, which can be modified by changing the initialization in the code.
- The delay module is then used in sendCharacter module, which is used to display current digit in the LCD display unit.
- The sendCharacter module is then used in displayClock module for displaying the current clock state on LCD.
- The updateClock module updates the state of the clock and the resetDisplay module is used to rest the LCD display so as to make room for the next clock state.
- The LCD module consists of 16 rows and 2 columns of 5x8 dot matrices.
- Name of the pins and their corresponding functions are as follows:

Pin Name	Function
VSS	Must be grounded
VCC	5V DC power supply
RS	Register Selection
R/W	Read/write
E	Enable
DB[7:0]	Data

- From the circuit diagram, it can be observed that P1.3 is the RS of the LCD display, and P1.2 is the E of the LCD display.
- To execute any set of command, a negative edge has to be generated by E, i.e., set P1.2 to logic high and then to logic low and add some delay
- There are two types of register modes:
    - **Command mode**: Indicates flow of instruction to the LCD module, RS (P1.3) must be set to logic low to select this register mode
    - **Data mode**: Indicates flow of data to the LCD module, RS (P1.3) must be set to logic high to select this register mode
- Every operation linked with LCD display has a unique hexadecimal code. Some of them are:

Code (in hexadecimal)	Operation
0F	LCD ON, cursor ON, blinking ON
01	Clear screen
02	Return home
04	Decrement cursor
06	Increment cursor
0E	Display ON, cursor OFF
80	Force cursor to the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
38	Use 2 lines and 5x7 matrix
83	Cursor line 1 position 3
3C	Activate second line
08	Display OFF, cursor OFF
C1	Jump to second line, position 1
C2	Jump to second line, position 2
0C	Display ON, cursor OFF

- To perform any operation with code, say 0x75 = 01110101B, we divide the instruction into high nibble set (0111 in this case) and low nibble set (0101 in this case). Then we do:

```
CLR P1.7     ;  0|
SETB P1.6    ;  1|
SETB P1.5    ;  1|
SETB P1.4    ;  1| high nibble set

SETB P1.2    ;   |
CLR P1.2     ;   | negative edge on E

CLR P1.7     ;  0|
SETB P1.6    ;  1|
CLR P1.5     ;  0|
SETB P1.4    ;  1| low nibble set

SETB P1.2    ;   |
CLR P1.2     ;   | negative edge on E

CALL delay
```