ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
ORGANISATION OF ISLAMIC COOPERATION (OIC)
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 4705-Microcontroller Based System Design
COMPLEX ENGINEERING PROBLEM

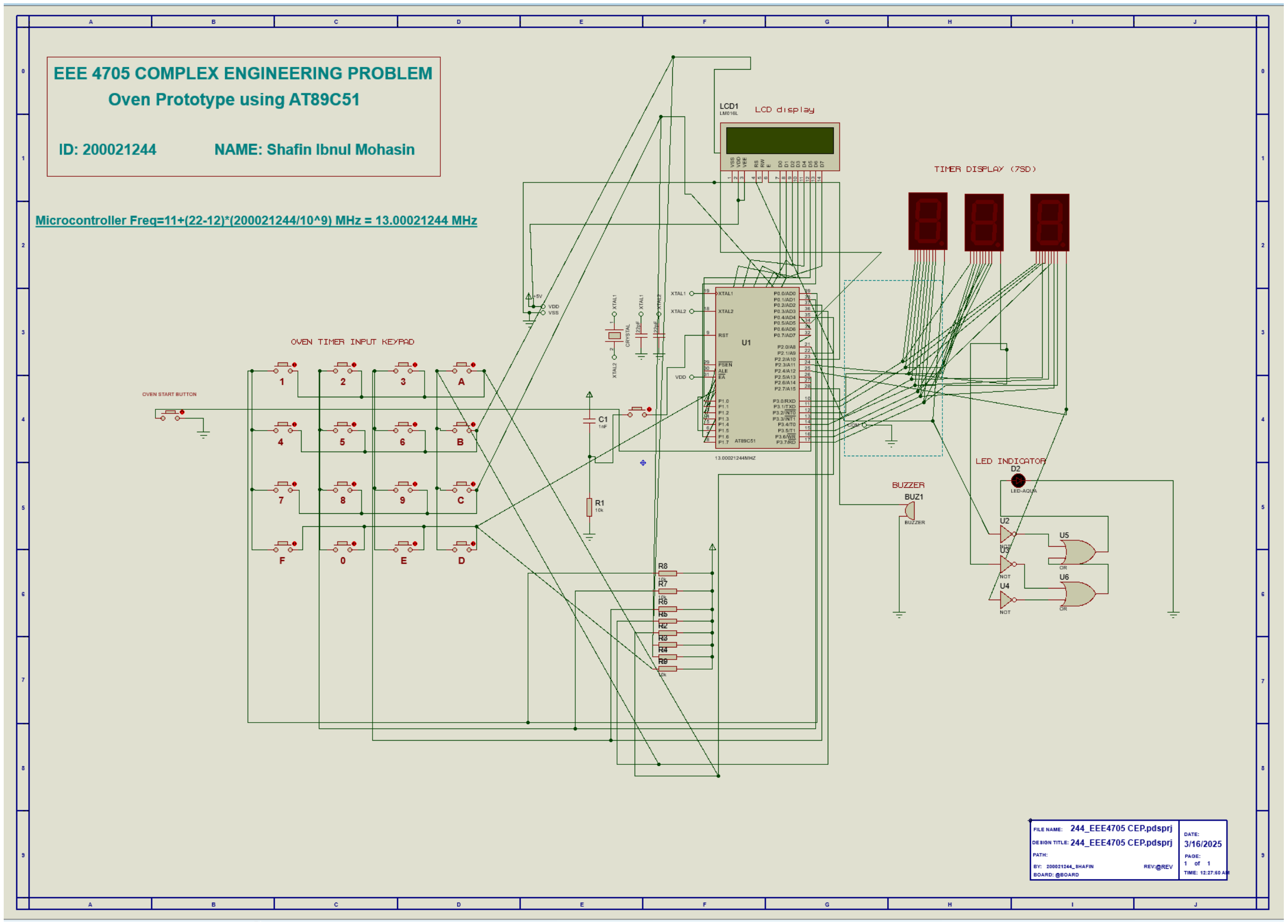
Instructor Name: **Md. Arif Hossain.**
Assistant Professor
Dept. Of EEE, IUT.

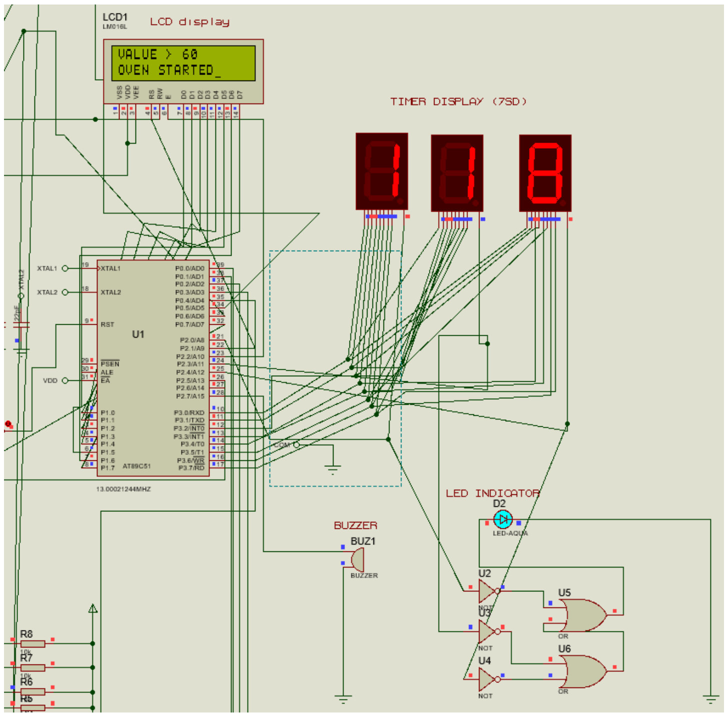
NAME	ID
Shafin Ibnul Mohasin **Sec. B** **Dept.: EEE**	**200021244**

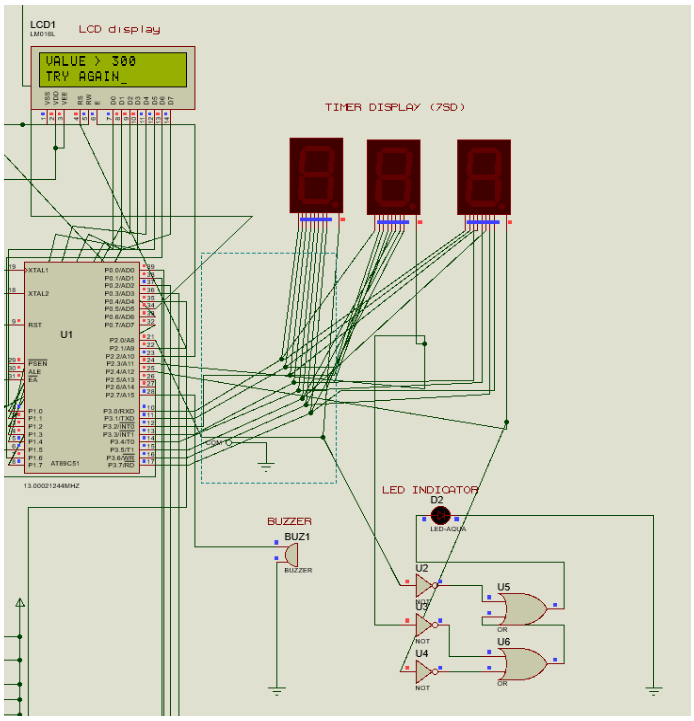
***Submission Date*: 18/03/2025.**

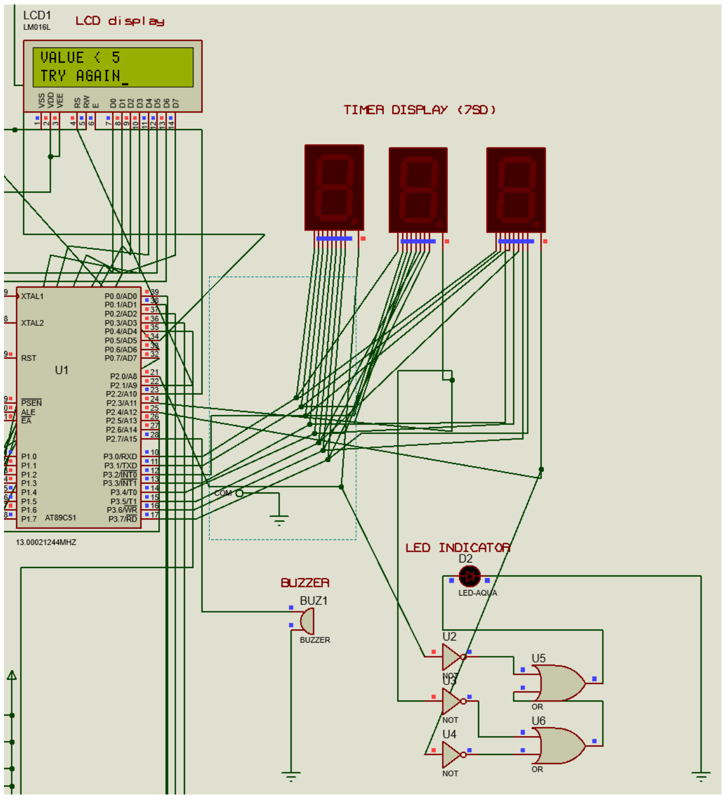
Microcontroller-Based Interactive Oven Control System with Real-Time Timer Operations and Emergency Features Using AT89C51

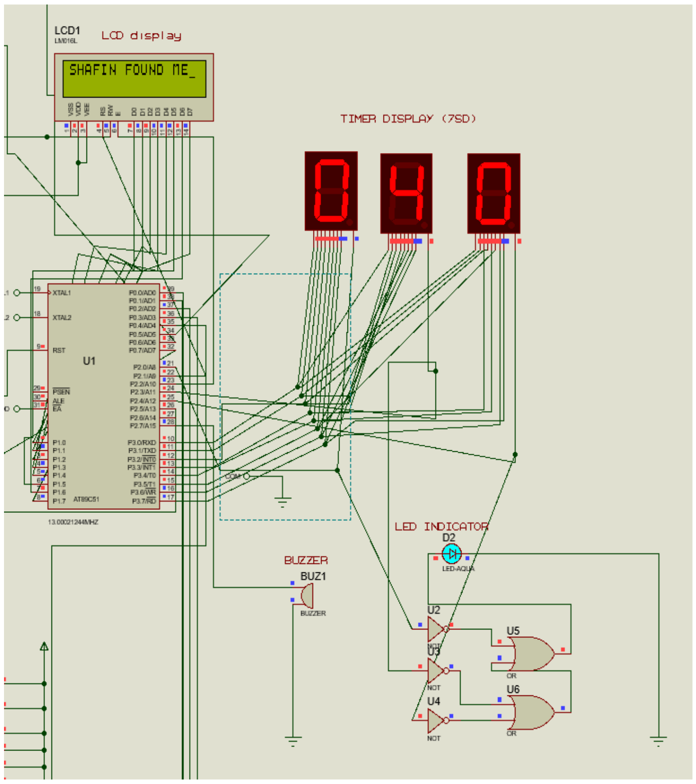
<u>Proteus Layout & Simulation:</u>




LCD1
LM016L
LCD display
VALUE > 60
OVEN STARTED_
TIMER DISPLAY (7SD)
U1
AT89C51
LED INDICATOR
D2
LED-AQUA
BUZZER
BUZ1
U2
U3
U4
U5
U6
NOT
OR
R8
R7
R6
R5


LCD1
LM016L
LCD display
VALUE > 300
TRY AGAIN_
TIMER DISPLAY (7SD)
U1
AT89C51
LED INDICATOR
D2
LED-AQUA
BUZZER
BUZ1
U2
U3
U4
U5
U6
NOT
OR


LCD1
LM016L
LCD display
VALUE < 5
TRY AGAIN_
TIMER DISPLAY (7SD)
U1
AT89C51
LED INDICATOR
D2
LED-AQUA
BUZZER
BUZ1
U2
U3
U4
U5
U6
NOT
OR


LCD1
LM016L
LCD display
SHAFIN FOUND ME_
TIMER DISPLAY (7SD)
U1
AT89C51
LED INDICATOR
D2
LED-AQUA
BUZZER
BUZ1
U2
U3
U4
U5
U6
NOT
OR

Code:

```
;NAME: Shafin Ibnul Mohasin ID: 200021244
      org   0000h

INIT_ALL:        MOV      P3,#00000000B    ; Clear port 3
        MOV P0, #0FEH               ; Initialize port 0
        MOV 30H,#0                 ; Reset memory location 30H
        MOV 32H,#0                 ; Reset memory location 32H
        MOV R0,#0                  ; Clear register R0
        MOV R7, #15                ; Set R7 to 15
        mov r5,#00H                ; Clear register R5
        MOV 69H,0H                 ; Clear memory location 69H
        CLR P2.7                   ; Clear buzzer pin
        MOV P1, #00000000B         ; Clear port 1

REGISTERS_SETUP:
MOV R3, #00H                ; Clear register R3
MOV R1, #00H                ; Clear register R1
MOV R2, #00H                ; Clear register R2


PORT_SETUP:
RS EQU P2.1                 ; Define RS pin for LCD
EN EQU P2.2                 ; Define EN pin for LCD


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SETUP_LCD:
MOV R3, #38H                ; Set 8-bit mode, 2 lines
ACALL SEND_CMD               ; Send command to LCD
MOV R3, #0EH                ; Display on, cursor on
ACALL SEND_CMD
MOV R3, #80H                ; Set cursor to first line
ACALL SEND_CMD
MOV R3, #01H                ; Clear display
ACALL SEND_CMD


MAIN_SCAN:       LCALL SCAN      ; Scan keypad
        MOV A,R0                   ; Move key value to A
        JZ MAIN_SCAN               ; If zero, keep scanning

        MOV 40H,A                  ; Store first digit

        lcall WAIT_KEYRELEASE      ; Delay for key debounce
MAIN_SCAN2:      LCALL SCAN      ; Scan keypad for second digit
        MOV A,R0                   ; Move key value to A
        JZ MAIN_SCAN2              ; If zero, keep scanning

        MOV 44H,A                  ; Store second digit

        lcall WAIT_KEYRELEASE      ; Delay for key debounce
MAIN_SCAN3:      LCALL SCAN      ; Scan keypad for third digit
        MOV A,R0                   ; Move key value to A
        JZ MAIN_SCAN3              ; If zero, keep scanning

        MOV 53H,A                  ; Store third digit


WAIT_START:      JB P2.5, WAIT_START      ; Wait until start button pressed
        ANL 53H,#00001111B         ; Mask upper nibble (keep only lower 4 bits)
        ANL 40H,#00001111B         ; Mask upper nibble
        ANL 44H,#00001111B         ; Mask upper nibble

        MOV A,44H                  ; Get second digit
        MOV B,A                    ; Store in B
        MOV A,#10                  ; Multiply by 10
        MUL AB                     ; Perform multiplication
        ADD A,53H                  ; Add third digit
        MOV 60H,A                  ; Store result in 60H


        MOV A,40H                  ; Get first digit
        MOV B,A                    ; Move to B
        MOV A,#100                 ; Multiply by 100
        MUL AB                     ; Perform multiplication
        MOV 62H,A                  ; Store lower byte
        MOV A,B                    ; Get upper byte
        MOV 61H,A                  ; Store upper byte
        MOV A,62H                  ; Get lower byte
        ADD A,60H                  ; Add previous result
        MOV 62H,A                  ; Store new result
        JNC THRESHOLD_CHECK        ; Check if no carry
        INC 61H                    ; Increment upper byte if carry

```

```
THRESHOLD_CHECK:  MOV A,61H       ; Get high byte
        CJNE A,#01H,THRESHOLD_CHECK1  ; Compare with 01H (300 high byte)
        MOV A,62H                     ; Get low byte
        CJNE A,#2dH,THRESHOLD_CHECK2  ; Compare with 2DH (300 low byte)
        JMP GREATER_300               ; Exactly 300, treat as > 300

THRESHOLD_CHECK1: JC  THRESHOLD_CHECK5 ; If high byte < 01H, number is < 300
        JMP GREATER_300                ; If high byte > 01H, number is > 300

THRESHOLD_CHECK2: JC THRESHOLD_CHECK5  ; If low byte < 2DH, number is < 300
        JMP GREATER_300                ; If low byte > 2DH, number is > 300

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
THRESHOLD_CHECK5:  MOV A,61H           ; Get high byte
        JNZ THRESHOLD_60               ; If high byte > 0, number is > 5
        MOV A,62H                      ; Get low byte
        CJNE A,#05H,CHECK_5_TEMP       ; Compare with 5
        JMP THRESHOLD_60               ; Exactly 5, check next threshold

CHECK_5_TEMP: JC BELOW_5               ; If < 5, handle separately
        JMP THRESHOLD_60               ; If > 5, check next threshold

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



BELOW_5:  MOV DPTR,#BELOW_5_TEXT       ; Load message address
BELOW_5_LOOP:MOV A,#00H                ; Clear A
         MOVC A,@A+DPTR                    ; Get character
         JZ TMP_LOOP                       ; If zero, end of string
         MOV R3,A                          ; Move to R3
         ACALL DISPLAY_CHAR                ; Display character
         INC DPTR                          ; Next character
         LJMP BELOW_5_LOOP                 ; Continue


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

TMP_LOOP  : MOV R3, #0C0H              ; Set cursor to second line
         ACALL SEND_CMD                    ; Send command
         MOV DPTR,#RETRY_TEXT              ; Load retry message
RETRY_LOOP:MOV A,#00H                  ; Clear A
         MOVC A,@A+DPTR                    ; Get character
         JZ RETRY_WAIT                     ; If zero, end of string
         MOV R3,A                          ; Move to R3
         ACALL DISPLAY_CHAR                ; Display character
         INC DPTR                          ; Next character
         LJMP RETRY_LOOP                   ; Continue

RETRY_WAIT:      LCALL LONG_DELAY           ; Wait for a while
         Ljmp INIT_ALL                     ; Restart program


THRESHOLD_60:  MOV A,61H                ; Get high byte
        JNZ  ABOVE_60                   ; If high byte > 0, number is > 60
        MOV A,62H                       ; Get low byte
        CJNE A,#3CH,CHECK_60_TEMP       ; Compare with 60 (3Ch)
        JMP ABOVE_60                    ; Exactly 60, treat as > 60

CHECK_60_TEMP: JC BELOW_60              ; If < 60, handle separately
        JMP ABOVE_60                    ; If > 60, handle accordingly



ABOVE_60 : MOV DPTR,#ABOVE_60_TEXT      ; Load message address
ABOVE_60_LOOP:MOV A,#00H                ; Clear A
         MOVC A,@A+DPTR                     ; Get character
         JZ OVEN_START_2                    ; If zero, end of string
         MOV R3,A                           ; Move to R3
         ACALL DISPLAY_CHAR                 ; Display character
         INC DPTR                           ; Next character
         LJMP ABOVE_60_LOOP                 ; Continue


BELOW_60 : MOV DPTR,#BELOW_60_TEXT      ; Load message address
BELOW_60_LOOP:MOV A,#00H                ; Clear A
         MOVC A,@A+DPTR                     ; Get character
         JZ OVEN_START_1                    ; If zero, end of string
         MOV R3,A                           ; Move to R3
         ACALL DISPLAY_CHAR                 ; Display character
         INC DPTR                           ; Next character
         LJMP BELOW_60_LOOP                 ; Continue


GREATER_300 : MOV DPTR,#GREATER_300_TEXT ; Load message address
GREATER_300_LOOP:MOV A,#00H             ; Clear A
         MOVC A,@A+DPTR                     ; Get character
         JZ TMP_LOOP                        ; If zero, goto retry
         MOV R3,A                           ; Move to R3
         ACALL DISPLAY_CHAR                 ; Display character
         INC DPTR                           ; Next character
         LJMP GREATER_300_LOOP                   ; Continue


OVEN_START_2:MOV DPTR,#OVEN_START_TEXT  ; Load oven message
         MOV R3, #0C0H                           ; Set cursor to second line
         ACALL SEND_CMD                          ; Send command
OVEN_START_LOOP2:MOV A,#00H             ; Clear A
         MOVC A,@A+DPTR                          ; Get character
         JZ TIMER_LOOP2                          ; If zero, start timer
         MOV R3,A                                ; Move to R3
         ACALL DISPLAY_CHAR                      ; Display character
         INC DPTR                                ; Next character
         LJMP OVEN_START_LOOP2                   ; Continue


OVEN_START_1:MOV DPTR,#OVEN_START_TEXT  ; Load oven message
         MOV R3, #0C0H                           ; Set cursor to second line
         ACALL SEND_CMD                          ; Send command
OVEN_START_LOOP:MOV A,#00H              ; Clear A
         MOVC A,@A+DPTR                          ; Get character
         JZ TIMER_LOOP                           ; If zero, start timer
         MOV R3,A                                ; Move to R3
         ACALL DISPLAY_CHAR                      ; Display character
         INC DPTR                                ; Next character
         LJMP OVEN_START_LOOP                    ; Continue

TIMER_LOOP2:MOV  R6,#20                ; Set counter to 5
;LCALL DISPLAY_FACT1                    ; Display fact (commented out)
TIMER_LOOP2_TEMP:
         LCALL DELAY_ONE_SEC                     ; Wait one second
         LCALL DECREMENT_NUMBER                  ; Decrement the timer
         DJNZ R6,TIMER_LOOP2_TEMP                ; Loop until R6 is zero
         LCALL DISPLAY_RANDOM_FACT               ; Display random fact
         MOV     R6,#20                            ; Reset counter
        SJMP TIMER_LOOP2_TEMP                  ; Continue timer loop


TIMER_LOOP: LCALL LONG_DELAY            ; Wait for a while
LCALL DISPLAY_MY_FACT                   ; Display my fact
TIMER_LOOP_TEMP:LCALL DELAY_ONE_SEC     ; Wait one second
         LCALL DECREMENT_NUMBER                  ; Decrement the timer

        SJMP TIMER_LOOP_TEMP                     ; Continue timer loop



DISPLAY_MY_FACT:
MOV DPTR,#MY_FACT_TEXT                  ; Load fact message
         MOV R3, #01H                            ; Clear display and home cursor
         ACALL SEND_CMD                          ; Send command
DISPLAY_MY_FACT_LOOP:MOV A,#00H         ; Clear A
         MOVC A,@A+DPTR                          ; Get character
         JZ DISPLAY_MY_FACT_END                  ; If zero, end of string
         MOV R3,A                                ; Move to R3
         ACALL DISPLAY_CHAR                      ; Display character
         INC DPTR                                ; Next character
         LJMP DISPLAY_MY_FACT_LOOP               ; Continue
DISPLAY_MY_FACT_END:
RET                                     ; Return



DISPLAY_FACT1:
MOV DPTR,#FACT1_TEXT                    ; Load fact 1 message
         MOV R3, #01H                            ; Clear display and home cursor
         ACALL SEND_CMD                          ; Send command
DISPLAY_FACT1_LOOP:MOV A,#00H           ; Clear A
         MOVC A,@A+DPTR                          ; Get character
         JZ DISPLAY_FACT1_END                    ; If zero, end of string
         MOV R3,A                                ; Move to R3
         ACALL DISPLAY_CHAR                      ; Display character
         INC DPTR                                ; Next character
         LJMP DISPLAY_FACT1_LOOP                 ; Continue
DISPLAY_FACT1_END:
RET                                     ; Return

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;-----------------------------------------------------------------
DISPLAY_RANDOM_FACT:

INC R5                                  ; Increment fact counter
         CJNE R5,#01H,DISPLAY_FACT2              ; Check if fact 1
         MOV DPTR,#FACT2_TEXT                    ; Load fact 2 message
         MOV R3, #01H                            ; Clear display and home cursor
         ACALL SEND_CMD                          ; Send command
DISPLAY_FACT2_LOOP:MOV A,#00H           ; Clear A
         MOVC A,@A+DPTR                          ; Get character
         JZ DISPLAY_RANDOM_FACT_END              ; If zero, end of string
         MOV R3,A                                ; Move to R3
         ACALL DISPLAY_CHAR                      ; Display character
         INC DPTR                                ; Next character
         LJMP DISPLAY_FACT2_LOOP             ; Continue

DISPLAY_FACT2:
         CJNE R5,#02H,DISPLAY_FACT3          ; Check if fact 2
         MOV DPTR,#FACT3_TEXT                ; Load fact 3 message
         MOV R3, #01H                        ; Clear display and home cursor
         ACALL SEND_CMD                      ; Send command
DISPLAY_FACT3_LOOP:MOV A,#00H           ; Clear A
         MOVC A,@A+DPTR                      ; Get character
         JZ DISPLAY_RANDOM_FACT_END          ; If zero, end of string
         MOV R3,A                            ; Move to R3
         ACALL DISPLAY_CHAR                  ; Display character
         INC DPTR                            ; Next character
         LJMP DISPLAY_FACT3_LOOP             ; Continue


DISPLAY_FACT3:
         CJNE R5,#03H,DISPLAY_FACT4          ; Check if fact 3
         MOV DPTR,#FACT4_TEXT                ; Load fact 4 message
         MOV R3, #01H                        ; Clear display and home cursor
         ACALL SEND_CMD                      ; Send command
DISPLAY_FACT4_LOOP:MOV A,#00H           ; Clear A
         MOVC A,@A+DPTR                      ; Get character
         JZ DISPLAY_RANDOM_FACT_END          ; If zero, end of string
         MOV R3,A                            ; Move to R3
         ACALL DISPLAY_CHAR                  ; Display character
         INC DPTR                            ; Next character
         LJMP DISPLAY_FACT4_LOOP             ; Continue


DISPLAY_FACT4:
         CJNE R5,#04H,DISPLAY_FACT5          ; Check if fact 4
         MOV DPTR,#FACT5_TEXT                ; Load fact 5 message
         MOV R3, #01H                        ; Clear display and home cursor
         ACALL SEND_CMD                      ; Send command
DISPLAY_FACT5_LOOP:MOV A,#00H           ; Clear A
         MOVC A,@A+DPTR                      ; Get character
         JZ DISPLAY_RANDOM_FACT_END          ; If zero, end of string
         MOV R3,A                            ; Move to R3
         ACALL DISPLAY_CHAR                  ; Display character
         INC DPTR                            ; Next character
         LJMP DISPLAY_FACT5_LOOP             ; Continue


DISPLAY_FACT5:
         CJNE R5,#05H,DISPLAY_FACT6          ; Check if fact 5
         MOV DPTR,#FACT6_TEXT                ; Load fact 6 message
         MOV R3, #01H                        ; Clear display and home cursor
         ACALL SEND_CMD                      ; Send command
DISPLAY_FACT6_LOOP:MOV A,#00H           ; Clear A
         MOVC A,@A+DPTR                      ; Get character
         JZ DISPLAY_RANDOM_FACT_END          ; If zero, end of string
         MOV R3,A                            ; Move to R3
         ACALL DISPLAY_CHAR                  ; Display character
         INC DPTR                            ; Next character
         LJMP DISPLAY_FACT6_LOOP             ; Continue


DISPLAY_FACT6:
         CJNE R5,#06H,RESET_FACT_COUNTER     ; Check if fact 6
         MOV DPTR,#FACT7_TEXT                ; Load fact 7 message
         MOV R3, #01H                        ; Clear display and home cursor
         ACALL SEND_CMD                      ; Send command
DISPLAY_FACT7_LOOP:MOV A,#00H           ; Clear A
         MOVC A,@A+DPTR                      ; Get character
         JZ DISPLAY_RANDOM_FACT_END          ; If zero, end of string
         MOV R3,A                            ; Move to R3
         ACALL DISPLAY_CHAR                  ; Display character
         INC DPTR                            ; Next character
         LJMP DISPLAY_FACT7_LOOP             ; Continue


RESET_FACT_COUNTER: MOV R5,#0H          ; Reset fact counter to 0
         LJMP DISPLAY_RANDOM_FACT            ; Go back to display first fact


DISPLAY_RANDOM_FACT_END:
RET                                          ; Return from subroutine
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DECREMENT_NUMBER:
       DEC 53H                          ; Decrement units digit
        MOV A, 53H                      ; Move to accumulator

        CJNE A, #11111111B, CONTINUE_TIMER  ; Check if underflow

        ; Reset units digit and decrement tens digit
        MOV 53H, #9                     ; Reset to 9
        DEC 44H                         ; Decrement tens digit
        MOV A, 44H                      ; Move to accumulator

        CJNE A, #11111111B, CONTINUE_TIMER  ; Check if underflow

        ; Reset tens digit and decrement hundreds digit
        MOV 44H, #9                     ; Reset to 9
        DEC 40H                         ; Decrement hundreds digit
        MOV A, 40H                      ; Move to accumulator

        CJNE A, #11111111B, CONTINUE_TIMER  ; Check if underflow

        LJMP TIMER_FINISHED             ; Timer has reached zero

CONTINUE_TIMER:
RET                                     ; Return from subroutine

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TIMER_FINISHED:
        MOV R3, #01H                        ; Clear display and home cursor
        ACALL SEND_CMD                      ; Send command
        MOV DPTR,#FINISHED_TEXT             ; Load finished message
FINISHED_LOOP:MOV A,#00H                ; Clear A
        MOVC A,@A+DPTR                      ; Get character
        JZ ACTIVATE_BUZZER                  ; If zero, end of string
        MOV R3,A                            ; Move to R3
        ACALL DISPLAY_CHAR                  ; Display character
        INC DPTR                            ; Next character
        LJMP FINISHED_LOOP                  ; Continue

ACTIVATE_BUZZER: SETB P2.7              ; Turn on buzzer
LCALL LONG_DELAY                        ; Wait for a while
CLR P2.7                                ; Turn off buzzer

WAIT_RESTART:JB P2.6, WAIT_RESTART      ; Wait for restart button
        LJMP INIT_ALL                       ; Reset system

DISPLAY_CHAR:
MOV P1, R3                              ; Move character data to P1
SETB RS                                 ; Select data register
SETB EN                                 ; Enable high
CLR EN                                  ; Enable low (latch data)
ACALL DELAY                             ; Wait for LCD to process
RET                                     ; Return from subroutine

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SEND_CMD:
MOV P1, R3                              ; Move command data to P1
CLR RS                                  ; Select command register
SETB EN                                 ; Enable high
CLR EN                                  ; Enable low (latch command)
ACALL DELAY                             ; Wait for LCD to process
RET                                     ; Return from subroutine



SCAN:
KEY_LOOP:
        JNB     P0.0, COL1                  ; Check if column 1 is active
        JNB     P0.1, COL2                  ; Check if column 2 is active
        JNB     P0.2, COL3                  ; Check if column 3 is active
        JNB     P0.3, COL4                  ; Check if column 4 is active
        SJMP    EXIT_SCAN                   ; No key pressed, exit
COL1:
        JNB     P0.4, NUMBER_1              ; Check for key 1
        JNB     P0.5, NUMBER_4              ; Check for key 4
        JNB     P0.6, NUMBER_7              ; Check for key 7
        JNB     P0.7, JUMP_F                ; Check for key F
        SETB    P0.0                        ; Reset column 1
        CLR     P0.1                        ; Select column 2
        SJMP    EXIT_SCAN                   ; Exit scan
COL2:
        JNB     P0.4, NUMBER_2              ; Check for key 2
        JNB     P0.5, NUMBER_5              ; Check for key 5
        JNB     P0.6, NUMBER_8              ; Check for key 8
        JNB     P0.7, NUMBER_0              ; Check for key 0
        SETB    P0.1                        ; Reset column 2
        CLR     P0.2                        ; Select column 3
        SJMP    EXIT_SCAN                   ; Exit scan
COL3:
        JNB     P0.4, NUMBER_3              ; Check for key 3
        JNB     P0.5, NUMBER_6              ; Check for key 6
        JNB     P0.6, NUMBER_9              ; Check for key 9
        JNB     P0.7, JUMP_E                ; Check for key E
        SETB    P0.2                        ; Reset column 3
        CLR     P0.3                        ; Select column 4
        SJMP    EXIT_SCAN                   ; Exit scan
COL4:
        JNB     P0.4, NUMBER_A              ; Check for key A
        JNB     P0.5, NUMBER_B              ; Check for key B
        JNB     P0.6, NUMBER_C              ; Check for key C
        JNB     P0.7, JUMP_D                ; Check for key D
        SETB    P0.3                        ; Reset column 4
        CLR     P0.0                        ; Select column 1
        LJMP    EXIT_SCAN                   ; Exit scan
```

```
EXIT_SCAN:
        RET                                  ; Return from subroutine


JUMP_A: LJMP NUMBER_A                   ; Jump to key A handler
JUMP_B: LJMP NUMBER_B                   ; Jump to key B handler
JUMP_C: LJMP NUMBER_C                   ; Jump to key C handler
JUMP_D: LJMP NUMBER_D                   ; Jump to key D handler
JUMP_E: LJMP NUMBER_E                   ; Jump to key E handler
JUMP_F: LJMP NUMBER_F                   ; Jump to key F handler


NUMBER_0:
        MOV     R0, #16D                      ; Store key value 0
        LJMP    KEY_LOOP                      ; Return to scanning
NUMBER_1:
        MOV     R0, #1D                       ; Store key value 1
        LJMP    KEY_LOOP                      ; Return to scanning
NUMBER_2:
        MOV     R0, #2D                       ; Store key value 2
        LJMP    KEY_LOOP                      ; Return to scanning
NUMBER_3:
        MOV     R0, #3D                       ; Store key value 3
        LJMP    KEY_LOOP                      ; Return to scanning
NUMBER_4:
        MOV     R0, #4D                       ; Store key value 4
        LJMP    KEY_LOOP                      ; Return to scanning
NUMBER_5:
        MOV     R0, #5D                       ; Store key value 5
        LJMP    KEY_LOOP                      ; Return to scanning
NUMBER_6:
        MOV     R0, #6D                       ; Store key value 6
        LJMP    KEY_LOOP                      ; Return to scanning
NUMBER_7:
        MOV     R0, #7D                       ; Store key value 7
        LJMP    KEY_LOOP                      ; Return to scanning
NUMBER_8:
        MOV     R0, #8D                       ; Store key value 8
        LJMP    KEY_LOOP                      ; Return to scanning
NUMBER_9:
        MOV     R0, #9D                       ; Store key value 9
        LJMP    KEY_LOOP                      ; Return to scanning
NUMBER_A:
        MOV R0, #10                          ; Store key value A
        LJMP KEY_LOOP                        ; Return to scanning
NUMBER_B:
        MOV R0, #11                          ; Store key value B
        LJMP KEY_LOOP                        ; Return to scanning
NUMBER_C:
        MOV R0, #12                          ; Store key value C
        LJMP KEY_LOOP                        ; Return to scanning
NUMBER_D:
        MOV R0, #13                          ; Store key value D
        LJMP KEY_LOOP                        ; Return to scanning
NUMBER_E:
        MOV R0, #14                          ; Store key value E
        LJMP KEY_LOOP                        ; Return to scanning
NUMBER_F:
        MOV R0, #15                          ; Store key value F
        LJMP KEY_LOOP                        ; Return to scanning


SHOW_DIGIT1: CLR P2.0                   ; Select first digit
        ;MOV A,30H
        ;JNZ DISP1DONE

        MOV     A,40h                             ; Get hundreds digit
        mov     dptr,#SEGMENT_PATTERNS        ; Load segment pattern table
        movc    A,@a+dptr                     ; Get pattern for digit
        mov     P3,A                          ; Output to display
        LCALL   SHORT_DELAY                   ; Small delay
        MOV P3,#00H                         ; Turn off segments
        SETB P2.0                           ; Deselect display
        RET                                 ; Return from subroutine

SHOW_DIGIT2: CLR P2.3                   ; Select second digit
        ;MOV A,30H
        ;JNZ DISP1DONE

        MOV     A,44h                             ; Get tens digit
        mov     dptr,#SEGMENT_PATTERNS        ; Load segment pattern table
        movc    A,@a+dptr                     ; Get pattern for digit
        mov     P3,A                          ; Output to display
        LCALL   SHORT_DELAY                   ; Small delay
        MOV P3,#00H                         ; Turn off segments

        SETB P2.3                           ; Deselect display
        RET                                 ; Return from subroutine


SHOW_DIGIT3: CLR P2.4                   ; Select third digit
```

```
         ;MOV A,30H
         ;JNZ DISP1DONE

         MOV      A,53h                         ; Get units digit
         mov      dptr,#SEGMENT_PATTERNS       ; Load segment pattern table
         movc     A,@a+dptr                   ; Get pattern for digit
         mov      P3,A                        ; Output to display
         LCALL    SHORT_DELAY                 ; Small delay
         MOV P3,#00H                         ; Turn off segments

         SETB P2.4                           ; Deselect display
         RET                                 ; Return from subroutine
SHORT_DELAY:     MOV      R1, #10             ; Short delay routine
HERE2:   MOV      R2, #255               ; Inner loop count
HERE:    DJNZ     R2, HERE           ; Decrement inner loop
         DJNZ     R1, HERE2                   ; Decrement outer loop
         RET                                 ; Return from subroutine


DELAY:   MOV      R1, #50                     ; Medium delay routine
HER2:    MOV      R2, #255               ; Inner loop count
HER:     DJNZ     R2, HER                ; Decrement inner loop
         DJNZ     R1, HER2                    ; Decrement outer loop
         RET                                 ; Return from subroutine

LONG_DELAY:  MOV R0, #10                    ; Long delay routine
HE3:         MOV R1, #255                   ; Outer loop count
HE2:         MOV R2, #255                   ; Middle loop count
HE:          DJNZ R2, HE                    ; Decrement inner loop
             DJNZ R1, HE2                   ; Decrement middle loop
             DJNZ R0, HE3                   ; Decrement outer loop
             RET                            ; Return from subroutine

WAIT_KEYRELEASE:  MOV R0, #5                ; Key debounce delay
sHE3:         MOV R1, #255                  ; Outer loop count
sHE2:         MOV R2, #255                  ; Middle loop count
sHE:          DJNZ R2, sHE                  ; Decrement inner loop
             DJNZ R1, sHE2                  ; Decrement middle loop
             DJNZ R0, sHE3                  ; Decrement outer loop
             RET                            ; Return from subroutine

DELAY_ONE_SEC:
    CLR TR0                                 ; Stop Timer 0
    CLR TF0                                 ; Clear Timer 0 overflow flag
    MOV TMOD, #01H                          ; Timer 0 in 16-bit mode

    MOV TH0, #3CH                           ; High byte of initial value
    MOV TL0, #98H                           ; Low byte of initial value
    SETB TR0                                ; Start Timer 0

WAIT_TIMER:LCALL SHOW_DIGIT1                ; Display first digit
         LCALL SHOW_DIGIT2                      ; Display second digit
         LCALL SHOW_DIGIT3                  ; Display third digit
    JNB TF0, WAIT_TIMER                     ; Wait for timer overflow
    CLR TR0                                 ; Stop Timer 0
    CLR TF0                                 ; Clear overflow flag
    DJNZ R7, DELAY_ONE_SEC                  ; Decrement R7 and loop if not zero
    MOV R7, #15                             ; Reset counter
    ;SJMP DELAY_LOOP                        ; (Commented out)
RET                                         ; Return from subroutine

org  600h
;00111001B
SEGMENT_PATTERNS:DB 3FH,06H,05BH,04FH,066H,06DH, 07DH,07H,07FH,06FH, 077H,07CH,039H,05EH,079H,071H,3FH

OVEN_START_TEXT: DB "OVEN STARTED",0

GREATER_300_TEXT: DB "VALUE > 300",0

BELOW_5_TEXT: DB "VALUE < 5",0

ABOVE_60_TEXT: DB "VALUE > 60",0

BELOW_60_TEXT: DB "VALUE < 60",0
FINISHED_TEXT: DB "OVEN STOPPED",0
RETRY_TEXT: DB "TRY AGAIN",0
FACT1_TEXT: DB "Cats love naps",0
FACT2_TEXT: DB "Ctrl+Z saves",0
FACT3_TEXT: DB "DC > MARVEL",0
FACT4_TEXT: DB "VR feels real",0
FACT5_TEXT: DB "Linux is free",0
FACT6_TEXT: DB "Dark mode saves",0
FACT7_TEXT: DB "IUT food great",0
FACT8_TEXT: DB "Frogs freeze",0
FACT9_TEXT: DB "DC > MARVEL",0
FACT10_TEXT: DB "Clock ticking",0
FACT11_TEXT: DB "Batman = goat",0

MY_FACT_TEXT: DB "SHAFIN FOUND ME",0

      END
```