



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**

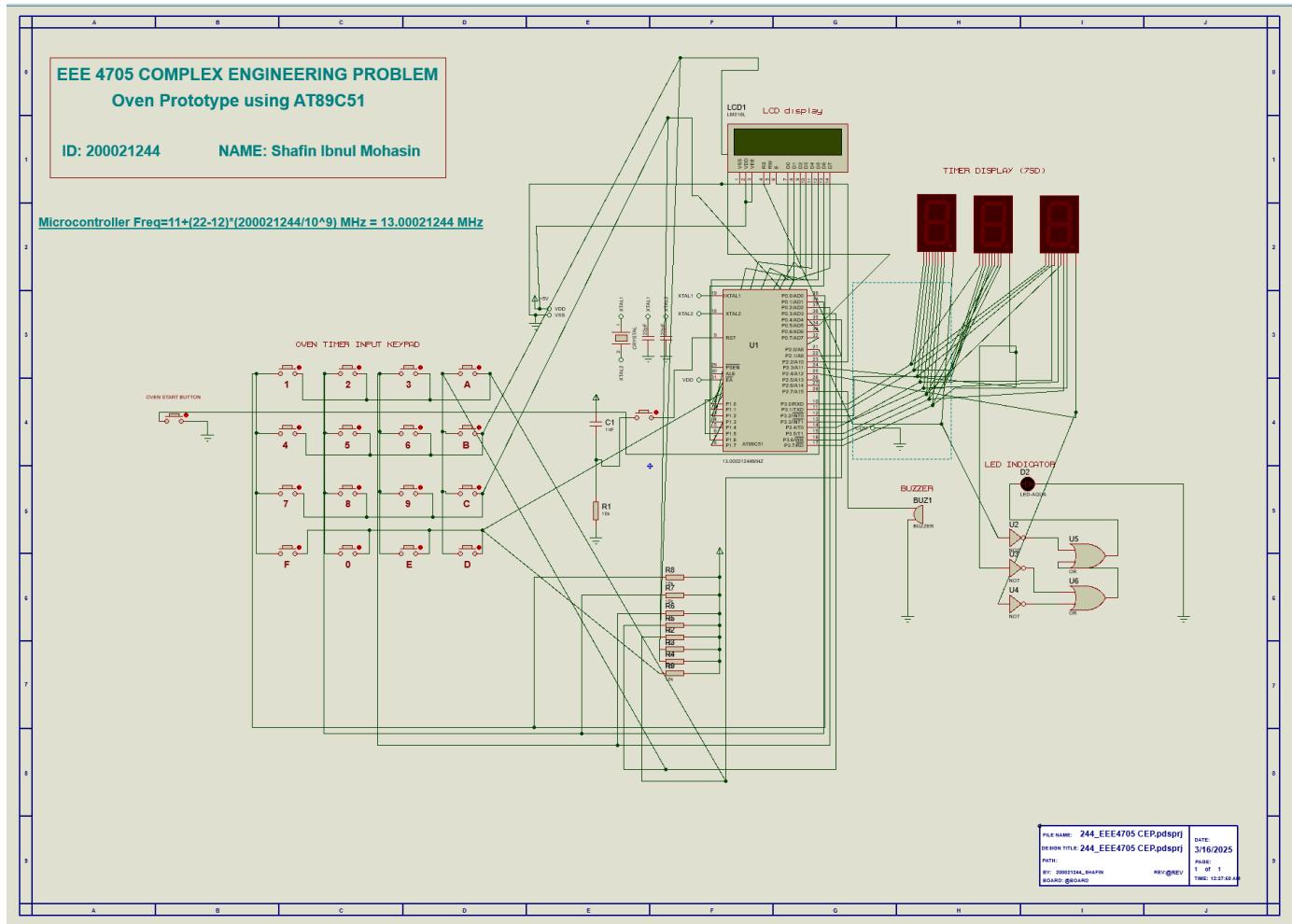
Shafin Ibnul Mohasin Sec. B Dept.: EEE	200021244
--	-----------

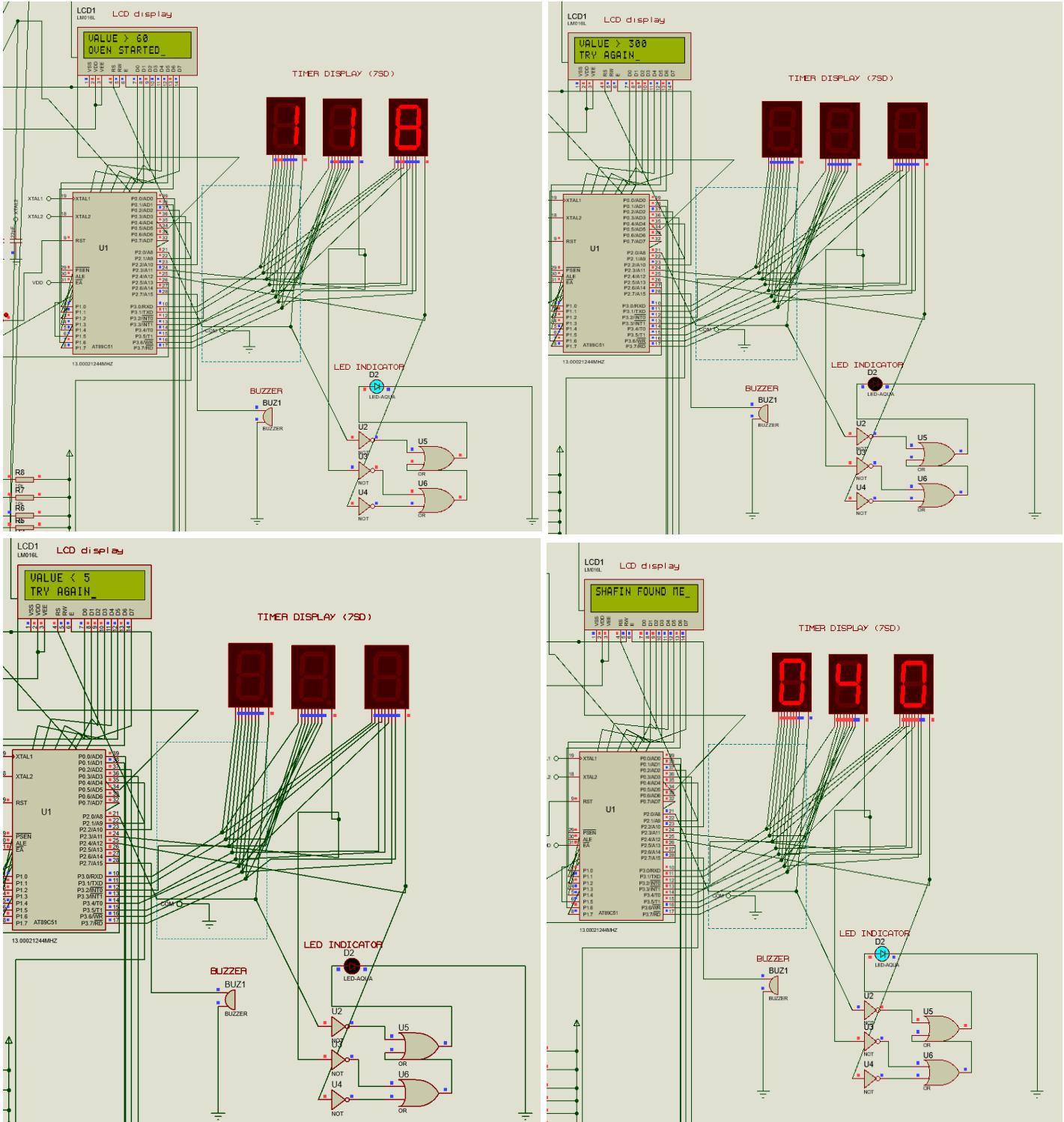
**Submission Date:** 18/03/2025.

---

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

## Proteus Layout & Simulation:





## Code:

```
1. ;NAME: Shafin IbnuL Mohasin ID: 200021244
2. org 0000h
3.
4. INIT_ALL:      MOV     P3,#0000000B ; Clear port 3
5.             MOV P0, #0FEH ; Initialize port 0
6.             MOV 30H,#0 ; Reset memory location 30H
7.             MOV 32H,#0 ; Reset memory location 32H
8.             MOV R0,#0 ; Clear register R0
9.             MOV R7, #15 ; Set R7 to 15
10.            mov r5,#00H ; Clear register R5
11.            MOV 69H,0H ; Clear memory location 69H
12.            CLR P2.7 ; Clear buzzer pin
13.            MOV P1, #0000000B ; Clear port 1
14.
15. REGISTERS_SETUP:
16. MOV R3, #00H ; Clear register R3
17. MOV R1, #00H ; Clear register R1
18. MOV R2, #00H ; Clear register R2
19.
20.
21.
22. PORT_SETUP:
23. RS EQU P2.1 ; Define RS pin for LCD
24. EN EQU P2.2 ; Define EN pin for LCD
25.
26.
27. ;;;;;;;;;;;;;;;
28. SETUP_LCD:
29. MOV R3, #38H ; Set 8-bit mode, 2 lines
30. ACALL SEND_CMD ; Send command to LCD
31. MOV R3, #0EH ; Display on, cursor on
32. ACALL SEND_CMD
33. MOV R3, #80H ; Set cursor to first line
34. ACALL SEND_CMD
35. MOV R3, #01H ; Clear display
36. ACALL SEND_CMD
37.
38.
39. MAIN_SCAN:      LCALL SCAN ; Scan keypad
40.             MOV A,R0 ; Move key value to A
41.             JZ MAIN_SCAN ; If zero, keep scanning
42.
43.             MOV 40H,A ; Store first digit
44.
45.             lcall WAIT_KEYRELEASE ; Delay for key debounce
46. MAIN_SCAN2:      LCALL SCAN ; Scan keypad for second digit
47.             MOV A,R0 ; Move key value to A
48.             JZ MAIN_SCAN2 ; If zero, keep scanning
49.
50.             MOV 44H,A ; Store second digit
51.
52.             lcall WAIT_KEYRELEASE ; Delay for key debounce
53. MAIN_SCAN3:      LCALL SCAN ; Scan keypad for third digit
54.             MOV A,R0 ; Move key value to A
55.             JZ MAIN_SCAN3 ; If zero, keep scanning
56.
57.             MOV 53H,A ; Store third digit
58.
59.
60. WAIT_START:      JB P2.5, WAIT_START ; Wait until start button pressed
61.             ANL 53H,#00001111B ; Mask upper nibble (keep only lower 4 bits)
62.             ANL 40H,#00001111B ; Mask upper nibble
63.             ANL 44H,#00001111B ; Mask upper nibble
64.
65.             MOV A,44H ; Get second digit
66.             MOV B,A ; Store in B
67.             MOV A,#10 ; Multiply by 10
68.             MUL AB ; Perform multiplication
69.             ADD A,53H ; Add third digit
70.             MOV 60H,A ; Store result in 60H
71.
72.
73.             MOV A,40H ; Get first digit
74.             MOV B,A ; Move to B
75.             MOV A,#100 ; Multiply by 100
76.             MUL AB ; Perform multiplication
77.             MOV 62H,A ; Store lower byte
78.             MOV A,B ; Get upper byte
79.             MOV 61H,A ; Store upper byte
80.             MOV A,62H ; Get lower byte
81.             ADD A,60H ; Add previous result
82.             MOV 62H,A ; Store new result
83.             JNC THRESHOLD_CHECK ; Check if no carry
84.             INC 61H ; Increment upper byte if carry
85.
86.
```

```

87. THRESHOLD_CHECK: MOV A,61H ; Get high byte
88. CJNE A,#01H,THRESHOLD_CHECK1 ; Compare with 01H (300 high byte)
89. MOV A,62H ; Get low byte
90. CJNE A,#2DH,THRESHOLD_CHECK2 ; Compare with 2DH (300 low byte)
91. JMP GREATER_300 ; Exactly 300, treat as > 300
92.
93. THRESHOLD_CHECK1: JC THRESHOLD_CHECK5 ; If high byte < 01H, number is < 300
94. JMP GREATER_300 ; If high byte > 01H, number is > 300
95.
96. THRESHOLD_CHECK2: JC THRESHOLD_CHECK5 ; If low byte < 2DH, number is < 300
97. JMP GREATER_300 ; If low byte > 2DH, number is > 300
98.
99. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
100. THRESHOLD_CHECK5: MOV A,61H ; Get high byte
101. JNZ THRESHOLD_60 ; If high byte > 0, number is > 5
102. MOV A,62H ; Get low byte
103. CJNE A,#05H,CHECK_5_TEMP ; Compare with 5
104. JMP THRESHOLD_60 ; Exactly 5, check next threshold
105.
106. CHECK_5_TEMP: JC BELOW_5 ; If < 5, handle separately
107. JMP THRESHOLD_60 ; If > 5, check next threshold
108.
109. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
110.
111.
112.
113. BELOW_5: MOV DPTR,#BELOW_5_TEXT ; Load message address
114. BELOW_5_LOOP:MOV A,#00H ; Clear A
115. MOVC A,@A+DPTR ; Get character
116. JZ TMP_LOOP ; If zero, end of string
117. MOV R3,A ; Move to R3
118. ACALL DISPLAY_CHAR ; Display character
119. INC DPTR ; Next character
120. LJMP BELOW_5_LOOP ; Continue
121.
122.
123. ;;;;;;;;;;;;;;;;;;;
124.
125. TMP_LOOP : MOV R3, #0COH ; Set cursor to second line
126. ACALL SEND_CMD ; Send command
127. MOV DPTR,#RETRY_TEXT ; Load retry message
128. RETRY_LOOP:MOV A,#00H ; Clear A
129. MOVC A,@A+DPTR ; Get character
130. JZ RETRY_WAIT ; If zero, end of string
131. MOV R3,A ; Move to R3
132. ACALL DISPLAY_CHAR ; Display character
133. INC DPTR ; Next character
134. LJMP RETRY_LOOP ; Continue
135.
136. RETRY_WAIT: LCALL LONG_DELAY ; Wait for a while
137. LJMP INIT_ALL ; Restart program
138.
139.
140. THRESHOLD_60: MOV A,61H ; Get high byte
141. JNZ ABOVE_60 ; If high byte > 0, number is > 60
142. MOV A,62H ; Get low byte
143. CJNE A,#3CH,CHECK_60_TEMP ; Compare with 60 (3ch)
144. JMP ABOVE_60 ; Exactly 60, treat as > 60
145.
146. CHECK_60_TEMP: JC BELOW_60 ; If < 60, handle separately
147. JMP ABOVE_60 ; If > 60, handle accordingly
148.
149.
150.
151. ABOVE_60 : MOV DPTR,#ABOVE_60_TEXT ; Load message address
152. ABOVE_60_LOOP:MOV A,#00H ; Clear A
153. MOVC A,@A+DPTR ; Get character
154. JZ OVEN_START_2 ; If zero, end of string
155. MOV R3,A ; Move to R3
156. ACALL DISPLAY_CHAR ; Display character
157. INC DPTR ; Next character
158. LJMP ABOVE_60_LOOP ; Continue
159.
160.
161. BELOW_60 : MOV DPTR,#BELOW_60_TEXT ; Load message address
162. BELOW_60_LOOP:MOV A,#00H ; Clear A
163. MOVC A,@A+DPTR ; Get character
164. JZ OVEN_START_1 ; If zero, end of string
165. MOV R3,A ; Move to R3
166. ACALL DISPLAY_CHAR ; Display character
167. INC DPTR ; Next character
168. LJMP BELOW_60_LOOP ; Continue
169.
170.
171. GREATER_300 : MOV DPTR,#GREATER_300_TEXT ; Load message address
172. GREATER_300_LOOP:MOV A,#00H ; Clear A
173. MOVC A,@A+DPTR ; Get character
174. JZ TMP_LOOP ; If zero, goto retry
175. MOV R3,A ; Move to R3
176. ACALL DISPLAY_CHAR ; Display character
177. INC DPTR ; Next character

```

```

178.      LJMP GREATER_300_LOOP           ; Continue
179.
180.
181. OVEN_START_2:MOV DPTR,#OVEN_START_TEXT    ; Load oven message
182.         MOV R3, #0COH                 ; Set cursor to second line
183.         ACALL SEND_CMD              ; Send command
184. OVEN_START_LOOP2:MOV A,#00H               ; Clear A
185.         MOVC A,@A+DPTR             ; Get character
186.         JZ TIMER_LOOP2            ; If zero, start timer
187.         MOV R3,A                  ; Move to R3
188.         ACALL DISPLAY_CHAR       ; Display character
189.         INC DPTR                ; Next character
190.         LJMP OVEN_START_LOOP2     ; Continue
191.
192.
193. OVEN_START_1:MOV DPTR,#OVEN_START_TEXT    ; Load oven message
194.         MOV R3, #0COH                 ; Set cursor to second line
195.         ACALL SEND_CMD              ; Send command
196. OVEN_START_LOOP:MOV A,#00H               ; Clear A
197.         MOVC A,@A+DPTR             ; Get character
198.         JZ TIMER_LOOP             ; If zero, start timer
199.         MOV R3,A                  ; Move to R3
200.         ACALL DISPLAY_CHAR       ; Display character
201.         INC DPTR                ; Next character
202.         LJMP OVEN_START_LOOP     ; Continue
203.
204. TIMER_LOOP2:MOV R6,#20                  ; Set counter to 5
205. ;LCALL DISPLAY_FACT1              ; Display fact (commented out)
206. TIMER_LOOP2_TEMP:
207.         LCALL DELAY_ONE_SEC        ; Wait one second
208.         LCALL DECREMENT_NUMBER    ; Decrement the timer
209.         DJNZ R6,TIMER_LOOP2_TEMP  ; Loop until R6 is zero
210.         LCALL DISPLAY_RANDOM_FACT ; Display random fact
211.         MOV R6,#20                ; Reset counter
212.         SJMP TIMER_LOOP2_TEMP     ; Continue timer loop
213.
214.
215. TIMER_LOOP: LCALL LONG_DELAY          ; Wait for a while
216. LCALL DISPLAY_MY_FACT              ; Display my fact
217. TIMER_LOOP_TEMP:LCALL DELAY_ONE_SEC  ; Wait one second
218.         LCALL DECREMENT_NUMBER    ; Decrement the timer
219.
220.         SJMP TIMER_LOOP_TEMP      ; Continue timer loop
221.
222.
223.
224. DISPLAY_MY_FACT:
225. MOV DPTR,#MY_FACT_TEXT              ; Load fact message
226.         MOV R3, #01H                 ; Clear display and home cursor
227.         ACALL SEND_CMD              ; Send command
228. DISPLAY_MY_FACT_LOOP:MOV A,#00H      ; Clear A
229.         MOVC A,@A+DPTR             ; Get character
230.         JZ DISPLAY_MY_FACT_END    ; If zero, end of string
231.         MOV R3,A                  ; Move to R3
232.         ACALL DISPLAY_CHAR       ; Display character
233.         INC DPTR                ; Next character
234.         LJMP DISPLAY_MY_FACT_LOOP ; Continue
235. DISPLAY_MY_FACT_END:
236. RET                                ; Return
237.
238.
239.
240. DISPLAY_FACT1:
241. MOV DPTR,#FACT1_TEXT                ; Load fact 1 message
242.         MOV R3, #01H                 ; Clear display and home cursor
243.         ACALL SEND_CMD              ; Send command
244. DISPLAY_FACT1_LOOP:MOV A,#00H      ; Clear A
245.         MOVC A,@A+DPTR             ; Get character
246.         JZ DISPLAY_FACT1_END    ; If zero, end of string
247.         MOV R3,A                  ; Move to R3
248.         ACALL DISPLAY_CHAR       ; Display character
249.         INC DPTR                ; Next character
250.         LJMP DISPLAY_FACT1_LOOP   ; Continue
251. DISPLAY_FACT1_END:
252. RET                                ; Return
253.
254. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;-----
255. DISPLAY_RANDOM_FACT:
256.
257. INC R5                               ; Increment fact counter
258. CJNE R5,#01H,DISPLAY_FACT2          ; Check if fact 1
259. MOV DPTR,#FACT2_TEXT                ; Load fact 2 message
260.         MOV R3, #01H                 ; Clear display and home cursor
261.         ACALL SEND_CMD              ; Send command
262. DISPLAY_FACT2_LOOP:MOV A,#00H      ; Clear A
263.         MOVC A,@A+DPTR             ; Get character
264.         JZ DISPLAY_RANDOM_FACT_END ; If zero, end of string
265.         MOV R3,A                  ; Move to R3
266.         ACALL DISPLAY_CHAR       ; Display character
267.         INC DPTR                ; Next character

```

```

268.      LJMP DISPLAY_FACT2_LOOP           ; Continue
269.
270. DISPLAY_FACT2:
271.     CJNE R5,#02H,DISPLAY_FACT3
272.     MOV DPTR,#FACT3_TEXT
273.     MOV R3, #01H
274.     ACALL SEND_CMD
275. DISPLAY_FACT3_LOOP:MOV A,#00H
276.     MOVC A,@A+DPTR
277.     JZ DISPLAY_RANDOM_FACT_END
278.     MOV R3,A
279.     ACALL DISPLAY_CHAR
280.     INC DPTR
281.     LJMP DISPLAY_FACT3_LOOP
282.
283.
284. DISPLAY_FACT3:
285.     CJNE R5,#03H,DISPLAY_FACT4
286.     MOV DPTR,#FACT4_TEXT
287.     MOV R3, #01H
288.     ACALL SEND_CMD
289. DISPLAY_FACT4_LOOP:MOV A,#00H
290.     MOVC A,@A+DPTR
291.     JZ DISPLAY_RANDOM_FACT_END
292.     MOV R3,A
293.     ACALL DISPLAY_CHAR
294.     INC DPTR
295.     LJMP DISPLAY_FACT4_LOOP
296.
297.
298. DISPLAY_FACT4:
299.     CJNE R5,#04H,DISPLAY_FACT5
300.     MOV DPTR,#FACT5_TEXT
301.     MOV R3, #01H
302.     ACALL SEND_CMD
303. DISPLAY_FACT5_LOOP:MOV A,#00H
304.     MOVC A,@A+DPTR
305.     JZ DISPLAY_RANDOM_FACT_END
306.     MOV R3,A
307.     ACALL DISPLAY_CHAR
308.     INC DPTR
309.     LJMP DISPLAY_FACT5_LOOP
310.
311.
312. DISPLAY_FACT5:
313.     CJNE R5,#05H,DISPLAY_FACT6
314.     MOV DPTR,#FACT6_TEXT
315.     MOV R3, #01H
316.     ACALL SEND_CMD
317. DISPLAY_FACT6_LOOP:MOV A,#00H
318.     MOVC A,@A+DPTR
319.     JZ DISPLAY_RANDOM_FACT_END
320.     MOV R3,A
321.     ACALL DISPLAY_CHAR
322.     INC DPTR
323.     LJMP DISPLAY_FACT6_LOOP
324.
325.
326. DISPLAY_FACT6:
327.     CJNE R5,#06H,RESET_FACT_COUNTER
328.     MOV DPTR,#FACT7_TEXT
329.     MOV R3, #01H
330.     ACALL SEND_CMD
331. DISPLAY_FACT7_LOOP:MOV A,#00H
332.     MOVC A,@A+DPTR
333.     JZ DISPLAY_RANDOM_FACT_END
334.     MOV R3,A
335.     ACALL DISPLAY_CHAR
336.     INC DPTR
337.     LJMP DISPLAY_FACT7_LOOP
338.
339.
340. RESET_FACT_COUNTER: MOV R5,#0H
341.     LJMP DISPLAY_RANDOM_FACT           ; Reset fact counter to 0
342.     ; Go back to display first fact
343.
344. DISPLAY_RANDOM_FACT_END:
345. RET                         ; Return from subroutine
346. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
347. DECREMENT_NUMBER:
348.     DEC 53H                     ; Decrement units digit
349.     MOV A, 53H                  ; Move to accumulator
350.
351.     CJNE A, #1111111B, CONTINUE_TIMER ; Check if underflow
352.
353.     ; Reset units digit and decrement tens digit
354.     MOV 53H, #9                 ; Reset to 9
355.     DEC 44H                     ; Decrement tens digit
356.     MOV A, 44H                  ; Move to accumulator
357.
358.     CJNE A, #1111111B, CONTINUE_TIMER ; Check if underflow

```

```

359.      ; Reset tens digit and decrement hundreds digit
360.      MOV 44H, #9          ; Reset to 9
361.      DEC 40H          ; Decrement hundreds digit
362.      MOV A, 40H          ; Move to accumulator
363.
364.      CJNE A, #11111111B, CONTINUE_TIMER ; Check if underflow
365.      LJMP TIMER_FINISHED           ; Timer has reached zero
366.
367.      RET                    ; Return from subroutine
368.
369. CONTINUE_TIMER:
370.      RET
371.
372.      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
373. TIMER_FINISHED:
374.      MOV R3, #01H          ; Clear display and home cursor
375.      ACALL SEND_CMD       ; Send command
376.      MOV DPTR, #FINISHED_TEXT ; Load finished message
377. FINISHED_LOOP:MOV A,#00H
378.      MOVC A,@A+DPTR        ; Clear A
379.      JZ ACTIVATE_BUZZER    ; Get character
380.      MOV R3,A              ; If zero, end of string
381.      ACALL DISPLAY_CHAR     ; Move to 'R3
382.      INC DPTR             ; Display character
383.      LJMP FINISHED_LOOP     ; Next character
384.      ; Continue
385. ACTIVATE_BUZZER: SETB P2.7
386. LCALL LONG_DELAY
387. CLR P2.7
388.
389. WAIT_RESTART:JB P2.6, WAIT_RESTART
390.      LJMP INIT_ALL          ; Wait for restart button
391.      ; Reset system
392.
393. DISPLAY_CHAR:
394.      MOV P1, R3            ; Move character data to P1
395.      SETB RS              ; Select data register
396.      SETB EN              ; Enable high
397.      CLR EN              ; Enable low (latch data)
398.      ACALL DELAY          ; Wait for LCD to process
399.      RET                  ; Return from subroutine
400.      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
401. SEND_CMD:
402.      MOV P1, R3            ; Move command data to P1
403.      CLR RS              ; Select command register
404.      SETB EN              ; Enable high
405.      CLR EN              ; Enable low (latch command)
406.      ACALL DELAY          ; Wait for LCD to process
407.      RET                  ; Return from subroutine
408.
409.
410.
411. SCAN:
412. KEY_LOOP:
413.      JNB P0.0, COL1          ; Check if column 1 is active
414.      JNB P0.1, COL2          ; Check if column 2 is active
415.      JNB P0.2, COL3          ; Check if column 3 is active
416.      JNB P0.3, COL4          ; Check if column 4 is active
417.      SJMP EXIT_SCAN         ; No key pressed, exit
418. COL1:
419.      JNB P0.4, NUMBER_1        ; Check for key 1
420.      JNB P0.5, NUMBER_4        ; Check for key 4
421.      JNB P0.6, NUMBER_7        ; Check for key 7
422.      JNB P0.7, JUMP_F          ; Check for key F
423.      SETB P0.0                ; Reset column 1
424.      CLR P0.1                ; Select column 2
425.      SJMP EXIT_SCAN         ; Exit scan
426. COL2:
427.      JNB P0.4, NUMBER_2        ; Check for key 2
428.      JNB P0.5, NUMBER_5        ; Check for key 5
429.      JNB P0.6, NUMBER_8        ; Check for key 8
430.      JNB P0.7, NUMBER_0        ; Check for key 0
431.      SETB P0.1                ; Reset column 2
432.      CLR P0.2                ; Select column 3
433.      SJMP EXIT_SCAN         ; Exit scan
434. COL3:
435.      JNB P0.4, NUMBER_3        ; Check for key 3
436.      JNB P0.5, NUMBER_6        ; Check for key 6
437.      JNB P0.6, NUMBER_9        ; Check for key 9
438.      JNB P0.7, JUMP_E          ; Check for key E
439.      SETB P0.2                ; Reset column 3
440.      CLR P0.3                ; Select column 4
441.      SJMP EXIT_SCAN         ; Exit scan
442. COL4:
443.      JNB P0.4, NUMBER_A        ; Check for key A
444.      JNB P0.5, NUMBER_B        ; Check for key B
445.      JNB P0.6, NUMBER_C        ; Check for key C
446.      JNB P0.7, JUMP_D          ; Check for key D
447.      SETB P0.3                ; Reset column 4
448.      CLR P0.0                ; Select column 1
449.      LJMP EXIT_SCAN         ; Exit scan

```

```

450. EXIT_SCAN:
451.         RET          ; Return from subroutine
452.
453.
454. JUMP_A: LJMP NUMBER_A      ; Jump to key A handler
455. JUMP_B: LJMP NUMBER_B      ; Jump to key B handler
456. JUMP_C: LJMP NUMBER_C      ; Jump to key C handler
457. JUMP_D: LJMP NUMBER_D      ; Jump to key D handler
458. JUMP_E: LJMP NUMBER_E      ; Jump to key E handler
459. JUMP_F: LJMP NUMBER_F      ; Jump to key F handler
460.
461.
462. NUMBER_0:
463.         MOV    R0, #16D      ; Store key value 0
464.         LJMP KEY_LOOP      ; Return to scanning
465. NUMBER_1:
466.         MOV    R0, #1D       ; Store key value 1
467.         LJMP KEY_LOOP      ; Return to scanning
468. NUMBER_2:
469.         MOV    R0, #2D       ; Store key value 2
470.         LJMP KEY_LOOP      ; Return to scanning
471. NUMBER_3:
472.         MOV    R0, #3D       ; Store key value 3
473.         LJMP KEY_LOOP      ; Return to scanning
474. NUMBER_4:
475.         MOV    R0, #4D       ; Store key value 4
476.         LJMP KEY_LOOP      ; Return to scanning
477. NUMBER_5:
478.         MOV    R0, #5D       ; Store key value 5
479.         LJMP KEY_LOOP      ; Return to scanning
480. NUMBER_6:
481.         MOV    R0, #6D       ; Store key value 6
482.         LJMP KEY_LOOP      ; Return to scanning
483. NUMBER_7:
484.         MOV    R0, #7D       ; Store key value 7
485.         LJMP KEY_LOOP      ; Return to scanning
486. NUMBER_8:
487.         MOV    R0, #8D       ; Store key value 8
488.         LJMP KEY_LOOP      ; Return to scanning
489. NUMBER_9:
490.         MOV    R0, #9D       ; Store key value 9
491.         LJMP KEY_LOOP      ; Return to scanning
492. NUMBER_A:
493.         MOV    R0, #10        ; Store key value A
494.         LJMP KEY_LOOP      ; Return to scanning
495. NUMBER_B:
496.         MOV    R0, #11        ; Store key value B
497.         LJMP KEY_LOOP      ; Return to scanning
498. NUMBER_C:
499.         MOV    R0, #12        ; Store key value C
500.         LJMP KEY_LOOP      ; Return to scanning
501. NUMBER_D:
502.         MOV    R0, #13        ; Store key value D
503.         LJMP KEY_LOOP      ; Return to scanning
504. NUMBER_E:
505.         MOV    R0, #14        ; Store key value E
506.         LJMP KEY_LOOP      ; Return to scanning
507. NUMBER_F:
508.         MOV    R0, #15        ; Store key value F
509.         LJMP KEY_LOOP      ; Return to scanning
510.
511.
512. SHOW_DIGIT1: CLR P2.0      ; Select first digit
513.         ;MOV A,30H
514.         ;JNZ DISP1DONE
515.
516.         MOV    A,40h        ; Get hundreds digit
517.         mov    dptr,#SEGMENT_PATTERNS ; Load segment pattern table
518.         movc   A,@a+dptr      ; Get pattern for digit
519.         mov    P3,A           ; Output to display
520.         LCALL   SHORT_DELAY    ; Small delay
521.         MOV    P3,#00H        ; Turn off segments
522.         SETB   P2.0          ; Deselect display
523.         RET                ; Return from subroutine
524.
525. SHOW_DIGIT2: CLR P2.3      ; Select second digit
526.         ;MOV A,30H
527.         ;JNZ DISP1DONE
528.
529.         MOV    A,44h        ; Get tens digit
530.         mov    dptr,#SEGMENT_PATTERNS ; Load segment pattern table
531.         movc   A,@a+dptr      ; Get pattern for digit
532.         mov    P3,A           ; Output to display
533.         LCALL   SHORT_DELAY    ; Small delay
534.         MOV    P3,#00H        ; Turn off segments
535.
536.         SETB   P2.3          ; Deselect display
537.         RET                ; Return from subroutine
538.
539.
540. SHOW_DIGIT3: CLR P2.4      ; Select third digit

```

```

541.      ;MOV A,30H
542.      ;JNZ DISP1DONE
543.
544.      MOV     A,53h          ; Get units digit
545.      mov    dptr,#SEGMENT_PATTERNS ; Load segment pattern table
546.      MOVC   A,@a+dptr ; Get pattern for digit
547.      MOV    P3,A          ; Output to display
548.      LCALL  SHORT_DELAY ; Small delay
549.      MOV    P3,#00H        ; Turn off segments
550.
551.      SETB   P2.4          ; Deselect display
552.      RET
553. SHORT_DELAY:  MOV    R1, #10      ; Short delay routine
554. HERE2:   MOV    R2, HERE      ; Inner loop count
555. HERE:    DJNZ   R2, HERE      ; Decrement inner loop
556.         DJNZ   R1, HERE2    ; Decrement outer loop
557.         RET
558.
559.
560. DELAY:   MOV    R1, #50      ; Medium delay routine
561. HER2:   MOV    R2, #255     ; Inner loop count
562. HER:    DJNZ   R2, HER      ; Decrement inner loop
563.         DJNZ   R1, HER2    ; Decrement outer loop
564.         RET
565.
566. LONG_DELAY: MOV   R0, #10      ; Long delay routine
567. HE3:    MOV    R1, #255     ; Outer loop count
568. HE2:    MOV    R2, #255     ; Middle loop count
569. HE:     DJNZ   R2, HE       ; Decrement inner loop
570.         DJNZ   R1, HE2     ; Decrement middle loop
571.         DJNZ   R0, HE3     ; Decrement outer loop
572.         RET
573.
574. WAIT_KEYRELEASE: MOV   R0, #5      ; Key debounce delay
575. SHE3:   MOV    R1, #255     ; Outer loop count
576. SHE2:   MOV    R2, #255     ; Middle loop count
577. SHE:    DJNZ   R2, SHE      ; Decrement inner loop
578.         DJNZ   R1, SHE2    ; Decrement middle loop
579.         DJNZ   R0, SHE3    ; Decrement outer loop
580.         RET
581.
582. DELAY_ONE_SEC:
583.     CLR   TR0          ; Stop Timer 0
584.     CLR   TF0          ; Clear Timer 0 overflow flag
585.     MOV    TMOD, #01H    ; Timer 0 in 16-bit mode
586.
587.     MOV   TH0, #3CH     ; High byte of initial value
588.     MOV   TL0, #98H     ; Low byte of initial value
589.     SETB  TR0          ; Start Timer 0
590.
591. WAIT_TIMER:LCALL SHOW_DIGIT1 ; Display first digit
592.         LCALL SHOW_DIGIT2 ; Display second digit
593.         LCALL SHOW_DIGIT3 ; Display third digit
594.     JNB   TF0, WAIT_TIMER ; Wait for timer overflow
595.     CLR   TR0          ; Stop Timer 0
596.     CLR   TF0          ; Clear overflow flag
597.     DJNZ  R7, DELAY_ONE_SEC ; Decrement R7 and loop if not zero
598.     MOV   R7, #15        ; Reset counter
599. ;SJMP  DELAY_LOOP    ; (Commented out)
600. RET
601.
602. org 600h
603. ;00111001B
604. SEGMENT_PATTERNS:DB 3FH,06H,05BH,04FH,066H,06DH, 07DH,07H,07FH,06FH, 077H,07CH,039H,05EH,079H,071H,3FH
605.
606. OVEN_START_TEXT: DB "OVEN STARTED",0
607.
608. GREATER_300_TEXT: DB "VALUE > 300",0
609.
610. BELOW_5_TEXT: DB "VALUE < 5",0
611.
612. ABOVE_60_TEXT: DB "VALUE > 60",0
613.
614. BELOW_60_TEXT: DB "VALUE < 60",0
615. FINISHED_TEXT: DB "OVEN STOPPED",0
616. RETRY_TEXT: DB "TRY AGAIN",0
617. FACT1_TEXT: DB "Cats love naps",0
618. FACT2_TEXT: DB "Ctrl+Z saves",0
619. FACT3_TEXT: DB "DC > MARVEL",0
620. FACT4_TEXT: DB "VR feels real",0
621. FACT5_TEXT: DB "Linux is free",0
622. FACT6_TEXT: DB "Dark mode saves",0
623. FACT7_TEXT: DB "IUT food great",0
624. FACT8_TEXT: DB "Frogs freeze",0
625. FACT9_TEXT: DB "DC > MARVEL",0
626. FACT10_TEXT: DB "Clock ticking",0
627. FACT11_TEXT: DB "Batman = goat",0
628.
629. MY_FACT_TEXT: DB "SHAFIN FOUND ME",0
630.
631. END

```