



ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
ORGANISATION OF ISLAMIC COOPERATION (OIC)

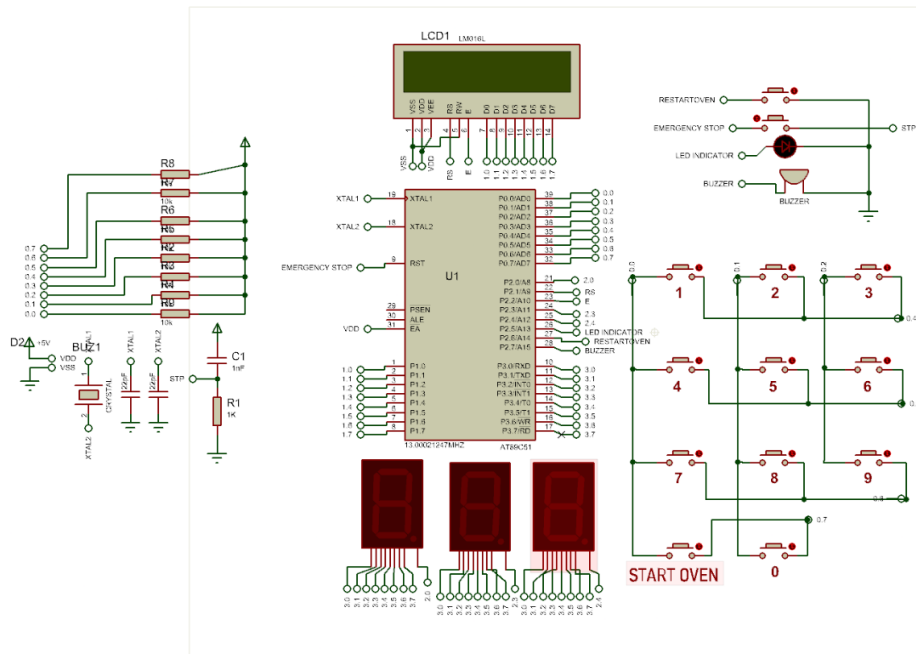
DEPARTMENT OF ELECTRICAL AND ELECTRONIC
ENGINEERING

NAME : K. M. Sirazul Monir STUDENT ID : 200021247
DEPARTMENT : EEE SECTION : B

DATE OF SUBMISSION : 18/3/2025

COURSE NO.	: EEE 4705
COURSE TITLE	: Microcontroller Based System Design
ASSIGNMENT NO	: 01
ASSIGNMENT NAME	: Complex Engineering Problem

Proteus schematic Screenshot



Jurgen Smart Oven Prototype

Based on AT89S52/AT89C51 microcontroller

Features:

1. **User-Defined Timer** – Allows users to set cooking time from **5 to 300 seconds** using a **key-pad**.
2. **Rejecting Out of range Input** – Rejects inputs **more than 300s and less than 5s**.
3. **Countdown Display** – Displays the countdown on **three 7-segment displays (7SDs)**.
4. **LCD Message Display** – Displays different messages based on the set time
 - a. **Time<60s** : Shows a fixed message.
 - b. **Time>60s** : Shows Random facts about Food and cooking.
5. **Buzzer Notification** – The **buzzer** sounds when the countdown reaches zero.
6. **LED Status Indication** – LEDs indicate the oven's working state.
7. **Start Button** – Begins the countdown once the user inputs time.
8. **Restart Button** – The oven features a **Restart Button** after operation is complete.
9. **Emergency Stop Button** – Instantly stops the oven in any situation.

The oven provides an engaging user experience while it is operating.

Frequency Of the microcontroller = $11 + (22 - 12) * (\text{student ID} / 10^9)$ Mhz = **13.00021247MHZ**

Edit Component

Part Reference: Hidden: ☐

Part Value: Hidden: ☐

Element:

PCB Package:

Program File:

Clock Frequency:

Advanced Properties:

Other Properties:

☐ Exclude from Simulation ☐ Attach hierarchy module

☐ Exclude from PCB Layout ☐ Hide common pins

☐ Exclude from Current Variant ☐ Edit all properties as text

CODE

```
1. ; Original Jurgen oven code
2. ; All rights reserved by :
3. ; :::::::::::::::::::::::::::::: K. M. Sirazul Monir ::::::::::::::::::::::::::::::
4.
5.
6.
7.
8.     org 0000h
9.
10.    ; Initialize all ports to default state
11. INITIALIZE: MOV P3, #00000000B ; Clear Port 3
12. MOV P0, #0FEH ; Set up keypad scanning
13. MOV 30H, #0 ; Clear memory variables
14. MOV 32H, #0
15. MOV R0, #0 ; Reset register counters
16. MOV R7, #15 ; Set timer constant
17. MOV R5, #00H ; Clear fact counter
18. MOV 69H, 0H ; Clear memory location 69H
19. CLR P2.7 ; Turn off buzzer
20. MOV P1, #00000000B ; Initialize display port
21. MOV TMOD, #11H ; Set timer mode
22. MOV TH1, #3CH ; Initialize timer high byte
23. MOV TL1, #98H ; Initialize timer low byte
24. SETB TR1 ; Start timer
25. CLR P2.5 ; Turn off heating element
26.
27. ; Initialize registers for various operations
28. REGISTER_INIT:
29. MOV R3, #00H ; Clear display register
30. MOV R1, #00H ; Clear memory pointer
31. MOV R2, #00H ; Clear general purpose register
32.
33.
34. ; Define LCD interface pins
35. IO_DEFINITION:
36. RS EQU P2.1 ; Register Select pin for LCD
37. EN EQU P2.2 ; Enable pin for LCD
38.
39.
40. ; ::::::::::::::::::::::::::::::::::::::
41. ; Initialize LCD with standard commands
42. LCD_INIT:
43. MOV R3, #38H ; Function set: 8-bit, 2 lines, 5x7 font
44. ACALL COMMAND ; Send command to LCD
45. MOV R3, #0EH ; Display on, cursor on
46. ACALL COMMAND
47. MOV R3, #80H ; Set cursor to beginning of first line
48. ACALL COMMAND
49. MOV R3, #01H ; Clear display
50. ACALL COMMAND
51.
52. ; ::::::::::::::::::::::::::::::::::::::
53. ; Display "ENTER TIME IN s:" message
54. PROMPT_TIME_ENTRY: MOV DPTR, #TIME_PROMPT
55. DISPLAY_PROMPT: MOV A, #00H
56. MOVC A, @A+DPTR
57. JZ TIME_INPUT_LOOP ; Jump if end of message (zero terminator)
58. MOV R3, A
59. ACALL DISPLAY ; Display character
60. INC DPTR
61. LJMP DISPLAY_PROMPT
62.
63.
64. ; wait for first digit input from keypad
65. TIME_INPUT_LOOP: LCALL SCAN
66. MOV A, R0
67. JZ TIME_INPUT_LOOP ; If no key pressed, keep scanning
68.
69. MOV 40H, A ; Store first digit (hundreds place)
70. ANL 40H, #00001111B ; Mask upper bits to get digit value only
71. MOV R1, #40H
72. CJNE @R1, #0AH, CHECK_KEY_B ; Check if valid digit (not A)
73. SJMP TIME_INPUT_LOOP
74. CHECK_KEY_B: CJNE @R1, #0BH, CHECK_KEY_C ; Check if not B
75. SJMP TIME_INPUT_LOOP
76. CHECK_KEY_C: CJNE @R1, #0CH, CHECK_KEY_D ; Check if not C
77. SJMP TIME_INPUT_LOOP
78. CHECK_KEY_D: CJNE @R1, #0DH, CHECK_KEY_E ; Check if not D
79. SJMP TIME_INPUT_LOOP
80. CHECK_KEY_E: CJNE @R1, #0EH, CHECK_KEY_F ; Check if not E
81. SJMP TIME_INPUT_LOOP
82. CHECK_KEY_F: CJNE @R1, #0FH, SHOW_DIGIT1 ; Check if not F
83. SJMP TIME_INPUT_LOOP
84.
85. ; Display first digit on LCD
86. SHOW_DIGIT1: MOV R3, #0C0H ; Set cursor to second line
87. ACALL COMMAND
88. MOV A, 40H
89. ADD A, #30H ; Convert digit to ASCII
90. MOV R3, A
91. ACALL DISPLAY
92. lcall SHORT_DELAY ; Small delay between keypresses
93.
94.
95. ; wait for second digit input from keypad
96. TENS_DIGIT_INPUT: LCALL SCAN
```

```

97.     MOV A,R0
98.     JZ TENS_DIGIT_INPUT      ; If no key pressed, keep scanning
99.     MOV 44H,A                ; Store second digit (tens place)
100.    ANL 44H,#00001111B       ; Mask upper bits
101.
102.
103.     MOV R1,#44H
104.     CJNE @R1,#0AH,CHECK_TENS_B ; Check if valid digit (not A)
105.     SJMP TENS_DIGIT_INPUT
106. CHECK_TENS_B: CJNE @R1,#0BH,CHECK_TENS_C ; Check if not B
107.     SJMP TENS_DIGIT_INPUT
108. CHECK_TENS_C: CJNE @R1,#0CH,CHECK_TENS_D ; Check if not C
109.     SJMP TENS_DIGIT_INPUT
110. CHECK_TENS_D: CJNE @R1,#0DH,CHECK_TENS_E ; Check if not D
111.     SJMP TENS_DIGIT_INPUT
112. CHECK_TENS_E: CJNE @R1,#0EH,CHECK_TENS_F ; Check if not E
113.     SJMP TENS_DIGIT_INPUT
114. CHECK_TENS_F: CJNE @R1,#0FH,SHOW_DIGIT2 ; Check if not F
115.     SJMP TENS_DIGIT_INPUT
116.
117.
118. ; Display second digit on LCD
119. SHOW_DIGIT2: MOV A,44H
120.     ADD A,#30H                ; Convert digit to ASCII
121.     MOV R3,A
122.     ACALL DISPLAY
123.     lcall SHORT_DELAY         ; Small delay between keypresses
124.
125.
126. ; wait for third digit input from keypad
127. ONES_DIGIT_INPUT: LCALL SCAN
128.     MOV A,R0
129.     JZ ONES_DIGIT_INPUT      ; If no key pressed, keep scanning
130.     MOV 53H,A                ; Store third digit (ones place)
131.     ANL 53H,#00001111B       ; Mask upper bits
132.
133.
134.     MOV R1,#53H
135.     CJNE @R1,#0AH,CHECK_ONES_B ; Check if valid digit (not A)
136.     SJMP ONES_DIGIT_INPUT
137. CHECK_ONES_B: CJNE @R1,#0BH,CHECK_ONES_C ; Check if not B
138.     SJMP ONES_DIGIT_INPUT
139. CHECK_ONES_C: CJNE @R1,#0CH,CHECK_ONES_D ; Check if not C
140.     SJMP ONES_DIGIT_INPUT
141. CHECK_ONES_D: CJNE @R1,#0DH,CHECK_ONES_E ; Check if not D
142.     SJMP ONES_DIGIT_INPUT
143. CHECK_ONES_E: CJNE @R1,#0EH,CHECK_ONES_F ; Check if not E
144.     SJMP ONES_DIGIT_INPUT
145. CHECK_ONES_F: CJNE @R1,#0FH,SHOW_DIGIT3 ; Check if not F
146.     SJMP ONES_DIGIT_INPUT
147.
148.
149. ; Display third digit on LCD
150. SHOW_DIGIT3: MOV A,53H
151.     ADD A,#30H                ; Convert digit to ASCII
152.     MOV R3,A
153.     ACALL DISPLAY
154.     lcall SHORT_DELAY         ; Small delay between keypresses
155.
156.
157. ; wait for START key (F key)
158. WAIT_FOR_START: LCALL SCAN
159.     MOV A,R0
160.     ANL A,#00001111B
161.     CJNE A,#0FH,WAIT_FOR_START ; Keep waiting until F key is pressed
162.
163.
164. ; Calculate total time in seconds from the three digits entered
165.     MOV A,44H                ; Get tens digit
166.     MOV B,A
167.     MOV A,#10
168.     MUL AB                    ; Multiply tens digit by 10
169.     ADD A,53H                ; Add ones digit
170.     MOV 60H,A                ; Store tens + ones value
171.
172.
173.     MOV A,40H                ; Get hundreds digit
174.     MOV B,A
175.     MOV A,#100
176.     MUL AB                    ; Multiply hundreds digit by 100
177.     MOV 62H,A                ; Store low byte of result
178.     MOV A,B
179.     MOV 61H,A                ; Store high byte of result
180.     MOV A,62H
181.     ADD A,60H                ; Add (tens + ones) to (hundreds * 100)
182.     MOV 62H,A
183.     JNC CHECK_MAX_TIME       ; Check if carry occurred during addition
184.     INC 61H                  ; If carry, increment high byte
185.
186.
187. ; Check if time exceeds 300 seconds (maximum allowed)
188. CHECK_MAX_TIME: MOV A,61H    ; Load high byte into accumulator
189.     CJNE A,#01H,CHECK_UPPER_BYTE ; Compare with 01H (300 > 256)
190.     MOV A,62H                ; If high byte is 01H, check low byte
191.     CJNE A,#2DH,CHECK_LOWER_BYTE ; Compare with 2DH (45 decimal, 256+45=301)
192.     JMP TIME_OVER_300        ; If equal to 300 exactly, time is too large
193.
194. CHECK_UPPER_BYTE: JC CHECK_MIN_TIME ; If high byte < 01H, time is < 256
195.     JMP TIME_OVER_300        ; If high byte > 01H, time is > 300
196.
197. CHECK_LOWER_BYTE: JC CHECK_MIN_TIME ; If low byte < 2DH, time might be valid
198.     JMP TIME_OVER_300        ; If low byte > 2DH, time is > 300
199.
200. ; Check if time is less than 5 seconds (minimum allowed)
201.
202. CHECK_MIN_TIME: MOV A,61H    ; Load high byte into accumulator
203.     JNZ CHECK_MID_TIME       ; If high byte > 0, time is > 255
204.     MOV A,62H                ; Load low byte into accumulator

```

```

205.         CJNE A,#05H,CHECK_MIN_TIME_TEMP ; Compare with 5 seconds
206.         JMP CHECK_MID_TIME               ; If exactly 5, proceed to mid check
207.
208. CHECK_MIN_TIME_TEMP: JC TIME_UNDER_5    ; If Carry is set, time is < 5
209.         JMP CHECK_MID_TIME               ; If Carry is not set, time is > 5
210.
211. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
212.
213. ; Display error for time less than 5 seconds
214. TIME_UNDER_5: MOV DPTR,#TIME_TOO_SHORT_MSG
215.         MOV R3, #01H                     ; Clear display and set cursor to first position
216.         ACALL COMMAND
217. DISPLAY_SHORT_TIME_ERROR:MOV A,#00H
218.         MOVC A,@A+DPTR
219.         JZ SHOW_RETRY_MESSAGE
220.         MOV R3,A
221.         ACALL DISPLAY
222.         INC DPTR
223.         LJMP DISPLAY_SHORT_TIME_ERROR
224.
225. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
226.
227. ; Show retry message on second line
228. SHOW_RETRY_MESSAGE: MOV R3, #0C0H        ; Set cursor to second line
229.         ACALL COMMAND
230.         MOV DPTR,#RETRY_MESSAGE
231. DISPLAY_RETRY:MOV A,#00H
232.         MOVC A,@A+DPTR
233.         JZ RETRY_DELAY
234.         MOV R3,A
235.         ACALL DISPLAY
236.         INC DPTR
237.         LJMP DISPLAY_RETRY
238.
239. RETRY_DELAY: LCALL LONG_DELAY             ; Wait before restarting
240.         LJMP INITIALIZE                   ; Restart the program
241.
242. ; Check if time is more than 60 seconds (cooking method selection)
243. CHECK_MID_TIME: MOV A,61H                 ; Load high byte into accumulator
244.         JNZ TIME_OVER_60                 ; If high byte > 0, time is > 255 > 60
245.         MOV A,62H                         ; Load low byte into accumulator
246.         CJNE A,#3CH,CHECK_60_TEMP        ; Compare with 60 (3CH) seconds
247.         JMP TIME_OVER_60                 ; If exactly 60, consider it > 60
248.
249. CHECK_60_TEMP: JC TIME_UNDER_60          ; If Carry is set, time is < 60
250.         JMP TIME_OVER_60                 ; If Carry is not set, time is > 60
251.
252. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
253.
254. ; Handle time > 60 seconds cooking mode
255. TIME_OVER_60: MOV DPTR,#TIME_OVER_60_MSG
256.         MOV R3, #01H                     ; Clear display
257.         ACALL COMMAND
258. DISPLAY_OVER_60:MOV A,#00H
259.         MOVC A,@A+DPTR
260.         JZ START_OVEN_MESSAGE_2
261.         MOV R3,A
262.         ACALL DISPLAY
263.         INC DPTR
264.         LJMP DISPLAY_OVER_60
265.
266. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
267. ; Handle time < 60 seconds cooking mode
268. TIME_UNDER_60: MOV DPTR,#TIME_UNDER_60_MSG
269.         MOV R3, #01H                     ; Clear display
270.         ACALL COMMAND
271. DISPLAY_UNDER_60:MOV A,#00H
272.         MOVC A,@A+DPTR
273.         JZ START_OVEN_MESSAGE_1
274.         MOV R3,A
275.         ACALL DISPLAY
276.         INC DPTR
277.         LJMP DISPLAY_UNDER_60
278.
279. ; Display error for time > 300 seconds
280. TIME_OVER_300: MOV DPTR,#TIME_OVER_300_MSG
281.         MOV R3, #01H                     ; Clear display
282.         ACALL COMMAND
283. DISPLAY_OVER_300:MOV A,#00H
284.         MOVC A,@A+DPTR
285.         JZ SHOW_RETRY_MESSAGE
286.         MOV R3,A
287.         ACALL DISPLAY
288.         INC DPTR
289.         LJMP DISPLAY_OVER_300
290.
291. ; Show "OVEN STARTED" message for mode 2
292. START_OVEN_MESSAGE_2:MOV DPTR,#OVEN_STARTED_MSG
293.         MOV R3, #0C0H                     ; Set cursor to second line
294.         ACALL COMMAND
295. DISPLAY_START_2:MOV A,#00H
296.         MOVC A,@A+DPTR
297.         JZ COOKING_LOOP_2
298.         MOV R3,A
299.         ACALL DISPLAY
300.         INC DPTR
301.         LJMP DISPLAY_START_2
302.
303. ; Show "OVEN STARTED" message for mode 1
304. START_OVEN_MESSAGE_1:MOV DPTR,#OVEN_STARTED_MSG
305.         MOV R3, #0C0H                     ; Set cursor to second line
306.         ACALL COMMAND
307. DISPLAY_START_1:MOV A,#00H
308.         MOVC A,@A+DPTR
309.         JZ COOKING_LOOP_1
310.         MOV R3,A
311.         ACALL DISPLAY
312.

```

```

313.     INC DPTR
314.     LJMP DISPLAY_START_1
315.
316. ; Main cooking loop for mode 2 (higher power)
317. COOKING_LOOP_2: MOV     R6, #20          ; Initialize loop counter
318.     SETB P2.5                ; Turn on heating element
319. COOKING_COUNTDOWN_2:
320.     LCALL DELAY_1S            ; wait for 1 second
321.     LCALL DECREMENT_TIMER    ; Update the countdown
322.     DJNZ R6, COOKING_COUNTDOWN_2 ; Loop until counter expires
323.     LCALL UPDATE_DISPLAY_1    ; Update 7-segment display 1
324.     LCALL UPDATE_DISPLAY_2    ; Update 7-segment display 2
325.     LCALL UPDATE_DISPLAY_3    ; Update 7-segment display 3
326.     LCALL DISPLAY_RANDOM_FACT ; Show a random cooking fact
327.     MOV     R6, #20          ; Reset loop counter
328.     SJMP COOKING_COUNTDOWN_2 ; Continue cooking loop
329.
330. ; Main cooking loop for mode 1 (lower power)
331. COOKING_LOOP_1: LCALL LONG_DELAY ; Small initial delay
332.     SETB P2.5                ; Turn on heating element
333. DISPLAY_COOKING_TIP: LCALL DELAY_1S ; wait for 1 second
334.     LCALL DECREMENT_TIMER    ; Update the countdown
335.
336.     ; LCALL UPDATE_DISPLAY_1    ; Uncomment if using 7-segment displays
337.     ; LCALL UPDATE_DISPLAY_2    ; in mode 1
338.     ; LCALL UPDATE_DISPLAY_3
339.     SJMP DISPLAY_COOKING_TIP ; Continue cooking loop
340.
341.
342. ; Display cooking tip for mode 1
343. DISPLAY_COOKING_TIP_TEXT:
344.     MOV DPTR, #QUICK_COOK_TIP
345.     MOV R3, #01H              ; Clear display
346.     ACALL COMMAND
347. DISPLAY_TIP_LOOP: MOV A, #00H
348.     MOVC A, @A+DPTR
349.     JZ DISPLAY_TIP_END
350.     MOV R3, A
351.     ACALL DISPLAY
352.     INC DPTR
353.     LJMP DISPLAY_TIP_LOOP
354. DISPLAY_TIP_END:
355.     RET
356.
357.
358. ; Display rotating facts for mode 2
359. DISPLAY_COOKING_FACT:
360.     MOV DPTR, #DEFAULT_FACT
361.     MOV R3, #01H              ; Clear display
362.     ACALL COMMAND
363. DISPLAY_FACT_LOOP: MOV A, #00H
364.     MOVC A, @A+DPTR
365.     JZ DISPLAY_FACT_END
366.     MOV R3, A
367.     ACALL DISPLAY
368.     INC DPTR
369.     LJMP DISPLAY_FACT_LOOP
370. DISPLAY_FACT_END:
371.     RET
372.
373. ;-----
374. ; Display a random cooking fact from the fact library
375. DISPLAY_RANDOM_FACT: INC R5
376.     MOV A, TL1                ; Get a semi-random value from timer
377.
378.     ADD A, R5                  ; Combine with counter for better randomness
379.     ANL A, #00001111B
380.     MOV R5, A
381.     CJNE R5, #01H, CHECK_FACT_2 ; Check which fact to display
382.     MOV DPTR, #FACT_2_TEXT
383.     MOV R3, #01H              ; Clear display
384.     ACALL COMMAND
385. DISPLAY_FACT_2_LOOP: MOV A, #00H
386.     MOVC A, @A+DPTR
387.     JZ DISPLAY_RANDOM_FACT_END_TEMP
388.     MOV R3, A
389.     ACALL DISPLAY
390.
391.     INC DPTR
392.     LJMP DISPLAY_FACT_2_LOOP
393.
394. CHECK_FACT_2:
395.     CJNE R5, #02H, CHECK_FACT_3 ; Check for fact 3
396.     MOV DPTR, #FACT_3_TEXT
397.     MOV R3, #01H              ; Clear display
398.     ACALL COMMAND
399. DISPLAY_FACT_3_LOOP: MOV A, #00H
400.     MOVC A, @A+DPTR
401.     JZ DISPLAY_RANDOM_FACT_END_TEMP
402.     MOV R3, A
403.     ACALL DISPLAY
404.     INC DPTR
405.     LJMP DISPLAY_FACT_3_LOOP
406.
407.
408. CHECK_FACT_3:
409.     CJNE R5, #03H, CHECK_FACT_4 ; Check for fact 4
410.     MOV DPTR, #FACT_4_TEXT
411.     MOV R3, #01H              ; Clear display
412.     ACALL COMMAND
413. DISPLAY_FACT_4_LOOP: MOV A, #00H
414.     MOVC A, @A+DPTR
415.     JZ DISPLAY_RANDOM_FACT_END_TEMP
416.     MOV R3, A
417.     ACALL DISPLAY
418.     INC DPTR
419.     LJMP DISPLAY_FACT_4_LOOP
420.

```

```

421.
422. CHECK_FACT_4:
423.     CJNE R5,#04H,CHECK_FACT_5           ; check for fact 5
424.     MOV DPTR,#FACT_5_TEXT
425.     MOV R3,#01H                         ; clear display
426.     ACALL COMMAND
427. DISPLAY_FACT_5_LOOP:MOV A,#00H
428.     MOVC A,@A+DPTR
429.     JZ DISPLAY_RANDOM_FACT_END_TEMP
430.     MOV R3,A
431.     ACALL DISPLAY
432.     INC DPTR
433.     LJMP DISPLAY_FACT_5_LOOP
434.
435.
436. CHECK_FACT_5:
437.     CJNE R5,#05H,CHECK_FACT_6           ; check for fact 6
438.     MOV DPTR,#FACT_6_TEXT
439.     MOV R3,#01H                         ; clear display
440.     ACALL COMMAND
441. DISPLAY_FACT_6_LOOP:MOV A,#00H
442.     MOVC A,@A+DPTR
443.     JZ DISPLAY_RANDOM_FACT_END_TEMP
444.     MOV R3,A
445.     ACALL DISPLAY
446.     INC DPTR
447.     LJMP DISPLAY_FACT_6_LOOP
448.
449. DISPLAY_RANDOM_FACT_END_TEMP: LJMP DISPLAY_RANDOM_FACT_END
450.
451. CHECK_FACT_6:
452.     CJNE R5,#06H,CHECK_FACT_7           ; check for fact 7
453.     MOV DPTR,#FACT_7_TEXT
454.     MOV R3,#01H                         ; clear display
455.     ACALL COMMAND
456. DISPLAY_FACT_7_LOOP:MOV A,#00H
457.     MOVC A,@A+DPTR
458.     JZ DISPLAY_RANDOM_FACT_END
459.     MOV R3,A
460.     ACALL DISPLAY
461.     INC DPTR
462.     LJMP DISPLAY_FACT_7_LOOP
463.
464. CHECK_FACT_7:
465.     CJNE R5,#07H,CHECK_FACT_8           ; check for fact 8
466.     MOV DPTR,#FACT_8_TEXT
467.     MOV R3,#01H                         ; clear display
468.     ACALL COMMAND
469. DISPLAY_FACT_8_LOOP:MOV A,#00H
470.     MOVC A,@A+DPTR
471.     JZ DISPLAY_RANDOM_FACT_END
472.     MOV R3,A
473.     ACALL DISPLAY
474.     INC DPTR
475.     LJMP DISPLAY_FACT_8_LOOP
476.
477.
478. CHECK_FACT_8:
479.     CJNE R5,#08H,CHECK_FACT_9           ; check for fact 9
480.     MOV DPTR,#FACT_9_TEXT
481.     MOV R3,#01H                         ; clear display
482.     ACALL COMMAND
483. DISPLAY_FACT_9_LOOP:MOV A,#00H
484.     MOVC A,@A+DPTR
485.     JZ DISPLAY_RANDOM_FACT_END
486.     MOV R3,A
487.     ACALL DISPLAY
488.     INC DPTR
489.     LJMP DISPLAY_FACT_9_LOOP
490.
491.
492. CHECK_FACT_9:
493.     CJNE R5,#09H,CHECK_FACT_10          ; check for fact 10
494.     MOV DPTR,#FACT_10_TEXT
495.     MOV R3,#01H                         ; clear display
496.     ACALL COMMAND
497. DISPLAY_FACT_10_LOOP:MOV A,#00H
498.     MOVC A,@A+DPTR
499.     JZ DISPLAY_RANDOM_FACT_END
500.     MOV R3,A
501.     ACALL DISPLAY
502.     INC DPTR
503.     LJMP DISPLAY_FACT_10_LOOP
504.
505.
506. CHECK_FACT_10:
507.     CJNE R5,#10H,RESET_FACT_COUNTER      ; check for fact 11 or reset
508.     MOV DPTR,#FACT_11_TEXT
509.     MOV R3,#01H                         ; clear display
510.     ACALL COMMAND
511. DISPLAY_FACT_11_LOOP:MOV A,#00H
512.     MOVC A,@A+DPTR
513.     JZ DISPLAY_RANDOM_FACT_END
514.     MOV R3,A
515.     ACALL DISPLAY
516.     INC DPTR
517.     LJMP DISPLAY_FACT_11_LOOP
518.
519.
520. RESET_FACT_COUNTER: MOV R5,#0H           ; Reset fact counter and start again
521.     LJMP DISPLAY_RANDOM_FACT
522.
523. DISPLAY_RANDOM_FACT_END:
524.     RET
525.
526. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;-----
527. ; Decrement the cooking timer by 1 second
528. DECREMENT_TIMER:

```

```

529.      DEC 53H                      ; Decrement ones digit
530.      MOV A, 53H
531.
532.      CJNE A, #11111111B, CONTINUE_TIMER ; Check for underflow (FF)
533.
534.      ; Reset ones digit and decrement tens digit
535.      MOV 53H, #9
536.      DEC 44H
537.      MOV A, 44H
538.
539.      CJNE A, #11111111B, CONTINUE_TIMER
540.
541.      ; Reset tens digit and decrement hundreds digit
542.      MOV 44H, #9
543.      DEC 40H
544.      MOV A, 40H
545.
546.      CJNE A, #11111111B, CONTINUE_TIMER
547.
548.      LJMP COOKING_FINISHED          ; Timer has expired
549.
550. CONTINUE_TIMER:
551. RET
552.
553. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
554. ; Handle cooking completion
555. COOKING_FINISHED: CLR P2.5          ; Turn off heating element
556.      MOV R3, #01H                  ; Clear display
557.      ACALL COMMAND
558.      MOV DPTR, #COOKING_FINISHED_MSG
559. DISPLAY_FINISHED: MOV A, #00H
560.      MOVC A, @A+DPTR
561.      JZ SOUND_BUZZER
562.      MOV R3, A
563.      ACALL DISPLAY
564.      INC DPTR
565.      LJMP DISPLAY_FINISHED
566.
567. ; Sound buzzer to indicate cooking completion
568. SOUND_BUZZER: SETB P2.7             ; Turn on buzzer
569.      LCALL LONG_DELAY              ; Keep buzzer on for delay period
570.      CLR P2.7                     ; Turn off buzzer
571.
572. ; wait for reset button press
573. WAIT_FOR_RESET: JB P2.6, WAIT_FOR_RESET ; wait until reset button pressed
574.      LJMP INITIALIZE              ; Reset the system
575.
576. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
577.
578. ; LCD character display subroutine
579. DISPLAY:
580.      MOV P1, R3                    ; Send character data to port
581.      SETB RS                        ; Set Register Select for data
582.      SETB EN                        ; Enable pulse high
583.      CLR EN                         ; Enable pulse low
584.      ACALL DELAY                   ; Small delay
585.      RET
586.
587. ; LCD command subroutine
588. COMMAND:
589.      MOV P1, R3                    ; Send command data to port
590.      CLR RS                         ; Clear Register Select for command
591.      SETB EN                        ; Enable pulse high
592.      CLR EN                         ; Enable pulse low
593.      ACALL DELAY                   ; Small delay
594.      RET
595.
596. ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
597. ; keypad scanning subroutine
598. SCAN:
599. SCAN_START:
600.      JNB P0.0, COLUMN_1           ; Check column 1
601.      JNB P0.1, COLUMN_2           ; Check column 2
602.      JNB P0.2, COLUMN_3           ; Check column 3
603.      JNB P0.3, COLUMN_4           ; Check column 4
604.      SJMP EXIT_SCAN
605. COLUMN_1:
606.      JNB P0.4, KEY_1               ; Check for key 1
607.      JNB P0.5, KEY_4               ; Check for key 4
608.      JNB P0.6, KEY_7               ; Check for key 7
609.      JNB P0.7, JUMP_TO_KEY_F      ; Check for key F
610.      SETB P0.0
611.      CLR P0.1
612.      SJMP EXIT_SCAN
613. COLUMN_2:
614.      JNB P0.4, KEY_2               ; Check for key 2
615.      JNB P0.5, KEY_5               ; Check for key 5
616.      JNB P0.6, KEY_8               ; Check for key 8
617.      JNB P0.7, KEY_0               ; Check for key 0
618.      SETB P0.1
619.      CLR P0.2
620.      SJMP EXIT_SCAN
621. COLUMN_3:
622.      JNB P0.4, KEY_3               ; Check for key 3
623.      JNB P0.5, KEY_6               ; Check for key 6
624.      JNB P0.6, KEY_9               ; Check for key 9
625.      JNB P0.7, JUMP_TO_KEY_E      ; Check for key E
626.      SETB P0.2
627.      CLR P0.3
628.      SJMP EXIT_SCAN
629. COLUMN_4:
630.      JNB P0.4, JUMP_TO_KEY_A       ; Check for key A
631.      JNB P0.5, JUMP_TO_KEY_B       ; Check for key B
632.      JNB P0.6, JUMP_TO_KEY_C       ; Check for key C
633.      JNB P0.7, JUMP_TO_KEY_D       ; Check for key D
634.      SETB P0.3
635.      CLR P0.0
636.      LJMP EXIT_SCAN

```



```

637. EXIT_SCAN:
638.     RET
639.
640. ; Jump tables for key handling (to handle branch distance limitations)
641. JUMP_TO_KEY_A: LJMP KEY_A
642. JUMP_TO_KEY_B: LJMP KEY_B
643. JUMP_TO_KEY_C: LJMP KEY_C
644. JUMP_TO_KEY_D: LJMP KEY_D
645. JUMP_TO_KEY_E: LJMP KEY_E
646. JUMP_TO_KEY_F: LJMP KEY_F
647.
648. ; Key handler routines
649. KEY_0:
650.     MOV     R0, #16D           ; Store keycode for key 0
651.     LJMP    SCAN_START
652. KEY_1:
653.     MOV     R0, #1D           ; Store keycode for key 1
654.     LJMP    SCAN_START
655. KEY_2:
656.     MOV     R0, #2D           ; Store keycode for key 2
657.     LJMP    SCAN_START
658. KEY_3:
659.     MOV     R0, #3D           ; Store keycode for key 3
660.     LJMP    SCAN_START
661. KEY_4:
662.     MOV     R0, #4D           ; Store keycode for key 4
663.     LJMP    SCAN_START
664. KEY_5:
665.     MOV     R0, #5D           ; Store keycode for key 5
666.     LJMP    SCAN_START
667. KEY_6:
668.     MOV     R0, #6D           ; Store keycode for key 6
669.     LJMP    SCAN_START
670. KEY_7:
671.     MOV     R0, #7D           ; Store keycode for key 7
672.     LJMP    SCAN_START
673. KEY_8:
674.     MOV     R0, #8D           ; Store keycode for key 8
675.     LJMP    SCAN_START
676. KEY_9:
677.     MOV     R0, #9D           ; Store keycode for key 9
678.     LJMP    SCAN_START
679. KEY_A:
680.     MOV R0, #10               ; Store keycode for key A
681.     LJMP SCAN_START
682. KEY_B:
683.     MOV R0, #11               ; Store keycode for key B
684.     LJMP SCAN_START
685. KEY_C:
686.     MOV R0, #12               ; Store keycode for key C
687.     LJMP SCAN_START
688. KEY_D:
689.     MOV R0, #13               ; Store keycode for key D
690.     LJMP SCAN_START
691. KEY_E:
692.     MOV R0, #14               ; Store keycode for key E
693.     LJMP SCAN_START
694. KEY_F:
695.     MOV R0, #15               ; Store keycode for key F (START key)
696.     LJMP SCAN_START
697.
698. ; Update the first 7-segment display (hundreds place)
699. UPDATE_DISPLAY_1: CLR P2.0    ; Select first display
700.     MOV A, 30H
701.     JNZ DISPIDONE
702.
703.     MOV     A, 40h             ; Get hundreds digit
704.     MOV     dptr, #SEGMENT_PATTERNS ; Look up display pattern
705.     MOVC    A, @a+dptr
706.     MOV     P3, A             ; Output to port
707.     LCALL   DISPLAY_DELAY     ; Short delay
708.     MOV     P3, #00H          ; Clear the display
709.     SETB    P2.0              ; Deselect first display
710.     RET
711.
712. ; Update the second 7-segment display (tens place)
713. UPDATE_DISPLAY_2: CLR P2.3    ; Select second display
714.     MOV A, 30H
715.     JNZ DISPIDONE
716.
717.     MOV     A, 44h             ; Get tens digit
718.     MOV     dptr, #SEGMENT_PATTERNS ; Look up display pattern
719.     MOVC    A, @a+dptr
720.     MOV     P3, A             ; Output to port
721.     LCALL   DISPLAY_DELAY     ; Short delay
722.     MOV     P3, #00H          ; Clear the display
723.
724.     SETB    P2.3              ; Deselect second display
725.     RET
726.
727. ; Update the third 7-segment display (ones place)
728. UPDATE_DISPLAY_3: CLR P2.4    ; Select third display
729.     MOV A, 30H
730.     JNZ DISPIDONE
731.
732.     MOV     A, 53h             ; Get ones digit
733.     MOV     dptr, #SEGMENT_PATTERNS ; Look up display pattern
734.     MOVC    A, @a+dptr
735.     MOV     P3, A             ; Output to port
736.     LCALL   DISPLAY_DELAY     ; Short delay
737.     MOV     P3, #00H          ; Clear the display
738.
739.     SETB    P2.4              ; Deselect third display
740.     RET
741.
742. ; Timing and delay subroutines
743. DISPLAY_DELAY: MOV     R1, #10 ; Short delay for display refresh
744. HERE2:     MOV     R2, #255    ; Inner loop counter

```

```

745. HERE:      DJNZ      R2, HERE          ; Decrement inner counter
746.          DJNZ      R1, HERE2         ; Decrement outer counter
747.          RET
748.
749.
750. DELAY:      MOV       R1, #50          ; Medium delay for LCD operations
751. HER2:      MOV       R2, #255         ; Inner loop counter
752. HER:      DJNZ      R2, HER           ; Decrement inner counter
753.          DJNZ      R1, HER2         ; Decrement outer counter
754.          RET
755.
756. ; Long delay (approximately 5 seconds)
757. LONG_DELAY: MOV R0, #10              ; Outer loop counter
758. HE3:      MOV R1, #255              ; Middle loop counter
759. HE2:      MOV R2, #255              ; Inner loop counter
760. HE:      DJNZ R2, HE                ; Decrement inner counter
761.          DJNZ R1, HE2              ; Decrement middle counter
762.          DJNZ R0, HE3              ; Decrement outer counter
763.          RET
764.
765. ; Short delay for keypad debounce
766. SHORT_DELAY: MOV R0, #2             ; Outer loop counter
767. SHE3:      MOV R1, #255              ; Middle loop counter
768. SHE2:      MOV R2, #255              ; Inner loop counter
769. SHE:      DJNZ R2, SHE              ; Decrement inner counter
770.          DJNZ R1, SHE2              ; Decrement middle counter
771.          DJNZ R0, SHE3              ; Decrement outer counter
772.          RET
773.
774. ; One second delay using timer 0
775. DELAY_1S:
776.     CLR TR0                          ; Stop Timer 0
777.     CLR TF0                          ; Clear Timer 0 overflow flag
778.                                     ; Timer 0 in 16-bit mode
779.
780.     MOV TH0, #3CH                    ; High byte of initial value
781.     MOV TL0, #98H                    ; Low byte of initial value
782.     SETB TR0                          ; Start Timer 0
783.
784. WAIT_FOR_TIMER:
785.     LCALL UPDATE_DISPLAY_1           ; Update display while waiting
786.     LCALL UPDATE_DISPLAY_2
787.     LCALL UPDATE_DISPLAY_3
788.     JNB TF0, WAIT_FOR_TIMER          ; wait until Timer 0 overflows
789.     CLR TR0                          ; Stop Timer 0
790.     CLR TF0                          ; Clear overflow flag
791.     DJNZ R7, DELAY_1S                ; Decrement counter and repeat if not zero
792.     MOV R7, #20                      ; Reset counter
793.     SJMP DELAY_LOOP                 ; Removed loop
794. RET
795.     org 900h
796. ; 7-segment display patterns (common cathode)
797. SEGMENT_PATTERNS: DB 3FH,06H,05BH,04FH,066H,06DH, 07DH,07H,07FH,06FH, 077H,07CH,039H,05EH,079H,071H,3FH
798.
799. ; Text strings for LCD display
800. TIME_PROMPT: DB "ENTER TIME IN s:",0
801. OVEN_STARTED_MSG: DB "OVEN STARTED",0
802.
803. TIME_OVER_300_MSG: DB "TIME>300s",0
804.
805. TIME_TOO_SHORT_MSG: DB "TIME<5s",0
806.
807. TIME_OVER_60_MSG: DB "TIME>60s",0
808.
809. TIME_UNDER_60_MSG: DB "TIME<60s",0
810. COOKING_FINISHED_MSG: DB "OVEN STOPPED",0
811. RETRY_MESSAGE: DB "TRY AGAIN",0
812. DEFAULT_FACT: DB "COOKING",0
813. FACT_2_TEXT: DB "HEATWAVE COMING!",0
814. FACT_3_TEXT: DB "ZAPPING DINNER!",0
815. FACT_4_TEXT: DB "TEMPS RISING...",0
816. FACT_5_TEXT: DB "KILLING GERMS!",0
817. FACT_6_TEXT: DB "FLAVOR LOADING...",0
818. FACT_7_TEXT: DB "GETTING TOASTY!",0
819. FACT_8_TEXT: DB "BAKING MAGIC...",0
820. FACT_9_TEXT: DB "ALMOST READY",0
821. FACT_10_TEXT: DB "BAKE@350C=SAFE!",0
822. FACT_11_TEXT: DB "LOADING FOOD...",0
823.
824. QUICK_COOK_TIP: DB "READY IN 1MIN!",0
825.
826. ;=====
827.     END

```