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

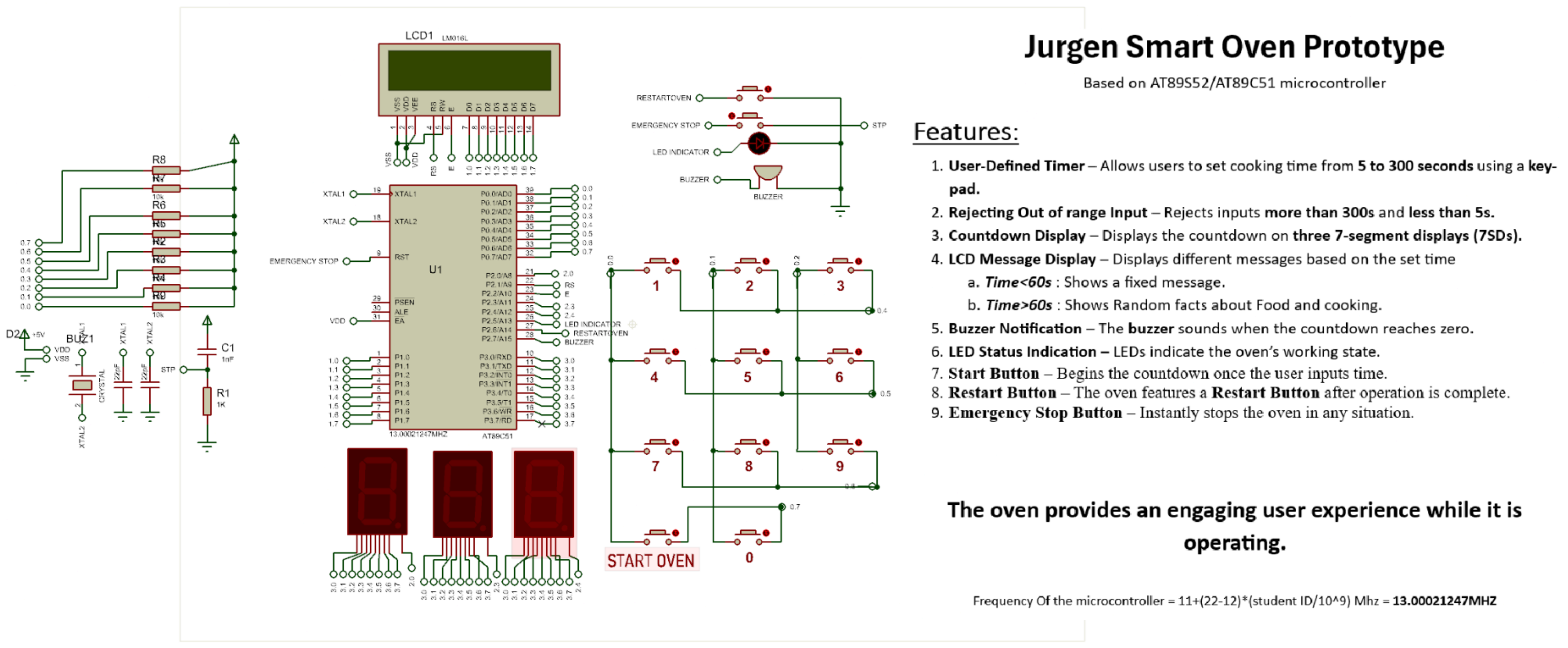
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

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

DATE OF SUBMISSION : 8/11/2024

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

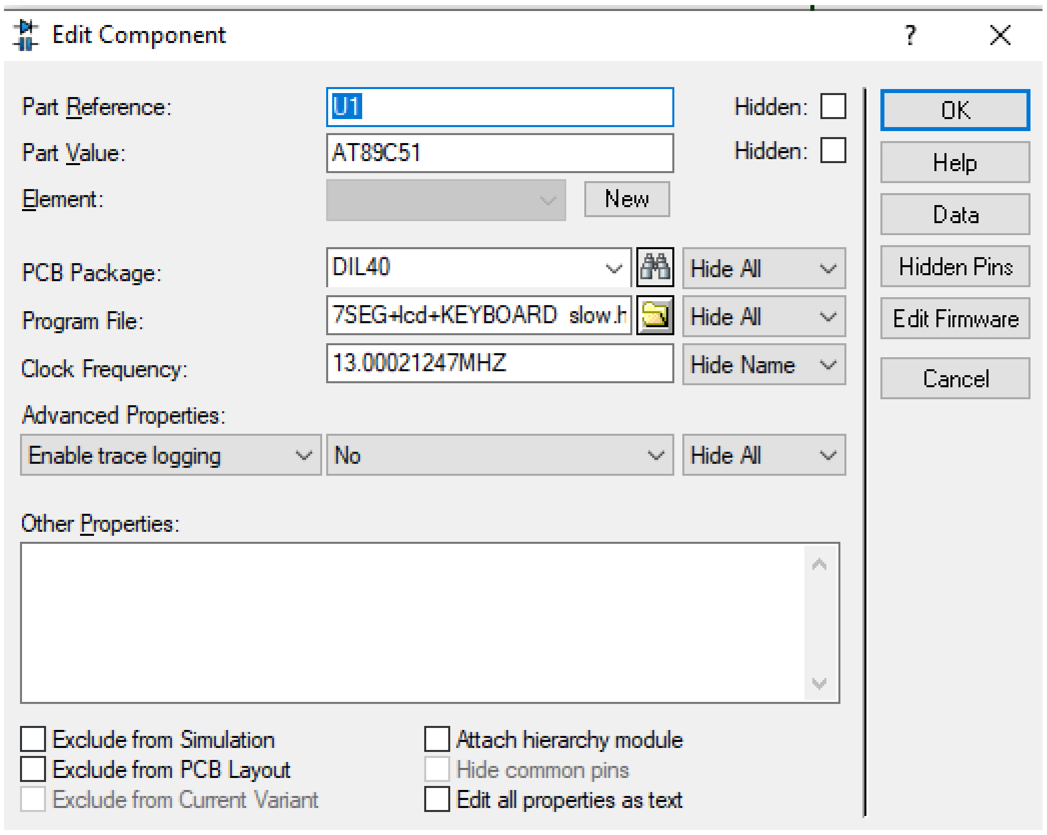
Based on AT89S52/AT89C51 microcontroller

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)*(student ID/10^9) Mhz = **13.00021247MHZ**



Edit Component

Part Reference:	U1	Hidden: ☐	
Part Value:	AT89C51	Hidden: ☐	
Element:		New	
PCB Package:	DIL40	Hide All	
Program File:	7SEG+lcd+KEYBOARD slow.h	Hide All	
Clock Frequency:	13.00021247MHZ	Hide Name	

Advanced Properties:

Enable trace logging	No	Hide All

Other Properties:

☐ Exclude from Simulation
☐ Exclude from PCB Layout
☐ Exclude from Current Variant
☐ Attach hierarchy module
☐ Hide common pins
☐ Edit all properties as text

CODE

```
; Original Jurgen oven code
; All rights reserved by :
;;;;;;;;;;;;;;;;;;;;;;;;;;;; K. M. Sirazul Monir ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



      org   0000h

       ;Initialize all ports to default state
INITIALIZE:    MOV     P3,#00000000B      ; Clear Port 3
       MOV P0, #0FEH               ; Set up keypad scanning
       MOV 30H,#0                  ; Clear memory variables
       MOV 32H,#0
       MOV R0,#0                   ; Reset register counters
       MOV R7, #15                 ; Set timer constant
       mov r5,#00H                 ; Clear fact counter
       MOV 69H,0H                  ; Clear memory location 69H
       CLR P2.7                    ; Turn off buzzer
       MOV P1, #00000000B          ; Initialize display port
       MOV TMOD, #11H              ; Set timer mode
       MOV TH1, #3CH               ; Initialize timer high byte
       MOV TL1, #98H               ; Initialize timer low byte
       SETB TR1                    ; Start timer
       CLR P2.5                    ; Turn off heating element

; Initialize registers for various operations
REGISTER_INIT:
MOV R3, #00H                        ; Clear display register
MOV R1, #00H                        ; Clear memory pointer
MOV R2, #00H                        ; Clear general purpose register


; Define LCD interface pins
IO_DEFINITION:
RS EQU P2.1                         ; Register Select pin for LCD
EN EQU P2.2                         ; Enable pin for LCD


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Initialize LCD with standard commands
LCD_INIT:
MOV R3, #38H                        ; Function set: 8-bit, 2 lines, 5x7 font
ACALL COMMAND                     ; Send command to LCD
MOV R3, #0EH                        ; Display on, cursor on
ACALL COMMAND
MOV R3, #80H                        ; Set cursor to beginning of first line
ACALL COMMAND
MOV R3, #01H                        ; Clear display
ACALL COMMAND

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Display "ENTER TIME IN s:" message
PROMPT_TIME_ENTRY: MOV DPTR,#TIME_PROMPT
DISPLAY_PROMPT:MOV A,#00H
       MOVC A,@A+DPTR
       JZ TIME_INPUT_LOOP             ; Jump if end of message (zero terminator)
       MOV R3,A
```

```
        ACALL DISPLAY              ; Display character
        INC DPTR
        LJMP DISPLAY_PROMPT


; Wait for first digit input from keypad
TIME_INPUT_LOOP:      LCALL SCAN
        MOV A,R0
        JZ TIME_INPUT_LOOP         ; If no key pressed, keep scanning

        MOV 40H,A                  ; Store first digit (hundreds place)
        ANL 40H,#00001111B         ; Mask upper bits to get digit value only
        MOV R1,#40H
        CJNE @R1,#0AH,CHECK_KEY_B  ; Check if valid digit (not A)
        SJMP TIME_INPUT_LOOP
CHECK_KEY_B:   CJNE @R1,#0BH,CHECK_KEY_C  ; Check if not B
        SJMP TIME_INPUT_LOOP
CHECK_KEY_C:   CJNE @R1,#0CH,CHECK_KEY_D  ; Check if not C
        SJMP TIME_INPUT_LOOP
CHECK_KEY_D:   CJNE @R1,#0DH,CHECK_KEY_E  ; Check if not D
        SJMP TIME_INPUT_LOOP
CHECK_KEY_E:   CJNE @R1,#0EH,CHECK_KEY_F  ; Check if not E
        SJMP TIME_INPUT_LOOP
CHECK_KEY_F:   CJNE @R1,#0FH,SHOW_DIGIT1  ; Check if not F
        SJMP TIME_INPUT_LOOP

; Display first digit on LCD
SHOW_DIGIT1:MOV R3, #0C0H              ; Set cursor to second line
        ACALL COMMAND
        MOV A,40H
        ADD A,#30H                 ; Convert digit to ASCII
        MOV R3,A
        ACALL DISPLAY
        lcall SHORT_DELAY          ; Small delay between keypresses


; Wait for second digit input from keypad
TENS_DIGIT_INPUT:     LCALL SCAN
        MOV A,R0
        JZ TENS_DIGIT_INPUT        ; If no key pressed, keep scanning
        MOV 44H,A                  ; Store second digit (tens place)
        ANL 44H,#00001111B         ; Mask upper bits


        MOV R1,#44H
        CJNE @R1,#0AH,CHECK_TENS_B ; Check if valid digit (not A)
        SJMP TENS_DIGIT_INPUT
CHECK_TENS_B:  CJNE @R1,#0BH,CHECK_TENS_C  ; Check if not B
        SJMP TENS_DIGIT_INPUT
CHECK_TENS_C:  CJNE @R1,#0CH,CHECK_TENS_D  ; Check if not C
        SJMP TENS_DIGIT_INPUT
CHECK_TENS_D:  CJNE @R1,#0DH,CHECK_TENS_E  ; Check if not D
        SJMP TENS_DIGIT_INPUT
CHECK_TENS_E:  CJNE @R1,#0EH,CHECK_TENS_F  ; Check if not E
        SJMP TENS_DIGIT_INPUT
CHECK_TENS_F:  CJNE @R1,#0FH,SHOW_DIGIT2   ; Check if not F
        SJMP TENS_DIGIT_INPUT


; Display second digit on LCD
SHOW_DIGIT2:   MOV A,44H
        ADD A,#30H                 ; Convert digit to ASCII
        MOV R3,A
        ACALL DISPLAY
        lcall SHORT_DELAY          ; Small delay between keypresses


; Wait for third digit input from keypad
ONES_DIGIT_INPUT:     LCALL SCAN
        MOV A,R0
```

```
       JZ ONES_DIGIT_INPUT        ; If no key pressed, keep scanning
       MOV 53H,A                  ; Store third digit (ones place)
       ANL 53H,#00001111B         ; Mask upper bits


       MOV R1,#53H
       CJNE @R1,#0AH,CHECK_ONES_B ; Check if valid digit (not A)
       SJMP ONES_DIGIT_INPUT
CHECK_ONES_B:  CJNE @R1,#0BH,CHECK_ONES_C  ; Check if not B
       SJMP ONES_DIGIT_INPUT
CHECK_ONES_C:  CJNE @R1,#0CH,CHECK_ONES_D  ; Check if not C
       SJMP ONES_DIGIT_INPUT
CHECK_ONES_D:  CJNE @R1,#0DH,CHECK_ONES_E  ; Check if not D
       SJMP ONES_DIGIT_INPUT
CHECK_ONES_E:  CJNE @R1,#0EH,CHECK_ONES_F  ; Check if not E
       SJMP ONES_DIGIT_INPUT
CHECK_ONES_F:  CJNE @R1,#0FH,SHOW_DIGIT3   ; Check if not F
       SJMP ONES_DIGIT_INPUT


; Display third digit on LCD
SHOW_DIGIT3:   MOV A,53H
       ADD A,#30H                 ; Convert digit to ASCII
       MOV R3,A
       ACALL DISPLAY
       lcall SHORT_DELAY          ; Small delay between keypresses


; Wait for START key (F key)
WAIT_FOR_START:       LCALL SCAN
       MOV A,R0
       ANL A,#00001111B
       CJNE A,#0FH,WAIT_FOR_START  ; Keep waiting until F key is pressed


; Calculate total time in seconds from the three digits entered
       MOV A,44H                  ; Get tens digit
       MOV B,A
       MOV A,#10
       MUL AB                     ; Multiply tens digit by 10
       ADD A,53H                  ; Add ones digit
       MOV 60H,A                  ; Store tens + ones value


       MOV A,40H                  ; Get hundreds digit
       MOV B,A
       MOV A,#100
       MUL AB                     ; Multiply hundreds digit by 100
       MOV 62H,A                  ; Store low byte of result
       MOV A,B
       MOV 61H,A                  ; Store high byte of result
       MOV A,62H
       ADD A,60H                  ; Add (tens + ones) to (hundreds * 100)
       MOV 62H,A
       JNC CHECK_MAX_TIME         ; Check if carry occurred during addition
       INC 61H                    ; If carry, increment high byte


; Check if time exceeds 300 seconds (maximum allowed)
CHECK_MAX_TIME:  MOV A,61H      ; Load high byte into accumulator
        CJNE A,#01H,CHECK_UPPER_BYTE  ; Compare with 01H (300 > 256)
        MOV A,62H                     ; If high byte is 01H, check low byte
        CJNE A,#2dH,CHECK_LOWER_BYTE  ; Compare with 2DH (45 decimal, 256+45=301)
        JMP TIME_OVER_300             ; If equal to 300 exactly, time is too large

CHECK_UPPER_BYTE: JC CHECK_MIN_TIME   ; If high byte < 01H, time is < 256
        JMP TIME_OVER_300             ; If high byte > 01H, time is > 300

CHECK_LOWER_BYTE: JC CHECK_MIN_TIME   ; If low byte < 2DH, time might be valid
        JMP TIME_OVER_300             ; If low byte > 2DH, time is > 300
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Check if time is less than 5 seconds (minimum allowed)
CHECK_MIN_TIME:  MOV A,61H            ; Load high byte into accumulator
        JNZ CHECK_MID_TIME            ; If high byte > 0, time is > 255
        MOV A,62H                     ; Load low byte into accumulator
        CJNE A,#05H,CHECK_MIN_TIME_TEMP ; Compare with 5 seconds
        JMP CHECK_MID_TIME            ; If exactly 5, proceed to mid check

CHECK_MIN_TIME_TEMP: JC TIME_UNDER_5  ; If Carry is set, time is < 5
        JMP CHECK_MID_TIME            ; If Carry is not set, time is > 5

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

; Display error for time less than 5 seconds
TIME_UNDER_5:  MOV DPTR,#TIME_TOO_SHORT_MSG
       MOV R3, #01H                ; Clear display and set cursor to first position
       ACALL COMMAND
DISPLAY_SHORT_TIME_ERROR:MOV A,#00H
       MOVC A,@A+DPTR
       JZ SHOW_RETRY_MESSAGE
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_SHORT_TIME_ERROR


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Show retry message on second line
SHOW_RETRY_MESSAGE: MOV R3, #0C0H      ; Set cursor to second line
       ACALL COMMAND
       MOV DPTR,#RETRY_MESSAGE
DISPLAY_RETRY:MOV A,#00H
       MOVC A,@A+DPTR
       JZ RETRY_DELAY
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_RETRY

RETRY_DELAY:   LCALL LONG_DELAY      ; Wait before restarting
       Ljmp INITIALIZE              ; Restart the program

; Check if time is more than 60 seconds (cooking method selection)
CHECK_MID_TIME:  MOV A,61H            ; Load high byte into accumulator
        JNZ TIME_OVER_60              ; If high byte > 0, time is > 255 > 60
        MOV A,62H                     ; Load low byte into accumulator
        CJNE A,#3CH,CHECK_60_TEMP     ; Compare with 60 (3CH) seconds
        JMP TIME_OVER_60              ; If exactly 60, consider it > 60

CHECK_60_TEMP: JC TIME_UNDER_60       ; If Carry is set, time is < 60
        JMP TIME_OVER_60              ; If Carry is not set, time is > 60


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Handle time > 60 seconds cooking mode
TIME_OVER_60: MOV DPTR,#TIME_OVER_60_MSG
       MOV R3, #01H                         ; Clear display
       ACALL COMMAND
DISPLAY_OVER_60:MOV A,#00H
       MOVC A,@A+DPTR
       JZ START_OVEN_MESSAGE_2
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_OVER_60

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Handle time < 60 seconds cooking mode
TIME_UNDER_60: MOV DPTR,#TIME_UNDER_60_MSG
```

```
       MOV R3, #01H                        ; Clear display
       ACALL COMMAND
DISPLAY_UNDER_60:MOV A,#00H
       MOVC A,@A+DPTR
       JZ START_OVEN_MESSAGE_1
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_UNDER_60

; Display error for time > 300 seconds
TIME_OVER_300: MOV DPTR,#TIME_OVER_300_MSG
       MOV R3, #01H                        ; Clear display
       ACALL COMMAND
DISPLAY_OVER_300:MOV A,#00H
       MOVC A,@A+DPTR
       JZ SHOW_RETRY_MESSAGE
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_OVER_300


; Show "OVEN STARTED" message for mode 2
START_OVEN_MESSAGE_2:MOV DPTR,#OVEN_STARTED_MSG
       MOV R3, #0C0H                       ; Set cursor to second line
       ACALL COMMAND
DISPLAY_START_2:MOV A,#00H
       MOVC A,@A+DPTR
       JZ COOKING_LOOP_2
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_START_2

; Show "OVEN STARTED" message for mode 1
START_OVEN_MESSAGE_1:MOV DPTR,#OVEN_STARTED_MSG
       MOV R3, #0C0H                       ; Set cursor to second line
       ACALL COMMAND
DISPLAY_START_1:MOV A,#00H
       MOVC A,@A+DPTR
       JZ COOKING_LOOP_1
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_START_1

; Main cooking loop for mode 2 (higher power)
COOKING_LOOP_2:MOV    R6,#20              ; Initialize loop counter
       SETB P2.5                           ; Turn on heating element
COOKING_COUNTDOWN_2:
       LCALL DELAY_1S                      ; Wait for 1 second
       LCALL DECREMENT_TIMER               ; Update the countdown
       DJNZ R6,COOKING_COUNTDOWN_2         ; Loop until counter expires
       LCALL UPDATE_DISPLAY_1              ; Update 7-segment display 1
       LCALL UPDATE_DISPLAY_2              ; Update 7-segment display 2
        LCALL UPDATE_DISPLAY_3              ; Update 7-segment display 3
       LCALL DISPLAY_RANDOM_FACT           ; Show a random cooking fact
       MOV    R6,#20                          ; Reset loop counter
        SJMP COOKING_COUNTDOWN_2            ; Continue cooking loop

; Main cooking loop for mode 1 (lower power)
COOKING_LOOP_1: LCALL LONG_DELAY          ; Small initial delay
SETB P2.5                                ; Turn on heating element
DISPLAY_COOKING_TIP:LCALL DELAY_1S       ; Wait for 1 second
       LCALL DECREMENT_TIMER               ; Update the countdown

        ;LCALL UPDATE_DISPLAY_1            ; Uncomment if using 7-segment displays
       ;LCALL UPDATE_DISPLAY_2             ; in mode 1
        ;LCALL UPDATE_DISPLAY_3
        SJMP DISPLAY_COOKING_TIP          ; Continue cooking loop


; Display cooking tip for mode 1
DISPLAY_COOKING_TIP_TEXT:
MOV DPTR,#QUICK_COOK_TIP
       MOV R3, #01H                       ; Clear display
       ACALL COMMAND
DISPLAY_TIP_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_TIP_END
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_TIP_LOOP
DISPLAY_TIP_END:
RET


; Display rotating facts for mode 2
DISPLAY_COOKING_FACT:
MOV DPTR,#DEFAULT_FACT
       MOV R3, #01H                       ; Clear display
       ACALL COMMAND
DISPLAY_FACT_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_FACT_END
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_FACT_LOOP
DISPLAY_FACT_END:
RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;-------------------------------------------------------
--------
; Display a random cooking fact from the fact library
DISPLAY_RANDOM_FACT:INC R5
       mov A,TL1                          ; Get a semi-random value from timer

       add a,r5                           ; Combine with counter for better randomness
       ;ANL A,#00001111B
       mov r5,A
       CJNE R5,#01H,CHECK_FACT_2          ; Check which fact to display
       MOV DPTR,#FACT_2_TEXT
       MOV R3, #01H                       ; Clear display
       ACALL COMMAND
DISPLAY_FACT_2_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_RANDOM_FACT_END_TEMP
       MOV R3,A
       ACALL DISPLAY

       INC DPTR
       LJMP DISPLAY_FACT_2_LOOP

CHECK_FACT_2:
       CJNE R5,#02H,CHECK_FACT_3          ; Check for fact 3
       MOV DPTR,#FACT_3_TEXT
       MOV R3, #01H                       ; Clear display
       ACALL COMMAND
DISPLAY_FACT_3_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_RANDOM_FACT_END_TEMP
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_FACT_3_LOOP
```

```
CHECK_FACT_3:
       CJNE R5,#03H,CHECK_FACT_4         ; Check for fact 4
       MOV DPTR,#FACT_4_TEXT
       MOV R3, #01H                      ; Clear display
       ACALL COMMAND
DISPLAY_FACT_4_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_RANDOM_FACT_END_TEMP
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_FACT_4_LOOP


CHECK_FACT_4:
       CJNE R5,#04H,CHECK_FACT_5         ; Check for fact 5
       MOV DPTR,#FACT_5_TEXT
       MOV R3, #01H                      ; Clear display
       ACALL COMMAND
DISPLAY_FACT_5_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_RANDOM_FACT_END_TEMP
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_FACT_5_LOOP


CHECK_FACT_5:
       CJNE R5,#05H,CHECK_FACT_6         ; Check for fact 6
       MOV DPTR,#FACT_6_TEXT
       MOV R3, #01H                      ; Clear display
       ACALL COMMAND
DISPLAY_FACT_6_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_RANDOM_FACT_END_TEMP
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_FACT_6_LOOP

DISPLAY_RANDOM_FACT_END_TEMP: LJMP DISPLAY_RANDOM_FACT_END

CHECK_FACT_6:
       CJNE R5,#06H,CHECK_FACT_7         ; Check for fact 7
       MOV DPTR,#FACT_7_TEXT
       MOV R3, #01H                      ; Clear display
       ACALL COMMAND
DISPLAY_FACT_7_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_RANDOM_FACT_END
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_FACT_7_LOOP

CHECK_FACT_7:
       CJNE R5,#07H,CHECK_FACT_8         ; Check for fact 8
       MOV DPTR,#FACT_8_TEXT
       MOV R3, #01H                      ; Clear display
       ACALL COMMAND
DISPLAY_FACT_8_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_RANDOM_FACT_END
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_FACT_8_LOOP
```

```
CHECK_FACT_8:
       CJNE R5,#08H,CHECK_FACT_9          ; Check for fact 9
       MOV DPTR,#FACT_9_TEXT
       MOV R3, #01H                       ; Clear display
       ACALL COMMAND
DISPLAY_FACT_9_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_RANDOM_FACT_END
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_FACT_9_LOOP


CHECK_FACT_9:
       CJNE R5,#09H,CHECK_FACT_10         ; Check for fact 10
       MOV DPTR,#FACT_10_TEXT
       MOV R3, #01H                       ; Clear display
       ACALL COMMAND
DISPLAY_FACT_10_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_RANDOM_FACT_END
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_FACT_10_LOOP


CHECK_FACT_10:
       CJNE R5,#10,RESET_FACT_COUNTER     ; Check for fact 11 or reset
       MOV DPTR,#FACT_11_TEXT
       MOV R3, #01H                       ; Clear display
       ACALL COMMAND
DISPLAY_FACT_11_LOOP:MOV A,#00H
       MOVC A,@A+DPTR
       JZ DISPLAY_RANDOM_FACT_END
       MOV R3,A
       ACALL DISPLAY
       INC DPTR
       LJMP DISPLAY_FACT_11_LOOP


RESET_FACT_COUNTER: MOV R5,#0H        ; Reset fact counter and start again
       LJMP DISPLAY_RANDOM_FACT


DISPLAY_RANDOM_FACT_END:
RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;----------------------------------------
; Decrement the cooking timer by 1 second
DECREMENT_TIMER:
       DEC 53H                        ; Decrement ones digit
        MOV A, 53H

        CJNE A, #11111111B, CONTINUE_TIMER  ; Check for underflow (FF)

        ; Reset ones digit and decrement tens digit
        MOV 53H, #9
        DEC 44H
        MOV A, 44H

        CJNE A, #11111111B, CONTINUE_TIMER

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

        CJNE A, #11111111B, CONTINUE_TIMER
```