Islamic University of Technology

ORGANIZATION OF ISLAMIC COOPERATION (OIC)

Department of Electrical and Electronic Engineering

Project Report

Course Code: EEE 4706

Course Name: Microcontroller-Based System Design Lab

Project Name: Real Time Clock

Group no: 1

Group Members:

Name	Section	ID
Abdullah Al Baki	A	190021125
Md. Sifat Aziz	A	190021131
Md. Faiyaz Abrar Fahim	A	190021137
Jamal Uddin	A	190021139
Ali Alusine Kamara	A	190021149

Table of Contents

Introduction

The implementation of the project involves the integration of both hardware and software components to create a real-time clock system based on the 8051 microcontroller. A real-time clock, which serves the purpose of timekeeping, is a computer clock that can be either microcontroller-based or an integrated circuit. It is commonly utilized in various devices such as computers, servers, and GPS systems to accurately track time and perform specific tasks at predetermined intervals. The real-time clock system can be realized through the utilization of an external pulse generator or by employing software programming internally. In this particular project, the software approach has been employed to achieve the functionality of the real-time clock. The project is done in two sections – one is the software section where the entire clock is built in the simulation software Proteus and in the other section the clock was realized on an 80S52 microcontroller board.

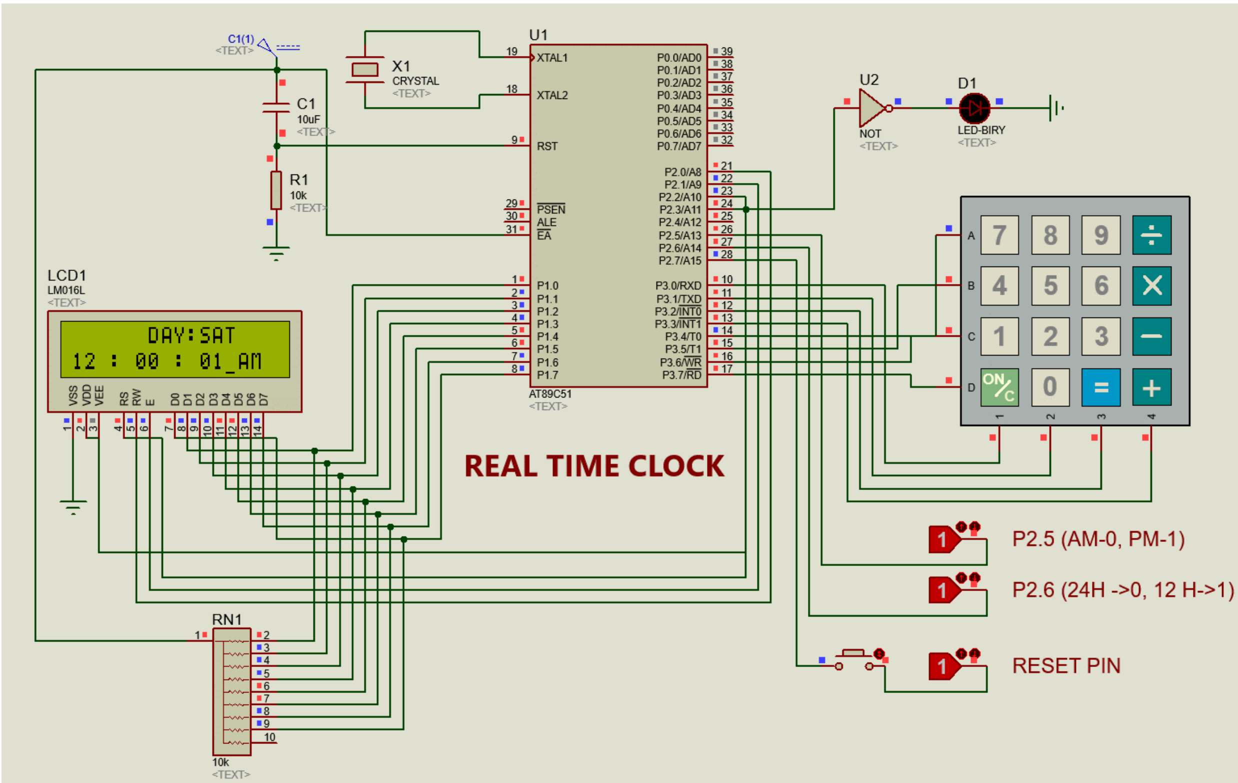
Objective

- To create a system that can accurately measure time in precise 1-second intervals
- Implementation and testing in a simulation environment to ensure its accuracy and reliability.
- Construction of the circuit according to the provided connection diagram.
- Programming into the microcontroller and integration with the circuit to create a fully functional time-counting system.
- Realizing the real-time clock in both software simulation and hardware setup.

Required Components

Sl. No	Component Name	Value/Model	Quantity
1	Microcontroller	AT89S52	1
2	Capacitor	47uF	2
3	Crystal Oscillator	11.0592 MHz	1
4	LCD Display	16*2 LCD	1
5	Keypad Matrix	N/A	1
6	Active Buzzer	2.3 kHz	1

Circuit Diagram



Features

Mandatory features:

1. **Use of LCD Display***:* A 16*2 LCD Display Module has been used to display the time in HH: MM: SS format.
2. **Clock Format***:* The clock supports 12 hr. format and it can show the part of the day with AM/PM.
3. **Day Format***:* The clock can also show the name of the day (i.e. SUN, MON, TUE etc.).

Additional features:

1. **Configurable Clock***:* The time of the clock can be configured using a keypad.
2. **Dual Format:** The clock will support 24 hr. clock format alongside 12 hr. format.
3. **Hourly Alarm***:* An active buzzer generates a beep sound after every hour to notify the user.
4. **Use of Delay***:* Programmed and calibrated delay has been used to generate 1-second delay and run the clock.

Working Principle

LCD Initialization

The MYLCD lookup table will be utilized to initialize the necessary steps for starting the display. Subsequently, the setup time display will become accessible.

Taking Input from Keypad

The keypad will be used to input ***seven clock values, six for clock and one for day*** which will then be saved in the RAM. Following this, the data will be transferred from the RAM to the respective registers. The screen will be cleared, and a new display showing the current operating time of the clock will be presented.

Clock Mechanism

The right hand of the second, R0, will be incremented. If R0 is not equal to 10, it will return to the loop and continue increasing R0. Once R0 reaches a value of 10, the left hand of the second, R1, will be incremented by 1, and R0 will be reset to zero before returning to the loop. The program will then check if R1 is equal to 6. If it is, the right hand of the minute, R2, will be incremented by 1, and R1 will be reset to zero before returning to the loop. If R1 is not equal to 6, it will simply return to the loop. When R2 reaches a value of 10, the left hand of the minute, R3, will be incremented by 1, and R2 will be reset to zero before returning to the loop. If R2 is not equal to 10, it will return to the loop. If R3 is equal to 6, the left hand of the hour, R4, will be incremented by 1, and R3 will be reset to zero before returning to the loop.

Mode selection

The clock mode selection is determined by the state of pin P2.6. If P2.6 is set to 0, the clock will operate in 24-hour mode. On the other hand, if P2.6 is not 0, the clock will be set to 12-hour mode. In the case of the 24-hour mode, the clock will first check if the left digit of the hour, represented by R5, is equal to 2. If it is, the clock will then verify if R4 is equal to 4. If R4 is indeed equal to 4, both R4 and R5 will be reset to their initial values, and the clock will return to its main loop. However, if R4 is not equal to 4, the loop will be repeated. If R5 is not equal to 2, the clock will check if R4 is equal to 10. If R4 is not equal to 10, the loop will be resumed. Otherwise, the value of R5 will be incremented by one, and the value of R4 will be set to 0 before returning to the loop. In contrast, when the 12-hour mode is selected, the clock will first evaluate if the left digit of the hour, represented by R5, is equal to 1. If it is, the clock will then check if R4 is equal to 3. If R4 is indeed equal to 3, R4 will be set to 1 and R5 will be reset to its initial value, and the cycle will continue. However, if R4 is not equal to 3, the loop will be repeated. If R5 is greater than 1, the clock will check if R4 is greater than 9. If R4 is not greater than 9, the loop will be resumed. Otherwise, the value of R5 will be increased by one, while the value of R4 will be set to 0.

Clock resetting

Resetting the clock involves utilizing the reset pin, P2.7. When this pin is activated, it initiates a process that clears the display, returns to the starting menu, and prompts for input values again. The new values provided will be stored, and a new time cycle will commence.

Buzzing of alarm

To indicate the passing of every hour, the activation of R4 by incrementing it by 1 triggers a buzzer. This buzzer will sound for a duration of 1 second. The buzzer is connected to 2.3. The buzzer gets turned on at active low signal.

Toggling of AM/PM

The toggling of AM and PM functionality is only applicable when the 12-hour mode is active. In this mode, the pin (P2.5) is utilized to select whether it is AM or PM. To perform the toggling operation, the system first checks if the value is 0. If it is, the system stores the binary value 00000000B in the B register. Otherwise, it stores the binary value 11111111B in the B register. Subsequently, the system checks if the value in the B register is zero. If it is, the display will show 'AM'. If it is not zero, the display will show 'PM'. After completing the 12-hour cycle, the value stored in the B register will be altered and returned to the loop. At this point, the display will show the toggled output of AM/PM.

Day display

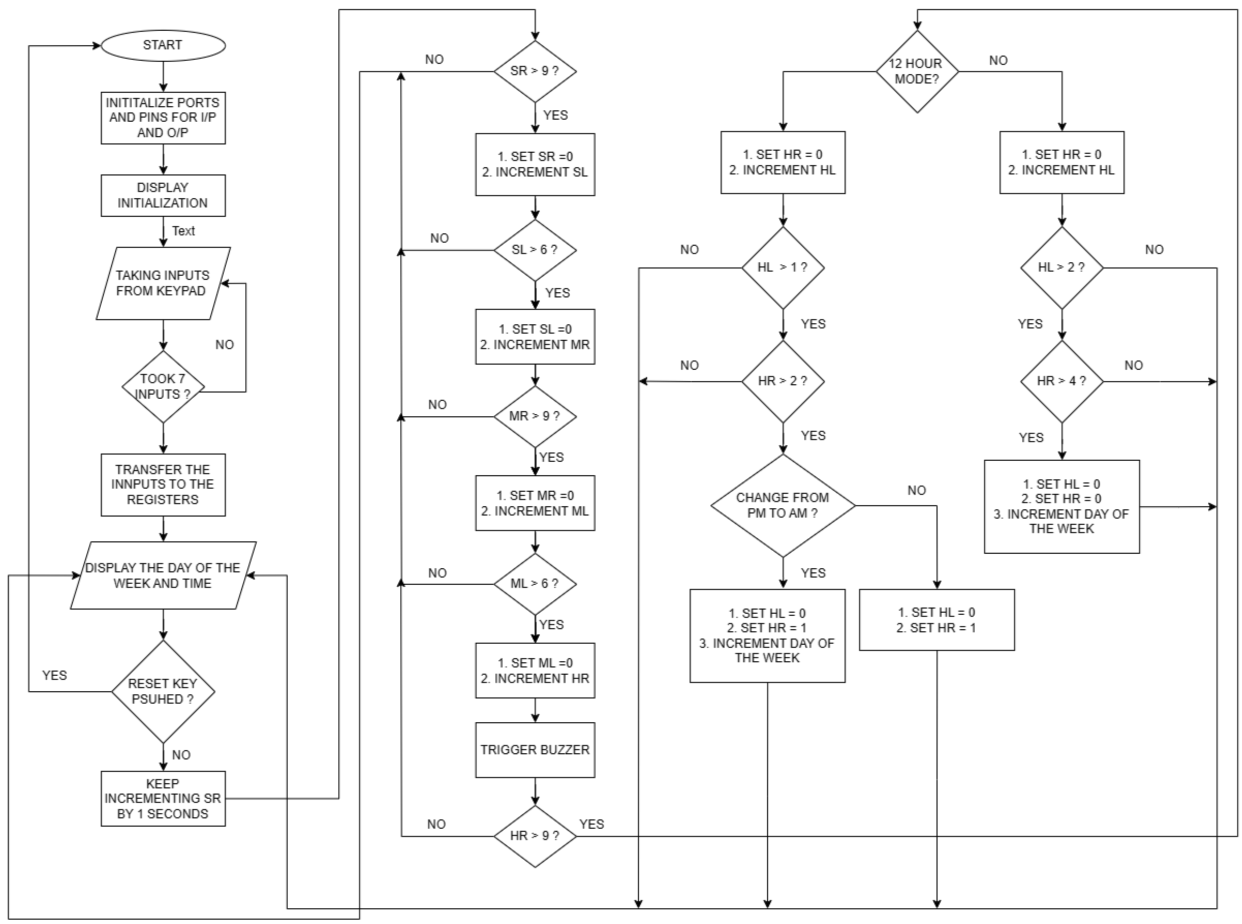
There is a seventh input to indicate the day is taken from the user which is stored in RAM locations. After 12 hours is passed in 12-hour mode if the PM is complemented then the day RAM location is incremented to show the passing of 1 day. And for 24-hour mode when R4, R5 is cleared then the day is incremented to show the passing of 1 day. And whenever the value is 7, when incremented it is restored to 1 to indicate passing from Friday to Saturday.

Flowchart

CLOCK FORMART

HOUR: MINUTE: SECOND

HL HR : ML MR : SL SR



Code

```
;--------------Start of coding------------

    ORG 0H

    PORT EQU P1     ; Port 1 is connected to display
    BUZZ EQU P2.3   ; Pin 2.3 is conected to buzzer

    RS EQU P2.0     ; Display RS pin is conncected to pin 2.0
    RW EQU P2.1 ; Display RW pin is conncected to pin 2.1
    E  EQU P2.2 ; Display E pin is conncected to pin 2.2

        JNB P2.5, J1    ; Switching betweeen AM/ PM
        MOV B,#11111111B    ; PM(B=1)
        SJMP J3

J1:    MOV B, #00000000B        ; AM(B=0)


;--------------I/O Initialization------------

J3: CLR  P2.6     ;Mode(12hr/24hr), 24H = LOGIC 0, 12H = LOGIC 1
    SETB BUZZ      ;Buzzer Pin
        CLR P2.7      ;Reset Pin

;------------DISPLAY INITIALIZATION----------

    MOV DPTR,#MYLCD     ;DPTR storing the LCD initialization sequences
C1:     CLR A
        MOVC A,@A+DPTR
        LCALL COMMAND
        LCALL DELAY
        JZ S1           ;Jump to S1 label when initialization commands are
executed
        INC DPTR
        SJMP C1

S1:
        MOV DPTR,#MSG1
D1:
    CLR A
    MOVC  A,@A+DPTR
    LCALL DISPLAY
    LCALL DELAY
        JZ S2  ;Runs rest of the code
        INC DPTR
        SJMP D1

;------------TAKING VALUES FROM KEYPAD AND STORING----------

S2:     MOV DPTR,#MY_COL  ;DPTR STORES LOCATION OF COLON SIGN

    MOV A,#0C2H       ;SET COLON CURSOR POSITION
    LCALL COMMAND
    CLR A
```

```
    MOVC  A,@A+DPTR   ;GET ASCII CODE FROM TABLE
    LCALL DISPLAY

    MOV A,#0C5H   ;SET COLON CURSOR POSITION
LCALL COMMAND
CLR A
    MOVC  A,@A+DPTR   ;GET ASCII CODE FROM TABLE
    LCALL DISPLAY

    ;---------- CURSOR POSITION OF HL
    CLR A
    MOV A,#0C0H
ACALL COMMAND
ACALL DELAY
;--------- INPUT OF HL VALUE
LCALL KEYPAD
LCALL DISPLAY
LCALL DELAY
ANL A,#0FH
MOV 40H,A   ;COPY A TO RAM LOCATION 40H
;---------- CURSOR POSITION OF HR
CLR A
    MOV A,#0C1H
ACALL COMMAND
ACALL DELAY
;--------- INPUT OF HR VALUE
LCALL KEYPAD
LCALL DISPLAY
LCALL DELAY
ANL A,#0FH
MOV 41H,A     ;COPY A TO RAM LOCATION 41H
    ;---------- CURSOR POSITION OF ML
    CLR A
    MOV A,#0C3H
ACALL COMMAND
ACALL DELAY
;--------- INPUT OF ML VALUE
LCALL KEYPAD
LCALL DISPLAY
LCALL DELAY
ANL A,#0FH
MOV 42H,A     ;COPY A TO RAM LOCATION 42H
;---------- CURSOR POSITION OF MR
CLR A
    MOV A,#0C4H
ACALL COMMAND
ACALL DELAY
;--------- INPUT OF MR VALUE
LCALL KEYPAD
LCALL DISPLAY
LCALL DELAY
ANL A,#0FH
MOV 43H,A     ;COPY A TO RAM LOCATION 43H
;---------- CURSOR POSITION OF SL
CLR A
    MOV A,#0C6H
ACALL COMMAND
```

```
    ACALL DELAY
    ;--------- INPUT OF SL VALUE
    LCALL KEYPAD
    LCALL DISPLAY
    LCALL DELAY
    ANL A,#0FH
    MOV 44H,A     ;COPY A TO RAM LOCATION 44H
    ;---------- CURSOR POSITION OF SR
    CLR A
        MOV A,#0C7H
    ACALL COMMAND
    ACALL DELAY
    ;--------- INPUT OF SR VALUE
    LCALL KEYPAD
    LCALL DISPLAY
    LCALL DELAY
    ANL A,#0FH
    MOV 45H,A     ;COPY A TO RAM LOCATION 45H

        ;------- DAY INPUT
    ;---------- CURSOR POSITION OF DAY
    CLR A
    MOV A,#0C9H
    ACALL COMMAND
    ;--------- INPUT OF DAY VALUE
    LCALL KEYPAD
    LCALL DISPLAY ;CALL DISPLAY SUBROUTINE
    LCALL DELAY ;GIVE LCD SOME TIME
    ANL A,#0FH
    MOV 46H,A ; COPY A TO RAM LOCATION 46

    SJMP START

;----------CLOCK DIGIT DECLARATION----------
;-----------SL SR : ML MR : HL HR-----------

START:  MOV R0,45H    ;SR : Second Right Digit
    MOV R1,44H    ;SL : Second Left Digit
    MOV R2,43H    ;MR : Minute Right Digit
    MOV R3,42H    ;ML : Minute Left Digit
    MOV R4,41H    ;HR : Hour Right Digit
    MOV R5,40H    ;HL : Hour Left Digit
    MOV R6,46H    ;DAY OF THE WEEK

    LCALL DELAY
    LCALL DELAY ;DELAY TO START THE CLOCK AFTER SET THE TIME

MOV A, #01 ;CLEAR LCD
ACALL COMMAND
ACALL DELAY

;----------DISPLAYING DAY OF THE WEEK----------
S_DATA2:
    MOV A, #80H ;CURSON AT LINE 1 FIRST POSITION
    ACALL COMMAND
    ACALL DELAY
    LCALL SHOW_DAY ;CALLING SHOW_DAY SUBROUTINE
```

```
S_DATA:

    JNB P2.6, J7     ;CHECKING IF IT S 12 HR OR 24 HR MODE
        SJMP J8

J8:     MOV DPTR,#msg4
    MOV A,#0CEH ;SET CURSOR POSITION OF M
    LCALL COMMAND
    CLR A
        MOVC  A,@A+DPTR ;GET ASCII CODE FROM TABLE
        LCALL DISPLAY

        CLR A
        MOV A,B
        JZ J5       ; AM OR PM DECIDER
        SJMP J6

J5:     MOV DPTR,#msg2
    MOV A,#0CDH ;SET CURSOR POSITION OF A
    LCALL COMMAND
    CLR A
        MOVC  A,@A+DPTR
        LCALL DISPLAY
        SJMP J7

J6:     MOV DPTR,#msg3
    MOV A,#0CDH ;SET CURSOR POSITION OF P
    LCALL COMMAND
    CLR A
        MOVC  A,@A+DPTR
        LCALL DISPLAY
        SJMP J7

J7:     MOV DPTR,#MY_COL
    MOV A,#0C3H ;SET CURSOR POSITION OF COLON
    LCALL COMMAND

    ;------DISPLAY 1ST COLON
    CLR A
        MOVC  A,@A+DPTR
        LCALL DISPLAY
        MOV DPTR,#MY_COL
        MOV A,#0C8H
        LCALL COMMAND

        ;------DISPLAY 2ND COLON
        CLR A
        MOVC  A,@A+DPTR
        LCALL DISPLAY

          MOV DPTR,#MY_NUMBER

          MOV A,#0C0H   ;SET CURSOR POSITION OF HL
          LCALL COMMAND
          LCALL DELAY
          ;-------- DISPLAY HL
```

```
          MOV A,R5
          MOVC  A,@A+DPTR
          LCALL DISPLAY
          LCALL DELAY

          MOV A,#0C1H   ;SET CURSOR POSITION OF HR
          LCALL COMMAND
          LCALL DELAY
          ;-------- DISPLAY HR
          MOV A,R4
          MOVC  A,@A+DPTR
          LCALL DISPLAY
          LCALL DELAY

          ;SET CURSOR POSITION OF ML
          MOV A,#0C5H
          LCALL COMMAND
          LCALL DELAY
          ;-------- DISPLAY ML
          MOV A,R3
          MOVC  A,@A+DPTR
          LCALL DISPLAY
          LCALL DELAY

          ;SET CURSOR POSITION OF MR
          MOV A,#0C6H
          LCALL COMMAND
          LCALL DELAY
          ;-------- DISPLAY MR
          MOV A,R2
          MOVC  A,@A+DPTR
          LCALL DISPLAY
          LCALL DELAY

          ;SET CURSOR POSITION OF SL
          MOV A,#0CAH
          LCALL COMMAND
          LCALL DELAY
          ;-------- DISPLAY SL
          MOV A,R1
          MOVC  A,@A+DPTR
          LCALL DISPLAY
          LCALL DELAY

          ;SET CURSOR POSITION OF SR
          MOV A,#0CBH
          LCALL COMMAND
          LCALL DELAY
          ;-------- DISPLAY SR
          MOV A,R0
          MOVC  A,@A+DPTR
          LCALL DISPLAY

          LCALL DELAY3  ;---------1 SECOND DELAY FOR THE SECOND---------

W1:       JNB P2.7,W2           ; RESET PIN
          MOV A,#01H        ; CLEAR SCREEN
```

```
          LCALL COMMAND
          LCALL DELAY
          LJMP S1         ; AGAIN INITIALIZE TO TAKE NEW INPUTS

;-------------------------CLOCK LOGIC------------------
;------------CLOCK FORMAT: HL HR : ML MR : SL SR---------

W2 :      INC R0    ;INCREMENT SR
          CJNE R0,#10,L2
          SJMP L9
L9:       MOV R0,#0
          INC R1    ;INCREMENT SL
          CJNE R1,#6,L2
          SJMP L10
L10:      MOV R1,#0
          INC R2    ;INCREMENT MR
          CJNE R2,#10,L2
          SJMP L7
L7:       MOV R2,#0
          INC R3    ;INCREMENT ML
          CJNE R3,#6,L2
          SJMP L8
L8:       MOV R3,#0
          INC R4    ;INCREMENT HR
          CLR BUZZ  ;HOURLY BUZZER ALARM
          LCALL DELAY3
          SETB BUZZ
          INC R0    ;COMPENSATION OF 1 SECOND FOR THE BUZZER

HERE:     JNB P2.6, G1     ;MODE SELECTION (12 HR OR 24 HR)
          LJMP G2

;---------- 24 HOUR CLOCK LOGIC ---------

G1:       CJNE R5,#2,MODE_1
          SJMP MODE_2

MODE_1:
          CJNE R4,#10,L2
          SJMP L3
L2:       LJMP S_DATA

L3:       MOV R4,#0
          INC R5
          CJNE R5,#2,L6
          LJMP S_DATA
MODE_2:
         CJNE R4,#4,L4
         SJMP L5
L4:      LJMP S_DATA
L5:      MOV R4,#0
         MOV R5,#0  ;AFTER COMPLETING 1 CYCLE OF 24 HOURS, RESET THE HL AND
HR
         INC 46H    ;DAY INCREMENT
         MOV R7,46H
         CJNE R7,#8,J9
         MOV 46H,#1 ;WHEN FRIDAY IS REACHED, MOVE TO SATURDAY
```

```
         SJMP L6

L6:     LJMP S_DATA2

;---------- 12 HOUR CLOCK LOGIC ---------

G2:  CJNE R5,#1,MODE_3
          SJMP MODE_4

MODE_3:
          CJNE R4,#10,Q2
          SJMP Q3
Q2:       LJMP S_DATA

Q3:       MOV R4,#0
          INC R5
          CJNE R5,#1,Q6
          LJMP S_DATA
MODE_4:  CJNE R4,#2,J9
         CLR A
         MOV A,B
         CPL A      ;TOGGLE THE AM/PM AFTER 12 HOURS
         MOV B,A
         CJNE A,#0,J9
         INC 46H    ;DAY INCREMENT
         MOV R7,46H
         CJNE R7,#8,J9
         MOV 46H,#1 ;WHEN FRIDAY IS REACHED, MOVE TO SATURDAY

J9:      CJNE R4,#3,Q4
         SJMP Q5
Q4:      LJMP S_DATA2
Q5:      MOV R4,#1  ;AFTER COMPLETING 1 CYCLE, IT WON T BE 00
         MOV R5,#0  ;RATHER THAN IT LL BE STARTED FROM 12:00:01

    ;WHEN ANOTHER HOUR WILL BE COMPLETED,
    ;THEN HR WILL 1 AND HL WILL BE 0 AS
    ;IN 12 HR MODE; AFTER 1 COMES AFTER 12

        SJMP Q6
Q6:     LJMP S_DATA2


KEYPAD:
    MOV A,#0FH
    MOV P3,A ;MAKE PORT 3 AS INPUT
K1:     MOV P3,#00001111B
    MOV A,P3    ;READ ALL COLUMNS, ENSURE ALL KEYS OPEN
    ANL A,#00001111B    ;MASK UNUSED BITS
    CJNE A,#00001111B,K1    ;CHECK TILL ALL KEYS RELEASED

K2:     ACALL DELAY
    MOV A,P3
    ANL A,#00001111B
    CJNE A,#00001111B,OVER
    SJMP K2
```

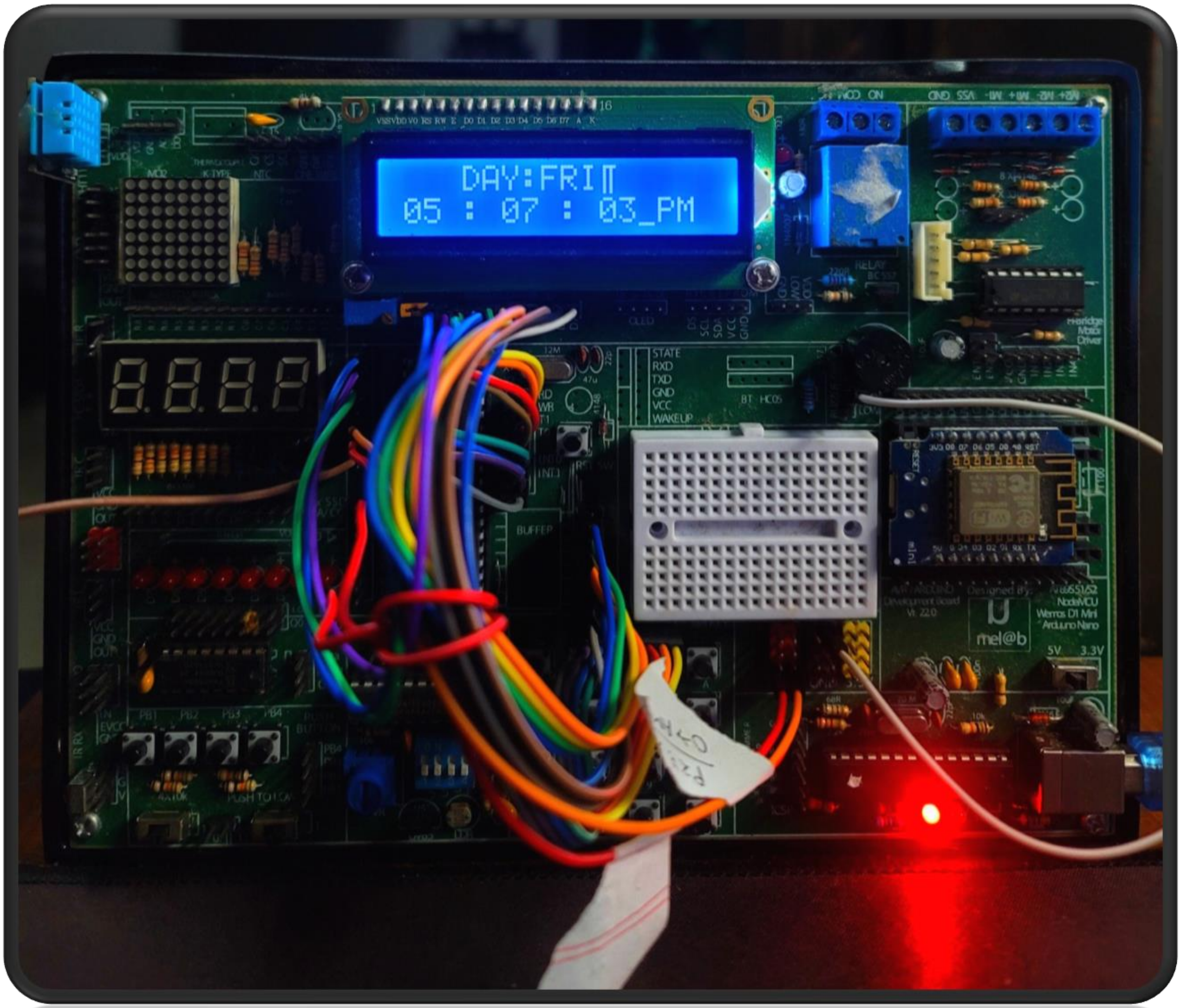
```
OVER:   ACALL DELAY
    MOV A,P3    ;CHECK KEY CLOSURE
    ANL A,#00001111B
    CJNE A,#00001111B,OVER1   ;KEY PRESSED, FIND RO
    SJMP K2     ;IF NONE, KEEP POLLING


OVER1:  MOV P3,#11101111B    ;GROUND ROW 0
    MOV A,P3         ;READ ALL COLUMNS
    ANL A,#00001111B     ;MASK UNUSED BITS
    CJNE A,#00001111B,ROW_0  ;KEY ROW 0, FIND THE COLUMN
    MOV P3,#11011111B   ;GROUND ROW 1
    MOV A,P3        ;READ ALL COLUMNS
    ANL A,#00001111B    ;MASK UNUSED BITS
    CJNE A,#00001111B,ROW_1 ;KEY ROW 1, FIND THE COLUMN
    MOV P3,#01111111B   ; SIMILLAR
    MOV A,P3
    ANL A,#00001111B
    CJNE A,#00001111B,ROW_2
    MOV P3,#10111111B
    MOV A,P3
    ANL A,#00001111B
    CJNE A,#00001111B,ROW_3

    LJMP K2      ;IF NONE, FALSE INPUT, REPEAT

ROW_0:  MOV DPTR,#KCODE0    ;SET DPTR=START OF ROW 0
    SJMP FIND       ;FIND COLUMN.KEY BELONGS TO
ROW_1:  MOV DPTR,#KCODE1    ;SET DPTR=START OF ROW 1
    SJMP FIND       ;FIND COLUMN.KEY BELONGS TO
ROW_2:  MOV DPTR,#KCODE3    ; SIMILLAR
    SJMP FIND
ROW_3:  MOV DPTR,#KCODE2

FIND:   RRC A           ;SEE IF ANY CY BIT IS LOW
    JNC MATCH       ;IF ZERO, GET THE ASCII CODE
    INC DPTR        ;POINT TO THE NEXT COLUMN ADDRESS
    SJMP FIND       ;KEEP SEARCHING

MATCH:  CLR A           ;SET A=0 (MATCH FOUND)
    MOVC A,@A+DPTR
    RET

COMMAND:
        MOV PORT,A
        CLR RS          ;RS=0 FOR COMMAND
        CLR RW          ;R/W=0 FOR WRITE
        SETB E          ;E=1 FOR HIGH PULSE
        ACALL DELAY
        CLR E           ;E=0 FOR H-TO-L PULSE
        RET
DISPLAY:
        MOV PORT,A
        SETB RS         ;RS=1 FOR DATA
        CLR RW          ;R/W=0 FOR WRITE
        SETB E          ;E=1 FOR HIGH PULSE
        ACALL DELAY
```

```
        CLR E           ;E=0 FOR H-TO-L PULSE
        RET

;-----------------DELAY FOR REFRESHING DISPLAY AND DATA PROCESSING

DELAY:  SETB PSW.2
        MOV R6, #10
H1:     MOV R7,#5      ;NORMAL DELAY
H2:     DJNZ R7,H2
        DJNZ R6,H1
        CLR PSW.2
        RET

;---------------DELAY FOR BUZZER

DELAY1: SETB PSW.4
        MOV R5,#235
H5:     MOV R6, #7  ;BUZZER
H3:     MOV R7,#230
H4:     DJNZ R7,H4
        DJNZ R6,H3
        DJNZ R5,H5
        CLR PSW.4
    RET

;--------------REAL TIME DELAY (1 SECOND)

DELAY3: SETB PSW.3
        MOV R5,#235
H6:     MOV R6,#230
H7:     MOV R7,#7
H8:     DJNZ R7,H8      ;1 SEC DELAY
        DJNZ R6,H7
        DJNZ R5,H6
        CLR PSW.3
        RET

;ASCII LOOK-UP TABLE FOR EACH ROW
KCODE0: DB '7','8','9','/' ;ROWnumber 0
KCODE1: DB '4','5','6','*' ;ROWnumber 1
KCODE2: DB '1','2','3','-' ;ROWnumber 2
KCODE3: DB 99H,'0','=','+' ;ROWnumber 3

;------------------SHOWING THE DAY
SHOW_DAY:
    CLR A
    MOV A,46H

    CJNE A,#1,DAY_2
    MOV DPTR,#MY_DAY_1
    LOOP_DAY_1:
    CLR A
    MOVC A,@A+DPTR ;GET ASCII CODE FROM TABLE
    LCALL DISPLAY ;CALL DISPLAY SUBROUTINE
    LCALL DELAY
    JZ DAY_2
    INC DPTR
```

```
SJMP LOOP_DAY_1

DAY_2:
CJNE A,#2,DAY_3
MOV DPTR,#MY_DAY_2
LOOP_DAY_2:
CLR A
MOVC A,@A+DPTR ;GET ASCII CODE FROM TABLE
LCALL DISPLAY ;CALL DISPLAY SUBROUTINE
LCALL DELAY
JZ DAY_3
INC DPTR
SJMP LOOP_DAY_2

DAY_3:
CJNE A,#3,DAY_4
MOV DPTR,#MY_DAY_3
LOOP_DAY_3:
CLR A
MOVC A,@A+DPTR ;GET ASCII CODE FROM TABLE
LCALL DISPLAY ;CALL DISPLAY SUBROUTINE
LCALL DELAY
JZ DAY_4
INC DPTR
SJMP LOOP_DAY_3

DAY_4:
CJNE A,#4,DAY_5
MOV DPTR,#MY_DAY_4
LOOP_DAY_4:
CLR A
MOVC A,@A+DPTR ;GET ASCII CODE FROM TABLE
LCALL DISPLAY ;CALL DISPLAY SUBROUTINE
LCALL DELAY
JZ DAY_5
INC DPTR
SJMP LOOP_DAY_4

DAY_5:
CJNE A,#5,DAY_6
MOV DPTR,#MY_DAY_5
LOOP_DAY_5:
CLR A
MOVC A,@A+DPTR ;GET ASCII CODE FROM TABLE
LCALL DISPLAY ;CALL DISPLAY SUBROUTINE
LCALL DELAY
JZ DAY_6
INC DPTR
SJMP LOOP_DAY_5

DAY_6:
CJNE A,#6,DAY_7
MOV DPTR,#MY_DAY_6
LOOP_DAY_6:
CLR A
MOVC A,@A+DPTR ;GET ASCII CODE FROM TABLE
LCALL DISPLAY ;CALL DISPLAY SUBROUTINE
```

```
    LCALL DELAY
    JZ DAY_7
    INC DPTR
    SJMP LOOP_DAY_6

    DAY_7:
    CJNE A,#7,DAY_8
    MOV DPTR,#MY_DAY_7
    LOOP_DAY_7:
    CLR A
    MOVC A,@A+DPTR ;GET ASCII CODE FROM TABLE
    LCALL DISPLAY ;CALL DISPLAY SUBROUTINE
    LCALL DELAY
    JZ DAY_8
    INC DPTR
    SJMP LOOP_DAY_7

    DAY_8:
    RET


ORG 600H
MSG1: DB "    INITIALIZE",0
MYLCD : DB 38H,0EH,01,06,80H,0
MY_COL: DB ":"
MY_NUMBER: DB "0","1","2","3","4","5","6","7","8","9"
MY_DAY_1: DB "      DAY:SAT",0
MY_DAY_2: DB "      DAY:SUN",0
MY_DAY_3: DB "      DAY:MON",0
MY_DAY_4: DB "      DAY:TUE",0
MY_DAY_5: DB "      DAY:WED",0
MY_DAY_6: DB "      DAY:THU",0
MY_DAY_7: DB "      DAY:FRI",0
msg2: DB "A"
msg3: DB "P"
msg4: DB "M"

END
```

Hardware Implementation



Problems Encountered

1. **Unpredictable behavior of Proteus***:* The simulator software occasionally exhibits unexpected output and behaves in an unpredictable manner, which hinders the addition of additional features and program improvements.
2. **Display freezing***:* In Proteus, there are instances where the display freezes and retains the time value for a longer duration than 1 second. However, following this glitch, the subsequent time updates occur much faster than 1 second, thereby rectifying the anomaly.
3. **Alarm glitch***:* When the buzzer beeps, the time freezes and skips the HH:00:00 time, immediately transitioning to HH:00:01 after HH:59:59. This occurs because the buzzer requires an additional second, causing the SR register to increment twice to compensate for this delay.
4. **Complexity of coding sequence***:* The complexity of the coding process has introduced errors, including breaches in the flow chart and unexpected output. Numerous backups and iterative approaches have been pursued to overcome this problem.
5. **Display glitch:** You have encountered a ‘pi’ looking character in the first line of the lcd. We are not sure weather it is a display glitch or code error.

Conclusion

This project has helped us improve our skills in applying assembly language programming to hardware, allowing us to effectively address real-world difficulties. Our understanding of assembly language has significantly increased, as seen by the way several features have been implemented successfully throughout the project. Even though some obstacles have emerged, it's important to understand that these are manageable problems that offer chances to bring the project's unrealized potential to light. Enhancing the project's overall development requires integrating the Real-Time Clock (RTC) module and adding sophisticated features like timers and interrupts. This tactical improvement strengthens the project's durability and promotes modularity, which is a major step toward its overall improvement.