Microprocessor & Interfacing Design Assignment

Digital Alarm Clock

SHAH NEEL KAUSHIK	**2016A7PS0076P**
AVIRAL AGRAWAL	**2016A7PS0077P**
JAMI HARSHITH	**2016A7PS078P**
MEHTA AASHAY PINKESH	**2016A7PS0079P**

Group Number - 32

Question 19

25.04.2018

0. TABLE OF CONTENTS

1. PROBLEM STATEMENT

Problem 19:

System to be designed:- Digital Clock

Description:- A Digital Alarm Clock that displays Time

Basic Functionalities:-

- Time is displayed in HH:MM:SS format along with date (dd/mm/20yy). Both 24 Hr and 12 Hr formats are available and can be decided by the user.
- All of the above can be set by the user.
- Alarm can be set to a particular hour and minutes.
- The time will be displayed and updated in real time on the LCD screen provided.
- The alarm, when it rings, plays the musical octave (Sa Re Ga Ma Pa...) for the entire minute.

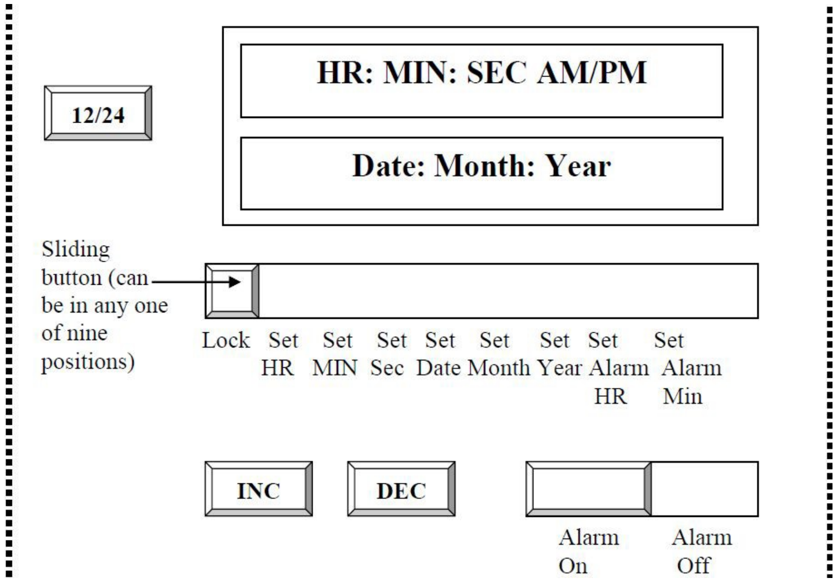
User Interface:-

- The LCD displays the current time and the date.
- Using the increment and decrement Set Switches, the user can set Seconds / Minutes / Hours / Date / Month / Year.
- Similarly, the user can set the Alarm Hour and Alarm Min using the same increment and decrement switches.
- A slider is available to decide upon the functionality that the user wishes to use. The table below lists them:

Value	Functionality	Explanation
8	LOCK	When the slider is in the LOCK position, the clock functions normally, i.e. the LCD displays the time and date. The time is updated in real time.
7	Set Hour	In the Set Hour position, the clock stops functioning. The LCD displays the current value of Hour, Minute and Second. The increment and decrement push buttons increment and decrement the Hour value respectively.
6	Set Minute	In the Set Minute position, the clock stops functioning. The LCD displays the current value of Hour, Minute and Second. The increment and decrement push buttons increment and decrement the Minute value respectively.
5	Set Second	In the Set Second position, the clock stops functioning. The LCD displays the current value of Hour, Minute and Second. The increment and decrement push buttons increment and decrement the Second value respectively.

4	Set Date	In the Set Date position, the clock stops functioning. The LCD displays the current value of Day, Month and Year. The increment and decrement push buttons increment and decrement the Day value respectively.
3	Set Month	In the Set Month position, the clock stops functioning. The LCD displays the current value of Day, Month and Year. The increment and decrement push buttons increment and decrement the Month value respectively.
2	Set Year	In the Set Year position, the clock stops functioning. The LCD displays the current value of Day, Month and Year. The increment and decrement push buttons increment and decrement the Year value respectively.
1	Set Alarm Minute	In the Set Alarm Minute position, the clock stops functioning. The LCD displays the current value of Alarm Hour and Alarm Minute. The increment and decrement push buttons increment and decrement the Alarm Minute value respectively.
0	Set Alarm Hour	In the Set Alarm Hour position, the clock stops functioning. The LCD displays the current value of Alarm Hour and Alarm Minute. The increment and decrement push buttons increment and decrement the Alarm Hour value respectively.

- The switches are available: 12/24 and Alarm On/Off, the position of which determine the display time format and turn the alarm on and off respectively.
- A diagram of the proposed user interface is given below for the sake of clarity:-



2. ASSUMPTIONS

- No. of days in a month is assumed to be 30, for the sake of simplicity. (We ignore the case of 31 days and the special case of 28 day month).
- We assume that the year is of the format 20XX. (X can be any digit).
- Alarm rings for 1 minute.
- The clock starts with the date 26/4//2018 at 07:00:50 PM. The actual time at the moment of running the simulation must be set manually.
- **NOTE: Simulation is not running in real time due to excessive CPU load.** More precisely**, All functionalities in the simulation run correctly, but the effects are delayed.** The clock fails to keep the real time in the simulation.

3. COMPONENTS USED

3.1. ICs

COMPONENT	MODEL NUMBER	NUMBER OF UNITS
Microprocessor	8086	1
Bidirectional Buffer	74LS245	2
Octal Latch	74LS373	3

3 to 8 line Decoder [DeMux]	74LS138	4
Programmable Interval Timer	8253A	2
Programmable Peripheral Interface	8255A	1
20 X 4 LCD	LM044L with hd44780	2
RAM	2732	2
ROM	6116	2

3.2. OTHER COMPONENTS

COMPONENT	MODEL NUMBER	NUMBER OF UNITS
PUSH BUTTON	SW-SPDT-MOM	2
SWITCH	SWITCH	2
BCD THUMB SWITCH	THUMB SWITCH-BCD	1
BUZZER	BUZZER	8
RELAY	RELAY	8

3.3. BASIC GATES

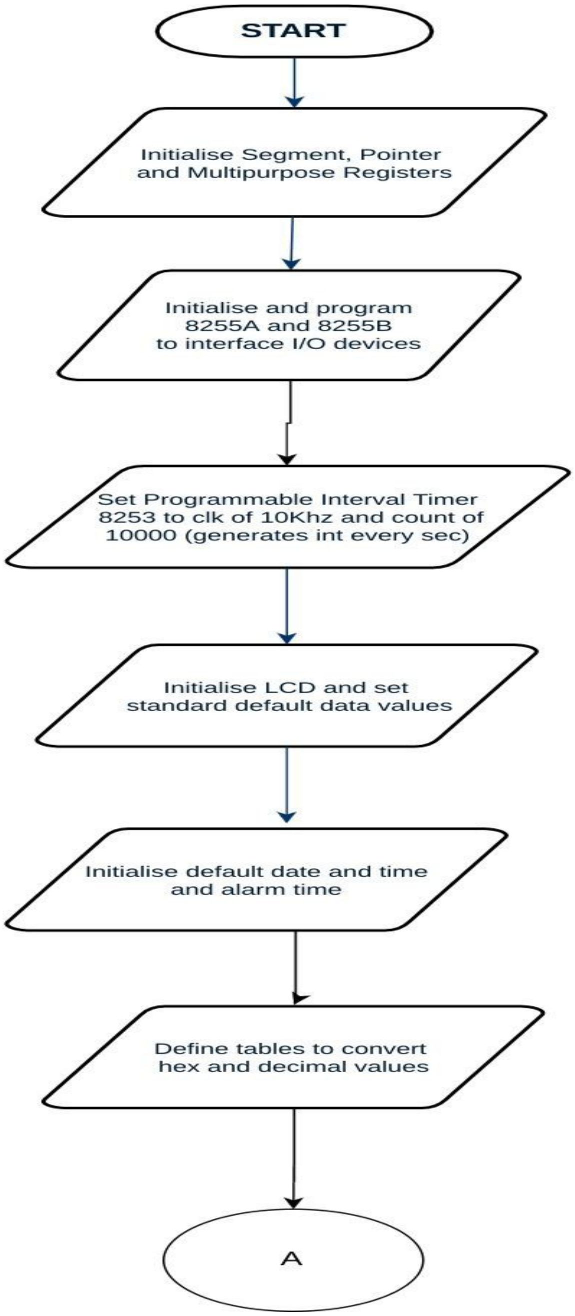
COMPONENT	MODEL NUMBER	NUMBER OF UNITS
AND	7408	1
OR	7432	6
NOT	7404	10

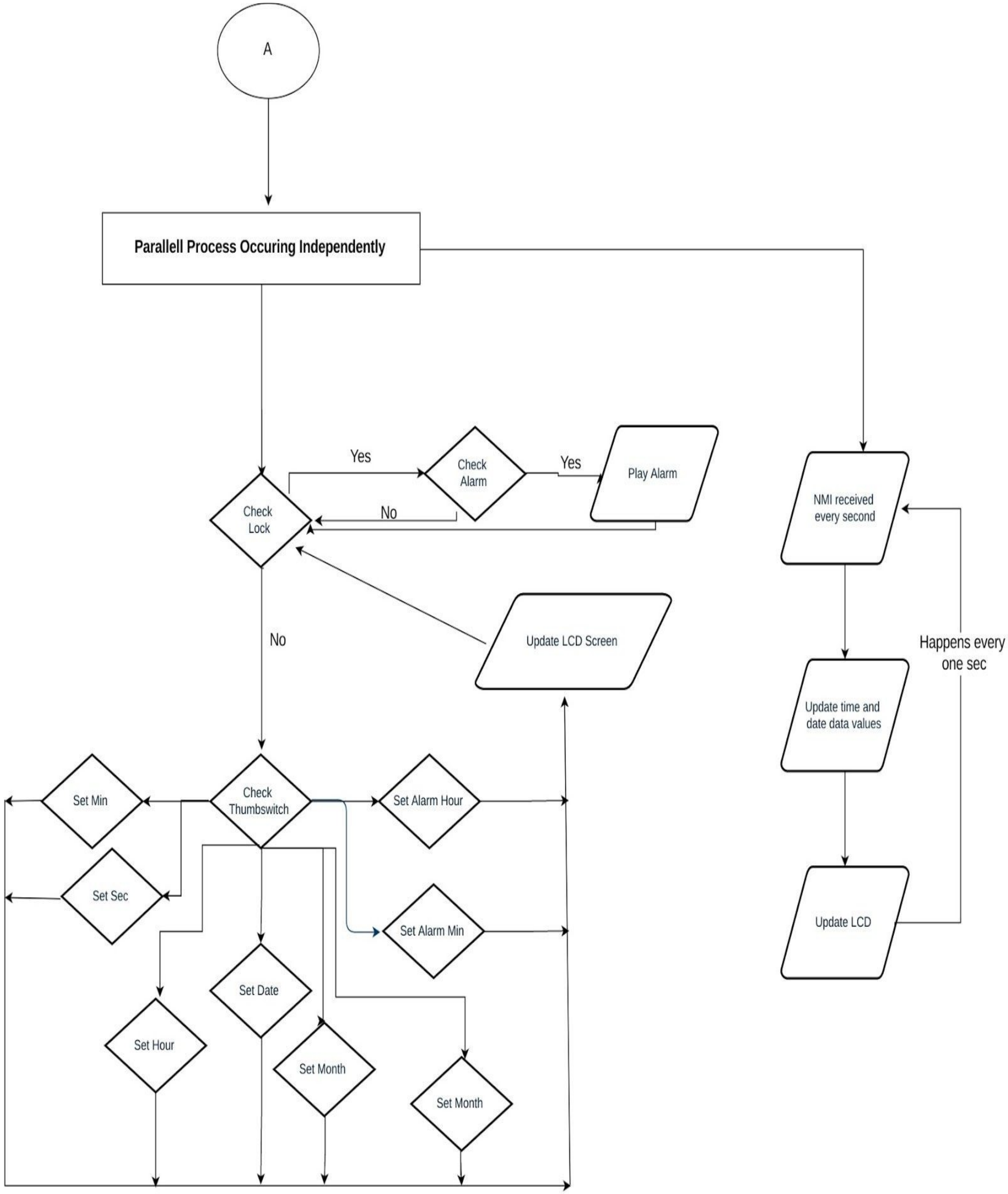
4. MEMORY ADDRESS MAP

Memory or I/O Device	Address Space
RAM - 2*2K	even chip:00000h - 007FEh odd chip: 00001h - 007FFh

Programmable Peripheral Interface A	00h - 07h
Programmable Peripheral Interface B	10h - 17h
Programmable Interval Generator	08h - 0Fh

5. FLOW CHART

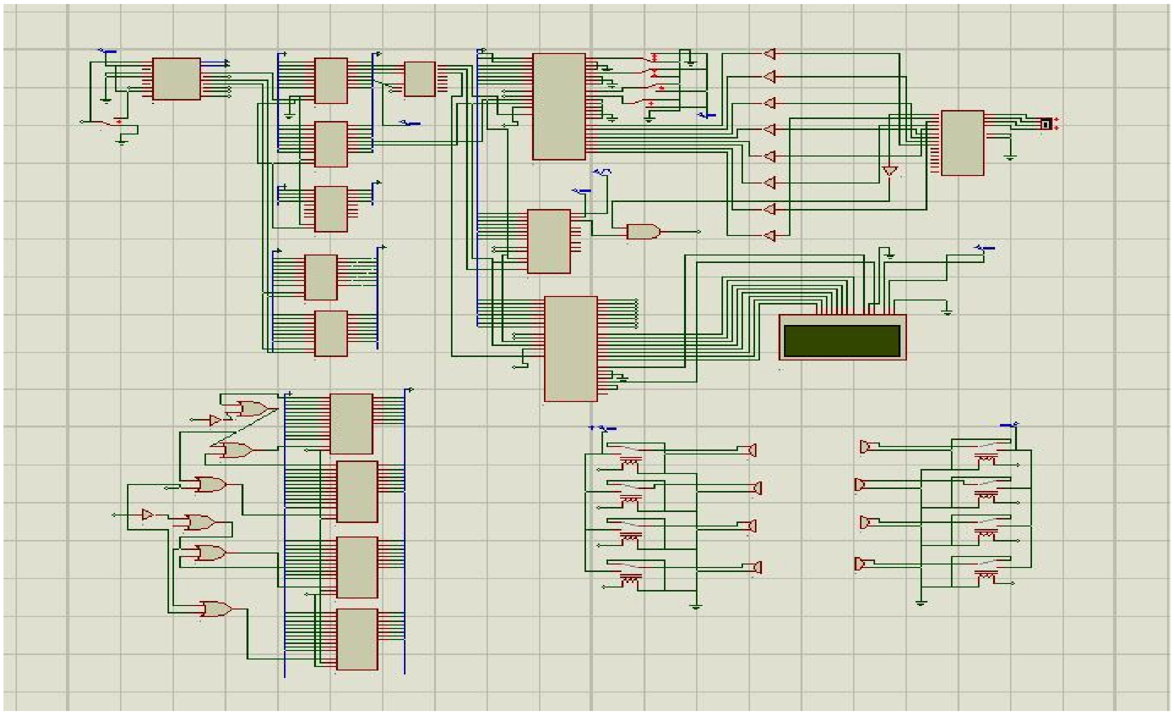

START
Initialise Segment, Pointer and Multipurpose Registers
Initialise and program 8255A and 8255B to interface I/O devices
Set Programmable Interval Timer 8253 to clk of 10Khz and count of 10000 (generates int every sec)
Initialise LCD and set standard default data values
Initialise default date and time and alarm time
Define tables to convert hex and decimal values
A


A
Parallell Process Occuring Independently
Check Lock
Yes
Check Alarm
Yes
Play Alarm
No
NMI received every second
No
Update LCD Screen
Happens every one sec
Update time and date data values
Update LCD
Set Min
Check Thumbswitch
Set Alarm Hour
Set Sec
Set Alarm Min
Set Date
Set Hour
Set Month
Set Month

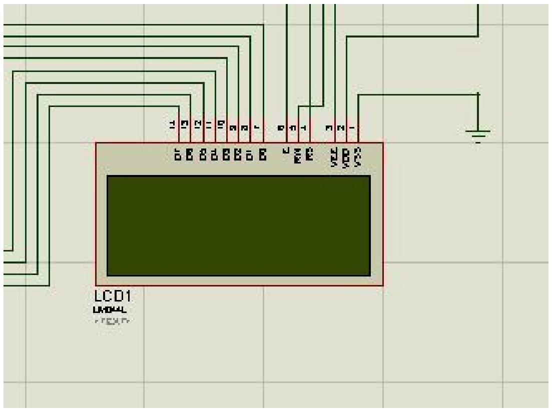
6. PROTEUS SIMULATION AND ASSEMBLY CODE

The code file (code.asm) and the proteus design file (simulation.dsm) have been mailed to the instructor at the required date.

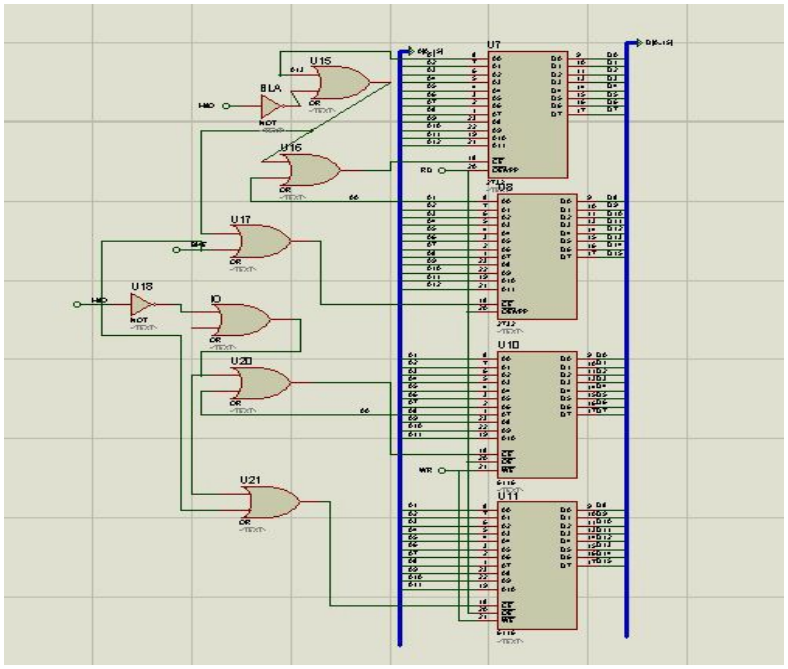
A snapshot of the final proteus design followed by individual snapshots of important components of the circuit has been given below for reference and clarity:



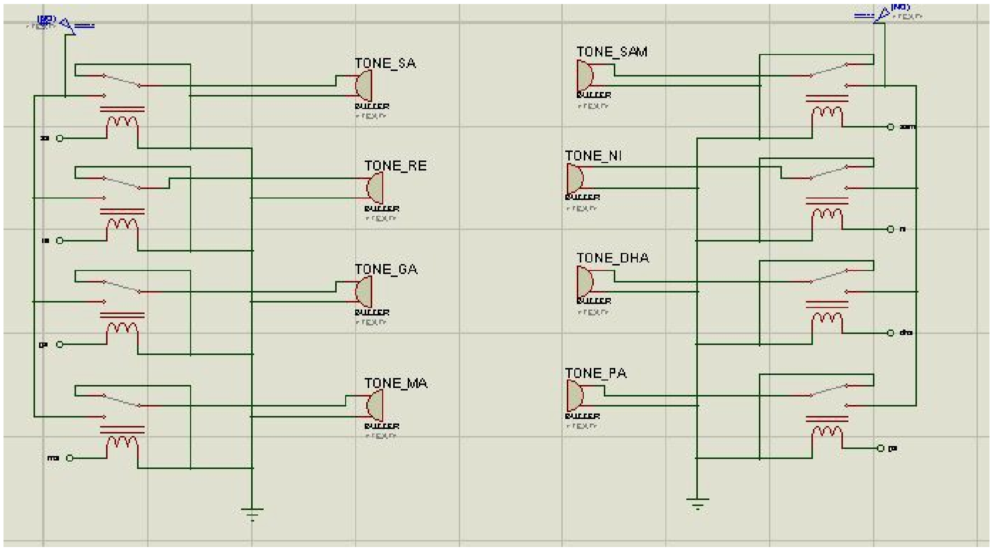
LCD (To display the time and date)



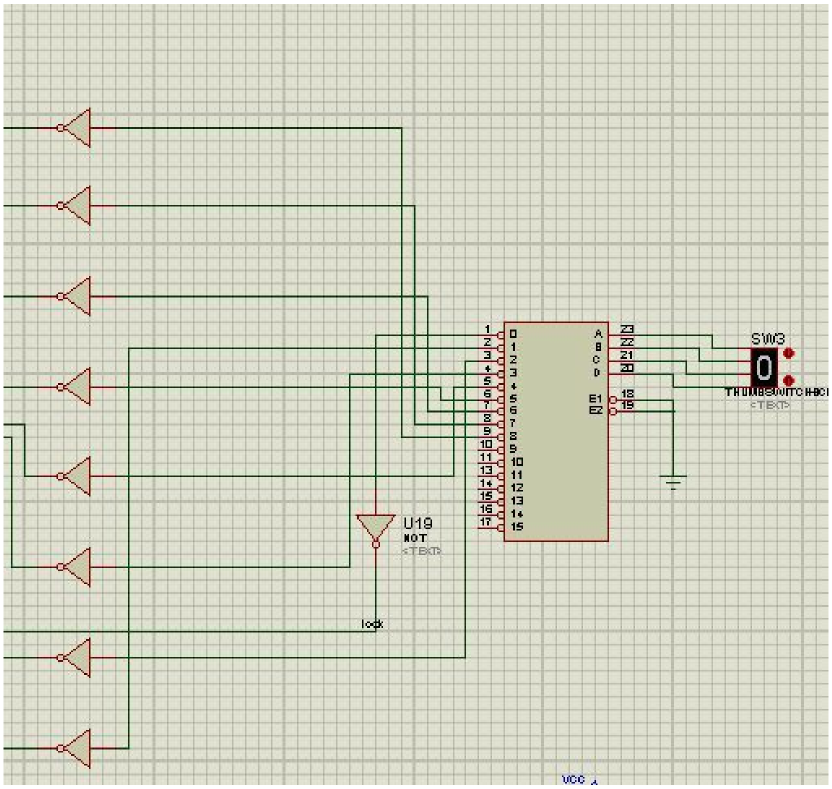
Memory (To store variables and conversion tables)



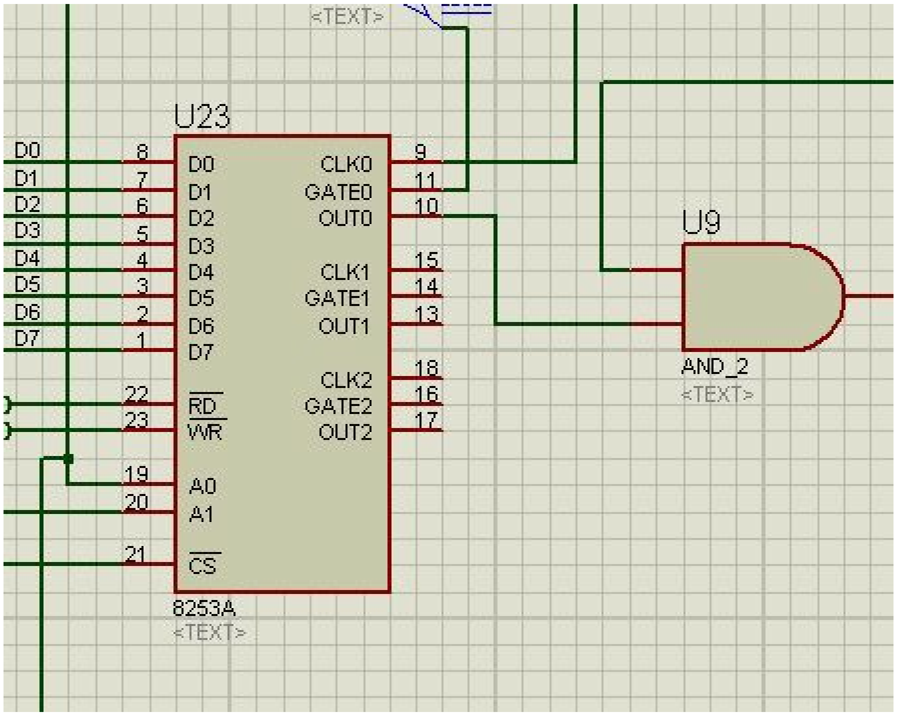
BUZZERS (To sound the alarm)



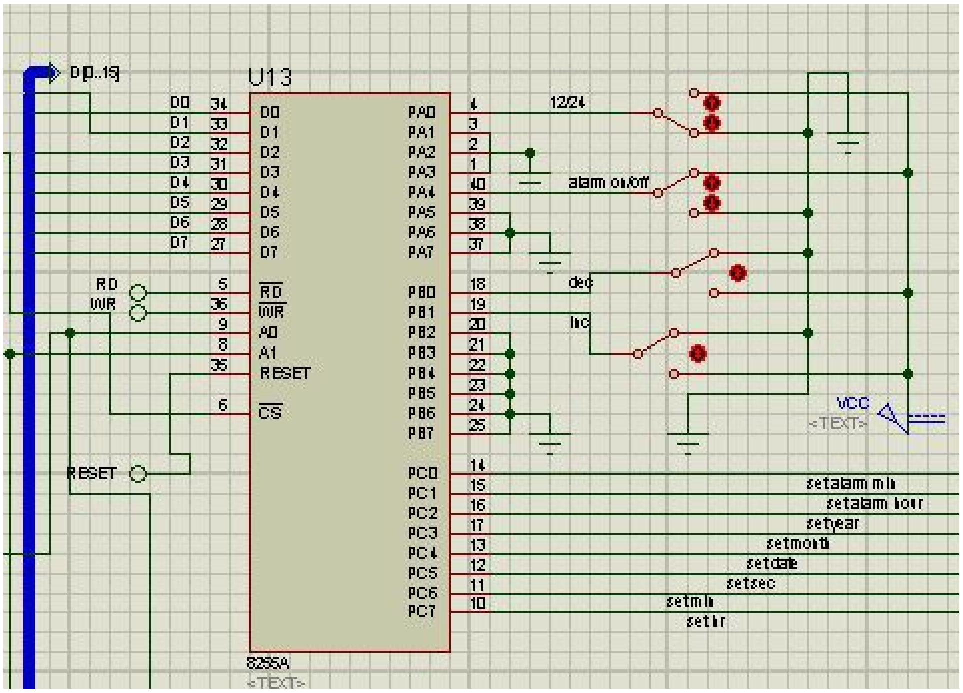
DECODER (To check status of Slider)



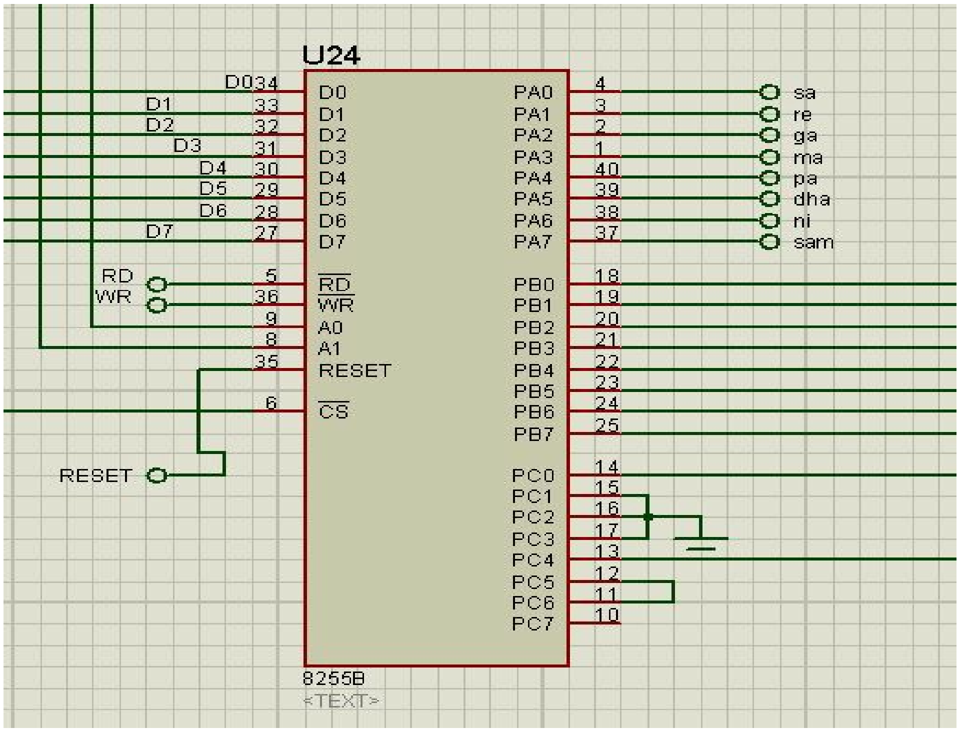
PROGRAMMABLE INTERVAL GENERATOR (To generate interrupts every second)



PROGRAMMABLE PERIPHERAL INTERFACE - A (For taking in inputs)



PROGRAMMABLE PERIPHERAL INTERFACE - B (For displaying outputs)



```
#make_bin#                                          ; .bin is a binary format like .com but allows for multiple
segments

; Initializing the values of segment registers and offset pointers in 8086

#LOAD_SEGMENT=0500h#                         ; the .bin file will be loaded to this address of 8086
#LOAD_OFFSET=0000h#

#CS=0500h#                                          ; sets the value of Code Segment Register
#IP=0000h#                                          ; sets the value of instruction pointer

#DS=0500h#                                          ; sets the value of Data Segment Register
#ES=0500h#                                          ; sets the value of Extra Segment Register

#SS=0500h#                                          ; sets the value of Stack Segment Register
#SP=FFFEh#                                          ; sets the value of Stack pointer


#AX=0000h#                                          ; intilializes the values of the general purpose registers
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
#BP=0000h#


        jmp    st1
        db   5 dup(0)
        dw start1
        dw 0000h
        db 4 dup (0)
        db 506 dup(0)
        db 506 dup(0)

st1:   cli                                     ; clear interrupt flag, because we won't be using maskable innterrupts

     mov     ax,200h               ; intialize ds, es,ss to start of RAM
     mov     ds,ax
     mov     es,ax
     mov     ss,ax
     mov     sp,0FFEH


        port1a equ 00h                       ; Variable declarations for 8255-1
        port1b equ 02h
        port1c equ 04h
        creg1 equ 06h

        port2a equ 10h                       ; Variable declarations for 8255-2
        port2b equ 12h
        port2c equ 14h
        creg2 equ 16h


        counter_0 equ 08h                    ; Variable declarations for 8253
```

```
creg equ 0Eh

mov al,10000000b                  ;programming 8255-2 - setting the required pins as inputs
out creg2,al

mov al,10011011b                  ;programming 8255-1 - setting the required pins as outputs
out creg1,al

; To run the clock, we need interrupts at an interval of 1 sec (so as to update the value of second).
; 8253 is running on 10KHz clock => We must divide by (10000)d or (2710)h in order to get a square wave of 0.5 sec up and 0.5 down
; Note that we are using mode 3, i.e. square wave generation mode

        mov     al,00110110b
        out     creg,al
        mov     al,10h
        out     counter_0,al
        mov     al,27h
        out     counter_0,al

;initialisation of lcd

       MOV AL, 38H                ;initialize LCD for 2 lines & 5*7 matrix
       out port2b,al
       mov al,01h                 ;clear LCD
       out port2c,al
       MOV AL, 00000000B          ;RS=0,R/W=0,E=0 for H-To-L pulse
       out port2c,al
       call delay_20ms

       MOV AL, 0EH                ;send command for LCD on, cursor on and no blink character
       out port2b,al
       mov al,01h
       out port2c,al
       mov al,00h
       out port2c,al
       call delay_20ms

       MOV AL, 06                 ;command for shifting cursor right
       out port2b,al
       mov al,01h
       out port2c,al
       mov al,00h
       out port2c,al
       call delay_20ms
;initialisation end

       ; Setting the default (i.e. starting values) of time, date and alarm time
       ; format_check represents whether we are using 12-hr or 24-hr clock. 1=24hr & 0=12hr
       ; If we are using 12-hr clock, phase bit shows whether we are in am or pm. 0 = am, 1 = pm

       mov second,50
       mov min,0
       mov hour,19
       mov hour_12,7
       mov format_check,0
       mov phase,1                       ; Initiazed time to 7:00:50 pm

       mov day,28
       mov month,4
       mov year,17                       ; setting date to 28/4/2017

       mov count_sec,60
       mov count_min,60
       mov count_hour,24
       mov count_day,30
       mov count_month,12         ; the total number of seconds in a min, minutes in an hour, etc.



       mov alarm_hour,19
       mov alarm_hour_12,7
       mov alarm_min,1
       mov alarm_phase,1          ; setting alarm for 7:01 pm


       mov chart_hex, 0           ; chart used to convert hex value into decimal
       mov t1,1                          ; decval = [baseaddr + hexval]
       mov t2,2
       mov t3,3
       mov t4,4
       mov t5,5
       mov t6,6
       mov t7,7
       mov t8,8
       mov t9,9
       mov t10,10
       mov t11,11
       mov t12,12
       mov t13,13
       mov t14,14
       mov t15,15
       mov t16,16
       mov t17,17
       mov t18,18
       mov t19,19
       mov t20,20
       mov t21,21
       mov t22,22
       mov t23,23
       mov t24,24
       mov t25,25
       mov t26,26
       mov t27,27
       mov t28,28
       mov t29,29
       mov t30,30
       mov t31,31
       mov t32,32
       mov t33,33
       mov t34,34
       mov t35,35
       mov t36,36
       mov t37,37
       mov t38,38
       mov t39,39
       mov t40,40
       mov t41,41
       mov t42,42
       mov t43,43
       mov t44,44
       mov t45,45
       mov t46,46
       mov t47,47
       mov t48,48
       mov t49,49
       mov t50,50
       mov t51,51
       mov t52,52
       mov t53,53
       mov t54,54
       mov t55,55
       mov t56,56
       mov t57,57
       mov t58,58
       mov t59,59
       mov t60,60
       mov t61,61
       mov t62,62


       mov chart_dec,0                          ; chart used to convert decimal value into hex
       mov d1,01h                               ; hexval = [baseaddr + decval]
       mov d2,02h
       mov d3,03h
       mov d4,04h
       mov d5,05h
       mov d6,06h
       mov d7,07h
       mov d8,08h
       mov d9,09h
       mov d10,10h
       mov d11,11h
       mov d12,12h
       mov d13,13h
       mov d14,14h
       mov d15,15h
       mov d16,16h
       mov d17,17h
       mov d18,18h
       mov d19,19h
       mov d20,20h
       mov d21,21h
       mov d22,22h
       mov d23,23h
       mov d24,24h
       mov d25,25h
       mov d26,26h
       mov d27,27h
       mov d28,28h
       mov d29,29h
       mov d30,30h
       mov d31,31h
       mov d32,32h
       mov d33,33h
       mov d34,34h
       mov d35,35h
       mov d36,36h
       mov d37,37h
       mov d38,38h
       mov d39,39h
       mov d40,40h
       mov d41,41h
       mov d42,42h
       mov d43,43h
       mov d44,44h
       mov d45,45h
       mov d46,46h
       mov d47,47h
       mov d48,48h
       mov d49,49h
       mov d50,50h
       mov d51,51h
       mov d52,52h
       mov d53,53h
       mov d54,54h
       mov d55,55h
       mov d56,56h
       mov d57,57h
       mov d58,58h
       mov d59,59h
       mov d60,60h
       mov d61,61h
       mov d62,62h

       mov chart_hex1, 0
       mov e1,1
       mov e2,2
       mov e3,3
       mov e4,4
       mov e5,5
       mov e6,6
       mov e7,7
       mov e8,8
       mov e9,9
       mov e10,10
       mov e11,11
       mov e12,12
       mov e13,13
       mov e14,14
       mov e15,15
       mov e16,16
       mov e17,17
       mov e18,18
       mov e19,19
       mov e20,20
       mov e21,21
       mov e22,22
       mov e23,23
       mov e24,24
       mov e25,25
       mov e26,26
       mov e27,27
       mov e28,28
       mov e29,29
       mov e30,30
       mov e31,31
       mov e32,32
       mov e33,33
       mov e34,34
       mov e35,35
       mov e36,36
       mov e37,37
       mov e38,38
       mov e39,39
       mov e40,40
       mov e41,41
       mov e42,42
       mov e43,43
       mov e44,44
       mov e45,45
       mov e46,46
       mov e47,47
       mov e48,48
       mov e49,49
       mov e50,50
       mov e51,51
       mov e52,52
       mov e53,53
       mov e54,54
       mov e55,55
       mov e56,56
       mov e57,57
       mov e58,58
       mov e59,59
       mov e60,60
       mov e61,61
       mov e62,62

       mov chart_dec1,0
       mov c1,01h
       mov c2,02h
       mov c3,03h
       mov c4,04h
       mov c5,05h
       mov c6,06h
       mov c7,07h
       mov c8,08h
       mov c9,09h
       mov c10,10h
```

```
                mov c11,11h
                mov c12,12h
                mov c13,13h
                mov c14,14h
                mov c15,15h
                mov c16,16h
                mov c17,17h
                mov c18,18h
                mov c19,19h
                mov c20,20h
                mov c21,21h
                mov c22,22h
                mov c23,23h
                mov c24,24h
                mov c25,25h
                mov c26,26h
                mov c27,27h
                mov c28,28h
                mov c29,29h
                mov c30,30h
                mov c31,31h
                mov c32,32h
                mov c33,33h
                mov c34,34h
                mov c35,35h
                mov c36,36h
                mov c37,37h
                mov c38,38h
                mov c39,39h
                mov c40,40h
                mov c41,41h
                mov c42,42h
                mov c43,43h
                mov c44,44h
                mov c45,45h
                mov c46,46h
                mov c47,47h
                mov c48,48h
                mov c49,49h
                mov c50,50h
                mov c51,51h
                mov c52,52h
                mov c53,53h
                mov c54,54h
                mov c55,55h
                mov c56,56h
                mov c57,57h
                mov c58,58h
                mov c59,59h
                mov c60,60h
                mov c61,61h
                mov c62,62h

                ; the main program
polling:
                in al,port1c
                cmp al,00
                jne x2                      ; Jump if the thumbswitch isn't in LOCK position
                call delay_20ms             ; Delay code execution by 20ms
                in al,port1c
                cmp al,00
                jne x2                      ;Again checking the same because of thumbswitch debounce property
                in al,port1a
                and al,10h
                cmp al,10h                  ;Checking whether the alarm is on or off.
                jne x1                      ;If alarm is not set and lock switch is on => Keep polling
                call buzzer                 ;Buzzer is called when when alarm is ON
x1:             jmp polling                 ;If alarm is not set and switch is in LOCK position => Keep polling
x2:             call debounce2          ;Since, the value is not 0 on the thumbswitch, we need to check for the value
using debounce

                ;To set Hour and Minute
                ;To check what the thumbswitch position means in form of a switch case construct.

mn0:    cmp al,80h              ;Set Hour
                jne mn1
                call set_hour
mn1:    cmp al,40h              ;Set Minute
                jne mn2
                call set_minute
mn2:    cmp al,20h              ;Set Second
                jne mn3
                call set_second
mn3:    cmp al,10h              ;Set Date
                jne mn4
                call set_date
mn4:    cmp al,08h              ;Set Month
                jne mn5
                call set_month
mn5:    cmp al,04h              ;Set Year
                jne mn6
                call set_year
mn6:    cmp al,02h              ;Set Alarm Hour
                jne mn7
                call set_alarm_hour
mn7:    cmp al,01h              ;Set Alarm Min
                jne mn8
                call set_alarm_min
mn8:            jmp polling     ; After one of them has been done, we need to repeat the polling process

;end of main program.


        ; ISR associated with NMI (given by 8253) => Compares seconds, minutes, .... , year increments them if
necessary
        ;For ex., if second reaches 60 => reset it to 0 and increment minute and so on for hour, day, month and year
start1:
        mov al,second                           ;inc second
        inc al
        mov second,al
        cmp al,count_sec                        ; see if the seconds have reached 60
        jne y1                                          ; if seconds has not reached 60 => only seconds will
```

```
change => y1 calls display subroutine

        mov second,00                       ; Else inc minutes and check if we need to increment hours
        mov al,min                                  ; If yes, then proceed to next part
        inc al
        mov min,al
        mov al,min
        cmp al,count_min
        jne y1                                      ; Else, y1 calls display subroutine

        mov min,00                                  ; Increment hour if required
        mov al,hour
        inc al
        mov hour,al
        cmp al,count_hour
        jne y1                                      ;If only the hour will change (i.e. day will not) y1 calls
display subroutine

        mov hour,00                                 ; Else put 00 in hour and increment day
        mov al,day
        inc al
        mov day,al
        mov al,day
        cmp al,count_day                    ; Compare day with 30
        jne y1                                      ; If month has not ended => y1 calls display subroutine

        mov day,1                                   ; Else put 01 in day and increment month
        mov al,month
        inc al
        mov month,al
        mov al,month
        cmp al,count_month                  ; compare month with 12
        jne y1                                      ; if month is less than 12 => y1 calls display subroutine

        mov month,1                                 ; Else increment year
        mov al,year
        inc al
        mov year,al
y1:     call display
        iret

; end of ISR associated with NMI. (given by 8253)

; procedure for delaying sequential execution of program by 20ms
delay_20ms proc near
                                push    cx
                                mov     cx,900d

dl1:                    nop
                                loop    dl1

                                pop     cx
                                ret
delay_20ms endp
```

```
;procedure to display clock
display proc near

        ;format_check => 1 = 24hr & 0 = 12hr
        ;phase => 0-am & 1-pm

                ;Putting format_check = whatever value that has been set in
                in al,port1a
                and al,01h
                mov format_check,al

                ;clearing screen
                mov al,01h
                out port2b,al
                mov al,01h
                out port2c,al
                mov al,00h
                out port2c,al
                call delay_20ms


                ;checking if 24_hr or 12_hr ( by using the value of format_check)
                mov al,format_check
                cmp al,1
                jne hr12                        ; jump if the format is 12 hr format

                ;display the 24 hr format time
                lea si,chart_hex
                mov cx,0
                mov al,hour

; Usage of charts : The hour value we have is in dec and we need to convert it into hex
; Hence, we get the location from chart_hex and at that location in chart_dec
; Our hex value corresponding to that dec value will be present and it will go into cx

hour_1:
                cmp al,[si]
                je hour1
                inc cx
                inc si
                jmp hour_1

hour1:
                lea di,chart_dec
                add di,cx
                mov al,[di]                     ; the hex value to be displayed is now al
                mov digit,al                ; moving the digit to al

                and al,0f0h                     ;masking units digit, because we'll start with printing the tens digit first
                mov cl,4
                rol al,cl
                add al,30h                      ; To get the corresponding ASCII value

                out port2b,al               ;displaying tens digit
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms


                mov al,digit                    ;masking tens digit, since now we have to display the units digit of hour
                and al,0fh
                add al,30h                              ; To get the corresponding ASCII value

                out port2b,al                   ;displaying ones digit
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms

                ; displaying time in 24 hour format completed
                ; we will jump to printing of ':'
                jmp skip

        ;       displaying hour in 12 hour format

hr12:mov al,hour
        cmp al,00                                       ;Hour is 0 => hour must be 12 am
        jne a1                                          ;Hence, phase = 0, since am
        mov hour_12,12
        mov phase,0
        jmp exit                                        ;Jumping to displaying  the 12 hour time format

a1:     cmp al,12                                       ;If it is 12 => it has to be pm
        jne a4
        mov hour_12,12
        mov phase,1
        jmp exit

a4:     cmp al,12                                       ;If it's below 12, display as it is and in am
        ja a2
        mov hour_12,al
        mov phase,0
        jmp exit

a2:     mov bl,12                                       ;If it's above 12, display after subtracting 12 and in pm
        sub al,bl
        mov hour_12,al
        mov phase,1

; Since we have the hour value in dec format, we use charts as described previously to find the corresponding hex value


exit:   lea si,chart_hex
                mov cx,0
                mov al,hour_12

dh_1:   cmp al,[si]
```

```
                je h1
                inc cx
                inc si
                jmp dh_1
h1:             lea di,chart_dec
                add di,cx
                mov al,[di]
                mov digit,al                    ; the hex value is now in digit

                and al,0f0h                             ;masking units digit
                mov cl,4
                rol al,cl
                add al,30h

                out port2b,al                   ;displaying tens digit
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms

                mov al,digit                    ;masking tens digit
                and al,0fh
                add al,30h

                out port2b,al                   ;displaying ones digit
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
        ;display_hour_12 completed

        ;displaying ':'
skip:           mov al,3ah              ;Ascii value of ':' is 3a
                out port2b,al           ; displaying ':'
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms


                lea si,chart_hex ;displaying min => The logic employed is similar to displaying hour, we just don't have
2 different formats now
                mov cx,0
                mov al,min
min_1:  cmp al,[si]                     ;min_1 finds out the hex value of minute in chart_hex
                je min1
                inc cx
                inc si
                jmp min_1
min1:   lea di,chart_dec                ;actually masks value and displays it
                add di,cx
                mov al,[di]
                mov digit,al
```

```
                                                ;masking units digit
                and al,0f0h
                mov cl,4
                rol al,cl
                add al,30h
                                                ;displaying tens digit

                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
                                                ;masking tens digit

                mov al,digit
                and al,0fh
                add al,30h
                                                ;displaying ones digit

                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
                                                ;minutes have been displayed completed


                ;displaying ':'
                mov al,3ah
                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al

                ;display sec
                lea si,chart_hex
                mov cx,0
                mov al,second
sec_1:          cmp al,[si]             ;Same concept as above sec_1 find hex value of second

                je sec1
                inc cx
                inc si
                jmp sec_1


sec1:   lea di,chart_dec        ;Actually displays the second value
                add di,cx
                mov al,[di]
                mov digit,al
                                                ;masking units digit
                and al,0f0h
                mov cl,4
                rol al,cl
                add al,30h
                                                ;displaying tens digit
                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
                                                        ;masking tens digit
                mov al,digit
                and al,0fh
                add al,30h
                                                        ;displaying ones digit
                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
                                                        ;display_sec completed

        ;Checking for format again because if the format is 24hr, we don't need to display am/pm.
                mov al,format_check
                cmp al,1
                je skip2

                                                        ;checking if am or pm
                mov al,phase
                cmp al,1
                je pm1

                ;Displaying 'am' using ASCII values and sequentially inputting them
                mov al,41h
                out port2b, al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms

                mov al,4dh
                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
                jmp skip2

                ;Displaying 'pm' using ASCII values and sequentially inputting them

pm1:
                mov al,50h
                out port2b, al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
                mov al,4Dh
                out port2b, al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms


skip2:
                ;moving to next line where we will display date/year

                mov al,11000000b
                out port2b,al
                mov al,01h
                out port2c,al
                mov al,00h
                out port2c,al
                call delay_20ms

                ;Displaying date
                lea si,chart_hex
                mov cx,0
                mov al,day
day_1:  cmp al,[si]
                je day1
                inc cx
                inc si
                jmp day_1

day1:   lea di,chart_dec
                add di,cx
                mov al,[di]
                mov digit,al

                and al,0f0h
                mov cl,4
                rol al,cl
                add al,30h
                                                        ;displaying tens digit
                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
                                                        ;masking_ones digit
                mov al,digit
                and al,0fh
                add al,30h
                                                        ;displaying ones digit
                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
                ;display_day completed

                ;displaying ':'
                mov al,3ah
                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms

                ;display month
                lea si,chart_hex
                mov cx,0
                mov al,month
mon_1:cmp al,[si]
                je mon1
                inc cx
                inc si
                jmp mon_1
mon1:   lea di,chart_dec
                add di,cx
                mov al,[di]
                mov digit,al

                and al,0f0h                     ;masking units digit
                mov cl,4
                rol al,cl
                add al,30h

                out port2b,al           ;displaying tens digit
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms

                mov al,digit            ;masking tens digit
                and al,0fh
                add al,30h

                out port2b,al           ;displaying ones digit
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
                ;display_month completed

                ;displaying ':'
                mov al,3ah
                out port2b,al
                mov al,11h
                out port2c,al
```

```
          mov al,10h
          out port2c,al
          call delay_20ms

          ;display year
          ;NOTE: We assumed that the year will be of the format 20XX;

          ;display 2
          mov al,32h
          out port2b,al
          mov al,11h
          out port2c,al
          mov al,10h
          out port2c,al
          call delay_20ms

          ;display 0
          mov al,30h
          out port2b,al
          mov al,11h
          out port2c,al
          mov al,10h
          out port2c,al
          call delay_20ms


          ;display year_last 2 digits
          lea si,chart_hex
          mov cx,0
          mov al,year
year_1: cmp al,[si]
          je year1
          inc cx
          inc si
          jmp year_1
year1:  lea di,chart_dec
          add di,cx
          mov al,[di]
          mov digit,al

          and al,0f0h                         ;masking ones digit
          mov cl,4
          rol al,cl
          add al,30h
                                                ;displaying tens digit
          out port2b,al
          mov al,11h
          out port2c,al
          mov al,10h
          out port2c,al
          call delay_20ms
                                                ;masking tens digit
          mov al,digit
          and al,0fh
          add al,30h
                                                ;displaying ones digit
```

```
                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms

                ;display_year completed





                ret
        display endp                            ;Display procedure ends here



        ;procedure to set hour
set_hour proc near
  call display

sh1:
        in al,port1c
        cmp al,80h
        jnz sh2                                         ; if set hour is not high then ret (sh2 => return)
        call debounce3_hour             ; if set hour is indeed high => we have to update the value of hour, then call
debounce3_hour
        in al,port1b
        cmp al,01h                                      ;To check if we have to increment or decrement
        jnz sh3                                         ;jump if increment
        mov bl,hour
          dec bl
        cmp bl,00
        jge sh5                         ; if hour is becomes less then 0, then it was earlier => make it 23


        mov bl,23                       ;If the hour value goes lower than 0 => make it 23
sh5:
        mov hour,bl
                                                ;Setting the hour_12 value
        mov al,hour
                cmp al,00               ; if al = 0 => it is 12 AM
                jne shr01
                mov hour_12 , 12
                mov phase,0
                jmp exit1
shr01:
                cmp al,12
                jne shr04               ; if al = 12 => it is 12 PM
                mov hour_12,12
                mov phase,1
                jmp exit1
shr04:
                cmp al,12               ; if al < 12 => it is am
```

```
                    ja shr02
                    mov hour_12,al
                    mov phase,0
                    jmp exit1
shr02:
                    sub al,12
                    mov hour_12,al
                    mov phase,1
                                                  ;To display the hour value we have computed
exit1:    call display
                    jmp sh1                       ;Because we need to decrement till dec is pressed => check again
sh3:
                    ;To check if we have to increment
                    cmp al,02h
                    jne sh1             ;If not => go and check the value of port1c again
                    mov bl,hour         ;Otherwise, increment
                    inc bl
                    cmp bl,24
                    jb sh6
                    mov bl,00           ;If it has become 24 => make it 0 because hour can't be > 23
sh6:      mov hour,bl
                                                  ;setting the value of hour_12
                    mov al,hour
                    cmp al,00
                    jne shr01
                    mov hour_12 , 12
                    mov phase,0
                    jmp exit2
shr11:              cmp al,12
                    jne shr04
                    mov hour_12,12
                    mov phase,1
                    jmp exit2
shr14:              cmp al,12
                    ja shr02
                    mov hour_12,al
                    mov phase,0
                    jmp exit2
shr12:              sub al,12
                    mov hour_12,al
                    mov phase,1
                                                  ;incrementing hour done end
exit2:    call display
                    jmp sh1
sh2:                ret                 ;End of set_hour

set_hour endp
          ;set_hour_debounce
                                        ; called when the value of set hour signal is indeed high, i.e. we must update the
value hour
debounce3_hour  proc near
                                        ;debounce3
inc1_1:  in al,port1a                   ;Checks format
                    and al,01h
                    mov bl,format_check ;Checks if the format is same
                    cmp al,bl
```

```
                je bro2_1                               ; if the formats are same, jump to bro2_1
                call display                    ; else format is diff. so we need to display the new value
bro2_1: in al,port1b
                cmp al,00
                jne inc1_1                      ; jump if not 0, i.e. either inc or dec is pressed
                in al,port1a
                and al,01h                      ;FORMAT'S VALUE IS IN AL
                mov bl,format_check
                cmp al,bl                               ;COMPARE FORMATS VALUE
                je inc2_1                               ;IF EQUAL JUMP TO INC2_1
                call display
inc2_1:
                in al,port1c                    ;CHECK IF SET HR HAI
                cmp al,80h
                jne trol_1                              ;NOT HAI TOH RETURN
                in al,port1a
                and al,01h
                mov bl,format_check
                cmp al,bl                               ;FORMAT CHECK AGAIN
                je inc3_1
                call display

inc3_1: in al,port1b                            ;Checking again and again(debounce)to confirm a valid press
        cmp al,00
        je inc2_1
        mov bl,al
        call delay_20ms
        in al,port1b
        cmp al,bl
        jne inc2_1
trol_1:ret
debounce3_hour endp
                                                ;set_month(Again the logic is similar to set hour)
set_month proc near
  call display
mo1:    in al,port1c
        cmp al,08h
        jne mo2
        call debounce3_month
        in al,port1b
        cmp al,01h
        jne mo3
        mov bl,month
        dec bl
        cmp bl,0
        jne mo4
        mov bl,12
mo4:    mov month,bl                            ;Decrement
        call display
        jmp mo1
mo3:    cmp al,02h
        jne mo1
        mov bl,month                                    ;Increment
        inc bl
        cmp bl,13
        jne mo5
        mov bl,1
mo5:    mov month,bl
        call display
        jmp mo1
mo2:    ret
set_month endp


                                                                ;set_minute function starts      ( The entire
logic is similar to set hour)
set_minute proc near

  call display

m1:     in al,port1c
        cmp al,40h
        jnz m2
        call debounce3_min
        in al,port1b
        cmp al,01h
        jnz m3
        mov bl,min                      ;Decrement
        cmp bl,0                        ;If min value = 0 => make it 60 so that it
        jnz m4                          ;doesn't go negative when we decrement
        add bl,60
m4:     dec bl
        mov min,bl
        call display
        jmp m1

m3:     cmp al,02h
        jnz m1                          ;Increment
        mov bl,min
        inc bl
        cmp bl,60
        jnz m5
        mov bl,0
m5:     mov min,bl
        call display
        jmp m1
m2:     ret
set_minute endp

                                                  ;set_date(Again the logic is similar to set hour)
set_date proc near
  call display
da1:    in al,port1c
        cmp al,10h
        jne da2
        call debounce3_day

        in al,port1b
        cmp al,01h
        jne da3

        mov bl,day
        dec bl
        cmp bl,0
        jne da4
        mov bl,count_day
da4:
        mov day,bl                                 ;Decrement
        call display
        jmp da1
da3:    cmp al,02h
        jne da1
        mov bl,day
        cmp bl,count_day                    ;Increment
        jne da5
        mov bl,0
da5:    inc bl
        mov day,bl
        call display
        jmp da1
da2:    ret
set_date endp

                                            ;set_second    (Again the logic is similar to set hour)
set_second proc near
  call display
s1:     in al,port1c
        cmp al,20h
        jnz s2
        call debounce3_sec
        in al,port1b
        cmp al,01h
        jnz s3
        mov bl,second
        cmp bl,0
        jnz s4
        add bl,60
s4:     dec bl                      ;Decrement
        mov second,bl
        call display
        jmp s1

s3:     cmp al,02h
        jnz s1                      ;Increment
        mov bl,second
        inc bl
        cmp bl,60
        jnz s5
        mov bl,0
s5:     mov second,bl
        call display
        jmp s1

s2:     ret

set_second endp
```

```
                                                            ;set_year(Again the logic is similar to set hour)
set_year proc near
   call display
ye1:      in al,port1c
          cmp al,04h
          jne ye2
          call debounce3_year
          in al,port1b
          cmp al,01h
          jne ye3
          mov bl,year
          cmp bl,00
          jne ye4
          inc bl
ye4:      dec bl                         ;Decrement
          mov year,bl
          call display
          jmp ye1
ye3:      cmp al,02h
          jne ye1
          mov bl,year

          cmp bl,99
          jne ye5
          dec bl                                 ;Increment
ye5:      inc bl
          mov year,bl
          call display
          jmp ye1
ye2:      ret
set_year endp



                                                 ;set_min_debouce
debounce3_min proc near
                                                 ;debounce3
inc1_2:  in al,port1a
                    and al,01h
                    mov bl,format_check
                    cmp al,bl
                    je bro2_2
                    call display
bro2_2: in al,port1b
                    cmp al,00
                    jne inc1_2
                    in al,port1a
                    and al,01h
                    mov bl,format_check
                    cmp al,bl
                    je inc2_2
                    call display
inc2_2: in al,port1c
                    cmp al,40h
                    jne trol_2
                    in al,port1a
                    and al,01h
                    mov bl,format_check
                    cmp al,bl
                    je inc3_2
                    call display

inc3_2:  in al,port1b
         cmp al,00
         je inc2_2
         mov bl,al
         call delay_20ms
         in al,port1b
         cmp al,bl
         jne inc2_2
trol_2:  ret
debounce3_min endp
                                                ;set_sec_debouce
debounce3_sec proc near
                                                ;debounce3
inc1_3:  in al,port1a
                    and al,01h
                    mov bl,format_check
                    cmp al,bl
                    je bro2_3
                    call display
bro2_3:in al,port1b
                    cmp al,00
                    jne inc1_3
                    in al,port1a
                    and al,01h
                    mov bl,format_check
                    cmp al,bl
                    je inc2_3
                    call display
inc2_3: in al,port1c
                    cmp al,20h
                    jne trol_3
                    in al,port1a
                    and al,01h
                    mov bl,format_check
                    cmp al,bl
                    je inc3_3
                    call display

inc3_3:  in al,port1b
         cmp al,00
         je inc2_3
         mov bl,al
         call delay_20ms
         in al,port1b
         cmp al,bl
         jne inc2_3
trol_3:  ret
debounce3_sec endp
```

```
                                   ;set_date_debounce
debounce3_day proc near
                                   ;debounce3
inc1_4:  in al,port1a
              and al,01h
              mov bl,format_check
              cmp al,bl
              je bro2_4
              call display
bro2_4: in al,port1b
              cmp al,00
              jne inc1_4
              in al,port1a
              and al,01h
              mov bl,format_check
              cmp al,bl
              je inc2_4
              call display
inc2_4: in al,port1c
              cmp al,10h
              jne trol_4
              in al,port1a
              and al,01h
              mov bl,format_check
              cmp al,bl
              je inc3_4
              call display

inc3_4:  in al,port1b
         cmp al,00
         je inc2_4
         mov bl,al
         call delay_20ms
         in al,port1b
         cmp al,bl
         jne inc2_4
trol_4:  ret
debounce3_day endp
                                          ;set_month_debouce
debounce3_month proc near
                                          ;debounce3
inc1_5:  in al,port1a
              and al,01h
              mov bl,format_check
              cmp al,bl
              je bro2_5
              call display
bro2_5: in al,port1b
              cmp al,00
              jne inc1_5
              in al,port1a
              and al,01h
              mov bl,format_check
              cmp al,bl
              je inc2_5
              call display
```

```
inc2_5: in al,port1c
                cmp al,08h
                jne trol_5
                in al,port1a
                and al,01h
                mov bl,format_check
                cmp al,bl
                je inc3_5
                call display

inc3_5:  in al,port1b
         cmp al,00
         je inc2_5
         mov bl,al
         call delay_20ms
         in al,port1b
         cmp al,bl
         jne inc2_5
trol_5:  ret
debounce3_month endp
                                                ;set_year_debounce
debounce3_year proc near
                                                ;debounce3
inc1_6:  in al,port1a
                and al,01h
                mov bl,format_check
                cmp al,bl
                je bro2_6
                call display
bro2_6:  in al,port1b
                cmp al,00
                jne inc1_6
                in al,port1a
                and al,01h
                mov bl,format_check
                cmp al,bl
                je inc2_6
                call display
inc2_6: in al,port1c
                cmp al,04h
                jne trol_6
                in al,port1a
                and al,01h
                mov bl,format_check
                cmp al,bl
                je inc3_6
                call display

inc3_6:  in al,port1b
         cmp al,00
         je inc2_6
         mov bl,al
         call delay_20ms
         in al,port1b
         cmp al,bl
         jne inc2_6
```

```
trol_6:   ret
debounce3_year endp
                                            ;set_alarmhour_debounce
debounce1_alarm_hour proc near
                                            ;debounce
linc1_7:in al,port1a
        and al,01h
        mov bl,format_check
        cmp al,bl
        je bro1_7                           ; if format is not equal then call display function
        call alarm_display
bro1_7: in al,port1b
                cmp al,00                   ; if neither inc or dec is pressed, go back until either of them is pressed
                jne linc1_7

                in al,port1a
                and al,01h
                mov bl,format_check
                cmp al,bl
                je linc2_7                                  ; if the format is changed, display again ( this is done again after
we ensured that either inc or dec is pressed)
                call alarm_display
linc2_7:in al,port1c
                cmp al,02h
                jne trol_7                                  ; if set alarm hour is not pressed, then return
                in al,port1a
                and al,01h
                mov bl,format_check
                cmp al,bl
                je linc3_7                                  ; if format is not same then call display
                call alarm_display

linc3_7:in al,port1b
                cmp al,00
                je linc2_7                                  ; if inc or dec is not pressed, then loop up until either of them is
pressed
                mov bl,al
                call delay_20ms
                in al,port1b
                cmp al,bl                                   ; again check after 20 ms if key is still pressed.
                jne linc2_7                                 ; by now we have ensured that the keys were indeed pressed
trol_7:   ret
debounce1_alarm_hour endp
                                                    ;set_alarmmin_debouce
debounce1_alarm_min proc near
                                                    ;debounce
linc1_8:in al,port1a
        and al,01h
        mov bl,format_check
        cmp al,bl
        je bro1_8
        call alarm_display
bro1_8: in al,port1b
                cmp al,00
                jne linc1_8
                in al,port1a
                and al,01h
                mov bl,format_check
                cmp al,bl
                je linc2_8
                call alarm_display
linc2_8:in al,port1c
                cmp al,01h
                jne trol_8
                in al,port1a
                and al,01h
                mov bl,format_check
                cmp al,bl
                je linc3_8
                call alarm_display

linc3_8:in al,port1b
                cmp al,00
                je linc2_8
                mov bl,al
                call delay_20ms
                in al,port1b
                cmp al,bl
                jne linc2_8
trol_8: ret
debounce1_alarm_min endp
                                        ;debounce end
                                        ;debounce2 for switch

        ; if something other than lock position is found on the thumbswitch in main program, we ensure that this is
user-intended and not noise by debounce
debounce2 proc near
deb:    in al,port1c
                mov bl,al
                call delay_20ms
                in al,port1c
                cmp al,bl                               ; if the two values are diff. => it was noise => go back to polling
                jne polling
                ret                                             ; else proceed

debounce2       endp

debounce1 proc near
                                        ;debounce
linc1:in al,port1a
        and al,01h
        mov bl,format_check
        cmp al,bl
        je bro1
        call alarm_display
bro1:   in al,port1b
                cmp al,00
                jne linc1

                in al,port1a
                and al,01h
```

```
                    mov bl,format_check
                    cmp al,bl
                    je linc2
                    call alarm_display
linc2:    in al,port1a
                    and al,01h
                    mov bl,format_check
                    cmp al,bl
                    je linc3
                    call alarm_display

linc3:    in al,port1b
                    cmp al,00
                    je linc2
                    mov bl,al
                    call delay_20ms
                    in al,port1b
                    cmp al,bl
                    jne linc2
          ret
debounce1 endp
                                                            ;set_alarm_hour

set_alarm_hour proc near

mov alarm_hour,00                                 ; default values of set_alarm hour
mov alarm_hour_12,12
mov alarm_phase,0

alh1:
          in al,port1c                            ;Check if set alarm_hour is acctive set or not
          cmp al,02h
          jne alh2                                          ;If set alarm hour is not selected, end the procedure.

          in al,port1a                            ;Checking alarm = on or off
          and al,10h
          cmp al,10h
          jne alh2                                          ;If off = end the procedure

          call alarm_display

          call debounce1_alarm_hour                         ;RSVDEL DISPLAY CHECK KIYA
                                                                              ; we ensured that either increment of
decremment is pressed
          in al,port1b
          cmp al,01h                                        ;Are we decrementing?
          jnz alh3                                          ; if not decrementing move to alh3
          mov bl,alarm_hour
          dec bl                                            ;decrement alarm_hour
          cmp bl,00
          jge alh5
          mov bl,23                                         ; if the hour becomes less than 0, make it 23

alh5:
          mov alarm_hour,bl
                                                                    ;Setting the alarm_hour for 12hr format
```

```
        mov al,alarm_hour
                cmp al,00
                jne alha1
                mov alarm_hour_12 , 12
                mov alarm_phase,0
                jmp exita1
alha1:
                cmp al,12
                jne alha4
                mov alarm_hour_12,12
                mov alarm_phase,1                               ;Taking care of the rollovers after 12 in 12hr format, etc
                jmp exita1
alha4:          cmp al,12
                ja alha2
                mov alarm_hour_12,al
                mov alarm_phase,0
                jmp exita1
alha2:          sub al,12
                mov alarm_hour_12,al
                mov alarm_phase,1
                                                        ;end
exita1: call alarm_display                              ;Need to display the time as it is being decremented
                jmp alh1                                                ;If we are supposed to decrement further
alh3:           cmp al,02h                                      ;if we are incrementing (same logic as decrementing

                jne alh1
                mov bl,alarm_hour
                inc bl
                cmp bl,24
                jb alh6
                mov bl,00
alh6:           mov alarm_hour,bl
                ;12hr
                mov al,alarm_hour
                cmp al,00
                jne alhb1                                               ;Taking care of the rollovers after 12 in 12hr
format, etc
                mov alarm_hour_12 , 12
                mov alarm_phase,0
                jmp exita2
alhb1:          cmp al,12
                jne alhb4
                mov alarm_hour_12,12
                mov alarm_phase,1
                jmp exita2
alhb4:          cmp al,12
                ja alhb2
                mov alarm_hour_12,al
                mov alarm_phase,0
                jmp exita2
alhb2:          sub al,12
                mov alarm_hour_12,al
                mov alarm_phase,1
                                                                ;end
exita2: call alarm_display
                jmp alh1                                ; if we are supposed to inncrement more than once
```

```
alh2:   ret

set_alarm_hour endp

                                                        ;set_alarm_min (same logic as set_alarm_hour)
set_alarm_min proc near
        mov alarm_min,00

al1:    in al,port1a
                and al,10h                              ;checking if set_alarm minute is on
                cmp al,10h
                jne al2                                 ;end if off
                in al,port1c
                cmp al,01h
                jnz al2
                call alarm_display
                call debounce1_alarm_min      ;call debounce
                in al,port1b
                in al,port1b
                cmp al,01h                              ; decrement
                jnz al3
                mov bl,alarm_min
                dec bl
                cmp bl,00
                jge al5
                mov bl,59
al5:            mov alarm_min,bl
                call alarm_display
                jmp al1                                 ;To check if we have to decrement further
al3:            cmp al,02h
                jne al1                                 ;here we increment
                mov bl,alarm_min
                inc bl
                cmp bl,60
                jne al6
                mov bl,00
al6:            mov alarm_min,bl
                call alarm_display
                jmp al1                                 ;if we have to increment more than once
al2:            ret



                set_alarm_min endp
;end of set_alarm_min procedure


        ; procedure to make the buzzer ring in the required sequence
buzzer proc near
                mov al,alarm_hour
                mov ah,hour
                cmp al,ah                                       ;if current hour is not equal to alarm_hour, then
quit
                jne esc1                                        ;else, check minutes
                mov al,alarm_min                                ; if current minutes is not equal to alarm_minute, then
quit
                mov ah,min                                              ; else, we must ring buzzer
                cmp al,ah
                jne esc1


                mov al,01h
                out port2a,al
buzz:
                in al,port1c
                cmp al,00                                       ;if all switches are off, then move forward. Else return
                jne esc1
                in al,port1a                            ;checking if alarm is on
                and al,10h
                cmp al,10h
                jne esc1                                        ; if not on, return
                mov ah,min                                      ; again comparing minutes with alarm_minutes ( this
will help terminate the ringing when the minute is over)
                cmp ah,alarm_min                        ; if now min != alarm_min => end procedure
                ;mov al,01h
                ;out port2a,al

                mov al, 01h
                out port2a, al

                push cx

                mov cx, 50
        sa:                                                             ; ring the buzzer of sa for 1 second
                call delay_20ms
                loop sa
                ; pop cx

                mov al, 02h                                     ; ring the buzzer of re for 1 second
                out port2a, al
                ; push cx
                mov cx, 50
        re:
                call delay_20ms
                loop re
                ; pop cx

                mov al, 04h                                     ; ring the buzzer of ga for 1 second
                out port2a, al
                ; push cx
                mov cx, 50
        ga:
                call delay_20ms
                loop ga
                ; pop cx

                mov al, 8                                       ; ring the buzzer of ma for 1 second
                out port2a, al
                ; push cx
                mov cx, 50
        ma:
                call delay_20ms
                loop ma
                ; pop cx


                mov al, 16                              ; ring the buzzer of pa for 1 second
                out port2a, al
                ; push cx
                mov cx, 50
        pa:
                call delay_20ms
                loop pa
                ; pop cx

                mov al, 32                              ; ring the buzzer of dha for 1 second
                out port2a, al
                ; push cx
                mov cx, 50

        dha:
                call delay_20ms
                loop dha
                ; pop cx

                mov al, 64                              ; ring the buzzer of ni for 1 second
                out port2a, al
                ; push cx
                mov cx, 50

        ni:
                call delay_20ms
                loop ni
                ; pop cx

                mov al, 128                             ; ring the buzzer of sa (major) for 1 second
                out port2a, al
                ; push cx
                mov cx, 50
        sam:
                call delay_20ms
                loop sam

                pop cx

                je buzz                                 ; if min == alarm_min, keep rining the buzzer

esc1:           mov al,00h
                out port2a,al                   ;Stop buzzer sound

                ret
buzzer endp

alarm_display proc near

        ;checking format
        in al,port1a
        and al,01h
        mov format_check,al
        ;display alarm program
        ;clearing screen
                mov al,01h
                out port2b,al
                mov al,01h
                out port2c,al
                mov al,00h
                out port2c,al
                call delay_20ms

        ; display hour
        ;checking format
                mov al,format_check
                cmp al,1
                jne ahr12

                ;display alarm_hour_24
                lea si,chart_hex1
                mov cx,0
                mov al,alarm_hour
ahour_1:cmp al,[si]
                je ahour1
                inc cx
                inc si
                jmp ahour_1
ahour1:         lea di,chart_dec1
                add di,cx
                mov al,[di]
                mov digit,al
                ;masking _tens digit
                and al,0f0h
                mov cl,4
                rol al,cl
                add al,30h
                ;displaying tens digit
                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
                ;masking_ones digit
                mov al,digit
                and al,0fh
                add al,30h
                ;displaying ones digit
                out port2b,al
                mov al,11h
                out port2c,al
                mov al,10h
                out port2c,al
                call delay_20ms
                ;display alarm_hour_24 completed
```

```
                        jmp skip3

                        ;display alarm_hour_12
ahr12:  lea si,chart_hex1
                        mov cx,0
                        mov al,alarm_hour_12
ahour_2:
                        cmp al,[si]
                        je ahour2
                        inc cx
                        inc si
                        jmp ahour_2
ahour2:                 lea di,chart_dec1
                        add di,cx
                        mov al,[di]
                        mov digit,al
                        ;masking _tens digit
                        and al,0f0h
                        mov cl,4
                        rol al,cl
                        add al,30h
                        ;displaying tens digit
                        out port2b,al
                        mov al,11h
                        out port2c,al
                        mov al,10h
                        out port2c,al
                        call delay_20ms
                        ;masking_ones digit
                        mov al,digit
                        and al,0fh
                        add al,30h
                        ;displaying ones digit
                        out port2b,al
                        mov al,11h
                        out port2c,al
                        mov al,10h
                        out port2c,al
                        call delay_20ms
                        ;display alarm_hour_12 completed

                        ;displaying ':'
skip3:  mov al,3ah
                        out port2b,al
                        mov al,11h
                        out port2c,al
                        mov al,10h
                        out port2c,al
                        call delay_20ms

                        ;display alarm_min
                        lea si,chart_hex1
                        mov cx,0
                        mov al,alarm_min
amin_1:cmp al,[si]
```

```
                  je amin1
                  inc cx
                  inc si
                  jmp amin_1
amin1:  lea di,chart_dec1
                  add di,cx
                  mov al,[di]
                  mov digit,al
                  ;masking _tens digit
                  and al,0f0h
                  mov cl,4
                  rol al,cl
                  add al,30h
                  ;displaying tens digit
                  out port2b,al
                  mov al,11h
                  out port2c,al
                  mov al,10h
                  out port2c,al
                  call delay_20ms
                  ;masking_ones digit
                  mov al,digit
                  and al,0fh
                  add al,30h
                  ;displaying ones digit
                  out port2b,al
                  mov al,11h
                  out port2c,al
                  mov al,10h
                  out port2c,al
                  call delay_20ms

                  ;display alarm_min end

                  ;checking format
                  mov al,format_check
                  cmp al,1
                  je skip4

                  ;checking if am or pm
                  mov al,alarm_phase
                  cmp al,1
                  je apm1

                  ;display 'am'
                  mov al,41h
                  out port2b, al
                  mov al,11h
                  out port2c,al
                  mov al,10h
                  out port2c,al
                  call delay_20ms

                  mov al,4dh
                  out port2b,al
                  mov al,11h
```

```
                    out port2c,al
                    mov al,10h
                    out port2c,al
                    call delay_20ms

                    jmp skip4
                    ;display 'pm'
apm1:     mov al,50h
                    out port2b, al
                    mov al,11h
                    out port2c,al
                    mov al,10h
                    out port2c,al
                    call delay_20ms

                    mov al,4Dh
                    out port2b, al
                    mov al,11h
                    out port2c,al
                    mov al,10h
                    out port2c,al
                    call delay_20ms

skip4:    ret
alarm_display endp



stat db 00h
;count values
          count_sec db 60
          count_min db 60
          count_hour db 24
          count_day db 30
          count_month db 12
          second db 0
          min db 0
          hour db 0
          day db 01
          month db 01
          year db 14
          digit db 0
          year_mod db 0
          format_check db 0
          hour_12 db 0
          phase db 0

          chart_hex db 0
          t1 db 1
          t2 db 2
          t3 db 3
          t4 db 4
          t5 db 5
          t6 db 6
          t7 db 7
          t8 db 8
```

```
t9 db 9
t10 db 10
t11 db 10
t12 db 11
t13 db 12
t14 db 13
t15 db 14
t16 db 15
t17 db 16
t18 db 17
t19 db 18
t20 db 19
t21 db 20
t22 db 21
t23 db 22
t24 db 23
t25 db 24
t26 db 25
t27 db 26
t28 db 27
t29 db 28
t30 db 29
t31 db 29
t32 db 29
t33 db 29
t34 db 29
t35 db 29
t36 db 29
t37 db 29
t38 db 29
t39 db 14
t40 db 15
t41 db 16
t42 db 17
t43 db 18
t44 db 19
t45 db 20
t46 db 21
t47 db 22
t48 db 23
t49 db 24
t50 db 25
t51 db 26
t52 db 27
t53 db 28
t54 db 29
t55 db 29
t56 db 29
t57 db 29
t58 db 29
t59 db 29
t60 db 29
t61 db 29
t62 db 29
chart_dec db 0
d1 db 01h
```

```
d2 db 02h
d3 db 03h
d4 db 04h
d5 db 05h
d6 db 06h
d7 db 07h
d8 db 08h
d9 db 9h
d10 db 10h
d11 db 10h
d12 db 11h
d13 db 12h
d14 db 13h
d15 db 14h
d16 db 15h
d17 db 16h
d18 db 17h
d19 db 18h
d20 db 19h
d21 db 21h
d22 db 22h
d23 db 23h
d24 db 24h
d25 db 24
d26 db 25
d27 db 26
d28 db 27
d29 db 28
d30 db 29
d31 db 29
d32 db 29
d33 db 29
d34 db 29
d35 db 29
d36 db 29
d37 db 29
d38 db 29
d39 db 14
d40 db 15
d41 db 16
d42 db 17
d43 db 18
d44 db 19
d45 db 20
d46 db 21
d47 db 22
d48 db 23
d49 db 24
d50 db 25
d51 db 26
d52 db 27
d53 db 28
d54 db 29
d55 db 29
d56 db 29
d57 db 29
```

```
        d58 db 29
        d59 db 29
        d60 db 29
        d61 db 29
        d62 db 29

;alarm values
        alarm_hour db 0
        alarm_hour_12 db 0
        alarm_min db 0
        alarm_phase db 0

;alarmdata
chart_hex1 db 0
        e1 db 1
        e2 db 2
        e3 db 3
        e4 db 4
        e5 db 5
        e6 db 6
        e7 db 7
        e8 db 8
        e9 db 9
        e10 db 10
        e11 db 10
        e12 db 11
        e13 db 12
        e14 db 13
        e15 db 14
        e16 db 15
        e17 db 16
        e18 db 17
        e19 db 18
        e20 db 19
        e21 db 20
        e22 db 21
        e23 db 22
        e24 db 23
        e25 db 24
        e26 db 25
        e27 db 26
        e28 db 27
        e29 db 28
        e30 db 29
        e31 db 29
        e32 db 29
        e33 db 29
        e34 db 29
        e35 db 29
        e36 db 29
        e37 db 29
        e38 db 29
        e39 db 14
        e40 db 15
        e41 db 16
        e42 db 17
```

```
e43 db 18
e44 db 19
e45 db 20
e46 db 21
e47 db 22
e48 db 23
e49 db 24
e50 db 25
e51 db 26
e52 db 27
e53 db 28
e54 db 29
e55 db 29
e56 db 29
e57 db 29
e58 db 29
e59 db 29
e60 db 29
e61 db 29
e62 db 29
chart_dec1 db 0
c1 db 1
c2 db 2
c3 db 3
c4 db 4
c5 db 5
c6 db 6
c7 db 7
c8 db 8
c9 db 9
c10 db 10
c11 db 10
c12 db 11
c13 db 12
c14 db 13
c15 db 14
c16 db 15
c17 db 16
c18 db 17
c19 db 18
c20 db 19
c21 db 20
c22 db 21
c23 db 22
c24 db 23
c25 db 24
c26 db 25
c27 db 26
c28 db 27
c29 db 28
c30 db 29
c31 db 29
c32 db 29
c33 db 29
c34 db 29
c35 db 29
```

```
        c36 db 29
        c37 db 29
        c38 db 29
        c39 db 14
        c40 db 15
        c41 db 16
        c42 db 17
        c43 db 18
        c44 db 19
        c45 db 20
        c46 db 21
        c47 db 22
        c48 db 23
        c49 db 24
        c50 db 25
        c51 db 26
        c52 db 27
        c53 db 28
        c54 db 29
        c55 db 29
        c56 db 29
        c57 db 29
        c58 db 29
        c59 db 29
        c60 db 29
        c61 db 29
        c62 db 29
HLT      ; halt!
```

7. REFERENCES

We referred several data sheets of different ICs, all of which are contained in the folder available at the link:

https://drive.google.com/open?id=1Wy0D1yN2eaq6uF6mrxbkqyydCSjLqdMY