BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI, RAJASTHAN - 333031

25th April, 2019

CS F241
Microprocessor Programming and Interfacing

Fan Speed Sensing and Control

(Design assignment)

Batch No: 83

LAKSH SINGLA (**2017A7PS0082P**)

ADITYA UPADHYAY (**2017A7PS0083P**)

ANIRUDDHA JAYANT KARAJGI (**2017A7PS0084P**)

PRATIK RAJESH KAKADE (**2017A7PS0086P**)

(An assignment submitted by the above signed in partial fulfilment of the requirement of the course - CS F241 Microprocessor Programming and Interfacing)

Problem Statement Description

This system senses the speed at which the fan is rotating and adjusts the speed, based on the user input. The user can select three different speeds of the fan. The current speed should be sensed and the control mechanism should gradually increase the speed to the desired speed.

User Interface:

1. Fan starts when user presses 'Start' button.
2. User can then set the required speed by using a keypad interface. This speed value should be displayed on the display.
3. After setting speed initially, user should be able to change the fan speed setting by an up and down switch. Each press on this arrow button increases/ decreases the speed by 1 unit. Min speed value is 1, whereas maximum speed value is 5 Units. Pressing 'UP' button after reaching to value 5, should not change the display value or setting of fan speed. Same is true for lower bound.
4. Fan can be stopped by pressing 'Stop' button.
5. User can also set the mode of fan as 'Auto' mode besides a 'Regular mode' setting. In Auto mode, user should be able to enter the value of time in terms of hours after which the Fan has to be switched off automatically. (For example, if value entered is 2, then the Fan should switch off after 2 hours from the time this setting is applied)

Specifications and Assumptions

Following are the specifications and assumptions of the system:

- The 5 different speeds of the DC motor correspond to applying approximately 1V, 2V, 3V, 4V and 5V difference across its terminals (subject to resolution error of DAC).
- The user can enter hour between 1 to 9 (both inclusive) for auto mode. While simulating in Proteus, the time is scaled down to 10 seconds (i.e. auto mode on 5 will cause the fan to shutdown in 50 seconds rather than 5 hours). In real system, the counter value fed into 8253 can be changed to adjust accordingly. Alternatively, the clock frequency in to CLK0 of 8253 used can be changed.
- Green colored 7 segment display shows the speed of the fan, while the red one shows the status of the auto mode
- The user flow is like this:
    - User powers on the circuitry, and waits for initialization of the system, completed when both the displays show '-' on the display (setting up ports, clock etc)
    - User enters '1' - '5', corresponding to the initial speed of the fan.
    - User presses 'ON/C' button on the keypad to turn on the fan.
    - User can press '+' to increase the speed of the fan, and '-' to decrease the speed of the fan. Once started, the user cannot directly jump to any arbitrary speed of the fan by using the numpad.
    - To setup auto mode, user presses '=' key. The red display switches from '-' to 'A', displaying that auto mode is ready to be setup. User presses '1' - '9' to signify the number of hours (or the number of 10 seconds, in simulation) after which the fan must be turned off. (NOTE: During this whole process, the user can press 'X' key on the keypad to cancel the Auto mode setup process, however once setup, Auto mode cannot be cancelled). After finalizing the time requirement, the user is to press '=' again to set the auto mode into motion. The fan will shutdown after the desired time.
    - During the above whole process, the user can press 'ON/C' to manually shutdown the fan. (NOTE: User can shutdown the fan in between auto mode, without any side effects).
- Initial address executed by 8086 is 0x00000 (as opposed to 0xFFFF0 in real microprocessor).
- Only 8 bit addresses are being decoded for I/O space, as is done in most of the dedicated circuits. Therefore, we are assuming that in the future, the system won't grow much complex (that 16 bit addresses needed for decoding I/O devices).

List of Hardware Used

.

Component	Quantity	Purpose
8086 MICROPROCESSOR	1	CPU
74LS138 3:8 DECODER	3	Selects correct I/O port or memory chips
74LS245 OCTAL BUS TRANSCEIVER	2	Used as a data buffer
74LS373 OCTAL LATCH	3	Used to latch data
2732 EPROM	2	Read only memory which stores code segment
6116 CMOS STATIC RAM	2	R/W memory which contains data segment and stack segment
8253A PROGRAMMABLE INTERVAL TIMER	1	Works as a counter in auto mode
8255A PROGRAMMABLE PERIPHERAL INTERFACE	2	Used as an interface between I/O devices and CPU
DAC DIGITAL TO ANALOG CONVERTER	1	Converts digital signals to analog signals
FAN-DC	1	Fan dc motor
KEYPAD	1	Used as a user input device
74LS04 NOT	1 (1 NOT gate used)	Works as an inverter
74LS32 OR	4 (15 OR gates used)	Used to perform logical OR
SEVEN SEGMENT DISPLAY	2	Used as an output device to display the key pressed
8284 clock generator	1	Generates a clock signal for 8086
Single Pole Double Throw Switch	1	Reset switch of system

Memory Mapping

ROM-2732(4k / chip)

ROM1 and ROM2(Even + Odd): 0x20000-0x21FFF

RAM-6116(2k / chip)

RAM1 and RAM2(Even + Odd): 0x00000-0x00FFF

I/O Mapping

Port	Port Address	Input/Output	Device
Port A	00H	Output	7 Segment Display
Port B	02H	Output	7 Segment Display
Port C Lower	04H	Output	Keypad Columns
Port C Upper	04H	Input	Keypad Rows
Control Register	06H	-	-

Port	Port Address	Input/Output	Device
Clock 0	08H	Output	Counter input
Counter Control Register	0EH	Output	Counter Control Word

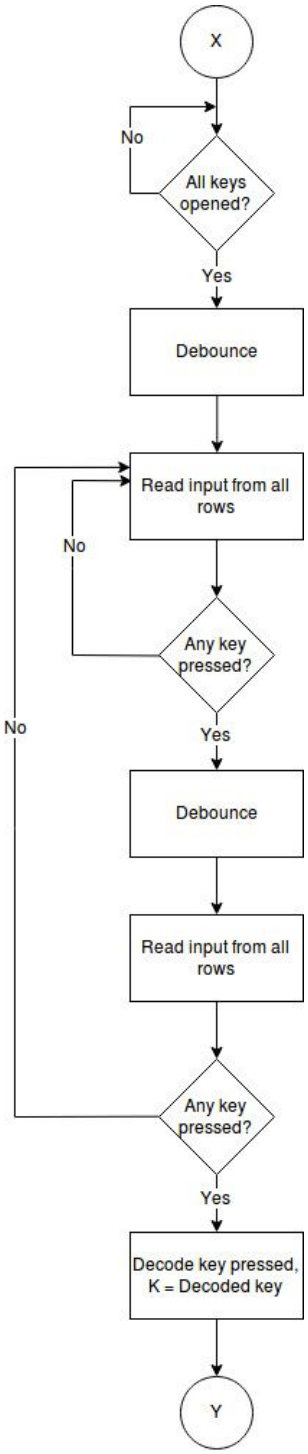
Port	Port Address	Input/Output	Device
Port A	10H	Output	DAC

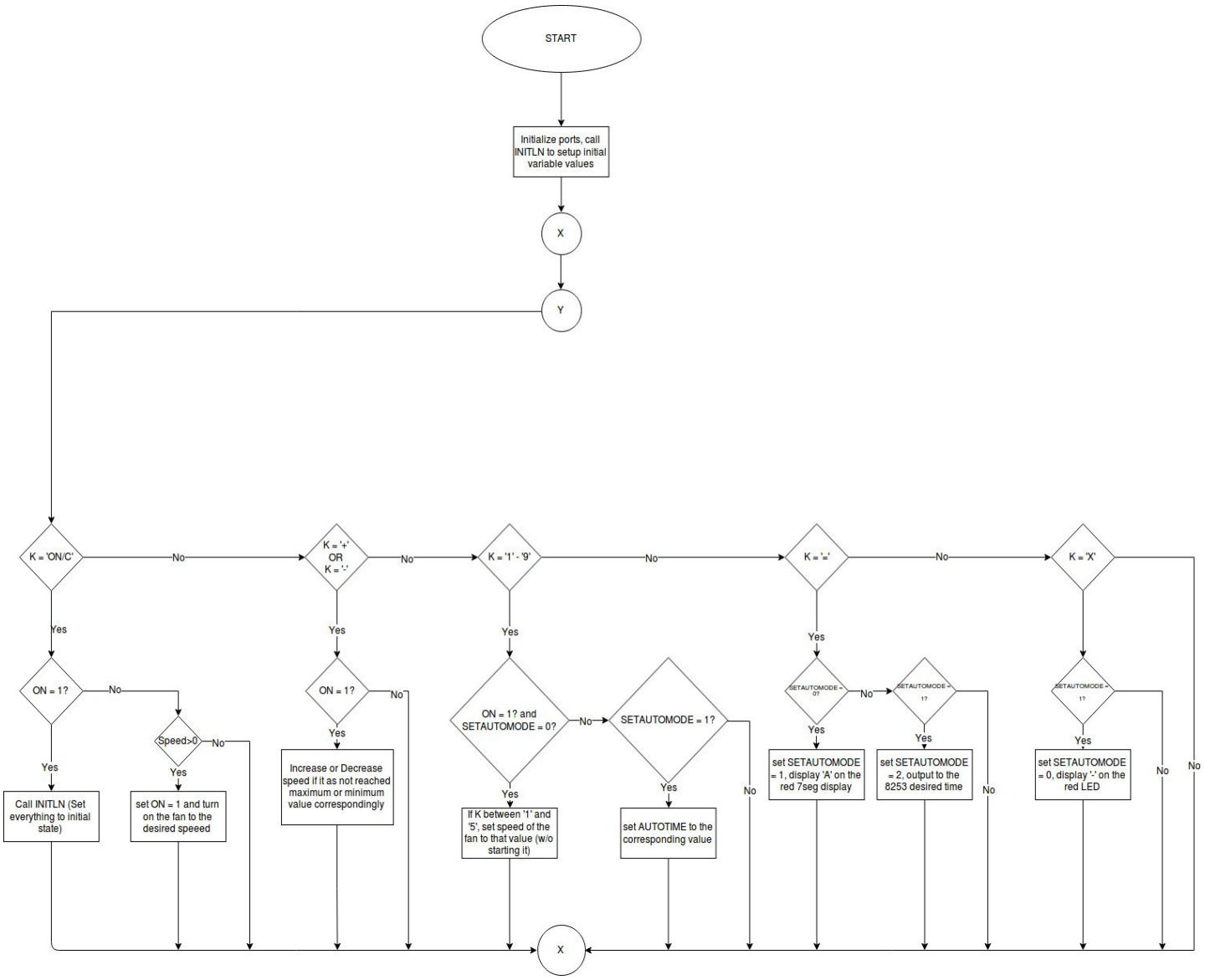
Port B	-	-	-
Port C Lower	-	-	-
Port C Upper	-	-	-
Control Register	16H	-	-

IVT Table

Interrupt Vector	Usage	How is the interrupt invoked
02H CS:IP = 0000:TESTINTR	Used by clock to interrupt the microprocessor after it has counted required seconds, to signal 8086 that the time specified by user in auto mode is over, and the fan is to be shutdown.	As the system consists of only one hardware interrupt (when clock reaches particular time), it is mapped to non maskable interrupt therefore it is invoked by setting logic 1 on NMI pin

Flow Chart


X
No
All keys opened?
Yes
Debounce
Read input from all rows
No
Any key pressed?
No
Yes
Debounce
Read input from all rows
Any key pressed?
Yes
Decode key pressed, K = Decoded key
Y



(Please zoom in to view the flowchart in detail)

Code

```
#make_bin#

; BIN is plain binary format similar to .com format, but not limited to 1 segment;
; All values between # are directives, these values are saved into a separate .binf
file.
; Before loading .bin file emulator reads .binf file with the same file name.

; All directives are optional, if you don't need them, delete them.

; set loading address, .bin file will be loaded to this address:
#LOAD_SEGMENT=0500h#
#LOAD_OFFSET=0000h#

; set entry point:
#CS=0500h#   ; same as loading segment
#IP=0000h#   ; same as loading offset

; set segment registers
#DS=0500h#   ; same as loading segment
#ES=0500h#   ; same as loading segment

; set stack
#SS=0500h#   ; same as loading segment
#SP=FFFEh#   ; set to top of loading segment

; set general registers (optional)
#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
#BP=0000h#

JMP  _CODE
             DB           6 DUP(0)            ;PADDING, TO SETUP INT 02H
             DW           TESTINTR
             DW           0000
TABLE_K      DB           0EEH,0EDH,0EBH,0E7H,
             DB           0DEH,0DDH,0DBH,0D7H,
             DB           0BEH,0BDH,0BBH,0B7H,
             DB           07EH,07DH,07BH,077H
```

```
;7SEG DISPLAY TABLE NUMERIC
TABLE_D      DB           3FH, 06H, 5BH, 4FH, 66H, 6DH
             DB           7DH, 27H, 7FH, 6FH, 77H, 7CH,
             DB           39H, 5EH, 79H, 71H
             ;NON NUMERIC PART
             DB           40H                 ; '-'
TABLE_D_R    EQU          16
;PORT ADDRESSES FOR FIRST 8255A
PORTA        EQU    00H
PORTB        EQU    02H
PORTC        EQU    04H
CREG         EQU    06H

;PORT ADDRESSES FOR SECOND 8255A
PORT2A       EQU          10H
CREG2        EQU          16H

;PORT ADDRESSES FOR 8253 TIMER
TIMER1       EQU          08H
TCREG        EQU          0EH

;OUTPUT BITS CORRESPONDING TO DIFFERENT FAN SPEEDS
FS1                 EQU          51*1
FS2                 EQU          51*2
FS3                 EQU          51*3
FS4                 EQU          51*4
FS5                 EQU          51*5

;DELAY VARIABLES CORRESPONDING TO 1-9
D1                  EQU          10
D2                  EQU          20
D3                  EQU          30
D4                  EQU          40
D5                  EQU          50
D6                  EQU          60
D7                  EQU          70
D8                  EQU          80
D9                  EQU          90


;STATE VARIABLES
ON           DB           0
AUTO         DB           0
SPEED        DB           0
SETAUTOMODE  DB           0
AUTOTIME     DB           0

;START BY INITIALIZING BOTH 8255A
```

```
_CODE:       MOV    AL, 10001000B
             OUT    CREG, AL
             MOV    AL, 10001011B
             OUT    CREG2, AL

;INITIALIZING INTERFACE
             CALL   INITLN

;SEND 00H TO KEYPAD COLUMNS
A0:          MOV    AL, 00H
             OUT    PORTC, AL

;CHECK FOR ALL KEY RELEASE
A1:          IN     AL, PORTC
             AND    AL, 0F0H
             CMP    AL, 0F0H
             JNZ    A1
             CALL   DELAY20

;CHECK IF ANY KEYPRESSED
             MOV    AL, 00H
             OUT    PORTC, AL
A2:          IN     AL, PORTC
             AND    AL, 0F0H
             CMP    AL, 0F0H
             JZ     A2
             CALL   DELAY20

;DEBOUNCE KEYPRESS
             MOV    AL, 00H
             OUT    PORTC, AL
             IN     AL, PORTC
             AND    AL, 0F0H
             CMP    AL, 0F0H
             JZ     A2

;KEY PRESS COLUMN 1
             MOV    AL, 0EH
             MOV    BL, AL
             OUT    PORTC, AL
             IN     AL, PORTC
             AND    AL, 0F0H
             CMP    AL, 0F0H
             JNZ    A3z

;PRESS COLUMN 2
             MOV    AL, 0DH
             MOV    BL, AL
```

```
            OUT    PORTC, AL
            IN     AL, PORTC
            AND    AL, 0F0H
            CMP    AL, 0F0H
            JNZ    A3

    ;KEY PRESS COLUMN 3
            MOV    AL, 0BH
            MOV    BL, AL
            OUT    PORTC, AL
            IN     AL, PORTC
            AND    AL, 0F0H
            CMP    AL, 0F0H
            JNZ    A3

    ;KEY PRESS COLUMN 4
            MOV    AL, 07H
            MOV    BL, AL
            OUT    PORTC, AL
            IN     AL, PORTC
            AND    AL, 0F0H
            CMP    AL, 0F0H
            JZ     A2

    ;DECODE KEY
A3:   OR      AL, BL
            MOV     CX, 10H
            MOV     DI, 00H
            ;MOV         BX, OFFSET TABLE_K
            ;MOV DI, DS:T_KBRD

A4:   CMP     AL, [TABLE_K + DI]
            JZ      A5
            INC     DI
            LOOP    A4


A5:   CMP DI, 0
            JZ X00
            CMP DI, 1
            JZ X01
            CMP DI, 2
            JZ X02
            CMP DI, 3
            JZ X03
            CMP DI, 4
            JZ X04
            CMP DI, 5
```

```
            JZ X05
            CMP DI, 6
            JZ X06
            CMP DI, 7
            JZ X07
            CMP DI, 8
            JZ X08
            CMP DI, 9
            JZ X09
            CMP DI, 10
            JZ X10
            CMP DI, 11
            JZ X11
            CMP DI, 12
            JZ X12
            CMP DI, 13
            JZ X13
            CMP DI, 14
            JZ X14
            CMP DI, 15
            JZ X15


X00:  CALL KEY00
            JMP REPL
X01:  CALL KEY01
            JMP REPL
X02:  CALL KEY02
            JMP REPL
X03:  CALL KEY03
            JMP REPL
X04:  CALL KEY04
            JMP REPL
X05:  CALL KEY05
            JMP REPL
X06:  CALL KEY06
            JMP REPL
X07:  CALL KEY07
            JMP REPL
X08:  CALL KEY08
            JMP REPL
X09:  CALL KEY09
            JMP REPL
X10:  CALL KEY10
            JMP REPL
X11:  CALL KEY11
            JMP REPL
X12:  CALL KEY12
```

```
              JMP REPL
X13:   CALL KEY13
              JMP REPL
X14:   CALL KEY14
              JMP REPL
X15:   CALL KEY15
              JMP REPL

       ;REPL LOOP AGAIN
REPL:  JMP A0

       ;CODE SHOULD NEVER REACH HERE
       ;IN CASE IT DOES DO NOT ALLOW IT TO PROCEED FURTHER
_STOP: JMP _STOP




;DELAY OF 20MS
DELAY20 PROC NEAR
    MOV     CX, 2220
X9: LOOP    X9
    RET
DELAY20 ENDP

;EVENT HANDLERS FOR DIFFERENT DIFFERENT KEYPRESSES
;KEYPAD ASSUMED TO BE LAID OUT LIKE
;      00     01     02     03
;      04     05     06     07
;      08     09     10     11
;      12     13     14     15

;'7'
KEY00 PROC NEAR
PUSH SI
CMP ON, 0
JZ KEY00_ALWAYS
CMP SETAUTOMODE, 1
JNZ KEY00_ALWAYS
MOV AUTOTIME, D7
MOV SI, 7
CALL DISPLAY2
KEY00_ALWAYS: POP SI
RET
KEY00 ENDP

;'8'
KEY01 PROC NEAR
```

```
PUSH SI
CMP ON, 0
JZ KEY01_ALWAYS
CMP SETAUTOMODE, 1
JNZ KEY01_ALWAYS
MOV AUTOTIME, D8
MOV SI, 8
CALL DISPLAY2
KEY01_ALWAYS: POP SI
RET
KEY01 ENDP

;'9'
KEY02 PROC NEAR
PUSH SI
CMP ON, 0
JZ KEY02_ALWAYS
CMP SETAUTOMODE, 1
JNZ KEY02_ALWAYS
MOV AUTOTIME, D9
MOV SI, 9
CALL DISPLAY2
KEY02_ALWAYS: POP SI
RET
KEY02 ENDP

KEY03 PROC NEAR
RET
KEY03 ENDP

;'4'
KEY04 PROC NEAR
PUSH SI
CMP ON, 0
JNZ KEY04_NOT_ON
MOV SPEED, 4
MOV SI, 4
CALL DISPLAY1
JMP KEY04_ALWAYS
KEY04_NOT_ON: CMP SETAUTOMODE, 1
JNZ KEY04_ALWAYS
MOV AUTOTIME, D4
MOV SI, 4
CALL DISPLAY2
KEY04_ALWAYS: POP SI
RET
KEY04 ENDP
```

```
;'5'
KEY05 PROC NEAR
PUSH SI
CMP ON, 0
JNZ KEY05_NOT_ON
MOV SPEED, 5
MOV SI, 5
CALL DISPLAY1
JMP KEY05_ALWAYS
KEY05_NOT_ON: CMP SETAUTOMODE, 1
JNZ KEY05_ALWAYS
MOV AUTOTIME, D5
MOV SI, 5
CALL DISPLAY2
KEY05_ALWAYS: POP SI
RET
KEY05 ENDP

;'6'
KEY06 PROC NEAR
PUSH SI
CMP ON, 0
JZ KEY06_ALWAYS
CMP SETAUTOMODE, 1
JNZ KEY06_ALWAYS
MOV AUTOTIME, D6
MOV SI, 6
CALL DISPLAY2
KEY06_ALWAYS: POP SI
RET
KEY06 ENDP

;'X'
KEY07 PROC NEAR
PUSH SI
CMP SETAUTOMODE, 2
JZ KEY07_ALWAYS
MOV SETAUTOMODE, 0
MOV SI, 16
CALL DISPLAY2
KEY07_ALWAYS: POP SI
RET
KEY07 ENDP

;'1'
KEY08 PROC NEAR
PUSH SI
CMP ON, 0
```

```
JNZ KEY08_NOT_ON
MOV SPEED, 1
MOV SI, 1
CALL DISPLAY1
JMP KEY08_ALWAYS
KEY08_NOT_ON: CMP SETAUTOMODE, 1
JNZ KEY08_ALWAYS
MOV AUTOTIME, D1
MOV SI, 1
CALL DISPLAY2
KEY08_ALWAYS: POP SI
RET
KEY08 ENDP

;'2'
KEY09 PROC NEAR
PUSH SI
CMP ON, 0
JNZ KEY09_NOT_ON
MOV SPEED, 2
MOV SI, 2
CALL DISPLAY1
JMP KEY09_ALWAYS
KEY09_NOT_ON: CMP SETAUTOMODE, 1
JNZ KEY09_ALWAYS
MOV AUTOTIME, D2
MOV SI, 2
CALL DISPLAY2
KEY09_ALWAYS: POP SI
RET
KEY09 ENDP

;'3'
KEY10 PROC NEAR
PUSH SI
CMP ON, 0
JNZ KEY10_NOT_ON
MOV SPEED, 3
MOV SI, 3
CALL DISPLAY1
JMP KEY10_ALWAYS
KEY10_NOT_ON: CMP SETAUTOMODE, 1
JNZ KEY04_ALWAYS
MOV AUTOTIME, D3
MOV SI, 3
CALL DISPLAY2
KEY10_ALWAYS: POP SI
RET
```

```
KEY10 ENDP

;'-'
KEY11 PROC NEAR
PUSH SI
PUSH DX
CMP SPEED, 1
JLE KEY11_ALWAYS
DEC SPEED
MOV DL, SPEED
MOV DH, 0
MOV SI, DX
CALL DISPLAY1
CALL SETSPEED
KEY11_ALWAYS: POP SI
POP DX
RET
KEY11 ENDP

;ON/OFF BUTTON
KEY12 PROC NEAR
CMP ON, 0
JZ KEY12_NOTON
CMP ON, 1
JZ KEY12_ON

KEY12_NOTON: CMP SPEED, 0
JZ KEY12_ALWAYS
MOV ON, 1
CALL SETSPEED
JMP KEY12_ALWAYS

KEY12_ON: MOV ON, 0
CALL INITLN
JMP KEY12_ALWAYS

KEY12_ALWAYS: RET
KEY12 ENDP

KEY13 PROC NEAR
RET
KEY13 ENDP

;'='
KEY14 PROC NEAR
PUSH SI
CMP ON, 0
JZ KEY14_ALWAYS
```

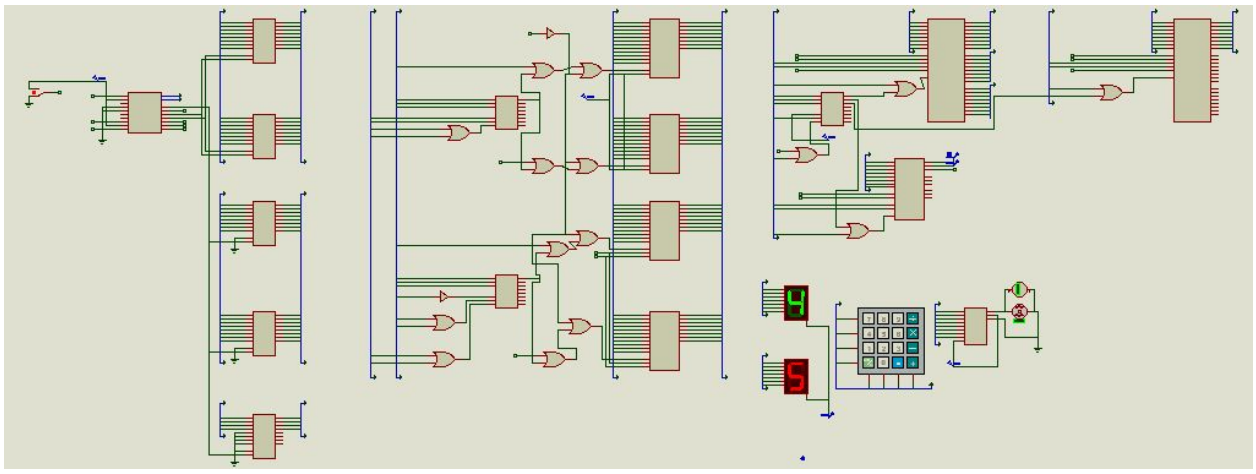
```
CMP SETAUTOMODE, 2
JZ KEY14_ALWAYS
CMP SETAUTOMODE, 0
JZ AUTO_AT_0
CMP SETAUTOMODE, 1
JZ AUTO_AT_1

AUTO_AT_0: MOV SETAUTOMODE, 1
MOV SI, 10
CALL DISPLAY2
JMP KEY14_ALWAYS

AUTO_AT_1: CMP AUTOTIME, 0
JLE KEY14_ALWAYS
MOV SETAUTOMODE, 2
CALL SETTIMER
JMP KEY14_ALWAYS

KEY14_ALWAYS: POP SI
RET
KEY14 ENDP

;'+'
KEY15 PROC NEAR
PUSH SI
PUSH DX
CMP SPEED, 5
JGE KEY15_ALWAYS
INC SPEED
MOV DL, SPEED
MOV DH, 0
MOV SI, DX
CALL DISPLAY1
CALL SETSPEED
KEY15_ALWAYS: POP SI
POP DX
RET
KEY15 ENDP

INITLN PROC NEAR
PUSH AX
PUSH SI

;INITIALISE ALL VARIABLES
MOV    ON, 0
MOV    AUTO, 0
MOV    SPEED, 0
MOV    SETAUTOMODE, 0
```

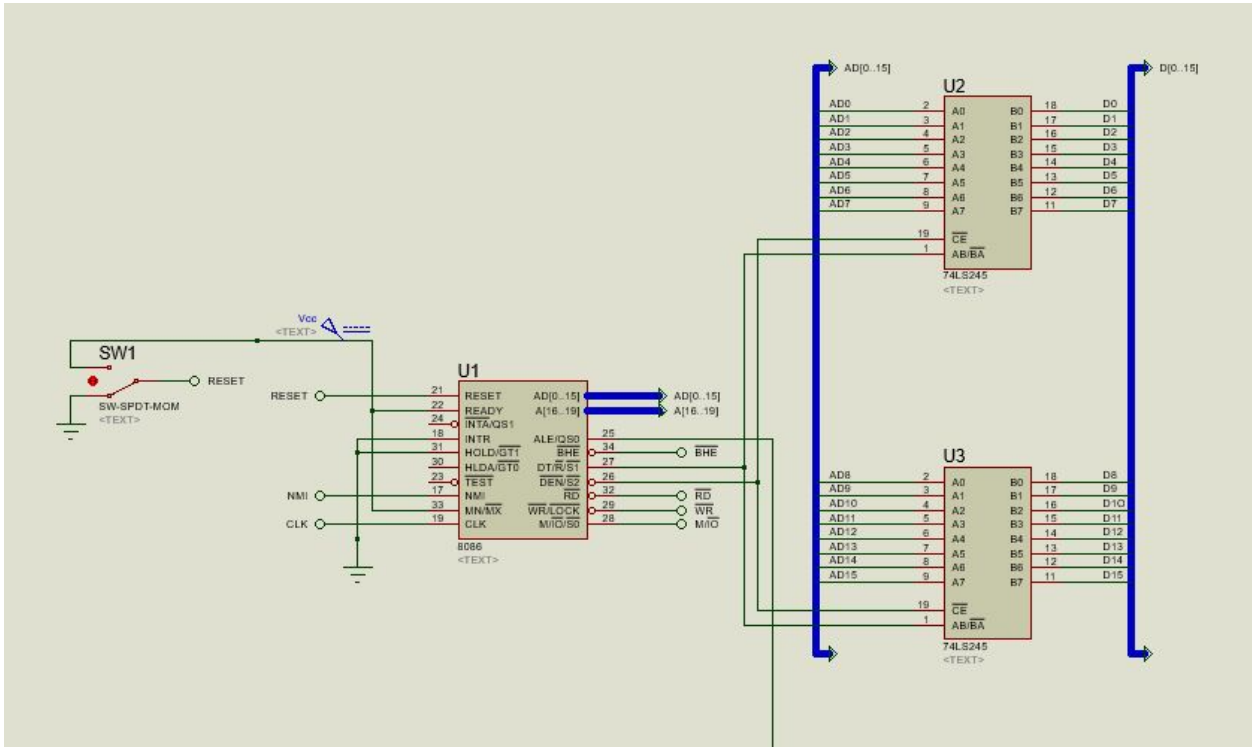
```
MOV    AUTOTIME, 0

;OUTPUT 0 ON THE FAN CONTROL
MOV AL, 0
OUT PORT2A, AL

;SETUP THE TCREG IN THE 8253 TIMER SO THAT OUT GOES LOW
MOV AL, 00010000B
OUT TCREG, AL

;DISPLAY '-' ON BOTH OF THE DISPLAYS
MOV SI, 16
CALL DISPLAY1
CALL DISPLAY2
INITLN_ALWAYS: POP SI
POP AX
RET
INITLN ENDP

;SET - IN FIRST SSD
DISPLAY1 PROC NEAR
PUSH BX
PUSH AX
MOV BX, OFFSET TABLE_D
CMP SI, 0
JL     DISPLAY1_ALWAYS
CMP SI, TABLE_D_R
JG     DISPLAY1_ALWAYS
MOV AL, [BX+SI]
OUT PORTA, AL
DISPLAY1_ALWAYS: POP BX
POP AX
RET
DISPLAY1 ENDP

;SET - IN SECOND SSD
DISPLAY2 PROC NEAR
PUSH BX
PUSH AX
MOV BX, OFFSET TABLE_D
CMP SI, 0
JL     DISPLAY2_ALWAYS
CMP SI, TABLE_D_R
JG     DISPLAY2_ALWAYS
MOV AL, [BX+SI]
OUT PORTB, AL
DISPLAY2_ALWAYS: POP BX
POP AX
```

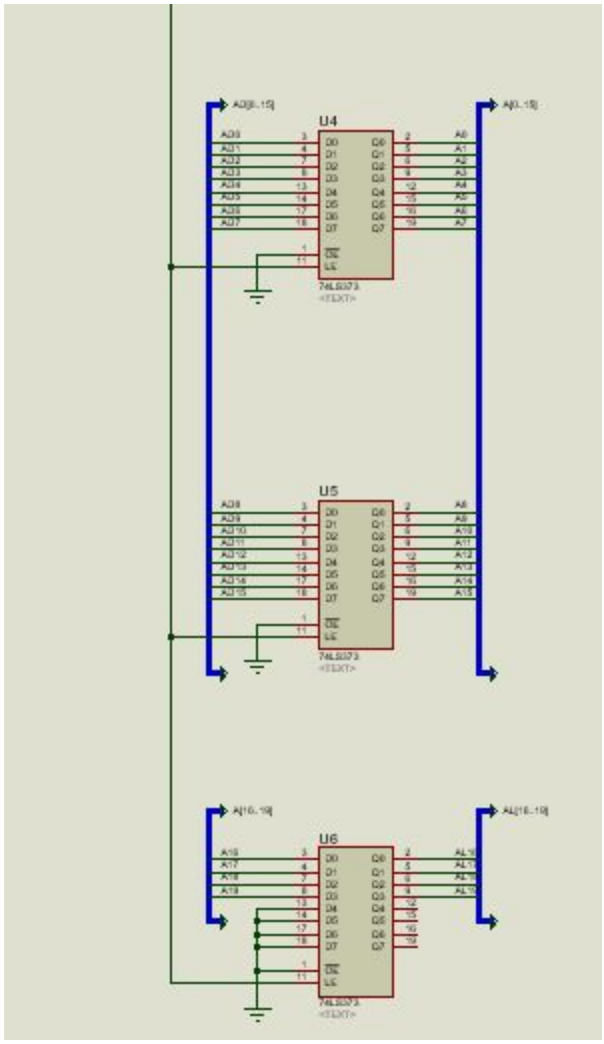
```
RET
DISPLAY2 ENDP

SETTIMER PROC NEAR
PUSH AX
;MOV AL, 00010000B
;OUT TCREG, AL
MOV AL, AUTOTIME
OUT TIMER1, AL
POP AX
RET
SETTIMER ENDP

;SETS THE SPEED OF THE FAN,
;ACCORDING TO THE VALUE IN THE SPEED VARIABLE
SETSPEED PROC NEAR
PUSH AX
CMP SPEED, 1
JZ SETSPEED_1
CMP SPEED, 2
JZ SETSPEED_2
CMP SPEED, 3
JZ SETSPEED_3
CMP SPEED, 4
JZ SETSPEED_4
CMP SPEED, 5
JZ SETSPEED_5
JMP SETSPEED_ALWAYS

SETSPEED_1: MOV AL, FS1
OUT PORT2A, AL
JMP SETSPEED_ALWAYS
SETSPEED_2: MOV AL, FS2
OUT PORT2A, AL
JMP SETSPEED_ALWAYS
SETSPEED_3: MOV AL, FS3
OUT PORT2A, AL
JMP SETSPEED_ALWAYS
SETSPEED_4: MOV AL, FS4
OUT PORT2A, AL
JMP SETSPEED_ALWAYS
SETSPEED_5: MOV AL, FS5
OUT PORT2A, AL
JMP SETSPEED_ALWAYS

SETSPEED_ALWAYS: POP AX
RET
SETSPEED ENDP
```

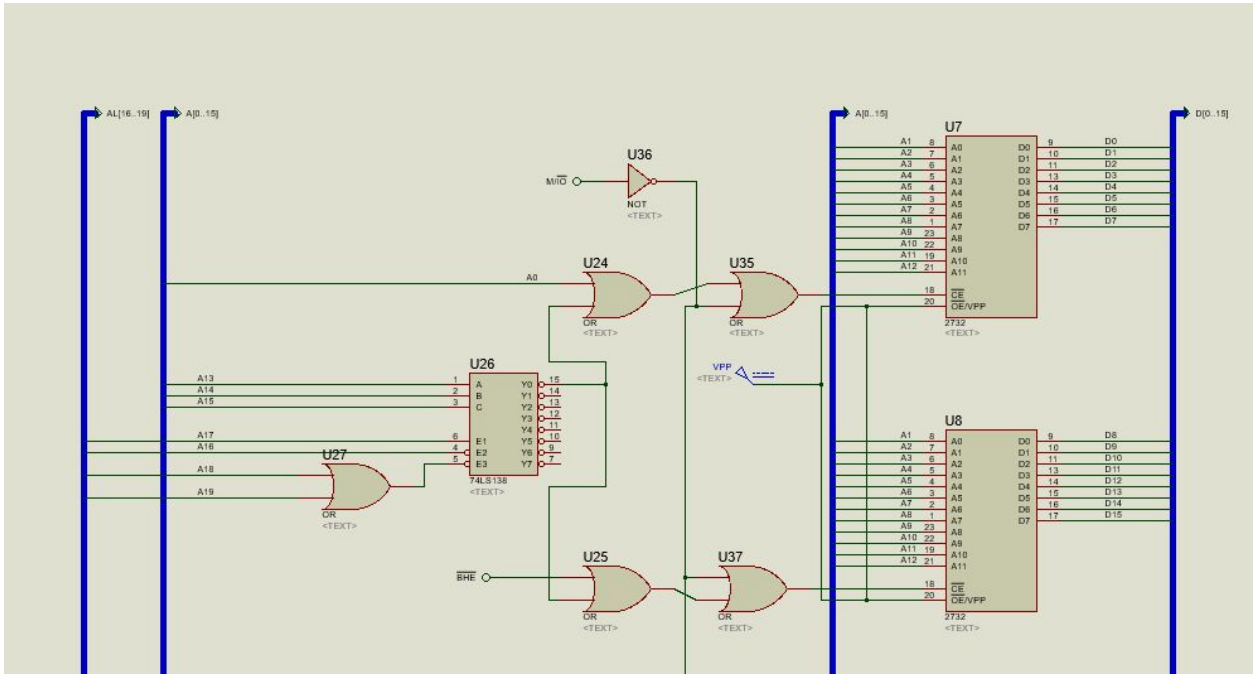
```
TESTINTR PROC NEAR
MOV ON, 0
CALL INITLN
TESTINTR_ALWAYS: IRET
TESTINTR ENDP

HLT           ; halt!
```

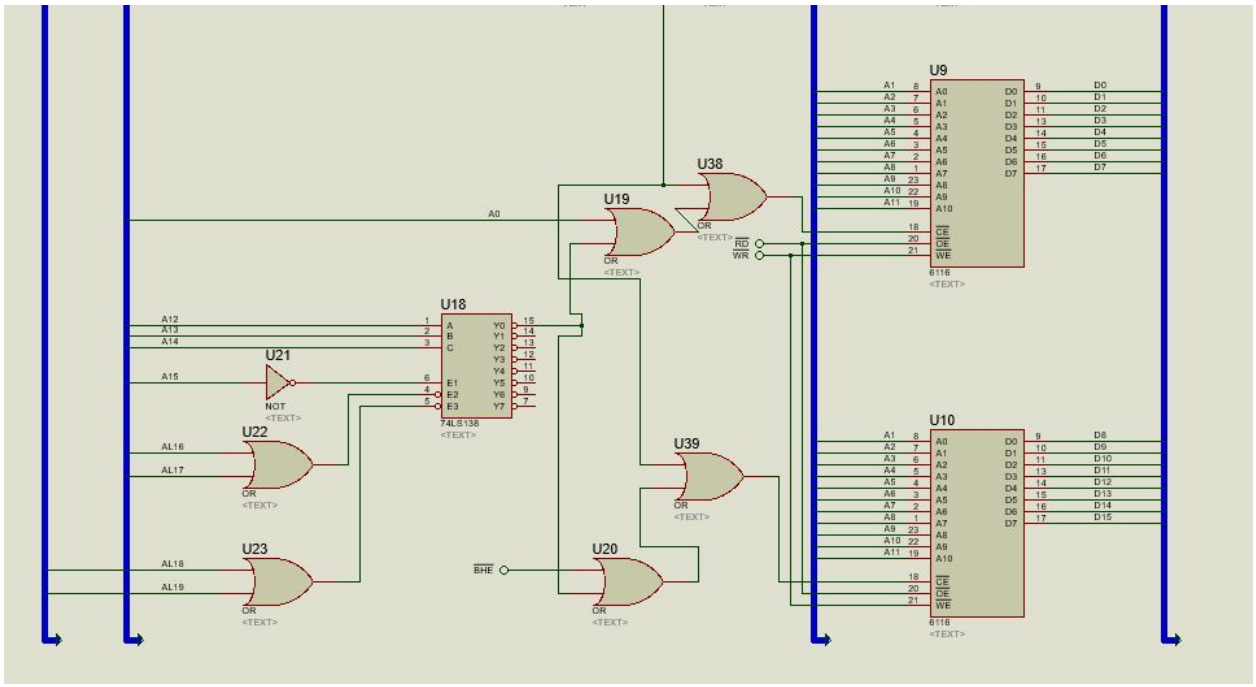
Circuit Diagram

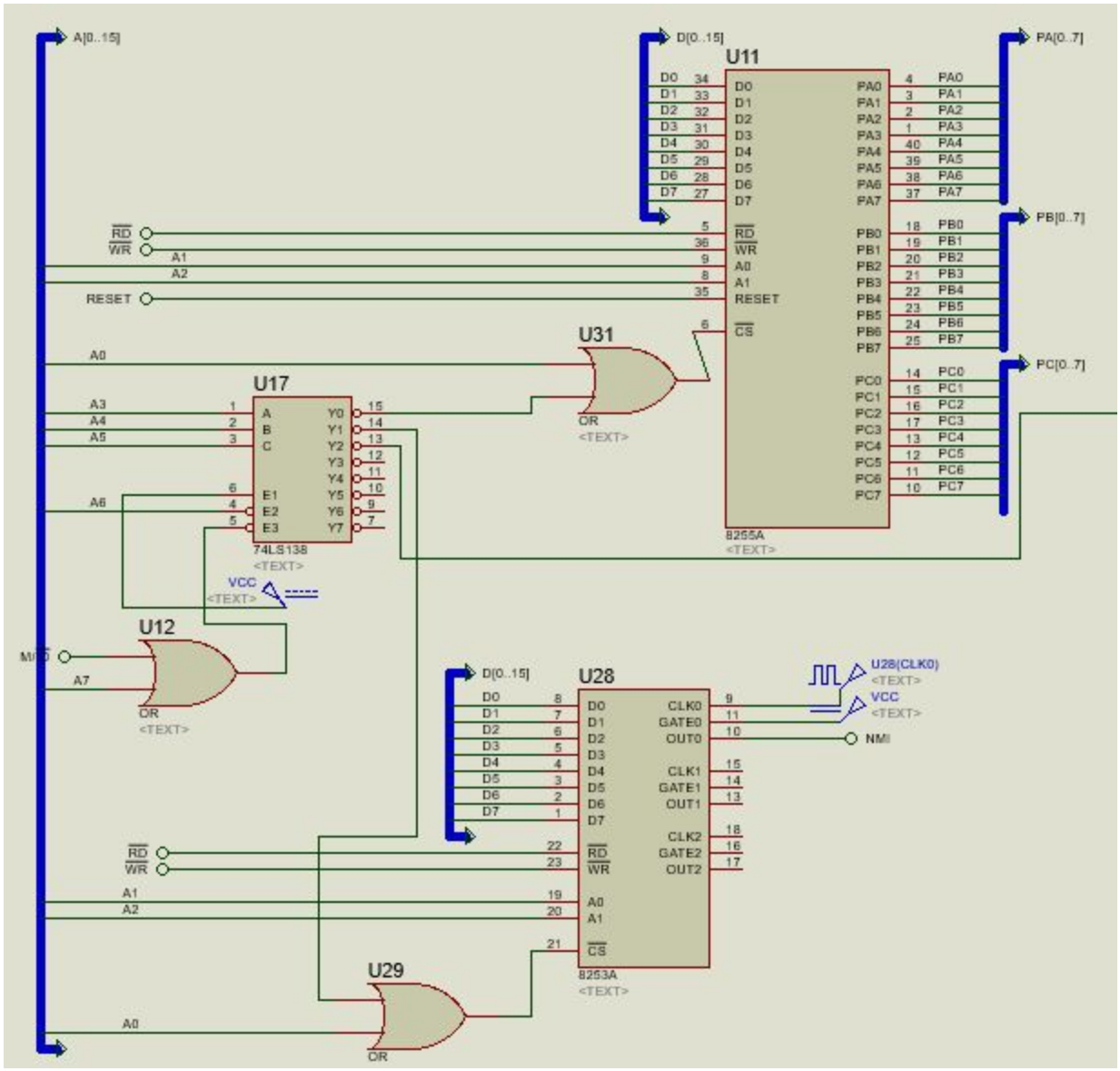


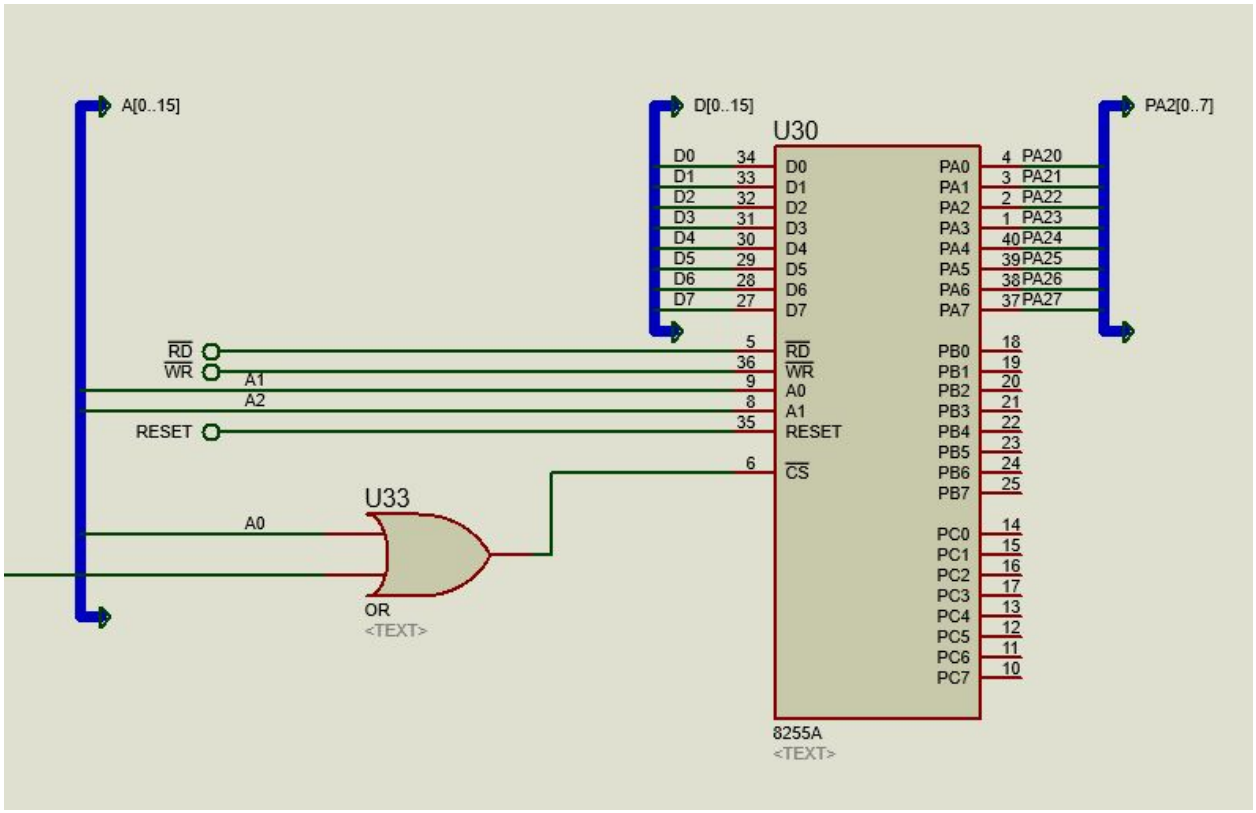


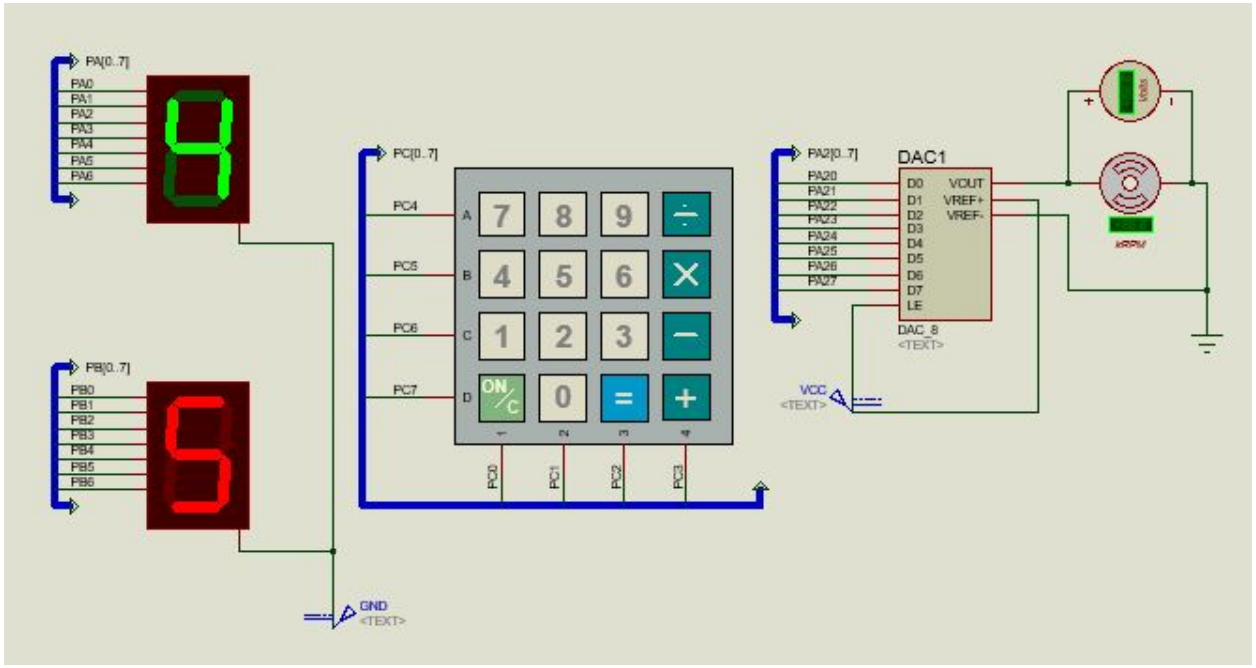

U4
U5
U6
74LS373


AL[16..19]
A[0..15]
A[0..15]
D[0..15]
U36
M/IO
NOT
<TEXT>
U24
U35
A0
OR
<TEXT>
U7
A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A12
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11
D0 D1 D2 D3 D4 D5 D6 D7
CE
OE/VPP
2732
U26
A13 A14 A15
A17 A16
A18
A19
A B C
E1 E2 E3
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
74LS138
U27
VPP
U8
D8 D9 D10 D11 D12 D13 D14 D15
BHE
U25
U37


U9
U38
U19
A0
RD
WR
U18
A12 A13 A14
A15
U21
NOT
U22
AL16
AL17
U23
AL18
AL19
U39
U20
BHE
U10
6116
CE
WE


A[0..15]
D[0..15]
U11
PA[0..7]
PB[0..7]
PC[0..7]
RD
WR
A1
A2
RESET
A0
U31
OR
<TEXT>
U17
A3
A4
A5
A6
74LS138
<TEXT>
VCC
<TEXT>
8255A
<TEXT>
U12
M/IO
A7
OR
<TEXT>
D[0..15]
U28
U28(CLK0)
<TEXT>
VCC
<TEXT>
NMI
RD
WR
A1
A2
8253A
<TEXT>
U29
A0
OR


A[0..15]
D[0..15]
U30
PA2[0..7]
D0 34
D1 33
D2 32
D3 31
D4 30
D5 29
D6 28
D7 27
D0
D1
D2
D3
D4
D5
D6
D7
PA0 4 PA20
PA1 3 PA21
PA2 2 PA22
PA3 1 PA23
PA4 40 PA24
PA5 39 PA25
PA6 38 PA26
PA7 37 PA27
RD 5
WR 36
A0 9
A1 8
RESET 35
CS 6
PB0 18
PB1 19
PB2 20
PB3 21
PB4 22
PB5 23
PB6 24
PB7 25
PC0 14
PC1 15
PC2 16
PC3 17
PC4 13
PC5 12
PC6 11
PC7 10
RD
WR
A1
A2
RESET
A0
U33
OR
<TEXT>
8255A
<TEXT>


PA[0..7]
PB[0..7]
PC[0..7]
PA2[0..7]
DAC1
DAC_8
<TEXT>
VCC
<TEXT>
GND
<TEXT>
VOUT
VREF+
VREF-
LE
PC4
PC5
PC6
PC7
PC0
PC1
PC2
PC3
ON/C

‘

Specific Datasheets

- DAC0808

http://www.ti.com/lit/ds/symlink/dac0808.pdf

(NOTE: In proteus, DAC_08 was used, which is the one provided in the components to keep the design simple. In real design, DAC0808 would required to be used along with op amp LF351 and resistors and transistors, as shown in the Figure1 of DAC0808's datasheet)

- 8284 (Clock generator)

http://home.etf.rs/~vm/os/mips/razno/datasheets/8284.pdf

(NOTE: Clock generator is not shown in Proteus because of unavailability of model. It's connections as per the specifications are displayed on the chart paper)