MICROPROCESSORS AND INTERFACING

Design Project

25/04/2019

Question – 24
Group 40

Neelabh Sinha 2016B5A80600P
Parth Kashikar 2016B5A80656P
Ayush Yembarwar 2016B5A80657P
Aakash Bist 2016B5A80675P

Prepared as a part of
INSTR F241, Microprocessors and Interfacing

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
April, 2019

The Problem Statement (Weather Monitoring Station)

System Description: This system monitors weather parameters such as: Air Temperature, Air-Humidity, barometric Pressure and wind-speed. It then displays the average over regular intervals of an hour on a seven-segment display. The Display is continuous. Update of the display is done once in an hour. Weather parameters are sensed at regular intervals of 2 minutes.

The display is of the format: "Temperature – Value 0C" and so on.

- Other than the regular display, the user can request the display of the weather parameters to be updated at any point of time by pressing a push button key. The accuracy of the parameters monitored has to be up to two decimal points.

SPECIFICATIONS

The analog input for the system is received from the sensors which are connected to an 8 bit parallel ADC (0808). These sensor modules generate analog voltages ~0-5V which is connected to the ADC, which in turn, generates an 8 bit value between 0 and 255.

There is an 8259 Programmable Interrupt controller device that accepts four interrupts from various sources, namely the timers, an external button and an EOC interrupt from the ADC. The IVT for the 8259 is stored in the ROM at a vector address of 80h onwards (corresponding to a memory address 80h*4=00200h). There are two timer IC's (8253) generating interrupts every 2 minutes and every one hour.

Every two minutes, an interrupt is generated and an ISR is invoked in which the ADC value is read and this digital data is stored in the RAM. It is as though an array of 30 elements is maintained for each sensor, where after the 30th reading of data, the next value is stored in the first position. Therefore, the past 30 readings are always maintained.

Every one hour, there is an interrupt generated that invokes an ISR that averages the values for the past hour. For the first hour, averaging is done for only the number of values available. After averaging, the values are scaled according to the specifications of the sensors. This scaled and average value is displayed.

There is also an external button which on pressing, generates an interrupt which takes a reading and averages the past 30 readings (including the current reading i.e. the past hour). This displays value on the LCD as per the request of the external button.

MEMORY INTERFACING

The memory interfaced uses 6116 RAM chips and 2732 ROM chips to interface a total of 8k of ROM and 4k of RAM. Addressing starts at 00000h so that the complete memory addressing is as:

ROM1 – 00000h- 01FFFh

RAM1 – 02000h- 02FFFh

Both, even and odd banks have been incorporated in the design. The decoding logic is obtained from:

ROM1:

19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

RAM1:

19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1

Hence, to decode memory, we use bits A15, A14 and A13 of the address lines.

I/O INTERFACING

The following I/O devices need to be interfaced to address lines:

- 8259
- Two 8255s (labeled 8255a and 8255b)
- Two 8253s (labeled 8253a and 8253b)

The addressing is:

- 8259 (Interrupt controller) - 04000h
- 8255a (for LCD operations) – 04010h
- 8255b(for ADC operations) – 04020h
- 8253a (2 min timer) – 04030h
- 8253b (1hr timer) -- 04040h

ASSUMPTIONS

Some assumptions are being made in consideration for the design:

- The display on the LCD displays an average of the previous 30 values read, i.e., the previous hour.
- Each time the user presses the external button, the clocks are not reset, implying that the next reading continues to take place as per the original 2 minute scheme which is set. On the button press, a new value is taken, added to data stored in memory and then, the past 30 values are taken
  for averaging, scaling and displaying on the LCD monitor.
- The button press does not clash with the 2 minute interrupt in normal usage. This is a fair assumption to make , as the probability for the same is very small in real-time usage of the weather
  monitoring station.
- In case of clash during operation (highly unlikely), and non-servicing of button interrupt, a second press will ensure the servicing of the interrupts, without affecting the 2minute interrupt-servicing.
- For the simulations and debugging, we have connected a faster (than 2 min) output of clock to see the output changes. In actual usage, 2 minute interrupt is used.

INTEGRATED CIRCUITS AND DEVICES USED

1) Random Access Memory UT-6116
2) Read Only Memory UT-2732
3) Two 8 bit latches 74LS373
4) 8 bit bidirectional buffer 74LS245
5) Microprocessor 8086
6) Programmable Interrupt Controller 8259
7) Programmable Peripheral Interface 8255
8) 8x1 Decoders – 74LS138
9) Programmable Interval Timer 8253
10) Analog to Digital Converter AD0808
11) External Push Button
12) Resistors – 10k ohm
13) Quad 2-input OR gate chip 7432
14) Temperature, Humidity, Pressure and Wind speed Sensors (using appropriate voltage generators)

SENSORS USED

Temperature:

AD8494

Sensing Temperature	**5°C ~ 50°C**
Output Type	**Analog**
Accuracy	**± 1°C**

Humidity:

HF 3223 / HTF 3223 Temperature and Humidity Module

Humidity Range	0% ~ 99% RH
Operating Temperature	-30°C ~ 80°C
Sensitivity	-
Accuracy	± 5% RH
Response Time	10s
Output	Linear Voltage

Pressure:

KP125 Absolute Pressure Sensor

Humidity Range	0% ~ 99% RH
Operating Pressure	5.80 ~ 16.68 PSI, 40 ~ 115 kPa
Port Size	-
Accuracy	± 1.5%
Voltage – Supply	4.5 V ~ 5.5 V
Output	0.5 V ~ 4.5 V

Wind Speed:

WE550 Wind Speed Sensor

Threshold	<=3 mph (1.35 m/s)
Output	4-20 mA
Range	>= 4 mph to 110 mph (>= 1.8 to 50 m/s)
Accuracy	*:* 0.2 mph over the range 11 to 110 mph (0.09 m/s from 4.9 to 24.6 m/s)
Operating Voltage	10-26 VDC
Warm Up Time	3 seconds minimum
Operating Temp	-40 to +131°F (-40 to +55°C

CALCULATIONS FOR SCALING

The ADC used in the design produces a voltage between 0 and 255d for the sensors. To scale it to the values for Pressure, Temperature and Humidity, we use a scaling function that employs the following formulae:

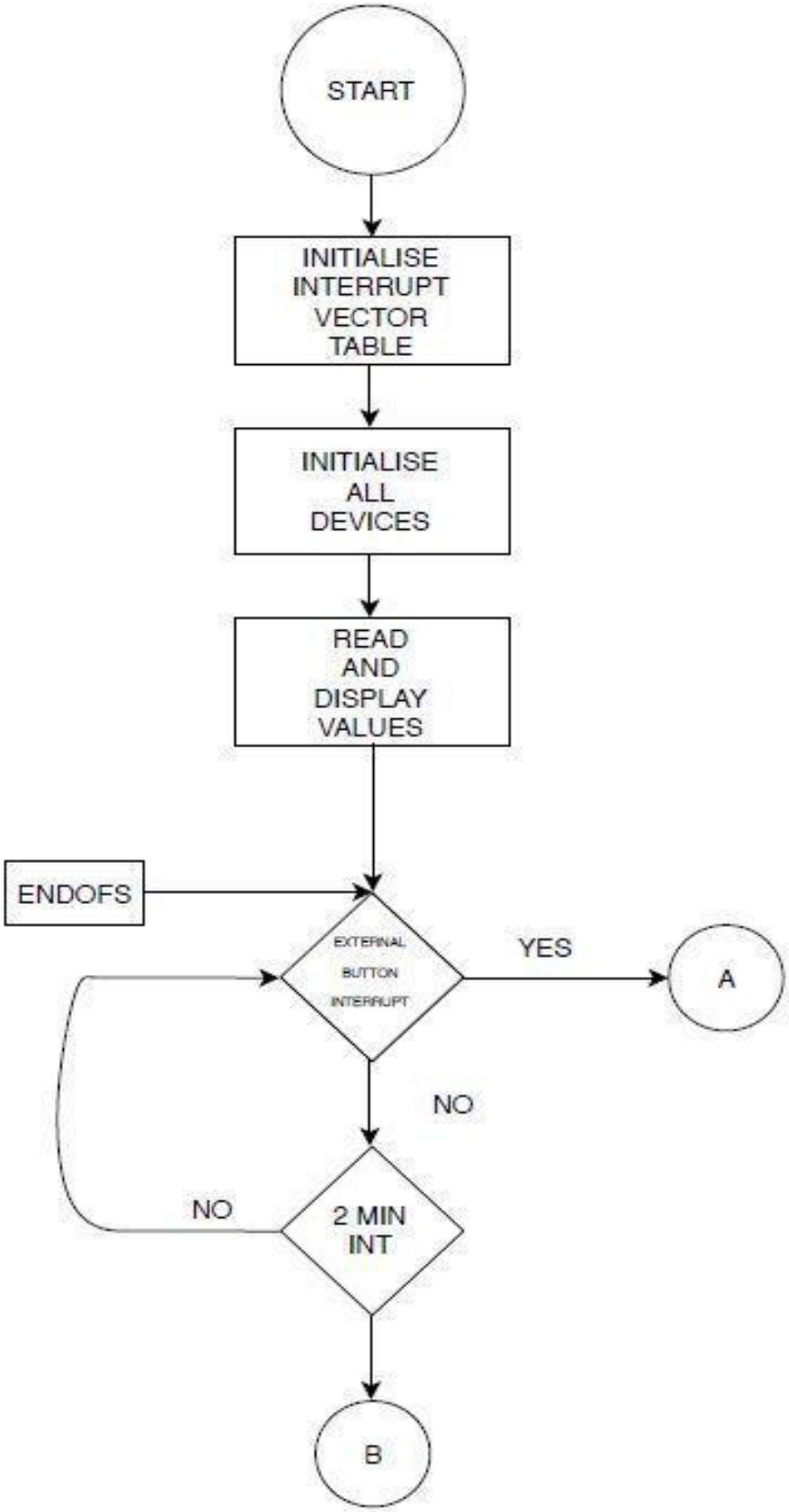
Pressure: (0-2bar)
Hex value is obtained by: ADC value *02h/FFh

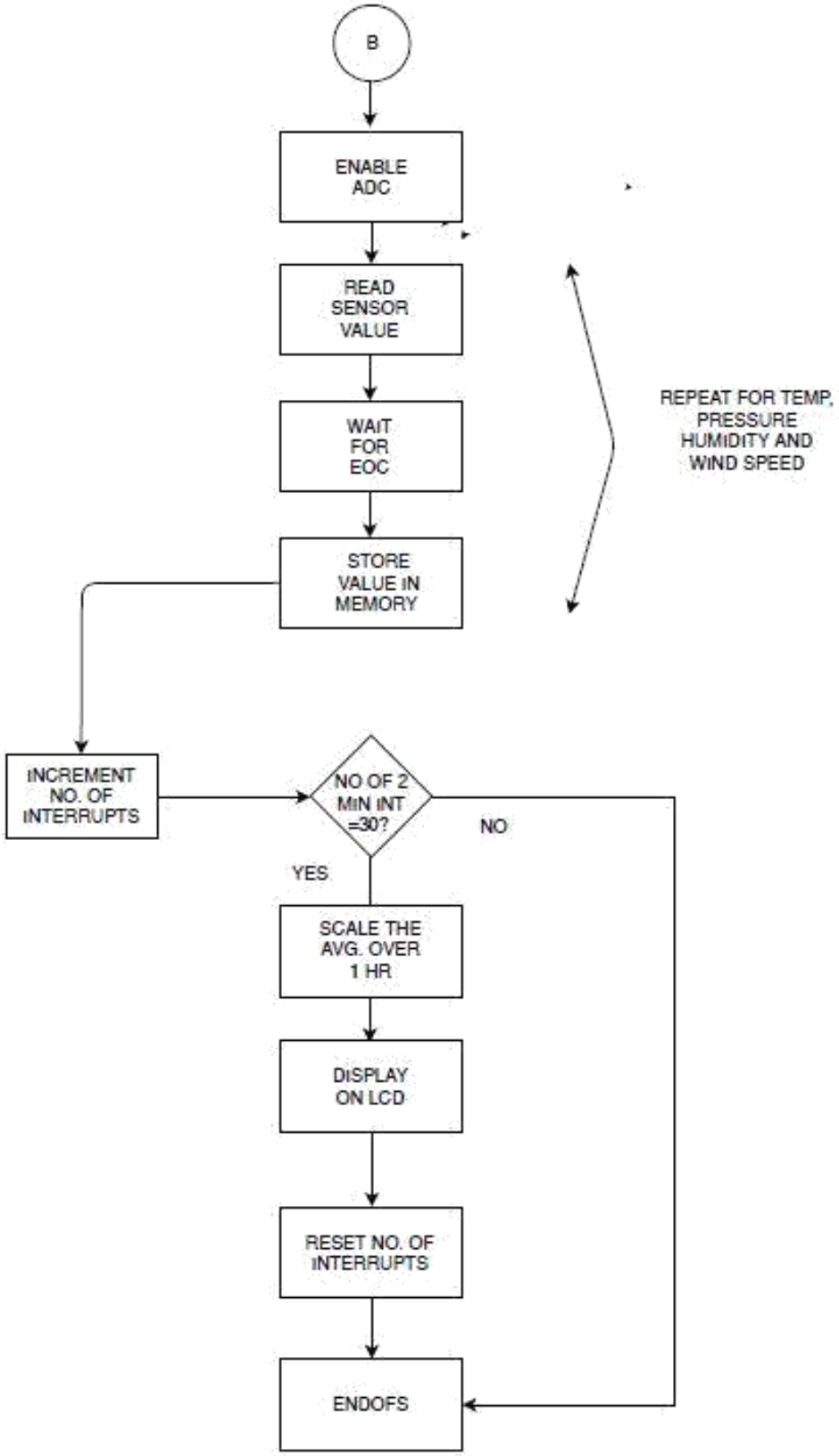
Temperature: (5-50°C): ADC value *32h/FFh

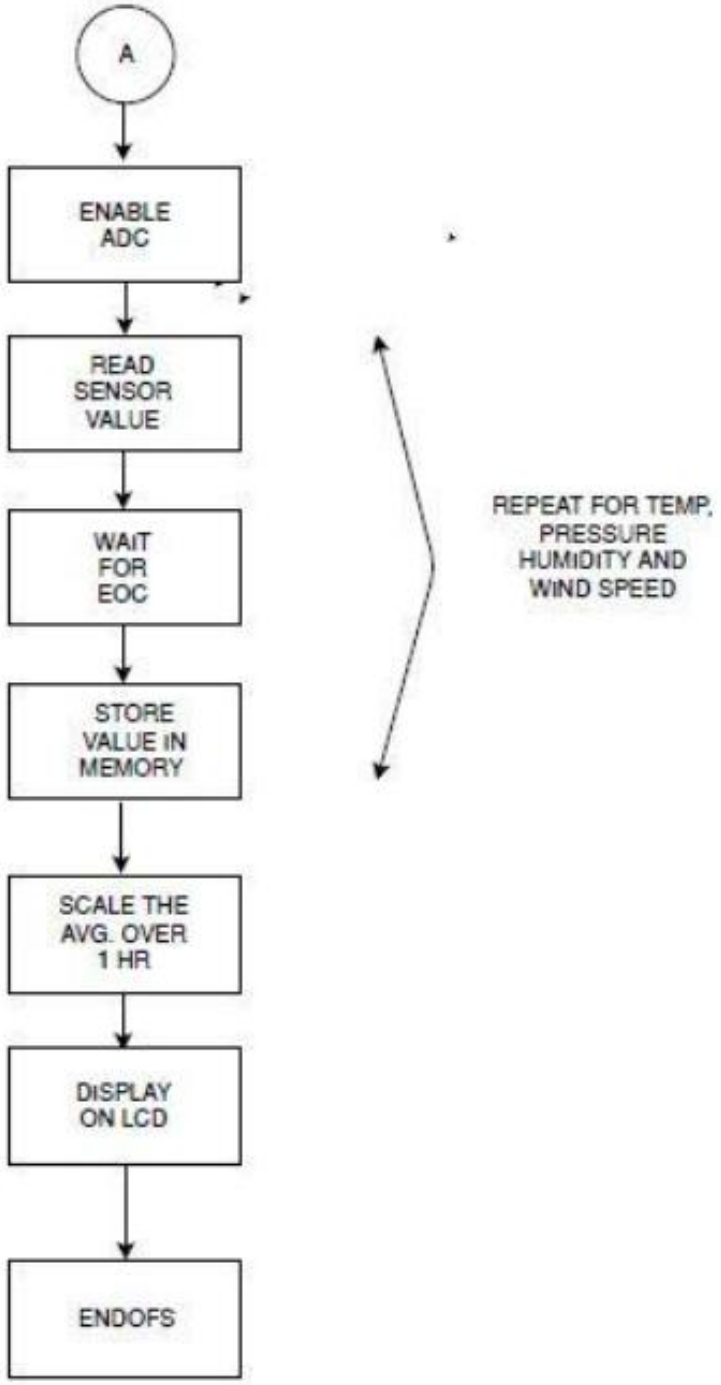
Humidity: (0-99%): ADC value *63h/FFh

Windspeed: (0-30mph): ADC value *1D/FFh.

FLOWCHART


START
INITIALISE INTERRUPT VECTOR TABLE
INITIALISE ALL DEVICES
READ AND DISPLAY VALUES
ENDOFS
EXTERNAL BUTTON INTERRUPT
YES
A
NO
NO
2 MIN INT
B

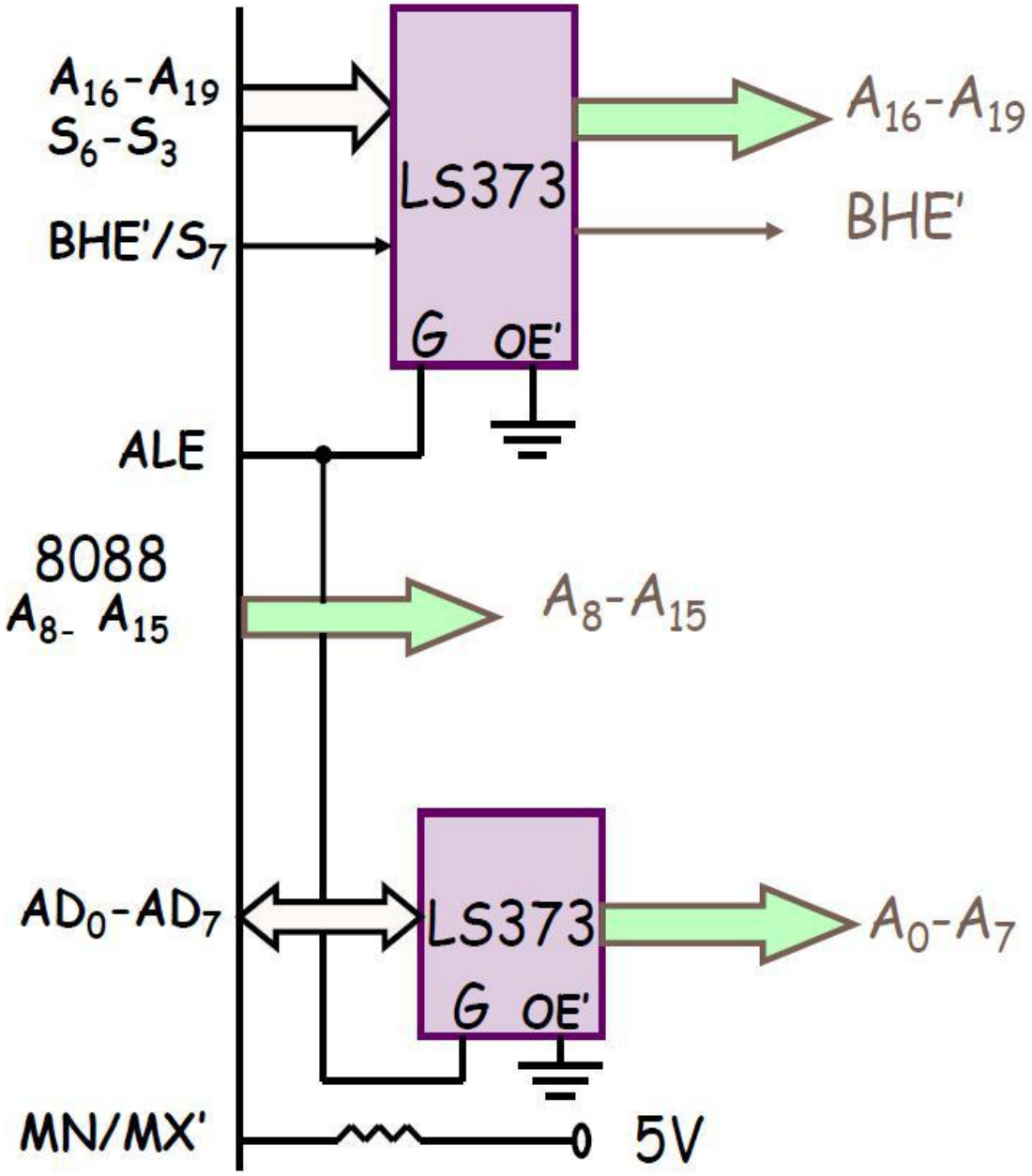

B
ENABLE ADC
READ SENSOR VALUE
WAIT FOR EOC
STORE VALUE IN MEMORY
REPEAT FOR TEMP, PRESSURE HUMIDITY AND WIND SPEED
INCREMENT NO. OF INTERRUPTS
NO OF 2 MIN INT =30?
NO
YES
SCALE THE AVG. OVER 1 HR
DISPLAY ON LCD
RESET NO. OF INTERRUPTS
ENDOFS


A
ENABLE ADC
READ SENSOR VALUE
WAIT FOR EOC
STORE VALUE IN MEMORY
SCALE THE AVG. OVER 1 HR
DISPLAY ON LCD
ENDOFS
REPEAT FOR TEMP, PRESSURE HUMIDITY AND WIND SPEED

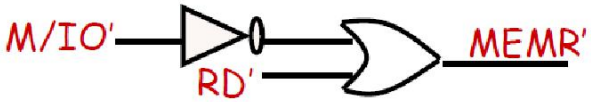
CIRCUIT DIAGRAM:

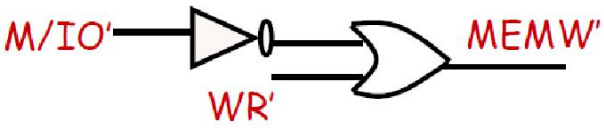
System bus (Address):`

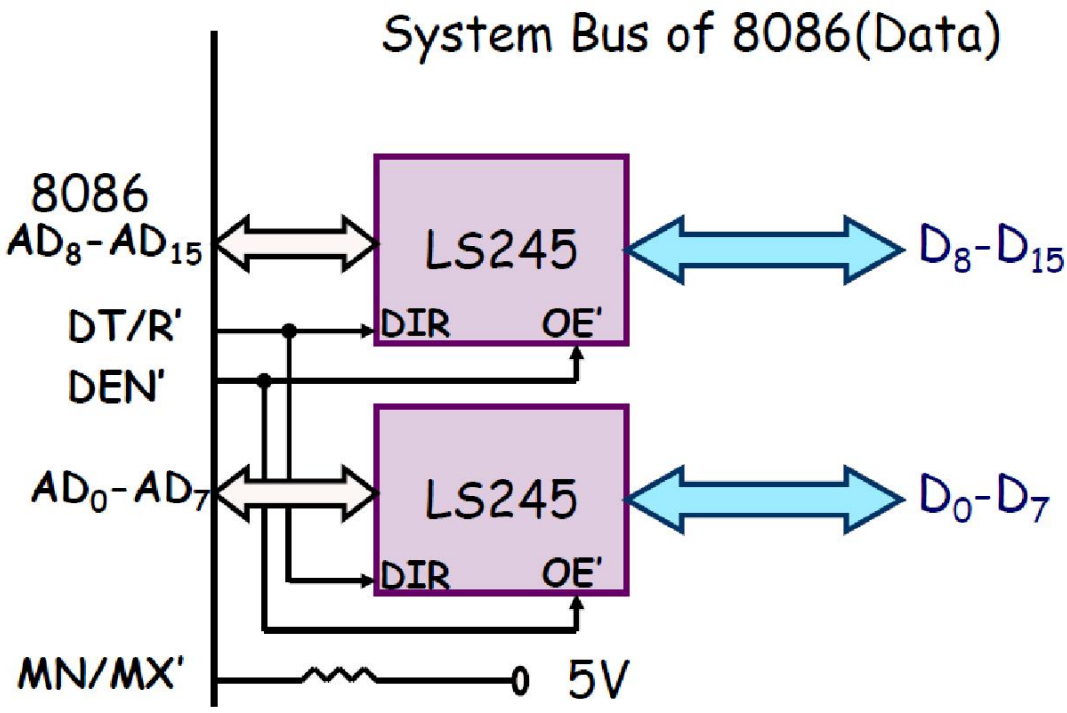
8086

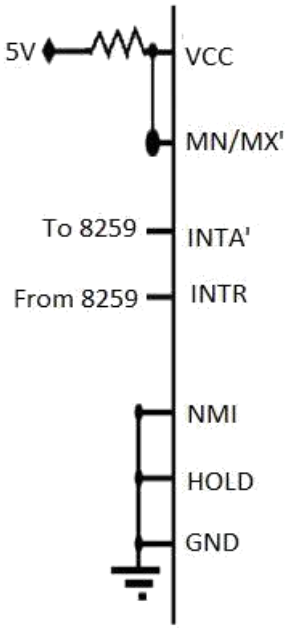


System Bus (Control and Data):

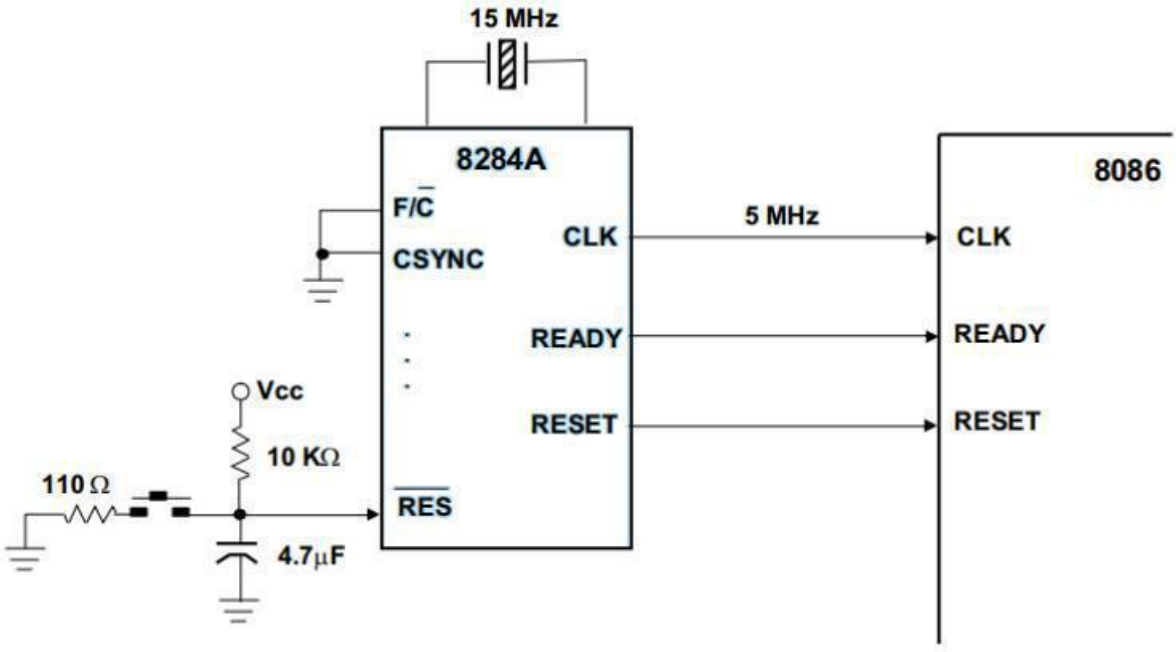






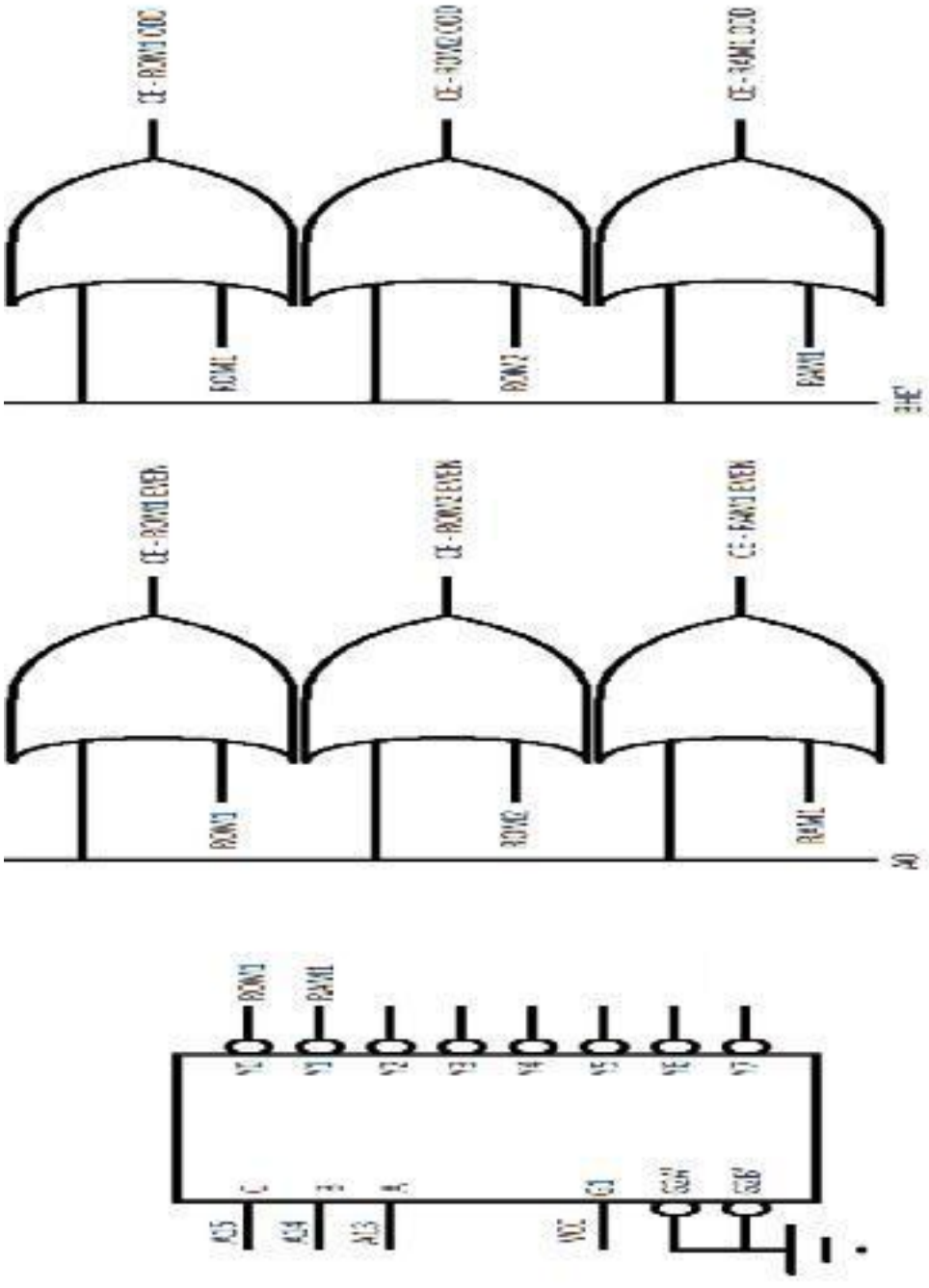


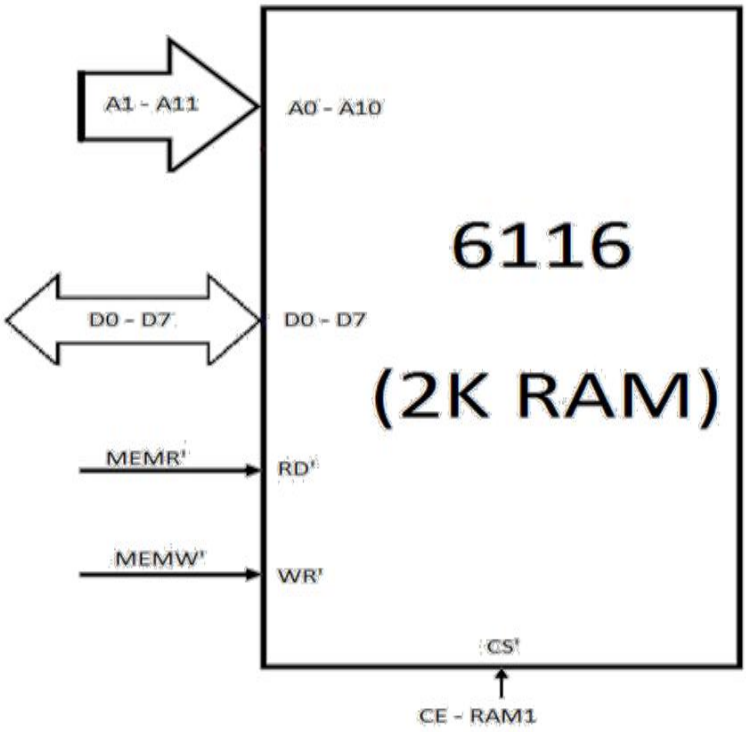
8086 inputs

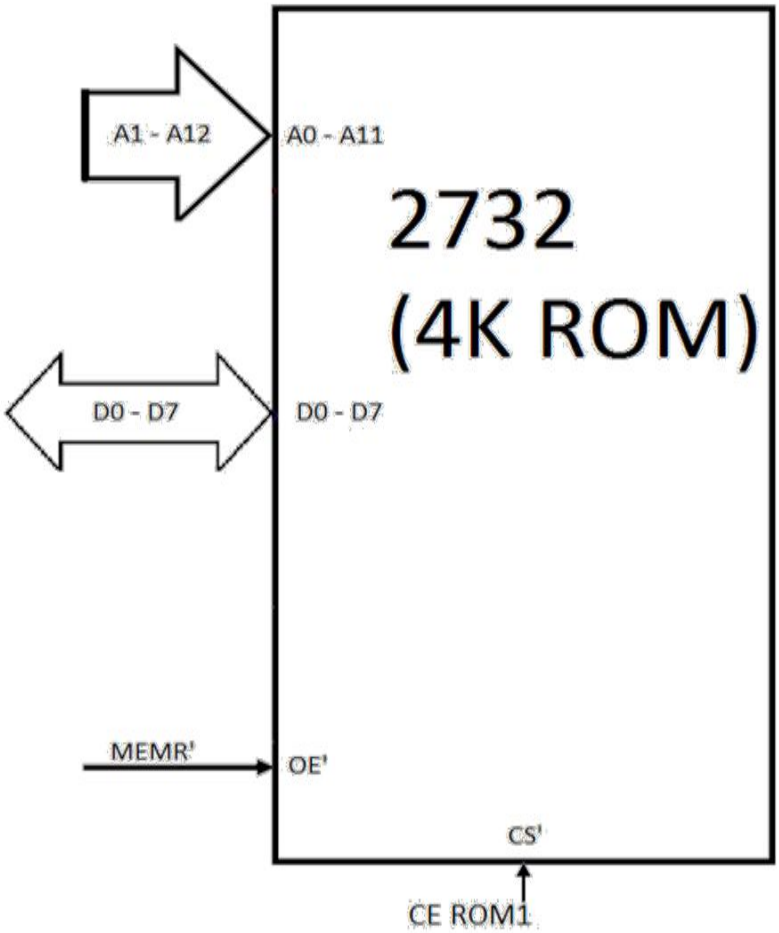


Clock Generator (8284)

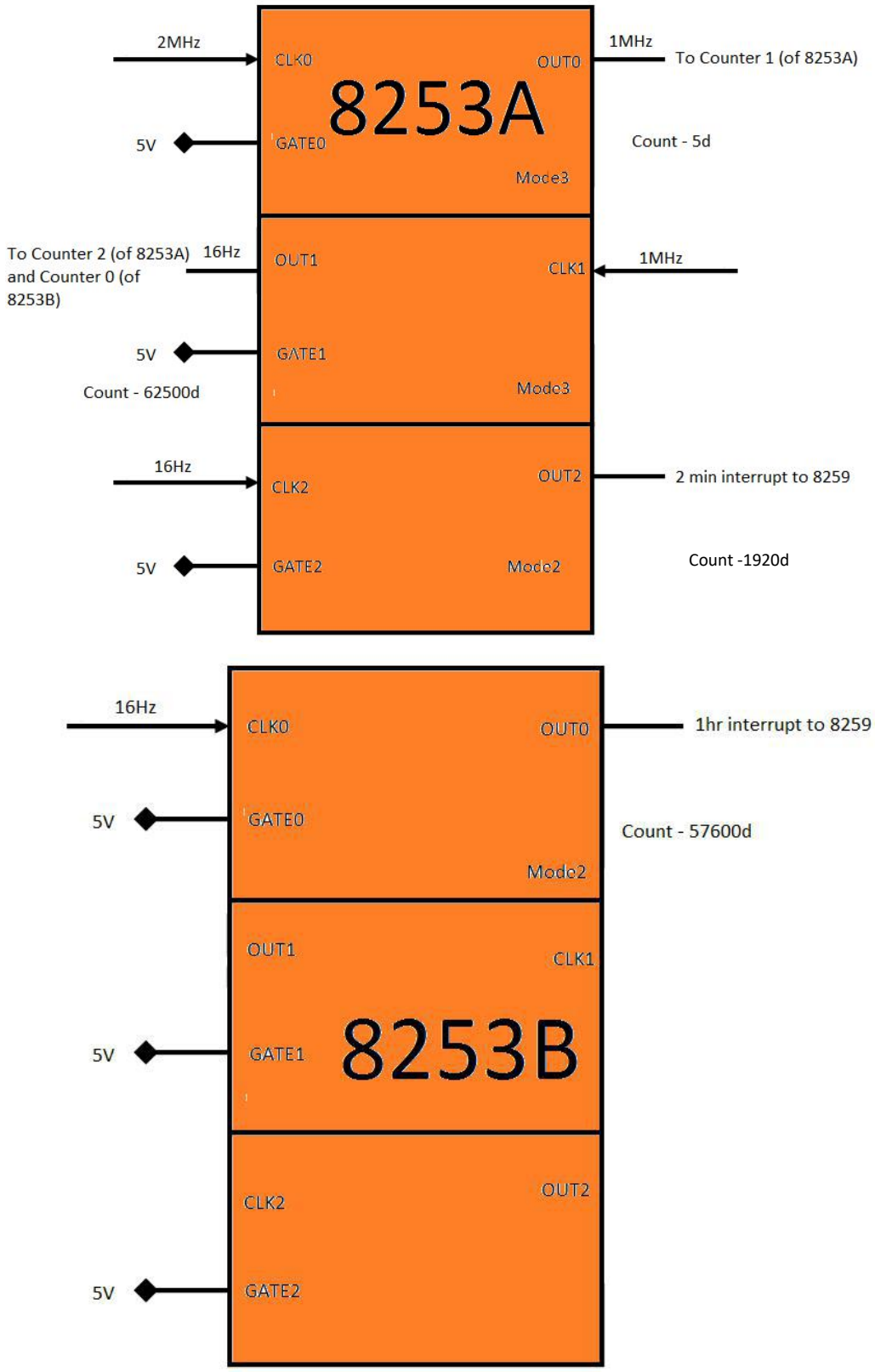
Memory Interfacing:



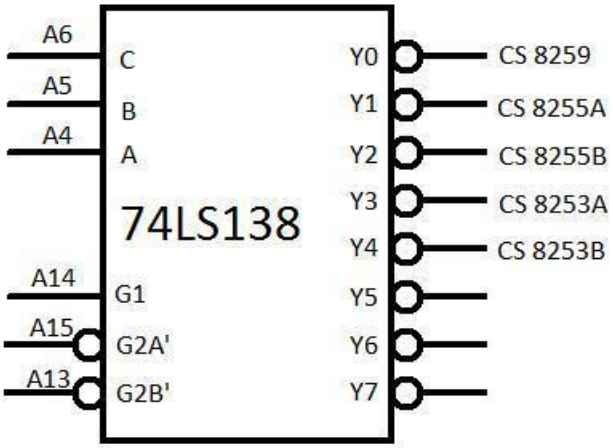

A1 - A11
A0 - A10
6116
D0 - D7
D0 - D7
(2K RAM)
MEMR'
RD'
MEMW'
WR'
CS'
CE - RAM1


A1 - A12
A0 - A11
2732
(4K ROM)
D0 - D7
D0 - D7
MEMR'
OE'
CS'
CE ROM1

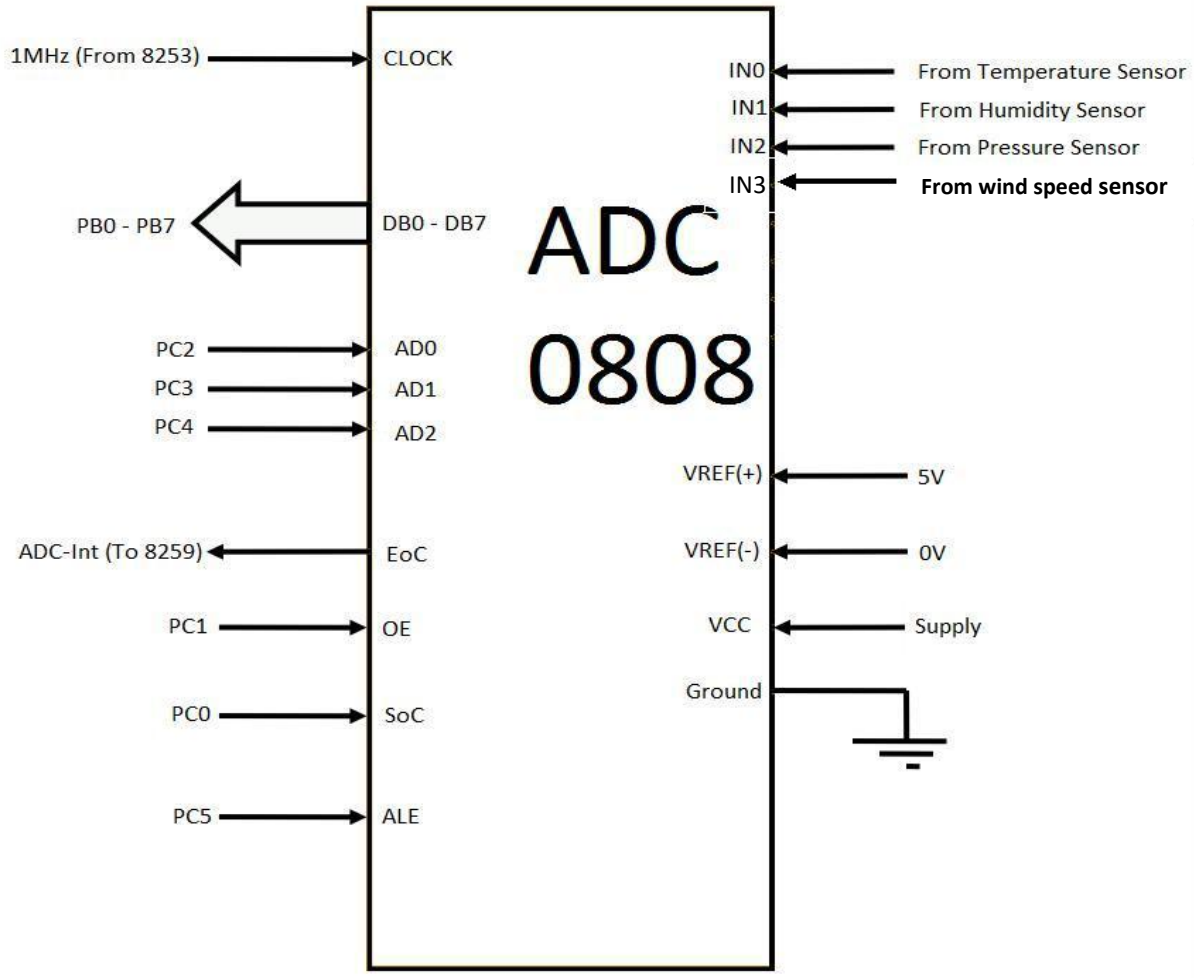
8253-Timer:


2MHz
CLK0
OUT0
1MHz
To Counter 1 (of 8253A)
8253A
5V
GATE0
Count - 5d
Mode3
To Counter 2 (of 8253A) and Counter 0 (of 8253B)
16Hz
OUT1
CLK1
1MHz
5V
GATE1
Count - 62500d
Mode3
16Hz
CLK2
OUT2
2 min interrupt to 8259
5V
GATE2
Mode2
Count -1920d
16Hz
CLK0
OUT0
1hr interrupt to 8259
5V
GATE0
Count - 57600d
Mode2
OUT1
CLK1
5V
GATE1
8253B
CLK2
OUT2
5V
GATE2

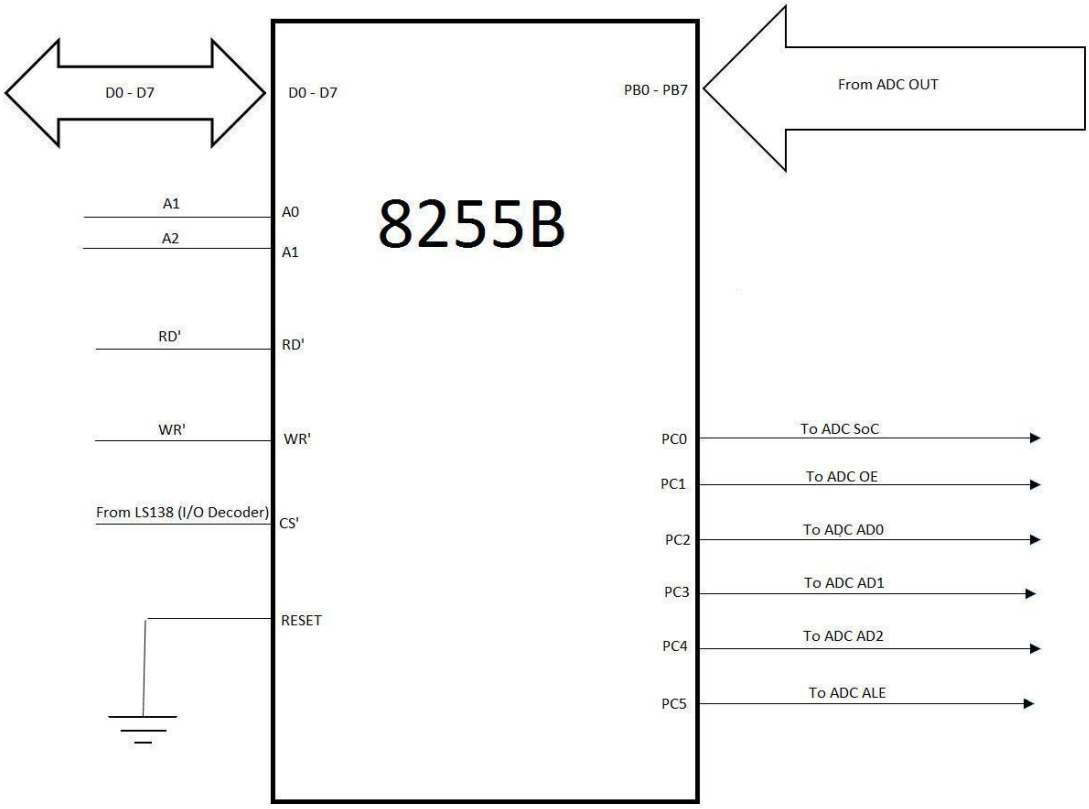
IO-decoder



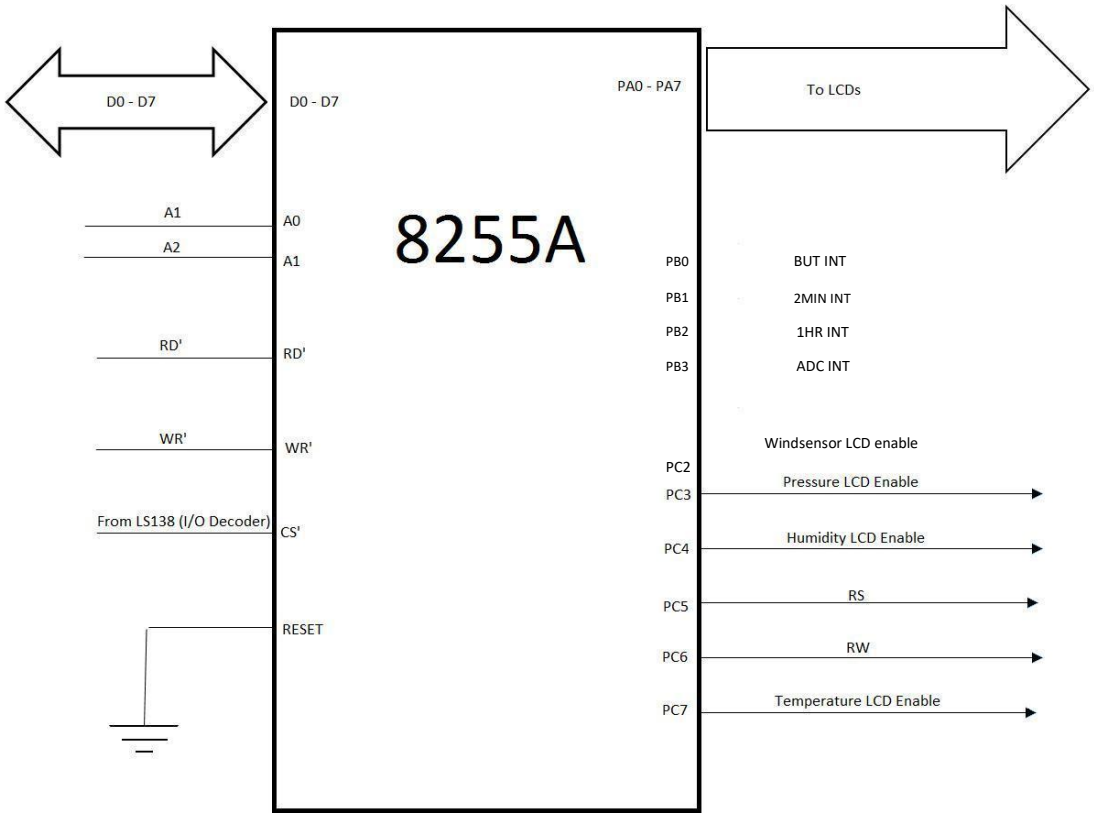
ADC:



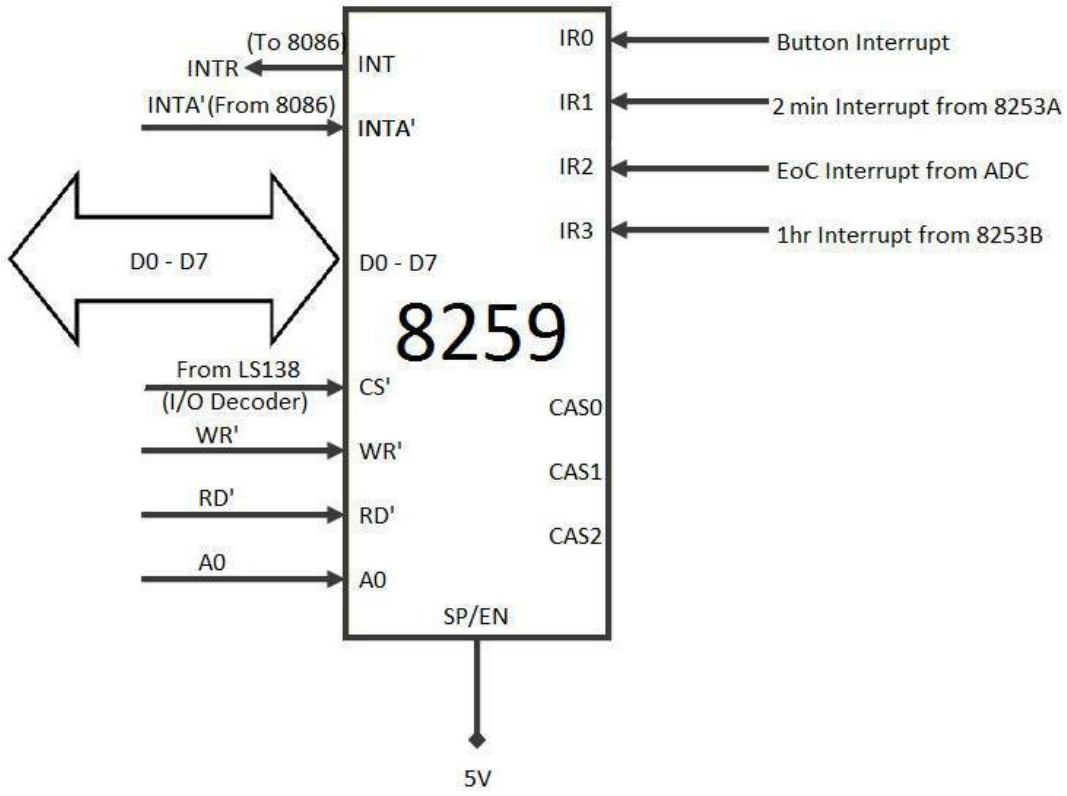
8255-B: (For the ADC)



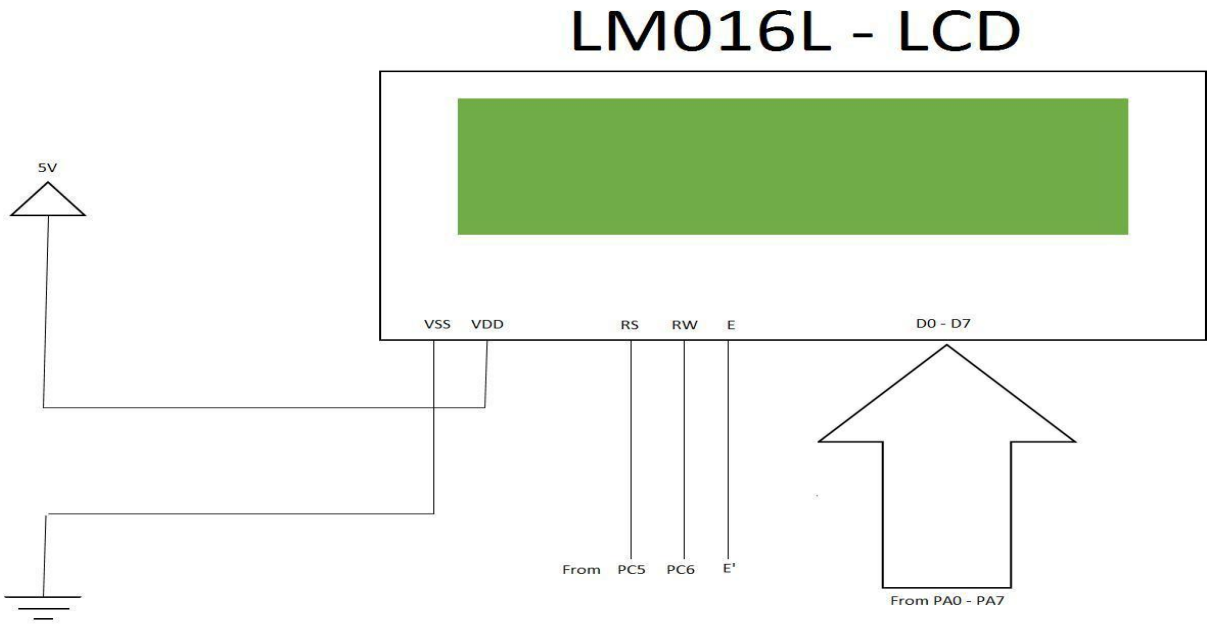
8255-A: (For display)



8259:

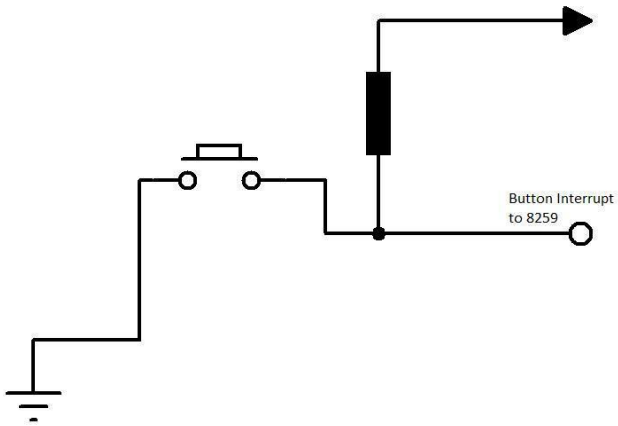


Display:



E' IS PC7 FOR TEMP LCD, PC4 FOR HUMIDITY LCD, PC2 FOR PRESSURE LCD,PC2 FOR WIND SPEED

Button



DATASHEETS FOR SENSORS

Lm016l- display
http://www.datasheetlib.com/datasheet/1311536/lm016l_hitachi-semiconductor.html
Temperature sensor(AD8494)
https://www.digchip.com/data/041/041-93390-AD8494.pdf
Humidity sensor(HF 323)
https://www.digchip.com/datasheets/parts/datasheet/816/HF3223-pdf.php
Pressure sensor (KP125)
http://media.digikey.com/PDF/Data%20Sheets/Infineon%20PDFs/KP125_V1+0_DS_Rev2+14.pdf
WE 550 Wind Speed Sensor
http://www.globalw.com/products/we550.html#Specifications

CODE

```
#LOAD_SEGMENT=0100h#
#LOAD_OFFSET=0000h#

#CS=0100h#
#IP=0000h#

#DS=0000h#
#ES=0000h#

#SS=0000h#
#SP=FFFEh#

#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
#BP=0000h#

;-------------------------IVT-init-----------------------;
mov ax, offset isr0
mov [00200h], ax
mov ax, seg isr0
mov [00202h], ax

mov ax, offset isr1
mov [00204h], ax
mov ax, seg isr1
mov [00206h], ax

mov ax, offset isr2
mov [00208h], ax
mov ax, seg isr2
mov [0020Ah], ax

mov ax, offset isr3
mov [0020Ch], ax
mov ax, seg isr3
mov [0020Eh], ax

;-------------------------IVT-init-----------------------;
jmp start
db 512 dup(0)


;-------------------------Data-segment-----------------------;
cstatea db 00h
cstateb db 00h
lcdln1 db 'Temp(C): '
lcdln2 db 16 dup('-')
lcdln3 db 16 dup('.')
lcdln4 db 16 dup('*')
```

```
lcdcnt1 db 9d
lcdcnt2 db 16d
lcdcnt3 db 16d
lcdcnt4 db 16d



flagcount       dw 0
vals            db 30 dup(0)
ctr             dw 0
readyForHour db 1 dup(0)

thp db 1 dup(0)


lcdln11 db 'Humi1(%): '
lcdln22 db 16 dup('-')
lcdln33 db 16 dup('.')
lcdln44 db 16 dup('*')
lcdcnt11 db 9d
lcdcnt22 db 16d
lcdcnt33 db 16d
lcdcnt44 db 16d



;Humi~
flagcount11     dw 0
vals11 ctr11    db 30 dup(0)
                dw 0

;------------
lcdln111 db 'Pres(Ba):'
lcdln222 db 16 dup('-')
lcdln333 db 16 dup('.')
lcdln444 db 16 dup('*')
lcdcnt111 db 9d
lcdcnt222 db 16d
lcdcnt333 db 16d
lcdcnt444 db 16d


;Pres
flagcount111    dw 0
vals111         db 30 dup(0)
ctr111          dw 0

;-------------

lcdln1111 db 'Spd(mph):'
lcdln2222 db 16 dup('-')
lcdln3333 db 16 dup('.')
lcdln4444 db 16 dup('*')
lcdcnt1111 db 9d
lcdcnt2222 db 16d
lcdcnt3333 db 16d
lcdcnt4444 db 16d
```

```
;Pres
flagcount1111
vals1111
ctr1111
dw 0
db 30 dup(0)
dw
numstr db 16 dup(0)

q db 0
r db 0

divby dw 30d
updatenow db 00h


;------------------------start-inits-----------------------;

start: cli

a8259 equ 4000h
a8255 equ 4010h
b8255 equ 4020h
a8253 equ 4030h
b8253 equ 4040h


8259_init:

;icw1
mov al, 00010011b  ;ICW4 Needed (single 8259)
mov dx, a8259+00h  ; dx has 1st address of 8259
out dx, al

;icw2
mov al, 10000000b  ; dx has 2nd address of 8259
mov dx, a8259+02h ; 80h is generated for IR0 ie But-INT
out dx,al

;icw4
mov al, 00000011b       ; rest follow 80h - 87h
out dx,al

;ocw1
mov al, 11111110b ; non buffered mode with AEOI enabled
out dx, al

; Initialising 8255A ...

8255_init:
mov al, 10000010b       ;Cmnd Word - port a(o/p), !(prt B: i/p)
mov dx, a8255+06h
out dx, al

; !(Same as prev 8255) B - i/p C - for controlling ADC

mov al, 10000010b
mov dx, b8255+06h
```

```
out dx, al

8253_init:          ; counter0 - sq. wave - binary i/p(2MHz-i/p)
;1Mhz
mov al, 00010110b
mov dx, a8253+06h

out dx, al
mov al, 02h
mov dx, a8253+00h      ; To divide by 2 - to give 1MHz
out dx, al


;16hz
mov al, 01110110b      ;counter1 - sq. wave - binary i/p
mov dx, a8253+06h
out dx, al
mov al, 24h            ; count = 62500 = 0f424h
mov dx, a8253+02h
out dx, al
mov al, 0F4h
out dx, al

;2min
mov al, 10110100b      ;cntr3 - Using mode 2 - every 2 min(low)
mov dx, a8253+06h      ;Must be inverted and given as interrupt
out dx, al
mov al, 80h
mov dx, a8253+04h
out dx, al
mov al, 07h
out dx, al

;1hr
mov al, 00110100b      ;counter 0 - 16Hz to 1Hr pulse
mov dx, b8253+06h
out dx, al
mov al, 00h
mov dx, b8253+00h
out dx, al
mov al, 0E1h
out dx, al

;------------For temperature
LCD_init:
lcden equ 80h
lcdrw equ 40h
lcdrs equ 20h

aclrb lcdrw
lcd_out 38h
lcd_out 0Eh
lcd_out 06h

lcd_clear
;------------For humidity

LCD_init1:
lcden1 equ 10h
lcdrw equ 40h
```

```
lcdrs equ 20h

aclrb lcdrw
lcd_out1 38h
lcd_out1 0Eh
lcd_out1 06h

lcd_clear
;------------For pressure

LCD_init11:
lcden11 equ 08h
lcdrw equ 40h
lcdrs equ 20h

aclrb lcdrw
lcd_out11 38h
lcd_out11 0Eh
lcd_out11 06h

lcd_clear

;------------For wind speed
LCD_init112:
lcden112 equ 04h
lcdrw equ 40h
lcdrs equ 20h

aclrb lcdrw
lcd_out112 38h
lcd_out112 0Eh
lcd_out112 06h

lcd_clear

;-------------------------start-code-------------------------;

; Perform Initial display ...

;turn on adc - temperature
mov thp,00h
int 81h

;wait for eoc

eocint08:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jnz eocint08

eocint18:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jz eocint18

;Store Values - Temperature
```

```
mov thp,00h
int 83h

;Repeat process for Humi~ty
;--------------------------
mov thp,01h
int 81h     ; do same for humi~

;wait for eoc
eocint010:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jnz eocint010

eocint101:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jz eocint101


mov thp,01h
int 83h
;--------------------------
; FOR pressure

mov thp,11h
int 81h     ; do same for pressure

;wait for eoc
eocint0109:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jnz eocint0109

eocint1018:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jz eocint1018


mov thp,11h
int 83h
;------------------
;For windpseed
 mov thp,12h
int 81h     ; do same for wspeed

;wait for eoc
eocint0111:
mov dx, a8255+02h
```

```
in al, dx
mov bl, al
and bl, 08h
jnz eocint0111

eocint1111:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h

jz eocint1111


mov thp,12h
int 83h
;----------
;---------- end
mov thp,00h
int 82h
mov thp,01h
int 82h
mov thp,11h
int 82h
mov thp,12h
int 82h

; --------END of initial display ----------


;poll portb of a8255 forever
xinf:

;check if button is pressed using a flag stored in memory
mov al, updatenow
cmp al, 01h
jnz cont
mov updatenow, 00h

mov thp,00h        ;FOR TEMPERATURE
int 81h ;turn on adc

;wait for eoc
eocint0:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jnz eocint0

eocint1:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jz eocint1
```

```
mov thp,00h
int 83h
;-----------------------------
mov thp,01h
int 81h    ; do same for humi

;wait for eoc
eocint00:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jnz eocint00

eocint11:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jz eocint11


mov thp,01h
int 83h


;--------------------------
mov thp,11h ; FOR PRESSURE
int 81h



;wait for eoc
eocint000:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jnz eocint000

eocint111:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jz eocint111

mov thp,11h
int 83h
;--------------------
mov thp,12h ; FOR Windspeed
int 81h



;wait for eoc
eocint002:
mov dx, a8255+02h
in al, dx
mov bl, al
```

```
and bl, 08h
jnz eocint002

eocint112:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jz eocint112

mov thp,12h
int 83h
;-----------------------------


mov thp,00h
int 82h
mov thp,01h
int 82h
mov thp,11h
int 82h
mov thp,12h  ;
int 82h



;regular polling
cont:
mov dx, a8255+02h
in al, dx

mov bl, al
and bl, 01h
jz butint

mov bl, al
and bl, 02h
jz twomin

mov bl, al
and bl, 04h
jz onehr

mov bl, al
and bl, 08h
jz eocint
jmp xinf

;low logic detected. Wait for whole pulse
butint:
in al, dx
and al, 01h
jz butint

int 80h
jmp xinf

twomin:
in al, dx
```

```
and al, 02h
jz twomin
;----------------------Next part starts here
mov thp,00h
int 81h

eocint09:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jnz eocint09

eocint19:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jz eocint19

mov thp,00h
int 83h

;---------------
mov thp,01h
int 81h


;wait for eoc
eocint009:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jnz eocint009


eocint119:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jz eocint119

mov thp,01h
int 83h

;------------------
mov thp,11h
int 81h



;wait for eoc
eocint00999:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jnz eocint00999
```

```
eocint11999:
mov dx, a8255+02h
in al, dx
mov bl, al
and bl, 08h
jz eocint11999

mov thp,11h
int 83h


;Change value of no. of 2 min intervals taken during simulation

inc readyForHour

cmp readyForHour,02h

jnz donotcallonehour

mov thp,00h
int 82h ; Call the 1 hour interrupt
mov thp,01h
int 82h
mov thp,11h
int 82h
mov thp,12h
int 82h

mov readyForHour,00h ; Reset the 30, 2-min interval count


donotcallonehour: jmp xinf


; -------

onehr:
in al, dx
and al, 04h
jz onehr

int 82h
jmp xinf

eocint:
in al, dx
and al, 08h
jz eocint

mov thp,00h
int 83h
mov thp,01h
int 83h
mov thp,11h
int 83h


jmp xinf
```

```
jmp quit



;------------------------start-macros----------------------;
pushall macro
   push ax
   push bx
   push cx
   push dx
   push si
   push di
endm

popall macro
   pop di
   pop si
   pop dx
   pop cx
   pop bx
   pop ax
endm

;set/clear pins since BSR was faulty
asetb   macro mbit
   pushall
   mov al, mbit
   mov bl, cstatea
   or al, bl
   mov dx, a8255+04h
   out dx, al
   mov bl, al
   mov cstatea, bl
   popall
endm


aclrb   macro mbit
   pushall
   mov al, mbit
   xor al, 0FFh
   mov bl, cstatea
   and al, bl
   mov dx, a8255+04h
   out dx, al
   mov bl, al
   mov cstatea, bl
   popall
endm

adcst equ 01h
adcoe equ 02h
adcA equ 04h
adcB equ 08h
adcC equ 10h
adcALE equ 20h
```

```
bsetb   macro mbit
   pushall
   mov al, mbit
   mov bl, cstateb
   or al, bl
   mov dx, b8255+04h
   out dx, al
   mov bl, al
   mov cstateb, bl
   popall
endm


bclrb   macro mbit
   pushall
   mov al, mbit
   xor al, 0FFh
   mov bl, cstateb
   and al, bl
   mov dx, b8255+04h
   out dx, al
   mov bl, al
   mov cstateb, bl
   popall
endm

lcd_out   macro dat
   aclrb lcdrs
   pushall
   mov al, dat
   mov dx, a8255+00h
   out dx, al
   asetb lcden
   aclrb lcden
   call delay_20ms
   popall
endm


lcd_out1   macro dat
   aclrb lcdrs
   pushall
   mov al, dat
   mov dx, a8255+00h
   out dx, al
   asetb lcden1
   aclrb lcden1
   call delay_20ms
   popall
endm

lcd_out11   macro dat
   aclrb lcdrs
   pushall
   mov al, dat
   mov dx, a8255+00h
   out dx, al
   asetb lcden11
```

```
   aclrb lcden11
   call delay_20ms
   popall
endm

lcd_out112    macro dat
   aclrb lcdrs
   pushall
   mov al, dat
   mov dx, a8255+00h
   out dx, al
   asetb lcden112
   aclrb lcden112
   call delay_20ms
   popall
endm


lcd_write_char macro dat
   asetb lcdrs
   pushall
   mov al, dat
   mov dx, a8255+00h
   out dx, al
   asetb lcden
   aclrb lcden
   ;call delay_20ms
   popall

endm


lcd_write_char1 macro dat
   asetb lcdrs
   pushall
   mov al, dat
   mov dx, a8255+00h
   out dx, al
   asetb lcden1
   aclrb lcden1
   ;call delay_20ms
   popall

endm

lcd_write_char11 macro dat
   asetb lcdrs
   pushall
   mov al, dat
   mov dx, a8255+00h
   out dx, al
   asetb lcden11
   aclrb lcden11
   ;call delay_20ms
   popall

endm

lcd_write_char112 macro dat
   asetb lcdrs
```

```
   pushall
   mov al, dat
   mov dx, a8255+00h
   out dx, al
   asetb lcden112
   aclrb lcden112
   ;call delay_20ms
   popall

endm

lcd_clear macro
   lcd_out 01h
endm

;division routine since div was acting strange
wow_divide macro divi

   pushall
   mov cx, 00
   mov bx, divi

   loopy:
   sub ax, bx
   inc cx
   cmp ax, 0
   jge loopy

         dec cx
         add ax, bx

         mov r, al
         mov q, cl

         popall
endm


wow_divide1 macro divi

   pushall
   mov cx, 00
   mov bx, divi

   loopy1:
   sub ax, bx
         inc cx
   cmp ax, 0
   jge loopy1

         dec cx
         add ax, bx

         mov r, al
         mov q, cl

         popall
endm

wow_divide2 macro divi
```

```
   pushall
   mov cx, 00
   mov bx, divi

   loopy2:
   sub ax, bx
            inc cx
   cmp ax, 0
   jge loopy2

            dec cx
            add ax, bx

            mov r, al
            mov q, cl

            popall
endm

wow_divide3 macro divi

   pushall
   mov cx, 00
   mov bx, divi
   loopy3:
   sub ax, bx
            inc cx
   cmp ax, 0
   jge loopy3

            dec cx
            add ax, bx

            mov r, al
            mov q, cl

            popall
endm

;-------------------------Start-Procedure Defs-----------------------;

;delay proc: just a huge loop
delay_20ms proc near
   mov dx, 10
r1: mov cx, 2353
r2: loop r2
   dec dx
   jne r1
   ret
delay_20ms endp


;write a string in memory to LCD
write_string proc near
   lea si, lcdln1
   mov cl, lcdcnt1
l1: lcd_write_char [si]
   inc si
   loop l1
```

```
   ret
write_string endp


;write a string in memory to LCD
write_string1 proc near
   lea si, lcdln11
   mov cl, lcdcnt11
l11: lcd_write_char1 [si]
   inc si
   loop l11
   ret
write_string1 endp


;write a string in memory to LCD
write_string11 proc near
   lea si, lcdln111
   mov cl, lcdcnt111
l111: lcd_write_char11 [si]
   inc si
   loop l111
   ret
write_string11 endp

;write a string in memory to LCD
write_string111 proc near
   lea si, lcdln1111
   mov cl, lcdcnt1111
l112: lcd_write_char112 [si]
   inc si
   loop l112
   ret
write_string111 endp



;-----------------------------------------------
;Scale Humidity

convert_humi proc near

   ;get it to scale (0-99%)
   mov ah, 00h
   mov al, q
   mov bl, 99d
   mul bl
   mov bl, 0FFh          ;FFh is the max o/p from ADC
   div bl


   ;split the numbers
   mov ah, 00h
   mov bl, 10d
   div bl

   lea si, numstr        ;Load appropriate ascii value(quo)
   add ax, 3030h
   mov [si], al
   mov [si+1], ah
```

```
mov al, r
mov ah, 00h

mov bx, 100d
mul bx
mov bl, 12d
div bl

mov ah, 00h
mov bl, 10d
div bl
add ax, 3030h

mov  [si+2],  al    ;Load appropriate ascii value(rem)
mov [si+3], ah

ret
```

```
convert_humi endp

; Scaling fns ---------------------------------------------

;Scale temperature
convert_temp proc near


   ;get it to scale (5-50 C)
   mov ah, 00h
   mov al, q
   mov bl, 45d
   mul bl
   mov bl, 0FFh
   div bl
            add ax, 05h

   ;split the numbers
   mov ah, 00h
   mov bl, 10d
   div bl

   lea si, numstr
   add ax, 3030h
   mov [si], al
   mov [si+1], ah

   mov al, r
   mov ah, 00h

   mov bx, 100d
   mul bx
   mov bl, 12d
   div bl

   mov ah, 00h
   mov bl, 10d
   div bl
   add ax, 3030h

   mov [si+2], al
   mov [si+3], ah

   ret

convert_temp endp


;-------------------------------------------------------------------------
;Scale Pressure

convert_pres proc near

   ;get it to scale (0-2 Bar)
   mov ah, 00h
   mov al, q
   mov bl, 02d
   mul bl
```

```
    mov bl, 0FFh          ;FFh is the max o/p from ADC
    div bl


    ;split the numbers
    mov ah, 00h
    mov bl, 10d
    div bl

    lea si, numstr        ;Load appropriate ascii value(quo)
    add ax, 3030h
    mov [si], al
    mov [si+1], ah

    mov al, r
    mov ah, 00h

    mov bx, 100d
    mul bx
    mov bl, 12d
    div bl

    mov ah, 00h
    mov bl, 10d
    div bl
    add ax, 3030h

    mov [si+2], al     ;Load appropriate ascii value(rem)
    mov [si+3], ah

    ret

convert_pres endp

;----------------------------
;Scale wind speed
convert_wind proc near

    ;get it to scale (0-30mph)
    mov ah, 00h
    mov al, q
    mov bl, 30d
    mul bl
    mov bl, 0FFh          ;FFh is the max o/p from ADC
    div bl


    ;split the numbers
    mov ah, 00h
    mov bl, 10d
    div bl

    lea si, numstr        ;Load appropriate ascii value(quo)
    add ax, 3030h
    mov [si], al
    mov [si+1], ah
```

```
   mov al, r
   mov ah, 00h

   mov bx, 100d
   mul bx
   mov bl, 12d
   div bl

   mov ah, 00h
   mov bl, 10d
   div bl
   add ax, 3030h
   mov [si+2], al

;Load appropriate ascii value(rem
mov [si+3], ah

   ret

convert_wind endp
;-----------------------------

;output ascii equiv values on LCD1 from mem location

num_out proc near

   lcd_out 01h
   call write_string

   mov al, numstr
   mov ah, numstr+1
   lcd_write_char al
   lcd_write_char ah
   lcd_write_char '.'
   mov al, numstr+2
   mov ah, numstr+3

   lcd_write_char al
   lcd_write_char ah


   ret

num_out endp


;output ascii equiv values on LCD2 from mem location
num_out1 proc near

   lcd_out1 01h
   call write_string1

   mov al, numstr
   mov ah, numstr+1
   lcd_write_char1 al
   lcd_write_char1 ah
   lcd_write_char1 '.'
   mov al, numstr+2
   mov ah, numstr+3
```

```
   lcd_write_char1 al
   lcd_write_char1 ah


   ret

num_out1 endp


;output ascii equiv values on LCD3 from mem location
num_out11 proc near

   lcd_out11 01h
   call write_string11

   mov al, numstr
   mov ah, numstr+1
   lcd_write_char11 al
   lcd_write_char11 ah
   lcd_write_char11 '.'
   mov al, numstr+2
   mov ah, numstr+3

   lcd_write_char11 al
   lcd_write_char11 ah


   ret

num_out11 endp

;output ascii equiv values on LCD4 from mem location
num_out112 proc near

   lcd_out112 01h
   call write_string111

   mov al, numstr
   mov ah, numstr+1
   lcd_write_char112 al
   lcd_write_char112 ah
   lcd_write_char112 '.'
   mov al, numstr+2
   mov ah, numstr+3

   lcd_write_char112 al
   lcd_write_char112 ah


   ret

num_out112 endp

; ------------------------- End of procedure Defs -------------------------

;-----------------------Start Of ISRs----------------------;

;5 minute interrupt
isr1:
```

```
        ; First make OE high
        PC1 bsetb adcoe


cmp thp,00h
jnz humiisr1

        ;Assuming that CBA is connected to PC 4-3-2
        ;select channel 000
        bclrb adcA
        bclrb adcB
        bclrb adcC

        ;Now make a high-low pulse on ALE;PC5
        bsetb adcALE
        bclrb adcALE
        ;High-low pulse on SOC - connected to PC0
        bsetb adcst
        bclrb adcst

        ;now wait for EOC interrupt
jmp isr1end
humiisr1: cmp thp,01h jnz presisr1
```

```
; thp == 1
;select channel 001
        bsetb adcA
        bclrb adcB
        bclrb adcC

        ;Now make a high-low pulse on ALE;PC5
        bsetb adcALE
        bclrb adcALE

        ;High-low pulse on SOC - connected to PC0
        bsetb adcst
        bclrb adcst

   jmp isr1end

   presisr1:
cmp thp,11h
   jnz windisr1

   ;select channel 010
        bclrb adcA
        bsetb adcB
        bclrb adcC

        ;Now make a high-low pulse on ALE;PC5
        bsetb adcALE
        bclrb adcALE

        ;High-low pulse on SOC - connected to PC0
        bsetb adcst
        bclrb adcst

        windisr1:



        ;select channel 011
        bsetb adcA
        bsetb adcB
        bclrb adcC

        ;Now make a high-low pulse on ALE;PC5
        bsetb adcALE
        bclrb adcALE

        ;High-low pulse on SOC - connected to PC0
        bsetb adcst
        bclrb adcst

        ;now wait for EOC interrupt

isr1end:

iret
;-----------------------------------------
;-----------------------------------------

;EOC interrupt
```

```
isr3:

        tempisr3:
   cmp thp,00h
   jnz humiisr3


        mov dx, b8255+02h
        in al, dx

        ;Finally make OE low
        bclrb adcoe


        cmp flagcount, 0
        jnz x4
        ;for the first hour, flagcnt = 0; for consecutive iterations, it'll be >0

        mov bx, ctr
        lea si, vals

        mov [si+bx], al
        inc bx
        mov ctr, bx
        cmp bx, 30
        jnz x5

        mov flagcount, 1
        mov ctr, 0
        jmp endisr1

        x4:     mov bx, ctr
        lea si, vals
        mov [si+bx], al
        inc bx
        cmp bx, 30
        jnz x5
        mov bx, 0

        x5:     mov ctr, bx

   jmp endisr1

   humiisr3:    ;thp == 1
   cmp thp,01h
   jnz presisr3


        mov dx, b8255+02h
        in al, dx

        ;Finally make OE low
```

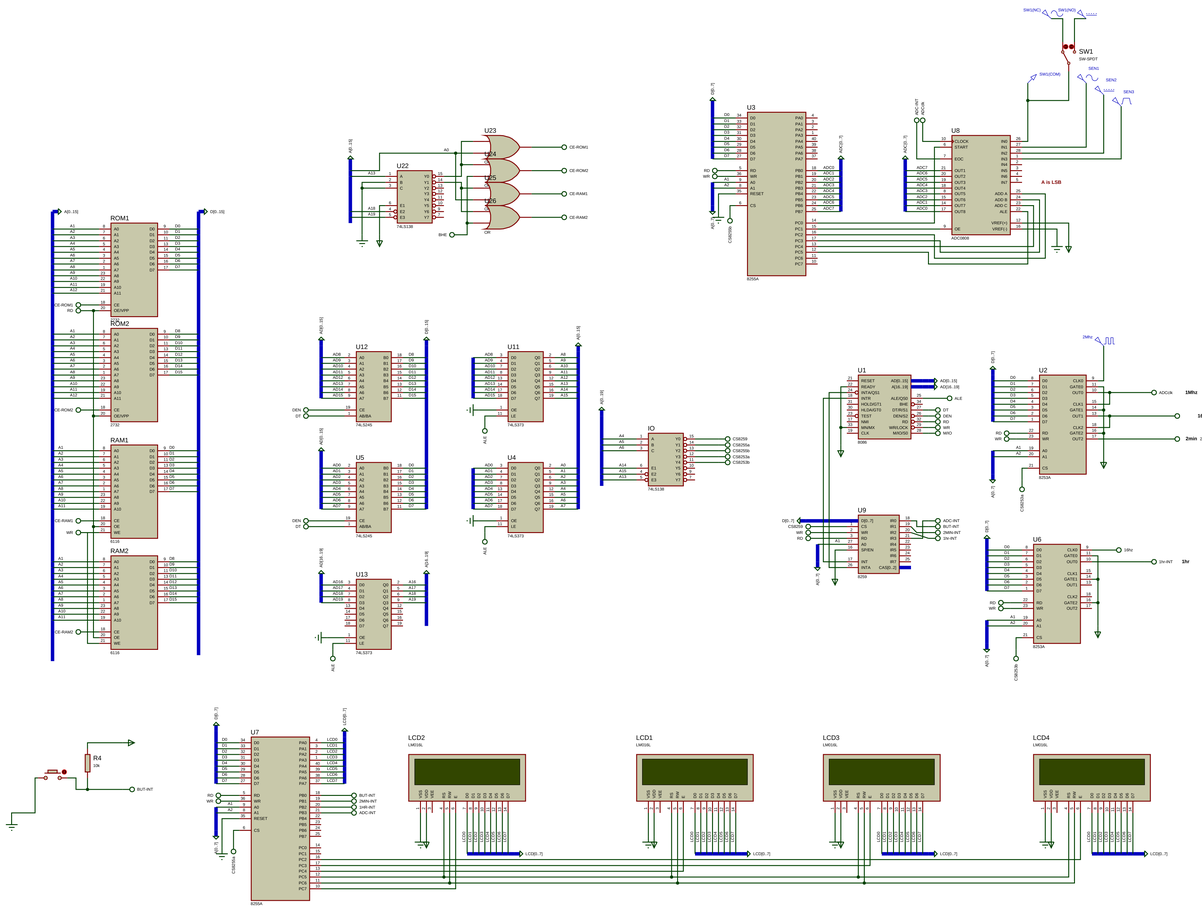
```
        bclrb adcoe


        cmp flagcount11, 0
        jnz x41

        mov bx, ctr11
        lea si, vals11

        mov [si+bx], al
        inc bx
        mov ctr11, bx
        cmp bx, 30
        jnz x51

        mov flagcount11, 1
        mov ctr11, 0
        jmp endisr1

        x41:    mov bx, ctr11
        lea si, vals11
        mov [si+bx], al
        inc bx
        cmp bx, 30
        jnz x51
        mov bx, 0

        x51:    mov ctr11, bx
jmp endisr1

presisr3:
        cmp thp,11h
jnz windisr3

        mov dx, b8255+02h
        in al, dx

        ;Finally make OE low
        bclrb adcoe


        cmp flagcount111, 0
        jnz x411

        mov bx, ctr111
        lea si, vals111

        mov [si+bx], al
        inc bx
        mov ctr111, bx
        cmp bx, 30
        jnz x511

        mov flagcount111, 1
        mov ctr111, 0
        jmp endisr1

        x411:   mov bx, ctr111
        lea si, vals111
        mov [si+bx], al
```

```
        inc bx
        cmp bx, 30
        jnz x511
        mov bx, 0

        x511:    mov ctr111, bx


        windisr3:

        mov dx, b8255+02h
        in al, dx

        ;Finally make OE low
        bclrb adcoe


        cmp flagcount1111, 0
        jnz x4111
        ;for the first hour, flagcnt = 0; for consecutive iterations, it'll be >0

        mov bx, ctr1111

        lea si, vals1111

        mov [si+bx], al
        inc bx
        mov ctr1111, bx
        cmp bx, 30
        jnz x5111

        mov flagcount1111, 1
        mov ctr1111, 0
        jmp endisr1

        x4111:  mov bx, ctr1111
        lea si, vals1111
        mov [si+bx], al
        inc bx
        cmp bx, 30
        jnz x5111
        mov bx, 0

        x5111:  mov ctr1111, bx


        endisr1:
iret

;1hr int
isr2:

cmp  thp,00h
jnz humiisr2        ;--------------------
```

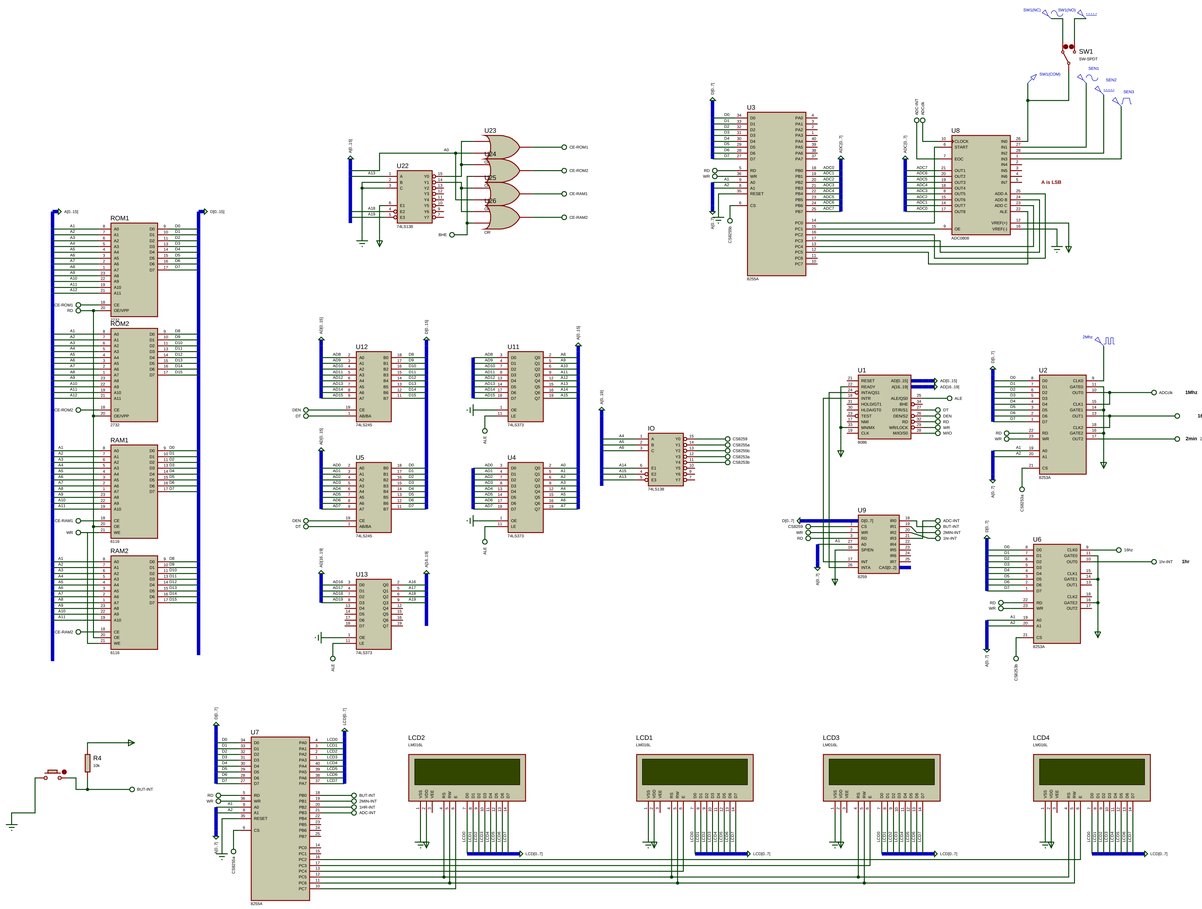
```
        mov bx, 00h
        mov cx, 30d
        lea si, vals

        xadd:
        mov dl, [si]
        mov dh, 00h
        add bx, dx
        inc si
        dec cx
        jnz xadd
        mov ax, bx

        mov dx, flagcount
        cmp dx, 1
        jnz x2

        mov divby, 30d
        jmp x3

        x2:
        mov dx, ctr
        mov divby, dx
        x3:
        wow_divide divby

        call convert_temp
        call num_out

jmp endisr2:

   humiisr2:    ;---------------
   cmp thp,01h
   jnz presisr2

        mov bx, 00h
        mov cx, 30d
        lea si, vals11

        xadd1:
        mov dl, [si]
        mov dh, 00h
        add bx, dx
        inc si
        dec cx
        jnz xadd1
        mov ax, bx

        mov dx, flagcount11
        cmp dx, 1
        jnz x21

        mov divby, 30d
        jmp x31

        x21:
        mov dx, ctr11
        mov divby, dx
        x31:
```

```
        wow_divide1 divby

        call convert_humi
        call num_out1
jmp endisr2
presisr2:   ;----------
cmp thp,11h
jnz windisr2

        mov bx, 00h
        mov cx, 30d
        lea si, vals111

        xadd11:
        mov dl, [si]
        mov dh, 00h
        add bx, dx
        inc si
        dec cx
        jnz xadd11
        mov ax, bx

        mov dx, flagcount111
        cmp dx, 1
        jnz x211

        mov divby, 30d
        jmp x311

        x211:
        mov dx, ctr111
        mov divby, dx
        x311:

        wow_divide2 divby

        call convert_pres
        call num_out11

windisr2:   ;----------

        mov bx, 00h
        mov cx, 30d
        lea si, vals1111

        xadd111:
        mov dl, [si]
        mov dh, 00h
        add bx, dx
        inc si
        dec cx
        jnz xadd111
        mov ax, bx

        mov dx, flagcount1111
        cmp dx, 1
        jnz x2111
```

```
        mov divby, 30d
        jmp x3111

        x2111:
        mov dx, ctr1111
        mov divby, dx
        x3111:

        wow_divide3 divby

        call convert_wind
        call num_out112

endisr2:

iret

;button interrupt
isr0:
        mov updatenow, 01h
iret

;-----------------------END OF ISRs-------------------------;



quit:
hlt
```



PALE NAME:	Final_Design Mod.DSN							DATE:	26-APR-18
DESIGN TITLE:	G:\Weather Station\Final\Final_Design Mod.DSN							PAGE:	1 of 1
PATH:	G:\Weather Station\Final\Final_Design Mod.DSN								
BY:	<NONE>							REV:<NONE>	
								TIME:	10:09:03 AM



PALE NAME:	Final_Design Mod.DSN							DATE:	26-APR-18
DESIGN TITLE:	G:\Weather Station\Final\Final_Design Mod.DSN							PAGE:	1 of 1
PATH:	G:\Weather Station\Final\Final_Design Mod.DSN								
BY:	<NONE>							REV:<NONE>	
								TIME:	10:09:03 AM