Problem Number: 13 **Group Number: 96**

Flour Packing Machine

By

GAJULA SAI SARATH KRISHNA	**2017A7PS0154P**
DESHMUKH ADVAIT MAHESH	**2017A7PS0155P**
ADITYA MITHAL	**2017A7PS0157P**
PAIDIPELLY HEMANTH RAO	**2017A7PS0159P**

CS F241 – Microprocessors Programming And Interfacing

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI RAJASTHAN -333031

April 25,2019

PROBLEM STATEMENT:

Design a Microprocessor based flour packing system. The flour to be packed is contained in a tower. The user keys-in the required amount of flour per packet which could be 5, 10 or 20kgs. The system should take the input and pack the specified amount of flour upon press of a START key. It is also required to monitor the temperature of the floor where packing is going on. This temperature range can be user settable which should also be displayed on a seven segment display. An alarm for any malfunctioning of the system like out of range temperature should be provided.

DESIGN SPECIFICATIONS:

Temperature Sensor

Calibrated in degree Celsius

Linear+10mV/degree Celsius scale factor

Operating voltage: 4-20 Volt

Output range: 2−150degreeCelsius

Accuracy: +-0.2degreeCelsius

Weight Sensor

Calibrated in pounds

Linear 10mV/ pound scale factor

Output Range: 0-500 pounds

ASSUMPTIONS

- User inputs the temperature in °C.
- Four temperature sensors (LM 35) are sufficient to monitor the packaging area.
- Minimum temperature entered is 2°C.
- Maximum temperature entered is 99°C.
- Minimum temperature range is less than maximum temperature range.
- Weigh per packet is entered in Kgs and is less than 99Kgs.
- All user inputs should be whole numbers.
- '+' button on keypad is taken to be 'Start'.
- '-' button on keypad is taken to be 'Weight'.
- '*' button on keypad is taken to be 'Temp.Higher'.
- '/' button keypad is taken to be 'Temp.Lower'.

COMPONENTS USED

Sr. No.	Components Used	Quantity	Purpose
1	8086	1	Central processor
2	8255	3	PPI for I/)
3	8253	1	Programmable interval timer
4	6116 RAM	1	RAM for the Memory
5	2732 ROM	1	EPROM
6	74LS138	1	Address Decoder
7	73LS373	11	Latching the bus
8	74LS245	2	Bi-Directional Buffer
9	L293D	1	Motor Driver
10	ADC0808	1	ADC 8 Channel 8 bit
11	7447	7	BCD to Seven Segment Display
12	Load Cell	1	Weight Sensor
13	LM35	4	Temperature Sensor
14	OR Gate	6	
15	Keypad	1	16 Key Matrix
16	7- Segment common anode Display	6	O/p Display
17	7-Segment common anode Multiplexed	1	O/p Display
18	LED	3	Output Status
19	74LS447	1	Decoder
20	2716 ROM	1	EPROM
21	Motor	1	For rotation of the Belt

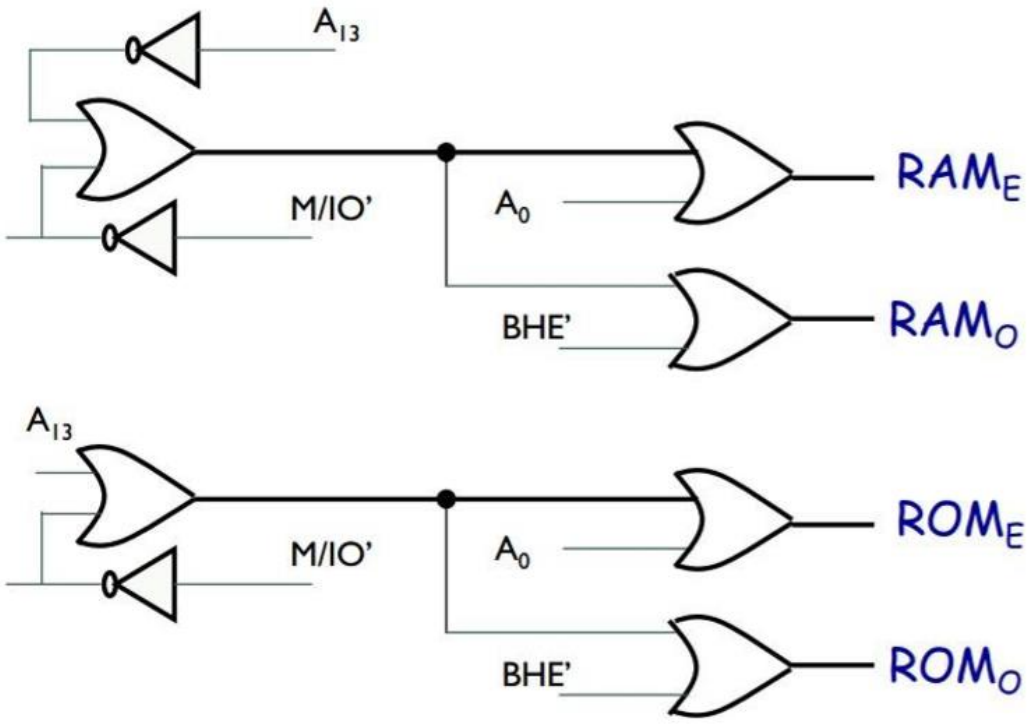
Complete Address Mapping of Memory and I/O Devices

RAM (min 2k chip): 4k

ROM (min 4k chip): 8k

ROM1: 00000h -01FFFh

RAM1 :02000h-02FFFh



8255(1) **Port A:** 00h

Port B: 02h

Port C: 04h

CWR: 06h

8255(2) **Port A: 10h**

Port B: 12h

Port C: 14h

CWR: 16h

8255(3) **Port A: 18h**

Port B: 1Ah

Port C: 1Ch

CWR: 1Eh

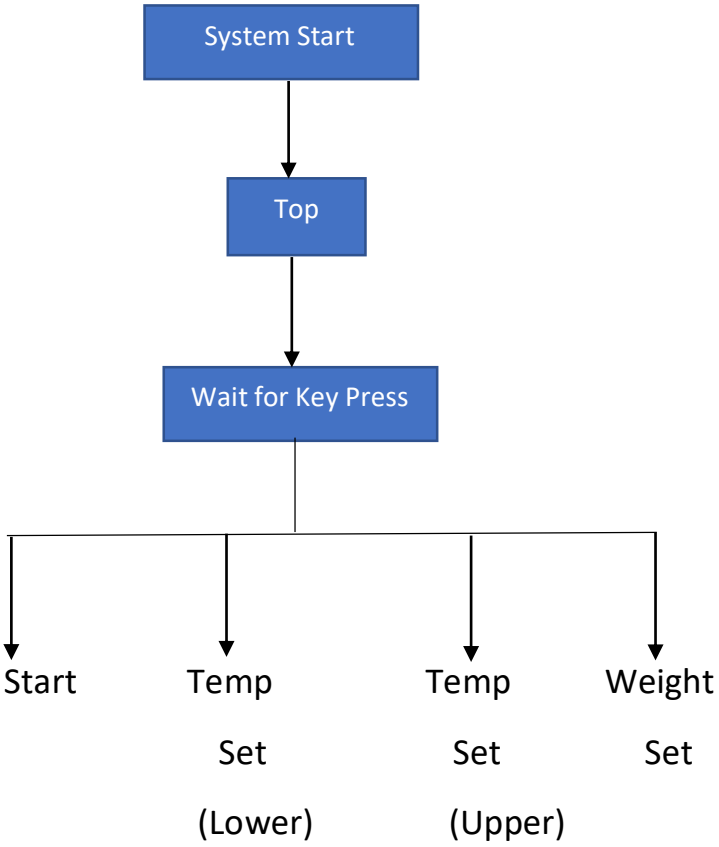
8253 **Counter 0: 8h**

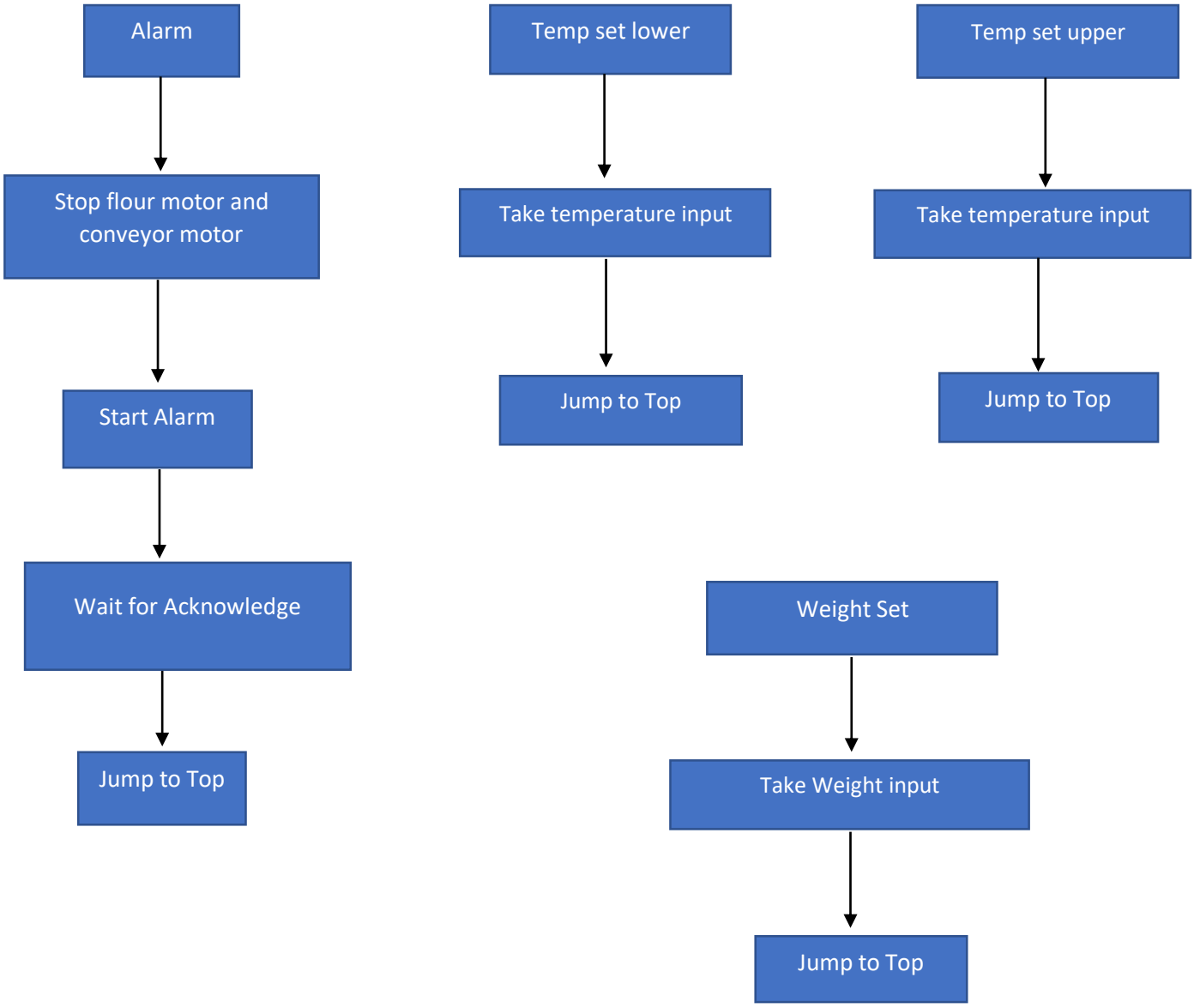
Counter 1: 0Ah

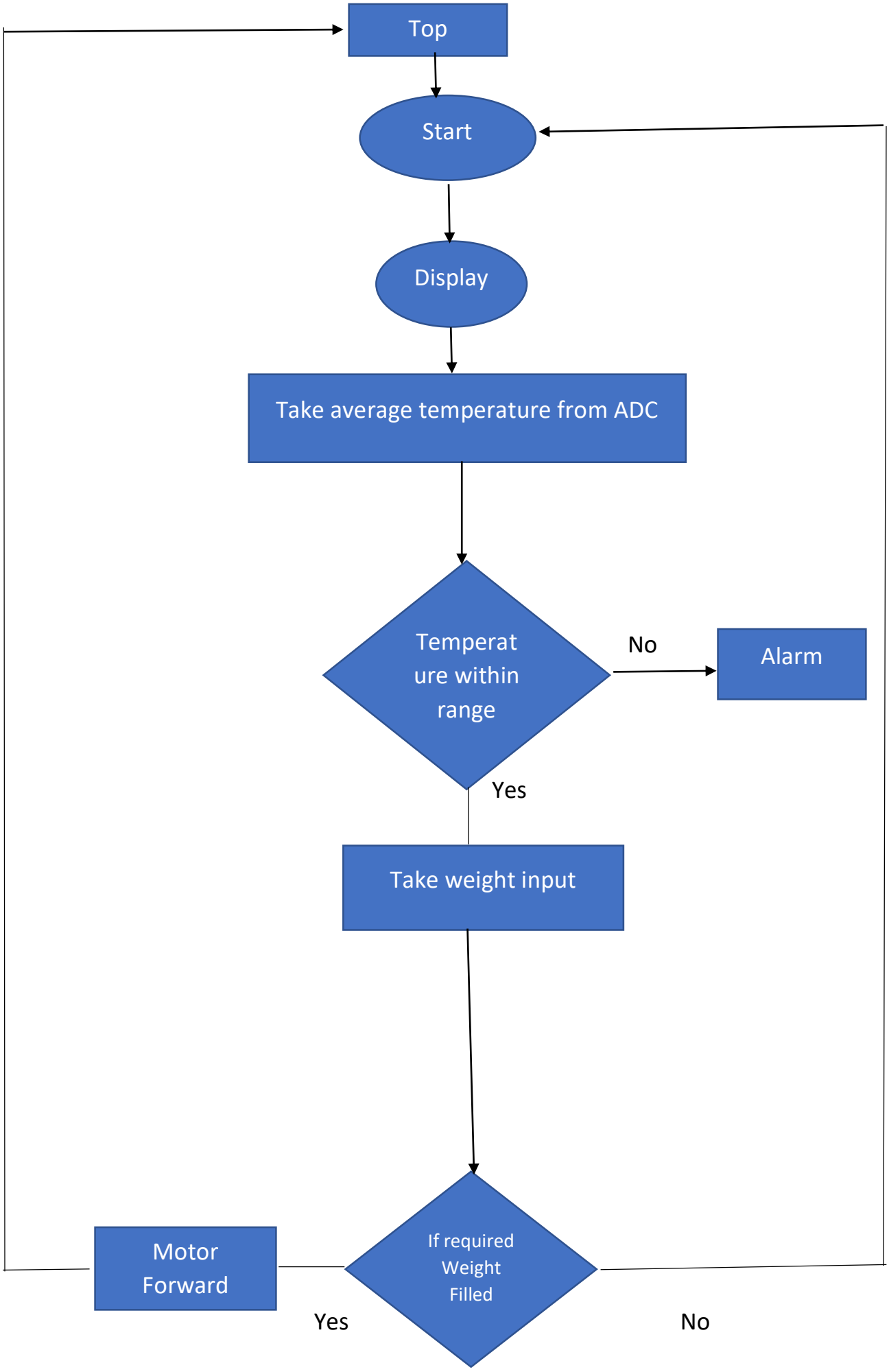
Counter 2: 0Ch

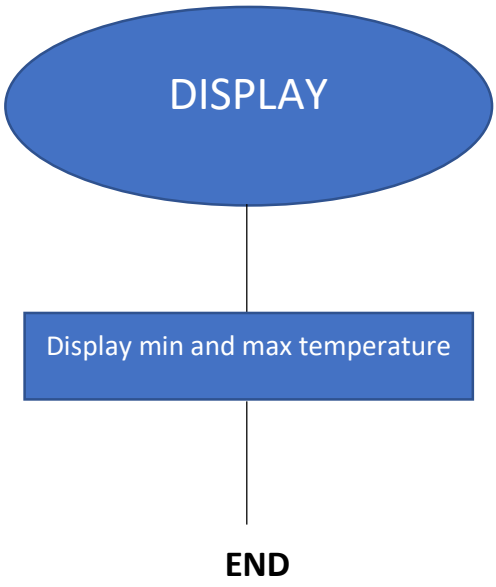
CWR: 0Eh

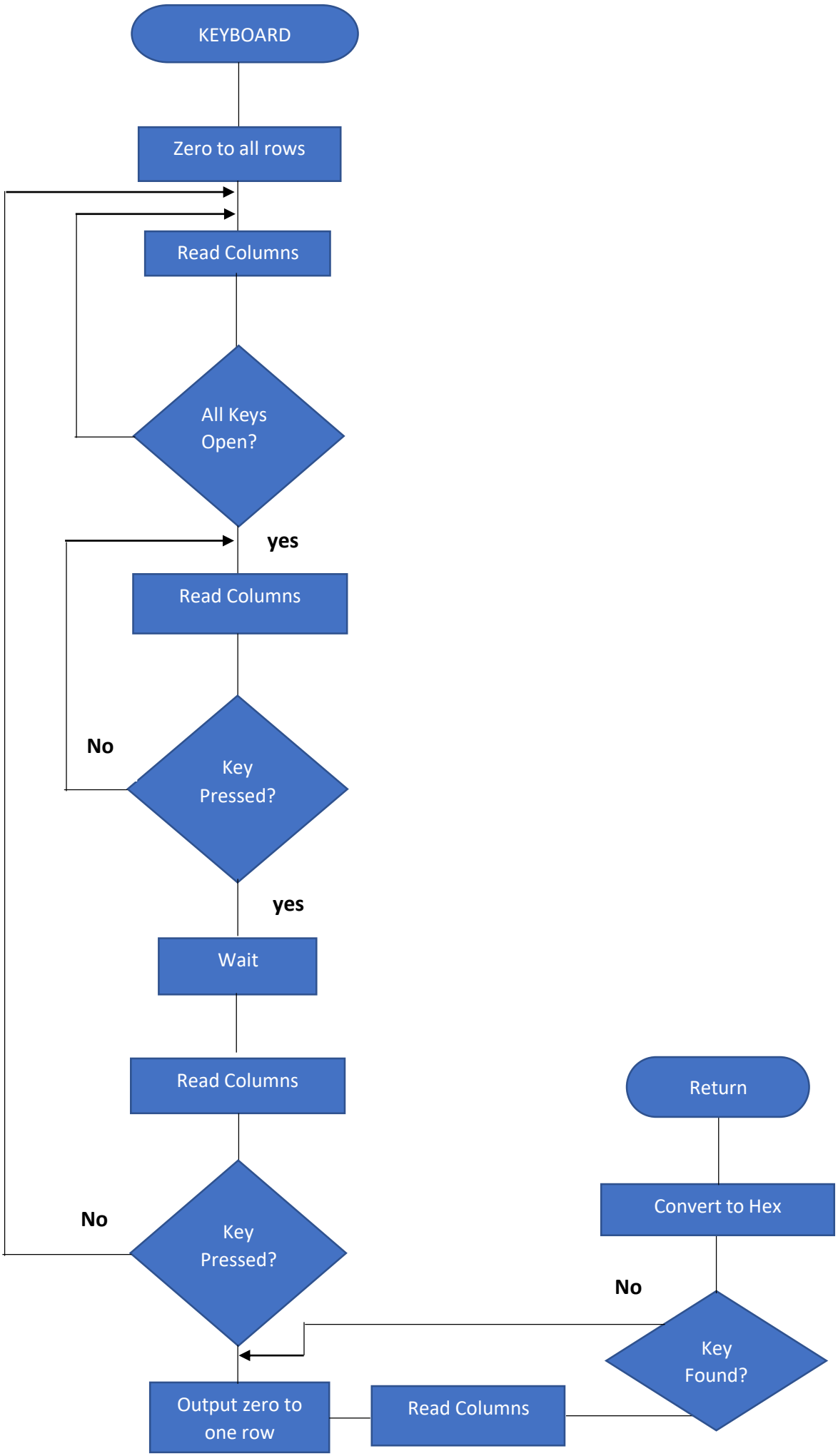
Flow Charts


System Start
Top
Wait for Key Press
Start
Temp Set (Lower)
Temp Set (Upper)
Weight Set

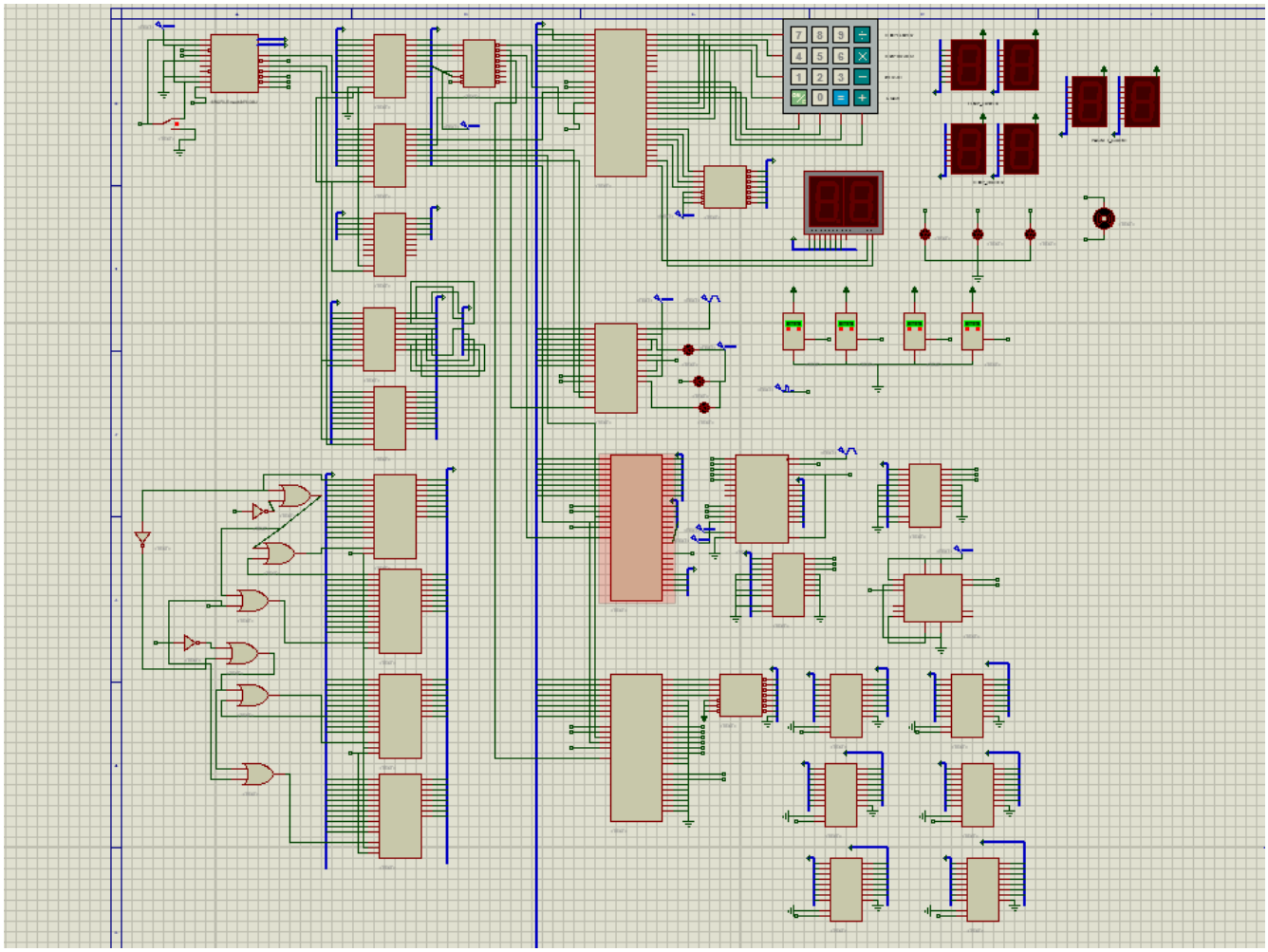

Alarm
Stop flour motor and conveyor motor
Start Alarm
Wait for Acknowledge
Jump to Top
Temp set lower
Take temperature input
Jump to Top
Temp set upper
Take temperature input
Jump to Top
Weight Set
Take Weight input
Jump to Top


Top
Start
Display
Take average temperature from ADC
Temperat ure within range
No
Alarm
Yes
Take weight input
If required Weight Filled
Motor Forward
Yes
No

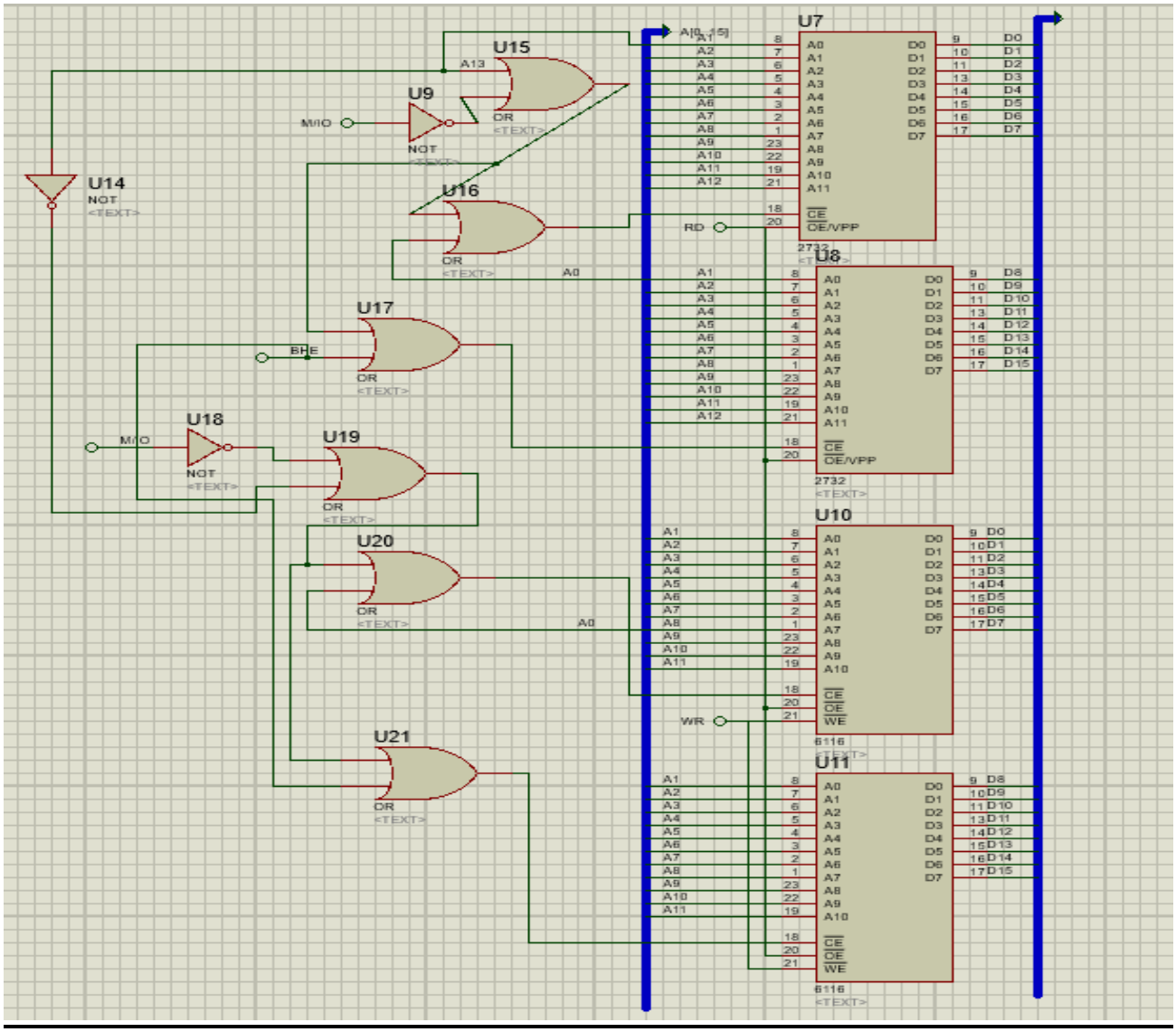

DISPLAY
Display min and max temperature
END


KEYBOARD
Zero to all rows
Read Columns
All Keys Open?
yes
Read Columns
No
Key Pressed?
yes
Wait
Read Columns
No
Key Pressed?
Output zero to one row
Read Columns
Key Found?
No
Convert to Hex
Return

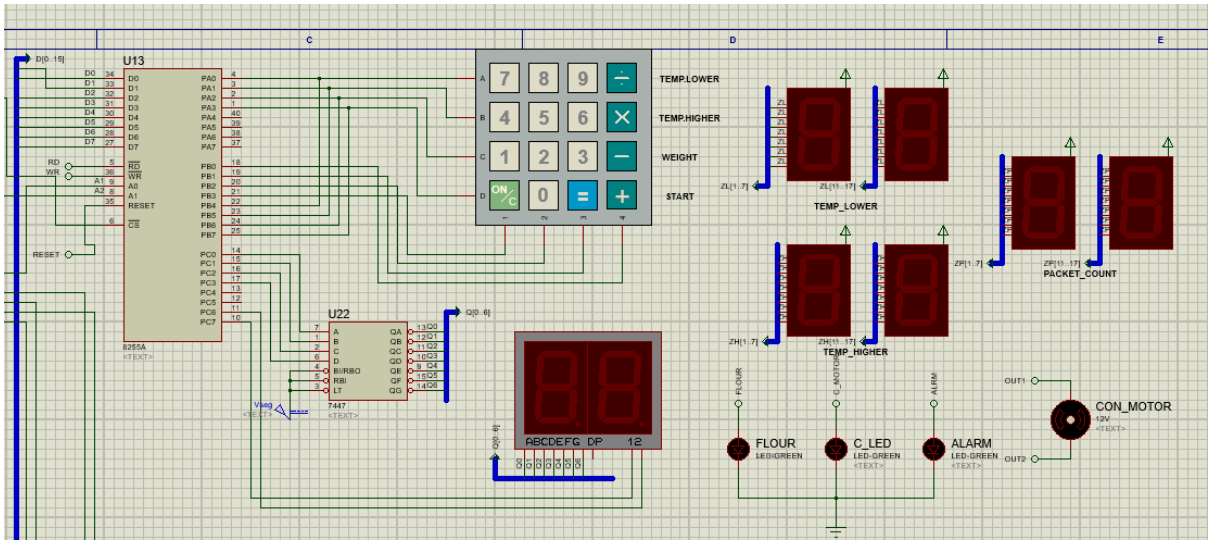
Complete Design



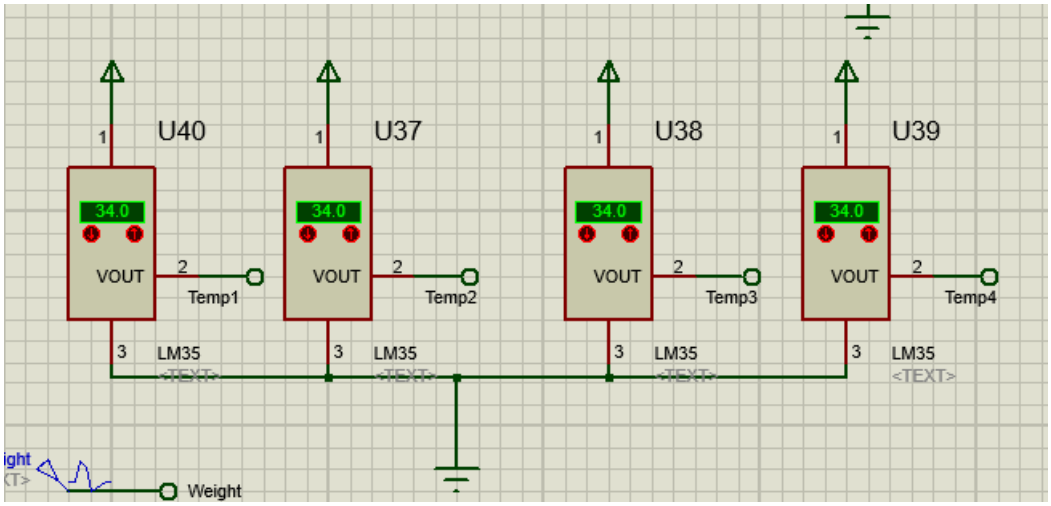
Memory Interfacing



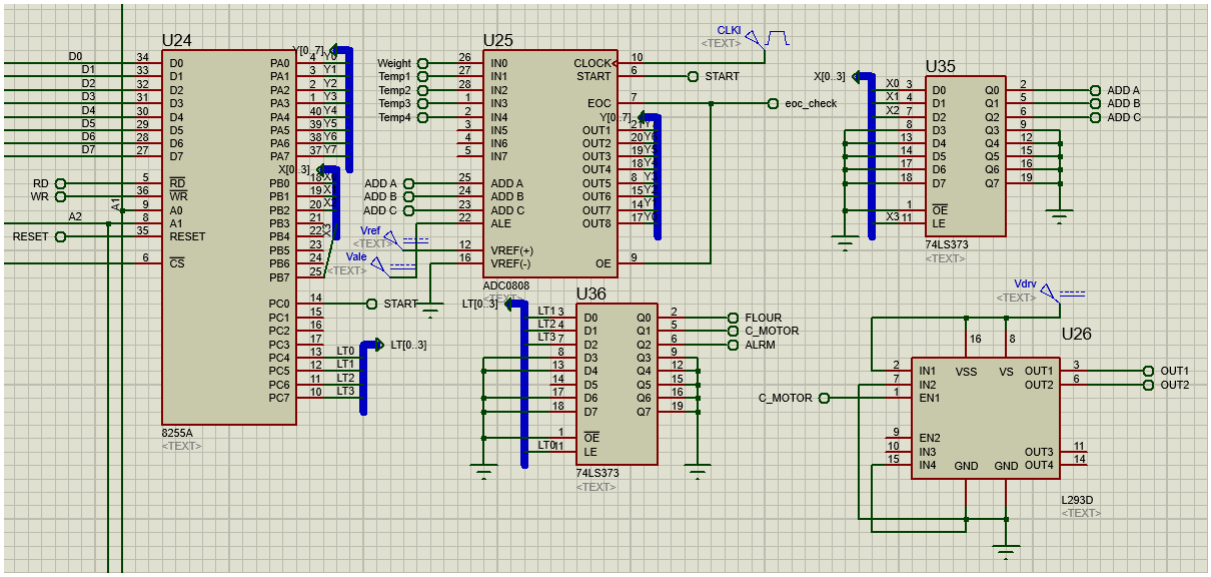
Keypad, Display, Motor and LEDs



Sensors



ADC0808


U24
8255A
U25
ADC0808
U35
74LS373
U36
74LS373
U26
L293D
CLKI
START
eoc_check
Vref
Vale
Vdrv
FLOUR
C_MOTOR
ALRM
OUT1
OUT2
ADD A
ADD B
ADD C
Weight
Temp1
Temp2
Temp3
Temp4
RESET
RD
WR

CODE

#make_bin#

#LOAD_SEGMENT=FFFFh#

#LOAD_OFFSET=0000h#

#CS=0000h#

#IP=0000h#

#DS=0000h#

#ES=0000h#

#SS=0000h#

#SP=FFFEh#

#AX=0000h#

#BX=0000h#

#CX=0000h#

#DX=0000h#

#SI=0000h#

#DI=0000h#

#BP=0000h#

;Code----------

```
        jmp     st1
        db              509 dup(0)


        TABLE           db
    77h,7Bh,7Dh,7Eh,0B7h,0BBh,0BDh,0BEh,0D7h,0DBh,0D
Dh,0DEh,0E7h,0EBh,0EDh,0EEh
        TABLE_D         db
    00h,01h,02h,03h,04h,05h,06h,07h,08h,09h,0AH,0bH,0C
H,0DH,0eH,0FH


        weight1         db  ?
        weight2         db  ?
        weight_mod      db  ?
        temp1           db  ?
        temp2           db  ?
        temp1U          db  ?
```

```
        temp2U      db   ?
        temp_mod    db   ?
        tempU_mod   db   ?
        weight_read db   ?
        temp_read   db   ?
        key_in      db   ?
        counter1 db  ?
        counter2 db  ?
        hr          db   ?
    db              465 dup(0)


;main program-------------------------------------------------------
----------

     st1: cli


;intialize ds, es, ss to start of RAM-------------------------------
-----------

        mov     ax, 0200h
```

```
        mov     ds, ax
        mov     es, ax
        mov     ss, ax
        mov     sp, 0FFFEH


;initializing variables-----------------------------------------------
-------

        mov     tempU_mod, 99
        mov     temp1, 0
        mov     temp2, 2
        mov     temp1U, 9
        mov     temp2U, 9
        mov     weight1, 0
        mov     weight2, 0
        mov     al, 02h
        mov     temp_mod, al
        mov     al, 00h
        mov     weight_mod, al
        mov     counter1, 0
        mov     counter2, 0
```

```
        mov     hr, 0


;8255(keypad) initializing-----------------------------------------------
----------
    ;starting address 00h
    ;port a, port c o/p          port b i/p

        mov     al, 10000010b
        out     06h, al


;8253(timer) initializing-------------------------------------------------
---------
    ;initial clock 10kHz
    ;starting address 08h
    ;counter 0

        mov     al, 00110110b
        out     0Eh, al
        mov     al, 64h     ;count 100      100hz clock
        out     08h, al
```

```
        mov     al, 00h
        out   08h, al

    ;counter 1

        mov     al, 01110110b
        out   0Eh, al
        mov     al, 14h     ;count 20 5Hz clock  ;in
simulation this clock is used If 1 hour clock is need put output
of counter 2 in hr label
        out   0Ah, al
        mov     al, 00h
        out   0Ah, al

    ;counter 2

        mov     al, 10110110b
        out     0Eh, al
        mov     al, 0a0h    ;count 36000   1Hr Timer
        out     0Ch, al
        mov     al, 8ch
        out     0Ch, al
```

```
;8255(ADC) initializing----------------------------------------------
----------
        ;starting address 10h
        ;port a , port c o/p   port b I/p
                mov         al, 10010000b
                out         16h, al



;8255(7 Segment) initihalizing---------------------------------------
--------------
        ;starting address 18h
        ;port a, port b o/p          port c i/p

                mov         al, 10001001b
                out         1eh, al



;initializing over---------------------------------------------------
-------

        top:
```

```
;every motor and led is off
        mov         al, 10h
        out         14h, al
;latch enable off
        mov         al, 00h
        out   14h, al



        call keybrd


        cmp         al, 0EEh
        jz      t1
        cmp al, 0DEh
        jz      t2
        cmp al, 0BEh
        jz      w
        cmp         al, 7Eh
        jz      s

        jmp  top
```

```
;setting minimum temperature----------------------------------------
--------------------

        t1:
                waitstate:
                                in      al, 02h
                                and  al, 0fh
                                cmp al, 0fh
                                jnz waitstate

                mov         al, 41h
                out         04h, al

                call   TempSetL
                jmp  top



;setting maximum temperature----------------------------------------
----------------

        t2:
                waitstate1:
```

```
                    in      al, 02h
                    and al, 0fh
                    cmp al, 0fh
                    jnz waitstate1

        mov         al, 42h
        out   04h, al

        call   TempSetU
        jmp top


;setting weight per packet----------------------------------------------
------------

    w:
        waitstate2:
                    in      al, 02h
                    and  al, 0fh
                    cmp al,0fh
                    jnz waitstate2
```

```
        mov     al, 43h
        out     04h, al

        call   WeightSet
        jmp  top


;On pressing Start-------------------------------------------------------
---------

    s:
        waitstate3:
                    in     al, 02h
                    and al, 0fh
                    cmp al, 0fh
                    jnz waitstate3

        mov     al, 44h
        out   04h, al

        jmp Start
```

```
;Start-------------------------------------------------------------------------
-----

        Start:
                CALL display



;taking temperature inputs from sensors and comparing from
required conditions--------------------

;sensor 1
        ;start flour stop conv stop alarm

                mov     al, 30h
                out     14h, al
                mov     al, 00h ;latch enable off
                out   14h, al


                mov     al, 81h   ;select adc channel 1
                out   12h, al
                mov     al, 00h   ; latch enable off
```

```
        out   12h, al

;high to low transition

        mov         al, 00000001b   ;high
        out   16h, al
        mov         al, 00000000b   ;low
        out   16h, al

        call delay1

;check eoc for high

x123:
        in          al, 1ch
        and  al, 01h
        cmp         al, 00h
        jz      x123

;initialize 8255 again for taking input

        mov         al,     10010000b
```

```
        out   16h, al

    ;after initializing making sure flour motor on

        mov       al, 30h
        out   14h, al
        mov       al, 00h
        out   14h, al

    ;reading input from ADC
        in        al, 10h
    ;store input
        mov       temp_read, al


;sensor 2

        mov       al, 82h   ;select adc channel 2
        out   12h, al
        mov       al, 00h   ; latch enable off
        out   12h, al
```

```
;high to low transition

        mov         al, 00000001b   ;high
        out   16h, al
        mov         al, 00000000b   ;low
        out   16h, al

        call delay1

;check eoc for high

x126:
        in          al, 1ch
        and  al, 01h
        cmp         al, 00h
        jz      x126

;initialize 8255 again for taking input

        mov         al,     10010000b
        out   16h, al
```

```
        ;after initializing making sure flour motor on

                mov     al, 30h
                out   14h, al
                mov     al, 00h
                out   14h, al

        ;reading input from ADC
                in      al, 10h
        ;store input
                add  temp_read, al


;sensor 3----------

                mov     al, 83h   ;select adc channel 3
                out   12h, al
                mov     al, 00h   ; latch enable off
                out   12h, al

        ;high to low transition
```

```
        mov         al, 00000001b   ;high
        out   16h, al
        mov         al, 00000000b   ;low
        out   16h, al

        call delay1

;check eoc for high

x124:
        in          al, 1ch
        and  al, 01h
        cmp         al, 00h
        jz      x124

;initialize 8255 again for taking input

        mov         al,     10010000b
        out   16h, al

;after initializing making sure flour motor on
```

```
        mov     al, 30h
        out   14h, al
        mov     al, 00h
        out   14h, al

    ;reading input from ADC
        in      al, 10h
    ;store input
        add  temp_read, al


;sensor 4

        mov     al, 84h   ;select adc channel 4
        out   12h, al
        mov     al, 00h   ; latch enable off
        out   12h, al

    ;high to low transition

        mov     al, 00000001b   ;high
        out   16h, al
```

```
        mov         al, 00000000b   ;low
        out   16h, al

        call delay1

;check eoc for high

x125:
        in          al, 1ch
        and  al, 01h
        cmp         al, 00h
        jz      x125

;initialize 8255 again for taking input

        mov         al,     10010000b
        out   16h, al

;after initializing making sure flour motor on

        mov         al, 30h
        out   14h, al
```

```
        mov     al, 00h
        out   14h, al

    ;reading input from ADC
        in      al, 10h
    ;store input
        add  temp_read, al


;calculating temperature avg----------------------------------------
-------------

        mov     al, temp_read
        shr   al, 01
        shr   al, 01

        mov     temp_read, al

    ;compare and call alarm procs

        mov     ah, temp_read
        mov     al, temp_mod
```

```
            cmp ah, al
            jb     Alarm              ;jump to alarm if below lower
limit

            mov        al, tempU_mod

            cmp ah, al
            ja  Alarm          ;jump to alarm if above upper limit

            jmp Mf



;taking weight input from sensor and comparing----------------
--------------------

            mov        al, 80h   ; select adc channel 0
            out   12h, al

            mov        al, 00h   ; latch enable off
            out   12h, al
```

```
;high to low transition

        mov     al, 00000001b  ;high
        out   16h, al
        mov     al, 00000000b   ;low
        out   16h, al

        call   delay1

;check for high eoc

x1:
        in      al, 1ch
        and     al, 01h
        cmp     al, 00h
        jz     x1

;initialize again for reading input

        mov     al,    10010000b
        out   16h,al
```

```
;start flour stop conv stop alarm

        mov         al, 30h
        out   14h, al
        mov         al, 00h      ; latch enable off
        out   14h, al

;read input
        in          al, 10h
;store input
        mov         weight_read, al


;compare weight

        mov         al, weight_mod
        mov         ah, weight_read

        cmp ah, al
        jae  Mf
```

```
;Alarm on code-------------------------------------------------------
----------

        Alarm:
                mov         al, 90h     ;start alarm
                out   14h, al

                mov         al, 00h     ;latch enable off
                out   14h, al

                jmp Ack



;Acknowledging alarm-----------------------------------------------
-------------

        Ack:
                mov         al, 00h
                call   keybrd
```

```
waitstate7:
        in          al, 02h
        and  al, 0fh
        cmp al, 0fh
        jnz    waitstate7

        mov         al, 10h     ;stop alarm
        out   14h, al

        mov         al, 00h     ;latch enable off
        out   14h, al

        jmp top

;Rotating Motor-------------------------------------------------------
----------

        Mf:
        ;stop flour and start conveyer

        mov         al, 50h
```

```
        out   14h, al

        mov         al, 00h     ;latch enable off
        out   14h, al

        ;call delay
        call delay

    ;stop conveyer and start flour

        mov         al, 30h
        out   14h, al

        mov         al, 00h     ;latch enable off
        out   14h, al


        jmp top


;-------------------------------------------------------------------------
---
```

```
;procedures-----------------------------------------------------------
---------

;procedure for short delay---------------------------------------------
-------------

delay1 proc near
        mov     cx, 300
    nxt11:
        LOOP    nxt11
        ret
delay1 endp


;procedure for long delay----------------------------------------------
------------

delay proc near
        mov     cx, 1000
    x2:
        nop
        nop
        loop x2
```

```
        ret
delay endp


;procedure for weightset----------------------------------------------
------------

WeightSet proc near
        PUSHF
        PUSH BX
        PUSH CX
        PUSH DX

        call   keybrd
        call   key_press

        mov        weight1, al
        mov        weight_mod, al

    ;display tens place value

        mov        weight1, al
```

```
        or          al, 80h
        out   04h, al

;mul tens place val by 10

        mov         cx, 09h
        mov         al, weight_mod
        mov         bl, weight_mod
wt_loop:
        add  al, bl
        loop        wt_loop

        mov         weight_mod, al

;wait for key release

waitstate4:
        in          al, 02h
        and  al, 0fh
        cmp al, 0fh
        jnz waitstate4
```

```
;wait for key press

        call    keybrd
        call    key_press

        mov         weight2, al
        add  weight_mod, al

;display units place value

        mov         al, weight2
        or          al, 40h
        out   04h, al

        mov         al, weight_mod
        mov         ah, 00h

        POP DX
        POP CX
        POP BX
        POPF
```

```
        ret
WeightSet endp


;procedure for tempset minimum----------------------------------
-------------------

TempSetL proc near

        PUSHF
        PUSH BX
        PUSH CX
        PUSH DX

        mov     al, 00h
        out   04h, al

        call   keybrd
        call   key_press
        mov     temp1,al
        mov     temp_mod, al
```

```
        mov         al, temp1

;display tens place

        or          al, 80h
        out   04h, al

;mul tens place val by 10

        mov         cx, 09h
        mov         bl, temp_mod
        mov         al, temp_mod
add_loop:
        add  al, bl
        loop        add_loop

        mov         temp_mod, al

;wait for key release

waitstate5:
        in          al, 02h
```

```
        and  al, 0fh
        cmp al, 0fh
        jnz    waitstate5

        call   keybrd
        call   key_press

        mov        temp2, al
        add  temp_mod, al

        mov        al, temp_mod
        mov        ah, temp2

        mov        al, temp2

;display units place

        or         al, 40h
        out   04h, al

        mov        al, temp_mod
        mov        ah, 00h
```

```
        POP DX
        POP CX
        POP BX
        POPF

        ret
TempSetL endp

;procedure working for tempset maximum----------------------
---------------------

TempSetU proc near
        PUSHF
        PUSH BX
        PUSH CX
        PUSH DX

    ;wait for key press

        call   keybrd
```

```
        call   key_press

;store values

        mov         temp1U, al
        mov         tempU_mod, al

;display tens place

        or          al, 80h
        out   04h, al

;mul tens place val by 10

        mov         cx, 09
        mov         al, tempU_mod
        mov         bl, tempU_mod
add_loopU:
        add  al, bl
        loop        add_loopU

        mov tempU_mod, al
```

```
;wait for key release

waitstate6:
        in          al, 02h
        and  al, 0fh
        cmp al, 0fh
        jnz waitstate6

;wait for key press

        call   keybrd
        call   key_press

        mov        temp2U, al
        add  tempU_mod, al

        mov        temp2U, al

        ;display units place

        or          al, 40h
```

```
        out   04h, al

        mov       al, tempU_mod
        mov       ah, 00

        POP DX
        POP CX
        POP BX
        POPF

        ret
TempSetU endp

;procedure for keyboard----------------------------------------------
-------------

keybrd proc near

        PUSHF
        PUSH BX
        PUSH CX
```

```
        PUSH DX

;send 0's to all rows

        mov         al, 00
        mov         dx, 00h
        out   dx, al



;read columns

        mov         dx, 02h             ;load input port address

wait_open:
        in          al, dx
        and  al, 0fh
        cmp al, 0fh
        jne wait_open



;read colunms to see if key is pressed
```

```
wait_press:
        in              al, dx
        and  al, 0fh
        cmp al, 0fh
        je wait_press


;debounce

        mov         cx, 0127h               ;2.5 ms
delay123:
        loop        delay123


;read columns to see if key still pressed

        in              al, dx
        and  al, 0fh
        cmp al, 0fh
        je wait_press
```

```
;find key

        mov         al, 0feh
        mov         cl, al

next_row:
        mov         dx, 00h
        out   dx, al
        mov         dx, 02h
        in          al, dx
        and  al, 0fh
        cmp al, 0fh
        jne encode
        rol    cl, 01
        mov         al, cl
        jmp next_row

encode:
        mov         bx, 000fh
        in          al, dx

try_next :
```

```
        cmp al, table[bx]
        je done
        dec bx
        jns try_next
        mov     ah, 01h
        jmp exit

    done:
        mov     al, bl
        mov     key_in, al
        mov     ah, 00h

    exit:
        POP DX
        POP CX
        POP BX
        POPF

        ret

keybrd endp
```

```
;procedure for decoding the pressed key----------------------------
-----------------

key_press proc near

        PUSHF
        PUSH BX
        PUSH CX
        PUSH DX


    x21:
        cmp     al, 7Bh
        jnz x5
        mov     al, 00h
        jmp x0

    x5:
        cmp     al, 0B7h
        jnz x6
        mov     al, 01h
```

```
        jmp x0

x6:
        cmp     al, 0BBh
        jnz x7
        mov     al, 02h
        jmp x0

x7:
        cmp al, 0BDh
        jnz x9
        mov     al, 03h
        jmp x0

x9:
        cmp al, 0D7h
        jnz xA
        mov     al, 04h
        jmp x0

xA:
        cmp     al, 0DBh
```

```
        jnz xB
        mov         al, 05h
        jmp x0

xB:
        cmp al, 0DDh
        jnz xD
        mov         al, 06h
        jmp x0

xD:
        cmp al, 0E7h
        jnz xE
        mov         al, 07h
        jmp x0

xE:
        cmp al, 0EBh
        jnz xF
        mov         al, 08h
        jmp x0
```

```
        xF:
                cmp al, 0EDh
                jnz x0
                mov         al, 09h
                jmp x0

        x0:
                nop

                POP DX
                POP CX
                POP BX
                POPF

                ret
key_press endp


;procedure for displaying the values on 7 Segment----------
--------------------

display proc near
```

```
pushf
push bx
push cx
push dx

mov         al, 00h
out   1ah, al

mov         al, 01h
out   1ah, al
mov         al, temp1
out   18h, al

mov         al, 02h
out   1ah, al
mov         al, temp2
out   18h, al

mov         al, 04h
out   1ah, al
mov         al, temp1U
out   18h, al
```

```
        mov     al, 08h
        out   1ah, al
        mov     al, temp2U
        out   18h, al


;checking for hr timer then only display

        in      al, 1ch
        and  al, 02h
        cmp al, hr
        je skipc

        mov     bl, hr

        cmp bl, 0
        jne one

        mov     hr, 02h
        jmp ahead
```

```
one:
        mov         hr, 0

ahead:

        mov         al, 10h
        out   1ah, al
        mov         al, counter2
        out   18h, al

        mov         al, 20h
        out   1ah, al
        mov         al, counter1
        out   18h, al

skipc:
        mov         al, 00h
        out   1ah, al

        pop dx
        pop cx
        pop bx
```

```
        popf


        ret
display endp
;
-------------------------------------------------------------------------
```

REFERENCES

ADC0808

http://html.alldatasheet.com/html- pdf/8097/NSC/ADC0808/38/1/ADC0808.html

Load Cell

http://www.velamed.com/englisch/products/electromyography/noraxon - sensors/noraxon-sensors.html

LS138

http://www.alldatasheet.com/datasheet-pdf/pdf/46206/SLS/LS138.html

74373

http://www.alldatasheet.com/datasheet-pdf/pdf/192081/TI/LS373.html

74245

http://www.alldatasheet.com/datasheet- pdf/pdf/44472/SIEMENS/BF245.html

L293D

http://www.ti.com/lit/ds/symlink/l293d.pdf

LM 35

http://www.ti.com/lit/ds/symlink/lm35.pdf