Microprocessors and Interfacing

Flour Packing Machine

Question No: 13

Group: 84

Submitted By:

Sarkar Bharat A Waresh (2017A7PS0087P)

Vaishnavi Kotturu (2017A7PS0088P)

Saurav Virmani (2017A7PS0090P)

Vineet Golchha Jain (2017A7PS0092P)

Submitted To:

Assistant Prof. Nitin Chaturvedi

An assignment submitted in partial fulfilment of the course requirements for

Microprocessor Programming and Interfacing

BITS Pilani, Pilani Campus

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 flour 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.

Assumptions:

- User inputs the temperature in ° C
- The temperature where the packing is going on is adjusted manually in range : 0° C to 250° C
- Minimum temperature entered is 0° C
- Maximum temperature entered is 99° C
- All user inputs should be whole numbers.
- Rate of flow of flour varies linearly with the time for which the valve is kept open.
- Assumptions for keypad :
    - User presses '/' key on the keypad to set the minimum temperature.
    - User presses 'x' key on the keypad to set the maximum temperature.
    - User presses '-' key on the keypad to set the desired weight.
    - User presses '+' key on the keypad to start the process of filling the packets.
    - User presses '=' key on the keypad to reset the minimum and maximum temperature values entered. It also disables the alarm (if ringing).
    - User presses 'ON/C' key on the keypad to Enter.

Components Used

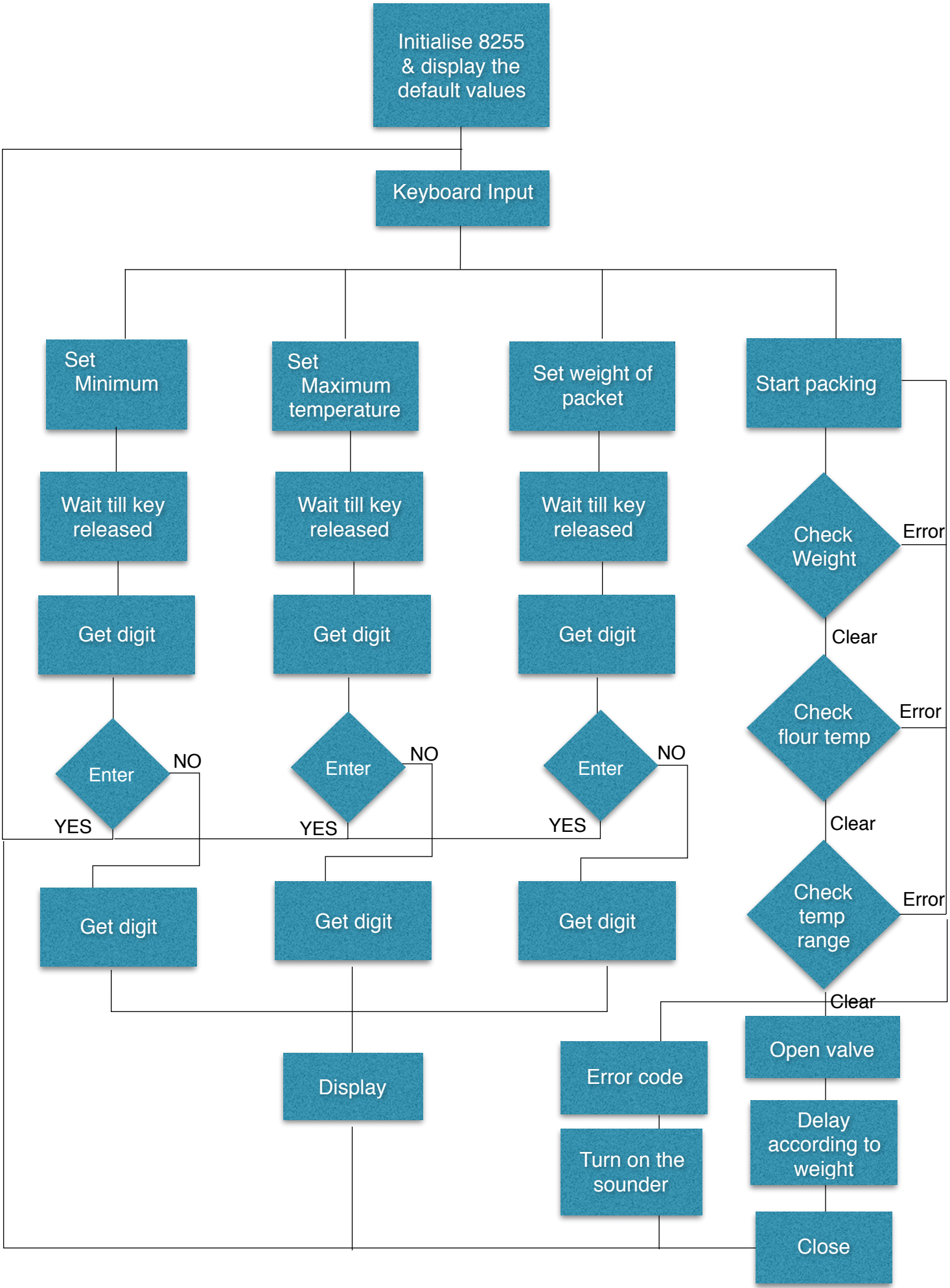
S. No.	Components Used	Quantity	Purpose
1	7-Segment common cathode Display	4	To display minimum & maximum temperature
2	Keypad - small Calc	1	To take user input for weight, minimum & maximum temperature and to start.
3	ADC0808	1	To convert Analog signal by LM35 Temperature sensor to digital.
4	LM35 (Precision Centigrade Temperature Sensor)	1	To sense the temperature of the flour packed.
5	8255A (Programmable Peripheral Interface)	4	To interface ADC0808, keypad, sounder, stepper motor and 7-segment display.
6	74LS245 (Octal Bus Tri-state Transceivers)	2	To amplify the data received and sent.
7	74LS138 (3:8 decoder)	1	To decode the address so as to select different memory chips and input & output devices.
8	8086 Microprocessor	1	To run the code stored in the memory chips.
9	74LS373 (Octal latches with 3 state output)	3	To latch the address generated for the entire machine cycle.
10	2732- 32K (4K*8) EPROM	2	To store the code.
11	6116- 16K (2K*8) Static RAM	2	To store the values of the variables used in the code.
12	Sounder (Piezo Sounder model outputs)	1	Produces sound when : minimum temperature> maximum temperature OR weight != 5 or 10 or 20kgs. OR Temperature of flour out of range. OR Wrong key entered.
13	Motor-Stepper (Unipolar stepper motor)	1	Used to open and close the valve & while flour packing.
14	2-input OR Gate	10	Helps in decoding memory addressing and M/IO’
15	NOT Gate	2	Helps in decoding memory addressing and M/IO’

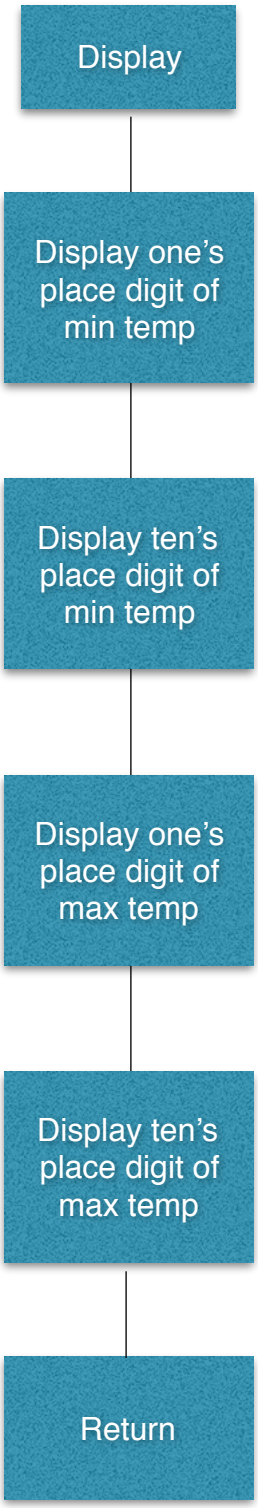
Memory and I/O Interfacing

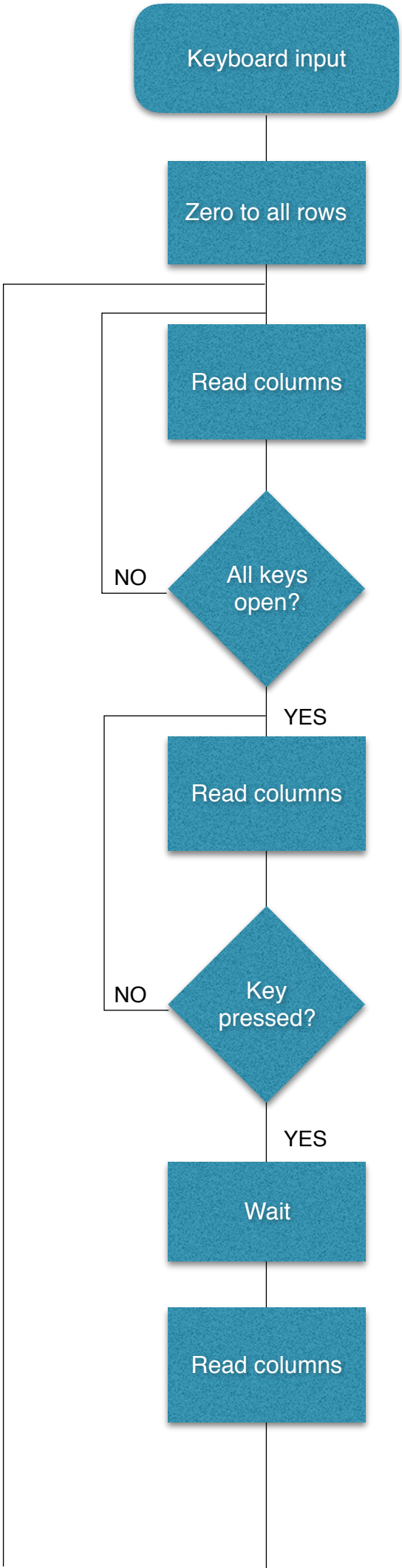
	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	BHE'	M/IO'
EVEN RAM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1
ODD RAM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1
EVEN ROM	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
ODD ROM	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
CS1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
CS2	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
CS3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
CS4	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

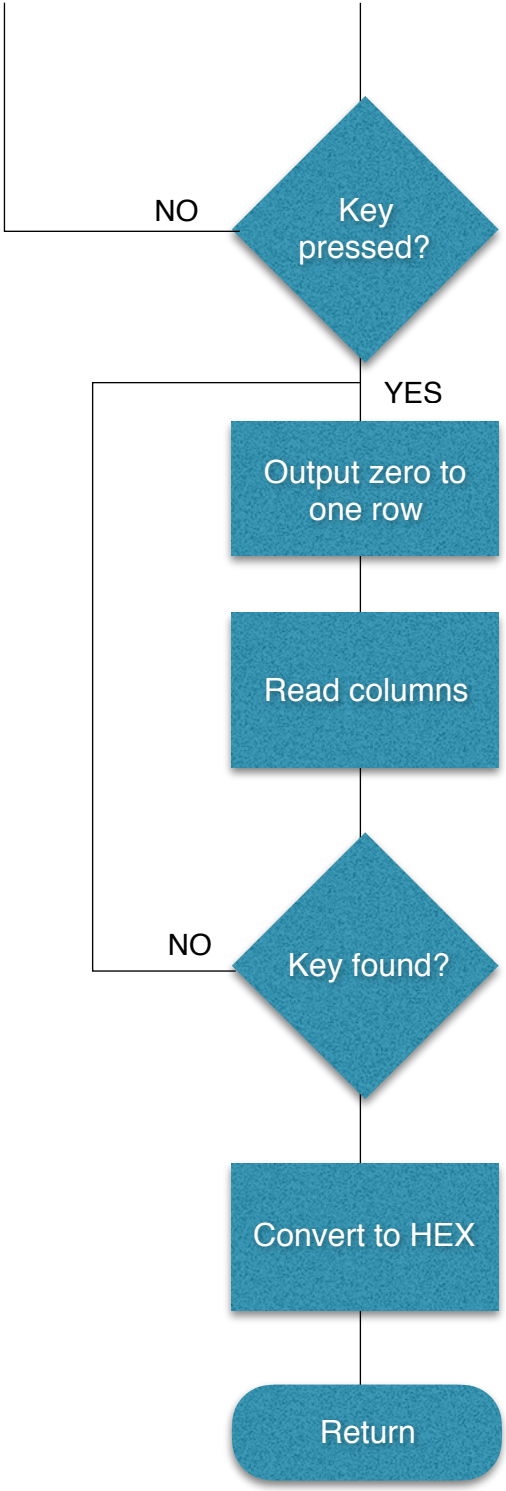
ITEM	ADDRESS	REMARK
MEMORY		
RAM	00000H-00FFFH	Even and Odd RAM depending on A0 and BHE’ 4K->2K*2
ROM	02000H-03FFFH	Even and Odd ROM depending on A0 and BHE’ 8K->4K*2
8255(#1)		Connects 7-segment display with minimum temperature
PORT A	4000H	
PORT B	4002H	
PORT C	4004H	
CONTROL WORD REGISTER	4006H	
8255(#2)		Connects 7-segment display with maximum temperature
PORT A	6000H	
PORT B	6002H	
PORT C	6004H	
CONTROL WORD REGISTER	6006H	
8255(#3)		Connects keypad, sounder and stepper motor
PORT A	8000H	
PORT B	8002H	
PORT C	8004H	
CONTROL WORD REGISTER	8006H	
8255(#4)		Interfaces ADC0808
PORT A	A000H	
PORT B	A002H	
PORT C	A004H	
CONTROL WORD REGISTER	A006H	

FLOW CHART


Initialise 8255 & display the default values
Keyboard Input
Set Minimum
Set Maximum temperature
Set weight of packet
Start packing
Wait till key released
Wait till key released
Wait till key released
Check Weight
Error
Clear
Get digit
Get digit
Get digit
Check flour temp
Error
Clear
Enter
NO
YES
Enter
NO
YES
Enter
NO
YES
Check temp range
Error
Clear
Get digit
Get digit
Get digit
Open valve
Display
Error code
Delay according to weight
Turn on the sounder
Close


Display
Display one's place digit of min temp
Display ten's place digit of min temp
Display one's place digit of max temp
Display ten's place digit of max temp
Return


Keyboard input
Zero to all rows
Read columns
All keys open?
NO
YES
Read columns
Key pressed?
NO
YES
Wait
Read columns


NO
Key pressed?
YES
Output zero to one row
Read columns
NO
Key found?
Convert to HEX
Return

ALP :

```
.model small
.data
    org 0000h
    sevtab db 3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh
    keytab db 77h,7Bh,7Dh,7Eh,0B7h,0BBh,0BDh,0BEh,0D7h,0DBh,0DDh,
0DEh,0E7h,0EBh,0EDh,0EEh
    total db 100
    weight db 05h
    temp db ?
    tempL db 00h
    tempU db 5Ah
    minTL db 00h
    minTU db 00h
    maxTL db 00h
    maxTU db 09h
    keyin db 0
.code
.startup
;main program
    st1:
     cli
;initialize 8255 7 segment display
     mov dx,4006h
     mov al,10000000b
```

```
        out dx,al

;initializze 8255 7 segment display
        mov dx,6006h
        mov al,10000000b
        out dx,al

;initialize 8255 with stepper motor, buzzer, keypad
        mov dx,8006h
        mov al,10000010b
        out dx,al

;initialize 8255 with ADC
        mov dx,0A006h
        mov al,10011000b
        out dx,al

        call display
;initializing over
top:
        call keyboard

        cmp al,0E7h;  setting min temp
        jz t1
```

```
      cmp al,0D7h; setting max temp
      jz t2

      cmp al,0B7h ; setting weight
      jz w

      cmp al,77h ; checking start
      jz st

      cmp al,7Bh  ; checking if reset is pressed
      jnz k:
      call reset

     k:
      jmp top

;setting minimum temp.....

      t1:
        waitstate0:              ; waiting until the key is released
            mov dx,8002h
            in al,dx
            and al,0fh
```

```
        cmp al,0fh
        jnz waitstate0

      call TempSetL
      jmp top

;setting maximum temperature .....

    t2:
      waitstate1:          ;waiting until the key is released
        mov dx,8002h
        in al,dx
        and al,0fh
        cmp al,0fh
        jnz waitstate1

      call TempSetU
      jmp top

;setting weight per packet .....
     w:
      waitstate2:          ;waiting until the key is released
        mov dx,8002h
        in al,dx
```

```
            and al,0fh
            cmp al,0fh
            jnz waitstate2

         call WeightSet
         jmp top

;on pressing start

        st:
         waitstate3:         ;waiting until the key is released
            mov dx,8002h
            in al,dx
            and al,0fh
            cmp al,0fh
            jnz waitstate3

         jmp Start

;Start .....................

        Start:
            CALL display
```

```
call checkWRem      ; to check whether total flour > 0
cmp al,0FFh
jz top

call checkWeight    ; checking whether the weight entered is correct
cmp al,0FFh
jz top

call checkTempRange ;checking whether the maximum temp is greater than
min temp
cmp al,0FFh
jz top

call checkTemp      ;checking whether the flour temp is in the range
cmp al,0FFh
jz top
                    ;rotating the stepper motor
mov dx,8004h
mov al,10h
out dx,al

;call delayR
;call delayR
    mov       cx, 0ffffh
    delayOen:
```

```
            loop        delayOen



mov dx,8004h
mov al,20h
out dx,al

mov bl,weight

xt:                 ;delay according to the weight
   mov cx,0ffffh
    delayOpen:
            loop        delayOpen
            dec bl
            jnz xt

    mov al,10h              ;rotating stepper motor in opposite direction
    out dx,al

mov cx,0ffffh
delayOpn:
        loop    delayOpn

    mov al,80h
```

```
        out dx,al

        mov al,weight     ; subtracting from the total weight
        sub total,al

        jmp top

checkWRem proc near       ; checking the total weight remaining

    mov al,weight
    cmp al,total
    jle wre

    call errorCode
    mov al,0FFh

    wre:
    ret
checkWRem endp

checkWeight proc near             ;procedure to checkthe weight

    cmp weight,05h
    jz we
```

```
        cmp weight,0Ah
        jz we

        cmp weight,14h
        jz we

        call errorCode
        mov al,0FFh

        we:
        ret

checkWeight endp

checkTempRange proc near        ; procedure to check min and max temp

        mov al,maxTU
        cmp minTU,al
        jl te

        mov al,maxTL
        cmp minTL,al
        jl te
```

```
        call errorCode
        mov al,0FFh

        te:
        ret

checkTempRange endp

reset proc near                 ; reset all the values to initial value
    pushf
    push dx

    mov tempL,00h
    mov tempU,5Ah
    mov minTL,00h
    mov minTU,00h
    mov maxTL,00h
    mov maxTU,09h
    mov total,100
    mov weight,05h

    mov dx,8004h
    mov al,00h
```

```
    out dx,al

    call display

    pop dx
    popf

    ret

reset endp

checkTemp proc near          ;checking temp of flour in range
    pushf
    push dx
    push cx
    push bx

    mov al,02h
    mov dx,0A002h
    out dx,al

    mov al,00h
    mov dx,0A004h
```

```
out dx,al

mov al,01h
out dx,al

mov al,00h
out dx,al

        mov   cx, 0FFFFh                ;2.5 ms
  delayEOC:
        loop   delayEOC

wait:
   in al,dx
   rcl al,1
   jnc wait
mov dx,0A000h

in al,dx; getting the temperature input

sub al,74h ;116 stands for 0 degree celsius and 137 for 250
mov temp,al

mov cx,09h
```

```
TsLoop:
    add temp,al
    loop TsLoop

mov al,temp

cmp al,tempL
jl err
cmp al,tempU
jg err
jmp ok

err:
    call errorCode        ;error if out of range
    mov al,0FFh

ok:
    pop bx
    pop cx
    pop dx
    popf

ret
```

```
checkTemp endp

;procedure for setting weight
WeightSet proc near
    pushf
    push bx
    push cx
    push dx

    call keyboard
    call keyPress
    mov weight,al
    mov bl,al
    mov dx,8002h

    waitstate5:
        in al,dx
        and al,0fh
        cmp al,0fh
        jnz waitstate5

    call keyboard
    cmp al,7Eh
```

```
jz wr

call keyPress

mov cx,09h
wLoop:
    add weight,bl
    loop wLoop

add weight,al

waitstate4:
    in al,dx
    and al,0fh
    cmp al,0fh
    jnz waitstate4


wr:
    pop dx
    pop cx
    pop bx
    popf
```

```
    ret

WeightSet endp

;procedure for setting min temp

TempSetL proc near

    pushf
    push bx
    push cx
    push dx

    call keyboard
    call keyPress
    mov minTU,00h
    mov minTL,al
    mov tempL,al
    mov bl,al
    mov dx,8002h

    waitstate6:
        in al,dx
        and al,0fh
```

```
    cmp al,0fh
    jnz waitstate6



call keyboard
cmp al,7Eh
jz wr1

call keyPress

mov cx,09h
tLLoop:
    add tempL,bl
    loop tLLoop

add tempL,al

mov minTU,bl

mov minTL,al

waitstate7:
    in al,dx
    and al,0fh
```

```
        cmp al,0fh
        jnz waitstate7


    wr1:
        call display
        pop dx
        pop cx
        pop bx
        popf

    ret

TempSetL endp

;procedure for settign max temp

TempSetU proc near

    pushf
    push bx
    push cx
    push dx
```

```
call keyboard
call keyPress
mov maxTU,00h
mov maxTL,al
mov tempU,al
mov bl,al
mov dx,8002h

waitstate9:
    in al,dx
    and al,0fh
    cmp al,0fh
    jnz waitstate9


call keyboard
cmp al,7Eh
jz wr2

call keyPress

mov cx,09h
tULoop:
    add tempU,bl
```

```
    loop tULoop

add tempU,al

mov maxTU,bl

mov maxTL,al

waitstate8:
    in al,dx
    and al,0fh
    cmp al,0fh
    jnz waitstate8


wr2:
    call display
    pop dx
    pop cx
    pop bx
    popf

ret
```

```
TempSetU endp

;procedure for display of temperature
display proc near
    pushf
    push bx
    push cx
    push dx

    mov dx,4000h
    mov bh,00h
    mov bl,minTU
    mov al,sevtab[bx]
    out dx,al

    mov dx,4002h
    mov bh,00h
    mov bl,minTL
    mov al,sevtab[bx]
    out dx,al

    mov dx,6000h
    mov bh,00h
```

```
        mov bl,maxTU
        mov al,sevtab[bx]
        out dx,al

        mov dx,6002h
        mov bh,00h
        mov bl,maxTL
        mov al,sevtab[bx]
        out dx,al

        pop dx
        pop cx
        pop bx
        popf

        ret

display endp

;procedure for keyboard

keyboard proc near

                PUSHF
```

```
        PUSH BX
        PUSH CX
        PUSH DX

;send 0's to all rows

        mov   al, 0f0h
        mov   dx, 8000h
        out     dx, al


;read columns

        mov   dx, 8002h             ;load input port address

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


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

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



;debounce

        mov     cx, 0027h           ;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, 8000h
        out    dx, al
        mov  dx, 8002h
        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, keytab[bx]
        je done
        dec bx
        jnz try_next
```

```
        mov   ah, 01h
        jmp exit

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

    exit:
        POP DX
        POP CX
        POP BX
        POPF

        ret

keyboard endp

keyPress proc near

    pushf
    push bx
    push cx
```

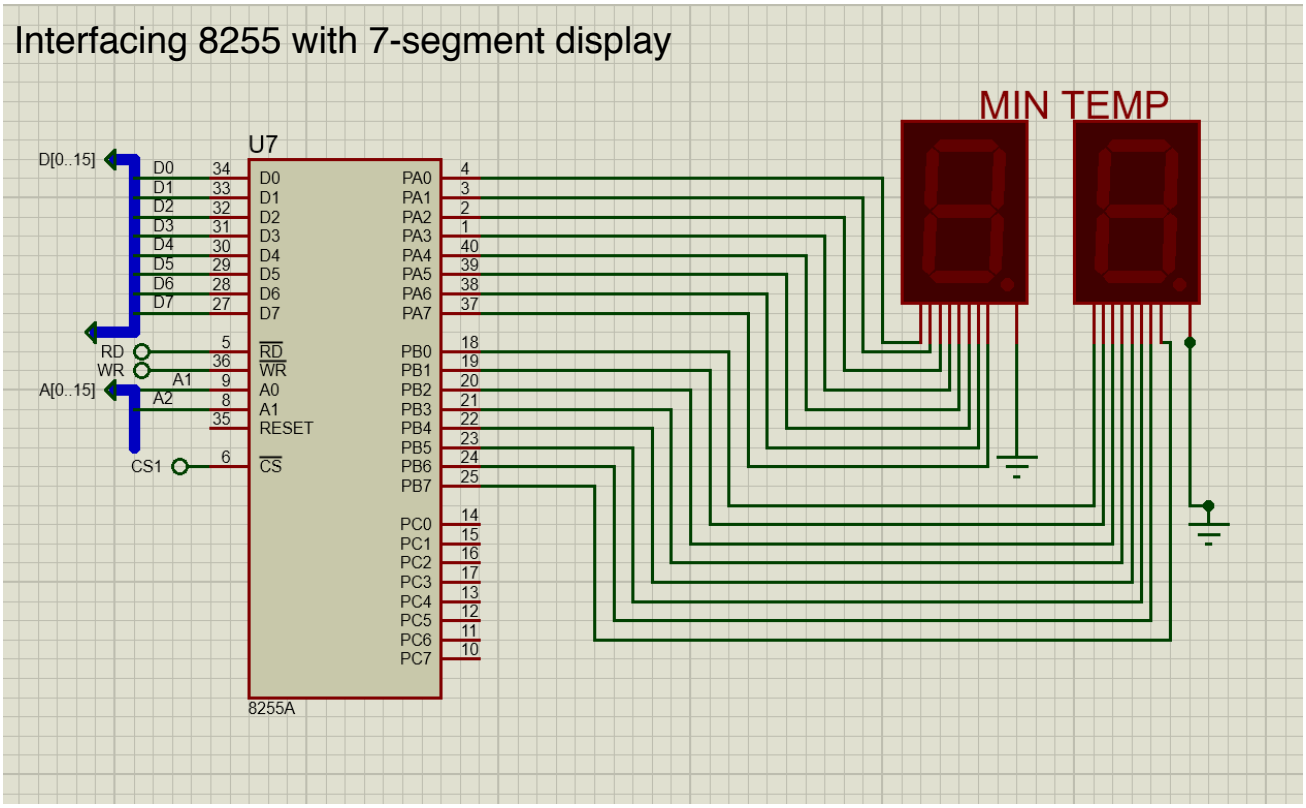
```
    push dx
x1:
    cmp al,0BEh
    jnz x2
    mov al,01h
    jmp r1

x2:
    cmp al,0BDh
    jnz x3
    mov al,02h
    jmp r1

x3:
    cmp al,0BBh
    jnz x4
    mov al,03h
    jmp r1

x4:
    cmp al,0DEh
    jnz x5
    mov al,04h
    jmp r1
```

```
x5:
    cmp al,0DDh
    jnz x6
    mov al,05h
    jmp r1

x6:
    cmp al,0DBh
    jnz x7
    mov al,06h
    jmp r1

x7:
    cmp al,0EEh
    jnz x8
    mov al,07h
    jmp r1

x8:
    cmp al,0EDh
    jnz x9
    mov al,08h
    jmp r1
```

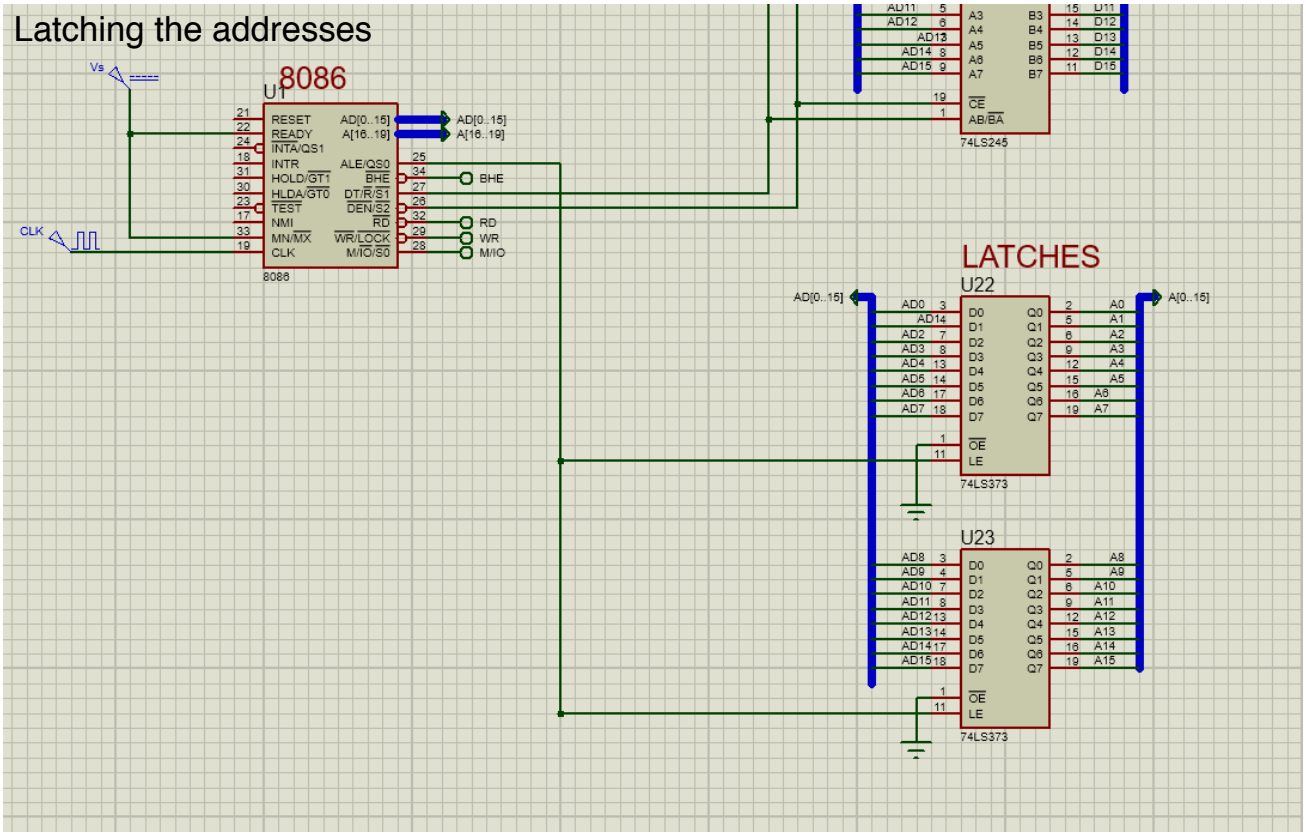
```
x9:
    cmp al,0EBh
    jnz x0
    mov al,09h
    jmp r1

x0:
    cmp al,7Dh
    jnz xerr
    mov al,00h
    jmp r1

xerr:
    call errorCode

r1:
    nop

    pop dx
    pop cx
    pop bx
    popf
```

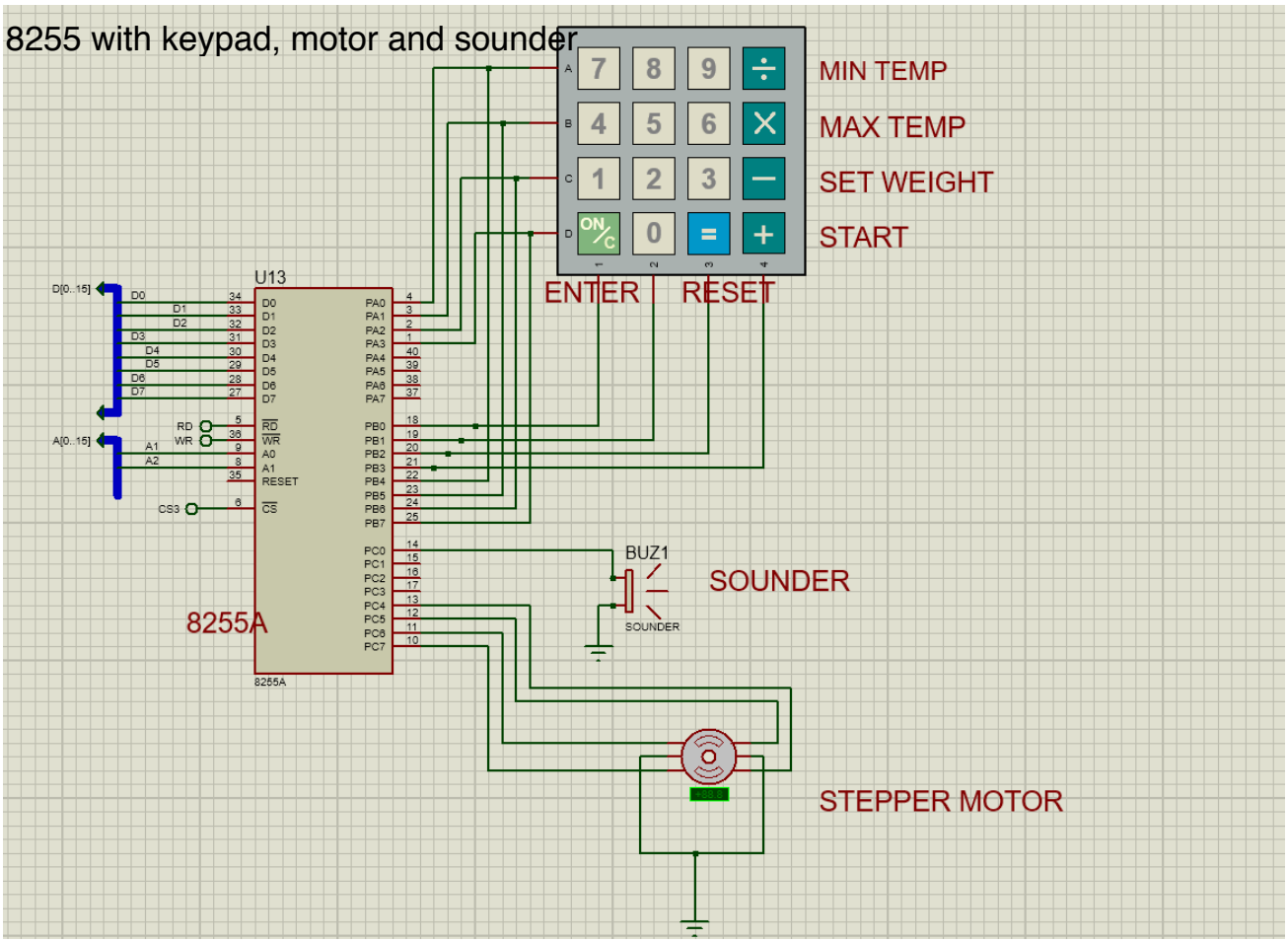
```
    ret

keyPress endp

errorCode proc near

    pushf
    push dx
    push bx
    push cx

    mov dx,8004h          ;turn on the sounder
    mov al,0Fh
    out dx,al

    pop cx
    pop bx
    pop dx
    popf
    ret

errorCode endp
.exit
end
```

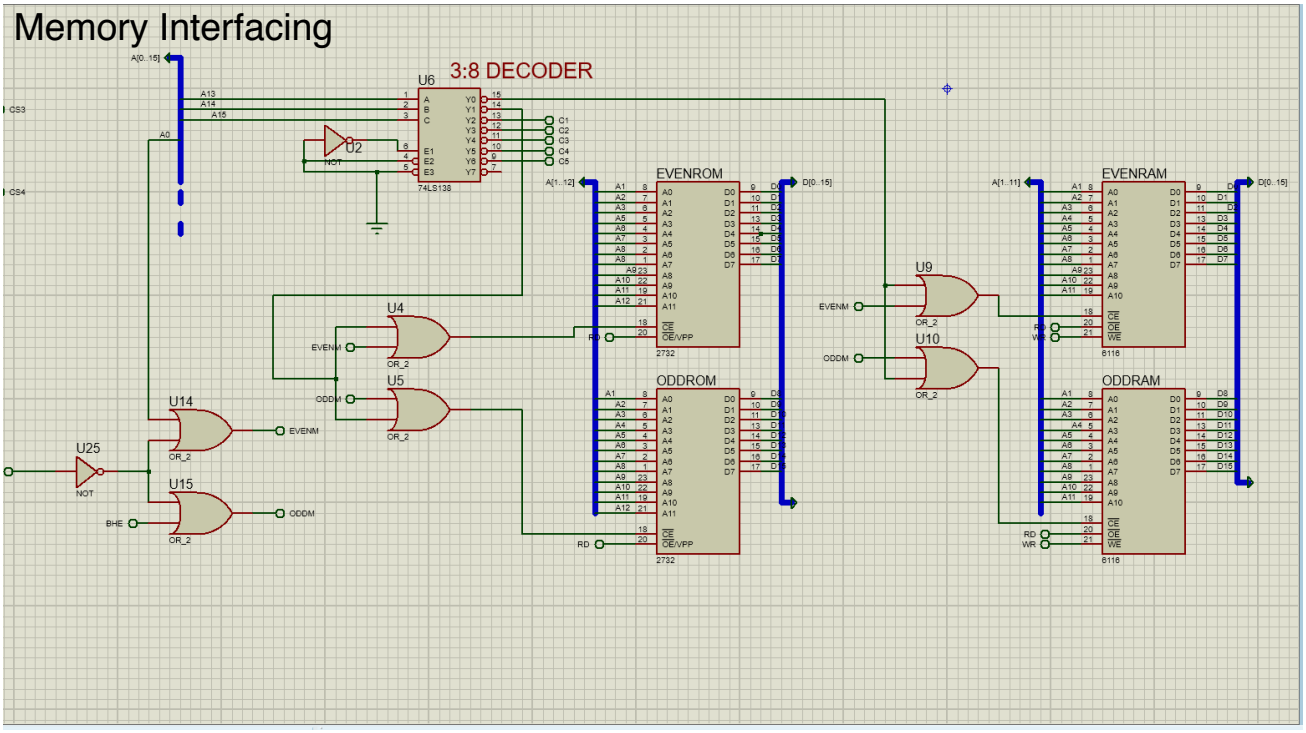
Interfacing 8255 with 7-segment display



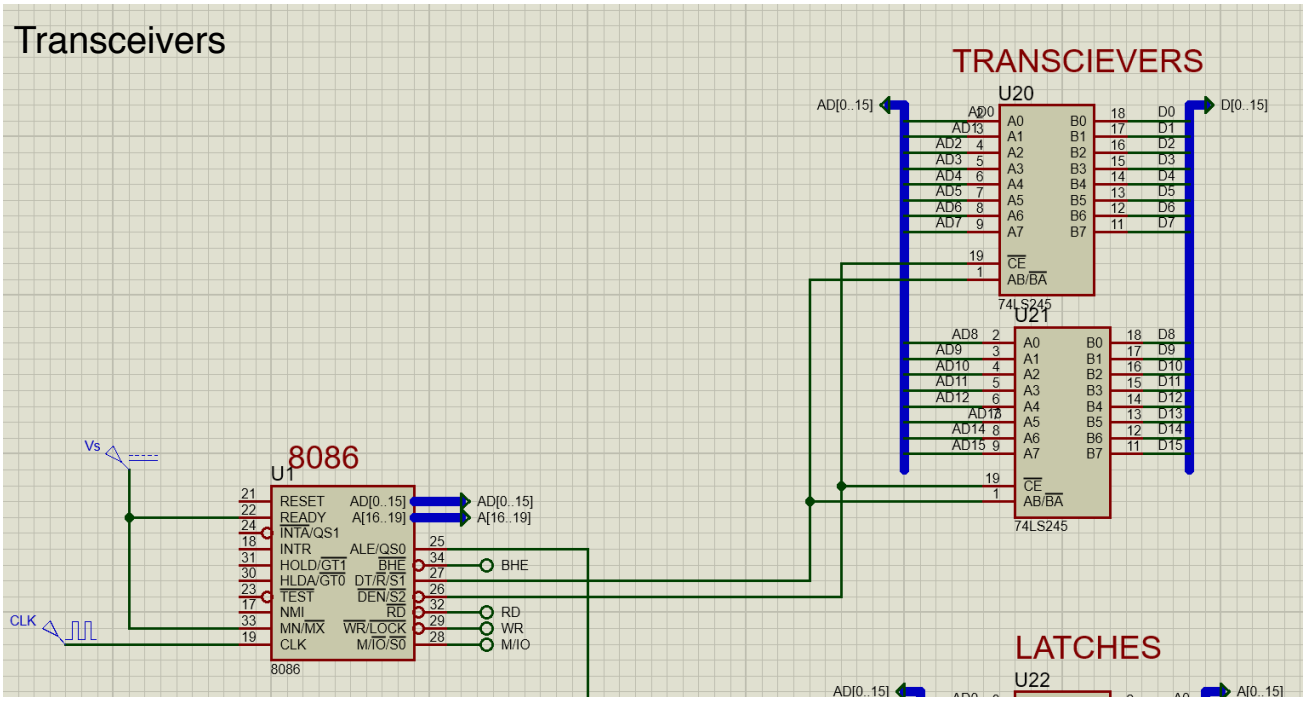
Latching the addresses



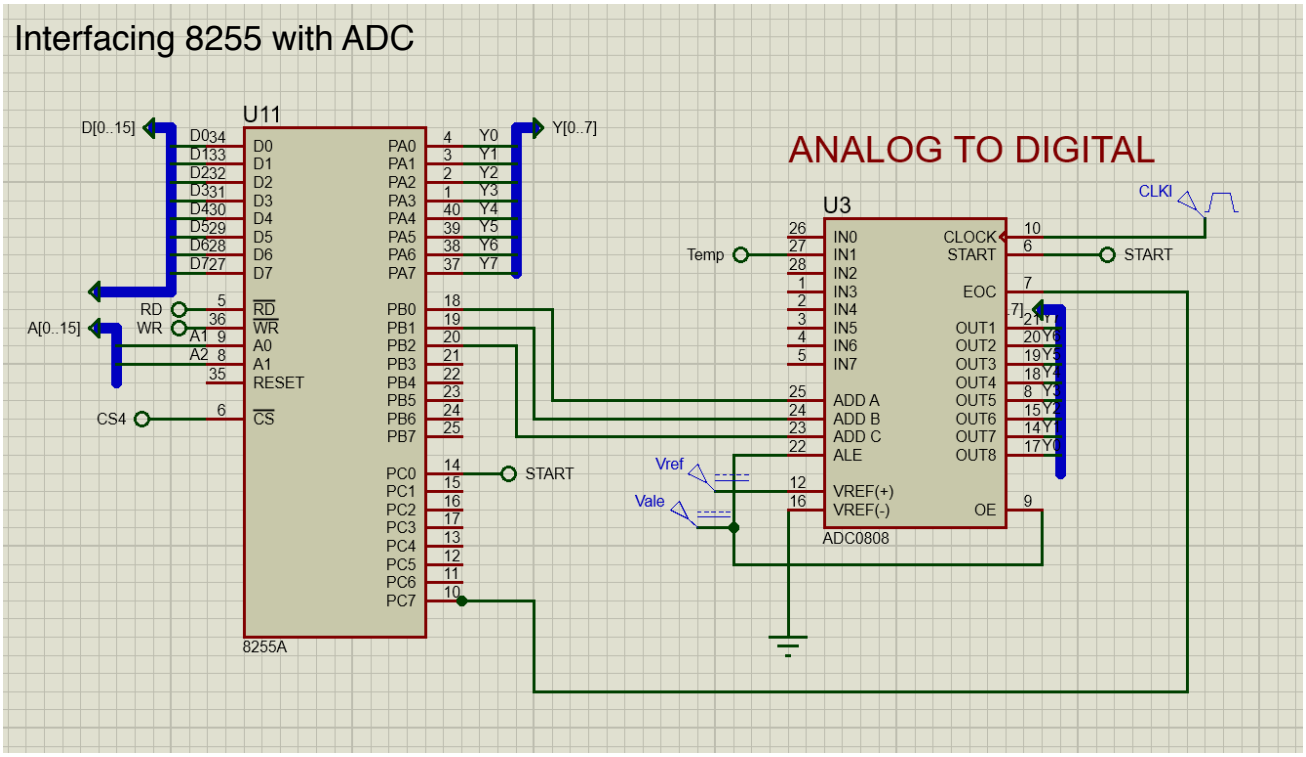

8255 with keypad, motor and sounder
MIN TEMP
MAX TEMP
SET WEIGHT
START
ENTER
RESET
U13
8255A
BUZ1
SOUNDER
STEPPER MOTOR


Memory Interfacing
3:8 DECODER
U6
74LS138
EVENROM
ODDROM
EVENRAM
ODDRAM
2732
6116
U4
U5
U9
U10
U14
U15
U25

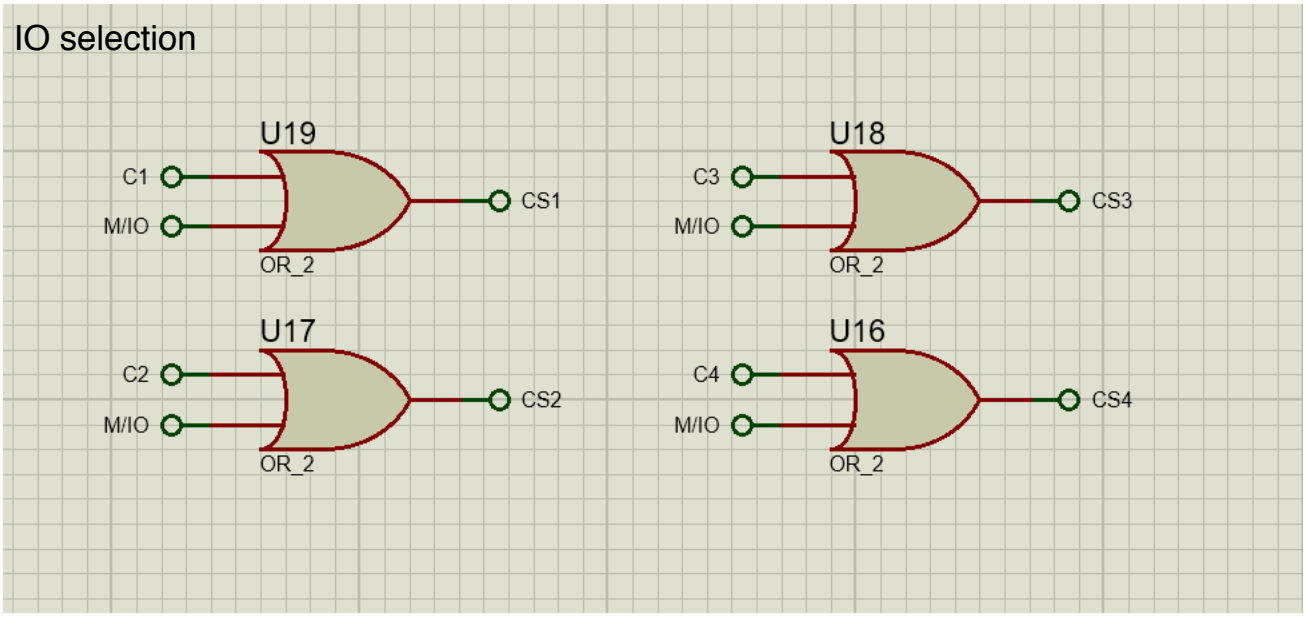
Transceivers



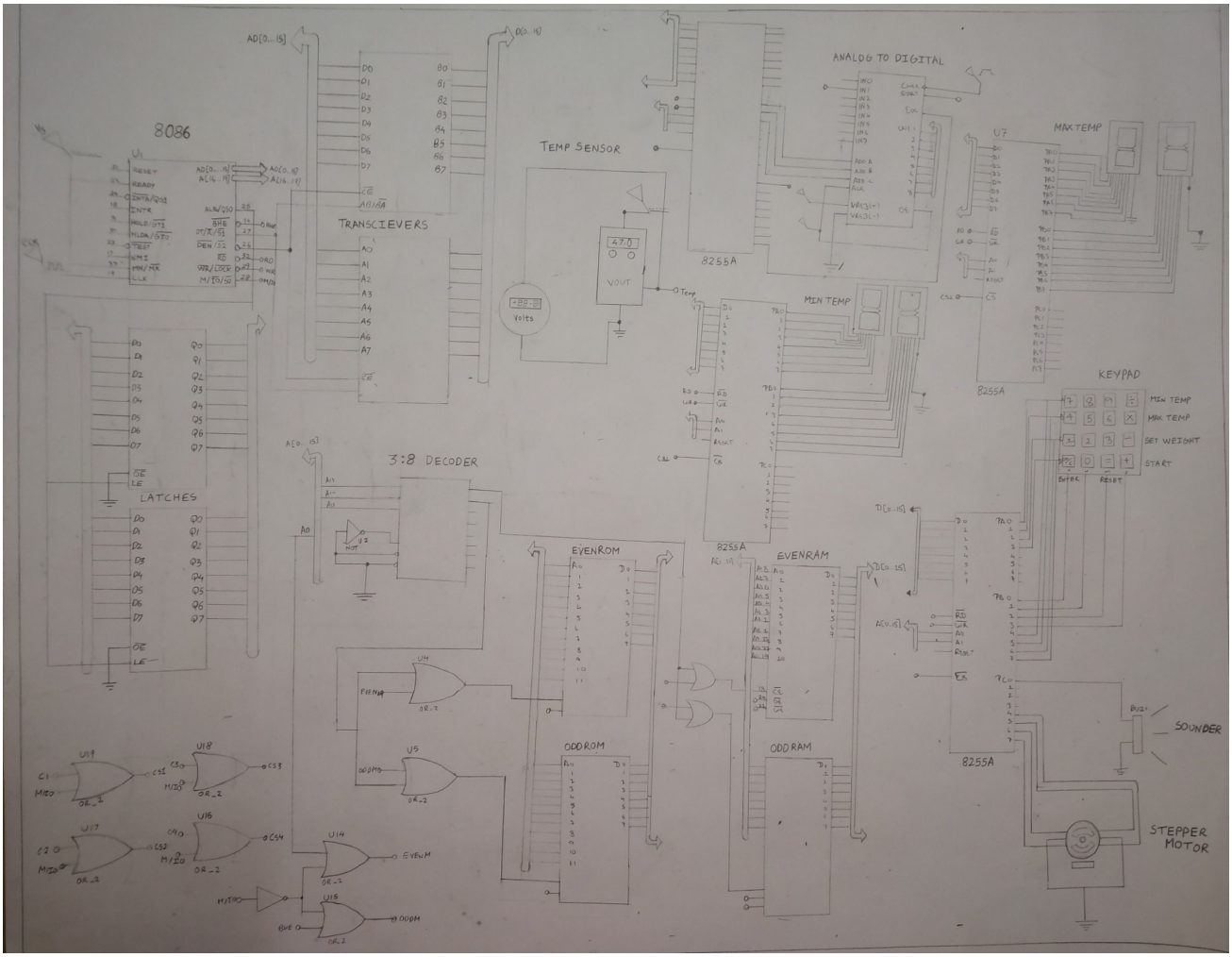
Interfacing 8255 with ADC



IO selection



CHART


8086
TRANSCIEVERS
TEMP SENSOR
ANALOG TO DIGITAL
8255A
MAX TEMP
MIN TEMP
KEYPAD
MIN TEMP
MAX TEMP
SET WEIGHT
START
ENTER
RESET
LATCHES
3:8 DECODER
EVENROM
EVENRAM
ODDROM
ODDRAM
SOUNDER
STEPPER MOTOR

REFERENCES

ADC0808 -analog to digital converter :

http://www.ti.com/lit/ds/symlink/adc0808-n.pdf

LM35- precision centigrade temperature sensor :

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

8255a -programmable peripheral interface :

https://www.renesas.com/in/en/www/doc/datasheet/82c55a.pdf

74LS245- octal bus tri-state transceivers :

http://www.ti.com/lit/ds/symlink/sn54ls245-sp.pdf

74LS138 -3 To 8 decoder :

http://www.ti.com/lit/ds/sdls014/sdls014.pdf

8068 microprocessor :

https://www.ceibo.com/eng/datasheets/Intel-80C86-Data-Sheet.pdf

74LS373 Octal tristate latches :

https://ecee.colorado.edu/~mcclurel/sn74ls373rev5.pdf

2732 4K*8 EPROM :

http://ee.hawaii.edu/~sasaki/EE260/Labs/Datasheets/m2732a.pdf

6116 2K*8 RAM :

http://www.princeton.edu/~mae412/HANDOUTS/Datasheets/6116.pdf