BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

A REPORT ON BATCH WEIGHING SYSTEM

BY

JOYOTI DISHA RAJBONGSHI	2017A8PS0418P
ANUJ HYDRABADI	2017A8PS0420P
BOTLAGUDURU SATVIK	2017A8PS0421P
TANISHQ MALHOTRA	2017A8PS0425P

FOR

INSTR F241 MICROPROCESSOR PROGRAMMING AND INTERFACING

PROBLEM STATEMENT

P8

A microprocessor system is to be designed as a batch weighing machine. The system is interfaced to three load cells by means of a 10-bit A/D converter.

The conditioned output of the load cells is given by the equation: Vout = K x weight (Kgs.), where K is dependent on the property of the sensor.

The system monitors the output of the load cells and finds out the total weight by taking the average of the three values that are sensed by each load cell. This value is displayed on a seven-segment display. When this value exceeds 50 kgs, an output port, which is connected to a relay, is switched on to sound an alarm. Design the necessary hardware and software for implementing the above-mentioned task.

ASSUMPTIONS

1) All load cells are different and hence will give different values of weight.
2) Weight is equivalent to the input voltage of load cell
3) Vout of the load cell is the voltage given by amplifier
4) The A/D converter is 8-bit.

BRIEF SYSTEM DESCRIPTION WITH COMPONENTS USED

Three load cells are used for sensing the weights. The voltage output of the load cells is scaled to 5-volt range with the help of resistance. When the simulation is started, the analog voltage value is converted to its digital equivalent by means of an A/D converter ADC 0808. This value is then multiplied by the conversion factor (100/255) which is used to calculate the real value of weight calculated by load cell. The calculated weight is then compared with the limiting value of the weight which is 50 Kg. If the weight is below this limiting value, it is displayed in the seven-segment display. If not, an alarm is sounded.

Chip Number	**Quantity**	**Chip**	**Purpose**
8086	1	Microprocessor	Central Processing Unit
6116	2	RAM	Read Write Memory to house Data segment and Stack segment
2732	2	EPROM	Read Only Erasable Programmable memory to house the code
8255	1	Programmable Peripheral Interface	Provides I/O port for the other devices
74138	1	3:8 Decoder	Decode the memory/IO and

			read/write signals
7seg-mpx2-ca	1	Seven Segment Display	Display the output Values
ADC 0808	1	Analog to Digital Converter	Converts the analog voltage to its digital equivalent
7447	1	BCD to 7-Segment converter	Converts a BCD value to value required by 7 Segment Display
74LS245	2	8-bit buffer bidirectional buffers	Buffering Data bus
74LS373	3	8-bit latches	Latching the address bus

Apart from the above-mentioned chips, these components are also interfaced:

- Two switches
- One LED –Red
- Alarm Device
- Logic gates and resistors
- A Relay for LED
- VCVS
- Voltage Sources

MEMORY ORGANIZATION

The system uses 4KB of RAM and 8KB of ROM. Both consist of two chips of 2KB and 4 kb size each. They are organized into odd and even bank to facilitate both byte and word size data transfers.

Random Access Memory:
Starting Address: 02000h
Ending Address: 02FFFh

Read Only Memory:
Starting Address: 00000h
Ending Address: 01FFFh

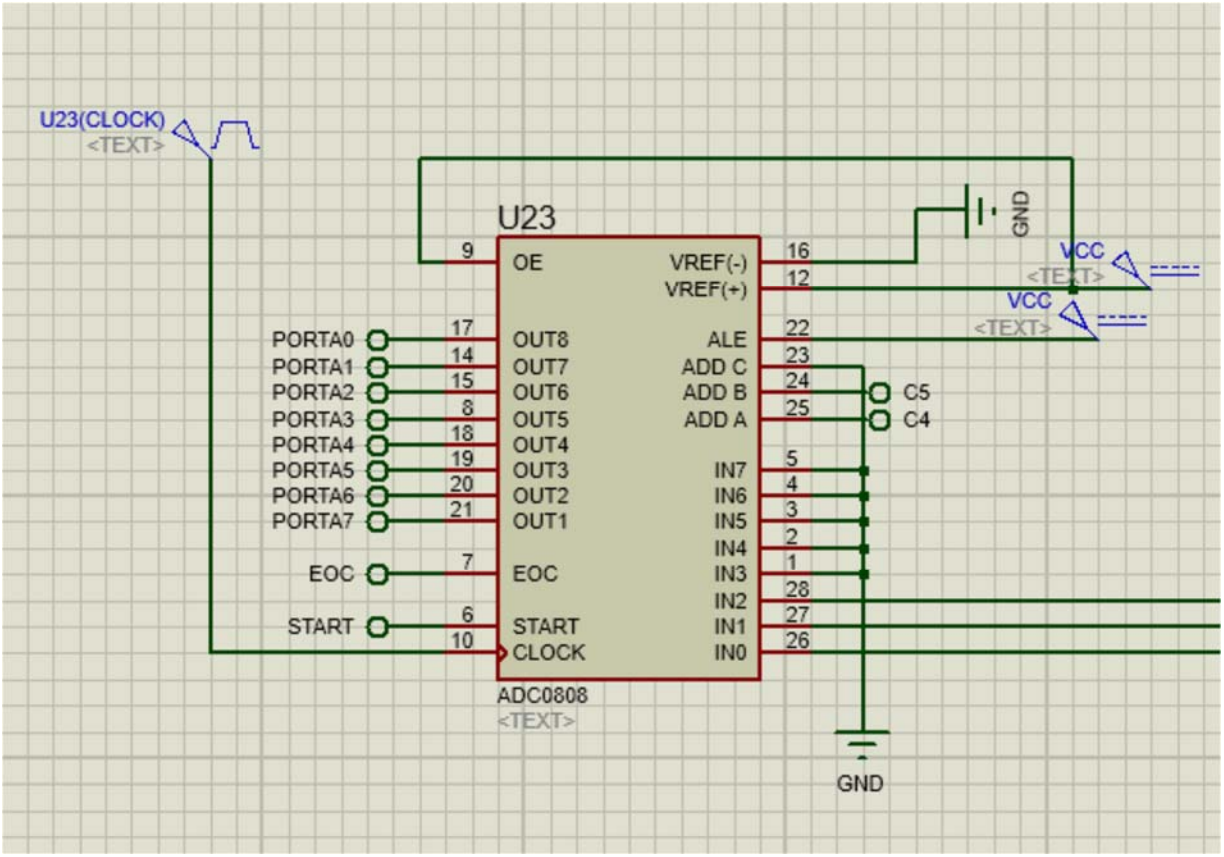
I/O ORGANIZATION

An 8255 is used to communicate with I/O devices. Its Interfacing has been shown below:

Port	**Port address**	**Mode**	**Input/output**	**Connected to**
A	00H	0	Input	ADC
B	02H	0	Output	7447
C Lower	04H		Output	PC0 – 7 Segment Decoder PC1 – 7 Segment Decoder PC2 – Relay for alarm and led PC3 – Start of ADC
C Upper	04H		Output	PC4, PC5- Select lines of ADC's Input (Here these will be used only in BSR-Bitset Reset Mode) PC7 – EOC of ADC
Control Register	06H			

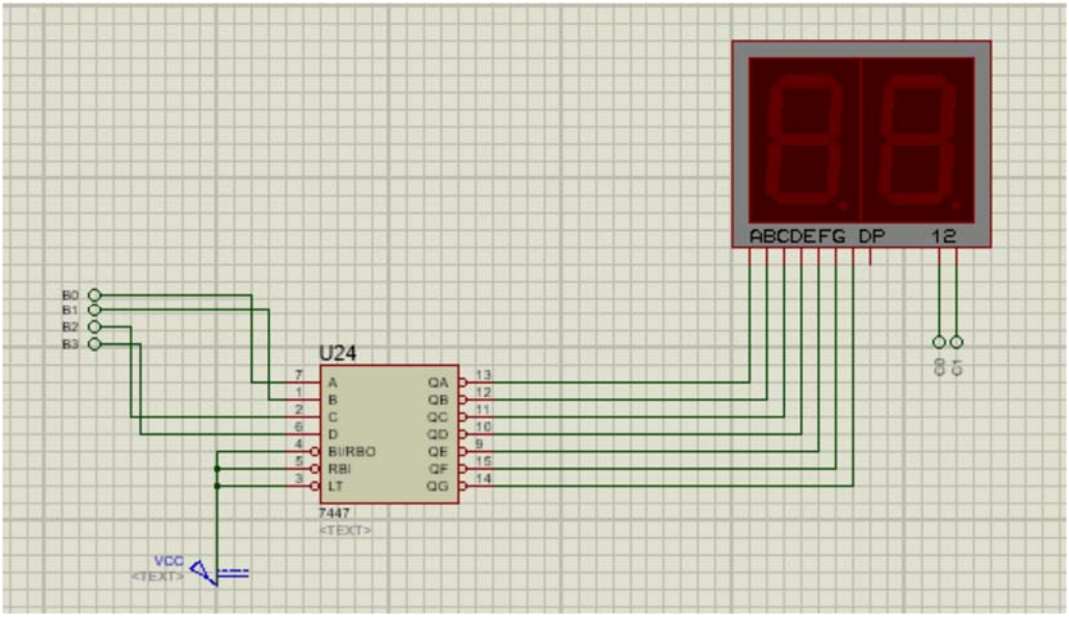
8-BIT ANALOG TO DIGITAL CONVERTER

ADC 0808 has been used with a clock connected at a pulsed high voltage of 5V and it takes 3 input signals in the form of voltages from 3 load cells as shown below. 2 select lines coming from 8255 are used to select among these 3 signals and thus the output 8-bits go into port A of 8255.

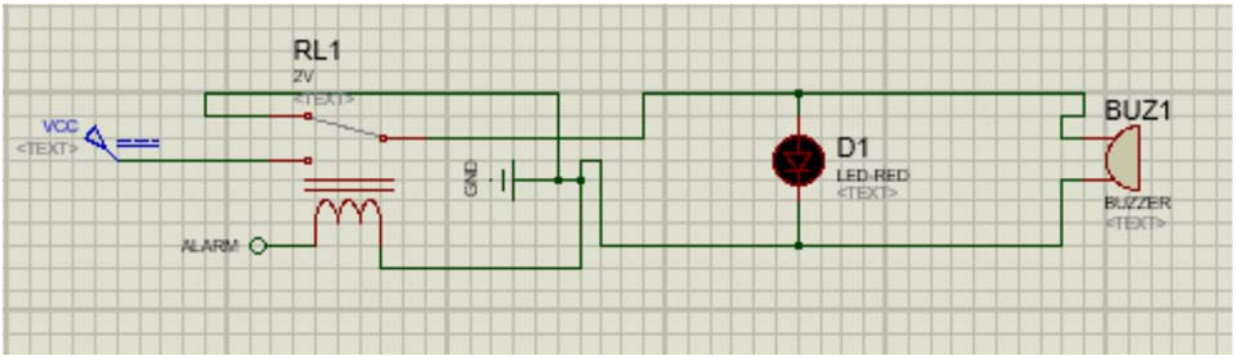


OTHER DEVICES

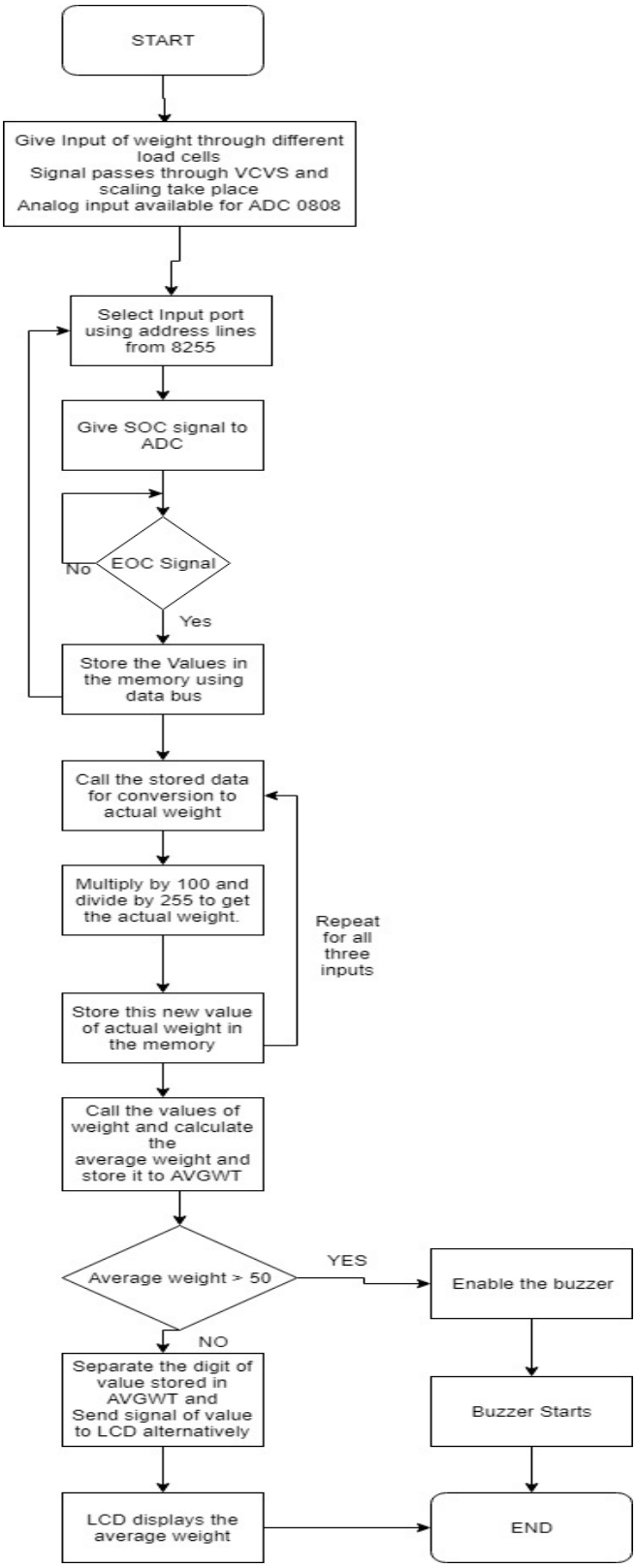
Output 7-segment display along with 7447:



Alarm for testing of heavy weights (weight more than 50kgs) in parallel with an LED:



FLOWCHART


START
Give Input of weight through different load cells
Signal passes through VCVS and scaling take place
Analog input available for ADC 0808
Select Input port using address lines from 8255
Give SOC signal to ADC
No
EOC Signal
Yes
Store the Values in the memory using data bus
Call the stored data for conversion to actual weight
Multiply by 100 and divide by 255 to get the actual weight.
Store this new value of actual weight in the memory
Repeat for all three inputs
Call the values of weight and calculate the average weight and store it to AVGWT
Average weight > 50
YES
Enable the buzzer
Buzzer Starts
NO
Separate the digit of value stored in AVGWT and Send signal of value to LCD alternatively
LCD displays the average weight
END

CODE

```
#make_bin#

.model tiny

.data


; multiplying individual weights by 390 to convert the range of 0-255 to 0-100

MLP DW 390


; Divide by 1000, so its effectively multiplying with 0.39

DIVI DW 1000


DSDIV DB 10                ;to separate the unit and tens digits

AVG DB 03


UNITS DB ?

TENS DB ?

WT1 DB ?

WT2 DB ?

WT3 DB ?

WTAVG DB ?



porta equ 00h

portb equ 02h

portc equ 04h


creg equ 06h
```

```
stack dw 100 dup(?)
top_stack label word


.code
.startup

LEA SP,top_stack                    ;initializing stack for the sub routine
MOV AL,90h                          ;10010000b
        ;port A input
        ;port B & C output of 8255
OUT creg,AL



;BSR mode to select pins of ADC
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Selecting IN0;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MOV AL,08h              ;00001000b
OUT creg,AL     ;PC4 logic zero B is 0
MOV AL,0ah              ;00001010b
OUT creg,AL     ;PC5 logic zero C is 0, A was grounded



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;IN0 of ADC is selected;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;sending low and high pulses with delay to ADC to convert analog to digital
CALL delay_soc
;;;;;;;;;Sends high and low pulse with delay in between on pc3;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;Checking of EOC;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MOV AL,90h  ;10010000b port A input & port B & C output
OUT creg,AL

;CHECKING FOR END OF CONVERSION ;(whether EOC= LOGIC 1)
;CHECKING STARTS

X1:
IN AL,04h          ;loading in the results of port c pins
AND AL,80h         ; considering only the result of pin 7 (eoc)
JZ X1              ;;if eoc ==0, check again till eoc ==1

;;;;;;;;;;;;;;;Taking input of in0;;;;;;;;;;;;;;;;;;;;;;;
MOV AL,90h         ;10010000b port A input & port B & C output,
OUT creg,AL        ; A has values of load cell
IN AL,porta        ;dig.o/p of ADC is read
MOV WT1,AL         ;store o/p into mem as weight 1

;;;;;;;;;;;;;;;;Selecting input of IN1

MOV AL,09h         ;00001001b
OUT creg,AL        ;pc4 logic one
MOV AL,0ah         ;00001010b
OUT creg,AL        ;pc5 logic zero
```

```
;IN1 is selected

CALL delay_soc          ;100 ns low to ADC
MOV AL,90h              ;10010000b port A input & port B & C output
OUT creg,AL

;;;;;;;;;;;;Checking for EOC;;;;;;;;;;;;;;;;;;
X2:
IN AL,portc             ; reading port c results
AND AL,80h              ; considering only pin 7.
JZ X2

;;;;;;;;;;;;;;;;Taking input of in2 and saving
MOV AL,90h              ;1001000b
OUT creg,AL
IN AL,portA             ;dig o/p. of ADC is read
MOV WT2,AL    ;store o/p. into mem as weight 2.


;Selected IN2


MOV AL,08h              ;00001000b
OUT creg,AL     ;set PC4 logic zero
MOV AL,0bh              ;00001011b
OUT creg,AL     ;set PC5 logic one

;IN2 is selected
```

```
CALL delay_soc          ;100ns low to ADC.

MOV AL,90h              ;10010000b
OUT creg,AL

;;;;;;;;;;;;;;Checking whether in3 is still computing;;;;;;;;;;;
X3:

IN AL,portc             ; reading port c data
AND AL,80h              ; data of only pin 7 is considered.
JZ X3

;;;;;;;;;;;;;;;;;;Saving in3;;;;;;;;;;;;;
MOV AL,90h              ;10010000b
OUT creg,AL
IN AL,porta             ;dig o/p of ADC is read
MOV WT3,AL    ;Store o/p into mem as weight 3.


;;;;;;;;;;;;;;;;;;;;Calculating converted Weight


MOV AH,00h
MOV AL,WT1    ; AX register now holds weight 1.
MUL MLP
DIV DIVI                ;changing the range of data to 0-100
MOV WT1,AL    ;i/p wt of load1 moved to WT1

MOV AH,00h
```

```
MOV AL,WT2      ; AX register now holds weight 2.
MUL MLP
DIV DIVI                    ;changing the range of data to 0-100
MOV WT2,AL      ;i/p wt of load2 moved to WT2

MOV AH,00h
MOV AL,WT3      ; AX register now holds weight 3.
MUL MLP
DIV DIVI                    ;changing the range of data to 0-100
MOV WT3,AL      ;i/p wt of load3 moved to wt3

;Calculating average of WT1,WT2 and WT3

mov ah,00h
mov bh,00h
MOV Al,WT1
MOV bl,WT2
ADD AX,BX                   ; adding wt1 and wt2
MOV bl,WT3
ADD AX,BX                   ; total wt is now stored in ax
DIV AVG                     ;Avg of 3 wts moved to AL
MOV WTAVG,AL                ;Moving avg to mem in wtavg
ADD AL,1
CMP AL,50                   ;Check if AVGWT < 50kg
jb FINAL_DISPLAY     ;if avg is below 50 display on screen
;;;;;;;;;;;;;;;;;;;;;;Average weight has been calculated;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;Alarm Part;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
s1:
```

```
MOV AL,05h                ;00000101b

;;;;;;;Making pc2 1 to ring alarm;;;;;;;;;;;;;;;;

;;;;;;;;;;;This will send the machine into an infinite loop of alarm.. If you reset the whole circuit then only machine will stop ringing ;;;;

OUT creg,AL       ;alarm if(load>50kg)

JMP END_LAST

;display (wt<50kg)

FINAL_DISPLAY: MOV AH,00h

MOV AL,WTAVG

DIV DSDIV                 ;Separating two digits of weight

;Storing digits in mem

MOV TENS,AL

MOV UNITS,AH

Y1:

MOV AL,01h                ;00000001b

OUT creg,AL       ;(pc0)

;;;;;;;;;;;;;PC0 is connected to units display, this activates it;;;;;;;;;;;;;;;;

;Switch on units digit display
```

```
;;;;;;;;;;;;;;;;Making Port B output;;;;;;;;;;;;;;;;;;;;;
MOV AL,90h              ;10010000b
OUT creg,AL




MOV AL,UNITS
OUT portb,AL

;;;;;;;;;;;;;;Reset pc0 to switch off display of units;;;;;;;;;;;;;;;;
MOV AL,00h
OUT creg,AL

;;;;;;;;;;;;;;;;;Set pc1 to switch on display of tens;;;;;;;;;;;;;;;;
MOV AL,03h              ;00000011b
OUT creg,AL     ;(pc1)



;;;;;;;;;;;;;;;;Show tens digit display by making port b output
MOV AL,90h              ;10010000b
OUT creg,AL

;i/o mode to input 7447

;;;;;;;;;;;;;;;Showing tens value on port b;;;;;;;;;;;;;;;;
MOV AL,TENS
OUT portb,AL

   ;;;;;;;;;;;;;;;;Reset pc1 to switch off tens display;;;;;;;;;;;
```

```
MOV AL,02h                ;00000010b
OUT creg,AL

JMP Y1
 END_LAST: mov cx,00h
;;;;;;;;;;;;;;Creating slight delay;;;;;;;;;;;;;;;;;;
mov cx,0ffh
 t:dec cx
cmp cx,00h
JNZ t
Jmp s1                    ; loop for sounding the buzzer.

.exit

;;;;;;;;;;;;;;;;;Pulse generating procedure;;;;;;;;;;;;;;;;;;

delay_soc proc near
MOV AL,06h                ;00000110b
OUT creg,AL       ;set pc3 low
mov cx,00h
mov cx,01h                        ;For delay between high and low
dec cx
MOV AL,07h                ;00000111b
OUT creg,AL       ;set pc3 high
RET
delay_soc endp
End
```

COMPLETE CIRCUIT DIAGRAM