Microprocessor Design Project Report

BATCH WEIGHING MACHINE

Prepared in partial fulfillment of the requirements for the course-

Microprocessors and Interfacing
(CS/EEE/INSTR F241)

Submitted To:

Prof. K.R. Anupama
(Department of Electrical and Electronical Engineering

Submitted By:

Aseem Juneja(2018A7PS0726G)
Aayush Mathur(2018A7PS0729G)
Sridhar Dhamija(2018A8PS0707)
Priyesh Kant(2018A8PS0467G)
Aman Gehani(2018A8PS0061G)

Contents

1 Acknowledgement

We express our sincere gratitude to Dr. K.R. Anupama, for giving us this opportunity to work on such an amazing project.

Such an application has helped us gained knowledge of the principles of microprocessor interfacing and hardware programming, which has been applied at various stages of development of this project.

We are also indebted to other instructors in this course for guiding us during the whole project. The project has given us a great insight into the depth of Microprocessor Programming and Interfacing. It helped us in practically applying the major principles of interfacing viz. memory interfacing and I/O interfacing.

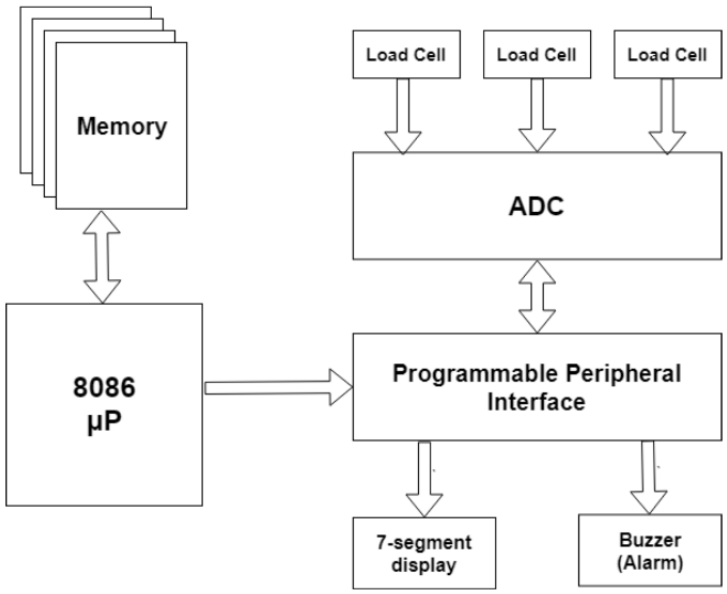
Finally we extend our thanks to friends for their continuous support and encouragement.

2 Summary

An x86 microprocessor system is designed as a Batch Weighing Machine interfaced to three load cells. The system monitors the output of the load cells and finds the total weight with values sensed by the load cells. A seven-segment display and an alarm were used for visual and auditory feedback respectively.

The entire project can be looked into 4 segments:

- The Brain: This involves working with the microprocessor 8086. It is via this component that all the other components are combined.
- Memory Interfacing: This part makes use of RAM, ROM and decoder to allocate memory for doing all the operations. This involves interfacing RAM and ROM with the brain.
- Making Input and Output Devices Work: This part involves using input and output devices to work.
  Input Devices: This starts with measuring the weight via load cells, converting that to digital signal after amplification.
  Output Devices: This starts with using Seven Segment LCD Display to get the output of weight and buzzer to create an alarm if the weight exceeds 99Kgs.
- I/O Interfacing: This involves connecting the i/o devices to the brain(micrprocessor) via programmable peripheral interface(8255), latches and buffers.



The weight is sensed by three load cells. The load cells we have used have an output range of 0-10mV. When simulation starts, the analog voltage value already loaded usiing potentiometer is conveted to its digital equivalent by means of an ADC. This value is then multiplied by the conversion factor which is dependent on the relation between output voltage and weight. The process is repeated for three load cells and the average is calculated. The calculating weight is then compared with the limiting value of the weight which is 100kg. If the weight is above this limit value the alarm is sounded. The weight is then displayed in four seven-segment displays and the decimal part of the weight s is displayed in the other two displays. Then the system moves to a polling state, where it keeps looking at the stop-alarm switch. If the switch is pressed, the alarm is stopped, but the system still remains in the polling phase. For further weight reads, user can change the weights using potentiometer, and turn the read-weight switch ON, which triggers NMI, restarting the process of weight reading and display.

3 Problem Statement

A microprocessor system is to be designed as a batch weighing machine. The system is interfaced to three load cells by means of an 8 bit A/D converter. The conditioned output of the load cells is given by the equation:

V = 0.025*weight(Kgs)

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 99 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. Once the objects are placed on the load cell user presses a switch labelled weigh.

4 Assumptions

- The starting address of ROM is 00000h
- The starting address of RAM is 02000h
- The maximum capacity of weight that can be measured by each load cell is 200kgs but the ADC can measure only up to 256 bytes. Since we are using an 8 bit ADC (ADC 0808).
- Each load cell gives very low voltage output (mV) which has to be amplified since the 0808 ADC works in the range of volts.
- We have used potentiometer to input the amplified values of the load cell input.
- Firmware- Implemented using emu8086 and proteus7
- If the alarm goes off, the user may stop the alarm using the LogicState component for Alarm. However, the LogicState must be set to 0 before taking the weight.

5 Components Used

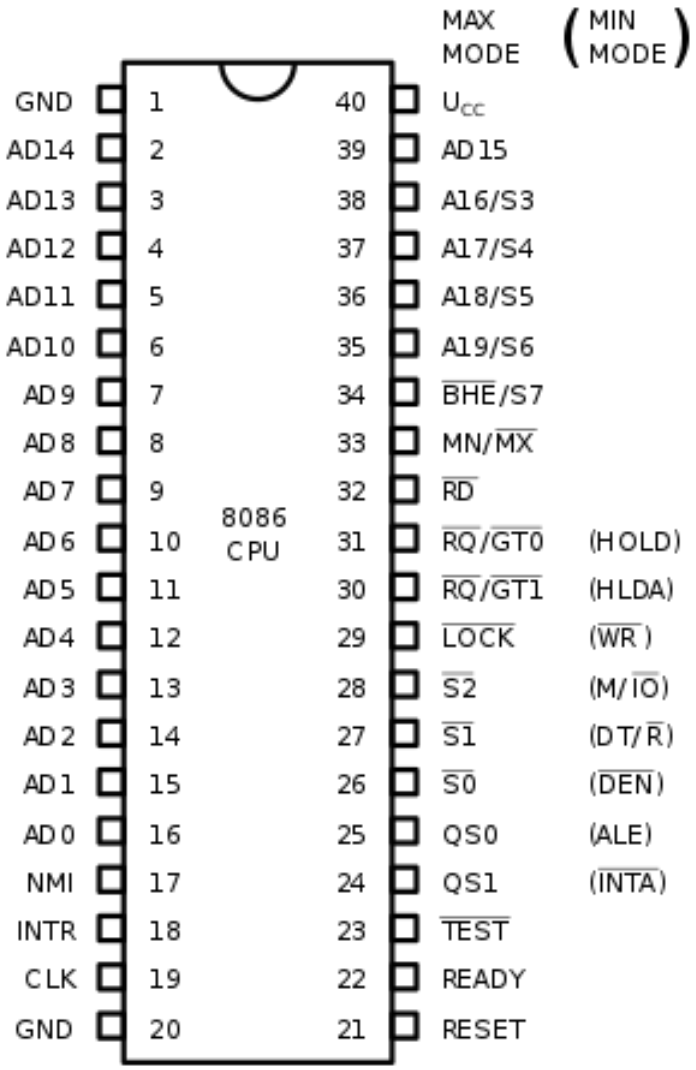
Description	Name	Quantity
Microprocessor	8086	1
Programmable Peripheral Interface	8255	2
Analog to Digital COnverter	0808	1
Load Cell	Combination of op amp and resistors	3
Relay(12V)	Relay	1
Buzzers(12V)	Buzzer	1
7 Segment Display	Common Anode	4
BCD to 7 Segment Decoder	7447	1
Switch	SW-SPDT-MOM	2
Op Amp	INA122	3
RAM	61666(4K)	2
ROM	2732(8K)	2
OR Gates	IC 7432	1
Clock Generator	8284	1
Latches	74LS373	3
Buffers	74LS245	2

Table 1: Components Used

6 Learning About the Components

6.1 8086

8086 is a central processor used in the project. This can be named as the brain of the project. It is via this brain that all other components are linked. Have a look at the pin diagram of 8086.



6.2 Components for Creating the Input

6.2.1 Load Cell

Load Cell performs the job of measuring the force and outputs the force in form of electrical signal.A load cell consists of four strain gauges in a Wheatstone configuration. The electrical signal output from a load cell is normally in the order of millivolts and requires amplification by an instrumentation amplifier before it can be used. The output of load cell is given as input in amplifier which will finally produce a voltage in range of 0-5V.

Capacity of the load cells = 200Kg
Excitation of load cells = 10V

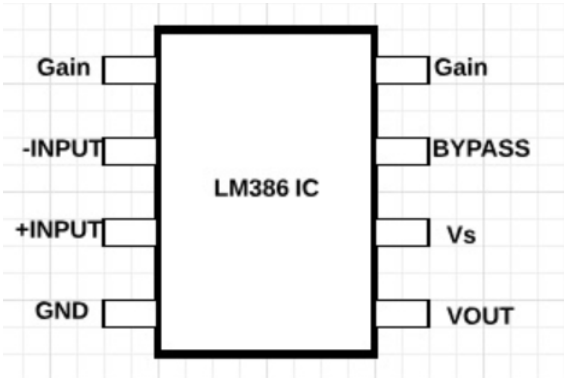
Rated output of the load cell at this capacity = 2mV/V
Output Voltage of cells = (Rated Output) * (Excitation Value)
Output Voltage of cells = 2mV/V * 10V = 20mV.

This is given as an input to ADC. But we need to amplify this voltage before passing it as an input to ADC.

6.2.2 Amplifier

The role of Amplifier is to convert small input voltage into large output voltage.

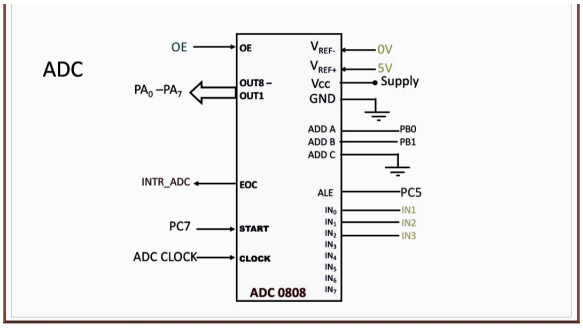
In our project, we pass 20mV of voltage as input and expect amplified output. Look at the IC of Amplifier :



6.2.3 Analog to Digital Converter

The result of load cells passed from amplifier is of the form of electrical signal. To pass this result to the microprocessor, we need to convert the signal from its analog form to digital which is done by ADC.

Resolution of ADC = 0.025V
8 bit ADC can produce = 256 outputs
V(out) = Resolution*Weight
Max(Weight) = 200Kgs
256 values map 200 kgs
Therefore, conversion constant is 200/256 = 0.78125

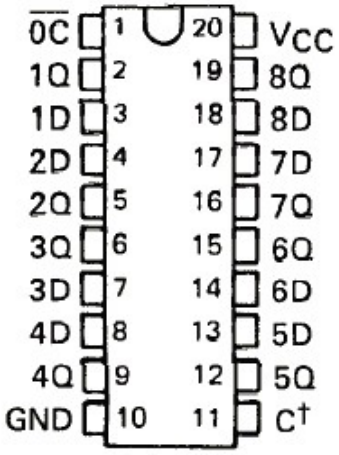


6.3 Components for Connecting I/O Interfacing

6.3.1 Latches

Latch is used to make the connection of microprocessor with output devices. Since the result stays on data lines for short period of time, we use latches so that we can retrieve the results from the microprocessor and

can store it to pass it to output devices for results. Along with that they also provide the I/O isolation from microprocessor. In our design, we have 19 data lines which carry output, and latches have 8 input lines, thus 3 latches are used. Latches are used for demultiplexing and latching the address bus.

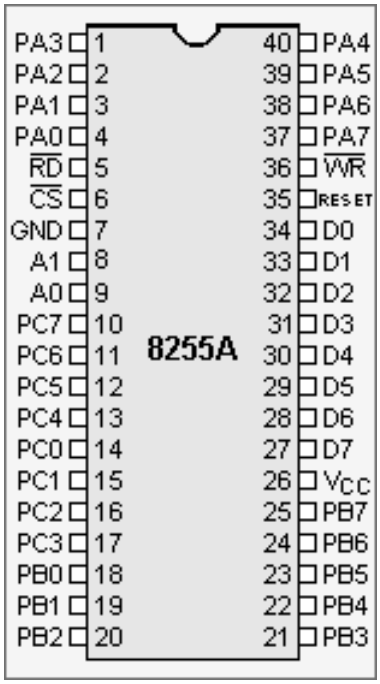


6.3.2 Buffer

Buffer is used to connect input devices to the microprocessor. It's main use is to provide isolation to i/o devices from Microprocessor, strengthening the imput signal and to make data available for long time. In our design, we used to buffer data bus.

6.3.3 Programmable Peripheral Interface

8255 is a peripheral interface which is used to interface I/O devices to Microprocessor. Have a look at the pin diagram:



In our design, two such chips are used, one which connects the input device i.e. the result of ADC to the microprocessor and the other which connects Seven-Segment Display and Buzzer to the microprocessor.

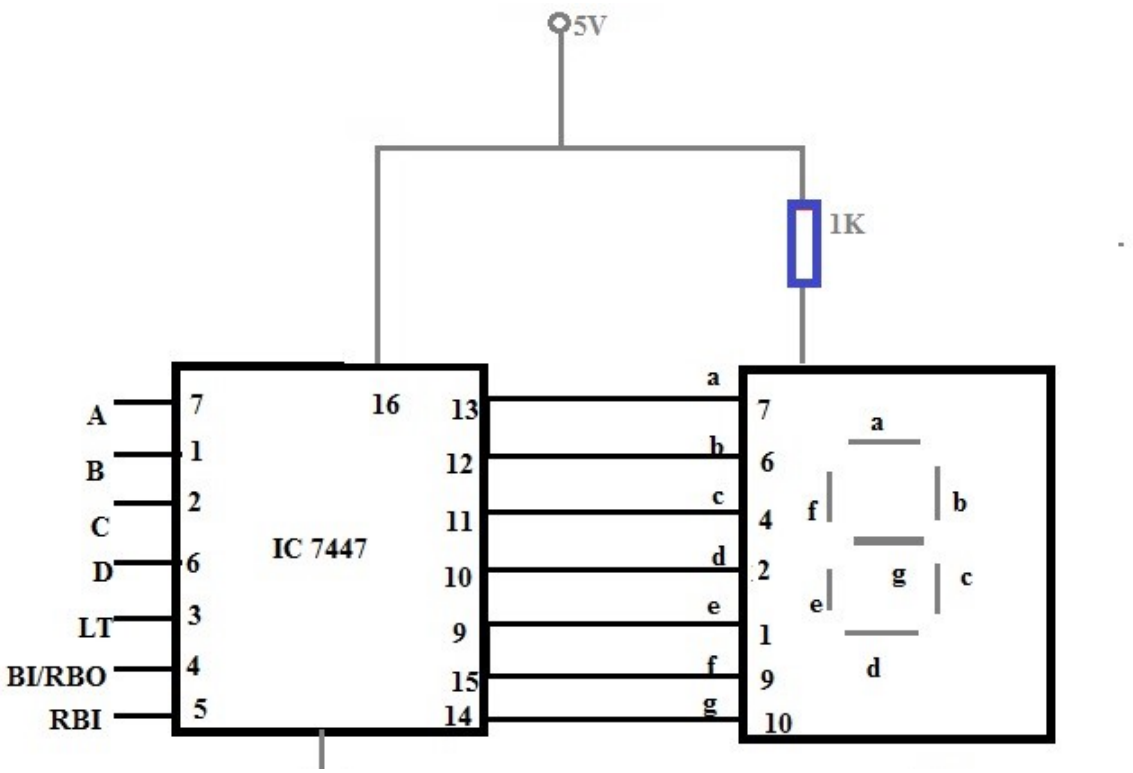
6.4 Components for getting output

6.4.1 Buzzer

This is used to create an alarming sound, if the output exceeds 99Kg. This is connected to the peripheral interface which further is connected to MUP.

6.4.2 BCD to 7 segment Decoder and 7 segment Display

In Binary Coded Decimal encoding scheme each of the decimal numbers is represented by its equivalent binary pattern, whereas, Seven segment display is an electronic device which consists of seven Light Emitting Diodes arranged in a some definite pattern, which is used to display Hexadecimal numerals. Now, to convert BCD to 7 segment segment display, a decoder is used in between.



7 Address Map

7.1 Memory Interfacing

ROM1: 00000 - 01FFF
ROM2: FE000 - FFFFF
RAM1: FE000 - FFFFF

ROM1(8K) is further divided into:-
1) ROM1 Even (4K)
2) ROM1 Odd (4K)

ROM2(8K) is further divided into :-
1) ROM1 Even (4K)
2) ROM1 Odd (4K)

RAM1(4K) is further divided into :-
1) RAM1 Even (2K)
2) RAM1 Odd (2K)

A20	A19	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
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

Table 2: ROM1

A20	A19	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	1	1	1	1	1	1	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	1	1	1	1	1	1	1

Table 3: ROM2

A20	A19	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
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

Table 4: RAM1

7.2 Input/Output Map

Two Peripheral Interfaces are used to communicate with input and output devices. It is organized in the following manner:-

8255(1):
Base Address - 00h

Port	Port Address	Mode	Input/Output	Connected to
A(PA0-PA2)	00h	0	Output	add A,B, C
B	02h	0	Input	OUT 1-8
C Lower(PC0, PC1)	04h	-	Input	PC0 to Start pin, PC1 to ALE pin
C Upper (PC4 only)	04h	-	Input	ADC Converter(EOC Pin)
Control Register	06h	-	-	-

Table 5: Peripheral Interface No. 1

8255(2)-:
Base Address - 10h

Port	Port Address	Mode	Input/Output	Connected to
A	10h	0	Output	2*(BCD to 7 seg decoder)
B	12h	0	Output	2*(BCD to 7 seg decoder)
C Lower(PC0)	14h	-	onput	Buzzer
C Upper (PC4 only)	14h	-	Input	Stop Alarm Switch
Control Register	16h	-	-	-
-				

Table 6: Peripheral Interface No. 1

IVT TABLE: Only non maskable interrupt is used, which by default maps to the vector number 02h. Interrupt is invoked when the read-weight switch is pressed to ON. It runs the weight-reading code sequence.

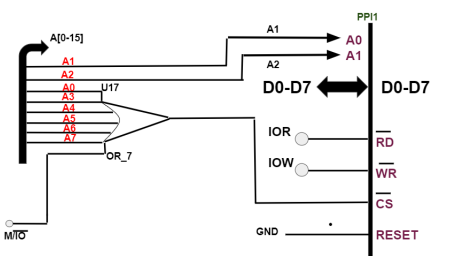


Figure 1: PPI (1)

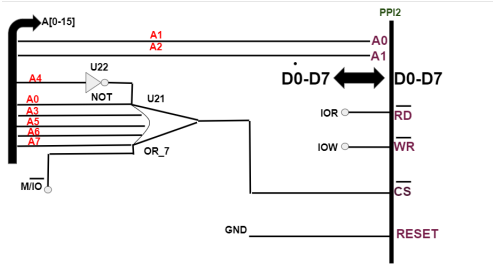
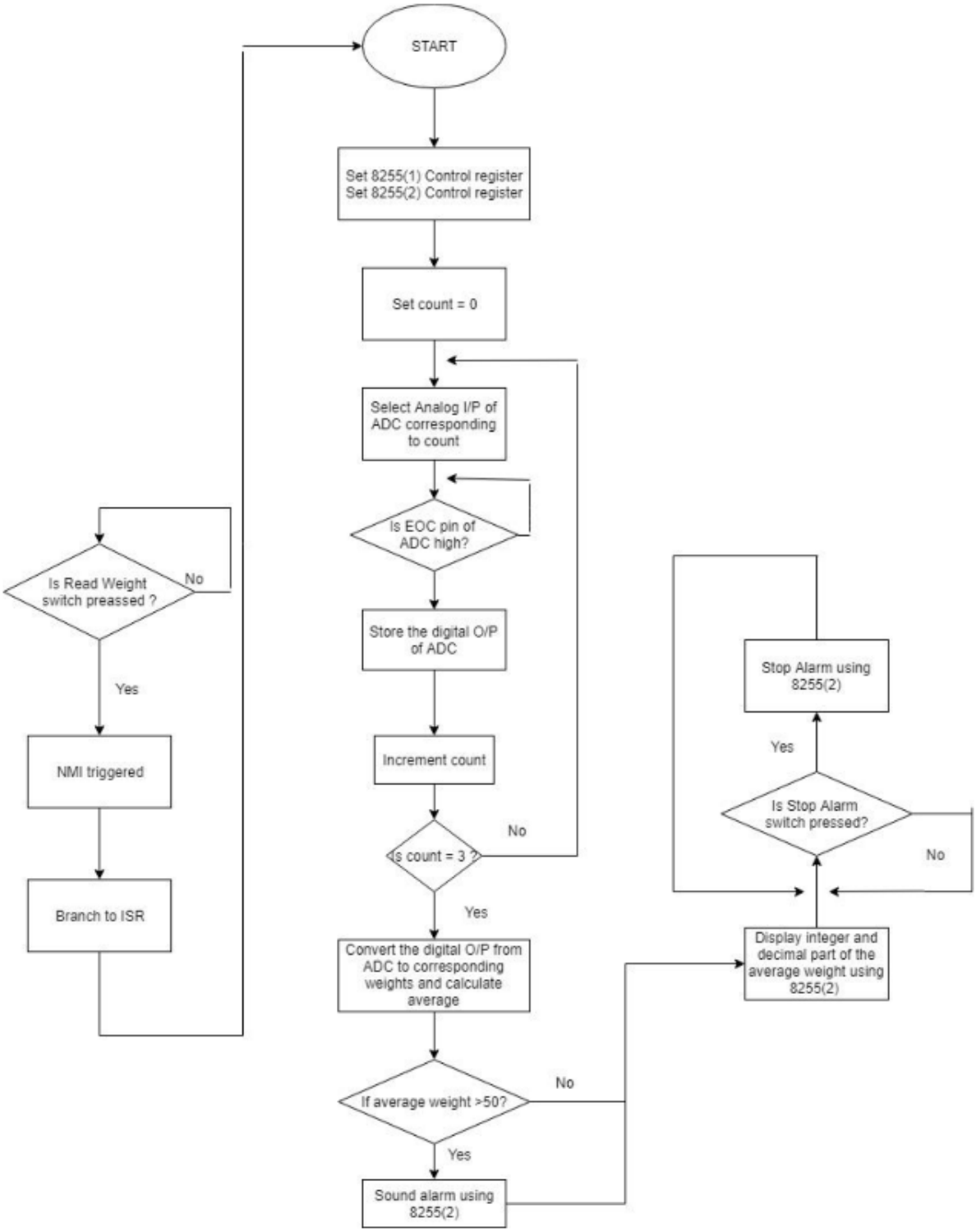


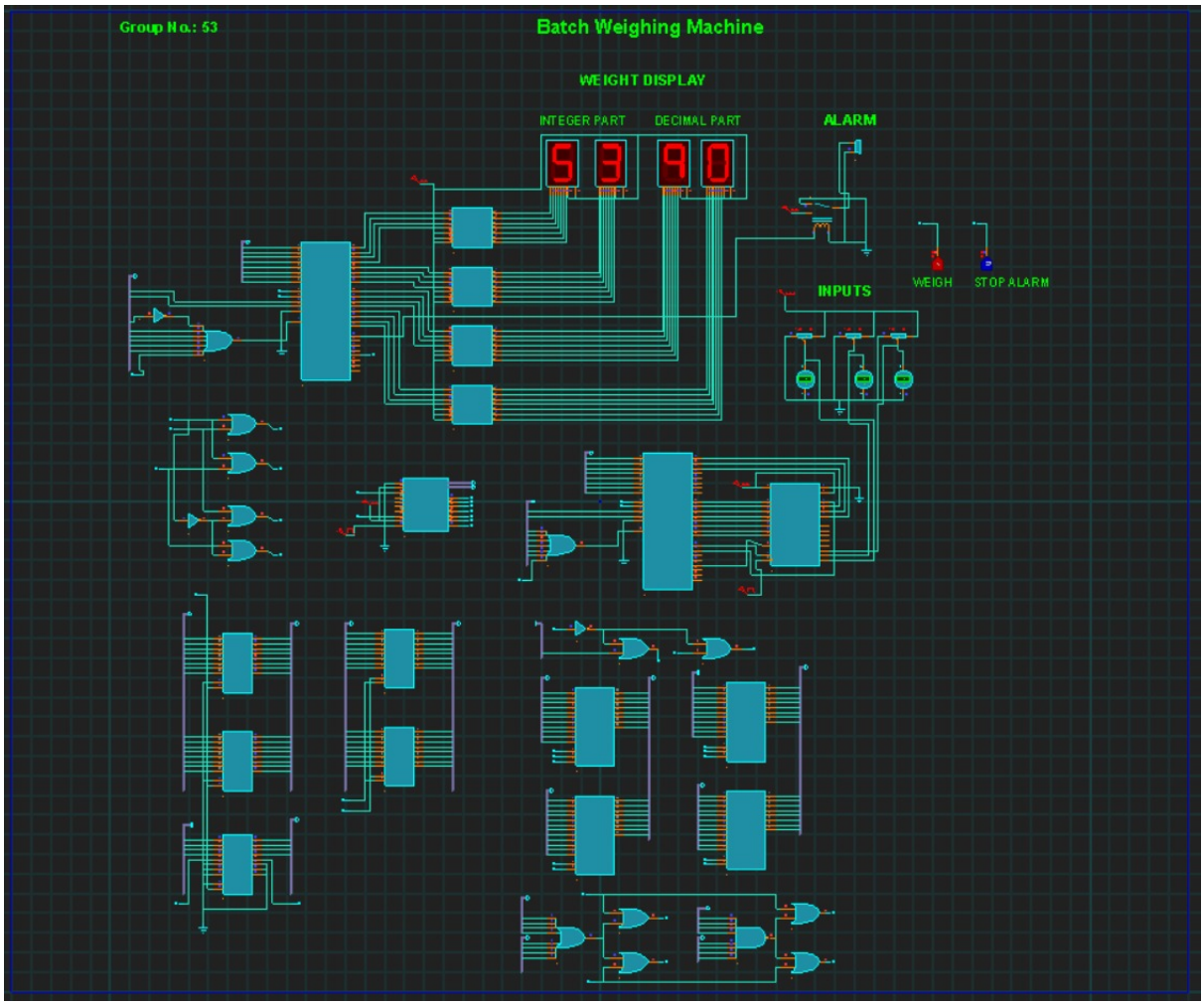
Figure 2: Left Side of PPI (2)

8 FlowChart


START
Set 8255(1) Control register
Set 8255(2) Control register
Set count = 0
Select Analog I/P of ADC corresponding to count
Is EOC pin of ADC high?
Store the digital O/P of ADC
Increment count
Is count = 3 ?
No
Yes
Convert the digital O/P from ADC to corresponding weights and calculate average
If average weight >50?
No
Yes
Sound alarm using 8255(2)
Display integer and decimal part of the average weight using 8255(2)
Is Stop Alarm switch pressed?
No
Yes
Stop Alarm using 8255(2)
Is Read Weight switch preassed ?
No
Yes
NMI triggered
Branch to ISR

9 Design

Check the given link to get the proteus file of given project. Indentation and visual understanding has been focused upon along with the logic behind the work. Link: `https://github.com/dhamija-sridhar/Batch-Weighing-Machine`



10 Assembly Language Code

Check the given link to find the ASM code behind this project. Link: `https://github.com/dhamija-sridhar/Batch-Weighing-Machine`

```
CREG1 EQU 06H

PORTA2 EQU 10H
PORTB2 EQU 12H
PORTC2 EQU 14H
CREG2 EQU 16H

ANLG DB 00H, 01H, 02H
WEIGHTS DB 3 DUP(?)
.........
START:.........

    ; SET ALARM OFF INITIALLY
    MOV AL, 10001000B
    OUT CREG2, AL
    
    MOV AL, 00H
    OUT PORTC2, AL
    
READWEIGHT:
    ; INITIALIZE THE REGISTERS
    MOV AX, 0
    MOV BX, 0
    MOV BP, 0
    MOV SI, OFFSET ANLG
    MOV DI, OFFSET WEIGHTS

WEIGHTLOOP:
    ; CONFIGURING CONTROL REGISTER FOR 8255-1
    MOV AL, 10001010B
    OUT CREG1, AL

    ; SELECTING ANALOG I/P CORRESPONDING TO THE CURRENT ITERATION
    MOV AL, [SI]
    OUT PORTA1, AL
    INC SI
    
    ; HANDLING THE ALE AND START PINS OF ADC
    MOV AL, 02H
    OUT PORTC1, AL
    MOV AL, 01H
```

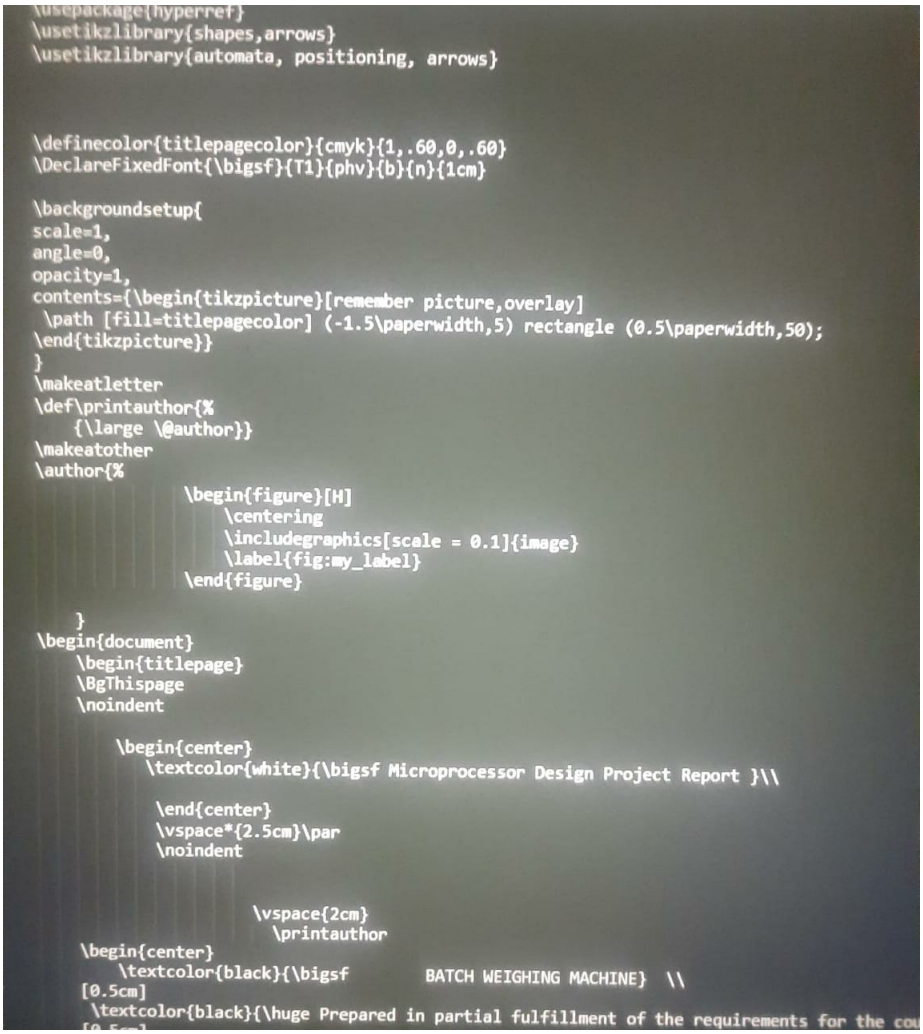
11 Design Presentation

Check the design presentation using this link.

Link: https://github.com/dhamija-sridhar/Batch-Weighing-Machine

12 Report

This report is made after the completion of entire project to portray the project and to help others understand it easily.
The entire report has been made using LATEX Language, and everything is coded in that language.



```
\usepackage{hyperref}
\usetikzlibrary{shapes,arrows}
\usetikzlibrary{automata, positioning, arrows}



\definecolor{titlepagecolor}{cmyk}{1,.60,0,.60}
\DeclareFixedFont{\bigsf}{T1}{phv}{b}{n}{1cm}

\backgroundsetup{
scale=1,
angle=0,
opacity=1,
contents={\begin{tikzpicture}[remember picture,overlay]
 \path [fill=titlepagecolor] (-1.5\paperwidth,5) rectangle (0.5\paperwidth,50);
\end{tikzpicture}}
}
\makeatletter
\def\printauthor{%
    {\large \@author}}
\makeatother
\author{%
                \begin{figure}[H]
                    \centering
                    \includegraphics[scale = 0.1]{image}
                    \label{fig:my_label}
                \end{figure}

    }
\begin{document}
    \begin{titlepage}
    \BgThispage
    \noindent

        \begin{center}
           \textcolor{white}{\bigsf Microprocessor Design Project Report }\\

            \end{center}
            \vspace*{2.5cm}\par
            \noindent


                        \vspace{2cm}
                          \printauthor
    \begin{center}
        \textcolor{black}{\bigsf        BATCH WEIGHING MACHINE}  \\
    [0.5cm]
     \textcolor{black}{\huge Prepared in partial fulfillment of the requirements for the cou
```

13 Scope of Improvement

- The system provides an accuracy just upto two decimal digits. There's a scope of improvement in this.
- The system doesnot automatically check for new weights. User intervention is required whenever a weight is to be measured.
- The least weight that can be measured depends on ratings of load cell and output lines of ADC.
- The alarm can't be stopped automatically. User intervention is required for pressing the stop-alarm switch.
- For same consective reading, the user gets error.

A Links to different datasheets of components used

INA22: http://www.ti.com/lit/ds/symlink/ina122.pdf

ADC0808: http://www.ti.com/lit/ds/symlink/adc0809-n.pdf

8255: http://www.eie.polyu.edu.hk/~enyhchan/c8255.pdf

8086: http://www.datasheetspdf.com/PDF/8086/499305/5

7447: http://www.datasheetcatalog.com/datasheets_pdf/D/M/7/4/DM7447A.shtml

8284: http://www.datasheetspdf.com/datasheet/8284A.html

74138: http://www.ti.com/lit/ds/symlink/sn74ls138.pdf

2732: http://pdf1.alldatasheet.com/datasheet-pdf/view/129050/FAIRCHILD/2732.html

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