DESIGN AND FABRICATION OF A MICROCONTROLLER BASED ELECTRONIC WEIGHING MACHINE IN HIGH MASS REGIME

BY

MUNYAO KITAVI
REG. NO: I56/7254/02

A thesis submitted in partial fulfillment of the requirements for the award of the degree of Master of Science in the School of Pure And Applied Sciences of Kenyatta University

AUGUST 2009

DECLARATION

This Thesis is my original work and has not been presented for the award of a degree at any other University.

MUNYAO KITAVI **SIGN:……DATE:……...**
REG. NO: I56/7254/2002

This Thesis has been submitted for examination with our approval as University Supervisors.

DR. G. A. IBITOLA **SIGN:……DATE:……...**
DEPARTMENT OF PHYSICS,
KENYATTA UNIVERSITY,
P.O. BOX 43844,
NAIROBI, KENYA

DR. A. S. MERENGA **SIGN:……DATE:……...**
DEPARTMENT OF PHYSICS,
KENYATTA UNIVERSITY,
P.O. BOX 43844,
NAIROBI, KENYA

DEDICATION

To Mom and Dad, for showing me the value of hard work, patience and pride in a job well done.

ACKNOWLEDGEMENTS

Thanks are due to the following senior lecturers for reading my proposal and providing numerous valuable suggestions that have contributed to the realization of this research work: Dr. Gilbert A. Ibitola and Dr. A.S. Merenga both from Department of Physics, Kenyatta University. A big thank you is extended to Mr. Kiptui, a lecturer in the same department for the part he played.

Gratitude also goes to the Kenyatta University Physics Laboratory Technicians for their cooperation and support, other lecturers in the department for their criticism at the proposal stage. This, to me, was an eye opener.

A special thank you is extended to Ahmed Ali for providing the software programs that I have used in the hardware and software designs. His support, both moral and technical cannot be quantified.

I am grateful to Kennedy Ogola from KPLC Co. Ltd.; Kenneth Owuor from SkyTech Ltd.; Joseph Kyalo and Muriuki Mwangi from Computer Department-Sunshine Secondary School and David Musyimi of Kenyatta University for their technical assistance and encouragement. I further extend gratitude to Intel Corporation, Atmel Corporation, Signetics (Philips) Corporation, Alpha Group, Inc., Texas Instruments, Inc. for providing data sheets for the components used in this work.

Lastly, I wish to extend my heart felt gratitude to my loving wife, Gladys, and humble children; Shirlene, Samuel and Collins. They, without tiring, stood with and walked beside me all the time.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Electronic weighing technology presents management with rapid, timely and accurate information that provides quick turnaround times for customers. Its popularity can be attested by its use in all the sectors of the industry. In fact, its spectrum of use spans from the traditional retail industry, to manufacturing and warehousing, to postal, health and transport industry. Ordinary microprocessor based weighing systems that utilize single point load cells, have a huge circuitry and hence render them complex to design and implement. They further have low speed and resolution. One other class of microprocessor based weighing balances utilizes faster microprocessors and has relatively high resolution. These devices often have a prohibitive price tag, as they comprise many external devices/chips. Another type of electronic weighing system utilizes a microcontroller and an 8-bit analogue-to-digital converter. It has in-built memory units to store program, data and variables. In both cases, there is compromise between range and resolution. This research work describes the design and implementation of a low-cost, high-resolution and portable digital electronic weighing system ideal for domestic, laboratory and commercial use. The system has highly reduced circuitry as it utilizes a standalone microcontroller chip. It essentially comprises of an 8-bit 8051 microcontroller with a local memory module for storing data from the ADC. High resolution, without compromising range, is achieved by designing a program for interfacing a serial 10-bit ADC to the 8-bit microcontroller. This is due to the in-built Serial Programming Interface (SPI) of the microcontroller used. The developed electronic weighing balance senses, measures and displays the mass, placed on a single point load cell, on an LCD display. The system fabricated measures weights ranging from 0-19 kg.

CHAPTER 1

INTRODUCTION

1.1 Background

Many design applications are based on combinational logic gates and sequential logic ICs. One example is a traffic light controller that goes through the sequence green- yellow-red. To implement the circuit using combinational and sequential logic, some counter ICs for the timing; a shift register for sequencing the lights would be used. A D flip-flop would be used if there is need to interrupt the sequence with a pedestrian cross-walk pushbutton. A complete design solution is easily obtained within the realm of Small Scale Integration (SSI) and Medium Scale Integration (MSI) Integrated Circuits (ICs) [Kleitz, 2003].

However, for complex electronic control applications this approach becomes incapacitated. For example, in a modern automobile, there are several analog quantities to monitor, such as engine speed, manifold pressure, and coolant temperature; and there are several digital control functions to perform, such as spark plug timing, fuel mixture control, and radiator circulation control. The operation is further complicated by the calculations and decisions that have to be made on a continuing basis. A microprocessor-based system would be needed for this application [Ramsay 2000].

A microprocessor-based solution should be considered whenever an application involves making calculations, making decisions based on external stimulus, and maintaining memory of past events. A microprocessor offers several advantages over the “hard-wired” SSI/MSI IC approach.

First, the microprocessor itself is a general-purpose device. It takes a unique personality by the software program instructions given by the designer. Its capacity to perform arithmetic, make comparisons, and update memory makes it a very powerful digital problem solver. Changing a few program instructions, unlike the hard-wired system that may have to be totally redesigned and reconstructed, can usually make changes to an application. Despite all these, microprocessor-based system applications have a number of disadvantages. External support chips are incorporated into most of these applications leading to too much circuitry and cost [Houpis 1992].

This research work was aimed at designing and constructing a high range, high-resolution micro controller-based electronic weighing balance, which can be reprogrammed to perform other different or additional tasks. It is a low-cost prototype solution based on the 8051 micro controller. The resultant weighing balance has, within the microcontroller, 8K bytes of in-system reprogrammable downloadable flash memory for storing the program, 2K bytes of EEPROM memory for data storage and 256 bytes of internal RAM for storing variables [www.atmel.com].

1.2 Problem statement and justification

Most of the existing electronic weighing balances use customized chips and are therefore not flexible. They only perform certain designated tasks. Those that use microprocessors do not have in-built memory modules, hence the need for external support chips. They come with pricey tags. In most microprocessor based weighing balances, there is a compromise between range and resolution. On system reprogramming is not easily done in the case of microprocessor-based systems.

The use of a microcontroller in this research work offers greater flexibility, since bits in memories replace wired connections. Reprogramming the system attains any modification in the hardware design to improve the system.

1.3 Objectives

The general objective of this research work is to design and construct an 8051 Microcontroller based electronic weighing balance. The specific objectives are:

(i) Mount the encapsulated single point load cell.

(ii) Design and build a signal conditioning circuitry for the load cell output.

(iii) Develop a program for interfacing the 10-bit ADC to the 8-bit microcontroller and for implementing the analog-to-digital conversion.

(iv) Develop a program for the LCD display control.

(v) Develop a program for the system using M-IDE Studio for MCS-51 and load it, from a PC, into the micro controller via the PM51 programmer.

(vi) Design schematic, then PCB using Express PCB and finally mount the components.

(vii) Calibrate the system.

(viii) Take measurements using the balance.

1.4 Significance and Anticipated Output

The 8051 micro controller based electronic weighing balance, in this research work, is for measuring and displaying mass on an LCD display. This research work utilizes the technique of interfacing a 10-bit ADC to an 8-bit micro controller. The system is able to sense, measure and display mass ranging from 0 to 19 kg with a sensitivity of 40 g per every volt of excitation voltage.

CHAPTER 2

LITERATURE REVIEW

2.1 Analog Electronic Weighing Balances

Current analog scales employ sensing devices such as transducer load cell or piezoelectric sensors. They do not utilize a digital processor and their readouts to the user are analog hence the name. The scales either have a graduated dial scale and a rotatable pointer or a calibrated rotating dial. Figure 2.1 is a functional block diagram of an analog weighing scale [Mayer, 2000].

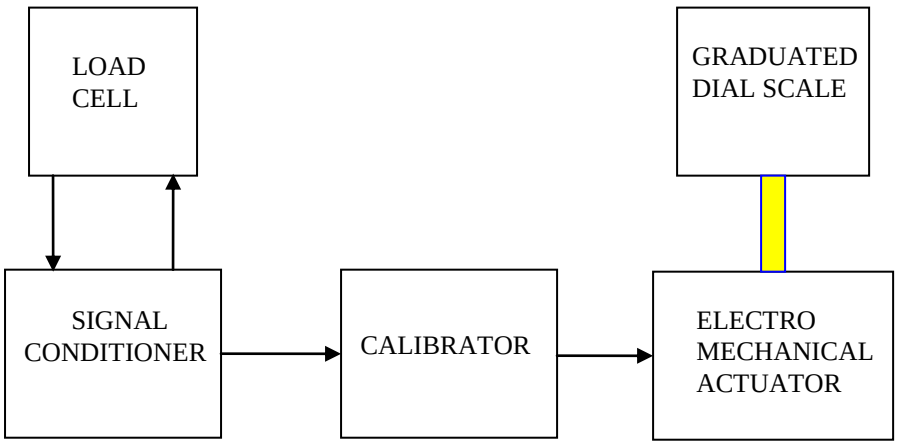


Figure 2.1: Functional Block Diagram of an Analog Scale

This type of weighing balance has a number of disadvantages. Firstly, the dial has a poor resolution in that small differences in weight between different objects are not easily measured or detected. Secondly, the device has no memory thus the weight of a previously measured object is not stored and can only be recalled mentally or by keeping a written record of the measurements. Thirdly, calibration has to be done at the start of every measurement making it tedious and cumbersome. Fourthly, the device has high electrical power consumption particularly due to the electro mechanical actuator that works on heavy current.

Lastly, the efficiency of the machine is appreciably low due to the moving parts of the actuator in addition to the obvious high cost maintenance [Nishiyama, 2001].

The calibrator has, in some cases, been substituted by a digital processor circuitry that generates a digital signal for driving the actuator that in operates the analog dial. Retaining this kind of dial leaves the problem of low resolution unaddressed [Nishiyama, 2001].

2.2 8085A Microprocessor based electronic weighing balance

Figure 2.1 shows the functional block diagram of a microprocessor based electronic weighing balance.

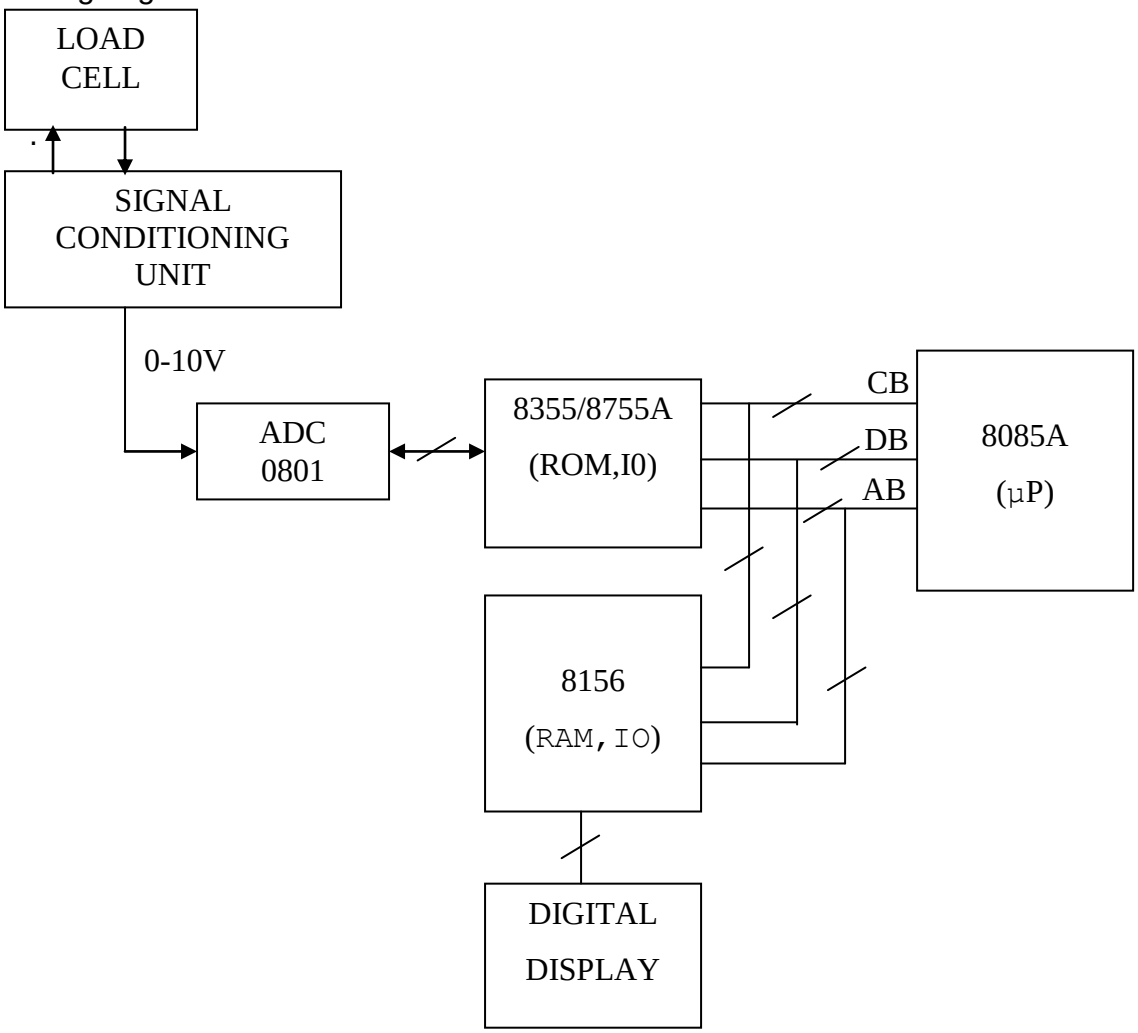


Figure 2.2: Functional block diagram of an 8085A Microprocessor Based Electronic Weighing Balance.

The electronic weighing balance circuit is centered on the three–chip minimum-component 8085A microprocessor system. Two I/O ports on 8355/8755A are used for the A/D hand shaking. The signal conditioning circuitry, which receives the bridge output analogue signal from the load cell, drives the 8-bit ADC. By choosing an appropriate reference voltage; the binary output of the ADC will increase in linear steps numerically equal to the mass placed on the load cell [Usher, 1996].

In order to display the mass as a decimal number, the 8-bit binary output has to be converted to binary-coded decimal (BCD) digits and output them to the digital display. The 8–bit port A on the 8156 is used to drive the digital display. The decoder/driver circuitry accepts BCD data, converts them to seven-segment code, and turns on the appropriate segments. The binary data that the microprocessor receives from the ADC are not in a form that can be used by the display. For example, if the mass is 20 kg, the ADC will output $00010100_{2.}$ This has to be converted to 00100000_{BCD} before being output to the display. This conversion could be done with MSI ICs specifically designed for binary to BCD conversion. Another way is to write a software sub routine to perform the conversion.

A third way is to use a look-up table. The table look-up technique is very useful for nonlinear and other complex data conversion. Its disadvantage is that it uses a lot of very valuable memory to hold the table entries. In this application, since the results are linear and have a one-to-one correlation, a simple software algorithm can be written to do the conversion.

Microprocessor based electronic weighing balances have been observed to have drawbacks such as the low resolution, a lot of space for the many hardware components used, low range, high cost and limited I/O [Kleitz, 2003].

2.3 Intel 8051 Microcontroller Based Electronic Weighing Balance

Most control applications require extensive I/O and need to work with individual bits. The Intel 8051 sufficiently meets both of these needs by having 32 I/O and a CPU instruction set that handles a single–bit I/O, bit manipulation, and bit checking.

Figure 2.3 shows the functional block diagram of an Intel 8051 microcontroller based electronic weighing balance.

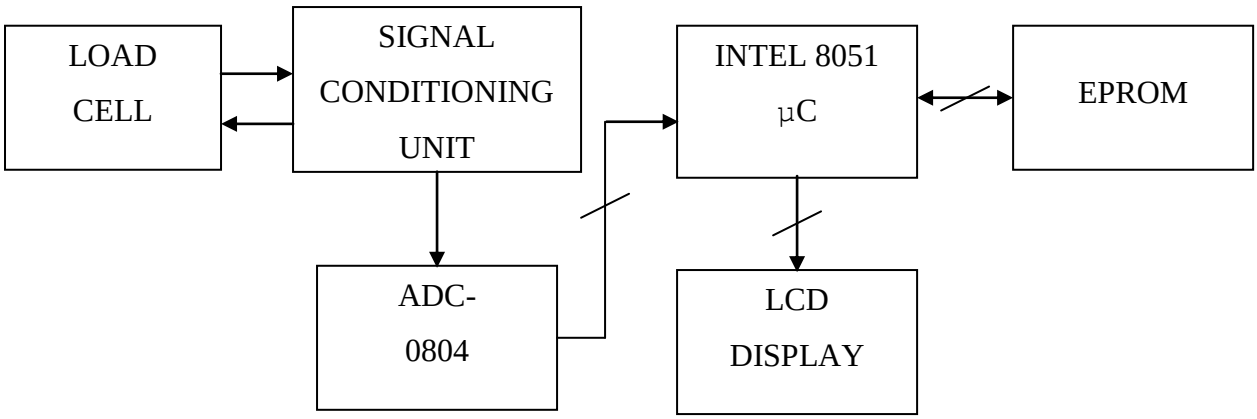


Figure 2.3: Functional Block diagram of the Intel 8051 micro controller–based electronic weighing balance

ADC and display binary coded decimals (BCD) output of the mass. The single point load cell sensor is used to measure the mass. It outputs 2mV for every volt of excitation voltage for a mass of 30kg. This millivolt value is converted into an 8-bit binary string, which is input to the micro controller. The Microcontroller is used to start the ADC conversion (SC) and then monitor the end of conversion (EOC) line to wait for the conversion to be complete. Once the conversion is complete, the micro controller then has to read the digital string and convert the hexadecimal value into a BCD value so that the user on the LCD display can read it.

Although the Intel 8051 micro controller is a better solution to dedicated control applications as compared to the Intel 8085 microprocessor, it has limitations such as low sensitivity, lacks internal EPROM, program memory cannot be reprogrammed, cost still relatively high and lacks an SPI serial interface [Meyerle, 1983].

2.4 Atmel AT89S8252 Microcontroller Based Electronic Weighing Balance

The AT89S8252 Micro controller used in this research is a superior fashion of the Intel 8051 family. It is a low power, high performance CMOS 8-bit micro controller with 8k bytes of downloadable flash programmable and erasable read only memory and 2k bytes of EEPROM. It is compatible with the industry standard 8051 instruction set and pin out. The on-chip downloadable flash allows the program memory to be reprogrammed in system through an SPI serial interface or by a conventional non-volatile memory programmer. By combining a versatile 8-bit CPU with downloadable flash on a monolithic chip, the Atmel AT89S8252 is a powerful microcomputer, which provides a high flexible and cost effective solution to many embedded control applications [Kleitz, 2003]. Figure 2.4 shows the functional block diagram of the Atmel AT89S8252 microcontroller based electronic weighing balance.

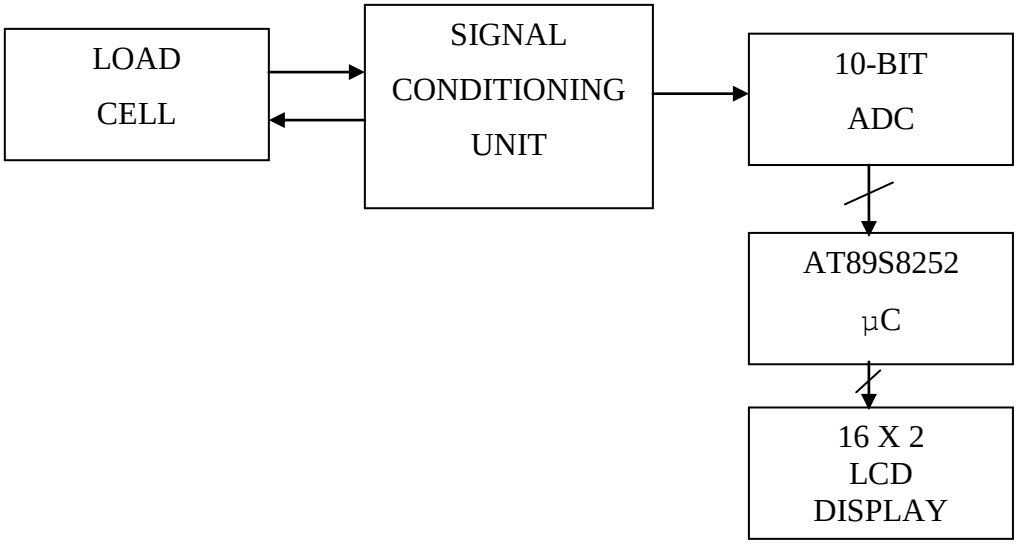


Figure 2.4: Functional block diagram of the Atmel AT89S8252 Microcontroller Based Electronic Weighing Balance

The encapsulated single point load cell has a capacity of 30 kg. The load cell is used for measuring the mass placed on its platter. It outputs 2 mV for every volt of the excitation voltage when the mass is 30 kg.

The instrumentation amplifier is used for driving the 10-bit ADC. The use of the 10–bit ADC provides higher resolution than the case of the 8-bit ADC. The ADC is interface able with the microcontroller because of the Serial Programming Interface (SPI) serial interface within the AT89S8252. The microcontroller reads the mass input to the ADC and displays a binary coded decimal (BCD) output of the mass. The 2 mV value is converted into a 10-bit binary string, which is input to the micro controller.

The microcontroller is used to start the ADC conversion (SC) and then monitor the end of conversion (EOC) line to wait for the conversion to be complete. Once the conversion is complete, the microcontroller then has to read the digital string and convert the hexadecimal value into a BCD value so that the user on the LCD display can read it. This Atmel AT89S8252 microcontroller based electronic weighing balance has the following advantages: high flexibility, low cost, high resolution, low power consumption, wide range, little space and is highly portable [Kleitz, 2003].

CHAPTER 3

THE SYSTEM DESIGNS

3.1 Theory

3.1.1 Strain gauge load cells

A load cell is a package with strain gauge elements and/or resistors in one housing, sealed and encapsulated for protection. Load cells are designed to sense force or weight under a wide range of adverse conditions. They are not only the most essential part of an electronic weighing system, but also the most vulnerable.

Load cell selection in the context of trouble free operation concerns itself primarily with the right capacity, accuracy class and environmental protection, rather than with a particular measuring principle like bending, shear, compression or ring torsion. While saying this, it should be recognized that a particular measuring principle might offer distinct advantages in terms of overload capabilities or the ease of mounting [Usher, 1996].

The sensing or spring element is the main structural component of the load cell. The element is designed in such a way that it develops a strain, directly proportional to the load applied. Sensing elements are normally made of high strength alloy steels (nickel plated for environmental protection), precipitation-hardened stainless steels, heat-treated aluminum alloys, or beryllium copper alloys.

Figure 3.1 shows a foil-type strain gauge

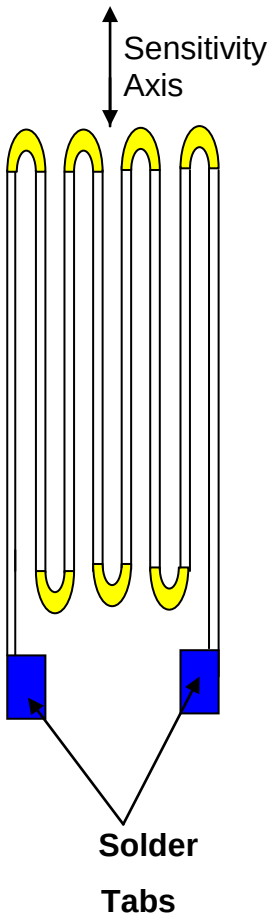


Figure 3.1: A foil-type strain gauge

The gauge in this load cell is the foil type. This is simply a thin electrical conductor that is looped back-and-forth and bonded securely to the piece of material to be strained.

By bonding strain gauges to a precisely machined element, the force applied can be identified in terms of resistance change. The strain gauges, usually four or a multiple of four, are connected into a Wheatstone bridge configuration in order to convert the very small change in resistance into a usable electrical signal. Passive components such as resistors and temperature depending wires are used to compensate and calibrate the bridge output signal [Kleitz, 2003].

3.1.1.1 Bending load cells

Bending load cells, in which sensing elements are subjected to bending, are widely used in many configurations for commercial transducers. They offer high strain levels at relatively low forces.

Furthermore, in case of a beam with a symmetrical cross-section about the bending axis, there are always two surfaces subjected to equal strains of opposite sign. This offers a convenient means of implementing a full bridge circuit and temperature compensation is relatively easy.

The gauges are cemented directly to the surface of the structural member (element) under test so that changes in length are detected by changes in the resistance of the gauge. The magnitude of the change in resistance can be determined from

$$R = \frac{\rho L}{A} \qquad \textbf{(Eq. 3-1)}$$

where *L* is the total length, *A* is the cross-sectional area, and *ρ* is the resistivity of the foil strain gauge.

Taking the logarithm of the strain and the differential of both sides,

log R = log ρ + log L–log A

$$\frac{\Delta R}{R} = \frac{\Delta \rho}{\rho} + \frac{\Delta L}{L} - \frac{\Delta A}{A} \qquad \textbf{(Eq. 3-2)}$$

Introducing Poisson's ratio, *σ*, for the gauge material,

$$\frac{\Delta A}{A} = -\frac{2\sigma \Delta L}{L} \qquad \textbf{(Eq. 3-3)}$$

into equation 3-2 yields,

$$\frac{\Delta R}{R} = \left(\frac{\Delta L}{L}\right)(1 + 2\sigma) + \frac{\Delta \rho}{\rho} \qquad \textbf{(Eq. 3-4)}$$

Finally, the gauge factor $K = \left(\frac{\Delta R}{R}\right)\frac{\Delta L}{L}$ is given by:

$$K = 1 + 2\sigma + \left(\frac{L}{\rho}\right)\left(\frac{\partial \rho}{\partial L}\right) \qquad \textbf{(Eq. 3-5)}$$

According to equation 3-5, the change in resistance arises from changes in the gauge dimensions because of mechanical strain plus the possibility of changes in resistivity. Bridge circuits are universally used to measure the changes in resistance of strain gauges because small resistance changes can be measured and including two identical strain gauges in adjacent arms of the bridge easily effects temperature compensation. One gauge is subjected to the mechanical strain and the other is isolated from the strain but positioned so that its temperature environment is the same as that of the strain detector. Thus, changes in temperature influence each gauge equally and the bridge remains balanced [Brophy, 1990].

Applying a force to the material compresses it slightly, which compresses the strain gauge in the direction of its sensitivity axis. As the strain gauge conductor is compressed, its cross-sectional area increases, length decreases and therefore resistance decreases. This change in resistance is very slight, usually milliohms, and is converted to a voltage and amplified before it is fed to the ADC.

Unlike the Wheatstone and the Kelvin bridges, which provide measurement at a condition of perfect balance and therefore function irrespective of source (or excitation) voltage, the amount of source voltage matters in an unbalanced bridge like the one in this work. Therefore, strain gauge bridges are rated in millivolts of imbalance produced per volt of excitation per unit measure of force. The strain gauge used in this work for measuring mass is rated 2 mV.

This means at exactly 30 kg applied comprehensive force, the bridge will be unbalanced by 2 mV for every volt of excitation voltage [Kleitz, 2003]**.**

3.1.2 Instrumentation amplifier

Many sensors exhibit a change in electrical resistance in response to the quantity they are trying to measure. Some examples include force sensing resistors, which change their resistance when a force is applied, thermistors, which change resistance as a function of temperature, and carbon microphones, which alter their resistance in response to changing acoustical pressure. In all these cases, one must be able to convert the resistance of the device into a usable voltage, which can be read by an analog-to-digital converter (ADC).

There are two ways to convert resistance of a sensor to a voltage. The first, and the simplest way is to apply a voltage to a resistor divider network composed of a reference resistor and the sensor as shown in figure 3.2.

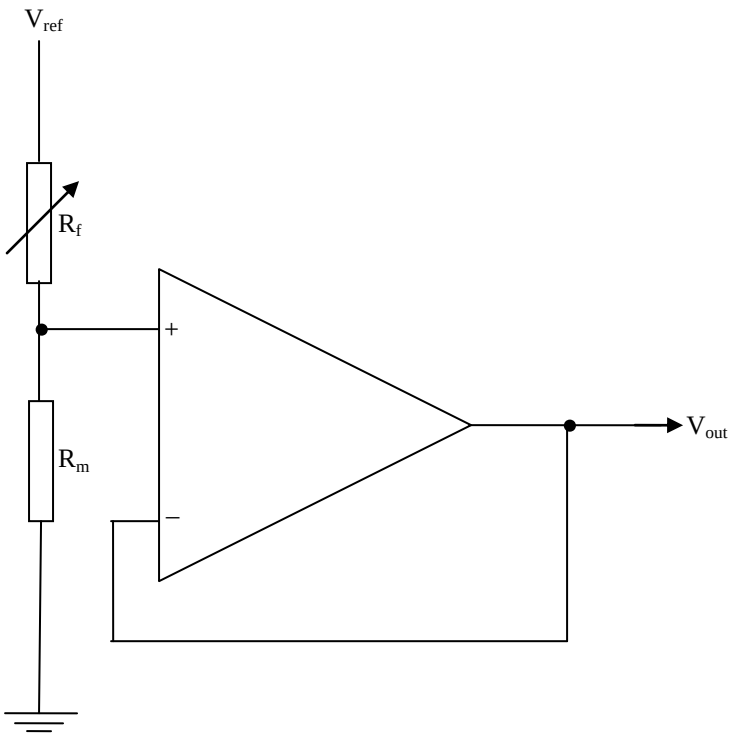


Figure 3.2: Resistance to Voltage Conversion

The voltage that appears across the sensor (or the reference resistor) is then buffered before being sent to the ADC. The output voltage is given by:

$$V_{out} = \left(\frac{R_m}{R_m + R_f} \right) \cdot V_{ex} \qquad \textbf{(Eq. 3-6)}$$

$$\Rightarrow V_{out} = \left\{ \frac{1}{1 + R_f / R_m} \right\} \cdot V_{ex} \qquad \textbf{(Eq. 3-7)}$$

where V_{ex} is excitation voltage, R_m is sensor resistance, V_m is the p.d. across the the sensor and R_f is the reference resistance.

The problem with this method of measuring resistance is that the amplifier is amplifying the entire voltage measured across the sensor. It would be much better to amplify only the change in the resistance of the sensor. This can be accomplished using a bridge circuit and an instrumentation amplifier.

A typical amplifier configuration is shown in figure 3.3. The four gauges G1, G2, G3 and G4 are nominally identical, and of unstrained resistance R. They are connected differentially, so that the equivalent circuit is as shown in figure 3.4.

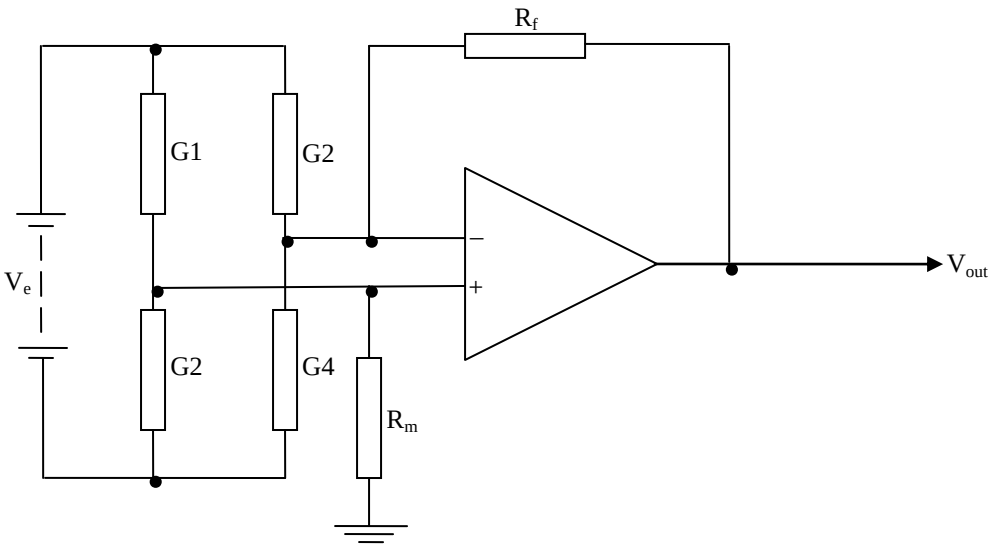


Figure 3.3: Strain Gauge Amplifier

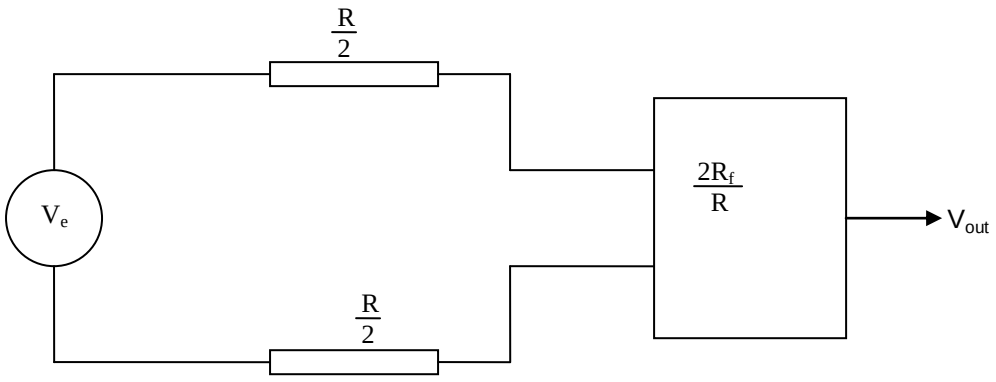


Figure 3.4: Equivalent Circuit for Strain Gauge Amplifier

In this case R is the resistance of the gauges G1, G2, G3 and G4.

The output is directly proportional to the strain since

$$\frac{\Delta R}{R} = K\varepsilon \qquad \textbf{(Eq. 3-9)}$$

It can be seen that $V_{out} = V_{ex} \cdot \left\{ \frac{\Delta R}{R} \left(\frac{Rf}{R/2} \right) \right\}$ hence:

$$V_{out} = V_{ex} \cdot \left\{ \varepsilon \left(\frac{Rf}{R/2} \right) \right\} \cdot K \qquad \textbf{(Eq.3-10)}$$

where ε is the strain and V_{ex} is the excitation voltage.

3.1.3 Analog-to-Digital Converters (ADCs)

Analog-to-digital converters (ADCs) transform analog voltage to a binary number, and then eventually to a digital number (base 10) for reading on a meter, monitor or chart. The number of binary digits (bits) that represents the digital number determines the ADC resolution. However, the digital number is only an approximation of the true value of the analog voltage at a particular instant because the voltage can only be represented (digitally) in discrete steps. How closely the digital number approximates the analog value also depends on the ADC resolution. A mathematical relationship conveniently shows how the number of bits an ADC handles determines its specific theoretical resolution. An n-bit ADC has a resolution of one part in 2^n. For example, a 12-bit ADC has a resolution of one part in 4096, where 2^{12} = 4096.Thus, a 12-bit ADC with a maximum input of 10Vdc can resolve the measurement into 10Vdc/4096 = 0.00244Vdc = 2.44mV. Similarly, for the same 0 to 10-Vdc range, a 16-bit ADC resolution is $10/2^{16}$ = 10/65536 = 0.153 mV. The resolution is usually specified with respect to the full range reading of the ADC, not with respect to the measured value at any particular instant [Brophy, 1990].

There are many types of ADCs. These include:

- Successive-Approximation ADCs
- Voltage-to-frequency ADCs
- Integrating (dual slope) ADCs
- Sigma-Delta ADCs
- Flash converter
- Counter-Ramp ADCs

Since in this thesis, only successive approximation ADC has been used, a brief description of this ADC is provided in the next section.

3.1.3.1 Successive-Approximation ADCs

A successive-approximation converter, shown in figure 3.5, is composed of a digital-to-analog converter (DAC), a single comparator, and some control logic and registers.

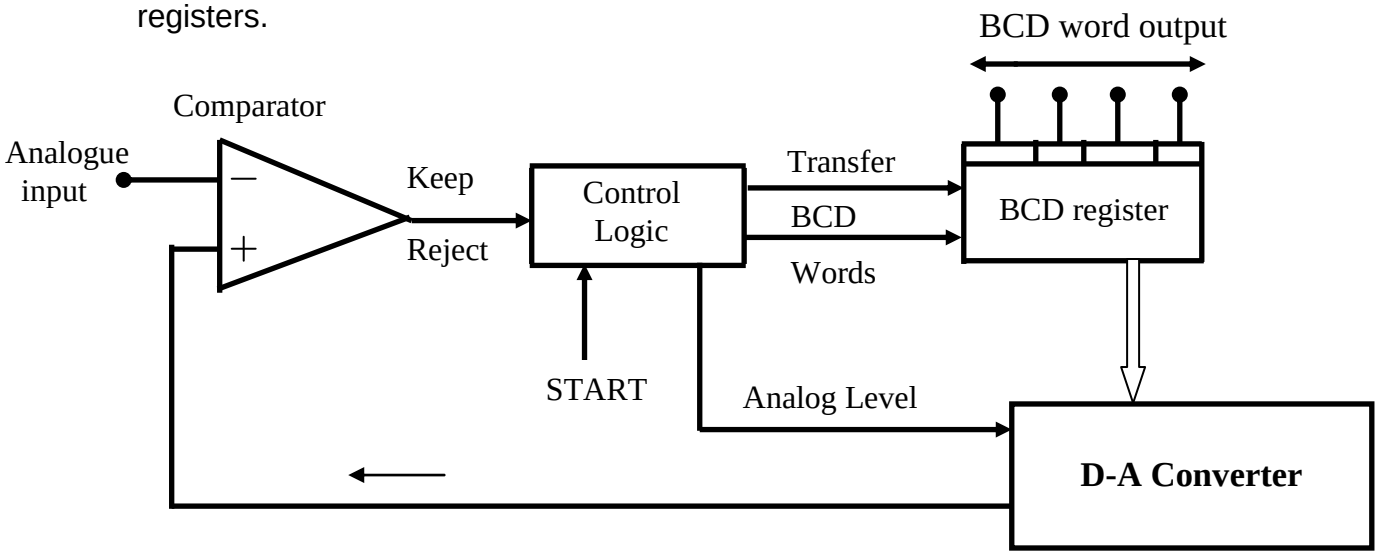


Figure 3.5: Simplified block diagram of successive approximation A-D converter

When the analog voltage to be measured is presented at the input to the comparator, the system control logic initially sets all bits to zero. Then the DAC's most significant bit (MSB) is set to 1, which forces the DAC output to ½ of full scale (in the case of a 10V full scale system, the DAC outputs 5.0 V). The comparator then compares the analog output of the DAC output to the input signal, and if the DAC output is lower than the input signal, (the signal is greater than ½ full scale), the MSB remains set at 1. If the DAC output is higher than the input signal, the MSB resets to zero. Next, the second MSB with a weight of ¼ of full scale turns on (sets to 1) and forces the output of the DAC to either ¾ full scale (if the MSB remained at 1) or ¼ full scale (if the MSB reset to zero).

The comparator once more compares the DAC output to the input signal and the second bit either remains on (set to 1) if the DAC output is lower than the input signal or resets to zero if the DAC output is higher than the input signal. The third MSB is then compared the same way and the process continues in order of descending bit weight until the LSB is compared. At the end of the process, the output register contains the digital code representing the analog input signal [Brophy, 1990]. Successive approximation ADC's are relatively slow because comparisons run serially, and the ADC must pause at each step to set the DAC and wait for its output to settle. However, conversion rates easily can reach over 1 MHZ. The method of narrowing in on the unknown analogue voltage is much improved and circuit is simple [Kleitz, 2003]. Also, fast 12 and 16-bit successive-approximation ADCs are available and relatively inexpensive, which accounts for their wide use in many PC-based data acquisition systems [Tseng, 1982].

3.1.3.2 Accuracy and Resolution of an ADC

Accuracy is one of the most critical factors to consider when specifying an ADC for test and measurement applications. Unfortunately, it's often confused with resolution, and although related, they are distinctly different. The ADC accuracy is the percentage error in the converted value while resolution is the smallest value into which the full range ADC reading is divided. Every ADC measurement contains a variety of unavoidable, independent errors that influence its accuracy. When, σ represents each independent error, the total error can be shown a

$$\sigma_{total} = \sqrt{\sum \sigma^2} \qquad \textbf{(Eq. 3-11)}$$

This equation includes a variety of errors such as sensor anomalies, noise amplifier gain and offset, ADC quantization (resolution) error, and other factors.

Accuracy of a data acquisition (DA) device e.g. an ADC is defined as the sum of three components stated in terms of the reading, range, and the least significant bit (LSB). It is a measure of the extend to which the device is error-free. Generally, accuracy is not constant over an entire measurement range; it varies with the reading magnitude. On the other hand, resolution is defined as the smallest incremental change the ADC can recognize. For example, a 10-bit ADC has a resolution of 1 part in 1024.

3.1.4 The 8051 Microcontroller

3.1.4.1 The Microcontroller and the general purpose Microprocessor

By microprocessor is meant the general purpose microprocessors such as Intel's X86 family (8086, 80286, 80386, up to Pentium Series) or Motorola's 689X0 family (68000, 68020, 68030 e.t.c.). ,

These microprocessors contain no RAM, no ROM, and no I/O ports on the chip itself.

Hence, for this reason they are referred to as general-purpose microprocessors.

A system designed with these general-purpose microprocessors must add external RAM, ROM, I/O ports and timers externally to make them functional. Although the addition of external RAM, ROM and I/O ports makes the system bulky and much more expensive; they have the advantage of versatility such that the designer can decide on the amount of RAM, ROM and I/O ports needed to fit a certain task.

This is not usually the case with microcontrollers. A microcontroller has a CPU (a microprocessor) in addition to a fixed amount of RAM, ROM, I/O ports, and a timer are all embedded onto a single chip. The fixed amount of on chip RAM, ROM and a number of I/O ports makes it ideal for many applications whereby cost and space are critical. In many applications, for example a washing machine, there is no need for the computing power of a high speed processor such as 80486 or Pentium processors.

In numerous applications, the space it takes, the power it consumes, and the price per unit are much more critical considerations than the computing power. These applications most often require some I/O operations to read signals and turn on and off certain bits.

For this reason, some call these micro controllers “kitty-bitty-processors”. Some micro controller manufacturers have gone as far as integrating Analogue-to-Digital Converters and other peripherals into the micro controller [Masidi, 2002].

3.1.4.2 Microcontrollers for embedded systems

Microcontrollers are used in numerous embedded system products. An embedded system uses a microcontroller or a microprocessor to do a single task only. A good example of an embedded system is a printer. It fetches data and prints it.

This kind of single tasking can be contrasted with a multi tasking microprocessor such as a Pentium based PC. A PC can be used for a myriad of applications such as word processors, network servers, multimedia applications and so on. The main reason a PC can run numerous applications is that it has RAM and an operating system that loads the application software into RAM and lets the CPU run it. In an embedded system, there is only one application software that is typically stored or “burned” in ROM. A PC consists of embedded system peripherals such the keyboard, hard-disk controllers, printer, modem, and so on [Predko, 2000].

3.1.4.3 History of the 8051

Micro controllers were introduced when the skills of the semiconductor industry allowed several functions such as CPU, Memory and I/O to be integrated onto one piece of silicon. This in turn reduced the size (and the power consumption) of any microprocessor-based solution.

Because of the low cost and ease of integration within an application, they are used wherever possible to reduce the chip count of a piece of electronics. The word ‘intelligent’ tends to be applied to any device that contains some sort of processing or memory capability.

The intelligence of course belongs to the hardware and software designers who program the devices. Intel introduced the 8051 family in 1980, and from a simple micro controller, it has grown into at least 30 different versions and have been second-sourced by numerous manufacturers. The 8051 was introduced to replace the older 8048 which was originally used a s a keyboard handler for IBM PC’s.

The original product lineup was as follows:

ROM-Based 8051: The program memory is on-chip and is specified by the user. The manufacturer mask-programs the silicon disc so that the CPU can only execute the one program.

EPROM-Based 8751: This variant is used mostly for development or very small production runs. The advantage is that the memory may be erased and ‘burned’ again.

ROM-less 8031: This chip needs an external memory (EPROM) to hold the program. To access the memory, the 8031 has to use some of its input/output (I/O) ports.

This means that an 8031-Based Micro controller has only 14 I/O lines left to directly interface to the outside world. When referring to the family, most engineers simply refer to the 8051. It was introduced at a time when a single +5V supply was a pleasant relief to digital system engineers. The original 40-pin DIL package was easily configured and the availability of 4 X 8 ports was pure luxury.

The basic device has the following features:

128 bytes of on-chip RAM (expandable to 64 kB externally), 32 I/O lines, two 16-bit timers, six sources of program interrupt and Full duplex Serial port (UART).

Of the leading 8-bit micro controllers, the 8051 family has the largest number of diversified (multiple source) suppliers. Nowadays, there are quite a number of companies that manufacture the MCS-51 based family of micro controllers. These include: Atmel, Philips/Signetics, AMD, Siemens and Dallas Semiconductor.

3.1.4.4 Internal components of the 8051 microcontroller

The generic MCS-51 IC is the 8051 and consists of the following internal components: 4kB ROM, 128 bytes RAM, four 8-Bit I/O ports, two 16-Bit timers, full duplex serial Interface, 64kB external code memory Space, 64kB external data, memory space, Boolean processor which is one bit, 210 bit addressable locations, and 4 µs multiply/divide for a 12-MHz crystal.

Figure 3.6 shows block diagram of the internal hardware architecture of the 8051.

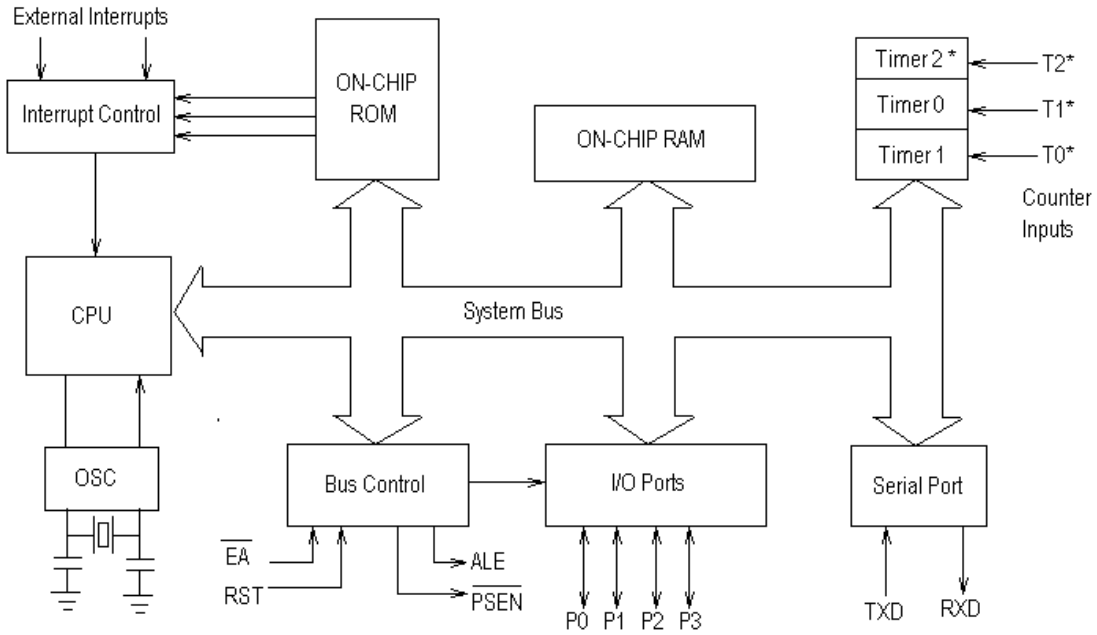


Figure 3.6: Basic Internal Architecture of the 8051 Microcontroller [Mackenzie, 1999]

3.1.4.5 Hardware Summary (Around the Pins)

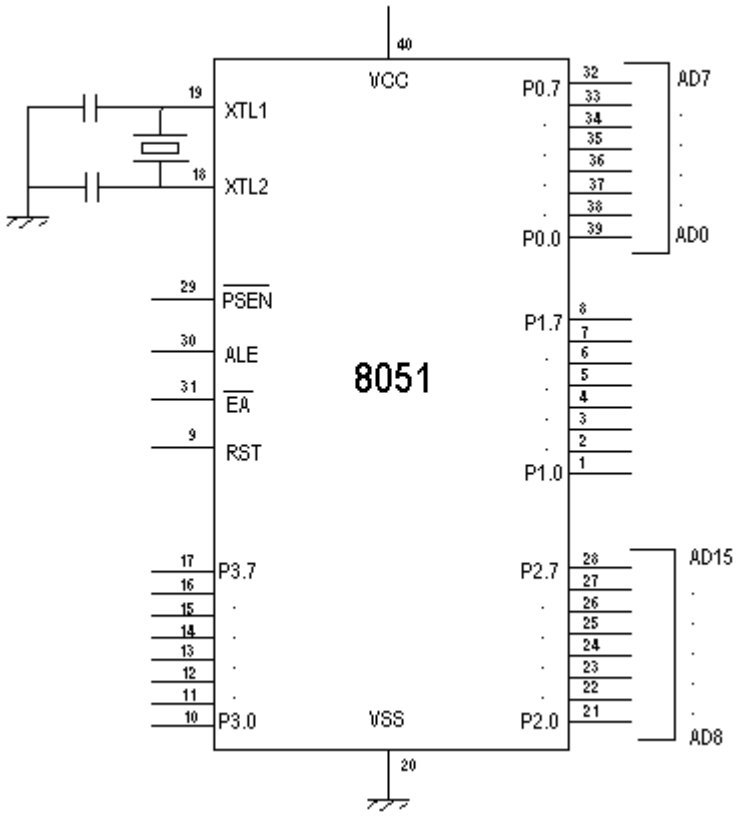


Figure 3.7: The 8051 Pin-Outs [Mackenzie, 1999]

The Intel 8051 is a 40-pin integrated circuit (IC) chip. The four I/O ports require a total of thirty-two pins. In order to provide for the other microcontroller control signals, most of those pins have alternate functions. The 8051 is able to handle the overlapping address spaces used by internal memory, external memory, and the special function registers.

3.1.4.5.1 Port 0

Port 0 is a dual-purpose, serving as either an 8-bit bidirectional I/O port on (P0.0-P0.7) or the low-order multiplexed address/data bus (AD_0-AD_7). When used as I/O port, it sink up to 8 LS TTL loads in the LOW condition and is a float for the HIGH condition. The alternate port designations, AD_0-AD_7, are used to access external memory. The AD lines are demultiplexed into A_0-A_7 and D_0-D_7 by using the ALE signal.

3.1.4.5.2 Port 1

Port 1 is an 8-bit bidirectional I/O port on (P1.0-P1.7) and can sink or source up to 4 LS TTL loads. No alternate functions are assigned for Port 1 pins; thus they are used solely for interfacing to external devices.

3.1.4.5.3 Port 2

Port 2 is a dual purpose port serving as either an 8-bit bidirectional I/O port (P2.0-P2.7), or as the high-order address bus (A_8-A_{15}) for access to external code memory. As an I/O port it can sink or source up to 4 LS TTL loads.

3.1.4.5.4 Port 3

Port 3 is a dual purpose port (P3.0-P3.7), serving as an 8-bit bidirectional I/O port that can sink or source up to 4 LS TTL loads or as special-purpose I/O to provide the functions listed in Table 3.1.

Table 3.1: The alternative functions of port 3.

BIT	NAME	BIT ADDRESS	ALTERNATE FUNCTION
P3.0	RXD	B0H	Receive Data for serial Port
P3.1	TXD	B1H	Transmit Data for Serial Port
P3.2	/INTO	B2H	External Interrupt 0
P3.3	/INT1	B3H	External Interrupt 1
P3.4	T0	B4H	Timer/Counter 0 External Input
P3.5	T1	B5H	Timer/Counter 1 External Input
P3.6	/WR	B6H	External data memory WR strobe
P3.7	/RD	B7H	External data memory RD strobe

3.1.4.5.5 (PSEN)'

The Program Store Enable is read strobe for external program memory which is connected to the Output Enable (OE)' of the external ROM or EPROM.

3.1.4.5.6 ALE

Address Latch Enable output pulse for latching the low-order byte of the address during accesses to external memory. It is also the program pulse input (PROG)' during programming of the EPROM versions of the 8051 and the FLASH based microcontrollers e.g. The Atmel AT89C51.

3.1.4.5.7 (EA)'

The External Access is tied LOW to enable the microcontroller to fetch its program code from an external memory IC. It also receives the 21-V programming supply voltage (VPP) for programming the EPROM parts.

3.1.4.5.8 RST

The Reset input on pin 9 resets the microcontroller when active HIGH. For normal operation, RST is low.

3.1.4.5.9 XTAL1, XTAL2

XTAL1 is the input part of internal oscillator. It is used for synchronizing the MCU with another circuit. Input of the internal oscillator connects to XTAL2.When an external oscillator is used, XTAL2 is without a function.

3.1.4.5.10 Vcc and GND

These are the +5V supply (Vcc = Pin 40) and Ground pin (GND = Pin 20) respectively.

3.1.4.5.11 I/O Port Structure

The internal circuitry for the port pins is shown in figure 3.8

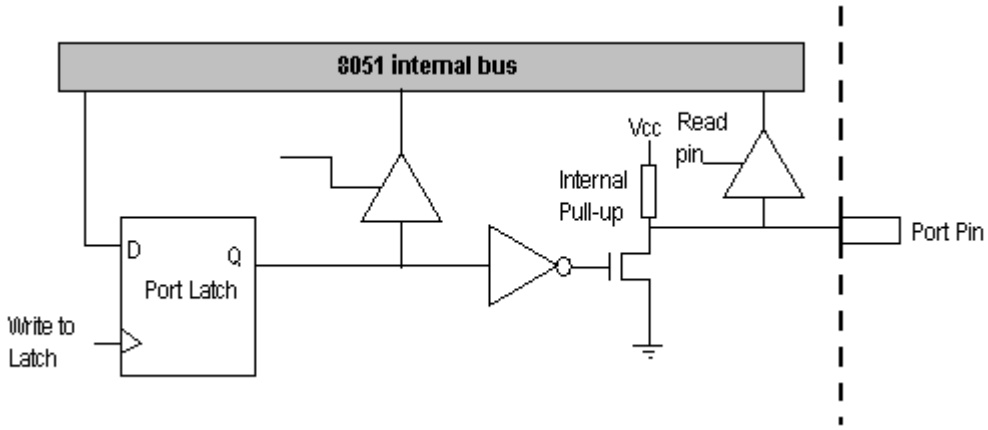


Figure 3.8: Circuitry for I/O Ports

Data is loaded into a port latch when a port pin is written to. The latch drives a field-effect transistor connected to the port pin. The drive capabilities are low-power Skottky loads for ports 1, 2 and 3, and eight LS loads for port 0. When port 0 is functioning as the external address/data bus, the pull-up resistor is present, otherwise it is absent. Depending on the input characteristics of the device being driven by the port, an external pull-up resistor may be needed. There is both a ‘read latch’ and ‘read pin’ capability. A read-modify-write operation instruction (e.g. CPL P1.3) reads the latch to avoid misinterpreting the voltage level. This happens when the pin is heavily loaded like when driving the base of a transistor. Instructions that input a port bit (e.g., MOV C, P1.3) read the pin. The port latch must contain a 1; in this case, otherwise the FET driver is ON and pulls the output low.

A system reset sets all port latches, so port pins may be used as inputs without explicitly setting the port latches. However, if a port latch is cleared (e.g., CLR P1.3), then it cannot function as an input unless the latch is set first (i.e., SETB P1.3).

3.1.4.6 Memory Organization of the 8051

All 80C51 devices have separate address spaces for program and data memory. The logical separation of program and data memory allows the data memory to be accessed by 8-bit addresses, which can be quickly stored and manipulated by an 8-bit CPU. Nevertheless, 16-bit data memory addresses can also be generated through the DPTR register. Program memory (ROM, EPROM) can only be read, not written to. There can be 64 k bytes of program memory. In the 89S51, the lowest 4K bytes of program are on-chip. In the ROMless, versions, all program memory is external. Data Memory (RAM) occupies a separate address space from Program Memory. In the 80C51, the lowest 128 bytes of data memory are on-chip. Up to 64K bytes of external RAM can be addressed in the external Data Memory. In the ROMless version, the lowest 128 bytes are on-chip.

Table 3.2 summarizes the 8051 family of microcontrollers.

Table 3.2: Comparison of MCS-51 ICs

Device number	Internal memory		Timers/event counters	Interrupt sources
	Program	Data		
8051	4k x 8 ROM	128 x 8 RAM	2 x 16-bit	5
8751H	4k x 8 EPROM	128 x 8 RAM	2 x 16-bit	5
8031	None	128 x 8 RAM	2 x 16-bit	5
8052AH	8k x 8 ROM	256 x 8 RAM	3 x 16-bit	6
8752BH	8k x 8 EPROM	256 x 8 RAM	3 x 16-bit	6
8032AH	None	256 x 8 RAM	3 x 16-bit	6

The on-chip RAM contains a rich arrangement of general-purpose storage, bit-addressable storage, register banks, and special function registers. Two notable features are:

a) The registers and I/O ports are memory mapped and accessible like any other memory location, and

b) The stack resides within the internal RAM, rather than in external RAM as typical with microprocessors.

3.1.5 Liquid crystal displays (LCDs)

An LCD is based on the property of nematics, smectics or cholesterics that alter the polarization of transmitted light in response to an applied electric field.

A liquid crystal is composed of large elongated molecules that are organized in an ordered structure, which rotates the plane of polarization of light energy. In an electric field, the molecules line up parallel to the field and the rotation effect is lost. This unique property is used in a liquid crystal display or LCD, by placing a liquid crystal layer between two glass plates each coated with transparent conducting electrodes (as in, for example, the seven-segment pattern) and two polarizing layers oriented such that their polarizations are at right angles. The side away from the viewing side is also provided with a mirrored surface.

Figure 3.9 shows the structure of a liquid crystal display.

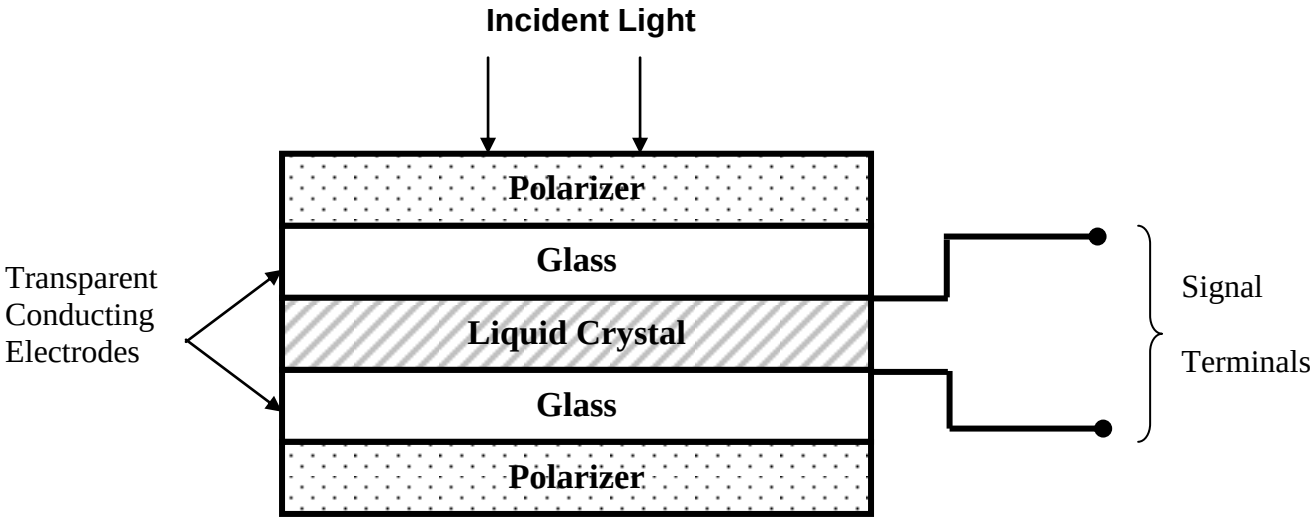


Figure 3.9: Structure of a Liquid Crystal Display

In the absence of an electric field, incident light is polarized by the top polarizer and then rotated at 90^{o} by the liquid crystal so that the light passes through the bottom polarizer. Upon reflection, the light energy passes back up through the stack in reverse order and emerges from the top polarizer.

When a voltage signal is applied to the conducting electrode pattern, polarization rotation does not occur in the liquid crystal layer and the closed top and bottom polarizers absorb the light to make the conducting electrode area appear dark against the surrounding light background. The principal advantage of the LCD is that very little electric power is required to activate the display since the electrode pattern is essentially a small capacitor with liquid crystal as dielectric [Brophy, 1990].

Frequently, an 8051 program must interact with the outside world using input and output devices that communicate directly with a human being. One of the most common devices attached to an 8051 is an LCD display. Some of the most common LCDs connected to the 8051 are 16x2 and 20x2 displays. This means 16 characters per line by 2 lines and 20 characters per line by 2 lines, respectively.

Fortunately, a very popular standard exists which allows us to communicate with the vast majority of LCDs regardless of their manufacturer. The standard is referred to as HD44780U, which refers to the controller chip, which receives data from an external source (in this case, the 8051) and communicates directly with the LCD.

In recent years, the LCD is finding widespread use replacing LEDs (seven segment displays). This is due to the following reasons;

- The declining prices of LCDs.
- The ability to display numbers, characters, and graphics. This is in contrast with LEDs, which are limited to numbers and a few characters.
- Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD. In contrast, LEDs must be refreshed by the CPU (or in some other way) to keep displaying the data.

- The ease of programming for characters and graphics.

3.1.5.1 LCD Pin Descriptions

The LCD discussed in this section has 16-Pins.The function of each pin is given in Table 3.3.

Table 3.3: Summary of the 16X2 LCD pin-outs

PIN	SYMBOL	I/O	DESCRIPTION
1	Vss	--	Ground
2	Vcc	--	+5V Supply
3	Vee	--	Contrast
4	RS	I	0=Command Register, 1=data Register
5	R/W	I	0=Write, 1=Read
6	E	I	Enable
7	DB0	I/O	Data bus
8	DB1	I/O	Data bus
9	DB2	I/O	Data bus
10	DB3	I/O	Data bus
11	DB4	I/O	Data bus
12	DB5	I/O	Data bus
13	DB6	I/O	Data bus
14	DB7	I/O	Data bus
15	BL+	--	Back Light +Supply
16	BL-	--	Back Light –Supply

3.1.5.1.1 V_{ss}, V_{cc} and V_{ee}

While V_{ss} and V_{cc} are used for ground and +5V supply respectively, V_{ee} is used for controlling the contrast of the display.

3.1.5.1.2 Register Select- RS

There are two important registers within the LCD. The RS pin is used for their selection as follows. If RS = 0, then the instruction command mode is selected, allowing the user to send a command such as clearing the display, cursor at home, etc. If RS = 1, then the data register is selected, allowing the user to send data to be displayed on the screen.

3.1.5.1.3 Read/Write- R/(W)'

The R/(W)´ input allows the user to write information to the LCD or read information from it. R/(W)´ = 1 when reading; R/(W)´ = 0 when writing.

3.1.5.1.4 Enable- E

The enable pin is used by the LCD to latch information presented to its data pins. When data is supplied to data pins, a high-to-low pulse must be applied to this pin in order for the LCD to latch in the data presented at the data pins. This pulse must be a minimum of 450 ns wide.

3.1.5.1.5 Data lines-D_0–D_7

The 8- data pins are used to send information to the LCD or read the contents of the LCD's internal registers. To display characters, ASCII codes are sent to these pins while making RS = 1.

There are also commands that are sent to the data bus such as to clear the display or force the cursor home. Table 3.4 lists these instruction commands.

Table 3.4: LCD command codes

HEX CODE	COMMAND TO LCD
1	Clear Display screen
2	Return home
4	Shift cursor left
6	Shift cursor right
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display off, cursor blinking
F	Shift cursor position to left
10	Shift cursor position to right
14	Shift entire display to the left
18	Shift entire display to the right
1C	Force cursor to first line
80	Force cursor to second line
38	2 line and 5X7 matrix

The programs for interfacing the ADC to the microcontroller and LCD control are in assembly language. A brief overview of programming languages is presented in the next section.

3.1.6 Programming Languages

Computer software is generally divided into two broad categories, i.e., operating system software and application (user) software. Operating system software is the collection of programs, which are needed in the creation, preparation, and execution of other programs. Application software consists of those programs generated by the various users of the system in their attempt to apply the computer to solving their problems.

There are three levels of programming, namely: Machine Language, Assembly Language and High Level Language.

3.1.6.1 Machine Language

Machine language is the final step in creating an executable program for the microprocessor/micro controller. In this step, we must determine the actual binary codes that will be stored in memory to be read by the microprocessor/microcontroller. First, we have to determine what memory locations will be used for our program. This depends on the memory map assignments made in the system hardware design. The first step in a hand assembly is to determine the code for MVI A in the case of the 8085 microprocessor or, MOV A in case of the 8051 Microcontroller. This is known as the opcode (Operation Code) and is found in the microprocessor/micro controller Assembly Language Reference chart.

Instructions for storing the program into memory are given by the manufacturer of the microprocessor/micro controller. Programming in machine language is very tedious and hence the introduction of assembly language described in the next section.

3.1.6.2 Assembly Language

Assembly language is written using mnemonics: MOV, JNZ, ADD, DIV, etc. The term mnemonics is defined as “abbreviations used to assist the memory”.

The mnemonics, MOV, stands for “move”. The instruction MOV A, 20H moves the contents of RAM location 20H to the accumulator. If an assembler software package is used, instead of hand assembly, then the machine code that is generated is usually saved on a computer disk or used to program an EPROM. The EPROM is placed in a custom microprocessor/micro controller hardware design.

The mnemonic source program prepared by the programmer using the symbolic instruction set appropriate to a given microprocessor/micro controller is translated by its assembler into a corresponding machine language object program which is subsequently executed. Practical assemblers provide error messages, which are very helpful to the programmer in preparing new routines. This level of programming is very efficient in the use of computer memory space and in the total time required to execute the program.

In a “structured programming” technique, the program, written in assembly language, is broken into modules, or subroutines. Each module has a specific function and can be written and tested on its own. This is very helpful for program development and debugging. Each program is entered in a different block of memory.

3.1.6.3 High-Level Language

Because each detailed step in the program must be included, programming in assembly language is also tedious.

Higher-level languages have been developed in which single statements can be translated into large groups of machine language instructions. In this case, the translating program is called a compiler because, in effect, the computer compiles its own program following quite general instructions. It is much simpler to program in High-level Language than either in assembly language or machine language. It often turns out, however, that a machine language program compiled in this fashion is inefficient in the use of computer memory space or in the total time required to execute the program. This results from the generalities inherent in a compiler. Thus while FORTRAN, BASIC, Pascal and other high-level languages are widely used, assembly languages are necessary to realize the maximum capabilities of any computer [Brophy, 1990].

CHAPTER 4

RESEARCH METHODOLOGY

A study of the operation of a data acquisition system, similar to the one in this research, was carried out. This system is described in section 4.1

4.1 Data Acquisition System

The computerized acquisition of analog quantities is becoming more important than ever in today's automated world. Computer systems are capable of scanning several analog in-puts on a particular schedule to monitor critical quantities and acquire data for future recall. A typical eight channel computerized data acquisition system (DAS) is shown in figure 4.1. The entire system communicates via two common buses, the data bus and the control bus. The data bus determines the information carrying capability of a particular processor or microprocessor based system. In this case, there are three devices on the data bus: the analogue-to-digital converter (ADC), the micro-processor/microcontroller, and memory. The control bus passes control signals to and from the various devices for such things as chip select (CS)', output enables (RD)', system clock, triggers, and selects. Each of the eight transducers is set up to output a voltage that is proportional to the analog quantity being measured. The task of the microprocessor/microcontroller is to scan all the quantities at some precise interval and store the digital results in memory for future use. To do this, the microprocessor must enable and send the proper control signals to each of the devices, in order, starting with the multiplexer and ending with the ADC.

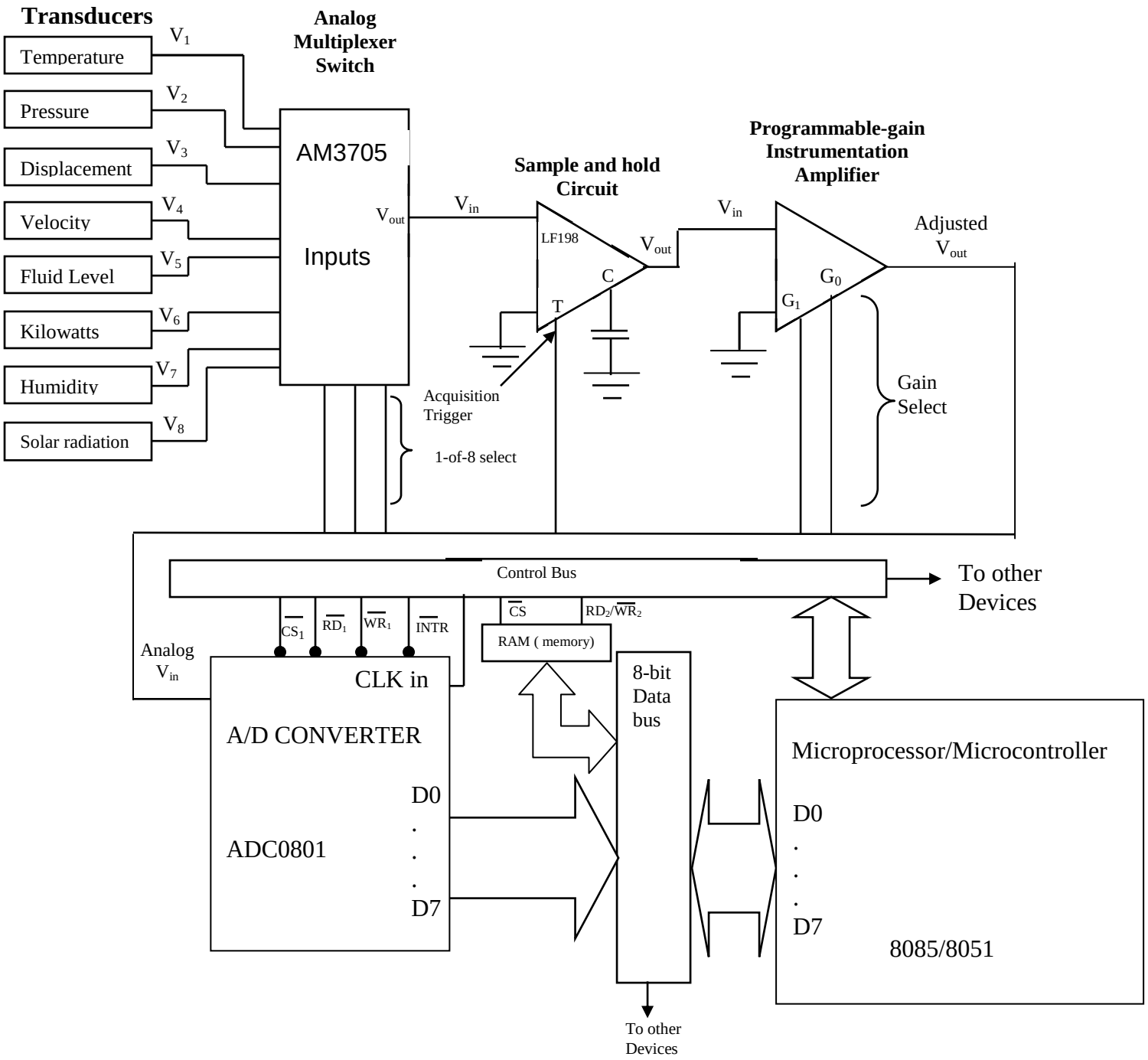


Figure 4.1: Data Acquisition System

4.1.1 Analog Multiplexer Switch (AM3705)

The multiplexer reduces circuit complexity and eliminates duplication of circuitry by allowing each of the eight-transducer outputs to take turns traveling through the other devices.

The microprocessor selects each of the transducers at the appropriate time by setting up the appropriate binary select code on the A, B, C inputs via the control bus. That allows the selected transducer signal to pass through to the next device.

4.1.2 Sample-and-Hold Circuit (LF198)

Since analog quantities can be constantly varying, it is important to be able to select a precise time to take the measurement. The sample-and-hold circuit, with its external hold capacitor, allows the system to take (sample) and hold an analog value at the precise instant that the microprocessor issues the acquisition trigger.

4.1.3 Programmable-Gain Instrumentation Amplifier (LH0084)

Each of the eight transducers has different full-scale output ratings. For instance, the temperature transducer may output in the range 0 to 5 V while the pressure transducer may only output 0 to 500 mV. The LH0084 is capable of being programmed via the gain select inputs, for gains of 1, 2, 5 or 10. When it is time to read the pressure transducer, the microprocessor/micro controller will program the gain for 10 so that the range will be 0 to 5 V, to match that of the other transducers. That way the ADC can always operate in its most accurate range 0 to 5 V.

4.1.4 Analog-to-Digital Converter (ADC0801)

The ADC receives the adjusted analog voltage and converts it to an equivalent 8-bit binary string. To do that, the microprocessor/microcontroller issues chip select (CS)' and start conversion ((WR)' or STC) pulses. When the end-of-conversion (INTR or EOC) lines go LOW, the microprocessor/micro controller issues an output enable (RD or OE) to read the data (D_0 to D_7) that pass, via the data bus, into the microprocessor/microcontroller and then the random-access-memory (RAM) chip.

This cycle repeats for all eight transducers whenever the microprocessor determines that it is time for the next scan. Other software routines executed by the microprocessor/micro controller will act on the data that have been gathered. Some possible responses to the measured results might be to sound an alarm, speed up a fan, reduce energy consumption, increase a fluid level, or simply produce a tabular report on the measured quantities.

Work on microcontroller based applications is vast. This includes: 8051 microcontroller based: digital centigrade thermometer, temperature-dependent pulse–width–modulated (PWM) speed control application, integrating solar radiometer and portable adapter for barcode scanners [Kleitz 2003].

The hardware components for the microcontroller based electronic weighing balance were selected on the basis of logic family compatibility, current and voltage ratings, and operating temperature. Therefore, the major components used in this research work are: Single Point Load Cell, INA125 instrumentation amplifier, TLC1549 10-bit ADC, AT89S8252 µController and a 16x2 Liquid Crystal Display (LCD). The design, fabrication, programming and testing of the system was done as described in the following sections.

4.2 Mounting the single point load cell

The sensing element was mechanically mounted in such a way that it is subjected to bending moment when loaded. In this position, it acts as a bending beam, which offers high strain levels at relatively low forces.

Figure 4.2 shows how the load cell was mounted.

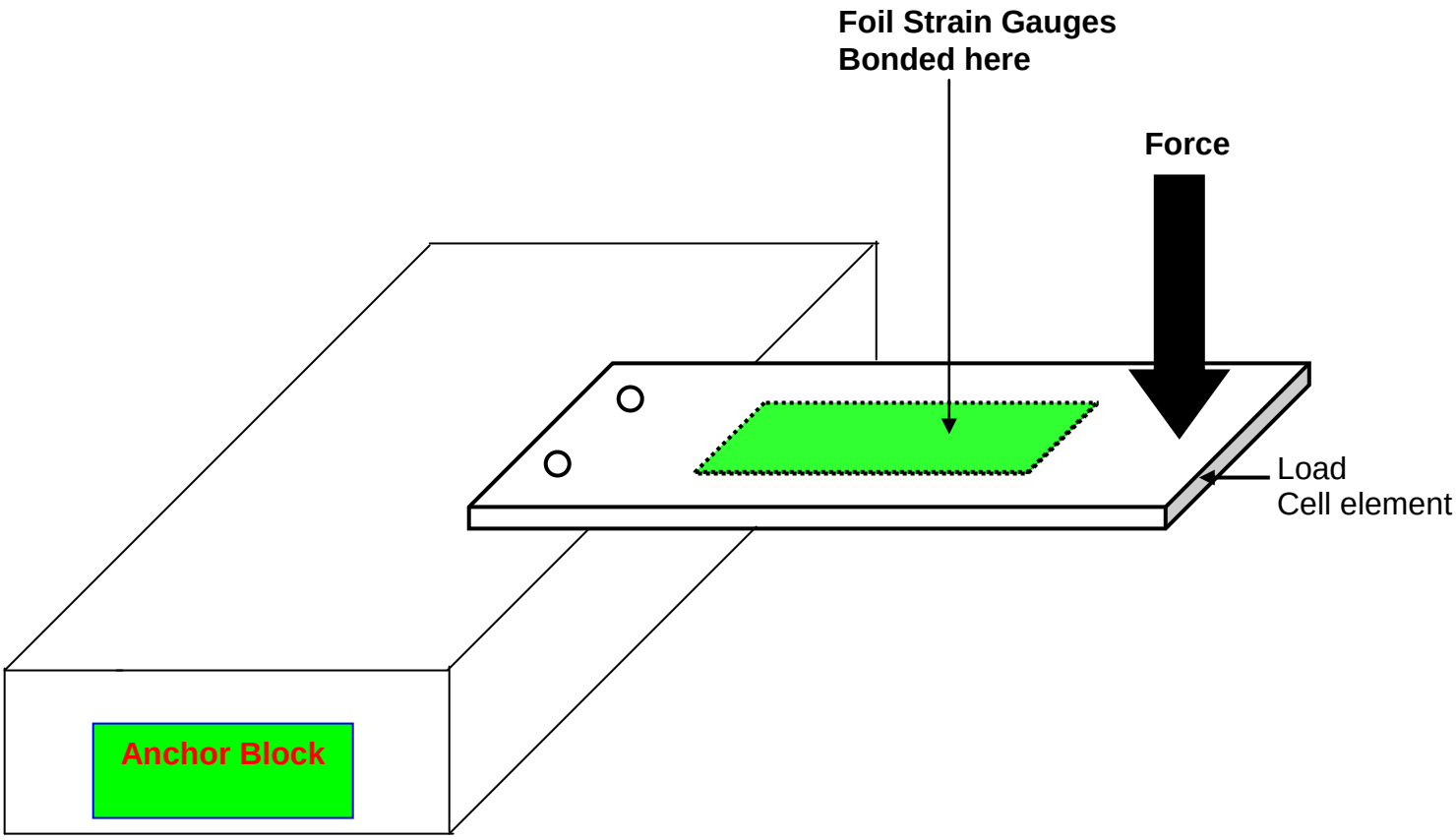


Figure 4.2: Principle used in mounting the Load Cell

4.3 Designing the signal conditioning circuit

The signal conditioning circuit consists of the instrumentation amplifier INA125 IC, a 0.1 µF capacitor, a 10 kΩ potentiometer and NPN signal transistor connected externally. The INA125 is a low-power, high-accuracy instrumentation amplifier with a precision voltage reference. It provides complete bridge excitation and precision differential-input amplification on a single integrated circuit. A single external resistor sets any gain from 4 to 10,000. The INA 125 is laser-trimmed for low offset voltage (250 µV), low offset drift (2 µV/ ^{o}C), and high common-mode rejection (100dB at G = 100). It operates on single (+2.7 V to + 36 V) or dual (±1.35 V to ± 18 V) supplies.

The voltage reference is externally adjustable with pin-selectable voltages of 2.5 V, 5 V, or 10 V, allowing use with a variety of transducers.

The reference voltage is accurate ±0.5% (max) with ±35ppm/^{o} drift (max). Sleep mode allows shutdown and duty cycle operation to save power. The INA125 is available in 16-pin plastic DIP and SO-16 surface-mount packages.

Some other areas of application of the device are:

- Pressure and temperature bridge amplifiers
- Industrial process control
- Factory automation
- Multi-channel data acquisition
- Battery operated systems
- General-purpose instrumentation [INA125P data sheet].

The signal conditioning circuitry in this research work was designed, using ExpressPCB, tested and then implemented on a PCB board. After implementing the signal conditioning circuit on a PCB, it was interfaced to the load cell. The 10 kΩ potentiometer was adjusted to give an output of 0 V at zero load (1.2 kg dead load), and 5 V at a load of 19 kg.

Figure 4.3 shows the signal conditioner circuit schematic.

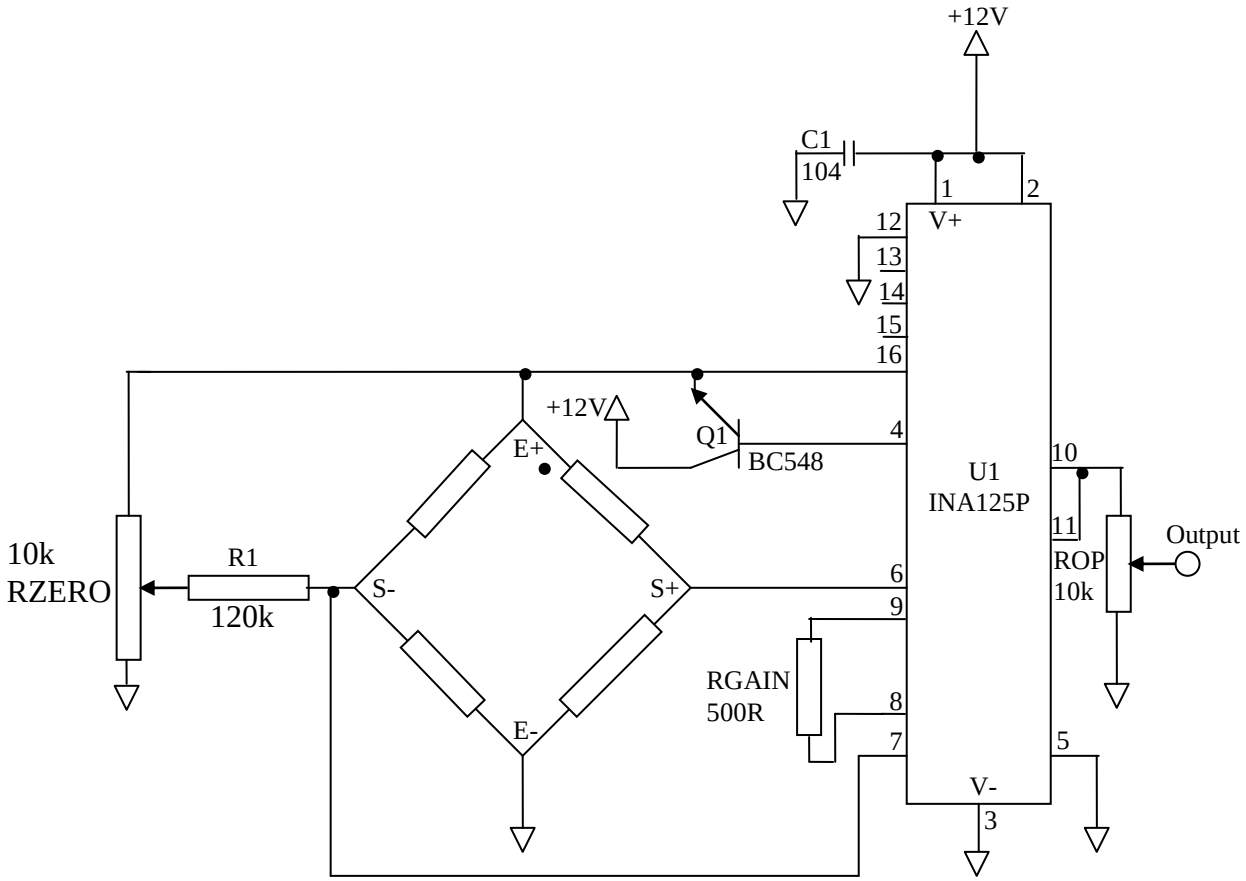


Figure 4.3: Signal Conditioner Circuit Schematic diagram

4.4 Designing the microcontroller circuitry

The schematic diagram for the microcontroller circuit was designed using ExpressSCH Program and the component layout begun by running ExpressPCB.

Figure 4.4 shows the digital circuit schematic.

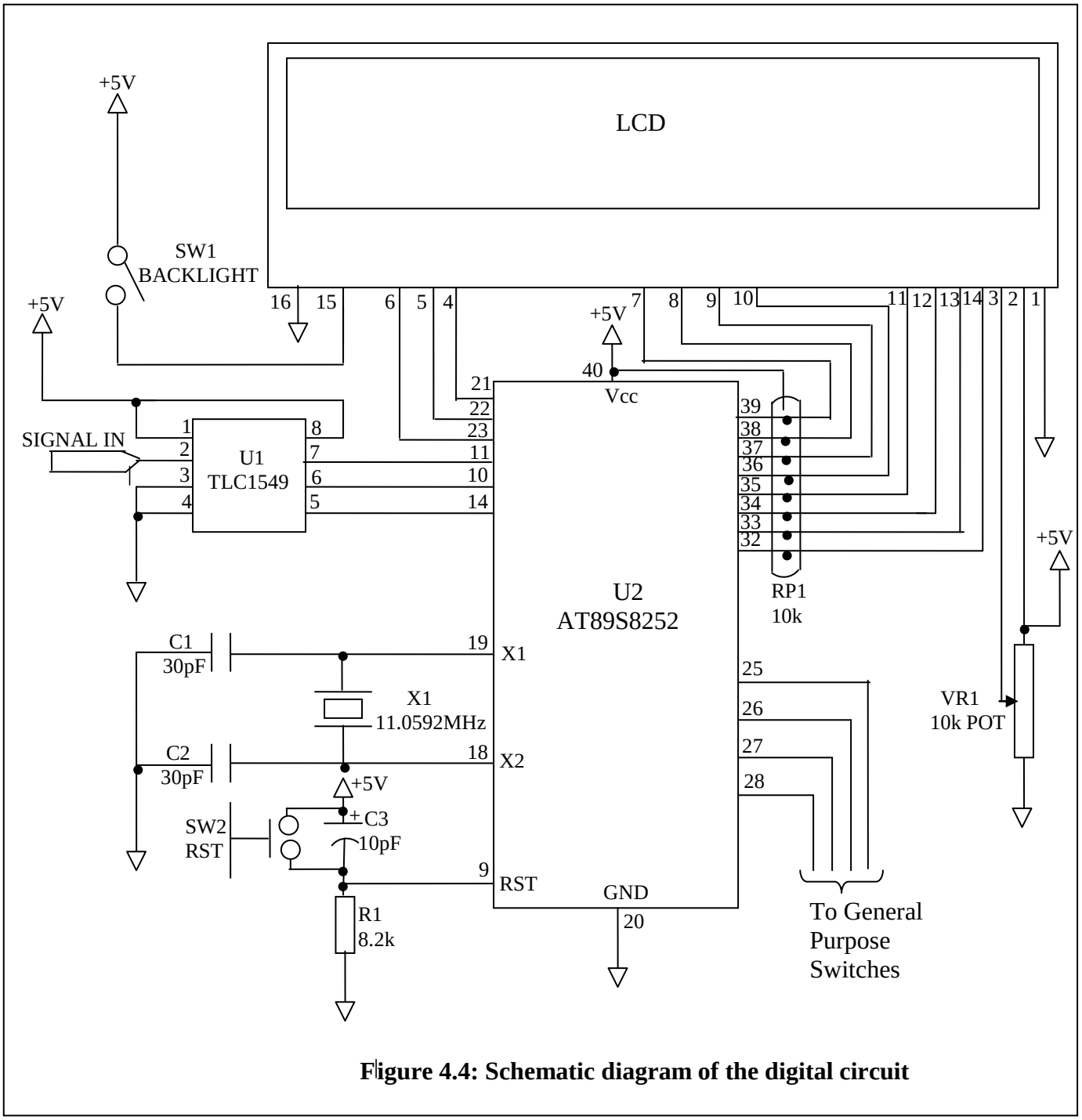


Figure 4.4: Schematic diagram of the digital circuit

The circuit was then implemented on a PCB board and thereafter interfaced to the analogue devices. The TLC 1549C used is a 10-bit, switched capacitor, successive–approximation analogue-to-digital converter. This device has two digital inputs and a 3-state output namely: chip select (CS), input-output clock (I/O clock), and data output (DATA OUT) that provide a three-wire interface to the serial port of the host micro controller.

The sample-and-hold function is automatic. The converter incorporated in this device features differential high impedance reference inputs that facilitate ratio metric conversion, scaling and isolation of analogue circuitry from logic and supply noise. A switched capacitor design allows low-error conversion over the full operating free-air temperature range. The ADC receives adjusted analog voltage from the conditioning circuitry and converts it to an equivalent 10-bit binary string. To do that, the microcontroller issues chip select (CS_1)' and start conversion (WR_1)' pulses. When the end-of-conversion (INTR) line goes low, the micro-controller issues an output enable (RD_1) to read to the data (D_0 to D_7) that pass, via the data bus, into the micro controller's random-access memory (RAM) [TLC1549C, 10-bit ADC data sheet]. Figure 4.5 shows how the 10-bit ADC was interfaced to the instrumentation amplifier output.

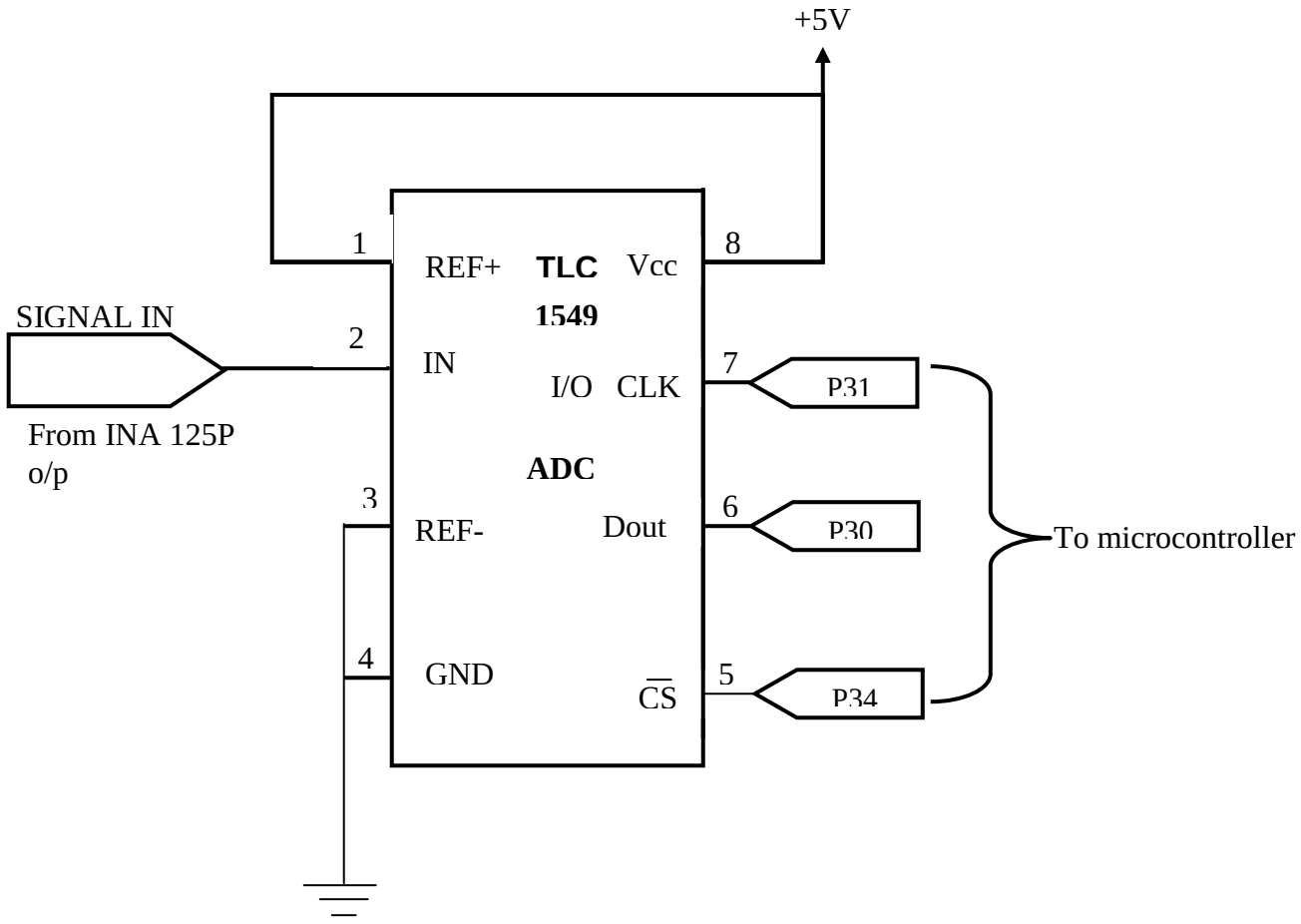


Figure 4.5: Schematic illustration of how the signal conditioner is interfaced to the 10-bit ADC

The Atmel AT89S8252 microcontroller used is a superior version of the 8051 family. It is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of downloadable flash programmable and erasable read-only-memory. It has 2K bytes of EEPROM. The device is manufactured using Atmel's high-density non-volatile memory technology and is compatible with the industry standard 80C51 instruction set and pinout. The on-chip downloadable flash allows the program memory to be reprogrammed in-system through an SPI serial interface or by a conventional non-volatile memory programmer.

By combining a versatile 8-bit CPU with downloadable flash on a monolithic chip, the Atmel AT89S8252 is a powerful microcomputer, which provides a high flexible and cost effective solution to many embedded control applications. The AT89S8252 provides the following standard features: 8K bytes of downloadable Flash, 2K bytes of EEPROM, 256 bytes of RAM, 32 I/O lines, programmable watch-dog timer, two Data Pointers, three 16-bit timer/counters, a six-vector two-level inter-architecture, a full duplex serial port, on-chip oscillator, and clock circuitry.

In addition, the AT89S8252 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The power down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next interrupt reset. The downloadable flash can be changed by a single byte at a time and is accessible through the SPI serial interface. Holding RESET active forces the SPI bus into a serial programming interface and allows the program memory to be written to or read from unless Lock Bit 2 has been activated [Predko, 2000].

A double line, 16-character alphanumeric dot matrix LCD unit was used. It has an LED backlight. The LCD display is self-contained and incorporates a CMOS microprocessor and all the supporting devices such as character generator ROM and display drivers. The module utilizes a 5 x 7 dot matrix format with a cursor and is capable of displaying the full ASCII character set plus up to eight additional user programmable custom symbols. The displays are virtually burden free to the microcontroller. Internal registers can store up to 80 characters and all updating and refreshing are done internally. Software development is greatly eased by simple instructions.

4.5 Designing the System Power Supply

The designed system has a number of ICs with different voltage ratings. Some of these require 12 V_{dc} while others work on a 5 V_{dc}. In an ordinary power supply, the voltage regulation is poor that is the V_{dc} output voltage changes appreciably with load current. Moreover, output voltage also changes due to variations in the input Ac. voltage. This is due to the following reasons:

Firstly, in practice, there are considerable variations in the a. c. line voltage caused by outside factors beyond control. This changes the output voltage. The weighing balance, just like most of the electronic circuits, would fail to work satisfactorily on such output voltage fluctuations. This necessitates the use of a regulated d. c. power supply.

Secondly, the internal resistance of an ordinary power supply is relatively large (> 30 Ω). Therefore, output voltage is markedly affected by the amount of load current drawn from the supply.

These variations in d. c. voltage may cause erratic operation of the system electronic circuits. Therefore, a regulated d. c. power supply is the only solution in such situations. Two regulators, one of which supplies 12 V and the other 5 V output were used.

In each voltage regulator circuitry, output voltage is measured, compared with the desired value, and conditions adjusted automatically so that the difference between the two is zero.

Figure 4.6 shows the regulated power supply circuit diagram.

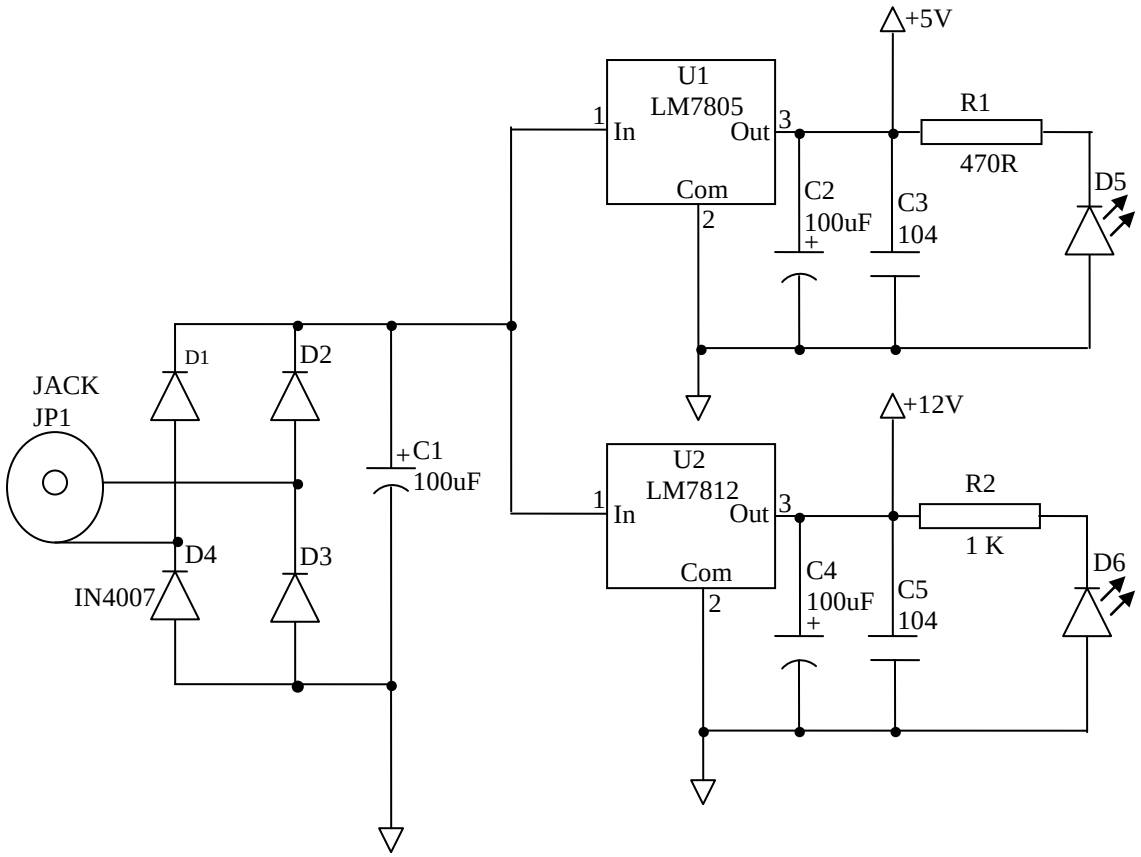


Figure 4.6: Regulated Power Supply Circuit Diagram

4.6 Designing the system software

The system program was written in assembly language using the structured programming technique. Development tools used is M-IDE studio for MCS-51.

The written code was then evaluated using Emulator 8051 Evaluation Version 1.0-TS controls. The entire code is presented in appendix D. The software design, as shown in figure 4.7, is essentially a continuously repetitive flow of processing cycles. The loops/runs are continuously being executed, while the subroutines are called during each run/loop.

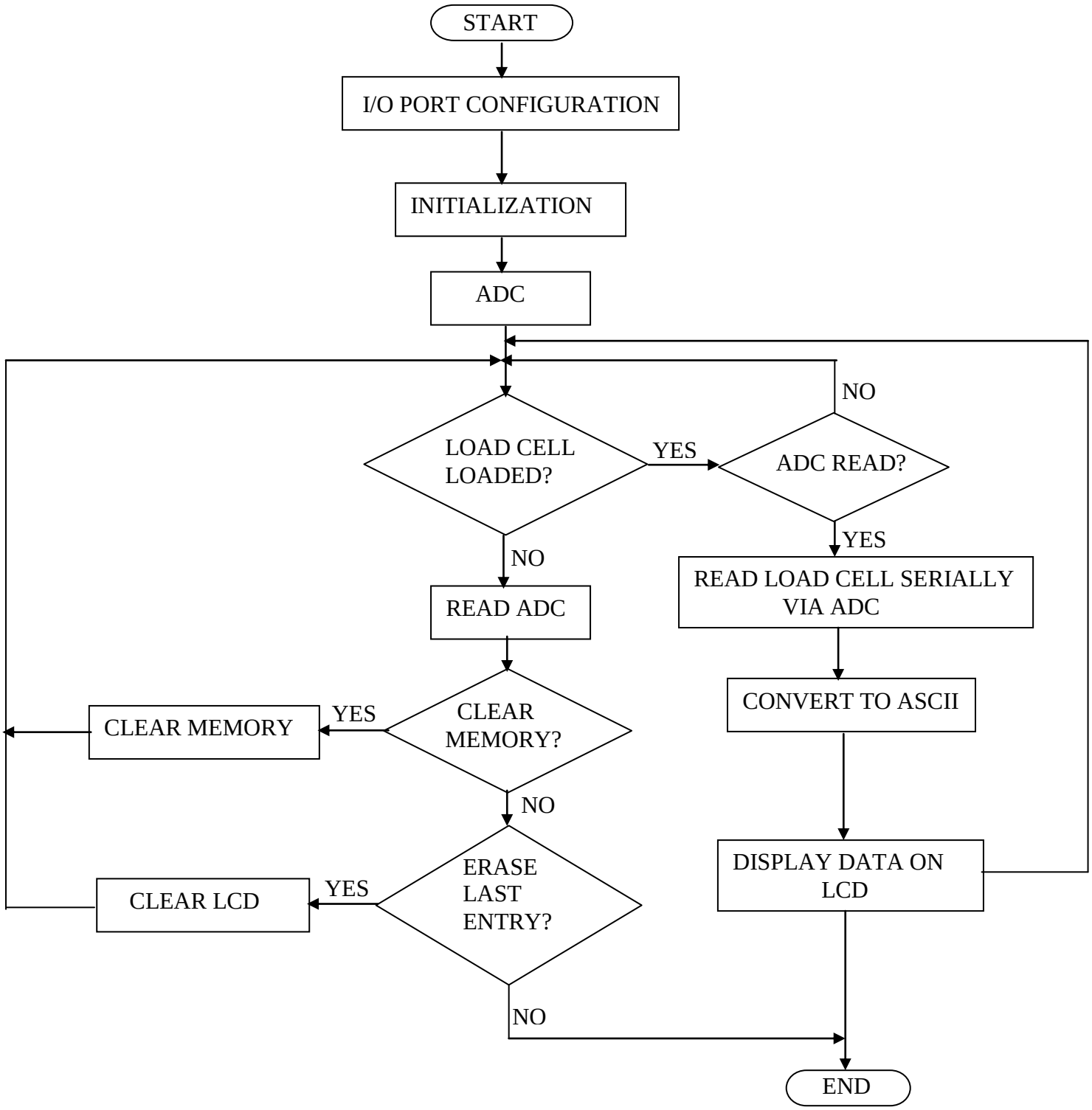


Figure 4.7: Overall System Software Flowchart

The entire software therefore comprises the following essential modules, namely initialization, main program, ADC read, serial communication, memory storage and display. These modules are described in the next sections.

4.6.1 Main Program

This is the program that calls the different subroutines as need arises.

4.6.2 Initialization

This module performs initialization for both the I/O ports of the microcontroller and the LCD when the system is activated. The I/O ports are set to operate in mode 0 i.e. simple input-output with handshaking. In this mode, the ports function as level-sensitive inputs or latched outputs. Initialization to the LCD is done to ensure that the display windows of the LCD are correctly formatted. It then clears the entire display and is ready for operation

4.6.3 ADC Read

Functions supported by this subroutine include the reading in of the serial data from the TLC1549 10-bit ADC, clearing of the 256B RAM upon successful reading, placing the most significant byte in address 20H and least significant byte in address 21H and deletion of the last read ADC data.

4.6.4 Memory Storage

The memory module allows the microcontroller to first store the read ADC data in addresses 20H and 21H and convert it to ASC11. The 5V ADC signal input becomes 1024 decimal. The ASC11 data is then passed and stored in 30H, 31H, 32H, and 33H respectively. Data is stored sequentially in the RAM.

4.6.5 Display

Upon initialization, the LCD displays, the ADC read value in the decimal format. The values are read from port O of the micro controller as routed by the microcontroller itself.

4.6.6 Serial Communication

This module is responsible for the serial communication that takes place during the display of the introductory remarks. For purposes of interface, this module generates the input clock needed to synchronize the transfer.

4.7 Coding the AT89S8252 Microcontroller

The code was ‘burnt’ into the microcontroller using PM-51 programming kit. The microcontroller IC was fitted into the programmer holder in the right orientation. Figure 4.8 shows the arrangement when programming the microcontroller chip.

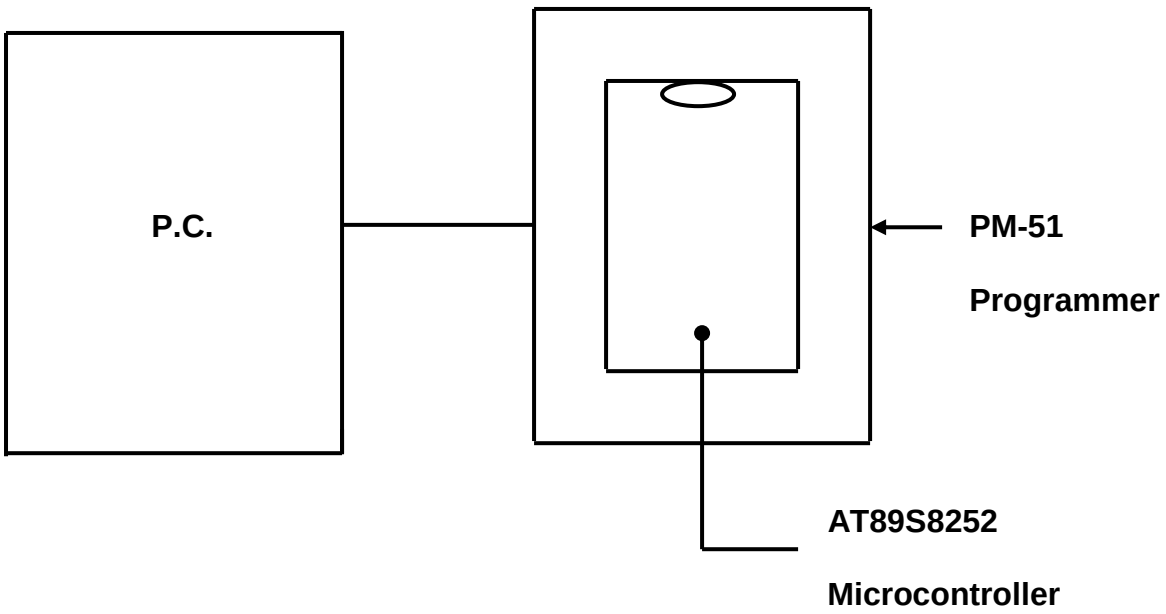


Figure 4.8: Programming the Microcontroller

The microcontroller IC was then removed and inserted into its holder in the microcontroller circuit.

4.8 Testing the System

4.8.1 Testing the Signal Conditioner

The output of the signal conditioner was monitored using a digital multimeter (DMM) as the load on the load cell platter was varied. Figure 4.9 shows the arrangement during this test.

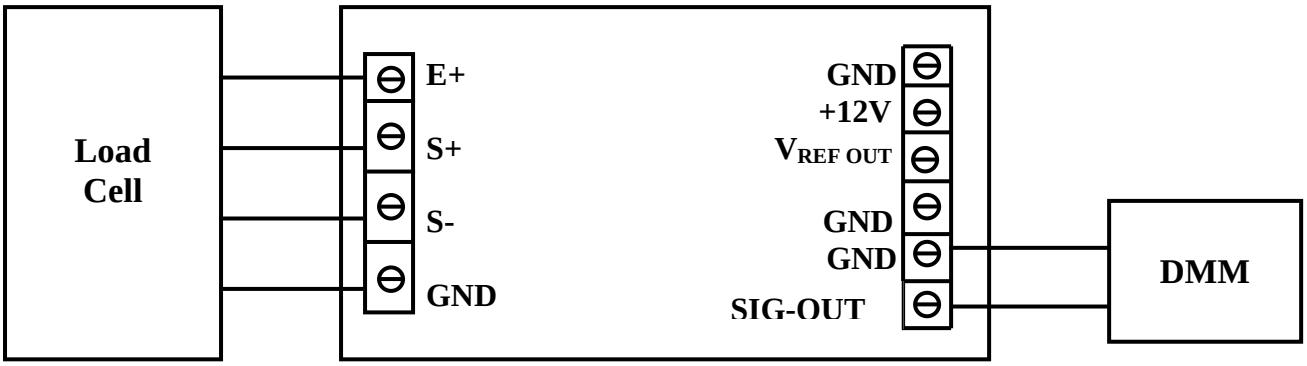


Figure 4.9: Testing the Signal Conditioner

4.8.2: Testing the Load Cell

Before powering the system, the bridge nominal resistance of the load cell was measured using the DMM connected across the s+ and s- terminals and in the resistance mode. With power on, the DMM was used to measure the bridge resistance change at zero load and then as the latter increased. The DMM was also used to monitor the load cell output voltage as load was varied.

4.8.3: Testing the entire System

The logic probe was used to test each IC by placing its metal tip on the relevant IC pin and on the printed-circuit traces to exercise their functions. The indicator lamp of the logic probe glows brightly when the digital level at a particular point is HIGH (1), goes out for LOW (0) level and dims if the level is floating.

The logic pulser was utilized to provide digital pulses to the circuits under test.

By applying a pulse to each circuit and simultaneously observing the logic probe, it was possible to tell whether the pulse signal was getting through the IC or device as expected.

Checking whether the LEDs lighted when the system was powered tested the whole circuit. Monitoring the display on the Liquid Crystal Display (LCD) display as the load was varied further tested the system.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Bridge output and input resistance

As measured using a digital multimeter (DMM), the bridge output and input resistances were 409 Ω and 351 Ω respectively. These values remained the same even when load on the platter was varied. According to the load cell calibration certificate from the manufacturer, these are 410 Ω and 350 Ω respectively. It can be noted that errors in the measurements are very small.

5.2 Comparison of the developed system and a commercial one

The developed electronic weighing balance is intended for the measurement of mass in kilograms. Standard masses were weighed using the two balances and results entered in table 5.1.

Table 5.1: Comparison of the commercial and the developed electronic balance

Mass (kg) by commercial balance	Mass (kg) by the developed balance	% Error
1.000	1.042	4.2
1.500	1.562	4.1
5.000	5.200	4.0
10.000	10.360	3.6
15.000	15.525	3.5
17.000	17.510	3.0
19.000	19.456	2.4

5.3 Some of the factors contributing to error

The quality of measurement of the developed system so much depends on the ADC's accuracy and resolution. The accuracy of an ADC suffers as the input voltage is decreased. This is so because V_{cc}-to-digital ground lines are inherently noisy due to the switching transients of the digital signals. Using separate analogue and digital grounds is not mandatory, but when used it ensures that the analogue voltage comparator will not switch falsely due to electrical noise and jitter.

The TLC 1549 ADC has two reference inputs. These values establish the upper and lower limits of the analog input to produce a full-scale and zero reading respectively. These values of REF+, REF-, and the analog input should not exceed the positive supply or be lower than the GND consistent with the specified absolute maximum ratings. The digital output is at full scale when the input signal is equal to or higher than REF+ and at zero when the input signal is equal to or lower than REF-. Every ADC measurement contains a variety of unavoidable, independent errors that influence its accuracy. When σ_i represents each independent error, the total error can be expressed as $\sigma_{total}=\sqrt{(\Sigma_i \sigma_i^2)}$.

This equation includes of a variety of errors such as sensor anomalies, noise, amplifier gain and offset, ADC quantization (resolution error), and other factors. In a theoretically perfect ADC, any particular analogue voltage measured should be represented by a unique digital code, accurate to an infinite number of digits. But in a real ADC, small but finite gaps exist between one digital number and a consecutive digital number, and the amount depends on the smallest quantum value that the ADC can resolve.

5.3.1 ADC Accuracy versus System Accuracy

Accuracy of a data acquisition (DA) device e.g. an ADC is defined as the sum of three components stated in terms of the reading, range, and the least significant bit (LSB). It is a measure of the extend to which the device is error-free. Generally, accuracy is not constant over an entire measurement range; it varies with the reading magnitude. On the other hand, resolution is defined as the smallest incremental change the ADC can recognize. For example, a 10-bit ADC has a resolution of 1 part in 1024.

5.3.2 Calibration

ADCs may be calibrated with hardware, software, or a combination of the two. Calibration in this case means adjusting the gain and offset of an ADC channel to obtain the specified input-to-output transfer function. In a hardware configuration, for example, the instrumentation amplifier driving the ADC has its offset and gain adjusted with trim pots, and changing the ADC's reference voltage changes its gain. In hardware/software calibrations, the software instructs DACs to null offsets and set full-scale voltages. Lastly, in a software calibration, correction factors are stored in nonvolatile memory in the data acquisition system or in the computer and are used to calculate the correct digital value based on the readings from the ADC. ADCs are factory calibrated before being shipped, but time and operating temperature can change the settings. ADCs need to be recalibrated usually after six months to a year, and possibly more often for ADCs with resolutions of 16 bits or more. Calibration procedures vary, but all usually require a stable reference source and an indicating meter of (at least three times) greater accuracy than the device being calibrated.

Offset is typically set to zero with zero input, and the gain is set to full scale with the precise, full-scale voltage applied to the input.

5.3.3 Linearity

When the input voltage and the ADC output readings deviate from the theoretical diagonal line (representing infinite resolution) more than the ideal step function, the ADC error is nearly impossible to eliminate by calibration. The diagonal line represents an ideal, infinite-resolution relationship between input and output. This type of ADC error is called a nonlinearity error. Nonlinearities in a calibrated ADC produce the largest errors near the middle of the input range. As a rule of thumb, nonlinearity in a good ADC should be one LSB or less.

5.3.4 Missing Codes

A quality ADC should generate an accurate output for any input voltage within its resolution, that is, it should not skip any successive digital codes. But some ADCs cannot produce an accurate digital output for a specific analog input. For example, a particular 3-bit ADC does not provide an output representing the number four for any input voltage. This type of error affects both the accuracy and the resolution of the ADC.

5.3.5 Noise

An ADC can compromise accuracy when excessive electrical noise interferes with the measured signal, whether that signal is in millivolts or much larger. For example, many ADCs that reside on cards and plug into a PC expansion bus can encounter excessive electrical noise that seriously affects their accuracy, repeatability, and stability. But an ADC does not have to be connected directly to the bus within the computer. An ADC mounted in an external enclosure often solves the problem.

When an ADC inside a computer has to be located, its noise level is checked. Connecting the ADC's input terminal to the signal common terminal should produce an output of zero volts. If it still reads a value when shorted, the noise is being generated on the circuit card and will interfere with the desired input signal. More critical diagnostics are necessary when using an external power supply because noise also can arise from both the power supply and the input leads.

Electrical noise in the system could also result from ground loops. Theses loops often occur when two or more devices in a system, such as a measurement instrument and a transducer, are connectected to ground terminals at different physical locations. Slight differences in the actual potential of each ground point generate a current flow from one device to the other. This current, which often flows through the low potential lead of a pair of measurement wires generates a voltage drop that appears as noise and measurement inaccuracy at the signal conditioner or ADC input.

The electrons in the conductors possess varying amounts of energy by virtue of the temperature of the conductors. The slight fluctuations in energy about the values specified by the most probable distribution are very small, but are sufficient to produce small noise potentials within the conductors.

These random fluctuations produced by thermal agitation of the electrons are called the thermal, or Johnson, noise. The rms value of the thermal-resistance noise voltage, V_n over a frequency range f_H-f_L is given by:

$$V_n^2 = 4\bar{k}TRB \qquad \textbf{(Eq. 5-1)}$$

where $\bar{k}$ = Boltzman constant, JK^{-1}

T = resistor temperature, K

R = resistance, Ω

B = f_H-f_L = bandwidth, Hz

There is also a possibility of having shot, or Schottky, noise which is attributed to the discrete-particle nature of the current carriers in the semiconductors involved. The current flow fluctuates just like the number of carriers does. It is only the time-average flow which is measured as the constant current. The mean-square noise current in any device is given by:

$$I_n^2 = 2qI_{dc}B \qquad \textbf{(Eq. 5-2)}$$

where q = electronic charge magnitude

I_{dc} = dc current

B = bandwidth

5.4 System Evaluation

To highlight the merits of the implemented Microcontroller Based Electronic Weighing Balance, a comparison is made with the commercially related products, related applications designed and implemented using a computer or a microprocessor or the industry standard Intel 8051 microcontroller. The six factors used in the comparison are namely; resolution and range, portability, number of main ICs, flexibility. PCB size and display capabilities. The commercially related products used in the comparison are the electronic balances commonly used in retail butcheries e.g. CAS AP scales and those that use batteries and commonly used in scientific research laboratories e.g. KD 7000 digital balances.

The summary of the comparison is tabulated in Table 5.2.

Table 5.2: Summary of comparison of the developed system with some existing weighing balances

	No. of main ICs	Portability	PCB Size (cm)	Flexibility	Range & Resolution	Display Capability
Developed MCBEWB	4	Unlimited	7 by 8	Extremely Flexible	0-19 kg, 40 g	Wide
BEWB	>9	Limited	15 by 17	Non-Flexible	0-15 kg, 50 g	Narrow
SRLEWB	>9	Unlimited	13 by 14	Non-Flexible	0-15 kg, 50 g	Narrow

Abbreviations:

MCBEWB = Microcontroller Based Electronic Weighing Balance

BEWB = Butchery Electronic Weighing Balance

SRLEWB = Science Research Laboratory Electronic Weighing Balance.

5.4.1 Number of main ICs

In terms of the number of components, the developed Atmel AT89S8252 Microcontroller Based Electronic Weighing Balance (MCBEWB) uses the least. It uses the mains power supply for its internal regulated power supply. The microcontroller chip has adequate in-built Flash Memory, EEPROM, RAM and I/O ports.

Any microprocessor based electronic weighing balance would require an external RAM, an external EPROM and an I/O device among other support chips.
This renders the latter extremely expensive. The Scientific Research Laboratory Electronic Weighing Balance (SRLEWB) uses batteries, which are very expensive.

5.4.2 Portability

From the portability perspective, the developed system has unlimited range as all that is needed is a single external power supply. It has an in-built regulated power supply and works well with the mains power supply. Moreover, it has been designed to make it as compact as possible. The Butchery Electronic Weighing Balance (BEWB) is bulky and therefore has limited portability.

5.4.3 Space

In terms of space, the developed system required least space for both the digital circuitry and the analog circuitry. This is attributed to the fact that minimal number of chips was used and to crown it all, the system was implemented on PCB boards. In addition, data is serially sent from the 10-bit ADC to the microcontroller. Only three lines are used to interface the ADC to the micro controller, making the circuit design easier. Owing to the many external chips used, in BEWBs and SRLEWBs, a lot of space is required for circuit implementation.

5.4.4 Flexibility

The developed system is considered the most flexible, as in-system reprogramming is possible. This is courtesy of the built-in Serial Programmable Interface (SPI) of the AT89S8252 microcontroller. Program can be written/erased up to 1000 times.
Some BEWBs and SRLEWBs use dedicated chips like those in DMMs.
They only perform specific tasks and cannot be changed or modified to do otherwise.
For those that are microprocessor based, in-system reprogramming is not possible.
Microprocessor based related Control/Data Acquisition Applications use external EPROM for program storage. To change the program, it is erased by exposing it to UV, a tedious process indeed.

5.4.5 Range and Resolution

The developed electronic weighing balance has tried to address both resolution and range. It is the best in as far as striking a balance between range and resolution goes. This is a modest system with a reasonable range while at the same time being sensitive enough. This is quite rare in virtually all-electronic weighing systems.

5.4.6 Display Capability

In as far as display capability is concerned; the developed system enjoys the widest capability. Using the ASCII code, it can display all alphanumeric data/information. It can display special characters. BEWBs and SRLEWBs can only display numeric data/information. Information in letters or symbols is usually engraved alongside the LCD display.

CHAPTER 6

CONCLUSION AND OUTLOOK

6.1 Conclusion

This research work has presented a new concept of making a flexible electronic weighing balance, which is indeed a data acquisition system, through the use of a low-cost microcontroller that has a built-in Serial Programming Interface (SPI). The electronic balance is developed based on the Atmel AT89S8252 microcontroller that serially interfaced to a 10-bit ADC. The microcontroller has 8 kB of in-system reprogrammable downloadable flash memory for storing the program, 2 kB of EEPROM Memory for data and 256 bytes of internal RAM for storing variables. The transducer used is a foil strain gauge based load cell mounted to operate by the bending principle. When a load is placed on the platter, the gauge resistance changes. The load cell bridge circuit converts this resistance change to an electrical voltage, which is in turn fed to and amplified by the INA125P instrumentation amplifier and associated circuitry. The signal conditioner is interfaced to the ADC. The analog and digital circuits were designed and implemented on PCB boards using ExpressPCB. The system code was designed and developed using M-IDE Studio for MCS-51. It was ‘burnt’ into the microcontroller via the PM-51 programmer. In comparison with commercially available electronic weighing balances, the developed microcontroller based electronic weighing balance is low-cost, flexible and portable. The fabricated system is able to measure mass in the range of 0 to 19 kg to a sensitivity of 40 g per volt of excitation voltage.

6.2 Recommendations

The proposed weighing system can be employed in the laboratory for measurement of mass in the high regime, teaching on microcontroller 8051 applications and commercial purposes like in butcheries.

6.3 Suggestions for further work or outlook

In connection with this research project, the following suggestions for further work are made:

(i) Design, develop and implement a better signal conditioning circuitry

(ii) Design, develop and implement a better program for the system

(iii) Modify the designs to attain a sensitivity of 30 g per division, a range of 0 to 30 kg and higher accuracy.

REFERENCES

Belove, C. (1986), *Handbook of Modern Electronics and Electrical Engineering,* John Wiley and Sons Inc. U.S.A., pp 2-19.

Bentley, J. P., *Measurement Systems*, Addison Wesley Longman Ltd., UK, 2000.

Borer, J. (1991), *Microprocessor in Process Control,* University Press Great Britain, pp 21-30.

Braby, P. W., *Electronics: A Practical Introduction*, John Wiley & Sons, Canada Limited, 1983.

Brophy, J.J.(1990) *Basic Electronics for Scientists,* McGraw-Hill International Editions, fifth Edition,pp 462.

Chandhuri, N. (1985), *A Technique for Simultaneous Measurement with a Microcomputer:* IEEE Transactions on Industrial Electronics, Vol.32 No.2, p9.

Commer, D.J (1987) *State Machine versus Microprocessor Controller: Digital System Design.* Vol. E 30 NO.2, pp 5-8.

David, et al, *Using the 8051 Micro controller with Resonant Transducers*, IEEE Transaction on Industrial Electronics, Vol. IE-32 No. 2, November 1985.

Dorf, R (1993), *Electrical Engineering Handbook,* CRC press, U. S. A.

Downtown, A. C. (1980), *Computers and Microprocessors: Components and Systems.* Nostral Van Reinhold (UK) CO.LTD, England, pp 1-23.

Fuller, J.L (2002), www.google.com, *Introduction to Robotics.*

Gaonkar, R.S (1996), *Microprocessor Architecture, Programming and Applications with the 8085,* Third Edition, Prentice Hall. Inc, U.S.A.

Gopal. M. (1989), *Digital Control Engineering,* Willey Eastern Ltd-New Delhi, India. pp 430.

Grabel, A. and Millman, *Microelectronics*, Mc Graw-Hill, 1987.

Hoekstra. L.R.(1986). *Robotics and Automated Systems,* South Western Publishers Co. Ohio, pp 12 -87.

Horowitz Paul and Winfield Hill, *The art of electronics*, Cambridge University press, 1995.

Houpis, H.C and Lamount, B.G (1992), *Digital Control Systems: Theory, Hardware, Software,* McGraw Hall Inc; Singapore, pp 234.

James, M. R., *Microcontroller Cookbook, PIC and 8051,* Newness Publications, 1987.

Karimi. P., *Microprocessor Controlled Multifunction Generator*, MSc. Thesis, Kenyatta University, 2000.

Kleitz, William (2003), *Digital and Microprocessor Fundamentals-Theory and Applications,* Pearson Education Inc. 4th Edition Upper Saddle River, New Jersey (U.S.A), pp 251-587.

Lockwood, F.B. and Duncan. R. (1971), *Electrical Engineering Principles,* Heinemann Publishing Ltd, London, pp 1-90.

Louw.A. and Thomas, M.E. (1984), *Scientific Instruments, Journal of Physics, The Institute of Physics,* Great Britain, August, Vol.17 No.8, pp 63-714.

Mackenzie, S. (1999), *The 8051 Microcontroller*, Prentice Hall Inc; U. S. A., pp 1-25.

Masidi Ali Muhammad and Masidi Gillespie Janice. *The 8051 Microcontroller and Embedded Systems*, Pearson Education Asia, 2002.

Mazza, L., *Micro controller-based motor speed control,* Web electric Magazine, 2001.

Mehta, V. K. & Mehta R., *Principles of Electronics*, S. Chand & Company Ltd., 2005.

Meyerle, G. (1983), *16-bit Microprocessor Technology, Computers and Electronics,* Vol.21 No.3, pp 1-89.

Millman and Halkias, *Integrated Electronics: Analog and Digital Systems*, Mc Graw-Hill, 1986.

National Semiconductor Resources Website at www.national.com.

Owade, M., *Design and Development of a Programmable Laboratory Interface System with an illustrative use in Resistivity Experiment*, MSc.Thesis, Kenyatta University, 1998.

Predko, Mike, *Programming and customizing the 8051 Microcontroller*, Pearson Education, 2000.

Ramsay, D.C. (2000), *Engineering Instrumentation and Control,* Stanley Thorner (Publishers) Ltd. 2nd Edition, pp 29-123.

Reaves, E.A. (1995), *Newness Electrical Pocket Book,* Newness, Great Britain, pp 5-18.

Ronald, J. T., and Neal, S. W., *Digital Systems: Principles and Applications*, 8th edition, Prentice Hall of India, New Delhi, 2004.

Rotich, R. K., *Design and Fabrication of a Microprocessor-Based Industrial Robot Control System,* MSc. Thesis, Kenyatta University, 2005.

Ryan, W.www.google.com, (2003) *Choosing the right strain-Gauge for your Applications,* pp 11.

Sawhney, A. K., *A course in Electrical and Electronic Measurements and Instrumentation*, Educational and Technical Publishers, New Delhi, 2003.

Sinclair, I. (1991), *Practical Electronics Handbook,* Newness, Great Britain, pp 1-11.
Siskind, C.S. (1988), *Electrical Control Systems in Industry,* McGraw-Hill Company, U.S.A, pp 1-115.

Sivakumar, K. (1987), *Digital Control of a Machine, Electronics Project,* EFY Enterprises Publishers, Great Britain, Vol.8, pp 12-18.

Tierman, C. (2001), *Realizing benefits from IT,* IMIS VOL 11, NO.5, October, pp 32-76.

Tooley, M. (1996), *Electronics Circuits Handbook, Design, Testing and Construction,* Second Edition, Newness, Great Britain, pp 1-41.

Torrens R., *PWM speed control*, June 2002,www.4qdtec.com/pwm-02.html

Tseng, V. (1982), *Microprocessor Development and Development Systems,* Granada, Great Britain, pp 1-154.

Usher, M.J. and Keating, D.A. (1996). *Sensors and Transducers: Characteristics, Applications, Instrumentation, Interfacing,* Macmillan, 2nd Edition, pp 222.

Utietz, S. (1991). *Electronic Circuits Designs and Applications,* Verlag-Springer, Germany, pp 15-90.

www.acabus.com *(Ace bus) Downloadable 8051 Micro controller Editor, Assembler and Simulator.*

www.amambonet/8085. html *(ABC Creations) 8085 simulator.*

www.amd.com) *Advanced Micro Devices, Inc) Microprocessors, EPROMS and Flash Memory.*

www.analog.com *(Analog Devices, Inc) Analog ICs, ADCs, and DACs.*

www.bipom.com *(BiPom Electronics, Inc) Downloadable 8051 Micro controller editor, assembler and simulator.*

www.electroguys.com/carparking/12/7/2005.

www.expresspcb.com-support@expresspcb.com.

www.google.com (2003), *Weighing Scales Based on Lower-Power Strain- Gauge Circuits,* pp 2.

APPENDIX A: Load Cell Calibration Certificate

Load Cell Calibration Certificate

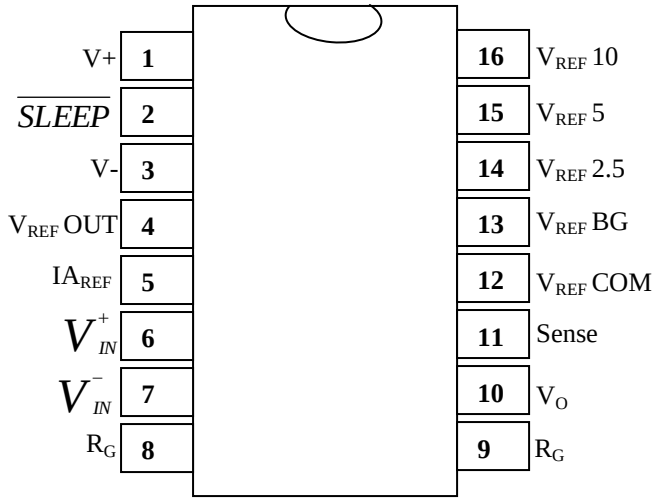
```
Model #: LPS-30kg-S
Serial #: L94298
Capacity:   30kg
```

Wiring Code:

+ Excitation : Red	+Signal : Green
- Excitation : Blue	- Signal : White

Full Scale Output : 1.9230 mV/V
Zero Balance : 3.0% FS
Rated Excitation : 10V AC or DC
Compensated Temp. Range : 14° to 104° F
Insulation Resistance : > 5000 MΩ AT 50 VDC
Barometric Effect : Nil
Input Resistance : 410 Ω
Output Resistance : 350 Ω
Minimum Dead Load : 1.200 kg
Vmin : 3.760
Safe Load Limit : 150% OF CAPACITY
Direction of Loading : TENSION

APPENDIX B: INA125P Pin Configuration



ABSOLUTE MAXIMUM RATINGS

Power Supply Voltage, V+ to V- ……..... 36 V

Input Signal Voltage ……. +/- 40 V

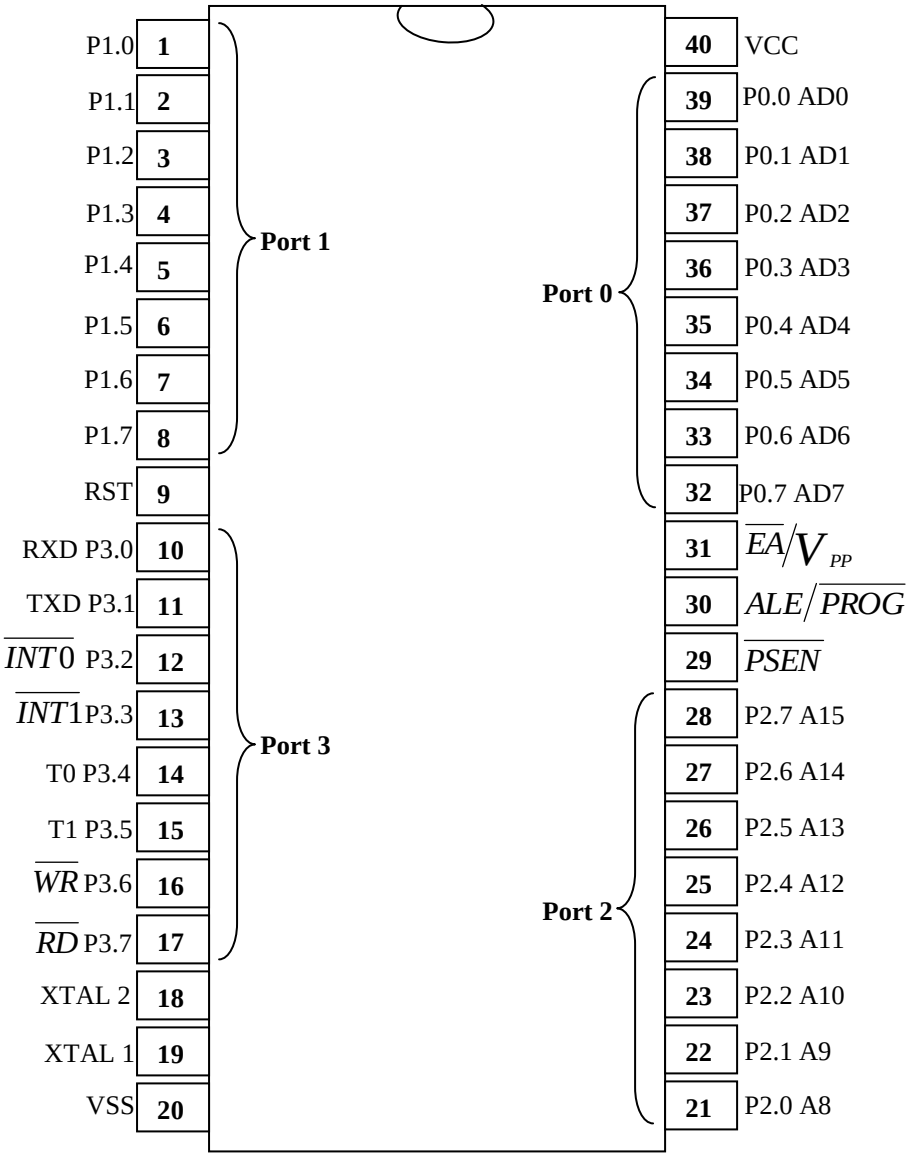
Output Short Circuit ……... Continuous

Operating Temperature…… -55 ^{o}C to +125 ^{o}C

Storage Temperature…… -55 ^{o}C to +125 ^{o}C

Lead Temperature (soldering, 10 s)……+300 ^{o}C

APPENDIX C: The 8051 Microcontroller Pin Configuration



APPENDIX D-I: The Intel MCS-51 Programmer’s Guide and Instruction Set

APPENDIX D-II: The Intel MCS-51 Programmer's Guide and Instruction Set

APPENDIX D-III: The Intel MCS-51 Programmer's Guide and Instruction Set

APPENDIX D-IV: The Intel MCS-51 Programmer's Guide and Instruction Set

APPENDIX E: Program Listing

Program Designer: Munyao Kitavi

Software Title: Weigh_Scale.A51

Development Tools: M-IDE Studio for MCS-51

Programming kit: PM-51 Version 1.0

```
;***************************************************************************


LDC_DATABUS EQU P0    ;DEFINE LCD PORTS
RS EQU   P2.2         ;DEFINE LCD CONTROL PINS
R_W EQU  P2.1         ;
E EQU    P2.0         ;

MSBYTE EQU 20H       ;* * * * * * * * * * * * * * * * * * * *
LSBYTE EQU 21H       ;* Name data destinations            *
COUNT EQU R3         ;* and COUNT register                *
                     ;* * * * * * * * * * * * * * * * * * *


;MAIN PROGRAM

ORG 0H

ACALL DISP_UNIV
ACALL PYS_DEPT
ACALL DISP_WELCOME
ACALL DISP_NAME
ACALL SW_VERSION

MOV R7,#3

INIT_LOOP:

ACALL INITIAL_0
ACALL INITIAL_1
ACALL INITIAL_2
ACALL INITIAL_3
ACALL INITIAL_4

DJNZ R7,INIT_LOOP


;MAIN PROGRAM

LOOP_MAIN:

LCALL ADC_DATA
LCALL ASCII_CONVERTER
LCALL LCD_OUTPUT



LCALL DELAY1

LJMP LOOP_MAIN
```

```
;*************************************************************************
****

;this subroutine gets Data from the ADC chip - TLC1549.
;P3.1 = CLK
;P3.0 = DOUT
;P3.4 = CS


;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                                                                       *
;* TLC1549 - 80C51 Interface Program                                     *
;*                                                                       *
;* This program contains a subroutine ADC which reads                    *
;* in the serial data from the TLC1549 10-bit ADC                        *
;* and places the most significant byte in address 20H                   *
;* and least significant byte in address 21H                             *
;*                                                                       *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

ADC_DATA:

START: MOV COUNT, #04H                  ;Set COUNT=2 (Do 2 conversions)
       JMP ADC                          ;Jump to subroutine ADC
       JMP START                        ;Repeat above again
  ADC: SETB P3.4                        ;* * * * * * * * * * * * * * * * *
* *
       MOV A, #20H                      ;*      Set Port3 (bit 4) high
*
DELAY: DEC A                            ;*     (Sets CS of TLV1549 high)
*
       JNZ DELAY                        ;* * * * * * * * * * * * * * * * *
* *
       CLR P3.4
       MOV SCON, #10H
READ1: JNB SCON.0, READ1              ;Read in
       CLR SCON.0                       ;most significant
       MOV R0,SBUF                      ;byte, place in R0
READ2: JNB SCON.0, READ2              ;Read in
       CLR SCON.4                       ;least significant
       CLR SCON.0                       ;byte,
       MOV R1, SBUF                     ;place in R1
       DEC COUNT                        ;COUNT-1
       MOV A, COUNT                     ;
       JNZ ADC                          ;If COUNT not = 0,
                                        ;do another conversion
      MOV R4, #08H                      ;Put 08H in R4
      MOV R2, #01H
 LOOP_FORMAT: CLR C    ;* * * * * * * * * * * * * * * * * * *
      MOV A, R0        ;*                                   *
      RRC A            ;*                                   *
      MOV R0, A        ;*          Reformats MSByte         *
      MOV A, R2        ;*                                   *
      RLC A            ;*                                   *
      MOV R2, A        ;*                                   *
      DEC R4           ;*                                   *
      MOV A, R4        ;*                                   *
      JNZ LOOP_FORMAT  ;* * * * * * * * * * * * * * * * * * *
      MOV A, R2        ;
```

```
      MOV 20H, A       ;
      MOV A, R1        ;* * * * * * * * * * * * * * * * * * * *


      RRC A            ;*                                     *


      MOV 21H, A       ;*          Reformats LSByte           *
      MOV 21H.1, C     ;*                                     *
      CLR 21H.7        ;* * * * * * * * * * * * * * * * * * * *


      MOV A,MSBYTE      ;A = D9-D2
      ANL A,#00111111B  ; MASK D9 & D8
      RL A
      RL A
      ORL A, LSBYTE
      MOV R0,A          ; R0 = D7,D6,D5,D4,D3,D2,D1,D0



      MOV A,MSBYTE      ; A = D9-D2
      ANL A,#11000000B  ; MASK D7 - D2
      RL A
      RL A
      MOV R1,A         ;R1 = 0,0,0,0,0,0,D9,D8


 RET                   ;Return from subroutine

;*************************************************************************
****************

;Subroutine for converting ADC data into ASCII.
;hence 5V input to ADC becomes 1024Dec
;The ASCII numbers are stored in 30H,31H,32H,33H Respectively.

;converting 10-bit binary to ASCII
;upon return, 30H,31H,32H,33H have ASCII data (33H has LSD)

;*************************************************************************
********

ASCII_CONVERTER:

        ;This program contains a Subroutine that Multiplies a 10-bit binary
Number by two.
        ;It does this by Shifting to the left the bits.

;R0 = LOW BYTE
;R1 = HIGH BYTE

MOV A,R0    ;load ACC with low byte first
RLC A       ;Rotate bits left thru Carry bit (i.e MUL by 2)
CLR ACC.0   ;CLEAR D0 TO PREVENT DATA CORRUPTION
MOV R0,A    ;Restore low byte divided value back into R0 LBYTE


MOV A,R1    ;load ACC with high byte next
RLC A       ;Rotate bits left thru Carry bit (i.e MUL by 2)
MOV R1,A    ;Restore high byte divided value back into R1 HBYTE
```

```
;=====================================================================
; subroutine DIV16
; 16-Bit / 16-Bit to 16-Bit Quotient & Remainder Unsigned Divide
;
; input:    r1, r0 = Dividend X
;           r3, r2 = Divisor Y
;


; output:   r1, r0 = quotient Q of division Q = X / Y
;           r3, r2 = remainder
;
; alters:   acc, B, dpl, dph, r4, r5, r6, r7, flags
;=====================================================================

;R1 = HIGH BYTE
;R0 = LOW BYTE

MOV R3,#0   ; LEAVE AS ZERO
MOV R2,#10  ; THIS IS 10 DEC FOR DIVISIVE PURPOSES


 DIV16:        MOV     R7, #0          ; clear partial remainder
               MOV     R6, #0
               MOV     B, #16          ; set loop count

div_loop:      CLR     C               ; clear carry flag
               MOV     A, r0           ; shift the highest bit of
               RLC     A               ; the dividend into...
               MOV     R0, A
               MOV     A, R1
               RLC     A
               MOV     R1, A
               MOV     A, r6           ; ... the lowest bit of the
               RLC     A               ; partial remainder
               MOV     R6, A
               MOV     A, R7
               RLC     A
               MOV     R7, A
               MOV     A, R6           ; trial subtract divisor
               CLR     C               ; from partial remainder
               SUBB    A, R2
               MOV     dpl, A
               MOV     A, R7
               SUBB    A, R3
               MOV     dph, A
               CPL     C               ; complement external borrow
               JNC     div_1           ; update partial remainder if
                                       ; borrow
               MOV     R7, dph         ; update partial remainder
               MOV     R6, dpl
div_1:         MOV     A, r4           ; shift result bit into partial
               RLC     A               ; quotient
               MOV     R4, A
               MOV     A, R5
               RLC     A
```

```
                MOV     R5, A
                DJNZ    B, div_loop
                MOV     A, R5           ; put quotient in R0, And R1
                MOV     R1, A
                MOV     A, R4
                MOV     R0, A
                MOV     A, R7           ; get remainder, saved before the
                MOV     R3, A           ; last subtraction


                MOV     A, R6
                MOV     R2, A

;NOW START CONVERTING VALUES TO ASCII FOR DISPLAY PURPOSES

MOV A,R2      ;load Acc with R2,containing first remainder of 3ffx2
ORL A,#30H    ;make into ASCII
MOV 33H,A     ;store into 23h RAM location



MOV A,R0      ;load ACC with contents of R0 containing first quotient of
3ffx2

MOV B,#10     ;Now start converting Binary to ASCII
DIV AB
MOV 32H,B     ;Save rem. in 22H=LSD

MOV B,#10     ;Divide again
DIV AB

ORL A,#30H    ;Make the MSD into ASCII
MOV 30H,A     ;Save this MSD in 20H

MOV A,B       ;Load Acc. with Rem.
ORL A,#30H    ;Convert it into ASCII
MOV 31H,A     ;Save this in 21H

MOV A,32H     ;Load 22H in Acc.
ORL A,#30H    ;Convert this to ASCII
MOV 32H,A     ;Reload back in 22H

;30H=MSD,    31H=Middle MSD    32H=Middle LSD   33H=LSD

          RET

;*************************************************************************
***********************
LCD_OUTPUT:
;Check busy flag before sending data, command to LCD


        MOV     A,#38H        ;init. LCD 2 lines,5x7 matrix
        ACALL   COMMAND       ;issue command
        MOV     A,#0CH        ;LCD on, cursor oFF
        ACALL   COMMAND       ;issue command
        MOV     A,#01H        ;clear LCD command
        ACALL   COMMAND       ;issue command
        MOV     A,#06H        ;shift cursor right
        ACALL   COMMAND       ;issue command
        MOV     A,#80H        ;cursor: line 1, pos. 1
```

```
ACALL   COMMAND       ;command subroutine


MOV     A,#' '        ;display SPACE
ACALL   DATA_DISPLAY
MOV     A,#' '        ;display SPACE
ACALL   DATA_DISPLAY
MOV     A,#' '        ;display SPACE
ACALL   DATA_DISPLAY
MOV     A,30H         ;display MSD
ACALL   DATA_DISPLAY
MOV     A,31H         ;display Middle MSD
ACALL   DATA_DISPLAY


MOV     A,#'.'        ;display "."
ACALL   DATA_DISPLAY
MOV     A,32H         ;display middle LSD
ACALL   DATA_DISPLAY
MOV     A,33H         ;display LSD
ACALL   DATA_DISPLAY
MOV     A,#'0'        ;display 0
ACALL   DATA_DISPLAY


MOV     A,#' '         ;display SPACE
ACALL   DATA_DISPLAY


MOV     A,#'K'         ;display letter K
ACALL   DATA_DISPLAY
MOV     A,#'g'         ;display letter g
ACALL   DATA_DISPLAY
MOV     A,#'s'         ;display letter s
ACALL   DATA_DISPLAY

;****************************************

MOV     A,#0C0H         ;cursor: line 2, pos. 1
ACALL   COMMAND       ;command subroutine

MOV     A,#'R'        ;display letter R
ACALL   DATA_DISPLAY
MOV     A,#'a'         ;display letter a
ACALL   DATA_DISPLAY
MOV     A,#'n'         ;display letter n
ACALL   DATA_DISPLAY
MOV     A,#'g'         ;display letter g
ACALL   DATA_DISPLAY
 MOV     A,#'e'        ;display letter e
ACALL   DATA_DISPLAY
MOV     A,#' '         ;display SPACE
ACALL   DATA_DISPLAY
MOV     A,#'0'         ;display letter 0
ACALL   DATA_DISPLAY
MOV     A,#'-'         ;display letter -
ACALL   DATA_DISPLAY
MOV     A,#'2'         ;display letter 2
ACALL   DATA_DISPLAY
MOV     A,#'0'         ;display letter 0
ACALL   DATA_DISPLAY
MOV     A,#'K'         ;display letter K
```

```
        ACALL   DATA_DISPLAY
        MOV     A,#'g'        ;display letter g
        ACALL   DATA_DISPLAY
        MOV     A,#'s'        ;display letter s
        ACALL   DATA_DISPLAY


;*************************************************************************
*****

COMMAND: ACALL   READY                  ;is LCD ready?
        MOV     LDC_DATABUS,A           ;issue command code
        CLR     RS                      ;RS=0 for command
        CLR     R_W                     ;R/W=0 to write to LCD
        SETB    E                       ;E=1 for H-to-L pulse
        CLR     E                       ;E=0 ,latch in
        RET


;*************************************************************************
*****

DATA_DISPLAY:
        ACALL   READY                  ;is LCD ready?
        MOV     LDC_DATABUS,A          ;issue data
        SETB    RS                     ;RS=1 for data
        CLR     R_W                    ;R/W=0 to write to LCD
        SETB    E                      ;E=1 for H-to-L pulse
        CLR     E                      ;E=0, latch in
        RET

;*************************************************************************
*****



READY:
     SETB    P0.7                    ;make P1.7 input port
     CLR     RS                      ;RS=0 access command reg
     SETB    R_W                     ;R/W=1 read command reg

;read command reg and check busy flag

BACK:CLR     E                      ;E=1 for H-to-L pulse
     SETB    E                      ;E=0 H-to-L pulse
     JB      P0.7,BACK              ;stay until busy flag=0
     RET

;*************************************************************************
*****

DELAY1:MOV TMOD,#10H ; timer mode 2, auto reload
       MOV R0,#15    ; loop number 15 times
AGAIN:MOV TL1,#08H   ; set delay for one loop
       MOV TH1,#01H  ;
       SETB TR1      ;run timer
BACK1:JNB TF1,BACK1  ; check for overflow flag
       CLR TR1       ;stop timer
       CLR TF1       ;clear flag
       DJNZ R0,AGAIN ;loop

RET
```

```
;*************************************************************************
*****

SHORT_DELAY:

MOV R7,#200                     ;load 200 to register
WAIT_HERE:DJNZ R7,WAIT_HERE     ;short delay loop 200 times


;*************************************************************************
*******
LONG_DELAY:MOV TMOD,#10H ; timer mode 2, auto reload
       MOV R0,#60        ;loop number 60 times
RPT:MOV TL1,#08H         ;loop number 60 times
       MOV TH1,#01H      ;
       SETB TR1          ;run timer
BACK2:JNB TF1,BACK2      ; check for overflow flag
       CLR TR1           ;stop timer
       CLR TF1           ;clear flag
       DJNZ R0,RPT       ;loop

RET



;*************************************************************************
*******
LONG_DELAY0:MOV TMOD,#10H  ; timer mode 2, auto reload
       MOV R0,#4           ;loop number 4 times
RPT1:MOV TL1,#08H          ;loop number 8 times
       MOV TH1,#01H        ;
       SETB TR1            ;run timer
BACK5:JNB TF1,BACK5        ; check for overflow flag
       CLR TR1             ;stop timer
       CLR TF1             ;clear flag
       DJNZ R0,RPT1        ;loop

RET

;*************************************************************************
*******

        DISP_UNIV:

        MOV     A,#38H        ;init. LCD 2 lines,5x7 matrix
        ACALL   COMMAND       ;issue command
        MOV     A,#0CH        ;LCD on, cursor oFF
        ACALL   COMMAND       ;issue command
        MOV     A,#01H        ;clear LCD command
        ACALL   COMMAND       ;issue command
        MOV     A,#06H        ;shift cursor right
        ACALL   COMMAND       ;issue command
        MOV     A,#80H        ;cursor: line 1, pos. 1
        ACALL   COMMAND       ;command subroutine


          MOV DPTR,#KENYATTA  ;load DPTR with adress of label
          MOV R2,#16          ;loop number 16
BACK_KEN: CLR A               ;
          MOVC A,@A+DPTR      ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A   ;load lcd with ascii data
```

```
          ACALL   DATA_DISPLAY;call display command
          INC DPTR            ;point to next adress
          DJNZ R2,BACK_KEN    ;loop 16 times


        MOV     A,#0C0H         ;cursor: line 2, pos. 1
        ACALL   COMMAND         ;command subroutine

          MOV DPTR,#UNIVERSITY ;load DPTR with adress of label
          MOV R2,#16           ;loop number 16
BACK_UNIV: CLR A
          MOVC A,@A+DPTR       ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A    ;load lcd with ascii data
         ACALL   DATA_DISPLAY  ;call display command
          INC DPTR             ;point to next adress
          DJNZ R2,BACK_UNIV    ;loop 16 times


   LCALL LONG_DELAY
          RET




  ;***********************************************8

   PYS_DEPT:

        MOV     A,#38H          ;init. LCD 2 lines,5x7 matrix
        ACALL   COMMAND         ;issue command
        MOV     A,#0CH          ;LCD on, cursor oFF
        ACALL   COMMAND         ;issue command
        MOV     A,#01H          ;clear LCD command
        ACALL   COMMAND         ;issue command
        MOV     A,#06H          ;shift cursor right
        ACALL   COMMAND         ;issue command
        MOV     A,#81H          ;cursor: line 1, pos. 2
        ACALL   COMMAND         ;command subroutine


          MOV DPTR,#PHYSICS  ;load DPTR with adress of label
          MOV R2,#16         ;loop number 16
BACK_phy: CLR A
          MOVC A,@A+DPTR     ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A  ;load lcd with ascii data


         ACALL   DATA_DISPLAY;call display command
          INC DPTR           ;point to next adress
          DJNZ R2,BACK_phy   ;loop 16 times


        MOV     A,#0C1H         ;cursor: line 2, pos. 1
        ACALL   COMMAND         ;command subroutine
```

```
          MOV DPTR,#MSC     ;load DPTR with adress of label
          MOV R2,#16        ;loop number 16
BACK_MSC: CLR A
          MOVC A,@A+DPTR     ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A  ;load lcd with ascii data
         ACALL   DATA_DISPLAY;call display command
          INC DPTR           ;point to next adress
          DJNZ R2,BACK_MSC   ;loop 16 times


   LCALL LONG_DELAY
          RET


    ;***********************************************8


           DISP_WELCOME:

        MOV     A,#38H        ;init. LCD 2 lines,5x7 matrix
        ACALL   COMMAND       ;issue command
        MOV     A,#0CH        ;LCD on, cursor oFF
        ACALL   COMMAND       ;issue command
        MOV     A,#01H        ;clear LCD command
        ACALL   COMMAND       ;issue command
        MOV     A,#06H        ;shift cursor right
        ACALL   COMMAND       ;issue command
        MOV     A,#83H        ;cursor: line 1, pos. 3
        ACALL   COMMAND       ;command subroutine




          MOV DPTR,#WELCOME1 ;load DPTR with adress of label
          MOV R2,#16         ;loop number 16
BACK_WEL1: CLR A
          MOVC A,@A+DPTR     ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A  ;load lcd with ascii data
         ACALL   DATA_DISPLAY;call display command
          INC DPTR           ;point to next adress
          DJNZ R2,BACK_WEL1  ;loop 16 times



        MOV     A,#0C0H        ;cursor: line 2, pos. 1
        ACALL   COMMAND        ;command subroutine

          MOV DPTR,#WELCOME2  ;load DPTR with adress of label
          MOV R2,#16          ;loop number 16
BACK_WEL2: CLR A
          MOVC A,@A+DPTR      ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A   ;load lcd with ascii data
         ACALL   DATA_DISPLAY ;call display command
          INC DPTR            ;point to next adress
          DJNZ R2,BACK_WEL2   ;loop 16 times
```

```
   LCALL LONG_DELAY
         RET


    ;****************************************

          DISP_NAME:

        MOV     A,#38H        ;init. LCD 2 lines,5x7 matrix
        ACALL   COMMAND       ;issue command
        MOV     A,#0CH        ;LCD on, cursor oFF
        ACALL   COMMAND       ;issue command
        MOV     A,#01H        ;clear LCD command
        ACALL   COMMAND       ;issue command
        MOV     A,#06H        ;shift cursor right
        ACALL   COMMAND       ;issue command
        MOV     A,#80H        ;cursor: line 1, pos. 3
        ACALL   COMMAND       ;command subroutine


          MOV DPTR,#KITAVI ;load DPTR with adress of label
          MOV R2,#16       ;loop number 16
BACK_KIT: CLR A
          MOVC A,@A+DPTR    ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A ;load lcd with ascii data
         ACALL   DATA_DISPLAY;call display command
          INC DPTR          ;point to next adress
          DJNZ R2,BACK_KIT  ;loop 16 times

        MOV     A,#0C0H        ;cursor: line 2, pos. 1
        ACALL   COMMAND       ;command subroutine

          MOV DPTR,#REG   ;load DPTR with adress of label
          MOV R2,#16      ;loop number 16
BACK_REG: CLR A
          MOVC A,@A+DPTR  ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A;load lcd with ascii data
         ACALL   DATA_DISPLAY;call display command


          INC DPTR        ;point to next adress
          DJNZ R2,BACK_REG;loop 16 times



   LCALL LONG_DELAY
         RET

    ;****************************************

          SW_VERSION:

        MOV     A,#38H        ;init. LCD 2 lines,5x7 matrix
        ACALL   COMMAND       ;issue command
        MOV     A,#0CH        ;LCD on, cursor oFF
        ACALL   COMMAND       ;issue command
        MOV     A,#01H        ;clear LCD command
        ACALL   COMMAND       ;issue command
        MOV     A,#06H        ;shift cursor right
        ACALL   COMMAND       ;issue command
        MOV     A,#80H        ;cursor: line 1, pos. 3
        ACALL   COMMAND       ;command subroutine
```

```
          MOV DPTR,#SW_VER1;load DPTR with adress of label
          MOV R2,#16;loop number 16


BACK_SW_VER1: CLR A
          MOVC A,@A+DPTR;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A;load lcd with ascii data
         ACALL   DATA_DISPLAY;call display command
          INC DPTR;point to next adress
          DJNZ R2,BACK_SW_VER1;loop 16 times

        MOV     A,#0C0H        ;cursor: line 2, pos. 1
        ACALL   COMMAND        ;command subroutine

          MOV DPTR,#SW_VER2 ;load DPTR with adress of label
          MOV R2,#16        ;loop number 16
BACK_SW_VER2: CLR A
          MOVC A,@A+DPTR    ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A ;load lcd with ascii data
         ACALL   DATA_DISPLAY;call display command
          INC DPTR          ;point to next adress
          DJNZ R2,BACK_SW_VER2;loop 16 times

   LCALL LONG_DELAY
          RET

  ;****************************************

          INITIAL_0:

        MOV     A,#38H        ;init. LCD 2 lines,5x7 matrix
        ACALL   COMMAND       ;issue command
        MOV     A,#0CH        ;LCD on, cursor oFF
        ACALL   COMMAND       ;issue command
        MOV     A,#01H        ;clear LCD command
        ACALL   COMMAND       ;issue command
        MOV     A,#06H        ;shift cursor right
        ACALL   COMMAND       ;issue command
        MOV     A,#80H        ;cursor: line 1, pos. 3
        ACALL   COMMAND       ;command subroutine



          MOV DPTR,#INITIAL0 ;load DPTR with adress of label
          MOV R2,#16         ;loop number 16
BACK_INITIAL0: CLR A
          MOVC A,@A+DPTR     ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A  ;load lcd with ascii data
         ACALL   DATA_DISPLAY;call display command
          INC DPTR           ;point to next adress
          DJNZ R2,BACK_INITIAL0;loop 16 times

   LCALL LONG_DELAY0
          RET

;****************************************

          INITIAL_1:
```

```
        MOV     A,#38H        ;init. LCD 2 lines,5x7 matrix
        ACALL   COMMAND       ;issue command
        MOV     A,#0CH        ;LCD on, cursor oFF
        ACALL   COMMAND       ;issue command
        MOV     A,#01H        ;clear LCD command
        ACALL   COMMAND       ;issue command
        MOV     A,#06H        ;shift cursor right
        ACALL   COMMAND       ;issue command
        MOV     A,#80H        ;cursor: line 1, pos. 3
        ACALL   COMMAND       ;command subroutine



          MOV DPTR,#INITIAL1 ;load DPTR with adress of label
          MOV R2,#16         ;loop number 16
BACK_INITIAL1: CLR A
          MOVC A,@A+DPTR     ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A  ;load lcd with ascii data
         ACALL   DATA_DISPLAY;call display command
          INC DPTR           ;point to next adress
          DJNZ R2,BACK_INITIAL1;loop 16 times


   LCALL LONG_DELAY0
          RET

;*****************************************

          INITIAL_2:

        MOV     A,#38H        ;init. LCD 2 lines,5x7 matrix
        ACALL   COMMAND       ;issue command
        MOV     A,#0CH        ;LCD on, cursor oFF
        ACALL   COMMAND       ;issue command
        MOV     A,#01H        ;clear LCD command
        ACALL   COMMAND       ;issue command
        MOV     A,#06H        ;shift cursor right
        ACALL   COMMAND       ;issue command
        MOV     A,#80H        ;cursor: line 1, pos. 3
        ACALL   COMMAND       ;command subroutine


          MOV DPTR,#INITIAL2  ;load DPTR with adress of label
          MOV R2,#16          ;loop number 16
BACK_INITIAL2: CLR A
          MOVC A,@A+DPTR      ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A   ;load lcd with ascii data


         ACALL   DATA_DISPLAY ;call display command
          INC DPTR            ;point to next adress
          DJNZ R2,BACK_INITIAL2;loop 16 times


   LCALL LONG_DELAY0
          RET

;*****************************************

          INITIAL_3:
```

```
        MOV     A,#38H        ;init. LCD 2 lines,5x7 matrix
        ACALL   COMMAND       ;issue command
        MOV     A,#0CH        ;LCD on, cursor oFF
        ACALL   COMMAND       ;issue command
        MOV     A,#01H        ;clear LCD command
        ACALL   COMMAND       ;issue command
        MOV     A,#06H        ;shift cursor right
        ACALL   COMMAND       ;issue command
        MOV     A,#80H        ;cursor: line 1, pos. 3
        ACALL   COMMAND       ;command subroutine


          MOV DPTR,#INITIAL3;load DPTR with adress of label
          MOV R2,#16        ;loop number 16
BACK_INITIAL3: CLR A
          MOVC A,@A+DPTR    ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A ;load lcd with ascii data
         ACALL   DATA_DISPLAY;call display command


          INC DPTR          ;point to next adress
          DJNZ R2,BACK_INITIAL3;loop 16 times


   LCALL LONG_DELAY0
          RET

;*****************************************

          INITIAL_4:

        MOV     A,#38H        ;init. LCD 2 lines,5x7 matrix
        ACALL   COMMAND       ;issue command
        MOV     A,#0CH        ;LCD on, cursor oFF
        ACALL   COMMAND       ;issue command
        MOV     A,#01H        ;clear LCD command
        ACALL   COMMAND       ;issue command
        MOV     A,#06H        ;shift cursor right
        ACALL   COMMAND       ;issue command
        MOV     A,#80H        ;cursor: line 1, pos. 3
        ACALL   COMMAND       ;command subroutine


          MOV DPTR,#INITIAL4 ;load DPTR with adress of label
          MOV R2,#16         ;loop number 16
BACK_INITIAL4: CLR A
          MOVC A,@A+DPTR     ;load acc with contents of adress of DPTR
          MOV LDC_DATABUS,A  ;load lcd with ascii data
         ACALL   DATA_DISPLAY;call display command
          INC DPTR           ;point to next adress
          DJNZ R2,BACK_INITIAL4;loop 16 times



   LCALL LONG_DELAY0
          RET

;*****************************************

ORG 500H
```

```
KENYATTA: DB "    Kenyatta    "

ORG 520H
UNIVERSITY: DB "   University   "

ORG 540H
PHYSICS: DB " Physics Dept.  "

ORG 560H
MSC: DB " MSc. Project   "

ORG 580H
WELCOME1: DB "Electronic   "

ORG 600H
WELCOME2: DB "Weighing Balance"


ORG 620H
KITAVI: DB "By Kitavi Munyao"

ORG 640H
REG: DB "REG: I56-7254-02"

ORG 660H


SW_VER1: DB "Weigh_Scale.A51 "

ORG 680H
SW_VER2: DB "S/W Version 1.0 "

ORG 700H
INITIAL0: DB "Initialising    "

ORG 720H
INITIAL1: DB "Initialising.   "

ORG 740H
INITIAL2: DB "Initialising..  "

ORG 760H
INITIAL3: DB "Initialising... "


ORG 780H
INITIAL4: DB "Initialising...."

      ;*****************************************

END
```

APPENDIX F: Steps involved in the PCB Fabrication

1. Design of the board layout

The PCB track layout was drawn using ExpressPCB software.

The image was reversed, using the 'flip horizontal' function, before printing.

The image was printed on normal A4 paper to make sure that it was correct in size.

It was checked carefully.

2. Printing the negative plastic film

When the layout was done, the board layers were printed onto glossy photo papers with a laser printer, top layer on one and bottom layer on the other.

3. Exposing/developing the board

The copper clad board was cleaned with steel wool. Very fine wet sandpaper could also do.

It was dried thoroughly.

The board had to be clean and free from fingerprints.

The glossy photo papers were placed face down on the copper clad board top and bottom surfaces.

The board "image" was transferred to the bare copper board in a UV box.

Areas of the PCB exposed to UV light (through the film) turned into a protective plastic film.

3-5 minutes exposure time produced good results.

After laminating, and allowing cooling, the board with the paper stuck to it was soaked in water solution of NaOH to remove the paper, leaving only the toner behind. The developing phase took about 1-2 minutes.

The board was then showered with fresh water to remove NaOH remains.

4. The Etching process

The board was etched in a Ferric Chloride etching solution (etchant). Ammonium Persulfate could also serve the purpose. The etchant reacts with exposed copper and removes it from the PC board leaving the copper covered with resist (toner, rub-on patterns, tape, permanent marker).

This process took about10-30 minutes.

After etching, the board was rinsed under a cold tap.

The toner (etch resist) or protective plastic film was then removed with Acetone (solvent). Steel wool could also do.
The board was dried.

5. Tinning

The PC board was tinned using a soldering iron and a small piece of tinned solderwick. Tinning isn't absolutely necessary but it improves the appearance of the board, and prevents the copper from oxidizing before it's time to solder the parts to the board.

6. Drilling

A 0.8 mm PCB drill bit was used for drilling out all of the leaded component holes.
Some 1.0 mm holes were drilled out for connectors.
3.0 mm holes for corner fixings (mounting holes) were drilled.

7. **Soldering**

Finally, the copper was cleaned using a PCB rubber ready for soldering.
The components were then mounted and soldered in.

APPENDIX G: Photograph of the MCBEWB

APPENDIX H: GLOSARY

Capacity: The greatest load a weighing machine is designed to weigh.
Sometimes marked on it as ‘Max’.

Calibration: Specific types of measurement performed on measuring instruments to establish the relationship between the indicated values and known values of a measured quantity.
NB: The term ‘calibration’ as defined internationally does not
include adjustment of the instrument.

Conventional mass: For a weight taken at 20 ^{o}C, the conventional mass is the mass of a reference weight of a density of 8000 kgm^{-3} which it balances in air of a density of 1.2 kgm^{-3}.

Discrimination: The smallest change in mass that can be detected by the weighing machine.

Range: The least and greatest load for which a machine can be used, and for which continuous mass values will be displayed with the same resolution.

Repeatability: A measure of a weighing machine’s ability to display the same result when repeated measurements are made under the same weighing conditions.

Resolution: The mass value of the smallest scale or digital interval displayed by the weighing machine. Sometimes marked on it as ‘d’.

Span: The mass value of the difference between the greatest and least load for which continuous mass values will be displayed with the same resolution.

Sensitivity: The number of divisions changes in reading per unit mass.

Tare Facility: Enables the weighing machine reading to be adjusted to read zero with an object on the load receptor.

Turning point: The reading at the extremity of the swing of the pointer, i.e., where it changes its direction of motion.

Uncertainty: The amount by which a true value may differ from a measured value, at a given confidence level.

APPENDIX I: MCBEWB CIRCUIT SCHEMATIC

MCS®-51 INSTRUCTION SET

8051 Instruction Set Summary

Interrupt Response Time: Refer to Hardware description.

Instructions that affect flag settings (1)

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C, bit	X		
MUL	O	X		ANL C, /bit	X		
DIV	O	X		ORL C, bit	X		
DA	X			ORL C, bit	X		
RRC	X			MOV C, bit	X		
RLC	X			CJNE	X		
SETB C	1						

(1) Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on instruction set and addressing modes:

Rn - Register R7-R0 of the currently selected Register Bank.

Direct - 8-bit internal data locations address. This could be an internal Data RAM Locations (0-127) or a SFR (ie. 1/0 Port, control register, status register etc. (128-255).

@Ri - 8-bit internal data RAM location (o-255) addressed indirectly through register R1 or RO

data - 8-bit included in instruction.

data 1 - 16-bit constant included in instruction.

addr 16 -16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K- byte Program Memory Address space.

addr 11 - 11-bit destination address. Used by ACALL & AJMP. The branch will be within the the same 2K – byte page of program memory as the first byte of the following instruction.

rel - Signed (two's complement) 8 bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.

bit - Direct Addressed bit in Internal Data RAM or Special Function

	Mnemonic	Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A, direct	Add register byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM To Accumulator	1	12
ADD	A,#data	Add immediate Data to Accumulator	2	12
ADDC	A, Rn	Add register to Accumulator	1	12
ADDC	A, direct	Add Direct byte to Accumulator	2	12
ADDC	A, @Ri	Add indirect RAM To Accumulator with Carry.	1	12
ADDC	A, #data	Add immediate Data to Acc with carry	2	12
SUBB	A, Rn	Subtract register from Acc with borrow	1	12
SUBB	A, direct	Subtract direct Byte from ACC With borrow	2	12
SUBB	A,@Ri	Subtract indirect Byte from ACC with borrow	2	12
SUBB	A, #data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12
INC	direct	Increment direct Byte	2	12
INC	@Ri	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct Byte	2	12
DEC	@Ri	Decrement Indirect RAM	2	12

8051 Instruction Set Summary (continued)

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS (continued)				
INC	DPTR	Increment data Pointer	1	24
MUL	AB	Multiply A& B	1	48
DIV	AB	Divide A & B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A, Rn	AND register to Accumulator	1	12
ANL	A, direct	AND Direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM To Accumulator	1	12
ANL	A, #data	AND immediate Data to Accumulator	2	12
ANL	direct, A	AND accumulator To direct byte	2	12
ANL	direct, #data	AND immediate data To direct byte.	3	24
ORL	A, Rn	or register to accumulator	1	12
ORL	A, direct	OR direct byte to Accumulator	2	12
ORL	A, @Ri	OR indirect RAM To Accumulator	1	12
ORL	A, #data	OR immediate data To accumulator.	2	12
ORL	direct, A	or accumulator To direct byte	3	12
ORL	direct, #data	OR immediate data To direct byte	1	24
XRL	A, Rn	Exclusive-OR register to Accumulator	2	12
XRL	A, direct	Exclusive OR direct byte to Accumulator.	1	12
XRL	A,@Ri	Exclusive OR Indirect RAM to Accumulator	1	12
XRL	A, #data	Exclusive OR Immediate data to accumulator	2	12
XRL	direct, A	Exclusive OR Accumulator to direct byte	2	12
XRL	direct, #data	Exclusive OR Immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS (continued)				
RL	A	Rotate Accumulator left	1	12
RLC	A	Rotate Accumulator left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swab Nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A, direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12
MOV	A, #data	Move immediate data to Accumulator	2	12
MOV	Rn, A	Move Accumulator to register	1	12
MOV	Rn, direct	Move direct byte to register	2	24
MOV	Rn, # data	Move immediate data to register	2	12
MOV	direct, A	Move Accumulator to direct byte	2	12
MOV	direct, Rn	Move register to direct byte	2	24
MOV	direct, direct	Move direct byte to direct	3	24
MOV	direct,@ Ri	Move indirect RAM to direct byte	2	24
MOV	direct, #data	Move immediate data to direct byte	3	24
MOV	@ Ri, A	Move accumulator to indirect RAM.	2	12

8051 Instruction Set Summary (Continued)

Mnemonic	Description	Byte	Oscillator Period
DATA TRANSFER (Continued)			
MOV @Ri,direct	Move direct byte to indirect RAM	2	24
MOV @Ri,# data	Move immediate data to indirect RAM	2	12
MOV DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC A,@A+DPTR	Move code byte relative to DPTR to Acc	1	24
MOVC A,@A+PC	Move code byte relative to PC to Acc	1	24
MOVX A,@Ri	Move External RAM(8-bit addr)to Acc	1	24
MOVX A,@DPTR	Move external RAM(16-bit addr)to Acc	1	24
MOVX @Ri,A	Move Acc to External RAM(8-bit addr)	1	24
MOVX @DPTR,A	Move Acc to External RAM(16-bit addr)	1	24
PUSH direct	Push direct byte onto stack	2	24
POP direct	Pop direct byte from stack	2	24
XCH A,Rn	Exchange register with accumulator	1	12
XCH A,DIRECT	Exchange direct byte with Accumulator	2	12
XCH A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD A,@Ri	Exchange low order Digit indirect RAM with Acc	1	12

Mnemonic	Description	Byte	Oscillator Period
BOOLEAN VARIABLE MANIPULATION			
CLR C	Clear Carry	1	12
CLR bit	Clear direct bit	2	12
SETB C	Set Carry	1	12
SETB bit	Set direct bit	2	12
CPL C	Complement Carry	1	12
CPL bit	Complement direct bit	2	12
ANL C,bit	AND direct bit to CARRY	2	24
ANL C,/bit	AND complement of direct bit to Carry	2	24
ORL C,bit	OR direct bit to carry	2	24
ORL C,/bit	OR complement of direct bit to Carry	2	24
MOV C,bit	Move direct bit to carry	2	12
MOV bit,C	Move carry to direct bit	2	24
JC rel	Jump if Carry is set	2	24
JNC rel	Jump if Carry is not set	2	24
JB bit,rel	Jump if direct Bit is set	3	24
JNB bit,rel	Jump if direct Bit is not set	3	24
JBC bit,rel	Jump if direct Bit is set &clear bit	3	24
PROGRAM BRANCHING			
ACALL addr11	Absolute Subroutine Call	2	24
LCALL addr16	Long Subroutine Call	3	24
RET	Return from Subroutine	1	24
RET 1	Return from interrupt	1	24
AJMP addr11	Absolute Jump	2	24
LJMP addr16	Long Jump	3	24
SJMP rel	Short Jump(relative addr)	2	24