DESIGN AND CONSTRUCTION OF A MICROCONTROLLER-BASED TEMPERATURE CONTROLLER WITH DIGITAL DISPLAY

BY

ADIDE SAMUEL EMELIS
2004/18813EE

A Thesis submitted to the Department of Electrical and Computer Engineering, in partial fulfillment of the requirement for the award of Bachelor of Engineering (B.Eng), School of Engineering and Engineering Technology, Federal University of Technology, Minna, Niger state.

DECEMBER, 2009

DEDICATION

This project is dedicated to my parents, Mr. and Mrs. Emelis for all their love and support throughout the period of my studies.

DECLARATION

I, ADIDE SAMUEL EMELIS, declare that this work was done by me and has never been presented elsewhere for the award of a degree. I also hereby relinquish the copyright to the Federal University of Technology, Minna.

ADIDE SAMUEL EMELIS
(Name of Student)

25/01/10
(Signature and Date)

ENGR. J.A. ABOLARINWA
(Supervisor)

22/01/2010
(Signature and Date)

for ENGR. DR. Y.A. ADEDIRAN
(Head Of Department)

(May 6, 2010)
(Signature and Date)

DR. (MRS) B.A. ADEBIYI
(External Examiner)

(Signature and Date)

ACKNOWLEDGEMENT

I will like to give thanks to God almighty for His grace, helping me to commence and complete my undergraduate studies. He has been my help throughout my school years. It hasn't been so easy, but God has kept me. I am so hopeful and have this assurance that He will yet keep me and grant me success throughout life.

I will also use this medium to show my deep and profound gratitude to my wonderful parents, Mr. and Mrs. E.A Emelis, for their contributions to my success. I also want to thank my siblings and friends for the wonderful encouragements they gave me in the course of my studies. It is my prayer that God would bless you and continue to guide you and meet you all at your various points of needs, Amen.

Finally, I want to thank my project supervisor, Engr. J.A. Abolarinwa, for his uncompromising stand and guidance, which aided me in realizing this project. May God bless you and help you always in the course of your career, Amen.

ABSTRACT

This project involves the design and construction of a simple temperature-controlling device using a microcontroller. The whole system incorporates a microcontroller which is interfaced to other components. These components include a temperature sensor, an analogue-to-digital converter (ADC) and the LCD. The temperature sensor used is an integrated circuit (LM 35). This IC senses temperature and generates an analogue voltage that is proportional to the temperature it senses as its output. This analogue output is then converted to its digital equivalent by the ADC. The microcontroller through the program code written to it then converts the binary values to its equivalent temperature in degree centigrade and displays the values on the LCD, it also sends an instruction to switch off a heating element if the preset temperature is exceeded.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

1.1 INTRODUCTION

From time immemorial, temperature and its measurement has been of great importance to man. Temperature affects all spheres of life; it is evident in our daily lives- the way we dress, eat, e.t.c.

Temperature is a measure of hotness or coldness and is measured using a thermometer[2]. Temperature control on the other hand is a process in which change of temperature of a space (and objects collectively there within) is measured or otherwise detected and the passage of heat energy into or out of the space is adjusted to achieve a desired average temperature. [4]

Since temperature is the degree of hotness or coldness of a body or system, it is very fundamental in the functionality of the system. This is due to the fact that most biological, chemical, electronic and physical systems are affected by temperature. Some processes work satisfactorily only within a specific range of temperature, thus a temperature-monitoring device is incorporated into such processes to regulate the temperature. [5]

This project aims at achieving a temperature-monitoring device that switches off automatically [using a microcontroller] when the preset temperature is exceeded.

1.2 AIMS AND OBJECTIVES

This project is aimed at designing and constructing a simple temperature-regulating device which senses temperature [using an LM 35DZ sensor], compares it with an already preset temperature range and then sends an instruction via a programmed microcontroller to switch off a heating element.

The project aims at improving on previous efforts by the addition of a keypad which can be used to vary temperature ranges at either high or low modes. Also, LCD display is used in this project as against LED display used in previous efforts. This ensures less power consumption.

1.3 METHODOLOGY

The system is made up of a 5V power supply, LM35DZ temperature sensor, analogue-to-digital converter (ADC 0804), 89S52 microcontroller and LCD display.

In the power supply unit, a 240V/12V transformer was interposed between the bridge rectifier and the 240V A.C supply. The rectified voltage was smoothened by a 2200μF capacitor. The voltage was regulated by a 5V voltage regulator (LM 7805).

The LM 35 sensor sensed the ambient temperature and converted it to analogue output voltage. This analogue output was converted to its digital equivalent by the analogue-to-digital converter. The microcontroller, through the program codes written to it converts the digital output to degree Celsius and compares it with the preset temperature values. It then sends a pulse to switch off the heating element.

The LCD display now displays the outputs of the microcontroller for visual observation. This is made possible by the connections between the LCD and the microcontroller circuitry as explained in chapter three.

1.4 SCOPE AND LIMITATIONS

The project is designed to accommodate a temperature range of 0 degrees Celsius to 100 degrees Celsius. Thus, it would not be applicable to temperatures higher than 100 degrees Celsius or lower than 0 degrees Celsius.

Also, the circuit runs on PHCN supply; so once the supply goes off, the whole circuit shuts down. It is therefore necessary to ensure constant power supply.

Also, the circuit runs on PHCN supply, so once the supply goes off, the whole circuit shuts down. It is therefore necessary to ensure constant power supply.

1.5 PROJECT OUTLINE

Chapter one gives a general introduction of what the project is all about, its aims and objectives, methodology and scope of the project.

The previous work done with respect to this project is discussed as the literature review of chapter two.

Chapter three covers the various steps taken in the design and implementation of the project work and calculations involved.

The tests carried out, results obtained and discussion of such results come under chapter four.

The conclusion and recommendation constitutes chapter five.

CHAPTER TWO

LITERATURE REVIEW

2.1 TEMPERATURE HISTORY

Temperature is by far the most measured parameter. It impacts the physical, chemical and biological world in numerous ways. However, a full appreciation of the complexities of temperature and its measurement has been relatively slow to develop.

Intuitively, people have known about temperature for a long time: fire is hot and snow is cold. The realization that some things are hotter than others and that some are colder must have long preceded the concept of temperature in the history of humans. Although ancient cooks and bakers, smelters and smiths, potters and ceramists did not know the distinction between heat and temperature, their works were involved in temperature. We can tell a toddler some things are too hot and others are cold even when we don't know the temperature of such things.

Temperature may be said to have been derived from the recognition of the need to define these differences more precisely than by the positive, comparative and superlative of the adjectives "hot" and "cold".

Aristotle postulated his four qualities: the hot, the cold, the moist and the dry; set up a conception which endured for about two thousand years.

The ideas of Aristotle were adopted by Galen (A.D. 130-200), who was the first man to describe heat and cold by a number about fifteen hundred years ago. The word temperature originated from temper, after Galen determined the "complexion" of a person by the proportion in which the above four qualities were tempered. He gave the earliest notion of a standard of temperature, although the temperature meant the tempering of the qualities in a substance until the eighteenth century [6].

Greater knowledge was gained as man attempted to work with metals through the bronze and iron ages. Most of these technological processes required some degree of regulation over temperature, but in order to regulate, the ability to measure what you are regulating is required.

Record shows that researches and implementation on thermostatic control systems did not seriously take off until the '80s when the need became very important to control systems automatically

In the first century, the Romans had developed the "hypocaust" which provided a better way to heat a room [8]. The hypocaust was a basement with a low arched vault furnace made of brick or stone. The hot air from the furnace passed through tile flues in the room above the furnace, heating the room.

In 1743, Benjamin Franklin invented the iron Franklin stove and this provided a new way to heat rooms. The fire was contained in combustion chambers usually made of ceramic or iron sections. It heated the stove walls which in turn heated the room.

As technology advanced, the word "thermostat" was introduced in 1930 by Andrew Ure, a Scottish professor of chemistry who issued a patent on what he called a "heat-responsive element" consisting of a bar of steel united to zinc by numerous rivets. [9]. This bimetallic bar bends with temperature change because of the different expansion rates of the metal strips, and the bendings can be used to actuate values to control heating systems

These days, thermostats used in homes or small buildings are usually low-voltage electric controls that turn ON or OFF as required. [10]

The concept of temperature has been the most difficult of the common properties of matter to define clearly. It is helpful to review the historical development of temperature

measurement because only after crude methods of temperature measurements were developed could the concept of temperature really be defined.

2.1.1 EARLY TEMPERATURE MEASUREMENT

The concept of temperature, as the scientific principles on which temperature measurement is based, evolved as a part of the development of science. The instrument by which the temperature of bodies is measured is called a thermometer, and the method of constructing and using thermometers is called thermometry.

The history of thermometry was a part of the history of science. The history gives insight into the meaning of temperature and its measurement units, as well as some background regarding other concerns which need to be met.

The ancient Greek knew the expansion of air by heat long time ago. The earliest writings concerning this phenomenon were the works of Philo of Byzantium and Heron of Alexandria. Due to the intercourse of culture, their works were translated into Arabic and Latin.

Della Porta was an important disseminator whose chief work Magia Naturalis (natural magic, 1558, 1589) not only included many classical antiquity and tradition, but also originated some experiments and contrivances. A simple air thermoscope was described, which traps air in a bulb so that the air expands or contracts in response to a temperature increase or decrease, it moves a liquid column in a long tube. Thermoscope is not a thermometer but is its predecessor. The logical distinction between the two is that the thermo meter possesses a scale, while a thermoscope does not [12].

The history of knowledge decided the earliest thermometer was the air thermometer after the air thermoscope appeared. There were four inventors of thermometer at the first few years of the seventeenth century, they were Galileo, Santorio, Fludd and Drebbel.

Until about 260 years ago, temperature measurement was very subjective. For hot metals, e colour of the glow was a good indicator. For intermediate temperatures, the impact on rious materials could be determined. For example, does the temperature melt sulphur, lead, or ax, or boil water? In other words, a number of fixed points could be defined, but there was no ale or any way to measure the temperature between these points. It is however possible that ere is a gap in the recorded history of technology in this regard as it is difficult to believe that e Egyptians, Assyrians, Greeks, Romans, or Chinese did not measure temperatures in some ay.

Galileo invented the first documented thermometer in about 1592. It was an air thermometer consisting of a glass bulb with a long tube attached. The tube was dipped into a cooled liquid and the bulb was warmed, expanding the air inside. As the air continued to expand, some of it escaped. When the heat was removed, the remaining air contracted causing the liquid to rise in the tube and indicating a change in temperature. This type of thermometer is sensitive, but is affected by changes in atmospheric pressure [13].

By modern standard however, the first physiologist Santorio Santorii (Sanctorius) of Padua would be regarded as the discoverer of the thermometer, as he published the earliest account of it in part three of his commentaria in artem Medicinalem Galenis (Venice 1612), of which the imprimatur is dated 1611. Santorio claimed he had adapted thermometer from Heron. He used the instrument to estimate the heat of a patient's heart by measuring the heat of the expired air.

His method of measurement was quite different from that of modern physicians. He measured the rate of change of the temperature of the thermometer by observing the distance through which the liquid fell during ten beats of a pulsilogium, a small pendulum. The result

depended on not only on the patient's temperature, but on the rapidity of his peripheral circulation. That was a fast indicator of fever because the ordinary thermometer took much longer to attain the temperature of a normal person than that of a fever patient.

2.1.2 THE EIGHTEENTH CENTURY: CELSIUS AND FAHRENHEIT

By the early 18th century, as many as 35 different temperature scales had been devised. In 1714, Daniel Gabriel Fahrenheit invented both the mercury and the alcohol thermometer. Fahrenheit's mercury thermometer consists of a capillary tube which after being filled with mercury is heated to expand the mercury and expel the air from the tube. The tube is then sealed, leaving the mercury free to expand and contract with temperature changes. Although the mercury thermometer is not as sensitive as the air thermometer, by being sealed, it is not affected by the atmospheric pressure. Mercury freezes at -39 degrees Celsius, so it cannot be used to measure temperature below this point. Alcohol, on the other hand, freezes at -113 degrees Celsius, allowing much lower temperatures to be measured.

At the time, thermometers were calibrated between the freezing point of salted water and the human body temperature. Salt added to crushed wet ice produced the lowest artificially created temperatures at the time. The common Flemish thermometers of the day divided this range into twelve points. Fahrenheit further subdivided this range into ninety-six points, giving his thermometers more resolution and a temperature scale very close to today's Fahrenheit scale.

Later in the 18th century, Anders Celsius realized that it would be advantageous to use more common calibration references and to divide the scale into 100 increments instead of 96. He choose to use one hundred degrees as the freezing point and zero degrees as the boiling point of water. Sensibly, the scale was later reversed and the centigrade scale was born.

2.1.3 THE NINETEENTH CENTURY: A PRODUCTIVE ERA

The early 1800's were very productive in the area of temperature measurement and understanding. William Thomson (later Lord Kelvin) postulated the existence of an absolute zero. Sir William Hershel discovered that when sunlight was spread into a colour swath using a prism, he could detect an increase in temperature when moving a blackened thermometer across the spectrum of colours. Hershel found that the heating effect increased toward and beyond the red in the region we now call 'infrared'. He measured radiation effects from fires, candles and stoves, and deduced the similarity of light and radiant heat. However, it was not until well into the following century that this knowledge was exploited to measure temperature.

In 1821, T J. Seebeck discovered that a current could be produced by unequally heating two junctions of two dissimilar metals- the thermocouple effect. Seebeck assigned constants to each type of metal and used these constants to compute total amount of current flowing. Also, in 1821, Sir Humphrey Davy discovered that all metals have a positive temperature coefficient of resistance and that platinum could be used as an excellent temperature detector (RTD). These two discoveries marked the beginning of serious electrical sensors.

Gradually, the scientific community learnt how to measure temperature with greater precision. For example it was realized by Thomas Stevenson that air temperature measurement needed to occur in a space shielded from the sun's radiation and rain. For this purpose he developed what is now known as the Stevenson screen. It is still in wide use.

The late 19th century saw the introduction of bimetallic temperature sensor. These thermometers contain no liquid but operate on the principle of unequal expansion between two metals. Since different metals expand at different rates, one metal that is bonded to another will bend in one direction when heated and will bend in the opposite direction when cooled (hence

he term Bimetallic Thermometer or BiMets). This bending motion is transmitted, by a suitable echanical linkage, to a pointer that moves across a calibrated scale. Although not as accurate as quid in glass thermometers, BiMets are more hardy, easy to read and have a wider span, aking them more ideal for many industrial applications.

2.1.4 THE TWENTIETH CENTURY: FURTHER DISCOVERY, REFINEMENT AND RECOGNITION

The 20th century has seen the discovery of semiconductor devices such as the thermistor e integrated circuit sensor, a range of non-contact sensors and also fibre-optic temperature sensors. Also, Lord Kelvin was finally rewarded for his early work in temperature measurement The increments of the Kelvin scale were changed from degrees to Kelvins. Now we no longer say "one hundred degrees Kelvin;" we instead say "one hundred Kelvins". The centigrade scale was changed to the Celsius scale in honour of Anders Celsius.

The 20th century also saw the refinement of the temperature scale. Temperatures can now be measured to within about 0.001 degrees Celsius over a wide range, although it is not a simple task.

2.1.5 USE OF COMPUTERS AND COMPUTER ACCESSORIES IN CONTROL

The application of computer control has really grown tremendously and has been brought about largely by the rapid advances in design of hardware and cost reduction. In recent years, digital control application to industrial and other processes has gained much ground. This can be properly demonstrated by the extent to which microcontrollers and other computer accessories have become an everyday component of very wide range of electronic systems. These days, microcontrollers can be programmed to execute various instructions, thereby making life much easier.

2.2 TYPES OF THERMOSTATIC CONTROLS

There are five types of thermostatic controls: electromechanical, digital, hybrid, light-sensing and microcontroller-based thermostats.

2.2.1 ELECTROMECHANICAL (EM) THERMOSTATS

These are usually the easiest thermostats to operate. They typically have manual controls such as moveable tabs and sliding levers. These thermostats work with most conventional heating and cooling systems. EM controls have limited flexibility and can store only the same settings for each day, and as such, are best suited for regular schedule.

2.2.2 DIGITAL THERMOSTATS

These thermostats are identified by their LED or LCD digital readout and data entry pad or buttons. They offer the widest range of features and flexibility. Digital thermostats can be used with most heating and cooling systems. They provide precise temperature control and permit custom schedule.

2.2.3 HYBRID THERMOSTATS

Hybrid thermostats combine the technology of digital control with manual slides and knobs to simplify use and maintain flexibility. Hybrids are available for most systems which include heat pumps.

2.2.4 LIGHT-SENSING HEAT THERMOSTATS

These thermostats rely on lighting level as preset by the user to activate the heating system. When lighting is reduced, a photocell inside the thermostat senses unoccupied conditions and allows space temperature to fall below the occupied temperature setting. When the lighting level increases to normal, the temperature automatically adjusts to comfort conditions.

2.2.5 MICROCONTROLLER-BASED THERMOSTATS

These make use of microcontroller as the main control unit. It helps to control so many appliances such as incubators, microwave oven, heater e.t c. This is what this project is centered on.

2.3 THEORETICAL BACKGROUND

A temperature controller is a device that regulates temperature automatically with the aid of a micro-controller. It consists of a temperature-sensing unit, an analogue-to-digital converter, micro-controller and the power supply unit.

2.3.1 TEMPERATURE SENSOR

A temperature sensor is a device that senses temperature variation in an environment to give useful electrical signals. Sensors can also be called transducers because they convert one physical quantity to another physical quantity.

For the purpose of this project, the LM 35 temperature sensor is used. It is a precision integrated-circuit temperature sensor whose output voltage is linearly proportional to the Celsius temperature. It has an advantage over linear temperature sensors calibrated in degree Kelvin as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling.

It also does not require any external calibration to provide typical accuracies of ± ¼°c at room temperature and ± ¾°c over a full −55° to +150°c temperature range.

2.3.2 ANALOGUE-TO-DIGITAL CONVERTER

This is a circuit that converts an input analogue voltage to its digital equivalent. The ADC used for this project is the ADC 0804 which is a CMOS 8-bit successive-approximation analogue-to-digital converter that uses a modified potentiometric (256R) ladder. It is designed to

operate from common micro-controller buses, with the three-state output latches driving the data bus. It can be made to appear to the microprocessor as a memory location or an input/output port, hence no interfacing logic is required.

2.3.3 MICROCONTROLLER

Microcontroller can be seen as a "special purpose computer" that can run any of thousands of programs. It consists of a central processing unit (CPU), instruction decoder, arithmetic logic unit and register. [11]

The microcontroller is embedded inside some other device (like the temperature controller) in order to control the features or actions of the device. Microcontrollers are usually dedicated to one task and run one specific program. It takes input from the device it is controlling and controls the device by sending signals to different components in the device.

For the purpose of this project, an 89S52 microcontroller is used. It is a low-power, high performance CMOS 8-bit microcontroller with 8 kilobytes of in-system programmable flash memory. It provides a highly flexible and cost-effective solution to many embedded control applications.

CHAPTER THREE

SYSTEM DESIGN AND ANALYSIS

This chapter will describe the design and implementation of the project. The chapter is divided into two sections, which are further divided into sub-sections. The two sections are the hardware section and the software section.

3.1 HARDWARE SECTION

This section comprise of the temperature sensor, analogue -to-digital converter, microcontroller, and display unit. The block diagram of the system is shown in figure 3.1

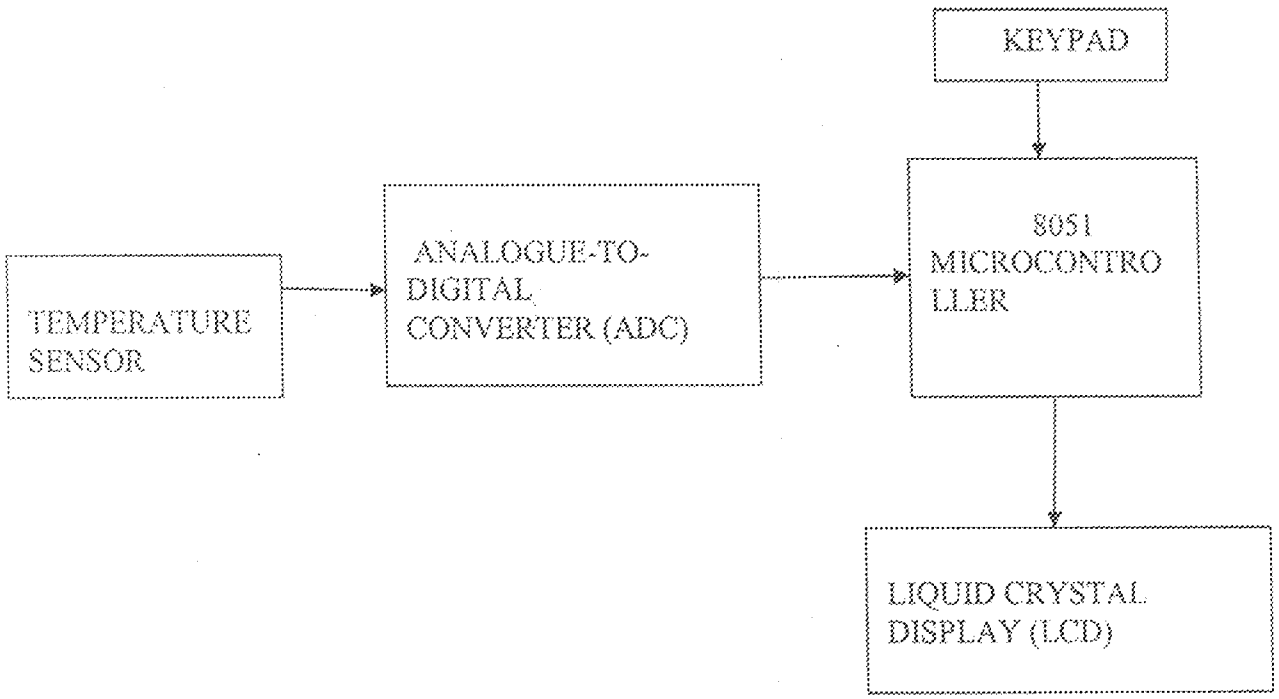

Fig 3.1 Block diagram of hardware section

From the block diagram above, temperature is converted to analogue voltage by the temperature sensor unit. The analog voltage is converted to digital signal by the ADC and sent to the microcontroller. The microcontroller processes and displays the temperature on the LCD.

3.1.1 TEMPERATURE SENSOR UNIT

This unit converts a non electrical quantity (temperature) to an electrical signal (voltage). To achieve this, a temperature sensor is used. One common temperature sensor in the market is an LM35DZ. There are other temperature sensing components like LM334, DS1820, thermistors, etc. This project will make use of the LM35DZ; this is because of the availability and the range of temperature it can handle.

The LM35DZ is a precision semiconductor temperature sensor giving an output of 10mV per degree centigrade. Unlike devices with output proportional to absolute temperature in degree centigrade, there is no large offset, voltage which in most application has to be removed. The features and voltage characteristic include the following.

1. Accuracy of ¼ degree centigrade at room temperature
2. Supply voltage from +2V dc to 35V dc
3. Output voltage from +6V dc to -1.0V dc
4. Output is proportional to degree centigrade from 0 °C to 100 °C

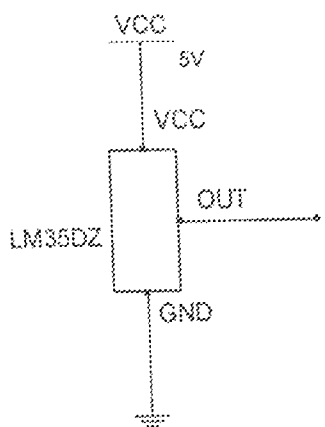

Fig 3.2: Symbol of an LM35DZ

1.2 ANALOGUE- TO- DIGITAL CONVERTER (ADC) UNIT

The ADC unit is necessary to convert the analogue voltage from the LM35 to digital voltage, so that it can be fed into a microprocessor or a microcontroller. There are different types of Analogue to Digital Converter (ADC), this project will make use of an ADC0804. This is because of its simplicity in interfacing to microcontrollers as well as availability.

The ADC0804 is a CMOS 8-bit successive approximation A/D converter that uses a differential potentiometric ladder-similar to the 256R products. This converter is designed to allow operation with the NSC800 and INS8080A derivative control bus with TRI-STATE output latches directly driving the data bus. The ADC0804 appear like memory locations or I/O ports to the microprocessor or microcontroller and no interfacing logic is needed. Differential analog voltage inputs allow increasing the common-mode rejection and offsetting the analog zero input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution. Features of the ADC0804 include the following:

1. Compatible with microprocessor or microcontroller with no interfacing logic needed, access time - 135 ns.
2. Easy interface to all microprocessors, or operates as a standalone device.
3. Differential analog voltage inputs
4. Logic inputs and outputs meet both MOS and TTL voltage level specifications
5. On-chip clock generator
6. 0V to 5V analog input voltage range with single 5V supply

The circuit below shows how an ADC0804 can be configured when connected to a microcontroller.

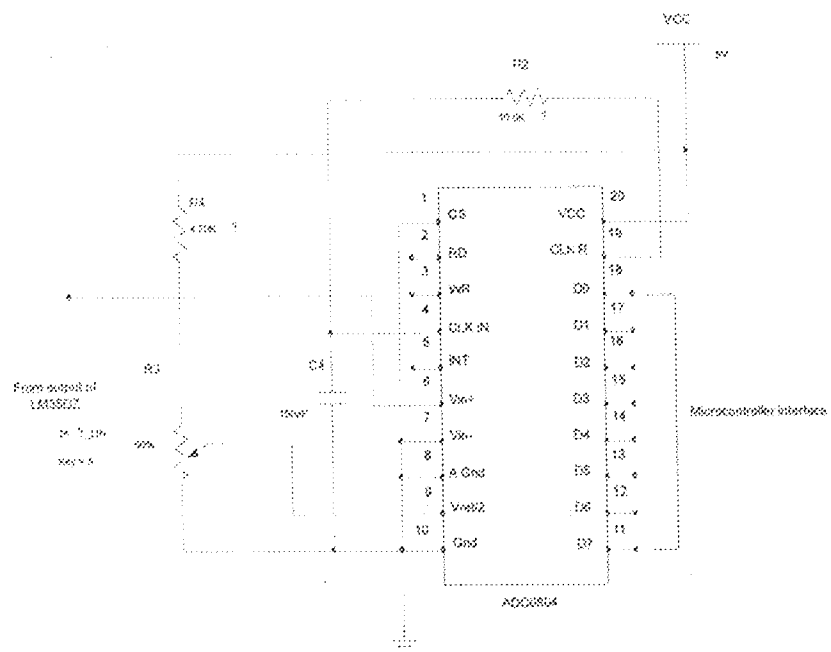

Fig.3.3 ADC hardware configuration circuitry

The variable resistor (2K ohms) is used to adjust the reference voltage to 1.275 V dc. Recall that the analog voltage from the LM35DZ changes by 10mV per degree. Therefore for the ADC to respond to a 10mV change, the step size of the ADC has to be 10mV. Mathematically;

$$step\ size = \frac{2Vref}{255} \quad \ldots\ldots 3.1$$

$$Vref = \frac{step\ size \times 255}{2} \quad \ldots\ldots 3.2$$

$$where\ step\ size = 0.01$$

$$Vref = \frac{0.01 \times 255}{2} \quad \ldots\ldots 3.3$$

$$Vref = 1.275V$$

Therefore the voltage that will be supplied to the reference pin (pin 9) of the ADC will be 1.28V approximately. This can be achieved by using a variable resistor. Mathematically;

$$V1 = 1.49\text{v}$$

$$V1 = \frac{R1}{R1+R2} \times V_R \quad \ldots\ldots 3.4$$

$$1.49 = \frac{R1}{R1+R2} \times 5V \quad \ldots\ldots 3.5$$

$$1.49 = \frac{R1}{R1+4.7K} \times 5\text{V} \quad \ldots\ldots 3.6$$

$$1.49R1 + 7.003 = 5R1 \quad \ldots\ldots 3.7$$

$$7.003 = 5R1 - 1.49R1 \quad \ldots\ldots 3.8$$

$$7.003 = 3.51R1 \quad \ldots\ldots 3.9$$

$$R1 = 7.003/3.51 \quad \ldots\ldots 3.10$$

$$R1 = 1.99 \simeq 2.0$$

$R1 = 2K$ (which is the variable resistor)

This is then adjusted to 1.28v which is the Vref

3.1.3 MICROCONTROLLER UNIT

The microcontroller unit is the heart of the entire project. It consists of a microcontroller and some configuration component. The microcontroller used for the project is an 8051 microcontroller from Atmel Corporation (AT89S52). Features of the AT89S52 include:

1. Compatible with MCS-51™ Products
2. 8K Bytes of In-System Reprogrammable Flash Memory
3. Endurance: 1,000 Write/Erase Cycles
4. Fully Static Operation: 0 Hz to 24 MHz
5. Three-level Program Memory Lock
6. 256 x 8-bit Internal RAM

7. 32 Programmable I/O Lines

8. Three 16-bit Timer/Counters

9. Eight Interrupt Sources

10. Low-power Idle and Power-down Modes

The AT89S52 needs to be hardware configured. The circuit below shows how this can be done.

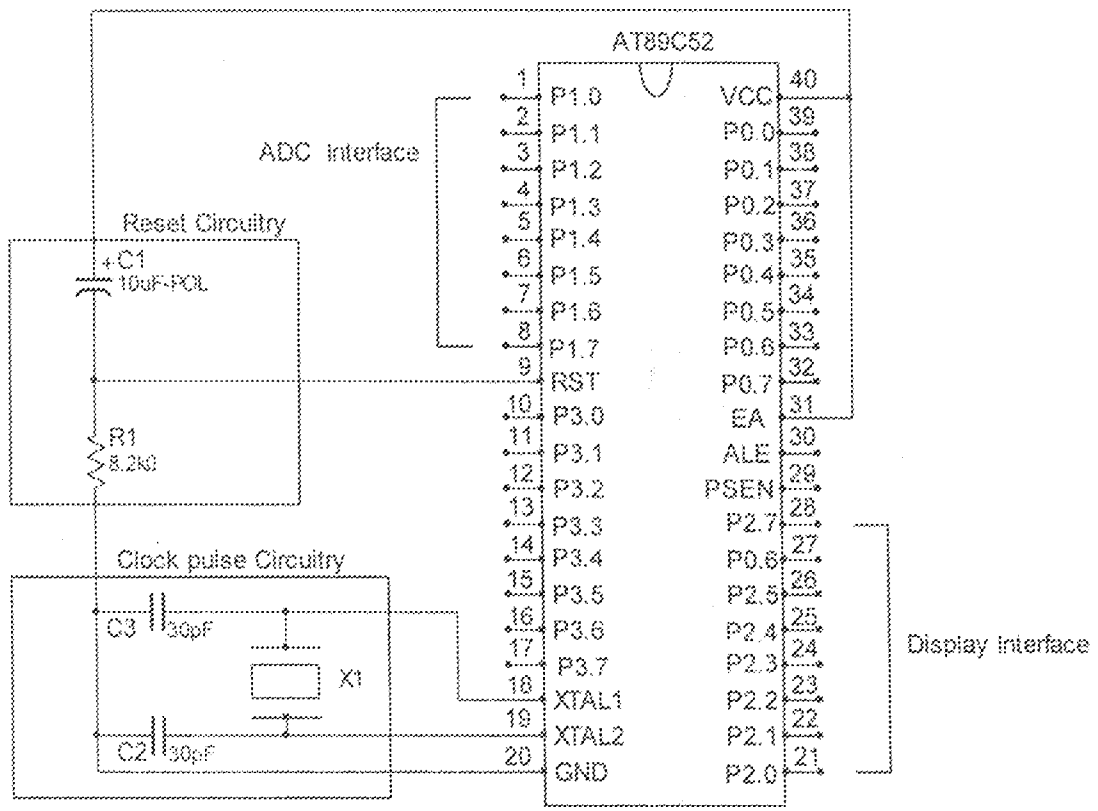

Fig3.4: AT89S52 hardware configuration circuitry

The reset circuitry is needed for manual resetting of the microcontroller to the initial start-up state. The values for R1 and C1 are 8K ohm and 10uf respectively. These values where recommended by the manufacturer in the datasheet of the AT89S52.

The clock circuitry provides the necessary clock pulse for the microcontroller to execute its instructions. The conditions for selecting the values in the clock circuitry are

$$C2 = C3 = 30 \pm 10pF$$
$$X1 \leq \text{Speed of the microcontroller(24Mhz)}$$

Recommended values are 30pF and 11.0592 MHz With the 11.0592 MHz crystal, the number of instruction the AT89S52 can executes in one second is represented mathematically is shown below;

$$\text{Numberof instructions perseconds} = \frac{\text{Crystalvalue(Hz)}}{12} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.11$$
$$= \frac{11059200}{12}$$
$$= 921600$$

The AT89C52 microcontroller will not function without a program loaded into its program memory. The program development for the AT89S52 will be discussed later..

3.1.4 LIQUID CRYSTAL DISPLAY (LCD) UNIT

This unit is used for displaying the status of the project. This project will make use of a 16 by 2 Line character LCD (liquid crystal display). The Pin out of the LDC is shown below.

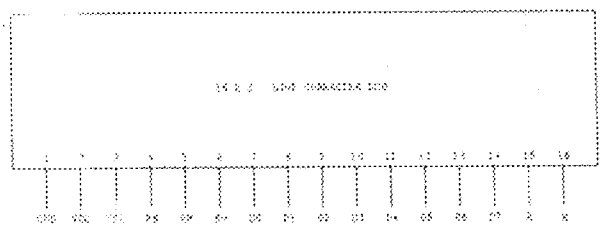

Fig3.5: LCD pin out.

For further information about the LCD, check the data sheet.

3.1.5 POWER SUPPLY UNIT

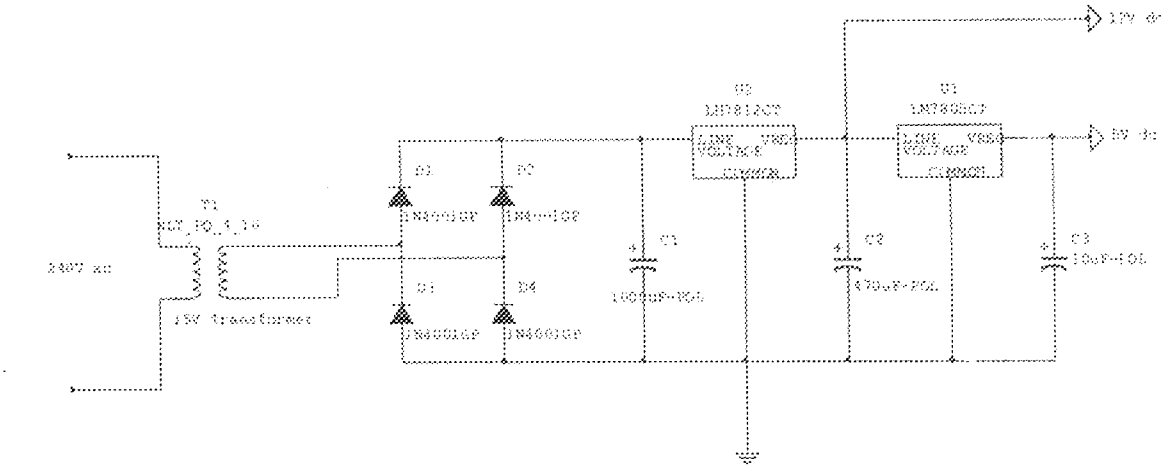

Fig 3.6: Power supply circuitry

From the circuitry above, the AC power supply was transformed from 240Vac to 24V ac. The transformed voltage was the rectified to DC using a bridge rectifier and filtered using a 2200µF capacitor. The rectified dc was then regulated to 12V dc using a 12V regulator (LM7812). A 5V regulator (LM7805) is connected to the output of the LM7812 to generate the 5V needed by the microcontroller.

3.2 SOFTWARE SECTION

This section will describe the step taken in programming the AT89S52. The section is divided into two sub-sections, the program development section and the program loading section.

3.2.1 PROGRAM DEVELOPMENT

The temperature regulator program was written in assembly language and developed using Kiel C51 uVision3 Studio.

The assembly code for the program is included in the Appendix A. Timer1 is used to measure the time interval that the microcontroller reads the ADC. Recall that Timer1 is a 16 bit timer meaning that it can count up to 65536 values. The ADC is read at an interval rate of

approximately 0.5 seconds. This time interval is also used in updating the LCD with the temperature value. To achieve this time interval, the following was adopted mathematically,

$$\text{Numberof instructions per seconds} = \frac{\text{Crystalvalue(Hz)}}{12} \quad \ldots\ldots 3.12$$

$$= \frac{11059200}{12}$$

$$= 921600$$

This means that at 11.0592 MHz, the AT89S52 microcontroller can execute 921600 instructions in one second.

Therefore the number of instructions the AT89S52 can execute in 0.5 seconds is shown mathematically below.

$$\text{Number of instructions at 0.5 seconds} = 921600 \times 0.5$$

$$= 460800$$

Remember that Timer1 can only count up to 65536. Dividing 460800 by 65536 will give the numbers of time Timer1 will count 460800.

The ADC was read for 8 timer1 overflows (an overflow is when the timer has counter to the maximum value and reset to zero). Therefore the actual time interval the AT89S52 reads the ADC is shown mathematically below.

$$\text{Numbers of instructions} = 65536 \times 8$$

$$= 524288$$

$$\text{therefore, the time interval in seconds} = \frac{524288}{921600} \quad \ldots\ldots 3.13$$

$$= 0.5688889 \text{ seconds}$$

3.2.2 PROGRAM LOADING

The program was loaded in to the AT89S52 using an 8051 programmer manufactured by Electrohive Embedded System, Nigeria. The 8051 programmer connects to the personal Computer via a serial port (COM1).

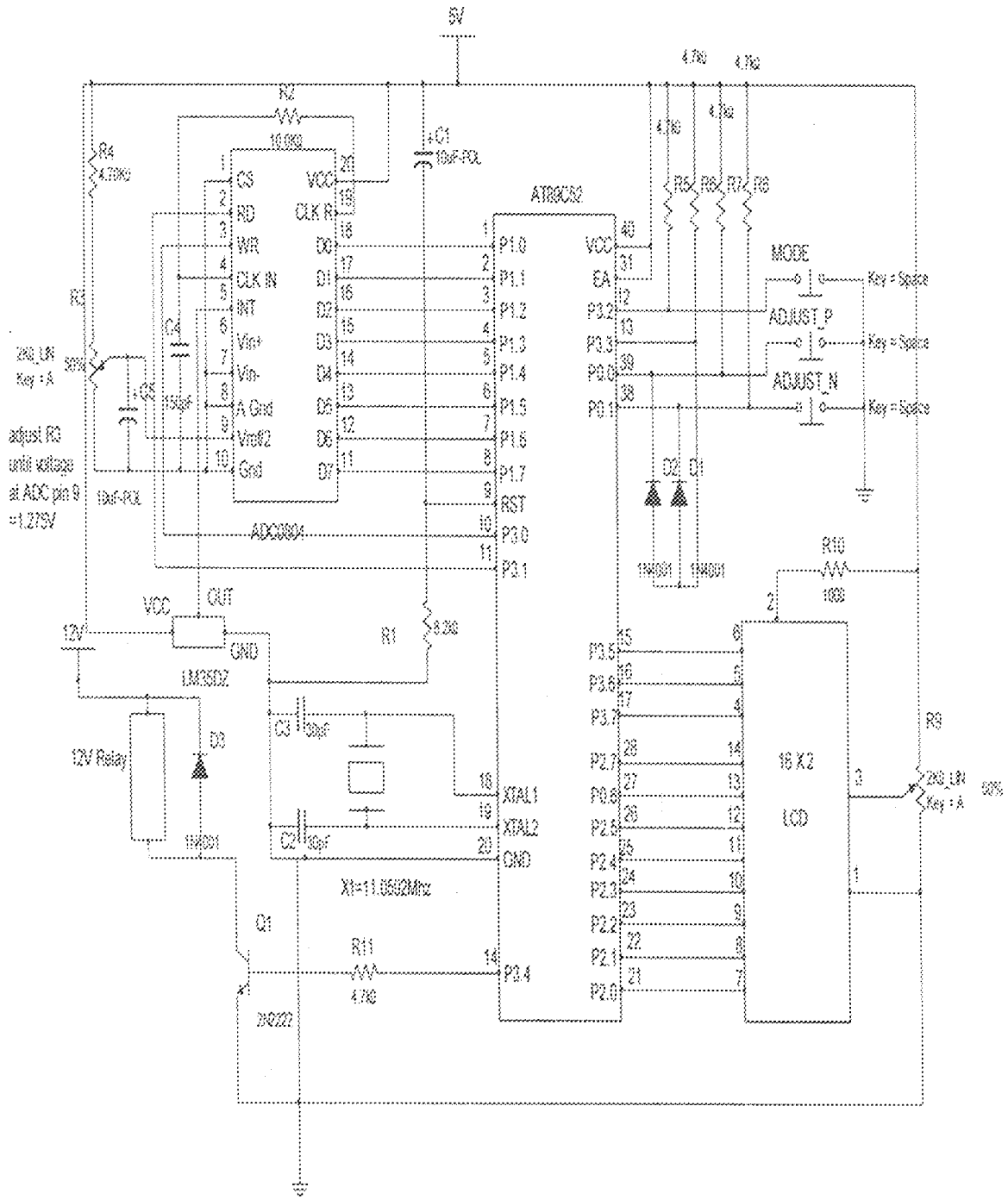

Fig 3.8: Circuit diagram of the design

CHAPTER FOUR

CONSTRUCTION, TESTING AND RESULT

4.1 PROJECT CONSTRUCTION

The construction of the temperature controller was done in phases. First, the required components were inspected and tested to ensure they had the correct ratings. They were subsequently mounted on the breadboard according to the circuit diagram.

The circuit was then tested and transferred to the Vero board after thorough cleaning of the surface of the Vero board. Modular approach was adopted and each module was separately and independently soldered on a sectioned Vero board.

The modular approach allowed each sub circuit to be soldered and tested appropriately before connection. Thereafter, the entire circuit was powered and tested accordingly.

4.2 TESTING

The circuit was designed and soldered on a Vero board on which a temporary test was carried out. Components were tested individually and subsequently, modules of the circuit were tested.

Components like variable resistors (for adjusting the contrast of the LCD) were first calculated to be 2kΩ (in chapter three), which was later varied to 1.3kΩ. However, during testing, I found out that 1kΩ was more suitable because it provides a brighter output of the LCD. This is because the 1kΩ resistor offers less resistance to the flow of current when compared to the 1.3kΩ resistor.

Furthermore, after the whole soldering, wire links and connections, the circuit continuity was carefully tested. This test was carried out to confirm and check that the soldering was done

along the specified design, so that any deviation noticed is carefully amended. After this, the circuit was powered to see the outcome of the initial process.

At first, the operation of the circuit was not according to the specified design. The errors that occurred were carefully corrected. These errors were Circuit Bridge and unwanted soldering gaps. After the corrections, the circuit was working in accordance with the design.

Finally, the digital display (LCD) displayed the intended outputs.

4.3 DISCUSSIONS AND RESULTS

The table below shows the behavior of the controller when tested with different preset temperatures:

Table 4.1 Table showing behavior of temperature controller at preset temperatures.

Preset temperature	**Ambient temperature**	**Time taken to attain preset temperature**	**Action of temperature controller**
36°C	32°C	1 Minute	Supply was cut to heating element
60°C	31°C	3 Minutes	supply was cut to heating element
21°C	32°C	-	Temperature controller switched off automatically

From the table above, various preset temperatures were set and the time taken to attain such temperatures was observed. Once the preset temperature was attained, power supply was cut to the heating element-this was the expected result.

Also, when the preset temperature was set below the ambient temperature, the temperature controller switched off automatically.

4.4 CASING

This was constructed by using chip board, which was then painted. They were coupled together using adhesive gum. The chip board was shaped into a rectangular box and outlets were made on the tiles by drilling some parts to provide adequate ventilation for the components as well as provide an outlet for the LCD screen.

4.6 PROBLEMS ENCOUNTERED

The following problems and difficulties were encountered during the design and construction process:

1. The problem of short-circuit on the board due to bridging of wires.
2. The problem of soldering the LCD. This problem arised due to the bridging of the jumper wires used to link the display together.
3. Damage of certain components due to excessive heat while soldering.
4. Problem of memory overlap encountered during the coding which resulted in error in display.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCLUSION

This project is a prototype of a microcontroller-based temperature controller that can sense temperature, indicate it on an LCD display, and compare it with preset temperature ranges to switch off a heating element. The project is working according to the specified design and the digital display system used in this project has solved the parallax-error problems encountered in reading analogue thermometers.

While the system is suitable for both domestic and industrial applications, it may not be suitable for measurement at higher temperatures.

5.2 RECOMMENDATION

For a more sensitive and efficient system, the following recommendations are made:

1. An amplifier should be used to interface both the sensor and the ADC.
2. Constant power supply should be made available by anyone undertaking the project in future.
3. A rechargeable 9V battery can be introduced into the system in place of PHCN supply.
4. An uninterruptible power supply (UPS) can be used as an alternative power supply.

REFERENCES

[1] Applications of LM35 sensor (2005, 6 August-last update) [online]. Available: http://www.nationalsemiconductors/productspecification.htm [2009, September 12].

[2] History of temperature (2002, 21 August-last update [online]. Available: http://www.capgo.com/resources/intereststories/temphistory.html [2009, November 6]

[3] Adediran Y.A., (2000), Applied electricity 1st edition. Pp 140-160.

[4] How microcontrollers work (2007, 13 October-last update) [online]. Available: http://electronics.howstuffworks.com/microcontroller.htm [2009, May 4].

[5] Liquid crystal display (2008, 23 September-last update) [online]. Available: http://wikipedia.org/wiki/liquid_crystal_display.htm [2009, May 4].

[6] Digital converter (2000, 21 August-last update) [online]. Available: http://enwikipedia.org/wiki/analogue_to_digital_converter [2009, November 6]

[7] Ajaguna B.J. (2005), "Design and Construction of a digital temperature measuring instrument" (unpublished), B.Eng. department of Electrical/computer Engineering, F.U.T, Minna, Niger State, Nigeria.

[8] John C.M, (2001), Applied Electronics. First edition. Pp 90,101.

[9] Albert P.M, (1992), Malvind Electronics principle, Pp 34-42

[10] Kenneth J.A, (2002), The 8051 microcontroller: Architecture, Programming and Applications, Pp 241-243.

[11] Fafemi A.T, (2003) "Design and Construction of a computerized thermometer" (unpublished). B.Eng. Department of Electrical/Computer Engineering F.U.T, Minna, Niger State, Nigeria.

[12] " Who invented the Thermometer-Fahrenheit, Celsius and Kelvin scales?" (1999, 10 September-last update). [online]. Available: http://www.about.com [2009, July 25].

[13] The history of temperature and thermometry (2007, 23 September-last update) [online]. Available:http://www.zytemp.com/tutorial/history_of_temp_and_thermometry.htm [2009, 23 November].

[14] Paul H.B, and Winfield H.A, (1997), The Art of Electronics, Cambridge University Press. Pp 338, 467.

[15] Microcontroller-Based Regulators (2004, 4 February-last update). [online]. Available: http://www.wikipedia.org/wiki/microcontroller_based_regulators. [2009,November 23].

APPENDIX

Assembly Language And Codes

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

;-------------------------------------------------------------------------

;   DEFINATIONS

;-------------------------------------------------------------------------

;-------------------------------------------------------------------------


;Registers

ADCTimer            equ 00h

ActualTemp          equ 01h

PresetTemp          equ 02h

LCDData             equ 03h

PRNTCtr             equ 04h

BlinkTimerCtr       equ 05h

Ctr1                equ 06h

Ctr2                equ 07h


ActualTempH         equ 08h

ActualTempL         equ 09h


PresetTempH         equ 0Ah

PresetTempL         equ 08h
```

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

;Bit Memories

Modeflag                equ 20h

StatusFlag      equ 21h


ModeFlag0               equ 00h

ModeFlag1       equ 01h

ModeFlag2       equ 02h


BlinkFlag       equ 08h


;-------------------------------------------------------------------------

;Ports

LCDDataBus              equ P2

ADCDataBus      equ P1


;-------------------------------------------------------------------------

;Bits

Relay           equ P3.4
```

```
                                                    ;LCD Pin defination
RS                          equ P3.7            ;Register Select Pin
RW                          equ P3.6            ;Read Write Pin
EN                      equ P3.5            ;Enable Pin

BF                          equ P2.7            ;Busyflag

                                                    ;ADC Pin defination
ADCRead         equ P3.1
ADCWrite        equ P3.0

                                                    ;Key Button Pin defination
bMode           equ P3.2
ExtInt1         equ P3.3
bAdjustP            equ P0.0
bAdjustN            equ P0.1

;=========================================================================

;Constant
FuncSet             equ 56      ;16X2 lines, 5X7 dot matrix
D1C0B0              equ 12      ;Display:ON, Cursor:OFF, Blink:OFF
```

```
ClearDisplay    equ 1
EntryMode       equ 6
ReturnHome              equ 2


Timer0H         equ 165
Timer0L         equ 255

Timer1H         equ 0
Timer1L         equ 0
;-----------------------------------------------------------------
;-----------------------------------------------------------------
;   VECTOR ADDRESSES
;-----------------------------------------------------------------
;-----------------------------------------------------------------

;-----------------------------------------------------------------
        Org 0000h       ;RESET VECTOR ADDRESS
  ljmp Start    ;Jump to start of program
;-----------------------------------------------------------------
  Org 0003h     ;EXTERNAL INTERRUPT0 VECTOR ADDRESS

        acall IMode

        reti
```

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

  Org 000Bh       ;TIMER0 INTERRUPT VECTOR ADDRESS

        clr TR0           ;Stop Timer0

        acall BlinkTimer

        mov TH0,#Timer0H     ;Timer1 reload value= 3035 (bdb)
  mov TL0,#Timer0L
        setb TR0                    ;Start Timer0

        reti                        ;Not used
;-------------------------------------------------------------------------

  Org 0013h         ;EXTERNAL INTERRUPT1 VECTOR ADDRESS

        acall Adjust

        reti
;-------------------------------------------------------------------------

  Org 001Bh         ;TIMER1 INTERRUPT VECTOR ADDRESS

        acall nReadADC

        reti
;-------------------------------------------------------------------------
```

```
Org 0023h       ;SERIAL INTERRUPT VECTOR ADDRESS
reti            ;Not used
;-----------------------------------------------------------------------
Org 002Bh       ;TIMER2 INTERRUPT VECTOR ADDRESS
reti            ;Not used
;-----------------------------------------------------------------------
        Org 0038h                                   ;Program starts here
Start:
mov SP,#40h                                         ;Stack Pionter intialized

        clr RS0                                     ;Bank0 selected
clr RS1

        clr Relay                                           ;Off Relay
        mov ModeFlag,#0
        setb ModeFlag0                              ;Enable Mode0

        mov StatusFlag,#0
        setb BlinKFlag                              ;Disable Blinking of PresetTemp

        setb bMode
        setb Extint1
        setb bAdjustP
        setb bAdjustN
```

```
        acall Initialize_LCD                    ;Call Initialize LCD subroutine

        acall Setup_External_Interrupt          ;Call Setup external interrupt subroutine

        acall Setup_Timers                      ;Call Setup Timers subroutine

        acall InitializeADC                     ;Call Initialize ADC subroutine


        setb EA                                 ;Enable Global Interrupt

        ajmp $                                  ;Wiat here for
interrupts

;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
;   INTERRUPT CALLS
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
IMode:
        push acc
    mov A,PSW
        push acc
        mov A,DPH
```

```
        push acc
        mov A,DPL
        push acc

        jb Modeflag0,IModeFlag0                 ;This subroutine is used to
change the modes
        jb Modeflag1,IModeFlag1                 ;It loop from mode0 to mode2
        jb Modeflag2,IModeFlag2
        ajmp EndMD

IModeFlag0:
        mov ModeFlag,#0                         ;Clear modeflag
        setb ModeFlag1                          ;set to mode1
        ajmp EndMD                              ;jump to end of mode
interrup routine

IModeFlag1:
        mov ModeFlag,#0                         ;Clear modeflag
        setb ModeFlag2                          ;set to mode2

        setb Relay
        acall Update_LCD_With_Device_Status_ON
        acall Update_LCD_With_PresetTemp
        ajmp EndMD                              ;jump to end of mode
interrup routine

IModeFlag2:
```

```
        mov ModeFlag,#0                                         ;Clear modeflag
        setb ModeFlag0                                  ;set to mode0
        clr Relay
        acall Update_LCD_With_Device_Status_OFF

        ajmp EndMD                                              ;jump to end of mode
interrup routine


EndMD:

        pop acc
        mov DPL,A
        pop acc
        mov DPH,A
   pop acc
        mov PSW,A
        pop acc

        ret
;*********************************************************************
Adjust:
        push acc
    mov A,PSW
        push acc
        mov A,DPH
```

```
        push acc
        mov A,DPL
        push acc

        jb ModeFlag1,ProceedAD0                    ;End Intturupt if not in mode1.
mode1 is only used to adjust the PresetTemp
        ajmp EndAD
ProceedAD0:
        jnb bAdjustP,Increment_PresetTemp          ;If bAdjustP key was presesd jump to
IncPresetTemp
        jnb bAdjustN,Decrement_PresetTemp          ;If bAdjustN key was presesd junp to
DecPresetTemp
        ajmp EndAD                                 ;End the interrup if either of the
key is not pressed

Increment_PresetTemp:
  inc PresetTemp                                   ;Increament PresetTemp
  mov A,PresetTemp                                 ;Conpare it with 100
  cjne A,#100,NotEqualTo100                ;If not equal to 100, jump to NotEqualto100
EqualTo100:
        mov PresetTemp,#99                         ;Else reset Presettemp to 99
        ajmp Convert_PresetTemp_To_ASCII     ;jump to ConvertPresetTemp2ASCII
NotEqualTo100:
        jc      Convert_PresetTemp_To_ASCII        ;Test to know if Presettemp is greater than
100, jump if lessthan 100 to ConvertPresetTemp2ASCII
        mov PresetTemp,#99                         ;Else reset it to 99
        ajmp Convert_PresetTemp_To_ASCII
```

```
Decrement_PresetTemp:
        dec PresetTemp                                  ;Decrement PresetTemp
        mov A,PresetTemp
        cjne A,#255,NotEqualTo255                       ;Jump to NotEqualTo255 if Presettemp is not equal
to 255
        mov PresetTemp,#0                               ;Else reset PresetTemp to zero

NotEqualTo255:
   ajmp Convert_PresetTemp_To_ASCII     ;jump to ConvertPresetTemp2ASCII

Convert_PresetTemp_To_ASCII:                    ;Dividing PrestTemp by 10 will split the Presettemp
into two digits
        mov A,PresetTemp                                ;the Tens will be saved in
Acumulator, while the units will be in B register
        mov B,#10                                       ;A anb B are then
loaded into ActualtempH and ActualtempL respectiveiy
        div AB                                          ;The above process can
be considered as a binary to two digit BCD conversion
        mov PresetTempH,A
        mov PresetTempL,B

        mov A,#48                                       ;Adding 48 will convert
the BCD to its ASCII equivalent.
        add A,PresetTempH                               ;Converting from binary to
ASCII is necessary becuase
        mov PresetTempH,A                               ;the LCD does not understands
binary but ASCII.
```

```
        mov A,#48
        add A,PresetTempL
        mov PresetTempL,A

        acall Update_LCD_With_PresetTemp        ;call this subroutine to update the LCD with the
PresetTemp value

EndAD:
  acall Debounce                ;Delay to avoid the debounce of the key

        pop acc
        mov DPL,A
        pop acc
        mov DPH,A
  pop acc
        mov PSW,A
        pop acc

        ret
;-------------------------------------------------------------------------
nReadADC:
  push acc
  mov A,PSW
        push acc
        mov A,DPH
```

```
        push acc
        mov A,DPL
        push acc


djnz ADCTimer,EndRADC
mov ADCTimer,#2

IRead:

        clr ADCWrite
        acall Delay
        setb  ADCWrite

        acall Delay
        acall Delay
        acall Delay
        acall Delay
        acall Delay
        acall Delay

        clr ADCRead
        acall Delay
```

```
        mov ActualTemp,ADCDataBus

        setb ADCRead

        acall Delay

        acall Convert_ActualTemp_To_ASCII

        acall Update_LCD_With_ActualTemp

        acall CompareTemp

EndRADC:
        pop acc
        mov DPL,A
        pop acc
        mov DPH,A
    pop acc
        mov PSW,A
        pop acc
        ret

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

```
BlinkTimer:
        push acc
   mov A,PSW
        push acc
        mov A,DPH
        push acc
        mov A,DPL
        push acc

        djnz BlinkTimerCtr,EndBT
        mov BlinkTimerCtr,#10
        jnb ModeFlag1,EndBT

        jb BlinKFlag,Clear_BlinkFlag

 Set_BlinkFlag:
         setb BlinkFlag
         acall Update_LCD_With_PresetTemp
         ajmp EndBT

  Clear_BlinkFlag:
         clr BlinkFlag
         acall Clear_PresetTemp
         ajmp EndBT
```

```
EndBT:
        pop acc
        mov DPL,A
        pop acc
        mov DPH,A
    pop acc
        mov PSW,A
        pop acc

        ret

;-----------------------------------------------------------------
;-----------------------------------------------------------------
;   SUBROUTINE CALLS
;-----------------------------------------------------------------
;-----------------------------------------------------------------
Setup_External_Interrupt:

        setb Extint1                    ;Enable Pins as input
        setb bMode

        setb bAdjustP
        setb bAdjustN
```

```
        setb EX0                        ;External Interrupt0 enabled
  clr IT0                               ;Extrenal Interrupt0 on 1-0 transition

  setb EX1                              ;External Interrupt1 enabled
  clr IT1                               ;Extrenal Interrupt1 on 1-0 transition

        ret
;--------------------------------------------------------------------------------
Setup_Timers:

        mov ADCTimer,#2
        mov BlinkTimerCtr,#10

        mov TH0,#Timer0H                ;Timer0 reload value= 3035
  mov TL0,#Timer0H
        setb ET0                        ;Timer0 interrupt enabled
        setb TR0                        ;Start Timer0

        mov TH1,#Timer1H                ;Timer1 reload value= 3035
  mov TL1,#Timer1H
        setb ET1                        ;Timer1 interrupt enabled
        setb TR1                        ;Start Timer1

        ret
```

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

Initialize_LCD:
                                        ;The initial state of the LCD controll
        setb BF
pins
                                        ;as stated by the datasheet
        clr RS
        clr RW
        clr EN

                                        ;This register is used by the
        mov PRNTCtr,#16
PrintString subroutine
                                        ;for printing string to
the LCD. max value=16
        acall LCDBusy

        mov LCDdata,#FuncSet            ;Enable 2 Lines, 5X7 dot matrix,
        acall SendCommand
    mov LCDdata,#0D1C080                ;Display on, Cusor off, Blink off
        acall SendCommand
    mov LCDdata,#ClearDisplay           ;Clear display
        acall SendCommand
    mov LCDdata,#EntryMode
        acall SendCommand

        acall LDelay

                                        ;Display Loading
        mov dptr,#Loading
        acall Display
```

```
acall Wait0
mov LCDData,#203
acall SendCommand
acall Wait0
mov LCDData,#'.'
acall SendData
acall Wait0
mov LCDData,#'.'
acall SendData
acall Wait0
mov LCDData,#'.'
acall SendData
acall Wait0
mov LCDData,#'.'
acall SendData
acall Wait0
mov LCDData,#'.'
acall SendData
acall Wait

mov dptr,#Welcome0                    ;Display welcome0
acall Display
acall Wait
```

```
        mov dptr,#DisplayS                          ;Display Screen
        acall Display
        acall Wait
        ret
;------------------------------------------------------------------
SendCommand:

        mov LCDDataBus,LCDData                      ;This subroutine is used to send a
command to the LCD
        clr RS
        clr RW
        setb EN
        clr EN
        acall LCDBusy

        ret
;------------------------------------------------------------------
SendData:

    mov LCDDataBus,LCDData                          ;This subroutine is used to send a data to the LCD
        setb RS
        clr RW
        setb EN
        clr EN
        acall LCDBusy
```

```
        ret
;-------------------------------------------------------------------------
LCDBusy:

        setb BF                         ;This subroutine is used to test
the busy flag of the LCD
        setb EN                         ;If the busy flag is '1' , the LCD is
not ready
        clr RS                          ;If the busy flag is '0', the LCD is ready
        setb RW
CkeckBF:
        clr EN
        setb EN
        jb BF,CkeckBF
        ret
;-------------------------------------------------------------------------
Display:                                ;This subroutine is is to print character
to the entire LCD display
        mov LCDdata,#128                ;ie print on both Row1 and Row2
        acall SendCommand               ;Row1 start address is 128
  acall SendString                      ;row2 start address is 192

        mov LCDdata,#192
        acall SendCommand
  acall SendString
  ret
;-------------------------------------------------------------------------
```

```
SendString:
        clr A                                   ;This suroutine is used to print
characters to the LCD
        movc A,@A+dptr                          ;the numbers of character
printed by this subroutine
        mov LCDData,A                           ;is equivalent to the number found in
resgister PRNTCtrr
        acall SendData
        inc dptr
        djnz PRNTCtr,SendString
EndPRNT:
   mov PRNTCtr,#16
        ret
;-------------------------------------------------------------------------
InitializeADC:
        setb ADCRead                            ;Set Pins to high, the data sheet for the
ADC0804 require that
   setb ADCWrite                                ;the intial state of this pins to be high

        mov ADCTimer,#2                         ;use to delay the ADC read time
to be 1 second

        ret
;-------------------------------------------------------------------------
CompareTemp:
        jb Modeflag2,ProceedCT0                 ;This function will only be executer
when in mode1
```

```
        ajmp EndCT                                          ;It compares the value in the
PresetTemp with that in the

ProceedCT0:                                                 ;ActualTemp. If the value of the
ActualTemp is greater or

        mov A,ActualTemp                                    ;equal to that in the PresetTemp, the
relay will be put off.

        cjne A,PresetTemp,Not_Equal


Not_Equal:

        jnc Actualtemp_is_Greater_Than_or_Equal_To_PresetTemp

        ajmp EndCT

Actualtemp_is_Greater_Than_or_Equal_To_PresetTemp:

        clr Relay

        mov ModeFlag,#0

        setb Modeflag0

        acall Update_LCD_With_Device_Status_OFF


EndCT:

        ret



;=========================================================

Update_LCD_With_PresetTemp:


        mov LCDData,#198                                    ;This subroutine will print the Tens and Units
of the PresetTemp on

        acall SendCommand                                   ;Row2 position 10 and 11 on the LCD
```

```
        acall Wait0
        mov LCDData,PresetTemphH
        acall SendData
        mov LCDData,PresetTemphL
        acall SendData
        ret
;-------------------------------------------------------------------------
Clear_PresetTemp:

        mov LCDData,#198                ;This subroutine will print the Tens and Units
of the actualTemp on
        acall SendCommand               ;Row2 position 16 and 17 on the LCD
        acall Wait0
        mov LCDData,#' '
        acall SendData
        mov LCDData,#' '
        acall SendData
        ret
;-------------------------------------------------------------------------
Convert_ActualTemp_To_ASCII:            ;Dividing PresiTemp by 10 will split the Presettemp
into two digits
        mov A,ActualTemp                ;the Tens will be saved in
Acumulator, while the units will be in B register
        mov B,#10                       ;A anb B are then
loaded into ActualtempH and ActualtempL respectively
        div AB                          ;The above process can
be considered as a binary to two digit BCD conversion
```

```
        mov ActualTempH,A
        mov ActualTempL,B

        mov A,#48                                   ;Adding 48 will convert
the BCD to its ASCII equivalent.
        add A,ActualTempH                           ;Converting from binary to
ASCII is necessary because
        mov ActualTempH,A                           ;the LCD does not understands
binary but ASCII.

        mov A,#48
        add A,ActualTempL
        mov ActualTempL,A
        ret
;---------------------------------------------------------------------

Update_LCD_With_ActualTemp:

        mov LCDData,#204                    ;This subroutine will print the Tens and Units
of the actualTemp on
        acall SendCommand                   ;Row2 position 16 and 17 on the LCD
        acall Wait0
        mov LCDData,ActualTempH
        acall SendData
        mov LCDData,ActualTempL
        acall SendData
        ret
```

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

Update_LCD_With_Device_Status_ON:


   mov LCDData,#193                    ;This subroutine will print 'ON' on Row2  position 3
        acall SendCommand
        mov dptr,#DisplayON
        mov PRNTCtr,#3
        acall SendString


        ret
;********** ********** ********** ********** ********** ********** ********** **

Update_LCD_With_Device_Status_OFF:

                                                                    ;This subroutine will print 'OFF'
on Row2  position 2
        mov LCDData,#193
        acall SendCommand
        mov dptr,#DisplayOFF
        mov PRNTCtr,#3
        acall SendString


        ret
;*** ********** ********** ********** ********** ********** ********** ********** **

 Debounce:
        acall wait                                                  ;This is a delay
subroutine for key bounce
```

```
        ret
;---------------------------------------------------------------

Delay:                                          ;This is a delay subroutine
        mov Ctr2,#255
DO:
        mov Ctr1,#255
        djnz Ctr1,$
        djnz Ctr2,DO
        ret

LDelay:
        acall Delay
        acall Delay
        acall Delay
        ret


Wait:
  acall LDelay
        acall LDelay
        acall LDelay
        acall LDelay
        acall LDelay
Wait0:
```

```
        acall LDelay
        acall LDelay
        acall LDelay
        ret

;=================================================================
;=================================================================
;       LOOK UP TABLE
;=================================================================
;=================================================================
Loading:
    db '   Loading   '
        db 'please wait  '

Welcome0:
        db ' Digital     '
    db '  Thermometer '

displayS:
        db 'OSts PTemp ATemp'
db ' OFF','  ','00',223,'C','  ','00',223,'C'

layON:
```

```
        db 'ON'

DisplayOff:
        db 'OFF'

;-----------------------------------------------------------------------

;-----------------------------------------------------------------------

        end                                     ;This instruction is only needed by the compiler
```