DESIGN AND FABRICATION OF AN ELECTRONIC SYSTEM FOR MONITORING AND CONTROLLING TEMPERATURE, LIGHT ILLUMINANCE AND HUMIDITY IN A GREENHOUSE

WYCLIFFE OBANDA NYANYA
I56/0243/2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (ELECTRONICS AND INSTRUMENTATION) IN THE SCHOOL OF PURE AND APPLIED SCIENCES OF KENYATTA UNIVERSITY

APRIL 2010

DECLARATION

I declare that the work presented in this thesis is my original work and has not been presented for a degree in any other university or for any other award.

WYCLIFFE OBANDA NYANYA Signature…… Date……

PHYSICS DEPARTMENT

KENYATTA UNIVERSITY

We confirm that the candidate carried out the work reported in this thesis under our supervision.

DR. PATRICK M. KARIMI Signature……..... Date…….....

PHYSICS DEPARTMENT

KENYATTA UNIVERSITY

P. O. BOX 43844-00100 GPO

NAIROBI-KENYA

DR. ABDALLAH S. MERENGA Signature……**....** Date ……

PHYSICS DEPARTMENT

KENYATTA UNIVERSITY

P. O. BOX 43844-00100 GPO

NAIROBI- KENYA

DEDICATION

This work is dedicated to my wife, Elizabeth, my daughters Sharon and Louise, my son Mark and to my late father Herbert Ambundo who would have loved to see it all.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank all my lecturers in the physics department for instilling confidence in me when pursuing this course. Special thanks go to my supervisors, Dr. P. M. Karimi and Dr. A. S. Merenga, for their guidance throughout this research. I also wish to thank my colleague John Githaiga for his useful suggestions. The technicians in the Physics department were very helpful to me during the construction of the system. I also wish to thank my wife Elizabeth Obanda for her moral support and patience. My daughters Sharon and Louise and my son Mark were very patient with me during long absences. Lastly but not least, I wish to thank my employer, the Teachers' Service Commission for granting me a 6 month study leave which enabled me complete the system construction and implementation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AC	- Alternating Current
ADC	- Analogue-Digital Converter
ALU	- Arithmetic Logic Unit
BCD	- Binary Coded Decimal
CISC	- Complex Instruction Set Computer
CMOS	- Complementary Metal Oxide Semiconductor
CPU	- Central Process Unit
DC	- Direct Current
DIL	- Dual-In-Line
DMA	- Direct Memory Access
DOS	- Disk Operating System
DRAM	- Dynamic Random Access Memory
EEPROM	- Electrical Erasable Programmable Read Only Memory
EPROM	- Erasable Programmable Read Only Memory
EU	- European Union
I/O	- Input/Output
IC	- Integrated Circuit
IPR	- Intellectual Property Rights
IR	- Instruction Register
ISR	- Interrupt Service Routine
LDR	- Light Dependant Resistor
LED	- Light Emitting Diode

LVDT - Linear Variable Displacement Transducer

MIPS - Million Instructions Per Second

OTP - One Time Programming

PAR - Photosynthetically Active Radiation

PC - Personal Computer

PC - Program Counter

PCB - Printed Circuit Board

PIA - Peripheral Interface Adapter

PMP - Post-Metal Programming

PWM - Pulse Width Modulator

RAM - Random Access Memory

RISC - Reduced Instruction Set Computer

RLS - Reflective Light Sensors

ROM - Read Only Memory

RTD - Resistance Dependant Resistor

RVDT - Resistance Displacement Transducer

SRAM - Static Random Access Memory

TRIAC - TRIode for Alternating Current

TTL -Transistor Transistor Logic

UART - Universal Asynchronous Receiver-Transmitter

ABSTRACT

Climate uncertainties pose a challenge to the production of food to feed the growing billions on the globe. Climatic conditions can change abruptly ruining farming efforts and leading to food shortages. Greenhouses attempt to solve this problem by enclosing crops in a climatically controlled environment. Crop production in greenhouse environment has the advantage of obtaining reliable and continuous output throughout the year. Each greenhouse has distinct characteristics. Data on these characteristics need to be collected at regular intervals. For each type of crop, these characteristics need to be controlled within the specified limits to achieve the maximum efficiency and yields. In the past, greenhouses employed electromechanical devices such as thermostats to monitor and control the environment. Mechanical systems lack the flexibility and precision required for greenhouse control. Some modern greenhouses use computers to control the environment. Computers require programming skills, are bulky and costly. A greenhouse monitoring and control system that is compact and thus portable, adaptable, cheap and easy to assemble has been developed. Based on the Z80 microprocessor, the hardware consists of a Z80 CPU, two 8255 PIA's, a 2716 EPROM, a 6116 RAM, an HIH-4000 humidity sensor, an LM-35 temperature sensor and two ADC 0804 analogue to digital converters, a clock and reset circuitry and 7-segment displays and LEDs. The software has been coded in the Z80 assembly language. This system monitors temperature, humidity and light illuminance on a continuous basis. It measures and displays these parameters and activates appropriate devices whenever these variables fall outside predetermined ranges. Relative humidity is controlled by a water sprinkler, temperature by a fan and heater and light illuminance by a lighting system.

Chapter 1

INTRODUCTION

1.1 Background to the Study

A large section of Kenya is arid or semi arid. In recent years, the country has experienced severe droughts and rainy seasons that have been accompanied by floods which have swept away crops and displaced people. The climate uncertainties have led to severe food shortage and starvation. One possible solution to achieving food security is to use greenhouses. In greenhouses, the conditions are controlled to ensure maximum crop yield irrespective of the weather. In Kenya, greenhouses are being used in Naivasha to grow flowers, most of which are exported, earning the country revenue. Greenhouses could be automated by employing microprocessor systems that would monitor light illuminance, relative humidity and temperature and keep these at optimum levels on a continuous 24-hour basis.

In automated greenhouses it is possible to manipulate crop production for certain economic advantages. For instance, a grower may lower temperature to delay flowering so that the flowers are ready for picking say on a Monday rather than having them picked over a weekend when he might pay more to workers and there may be no flights available (Plaksina and Raush, 2005). Facilities for plant growth research where a high degree of climatic control is required could also make use of automated greenhouses. Thus the need for automating control of the greenhouse parameters has varied applications.

This chapter will first consider the statement of the research problem where greenhouse control systems are compared, and then the objectives of this work will be stated. The chapter will conclude with a discussion of the rationale for this research work.

1.2 Statement of the Research Problem

It is known that crops do well if the right conditions are met throughout their lifetime. With the weather conditions being increasingly unpredictable in Kenya, the challenge of producing crops like vegetables and flowers that are needed all year round, and that cannot be stored for long, can only be met by using greenhouses. There are numerous greenhouse environmental control systems present on the market. They have certain disadvantages: most of them are costly and lack flexibility. They also have a dependence on the manufacturer, which means less safety for the investment (Plaksina and Raush, 2005).

Some greenhouse control systems use PCs (Personal Computers) to control the greenhouse environment. Marhaento and Singh (2002) used a PC in their greenhouse controller while Dogra and Chandra (2005) used a PC with the Intel 386 microprocessor in their greenhouse control system. It may be easier to just program a computer to control a greenhouse but there are certain disadvantages. First, PCs are costly since one has to pay for everything that comes with the PC architecture such as the keyboard, disk drives and so on, even if the application does not need them (Ball, 2002). Also, with a PC, the intellectual rights of the system designer cannot be guaranteed. With a microprocessor, the hardware can be identified but the software that runs it is difficult to decipher (Heath, 2003).

In this work the Z80 microprocessor has been employed. It has the advantage of being readily available and is relatively cheap besides having an instruction set that is easy to understand. Also, the Z80 microprocessor has a minute power consumption and uses a single power source of +5 V, which greatly simplifies the circuitry, (Rongen, 2005). The objectives of this work are listed below.

1.3 Objectives

The general objective of this research work was to design and fabricate a Z80 microprocessor based system to monitor and control temperature, relative humidity and light illuminance in a greenhouse. The specific objectives were:

i. To design and fabricate a microprocessor based system using the Z80 CPU,

ii. To develop software for monitoring and controlling the system such that:

   a. The system measures temperature, displays it, and controls a heater or fan to keep the temperature within the desired range.
   b. The system measures relative humidity, displays it, and controls a water sprinkler to keep the humidity above the minimum desired range.
   c. The system measures light level and controls a lighting system to keep the light level at the desired value.

1.4 Rationale

Although Kenya's economy is agricultural based, farming in rural Kenya is mainly subsistence. Farmers toil on their farms, but due to the climatic uncertainties, the harvests are dismal most of the time. This leads to food insecurity with the accompanying hunger. Farming efforts need to be

integrated with environmental control systems. This will reward the farmers and ensure food security for the country.

Greenhouse control systems based on microprocessor technology mainly use PCs. These are expensive and require knowledge of computers and programming. This project has used a Z80 microprocessor to control a greenhouse environment. The control system can be fabricated easily and cheaply. This system can also be used in research facilities for plant growth where precision climatic control is required. Such facilities can in turn develop better seeds suited for particular climatic regions, thus improving food security.

There are several greenhouse control systems in use. The next chapter will discuss some of these systems and show how the various environmental parameters in the greenhouses are controlled.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

This chapter will first discuss the greenhouse effect which forms the basis of greenhouses. Then the chapter gives a brief history of greenhouses and the trend in greenhouse technology development. The benefits of greenhouse cultivation are enumerated and some automated greenhouse control systems discussed. The management of light illuminance, temperature and relative humidity is discussed. The chapter concludes with a discussion on the challenges of optimizing the environmental parameters in greenhouses using computerized control systems.

2.2 Greenhouse Effect

The greenhouse effect is the difference between the amount of infrared energy emitted by the earth's surface and that measured at the top of the atmosphere. It results when some atmospheric trace gases, permit incoming solar radiation to reach the surface of the earth unhindered while restricting the outward flow of infrared radiation. These atmospheric trace gases are referred to as greenhouse gases. They absorb and reradiate this outgoing radiation, effectively storing some of the heat in the atmosphere, thus producing a net warming of the surface. This process is called the greenhouse effect and is the basis of greenhouses.

The greenhouse effect plays a crucial role in maintaining a life-sustaining environment on the earth. If there was no greenhouse effect, the temperature of the earth would be determined by the amount of incoming solar radiation that reaches and heats its surface. This temperature would not be able to sustain life as will be demonstrated next.

The amount of incoming solar radiation, W, received at the earth's surface is given by

$$W = \pi R^2 S(1-A) \tag{3.1}$$

where,

R = 6.63 x 10^2 m, is the radius of the earth,

S = 1372 W/m^2, is the solar constant; and

A = 0.33, is the albedo of the earth's surface

This amount of incoming solar radiation reaches the surface of the earth and heats it to a temperature, called the effective temperature, T_e. Supposing that the earth emits heat like a blackbody, each square meter of the earth's surface radiates infrared radiation according to Stefan-Boltzmann law, which states that the emission of infrared radiation, E, is given by

$$E = \sigma T_e^{\,4} \tag{3.2}$$

where

σ = 5.76 x 10^{-8} W/m^2 K^4, is the Stefan-Boltzmann constant.

Hence, the total amount of infrared radiation, X, emitted by the earth's surface is given by

$$X = 4\pi R^2 \sigma T_e^{\,4} \tag{3.3}$$

Since there is a balance between the incoming solar radiation reaching the surface and the outgoing infrared radiation emitted at the surface, these two terms in equation 3.1 and 3.3 are equated and solved for the effective temperature, T_e. Thus

$$T_e = \left[(S(1-A)/4\sigma)\right]^{1/4}, \tag{3.4}$$

which when solved yields T_e= 253 K as the effective temperature for the earth. At this temperature (-20 ^oC), the earth would be very cold. However, actual measurements indicate that the mean temperature of the earth averaged over the year and over all latitudes is about 288 K (15 ^oC), rather

than 253 K. This difference is due to the greenhouse effect. This effect is the basis of greenhouses for crop production. The next section considers the development of greenhouses and some control systems in use.

2.3 Greenhouses

Greenhouse technology is more than 200 years old and was pioneered by the Europeans. In the entire world the area under greenhouse cultivation as reported by for the year 1999-2000 was about 275,000 hectares. In this same period, India, for instance, was reported to have a total area under greenhouse cultivation of 110 hectares. Vegetables such as brinjal, capsicum, tomato, etc were being grown in India throughout the year in hostile climates, whereas the green leafy vegetables are being grown in the long frozen winter months when the average temperature reaches -30.2 ^{o}C (Nabard, 2006).

The benefits which can be derived from greenhouse cultivation are as follows (Hanan, 1998):

i. Environment control allows raising plants anywhere in the world at any time of the year, i.e., crops could be grown under the inclement climatic conditions when it would not be otherwise possible to grow crops under the open field conditions.
ii. The crop yields are at the maximum level per unit area, per unit volume and per unit input basis.
iii. The control of the microcosm allows the production of higher quality products which are free from insect attack, pathogens and chemical residue.
iv. High value and high quality crops could be grown for export markets.
v. Income from the small and the marginal land holdings maintained by the farmer can be increased by producing crops meant for the export markets.

vi. It can be used to generate self employment for the educated rural youth in the farm sector.

Greenhouse control systems have become more automated in recent times. Marhaento and Singh (2002) developed a programmable environment controller for the greenhouse environment using a PC. The controller consisted of ten temperature sensors and two probes (water sensors). The relative humidity was measured using a wet bulb temperature sensor. The signal processing consisted of a multiplexer, an amplifier, ADC (Analogue Digital Converter) and interface card. The software was compiled under DOS. They reported that the lower limits had to be set slightly higher and the higher limits slightly lower to prevent overshoot due to delay in response time of the controller system and the sensors.

Leung et al. (2005) constructed an experimental greenhouse from Melinex panels which are air inflatable. Unlike conventional glasshouses which are subjected to infiltration and constant air changes, the Melinex greenhouse creates a sealed unit which accumulates humidity through plant transpiration. Excessive water vapour, whilst being detrimental to the healthy growth and quality of the crop, is a valuable source of heat energy. Dehumidification of the greenhouse is achieved by a heat pump which also provides the basic heating load. Excess latent heat is stored in a water bath for use in adverse weather conditions. The system components are linked through a microprocessor based controller.

Dogra and Chandra (2005) employed sensors designed using silicon diodes connected across a bridge measuring mho changes per change in resistance of silicon per unit change in input signal. For the controller they used an Intel 386 board with AD/DA 16 channels. All output channels were connected to triggering devices through opto-couplers.

A basic requirement for any greenhouse is the enclosure or structure. The next section discusses different structures in use and how these relate to light management, temperature management and relative humidity management.

2.4 Greenhouse Structures and Light Management

A greenhouse requires an enclosure, a structure, to protect the plants from adverse environmental parameters. The structure represents both the barrier to direct contact to the external environment and the containment of the internal environment to be controlled. The covering material by design should allow for maximum light penetration for the growing crops.

A typical greenhouse covering intercepts a percentage of light falling on it allowing a maximum of 80% of the light to reach the crop at around noon, with an overall average of 68% over the day (Wilson et al., 1992). The covering also partially diffuses or scatters the light coming into the greenhouse so that it is not all moving in one direction. This is advantageous as the scattered light tends to reach more leaves in the canopy than directional light which throws more shadows.

Greenhouse covering materials such as glass panels, polycarbonate panels, and polyethylene skins are in use. The determining factors of the material to be used are usually cost and length of service. Glass is more expensive, but will generally have a longer service life than either polycarbonate or polyethylene (Khosla, 1999). The ability of the covering to allow light into the greenhouse and yet reduce the heat loss from the greenhouse to the environment during the winter is also a factor. New coverings are being developed which selectively exclude certain wavelengths of light and as a result can help in reducing insect and disease problems.

Light is the most important variable affecting productivity in greenhouses (Wilson et al., 1992; Papadopoulos and Pararajasingham, 1997). Light intensity needs to be optimized. When light intensities are too high, screening or shading is employed (Stanghellini and Meurs, 1992). Plants respond to light in the Photosynthetically Active Radiation (PAR) and use this light to drive photosynthesis. PAR is defined as radiation in the 400 to 700 nm waveband.

Plants make better use of some wavelengths in PAR waveband than others. Figure 2.1 represents the photosynthetic action spectrum of plants showing the variation of relative rate of photosynthesis of plants over the range of PAR. All plants show a peak of light use in the red region, approximately 650 nm and a smaller peak in the blue region at 450 nm (Salisbury and Ross, 1978). Plants are relatively inefficient at using light and are only able to use about a maximum of 22% of the light absorbed in the 400 to 700 nm region (Salisbury and Ross, 1978).

A greenhouse covering that filters light at the two peaks should never be used. When light levels are limiting, supplementary artificial lighting will increase plant growth and yield. The amount of supplemental light required ranges between 120 - 180 W/m^2. This can be supplied by 400 W bulbs (Demers et al., 1991). Care should be exercised so that plants are not subjected to this supplemental light longer than is necessary. Apart from the harm to the plants, it would also be costly.

The greenhouse structure used in this work was designed by making metal loops of about 34 cm from the base and mounting these on a flat wooden surface of size 80 cm by 42 cm. The top was covered by a transparent polythene sheet of transmittance 45%. The front and the back of the enclosure had vents near the top and bottom to allow air to circulate freely. The fan was positioned

to form part of one of these vents. The sensors were mounted on IC sockets that were soldered onto small portions of PCB boards. The temperature sensor was placed in the middle of the greenhouse. The humidity sensor was placed at the middle and the light sensor placed outside the enclosure. The sprinkler was placed inside the greenhouse and attached to a water source. The pipe from the sprinkler was placed inside the enclosure at the top taking care that the water sprinkled would not fall onto the sensors.

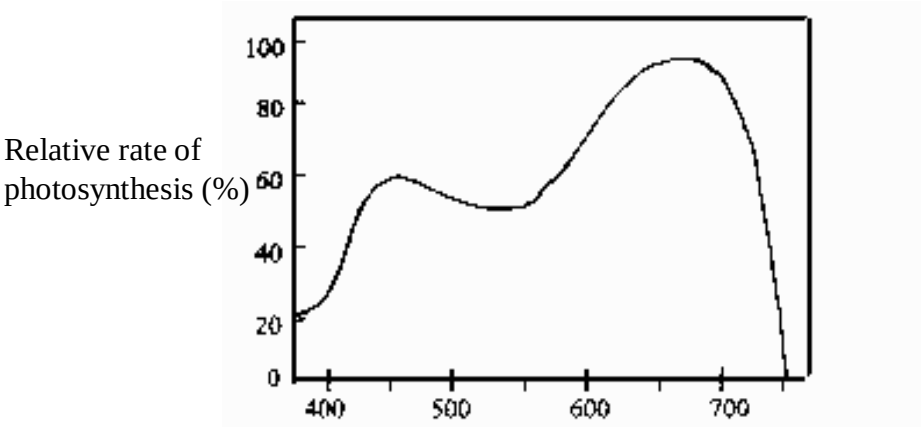


Figure 2.1: Photosynthetic action spectrum of plants showing relative rate of photosynthesis against wavelength in the 400-700 nm range. All plants show a peak at 650 nm (red region) and a smaller peak at 450 nm (blue region). Only a maximum of 22% of light absorbed in this range is actually used by the plants (Salisbury and Ross, 1978).

Light in a greenhouse causes temperature to rise and this in turn affects the relative humidity. The next sections will discuss the management of temperature and the management of humidity in greenhouses.

2.5 Temperature Management

Control of temperature is an important tool for the control of crop growth (Koning, 1996). For the control of crop growth, average temperature over one or several days is more important than the day/night temperature differences (Bakker, 1989; Koning, 1996). Various greenhouse crops show a high correlation between growth, yield and the 24-hour mean temperature (Bakker, 1989; Portree, 1996).

Temperature can be manipulated to direct the plant to be more generative, or more vegetative in growth. Optimum photosynthesis occurs between 21 ^{o}C and 22 ^{o}C (Portree, 1996). This temperature serves as the target for managing temperatures during daytime when photosynthesis occurs. The role of temperature in the optimization of plant performance and yield is based on the temperature of the plants rather than the temperature of the air. Plant temperatures are usually within a degree of air temperature, however during the high light periods of the year, plant tissues exposed to high light can reach 10 to 12 ^{o}C higher than air temperatures. Strategies to counter this include shading and evaporative cooling to reduce overheating of the plant tissues. Infrared thermometers are useful for determining actual leaf temperature.

Precision heating of specific areas within the crop canopy add another dimension of air temperature control beyond maintaining optimum temperatures of the entire greenhouse air mass. Using heating pipes that can be raised and lowered, heat can be applied close to flowers and developing fruit to provide optimum temperatures for maximum development in spite of the day-night temperature fluctuations required to signal the plant to produce more flowers. The rate of fruit development can be enhanced with little effect on overall plant development and flower set (Koning, 1996). Precise

application of heat in this manner can avoid the problem of low temperatures to the flowers and fruit which are known to disturb flowering and fruit set (Bakker, 1989).

Problems with low night temperatures can be sporadic in the greenhouse during the cold winter months and can occur even if the environmental control system is apparently meeting and maintaining the set optimum temperature targets. The primary reasons are lags in response time between the system's detection of the heating setpoints temperature and when the operation of the system is able to provide the required heat throughout the greenhouse, and also, specific temperature variations in the greenhouse due to drafts. In our work we have set the control system to begin intervention before the set target has been reached. This prevents overshoots. The point at which the system should begin the intervention is determined by trial and error.

Target temperatures for the root zone are 18 to 21 ^{o}C. Control can be maintained by monitoring the temperature at the roots and maintaining the pipes at a temperature that ensures optimum root zone temperatures. The use of tempered irrigation water (20 ^{o}C is optimum) can be used and this minimizes the shock to the root system associated with the delivery of cold irrigation water. Root injury can begin to occur at temperatures in excess of 23 ^{o}C in direct contact with the roots.

Heat can be applied to the air to influence the plant canopy or to the floor to influence the root system**.** The ventilation system provides the means by which the greenhouse air is circulated, mixed and exchanged. It allows for a more uniform climate and helps to distribute heat from the heating system as well as removing heat from the greenhouse when cooling is required.

Ventilation systems can use exhaust fans or, convection currents of air to exit the greenhouse through ridge or gutter vents. Additional air circulation within the greenhouse can provide for more uniform distribution of carbon dioxide, humidity and temperature especially during the winter (Brugger et al., 1987). The fans should be adequately sized to ensure that proper mixing of the air occurs without the fans being over-sized which can cause excessive air movement reducing yield (Brugger et al., 1987). The general recommendation for sizing is a fan capacity of 0.9 to 1.1 cubic meters per minute per square meter of floor area with a velocity no greater that 1 meter per second across the plants (Brugger et al., 1987).

During periods of high light intensity, air temperatures rise inside the greenhouse and cooling is required. Increasing ventilation rates serves to bring cooler, outside air into the greenhouse, but during typical summer months, ventilation alone is often not enough to maintain optimum greenhouse air temperatures. Cooling systems must be used to ensure optimum growing temperatures are maintained. These cooling systems also serve to humidify the greenhouse. Evaporative cooling is most effective in areas where the outside relative humidity is less than 60% (Brugger et al., 1987). The next section discusses the management of relative humidity.

2.6 Relative Humidity Management

Relative humidity is defined as the amount of water vapour in the air compared to the maximum amount of water vapour the air is able to hold at that temperature (Portree, 1996). The use of relative humidity for control of the water content of the greenhouse air mass has commonly been approached by maintaining the relative humidity below threshold values, one for the day and one for the night (Stanghellini and Van Meurs, 1992). Humidity levels high enough to favour disease organisms must

be avoided (Stanghellini and Van Meurs, 1992). The sole use of relative humidity as the basis of controlling greenhouse air water content does not allow for optimization of the growing environment, as it does not provide a firm basis for dealing with plant processes such as transpiration in a direct manner (Hanan, 1998). The common purpose of humidity control is to sustain a minimal rate of transpiration (Stanghellini and Meurs, 1992).

Plants exchange energy with the environment primarily through transpiration (Papadakis et al., 1994). This plant process is almost exclusively responsible for the subtropical climate in the greenhouse (Papadakis et al., 1994). Seventy percent of the light energy falling on a greenhouse crop goes towards transpiration (Hanan, 1998), and most of the irrigation water applied to the crop is lost through transpiration (Papadakis et al., 1994). Transpiration rate can determine the maximum efficiency at which photosynthesis occurs, how efficiently nutrients are brought into the plant and combined with the products of photosynthesis, and how these resources for growth are distributed throughout the plant. The transpiration rate of a given greenhouse crop is a function of temperature, humidity and light. For a given natural light level, the crops transpiration is primarily determined by the temperature and humidity in the greenhouse (Stanghellini and Van Meurs, 1992). For each level of natural light received into the greenhouse, a transpiration setpoint should allow for the determination of optimal temperature and humidity setpoints (Stanghellini and Van Meurs, 1992).

The plants themselves exert tremendous influence on the greenhouse climate (Lange and Tantau, 1996). Transpiration not only serves to add moisture to the environment, but is also the mechanism by which plants cool themselves and add heat to the environment (Papadakis et al., 1994). Optimization of transpiration rates through management of air temperature and relative humidity

can change over the course of the season. Early in the season, when plants are young and the outside temperatures are cold, both heat and humidity can be applied to maintain temperature and humidity targets. As the season progresses and the crops mature, increasing light intensity increases the transpiration rate and the moisture content of the air. To maintain optimum rates of transpiration, venting can be employed to reduce the relative humidity in the air. When some form of additional cooling is required, mist systems or pad and fan evaporative cooling can be used to both reduce the amount of ventilation for cooling as well as to add moisture to the air.

2.7 Computerized Environmental Control Systems

Computerized environmental control systems allow growers the ability to integrate the control of all the systems involved in manipulating the greenhouse environment. The effect is to turn the entire greenhouse and its component systems into a single instrument for control, where optimum environmental parameters are defined, and control is the result of the on-going input of the component systems acting in concert. Optimization of the environment for maximum crop production requires timely responses to changes in the environment and the changing requirements of the crop.

The greenhouse environment changes as the crop responds to its environment and the environment changes in response to the activity of the crop. Fast crop processes, such as photosynthesis are considered to respond instantaneously to the changing environment (Seginer, 1996). Due to the dynamics of the greenhouse, the inertia of the environment, it takes upwards of 15 minutes to implement changes to the environment (Seginer, 1996). Much of the disturbance to the greenhouse environment is due to the outside environment, i.e., the normal cycle of the day-night periods,

outside temperatures and the effects of scattered clouds on an otherwise sunny day (Seginer, 1996). The environmental control system has to continually work to modify the environment to optimize crop performance in response to on-going change of the dynamic environment.

The ability of a computer system to control the environment is only as good as the information it receives from the environment. The computer's contact with the environment occurs through various sensors recording temperature, relative humidity and light levels. It is important that quality sensors are used and routinely maintained to ensure that they are operating properly. Sensor placement is also important to ensure accurate readings of the crop environment, for example, a temperature sensor placed in direct sunlight is going to give a different set of readings than a temperature sensor placed within the crop canopy.

The control system should be able to optimize the variables for each part of the crop: the shoots, the leaves, the flowers, the fruits and the roots. To achieve this, the sensors should measure air, leaf and root temperatures, humidity, wind drafts and light levels at varies points and have these optimized individually. The optimization will vary depending on the time of day/night and the stage of growth of the crop. Care should be taken to prevent overshot due to time lag by using loops in the control algorithm to continuously monitor the variables and based on anticipation and history, respond appropriately. In this work, rather than run the sprinkler until the humidity sensor show that the desired levels have been attained, the sprinkler is activated for 2.5 s followed by a delay loop of 1 minute. This enables the system to have time to respond to the changes introduced. The photo period can also be varied appropriately and rather than switch on lights or off suddenly, these can be faded in or out to avoid sudden changes which might stress the crops.

A greenhouse control system needs to be able to optimize environmental conditions to ensure sustained crop production. To be able to design such a system, knowledge of various hardware devices to be used and their limitations is necessary. Programming skills are also required to deal with software issues. Chapter 3 will discuss the theory of the hardware of the control system, the peripheral devices, the sensors and the software.

Chapter 3

MICROPROCESSOR OPERATION AND PERIPHERAL DEVICES

3.1 Introduction

Greenhouses were initially primarily designed to conserve heat energy during cold seasons but have developed to include other parameters affecting crop production. The complexity of the greenhouse environmental control demands a control system that responds to all the variables of crop production in greenhouses. The environmental variables in greenhouses are interrelated, and thus affect each other. Microprocessor controlled systems can meet such a challenge. This chapter dwells on microprocessors in detail with emphasis on the Z80 microprocessor, which forms the basis of this work. Other hardware accessories used in this design are also discussed. Programming languages are mentioned and the Z80 assembler language highlighted. The next sections will discuss microprocessors with emphasis on the Z80 microprocessor which has been used in this research work.

3.2 Microprocessors

A microprocessor is a digital device that receives data or information, processes it in accordance with a stored software program and outputs signals in digital form. It accesses memory units, fetches and executes instructions, performs arithmetic and logical calculations, monitors external events, keeps track of where data information has to go and controls the timing and sequence of operations in the system (Anderson, 1984; Sivakumar, 1987; Bevolt, 1995). The main constituents of a microprocessor are:

i. The central processing unit (CPU), to process the data according to a program.

ii. Primary memory that include RAM and ROM.

iii. The input and output interfaces to handle communication between the computer and the outside world.

A microprocessor is not designed to be programmed by the end user. A user can make choices concerning functionality but cannot change the functionality of the system by adding or replacing the software. Thus microprocessors are sometimes referred to as embedded systems (Heath, 2003). The microprocessor was originally developed to replace a mass of logic that was used to create the first electronic calculators in the early 1970s. The calculators were made from discrete logic chips and many hundreds of these chips were needed just to create a simple four function calculator. With IC technology the individual logic functions were integrated to create higher level functions. Soon complete calculators were integrated on one chip. Instead of developing a new chip every time changes or improvements were required, a chip with programmable capability was built. This chip came to be known as the microprocessor.

The need to add or remove functionality is possible in microprocessors and is done by changing the software while keeping the hardware the same. This reduces the cost of production since many different systems can use the same hardware base. The same mechanism that allows new functionality to be added is also beneficial in allowing bugs to be solved through changing software (Wills, 2002). Protection of IPR (Intellectual Property Rights) is also possible with embedded systems. The hardware can be identified but the software that supplies the systems functionality can be hidden and is more difficult to analyze (Hersh, 2004). This is an advantage of a microprocessor over a PC system.

3.2.1 Microprocessor Fabrication Techniques

One of the common fabrication techniques of microprocessors is the CMOS (Complementary Metal Oxide Semiconductor) process. CMOS requires much less power than older fabrication techniques since it permits battery operation. CMOS chips also can be fully or near fully static, which means that the clock can be slowed (or even stopped) putting the chip in sleep mode. CMOS has a much higher immunity to noise (power fluctuations or spikes) than the older fabrication techniques. PMP (Post Metal Programming) is another fabrication technique from National Semiconductor that involves a high-energy ion implantation process that allows microcontroller ROM to be programmed after final metallization.

3.2.2 Microprocessors Types

Microprocessors fall into several architectural forms. Microcontrollers based on the Von-Neumann architecture have a single data bus that is used to fetch both instructions and data. Program instructions and data are stored in a common main memory. When such a controller addresses main memory, it first fetches an instruction, and then it fetches the data to support the instruction. The two separate fetches slow down the controller's operation.

Microcontrollers based on the Harvard architecture have separate data bus and an instruction bus. This allows execution to occur in parallel. As an instruction is being pre-fetched, the current instruction is executing on the data bus. Once the current instruction is complete, the next instruction is ready to go. This pre-fetch theoretically allows for much faster execution than the Von-Neumann architecture.

Almost all of today's microcontrollers are based on the CISC (Complex Instruction Set Computer) concept. The typical CISC microcontroller has between 100 and 250 instructions many of them very powerful and very specialized for specific control tasks (Bolton, 2000). It is quite common for the instructions to all behave quite differently. Some might only operate on certain address spaces or registers, and others might only recognize certain addressing modes. CISC architecture instructions are macro-like and allow programmers to use one instruction in place of many simpler instructions. Some of the common microprocessors are now discussed. A brief history of their development is given and some features highlighted.

3.2.2.1 Intel 8048

The Intel 8048 microprocessor has a modified Harvard architecture with programmable ROM on-chip with an additional 64 to 256 bytes of RAM also on-chip. I/O is mapped in its own space. Intel's second generation of microcontrollers, the 8051, has a separate address space for program memory and data memory. The program memory can be up to 64K, and can address up to 64K of external data memory. The memory can only be accessed by indirect addressing. In this microprocessor, instructions can single out bits just about anywhere (RAM, accumulators, I/O registers, etc.), perform complex bit tests and comparisons, and then execute relative jumps based on the results. Intel has recently introduced a third generation microprocessor, the 80c196, a 16-bit processor mainly available in CMOS. It runs at 50 MHz and its other features include hardware multiply and divide, 6 addressing modes, high speed I/O, ADCs, serial communications channel, up to 40 I/O ports, 8 source priority interrupt controller, PWM (Pulse Width Modulator) generator and watchdog timer.

3.2.2.2 Motorola MC14500

The MC14500 Motorola microprocessor is a RISC (Reduced Instruction Set Computer) processor with 16 instructions and one addressing mode, no memory boundary (an infinite amount of memory) and comes in a small package (16 pin). The 68HC05 has a Von-Neumann architecture in which instructions, data, I/O and timers all share the same space. Stack pointer is 5 bits wide which limits the stack to 32 bytes deep. Some members of this family include on-chip ADC, PLL frequency synthesizer, and serial I/O and software security.

3.2.2.3 National Semiconductor COP400

The COP400 family is a P2CMOS 4-bit microcontroller from National Semiconductor which offers 512 bytes to 2K ROM and 32x4 to 160x4 RAM. Functions include microwire, timers' counters, 2.3 to 6.0 volt operation, ROMless modes, and OTP (One Time Programming) support. The COP400 family is known for its versatility. Over 60 different compatible devices are available for a wide range of requirements.

3.2.2.4 Texas Instruments TMS370

TMS370 from Texas Instruments is similar to the 8051 in having 256 registers, A and B accumulators, stack in the register page, etc. It also has a host of onboard support devices. Some members have all of them while others have a subset. The peripherals include: RAM, ROM (mask, OTP or EEPROM), two timers (configurable as timers/counters/comparators/PWM output), watchdog timer, synchronous serial port, asynchronous serial port, ADC (8-bit, 8-channel) and interrupts. Instruction set is mostly 8-bit with some 16-bit support. It has several addressing modes,

8 x 8 multiply and 16/8 divide. Clock speeds are up to 20 MHz which gives 5 MHz for bus access and instruction cycles.

3.2.2.5 Zilog Z80

The Z80 is a CISC microprocessor (Bolton, 2000) and was designed by Zilog. Its unique architecture includes three memory spaces: program memory, data memory and a CPU register file. On-chip features include UART (Universal Asynchronous Receiver Transmitter), timers, DMA and up to 40 I/O lines. Some versions include a synchronous/asynchronous serial channel and features fast interrupt response with 37 interrupt sources (Anderson, 1984). The Zilog Z80 is one of the 8-bit microprocessors available in the market today. It was produced in 1978 as an improvement of Intel 8080. In this research work the Z80 CPU has been employed. The next section discusses the Z80 microprocessor in detail. This includes its architecture, the registers and the flags.

3.3 The Z80 Microprocessor

The Z80 microprocessor is an 8-bit CPU with a 16-bit address bus capable of direct access to 64k of memory space. It offers about 1MIPS (Million Instructions Per Second) at 4 MHz clock speed with a minimum instruction time of 1μs and a maximum instruction time of 5.75 μs (Heath, 2003). It has a high level of integration, a powerful instruction set and minute power consumption. Operating from a single +5 V power supply and a single external clock, the processor contains 17 internal registers and in-built dynamic RAM refresh circuitry. It has three modes of interrupt response (Martin, 1982). Figure 3.1 shows the Z80 architecture. The Z80 is attractive due to its availability and low cost. The Z80 CPU has been manufactured in models A, B and C differing only in maximum speed (Zilog Z80 8-bit Microprocessor, htm, 2001).

The programming model of the Z80 microprocessor includes an accumulator, six main and six alternate registers, flag register, stack pointer, program counter and two index registers. These registers are now considered.

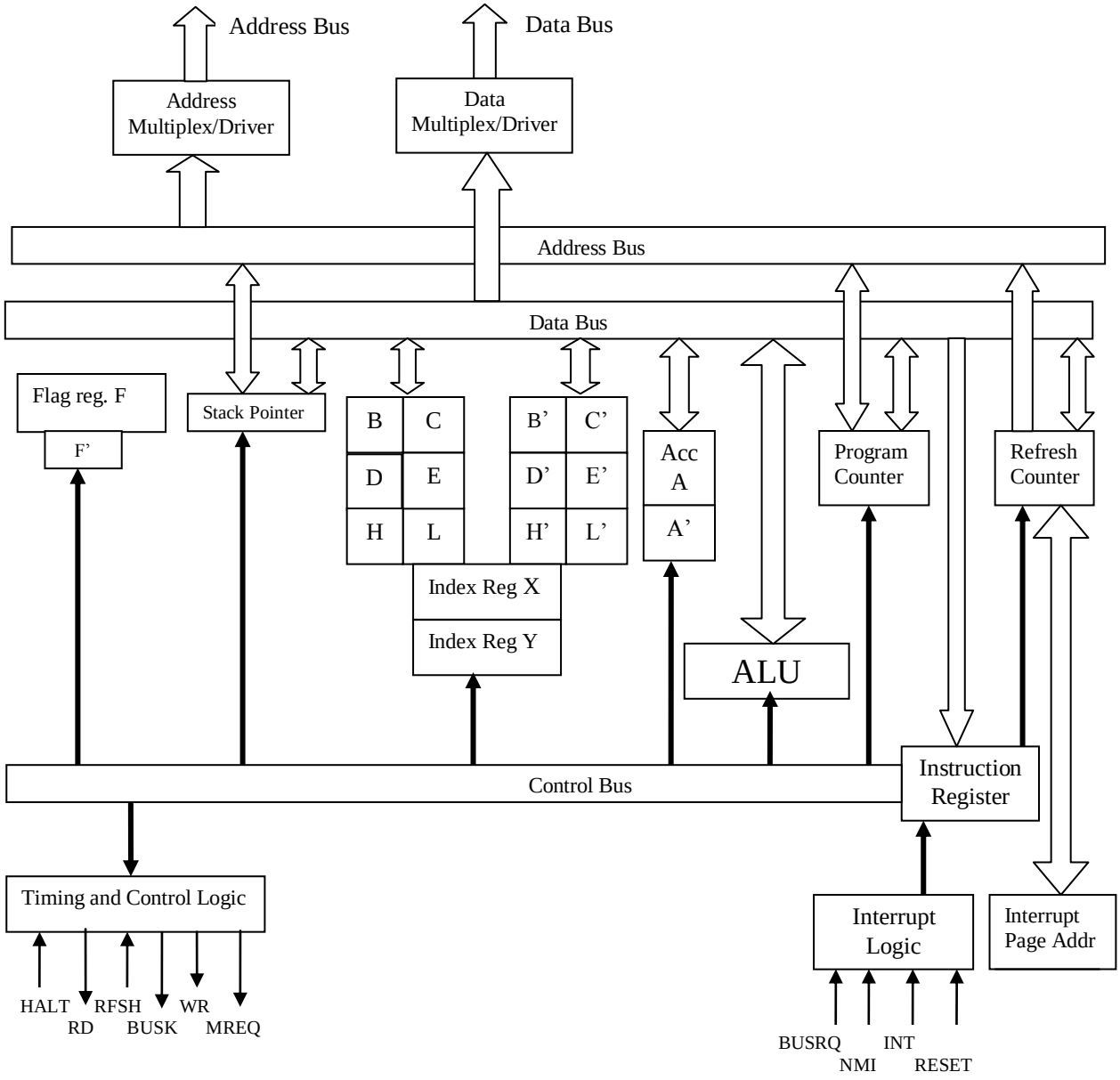


Figure 3.1: The Z80 microprocessor architecture which shows the registers, program and refresh counters, stack pointer, flag registers, address, data and control buses, interrupt, timing and control logics and the arithmetic logic unit (ALU).

3.3.1 Registers

These are memory locations within the microprocessor and are used to store information involved in program execution. In the Z80 microprocessor the accumulator register A is where data for an input to the ALU (Arithmetic Logic Unit) is temporarily stored. It is a temporary holding register for data to be operated on by the ALU and also, after the operation for holding the results. The ALU performs arithmetic and logic computations. Six 8-bit registers can be paired together to create three 16-bit registers for use as data storage or address pointers. The two register sets are the main (A-F, H, L) and the alternate (A'-F', H', L'). Each is 8-bit and only one set can be used at a time.

3.3.1.1 The Flag and Program Counter Registers

Flag registers or status registers contains information concerning the latest process carried out in the arithmetic and logic unit. It contains individual bits, called flags, with each bit having special significance. The status of the latest operation is indicated by each flag being set or reset. Thus, they could be used to indicate weather the last operation resulted in a negative result, a zero result, a carry output, an overflow or weather the program is allowed to be interrupted so an external event can occur. Table 3.1 shows the Z80 flags.

Flag	Set (1)	Reset (0)
Z	result is zero	result in not zero
N	result is negative	result is not negative
C	Carry is generated	carry is not generated
V	overflow occurs	overflow does not occur
I	interrupt ignored	interrupt is processed normally

Table 3.1: Common flags for the Z80 microprocessor showing the conditions for the set and reset status

The Z80 flag register has 8-bits but only 6 are used. Reference to flag register enables program flow to be controlled e.g. if results in previous ALU were zero, execution could be diverted. The contents of a flag register can be used to determine a JUMP, CALL or BRANCH by making reference to the previous ALU operation immediately before the particular instruction.

Program counter register is used to allow the microprocessor to keep track of its position in a program. It contains the address of the memory location that contains the next program instruction. As each instruction is executed the program counter is incremented by 1 each time so that the microprocessor executes instructions sequentially unless an instruction, such as JUMP, changes the program counter out of sequence.

3.3.1.2 The Stack and Instruction Registers

The Stack is a reserved area of RAM used for the temporary storage of data. The contents of the register may be stored temporarily or changed over via the stack. The return address following a CALL to a subroutine may be saved automatically using the stack, or the data and address may be saved during the servicing of an interrupt. Microprocessors can execute a program, while at the same time accepting external signals and responding to them: this is done by allowing the external device to interrupt normal program execution. When the interrupt is received, register contents and appropriate address information are quickly saved on the stack, the interrupting peripheral serviced, data is retrieved from the stack and program execution proceeds.

PUSH and POP are two instructions of the Z80 microprocessor associated with stack action. Data is pushed onto the stack and popped off it. Register pairs AF, BC, DE, HL, and index register IX and

IY can be stored on the stack. The last used address of the stack is kept in a special CPU register called the stack pointer (SP). The stack is arranged as a LIFO (last in first out) file, and so what is pushed in first is popped out last.

Special purpose registers such as the index register allows the original address specified in an instruction to be modified by the contents of this register. Instruction register (IR) stores an instruction. After fetching an instruction from the memory, the CPU stores it in the IR. It can then be decoded and used to execute an operation.

3.3.2 Addressing Modes

There are several ways of loading a number onto a register. If the operation is followed by the actual number to be loaded, then this is referred to as immediate addressing. If it is followed by the memory location where the number is stored then its direct or absolute addressing. And if the number is contained in another register it is referred to as register addressing. Other modes of addressing for the Z80 are implied addressing, where the operand is inherent in the instruction itself (e.g. NOP, INC A) and indexed addressing (e.g. LD A, IX+30H), which allows a base addition to be specified in one of the index registers to which a displacement is added. This mode allows easy reference to data tables which if standardized can be looked up with common offset values. Combinations such as immediate extended mode (e.g. LD BC 5DF0H) and register indirect (LD A, HL) are also possible. In bit addressing, the individual bits of a register or memory location may be inspected, set or reset (e.g. BIT 4, A: zero flag is set if bit 4 of register A is zero, and is reset otherwise). The opcode always comes first and tells the CPU what instruction is to be carried out

and then the operand supplies the rest of the information such as the number to be loaded or the memory location containing the number.

3.3.3 Fetch Execute Sequence

When a machine code program is run, the CPU of the Z80 microprocessor is fetching the opcode from memory into the instruction register and executing the instruction within the CPU. The CPU initially looks for the first instruction from the 'start address' or origin (ORG), and after that the PC will increment so that the CPU always knows where it has to go. If an unconditional jump instruction is met, the PC changes immediately to that value and jumps to it. If the jump is conditional, the jump is carried out only if the condition is met. If the jump is direct, program execution is continued from the address specified and if relative, the value given is added to the contents of the PC and the program jumps to that address.

3.3.4 Z80 CPU Pin Description

The Z80 microprocessor has 40 pins as shown in figure 3.2. Pin 17 - 28 are all active low. The pin out description is as follows:

ADDRESS: (Pins 1-5 and 30-40). The address bus is provided by a 16-bit line from A_0-A_{15} with A_0 –A_{10} (pins 30-40) and A_{11}-A_{15} (pins 1-5).

DATA: (Pins 7-10 and 12-15). The data bus is provided by 8- bit bus with the pin distribution as follows: D_0-14, D_1-15, and D_2-12, D_3-8, D_4-7, D_5-9, D_6-10 and D_7-13.

V_{cc}: (Pin 11). This is used to supply power to the CPU. It is connected to +5 V.

GND: (Pin 29). This is the ground pin and is set at 0 V.

$\overline{RD}$: (Pin 21). Memory read indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate onto the CPU data bus.

$\overline{WR}$: (Pin 22). Memory write indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.

RST: (Pin 26). The reset pin resets the CPU and initializes its operation.

RFSH: (Pin 28). The refresh signal refreshes dynamic memory to retain data

CLK: (Pin 6). The clock signal is applied to this pin. This signal is used to synchronize CPU operation and is obtained from an external clock circuit.

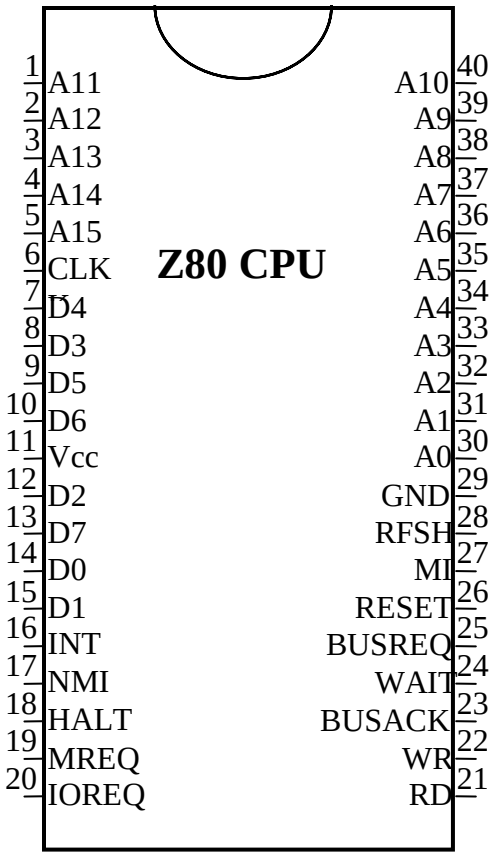


Figure 3.2: Pin out for the Z80 CPU showing the function of each pin. Pins 17-28 are all active low. The clock (CLK) pin 6 and reset pin 26 require external circuitry.

$\overline{\text{HALT}}$: (Pin 28). Halt state indicates that the CPU has executed a halt instruction and is awaiting an interrupt before operation can resume. While halted, the CPU executes NOPs (No OPerations) to maintain memory refresh.

$\overline{\text{INT}}$: (Pin 16). Interrupt request indicates that an I/O device is ready to be serviced.

$\overline{\text{IOREQ}}$: (Pin 20). Input/output request indicates that the lower half of the address bus holds a valid data I/O read or write operation.

$\overline{\text{MREQ}}$: (Pin 19). Memory request indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{\text{MI}}$: (Pin 27). Machine cycle, together with MREQ, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution.

$\overline{\text{NMI}}$: (Pin 17). Non maskable interrupt indicates an interrupt that cannot be ignored by the CPU and is reserved for critical situations such as power failure.

$\overline{\text{BUSACK}}$: (Pin 23). Bus acknowledge indicates to the requesting device that the CPU address bus, data bus and control signals, $\overline{\text{MREQ}}$, $\overline{\text{IOREQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$, have entered their high impedance states. The external circuitry can now control these lines.

$\overline{\text{BUREQ}}$: (Pin 25). Bus request forces the CPU address bus, data bus and control signals $\overline{\text{MREQ}}$, $\overline{\text{IOREQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$, to go to high impedance states so that the external circuitry can now control these lines.

3.4 The Clock System

A computers system's clock is measured as a frequency, usually expressed as a number of cycles per second. It consists of a crystal oscillator made of silver quartz in small tin container. When voltage

is applied to the quartz, it begins to oscillate at frequency rates dictated by the size of the crystal (Sivakumar, 1987). The oscillations emanate from the crystal in the form of a current that alternates at the frequency of the crystal. This alternating current is the clock signal. A typical computer system runs millions of these cycles per second, so that the speed is usually measured in megahertz (MHz). Design of a digital system may be synchronous or asynchronous. In synchronous circuits the output is synchronized to a clock signal. In choosing the clock frequency, consideration must be given to the overall effect on the system since memory devices should operate at higher speeds as compared to the processor. The Z80 requires a single phase 0 V to 5 V clock signal. This can be generated by a Transistor Transistor Logic (TTL) microprocessor crystal-controlled clock oscillator that uses part of 7404 hex inverter. The clock crystal used in this research had a frequency of 1.687 MHz. Figure 3.3 shows the circuit diagram for generating the clock signal.

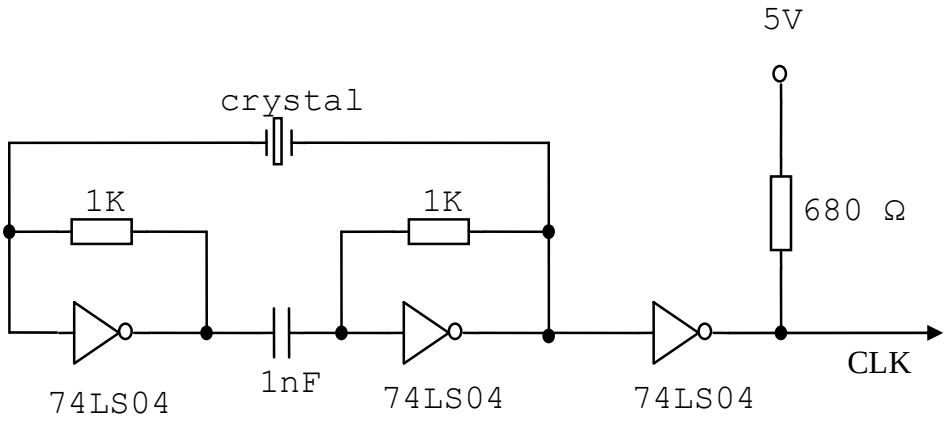


Figure 3.3: Circuit diagram for generating the clock signal. It uses part of the hex inverter IC 7404

The use of a crystal clock ensures precise and constant frequency. One period of waveform is called the 'T' cycle. All instructions to CPU take from 3 to 6 clock periods to complete. The basic operations composed of T cycles are called machine cycles. A timing diagram has to show the

relationship between clock pulses and bus signals. For an instruction opcode fetch for instance, we have to show the clock pulses, the address lines, the data lines, the relevant control lines, MREQ and $\overline{RD}$. Figure 3.4 shows the timing diagram for a fetch instruction.

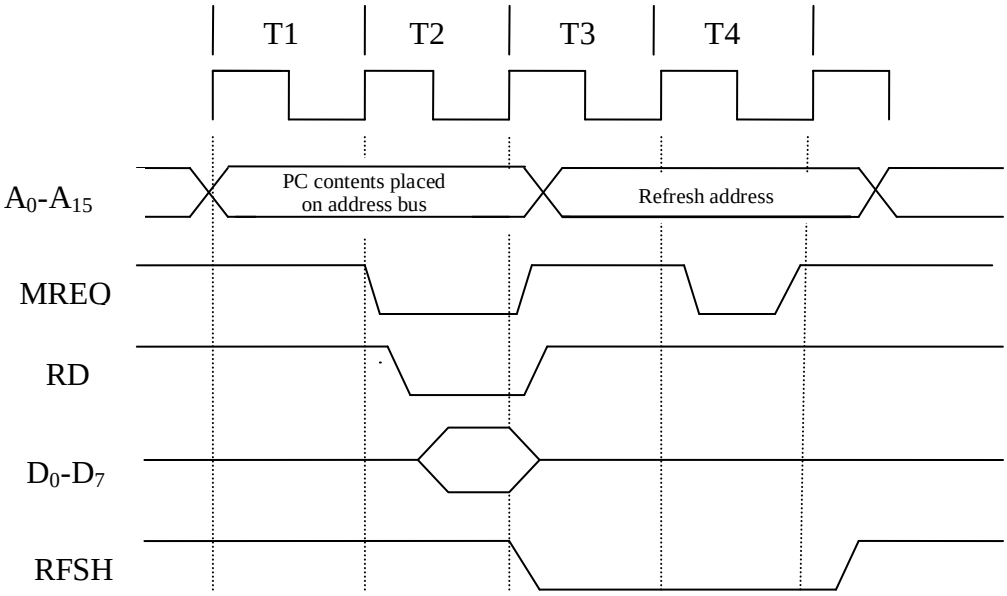


Figure 3.4: Z80 Timing diagram for fetch instruction. The $\overline{MREQ}$ and $\overline{RD}$ lines go low and then one half clock time later, the data is placed on the data bus.

The procedure is as follows: The program counter contents are placed on the address bus at the beginning of the MI machine cycle. And MI line goes low. One half clock time later, the memory request, MREQ, line goes active low. The read line, $\overline{RD}$, also goes active low. The data is put on the bus and the $\overline{RD}$ and $\overline{MREQ}$ signals go high. During T_3 and T_4 the address signal is refreshed. The $\overline{RD}$ part of the cycle can be prolonged by activating the $\overline{WAIT}$ line during the fetch cycle. For the write, $\overline{WR}$, the $\overline{MREQ}$ goes low but $\overline{RD}$ stays high as shown in figure 3.5. Data is written into the memory address.

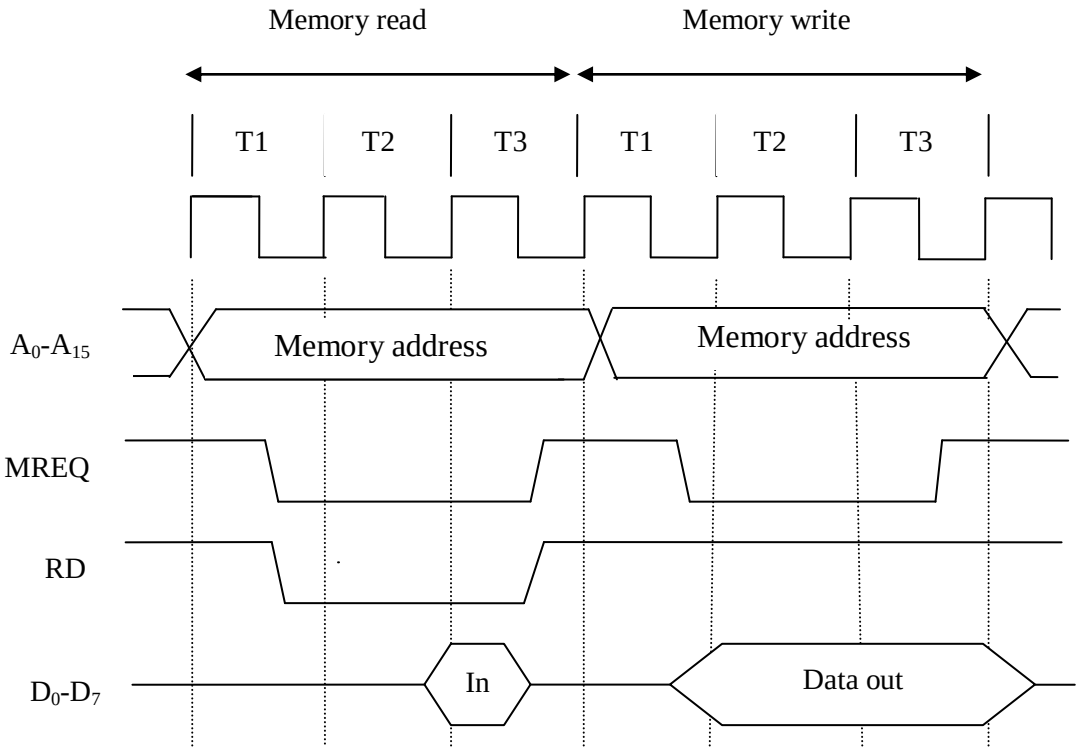


Figure 3.5: The Z80 timing diagram for read or write instruction. The $\overline{\text{MREQ}}$ and $\overline{\text{RD}}$ lines go low, the data is placed on the data bus and then they both go high. When the $\overline{\text{MREQ}}$ line goes low while $\overline{\text{RD}}$ stays high, data is written into the memory.

3.5 Microprocessor Buses

In order to facilitate the transfer of data and instructions between the microprocessor, memory and input/output devices, three major buses are used: the address bus, the data bus and the control bus. Any transmission medium that has more than one outlet at each end is called a bus. The processor data bus is a bundle of wires (or pins) used to send and receive data. The more signals that can be sent at the same time, the more data can be transmitted in a specified interval and, therefore, the faster the bus.

The address bus is the set of wires that carry the addressing information used to describe the memory location to which data is being sent, or from which the data is being retrieved. As with the data bus, each wire in an address bus carries a single bit of information. This single bit is a binary digit in the address. The more the wires (digits) used for these addresses, the greater the total number of address location is realized. The size (or width) of the address bus indicates the maximum amount of memory locations that a microprocessor can address. This number of address locations which can be accessed by the processor is known as its physical address space and is often expressed in terms of kilobytes, megabytes or gigabytes.

The control bus is of varying width, depending on the microprocessor being used. It carries control signals that are tapped into the microprocessor by other ICs to tell what type of operation is being performed. From these signals, the ICs can tell if the operation is a read, a write, an I/O, a memory access, or some other operation. These timing signals synchronize the flow of data between the processor and the memory sub-system or peripheral devices.

The size of the internal register is a good indication of how much information the processor can operate on at a time. Most advanced processors, from the 386 to the Pentium, use 32-bit internal registers and 32 to 64 bit data buses. Some processors have an internal data bus (made up of data paths and of storage units called registers) that is different from the external data bus. The 8088 and 386SX are examples of this structure. Each chip has an internal data bus twice the width of the external bus. These designs are called hybrid designs. The 386SX, for example, can pass data around internally with a full 32-bit register size. For communications with the outside world, however, the chip is restricted to a 16-bit wide data path. Internal registers often are larger than the

data bus, and thus the chip requires two cycles to fill a register before the register can be operated on.

3.6 System Interrupts

A peripheral is any electronic or electromechanical device which is external to the CPU such as a keyboard, a light display, a temperature sensor etc. Peripherals need to be serviced and the CPU should only be 'interrupted' when this need arises (Anderson, 1984).

An interrupt is like a CALL instruction. The peripheral supplies a signal indicating that it requires attention. The signal is taken to the interrupt pin INT or NMI, directly if there is only one peripheral or via some sort of prioritizing logic if there is more than one. At this point a special interrupt service routine (ISR) is assessed as the stack saves the return address. The ISR has a RET statement at the end of it or rather a 'RETI' (RETurn from Interrupt) in this case: this causes the return address to be popped off the stack and program execution continues as normal. A maskable interrupt can be turned off or disabled by the CPU. A non maskable interrupt cannot be ignored by the CPU and must be serviced. This type is reserved for critical situations such as power failure indication. If the system experiences a power failure the filter capacitors in the power supply will be able to hold up the supply voltage for several milliseconds. During this time, important data in RAM can be quickly written into disk for safekeeping by the non maskable ISR (Uffenbeck, 2003).

Interrupts are automatically disabled whenever a reset occurs or immediately after receipt of an interrupt request. A reset is a special type of interrupt. When a reset signal occurs, the main program stops running and usually the registers are reinitialized.

3.7 CPU and Peripheral Handshaking

The CPU and peripheral devices need to be synchronized so that the reading and writing occurs when the other is ready to transmit or receive data. One way is to use a delay routine, but this wastes CPU time. Another way peripherals and CPU can be synchronized is by means of interrupts. During an interrupt the microprocessor stops normal program execution in order to handle the interrupting device. A hardware method that can be used to achieve synchronization is to use I/O lines, called handshake lines, between the microprocessor and the peripheral to control the timing of data transfer. This process is termed handshaking.

Data input from asynchronous peripherals require that the microprocessor and the peripherals be similarly synchronized. If the microprocessor is faster than the peripheral, data may be lost unless handshaking is done. The microprocessor may be slowed down but some peripherals cannot be speeded up to achieve synchrony. In other instances peripherals such as hard disk are faster than the microprocessor. These cannot be slowed down and neither can the microprocessor be speeded up. Synchronization is achieved by handshaking signals between the microprocessor and the peripheral. These are in the form of "BUSY", "READY" and "ACKNOWLEDGE" flags. The peripheral sets the BUSY/READY flag low when it is ready for new data. The handshaking sequence begins with the microprocessor checking this flag. If it is low, it outputs a stroke pulse. This tells the input/output device that data are on the bus and should be fetched. The peripheral now sets its BUSY/READY flag while the character is being received and processed. Eventually an acknowledge signal is output by the peripheral. This signal acknowledges receipt of data and indicates that new data can be placed on the bus (Uffenbeck, 2003).

3.8 Primary Memory

Memories are storage devices used for storing programs and data and providing the data on request to other devices. They are made of arrays of registers arranged in a sequence. The registers are grouped in powers of 2 so that 2^{10} =1024 is an 8-bit memory with a value of 1K. Memories store binary information as instructions and data and provide that information to the microprocessor upon request. To execute programs, the microprocessor reads instructions and data from memory and performs computing operations in its ALU. Results are transferred to the output section to be displayed or stored in memory for later use. There are two types of memory, RAM and ROM.

3.8.1 Random Access Memory (RAM)

RAM is a volatile memory chip used for storage of users program. Information can be written into and read back out of it but information is lost when the computer system is switched off. Programs to be run currently are stored on RAM. There are different types of RAM. In SRAM (Static RAM), no special timing circuit is required. In DRAM (Dynamic RAM), timing circuits are required since the memory cells have to be refreshed by reading the contents of each cell and then writing it back again immediately (every 20 ms or so) afterwards. This is because the tiny capacitors that store data cannot hold charge for long and if not refreshed periodically data would be lost.

Battery backed-up static RAM is useful when a large non-volatile program and data space is required. A major advantage of static RAM is that it is much faster than other types of non-volatile memory, so it is well suited for high performance application. There also are no limits as to the number of times that it may be written to, so it is perfect for applications that keep and manipulate large amounts of data locally.

The RAM used in this research work is a 6116 2K IC. It is a 2048 x 8 bit static RAM built in CMOS and pin compatible with 2K EPROMs. It offers access times of 150 ns and data retention at voltages down to 2 V with standby currents as small as 10 nA at 3 V. The chip operates from a single voltage supply of +5 V and is pin compatible with the Z80. The IC 6116 has 24 pins shown in the pin out diagram in figure 3.6.

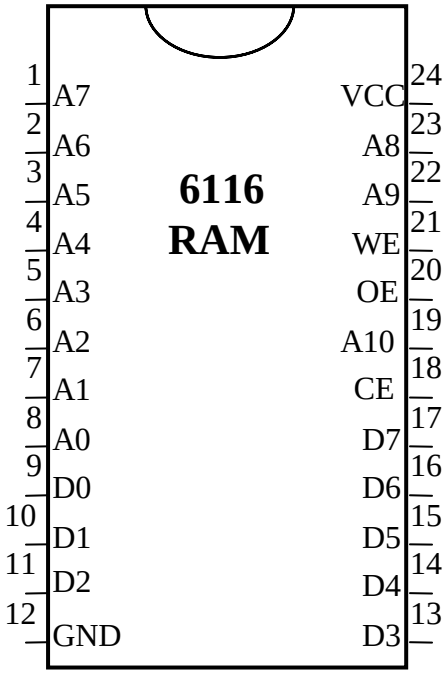


Figure 3.6: Pin out diagram of a 6116 RAM. This is an 8-bit IC with 11 address lines compatible with the Z80 microprocessor. The $\overline{CE}$, $\overline{OE}$, and $\overline{WE}$ are all active low.

3.8.2 Read Only Memory (ROM)

The second type of memory is the ROM and is a non-volatile memory chip that can only be read from by a microprocessor but cannot be written into. Since it is non-volatile, programs can be permanently stored in it. Some types of ROM available are EEPROM (Electrically Erasable Programmable Read Only Memory) and then EPROM (Erasable Programmable Read Only

Memory). EEPROM is relatively slow, and the number of erase/write cycles allowed in its lifetime is limited. When there is a requirement for large amounts of non-volatile program memory, EPROM is used. It is both faster and permits more erase/write cycles than EEPROM.

This research work has employed the 2716 EPROM. It is a 16,384-bit ultraviolet erasable read only memory organized as 2048 x 8 bit words. The pin out diagram for the 2716 EPROM is shown in figure 3.7. It has an access time of 350 ns. The inputs and outputs are TTL compatible.

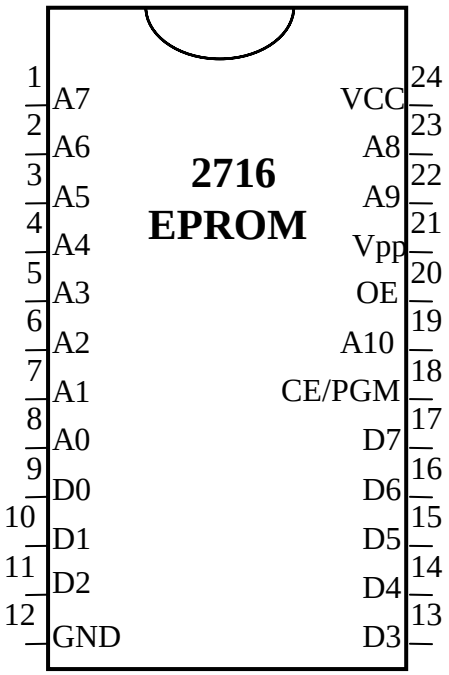


Figure 3.7: Pin out diagram of 2716 EPROM. Pins 21 (Vpp) and 18 (PGM) are used when burning the program into the chip.

3.9 System Interfacing

When a microprocessor is used to control some system it has to accept input information, respond to it and produce signals to implement the required control action. Thus there can be input from sensors to feed data in and output control signal to external devices such as relays and motors. The

term peripheral is used for a device such as a sensor, keyboard, etc, which is connected to a microprocessor. It is normally not possible to connect such peripheral devices directly to the microprocessors' bus system due to lack of compatibility in signal form and levels. Thus a circuit known as an interface is required and is connected between the device and the microprocessor.

Input-output systems are used to interface a microprocessor with peripheral devices. These may be constructed from individual chips or a programmable universal interface. The peripheral connected to a microprocessor system requires more current than can be supplied by the microprocessor. This problem is overcome by connecting the peripheral to the bus by means of a buffer which acts as a current amplifier. Buffers can also provide isolation between the microprocessor and the higher power system.

Since a number of peripherals may have to share the same data lines, there is need for the microprocessor to enable just one of the devices at a time with the others disabled. This is achieved by tristate buffers, where the output can be low, high or floating. Buffers are available as ICs e.g. SN 74LS244. This has 8 non-inverting enable high buffers.

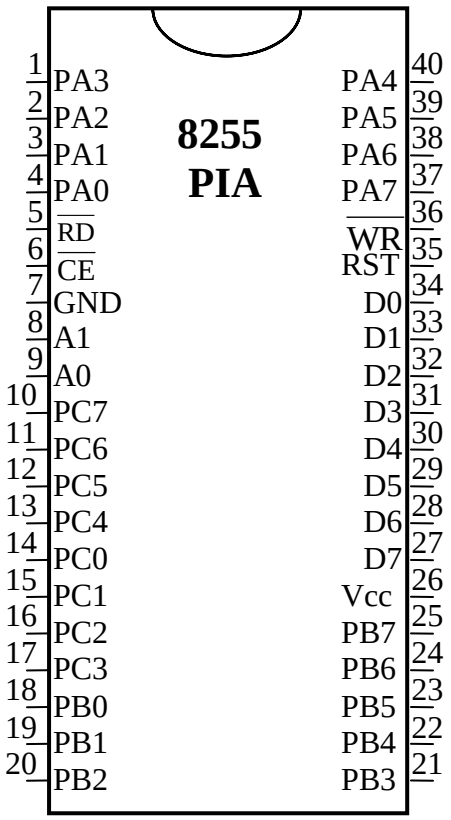
There are two hardware techniques for getting data into and out of a microprocessor. The first is the parallel interface. Data is read and written into the input output device through ports that are one byte wide. All data bits are transferred in parallel. The second technique is the serial interface where a parallel to serial converter is used to transmit the 8-bit data one after the other in time and then a serial to parallel converter used to reform the parallel data bytes. The advantage of this technique is that only three wires are needed to implement the interface: serial data in, serial data out and ground.

Three conductors are cheaper than eight or nine and they can be transmitted over telephone lines by a modem. The disadvantage is that it takes at least 8 times longer than in parallel transmission (Uffenbeck, 2003). The interface used in this research work is the Intel 8255 PIA which is discussed in the next section.

3.9.1 Intel 8255 Programmable Interface adapter (PIA)

The 8255 PIA chip is a general purpose programmable I/O device designed for use with Intel microprocessors. It has 24 I/O pins, which may be individually programmed in two groups of twelve and used in three major modes of operation (Tooley, 1996). It is a powerful tool with enough flexibility to interface almost any I/O device to the CPU bus without additional external logic. Each peripheral device has a unique service routine associated with it, which manages the software interface between it and the CPU. The functional configuration of the 8255 PIA is programmed according to this service routine and actually becomes an extension of the system software. Figure 3.8 shows a schematic diagram of the 8255 PIA.

Interfacing and control between the CPU and the PIA is performed using the data bus D_0-D_7, and the standard, $\overline{RD}$, $\overline{WR}$ and chip select $\overline{CE}$ controls. All three 8255 ports have I_{OC}=107 mA and I_{OH}=200 µA. Thus they can drive one TTL load. Ports B and C can service any set of eight lines with 1 mA at 1.5 V and thus can drive solid state relays and transistor drivers (Bolton, 2000). LEDs can be used to show the status of any of these ports. Due to its low power consumption, it is well suited to microcomputers. For LED requiring 10-20 mA current a TTL buffer should be used.



.

Figure 3.8: Pin configuration of Intel 8255 PIA. Pins 5 ($\overline{RD}$), 6 ($\overline{CE}$) and 36 ($\overline{WR}$) are all active low while 35 (reset) is active high.

3.9.2 Modes of Operation

In the program mode (mode 0), each group of twelve I/O pins may be programmed in sets of four to be input or output. This mode provides a simple input and output operations when no handshaking is required. In this mode, data is simply written into or read from the specified port.

In mode 1, each group may be programmed to have 18 lines of input or output. Of the remaining four pins, three are used for handshaking and interrupt control signals. Mode 1 is basically the same as mode 0, except that the data is transferred under the control of handshaking (strobe) pulse. The handshaking signals are derived from port C. Therefore, each of the two groups A and B consists of 8-bit data port (port A or Port B) and one 4-bit control port (upper port C or lower port C). As

before, port A or port B can be individually configured as input or output ports. Since data is transferred using a strobe, both inputs and outputs are latched.

Mode 2 is a bi-directional bus, which uses 8-lines for a bi-directional bus and five lines borrowing one from the other group for handshaking. Mode 2 provides a means of communicating with a peripheral through a bi-directional I/O bus. However, only port A is applicable to mode 2; port B cannot be configured bi-directionally. Mode 2 is structured on a single 8-bit bus for both transmission and reception of data. Control signals are provided by a 5-bit control port (port C) to maintain proper flow discipline in a manner similar to mode 1.

The modes of the PIA are configured by a control word send to the control register. Bit 3 selects mode 0 and 1 and bit 6 and 7 selects mode 0, 1 or 2. Bit 8 sets the mode flag as shown in figure 3.9. The 24 I/O pins are divided into two groups as follows: Group A- port A and upper port C, group B- port B and lower port C. Any of the four ports: port A, lower port C, upper port C and port B, can be input or output. A total of 16 different input/output configurations are possible. Data sent from CPU gets latched on the ports, but the inputs to the ports from the I/O devices are not latched.

The system software can select any of the three modes of operation. When the reset input goes “high”, all ports will be set to the input mode. After the reset is removed, the PIA can remain in the input mode with no additional initialization required. During the execution of the system program, any of the other modes may be selected using a single output instruction.

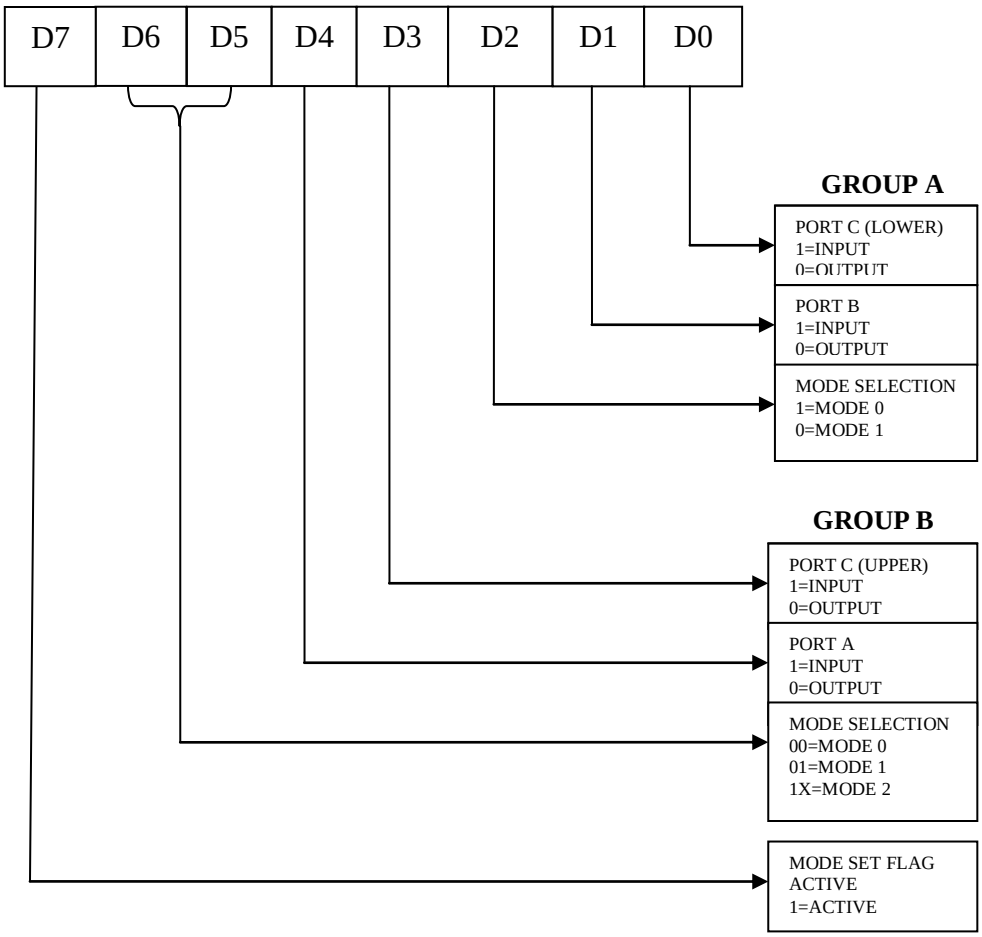


Figure 3.9: The system control code word format for configuring the ports for the 8255 PIA's. Three mode configurations are possible, mode 0, mode 1 and mode 2, with mode 2 providing a bi-directional bus. The modes are configured by sending the appropriate control word to the control register.

3.9.3 Interfacing to the Mains

Optically coupled devices and triacs can be used to ensure complete isolation of the microprocessor system from the mains. A TRIAC, (TRIode for Alternating Current) is an electronic component approximately equivalent to two silicon-controlled rectifiers joined in inverse parallel (paralleled but with the polarity reversed) and with their gates connected together. This results in a bidirectional electronic switch which can conduct current in either direction when it is triggered. It can be

triggered by either a positive or a negative voltage being applied to its gate electrode. Once triggered, the device continues to conduct until the current through it drops below a certain threshold value, such as at the end of a half-cycle of alternating current mains power. This makes the triac a very convenient switch for AC circuits, allowing the control of very large power flows with milliampere-scale control currents. Low power triacs are used in many applications such as light dimmers, speed controls for electric fans and other electric motors, and in the modern computerized control circuits of many household appliances. However, when used with inductive loads such as electric fans, care must be taken to assure that the triac will turn off correctly at the end of each half-cycle of the ac power. The opto isolator comes in a standard DIL package. A typical device offers 5000 V isolation between the input and output.

3.10 Signal Conditioners

Signal conditioning includes the amplification, filtering, converting, and other processes required to make sensor output suitable for reading by computer boards. It is primarily utilized for data acquisition, in which sensor signals must be normalized and filtered to levels suitable for analogue-to-digital conversion so they can be read by computerized devices. Signal conditioning type, form factor, device specifications, signal inputs, sensor inputs, transducers and excitation, outputs and user interface are all important to consider when searching for signal conditioning devices. Other specifications to consider include application, software, memory and storage, network specifications, filter specifications, amplifier specifications and environmental parameters.

Types of devices that use signal conditioning include signal filters, instrument amplifiers, sample-and-hold amplifiers, isolation amplifiers, signal isolators, multiplexers, bridge conditioners, analog-

to-digital converters, digital-to-analog converters, frequency converters or translators, voltage converters or inverters, frequency-to-voltage converters, voltage-to-frequency converters, current-to-voltage converters, current loop converters and charge-converters. Transducer output voltages are usually in the millivolts range and as such require amplification before a useful signal can be obtained for interfacing purposes. The amplification process can itself introduce other problematic features including noise, stability, common mode rejection and distortion (Fraser and Milne, 1999).

Device specifications that are important to consider when searching for signal conditioning instruments include differential analog input channels, digital I/O channels and accuracy. Differential channels use the difference between two signals as an input; common mode is filtered out. In some systems, differential inputs are combinations of two single-ended inputs; in this case, twice the number of differential channels would be available as single-ended inputs. Digital or discrete channels are used for low-level on-off signals used in applications such as communication, user interface, or control. Accuracy depends on the signal conditioning linearity, hysteresis, temperature considerations, etc. It is usually represented as percentage full scale of the measurement range.

Signal inputs accepted by signal conditioners include DC voltage and current, AC voltage and current, frequency and charge. Sensor inputs can be accelerometer, thermocouple, thermistors, RTD (Resistance Temperature Detector), strain gauge or bridge, and LVDT (Linear Variable Displacement Transducer) or RVDT (Resistance Variable Displacement Transducer). Specialized inputs include encoder, counter or tachometer, timer or clock, relay or switch, and other specialized inputs. Outputs for signal conditioning equipment can be voltage, current, frequency, timer or

counter, relay, resistance or potentiometer, and other specialized outputs (http://data-acquisition.globalspec.com).

The next sections give an overview of the various sensors used to measure temperature, light and relative humidity. The sensors used in this research work are then highlighted and discussed.

3.11 Temperature Sensors

Several temperature sensing techniques are currently in use. Some of these include RTDs, thermocouples, thermistors, and sensor ICs. The choice of a sensor depends on the required temperature range, linearity, accuracy, cost, features, and ease of designing the necessary support circuitry.

RTDs use a sensing element whose resistance varies with temperature. A platinum RTD consists of a coil of platinum wire wound around a bobbin, or a film of platinum deposited on a substrate. The sensors resistance-temperature curve is a nearly-linear function. This nonlinearity curve is a disadvantage but since it is predictable it can be corrected. To process an RTD signal, a known accurate current is forced through the sensor, and the voltage across the sensor is measured. The cost of RTDs tends to be high. Also, self-heating can occur in these devices. The power required to energize the sensor raises its temperature, which affects measurement accuracy. Circuits that drive the sensor with a few milliamps of current can develop self heating errors of several degrees.

Another type of resistive sensor is the thermistor. A thermistor's resistance-temperature function is very nonlinear, and thus to measure a wide range of temperatures, it is necessary to perform

substantial linearization. Thermistors can be affected by self-heating, usually at higher temperatures where their resistances are lower.

A thermocouple consists of a junction of two wires made of different materials. One junction is at the temperature to be measured and the other at a known temperature. The output voltage is approximately proportional to the difference in temperature between the junctions. Because thermocouples sensitivity is rather small (in the order of microvolt per degree C), a low-offset amplifier is needed to produce a usable output voltage. Nonlinearities in the temperature-to-voltage transfer function amount to many degrees over a thermocouples operating range and, as with RTDs and thermistors, often necessitate compensation circuits or lookup tables. They have low thermal mass and wide operating temperature range, which can extend to about 1700 ^{o}C with common types. A silicon temperature sensor is an integrated circuit, and includes an extensive signal processing circuitry within the same package as the sensor. These sensors operate over the nominal IC temperature range of -55 ^{o}C to +150 ^{o}C. Examples are LM235 and LM135. The sensors in this family operate like 2-terminal shunt voltage references; the third terminal allows one to adjust the accuracy. The error of an untrimmed LM135A over the full -55 ^{o}C to +150 ^{o}C range is less than ±2.7 ^{o}C. Using an external trim pot to adjust the accuracy reduces the error to less than ±1 ^{o}C over the same temperature range. These sensors can measure temperatures below 0 ^{o}C by using a pull-down resistor from the output pin to a voltage below the ground.

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius temperature. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±1⁄4 ^{o}C at room temperature and ±3⁄4 ^{o}C over a full −55

to +150 ^{o}C temperature range. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies of 4 to 30 volts, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1 ^{o}C in still air. The LM35 is rated to operate over a −55 ^{o}C to +150 ^{o}C temperature range, while the LM35C is rated for a −40 ^{o}C to +110 ^{o}C range. The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Any linear circuit connected to wires in a hostile environment can have its performance adversely affected by intense electromagnetic sources such as relays, radio transmitters, motors with arcing brushes, etc., as its wiring can act as a receiving antenna and its internal junctions can act as rectifiers. In such cases, a 0.1µF bypass capacitor from the power supply pin to ground will help clean up power supply noise. Output filtering can be added as well.

When the LM35 is applied with a 200 Ω load resistor it is relatively immune to wiring capacitance because the capacitance forms a bypass from ground to input, not to output. The two-wire lead is twisted to minimize stray capacitance. The circuit diagram in figure 3.10 shows the LM35 two-wire remote sensor. The output, V_{out} is 10 mV/^{o}C and is 1 ^{o}C higher than the ambient temperature. The device has a range of +2 ^{o}C to +40 ^{o}C. The 6.8K resistor can be varied using a 10 Ω rheostat for gain adjustment.

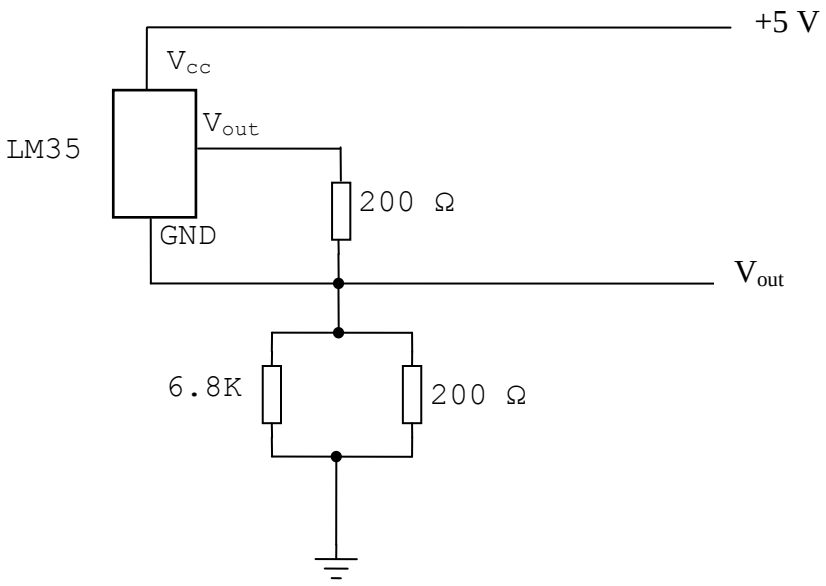


Figure 3.10: Two-wire remote temperature sensor based on LM35 IC

A temperature sensor produces an output, whether analog or digital, that depends on the temperature of the sensor. Heat is conducted to the sensing element through the sensors package and its metal leads. A sensor in a metal package such as an LM35 will have a dominant thermal path through the package. For sensors in plastic packages, the leads provide the dominant thermal path. A board-mounted IC sensor will measure the temperature of the circuit board (especially the traces to which the leads are soldered). If the board's temperature is very close to the ambient air temperature, the sensor's temperature will also be very near that of the ambient air. To measure the temperature of something other than the circuit board, the sensor and its leads should be at the same temperature as the object. This usually involves making good mechanical and thermal contact by, for example, attaching the sensor (and its leads) to the object with thermally-conductive epoxy. If electrical connections can be made directly from the sensor's leads to the object, soldering the leads of an IC sensor to the object will give a good thermal connection.

If the ambient air temperature is the same as that of the surface being measured, the sensor will be within a fraction of a degree of the surface temperature. If the air temperature is much higher or lower than the surface temperature, the temperature of the sensor will be at an intermediate temperature between the surface temperature and the air temperature. A sensor in a plastic package (a TO-92 or SOT-23, for example) will indicate a temperature very close to that of its leads (which will be very close to the circuit board's temperature), with air temperature having a less significant effect. A sensor in metal package will usually be influenced more by air temperature. The influence of air temperature can be further increased by gluing or clamping a heat sink to the metal package.

If liquid temperature is to be measured, a sensor can be mounted inside a sealed-end metal tube and can then be dipped into a bath or screwed into a threaded hole in a tank. Temperature sensors and any accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true for current temperature sensors if the circuit may operate at cold temperatures where condensation can occur.

To measure air temperature using a sensor in a metal can, a clip-on heat sink can bring the sensor's temperature close to ambient. This decreases the thermal time constant and speeds up the response time in slowly moving air. A small thermal mass may be added to the sensor to give the steadiest reading despite small deviations in the air temperature. If the sensor is in a plastic package, it may need to be mounted on a small "subboard", which can then be thermally isolated from the main board with long leads (National Semiconductor, 1995).

3.12 Relative Humidity Sensors

Relative humidity (RH) is the ratio of the amount of moisture in the air (via moisture mass or vapour pressure) and the maximum amount of moisture that could exist in the air at a specific temperature (via maximum moisture mass or saturation vapour pressure) expressed as a percentage. Wet bulb temperatures can be used along with the dry bulb temperature to calculate relative humidity (Habby, 2005). Dry bulb temperature is the actual air temperature while wet bulb temperature is the lowest temperature that can be obtained by evaporating water into the air at constant pressure. Since evaporation takes up heat, the thermometer whose bulb is surrounded by a wet cloth will cool to a lower temperature than a thermometer with a dry bulb at the same time and place.

If the air temperature is T_c and the wet bulb temperature T_{wb}, then first we calculate the actual mixing ratio of the air (W) using the following formula (Palmer, 2000):

$$W = \left[C_p(T_c - T_{wb}) - L_v(E_{swb}/P)\right] / \left[-(T_c - T_{wb})(C_{pv}) - L_v\right] \quad (3.5)$$

where,

W = actual mixing ratio of air,

C_p = specific heat of dry air at constant pressure (J/g) ~1.005 J/g,

C_{pv} = specific heat of water vapour at constant pressure (J/g) ~4.186 J/g,

L_v = Latent heat of vaporization (J/g) ~2500 J/g,

T_c = air temperature in degrees Celsius,

T_{wb} = wet bulb temperature in degrees Celsius,

E_{swb} = saturation vapour pressure at the wet bulb temperature (mb) and

P = atmospheric pressure at surface~1013 mb at sea-level.

Once we have the actual vapour pressure, we can use the following formula to calculate the saturation mixing ratio for the air.

$$W_s = \frac{E_{swb}}{P} \tag{3.6}$$

Once we have the actual mixing ratio and the saturation mixing ratio, we can use the following formula to calculate relative humidity.

$$RH = \frac{W}{W_s} 100\% \tag{3.7}$$

The latent heat of vaporization (L_v) varies slightly with temperature. The value given above is an approximate value for the standard atmosphere at 0 ^oC. Due to the large numbers of approximations using these formulas, the final result has an error of 10 percent (Palmer, 2000).

The humidity of the air can be measured using a capacitive humidity sensor which operates by sensing changes in capacitance of a thin-film polymer membrane as it absorbs moisture from its surroundings. The sensor is made up of a polymer film coated on both sides with a very thin air-permeable gold layer to form a capacitive element. Changes in relative humidity of the surrounding air cause a change in dielectric constant of the polymer film leading to a change of sensor capacitance. This capacitance is then measured by an onboard electronic circuit.

Conventional sensors have three primary weaknesses. Due to the relatively large dimensions of the sensor elements, as well as the aging of the polymer layer, they have a poor long-term stability. Also, before use, capacitive humidity sensors must undergo a complicated calibration process. The user must have complex and expensive calibration and reference systems, as well as external electronic components, such as memory components. Lastly, additional problems arise directly from

the analogue measurement principle, which links the stability of the operating voltage inseparably to the sensor accuracy (Christian, 2006).

The SHT11 is a highly integrated and extremely small humidity sensor. It is a CMOS multi sensor module on a single chip with a calibrated digital output. The device includes two calibrated micro sensors for relative humidity and temperature which are coupled to an amplification, analogue-to-digital converter and serial interface circuit on the same chip. A micro-machined finger electrode system with different protective and polymer cover layers forms the capacitance for the sensor chip, and, in addition to providing the sensor property, simultaneously protects the sensor from interference. The analogue to digital conversion, which is also performed on chip, makes the signal extremely insensitive to noise. A checksum generated by the chip itself is used for additional reliability. Other advantages include very short response times (4 s at lie), high precision (±2% to ±5% according to configuration), low power consumption (<3 µA at standby), and small footprint (7 x 5 x 3 mm). The sensor chip can be connected directly to any microprocessor system by means of the digital 2-wire interface. This sensor overcomes the disadvantages associated with conventional sensors.

The HIH-4000 humidity sensor is a laser trimmed capacitive sensing element which outputs a voltage that is linearly proportional to the relative humidity. The output voltage of the HIH-4000 is ratiometric to the supply voltage of the sensor. Thus, in order to calculate relative humidity, not only must the output voltage of the sensor be recorded, but also the supply voltage must be monitored. It has an on-chip integrated signal conditioning. The sensing elements multilayer construction

provides excellent resistance to application hazards such as wetting, dust, oils, and common environmental chemicals.

Its output is linear and proportional to relative humidity and it varies from 0.8 V to 3.9 V. It draws a current of 200 µA at 5 V and has an accuracy of 2% RH with a response time of 50 s in still air at 25 ^{o}C. It has an output of 0.8 V to 3.9 V and operates on a voltage of 4.0 to 5.8 V. its output voltage varies from 0.8 V to 3.9 V. Its temperature operating range is -40 ^{o}C to 85 ^{o}C. It comes in a 3 pin, solderable SIP in molded thermoset plastic housing with thermoplastic cover (www.honeywell.com.sensing, 1998; Arwtray, 1998). This sensor is light sensitive and should therefore be shielded from bright light. This research has made use of the HIH-4000 humidity sensor.

3.13 Light Sensors

Light sensors often use an infrared LED as a light source. Infrared LEDs have a greater intensity than LEDs that emit visible light. And when infrared photodiodes are used the sensors are relatively insensitive to ambient light.

Photoelectric light sources are often modulated at a given frequency to prevent interference from ambient light. But flashes or reflections can still fool light-activated sensors. This problem is solved by using a modulated receiver. Here the detector is synchronized to the light source frequency. For example, suppose emitted light has a frequency of 5 kHz, with pulse duration of 10 µs. The emitter LED and detector need only be on for 10 µs. They can be off for the 190 µs that light is not

transmitted. Such third-generation devices have fewer false signals because the chance of ambient light activating the receiver is only 1 in 19.

One trend in photoelectric sensors is the development of modular devices. Sensor heads can be combined with a separate base and power supply, and with various logic options. The approach tailors a sensor for particular needs. Another advantage is that sensor heads or logic options can be changed in the field without disturbing wiring or beam alignment. This minimizes downtime. Reflective light sensors (RLSs) are frequently employed to sense an object's presence. An RLS differs from other light sensors in that the target does not simply break a light beam but reflects light to a detector. Through the years, RLSs have been refined so that they are less sensitive to ambient light and can detect minute or transparent objects. Some devices can even determine the distance.

Light Dependent Resistors (LDRs) are devices whose resistance varies with light intensity. When these are connected in a circuit to form part of a potential divider, the output of the circuit varies from V_{cc} to 0 depending on the light intensity. There are just two ways of constructing the voltage divider, with the LDR at the top as shown in figure 3.11, or with the LDR at the bottom.

V_{out} is given by,

$$V_{OUT} = \left\{ \frac{R_{LDR}}{R_{LDR} + R} \right\} V_{CC} \qquad (3.4)$$

When R_{LDR} tends to 0, V_{out} will tend to 0 and when it tends to infinity, V_{out} becomes equal to V_{cc}. thus this circuit can be used to sense light or darkness depending on the configuration.

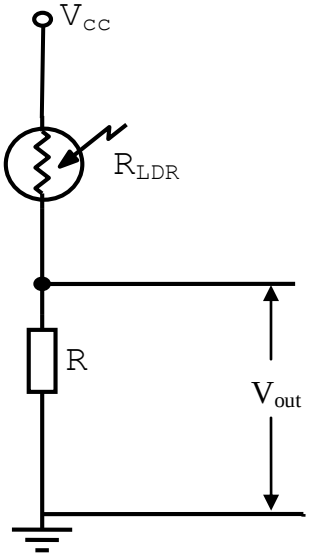


Figure 3.11: Circuit configuration for light sensor illustrating a voltage divide network. V_{out} will depend on the resistance of the LDR. It will be V_{cc} for total darkness and 0 for bright light.

The sensors used in this work all have analogue outputs. This voltages need to be digitized so that they can be interfaced to the microprocessor. This is done by analogue digital converters. This is discussed in the next section.

3.14 Analogue Digital Converters (ADC)

The ADC0804 is CMOS 8-bit successive approximation ADC converter that uses a differential potentiometric ladder and has been used in this research work. This ADC appears like memory

location or an I/O port to the microprocessor and no interfacing logic is needed. The voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution. It is compatible with Z80 microprocessor. It has an access time of 135 ns and works with 2.5 V voltage reference. It has an on-chip clock generator. It uses a single voltage supply of +5 V supply. It has a conversion time of 100 µs and requires a voltage input of 2.0 - 15 V. No zero adjust is required. It is available in a standard 20-pin DIL package. The pinout for the ADC0804 is shown in figure 3.12.

3.15 Programming Languages

Microprocessors can only run with a program. This section looks at programming languages in general and then focuses on the programming language used in this research work.

A program is a sequence of instructions that is used to operate a microprocessor system to produce specific results (Bolton, 2000). To run a program, a microprocessor must have the program stored in binary form in successive memory locations. The set of instructions that can be used to construct a program is called a programming language. There are three language levels that can be used to write a program for a microprocessor: machine language, assembly language and high-level language (Ramsay, 1981). The next sections will discus the languages in historical order, starting with machine language, then assembly language and finally high level language.

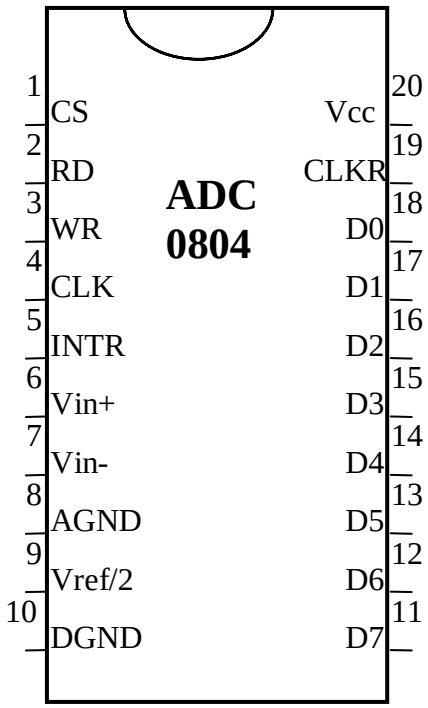


Figure 3.12: Pin out for the ADC 0804 analogue digital converter. This chip has an internal clock and offers an 8-bit resolution. Pins 1-3 are all active low.

3.15.1 Machine Language

Machine Language is a program written as a sequence of binary codes for the instructions that a microcomputer is to execute. The code is defined by the designer and is specific for the particular microprocessor. Small monitor programs allow the entry of these instructions and data via hex keypads or terminals by the programmer. The disadvantage of this method is that programs are hand-coded, which is a slow process and also leads to translation errors e.g., the programmer could easily misread 8 as the value B, I for 1 or O for 0. The assembly language offers some advantage as seen in the next section.

3.15.2 Assembly Language

Assembly language programming is done by writing machine instructions in mnemonic form. Using an assembler program, these mnemonics are then converted into actual processor instructions and associated data. Mnemonics are symbols or the shortened form of the English words for the operations to be performed by instructions. Mnemonics are more meaningful than hexadecimal or binary values and thus reduce the chances of making an error. They are also easier to remember than bit values. Assembly language statements are usually written in a standard form that has four fields: Label field, OPCODE field, Operand field and Comment field. Table 3.2 shows the fields and their examples.

A label is a symbol or group of symbols used to represent an address which is not specifically known at the time the statement is written. A colon usually follows labels. Here, the label START is equal to the address of the instruction LD A, 00H. The OPCODE (operation code) field of the instruction contains the mnemonic for the instruction to be performed. The operand field of the statement contains the data, the memory address, the port address or the name of the register on which the instruction is to be performed. The comment field starts with a semicolon or an asterisk and is not part of the machine language program. The comments in a program remind one of the functions that an instruction performs in the program.

The input program undergoes the processing whereby the assembler reads the source program saved in the storage device (hard disk or a floppy disk). The assembler program generates object codes and list files as output. The object file contains the machine codes for instruction and information

about the addresses of instructions and is the one to be loaded into the memory for execution. The list file contains the assembly language statements, the binary codes for each instruction in hexadecimal notation, the offset address for each instruction and it also indicates any typing or syntax errors made in typing the source program.

Label Field	OPCODE Field	Operand Field	Comment
START:	LD A,	00H	; Clear the A register.

Table 3.2: Assembly language coding field for opcode illustrating the instruction: load the accumulator with 00H. The label START is the address of this instruction and can be referred to in the program by a jump instruction.

Assemblers translate the source program written in symbolic language into an object binary program. The symbolic version of the software is a called a source program and the hexadecimal machine version the object program. The source program provides the assembler with the source of conversion, and the hexadecimal machine language output of the assembler is its object. Most assemblers convert the source code into object code by passing or scanning the code twice. This type of assembler is called a two-pass assembler. During the first pass of the assembler, the source program is scanned and a table is constructed by the assembler. Entry in the label table contains the label and the address at which the label appears in the program. The assembler always assembles the first address of the program is stored at memory location 0000H unless directed otherwise with the ORG command. During the first pass the assembler determines the length of each instruction by updating an internal program counter. This counter allows the assembler to complete the label by

equating each label to this program counter. Once the label table is complete, the assembler begins its second pass of the source program.

Some of the advantages assembly language programs have are reduced errors and faster translation times and any changes can be made easily and fast. The disadvantages of assembly language programming are the programmer requires knowledge of the processor architecture and instruction set and that any change in the hardware requires complete rewriting of the program. Complex programs are best written in high level languages which are now described.

3.15.3 High Level Languages

High-level languages use program statements that are more ordinary than those of assembly language. Each high-level language statement may represent many machine code instructions. An interpreter program or a compiler program is used to translate high-level language statements to machine codes, which can be loaded into memory and executed. Programs written in high-level language require more memory, execute more slowly than the same programs written in assembly language program. Programs that involve a lot of hardware control such as robots and factory control system, and must run as quickly as possible are usually written in assembly language program. Complex data processing programs that manipulate massive amounts of data, such as insurance company records are best written in a high-level language (Ramsay, 1981).

In this research work the Z80 assembler language was used. The next section gives an overview of this language and presents some common instructions in the Z80 assembly language.

3.15.4 Some Z80 Instructions

The Z80 has a total of 256 instructions. Some of the commonly used instructions are presented. The LD instruction (load) is used to transfer data between registers and to and from memory. When used with the accumulator, the data in the A register (the accumulator) remains unchanged. The other registers are B, C, D, H and L. Some registers can be grouped together and treated as though these pairs were one. Thus H and L can be grouped to become HL register pair. Other pairs used are BC, DE and HL. We can transfer data from one double register pair to another. The memory location is usually given by the HL register pair and shown as (HL). Data can only be transferred to and from the A register. A second way to transfer data to and from memory is to use a number instead of a register pair. A third way is to use the IX and IY registers, which transfers contents in successive memory locations.

Z80 also has prime registers, which are designated A', B', C', D', E', H' and L'. These cannot be accessed directly, but they can be swapped with the ordinary registers. The instruction EXX swaps BC DE and HL with the prime registers and vice versa. Several other commands exist that swap registers. For instance, EX AF, AF' swaps the A register and the flags with the corresponding prime registers. The SP register (stack pointer) is used as a pointer. Because memory locations are used to store data you have to set where you want the stack to be before using PUSH and POP. Most programs would use a maximum of 20 PUSHes before POPs and thus about 40 memory locations below the initial value of the SP are not used. The ORG instruction is used to reserve this memory. The PUSH and POP instruction both work on double register pairs only (Bolton, 2000).

Registers can be added or subtracted, but multiplication and division require a short program. Other instructions exist to compare two numbers and increment or decrement registers. When two numbers are added or subtracted there needs to be a way of signaling whether a carry or borrow has occurred. In addition to a carry five other features of the operation, such as equal to, zero and negative or positive are signaled. This is done using the flags.

The C or carry flag is set if the result of an ADD (addition) is too great for the register to hold the value. In subtraction it is set if the answer is less than 0, necessitating a borrow. The Z or zero flag, is set to 1 if the result is zero, and reset to 0 if the result is not zero. The P or parity flag is set when the parity is odd, and reset when it is even. It is also used by some instructions to signify overflow, and thus is sometimes written as the P/V flag. The S or sign flag, is set to 1 if the result of an operation is between 0 and 127, and reset to 0 if between 128 and 255. The H or half carry flag is used when converting hexadecimal values to BCD. The N flag indicates whether the last instruction was an ADD or SUB.

The 8-bit instructions all add either a number, or another register to the A register. The flags are set on the result contained in the A register. If we want to add numbers that are more than can be stored in the A register (>256), then the HL, IX or IY registers can be used. The set of add with carry instructions (ADC) are essentially the same as the ADD set, but add the carry flag as well. The ADC instruction allows multiple precision adding. The least most significant bytes are added, and the carry is propagated to the next most significant bytes by using ADC.

There are two subtraction instructions, SUB which is the opposite of ADD, and SBC which is subtract with borrow. SUB is used for 8 bit subtractions. The number or register is subtracted from the A register and the result stored in the A register. The SBC group is the opposite of the ADC group. The register or number is subtracted from the A register, along with the C flag, and the result stored in the A register. The CP instruction (compare), can be thought of as an 8 bit SUB instruction in the way the flags are changed, except that the contents of the A register remain unchanged.

The INC instruction (increment) is the same as adding 1. The results of all increments to single registers affect the flags as if an ADD had been performed. The DEC instruction (decrement) is the opposite of the increment group. It subtracts 1 from the register in question. As before, single register decrements affect the flags as a subtract would, but double register decrements leave the flags as they are. The JP instruction (jump) causes the program to move to the specified location. This instruction can be conditional or unconditional. The following are the conditional jump instructions: NZ - if the zero flag is not set, Z - if the zero flag is set, NC - if the carry flag is not set and C - if the carry flag is set. If the conditions are not met the program simply moves to the next instruction.

Subroutines are called by a CALL and terminated by a RET instruction. The return location is stored on the stack. CALL's and RET's can be conditional. There are two additional returns, RETI which returns from interrupts and RETN from non maskable interrupts. Since interrupts are not implemented on the interpreter these two instructions do nothing. They can be included if they will be used in the final compiled program.

Logic AND, OR and XOR instructions all work on the A register and perform bit by bit comparisons with the appropriate register according to Boolean algebra rules. The flags are also set by the result in the A register. The set and reset instructions allow a specific bit in a register or memory location to be set to 1, reset to 0 or tested to see if it is 1 or 0. The RES instruction is the opposite of the SET instruction; it changes the appropriate bit to 0. Flags are not affected by either the SET or RES instructions. The BIT instruction is used to test whether a specific bit is a zero or a one.

Other instructions rotate and shift contents of registers left or right. NOP is the simplest Z80 instruction, and simply does nothing. NOP's are useful for creating delays. The Z80 performs NOP's to ensure proper memory refresh. HALT instruction halts the CPU. DI disables the interrupts. EI enables the interrupts. IM instruction sets the interrupt mode. NEG negates the A register. CPL complements the A register. All 0's become 1's and all 1's 0's.

Apart from the instructions that generate code, there are additional commands which are needed to test and run the programs. For instance, the semicolon is used to indicate a comment. These are ignored by the program. END is used to indicate the end of a program. It should always be the last statement in a program; otherwise an error message will appear. EQU is used to set labels to particular values. This enables the way the program works to be made a lot clearer. Labels should be assigned values before they are encountered in the program. The value of a label cannot be changed once it has been assigned. ORG is used to set the origin of the program when it is compiled. When the program is finally dumped into a Z80 computer's memory, some way of indicating where the program will start is needed. In the Z80 it is customary to start the program not in memory location

0 but at memory location 100H. The first instruction in the program is a jump to location 100H. This is because the interrupts jump to locations between 5H and 100H (Moxham, 1996). A detailed list of opcodes for the Z80 is shown in appendix 4.

3.16 Circuit Boards

3.16.1 Introduction

Circuit boards are devices onto which electronic circuits are assembled. They provide facilities for attaching components such as ICs or resistors in a fixed orientation or in an optimum manner. There are two basic types of circuit boards: The printed circuit board and the breadboard. The control system in this research work was initially assembled on a breadboard. Once it was confirmed to be working it was etched onto a printed circuit board. The next sections will discuss these two circuit boards and offer some limitations and advantages of each.

3.16.2 Printed Circuit Boards

A printed circuit board (PCB) is used to mechanically support and electrically connect electronic components using conductive pathways, or traces, etched from copper sheets laminated onto a non-conductive substrate. Most are composed of between one and sixteen conductive layers separated and supported by layers of insulating material (substrates) laminated together. Layers may be connected together through drilled holes called vias, which are electroplated, or small rivets may be inserted.

PCB substrates frequently are made of paper impregnated with phenolic resin or a woven fiberglass mat impregnated with a flame resistant epoxy resin. PCBs for high power radio frequency

work use plastics with low dielectric constant (permittivity) and dissipation factor such as polyimide, polystyrene and cross-linked polystyrene. PCBs designed for use in vacuum or in zero gravity, as in spacecraft, being unable to rely on convection cooling, often have thick copper or aluminum cores to dissipate heat from electrical components.

Some PCBs are designed to be very or slightly flexible to save space. For instance, PCBs inside cameras and hearing aids are almost always made of flex circuits so they can be folded up to fit into the limited available space. Power electronic applications require low-thermal resistivity substrates, with thick copper track to carry high currents.

Most PCBs are made by adhering a layer of copper over the entire substrate, sometimes on both sides then removing unwanted copper after applying a temporary mask (e.g. by etching in ferric chloride), leaving only the desired copper traces. Holes, or vias, through a PCB are typically drilled with tiny drill bits made of solid tungsten carbide. When very small vias are required the vias may be evaporated by lasers. The walls of the holes, for boards with 2 or more layers, are plated with copper to form plated-through holes that electrically connect the conducting layers of the PCB.
The pads and leads to which components will be mounted are typically plated, because the bare copper is not readily solderable. Traditionally, any exposed copper was plated with solder, a tin-lead alloy. New solder compounds are now used to achieve compliance with the directive in the EU (European Union) which restricts the use of lead. Areas that should not be soldered to may be covered with a polymer solder resist coating which prevents short circuits between nearby component leads (http://en.wikipeadia.org/wiki/Printed Circuit board, 2006).

Printed circuit boards require soldering. A poorly soldered joint can greatly affect small current flow in circuits and can cause equipment failure. The board or a component can be damaged with too much heat. Sloppy soldering can cause bridges between two adjacent foils preventing the circuit from functioning.

Solder is a fusible alloy composed of tin and lead. Solder has a melting temperature around 360 ^{o}C to 370 ^{o}C, making it ideal for forming a metallic joint between two metals. Solder is identified by the ratio of tin-to-lead. The most common ratios are 40/60, 50/50 and 60/40. Solder with a greater tin content melts at a lower temperature, takes less time to harden, and generally makes it easier to do a good soldering job. The ratio of tin is a main factor in the strength of the solder joint. Solder with a greater tin content has a greater holding ability under stress. Solder with a tin ratio of 60% is the strongest, while solder with less than 30% would be undesirable (www.electronickits.com, 2006).

Most solder contains flux in the hollow core of the solder allowing it to be applied automatically when you heat the solder. The flux will remove any oxide film on the metals soldered creating a good metal-to-metal contact. There are three types of solder fluxes: chloride, organic and rosin. In the electronics industry, only the rosin type is used. Rosin flux comes in two types, pure and active. The most reliable is the pure type, since it does not cause dendrites between tracks on the PC board as the active type does. Due to the highly corrosive and moisture attracting characteristics of the chloride and organic type fluxes, they should not be used in electronics.

A number of different types of soldering devices: irons, guns and stations are available today. Irons are used for light to medium work and guns are for medium to heavy-duty work. The station type can range from light to heavy-duty. For PCB boards, a soldering iron is ideal. Iron sizes vary from 15 to over 500 watts. For working on PCB boards, irons ranging from 15 to 40 watts are suitable. An iron with a higher wattage rating higher than 40 watt, may damage the copper tracks on the PCB board.

Most tips for the soldering iron are made of copper. To increase their lifetime, tips can be coated with iron, but this decreases the heat transfer rate. The tip should be tinned by lightly coating it with solder to prevent it from oxidation. The tip becomes pitted (black spots) from normal use. These spots can be removed by scraping them with a knife or filing them and re-tinning the tip. The tip should be cleaned by wiping it with a wet rag or sponge as this makes soldering much easier. Tips are manufactured in a variety of shapes. Proper solder adhesion requires that the metal surface be free of dirt and grease. The flux only removes the oxides so a brush or rag can be used to clean metal. There are contact cleaners in aerosol cans and other solvents available.

Soldering is done from the copper foil side only. The components are placed as a close to the board as possible and the leads bent to hold the parts in place. The soldering iron tip is pushed against both the lead and the circuit board foil and a small amount of solder applied to the tip. This allows the heat to leave the iron and flow into the foil. The solder is immediately applied to the opposite side of the connection. The solder melts and after flowing around the connection the solder and iron are removed without allowing the component to move. After cooling, the connection should be bright, shiny and smooth. Sometimes desoldering is necessary when correcting a mistake. The metal foil

can be easily pulled up or broken from excessive heat. The least amount of heat as possible should be used.

3.16.3 Breadboards

A breadboard is a reusable solderless device used to build a prototype of an electronic circuit and for experimenting with circuit designs. This is in contrast to PCBs which are used to build more permanent prototypes and cannot easily be reused. A modern solderless breadboard consists of a perforated block of plastic with numerous tin plated phosphor bronze spring clips under the perforations. ICs in DIL packages can be inserted to straddle the centerline of the block. Interconnecting wires and discrete component leads can be inserted into the remaining free holes to complete the circuit topology. Using these it is possible to prototype a variety of electronic systems. However, due to large stray capacitance, solderless breadboards are limited to operating at relatively low frequencies, say less than 10 MHz, depending on the nature of the circuit.

Prototyping with breadboards is too clumsy and unreliable for complicated systems, such as modern computers comprising millions of transistors, diodes and resistors. Modern circuit designs are generally developed using a schematic capture and simulation system, and tested in simulation before the first prototype circuits are built on a printed circuit board. Integrated circuit designs are a more extreme version of the same process. However, breadboard prototyping techniques are still used for some specialized applications such as broadband RF circuits, or where software models of components are inexact or incomplete (http://en.wikipeadia.org/wiki/Breadboard, 2006). The next chapter presents the system design where the hardware and software designs are discussed.

Chapter 4

DESIGN OF THE GREENHOUSE CONTROLLER

4.1 Introduction

In this chapter, the greenhouse control system is designed and implemented. The design begins with the control algorithm, followed by the hardware and then the software design. Next the greenhouse structure is designed. The hardware and software are then integrated, the controlling devices arranged in the greenhouse and then the system is tested. The next section gives an overview of the greenhouse system design.

4.2 Greenhouse System Design

This research work aimed at designing a greenhouse where temperature, humidity and light level were to be under automatic control continuously. To achieve this required an enclosed structure into which sensors and controlling devices for these parameters were placed. To achieve automation on a continuous basis, a microcomputer system was to be used.

The sensors used give analogue output signals that are not compatible with microprocessors. They thus needed not only signal conditioning but also conversion to digital. The system would thus include signal conditioners and ADCs. The controlling devices and the display would also require an interface to be able to communicate with the microprocessor. The design for this system is shown in figure 4.1. All the sensors for temperature, humidity and light level are signal conditioned. The humidity and temperature sensors are interfaced to ADCs. The diagram also shows the actuator interfaced to the microcomputer. Also interfaced are the displays for each of the three parameters.

After setting out this algorithm, the next sections will now consider each part of the design in detail, beginning with the hardware design

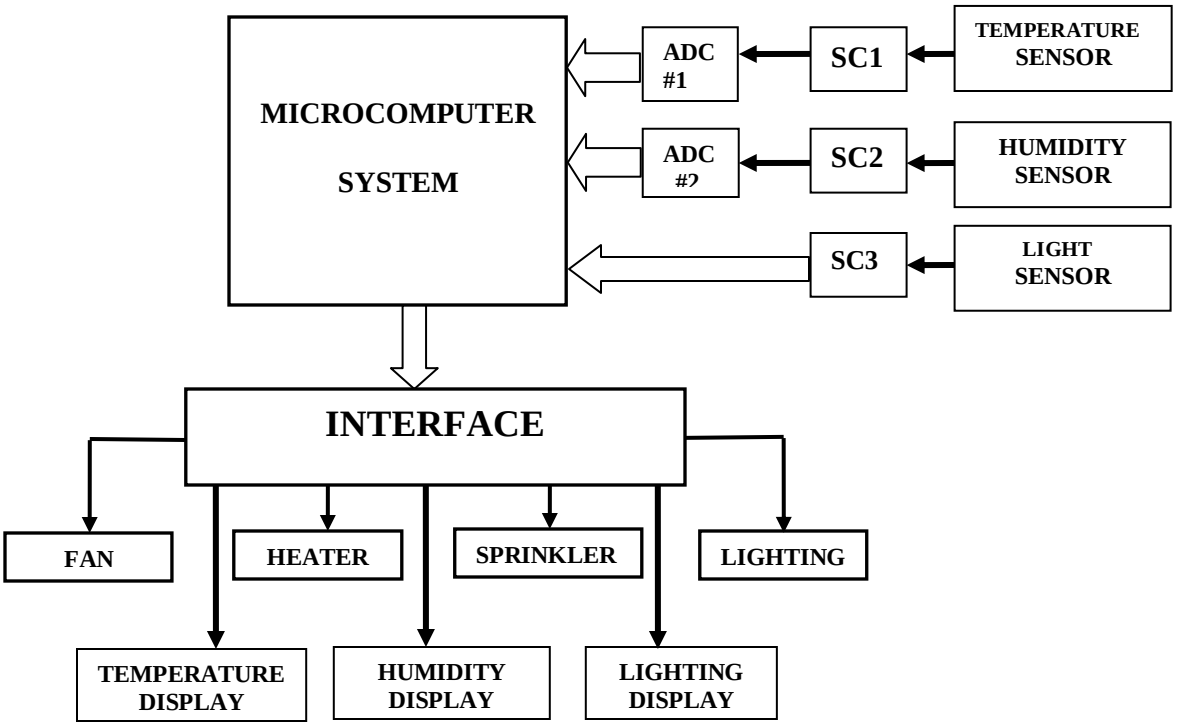


Figure 4.1: Design of the complete greenhouse control system showing microcomputer system and interfacing to the sensors, the controlling devices and display system.

4.3 The Hardware Design

In this thesis, the microcomputer system shown in figure 4.2 was used. It consists of a Z80 CPU, clock circuitry, reset circuit, a 74LS139 address decoder, 6116 RAM, 2716 EPROM and two 8255 PIAs.

The hardware components were selected to offer compatibility in logic family, operating temperature, frequency response and current voltage ratings. All the components used operated with

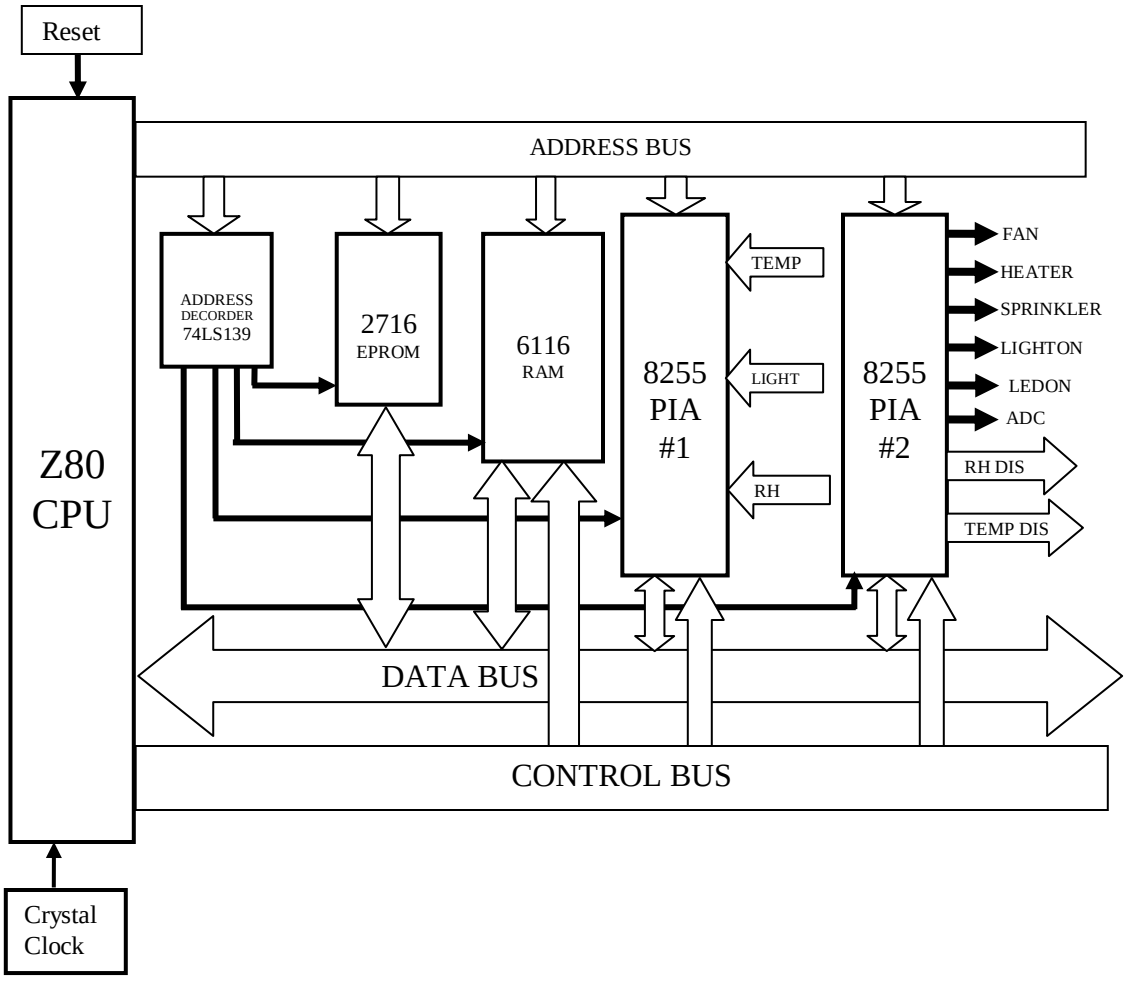


Figure 4.2: Block diagram of the Z80 microcomputer system showing the busses, the ROM and RAM, address decoder and the PIAs

a single power source of +5 V. The clock used had a frequency of 1.687 MHz. The diagram shows the 3 buses. The data bus is bidirectional, while the address and control busses are unidirectional. The next section discusses the circuit design

4.4 Circuit Design

The circuit was designed on breadboards, which provide facilities for attaching components such as ICs or resistors. After the circuit was confirmed to be correctly wired, schematics of the circuit were

made using the EASYPC software. Two schematics were made, one of the control unit and the other for the display unit. The schematics were then used to etch the circuits on two PCBs. The complete circuit diagram for the greenhouse control system is shown in appendix 5. The next section will now consider the design of the system software.

4.5 The System Software Design

To develop the software, the requirements of the system were clearly defined. This included defining segment registers to be used, the memory map, port addresses, input/output devices, the subroutines and finally writing the assembly language for the Z80 CPU and coding in Z80 assembler. Figure 4.3 shows the modularized structure of the controlling system while figure 4.4 shows the flowchart for the system software design.

A brief description of each subroutine is now given:

MAIN:	This program calls different subroutines for execution.
EQUATES:	This program initialises and code the ports, stack pointer and other needed interrupts.
LIGHT:	This subroutine detects the light level, displays it, and if the level is insufficient it calls LIGHTON subroutine.
LIGHTON:	This subroutine controls the lighting system.
TEMP:	This subroutine monitors and displays the temperature, and if the temperature rises above a predetermined value, it calls FAN subroutine. If the temperature falls below a predetermined value it calls HEATER subroutine
FAN:	This subroutine controls the fan.

HEATER: This subroutine controls the heater.

RELHUM: This subroutine monitors relative humidity, displays it, and if it falls below a predetermined level it calls SPRINKLER subroutine.

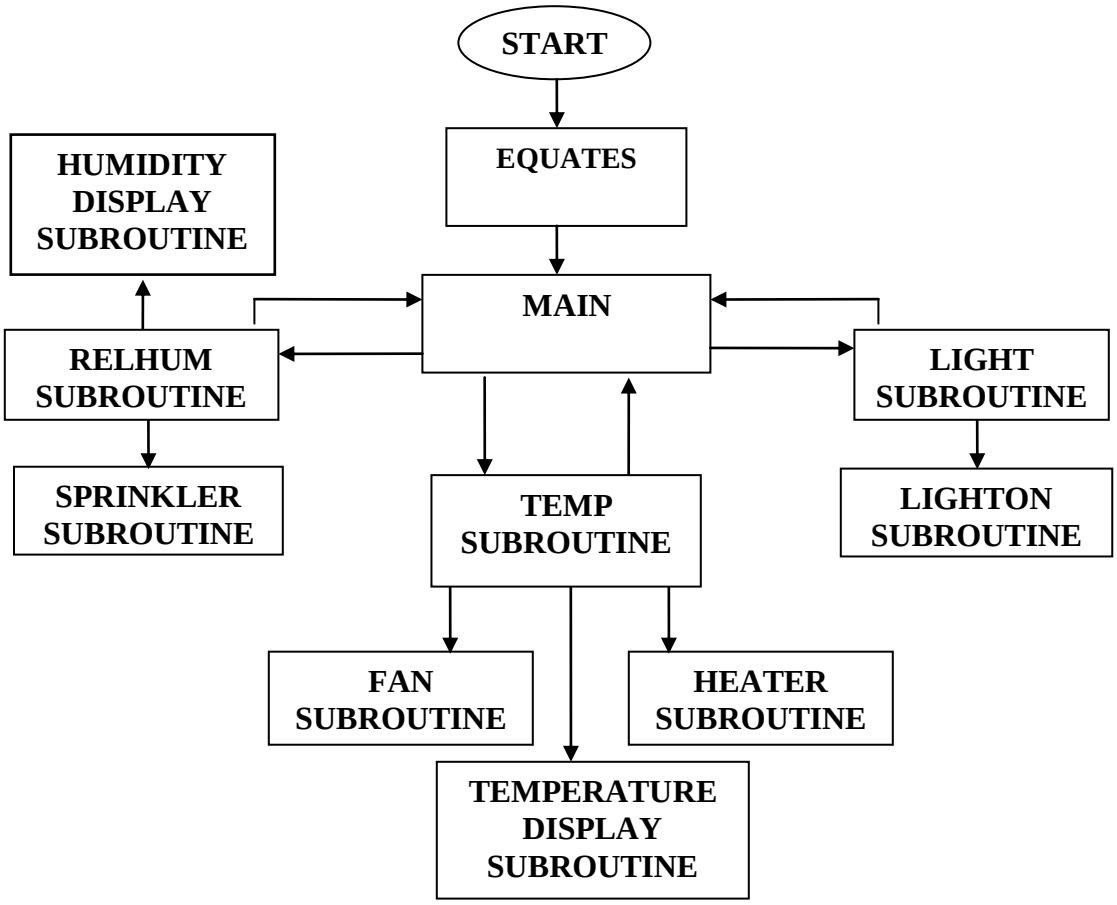


Figure 4.3: Modularized structure of the software design showing the main, the equates for defining variables and the subroutines for reading the sensors, displaying the variables and controlling the devices

SPRINKLER: This subroutine controls the water sprinkler.

DELAY: This subroutine allows time to read the displays.

ADC: This subroutine initiates conversion of the analogue signal and calls DISPLAY

DISPLAY: This subroutine displays the BCD value of the signal.

After the program had been written it was then burnt into the EPROM. To program the EPROM the Z80 compiler program was used. This program has 3 files all in DOS: Command, Ed and Xasm. The files are copied onto a floppy diskette, which is then opened in DOS. After the A directory is opened, the file "edit test.src" is opened. This opens a blue window in a DOS environment where the program is entered line by line via the keyboard using Z80 mnemonics. On completion, it is saved and exited back to DOS. The program is now complied by typing "xasm test.src" pressing enter 3 times. The compiled program appears. If errors are revealed, the procedure to access the file "edit test.src" is repeated and the errors are debugged. Once there are no errors "xasm test.src" is typed and enter pressed. Next to the box [test.bin] we type "test.lsp" and next to the box [con.lst] we type "test.bin" and then press enter. This compiles the program. Three files are created on the floppy disk: test.src contains the program; test.bin contains the compiled program while test.lsp contains the program in machine language. The files can be viewed by typing "edit" followed by the file name. The file test.lsp is burnt into the EPROM by the universal programmer interfaced to a computer. The final complete program that was used is shown in appendix 3.

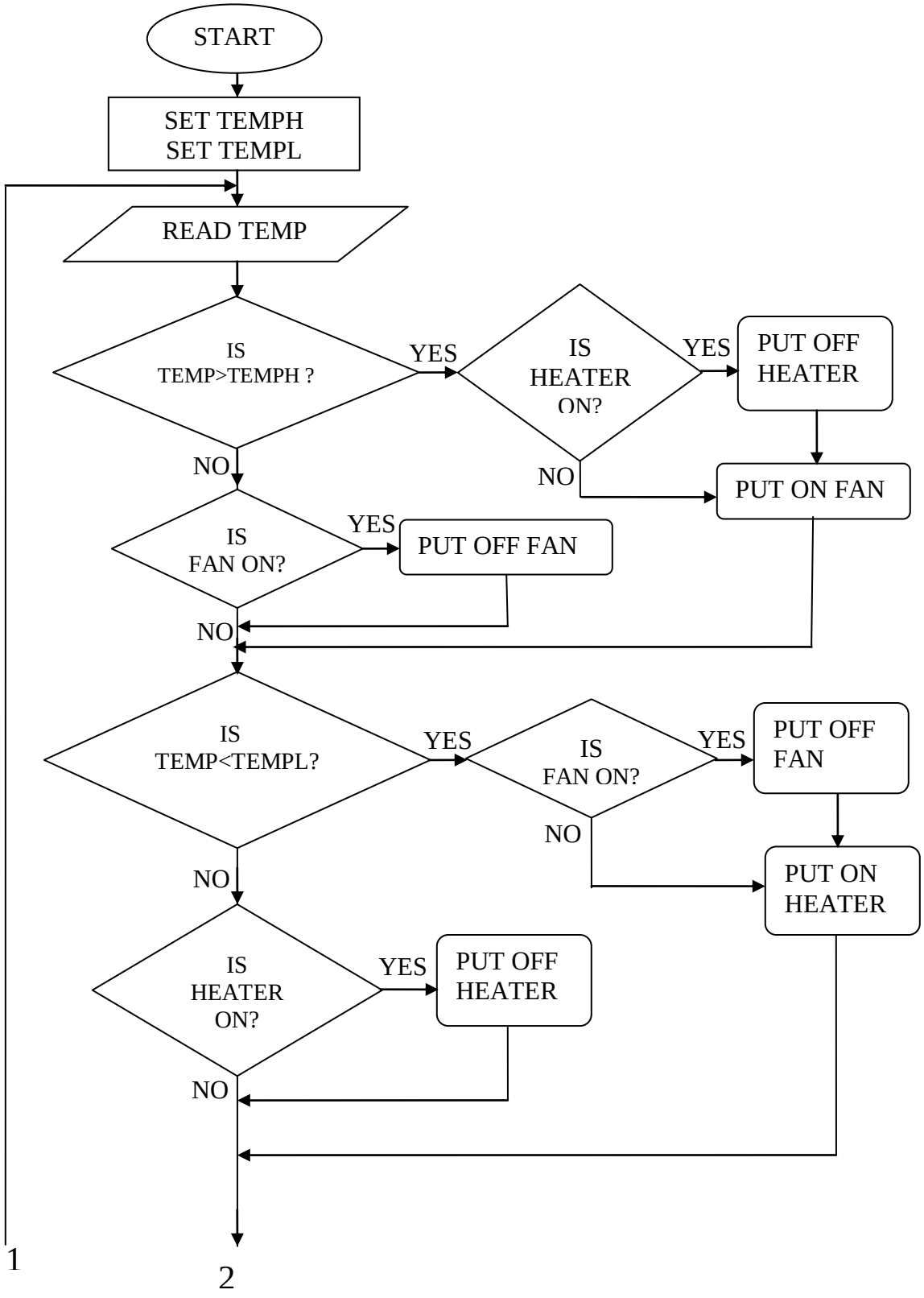


Figure 4.4 *(Contd)*

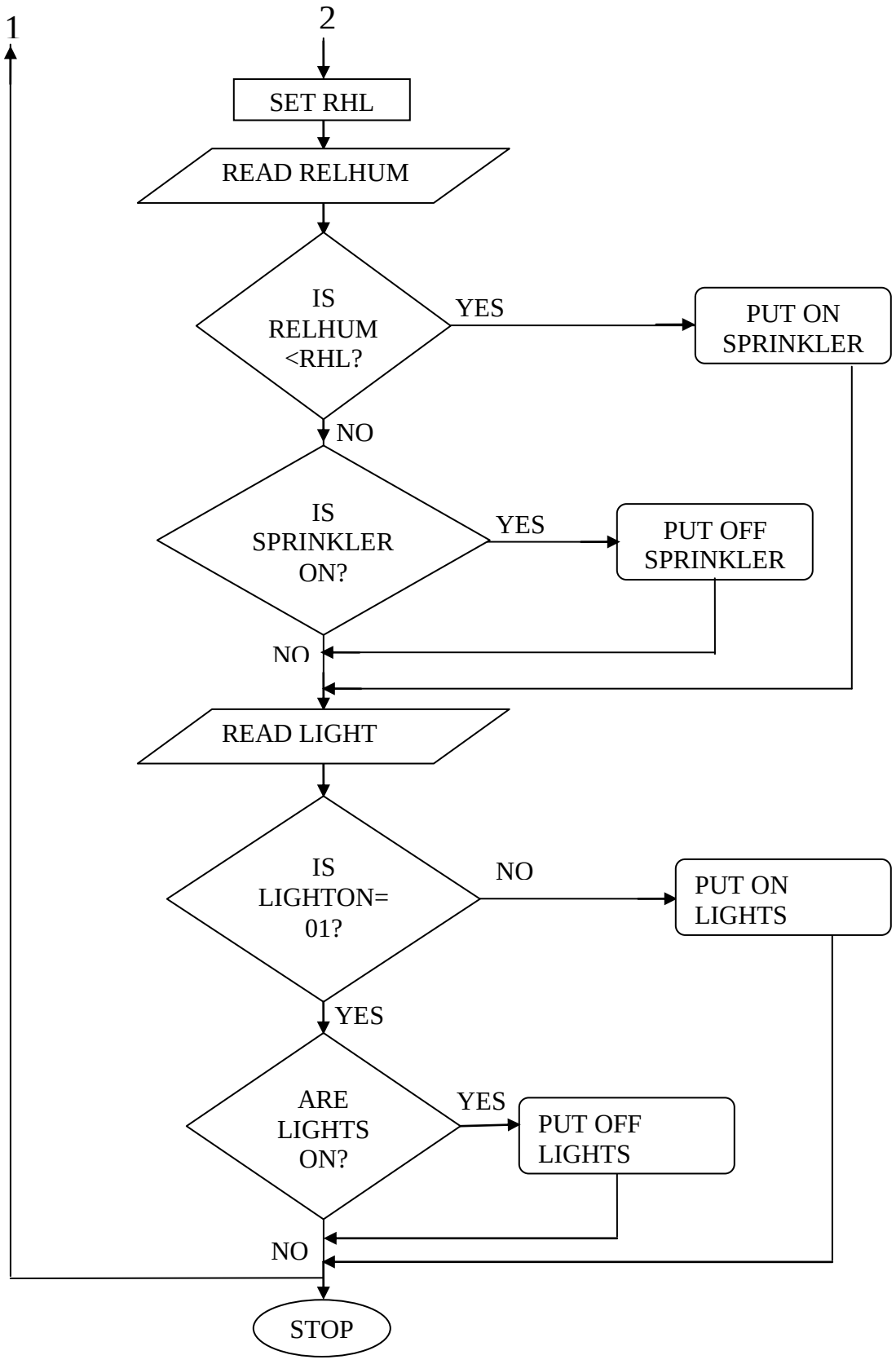


Figure 4.4: Flowchart for system software design

4.5.1 Device Selection

The selection of RAM and ROM is done by the software. The decoder is enabled by a signal sent by the MREQ of the Z80 CPU. The A_{11} and A_{12} address lines are used to select the appropriate memory. Table 4.1 shows how these lines are used in the selection of RAM and ROM

Enable		Outputs				Memory
A_{12}	A_{11}	F_0	F_1	F_2	F_3	
0	0	0	1	1	1	ROM
0	1	1	0	1	1	RAM
1	0	1	1	0	1	not used
1	1	1	1	1	0	not used

Table 4.1: Address decoding for RAM and ROM memory. Outputs F_2 and F_3 would be used if additional memory was required. In this research they were not used.

Enable		Outputs				Device
A_3	A_2	F_0	F_1	F_2	F_3	
0	0	0	1	1	1	PIA#1
0	1	1	0	1	1	PIA#2
1	0	1	1	0	1	not used
1	1	1	1	1	0	not used

Table 4.2: Address decoding for PIA's. Outputs F_2 and F_3 would be used if additional I/O devices were required. In this research they were not used.

To select the PIAs, the IOREQ pin is used to enable the decoder which then uses address lines A_2 and A_3 to select the appropriate device according to table 4.2.

4.5.2 Configuring the PIAs

The control system uses two 8255 PIAs. These PIAs are input/output memory mapped. The control word port configuration and port address is shown in the table 4.3. The first PIA is configured to have all the 24 I/O lines as inputs while the second is configured to have all its 24 I/O lines as outputs. This is done by loading the accumulator with the appropriate control word and then outputting it to the PIA. This configures the ports of the particular PIA. The control words are shown in table 4.4.

8255 PIA	MSB	MODE SELECTION		PORT A	PORT C (UPPER)	MODE	PORT B	PORT C (LOWER)	HEX
	D7	D6	D5	D4	D3	D2	D1	D0	
#1	1	0	0	1	1	0	1	1	9BH
#2	1	0	0	0	0	0	0	0	80H

Table 4.3: The 8255 PIA control word format. Control word 9BH selects PIA #1 while 80H selects PIA #2 both in mode 0.

To select a PIA we use the address of its command register and then we select a port by using its address as shown in table 4.4

	Address	Port	Configured as	Function
8255 PIA #1	00H	A	input	reads temperature from sensor
	01H	B	input	reads relative humidity from sensor
	02H	C	input	reads light level from sensor and controls ADCs
	03H	Command Register		
8255 PIA #2	04H	A	output	controls actuators in the greenhouse
	05H	B	output	displays temperature on a 7-segment
	06H	C	output	displays relative humidity on a 7-segment
	07H	Command Register		

Table 4.4: Configuring the PIA's. PIA# 1 is configured to have all ports input whle PIA #2 has all ports outputs. The ports are assigned addresses as shown.

A port is selected using the port select inputs A_0 and A_1, connected to the least significant bits of the address bus. Table 4.5 shows the selection of various ports and control register in PIA #1.

A_0	A_1	Communication
0	0	Port A and data bus
0	1	Port B and data bus
1	0	Port C and data bus
1	1	Data bus and control register

Table 4.5: Address selection of ports and control register using address lines A_0 and A_1 connected to PIA 1 and 2.

The output of port A of PIA #2 acts as the control for the devices in the greenhouse. The addresses and the controlled devices are shown in the table 4.6.

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	HEX	DEVICE
0	0	0	0	0	0	0	1	01H	Fan
0	0	0	0	0	0	1	0	02H	Heater
0	0	0	0	0	1	0	0	04H	Sprinkler
0	0	0	0	1	0	0	0	08H	Lighting
0	0	0	1	0	0	0	0	10H	LED

Table 4.6: Addresses and functions of output port A of PIA # 2. The addresses are selected such that only one bit is high to activate the particular device. Although up to 8 bits could be used to control 8 devices, only the first 5 bits were used in the research to control 5 devices.

To read the inputs of the PIA#1 we define PIA#1 by the command register and then we assign addresses to each port as shown in table 4.6. Each time we load the A register with any of these addresses, the contents of that port are read into the register.

4.5.3 Memory Map

Memory map is whereby physical memory address is used to address different logical locations (Downton, 1984). The microprocessor uses 16 address lines to identify an I/O device as if it were a memory register, using the same control signals (Memory Read or Memory Write) and instructions as those of memory sequentially-access or random access. Memory of the Z80 is organized into 8-bit quantities called bytes. Each byte has a unique 16-bit binary address corresponding to its sequential position in memory. The Z80 uses its 16 address lines to directly address up to 65,536

(2^{16}) bytes of memory which are in different combinations of 0's and 1's to locate the RAM, EPROM, and other input/output devices. To address these locations, binary address usually represented in hexadecimal is used. The Z80 program instruction may be one, two or three bytes in length. Multiple byte instructions are stored in successive memory locations with the first address of the first byte being used as the address instructions.

The ROM holds the system's program. When the reset key is pressed, the program counter is set to zero and therefore the EPROM 2716 is mapped to address location 0000H-07FFH. The RAM 6116 provides space for user program and contains the system stack and is mapped to 0800H-0BFFH. Figure 4.5 shows the systems memory map.

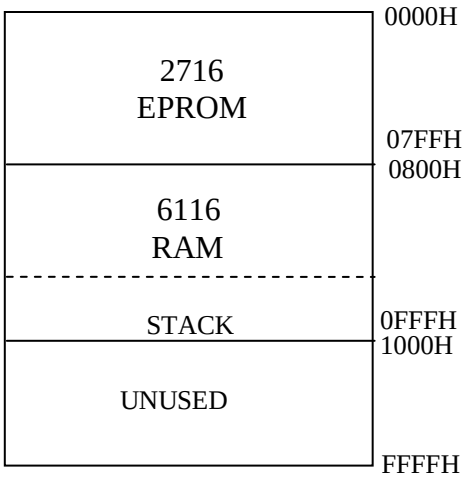


Figure 4.5: Memory map showing the mapping for ROM and RAM. ROM is mapped to 0000-07FFH and RAM 0800-0FFFH. The stack is initialized at 1000H.

Chapter 5

RESULTS AND DISCUSSION

5.1 Introduction

This chapter gives an overview of the assembling, testing and implementation of the system. It begins with the hardware assembly, and then discusses the software design and implementation. Results are then presented, the performance evaluated and a conclusion made. The next section begins by considering the assembling of the hardware.

5.2 Assembling the Hardware

The hardware assembly was done on breadboards in stages. The Z80 CPU was assembled first, then the clock and reset circuitry, followed by the memory. Then the PIAs were added. Finally, the sensors, the ADCs and the display unit were assembled. After testing the system at each stage and debugging, it was then etched onto a PCB. Each of these processes is now discussed, giving a detailed description of the assembling and testing, beginning with the Z80 microcomputer.

5.2.1 Assembling the Z80 Microcomputer

The Z80 CPU was assembled on a breadboard and the busses wired as follows: address lines A_0-A_{10} (pins 8-1, 23, 22 and 19) were connected to the address bus and data lines D_0-D_7 (pins 9-11, 13-17) connected to the data bus. This IC was connected to the power supply via V_{cc} (pin 24) to +5V and ground, GND (pin 12) to 0 V. The clock, CLK (pin 6), was connected to the output of the clock circuit, while the reset, RST (pin 26), was connected to the output of the reset circuit. The address pins A_{13}-A_{15} (pins 3-5), bus acknowledge, BUSACK (pin 23), memory refresh, RFSH (pin 28), halt, HALT (pin 18), and maskable interrupt, MI (pin 27), were all NC (not connected). The wait state,

$\overline{WAIT}$ (pin 24), non-maskable interrupt, $\overline{NMI}$ (pin 17), interrupt, $\overline{INT}$ (pin16), were all connected to V_{cc} to deactivate them since they are all active low.

Then the address decoder 74LS139 was assembled and connected to the Z80 CPU as follows: enable, E_a (pin 1), of the decoder to memory request, MRQ (pin 19), of the CPU, enable, E_b (pin 15), to input/output request, IOREQ (pin 20), input $A1_a$ (pin 3) to address A_{11} (pin 1) and input $A0_a$ (pin 2) to A_{12} (pin 2). This IC was connected to the power supply via V_{cc} (pin 16) to +5 V and ground, GND (pin 7), to 0 V.

The memory ICs where now added and were both connected to the Z80 CPU as follows: address lines A_0-A_{10} (pins 8-1, 23, 22 and 19) were connected to the address bus and data lines D_0-D_7 (pins 9-11, 13-17) connected to the data bus. The output enable, OE (pin 20), pins of both ICs were connected to the read pin, $\overline{RD}$ (pin 21), on the CPU. Both ICs were supplied by power via V_{cc} (pin 24) to +5 V and ground, GND (pin 12), to 0 V. The chip enable pin, CE (pin 18), of the EPROM was connected to the address decoder output $Q1_a$ (pin 5), while that of the RAM to output $Q0_a$. The write enable pin, WE (pin 21), of the RAM was connected to the write pin, WR (pin 22), of the CPU. Pin V_{pp} (pin 21) of the EPROM was NC. It is used only during the programming of the chip.

The reset circuit was tested using an LED. When the push switch was closed and then opened the LED connected to the reset pin of the Z80 indicated a low to high transition. The clock circuit was tested using a CRO. The output gave a square wave whose frequency matched that of the crystal oscillator used.

At this point, the assembled circuit was tested by burning a program into the EPROM and plugging it into the circuit. The procedure for this is discussed later in this work. When the reset switch was pressed and released the data and address bus showed some activity when tested with a logic probe, indicating that the program was running.

5.2.2 Assembling the PIAs

The PIAs were now assembled. The data lines D_0-D_7 (pins 34-27) were connected to the data bus. The address inputs A_1 (pin 8) and A_0 (pin 9), which are used to select the ports, were connected to the address bus of the CPU. The RST (pin 35) pins of the PIAs were connected to the reset pin (pin 26) of the CPU, the WR (pin 36) to WR (pin 22) and the $\overline{RD}$ (pin 5) to $\overline{RD}$ (pin 21) of the CPU. These ICs were connected to the power supply via V_{cc} (pin 26) to +5 V and ground, GND (pin 7), to 0 V. The chip select, CE (pin 6), of PIA #1 was connected to the decoder output $Q1_b$ (pin 11) and that of PIA #2 to $Q0_b$ (pin 12). The PIAs are selected by the decoder using two address lines A_2 and A_3. Each port of the PIAs was connected to a bus. PIA #1 port A, PA_0-PA_7 (pins 4-1, 40-37), was connected to bus PA1, port B, PB_0-PB_7 (pins 18-25), to bus PB1 and port C, PC_0-PC_7 (pins 14-17, 13-10), to bus PC1. PIA #2 port A, PA_0-PA_7 (pins 4-1, 40-37), was connected to bus PA2, port B, PB_0-PB_7 (pins 18-25), to bus PB2 and port C, PC_0-PC_7 (pins 14-17, 13-10), to bus PC2.

A test program was burnt into the EPROM to test the system at this stage. In this program all ports were configured to be outputs. This program outputs alternate 0 and 1 (10101010) to all the ports of the two PIAs one at a time and repeats with the bits reversed (01010101). This program is shown in appendix 1. Using LEDs connected to all the ports the system was confirmed to be working. The next test program involved counting 0 to 9. The program is shown in appendix 2. Using the 7-

segment displays connected to all the ports, the system was confirmed to be able to output the numbers 0-9 to the displays.

5.2.3 Assembling the ADCs

Two ADCs were used, one for temperature and the other for humidity. The ADCs were assembled in the circuit as follows: The data lines D_0-D_7 (pins 18-11) of ADC #1 was connected to the PC1 bus of the PIAs and that of ADC #2 to the PA1 bus. The control lines of both ADCs were connected as follows: chip select, CS (pin 1), read $\overline{RD}$ (pin 2) and write WR (pin 3) to PA_0, PA_1 and PA_2 respectively of bus PA2. The clock pins, CLKR (pin19) and CLK IN (pin 4), were connected to an external circuit to drive the internal clock for these ICs. Voltage reference, Vref/2 (pin 9) was connected to a potential divider circuit to supply the appropriate reference voltage for each ADC. The temperature and humidity sensors were connected to V_{in}+ (pin 6) and V_{in} – (pin 7) of the appropriate ADC. The interrupt pins, INTR (pin 5), of ADC #1 and ADC #2 were connected to PB_0 and PB_1 respectively of bus PB2. These ICs was connected to the power supply via V_{cc} (pin 20) to +5 V and digital ground, DGND (pin 10), to 0 V. Pin 8, analogue ground, AGND was also grounded for both ICs.

The ADCs were tested by taking $\overline{CS}$ and $\overline{RD}$ low and then taking $\overline{RD}$ high followed by $\overline{WR}$ low and checking the output on LEDs connected to the output.

5.2.4 Assembling the Displays

The display assembly consisted of 7–segment display units with 74LS48 IC drivers. Temperature and humidity were to be displayed on two 7-segment units each. Limiting resistors were used for the

7-segment units. The temperature display was connected to bus PB2 of the PIAs while the humidity display was connected to bus PC2. The system status was displayed using a single LED that was to be under microprocessor control. Its input was connected to PA3 of bus PC1 through a 270Ω limiting resistor.

5.2.5 Assembling the Sensors

The LM35 temperature sensor was mounted on a socket assembled on a small portion of a PCB. Its output was connected to V_{in} + of ADC #2. The HIH-4000 humidity sensor was also mounted a socket assembled on a small portion of a PCB. Its output was connected to V_{in} + of ADC #2. Both sensors were powered by a + 5 V supply and were placed inside the greenhouse structure. The LDR sensor was similarly mounted and placed on the roof of the greenhouse structure. Its output was connected to PB2 of bus PB1. The sensors were tested by connecting a digital multimeter to their outputs. The voltage levels observed were as expected. The LM35 sensors gave an output in terms of millivolts and the HIH-4000 in volts. The LDR's output went to 4.7 V when covered and 0 V when exposed to bright light.

5.2.6 Assembling the Actuators

The actuators consisted of a sprinkler (motorcar wiper motor) connected to a water supply and to a pipe with jets, a fan (model JD-9225M12S brushless 12 V 0.20 A used in CPUs), a heater (12 V car bulb in an aluminum foil) and a lighting system (12 V car bulb in a holder). All the actuators were run from a 12 V ac source. These actuators were interfaced to the microprocessor through the opto – isolator circuit shown in figure 5.1.

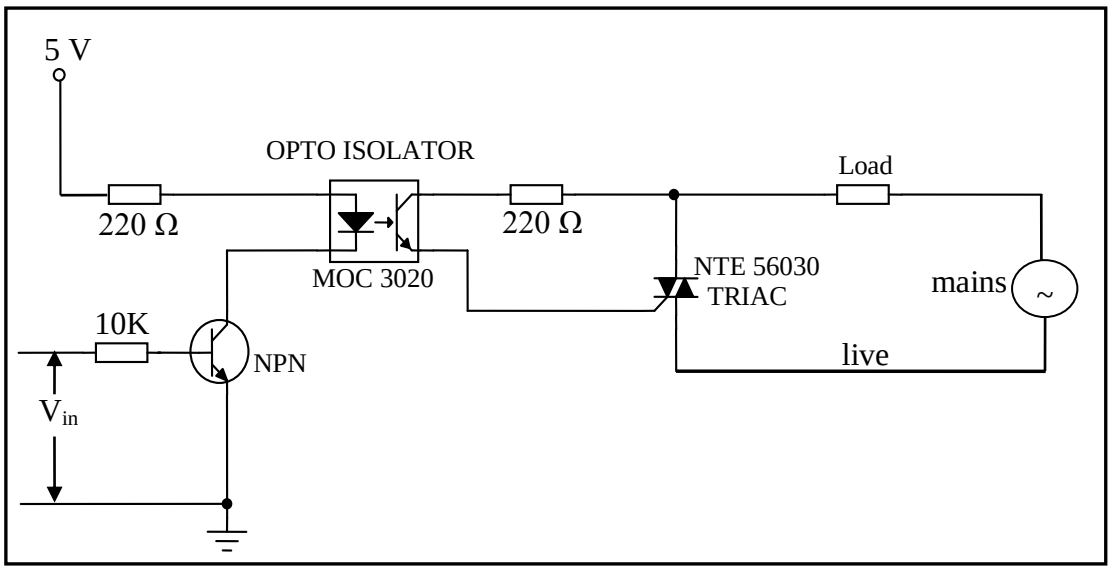


Figure 5.1: Mains isolation circuit showing the opto isolator MOC 3020 and NTE 56030 triac. The opto isolator is send into conduction when the input, Vin, goes high and this triggers the triac into conduction. This circuit can switch loads of up to 1000 V ac with an input of 5 V.

The opto triac circuit was tested by connecting +5 V to the input $V_{in.}$ A 75 W, 240 V bulb connected to the circuit lit. When the input was changed to 0 V, the bulb went off. Four of these circuits were assembled and their inputs connected to bus PB1 as follows: lighting, sprinkler, heater and fan to PA4, PA5, PA6 and PA7 respectively.

5.3 The software

The control program was written in modules. Each module was first tested by simulation using the Z80 Simulatoride software. Debugging was done at this stage. Once the simulations were working, the modules were integrated into the program and split into 3 parts: temperature control, humidity control and light control programs. These programs were burnt into the EPROM one at a time and the system tested. Further debugging was done until the programs were responding as desired. These

programs were now integrated to form the main program which was then burnt into the EPROM and tested in the circuit. The programming of the EPROM will be discussed shortly.

In the system control program, PIA #1 was configured to have all its three ports as input while PIA #2 had all its ports configured as outputs. For PIA #1, port A was interfaced to the temperature sensor, port B to the relative humidity sensor and port C to the light sensor and ADC's INTR pin. For PIA #2, port A was used to control the ADCs, actuators and status LED, port B to display the temperature and port C to display relative humidity. The port A of PIA #2 referred to as the control was connected as follows: bit 1-fan, bit 2-heater, bit 3- water sprinkler, bit 4-lighting system and bit 5-status LED which would indicate the status of the system. The other bits were used to control the ADCs: bit 6- $\overline{CS}$, bit 7- $\overline{WR}$ and bit 8– $\overline{RD}$. The bits used to control actuators were connected to opto isolators which would trigger power triacs to switch the devices.

The systems software design involved setting the higher temperature, TEMPH and lower temperature, TEMPL, and then reading the temperature sensor. If the temperature was greater than TEMPH, the program would check the status of the heater and if on, it would be put off before the fan was activated. If the temperature was less than TEMPL, the status of the fan would be checked and if on it is put off before the heater was activated. The program then sets the lower relative humidity limit, RHL. Then the relative humidity RELHUM is read. If it is less than RHL, the sprinkler would be activated for 2.5s at a time as discussed later. The lighting status, LIGHTON, would be read and compared with bit 1. If the light condition was sufficient, the lighting status would be checked and if on, the lights would be switched off. If the light levels were insufficient, the lights would be switched on. The program would then loop back to the beginning.

The system status LED was set to go off briefly and then on just before reading any sensor. This was to be a visual indication that the system was running. The final program used for the system is shown in appendix 3. The procedure of burning a programming the EPROM is now discussed.

A transparent window on top of the EPROM IC allows the erasure of the bit pattern by exposing the chip to ultraviolet light at 253.7 nm with incident energy of 15 $Ws.cm^2$. Thus with a 12 mW/cm^2 UV tube and the device positioned one inch from it with no intervening filter or glass, the IC will completely erase in about 20 minutes (Anderson, 1984). Once programmed the window is covered with an opaque material. If uncovered it would take some months before program corruption was experienced through the effects of natural sunlight which contains some uv light (Fraser and Milne, 1999).

The EPROM IC has six modes: read, power down, program, program verify, deselect and program inhibit. In the read mode, CE/PGM (pin 18) is low, OE (pin 20) is low, V_{pp} (pin 21) $=V_{cc}=+5$ V. In the program mode, CE/PGM is pulsed down from low to high, OE is high, V_{pp} =+25 V and $V_{cc}=+5$ V. Programming is achieved by applying +25 V to V_{pp} and with the address and data lines stable applying a +5 V pulse to PGM. One pulse is required for each location.

To burn a program into the EPROM, the universal programmer and tester interfaced with a computer is used. The EPROM is inserted into the universal programmer socket ensuring that there is proper contact. The socket is then locked. The “blank process” of the universal programmer software is then activated. A “pass” displayed on the screen indicates the chip is blank and can be programmed. Otherwise a “fail” would necessitate putting the chip back in the UV eraser. The

"process" for burning the program into the chip is now initiated and a "pass" indicates the program has been burnt into the EPROM. The socket is unclipped and then the chip is removed. The window on the chip is then sealed with a light proof material. The EPROM is then transferred to the circuit.

5.4 Results

The system was assembled, programmed and tested. It was allowed to run and readings of various parameters taken and compared with those of standard meters. The next section discusses the results of each parameter starting with temperature.

5.4.1 Temperature

The LM35 temperature sensor was wired in a circuit as shown in figure 5.2. The sensor has an output of 10 mV/^{o}C. The output was checked with a digital voltmeter and found to be in the range of mV. When this output was displayed, it was found to be lower by 5 degrees than the temperature of the digital thermocouple thermometer that we used as our standard. To calibrate the sensor we added a constant 5 degrees to the output of the sensor through software.

To test our systems thermometer, the junction of the thermocouple connected to a digital meter with a sensitivity of 0.1 ^{o}C was placed next to the sensor. A 100 W tungsten filament lamp with a reflector was positioned to supply heat to the devices. The lamp was switched on for 10 minutes. The temperature readings on both meters were recorded at intervals of 1 minute for 40 minutes. Figure 5.3 shows the variation of temperature against time for the thermometers, with the error bars for the system sensor shown. The data gave a correlation coefficient of 0.9934, which indicated a close agreement between the digital thermometer and the sensor.

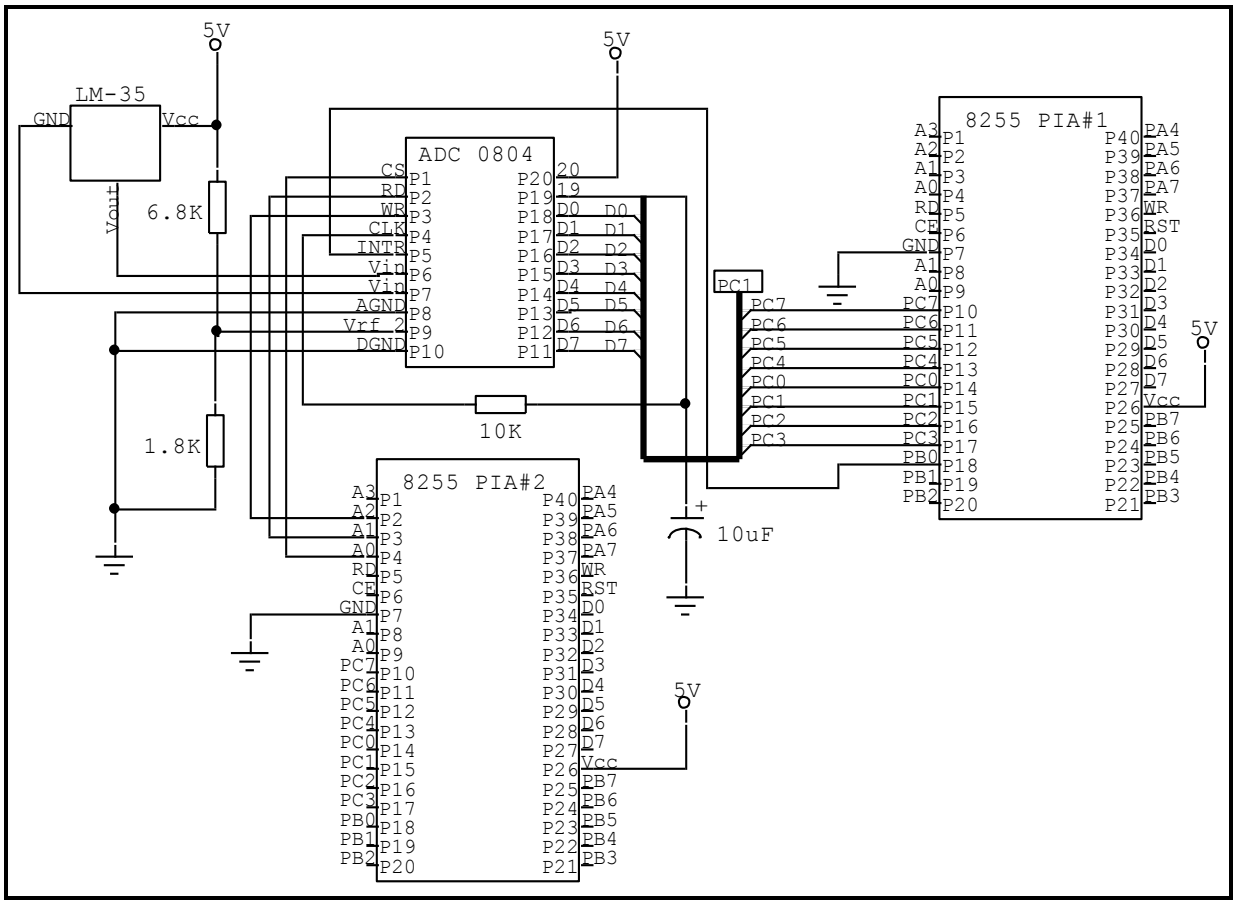


Figure 5.2: Circuit diagram for the LM35 temperature sensor

When the LM35 sensor was fitted in the greenhouse it lost its calibration by 4 ^{o}C. The connecting wires were now longer and this increased the sensors output resistance. We re-calibrated it using software by adding 09 (up from 05 added previously) to the output.

5.4.2 Light

The light sensor circuit was connected as shown in figure 5.4. It was tested by monitoring the voltage levels using a digital multimeter connected across it in the circuit. The voltage varied inversely with the light level the LDR was exposed to. When completely covered the output was in the range of Vcc. When exposed the voltage fell to 0 V for high light levels.

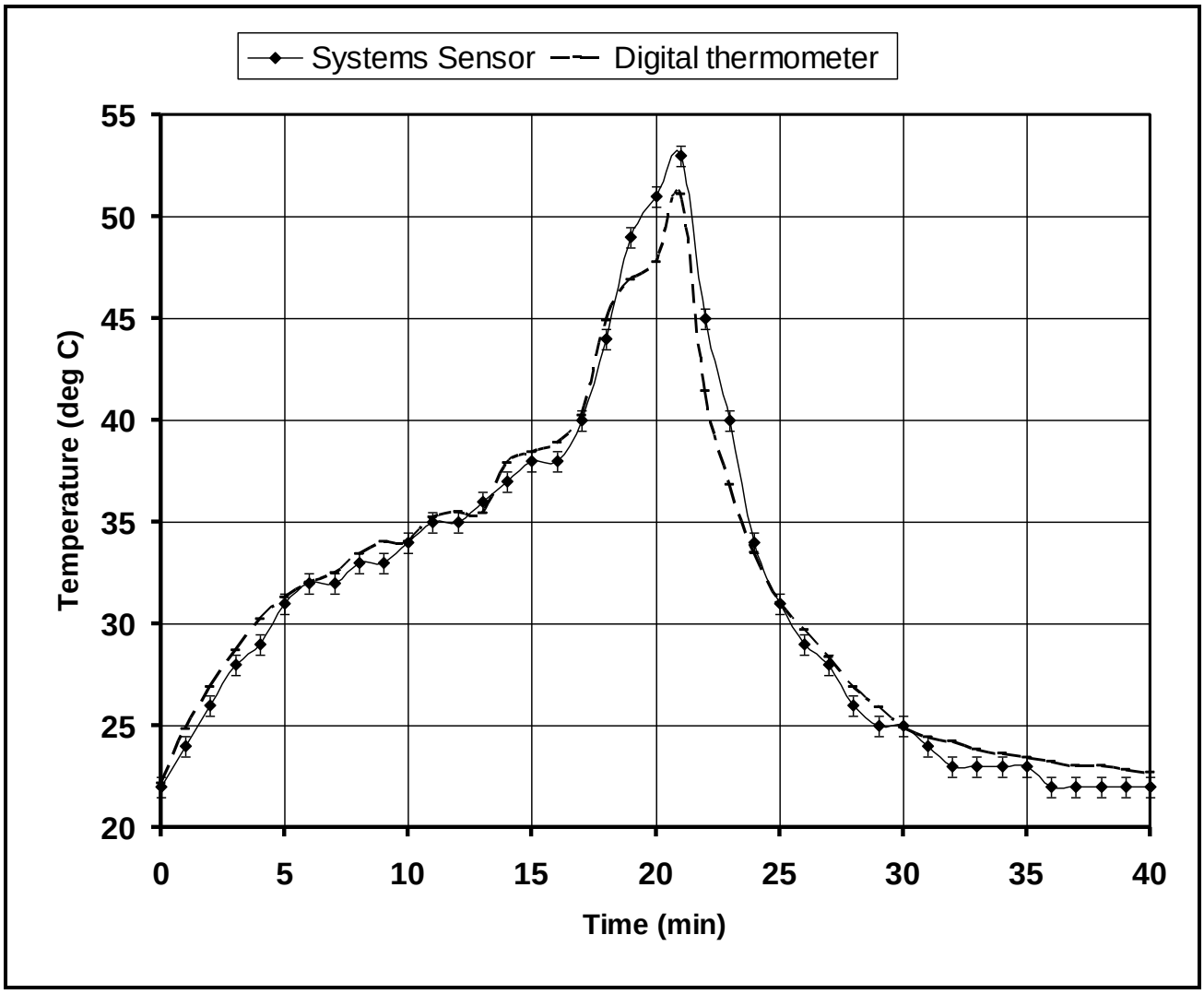


Figure 5.3: Temperature-time graph for digital thermometer and LM 35 temperature sensor. The correlation was fond to be 0.9934 indicating that the system sensor was reliable.

Data for light levels of the LDR was collected over a 24 hour period with the sensor placed on the roof of the laboratory. The sensor switched at 1.52 V, which corresponded to 6.45 pm and 5.55 am. Figure 5.5 shows the variation of light levels in volts against time for the 24 hour period. Time 0 min corresponds to 12.45 pm and 1440 min to 12.45 pm of the following day. The switching point could be adjusted as desired by varying the 6.8 K resistor in series with the LDR. Reducing this resistance would increase the photoperiod.

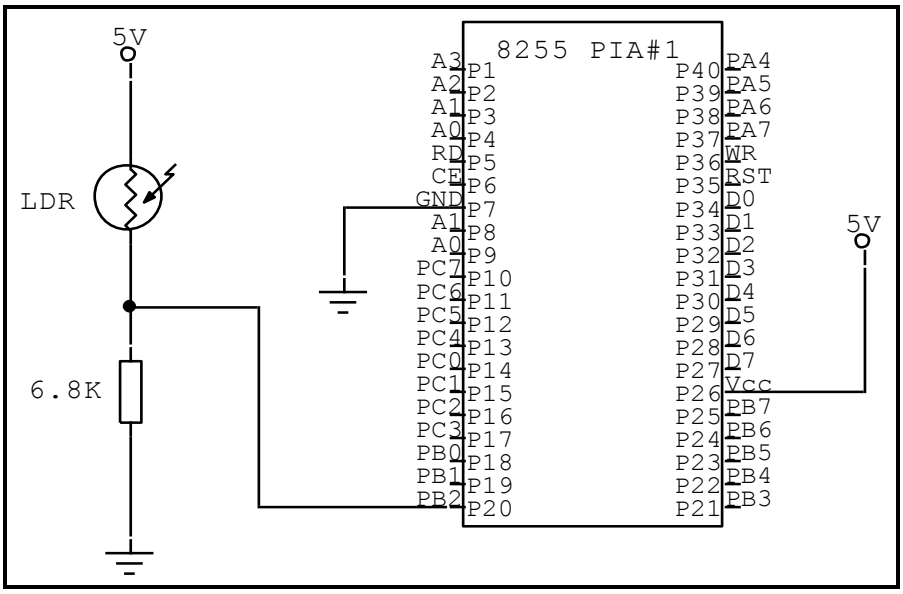


Figure 5.4: Circuit diagram for the LDR light sensor. The potential divider can be adjusted by varying the 6.8K resistor to vary the photoperiod.

To determine the light levels passing through the covering material used on the greenhouse, a sample of the polythene was analyzed using a spectrometer. Figure 5.6 shows the transmittance curve. The average transmittance over the visible spectrum of light is 45%, with a peak of 50% in the 820 nm region. Only about half of the light levels reach the greenhouse. The photoperiod could be adjusted according to the needs of the crop in the greenhouse, or to compensate for the reflectance of the covering material. Supplemental lighting could be used if more light is required, or the covering material could be changed as needed. The graph shows that for polythene, higher wavelengths are transmitted more than lower wavelengths. This could determine the choice of the type of bulb to be used in providing the supplemental lighting. If more of the higher wavelengths were needed, filament bulbs as opposed to, say, fluorescent tubes could be used.

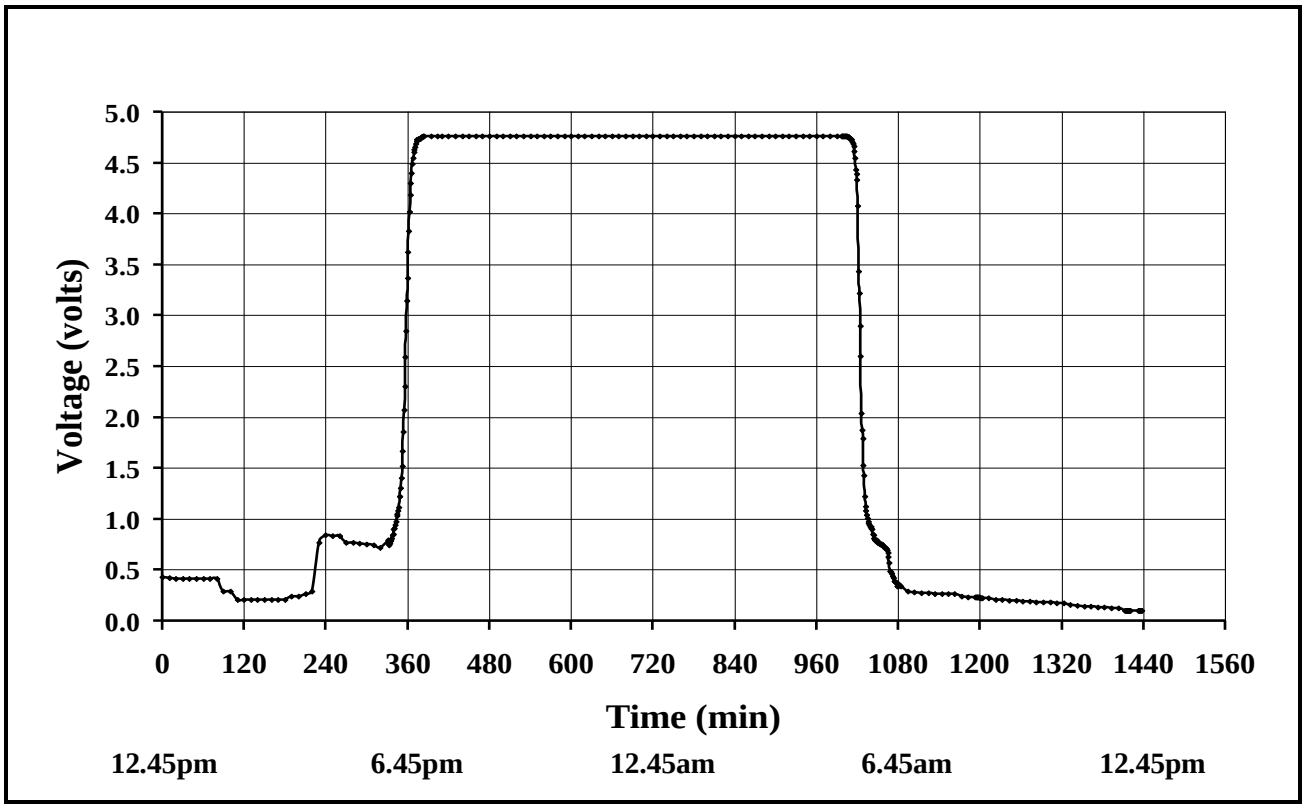


Figure 5.5: Voltage-time variation for LDR light sensor for a 24 hour period. Time 0 min corresponds to 12.45 pm when collection of data started and 1440 min to 12.45 pm the following day.

5.4.3 Relative Humidity

The HIH-4000 humidity sensor was wired in a circuit as shown in figure 5.7. Its reading was compared to that of the hygrometer. It was calibrated by adding a 1.5 MΩ resistor in series with its output and then fine tuned through software. To test the systems humidity sensor, the sensor and a hygrometer were placed inside the greenhouse. Some warm water was introduced into the greenhouse for 5 min and then withdrawn. Data was collected from the two meters for a total of 40 minutes. Figure 5.8 shows the variation of relative humidity against time, with the error bars for the system sensor shown. The data gave a correlation coefficient of 0.7843, which indicated a close agreement between the hygrometer and the sensor.

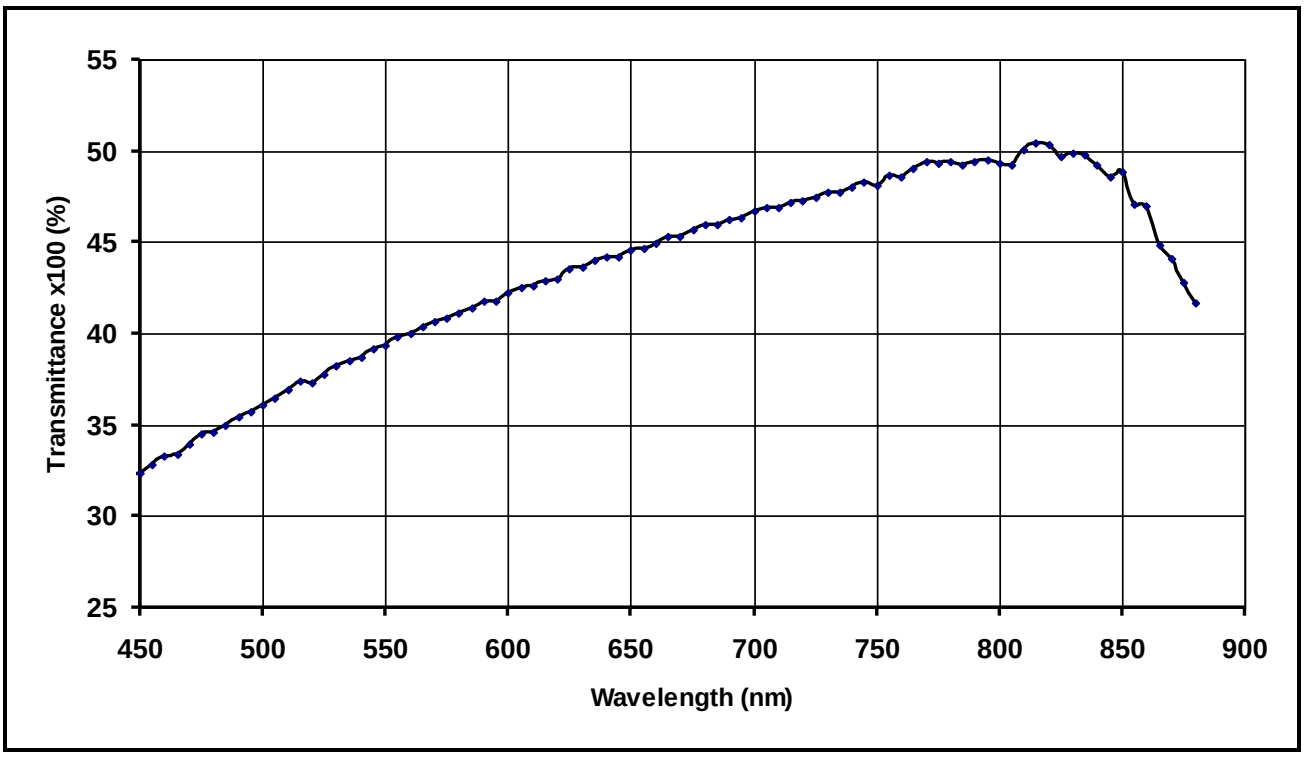


Figure 5.6: Transmittance of the greenhouse covering material (polythene). The average transmittance for the visible range is about 45%.

5.5 Troubleshooting

Each connection on the breadboard was tested for continuity before the IC's were plugged in their sockets. Then the pins were checked for continuity with the sockets. When the circuit was confirmed to be working it was then etched on a PCB.

The crystal used had a frequency of 1.687 MHz. Delay subroutines were used to synchronise the hardware. Each lasted 0.8 s. The HIH-4000 sensor has a response time of 50 s and required a longer delay. A long delay, LDELAY, lasting 50.4 s was introduced and used to initialise the sensor. The total execution time in the first run until all sensors had been read was 1.04 minutes.

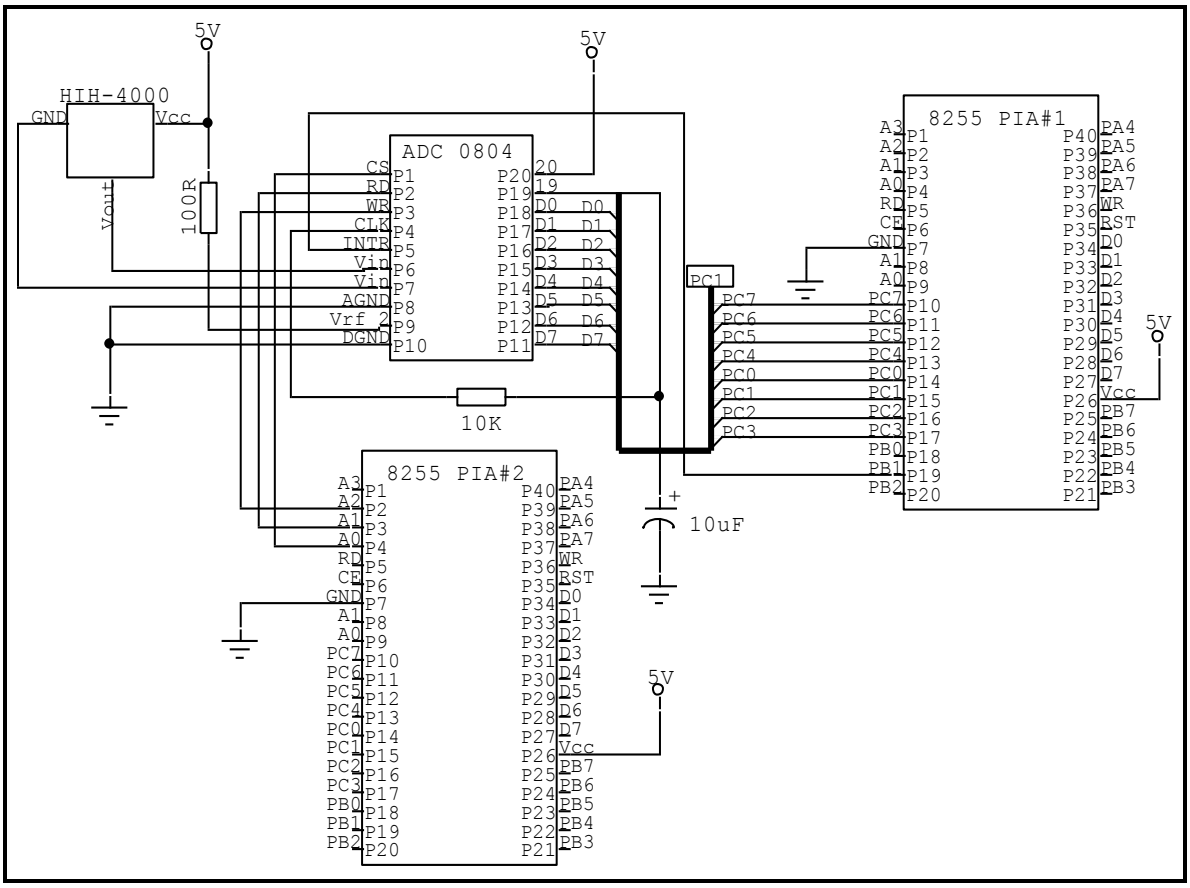


Figure 5.7: Circuit diagram for the HIH-4000 humidity sensor

The system status LED was wired to V_{cc} and to the control line and thus needed 0 V rather than a 5 V to activate to give it. This was adjusted through software. Sometimes the system could hang and it would not be immediately evident. It would be assumed that all conditions are optimum and that program execution was proceeding. To overcome this, a visual display was inserted by enabling the status LED to go off for 2.5 s before proceeding to the next sensor. This was done through a subroutine LEDONOFF.

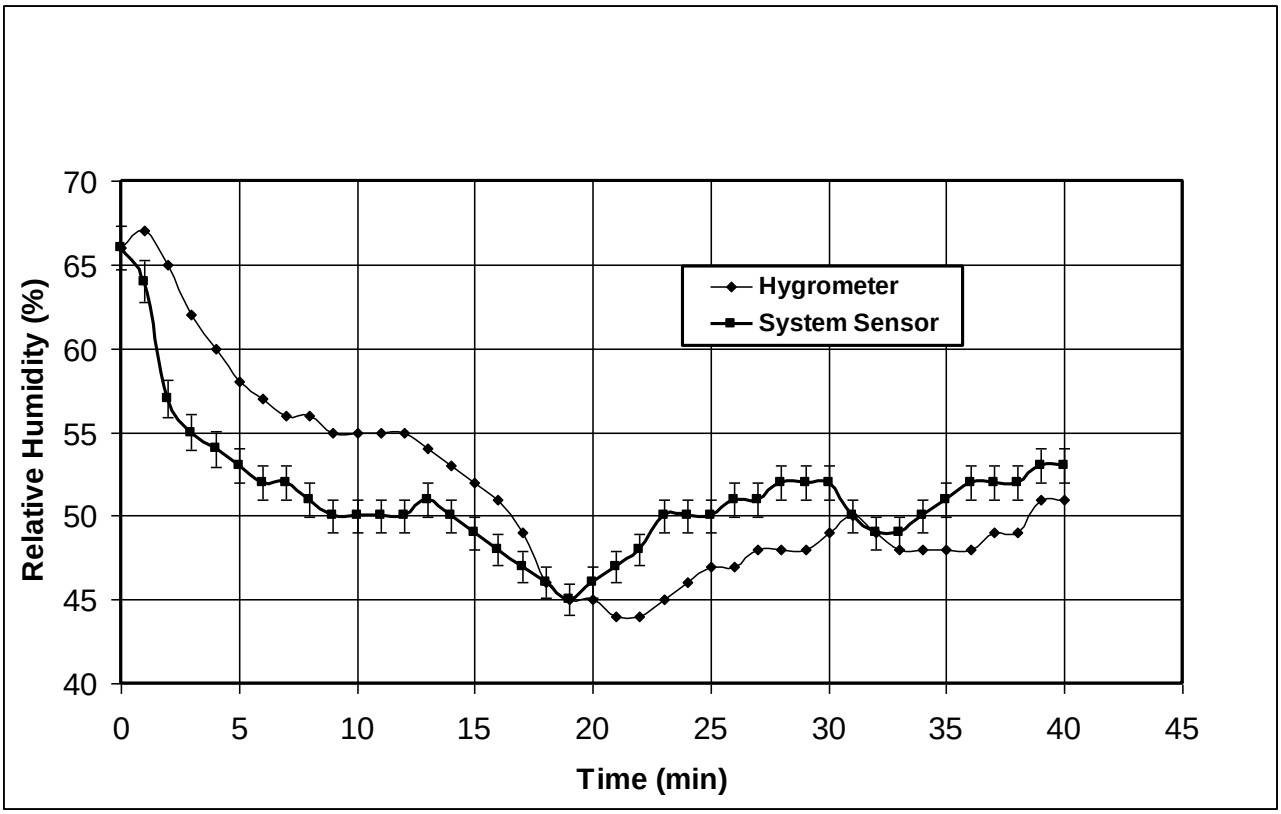


Figure 5.8: Relative humidity-time graph for hygrometer and HIH-4000 humidity sensor The HIH-4000 sensor has an error of 2%. The digital hygrometer has a sensitivity of 1%. The correlation was 0.7843 which indicated that the systems humidity sensor was reliable.

The displays for humidity and temperature were showing spurious values initially due to a hardware oversight in the connection of the 7 segment units. The bits had been connected in the reverse order. Rather than correct this on the PCB which would have involved soldering out all the bits and then rerouting them, a software method was used. A subroutine, TRANSPOSE, was used to transpose each set of 4 bits before outputting the readings to the displays. This corrected the order of the bits thus giving the correct values in BCD.

When water was sprinkled the relative humidity would not change immediately since the evaporation process required time. The humidity sensor also required 50 s to detect any change thereof. To prevent the greenhouse being flooded, the relative humidity subroutine was adjusted so that water could be sprinkled for only 2.5 s in each cycle. A long delay was inserted after each sprinkle so that it would take at least 1 minute before the next sprinkle. The next section describes the systems performance showing how it responded to changes in the greenhouse environment

5.6 Performance

The system would begin to run when the reset button was depressed. The status LED would light and after 56 s the temperature would be read and displayed. Shortly thereafter the light level would be read and then at 1.04 minutes the humidity sensor would be read and displayed. The system would take a maximum of 1 minute to respond to a parameter change. To test the response of the system to environmental changes, the greenhouse was subjected to darkness, heating, air flow or a combination of these disturbances and the systems response noted. The results are discussed in the next sections.

5.6.1 Blowing Air

Using a blower, air at room temperature was blown into the greenhouse for 5 minutes when the greenhouse was at steady state. The temperature did not change while the relative humidity varied very slightly. Since the greenhouse temperature was the same as the room temperature, blowing air at room temperature did not have any effect on the temperature. No actuator was activated.

5.6.2 Darkness

Darkness was simulated by switching off external lights and observing the response of the system. The diagram in figure 5.9 summarises the systems response. The system was in steady conditions when darkness set in at 5 minutes. The lights in the greenhouse lit at 6 minutes. The humidity began to drop and at time 14 minutes the sprinkler was activated. The temperature remained steady until 12 minutes when it increased by 1 ^{o}C. When the darkness was terminated at 17 minutes, the lights in the greenhouse went off at 19 minutes and the sprinkler at 24 minutes. The increase in temperature and the decrease of the humidity were attributed to the heating effect of the lighting system. This effect could be controlled in a greenhouse by selecting an appropriate lighting lamp.

5.6.3 Applying Heat

The systems response to heat was tested by applying heat in the greenhouse using a 40W electric bulb. Figure 5.10 shows the results. The bulb was switched on at 5 minutes and terminated at 20 minutes. Temperature rose significantly and at 19 minutes the fan was activated. Humidity dropped to low levels activating the sprinkler at 20 minutes. Temperature and humidity were restored to optimum values at the 24 minutes.

5.6.4 Applying Heat and Air

The greenhouse was heated using a hot plate as air at room temperature was blown in at 5 minutes. This was discontinued at 10 minutes. The fan and the sprinkler were activated at 7 minutes and the

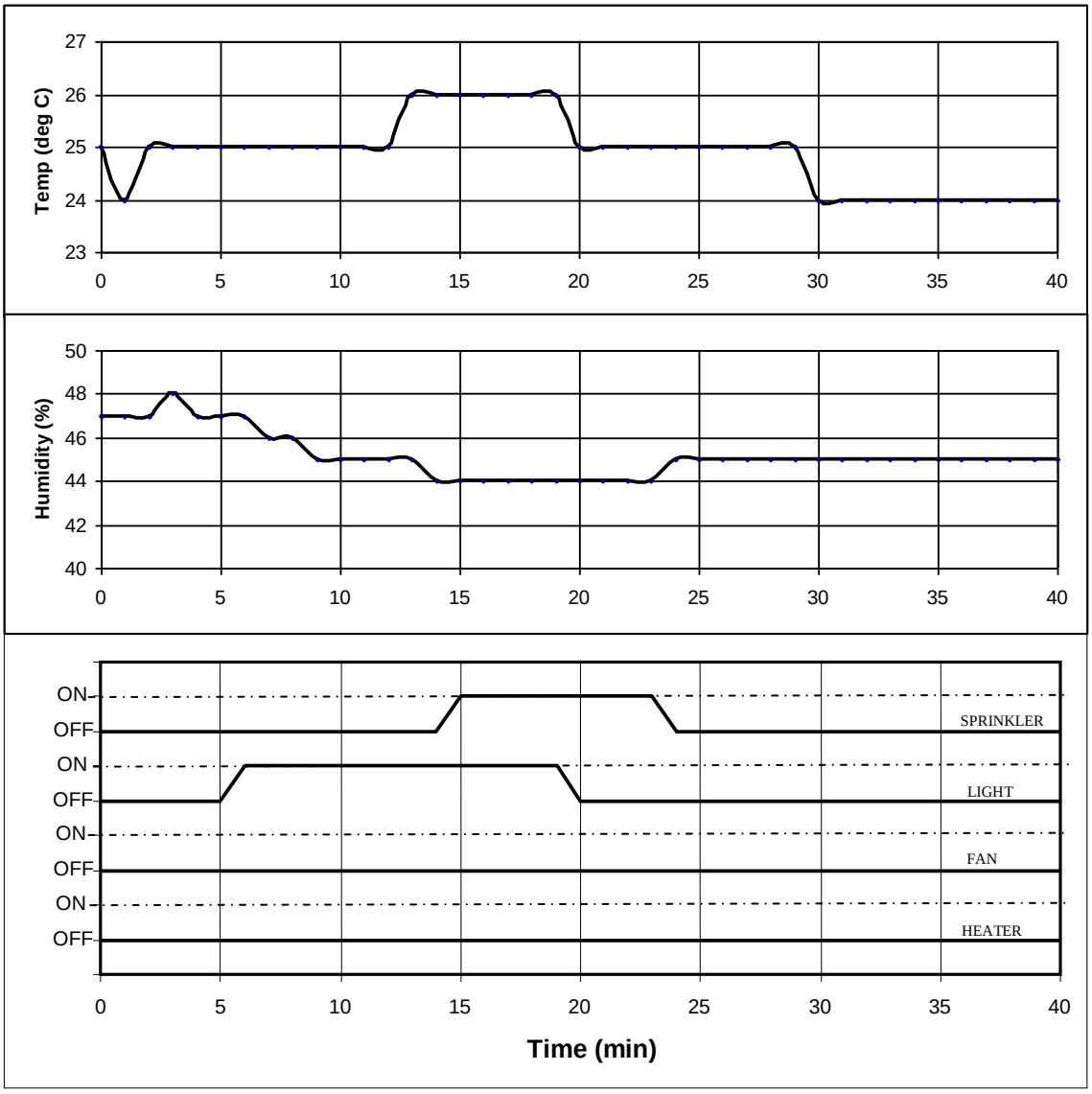


Figure 5.9: Systems response to darkness. The lights and the sprinkler are activated at different times. The fan and heater were not affected.

rate of temperature rise decreased slightly despite continued heating. The summarized results are shown in figure 5.11.

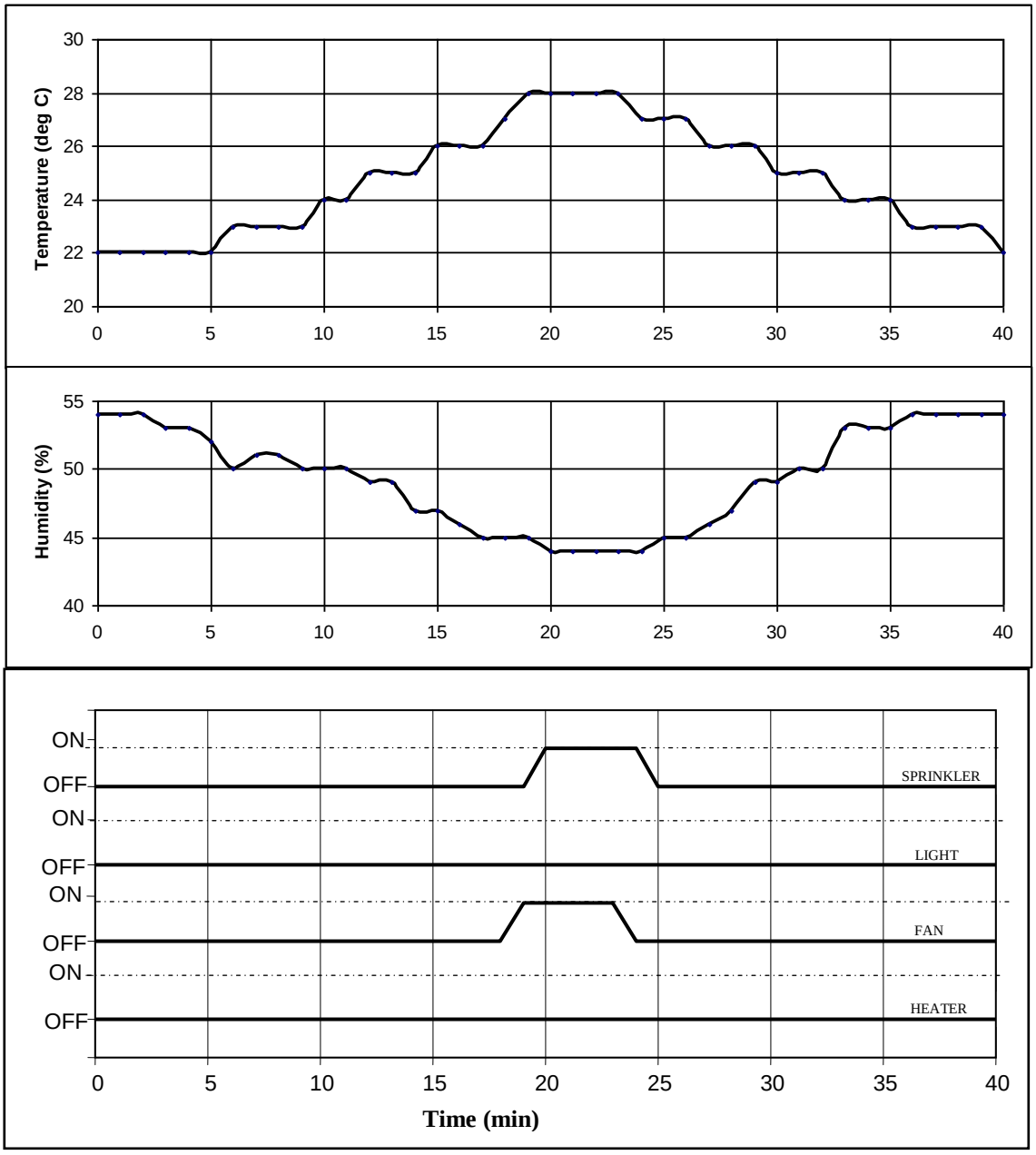


Figure 5.10: Systems response to heat. The fan and sprinkler are activated due the increased temperature and decreased humidity.

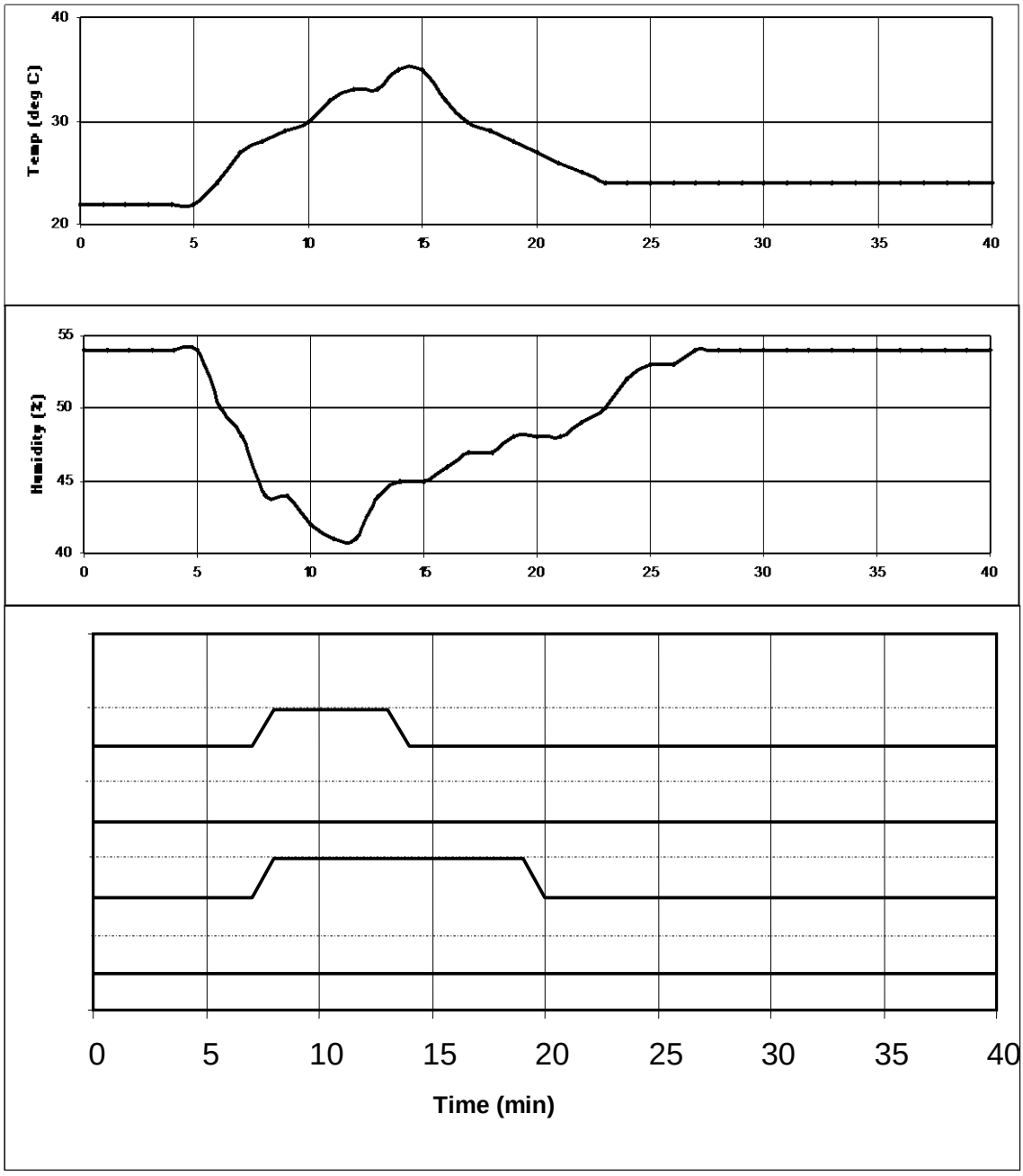


Figure 5.11: Systems response to heat and air. The fan and sprinkler were activated due the increased temperature and decreased humidity. The humidity was restored faster than the temperature.

Chapter 6

CONCLUSION AND OUTLOOK

6.1 Conclusion

A new greenhouse electronic control system based on the Z80 microprocessor has been developed and is fully described in this thesis. The system is compact, small, portable and affordable. It uses one voltage supply of +5 V and can use devices running on mains to control the greenhouse environment. The system monitors temperature, displays it, and controls a heater and fan to keep the temperature within a predetermined range. The system also monitors relative humidity, displays it and controls a water sprinkler to keep the relative humidity at an optimum value. Finally, the system detects light levels and controls the lighting system to keep light levels within predetermined levels in the greenhouse. In this way the system is able to optimize environmental parameters in the greenhouse as desired and set in the program. Thus our research objectives have been met.

6.2 Recommendations for Further Work

The greenhouse control system based on the Z80 microprocessor has been fully discussed in this thesis. We now present some recommendations and suggestions for further work that could be undertaken to improve the system.

The system developed uses single sensors to measure temperature, humidity and light. Whereas light levels are likely to be fairly constant within a greenhouse during the day, the same is not true for temperature and to a lesser extent, relative humidity. Thus, in a large greenhouse, several sensors could be placed at various places to measure, say, temperature. The average could be computed and used as basis for control.

Additionally, the roots, leaves, shoots, flowers and fruits require different temperatures. Infra red sensors can be used to monitor the temperature of leaves. Sensors embedded in the soil can monitor the root temperature. The temperature of the fruits or flowers can be monitored by attaching sensors on or next to them. These temperatures could then be controlled for optimum crop conditions.

The system could be further improved by interfacing the system to a computer to collect data on a continuous basis outputting the results on to a file. This data would be useful to identify equipment failure and bugs in the control algorithm and thus act as a basis for improving the system.

REFERENCES

Anderson, J. S. (1984). *Microprocessor Technology*. Buterrworth-Heinemann Ltd. Pp 37-46, 118-119, 190-195, 245, 290, 291.

Bakker, J. C. (1989). *The Effects of Temperature on Flowering, Fruit Set and Fruit Development of Glasshouse Sweet Pepper (Capsicum annuum L.)*. Journal of Horticultural Science 64(3):313-320.

Ball, S. R. (2002). *Embedded Microprocessor Systems*. Butterworth- Heinemann Ltd. Pp 219.

Bevolt, C. (1995). *Electronic Circuits: Discrete and Integrated*. McGraw-Hill Book Company, New York. Pp 95.

Bolton, W. (2000). *Microprocessor Systems.* Pearsons Education Limited. Pp 27-33, 152, 205, 224, 234.

Brugger, M. F., Short, T. H. and Bauerie, W. L. (1987). *An Evaluation of Horizontal Airflow in Six Commercial Greenhouses.* American Society of Agricultural Engineers. Summer meeting presentation number 37-4020.

Christian, S. (2006). *CMOS Humidity Sensors*. www.semirion.com Microsoft Internet Explorer. Pp 1, 2.

Demers, D. A., Charbonneau, J. J. and Gosselin, A. (1991). *Effects of Supplementary Lighting on the Growth and Production Activity of Greenhouse Sweet Pepper,* Can J. Plant. Sci. 71:587-594.

Demers, D. A. and Gosselin, A. (1998). *Effects of Supplemental Light Duration on Greenhouse Sweet Pepper Plants and Fruit Yields.* J. Amer. Soc Hort. Sci. 123(2):202-207.

Dogra, A. K. and Chandra, P. (2005). *Development of an Intelligent Controller for Greenhouse Management.* Microsoft Internet Explorer. Pp 1.

Downton, A. C. (1984). *Computers and Microprocessor, Components and Systems*. Van Nostrad Reinhold (UK) Co.Ltd. England.

Fraser, C. J. and Milne, J. S. (1999). *Microcomputer Applications in Measurement Systems*. Macmillan Education Ltd. Pp 91, 93.

Habby, J. (2005). http://www.unet.maine.edu/mtr101s99/LABS/LAB5/rh.htm. *USA TODAY.* Microsoft Internet Explorer.

Hanan, J. J. (1998). *Greenhouses: Advanced Technology for Protected Horticulture*. CRC Press, Boca Raton. Pp 66.

Heath, S. (2003). *Embedded Systems Design*. Buterrworth-Heinemann Ltd.

Hersh, R. (2004). *Embedded Processor and Microcontroller Primer and FAQ*. Microsoft Internet Explorer.

http://data-acquisition.globalspec.com. (2007). Pp1-5.

http://www.electonickits.com. (2006). *Carls Electronics.* Microsoft Internet Explorer.

http://www.en.wikipeadia.org/wiki/Breadboard. (2006). *Breadboard.* Microsoft Internet Explorer.

http://www.en.wikipeadia.org/wiki/Printed Circuit board. (2006). *Printed Circuit Board.* Microsoft Internet Explorer.

Khosla, S. (1999). *In "Greenhouse Vegetable Experts Discuss the Future of the Industry"*. Tomato Magazine, August, 1999. Yakima Washington.

Koning, D. (1996). *Quantifying the Responses to Temperature of Different Plant Processes involved in Growth and Development of Glasshouse Tomato*. Acta Horticulturae 406:99-104.

Lange, D. and Tantau, H. J. (1996). *Climate Management for Disease Control Investigations on Control strategies, Plant Densities and Irrigation Systems*. Acta Horticulturae 406:105-113.

Leung, C. M., Wilson, D. R., Marquand, C. J. and Tassou, S. A. (2005). *A Knowledge Based System for the Control of a Greenhouse Environment*. HIS Acta Horticulture 174: Symposium Greenhouse Climate and its Control. Microsoft Internet Explorer.

Marhaento, B. and Singh, G. (2002). *Development of a Computer Based Greenhouse Environment Controller.* American Society of Agriculture and Biological Engineers. Michigan. www.asabe.org. Pp 136-146.

Martin, G. (1982). *Microprocessor Application Level III.* Hutchison and Co. (Publishers) Ltd.

Moxham, J. (1996). *ZINT Z80 Interpreter.* Microsoft Internet Explorer. Pp 1-15, 20, 26.

Nabard, (2006). http://www.nabard.org/roles/ms/ph/polyhouse.htm. *Bankable Scheme/ Area Development Project on Cultivation of high value cash crops under Greenhouse/Polyhouse in Darjeeling hills region.* Microsoft Internet Explorer. Pp 1-6.

National Semiconductor. (1995). *National Semiconductor's Handbook.* Microsoft Internet Explorer. Pp 6.

Palmer, C. (2000). http://www.usatoday.com/weather/whumcalc.htm. USA TODAY. Microsoft Internet Explorer. Pp1-4.

Papadakis, G., Frangoudakis, A. and Kyritsis, S. (1994). *Experimental Investigation and Modeling of Heat and Mass Transfer Between a Tomato Crop and the Greenhouse Environment*. Journal of Agricultural Engineering. Res. 57:217-227.

Papadopoulos, A. P. and Pararajasingham, S. (1997). *The Influence of Plant Spacing on Light Interception and Use in Greenhouse Tomato (Lycopersicon Esculentum Mill.): A Review.* Scientia Horticulturae 69:1-27.

Plaksina, O. and Raush, T. (2005). *Development of a KNX Based Greenhouse Climate Control*. Microsoft Internet Explorer. Pp 1-4.

Portree, J. (1996). *Greenhouse Vegetable Production Guide for Commercial Growers.* Province of British Columbia Ministry of Agriculture, Fisheries and Food.

Ramsay, D. C. (1981). *Engineering Instrumentation and Control*. Pitman Press, Great Britain.

Rongen, H. (2005). *Introduction to Microprocessors and Microcomputers*. Microsoft Internet Explorer. Pp 15.

Salisbury, F. B. and Ross, C. W. (1978). *Plant Physiology* 2nd Edition. Wadsworth Publishing Company, Inc. Belmont California.

Seginer, I. (1996). *Optimal Control of the Greenhouse Environment: An Overview.* Acta Horticulturae 406:191-201.

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

Stanghellini, C. and Meurs, V. W. M. (1992). *Environmental Control of a Tomato Crop Using a Transpiration Model*. Acta Horticulturae 303:23-30.

Stanghellini, C. and Meurs, V. W. M. (1992). *Environmental Control of Greenhouse Crop Transpiration.* J. Agric. Engng Res. 51:297-311.

Tooley, M. (1996). *Electronics Circuits Handbook, Design Testing and Construction.* Newness, Great Britain.

Uffenbeck, J. (2003). *Microcomputers and Microprocessors*. Prentice-Hall of India Limited. Pp 286, 297.

Wills, A. (2002). *Development and Test of Sensor-Aided Microcontroller Based Irrigation System with Web Browser Interface*. Microsoft Internet Explorer. Pp2, 9.

Wilson, J. W., Hand, D. W and Hannah, M. A. (1992). *Light Interception and Photosynthetic Efficiency in Some Glasshouse Crops*. Journal of Experimental Botany 43(248):363-373.

Zilog Z80 8-bit Microprocessor, htm, (2001). Microsoft Internet Explorer. Pp 451-453.

APPENDICES

1 TEST BIT PROGRAM

```
*****************************************************************
*Test program to check bits. Puts alternate bits ON and moves thro all
*ports of the PIAs one at a time and then reverses the bits in the
*next round. This process is repeated continuously
*****************************************************************

          ORG 0000H      ;origin

          LD A,80H       ;control word
          OUT (0003H),A  ;all ports PIA#1 output
          OUT (0007H),A  ;all ports PIA#2 output
          LD SP,03E0HH    ;initialize SP

START:    LD A,55H       ;load 01010101

          OUT (0000H),A  ;display port A PIA#1
          CALL DELAY
          OUT (0001H),A  ;display port B PIA#1
          CALL DELAY
          OUT (0002H),A  ;display port C PIA#1
          CALL DELAY

          OUT (0004H),A  ;display port A PIA#2
          CALL DELAY
          OUT (0005H),A  ;display port B PIA#2
          CALL DELAY
          OUT (0006H),A  ;display port C PIA#2
          CALL DELAY

          LD A,0AAH       ;load 10101010

          OUT (0000H),A  ;display port A PIA#1
          CALL DELAY
          OUT (0001H),A  ;display port B PIA#1
          CALL DELAY
          OUT (0002H),A  ;display port C PIA#1
          CALL DELAY

          OUT (0004H),A  ;display port A PIA#2
          CALL DELAY
          OUT (0005H),A  ;display port B PIA#2
          CALL DELAY
          OUT (0006H),A  ;display port C PIA#2
```

```
          CALL DELAY

          JP START       ;repeat

DELAY:    PUSH AF
          PUSH BC
          LD BC,0A8FFH   ;delay counter ~1s

DLOOP:    DEC BC         ;decrement value
          LD A,C
          OR B           ;test zero flag
          JR NZ,DLOOP    ;repeat till zero

          POP BC
          POP AF
          RET
```

2 COUNT PROGRAM

```
;*********************************************************************
; Program to count up continuously from 0 to 9 and output to all ports
;one at time continuously
;*********************************************************************

        ORG 0000H       ;origin

DATA:   LD HL,0800H
        LD (HL),3FH     ;0
        INC HL
        LD (HL),06H     ;1
        INC HL
        LD (HL),5BH     ;2
        INC HL
        LD (HL),4FH     ;3
        INC HL
        LD (HL),76H     ;4
        INC HL
        LD (HL),6DH     ;5
        INC HL
        LD (HL),7DH     ;6
        INC HL
        LD (HL),0FH     ;7
        INC HL
        LD (HL),7FH     ;8
        INC HL
        LD (HL),6FH     ;9
        INC HL

START:   LD A,0FH       ;port A mode 0
         OUT (0002H),A ;i.e. to o/p
         LD SP,1000H   ;initialize stack pointer

REPEAT:  LD HL,0800H   ;first num
GO:      LD A,(HL)     ;num into A
         OUT (0000H),A ;output num
         INC HL
         CALL DELAY
         CP 6FH        ;is it 9?
         JR Z,REPEAT   ;yes? JR
         JP GO         ;if not, GO!

DELAY:   LD B,0DDH
LOOP2:   LD C,0FFH
LOOP1:   DEC C
```

```
JP NZ,LOOP1
DEC B
JR NZ,LOOP2
RET
```

3 GREENHOUSE CONTROL PROGRAM

```
;-------------------------------------------------------------------------------
;                        GREENHOUSE CONTROL PROGRAM
;-------------------------------------------------------------------------------
PROG            EQU 0000H         ;origin
PIA1            EQU 0003H         ;8255 PIA #1 control port
PIA2            EQU 0007H         ;8255 PIA #2 control port
CTWD1           EQU 009BH         ;configures all ports inputs
CTWD2           EQU 0080H         ;configures all ports outputs
;-------------------------------------------------------------------------------
;LIGHTIN- light in accepts input from light sensor
;   RHIN- relative humidity in accepts input from humidity sensor via the ADC
; TEMPIN- temperature in accepts input from temperature sensor via the ADC
;-------------------------------------------------------------------------------
RHIN            EQU 0002H         ;8255 PIA #1 port C
LIGHTIN         EQU 0001H         ;8255 PIA #1 port B
TEMPIN          EQU 0000H         ;8255 PIA #1 port A
;-------------------------------------------------------------------------------
;  RHDIS- displays relative humidity on a 7-segment display
;TEMPDIS- displays temperature on a 7-segment display
;   CTRL- controls the actuators by using its single bits for each
;  MEMBF- memory buffer stores values for temp and RH in RAM
;-------------------------------------------------------------------------------
RHDIS           EQU 0006H         ;8255 PIA #2 port C
TEMPDIS         EQU 0005H         ;8255 PIA #2 port B
CTRL            EQU 0004H         ;8255 PIA #2 port A

MEMBF1          EQU 0800H         ;i/p mem buffer for temp
MEMBF2          EQU 0805H         ;o/p mem buffer for temp
MEMBF3          EQU 0810H         ;i/p mem buffer for RH
MEMBF4          EQU 0815H         ;o/p mem buffer for RH
;------------------------------------------------------------ ----------------------
;sets temperature and relative humidity limits
;-------------------------------------------------------------------------------
TEMPL           EQU 19H           ;lower temp limit=25degC
TEMPH           EQU 1CH           ;higher temp limit=28degC
RHL             EQU 37H           ;lower level of RH=50%
;-------------------------------------------------------------------------------
;                          MAIN
;-------------------------------------------------------------------------------
                ORG 0000H         ;origin
                LD SP,0900H       ;initialize stack pointer

                CALL INIT         ;initialize, configure ports
                CALL INITADC      ;initialize ADCs, reset displays, status LED

                CALL LDELAY       ;HIH-4000 has a response time of 50s
                CALL LEDONOFF     ;shows system running

MAIN:           CALL CNVT         ;initiates ADC conversion
                CALL ENDCONTEMP   ;check for end of conversion for temperature
                CALL TEMPCTRL     ;read temp and control the fan and heater
                CALL TEMPDISP     ;converts temp to BCD,display on 7-seg in deg C

                CALL LEDONOFF     ;puts status led off for 2s after each run
```

```
                CALL LIGHT          ;read light levels and control lighting system

                CALL LEDONOFF       ;puts status led off for 2s after each run

                CALL CNVT           ;initiates ADC conversion
                CALL ENDCONRH       ;check for end of conversion for rel humidity
                CALL RHCTRL         ;read rel humidity, and control the sprinkler
                CALL RHDISP         ;converts rel hum to BCD o/p on 7-seg in %

                CALL LEDONOFF       ;puts status led off for 2s after each run

                JP MAIN
                HALT
;-------------------------------------------------------------------------------
;initialize CPU, interrupts and configuring I/O Ports OF PIA#1 and PIA#2
;-------------------------------------------------------------------------------
INIT:           CALL DELAY          ;initialize CPU
                EI                  ;enable interrupt
                DI                  ;disable interrupt
                LD A,CTWD1          ;load A with cntrl word 1
                OUT (PIA1),A        ;all ports PIA 1 inputs
                LD A,CTWD2          ;load A with cntrl word 2
                OUT (PIA2),A        ;all ports PIA 2 outputs
                RET
;-------------------------------------------------------------------------------
;uses D reg to control the following for the ADCs:
;bit 0- CS, active low, bit 1 - RD, active low, bit 2 - WR, active low
;bit 3- system status, when set shows system is running, bit 7- fans
;bit 6- heaters,  bit 5- sprinklers, bit 4- lighting system
;-------------------------------------------------------------------------------
INITADC:        LD D,07H            ;0000 0111, ;initialize ADCs, led on
                LD A,D
                OUT (CTRL),A
                CALL DELAY
                CALL DELAY

                LD D,A
                OUT (CTRL),A
                LD A,00H
                OUT (TEMPDIS),A     ;clears the 7-segment temp display
                CALL DELAY
                OUT (RHDIS),A       ;clears the 7-segment rel hum display
                CALL DELAY
                RET
;-------------------------------------------------------------------------------
; initiates conversion by ADC
; use port CTRL bits 0-2 to initiate and stop conversion of ADCs by controlling ; CS-
bit 0, RD-bit 1 and WR-bit 2, all active low
;-------------------------------------------------------------------------------
CNVT:           RES 0,D             ;XXXX 0110
                LD A,D
                OUT (CTRL),A        ;puts CS low
                NOP
                NOP                 ;short delay to synchronize timing for ADC
                RES 2,D             ;XXXX 0010
                LD A,D
```

```
                OUT (CTRL),A      ;puts WR low, conversion starts
                CALL DELAY
                CALL DELAY

                SET 2,D
                LD A,D
                OUT (CTRL),A      ;puts WR high
                NOP
                NOP               ;short delay
                SET 0,D           ;XXXX 0111
                LD A,D
                OUT (CTRL),A      ;puts CS high, holds converter
                RET
;-------------------------------------------------------------------------------
;use bit 0 of LIGHTIN to check INTR of ADC#1 for end of conversion for temperature
;-------------------------------------------------------------------------------
ENDCONTEMP:     IN A,(LIGHTIN)     ;i/p port B, PIA#1,bit 0 to INTR of ADC#1
                BIT 0,A            ;test bit 0, if set jump to repeat
                JP NZ,ENDCONTEMP   ;checks INTR for EOC and if "1",loop
                CALL DELAY
                CALL DELAY

                RES 0,D
                LD A,D
                OUT (CTRL),A      ;puts CS low
                NOP
                NOP               ;short delay

                RES 1,D           ;XXXX 0100
                LD A,D            ;puts CS and RD low
                OUT (CTRL),A      ;puts RD low, reads data from ADC#1
                RET
;-------------------------------------------------------------------------------
;sets limits, reads and controls temperature using fans and heaters
;-------------------------------------------------------------------------------
TEMPCTRL:       IN A,(TEMPIN)     ;read temp from ADC#1
                ADD A,06H         ;add 06D to calibrate sensor
                LD E,00H          ;clear E reg
                LD E,A            ;store temp in E reg
                CALL DELAY

                LD B,TEMPL        ;sets lower temp
                CP B              ;compare TEMPIN with TEMPL
                JP C,HEATER       ;if A<C, puts HEATER on

                LD B,TEMPH        ;sets upper temp
                CP B              ;compare TEMPIN with TEMPH
                JP NC,FAN         ;if A>B, puts FAN on

                JP C,FANHTROFF    ;puts off fan and heater

NEXT1:          SET 1,D
                LD A,D
                OUT (CTRL),A
                SET 0,D
                LD A,D
                OUT (CTRL),A       ;XXXX 0111, ready for next cycle
```

```
              RET
;-------------------------------------------------------------------------------
;converts temperature from binary to BCD and displays on 7-seg display
;-------------------------------------------------------------------------------
TEMPDISP:     PUSH BC
              LD A,E            ;restore temp in A
              LD (MEMBF1),A     ;store temp in membf1
              LD HL,MEMBF1      ;points HL to bin num store
              LD A,(HL)         ;transfer byte
              LD HL,MEMBF2      ;point HL to o/p buff mem
              CALL BIN2BCD      ;call bcd converter
              LD A,(MEMBF2)     ;load A with temp in bin
              RRCA
              RRCA
              RRCA
              RRCA              ;changes msb to lsb
              ADD A,C
              CALL TRANSPOSE    ;transpose the display
              OUT (TEMPDIS),A   ;display temp on 7-Seg
              CALL DELAY
              CALL DELAY
              POP BC            ;restores C counter
              RET
;-------------------------------------------------------------------------------
;activates the fans when temperature rises above TEMPH, puts heater off if on
;-------------------------------------------------------------------------------
FAN:          RES 6,D
              LD A,D
              OUT (CTRL),A      ;puts HEATER off
              CALL DELAY
              CALL DELAY

              SET 7,D
              LD A,D
              OUT (CTRL),A      ;puts FAN on
              CALL DELAY
              JP NEXT1
;-------------------------------------------------------------------------------
;activates the heater when temperature falls below TEMPL, puts fan off if on
;-------------------------------------------------------------------------------
HEATER:       RES 7,D
              LD A,D
              OUT (CTRL),A      ;puts FAN off
              CALL DELAY
              CALL DELAY

              SET 6,D
              LD A,D
              OUT (CTRL),A      ;puts HEATER on
              CALL DELAY
              JP NEXT1
;-------------------------------------------------------------------------------
;puts fan or heater off if on
;-------------------------------------------------------------------------------
FANHTROFF:    PUSH AF

              RES 7,D
```

```
                LD A,D
                OUT (CTRL),A      ;puts FAN off
                CALL DELAY
                CALL DELAY

                RES 6,D
                LD A,D
                OUT (CTRL),A      ;puts HEATER off
                CALL DELAY

                POP AF
                RET
;-------------------------------------------------------------------------------
;controls light levels by using the lighting system
;-------------------------------------------------------------------------------
LIGHT:          IN A,(LIGHTIN)    ;i/p from light sensor on
                CALL DELAY
                BIT 2,A           ;bit 2 of LIGHTIN port
                JP NZ,LIGHTON     ;if 0, jump to LIGHTON

                JP Z,LIGHTOFF     ;if 1, put off lights

NEXT2:          RET
;-------------------------------------------------------------------------------
;activates the lighting system
;-------------------------------------------------------------------------------
LIGHTON:        PUSH AF

                SET 4,D
                LD A,D
                OUT (CTRL),A      ;puts LIGHTS on
                CALL DELAY

                POP AF
                JP NEXT2
;-------------------------------------------------------------------------------
;deactivates the lighting system
;-------------------------------------------------------------------------------
LIGHTOFF:       PUSH AF

                RES 4,D
                LD A,D
                OUT (CTRL),A      ;puts LIGHTS off
                CALL DELAY

                POP AF
                RET
;-------------------------------------------------------------------------------
; Uses bit 1 of LIGHTIN to check INTR of ADC#2 for end of conversion
;-------------------------------------------------------------------------------
ENDCONRH:       IN A,(LIGHTIN)    ;input port B, PIA#1,
                BIT 1,A           ;INTR connected to bit 1 of port A
                JP NZ,ENDCONRH   ;checks INTR for EOC, if "1",loops
                CALL DELAY
                CALL DELAY

                RES 0,D
```

```
                LD A,D
                OUT (CTRL),A
                NOP
                NOP
                RES 1,D           ;XXXX 1100
                LD A,D            ;puts CS and RD low
                OUT (CTRL),A      ;ready to read data from ADC#2
                RET
;-------------------------------------------------------------------------------
; sets limit, reads and controls relative humidity using a water sprinkler
;-------------------------------------------------------------------------------
RHCTRL:        IN A,(RHIN)       ;read RH from ADC#2
               ADD A,06H         ;add 06D to calibrate sensor

                LD E,00H         ;clear E reg
                LD E,A           ;save rel hum in E reg
                CALL DELAY

                LD B,RHL          ;load B with lower RH limit
                CP B              ;compare RH with RHL

                JP C,SPRKLR       ;if A<B,jump to SPRINKLER

                SET 1,D
                LD A,D
                OUT (CTRL),A
                SET 0,D
                LD A,D
                OUT (CTRL),A       ;XXXX 0111, ready for next cycle
                RET
;-------------------------------------------------------------------------
; converts relative humidity from binary to BCD and displays on 7-seg display
;-------------------------------------------------------------------------------
RHDISP:         PUSH BC
                LD A,E
                LD (MEMBF3),A     ;save RH in membf3
                LD HL,MEMBF3      ;point HL  bin num stored
                LD A,(HL)         ;transfer byte
                LD HL,MEMBF4      ;point HL o/p buff mem
                CALL BIN2BCD      ;call bcd converter
                LD A,(MEMBF4)     ;load A byte 1
                RRCA
                RRCA
                RRCA
                RRCA
                ADD A,C
                CALL TRANSPOSE
                OUT (RHDIS),A     ;display RH on 7-Segment
                CALL DELAY
                CALL DELAY
                POP BC
                RET
;-------------------------------------------------------------------------------
; activates the sprinkler for 4s in each run when humidity falls below RHL
;-------------------------------------------------------------------------------
SPRKLR:         PUSH AF
```

```
                SET 5,D           ;sets bit 5 of CTRL
                LD A,D
                OUT (CTRL),A      ;puts SPRINKLER on
                CALL DELAY
                CALL DELAY
                CALL DELAY
                CALL DELAY

                RES 5,D           ;sets bit 5 of CTRL
                LD A,D
                OUT (CTRL),A      ;puts SPRINKLER off after 2.5s

                CALL LDELAY       ;enable time for humidity to change

                POP AF
                RET
;-------------------------------------------------------------------------------
; puts system status led off for 2s after each complete run
; ensures system is not hanging
;-------------------------------------------------------------------------------
LEDONOFF:       SET 3,D
                LD A,D
                OUT (CTRL),A      ;puts status led off for 2s
                CALL DELAY
                CALL DELAY
                CALL DELAY
                RES 3,D           ;restores led
                RET
;-------------------------------------------------------------------------------
; converts binary numbers to BCD for displaying on 7-segment
;-------------------------------------------------------------------------------
BIN2BCD:        LD B,10          ;load 10 into register B
                LD (HL),0FFH     ;load buffer with -1

LOOP:           INC (HL)         ;clr buffer and incrmt
                SUB B            ;sub pwr of 10 from bin no
                JR NC,LOOP       ;if num>pwr of 10, inc buf
                ADD A,B          ;add pwr of 10 to get rem
                INC HL           ;go to next buffer location
                LD (HL),A        ;store BCD1
                LD C,A            ;store byte 2
                RET
;-------------------------------------------------------------------------------
; takes care of a hardware 7-seg display configuration
;-------------------------------------------------------------------------------
TRANSPOSE:      PUSH DE
                PUSH BC

                LD E,A
                LD B,8
TRANSPOSE1:     LD A,E
                RRCA
                LD E,A
                LD A,D
                RLA
                LD D,A
                DEC B
```

```
                JP NZ,TRANSPOSE1
                LD A,D

                POP BC
                POP DE
                RET
;-------------------------------------------------------------------------------
; enables a 0.8s delay
;-------------------------------------------------------------------------------
DELAY:          PUSH AF
                PUSH BC

                LD B,0BBH
LOOP2:          LD C,0FFH
LOOP1:          DEC C
                JR NZ,LOOP1
                DEC B
                JR NZ,LOOP2

                POP BC
                POP AF
                RET
;-------------------------------------------------------------------------------
; enables a 50s delay
;-------------------------------------------------------------------------------
LDELAY:         PUSH AF
                PUSH BC

                LD C,63H
LOOP3:          CALL DELAY
                DEC C
                JP NZ,LOOP3

                POP BC
                POP AF
                RET
```

4 ZILOG Z80 INSTRUCTIONS

This is a summary of the opcodes of the Zilog Z80.If an EDxx instruction is not listed, it should operate as two NOPs. If a DDxx or FDxx instruction is not listed, it should operate as without the DD or FD prefix, and the DD or FD prefix itself should operate as a NOP. An asterisk * means undocumented. A LD A,RLC (IX+d) means that the result of RLC (IX+d) is not only stored in (IX+d), but also in A. SLL x operates the same as SLA x, except that SLL inserts 1 to the left. OUT (C),0 always outs zero. IN F,(C) / IN (C) does not store the result from the input. It only affects the flags, as the other IN r,(C) instructions do. These two mnemonics both refer to the same undocumented instruction. IM 0/1 sets the Z80 in IM 0 or in IM 1 mode, unknown which one at this moment.

Opcode	Mnemonic	T	M	M1
00	NOP	4	1	1
01 n n	LD BC,nn	10	3	1
02	LD (BC),A	7	2	1
03	INC BC	6	1	1
04	INC B	4	1	1
05	DEC B	4	1	1
06 n	LD B,n	7	2	1
07	RLCA	4	1	1
08	EX AF,AF’	4	1	1
09	ADD HL,BC	11	3	1
0A	LD A,(BC)	7	2	1
0F	RRCA	4	1	1
10 e	DJNZ (PC+e)	8/13	2/3	1/1
11 n n	LD DE,nn	10	3	1

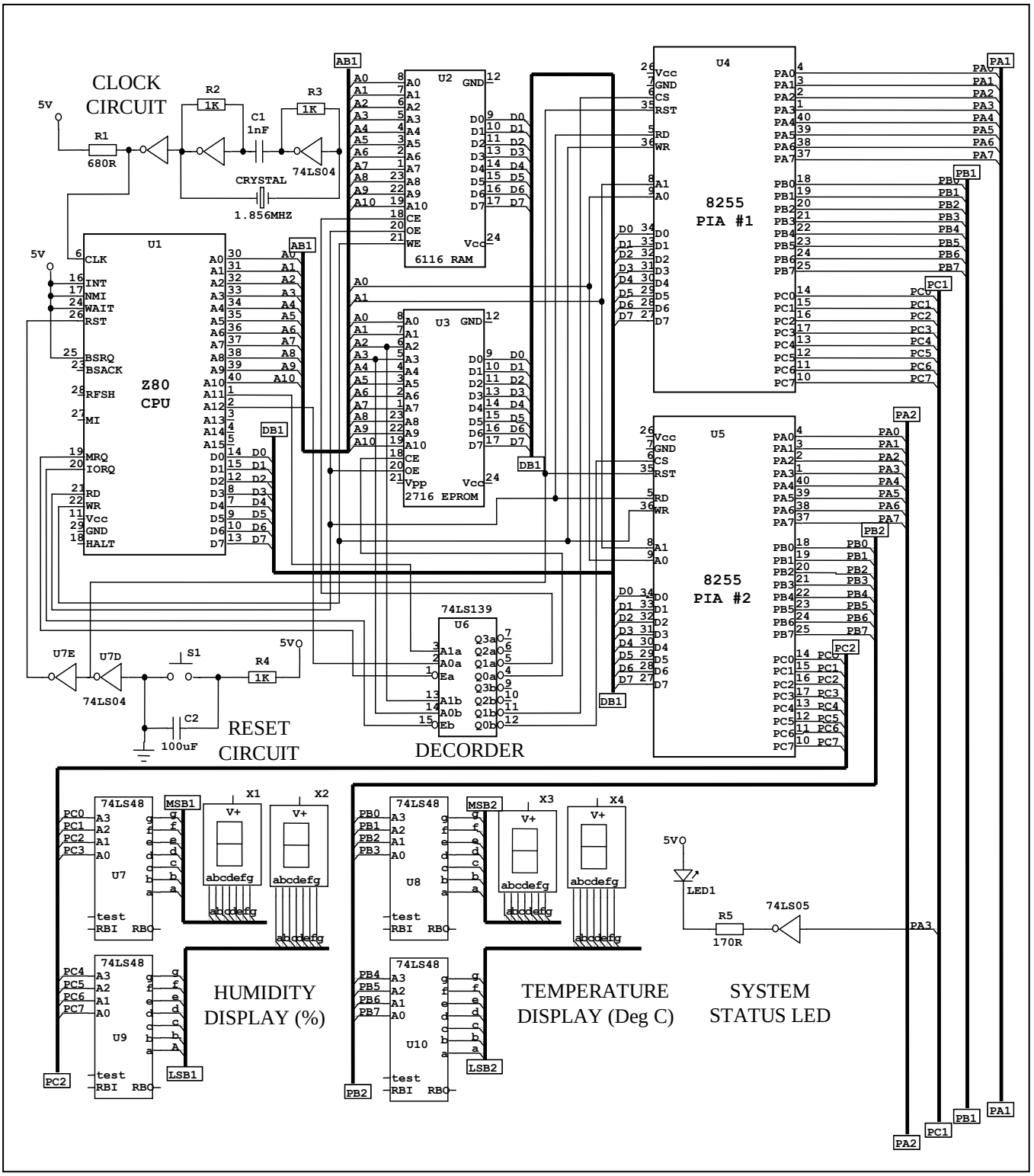
Opcode	Mnemonic	T	M	M1
12	LD (DE),A	7	2	1
17	RLA	4	1	1
18 e	JR (PC+e)	12	3	1
19	ADD HL,DE	11	3	1
1A	LD A,(DE)	7	2	1
1F	RRA	4	1	1
20 e	JR NZ,(PC+e)	12/7	3/2	1/1
21 n n	LD HL,nn	10	3	1
22 n n	LD (nn),HL	16	5	3
27	DAA	4	1	1
28 e	JR Z,(PC+e)	12/7	3/2	1/1
29	ADD HL,HL	11	3	1
2A n n	LD HL,(nn)	16	5	1
2F	CPL	4	1	1
30 e	JR NC,(PC+e)	12/7	3/2	1/1
31 n n	LD SP,nn	10	3	1
32 n n	LD (nn),A	13	4	1
33	INC SP	6	1	1
34	INC (HL)	11	3	1
35	DEC (HL)	11	3	1
36 n	LD (HL),n	10	3	1
37	SCF	4	1	1
38 e	JR C,(PC+e)	12/7	3/2	1/1
39	ADD HL,SP	11	3	1
3A n n	LD A,(nn)	13	4	1
3B	DEC SP	6	1	1
3E n	LD A,n	7	2	1
3F	CCF	4	1	1
41	LD B,C	4	1	1
46	LD B,(HL)	7	2	1

Opcode	Mnemonic	T	M	M1
71	LD (HL),C	7	2	1
76	HALT	4	1	1
80	ADD A,B	4	1	1
86	ADD A,(HL)	7	2	1
88	ADC A,B	4	1	1
8E	ADC A,(HL)	7	2	1
90	SUB B	4	1	1
91	SUB C	4	1	1
92	SUB D	4	1	1
96	SUB (HL)	7	2	1
97	SUB A	4	1	1
98	SBC A,B	4	1	1
9E	SBC A,(HL)	7	2	1
9F	SBC A,A	4	1	1
A0	AND B	4	1	1
A8	XOR B	4	1	1
AE	XOR (HL)	7	2	1
B0	OR B	4	1	1
B6	OR (HL)	7	2	1
B8	CP B	4	1	1
BE	CP (HL)	7	2	1
C0	RET NZ	11/5	3/1	1/1
C1	POP BC	10	3	1
C2 n n	JP NZ,(nn)	10	3	1
C3 n n	JP (nn)	10	3	1
C4 n n	CALL NZ,(nn)	17/10	5/3	1/1
C5	PUSH BC	11	3	1
C6 n	ADD A,n	7	2	1
C7	RST 0H	11	3	1
C8	RET Z	11/5	3/1	1/1

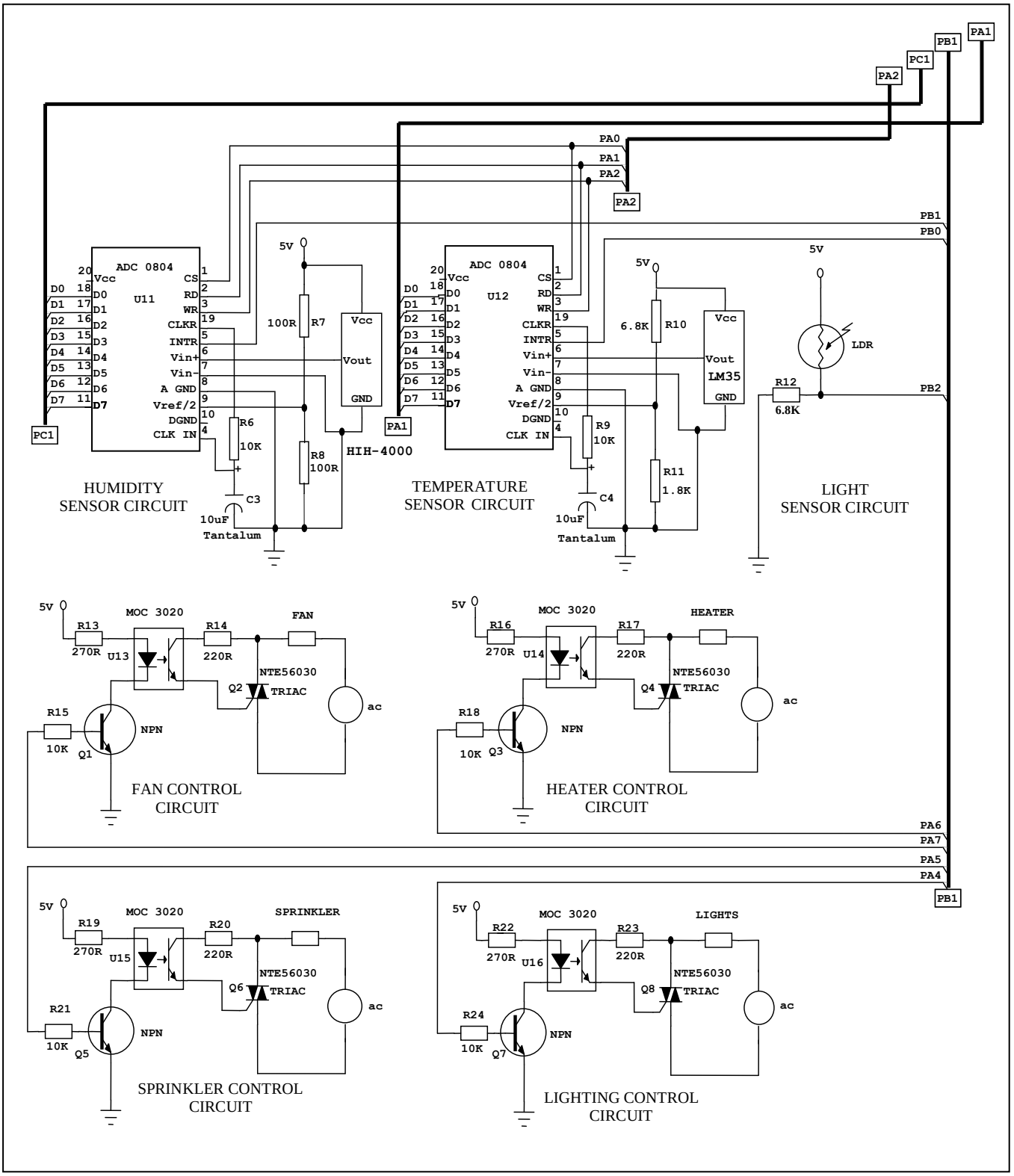
Opcode	Mnemonic	T	M	M1
C9	RET	10	3	1
CA n n	JP Z,(nn)	10	3	1
CB00	RLC B	8	2	2
CB06	RLC (HL)	15	4	2
CB08	RRC B	8	2	2
CB0E	RRC (HL)	15	4	2
CB10	RL B	8	2	2
CB16	RL (HL)	15	4	2
CB18	RR B	8	2	2
CB1E	RR (HL)	15	4	2
CB26	SLA (HL)	15	4	2
CB2D	SRA L	8	2	2
CB2E	SRA (HL)	15	4	2
CB30	SLL B*	8	2	2
CB36	SLL (HL)*	15	4	2
CB38	SRL B	8	2	2
CB3E	SRL (HL)	15	4	2
CB40	BIT 0,B	8	2	2
CB46	BIT 0,(HL)	12	3	2
CB80	RES 0,B	8	2	2
CBBE	RES 7,(HL)	15	4	2
CBBF	RES 7,A	8	2	2
CBC0	SET 0,B	8	2	2
CBFF	SET 7,A	8	2	2
CC n n	CALL Z,(nn)	17/10	5/3	1/1
CD n n	CALL (nn)	17	5	1
CE n	ADC A,n	7	2	1
CF	RST 8H	11	3	1
D0	RET NC	5	1	1
D1	POP DE	10	3	1

Opcode	Mnemonic	T	M	M1
D2 n n	JP NC,(nn)	10	3	1
D3 n	OUT (n),A	11	3	1
D4 n n	CALL NC,(nn)	17/10	5/3	1/1
D5	PUSH DE	11	3	1
D6 n	SUB n	7	2	1
D7	RST 10H	11	3	1
D8	RET C	5	1	1
D9	EXX	4	1	1
DA n n	JP C,(nn)	10	3	1
DB n	IN A,(n)	11	3	1
DC n n	CALL C,(nn)	17/10	5/3	1

5 CIRCUIT DIAGRAM FOR THE GREENHOUSE CONTROL SYSTEM

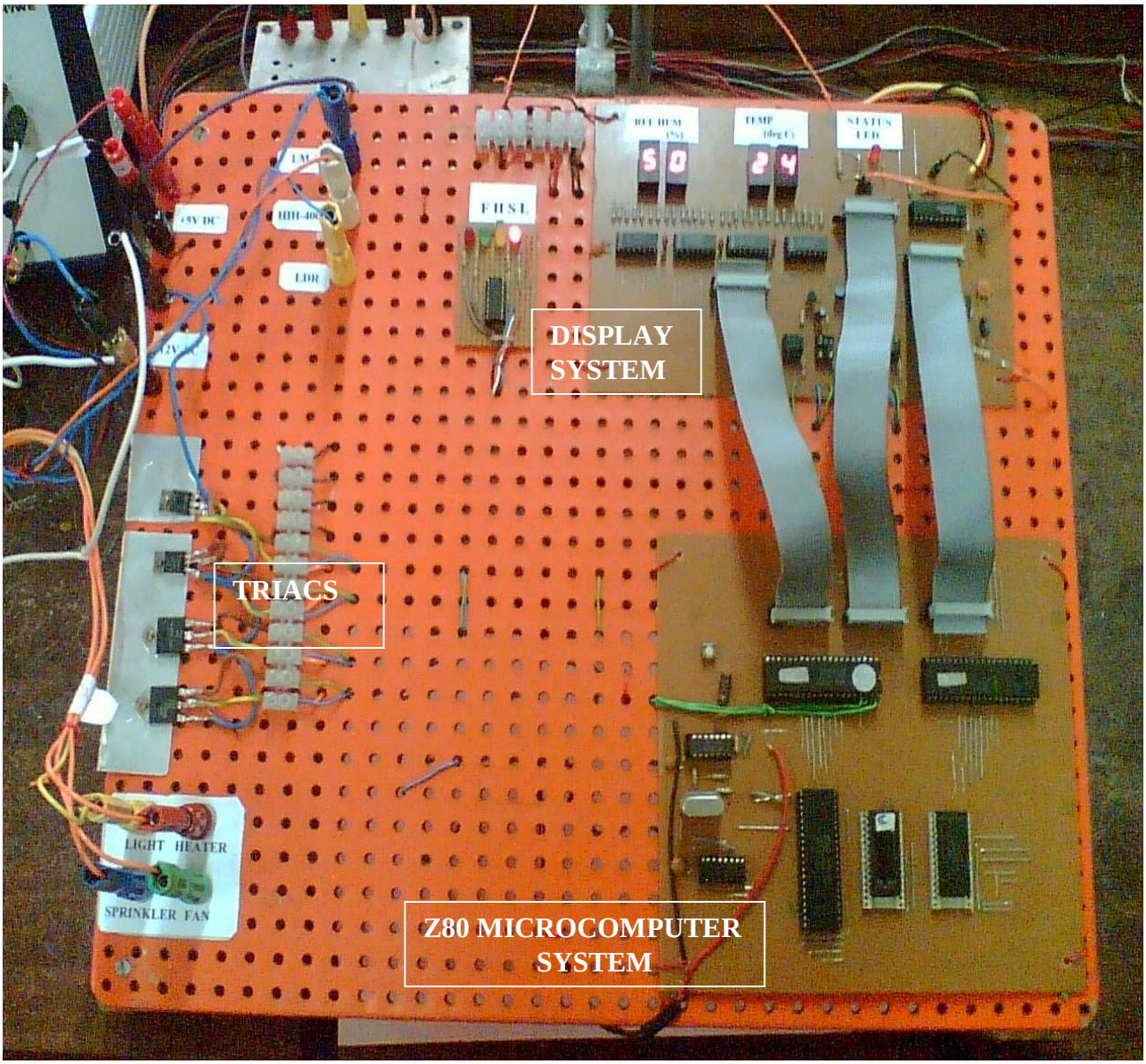


Complete circuit diagram (*contd.*)

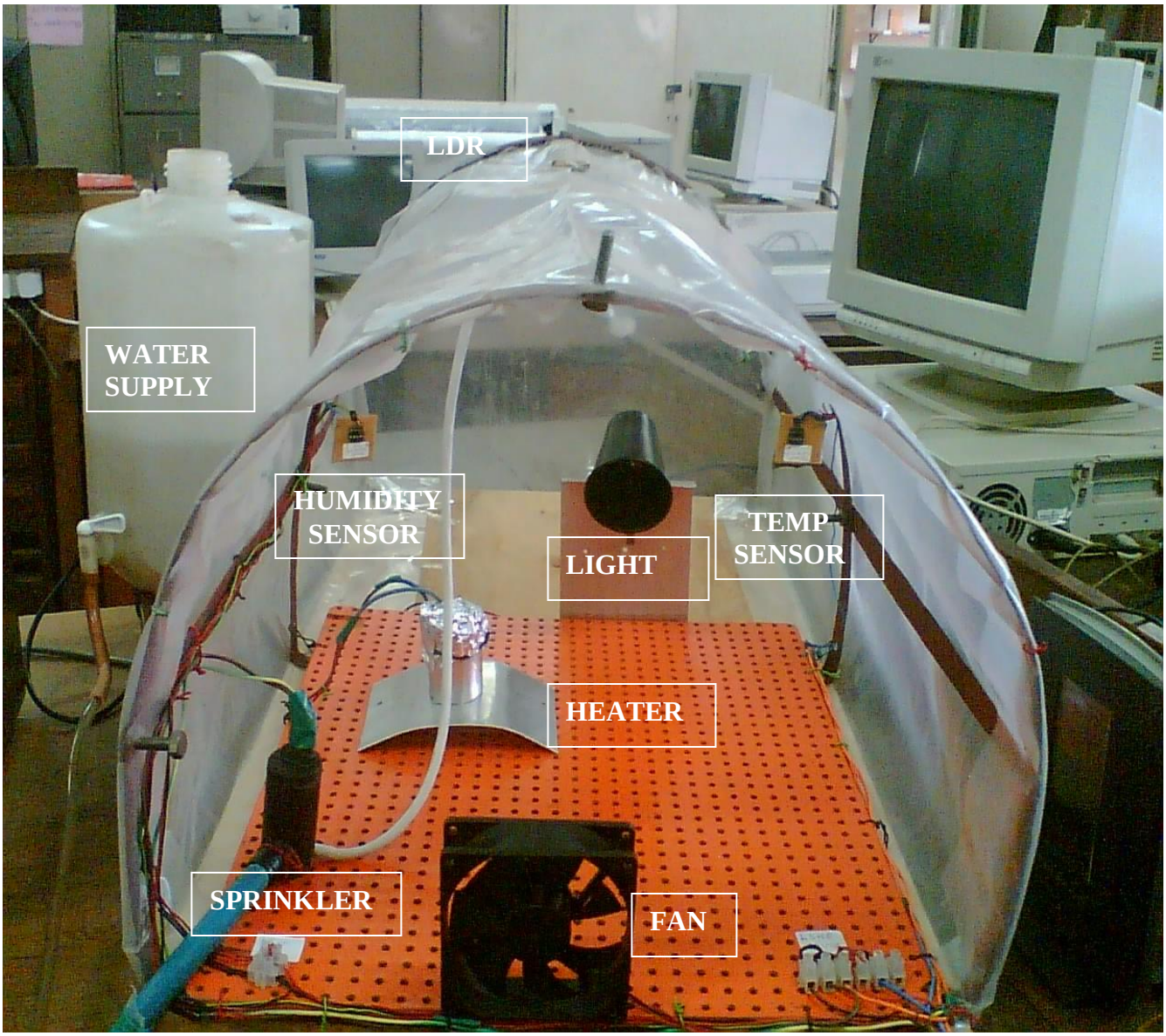


Complete circuit diagram

6 PLATE 1: THE CONTROL SYSTEM



7 PLATE 2: THE GREENHOUSE STRUCTURE



8 PLATE 3: THE COMPLETE GREENHOUSE SYSTEM