



NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Electrical and Electronic
Engineering

Elevator Prototype Using
Microcontrollers

Student: Khaled Walid A'amar (20020911)

Supervisor: Assoc. Prof. Dr. Adnan Khashman

Lefkoşa - 2006

ACKNOWLEDGMENTS

My utmost thanks to my Lord Allah that I could complete my graduation project. I could not have prepared this project without the generous help of my supervisor, colleagues, friends, and family.

First, I would like to thank my supervisor Assoc. Prof. Dr. Adnan Khashman for his invaluable advice, and belief in my work and myself over all the courses of this Degree. Assoc. Prof. Dr. Adnan Khashman supplied the warmth, enthusiasm, and clarity of judgement that every student hopes for. Going beyond the limited role of literary agent, he provided valuable advice at each stage of the preparation of this project.

I would like to express my gratitude to Prof. Dr. Dogan Ibrahim for his help and support, and to Prof. Dr. Fakhraddin Mamedov for him because he provided valuable advice at each stage of the preparation of this project also.

I will never forget the help that i got from this university for continueing my education especially from Prof. Dr Şenol Bektaş, so my regards and my love to him.

My depest thanks are to my family. I could never have prepared this project without the encouragement and support of my parents, brothers and sister.

The root of this success lies under the most affectionate wish of my loving FATHER. I am grateful to him to assist me to grow in knowledge. I salute you, my father.

I would also like to thank all my friends for their help and for their patience.

ABSTRACT

As the life is getting more complicated, every one in *this* world searches for the comfortable life, thus in the modern life the buildings are getting longer, which invite us to use the elevators, the elevators are controlled by PIC microcontrollers where they are replaced with the old wired control systems.

Actually, the elevators are the solution of carrying heaviness over long heights, therefore the need increased recently for these elevators, so the incremental demands made this project one of the optimum business alternatives.

This project represents the elevator prototype that is controlled by using PIC microcontroller, this graduation project consists software and hardware sides, whereas the software side includes microBasic language, compiler is used to compile the instructions for the typed program and respectively the IC prog is going to send the data to the programmer.

The elevator consists of three storeys and it has a bush buttons as callers thus the stepper motor are rotating up or down according to the place of the elevator's room so *this* mechanism needs a specific time.

The fully detailed explanations are included about the microBasic language and its compiler; the electronic components are explained as well.

INTRODUCTION

The elevator is a very important auxiliary tool in the modern life where the long buildings are existed, so to be familiar with this application I have chosen this project which considered as a good starting point for successful business, thus the modern programmed elevators use PIC microcontrollers.

The uncountable advantages were the cause of preference it against the wired control systems and it took that ease during the control applications by the possibility of varying and fixing the action from the software program.

Fundamental problems related with hardware and software implementation are considered, a fully detailed explanation have provided in the chapters as following:

The first chapter represents classification of electronic components which used in the elevator prototype project, the data sheets, pieces shapes and their functions were included into this chapter.

Chapter two is devoted to the microcontrollers generally whereas the PIC16F877A was specified because of its adjectives which give it impartiality existence in the manufacture control fields.

Chapter three presents the information about the microBasic language and its compiler which convert the instructions among the user and the programmer.

Chapter four is devoted to the programmer and its construction with a declaration of the PIC's that can be placed upon the Ziff socket.

Chapter five talks about the elevator prototype using microcontrollers' project, circuit diagrams are explained with their functions, thus the analyses of the circuits are followed respectively.

The conclusion presents important results and notifications about the elevator prototype project by the author of the thesis and practical realization of the elevator prototype using microcontrollers' project.

TABLE OF CONTENTS

ACKNOWLEDGMENT	i
ABSTRACT	ü
INTRODUCTION	ili
TABLE OF CONTENTS	iv
1. ELECTRONIC COMPONENTS	1
1.1. Overview	1
1.2. Components	1
1.2.1. Resistors	1
1.2.2. Capacitors	3
1.2.2.1. Capacity	3
1.2.2.2. Capacitor and DC voltage	4
1.2.2.3. Capacitors and AC voltage	4
1.2.3. Semiconductor	4
1.2.3.1. Diodes	5
1.2.3.2. Led	6
1.2.3.3. Transistors	7
1.2.4. Switches	9
1.2.4.1. (ON)-OFF Push-to-make	9
1.2.4.2. ON-(OFF) Push-to-break	9
1.2.5. Solenoid lock	10
1.2.6. Stepper Motor	10
1.2.7. L293D	11
1.3. Safety Guidelines	12
1.4. Summary	12
2. MICROCONTROLLERS	13
2.1. Overview	13
2.2. What is a Microcontroller?	13
2.3. General Characteristic of Microcontrollers	
2.4. The Advantages of Microcontroller	17

2.4.1. Coder Efficiency	17
2.4.2. Reliability	17
2.4.3. Speed	17
2.4.4. Static Operation	18
2.4.5. Drive Capability	18
2.4.6. Several Options	18
2.4.7. Security	18
2.4.8. Versatility	18
2.4.9. Development Tools	19
2.5. PIC16F877A Features	19
2.5.1. General Features	19
2.5.2. Special Microcontroller Features	19
2.5.3. Peripheral Features	20
2.5.4. Analog Features	20
2.6. Microchip Families	21
2.7. Memory Types	21
2.8. Core Architecture of PIC Microcontrollers	
2.9. Summary	24
3. MICROBASIC LANGUAGE	25
3.1. Overview	25
3.2. Advantages of MicroBasic	25
3.3. Basic Editor Features	
3.4. Creating First Project	27
3.5. Error Window	32
3.6. Assembly View	32
3.7. Statistics	33
3.7.1. Memory Usage Window	33
3.7.2. Procedures (Graph) Window	34
3.7.3. Procedures (Locations) Window	34
3.7.4. Procedures (Details) Window	35
3.7.5. RAM Window	35

3.7.6. ROM Window	36
3.8. Identifiers	36
3.8.1. Rules	36
3.8.2. Note	37
3.8.3. Examples	37
3.9. Keywords	38
3.10. Data Types	39
3.10.1. Simple	39
3.10.2. Structured	39
3.10.3. Sign	40
3.10.4. Array	40
3.10.4.1. Array and Operators	41
3.10.4.2. Array and PIC	41
3.10.5. Strings	43
3.10.5.1. Strings and Assignment	43
3.10.5.2. Length	44
3.12. The Loops	45
3.12.1. Do..Loop Until Statement	46
3.12.2. While Statement	47
3.12.3. For Statement	48
3.13. Summary	49
4. THE PIC PROGRAMMER	50
4.1. Overview	50
4.2. The Characteristics of PIC Microcontroller	
4.3. Supported Microcontrollers	51
4.4. Setting the IC Prog software	52
4.5. Troubleshooting	53
4.6. Common Errors	55
4.6.1. Privileged Instruction	55
4.6.2. Varify Failed	55
4.7. Summary	55

5.	Elevator Prototype Using Microcontroller	56
5.1.	Overview	56
5.2.	The Project Description	56
5.2.1.	The Electronic Parts	57
5.2.2.	The Circuit Connections	58
5.3.	The Circuit Analysis	58
5.3.1.	The Power Circuit	59
5.3.2.	The PIC16F877 A Related Circuit	59
5.3.3.	The I/O Ports Connections	60
5.3.3.1.	Leds Circuit Connection	60
5.3.3.2.	Push Button Connection	60
5.3.3.3.	Solenoid Locks Connection	60
5.3.3.4.	Stepper motor connection	61
5.4.	The Elevator's microBasic Program	62
5.6.	Summary	62
.....		
	CONCLUSION	63
	REFERENCES	64
	APPENDIX	65

CHAPTER ONE

ELECTRONIC COMPONENTS

1.1 Overview

This chapter presents an introduction to electronic components that are commonly used in hardware projects. Safety guidelines for electronic projects will also be described.

1.2 Components

In this section a detailed explanation will be given for each hardware component used in setting up the electronic circuit.

1.2.1 Resistors

Resistors are electronic components used extensively on the circuit boards of electronic equipment. Resistors are usually used to limit current.

They are color coded with stripes to reveal their resistance value (in ohms) as well as their manufacturing tolerance. Resistors, like diodes and relays, are another of the electrical components that should have a section in the installer's parts bin. They have become a necessity for the mobile electronics installer, whether it is for door locks, timing circuits, remote starts, or just to discharge a stiffening capacitor.

Resistors are components that resist the flow of electrical current in the case of higher value of resistance is located (measured in ohms) then the lower current will be measured.

Resistors are color coded to read the color code of a common 4 band 1K ohm resistor with a 5% tolerance, start at the opposite side of the GOLD tolerance band and read from left to right. Write down the corresponding number from the color chart below for the 1st color band BROWN. To the right of that number, the corresponding

number should be written for the 2nd band BLACK. Now that number should be multiplied (it should have 10) by the corresponding multiplier number of the 3rd band (RED) (100). Your answer will be 1000 or 1K. As shown in figure 1.1.

If a resistor has 5 color bands, the corresponding number of the 3rd band has to be written to the right of the 2nd before multiplying it by the corresponding number of the multiplier band. If only 4 color bands that include a tolerance band, this column must be ignored and gone straight to the multiplier.

The tolerance band is usually gold or silver, but some may have none. Because resistors are not the exact value as indicated by the color bands, manufactures have included a tolerance color band to indicate the accuracy of the resistor. Gold band indicates the resistor is within 5% of what is indicated. Silver = 10% and None = 20%. The 1K ohm resistor in the example below, may have an actual measurement anywhere from 950 ohms to 1050 ohms. If a resistor does not have a tolerance band, start from the band closest to a lead. This will be the 1st band. If the color bands are unable to be read then the multimeter has to be used.

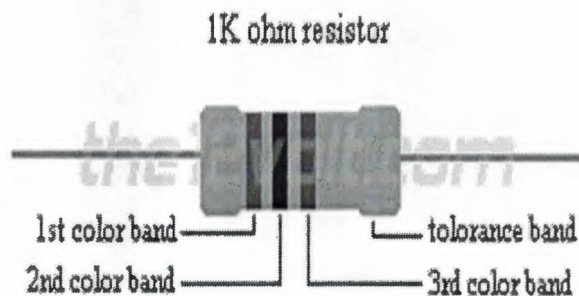


Figure 1.1 Resistor [11]

1.2.2 Capacitors

A capacitor is an electronic device which consists of two plates (electrically conductive material) separated by an insulator. The capacitor's value (its 'capacitance') is largely determined by the total surface area of the plates and the distance between the plates (determined by the insulator's thickness).

A capacitor's value is commonly referred to in microfarads, one millionth of a farad. It is expressed in micro farads because the farad is such a large amount of capacitance that it would be impractical to use in most situations.

1.2.2.1 Capacity

In the following diagram (Figure 1.2), 2 tanks are seen (capacitors) of different diameter (different capacitance). It should be readily understood that the larger tank can hold more water (if they're filling to the same level (voltage)). The larger capacitor has more area in which to store water. Just as the larger capacitor's larger plate area would be able to hold more electrons.

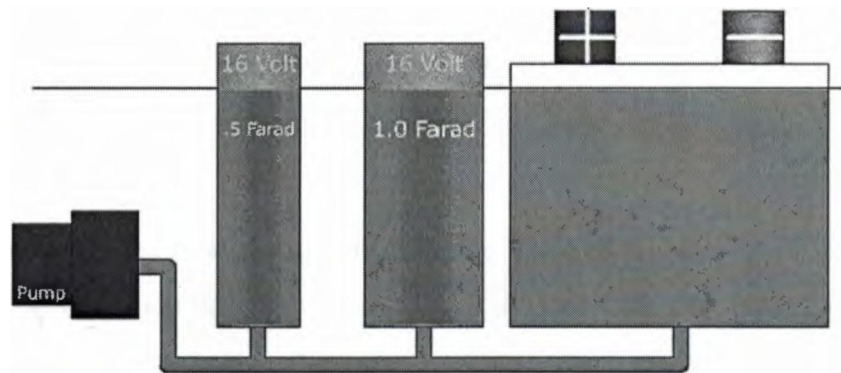


Figure 1.2 Capacities [8]

1.2.2.2 Capacitor and DC voltage

When a DC voltage source is applied to a capacitor there is an initial surge of current, when the voltage across the terminals of the capacitor is equal to the applied voltage, the current flow stops. When the current stops flowing from the power supply to the capacitor, the capacitor is 'charged'. If the DC source is removed from the capacitor, the capacitor will retain a voltage across its terminals (it will remain charged). The capacitor can be discharged by touching the capacitor's external leads together. When using very large capacitors (1/2 farad or more) in a car, the capacitor partially discharges into the amplifier's power supply when the voltage from the alternator or battery starts to fall. The discharge is only for a fraction of a second. The capacitor can not act like a battery. It only serves to fill in what would otherwise be very small dips in the supply voltage .

1.2.2.3 Capacitors and AC voltage

Generally, if an AC voltage source is connected to a capacitor, the current will flow through the capacitor until the source is removed. There are exceptions to this situation and the A.C. current flow through any capacitor is dependent on the frequency of the applied A.C. signal and the value of the capacitor.

1.2.3 Semiconductor

Semiconductor has a large amount of types. Transistors have three lead-out wires are called the base, emitter and conductor. It is essential that these are connected correctly, as there is no chance of project working if they are not. Fortunately modern transistors are not easily damaged, and incorrect connection is not likely to damage a device (or other components in the circuit) only one type is used in this project to drive a required voltage to a solenoid locks.

1.2.3.1 Diodes

Diodes are non-linear circuit elements. It is made of two different types of semiconductors right next to each other. Qualitatively we can just think of an ideal diode as having two regions: a conduction region of zero resistance and an infinite resistance non-conduction region. For many circuit applications, the behavior of a (junction) diode depends on its polarity in the circuit. If the diode is reverse biased (positive potential on N-type material) the current through the diode is very small. The following figures show the characteristic of diode.

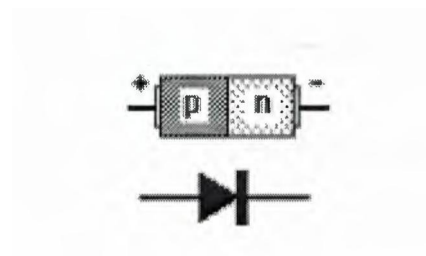


Figure 1.3 Diode [12]

- **Forward Biased P-N Junction:** forward biasing the p-n junction drives holes to the junction from the p material and electrons to the junction from the n-type material. At the junction the electrons and holes combine so that a continuous current can be maintained.

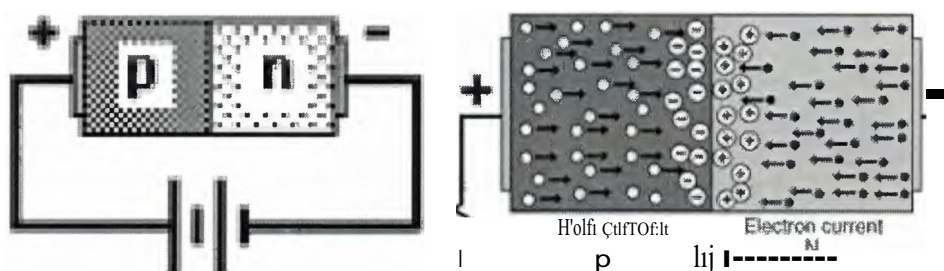


Figure 1.4 Forward Biased P-N Junction [12]

- Reverse Biased P-N Junction: the application of a reverse voltage to the p-n junction will cause a transient current to flow as both electrons and holes are pulled away from the junction. When the potential formed by the widened depletion layer equals the applied voltage, the current will cease except for the small thermal current

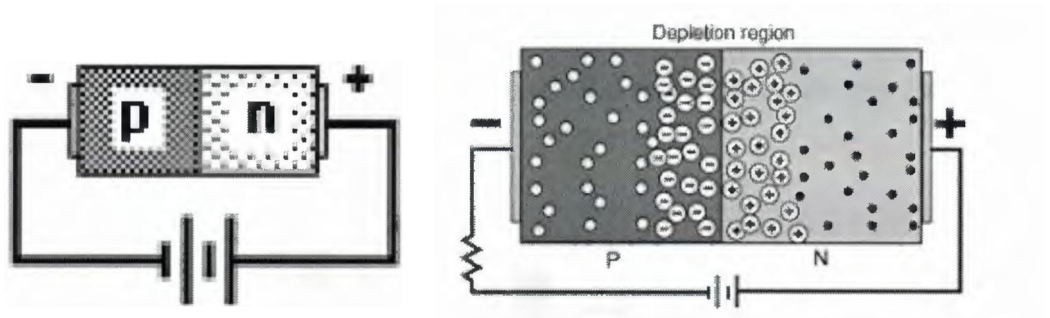


Figure 1.5 Reverse Biased P-N Junction [12]

1.2.3.2 Led

The leds are used in our project as different colors tell that which level of elevator that we are in. Thus each color has its own threshold voltage, so several values of resistors should be connected to protect leds, the led connection appears in figure 1.6.

LED connection

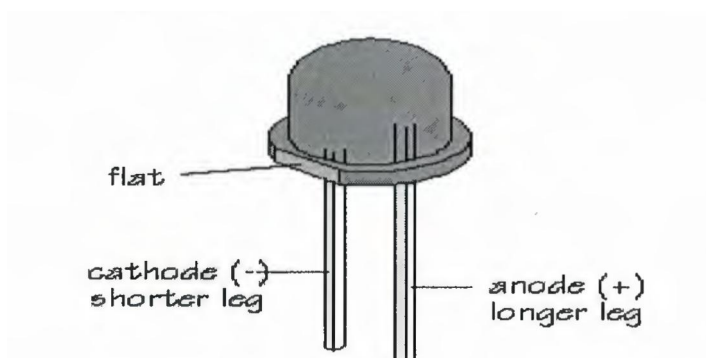


Figure 1.6 LED Connections [6]

1.2.3.3 Transistors

A Bipolar Transistor essentially consists of a pair of PN Junction Diodes that are joined back-to-back. This forms a sort of a sandwich where one kind of semiconductor is placed in-between two others. There are therefore two kinds of bipolar sandwich, the NPN and PNP varieties. The three layers of the sandwich are conventionally called the Collector, Base, and Emitter. The reasons for these names will become clear later once we see how the transistor works. As shown in the figure 1.7 there are two symbol of type of bipolar transistors.

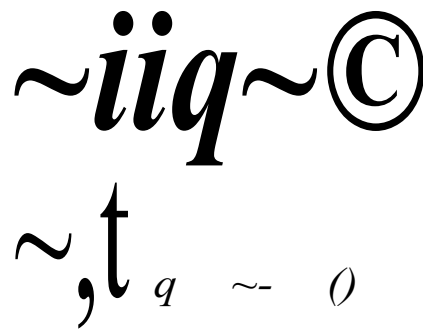


Figure 1.7 Symbol of NPN and PNP transistors [6]

Some of the basic properties exhibited by a Bipolar Transistor are immediately recognizable as being diode-like. However, when the 'filling' of the sandwich is fairly thin some interesting effects become possible that allow us to use the Transistor as an amplifier or a switch. To see how the Bipolar Transistor works we can concentrate on The NPN variety. The figure 4.8 shows the energy levels in an NPN transistor.

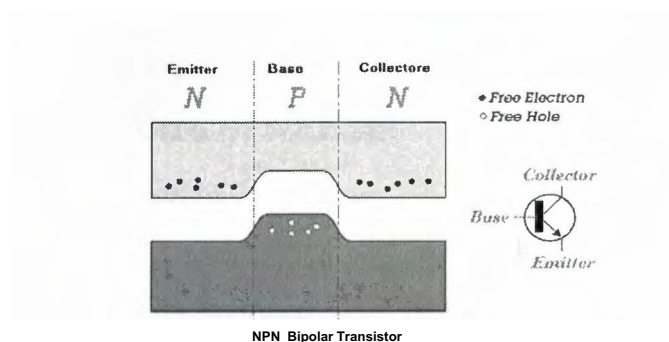


Figure 1.8 the energy levels in an NPN transistor [6]

Figure 1.8 shows the energy levels in an NPN transistor when we aren't externally applying any voltages. We can see that the arrangement looks like a back-to-back pair of PN Diode junctions with a thin P-type filling between two N-type slices of 'bread'. In each of the N-type layers conduction can take place by the free movement of electrons in the conduction band. In the P-type (filling) layer conduction can take place by the movement of the free holes in the valence band. However, in the absence of any externally applied electric field, we find that depletion zones form at both PN-Junctions, so no charge wants to move from one layer to another.

Consider now what happens when we apply a moderate voltage between the Collector and Base parts of the transistor. The polarity of the applied voltage is chosen to increase the force pulling the N-type electrons and P-type holes apart. (I.e. we make the Collector positive with respect to the Base.) This widens the depletion zone between the Collector and base and so no current will flow. In effect we have reverse-biased the Base-Collector diode junction. The precise value of the Base-Collector voltage we choose doesn't really matter to what happens provided we don't make it too big and blow up the transistor! So for the sake of example we can imagine applying a 10 Volt Base-Collector voltage. As shown in the figure 1.9 the applying collector-base voltage.

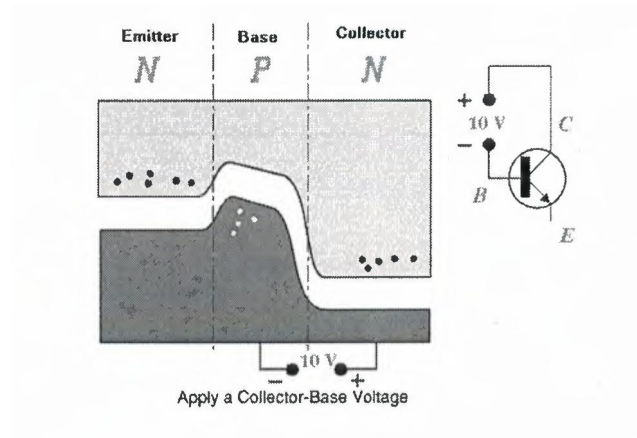


Figure 1.9 the applying collector-base voltage [6]

1.2.4 Switches

1.2.4.1 (ON)-OFF Push-to-make

A push-to-make switch returns to its normally open (OFF) position when the button is released, this is shown by the brackets around ON. This is the standard doorbell switch, show figure 4.10.

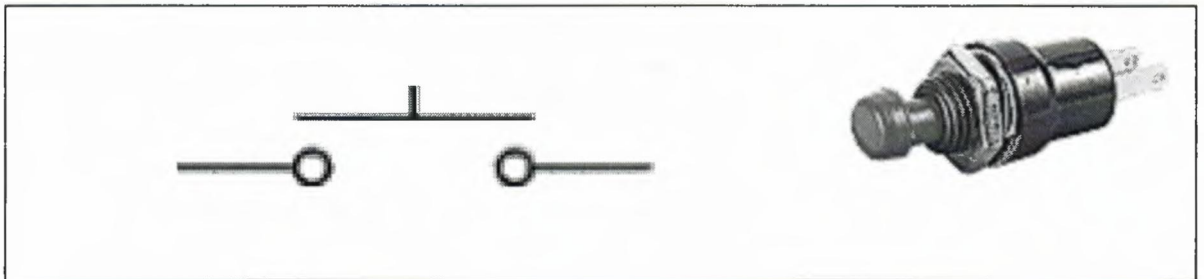


Figure 1.10 Push-to-make switches [7]

1.2.4.2 ON-(OFF) Push-to-break

A push-to-break switch returns to its normally closed (ON) position when the button is released, show figure 1.11.

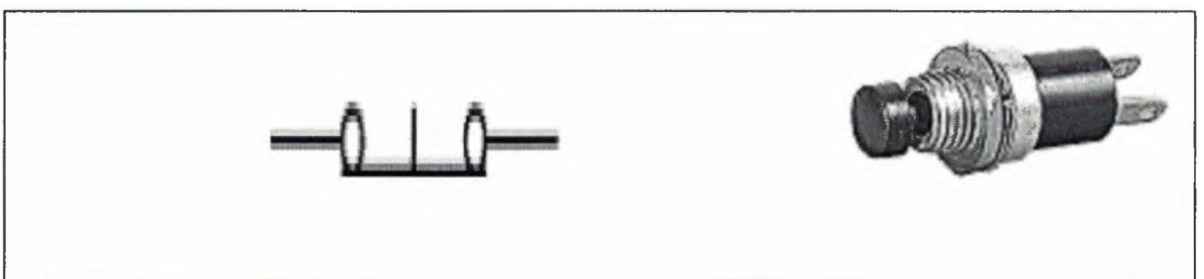


Figure 1.11 Push-to-break switches [7]

1.2.5 Solenoid lock

This solenoid locks have been connected to the circuit for adapting the case of doors during the movement of elevator's box, thus in the case of affecting by electrical pulse, it going to be open immediately, while normally they are closed.

The figure 1.12 shows the shape of solenoid lock which contains a coil affecting magneto motive force upon a central bar.



Figure 1.12 Solenoid Lock

1.2.6 Stepper Motor

Stepper motors operate differently from DC motors. When power is applied to a DC motor, the rotor begins turning smoothly. Speed is measured in revolutions per minute (RPM) and is a function of voltage, current, and load on the motor. The precise positioning of the motor's rotor is not usually possible or desirable. A stepper motor, on the other hand, runs on a controlled sequence of electric pulses to the windings of the motor. Each pulse rotates the stepper motor's rotor by a precise increment. Each increment of the rotor is referred to as a step, hence the name stepper motors. The figure 1.13 shows the construction of stepper motor. [1]

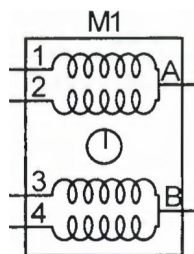


Figure 1.13 Internal construction of stepper motor

1.2.7 L293D

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoids, DC and stepping motors) and switching power transistors.

To simplify use as two bridges each pair of channels is equipped with an enable input, a separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 kHz, the L293D is assembled in a 16 lead plastic package which has 4 center pins connected together and used for heat sinking. The L293DD is assembled in a 20 lead surface mount which has 8 center pins connected together and used for heat sinking.

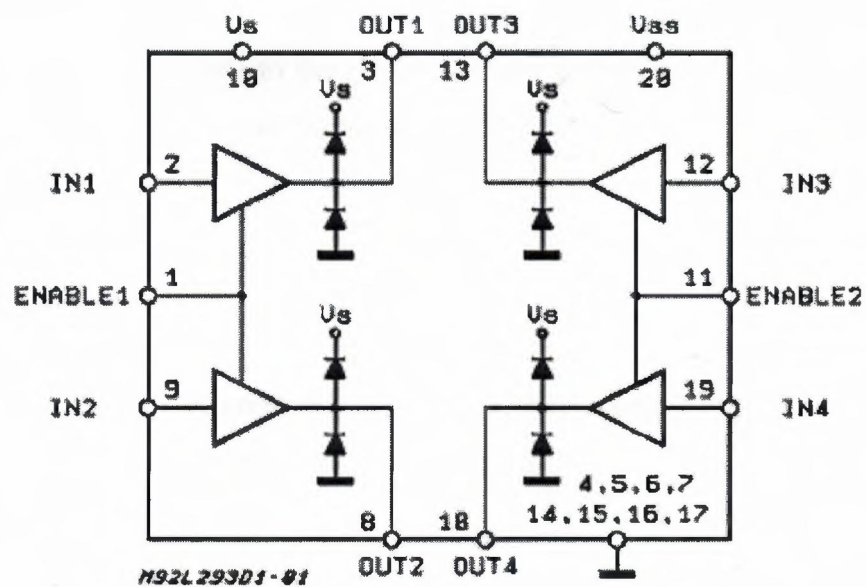


Figure 1.14 L293D block diagram

1.3 Safety Guidelines

In this project, low voltage applications are used. Thus, safety guidelines are not in concern of human safety but in components safety, although we cannot avoid the technical mistakes which can occur during connecting parts and soldering them to the circuit, so we have to be careful from current and heat.

- One of the components which are used in this circuit is the electrolytic capacitor, this element has two poles and when connected to the circuit we have to care about its polarity so as to avoid damaging it.
- While connecting the circuit components to the power supply we have to be aware of misconnecting its polarity to assure the safety of used components.
- While the circuit is on, avoid touching the sensitive components like the transistor, diodes to avoid interfering with the output signal.
- While soldering the parts to the circuit we have to be careful so as not to burn the parts which are sensitive and can be harmed by heat.

1.4 Summary

This chapter presented an introduction to electronic components that are commonly used in hardware projects and how they function, how they must be connected. By applying the safety guidelines.

CHAPTER TWO

MICROCONTROLLERS

2.1 Overview

This chapter presents information on microcontrollers. General characteristics of microcontrollers in addition to their advantages which gives a significant role among the control fields are presented.

2.2 What is a Microcontroller?

A microcontroller is an inexpensive single-chip computer. Single chip means that the entire computer system lies within the confines of the integrated circuit. The microcontroller existing on the encapsulated sliver of silicon has features and similarities to our standard personal computers. Primarily, the microcontroller is capable of storing and running a program, its most important feature.

The microcontroller contains a central processing unit (**CPU**), random-access memory (**RAM**), read-only memory (**ROM**), electrically erasable programmable read-only memory (**EEPROM**), input / output (**I/O**) lines, serial and parallel ports, timers, and other built-in peripherals, such as analog-to-digital (**AID**) and digital-to-analog (**DIA**) converters.

The microcontroller's ability to store and run unique programs makes it extremely versatile. For instance, a microcontroller can be programmed to make decisions and perform functions based on predetermined situations (I/O line logic) and selections its electronic circuits.

Other programs can make the microcontroller behave like a neural circuit or a fuzzy logic controller. Microcontrollers are responsible for the intelligence in most smart devices on the consumer market.

Microcontrollers are the future of electronics, so that look in any hobbyist electronics magazine and you will see articles that feature the use of microcontrollers either directly or embedded inside a circuit's design. Because of their versatility, they add a lot of power, control, and options for a small cost. It ,therefore, becomes essential that the electronics engineer or hobbyist learns to program these microcontrollers in order to maintain a level of competence and to gain the advantages that microcontrollers can provide in their own circuit designs. In addition, if you examine consumer electronics, you will find microcontrollers embedded in just about everything.

A large variety of microcontrollers exist on the market today. We will focus on a few versatile microcontroller chip called PIC chips (or PICMicro chips) from Microchip Technologies. Specifically, we will use PIC 16F877A because of its own advanced features. [1]

2.3 General Characteristic of Microcontrollers

The programmable control system and the hard-wired control system have different criterions as written in Table 2.1, thus they have differences in criterions like strategy of control, processing, control strategy adjustment, the occupied area, the cost, chronological functions assurance, repetition of control system, reliability, allocation in network and maintenance.

These comparisons between several control systems show us the significant role of microcontroller as the programming control system.

Table 2.1 Comparisons between several control systems [2]

Criterion	Hard-wired	Programmable
Strategy of control.	Elements with wires in a circuit.	Program in a circuit.
Processing.	Depends on relays and other elements.	Programmable by computer's processor.
control strategy adjustment.	Very complicated.	Easy by varying the program.
The occupied area.	Very big area.	Very small area.
The cost.	Expensive	cheap
Chronological functions assurance.	Done by timer relays.	Done by programmed chronological timers
Repetition of control system.	Needs a high effort.	Only copying the program.
Reliability.	Low because of the mechanical movement.	High because of the electronic elements usage.
Allocation in network	Very hard.	Simple.
Maintenance.	Needs a time and monetary ability.	Directly and no need for payment.

According to what we have read from the table, microcontrollers are preferred to other hard-wired control system. The difference between the microcontroller and the microprocessors that the microcontroller contains RAM, **ROM** and **EEPROM** with addition to the peripherals system communication ports beside ADCs analog to digital converters, the comparison appears in the following table (table 2.2):

Table 2.2 Comparisons between microprocessors and microcontrollers [2]

Characteristics	Microprocessor	Microcontroller
ALU.	Available	Available
PC.	Available	Available
ROM, RAM.	Not available	Available
Timers/Counters.	Not available	Available
I/O Ports.	Not available	Available
Communication Ports.	Not available	Available
ADC, PWM, Capture, Data EEPROM, LCD Driver.	Not available	Available
Addressing	Available	Available

We can express the relationship between the microprocessors and the microcontrollers as: $\text{Microcontroller} = \text{Microprocessor} + \text{some Peripherals}$

There are a lot of kinds of Microcontrollers such as: Atmel, Microchip, Siemens and Matra, whereas the most popular one is Microchip's product because of its free software availability in market over the whole world. Interestingly, the programmable logic controllers (PLCs) are basically microcontroller provided with supporting units.

$$\text{PLC} = \text{Microcontroller} + \text{supporting unit}$$

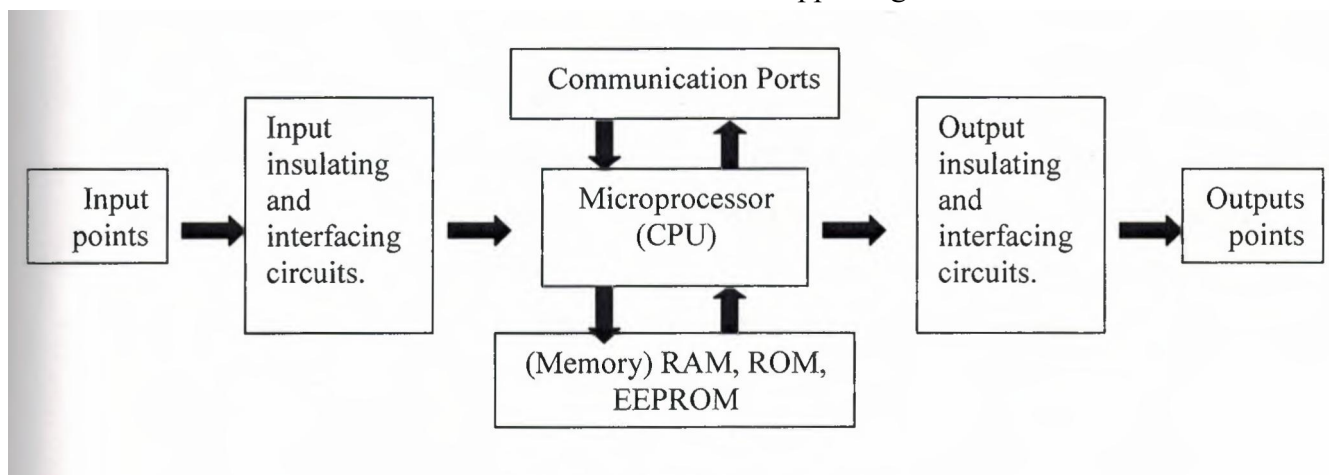


Figure 2.1 PLC diagram [2]

2.4 The Advantages of Microcontroller

The microcontrollers play a significant role in the control fields, which means they have advantages effecting positively among the industrial life.

2.4.1 Coder Efficiency

The PIC technology is the 8-bit microcontroller which depends on hard technique thus it has two separated internal paths, one of them is responsible to carry the data into data memory and the second path aimed to carry the instructions the program memory which refer to increasing the speed.

2.4.2 Reliability

The PIC microcontroller has a program word which limited to 12-bit or 14-bit so that there is no possibility to be transferred erroneously to data memory which limited to 8-bit whereas this mistake could happen within other microcontrollers.

2.4.3 Speed

To compute the speed of PIC microcontroller, the oscillator's frequency should be divided over number 4 and accordingly we will find the wanted time for the execution of one instruction which means that if we have 4MHz oscillator then $1\mu\text{s}$ is the required time for one instruction, i.e. one million instructions could be executed during one second.

The greatest number of instructions can be executed during one second is five millions (Oscillator frequency maximized to 20MHz).

2.4.4 Static Operation

The PIC microcontroller has a fully static microprocessor, that means when the PIC will be adjusted to standby case then the registers will not lose their contents.

2.4.5 Drive Capability

The PIC microcontroller could drive LED's, triacs and other components therefore it can sink 25mA for its input ports whereas it can give about 20mA for its output ports as a source.

2.4.6 Several Options

Several options are available in PIC microcontroller such as speed, temperature, package, I/O pins, serial communications, ADCs.

2.4.7 Security

The PIC microcontroller has a code protection which protects the program from any interference.

2.4.8 Versatility

The usage fields of PIC include:

- 1- Computer peripherals.
- 2- Manufacturer control systems.

- 3- Alarm and safety systems.
- 4- Telecommunication.
- 5- Management and office systems.
- 6- Several applications distributed over different fields.

2.4.9 Development Tools

The PIC microcontroller has several development tools such as the simulator and the emulator with addition to many kinds of programmers and compilers.

2.5 PIC16F877 A Features

The microcontroller has a several features causing its availability among the automation and control systems, thus microcontroller has specialty to be preferable through practical applications.

2.5.1 General Features

- Operating speed: 20 MHz, 200 ns instruction cycle
- Industrial temperature range (-40° to +85°C)
- 15 Interrupt Sources
- 35 single-word instructions
- All single-cycle instructions except for program branches (two-cycle)

2.5.2 Special Microcontroller Features

- Flash Memory: 14.3 Kbytes (8192 words)
- Data SRAM: 368 bytes
- Data EEPROM: 256 bytes
- Self-reprogrammable under software control

- In-Circuit Serial Programming via two pins (5V)
- Watchdog Timer with on-chip RC oscillator
- Programmable code protection
- Power-saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug via two pins

2.5.3 Peripheral Features

- 33 I/O pins; 5 I/O ports
- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer 1: 16-bit timer/counter with prescaler
 - o Can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
 - o 16-bit Capture input; max resolution 12.5 ns
 - o 16-bit Compare; max resolution 200 ns
 - o 10-bit PWM
- Synchronous Serial Port with two modes:
 - o SPI Master
 - o I2C Master and Slave
- USART/SCI with 9-bit address detection
- Parallel Slave Port (PSP)
 - o 8-bits wide with external RD, WR and CS controls
- Brown-Out detection circuitry for Brown-Out Reset

2.5.4 Analog Features

- 10-bit, 8-channel *A/D* Converters
- Brown-Out Reset

- Analog Comparator module
 - o 2 analog comparators
 - o Programmable on-chip voltage reference module
 - o Programmable input multiplexing from device inputs and internal VREF
 - o Comparator outputs are externally accessible

2.6 Microchip Families

There are three main families of Microchip and each kind of PIC contains special features thus our PIC16F877A which we are dealing with, belongs the Mid-range family, it consists of 40 pins and 35 instruction set. The families of Microchip are summarized as following:

Table 2.3 The families of Microchip [3]

Low level	12xxx	8-pins	33 instruction set
Mid - range	16xxx	18-44 pins	35 instruction set
<i>NIA</i>	17xxx	<i>NIA</i>	<i>NIA</i>
High - performance	18xxx	18-84 pins	77 instruction set

2.7 Memory Types

Program memory (FLASH)-for storing a written program. Since memory made in FLASH technology can be programmed and cleared more than once, it microcontroller suitable device development.

EEPROM- data memory that needs to be saved when there is no supply. It is usually used for storing important data that must not be lost if power supply suddenly stop one such data is an assigned temperature in temperature regulators. If during a loss of power was lost, we would have to make the adjustment once again upon return of supply. Thus our data self-reliance.

The read only memory relates with the model of PIC as appears in (Table 2.4) whereas our PIC16F877A contains EEPROM (electrical erasable program read only memory).

Table 2.4 The memory ROM [3]

ROM	16CRXX
EPROM	16CXX
EEPROM	16FXX (Flash)

RAM - data memory used by a program during its execution. In RAM are stored all inter-results or temporary data during run-time.

2.8 Core Architecture of PIC Microcontrollers

The core architectures of PIC16F877A microcontroller can be shown in Figure 1.2 and Figure 1.3 below, thus it contains 40-pins and accordingly each pin has its own function as it going to be fully explained in the next pages.

Figure 2.2 PIC16F877A shape [4]

According to the Figure 2.3 we denoted that each pin has its own function so that to deal with sensors and actuators several ports have been needed to response the programming demand, therefore these ports are distributed as follows:

PORTA-RA# ~ (pin2 _ pin?) = 6 pins.

PORTB-RB# ~ (pin33 _ pin40) = 8 pins.

PORTC-RC# ~ (pin23 _ pin26) + (pin16 _ pin18) = 8 pins.

PORTD-RD# ~ (pin9 _ pin22) + (pin27 _ pin30) = 8 pins.

PORTE-RE# ~ (pin8 _ pin0) = 3 pins.

All these ports can be used as inputs or outputs respectively but the important thing is to know which kind of signal (digital or analog) they can carry.

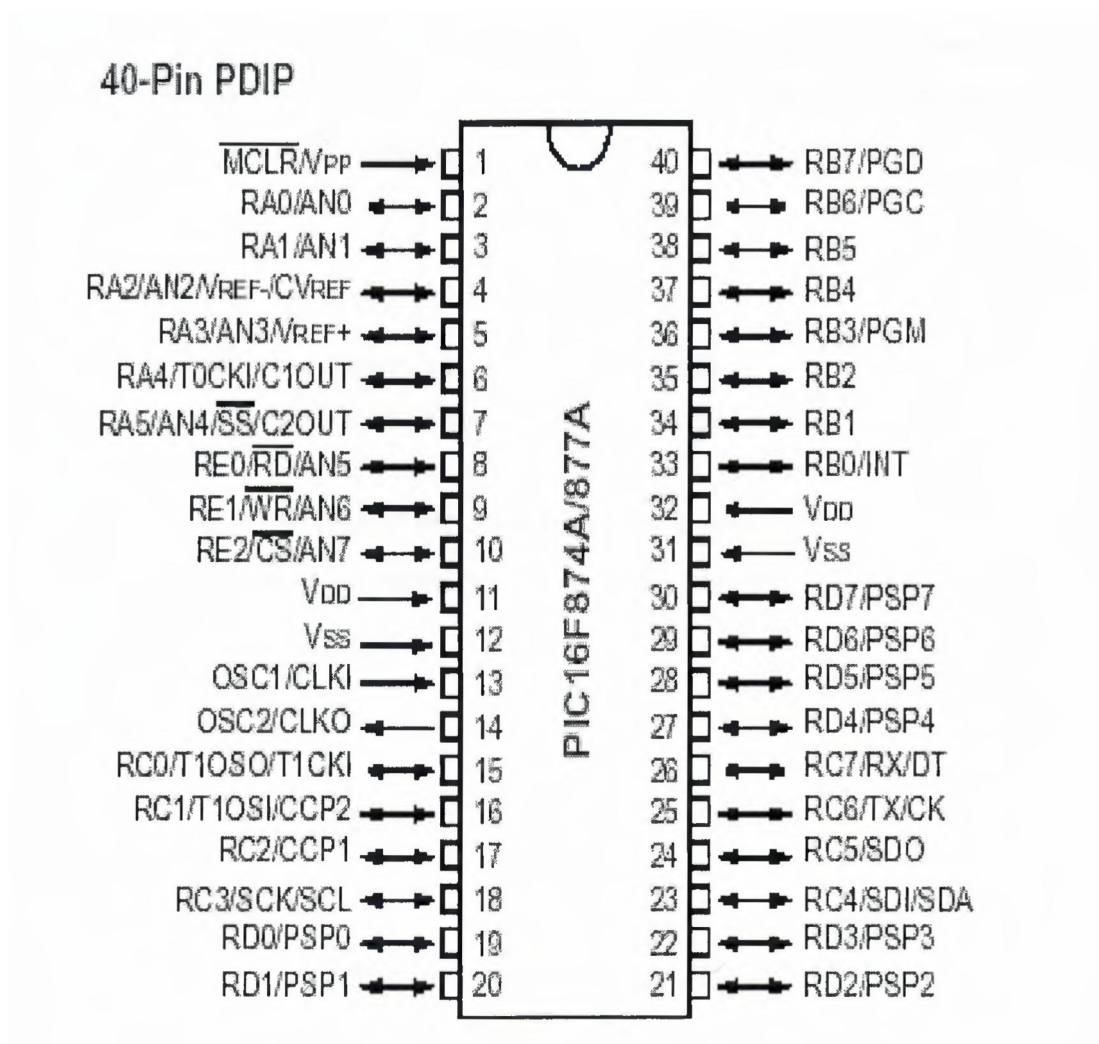


Figure 2.3 The core architecture of PIC16F877A [4]

For example:

PIN 33 is a digital port so its signal needs to be converted in the case of analog signal, whereas, pin 2 till pin 5 & pin 7 till pin 10 are analog ports.

Other pins are used either for power supplying or for another several purposes like clocking system, interruption tasks, resetting the microcontroller.

Specifically, this kind of microcontroller PIC16F877A has five ports which mean it can be applied for large control systems.

The PIC16F877A takes and supplies five volt and its current is limited to 20mA therefore in the case of connecting power electronics piece then snubbers are required to avoid the unwonted voltages values.

2.9 Summary

In this chapter, information has been fully presented about the microcontrollers, specifically, the microcontrollers of microchip which is called PIC.

PIC16F877A was the kind of PIC microcontroller that we will use in this project; it is the most suitable PIC for the elevator because of its high pins numbers distributed over different five ports.

CHAPTER THREE

MICROBASIC LANGUAGE

3.1 Overview

This chapter presents an introduction to mikroBasic language, which is the software used to program the PIC microcontroller. MikroBasic is a Windows-based Integrated Development Environment, and is much more than just Basic compiler for PIC MCUs.

3.2 Advantages of MikroBasic

With mikroBasic, you can:

1. Create Basic source code using the built-in Code Editor
2. Compile and link your source code
3. Inspect program flow and debug executable logic with Debugger
4. Monitor variables in Watch Window
5. Get error reports
6. Get detailed statistics (how compiled code utilizes PIC MCU memory, hex Map, charts and more ...)

The parts of mikroBasic window have been explained to define the function of each part, thus it appears in figure 3.1 respectively for each.

Code Editor features adjustable Syntax Highlighting, Code Assistant, Parameters Assistant, Auto Correct for common typos, and Code Templates.

Code browser, Keyboard shortcut browser, and Quick Help browser are at your disposal

For easier project management.

Error Window displays all errors detected during compiling and linking.

Watch Window enables you to monitor variables, registers and PIC MCU Memory.

New Project Wizard is fast, reliable, and easy way to create a project.

Source-level Debugger lets you debug executable logic step-by-step by watching program flow.

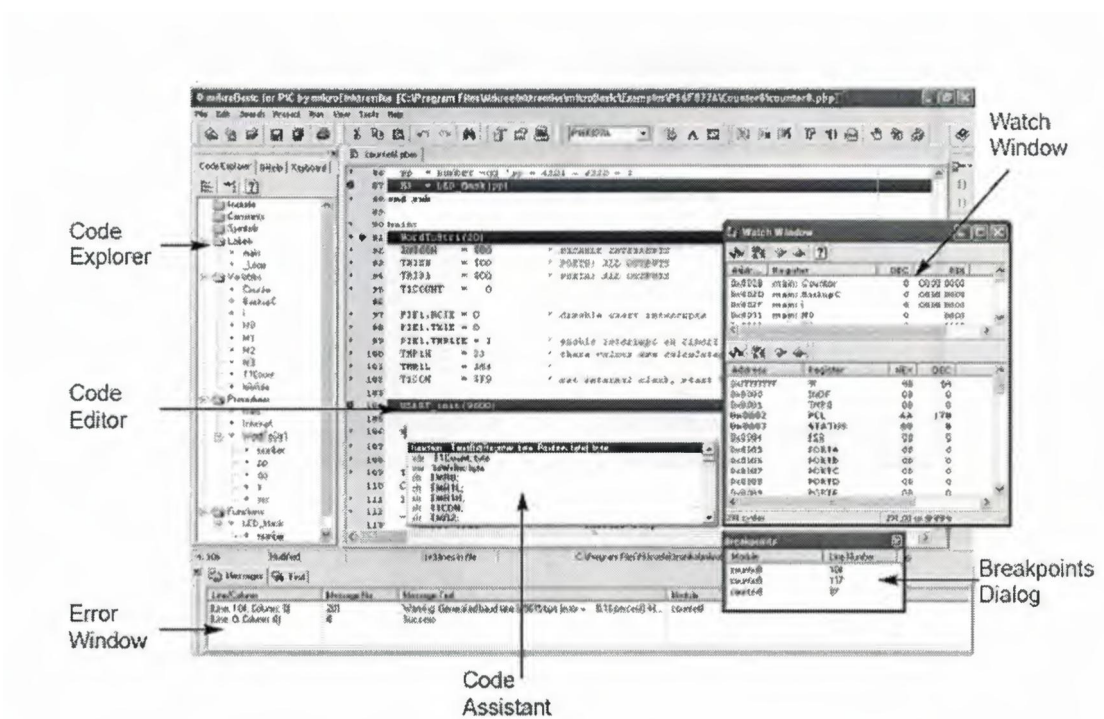


Figure 3.1 microBasic window parts [5]

3.3 Basic Editor Features

General code editing is same as working with any standard text-editor, including familiar copy, Paste, and Undo actions, common for windows environment.

Advanced code editing includes:

- Adjustable Syntax Highlighting
- Code Assistant, Parameters Assistant, Code Templates
- Auto Correct for common typos

You can configure Syntax Highlighting, Code Assistant and Auto Correct from Editor Settings dialog. To access this window, click Tools > Options from dropdown menu, or click Tools icon in Settings toolbar, show Figure 3.2 below:



Figure 3.2 Editor Settings [5]

3.4 Creating First Project

Step 1

From a drop-down menu, select: Project> New Project, or click New Project icon



Figure 3.3 Step One [5]

Step 2

Fill the New Project Wizard dialog with correct values to set up your new project.

- Select a device for your project from the drop-down menu
- Set configuration bits (Device Flags) by clicking Default push-button.

- Select Device Clock by entering appropriate value in edit box.
- Enter a name for your new project
- Enter project description edit box for closer information about your project
- Enter project path

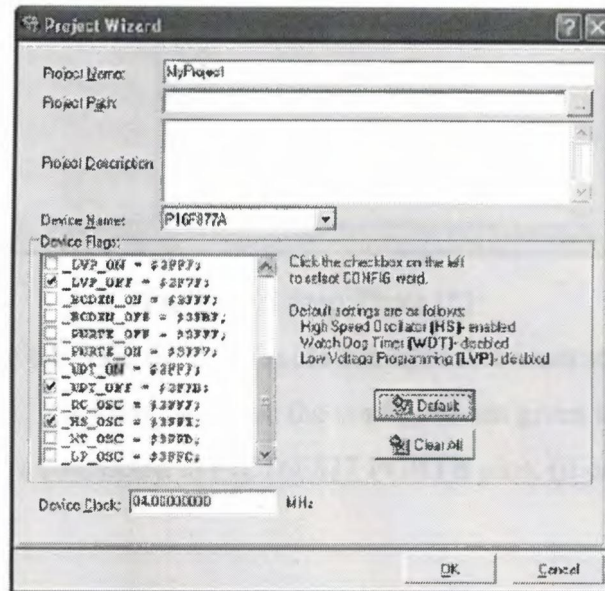


Figure 3.4 Step Two [5]

After you have set up your project, select OK push button in New Project Wizard dialog box. mikroBasic will create project for you and automatically open the program file in code editor. Now we can write the source code.

Step 3

After you have successfully created an empty project with New Project Wizard, Code Editor will display an empty program file, named same as your project.

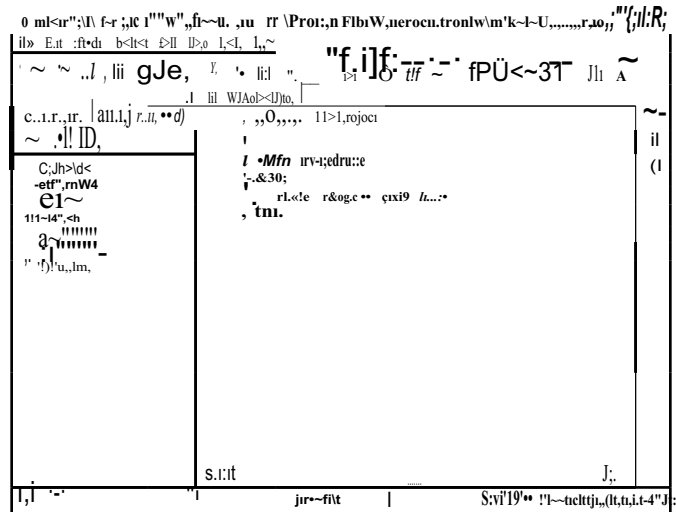


Figure 3.5 Step Three [5]

Now we can write the code for this simple example. We want to make LED diode blink once per second. Assuming we have the configuration given in the following figure, LED diodes are connected to PIC16F877 PORTB pins. (it can be any other PIC that has PORTB)

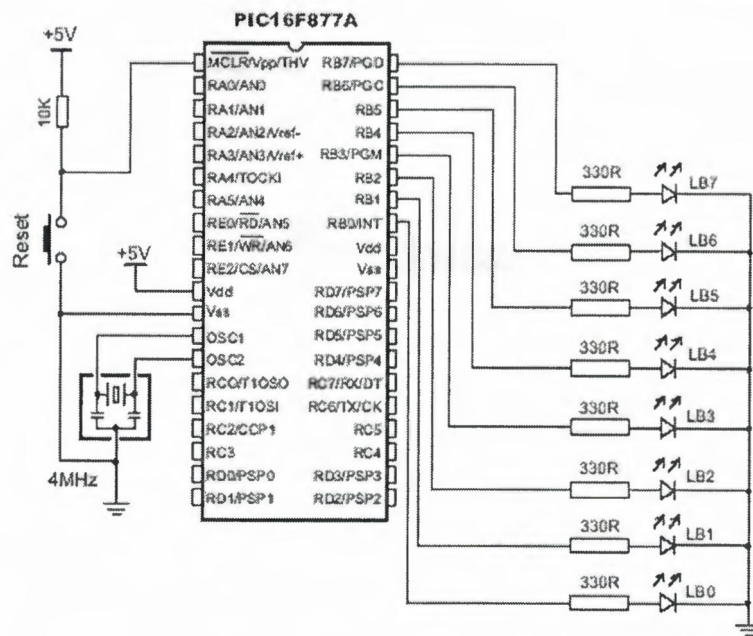


Figure 3.6 Step Three [5]

In this configuration, LED will emit light when voltage on pin is high (5V), and will be off when voltage on pin is low (0V). We have to designate PORTB pins as output, and change its value every second. Listing of program is below

program My_LED

main:

TRISB =0' configure pins of PORTB as output

eloop:

PORTB = \$FF' turn on diodes on PORTB

delay_ms(1000)' wait 1 second

PORTB =0' turn off diodes on PORTB

delay_ms(1000)wait 1 second

goto eloop ' stay in a loop

end.

Step 4

Before compiling, it is recommended to save the project (menu choice File>Save All). Now you can compile your code by selecting menu Run> Compile, or by clicking the Compile icon.

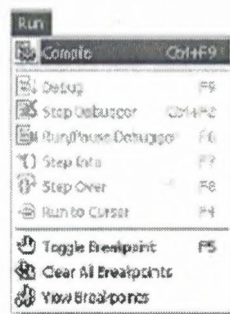


Figure 3.7 Step Four [5]

mikroBasic has generated hex file which can be used to program PIC MCU. But before that, let's check our program with the Debugger. Also mikroBasic generates list and assembly files.

Step 5

After successful compiling, we can use mikroBasic Debugger to check our program behavior before we feed it to the device (PIC16F877 or other). For a simple program such as this, simulation is not really necessary, but it is a requirement for more complex programs. To start the Debugger, select Run > Debug, or click the Debug icon, or simply hit F9.

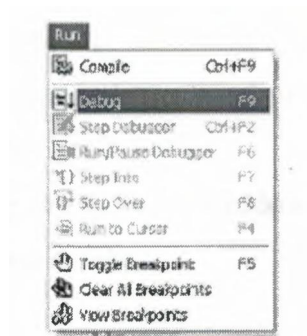


Figure 3.8 Step Five [5]

Upon starting the Debugger, Watch Window appears, and the active line in Code Editor marks the instruction to be executed next. We will set the breakpoint at line 7 by positioning the cursor to that line and toggling the breakpoint (Run> Toggle Breakpoint or F5). See the following image.

We will use the Step Over option (Run> Step Over or F8) to execute the current program line. Now, you can see the changes in variables, SFR registers, etc, in the Watch Window - items that have changed are marked red, as shown in the image below.

We could have used Run/Pause (F6) option to execute all the instructions between the active line and the breakpoint (Run > Run/Pause Debugger).

Step 6

Now we can use hex file and feed it to the device (PIC16F877 or other). In order to do so hex file must be loaded in programmer (PIC Flash by mikroElektronika or any other).

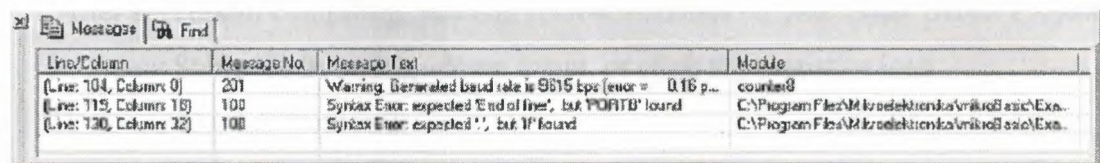
3.5 Error Window

In case that errors were encountered during compiling, compiler will report them and won't generate a hex file. Error Window will be prompted at the bottom of the main window.

Error Window is located under message tab, and displays location and type of errors compiler has encountered. Compiler also reports warnings, but these do not affect generating hex code. Only errors can interfere with generation of hex.

Double clicking the message line in Error Window results in highlighting the line of source code where the error took place.

Double clicking the message line in Error Window results in highlighting the line of source code where the error took place.



Line/Column	Message No.	Message Text	Module
(Line: 104, Column: 0)	201	Warning: Generated baud rate is 9615 bps (error = 0.16 p...	count8
(Line: 115, Column: 18)	100	Syntax Error: expected 'End of line', but 'PORTB' found	C:\Program Files\Mikrotektronika\mikroBasic\Exa...
(Line: 120, Column: 32)	100	Syntax Error: expected ',', but '}' found	C:\Program Files\Mikrotektronika\mikroBasic\Exa...

Figure 3.9 Error Window [5]

3.6 Assembly View

After compiling your program in mikroBasic, you can click toolbar Assembly icon or select Project> View Assembly from drop-down menu to review generated assembly code in a new tab window. Assembly is human readable with symbolic names. All physical addresses and other information can be found in Statistics or in list file.

If program is not compiled and there is no assembly file, starting this option will compile your code and then display assembly.


```

1  /
2  / ASM code generated by mikrotex2asmMachine for ZTC - V. 2.0.0.0
3  / Date/Time: 7/10/2004 13:09:39
4  / Info: http://www.mikrotex2asmMachine.de.yu
5  /
6  /
7  /--- procedure interrupt ---
8  interrupt:
9      MOVWF 1_STACK_2
10     SHLWF STATUS,W
11     CLRF STATUS
12     MOVWF 1_STACK_3
13     MOVF ZSR,W
14     MOVWF 1_STACK_0
15     MOVF PCLATH,W
16     MOVWF 1_STACK_4
17     CLRF INTCON
18     MOVWF 1
19     ADDWF main_global_TiCount,W
20     MOVWF main_global_TiCount
21     MOVWF 1
22     SUBWF main_global_TiCount,W
23     BTFSS STATUS,Z
24     GOTO 1_COUNTER_1
25     j_counter_0:
26     MOVWF main_global_counter_1,W
27     MOVWF main_global_counter
28     MOVWF main_global_counter,W

```

Figure 3.10 Assembly View [5]

3.7 Statistics

After successful compiling, you can review statistics on your code. Select Project ~ View Statistics from drop-down menu, or click the Statistics icon.

There are five tab windows:

3.7.1 Memory Usage Window

Provides overview of RAM and ROM memory usage in form of histogram.

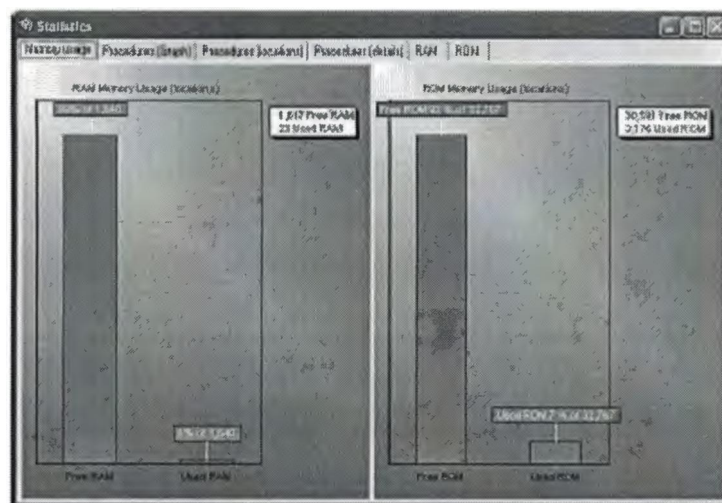


Figure 3.11 Memory Usage Windows [5]

3.7.2 Procedures (Graph) Window

Displays procedures and functions in form of histogram, according to their memory allotment.

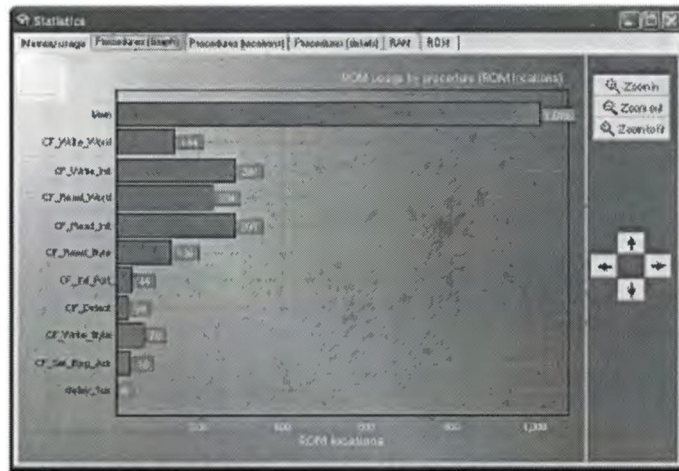


Figure 3.12 Procedures (Graph) Window [5]

3.7.3 Procedures (Locations) Window

Displays how procedures and functions are located in microcontroller's memory.

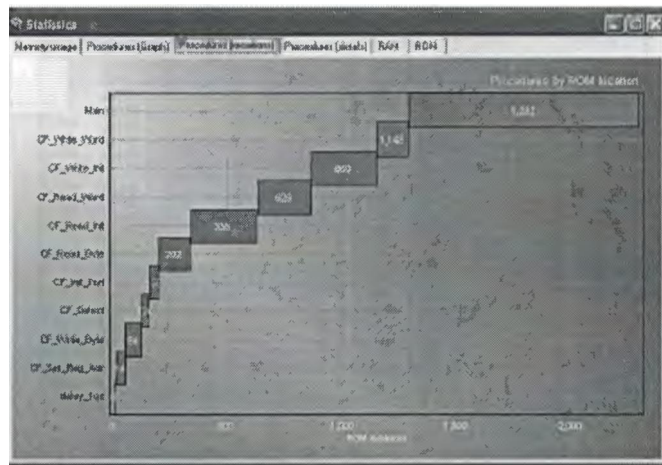


Figure 3.13 Procedures (Locations) Window [5]

3.7.4 Procedures (Details) Window

Displays complete call tree, along with details for each procedure and function: size, start and end address, frequency in program, return type, etc.

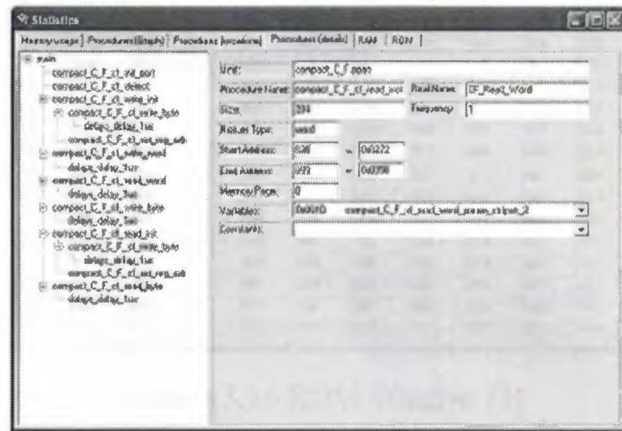


Figure 3.14 Procedures (Details) Window [5]

3.7.5 RAM Window

Summarizes all GPR and SFR registers and their addresses. Also displays symbolic names of variables and their addresses.

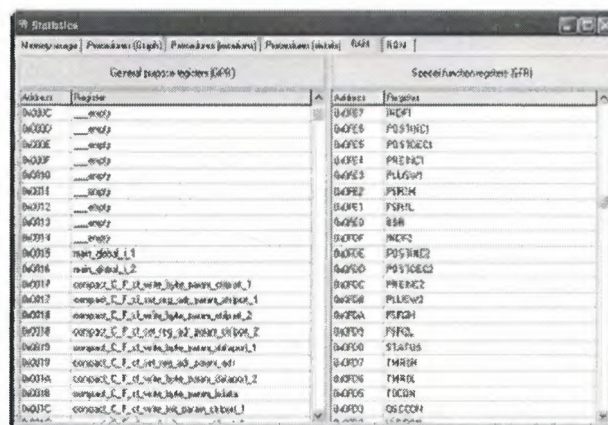


Figure 3.15 RAM Window [5]

3.7.6 ROM Window

Lists op-codes and their addresses in form of a human readable hex code.

Memory (range)		Procedure (address)		Procedure (address)		RAM		ROM	
Address	Op-code	Address	Op-code	Address	Op-code	Address	Op-code	Address	Op-code
0010	FF86	0010	FF86	0010	FF86	0010	FF86	0010	FF86
0020	C018	0020	FFEA	0020	FFEA	0020	FFEA	0020	FFEA
0030	C019	0030	FF89	0030	FF89	0030	FF89	0030	FF89
0040	C01A	0040	FFEA	0040	FFEA	0040	FFEA	0040	FFEA
0050	0A00	0050	1C00	0050	1C00	0050	1C00	0050	1C00
0060	F000	0060	C017	0060	C017	0060	C017	0060	C017
0070	F000	0070	0010	0070	0010	0070	0010	0070	0010
0080	2A00	0080	0C00	0080	0C00	0080	0C00	0080	0C00
0090	C01C	0090	FF89	0090	FF89	0090	FF89	0090	FF89
00A0	00EF	00A0	C01E	00A0	C01E	00A0	C01E	00A0	C01E
00B0	0012	00B0	C01E	00B0	C01E	00B0	C01E	00B0	C01E
00C0	001F	00C0	0C01	00C0	0C01	00C0	0C01	00C0	0C01
00D0	0A00	00D0	0C00	00D0	0C00	00D0	0C00	00D0	0C00
00E0	FF89	00E0	C01B	00E0	C01B	00E0	C01B	00E0	C01B
00F0	00EF	00F0	0C04	00F0	0C04	00F0	0C04	00F0	0C04
0100	C01C	0100	FF89	0100	FF89	0100	FF89	0100	FF89

Figure 3.16 ROM Window [5]

3.8 Identifiers

Identifiers are names used for referencing the stored values, such as variables and constants. Every program, procedure, and function must be identified (hence the term) by an identifier.

3.8.1 Rules

Valid identifier:

1. must begin with a letter of English alphabet or possibly the underscore **U**
2. can be followed by alphanumeric characters and the underscore **U**
3. may not contain special characters:

- ! @ # \$ % & * () + ' - = { } [] : " ; | < > ? ' . / \

mikroBasic is not case sensitive. First, FIRST, and flrST are an equivalent identifier.

3.8.2 Note

Elements ignored by the compiler include spaces, new lines, and tabs. All these elements are collectively known as the white space. White space serves only to make the code more legible; it does not affect the actual compiling.

Several identifiers are reserved in mikroBasic - you cannot use them as your own identifiers. Please refer to Keywords. Also, mikroBasic has several pre-defined identifiers. Pre-defined identifiers are listed in the chapter Library Functions and Procedures.

3.8.3 Examples

' Valid identifier examples

temperature_VI

Pressure

no hit

dat

' Some invalid identifier examples

7temp ' cannot begin with a numeral

%higher ' cannot contain special characters

xor ' cannot match reserved word

j23.07.04' cannot contain special characters

3.9 Keywords

The following keywords (reserved words) cannot be redefined or used as identifiers.

Table3.1 Keywords [5]

asoline	aos
ana	array
asm	begin
ooolean	case
clear	cnr
clear	case:
aiV	do
ooole	else
ena	exit
ror	runcuon
goto	gosuo
n	in
mt	interrupt
is	loop
moo	new
next	not
or	print
procedure	program
nam	read
selec1	step
sumg	swncn
men	to
module	unni
inciuoe	aim
wena	while
wim	xor

In mikroBasic, all SFR (Special Function Registers) are defined as global variables and represent special reserved words that cannot be redefined. For example - TMRO,PCL, STATUS, etc.

Also, mikroBasic has a number of predefined identifiers (refer to Library Routines). These can be replaced by your own definitions, but that would impede the functionality of mikroBasic.

3.10 Data Types

Type determines the allowed range of values for variable, and which operations may be performed on it. It also determines the amount of memory used for one instance of that variable.

3.10.1 Simple

To deal with programming within memory, the range of instructions should be known.

Table 3.2 Simple (5]

Type	Size	Range of values
byte	8-bit	0 .. 255
char*	8-bit	0 .. 255
word	16-bit	0 .. 65535
short	8-bit	-128 .. 127
Integer	16-bit	-32768 .. 32767
longint	32-bit	-2147483648 .. 147483647

3.10.2 Structured

Array represents an indexed collection of elements of the same type, often called the base type. Base type can be any simple type.

String represents a sequence of characters. It is an array that holds characters and the first element of string holds the number of characters (max number is 255).

3.10.3 Sign

Sign is important attribute of data types, and affects the way variable is treated by the compiler.

Unsigned can hold only positive numbers:

byte 0 .. 255

word 0 .. 65535

Signed can hold both positive and negative numbers:

short -128 .. 127

integer -32768 .. 32767

longint -2147483648 .. 2147483647

3.10.4 Array

Array is a set of data stored in consecutive memory locations. Defining an array and manipulating its elements is simple. Elements of array are always of same data type (any simple).

```
dim days_of_the_week as byte[7]
```

```
dim months as byte[12]
```

```
dim AD_Conversion_result as word[lü]
```

First declaration above generates 7 variables of byte type. These can be accessed by array name followed by number in the square brackets [] (this number is also known as index). Indexing is zero based, meaning that in our example, index spans numbers from 0 to 6. Instead of byte, you can define array of any other simple type (word, short, integer or longint).

Note that:

dim something as **integer[IO]**

occupies 20 RAM locations (bytes), not 10.

3.10.4.1 Array and Operators

You can use any kind of operator with array elements - Arithmetic Operators, Logical (Bitwise) Operators, and Relation (Comparison) Operators. Technically, array element is treated as a simple type. Also, instead of a number, index can be any expression with result type of byte.

For example:

$m[a + b] = 90$

$m[1] = m[2] + 67$

$m[1] = m[2] \text{ div } m[3]$

3.10.4.2 Array and PIC

When you declare an array, mikroBasic allocates a certain amount of RAM for it. Elements of array consume consecutive RAM locations; in case of array of bytes, if the address of $m[0]$ is $0x23$, $m[1]$ will be at $0x24$, and so on. Accessing these elements is almost as fast as accessing any variable of simple type. Instead of byte you can define array of any other simple type (word, short, integer or longint). Don't forget that you are restricted by the amount of free space in PIC RAM memory.

For example:

dim size as **longint[IO]**

occupies 40 RAM locations (bytes).

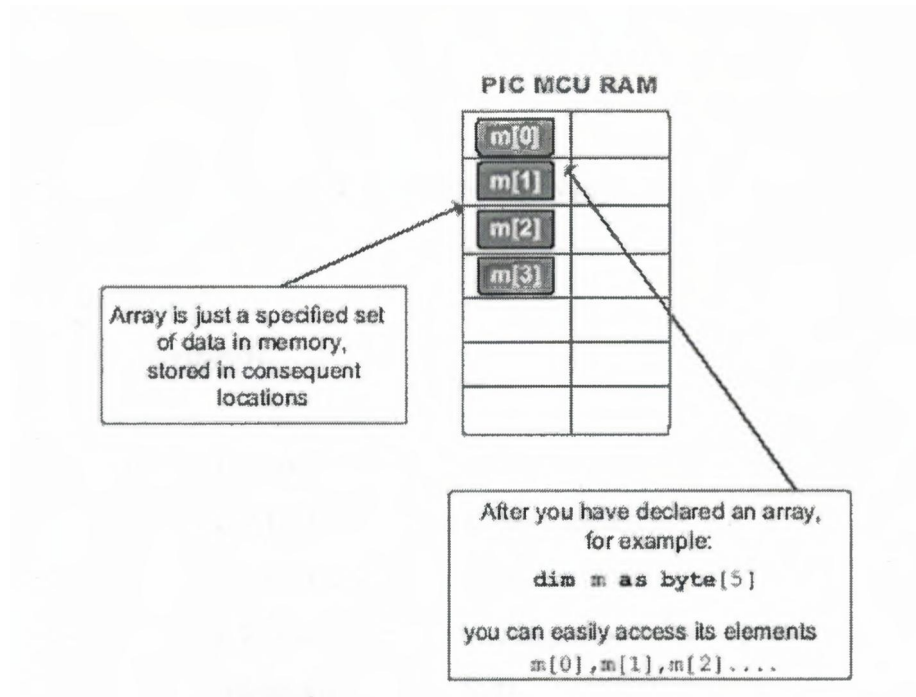


Figure 3.17 Array and PIC [5]

Example:

```

program Array_test
dim mas byte[13]
dim j as byte[5]
j[0] = m[3] + 6
m[4] = m[2] mod 3
j[2] = notj[0]
if m[0] > 0 then
m[1] = 9
else
m[1] = 90
end if
end.

```

3.10.5 Strings

String represents a sequence of characters. String type is similar to array, but can hold only characters.

```
dim M_name as string[16]
dim Start_message as string[6]
```

For each string declaration, compiler will reserve the appropriate amount of memory locations. For example, string M_name will take 16+1 locations; additional memory location is reserved to contain the length of the string.

If we assign string literal to variable M_name, M_name = "mik", then:
M_name[0] will be 3 (contains length of the string)
M_name[1] will be 'm'
M_name[2] will be 'i'
M_name[3] will be 'k'
and all other locations will be undefined.

3.10.5.1 Strings and Assignment

Assignment operator can be used with string variables:

```
dim M as string[20]
S as string[8]
main:
M = "port" ' Assign 'port' to M
S = "portl" 'Assign 'portl' to S
end.
```

3.10.5.2 Length

mikroBasic includes a built-in function Length for working with strings:

sub function Length(**dim** text as **string**) as **byte**

It returns string length as byte, and is quite useful for handling characters within string:

```
M = "mikroElektronika"
```

```
for i = 1 to Length(M)
```

```
LCD_ Chr(l,i,M[i])
```

```
nexti
```

3.11 If Statement

There are two forms of if statement:

Syntax of if. then statement is:

if expression **then**

statements

end if

where expression returns a True or False value. If expression is True, then statement is executed, otherwise it's not.

Syntax of if...Then ...Else statement is:

if expression **then**

statements 1

else

statements2

end if

Where expression returns a True or False value. If expression is True, then statements1 are executed; otherwise statements2 are executed. Statements1 and statements2 can be statements of any type.

Nested if statements require additional attention. General rule is that the nested conditionals are parsed starting from the innermost conditional, with each else bound to the nearest available if on its left.

```
if expression1 then  
if expression2 then  
statements1  
else  
statements2  
end if  
end if
```

3.12 The Loops

Loops are a specific way to control the program flow. By using loops, you can execute a sequence of statements repeatedly, with a control condition or variable to determine when the execution stops.

You can use the standard break and continue to control the flow of a do..loop until, while, or for statement. Break terminates the statement in which it occurs, while continue begins executing the next iteration of the sequence.

mikroBasic has three kinds of control loop instructions:

- DO..LOOP UNTIL statement
- WHILE statement
- FOR statement

Note that certain operations may take longer time to be executed, which can lead to undesired consequences. If you add two variables of short type and assign the result to short, it will be faster than to add two longint and assign value to longint, naturally.

Take a look at the following code :

```
dim Sa as short
dim Sb as short
dim Saaaa as longint
dim Sbbbbb as longint
for Sa = 0 to 100
  Sb = Sb + 2
next Sa
for Saaaa = 0 to 100
  Sbbbbb = Sbbbbb + 2
next Saaaa
end.
```

PIC will execute the first loop considerably faster.

3.12.1 Do..Loop Until Statement

Syntax of do..loop statement is:

do

statement I

statement N

loop until expression

where expression returns a True or False value. Do..loop statement executes statement_1 ... statement_N continually, checking the expression after each iteration. Eventually, when expression returns True, do..loop statement terminates. The sequence is executed at least once because the check takes place in the end.

Example:

```
i = 0
do
i = i + 1 ' execute these 2 statements
,PORTB = i ' until i equals 10 (ten)
loop until i = 10
```

3.12.2 While Statement

Syntax of while statement is:

```
while expression
statement 0
statement 1

statement N
wend
```

Expression is tested first. If it returns True, all the following statements enclosed by while and wend will be executed. It will keep on executing statements until the expression returns False.

Eventually, as expression returns False, while will be terminated without executing statements.

While is similar to do..loop until, except the check is performed at the beginning of the loop. If expression returns False upon first test, statements will not be executed.

```

while i < 90
  i = i + 1
wend

```

```

while i > 0
  i = i div 3
  PORTA = i
wend

```

3.12.3 For Statement

For statement requires you to specify the number of iterations you want the loop to go through. Syntax of for statement is:

```

for counter = initial Value to final Value [step step value]
statement 1
statement 2
...
statement N
next counter

```

Where counter is variable; initial Value and final Value are expressions compatible with counter; statement is any statement that does not change the value of counter; step value is value that is added to the counter in each iteration.

Step value is optional, and defaults to 1 if not stated otherwise. Be careful when using large values for step value, as overflow may occur. Every statement between for and next will be executed once for each iteration.

Example:

Here is a simple example of a for loop used for emitting hex code on PORTB. Nine digits will be printed with one second delay, by incrementing the counter.


```
for i = 1 to 9  
portb = i  
delay_ms(1000)  
next i
```

3.13 Summary

In this chapter, while the importance of microBasic language exercising its influence through PIC microBasic compiler, detailed information have been denoted and covered.

This compiler was one of the best programming software for microcontrollers, thus it is used for our application in this project.

CHAPTER FOUR

THE PIC PROGRAMMER

4.1 Overview

This chapter presents information on PIC microcontroller. General characteristics of PIC microcontroller in addition to its characteristics which gives a significant role among the control fields are presented.

4.2 The Characteristics of PIC Microcontroller

The PIC programmer is able to program most microchip 8,14,18,28 and 40 microcontrollers, from both Mid-range 16xxx and High-End 18xxx microcontrollers, using the IC-Prog Universal Serial programmer which is free software. Along with the programmer and the microcontroller; a wall mount power supply of 12-15Volts DC output, which can supply a minimum of 200mA of current is needed. Using a parallel port extension cable, will make your life easier, especially when the microcontroller many times is needed to be programmed, but still the programmer directly to the parallel port of your PC can be connected. Settlement of the different Microchip microcontrollers:

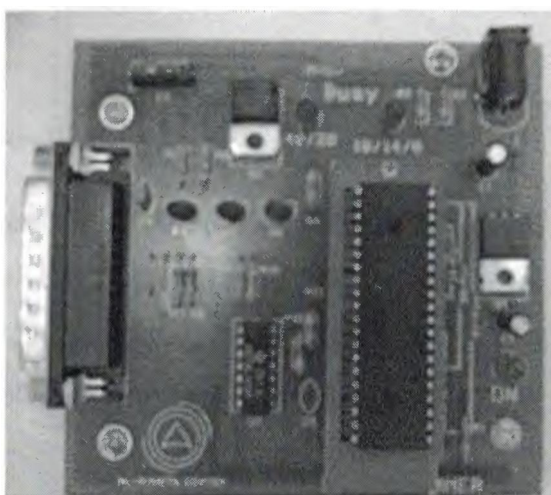


Figure 4.1 PIC programmer

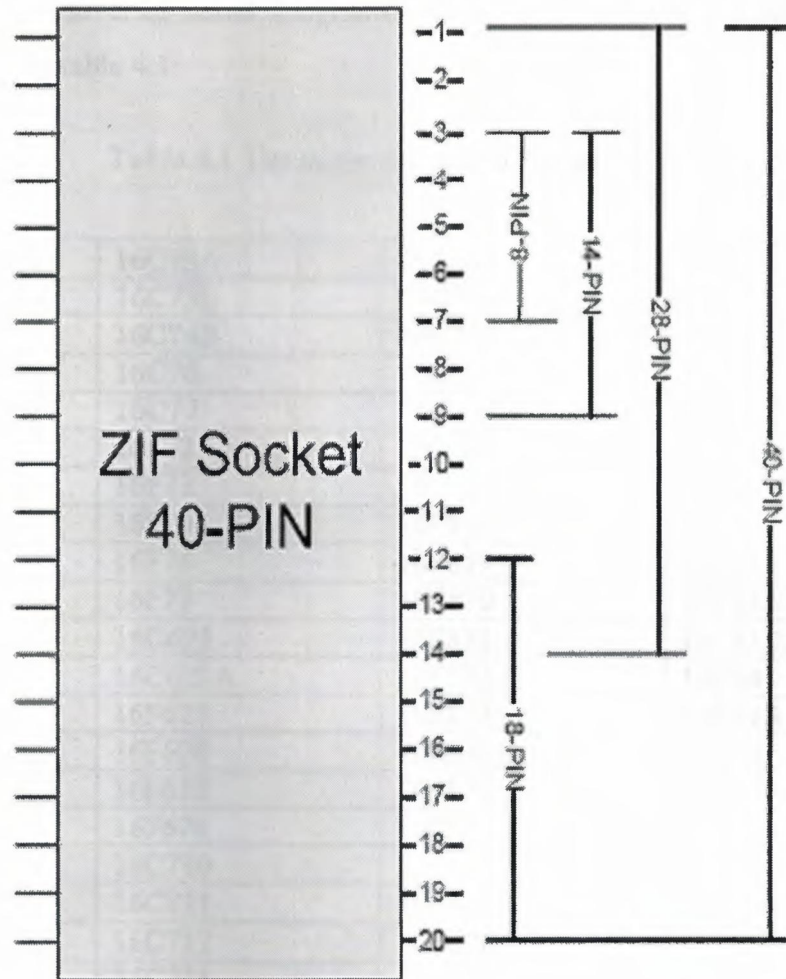


Figure 4.2 Different PIC layout

the manual should be always consulted before inserting any device in the ZIF Socket.

4.3 Supported Microcontrollers

Microchip microcontrollers is Pin-Out Compatible, that is, all the same pin count PIC microcontrollers can be programmed by using the same hardware, using a compatible software is provided, the PIC Programmer circuitry can support the Microchip 8, 14, 18, 28, 40 Pin microcontrollers.

The IC-Prog Universal Serial Programmer Ver1.05C supports the following PIC microcontrollers in table 4.1:

Table 4.1 The supported microcontrollers

12C508	16C73A	16C765	18F242
12C508A	16C73B	16C770	18F248
12C509	16C74B	16C771	18F252
12C509A	16C76	16C773	18F258
12C671	16C77	16C774	18F442
12C672	16F72	16C781	18F448
12F629	16F73	16C782	18F452
12F675	16F74	16F818	18F458
16C433	16F76	16F819	18F1320
16C54	16F77	16F870	18F2.320
16C56	16C622	16F871	18F4320
16CS&	16C622A	16F872	16F84
16C61	16F627	16F873	16F84A
16C62A	16F628	16F873A	16C84
16C62B	16F630	16F874	16F876
16C63	16F676	16C923	16F876A
16C63A	16C710	16C924	16F877
16C64A	16C711	16C620	16F877A
16C65A	16C712	16C620A	
16C6SB	16C715	16C621	
16C66	16C716	16C621A	
16C67	16C717	16F83	
16C71	16C745		
16C72	16C505		
16C72A			

4.4 Setting the IC Prog software

Copy-paste the whole folder named IC-Prog from the CD-ROM as it is to the desktop, copying the program alone is forbidden whereas the whole folder can be copied, then it should be able to be operated from its folder.

4.5 Troubleshooting

This device is experimentally tested and verified to be operating fine, any comments regarding this device would be appreciated, perform this procedure if any error messages occur:

- 1) The IC-Prog software should be opened from the Settings pull-down menu; select Hardware (or F3 can be pressed directly), the following window should appear:

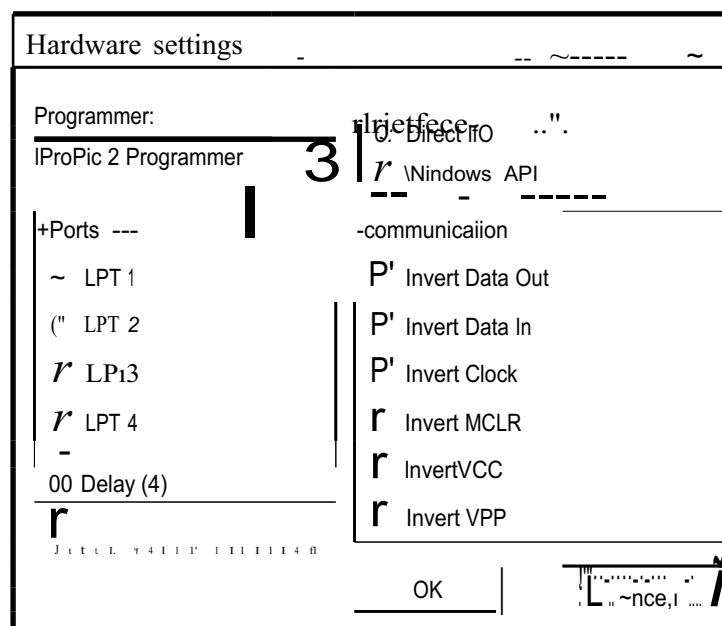


Figure 4.3 IC-Prog Hardware settings window

If it is not as Fig.2 then it should be configured to be so.

- 2) If Windows XP are used or Windows 2000, then it should be made sure that the WinNT/2000/XP driver is installed, to check it; the program has to be opened from the Settings pull-down menu; selecting Options, then selecting the Misc tap, checking the "Enable NT/2000/XP Driver" check box, it should look like the following window:

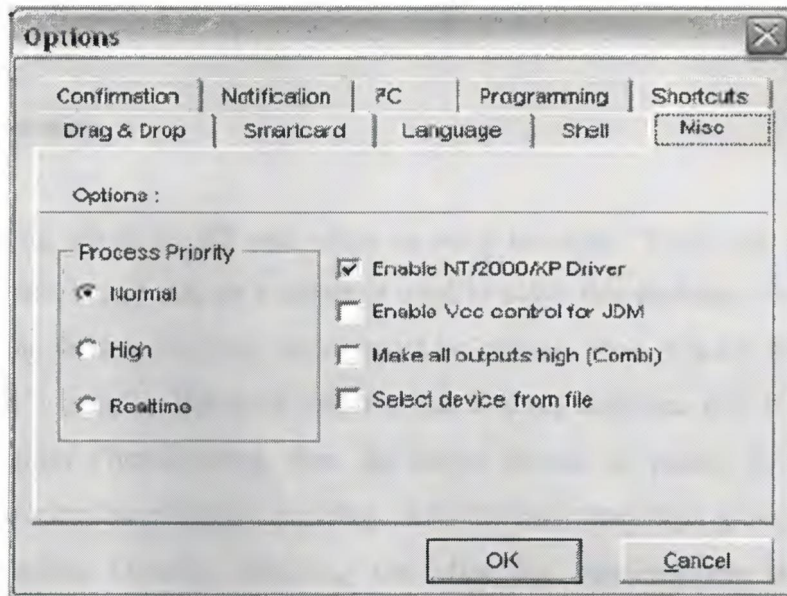


Figure 4.4 IC-Prog Options window

If the window doesn't appear to be like Fig.4.4, then the IC-Prog folder must be opened to make sure that the "IC Prog" driver is installed in the same folder of the IC-Prog.exe; like the following folder window:

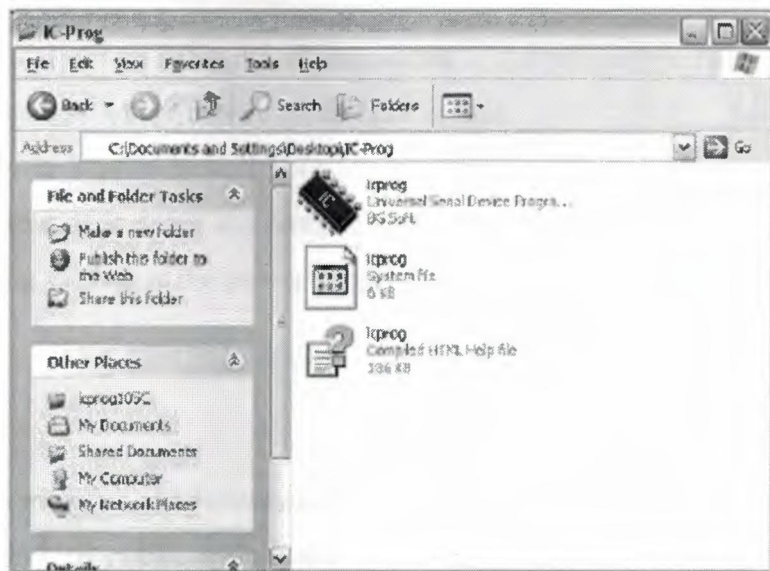


Figure 4.5 IC-Prog containing Folder

4.6 Common Errors

4.6.1 Privileged Instruction

In the case of using windows XP and when an error message "Privileged instruction" appears then the program hangs-out, so a driver is used to solve this problem, from the CD-ROM included with the device, IC-Prog folder must be opened, then WinXP driver folder, and copying the driver "icprog" to the same folder of the IC-Prog software, (i.e. If you run the IC-Prog from C:\Program Files\IC-Prog, then the driver should be pasted in C:\Program Files\IC-Prog). Then during the program pressing ok to the Error message, going to Settings pull-down menu, selecting Options, selecting the Misc tap checking the box "Enable NT/2000/XP Driver", pressing OK to the message appears, the IC-Prog will then restart, then the device should program properly.

4.6.2 Varify Failed

The IC-Prog operates fine but while programing a device; a "Verify failed at address 0000h!" message appears,thus the following stepes should be followen:

- 1) Checking if the parallel port cable is well connected to the programmer and the PC port.
- 2) Checking the settlement of the device on the ZIF socket.
- 3) Checking the power supply is well connected to the programmer hardware, the ON led should be well lighting, if it is not the case, then probably the power supply is not sourcing enough current, trying to change the power supply.

4.7 Summary

This chapter has included the information about how to deal with PIC programmer and how to fit the suitable kind of PIC with its true position upon the programmer.

CHAPTER FIVE

ELEVATOR PROTOTYPE USING MICROCONTROLLER

5.1 Overview

This chapter presents a description for the project and the circuit analysis, beside to the elevator's microBasic program, fully detailed explanation over whole system follows in this chapter.

5.2 The Project Description

The project is all about elevator uses PIC16F877A microcontroller, thus it separated into hardware and software action, it consists of three storeys, the shape of the elevator is in figure 5.1 below:

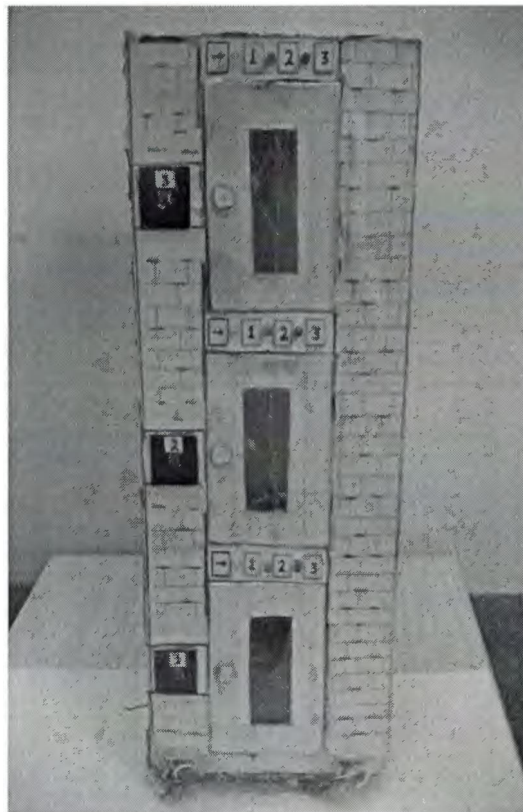


Figure 5.1 The prototype elevator

5.2.1 The Electronic Parts

The electronic Parts that were used in the project are denoted below:

Table 5.1 The used electronic pieces

The electronic piece	Appliances
Red led	4
Green led	3
Yellow led	3
Regulator 7805 (5V,1A)	1
Oscillator (crystal)	1
Multilayer capacitor (0.1 Uf)	2
Capacitor (22PF)	2
Capacitor (220uF-35V)	1
Capacitor (220uF-25V)	1
Resistor (3000)	6
Resistor (4.7kü)	3
Resistor (51 Oü)	9
Resistor (1 Okü)	4
Resistor (1 kü)	1
Transistor NPN (2N2222)	3
L293B	1
Push button	4
Adaptor (AC-DC- {•+})	1
Stepper motor	1
PIC16F877A	1

~2.2 The Circuit Connections

The circuit was built according to the characterizes of PIC16F877A microcontroller, thus the different voltage values usage was avoided by using opto couplers, the figure 5.2 shows the circuit diagram of the elevator.

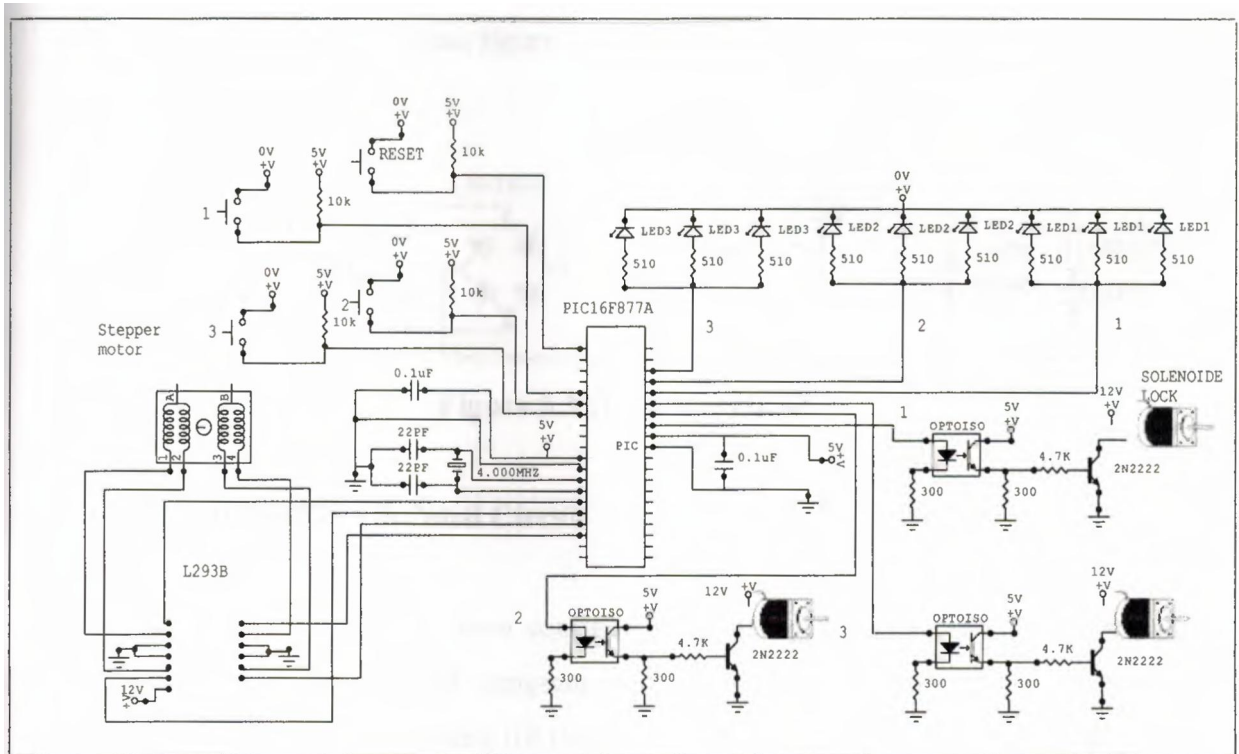


Figure 5.2 Elevator circuit diagram

5.3 The Circuit Analysis

The circuit is a combination of several special circuits to arrive the required purpose till actuate the elevator properly, microcontroller, IC, coils applications devices and protection pieces are used.

5.3.1 The Power Circuit

AC-DC adaptor is used as a supplier for the circuit, thus it gets the voltage from the common AC voltage source (230V-50Hz), it has 230V-50Hz 12W in its Primary side, whereas it has 12V == 500mA 6VA in its secondary side, the 7805 regulator is used to regulate the voltage value till 5V and a maximum current limited up to 1A, the capacitors are used to filtrate the signal, where the red led is located to indicate the power existence and 1K.O. resistor is connected to the led for limiting the entering current to avoid burning the led, figure 5.3 shows the power circuit diagram.

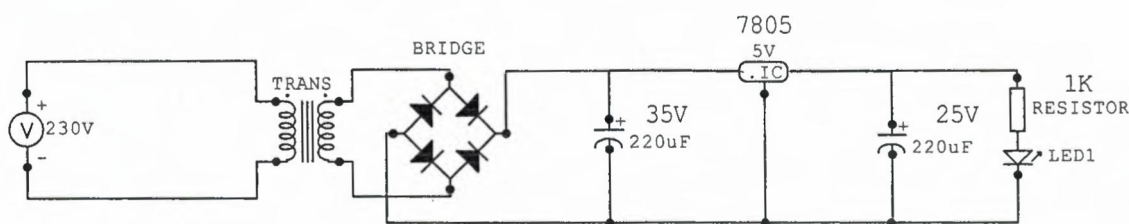


Figure 5.3 The power circuit

5.3.2 The PIC16F877A Related Circuit

The PIC16F877A has its own configured circuit, thus pin one (MCLR) is the pin where resets the downloaded program during pressing on the reset push button, Oscillator is used to clock wising till timing the instruction in a specific interval points to how much fast achieving the program process, the voltage is supplied through certain pins while the grounding occur by other pins as shown in figure 5.4 below:

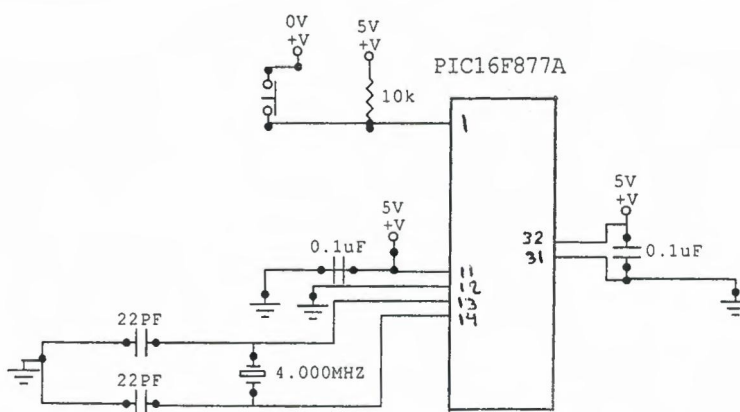


Figure 5.4 The 16F877A related circuit

5.3.3 The I/O Ports Connections

The I/O ports can be chosen among five groups according to the required process that needs to actuate specific amount of sensors and actuators, thus the ports are :

PORTA-RA# -+ (pin2 _ pin7) = 6 pins.

PORTB-RB# -+ (pin33 _ pin40) = 8 pins.

PORTC-RC# -+ (pin23 _ pin26) + (pin16 _ pin18) = 8 pins.

PORTD-RD# -+ (pin19 _ pin22) + (pin27 _ pin30) = 8 pins.

PORTE-RE# -+ (pin8 _ pin10) = 3 pins.

5.3.3.1 Leds Circuit Connection

The leds are used in our project as different colors tell that which level of elevator that we are in. Thus each color has its own threshold voltage, so several values of resistors should be connected to protect leds, in our project it behaves as an output.

5.3.3.2 Push Button Connection

A push-to-make switch returns to its normally open (OFF) position when the button is released, this is shown by the brackets around ON. This is the standard doorbell switch, thus in our project it behaves as a sensor (input).

5.3.3.3 Solenoid Locks Connection

This solenoid locks have been connected to the circuit for adapting the case of doors during the movement of elevator's box, thus in the case of affecting by electrical pulse, it going to be open immediately, while normally they are closed.

The solenoid lock needs more than 12V whereas 5V is supported from PIC which means that it needs external voltage to be actuated, so, to solve this problem the opto couplers are used as shown in figure 5.5 below:

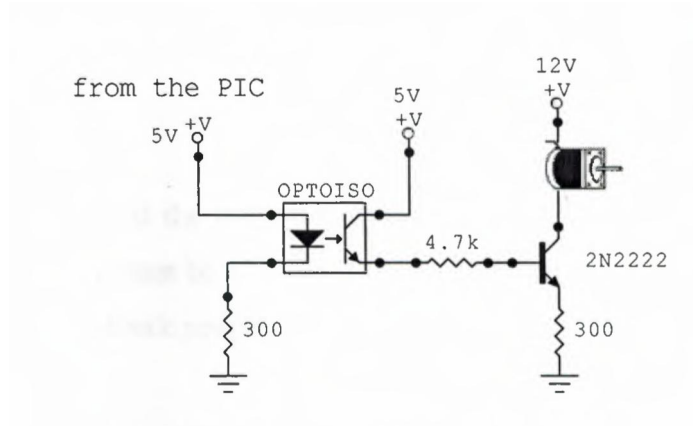


Figure 5.5 Solenoid locks connection

5.3.3.4 Stepper motor connection

Stepper motors operate differently from DC motors. When power is applied to a DC motor, the rotor begins turning smoothly. Speed is measured in revolutions per minute (RPM) and is a function of voltage, current, and load on the motor. The precise positioning of the motor's rotor is not usually possible or desirable. A stepper motor, on the other hand, runs on a controlled sequence of electric pulses to the windings of the motor. Each pulse rotates the stepper motor's rotor by a precise increment. Each increment of the rotor is referred to as a step, hence the name stepper motors, from figure 5.2 we observe that L293B should be connected to the stepper motor, respectively each pin responsible to actuate certain coil in the stepper motor during tidy steps programmed as elevator up or down.

5.4 The Elevator's microBasic Program

The software can be briefly described and fully listed in an appendix at the end of the project.

5.6 Summary

This chapter presented the circuit analyses that are used in the projects and how they function, how they must be connected beside it talked about the project description and the elevator's microBasic program.

CONCLUSION

In this project we have arrived to many points. We tried to apply the optimum controlled design while we were developing the system, so all were about how to deal with the programmed systems instead of the wired control system.

When we started the project, we were realizing the importance of the elevators in our modern life where the people look for comfortable applications.

Many software programs were required such as the microBasic compiler and the ICprog, whereas the hardware side was represented as the programmer device and the driving elevator circuit.

The PIC16F877A was used because of its several good features, the PIC microcontrollers were more preferable than any wired system, actually, the trouble shooting

simplicity during the programmable system against the complicated wired system was the main reason of compensation the PIC in the long buildings, manufacturing applications and in carrying heavy targets.

followed respectively. A fully detailed explanation was provided in the chapters as following:

The first chapter represented classification of electronic components which used in the elevator prototype project, the data sheets, pieces shapes and their functions were included into this chapter.

Chapter two is devoted to the microcontrollers generally whereas the PIC 16F877 A was specified because of its advantages which give it impartiality existence in the manufacture control fields.

Chapter three presented the information about the microBasic language and its compiler which convert the instructions among the user and the programmer.

Chapter four is devoted to the programmer and its construction with a declaration of the PIC's that can be placed upon the Ziff socket.

Chapter five talked about the elevator prototype using microcontrollers' project, circuit diagrams are explained with their functions.

REFERENCES

- [1] John Lovine, PIC microcontroller project book, second edition, McGraw-Hill, New York_2004.
- [2] Nezar Khateb, system design using Microchip controller, first edition, RAY publishing and science, Syria_2001.
- [3] Electronic Al_Baheth center, advance control system using PIC microcontrollers, course duration: 40_hours, Amman_2005.
- [4] PIC microcontrollers, [http://www.mikroelektronika.co.yu/english/product/books/ PIC book/ I-chapter.htm](http://www.mikroelektronika.co.yu/english/product/books/PIC_book/I-chapter.htm), 2005.
- [5] Basic compiler for Microchip PIC microcontrollers, user's manual, <http://www.mikroelektronika.co.yu>, 2005.
- [6] <http://www.resistor.com>
- [7] [http://www.audioheritage.org/main/html/products/altec/604. Html](http://www.audioheritage.org/main/html/products/altec/604.Html)
- [8] <http://www.statcounter.com>
- [9] <http://www.the12volt.com>
- [10] <http://www.the12volt.com/diodes/diodes.asp>

APPENDIX

The microBasic elevator program is listed below:

```
*****
```

```
program Khaled_Elevator
```

```
DIM I AS BYTE
```

```
DIMJ AS BYTE
```

```
DIM ONE FLOOR AS BYTE
```

```
DIM CURRENT POSITION AS BYTE
```

```
SUB PROCEDURE MOVE_STEPPER (DIM POSITION AS BYTE, DIM  
DIRECTION AS BYTE)
```

```
IF DIRECTION== 1 THEN 'UPPER MOVING: ASUMING THAT THE  
CONFIGURATION OF
```

```
FOR I= 0 TO POSITION 'THE STEPPER INPUTS WILL MOVE THE  
ELEVATOR UP
```

```
FOR J = 0 TO ONE FLOOR
```

```
PORTC.0 = 1
```

```
DELAY_MS(10)
```

```
PORTC.1 = 1
```

```
DELAY_MS(10)
```

```
PORTC.2 = 1
```

```
DELAY_MS(10)
```

```
PORTC.3 = 1
```

```
DELAY_MS(10)
```

```
NEXTJ
```

```
NEXT I
```

```
END IF
```

```

IF DIRECTION= 0 THEN 'DOWN MOVING
FOR I= 0 TO POSITION
    FOR J = 0 TO ONE_FLOOR
        PORTC.3 = 1
        DELAY_MS(10)
        PORTC.2 = 1
        DELAY_MS(10)
        PORTC.1 = 1
        DELAY_MS(10)
        PORTC.0= 1
        DELAY_MS(10)
    NEXTJ
NEXT I
END IF

END SUB

'main procedure
main:
    TRISA= 0xFF
    TRISB = 0
    TRISC = 0
    CURRENT_POSITION = 0
    ONE_FLOOR= 20 'THE ONE_FLOOR IS THE VALUE OF THE NUMBER OF
    LOOPS OF THE
        ' STEPPER TO MOVE FROM ONE FLOOR TO ANOTHER
        '***YOU SHOULD TEST AND CALIBRATE THIS VALUE FOR THE
    OPTIMAL
        'PERFORMANCE*****
        '***THE VALUE OF 20 DOESN'T MEAN ANYTHING YET***
RUN TIME:

```

*****GROUND FLOOR*****

IF PORTA.3 = 0 THEN

SELECT CASE CURRENT POSITION

CASE 0 'IF THE ELEVATOR IS ALREADY IN THE GROUND FLOOR

GOTO GROUND FLOOR

CASE 1 'IF THE ELEVATOR IS IN THE FIRST FLOOR

MOVE_STEPPER(1,0) 'MOVE DOWN TO THE GROUND FLOOR

CASE 2 'IF THE ELEVATOR IS IN THE SECOND FLOOR

MOVE_STEPPER(2,0) 'MOVE DOWN TO THE GROUND FLOOR

END SELECT

GROUND FLOOR:

PORTB.0 = 1 'DIRECTLY ENABLE GROUND SOLENOID (OPEN THE DOOR)

CURRENT POSITION= 0 'MARK THE CURRENT POSITION AS GROUND FLOOR

END IF

*****FIRST FLOOR*****

IF PORTA.4 = 0 THEN

SELECT CASE CURRENT POSITION

CASE 0 'IF THE ELEVATOR IS IN THE GROUND FLOOR

MOVE_STEPPER(1,1) 'MOVE UP TO THE FIRST FLOOR

CASE 1 'IF THE ELEVATOR IS ALREADY IN THE FIRST FLOOR

GOTO FIRST FLOOR

CASE 2 'IF THE ELEVATOR IS IN THE SECOND FLOOR

MOVE_STEPPER(1,0) 'MOVE DOWN TO THE FIRST FLOOR

END SELECT

FIRST FLOOR:

```
PORTB.1 = 1      'DIRECTLY ENABLE FIRST SOLENOID (OPEN THE
DOOR)
CURRENT POSITION = 1 'MARK THE CURRENT POSITION AS GROUND
FLOOR
END IF
```

*****SECOUND FLOOR*****

```
IF PORTA.5 = 0 THEN
SELECT CASE CURRENT POSITION
CASE 0      'IF THE ELEVATOR IS IN THE GOUND FLOOR
MOVE_STEPPER(2,1)  'MOVE UP TO THE SECOUND FLOOR

CASE 1      'IF THE ELEVATOR IS IN THE FIRST FLOOR
MOVE_STEPPER(1,1)  'MOVE UP TO THE SECOUND FLOOR

CASE2      'IF THE ELEVATOR IS ALREADY IN THE SECOUND
FLOOR
GOTO SECOUND FLOOR
END SELECT
```

SECOUND FLOOR:

```
PORTB.2 = 1      'DIRECTLY ENABLE SECOUND SOLENOID (OPEN THE
DOOR)
CURRENT POSITION = 2 'MARK THE CURRETN POSITION AS SECOUND
FLOOR
END IF
```

GOTO RUN TIME

end.
