

NEAR EAST UNIVERSITY



Faculty of Engineering

Department of Computer Engineering

Home Appliance Controlling via Internet

Graduation Project

COM- 400

Student: **Mohammad Tailakh (20000713)**
grouped with

Ahmad AL-sayed (20002105)

Supervisor: **Assoc. Prof. Dr. Adnan Khashman**

Nicosia – 2004





ACKNOWLEDGEMENTS

Foremost I would like to pay my special thanks to my parents, who helped me on every phase of my life. They boosted me up about my studies as well as my life. I am very much thankful and grateful to my father whose sacrifice to make this day come true and my mother whose prayers and love for me has encouraged me so make this day come true. I will never ever forget my uncles and my grand father, for his encouraged and love. It is only because of them that today I am capable of completing my degree.

Secondly I would like to present my special appreciation to my supervisor Assoc. Prof. Dr. Adnan Khashman, without him it is impossible for me to complete the project. His trust in my work and me and his priceless awareness for the project has made me do my work with full interest. His friendly behavior with me and his words of encouragement kept me doing my project.

I am also very much grateful to my partner in this project Ahmad Al-sayed and all my companions and my housemate who gave me their precious time to help me to encourage me their ever devotion and all valuable information which I really need to complete my project.

Further I am thankful to Near East University academic staff and all those persons who helped me or encouraged me incompletion of my project.

"Thanks!"

ABSTRACT

Software simulation is a useful tool during the development of any complex system. By simulating the system before it is implemented, it is possible to specify the system requirements and to ensure that new components being developed are working correctly, moreover, many of LPT application use this technique to be under control from the computer, and this technique can be simulate the reality environment and change it to virtual state.

However, the main advantage for this simulation is to control the peripheral devices via computer aid, internet as well.

This project deals with simple software simulation using Visual Basic (VB) programming language to control the outside devices from the computer and to set the start and end time from the user and controlled via computer without interventions of the user.

TABLE OF CONTENTS

ACKNOWLEDGMENT	i
ABSTRACT	ii
TABLE OF CONTENTS	iii
INTRODUCTION	1
CHAPTER ONE: HOW DOES THE INTERNET WORK	3
1.1 Overview	3
1.2 Internet Addresses	3
1.3 Protocol Stacks and Packets	4
1.4 Networking Infrastructure	7
1.5 Internet Infrastructure	8
1.6 The Internet Routing Hierarchy	9
1.7 Domain Names and Address Resolution	11
1.8 Internet Protocols Revisited	12
1.8.1 Application Protocols: HTTP and the World Wide Web	12
1.8.2 Application Protocols: SMTP and Electronic Mail	14
1.8.3 Transmission Control Protocol	14
1.8.4 Internet Protocol	16
1.9 Summary	18

3.3.2 PS/2-Types (Simple Bi-directional)	38
3.3.3 EPP	38
3.3.4 ECP	39
3.3.5 Multi-Mode Ports	39
3.4 Parallel Port Resources	39
3.4.1 Addressing	39
3.4.2 Interrupts	41
3.4.3 DMA Channels	41
3.4.4 Finding Existing Ports	41
3.5 Configuring	42
3.5.1 Port Options	42
3.5.1.1 Multi-mode Ports	43
3.6 Port Hardware	44
3.6.1 Connectors	44
3.6.2 The Circuits Inside	46
3.6.3 Cables	47
3.7 Multiple Uses For One Port	47
3.7.1 Security Keys	48
3.8 Alternatives To Parallel Port	49
3.8.1 Other Parallel Interfaces	49
3.8.1.1 SCSI	49
3.8.1.2 IEEE 488	49

3.8.2 Serial Interfaces	50
3.9 Summary	51
CHAPTER FOUR: CIRCUIT COMPONENTS	52
4.1 Overview	52
4.2 Introduction To Electronics	52
4.3 Components	54
4.3.1 Resistors	54
4.3.1.1 Fixed value resistors	56
4.3.1.2 Resistor Color Code	57
4.3.1.3 Resistor In Series and Parallel	58
4.3.1.4 Variable Resistors	60
4.3.2 Semiconductors	60
4.3.2.1 Transistors	62
4.3.2.2 Light Emitting Diodes (LEDs)	64
4.3.3 Relay Driver	66
4.4 Summary	68
CONCLUSION	69
REFERENCES	70
APPENDIX A	1-A

INTRODUCTION

Distance controlling is one of the fastest growing applications areas of Internet where many companies are now offering house intelligent products controlling via remote control or via normal telephone line, which called intelligent houses.

With the aid of the Internet technology, millions of computers are now networked together around the world. This project intends to use a computer driven peripherals devices such as controlling the light in the house via Internet as an example to demonstrate how a computer can be interfaced with external electronic circuits through the parallel port. This project also intends to show how software programs could monitor these devices.

Why we used parallel port, Parallel port is a simple and inexpensive tool for building computer-controlled devices and projects. The simplicity and ease of programming makes parallel port popular in electronics hobbyist world. The parallel port is often used in Computer controlled robots, home automation.

Several simple circuits will be built to connect to the parallel port. The programming tool chosen to program the parallel port is Visual Basic programming language .As VB is the programming language taught in most high schools, it would be very appropriate to use VB as the software tool. Moreover, VB allows users to create the application GUI easily in terms of labels, text boxes, buttons and etc.

The aims of this project are

- Getting an experience for dealing with this simple type of LPT controlling by building and testing the main function of it.
- Introduce the main functions by using VB to control the home appliance through LPT.
- Getting the main functions of the LPT and how to control it through programming languages such as VB.

This project is divided into two section hardware and software.

- Software section

Chapter 1 describes the basic client-server architectures.

Chapter 2 describes how the visual basic program built.

- Hardware section

Chapter 3 describes the basic principles of parallel port (LPT).

Chapter 2 describes the components of the external circuit.

CHAPTER ONE

HOW DOES THE INTERNET WORK

1.1 Overview

This chapter explains the underlying infrastructure and technologies that make the Internet work. It does not go into great depth, but covers enough of each area to give a basic understanding of the concepts involved.

1.2 Internet Addresses

The Internet's growth has become explosive and it seems impossible to escape the bombardment of www.com's seen constantly on television, heard on radio, and seen in magazines. Because the Internet has become such a large part of our lives, a good understanding is needed to use this new tool most effectively.

Because the Internet is a global network of computers each computer connected to the Internet must have a unique address. Internet addresses are in the form nnn.nnn.nnn.nnn where nnn must be a number from 0 - 255. This address is known as an IP address. (IP stands for Internet Protocol; more on this later.)

The picture below illustrates two computers connected to the Internet; your computer with IP address 1.2.3.4 and another computer with IP address 5.6.7.8. The Internet is represented as an abstract object in-between. (As this paper progresses, the Internet portion of Diagram 1 will be explained and redrawn several times as the details of the Internet are exposed.)



Figure 1.1 Internet portion [2]

If you connect to the Internet through an Internet Service Provider (ISP), you are usually assigned a temporary IP address for the duration of your dial-in session. If you connect to the Internet from a local area network (LAN) your computer might have a permanent IP address or it might obtain a temporary one from a DHCP (Dynamic Host Configuration Protocol) server. In any case, if you are connected to the Internet, your computer has a unique IP address.

If you're using Microsoft Windows or a flavor of Unix and have a connection to the Internet, there is a handy program to see if a computer on the Internet is alive. It's called ping, probably after the sound made by older submarine sonar systems.¹ If you are using Windows, start a command prompt window. If you're using a flavor of Unix, get to a command prompt. Type `ping www.yahoo.com`. The ping program will send a 'ping' (actually an ICMP (Internet Control Message Protocol) echo request message) to the named computer.

The pinged computer will respond with a reply. The ping program will count the time expired until the reply comes back (if it does). Also, if you enter a domain name (i.e. `www.yahoo.com`) instead of an IP address, ping will resolve the domain name and display the computer's IP address. More on domain names and address resolution later.

1.3 Protocol Stacks and Packets

So your computer is connected to the Internet and has a unique address. How does it 'communicate' to other computers connected to the Internet? An example should serve here: Let's say your IP address is 1.2.3.4 and you want to send a message to the computer 5.6.7.8. The message you want to send is "Hello computer 5.6.7.8!". Obviously, the message must be transmitted over whatever kind of wire connects your computer to the Internet. Let's say you've dialed into your ISP from home and the message must be transmitted over the phone line. Therefore the message must be translated from alphabetic text into electronic signals, transmitted over the Internet, and then translated back into alphabetic text. How is this accomplished? Through the use of a protocol stack. Every computer needs one to communicate on the Internet and it is usually built into the computer's operating system (i.e. Windows, Unix, etc.).

Table 1.1 The TCP/IP stack [2]

Protocol Layer	Comments
Application Protocols Layer	Protocols specific to applications such as WWW, e-mail, FTP, etc.
Transmission Control Protocol Layer	TCP directs packets to a specific application on a computer using a port number.
Internet Protocol Layer	IP directs packets to a specific computer using an IP address.
Hardware Layer	Converts binary packet data to network signals and back. (E.g. ethernet network card, modem for phone lines, etc.)

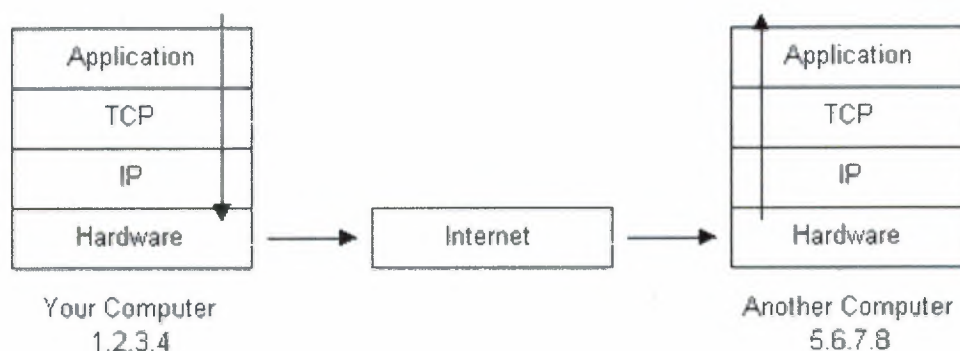


Figure 1.2 Protocol stack [1]

The protocol stack used on the Internet is referred to as the TCP/IP protocol stack because of the two major communication protocols used. The TCP/IP stack looks like the table1.1

If we were to follow the path that the message "Hello computer 5.6.7.8!" took from our computer to the computer with IP address 5.6.7.8, it would happen as shown in figure 1.2

1. The message would start at the top of the protocol stack on your computer and work its way downward.
2. If the message to be sent is long, each stack layer that the message passes through may break the message up into smaller chunks of data. This is because data sent over the Internet (and most computer networks) are sent in manageable chunks. On the Internet, these chunks of data are known as packets.
3. The packets would go through the Application Layer and continue to the TCP layer. Each packet is assigned a port number. Ports will be explained later, but suffice to say that many programs may be using the TCP/IP stack and sending messages. We need to know which program on the destination computer needs to receive the message because it will be listening on a specific port.
4. After going through the TCP layer, the packets proceed to the IP layer. This is where each packet receives its destination address, 5.6.7.8.
5. Now that our message packets have a port number and an IP address, they are ready to be sent over the Internet. The hardware layer takes care of turning our packets containing the alphabetic text of our message into electronic signals and transmitting them over the phone line.
6. On the other end of the phone line your ISP has a direct connection to the Internet. The ISP's router examines the destination address in each packet and determines where to send it. Often, the packet's next stop is another router. More on routers and Internet infrastructure later.
7. Eventually, the packets reach computer 5.6.7.8. Here, the packets start at the bottom of the destination computer's TCP/IP stack and work upwards.
8. As the packets go upwards through the stack, all routing data that the sending computer's stack added (such as IP address and port number) is stripped from the packets.

1.4 Networking Infrastructure

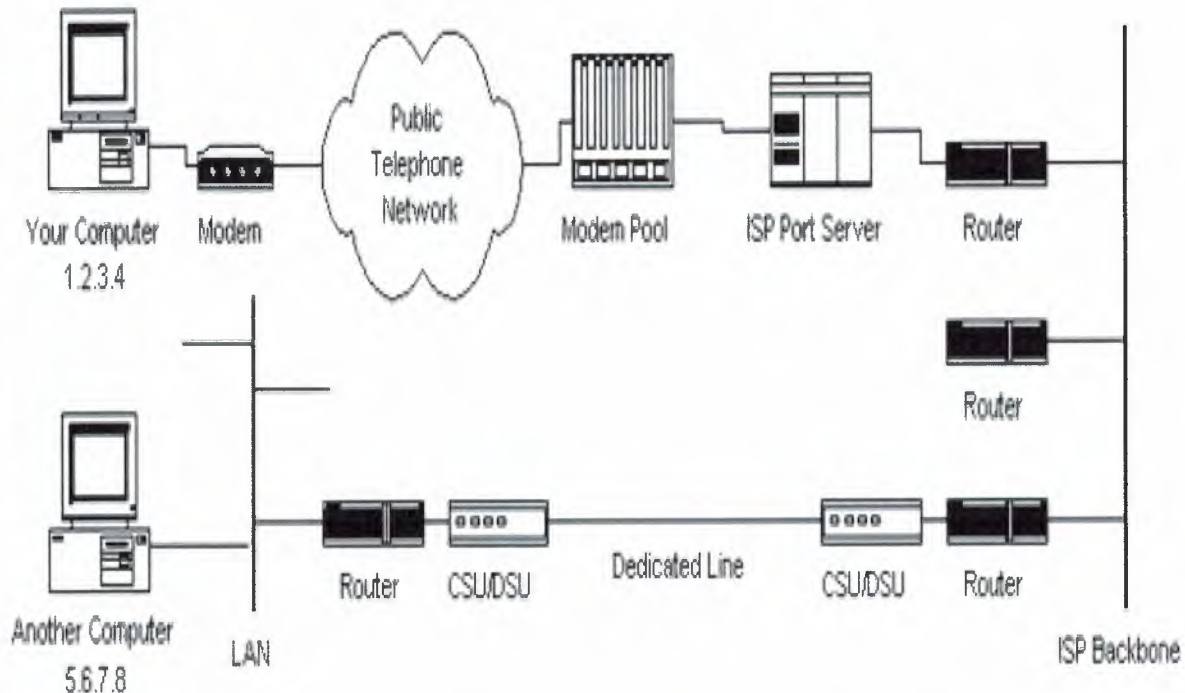


Figure 1.3 Infrastructure of Networking [1]

Here we see in figure 1.3 redrawn with more detail. The physical connection through the phone network to the Internet Service Provider might have been easy to guess, but beyond that might bear some explanation.

The ISP maintains a pool of modems for their dial-in customers. This is managed by some form of computer (usually a dedicated one), which controls data flow from the modem pool to a backbone or dedicated line router. This setup may be referred to as a port server, as it 'serves' access to the network. Billing and usage information is usually collected here as well.

After your packets traverse the phone network and your ISP's local equipment, they are routed onto the ISP's backbone or a backbone the ISP buys bandwidth from. From here the packets will usually journey through several routers and over several backbones, dedicated lines, and other networks until they find their destination, the computer with address 5.6.7.8. But wouldn't it would be nice if we knew the exact

route our packets were taking over the Internet? As it turns out, there is a way by using Traceroute Program. If you're using Microsoft Windows or a flavor of Unix and have a connection to the Internet, here is another handy Internet program. This one is called traceroute and it shows the path your packets are taking to a given Internet destination. Like ping, you must use traceroute from a command prompt. In Windows, use `tracert www.yahoo.com`. From a Unix prompt, type `traceroute www.yahoo.com`. Like ping, you may also enter IP addresses instead of domain names. Traceroute will print out a list of all the routers, computers, and any other Internet entities that your packets must travel through to get to their destination.

If you use traceroute, you'll notice that your packets must travel through many things to get to their destination. Most have long names such as `sjc2-core1-h2-0-0.atlas.digex.net` and `fddi0-0.br4.SJC.globalcenter.net`. These are Internet routers that decide where to send your packets. Several routers are shown in figure 1.3, but only a few. figure 1.3 is meant to show a simple network structure. The Internet is much more complex

1.5 Internet Infrastructure

The Internet backbone is made up of many large networks, which interconnect with each other. These large networks are known as Network Service Providers or NSPs. Some of the large NSPs are UUNet, CerfNet, IBM, BBN Planet, SprintNet, PSINet, as well as others. These networks peer with each other to exchange packet traffic. Each NSP is required to connect to three Network Access Points or NAPs. At the NAPs, packet traffic may jump from one NSP's backbone to another NSP's backbone. NSPs also interconnect at Metropolitan Area Exchanges or MAEs. MAEs serve the same purpose as the NAPs but are privately owned. NAPs were the original Internet interconnect points. Both NAPs and MAEs are referred to as Internet Exchange Points or IXs. NSPs also sell bandwidth to smaller networks, such as ISPs and smaller bandwidth providers. Below is a figure showing this hierarchical infrastructure [1].

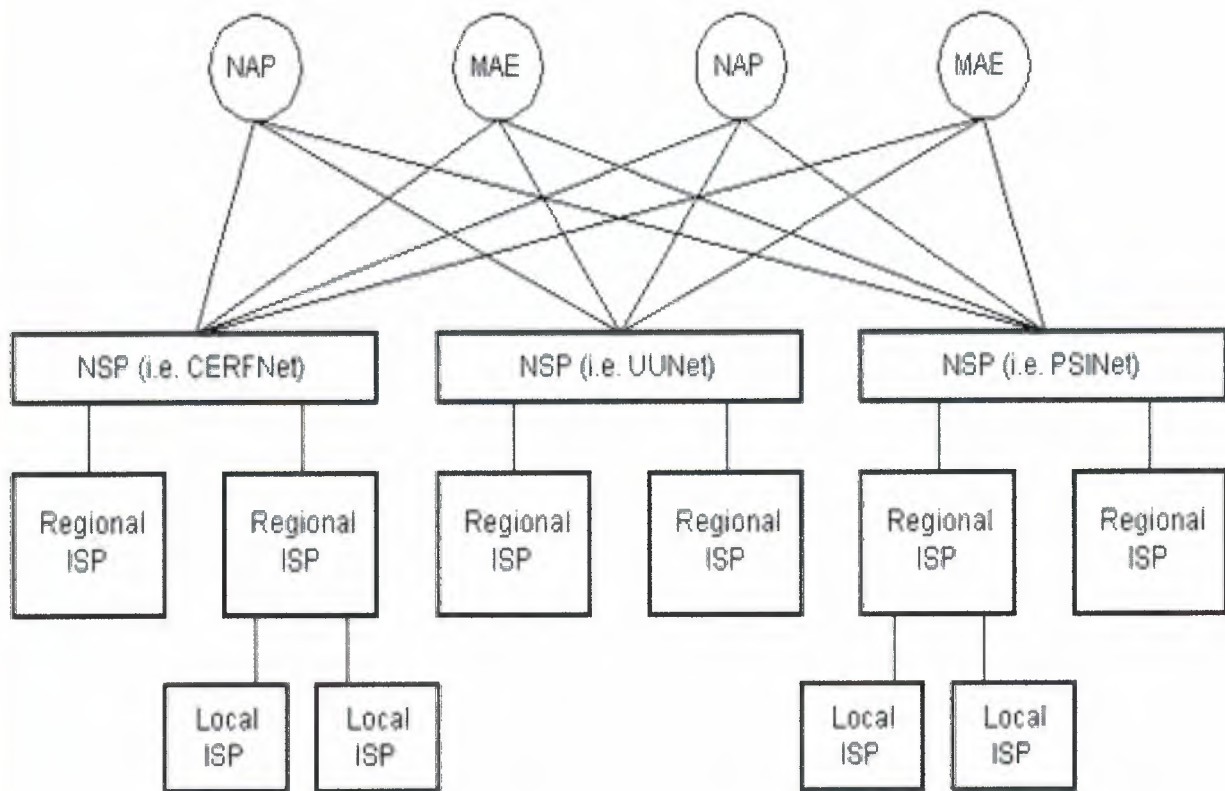


Figure 1.4 Hierarchical Infrastructure of Internet [1]

This is not a true representation of an actual piece of the Internet. figure 1.4 is only meant to demonstrate how the NSPs could interconnect with each other and smaller ISPs. None of the physical network components are shown in figure 1.4 as they are in figure 1.3. This is because a single NSP's backbone infrastructure is a complex drawing by itself. Most NSPs publish maps of their network infrastructure on their web sites and can be found easily. To draw an actual map of the Internet would be nearly impossible due to it's size, complexity, and ever changing structure.

1.6 The Internet Routing Hierarchy

So how do packets find their way across the Internet? Does every computer connected to the Internet know where the other computers are? Do packets simply get 'broadcast' to every computer on the Internet? The answer to both the proceeding questions is 'no'. No computer knows where any of the other computers are, and packets do not get

sent to every computer. The information used to get packets to their destinations is contained in routing tables kept by each router connected to the Internet.

Routers are packet switches. A router is usually connected between networks to route packets between them. Each router knows about its sub-networks and which IP addresses they use. The router usually doesn't know what IP addresses are 'above' it. Examine figure 1.5 below. The black boxes connecting the backbones are routers. The larger NSP backbones at the top are connected at a NAP. Under them are several sub-networks, and under them, more sub-networks. At the bottom are two local area networks with computers attached.

When a packet arrives at a router, the router examines the IP address put there by the IP protocol layer on the originating computer. The router checks it's routing table. If the network containing the IP address is found, the packet is sent to that network. If the network containing the IP address is not found, then the router sends the packet on a default route, usually up the backbone hierarchy to the next router. Hopefully the next router will know where to send the packet [1].

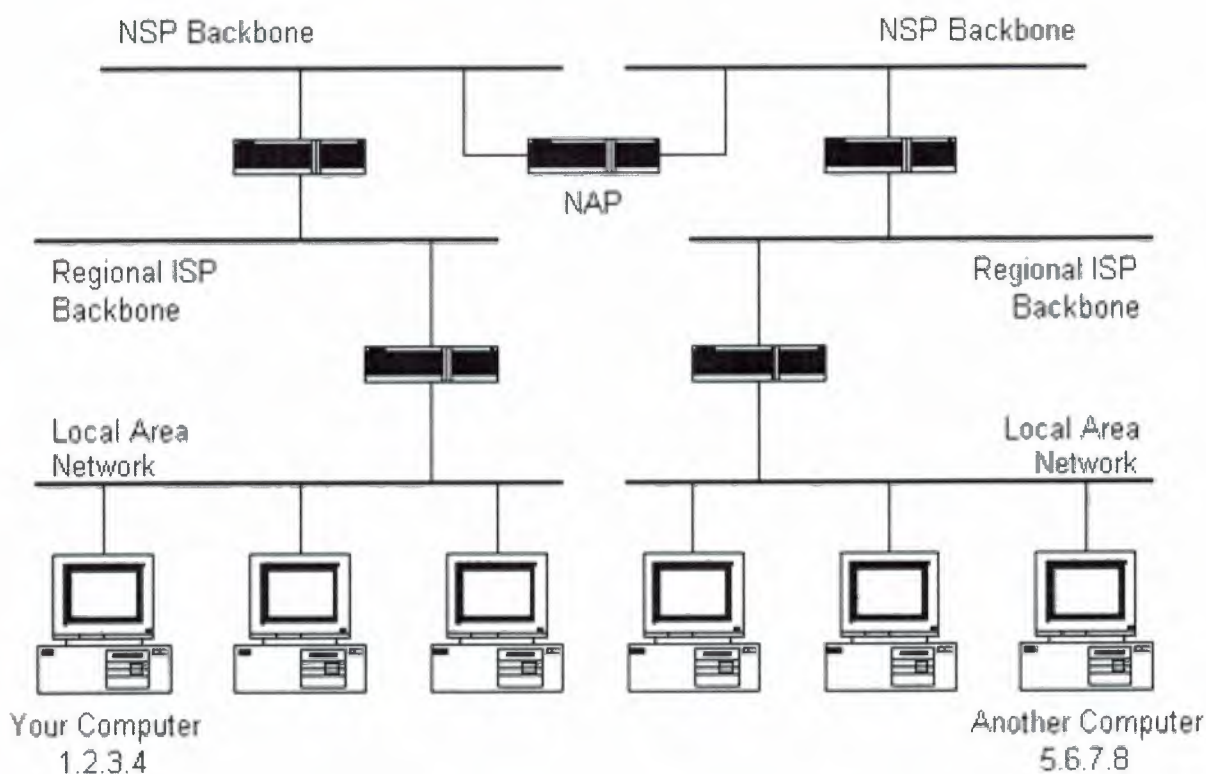


Figure 1.5 Routers Interconnection [2]

If it does not, again the packet is routed upwards until it reaches a NSP backbone. The routers connected to the NSP backbones hold the largest routing tables and here the packet will be routed to the correct backbone, where it will begin its journey 'downward' through smaller and smaller networks until it finds its destination.

1.7 Domain Names and Address Resolution

But what if you don't know the IP address of the computer you want to connect to? What if you need to access a web server referred to as `www.anothercomputer.com`? How does your web browser know where on the Internet this computer lives? The answer to all these questions is the Domain Name Service or DNS. The DNS is a distributed database, which keeps track of computer's names and their corresponding IP addresses on the Internet.

Many computers connected to the Internet host part of the DNS database and the software that allows others to access it. These computers are known as DNS servers. No DNS server contains the entire database; they only contain a subset of it. If a DNS server does not contain the domain name requested by another computer, the DNS server re-directs the requesting computer to another DNS server.

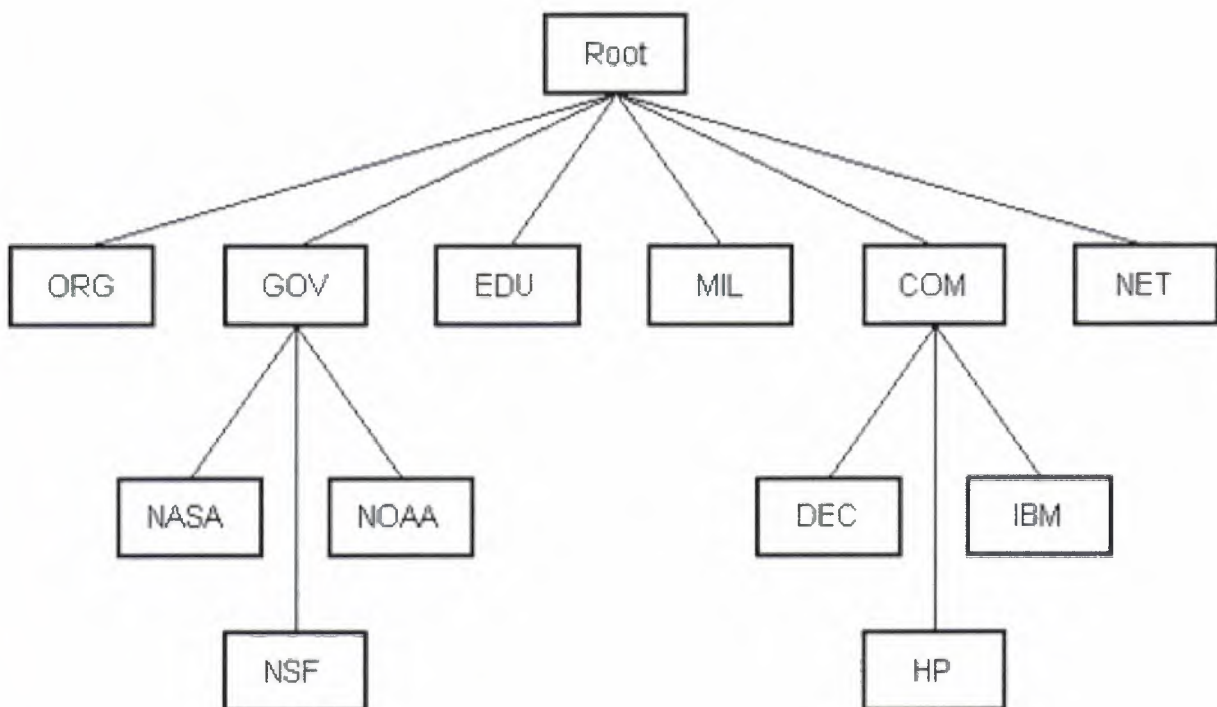


Figure 1.6 How The Domain Name Server (DNS) Work [1]

The Domain Name Service is structured as a hierarchy similar to the IP routing hierarchy. The computer requesting a name resolution will be re-directed 'up' the hierarchy until a DNS server is found that can resolve the domain name in the request. Figure 1.6 illustrates a portion of the hierarchy. At the top of the tree are the domain roots. Some of the older, more common domains are seen near the top. What is not shown are the multitude of DNS servers around the world, which form the rest of the hierarchy.

When an Internet connection is setup (e.g. for a LAN or Dial-Up Networking in Windows), one primary and one or more secondary DNS servers are usually specified as part of the installation. This way, any Internet applications that need domain name resolution will be able to function correctly. For example, when you enter a web address into your web browser, the browser first connects to your primary DNS server. After obtaining the IP address for the domain name you entered, the browser then connects to the target computer and requests the web page you wanted.

1.8 Internet Protocols Revisited

As hinted to earlier in the section about protocol stacks, one may surmise that there are many protocols that are used on the Internet. This is true; there are many communication protocols required for the Internet to function. These include the TCP and IP protocols, routing protocols, medium access control protocols, application level protocols, etc. The following sections describe some of the more important and commonly used protocols on the Internet. Higher level protocols are discussed first, followed by lower level protocols [1].

1.8.1 Application Protocols: HTTP and the World Wide Web

One of the most commonly used services on the Internet is the World Wide Web (WWW). The application protocol that makes the web work is Hypertext Transfer Protocol or HTTP. Do not confuse this with the Hypertext Markup Language (HTML). HTML is the language used to write web pages. HTTP is the protocol that

web browsers and web servers use to communicate with each other over the Internet. It is an application level protocol because it sits on top of the TCP layer in the protocol stack and is used by specific applications to talk to one another. In this case the applications are web browsers and web servers.

HTTP is a connectionless text based protocol. Clients (web browsers) send requests to web servers for web elements such as web pages and images. After a server services the request, the connection between client and server across the Internet is disconnected. A new connection must be made for each request. Most protocols are connection oriented. This means that the two computers communicating with each other keep the connection open over the Internet. HTTP does not however. Before an HTTP request can be made by a client, a new connection must be made to the server.

When you type a URL into a web browser, this is what happens:

1. If the URL contains a domain name, the browser first connects to a domain name server and retrieves the corresponding IP address for the web server.
2. The web browser connects to the web server and sends an HTTP request (via the protocol stack) for the desired web page.
3. The web server receives the request and checks for the desired page. If the page exists, the web server sends it. If the server cannot find the requested page, it will send an HTTP 404 error message. (404 means 'Page Not Found' as anyone who has surfed the web probably knows.)
4. The web browser receives the page back and the connection is closed.
5. The browser then parses through the page and looks for other page elements it needs to complete the web page. These usually include images, applets, etc.
6. For each element needed, the browser makes additional connections and HTTP requests to the server for each element.
7. When the browser has finished loading all images, applets, etc. the page will be completely loaded in the browser window.

1.8.2 Application Protocols: SMTP and Electronic Mail

Another commonly used Internet service is electronic mail. E-mail uses an application level protocol called Simple Mail Transfer Protocol or SMTP. SMTP is also a text based protocol, but unlike HTTP, SMTP is connection oriented. SMTP is also more complicated than HTTP. There are many more commands and considerations in SMTP than there are in HTTP.

When you open your mail client to read your e-mail, this is what typically happens:

1. The mail client (Netscape Mail, Lotus Notes, Microsoft Outlook, etc.) opens a connection to its default mail server. The mail server's IP address or domain name is typically setup when the mail client is installed.
2. The mail server will always transmit the first message to identify itself.
3. The client will send an SMTP HELO command to which the server will respond with a 250 OK message.
4. Depending on whether the client is checking mail, sending mail, etc. the appropriate SMTP commands will be sent to the server, which will respond accordingly.

This request/response transaction will continue until the client sends an SMTP QUIT command. The server will then say goodbye and the connection will be closed

1.8.3 Transmission Control Protocol

Under the application layer in the protocol stack is the TCP layer. When applications open a connection to another computer on the Internet, the messages they send (using a specific application layer protocol) get passed down the stack to the TCP layer. TCP is responsible for routing application protocols to the correct application on the destination computer. To accomplish this, port numbers are used. Ports can be thought of as separate channels on each computer. For example, you can surf the web while reading e-mail. This is because these two applications (the web browser and the mail client) used different port numbers. When a packet arrives at a computer and makes

its way up the protocol stack, the TCP layer decides which application receives the packet based on a port number.

TCP works like this [1]:

- When the TCP layer receives the application layer protocol data from above, it segments it into manageable 'chunks' and then adds a TCP header with specific TCP information to each 'chunk'. The information contained in the TCP header includes the port number of the application the data needs to be sent to.
- When the TCP layer receives a packet from the IP layer below it, the TCP layer strips the TCP header data from the packet, does some data reconstruction if necessary, and then sends the data to the correct application using the port number taken from the TCP header.

This is how TCP routes the data moving through the protocol stack to the correct application.

TCP is not a textual protocol. TCP is a connection-oriented, reliable, byte stream service. Connection-oriented means that two applications using TCP must first establish a connection before exchanging data. TCP is reliable because for each packet received, an acknowledgement is sent to the sender to confirm the delivery. TCP also includes a checksum in its header for error-checking the received data. The TCP header looks like figure 1.7.

Notice that there is no place for an IP address in the TCP header. This is because TCP doesn't know anything about IP addresses. TCP's job is to get application level data from application to application reliably. The task of getting data from computer to computer is the job of IP.

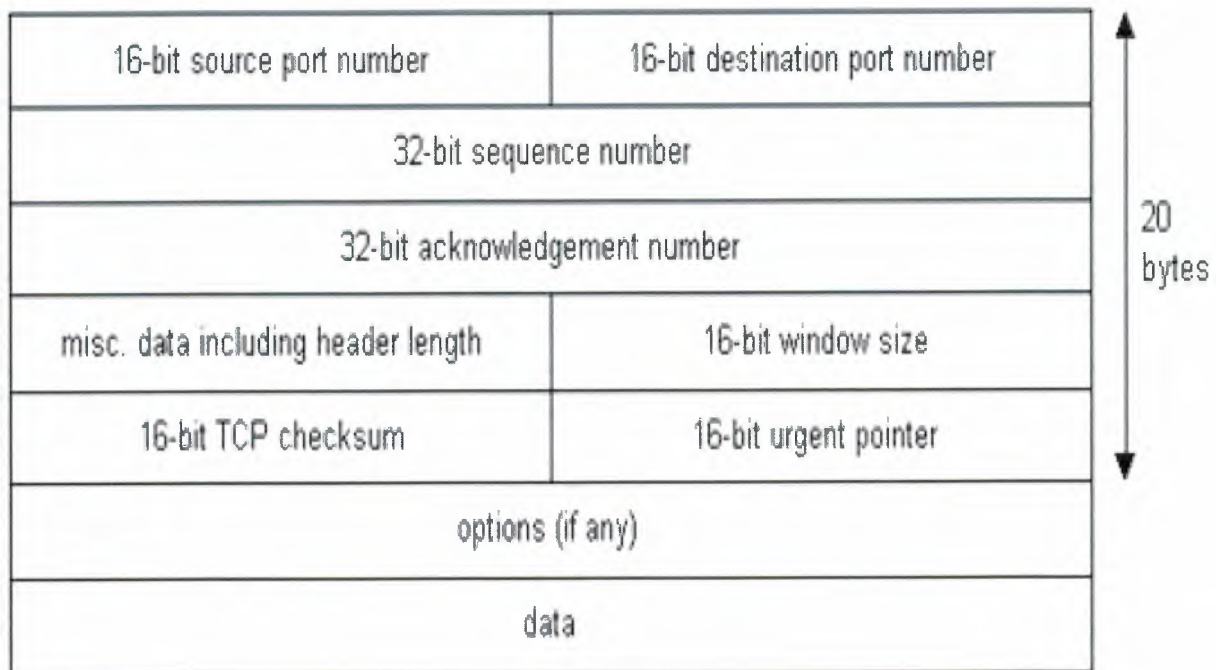


Figure 1.7 TCP Headers [1]

1.8.4 Internet Protocol

Unlike TCP, IP is an unreliable, connectionless protocol. IP doesn't care whether a packet gets to its destination or not. Nor does IP know about connections and port numbers. IP's job is to send and route packets to other computers. IP packets are independent entities and may arrive out of order or not at all. It is TCP's job to make sure packets arrive and are in the correct order. About the only thing IP has in common with TCP is the way it receives data and adds its own IP header information to the TCP data. The IP header looks like figure 1.8.

In figure 1.8 we see the IP addresses of the sending and receiving computers in the IP header. In figure 1.9 shows what a packet looks like after passing through the application layer, TCP layer, and IP layer. The application layer data is segmented in the TCP layer, the TCP header is added, the packet continues to the IP layer, the IP header is added, and then the packet is transmitted across the Internet.

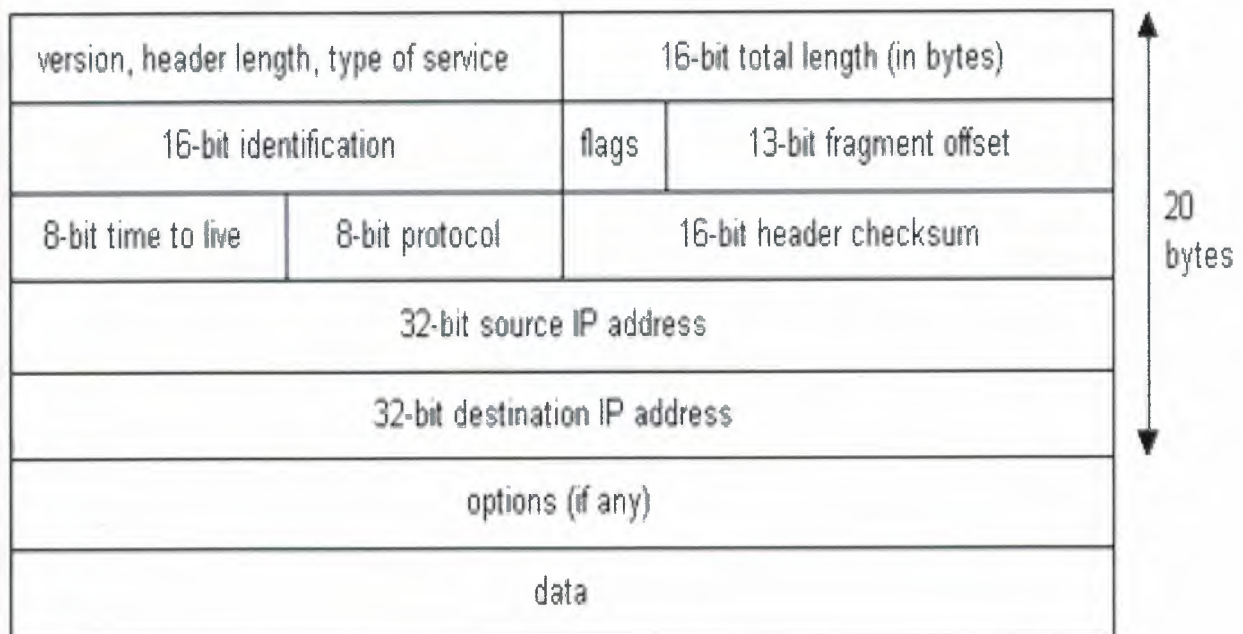


Figure 1.8 IP Headers [2]

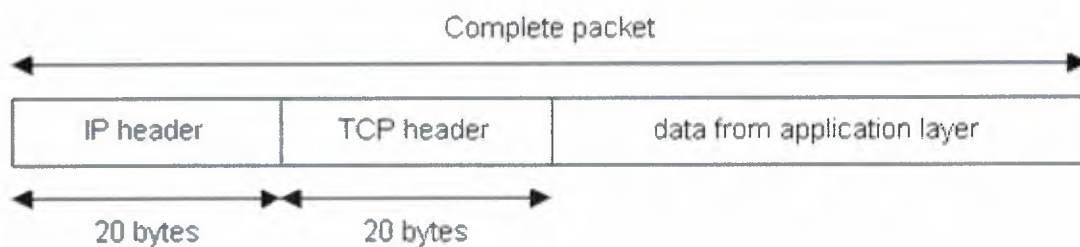


Figure 1.9 The Packet looks after passing Application, TCP and IP layers [2]

1.9 Summary

We explained in this chapter how the data passing through the network layers in the internet, and the principles of the TCP/IP protocol is described as this protocol is used in the client-server.

The underlying infrastructure and technologies that make the internet work, and the infrastructure of NSP, NAP, IX and MAE.

CHAPTER TWO

SOFTWARE SETUP

2.1. Overview

In the previous chapter we have define the hardware setup, while in this chapter the software setup will be defined. We shall begin with the program language used for the main algorithms and codes. Furthermore, we shall give an example of the designed GUI and the technical specification of the software structure.

2.2. What Is Visual Basic

Visual Basic (VB) is an object-oriented, event-driven GUI programming language developed by Microsoft Corporation for Windows application development.

Bill Gates, the CEO and cofounder of Microsoft, developed the original DOS Basic interpreter as one of his last programming projects more than a decade ago. Since Visual Basic has grown to more than several hundreds keywords reflecting its rich object-oriented nature.[3]

2.3. Why Visual Basic

Visual Basic is a useful language for many reasons:

- We can have multiple windows on one screen. These windows have full access to the clipboard and to the information in most other windows application running at same time.
- Microsoft Visual Basic is the quickest and easier way to create applications for Microsoft Windows, which run quickly.

- Visual Basic makes it easy to build large programs by allowing modern modular programming techniques. This means you can break a program into easy-to-handle modules.
- The programming language built into Visual Basic has easy-to use graphics statements, powerful built-in function for mathematics and string manipulations, and sophisticated file-handling capabilities.

2.4. Modules

Code in Visual Basic is stored in modules. There are three kinds of modules [3]:

- Form.
- Standard.
- Class.

Simple applications can consist of just a single form, and all of the code in the application resides in that form module. As your applications get larger and more sophisticated, you add additional forms. Eventually you might find that there is common code you want to execute in several forms. You do not want to duplicate the code in both forms, so you create a separate module containing a procedure that implements the common code. This separate module should be a standard module.

Over time, you can buildup a library of modules containing shared procedures.

Each standard, class, and form module can contain:

- Declarations. You can place constant, type, variable, and dynamic-link library (DLL) procedure declarations at the module level of form, class or standard modules.
- Procedure. As sub, Function, or Property procedure contains pieces of code that can be executed as a unit.

2.4.1 Form Modules

Form modules (FRM file name extension) are the foundation of most Visual Basic applications. They can contain procedures that handle events, general procedures, and form-level declarations of variables, constants, types, and external procedures

If you were to look at a form module in text editor, you would also see descriptions of the form and its controls, including their property setting. The code that you write in a form module is specific to the particular application to which the form belongs, it might also reference other forms or objects within that application.

2.4.2 Standard Modules

Standard modules (BAS file name extension) are containers for procedures and declarations commonly accessed by other modules within the application. They can contain global (available to the whole application) or module-level declarations of variables, constant, types, external procedures, and global procedure. The code that you write in a standard module isn't necessarily tied to a particular applications, if you're careful not to reference forms or controls by name, a standard module can be reused in many different applications.

2.4.3 Class Modules

Class modules (CLS file name extension) are the foundation of object-oriented programming in Visual Basic you can write code in class modules to create new objects. These new objects can be including your own customized properties and methods. Actually, forms are just class modules that can have controls placed on them and can display form windows.

2.4.4 Input/Output Module

In this project, the I/O module should be installed; this module is responsible to make a connection between the software and hardware environments. Further more, we have installed this module as (IO.DLL) through windows platform to system32. After that we should call this module to Visual Basic program before building the main forms or the user interface as shown in figure 2.1.

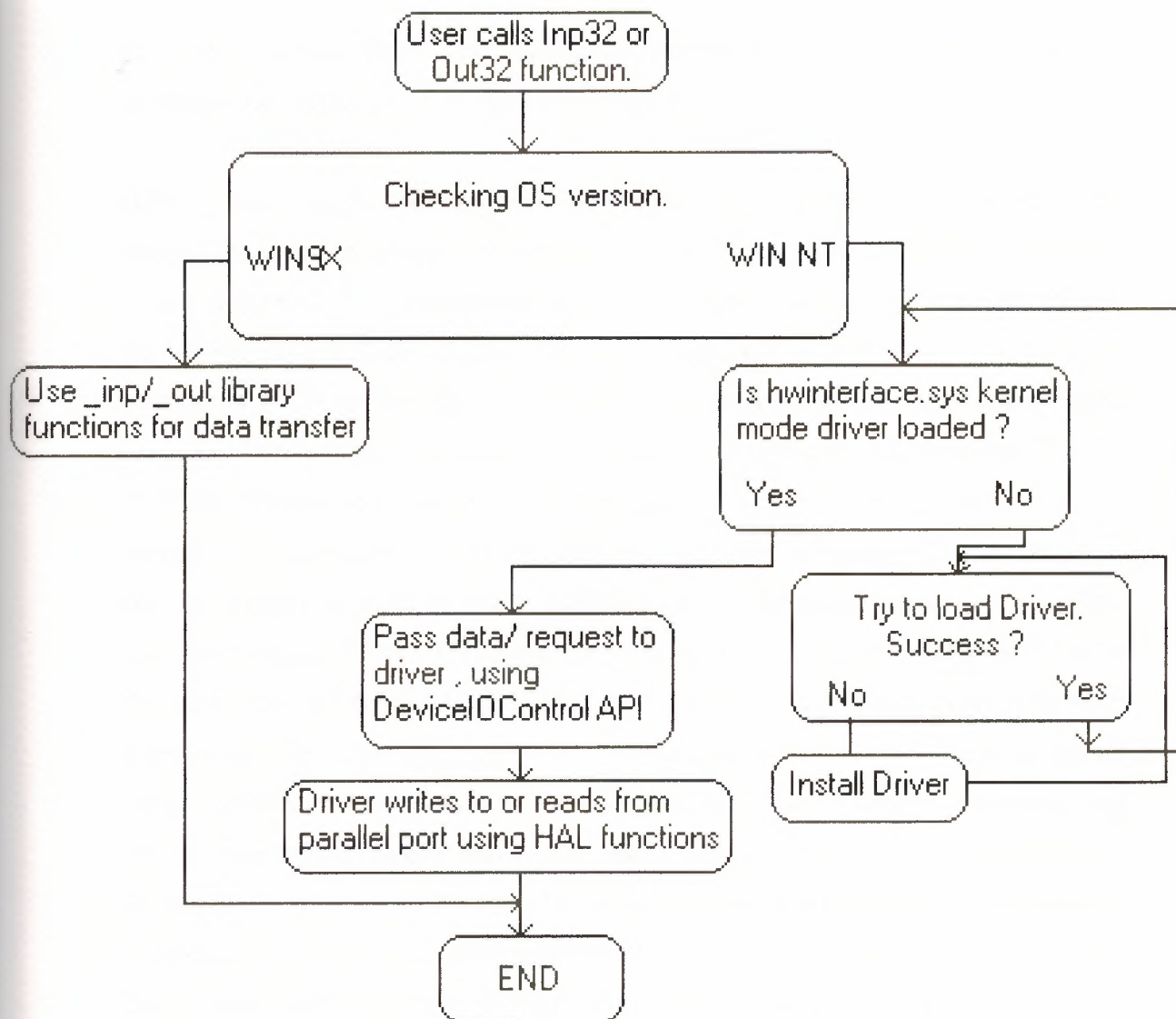


Figure 2.1 Input Output (I/O) Flowchart [4]

Writing programs to talk with parallel port was pretty easy in old DOS days and in Win95/98 too. We can use Inporb and outportb or _inp() or _Outp functions in our program without any problem if we are running the program on Dos or WIN95/98. But entering to the new era of NT clone operating systems like WIN NT4, WIN2000, WINXP, all this simplicity goes away. Being interested in Parallel port interfacing and programming you might have experienced the problems in writing a program that can talk to parallel port successfully in NT based operating systems. When we are trying to run a program which is written using the conventional software functions like Inporb, outportb, _inp() or _Outp on a WINNT or WIN2000 system, it will show an error message that "The exception privileged instruction occurred in the application at location " as shown in figure 2.2.

Being a very secure operating system, Windows NT assigns some privileges and restrictions to different types of programs running on it. It classifies all the programs in to two categories, User mode and Kernel mode ie; running in ring3 and ring0 modes. user mode programs are running in ring3 mode and Kernel mode programs are running in ring0 mode. The programs you generally write falls in the user mode category. The user mode programs are restricted to use certain instructions like IN, OUT etc.. Whenever the operating system find that a user mode program is trying to execute such instructions, the operating system stops execution of those programs and will display an error message. Eventually our interfacing programs stops where they are executing IN or OUT instructions to read or write data to parallel port. But in the same time Kernel mode programs are in no way restricted in executing such instructions. Device drivers are capable of running in kernel mode. So the workaround for the above stated problem is to write a kernel mode driver capable of reading and writing data to parallel port and let the user mode program to communicate with it. Writing a driver is not an easy job for even experienced programmers. But writing a simple driver for communicating with parallel port is a simple task when drivers like USB, sound card etc.. are concerned. Even though you get a working driver from somewhere else, installing and configuring it can be very cumbersome task [4].

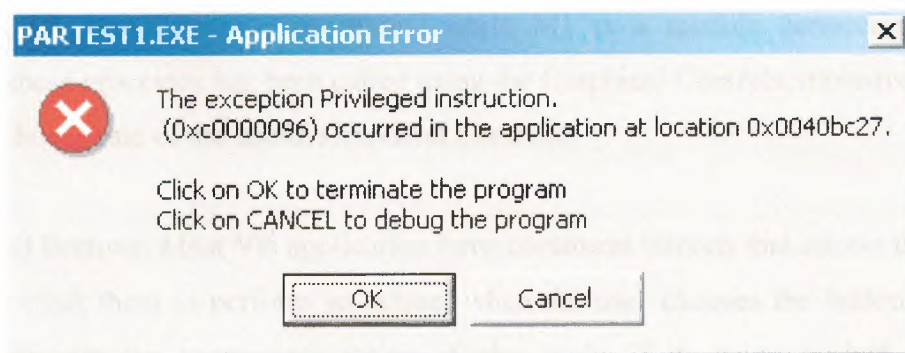


Figure 2.2 Error message box [4]

So the solution for that error by Introducing Inpout32.dll for WIN 98/NT/2000/XP. This dll have the following features

- 1) Works seamless with all versions of windows (WIN 98, NT, 200 and XP)
- 2) Using a kernel mode driver embedded in the dll
- 3) No special software or driver installation required
- 4) Driver will be automatically installed and configured automatically when the dll is loaded
- 5) No special APIs required only two functions Inp32 and Out32
- 6) Can be easily used with VC++ and VB

2.5 The GUI (Graphical User Interface)

The GUI is a kind of user interface, it might be a complex or simple user interface depending on the application. The GUI includes some hidden codes that generate the application as shown in figure2.3 we can imagine the main operation for this topic.

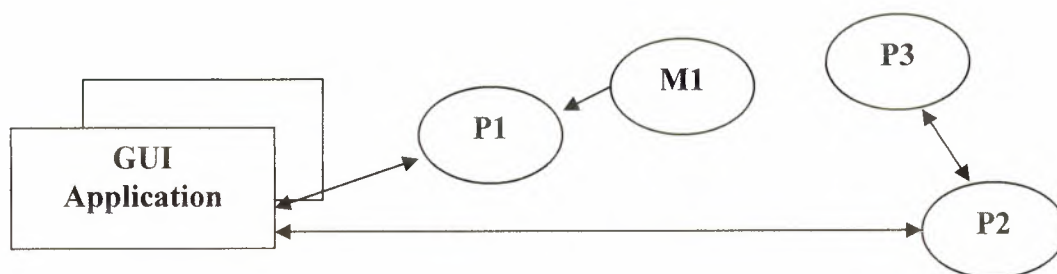


Figure 2.3 The GUI and the hidden process [5]

Here P1, P2, and P3 are processes, while M1 is a module connecting with process1, these processes have been called using the Graphical Controls, moreover, here we shall show some of the useful Graphical Controls:

-Command Buttons: Most VB application have command buttons that allows the user to simply click them to perform an action, when the user chooses the button, it not only carries out the appropriate action, it also looks if its being pushed in and relased. whenever the user clicks a button, the click event procedure is invoked.

-Text Boxes: Text Boxes are controls that can be used to get input from the user to display text.

-Message Boxes: Message Boxes display information in dialog box superimposed on the form

-List Boxes: the List Boxes display data in a list, they are used to output a data and display it either by sorting or without sorting, moreover, they are used for both numeric or string data type.

-Picture Boxes: they are used for displaying shapes or string separately of the main form, they have their own coordinates for controlling where the picture will be shown, and their own control.

-Labels: they are used to display some texts as a label, but they are nit used for input data.

2.5.1 The Developed GUI

The developed GUI is one of important parts of this project. The designed GUI is a simple one, with out any complicated ideas, the difficult things is how to write the code for that GUI, however, our GUI consists of:

- Command buttons
- Labels
- Timers
- Text boxes
- Check boxes
- Combo boxes

As shown in figure 2.4 we can see the main user interface that has been used.

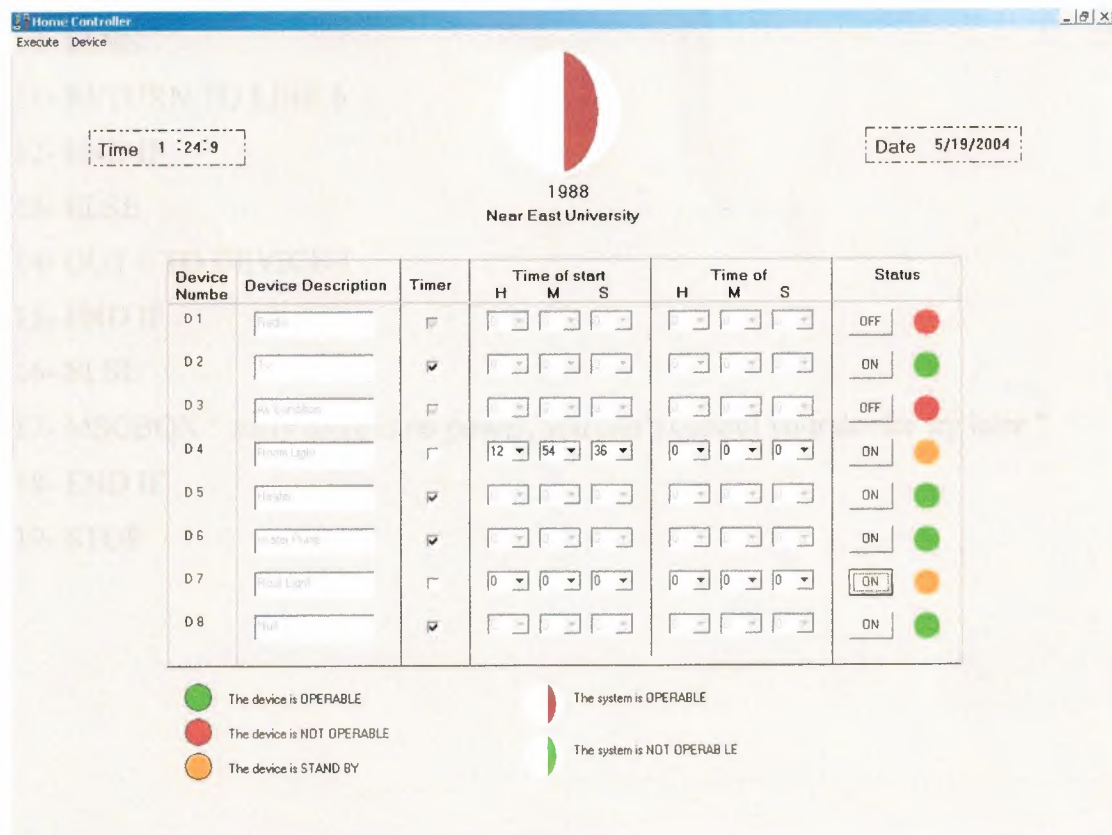


Figure 2.4 The Developed GUI

2.6 Turning the device ON/OFF manually

This algorithm used to control the device manually by clicking the corresponding command ON or OFF, the flowchart in figure 2.5 illustrates the mechanism of this algorithm.

```
1- BEGIN
2- CHECK FOR THE POWER
3- IF POWER =1 THEN
4- CHECKBOX1=CHECKED
5- CHECK FOR COMMAND1
6- IF COMMAND1=1 THEN
7- OUT 1 TO DEVICE 1
8- IF COMMAND1=0 THEN
9- OUT 0 TO DEVICE 1
10- ELSE
11- RETURN TO LINE 6
12- END IF
13- ELSE
14- OUT 0 TO DEVICE 1
15- END IF
16- ELSE
17- MSGBOX " sorry there is no power, you can't control your device try later "
18- END IF
19- STOP
```

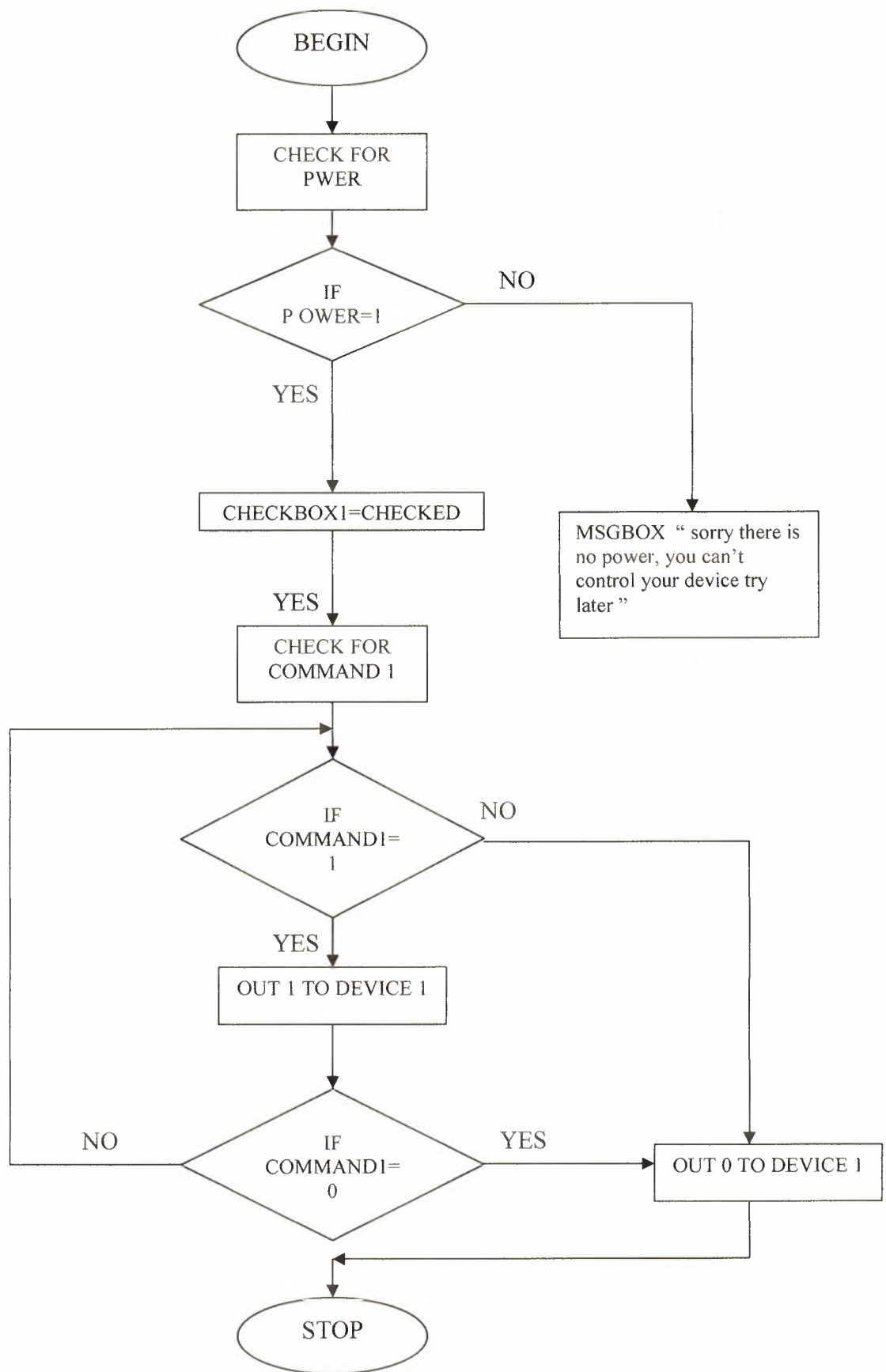
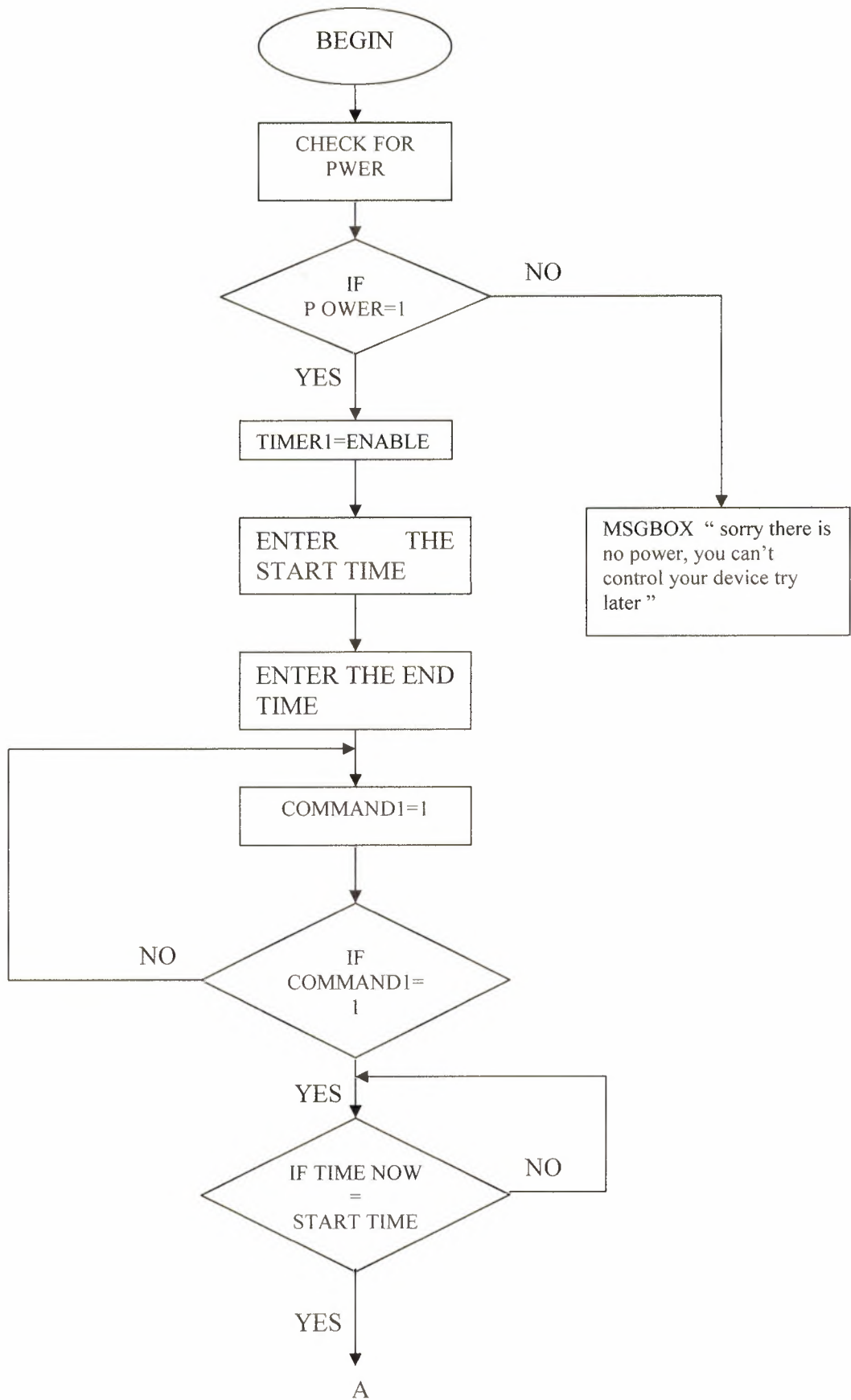



Figure 2.5 Turning the device ON/OFF manually flowchart

2.6.1 Turning the device ON/OFF automatically

This algorithm used to control the device automatically by setting the start time and end time by the user then activate the corresponding command, the flowchart in figure 2.6 illustrates the mechanism of this algorithm.

```
1- BEGIN
2- IF POWER = 1 THEN
3-  TIMER 1 =1
4-  ENTER THE START TIME
5-  ENTER THE END TIME
6-  COMMAND1 = 1
7-  IF COMMAND 1= 1 THEN
8-   IF TIME NOW = START TIME THEN
9-    DEVICE 1 = 1
10-  ELSE
11-   RETURN TO LINE 8
12- END IF
13- IF COMMAND 1= 0 THEN
14-  DEVICE 1 = 0
15- ELSE
16-  IF TIME NOW = END TIME THEN
17-   DEVICE = 0
18- ELSE
19-   RETURN TO LINE 13
20- END IF
21- ELSE
22-  RETURN TO LINE 6
23- END IF
24- END IF
25- ELSE
26-  MSGBOX “ sorry there is no power, you can’t control your device try later ”
27- END IF
```

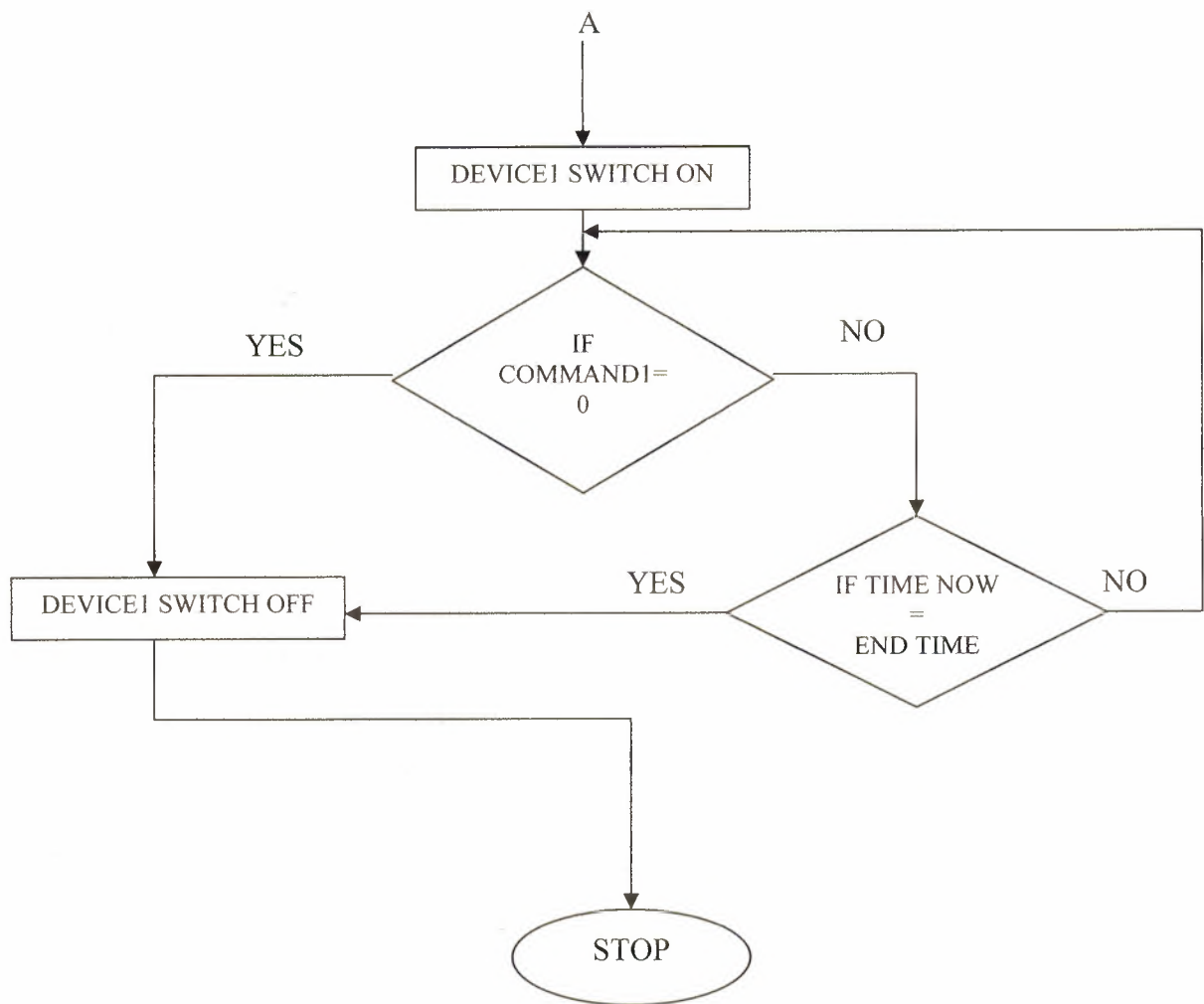
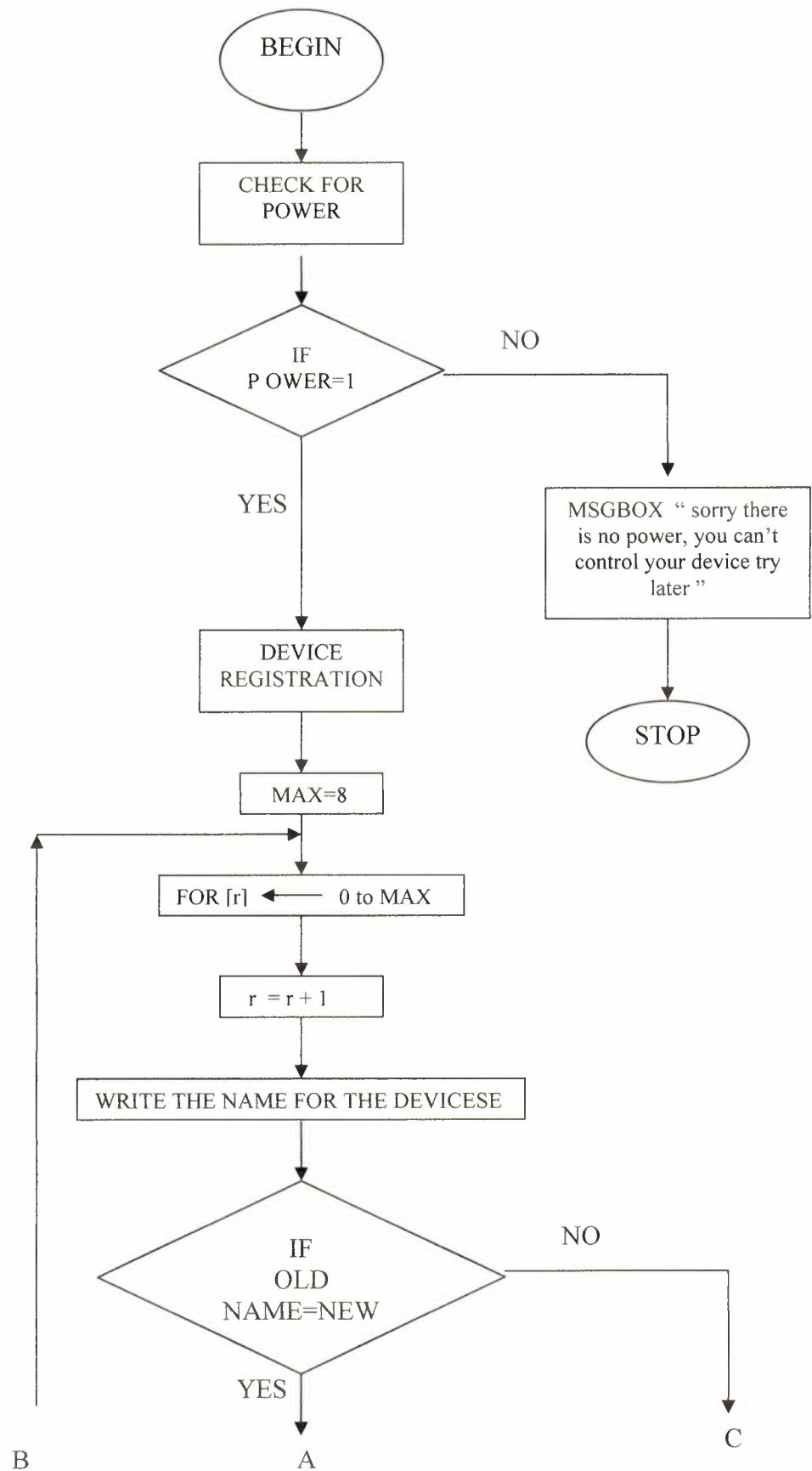


Figure 2.6 Turning the device ON/OFF automatically

2.6.2 Device Registrations

This algorithm used to named the devices and display it on the main form, the flowchart in figure 2.6 illustrates the mechanism of this algorithm.

```
1-BEGIN
2-CHECK FOR POWER
3-IF POWER = 1 THEN
4- DEVICE REGISTRATION
5- MAX=8
6- FOR [r] ← 0 to MAX
7- r = r + 1
8- WRITE THE NAME FOR THE DEVICES
9- IF OLD NAME=NEW NAME THEN
10- A= KEEP THE NAME WITHOUT CHANGE
11-ELSE
12- B = REPLACE THE OLD NAME BY THE NEW NAME
13- END IF
14- C = A OR B
15- IF r =MAX THEN
16- MSGBOX "you have named all the devices"
17-ELSE
18-RETURN TO LINE 6
19-END IF
20-ELSE
21- MSGBOX " sorry there is no power, you can't control your device try later "
22-END IF
23- STOP
```



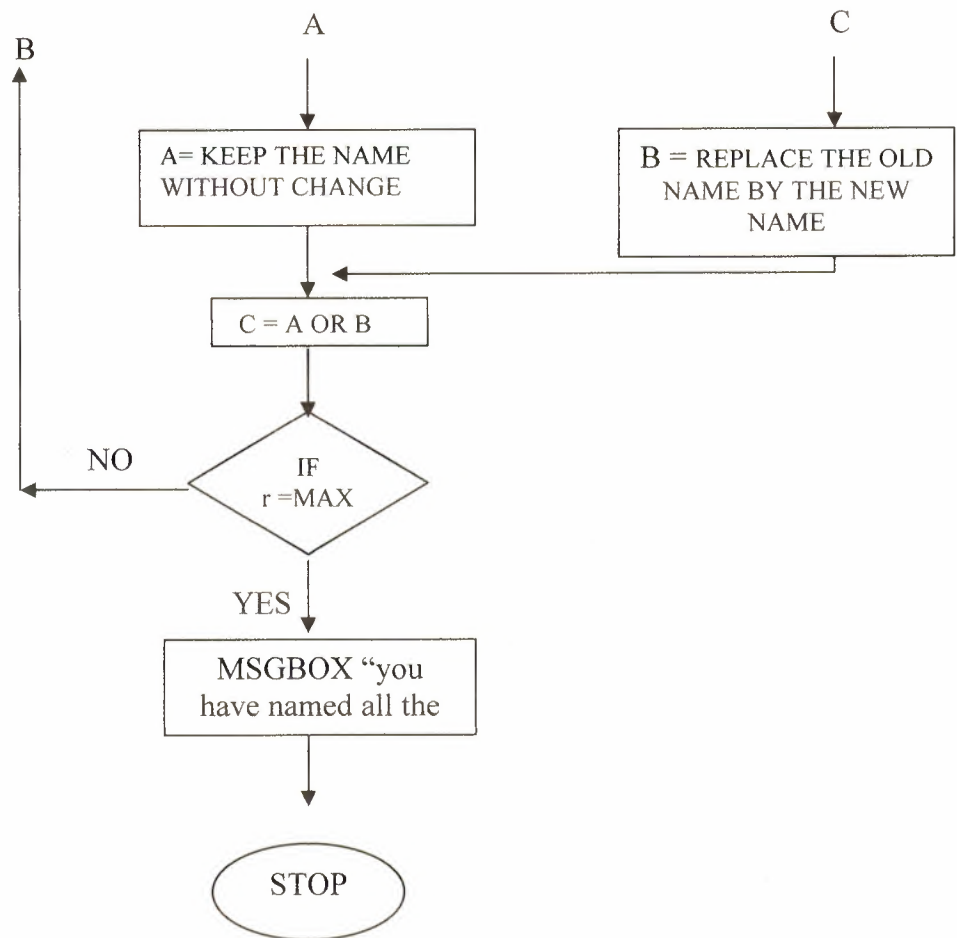


Figure 2.7 Device Registrations

2.7 Summary

In this chapter we have defined the following

- We have provided an introduction about visual basic language and the reason for using this language.
- Moreover, we have discussed the GUI, in the other hand we have shown the designed of GUI.
- Some of the technical specification have been introduced in this chapter
- Eventually, we have shown the main algorithms, flowcharts.

CHAPTER THREE

PROGRAMMING, INTERFACING AND USING THE PC's PARALLEL PORT

3.1 Overview

A first step in exploring the parallel port is learning how to get the most from a port with your everyday applications and peripherals. Things to know include how to find, configure, and install a port, how and when to use the new bi-directional, EPP, and ECP modes, and how to handle a system with multiple parallel port peripherals. This chapter presents essential information and tips relating to these topics.

3.2 Parallel Port Essentials

In the computer world, a port is a set of signal lines that the microprocessor, or CPU, uses to exchange data with other components. Typical uses for ports are communicating with printers, modems, keyboards, and displays, or just about any component or device except system memory. Most computer ports are digital, where each signal, or bit, is 0 or 1. A parallel port transfers multiple bits at once, while a serial port transfers a bit at a time (though it may transfer in both directions at once).

This chapter is about a specific type of parallel port: the one found on just about every PC, or IBM-compatible personal computer. Along with the RS-232 serial port, the parallel port is a workhorse of PC communications. On newer PCs, you may find other ports such as SCSI, USB, and IrDA, but the parallel port remains popular because it's capable, flexible, and every PC has one.

The term PC-compatible, or PC for short refers to the IBM PC and any of the many, many personal computers derived from it. From another angle, a PC is any computer that can run Microsoft's MS-DOS operating system and whose expansion bus is compatible with the ISA bus in the original IBM PC. The category includes the PC, XT, AT, PS/2, and most computers with 80x86, Pentium, and compatible CPUs.

It does not include the Macintosh, Amiga, or IBM mainframes, though these and other computer types may have ports that are similar to the parallel port on the PC.

The original PC's parallel port had eight outputs, five inputs, and four bi-directional lines. These are enough for communicating with many types of peripherals. On many newer PCs, the eight outputs can also serve as inputs, for faster communications with scanners, drives, and other devices that send data to the PC. The parallel port was designed as a printer port, and many of the original names for the port's signals (Paper End, Auto Linefeed) reflect that use. But these days, you can find all kinds of things besides printers connected to the port. The term peripheral, or peripheral device is a catchall category that includes printers, scanners, modems, and other devices that connect to a PC [6].

3.3 Port Types

As the design of the PC evolved, several manufacturers introduced improved versions of the parallel port. The new port types are compatible with the original design, but add new abilities, mainly for increased speed.

Speed is important because as computers and peripherals have gotten faster, the jobs they do have become more complicated, and the amount of information they need to exchange has increased. The original parallel port was plenty fast enough for sending bytes representing ASCII text characters to a dot matrix or daisy wheel printer. But modern printers need to receive much more information to print a page with multiple fonts and detailed graphics, often in color. The faster the computer can transmit the information, the faster the printer can begin processing and printing the result.

A fast interface also makes it feasible to use portable, external versions of peripherals that you would otherwise have to install inside the computer. A parallel-port tape or disk drive is easy to move from system to system, and for occasional use, such as making back-ups, you can use one unit for several systems. Because a backup may involve copying hundreds of Megabytes, the interface has to be fast to be worthwhile.

This chapter covers the new port types and brief explanations for each, the available types are [6]:

3.3.1 Original (SPP)

The parallel port in the original IBM PC, and any port that emulates the original port's design, is sometimes called the SPP, for standard parallel port, even though the original port had no written standard beyond the schematic diagrams and documentation for the IBM PC. Other names used are AT-type or ISA-compatible.

The port in the original PC was based on an existing Centronics printer interface. However, the PC introduced a few differences, which other systems have continued.

SPPs can transfer eight bits at once to a peripheral, using a protocol similar to that used by the original Centronics interface. The SPP doesn't have a byte-wide input port, but for PC-to-peripheral transfers, SPPs can use a Nibble mode that transfers each byte 4 bits at a time. Nibble mode is slow, but has become popular as a way to use the parallel port for input.

3.3.2 PS/2-Types (Simple Bi-directional)

An early improvement to the parallel port was the bi-directional data port introduced on IBM's model PS/2. The bi-directional port enables a peripheral to transfer eight bits at once to a PC. The term PS/2-type has come to refer to any parallel port that has a bi-directional data port but doesn't support the EPP or ECP modes described below. Byte mode is an 8-bit data-transfer protocol that PS/2-type ports can use to transfer data from the peripheral to the PC [6].

3.3.3 EPP

The EPP (enhanced parallel port) was originally developed by chipmaker Intel, PC manufacturer Zenith, and Xircom, a maker of parallel-port networking products. As on the PS/2-type port, the data lines are bi-directional. An EPP can read or write a

byte of data in one cycle of the ISA expansion bus, or about 1 microsecond, including handshaking, compared to four cycles for an SPP or PS/2-type port. An EPP can switch directions quickly, so it's very efficient when used with disk and tape drives and other devices that transfer data in both directions. An EPP can also emulate an SPP, and some EPPs can emulate a PS/2-type port [6].

3.3.4 ECP

The ECP (extended capabilities port) was first proposed by Microsoft. Like the EPP, the ECP is bi-directional and can transfer data at ISA-bus speeds. ECPs have buffers and support for DMA (direct memory access) transfers and data compression. ECP transfers are useful for printers, scanners, and other peripherals that transfer large blocks of data. An ECP can also emulate an SPP or PS/2-type port, and many ECPs can emulate an EPP as well [6].

3.3.5 Multi-Mode Ports

Many newer ports are multi-mode ports that can emulate some or all of the above types. They often include configuration options that can make all of the port types available, or allow certain modes while locking out the others [6].

3.4 Parallel Port Resources

The parallel port uses a variety of the computer's resources. Every port uses a range of addresses, though the number and location of addresses varies. Many ports have an assigned IRQ (interrupt request) level, and ECPs may have an assigned DMA channel. The resources assigned to a port can't conflict with those used by other system components, including other parallel ports [6]

3.4.1 Addressing

The standard parallel port uses three contiguous addresses, as shown in table 3.1, the first address is the port's base address, also called the Data register or just the port address. The second address is the port's Status register, and the third is the Control register.

EPPs and ECPs reserve additional addresses for each port. An EPP adds five registers at base address + 3 through base address + 7, and an ECP adds three registers at base address + 400h through base address + 402h. For a base address of 378h, the EPP registers are at 37Bh through 37Fh, and the ECP registers are at 778h through 77Fh.

On early PCs, the parallel port had a base address of 3BCh. On newer systems, the parallel port is most often at 378h. But all three addresses are reserved for parallel ports, and if the port's hardware allows it, you can configure a port at any of the addresses. However, you normally can't have an EPP at base address 3BCh, because the added EPP registers at this address may be used by the video display.

IBM's Type 3 PS/2 port also had three additional registers, at base address +3 through base address + 5, and allowed a base address of 1278h or 1378h. Most often, DOS and Windows refer to the first port in numerical order as LPT1, the second, LPT2, and the third, LPT3. So on boot up, LPT1 is most often at 378h, but it may be at any of the three addresses. LPT2, if it exists, may be at 378h or 278h, and LPT3 can only be at 278h. Various configuration techniques can change these assignments, however, so not all systems will follow this convention. LPT stands for line printer, reflecting the port's original intended use.

If your port's hardware allows it, you can add a port at any unused port address in the system. Not all software will recognize these non-standard ports as LPT ports, but you can access them with software that writes directly to the port registers [6].

Table 3.1 Locations and Functions

LPT NO.	DATA ADDRESS	STATUS ADDRESS	CONTROL ADDRESS	OPERATION
LPT 1	3BC h	3BD h	3BE h	Read / Write
LPT 2	378 h	379 h	37A h	Read only
LPT 3	278 h	279 h	27A h	Read / write

3.4.2 Interrupts

Most parallel ports are capable of detecting interrupt signals from a peripheral. The peripheral may use an interrupt to announce that it's ready to receive a byte, or that it has a byte to send. To use interrupts, a parallel port must have an assigned interrupt-request level (IRQ).

Conventionally, LPT1 uses IRQ7 and LPT2 uses IRQ5. But many sound cards use IRQ5, and because free IRQ levels can be scarce on a system, even IRQ7 may be reserved by another device. Some ports allow choosing other IRQ levels besides these two.

Many printer drivers and many other applications and drivers that access the parallel port don't require parallel-port interrupts. If you select no IRQ level for a port, the port will still work in most cases, though sometimes not as efficiently, and you can use the IRQ level for something else [6].

3.4.3 DMA Channels

ECPs can use direct memory access (DMA) for data transfers at the parallel port. During the DMA transfers, the CPU is free to do other things, so DMA transfers can result in faster performance overall. In order to use DMA, the port must have an assigned DMA channel, in the range 0 to 3 [6].

3.4.5 Finding Existing Ports

DOS and Windows include utilities for finding existing ports and examining other system resources. In Windows XP, click on Control Panel, System, Hardware, click Devices Manager button, move to Ports (COM&LPT), double click on it, then LPT port's address will be shown as figure 3.1 by right clicking on properties then Resources tab [6].

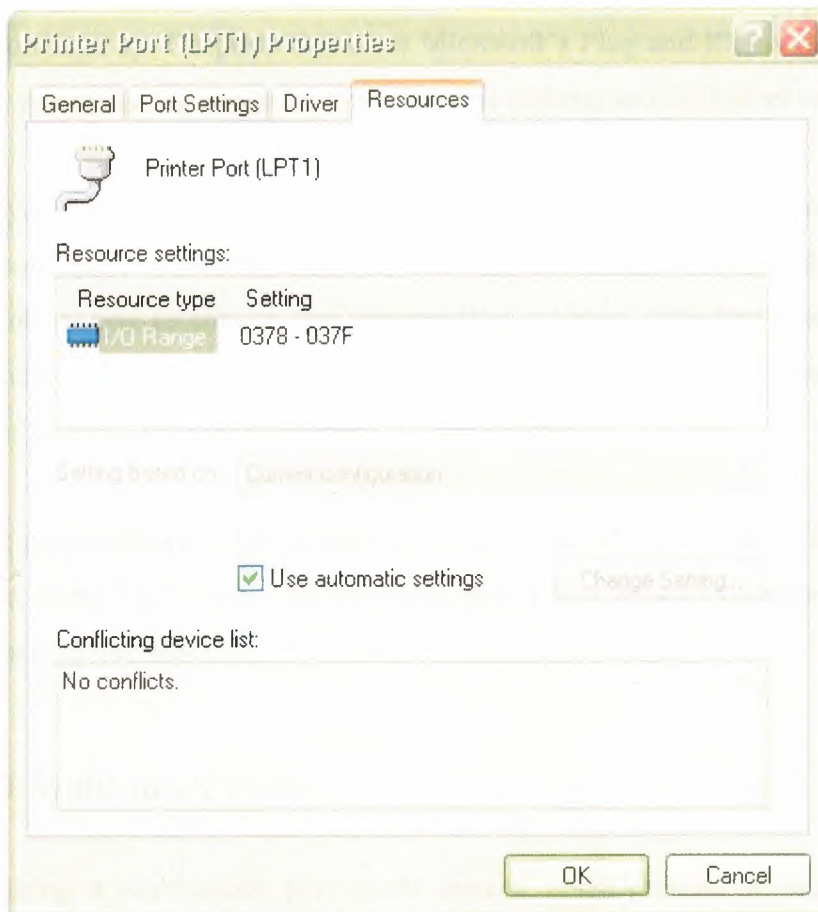


Figure 3.1 Printer Port (LPT1) Properties

3.5 Configuring

The parallel port that comes with a PC will have an assigned address and possibly an IRQ level and DMA channel. Multi-mode ports may also be configured with specific modes enabled. You can change some or all of these assignments to match your needs. If you're adding a new port, you need to configure it, making sure that it doesn't conflict with existing ports and other resources [6].

3.5.1 Port Options

There is no standard method for configuring a port. Some ports, especially older ones, use jumper blocks or switches to select different options. Others allow configuring in software, using a utility provided on disk. A port on a system motherboard may have

configuration options in the system setup screens (the CMOS setup) that you can access on boot up. On ports that meet Microsoft's Plug and Play standard, Windows XP can automatically assign an available port address and IRQ level to a port.

Check your system or port's documentation for specifics on how to configure a port. Some ports allow a choice of just one or two of the three conventional base addresses. A few allow you to choose any uncommitted address, including non-standard ones. On some boards, the jumpers or switches are labeled, which is extremely handy when you don't have other documentation (or can't find it).

If your port supports ECP transfers, assign it an IRQ level and DMA channel if possible. Most ECP drivers do use these, and if they're not available, the driver will revert to a slower mode.

3.5.1.1 Multi-mode Ports

Configuring a multi-mode port needs special consideration. A multi-mode port's controller chip supports a variety of modes that emulate different port types. In addition to the configuration options described above, on most multi-mode ports, you also have to select a port type to emulate.

The problem is that there is no single standard for the basic setup on the controller chips, and there are many different chips! Usually the setup involves writing to configuration registers in the chip, but the location and means of accessing the registers varies.

For this reason, every port should come with a simple way to configure the port. If the port is on the motherboard, look in the CMOS setup screens that you can access on boot up. Other ports may use jumpers to enable the modes, or have configuration software on disk.

The provided setup routines don't always offer all of the available options or explain the meaning of each option clearly. For example, one CMOS setup I've seen allows only the choice of AT or PS/2-type port. The PS/2 option actually configures the port as an ECP, with the ECP's PS/2 mode selected, but there is no documentation

explaining this. The only way to find out what mode is actually selected is to read the chip's configuration registers. And although the port also supports EPP, the CMOS setup includes no way to enable it, so again, accessing the configuration registers is the only option.

If your port is EPP or ECP capable but the setup utility doesn't offer these as choices, a last resort is to identify the controller chip, obtain and study its data sheet, and write your own program to configure the port.

The exact terminology and the number of available options can vary, but these are typical configuration options for a multi-mode port:

SPP. Emulates the original port. Also called AT-type or ISA-compatible.

PS/2, or simple bi-directional. Like an SPP, except that the data port is bi-directional.

EPP. Can do EPP transfers. Also emulates an SPP. Some EPPs can emulate a PS/2 type port.

ECP. Can do ECP transfers. The ECP's internal modes enable the port to emulate an SPP or PS/2-type port. An additional internal mode, Fast Centronics, or Parallel Port FIFO, uses the ECP's buffer for faster data transfers with many old-style (SPP) peripherals.

ECP + EPP. An ECP that supports the ECP's internal mode 100, which emulates an EPP. The most flexible port type, because it can emulate all of the others.

3.6 Port Hardware

The parallel port's hardware includes the back-panel connector and the circuits and cabling between the connector and the system's expansion bus. The PC's microprocessor uses the expansion bus's data, address, and control lines to transfer information between the parallel port and the CPU, memory, and other system components.

3.6.1 Connectors

The PC's back panel has the connector for plugging in a cable to a printer or other device with a parallel-port interface. Most parallel ports use the 25-contact D-sub

connector shown in Figure 3.2. Other names for this connector are the subminiature D, DB25, D-shell, or just D connector. The IEEE 1284 standard for the parallel port calls it the IEEE 1284-A connector.

The parallel-port connector is usually the only female 25-pin D-sub on the back panel, so there should be little confusion with other connectors. Some serial ports use a 25-contact D-sub, but with few exceptions, a 25-pin serial D-sub on a PC is male, with the female connector on the cable the reverse of the parallel-port convention. (Other serial ports use 9-pin D-sub instead.).

The pin outs of DB25 (LPT) connector is shown in Figure 3.2, The lines in DB25 connector are divided in to three groups, they are

- Data lines (data bus)
- Control lines
- Status lines

As the name refers, data is transferred over data lines, Control lines are used to control the peripheral and of course, the peripheral returns status signals back computer through Status lines. These lines are connected to Data, Control And Status registers internally. The details of parallel port signal lines are shown in table 3.2 [7].

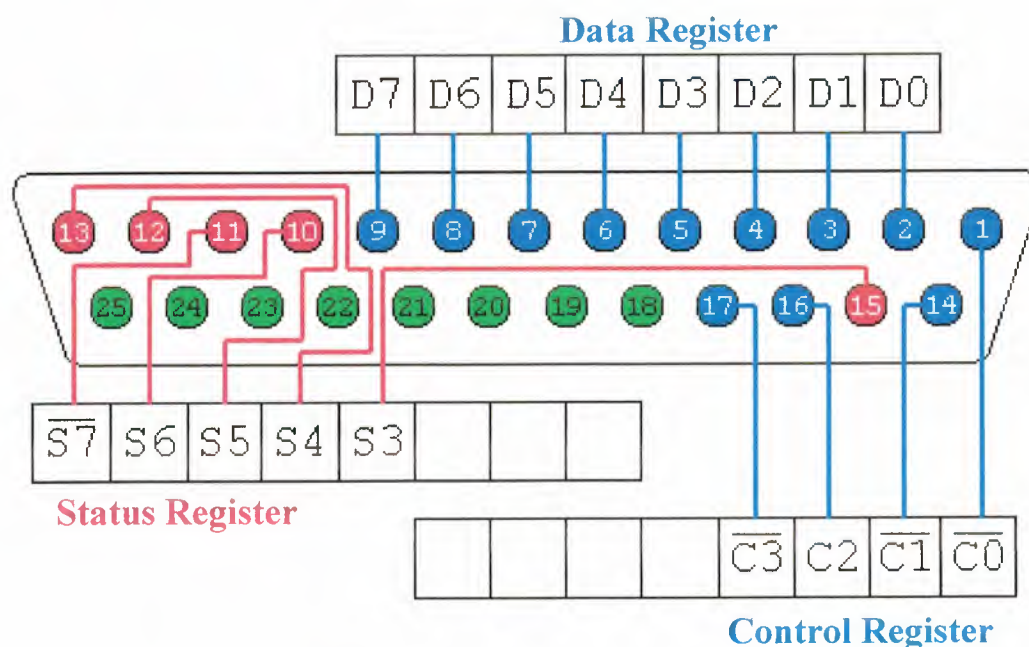


Figure 3.2 The pin outs of DB25 [7]

Table 3.2 Pin assignment of the D-type 25 pin parallel port connector [4]

Pin No (D_Type 25)	SPP Signal	Direction In/ Out	Register	Hardware Inverted
1	nStrobe	IN/OUT	Control	Yes
2	Data 0	OUT	Data	
3	Data 1	OUT	Data	
4	Data 2	OUT	Data	
5	Data 3	OUT	Data	
6	Data 4	OUT	Data	
7	Data 5	OUT	Data	
8	Data 6	OUT	Data	
9	Data 7	OUT	Data	
10	nAek	IN	Status	
11	Busy	IN	Status	Yes
12	P-out/End	IN	Status	
13	Select	IN	Status	
14	nAuto-Linefeed	IN/OUT	Control	Yes
15	nError/nFault	IN	Status	
16	nInitialize	IN/OUT	Control	
17	nSelect	IN/OUT	Control	Yes
18_25		GROUND		

3.6.2 The Circuits Inside

Inside the computer, the parallel-port circuits may be on the motherboard or on a card that plugs into the expansion bus.

The motherboard is the main circuit board that holds the computer's microprocessor chip as well as other circuits and slots for expansion cards. Because just about all

computers have a parallel port, the port circuits are often right on the motherboard, freeing the expansion slot for other uses. Notebook and laptop computers don't have expansion slots, so the port circuits in these computers must reside on the system's main circuit board.

The port circuits connect to address, data, and control lines on the expansion bus, and these in turn interface to the microprocessor and other system components.

3.6.3 Cables

Most printer cables have a 25-pin male D-sub connector on one end as shown in Figure 3.2 and a male 36-contact connector on the other. Many refer to the 36-contact connector as the Centronics connector, because it's the same type formerly used on Centronics printers. Other names are parallel-interface connector or just printer connector. IEEE 1284 calls it the 1284-B connector.

Peripherals other than printers may use different connectors and require different cables. Some use a 25-pin D-sub like the one on the PC. A device that uses only a few of the port's signals may use a telephone connector, either a 4-wire RJ11 or an 8-wire RJ45. Newer peripherals may have the 36-contact 1284-C connector. In any case, because the parallel-port's outputs aren't designed for transmitting over long distances, it's best to keep the cable short: 6 to 10 feet, or 33 feet for an IEEE-1284-compliant cable [6].

3.7 Multiple Uses for One Port

If you have more than one parallel-port peripheral, the easiest solution is to add a port for each. But there may be times when multiple ports aren't an option. In this case, the alternatives are to swap cables as needed, use a switch box, or daisy-chain multiple devices to one port.

If you use only one device at a time and switch only occasionally, it's easy enough to move the cable when you want to use a different device. For frequent swapping, a more convenient solution is a switch box. A typical manual switch box has three

female D-sub connectors. A switch enables you route the contacts of one connector to either of the others. To use the switch box to access two peripherals on one port, you'll need a cable with two male D-sub connectors to connect the PC to the switch box, plus an appropriate cable from the switch box to each peripheral.

You can also use a switch box to enable two PCs to share one printer or other peripheral. This requires two cables with two male D-sub connectors on each, and one peripheral cable. Switch boxes with many other connector types are also available.

Manual switches are inexpensive, though some printer manufacturers warn that using them may damage the devices they connect to. A safer choice is a switch that uses active electronic circuits to route the signals. Some auto-sensing switches enable you to connect multiple computers to one printer, with first-come, first-served access. When a printer is idle, any computer can access it. When the printer is in use, the switch prevents the other computers from accessing it. However, these switches may not work properly if the peripherals use bi-directional communications, or if the peripheral uses the control or status signals in an unconventional way.

The parallel ports on some newer peripherals support a daisy-chain protocol that allows up to eight devices to connect to a single port. The PC assigns a unique address to each peripheral, which then ignores communications intended for the other devices in the chain. The software drivers for these devices must use the protocol when they access the port.

3.7.1 Security Keys

Security keys, or dangles, are a form of copy protection that often uses the parallel port. Some software usually expensive, specialized applications include a security key that you must plug into the parallel port in order to run the software. If you don't have the key installed, the software won't run.

The key is a small device with a male D-sub connector on one end and a female D-sub on the other. You plug the key into the parallel-port connector, and then plug your regular cable into the security key. When the software runs, it attempts to find and communicate with the key, which contains a code that the software recognizes. The

key usually doesn't use any conventional handshaking signals, so it should be able to live in harmony with other devices connected to the port.

The keys do require power, however. If you have a key that draws more than a small amount of current, and if your parallel port has weak outputs, you may have problems in using other devices on the same port as the key.

3.8 Alternatives to the Parallel Port

The parallel port is just one of many ways to interface inputs and outputs to a computer. In spite of its many virtues, the parallel port isn't the best solution for every project. These are some of the alternatives:

3.8.1 Other Parallel Interfaces

SCSI and IEEE-488 are two other parallel interfaces used by some PCs.

3.8.1.1 SCSI

SCSI (small computer system interface) is a parallel interface that allows up to seven devices to connect to a PC along a single cable, with each device having a unique address. Many computers use SCSI for interfacing to internal or external hard drives, tape backups, and CD-ROMs. SCSI interfaces are fast, and the cable can be as long as 19 feet (6 meters). But the parallel-port interface is simpler, cheaper, and much more common [6].

3.8.1.2 IEEE 488

The IEEE-488 interface began as Hewlett Packard's GPIB (general-purpose interface bus). It's a parallel interface that enables up to 15 devices to communicate at speeds of up to 1 Megabit per second. This interface has long been popular for interfacing to lab instruments. Expansion cards with IEEE-488 interfaces are available [6].

3.8.2 Serial Interfaces

One large group of parallel-port alternatives is serial interfaces, where data bits travel on a single wire or pair of wires (or in the case of wireless links, a single transmission path.) Both ends of the link require hardware or software to translate between serial and parallel data. There are many types of serial interfaces available for PCs, ranging from the ubiquitous RS-232 port to the newer RS-485, USB, IEEE-1394, and IrDA [6].

3.9 Summary

In this chapter we defined the parallel port (LPT), which we have in all IBM compatible PC's. The D-Type 25 pin male connector, which has 5 status lines and 8 data lines. It's found commonly on the back of your PC as a D-Type 25 Pin female and we mentioned a bout SPP, PS2, EPP, and ECP as types of parallel port. Also the three reserved address for the port 3BC, 378, 278. Finally we discussed a bout SCSI and IEEE 488 as parallel interfaces that we can use rather than parallel port.

CHAPTER FOUR

CIRCUIT COMPONENTS

4.1 OVERVIEW

In this chapter we will describe the characteristics of the components which we used in the outside device to build the circuit and to make the connection between the computer and outside device. We will define the main characteristics of the Resisters, diodes, transistors and relays.

4.2 INTRODUCTION TO ELECTRONICS

Electrons and protons have the electrical property of charge. Protons have positive charge and electrons have negative charge and they normally balance each other out. We don't really need to know what charge is. It's just a property like weight or color, but it is this property which makes the whole of electronics happens. But keep in mind the fact that opposite charges attract and similar charges repel.

When electrons move together in a unified way we say there is a current flowing. Electrons are actually moving all the time in materials like metals but moving in a random disordered way. A current is when they all move together in one particular direction.

When you touch a lift button having walked across a synthetic carpet and you feel a shock that is electrons flowing through you to the ground. That's all a current is, simply the movement of electrons in a particular direction.

Electrons can't flow through every material. Materials that allow a current to flow easily are called conductors. Materials that don't allow a current to flow are called non-conductors or insulators. Metals are the most common conductors, plastics are typical insulators.



Conductor's	non-conductors
-------------	----------------

Gold	plastic
Copper	wood
Carbon	air

Copper is a good conductor. Copper tracks are used on the printed circuit boards to connect the components together. Solder is another good conductor. The solder makes the actual join between the leg of the component and the track.

The plastic that a printed circuit board is made of is an insulator. Currents can only flow up and down the copper tracks and not jump from one to another. For the same reason wires are surrounded by plastic coatings to stop them conducting where they shouldn't.

There are certain materials that are between the two extremes of conductor and non-conductor; we will come to them later.

A battery supplies the 'force' that makes the electrons move. This force is called the voltage. The bigger the voltage the more force. Mains electricity which is 240 volts is more powerful than an ordinary 9 volt battery.

Currents are measured in amps, and voltages are measured in volts (after the scientists Ampere and Volta). Voltages are sometimes called potential differences, or electromotive forces, but we won't use these terms here.

There is a big confusion for many people as to the difference between voltage and current. They talk about so many volts going through something when they really mean amps. So let's think about things in a different way.

Imagine water flowing through a pipe filling up a pond. The water represents the electrons and the pipe represents the wire. A pump provides the pressure to force the water through the pipe. The pump is the battery. How much water flows out the end of the pipe each second is the current. How hard the water is being pumped is the voltage.

A narrow pipe will take a long time to fill the pond, whereas a broad pipe will do it much faster using the same pump. Clearly the rate of flow depends on the thickness of the pipe. So we have the situation where the same voltage (pump pressure) can give rise to different currents (flow rate) depending on the pipe. Try to guess what the thickness of the pipe represents in this model of things (answer later).

An electric current requires a complete path - a circuit - before it can flow. In a circuit with a battery, the battery is both the starting flag and the finishing line for the electrons. A chemical reaction in the battery releases electrons which flow around the circuit and then back into the battery. The battery keeps the current flowing, feeding electrons in at one end and collecting them at the other. It takes energy to do this and so after a while the battery wears out.

Current flows into a component and the same amount of current always flows out of the component. It is not 'used up' in any way. As the current passes through components things happen (an LED lights up for instance).

4.3 Components

The main problem for someone starting electronics as a hobby is simply that of identifying the various components used in projects. Before proceeding to the projects a brief description of the components used in them will be given so that even a complete beginner should have no difficulty in sorting out which component is which, and connecting each component into circuit correctly.

4.3.1 Resistors

Electrons move more easily through some materials than others when a voltage is applied. In metals the electrons are held so loosely that they move almost without any hindrance. We measure how much opposition there is to an electric current as resistance.

Resistors come somewhere between conductors, which conduct easily, and insulators, which don't conduct at all. Resistance is measured in ohms after the discoverer of a law relating voltage to current. Ohms are represented by the Greek letter omega.

Think back to the model of water flowing in a pipe. The thickness of the pipe must represent the resistance. The narrower the pipe the harder it is for the water to get through and hence the greater the resistance. For a particular pump the time taken to fill the pond is directly related to the pipe thickness. Make the pipe twice the size and the flow rate doubles, and the pond fills in half the time.

The resistors used in the kits are made of a thin film of carbon deposited on a ceramic rod. The less carbon the higher the resistance. They are then given a tough outer coating and some colored bands are painted on.

The main function of resistors in a circuit is to control the flow of current to other components. Take an LED (light) for example. If too much current flows through an LED it is destroyed. So a resistor is used to limit the current.

When a current flows through a resistor energy is wasted and the resistor heats up. The greater the resistance the hotter it gets. The battery has to do work to force the electrons through the resistor and this work ends up as heat energy in the resistor.

An important property to know about a resistor is how much heat energy it can withstand before it's damaged. resistors can dissipate about a 1/4 Watt of heat (compare this with a domestic kettle which uses up to 3 000 Watts to boil water).

It's difficult to make a resistor to an exact value (and in most circuits it is not critical anyway). Resistances are given with a certain accuracy or tolerance. This is expressed as being plus or minus so much of a percentage. A 10% resistor with a stated value of 100 ohms could have a resistance anywhere between 90 ohms and 110 ohms. The resistors are 5% (that's what the gold band means) which is more than enough accuracy.

Real resistances vary over an enormous range. In the Lie Detector there is a 1 000 000 ohms resistor alongside a 470 ohms resistor. In circuit diagrams you will often see an 'R' instead of omega to represent ohms. This is a convention that dates from before the

days of computers and laser printers when Greek letters were rarely found on typewriters. The letter 'k' means a thousand and its position shows the position of the decimal point.

Here are some examples:

10R = 10 ohms

10k = 10 kilohms = 10 000 ohms

4k7 = 4.7 kilohms = 4 700 ohms

4.3.1.1 Fixed value resistors

During manufacture, a thin film of carbon is deposited onto a small ceramic rod. The resistive coating is spiraled away in an automatic machine until the resistance between the two ends of the rod is as close as possible to the correct value. Metal leads and end caps are added; the resistor is covered with an insulating coating and finally painted with colored bands to indicate the resistor value

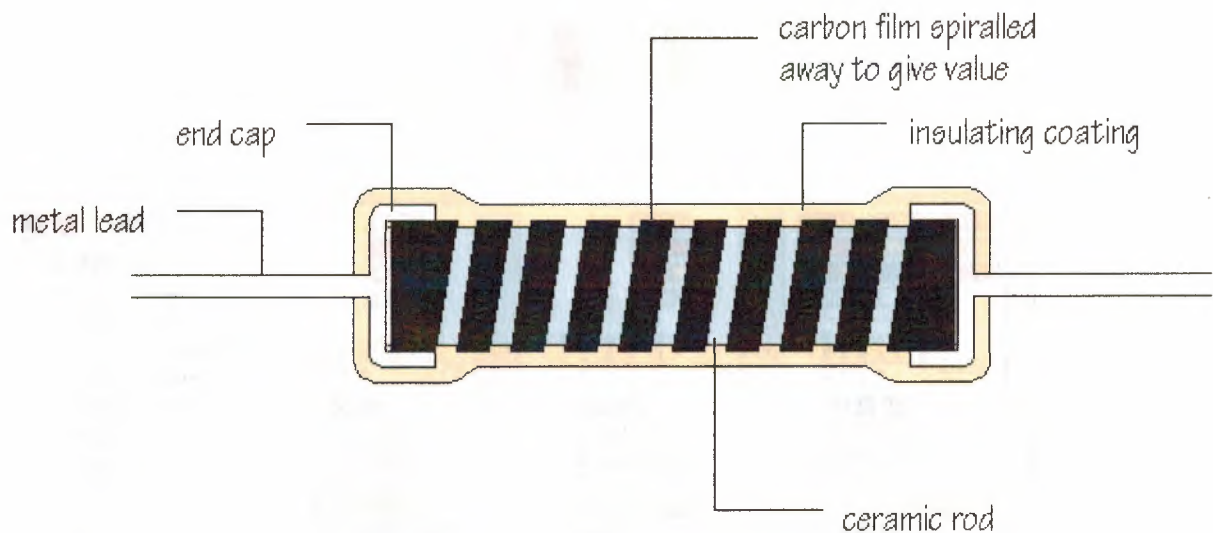


Figure 4.1 The diagram shows the construction of a carbon film resistor[8].

Carbon film resistors are cheap and easily available, with values within $\pm 10\%$ or $\pm 5\%$ of their marked, or 'nominal' value. Metal film and metal oxide resistors are made in a

similar way, but can be made more accurately to within $\pm 2\%$ or $\pm 1\%$ of their nominal value. There are some differences in performance between these resistor types, but none which affect their use in simple circuits.

Wire wound resistors are made by winding thin wire onto a ceramic rod. They can be made extremely accurately for use in multimeters, oscilloscopes and other measuring equipment. Some types of wire wound resistors can pass large currents without overheating and are used in power supplies and other high current circuits.

4.3.1.2 Resistor Color Code

The resistor color code is a way of showing the value of a resistor. Instead of writing the resistance on its body, which would often be too small to read, a color code is used. Ten different colors represent the numbers 0 to 9. The first two colored bands on the body are the first two digits of the resistance, and the third band is the 'multiplier'. Multiplier just means the number of zeroes to add after the first two digits. Red represents the number 2, so a resistor with red, red, red bands has a resistance of 2 followed by 2 followed by 2 zeroes, which is 2 200 ohms or 2.2 kilohms.

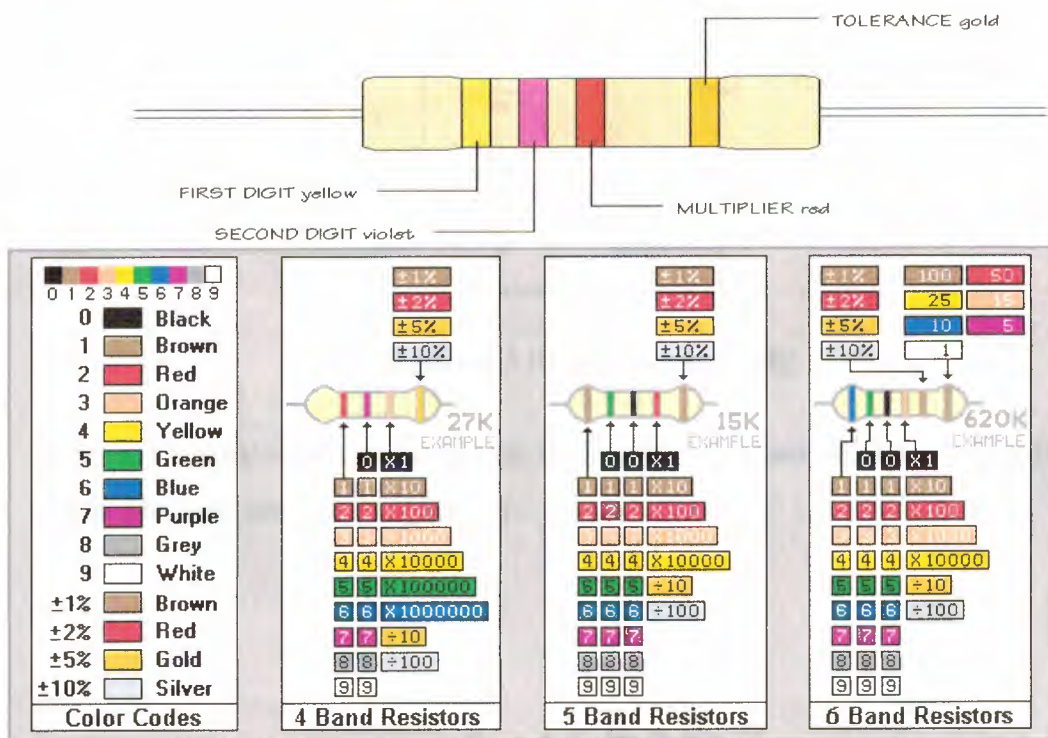


Figure 4.2 Color code identification [8].

While these codes are most often associated with resistors, then can also apply to capacitors and other components.

The standard color coding method for resistors uses a different color to represent each number 0 to 9: black, brown, red, orange, yellow, green, blue, purple, grey, white. On a 4 band resistor, the first two bands represent the significant digits. On a 5 and 6 band, the first three bands are the significant digits. The next band represents the multiplier or "decade".

4.3.1.3 Resistors In Series and Parallel

In a series circuit, the current flowing is the same at all points. The circuit diagram shows two resistors connected in series with a 6 V battery:

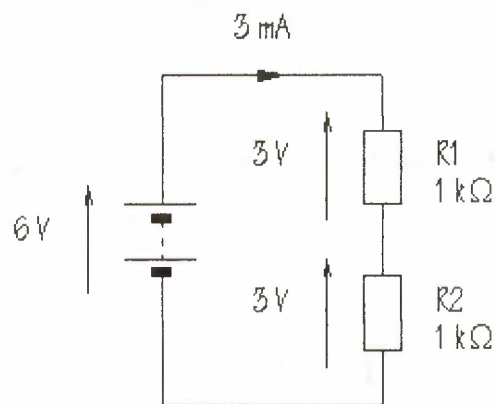


Figure 4.3 Resistors in series[8].

It doesn't matter where in the circuit the current is measured; the result will be the same. The total resistance is given by:

$$R_{\text{total}} = R_1 + R_2$$

The next circuit shows two resistors connected in parallel to a 6 V battery:

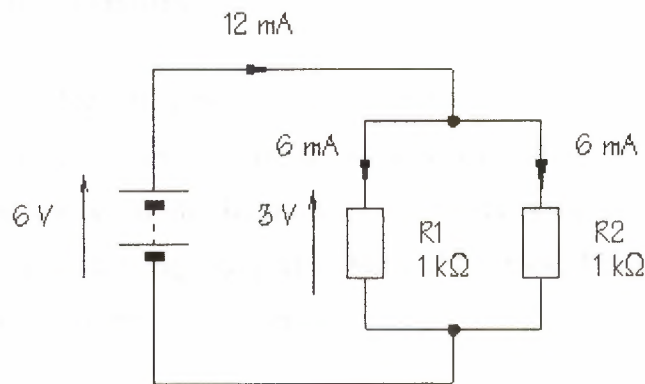


Figure 4.4 Resistors in parallel[8].

Parallel circuits always provide alternative pathways for current flow. The total resistance is calculated from:

$$R_{\text{total}} = \frac{R_1 \times R_2}{R_1 + R_2}$$

This is called the product over sum formula and works for any *two* resistors in parallel. An alternative formula is:

$$\frac{1}{R_{\text{total}}} = \frac{1}{R_1} + \frac{1}{R_2}$$

This formula can be extended to work for more than two resistors in parallel, but lends itself less easily to mental arithmetic. Both formulae are correct.

4.3.1.4 Variable Resistors

Unsurprisingly, variable resistors are resistors whose resistance can be varied. The variable resistors (called presets) have a metal wiper resting on a circular track of carbon. The wiper moves along the track as the preset is turned. The current flows through the wiper and then through part of the carbon track. The more of the track it has to go through the greater the resistance.

Presets have three legs. The top leg connects to the wiper and the other two legs to the two ends of the track. Generally only one of the track legs is actually used.

Variable resistors are used in circuits to vary things that need changing, like volume etc.

4.3.2 Semiconductors

Now we come to what is probably the most important discovery in electronics this century. Without this discovery we wouldn't have televisions, computers, space rockets or transistor radios. Unfortunately it's also one of the hardest areas to understand in electronics. But don't lose heart, read the section through a few times until you've grasped the ideas.

Recall that the reason that metals are such good conductors is that they have lots of electrons, which are so loosely held that they're easily able to move when a voltage is applied. Insulators have fixed electrons and so are not able to conduct. Certain materials, called semiconductors, are insulators that have a few loose electrons. They are partly able to conduct a current.

The free electrons in semiconductors leave behind a fixed positive charge when they move about (the protons in the atoms they come from). Charged atoms are called ions. The positive ions in semiconductors are able to capture electrons from nearby atoms.

current through a semiconductor, electrons moving in one direction and holes in the other. There are two kinds of current carriers.

The holes don't really move of course. It is just fixed positive ions grabbing neighboring electrons, but it appears as if holes are moving.

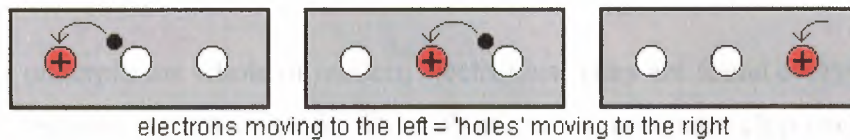


Figure 4.5 Moving of electrons[8].

In a pure semiconductor there are not enough free electrons and holes to be of much use. Their number can be greatly increased however by adding an impurity, called a donor. If the donor gives up some extra free electrons we get an n-type semiconductor (n for negative). If the donor soaks up some of the free electrons we get a p-type semiconductor (p for positive). In both cases the impurity donates extra current carriers to the semiconductor.

In n-type semiconductors there are more electrons than holes and they are the main current carriers. In p-type semiconductors there are more holes than electrons and they are the main current carriers. The donor atoms become either positive ions (n-type) or negative ions (p-type).

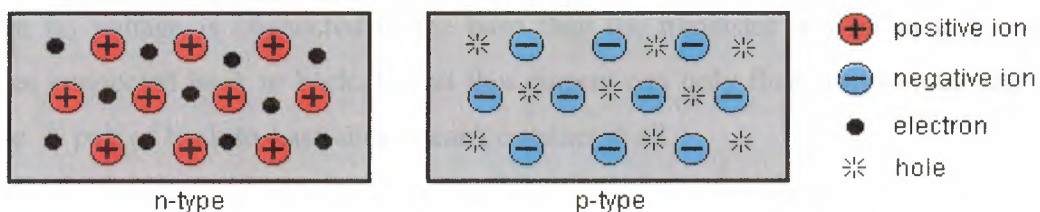


Figure 4.6 The two types of semiconductors[8].

The most common semiconductors are silicon (basically sand) and germanium. Common donors are arsenic and phosphorus.

When we combine n-type and p-type semiconductors together we make useful devices, like transistors and diodes and silicon chips.

4.3.2.1 Transistors

Transistors underpin the whole of modern electronics. They are found everywhere - in watches, calculators, microwaves, hi-fi's. A Pentium(tm) computer chip contains over a million transistors!

Transistors work in two ways. They can work as switches (turning currents on and off) and as amplifiers (making currents bigger). We'll only be looking at them as switches here. To understand them as amplifiers would involve a little mathematics.

Transistors are sandwiches of three pieces of semiconductor material. A thin slice of n-type or p-type semiconductor is sandwiched between two layers of the opposite type. This gives two junctions rather than the one found in a diode. If the thin slice is n-type the transistor is called a p-n-p transistor, and if the thin slice is p-type it is called a n-p-n transistor. The middle layer is always called the base, and the outer two layers are called the collector and the emitter.

We will consider the (more common) n-p-n transistor here, as used in the circuits. In a n-p-n transistor electrons are the main current carriers (because n-type material predominates).

When no voltage is connected to the base then the transistor is equivalent to two diodes connected back to back. Recall that current can only flow one way through a diode. A pair of back-to-back diodes can't conduct at all.

If a small voltage is applied to the base (enough to remove the depletion layer in the lower junction), current flows from emitter to base like a normal diode. Once current is flowing however it is able to sweep straight through the very thin base region and into the collector. Only a small part of the current flows out of the base. The transistor

is flowing however it is able to sweep straight through the very thin base region and into the collector. Only a small part of the current flows out of the base. The transistor is now conducting through both junctions. A few of the electrons are consumed by the holes in the p-type region of the base, but most of them go straight through.

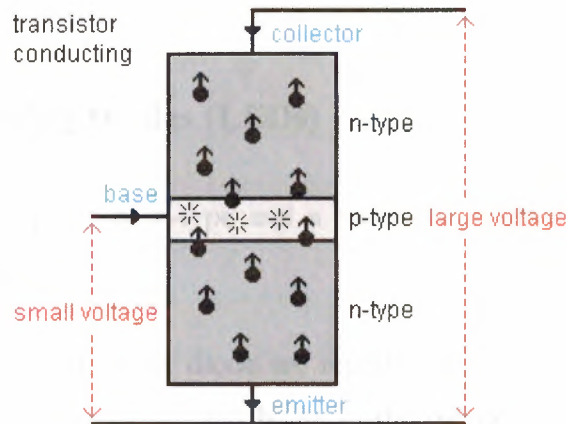


Figure4.7 Transistor conducting[8].

The difference between PNP and NPN transistors is that NPN use electrons as carriers of current and PNP use a lack of electrons (known as "holes"). Basically, nothing moves very far at a time. One atom simply robs an electron from an adjacent atom so you get the impression of "flow". It's a bit like "light pipes". In the case of "N" material, there are lots of spare electrons. In the case of "P" there aren't. In fact "P" is gasping for electrons.

Electrons enter the emitter from the battery and come out of the collector. (Isn't that rather illogical you might say, electrons emitted from the collector? Yes it is, but the parts of a transistor are named with respect to conventional current, an imaginary current which flows in the opposite direction to real electron current.)

Now we can see how a transistor acts as a switch. A small voltage applied to the base switches the transistor on, allowing a current to flow in the rest of the transistor.



This is the symbol used to represent an "NPN" transistor.

You can distinguish this from a "PNP" transistor (right) by the arrow which indicates current flow direction.



Figure 4.8 The difference between PNP and NPN transistors[8].

4.3.2.2 Light Emitting Diodes (LEDs)

A diode consists of a piece of n-type and a piece of p-type semiconductor joined together to form a junction.

Electrons in the n-type half of the diode are repelled away from the junction by the negative ions in the p-type region, and holes in the p-type half are repelled by the positive ions in the n-type region. A space on either side of the junction is left without either kind of current carriers. This is known as the depletion layer. As there are no current carriers in this layer no current can flow. The depletion layer is, in effect, an insulator.

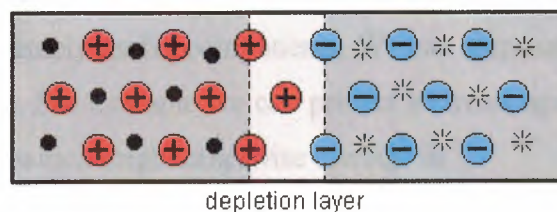


Figure 4.9 Depletion layer[8].

Now consider what would happen if we connected a small voltage to the diode. Connected one way it would attract the current carriers away from the junction and make the depletion layer wider. Connected the other way it would repel the carriers and drive them towards the junction, so reducing the depletion layer. In neither case would any current flow because there would always be some of the depletion layer left.

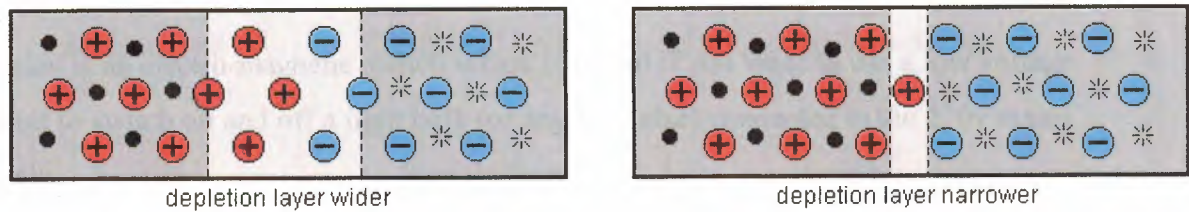


Figure 4.10 Reducing the depletion layer[8].

Now consider increasing the voltage. In one direction there is still no current because the depletion layer is even wider, but in the other direction the layer disappears completely and current can flow. Above a certain voltage the diode acts like a conductor. As electrons and holes meet each other at the junction they combine and disappear. The battery keeps the diode supplied with current carriers.

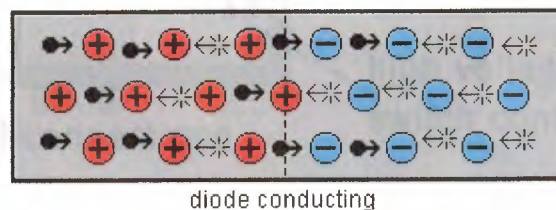


Figure 4.11 Diode conducting[8].

Thus a diode is a device, which is an insulator in one direction and a conductor in the other. Diodes are extremely useful components. We can stop currents going where we don't want them to go. For example we can protect a circuit against the battery being connected backwards which might otherwise damage it.

Light emitting diodes (LEDs) are special diodes that give out light when they conduct. The fact that they only conduct in one direction is often incidental to their use in a circuit. They are usually just being used as lights. They are small and cheap and they last practically forever, unlike traditional light bulbs which can burn out.

The light comes from the energy given up when electrons combine with holes at the junction. The color of the light depends on the impurity in the semiconductor. It is easy to make bright red, green and yellow LEDs but technology has not cracked the problem of making cheap blue LEDs yet.

4.3.3 Relay Driver

A relay is an electro-magnetic switch which is useful if you want to use a low voltage circuit to switch on and off a light bulb (or anything else) connected to the 220v mains supply.

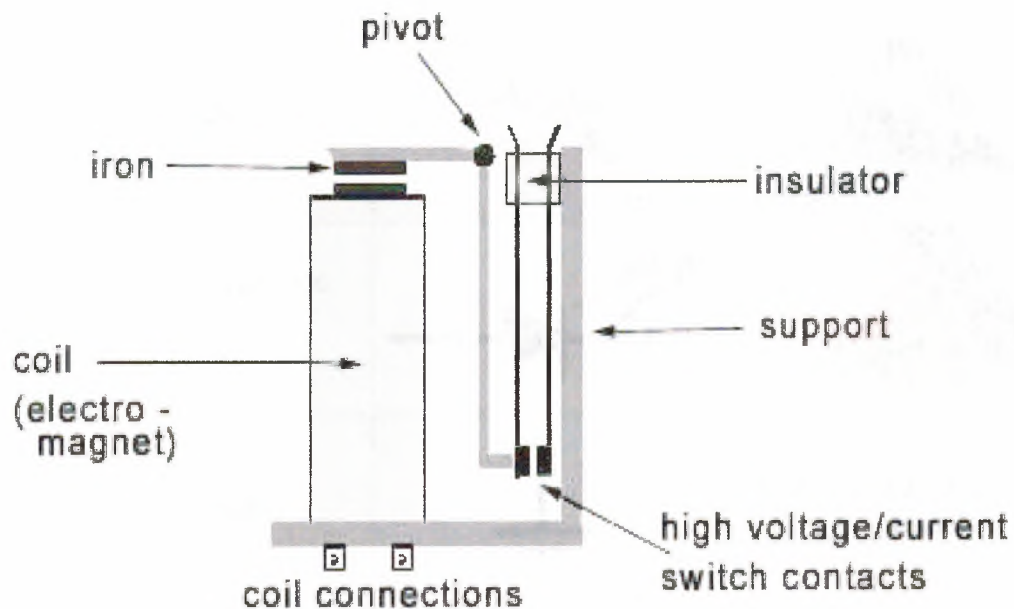













Table 4.1 Description of symbols and schematic components for our circuit[8].

Component	Schematic Symbol	Actual appearance
Resister		
Variable Resister		
Light emitting diode (LED)		
Earth Ground		This is just a connection to ground.
NPN Bipolar Transistor		
PNP Bipolar Transistor		

4.4 Summary

In this chapter we described the electrons and protons that have the electrical property of charge, where the Protons have positive charge and electrons have negative charge and they normally balance each other out.

And we show the components of our circuit and a brief description to each of them like Resistor, Transistor, and Relay. And this descriptions of the components used in them will be given so that even a complete beginner should have no difficulty in sorting out which component is which, and connecting each component into circuit correctly.

CONCLUSION

Because the Internet has become such a large part of our lives, a good understanding is needed to use this new tool most effectively. So our first chapter explains the underlying infrastructure and technologies that make the Internet work.

And every practical subject should have a hardware and software setup to give an idea about the real project. In chapter two and four we have defined some of the hardware component beginning from the main interface block diagram and ending by the used interface circuit, furthermore, in chapter two we have illustrated the main software components including the related algorithms, and flow charts.

Also as a hardware setup in chapter three is about a specific type of parallel port {LPT} the one found on just about every PC, or IBM-compatible personal computer. The D-Type 25 pin male connector, which has 5 status lines and 8 data lines. It's found commonly on the back of your PC as a D-Type 25 Pin female and we mentioned about SPP, PS2, EPP, and ECP as types of parallel port.

In this project we have achieved our aims, building this simple project and getting an experience in writing a program give us a setup for ward in designing good project, however, we have compared this project with other commercial LPT peripherals controlling devices, it might be better in some actions and cheaper to be done.

In conclusion, this project is a simple implementation of device control via internet. In future such applications will be more commonly where this type of LPT controlling devices can be use in the reality environments.

REFERENCES

- [1] <http://www.ietf.org> is the home page of the Internet Engineering Task Force. This body is greatly responsible for the development of Internet protocols and the like. 18 March 2004
- [2] <http://www.nexor.com/public/rfc/index/rfc.html> is an excellent RFC search engine useful for finding any RFC. 18 March 2004.
- [3] Software Simulation for Robot Control, Near East University graduation projects library. 5 April 2004
- [4] http://www.logicx4u.net/input_theory.htm. Functions are compatible with Jan Axel sons Inpout32.dll, so this dll can be used with the sample programs available with the book Parallel Port Complete, without any modification. 11 April 2004.
- [5] How TO Program Visual Basic6 {Deitel and Deitel Nieto}, 20 April 2004.
- [6] Parallel Port Complete: Programming, Interfacing, and Using the PC's Parallel Printer Port Jan Axelson 1996, Lakeview Research 343 pages Covers all of the port's modes and how to use them in custom applications. 20 May 2004.
- [7] <http://www.doc.ic.ac.uk/~ih/doc/par/doc/program.html>, Programming the Parallel Port: Interfacing the PC for Data Acquisition & Process Control, 25 May 2004.
- [8] <http://www.madlap.org>, this web page describe the main characteristics of electronics components, 28 May 2004.

APPENDIX A

HARDWARE SETUP

1. Circuit A

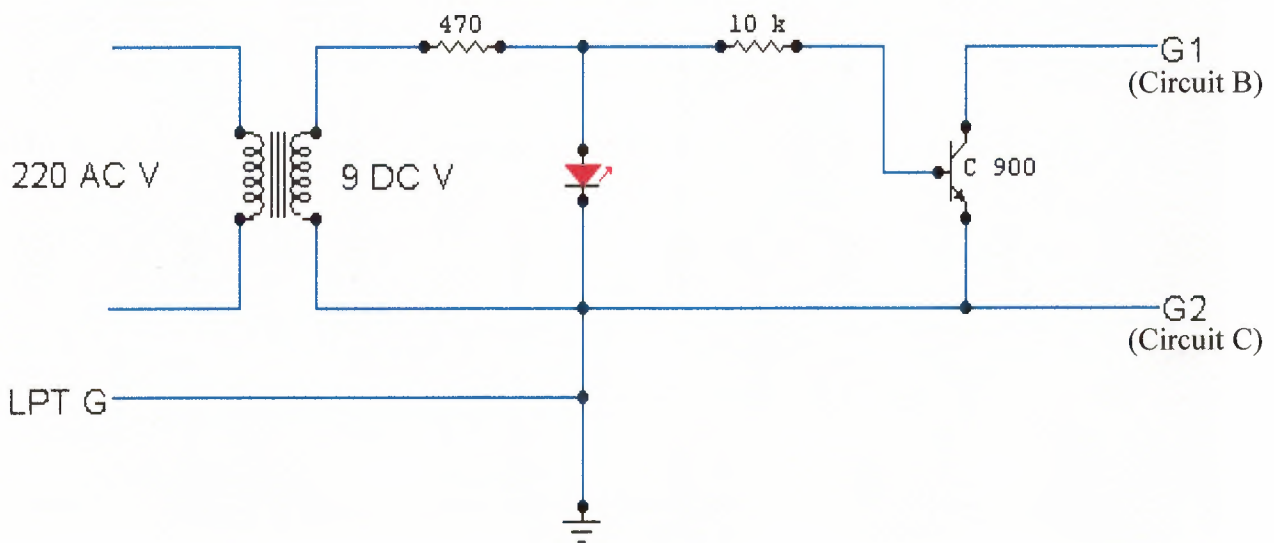


Figure 1 Circuit A

The components of the circuit are:

- Transformer form 220V to 9V.
- Resistor 470 Ω & 10 K Ω .
- Transistor c900.
- LED.

2. Circuit B

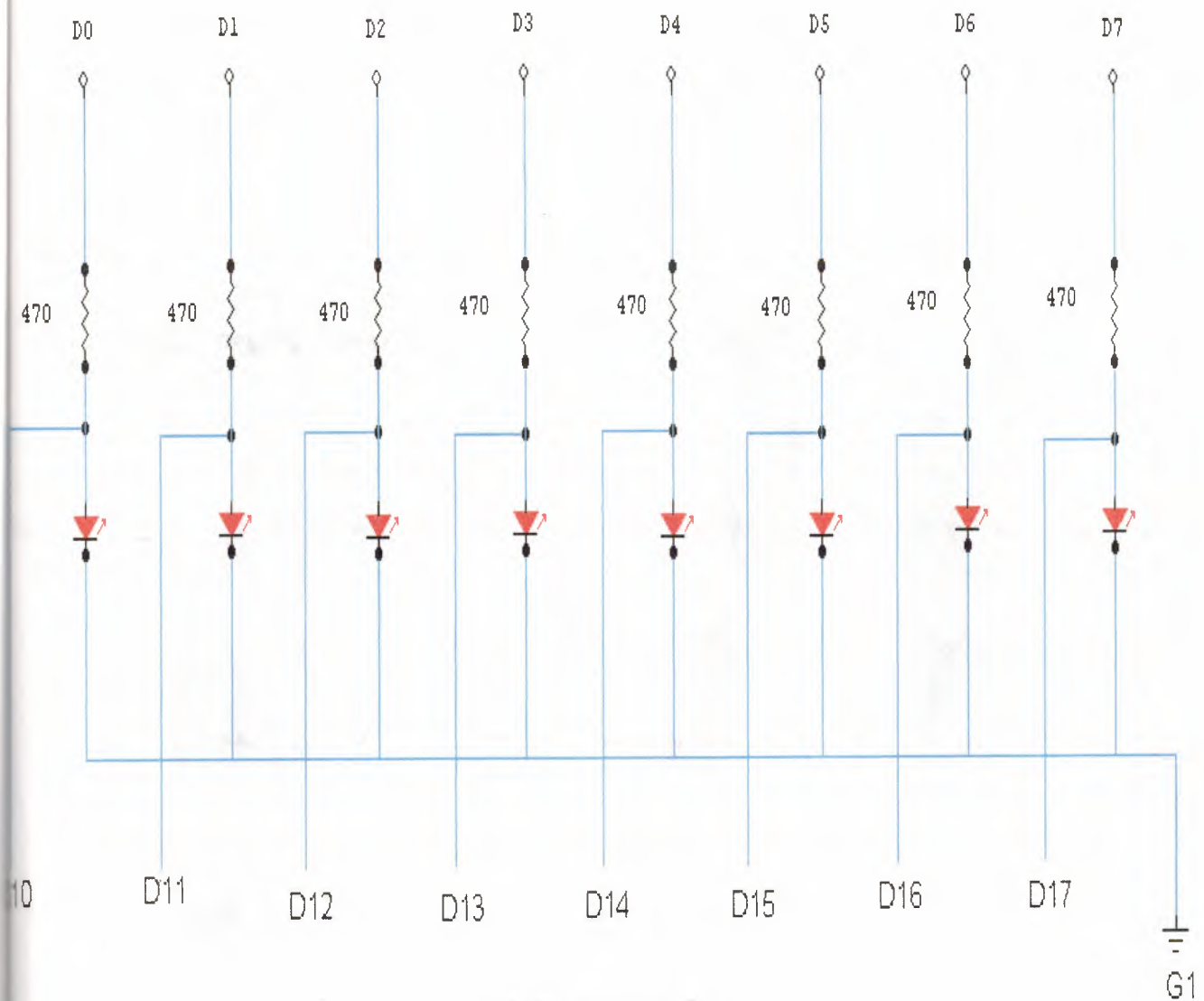


Figure 2 Circuit B

The components of the circuit are:

- Resistor 470 Ω (8)
- LED (8)

3. Circuit C

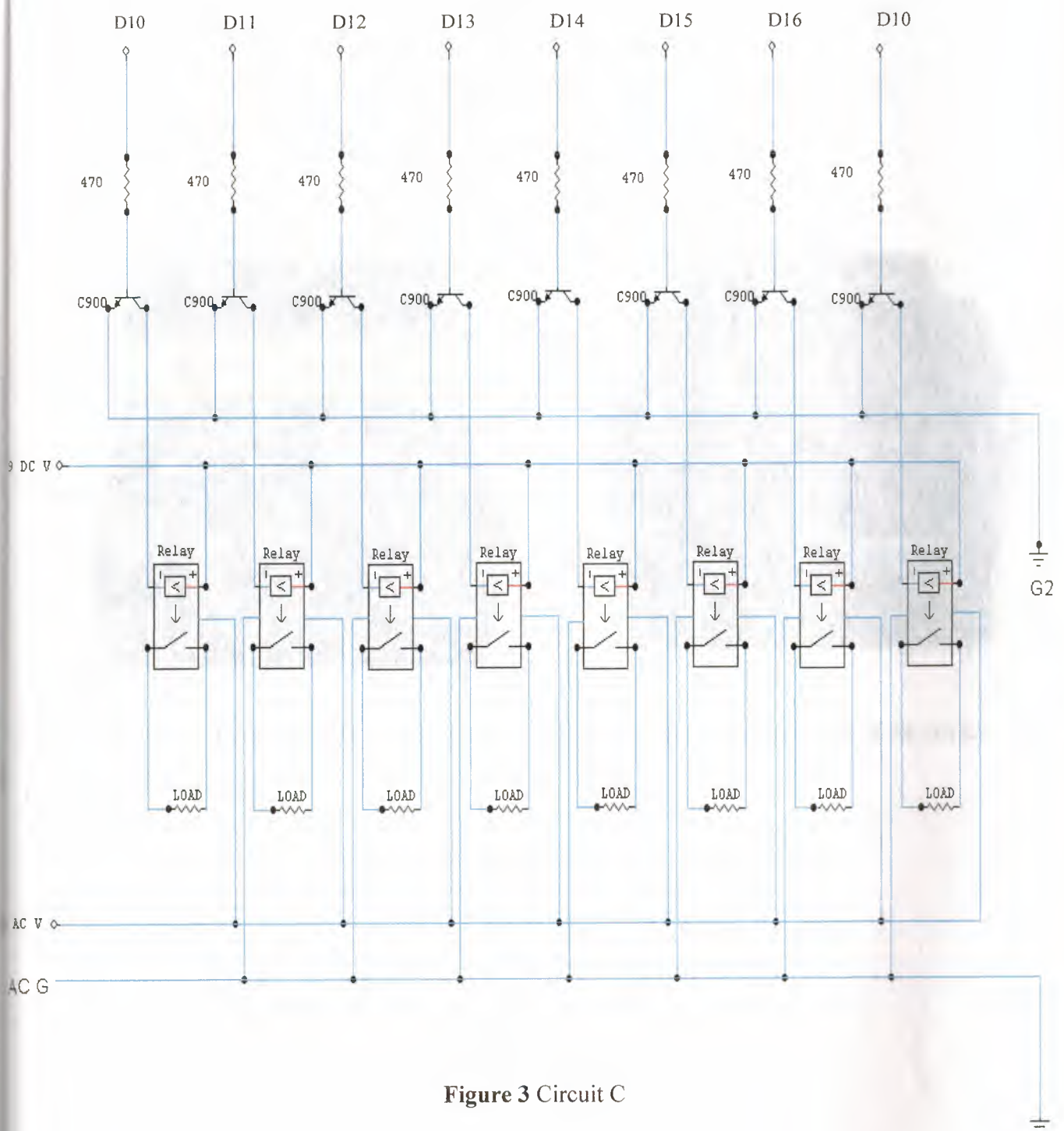


Figure 3 Circuit C

The components of the circuit are:

- Resistor 470 Ω (8).
- Transistor C900 (8).
- Relay up to 5 A, 6 to 30 V.

4. The used device

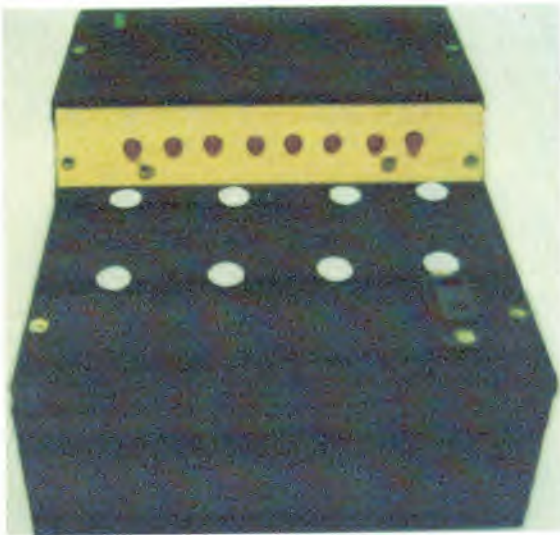


Figure 4 the front side of the device



Figure 4 the rear side of the device