

**SP03160**



# **COMPUTER CONTROLLED STEPPER MOTOR OVER A NETWORK**

**By**

**Ruchi Jha-031012**

**Nilesh Ranjan-031087**

**Manish Kumar Sachdeva-031092**



**May-2007**

**Submitted in partial fulfillment of the  
Degree of Bachelor of Technology**

**DEPARTMENT OF ELECTRONICS AND  
COMMUNICATION ENGINEERING**

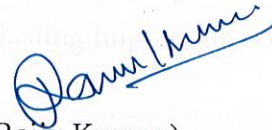
**JAYPEE UNIVERSITY OF INFORMATION  
TECHNOLOGY-WAKNAGHAT**

**May-2007**

## CERTIFICATE

This is to certify that the work entitled, "Computer Controlled Stepper Motor over a Network" submitted by Manish Kumar Sachdeva, Ruchi Jha and Nilesh Ranjan in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering in 2007 of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Dated : 30/05/07



( Rajiv Kumar )

Sr. Lecturer, ECE Department,  
Jaypee University of Information Technology,  
Waknaghat, Solan, Himachal Pradesh – 173215.



## ACKNOWLEDGEMENT

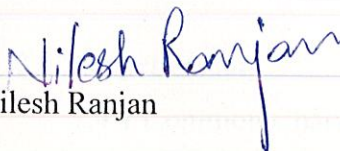
*"TELL ME AND I WILL FORGET  
TEACH ME AND I WILL REMEMBER  
INVOLVE ME AND I WILL LEARN"*

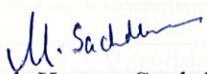
No research endeavor is a sole exercise; various individuals in their own capacity at some point or other contributed in bringing of fruition of the research endeavor, in acknowledging their guidance, support and assistance, we humbly thank them.

We would like to express our sincere thanks and gratitude to Mr. Rajiv Kumar, Department of Electronics and Communication Engineering, under whose able guidance we managed to give our endeavor the desired shape, his help, stimulating suggestions and constant encouragement helped us at every phase of the project. His enthusiasm and the view of producing quality substance has caste a deep and a long lasting impression on us.

Finally, we thank our colleagues for their constant support and encouragement. Their unobtrusive support and suggestions bolstered our confidence as usual. Their inspiring words will be a guiding force in all our endeavors to attain greater heights.

Ruchi Jha

  
Nilesch Ranjan

  
Manish Kumar Sachdeva



## TABLE OF CONTENTS

<b>TITLE</b>	<b>PAGE</b>
Certificate	i
Acknowledgement	ii
Table of contents	iii-iv
List of Figures	v
Abstract	vi
<b>1. INTRODUCTION</b>	<b>1-6</b>
1.1 Overview	1
1.2 Features of The Project	5
<b>2. HARDWARE DETAILS</b>	<b>7-35</b>
2.1 Universal Serial Bus (USB)	7
2.1.1 Human Interface Device	10
2.1.2 USB Connectors	11
2.1.3 USB Power	14
2.2 8255 Digital PIO Card	15
2.2.1 IC 8255 Programmable Peripheral Interface	19
2.2.2 Universal Serial Bus Microcontroller	22
2.3 Relay/Stepper Motor Card Interface	23
2.3.1 ULN 2803	24
2.4 Stepper Motor	27
2.4.1 Common Characteristics of Stepper Motor	31
2.4.2 Permanent Magnet Stepper Motor	33
2.4.3 Variable Reluctance Stepper Motor	35



<b>3. SOFTWARE AND NETWORK</b>	<b>36-55</b>
<b>3.1 ActiveX</b>	<b>36</b>
<b>3.1.1 ActiveX Control</b>	<b>38</b>
<b>3.1.2 USB ActiveX(.OCX)</b>	<b>39</b>
<b>3.1.3 Windows sockets</b>	<b>40</b>
 <b>3.2 The Client Program</b>	 <b>45</b>
<b>3.2.1 Coding For Client Software</b>	<b>46</b>
 <b>3.3 The Server Program</b>	 <b>49</b>
<b>3.3.1 Coding For Server Software</b>	<b>50</b>
 <b>4. WORKING AND OPERATION</b>	 <b>56-58</b>
 <b>5. CONCLUSION AND FUTURE VISION</b>	 <b>59-60</b>
 <b>Bibliography</b>	 <b>61</b>



## LIST OF FIGURES

Figure 1.	Client PC controlling hardware(motors) installed on Server PC	6
Figure 2.	The USB Icon, found on all USB Plugs	7
Figure 3.	Male and Female USB Connectors	9
Figure 4.	Type A and B USB Plugs	11
Figure 5.	USB Series "A" Plug	13
Figure 6.	USB Digital PIO Card	15
Figure 7.	Block Diagram of USB Digital PIO Card	16
Figure 8.	Port Connector(Pin-Out) for Port A, B and C	17
Figure 9.	8 Digital line ports	18
Figure 10.	Control Word in 8255	20
Figure 11.	Stepper Motor Driver Circuit using ULN 2803	23
Figure 12.	Stepper Motor Driver Circuit connected to USB Digital PIO Card	24
Figure 13.	Connection from ULN 2803 to the Stepper Motor	25
Figure 14.	Block Diagram of complete circuit	26
Figure 15.	Stepper Motor used in the Project	27
Figure 16.	Bipolar Stepper Motor	33
Figure 17.	Unipolar Stepper Motor	34
Figure 18.	Adding USB.OCX	39
Figure 19.	USB.OCX Installed	40
Figure 20.	Selecting Winsock Control	44
Figure 21.	Winsock Installed	44
Figure 22.	VB GUI Client Motor	45
Figure 23.	VB GUI Server Motor	49
Figure 24.	Client Computer	57
Figure 25.	Server Computer	57
Figure 26.	Client Software	58
Figure 27.	Server Software	58



## **ABSTRACT**

The project "Computer Controlled Stepper Motor over a Network" is a part of our B.Tech curriculum at Jaypee University of Information Technology, Solan. Our aim by the means of this project is to implement a working PC interfaced peripheral (Stepper Motor) with VB GUI. The GUI should be enhanced with at least one ActiveX component. Additionally the Stepper Motor should be controlled over the Internet through a TCP/IP (Transmission Control Protocol/Internet Protocol) client/server program. This effort of ours has taken us into a position where we can think of numerous applications of the technology around us.

We hope this effort of ours will encourage people to find simple applications of the complex technology around us that affects our daily lives. This document is an aid to the future engineers to understand the application, features and future derivatives of this project.



## CHAPTER 1

### INTRODUCTION

---

#### 1.1 Overview

Computer controlled systems now a days find application almost everywhere. The range of their applications range from most ordinary ones used for daily purposes to the most sophisticated ones. The ever-increasing availability of reliable, low-cost, high-performance computing hardware has been one of the key drivers behind the increase of computer-based control of processes. Such controllers can perform a variety of complicated tasks which

- 1 Increase the overall process performance and repeatability
- 2 Decrease the operating cost
- 3 Meet stringent safety and environmental constraints
- 4 Vastly improve the human user interface.

Because of the technical innovations taking place in every field so rapidly, we are able to make our lives more and more comfortable. These innovations have also affected the field of computer controlled systems significantly. Typical control systems like modern washers use computer control to provide a cleaner laundry, using less energy and less water. Computers enable car engines to deliver more horsepower, with lower fuel consumption and emissions, and allow sophisticated diagnostic tests. Industrial controllers enable more efficient manufacturing, while state-of-the-art control in aerospace applications is responsible for the extraordinary performance of fighter aircraft, and the cost efficiency of commercial airliners.

In addition to the advances in hardware, the software evolution has played a key role in enabling these applications. Controller evaluation, Commissioning, and Maintenance have taken new meaning with the advancement of modern hardware. As the hardware is growing at a rapid speed, the software interface to that particular hardware is also evolving. The web gives the ability to view and control the process on a computer, monitoring real-time information via the Internet.

New emerging technologies have also reduced the space occupied before improving themselves in terms of output and performance. Taking an example of a modern process controller which can deliver the computing power of an old workstation and occupy one fourth of the volume of a notebook PC, we can easily illustrate this point. These controllers can now be "embedded" in a process tool and they perform a variety of tasks to ensure process reliability and accurate diagnostics. By doing so locally, independent of a central monitoring computer, they free up resources and improve the fault-tolerance of the overall system. Although, they require more elaborate communications and networking still its better to use them.

The control of equipments with the help of Internet is a big improvement over existing control systems. Internet applications for the PC are numerous but could be costumed for equipment control, monitoring system, calibration etc. The Web-enabled features will be soon request for lot of application and for that also no special acquisition software will be required to be installed on the computer, What would be needed only would be just a web browser and a TCP/IP client/server program.

The Internet allows one PC (called a client) to control hardware (like motors and relays) installed on another PC (called a server). In other words, we can remotely control or monitor devices. Since the Internet is just a medium for computers to "talk" to each other, it enables us to perform applications like automating your home (e.g. turn on/off air-conditioning) and data acquisition (e.g. measure temperatures).



One significant recent development that has taken place in computer controlled systems is CNC. CNC stands for computer numerical control. This refers specifically to a computer "controller" that reads instructions and drives the machine tool. The introduction of CNC machines radically changed the manufacturing industry.

With the increased automation of manufacturing processes with CNC machining, considerable improvements in consistency and quality have been achieved. CNC automation reduced the frequency of errors and provided CNC operators with time to perform additional tasks. CNC automation also allows for more flexibility in the way parts are held in the manufacturing process and the time required to change the machine to produce different components.

When we look at the big picture, the building blocks for controlling and monitoring hardware over the Internet can be used for home or industrial automation, robotics and data acquisition. Robot vision is an emerging technology with applications in such areas as robot navigation, control of robot formations, and exploration of hazardous environments.

Remote experiments, using tele operated hardware via the internet, are exchanged within a worldwide network. A few examples are control engineering tasks such as process modeling, parameter identification, and controller design. These technologies offer also interesting technology transfer potential for industrial tele maintenance.

The use of tele-operation, or remote operation of a robot by a human operator, allows the operator to control the robot from a remote location and guide the robots actions. The remote control of robots or machines is something quite usual in the industry. Different techniques and tools exist to communicate over the internet. The tele manipulation of a manufacturing system corresponds to the control by a distant user of a system composed of different tools. The objective is to authorize the user to manipulate this distant system as if it were close to him.

The increased capabilities of telematics technologies i.e a combination of telecommunication and informatics enable the provision of a continuously growing spectrum of services at remote locations. Interesting applications are reported in particular in the areas of industrial automation, telerobotics for hazardous environments, spacecraft telemetry and tele command, traffic control, smart homes, tele education and tele medicine.

Challenging problems in tele operations have been first solved in the context of space exploration. The related technologies were transferred to remotely controlled robots in hazardous environments. While in the commercial field, telemaintenance of computers and software is already well introduced, only few demonstration applications for teleservicing of machines in industrial production are available.

The main technical research areas, summarised in the discipline telematics, concern remote sensor data acquisition, preprocessing and transfer, as well as remote control activities to be initiated by the human teleoperator. In particular the telecommunication basis For teleoperations via internet, the server is located on the robot and has to provide a continuous stream of sensor data to the client located at the operator's place. User friendly interfaces for teleoperations play a key role to allow teleservicing of complex processes by remote human operators. Modules providing functionalities, like access to different sensor types, sensor data preprocessing methods, data recording for post processing, autonomy features, predictions by simulations, can be integrated into the generic user interface according to the specific application needs. Thus teleservicing of remote equipment *can* be made easier and more comfortable.

The most advanced pieces of scientific equipment are often both very rare and very expensive. This is a poor combination if your research needs dictate the use of such a device. Advances in modern telecommunication and data processing technologies enable innovative teleservicing applications with an interesting economic potential. For example, scanning electron microscopes, depending on quality, cost millions of dollars. This high price prohibits their purchase for occasional research use.



This high price also decreases the probability that there is a facility near by that has both, a scanning electron microscope, and has time to let you use it. One solution to this scenario is to create a product that allows scanning electron microscopes and possibly other equipment to be used remotely. Such a device would utilize the internet to pass commands to the microscope, as well as return data from the microscope back to the user.

The purpose of this project is to allow remote operation of a stepper motor that is controlled through the operator's interface. Stepper Motor are one of the most widely used prime movers in industry today. A major reason for the use of Stepper motor in electromechanical control systems is the ease with which speed can be controlled. The polarity of the applied voltage determines the direction of rotation. In addition to control of the motor, the operator will be able to attach any kind of equipments like robotic arm, printers etc. In addition to tele-operation by a human operator, the inclusion of the stepper motor allows this system to be used in applications that require knowledge of the precise location of the camera.

## **1.2 Features of The Project**

The Project goal is to implement a working PC interfaced peripheral ( In our case it is a Stepper Motor ) with VB GUI.. The GUI should be enhanced with at least one ActiveX component. Additionally the Stepper Motor should be controlled over the Internet through a TCP/IP (Transmission Control Protocol/Internet Protocol) client/server program. Visual Basic (VB) is a powerful, high-level Windows-based programming language with a quick learning curve. High-level means that the programmer can avoid tedious coding, one example being, Internet-related functions. A *client* PC requests a *server* PC to perform actions.

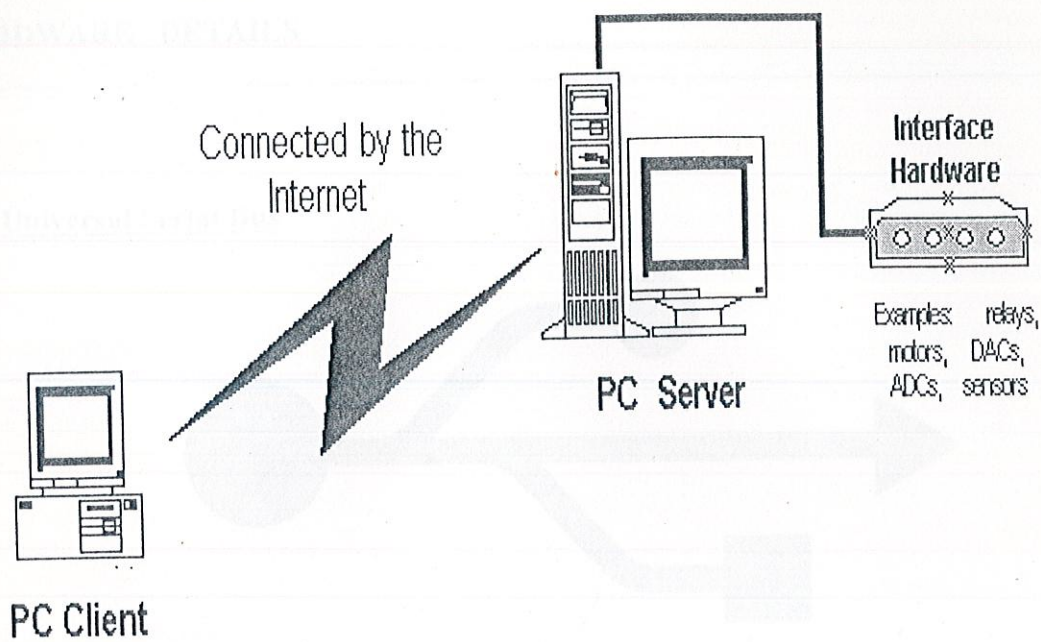


Figure 1. Client PC controlling hardware (like motors and relays) installed on Server PC

Two softwares are required for the above purpose

- 1 A server software - to interface with the mechanical device and the internet
- 2 Client Software - to interface with the user and the internet

A simple chat program can be written in VB to exchange text messages between a client and server. Two programs need to be written (one running on the client PC, the other on the server PC) resulting in a teletype-like application; two people can type messages to each other over the Internet. This helps in developing programs to control hardware devices over the Internet. The description of the client and the server program is mentioned in the coming section.



#### 2.1 Universal Serial Bus

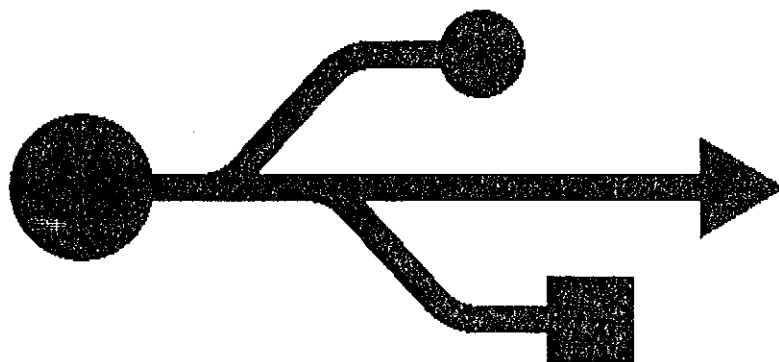


Figure 2. The USB Icon, found on all USB Plugs

**Universal Serial Bus (USB)** is a serial bus standard to interface devices. A major component in the legacy-free PC, USB was designed to allow peripherals to be connected using a single standardised interface socket, to improve plug-and-play capabilities by allowing devices to be connected and disconnected without rebooting the computer (hot swapping). Other convenient features include powering low-consumption devices without the need for an external power supply and allowing some devices to be used without requiring individual device drivers to be installed.

USB is intended to help retire all legacy serial and parallel ports. USB can connect computer peripherals such as mouse devices, keyboards, PDAs, gamepads and joysticks, scanners, digital cameras and printers. For many devices such as scanners and digital cameras, USB has become the standard connection method. USB is also used extensively to connect non-networked printers; USB simplifies connecting several printers to one computer. USB was originally designed for personal computers, but it has become commonplace on other devices such as PDAs and video game consoles. In 2004, there were about 1 billion USB devices in the world.

The design of USB is standardized by the USB Implementers Forum (USB-IF), an industry standards body incorporating leading companies from the computer and electronics industries. Notable members have included Apple Computer, Hewlett-Packard, NEC, Microsoft, Intel, and Agere.

A USB system has an asymmetric design, consisting of a host controller and multiple daisy-chained peripheral devices. Additional USB hubs may be included in the chain, allowing branching into a tree structure, subject to a limit of 5 levels of branching per controller. No more than 127 devices, including the bus devices, may be connected to a single host controller. Modern computers often have several host controllers, allowing a very large number of USB devices to be connected. USB cables do not need to be terminated.

In USB terminology, individual devices are referred to as *functions*, because each individual physical device may actually host several functions, such as a webcam with a built-in microphone. Functions are linked in series through *hubs*. The hubs are special-purpose devices that are not considered functions. There always exists one hub known as the root hub, which is attached directly to the host controller.

Functions and hubs have associated *pipes* (logical channels). Pipes are connections from the host controller to a logical entity on the device named an *endpoint*. The term *endpoint* is also occasionally used to refer to the entire pipe. A function can have up to 32 active pipes, 16 into the host controller and 16 out of the controller. Each endpoint can transfer data in one direction only, either into or out of the device/function, so each pipe is uni-directional.

When a device is first connected, the host enumerates and recognizes it, and loads the device driver it needs. When a function or hub is attached to the host controller through any hub on the bus, it is given a unique 7 bit address on the bus by the host controller. The host controller then polls the bus for traffic, usually in a round-robin fashion, so no function can transfer any data on the bus without explicit request from the host controller.

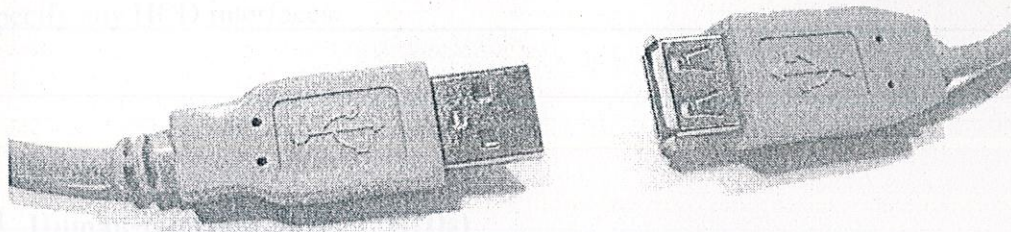


Figure 3. Male and Female USB Connectors

The computer hardware that contains the host controller and the root hub has an interface geared toward the programmer which is called *Host Controller Device* (HCD) and is defined by the hardware implementer.

In the version 1.x age, there were two competing HCD implementations, *Open Host Controller Interface* (OHCI) and *Universal Host Controller Interface* (UHCI). OHCI was developed by Compaq, Microsoft and National Semiconductor; UHCI was by Intel.

VIA Technologies licensed the UHCI standard from Intel; all other chipset implementers use OHCI. UHCI is more software-driven, making UHCI slightly more processor-intensive than OHCI but cheaper to implement. The dueling implementations forced operating system vendors and hardware vendors to develop and test on both implementations which increased cost.

During the design phase of USB 2.0 the USB-IF insisted on only one implementation. The USB 2.0 HCD implementation is called the *Enhanced Host Controller Interface* (EHCI). Only EHCI can support hi-speed transfers. Most of PCI-based EHCI controllers contain other HCD implementations called 'companion host controller' to support Full



Speed and Low Speed devices. The virtual HCD on Intel and VIA EHCI controllers are UHCI. All other vendors use virtual OHCI controllers.

HCD standards are out of the USB specification's scope, and the USB specification does not specify any HCD interfaces.

### **2.1.1 Human-interface devices (HIDs)**

Mice and keyboards are frequently fitted with USB connectors, but because most PC motherboards still retain PS/2 connectors for the keyboard and mouse as of 2006, are generally supplied with a small USB-to-PS/2 adaptor so that they can be used with either USB or PS/2 ports. There is no logic inside these adaptors: they make use of the fact that such HID interfaces are equipped with controllers that are capable of serving both the USB and the PS/2 protocol, and automatically detect which type of port they are plugged in to. Joysticks, keypads, tablets and other human-interface devices are also progressively migrating from MIDI, PC game port, and PS/2 connectors to USB.

Apple Macintosh computers have used USB exclusively for all wired mice and keyboards since January 1999.

### **2.1.2 USB connectors**

The connectors which the USB committee specified were designed to support a number of USB's underlying goals, and to reflect lessons learned from the varied menagerie of connectors then in service.

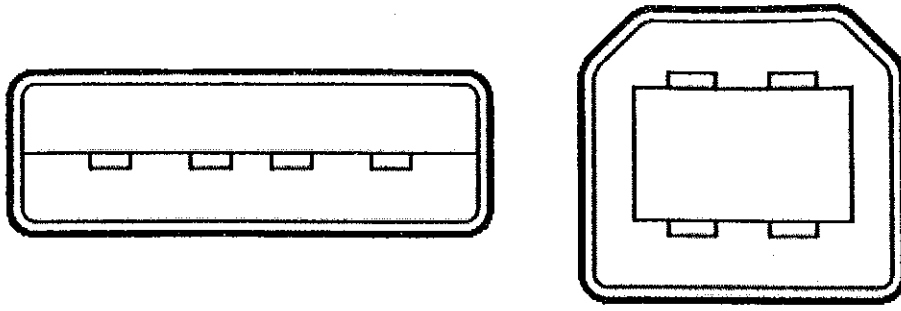


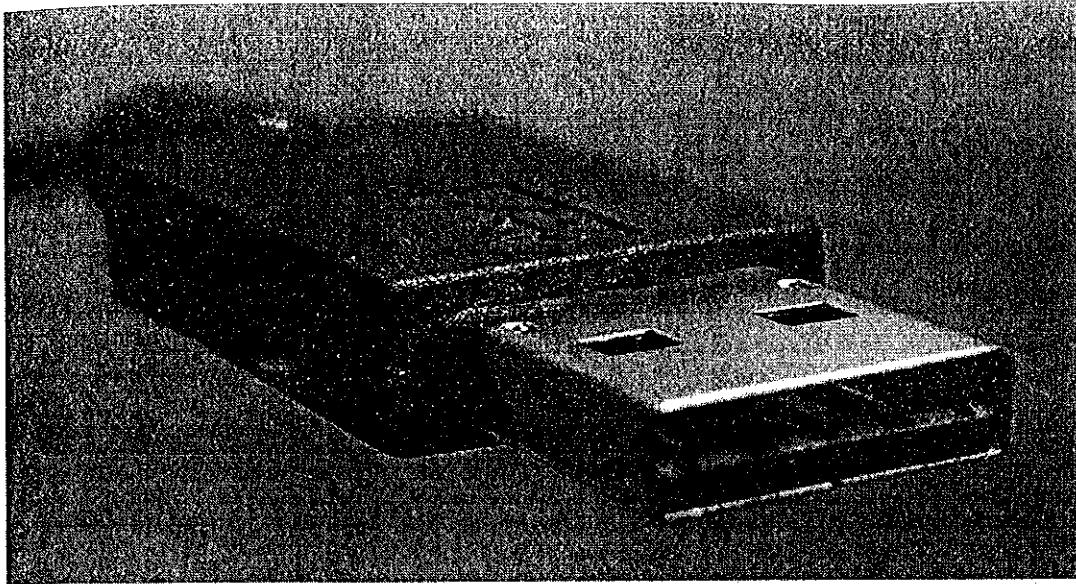
Figure 4. Type A and B USB Plugs

In particular:

- The connectors are designed to be robust. Many previous connector designs were fragile, with pins or other delicate components prone to bending or breaking, even with the application of only very modest force. The electrical contacts in a USB connector are protected by an adjacent plastic tongue, and the entire connecting assembly is further protected by an enclosing metal sheath. As a result USB connectors can safely be handled, inserted, and removed, even by a small child. The encasing sheath and the tough moulded plug body mean that a connector can be dropped, stepped upon, even crushed or struck, all without damage; a considerable degree of force is needed to significantly damage a USB connector.
- It is difficult to incorrectly attach a USB connector. Connectors cannot be plugged-in upside down, and it is clear from the appearance and kinesthetic sensation of making a connection when the plug and socket are correctly mated. However, it is not obvious at a glance to the inexperienced user (or to a user without sight of the installation) which way round a connector goes, so it is often necessary to try both ways.
- The connectors are particularly cheap to manufacture.
- The connectors enforce the directed topology of a USB network. USB does not support cyclical networks, so the connectors from incompatible.

- USB devices are themselves incompatible. Unlike other communications systems (e.g. RJ-45 cabling) gender-changers are almost never used, making it difficult to create a cyclic USB network.
- A moderate insertion/removal force is specified. USB cables and small USB devices are held in place by the gripping force from the receptacle (without the need for the screws, clips, or thumbturns other connectors require). The force needed to make or break a connection is modest, allowing connections to be made in awkward circumstances or by those with motor disabilities.
- The connector construction always ensures that the external sheath on the plug contacts with its counterpart in the receptacle before the four connectors within are connected. This sheath is typically connected to the system ground, allowing otherwise damaging static charges to be safely discharged by this route (rather than via delicate electronic components). This means of enclosure also means that there is a (moderate) degree of protection from electromagnetic interference afforded to the USB signal while it travels through the mated connector pair (this is the only location when the otherwise twisted data pair must travel a distance in parallel). In addition, the power and common connections are made after the system ground but before the data connections. This type of staged make-break timing allows for safe hot-swapping and has long been common practice in the design of connectors in the aerospace industry.
- The USB standard specifies relatively low tolerances for compliant USB connectors, intending to minimize incompatibilities in connectors produced by different vendors (a goal that has been very successfully achieved). Unlike most other connector standards, the USB spec also defines limits to the size of a connecting device in the area around its plug.

This was done to avoid circumstances where a device complies with the connector specification but its large size blocks adjacent ports. Compliant devices must either fit within the size restrictions or support a compliant extension cable which does.



**Figure 5. USB Series "A" Plug**

Cables have only plugs, and hosts and devices have only receptacles. Hosts have type-A receptacles; devices have type-B. Type-A plugs only mate with type-A receptacles, and type-B with type-B. There are several types of USB connectors, and some have been added as the specification has progressed. An extension to USB called USB On-The-Go allows a single port to act as either a host or a device - chosen by which end of the cable plugs into the socket on the unit. Even after the cable is hooked up and the units are talking, the two units may "swap" ends under program control.

This facility targets units such as PDAs where the USB link might connect to a PC's host port as a device in one instance, yet connect as a host itself to a keyboard and mouse device in another instance.



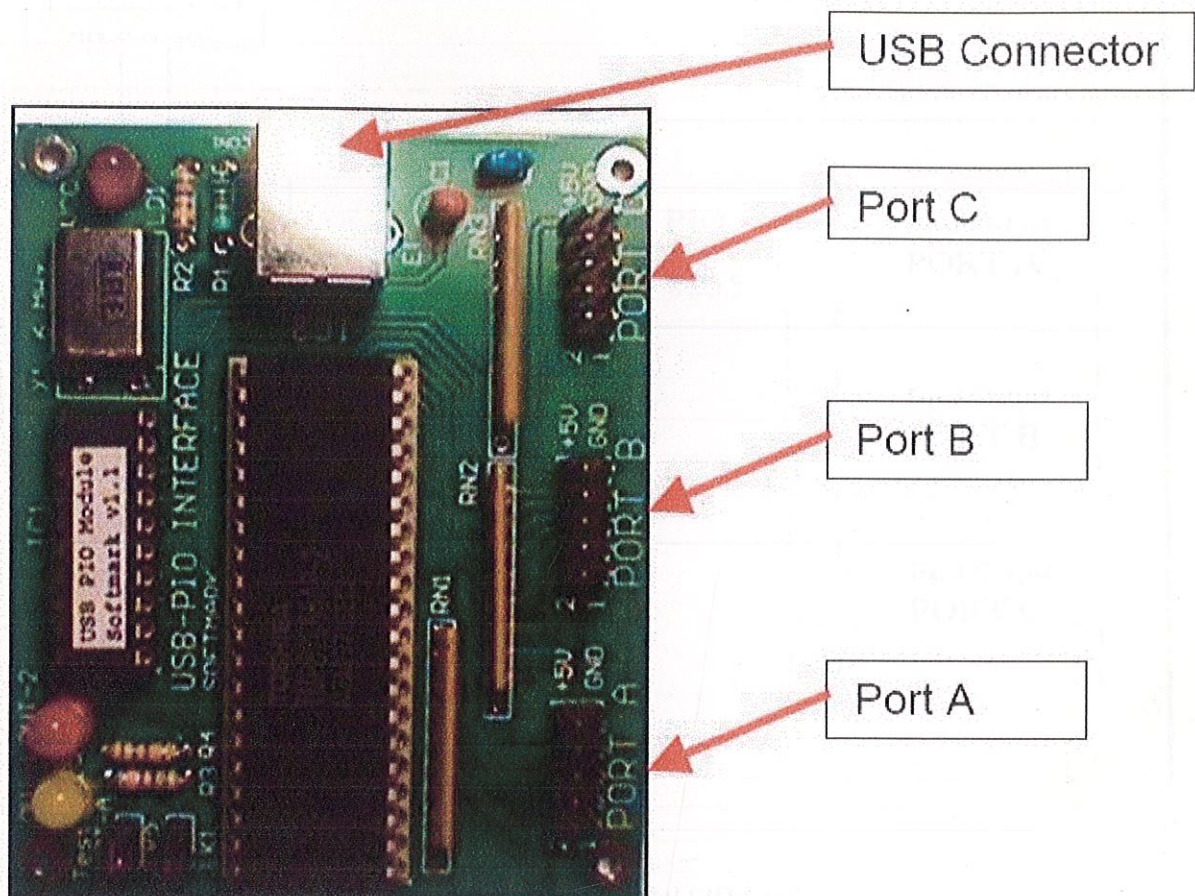
### 2.1.3 USB Power

The USB specification provides a 5 V (volts) supply on a single wire from which connected USB devices may draw power. The specification provides for no more than 5.25 V and no less than 4.35 V between the +ve and -ve bus power lines. Initially, a device is only allowed to draw 100 mA. It may request more current from the upstream device in units of 100 mA up to a maximum of 500 mA.

If a bus-powered hub is used, the devices downstream may only use a total of four units — 400 mA — of current. This limits compliant bus-powered hubs to 4 ports, among other things. The host operating system typically keeps track of the power requirements of the USB network and may warn the computer's operator when a given segment requires more power than is available.

Some USB devices draw more power than is permitted by the specification for a single port. This is a common requirement of external hard and optical disc drives and other devices with motors or lamps. Such devices can be used with an external power supply of adequate rating; some external hubs may, in practice, supply sufficient power. For portable devices where external power is not available, but not more than 1 A is required at 5 V, devices may have connectors to allow the use of two USB cables, doubling available power but reducing the number of USB ports available to other devices.

## 2.2 8255 Digital PIO Card



**Figure 6. USB Digital PIO Card**

This 8255 PIO card can be used to control 24 digital lines. The well-known IC 8255 is used as the PIO Controller. This Card is multipurpose and can be used for a variety of purposes. For instance, it can help you speed up your software development time, and it can assist you in controlling your own projects by using the latest USB technology.

The USB PIO Card is connected to the personal computer via cable (type A to B). Ports A, B and C are bi-directional i.e they can be used for input as well as output. They can be set as inputs or outputs via control lines from the USB Controller.

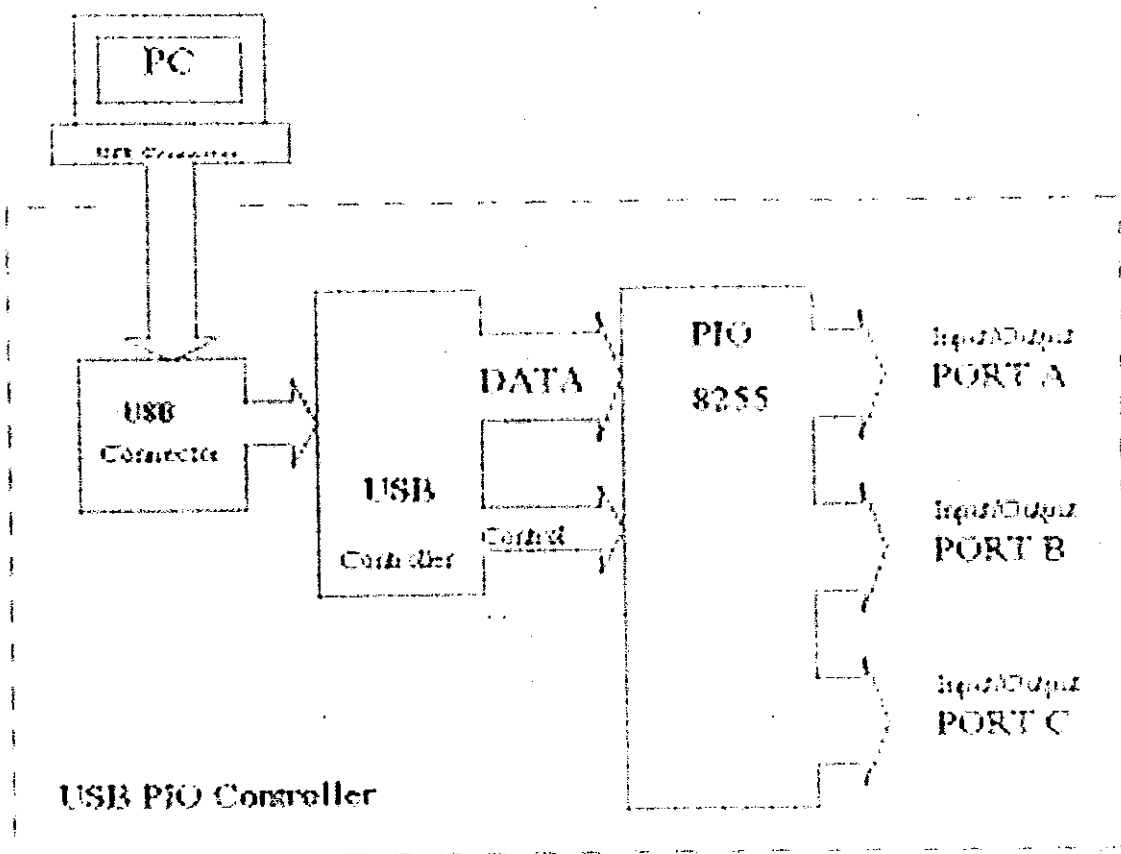


Figure 7. Block Diagram of USB Digital PIO Card

The project features three ports: A, B and C. These ports can be set as either inputs or outputs. Port C can be set both as an input and output. The USB\_OCX component will be used to write to and read from the ports. With this component you can fully control the USB PIO Card and all modules from your application.

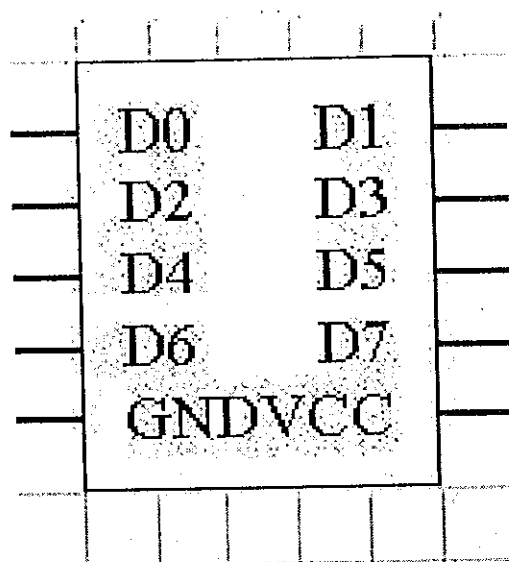
**The USB PIO Card may also be used to control modules such as:**

- ADC converters
- DAC converters
- PLL generators

- Relay Cards
- Stepper Motor Cards
- Switch Cards
- LED Indicator Cards

**Features of the 8255 PIO card:**

- 24 digital input/output lines
- 3 bi-directional 8-bit ports programmable by software
- digital output capable of driving relays
- 2,500V isolation between the computer and the controlled hardware
- USB\_OCX tool to control the USB PIO Card from your own application.



**Figure 8. Port Connector(Pin-Out) for Port A, B and C**

The output from a particular port can be used to drive, for example, a relay or a stepper motor. This interface can be used to drive devices which use no more than 50V and their output current is maximum 1A from one line.



PIO Controller

8255

D0  
D1  
D2  
D3  
D4  
D5  
D6  
D7

PORT A

D0  
D1  
D2  
D3  
D4  
D5  
D6  
D7

PORT B

D0  
D1  
D2  
D3  
D4  
D5  
D6  
D7

PORT C

Figure 9. 8 Digital line ports. These lines can be easily set (for High or Low level)

All ports are 8-bit TTL lines. The USB PIO Card is powered by the USB bus line. The maximum current from this line is 100mA. The current taken by the Card is about 10mA.

As driving relays or stepper motors will require the use of current drivers, the USB bus doesn't give you enough current. For that reason, external power supply should be there.

### 2.2.1 IC 8255 Programmable Peripheral Interface

IC 8255 is the most important component of 8255 PIO card. The 8255A is a programmable peripheral interface (PPI) device. Its function is that of a general purposes I/O component to Interface peripheral equipment to the microcomputer system bus. The functional configuration of the 8255A is programmed by the systems software so that normally no external logic is necessary to interface peripheral devices or structures.

The 8255A contains three 8-bit ports (A, B and C) as shown in the figure. All of the ports can be configured in a wide variety of functional characteristics by the system software but each has its own special features or personally to further enhance the power and flexibility of the 8255A.

**Port A.** One 8 bit data output latch/buffer and one 8-bit data input latch.

**Port B.** One 8-bit data output latch/buffer and one 8-bit data input buffer.

**Port C.** One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the controls signal outputs and status signal inputs in conjunction with ports A and B.

The 8255 allows for three distinct operating modes (Modes 0, 1 and 2) as follows:

- Mode 0: Ports A and B operate as either inputs or outputs and Port C is divided into two 4-bit groups either of which can be operated as inputs or outputs
- Mode 1: Same as Mode 0 but Port C is used for handshaking and control
- Mode 2: Port A is bidirectional (both input and output) and Port C is used for handshaking. Port B is not used.

Each port is 8-bit TTL-compatible. The various modes can be set by writing a special value to the control port. The control port is Base Address + 3.

Peripheral device in a microcomputer system usually has a "service routine" associated with it. This routine manages the software interface between the device and the CPU. By examining the I/O devices interface characteristics for data transfer and timing, a control word can be developed to initialize the 8255 to exactly "fit" the application.

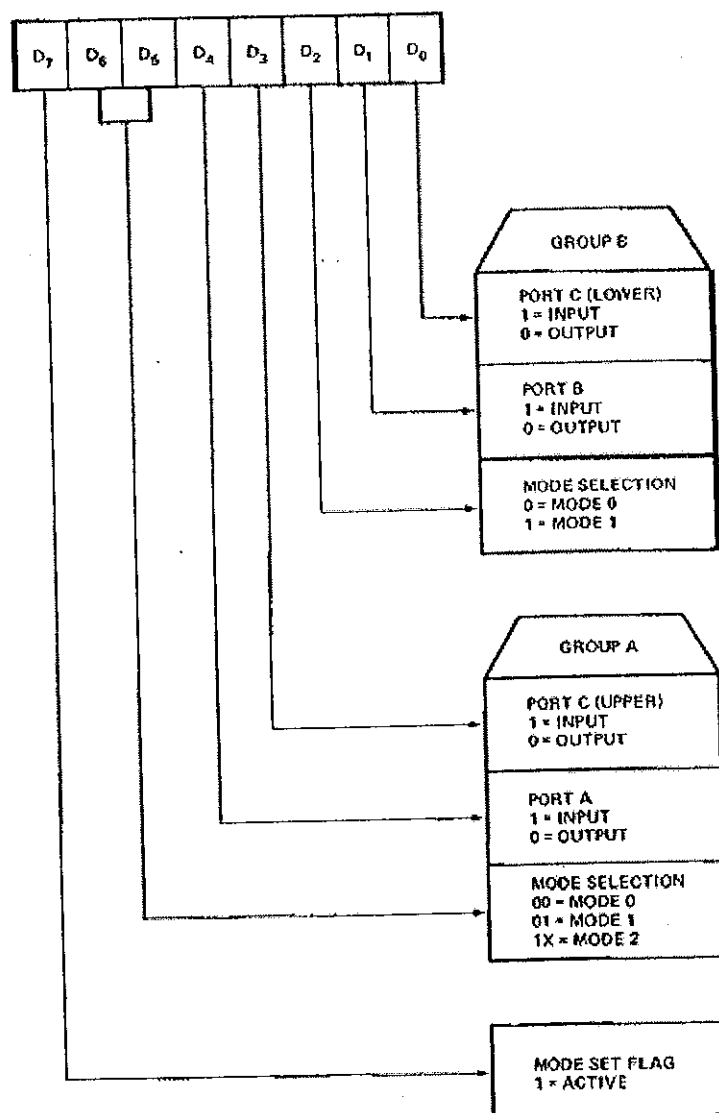
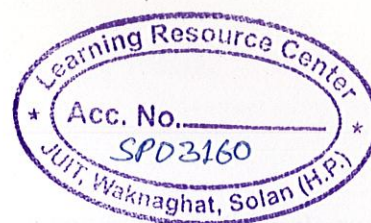


Figure 10. Control Word in 8255



### Mode 0 Basic Functional Definitions:



- \*Two 8-bit ports and two 4-bit port
- \*Any port can be input or output.
- \*Outputs are not latched.
- \*Inputs are not latched.
- \*16 different Input/output configurations are not possible in this Mode.

### Mode 1 Basic Functional Definitions:

- \*Two groups (Group A and Group B)
- \*Each group contains one 8-bit data port and one 4-bit control/data port
- \*The 8-bit data port can be either Inputs or output Both inputs and outputs are latched.
- \*The 4-bit port is used for control and status of the 8-bit data port.

### MODE 2 Basic Functional Definitions:

- \*Used in Group A only.
- \*One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Port (Port C).
- \*Both Inputs and Outputs are latched.
- \*The 5-bit control port (Port C) is used for control and status for the 8-bit,bi-directional bus port (Port A).

The 8255 has three 8-bit TTL-compatible I/O ports. Thus technically you could control up to 24 individual devices (for example: a home security device, with a digital input detecting if someone rang your doorbell, and having a digital output turning on a light switch, the other remaining DIOs could then be used to detect for sound detection, doors being opened, digitally dial 911, detect smoke etc.).



### 2.2.2 Universal Serial Bus Microcontroller

USB microcontrollers are optimized for human-interface computer peripherals such as a mouse, joystick, and gamepad. These USB microcontrollers conform to the low-speed (1.5 Mbps) requirements of the USB Specification version 1.1. Each microcontroller is a self-contained unit with: a USB interface engine, USB transceivers, an 8-bit RISC microcontroller, a clock oscillator, timers, and program memory. Each microcontroller supports one USB device address and two endpoints. It offer 4 Kbytes of EPROM. The program memory space is divided into two functional groups: interrupt vectors and program code.

The interrupt vectors occupy the first 16 bytes of the program space. Each vector is 2 bytes long. After a reset, the Program Counter points to location zero of the program space. The USB Controller includes 128 bytes of data RAM. The upper 16 bytes of the data memory are used as USB FIFOs for Endpoint 0 and Endpoint 1. Each endpoint is associated with an 8-byte FIFO.

The USB controller includes two pointers into data RAM, the Program Stack Pointer (PSP) and the Data Stack Pointer (DSP). It Conforms to USB 1.5 Mbps Specification, Version 1.1 and supports 1 device address and 2 endpoints (1 control endpoint and 1 data endpoint) Operating voltage from 4.0V to 5.25 VDC. Operating temperature from 0 to 70 degree Celsius



### 2.3 Relay/Stepper Motor Card Interface

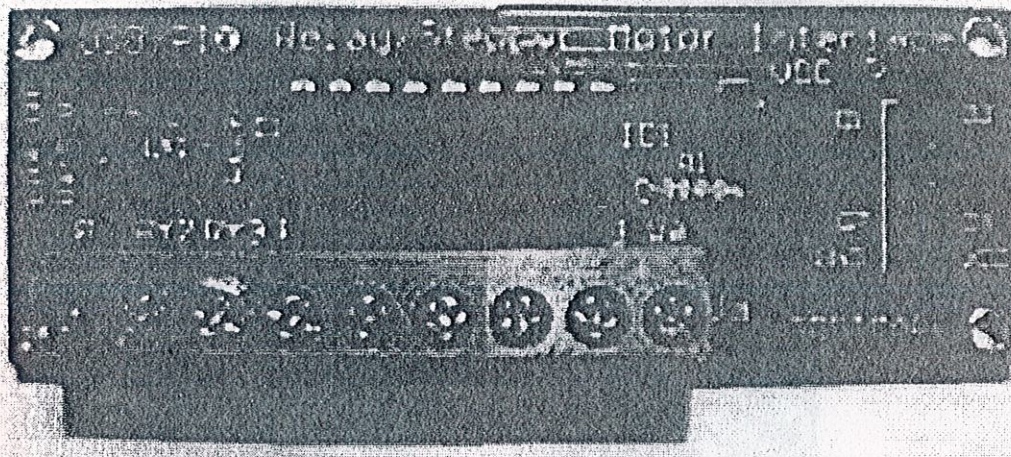
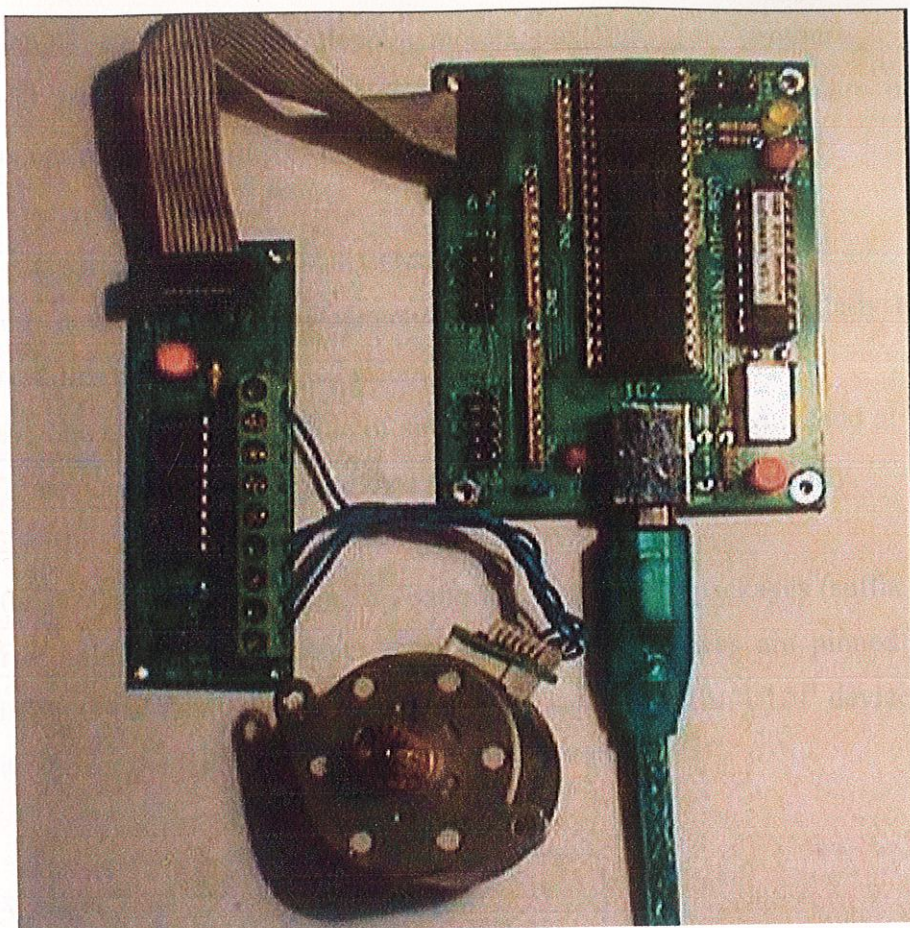


Figure 11. Stepper Motor Driver Circuit using ULN 2803

A stepper motor can be controlled via the USB PIO Card. You can control two stepper motors from one port. Only four lines from a port have to be used to drive one stepper motor. The maximum voltage supplied to a card can be 50V and maximum current per coil is 1A.

The output from a particular port will drive the current driver based on the ULN 2803. Open collector outputs from the driver can be used to directly drive relays or stepper motor coils.





**Figure 12. Stepper Motor Driver Circuit connected to USB Digital PIO Card**

### **2.3.1 ULN 2803**

Featuring continuous load current ratings to 500 mA for each of the drivers, ULN 2803 highvoltage, high-current Darlington arrays are ideally suited for interfacing between low-level logic circuitry and multiple peripheral power loads. Typical power loads totaling over 260 W ( $350 \text{ mA} \times 8, 95 \text{ V}$ ) can be controlled at an appropriate duty cycle depending on ambient temperature and number of drivers turned on simultaneously.



Typical loads include relays, solenoids, stepping motors, magnetic print hammers, multiplexed LED and incandescent displays, and heaters. All devices feature open-collector outputs with integral clamp diodes.

The ULN 2803 has series input resistors selected for operation directly with 5 V TTL or CMOS. These devices will handle numerous interface needs — particularly those beyond the capabilities of standard logic buffers. The ULN 2803 is a standard Darlington array. The outputs are capable of sinking 500 mA and will withstand at least 50 V in the off state. Outputs may be paralleled for higher load current capability.

These Darlington arrays are furnished in 18-pin dual in-line plastic packages (suffix 'A') or 18-lead small-outline plastic packages (suffix 'LW'). All devices are pinned with outputs opposite inputs to facilitate ease of circuit board layout. Prefix 'ULN' devices are rated for operation over the temperature range of  $-20^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ .

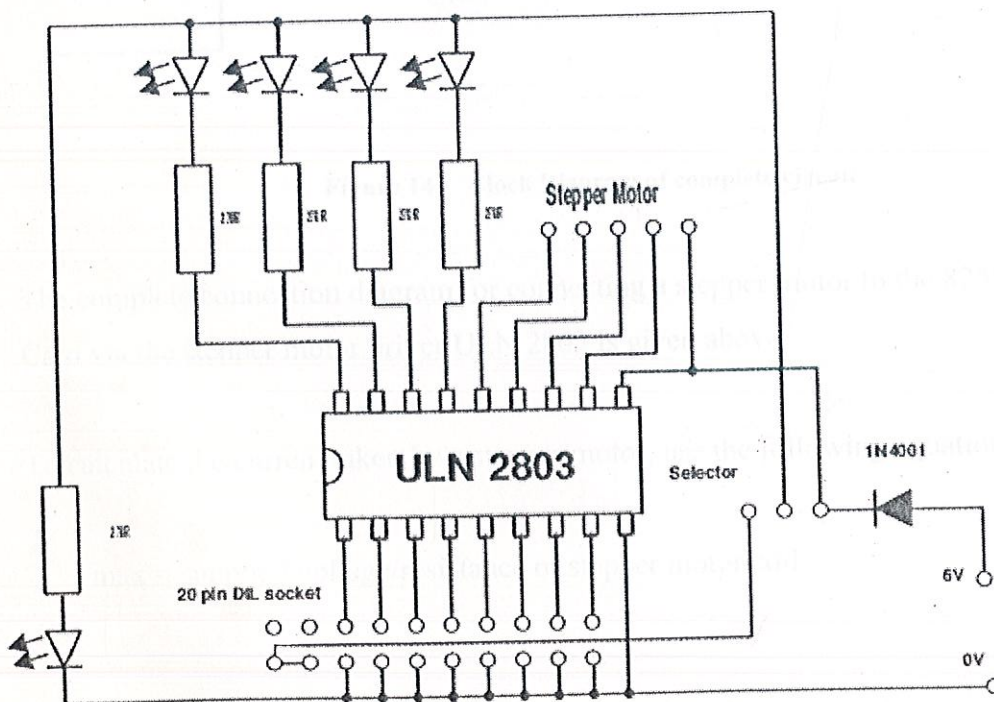


Figure 13. Connection from ULN 2803 to the Stepper Motor



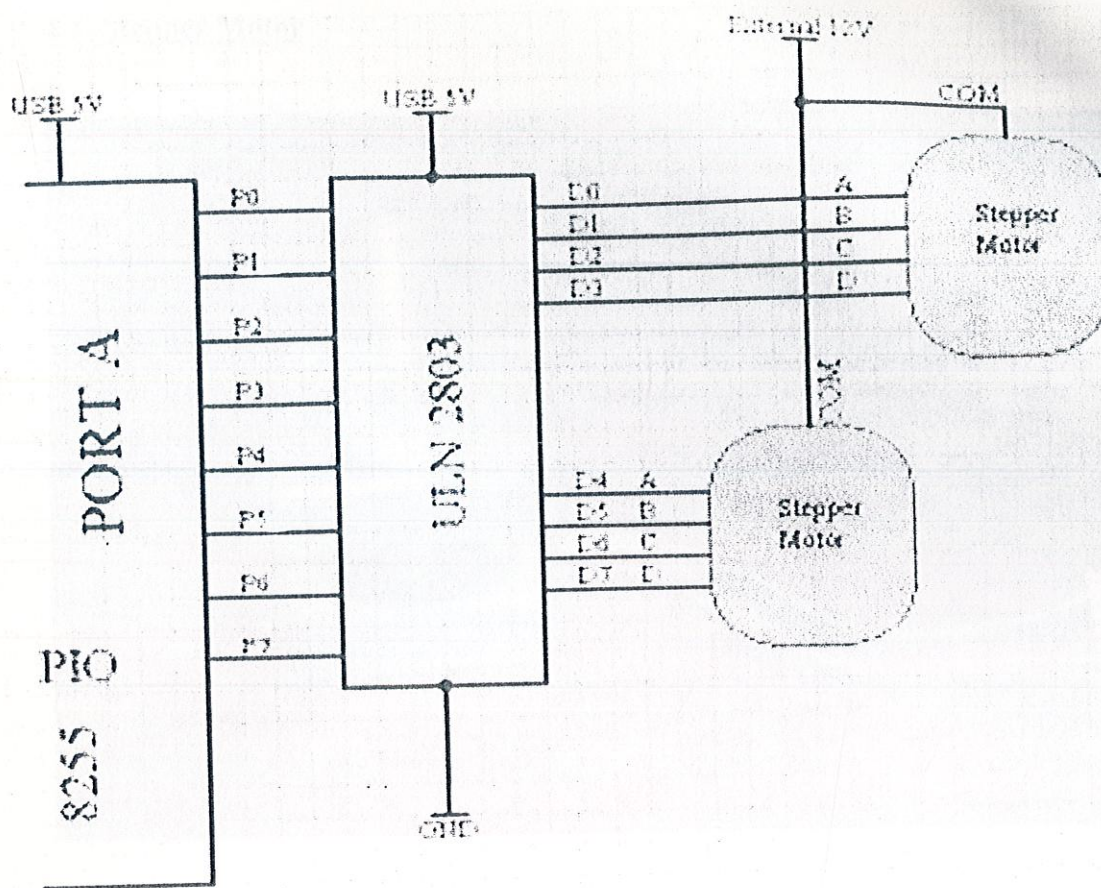


Figure 14. Block Diagram of complete circuit

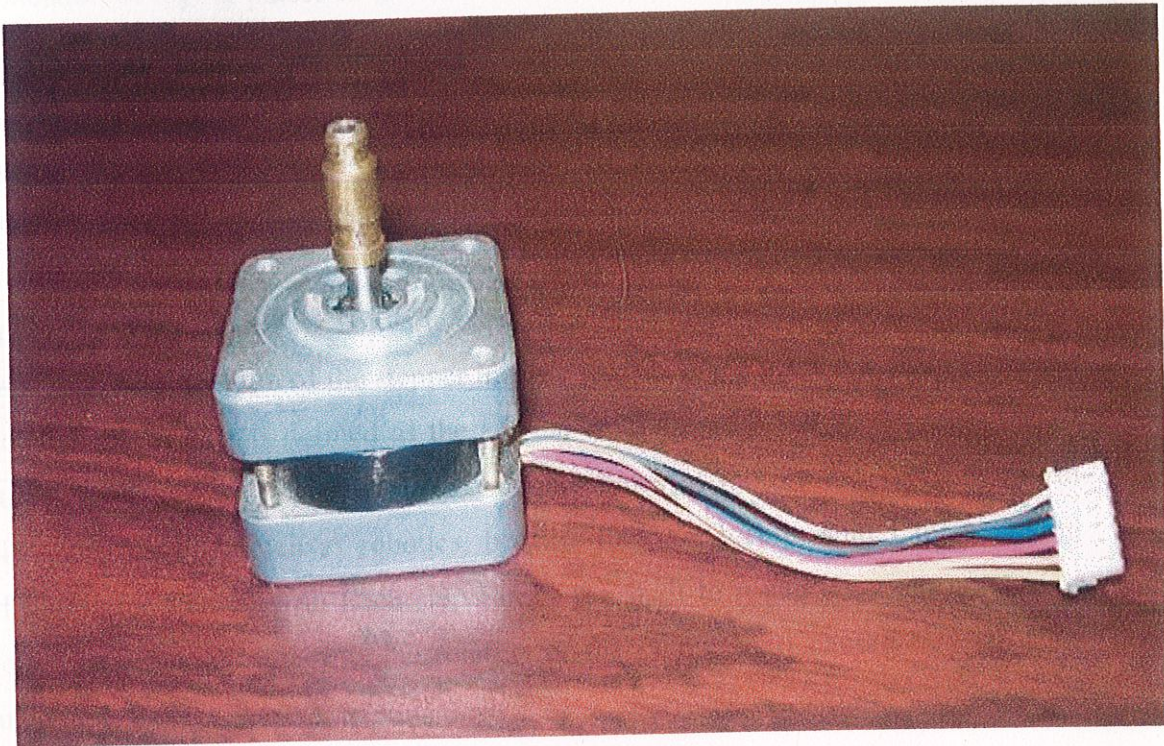
The complete connection diagram for connecting a stepper motor to the 8255 Digital PIO Card via the stepper motor driver ULN 2803 is given above.

To calculate the current taken by a stepper motor, use the following equation:

$$I_{\text{max}} = \text{supplied voltage} / \text{resistance of stepper motor coil}$$



## 2.4 Stepper Motor



**Figure 15. Stepper Motor used in the Project**

A stepper motor is a permanent magnet or variable reluctance dc motor that has the following performance characteristics:

1. Rotation in both Directions,
2. Precision Angular incremental changes,
3. Repetition of accurate motion or velocity profiles,
4. A Holding torque at Zero Speed, and
5. Capability for Digital Control.



A stepper motor can move in accurate angular increments known as steps in response to the application of digital pulses to an electric drive circuit from a digital controller. The number and rate of the pulses control the position and speed of the motor shaft.

Generally, stepper motors are manufactured with steps per revolution of 12, 24, 72, 144, 180, and 200, resulting in shaft increments of 30, 15, 5, 2.5, 2, and 1.8 degrees per step.

A **stepper**, or **stepping motor** converts electronic pulses into proportionate mechanical movement. Each revolution of the stepper motor's shaft is made up of a series of discrete individual steps. A step is defined as the angular rotation produced by the output shaft each time the motor receives a step pulse. These types of motors are very popular in digital control circuits, such as robotics, because they are ideally suited for receiving digital pulses for step control. Each step causes the shaft to rotate a certain number of degrees. A **step angle** represents the rotation of the output shaft caused by each step, measured in degrees.

A **stepper motor** is a brushless, synchronous electric motor that can divide a full rotation into a large number of steps, for example, 200 steps. Thus the motor can be turned to a precise angle. Stepper motors operate differently from normal DC motors, which simply spin when voltage is applied to their terminals. Stepper motors, on the other hand, effectively have multiple "toothed" electromagnets arranged around a central metal gear, as shown at right. To make the motor shaft turn, first one electromagnet is given power, which makes the gear's teeth magnetically attracted to the electromagnet's teeth. When the gear's teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet. So when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one, and from there the process is repeated. Each of those slight rotations is called a "step." In that way, the motor can be turned a precise angle. There are two basic arrangements for the electromagnetic coils: bipolar and unipolar.



Stepping motors can be viewed as electric motors without commutators. Typically, all windings in the motor are part of the stator, and the rotor is either a permanent magnet or, in the case of variable reluctance motors, a toothed block of some magnetically soft material. All of the commutation must be handled externally by the motor controller, and typically, the motors and controllers are designed so that the motor may be held in any fixed position as well as being rotated one way or the other. Most steppers, as they are also known, can be stepped at audio frequencies, allowing them to spin quite quickly, and with an appropriate controller, they may be started and stopped "on a dime" at controlled orientations.

For some applications, there is a choice between using servomotors and stepping motors. Both types of motors offer similar opportunities for precise positioning, but they differ in a number of ways. Servomotors require analog feedback control systems of some type. Typically, this involves a potentiometer to provide feedback about the rotor position, and some mix of circuitry to drive a current through the motor inversely proportional to the difference between the desired position and the current position.

In making a choice between steppers and servos, a number of issues must be considered; which of these will matter depends on the application. For example, the repeatability of positioning done with a stepping motor depends on the geometry of the motor rotor, while the repeatability of positioning done with a servomotor generally depends on the stability of the potentiometer and other analog components in the feedback circuit.

Stepping motors can be used in simple open-loop control systems; these are generally adequate for systems that operate at low accelerations with static loads, but closed loop control may be essential for high accelerations, particularly if they involve variable loads. If a stepper in an open-loop control system is overtorqued, all knowledge of rotor position is lost and the system must be reinitialized; servomotors are not subject to this problem.



Stepper motors are most commonly controlled by microprocessors or custom controller ICs and the current is often switched by stepper motor driver ICs or power transistors. Precise motion is possible but the complexity usually lands the hobbyist's stepper motors in the "maybe someday" parts bin. But steppers may be used for a variety of applications without complex circuitry or programming. At first glance the stepper motor looks a bit intimidating since there are at least four wires and often there are six. Most steppers have two independent windings and some are center-tapped, hence the four or six wires. A quick ohmmeter check will determine which wires belong together and the center-tap may be identified by measuring the resistance between the wires; the center-tap will measure  $1/2$  the total winding resistance to either end of the coil. Tie the wires that belong together in a knot and tie another knot in the center-tap wire for easy identification later. Stepper motors have become quite abundant and are available in all shapes and sizes from many surplus dealers. Experimenters can also salvage excellent steppers from old office and computer equipment.

Steppers move in small increments usually indicated on the label in degrees. To make a stepper motor spin in one direction current is passed through one winding, then the other, then through the first winding with the opposite polarity, then the second with flipped polarity, too. This sequence is repeated for continuous rotation. The direction of rotation depends upon which winding is the "leader" and which is the "follower". The rotation will reverse if either winding is reversed. The center-tapped versions simplify the reversal of current since the center-tap may be tied to Vcc and each end of the coil may be alternately pulled to ground. Non-tapped motors require a bipolar drive voltage or a bit more switching circuitry. If current is applied to both windings, the stepper will settle between two steps (this is often called a "half-step"). Taking the half-step idea to the extreme, one could apply two quadrature sinewaves to the windings and get very smooth rotation. This technique would not be particularly efficient since the controller would be dissipating at least as much power as the motor but, if smooth motion is required, it might be worth a try! Or, for those who don't mind complexity, the sinewaves could be efficiently approximated by using variable duty-cycle pulses.



Steppers make excellent low power generators and surprisingly efficient low power motors for low RPM applications. As a starting point, try connecting the windings of two steppers together. Pick steppers that turn freely so that internal friction doesn't spoil the experiment. When you spin one motor shaft, the other will follow. Admittedly, there is little torque. But it does illustrate that steppers may be used to generate electricity.

Stepper motors, however, behave differently than standard DC motors. First of all, they cannot run freely by themselves. Stepper motors do as their name suggests -- they "step" a little bit at a time. Stepper motors also differ from DC motors in their torque-speed relationship. DC motors generally are not very good at producing high torque at low speeds, without the aid of a gearing mechanism. Stepper motors, on the other hand, work in the opposite manner. They produce the highest torque at *low* speeds. Stepper motors also have another characteristic, *holding torque*, which is not present in DC motors. Holding torque allows a stepper motor to hold its position firmly when not turning. This can be useful for applications where the motor may be starting and stopping, while the force acting against the motor remains present. This eliminates the need for a mechanical brake mechanism. Steppers don't simply respond to a clock signal, they have several windings which need to be energized in the correct sequence before the motor's shaft will rotate. Reversing the order of the sequence will cause the motor to rotate the other way. If the control signals are not sent in the correct order, the motor will not turn properly. It may simply buzz and not move, or it may actually turn, but in a rough or jerky manner. A circuit which is responsible for converting step and direction signals into winding energization patterns is called a *translator*. Most stepper motor control systems include a *driver* in addition to the translator, to handle the current drawn by the motor's windings.

#### **2.4.1 Common Characteristics Of Stepper Motors:**

Stepper motors are not just rated by voltage. The following elements characterize a given Stepper Motor:



### Voltage

Stepper motors usually have a voltage rating. This is either printed directly on the unit, or is specified in the motor's datasheet. Exceeding the rated voltage is sometimes necessary to obtain the desired torque from a given motor, but doing so may produce excessive heat and/or shorten the life of the motor.

### Resistance

Resistance-per-winding is another characteristic of a stepper motor. This resistance will determine current draw of the motor, as well as affect the motor's torque curve and maximum operating speed.

### Degrees per step

This is often the most important factor in choosing a stepper motor for a given application. This factor specifies the number of degrees the shaft will rotate for each full step. Half step operation of the motor will double the number of steps/revolution, and cut the degrees-per-step in half. For unmarked motors, it is often possible to carefully count, by hand, the number of steps per revolution of the motor. The degrees per step can be calculated by dividing 360 by the number of steps in 1 complete revolution. Common degree/step numbers include: 0.72, 1.8, 3.6, 7.5, 15, and even 90. Degrees per step is often referred to as the *resolution* of the motor. As in the case of an unmarked motor, if a motor has only the number of steps/revolution printed on it, dividing 360 by this number will yield the degree/step value.

Stepper motors fall into two basic categories: Permanent magnet and variable reluctance. The type of motor determines the type of drivers, and the type of translator used. Of the permanent magnet stepper motors, there are several "subflavors" available. These include the Unipolar, Bipolar, and Multiphase varieties. They are powered by dc current sources and require digital circuitry to produce the coil energizing sequences for rotation of the motor. Feedback is not always required for control, but the use of an encoder or other position sensor can ensure accuracy when it is essential. The advantage of operating without feedback is that a closed loop control system is not required.



## 2.4.2 Permanent Magnet Stepper Motors

### Bipolar stepper motor

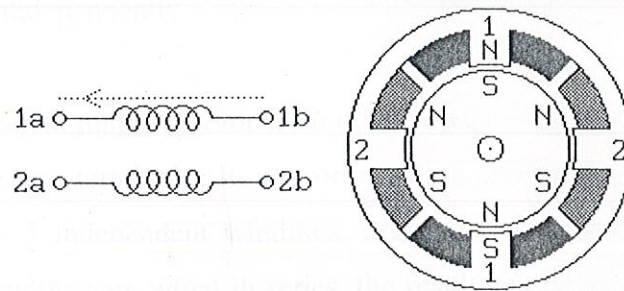


Figure 16. Bipolar Stepper Motor

Bipolar stepper motors generally have four leads, in two pairs, each pair powering one coil. These motors are common and cheap -- as an example, the stepper motor that drives a floppy drive read / write head is a bipolar stepper motor. To rotate the motor, we simply supply the two coils with phase-shifted pulse trains. If you have no data on a given bipolar motor, you can identify which leads correspond to the windings of the motor with an ohmmeter -- just check for continuity between the leads at the end of each winding.

There are only two coils, and current must be sent through a coil first in one direction and then in the other direction; thus the name bipolar. Bipolar motors need more than 4 transistors to operate them, but they are also more powerful than a unipolar motor of the same weight. A circuit known as an "H-bridge" is used to drive Bipolar stepper motors. Each coil of the stepper motor needs its own H-bridge driver circuit. To be able to send current in both directions, engineers can use an H-bridge to control each coil or a step motor driver chip.

Bipolar permanent magnet and hybrid motors are constructed with exactly the same mechanism as is used on unipolar motors, but the two windings are wired more simply, with no center taps. Thus, the motor itself is simpler but the drive circuitry needed to reverse the polarity of each pair of motor poles is more complex.



The drive circuitry for such a motor requires an H-bridge control circuit for each winding. Briefly, an H-bridge allows the polarity of the power applied to each end of each winding to be controlled independently.

To distinguish a bipolar permanent magnet motor from other 4 wire motors, measure the resistances between the different terminals. It is worth noting that some permanent magnet stepping motors have 4 independent windings, organized as two sets of two. Within each set, if the two windings are wired in series, the result can be used as a high voltage bipolar motor. If they are wired in parallel, the result can be used as a low voltage bipolar motor. If they are wired in series with a center tap, the result can be used as a low voltage unipolar motor.

Unipolar  
stepper motor

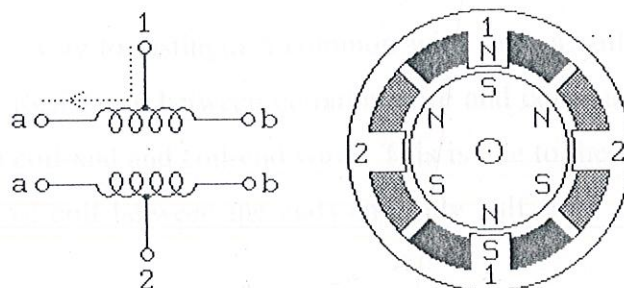


Figure 17. Unipolar Stepper Motor

Unipolar stepper motors generally have six leads, in two triplets, each triplet powering one coil with a center tap. On occasion, you'll find a unipolar stepper with only five leads --here the two "common" leads (center taps) have been tied together internally. Note that a unipolar motor's rotor is identical to a bipolar motor's rotor. Often unipolar motors are controlled by using the center taps as a common point (cathode or anode), and then just switching drive voltage from one end of the given coil to the other.

One can drive unipolar stepper motors in a bipolar fashion (i.e. by just ignoring the center taps). In a unipolar stepper motor, there are four separate electromagnets. To turn the motor, first coil "1" is given current, then it's turned off and coil 2 is given current, then coil 3, then 4, and then 1 again in a repeating pattern. Current is only sent through the coils in one direction; thus the name unipolar.

A unipolar stepper motor will have 5 or 6 wires coming out of it. Four of those wires are each connected to one end of one coil. The extra wire (or 2) is called "common." To operate the motor, the "common" wire(s) is(are) connected to the supply voltage, and the other four wires are connected to ground through transistors, so the transistors control whether current flows or not. This ease of operation makes unipolar motors popular with hobbyists; they are probably the cheapest way to get precise angular movements.

(For the experimenter, one way to distinguish common wire from a coil-end wire is by measuring the resistance. Resistance between common wire and coil-end wire is always half of what it is between coil-end and coil-end wires. This is due to the fact that there is actually twice the length of coil between the ends and only half from center (common wire) to the end.)

### 2.4.3 Variable Reluctance Stepper Motors

Sometimes referred to as *Hybrid* motors, variable reluctance stepper motors are the simplest to control over other types of stepper motors. Their drive sequence is simply to energize each of the windings in order, one after the other (see drive pattern table below). This type of stepper motor will often have only *one* lead, which is the common lead for all the other leads. This type of motor feels like a DC motor when the shaft is spun by hand; it turns freely and you cannot feel the steps. This type of stepper motor is not permanently magnetized like its unipolar and bipolar counterparts.

## CHAPTER 3

### SOFTWARE AND NETWORK

---

#### 3.1 Active X

ActiveX is a set of technologies from Microsoft that enables interactive content for the World Wide Web. Before ActiveX, Web content was static, 2-dimensional text and graphics. With ActiveX, Web sites come alive using multimedia effects, interactive objects, and sophisticated applications that create a user experience comparable to that of high-quality CD-ROM titles. ActiveX provides the glue that ties together a wide assortment of technology building blocks to enable these "active" Web sites. ActiveX includes both client and server technologies.

ActiveX Controls are the interactive objects in a Web page that provide interactive and user-controllable functions and hence enliven the experience of a Web site. ActiveX Documents enable users to view non-HTML documents, such as Microsoft Excel or Word files, through a Web browser. Active Scripting controls the integrated behavior of several ActiveX controls and/or Java Applets from the browser or server. Java™ Virtual Machine is the code that enables any ActiveX-supported browser such as Internet Explorer 3.0 to run Java applets and to integrate Java applets with ActiveX controls. ActiveX Server Framework provides a number of Web server-based functions such as security, database access, and others.

ActiveX brings innovation and interactivity to the Web. Because it is supported by many different languages and tools, it enables developers with varied backgrounds and expertise to bring their creativity to the Web.

ActiveX takes the most creative and innovative software development efforts and enables them to work together seamlessly in a Web site. With thousands of these software components already existing, an exciting collection of interactive objects is available for immediate use by Web producers.

ActiveX makes it fast and easy for developers and Web producers to create unique, interactive Web sites that will make the Internet fundamentally more useful and productive. Web producers don't have to start from scratch and build all the parts of their interactive Web site by hand, because there are already more than 1,000 reusable controls available in the market. Developers and Webmasters can make use of their current expertise to more quickly create compelling content. They can also accommodate a wide range of users, as ActiveX will be supported on multiple operating system platforms.

The ultimate goal of activex is to deliver optimal quality of service to users. For example, because browser speed is one of the primary factors in users' perception of quality, this section aims to provide solutions that allow an HTML document or page to become visible as soon as possible and interactive very shortly thereafter, while allowing controls to retrieve large data blocks in the background.

ActiveX controls are not inherently unsafe, especially when used within a development package like Visual Basic. In that context, assuming the author is trusted, they are no less safe than the other programming code the developer who uses the controls may write.

ActiveX controls are instead unsafe for users of Internet Explorer who turn on the browser's ability to download and activate ActiveX controls within a web page. The problems occur when a user surfs to a non-trusted web page and that web page contains a malicious ActiveX control. This is a very common means of distributing malware such as adware and spyware to unwitting users of Internet Explorer; as such Internet Explorer users should configure their browsers to not install ActiveX controls from untrusted sites. Alternatively, a user can choose to use an alternative web browser that does not use the Trident rendering engine.



### 3.1.1 ActiveX Control (.OCX)

ActiveX Control provides ActiveWire interface to the Internet browser.script language in HTML. Every functions in DLL are also implemented in OCX, so functions callable from .EXE can also called from HTML. Examples of ActiveX controls are: Adobe Reader, Adobe Flash Player, Apple QuickTime Player, Microsoft Windows Media Player, Real Networks RealPlayer, and Sun Java Virtual Machine.

Note: Microsoft Internet Explorer 4.0 and above is currently supported with ActiveX interface.

An OCX is an Object Linking and Embedding (OLE) custom control, 'a special-purpose program that can be created for use by applications running on Microsoft's Windows systems. OCXs provide such functions as handling scroll bar movement and window resizing. If you have a Windows system, you'll find a number of files in your Windows directory with the OCX file name suffix.

Object Linking and Embedding was designed to support compound documents (which contain multiple information types, such as text, graphic images, sound, motion video). The Windows desktop is an example of a compound document and Microsoft used OLE to build it. OLE and the Component Object Model (COM), a more general concept that succeeded OLE, support the development of "plug-and-play" programs that can be written in any language and used dynamically by any application in the system. These programs are known as components and the application in which they are run is known as a container. This component-based approach to application development reduces development time and improves the program capability and quality. Windows application development programs such as PowerBuilder and Microsoft Access take advantage of OCXs.

Microsoft now calls an OCX an ActiveX control, the component object under Microsoft's set of ActiveX technologies, of which the fundamental concept is the Component Object Model (COM) and, in a network, the Distributed Component Object Model (DCOM).

An OCX or ActiveX control is actually implemented as a dynamic link library DLL module.

### 3.1.2 USB ActiveX

It is an ActiveX designed to assist in USB development. This OCX will simplify driving the USB hardware.

#### How to call USB1.ocx

From the menu, select "Project" and go down to "Components". Click on that and you will see a list of all the registered components. If USB1.ocx is still not there, click "Browse" and go to the c:\windows\system directory. Click on USB1.ocx and you will see it appear on the list of components.

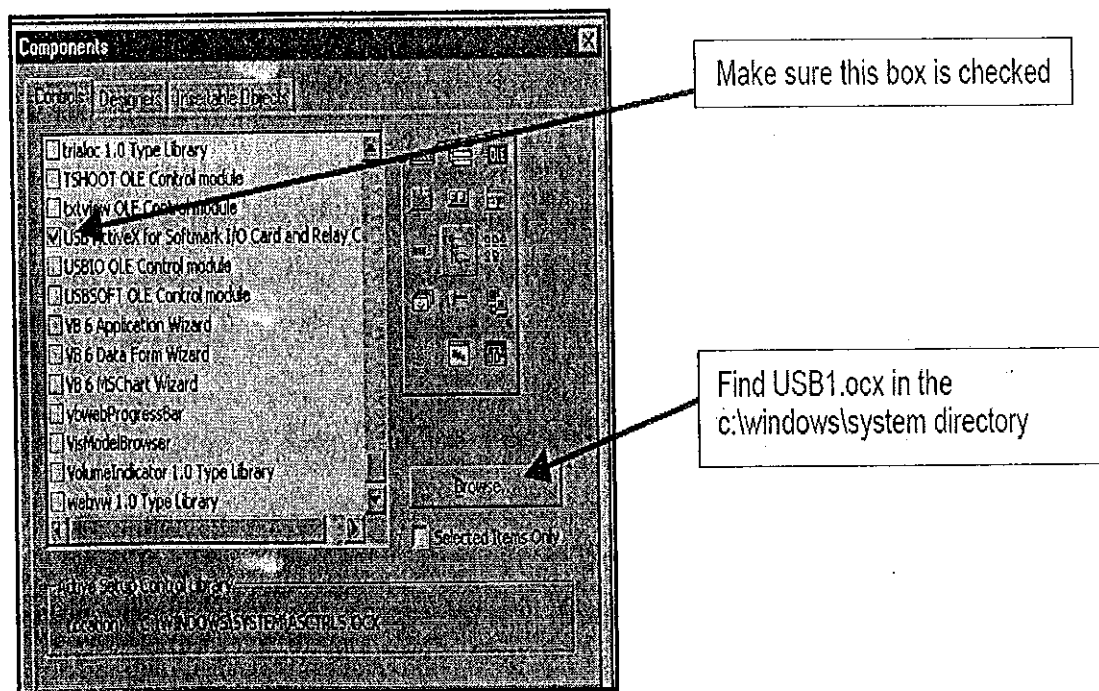


Figure 18. Adding USB.OCX

After checking "USB Active X", the OCX will be added to the tool bar and will be ready to be used.

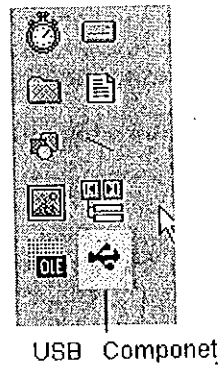


Figure 19. USB.OCX Installed

### 3.1.3 Windows Sockets

**Windows Sockets** which was later shortened to **Winsock**, is a technical specification that defines how Windows network software should access network services, especially TCP/IP. Windows socket is considered one of the most important building-blocks in the Windows family of operating systems. Its specification defines a network programming interface for Microsoft Windows which is based on the "socket" paradigm. It defines a standard interface between a Windows TCP/IP client application (such as an FTP client) and the underlying TCP/IP protocol stack.

The nomenclature is based on the Sockets API model used in Berkeley UNIX for communications between programs. It encompasses both familiar Berkeley socket style routines and a set of Windows-specific extensions designed to allow the programmer to take advantage of the message-driven nature of Windows. Without it, Windows would not offer easy and transparent access to Internet-enabled applications, such as the one used for viewing any web page - the web browser (Internet Explorer, Mozilla Firefox, Netscape, Opera, etc.)



Network software which conforms to this Windows Sockets specification will be considered "Windows Sockets Compliant". Suppliers of interfaces which are "Windows Sockets Compliant" shall be referred to as "Windows Sockets Suppliers". To be Windows Sockets Compliant, a vendor must implement 100% of this Windows Sockets specification. Regardless of which product the user prefers, it will still work with Windows Sockets, which exemplifies the transparency and compatibility work that was the goal in the development of the **Windows Sockets API**.

Applications which are capable of operating with any "Windows Sockets Compliant" protocol implementation will be considered as having a "Windows Sockets Interface" and will be referred to as "Windows Sockets Applications". Initially, all the participating developers resisted the shortening of the name to Winsock for a long time, since there was much confusion among users between the API and the DLL library file (winsock.dll) which only exposed the common WSA interfaces to applications above it. Users would commonly believe that only making sure the DLL file was present on a system would provide full TCP/IP protocol support.

The basic building block for communication is the socket. A socket is an endpoint of communication to which a name may be bound. Each socket in use has a type and an associated process. Sockets exist within communication domains. A communication domain is an abstraction introduced to bundle common properties of threads communicating through sockets. Sockets normally exchange data only with sockets in the same domain it may be possible to cross domain boundaries, but only if some translation process is performed. The Windows Sockets facilities support a single communication domain: the Internet domain, which is used by processes which communicate using the Internet Protocol Suite. Its future versions of this specification may include additional domains.

The Winsock specification defines two interfaces: the API used by application developers, and the SPI, which provides a means for network software developers to add new protocol modules to the system. Each interface represents a contract.

The API guarantees that a conforming application will function correctly with a conformant protocol implementation from any network software vendor. The SPI contract guarantees that a conforming protocol module may be added to Windows and will thereby be usable by an API-conformant application. Although these contracts were important when Winsock was first released, they are now of only academic interest. Microsoft has shipped a high-quality TCP/IP stack with all recent versions of Windows, and there are no significant independent alternatives. Nor has there been significant interest in implementing protocols other than TCP/IP.

Winsock is based on BSD sockets, but provides additional functionality to allow the API to comply with the standard Windows programming model. The Winsock API covered almost all the features of the BSD sockets API, but there were some unavoidable obstacles which mostly arose out of fundamental differences between Windows and Unix (though to be fair Winsock differed less from BSD sockets than the latter did from STREAMS).

Sockets are typed according to the communication properties visible to a user. Applications are presumed to communicate only between sockets of the same type, although there is nothing that prevents communication between sockets of different types should the underlying communication protocols support this. Two types of sockets currently are available to a user. A stream socket provides for the bi-directional, reliable, sequenced, and unduplicated flow of data without record boundaries.

By using a datagram socket, it is possible to send broadcast packets on many networks supported by the system. The network itself must support broadcast: the system provides no simulation of broadcast in software. Broadcast messages can place a high load on a network. Consequently, the ability to send broadcast packets has been limited to sockets which are explicitly marked as allowing broadcasting. Broadcast is typically used for one of two reasons: it is desired to find a resource on a local network without prior knowledge of its address, or important functions such as routing require that information be sent to all accessible neighbors.

A datagram socket supports bi-directional flow of data which is not promised to be sequenced, reliable, or unduplicated. That is, a process receiving messages on a datagram socket may find messages duplicated, and, possibly, in an order different from the order in which it was sent. An important characteristic of a datagram socket is that record boundaries in data are preserved. Datagram sockets closely model the facilities found in many contemporary packet switched networks such as Ethernet.

It defines a standard interface between a Windows TCP/IP client application (such as an FTP client or a Gopher client) and the underlying TCP/IP protocol stack. The nomenclature is based on the Sockets API model used in Berkeley UNIX for communications between programs. It must be considered one of the most important building-blocks in the Windows family of operating systems, since without it, Windows would not offer easy and transparent access to Internet-enabled applications, such as the one used for viewing this page - the web browser (Internet Explorer, Mozilla Firefox, Netscape, Opera, etc.) Regardless of which product the user prefers, it will still work with Windows Sockets, which exemplifies the transparency and compatibility work that was the goal in the development of the Windows Sockets API. Initially, all the participating developers resisted the shortening of the name to Winsock for a long time, since there was much confusion among users between the API and the DLL library file (winsock.dll) which only exposed the common WSA interfaces to applications above it. Users would commonly believe that only making sure the DLL file was present on a system would provide full TCP/IP protocol support.

The **Winsock** component shown in the lower left corner (named Client Motor isn't part of the standard VB toolbox. Rather it is added using the following steps:

- By clicking Project then Components and then selecting Microsoft Winsock Control 6.0. The following screen shot will be seen



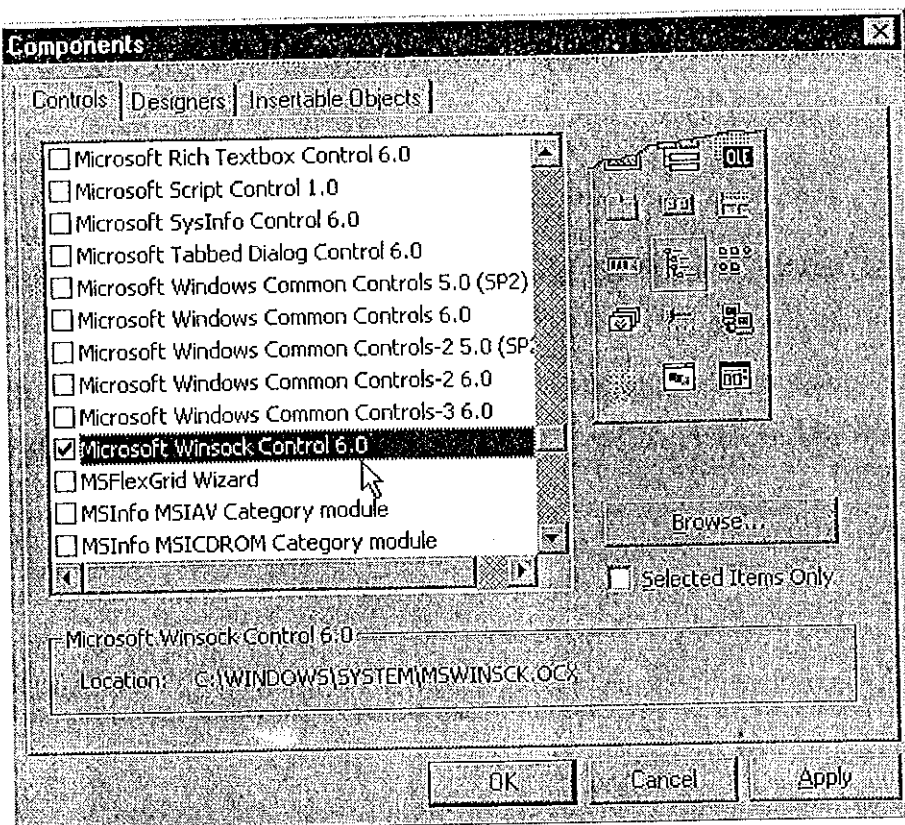
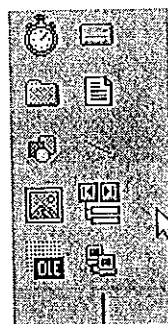


Figure 20. Selecting Winsock Control

- Clicking OK will then export the Winsock Component to the VB toolbox as shown below.



Winsock component  
is added

Figure 21. Winsock Installed

### 3.2 The Client Program

Running Client Motor will prompt to enter an IP address. We have a corresponding server program as mentioned earlier, with the help of which we can have proper communication between the server and client. This code uses Microsoft's Winsock component. Winsock is Microsoft's high-level component that allows a VB programmer to quickly develop Internet-based programs like Chat, FTP and Browsers.

Below is a screen shot of the VB GUI Client Motor.

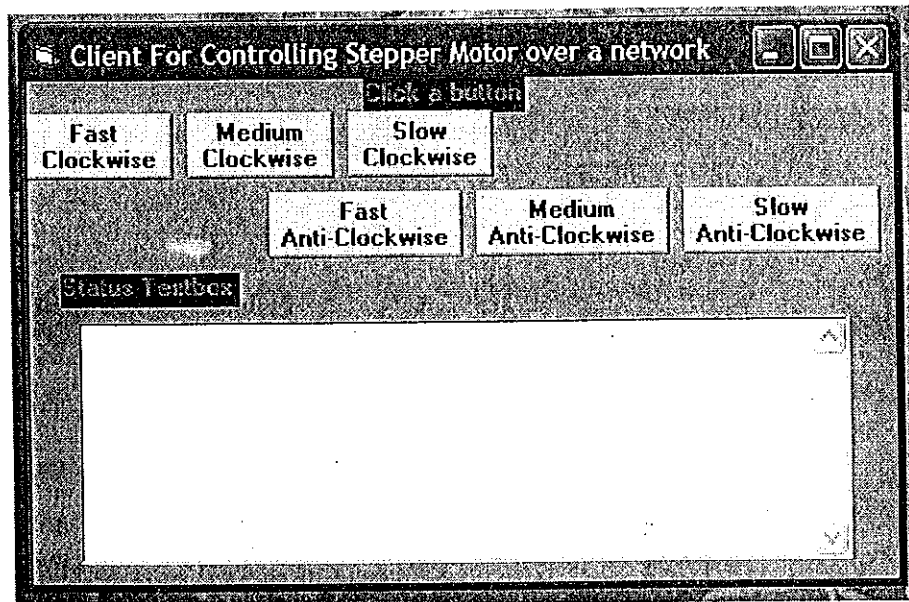


Figure 22. VB GUI Client Motor

### 3.2.1 Coding for the Client Software

```
Private Sub cmdFive_Click()  
    ' Send five to server  
    Call tcpClient.SendData("99")  
    txtOutput.Text = txtOutput.Text & _  
        "Client clicked Slow Anti-Clockwise" & vbCrLf  
    txtOutput.SelStart = Len(txtOutput.Text)  
End Sub  
  
Private Sub cmdFour_Click()  
    ' Send four to server  
    Call tcpClient.SendData("49")  
    txtOutput.Text = txtOutput.Text & _  
        "Client clicked Medium Anti-Clockwise" & vbCrLf  
    txtOutput.SelStart = Len(txtOutput.Text)  
End Sub  
  
Private Sub cmdOne_Click()  
    ' Send one to server  
    Call tcpClient.SendData("50")  
    txtOutput.Text = txtOutput.Text & _  
        "Client clicked Medium Clockwise" & vbCrLf  
    txtOutput.SelStart = Len(txtOutput.Text)  
End Sub  
  
Private Sub cmdThree_Click()  
    ' Send three to server  
    Call tcpClient.SendData("1")  
    txtOutput.Text = txtOutput.Text & _  
        "Client clicked Fast Anti-Clockwise" & vbCrLf
```



```
txtOutput.SelStart = Len(txtOutput.Text)
End Sub
```

```
Private Sub cmdTwo_Click()
    ' Send two to server
    Call tcpClient.SendData("100")
    txtOutput.Text = txtOutput.Text & _
        "Client clicked Slow Clockwise" & vbCrLf
    txtOutput.SelStart = Len(txtOutput.Text)
End Sub
```

```
Private Sub cmdZero_Click()
    ' Send zero to server
    Call tcpClient.SendData("0")
    txtOutput.Text = txtOutput.Text & _
        "Client clicked Fast Clockwise" & vbCrLf
    txtOutput.SelStart = Len(txtOutput.Text)
End Sub
```

```
Private Sub Form_Load()
    ' Set up local port and wait for connection
    tcpClient.RemoteHost = InputBox("Enter the remote host IP Address", _
        "IP Address", "localhost")

    If tcpClient.RemoteHost = "" Then
        tcpClient.RemoteHost = "localhost"
    End If

    tcpClient.RemotePort = 5000 ' server port
    Call tcpClient.Connect ' connect to RemoteHost address
```

End Sub

Private Sub tcpClient\_Connect()

' When connection occurs, display a message

txtOutput.Text = "Connected to IP address: " & \_  
tcpClient.RemoteHostIP & vbCrLf & "Port #: " & \_  
tcpClient.RemotePort & vbCrLf & vbCrLf

End Sub

Private Sub tcpClient\_DataArrival(ByVal bytesTotal As Long)

Dim message As String

Call tcpClient.GetData(message) ' get data from server

txtOutput.Text = txtOutput.Text & message & vbCrLf

txtOutput.SelStart = Len(txtOutput.Text)

End Sub

Private Sub tcpClient\_Error(ByVal Number As Integer, Description As String, ByVal  
Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext  
As Long, CancelDisplay As Boolean)

' If the client fails to connect to server, then this code executes

Dim result As Integer

result = MsgBox(Source & ": " & Description & vbCrLf & "Doh! Can't connect to  
server!", \_

vbOKOnly, "TCP/IP Error")

' Source variable is the control (winsock in this case) causing the error

' Description variable cites the error message

' vbOKOnly is the control button

' TCP/IP Error is the MsgBox caption

End

### 3.3 The Server Program

Server Motor is a server code also using Microsoft's Winsock component. Executing Server Motor on a PC connected to the Internet will listen to connection attempts made by the client.

Below is a screen shot of the VB GUI Server Motor.

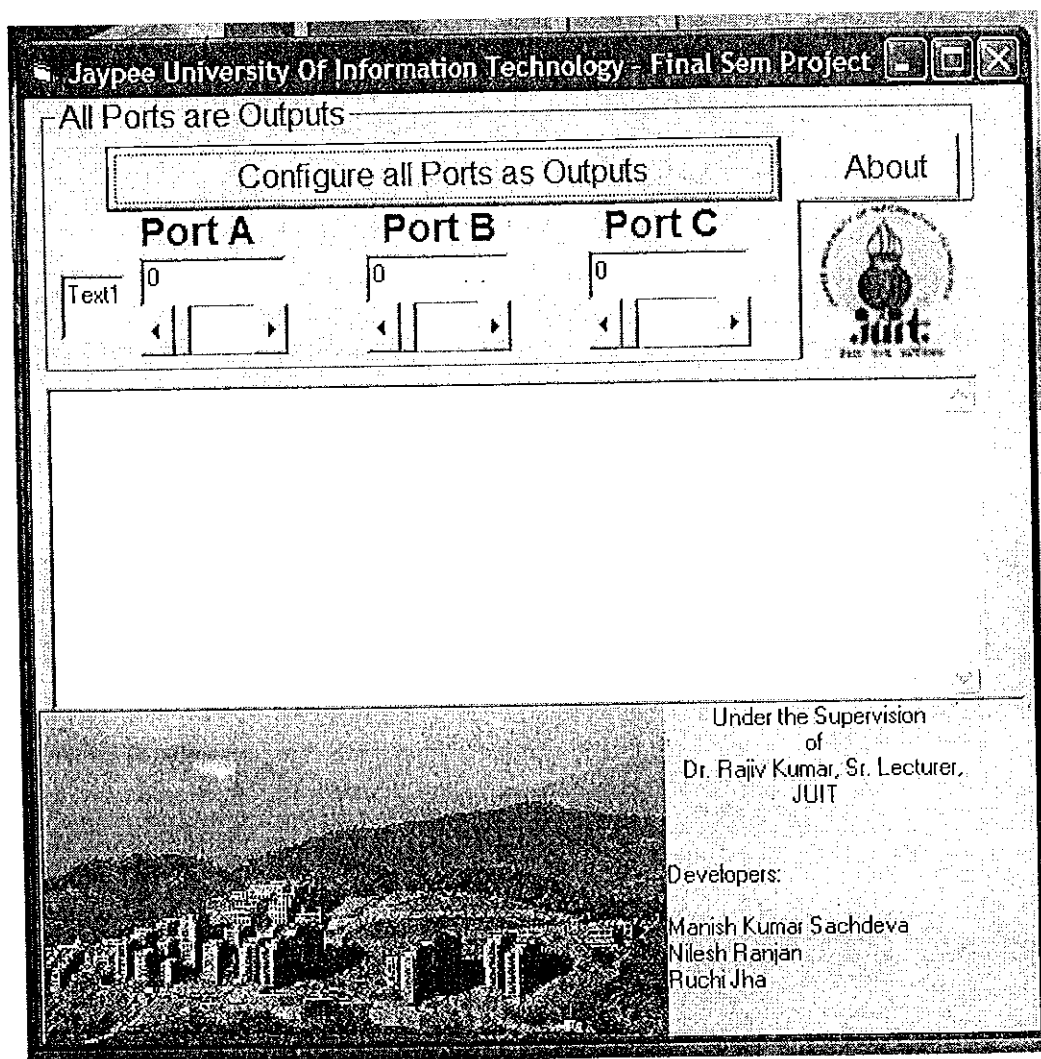


Figure 23. VB GUI Server Motor



### 3.3.1 Coding for the Server Software

```
Private Sub Command1_Click()
Timer1.Enabled = False
Timer2.Enabled = False
'CONFIGURE ALL PORTS AS OUTPUTS
'SET FOR TRANSFER CONFIGURATION
'D2=1 A0
'D0=1 A1
'D3=1 RD
'D1=0 WR
USB1.WR_P1 ((D2 * 1) + (D0 * 1) + (D3 * 1) + (D1 * 1))
'SET ALL PORTS AS OUTPUTS.128 TO BE SENT TO P0
USB1.WR_P0 (128) 'SET ALL PORTS AS OUTPUTS
'SET WR LINE TO HIGH
USB1.WR_P1 ((D2 * 1) + (D0 * 1) + (D3 * 1) + (D1 * 0))
'SET WR LINE TO LOW
USB1.WR_P1 ((D2 * 1) + (D0 * 1) + (D3 * 1) + (D1 * 1))
'Sleep (20)
End Sub

Private Sub Form_Load()
tcpServer.LocalPort = 5000
    Call tcpServer.Listen
D0 = 1
D1 = 2
D2 = 4
D3 = 8
End Sub

Private Sub HScroll4_Change()
'Text3 = HScroll4.Value
For i = 1 To CInt(Text10.Text)
```

```

Text3 = 8
Sleep (0)
Call WR_PORT_A
Text3 = 4
Sleep (0)
Call WR_PORT_A
Text3 = 2
Sleep (0)
Call WR_PORT_A
Text3 = 1
Sleep (0)
Call WR_PORT_A
Next i
End Sub

Private Sub HScroll4_Scroll()
'Text3 = HScroll4.Value
Call WR_PORT_A
End Sub

Private Sub tcpServer_Close()
    Call tcpServer.Close ' Client closed, server should too
    txtOutput.Text = txtOutput.Text & "Client closed connection." & vbCrLf
    txtOutput.SelStart = Len(txtOutput.Text)
    Call tcpServer.Listen ' listen for next connection
End Sub

Private Sub tcpServer_ConnectionRequest(ByVal requestID As Long)
    ' Ensure that tcpServer is closed
    ' before accepting a new connection
    If tcpServer.State <> sckClosed Then
        Call tcpServer.Close
    End If

```

```

Call tcpServer.Accept(requestID) ' Accept connection
' Display following message on Server Status window
txtOutput.Text = "Client from IP Address: " & _
    tcpServer.RemoteHostIP & " is successful" & vbCrLf & _
    "Port #: " & tcpServer.RemotePort & vbCrLf
End Sub

Private Sub tcpServer_DataArrival(ByVal bytesTotal As Long)
    Dim messageFromClient As String ' The button Client clicked
    Dim numericValue As Integer    ' The numeric value of string

    Call tcpServer.GetData(messageFromClient) ' Get Client's button click
    ' Convert Client's button click value to a numerica value
    numericValue = Val(messageFromClient)
    ' Display which button client clicked in Server's Status window

    txtOutput.SelStart = Len(txtOutput.Text)

    ' Light up the binary equivalent of client's button click on LEDs
    Text10 = 90
    'Clockwise direction
    If numericValue = 0 Then
        txtOutput.Text = txtOutput.Text & "Client clicked the button with Fast speed in
Clockwise direction" & vbCrLf
        For i = 1 To CInt(Text10.Text)
            Text3 = 1
            Sleep (numericValue)
            Call WR_PORT_A
            Text3 = 2
            Sleep (numericValue)

```



```

Call WR_PORT_A
Text3 = 4
Sleep (numericValue)
Call WR_PORT_A
Text3 = 8
Sleep (numericValue)
Call WR_PORT_A
Next i
End If
If numericValue = 50 Then
txtOutput.Text = txtOutput.Text & "Client clicked the button with Medium speed in
Clockwise direction" & vbCrLf
    For i = 1 To CInt(Text10.Text)
Text3 = 1
Sleep (numericValue)
Call WR_PORT_A
Text3 = 2
Sleep (numericValue)
Call WR_PORT_A
Text3 = 4
Sleep (numericValue)
Call WR_PORT_A
Text3 = 8
Sleep (numericValue)
Call WR_PORT_A
Next i
End If
If numericValue = 100 Then
txtOutput.Text = txtOutput.Text & "Client clicked the button with Slow speed in
Clockwise direction" & vbCrLf
    For i = 1 To CInt(Text10.Text)

```

```

Text3 = 1
Sleep (numericValue)
Call WR_PORT_A
Text3 = 2
Sleep (numericValue)
Call WR_PORT_A
Text3 = 4
Sleep (numericValue)
Call WR_PORT_A
Text3 = 8
Sleep (numericValue)
Call WR_PORT_A
Next i
End If

```

```

' Set acknowledgement message back to client
Call tcpServer.SendData("Server finished running the motor")

```

```

End Sub

```

```

Private Sub tcpServer_Error(ByVal Number As Integer, Description As String, ByVal
Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext
As Long, CancelDisplay As Boolean)

```

```

    Dim result As Integer
    result = MsgBox(Source & ": " & Description, _
        vbOKOnly, "TCP/IP Error")

```

```

End
End Sub

```

Server Motor program must be executed before running Client Motor since it requires the server's IP address. When Client Motor first executes, Form\_Load is launched; an input box prompts the user to enter the server's IP address. By default it uses localhost. Next, Client Motor.RemotePort uses the value 5000 and is called a port number. This number can range from 1024 to 65535 (typically port numbers less than 1024 are reserved for system services). However, the port numbers of the client and server must match. The corresponding server code will also use 5000. Clicking OK after entering an IP address will attempt a client-server connection by calling Client Motor\_Connect() which then enables the user to send instructions. If the connection succeeds, the user can send an instruction.

Client Motor\_SendData sends the string Client >>> " plus whatever instructions we have sent. It prints whatever instructions we typed into the txtOutput box, along with some carriage returns and line feeds (vbCrLf).

Any instructions sent from the server are handled by the Client Motor\_DataArrival() procedure. GetData is a property of the Winsock control. As such Client Motor.GetData collects any instructions sent from the server into the string variable message. The string is then printed in the txtOutput text box.

Killing the program will execute the Client Motor\_Close() procedure. It greys out the Send Data button, shuts down the client and prints out the message Server closed connection. Form\_Terminate also closes the client's Winsock control using a Client Motor.Close call.

Client Motor\_Error() handles any errors. A common error might be a mistyped IP address. Form\_Resize handles any attempts to resize the GUI.



## CHAPTER 4

### WORKING AND OPERATION

---

#### Here is how the system works:

- Control data is passed between the server and client using WinSock TCP/IP.
- When a command is sent from the client to the server, the server determines the instruction.
- The server sends commands to the 8255 card by way of the OCX ActiveX file.
- Once the server has completed sending it to 8255, it further sends the instruction to the stepper motor.
- Upon completion, the server sends a message to the client to indicate that the motor has received the instruction.

#### Operating Procedure

These steps must be taken in the following order to successfully run the internet appliance:

- Start Up Procedure:

- Connect appliance to the server PC.
- Confirm network connectivity between server and client PCs.
- Determine IP address of server PC.
- Start the Stepper Control Server software on the server PC.
- Start the Stepper Control Client software on the client PC.
- Enter the server PC's IP address when the Stepper Control Client software requests it and the data is transferred.

- Shut Down Procedure:

- Close Stepper Control Client software.
- Close Stepper Control Server software.

Two computers for communicating namely the server and the client computers



Figure 24. Client Computer

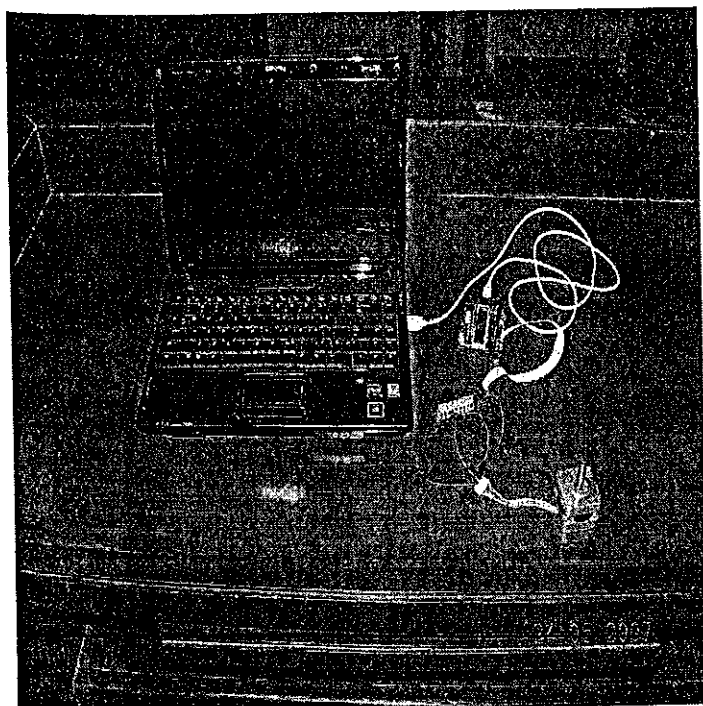
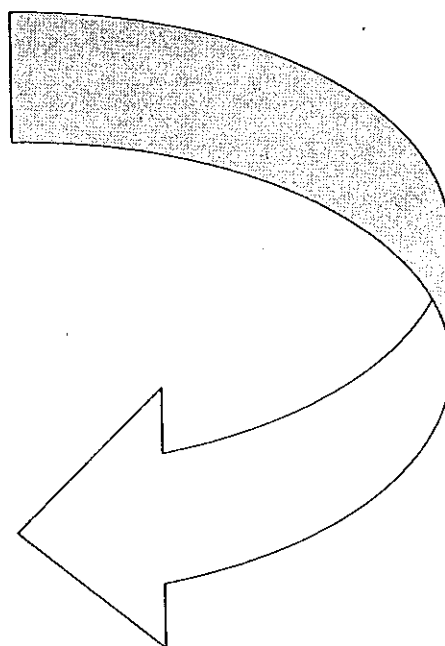


Figure 25. Server Computer



Successful Connection between the Server and the Client Computers

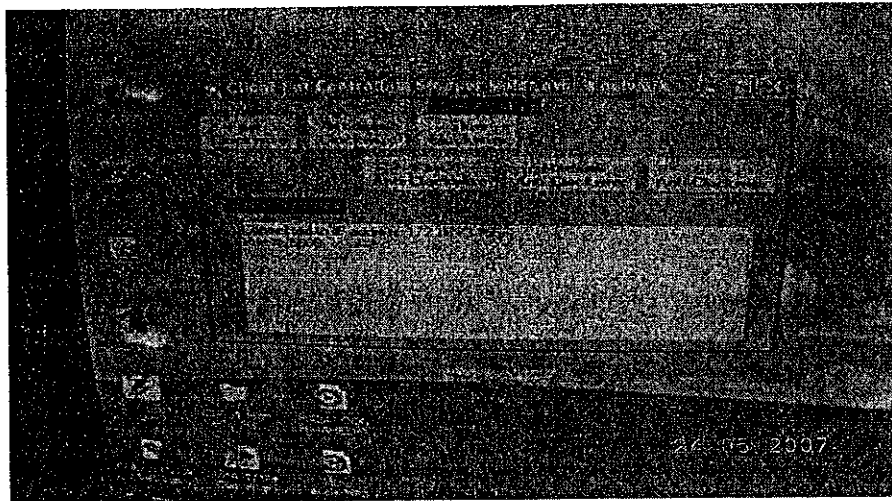


Figure 26. Client Software

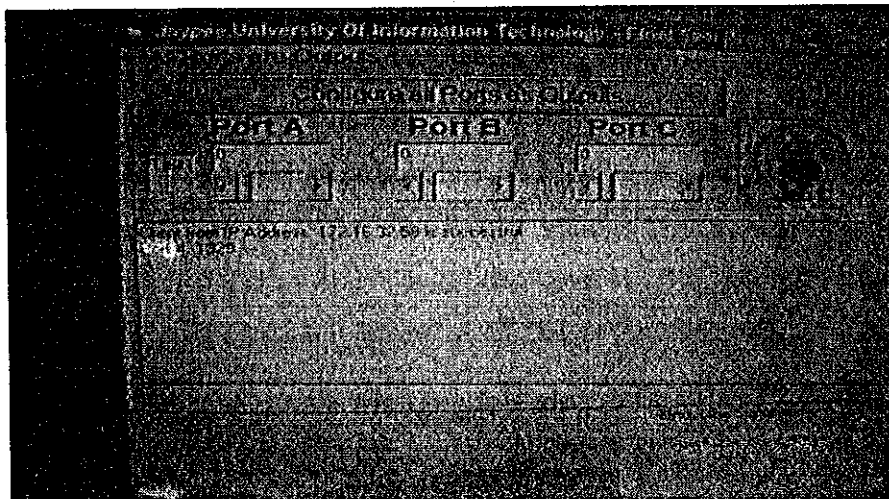
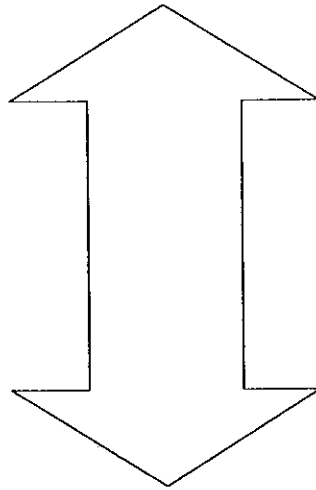


Figure 27. Server Software



## CHAPTER 5

### FUTURE VISION AND CONCLUSION

---

Future potential work derived from this tutorial includes using a web server to display data acquired from sensors interfaced to the server PC. Clients would be thus able to monitor data remotely over the Internet. Webcams can also be attached to the server so that Clients can get visual feedback on whether the remote server indeed actuate your peripheral. Some of these concepts are cited in the Course's web page. In the big picture, the building blocks for controlling and monitor hardware over the Internet can be used for home or industrial automation, robotics and data acquisition. In turning on/off an LED you really can light up the world.

Applications for the PC are numerous but could be custom equipment control, monitoring system, calibration etc. The worldwide familiarity, standardization, and availability of Internet, along with its current and potential performance levels, has prompted increased consideration of Internet as a viable technology for distributing data acquisition tasks, control systems, monitoring etc.

Internet is now available as a direct communications link on a growing number of data acquisition instruments and industrial I/O devices (e.g. Omega iSeries devices ).

Today, data acquisition and I/O products are usually used in conjunction with a desktop, industrial or notebook PC. Plug-in boards with high-speed data acquisition capabilities are widely available for PCI and ISA. Very important advantage of Internet is its worldwide connectivity and built-in compability with any PC platform. When a piece of equipment is networked through the internet, two main things happen:

- Expensive and rare pieces of equipment may be brought closer to users all over the world. This allows those who can not justify purchasing equipment to use it after an agreement is made with a host organization.

- Organizations considering purchase of expensive equipment may be able to diffuse the cost of the equipment through renting time on the machine over the internet. The customer base for this use is of course worldwide.

The web gives the ability to view and control the process on a computer, monitoring real-time information via the Internet. No special acquisition software needs to be installed on the computer, only a web browser and a TCP/IP client/server program. Distance between Client and Server is not important, one could set up peripheral and server anywhere in the world and user(s) could do control, monitoring, communications, data acquisition etc.

The web-enabled unit could be with features like e-mail notification, alarms, and security password. Two or more clients could work together at time on the same application, communicate, control and monitoring real-time information from Internet enabled PCs anywhere in the world.

## BIBLIOGRAPHY

### Books :-

1. Ramesh Gaonkar. *Microprocessor Architecture, Programming, and Applications with the 8085*. Penram International Publishing(India)Private Limited.
2. Andrew S. Tanenbaum. *Computer Networks Fourth Edition*. Pearson Education.
3. Peter Wright . *Beginning Visual Basic 6 Objects*. 1998 Wrox Press.

### Web Pages :-

[www.pages.drexel.edu/~zh25/](http://www.pages.drexel.edu/~zh25/)

[www.boondog.com/](http://www.boondog.com/)

[www.ar.com.au/~softmark/](http://www.ar.com.au/~softmark/)

[www.en.wikipedia.org/wiki/](http://www.en.wikipedia.org/wiki/)

[www.cs.uiowa.edu/~jones/step/types.html](http://www.cs.uiowa.edu/~jones/step/types.html)

[www.educyclopedia.be/electronics/motorstepdriver.htm](http://www.educyclopedia.be/electronics/motorstepdriver.htm)

[www.electronics-diy.com/electronics/stepper\\_motors.php](http://www.electronics-diy.com/electronics/stepper_motors.php)

[www.roboticsindia.com/](http://www.roboticsindia.com/)

[www.doc.ic.ac.uk/~ih/doc/stepper/control2/connect.html](http://www.doc.ic.ac.uk/~ih/doc/stepper/control2/connect.html)