PC TO PC FIBER OPTIC COMMUNCATION

A final year project submitted to Mekelle University

for award of degree of

BACHELOR OF SCIENCE

IN

ELECTRICAL AND COMPUTER ENGINEERING

Under The Supervision of

Dr. DILLIP S.MALI

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

TIGRAY INSTITUTE OF TECHNOLOGY

MEKELLE UNIVERSITY

MEKELLE

JUNE 2010

ACKNOWLEDGEMENT

There is much difference between saying and doing. So, to get theoretical knowledge is not enough. It provides only the base. We faced so many difficulties in this project because of making this project we had to learn C# language. We believe that this type of project will increase our strength regarding to our subjects. And we believe that “Practical leads a man towards Performance”.

We would like to express our sincere thanks to Dr.Dillip S.Mali for his kind, ordinal and valuable guidance and support during the project work.

We are also thankful to Ato Sisay Teklay Head of ECE for providing lab facilities.

Finally, we would like to thank our colleagues who were giving support and help us when we are in trouble. At last we would like thank for the Department of ECE who gave us the chance to make the project.

CERTIFICATE

This is to certify that the project entitled "*PC TO PC FIBER OPTIC COMMUNICATION*" which is being submitted here for the award of the degree of Bachelor of Science in Electrical Engineering of Mekelle University is the result of the project work carried out by:

- **BIRHANU TESFAY**
- **EPHREM HAILU**
- **GOITOM YISFA**
- **HAFTAMU GEBREZGABHER**

under my supervision and guidance. To the best of our knowledge and belief, the work embodied in this project has not formed earlier the basis for the award of any degree.

_____	_____
Ato Sisay Teklay (M.Tech)	Dr. Dilip S. Mali
Head of Department	Advisor, Associate Professor
Electrical & Computer Engineering	Electrical & Computer Engineering
Mekelle University	Mekelle University
Mekelle	Mekelle

EXECUTIVE SUMMARY

The project PC TO PC FIBER OPTIC COMMUNICATION deals with data transfer from one computer to another. It uses C# programming and the serial ports of the Computer. The ports are programmed in C#. MAX 232 is used to convert RS 232 logic to TTL logic and then an optical transmitter circuit is used to transmit data via fiber optic cable.

The optical transmitter circuit has a LED which is matched as far as the cable and MAX232 is concerned. At the receiver an optical receiver circuit is used which receives data using a photo diode and a MAX 232 again to convert TTL logic to RS 232 for the serial port at the receiving end computer.

The desired baud rate is set using the program. For transmitting data the program is executed once and whatever is present at the serial port is sent to the other computer via the fiber optic cable. At the receiver the program is executed to receive data on the serial port.

TABLE OF CONTENTS

ACKNOWLEDGEMENT……I

EXECUTIVE SUMMARY…….....III

APPENDECIES…….VI

LIST OF FIGURES……..VII

LIST OF TABLES……..VIII

ABBREVIATION.....IX

BACK GROUND……X

CHAPTER 1

INTRODUCTION……1

1.1. CONCEPT USED…….1

1.1.1.GUIDED OPTICAL COMMUNICATION…….1

1.1.2.APPLICATIONS……2

1.2. ABOUT FIBER OPTIC CABL.....2

1.2.1.GENERAL INFORMATION…….2

1.2.2.WHY USE A FIBER OPTIC CABLE……3

CHAPTER 2

COMPONENT STUDY……4

2.1. FIBER OPTIC CABLE……4

2.2. SELECTION OF PC PORT…….6

2.3. CONNECTORS……7

2.3.1.DB-9 CONNECTOR (9 –Pin Connector)……..7

2.3.2.DB-25 CONNECTOR (25 –Pin Connector)…….8

2.4. MAX232 CONVERTER IC……10

2.5. PHOTO DIOD……...…11

2.6. LIGHTEMITTINGDIODE(LED)……..13

2.7. CAPACITOR ……15

2.8. TRANSISTOR (PNP 2N3906)……..16

CHAPTER 3

PROJECT DESCRIPTION……17

3.1. HARD WARE……17

3.1.1.Overall System Block Diagram……17

3.1.2.TRANSMITTER SECTION……18

3.1.3.RECEIVER SECTION……18

3.2. SOFTWARE……20

3.2.1.PROGRAM FLOW CHART……20

3.2.2.GRAPHICAL USER INTERFACE (GUI)……22

CHAPTER 4

CONCLUSION AND DESCRIPTION……25

4.1. CONCLUSION……25

4.2. RECOMMENDATION FOR THE FUTURE WORK……25

REFERENCES……...54

APPENDECIES

Appendix I: Code for the Login page……....27

Appendix II: Code for the Main Page……29

Appendix III: Code for the Communication Manager…….……...33

Appendix IV: pc to pc chatting in console application ……... …….……46

Appendix V: Group Members ……. …….…….53

LIST OF FIGURES

Figure1. Fiber optic section……4

Figure2. Fiber Optic Light Propagation…….…….5

Figure3. Plastic Fiber Optic Cable……..…….5

Figure4. Pin Diagram of DB-9 Connector……...….7

Figure5. Pin Diagram of DB-25 Connector…….....8

Figure6. MAX232 Level Converter IC……11

Figure7. Photo Diode……...13

Figure8. Parts of an LED……..……14

Figure9. Capacitor……15

Figure10. PNP Transistor (2N3904) ……...16

Figure11. Fiber Optic Link Design Block Diagram…….17

Figure12. Circuit Diagram of Receiver and Transmitter (transceiver) Section of PC to PC Communication using FIBER OPTIC……..19

Figure13. Graphical user interface……...……20

Figure14.GUI for main data transfer page……...21

Figure15 sample selection of settings by a user……..…….22

LIST OF TABLES

Table1: RS232 Voltage Level……10

Table2: Max-232 Conversion level……..…..11

ABBREVIATION

CATV.....CABLE TV

CMOS.....COMPLEMENTARY METAL OXIDE SEMICONDUTOR

ESP.....ENHANCED SERIAL PORT

GUI.....GRAPHICAL USER INTERFACE

LED.....LIGHT EMITTING DIODE

IR LED.....INFRARED LIGHT EMITTING DIODE

ILD.....INJECTION LASER DIODE

TTL.....TRANSISTOR TRANSISTOR LOGIC

UART.....UNIVERSAL ASYNCHRONUOUS RECEIVER TRANSMITTER

TXD.....TRASMIT DATA

RXD.....RECEIVE DATA

RS-232.....RECOMMENDED STANDARD 232

BACKGROUND

Data transfer is the need of today's world. Most of people can use internet for this. But presently it would be time consuming. One can implement data transfer using wireless medium. But then it will be a costly affair. The need, therefore, is felt for fiber optic communication which is cheaper and more suitable for the task. It is cheaper than wireless medium and is prone to lesser loss as compared to wireless medium.

Fiber optics has their own advantages; larger bandwidth, high speeds, immunity to noise, security only to name a few. Fiber optic transmission media offers a far more secure medium than copper-based or wireless technologies. Therefore, we need to use optic fibers for the internet and LANs. This project can be used to chat between two PCs. It can help transfer files at the desired speed between two PCs. If we expand further, a network of PCs can be connected as it is observed in the cable net available today.

The conventional LAN cables can be replaced by fiber optic cables which can provide much higher speeds and also larger information carrying capacity. To have a better understanding of what a PC to PC fiber optic communication is, it helps to understand what a fiber optic cable is, and its purpose in the communication world. A fiber optic cable transmits a signal using a wave of light, while copper wires transmit a signal using an electric current. A typical fiber optic wire consists of a plastic sheath surrounding a glass tube. Inside the tube is a tiny core in which the light waves travel. Separating the core and the tube is a cladding that prevents the light waves from escaping. One wire design has a core diameter of 125 microns and a sheath diameter of 2.5 millimeters.

CHAPTER-1

INTRODUCTION

1.1 CONCEPT USED

1.1.1 GUIDED OPTICAL COMMUNICATION

During the past years, a new technology has become increasingly accepted in the field of communication transmission via optical fiber cables. In contrast to copper cable technology, the signals are transmitted optically with the aid of optical waveguides also called optical fibers. They have many advantages, such as: high capacity, low cost per channel, light weight and small size. Stored data can easily be displayed from receiver when it is required .Guided optical communication is a telecommunication technology that uses hair thin strands of glass or plastic to Transmit communications signals (data) between two points. In optical fiber communication an energy beam is collimated and guided through an optical fiber rather than transmitting through space which is free space optics. Throughout the world, fiber optics is replacing copper wire as the primary carrier of information. A fiber-optic system is similar to the copper wire system that fiber-optics is replacing. The difference is that fiber-optics use light pulses to transmit information down fiber lines instead of using electronic pulses to transmit information down copper lines. Looking at the components in a fiber-optic chain will give a better understanding of how the system works in conjunction with wire based systems. At one end of the system is a transmitter. This is the place of origin for information coming on to fiber-optic lines. The transmitter accepts coded electronic pulse information coming from copper wire. It then processes and translates that information into equivalently coded light pulses.

1.1.2 APPLICATIONS

Fiber is already used in:

- Telephones: > 90% of all long distance telephony and > 50% of all local telephony
- Internet
- LANs - Most LAN (computer network) backbones
- CATV - for video, voice and Internet connections (Many video surveillance links)
- Utilities - management of power grid
- Security - closed-circuit TV and intrusion sensors
- Military - everywhere!

1.2 ABOUT FIBER OPTIC CABLE

1.2.1 General Information

A fiber-optic system is similar to the copper wire system that fiber-optics is replacing. The difference is that fiber-optics use light pulses to transmit information down fiber lines instead of using electronic pulses to transmit information down copper lines. Looking at the components in a fiber-optic chain will give a better understanding of how the system works in conjunction with wire based systems. At one end of the system is a transmitter. This is the place of origin for information coming on to fiber-optic lines. The transmitter accepts coded electronic pulse information coming from copper wire. It then processes and translates that information into equivalently coded light pulses. A light-emitting diode (LED) or an injection-laser diode (ILD) can be used for generating the light pulses. Using a lens, the light pulses are funneled into the fiber-optic medium where they transmit themselves down the line. Think of a fiber cable in terms of very long cardboard roll (from the inside roll of paper towel) that is coated with a mirror. If you shine a flashlight in one you can see light at the far end - even if bent the roll around a corner.

1.2.2 Why use a Fiber Optic cable?

Fiber optic systems transmit using infrared light, invisible to the human eye, because it goes further in the optical fiber at those wavelengths. The result is that fiber optic transmission media are the media of choice when it comes to “long haul” applications such as intercontinental, cross-continental and oceanic (marine) backbone links. It is also the preferred medium for tier one ISP backbone links. This means that new WAN implementations and applications are now predominantly fiber optic cable based. Wireless rollouts being the major exception. Fiber is the least expensive, most reliable method for high speed and/or long distance communications. While we already transmit signals at Gigabits per second speeds, we have only started to utilize the potential bandwidth of fiber.

- **Speed:** Fiber optic networks operate at high speeds - up into the gigabits
- **Bandwidth:** large carrying capacity
- **Distance:** Signals can be transmitted further without needing to be strengthened.
- **Resistance:** Greater resistance to electromagnetic noise such as radios, motors or other nearby cables.
- **Maintenance:** Fiber optic cables costs much less to maintain.in recent years it has become apparent that fiber-optics are steadily replacing copper wire as an appropriate means of communication signal transmission. They span the long distances between local phone systems as well as providing the backbone for many network systems. Other system users include cable television services, university campuses, office buildings, industrial plants, and electric utility companies.

CHAPTER-2

COMPONENT STUDY

2.1 FIBER OPTIC CABLE

A fiber optic cable is a cylindrical pipe. It may be made out of glass or plastic or a combination of glass and plastic. Basically, a fiber optic cable is composed of two concentric layers termed the core and the cladding. These are shown on the right side of Figure 1.

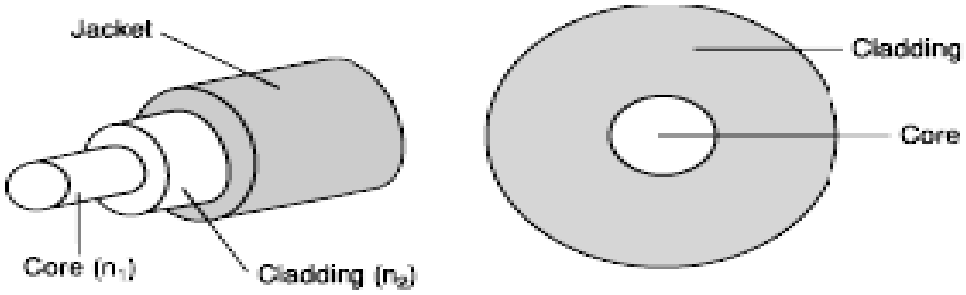


Figure1. Fiber optic section

The core and cladding have different indices of refraction with the core having n1 and the cladding n2. Light is piped through the core. A fiber optic cable has an additional coating around the cladding called the jacket. Core, cladding and jacket are all shown in the three dimensional view on the left side of Figure 1. The jacket usually consists of one or more layers of polymer. Its role is to protect the core and cladding from shocks that might affect their optical or physical properties. It acts as a shock absorber. The jacket also provides protection from abrasions, solvents and other contaminants. The jacket does not have any optical properties that might affect the propagation of light within the fiber optic cable. The illustration on the left side of Figure 1 is somewhat simplistic. In actuality, there may be a strength member added to the fiber optic cable so that it can be pulled during installation. (Figure 1).This would be added just inside the jacket. There may be a buffer between the strength member and the cladding. This protects the core and cladding from damage and allows the fiber optic cable to be bundled with other fiber optic cables.

How is light guided down in the fiber optic cable in the core?

This occurs because the core and cladding have different refraction index of the core, n1, always being greater than the index of the cladding, n2. Figure 2 shows how this is employed to effect the propagation of light down the fiber optic cable and confine it to the core.

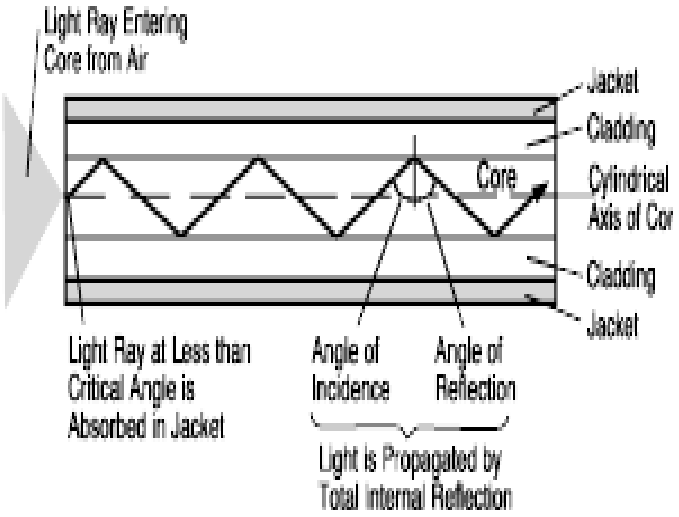


Figure2. Fiber Optic Light Propagation

As illustrated a light ray is injected into the fiber optic cable on the right. If the light ray is injected and strikes the core-to-cladding interface at an angle greater than an entity called the critical angle then it is reflected back into the core. Since the angle of incidence is always equal to the angle of reflection the reflected light will again be reflected. The light ray will then continue this bouncing path down the length of the fiber optic cable. If the light ray strikes the core-to-cladding interface at an angle less than the critical angle then it passes into the cladding where it is attenuated very rapidly with propagation distance. Light can be guided down the fiber optic cable if it enters at less than the **critical angle**.

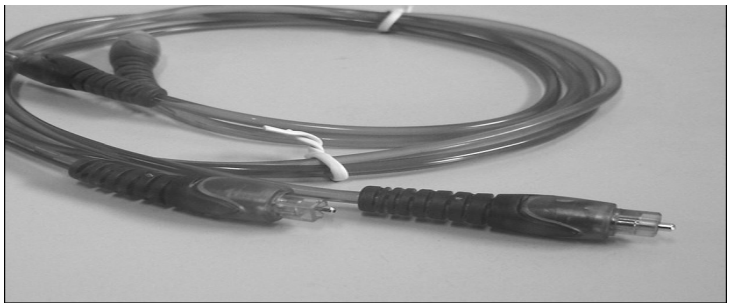


Figure3. Plastic Fiber Optic Cable

The cable used in our project has the following features:

- Length of 1 meter
- Diameter of approximately 3mm
- Type of Fiber material is plastic

2.2 SELECTION OF PC PORT

Why Serial Port?

1. Serial Cables can be longer than Parallel cables. The serial port transmits a '1' as -3 to -25 volts and a '0' as +3 to +25 volts where as a parallel port transmits a '0' as 0v and a'1' as 5v. Therefore the serial port can have a maximum swing of 50V compared to the parallel port which has a maximum swing of 5 Volts. Therefore cable loss is not going to be as much of a problem for serial cables as they are for parallel.

2. You don't need as many wires as parallel transmission. If your device needs to be mounted a far distance away from the computer then 3 core cable (Null Modem Configuration) is going to be a lot cheaper that running 19 or 25 core cable. However you must take into account the cost of the interfacing at each end.

3. Infra-Red devices have proven quite popular recently. You may have seen many electronic diaries and palmtop computers which have infra-red capabilities build in. However could you imagine transmitting 8 bits of data at the one time across the room being able to (from the devices point of view) decipher which bits are which? Therefore serial transmission is used where one bit is sent at a time. IrDA-1 was capable of 115.2k baud and was interfaced into a UART. The pulse length however was cut down to 3/16th of a RS232 bit length to conserve power considering these devices are mainly used on diaries, laptops and palmtops.

4. Microcontrollers have also proven to be quite popular recently. Many of these have in built SCI (Serial Communications Interfaces) which can be used to talk to the outside world. Serial Communication reduces the pin count of these MPU's. Only two pins are commonly used, Transmit Data (TXD) and Receive Data (RXD) compared with at least 8 pins if you use a 8 bit Parallel method (You may also require a Strobe).

The name "serial" comes from the fact that a serial port "serializes" data. That is, it takes a byte of data and transmits the 8 bits in the byte one at a time. The advantage is that a serial port needs only one wire to transmit the 8 bits (while a parallel port needs 8). The disadvantage is that it takes 8 times longer to transmit the data than it would if there were 8

wires. Serial ports lower cable costs and make cables smaller. Before each byte of data, a serial port sends a start bit, which is a single bit with a value of 0. After each byte of data, it sends a stop bit to signal that the byte is complete. It may also send a parity bit.

Serial ports, also called communication (COM) ports, are bi-directional. Bidirectional communication allows each device to receive data as well as transmit it. Serial devices use different pins to receive and transmit data -- using the same pins would limit communication to half-duplex, meaning that information could only travel in one direction at a time. Using different pins allows for full-duplex communication, in which information can travel in both directions at once. Serial ports rely on a special controller chip, the Universal Asynchronous Receiver/Transmitter (UART), to function properly. The UART chip takes the parallel output of the computer's system bus and transforms it into serial form for transmission through the serial port. In order to function faster, most UART chips have a built-in buffer of anywhere from 16 to 64 kilobytes. This buffer allows the chip to cache data coming in from the system bus while it is processing data going out to the serial port. While most standard serial ports have a maximum transfer rate of 115 Kbps (kilobits per second), high speed serial ports, such as Enhanced Serial Port (ESP) and Super Enhanced Serial Port (Super ESP), can reach data transfer rates of 460 Kbps.

2.3 CONNECTORS

We will be using the **9/25** pin connector for our project and its pin configuration is as follows:

2.3.1 DB-9 CONNECTOR (9 –Pin Connector)

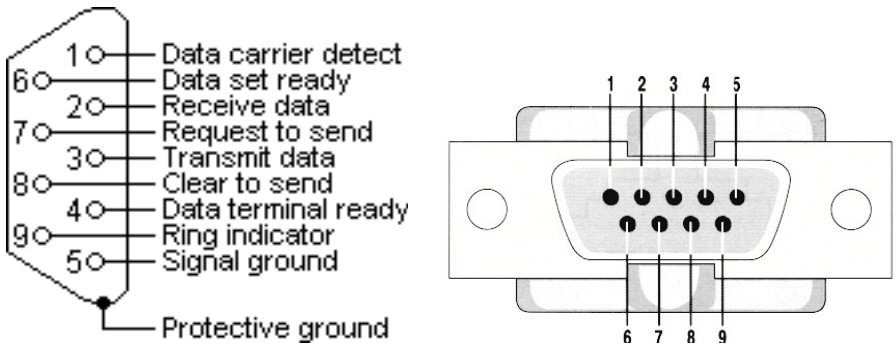


Figure4. Pin Diagram of DB-9 Connector

1. Data Carrier Detect - Determines if the modem is connected to a working phone line.

2. Receive Data - Computer receives information sent from the modem.

3. Transmit Data - Computer sends information to the modem.

4. Data Terminal Ready - Computer tells the modem that it is ready to talk.

5. Signal Ground - Pin is grounded.

6. Data Set Ready - Modem tells the computer that it is ready to talk.

7. Request To Send - Computer asks the modem if it can send information.

8. Clear To Send - Modem tells the computer that it can send information.

9. Ring Indicator - Once a call has been placed, computer acknowledges signal (sent from modem) that a ring is detected.

2.3.2 DB-25 CONNECTOR (25 –Pin Connector)

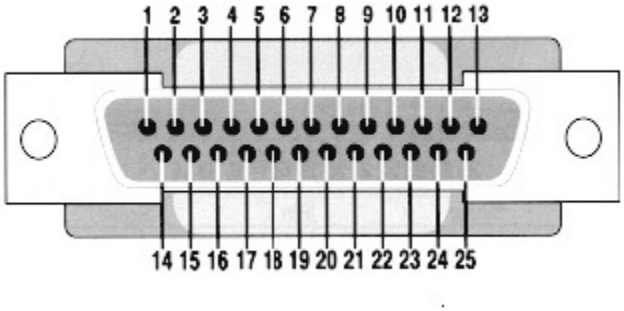


Figure5. Pin Diagram of DB-25 Connector

1. Not Used

2. Transmit Data - Computer sends information to the modem.

3. Receive Data - Computer receives information sent from the modem.

4. Request To Send - Computer asks the modem if it can send information.

5. Clear To Send - Modem tells the computer that it can send information.

6. Data Set Ready - Modem tells the computer that it is ready to talk.

7. Signal Ground - Pin is grounded.

8. Received Line Signal Detector - Determines if the modem is connected to a working phone line.

9. Not Used:

10. Not Used

11. Not Used

12. Not Used

13. Not Used

14. Not Used

15. Not Used

16. Not Used

17. Not Used

18. Not Used

19. Not Used

20. Data Terminal Ready - Computer tells the modem that it is ready to talk.

21. Not Used

22. Ring Indicator - Once a call has been placed, computer acknowledges signal (sent from modem) that a ring is detected.

23. Not Used

24. Not Used

25. Not Used

2.4 MAX232 CONVERTER IC

- ✓ **Applicability**

This module is primary of interest for people building their own electronics with an RS-232 interface. Off-the-shelf computers with RS-232 interfaces already contain the necessary electronics, and there is no need to add the circuitry as described here.

- ✓ **Logic Signal Voltage**

Almost all digital devices which we use require either TTL or CMOS logic levels. Therefore the first step to connecting a device to the RS-232 port is to transform the RS-232 levels back into 0 and 5 Volts.

Serial RS-232 (V.24) communication works with voltages (between -15V ... -3V are used to transmit a binary '1' and +3V ... +15V to transmit a binary '0') which are not compatible with today's computer logic voltages. On the other hand, classic TTL computer logic operates between 0V ... +5V (roughly 0V ... +0.8V referred to as low for binary '0', +2V ... +5V for high binary '1'). Modern low-power logic operates in the range of 0V ... +3.3V or even lower. So, the maximum RS-232 signal levels are far too high for today's computer logic electronics, and the negative RS-232 voltage can't be supported at all by the computer logic. Therefore, to receive serial data from an RS-232 interface the voltage has to be reduced, and the 0 and 1 voltage levels inverted. In the other direction (sending data from some logic over RS-232) the low logic voltage has to be "bumped up", and a negative voltage has to be generated, too.

```
   RS-232          TTL          Logic
-----------------------------------------------
 -15V ...  -3V <-> +2V ... +5V    <-> 1
  +3V ... +15V <->   0V ... +0.8V <-> 0
```

Table1. RS232 Voltage Level

All this can be done with conventional analog electronics, e.g. a particular power supply and a couple of transistors or the once popular 1488 (transmitter) and 1489 (receiver) ICs. However, since more than a decade it has become standard in amateur electronics to do the necessary signal level conversion with an integrated circuit (IC) from the MAX232 family (typically a MAX232A or some clone). In fact, it is hard to find some RS-232 circuitry in amateur electronics without a MAX232A or some clone.

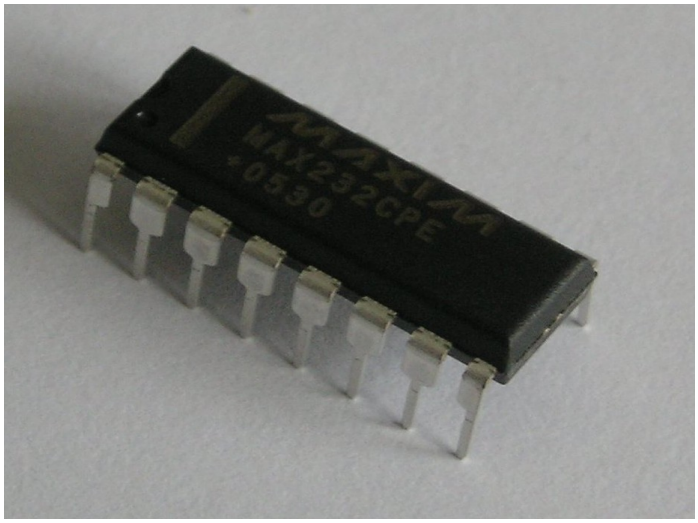


Figure6. MAX232 Level Converter IC

Max-232 Conversion level	
TTL+5	-9 RS232
TTL0V	+9 RS232
RS232+9v	0v TTL
RS232-9v	5v TTL

Table2.Max-232 Conversion level

2.5 PHOTODIODE

A **photodiode** is a type of photo detector capable of converting light in to either current or voltage, depending upon the mode of operation.

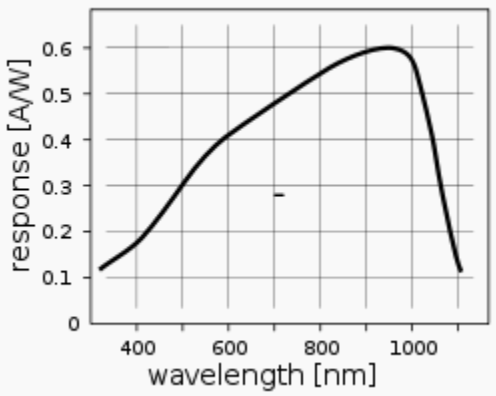
Photodiodes are similar to regular semiconductor diodes except that they may be either exposed (to detect vacuum UV or X-rays) or packaged with

a window or optical fiber connection to allow light to reach the sensitive part of the device. Many diodes designed for use specifically as a photodiode will also use a PIN junction rather than the typical PN junction.

Principle of operation

A photodiode is a PN junction or PIN structure. When a photon of sufficient energy strikes the diode, it excites an electron, thereby creating a mobile electron and a positively charged electron hole. If the absorption occurs in the junction's depletion region, or one diffusion length away from it, these carriers are swept from the junction by the built-in field of the depletion region. Thus holes move toward the anode, and electrons toward the cathode, and a photocurrent is produced.

Features



Graph1. Response of a silicon photo diode vs. wavelength of the incident light

Critical performance parameters of a photodiode include:

- **<u>responsivity</u>**
  - The ratio of generated photocurrent to incident light power, typically expressed in A/W when used in photoconductive mode. The responsivity may also be

expressed as quantum efficiency, or the ratio of the number of photo generated carriers to incident photons and thus a unit less quantity.

- **dark current**
  - The current through the photodiode in the absence of light, when it is operated in photoconductive mode. The dark current includes photocurrent generated by background radiation and the saturation current of the semiconductor junction. Dark current must be accounted for by calibration if a photodiode is used to make an accurate optical power measurement, and it is also a source of noise when a photodiode is used in an optical communication system.
- **noise-equivalent power**
  - (NEP) The minimum input optical power to generate photocurrent, equal to the rms noise current in a 1 hertz bandwidth. The related characteristic detectivity (D) is the inverse of NEP, 1/NEP; and the specific detectivity ($D^\star$) is the detectivity normalized to the area (A) of the photo detector, $D^\star = D\sqrt{A}$. The NEP is roughly the minimum detectable input power of a photodiode.

When a photodiode is used in an optical communication system, these parameters contribute to the sensitivity of the optical receiver, which is the minimum input power required for the receiver to achieve a specified bit error ratio.



Figure7. Photo Diode

2.6 LIGHT EMITTING DIODE (LED)

A **light-emitting diode** (**LED**) is a semiconductor light source. LEDs are used as indicator lamps in many devices, and are increasingly used for lighting. Introduced as a practical electronic component in 1962, early LEDs emitted low-intensity red light, but modern versions are available across the visible, ultraviolet and infrared wavelengths, with very high brightness.

The LED is based on the semiconductor diode. When a diode is forward biased (switched on),electrons are able to recombine with holes within the device, releasing energy in the form of photons. This effect is called electroluminescence and the color of the light (corresponding to the energy of the photon) is determined by the energy gap of the semiconductor. An LED is usually small in area (less than 1 mm^2), and integrated optical components are used to shape its radiation pattern and assist in reflection. LEDs present many advantages over incandescent light sources including lower energy consumption, longer lifetime, improved robustness, smaller size, faster switching, and greater durability and reliability. However, they are relatively expensive and require more precise current and heat management than traditional light sources. Current LED products for general lighting are more expensive to buy than fluorescent lamp sources of comparable output.

They also enjoy use in applications as diverse as replacements for traditional light sources in automotive (particularly indicators) and in traffic signals. Airbus has used LED lighting in their A320 Enhanced since 2007, and Boeing plans its use in the 787. The compact size of LEDs has allowed new text and video displays and sensors to be

developed, while their high switching rates are useful in advanced communications technology

Technology

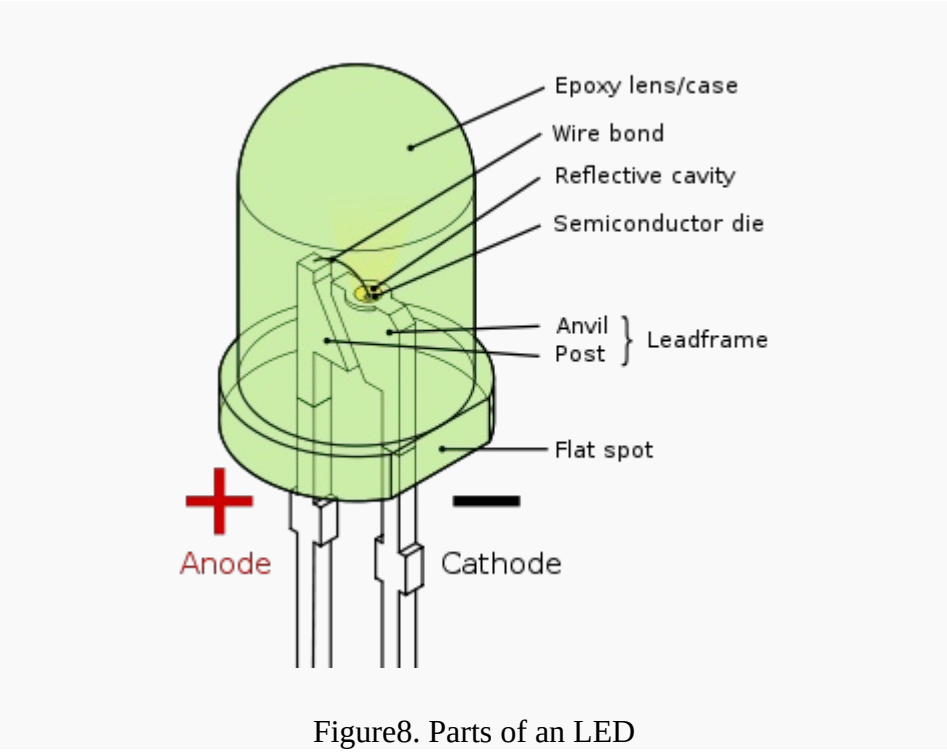


Figure8. Parts of an LED

Like a normal diode, the LED consists of a chip of semiconducting material doped with impurities to create a p-n junction. As in other diodes, current flows easily from the p-side, or anode, to the n-side, or cathode, but not in the reverse direction. Charge-carriers—electrons and holes—flow into the junction from electrodes with different voltages. When an electron meets a hole, it falls into a lower energy level, and releases energy in the form of a photon.

The wavelength of the light emitted, and therefore its color, depends on the band gap energy of the materials forming the p-n junction.
In silicon or germanium diodes, the electrons and holes recombine by a non-radiative transition which produces no optical emission, because these are indirect band gap materials. The materials used for the LED have a direct band gap with energies corresponding to near-infrared, visible or near-ultraviolet light.

LED development began with infrared and red devices made with gallium arsenide. Advances in materials science have made possible the production of devices with ever-shorter wavelengths, producing light in a variety of colors.

LEDs are usually built on an n-type substrate, with an electrode attached to the p-type layer deposited on its surface. P-type substrates, while less common, occur as well. Many commercial LEDs, especially GaN/InGaN, also use sapphire substrate.

Most materials used for LED production have very high refractive indices. This means that much light will be reflected back into the material at the material/air surface interface. Therefore Light extraction in LEDs is an important aspect of LED production, subject to much research and development.

2.7 CAPACITOR

The capacitor plays a crucial role in electronics -- it stores electrons for when they're needed most. Capacitors consist of two conducting plates placed near each other. Inside the capacitor, the terminals connect to two metal plates separated by a dielectric. The dielectric can be air, paper, plastic or anything else that does not conduct electricity and keeps the plates from touching each other.



Figure9. Capacitor

They can store electric charge for later discharge. Direct current through a capacitor will charge the capacitor for a short time, and then stop flowing. Alternating current, because of the changing electric fields it generates, can “flow” across a capacitor.

2.8 TRANSISTOR (PNP 2N3906)

FEATURES:

- Low switching noise
- Low forward voltage drop
- Low thermal resistance
- High switching capability
- High surge capability
- High reliability

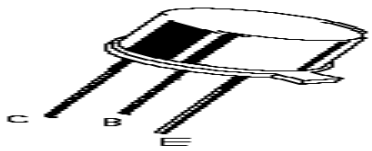


Figure10. PNP Transistor (2N3904)

DEVICE SPECIFICATION	
TYPE	2N3906
POLARITY	PNP
APPLICATION	General Purpose Medium

Table3. PNP Transistor (2N3904) Specification

CHAPTER-3

PROJECT DESCRIPTION

3.1 HARDWARE

3.1.1 Overall System Block Diagram

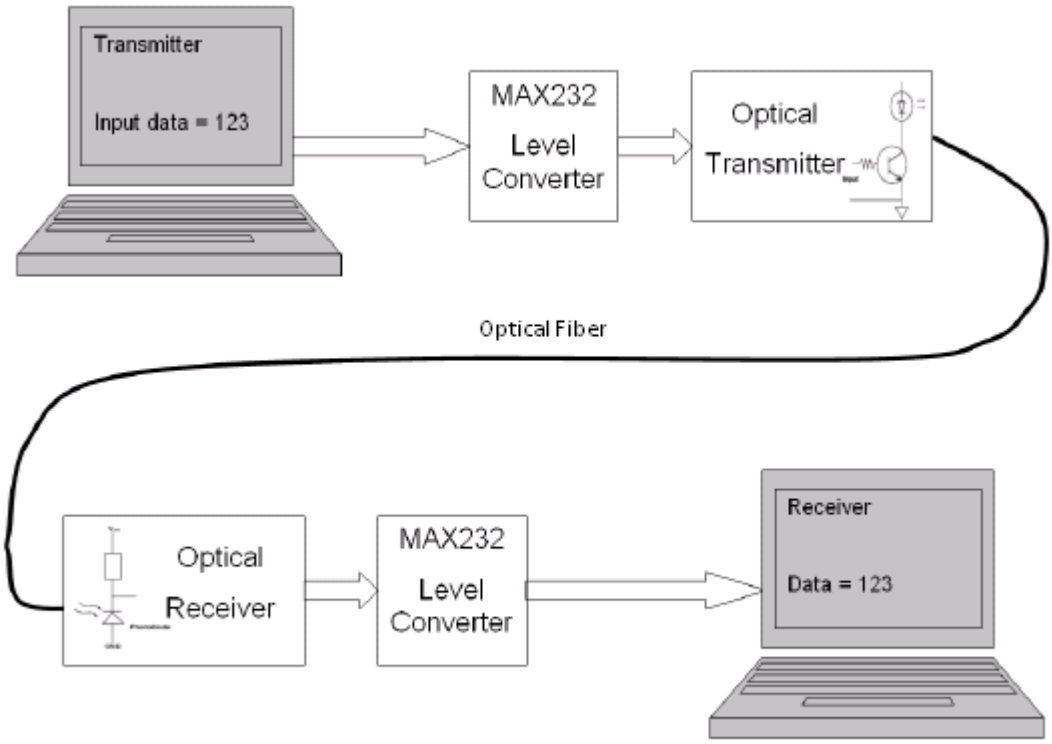


Figure11. Fiber Optic Link Design Block Diagram

There are two sections in our project. One is the transmitter section and the other being the receiver section.

3.1.2 TRANSMITTER SECTION

Data signals transmitted through pin 3 of 9-pin ‘D’ connector of RS-232 COM port are sent to pin 8 of MAX232 and it converts these RS-232 compatible levels of +9V or -9V to 0/5 volt TTL levels as given in table.

The output of MAX232 drives the PNP transistor through a bias resistor of 1k ohm, to switch on “IR LEDS” and also visible LED. Here actually when the output of MAX232 is 0V at that time the PNP transistor will be conduct and the IRLED as well visible LED will glow. And when the output of MAX232 is 5V at that time the PNP transistor will not conduct it is in cut-off region. So, at that time the IRLED as well the visible LED will not glow.

We can use laser diodes and LEDs for the transmitter. Both have their own advantages and disadvantages. But LEDs are more reliable and also cheap, so we will be using a LED. There are several different schemes for carrying out the modulation function. These are respectively: Intensity Modulation, Frequency Shift Keying, Phase Shift Keying and Polarization Modulation. Within the context of a premise fiber optic data link the only one really employed is Intensity Modulation.

It must be noted that one reason for the popularity of Intensity Modulation is its suitability for operation with LED's. An LED can only produce incoherent optical power. Since Intensity Modulation does not require coherence it can be used with an LED.

3.1.3 RECEIVER SECTION

The IR signals are detected by a photo diode (D1).A Photo Diode is reverse biased & breaks down when IR light falls on its junction .The detected TTL level (0/5V)signals are coupled to pin10 of MAX 232 IC .These TTL levels are converted to+9Vor -9V levels internally(as per Table-1) &output at pin 7.

A visible LED1 at pin 7 of MAX232 IC indicates that the signals are being received. Pin 7 is also connected to pin 2 of pin 9 ‘D’ connector used for the serial port in the PC, so that the data may be read .The optical signals received by the photodiodes are in fact converted to

electrical pulses and both the pcs “think” that there is null modem cable connected between them.

The Receiver component serves two functions. First, it must sense or detect the light coupled out of the fiber optic cable then convert the light into an electrical signal. Secondly, it must demodulate this light to determine the identity of the binary data that it represents. In total, it must detect light and then measure the relevant Information bearing light wave parameters. A Receiver is generally designed with a Transmitter. The very heart of the Receiver is the means for sensing the light output of the fiber optic cable. Light is detected and then converted to an electrical signal. The demodulation decision process is carried out on the resulting electrical signal. The light detection is carried out by a photodiode. This senses light and converts it into an electrical current.

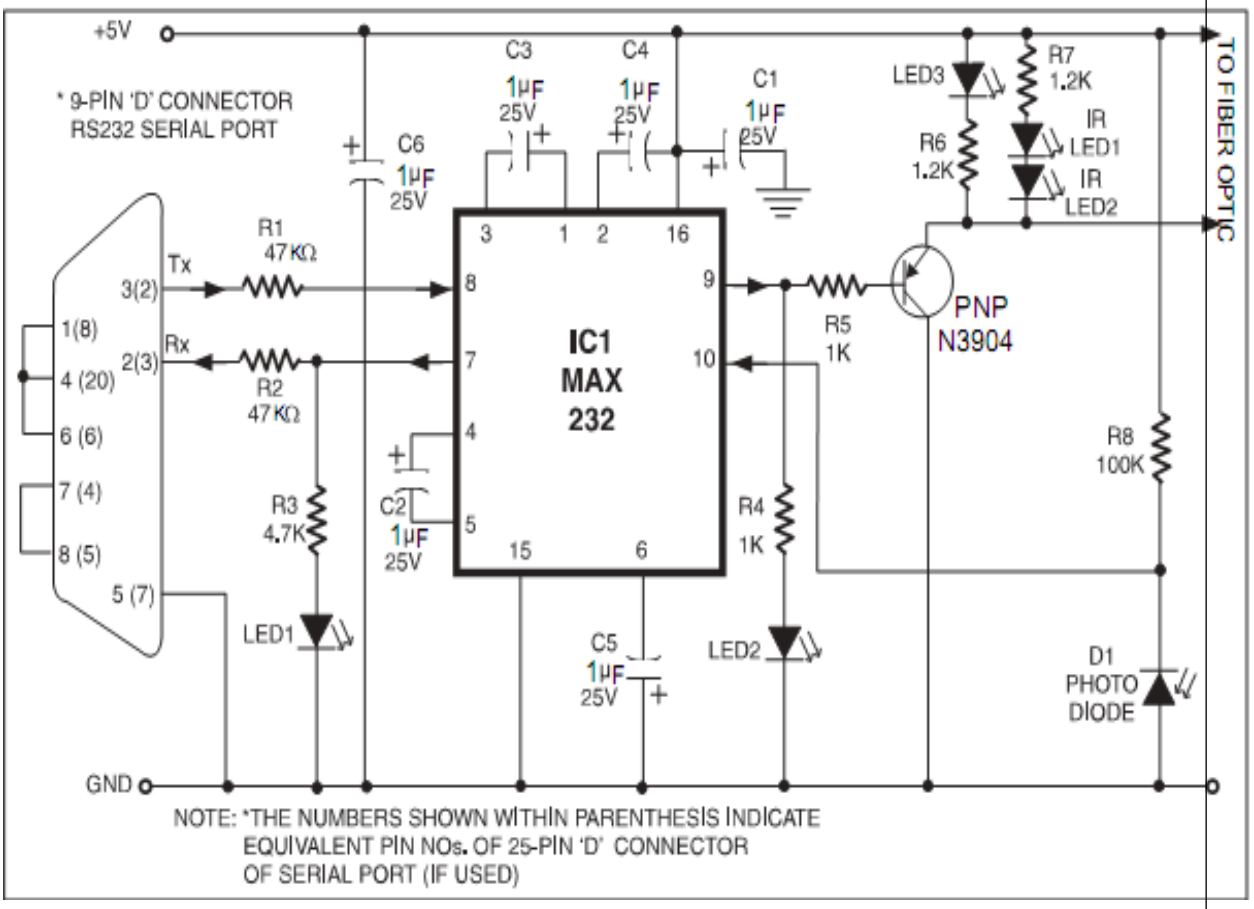


Figure12. Circuit Diagram of Receiver and Transmitter (transceiver) Section of PC to PC Communication using FIBER OPTIC

3.2 SOFTWARE

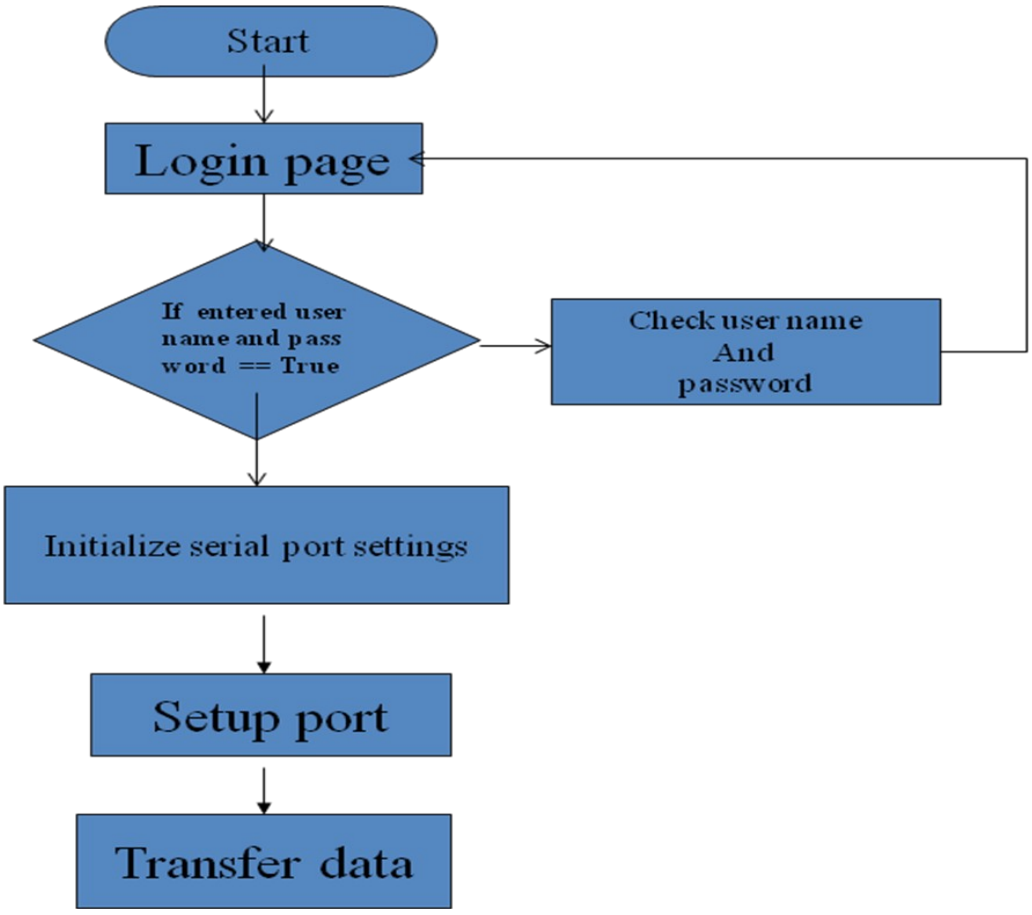
In serial communication, a byte of data is transferred through a single wire one bit at a time. The packets contain a start bit, data, and stop bit. Once the start bit has been sent, the transmitter sends the actual data bits. There may either be 5, 6, 7, or 8 data bits, depending on the number you have selected. Both receiver and the transmitter must agree on the number of data bits, as well as the baud rate.

The software used to test this hardware, can transmit digits from one PC to the other. It programs the serial port control register to adjust the speed. When both the PCs are matched in speed, the transmitter program sends the digits continuously with the start and stop bits. At the receiver, the program scans the status register to check if the data has arrived.

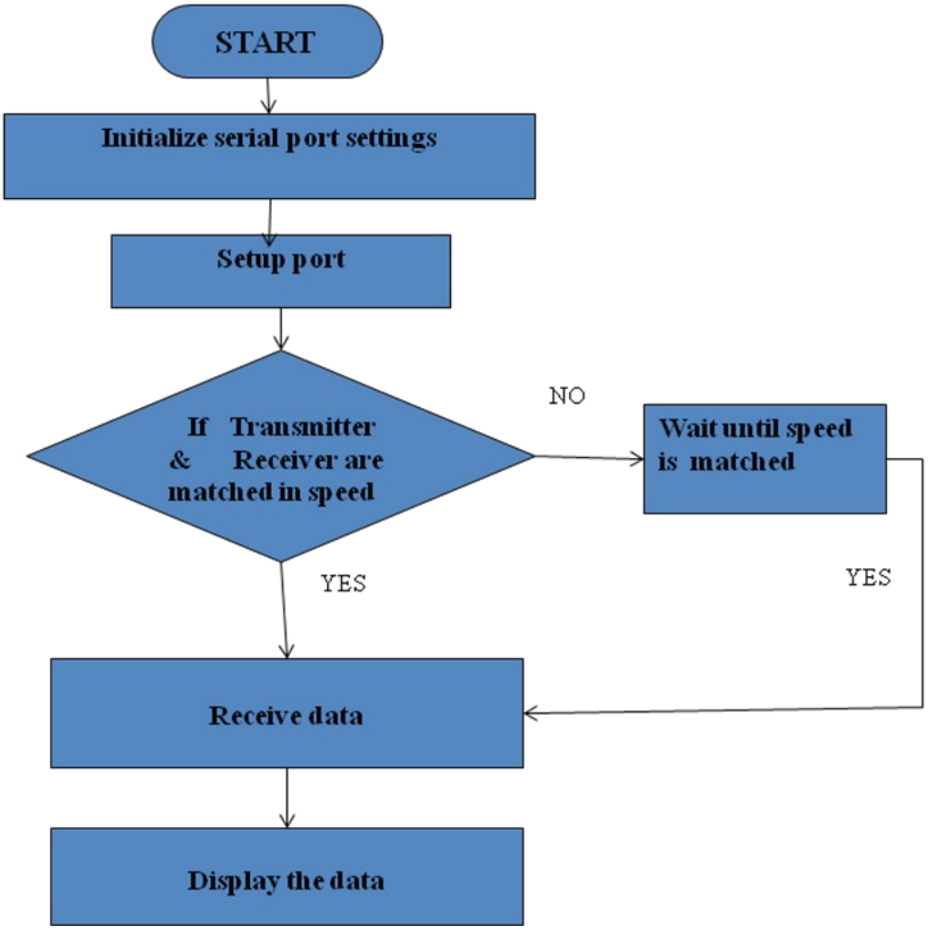
As soon as the start bit is received the data are read and displayed. This program can be expanded to support a two way link. It can also be used to transmit the data as well as receive it.

3.2.1 PROGRAM FLOW CHART

Transmit Data



Receive Data


START
Initialize serial port settings
Setup port
If Transmitter & Receiver are matched in speed
NO
Wait until speed is matched
YES
YES
Receive data
Display the data

3.2.2 GRAPHICAL USER INTERFACE(GUI)

The GUI (Graphical User Interface) is the main program and was written in C#. It calls the C# programs, provides user control. Upon successful return from configuration, the GUI waits for user input. The user enters an input using the mouse or keyboard.

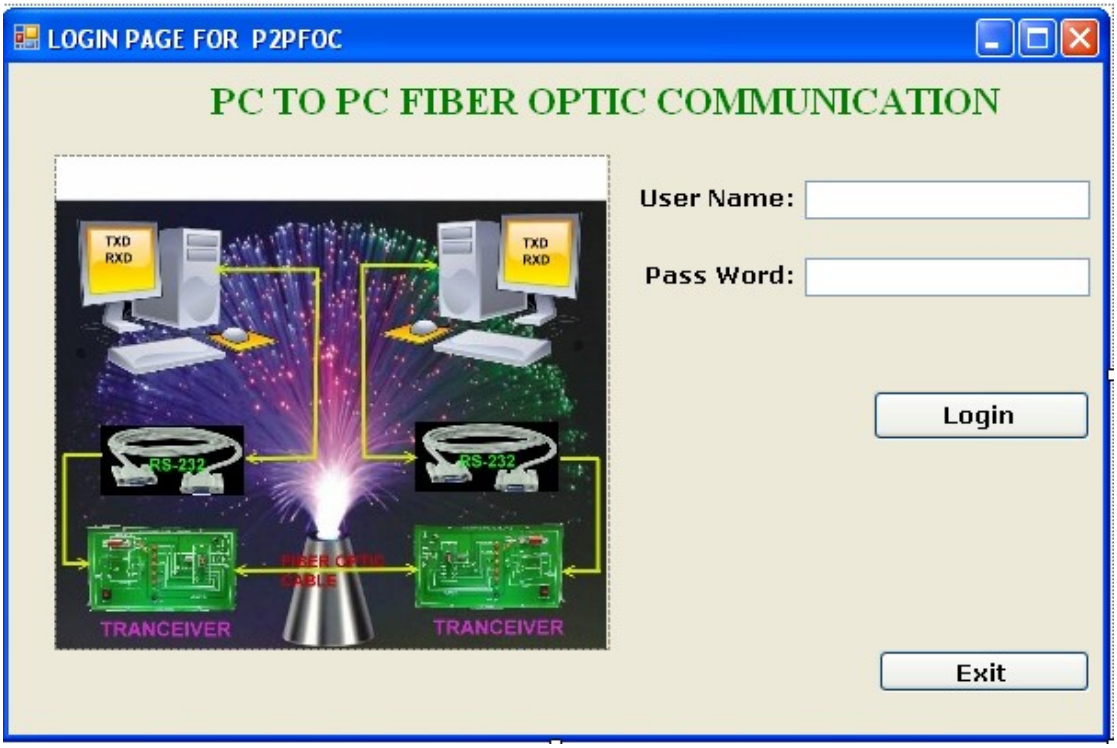


Figure13. Graphical user interface

This an introductory login page for security of the main page. Here, if a valid user name and password is input and along pressing the log in button, the main page will be displayed, hence access is granted, if wrong password or user name is used access will not be granted.

The main data transfer page is shown below in figure12.

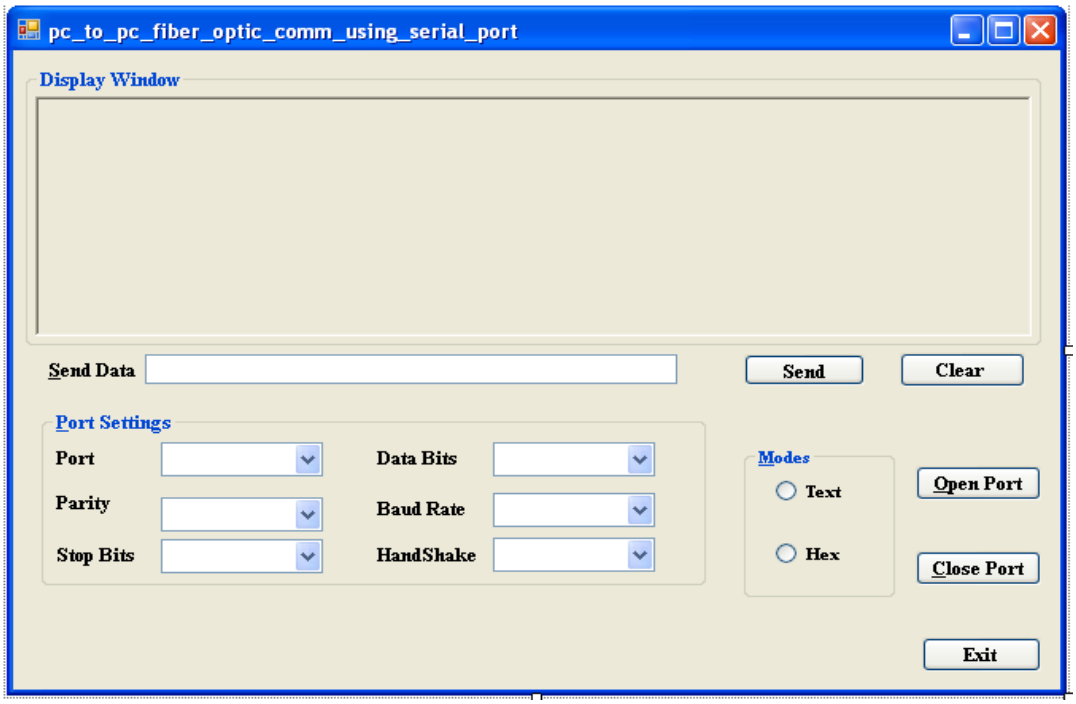


Figure14.GUI for main data transfer page.

For communication to take place, first open port (basically this port is serial) button should pressed. Since the two PCs are needed to communicate the same activity is done in both of them. As soon as the serial port is open, the two pcs are connected by the program. Figure12 shows several inputs the user can select: COM port, set parity bit, stop bit, data bits, baud rate, handshake from the port settings and chose text or hex-decimal mode.

The user uses the send button to load data and the clear button clears what is already in the display area. The exit button will close the main window displayed

The figure13 below shows sample window chosen by random user with his typical setting selections.

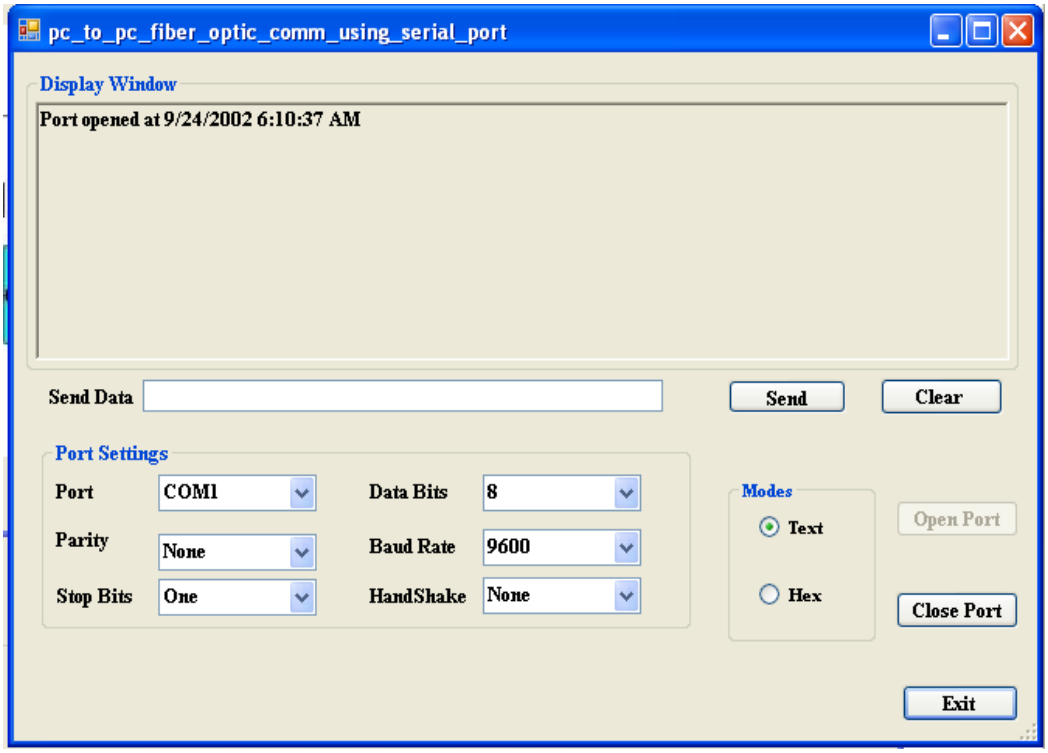


Figure15. Sample selection of settings by a user

CHAPTER-4

CONCLUSION AND RECOMMENDATION

4.1 CONCLUSION

Fiber optic transmission has found a vast array of applications in computer systems. As we move towards a more sophisticated and modern future, the uses of fiber optics are going to grow in all computer systems as well as telecommunication networks. The ease of maintenance the fiber cables have, Low attenuation and superior signal integrity make it to be the ultimate choice for communication.

In this project we have aimed to transfer data between two PCs using the fiber optic cable as a medium. Starting with a simple circuit diagram model we have gradually added more complex feature to make our model as close to real life as possible. Additionally, we have produced the software that is needed to drive the system. Our team has managed to communicate PCs via fiber optic cable. Basically the transfer of data can be done by copper, wireless or internet. But Fiber is the communications medium of choice, since its greater distance and bandwidth capabilities make it either the only choice or considerably lessexpensive than copper or wireless. Only inside buildings is there a choice to be made, and that choice is affected by economics, network architecture and the tradition of using copper inside buildings. The project we have designed shows how fiber optic communication is well suited as large amounts of information can be transported at one time with a very small core of fiber optic cable and with a most secure form of communication. An optic wire cannot be tapped into like a copper wire can. This creates an extremely secure way to transfer data.

4.2 RECOMMENDATION FOR THE FUTURE WORK

During the whole process of developing the system, there are several skills that we successfully managed to learn or improve it. When planning the functionality of the system, we had experienced the whole process of software development life cycle. Developing this system greatly improves our programming skills in visual studio C# language. During the development process, there were a lot of problems in the programming part, but with persistent research in the internet and reading books, it had greatly improved our knowledge in programming skills. After having the system built, we need to deploy the system so that the

system really works in real life. During the deployment period, we had learned how to interface and integrate the whole system.

Related with present work, a number of areas deserving further attention to be explored have been identified as follows:

- This project can be extended to by modifying the transceiver circuit for using the system for LANs and other security demanding systems.
- The software can be enhanced with features that support internet connections and secure internet access will be afforded as a result.

Appendix I: Code for the Login page

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using pc_to_pc_fiber_optic_communication;


namespace pc_to_pc_fiber_optic_communication
{
    public partial class loginpage : Form
    {
        pc_to_pc_fiber_optic_comm_using_serial_port con =
new pc_to_pc_fiber_optic_comm_using_serial_port();
        public loginpage()
        {
            InitializeComponent();
        }


        private void btnlogin_Click(object sender,
EventArgs e)
```

```
        {
            if (txtusername.Text == "birhanu tesfay" &&
txtpassword.Text == "abc")
            {
                con.ShowDialog();
                this.Hide();
            }
            else if (txtusername.Text == "ephrem hailu" &&
txtpassword.Text == "abc")
            {
                con.ShowDialog();
                this.Hide();
            }
            else if (txtusername.Text == "goitom yisfa" &&
txtpassword.Text == "abc")
            {
                con.ShowDialog();
                this.Hide();
            }
            else if (txtusername.Text == "haftamu g/her" &&
txtpassword.Text == "abc")
            {
                con.ShowDialog();
                this.Hide();
            }
            else
            {
                MessageBox.Show("Check your Username or
Password");
                txtpassword.Text = "";
                txtusername.Text = "";
            }
```

```
        }
    }
}
```

Appendix II: Code for the Main Page

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;


namespace pc_to_pc_fiber_optic_communication
{
    public partial class
pc_to_pc_fiber_optic_comm_using_serial_port : Form
    {
        comm_man comm = new comm_man();
        string transType = string.Empty;


        public
pc_to_pc_fiber_optic_comm_using_serial_port()
        {
            InitializeComponent();
        }
```

```
        private void
pc_to_pc_fiber_optic_comm_using_serial_port_Load(object
sender, EventArgs e)
        {
            LoadValues();
            SetDefaults();
            SetControlState();
        }

        private void btnopen_Click(object sender, EventArgs
e)
        {
            comm.Parity = cboParity.Text;
            comm.StopBits = cboStop.Text;
            comm.DataBits = cboData.Text;
            comm.BaudRate = cboBaud.Text;
            comm.HandShake = cbohand.Text;
            comm.DisplayWindow = rtbDisplay;
            comm.OpenPort();
            txtsend.Enabled = true;
            btnopen.Enabled = false;
            btnclose.Enabled = true;
            btnsend.Enabled = true;
            btnclear.Enabled = true;
        }

        // Method to initialize serial port
        // values to standard defaults
        private void SetDefaults()
        {
            cboPort.SelectedIndex = 0;
            cboBaud.SelectedText = "9600";
```

```
            cboParity.SelectedIndex = 0;
            cbohand.SelectedIndex = 0;
            cboStop.SelectedIndex = 1;
            cboData.SelectedText = "8";
        }
        // methos to load our serial
        // port option values

        private void LoadValues()
        {
            comm.SetPortNameValues(cboPort);
            comm.SetParityValues(cboParity);
            comm.SetStopBitValues(cboStop);
            comm.SetHandShakeValues(cbohand);
            }
        // method to set the state of controls
        // when the form first loads

        private void SetControlState()
        {
            rdoText.Checked = true;
            btnsend.Enabled = false;
            btnclose.Enabled = false;
            btnclear.Enabled = false;
        }

        private void btnsend_Click(object sender, EventArgs
e)
        {
            comm.WriteData(txtsend.Text);
            txtsend.Clear();
```

```
        }


        private void rdoHex_CheckedChanged(object sender,
EventArgs e)
        {
            if (rdoHex.Checked == true)
            {
                comm.CurrentTransmissionType =
pc_to_pc_fiber_optic_communication.comm_man.TransmissionTyp
e.Hex;


            }
            else
            {
                comm.CurrentTransmissionType =
pc_to_pc_fiber_optic_communication.comm_man.TransmissionTyp
e.Text;
            }
        }


        private void btnclear_Click(object sender,
EventArgs e)
        {
            rtbDisplay.Clear();
        }


        private void btnclose_Click(object sender,
EventArgs e)
        {
            btnclose.Enabled = false;
            btnopen.Enabled = true;
            btnsend.Enabled = false;
            btnclear.Enabled = false;
```

```
            txtsend.Enabled = false;
        }

        private void btnexit_Click(object sender, EventArgs
e)
        {
            Dispose();
        }

    }
}
```

Appendix III: Code for the Communication Manager

```
using System;
using System.Text;
using System.Drawing;
using System.IO.Ports;
using System.Windows.Forms;

namespace pc_to_pc_fiber_optic_communication
{
    class comm_man
    {
        #region Enums

        // enumeration to hold our transmission types

        public enum TransmissionType { Text, Hex }
```

```
        // enumeration to hold our message types

        public enum MessageType { Incoming, Outgoing,
Normal, Warning, Error };
        #endregion

        #region Variables
        //property variables
        private string baudRate = string.Empty;
        private string parity = string.Empty;
        private string stopBits = string.Empty;
        private string dataBits = string.Empty;
        private string portName = string.Empty;
        private string handShake = string.Empty;
        private TransmissionType transType;
        private RichTextBox displayWindow;
        //global manager variables
        private Color[] MessageColor = { Color.Blue,
Color.Green, Color.Black, Color.Orange, Color.Red };
        private SerialPort comPort = new SerialPort();
        #endregion

        #region Properties

        // Property to hold the BaudRate
        // of our manager class


        public string BaudRate
```

```
{
    get { return baudRate; }
    set { baudRate = value; }
}


// property to hold the Parity
// of our manager class

public string Parity
{
    get { return parity; }
    set { parity = value; }
}


// property to hold the StopBits
// of our manager class

public string StopBits
{
    get { return stopBits; }
    set { stopBits = value; }
}


// property to hold the DataBits
// of our manager class

public string DataBits
```

```
{
    get { return dataBits; }
    set { dataBits = value; }
}


// property to hold the PortName
// of our manager class

public string PortName
{
    get { return portName; }
    set { portName = value; }
}


// property to hold the HandShake
// of our manager class

public string HandShake
{
    get { return handShake; }
    set { handShake = value; }
}

// property to hold our TransmissionType
// of our manager class

public TransmissionType CurrentTransmissionType
{
```

```
            get { return transType; }
            set { transType = value; }
        }


        // property to hold our display window
        // value

        public RichTextBox DisplayWindow
        {
            get { return displayWindow; }
            set { displayWindow = value; }
        }
        #endregion

        #region Constructors

        // Constructor to set the properties of our Manager
Class



        public comm_man(string baud, string par, string
sBits, string dBits, string name,string hand, RichTextBox
rtb)
        {
            baudRate = baud;//Desired BaudRate
            parity = par;//Desired Parity
            stopBits = sBits;//Desired StopBits
            dataBits = dBits;//Desired DataBits
            portName = name;//Desired PortName
            handShake = hand;  //Desired HAndShake
            displayWindow = rtb;
```

```
            //now add an event handler
            comPort.DataReceived += new
SerialDataReceivedEventHandler(comPort_DataReceived);
        }

        // Comstructor to set the properties of our
        // serial port communicator to nothing

        public comm_man()
        {
            baudRate = string.Empty;
            parity = string.Empty;
            stopBits = string.Empty;
            dataBits = string.Empty;
            handShake = string.Empty;
            portName = "COM1";
            displayWindow = null;
            //add event handler
            comPort.DataReceived += new
SerialDataReceivedEventHandler(comPort_DataReceived);
        }
        #endregion

        #region WriteData
        public void WriteData(string msg)
        {
            switch (CurrentTransmissionType)
            {
                case TransmissionType.Text:
                    //first make sure the port is open
```

```
                    //if its not open then open it
                    if (!(comPort.IsOpen == true))
                        comPort.Open();
                    //send the message to the port
                    comPort.Write(msg);
                    //display the message
                    DisplayData(MessageType.Outgoing, msg
+ "\n");
                    break;
                case TransmissionType.Hex:
                    try
                    {
                        //convert the message to byte array
                        byte[] newMsg = HexToByte(msg);
                        //send the message to the port
                        comPort.Write(newMsg, 0,
newMsg.Length);
                        //convert back to hex and display
                        DisplayData(MessageType.Outgoing,
ByteToHex(newMsg)  + "\n");
                    }
                    catch (FormatException ex)
                    {
                        //display error message
                        DisplayData(MessageType.Error,
ex.Message);
                    }
                    finally
                    {
                        displayWindow.SelectAll();
                    }
                    break;
```

```
                default:
                    //first make sure the port is open
                    //if its not open then open it
                    if (!(comPort.IsOpen == true))
                        comPort.Open();
                    //send the message to the port
                    comPort.Write(msg);
                    //display the message
                    DisplayData(MessageType.Outgoing, msg
+ "\n");
                    break;

            }
        }
        #endregion

        #region HexToByte

        // method to convert hex string into a byte array
        //string to convert a byte array
        private byte[] HexToByte(string msg)
        {
            //remove any spaces from the string
            msg = msg.Replace(" ", "");
            //create a byte array the length of the
            //divided by 2 (Hex is 2 characters in length)
            byte[] comBuffer = new byte[msg.Length / 2];
            //loop through the length of the provided
string
            for (int i = 0; i < msg.Length; i += 2)
                //convert each set of 2 characters to a
byte
```

```
                //and add to the array
                comBuffer[i / 2] =
(byte)Convert.ToByte(msg.Substring(i, 2), 16);
            //return the array
            return comBuffer;
        }
        #endregion

        #region ByteToHex

        // method to convert a byte array into a hex string
        //byte array to convert</param>
        //a hex string</returns>
        private string ByteToHex(byte[] comByte)
        {
            //create a new StringBuilder object
            StringBuilder builder = new
StringBuilder(comByte.Length * 3);
            //loop through each byte in the array
            foreach (byte data in comByte)
                //convert the byte to a string and add to
the stringbuilder
                builder.Append(Convert.ToString(data,
16).PadLeft(2, '0').PadRight(3, ' '));
            //return the converted value
            return builder.ToString().ToUpper();
        }
        #endregion

        #region DisplayData

        // method to display the data to & from the port
```

```
        // on the screen
        //MessageType of the message
        //Message to display
        [STAThread]
        private void DisplayData(MessageType type, string
msg)
        {
            displayWindow.Invoke(new EventHandler(delegate
        {
            displayWindow.SelectedText = string.Empty;
            displayWindow.SelectionFont = new
Font(displayWindow.SelectionFont, FontStyle.Bold);
            displayWindow.SelectionColor =
MessageColor[(int)type];
            displayWindow.AppendText(msg);
            displayWindow.ScrollToCaret();
        }));
        }
        #endregion

        #region OpenPort
        public bool OpenPort()
        {
            try
            {
                //first check if the port is already open
                //if its open then close it
                if (comPort.IsOpen == true)
comPort.Close();

                //set the properties of our SerialPort
Object
```

```
                comPort.BaudRate = int.Parse(baudRate);
//BaudRate

                comPort.DataBits = int.Parse(dataBits);
//DataBits

                comPort.StopBits =
(StopBits)Enum.Parse(typeof(StopBits), stopBits);
//StopBits

                comPort.Parity =
(Parity)Enum.Parse(typeof(Parity), parity);    //Parity

                comPort.Handshake =
(Handshake)Enum.Parse(typeof(Handshake), handShake);

                comPort.PortName = portName;   //PortName

                //now open the port

                comPort.Open();

                //display message

                DisplayData(MessageType.Normal,"Port opened
at " + DateTime.Now + "\n");

                //return true

                return true;

            }

            catch (Exception ex)

            {

                DisplayData(MessageType.Error, ex.Message);

                return false;

            }

        }

        #endregion


        #region SetParityValues

        public void SetParityValues(object obj)

        {

            foreach (string str in
Enum.GetNames(typeof(Parity)))

            {
```

```
                ((ComboBox)obj).Items.Add(str);
            }
        }
        #endregion

        #region SetStopBitValues
        public void SetStopBitValues(object obj)
        {
            foreach (string str in
Enum.GetNames(typeof(StopBits)))
            {
                ((ComboBox)obj).Items.Add(str);
            }
        }
        #endregion

        #region SetPortNameValues
        public void SetPortNameValues(object obj)
        {

            foreach (string str in
SerialPort.GetPortNames())
            {
                ((ComboBox)obj).Items.Add(str);
            }
        }
        #endregion

        #region SetHandShakeValues
        public void SetHandShakeValues(object obj)
        {
```

```
            foreach (string str in
Enum.GetNames(typeof(Handshake)))
            {
                ((ComboBox)obj).Items.Add(str);
            }
        }
        #endregion

        #region comPort_DataReceived

        // method that will be called when there's data
waiting in the buffer

        void comPort_DataReceived(object sender,
SerialDataReceivedEventArgs e)
        {
            //determine the mode the user selected
(binary/string)
            switch (CurrentTransmissionType)
            {
                //user chose string
                case TransmissionType.Text:
                    //read data waiting in the buffer
                    string msg = comPort.ReadExisting();
                    //display the data to the user
                    DisplayData(MessageType.Incoming, msg
+ "\n");

                    break;
                //user chose binary
                case TransmissionType.Hex:
                    //retrieve number of bytes in the
buffer
```

```
                    int bytes = comPort.BytesToRead;
                    //create a byte array to hold the
awaiting data
                    byte[] comBuffer = new byte[bytes];
                    //read the data and store it
                    comPort.Read(comBuffer, 0, bytes);
                    //display the data to the user
                    DisplayData(MessageType.Incoming,
ByteToHex(comBuffer)  + "\n");
                    break;
                default:
                    //read data waiting in the buffer
                    string str = comPort.ReadExisting();
                    //display the data to the user
                    DisplayData(MessageType.Incoming, str +
"\n");
                    break;
            }
        }
        #endregion


    }
}
```

Appendix IV: PC TO PC CHATING IN CONSOLE APPLICATION

```
using System;
```

```
using System.IO.Ports;
using System.Threading;
namespace pc2pc_fiber_optic_communication
{
    class Program
    {
        static bool Continue;
        static SerialPort serialPort;
        static void Main(string[] args)
        {
            string UserName;
            string message;
            StringComparer stringComparer =
StringComparer.OrdinalIgnoreCase;
            Thread readThread = new Thread(Read);


            // Create a new SerialPort object with default
settings.
            serialPort = new SerialPort();


            // Allow the user to set the appropriate
properties.
            serialPort.PortName =
SetPortName(serialPort.PortName);
            serialPort.BaudRate =
SetPortBaudRate(serialPort.BaudRate);
            serialPort.Parity =
SetPortParity(serialPort.Parity);
            serialPort.DataBits =
SetPortDataBits(serialPort.DataBits);
            serialPort.StopBits =
SetPortStopBits(serialPort.StopBits);
            serialPort.Handshake =
SetPortHandshake(serialPort.Handshake);
```

```
            // Set the read/write timeouts
            serialPort.ReadTimeout = 500;
            serialPort.WriteTimeout = 500;

            serialPort.Open();
            Continue = true;
            readThread.Start();

            Console.Write("Name: ");
            UserName = Console.ReadLine();

            Console.WriteLine("TYPE quit TO EXIT");

            while (Continue)
            {
                message = Console.ReadLine();

                if (stringComparer.Equals("quit", message))
                {
                    Continue = false;
                }
                else
                {
                    serialPort.WriteLine(
                        String.Format("<{0}>: {1}",
UserName, message));
                }
            }

            readThread.Join();
```

```
            serialPort.Close();
        }

        public static void Read()
        {
            while (Continue)
            {
                try
                {
                    string message = serialPort.ReadLine();
                    Console.WriteLine(message);
                }
                catch (TimeoutException) { }
            }
        }

        public static string SetPortName(string
defaultPortName)
        {
            string portName;

            Console.WriteLine("Available Ports:");
            foreach (string s in SerialPort.GetPortNames())
            {
                Console.WriteLine("   {0}", s);
            }

            Console.Write("COM port({0}): ",
defaultPortName);
            portName = Console.ReadLine();
```

```
            if (portName == "")
            {
                portName = defaultPortName;
            }
            return portName;
        }

        public static int SetPortBaudRate(int
defaultPortBaudRate)
        {
            string baudRate;

            Console.Write("Baud Rate({0}): ",
defaultPortBaudRate);
            baudRate = Console.ReadLine();

            if (baudRate == "")
            {
                baudRate = defaultPortBaudRate.ToString();
            }

            return int.Parse(baudRate);
        }

        public static Parity SetPortParity(Parity
defaultPortParity)
        {
            string parity;

            Console.WriteLine("Parity :");
            foreach (string s in
Enum.GetNames(typeof(Parity)))
```

```
            {
                Console.WriteLine("   {0}", s);
            }

            Console.Write("Parity({0}):",
defaultPortParity.ToString());
            parity = Console.ReadLine();

            if (parity == "")
            {
                parity = defaultPortParity.ToString();
            }

            return (Parity)Enum.Parse(typeof(Parity),
parity);
        }

        public static int SetPortDataBits(int
defaultPortDataBits)
        {
            string dataBits;

            Console.Write("Data Bits({0}): ",
defaultPortDataBits);
            dataBits = Console.ReadLine();

            if (dataBits == "")
            {
                dataBits = defaultPortDataBits.ToString();
            }

            return int.Parse(dataBits);
```

```
        }


        public static StopBits SetPortStopBits(StopBits
defaultPortStopBits)

        {

            string stopBits;


            Console.WriteLine("Stop Bits:");

            foreach (string s in
Enum.GetNames(typeof(StopBits)))

            {

                Console.WriteLine("   {0}", s);

            }


            Console.Write("Stop Bits({0}):",
defaultPortStopBits.ToString());

            stopBits = Console.ReadLine();


            if (stopBits == "")

            {

                stopBits = defaultPortStopBits.ToString();

            }


            return (StopBits)Enum.Parse(typeof(StopBits),
stopBits);

        }


        public static Handshake SetPortHandshake(Handshake
defaultPortHandshake)

        {

            string handshake;
```

```
            Console.WriteLine("Handshake:");

            foreach (string s in
Enum.GetNames(typeof(Handshake)))

            {

                Console.WriteLine("   {0}", s);

            }


            Console.Write("Handshake({0}):",
defaultPortHandshake.ToString());

            handshake = Console.ReadLine();


            if (handshake == "")

            {

                handshake =
defaultPortHandshake.ToString();

            }


            return (Handshake)Enum.Parse(typeof(Handshake),
handshake);

        }

    }

}
```

//……

end……\\

Appendix V: Group members

BIRHANU TESFAY……....FST/UR123/99

EPHREM HAILU…….....FST/UR130/99

GOITOM YISFA…….FST/UR138/99

HAFTAMU GEBREEGZIABHER…..FST/UR140/99

REFERENCES

- http://msdn.microsoft.com/en-us/library/system.io.ports
- http://electrosofts.com/serial/
- www.howstuffworks.com
- www.RS232 Tutorial on Data Interface and cables.htm
- www.MAX232N Bus-Line Transceivers (MAX232CPE).htm
- www. Fiber optic cable tutorial .htm