1990

Free space laser communications on the Macintosh computers

Sohrab Modi

Rochester Institute of Technology
School of Computer Science and Technology

Free Space Laser Communications on the Macintosh Computers

by

Sohrab Modi

A thesis, submitted to
the Faculty of the School of Computer Science and Technology
in partial fulfillment of the requirements for the degree of
Master of Computer Science

Approved by :

3/19/90

Dr. James Heliotis (Chairman), Dept. of Computer Science,
Rochester Institute of Technology, New York 14623

3/20/90

Dr. Ronald Jodoin, Dept. of Physics,
Rochester Institute of Technology, New York 14623

3/20/90

Prof. Alan Kaminsky, Dept. of Information Technology,
Rochester Institute of Technology, New York 14623

Febuary 26 1990

I Sohrab F Modi hereby grant permission to the Wallace Memorial Library of RIT to reproduce, part of the thesis "Free Space Laser Communications on the Macintosh Computers" which does not include the program code. I prefer to be contacted each time a request for reproduction of the code is made.

March 21 1990 Sohrab Modi

Acknowledgements

This thesis is dedicated to my late grand father Dr. S. N. Deboo whose help and encourgment made it possible for me to achieve all my goals.

I wish to thank my advisor Dr. James Heliotis who always found the time to help advice and encourage me in this project, at times when I needed it most. I would also like to thank my advisors Dr. Ronald Jodoin and Prof. Alan Kaminsky without whose help I would not have been able to complete this project.

A special thank you to Dr. Arthur Kovacs for allowing me the use of the laboratory and any equipment that was available with the Physics department, to help complete my experiment. I also wish to thank Bill Vandeveer for all his help and advice and for always coming up with the equipment I needed. A special thank you to Prof. Guy Johnson for allowing me the use of his Macintosh Computer.

Finally, a very special thank you to my family who stood by me in my time of need and helped me attain my Goals.

Abstract

Lasers have proved their reliability in high speed communications through fibre optics. The disadvantage with fibre optics is the high cost and the need for a cable to connect the sender to the receivers, hence making it impractical to have fibre optic cables over large distances. The move today is to communicate between points with a free space laser (ie. a beam unbounded by cables). Semi conductor lasers as used in fibre optic cables are generally not strong enough to communicate over very large distances. Hence a high power laser is used with a modulation technique to transmit this data across.

This thesis demonstrates the use of a Helium Neon laser to communicate between two Macintosh computers. Bit signals generated by the Sender are converted to laser pulses with the help of crystals. These pulses are again regenerated as bit signals at the receiver.This complete hardware interface of the Macintosh to the laser was designed and built for this thesis.

The Software Protocol written for this thesis is similar to the sliding window protocol used by the ISO reference model. The protocol has been modified to handle high speed transmissions. The windowing environment, has been set up with pull down menu and submenu systems. Dialog boxes generate responses to errors. Disturbances in the beam result in generation of Nacks and the re transmission of bad buffers.

Table of Contents

CHAPTER 1
INTRODUCTION

1.0 INTRODUCTION.

Free space laser communications, is a technique by which data can be transmitted over a laser beam. The advantage of using light over radio waves lies in the higher bandwith. Also as Information capacity is proportional to carrier frequency and as the carrier frequency for light is high, it makes light a good medium for transporting data. Light however has a disadvantage; it tends to dissipate into its surroundings due to absorptions. Laser beams brighter and more directional and are therefore capable of travelling great distances before dissipating.

The idea of this thesis is to show that communications over a laser beam is possible under the simplest of conditions and at a minimal cost. Here I have used two Macintosh computers to act as sender and receiver. An electro optic light modulator is attached to the sender to act as a high speed shutter while a light receiving diode is attached to the receiver. A helium neon laser beam generated at source flies through the light modulator and is received at the diode.

The problem I have to deal with here is a two fold one. The first being the hardware interface of the electro optic light modulator (Pockel cell) to the Macintosh computer's serial port. The second being the software support to transmit and receive data over the helium neon Laser beam in a way that would minimize the errors in transmission. Also as I will be using the Macintosh computers I would be required to build some sort of display environment (like a window). Each topic will be addressed individually in the sections given below.

1.1 History of Free space Laser Communications. (1)

The birth of free space laser communications gave way to the discovery of what we know today as the Optical Fibre technology. However Fibre Optic cable also has its draw backs. The cost of having a Fibre Optic cable is prohibitive, for example the cost of having a Fibre Optic cable link between Japan and the United States which is in the process of being installed, is estimated to cost $ 700 Million. None the less due to high traffic conditions between the two countries, it is at this point in time more efficient to have this link than to have another satellite to handle the communication. This is mainly attributed to the fact that the Laser beam inside the Optical fibre can carry upto a few Gigabits /sec with a high degree of efficiency.

Research in free space Laser communications on a commercial basis started in earnest in 1962. Twenty seven years later there is still no known operational free space laser communication system. **ESA** (European Space Agency) is however developing an optical communication system for data link between two Geostationary satellites, or between a geostationary satellite and a low earth orbiting satellite. **(2)**

The program known as **SILEX** (Semiconductor Inter satellite Laser Experiment) was selected as the bottom line for the experiment. The following is the schedule for the progress of the European **D**ata **R**elay **S**atellite (known as **DRS**)

1989	Components qualification
1993 - 1994	In - Orbit demonstration of optical payload
1996	operational DRS

Funding in the United States has been mainly from

1) AirForce :-	Guiding System, tracking systems (Lasercom).
2) Navy & DARPA :-	Space/Subspace Communication.
3) NASA & JPL :-	Deep Space Communications.

Funding however has been the major source of delay for this program. Testing started in the1960's after successful development of an engineering model which demonstrated its ability to work under adverse conditions. A space Flight using a Nd: YAG (yttrium Aluminum Garnet) laser was funded, but the program contracted for in 1975 got caught in red tape, and had its funding cut one afternoon at the Pentagon, when an extra $ 300 Million was needed for the MX missile project.

The Space Flight Test system was near death when a last minute effort was made to revive it. The budget was untill 1977, and it was in December 1976 that a desperate bid to extend it through 1978 was made.

A test flight was quickly set up which proved to be a success. A plane (C - 135) flew into White Planes (N.M) while transmitting data at rates of 1 G bit/sec. This successful flight brought back confidence in the program and opened doors for the flight tracking systems.

The United States Airforce had its own problems with what was called Lasercom, in terms of delivery dates. Finally the project was terminated and came under shelter of SDI. The Lasercom Space Measurement Unit finally became operational and worked well when it became the SDI Laser radar experiment in 1986.

NASA which had been stopped by Congress from funding advanced communications, was brought in to work on a Laser Communication system on the Advanced Communication Technology Satellite (ACTS). NASA is working on this project along with Lincoln Labs (M.I.T.). which is developing a semi conductor Laser Communication Link, and Goddard is developing a semi conductor direct detection link.

Meanwhile the Navy and DARPA got together to develop a Submarine based communication Link. Blue and Green light, which travels through water at a relatively low loss offer Laser links as a means of improving sub communications. Development on this project took place and a successful air-to-submarine test was concluded when a plane flying through the clouds communicated with a submarine using a doubled Nd: YAG Laser.

There are four major reasons as to why Laser communications should be addressed. They are :

1) Wide Bandwith	Greater than 2 Gigabits /sec.
2) Small Antenna size	key viewing area as beam is focased.
3) Privacy	Interception of beam very difficult.
4) Jam Resistance	Jamming very difficult due to space orbital dynamics and narrow beams.

1.2 Hardware requirements and their drawbacks for the Thesis.

This laboratory set up is meant to show that such transmissions can take place and can be interfaced with any computer. The Macintosh uses an RS-422 type serial interface.

Although the RS-422 is logically the same as the RS-232 its electrical standards are different. The technique of ascertaining the polarity of the port by locating the negative voltage to the transmitter cannot be applied to an RS-422 port. This is because the Transmit and Receive lines of the RS-422 work on a differential voltage. The equipment used in the experiment for this Thesis is :-

1) Helium-Neon Laser source.

2) Electro-Optic Light Modulator (Pockel cell).

3) Pascal Laser switch (refer to appendix G).

4) a) Quadruple line receiver Ti 75189. b) Power transistor ECG 287.

5) 2 Power supplies for -18 V to + 18 V, one power supply from 0 to 300 V.

6) One Polarizer and one Analyzer.

7) One 150 Mhz Oscilloscope to keep track of input and output signals.

8) Two Macintosh computers.

9) differential line driver UA 9638 CP (Texas Inst.)

10) 8-pin DIN plugs, breadboards, etc.

Although the equipment has been so chosen to handle high transmission rates there are a lot of factors that affect those rates. Rise time of transistor is 2×10^{-8} sec, that of the Pockel cell is 2×10^{-11} sec,and of the Pascol switch is 2×10^{-9} sec. For a high transmission rate of 57600 bits/sec and up, absolute grounding of the equipment is essential. Another thing that is required in high speed transmission is, that the chips and power transistors used must be soldered into the printed cuircuit boards.

```
/*************************************************************************/
/*  The above routine calls this routine to check to see if this is a valid */
/*  header. It is a valid header if the packet sequence number is the one it*/
/*  expects or if the packet sequence is less than the present working wind */
/*  but greater than the previous working window (that is it checks to see  */
/*  if a retransmission is taking place of the previous window. Else it will*/
/*  send a Negitive acknowlegment.                                          */
/*************************************************************************/

CheckHeader( localSize )
int     localSize;
{

    int         previousStartWindow;
    Boolean     windowTurned = FALSE;
    previousStartWindow = startWorkWindow - NUM_OF_BUFFERS;

    if(previousStartWindow < 0)
    {
        previousStartWindow += (int)TOTAL_BUFFERS;
        windowTurned = TRUE;
    }


    /* This indicates that it has received the correct sequence number */
    if(pacSeqNum == readBuffNum )
        return(localSize);

    /*  this indicates that the ack/Nack had got lost and the sender
        retransmitted the previous Buffer so acknowledge its receipt */
    else if(( pacSeqNum >= previousStartWindow ) &&
            ( pacSeqNum < readBuffNum ))
    {
                skipPacket = TRUE;
                return(localSize);
    }
    else if((windowTurned == TRUE) && (((pacSeqNum >= previousStartWindow) &&
            (pacSeqNum <= 44)) || ((pacSeqNum >= 0) && (pacSeqNum< readBuffNum))))
    {
                skipPacket = TRUE;
                return(localSize);
    }

    /* We want to quit so inform Sender */
    else if((pacSeqNum == (unsigned char)END_HANDSHAKE) && (localSize == 0))
    {
        receivedClosing = TRUE;
        SendAckSender((unsigned char)END_HANDSHAKE);
        fclose(fwrit);
        return(localSize);
    }
    else
        return(-1);
}
```

However as these have a tendency to blow quite often (due to open wiring and adverse temperature conditions) it is necessary to mount them on breadboards. Open long wiring and extra connections (oscilloscope probes), extend the rise times. This affects the switching time and therefore a high transmission rate is not possible under these lab conditions.

CHAPTER 2
HARDWARE ITS
THEORY, DESIGN & IMPLEMENTATION

2.0.0 HARDWARE INTRODUCTION.

The equipment used in this demonstration is not the best equipment to use for the experiment but was easily avalable and very inexpensive (expense being a major factor). Also, to make the equipment as portable as possible most connectors and cuircuits are made in such a way that they can be easily dismantled. Low transmission rate is the price I have to pay for the above two factors.

2.1.0 A note about Lasers.

Laser stands for **L**ight **A**mplification by **S**timulated **E**mission of **R**adiation. A lot can be written about Lasers, but would go beyond the scope of this thesis. However a brief description could be useful.

A laser is a device that produces a highly directional and intense beam of coherent light. Ordinarily, when a beam of light travels through matter, the effects of absorption and spontaneous emission overpower the effects of stimulated emission. Hence there is a net reduction in intensity of the beam.

In the laser however, conditions are arranged so that stimulated emissions are dominant over absorptions and spontaneous emissions. The dominance of stimulated emissions creates a pure light source of a single frequency. This makes lasers an ideal medium for transporting data. Also a laser is an ideal medium due to its high frequency. **(5)**

2.1.1 Stimulated Emissions.

In the case of light, when photons enter a slab of material containing photon - absorbing atoms, the intensity of the beam is reduced. Each atom that absorbs the photon is left in an excited state and its energy is increased by hv (where h is Plancks constant and v is the frequency). It could stay in this excited state or in only 1 $x10^{-8}$ seconds emit the photon and return to its former state. The emitted photon however need not travel in the direction of propagation of the light beam thereby reducing the intensity of the light beam. In the case of stimulated emissions the photons striking an excited atom causes the excited atom to lose a photon hence the two photons travel hand in hand along the direction of propagation thereby increasing the intensity of the beam. However even in the stimulated emissions the net result is a loss in intensity, as their are many more absorptions than there are stimulated emissions, the reason being that most atoms are in their lowest energy state. **(7)**

2.1.2 Amplification through population inversion.

Population Inversion creates a situation where there are relatively many more atoms in an excited state capable of emitting photons than there are atoms in a state that can absorb them.Hence, a beam of photons passing through an inverted population gets amplified as there are more stimulated emissions than there are absorptions. Each photon of the incident beam can trigger the emission of others and each newborn photon can travel on to simulate more emissions. If a sufficient number of atoms are in an excited state an avalanche of photons can result. Population inversion can be obtained by what is called as optical pumping. **(7)**

2.1.3 Different types of Lasers and the Helium-Neon Laser.

There are many different types of lasers.

1) Ruby laser: Active medium consists of Al_2O_3 with a small concentration of Cr_2O_3. Laser rods are single crystals of ruby. Laser action takes place due to transitions from level 2 to level 1. Energy of transition is 1.8 eV, and the wavelength is 694nm.

2) Neodymium laser: This, like a ruby laser is also another optically pumped laser material. Lasers of this type are Nd:glass and Nd:YAG (yttrium aluminum garnet) The latter being a strong contender for satellite communications.

3) Organic dye lasers: This is another class of optically pumped systems. The active material is an organic dye dissolved in a suitable liquid. The dye molecules are a three level system with energy levels represented as bands. The width of the bands give rise to a broad fluorescence hence giving rise to what is known as tunable Lasers.

4) Ion lasers: Argon-Ion Lasers can be made to oscillate at several wavelengths. These are generally seen at the Blue-Green end in the visible spectrum. These have been used by the US Navy for submarine communications. Due to the high energy required to ionize the atom and raise it to an excited state the efficiencies of all arc discharge Lasers is very low. However once population inversion has taken place then these Lasers have a high gain and can produce several watts of continuous output power.

5) CO_2 lasers: These Lasers operate in the infrared region and are invisible to the naked eye. They can be used as continuous, pulsed or Q-switched type lasers.

Even a small fraction of a watt is capable of heating most materials to an incandescence in a short time. Hence they are used in things as welding and cutting..

6) Other Gas Lasers: These are helium-cadmium, water vapor, HCN, Nitrogen, TEA (transversely excited lasers) etc.

7) Semiconductor Laser: This is also known as a Junction laser or injection laser. They are important in optical communications and optical computer design. They are generally pn junction diodes of Gallium Aluminum Arsenide.

8) Helium-Neon Laser: This is the laser I will use for my thesis. The two essential components of a laser are the active medium and the optical resonator. In the Helium-Neon laser the active medium is a mixture of Helium and Neon. An active discharge will cause some of the atoms to ionize. Hence the active medium also contains positive ions as well as electrons. The optical resonator usually consists of a pair of mirrors which are made highly reflective. **(7)**

frequency of radiation

$$\upsilon = 4.7375 \times 10^{14} \text{ Hz}$$

The Laser wavelength

$$\lambda = \frac{c}{\upsilon} = 6.328 \times 10^{-7} \text{ m or } 632.8 \text{ nm}$$ (C = velocity of light)

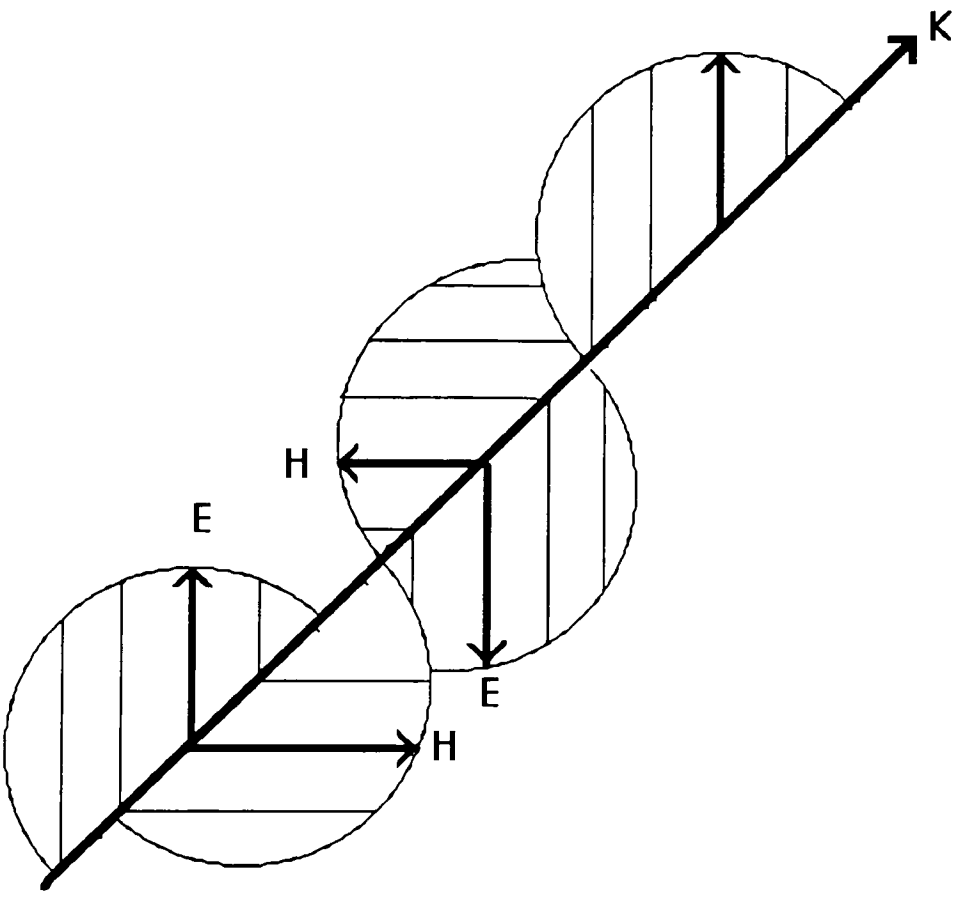
(this wave length gives the laser a red color).

2. 2. 0 Polarization Effects and Electromagnetic Effects.

2. 2. 1 A brief note on Polarized light and types of polarized light.

Light consists of time-varying electric and magnetic fields. These fields are vectors and their directions are always perpendicular to the direction of propagation.

When the electric-field vector E of a light wave lies in only one plane the light is said to be plane polarized. The magnetic field vector H is then perpendicular to both the direction of propagation and the electric-fields vector .



2. 2. 2 Electro - Optics

When certain liquids are placed in electric fields their molecules tend to align themselves parallel to the direction of the electric field. The greater the field strength the more complete the alignment of the molecules. As the molecules are not symmetrical, the alignment causes the liquid to become anisotropic and birefringent. Anisotropic materials are substances whose physical properties vary with direction. The anisotropy is determined by the crystal structure of the material. If a narrow beam of unpolarized light were shone on a crystal and two emerging beams were polarized orthogonally to one another,then such a phenomena is termed as *birefringence*. Such electric field induced birefringence in isotropic liquids is called as *Kerr effect.*

A Kerr cell is a cell containing Nitrobenzene or another suitable liquid between two flat parallel plates spaced by several millimeters. The Potential Difference between the plates is typically between 10 or 20 kV.

A polarizer is a device that transmits, only light whose component of electric field vector is parallel to an imaginary line known as the axis of the polarizer. The below diagram explains what is meant by cross polarizers.

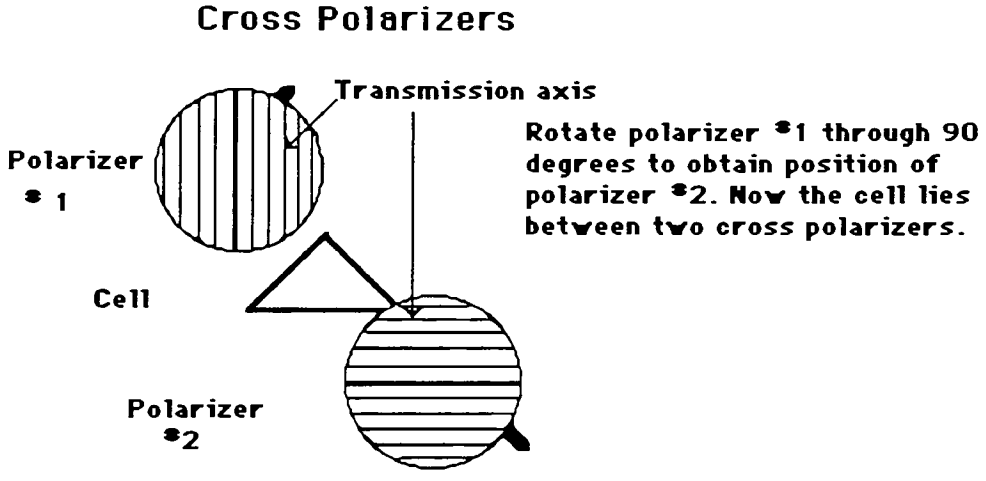


Figure 2.0

Hence if the cell is located between cross-polarizers as used in the experiment, it may be used as a fast shutter known as an electro-optic shutter.

In an electro-optic shutter, the direction of the electric fields is 45 degrees to the direction of the polarizer axis. When the field strength is zero the polarizers transmit no light. Whenever a voltage is applied to the cell it behaves as a half wave plate (explained in section 2.2.3) and rotates the plane of polarization by 90 degrees. The Kerr effect is very fast and the shutter speeds are limited only by the difficulty of generating fast electronic pulses in the Kilovolt range. Hence in this experiment I have instead used the *Pockel cell.* **(3)**

2. 2. 3 Wave Plates.

Let a thin slab of birefringent material be cut so that its optic axis lies in the plane of the surface as shown in the below figure.

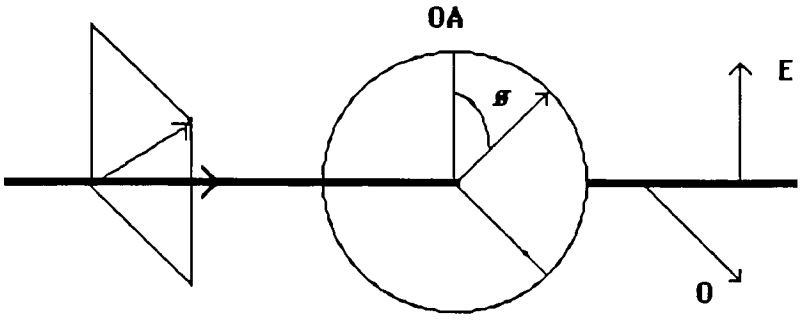


Figure 2.1

Suppose that plane polarized light falls at normal incident to the slab. Its electric feild vector then makes an angle ø with the optic axis. We resolve the vector into components perpendicular and parallel to the optic axis. The component parallel to the optic axis propagates through the crystal as an extraordinary ray while the component perpindicular to the optic axis propagates as an ordinary ray. Consider the optic axis to be a fast axis.

An optic axis is considered as a fast axis if a wave travels faster along the optic axis then perpendicular to it. It is dependent on the crystal. Calcite has a fast axis while mica has a slow axis. As the extraordinary ray is incident normal to the optic axis the extraordinary ray propagates through the crystal unchanged in direction but somewhat faster than the ordinary ray. Let the index of refraction for for ordinary rays be n_o and for extraordinary waves be n_e. If d is the thickness of the material then the extraordinary ray will lead the ordinary ray by $(n_o - n_e)d$ after leaving the crystal.

As a special case consider when $\phi = \pi/4$. Then the ordinary rays and the extraordinary rays will have the same magnitude. Further suppose that (no-ne)d is equal to $\lambda/4$ at some specific wavelength λ . Then the phase difference between the two plates is given as $\pi/2$ and the light transmitted by the crystal will be circularly polarized.The crystal is then called a quarter wave plate. Half wave plates cause a phase shift of π. The emerging wave is plane polarized with its electric field vector making an angle of ϕ with the optic axis. The half wave plate can therefore be used to rotate the plane of polarization through 2 ϕ.

2.2.4 The Pockel Cell.

The Pockel effect is an electro-optic effect that is observed in certain crystals such as Potassium Dihyhdrogen Phospate (KDP) or Potassium Dideuterium Phospate (KD*P). It differs from the Kerr effect in two ways. The Pockels effect is linear in applied electric fields, that is the angle of rotation of plane of polarization ϕ is proportional to the voltage whereas the Kerr effect is quadratic in applied electric fields, that is the angle of rotation of plane of polarization ϕ is proportional to the square of the voltage. The voltage required to drive the Kerr cell is very much larger than the Pockel cell.

The Pockel cell's electro-optic effect describes the relative phase change produced in polarized light passing through certain uniaxial crystals which are under stress of an electric field.

If the electric field is applied parallel to the optic axis the case is known as longitudinal Pockel effect. If the electric field is applied perpendicular t he optic axis the mode of operation is called as Transverse Pockel effect.

The transverse Pockel effect has many advantages over the longitudinal effect. First the electrodes lie parallel to the beam and do not obscure it (shown in figure below). Second the index difference $(n_0 \quad n_e)'$ depends on the electric-field strength in the crystal, and not on the voltage between the electrodes. In the case of the Longitudinal Pockel cell if the length of the crystal is increased but the voltage maintained constant the electric field in the crystal will decrease proportionaltely. Hence the index difference will decrease and the phase diffrence or retardation between the two polarizations will remain independent of the length of the crystal.

On the other hand, in a transverse Pockel cell, the electrodes need be seperated by the diameter of the beam and no more. The electric field strength in the crystal depends on the seperation between the electrodes and not on the length of the crystal. Consequently lengthening the crystal and maintaining the same seperation between the electrodes will result in increased retardation.

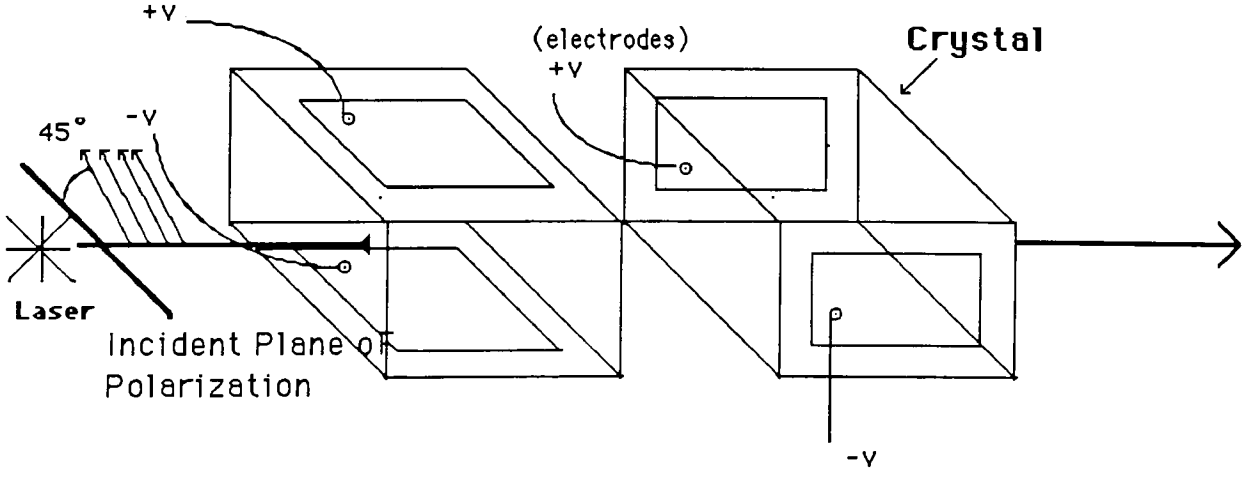


Figure 2.2

I have used a Pockel cell of the transverse fields operation made by Lasermetrics Inc. This has Potassium Diderterium Phosphate as its crystal material, its window material is made of Fused Silica and requires a voltage of 250 V for it to behave as a half wave plate. **(4)**

2. 3. 0 A brief discretion of how transmission takes place.

2. 3. 1 At the Serial Port.

Let us begin at the Serial Port of the Sender. As explained before the serial port is a RS-422 type. Generally a one bit is represented by some voltage greater than +3 Volts and a 0 bit is represented by some voltage less than -3 Volts. The signal is said to be differential as these voltages are not relative to ground but instead depend on voltage diffrence between two lines.This differential signal has made things easier for me in one of the circuits that I have built. To convert a TTL signal to an RS-232 signal a line driver chip would be needed (Ti 75188).

2. 3. 2 Circuit #1.

The differential signal coming out of the Sender's serial port is fed into the line receiver. The line receiver chip made by Texas Instruments is a quadruple receiver. The output is fed into one of the receivers and a +5 Volt supply is fed at pin number 15. The line receiver converts the differential signal into a TTL signal. This conversion from RS-422 to TTL is necessary as the power transistor to which this signal is fed switches between 0 and +5 Volts.

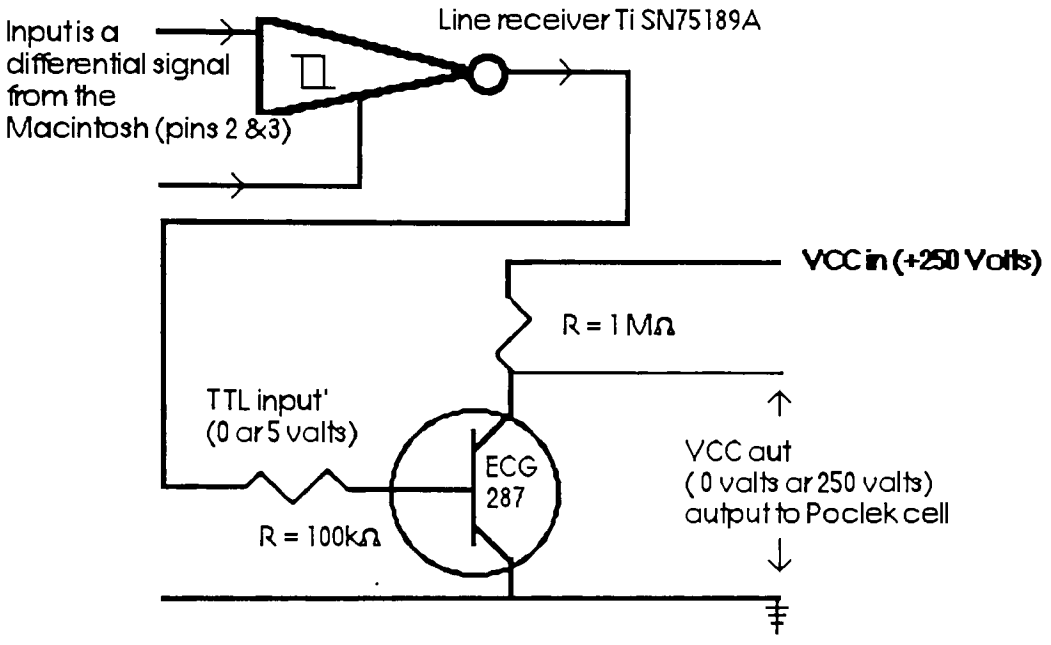


Figure 2.3

When a 1 bit is output from the serial port the line receiver puts out +5 volts on the base of the transistar. At this point the transistor is on and no significant voltage (about +5 Volts) appears between the collector and the emmiter of the transistor. However when a 0 bit is output from the serial port the the receiver sends 0 volts to the transistor. Now the transistor goes off and 250 V appears from between the collector and the emmiter. Hence as the stream of bits erupt from the serial port the transistor switches between +5 and 250 Valts. At this point it is important to mention something else. The bit rate will determine how long the 250 Volts or the +5 Volts be thrown for.

The transistor has a response of 50 MHz or rise time of 2 x10 $^{-8}$ seconds. Now for a 2400 bps transmission rate the voltage pulse sent from the serial port is for a time duration of 4.17 x10 $^{-4}$ seconds.

As the rise time for the transistor is only 2 x10 $^{-8}$ seconds 250 Volts is output for 99.995 % of the cycle. This fraction of time lost due to the rise time is well into the uncertainty range and would create no loss in pulses generated.

2.3.3 At the Pockel Cell

The Pockel cell is connected to the other end of circuit #1 (Appendix A). The Helium Neon Laser source is so positioned that the Laser beam enters the Polarizar and goes through the aperture of the Pockel cell. The beam exists from the other side and goes through the analyzer. The beam that shoots out can either be received directly by a laser switch or can be bounced off mirrors before being received.

As explained in section 2. 2. 2, the direction of the electric field is at 45 degrees to the Polarizer and analyzer axis. When the field strength is zero the Polarizers transmit no light. Going by this concept, when a 1 bit leaves the serial port (as a +5 Volt) it enters circuit # 1 and puts the transistor on, which throws negligible potential accross the Pockel cell. This has no effect on the cell and the apparatus behaves as explained above. The intensity of the beam is thereby cut. However here we have to make a small change to the system. We need to maximize the intensity when a 1 bit leaves the serial port. This restriction is brought on by the Pascal switch. Hence instead of modifying circuit #1 all we need to do now is rotate the plane of polarization through 45 degrees. This is done by adjusting the Polarizer and the Analyzer such that they are both in phase with one another. Hence as shown in figure 2.0 the polarizer #2 would have to be moved back 45 degrees. Now the intensity of the beam is maximized. Figure 2.5 gives the entire picture.

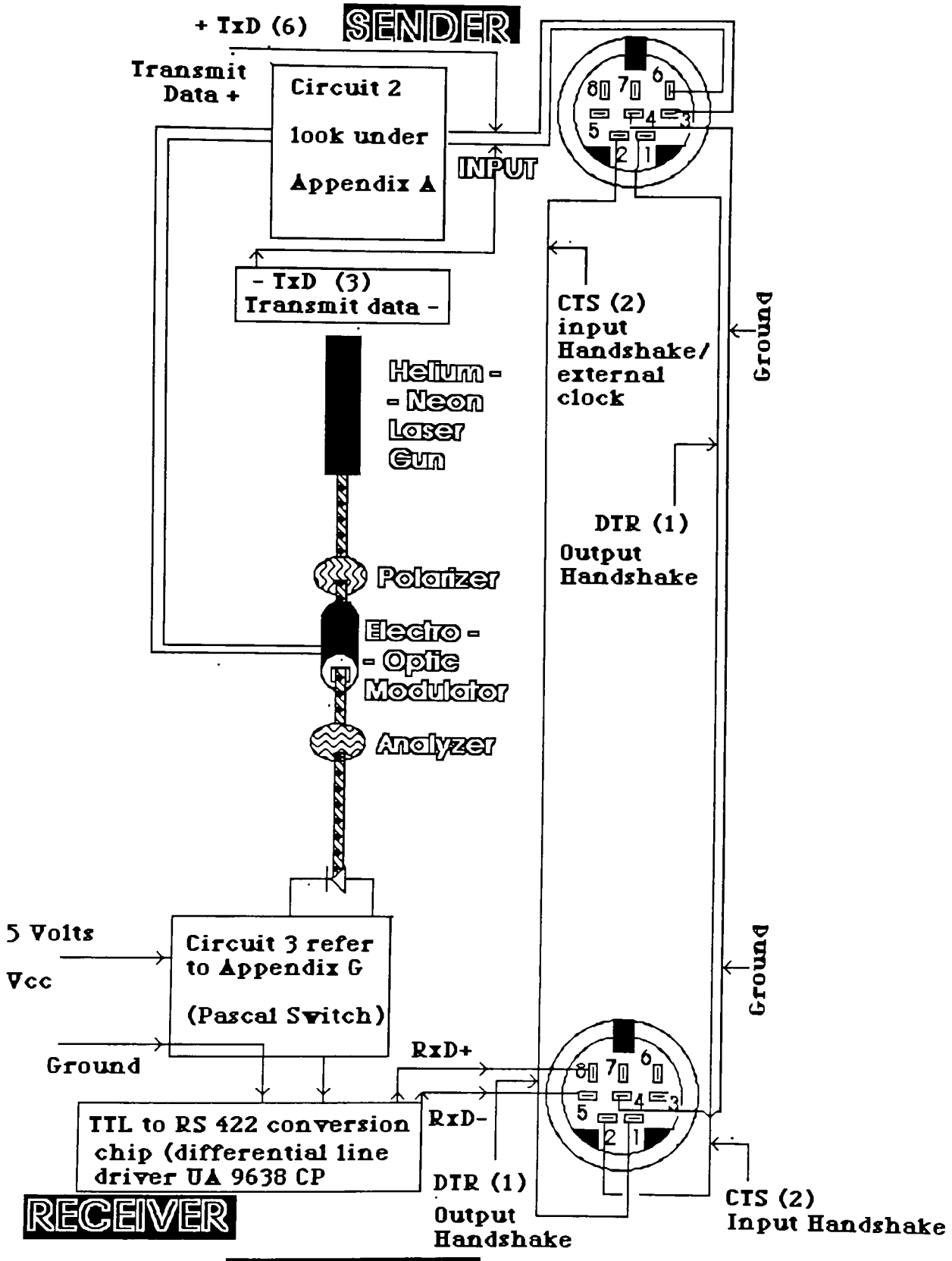

+ TxD (6)
SENDER
Transmit Data +
Circuit 2
look under
Appendix A
INPUT
- TxD (3)
Transmit data -
Helium - - Neon Laser Gun
Polarizer
Electro - - Optic Modulator
Analyzer
CTS (2) input Handshake/ external clock
Ground
DTR (1) Output Handshake
5 Volts
Vcc
Circuit 3 refer to Appendix G
(Pascal Switch)
Ground
RxD+
RxD-
TTL to RS 422 conversion chip (differential line driver UA 9638 CP
RECEIVER
DTR (1) Output Handshake
Ground
CTS (2) Input Handshake

Figure 2.5

Hence when a 1 bit leaves the sender's serial port the transistor is on and + 5 Volts appears between the emmiter and the collector. This +5 Volts is fed to the Pockel cell making no change to the crystals. The uniaxial crystals are now under a near-zero field strength making no change. Hence the Polarizar and Anylazer maximize the intensity of the beam.

Now when a 0 bit leaves the Sender's serial port (> -3 V) it is represented as a 0 volt at the base of the transistor, thereby turning the transistor off. Now 250 Volts appears from the power supply and between the collector and emmiter to the Pockel cell. This 250 Volts is the half Wave Voltage required to drive the Pockel cell. The Uniaxial crystals inside the Pockel cell coming under influence of this electric field rotate the plane of polarization through 90 degrees and in essence bring about a shift in Phase. Because of this in combination with the orientations of the polarizer and analyzer,the intensity of the beam gets cut.

2. 3. 4 The Pascal Switch (Laser switch).

The Pascal switch is a device for receiving the intensity of the beam and then generating a pulse depending on the intensity. The main component of the switch is a Photodiode for receiving the beam. The Pascal switch has a +5 Volt supply attached. When the intensity of the beam is maximized at the diode the circuit drawn below will throw 5 volts to the output.

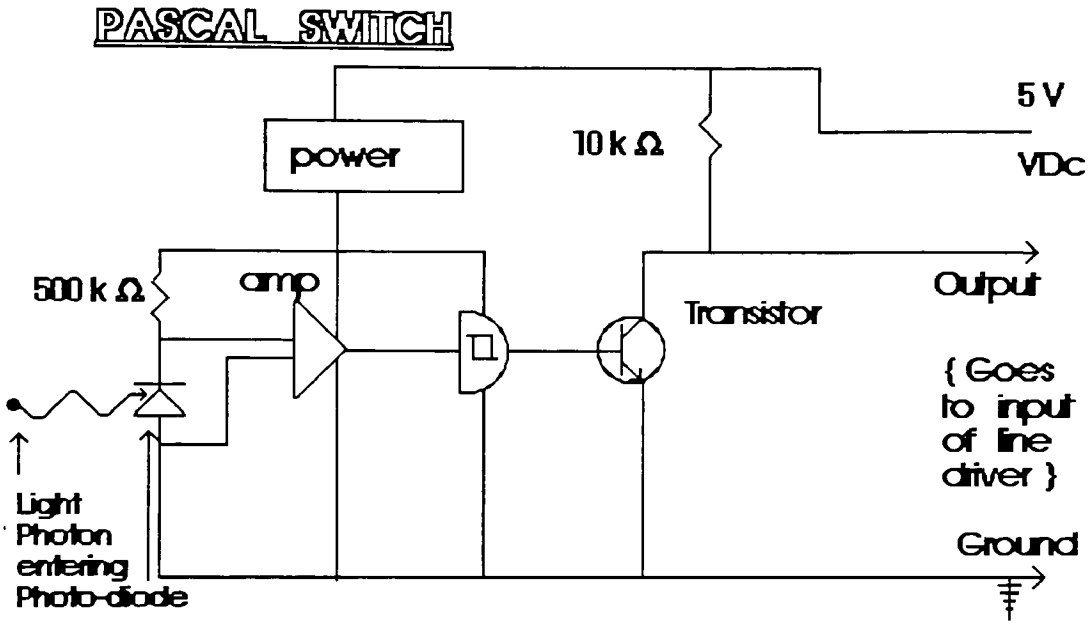


Figure 2.6

If however the intensity is minimized 0 volts is thrown at the output. Hence the Pascal switch switches between 0 and +5 Volts (TTL). However the Macintosh must accept a RS-422 type signal. Hence this output is fed into a differential line driver of the type made by Texas Instruments(UA 9638 CP). The other end of the line driver is connected to the Receiver's serial port. This signal is fed into pins 5 (RxD-) and 8 (RxD+).

2. 3. 5 Data Transmission Summary.

Characters are sent from the serial port in their binary form. A high bit or 1 bit is represented by a differential voltage greater than +3 Volts emerging from the serial port for a particular length of time depending on the bit rate set up. A low bit or a 0 bit is represented by a differential voltage of less than -3 Volts. The serial port when unused is always in a high state. When transmission is to take place the port goes into a low state before transmission of data can take place. When a 0 bit gets transmitted (-3 volts), it first gets converted to TTL by the line receiver chip. This voltage is fed in at the base of the transistor (as shown in figure 2.3) which turns the transistor off thereby throwing 250 volts between the base and the collector. This 250 volts is fed into the Pockel cell. When the uniaxil crystals in the pockel cell come under influence of the electric field generated by this voltage, they change the plane of polarization of the laser beam which is incident on them. This brings about a phase and amplitude change. This amplitude change is represented by a drop in intensity of the beam. Hence the Pockel cell behaves as a high speed optical shutter. This drop in intensity which is monitored by the Pascal switch forces the circuit in the switch to pass 0 volts to the differential line driver. This differential line driver now feeds a low to the receive ports in the Receiver Macintosh. Hence a 0 bit sent out from the serial port of the Sender is received as a 0 bit by the receiver. Similarly in the case of a 1 bit the reverse takes place whereby the Pockel cell does not come under influence of the electric field thereby registering no drop in intensity.

Due to equipment restrictions only forward transmission of data will take place over the laser beam. That is transmissions from sender to receiver. However any transmissions in the reverse direction (from receiver to sender) will take place over a wire as shown in the diagram.

CHAPTER 3
SOFTWARE DESIGN

3.0.0 Introduction.

The protocol used in the transmission of data over the laser is based on the sliding window protocol. Sliding window protocols are used to reduce round trip propagation delay. This chapter only deals with the basic design of the protocol. It states its uses, effectiveness and puts forward arguments in its favour. Most of the discussion in this chapter has been implemented in the supporting program. Topics discussed here, but not implemented rises from the fact that there is an alternative or easier way to implement the same thought. Implementation details are discussed in chapter 5.

3.1.0 Understanding the need for sliding window protocols.

Consider a file of 10K Bytes being transmitted through a serial port not using the sliding window protocol. Let the sender send a packet. The receiver receives it and then sends an acknowledgment (ack) back. In both cases there is a delay. Once the sender sends the file it waits for the ack to come back, and when the receiver sends the ack it waits till the next file is sent . This delay is called propagation delay. The sliding window protocol is meant to reduce this delay. If the propagation delay were known or calculated then we could transmit multiple buffers from the sender until it was time to receive the ack. Say for example the output buffer is broken up into 5 buffers of 1 K bytes each and sender sends all 5 buffers.

Now by the time the last buffer is sent it should start receiving the ack for buffer 1. If it receives a positive ack for that buffer it will free that buffer and load the next message. If it receives a negative ack for buffer 1 it will retransmit the same buffer. If buffer 1 times out or it receives an ack for the next buffer then it retransmits the old buffer again. Similarly it will repeat itself through all the buffers until the entire file is sent.

3. 1. 1 Higher Transmission Rates.

A thing to remember in communications over a laser is to try and maximize the transmission rate and reduce delays resulting in processing and propagation. High speed transmissions will also require very high speeds in processing. Consider the case where packets arrive at two GBits/sec. The receiver must be in a position to either tell the sender to slow down if it cannot keep polling and checking the packets that fast or, it should have large serial port input buffers to place the information in as it comes in and then pull out from this input buffer. I have gone with the latter (large serial port input buffers) as it creates less timing problems and helps me make the system more concurrent. Also having this method does away with the need of having the receiver tell the sender to slow down, which will be in the form of a negative acknowledgment, as the only way the receiver knows that it cannot keep up, is if it starts loosing data. A drawback with large buffers is when a Nack takes place. When this happens the receiver may have in its buffer, packets sent by the sender before the nack was generated, but not yet accepted. These packets are now redundent. However the receiver will have to go through its entire large input buffer to search out the header for that packet sequence number. This causes a problem. It increases processing which may result in the buffer to overflow if it cannot find the newly re-transmitted packet in time.

3. 1. 2 Changes to the Protocol.

The sliding window Protocol has been modified to receive acknowledgments after every window of fifteen packets, as opposed to every packet as defined under the reference model. This eases the overhead on the processing by having to do away with timers on every packet buffer. Also, if the transmission is good then it does away with the over head of having to send an ack for every packet for the receiver as well as having to receive an ack for every packet for the sender. The drawback here is that if an ack gets lost there was no way of know which buffers reached the receiver. Hence the entire window of fifteen packets has to retransmitted.

Other questions are, how many sequence numbers should one have, how many packets should be in each window, how do we maintain a sliding window protocol, and how does one calculate the time out. The number of sequence numbers, the number of packets within a window and the size of the packet could be left up to the programmer. The important consideration in judging these sizes would be transmission speeds, propogation delay, buffer space and the probability for the packet to get through in the minimum number of re-transmissions. The concept of how the sliding window protocol is maintained is explained in the 3 case scenarios explained in this chapter.

3. 1. 3. Calculation of time outs.

Let

ticks = the number of ticks of the CPU clock/second for that system.

timeSent = the time the packet was sent out at .

item = data unit exmple:- character, integer.

distanceTravelled = distance to receiver in meters.

errorFactor = a number which will allow for tolerance.

Then

$$\text{oneBuffTimeOut} = \frac{\text{number of items per packet} * \text{number of bits per item}}{\text{transmission rate(in bits) or baud rate}} * \text{ticks}$$

$$\text{propogatioin Delay} = \frac{\text{distanceTravelled}}{3 * 10^{8} \text{ meters/ sec}} \quad \text{(speed of light)}$$

timeOut = (oneBuffTimeOut * numForPackets) + propagation delay.

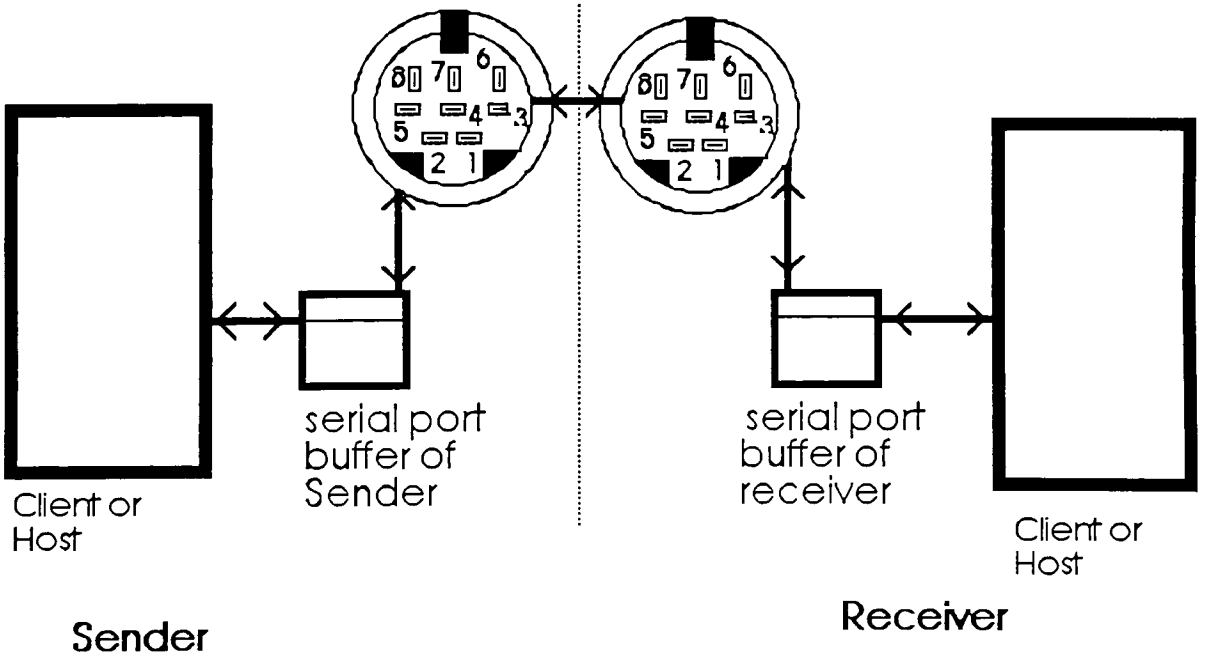
Hence

```
if ( present time >= TimeSent + timeOut)
        Raise Error.
```

There could be other factors that could have bearing on a time out period as is the case in this program implementation and is discussed in chapter 5.

3.2.0 The Protocol Scenario.

A file to be transferred from one terminal to another is first broken up into small packets. Each packet or frame is broken up into a 5 byte header followed by data. The header describes the logistics of the packet following it (for example the size, packet number, etc). Details about the header are described in chapter 5. The packet itself contains data followed by an end of packet marker. If the end of packet marker exists as data in the packet then another marker is put before it to identify it from the real end of packet marker. This process is called as byte stuffing and has been used in the implementation. Details about the packet and headers are discussed in chapter 5.



3. 2. 1 Scenario # 1 : All frames sent correctly.

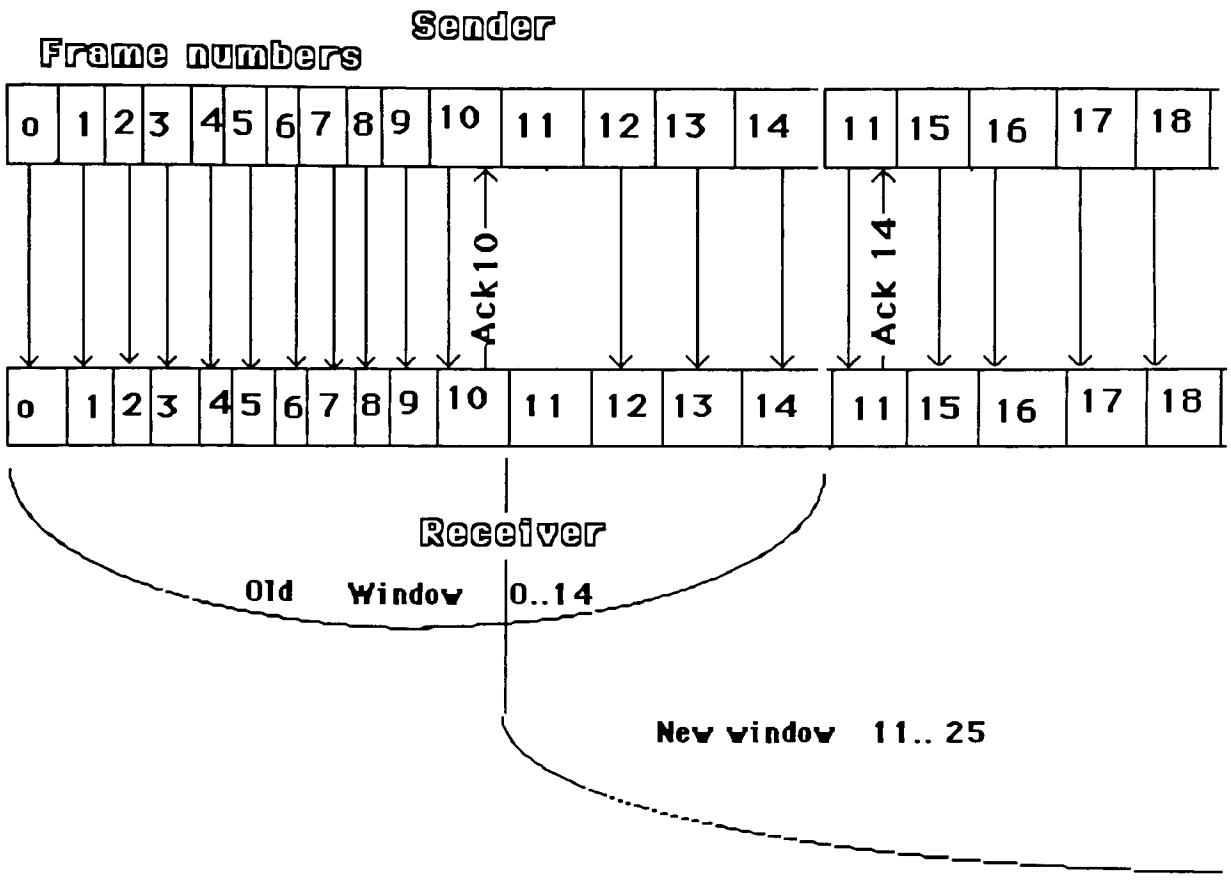
Consider the scenario where no error takes place when transmitting the entire file.

Let us assume that the sender has sent out 15 (0..14) frames. The receiver will accept all frames whose sequence it is expecting. It expects these sequence numbers to come in order. It accepts each frame by checking the control character and counting the number of bytes in each frame. When the receiver receives all 15 (0..14) frames it sends an ack back for the 14th frame. It now moves its window down another 15 buffers thereby accepting all frames from 15..29. The sender in the mean while does not wait for the N/ack of the previous window (0..14) to come in.

While transmitting frames 15..29 it should receive an ack for frame 14. It however starts a timer at the end of sending the last buffer of a window (in this case at 14). If by the time the timeout period is over and it has not received an ack for the previous window it stops its transmission and retransmits the previous window. If however it receives no acks or timeouts it keeps receiving acks correctly and then sends an end transmission header.

3. 2. 2 Scenario # 2 : Negative acknowledgments received.

Now consider the scenario where some of the frames have to be retransmitted due to errors. As an example say that all 15 frames were sent out, 0..14. Now the receiver sees frame 11 as an error. It sends out a Nack for frame 11 and discards everything following this frame. The sender gets the Nack for frame 11. On receiving a Nack it stops transmission and makes the lower end of its new working window the frame number for which the Nack was received. The new working window for the sender and the receiver now becomes 11..25.



The sender now moves its window from 11 to 25 and it transmits these buffers and then transmits the next window (that is from 26..40). The receiver at its end does a similar thing. It moves its window up by sending all the buffers 0..10 to the client. The receiver then discards all frames received after the frame for which the Nack was sent and gets ready to accept frames 11..25.

3.2.3 Scenario # 3 : Sender time outs.

Consider the case where the N/ack coming in gets lost. The sender would be left waiting forever for the acknowledgment. Therefore the sender must be made to time out after a certain time. This fixed,preset time is defined by the timeout calculation as shown in section 3.1.3. For the sake of simplicity let us consider this time to be a quarter second. Let the sender send out its first two windows of 30 frames (0..29). Let us assume that the first window of 15 frames all arrived safely and that an ack was sent back. The receiver has now updated its window to receive frames 15..29. Their ack however gets lost. The sender keeps track of what time each window was sent out by means of the system clock. While the sender has the timer running, it is mean while transmitting its next window. If the N/ack comes back within this time out period one of the above two scenarios (#1 and #2) is engaged.

If however an ack is not received within this time limit the sender times out and retransmits all the frames for that previous window again. The Receiver receives these frames and accepts them knowing that they are part of the previous window and then discards them. as it already has them. It however sends an ack back for the last frame received in or else the sender would time out again and again retransmit. The problem arises when a Nack gets lost and a time out takes place.

Consider as an example: Let the sender send packets 0..14. Let the packet number 11 generate a Nack. Let this Nack be lost. The sender sends packet 14,starts a timer and expects an ack for 14 before the timer runs out. However the ack for 14 will never come as receiver has updated its window to start from the bad packet (packet # 11) and the next ack it sends will be for packet #25.

The sender will eventually time out and retransmit all packets 0 through 14 again. This time however it receives an ack for packet #10. As 10 lies within its working window it now realizes that the Nack was lost and updates its window to start at 11 and end at 25.

3. 2. 4 Scenario # 4 : Receiver time outs.

Though Receiver time outs are not required as it is handled by the sender timing out, it could be implemented as an efficiency measure. For example, in the case where the sender sends packets 0..14. If a Nack is generated for packet #1 but this Nack gets lost, then the Receiver has to wait till the sender times out before hoping for a retransmission. This however can be avoided by the receiver timing out and sending a Nack again.

3. 2. 5 Headers & Errors

The acceptability of errors in communications depends on the content, the data and the ultimate use to which the data is to be put. Extreme high error rates are acceptable in many cases for example: in certain forms of telecommunication, as the human brain has the ability to understand even highly corrupted data. However no errors are acceptable in things like missile guidance systems. Stronger error detection schemes will slow down the transmission rates and hence the user must eventually justify what sort of error detection and correction scheme should be applied. The one advantage of laser communications is that it is neither susceptible to transient errors (those that arise due to electromagnetic effects) nor generation of noise due to interference, common to radio wave transmissions. Interference with the beam, however, will result in loss of data and not the generation of noise.

The header is an important part in the detection of errors. The header must be such that it could easily be recognized within chunks of data. This is specially important in the case of this implementation as all incomming packets are buffered before they are actually processed. It is important that the header explicitly state the packet sequence number and the number of bytes in the packet. The actual number of data characters in the packet could be left up to the programmer to implement and I have chosen 1023 as the number of data characters in the packet. There is also an end of packet marker, bringing the total size of the packet 1024 bytes (0..1023 as represented in the program). Hence in the detection of errors, the packet sequence number is the first thing it looks for in the header. If it is the correct packet sequence number or the one it expects then it could do one of two things 1) it could go on looking for a end of packet marker and if it finds one it could check to see if it is has picked up the number of characters as specified in the packet header. If it has, then it would know that it has picked up the entire packet. If however it has not picked up enough characters and it finds an end of packet marker then it knows that the marker is to be treated as data and moves on to find another one. If it does not find the marker by the time it pulls out the number of characters as specified by the header, the receiver then knows that something has gone wrong and sends a nack for the sequence number it is holding.

2) Another way of handling the same situation is where the receiver picks up the number of characters specified in the header from the large input port buffer. It then checks to see if the last character picked up is equal to the end of packet marker. This technique is the one that I have implemented in the protocol.

As a further measure in detecting errors a technique more commonly known as byte stuffing can be implemented. It is implemented in the current version of the program I am running. If the end of packet marker exists as data within the packet itself then, another marker is put in front or behind it indicating that this is data and must not be treated as the end of packet. Hence when it encounters 2 such consecutive markers, and the number of characters pulled out for that packet is less than the number of characters in the header , it realizes that the end of packet marker is part of the data and hence accepts one of them and discards the other. As in the case of all error checking schemes there are no guarantees, but at least this will reduce the chance of errors.

The error detection scheme mentioned here has worked flawlessly in all the tests conducted and has detected errors correctly. Another detection scheme implemented is the actual counting of characters after the byte stuffing and the end of packet marker has been stripped off.

3. 3. 0 Conclusions.

In my opinion the perfect protocol, that is one that will to suit all needs with maximum optimization(if such a thing does exist) is still a long way from being solved. There are many constraints in this problem which add to making an solution very difficult.

None the less the error detection scheme and the time optimization scheme proposed by the sliding window protocol is good and in lieu of anything better, is a scheme I would propose be used. I also feel the modifications to the sliding window protocol that I have put forward are also valid situations though I do have recommendations put forward in chapter 6 which I feel will aid in easing the implementation.

CHAPTER 4
PROGRAMMING DESIGN

4.0.0 Introduction.

This chapter will deal with the peculiarities of the language and the display environment which played an integral part in the design of the software for this program.

4.1.0 The Language .

The program to implement the protocol is written in the C language. The Compiler used is called Lightspeed C developed by Think Technologies. This compiler follows the syntax as defined under ANSI C. There are advantages and disadvantages in this compiler and a brief discussion of them will follow. None the less this compiler is good in the sense that it supports the entire ANSI C language and yet, is fast enough to use effectively. Lightspeed C is fast and very easy to learn.The unusual thing in using the Macintosh is how to adapt oneself to use the toolBox routines which are written in Pascal. This is discussed in 3.1.1 and its use can be seen extensively in the code.

4.1.1 The Think C Language.

a) Under Think C pascal is a reserved word. An identifier which has been declared as pascal must have type "function returning...". A pascal function is called using pascal calling conventions.

b) Under C, each string is terminated by a null byte ('\0'). The program scans the string till it finds the null byte in order for it to know the end of string. However the Macintosh tool box routines are written in Pascal. Unlike C, the Pascal string is not terminated by a null byte. Instead it contains a byte in the beginning which indicates the length of the string.

example:

C version :- DrawString("HELLO WORLD**0**")

Pascal version :- DrawString("**11**HELLO WORLD")

Passing a string of C type into a tool Box routine requiring a Pascal type will cause garbage to be printed on the screen along with parts of the string. To over come this problem Think C has a special identifier which converts C strings to a Pascal string type. This is done by putting either \p or \P as the first character in the string.

Hence it appears something like this :- DrawString("\pHello World").

c) When passing parameters into pascal routines the need to pass it by reference may arise. Think C has rules that allows the C programmer to do the right thing.

4.2.0 Events, their use and how they have affected the software design.

In the Macintosh,an event can be generated by the user of the application, other tool box routines, device drivers, or the application itself.

Macintosh events hold descriptions of these occurrences in a part of the operating system called the Event Manager. The Event Manager captures some important information about the occurrence in an Event Record.

An application can get information by retrieving Event Records from the event queue one at a time. The Event queue is a FIFO queue. The Event Manager gets events from all the different sources, queues them up and passes them to your application one at a time. The type of event occurred is stored in the event record as an integer number. There are 12 predefined events and 4 user defined events that can occur. The events described are.

Event	Integer Value
nullEvent	0
mouseDown	1
mouseUp	2
keydown	3
keyUp	4
autoKey	5
updateEvt	6
diskEvt	7
activateEvt	8
networkEvt	10
driverEvt	11
app1Evt	12
app2Evt	13
app3Evt	14
app4Evt	15

Hence when the mouse is pressed down the EventRecord field what contains 1. The where field contains the position of the mouse in global coordinates at the time the event was posted. The message field of the event record contains the event message, which conveys additional information about the event. The type of information conveyed will depend on the type of event. For example if I were to press down the letter C on the keyboard, then the EventRecord.what = 3 and EventRecord.message = 'C'.

All the above three have been used extensively in the supporting software protocol.

Another thing used, though not quite in conjunction with events, is TickCount(). In the event record whenever an event takes place the event time is stored in terms of TickCounts.

TickCounts return a long number and represent the number of ticks taken place from startup. TickCounts get updated every 1/60 of a second. This is an important fraction in the calculation for Time outs and propagation delay and has been widely used in the program.

Examples of use of events within the program.

Module : **procedures.c.**

procedure : **EventHandler().**

type : **mouseDown event.**

```
EventHandler(event)
EventRecord          *event;
{
        switch(event->what)
        {
                case nullEvent:
                        break;
                case   mouseDown:          <---------- case   1:
                        HandleMouseDown();
                        break;
```

Module : **procedures.c.**

procedure : **EventHandler().**

type : **message event.**

```
                        :
                case mouseDown:
                        HandleMouseDown();
                        break;
                case mouseUp:
                        break;
                case keyDown:
                        putout((int) event-> message  & 0x7f);
                        break;
                        :
```

Module : **display.c**

procedure : **EventHandler().**

type : **where event.**

```
HandleMouseDown()
{
        WindowPtr   whichWindow;
        short int   thePart;
        long int    menuChoice,
                    windSize;

        thePart = FindWindow( myEvent.where, &whichWindow ); /*    "tool box"
        * /
        switch ( thePart )              /* the part returns a number describing where
        * /
        {
                case inMenuBar:
                        :
                        :
                        :
```

Examples of TickCounts will be discussed later.

4. 3. 0 Macintosh Resources , and its implementation in the program.

A better presentation of the program to the user will always be an added attraction to to use the program. The use of windows, menu bars, dialog boxes, help not only to attract the user but also helps in the learning process. With the ease of use, as portrayed under the windowing environments, the user has the ability to master a program in a short time.

Most of the environment built on the Macintosh is done by tool Box routine calls. For example if we wish to create a window, we call the tool box routine :-

NewWindow(p, bounds, title, visible, procID, behind, goAwayFlag, refCon);

where:

p	is of type	Ptr.
bounds	is of type	Rect.
title	is of type	STR255.
visible	is of type	BOOLEAN.
procID	is of type	Integer.
behind	is of type	WindowPtr.
goAwayFlag	is of type	Boolean.
refCon	is of type	Long Integer.

Now instead of having to figure out all this data before having to open a window the Macintosh has special routines which aid in the development of this environment. Data governing building environments can be put in a special area assigned by the Macintosh Operating system called as the resource fork. A call to this resource can be made by a user defined resource ID number.

To put this data in the resource fork Apple has built special editors. The editor used here is called ResEdit. ResEdit will lets you choose the type of item you wish to build and helps you go through it step by step in the process.

For example in building the window, instead of having to figure out all the data that goes in to the routine NewWindow(...) ResEdit lets you draw your window size on the screen thereby locating its starting position, its size, etc. You now assign this window resource a name, and an ID number. Now to call the resource simply pass the resource ID to the respective routine.

That is :

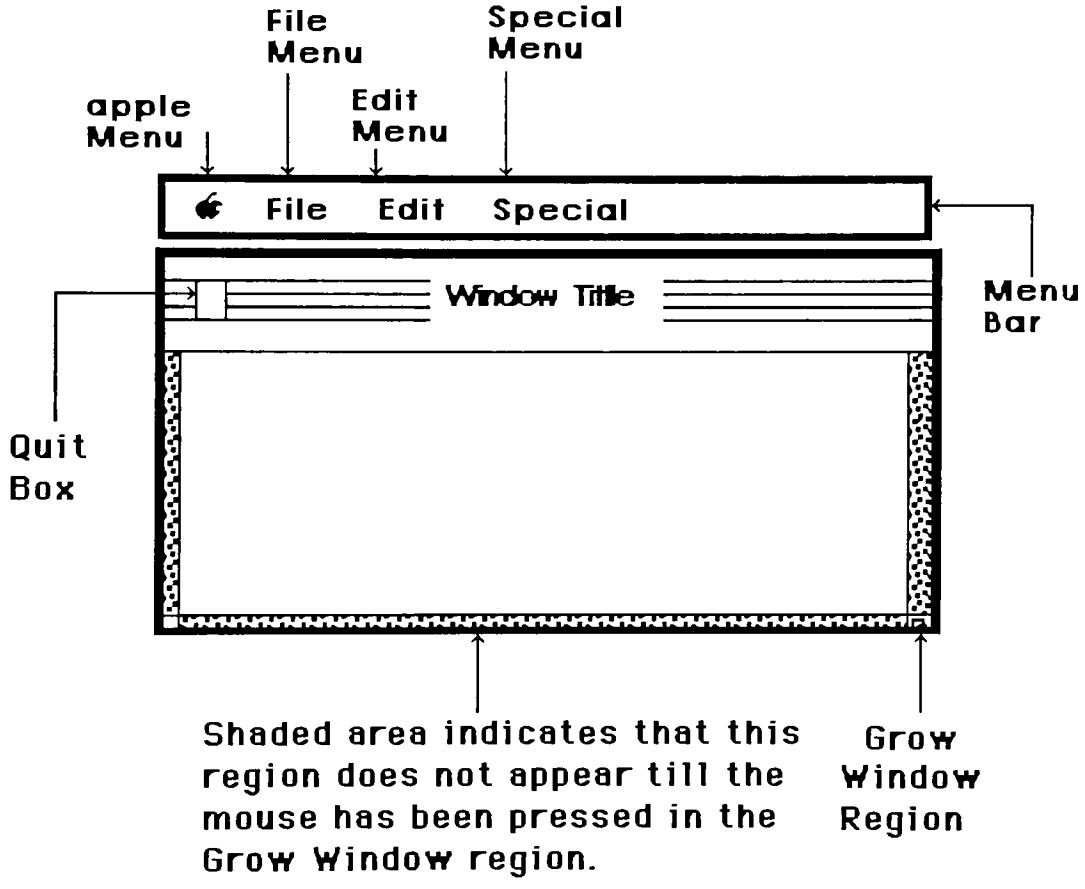
GetNewWindow(windowID,wStorage,behind)

In comparing the two calls one can see that the parameter windowID has replaced the parameters boundsRect, title, visible, procID, goAwayFlag and refCon.

The resources used to build the environment for Demonstrating this protocol are Windows, Menu Bars, all menus and sub menu systems,Dialog Boxes, Alert boxes, error strings, greeting picture, and different cursor shapes. The only unusual factor that occured was in building the sub menu systems, one had to toy around with a bit of hex code. This is because ResEdit does not directly support sub menus.

4.4.0 The Display environment.

4.4.1 The Model of the window environment.

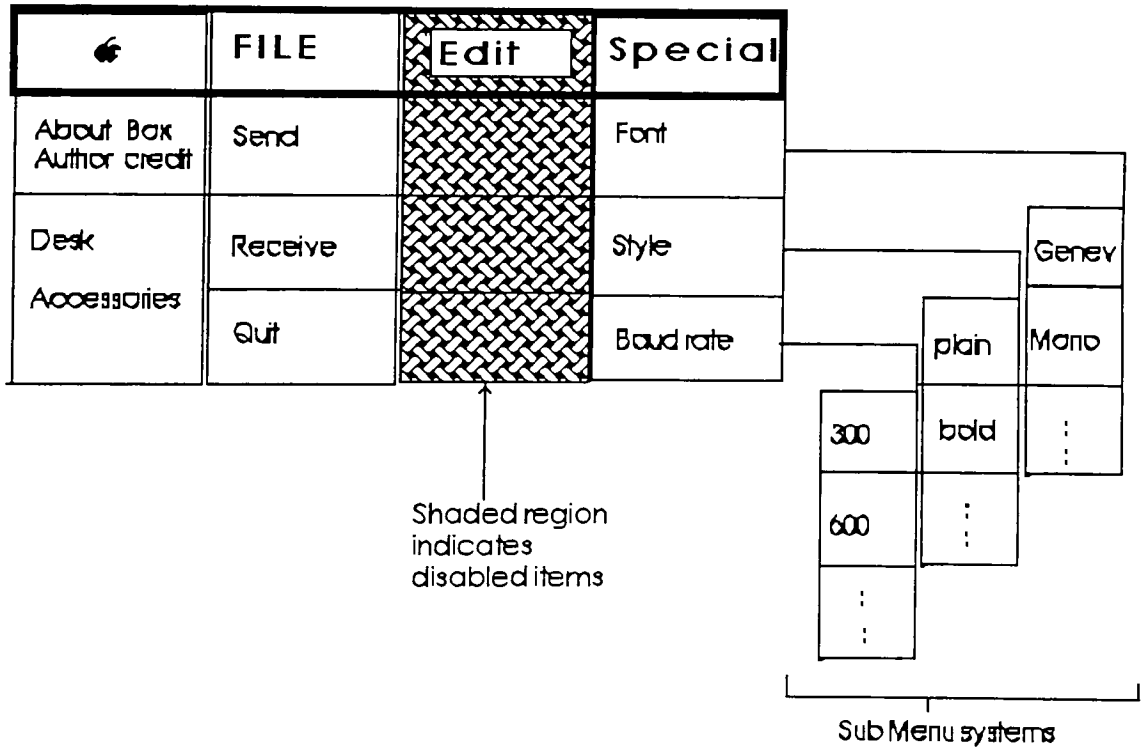


The window shown here is the type that appears on screen when transmission is taking place. Pressing the mouse in the quit region (called as the goAway region) will terminate the program. When the mouse is pressed in the grow Box region the shaded region appears, and by dragging the window on the screen it is possible to re-size the window.

When this happens printing in the screen gets adjusted to the screens lower bounds. If the mouse were pressed in the window tittle region and dragged the window could move around the screen. The window drag rectangle has been set so that one cannot go beyond the screen bounds.

4. 4. 2 The Menu Bar.

In a nutshell the menu bar has four items. The apple menu, which contains a credits item (it displays a dialog box indicating the author of the program) and different desk accessories.

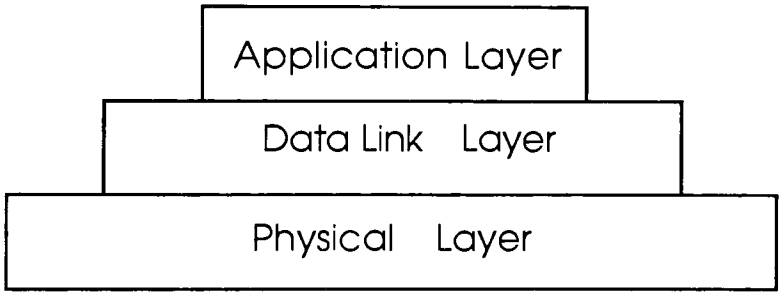


The File menu contains three items , send receive and quit. The edit menu which is disabled but must exist as certain desk accessories must require its existence in the menu bar. Finally a special menu which contains 3 sub menus, one for font, one for style and one for Baud rate.

CHAPTER 5
SOFTWARE IMPLEMENTATION

5.0.0 Introduction.

This chapter will deal with the implementation of the design as stipulated in chapter 3. Chapter 4 also plays an important role in the design and flow of the program. This chapter shows and explains certain implementation details which were discussed in chapter 3 and explains why they were used. The program can be represented as a 3 layered protocol as shown in the figure.



The physical layer, implements all the hardware requirments as explained in chapter 2. The datalink layer implements the sofware protocols as explained in chapter 3. The top most layer could be called a presentation layer. This layer is responsible for handling the cuncurrent eviornment and for handling the handshake procedures. Within this program most of the datalink layer exists in the modules receiver.c and sender.c while most of the presentation layer exists in the module display.c.

5. 1. 0 The program structure.

Before we get down to bits and bytes let us take a minute to try and understand the program structure. This will aid in following and understanding the program. This program has been broken up into 5 main modules.

ModiTalk.c : Contains the main loop and also deals with initialization routines.

Procedures.c : Handles procedures used by both the sender and receiver, it also contains the event loop as well as handshaking procedures.

Display.c : Handles the Macintosh environment, windows, menu bars, etc.

Receiver.c : Deals with all routines of the Receiver.

Sender.c : Deals with all routines of the sender.

As there is a concurrent mode supported by this program there are numerous flags and numbers to deal with. Hence instead of having to declare them in each module they have been placed in two separate header files, decl.h & extern.h. All constant declarations are placed in a file Header.h. Also as most of the resources were built using ResEdit there is no distinctive way of showing what type of resources were built. However the file ModiTalk.r contains a machine dump of the resource. Though the machine dump itself is unintelligible to the average user it none the less explicitly states what each section contains giving the user an idea of what was built in the resource.

5. 1. 1 The Program Flow

Before going into details let us understand the basic flow of the program. The main loop contains only 3 items which it goes through till the end.

```
while( myDone == FALSE )
{
        if(finishedReading == TRUE && finishedWriting == TRUE)
                transOver();
        else
        {
                GetNextEvent( eventMask,  &myEvent  );
                EventHandler(&myEvent);
                getblock();
        }
}
```

The way events are retrieved from the event queue is by a call to the tool box routine GetNextEvent or WaitNextEvent. I have chosen to implement GetNextEvent as WaitNextEvent was not existent on older system versions. GetNextEvent accepts two parameters. 1) an event mask and 2) an event record. The structure of this record was discussed in section 4.2.0 in the last chapter. Event mask lets you mask out all events except for the ones specified.

For example

GetNextEvent(mouseDown | keyBoard, &everyEvent)

In this case all events except mouseDown & keyboard events would be masked out. This event record obtained is passed on to my routine "EventHandler". This routine located in procedures.c handles all types of events generated. Control is passed on to routines which handle particular events, most of which are located in the module display.c. A detailed discussion of how this takes place will follow later. The last statement of the loop is a call to the procedure getBlock. This is the entry point to the protocol. Getblock checks to see if the handshake is done. If it is not done then it goes to handle the handshake. If the handshake has taken place it checks to see if the terminal is a sender or receiver and on that basis handles routines for their respective types. Hence in a nut shell the flow looks like the diagram shown here.

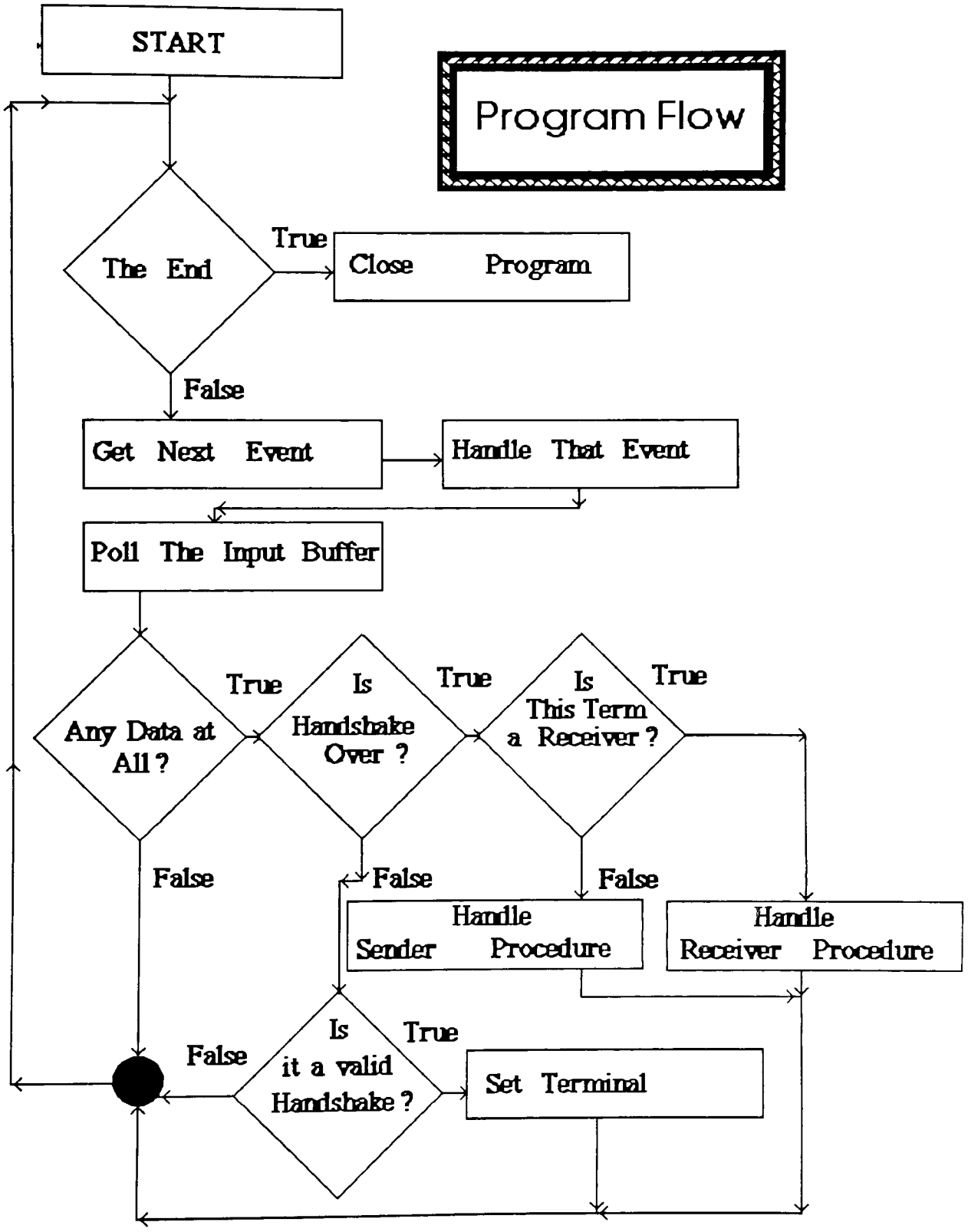

START
Program Flow
The End
True
Close Program
False
Get Next Event
Handle That Event
Poll The Input Buffer
Any Data at All ?
True
Is Handshake Over ?
True
Is This Term a Receiver ?
True
False
False
False
Handle Sender Procedure
Handle Receiver Procedure
Is it a valid Handshake ?
True
False
Set Terminal

Figure 5. 1

5. 1. 2 Initialization.

Most of the initialization takes place in the module ModiTalk.c. The first calls are to initialize the tool box calls, for example InitGraf (&thePort). This call initializes the grafPort which is a complete drawing environment that defines where and how graphic operations take place. Similarly there are calls to initialize window, menus, dialogs, cursor, fonts, menu bars etc.

Window Init is a procedure that displays the window whose dimensions are in the resource. At first the system opens up two windows. The first window is the data window and the second but smaller window is the handshake window.

```
WindowInit()
{
        myWindow = GetNewWindow( BASE_RES_ID,NIL_POINTER,MOVE_TO_FRONT );
        myHandShakeWind = GetNewWindow( BASE_RES_ID + 2, NIL_POINTER,
                                                          MOVE_TO_FRONT );
        ShowWindow( myWindow);
        ShowWindow( myHandShakeWind);
        SetPort( myHandShakeWind);
        MoveTo( HORIZONTAL_PIXEL,VERTICAL_PIXEL );

}
```

As explained in the last chapter a call to GetNewWindow (BASE_RES_ID,..... would get data for a window from a resource whose ID was = BASE_RES_ID. A call to Show Window will show the window created and a call to SetPort would activate the current window. All data related to the handshaking procedures would be displayed in the handshake window.

The reason for having 2 windows opened is not only for display purposes. Under the Macintosh programming enviornment, whenever a window is re-sized the entire window has to be redrawn. Hence all data stored in that window has to be placed in a linked list which has to be constantly updated.This starts building an overhead you would like to avoid. Instead it would be a lot easier to implement it so that the handshake display takes place in another window and when over, the window is closed. The data display is now displayed in another window. This is because the data display is fixed and does not requireback up which would need to be updated. A call to update the window would be able to redraw the window with the minimum difficulty. This call shown below is in the module procedure.c.

```
HandleUpdateEvent()
{
        if(myTermIs.readyToStart)
        {
                if(SENDER_TERM)
                {
                        HandleSenderScreen();
                        PrintSenderScreen();
                }
                else
                {
                        HandleReceiverScreen();
                        PrintReceiverScreen();
                }
        }
}
```

A call to InitBuffers initializes certain boolean values allocates the required buffer space,initializes the buffer space and calculates the time out period. As explained in chapter 3 , the time out period could have other factors associated with it, and was given as :

timeOut = ((oneBuffTimeOut + propagationDelay) * numForTimeOut)
+ propagation delay.

However I have added one more factor to this called readTime. As a timer is started after every window is sent out there also exists a small amount of time required for File-IO. This File-Io time is dependent on 1) the processor, 2) if the IO is taking place from a hard drive or a floppy drive, 3) what version of system one is using and 4) what sort of background jobs are existent. The factor I have reached is 200 ticks of the clock, and this has been reached after a lot of trial and error methods taking into consideration the floppy drive, the hard drive as well as the data rates. Hence the time out period now stands at.

timeOut = ((oneBuffTimeOut + propagationDelay) * numForTimeOut)
+ propagation delay + readTime.

A last call that I will discuss here before we move onto initializing the serial port is the call to SetUpSizeRect (in ModiTalk.c) and SetUpDragRect (in display.c). Set UpSizeRect will return the current dimensions of the window taking the screen into consideration. This is important in a call to growWindow, as well as for scrolling the window in printing to the screen. Set UpDragRect will set up the drag area for the window depending on the screen bounds.

5. 1. 3 Initializing the serial port.

Applications in the Macintosh communicate with devices through the Device Manager. The Device Manager is part of the operating system that is responsible for communicating with devices . There are two main types of devices in the Macintosh 1) block read devices as in the case of disk drives and 2) character devices like the serial port, which read or write one byte at a time.

The Device Manager does not manipulate devices directly but instead calls device drivers to do the work. Hence device drivers are programs written which take data coming from the Device Manager and convert them into actions or visa versa.

There are 2 serial drivers that a programmer can make use of 1) ROM serial driver is the type I have used and 2) the RAM serial driver.

A call to open the device driver must be issued to be able to use it. Before a device driver is opened identification is done by the name. After it is opened it is identified by its reference number. There exist two serial ports in the Macintosh 1) the Modem Port and 2) the printer port. The printer port is use to communicate with slower devices like the printer. Hence I have used the modem port for my serial communications.

The driver names and reference number are :

As I am using the Modem port for communicating a call to open the port is made in this form

```
int             e;
        e  =  OpenDriver("\P.AOut", &PortA.refout);
        e  =  Open Driver("\P.AIn", &PortA.refIn);
```

This call to the tool box routine Open Driver will open the device specified by the name and return its reference number to the variable specified in the second parameter. Now any calls to use the driver can be made by specifying the variable holding that reference number. The variable refout and refin are part of the structure defined in the header decl.h and extern.h. These have been declared as part of a structure called PortA and also holds information on the data rate, parity, stopbits, and startbits.

It is defined as follows:

Hence now a call to read from the port will be :

FSRead(PortA.refin, &numberToBeRead, Pointer)

and a call to write will be :

FSWrite(PortA.refout, &numberToBeWritten, Pointer)

These calls are used extensively throughout the program. The data rate, parity, databits and stop bits default to 9600 bps, no parity, 8 databits & 1 stop bit. A call to Reset the serial driver to default to these settings is made by calling the procedure :

SerReset(PortA.refin,PortA.Baud | PortA.parity | PortA.stopbits | PortA.databits);

The Baud rates can be changed and can be chosen as any one of the following 300,600,1200,2400,3600,4800,7200,9600,19200 or 57600. The parity bit databits and stopbits cannot be changed in this implementation.

The serial drivers have input buffers associated with them. The size of this buffer is 64K bytes. This size can however be reset as is done in the case of this implementation. Both the input and the output buffers can be reset by stating the reference number.

5. 2. 0 Introduction to the protocol Implementation.

Consider what happens when the program control shifts to the procedure call getblock() (this procedure is located in module procedure.c).

```
getblock()
{
        long            whatread;

        SerGetBuf( PortA.refin, &whatread ); /*      anything      there?      */
        if( myTermIs.readyToStart == TRUE )
        {
                if(SENDER_TERM)                 /* if terminal is a sender   */
                {
                        if( nackReceived)
                                SendNackBuffer();
                        else if(( !finishedReading ) && ( ackIsReceived == TRUE ))
                                HandleReceiveMyAck();
                        else
                                DataAtSenderPort();
                }
                else if( RECEIVER_TERM )      /* if terminal is a receiver   */
                        if(( whatread > 0 ) && (myTermIs.readyToStart == TRUE ))
                                        HandleReceiveData(whatread);
                firstTimeAround = FALSE;
        }
        else if(whatread > 0)                   /* else if handshake not done */
                HandleHandshake();
}
```

serGetBuf(PortA.refin, &whatread) is a poll to the serial port's input buffer "inbuff". This procedure returns the number of characters that are in the input buffer through the variable whatread . If the terminals have not been configured (if they have been configured then myTermIs.readyToStart = True, and one acts as sender and one acts as receiver) the procedure shifts to the handshaking protocol.

You want to go in and receive this handshake from the other terminal only if the other terminal has sent something. Hence if whatread is greater than 0 then we know that there is something in the data port and we must go fetch it. If however whatread = 0 then we know that the input buffer is empty and we come back to our main loop.

To send a handshake signal we have configured the terminal from the menu "File". This menu has three items. Send, receive & quit. Choosing send will configure that terminal to act as a sender, similarly choosing receive will configure that terminal to act as receiver. In choosing these items we genarate an event (in this case a mouseDown) hence it is important that we return to the while loop constantly to handle these events. If however no mouse down event or any other event took place then the control would return back to getBlock().

5. 2. 1 The Handshaking Implementation

Let us assume that the other terminal (call it as terminal A) has sent a handshake indicating its desire to act as sender. This terminal (call it as terminal B) receives the handshake and buffers it in the serial input buffer. Now the control shifts to getblock it comes to the part

```
else if(whatread > 0)                    /* else if handshake not done */
        HandleHandshake();
```

This time whatread is greater than 0 hence control shifts to the procedure HandleHandshake. Now to prevent errors from taking place 'B' must check to see that this is a valid handshake signal else an error should be raised.

If it is a valid signal 'B' must accept this sender handshake and prevent errors from being generated. Handshake errors can be generated in the form of badly configured terminals.

Error 1) If both terminals were chosen to be the same configuration (ex: both are made to act as senders).

Error 2) A previously chosen sender now tries to reconfigure itself as a receiver (or visa versa).

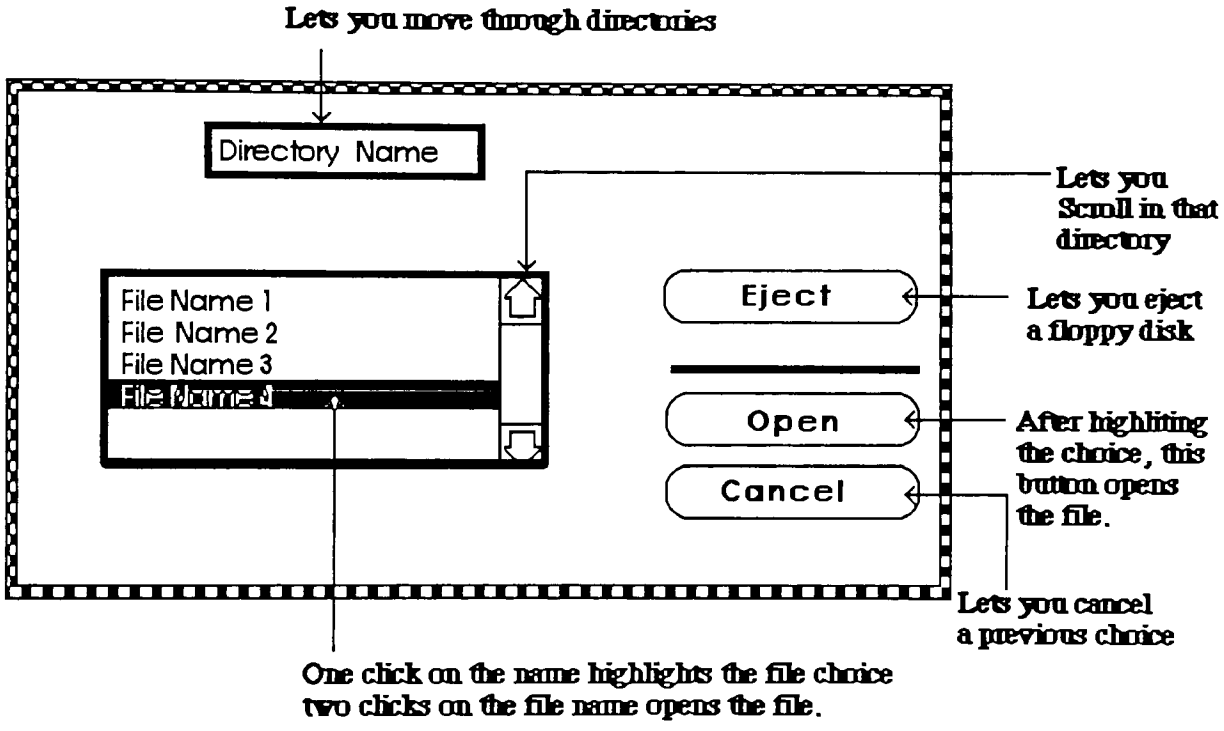
Error 1 is prevented by the terminal that receives the handshake. That is consider terminal 'B' receiving a sender handshake from terminal 'A'. 'B' accepts the handshake but disables the item 'send' in its menu bar. Now the user can only choose 'B' to act as a receiver.

However this gives rise to another problem.

What if both the terminals were to be configured to act as senders at the same time. Here 'A' sends to 'B' a signal indicating its desire to act as sender and 'B' sends to 'A' its desire to act as sender before it receives this signal from 'A'. At this stage both terminals pop alert boxes up indicating that the other terminal has also been chosen to act the same type. On seeing this alert box pop up on the screen one of the users now realizes that it has made a mistake now reconfigers its terminal to act as receiver. these alert boxes disappear when the mouse is pressed in a particular region in the box.

Error 2 is prevented by flashing the alert box indicating that that terminal has already been chosen to act as a sender.

On choosing the terminal to act as a sender a dialog box pops up which displays the current files in the directory. It has a little box on top which indicates the directory name. By going into this little box one can change the directory that they are in.



A receiver dialog box is similar with the exception that there is another box below the file choice box which lets you type in the name of your file. The name defaults to 'LASER'. Once the correct handshake is received by both the terminals a ready to begin signal is sent by both and the transmission begins.

5. 3. 0 The Receiver Protocol.

Now let us again come to the procedure call get block(). This time however The handshake has already taken place and the terminal 'B' is set to act as a sender. The main program control for the receiver resides in the main loop and not at the serial port. By this I mean that the receiver is constantly handling events, but the receiver keeps polling the input ports serial buffer for data. Say for example the receiver had a 100 characters of data of a large packet of 1023 in its input buffer when it polls. On seeing the data in the buffer the receiver will pull it out and place it in a separate buffer, but it will not wait at this point to receive the end of the packet. Instead it moves on to again handling events, or if in, case it has received the entire packet it will check its validity. Another point to remember here is that if the receiver has now pulled out 1000 characters of a 1023 packet size and then sees another 300 characters in the buffer it must at this stage pull out only 23 characters and not the entire 300 as that would cause an error in the packet size. Once it has pulled a packet out it goes in to check for byte stuffing and errors. The implementation also could have been such that, the receiver instead of pulling out the 100 characters of data could simply poll the port and see if the entire packet has come in and then, remove the entire packet in one sweep. This is definitely easier to implement then having to keep track of how many characters there are in the packet, how many have been pulled out and how many will be needed to pull out to finish the packet. However this has a draw back. Consider the case where the serial ports input buffer can hold 2000 characters and the transmission rate is 9600 bps. There are 1000 characters in the buffer which have arrived out of a packet of 1023 when the receiver has polled the port.

On seeing only 1000 character the receiver does not pick up the data but goes back into handling events. The user now wishing to re-size its screen, takes only 1 second to do so. As the receiver already has 1000 characters in the buffer and it is running at 9600 bps the next time it comes around to poll there will be 1200 extra characters come in. hence there should be 2200 characters in the buffer. As the buffer size was only 2000 bytes it would have corrupted some data. This case is particularly true at higher data rates. The ideal buffer size to use for this implementation to definitely prevent this situation from arising would of course be the size of one window + the size of "how many buffers would be filled before a time out takes place", as this would guarantee that a time out would occur before the input buffer was completely full. However the drawback here is that the number of buffers that would be filled before a time out occurs is dependent on a lot of factors and hence there is no fixed number that one can put a finger on. None the less the memory size would be, in this implementation itself, close to 20K bytes for the serial port's input buffer. Now not only is this a lot of memory to keep locked away doing nothing (in fact one of the Macintosh's did run out of Memory at one stage in my testing when I had assigned it a buffer size of 15 K) but it also causes a problem when a Nack is sent as the receiver has to either clean out 20 K of memory or go searching through 20 K of memory for a header. A good memory size that I have arrived at for this implementation is 6K bytes. For transmission of 9600 Baud the user has 5 seconds before its buffer will overrun, this 5 second wait is long enough for the user to either resize its window or change its font display or change its display style. This situation will hold even if the user tries to do multiple things like, re-size the window,change its display style and change its font.

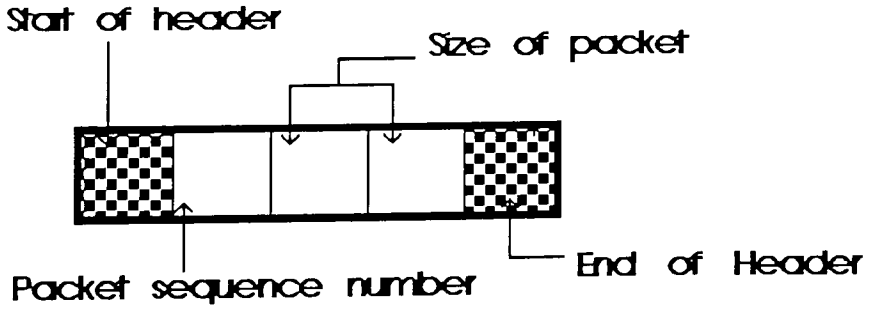
This is because events are handled one at a time and the receiver will poll the input buffer after every event.

If however the the user is going to use the desk accessories a time out could take place or a Nack could be generated. However this is unavoidable as there is no way of saying exactly how long a user would take in the accessory.

An important point to remember here is, as discussed in chapter 3, if a receiver time out were to be implemented as an efficiency measure then that timer would have to be blocked every time you handled an event. Otherwise it could generate a nack for the next packet when in reality the packet is actually sitting in the input buffer. It would not cause a fatal error but would result in the unnecessary re-transmission of a packet.

5. 3. 1 Receiving the packet.

Now that we have a good over all picture of the entire situation let us go in and see how the packet is received and checked. If a new packet is expected then a flag is set indicating that the next thing to receive will be a header. Hence as soon as a character comes in the receiver loops and pulls out 5 characters from the buffer. It pulls out Exactly 5 characters as this is the header size.



The first and last characters are the start of header marker and end of header marker which I have chosen as 0x07 (BELL). The second character chosen is the packet sequence number. As the packet sequence number should never be less than 0 or greater than 45 no more than 8 bits will ever be needed. The third item in the packet is the packet size. As the packet size could be more than 256 more than 8 bits are needed to keep track of its size. However as the sender reads in only 1023 characters of data no more than 2047 characters will ever be in a packet.

As there are 1023 data characters in a packet then, in the worst case if every data character had a byte stuffed, then there would be 2046 characters in the packet. There is also 1 end of packet character to bring the packet size up to 2047. Hence in this implementation no more than 16 bits would be need to represent it size of the packet.

Once the header is picked up its validity is confirmed. This is done in the following fashion. It first checks to see that there exists a start of header and a end of header in the in the correct positions. It then picks up the third and fourth character in the packet, concatenates them and gets the integer size out. The packet sequence number is then obtained and its validity is checked. It could raise a nack if the packet sequence is not the one it is expecting (it expects packet sequences to appear in order), or if the packet sequence number is not a sequence of the previous window. If not it will raise an error, not for the sequence number it has obtained, but for the sequence number it is expecting. If however it accepts the header and finds an error in the packet it can at this stage send an error for the sequence number it has obtained.

So as not to cause any confusion, the receiver updates the next packet to get only after the previous packet was written to file and not after obtaining the correct header.

Now the receiver goes ahead and obtains the packet.

As explained before it does this by getting to know how many characters there are in the buffer if there are enough it pulls them all out. I f not, it pulls out all it can and comes around another time to repeat the process. This is shown in the procedures 'HandleReceiveData()' and procedure 'ReadThePacket'. A brief description of how it happens is shown below. To distinguish the code from text the code is put in italic.

```
HandleReceiveData(whatread)
long int            whatread;
{
        unsigned    char        headerBuff(10);
        int         count;
        int                 GetHeader();
        int         e;
```

If we are in the middle of reading a packet then readCompPacket = FALSE if we have to read a new packet then we have to get the header and readCompPacket = TRUE.

```
        if(( readCompPacket == TRUE ))
        {
        this part gets the 5 characters of the header and now it goes to the procedure
        GetHeader from where the header is identified and from were the procedure
        CheckHeader checks the correct sequence Number. If there is an error in the
        sequence number it returns a   '-1 '.
        totCharInPacket = GetHeader(headerBuff);
                if( totCharInPacket != -1)
                {
                        if(receivedClosing == TRUE)
                        {
                        This indicates that we received a closing handshake in the
                        header.
                                finishedReading = TRUE;
                                finishedWriting = TRUE;
                        }
                        else
                        {
                        Go to procedure ReadThePacket and  get the Packet.
                                readCompPacket = FALSE;
                                dataCharReceived = 0;
                                ReadThePacket();
                        }
                }
```

```
            else
            {
            There was a problem in the header so send a nack of the packet you
            are expecting ( readBuffNum)
                    readCompPacket = TRUE;
                    SendNackSender(readBuffNum);
            }
    }
    if you were in the middle of reading a packet but the last time around you did not
    have enough characters to complete the packet then control would come
    down here to this pointt when you came around to the procedure
    HandlleReceiveData.
    else if(( readCompPacket == FALSE ))
            ReadThePacket();
}


ReadThePacket()
{
    int             e,i,j,charRecv,temp,goodPacket;
    long            whatread;
    unsigned  char          tempBuff(2000);

    Poll the port and see how many characters are there.
    SerGetBuf(PortA.refin, &whatread);

    calculate to see whether the number of characters already obtained in the
    previous visit to this procedure + the new characters in the buffer is greater than
    the packet size received. (if this is the first time around into this packet then
    dataChar received = 0 else datacharReceived = charRecv of the previous visit).
    If the number of characters is greater than the packet size then get only the
    packet size out " whatread = totCharInPacket - (dataCharReceived -1)".

    charRecv = dataCharReceived + (int)whatread;
    if( charRecv >= totCharInPacket)
    {
            whatread = (long)(totCharInPacket - (dataCharReceived -1));
            readCompPacket = TRUE;
    }

    e = FSRead(PortA.refin, &whatread, tempBuff);
    temp = dataCharReceived;

    for(i = 0; i <= (int)whatread; i++)
            checkBuff(temp++) = tempBuff(i);
    dataCharReceived = charRecv;
```

Now we come to the part where we check the packet to see if it is a valid packet or not. If the packet is a repetition of a previous buffer then in the procedure CheckHeader skipPacket is Set to TRUE so that the packet is discarded. If skip packet is FALSE then it goes to the procedure Check packet. to check if the packet is correctly received in terms of the number of characters. Also all the byte stuffing is removed in that procedure. If all is well it returns a TRUE and the packet gets copied to the file.

```
if(readCompPacket == TRUE)
{
        dataCharReceived = 0;
        if( skipPacket == TRUE )
                SkipThisPacket();
        else
        {
                goodPacket = CheckPacket();
                if( goodPacket)
                {
                        SendToOutputFile();
```

if the packet number is the one for which an ack is to be sent then send an ack and calculate at what packet number the next ack should be sent. This is done in procedure AcknowledgeOldAck.

```
                        if( pacSeqNum == sendAckAt)
                                AcknowledgeOldWindow();
                        GetMyNextBufferNumber();
                }
                else if(goodPacket == 0)
                {
```

if there is an error within the packet then control comes to this point and raises an error.

```
                        dataCharReceived = 0;
                        SendNackSender(pacSeqNum)
                }
        }
    }
}
```

As explained above, when the entire packet is read in it first checks to see if this packet is a previously re-transmitted packet. If it is, it simply skips it and goes out to the main loop and gets the next one in. This will happen in the case where the ack or Nack got lost and the entire window got re-transmitted.

However on receiving the last packet of the re-transmitted window the receiver will send out an acknowledgment for that window, else the sender will time out again and re-transmit the same window again and again. Hence the system will deadlock. if however it receives a new packet, then the call to the routine goodPacket = CheckPacket() will return a True or False. CheckPacket 1) removes the byte stuffing. If there is an error in the byte stuffing it returns a False. 2) counts the number of actual data characters. 3) checks to see if the last character is a end of packet marker. 4) checks to see if this is the last packet (it can identify this by a special header the sender sends), if it is not then the receiver checks to see that there are 1023 characters.

If the packet goes through all these tests without returning a zero then the packet gets written to file. Next it checks to see if this packet was the last one in the present working window. If it is then it sends an ack for the packet and starts its new window from the next packet. It then updates the next packet sequence number to expect and then breaks back in the main loop to get the next header.

5. 3. 2 Receiver ack & nacks.

Acks & Nacks are sent through the serial port. They consist of two bytes. The first byte consists of the packet sequence number for which the N/Ack is being sent for. The second byte indicates if it is a nack or ack. When an ack is sent the next buffer to expect is the present buffer + 1. However let us examine what happens when a nack is sent. when a packet is corrupted or the header is corrupted a nack is generated. If the packet is corrupted then the next packet to receive is the same packet number.

If the header is corrupted then the next packet to receive will be the next packet it expects. Let us consider that the sender has sent 4 packets.

The receiver has received the first one and realizes that there is an error so it sends a nack for packet 1. The sender in the mean while will go on sending packets till it gets a chance to poll its input buffer and check to see if there is any data there. The way this has been implemented is that the the sender will check it port's input buffer only after sending a packet (if it is in the process of sending packets). Hence the sender will at most send out one more packet before it realizes that a nack is at the port and that it has to retransmit that buffer. By the time the receiver realizes that the packet it received is bad and sends a nack the sender is in the process of sending packet 3. So the receiver has at least 2048 characters of data in it input buffer to go through before it can find the header of the re-transmitted packet. Instead of doing this the receiver waits for a time period = the time it would take to transmit a packet. It then clears the serial ports input buffer and then then starts the polling process again.

The sender on the other hand, on receiving a nack does not retransmit the packet immediately. It instead waits for a time period greater than the transmit time for one packet and then re-transmits the packet. I however have two procedures in the module receiver.c called as GetThisPacket & getThePacSize. These two procedures would find the packet header for you in the input port's buffer. However I found that clearing the buffer was more efficient. I feel this process of trying to search the header takes too long and if the port's buffer was less than 2000 bytes then there was a tendency to overflow as the data was coming in faster than the receiver could process.

If we do not want to wait for a whole buffer time out then we could poll the input buffer of the sender after every character . Though this will not take too much effort to implement it will slow down the process, so much so that it will destroy the very meaning of high speed transmission.

Now let us move on and try understand the senders implementation.

5.4.0 The Sender Implementation.

The implementation here has been made even more complicated by using a floating window scheme rather than a fixed window scheme. What this means is that there is no way of saying that a window will always contain a fixed set of sequence numbers. Though this maintains an idea that every window must be of the same size the implementation gets complicted when Nacks arrive.

In the case of the receiver, the receiver mainly starts all it routines in the procedure call getblock(). The sender starts it first call in the routine HandleBegin in the module procedures.c.

```
HandleBegin()
{
        int tempToSend;

        StartingDeclarations();
```

starting declarations are just some variables initializazed to values, ex start window from 0, last buffer sent = 0... etc, as there were many declarations of this nature I deceided to move it into another procedure to help make this procedure more readable.

An explanation of what happens now is given after the code.

```
        if(SENDER_TERM == TRUE)
        {
                ReadIntoBuffer(readNext,29);
                readNext = 30;
                if(!finishedReading)
                {
                        SendFileModem(0,WINDOW_LENGTH);
                        if(nackReceived == FALSE)
                                SendFileModem(15,WINDOW_LENGTH);
                }
```

```
                else
                {
                        ackToReceive = lastBuffer;
                        if(ackToReceive > WINDOW_LENGTH)
                        {
                                tempToSend = lastBuffer - WINDOW_LENGTH - 1;
                                SendFileModem(0,WINDOW_LENGTH);
                                if(nackReceived == FALSE)
                                        SendFileModem(15,tempToSend);
                        }
                        else
                                SendFileModem(0,lastBuffer);

                        if(nackReceived == FALSE)
                        {
                                SendHeader((int)NULL_SEQ,(int)END_HANDSHAKE);
                                lastPacketSent = TRUE;
                        }
                }
        }

}
```

Here the sender first reads 30 buffers from the file it has to send. Thirty buffers (0..29) constitute up to two window frames. The procedure ReadIntoBuffers(From, number of Buffers) is a procedure that accepts a starting number to read from and the number of buffers that have to be read. The reason I am not sending the ending sequence number is because I want this procedure to handle the case when the sequence numbers reapet themselves. By this I mean :

Consider the case when there was no floating window concept. We could now definitely say that the windows will be 0..14, 15..29, and 30..44. Hence when reading the buffers we could have something that could say to the effect

```
for ( presentBuffer = BufferFrom, presentBuffer <= bufferTo, presentBuffer ++)
        Go On reading;
```

However this case will not hold in the floating window concept because it is very possible that the "bufferTo" is less than the "Buffer From" sequence number.

example : if while sending buffers 15..29 a Nack took place at 27. Now the window extends from 27..41. The next window will now be from 42..11. Hence BufferTo is less than BufferFrom and so the above for statement would cause an error. Hence the ReadIntoBuffer routines contain two parameters 1) the BufferFrom & the number of buffer which is always 14 (14 as the bufferFrom is also counted).

Hence comming back, the next thing we do is initialize the next buffer to send (readNext = 30). We now send off the buffers from the serial port by the call to procedure SendFileModem(bufferFrom, NumberOfBuffers). If the file transmitted without any problems then the Next set of 15 buffers is sent. What is hidden from the code right now is what happens were a Nack to arrive. We will come to that a little later.

Now however what happens if the file was very small and did not consist of 30 buffers. The ReadIntoBuffers routine then sets a flag "finishedReading = TRUE" and returns the number of buffers read. Hence we first check to see if the file is now completly read by the Boolean variable "finishedReading". If finished Reading = FALSE we go ahead with the scenario described above. If however finishedReading = TRUE, we now have to see which window the last buffer was read into. This is important as we need to know which ack to receive before sending a closing handshake. Hence if the last buffer read was in the window 0..14 (as an example say it was packet #7) then the closing handshake will be sent after packet # 7. The receiver in the mean while is expecting packets numbers 0..14 and is totally ignorant of the fact that the sender has to send only 7 packets. So instead of receiving the 8 $\underline{th}$ packet the receiver now receives a closing handshake.

It realizes at this stage that the file is over and sends a closing handshake back to the sender. The sender waits for this handshake to come in. If it does not come in during the expected time period it times out and retransmits the entire buffer again as it would mean the same thing as loosing an ack for 7. If the closing handshake comes in from the receiver then the file transmission is over.

Now Consider the case where the last packet is between sequence numbers 15..29 (as an example let packet #24 be the last packet) then the sender must send out the first window(0..14) start the timer running and then send out the second window. This is because of the way the timer is set up to activate. The time out timer will kick off once the last packet in a window is sent. Hence one has to take care in seeing that no more than 15 buffers are sent out through the serial port by the call SendFileModem. Therefore the call to send 24 packets would be in this form.

```
SendFileModem(0,14);
if(nackReceived == FALSE)
        SendFileModem(15,10) ;          /* as 15 +10 = 24 both inclusive */
```

5. 4.1 Some important Procedures in the Sender.

I would briefly like to describe 3 important procedures here which would be relevent to the understanding of the implementation of the timer and the Nacks. They are 1) ReadIntoBuffers, 2) SendFileModem, 3)DataAtSenderPort.

Let us start with ReadIntoBuffers().

This procedure call has 2 paramaters X and Y, where X is the buffer number you would like to read from and Y is the number of buffers to be read. Under most cases the value for Y would be 15.

However the first time around the value of Y would be 29. This Y value is the number of buffers to be read from the buffer number represented by X. It would be better explained by this example; condider the case where X is = 35 and Y =15.

Hence we want to read sequence numbers 35..44, 0..4.

The reading takes place into an array of pointers given as char *buffer(count).

Hence in this case count will go from 35..44 & 0..4. Another pointer called bufferPtr keeps track of the location of the array index (count).

A character is read in one at time by incrementing the bufferPtr for that index. When the bufferPtr has read in 1023 characters for that index it increments its index number and starts over.

```
bufferPtr = bufferCount (count )
for ( i = 0; i <= 1022; i++)
        *( bufferPtr  + i ) = get next character from file;
when finished reading the packet put an end of packet marker (EOP)
*(bufferPtr  +  ++i ) = End of packet marker
go back to while loop
```

This will read in the required number of buffers and also handle the change in sequence numbers. However as each character is read in a track is kept of what the chracter is and the number of characters read in to each packet. If a character matched the EOP character then another EOP character would be placed in after it and the packet size would be incremented by 1.

Another thing that is kept track of is the last packet sequence that was read into. This will play an important role in the implementation for negative acknowledgements. If the end of file marker was encountered then a flag would be set and the number of buffers read would be kept track off.

The procedure SendFileModem is very similar to the procedure ReadIntoBuffrer when it comes to Handling sequence numbers. Before the packet is sent the 5 byte header is generated using the packet sequence number and the packet size for that sequence number. Now an entire Buffer is specifed for output to the serial driver. After sending the entire packet the input port is polled to see if ther are any acknowledgenments out there. If an ack came in then it would set the flag for ackReceived. if however a Nack came in then it would break out from the while loop and go in to handle the nack.

This will be explaind in more detail later. The way the input port is polled after every packet is sent out is explained next in the procedure DataAtSenderPort. When the last packet is sent a timer is started and a flag indicating that it has started is set (startTimer). This timer will time out if the ack or nack is not received within the specified time out period.

The procedure DataAtSenderPort is called from two places. 1) From the procedure SendFileModem and from 2) the procedure getBlock. On entering the procedure a check is made to see if the timer flag is set. If it is set then it checks to make sure that a time out has not taken place. If a time out has taken place then it generates a time out in the same form as a Nack but with a sequence number of the start of the previous window. If a time out has not taken place then a check is made to see if it exists any data at the sender 's serial port's input buffer.

If there exists 2 bytes of data there then it is pulled out. Now it checks to see if the sender has finished reading the file or not. If it has not finished reading and if there is an ack for the sequence number it expects, then this sequence number is now stored as the old ack Received, and the ack to receive gets updated by 15 packets. A flag is set to indicate that the sender received the correct ack. The next time the program control shifts into the procedure getBlock the sender checks to see if the flag is set or not. If it is then it goes ahead and reads the next set of buffers. Another important thing to remember is that it will put the timer off once it receives the correct ack.

If however it comes into the procedure DataAtSender Port and it has finished reading then the ack to receive now becomes the last buffer read. It however does not set the flag, hence when program control shifts into the procedure getBlock the next time around it does not go in the procedure to attempt to read the next set of buffers. Instead it simply polls the port to see if anything has arrived.

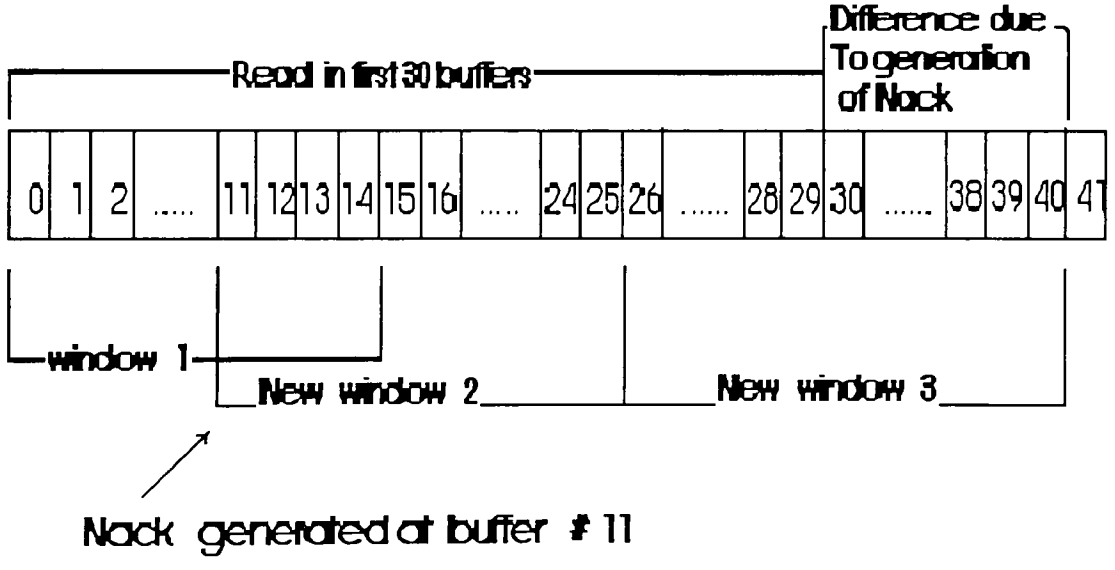
If it were to get a closing handshake then it knows that all the buffers were received correctly. If a wrong sequence number was received then the ack/nack would be ignored and eventually the sender would time out and retransmit.

If however a Nack was received then a flag would be set to say that a nack was received and the packet number for which it was received would be stored. Now all procedures that see this flag would end grasefully and give priority to handling the nack . This will be explained in the section 5.2.5.

As explained above the first 30 (0..29) buffers are read in. It then sends the first 15 packets with a call to SendFileModem(0,14). As soon as the sequence number 14 is sent out the timer is activated. On having received no nacks for packets (0..14) the sender moves on to send its next set of buffers with a call to SendFileModem(15,29).

5.4.2 The Nacks And The Time Outs.

To better understand what happens whenever a Nack is generated let us follow through with an example and a small diagram. As mentioned in chapter 3 whenever a nack is generated for a particular packet it is an automatic acknowledgement for the packets which have preceeded it. Let us consider the above mentioned example where a file of 54 K bytes is to be sent across. After reading in 30 buffers(0..29) a nack gets generated for 11. This is shown in the following diagram.



Let the sender be sending packet (0..14) through the serial port. Every time the sender sends out a packet it checks its input port's buffer for data. Nothing comes in until packet 10 is sent. Packet 11 however had an error and the receiver sent a Nack to the sender. The sender on finishing transmission of packet 11 (or packet 12 depending on when the nack was generated) now finds a Nack in its input buffer.

It picks it up and sets a flag indicating that it has received a Nack. It stores the Nack's sequence number and then returns back to the procedure it was last in (this was in procedure SendFileModem). This time however it does not carry on with the transmission as the nack was generated but insted breaks out of the loop and comes to the main loop to handle the next event. It then goes back into procedure getBlock. It now sees that the nack Rceived Flag is set and hence goes into the procedure SendNackBuffer. This procedure starts the sender window from the sequence number that the nack was received. It then needs to send 15 buffers and then another 15 (as it has to maintain a sliding window protocol). The sender now checks to see if it has enough buffers to send. It needs to send buffers (11..25). As it had read up to buffer 29 it sees that it has enough and goes ahead with the transmission. It has also updated all it variables to accept this new window, (ex : ack To Receive = 25, old ack received = 10 etc). However now when it sends its next window it is going to fall short of buffers. Sending out the second window is exactly in the same form as sending out a window if an ack was received. In fact the program was designed in such a way that this procedure "HandleReceiveMyAck" could be called from the procedure which handles the Nacks for sending out this second window. As the second window consists of sequence numbers (26..40) it is now 11 buffers short as it has read in only up to buffer #29. Hence the procedure routine ReadInDifference is called to read in the extra 11 Buffers.

Once the extra 11 buffers are read in the buffers are transmitted out from the serial port. If in case a Nack was generated for sequence number 28 instead of sequence number 11. The ReadInDifference routine will be called for this first window as it is falling short of buffers. Hence buffers (30..42) would be read in first then transmitted and then buffers (43..44 & 0..8) would be readin next.

Whenever an ack or Nack is not received for a particular window within a specified time period a timeout takes place. The previous window has to be retransmitted. This retransmission takes place in the same form as a generation of a Nack for the packet sequence number equel to the buffer number of the start of the previous window.. As there are are 3 windows before the sequence numbers repeat themselves there is always that 1 window gap before the sequence numbers repeat, hence the chance of sending a packet of the same working window is elemenated.

5.5.0 Conclusions.

A lot of testing has been done on this implementation and it seems to hold out fine. I do however have some suggestions for improvement in both the hardware as well as the software which are discussed in chapter 6.

Chapter 6
Recommendations and Conclusions

6.0.0 Introduction.

One of the drawbacks in this implementation was the quality of the equipment used. Most of the equipment was put together on the basis of availability and not on the basis of improving the transmissions. The highest data rate that I reached successfully without any errors was only 7200 bps. This is of course is in no comparison to the data rate a laser can in fact achieve. However it did demonstrate that the free space laser can be used as a carrier for transmitting data.

Some of the draw backs in the hardware were 1) The receptor diode 2) The need to switch between 0 and 250 Volts every time a 0 bit turned to a 1 bit 3) The lack of good aligning equipment 4) open unshielded wiring and the use of bread boards for building this circuitry.

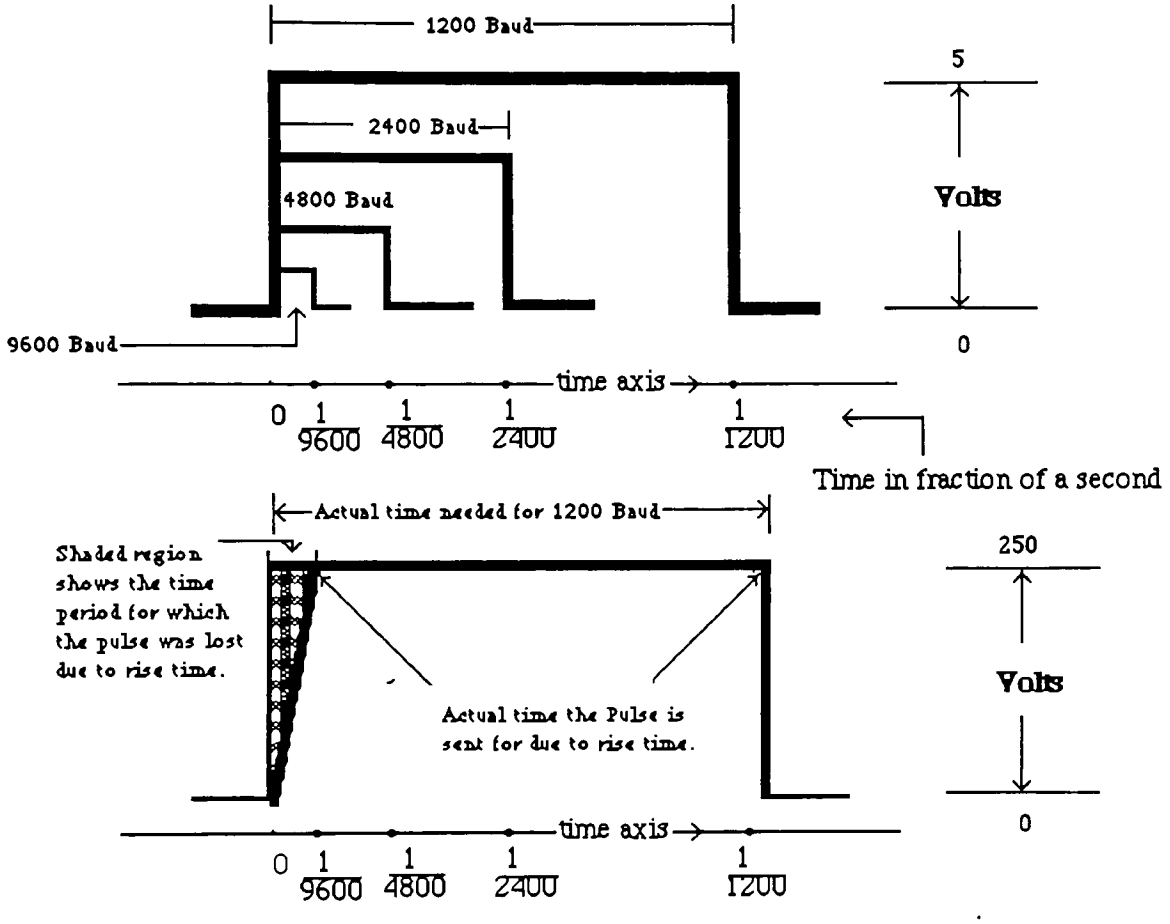
The two factors I would say that stopped the transmission of at least 100000 BPS were the first three cases. The pascal switch (appendix G) I was using to receive the transmission would behave sporadically and caused a lot of problems. This is possibly due to the low quality of the receptor diode.

It was very difficult to align the beam to the right position on the diode. A small fraction of a centimeter movement in any direction would cause misalignment . This could be caused by moving the Pockel cell, the laser source or the pascal switch.

Hence there was a need to mount all this on to equipment that would grip the Pockel cell, the laser source and the pascal switch firmly and yet allow the flexibility to move it within micrometer accuracy. Though this is easily obtainable it could cost a lot to implement and was unavailable to me at that time. Open unshielded wiring would cause an extra long rise time on the system. It would make a noticeable change only in very high transmission rates.

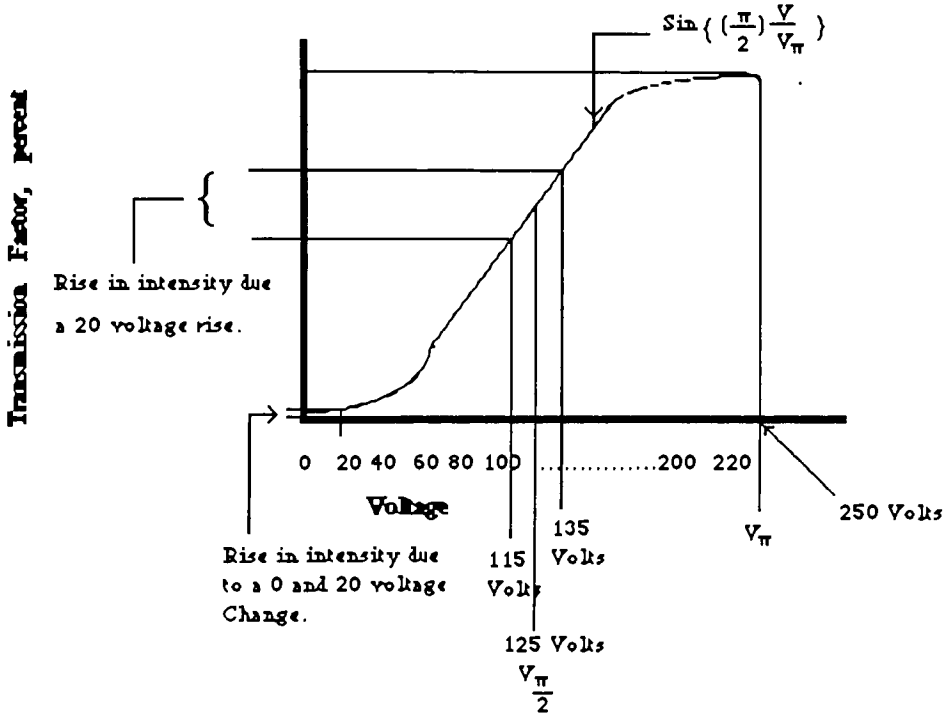
6. 1. 0 Improving the rise time.

A major contributor to the slower rate is the rise time involved in the voltage thrown to the Pockel cell. This rise time was generated by the circuit which would accept the differential output from the sender and throw either 0 or 250 volts into the pockel cell. This rise time was there because of the components & the bread board. Figure 6.0 shown on the following page is a diagram of the effect of rise time on 1200, 2400, 4800, 9600 bps transmission. From the diagram we can see the effect the rise time can have on 9600 bps transmission. Hence we need some way in reducing the rise time for our equipment. This is explained below.



From the diagram we can see that if instead of throwing 250 volts, if we could throw only say 20 volts our rise time could be cut drastically. The Pockal cell works in a way proportional to the voltage put in. Hence the more the voltage the greater the rotation of the electric field, till it finally reaches 250 V where the electric field is rotated through 90 degrees. Hence even feeding in 20 volts to the pockel cell does have much of an effect on the laser beam going through. Alternatively, we could make the receptor sensitive to this 20 volt change.

Unlike the set up we have now where a 0 bit is represented by maximizing the intensity and a 1 bit is obtained by a complete 90 degree rotation of the electric field thereby cutting the beam off completely, we will now be monitoring the intensity of the beam. The question is how accurately can we monitor the intensity. Let us look at what a graph of the intensity to the applied voltage would look like.



From the graph we see that due to the nonlinearity of the graph the 0 .. 20 volt rise will cause a negligible rise in intensity and would be very difficult to bias the receptor circuitry to monitor this small change in intensity. So the solution lies in trying to find a linear portion of the graph from where we could operate with a Plus or Minus 10 Volt range and yet have a large change in intensity. To reach a linear portion we need to rotate the electric field by 45 degrees. This can be achieved by putting a quarter wave plate between the crystal and the analyzer.

Now we have reached the quarter wave voltage. Hence the receptor should in theory be biased to receive the intensity required at the 125 volt mark. Now a 10 Volt rise or drop causes a reasonable large intensity change. Hence due to this 20 volt operating margin we can now reduce rise time considerable.

6. 2. 0 Recommendations in the software implementation.

6. 2. 1 Floating to fixed windows.

The software protocol implementation described in chapter 5 held out over the most rigorous tests. The one draw back however was the complexity of the code. The idea of the floating window seems like a good concept to implement and under most conditions it probably is. The draw back here however is, the complex code that has to be written to handle this. One of the factors that contributed to this complexity was when the nack was generated and it had to send out 15 buffers while it had only read in 10. Then it would have to calculate the number of buffers to send, read those in, and then send them. This however would not happen in all cases so code had to be built to check this. Another factor that contributed to a lot of complex code was that due to the floating window , one could never say where the window started from or where it would end. Similarly one could not place a finger on the start of the previous window. A window therefore could lie between sequence numbers which could wrap around. Hence extreme caution had to be used when using 'while' loops or 'for' loops as a condition in that loop would suddenly be satisfied, when in reality it is an error as the sequence numbers wrapped around.

Instead of increasing the complexity of the code by having a floating window I propose we maintain the receiving of acks after every 15 packets but have a fixed sequence numbers for window bounds.

We will have for example :

Window 1 Sequence numbers 0..14

window 2 Sequence numbers 15...29

window 3 Sequence numbers 29...44

Now when we read in a buffer of 15 (0..14) and a nack is generated for packet 11. Then instead of sending buffer 11..25 it will simply send buffers 11..14 and then expect an ack fro 14. This eases the implementation a lot and reduces the chances of errors taking place due to complexity. Now there is no need for calculating the difference of buffers to be read as there is always enough buffers read in. Also all window bounds are fixed hence the receiver knows when to send an ack and the receiver knows when to expect one. Now while and for statements can be used without fear of knowing that the sequence numbers may have wrapped around and hence they will cause errors.
The only draw back to this implementation is that whenever a nack is generated then on retransmission exactly one extra ack has to be sent. This I feel is however a small price to pay for having to reduce the complexity of the program code.

6. 2. 2 Closing the transmission and receiver time outs.

The way I have the system set up right now is that as soon as the file transfer takes place successfully the program quits. It would be better if the program would instead loop around and get ready to send or receive another file. This would save time in having to restart the whole program for sending or receiving another file.

Receiver time outs could also be implemented as an efficiency measure. This way the sender does not have to send an entire window before he realizes that an ack or nack was lost and has to retransmit again. However if receiver time out was implemented, then care should be taken to see that the input port is polled first, before the timer is checked for a time out, even though it may be way past the time out period. This is because the receiver may have been handling the environment and hence not had a chance to poll his port to check for data, when the time out took place.

6.3.0 Conclusions.

Free space laser communications is inevitable. This can be seen from the amount of research work that has suddenly started in this field. Some time back free space laser communications was looked upon only for areas of satellite communications and missile guidance systems. The commercial industry has suddenly shown great interest in this field. Most of the interest has been generated due to 1) high transmission rates, 2) very low B.E.R. (bit error ratio), 3) The ability to communicate through water and 4) the ease with which it could be set up (if the proper equipment was available).

A lot of industries like Bell Communications Research, Red Bank, New Jersey,have been looking at free space laser communications as a replacement for the optical fibre in areas where it is not possible to lay a cable, for example in a metropolitan area where a cable is needed between two adjacent buildings separated by a public road. The cost factor to lay a single subterranean cable between these two buildings could be prohibitive. Hence a free space laser communication system could be the answer.

A lot of research is in progress right now to study the feasibility of such a system and the outcome looks promising, in fact in this test conducted in New Jersey between two buildings separated by a distance of 6700 feet for a whole month showed very small B.E.R. values. This has built new hope and the system has been dubbed as "Quick start" broadband links.

The idea is also to use this free space laser link also in

1) interconnection of networks separated by, obstructions, highways, rivers etc.

2) immediate restoration of failed broadband links.

3) alternatives to microwave radio links where there is inadequate spectrum or there is a lot of congestion.

4) for temporary or special connection of LAN's (local area networks) to MAN's (metropolitan area networks).

None the less all this research and progress has given new hope to a faster and more reliable way of communicating data. We are still far from a dedicated commercial system, but we are now almost guaranteed of its existence as a future form of communication.

Bibliography

(1) **History of Space Laser Communications,** M.Ross, McDonnell Douglas Astronautics Co. Invited Paper for Proceedings of the International Society for Optical Engineering. Conference held in January 1988.

(2) **Experimental and preoperational optical intersatellite links,** M.witting, G. Oppenhaeusar, A. Hahne, European Space Agency (Netherlands). Paper for the Fifth Conference on Free Space Laser Technology.

(3) **Pockels Cell Primer,** Robert Goldstein,Laser Focus, Feb 1968.

(4) **Transverse Feild Electro-Optic Light Modulators,** Technical Memo 7011, Lasermetrics, Inc.

(5) **Optical Communication Systems,** John Gowar, University of Bristol, UK. Prentice/Hall International,Inc, London.

(6) **Laser Communication Systems,** William K Pratt, USC, Los Angeles. John Wiley & Sons, Inc., NewYork.

(7) **Optics and Lasers,** M.Young. Springr - Verlag, Volume 5, third edition.

(8) **The RS-232 Solution,** Joe Campbell.
Sybex, Second Edition.

(9) **Inside Macintosh Vol I & II,** Apple
Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

(10) **Macintosh Revealed Vol I & II,** Stephan Chernicoff
Hayden book company, Hasbrouck, New Jersey.

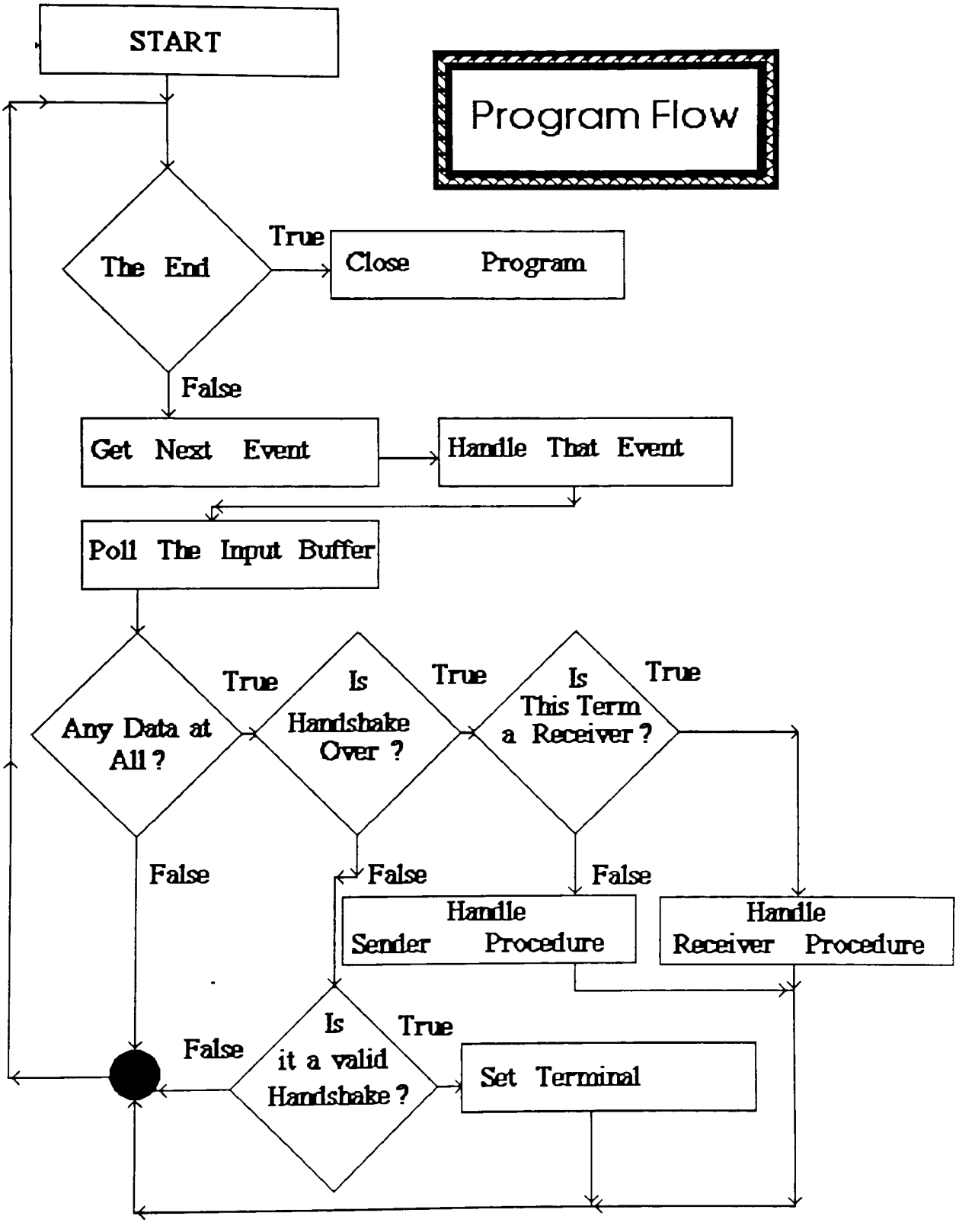
(11) **Macintosh Programming Primer,** Dave Mark & Cartwright Reed.
Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

(12) **C Programming Technique for Macintosh,** Zigurd R. Mednieks, T Schilke
Howard W. Sams & Company, Indiana.

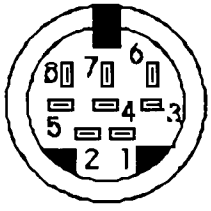
(13) **Computer Networks,** Andrew Tanenbaum, Vrije Universiteit, Amsterdam,
Netherlands.
Prentice-Hall, Inc, Englewood Cliffs, NJ.

Appendix A

Program Flow

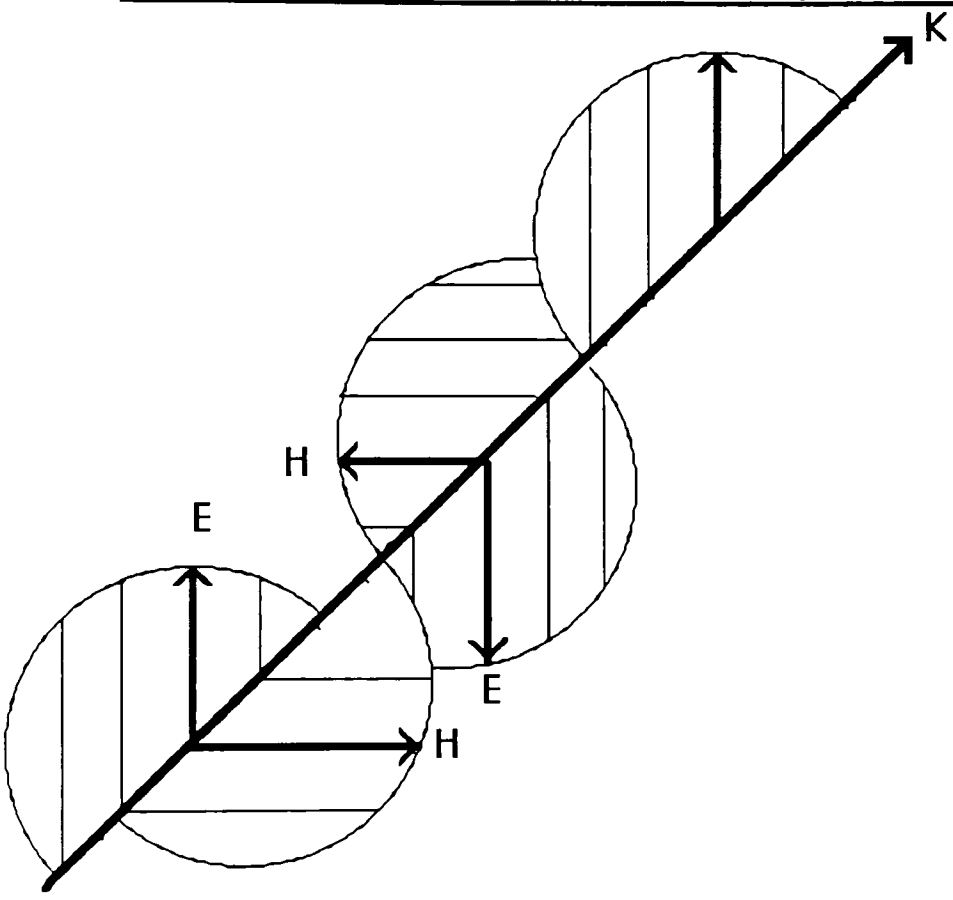

START
Program Flow
The End
True
Close Program
False
Get Next Event
Handle That Event
Poll The Input Buffer
Any Data at All ?
True
Is Handshake Over ?
True
Is This Term a Receiver ?
True
False
False
False
Handle Sender Procedure
Handle Receiver Procedure
Is it a valid Handshake ?
True
False
Set Terminal

Appendix B
Serial Port diagram



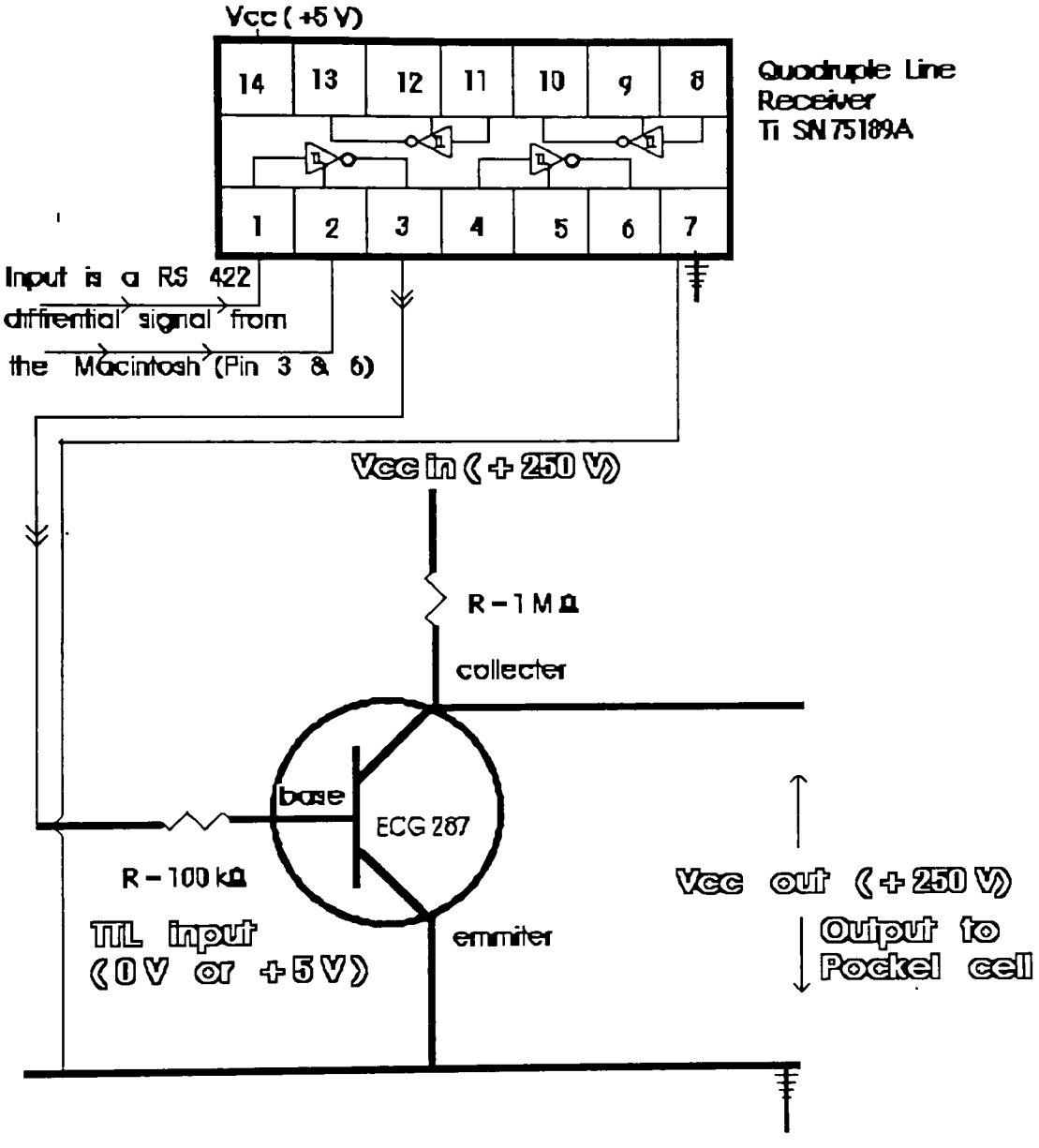
Pin numbers	Description
1	Output Handshake (DTR).
2	Input Handshake (CTR), external clock.
3	Transmit - voltage (TxD-).
4	Ground
5	Receive - voltage (RxD-).
6	Transmit + voltage (TxD+).
7	Unused.
8	Receive + voltage (RxD+).

Appendix C
Field vector direction in Polarization

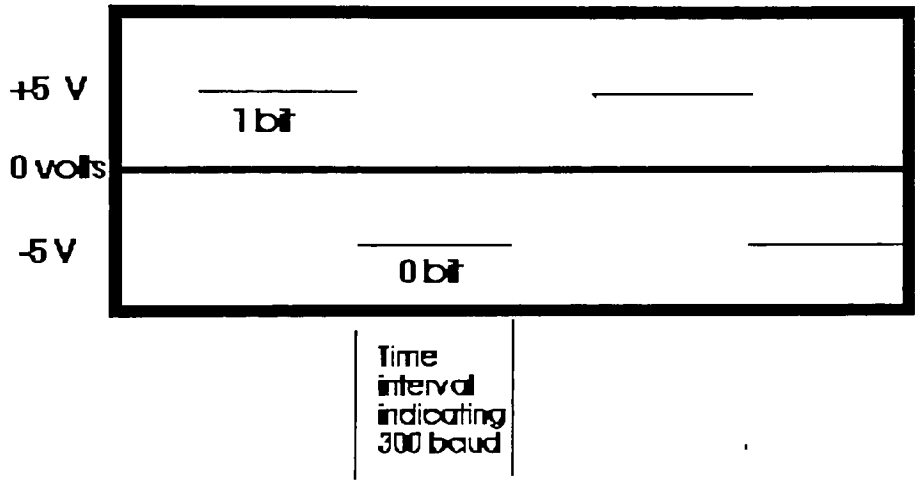


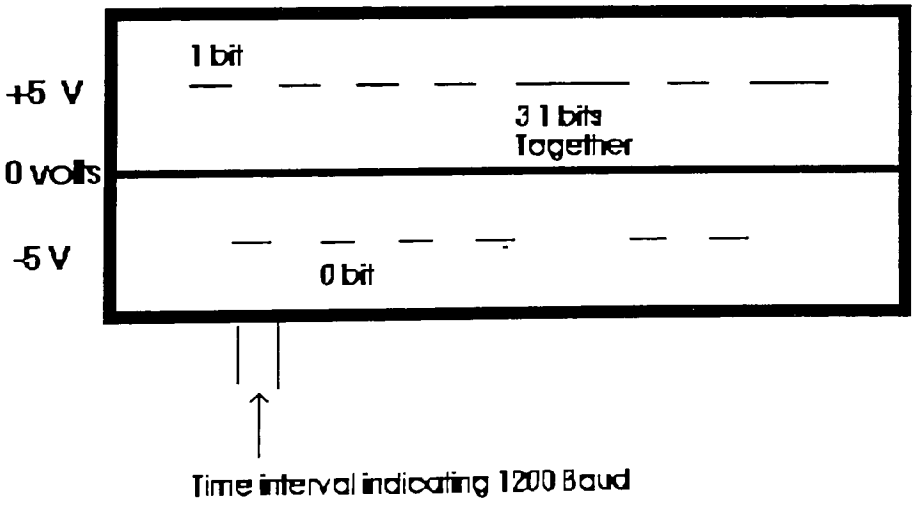
Appendix D

Circuit # 1


Vcc (+5 V)
14
13
12
11
10
9
8
1
2
3
4
5
6
7
Quadruple Line
Receiver
Ti SN75189A
Input is a RS 422
differential signal from
the Macintosh (Pin 3 & 6)
Vcc in (+ 250 V)
R - 1 MΩ
collecter
base
ECG 287
R - 100 kΩ
TTL input
(0 V or +5 V)
emmiter
Vcc out (+ 250 V)
Output to
Pockel cell

Appendix E
Oscilloscope representation





Appendix F
The Entire System

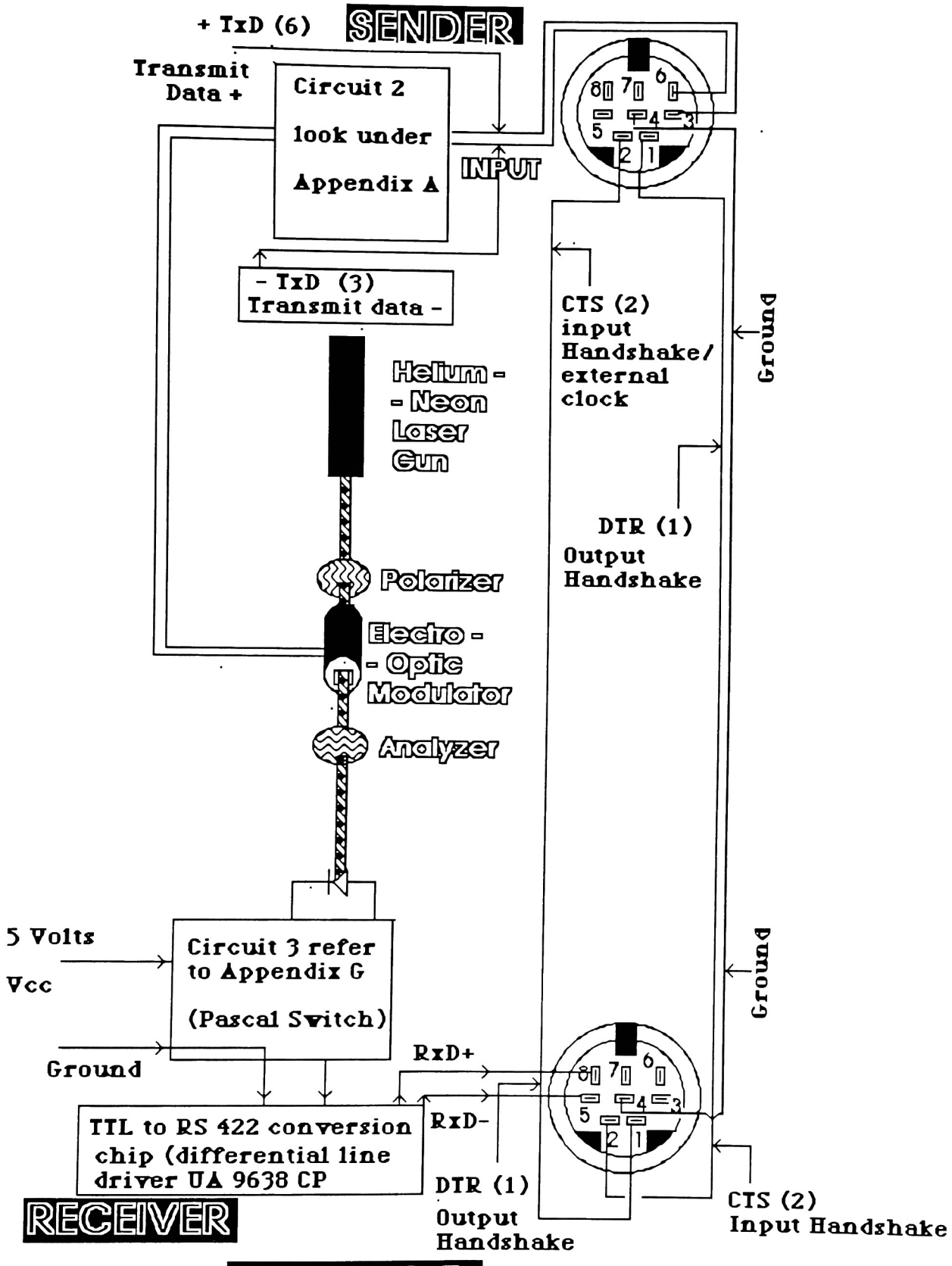
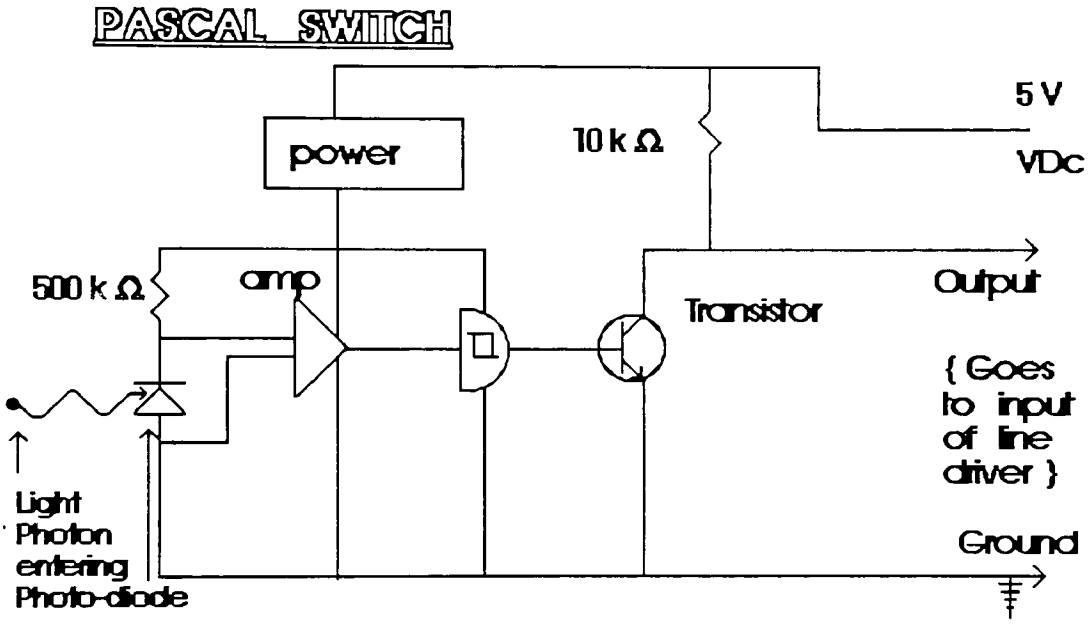
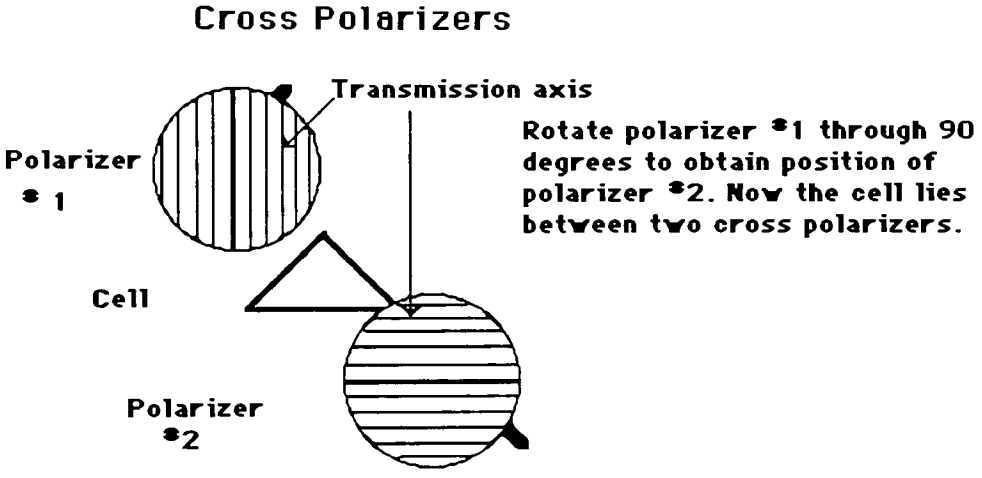

+ TxD (6)
SENDER
Transmit Data +
Circuit 2
look under
Appendix A
INPUT
- TxD (3)
Transmit data -
Helium - - Neon Laser Gun
Polarizer
Electro - - Optic Modulator
Analyzer
CTS (2) input Handshake/ external clock
Ground
DTR (1) Output Handshake
5 Volts
Vcc
Circuit 3 refer to Appendix G
(Pascal Switch)
Ground
RxD+
RxD-
TTL to RS 422 conversion chip (differential line driver UA 9638 CP
RECEIVER
DTR (1) Output Handshake
Ground
CTS (2) Input Handshake

Figure 2.5

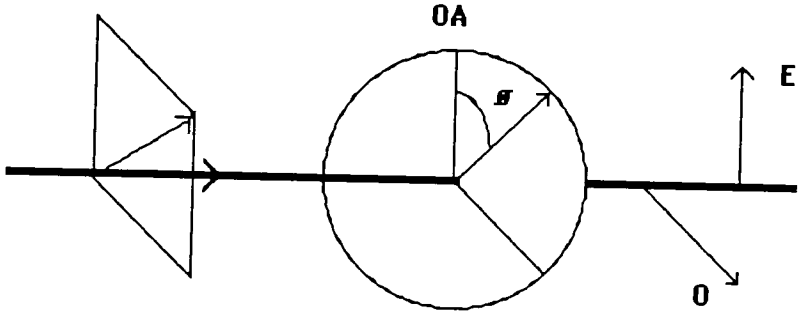
Appendix G
The Pascal Switch



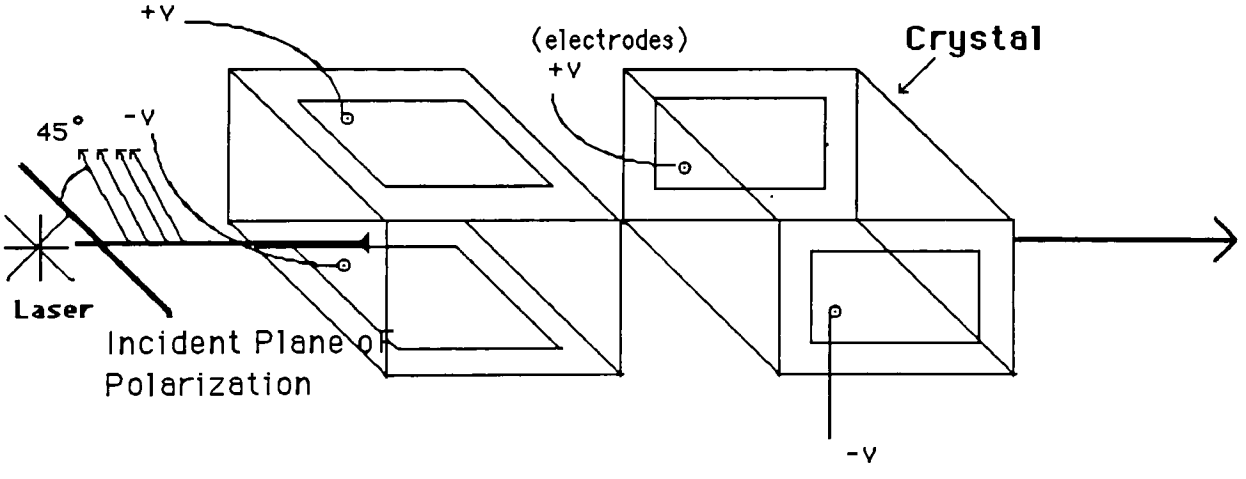
Appendix H
Cross polarizers



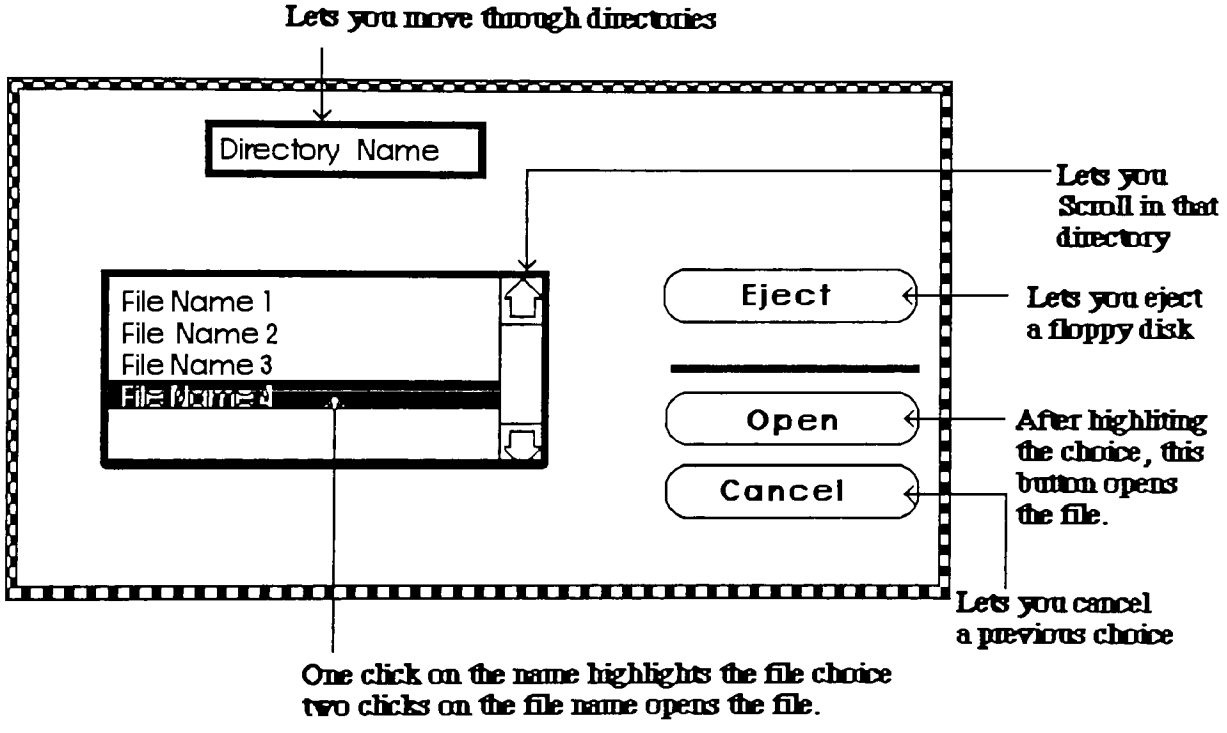
Appendix I
Wave Plate diagram



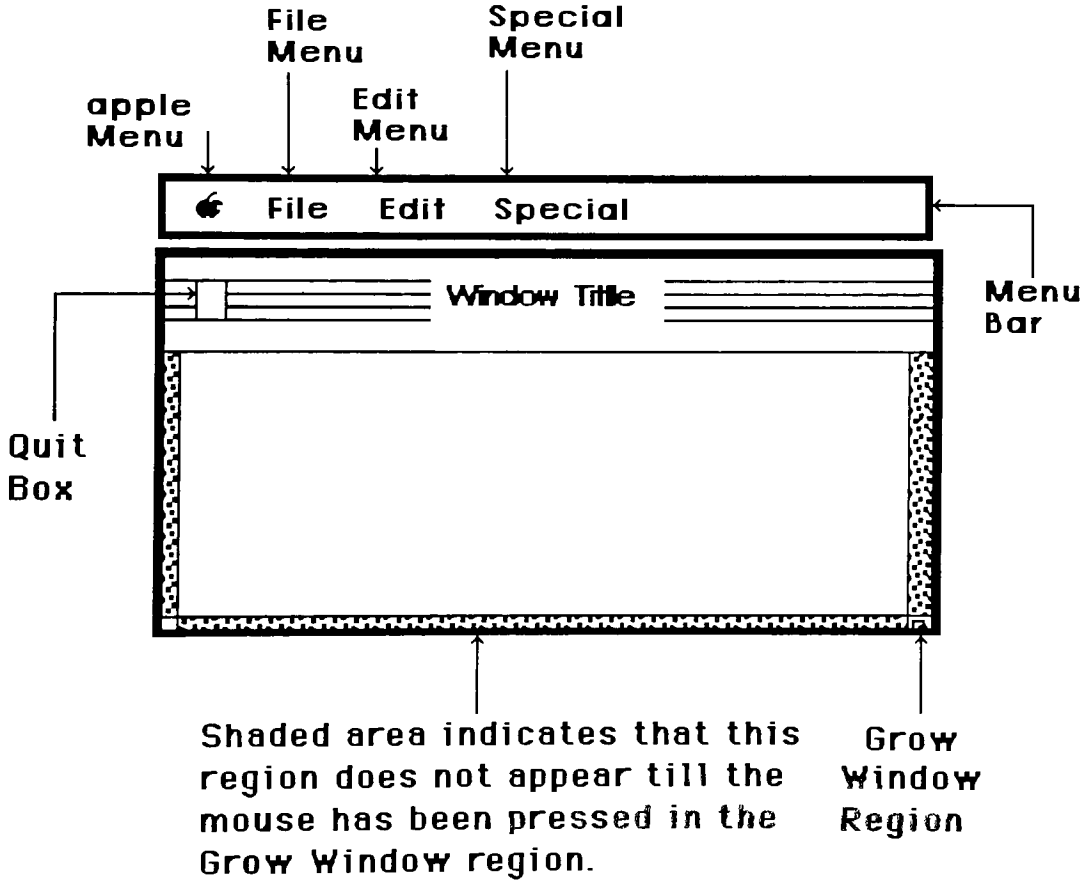
Appendix J
Inside the Pockel cell



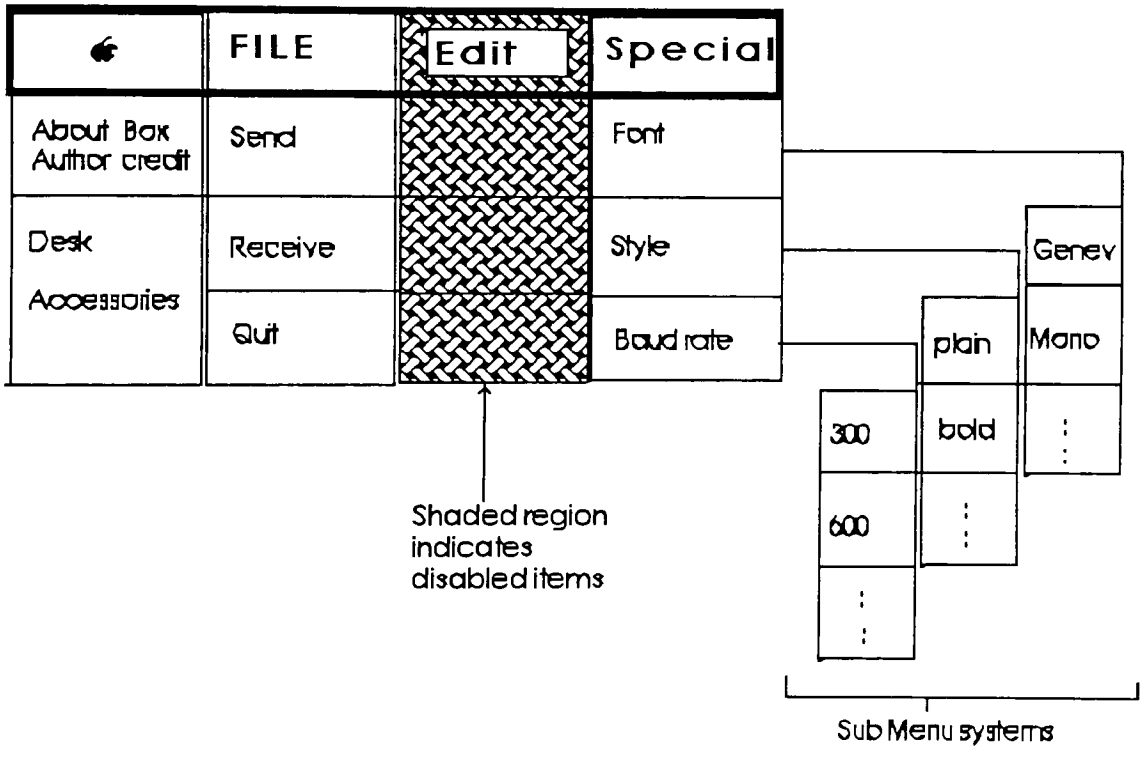
Appendix K
Dialog box



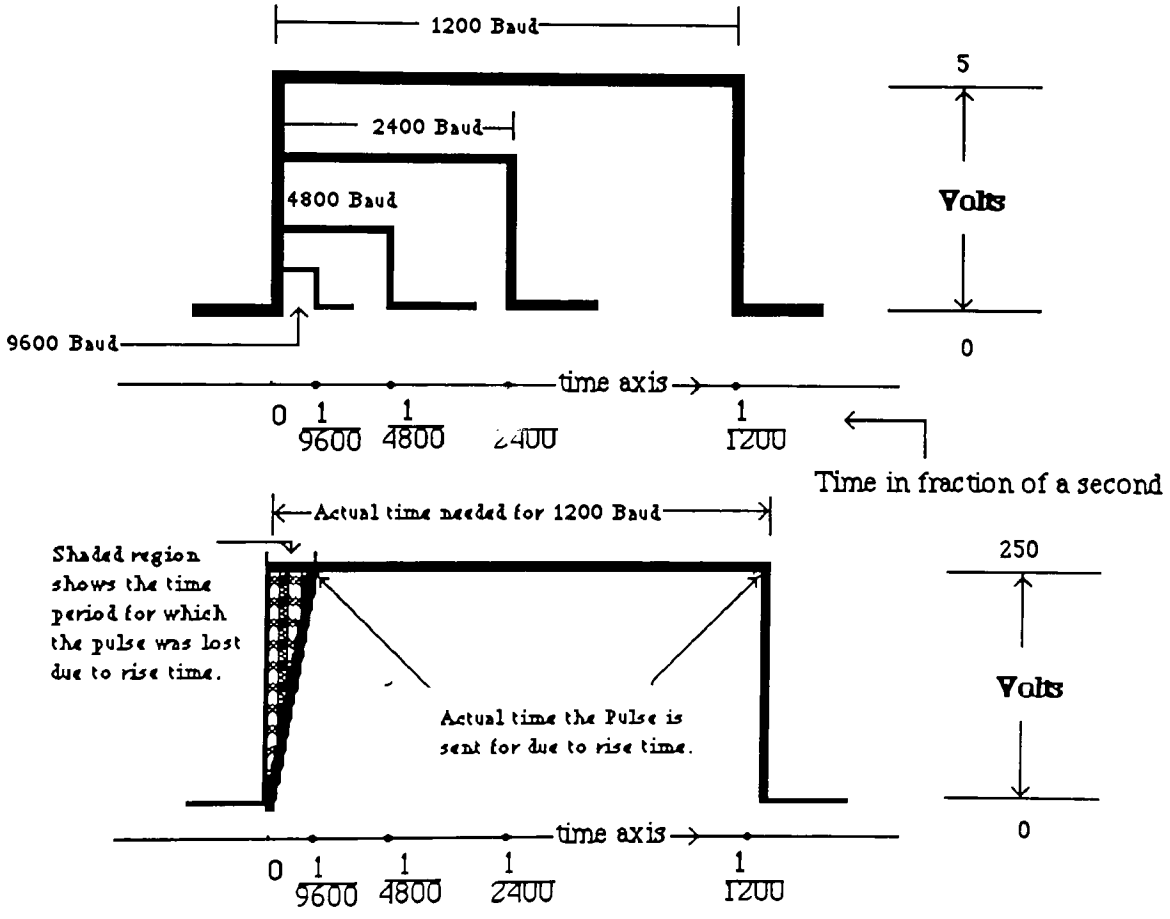
Appendix L
Window Display



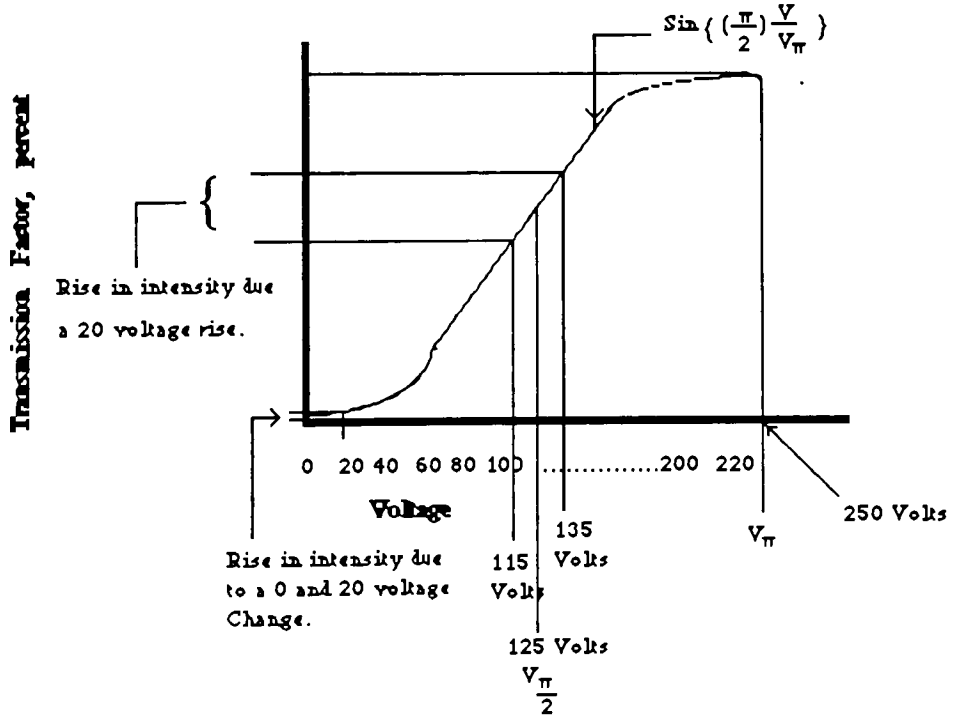
Appendix M
Menu Display



Appendix N
Rise Time Display



Appendix O
Ratio Graph of Intensity vs Voltage



Glossary

DARPA	:	Defense Advanced Research Projects Agency
DRS	:	Data Relay Satellite
ESA	:	European Space Agency
JPL	:	Jet Propulsion Laboratories (CalTech.)
LASER	:	Light Amplification by Stimulated Emmision of Radiation
NASA	:	National Aeronotical Space Agency
N.M.	:	New Mexico
Silex	:	Semiconductor Inter Satellite Laser EXperiment
YAG	:	Yttrium Aluminum Garnet
BPS	:	Bits Per Second
LAN	:	Local area Networks
MAN	:	Metropolitan area Networks
ISO	:	International Standerds Model
OSI	:	Open Systems InterConnection

Missing Page

PROGRAM CODE

```
/*****************************************************************************/
/*                        RESOURCE    FILE                                   */
/*                        FOR PROGRAM MODITALK                               */
/*                        FILENAME:   ModiTalk.r                             */
/*                        DATE    :   JANUARY 1990                           */
/*                        AUTHOR  :   SOHRAB MODI                            */
/*****************************************************************************/
/*****************************************************************************/
/*  This is a resource file. This file contains information on windows,      */
/*  menubars, dialog boxes, dialog strings, greetring picture, meuus, alerts*/
/*  and cursers. This file was created using Res Edit. This code dump was    */
/*  obtained using MPW  derez function.                                      */
/*****************************************************************************/
data 'WIND' (400, "Laser Comm") {
    $"00 28 00 06 00 E2 01 F9 00 00 01 00 01 00 00 00"    /* .(.............. */
    $"00 00 23 4C 61 73 65 72 20 43 6F 6D 6D 75 6E 69"    /* ..#Laser Communi */
    $"63 61 74 69 6F 6E 73 20 42 79 20 53 6F 68 72 61"    /* cations By Sohra */
    $"62 20 4D 6F 64 69"                                  /* b Modi */
};

data 'WIND' (401, "greeting window") {
    $"00 14 00 01 01 55 01 FE 00 01 01 00 01 00 00 00"    /* .....U.......... */
    $"00 00 00"                                           /* ... */
};

data 'WIND' (402, "HANDSHAKE WINDOW") {
    $"01 01 00 02 01 4F 01 F9 00 00 01 00 01 00 00 00"    /* .....O.......... */
    $"00 00 10 48 41 4E 44 53 48 41 4B 45 20 57 49 4E"    /* ...HANDSHAKE WIN */
    $"44 4F 57"                                           /* DOW */
};

data 'MBAR' (400) {
    $"00 04 01 90 01 91 01 92 01 93"                      /* ...ê.ë.í.ì */
};

data 'MENU' (400, "Apple") {
    $"01 90 00 00 00 00 00 00 00 00 FF FF FF FB 01 14"    /* .ê.............. */
    $"0C 41 62 6F 75 74 20 4C 61 73 63 6F 6D 00 00 00"    /* .About Lascom... */
    $"00 01 2D 00 00 00 01 00"                            /* ..-..... */
};

data 'MENU' (401, "File") {
    $"01 91 00 00 00 00 00 00 00 00 FF FF FF FF 04 46"    /* .ë.............F */
    $"69 6C 65 04 53 65 6E 64 00 53 00 00 07 52 65 63"    /* ile.Send.S...Rec */
    $"65 69 76 65 00 52 00 00 04 51 75 69 74 00 51 00"    /* eive.R...Quit.Q. */
    $"00 00"                                              /* .. */
};

data 'MENU' (402, "Edit") {
    $"01 92 00 00 00 00 00 00 00 00 00 00 00 00 04 45"    /* .í.............E */
    $"64 69 74 04 55 6E 64 6F 00 5A 00 00 01 2D 00 00"    /* dit.Undo.Z...-.. */
    $"00 00 03 43 75 74 00 58 00 00 04 43 6F 70 79 00"    /* ...Cut.X...Copy. */
    $"43 00 00 05 50 61 73 74 65 00 56 00 00 05 43 6C"    /* C...Paste.V...Cl */
    $"65 61 72 00 00 00 00 00"                            /* ear..... */
};
```

```
data 'MENU' (403, "Special") {
    $"01 93 00 00 00 00 00 00 00 FF FF FF FF 07 53"    /* .ì.............S */
    $"70 65 63 69 61 6C 04 46 6F 6E 74 00 1B 64 00 05"    /* pecial.Font..d.. */
    $"53 74 79 6C 65 00 1B 65 00 0A 42 61 75 64 20 52"    /* Style..e..Baud R */
    $"61 74 65 73 00 1B 66 00 00"                         /* ates..f.. */
};

data 'MENU' (100, "Font") {
    $"00 64 00 00 00 00 00 00 00 FF FF FF FF 04 46"    /* .d.............F */
    $"6F 6E 74 00"                                        /* ont. */
};

data 'MENU' (101, "Style") {
    $"00 65 00 00 00 00 00 00 00 FF FF FF FF 05 53"    /* .e.............S */
    $"74 79 6C 65 05 50 6C 61 69 6E 00 00 00 00 04 42"    /* tyle.Plain.....B */
    $"6F 6C 64 00 00 00 01 06 49 74 61 6C 69 63 00 00"    /* old.....Italic.. */
    $"00 02 09 55 6E 64 65 72 6C 69 6E 65 00 00 00 04"    /* ..ΔUnderline.... */
    $"07 4F 75 74 6C 69 6E 65 00 00 00 08 06 53 68 61"    /* .Outline.....Sha */
    $"64 6F 77 00 00 00 10 00"                            /* dow..... */
};

data 'MENU' (102, "Baud Rate") {
    $"00 66 00 00 00 00 00 00 00 00 FF FF FF FF 09 42"    /* .f............ΔB */
    $"61 75 64 20 52 61 74 65 03 33 30 30 00 00 00 00"    /* aud Rate.300.... */
    $"03 36 30 30 00 00 00 00 04 31 32 30 30 00 00 00"    /* .600.....1200... */
    $"00 04 31 38 30 30 00 00 00 00 04 32 34 30 30 00"    /* ..1800.....2400. */
    $"00 00 00 04 33 36 30 30 00 00 00 00 04 34 38 30"    /* ....3600.....480 */
    $"30 00 00 00 00 04 37 32 30 30 00 00 00 C0 04 39"    /* 0.....7200...¿.9 */
    $"36 30 30 00 00 00 00 05 31 39 32 30 30 00 00 00"    /* 600.....19200... */
    $"00 05 35 37 36 30 30 00 00 00 00 00"                /* ..57600..... */
};

data 'DITL' (400) {
    $"00 01 00 00 00 00 00 47 00 75 00 5B 00 B1 04 02"    /* .......G.u.[.±.. */
    $"4F 4B 00 00 00 00 00 07 00 46 00 3D 01 18 08 4D"    /* OK.......F.=...M */
    $"4C 61 73 65 72 20 43 6F 6D 6D 75 6E 69 63 61 74"    /* Laser Communicat */
    $"69 6F 6E 20 73 79 73 74 65 6D 73 0D 20 20 20 20"    /* ion systems¬     */
    $"20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20"    /*                  */
    $"20 42 79 0D 20 20 20 20 20 20 20 20 20 20 20 20"    /*  By¬             */
    $"20 20 53 6F 68 72 61 62 20 4D 6F 64 69 00"          /*   Sohrab Modi. */
};

data 'DITL' (401, "For String Display") {
    $"00 01 00 00 00 00 00 7B 00 4B 00 8F 00 87 04 02"    /* .......{.K.è.á.. */
    $"4F 4B 00 00 00 00 00 26 00 0B 00 60 01 16 08 05"    /* OK.....&...`.... */
    $"20 20 0D 5E 30 0B"                                  /*   ¬^0. */
};

data 'ALRT' (400) {
    $"00 28 00 28 00 8E 01 4C 01 90 66 66"                /* .(.(.é.L.êff */
};

data 'ALRT' (401) {
    $"00 28 00 28 00 F0 01 18 01 91 77 77"                /* .(.(.....ëww */
};
```

```
data 'STR ' (400, "File Err Reading") {
    $"37 45 72 72 6F 72 20 3A 20 43 6F 75 6C 64 20 4E"    /* 7Error : Could N */
    $"6F 74 20 4F 70 65 6E 20 46 69 6C 65 20 66 6F 72"    /* ot Open File for */
    $"20 52 65 61 64 69 6E 67 20 49 20 77 69 6C 6C 20"    /*  Reading I will  */
    $"45 78 69 74 20 4E 4F 57"                            /* Exit NOW */
};

data 'STR ' (401, "File err Writing") {
    $"37 45 72 72 6F 72 20 3A 20 43 6F 75 6C 64 20 4E"    /* 7Error : Could N */
    $"6F 74 20 4F 70 65 6E 20 46 69 6C 65 20 66 6F 72"    /* ot Open File for */
    $"20 57 72 69 74 69 6E 67 20 49 20 77 69 6C 6C 20"    /*  Writing I will  */
    $"45 78 69 74 20 4E 4F 57"                            /* Exit NOW */
};

data 'STR ' (402, "Both Senders") {
    $"34 45 72 72 6F 72 20 3A 20 42 6F 74 68 20 20 54"    /* 4Error : Both  T */
    $"65 72 6D 69 6E 61 6C 73 20 61 72 65 0D 20 73 65"    /* erminals are¬ se */
    $"74 20 74 6F 20 61 63 74 20 61 73 20 53 65 6E 64"    /* t to act as Send */
    $"65 72 73 2E 20"                                     /* ers.  */
};

data 'STR ' (403, "Both Recivers") {
    $"32 45 72 72 6F 72 3A 20 42 6F 74 68 20 54 65 72"    /* 2Error: Both Ter */
    $"6D 69 6E 61 6C 73 20 61 72 65 0D 20 73 65 74 20"    /* minals are¬ set  */
    $"74 6F 20 61 63 74 20 61 73 20 52 65 63 69 76 65"    /* to act as Recive */
    $"72 73 2E"                                           /* rs. */
};

data 'STR ' (404, "FATAL ERR") {
    $"31 2A 2A 46 41 54 41 4C 20 45 52 52 4F 52 20 49"    /* 1**FATAL ERROR I */
    $"20 43 41 4E 4E 4F 54 20 52 45 43 4F 56 45 52 2A"    /*  CANNOT RECOVER* */
    $"2A 0D 49 20 57 49 4C 4C 20 45 58 49 54 20 4E 4F"    /* *¬I WILL EXIT NO */
    $"57 20"                                              /* W  */
};

data 'STR ' (405, "BEGIN TRANSFER") {
    $"19 2A 2A 20 49 20 41 4D 20 52 45 41 44 59 20 54"    /* .** I AM READY T */
    $"4F 20 42 45 47 49 4E 20 2A 2A"                      /* O BEGIN ** */
};

data 'STR ' (406, "BAD_SHAKE") {
    $"3E 42 41 44 20 53 49 47 4E 41 4C 20 52 45 43 45"    /* >BAD SIGNAL RECE */
    $"49 56 45 44 20 42 59 20 4F 54 48 45 52 20 0D 54"    /* IVED BY OTHER ¬T */
    $"45 52 4D 49 4E 41 4C 2E 20 50 4C 45 41 53 45 20"    /* ERMINAL. PLEASE  */
    $"52 45 53 45 54 20 54 45 52 4D 49 4E 41 4C 2E"       /* RESET TERMINAL. */
};

data 'STR ' (407, "THE END") {
    $"2D 49 20 48 41 56 45 20 46 49 4E 49 53 48 45 44"    /* -I HAVE FINISHED */
    $"20 54 52 41 4E 53 4D 49 54 54 49 4E 47 0D 49 20"    /*  TRANSMITTING¬I  */
    $"57 49 4C 4C 20 51 55 49 54 20 4E 4F 57 2E"          /* WILL QUIT NOW. */
};

data 'STR ' (408, "TRANS_OVER_RE") {
    $"2B 49 20 48 41 56 45 20 20 46 49 4E 49 53 48 45"    /* +I HAVE  FINISHE */
    $"44 20 52 45 43 45 49 56 49 4E 47 2E 0D 49 57 49"    /* D RECEIVING.¬IWI */
```

```
    $"4C 4C 20 51 55 49 54 20 4E 4F 57 2E"                  /* LL QUIT NOW. */
};

data 'STR ' (409, "redefine sender") {
    $"65 53 6F 72 72 79 20 21 21 21 20 54 68 69 73 20"      /* eSorry !!! This  */
    $"74 65 72 6D 69 6E 61 6C 20 69 73 20 61 6C 72 65"      /* terminal is alre */
    $"61 64 79 0D 64 65 66 69 6E 65 64 20 74 6F 20 61"      /* ady¬defined to a */
    $"63 74 20 61 73 20 53 65 6E 64 65 72 2E 20 50 6C"      /* ct as Sender. Pl */
    $"65 61 73 65 0D 51 75 69 74 20 74 6F 20 72 65 20"      /* ease¬Quit to re  */
    $"2D 20 64 65 66 69 6E 65 20 74 68 65 20 54 65 72"      /* - define the Ter */
    $"6D 69 6E 61 6C 2E"                                    /* minal. */
};

data 'STR ' (40, "redefine sender") {
    $"47 53 6F 72 72 79 20 21 21 21 20 54 68 69 73 20"      /* GSorry !!! This  */
    $"74 65 72 6D 69 6E 61 6C 20 69 73 20 64 65 66 69"      /* terminal is defi */
    $"6E 65 64 20 0D 74 6F 20 61 63 74 20 61 73 20 53"      /* ned ¬to act as S */
    $"65 6E 64 65 72 2E 20 51 75 69 74 20 74 6F 20 72"      /* ender. Quit to r */
    $"65 2D 64 65 66 69 6E 65"                              /* e-define */
};

data 'STR ' (410, "redefine receiver") {
    $"47 53 6F 72 72 79 20 21 20 54 68 69 73 20 74 65"      /* GSorry ! This te */
    $"72 6D 69 6E 61 6C 20 69 73 20 64 65 66 69 6E 65"      /* rminal is define */
    $"64 20 74 6F 20 0D 61 63 74 20 61 73 20 52 65 63"      /* d to ¬act as Rec */
    $"65 69 76 65 72 2E 20 51 75 69 74 20 74 6F 20 72"      /* eiver. Quit to r */
    $"65 2D 64 65 66 69 6E 65"                              /* e-define */
};

data 'PICT' (18562) {
    $"00 00 00 00 00 00 00 00 00 00"                        /* .......... */
};

data 'PICT' (400) {
    $"20 4B 00 00 00 00 01 55 02 00 11 01 A0 00 82 A0"      /*  K.....U....†.Ç† */
    $"00 8E 01 00 0A 00 00 00 00 02 D0 02 40 98 00 40"      /* .é........-.@ò.@ */
    $"00 00 00 00 00 30 02 00 00 00 00 00 00 30 02 00"      /* .....0.......0.. */
    $"00 00 00 00 00 30 02 00 00 01 08 00 17 D4 FF 00"      /* .....0.......'.. */
    $"FB F0 FF 08 00 1F D4 FF 00 FD F0 FF 08 00 3F D4"      /* ......'.......?' */
    $"FF 00 F9 F0 FF 04 00 5F C2 FF 04 00 BF C2 FF 02"      /* ......._¬...ø¬.. */
    $"C1 FF 02 C1 FF 02 C1 FF 02 C1 FF 02 C1 FF 06 EF"      /* ¡..¡..¡..¡..¡... */
    $"FF 00 EF D4 FF 06 EF FF 00 F7 D4 FF 0F EF FF 00"      /* ...'.....'..... */
    $"D5 FA FF 01 FC 7F E4 FF 00 FD FB FF 0A EF FF 00"      /* ’............... */
    $"A2 DB FF 00 F8 FB FF 0B EF FF 01 40 BF DC FF 00"      /* ¢..........@ø... */
    $"F9 FB FF 19 F0 FF 05 FE 81 5F FF C0 3F FC FF 01"      /* ........Å_.¿?... */
    $"E0 0F F9 FF 05 FC 7F FF FF FE 3F ED FF 18 F0 FF"      /* ..........?..... */
    $"04 FD 00 2F FF F0 FB FF 01 87 0F F9 FF 05 F8 7F"      /* .../.....á...... */
    $"FF FF 0C 3F ED FF 18 F0 FF 04 FE 80 57 FF F0 FB"      /* ...?.......ÄW... */
    $"FF 01 0F 8F F9 FF 00 F8 FE FF 01 0C 7F ED FF 19"      /* ...è............ */
    $"F7 FF 00 5F FB FF 04 F5 00 BF FF F0 FC FF 02 FE"      /* ..._.....ø...... */
    $"1F 9F F6 FF 01 FE 1F EC FF 28 F7 FF 00 0F FB FF"      /* .ü.......(...... */
    $"1E FE 80 5F FF E1 FF 01 01 C1 C0 7E 1F 9E 03 00"      /* ..Ä_.....¡¿~.û.. */
    $"41 C0 10 78 61 80 78 70 3C 00 04 38 0E 01 C0 7F"      /* A¿.xaÄxp<..8..¿. */
    $"F1 FF 28 F8 FF 01 FE 07 FA FF 1D 41 7F FF E1 FC"      /* ..(........A.... */
    $"02 11 80 00 7C 3F FC 20 04 10 01 04 00 80 00 00"      /* ..Ä.|?. .....Ä.. */
    $"62 30 0E 00 30 80 00 84 7F F1 FF 27 F8 FF 01 FE"      /* b0..0Ä.Ñ...'.... */
```

```
$"0F FA FF 1C 95 7F FF E1 F8 C2 13 08 80 FC 3F F8"    /* ....ï....¬..Ä.?. */
$"61 0C 30 C3 0C 30 C3 0C 30 C2 23 0C 38 61 84 30"    /* a.0√.0√.0¬#.8aÑ0 */
$"84 F0 FF 27 F8 FF 01 FE 0F FA FF 1C EA FF FF E1"    /* Ñ..'............ */
$"F0 C2 0F 11 87 FC 3F F8 61 0C 30 C3 0C 30 C3 0C"    /* .¬..á.?.a.0√.0√. */
$"30 C3 C3 0C 38 61 84 30 83 F0 FF 2A FC FF 00 BF"    /* 0√√.8aÑ0É..*...ø */
$"FD FF 00 5F FA FF 1C D7 FF FF E1 F1 C3 06 03 87"    /* ..._...◊....√..á */
$"FC 3F 90 E1 0C 30 C3 0C 30 C3 0C 30 87 C7 0C 38"    /* .?ê..0√.0√.0á«.8 */
$"43 84 30 C1 F0 FF 26 FC FF 00 0F F5 FF 1C EF FF"    /* CÑ0¡..&......... */
$"FF C3 C1 84 82 1C 0F FC 3F 30 C2 18 61 86 18 61"    /* .√¡ÑÇ...?0¬.aÜ.a */
$"86 18 61 87 06 18 70 C3 08 61 20 F0 FF 23 FC FF"    /* Ü.aá..p√.a ..#.. */
$"00 0F F2 FF 19 C3 81 84 C2 19 0F FC 3E 30 C2 18"    /* .....√ÅÑ¬...>0¬. */
$"61 86 18 61 06 18 61 86 06 18 40 C3 08 61 30 F0"    /* aÜ.a..aÜ..@√.a0. */
$"FF 23 FC FF 00 4F F2 FF 19 C3 00 00 42 03 0F FE"    /* .#...O...√..B... */
$"1C 30 86 18 61 86 18 60 02 18 61 80 80 08 10 C2"    /* .0Ü.aÜ.`..aÄÄ..¬ */
$"18 61 10 F0 FF 22 FC FF 00 DF F2 FF 09 00 10 84"    /* .a..."......Δ..Ñ */
$"07 06 07 FF 80 F8 0C FD 00 0B 30 8C 00 00 C1 C2"    /* ....Ä.....0å..¡¬ */
$"1C 20 60 30 00 01 F0 FF 02 C1 FF 02 C1 FF 02 C1"    /* . `0.....¡..¡..¡ */
$"FF 02 C1 FF 02 C1 FF 06 E3 FF 00 E1 E0 FF 06 E3"    /* ..¡..¡.......... */
$"FF 00 81 E0 FF 06 E3 FF 00 E1 E0 FF 06 E3 FF 00"    /* ..Å.............. */
$"E1 E0 FF 06 E3 FF 00 C3 E0 FF 08 E3 FF 02 C0 20"    /* .......√......¿  */
$"0F E2 FF 08 E3 FF 02 C0 10 CF E2 FF 08 E3 FF 02"    /* .......¿.œ...... */
$"86 10 CF E2 FF 0C E3 FF 02 86 10 8F FC FF 00 9F"    /* Ü.œ......Ü.è...ü */
$"E8 FF 10 E3 FF 02 86 10 9F FC FF 00 8F FB FF 00"    /* ......Ü.ü...è... */
$"DF EF FF 10 E3 FF 02 0C 38 1F FC FF 00 3F FB FF"    /* ........8....?.. */
$"00 2F EF FF 0C E3 FF 02 0C 38 3F F5 FF 00 07 EF"    /* ./.......8?..... */
$"FF 0C E3 FF 02 08 78 3F F5 FF 00 4B EF FF 0C E3"    /* ......x?...K.... */
$"FF 02 01 F8 7F F5 FF 00 4F EF FF 0A E2 FF 00 FC"    /* ........O....... */
$"F4 FF 00 5F EF FF 98 00 40 00 30 00 00 00 60 02"    /* ..._..ò.@.0...`. */
$"00 00 30 00 00 00 60 02 00 00 30 00 00 00 60 02"    /* ..0...`...0...`. */
$"00 00 01 0A E2 FF 00 F8 F4 FF 00 EF EF FF 06 E3"    /* ................ */
$"FF 00 87 E0 FF 15 E6 FF 0F E0 3F FE 07 FF FF FC"    /* ..á.......?..... */
$"0F FF FF 03 F0 3F FF C0 DF EC FF 15 E6 FF 03 C6"    /* .....?.¿.......Δ */
$"1F FF 87 FE FF 08 0F FF FF C3 F0 FF FF F0 9F EC"    /* ..á......√....ü. */
$"FF 1D FA FF 00 FE F7 FF 00 9F F9 FF 03 87 3F FF"    /* .........ü...á?. */
$"87 FE FF 08 0F FF FF C3 E0 FF FF F0 9F EC FF 20"    /* á......√....ü..  */
$"FA FF 01 FC 7F FB FF 03 EB FF FF EF F9 FF 0E 87"    /* ..............á */
$"3F FF 0F FF FF FE 1F FF FF 83 C0 FF FF E1 EB FF"    /* ?........É¿..... */
$"22 FA FF 01 F0 1F FB FF 00 C3 F6 FF 0F 83 FC 07"    /* "........√...É.. */
$"00 F0 1C 06 01 FF FF 81 C0 F0 1F 01 8F F2 FF 00"    /* .......Å¿...è... */
$"3F FC FF 26 FA FF 01 FC 7F FB FF 00 D5 FC FF 00"    /* ?..&........'... */
$"9F FC FF 0F C1 F8 43 08 00 10 0E 00 FF FF 81 80"    /* ü...¡.C.......ÅÄ */
$"E1 0C 20 0F F2 FF 00 DF FC FF 1D FA FF 00 FE FA"    /* .. ............. */
$"FF 00 D7 F6 FF 0F E0 F0 C2 18 60 23 0C 30 FF FF"    /* ..◊.....¬.`#.0.. */
$"01 10 C3 08 C3 0F EC FF 19 F2 FF 00 EF F6 FF 0F"    /* ..√.√........... */
$"F0 70 C2 18 61 C3 0C 30 FF FF 11 20 C3 00 C3 0F"    /* .p¬.a√.0... √.√. */
$"EC FF 15 E6 FF 0F F8 21 C2 18 61 C7 0C 30 FF FF"    /* .......!¬.a«.0.. */
$"10 21 87 01 C3 0F EC FF 15 E6 FF 0F 3C 21 84 30"    /* .!á.√.......<!Ñ0 */
$"C3 86 18 61 FF FE 30 61 86 01 86 0F EC FF 16 E7"    /* √Ü.a..0aÜ.Ü..... */
$"FF 10 FE 3C 21 84 30 C3 86 18 61 FF FE 30 E1 86"    /* ...<!Ñ0√Ü.a..0.Ü */
$"01 86 1F EC FF 19 EA FF 13 F7 FF FF FE 18 61 0C"    /* .Ü.............a. */
$"30 C3 80 08 43 FF FE 38 E1 84 20 00 1F EC FF 19"    /* 0√Ä.C..8.Ñ ..... */
$"EA FF 13 E3 FF C7 FF 00 F0 18 00 01 C2 18 0F FF"    /* .....«......¬... */
$"F8 09 80 40 70 84 1F EC FF 08 EA FF 02 F7 FF 83"    /* .ΔÄ@pÑ.........É */
$"DB FF 06 E8 FF 00 C7 DB FF 06 E8 FF 00 EF DB FF"    /* ......«......... */
$"02 C1 FF 02 C1 FF 06 DF FF 00 FD E4 FF 06 DF FF"    /* .¡..¡........... */
$"00 F8 E4 FF 06 E7 FF 00 DF DC FF 0A E7 FF 00 8F"    /* ..............è */
```

```
$"FB FF 00 FD E3 FF 07 E8 FF 01 DF DF DC FF 02 C1"    /* ..............¡ */
$"FF 06 E2 FF 00 FE E1 FF 06 E0 FF 00 FB E3 FF 09"    /* ..............Δ */
$"E3 FF 03 7F FF FF F4 E3 FF 0D ED FF 00 DF F8 FF"    /* .........¬..... */
$"03 EF FF FF D1 E3 FF 14 ED FF 00 DF FB FF 06 7F"    /* ....—.......... */
$"FF FA FF FF EF C6 F6 FF 00 7F EF FF 13 E4 FF 00"    /* .....Δ.......... */
$"F4 FE FF 00 DB F7 FF 01 FD 7F FB FF 00 BF F6 FF"    /* ............ø.. */
$"17 E8 FF 08 DF FF FD FB D7 FF 7F FF 7F F7 FF 00"    /* ........◊....... */
$"E1 FB FF 01 F8 7F F6 FF 19 E8 FF 04 F3 FF FF FD"    /* ................ */
$"7B FE FF 01 DB F7 F8 FF 01 FE 7F FC FF 01 FE 0F"    /* {............... */
$"F6 FF 19 01 FF 7F EA FF 07 F7 FF FF FA 7F 7F FF"    /* ................ */
$"7F F6 FF 00 FD FB FF 01 FD 1F F6 FF 1B 01 FF 3F"    /* ..............? */
$"EE FF 00 7F FE FF 06 FB FF FF FD 5F B6 DF FE FF"    /* ..........._∂... */
$"00 FE F1 FF 00 5F F6 FF 1A 01 FF 7F EF FF 01 FD"    /* ....._.......... */
$"7F FD FF 08 BF FF FE FF FF 7F FF FF DF F0 FF 00"    /* ....ø........... */
$"BF F6 FF 10 ED FF 01 F5 5F FC FF 05 F7 F7 7F FF"    /* ø......._....... */
$"FF EF E2 FF 0C ED FF 01 EA 2F F8 FF 01 FE D7 E2"    /* ........./....◊. */
$"FF 12 ED FF 01 F4 5F FE FF 01 F7 FB FC FF 02 EF"    /* ......_......... */
$"FB F7 E4 FF 11 F8 FF 00 FD F7 FF 01 D0 0F F7 FF"    /* ............-... */
$"02 FE FF D7 E4 FF 16 F8 FF 00 F8 F7 FF 05 E8 17"    /* ...◊............ */
$"F7 FF FF BF FC FF 03 FD FF FF C7 E4 FF 14 F8 FF"    /* ...ø......«..... */
$"00 FD F7 FF 02 E8 2F FB FC FF 00 F7 FD FF 00 BF"    /* ....../........ø */
$"E3 FF 10 ED FF 01 F4 5F F8 FF 00 BF FE FF 01 FB"    /* ......._...ø.... */
$"FB E6 FF 16 ED FF 03 FB 5F FF FD FD FF 07 7F FE"    /* ........_....... */
$"FF FD EF FF FD F7 E6 FF 00 5F 18 F1 FF 00 7F FE"    /* ........._...... */
$"FF 00 FE FE FF 00 BB FA FF 03 FB FF FF E7 E7 FF"    /* ......ª......... */
$"01 FE 9F 17 F1 FF 00 0F F8 FF 03 FE FF BF BF FE"    /* ..ü.........øø. */
$"FF 03 7F EF FB FD E8 FF 01 FD 07 18 F5 FF 04 DF"    /*  */
$"FF FF FD 17 F8 FF 07 DD FF FF FB EE EF 7F FD E5"    /*  */
$"FF 01 FC 0F 98 00 40 00 60 00 00 00 90 02 00 00"    /* ....ò.@.`...ê... */
$"60 00 00 00 90 02 00 00 60 00 00 00 90 02 00 00"    /* `...ê...`...ê... */
$"01 1A F5 FF 04 8F FF FF FC 0F F9 FF 00 FB FE FF"    /* .....è.......... */
$"05 EF FD F6 DF FE EF E6 FF 01 FE 1F 1B F5 FF 00"    /*  */
$"DF FE FF 00 17 F9 FF 0B EF BF FF FD DD FF 7B B7"    /* .........ø....{Σ */
$"77 FF FF FB E7 FF 00 3F 17 F8 FF 00 EF FD FF 04"    /* w......?........ */
$"7F FF AF FF F7 F7 FF 05 F6 DB 6E EF FF FE E4 FF"    /* ..ø.......n..... */
$"1D F8 FF 00 C7 FE FF 05 FD 3F FF BF FF D7 FC FF"    /* ....«....?.ø.◊.. */
$"0A FD EF FF FB DB 7B B6 DB BD EF DF E4 FF 1E F8"    /* ......{∂.Ω...... */
$"FF 00 EF FE FF 01 F8 1F FE FF 03 C3 FF DF DF FD"    /* ...........√.... */
$"FF 08 FE DD B6 DD 6F 76 EE DF FE E4 FF 1B F4 FF"    /* ....∂.ov........ */
$"01 F8 0F FE FF 10 C7 FE FF AF FF FF BF ED FF F7"    /* ......«..ø..ø... */
$"DD B6 DB BB BB FB EF E4 FF 1E F4 FF 01 F4 17 FE"    /* .∂.ªª........... */
$"FF 13 FB FC 7F DF FF FF F6 FE F7 BB 77 6F 76 EE"    /* ..........ªwov. */
$"DD B6 DF DF FF EF E7 FF 1B F4 FF 01 F8 2F FE FF"    /* .∂........../.. */
$"03 EF DA 7F EF FE FF 09 B7 BB 6E DD DD BB B7 6E"    /* .......ΔΣªn..ªΣn */
$"EF BD E4 FF 1D F4 FF 01 FD 5F FD FF 03 EC BF FF"    /* .Ω......._....ø. */
$"F7 FE FF 0B EE DB BB 76 D6 DA DB BE FF FB FF 7F"    /* ......ªv÷..æ.... */
$"E7 FF 1A F4 FF 01 FE BF FD FF 01 FD 7F FE FF 0A"    /* .......ø........ */
$"EF DF BD F5 B6 DB BD B5 AD DB BD E4 FF 1A F3 FF"    /* ..Ω.∂.Ωμ≠.Ω..... */
$"00 7F FD FF 00 FE FD FF 0C DF F6 EE DB 6D B6 D6"    /* ............mø÷ */
$"EB 77 7D DE FF DF E6 FF 17 EB FF 0D FB FF FE DD"    /* .w}.........¬.... */
$"F7 76 DB 6B 6D B6 DD B7 7B BF FE FF 00 DF E9 FF"    /* .v.kmø.Σ{ø...... */
$"1B 02 FF FF FD EE FF 11 F7 FF DD EE DD DD B5 B6"    /* ............μø */
$"DB 6B 6E DD DD EF F7 FF FF FE E9 FF 19 02 FF FF"    /* .kn............. */
$"F9 EC FF 0F F7 7B B6 EE DB 6F 6D B6 DB B6 EE DB"    /* .....{∂..omø.∂.. */
$"DF FF FF F7 E9 FF 1C 02 FF FF F1 EF FF 0E F7 7F"    /*  */
$"7F FD BD ED B7 76 AB B6 DD AD AD B5 BE E8 FF 02"    /* ..Ω.Σv´∂.≠≠μæ... */
```

```
$"BF FF FF 1D 02 FF FF F5 EF FF 0F BF FE FD BF FF"    /* ø.........ø..ø. */
$"BB 6D DB 76 DB 6B 5A DA EF 6F 76 E9 FF 02 5F FF"    /* ªm.v.kZ..ov..._. */
$"FF 1E F2 FF 01 FC 7F FB FF 0E FB 7E DB 6E DB 6D"    /* ..........~.n.m */
$"AB 6D B6 ED 57 76 DD FB DF EB FF 03 FE 97 FF FF"    /* ´m∂.Wv.......ó.. */
$"21 F2 FF 00 F8 FD FF 12 BE FF FF FD F7 76 DB B6"    /* !.......æ....v.∂ */
$"DB 76 DA DB 56 AD BB B7 EE EF BF EC FF 03 FD 0F"    /* .v..V≠ª∑..ø..... */
$"FF FF 26 F2 FF 01 FD 7F FE FF 16 EE DF FF EF 6D"    /* ..&..........m */
$"DD B6 DB 6D AB 6D 6D BB 5B 6E ED BF BB FF DF FF"    /* .∂.m´mmª[n.øª... */
$"FF 7F F0 FF 03 FA 17 FF FF 24 F2 FF 00 FE FC FF"    /* .........$..... */
$"11 AF FF FE FB 6B 6B 6D B6 DD B5 B6 D6 B6 D7 5B"    /* .Ø...kkm∂.µ∂÷∂◊[ */
$"6D DD FB FE FF 00 7F F0 FF 03 FC 1F FF FF 1C EC"    /* m.............. */
$"FF 12 8F FB F7 DD DD B6 D6 DB AB 5B 5A BB 6D BA"    /* ..è....∂÷.´[Zªm∫ */
$"B6 DB 77 BD BF EC FF 02 BF FF FF 1C EC FF 12 B7"    /* ∂.wΩø...ø.....Σ */
$"7D EF EE EE DB 6B 6D 76 ED AD 6D B6 D7 6D B6 DB"    /* }....kmv.≠m∂◊m∂. */
$"FF EF EC FF 02 7F FF FF 1B F8 FF 00 BF F7 FF 11"    /* ...........ø... */
$"FE EE FF FB 75 B5 B5 B6 B6 DB 5A D6 D6 DB 6D 5B"    /* ....uµµ∂∂.Z÷÷.m[ */
$"6D B7 E7 FF 1D F8 FF 00 DF F7 FF 13 DF FD FD DF"    /* mΣ............. */
$"BB 6B 5B 5B 6D 56 AD 6D 6B 6D B6 ED BB 6D BF FB"    /* ªk[[mV≠mkm∂.ªmø. */
$"E9 FF 23 F8 FF 02 BF FF FB FA FF 17 FE FF BF BB"    /* ..#...ø.......øª */
$"76 ED DD B5 B5 AA DB 6B AA AD B6 D5 B6 DD DE EE"    /* v..µµ™.k™≠∂′∂... */
$"DF DF FB 7F EC FF 21 F8 FF 02 DF FF F3 F8 FF 15"    /* ......!........ */
$"7B FF EF 76 B6 D6 D6 D5 B5 56 D5 76 D5 6E DB 6E"    /* {..v∂÷÷′µV′v′n.n */
$"ED DB B7 FF FE 7F EC FF 1F F6 FF 00 F5 F8 FF 07"    /* ..Σ............ */
$"DF 6E DB BB 6B 5B 6A AE FE AA 0A AB 6E DB 6D DB"    /* .n.ªk[jÆ.™.´n.m. */
$"BB 76 EF 7F FD 7F EC FF 1D F6 FF 00 FB F7 FF 06"    /* ªv............. */
$"BB ED D6 D6 AD B5 B5 FD 55 09 AB 6D B6 B6 DD FF"    /* ª.÷÷≠µµ.UΔ´m∂∂.. */
$"7B FF FF BF EC FF 1A ED FF 14 BF 77 EE DB 6D BA"    /* {..ø......øw..m∫ */
$"56 D6 D6 AA AD AA AD 76 B5 6D DD B7 6D DF BF EA"    /* V÷÷™≠™≠vµm.∑m.ø. */
$"FF 1B EE FF 0A FE FD DD DB B6 DB 6D 5B 6D 5A B5"    /* .........∂.m[mZµ */
$"FE 55 07 AB 5B B5 6B 6D DB 77 7F EA FF 1B ED FF"    /* .U.´[µkm.w..... */
$"09 FB F7 FD ED B6 B6 B5 AA AB 56 FE AA 08 AD 6D"    /* Δ....∂∂µ™´V.™.≠m */
$"5B B6 DA ED DD FB FB EB FF 1A EC FF 09 BF 7E DB"    /* [∂.........Δø~. */
$"DB 6D 5A B5 55 5A D5 FE 55 07 AA AA DB 6D B3 7B"    /* .mZµUZ′.U.™™.m≥{ */
$"FF F7 EB FF 17 ED FF 07 FD FF DD B6 B5 B6 ED 6A"    /* ..........∂µ∂.j */
$"FB AA 06 B5 6D 6A D6 D5 DF 7D EA FF 14 EC FF 06"    /* .™.µmj÷′.}..... */
$"FE EE DB 5B 6B 56 B5 FA 55 04 AA B7 6B 6B 76 E9"    /* ...[kVµ.U.™∑kkv. */
$"FF 1E FE FF 00 9F F3 FF 09 F7 FF FB FF F7 6D AD"    /* .....ü..Δ.....m≠ */
$"B5 AB 56 FA AA 07 B5 55 B6 DD DD FF FF FE EC FF"    /* µ´V.™.µU∂..... */
$"1C FE FF 00 3F F1 FF 08 FE FB 6D DB 5B 5A DA AA"    /* ....?.....m.[Z.™ */
$"D5 FB 55 06 56 AE DB 6B 6E F7 FE EB FF 1A EC FF"    /* ′.U.VÆ.kn..... */
$"05 BF BB 6D B6 EB 6D F9 55 0B 6A D5 6D B6 DB AE"    /* .øªm∂.m.U.j′m∂.Æ */
$"F7 FF EF FF FF FE F0 FF 1C F2 FF 00 DF FB FF 03"    /* ............... */
$"6E DB 6D AD F7 AA 0B AD AB 6D B6 FB FF FF D7 FF"    /* n.m≠.™.≠´m∂...◊. */
$"FF FD 7F F1 FF 1C F2 FF 00 AF FD FF 04 FB F6 DD"    /* .........Ø..... */
$"B6 DB F5 55 0A 5D AD 6D BF BF FE FF F7 FF FE 7F"    /* ∂..U.]≠møø..... */
$"F1 FF 1C F2 FF 00 8F FC FF 03 BB EC DB 6D F6 AA"    /* ......è...ª..m.™ */
$"0B AB 6B 76 DB 76 FF FF F7 EF FF FD BF F1 FF 18"    /* .´kv.v......ø... */
$"F2 FF 00 F7 FD FF 05 76 EE DB 2D B6 B5 F6 55 05"    /* .......v..-∂µ.U. */
$"B6 DD B6 DF F7 BF EC FF 1B F2 FF 09 DF FF FF FE"    /* ∂.∂..ø.....Δ.... */
$"F7 ED BB B5 76 A9 FD 55 00 54 FA AA 04 AB 6B 6B"    /* ..ªµv©.U.T.™.´kk */
$"7D BE EB FF 19 EE FF 06 FB BB D6 DE AA D5 56 FD"    /* }æ.......ª÷.™′V. */
$"AA 00 A5 FB 55 05 6D B6 D5 DF 7F BF EC FF 22 F0"    /* ™.•.U.m∂′..ø..". */
$"FF 08 FE EF F7 EE FB 6B 76 AD 6A FD AA 0C 4A 94"    /* .......kv≠j.™.Jî */
$"95 55 55 5A AB 56 DB 6A ED FF F7 FD FF 00 FE F1"    /* ïUUZ´V.j........ */
$"FF 27 F6 FF 00 EF FE FF 08 FB FF FF D7 DD 77 6E"    /* .'..........◊.wn */
$"DD AB FD 55 04 54 A5 55 4A 52 FE AA 05 D5 AD 6D"    /* .´.U.T•UJR.™.′≠m */
```

```
$"A2 DF 6F FC FF 00 FB F1 FF 23 F6 FF 00 EB FC FF"    /* ¢.o......#..... */
$"08 FE EF FD BD DB B6 D5 AD 6A FD AA 02 2A 24 8A"    /* ....Ω.∂′≠j.™.*$ä */
$"FC 55 03 B6 C1 77 BF FC FF 00 F9 F1 FF 25 F6 FF"    /* .U.∂¡wø......%.. */
$"00 C7 FD FF 08 ED FF D7 EE EF 77 77 6E D5 FE 55"    /* .«.....◊..wwn′.U */
$"06 54 94 88 A9 24 95 52 FE AA 05 DB 52 BD EF FE"    /* .Tîà©$ïR.™..RΩ.. */
$"FD ED FF 24 F6 FF 00 DB FC FF 07 FD EF 7B 76 DD"    /* ...$.........{v. */
$"DA DB 56 FE AA 0F 91 22 22 92 49 25 2A AA AD 55"    /* ..V.™.ë""íI%*™≠U */
$"6D A1 6E FB FF 7F ED FF 98 00 40 00 90 00 00 00"    /* m°n.....ò.@.ê... */
$"C0 02 00 00 90 00 00 00 C0 02 00 00 90 00 00 00"    /* ¿...ê...¿...ê... */
$"C0 02 00 00 01 23 F6 FF 00 EF FB FF 06 FD F7 DD"    /* ¿....#.......... */
$"B6 B5 55 6A FE AA 05 54 49 49 24 92 49 FE 55 06"    /* ∂µUj.™.TII$íI.U. */
$"AA B6 D6 DB DE FB DD ED FF 22 F1 FF 07 F7 FF FE"    /* ™∂÷.....".....  */
$"DF 7B DB 6D AB FE 55 11 52 AA 92 12 49 24 94 A5"    /* .{.m´.U.R™í.I$î• */
$"4A AA B5 6B 55 B6 ED EE F7 EF EE FF 23 F6 FF 00"    /* J™µkU∂......#... */
$"EF FA FF 04 BB DE ED B6 DD FE AA 07 94 88 48 A4"    /* ....ª..∂..™.îàH§ */
$"12 49 22 4A FE AA 06 AD AB 6D BB BB BD DF EE FF"    /* .I"J.™.≠´mªªΩ... */
$"23 F6 FF 00 DD FC FF 05 7E FD FF 75 B6 DB FD 55"    /* #.......~..u∂..U */
$"07 2A A2 92 00 A0 92 49 29 FE 55 05 56 B5 B6 EE"    /* .*¢í.†íI).U.Vµ∂. */
$"F7 77 ED FF 25 F1 FF 1F FD FF BB DD DF 6D B5 B6"    /* .w..%.....ª..mµ∂ */
$"AA AA A9 52 94 20 A9 05 04 92 44 95 55 55 AA DB"    /* ™™©Rî ©..íDïUU™. */
$"6D BB BD DD BB FB DF EF F1 FF 22 EF FF 05 EF 77"    /* mªΩ.ª.....".....w */
$"B6 DB 5A DB FE 55 0F 08 21 4A 02 10 28 24 92 25"    /* ∂.Z..U..!J..($í% */
$"2A AA B5 AA B6 DD DB FE 77 00 7B F0 FF 25 F1 FF"    /* *™µ™∂...w.{..%.. */
$"07 FE FF BF BD ED AD AD 6A FE AA 08 65 4A 10 48"    /* ...øΩ.≠≠j.™.eJ.H */
$"40 42 89 25 4A FE AA 08 D5 5B 6E ED DD DD FF FF"    /* @Bâ%J.™.′[n..... */
$"BF F1 FF 22 F0 FF 12 BB FF EE DB 76 DA B5 55 55"    /* ø.."...ª...v.µUU */
$"49 10 90 80 90 04 00 22 48 95 FE 55 06 5A ED B5"    /* I.êÄê.."Hï.U.Z.µ */
$"B7 77 77 7F EF FF 22 EF FF 18 7E F7 B6 DB 56 D5"    /* ∑ww..."...~.∂.V′ */
$"55 55 24 A5 01 15 01 10 8A 48 12 28 A5 55 55 AB"    /* UU$•....äH.(•UU´ */
$"5B 6E DB FE BB 00 FE F0 FF 23 F1 FF 13 FB DF EF"    /* [n..ª....#..... */
$"FD ED B6 D5 6A AA AA A9 08 24 00 00 21 00 01 44"    /* ..∂′j™™©.$..!..D */
$"85 FD 55 06 AD B7 6D DD EE EF 77 F0 FF 23 F0 FF"    /* Ö.U.≠∑m...w..#.. */
$"1D F7 FD DF 7B 6D AD AD 55 54 84 52 80 00 02 00"    /* ....{m≠≠UTÑRÄ... */
$"00 82 08 28 95 55 55 6A B6 DA DB 6D DD DD DF F0"    /* .Ç.(ïUUj∂..m.... */
$"FF 24 F1 FF 1E FB FF BF 6D DE DB 76 B5 55 55 2A"    /* .$.....øm..vµUU* */
$"84 20 90 10 00 42 04 21 42 49 2A AA AA DB 55 B6"    /* Ñ ê..B.!BI*™™.U∂ */
$"DB 77 77 7B F0 FF 1F F0 FF 0C F7 EE FB BB 6E AB"    /* .ww{.....ªn´ */
$"55 55 54 90 28 44 01 FE 00 03 10 04 14 AA FD 55"    /* UUTê(D.....™.U */
$"02 BB 5B 6D ED FF 1E F0 FF 0B EF FF BE ED B6 DD"    /* .ª[m.....æ.∂. */
$"5A 95 49 25 40 80 FC 00 03 40 A1 11 4A FE AA 02"    /* ZïI%@Ä...@°.J.™. */
$"D5 B6 DF ED FF 21 00 7F F0 FF 04 76 DB 76 DB 6A"    /* ′∂...!.....v.v.j */
$"FE AA 01 48 02 FE 00 0C 10 00 00 04 04 45 2A AA"    /* .™.H.....E*™ */
$"AA D5 5B 6D BF ED FF 1C F1 FF 05 BD DD DD F7 BB"    /* ™′[mø.....Ω...ª */
$"6B FE 55 02 52 91 48 FA 00 02 49 12 52 FE 55 01"    /* k.U.RëH...I.R.U. */
$"6D B6 EC FF 1B 00 FD F3 FF 06 FE FF 7B BB 6D D6"    /* m∂.....{ªm÷ */
$"D5 FE AA 01 94 22 F8 00 00 4A FC AA 00 DB EC FF"    /* ′.™.î"...J.™.... */
$"1E 00 F8 F2 FF 0A FB EF EE F6 BB 6A B5 55 55 25"    /* .....ªjµUU% */
$"44 FA 00 08 02 09 10 92 AA AA AD 55 AF EC FF 1D"    /* D....Δ.í™™≠UØ... */
$"01 FF 0F F3 FF 0A FE FB B7 BB DD AB 55 55 54 A8"    /* ........∑ª.´UUT® */
$"08 F9 00 06 10 45 25 55 55 54 01 EB FF 1B 01 C0"    /* .....E%UUT.....¿ */
$"1F F3 FF 0A EF EE ED DB 6A D5 6A AA A9 02 90 F8"    /* ........j′j™©.ê. */
$"00 04 88 48 A5 55 57 EA FF 17 F1 FF 09 FE FB 7B"    /* ..àH•UW.....Δ..{ */
$"B6 B5 5A AB 55 4A 54 F7 00 04 12 92 95 55 1F EA"    /* ∂µZ´UJT....íïU.. */
$"FF 17 F1 FF 04 DD F7 DE DB 5A FE AA 01 A8 80 F7"    /* .........Z.™.®Ä. */
$"00 04 04 24 AA A8 7F EA FF 16 F2 FF 09 FE F7 BD"    /* ...$™®......Δ..Ω */
$"B7 6B AB 55 55 54 22 F6 00 03 20 89 29 53 E9 FF"    /* ∑k´UUT"... â)S.. */
```

```
$"18 F3 FF 0B FE EF 7B 77 6B B5 55 6A AA A9 48 90"    /* ......{wkµUj™©Hê */
$"F7 00 03 01 22 4A 9F E9 FF 15 F2 FF 09 FD EE DD"    /* ....."Jü.....Δ... */
$"B6 DB 5A AA AA 52 22 F5 00 02 08 92 7F E9 FF 15"    /* ∂.Z™™R"....í.... */
$"F3 FF 0A DF BB DF B6 DB 6D 6A AD 49 84 88 F5 00"    /* ....ª.∂.mj≠IÑà.. */
$"01 02 25 E8 FF 15 F2 FF 0A F6 FB 6B FF FF EA AA"    /* ..%.....k...™ */
$"B5 29 20 20 F6 00 01 08 0F E8 FF 13 F2 FF 01 EF"    /* µ)  ..... */
$"BE FE FF 04 FE 55 52 92 42 F5 00 00 03 E7 FF 12"    /* æ....URíB..... */
$"F2 FF 01 FB BF FD FF 03 FA AC A4 88 F7 00 00 03"    /* ....ø....¨§à.... */
$"E5 FF 0A EB FF 00 E8 F5 00 00 0F E5 FF 08 E7 FF"    /* ..... */
$"00 A0 F9 00 E4 FF 0A E7 FF 00 FE FA 00 00 0F E4"    /* .†..... */
$"FF 0A E6 FF 00 E0 FC 00 00 03 E3 FF 0A E6 FF 00"    /* ..... */
$"E8 FC 00 00 1F E3 FF 0A E6 FF 00 FE FC 00 00 7F"    /* ..... */
$"E3 FF 08 E5 FF 00 80 FD 00 E2 FF 0A E5 FF 00 C0"    /* ......Ä.....¿ */
$"FE 00 00 03 E2 FF 0A E5 FF 00 F0 FE 00 00 03 E2"    /* ..... */
$"FF 0A E5 FF 00 FE FE 00 00 0B E2 FF 09 E4 FF 03"    /* .....Δ... */
$"80 00 00 15 E2 FF 09 E4 FF 03 E0 00 00 2F E2 FF"    /* Ä.....Δ...../.. */
$"09 E4 FF 03 D4 00 00 5F E2 FF 09 E4 FF 03 FA 00"    /* Δ...'.._..Δ..... */
$"00 AF E2 FF 09 E4 FF 03 FA 80 00 5F E2 FF 09 E4"    /* .Ø..Δ....Ä._..Δ. */
$"FF 03 FD 00 03 BF E2 FF 08 E3 FF 02 D0 05 7F E2"    /* .....ø.....-... */
$"FF 07 E3 FF 01 A8 0A E1 FF 07 E3 FF 01 70 2B E1"    /* .....®.....p+. */
$"FF 98 00 40 00 C0 00 00 00 F0 02 00 00 C0 00 00"    /* .ò.@.¿.....¿.. */
$"00 F0 02 00 00 C0 00 00 00 F0 02 00 00 01 07 E3"    /* .....¿..... */
$"FF 01 DE AF E1 FF 07 E3 FF 01 F5 7F E1 FF 06 E3"    /* ...Ø..... */
$"FF 00 FE E0 FF 06 E3 FF 00 FD E0 FF 02 C1 FF 02"    /* .....¡.. */
$"C1 FF 06 E3 FF 00 F5 E0 FF 07 E3 FF 01 FB F7 E1"    /* ¡..... */
$"FF 06 E3 FF 00 F5 E0 FF 07 E3 FF 01 EF FB E1 FF"    /* ..... */
$"07 E3 FF 01 EF FD E1 FF 07 E3 FF 01 CF FE E1 FF"    /* .....œ... */
$"08 E3 FF 02 9F FF 7F E2 FF 07 E3 FF 01 3F FE E1"    /* ....ü.....?.. */
$"FF 08 E4 FF 02 FE BF FE E1 FF 09 E4 FF 03 FE BF"    /* ......ø...Δ....ø */
$"FF 5F E2 FF 09 E4 FF 03 FE 3F FF 5F E2 FF 09 E4"    /* ._..Δ....?._..Δ. */
$"FF 03 FD 7F FF AF E2 FF 09 E4 FF 03 FC BF FF D7"    /* .....Ø..Δ....ø.◊ */
$"E2 FF 09 E4 FF 03 FA BF FF ED E2 FF 09 E4 FF 03"    /* ..Δ....ø....Δ... */
$"F5 FF BF DB E2 FF 09 E4 FF 03 F5 FF FF ED E2 FF"    /* ..ø...Δ..... */
$"0D E9 FF 00 BF FD FF 03 FA FF FF F6 E2 FF 09 E4"    /* ¬...ø.....Δ. */
$"FF 03 F5 FF BF FB E2 FF 0D EA FF 00 F7 FC FF 03"    /* ....ø...¬..... */
$"C1 FF FF FA E2 FF 0D EA FF 00 DA FC FF 03 AB FF"    /* ¡.....¬.....´. */
$"FF FB E2 FF 0E EA FF 00 7F FC FF 04 46 FF BF FE"    /* .....F.ø. */
$"DF E3 FF 0F EB FF 00 B6 FC FF 05 FE 9D FF BF FB"    /* .....∂....ù.ø. */
$"6F E3 FF 10 EC FF 01 FB DF FC FF 01 FD 2F FE FF"    /* o...../.. */
$"00 5F E3 FF 0F EC FF 00 F7 FB FF 01 FA 5F FE FF"    /* ._.....¸.. */
$"00 B7 E3 FF 0F EC FF 00 FD FB FF 05 F4 3F FF BF"    /* .∑.....?.ø */
$"FF DB E3 FF 0B E5 FF 05 D1 7F FF BF FF ED E3 FF"    /* .....—-.ø.... */
$"12 EE FF 00 FE FC FF 08 F7 FF FF 44 EF FF BF FF"    /* .....D..ø. */
$"F6 E3 FF 11 EE FF 01 FD F7 FA FF 01 13 BF FE FF"    /* .....ø.. */
$"01 FB 7F E4 FF 13 EE FF 01 AE BF FD FF 03 EB FF"    /* .....Æø..... */
$"FE 8F FD FF 01 DD AD E4 FF 14 EF FF 02 FB 55 DF"    /* .è....≠.....U. */
$"FD FF 03 D5 7F FA 16 FD FF 01 EF 56 E4 FF 16 F0"    /* ...'.....V.... */
$"FF 02 F6 D5 BF FC FF 0A EF FF E0 FB FF FF 3F FF"    /* ...'ø.....?. */
$"FF ED BF E5 FF 15 F0 FF 01 DB 6B FB FF 02 F7 FF"    /* ..ø.....k..... */
$"CA FE FF 00 BF FE FF 00 5F E5 FF 16 F1 FF 02 AA"    /*  ...ø..._.....™ */
$"AD BF FB FF 02 5F FA 17 FE FF 00 3F FE FF 00 AB"    /* ≠ø..._.....?...´ */
$"E5 FF 15 F1 FF 01 DB 5A FA FF 02 7F F8 AE FE FF"    /* .....Z.....Æ.. */
$"00 3F FE FF 00 DA E5 FF 13 F1 FF 00 6D FA FF 03"    /* .?.....m... */
$"FB FF A3 F7 FB FF 02 FB FD B7 E6 FF 14 F2 FF 01"    /* ..£.....Σ..... */
$"DA B7 F9 FF 02 FE 8D DB FB FF 03 FE FF 55 7F E7"    /* .Σ....ç.....U.. */
$"FF 1B F3 FF 01 DF B7 FC FF 06 DF 7E EF FF F8 5F"    /* .....Σ....~..._ */
```

```
$"6F FE FF 00 BF FE FF 02 DF FA AF E7 FF 1B F3 FF"    /* o...ø.....Ø..... */
$"01 FE DF FD FF 06 FB 6E FF DB FF A3 FA FD FF 00"    /* .......n...£.... */
$"3F FE FF 02 EB FD 55 E7 FF 1D F4 FF 01 DF DB FD"    /* ?.....U......... */
$"FF 07 FD 6D DB FF 5F FE 0D EB FD FF 00 3F FE FF"    /* ...m.._.¬....?.. */
$"03 FD 7F EA BF E8 FF 1D F5 FF 02 F6 FD 6F FD FF"    /* ....ø........o.. */
$"07 FB FB EF BE AF F8 2F AF FD FF 00 BF FD FF 02"    /* ....æØ./Ø...ø... */
$"BF F6 DF E8 FF 1E F7 FF 03 F7 F6 BF 6F FC FF 07"    /* ø..........øo... */
$"BF BF 7F FB 7F A1 FD 5F FD FF 00 3F FD FF 02 EF"    /* øø...°._...?.... */
$"FF 57 E8 FF 1F F8 FF 04 EA BF D7 6D BF FD FF 07"    /* .W.......øÒmø... */
$"FA DB DE FF FF FD 0F F4 FC FF 00 3F FD FF 02 F5"    /* ...........?.... */
$"FF AA E8 FF 98 00 40 00 F0 00 00 01 20 02 00 00"    /* .™..ò.@..... ... */
$"F0 00 00 01 20 02 00 00 F0 00 00 01 20 02 00 00"    /* .... ........ ... */
$"01 1B F8 FF 03 DD F6 EF D7 FC FF 07 7F 7F FF FA"    /* ........◊....... */
$"DF E8 5F E3 F7 FF 03 FE BF FD AF E9 FF 18 F6 FF"    /* .._.....ø.Ø..... */
$"00 DB FC FF 08 FD FF FF F7 F7 7E 93 FF 0F F6 FF"    /* ..........~ì.... */
$"02 5F FE B5 E9 FF 1B F9 FF 02 EB FB FD FB FF 08"    /* ._.µ............ */
$"EB FF FF EF FF F8 2F F8 7F F6 FF 03 F7 FF EB 7F"    /* ....../......... */
$"EA FF 1C FB FF 04 FD FD DF FF 7B FB FF 07 EF FF"    /* ..........{..... */
$"DF 7F FF D2 DF EB F5 FF 03 FA FF F6 D7 EA FF 1B"    /* ...".........◊... */
$"FB FF 03 B6 EE FF FD FB FF 08 FB FF 6F B6 FF FF"    /* ...∂.........o∂.. */
$"05 FF C5 F4 FF 02 AF FF 5A EA FF 23 FD FF 05 FE"    /* ..≈...Ø.Z..#.... */
$"F6 DD F7 7F BF FC FF 09 F7 F7 7F DF 7F F7 FA 2F"    /* ....ø..Δ......./ */
$"FF 1F FB FF 00 1F FB FF 03 EF FF EA DF EB FF 22"    /* ..............." */
$"FD FF 04 4B 7B FF FE EA FB FF 08 DB FB EF FF FF"    /* ...K{........... */
$"EF E0 BF F8 FA FF 00 BF FB FF 04 FA FF F5 B6 DF"    /* ..ø....ø......∂. */
$"EC FF 23 FE FF 05 FB 57 FF FF F6 F7 FC FF 09 FE"    /* ..#....W......Δ. */
$"EF FD FB FF FB FF 8B 7F E3 FA FF 00 5F FA FF 03"    /* ......ã....._... */
$"5F FE DB B7 EC FF 23 07 FF FF EF F6 FF FE DD 5B"    /* _..Σ..#........[ */
$"FB FF 09 EF 77 FF DF FB 7F F4 2B FF 8F FA FF 00"    /* ..Δ.w.....+.è... */
$"1F FA FF 03 EB FF ED 5E EC FF 22 06 FF FF F6 ED"    /* .......^.."..... */
$"FF EF B6 FA FF 09 7F FB FF 6F EE FF E9 5F FD 3F"    /* ..∂..Δ...o..._.? */
$"FA FF 00 3F FA FF 03 F5 FF FA AD EC FF 22 06 FE"    /* ...?......≠.."... */
$"FF FF FB FB F7 7F FA FF 02 FB FD DF FE FF 02 A2"    /* ..............¢ */
$"FF F0 F9 FF 00 1F FA FF 04 FE BF FF DB 6F ED FF"    /* ..........ø..o.. */
$"21 05 FD FF FF F7 B6 FB F9 FF 08 F7 FF FD BF FF"    /* !.....∂.......ø. */
$"F5 0D FF C3 F9 FF 00 BF F9 FF 04 AF FF ED D5 7F"    /* .¬.√...ø...Ø..'. */
$"EE FF 1D 04 FF FF DD BD DB F9 FF 09 B7 FF FF FE"    /* .......Ω...ΔΣ... */
$"FF FF 40 5F FE 1F F0 FF 04 D5 FF FE AE DF EE FF"    /* ..@_.....'..Æ... */
$"1B 04 FF F7 F6 EE EF F9 FF 00 7B FC FF 03 0A BF"    /* ..........{....ø */
$"FC 3F F0 FF 04 FD 7F FF DB 6F EE FF 1A 02 FF BF"    /* .?.......o.....ø */
$"FB F8 FF 09 EF F7 FF FF FD FF E8 57 FF E1 EE FF"    /* ...Δ.......W.... */
$"03 5F FF FD DF EE FF 1B 02 FF ED 56 F8 FF 09 76"    /* ._..........V..Δv */
$"DF 7F FF FE FE C2 BF FF 87 EE FF 04 D7 FF FE ED"    /* .....¬ø.á...◊... */
$"F7 EF FF 1D 02 FF BB BB F9 FF 0A FD AD 6F FD FF"    /* ......ªª....≠o.. */
$"6F FD 05 FF FC 1F EE FF 05 EA FF FF DE BF DF F0"    /* o............ø.. */
$"FF 1C 02 FF FF FE F8 FF 09 77 BF DF F7 B7 F8 13"    /* ........Δwø..Σ.. */
$"FF F8 7F EE FF 05 FD 7F FF FD D7 B7 F0 FF 1A F5"    /* ..........◊Σ.... */
$"FF 08 FB 6A BA BF FF A2 2F FF C3 F8 FF 01 FE 1F"    /* ...jʃø.¢/.√..... */
$"F7 FF 03 AF FF FE AD EF FF 1A F4 FF 07 A5 F5 FF"    /* ...Ø..≠.....•... */
$"FD 08 DF FF 07 F8 FF 01 FE 3F F7 FF 04 D7 FF FF"    /* .........?...◊.. */
$"FA DF F0 FF 1B F4 FF 07 DF FB FF F2 73 FF FC 3F"    /* ............s..? */
$"F8 FF 01 FE 1F F7 FF 05 F5 7F FF F5 B7 BD F1 FF"    /* ............ΣΩ.. */
$"17 F1 FF 03 45 57 FF F0 F7 FF 01 FE 1F F7 FF 05"    /* ....EW.......... */
$"FE BF FF FE ED BB F1 FF 18 F2 FF 04 FA 13 BF FF"    /* .ø...ª........ø. */
$"83 F7 FF 01 FE 1F F6 FF 05 AF FF FF B6 F7 BF F2"    /* É........Ø..∂.ø. */
$"FF 1D F7 FF 09 EF FF BB FF FF E8 A5 7F FF 07 F7"    /* ....Δ..ª...•.... */
```

```
$"FF 01 FE 1F F6 FF 05 F5 FF FF EF BF F7 F2 FF 1E"    /* ..........ø.... */
$"F7 FF 09 F7 FE D7 FF FF 42 15 FF FC 2F F7 FF 01"    /* ..Δ..◊..B.../... */
$"FE 3F F6 FF 06 FE BF FF FF DB BF BF F3 FF 1A F6"    /* .?....ø...øø.... */
$"FF 07 F7 7F FF FA 11 6F FF F0 F6 FF 01 FE 1F F5"    /* .......o........ */
$"FF 04 57 FF FF 6D ED F2 FF 18 F3 FF 04 D0 96 BF"    /* ..W..m.......-ñø */
$"FF 83 F6 FF 01 FE 3F F5 FF 05 D7 FF FF F7 77 7B"    /* .É....?...◊...w{ */
$"F3 FF 1C F9 FF 01 FD BF FD FF 04 42 2F FF FE 07"    /* .......ø...B/... */
$"F5 FF 00 1F F5 FF 05 F9 FF FF FD BF F7 F3 FF 15"    /* ..........ø.... */
$"F4 FF 05 F4 15 DF FF F8 3F E8 FF 06 FE BF FF FF"    /* ........?....ø.. */
$"6B 6F 7F F4 FF 14 F4 FF 05 E8 AA EF FF F0 7F E7"    /* ko........™..... */
$"FF 05 AF FF FF F5 DF B7 F4 FF 1A FA FF 0A FB FF"    /* ..Ø....Σ........ */
$"FF DA FF FF 41 2B FF FF 83 E6 FF 06 EA FF FF FE"    /* ....A+..É....... */
$"BD DF BF F5 FF 14 F5 FF 05 FA 8A 57 FF FD 0F E6"    /* Ω.ø.......äW.... */
$"FF 00 FA FE FF 01 EB 6F F4 FF 15 F5 FF 05 E9 20"    /* .......o........ */
$"AF FF F8 3F E5 FF 06 5F FF FF F6 DF F5 BF F6 FF"    /* Ø..?..._.....ø.. */
$"15 F6 FF 05 FD 42 55 7F FF E0 E4 FF 00 AB FE FF"    /* .....BU......´.. */
$"02 6F EF 77 F6 FF 15 F6 FF 05 FA 14 AB FF FF 83"    /* .o.w........´..É */
$"E4 FF 00 F4 FE FF 02 DF F7 EF F6 FF 15 F7 FF 06"    /* ................ */
$"FE A1 41 5F FF EE 07 E4 FF 05 FD 5F FF FF F7 FE"    /* .°A_......._.... */
$"F5 FF 15 F6 FF 05 4A 2A BF FF F8 3F E3 FF 06 4F"    /* ......J*ø..?...O */
$"FF FF FE DF F7 BF F7 FF 17 F7 FF 06 FA 90 85 FF"    /* .....ø.......êÖ. */
$"FF E0 7F E3 FF 00 F5 FE FF 03 EF FB 6D BF F8 FF"    /* ............mø.. */
$"18 FA FF 08 EF FF FF A0 25 2F FF FF 43 E2 FF 06"    /* .......†%/..C... */
$"FA 7F FF FF FB FD DB F7 FF 18 FA FF 08 F7 FF FE"    /* ................ */
$"95 55 5F FF FE 0F E1 FF 00 9F FE FF 02 77 7F EF"    /* ïU_......ü...w.. */
$"F8 FF 16 F8 FF 06 FA 20 AA FF FF F0 3F E1 FF 00"    /* ....... ™...?... */
$"E3 FE FF 02 FB BB DF F8 FF 14 F8 FF 05 E8 0B 4F"    /* .....ª.........O */
$"FF FF C0 E0 FF 05 F5 7F FF FF FE EF F7 FF 19 F8"    /* ..¿............. */
$"FF 05 42 55 3F FF FF 83 F2 FF 01 FD 0F F1 FF 01"    /* ..BU?..É........ */
$"FE BF FE FF 00 77 F7 FF 1D FB FF 08 FD FF F8 10"    /* .ø...w.......... */
$"9A FF FF FC 1F F2 FF 01 FC 2F F0 FF 00 4F FE FF"    /* ö......../...O.. */
$"02 EE FF EF F9 FF 18 FC FF 09 F7 7F FF E9 45 4F"    /* .........Δ....EO */
$"FF FF F0 3F F2 FF 01 FC 0F F0 FF 00 F2 F3 FF 0B"    /* ...?............ */
$"F9 FF 05 42 50 5F FF FF E0 CF FF 0C FA FF 06 FC"    /* ...BP_...œ...... */
$"14 A5 FF FF FE 07 CF FF 31 FD FF 09 B7 FF FF EA"    /* .•....œ.1..ΔΣ... */
$"49 5B FF FF FC 0F FB FF 00 C3 FC FF 00 87 FD FF"    /* I[.......√...á.. */
$"16 E1 FF FF FE 1F FF C7 FF FF 87 FF FF F8 FE 1F"    /* .......«..á..... */
$"FF FF F0 FF FF C3 FF C7 F7 FF 98 00 40 01 20 00"    /* .....√.«..ò.@. . */
$"00 01 50 02 00 01 20 00 00 01 50 02 00 01 20 00"    /* ..P... ...P... . */
$"00 01 50 02 00 00 01 36 FD FF 09 DF FF FF 44 36"    /* ..P....6..Δ...D6 */
$"AF FF FF F0 3F FD FF 0D F8 FF E3 FF FE 3F FF FF"    /* Ø...?..¬.....?.. */
$"C7 FF F8 FF FF F1 FE FF 12 1F FF 87 FF FF C7 FF"    /* «..........á..«. */
$"FF F0 FF 1F FF FF F8 FF FF E3 FF 87 F7 FF 33 FB"    /* ............á..3. */
$"FF 06 BA 08 AF 7F FF FF 80 FC FF 23 F0 FC 63 FF"    /* ..∫.Ø...Ä..#..c. */
$"FE 3F FF FF C7 FF F8 8C 7F F1 F1 FF E3 1F FF 9F"    /* .?..«..å.......ü */
$"FF FF C4 7F FF F3 E3 1F FF FF F8 FF FF E3 FF 9F"    /* ..ƒ............ü */
$"F7 FF 32 FC FF 07 FE 80 AB 5F FF FF FE 07 FC FF"    /* ..2....Ä´_...... */
$"02 E0 FC 47 FC FF 1B 8F FF FF 8C 7F E3 F1 FF E2"    /* ...G...è..å..... */
$"3F FF 1F FF FF 8C 7F FF E3 E2 3F FF FF F1 FF FF"    /* ?....å....?..... */
$"C7 FF 1F F7 FF 34 FC FF 07 FE 2A 15 A7 FF FF F8"    /* «....4....*.ß... */
$"1F FC FF 23 C8 F0 00 E1 80 61 F0 00 82 00 20 00"    /* ...#»...Äa..Ç. . */
$"21 83 C0 87 80 07 0E 08 30 00 80 00 78 41 80 07"    /* !É¿áÄ...0.Ä.xAÄ. */
$"0F 86 10 38 70 C7 C2 0F F7 FF 37 01 FD 5E FE FF"    /* .Ü.8p«¬...7..^.. */
$"02 E8 84 AA FE FF 01 C0 7F FC FF 23 98 F8 C4 41"    /* ..Ñ™...¿...#ò.ƒA */
$"04 41 E0 88 81 11 11 18 C1 03 E3 03 C6 22 0F 10"    /* .A.àÅ...¡...Δ".. */
$"20 88 88 C4 F0 23 C6 22 0F 04 11 10 20 47 81 1F"    /*  ààƒ.#Δ".... GÅ. */
```

```
$"F7 FF 36 07 FF BF BF FF FF D2 51 55 FE FF 00 C0"    /* ..6..øø..“QU...¿ */
$"FB FF 23 80 F8 C4 01 1C 47 E3 88 81 11 11 18 80"    /* ..#Ä.ƒ..G.àÅ...Ä */
$"23 E2 23 C6 20 0F 02 07 88 88 C1 E2 23 C6 20 0F"    /* #.#Δ ...àà¡.#Δ . */
$"18 F1 02 04 47 11 1F F7 FF 33 FC FF 06 00 8E FB"    /* ....G....3....é. */
$"FF FF FE 07 FB FF 23 38 F1 88 87 88 E3 F1 11 10"    /* ......#8.àáà.... */
$"22 22 31 86 07 C6 23 8C 44 3E 20 47 11 11 C1 E2"    /* ""1Ü.Δ#åD> G..¡. */
$"07 8C 44 3F 88 E2 22 04 0F 10 3F F7 FF 37 00 BF"    /* .åD?à."...?..7.ø */
$"FE FF 07 EA 0A 15 6F FF FF F8 1F FC FF 24 FE 78"    /* ......o.....$.x */
$"F0 08 88 08 83 C1 01 02 22 22 00 08 07 C0 07 80"    /* ..à.É¡.""...¿.Ä */
$"44 4E 20 44 01 10 63 E0 47 80 44 4E 08 82 20 40"    /* DN D..c.GÄDN.Ç @ */
$"8F 02 3F F7 FF 36 FE FF 03 FD 43 A2 A1 FE FF 01"    /* è.?..6....C¢°... */
$"E0 7F FC FF 24 FC 30 70 00 42 10 07 C3 00 04 00"    /* ....$.0p.B..√... */
$"00 10 C3 03 C3 0F 80 02 1E 30 21 00 00 E3 F0 C7"    /* ..√.√.Ä..0!....« */
$"80 02 1E 1C 00 10 E1 07 86 3F F7 FF 1D FE FF 03"    /* Ä.......Ü?..... */
$"F5 AA DD 57 FE FF 00 41 E8 FF 01 FE 7F FE FF 02"    /* .™.W...A..... */
$"E7 FF CF F9 FF 01 FE 7F F7 FF 1E 0A F7 FF FF 50"    /* ..œ.....P */
$"11 E9 7F FF FF FC 07 E8 FF 01 FC 7F FE FF 02 0F"    /* ..... */
$"FF 8F F9 FF 01 FC 7F F7 FF 1F FE FF 07 24 AD B6"    /* .è.....$≠∂ */
$"BF FF FF F8 0F E8 FF 00 FC FD FF 02 1F FF 9F F9"    /* ø.....ü. */
$"FF 00 FC FB FF 00 EF FD FF 1D 05 FF FF FA 49 2E"    /* .....I. */
$"DD FE FF 01 E0 7F E9 FF 05 C3 E3 F8 63 80 7F FD"    /* .....√..cÄ.. */
$"FF 02 F1 FC 3F F1 FF 30 05 FF FF 50 94 DB BF FE"    /* ....?..0...Pî.ø. */
$"FF 00 01 FC FF 01 F8 0F FB FF 02 F0 7F 1F FE FF"    /* ..... */
$"00 8F FD FF 06 1F E3 C3 FC 40 44 7F FD FF 02 E1"    /* .è.....√.@D..... */
$"FE 3F FD FF 00 C7 F6 FF 38 04 FF FD 24 22 66 FE"    /* .?...«..8...$"f. */
$"FF 01 FE 07 FC FF 01 F1 8F FD FF 04 1F FF E6 7F"    /* .....è..... */
$"1F FE FF 00 8F FE FF 0E E3 1F E3 CF FC 4C 44 7F"    /* ....è.....œ.LD. */
$"FF FF F1 FF E7 C6 3F FD FF 00 C7 FE FF 00 F8 FA"    /* .....Δ?...«..... */
$"FF 2F 04 FF E8 41 4A CD FE FF 01 F0 1F FC FF 01"    /* ./...AJÕ..... */
$"E3 9F FD FF 03 1F FF C6 7F F9 FF 06 E3 FF C7 8F"    /* .ü.....Δ.....«è */
$"F8 8F 88 FE FF 04 F1 FF C7 C4 7F F9 FF 00 F8 FA"    /* .èà.....«ƒ..... */
$"FF 3B 04 FF 41 09 77 7F FE FF 01 C0 7F FC FF 23"    /* .;..AΔw....¿...# */
$"E3 FC 20 02 04 00 08 01 C3 C0 30 01 C3 0E 00 3C"    /* .. .....√¿0.√..< */
$"0E 00 00 30 47 00 08 00 88 00 1C 00 40 F0 83 00"    /* ...0G...à...@.É. */
$"0E 1C 08 60 FE 00 03 0E 00 20 43 FB FF 37 04 F6"    /* ...`..... C..7.. */
$"92 52 BB BF FE FF 00 81 FB FF 14 C7 F8 11 11 02"    /* íRªø...Å...«.... */
$"22 30 41 E1 82 20 88 82 0F 11 1E 04 08 06 20 47"    /* "0A.Ç àÇ..... G */
$"FB 88 0F 22 23 E0 47 8C 44 1E 00 40 88 88 04 44"    /* .à."#.GåD..@àà.D */
$"11 11 83 FB FF 38 03 EA A8 1D 7F FE FF 01 FC 0F"    /* ..É..8..®..... */
$"FB FF 14 C7 F1 11 11 02 22 20 47 F0 1E 00 88 1C"    /* ...«...." G...à. */
$"0F 11 1E 00 88 C6 04 47 FC 88 10 80 22 23 C4 47"    /* ....àΔ.G.à.Ä"#ƒG */
$"8C 40 1E 20 08 88 88 C4 40 11 11 8F FB FF 37 02"    /* å@. .ààƒ@..è..7. */
$"D1 08 AA FD FF 01 F0 1F FB FF 14 C7 91 02 22 20"    /* —.™.....«ë."  */
$"44 61 8F 98 1C 43 11 1C 3E 22 3C 40 11 8C 40 8F"    /* Daèò.C..>"<@.å@è */
$"FC 11 10 10 C4 47 C4 0F 18 88 7C 61 81 11 11 88"    /* ....ƒGƒ..à|aÅ..à */
$"88 62 23 C7 FB FF 36 02 44 55 75 FD FF 00 80 FA"    /* àb#«..6.DUu...Ä. */
$"FF 15 C2 30 22 22 04 ^4 02 0F 99 10 44 11 10 4E"    /* ..¬0""....ô.D..N */
$"22 3C 08 11 80 40 8F 10 FC 11 0F 04 41 C0 8F 00"    /* "<..Ä@è.....A¿è. */
$"88 9C 62 01 01 11 88 88 82 20 07 FB FF 37 02 B1"    /* àúb...ààÇ ...7.± */
$"0D 3B FD FF 00 41 FA FF 15 E0 78 40 00 0C 00 30"    /* ¬;...A....x@...0 */
$"07 83 80 20 00 86 1C 00 1C 1C 00 80 20 07 10 FD"    /* .ÉÄ .Ü.....Ä ... */
$"00 09 08 00 03 E1 8F 00 04 38 30 C1 FE 00 03 04"    /* .Δ....è..80¡.... */
$"00 01 0F FB FF 22 02 52 7E DF FE FF 01 FC 07 F7"    /* .....".R~..... */
$"FF 01 FC 7F F7 FF 00 F8 FC FF 02 3F FF 3F FB FF"    /* .....?.?.. */
$"00 9F FD FF 00 E3 F4 FF 22 01 54 AB FD FF 01 F0"    /* .ü.....".T´.... */
$"1F F7 FF 01 FC 7F F7 FF 00 F8 FD FF 03 FE 3F FE"    /* .....?. */
```

```
    $"3F FB FF 00 1F FD FF 00 E3 F4 FF 22 01 53 15 FD"    /* ?.........".S.. */
    $"FF 00 A0 F6 FF 01 F8 3F F7 FF 01 F0 7F FE FF 03"    /* ..†....?........ */
    $"FE 7F FE 7F FB FF 00 3F FD FF 00 C1 F4 FF 1C 01"    /* .......?...¡.... */
    $"AD 6B FD FF 00 01 F9 FF 06 F8 FF FF E1 FF FE 1F"    /* ≠k.............. */
    $"FD FF 01 FC 7F FB FF 00 E3 E4 FF 2E 01 B6 BF FE"    /* .............∂ø. */
    $"FF 01 F8 0F F9 FF 06 F0 FF FF F1 FF FF 1F FD FF"    /* ................ */
    $"03 F8 7C 3C 3F FD FF 09 C3 C1 FC 7F FF FF FE 3F"    /* ..|<?..Δ√¡.....? */
    $"FE 03 FB FF 02 FC 1F C7 F6 FF 31 01 DF 7F FE FF"    /* .......«..1..... */
    $"01 F0 1F F9 FF 06 F3 FF FE 31 FF FF 1F FD FF 11"    /* .........1...... */
    $"F9 FE 38 7F FF 1F FF FF CF 99 FC 7F FF FF FE 3F"    /* ..8.....œô.....? */
    $"FC 63 FD FF 04 C7 FF F9 9F C7 F6 FF 2B 00 EF FD"    /* .c...«..ü«..+... */
    $"FF 00 80 F8 FF 06 E3 FF FE 23 FF FE 3F FD FF 09"    /* ..Ä......#..?..Δ */
    $"F1 FE 30 7F FF 1F FF FF 8F 19 FB FF 01 F8 E7 FD"    /* ..0.....è....... */
    $"FF 03 C7 FF F1 9F F5 FF 34 00 BF FD FF 00 01 F8"    /* ..«..ü..4.ø..... */
    $"FF 28 C1 08 18 00 70 F8 38 40 02 18 7C 20 FC 00"    /* .(¡...p.8@..| .. */
    $"E0 C0 08 01 E1 07 0F 00 C0 07 0C 38 00 F8 FF 08"    /* .¿.....¿..8.... */
    $"00 81 00 02 00 70 F0 0C 00 70 C3 F9 FF 37 FD FF"    /* .Å...p...p√..7.. */
    $"01 F8 07 F8 FF 28 E2 04 1C 62 20 F0 30 08 80 10"    /* .....(...b .0.Ä. */
    $"78 11 FC 00 C0 82 30 41 C0 8F 86 08 82 22 08 3C"    /* x...¿Ç0A¿èÜ.Ç".< */
    $"44 71 FE 04 44 40 88 8C 10 78 60 88 22 20 83 FE"    /* Dq..D@àå.x`à" É. */
    $"FF 00 DF FD FF 33 FD FF 01 F0 1F F8 FF 28 E0 44"    /* .....3.......(.D */
    $"7C 62 00 E2 20 08 88 00 71 11 FC 08 88 8E 20 47"    /* |b.. .à.q...àé G */
    $"88 8F C0 78 02 20 70 3C 44 71 FC 44 44 40 88 88"    /* àè¿x. p<Dq.DD@àà */
    $"11 FC 07 80 22 07 03 F9 FF 33 FD FF 01 80 7F F8"    /* ...Ä"....3...Ä.. */
    $"FF 28 C4 40 F8 C4 43 E0 61 C1 18 61 F1 03 F9 11"    /* .(ƒ@.ƒC.a¡.a.... */
    $"81 C4 61 8F 88 1E 60 71 0C 44 70 F8 88 F1 E4 40"    /* Åƒaèà.`q.Dp.à..@ */
    $"88 88 11 18 63 E6 07 10 C4 47 0F F9 FF 33 FE FF"    /* àà..c...ƒG...3.. */
    $"01 FE 01 F7 FF 28 C4 08 F8 04 44 E0 62 4F 18 82"    /* .....(ƒ...D.bO.Ç */
    $"70 23 F9 11 81 04 02 0F 81 1E 64 41 10 44 41 38"    /* p#..Å...Å.dA.DA8 */
    $"88 F0 8C 08 88 81 01 00 83 E6 44 11 04 44 10 F9"    /* à.å.àÅ..É.D..D.. */
    $"FF 33 FE FF 01 F8 07 F7 FF 28 C6 10 78 00 21 F0"    /* .3.......(Δ.x.!. */
    $"30 C0 0C 30 F8 63 F0 20 C0 0C 30 07 C3 1E 0E 00"    /* 0¿.0.c. ¿.0.√... */
    $"80 02 18 70 00 78 1E 10 00 03 00 0C 01 E0 E0 08"    /* Ä..p.x.......... */
    $"00 21 80 F9 FF 1B FE FF 01 E0 1F F7 FF 00 CF FB"    /* .!Ä...........œ. */
    $"FF 00 90 FE FF 00 E7 FA FF 00 3F F6 FF 00 1F F0"    /* ..ê.......?..... */
    $"FF 1C FE FF 01 80 7F F7 FF 00 8F FB FF 00 18 FE"    /* .....Ä....è..... */
    $"FF 00 C7 FB FF 01 FE 3F F6 FF 00 1F F0 FF 1D 03"    /* ..«....?........ */
    $"FF FF FE 03 F6 FF 00 9F FB FF 00 81 FE FF 00 CF"    /* .......ü...Å...œ */
    $"FB FF 01 FE 7F F7 FF 01 FE 0F F0 FF 07 03 FF FF"    /* ................ */
    $"F8 0F C5 FF 07 03 FF FF C0 3F C5 FF 06 02 FF FF"    /* ..≈.....¿?≈..... */
    $"00 C4 FF 06 02 FF FE 01 C4 FF 06 02 FF F8 07 C4"    /* .ƒ......ƒ......ƒ */
    $"FF 06 02 FF E0 1F C4 FF 05 01 FF 00 C3 FF 05 01"    /* ......ƒ.....√... */
    $"FE 01 C3 FF 05 01 F0 0F C3 FF 05 01 C0 1F C3 FF"    /* ..√.....√...¿.√. */
    $"98 00 40 01 50 00 00 01 55 02 00 01 50 00 00 01"    /* ò.@.P...U...P... */
    $"55 02 00 01 50 00 00 01 55 02 00 00 01 04 00 00"    /* U...P...U....... */
    $"C2 FF 04 00 01 C2 FF 04 00 07 C2 FF 04 00 3F C2"    /* ¬....¬....¬...?¬ */
    $"FF 02 C1 FF A0 00 8F A0 00 83 FF"                   /* ..¡.†.è†.É. */
};

data 'CURS' (256, locked, preload) {
    $"3F 00 3F 00 3F 00 3F 00 40 80 84 40 84 40 84 60"    /* ?.?.?.?.@ÄÑ@Ñ@Ñ` */
    $"9C 60 80 40 80 40 40 80 3F 00 3F 00 3F 00 3F 00"    /* ú`Ä@Ä@@Ä?.?.?.?. */
    $"3F 00 3F 00 3F 00 3F 00 7F 80 FF C0 FF C0 FF C0"    /* ?.?.?.?..Ä.¿.¿.¿ */
    $"FF C0 FF C0 FF C0 7F 80 3F 00 3F 00 3F 00 3F 00"    /* .¿.¿.¿.Ä?.?.?.?. */
    $"00 08 00 08"                                        /* .... */
};
```

```
data 'CURS' (257, locked, preload) {
    $"3F 00 3F 00 3F 00 3F 00 40 80 80 40 81 40 82 60"    /* ?.?.?.?.@ÄÄ@Å@Ç` */
    $"9C 60 80 40 80 40 40 80 3F 00 3F 00 3F 00 3F 00"    /* ú`Ä@Ä@@Ä?.?.?.?. */
    $"3F 00 3F 00 3F 00 3F 00 7F 80 FF C0 FF C0 FF C0"    /* ?.?.?.?..Ä.¿.¿.¿ */
    $"FF C0 FF C0 FF C0 7F 80 3F 00 3F 00 3F 00 3F 00"    /* .¿.¿.¿.Ä?.?.?.?. */
    $"00 08 00 08"                                        /* .... */
};

data 'CURS' (258, locked, preload) {
    $"3F 00 3F 00 3F 00 3F 00 40 80 80 40 80 40 80 60"    /* ?.?.?.?.@ÄÄ@Ä@Ä` */
    $"9F 60 80 40 80 40 40 80 3F 00 3F 00 3F 00 3F 00"    /* ü`Ä@Ä@@Ä?.?.?.?. */
    $"3F 00 3F 00 3F 00 3F 00 7F 80 FF C0 FF C0 FF C0"    /* ?.?.?.?..Ä.¿.¿.¿ */
    $"FF C0 FF C0 FF C0 7F 80 3F 00 3F 00 3F 00 3F 00"    /* .¿.¿.¿.Ä?.?.?.?. */
    $"00 08 00 08"                                        /* .... */
};

data 'CURS' (259, locked, preload) {
    $"3F 00 3F 00 3F 00 3F 00 40 80 80 40 80 40 80 60"    /* ?.?.?.?.@ÄÄ@Ä@Ä` */
    $"9C 60 82 40 80 40 40 80 3F 00 3F 00 3F 00 3F 00"    /* ú`Ç@Ä@@Ä?.?.?.?. */
    $"3F 00 3F 00 3F 00 3F 00 7F 80 FF C0 FF C0 FF C0"    /* ?.?.?.?..Ä.¿.¿.¿ */
    $"FF C0 FF C0 FF C0 7F 80 3F 00 3F 00 3F 00 3F 00"    /* .¿.¿.¿.Ä?.?.?.?. */
    $"00 08 00 08"                                        /* .... */
};

data 'CURS' (260, locked, preload) {
    $"3F 00 3F 00 3F 00 3F 00 40 80 80 40 80 40 80 60"    /* ?.?.?.?.@ÄÄ@Ä@Ä` */
    $"9C 60 84 40 84 40 40 80 3F 00 3F 00 3F 00 3F 00"    /* ú`Ñ@Ñ@@Ä?.?.?.?. */
    $"3F 00 3F 00 3F 00 3F 00 7F 80 FF C0 FF C0 FF C0"    /* ?.?.?.?..Ä.¿.¿.¿ */
    $"FF C0 FF C0 FF C0 7F 80 3F 00 3F 00 3F 00 3F 00"    /* .¿.¿.¿.Ä?.?.?.?. */
    $"00 08 00 08"                                        /* .... */
};

data 'CURS' (261, locked, preload) {
    $"3F 00 3F 00 3F 00 3F 00 40 80 80 40 80 40 80 60"    /* ?.?.?.?.@ÄÄ@Ä@Ä` */
    $"9C 60 88 40 90 40 40 80 3F 00 3F 00 3F 00 3F 00"    /* ú`à@ê@@Ä?.?.?.?. */
    $"3F 00 3F 00 3F 00 3F 00 7F 80 FF C0 FF C0 FF C0"    /* ?.?.?.?..Ä.¿.¿.¿ */
    $"FF C0 FF C0 FF C0 7F 80 3F 00 3F 00 3F 00 3F 00"    /* .¿.¿.¿.Ä?.?.?.?. */
    $"00 08 00 08"                                        /* .... */
};

data 'CURS' (262, locked, preload) {
    $"3F 00 3F 00 3F 00 3F 00 40 80 80 40 80 40 80 60"    /* ?.?.?.?.@ÄÄ@Ä@Ä` */
    $"BC 60 80 40 80 40 40 80 3F 00 3F 00 3F 00 3F 00"    /* º`Ä@Ä@@Ä?.?.?.?. */
    $"3F 00 3F 00 3F 00 3F 00 7F 80 FF C0 FF C0 FF C0"    /* ?.?.?.?..Ä.¿.¿.¿ */
    $"FF C0 FF C0 FF C0 7F 80 3F 00 3F 00 3F 00 3F 00"    /* .¿.¿.¿.Ä?.?.?.?. */
    $"00 08 00 08"                                        /* .... */
};

data 'CURS' (263, locked, preload) {
    $"3F 00 3F 00 3F 00 3F 00 40 80 80 40 90 40 88 60"    /* ?.?.?.?.@ÄÄ@ê@à` */
    $"9C 60 80 40 80 40 40 80 3F 00 3F 00 3F 00 3F 00"    /* ú`Ä@Ä@@Ä?.?.?.?. */
    $"3F 00 3F 00 3F 00 3F 00 7F 80 FF C0 FF C0 FF C0"    /* ?.?.?.?..Ä.¿.¿.¿ */
    $"FF C0 FF C0 FF C0 7F 80 3F 00 3F 00 3F 00 3F 00"    /* .¿.¿.¿.Ä?.?.?.?. */
    $"00 08 00 08"                                        /* .... */
};
```

```
/*****************************************************************************/
/*                      HEADER  FILE                                         */
/*                      FOR PROGRAM MODITALK                                 */
/*                      FILENAME:   ModiTalk.h                               */
/*                      DATE    :   JANUARY 1990                             */
/*                      AUTHOR  :   SOHRAB MODI                              */
/*****************************************************************************/
/*****************************************************************************/
/*                   FILE INCLUDES                                           */
/*****************************************************************************/
#include <stdio.h>
#include <Quickdraw.h>
#include <EventMgr.h>
#include <SerialDvr.h>
#include <unix.h>
/*****************************************************************************/
/*                   GENERAL DEFINE STATEMENTS                               */
/*****************************************************************************/
#define BELL                '\007'
#define LAST_PACKET         '\008'
#define NIL_STRING          "\p"
#define SENDER_HANDSHAKE    1
#define RECEIVER_HANDSHAKE  2
#define FATAL_ERROR         3
#define READY_TO_BEGIN      4
#define BAD_HANDSHAKE       5
#define EXIT_DISPLAY        "\pBYE NOW"
#define BASE_RES_ID         400
#define NIL_POINTER         0L
#define MOVE_TO_FRONT       -1L
#define REMOVE_ALL_EVENTS   0
#define LEAVE_WHERE_IT_IS   FALSE
#define NORMAL_UPDATES      TRUE
#define SLEEP               0L
#define WNE_TRAP_NUM        0x60
#define UNIMPL_TRAP_NUM     0x9F
#define SUSPEND_RESUME_BIT  0x0001
#define ACTIVATING          1
#define RESUMING            1
#define TERec   struct TE
#define NUM_OF_BUFFERS      15
#define WINDOW_LENGTH       14
#define TURN_OVER           44
#define TOTAL_BUFFERS       45
#define ACK                 0
#define NACK                1
#define THE_END             100
#define END_HANDSHAKE       90
/*****************************************************************************/
/*                   DIALOG, WINDOW, MOUSE & MENU DEFINE STATEMENTS          */
/*****************************************************************************/
#define NIL_MOUSE_REGION    0L
#define MIN_WINDOW_HEIGHT   50
#define MIN_WINDOW_WIDTH    50
#define SCROLL_BAR_PIXELS   16
#define ROWHEIGHT           15
```

```
#define LEFTMARGIN              10
#define STARTROW                0
#define HORIZONTAL_OFFSET       0
#define HORIZONTAL_PIXEL        30
#define VERTICAL_PIXEL          50
#define NOT_A_NORMAL_MENU       -1
#define APPLE_MENU_ID           BASE_RES_ID
#define FILE_MENU_ID            BASE_RES_ID+1
#define FONT_MENU_ID            100
#define STYLE_MENU_ID           101
#define PLAIN                   0
#define PLAIN_ITEM              0
#define BOLD_ITEM               2
#define ITALIC_ITEM             3
#define UNDERLINE_ITEM          4
#define OUTLINE_ITEM            5
#define SHADOW_ITEM             6
#define REMOVE_CHECK_MARK       FALSE
#define ADD_CHECK_MARK          TRUE
#define TEXT_FONT_SIZE          12
#define ABOUT_ITEM              1
#define ABOUT_ALERT             400
#define SEND_ITEM               1
#define RECEIVE_ITEM            2
#define QUIT_ITEM               3
#define BAUD_RATE_ID            102
#define DRAG_THRESHOLD          30
#define STATIC_TEXT             1
#define OPEN_IT                 2
#define FILE_NAME               3
#define CANCEL_IT               4
#define NULL_SEQ                0
#define SENDER_END              200
#define DISPLAY_ALERT_ID        BASE_RES_ID+1
#define NUM_OF_WATCH_CURS       8
#define WATCH_CURS_ID           256
#define THREE_HUNDRED           1
#define SIX_HUNDRED             2
#define TWELVE_HUNDRED          3
#define EIGHTEEN_HUNDRED        4
#define TWENTY_FOUR_HUNDRED     5
#define THIRTY_SIX_HUNDRED      6
#define FORTY_EIGHT_HUNDRED     7
#define SEVENTY_TWO_HUNDRED     8
#define NINETY_SIX_HUNDRED      9
#define NINETEEN_THOU_TWO_HUN   10
#define FIFTY_SEVEN_THOU_SIX_HUN 11
/*****************************************************************************/
/*                    STRING DEFINE STATEMENTS                               */
/*****************************************************************************/
#define CANNOT_OPEN_FILE_READ       BASE_RES_ID
#define CANNOT_OPENFILE_WRITE       BASE_RES_ID+1
#define TERM_SPECIFIED_BADLY_SEND   BASE_RES_ID+2
#define TERM_SPECIFIED_BADLY_RECV   BASE_RES_ID+3
#define EXIT_PROGRAM                BASE_RES_ID+4
#define BEGIN_TRANSFER              BASE_RES_ID+5
```

```
#define BAD_SHAKE                   BASE_RES_ID+6
#define TRANS_OVER_SENDER           BASE_RES_ID+7
#define TRANS_OVER_RECEIVER         BASE_RES_ID+8
#define TERM_REDEFINED_SEND         BASE_RES_ID+9
#define TERM_REDEFINED_RECV         BASE_RES_ID+10
```

```
/*************************************************************************/
/*                      DECLARATION     FILE                             */
/*                      FOR PROGRAM MODITALK                             */
/*                      FILENAME:   Decl.h                               */
/*                      DATE    :   JANUARY 1990                         */
/*                      AUTHOR  :   SOHRAB MODI                          */
/*************************************************************************/
EventRecord     myEvent;

MenuHandle      myAppleMenu,
                myFontMenu,
                myStyleMenu,
                myBaudMenu;

Rect            myDragRect,         /* specifies the window Drag area    */
                mySizeRect,         /* specifies the window size         */
                seqRect,            /* creates a box for printing in     */
                ackRect,            /* creates a box for printing in     */
                saveFileRect,       /* creates a box for printing in     */
                expectPacketRect,   /* creates a box for printing in     */
                recvPacketRect,     /* creates a box for printing in     */
                windowStartRect,    /* creates a box for printing in     */
                timeOutRect,        /* creates a box for printing in     */
                pacSizeRect;        /* creates a box for printing in     */

Style           myCurrentStyle = PLAIN; /*  defines the style for printing  */

int             DATA_CHAR   = 1022, /* num of data char, 0..1022 = 1023  */
                sendAckAt,
                lastBuffer,     /* if EOF then this is last seq num sent */
                nackRecvFor,    /* which buffer num the nack is for      */
                ackToReceive,   /* which buffer an ack is expected for   */
                endPrevWind,    /* the previous window ended at this seq */
                myBaudRate,     /* specifies the baud rate               */
                startWorkWindow, /* first buffer in working window       */
                readBuffNum,    /* receiver expecting to read this num   */
                numberOfTimeOuts, /* calculates the num of time outs     */
                oldAckToReceive, /* what the previous ack received was   */
                myCurRow,        /* specifies cursor row position        */
                myMaxRow,        /* specefies maximum row position       */
                myLastFont,     /* speceifies what the last font chosen was */
                lastPacketNumber,/* gives the last sequence number       */
                lastBufferSent,     /* indicates the last buffer sent    */
                lastBufferRead,     /* indicates which buffer was read last */
                readNext,           /* which buffer to read in next      */
                readFrom,           /* first buffer of window for sender */
                pacSeqNum,          /* receiver gets this sequence number */
                dataCharReceived,   /* receiver keeps updting this till EOP */
                totCharInPacket,    /* receiver gets this for each poacket */
                packetSize[45];

char            *malloc(),          /* malloc redefined as char CAUTION!!! */
                *bufferCount[45],   /* 45 Buffers for holding data       */
                *theFilename,       /* holds the filename to send        */
                *myFilename,        /* holds the filename to receive     */
                *bufferPtr;         /* used as a temp pointer to data    */
```

```
/*************************************************************************/
/* These declarations are for holding and printing infomation            */
/*************************************************************************/
char            prBuff[256],            /*              Sender           */
                pacSeqBuff[256],        /*              Sender           */
                timeOutBuff[256],       /*              Sender           */
                ackBuff[256],           /*              Sender           */
                totSizeBuff[256],       /*        Sender & Receiver      */
                expectPacBuff[256],     /*            Receiver           */
                saveFileBuff[256],      /*            Receiver           */
                receivedPacBuff[256],   /*            Receiver           */
                newWindBuff[256];       /*            Receiver           */


unsigned    char inbuf[8000],           /* input buffer size*/
                 WriteFileBuff[2047],   /* holds characters to be be written */
                 checkBuff[2047];       /* receiver buffer size */

FILE            *fp,*fwrit;

Boolean         firstTimeAround     = TRUE,
                readCompPacket      = TRUE,
                packetChecked       = TRUE,
                timerStarted        = FALSE,
                startTimer          = FALSE,
                turnedOver          = FALSE,
                myDone              = FALSE,
                skipPacket          = FALSE,    /* duplicate packet window  */
                nackReceived        = FALSE,    /* tells if a nack is recv  */
                SENDER_TERM         = FALSE,
                RECEIVER_TERM       = FALSE,
                ackIsReceived       = FALSE,
                finishedReading     = FALSE,
                finishedWriting     = FALSE,
                ThisIsLastPac       = FALSE,    /* This is the last packet  */
                lastPacketSent      = FALSE,    /* true when sender finishes */
                receivedClosing     = FALSE,
                dontSendReadySignal = FALSE,
                otherTermReady      = FALSE;

WindowPtr       myWindow,
                myHandShakeWind;


long    int     timeSentAt,
                oneBuffTimeOut,
                propogationDelay,
                timeOut;
```

```
/*************************************************************************/
/*                        STRUCTURE DEFINATIONS                          */
/*************************************************************************/

/**************   For defining the Modem port parameters    *****************/

struct          port
                {
                    int         refin,
                                refout;

                    unsigned    short   baud,
                                        parity,
                                        stopbits,
                                        databits;
                } PortA;
/**************   For defining the terminal state          ******************/

struct          TERMINAL
                {
                    Boolean     sender,
                                receiver,
                                readyToStart;
                }myTermIs;
```

```
/*****************************************************************************/
/*                      DECLARATION     FILE                                 */
/*                      FOR PROGRAM MODITALK                                 */
/*                      FILENAME:   Extern.h                                 */
/*                      DATE    :   JANUARY 1990                             */
/*                      AUTHOR  :   SOHRAB MODI                              */
/*****************************************************************************/

extern  EventRecord     myEvent;

extern  MenuHandle      myAppleMenu,
                        myFontMenu,
                        myStyleMenu,
                        myBaudMenu;

extern  Style           myCurrentStyle;

extern  WindowPtr       myWindow,
                        myHandShakeWind;

extern  Rect            myDragRect,
                        mySizeRect,
                        seqRect,
                        ackRect,
                        saveFileRect,
                        expectPacketRect,
                        recvPacketRect,
                        windowStartRect,
                        timeOutRect,
                        pacSizeRect;

extern  FILE            *fp,
                        *fwrit;


extern  int             DATA_CHAR,
                        lastBuffer,
                        packetSize[45],
                        sendToQueue,
                        distanceTravelled,
                        sendAckAt,
                        myLastFont,
                        nackRecvFor,
                        queueIsAt,
                        lastBufferRead,
                        ackToReceive,
                        oldAckToReceive,
                        lastBufferSent,
                        myBaudRate,
                        endPrevWind,
                        queueIsAt,
                        startWorkWindow,
                        readNext,
                        myCurRow,
                        myMaxRow,
                        numberOfTimeOuts,
```

```
                        checkTimeOut,
                        readBuffNum,
                        lastPacketNumber,
                        dataCharReceived,
                        pacSeqNum,
                        totCharInPacket,
                        readFrom;


extern  char            *malloc(),
                        *bufferCount[45],
                        *theFilename,
                        *myFilename,
                        *bufferPtr;

/*************************************************************************/
/* These declarations are for holding and printing infomation            */
/*************************************************************************/
extern  char            prBuff[256],            /* Sender */
                        pacSeqBuff[256],        /* Sender */
                        timeOutBuff[256],       /* Sender */
                        ackBuff[256],           /* Sender */
                        totSizeBuff[256],       /* Sender & Receiver */
                        expectPacBuff[256],     /* Receiver */
                        saveFileBuff[256],      /* Receiver */
                        receivedPacBuff[256],   /* Receiver */
                        newWindBuff[256];       /* Receiver */



extern  unsigned    char inbuf[8000],
                         WriteFileBuff[2047],
                         checkBuff[2047];

extern  Boolean         firstTimeAround,
                        turnedOver ,
                        otherTermReady,
                        dontSendReadySignal,
                        nackReceived,
                        startTimer,
                        timerStarted,
                        readCompPacket,
                        packetChecked,
                        myDone,
                        ThisIsLastPac,
                        lastPacketSent,
                        skipPacket,
                        ackIsReceived,
                        SENDER_TERM,
                        RECEIVER_TERM,
                        firstTimeAround,
                        finishedReading,
                        receivedClosing,
                        finishedWriting;
```

```
extern  long    int     timeSentAt,
                        propogationDelay,
                        oneBuffTimeOut,
                        timeOut;

/***************** Structures ******************/

extern  struct          TERMINAL
                        {
                            Boolean     sender,
                                        receiver,
                                        readyToStart;
                        }myTermIs;


extern  struct          port
                        {
                            int         refin,
                                        refout;

                            unsigned    short   baud,
                                                parity,
                                                stopbits,
                                                databits;
                         }PortA;
```

```
/*****************************************************************************/
/*                      PROGRAM     FILE                                     */
/*                      FOR PROGRAM MODITALK                                 */
/*                      FILENAME:   ModiTalk.c                               */
/*                      DATE    :   JANUARY 1990                             */
/*                      AUTHOR  :   SOHRAB MODI                              */
/*****************************************************************************/
#include    "TrialNetHeader.h"
#include        "Decl.h"
/*****************************************************************************/
/*                              MAIN                                         */
/*****************************************************************************/
main()
{

    ToolBoxInit();
    MenuBarInit();
    InitSerial();
    SetUpSizeRect();
    SetUpDragRect();
    InitBuffers();
    GetGreetingWindow();
    WindowInit();
    SetUpWindow();
    get_curs_handles();


    firstTimeAround = TRUE;
    while( myDone == FALSE )
    {
        if(finishedReading == TRUE && finishedWriting == TRUE)
            transOver();
        else
        {
            GetNextEvent( everyEvent, &myEvent);
            EventHandler(&myEvent); /* "Procedures.c" handles next event */
            getblock();     /* "procedures.c" Polls the port for data*/
        }
    }
}

/*****************************************************************************/
/*                              Tool Box Init                                */
/*****************************************************************************/

ToolBoxInit()
{
    InitGraf ( &thePort );
    InitFonts();
    FlushEvents( everyEvent,REMOVE_ALL_EVENTS );
    InitWindows();
    InitMenus();
    TEInit();       /* For Text edit routines */
    InitDialogs( NIL_POINTER );
    InitCursor();
}
```

```
/*****************************************************************************/
/*                                Window Init                                */
/*****************************************************************************/

WindowInit()
{
    myWindow = GetNewWindow( BASE_RES_ID,NIL_POINTER,MOVE_TO_FRONT );
    myHandShakeWind = GetNewWindow( BASE_RES_ID + 2,NIL_POINTER,MOVE_TO_FRONT);
    ShowWindow( myWindow);
    ShowWindow( myHandShakeWind);
    SetPort( myHandShakeWind);
    MoveTo( HORIZONTAL_PIXEL,VERTICAL_PIXEL );
}

/*****************************************************************************/
/*                              Serial port init                             */
/*****************************************************************************/

InitSerial()
{

    unsigned    char    temp[100];
    int         e;          /* Error returned by routines */
    long        whatread;

    oneBuffTimeOut = 1.25 * 60L; /* corr to sending 1500 bytes/packet */
    myBaudRate = NINETY_SIX_HUNDRED;
    CheckItem( myBaudMenu,myBaudRate,ADD_CHECK_MARK);
    PortA.baud = baud9600; /* baud rate */
    PortA.parity = noParity; /* parity */
    PortA.databits = data8;  /* 8 bits of data */
    PortA.stopbits = stop10; /* 1 stop bit */

    e = OpenDriver("\P.AOut", &PortA.refout); /* open the ROM drivers */
    e = OpenDriver("\P.AIn", &PortA.refin);

            /* do the settings for the port.baud rate defaults to 9600 */
    SerReset(PortA.refin,
        PortA.baud | PortA.parity | PortA.stopbits | PortA.databits);
    SerReset(PortA.refout,
        PortA.baud | PortA.parity | PortA.stopbits | PortA.databits);

    SerSetBuf(PortA.refin, inbuf,6000); /* large input buffer size */

    SerGetBuf(PortA.refin, &whatread);      /* Clear any Junk in the buffer*/
    if(whatread > 0)
        e = FSRead(PortA.refin, &whatread,temp);
}
```

```
/*****************************************************************************/
/*                  This is to (m)allocate memory & init stuff               */
/*****************************************************************************/
InitBuffers()
{
    int Temp,i;
    long    errorFactor,distanceTravelled,readTime;

    TextFont( times );
    TextSize(12);
    myTermIs.sender     = FALSE,
    myTermIs.receiver   = FALSE,
    myTermIs.readyToStart = FALSE;
    readCompPacket      = TRUE;

    readTime            = 200L;
    errorFactor         = 50L;
    distanceTravelled   = 1000L; /* distance travelled is 1000 meters */
    propogationDelay    = (long)((distanceTravelled/300000 ) + errorFactor);
    timeOut             = (long)((oneBuffTimeOut + propogationDelay) * 15L) +
                                        propogationDelay + readTime;

    for(Temp = 0;Temp <= TOTAL_BUFFERS;Temp++)
        bufferCount[Temp] = malloc(2047);
    bufferPtr = malloc(2047);
            for(Temp = 0;Temp <= TOTAL_BUFFERS;Temp++)
        {
            bufferPtr = bufferCount[Temp];
            for(i = 0;i<= 2047;i++)
                *(bufferPtr + i) = 'T';
        }
}
/*****************************************************************************/
/*                           Initiliaze Menu Bar                             */
/*****************************************************************************/
MenuBarInit()
{
    Handle      myMenuBar;

    myMenuBar = GetNewMBar( BASE_RES_ID );    /* Loads MBAR resource  */
    SetMenuBar( myMenuBar );          /* Use This data as current MBAR res */
    myAppleMenu = GetMHandle( APPLE_MENU_ID );
    myFontMenu = GetMenu( FONT_MENU_ID );
    myStyleMenu = GetMenu( STYLE_MENU_ID );
    myBaudMenu = GetMenu( BAUD_RATE_ID );
    InsertMenu( myFontMenu, NOT_A_NORMAL_MENU );
    AddResMenu( myFontMenu, 'FONT' );
    InsertMenu( myStyleMenu, NOT_A_NORMAL_MENU );
    CheckItem( myStyleMenu, PLAIN_ITEM, TRUE );
    InsertMenu( myBaudMenu, NOT_A_NORMAL_MENU );
    CheckItem( myBaudMenu, PLAIN_ITEM, TRUE );
    AddResMenu( myAppleMenu, 'DRVR' );
    DrawMenuBar();
    myLastFont = 1;
    HandleFontChoice( myLastFont );
}
```

```
/*****************************************************************************/
/*                        Get The Winmdow Rectangle Size                     */
/*****************************************************************************/

SetUpSizeRect()
{

    mySizeRect.top = MIN_WINDOW_HEIGHT;
    mySizeRect.left = MIN_WINDOW_WIDTH;

    mySizeRect.bottom = screenBits.bounds.bottom - screenBits.bounds.top;
    mySizeRect.right = screenBits.bounds.right - screenBits.bounds.left;
}

/*****************************************************************************/
/*                            The End . Exit Normally                        */
/*****************************************************************************/

transOver()
{
    int     e;

    if(SENDER_TERM)
        PrintTheStr(TRANS_OVER_SENDER);
    else if(RECEIVER_TERM)
        PrintTheStr(TRANS_OVER_RECEIVER);
    myDone = TRUE;

    KillIO(PortA.refout);
    KillIO(PortA.refin);
}

/*****************************************************************************/
/*              Display Greeting message and Picture on start up             */
/*****************************************************************************/

GetGreetingWindow()
{
    WindowPtr       greetingWindow;
    PicHandle       thePicture;
    Rect            myRect;

    greetingWindow = GetNewWindow( BASE_RES_ID +1, NIL_POINTER, MOVE_TO_FRONT );
    SetPort(greetingWindow);
    thePicture = GetPicture( BASE_RES_ID );
    myRect = greetingWindow->portRect;
    CenterPict( thePicture, &myRect );
    DrawPicture( thePicture, &myRect );
    while(!Button());
    CloseWindow(greetingWindow);
    EmptyHandle(thePicture);
}
```

```
/*****************************************************************************/
/*  Center the above greeting Picture so it does not appear lopsided         */
/*****************************************************************************/

CenterPict( thePicture, myRectPtr )
PicHandle   thePicture;
Rect        *myRectPtr;
{
    Rect    windRect, pictureRect;

    windRect = *myRectPtr;
    pictureRect = (**( thePicture )).picFrame;
    myRectPtr->top = (windRect.bottom - windRect.top - (pictureRect.bottom - pictureRe
        / 2 + windRect.top;
    myRectPtr->bottom = myRectPtr->top + (pictureRect.bottom - pictureRect.top);
    myRectPtr->left = (windRect.right - windRect.left - (pictureRect.right - pictureRe
        / 2 + windRect.left;
    myRectPtr->right = myRectPtr->left + (pictureRect.right - pictureRect.left);
}

/*****************************************************************************/
/*                  Set Up the Row and Cursor positions                      */
/*****************************************************************************/

SetUpWindow()
{
    Rect    myTempRect;

    myTempRect = myWindow->portRect;
    myMaxRow = (myTempRect.bottom) - myTempRect.top - ROWHEIGHT;
    myCurRow = STARTROW;
}
```

```
/*************************************************************************/
/*                      PROGRAM     FILE                                 */
/*                      FOR PROGRAM MODITALK                             */
/*                      FILENAME:   Procedures.c                         */
/*                      DATE    :   JANUARY 1990                         */
/*                      AUTHOR  :   SOHRAB MODI                          */
/*************************************************************************/
#include            "TrialNetHeader.h"
#include            "Extern.h"
/*************************************************************************/
/*  This procedure has the next event  and goes through the case statements */
/*  to see what type of event occured and how to handle it. Withen comment  */
/*  statements anything enclosed withen "" indicates the file name. As most */
/*  of the procedure names are self discriptive an explanation is given only*/
/*  where the procedure routine exists.                                    */
/*************************************************************************/
EventHandler(event)
EventRecord     *event;
{
    switch(event->what)
    {
        case nullEvent:
            break;
        case mouseDown:
            HandleMouseDown();  /* "display.c"                          */
            break;
        case mouseUp:
            break;
        case keyDown:
            putout((int)event->message & 0x7f);
            break;
        case keyUp:
            break;
        case autoKey:
            putout((int)event->message & 0x7f);
            break;
        case updateEvt:
            BeginUpdate(event->message);
/*          HandleUpdateEvent(); */
            EndUpdate(event->message);
            break;
        case diskEvt:
            break;
        case activateEvt:
            break;
        case networkEvt:
            break;
        case driverEvt:
            break;
        case applEvt:
            break;
        case app2Evt:
            break;
        case app3Evt:
            break;
```

```
        case app4Evt:
            break;
    }
}
/*****************************************************************************/
/*      This takes keybord entries and sends them through the serial Port    */
/*****************************************************************************/

putout(c)
char c;
{
    long cnt = 1L;
    int e;

    e = FSWrite(PortA.refout, &cnt, &c); /* write out 1 character. */
}

/*****************************************************************************/
/*  This is a routine called every time the Progrom control moves around the*/
/*  the main loop. If the handshake is complete then depending on what the  */
/*  terminal was set to be (either Receiver or Sender) it will Poll the Port*/
/*  and handle the respective routines.                                      */
/*****************************************************************************/

getblock()
{

    long        whatread;           /* how many chars are actually waiting */

    SerGetBuf( PortA.refin, &whatread );    /* anything there? */
    if( myTermIs.readyToStart == TRUE )
    {
        if(SENDER_TERM)
        {
            if( nackReceived)
            {
                DrawMyString("\p I AM IN PROCEDURES NACK RECEIVED == TRUE");
                SendNackBuffer();   /*              "sender.c"              */
            }
            else if(( !finishedReading ) && ( ackIsReceived == TRUE ))
                HandleReceiveMyAck();   /*          "sender.c"              */

            else
                DataAtSenderPort(); /*              "sender.c"              */
        }
        else if( RECEIVER_TERM )
        {
            if(( whatread > 0 ) && (myTermIs.readyToStart == TRUE ))
                    HandleReceiveData(whatread);    /*      "receiver.c"    */
        }
        firstTimeAround = FALSE;
    }

    else if(whatread > 0)
        HandleHandshake();
}
```

```
/*************************************************************************/
/*  This procedure is activated when the terminals are yet unconfigured  */
/*  These Routines check for various handshake errors & prevent some from*/
/*  taking place. As an example if a terminal is configured as being a   */
/*  sender, it sends a handshake to the other terminal indicating that he is*/
/*  a sender. on receiving this the other terminal disables the send oprtion*/
/*  in the menu bar thereby stopping the user from configering the terminal */
/*  as a sender. If in case both computers were to send the same signal at  */
/*  the same time the dilog boxes would pop up indicating errors.          */
/*************************************************************************/
HandleHandshake()
{
    unsigned    char        c[2],num; /* big buffer */
    int                     e;  /* error codes */
    int         i,count;
    long        whatread;   /* how many chars are actually waiting */
    unsigned    int         temp;


    SerGetBuf(PortA.refin, &whatread);  /* anything there? */
    if(whatread >1)
        whatread = 1L;
    e = FSRead(PortA.refin, &whatread, c) ; /* Grab the chars */
    temp = (c[0] & 0x00ff);

    switch(temp)
    {
        case SENDER_HANDSHAKE :     /*This term must be a receiver */
                    /*  Is this term a sender ??? */
                if(SENDER_TERM == TRUE)
                    PrintTheStr(TERM_SPECIFIED_BADLY_SEND);
                else
                {
                    HandleHandshakeOfSender();
                    DisableItem(GetMHandle(FILE_MENU_ID), SEND_ITEM);
                }
                break;

        case RECEIVER_HANDSHAKE :   /*This term must be a sender */
                if(RECEIVER_TERM == TRUE) /* this term is receiver */
                    PrintTheStr(TERM_SPECIFIED_BADLY_RECV);
                else
                {
                    HandleHandshakeOfReceiver();
                    DisableItem(GetMHandle(FILE_MENU_ID), RECEIVE_ITEM);
                }
                break;

        case FATAL_ERROR :  /* FATAL ERROR exit the program */
                PrintTheStr(EXIT_PROGRAM);
                myDone = TRUE;
                break;
```

```
        case READY_TO_BEGIN:    /* ALL SET lets GO */
                    DisableItem(GetMHandle(BAUD_RATE_ID), 0L);
                    HandleBegin();
                    break;

        case BAD_HANDSHAKE:
                    PrintTheStr(BAD_SHAKE);
                    break;

        default:
                   num = (5 & 0xff);/* send signal back
                                       indicating bad Handshake */
                   putout(num);
                   break;
        }
}

/*************************************************************************/
/*  printf is not supported for printing text on the screen. Hence to print */
/*  to screen the text to be printed is copied into a buffer using sprintf  */
/*  example:- sprintf(prBuff,"\P I HAVE %d  Dollers",money). Then prBuff is */
/*  sent to this routine which checks to see that there is enough space to  */
/*  print it on screen or else it scrolls the window.                       */
/*************************************************************************/

DrawMyString( s )
Str255      s;
{
        int         i;


        if(myCurRow > myMaxRow)
            ScrollWindow();
        else
            myCurRow += ROWHEIGHT;
        MoveTo( LEFTMARGIN,myCurRow );
        DrawString( s );
}



/*************************************************************************/
/*  When a Negetive acknowledgment takes place this routine is called by    */
/*  the Receiver to dispose off any data in its buffers so it can receive   */
/*  the new retransmissions without having to go through the buffer looking */
/*  to find the correct header.                                             */
/*************************************************************************/

ClearTheInputBuffer()
{

    long        whatread;
    int         e;
    unsigned    char    MytempBuff[2050];
```

```
    do
    {
        SerGetBuf(PortA.refin, &whatread);
        if(whatread > 0)
        {
            if(whatread > 2047)
                whatread = 2047L;
            e = FSRead(PortA.refin, &whatread,MytempBuff); /* Grab the chars */
        }
    }while(whatread > 0);
}

/*****************************************************************************/
/*  When the position of the pen which draws the strings on screen hits the */
/*  bottom edge of the window this routine is called upon to scroll the     */
/*  window upwards.                                                          */
/*****************************************************************************/

ScrollWindow()
{
    RgnHandle       tempRgn;

    tempRgn = NewRgn();
    RectRgn(tempRgn, &myWindow->portRect);
    ScrollRect(&myWindow ->portRect,HORIZONTAL_OFFSET,-ROWHEIGHT,tempRgn);
    DisposeRgn( tempRgn );
}

/*****************************************************************************/
/*  This Routine will display various error messages which have been saved  */
/*  in the resource file. The type of string displayed will depend on the ID*/
/*  number passed into this routine.                                         */
/*****************************************************************************/

PrintTheStr( number )
int number;
{
    StringHandle    DisplayStrHandle;

    if ( ( DisplayStrHandle = GetString( number ) ) == NIL_POINTER )
        ParamText( EXIT_DISPLAY, NIL_STRING, NIL_STRING, NIL_STRING );
    else
    {

        HLock( DisplayStrHandle );
        ParamText( *DisplayStrHandle, NIL_STRING, NIL_STRING, NIL_STRING );
        HUnlock( DisplayStrHandle );
    }
    StopAlert( DISPLAY_ALERT_ID, NIL_POINTER );
    DisposHandle(DisplayStrHandle);
}
/*****************************************************************************/
/*  If the terminal was chosen to act as a sender the routine comes here to */
/*  start reading data into the buffers and then to start sending them      */
/*  through the serial port.                                                 */
/*****************************************************************************/
```

```
HandleBegin()
{

    int tempToSend;


    StartingDeclarations();
    if(SENDER_TERM == TRUE)
    {
        ReadIntoBuffer(readNext,29);
        readNext = 30;
        if(!finishedReading)
        {
            SendFileModem(0,WINDOW_LENGTH);
            if(nackReceived == FALSE)
                SendFileModem(15,WINDOW_LENGTH);
        }
        else
        {
            ackToReceive = lastBuffer;
            if(ackToReceive > WINDOW_LENGTH)
            {
                tempToSend = lastBuffer - WINDOW_LENGTH - 1;
                SendFileModem(0,WINDOW_LENGTH);
                if(nackReceived == FALSE)
                    SendFileModem(15,tempToSend);
            }
            else
                SendFileModem(0,lastBuffer);

            if(nackReceived == FALSE)
            {
                SendHeader((int)NULL_SEQ,(int)END_HANDSHAKE);
                lastPacketSent = TRUE;
            }
        }
    }
}

/*************************************************************************/
/*  These are just some initial declarations which are called upon before  */
/*  going into the above routine and starting the program                  */
/*************************************************************************/

StartingDeclarations()
{

    int   num;

    if(dontSendReadySignal == FALSE)
    {
        num = (4 & 0xff);
        putout(num);
    }
```

```
    CloseWindow(myHandShakeWind);
    SetPort(myWindow);
/*  DoSetUp();  */
    ackToReceive = 14;
    oldAckToReceive = 14;
    endPrevWind = 14;
    sprintf(ackBuff,"\p %d ",ackToReceive);
    numberOfTimeOuts = 0;
    sprintf(timeOutBuff,"\p %d ",numberOfTimeOuts);
/*  DrawMyString("\pBEGIN_TRANSFER "); */
    myTermIs.readyToStart = TRUE;
    lastBuffer      = 0;
    lastBufferSent  = 0;
    readNext        = 0;
    startWorkWindow = 0;
    readBuffNum     = 0;
    sendAckAt = startWorkWindow + WINDOW_LENGTH;
    if(sendAckAt > TURN_OVER)
        sendAckAt -= (int)TOTAL_BUFFERS;

}

/*****************************************************************************/
/*  Depending on what the terminal was chosen to be this procedure will     */
/*  go to the respective routine given below to print the display screen.   */
/*****************************************************************************/

DoSetUp()
{

    if(SENDER_TERM)
        HandleSenderScreen();
    else
        HandleReceiverScreen();
}

/*****************************************************************************/
/*      This initilizes the way the display for the sender would appear     */
/*****************************************************************************/

HandleSenderScreen()
{


    MoveTo( HORIZONTAL_PIXEL,VERTICAL_PIXEL );
    TextSize(12);
    TextFace(bold);
    DrawString("\p SENDING PACKET SEQUENCE NUMBER");
    SetRect(&seqRect,300,35,350,55);
    MoveTo( HORIZONTAL_PIXEL,VERTICAL_PIXEL  + 20);
    DrawString("\pTotalSize of Packet");
    SetRect(&pacSizeRect,300,55,350,75);
    MoveTo( HORIZONTAL_PIXEL,VERTICAL_PIXEL  + 40);
    DrawString("\pAck To Receive is");
    SetRect(&ackRect,300,75,350,95);
    MoveTo( HORIZONTAL_PIXEL,VERTICAL_PIXEL  + 60);
```

```
    DrawString("\pNumber of Time Outs");
    SetRect(&timeOutRect,300,95,350,115);
    PenSize(3,3);
    FrameRect(&seqRect);
    FrameRect(&pacSizeRect);
    FrameRect(&ackRect);
    FrameRect(&timeOutRect);
    PenSize(1,1);
    TextSize(9);
    TextFace(bold);
}

/*****************************************************************************/
/*      This initilizes the way the display for the receiver would appear    */
/*****************************************************************************/



HandleReceiverScreen()
{

    MoveTo( HORIZONTAL_PIXEL,VERTICAL_PIXEL );
    TextSize(12);
    TextFace(bold);
    DrawString("\pExpecting Packet Sequence Number");
    SetRect(&expectPacketRect,300,35,350,55);
    MoveTo( HORIZONTAL_PIXEL,VERTICAL_PIXEL  + 20);
    DrawString("\pReceived Packet Sequence Number");
    SetRect(&recvPacketRect,300,55,350,75);
    MoveTo( HORIZONTAL_PIXEL,VERTICAL_PIXEL  + 40);
    DrawString("\pTotal Size of Received Packet");
    SetRect(&pacSizeRect,300,75,350,95);
    MoveTo( HORIZONTAL_PIXEL,VERTICAL_PIXEL  + 60);
    DrawString("\pWindow Starts At");
    SetRect(&windowStartRect,300,95,350,115);
    MoveTo( HORIZONTAL_PIXEL,VERTICAL_PIXEL  + 80);
    DrawString("\pSave To File");
    SetRect(&saveFileRect,300,115,350,135);
    PenSize(3,3);
    FrameRect(&expectPacketRect);
    FrameRect(&recvPacketRect);
    FrameRect(&pacSizeRect);
    FrameRect(&windowStartRect);
    FrameRect(&saveFileRect);
    PenSize(1,1);
    TextSize(9);
    TextFace(bold);
}


/*****************************************************************************/
/*  This simply prints back on screen what was on it before the window was   */
/*  resized and erased                                                       */
/*****************************************************************************/

HandleUpdateEvent()
```

```
{

    if(myTermIs.readyToStart)
    {
        if(SENDER_TERM)
        {
            HandleSenderScreen();
            PrintSenderScreen();                    /* "sender.c"  */
        }
        else
        {
            HandleReceiverScreen();
            PrintReceiverScreen();                  /*   "Receiver.c"  */
        }
    }
}
```

```
/*****************************************************************************/
/*                      PROGRAM     FILE                                     */
/*                      FOR PROGRAM MODITALK                                 */
/*                      FILENAME:   display.c                                */
/*                      DATE    :   JANUARY 1990                             */
/*                      AUTHOR  :   SOHRAB MODI                              */
/*****************************************************************************/
#include            "TrialNetHeader.h"
#include            "Extern.h"
#include            "Pascal.h"
CursHandle          watchCursors[NUM_OF_WATCH_CURS];

/*****************************************************************************/
/*  When File-IO takes place at the Sender The cursor changes from the       */
/*  arrow to a watch. As time passes the watch hands move. This is possible  */
/*  as there are 8 watches put in the resourse and it goes through and       */
/*  displays each one giving it the appearance of motion.                    */
/*****************************************************************************/

get_curs_handles()
{

    int i;

    for(i=0; i<NUM_OF_WATCH_CURS; i++)
    {
        watchCursors[i] = GetCursor(WATCH_CURS_ID+i);    /* from the resource*/
    }
}


/*****************************************************************************/
/*              This simply displays a single watch                          */
/*****************************************************************************/


showWatch()
{
    SetCursor(&(*(*watchCursors[0])));
}


/*****************************************************************************/
/*          This simply displays all the 8 watch positions                   */
/*****************************************************************************/


showWatchs(count)
long count;
{

    int index;

    index = (int)(count%NUM_OF_WATCH_CURS);
    SetCursor(&(*(*watchCursors[index])));
}
```

```
/*************************************************************************/
/*          This brings the cursor bck to normal                         */
/*************************************************************************/


showArrow()
{
    SetCursor(&arrow);
}


/*************************************************************************/
/*  When a mouse down event takes place the Event handler routine in     */
/*  "procedures.c" passes the contrl to this routine. This routine finds out*/
/*  in which window the mouse down event took place( we have one window) */
/*  and if withen the current window it tries to locate where the event took*/
/*  place and goes to their respective routines to handle it.            */
/*************************************************************************/


HandleMouseDown()
{

    WindowPtr   whichWindow;
    short int   thePart;
    long int    menuChoice,
                windSize;

    thePart = FindWindow( myEvent.where, &whichWindow ); /*     "tool box"  */
    switch ( thePart )      /* the part returns a number discribing where   */
    {
        case inMenuBar:
            menuChoice = MenuSelect( myEvent.where ); /* "      tool Box"   */
            HandleMenuChoice( menuChoice );
            break;
        case inDrag :
            DragWindow( whichWindow, myEvent.where, &myDragRect );
            break;
        case inGrow:
            growWindow( whichWindow,myEvent.where );
        break;
        case inZoomIn:
        case inZoomOut:
            zoomWindow(whichWindow,myEvent.where,thePart);
        case inGoAway :
            myDone = TRUE;
            break;
    }
}
```

```
/*****************************************************************************/
/*  MenuSelect is a tollBox routine call which gets a point. The point corr.*/
/*  to the position of the Cursor so that the High word will locate which   */
/*  menu was chosen and the low Word will indicate which item in that menu  */
/*  was chosen.                                                              */
/*****************************************************************************/


HandleMenuChoice( menuChoice )
long int    menuChoice;
{
    int theMenu;
    int theItem;

    if ( menuChoice != 0 )
    {
        theMenu = HiWord( menuChoice );
        theItem = LoWord( menuChoice );
        switch ( theMenu )
        {
            case APPLE_MENU_ID :
                HandleAppleChoice( theItem );
                break;
            case FILE_MENU_ID  :
                HandleFileChoice( theItem );
                break;
            case FONT_MENU_ID  :
                HandleFontChoice( theItem );
                break;
            case STYLE_MENU_ID :
                HandleStyleChoice( theItem );
                break;
            case BAUD_RATE_ID  :
                HandleBaudRate( theItem);
                break;
        }
        HiliteMenu( 0 ); /* when a call to MenuSelect is made the Menu bar
                            Got inverted. this brings it back to normal     */
    }
}


/*****************************************************************************/
/*  This is the Menu which contains the little apple logo. This menu bar    */
/*  mainly contains desk accessories. The first Item  however displays      */
/*  credit to the author by opening an Alert box. The rest of the items can */
/*  be activated by a call to OpenDeskAcc.                                   */
/*****************************************************************************/

HandleAppleChoice( theItem )
int theItem;
{
    Str255      accName;
    int         accNumber;
    short int   itemNumber;
    DialogPtr   AboutDialog;
```

```
    switch ( theItem )
    {
        case ABOUT_ITEM :
            NoteAlert( ABOUT_ALERT, NIL_POINTER );  /*      "tool box"      */
            break;
        default :
            GetItem( myAppleMenu, theItem, accName );
            accNumber = OpenDeskAcc( accName );
            break;
    }
}
/*****************************************************************************/
/*  This procedure lets you pick a font from the font menu. Most of these   */
/*  are tool box calls. Check item will either remove a tick mark or place  */
/*  a tick mark next to the respective font chosen. GetItem simply returns  */
/*  a font Name & GetFNum uses the font name to return an ID number. Finally*/
/*  the font is changed using the Font ID number.                            */
/*****************************************************************************/

HandleFontChoice( theItem )
int theItem;
{
    int     fontNumber;
    Str255  fontName;

    CheckItem( myFontMenu, myLastFont, REMOVE_CHECK_MARK );
    CheckItem( myFontMenu, theItem, ADD_CHECK_MARK );
    myLastFont = theItem;
    GetItem( myFontMenu , theItem , fontName );
    GetFNum( fontName , &fontNumber );
    TextFont( fontNumber );
}
/*****************************************************************************/
/*  This Menu Item has 3 items in it 1) SEND which configers your terminal  */
/*  act as a sender. 2) RECEIVE which configers the terminal to act as a    */
/*  receiver. 3) Quit which lets you quit from the program.                  */
/*****************************************************************************/

HandleFileChoice( theItem )
int theItem;
{

    int         e;

    switch ( theItem )
    {
        case SEND_ITEM :
                if(RECEIVER_TERM == TRUE)       /* if terminal is already a */
                    PrintTheStr(TERM_REDEFINED_RECV);  /*receiver raise err */
                else if(myTermIs.receiver == TRUE)
                    PrintTheStr(TERM_REDEFINED_RECV);
                else
                    SenderHandshake();  /* else configre it tobe a sender   */
                break;
```

```
        case RECEIVE_ITEM :
                if(SENDER_TERM == TRUE)
                    PrintTheStr(TERM_REDEFINED_SEND);
                else if(myTermIs.sender == TRUE)
                    PrintTheStr(TERM_REDEFINED_SEND);
                else
                    ReceiverHandshake();
                break;

        case QUIT_ITEM :
                myDone = TRUE;              /* Exit Normally */
                e = CloseDriver( PortA.refout); /* open the ROM drivers */
                e = CloseDriver( PortA.refin);
                KillIO(PortA.refout);
                KillIO(PortA.refin);
                break;

    }
}

/*****************************************************************************/
/*  This procedure brings up a dialog box which lets you choose the file     */
/*  you wish to send. It lets you toggle through the directory to select     */
/*  a file. It then sets this terminal to be a sender and sends a sender     */
/*  handshake to the other terminal. If the terminal has already been chosen*/
/*  as a sender then choosing it again will close a previously opened file   */
/*  and let you open another file to send.                                   */
/*****************************************************************************/

SenderHandshake()
{
    SFReply         openFile,saveFile;
    char            c;



    GetFileName(&openFile); /*              "tool box"                       */
    if(openFile.good)
    {
        SetVol(openFile.fName,openFile.vRefNum);    /*      "tool box"      */
        c = (1 & 0xff);
        putout(c);
        DrawMyString("\p** I AM SENDING A HANDSHAKE INDICATING I AM SENDER **");
        theFilename = PtoCstr((char *)openFile.fName); /*   "compiler"      */
        if(!SENDER_TERM)
        {
            SENDER_TERM = TRUE;
            OpenFileRead(theFilename);
        }
        else
        {
            fclose(fp);
            OpenFileRead(theFilename);
        }
    }
}
```

```
/*************************************************************************/
/*  This procedure brings up a dialog box which lets you type in the name   */
/*  of the file you wish to receive. It lets you toggle through the         */
/*  directory to select in which directory you wish to put the file. It also*/
/*  defaults to a name LASER in order to save time. If a name given already */
/*  exists in the current directory it will respond with a dialog box.      */
/*  It then sets this terminal to be a sender and sends a sender handshake  */
/*  to the other terminal. If the terminal has already been chosen as a     */
/*  sender then choosing it again will close a previously opened file and   */
/*  lets you open another file to send.                                     */
/*************************************************************************/

ReceiverHandshake()
{

    SFReply         openFile,saveFile;
    char            c;

    SaveFileName(&saveFile);
    if(saveFile.good)
    {
        SetVol(saveFile.fName,saveFile.vRefNum);
        DrawMyString("\pI AM SENDING A HANDSHAKE INDICATING I AM RECEIVER");
        c = (2 & 0xff);
        putout(c);
        myFilename = PtoCstr((char *)saveFile.fName);
        if(!RECEIVER_TERM)
        {
            OpenFileWrite(myFilename);
            RECEIVER_TERM = TRUE;
        }
        else
        {
            fclose(fwrit);
            OpenFileWrite(myFilename);
        }

    }
}
/*************************************************************************/
/* This gets the file name as explained in SenderHandshake()               */
/*************************************************************************/

GetFileName( openFile )
SFReply     *openFile;
{
    Point       myPoint;
    SFTypeList  typeList;
    int         numTypes;

    myPoint.h = 100;
    myPoint.v = 100;
    typeList[ 0 ] = 'Text';
    numTypes = -1;
    SFGetFile( myPoint, "\p", 0L, numTypes,&typeList, 0L, openFile );
}
```

```
/*****************************************************************************/
/* This requests a file name as explained in ReceiverHandshake()             */
/*****************************************************************************/
SaveFileName(saveFile)
SFReply     *saveFile;
{
    Point       myPoint;
    myPoint.h = 100;
    myPoint.v = 100;
    SFPutFile( myPoint, "\p","\p LASER",0L, saveFile );     /* "tool box"   */
}
/*****************************************************************************/
/*              This procedure lets you choose the your style of text.       */
/*****************************************************************************/
HandleStyleChoice( theItem )
int theItem;
{
    switch( theItem )
    {
        case PLAIN_ITEM:
            myCurrentStyle = PLAIN;
            break;
        case BOLD_ITEM:
            if ( myCurrentStyle & bold )
                myCurrentStyle -= bold;
            else
                myCurrentStyle |= bold;
            break;
        case ITALIC_ITEM:
            if ( myCurrentStyle & italic )
                myCurrentStyle -= italic;
            else
                myCurrentStyle |= italic;
            break;
        case UNDERLINE_ITEM:
            if ( myCurrentStyle & underline )
                myCurrentStyle -= underline;
            else
                myCurrentStyle |= underline;
            break;
        case OUTLINE_ITEM:
            if ( myCurrentStyle & outline )
                myCurrentStyle -= outline;
            else
                myCurrentStyle |= outline;
            break;
        case SHADOW_ITEM:
            if ( myCurrentStyle & shadow )
                myCurrentStyle -= shadow;
            else
                myCurrentStyle |= shadow;
            break;
    }
    CheckStyles();
    TextFace( myCurrentStyle );
}
```

```
/*****************************************************************************/
/*  This procedure lets you put check marks against all the  styles chosen  */
/*****************************************************************************/

CheckStyles()
{
    CheckItem( myStyleMenu, PLAIN_ITEM,     myCurrentStyle == PLAIN );
    CheckItem( myStyleMenu, BOLD_ITEM,      myCurrentStyle & bold );
    CheckItem( myStyleMenu, ITALIC_ITEM,    myCurrentStyle & italic );
    CheckItem( myStyleMenu, UNDERLINE_ITEM, myCurrentStyle & underline );
    CheckItem( myStyleMenu, OUTLINE_ITEM,   myCurrentStyle & outline );
    CheckItem( myStyleMenu, SHADOW_ITEM,    myCurrentStyle & shadow );
}

/*****************************************************************************/
/*  This Procedure assigns the drag area of the window to be withen the     */
/*  screen bounds.                                                           */
/*****************************************************************************/

SetUpDragRect()
{
    myDragRect          = screenBits.bounds;
    myDragRect.left     += DRAG_THRESHOLD;
    myDragRect.right    -= DRAG_THRESHOLD;
    myDragRect.bottom   -= DRAG_THRESHOLD;
}
/*****************************************************************************/
/*  This procedure lets you choose diffrent baud rates. depending on the    */
/*  baud rate chosen a time out calculation for one buffer to transmit is   */
/*  undertaken. finally a check mark is placed against the chosen baud rate */
/*****************************************************************************/

HandleBaudRate( theItem )
int theItem;
{
    int     e;

    CheckItem( myBaudMenu,myBaudRate,REMOVE_CHECK_MARK);
    switch( theItem )
    {
        case THREE_HUNDRED:
                    myBaudRate = THREE_HUNDRED;
                    PortA.baud = baud300;
                    oneBuffTimeOut = 40L * 60L; /* corr to sending 1500 bytes/packet *
                    break;
        case SIX_HUNDRED:
                    myBaudRate = SIX_HUNDRED;
                    PortA.baud = baud600; /* baud rate */
                    oneBuffTimeOut = 20L * 60L; /* corr to sending 1500 bytes/packet *
                    break;
        case TWELVE_HUNDRED:
                    myBaudRate = TWELVE_HUNDRED;
                    PortA.baud = baud1200; /* baud rate */
                    oneBuffTimeOut = 10L * 60L; /* corr to sending 1500 bytes/packet *
                    break;
```

```
        case EIGHTEEN_HUNDRED:
                    myBaudRate = EIGHTEEN_HUNDRED;
                    PortA.baud = baud1800; /* baud rate */
                    oneBuffTimeOut = 6.69 * 60L; /* corr to sending 1500 bytes/packet
                    break;
        case TWENTY_FOUR_HUNDRED:
                    myBaudRate = TWENTY_FOUR_HUNDRED;
                    PortA.baud = baud2400; /* baud rate */
                    oneBuffTimeOut = 5L * 60L; /* corr to sending 1500 bytes/packet *.
                    break;
        case THIRTY_SIX_HUNDRED:
                    myBaudRate = THIRTY_SIX_HUNDRED;
                    PortA.baud = baud3600; /* baud rate */
                    oneBuffTimeOut = 3.35 * 60L; /* corr to sending 1500 bytes/packet
                    break;
        case FORTY_EIGHT_HUNDRED:
                    myBaudRate = FORTY_EIGHT_HUNDRED;
                    PortA.baud = baud4800; /* baud rate */
                    oneBuffTimeOut = 2.5 * 60L; /* corr to sending 1500 bytes/packet '
                    break;
        case SEVENTY_TWO_HUNDRED:
                    myBaudRate = SEVENTY_TWO_HUNDRED;
                    PortA.baud = baud7200; /* baud rate */
                    oneBuffTimeOut = 1.69 * 60L; /* corr to sending 1500 bytes/packet
                    break;
        case NINETY_SIX_HUNDRED:
                    myBaudRate = NINETY_SIX_HUNDRED;
                    PortA.baud = baud9600; /* baud rate */
                    oneBuffTimeOut = 1.25 * 60L; /* corr to sending 1500 bytes/packet
                    break;
        case NINETEEN_THOU_TWO_HUN:
                    myBaudRate = NINETEEN_THOU_TWO_HUN;
                    PortA.baud = baud19200; /* baud rate */
                    oneBuffTimeOut = 0.634 * 60L; /* corr to sending 1500 bytes/packet
                    break;
        case FIFTY_SEVEN_THOU_SIX_HUN:
                    myBaudRate = FIFTY_SEVEN_THOU_SIX_HUN;
                    PortA.baud = baud57600; /* baud rate */
                    oneBuffTimeOut = 0.217 * 60L; /* corr to sending 1500 bytes/packet
                    break;

    }
        /* Note oneBuffTimeOut = X * 60, the 60 is as the clock ticks every
           60 seconds. Also all time outs (X) are adjusted so that the
           calculation comes out to the next 60 th second. */

        CheckItem( myBaudMenu,myBaudRate,ADD_CHECK_MARK);
        SerReset(PortA.refin,
            PortA.baud | PortA.parity | PortA.stopbits | PortA.databits);
        SerReset(PortA.refout,
            PortA.baud | PortA.parity | PortA.stopbits | PortA.databits);
}
```

```
/*****************************************************************************/
/*  This procedure lets you zoom your working window                         */
/*****************************************************************************/

zoomWindow(wPtr,myPoint,thePart)
WindowPtr   wPtr;
Point       myPoint;
short       thePart;
{
    GrafPtr     oldPort;

    if ( TrackBox( wPtr, myPoint, thePart ) )
        {
            GetPort( &oldPort );
            SetPort( wPtr );
            EraseRect( &wPtr->portRect );
            ZoomWindow( wPtr, thePart, LEAVE_WHERE_IT_IS );
            InvalRect( &wPtr->portRect );
            SetPort( oldPort );
        }
}

/*****************************************************************************/
/*  Thius procedure lets you grow your working window                        */
/*****************************************************************************/

growWindow(wPtr,myPoint)
WindowPtr   wPtr;
Point       myPoint;
{

    long        windSize;
    GrafPtr     oldPort;
    int         windWidth,windHeight;

    windSize    = GrowWindow( wPtr, myPoint, &mySizeRect );
    windHeight  = HiWord( windSize );
    windWidth   = LoWord( windSize );
    if ( windSize != 0 )
    {
        GetPort( &oldPort );
        SetPort( wPtr );
        EraseRect( &wPtr->portRect );
        SizeWindow( wPtr,windWidth ,windHeight, NORMAL_UPDATES );
        InvalRect( &wPtr->portRect );
        mySizeRect = wPtr->portRect;
        SetUpWindow();
        SetPort( oldPort );

    }
}
```

```
/*************************************************************************/
/*                      PROGRAM     FILE                                 */
/*                      FOR PROGRAM MODITALK                             */
/*                      FILENAME:   receiver.c                           */
/*                      DATE    :   JANUARY 1990                         */
/*                      AUTHOR  :   SOHRAB MODI                          */
/*************************************************************************/
#include            "TrialNetHeader.h"
#include            "Extern.h"



/*************************************************************************/
/*              This procedure opens the file for writing                */
/*************************************************************************/

OpenFileWrite(myFilename)
char        *myFilename;
{



    if((fwrit = fopen(myFilename,"w")) == NULL)
    {
      sprintf(prBuff,"\pERROR : CANNOT OPEN FILE FOR WRITING ");
      DrawMyString(prBuff);
     }
}

/*************************************************************************/
/*  On receiving a receiver handshake from the other terminal the program */
/*  checks to see if his terminal has already been configured as a sender */
/*  If he has then he sends a ready to begin message or else accepts the  */
/*  receiver message and waits for the user to configure him as Sender.   */
/*************************************************************************/

HandleHandshakeOfReceiver()


    char        num;

    if(SENDER_TERM == TRUE)
    {
        DrawMyString("\p *** I HAVE RECEIVED A RECEIVER HANDSHAKE ***");
        myTermIs.sender = TRUE;
        num = (4 & 0xff);
        putout(num);
        dontSendReadySignal = TRUE;
    }
    else
    {
        myTermIs.sender = TRUE;
        DrawMyString("\p I HAVE RECEIVED A RECEIVER_HANDSHAKE ");
    }
}
```

```
/*************************************************************************/
/*  This routine polls at the input port and gets out any data that is   */
/*  waiting there. If it is a new packet then it waits for a 5 byte header */
/*  to come in. Once it gets the header it checks to see that the header  */
/*  is the one the receiver is expecting by calling the routine checkHeader.*/
/*  If it is it goes in and retrives data from the port for the packet.   */
/*  If however it comes to this routine when the header is received but the */
/*  entire packet is not then it goes directly to the routine to pick up  */
/*  the rest of the data. If there is an error in the header then it goes in*/
/*  raises an error condition.                                            */
/*************************************************************************/

HandleReceiveData(whatread)
long int        whatread;
{
    unsigned    char        headerBuff[10];
    int         count;
    int         GetHeader();
    int         e;

     if(( readCompPacket == TRUE ))
     {

        while(whatread < 5)
            SerGetBuf(PortA.refin, &whatread);
        if (whatread >5)
            whatread = 5L;
        e = FSRead(PortA.refin, &whatread,headerBuff);      /* Grab the chars */
        totCharInPacket = GetHeader(headerBuff);

        if( totCharInPacket != -1)
        {
            if(receivedClosing == TRUE)
            {
                finishedReading = TRUE;
                finishedWriting = TRUE;
            }
            else
            {
                readCompPacket = FALSE;
                dataCharReceived = 0;
                ReadThePacket();
            }
        }
        else
        {
            readCompPacket = TRUE;
            DrawMyString("\P ERROR :I AM IN SEND NACK");
            SendNackSender(readBuffNum);
        }
    }
    else if(( readCompPacket == FALSE ))
        ReadThePacket();
}
```

```
/*************************************************************************/
/*  This Procedure Polls the input Buffer and brings out the data to be put */
/*  for that packet. If the end of the packet is reached and there is more  */
/*  data at the port then the extra characters will not be polled. Only the */
/*  number of characters needed will be pulled out. Once the complete packet*/
/*  is pulled out then it is sent to the procedure CheckPacket, so that all */
/*  extra byte stuffing can be removed and the number of data characters can*/
/*  be counted. If all is well then it goes to the routine SendToOutputFile */
/*  which will write it to file. It then checks to see if this was the last */
/*  packet in the window. If it was then it sends an ack for the window and */
/*  starts a new working window in procedure AcknowledgeOldWindow.It then   */
/*  gets ready to receive the next packet. If a window was retransmitted    */
/*  then it skips the packet.                                               */
/*************************************************************************/

ReadThePacket()
{
    int         e,i,j,charRecv,temp,goodPacket;
    long        whatread;
    unsigned    char        tempBuff[2000];


    SerGetBuf(PortA.refin, &whatread);
    charRecv = dataCharReceived + (int)whatread;
    if( charRecv >= totCharInPacket)
    {
        whatread = (long)(totCharInPacket - (dataCharReceived -1));
        readCompPacket = TRUE;
    }

    e = FSRead(PortA.refin, &whatread, tempBuff);          /* Grab the chars */
    temp = dataCharReceived;

    for(i = 0; i <= (int)whatread; i++)
        checkBuff[temp++] = tempBuff[i];
    dataCharReceived = charRecv;

    if(readCompPacket == TRUE)
    {
        dataCharReceived = 0;
        if( skipPacket == TRUE )
            SkipThisPacket();
        else
        {
            goodPacket = CheckPacket();
            if( goodPacket)
            {
                SendToOutputFile();
                if( pacSeqNum == sendAckAt)
                    AcknowledgeOldWindow();
                GetMyNextBufferNumber();
            }
```

```
            else if(goodPacket == 0)
            {
                dataCharReceived = 0;
                DrawMyString("\pPACKET IS BAD AND HENCE I WILL SEND A NACK ");
                SendNackSender(pacSeqNum);
            }
        }
    }
}


/*****************************************************************************/
/*  This procedure simply tells the receiver which packet number to accept  */
/*****************************************************************************/

GetMyNextBufferNumber()
{

    readBuffNum = readBuffNum + 1;
    if( readBuffNum > TURN_OVER)
        readBuffNum = 0;
    DrawMyString("\p ");
    sprintf(prBuff,"\pI AM EXPECTING BUFFER NUMBER = %d   ",readBuffNum);
    sprintf(expectPacBuff,"\p %d ",readBuffNum);
    /*****   PrintReceiverScreen(); ****/
    DrawMyString(prBuff);
}

/*****************************************************************************/
/*  This procedure will acknowledge the recept of a window and start a new  */
/*  window.                                                                  */
/*****************************************************************************/

AcknowledgeOldWindow()


    endPrevWind = pacSeqNum;
    SendAckSender(pacSeqNum);
    startWorkWindow = pacSeqNum +1;
    if(startWorkWindow > TURN_OVER)
        startWorkWindow -= (int)TOTAL_BUFFERS;
    sendAckAt = startWorkWindow + WINDOW_LENGTH;
    if(sendAckAt > TURN_OVER)
        sendAckAt -= (int)TOTAL_BUFFERS;
    sprintf(prBuff,"\pTHE NEW WORKING WINOW NOW STARTS AT %d  ",startWorkWindow);
    DrawMyString(prBuff);
    sprintf(newWindBuff,"\p %d ",startWorkWindow);
    /*****   PrintReceiverScreen();     ****/
}

/*****************************************************************************/
/*  This procedure calls routine writeToFile to send the packet to the file */
/*****************************************************************************/
```

```
SendToOutputFile()
{

                sprintf(saveFileBuff,"\p    YES");
                /**** PrintReceiverScreen();        ****/
                WriteToFile(WriteFileBuff);
}

/*************************************************************************/
/*  This procedure removes any byte stuffig and does a word count on the    */
/*   packet.                                                               */
/*************************************************************************/

CheckPacket()
{

    int         dataCnt,temp,i;
    Boolean     error = FALSE;
    unsigned    char    tempBELL;

    tempBELL = (unsigned char)BELL;
    temp    = 0;
    i       = 0;
    dataCnt = 0;

    while(( dataCnt < totCharInPacket) && ( error == FALSE ))
    {
        if( checkBuff[i] == tempBELL )
        {
            i++;
            dataCnt++;
            if( checkBuff[i] == tempBELL )
                WriteFileBuff[temp] = checkBuff[i];
            else
                error = TRUE;
        }

        else if(checkBuff[i] != tempBELL)
            WriteFileBuff[temp] = checkBuff[i];

        i++;
        temp++;
        dataCnt++;
    }

    if( checkBuff[i] != tempBELL)           /* is the last character a bell */
        error == TRUE;

    if(( !ThisIsLastPac ) && (( temp - 1) != DATA_CHAR) )
        error = TRUE;

    if( error == FALSE)
        return(1);
    else
        return(0);
}
```

```
/*************************************************************************/
/*  This procedure gets the header which contains a Control character which */
/*  is followed by a packet sequence number, followed by a 2 byte number    */
/*  which indicates the size of the packet,followed by another Control Char */
/*************************************************************************/


GetHeader(c)
unsigned    char    *c;
{

    unsigned    int     sizeA,sizeB,TotalSize,sizeOfPac,temp;
    unsigned    char    headerA,headerB,pacseq;




    if(*c == (unsigned char)LAST_PACKET)
        ThisIsLastPac = TRUE;

    if(( *c == (unsigned char)BELL) || ( *c == (unsigned char)LAST_PACKET ))
    {
        pacSeqNum   = *(c+1);
        headerA     = *(c+2) ;
        headerB     = *(c+3);
        sizeA = (headerA  & 0x00ff);
        sizeB = ((headerB << 8) & 0xff00);
        TotalSize = (sizeA | sizeB);
        if(pacSeqNum != 90)
            sprintf(prBuff,"\p I HAVE RECEIVED PACKET SEQUENCE NUMBER %d  ",pacSeqNum)
        else if(pacSeqNum == 90)
            sprintf(prBuff,"\p I HAVE RECEIVED END SIGNAL");
        DrawMyString(prBuff);
        sprintf(prBuff,"\pTotal Size of the packet is %d  ",TotalSize);
        DrawMyString(prBuff);
        sprintf(totSizeBuff,"\p %d  ",TotalSize);
        sprintf(receivedPacBuff,"\p %d  ",pacSeqNum);
        /****   PrintReceiverScreen();      ****/


        if(( *(c + 4) == (unsigned char)BELL ) || ( *(c + 4) == (unsigned char)LAST_PA
        {
            sizeOfPac = CheckHeader(TotalSize);
            return(sizeOfPac);
        }
        else
            return( -1);
    }
    else
        return(-1);
}
```

```
/*****************************************************************************/
/*  If a retransmission took place then it will not write to file but will  */
/*  acknowledge the receipt of the old window.                               */
/*****************************************************************************/

SkipThisPacket()
{

    skipPacket = FALSE;
    sprintf(saveFileBuff,"\p     NO");
    /****   PrintReceiverScreen(); *****/
    /* ACK the previous window sent */
    if( pacSeqNum == endPrevWind)
    {
        sprintf(prBuff,"\p I HAVE SENT AN ACK FOR BUFFER %d  ",pacSeqNum);
        DrawMyString(prBuff);
        SendAckSender(pacSeqNum);
    }
}

/*****************************************************************************/
/*  This Procedure will print the need data on the screen. This routine is  */
/*  also called if an update event takes place.                              */
/*****************************************************************************/


PrintReceiverScreen()
{
    TextSize(9);
    TextFace(bold);
    EraseRect(&expectPacketRect);
    EraseRect(&recvPacketRect);
    EraseRect(&pacSizeRect);
    EraseRect(&windowStartRect);
    EraseRect(&saveFileRect);

    PenSize(3,3);
    FrameRect(&expectPacketRect);
    FrameRect(&recvPacketRect);
    FrameRect(&pacSizeRect);
    FrameRect(&windowStartRect);
    FrameRect(&saveFileRect);
    PenSize(1,1);
    MoveTo(300,VERTICAL_PIXEL);
    DrawString(expectPacBuff);
    MoveTo( 300,VERTICAL_PIXEL  + 20);
    DrawString(receivedPacBuff);
    MoveTo( 300,VERTICAL_PIXEL  + 40);
    DrawString(totSizeBuff);
    MoveTo( 300,VERTICAL_PIXEL  + 60);
    DrawString(newWindBuff);
    MoveTo( 300,VERTICAL_PIXEL  + 80);
    DrawString(saveFileBuff);

}
```

```
/*****************************************************************************/
/*  This procedure Sends an acknowledgment to the sender for the packet num.*/
/*****************************************************************************/

SendAckSender(bufNumber)
int     bufNumber;
{

    long        cnt = 1;
    unsigned    char        temp,
                            templ;
    int         e;

    temp    = (char)ACK;
    templ   = (char)bufNumber;
    e = FSWrite(PortA.refout, &cnt, &templ);
    e = FSWrite(PortA.refout, &cnt, &temp);

}


/*****************************************************************************/
/*  This procedure Sends a Negative acknowledgment to the sender for the    */
/*  packet number that it should expect. To over come the problem of having */
/*  to search through the buffer to check for the retransmittd header, the  */
/*  receiver instead waits for one buff timeout to take place and then      */
/*  clears his input buffer. The Sender will wait for a time =              */
/*  1.4 * one Buffer Timeout before he retransmits. This ensures that the   */
/*  newly retransmitted packet does not turn up when the receiver is        */
/*  cleaning out his input buffer.                                          */
/*****************************************************************************/


SendNackSender(bufNumber)
int     bufNumber;
{

    long        tempTime,
                whatread,
                cnt = 1;
    char        temp,
                templ,
                temp2;
    int         e,got;

    nackReceived = TRUE;
    temp  = NACK;
    templ = bufNumber;
    e = FSWrite(PortA.refout,&cnt,&templ);
    e = FSWrite(PortA.refout,&cnt,&temp);
    tempTime = TickCount();
    do
    {
        TickCount();
    }while(TickCount() < tempTime + oneBuffTimeOut);
```

```
    ClearTheInputBuffer();
    endPrevWind = bufNumber - 1;
    sprintf(prBuff,"\p************** End Previous Buffer %d  ********",endPrevWind);
    DrawMyString(prBuff);
    if(endPrevWind < 0)
        endPrevWind += (int)TOTAL_BUFFERS;
    startWorkWindow = bufNumber;
    sendAckAt = startWorkWindow + WINDOW_LENGTH;
    if(sendAckAt > TURN_OVER)
        sendAckAt -= (int)TOTAL_BUFFERS;
    readBuffNum = bufNumber;
    readCompPacket = TRUE;

}



/*****************************************************************************/
/*  These next two procedures are not activated during the current running  */
/*  Session. These procedures can be brought into effect with very small    */
/*  changes. When a Nack is sent by the receiver these procedures go through*/
/*  the input buffer and search out the correct header. This is a laborous  */
/*  task and can be avoided by simply timing out and clearing the buffer.   */
/*****************************************************************************/


GetThisPacket()
{

    Boolean     correctHeader = FALSE,
                theBell = FALSE,
                corrSeq = FALSE;
    long        whatread;
    unsigned    char    tempHold[5];
    int         temp,e;



    while(!correctHeader)
    {
        while(whatread < 1)
            SerGetBuf(PortA.refin, &whatread);
        whatread = 1L;
        e = FSRead(PortA.refin, &whatread,tempHold); /* Grab the chars */

/*      sprintf(prBuff,"\p%c",tempHold[0]);
        DrawMyString(prBuff); */

        if(tempHold[0] == (unsigned char)BELL)
        {
            DrawMyString("\p");
            DrawMyString("\pI am in received Bell");
            theBell = TRUE;
        }
```

```
        else if(( tempHold[0] == readBuffNum) && (theBell == TRUE))
        {
            DrawMyString("\p I am in received correct seq num");
            corrSeq = TRUE;
            sprintf(prBuff,"\p I have received sequence number %d ",tempHold[0]);
            DrawMyString(prBuff);
        }
        else
            theBell = FALSE;

        if((theBell) && (corrSeq))
        {
            temp = getThePacSize();
            if(temp != -1)
            {
                while(whatread < 1)
                    SerGetBuf(PortA.refin, &whatread);
                whatread = 1L;
                e = FSRead(PortA.refin, &whatread,tempHold); /* Grab the chars */
                if(tempHold[0] == (unsigned char)BELL)
                {
                    DrawMyString("\pI HAVE RECEIVED A BELL HA HA ");
                    totCharInPacket = temp;
                    pacSeqNum = readBuffNum;
                    correctHeader = TRUE;
                }
                else
                    theBell = FALSE;
            }
            else
             theBell = FALSE;
        }
    }
}


/*************************************************************************/
/*          This is used in conjunction with the above procedure         */
/*************************************************************************/


getThePacSize()
{

    unsigned    int     sizeA,sizeB,TotalSize;
    unsigned    char    headerA,headerB,tempHold[5];
    long        whatread;
    int         e;

    while(whatread < 2)
        SerGetBuf(PortA.refin, &whatread);
    whatread = 2L;
    e = FSRead(PortA.refin, &whatread,tempHold); /* Grab the chars */
```

```
    headerA     = tempHold[0];
    headerB     = tempHold[1];
    sizeA = (headerA  & 0x00ff);
    sizeB = ((headerB << 8) & 0xff00);
    TotalSize = (sizeA | sizeB);


    if((TotalSize >= 0) && (TotalSize <= 2047))
    {
        sprintf(prBuff,"\pTotal Size of the packet is %d  ",TotalSize);
        DrawMyString(prBuff);
        return(TotalSize);
    }
    else
        return(-1);

}


/*************************************************************************/
/*This procedurre take the packet and writes it to the currently opened file*/
/*************************************************************************/

WriteToFile(tempBuff)
unsigned    char        *tempBuff;
{
    int     count;

    for(count = 0;count<= dataCharReceived;count++)
        fputc(*(tempBuff + count),fwrit);
}
```

```
/*************************************************************************/
/*                      PROGRAM     FILE                                 */
/*                      FOR PROGRAM MODITALK                             */
/*                      FILENAME:   sender.c                             */
/*                      DATE    :   JANUARY 1990                         */
/*                      AUTHOR  :   SOHRAB MODI                          */
/*************************************************************************/
#include            "TrialNetHeader.h"
#include            "Extern.h"

/*************************************************************************/
/*              This procedure opens the file for reading                */
/*************************************************************************/

OpenFileRead(theFilename)
char    *theFilename;
{

    if(theFilename == NULL)
    {
        sprintf(prBuff,"\pFILENAME IS NULL");
        DrawMyString(prBuff);
    }
    if((fp = fopen(theFilename,"r")) == NULL)
    {
        sprintf(prBuff,"\p    ERROR : CANNOT OPEN FILE READ");
        DrawMyString(prBuff);
    }
}

/*************************************************************************/
/*  On receiving a sender handshake from the other terminal the program  */
/*  checks to see if his terminal has already been configured as a receiver */
/*  If he has then he sends a ready to begin message or else accepts the */
/*  sender message and waits for the user to configure him as receiver.  */
/*************************************************************************/

HandleHandshakeOfSender()
{

    char        num;

    if(RECEIVER_TERM == TRUE)
    {
        DrawMyString("\p ***** I HAVE RECEIVED A SENDER  HANDSHAKE *****");
        myTermIs.receiver = TRUE;
        num = (4 & 0xff);
        putout(num);                                    /*  "procedures.c"  */
        dontSendReadySignal = TRUE;
    }
    else
    {
        myTermIs.receiver = TRUE;
        DrawMyString("\p***** I HAVE RECEIVED A SENDER HANDSHAKE *****");
    }
```

```
}

/*****************************************************************************/
/*  This Procedure is activated every time the sender has sent a complete   */
/*  packet. This procedure polls the input buffer to check for either acks  */
/*  or Nacks. If an ack comes in, it  stops a timer which was activated when*/
/* the last packet was sent. If the correct ack is not received then the    */
/*  control shifts normally from what it was dooing and retransmits the     */
/*  entire last window. If a Nack were to come in then again the control    */
/*  shifts normally from what it was dooing and retransmits the buffer.     */
/*****************************************************************************/

DataAtSenderPort()
{
    unsigned    char        dataAtPortIs[100],tempchar;
    int         e,temp,startPrevWindo;
    long        whatread = 0L;



    sprintf(prBuff,"\p ACK NACK TO RECEIVE IS   %d  ",ackToReceive);
    DrawMyString(prBuff);

    startPrevWindo = ackToReceive - WINDOW_LENGTH;
    if(startPrevWindo < 0)
        startPrevWindo += (int)TOTAL_BUFFERS;

    if(startTimer && (TickCount() > timeSentAt + timeOut))
    {
        sprintf(prBuff,"\pTHE TIMER IS ON THE TIME IS %ld       ",TickCount());
        DrawMyString(prBuff);
        sprintf(prBuff,"\pTIME OUT IS AT %ld       ",(timeSentAt + timeOut));
        DrawMyString(prBuff);
        HandleTimeOut();
    }
    else
    {
        tempchar = (char)ACK;
        SerGetBuf(PortA.refin, &whatread);                          /* "Tool Box" */
        if (whatread >1L)
        {
            e = FSRead(PortA.refin, &whatread, dataAtPortIs); /* "Tool Box" */
            if(!finishedReading)
            {
                if( dataAtPortIs[1] == tempchar )
                {
                    if ( dataAtPortIs[0] == ackToReceive)
                    {
                        sprintf(prBuff,"\p ACK  RECEIVED IS %d   ",dataAtPortIs[0]);
                        DrawMyString(prBuff);
                        startTimer = FALSE;
                        HandleAckReceived();
                    }
```

```
                else if(( dataAtPortIs[0] < ackToReceive) &&
                        ( dataAtPortIs[0] >= startPrevWindo))
                {
                    sprintf(prBuff,"\p ACK  RECEIVED IS %d   ",dataAtPortIs[0]);
                    DrawMyString(prBuff);
                    HandleReceiveNack(dataAtPortIs[0] + 1);
                    sprintf(prBuff,"\p Sending NACK for %d   ",(dataAtPortIs[0] +
                    DrawMyString(prBuff);
                }
            }
            else if(dataAtPortIs[1] == (char)NACK)
            {
                sprintf(prBuff,"\p ** NACK  RECEIVED IS %d   **",dataAtPortIs[0]);
                DrawMyString(prBuff);
                startTimer = FALSE;
                HandleReceiveNack(dataAtPortIs[0]);
            }
        }
        else if((dataAtPortIs[0] == (unsigned char)END_HANDSHAKE) &&
                        (dataAtPortIs[1] == tempchar))
        {
                startTimer = FALSE;
                HandleCloseFile();
        }
        else if((dataAtPortIs[0] == ackToReceive)
                        && (dataAtPortIs[1] == tempchar))
        {
            sprintf(prBuff,"\p *** ACK  RECEIVED  IS %d   ***",dataAtPortIs[0]);
            DrawMyString(prBuff);
            startTimer = FALSE;
            oldAckToReceive = ackToReceive;
            ackToReceive = oldAckToReceive + 1 + lastBuffer;
            if(ackToReceive > TURN_OVER)
                ackToReceive -= (int)TOTAL_BUFFERS;
        }
        else if(dataAtPortIs[1] == (char)NACK)
        {
            sprintf(prBuff,"\p *** NACK  RECEIVED  IS %d   ***",dataAtPortIs[0]);
            DrawMyString(prBuff);
            startTimer = FALSE;
            HandleReceiveNack(dataAtPortIs[0]);
        }
    }
  }
}



/*************************************************************************/
/*  If the ack is received keep track of the old ack in case of a time out  */
/*  then calculate the new ack to receive.                                  */
/*************************************************************************/
```

```
HandleAckReceived()
{
    oldAckToReceive = ackToReceive;
    ackIsReceived = TRUE;
    ackToReceive += NUM_OF_BUFFERS;
    if(ackToReceive > TURN_OVER)
        ackToReceive -= (int)TOTAL_BUFFERS;
    sprintf(ackBuff,"\p %d ",ackToReceive);
    /****    PrintSenderScreen();****/
}

/*************************************************************************/
/*  If a time out were to take place then treat it as a Nack but calculate  */
/*  which buffer retransmission must take place.                            */
/*************************************************************************/


HandleTimeOut()
{
    int     temp;

    numberOfTimeOuts++;
    if(lastPacketSent == FALSE)
        temp  = ackToReceive - WINDOW_LENGTH;
    else if(lastPacketSent == TRUE)
    {
        lastPacketSent = FALSE;
        temp = ackToReceive - lastBuffer;
    }
    if(temp < 0)
        temp -= (int)TOTAL_BUFFERS;
    startTimer = FALSE;
    HandleReceiveNack(temp);
    sprintf(timeOutBuff,"\p %d ",numberOfTimeOuts);
    /****    PrintSenderScreen(); ****/
}



/*************************************************************************/
/*  We generally come to this procedure when a ackReceived flag is set and  */
/*  whatever the sender was transmitting has been sent out successfully.    */
/*  this procedure now gets ready to send out the next 15 buffers. It first */
/* checks to see if the number of buffers to send is less than the current  */
/*  number of buffers read in. If not it reads in the difference. If it     */
/*  encounters an EOF while reading in the buffers it only tranmists upto   */
/*  the last buffer read else it transmits the next fifteen buffers.        */
/*************************************************************************/

HandleReceiveMyAck()
{
    int     sendNextBuffer,
            LastNumToSend,
            temp,temp1,temp2;
```

```
    if(ackIsReceived == TRUE)
        ackIsReceived = FALSE;

    temp = readNext + WINDOW_LENGTH;
    if(temp > lastBufferRead)
        ReadInDifference();
    else if(turnedOver == TRUE)
    {
        ReadInDifference();
        turnedOver = FALSE;
    }
    else
    {
        readNext += NUM_OF_BUFFERS;
        if(readNext > TURN_OVER)
            readNext -= (int)TOTAL_BUFFERS;
    }
    sendNextBuffer = lastBufferSent + 1;
    if(sendNextBuffer > TURN_OVER)
        sendNextBuffer -= (int)TOTAL_BUFFERS;

    if(finishedReading)
    {
        SendFileModem(sendNextBuffer,lastBuffer);
        if((nackReceived == FALSE) && (lastBufferSent != lastPacketNumber))
        {
            ackToReceive = lastPacketNumber;
            sprintf(ackBuff,"\P %d  ",ackToReceive);
            /** PrintSenderScreen;  ***/
            startTimer = FALSE;
            temp2 = lastBufferSent +1;
            if(temp2 > TURN_OVER)
                temp2 -= (int)TOTAL_BUFFERS;
            temp1 = lastPacketNumber - (lastBufferSent + 1);
            if(temp1 < 0)
                temp1 += (int)TOTAL_BUFFERS;
            SendFileModem(temp2,temp1);
        }
        if(nackReceived == FALSE)
        {
            SendHeader((int)NULL_SEQ,(int)END_HANDSHAKE,(int)1);
            lastPacketSent = TRUE;
        }

    }
    else
        SendFileModem(sendNextBuffer,WINDOW_LENGTH);
}



/*************************************************************************/
/*  This procedure tells the system that a Nack is received so it sets a  */
/*  flag to close down what it was doing and goes into handling the Nack  */
/*************************************************************************/
```

```
HandleReceiveNack(tempBuffNum)
unsigned    char    tempBuffNum;
{
        nackReceived = TRUE;
        nackRecvFor  = tempBuffNum;
}

/*************************************************************************/
/*  This Procedure sends through the modem port buffers starting from    */
/*  the buffer number buffNum and going through the number of buffers    */
/*  given in numOfBuff. Before sending the very last packet of that file it */
/*  sends out a special header. After having sent a packet it polls the  */
/*  input port to check for acks/Nacks. If it were to encounter a nack it */
/*  exists from the while loop and goes in to handle the Nack. If however it*/
/*  transmits a complete window then it puts a timer on and goes into the */
/*  main loop.                                                           */
/*************************************************************************/


SendFileModem(buffFrom,numOfBuff)
int     buffFrom,numOfBuff; /* specify which buffer to use as start & num
                               of buffers to be filled */
{
int     buffTo,count,count1,e,lastPacToSend = 0;
long    cnt = 1L;
char    ch;


    buffTo = buffFrom + numOfBuff;
    count  = buffFrom;

    while( count <= buffTo )
    {
         if(count > TURN_OVER)
        {
            count -= (int)TOTAL_BUFFERS;
            buffTo -= (int)TOTAL_BUFFERS;
        }
        sprintf(prBuff,"\p");
        DrawMyString(prBuff);
        sprintf(prBuff,"\p SENDING BUFFER NUMBER %d ",count);
        DrawMyString(prBuff);
        sprintf(pacSeqBuff,"\p %d  ",count);
        /****        PrintSenderScreen();    *****/

        if(( finishedReading == TRUE ) && ( count == buffTo ))
            lastPacToSend = 1;
        else
            lastPacToSend = 0;

        SendHeader( packetSize[count],count,lastPacToSend );

        bufferPtr = bufferCount[count];
        count1 = 0;
```

```
        while( count1 <= packetSize[count] )
        {
            e = FSWrite(PortA.refout, &cnt, (bufferPtr + count1));
            count1++;
        }
        lastBufferSent = count;
        DataAtSenderPort();
        if(nackReceived == TRUE) /* break out of this loop if Nack Rec */
            count = buffTo;
        count++;
    }
    lastPacToSend = 0;
    if(nackReceived == FALSE)
        StartTheTimer();
}



/*****************************************************************************/
/*  This procedure sends out the header consisting of 5 Bytes. The first     */
/*  Byte contains a BELL. The second Byte consists of the packet             */
/*  sequence number. The third & fourth byte consists of the packet size     */
/*  and the last Byte is again a BELL to signify the end of Header.          */
/*****************************************************************************/


SendHeader( sizeOfPacket,count,thisIsLast )
int         count,sizeOfPacket,thisIsLast;
{

unsigned    char        headerA,
                        packetSeqNumber,
                        headerB;

unsigned    char        temp;

unsigned    int         e,temp1,temp2,sizeA,sizeB,TotalSize;

long        cnt = 1L;


        if(!thisIsLast)
            temp = BELL;
        else
            temp = LAST_PACKET;

        headerA =   sizeOfPacket;
        headerB =  (sizeOfPacket >> 8);
        sizeA = (headerA  & 0x00ff);
        sizeB = ((headerB << 8) & 0xff00);
        TotalSize = (sizeA | sizeB);
        packetSeqNumber = count ;
        sprintf(prBuff,"\p TotalSize of packet is %d  ",TotalSize);
        DrawMyString(prBuff);
        sprintf(totSizeBuff,"\p %d  ",TotalSize);
        /*****      PrintSenderScreen();    *******/
```

```
        e = FSWrite( PortA.refout, &cnt, &temp );
        e = FSWrite( PortA.refout, &cnt, &packetSeqNumber );
        e = FSWrite( PortA.refout, &cnt, &headerA );
        e = FSWrite( PortA.refout, &cnt, &headerB );
        e = FSWrite( PortA.refout, &cnt, &temp );
}


/*****************************************************************************/
/*  This procedure simply keeps track of what time the packet was sent out   */
/*  at. It the calls the procedure CalculateTimeOut to calculate time out    */
/*  for the number of buffers specified.                                     */
/*****************************************************************************/

StartTheTimer()
{
    startTimer = TRUE;
    timeSentAt = TickCount();
    sprintf(prBuff,"\p THE TIMER IS STARTED.TIME SENT AT IS %ld    ",timeSentAt);
    DrawMyString(prBuff);
    CalculateTimeOut(2);
}


/*****************************************************************************/
/* This procedure calculates the time out as given in the formula. readTime */
/*  is a factor assigned for the program to do file-io.                      */
/*****************************************************************************/

CalculateTimeOut(timeOutFor)
int     timeOutFor;
{
    long        readTime = 200L;

    timeOut = (long)(((oneBuffTimeOut + propogationDelay) *
                (long)(timeOutFor)) + propogationDelay + readTime);
}


/*****************************************************************************/
/*  This procedure fills in buffers. As File-IO takes time the cursor        */
/*  changes into a watch indicating IO taking place. It reads in the number */
/*  of buffers specified. If while reading in the buffer it encountrers a    */
/*  end of packet marker it simply puts another marker after it and increses*/
/*  the packet size by 1. When it reads in 1023 characters it puts an end of*/
/*  packet marker and goes on to fill in the next buffer. If however it      */
/*  encounters an end of file marker, it sets a flag and then exists from    */
/*  the while loop. It keeps track of the last buffer read to compare with   */
/*  the last buffer sent in the procedure HandleReceiveAck. Before breaking */
/*  from this routine it returns the cursor back to the arrow.               */
/*****************************************************************************/
```

```
ReadIntoBuffer(buffFrom,numOfBuff)
int     buffFrom,numOfBuff; /* specify which buffer to use as
                                    start & number of buffers to be filled */
{
    int         count,
                count1,
                buffToFill,
                c,
                buffTo,
                dataSize,
                i,
                temp,
                temp1;
    Boolean     notTheEnd = TRUE;
    long        watchCount = 0;



    buffTo      = buffFrom + numOfBuff;
    buffToFill  = 0;
    count       = buffFrom;

     while((notTheEnd == TRUE) && (count <= buffTo))
     {


        if(count > TURN_OVER)
        {
            count -= (int)TOTAL_BUFFERS;
            buffTo -= (int)TOTAL_BUFFERS;
        }

        showWatchs(watchCount++);
        packetSize[count] = 0;
        dataSize = 0;
        bufferPtr = bufferCount[count]; /* user the ptr for the second arry */


        while((notTheEnd == TRUE) && (dataSize  <= DATA_CHAR))
        {

            /* get character at a time and store in the second
               dimension of the array */
            *(bufferPtr + (packetSize[count])) = (char)fgetc(fp);

            /* Get a Char at a time and make sure it is not end of file.
                Then make sure that the char just read in is not a bell */

            if(*(bufferPtr + (packetSize[count])) != EOF)
            {
                dataSize++; /*  a count to make sure that there
                                are 1024 data char per packet */
                HandleNotEOF(&count);
            }
```

```
            else if(*(bufferPtr + (packetSize[count])) == EOF)
            {
                /* if EOF reached midway in packet
                   then set flag for EOF  */
                notTheEnd = FALSE;
                finishedReading = TRUE;
                HandleEOF(&dataSize,&buffFrom,&count);
            }
            /* increment packet size this goes in the header */
            packetSize[count]++;
        }
        /* End of the packet put a bell */
        *(bufferPtr + packetSize[count]) = (char)BELL;
        sprintf(prBuff,"\pTHE PACKET SEQUENCE NUMBER = %d  , the Size is %d  ",count,p
        DrawMyString(prBuff);
        count++;    /* increment count for next packet */
    }
    if(!finishedReading)
        lastBufferRead = buffTo;
    else
    {
        lastBufferRead = count - 1;
        if(lastBufferRead < 0)
            lastBufferRead += (int)TOTAL_BUFFERS;
    }
    showArrow();
}



/*************************************************************************/
/*  This routine is called from the above routine. It handles the case when */
/*  an end of file marker is reached. NOTE: it does not include this in the */
/*  data.                                                                  */
/*************************************************************************/


HandleEOF(dataSize,buffFrom,count)
int     *dataSize,*buffFrom,*count;
{
        int         c;

        lastBuffer = *count - *buffFrom;
        if(lastBuffer < 0)
            lastBuffer += (int)TOTAL_BUFFERS;
/*      if((*buffFrom > 0 ) && (*buffFrom <= *count))
            lastBuffer += 1;        */
                                /* decrement the last read character as it
                                                    contains the EOF marker */
        packetSize[*count]--;
        lastPacketNumber = *count;
}
```

```
/*************************************************************************/
/*  This procedure is called from the procedure ReadBuffers. It checks to   */
/*  see that if there exists an end of packet marker in the data then to    */
/*  byte stuff the packet with another one.                                 */
/*************************************************************************/

HandleNotEOF(count)
int     *count;
{

                        /* check to see if that Char is a bell. if it is
                            put  another bell after it and increment */
        if(*(bufferPtr + (packetSize[*count])) == (char)BELL)
        {
                    /*increment the packet size but not the data size */
            packetSize[*count]++;
            *(bufferPtr + (packetSize[*count])) = (char)BELL;
        }
}



/*************************************************************************/
/*  Whenever a Nack or time out takes place control eventually shifts here. */
/*  if before control shifts here the entire file has already been read into*/
/*  the buffers then the procedure Retransmit bad buffers is called which   */
/*  calculates the buffers to send and then transmits them. If the end of   */
/*  is not reached then the number of buffers to send is calculated. Now if */
/*  there exists buffres which have been read in but not sent and is equel  */
/*  to the number of buffers to send then again the routine to retransmit is*/
/*  called upon. If therte does not exist enough buffers to send then the   */
/*  difference is read in and then retransmitted. If however while reading  */
/*  the buffers here the End of file was encountered then a flag is set and */
/* the case handled seperatly when transmitting the buffers as seen in the  */
/*  procedure RetransmitBadBuffers.                                         */
/*************************************************************************/

SendNackBuffer()
{

    unsigned   char temp[2];
    int         e,newSend,oldLastBuffRead,toSend,turnOver,
                finishedReadingHere;


    finishedReadingHere = (int)FALSE;
    turnOver            = (int)FALSE;

    startTimer          = FALSE;
    if(!finishedReading)
    {
        oldLastBuffRead = lastBufferRead;
        ReadTheRest(&turnOver);
        GetAckToReceive(&finishedReadingHere,&oldLastBuffRead,&toSend);
    }
```

```
    sprintf(prBuff,"\p ACK TO RECEIVE IS %d  ",ackToReceive);
    DrawMyString(prBuff);
    sprintf(ackBuff,"\P %d  ",ackToReceive);
    /*** PrintSenderScreen; ***/
    HandleTimeToClearBuffer();
    ackIsReceived = FALSE;
    nackReceived = FALSE;
    RetransmitBadBuffers(&finishedReadingHere,&toSend,&turnOver);
}



/*****************************************************************************/
/*  This procedure as explained above, will retransmit bad buffers depending*/
/*  on different situations.
/*****************************************************************************/

RetransmitBadBuffers(finishedReadingHere,toSend,turnOver)
int     *finishedReadingHere,*toSend,*turnOver;
{


    int     temp,temp1;

    if(finishedReading)
    {
        if(*finishedReadingHere == (int)TRUE)
            *finishedReadingHere = (int)FALSE;
        else
        {
            *toSend = ackToReceive - nackRecvFor;
            if(*toSend < 0)
                *toSend += (int)TOTAL_BUFFERS;
        }

        SendFileModem(nackRecvFor,*toSend);
        if((nackReceived == FALSE) && (lastBufferSent != lastPacketNumber))
        {
            ackToReceive = lastPacketNumber;
            sprintf(ackBuff,"\P %d  ",ackToReceive);
            /** PrintSenderScreen;  ***/
            startTimer = FALSE;
            temp = lastBufferSent +1;
            if(temp > TURN_OVER)
                temp -= (int)TOTAL_BUFFERS;
            temp1 = lastPacketNumber - (lastBufferSent + 1);
            if(temp1 < 0)
                temp1 += (int)TOTAL_BUFFERS;
            SendFileModem(temp,temp1);
        }
         if(nackReceived == FALSE)
        {
            SendHeader((int)NULL_SEQ,(int)END_HANDSHAKE,(int)1);
            lastPacketSent = TRUE;
        }
```

```
    }
    else
    {
        SendFileModem(nackRecvFor,WINDOW_LENGTH);
        if(nackReceived == FALSE)
        {
            HandleReceiveMyAck();
            if(*turnOver == (int)TRUE)
                turnedOver = TRUE;
            *turnOver = (int)FALSE;
        }
    }
}


/*************************************************************************/
/* This procedure will calculate the next ack to receive depending on the   */
/*  Nack received.                                                          */
/*************************************************************************/


GetAckToReceive(finishedReadingHere,oldLastBuffRead,toSend)
int         *finishedReadingHere,*oldLastBuffRead, *toSend;
{

    int     BuffLeftToSend;

        if(!finishedReading)
            ackToReceive = nackRecvFor + WINDOW_LENGTH;
        else
        {
            BuffLeftToSend = *oldLastBuffRead - nackRecvFor;
            if(BuffLeftToSend < 0)
                BuffLeftToSend += (int)TOTAL_BUFFERS;
            *toSend = BuffLeftToSend + lastBuffer;
            lastBuffer = *toSend;    /* this is done in case a timout happens */
            ackToReceive  = nackRecvFor + *toSend;
            *finishedReadingHere = (int)TRUE;
        }

        if(ackToReceive > TURN_OVER)
            ackToReceive = ackToReceive - (int)TOTAL_BUFFERS;
}


/*************************************************************************/
/*  This procedure simply waits for 1.4 * one buffer timeout, to give the   */
/*  receiver enough time to clear out his input buffer and get ready to     */
/*  receive this retransmission.                                            */
/*************************************************************************/

HandleTimeToClearBuffer()
{

    long        tempTime;
```

```
    tempTime = TickCount();
    do
    {
        TickCount();
    }while(TickCount() < tempTime + (1.4 * oneBuffTimeOut));

}


/*****************************************************************************/
/*  This procedure is called from SendNackBuffer. This checks to see if      */
/*  there are enough buffers read but not sent. If there are not then it     */
/*  calls the routine ReadInDifference to read in the rest of the buffers    */
/*****************************************************************************/

ReadTheRest(turnOver)
int     *turnOver;
{

    int     temp1;
    Boolean tempTurned = FALSE;

    if(turnedOver == TRUE)
    {
        turnedOver = FALSE;
        *turnOver = (int)TRUE;
    }
    readNext = nackRecvFor;
    temp1 = readNext +  WINDOW_LENGTH;
    if(temp1 > TURN_OVER)
    {
        tempTurned = TRUE;
        temp1 -= (int)TOTAL_BUFFERS;
    }
    if(temp1 >lastBufferRead)
        ReadInDifference();
    else if((tempTurned ) && (temp1 - lastBufferRead < 0))
        ReadInDifference();
    else
    {
        readNext += NUM_OF_BUFFERS;
        if(readNext > TURN_OVER)
            readNext = readNext - (int)TOTAL_BUFFERS;
    }
    tempTurned = FALSE;
}


/*****************************************************************************/
/*  This procedure calculates the number of buffers to send. It then         */
/*  calculates the number of buffers read but not sent. It then reads in     */
/*  the difference so that there exist enough buffers now left to send.      */
/*****************************************************************************/
```

```
ReadInDifference()
{

    int     temp,templ,theNext;


    DrawMyString("\p I AM IN READ DIFFERENCE");
    temp = (readNext + WINDOW_LENGTH);
    if(temp > TURN_OVER)
        temp -= (int)TOTAL_BUFFERS;
    templ = temp - lastBufferRead -1;
    if(templ < 0)
        templ += (int)TOTAL_BUFFERS;
    theNext = lastBufferRead + 1;
    if(theNext > TURN_OVER)
        theNext -= (int)TOTAL_BUFFERS;
    ReadIntoBuffer(theNext,templ);
    readNext += NUM_OF_BUFFERS;
    if(readNext > TURN_OVER)
    {
        turnedOver = TRUE;
        readNext -= (int)TOTAL_BUFFERS;
    }
}

/*****************************************************************************/
/*  This Procedure will print the need data on the screen. This routine is  */
/*  also called if an update event takes place.                              */
/*****************************************************************************/

PrintSenderScreen()
{
    TextSize(9);
    TextFace(bold);
    EraseRect(&seqRect);
    EraseRect(&pacSizeRect);
    EraseRect(&ackRect);
    EraseRect(&timeOutRect);

    PenSize(3,3);
    FrameRect(&seqRect);
    FrameRect(&pacSizeRect);
    FrameRect(&ackRect);
    FrameRect(&timeOutRect);
    PenSize(1,1);

    MoveTo(300,VERTICAL_PIXEL);
    DrawString(pacSeqBuff);
    MoveTo( 300,VERTICAL_PIXEL  + 20);
    DrawString(totSizeBuff);
    MoveTo( 300,VERTICAL_PIXEL  + 40);
    DrawString(ackBuff);
    MoveTo( 300,VERTICAL_PIXEL  + 60);
    DrawString(timeOutBuff);

}
```

```
/*************************************************************************/
/*This procedure closes the file pointer and gets ready to end transmission.*/
/*************************************************************************/


HandleCloseFile()
{
        finishedWriting = TRUE;
        fclose(fp);
}
```