

*IN THE NAME OF ALLAH , THE MOST
GRACIOUS AND MERCIFUL*

COM
245

CTCNET FILE EXCHANGE UTILITY
FOR SCO XENIX AND AIX

BY

Reference
Quaid-i-Azam University
Central Library

MUHAMMAD ILYAS

A dissertation submitted to
Quaid-i-Azam University Islamabad
As a partial fulfilment of the requirement
For M.Sc. Degree in
COMPUTER SCIENCES

FEBRUARY, 1991.

QUAID-I-AZAM UNIVERSITY
(DEPARTMENT OF COMPUTER SCIENCES)

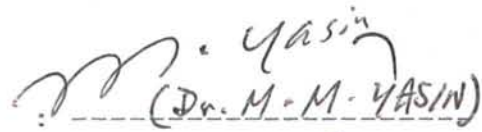
DATED: / /1991

FINAL APPROVAL

This is to certify that we have read the thesis submitted by Mr. Muhammad Ilyas and it is our judgment that this thesis is of sufficient standard to warrant its acceptance by the Quaid-i-Azam university Islamabad, for the degree of master of science in COMPUTER SCIENCES.

COMMITTEE.

1. EXTERNAL EXAMINER

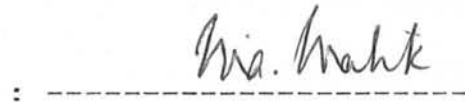

: ----- (Dr. M. M. YASIN)

2. SUPERVISOR


: -----

DR. MUHAMMAD AFZAL BHATTI
DEPARTMENT OF COMPUTER SCIENCES,
QUAID-I-AZAM UNIVERSITY,
ISLAMABAD.

3. CHAIRMAN


: -----

DR. MASUD AHMAD MALIK
DEPARTMENT OF COMPUTER SCIENCES,
QUAID-I-AZAM UNIVERSITY,
ISLAMABAD.

TO MY MOTHER

PROJECT BRIEF

PROJECT TITLE : CTCNET FILE EXCHANGE UTILITY
FOR SCO XENIX AND AIX.

OBJECTIVE : To Implement Enhanced CTCNET
File Exchange Protocol on
SCO XENIX and AIX.

OFFERED BY : Computer Training Centre

UNDERTAKEN BY : Muhammad Ilyas.

INTERNAL SUPERVISOR : Dr. Muhammad Afzal Bhatti.
Dept. Of Computer Sciences
Quaid-i-Azam University
Islamabad.

EXTERNAL SUPERVISOR : Dr. Faiz Ishaq
Head
Computer Training Centre
H-9, Islamabad.

DATE OF STARTING : September, 1990.

DATE OF COMPLETION : February, 1991.

LANGUAGE USED : Microsoft C ver 5.1,
XENIX C, AIX C.

OPERATING SYSTEMS USED : SCO XENIX, AIX, PC/DOS
MS/DOS.

SYSTEMS USED : PC AT, 386 Based
IBM RISC/6000 Model 320.

A B S T R A C T

This dissertation is concerned with the Implementation of Enhanced CTCNET File Exchange Protocol on SCO XENIX and AIX. The implementation of the CTCNET software on these systems provide a very comprehensive solution for Source/text file exchange across systems as well as transfer of all kinds of files from one system to another for backup, media interchange or transportation.

A C K N O W L E D G M E N T S

First, of all I am indebted to thank the Supreme Ruler the Almighty God whose divine help made me to complete this project successfully, no one, undoubtedly, can reach one's destination without HIS guidance.

As our intellect, our acumen and our abilities are all through the agencies of our teachers, I would like to thank my honourable teachers for the knowledge and proficiency I acquired from them. Special thanks must go to Dr. Muhammad Afzal Bhatti for his kind supervision and frequent help. His wise counsels and unfailing kindness meant so much to me.

I am deeply grateful to Dr. Faiz Ishaq, Head CTC, for helping me a lot in carrying out this project. I owe lots of thanks to Mr. Ikram-ul-Haq (PSO at CTC), as he guided me throughout the project. I am also thankful to the people working at CTC for their generous cooperation. I am, especially, much obliged to Mr. M. Javed Iqbal, Ms. Ghulam Fiza, Mr. Muhammad Jaudet and Mr. M. Farooq Azam for their kind consideration and assistance. They were always available with time and work whenever I claimed their help.

All my classfellows deserve acknowledgments, who had been very cooperative and helping towards my timely insensical attitude.

I am under obligation to pay sincerest thanks to my parents, my family members and my friends for their valuable moral support and constant encouragement.

Finally thanks and love to all those people also, who are irreplaceable for me and care to be worried about me.

P R E F A C E

This dissertation comprises six chapters. These chapters have been formed and arranged in such a way that the details of the project can be given in a simple but well-grounded form.

First chapter of this thesis gives a brief introduction to present system. It introduces the project and organization where this project has been conducted, to the reader. It also describes the need of carrying out this project. Chapter two describes the communication modes. Facilities provided by serial communication devices. It also holds discussion on the existing system. Chapter three presents the proposed extension to the existing system. The features which are to be added, have been described in detail. Chapter four contains a discussion on the data structures. It also describes the parameter setting of the terminal and programmes that are designed in this course. Chapter five is the **user system interface** to guide a new user of this system. Chapter six is the final chapter it describes the major features, the features lacking and the possible future extensions in the system.

Appendix A describes the complete Sender and Receiver logic of the Enhanced CTCNET File Exchange Protocol. Appendix B describes the format of the data structure header used for non text files. Appendix C represents the control characters used by the CTCNET software. Appendix D represent the messages reported by the CTCNET for the user facility, Appendix E describes the error messages displayed when the error occurs.

CONTENTS

	Page No.
1 INTRODUCTION	
1.1 INTRODUCTION TO CTCNET	1
1.2 INTRODUCTION TO PROJECT	2
1.3 INTRODUCTION TO COMPUTER TRAINING CENTER	4
1.4 OBJECTIVES OF THE PROJECT	5
2 EXISTING SYSTEM	
2.1 MODES OF TRANSMISSION	7
2.1.1 PARALLEL MODE	7
2.1.2 SERIAL MODE	8
2.1.2.1 ASYNCHRONOUS TRANSMISSION	8
2.1.2.2 SYNCHRONOUS TRANSMISSION	9
2.1.2.3 ISOCHRONOUS TRANSMISSION	10
2.2 SERIAL COMMUNICATION HARDWARE	10
2.2.1 MODEM	11
2.2.2 COMMUNICATION BETWEEN TWO COMPUTERS	11
2.2.3 A SERIAL INTERFACE 8251A	13
2.2.3.1 INITIALIZING 8251A	15
2.2.4 RS-232C SERIAL DATA STANDARD	17
2.2.4.1 RS-232C SIGNAL PINS	17
2.5 CTCNET PROTOCOL (Serial Communication Software)	18
2.5.1 INITIAL PROTOCOL AND EARLY IMPLEMENTATIONS	18
2.5.2 VMS AND DOS IMPLEMENTATIONS	20
2.5.3 Revised CTCNET File Exchange Protocol	23
2.6 EVALUATION OF CTCNET	24

3	PROPOSED SYSTEM	
3.1	IMPLEMENTATION OF THE REVISED CTCNET	25
	FILE EXCHANGE PROTOCOL	
3.2	IMPLEMENTATION OF CTCNET ON SCO XENIX AND AIX	26
3.3	PORTABILITY STANDARD	26
3.4	FILE TRANSFER	26
4	SYSTEM DESIGN AND DEVELOPMENT	
4.1	SOFTWARE SELECTION	28
4.2	DATA STRUCTURES	29
4.3	SERIAL PORT SELECTION	29
4.4	BAUD RATE SELECTION	30
4.5	READING FROM THE TERMINAL CONNECTED TO THE SERIAL PORT	32
4.6	TRANSMISSION OF ASCII FILES	32
4.6.1	TRANSMISSION OF ASCII FILES FROM SCO XENIX TO MS-DOS	34
4.6.2	TRANSMISSION OF ASCII FILES FROM SCO XENIX TO AIX	34
4.6.3	TRANSMISSION OF ASCII FILES FROM MSDOS/PCDOS TO SCO XENIX AND AIX	34
4.6.4	TRANSMISSION OF NON TEXT FILES	35
4.7	DATA TRANSMISSION IN CASE OF SEND/RECEIVE	35
4.7.1	SOFTWARE HANDSHAKE	36
4.7.2	INTEGRITY IN DATA TRANSMISSION	36
4.7.3	WORKING OF SEND	37
4.7.4	WORKING OF RECEIVE	38
4.8	FILE PACKAGING/UNPACKAGING	41
4.8.1	WORKING OF FILE PACKAGING	42

4.8.2	WORKING OF UNPACKAGING	43
5	USER SYSTEM INTERFACE	
5.1	HOW TO INVOKE THE CTCNET	45
5.1.1	SEND A FILE	45
5.1.2	RECIEVE FILE	46
5.2	USING FILE PACKAGING/UNPACKAGING	46
5.2.1	EXAMINE FILE	47
5.2.2	PACKAGE FILE	47
5.2.3	UNPACKAGE FILE	47
6	SYSTEM CONCLUSION	
6.1	SOME MAJOR FEATURES	48
6.2	REMEDIES	49
6.3	FUTURE EXPANSIONS	50
APPENDIX		
A	Enhanced CTCNET File Exchange protocol	
B	Header Format	
C	Special Characters	
D	Messages	
E	Error Messages	
BIBLIOGRAPHY		

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO CTCNET

In 1982, the CTC acquired sixteen Systime S-500 computer systems to be used for training purposes. These were 8086 microprocessor based machines running the CP/M-86 operating system. Each machine had a single 8-inch floppy disk drive. An Intellec Series II Microprocessor Development System was also purchased by the CTC. It used the Intel ISIS-II operating system. This system had three 8-inch floppy disk drives but these were not media compatible with the S-500 drives. It was necessary to evolve a scheme to transfer files across these systems through the use of the serial ports. This was the main motivation for defining a simple protocol for the exchange of data packets between computers of different origins using the standard RS 232 lines. The protocol was implemented in a package of programs developed to allow file exchange as well as the use of shared printers for the Systime S-500 and the Intellec Series II MDS. It was given the name CTCNET. The initial CTCNET protocol fixes the parameters to 9600 baud, no parity, 8 data bits and one stop bit. CTCNET was originally developed for transferring files (ASCII or binary) from different types of computers having different operating systems.

In 1985, the CTC had acquired a VAX-11/730 computer running the VAX/VMS operating system as well as a few IBM PC and compatible computers with the MSDOS/PCDOS (DOS) operating system. A Post Graduate Training Program (PGTP) project was, therefore, initiated to carry out the implementation of the CTCNET on the VMS and DOS operating systems. The work on the improvement of DOS and VMS implementations of CTCNET continued. The CTCNET File

Exchange Protocol was revised and improved in the beginning of 1990. An effort was made to get rid of all indefinite wait conditions as well as to report errors on the remote computer.

In the earlier version of the CTCNET File Exchange Protocol indefinite wait loops were executed at the time of initial handshake and also while waiting for the start of a packet or for its acknowledgment. The revised version specifies the time limits placed for waiting for the various conditions.

1.2 INTRODUCTION TO PROJECT

The usefulness of the CTCNET software continued to induce its development and evolution at the CTC. The acquisition of the SCO XENIX operating system by the CTC necessitated the implementation of CTCNET for this system. The University Grants Commission also acquired the IBM RS/6000 computer having AIX operating system and it was also considered to implement the software on this machine as well. Also provide a solid solution for transfer of non-text files mainly for backup, media interchange or transportation purposes. A project is initiated for implementing the latest version of the CTCNET software on SCO XENIX and AIX, also developing the file packaging and unpackaging software. For this the following facilities have to be provided.

a) **Avoid Indefinite Wait**

In the earlier version 2.00 of CTCNET File Exchange protocol(that transfer data files from VAX/VMS to MS-DOS and vice versa) indefinite wait loops were executed at the time of initial hand shake, waiting for the control characters and start of packet or for its acknowledgment. This resulted in frequent

hangup conditions if a character was missed or if the computer at one end went down. Due to this the new version of the CTCNET on SCO XENIX and AIX should provide the time out limit in receiving the characters from the serial line.

b) Portability

CTCNET was developed to transfer files across different systems having different operating systems through the use of the standard serial port RS 232C. So it was realized that the new system would continue to come in. With the acquisition of these new systems such as SCO XENIX and AIX it is better to write the CTCNET code in such a language which provide much more portability.

**c) Maintaining Original Characteristics And Functionality
Of The File**

The non-text files are transferred from one operating environment to an other mainly for backup, media interchange or transportation purposes, not to be usable on the other operating environment. So transferring non-text files from one environment to an other the file is packaged. The packaging software would package the contents of a file along with a header containing all its essential attributes and characteristics into an other file. This file could be transferred from one operating environment to an other, let us suppose you transferred packaged file from one environment SCO XENIX, AIX, DOS in the environment AIX and DOS this file is not packaged or unpackaged, but if you brought back to environment SCO XENIX upon return to environment SCO XENIX the file is unpackaged. The unpackaged software would read the header restore it and guarantee that the file has its original

characteristics and functionality.

1.3 INTRODUCTION TO COMPUTER TRAINING CENTER

Before discussing the project objectives of the study, it seems appropriate to say a few words about Computer Training Center (CTC), where this project was undertaken. Effective utilization of computers and the integration of microprocessors in intelligent automated equipment and industrial systems requires an understanding of computers beyond the mere knowledge of programming. An in depth understanding of the computer hardware, software and system design is required for the proper utilization and adaptation of computer technology to solve the problems facing us in our country. An integrated hardware and software knowledge base is therefore necessary. CTC is playing an important role in this connection.

In order to develop an infrastructure of computer science and engineering, University Grants Commission and Pakistan Atomic Energy Commission jointly set up the Computer Training Center. The Computer Training Center was established at the University Grants Commission Campus in August, 1982. the basic motive behind the establishment of CTC was to provide an institution for acquiring root-level knowledge about computer architecture and its functioning.

CTC has following systems for the purpose of research and training.

- VAX-11/730
- PDP-11/23
- Cromemco System 300

- Microprocessor Development System (Intel 287FD)
- IBM PC/AT
- ACORN Cambridge Workstations
- Systime S-500
- Intel SBC 86/12-A single board computer
- Ai-M16 Microcomputer
- Universal Programmer
- CTC PC/XT Lab It consists of Seventeen IBM PC/XT compatible computers, which are linked with each other in a group of four through a local area network. Sixteen workstations are for students and one for the instructor.

CTC is also working as a research center. The highly qualified and able faculty of CTC has worked on many useful projects. CTC developed in depth knowledge at the machine level for VAX 11/730 computer as well as the IBM PC family and compatible computers.

1.4 OBJECTIVES OF THE PROJECT

Keeping in view the requirements discussed in section 1.1, it is felt necessary to consider the implementation of the CTCNET software on SCO XENIX, AIX and DOS operating systems which will provide a very comprehensive solution for file exchange across the systems. The new version specifics the time limits as well as to report errors on the remote computer.

The portability of the new version of the CTCNET was highly required. The packaging/unpackaging software will guaranteed that the packaged file in one environment is not packaged or unpackaged in the other environment, but upon return

to the original environment the unpackaged software read the file header and restore the file with its original characteristics and functionality.

CHAPTER 2

EXISTING SYSTEM

The purpose of this chapter to describe the communication modes, serial communication is described as it is the only mode relevant to the work, facilities provided by serial communication devices, signal standards used by the CTCNET, the existing CTCNET software system.

2.1 MODES OF TRANSMISSION

Systems that transmit data must have consistent methods of transmission over communication channels. Data can be set over communication channels in two modes.

1 Parallel mode

2 Serial mode

2.1.1 PARALLEL MODE

The internal transfer of data within modern computers is done in parallel mode, because it is the fastest way of transmission. But this type of transmission is not possible to transmit data at long distances. In parallel transmission n bits are sent in one time cycle, all the bits of character are sent simultaneously either over separate lines or on different frequencies of the same line. That is why parallel transmission is not possible on low speed lines because its primary purpose is to speed up transmission between two points.

It requires higher cost transmission lines than serial transmission. Nether it is used on long distances, because the bit drift back and forth in time relation one an-other and may interfere with the bits of the preceding and following character.

2.1.2 SERIAL MODE

In serial transmission the transmitting device sends a bit followed by the time interval then a second bit and so on, until all the bits are transmitted. It takes n time cycles to transmit n bits. The mode that is used to transmit data over long distances is called Serial mode. In serial mode we use the low speed lines, which are not much costly and data can be sent over single line or pair of lines.

Mostly data over long distances is transferred serially. Three common usable transmission modes in Serial communication are .

- 1 Asynchronous transmission
- 2 Synchronous transmission
- 3 Isochronous transmission

2.1.2.1 ASYNCHRONOUS TRANSMISSION

Asynchronous transmission mode is often referred to as stop-start transmission. Because each data character has a bit which identifies its start and one or two bits which identify its end. Since each character is identifies individually, so character can be sent at any time in the same way that a person typed at key board at different rates. This is because the transmitting device can transmit a character at any time that is convenient and the receiving device will accept that character. Character can be sent at irregular intervals, for example one character per second or one character and then a ten second wait, to enable the receiver to recognize a character when it arrives. The Figure 2.1 shows the bit format often used for transmitting

asynchronous serial data.

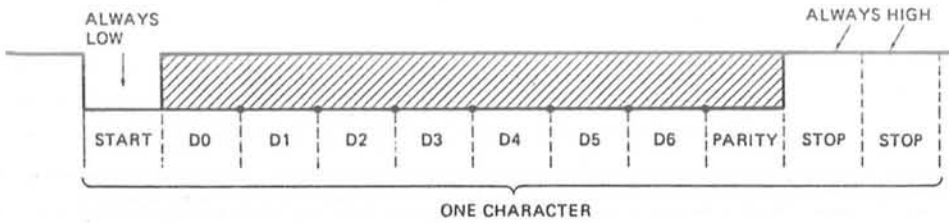


Figure 2.1

When no data is being sent the signal line is in a constant high or marking state. The beginning of data character is indicated by the line going low for one bit time this bit is called start bit. the data bits are then sent out on the line one after the other, depending on the system the data word may consist of 5,6,7 or 8 bits. The parity bit follows the data bits which is used to check errors in received data. Some systems do not insert or look for a parity bit. After the data bits and the parity bit the signal line is returned high for at least one bit time to identify the end of character. This always high bit is referred to as a stop bit. Some systems use two stop bits. So for transmitting a 7 bit data word such as an ASCII character 10 or 11 bits time are required.

2.1.2.2 SYNCHRONOUS TRANSMISSION

Synchronous transmission mode is used for the high speed transmission of a block of characters. In this method of transmission, both the sending device and the receiving device

are operated simultaneously and are resynchronized after each few thousand data signal bits are transmitted. Start/stop are not required for each character. Synchronization is established and maintained either when the line is ideal or just prior to the transmission of a data signal. This synchronization is established by passing a predetermined group of "sync" characters between the sending and the receiving devices.

The sending device sends a long stream of data bits that may have thousands of bits. The receiving device, knowing what code is being used, counts for the appropriate number of bits and assumes this is the first character and passes it to the computer. It then counts for the second character and so on.

2.1.2.3 ISOCHRONOUS TRANSMISSION

Isochronous transmission combines the elements of both synchronous and asynchronous data transmission. In isochronous transmission, as in asynchronous, each character is required to have both a start bit and a stop bit. However, as in synchronous data transmission, the transmitter and receiver are synchronized.

2.2 SERIAL COMMUNICATION HARDWARE

Interfacing a computer with serial data lines the data must be converted to and from serial form. A parallel-in-serial-out shift register and a serial-in-parallel-out shift register must be used to do this. For some cases of serial data transfer, hand shaking circuitry is also required to make sure that the transmitter does not send data faster than it can be read in by the receiving system. There are several programmable LSI devices available which contain most of the circuitry needed for serial

communication. A device such as Intel 8251A which can be programmed to do either asynchronous or synchronous communication is often called a universal synchronous-asynchronous receiver transmitter or USART.

2.2.1 MODEM

Before discussing how serial communication can be performed between two computers, it is better to give brief introduction about the modem. The word modem stands for two words **modulator** and **demodulator**. The modem that is transmitting the signal is the modulator, because it modulates or put some form of intelligence on the carrier wave. Receiving equipment is the demodulator because it demodulates or interprets the signal upon its receipt. The modem takes the binary signal (digital signal) from a computer and modulate it so it will become a continuous signal (analog signal) that can be transmitted over telephone lines. On the receiving side the modem demodulates the modulated carrier signal (analog signal) and converts it to a binary signal that will be meaningful to the computer.

2.2.2 COMMUNICATION BETWEEN TWO COMPUTERS

Modems and other equipment used to send serial data over long distances are known as, **data communication equipment (DCE)**. The terminals and computers that are sending the serial data are known as, **data terminal equipment (DTE)**. The signal names shown in Figure 2.2 are part of a serial data communications standard called RS-232C, which will be discussed in detail in this chapter. Here we just give an overview about the signals being

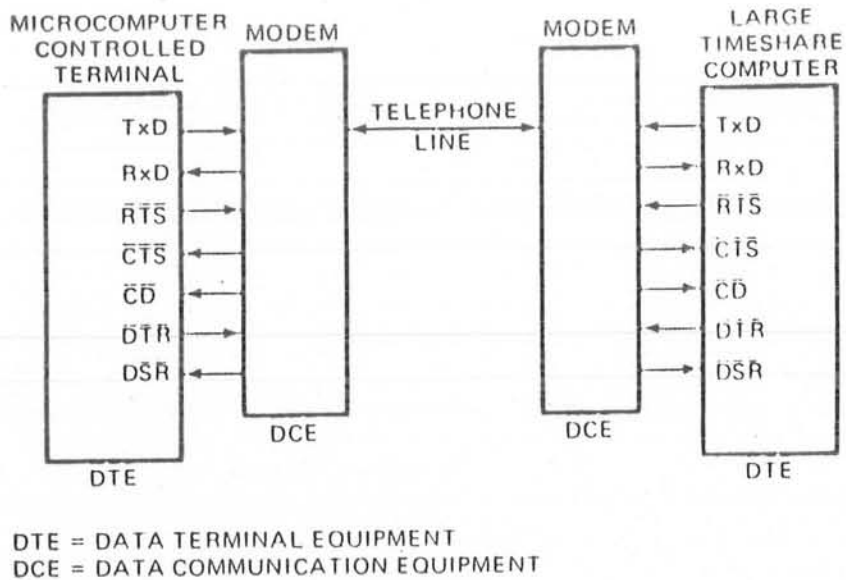


Figure 2.2

used. Note the direction arrowheads on each of these signals. Here is a sequence of signals that might occur when a user at a terminal wants to send some to the computer.

After the terminal power turned on and terminal runs any self_checks, it asserts the 'data terminal ready' (DTE) signal to tell the modem it is ready. When it is powered up and ready to transmit or receive data, the modem will assert the 'data-set - ready'(DSR) signal to the terminal. Under terminal control the modem then dials up the computer. If the computer is available,

it sends back a specified tone. Now, when the terminal has a character actually ready to send, it will assert a 'request-to-send' (RTS) signal to the modem. The modem will then assert its 'carrier-detect' (CD) signal to the terminal to indicate that it has established contact with the computer. When the modem is fully ready to transmit data it asserts the 'clear-to-send' (CTS) signal back to the terminal. The terminal then sends serial data characters to the modem. When the terminal has sent all the characters it wants to, it makes its RTS signal high. This causes the modem to unassert its CTS signal and stop transmitting. A similar handshake occurs between the modem and the computer at the other end of the data link.

2.2.3 A SERIAL INTERFACE 8251A

The 8251A is used as the serial port on the IBM PC synchronous communication board, and on many other boards. Figure 2.3 shows a block diagram and pin descriptions for the 8251A. The eight parallel lines, D7-D0 connect to the system data bus so that data words and control/status words can be transferred to and from the device. The chip-select CS input is connected to an address decoder so the device is enabled when addressed. The 8251A has two internal addresses, a control address which is selected when the C/D input is high, and a data address which is selected which is selected when the C/D input is low. The RESET, RD and WR lines are connected to the system signals with the same names. The clock input of the 8251A is usually connected to the system clock to synchronize internal operations with system operations.

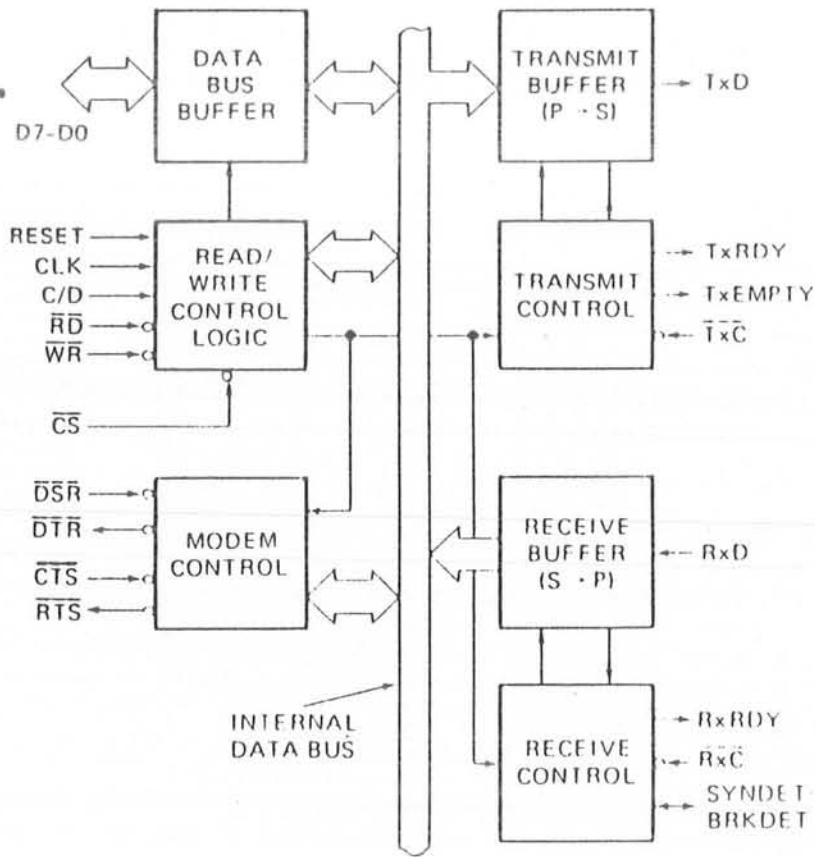


Figure 2.3

The signal labeled TxD on the upper right corner of the 8251A block diagram is the actual serial-data output. The pin labeled RxD is the serial-data input. The shift registers in the UART require clocks to shift the serial data in and out. Tx̄C is the transmit shift-register clock input, Rx̄C is the receive shift-register clock input. Usually these two inputs are tied together so they are driven by the same clock frequency. The

frequency of the applied clock signal must be 1 or 16 or 64 times the transmit and receive baud rate. Using a clock frequency higher than the baud rate allows the receive shift register to be clocked at the center of a bit time rather than at a transition. This reduces the chance of noise at a transition causing a read error.

The 8251A uses a double-buffered. Which means that one character can be loaded into a holding buffer while another character is being shifted out of the actual transmit shift register. The TxRDY output from the 8251A will go high when the holding buffer is empty and another character can be sent from the CPU. The TcEMPTY pin on the 8251A will go high when both the holding buffer and the transmit shift register are empty. The RxRDY of the 8251A will go high when a character has been shifted into the receiver buffer and is ready to be read out by the CPU. Incidentally, if a character is not ready out before another character is shifted in, the first character will be overwritten and lost.

The sync-detect/break-detect (SYNDET/BD) pin has two uses. When the device is operating in asynchronous mode, which we are interested in here, this pin will go high if the serial data input line ,RxD stays low for more then two character times, This signal then indicates an intentional break in data transmission or a break in the signal line.

The four signal connected to the box labeled MODEM CONTROL in the 8251A block diagram are handshake signals.

2.2.3.1 INITIALIZING 8251A

8251A is initialized with the mode word and a command word. Figures 2.4(a) and 2.4(b) show the formats for these words. Baud rate factor, specified by the two least-significant bits of the mode word. The character length specified by bits D2 and D3 in the mode word includes only the actual data bits, not the start bit, parity bit or stop bit(S).

After the mode word command word is send to an 8251A. A '1' in the least significant bit of the command word enables the transmitter section of the 8251A and the TxRDY output. When enabled, the 8251A TxRDY output will be asserted high if the CTS input has been asserted low, and the transmitter holding buffer is ready for an other character from the CPU. The TxRDY signal can be connected to an interrupt input on the CPU or an 8259A, so the character to be transmitted can be sent to the 8251A on the interrupt basis. When a character is written to the 8251A data address, the TxRDY will go low and remain low until the holding buffer is again ready for another character. Putting a '1' in bit D1 of the command word will cause the DTR output of the 8252A to be asserted low. This signal is used to tell a modem that a terminal/ computer is operational. A '1' in bit D2 of the command word enables the RxRDY output pin of the 8251A. If enabled, the RxRDY pin will go high when the 8251A has a character in its receiver buffer ready to be read. This signal can be connected to an interrupt input so that character can be read in on an interrupt basis. The RxRDY output is reset when a character is read from the 8251A.

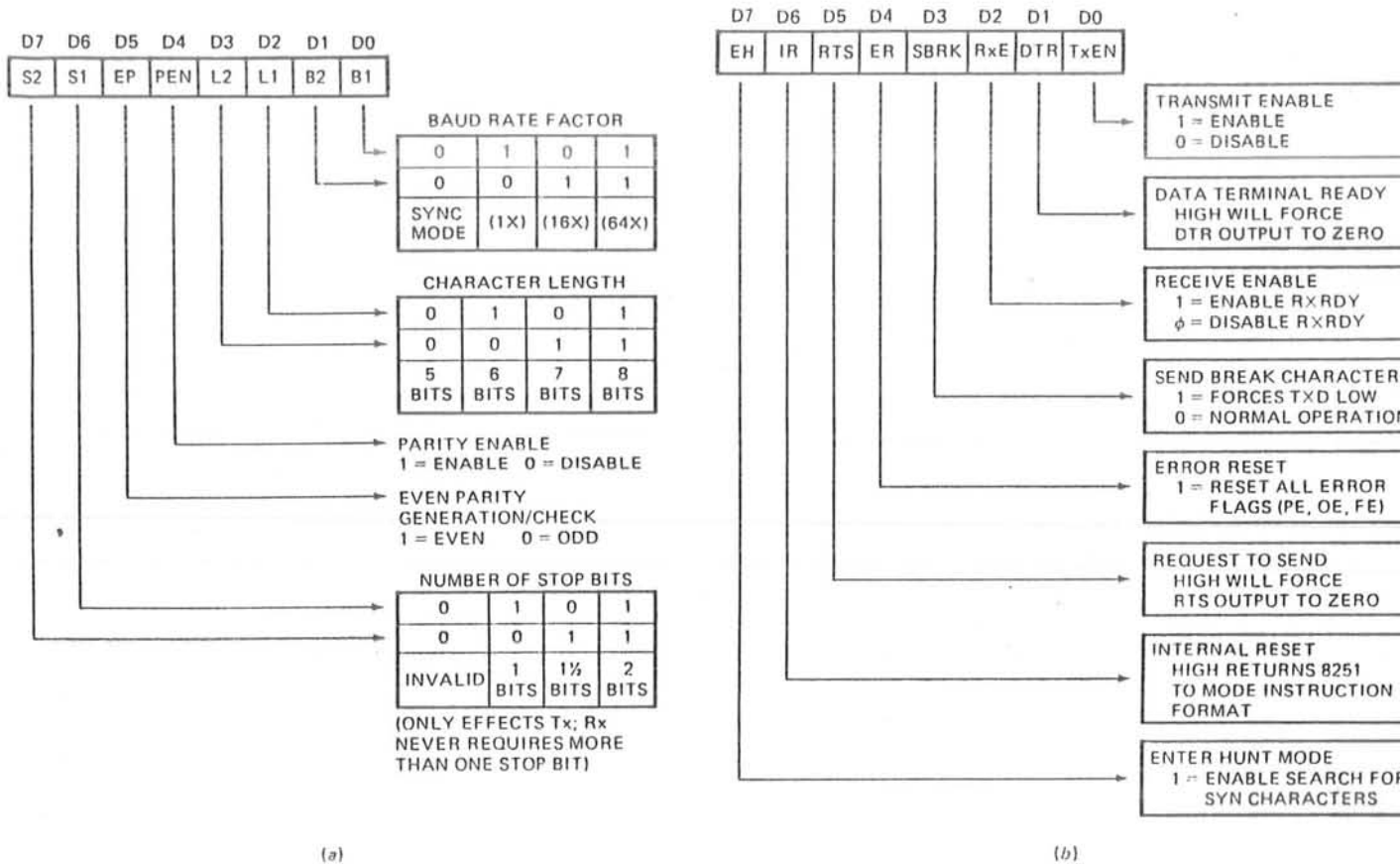


Figure 2.4

2.2.4 RS-232C SERIAL DATA STANDARD

In response to the need for signal and handshake standards between DTE (data terminal equipment) and DCE (data communication equipment), the Electronic Industries Association (EIA) developed EIA standard RS-232C. This standard describes the functions of 25 signal and hand-shake pins for serial data transfer. It also describes the voltage levels, rise and fall times, maximum bit rate, maximum capacitance for these signal lines.

RS-232C specifies 25 signal pins and it specifies that DTE connector should be a female. When wiring up these connectors, it is important to note the order in which the pins are numbered.

2.2.4.1 RS-232C SIGNAL PINS

RS-232C is a 25 pins connector but for most applications

only a few of these pins are used. The signal names, signal direction, and brief description for each of these mostly used pins are given below.

PIN NUMBER	COMMON NAME	RS-232C NAME	DESCRIPTION	SIGNAL DIRECTION ON DCE
2	TXD	BA	Transmit data	IN
3	RXD	BB	Receive data	OUT
4	RTS	CA	Request to send	IN
5	CTS	CB	Clear to send	OUT
6	DSR	CC	Data set ready	OUT
7	GND	AB	Signal ground	-
8	CD	CF	Received line signal detector	OUT
20	DTR	CD	Data terminal ready	IN

Note that the signal direction is specified with respect to the DCE. This convention is a part of the standard. The signal direction is very helpful when drawing circuits for connecting RS-232C equipment.

2.5 CTCNET PROTOCOL (Serial Communication Software)

The CTCNET protocol was defined for the exchange of data between computers of different origins using the standard RS232 serial line at CTC in 1982. The existing system has many versions, as described below.

2.5.1 INITIAL PROTOCOL AND EARLY IMPLEMENTATIONS

The RS 232 standard defines a number of hardware handshake lines in addition to the three bare minimum send data (pin no 2),

receive data (pin no 3) and the signal ground (pin no 7) connections. It was decided to use just the three lines in the CTCNET specification in order to keep the cable costs low. It was also decided to fix the serial communication line parameters to 9600 baud, no parity, 8 data bits and one stop bit. The selection of 8 bit data word was made to allow easy transmission of extended ASCII or binary data in addition to the normal ASCII data. Since the hardware handshake was given up, it was necessary to incorporate some kind of a handshake using software. It was also considered necessary to incorporate error checking in the data packets to guarantee the integrity of the transferred data. The initial data exchange protocol defined for the CTCNET is summarized as follows:

- Sender (originator) sends control-A at regular intervals until a control-B is received
- Receiver sends control-B
- Sender sends control-C as start of data identifier
- Sender sends a data record of 128 bytes followed by a two byte checksum
- Receiver checks the received record for its integrity
- Receiver sends control-D if data received without error
- Receiver sends control-E if checksum error detected to request retransmission of the same record -Any number of data packets may be transmitted (or retransmitted upon detection of an error)
- Sender sends control-Y to signal end of data
- Receiver sends control-Z to signal end of dialogue

The CP/M-86 (for the S-500) and the ISIS-II (for the Intel

MDS) implementations of the CTCNET were carried out at CTC. The usefulness of this scheme prompted to implement CTCNET on the various other computer systems which became available at the CTC. During 1983-84, further work in this direction was carried out as a CTC Post Graduate Training Program (PGTP) project. This expanded the scope of CTCNET to cover six different hardware and four different software platforms. These are listed in Table 1.

Table 1

The Early Implementations of CTCNET

S.No.	Computer System	Processor	Operating System
1.	Systime S-500	8086	CP/M-86
2.	Intellec MDS	8085	ISIS-II
3.	Hitachi MBE-16002	8088	CP/M-86
4.	ai-M16	8086	CP/M-86
5.	PDP-11/23	LSI-11	RSX-11M
6.	CROMEMCO Z-2D	Z-80	CDOS

Since the data exchange protocol is independent of the file structure of the operating system, it was thought that all types of files (i.e. source, object, executable etc.) could be exchanged and brought back to the original environment for use.

2.5.2 VMS AND DOS IMPLEMENTATIONS

By 1985, the CTC had acquired a VAX-11/730 computer running the VAX/VMS operating system along with a few IBM PC and compatible computers with the MSDOS/PCDOS operating system. Another PGTP project was, therefore, initiated to carry out the implementation of the CTCNET on the VMS and DOS operating systems. during 1985-86.

In this work, a more general solution to the problem of restoration of different types of files was worked out. A 256 byte header was transmitted with a file sent from a VMS environment. This header contained a signature identifying the file as having a VMS origin plus information which was extracted from the various access blocks using the VAX Record Management Services. This information was sufficient to recreate the file with exactly the same organization and characteristics at a later time. When such a file was received back in the VMS environment, this information could be used to recreate the file according to original specifications, thus ensuring full functionality.

The information extracted and transmitted as the 256 byte header is tabulated below in the form of the various fields and their lengths in bytes:

VMS Signature	8
File Organization	1
Record Attributes	1
Record Format	1
Maximum Record Size	2
Filename	115
Longest Record Length	2
Maximum Record Size	2
Allocation Quantity	4
Creation Date and Time	8
Expiration Date and Time	8
Backup Date and Time	8
File Protection	2

From the VMS environment, this CTCNET implementation

allowed a file to be sent out in one of three modes described hereafter. In the Formatted ASCII mode, the variable length records are read from a VMS file and carriage return and line feed characters are appended to it before it is transmitted. This ensures compatibility of the ASCII file at the other end. The Binary mode transfers files in blocks dumping bytes without regard for any file structure or organization. This mode may be used for transmitting nonstructured files containing raw binary data. It may also be used to restore a DOS file back to a PC which had earlier been received on the VAX using Binary mode. In the VMS File mode, the header information mentioned above is transmitted prior to dumping the contents of the file. This is the most powerful mode and allows all non ASCII VMS specific files to be transmitted guaranteeing full functionality on return to VMS environment.

Similarly, files can be received in the VMS environment in one of the three modes. In the Formatted ASCII mode, the received data is converted into records by locating and stripping the carriage return and line feed characters and stored in VMS record format. This mode may be used for exchanging all source and text files. In the Binary mode, the received data is stored in fixed 512 byte records in a sequential file organization. The VMS file mode uses the header information to create a file having exactly the same organization and structure as the original VMS file. This mode may be used for the exchange of all non ASCII VMS specific files. The work on the improvement of DOS and VMS implementations of CTCNET continued. In 1989, a major revision of

both the DOS and VMS implementations was carried out. An option for selecting the baud rate was added. This was necessary to prevent the frequent loss of data during sustained transmissions at 9600 baud to multi-user systems and allowed us to drop the less elegant 'slow' transmission option. Another option was incorporated to allow the selection of either COM1 or COM2 as the communication port for the transfer of data. This work was identified as CTCNET DOS Implementation Version 2.00. During the same period, the version 2.0 of the CTCNET implemented on the VMS

2.5.3 Revised CTCNET File Exchange Protocol

The CTCNET File Exchange Protocol was revised and improved in the beginning of 1990. It was reported in CTC Internal Note CTC-IN-05-90. An effort was made to get rid of all indefinite wait conditions as well as to report errors on the remote computer.

In the earlier version of the CTCNET File Exchange Protocol indefinite wait loops were executed at the time of initial handshake and also while waiting for the start of a packet or for its acknowledgment. This resulted in frequent hangup conditions if a character was missed or if the computer at one end went down or terminated the program due to an abnormal condition. The revised version specifies the time limits placed for waiting for the various conditions. These are summarized as under:

Initial handshake: 120 attempts at 1 sec intervals

Start of new packet: 60 seconds

Completion of started packet: 30 seconds

Acknowledgment of receipt of packet: 60 seconds

Version 2.10 of the CTCNET DOS and VMS implementations was

developed by using the revised protocol.

2.6 EVALUATION OF CTCNET

Many problems are faced when files are exchanged by using the version 2.0 of the CTCNET.

- i) Indefinite loops are executed at the time of initial handshake and also while waiting for the start of packet.
- ii) Messages are not displayed on the local computer and remote computer .
- iii) The CTCNET not exchange files from SCO XENIX and AIX .
- iv) No proper way of exchanging Non-text files .
- v) If the Non-text files are transferred from one environment to the other and brought back to the original environment no guarantee is provided that the file has its original characteristics and functionality.

CHAPTER 3

PROPOSED SYSTEM

This chapter describes some proposed extensions to the existing system CTCNET. The proposed system was designed after making a comprehensive study of the existing system. Major objectives of the proposed system are

- * To remove all disadvantages of the revised version of CTCNET File Exchange Protocol implemented on DOS.
- * Implementation of the new revised version of CTCNET on SCO XENIX and AIX.
- * The revised version of the CTCNET File EXchange Protocol implemented in such a way that it is highly portable.
- * Transfer of non text files in such a manner that the original characteristics and functionality of the file is maintained, when the file is transferred back to the original system.

3.1 IMPLEMENTATION OF THE REVISED CTCNET

FILE EXCHANGE PROTOCOL

The earlier implementation of the CTCNET software has drawbacks, the indefinite wait loops were executed at the time of initial handshake and also while waiting for the start of a packet or for its acknowledgment. This resulted in frequent hangup conditions, if a character was missed or if the computer at one end went down on the program is terminated due to abnormal condition. Errors are not reported on the local and remote computer for the user facility.

In the implementation of the revised CTCNET File Exchange Protocol an effort is made to get rid of all indefinite wait conditions. For the user facility the errors are reported on the local and remote computers.

3.2 IMPLEMENTATION OF CTCNET ON SCO XENIX AND AIX

The CTCNET was designed for transferring data between computers of different origin. The usefulness of the CTCNET software continued to induce its development and evaluation. The acquisition of the SCO XENIX operating system by the CTC necessitated the implementation of CTCNET for this system. As the diskette formatted on SCO XENIX 286 is not operatable on SCO XENIX 386 and vice versa. It is necessary to implement CTCNET on SCO XENIX. The University Grants Commission also acquired an IBM RISC/6000 computer with the AIX operating system. It was desirable to implement the CTCNET software on this system as well. Implementation of the CTCNET on SCO XENIX and AIX provide that all the text files exchanged from AIX, SCO XENIX and MSDOS/PCDOS are editable in any environment.

3.3 PORTABILITY STANDARD

The version 2.0 of the CTCNET was written in assembly language which is system dependent. CTCNET was developed to exchange files between computers of different origins using serial ports. For this it is proposed to write the CTCNET code in such a way that provide much more portability, to implement it on the other system very miner changes are required.

3.4 FILE TRANSFER

Text files are transferred from one operating environment to the other for saving the time of editing. Non text files are transferred from one operating environment to the other mainly for backup, media interchange and transportation purposes. These files or not usable on the other environments. For this non text

files are transferred as packaged files i.e. the necessary information about the file and the system are stored in a data structure given the name header, that header is written on the new file then the contents of the file are written, to make the new file as packaged file. To make the file usable it is necessary that the file has its original characteristics and functionality. It is also necessary the file is not unpackaged if the packaging and unpackaging environments does not cope.

CHAPTER 4

SYSTEM DESIGN AND

DEVELOPMENT

The review of the previous system led to the conclusion that the disadvantages of the existing system are becoming an unavoidable, hurdle in the way of making it more reliable, portable, accurate and efficient.

4.1 SOFTWARE SELECTION

Software selection is a very important step in system development. The changes that were suggested in the previous system required the following facilities to be provided by the software used.

- i) Capability of accessing absolute data locations in the memory.
- ii) Providing an access to the routines and code included.
- iii) Facility to access the time from the timer clock.
- iv) Facility to access the serial ports and manipulate them.
- v) The code provide high portability.

A number of languages are available and the most suitable one is selected after considering all the aspects of the problem. Since during the design of the new system, we are mainly concerned of using low level I/O routines to provide portability and efficiency.

Two options were left on the list after considering the above mentioned facilities that were to be provided by the compilers of the language to be used

- i) C language
- ii) Assembly language

Since SCO XENIX and AIX are UNIX like operating systems. The C language provide very strong features on UNIX like operating system. CTCNET was developed for exchanging files

between different origins. Implementation of CTCNET in C language provides high portability and with out any noticeable alterations the CTCNET software can be implemented on other systems.

Eventually, the decision was to be taken about using the C language.

4.2 DATA STRUCTURES

The CTCNET software uses different kinds of data structures depending upon the needs of various parts of the software. The header data structures for the MSDOS/PCDOS, SCO XENIX286 SCO XENIX 386 and AIX are shown in appendices B-1, B-2, B-3, B-4 respectively.

4.3 SERIAL PORT SELECTION

All utilities, application and data in the XENIX system are stored as files in the file structure. The files that represent physical devices are called physical files.

To a XENIX user, a device usually appears to act like a 'file'. A file consists of an ordered sequence of bytes. File that contain data are called 'regular files' and the files that represent physical devices are called 'special files'. Each special file has at least one name that represents the physical device. Before the user can request for I/O, the user must select the corresponding file name of the serial port and must have opened that special file.

Under SCO XENIX the tty1[a--h], tty1[A--H], tty2[a--h] and tty2[A--H] files provide access to the standard and optional serial ports (with or with out modem control), tty1a and tty1A both refer to Com1 where as tty2a and tty2A both refer to Com2.

- lower case letters indicate no modem control.
- upper case letters indicate the line has modem control.

Under AIX the tty1 and tty2 files provide access to the standard serial ports, where tty1 refer as com1 and tty2 as com2. Open the file corresponding to serial port by the open() system call and input and output is performed by the read() and write() system call. When a process write to a special file by the write system call the data written is sent to the associated physical device. Likewise when a process reads from a special file by the read() system call, the data is read from the associated device and returned to the process. 'Special files' are not real files, but instead they are pointer to the device drivers.

4.4 BAUD RATE SELECTION

All asynchronous communications ports use the same general interface, no matter what type of hardware is involved. When a terminal file is opened, it normally causes the process to wait until a connection is established. The ioctl() system call on SCO XENIX and AIX is used to control the terminal.

Syntax of the ioctl() system call is

```
ioctl(fields, command , arg)
```

Where fields is a file handle returned by the open() system call corresponding to the special file name entered by the user. Arg is a pointer to the termio (data structure used to control the terminals) data structure defined below. The command argument is

TCGETA Gets the parameters associated with the terminal and stores them in the structure referenced by arg.

TCSETA Sets the parameter associated with the terminal from the structure referenced by arg. The change is immediate.

Data structure used to control the terminals is

```
    NCC  8
    unsigned short  c_iflag
    unsigned short  c_oflag
    unsigned short  c_cflag
    unsigned short  c_lflag
    char            c_line
    unsigned char   c_cc[NCC]
```

The special control characters are defined by the array `c_cc`. The relative positions and initial values for each function are as follows.

0	VINTR	DEL
1	VQUIT	FS
2	VERASE	Ctrl-H
3	VKILL	Ctrl-U
4	VFOF/VMIN	EOT
5	VEOL/VTIME	NUL
6	Reserved	
7	SWTCH	NUL

The `c_iflag` field describes the basic terminal input control, `c_oflag` field specifies the system treatment of the output, `c_cflag` field describes the hardware control of the terminal, `c_lflag` field of the data structure is used by the line discipline to control terminal functions. The necessary field for setting the baud rate is `c_cflag`. As described the field `c_cflag` is used for the hardware control of the terminal. The bit

structure for setting the baud rate is

CBAUD	BITS	BAUD RATE
B0	0000000	Hang up
B50	0000001	50 baud
B75	0000002	75 baud
B110	0000003	110 baud
B134	0000004	134.5 baud
B150	0000005	150 baud
B200	0000006	200 baud
B300	0000007	300 baud
B600	0000010	600 baud
B1200	0000011	1200 baud
B1800	0000012	1800 baud
B2400	0000013	2400 baud
B4800	0000014	4800 baud
B9600	0000015	9600 baud

According to the baud rate selected these corresponding bits are set to position by using the `ioctl()` system call, e.g. first get the parameters of the terminal by using the command `TCGETA` in the `ioctl()` system call. Then set the baud rate with the help of OR operation, after making the bits on corresponding to the selected baud rate, use the command argument `TCSETA` in the `ioctl()` system call.

4.5 READING FROM THE TERMINAL CONNECTED TO THE SERIAL PORT

Normally, terminal input is processed in units of lines. This means that a program attempting to read will be suspended until an entire line has been entered. The `c_lflag` field of the

data structure defined in section 4.4 is used by the line discipline to control the terminal functions. From these line discipline functions if the function ICANON is set, canonical processing is enabled . This enables the erase and kill edit functions, and the assembly of input characters into lines delimited by EOF (end of file) and EOL (end of line). If ICANON is not set, read requests are satisfied directly from the input queue. When the communication port through which the terminal is attached is disabled then ICANON is not set but -ICANON (disable cnonical input) is set, read requests are satisfied directly from the input queue and not satisfied until at least VMIN (the minimum character for the read request satisfication) has expired. Default VMIN and VTIME values are 4 and 5 respectively. The VMIN and VTIME values stored in the position for the EOF and EOL characters. Since the transferring files with CTCNET it is designed the serial ports to which terminals are attached are disabled. To change the default values of VMIN, assign the new value at position 4 of the array c_cc, the c_cc is field of the data structure described in section 4.4 In the implementation of CTCNET on SCO XENIX and AIX the value of the VMIN is 1 and the value of VTIME is set according to the need of the protocol.

4.6 TRANSMISSION OF ASCII FILES

ASCII files are transferred from one environment to an other to save the time of editing, which is very tedious, time consuming and erroneous job. In one operating system file organization is different from the other operating system. For this the ASCII files transferred from one operating environment to the other remain editable to the other i.e. organized in the

same format as the receiving operating required.

4.6.1 TRANSMISSION OF ASCII FILES FROM SCO-XENIX TO MS-DOS

Under SCO-XENIX all files are accessed as stream of bytes, no other mode is provided for the ASCII files. Files consist of records of variable length and each record ends with the line-feed (ASCII LF) character. The CTCNET software simply transfer 128 bytes of data to the receiving computer. It is possible that these 128 bytes may contain (LF) character. On the receiving side i.e. MS-DOS CTCNET receive file in TEXT mode. Under MS-DOS each record ends with the ASCII CR & LF. If the file is opened in the TEXT mode and write operation is performed on that file then ever LF character is written as a pair of CR & LF in the same way as the DOS organize the file. For this transmission of ASCII files from SCO XENIX to MSDOS/PCDOS, simply 128 bytes of the files are transferred and MSDOS/PCDOS receives on the other side.

4.6.2 TRANSMISSION OF ASCII FILES FROM SCO XENIX TO AIX

AIX is UNIX like operating system. All files are accessed under AIX as stream of bytes and each record ends with the ASCII LF character. AIX and SCO XENIX organizes files in the same way. In two cases data is simply data read and transferred.

4.6.3 TRANSMISSION OF ASCII FILES FROM MSDOS/PCDOS TO SCO XENIX AND AIX

MSDOS/PCDOS files can be accessed in binary as well as text mode. In binary mode all the files are accessed but in text mode only ASCII files are accessed. If files are accessed in text mode each request reads the specified bytes and convert the ASCII LF

and CR into LF character e.g. if 128 bytes on the file has four LF and CR when these 128 bytes are read from the file the read request return 124 bytes and each LF and CR character is converted into LF only. SCO XENIX and AIX file organization requires only LF character at the end of each record. In MSDOS/PCDOS operating system data is read from the file in text mode, then each CR is extracted, so the data read from file is transferred. SCO XENIX and AIX write the data on the file as received.

4.6.4 TRANSMISSION OF NON TEXT FILES

Non text files are transferred from operating environment to the other mainly for backup, media interchange, and transportation purposes but not to be used there. CTCNET provide a special mode i.e. 'Transparent mode ' for the transfer of non text files. For the transmission of non text files it does not matter how operating system store the files, the files are simply accessed in binary mode and transmitted and received. The non text files transferred from one environment to the other and vice versa, it is necessary that the file has original characteristics and functionality otherwise the file will not be usable in the original environment. For this the packaging and unpackaging software is developed which is described in section 4.8.

4.7 DATA TRANSMISSION IN CASE OF SEND/RECEIVE

Hardware handshake has been described in chapter 2. It is necessary to incorporate some kind of handshake using software. It is necessary to incorporate error checking in the data packets transferred to guarantee the integrity of the data transferred.

4.7.1 SOFTWARE HANDSHAKE

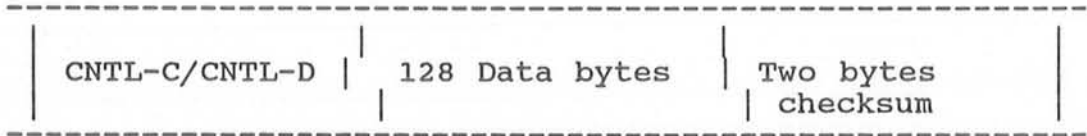
Before the data transmission between two computers take place the two computers exchange single-byte message. The sending computer send (CNTL-A) 'Ready to send' every 1 second upto 120 times and check for the (CNTL-B) 'Ready to receive' every second.

Where as the receiving computer wait for (CNTL-A) upto 120 seconds, if it received the (CNTL-A) then the receiving computer send (CNTL-B) 'Ready to receive' message. If the receiving computer receive CNTL-A within 120 seconds and sending computer receive CNTL-B within 120. This ensures that the two computers are connected properly and data can be exchanged. Otherwise ' No response from remote ' message is displayed on the local computer. The handshake is performed only once i.e. before the transmission of the file starts.

4.7.2 INTEGRITY IN DATA TRANSMISSION

To guarantee the integrity of the data transfer, data bytes are sandwich between single byte message (CNTL-C) 'Beginning of data buffer' and two byte checksum, this form 131 bytes long packet. This packet is then transmitted to the receiving computer and wait for reply for 60 seconds. The two byte checksum is calculated by summing all 128 data bytes. The packet is received by the receiving computer the sum of all 128 bytes received is again calculated by the receiving computer. The two sums, received and calculated are compared. When the two sums are the same then the transmitting computer is acknowledged by send (CNTL-D) 'Data buffer received ok' and negatively acknowledged by sending (CNTL-E) 'Retransmit data buffer' if the

two sums are not the same. The previously sent packet was again sent if the packet is negatively acknowledged by the receiving computer. New packet is sent to the receiving computer if the transmitting computer is positively acknowledged by the receiving computer. The checksum provides integrity for safe transfer of the 128 data bytes. The packet layout is given below.



The complete send and receive logic is shown in the APPENDIX A

4.7.3 WORKING OF SEND

CTCNET transfers all kinds of files text/non-text. SCO XENIX and AIX has only binary mode to access the files i.e. all files are transferred as binary files. MSDOS/PSDOS has text and binary modes for accessing files. Under MSDOS/PCDOS CTCNET transfer text files in text mode and non text files in binary mode.

- i) The file to be sent is opened for access by the open() system call. If file open operation is not successful then the message 'Error opening file' is displayed on the local computer and the user is asked to 'Enter the option' .
- ii) The message ' Ready to send ' (CNTL-A) is sent to the receiving computer and it waits for the reply of receiving computer for 1 second, if the reply of the receiving computer is not received within 1 second ,the message 'Ready to send' (CNTL-A) is again sent and it waits for the reply for 1 second. This

process of sending the message 'Ready to send' (CNTL-A) and waiting for the reply is repeated upto 120 seconds. If the sending computer not received the message 'Ready to receive' (CNTL-B) within 120 seconds, the message 'No response from the remote' is displayed on the local computer and user is asked to 'Press return to continue.

iii) If handshake is performed successfully then data is read from the file by using system call read(). If read operation is not successful the message 'Error reading file 'is displayed and CNTL-L is sent to the remote computer. File sending processes is terminated. User is asked to 'Press enter to continue. After each read operation the check to detect the end of file is also made.

iv) If the read operation is successful and end of file is not encountered, the message ' beginning of data buffer' (CNTL-C) is sent then 128 bytes of data and then 2 bytes checksum, first high byte and then low byte to form a packet of 131 bytes long are sent. After sending the packet the local computer waits up to 60 seconds for the reply of the receiving computer.

v) If no reply within sixty seconds then 'Remote time out' message is displayed on the local computer. If the reply of the receiving computer is CNTL-D 'Data buffer received ok' then packet received correctly and next packet is sent. If the reply is CNTL-E 'Retransmit data buffer' message is displayed, previously sent packet is again sent. If the reply is CNTL-L 'File system error on remote' is displayed and the user is asked to press any key to continue, after pressing the any key the option selection menu is displayed. If reply is any other character then 'Protocol error' is displayed and user is asked to

press any key to continue.

vi) If end of file was detected then the message 'End of file' CNTL-Y is transmitted to the receiving computer and is waited for the reply of receiving computer upto 60 seconds. If no reply till message 'Remote time out' is displayed on the local computer and the user go to the option selection menu. If CNTL-Z 'File received ok' message is received ,the file opened is closed and 'File transmitted' message is displayed on the local computer. If CNTL-L is received then 'File system error on remote' is displayed. If any other character is received then 'Protocol error ' is displayed and the user go to the option selection menu.

4.7.4 WORKING OF RECEIVE

i) A new file having the name which the user has entered, is created. If the file already exists then the contents of the file are truncated. If this file make operation is not successful then the message 'Error creating file is displayed and user go to the option selection menu

ii) After file creation the computer waits upto 120 seconds for the message 'Ready to send' (CNTL-A) of the transmitting computer. If the message CNTL-A is received within 120 seconds the receiving computer sends (CNTL-B) 'Ready to receive' message to the transmitting computer otherwise the message 'No response from remote' is displayed and the user is asked to 'Press enter to continue.

iii) The computer then waits for the character up to 60 seconds. If the receiving computer does not find any character within 60

second, then the message 'Remote time-out' is displayed and user is asked to 'Press enter to continue' ,after pressing the enter key the user go to the option solution menu.

iv) If the computer r receive 'Beginning of data buffer' (CNTL-C) it waits upto 30 seconds for 130 characters, 128 data bytes and two bytes checksum with the higher byte first and lower byte follows it. If 130 characters are not received within 30 seconds ,the 'Packet time-out' message is displayed on the receiving computer and 'Retransmit data buffer' (CNTL-E) message is sent to the transmitting computer. At the same time when the computer is receiving the data bytes it also calculates the checksum by adding the ASCII value of each byte, the calculated checksum is compared with the received checksum for the integrity of the data buffer received. If the two checksum are not equal then the receiving computer sends 'Retransmit data buffer' (CNTL-E) and the message 'checksum error' is displayed otherwise the message 'Data buffer received ok' (CNTL-D) is acknowledged.

v) After integrity the received data is written to the file if the write operation is not successful the message 'error writing file' is displayed and CNTL-L is sent to the transmitting computer, and the user is asked to 'Enter the option'.

vi) This data receiving process is repeated until (CNTL-Y) 'End of file ' message received. After receiving (CNTL-Y) the created file is closed. If close file operation is not successful then 'Error in closing file' is displayed and (CNTL-L) is sent to the transmitting computer user is asked 'Enter the option' otherwise (CNTL-Z) 'File received ok' message is sent to the transmitting computer and user is asked to 'Enter the option'.

vii) If the character received is CNTL-L then 'File system error on remote' is displayed . If any other character i.e. not CNTL-C, not CNTL-Y and not CNTL-L then 'Protocol error' is displayed and user is asked to 'Enter the option'.

4.8 FILE PACKAGING/UNPACKAGING

The file packaging/unpackaging part is separate from the file transfer part. The file transfer part would be almost identical for all systems The file packaging/unpackaging part is operating system dependent. The non-text files (such as object, executable, and other files) are transferred from one operating environment to another mainly for backup, media interchange or transportation purposes and not to be used there. However upon return to the original environment the (packaging/unpackaging) software must guarantee that the files are restored with its original characteristics and functionality. The packaging of a file is to package the contents of a file along with a header containing all its essential attributes and characteristics into another file. If the packaged file is transferred from environment SCO XENIX to AIX to MSDOS/PCDOS and then brought back to SCO XENIX. In the environment AIX and MSDOS/PCDOS the file is not usable and cannot be unpackaged , but on environment SCO XENIX the unpackaging program would read the header and restore the file with its original characteristics and functionality. The header formats for DOS, SCO XENIX 286, SCO XENIX 386 and AIX are shown in appendices B-1,B-2,B-3,B-4 respectively. The packaging/unpackaging software first of all open the file "Sysi" (system identification). If the file is not opened successfully

then the message 'file open error or file not found' is displayed. The file 'Sysi' contains system identification and the name of the operating systems. At present the 'Sysi' file contains the information about the four systems. These all listed in the table 4.1 given below.

TABEL 4.1

Operating system Identifications

Identification	operating systems
MSPCDOS	DOS (MSDOS/PCDOS)
UNIXSV01	SCO XENIX 286
UNIXSV02	SCO XENIX 386
UNIXSV20	AIX

In the 'Sysi' file you can easily append information about the other systems.

4.8.1 WORKING OF FILE PACKAGING

i) The file to be packaged is opened by the open() system call, if the file is not opened successfully the message 'Error opening file' is displayed and the user is asked to 'Enter the option'

ii) The first 128 bytes of the file are read and compared with the first 8 bytes with the system identification of the 'Sysi' file. If any identification from 'Sysi' file matches with the first 8 bytes of the file read the message 'File is already packaged' is displayed and the user is asked to enter the option, otherwise the new file name is made by original name of the file with out extension by appending '.pk' to every packaged file.

iii) The contents of the header are obtained by using the system calls as described in the appendix B and the header is written on

the new file ,then the contents of the file are read and written to the new file until the end of file is reached.

iv) Both files are closed by using the close() system call. If the close operation is successful then the message 'File +is successfully packaged' is displayed and the package file name along with the new file name is also displayed, other wise the 'File is not packaged successfully' is displayed and the user is asked to 'Enter the option'.

4.8.1 WORKING OF UNPACKAGING

i) The file to be unpackaged is opened by the OPEN() system call. if the file OPEN() operation is not successful 'FILE OPEN ERROR' and the user is asked to enter the option.

ii) The first 128 bytes of the file are read. From these first 8 bytes are compared with the system identification. If comparison performed is successful then 6 bytes after 8 bytes are compared with the system version. If the packaged version and the system version are same then the original characteristics and functionality of the file are maintained. From the fields stored in the header, header is restored from the file and data is written on the new file. The new file name is made by removing the extension '.pk' and the message 'File successfully unpackaged' is displayed on the screen.

iii) If the system version and packaged version are not same then the system warns the user that the packaged and unpackaged version are different and allows the user to unpackage the file with the message 'Do you want to unpackage the file Y/N'.

iv) The file is unpackaged if user presses Y. Upon pressing the

N the first 8 bytes of the file are compared with the system identification of the 'Sysi' file. If any comparison is performed, all necessary informations about the file and the operating system are displayed on the screen. Otherwise the 'File is not a packaged' is displayed on the screen and the user is asked to 'Enter the option'.

CHAPTER 5

SYSTEM USER

INTERFACE

The purpose of this chapter is to explain to the user the working of CTCNET software and packaging/Unpackaging software and to make use of it in the best possible way.

5.1 HOW TO INVOKE THE CTCNET

To invoke the CTCNET type the word CTCNET at operating system prompt then press the <return> key. The operating system will load and execute the CTCNET and on the screen the message shown in figure 5.1a, 5.1b and 5.1c appear. This message asked the user to enter the name of the serial communication port. If the name of the serial communication port is not correct. Then the message shown in figure 5.2 will be displayed and the user is again asked to enter the name of the serial port.

After entering the correct communication port the user is asked to select the baud rate by the selection menu shown in Figure 5.3. If the correct baud rate is not specified the menu shown in the figure 5.4 is displayed and the user is again asked to enter the baud rate. When correct baud rate is selected the menu shown in the Figure 5.5 is displayed. If the user selects the option '1' (send file) or '2' (receive file) then the menu shown in figure 5.6 is displayed.

5.1.1 SEND A FILE

Press '1' (file send) when the menu shown in the Figure 5.5 is displayed. After selection the file send option Figure 5.6 is displayed and user is asked to select the mode. If the user wants to send the TEXT file press mode 0, otherwise press 1 to transfer Non-text files. If the user does not enter '0' or '1' then the figure 5.7 is displayed. When the user enters the correct mode, the message 'File to be sent' is displayed. The

user enters the name of the file, if the two computers connected properly and the receiving computer is ready to receive then the transmission of the file will start. The message ' File transmitted ' is displayed, if the file has been transmitted correctly and the user is asked to 'Press enter to continue. After pressing the enter key the menu shown in figure 5.5 is displayed. If the user wants to transmit another file, then the process mentioned above is repeated.

It may also happen that during the transmission of the file the message 'Error reading file' is displayed try again to send the file repeating the above procedure.

5.1.2 RECEIVE FILE

Press '2' when the menu shown in Figure 5.5 is displayed . The procedure described in the section 5.1.1 is repeated but the message 'File to be sent' is displayed . Sometimes it may also happen that when you enter the file name the message 'Error opening file or file not created' is displayed. Sometimes the message 'Error writing file' is displayed. In these cases the user repeat the process mentioned above. If the file has been received the message 'File received' will be displayed. If the user desired to receive another file, repeat the steps described above.

5.2 USING FILE PACKAGING/UNPACKAGING

Enter the command 'PDACK' at the operating system prompt and press the <RETURN> key . If the executable file 'PDACK' is present the operating system will load and execute it. The menus shown in Figure 5.8a, 5.8b and 5.8c are displayed.

5.2.1 EXAMINE FILE

Press '1' to see the file is packaged or not . After pressing '1' the message 'Enter the file' is displayed. When the user enters the file name and press the <RETURN> key, the file is packaged and the Figure 5.9 is displayed on the screen, other wise the message 'The file is not a packaged file' and the user is asked to ' Press enter to continue. After pressing the <ENTER> key the menu shown in figure 5.8a or 5.8b or 5.8c is displayed.

5.2.2 PACKAGE FILE

Press '2' to package the unpackaged file. After pressing 2 the message 'Enter the file' is displayed. When the user enters the file name and press the <RETURN> key. The message 'File successfully packaged' and 'The packaged file name ' is displayed on the screen.

5.2.3 UNPACKAGE FILE

Press '3' to unpackage the package file. After pressing 3 the message 'Enter the file' is displayed. When the user enters the file name and press the <RETURN> key. The packaging/unpackaging software sees whether the file is packaged on the operating system upon which the user trying to unpackage. If the file is packaged on the operating system upon which unpackaging is made then the message 'File successfully unpackaged displayed, otherwise the Figure 5.10 is displayed on the screen. The user is asked to 'Press enter to continue' after pressing return key, the menu shown in figure 5.8 is displayed. The process is repeated if the user wants to package or unpackaging any other file. Enter 0 to go to the operating system prompt.

CTCNET File Exchange Utility

SCD XENIX Implementation Version 3.00

Select serial device to be used

The valid choices are tty1a and tty2a

Enter name of serial device:

Figure 5.1 a

CTCNET File Exchange Utility

AIX Implementation Version 3.00

Select serial device to be used

The valid choices are tty1 and tty2

Enter name of serial device:

Figure 5.1 b

CTCNET File Exchange Utility

PCDOS/MSDOS Implementation Version 3.00

Select serial device to be used

The valid choices are COM1 and COM2

Enter name of serial device:

Figure 5.1 c

Invalid serial device name

The valid choices are COM1 and COM2

Enter name of serial device:

Figure 5.2

Select baud rate

The valid choices are

9600, 4800, 2400, 1200,

600, 300 and 150

Enter baud rate:

Figure 5.3

Invalid baud rate specified

The valid choices are

9600, 4800, 2400, 1200,

600, 300 and 150

Enter baud rate:

Figure 5.4

Select one of the following options:

0 Exit CTCNET

1 Send a file

2 Receive a file

Enter the option [0/1/2]:

Figure 5.5

Select one of the following modes of transmission

0 Formatted ASCII

1 Transparent

Enter the mode [0 / 1] :

Figure 5.6

Invalid mode

Select one of the following modes of transmission

0 Formatted ASCII

1 Transparent

Enter the mode [0 / 1] :

Figure 5.7

Select one of the following options

0 Exit to XENIX

1 Examine file

2 Package file

3 Unpackage file

Enter the option [0/1/2/3]:

Figure 5.8 a

Select one of the following options

- 0 Exit to AIX
- 1 Examine file
- 2 Package file
- 3 Unpackage file

Enter the option [0/1/2/3]:

Figure 5.8 b

Select one of the following options

- 0 Exit to DOS
- 1 Examine file
- 2 Package file
- 3 Unpackage file

Enter the option [0/1/2/3]:

Figure 5.8 c

The file is packaged

The information about the file follows

System	DOS
Version	4.0
File name	timer.asm
File size	1530
Package date & time	27-2-1991 0:55
Creation date & time	21-1-1991 1:14

Press enter to continue

Figure 5.9

This file cannot be unpackaged on DOS

The information about the file follows

System	DOS
Version	4.0
File name	timer.asm
File size	1530
Package date & time	27-2-1991 0:55
Creation date & time	21-1-1991 1:14
Press enter to continue	

Figure 5.10

CHAPTER 6

SYSTEM CONCLUSION

6.1 SOME MAJOR FEATURES

In this chapter, the implementation of the revised version of the CTCNET and PACKAGING/UNPACKAGING described in this dissertation is evaluated. As was stated earlier in chapter one, the objective of this work was to implement latest version of CTCNET on SCO XENIX and AIX for efficient and reliable file exchange and provide solid solution for exchanging non text files. Some features of the system are given below.

i) Reliability

For a multitude of reasons, a program that works under normal circumstances may fail mysteriously at an unexpected time. An unreliable program is obviously frustrating the user, especially when the cause of the program failure is unknown. The implementation of revised version of CTCNET specifies the time limits i.e. after certain interval of time the user is responded, if the computer at one end went on down or the program is terminated due to abnormal conditions. This system has high degree of reliability.

ii) PORTABILITY

Portability means the program which is developed on one system can easily run, without any noticeable alteration, on other systems. The new system was implemented on SCO XENIX and AIX, thus it can be easily implemented on other UNIX like operating systems with minor changes in the CTCNET code.

iii) ACCURACY

Accuracy is the ratio of correct information to the total amount of information produced over a period. In this system the accuracy is hundred percent, i.e. unless the data transferred is

not received correctly. Unpackaged file has its original characteristic and functionality. The new system is very accurate. Validation checks are made, to ensure the accuracy of the system.

iv) **SUITABILITY**

This system is suitable for exchanging data between AIX, SCO XENIX and MSDOS/PCDOS, also it provides a suitable solution for transferring non text files.

v) **CONSISTENCY**

The new system has a high capability of consistent working. If the exchange operation is not proper the error is displayed.

vi) **FLEXIBILITY**

There is always a room for improvement. The new system was designed by using interactive modular approach and is highly flexible to cater further improvements in future.

vii) **EFFICIENCY**

The code is optimized to the maximum level. It is as efficient as it could be made. Actually the indefinite wait loops are replaced with the time limits. The program does not result in frequent hang-up conditions if any abnormal condition occurs.

viii) **EASY TO USE**

The new system is user friendly and users find no difficulty in getting required results.

6.2 REMEDIES

Although the system provides many useful features it lacks in several aspects. Some of these are as follows.

* The packaging/unpackaging software does not work if the 'sysi'

file is not present on the system.

- * The CTCNET software takes more time generating errors in the case of error conditions if the clock timer does not work properly.
- * The CTCNET transfer files within the limited area.
- * CTCNET software requires operator interaction at both ends, i.e. at sending and receiving ends.

6.3 FUTURE EXTENSIONS

The work on this project can be continued in future and some other features may be added. The probable extensions in the system are.

- * It is also possible to extend the system for transporting mail.
- * Operator interaction at both ends can be removed by fixing the role of the sender and receiver, also transfer of files over long distance using the modems and telephone lines.
- * CTCNET should be implemented on the every coming operating system.
- * Implementation of packaging/unpackaging software on VMS and other operating systems.
- * Develop software which will run in background and service file transfer requests using predefined connections. Some work in this direction has been initiated at CTC as a PGTP project.

APPENDIX A

Enhanced CTCNET File Exchange Protocol

The 'send' refers to sending to the remote computer on the serial line. The messages enclosed in single quotes are to be displayed on the local computer.

Sender Logic

Open file

```

if open error then
    'Error opening file'
quit

```

Handshake

```

send control-A every 1 second upto 120 times
check for control-B every second
    if received then
        continue
    else
        'No response from remote'
quit

```

Read and send records

```

if read error then
    'Error reading file'
    send control-L
    quit
send control-C, then record and then checksum
wait for reply for 60 seconds
    if no reply then
        'Remote timeout'
        quit
    else
        on control-D
            continue
        on control-E
            'Retransmitting packet'
            send packet again
        on control-L
            'File system error on remote'
            quit
    any other character
        'Protocol error'
        quit

```

```
On end of file
  send control-Y
  wait for reply for 60 seconds
  if no reply then
    'Remote timeout'
    quit
  else
    on control-Z
      'File transmitted'
      quit
    on control-L
      'File system error on remote'
      quit
    any other character
      'Protocol error'
      quit
```

Receiver Logic

```
Create file
  if error then
    'Error creating file'
    quit
```

```
Handshake
  wait for control-A for 120 seconds
  if received then
    send control-B
    continue
  else
    'No response from remote'
    quit
```

```
Wait for character for 60 seconds
  if no character then
    'Remote timeout'
    quit
  else
    on control-C
      look for 130 characters in 30 seconds
      if not received then
        'Packet timeout'
        send control-E
        continue
      if checksum error then
        'Checksum error'
        send control-E
        continue
      write record
```

```
        if write error then
            'Error writing file'
            send control-L
            quit
        else
            send control-D
            continue
    on control-Y
        close file
        if close error then
            'Error closing file'
            send control-L
            quit
        else
            'File received'
            send control-Z
            quit
    on control-L
        'File system error on remote'
        quit
    any other character
        'Protocol error'
        quit
```

APPENDIX B

CTCNET FILE PACKAGING SCHEME

DOS HEADER FORMAT

Offset	Description	Size (Bytes)	Contents	System call
0	Identification	8	MSPCDOS#	
8	Version	6	4.01##	int 21 30h
14	No. of 128 bytes block in header	1	1	
15	Packaging date	5		_dos_getdate _dos_gettime
20	File date stamp	5		_dos_findfirst
25	Reserved	23		
48	Attribute	1		_dos_findfirst
49	Reserved	3		
52	File size	4		_dos_findfirst
56	Reserved	8		
64	File name	64		_dos_findfirst

The entry in the version field is an example, at the time of packaging actual version number will be stored here.

CTCNET FILE PACKAGING SCHEME

XENIX 286 HEADER FORMAT

Offset	Description	Size (Bytes)	Contents	System call
0	Identification	8	UNIXSV01	
8	Version	6	2.2.1#	uname
14	No. of 128 bytes block in header	1	2	
15	Packaging date	5		time
20	File date stamp	5		stat
25	Reserved	23		
48	Mode	2		stat
50	Reserved	2		
52	File size	4		stat
56	Reserved	8		
64	File name	64		stat
128	Creation date and time	5		stat
133	User ID	2		stat
135	Group ID	2		stat
137	Links	2		stat
139	Reserved	117		

The entry in the version field is an example, at the time of packaging actual version number will be stored here.

CTCNET FILE PACKAGING SCHEME
XENIX 386 HEADER FORMAT

Offset	Description	Size (Bytes)	Contents	System call
0	Identification	8	UNIXSV02	
8	Version	6	2.3.1#	uname
14	No. of 128 bytes block in header	1	2	
15	Packaging date	5		time
20	File date stamp	5		stat
25	Reserved	23		
48	Mode	2		stat
50	Reserved	2		
52	File size	4		stat
56	Reserved	8		
64	File name	64		stat
128	Creation date and time	5		stat
133	User ID	2		stat
135	Group ID	2		stat
137	Links	2		stat
139	Reserved	117		

The entry in the version field is an example, at the time of packaging actual version number will be stored here.

CTCNET FILE PACKAGING SCHEME

AIX HEADER FORMAT

Offset	Description	Size (Bytes)	Contents	System call
0	Identification	8	UNIXSV20	
8	Version	6	3.1###	uname
14	No. of 128 bytes block in header	1	2	
15	Packaging date	5		time
20	File date stamp	5		stat
25	Reserved	23		
48	Mode	4		stat
52	File size	4		stat
56	Reserved	8		
64	File name	64		stat
128	Creation date and time	5		stat
133	User ID	2		stat
135	Group ID	2		stat
137	Links	2		stat
139	Reserved	117		

The entry in the version field is an example, at the time of packaging actual version number will be stored here.

APPENDIX C

Special characters used in CTCNET software, ASCII codes
and interpretation.

S.NO	CHARACTER	ASCII-CODE	INTERPRETATION
1.	CNTL-A	1	READY TO SEND.
2.	CNTL-B	2	READY TO RECEIVE.
3.	CNTL-C	3	BEGINNING OF DATA BUFFER.
4.	CNTL-D	4	DATA RECEIVED OK.
5.	CNTL-E	5	RETRANSMIT DATA BUFFER.
6.	CNTL-L	12	ERROR CONDITION.
7.	CNTL-Y	25	END-OF-FILE.
8.	CNTL-Z	26	FILE RECEIVED OK.

APPENDIX D

The messages displayed during the execution of the CTCNET software.

S.NO	MESSAGES
1	File to be sent
2	File to be received
3	File received
4	File transmitted
5	No response from remote
6	Remote timeout
7	Retransmitting packet
8	Packet timeout

APPENDIX E

The error messages displayed during the execution of the CTCNET.

S.NO	ERROR MESSAGES
1	Error opening file
2	Error creating file
3	Error reading file
4	Error writing file
5	Error closing file
6	Checksum error
7	Protocol error
8	File system error on remote

BIBLIOGRAPHY

1. Turbo C, User's Guide Borland International Inc., U.S.A. 1987.
2. Turbo C, Reference Guide, Borland International Inc., U.S.A. 1987.
3. Schildt, Herbert, C Made Easy Osbrine McGraw-Hill Inc., U.S.A. 1985.
4. Microsoft C Reference Manual Microsoft Inc., USA, 1987
5. Microsoft C User's Guide Microsoft Inc., USA, 1987.
6. Royer. Jeffery P., Hand Book of Software and Hardware interfacing for IBM PC., Prentice Hall Inc., 1978.
7. Douglas V. Hall., Microprocessors and interfacing programming and hardware MCGRAW-HILL INTERNATIONAL EDITIONS.
8. Norton prter., programmer's Guide to IBM PC, Microsoft.
9. Rebecca Thomas Lawrence R. Rogers Jean L. Yates Advanced Programmers Guide to UNIX System V International Edition 1986.
10. Alan Deikman UNIX Programming on the 80286/80386 Second Edition 1989.
11. XENIX System V Development System Programmer's Reference The Santa Cruze Operation, Inc., U.S.A. 1987.
12. XENIX System V Development System Programmer's Guide The Santa Cruz Operation, Inc., U.S.A. 1987.
13. XENIX System V Development System C User's Guide, The Santa Cruz Operation, Inc., U.S.A. 1987.
14. Communication Programming Concepts AIX Version 3 For

RISC System /6000.

- 15 General Programming concepts AIX Version 3 For RISC
System /6000.
- 16 Calls and Subroutines Reference User Interface AIX
Version 3 For RISC System /6000.