Design and implementation of a simulator for a local area network utilizing an IBM PC/AT or compatible computer

Christian G. Midgley

Follow this and additional works at: https://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
Rochester Institute of Technology
College of Engineering
Department of Computer Engineering

Design and Implementation of a Simulator for a Local Area Network Utilizing an IBM PC/AT or Compatible Computer.

Christian G. Midgley, Sr.
May 21, 1988

An intern project, submitted to the Faculty of the College of Engineering in Partial fulfillment of the requirements for the Masters of Engineering Degree in Computer Engineering.

Approved by:

George A. Brown

Roy G. Czernikowski
Abstract

The purpose of this project is to provide a fixture which may be used as a local area network for a course similar to the Small Systems Workshop course (EECC756) as taught in the spring of 1986. This fixture will provide the physical layer of the local area network. This network would be Ethernet like but use an RS-232C communication between the fixture and the nodes. In addition to simulating the ether the fixture would provide for on-line analysis of the packets being sent to provide feedback to the student as well as the professor.

The IBM PC/AT type personal computer was selected as the host for the fixture since it is a basic unit which is available at the current time. In addition it provided an excellent base operating system. The software that was developed makes use of the direct system calls to the MS-DOS operating system whenever possible.
# Table of Contents

Abstract ................................. \( i \)

Table of Contents ........................ \( ii \)

List of Tables .......................... \( vii \)

List of Figures .......................... \( viii \)

1.0 Introduction .......................... 1
  1.1 Goals ................................ 1

2.0 Fixture Description ...................... 2
  2.1 Hardware Description .................. 2
    2.1.1 IBM PC/AT or Compatible Computer  3
    2.1.2 Expansion CARD ..................... 3
  2.2 Network Description ................... 12
    2.2.1 Header ............................ 12
    2.2.2 Source Address ...................... 12
    2.2.3 Destination Address ................. 12
    2.2.4 Packet Size ........................ 13
    2.2.5 Packet Type ........................ 13
    2.2.6 Data ................................ 13
    2.2.7 Check Sum .......................... 13
  2.3 Interface Specification ............... 14
# Table of Contents (cont)

2.3.1 Physical Interface ....................................................... 14
2.3.2 Data Format .............................................................. 14

3.0 Programming Environment ................................................. 18
   3.1 MS-DOS .................................................................. 18
      3.1.1 Cls .................................................................. 18
      3.1.2 Copy .................................................................. 18
      3.1.3 More .................................................................. 19
      3.1.4 Pause .................................................................. 19
      3.1.5 Type .................................................................. 19
   3.2 Microsoft C ................................................................. 19
      3.2.1 Compiler ............................................................. 19
      3.2.2 Linker ................................................................. 20
      3.2.3 Make ................................................................. 20

4.0 Test Method .................................................................. 21

5.0 Discussion of Results ......................................................... 22

6.0 Code ........................................................................... 23
   6.1 Netsim.c Main() ......................................................... 23
# Table of Contents (cont)

<table>
<thead>
<tr>
<th>Section</th>
<th>File</th>
<th>Function(s)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2</td>
<td>Init_b.c</td>
<td>Init_b()</td>
<td>35</td>
</tr>
<tr>
<td>6.3</td>
<td>Pfile.c</td>
<td>Print_file()</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Print_file()</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Screen()</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Com()</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lpt()</td>
<td>41</td>
</tr>
<tr>
<td>6.4</td>
<td>Net_sup.c</td>
<td>Process()</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Outerr()</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shut_down()</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rsam()</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Psam()</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Packet_check()</td>
<td>55</td>
</tr>
<tr>
<td>6.5</td>
<td>Menu.c</td>
<td>Menu()</td>
<td>59</td>
</tr>
<tr>
<td>6.6</td>
<td>Misc.c</td>
<td>Center()</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Getline()</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Help()</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Init()</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Getone()</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check_end()</td>
<td>66</td>
</tr>
</tbody>
</table>
### Table of Contents (cont)

6.7 Phead.c Printhead() . . . . . . . . . . . 68
6.8 Rnet.c . . . . . . . . . . . . . 70
   6.8.1 Run_net() . . . . . . . . . . . . . 70
   6.8.2 Echeck() . . . . . . . . . . . . . 76
   6.8.3 Port_reset() . . . . . . . . . . . . . 77
6.9 SetMat.c Setmat() . . . . . . . . . . . 78
6.10 S_par.c . . . . . . . . . . . . . 80
   6.10.1 Set_parms() . . . . . . . . . . . . . 80
   6.10.2 Set_sd() . . . . . . . . . . . . . 88
   6.10.3 Sort_sd() . . . . . . . . . . . . . 89
   6.10.4 Port_parms() . . . . . . . . . . . . . 89
6.11 Include.h . . . . . . . . . . . . . 96
6.12 Defines.h . . . . . . . . . . . . . 97
6.13 Proj . . . . . . . . . . . . . 99
6.14 Help Files . . . . . . . . . . . . . 101
   6.14.1 Netsim0.hlp . . . . . . . . . . . . . 101
   6.14.2 Netsim1.hlp . . . . . . . . . . . . . 102
   6.14.3 Netsim2.hlp . . . . . . . . . . . . . 103
   6.14.4 Netsim3.hlp . . . . . . . . . . . . . 104
   6.14.5 Netsim4.hlp . . . . . . . . . . . . . 105
# Table of Contents (cont)

6.14.6 Netsim5.hlp  .................................................. 106
6.14.7 Netsim8.hlp  .................................................. 107
6.14.8 Netsim9.hlp  .................................................. 108
6.14.9 Netsim10.hlp .................................................... 109
6.14.10 Netsim11.hlp .................................................. 110

Appendix A  References .................................................. A-1
             Technical Manuals .............................................. A-2

Appendix B  Glossary of Terms ........................................ B-1
# List of Tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Bill of Material</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Port Addresses</td>
<td>11</td>
</tr>
<tr>
<td>2.3</td>
<td>Data Format Configuration Selection</td>
<td>15</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Expansion Card Bus Interface & Address Decode . 4
2.2 Expansion Card RTS Port & Programmable Baud Clock . 5
2.3 Expansion Card Port Selection . . . 6
2.4 Expansion Card Modems & Output . . . 7
2.5 Expansion Card Rx Available . . . . . 8
2.6 Expansion Card Component Location . . . . 9
2.7 Packet Description . . . . . . . . . 12
2.8 Data Format . . . . . . . 16
2.9 Configuration of Network as Tested and Demonstrated . 17
1.0 Introduction

The idea for this project came from the EECC756 (Small Systems Workshop) course which I completed in the Spring of 1986. This course included a laboratory which was compound by the lack of a fixture on which to develop the network nodes. As a result I undertook this project to develop a tool to aid in the understanding of the workings of Local Area Networks. This project will integrate both hardware (the interface PWBA) and software (the code running on the IBM PC/AT or Compatible Computer).

1.1 Goals

The single goal of this project was to provide a fixture which could be used as an educational tool to provide a controlled simulation of a Local area network. This goal was achieved by utilizing a straightforward communication device to be controlled by a host which required very little expertise to operate. The communication of the nodes uses a simple and straightforward protocol (serial ascii over an RS-232C protocol). In addition a simple yet concise statement is made concerning each packet sent.
2.0 Fixture Description

The fixture would be constructed using an IBM PC-AT or Compatible Computer as the host. An expansion card and a remote connection box would provide the interface to the nodes. The expansion card will allow for programmable distances between the nodes, compilation of network activity to the terminal, printing of summary information to a printer, and control of generating errors in the data at varied rates. The DB-25 connector for each of the eight network ports is located in a connection box which is external to the host PC. This connection box is linked to the simulator PWBA via a forty wire cable which allows for a five wire interface on each of the ports. These wires are TxD, RxD, CTS, RTS, and signal ground.

The RTS line of each port is ORed with a bit of the RTS register which allows for simple software control of these signals. The baud, number of data bits, number of stop bits, parity enable and odd/even parity are programmable by the user. Each port on the card has a pair of I/O addresses which are used for command and data access. An additional output address pair is allocated for a global write of commands and data to all ports at the same time.

2.1 Hardware Description

The fixture consists of two basic parts.
1). An IBM PC/AT or Compatible Computer, and
2). An Expansion Card for item 1 together with an extended connection box.

2.1 IBM PC/AT or Compatible Computer

The computer is a standard purchased item which is common both in educational facilities as well as in industry so no further description is provided.

2.1.2 Expansion Card

The Expansion Card consists of several key items.

1). Eight Intel 8251A PROGRAMMABLE COMMUNICATION INTERFACE.
2). Eight MC145406 MODEM CHIP
3). A PXO-768 PROGRAMMABLE 768K CRYSTAL OSCILLATOR

In addition to the key items the Expansion Card contains various standard TTL components.
FIG. 2.5
## BILL OF MATERIAL

<table>
<thead>
<tr>
<th>Component</th>
<th>Device</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>74LS00</td>
<td>QUAD 2 INPUT NAND GATE</td>
</tr>
<tr>
<td>U2-3</td>
<td>74LS138</td>
<td>ONE-OF-EIGHT DECODE/DE-MULTIPLEXER</td>
</tr>
<tr>
<td>U4-6,17</td>
<td>74LS32</td>
<td>QUAD 2 INPUT OR GATE</td>
</tr>
<tr>
<td>U7,16</td>
<td>74LS374</td>
<td>OCTAL D FLIP-FLOP, 3-STATE</td>
</tr>
<tr>
<td>U8-15</td>
<td>MC145406</td>
<td>MODEM CHIP</td>
</tr>
<tr>
<td>U18-19,22</td>
<td>74LS244</td>
<td>OCTAL BUS LINE DRIVER, 3-STATE</td>
</tr>
<tr>
<td>U20</td>
<td>74LS245</td>
<td>OCTAL BUS TRANSCEIVER, 3-STATE</td>
</tr>
<tr>
<td>U21</td>
<td>PXO-768</td>
<td>PROGRAMMABLE 768K CRYSTAL OSCILLATOR</td>
</tr>
<tr>
<td>U23-24</td>
<td>74LS08</td>
<td>QUAD 2 INPUT AND GATE</td>
</tr>
<tr>
<td>P1-8</td>
<td>8251A</td>
<td>PROGRAMMABLE COMMUNICATION</td>
</tr>
</tbody>
</table>

 INTERFACE

Table 2.1
## ADDRESS MAP

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30FH</td>
<td>P1 COMMAND REGISTER</td>
</tr>
<tr>
<td>30EH</td>
<td>P1 DATA REGISTER</td>
</tr>
<tr>
<td>30DH</td>
<td>P2 COMMAND REGISTER</td>
</tr>
<tr>
<td>30CH</td>
<td>P2 DATA REGISTER</td>
</tr>
<tr>
<td>30BH</td>
<td>P3 COMMAND REGISTER</td>
</tr>
<tr>
<td>30AH</td>
<td>P3 DATA REGISTER</td>
</tr>
<tr>
<td>309H</td>
<td>P4 COMMAND REGISTER</td>
</tr>
<tr>
<td>308H</td>
<td>P4 DATA REGISTER</td>
</tr>
<tr>
<td>307H</td>
<td>P5 COMMAND REGISTER</td>
</tr>
<tr>
<td>306H</td>
<td>P5 DATA REGISTER</td>
</tr>
<tr>
<td>305H</td>
<td>P6 COMMAND REGISTER</td>
</tr>
<tr>
<td>304H</td>
<td>P6 DATA REGISTER</td>
</tr>
<tr>
<td>303H</td>
<td>P7 COMMAND REGISTER</td>
</tr>
<tr>
<td>302H</td>
<td>P7 DATA REGISTER</td>
</tr>
<tr>
<td>301H</td>
<td>P8 COMMAND REGISTER</td>
</tr>
<tr>
<td>300H</td>
<td>P8 DATA REGISTER</td>
</tr>
<tr>
<td>31EH</td>
<td>U7 RTS-BUS LATCH</td>
</tr>
<tr>
<td>31CH</td>
<td>U22 Rx AVAILABLE PORT</td>
</tr>
<tr>
<td>31AH</td>
<td>U16 BAUD RATE LATCH/PORT RESET</td>
</tr>
<tr>
<td>319H</td>
<td>P1-8 COMMAND REGISTER (ACCESS ALL 8 PORTS, WRITE ONLY)</td>
</tr>
<tr>
<td>318H</td>
<td>P1-8 DATA REGISTER (ACCESS ALL 8 PORTS, WRITE ONLY)</td>
</tr>
</tbody>
</table>

Table 2.2
2.2 Network Description

The network would be similar to ethernet in its protocols. The main exception is the use of the RS-232C protocol for communications between the network and the nodes. The following is a list of the specifications which will be required by the network.

Packet Description

| A | A | B | C | D | E | DATA | F | F | F | F | F |

Fig. 2.7

2.2.1 Header (A) The header will consist of 2 Bytes. The value for the bytes will be 55H. This value is being used instead of AAH so as to enable multiple numbers of data bits within the RS-232C protocol.

2.2.2 Source Address (B) The source address will be the third byte of the packet and is set to a number between 0 and 7 representing the port number of the source node.

2.2.3 Destination Address (C) The destination address will be the fourth
byte of the packet and be set to a number between 0 and 7 representing the port number of the destination node.

2.2.4 Packet Size (D) The packet size will occupy the fifth byte of the packet and show the size of the entire packet. This value will include the 10 bytes described here and be in the form of an unsigned value 0-126 which will indicate 1 less than the actual size.

2.2.5 Packet Type (E) The sixth byte of the packet will indicate the type of packet. The valid types are:
1, Data
2, Ack
3, Nack

2.2.6 Data The seventh thru one hundred twenty third (123) bytes of the packet are available to the individual nodes to be utilized in any manner they see fit.

2.2.7 Check Sum (F) The Last 4 bytes of the packet will be allocated as a check sum. The network will not check the value of the check sum so the student can choose any appropriate method.
2.3 Interface Specification

The interface specified for this fixture is that of RS-232C with standard serial communication. See the following sections for details of the physical specifications and data format.

2.3.1 Physical Interface

As stated above the physical interface of the fixture is a 5 wire version of RS-232C, which means that the data is exchanged on two data lines. One of the data lines is set to transmit data to the node while the other is receiving. This data transfer is full duplex, meaning that transmission and receive can occur at the same time. The voltage levels are per the standard, ie a data "zero" is indicated by the presence of between +3Vdc and +15Vdc with the fixture using a nominal +12Vdc, while a data "one" is indicated by the presence of between -3Vdc and -15Vdc with the fixture using a nominal -12Vdc.

In addition to the data lines there are two lines, a RTS (request to send) line and a CTS (clear to send) line. These lines are used to determine the state of the node to fixture interface. If the CTS line is active then the fixture knows that there is a node on the line. The RTS line is used by the fixture to indicate to the node if the network is busy. It is thru this request to send line that collisions are minimized.

The final line is signal ground which needs no explanation.

2.3.2 Data Format
The data format is programmable by the user of the network. The following table (Table 2.3) shows the various possible data formats.

Data Format Configuration Selection

<table>
<thead>
<tr>
<th>ITEM</th>
<th>POSSIBLE CONFIGURATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop Bits</td>
<td>1, 1.5, 2</td>
</tr>
<tr>
<td>Data Bits</td>
<td>5, 6, 7, 8</td>
</tr>
<tr>
<td>Parity</td>
<td>Enabled or Disabled</td>
</tr>
<tr>
<td></td>
<td>If Enabled, Odd or Even</td>
</tr>
<tr>
<td>Baud Rate</td>
<td>1200, 2400, 4800, 9600</td>
</tr>
</tbody>
</table>

Table 2.3

In any case the data will be of the following form (Fig. 2.9).
Data Format

When enabled the parity bit is the end of the data section.

Fig 2.8
Configuration of Network as Tested and Demonstrated

Standard TTY Terminal

IBM PC/AT or CLONE

Connection Box

Simulator PWBA is placed in an I/O slot of PC #1.

Fig. 2.9
3.0 Programming Environment

The environment that was picked for this fixture was that of the IBM PC/AT or Compatible. This environment was picked for several reasons. The most important of these reasons were, 1) the availability of a host both at work as well as in the school lab., and 2) the availability of a C compiler for this computer.

3.1 MS-DOS System Calls

In order to both simplify the programming task as well as to ensure greater reliability several functions were implemented as direct MS-DOS System calls the following is a list of the calls with a short description.

3.1.1 CLS

The MS-DOS CLS command has the action of clearing the console screen. It takes no arguments, is an Internal type command and has a simple syntax of "CLS".

3.1.2 COPY

The MS-DOS COPY command has the action of copying a source file to a destination file. It takes one or two arguments a source file and an optional destination file. It is an Internal type command and is being used to copy the data file to either the COM1, COM2, LPT1, or LPT2 PORT in order to print the
data file. Therefore the syntax for this command is “COPY <filename> <printer port>.

3.1.3 MORE

The MS-DOS MORE command has the action of displaying a full screen of data and then waiting for a user response before continuing. It is used by redirecting the output of a command such as type (see 3.1.5) to its input. It is an external type command and is being used in the following syntax “TYPE <filename>|MORE”.

3.1.4 PAUSE

The MS-DOS PAUSE command has the action of suspending execution until the user types a key. It has a simple syntax of “PAUSE”.

3.1.5 TYPE

The MS-DOS TYPE command has the action of displaying a file to the console screen. It is an internal type command and has the syntax of “TYPE <filename>“.

3.2 Microsoft C

The Microsoft C Compiler was picked to compile and link the software for this project mainly because of its availability and my previous knowledge of this compiler.

3.2.1 Compiler
The Microsoft C Compiler takes a normal C source code file and produces a relocatable object code file. Although there are many options available, for most programs including this one, the standard options are sufficient.

3.2.2 Linker

The Microsoft C Linker takes as input one or more object files. These files could have been compiled by the C compiler or assembled by a compatible assembler. In the case of this project all of the object files were compiled by the Microsoft C compiler. As in the compiler the linker has many options but for this project the standard default options were used.

3.2.3 Make

Microsoft C includes a facility called MAKE. This program is a software control program. In other words it uses the date and time stamps on the source and object files as well as the final executable file and determines which files must be compiled and or linked in order to have the latest revision source contained in the executable version. This facility relieves the programmer of the task of determining which source files have been modified since the last compile and link. (See 6.13 for a sample of the control file for the make program on this project)
4.0 Test Method

After the PWBA was wire wrapped it was first necessary to determine if certain functions were operation correctly. The first step was to determine if any of the addresses could be accessed. The RTS Register and the Baud Port were chosen as they were the simplest to verify operation on. Various values were written the these ports to verify proper operation.

The next step was to verify the operation of the oscillator chip at all of the frequencies which were to be used. Next the chip select and I/O Read/Write pins of each port were verified to be properly decoding. After these diagnostic checks were preformed a TTY terminal was connected and the PC placed in the DEBUG mode to verify read and write to each of the ports as well as the command values to be used in the programming of the 8251’s. At this point a serious flaw was noted. At times the chips were in an improper mode and could need resetting which would not always work. A signal was added to the expansion card via an unused bit in the baud register. This signal replaced the PC reset signal to the 8251’s and allowed for software controlled hard reset or the ports.

Finally a program was written for the second PC to send valid packets to the network. This program along with the TTY terminal allowed for full testing of the network management code, and is the configuration which was demonstrated at the conclusion of the project.
5.0 Discussion of Results

The results of the project were very favorable. The network can easily handle and determine the accuracy of packets that are sent from node to node. Collisions can be detected and reported to the user. The single lacking test of the network was that of having the full eight nodes trying to send as many packets as possible, but the nature of the beast leads one to believe that with proper code running on the eight nodes there is a great possibility for success. Since this project was developed as a teaching aid for a network course it will be interesting to see what kind of modifications can be made by future students.

Probably the single most important factor of this project is that of the knowledge of the inner workings of the IBM PC/AT. The knowledge gained from completing this project has already been put to use in the work place. I have designed code which converts the PC into several different types of controllers as well as data acquisition modules.

This project has also allowed me to grow and develop a stronger outlook as to my abilities.
6.0 Code

The following pages are a complete listing of the source code for this project. The actual code from the PC files will be shown in italic.

6.1 Netsim.c

The Netsim.c file contains the Main function which is compiled as the starting point of the code.

"****************************************************************** *
FILE NETSIM.C

Network Simulator

Written by: Chris Midgley
Date written: 10/6/87
Date last revision: 5/11/88

This software is written as the controlling software for a project to design local area network simulator. This simulator was designed as part of the requirements for the Master of Engineering Degree in the Computer Engineering Department at the Rochester Institute of Technology.

The software of this project is written in several modules. Proj.c is the main module. The other modules are defined below.

INCLUDE.H

This file contains the required include files as well as the external (global) variable declarations.
DEFINES.H
This file contains the declarations of the various constants used within the modules.

INIT__B.C
This file contains the function init__b() which initializes the baud frequency generator to the default baud rate as defined in defines.h by the constant BAUD__RATE.

PFILE.C
This file contains the function print__file which will cause the data file to be closed, printed and reopened for append.

NET__SUP.C
This file contains the following functions: process(c), outerr(port,c), shut__down(port), rsam(), psam(port,andval), packet__check(). These files perform various support type functions.

MENU.C
This file contains the function menu() which controls the printing of the main menu as well as returning the selection.

MISC.C
This file contains the following functions: center(s), getline(s), help(num), init(), getone(), check__end(). These functions also are of a support nature.

PHEAD.C
This file contains the function printhead(). It is used to print the heading to the menu.

RNET.C
This file contains the functions: run_net() and echeck(). Run_net() is the function which actually runs the interface between the 8 nodes. Echeck is the random error generator.

SETMAT.C
This file contains the function setmat(dist) which sets up the distance matrix used in shutting down nodes.

S_PAR.C
This file contains the functions set_parms(), set_sd(), sort_sd(i) which are used to set up the network parameters such as baud and distances.

#include <stdio.h>
#include <string.h>
#include <conio.h>
#include "defines.h"

/*The following global variables are used to simplify the passing of many extra parameters. They were chosen for specific reasons and are only modified at specific points in the program.

fpw is a global pointer to the trace file.

filename is used as a place to store the MS-DOS name of the trace file.

option is used as a utility string and is declared globally to reduce the number of strings in local functions.
mode1 and mode2 are used for command and mode instructions to the 8251's.

baud_val is the current value used to set the 8251's. It is set as defined by the constants in include.h.

baud_rate is the integer value of the baud i.e. 9600, 4800.

error_rate is the inverse of the error percentage. It is used in the error generation by performing the modulus function on the random number.

sdmat and dist_mat are used in determining the network topology.

FILE *fpw,*fopen();
char filename[32],option[32];
unsigned char mode1,mode2;
int no_errors,baud_val,baud_rate,error_rate,smat[8][8][2],dist_mat[8][8];

main(argc,argv)
int argc;
char * argv[];
{
    int opt;
    int acount,i,h;

    /* initialize the network */
    init();
    init_b();
    mode1 = MODE1;
    mode2 = MODE2;
    system("cls");
    acount = argc;

    /* if no file name on the command line */
if (acount = = 1)
{
/*prompt for a data file */
    printfhead(); /*print header message*/
    printf("Enter the data file to use ");
    getline(filename);
}
else
/* else if more than one argument prompt for proper usage and exit */
    if (acount > 2)
    {
        printf("%s: usage: %s <filename>
", argv[0], argv[0]);
        return(-1);
    }
else
/* else make a copy of the command line filename in the global var. filename */
    strcpy(filename, argv[1]);

/*set-up the timer for 1 mSec. timing*/

    outp(TIM_CONT, TIM_MODE);
    outp(TIM_P2, TIM_LOW);
    outp(TIM_P2, TIM_HIGH);
    outp(TIM_WR, TIM.START);

/* loop until an exit condition sets i to 0 */

    i = 1;
    while (i)
    {

while (1)
{
/* if null filename prompt as invalid and get a new file name */
    if (filename[0] == 'n')
    {
        system("cls");
        printf("\n\n\n\nINVALID FILE NAME RE-ENTER ");
        getline(filename);
    }
    else
        break;
}

/* if file name is a ?, display help else test if file exists. If file exists
then prompt for action i.e. append, overwrite, rename, exit. If the file
does not exist then open for write. */
if (! (filename[0] = = '?'))
{
    if (acount = = 1)
        filename[strlen(filename)-1] = '0';
    acount = 1;
    strcat(filename, ".dat");

    if ((fpw = fopen(filename, "r")) != NULL)
    {
        system("cls");
        printf("\n\n************************\nfile %s already exists\nEnter option\n(a)ppend, (r)ename, (o)verwrite, or (e)xit ", filename);
        option[0] = getoneQ;
        fclose(fpw);
        if (option[0] = = 'q' || option[0] = = 'Q')
        {
            fflush(fpw);
        }
    }
fclose(fpw);
system("cls");
exit();
}

h = 1;

while (h)
{
    switch(option[0])
    {
        case '?':
            help(1);
            printf("\n\n***********************\n*nfile %s already exists\nEnter option\n(a)pend, (r)ename, (o)verwrite, or (e)xit "",filename);
            option[0] = getone();
            break;
        case 'r':
            case 'R':
                system("cls");
                printf("\n\nEnter the data file to use ");
                getline(filename);
                h = 0;
                i = 1;
                break;
        case 'a':
            case 'A':
                h = 0;
                if ((fpw = fopen(filename,"a")) != NULL)
                    i = 0;
                else
                    {
                        system("cls");
                    }
printf("Could not open %s
",filename);
printf("Enter the data file to use ");
}
break;
case 'o':
case 'O':
    h = 0;
    if((fpw = fopen(filename, "w")) != NULL)
        i = 0;
    else
        {
            system("cls");
            printf("Could not open %s
",filename);
            printf("Enter the data file to use ");
        }
    break;
case 'e':
case 'E':
case 'q':
case 'Q':
    system("cls");
    if(check_end() )
    {
        h = 0;
        fflush(fpw);
        fclose(fpw);
        system("cls");
        return(0);
    }
else
    {
        h = 0;
        i = 1;
        printf("Enter the data file to use ");
getline(filename);
    system("cls");

    }  
    break;
    default :  
        h = 1;
        system("cls");
        printf("INVALID OPTION\n");
        printf("\n\n************************

file %s already exists\n
Enter option\n(a)ppend, (r)ename, (o)verwrite, or (e)xit ",filename);
        option[0] = getone();
        break;
    }
}
}

else  
{
    if ((fpw = fopen(filename,"w")) != NULL)
        i = 0;
    else
        {
            system("cls");
            printf("\n\n\nCould not open %s\n",filename);
            printf("\n\n\nEnter the data file to use ");
            getline(filename);
        }
}
}
else  
{
    help(0);
    printf("\n\n\nEnter the data file to use ");
    getline(filename);
} }

system("cls");

/* loop thru menu and perform the requested action */
do {
    opt = menu();
    system("cls");
    switch (opt) {
        case 'q':
        case 'Q':
        case 'e':
        case 'E':
            if (check__end() ) {
                fflush(fpw);
                fclose(fpw);
                system("cls");
                return(1);
            }
            else {
                system("cls");
            }
            break;
        case '?':
            help(2);
            break;
        case 's':
        case 'S':
            if ((i = set__parms()) == -1)
if (check_end())
{
    fflush(fpw);
    fclose(fpw);
    system("cls");
    return(1);
}
else
{
    system("cls");
}
}
break;
case 'p':
case 'P':
    if ((i = print_file(filename)) = = -1)
    {
        fflush(fpw);
        fclose(fpw);
        if (check_end())
        {
            fflush(fpw);
            fclose(fpw);
            system("cls");
            return(1);
        }
        else
        {
            system("cls");
        }
    }
    break;
case 'r':
case 'R':
    if ((i = run_net()) = = -1)
    {
        fflush(fpw);
        fclose(fpw);
        if (check_end() )
        {
            fflush(fpw);
            fclose(fpw);
            system("cls");
            return(1);
        }
        else
        {
            system("cls");
        }
    }
    if (i = = -2)
    {
        system("cls");
        printf("NETWORK FAILURE RETURNING TO MENU\n");
    }
    break;

default:
    printf("INVALID OPTION\n");
    break;

    }
}
while (opt);
6.2 Init_b.c

This file contains the function INIT_B which is used to set up the
programmable crystal oscillator to the correct frequency.

/******************************************************************************/
FILE INIT_B.C

Network Simulator

Written by: Chris Midgley
Date written: 10/6/87
Date last revision: 5/11/88

This software is written as the controlling software for a project to design
local area network simulator. This simulator was designed as part of the
requirements for the Master of Engineering Degree in the Computer
Engineering Department at the Rochester Institute of Technology.

This function is called during initialization to set the system baud rate to
the default value as defined by the constant BAUD_RATE in defines.h. This function sets the oscillator port with the correct frequency for the selected
baud rate.
******************************************************************************/
#include "include.h"

init_b()
{
    int i;

\[ i = \text{BAUD\_RATE}; \]

\begin{verbatim}
switch(i)
{
    case 1200:
        baud_val = B1200;
        outp(BAUD_PORT,B1200R);
        outp(BAUD_PORT,B1200);
        break;
    case 2400:
        baud_val = B2400;
        outp(BAUD_PORT,B2400R);
        outp(BAUD_PORT,B2400);
        break;
    case 4800:
        baud_val = B4800;
        outp(BAUD_PORT,B4800R);
        outp(BAUD_PORT,B4800);
        break;
    case 9600:
        baud_val = B9600;
        outp(BAUD_PORT,B9600R);
        outp(BAUD_PORT,B9600);
        break;
}
\end{verbatim}
6.3 Pfile.c

This file contains the functions print__file, screen, com and lpt which are used to either display the current contents of the data log file to the screen or to copy them to a printer hooked to the serial or parallel port of the computer.

6.3.1 Print__file

This function oversees the display or printing of the data file. It takes care of closing, the call to display or print, and reopening the data file for append.

FILE PFILE.C

Network Simulator

Written by: Chris Midgley
Date written: 10/6/87
Date last revision: 5/11/88

This software is written as the controlling software for a project to design a local area network simulator. This simulator was designed as part of the requirements for the Master of Engineering Degree in the Computer Engineering Department at the Rochester Institute of Technology.

This function causes the data file to be printed to either the com1, com2, lpt1, lpt2 or the screen. It first flushes the file, then closes it, makes a MS-DOS system call to print or display and finally reopen the data file in the append mode.
#include "include.h"

print_file(filename)
char * filename;
{
    char c;

    fflush(fpw);
    fclose(fpw);
    system("cls");
    printf("\n\n");
    printf("PRINT SUB-MENU\n\n(S)creen\n(C)om\n(L)pt\n(E)xit\nEnter option: ");
    while (1)
    {
        if (kbhit())
        {
            c = getch();
            break;
        }
    }
    switch (c)
    {
    case 'S':
    case 's':
    screen();
    fpw = fopen(filename, "a");
    system("cls");
    return(1);
    case 'c':

    return(1);
}
case 'C':
    com();
    fpw = fopen(filename,"a");
    system("cls");
    return(1);
    case 'l':
    case 'L':
    lpt();
    fpw = fopen(filename,"a");
    system("cls");
    return(1);
    case 'e':
    case 'E':
    fpw = fopen(filename,"a");
    system("cls");
    return(1);
    default:
    system("cls");
    printf("\n\n INVALID OPTION  \n");
    break;

6.3.2 Screen

This function causes the data file to be displayed to the screen.

screen()
{
    chars[128];
system("cls");
strcpy(s, "type ");
strcat(s, filename);
strcat(s, "|more");
System(s);
}

6.3.3 Com

This function causes the data file to be printed to the serial port of the host.

com() {
    char s[128], s2[2], c = '0';

    strcpy(s, "copy ");
    strcpy(s2, "0");
    strcat(s, filename);
    while ((c != '1') && (c != '2'))
    {
        printf(" PORT 1 or 2 ");
        c = getch();
        if ((c != '1') && (c != '2'))
            printf("INVALID PORT------");
    }
    s2[0] = c;
    strcat(s, " com");
    strcat(s, s2);
    printf("%s\n", s);
    system(s);
}
6.3.4 lpt

This function causes the data file to be printed to the parallel port of the host.

lpt()
{
    char s[128], s2[2], c = '0';

    strcpy(s, "copy ");
    strcpy(s2, "0");
    strcat(s, filename);
    while ((c != '1') && (c != '2'))
    {
        printf(" PORT 1 or 2 ");
        c = getch();
        if ((c != '1') && (c != '2'))
            printf(" INVALID PORT------");
    }
    s2[0] = c;
    strcat(s, " lpt");
    strcat(s, s2);
    printf("%s
", s);
    system(s);
}
6.4 Net_sup.c

The Net_sup.c contains various functions which support the operation of the network. They are discussed in sections 6.4.1 thru 6.4.6.

6.4.1 Process

The function process is used to "process" each character that is received. It determines if a collision has occurred and if an error is to be induced it induces that error.

#include "include.h"
define TIME_OUT 2000

/* See file RNET.C for variable uses explanations. */
extern long int stime;
extern long int time_out;
extern int make_error, pn, cur_port, lpc, collision, longerr, ccount, pacsizel;
extern unsigned char packet[515];

/****************************
This function processes each character received to determine if a collision occurs or if any other problem exists. It files the character into the variable packet.
****************************************************************/
process(c)
unsigned char c;
{
  unsigned char ch;

ccount ++;
if (ccount == 12 && make_error)
  {
    printf("********NETWORK INDUCED ERROR********\n");
    fprintf(fpw,"********NETWORK INDUCED ERROR********\n");
  }
switch(c)
{
case 1:
  ch = inp(PORT1);
  if (ccount == 12 && make_error)
    outerr(PORT1,ch);
  else
    outp(PORTA,ch);
  break;

  case 2:
  ch = inp(PORT2);
  if (ccount == 12 && make_error)
outerr(PORT2, ch);
else
  outp(PORTA, ch);
break;

case 4:
  ch = inp(PORT3);
  if (ccount == 12 && make_error)
    outerr(PORT3, ch);
  else
    outp(PORTA, ch);
break;

case 8:
  ch = inp(PORT4);
  if (ccount == 12 && make_error)
    outerr(PORT4, ch);
  else
    outp(PORTA, ch);
break;

case 16:
  ch = inp(PORT5);
  if (ccount == 12 && make_error)
    outerr(PORT5, ch);
  else
    outp(PORTA, ch);
break;

case 32:
  ch = inp(PORT6);
  if (ccount == 12 && make_error)
    outerr(PORT6, ch);
  else
    outp(PORTA, ch);
break;

case 64:
  ch = inp(PORT7);
if (ccount == 12 && make_error)
    outerr(PORT7,ch);
else
    outp(PORTA,ch);
break;

case 128:
    ch = inp(PORT8);
    if (ccount == 12 && make_error)
        outerr(PORT8,ch);
    else
        outp(PORTA,ch);
    break;

default:
    if (! collision)
    {
        ch = 0;
        printf("\ncollision between ports ");
        fprintf(fpw,"\ncollision between ports ");
        if (c&1)
        {
            ch = inp(PORT1);
            printf("1 ");
            fprintf(fpw,"1 ");
        }
        if (c&2)
        {
            ch = inp(PORT2);
            printf("2 ");
            fprintf(fpw,"2 ");
        }
        if (c&4)
        {
            ch = inp(PORT3);
            printf("3 ");
        }
```
    fprintf(fpw, "3 ");
}
if (c&8)
{
    ch  = inp(PORT4);
    printf("4 ");
    fprintf(fpw, "4 ");
}
if (c&16)
{
    ch  = inp(PORT5);
    printf("5 ");
    fprintf(fpw, "5 ");
}
if (c&32)
{
    ch  = inp(PORT6);
    printf("6 ");
    fprintf(fpw, "6 ");
}
if (c&64)
{
    ch  = inp(PORT7);
    printf("7 ");
    fprintf(fpw, "7 ");
}
if (c&128)
{
    ch  = inp(PORT8);
    printf("8 ");
    fprintf(fpw, "8 ");
}

printf("\n");
```
fprintf(fpw, "\n");
outp(PORTA, ch);
return(1);
}
if (c != cur_port)
{
  if (! collision)
  {
    ch = 0;
    printf("collision between ports ");
    fprintf(fpw, "collision between ports ");
    if (cur_port & 1 || c & 1)
    {
      ch = inp(PORT1);
      printf("1 ");
      fprintf(fpw, "1 ");
    }
    if (cur_port & 2 || c & 2)
    {
      ch = inp(PORT2);
      printf("2 ");
      fprintf(fpw, "2 ");
    }
    if (cur_port & 4 || c & 4)
    {
      ch = inp(PORT3);
      printf("3 ");
      fprintf(fpw, "3 ");
    }
    if (cur_port & 8 || c & 8)
    {
      ch = inp(PORT4);
      printf("4 ");
      fprintf(fpw, "4 ");
if (cur_port & 16 || c & 16) {
    ch* = inp(PORT5);
    printf("5 ");
    fprintf(fpw, "5 ");
}
if (cur_port & 32 || c & 32) {
    ch* = inp(PORT6);
    printf("6 ");
    fprintf(fpw, "6 ");
}
if (cur_port & 64 || c & 64) {
    ch* = inp(PORT7);
    printf("7 ");
    fprintf(fpw, "7 ");
}
if (cur_port & 128 || c & 128) {
    ch* = inp(PORT8);
    printf("8 ");
    fprintf(fpw, "8 ");
}
printf("\n");
fprintf(fpw, "\n");
outp(PORTA, ch + 1);
return(1);
}

packet[count-1] = ch;
return(0);
6.4.2 Outerr

The function Outerr causes the proper character to be echoed to the sending unit but an error to be sent to all other nodes. The error is simply the character received plus one.

/***************************************************************************
 * This function causes a character to be sent to all ports incorrectly except for the port which sent it. It simulates an error happening during the character transmission.
 ***************************************************************************/

outerr(port, c)
int port;
unsigned char c;
{

switch (port)
{
  case PORT1:
    outp(PORT1, c);
    outp(PORT2, c + 1);
    outp(PORT3, c + 1);
    outp(PORT4, c + 1);
    outp(PORT5, c + 1);
    outp(PORT6, c + 1);
    outp(PORT7, c + 1);
    outp(PORT8, c + 1);
    break;
  case PORT2:
    outp(PORT1, c + 1);
    outp(PORT2, c);

  break;
}
outp(PORT3,c + 1);
outp(PORT4,c + 1);
outp(PORT5,c + 1);
outp(PORT6,c + 1);
outp(PORT7,c + 1);
outp(PORT8,c + 1);
break;

case PORT3:
    outp(PORT1,c + 1);
    outp(PORT2,c + 1);
    outp(PORT3,c);
    outp(PORT4,c + 1);
    outp(PORT5,c + 1);
    outp(PORT6,c + 1);
    outp(PORT7,c + 1);
    outp(PORT8,c + 1);
    break;

case PORT4:
    outp(PORT1,c + 1);
    outp(PORT2,c + 1);
    outp(PORT3,c + 1);
    outp(PORT4,c);
    outp(PORT5,c + 1);
    outp(PORT6,c + 1);
    outp(PORT7,c + 1);
    outp(PORT8,c + 1);
    break;

case PORT5:
    outp(PORT1,c + 1);
    outp(PORT2,c + 1);
    outp(PORT3,c + 1);
    outp(PORT4,c + 1);
    outp(PORT5,c);
    outp(PORT6,c + 1);
outp(PORT7, c + 1);
outp(PORT8, c + 1);
break;
case PORT6:
  outp(PORT1, c + 1);
  outp(PORT2, c + 1);
  outp(PORT3, c + 1);
  outp(PORT4, c + 1);
  outp(PORT5, c + 1);
  outp(PORT6, c);
  outp(PORT7, c + 1);
  outp(PORT8, c + 1);
  break;
case PORT7:
  outp(PORT1, c + 1);
  outp(PORT2, c + 1);
  outp(PORT3, c + 1);
  outp(PORT4, c + 1);
  outp(PORT5, c + 1);
  outp(PORT6, c + 1);
  outp(PORT7, c);
  outp(PORT8, c + 1);
  break;
case PORT8:
  outp(PORT1, c + 1);
  outp(PORT2, c + 1);
  outp(PORT3, c + 1);
  outp(PORT4, c + 1);
  outp(PORT5, c + 1);
  outp(PORT6, c + 1);
  outp(PORT7, c + 1);
  outp(PORT8, c);
  break;
}
6.4.3 Shut_down

The shut_down function causes the RTS lines of each port to become inactive in a sequence as determined by the network topology. This function is called when the first character of a packet is detected.

/***************************************************************
This function causes the ports to be shut off at the various time delays as determined by the topology of the network.
***************************************************************/

shut_down(port)
unsigned char port;

int rtsval, portnum, i, j;

rtsval = 0;

switch (port)
{
    case 1:
        portnum = 0;
        break;
    case 2:
        portnum = 1;
        break;
    case 4:
        portnum = 2;
        break;
    case 8:
        portnum = 3;
break;
case 16:
    portnum = 4;
    break;
case 32:
    portnum = 5;
    break;
case 64:
    portnum = 6;
    break;
case 128:
    portnum = 7;
    break;
default:
    outp(RTS_PORT,255);
    return(-1);
}

j = 0;

for (i = 1; i <= 500; i + +)
{
    if (j < 8 && sdmatrix[portnum][j][0] <= i)
    {
        rtsval = rtsval | sdmatrix[portnum][j][1];
        outp(RTS_PORT, rtsval);
        j + +;
    }
}

rtsval ` = port;
outp(RTS_PORT, rtsval);
return(portnum + 1);

}

6.4.4 Rsam

The function Rsam causes a 1 Milli Second delay. This causes the program to sample the ports once every milli second and also to count for error insertion as well as time out.

/***************************************************************************/

This function causes the 1 mSec. time to be invoked.
/******************************************************************************/

rsam()
{
    int i = 0;
    int j,k;
    char c;

    while (1)
    {
        if (psam(TIM_READ,32))
        {
            stime ++ ;
            while (1)
            {
                if (! (psam(TIM_READ,32)))
                break;
            }
            break;
        }
    }
6.4.5 Psam

The function Psam is used to test a bit of a port. It is passed the port address and the value of the bit to be tested. It returns a zero if the bit is not set and the value of the bit if it is.

/* ***************************************************************
   This function returns a boolean representing weather a bit of the 
   requested port is set or not. 
   ****************************************************************/
psam(port, andval)
int port, andval;
{
    char cr;

    cr = inp(port);
    return (cr & andval);
}

6.5.6 Packet_check

The Packet_check function evaluates the bytes in the packet and prints the results to both the screen and the data log file. It checks the packet for header, size, source, destination and type.
/********************************************
This function checks the packet for header and size accuracy.
*********************************************/
packet_check()
{

    int error = 0;

    if (collision)
        return(0);
    printf("CHECKING THE PACKET\n");
    if ( packet[0] != 0x55 || packet[1] != 0x55 )
    {
        printf("improper pre-amble\n");
        fprintf(fpw,"improper pre-amble\n");
        error = 1;
    }
    if (packet[2] != pn)
    {
        printf("packet port number(%d) does not match the port received from(%d)\n",packet[2],pn);
        fprintf(fpw,"packet port number(%d) does not match the port received from(%d)\n",packet[2],pn);
    }
    {
        printf("packet is from port INVALID ");
        fprintf(fpw,"packet is from port INVALID ");
    }
    else
    {
        printf("packet is from port %d ",packet[2]);
    }
fprintf(fpw, "packet is from port %d ", packet[2]);
}
{
    printf("to port INVALID ");
    fprintf(fpw, "to port INVALID ");
}
else
{
    printf("to port %d", packet[3]);
    fprintf(fpw, "to port %d", packet[3]);
}
if ((packet[4] + 10) == ccount)
{
    printf(" is of size %d and is of type ", packet[4] + 10);
    fprintf(fpw, " is of size %d and is of type ", packet[4] + 10);
}
else
{
    printf("\npacket size field does not match number of characters sent\nsize field = %d, number of data bytes = %d\npacket type is ", packet[4], ccount-10);
    fprintf(fpw, "packet size field does not match number of characters sent\nsize field = %d, number of data bytes = %d\npacket type is ", packet[4], ccount-10);
}
switch (packet[5])
{
case 1 :
    printf("data\n");
    fprintf(fpw, "data\n");
    break;

case 2 :
    printf("ack\n");
    fprintf(fpw, "ack\n");
    break;

    case 3:
      printf("nack\n");
      fprintf(fpw, "nack\n");
      break;

    default:
      printf("unknown\n");
      fprintf(fpw, "unknown\n");
      break;

  }
6.5 Menu.c

This file contains the function menu which causes the main menu of options to be displayed to the console screen. It also receives the single character response from the keyboard and passes it as the returned value.

/*******************************************************************************/
FILE MENU.C

Network Simulator

Written by: Chris Midgley
Date written: 10/6/87
Date last revision: 5/11/88

This software is written as the controlling software for a project to design a local area network simulator. This simulator was designed as part of the requirements for the Master of Engineering Degree in the Computer Engineering Department at the Rochester Institute of Technology.

This function causes the menu to be displayed on the screen. It reads standard input (the keyboard) and returns the value which was entered. All input validation is done at the upper level.

*******************************************************************************/
#include "include.h"

menu()
{
    char c;

    printhead();

}
```c
printf("\n\nOPTIONS:\n\n(S)et-up the System Parameters\n\n(R)un the Network Simulation\n\n(P)rint the Current Data File\n\n(E)xit to DOS\n\n(?) Help\n\n");
return(getone());
```
6.6 Misc.c

The file misc.c contains the functions center, getline, help, init, getone, and check_end. These functions as the name of the file denotes are of a miscellaneous nature.

6.6.1 Center

The function center causes the string passed as the argument to be centered on an 80 column screen.

FILE MISC.C

Network Simulator

Written by: Chris Midgley
Date written: 10/6/87
Date last revision: 5/11/88

This software is written as the controlling software for a project to design a local area network simulator. This simulator was designed as part of the requirements for the Master of Engineering Degree in the Computer Engineering Department at the Rochester Institute of Technology.

There are several functions in this file. They are described at their declaration line.

#include "include.h"

/* miscellaneous nature */
This function is used to center a string on the 80 column screen.

```c
center(s)
char * s;
{
    int i,j;
    j = strlen(s);
    j = 80-j;
    j/ = 2;
    for (i = 0; i < j; i++)
        printf(" ");
    printf("%s\n",s);
}
```

### 6.6.2 Getline

The function getline is passed the pointer to a string and returns the number of characters that it has placed into that string. It gets the characters from the keyboard and returns when a new line is encountered.

```c
getline(s)
char * s;
{
    int i = 0;
```
char c;

while (1)
{
    c = getchar();
    s[i + +] = c;
    if (c == '\n')
    {
        s[i] = '\0';
        return(i);
    }
}

6.6.3 Help

The function help is passed an integer parameter which it uses to determine which of the help files it will display to the console screen. The help files are displayed using the system call to the type command.

/***************************************************************
 This function displays the various help files associated with the project.
***************************************************************/

help(num)
int num;
{
    system("cls");
    printf("\n\n\n\n");
    switch(num)
    {
    case 0:
        system("type netsim0.hlp");


63
break;
case 1:
    system("type netsim1.hlp");
    break;
case 2:
    system("type netsim2.hlp");
    break;
case 3:
    system("type netsim3.hlp");
    break;
case 4:
    system("type netsim4.hlp");
    break;
case 5:
    system("type netsim5.hlp");
    break;
case 8:
    system("type netsim8.hlp");
    break;
case 9:
    system("type netsim9.hlp");
    break;
case 10:
    system("type netsim10.hlp");
    break;
case 11:
    system("type netsim11.hlp");
    break;
default:
    break;
}
printf("\nPRESS RETURN TO CONTINUE\n");
getline(option);
 system("cls");
6.6.4 Init

The function init causes the network to be initialized. It calls the functions setmat and set_sd to initialize the network topology variables. It also sets the error rate and baud rate variables to the initial values as defined in the defines.h file.

/************************************************************************
 * This program handles the initialization of many of the system parameters.
 * ************************************************************************/

init()
{
    int i,dist[8];

    for (i = 0; i < 8; i++)
        dist[i] = i*500/7;
    setmat(dist);
    set_sd();
    error_rate = ERROR_RATE;
    baud_rate = BAUD_RATE;
}

6.4.5 Getone
The getone function has the effect of waiting for the keyboard to be hit and then returning the character which was struck. If the character is a carriage return it returns the newline character.

/**********************************************************************
 * This function gets one character from the keyboard and echoes it back to the screen.
 ***********************************************************************/
getone()
{
    char c;

    while (1)
    {
        if (kbhit())
        {
            if ((c = getch()) == 13)
                printf("\n");
            else
                printf("\%c",c);
            return(c);
        }
    }
}

6.6.6 Check_end

The function check_end is called when ever the option of exit is called. It is responsible for displaying a message on the console screen requesting confirmation of the exit act. A user response of ‘Y’ or ‘y’ will cause the exit to
continue. Any other response will return the user to where he was and allow the same previous options.

/***************************************************************
This function checks to make sure that you want to exit the system. It returns a 1 if you want to exit and a 0 if you do not.
***************************************************************/
check_end()
{
  char c;
  printf("Are you sure you want to exit? (Y)es or (N)o ");
  if ((c = getone()) == 'y' || c == 'Y')
  {
    fclose(fpw);
    return(1);
  }
  else
  {
    return(0);
  }
}
6.7  phead.c

The file phead.c contains the function printhead which displays the heading message for the main menu of the fixture.

/*******************************************************************************/

FILE PHEAD.C

Network Simulator

  Written by:    Chris Midgley
  Date written:  10/6/87
  Date last revision:  5/11/88

This software is written as the controlling software for a project to design a local area network simulator. This simulator was designed as part of the requirements for the Master of Engineering Degree in the Computer Engineering Department at the Rochester Institute of Technology.

This function prints out the heading for the various display screens.
/*******************************************************************************/
#include "include.h"

printhead()
{
  printf("\n\n\n");
  center("Network Simulator");
  printf("\n");
  center("Version 1.1");
  printf("\n");
  printf("\n");
}
printf("\n");
}
6.8 Rnet.c

The file rnet.c contains two functions, run_net and e_check. These functions are key, along with the functions in net_sup.c, in the operation of the network.

6.8.1 Run_net

The function run_net controls the operation of the network. It is called from the main function and assumes control until the “M” key is struck to return to the menu, or the “E” key to exit NETSIM.

/****************************************************************/
FILE RNET.C

Network Simulator

Written by: Chris Midgley
Date written: 10/6/87
Date last revision: 5/11/88

This software is written as the controlling software for a project to design a local area network simulator. This simulator was designed as part of the requirements for the Master of Engineering Degree in the Computer Engineering Department at the Rochester Institute of Technology.

************************************************************************/
#include "include.h"
#define TIME_OUT 1000

/* etime is a counter which provides the random number for error generation

stime is the time out timer incremented by the mother board clock

time_out is the variable used for the actual time out it is first set to
the default value and then modified by the baud rate.

make_error is a flag which is set if the packet should have a network induced
error.

cur_port maintains the current active port and is used in detecting collisions.

lpc is a flag used to only print the long packet message

collision is the flag which is set if a collision

longerr is a flag which is set if a long packet

ccount is a character counter for the packet.

pacs is set from the packet size field and used to compare the packet
length and actual length.

packet is a character string which holds the packet */

long int etime,stime;
long int time_out;
int i, make_error,pn,cur_port,lpc,collision,longerr,ccount,pacsize;
unsigned char packet[265];
/* This function is the network operation function. It is called from the main program. It oversees the operation of the expansion board and logs the information transmitted over the net. */
run_net()
{
    unsigned char c;
    int col;

    time_out = TIME_OUT;
    longerr = 0;
    etime = inp(TIM_READ);

    /* modify the timeout value based on the baud rate. */
    switch (baud_val)
    {
      case B1200 :
        time_out* = 8;
        break;
      case B2400 :
        time_out* = 4;
        break;
      case B4800 :
        time_out* = 2;
        break;
      default :
        break;
    }

    /* reset ports */
    port__reset();

/* Loop looking for the first character of the packet. Process a packet when detected. */

while (1)
{

    lpc = 1;
    ccount = 0;
    collision = 0;

    if (kbhit())
    {
        if ((c = getch()) == 'm' || c == 'M')
        {
            system("cls");
            return(1);
        }
        else
            if (c == 'q' || c == 'Q' || c == 'e' || c == 'E')
            {
                system("cls");
                return(-1);
            }
    }

    stime = 0;

    outp(RTS_PORT,0);

    if (c = inp(AVAIL_PORT))
    {

73
pn = shut_down(c);
cur_port = c;
collision + = process(c);
printf("....
");
while(1)
{
    if (ccount == 256)
    {
        printf("NETWORK FAILURE RETURNING TO MENU\n");
        fprintf(fpw, "NETWORK FAILURE RETURNING TO MENU\n");
        return(-2);
    }
    if (ccount == 128 && lpc)
    {
        printf("LONG PACKET\n");
        fprintf(fpw, "LONG PACKET\n");
        lpc = 0;
    }
    if (c = inp(AVAIL_PORT))
    {
        stime = 0;
        collision + = process(c);
    }
    rsam();
    if (kbhit())
    {
        if ((c = getch()) = = 'm' || c = = 'M')
        {
            system("cls");
            return(1);
        }
        else
        {
            if (c = = 'q' || c = = 'Q' || c = = 'e' || c = = 'E')
            {
                // Code here
            }
        }
    }
}
{
    system("cls");
    return(-1);
}

if (stime > time_out)
{
    break;
}
else
{
    stime + +;
    etime + +;
}
}

port_reset();
}

/* If no collision then process the packet and print it. */
if (!collision)
{
    if (ccount)
    {
        packet_check();
        for (i = 0; i < ccount; i + +)
        {
            printf("%c",packet[i]);
            fprintf(fpw,"%c",packet[i]);
        }
        fprintf(fpw,"%c",packet[i]);
    }
}
6.8.2 Echeck

The function `echeck` determines if an error should be generated in the next packet.

```c
/* This function generates a random error based on the etime counter and the selected error rate. It takes the etime and mods by 100 then mods by the error rate. */
echeck()
{
    if (no_errors)
        return(1);
    else
        return(!((int)(etime%100L)) % error_rate));
}
```
6.8.3 Port_reset

The port_reset function is called at the start of the network run and then after each packet. It insures that all ports are in the proper configuration.

/* This function causes the ports on the 8251's to be totally reset. */
port_reset()
{
    make__error = echeck();
    outp(BAUD_PORT, PORT__RESET);
    outp(BAUD_PORT, baud__val);
    outp(RTS_PORT, 0x00);
    outp(CPORTA, mode1);
    outp(CPORTA, mode2);
}
6.9 Setmat.c

The file setmat.c contains the function setmat. This function sets up the network topology. It is called from the set_parms() function if topology set up is requested.

/***************************************************************************/
FILE SETMAT.C
Network Simulator

Written by: Chris Midgley
Date: 10/6/87

This software is written as the controlling software for a project to design a local area network simulator. This simulator was designed as part of the requirements for the Master of Engineering Degree in the Computer Engineering Department at the Rochester Institute of Technology.

#include "include.h"

/* This function sets up the network topology. It is called from the set_parms() function if topology set up is requested. */

setmat(dist)
int dist[];
{
    int i,j,temp;

    for (i = 0; i < 8; i++)
    {

for (j = 0; j < 8; j++)
{
    temp = dist[j] - dist[i];
    if (temp < 0)
        temp *= (-1);
    dist_mat[i][j] = temp;
}

}
6.10 s_par.c

The file s_par.c contains four functions, set_parms, set_sd, sort_sd and port_parms. These functions are responsible for the setup of the network parameters.

6.10.1 set_parms

The set_parms function is called by the main function when the user requests to setup the network. It modifies the data format variables as well as the error rate and network topology.

/* ****************************************** */

FILE s_par.c

Network Simulator

Written by: Chris Midgley
Date written: 10/6/87
Date last revision: 5/11/88

This software is written as the controlling software for a project to design a local area network simulator. This simulator was designed as part of the requirements for the Master of Engineering Degree in the Computer Engineering Department at the Rochester Institute of Technology.

/* ****************************************** */
#include "include.h"

/* This function sets the network parameters. */
set_parms()
{
    int i,j,k,ii,dist[8],temp;
    char c,dist__str[32];

    printf("CURRENT NETWORK NODE LOCATIONS:\n\n");
    for (i = 0; i < 8; i ++)
        printf("Node \%d ",i + 1);
    printf("\n");
    for (i = 0; i < 8; i ++)
        printf("\%6d ",dist__mat[i][0]);
    printf("\n");
    printf("\nDo you want to change them? ");
    dist__str[0] = getone();
    while (dist__str[0] = = '?' )
    {
        help(3);
        printf("CURRENT NETWORK NODE LOCATIONS:\n\n");
        for (i = 0; i < 8; i ++)
            printf("Node \%d ",i + 1);
        printf("\n");
        for (i = 0; i < 8; i ++)
            printf("\%6d ",dist__mat[i][0]);
        printf("\n");
        printf("\nDo you want to change them? ");
        dist__str[0] = getone();
    }
    if (dist__str[0] = = 'q' || dist__str[0] = = 'Q')
    {
        fflush(fpw);
        fclose(fpw);
        system("cls");
        return(-1);
    }
if (! (dist_str[0] == 'y' || dist_str[0] == 'Y'))
    system("cls");
else
{
    printf("You may now enter the location on the net of each
node.  

    Node 1 is fixed at the end of the network.  
    Each other node can be placed on the network at a point
up to

    "500 'meters' from Node 1, and at least 10 meters
apart.  

    for (i = 0; i < 8; i + +)
    {
        dist[i] = dist_mat[i][0];
    }
    for (i = 1; i < 8; i + +)
    {
        j = 1;
        while (j)
        {
            printf("Node %d location = %d
Do you want to
modify? ", i + 1, dist_mat[0][i]);
            if ((c = getoneO) != 'y' & c != 'Y')
                break;
            printf("Enter the location of Node %d: ", i + 1);
            getline(dist_str);
            temp = atoi(dist_str);
            if (temp > 0 && temp < = 500)
            {
                for (k = 0; k < 8; k + +)
                {
                    if (i == k || temp-dist[k] > = NODE_MIN ||
                        dist[k]-temp > = NODE_MIN)
else
{
    j = 1;
    system("cls");

    printf("******************************************************
    New value for node %d is within %d of node %d
    ******************************************************", i + 1, NO DE_MIN, k + 1);

    printf("CURRENT NETWORK NODE LOCATIONS:
    ");
    for (ii = 0; ii < 8; ii++)
    {
        printf("Node %d ", ii + 1);
        printf("\n");
        for (ii = 0; ii < 8; ii++)
        {
            printf("%6d ", dist[ii]);
            printf("\n");
            break;
        }
    }
    if (!j)
    {
        dist[i] = temp;
        printf("Node %d located at %d\n", i + 1, dist[i]);
    }
}
else

    printf("\nINVALID VALUE\n");

    printf("\n");
}

setmat(dist);
printf("Press Return to Continue\n");
getline(dist_str);
system("cls");
}
set_sd();
while (1)
{
    printf("Baud rate is set to %d\n\nDo you want to modify? ",baud_rate);
    dist_str[0] = getone();
    while (dist_str[0] = = '?')
    {
        help(5);
        printf("Baud rate is set to %d\n\nDo you want to modify? ",baud_rate);
        dist_str[0] = getone();
    }
    if (dist_str[0] = = 'q' || dist_str[0] = = 'Q')
    {
        fflush(fpw);
        fclose(fpw);
        system("cls");
        return(-1);
    }
    if (! (dist_str[0] = = 'y' || dist_str[0] = = 'Y'))
    {
        system("cls");
        break;
    }
    else
    {
        printf("\nEnter the new baud rate ");
        getline(dist_str);
        i = atoi(dist_str);  
}
system("cls");
if (i != 1200 && i != 2400 && i != 4800 && i != 9600) {
    printf("INVALID ERROR RATE INPUT\n");
} else {
    baud_rate = i;
}
}

switch(baud_rate) {
    case 1200:
        baud_val = B1200;
        outp(BAUD_PORT,B1200R);
        outp(BAUD_PORT,baud_val);
        break;
    case 2400:
        baud_val = B2400;
        outp(BAUD_PORT,B2400R);
        outp(BAUD_PORT,baud_val);
        break;
    case 4800:
        baud_val = B4800;
        outp(BAUD_PORT,B4800R);
        outp(BAUD_PORT,baud_val);
        break;
    case 9600:
        baud_val = B9600;
        outp(BAUD_PORT,B9600R);
        outp(BAUD_PORT,baud_val);
        break;
}
port_parms();
system("cls");
while (1) {
    if (no_errors),
        printf("Error rate is set to 0\%\n\nDo you want to modify? ");
    else
        printf("Error rate is set to %d\%\n\nDo you want to modify? ",(100/error_rate));
    dist_str[0] = getone();
    while (dist_str[0] = = '?')
    {
        help(4);
        if (no_errors),
            printf("Error rate is set to 0\%\n\nDo you want to modify? ");
        else
            printf("Error rate is set to %d\%\n\nDo you want to modify? ",(100/error_rate));
        dist_str[0] = getone();
    }
    if (dist_str[0] = = 'q' || dist_str[0] = = 'Q')
    {
        fflush(fpw);
        fclose(fpw);
        system("cls");
        return(-1);
    }
    if (! (dist_str[0] = = 'y' || dist_str[0] = = 'Y'))
    {
        system("cls");
        break;
    }
} else {

printf("Enter the new error \% \% ");
getline(dist_str);
i = atoi(dist_str);
no_errors = 0;
system("cls");
if (i < 0 || i > 100)
    printf("INVALID ERROR RATE INPUT\n ");
else
{
    if (i == 0)
        no_errors = 1;
    else
        if (i == 100)
            error_rate = 1;
        else
            if (i >= 50)
                error_rate = 2;
            else
                if (i >= 33)
                    error_rate = 3;
                else
                    if (i >= 25)
                        error_rate = 4;
                    else
                        if (i >= 20)
                            error_rate = 5;
                        else
                            if (i >= 16)
                                error_rate = 6;
                            else
                                if (i >= 14)
                                    error_rate = 7;
                                else
                                    if (i >= 12)
error_rate = 8;
else
    error_rate = 100/i;

6.10.2 Set_sd

The set_sd function causes the network distance variable to be sorted for each node. This comes into play when the network is shutting down the individual ports.

/* This function sorts the network distances and sets the distance matrix */
set_sd()
{
    int i,j,k;

    for (i = 0; i < 8; i++)
    {
        k = 1;
        for (j = 0; j < 8; j++)
        {
            sdmat[i][j][0] = dist_mat[i][j];
            sdmat[i][j][1] = k;
            k* = 2;
        }
    }

    for (i = 0; i < 8; i++)
6.10.3 Sort_sd

The sort_sd function performs the basic sort of the rows of the distance matrix in order to know the order of shutdown of each node.

```c
sort_sd(i)
int i;
{
    int temp,j,k;
    for (j = 0; j < 8; j++)
        for (k = j + 1; k < 8; k++)
            if (sdmat[i][j][0] > sdmat[i][k][0])
                {
                    temp = sdmat[i][j][0];
                    sdmat[i][j][0] = sdmat[i][k][0];
                    sdmat[i][k][0] = temp;
                    temp = sdmat[i][j][1];
                    sdmat[i][j][1] = sdmat[i][k][1];
                    sdmat[i][k][1] = temp;
                }
}
```

6.10.4 Port_parms

The function port_parms controls the user interface to set the data parameters such as baud rate, parity, etc.

```
/* This function sets the port parameters of baud, parity, stop bits and data bits. */
```
port_parms()
{
    char c;
    int i;
    while (1)
    {
        while (1)
        {
            
            system("cls");
            printf("Ports are set to:\n ");
            switch (model & 192)
            {
                case 128:
                    printf("1.5 stop bits\n ");
                    break;
                case 192:
                    printf("2 stop bits\n ");
                    break;
                case 64:
                    printf("1 stop bit\n ");
                    break;
            }
        if (mode1 & 16)
        {
            printf("parity enabled ");
            if (mode1 & 32)
                printf("even\n ");
            else
                printf("odd\n ");
        }
        else
        
    
}
```
printf("parity disabled\n ");
switch (mode1 & 12) {
    case 12:
        printf("8 bits\n ");
        break;
    case 8:
        printf("7 bits\n ");
        break;
    case 4:
        printf("6 bits\n ");
        break;
    case 0:
        printf("5 bits\n ");
        break;
}
printf("Modify (Y)es or (N)o ");
while (1)
    if (kbhit())
        break;
    if ((c = getch()) = = 'y' || c = = 'Y')
        {
            system("cls");
            i = 1;
            while (i)
            {
                printf("\n ");
                printf("\n\n\n (A) 1 stop bit\n\n (B) 1.5 stop bits\n\n (C) 2 stop bits\nEnter the choice: ");
                while (1)
                {
                    if (kbhit())
                    {
                        c = getch();
                    }
```
break;

switch (c)
{
    case 'a':
    case 'A':
        mode1& = 63;
        mode1| = 64;
        i = 0;
        break;
    case 'b':
    case 'B':
        mode1& = 63;
        mode1| = 128;
        i = 0;
        break;
    case 'c':
    case 'C':
        mode1& = 63;
        mode1| = 192;
        i = 0;
        break;
    case '?':
        help(11);
        break;
    default:
        system("cls");
        printf("INVALID OPTIONn");
        break;
}
break;
}  
else 
   if (c == '?') 
      help(8); 
   else 
      return(1); 
} 
system("cls"); 
while (1) 
{
   printf("Enable parity (Y)es or (N)o "); 
   while (1) 
   { 
      if (kbhit())
         
      
      
   }
   if (c == 'Y' || c == 'y')
   { 
      mode1 &= 207;
      printf("(E)ven or (O)dd "); 
      while (1) 
      { 
         if (kbhit())
            
         
      }
   }
   if (c == 'E' || c == 'e')
   { 
      mode1 |= 48;
      
   }
break;

else
{
    if (c == 'o' || c == 'O')
    {
        mode1 |= 16;
        break;
    }
    else
    {
        system("cls");
        printf("INVALID OPTION\n");
    }
}

else if (c == '?')
    help(9);
else
    break;

}

system("cls");
i = 1;
while (i)
{
    printf("Enter number of data bits 5, 6, 7, 8 ");
    while (1)
    {
        if (kbhit())
        {
            c = getch();
            break;
        }
    }
mode1 & = 243;
switch (c)
{
    case '5':
        i = 0;
    case '6':
        i = 0;
        mode1| = 4;
        break;
    case '7':
        i = 0;
        mode1| = 8;
        break;
    case '8':
        i = 0;
        mode1| = 12;
        break;
    case '?':
        help(10);
        break;
    default:
        system("cls");
        printf("INVALID OPTION\n");
        break;
}
6.11 Include.h

The file include.h contains the declarations of the external variables as well as the compiler include directives for the system include files and the defines.h file. It is included into all source files except for netsim.c.

.CENTER

INCLUDE.H

Network Simulator

Written by: Chris Midgley
Date written: 10/6/87
Date last revision: 5/11/88

This software is written as the controlling software for a project to design a local area network simulator. This simulator was designed as part of the requirements for the Master of Engineering Degree in the Computer Engineering Department at the Rochester Institute of Technology.

#include <stdio.h>
#include <string.h>
#include <conio.h>
#include "defines.h"

extern FILE *fpw,*fopen();
extern char filename[32],option[32];
extern unsigned char mode1,mode2;
extern int no__errors;
extern int baud_val;baud_rate,error_rate,smat[8][8][2],dist_mat[8][8][8];
6.12 Defines.h

The defines.h file contains all of the definitions of program constants such as port addresses and timer and baud setup values.

/lib/defines.h

FILE DEFINES.H

Network Simulator

Written by: Chris Midgley
Date written: 10/6/87
Date last revision: 5/11/88

This software is written as the controlling software for a project to design a local area network simulator. This simulator was designed as part of the requirements for the Master of Engineering Degree in the Computer Engineering Department at the Rochester Institute of Technology.

#define PORTA 0x318
#define CPORTA 0x319
#define PORT1 0x30E
#define PORT2 0x30C
#define PORT3 0x30A
#define PORT4 0x308
#define PORT5 0x306
#define PORT6 0x304
#define PORT7 0x302
#define PORT8 0x300
#define CPORT1 0x30F
#define CPORT2 0x30D
#define CPORT3 0x30B

97
#define CPORT4 0x309
#define CPORT5 0x307
#define CPORT6 0x305
#define CPORT7 0x303
#define CPORT8 0x301
#define RTS_PORT 0x31E
#define AVAIL_PORT 0x31C
#define BAUD_PORT 0x31A
#define MODE1 0xCA
#define MODE2 0x35
#define TIM_CONT 0x43
#define TIM_MODE 0xBE
#define TIM_P2 0x42
#define TIM_LOW 169
#define TIM_HIGH 4
#define TIM_READ 0x61
#define TIM_WR 0x61
#define TIM_START 0x31
#define ERROR_RATE 10
#define BAUD_RATE 4800
#define NODE_MIN 10
#define B1200R 0x4C
#define B1200 0x8C
#define B2400R 0x4A
#define B2400 0x8A
#define B4800R 0x41
#define B4800 0x81
#define B9600R 0x45
#define B9600 0x85
#define PORT_RESET 0x40
6.13 Proj

The file Proj is used by the make program to determine the interdependencies in the source code. It determines which files require recompile or relink when a source file is edited.

NETSIM.OBJ : NETSIM.C
MSC NETSIM.C ;

INIT__B.OBJ : INIT__B.C INCLUDE.H DEFINES.H
MSC INIT__B.C ;

PFFILE.OBJ : PFFILE.C INCLUDE.H DEFINES.H
MSC PFFILE.C ;

NET__SUP.OBJ : NET__SUP.C INCLUDE.H DEFINES.H
MSC NET__SUP.C ;

PHEAD.OBJ : PHEAD.C INCLUDE.H DEFINES.H
MSC PHEAD.C ;

MENU.OBJ : MENU.C INCLUDE.H DEFINES.H
MSC MENU.C ;

S__PAR.OBJ : S__PAR.C INCLUDE.H DEFINES.H
MSC S__PAR.C ;

SETMAT.OBJ : SETMAT.C INCLUDE.H DEFINES.H
MSC SETMAT.C ;

MISC.OBJ : MISC.C INCLUDE.H DEFINES.H
MSC MISC.C ;
RNET.OBJ : RNET.C INCLUDE.H DEFINES.H
MSC RNET.C;

NETSIM.EXE : NETSIM.OBJ PHEAD.OBJ MENU.OBJ S__PAR.OBJ SETMAT.OBJ
MISC.OBJ RNET.OBJ INIT__B.OBJ PFILE.OBJ NET__SUP.OBJ
MSCL NETSIM.OBJ PHEAD.OBJ MENU.OBJ S__PAR.OBJ SETMAT.OBJ
MISC.OBJ RNET.OBJ INIT__B.OBJ PFILE.OBJ NET__SUP.OBJ;
6.14 Help Files

The help files for the network simulator are kept in the current directory. They are in files whose names have the format NETSIMx.HLP where the x is the number of the help message. The system has 9 help files numbered 0, 1, 2, 3, 4, 5, 8, 9, 10 and 11. Note each help file will displayed on it's own page to simulate the console screen of the PC.

6.14.1 Netsim0.hlp

The netsim0.hlp file is displayed when you request help about entering a filename.

Enter File Name

The file name that you choose must conform to DOS specs.

Enter the name of your file with no extension.

A .dat will be appended to your name to get the complete name.
6.14.2 Netsim1.hlp

The netsim1.hlp file is displayed when you request help about how to handle a file that already exists.

Handling Existing Files

The following are the available choices:

a append to the existing file no loss of previous data

o overwrite file all existing data in that file is erased

r rename, use a new name for your data file

e exit system

? display this message
6.14.3 Netsim2.hlp

The netsim2.hlp file is displayed when you request help about The optiona in the main menu of the simulator.

This is the main menu for the simulator. There are five options to choose from:

'S' To set up the network parameters.

'R' To run the network.

'P' To print the current data file to a printer or to display it on the screen.

'E' To exit the program and return to DOS.

'?' To display this message.
6.14.4 Netsim3.hlp

The netsim3.hlp file is displayed when you request help about setting up the network topology.

*Each node of the network can be set at a location up to 500 meters from node 0.*

*Each node must also be a minimum of 10 from any other node.*

*This parameter is used to determine the delay in getting a message from node to node.*
6.14.5 NetSim4.hlp

The NetSim4.hlp file is displayed when you request help about the valid error rates that can be programmed into the simulator.

The following are valid error rates:

100  50  33  25  20  16  14
12   11  10   9   8   7   6
  5   4   3   2   1

Any inputs between two valid values will be truncated to the lower value.
6.14.6 Netsim5.hlp

The netsim5.hlp file is displayed when you request help about the valid baud rates that can be utilized in the simulator.

*The following are valid baud rates:*

| 9600 | 4800 | 2400 | 1200 |
6.14.7  Netsim8.hlp

The netsim8.hlp file is displayed when you request help about the data format which can be utilized for communicating between simulator and nodes.

You may modify the following parameters to the network:

- Number of stop bits  1, 1.5, 2

- Parity enabled

- Odd or Even parity

- Number of data bits  5, 6, 7, 8
6.14.8 Netsim9.hlp

The netsim9.hlp file is displayed when you request help about the available parity options for communicating between simulator and nodes.

Parity may be enabled or disabled. If you choose to enable parity you will be prompted for Odd or Even.
6.14.9 Netsim10.hlp

The netsim10.hlp file is displayed when you request help about the valid number of data bits which may be used for communicating between the simulator and nodes.

*You may choose from 5, 6, 7, or 8 data bits.*
6.14.10  Netsim11.hlp

The netsim11.hlp file is displayed when you request help about the valid number of stop bits that may be used to communicate between the simulator and the nodes.

You may choose from 1, 1.5, or 2 stop bits to be sent at

the end of each byte.
Appendix A

References

Digital Equipment Corporation Maynard, Ma., *Digital's Networks: An Architecture With A Future*, 1984

Digital Equipment Corporation Maynard, Ma., *Introduction to Local Area Networks*, 1984


Technical Manuals

Microsoft C Compiler Language Reference, Microsoft Corporation, 1986

Microsoft C Compiler Run-Time Library Reference, Microsoft Corporation, 1986

Microsoft C Compiler Users Guide, Microsoft Corporation, 1986


Programmable Crystal Oscillator, PXO Series 0.002 to 2 Mhz, Data Sheet Statek Corporation, Orange, Ca.


Telecommunications Device Data, Motorola Inc., 1985

The TTL Data Book, Volume 2 Texas Instruments Incorporated, 1984

TTL Data Manual, Signetics, 1987
## Glossary of Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud Rate</td>
<td>The number of bits per second which are being transferred.</td>
</tr>
<tr>
<td>Bit</td>
<td>The smallest entity in computer nomenclature. It signifies the value of 0 or 1</td>
</tr>
<tr>
<td>Byte</td>
<td>A group of 8 bits.</td>
</tr>
<tr>
<td>Check Sum</td>
<td>A value derived by some method of adding the data. In this network the method is left up to the user.</td>
</tr>
<tr>
<td>COMx</td>
<td>A serial port on the IBM PC/AT or compatible computer. The x is an integer 1 or 2 which designates which serial port.</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear To Send line in the RS-232C interface.</td>
</tr>
<tr>
<td>DB-25</td>
<td>A style of connector used in computer interfacing. It has a distinctive &quot;D&quot; shape, and in this case has 25 pins.</td>
</tr>
<tr>
<td>Destination Address</td>
<td>The port location of the node to which the packet is being sent.</td>
</tr>
<tr>
<td>Expansion Card</td>
<td>A printed (or wire wrapped) wiring board designed to operate in the I/O slot of an IBM PC/AT or Compatible Computer.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Header</td>
<td>A group of data bytes which signify the start of a packet. In this network the header is 2 bytes with the value 55 Hex.</td>
</tr>
<tr>
<td>IBM PC/AT</td>
<td>A Personal Computer manufactured by the International Business Machine Corporation.</td>
</tr>
<tr>
<td>LPTx</td>
<td>A parallel port on the IBM PC/AT or compatible computer. The x is an integer 1 or 2 which designates which serial port.</td>
</tr>
<tr>
<td>MS-DOS</td>
<td>A Disk Operating System written by Microsoft inc.</td>
</tr>
<tr>
<td>MS-DOS System Call</td>
<td>The change of program control to a program which is part of the operating system.</td>
</tr>
<tr>
<td>ORed</td>
<td>The process of activating if any one or more input is active</td>
</tr>
<tr>
<td>Packet</td>
<td>A group of data bytes assembled with a header and trailer in order to be sent over a Local Area Network</td>
</tr>
<tr>
<td>Parity</td>
<td>The Idea of placing a check sum within the data byte being transfered.</td>
</tr>
<tr>
<td>Physical Interface</td>
<td>The definition or the actual hardware connections of the network.</td>
</tr>
<tr>
<td>RS-232C</td>
<td>An interface standard recommended and published by the Electronic Industries Association.</td>
</tr>
<tr>
<td>RxD</td>
<td>Receive data line in the RS-232C interface.</td>
</tr>
<tr>
<td>RTS</td>
<td>Request To Send line in the RS-232C interface.</td>
</tr>
<tr>
<td><strong>Source Address</strong></td>
<td>The port location which the sending node is connected.</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td><strong>TxD</strong></td>
<td>Transmit data line in the RS-232C interface.</td>
</tr>
</tbody>
</table>