SPACE LAB SYSTEM ANALYSIS

INTERIM FINAL REPORT
OCTOBER 1987 - OCTOBER 1988

Submitted by:

T. B. Rives, Co-Principal Investigator
F. M. Ingels, Co-Principal Investigator

Mississippi State University
Electrical Engineering Department
Mississippi State, MS 39762
(601) 325-3912

Submitted to:

NASA/MSFC-EB32/JOBE
NAS8-36717
(205) 544-3555

(NASA-CR-183507) SPACE LAB SYSTEM ANALYSIS Interim Final Report, Oct. 1987 - Oct. 1988 (Mississippi State Univ.) 222 p CSCL 22B

N89-11792

Unclas

G3/18 0169994

OVERVIEW OF REPORT

An analytical analysis of the data flow within the SRB Automated Booster Assembly Checkout System has been conducted. This analysis which is Task 1 of the work statement is presented in Section 4.0 of this report. The results are summarized in Section 4.2 and indicate that the ABACS system will be able to carry a worst case load with only one Ethernet Local Area Network.

The ABACS data system model has been updated by a newly-coded PASCAL-based simulation program which has been installed on the HOSC VAX system. This model is described and documented in Sections 1.0, 2.0, 3.0, 6.0 and in the appendices of this report. This model is Task 2 of the work statement.

The Sectionn 5.0 of this report offer suggestions to fine tune the performance of the ETHERNET interconnection network, Task 3 of the work statement.

Suggestions for using the Nutcracker by Excelan to trace itinerate packets which appear on the network from time to time have been offered in discussions with the HOSC personnel, Task 4 of the statement of work.

Several visits to the HOSC facility were made during the course of the contract to install and demonstrate the simulation model. In addition, several visits were made to USBI BPC in Slidell, LA to discuss the ABACS system operation.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION TO ABACS

ABACS (Automated Booster Assembly Checkout System) is a system designed to facilitate the testing of the solid rocket booster (SRB) by providing an automatic checkout capability for specific components of the SRB. The National Aeronautics and Space Administration uses the SRB to place the United States Space Shuttle into orbit. The facility housing the installed ABACS system is the Assembly Refurbishment Facility located at the Kennedy Space Center, Cape Canaveral, Florida.

The actual ABACS system is a hardware/software system designed and built by the Slidell, LA section of the Booster Production Company Incorporated. The equipment will be used to test the first Space Shuttle mission following the Space Shuttle explosion of 1986.

Within any communication system such as ABACS, the possibility exists that data flow bottlenecks will occur with the result that the system is no longer able to handle the information transfer. This bottleneck could possibly occur at any level in ABACS: at the VAX Control Computer level, at the Station Controller level, or at the Local Area Network (LAN) level. The primary purpose of this research is to analyze the ABACS system and determine that the Ethernet LAN is capable of carrying the computational load in the nominal and in the worst case configurations. This work will be performed using two techniques: an analytical analysis of the data flow within the system and a simulation program which predicts the behavior of the system under differing configurations. Figure 1.0 illustrates the top-level

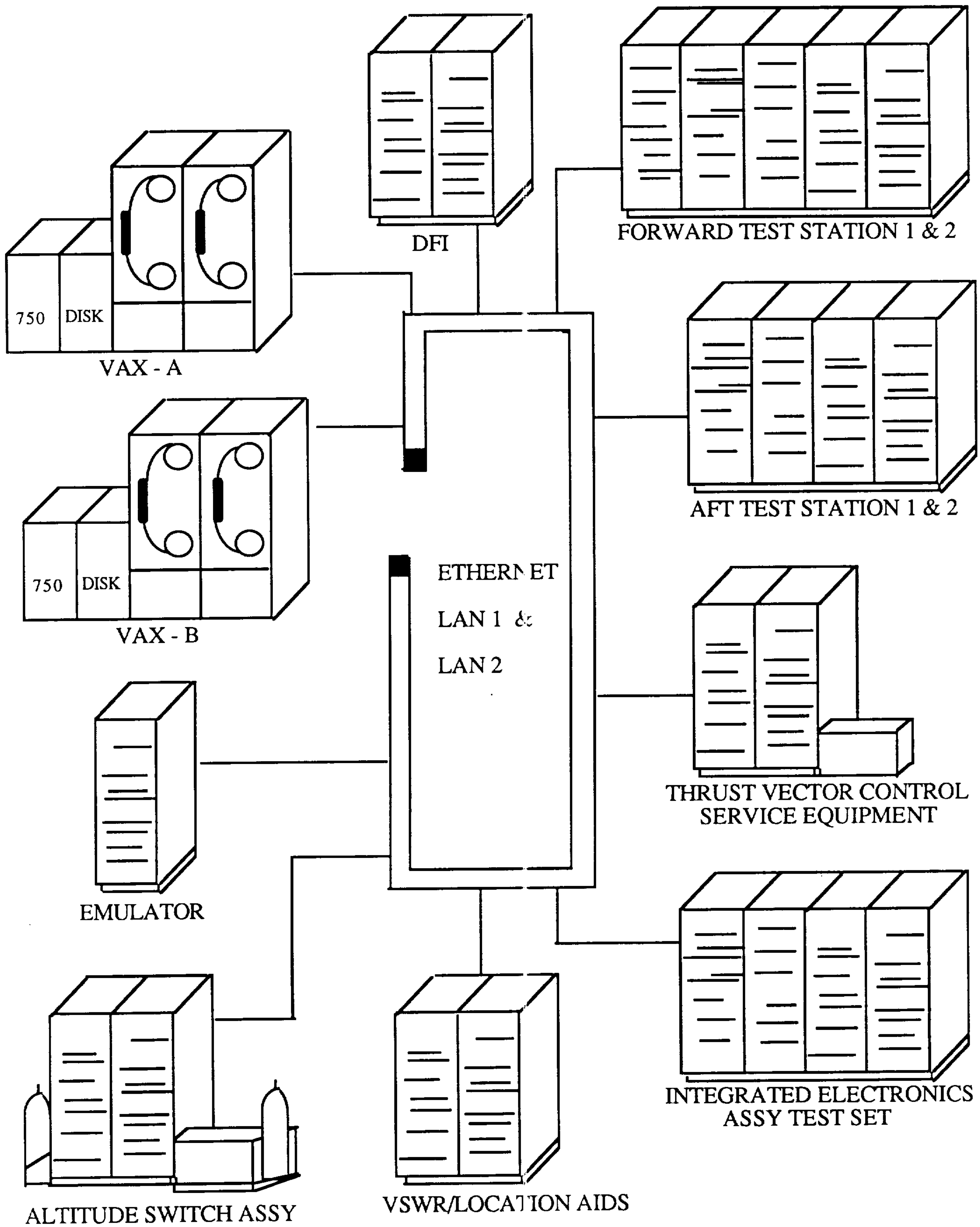


FIGURE 1.0 ABACS HARDWARE CONFIGURATION

ABACS system diagram. Figure 1.1 shows the physical system installation with the distance between Ethernet devices indicated in feet.

1.1 ABACS System Configuration.

The ABACS equipment consists of nine major subsystems and many supplementary devices. Only the major subsystems and the Ethernet LAN which pertain to the analysis of the LAN data communication will be documented in this research. For very detailed descriptions of each device and other elements of the ABACS system such as the Peripheral bus, the reader should consult the Operation and Maintenance Manuals provided by USBI BPC for each device. The primary system components are the Control Computer Systems: two VAX 11/750 computers (VAX), and the Station Controllers: two Forward Test Stations (FWD), two AFT Test Stations (AFT), the Thrust Vector Controller (TVC), the Altitude Switch Assembly/ Sensor Test Set (ASA), the VSWR/ Location Aids Test Set (VSWR/LA), the Integrated Electronics Assembly Test Set (IEA), the Development Flight Instrumentation Test Set (DFI), and the Emulator (EMU).

Figure 1.2 shows a functional overview of the ABACS system. The primary activity of the system is the VAX with station controller communication over the Ethernet bus. The actual system information flow is displayed in Figure 1.3. The information passed over Ethernet between the VAX and the station controllers include ATLAS (Abbreviated

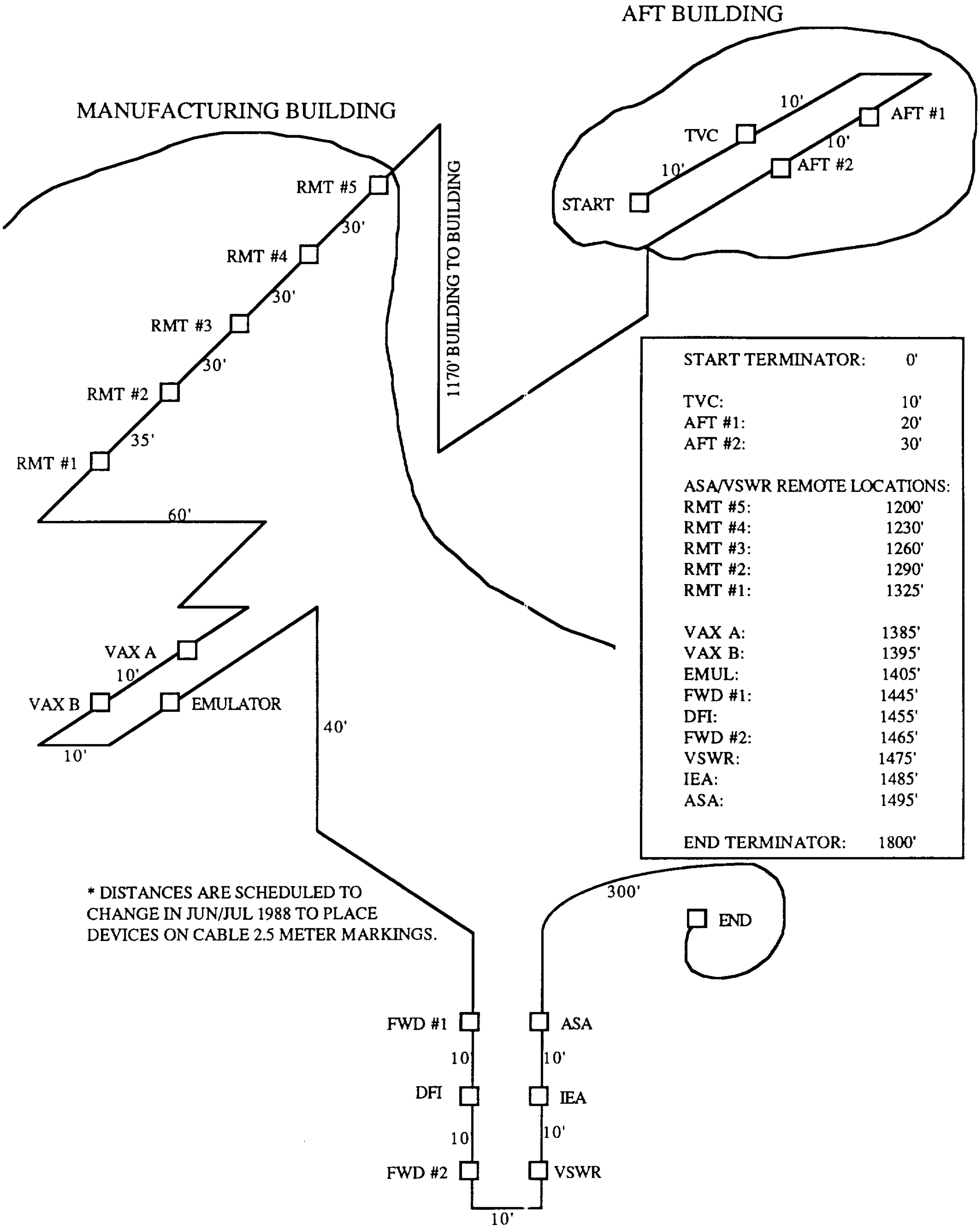


FIGURE 1.1 DISTANCES BETWEEN ETHERNET NODES AT KSC ESTIMATED IN FEET. DATED 27 JAN 88.

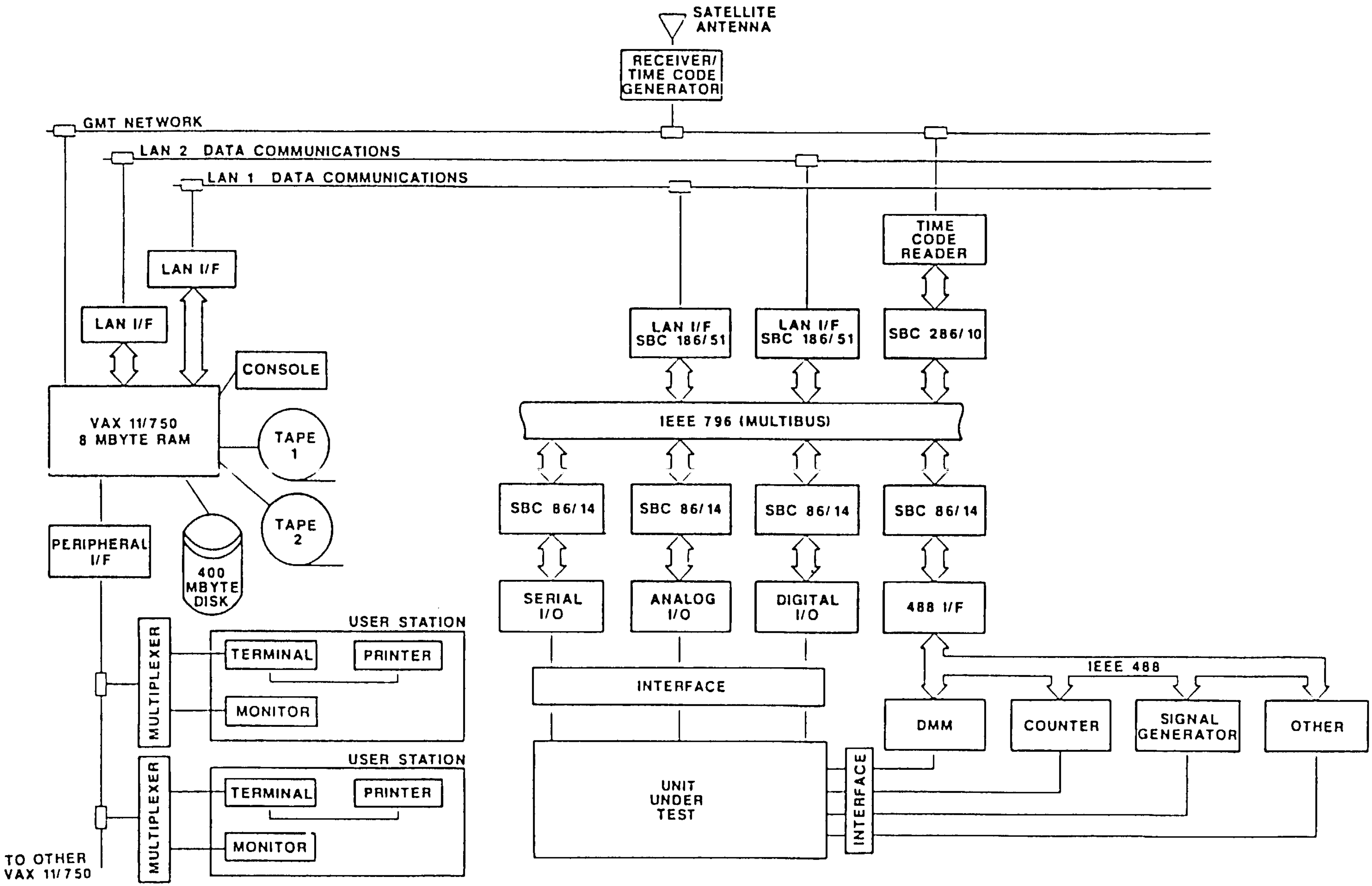


FIGURE 1.2 ABACS FUNCTIONAL CONCEPT

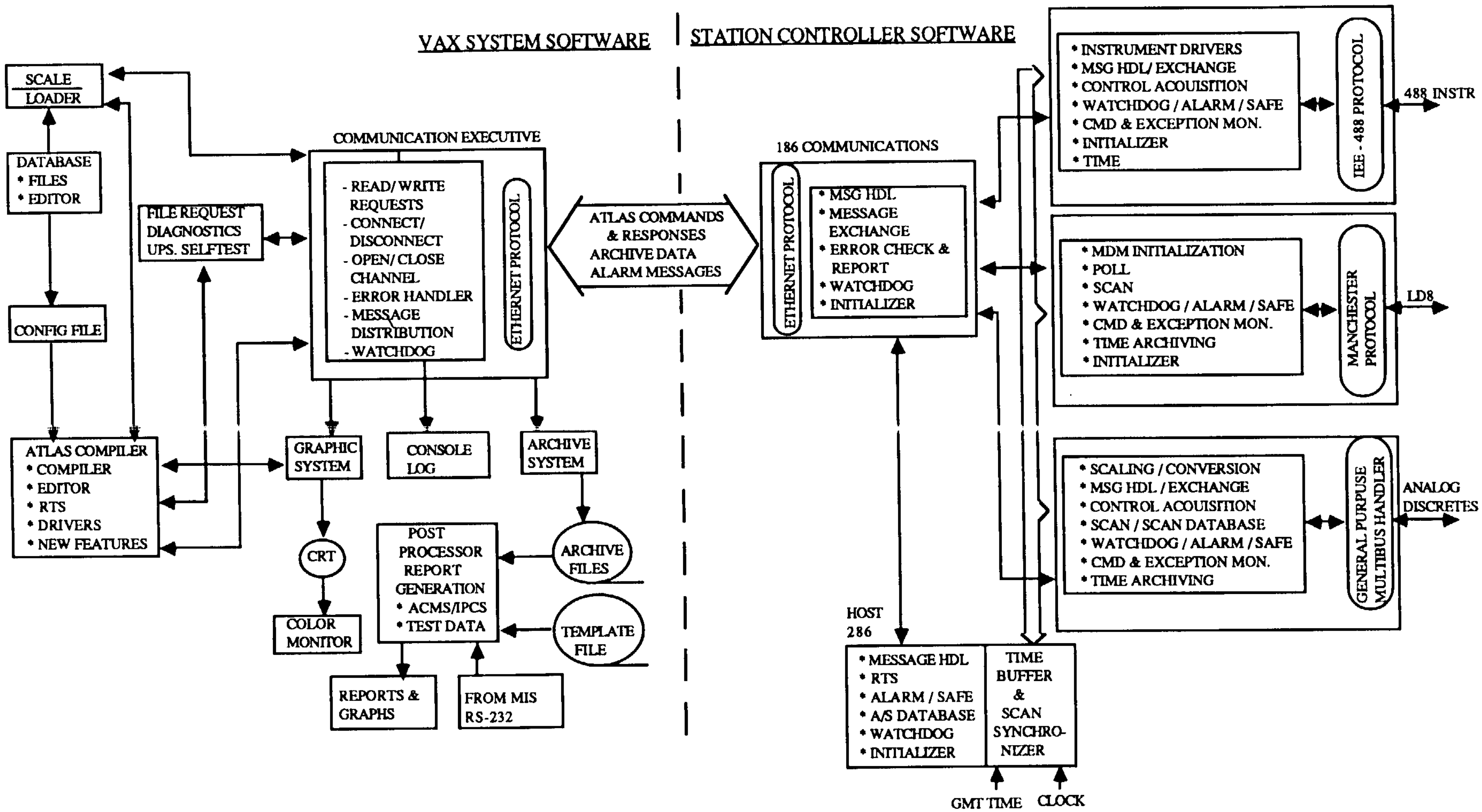


FIGURE 1.3 GENERAL ABACS SYSTEM SOFTWARE FLOW

Test Language for All Systems) commands with responses, archive data, and alarm messages.

1.1.1 VAX Computers.

The dual VAX 11/750 computers provide control for the overall system. The VAX resident runtime system initiates test execution, provides control, allows test station reporting, records archive information, and handles graphics. The actual hardware of the system will be introduced here in terms of the architecture and components of the control computer.

The ABACS system includes a computer assembly, a printer/plotter assembly, a VAX console terminal assembly, a dual mag tape assembly, and an Ethernet controller unit. Figure 1.4 displays the primary hardware of the ABACS control computer. A functional block diagram is shown in Figure 1.5.

The computer assembly is divided into two sections: the expansion unit and the CPU unit. The expansion unit houses the system disk which has a 475 Mbyte unformatted capacity. The expansion unit is located next to the CPU unit of the computer assembly. The CPU of each Control Computer is a Digital Equipment Corporation VAX 11/750.

The printer/plotter assembly includes a dot matrix printer with a maximum rate of 600 lines per minute. It has a resolution of 400

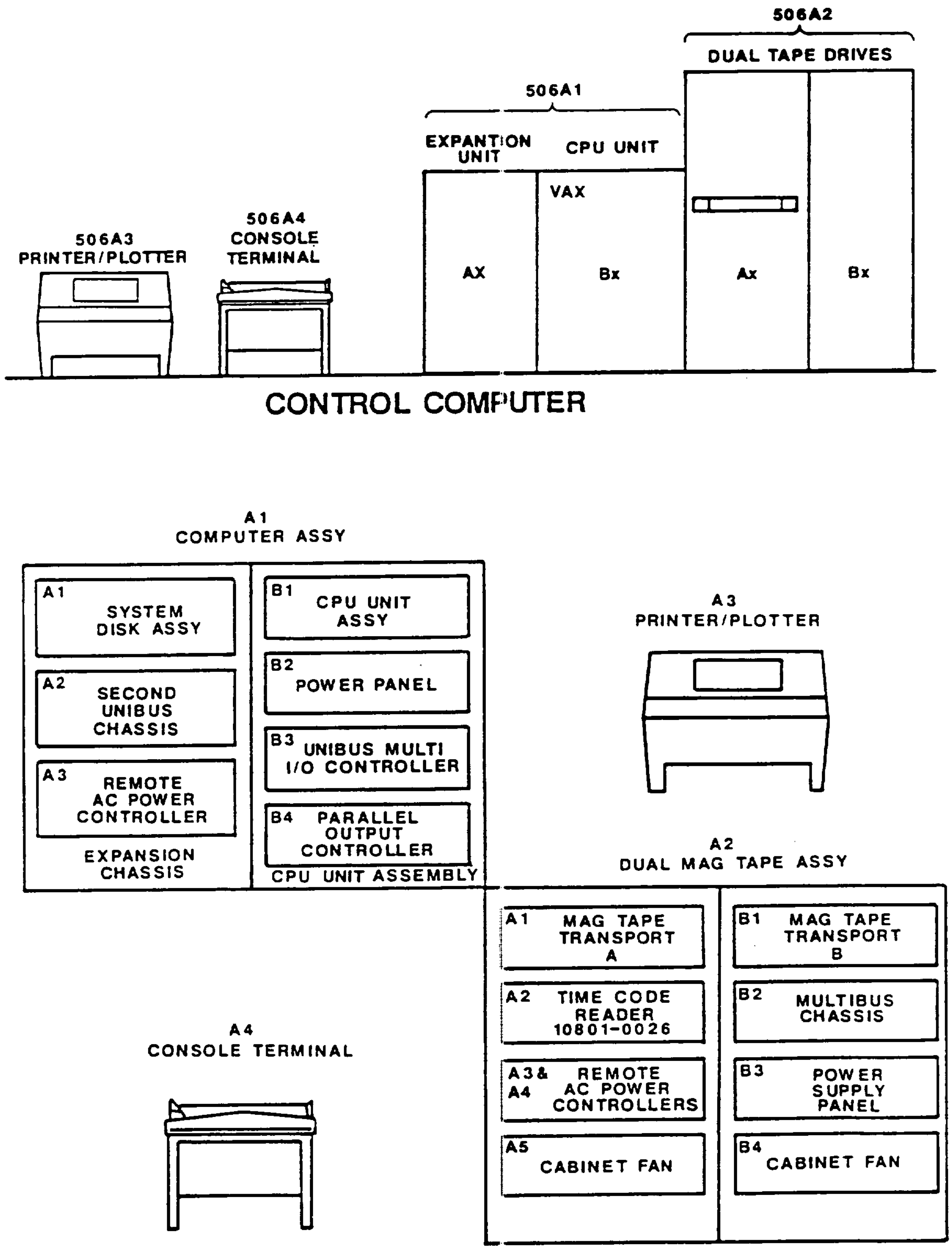
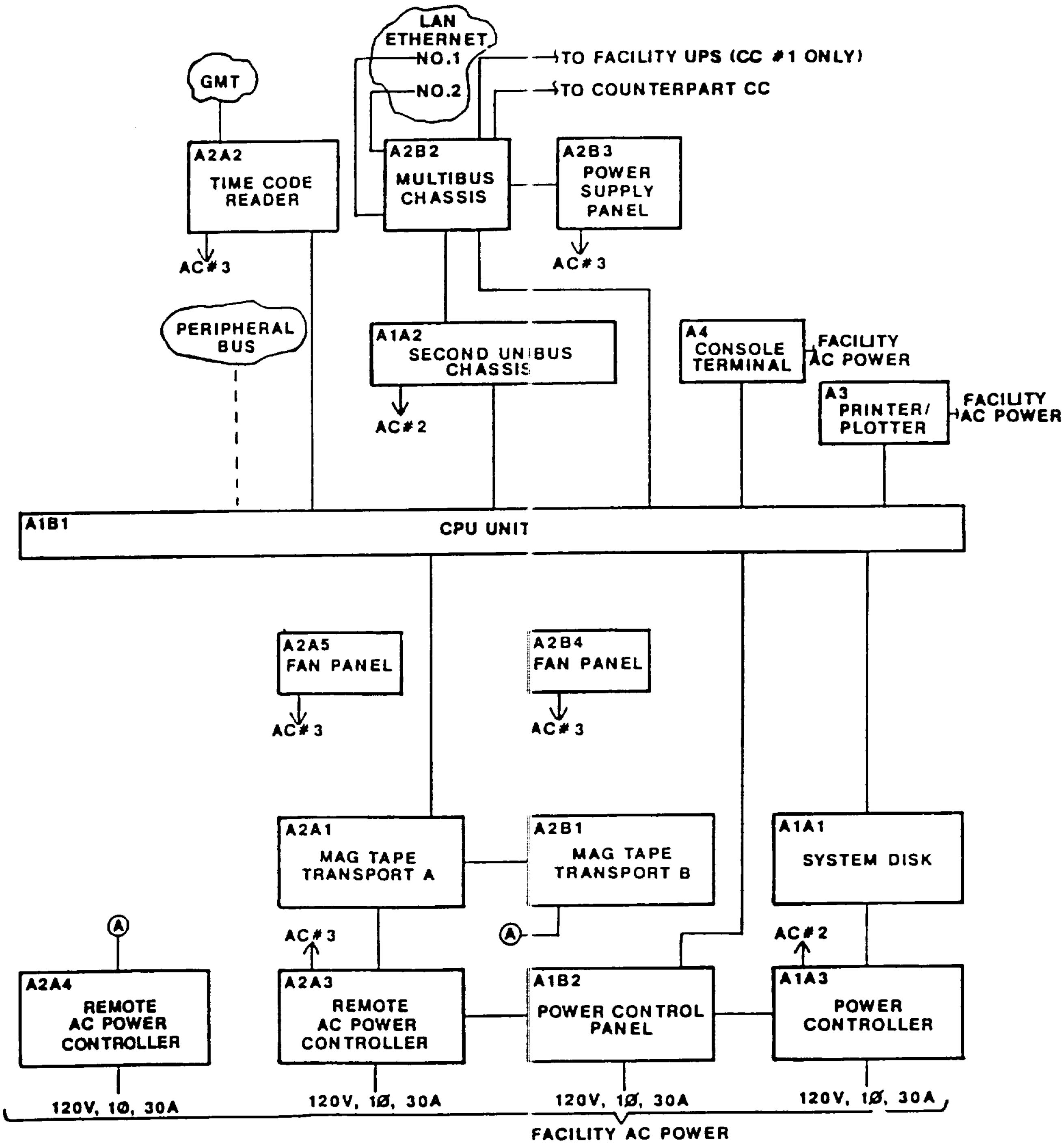


FIGURE 1.4 CONTROL COMPUTER HARDWARE



NOTES: 1. NOT AN EXACT DEPICTION OF CABLES SEE CABLE INTERCONNECT DIAGRAM 10808-0003.

2. ONE OF TWO IDENTICAL SYSTEMS SHOWN.

FIGURE 1.5 BLOCK DIAGRAM - CONTROL COMPUTER dots/inch horizontally and 288 dots/inch vertically.

The VAX console terminal is a LA-120 DECWRITER terminal. It has a wide range of features. The LA-120 is basically a typewriter-printer which has a dot matrix printer with a maximum print rate of 180 characters per second. Communication with the host computer is serial asynchronous and operates at 9600 baud.

The dual mag tape assembly uses two rack enclosures. Each rack houses a mag tape assembly with each using a standard ABACS time code reader. The Kennedy model 9400 Tri-density tape dive has three format modes: 6250 BPI group coded mode - nominal data transfer rate of 280-312 Kbytes/sec, 1600 BPI phase encoding mode - nominal data rate of 120 kbytes/sec, and 800 BPI non-return-to-zero mode - nominal data rate of 60 Kbytes/sec. Tape Drive B also houses the multibus chassis which provides Ethernet communication capabilities. Figure 1.6 shows the overall bus interface for the control computer and how the multibus chassis is interfaced to the CPU unit.

The multibus chassis which is also called the controller in the VAX contains two Intel Ethernet control cards (iSBC 186/51), two DR11W emulators (Ikon 10077), and one Intel Ethernet arbitration computer (iSBC 86/14). Appendix I includes the attributes of the ABACS board level hardware.

Basically, the control card accepts and interprets the data received

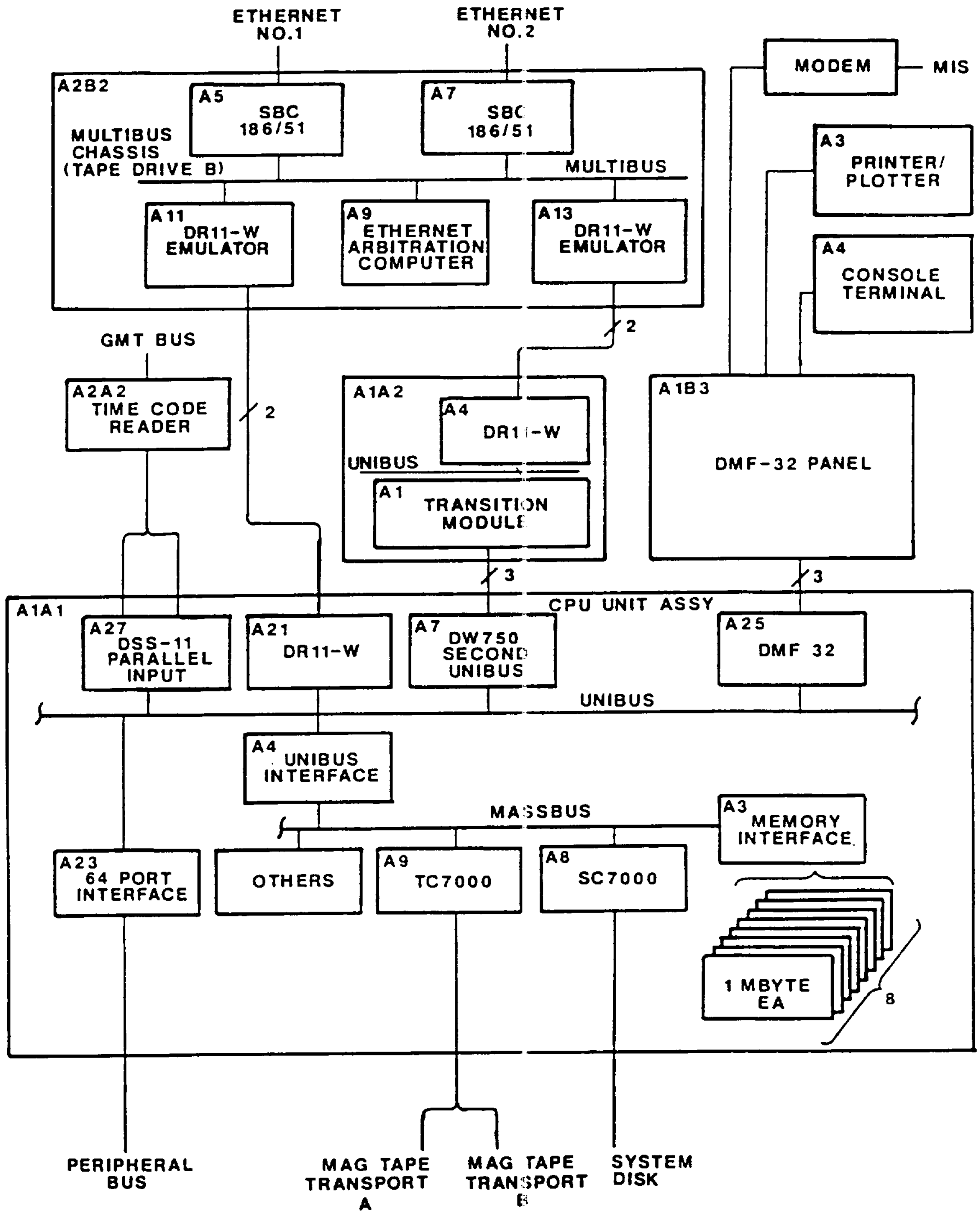


FIGURE 1.6 BUS INTERFACE - CONTROL COMPUTER

from the two Ethernet LAN's. If the information is to be passed to the to the VAX, the interface between the VAX and the controller utilizes two DR11W Digital Equipment Corporation interface modules in conjunction with two IKON DR11W compatible multibus boards. One DR11W board is used for transmit and the other for receive. The processes between the controller and the VAX handshake each operation. The DR11W is a general purpose, Unibus to DR11W compatible parallel, direct memory access (DMA) device with nominal transfer rate of 400 Kwords (16 bit) per second. The information is placed in memory using the DR11W DMA controller so that the ATLAS process may access it or the information is placed in archive storage. Figure 1.7 shows the information transfer within the VAX computer, and Appendix I includes a detailed description of the VAX to controller hardware operations.

1.1.2 Control Stations.

The control stations physically attach to portions of the SRB which are to be tested. This portion of the SRB is called the unit under test as shown in Figure 1.8. The station receives a command from the VAX control computer to perform some function. The station executes the command and responds to the VAX. The station also records readings from the unit under test and buffers the data before sending the information to the VAX as archive data.

The control stations are similarly designed. Figure 1.9 displays the basic hardware of a control station. All station controllers utilize

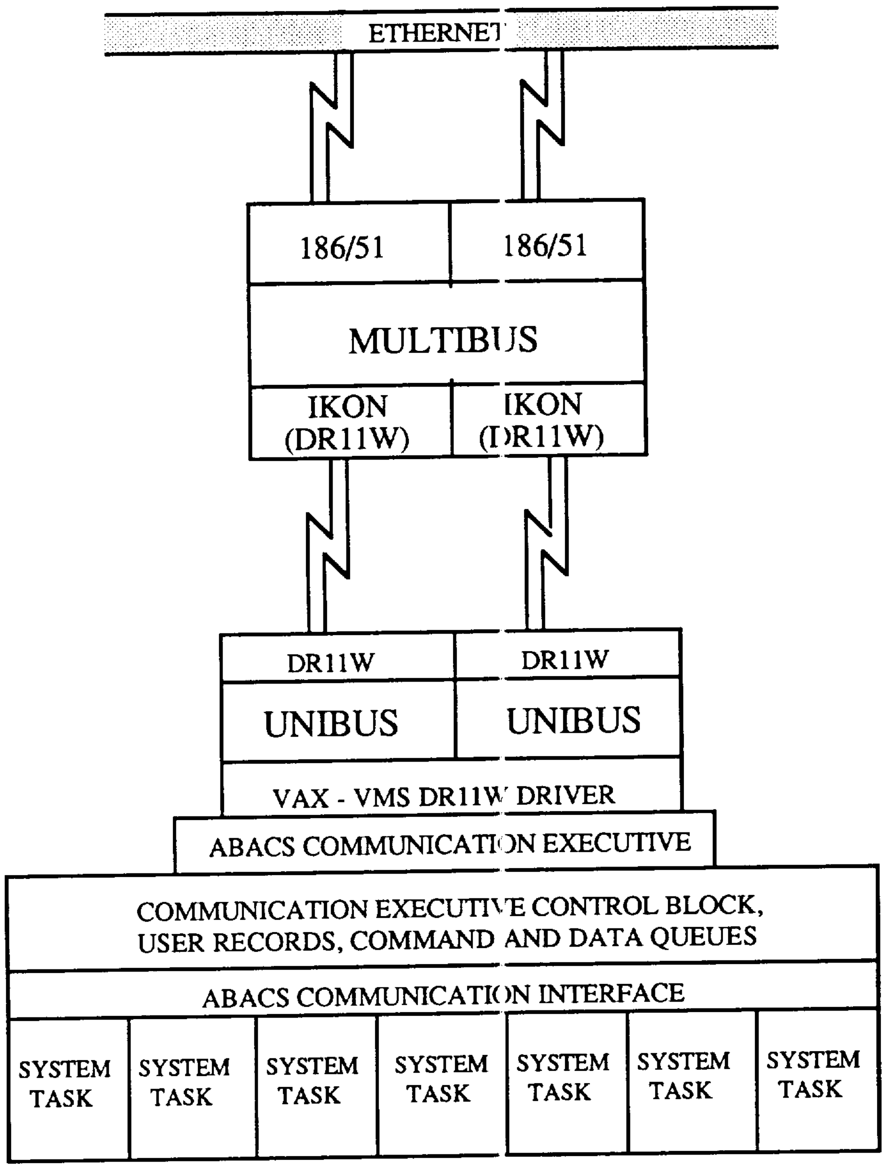


FIGURE 1.7 VAX - VMS ETHERNET COMMUNICATIONS

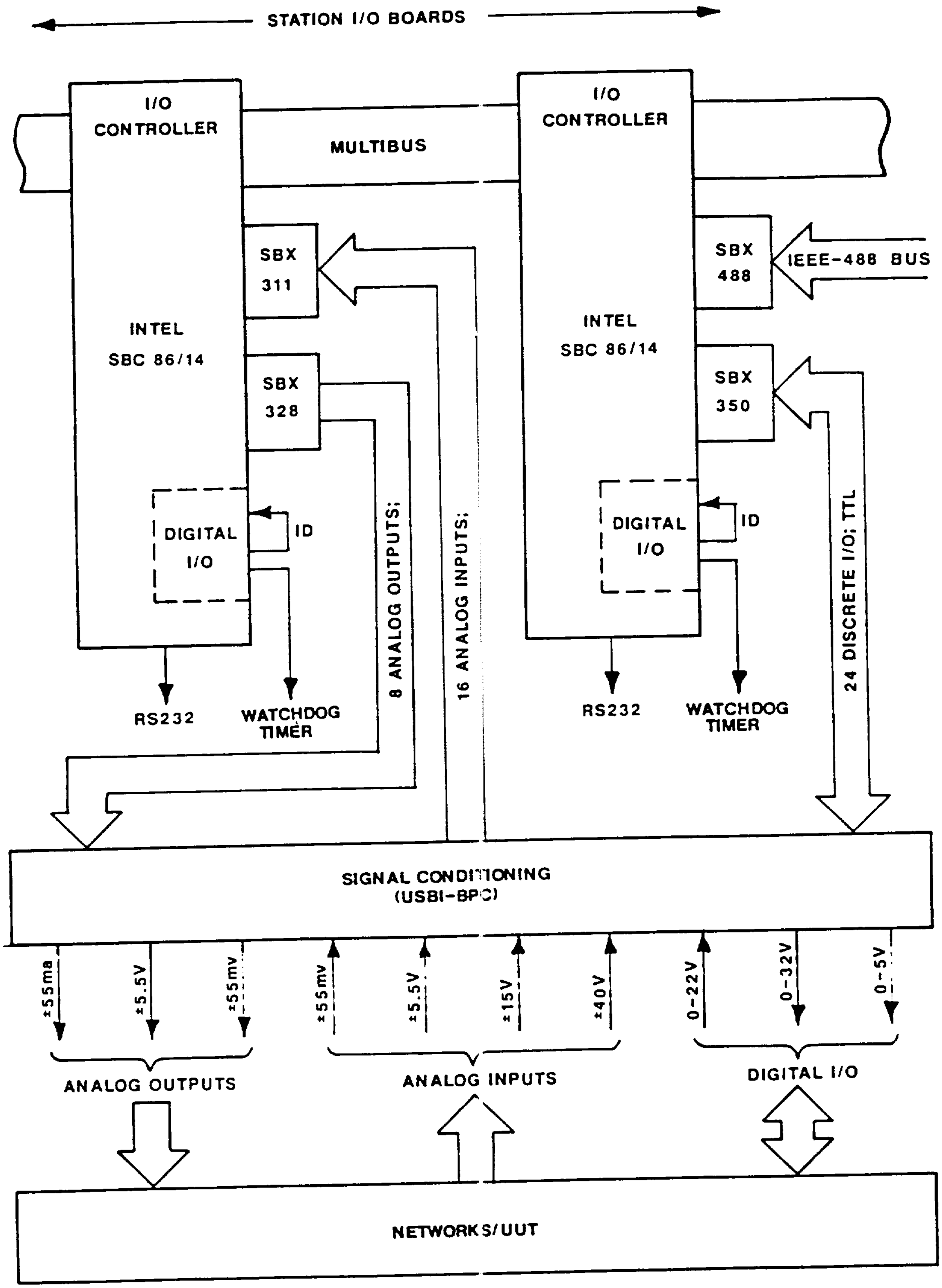


FIGURE 1.8 GENERAL CONTROLLER/ UNIT UNDER TEST INTERFACE

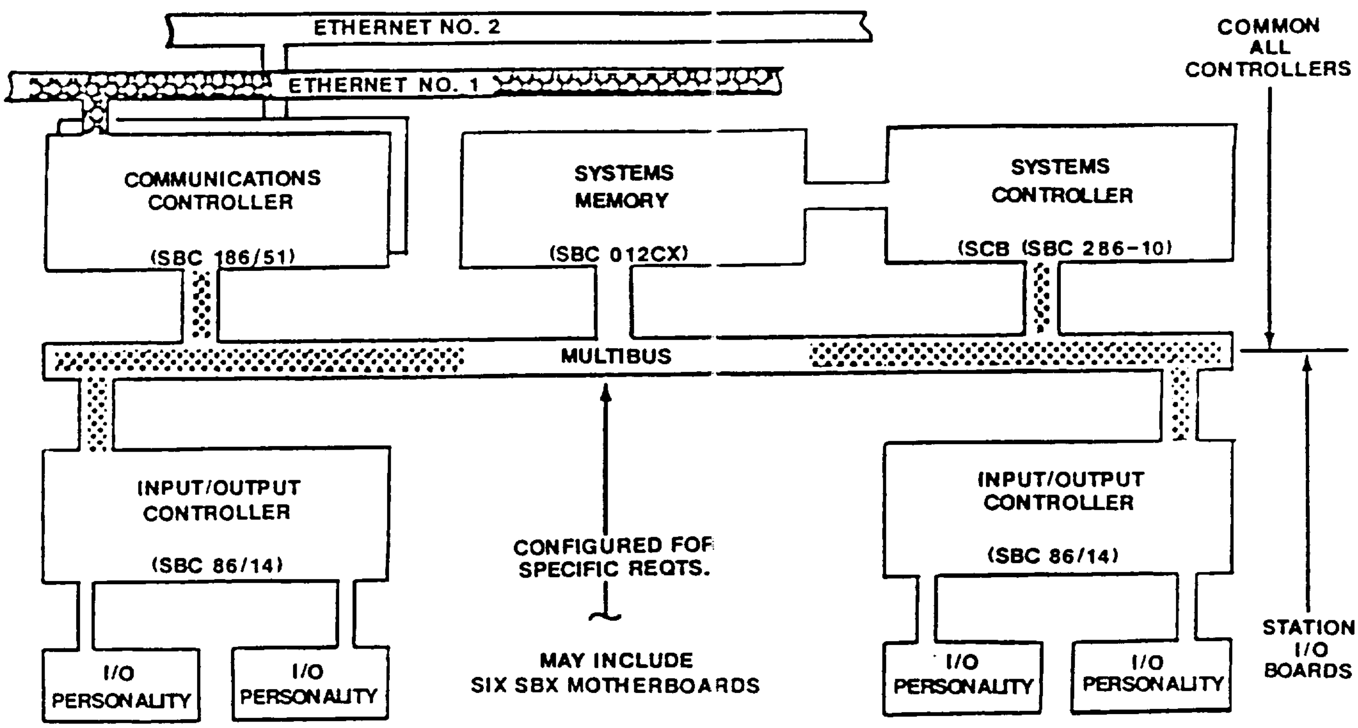


FIGURE 1.9 TYPICAL ABACS STATION CONTROLLER

two Intel Ethernet control cards (iSBC 186/51), one Intel Ethernet systems controller (iSBC 286/14), and a 512 Kbyte RAM memory board (iSBC 012CX). The individual controllers are then configured with iSBC 86/14 boards to accommodate specific requirements.

The number of iSBC 86/14 boards which allow archiving for each controller is included: TVC - 4, AFT - 5, FWD - 7, DFI - 3, EMU - 7, IEA - 3, ASA - 0, and VSWR/LA - 0. The unit under test is sampled 25 times per second. In the worst case the iSBC 86/14 would be required to record each sampled input value as it changed (input values are scanned on a 40 milli-second cycle). Therefore, in the worst case the 1500 byte buffer would be transmitted to the VAX every 40 milli-seconds. These values were taken from information provided by USBI BPC. The tables which show the board configuration and input/output operations are included in Appendix I. An attempt to determine the number of inputs per board is recorded in this appendix. These values are estimated based on the data provided.

Appendix I includes the attributes of the primary ABACS board level hardware. In addition, this appendix includes a detailed description of the operations for station to station connection over Ethernet, messages sent over Ethernet, messages received over Ethernet, and station to station disconnect over Ethernet. For additional hardware configuration information consult the USBI Operation and Maintenance Manual for the individual station.

1.1.3 ETHERNET Local Area Network.

The primary devices are interconnected by using dual Ethernet Local Area Network's (LAN). Each device attaches to both LAN's to allow a full duplex type communication link. One is used for transmitting information. The other is for receiving. Typically, both LAN's function as a coordinated pair. However, if one LAN is not operating properly, the other LAN is designed to maintain communication. It is noted here that both LAN's must be functioning properly before a station controller may be initialized to communicate with a control computer.

The devices attached to the Ethernet cable each have two iSBC 186/51 computer boards which interface the device to Ethernet. One computer board is used for each LAN. The VAX uses LAN 1 to transmit data which is received by the station controller. The station then transmits a response on LAN 2 which the VAX receives. If, however, a malfunction occurs in either LAN, then both the send and receive functions will be performed over the remaining LAN. Thus, the worst case condition for the system configuration would occur if one of the LAN's malfunctioned with data communications between all devices occurring at the maximum possible communications rate.

The ABACS Ethernet characteristics are summarized in the Operation and Maintenance Manual for the ABACS Complex including the Control Computer System:

1) Hardware: coaxial trunk cable, cable tap, transceiver, transceiver cable, and controller.

2) 50 Ohm IEEE 802.3 coaxial terminated trunk provides 1640 foot segments without repeaters.

3) Up to 100 multidrop nodes per segment (2.5 meter minimum spacing).

4) Non-intrusive cable taps provide ease of transceiver installation.

5) Carrier Sense Multiple-Access with Collision Detection (CSMA/CD) contention techniques at transceiver.

6) Transceiver cables (up to 50 meters) provide power and interface signals from the controller board.

7) 10 Mbit per second data rate.

8) Manchester encoded with 32 bit CRC.

9) Message length from 64 to 1518 bytes (46 to 1500 byte data field)

1.2 ABACS Traffic Analysis.

The control computers communicate with the station controllers using the Ethernet LAN. This section describes the information flow between the attached devices.

1.2.1 VAX Communication.

According to the original USBI BPC design specifications, the VAX computer could only archive four message streams at any point in time. However, the installed VAX system can handle six archive streams with a maximum of seven controllers archiving information.

The ATLAS Test Operating System is the primary mode of operation when the VAX communicates with a station controller. In this mode commands are sent to the station controller for execution. The frequency at which commands are sent to the controller is completely random. The commands may be initiated by a VAX user or a defined test sequence may be run.

The VAX and station communication begins when the VAX sends initialization information to the station. The stations iSBC 286 computer then enables archiving. And the iSBC 86/14 boards for that station send the archive buffer (about 1500 bytes) to the VAX. The board associated with the MDM returns 2 buffers. The user would then select the ATLAS program to be run or write ATLAS code which commands the station to perform a certain test. If the user selects an ATLAS program to be run (for example, the self test account or ACO test account), the software may take several minutes to load for execution. The application menu is then displayed so that the user can select the tests to be executed in sequence or can select individual tests for execution.

A majority of the commands sent to the station do not require a response from that station. These commands are setup messages. An estimate of 60 to 70 percent of the commands require no response.

A sequential command is one in which several (about 20) commands are packed into a single Ethernet message. Since this operation occurs infrequently, only about one in fifty out-going messages from the VAX are sequential operations. Most of the commands within the sequential command are setup commands so that perhaps 5 to 10 of the commands will require a response from the controller. The controller receives the single Ethernet message and interprets the individual commands. In this manner some of the VAX over-head is reduced as is Ethernet message traffic. Timing constraints are taken into consideration for station execution of commands.

The VAX is estimated by USBI BPC to be able to send out a maximum of 100 K bytes of data per second. An estimated 90 K bytes of data per second is the amount any VAX can receive and adequately handle. The values given here reflect the amount of data which may be handled. The acknowledge messages, watchdog timer messages, and header information contained in an Ethernet message are excluded.

1.2.2 Control Station Communication.

The basic station controller communications include responses to VAX commands and archiving information sent to VAX. The average length of time it take a station to produce a response to a VAX command is about two milli-seconds. This includes receiving the message at the station, interpreting the command, performing the command, and having the response available for transmission.

The stations which do archiving have buffers on board the iSBC 86/14 in which archive information is stored. The stations which do not do archiving are the ASA, the VSWR/LA, and the IEA. The 86/14 records sampled values in a 1500 byte buffer if the reading falls outside a specified normal range. When the buffer is filled it is sent to the VAX for archiving. If the buffer has not filled within a defined period of time, the contents of the buffer are sent to the VAX anyway. There are five 1500 bytes buffers on each 86/14 board so that only in an extreme case would all the buffers be filled and information lost. However, this situation would have resulted in the VAX choke off before the buffers had filled so that the system would have crashed before the data loss occurred.

The VAX receiving the archive information can handle a new archive message about every three milli-seconds. There are, however, 120 buffers within the VAX which initially store the incoming information. If these buffers are filled and the VAX is unable to process the data before new packets arrive, then the station controllers are informed of this condition and are directed to discontinue transmitting. If the station is not allowed to transmit, then the five archiving buffers on each of the 86/14 station controller boards will also fill. The watchdog timers will time out if no communication occurs within the specified time frame. If this occurs, the VAX and station will be disconnected.

1.2.3 Watchdog Timer Operations.

The watchdog timer is a process used in ABACS to verify connectivity of a station. The communication occurs at the iSBC 186/51 controller level so that the VAX is not burdened with additional overhead. The VAX controller sends a message to the station controller at one second intervals. The station controller responds immediately or within four seconds. The watchdog timer message is not sent to the station if the VAX and that station exchanged messages within the second. Since connectivity is automatically verified in that case, the watchdog one second counter is reset.

1.2.4 TVC and AFT Communication.

The TVC and one AFT communicate with each other using the Ethernet bus. This is the only activity of the bus which allows traffic between devices other than the VAX and station controllers.

The message traffic has a one-to-one correspondence. When the TVC sends a message to the AFT, the AFT responds with a message back to the TVC. There are approximately thirty messages in a second sent out during TVC and AFT communication.

1.2.5 Other Device Operations.

The primary operation which causes additional loading of the Ethernet bus is an automatic acknowledge of any packet received. Each message sent out over Ethernet requires an acknowledge from the receiving station to verify receipt of the message. The acknowledgement is performed at the iSBC 186/51 controller level so that the VAX or station is not interrupted from normal processing. The packet size is the smallest Ethernet packet size, 72 bytes.

Due to the ABACS hardware configuration, it is possible for the secondary VAX to communicate with other stations while tests are being conducted between VAX I and particular stations. The primary application of this activity is software testing using the Emulator as the receiving station with VAX II as the control computer. The Emulator may act as a Forward Station or an AFT Station.

1.3 Ethernet Protocol.

The Ethernet original baseband version was designed, developed, and patented by Xerox and was publicly announced in 1979. Since then a cooperative effort by Digital Equipment Corporation, Intel, and Xerox has produced an updated Ethernet which is considered the standard for cable-based Local Area Networks because it is very close to the IEEE 802 CSMA/CD standard. The Carrier Sense Multiple Access with Collision Detection (CSMA/CD) control technique is the more publicized method for bus/tree topologies. The CSMA/CD broadband version was developed and patented by MITRE as part of the MITREnet Local Area Network.

1.3.1 Function and Operation.

The Ethernet is basically a multi-access, packet-switched communications channel which is managed by the control technique CSMA/CD for carrying digital data among locally distributed computing systems. A primary goal of the Ethernet specification is compatibility. In fact, Ethernet was the first to accomplish this capability.

Using the CSMA/CD control technique, each station attached to the bus must contend with the other stations to access the bus. There is no central controller which allocates access to the channel. Each station must 'listen' (i.e. use carrier sense) to detect whether the bus is free. If another station is transmitting, a station must wait or defer its transmission until the bus is quiet. After gaining access to the bus, the transmitting station continues to monitor the medium to detect colliding transmissions on the bus. This is called 'listen while talk' and refers to carrier detection.

1.3.2 Data Format and Structure.

Each station on the common coaxial cable must be able to transmit and receive packets with the packet format and spacing as shown in Figure 1.10 [KI86]. A packet is made up of bytes. (One byte equals 8 bits.) The last bit of each byte is transmitted first, and the preamble begins a transmission. A packet may not exceed 1526 bytes or fall below 72 bytes. Included in each of these numbers is 8 bytes for the preamble, 14 bytes for the header, the data bytes, and 4 bytes for the CRC. Each field of the frame is defined as follows:

1) Preamble: 64 bits alternating 1's and 0's, and ending with two consecutive 1's. Used by the receiver to establish bit synchronization and then to locate the first bit of the frame.

2) Destination Address: 48 bits specifying the station or stations which are to receive the packet. The packet may go to one station, to a group of stations, or to all stations. This is

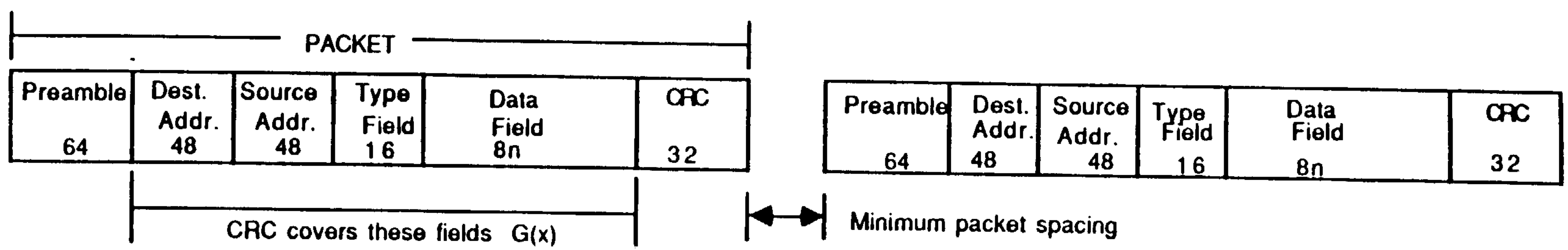


FIGURE 1.10 ETHERNET PACKET STRUCTURE

determined by the first bit: 0 - one destination, and 1 - multiple stations. If all 8 bits are set to 1, then the packet is broadcast to all.

3) Source Address: 48 bits specifying the station which is transmitting the packet.

4) Type Field: 16 bits identifying the type of higher level (ISO levels) protocol associated with the packet. Used to interpret the following data field.

5) Data Field: 46 to 1500 bytes of data or pad characters. A minimum combination of 46 bytes is required to ensure that the frame will be distinguishable from a collision fragment.

6) CRC - Packet Check Sequence: 32 bits containing a redundancy check. The check is defined by the generating polynomial:

$$G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1.$$

The address (destination/source), the type, and the data fields are covered by the CRC. The high-order term of the message polynomial which is divided by G(x) and produces the remainder R(x) is the first transmitted bit of the destination field. The first transmitted bit of the Packet Check Sequence field is the high-order term of R(x). A linear feedback register which is initially preset to all 1's is used in this algorithm. After the last data bit is transmitted, the contents of this register (the remainder) are inverted and transmitted as the CRC field. After receiving a good packet, the receiver's shift register contains 11000111000001001101110101111011($x^{31},\ldots,x^{0}$).

The Ethernet has an enforced waiting time on the bus of 9.6 micro-seconds. This is the minimum amount of time which must elapse after one transmission before another may begin. It takes 51.2 micro-seconds for one bit to travel from one end of the bus to the other (the round-trip propagation delay time). If any station receives a packet or bit sequence shorter than 72 bytes, the information is discarded and considered a collision fragment.

1.3.3 Hardware Characteristics.

The following three sections contain a brief overview of the hardware aspects of the Ethernet network system: channel encoding, carrier detection, and the transceivers. Additional information including detailed hardware specifications may be found in <u>Telecommunications and Data Communication System Design with Troubleshooting</u> by Harold Killen [KI86].

1.3.3.1 Channel Encoding.

The coaxial cable uses Manchester encoding which has a 50% duty cycle and insures a transition in the middle of every bit cell ('data transition'). The complement of the bit value is contained in the first half of the bit, and the second half contains the true value of the bit. (See Figure 1.11 [KI86].)

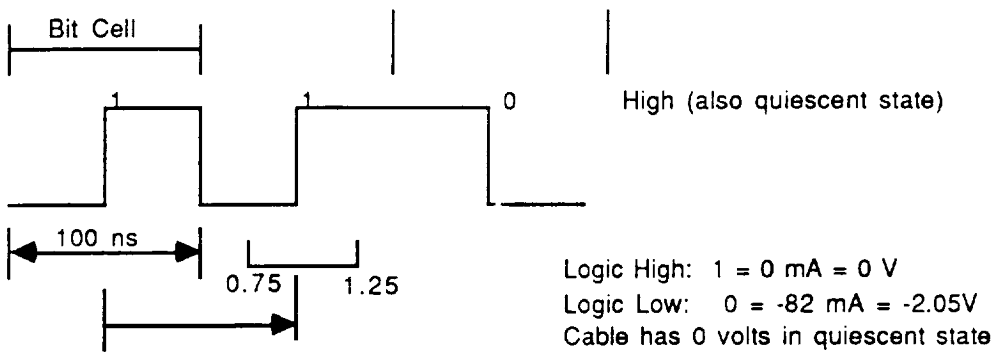


FIGURE 1.11 DETERMINATION OF CARRIER AT RECEIVER

1.3.3.2 Carrier.

When data transitions are present, a carrier is present. The carrier has been lost (indicating the end of a packet) if a transition is not seen between 0.75 and 1.25 bit times since the center of the last bit cell. For purposes of deferring, the term carrier means any activity on the cable, whether properly formed or not. Any activity on either receive or collision detect signals in the last 160 nano-seconds indicates carrier. (See Figure 1.11 [KI86].)

1.3.3.3 Transceiver.

At each station using the network, there are cables with taps which connect to a transceiver. The transceiver receives all signals on the cable, but only those addressed to it are received for action. The transceiver is also the device which transmits signals that are strong enough to propagate the information from one end of the cable to the other. (That is, every transmission on the cable will reach each transceiver.)

The transceiver was designed so that if it fails, the faulty device will not jam or pollute the Ethernet cable. In addition, the devices are simply built and inexpensive so that replacement of failed parts may be accomplished quickly. If a transceiver is unpowered, it disconnects itself from the cable. The transceiver also contains a watchdog timer circuit which detects incorrect behavior and shuts down the transmitter in this event. The maximum number of stations which may be attached to the cable is 1000, with the stations spaced at least 2.5 meters apart to reduce the chance that objectionable standing waves will result.

1.4 Research Objective.

Within a Local Area Network environment, any of the network resources may be changed. However, the performance of the system may also change without being readily apparent. The simulation model of the Ethernet network was designed and developed in this research project to allow the user to analyze, characterize, and predict the behavior of the ABACS System in a variety of scenarios. The model parameters may be set to reflect the ABACS system activity or used to predict the performance of future configurations. The ABACS configuration is displayed in Figure 1.1.

The first section of this report has provided an introduction to ABACS including system configuration with message traffic defined and a description of the Ethernet protocol. Chapter 2 includes the definition and development of the software model and describes the user interface. Chapter 3 discusses the outcome of several simulated scenarios. A worst case analytical analysis is given in Chapter 4. Suggestions for improving system responsiveness are included in Chapter 5. The conclusion assembles the results of this research in order to provide insight into the operations of the ABACS System using an Ethernet Local Area Network.

2.0 SIMULATION MODELING

The ABACS Ethernet simulation program provides the user a means of analyzing a proposed system configuration prior to hardware installation. This program provides a simulation configuration as described in Sections 1.1 and 1.2. The ABACS hardware includes two control computers and ten station controllers connected by an Ethernet LAN. The simulation program models the ABACS message traffic which is primarily VAX to station commands, station responses to VAX, and station archive data to VAX. The specific operations modeled are presented in Section 2.1.

Within the simulation, specific characteristics of the Ethernet protocol are modeled to accurately analyze the system performance. For instance, the simulation includes provisions for multiple stations trying to simultaneously access the bus by comparing the transmit times to see if they occur within a collision window. This window is actually a time period in which all stations trying to transmit within it collide. This collision happens because the signal has not had time to propagate to all parts of the network. Other stations also detect that the bus is free and begin transmission. The packets then collide. This collision window interval is calculated by using the propagation delay value between the two devices. If the transmit times occur within this window then a collision will occur. In the same fashion, stations may also have to defer a transmission if another station has gained access to the bus. A packet is deferred if the station wishes to transmit before the end of the slot time plus the propagation delay between the two devices plus the minimum delay time on the bus (9.6 micro-seconds). When a collision occurs, the stations involved must wait a random period of time before trying to transmit again.

The Ethernet protocol specifies an exponential backoff algorithm, which is required to help minimize repeated collisions, to generate the next transmit time. The backoff number is a random number between 0 and 2^n times 51.2 micro-seconds,where n is the number of the current retransmit attempt and n is less than 10. (The maximum end-to-end, round-trip propagation delay for a bit is 51.2 micro-seconds.) In addition, a packet transmission is aborted and a jam pattern of four bytes is transmitted on the bus when a collision is detected. This jamming sequence lasts long enough so that other stations involved in the collision notice the jamming pattern. The Ethernet protocol also specifies that there must be a minimum wait time of 9.6 micro-seconds between any two transmissions on the Ethernet cable.

The program presented here incorporates the Ethernet protocol characteristics as stated above. The following sections describe the simulation model activity, the simulation performance parameters, how to use the program, and the software design and construct.

2.1 Simulation Traffic Flow and User Interface.

The simulation program reflects the actual ABACS system operations as closely as possible. These sections describe the activity between the devices attached to the Ethernet bus within the simulation model. Any deviation from the way the system actually operates is discussed here. One example is the assumption within the simulation that there is only one Ethernet bus rather than two. Since the ABACS design specifies that the system must be able to function using only one Ethernet cable, this simulation allows the user to analyze the system in the worst case configuration. It also encourages the user to analyze the system performance in other possible configurations.

A user work chart is provided in Appendix I which shows the user input parameters for each device. The chart is formed by taping together pages 1 through 8. Along the top of the chart are the block devices attached to the bus. Underneath each device are blanks to fill in the input parameters for the simulation.

2.1.1 Allowed Simulation Configurations.

The simulation model assumes that there is one Ethernet device with at least one VAX and at least one station controller attached. Within ABACS, there are a maximum of two VAX computers and a maximum of ten station controllers. The simulation automatically allows five VAX computers and fifteen station controllers. These numbers may be increased by performing a minor software change to the simulation program. Within the simulation program, any VAX may be set to communicate with any station controller although VAX II is primarily a backup device for VAX I in ABACS.

The station controllers may be classified in two categories: those which perform archiving and those which do not. In ABACS, the IEA, the ASA, and the VSWR/LA do not currently perform archiving. Within this simulation model, all station controllers may be set to perform archiving. The number of iSBC 86/14 boards must also be specified by the user. The maximum number allowed in the simulation is seven. However, the realistic number of 'archiving' boards for each device varies. For example, the forward station has as many as seven, but the DFI only has four.

As in ABACS, the only stations which may be set to communicate with each other are the TVC and an AFT station. The simulation program will not allow the user to set other stations to communicate with each other.

Transparent to the user is the acknowledge message. The acknowledge is a 72 byte Ethernet message which is automatically sent from the receiving station to the station which transmitted.

In a 'realistic' worst case simulation of ABACS, the scenario should include only one VAX computer and all ten of the station controllers. The scenario should include seven devices performing archiving at the maximum rate possible and three stations which are not performing archiving. The TVC and an AFT should also be set to communicate. The VAX should be set to send station commands as fast as possible. Then this scenario would represent both a 'realistic' worst case scenario within ABACS and the maximum amount of information that would be transferred over Ethernet for ABACS.

2.1.2 VAX Commands to Station and Station Response.

The activity here is simply that the VAX sends a command or a group of commands (blocked commands) to a station and the station may respond to some or none of the commands. When a block command is received by a station controller, the station executes the commands individually and responds to the VAX appropriately. In the simulation model, the user is required to enter several parameters to allow the simulation program to produce much the same activity of ABACS.

The user must enter a range in which a new transmit time for the VAX may be generated. Since the VAX sends commands on a random time basis, some random value between the specified range will be added to the previous transmit time to generate the time at which the VAX will send its next command. An example might be 0.1 to 0.3. In this case, the VAX next transmit time will be generated by adding some random number between 0.1 and 0.3 to the last transmit time of the VAX. In this manner, ATLAS commands may be generated within the simulation program at a rate similar to an ABACS operator generating ATLAS commands or a test sequence execution.

The user must enter the ratio of single commands to be sent versus the number of blocked commands which are to be sent (groups of commands sent to a station in a single packet). For a blocked command the user also specifies the number of commands which constitute a block. Finally, the size of a command in bytes must be selected. An example could be the ratio of 5 to 1 where there will be 5 single commands sent for every 1 blocked command which may consist of 10 commands.

The ratio of commands requiring a station response versus those which do not must be specified by the user. An example could be the ratio of 10 commands which do not require a response to 1 which is issued a response by the station. For the station to respond, the user must specify the packet size of the response and the delay time before the response is transmitted. The delay time includes the average time the station takes to receive and interpret the command, perform the command, and package the command for transmission.

2.1.3 Station Controller Startup.

The simulation program user must specify the time in seconds in which the station comes on-line. The initialization sequence is similar to ABACS, but is generalized for simulation purposes. At the user specified time, the station will be sent packets from the VAX. The station will respond to each packet with a special acknowledge. After this exchange, the station will send the VAX information from each iSBC 86/14. The start-up sequence is complete when the VAX sends 100 initialization packets to the station (twenty 1500 byte buffers and eighty 20 byte buffers).

The user inputs are station start-up times, the number and size of packets the VAX sends to stations, and the number and size of 86/14 buffers to send to the VAX from each station.

2.1.4 Station Controller Archiving.

The iSBC 86/14 boards accumulate archive data which must be sent to the VAX. The station will transmit the archive buffer as frequently as the user specifies to model the actual ABACS buffer data accumulation. The simulation program allows the user to specify the number of iSBC 86/14 boards which are designated for archiving. For each of the buffers, the user specifies the size of the buffer in bytes and the number of seconds (or fraction of a second) it takes for the buffer to fill.

2.1.5 TVC and AFT Communication.

The user enters the number and size of the packets which are to be transmitted to the opposite device in one second. For the TVC/AFT communication the TVC is the initial device to transmit. When the AFT receives the packet, the device is delayed for a period of time before responding to the TVC. The TVC then delays for some period of time before sending another packet to the AFT. The user enters the amount of time that each device delays. If all the messages are unable to be transmitted in a one second time frame, then an error message will be displayed on the simulation printout.

2.1.6 Watchdog Timer Operations.

Although the user is not required to enter any information, a watchdog timer is implemented. Every one second the VAX control computer automatically sends an eighty byte packet to each station which communicates with that VAX. The station immediately responds to the VAX with an eighty byte packet.

2.1.7 VAX Simulation.

A special simulation feature is included to determine when a VAX computer is overloaded with data. The user enters the amount of data that a particular VAX may transmit in one second and the amount of data that it may receive in one second. If the VAX transmits or receives more than these user specified values, then an error message will be displayed on the simulation printout. For each VAX the amount of data transmitted and received in any second of the simulation will also be displayed on the simulation printout.

2.2 Performance Parameters.

There are three primary performance parameters which are of interest when analyzing a Local Area Network.

1) Throughput - The total amount of data which was actually transmitted successfully on the cable. Also defined by William Stallings in Local Networks: An Introduction [ST84] as the total rate of data being transmitted between nodes (carried load).

2) Delay - The amount of time that a packet must wait between the time when the packet is ready to be transmitted at a node and the time when transmission has been completed successfully.

3) Utilization - The total amount of data (or offered load) offered to the bus presented as a percentage of bus capacity. Also defined by [ST84] as the fraction of total capacity being used.

The throughput simulation results are given as simulated throughput and theoretical throughput. Both values are presented for comparison purposes. The simulated throughput is calculated as follows:

$$S = \frac{U}{B + I}$$

where B = average duration of the channel busy period
I = average duration of the channel idle time
U = average time during a cycle time that the channel functions without collisions
B + I = average cycle time.

Thus, the simulated throughput is a measurement of the channel activity. The above values are tallied as the program runs and keeps accurate records of exactly what is occurring on the bus. The theoretical throughput, however, is a calculated measurement. The following formula was derived by Killen [KI86] for the CSMA protocol:

$$T = \frac{Ge^{-AG}}{G(1 + 2A) + e^{-AG}}$$

where G = number of new packets per unit of time + number of retransmitted packets per unit of time, and
A = unit of propagation time.

Thus, G is the offered load to the system - the total amount of information new and repeat which was transmitted on the channel. This value is recorded as the program runs and is used in the above equation to produce the theoretical throughput. Notice, however, that the theoretical results will be much lower than the simulated results because the CSMA protocol does not have the collision detect capability or the exponential backoff calculations (to minimize repeated collisions) as does the CSMA/CD protocol which is modeled here. The theoretical value is included as a fundamental parameter used for comparison and verification of the simulation results.

To produce the efficiency measurement, $E = \frac{S}{G}$, where S is the simulated throughput and G is the offered load as defined above. The efficiency performance figure describes the percentage of time a transmission will occur with no collision.

In addition to the performance figures discussed, each device attached to the bus has statistics which are of interest when examining the overall performance. These performance figures include:

1) the total waiting time of a device due to packets being deferred from transmission,

2) the total waiting time of a device due to packet collisions,

3) the minimum amount of time a device has ever had to wait to access the bus due to a packet being deferred or being caught in a collision,

4) the maximum amount of time the device has had to wait for an individual packet to be transmitted (including a collision, random backoff, and retry),

5) the maximum total wait time of an individual packet due to repeated collisions.

A breakdown of the number of defers and number of collisions that a device has experienced is also included as well as a count of the number of packets it has received and transmitted. The maximum number of times that a packet collided for any device is recorded. Table 2.0 (page 46) depicts a typical printout of a simulation run.

The simulation program also records transmit and receive information for the VAX computers. If a VAX transmits more or receives more than the user specified amount, then a message indicating that the VAX is overloaded appears on the printout. In addition, a second by second summary of each VAX activity is recorded to show the transmitted bits and received bits in any second of the simulation. Another feature incorporated into the simulation is a message which is included on the printout if the packets between the TVC and AFT are not transmitted within a one second interval. This message informs the user that the message traffic was so heavy that the packets between the TVC and AFT could not be transmitted in the time allocated.

2.3 Simulation Software Design and Construct.

The Ethernet simulation source code is divided so that there is a short main routine which calls many subroutines to do the actual work. The primary logic is displayed in the flow chart of Figure 2.0. The program variables are set to their appropriate initial states in the subroutine Initialize. Configure is the subroutine which allows the operator to change the system configuration in order to fill the specific requirements. (See Section 2.1 for more details on the

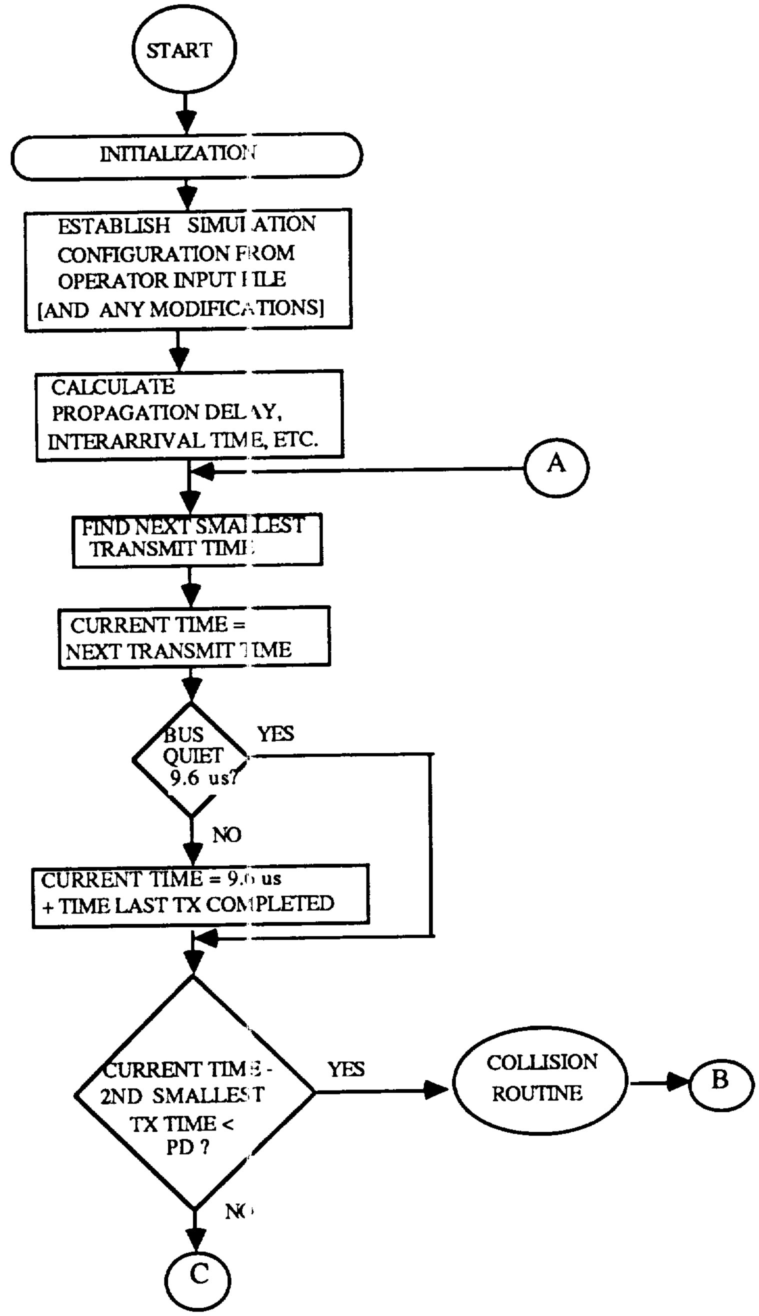


FIGURE 2.0 FLOW CHART OF ABACS SIMULATION

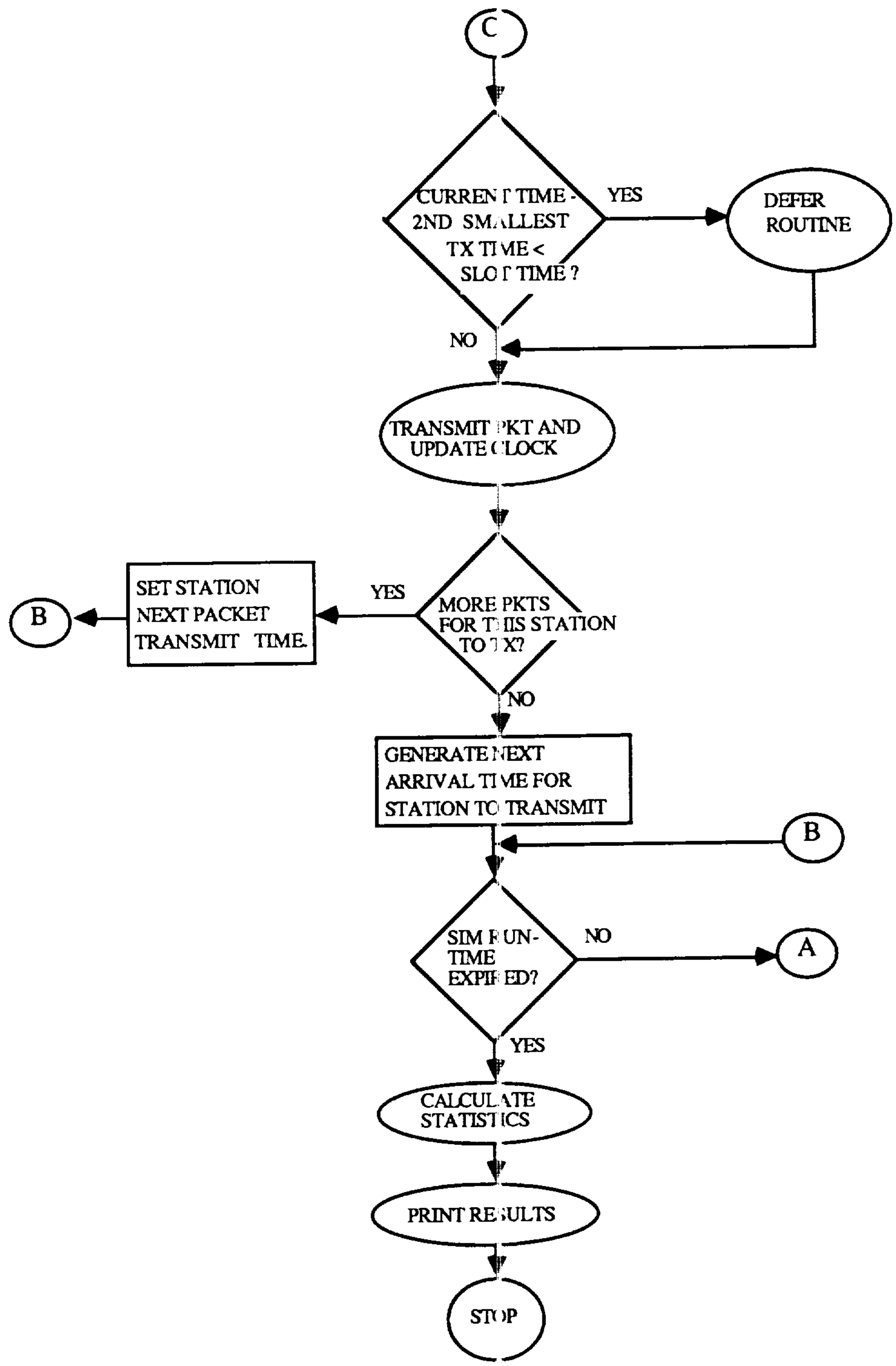


FIGURE 2.0 FLOW CHART OF ABACS SIMULATION (CONTINUED)

operator input parameters.) Configure calls Display, Modify, and Store to allow the operator to review and change any of the configuration parameters before beginning the simulation. Prtdata writes the configuration set up by the operator to a file called About so that this information may be printed when the simulation has finished.

The program activity begins by determining the next station which is to connect. The simulation begins when the routine FindNext is invoked to determine the next station to transmit a packet depending on the next smallest transmit time. A call to Acollision determines if there is a collision on the bus, performs the exponential backoff algorithm, insures a jam pattern is sent, and updates the affected system parameters (wait time, system clock, etc). If there is no collision, then the Checkdefer routine determines if there are any stations which must defer its transmission until the current transmission is complete. The most complicated subroutine may be the Update routine which handles updating the station which just transmitted. This includes setting the receiving station to respond with an acknowledge and initializing the receiving device to provide a response as required. These routines are repeated over and over until the program has performed the bus simulation as long as specified by the operator (sim time). The results of the simulation are added to the file About by the subroutine Prtres.

In addition, there are many support subroutines which are called by the controlling routines. The random number generator Ran returns some random number between 0 and 1. The routines Getvaxnxt and Getcontnxt determine the next operation to be transmitted for a specified device. This includes examining the acknowledgements, watchdog timer messages, commands, reponses, and archive messages to determine the next operation to be performed by this device. Similarly, the routines Setvaxtime and Setcontime handle generating the time for the next transmission and the message size which is to be allowed by the device and determine which commands sent by the VAX will receive a response. Upvax and Upcont update a particular device so that transmit times for other operations on the device do not fall below the current clock time within the simulation. The Setstartup routine handles a new device coming on-line. And the Setxtimes initializes a VAX or controller which has come on-line.

2.4 Verification of Model.

The information shown in Table 2.0 contains a summary of fifteen simulations with the system configuration modified each time. The graph of Figure 2.1 plots the offered load (G) to the system versus the simulated (S) and theoretical (T) throughputs for the fifteen examples. The theoretical curve is identical to the plot produced in [KI86]. The equation for the theoretical throughput is given in Section 2.1. However, the theoretical information only represents the CSMA protocol and not CSMA/CD. Comparing the two curves shows that the additional features - carrier detect and the exponential backoff algorithm - greatly increases the performance of the CSMA/CD protocol.

Two references [SH80] and [ABA77] also produce similar plots to show

TABLE 2.0 SIMULATION RESULTS, RUNS 1 - 15

SIMULATION RUN RESULTS

RUN NUM	OFF LOAD	SIM THRPT	THEO THRPT	EFF	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MIN PKT WAIT TIME	MAX PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
1	1.06E+00	1.06E+00	1.05E+00	9.99E+01	1.23E-03	1.76E-04	17	2	783	360	783	1.55E-11	1.20E-03	2	1.31E-03
2	9.51E+00	9.28E+00	8.69E+00	9.75E+01	3.20E-01	4.19E-01	755	392	5899	2835	5899	4.46E-11	2.79E-02	9	3.69E-02
3	1.60E+01	1.56E+01	1.38E+01	9.75E+01	6.45E-01	3.99E-01	1343	696	8230	4000	8230	1.55E-11	2.54E-02	8	2.80E-02
4	3.40E+01	3.26E+01	2.54E+01	9.58E+01	2.67E+00	8.88E-01	4674	2451	13699	6735	13699	1.55E-11	5.05E-02	12	1.03E-01
5	4.94E+01	4.63E+01	3.30E+01	9.36E+01	6.46E+00	2.59E+00	9951	4692	17633	8802	17633	4.46E-11	5.22E-02	13	1.30E-01
6	5.14E+01	4.74E+01	3.40E+01	9.22E+01	7.55E+00	3.98E+00	12695	6334	20851	10311	20851	1.55E-11	4.68E-02	14	1.24E-01
7	5.37E+01	4.94E+01	3.49E+01	9.20E+01	8.28E+00	4.22E+00	13637	6713	21483	10627	21483	1.55E-11	4.50E-02	15	1.67E-01
8	5.83E+01	5.32E+01	3.68E+01	9.13E+01	1.02E+01	6.58E+00	16128	7653	22659	11214	22659	4.46E-11	5.23E-02	15	2.16E-01
9	6.63E+01	6.00E+01	3.99E+01	9.05E+01	1.31E+01	1.45E+01	20131	9236	24691	12231	24691	1.55E-11	5.23E-02	16	2.11E-01
10	7.42E+01	6.65E+01	4.26E+01	8.96E+01	1.62E+01	2.74E+01	24268	11034	27282	13525	27282	1.17E-08	5.23E-02	16	2.39E-01
11	8.29E+01	7.31E+01	4.53E+01	8.82E+01	2.05E+01	5.95E+01	30544	13135	29088	14381	29088	1.55E-11	6.26E-02	16	2.57E-01
12	8.40E+01	7.40E+01	4.56E+01	8.82E+01	2.13E+01	6.48E+01	31230	13075	29239	14431	29239	1.55E-11	6.01E-02	16	2.56E-01
13	8.99E+01	7.75E+01	4.73E+01	8.63E+01	2.60E+01	1.06E+02	38542	15202	30564	15054	30564	1.55E-11	5.93E-02	16	2.98E-01
14	9.38E+01	8.07E+01	4.84E+01	8.60E+01	2.81E+01	1.48E+02	40736	15902	30616	15049	30616	1.55E-11	8.49E-02	16	4.53E-01
15	1.01E+02	8.59E+01	5.02E+01	8.52E+01	3.07E+01	1.97E+02	44623	17785	32005	15797	32005	1.72E-08	5.69E-02	16	4.17E-01

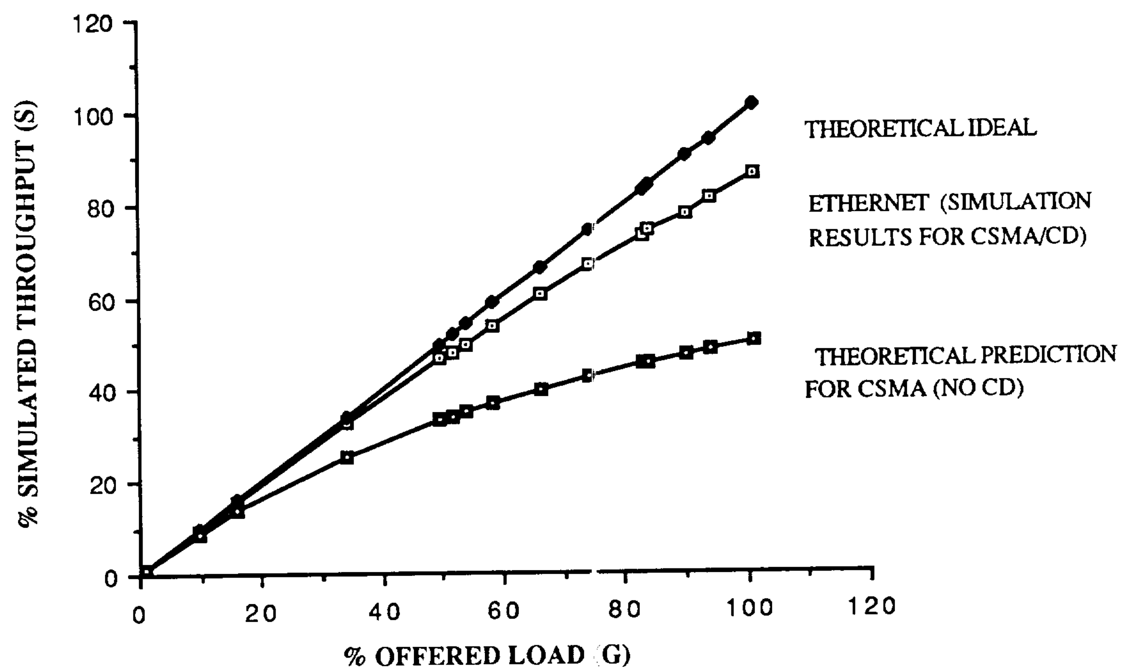


FIGURE 2.1 OFFERED LOAD VERSUS THROUGHPUT FOR FIFTEEN SIMULATION RUNS

the Ethernet bus utilization. According to Agrawala, Bryant, and Agre in 'An Analysis of an Ethernet-Like Protocol' [ABA77], the maximum throughput rate is 80% when the bus is fully loaded. And, Shoch and Hupp in 'Measured Performance of an Ethernet Local Area Network' [SH80] specify that the throughput rate when the offered load is 100% increases as packet sizes become larger - 512 bytes/packet = 96% throughput, 128 bytes/packet = 88% throughput, and 64 bytes/packet = 83% throughput due to the increased possibility of packet collisions. The results produced by this Ethernet simulation agree with these sources since the throughput rate at 100% offered load is about 85%. This figure is higher than the results of the simulation model presented by [ABA77] and is lower than the estimates of [SH80]. These sources provide a reference of comparison to verify proper operation of the Ethernet simulation.

2.5 Using the Program.

When using the simulation program, the user should have an idea of the expected results before they appear. For instance, with a lightly loaded bus the user should expect very few collisions and a very high efficiency rate. In this section, several simulation parameters will be discussed to help the user understand more fully the outcome of the simulation.

The user should review the work chart in Appendix I. By using this chart, the simulation input parameters may be determined before actual entry into the simulation program. This model was designed to simulate as closely as possible the activity of the ABACS Ethernet network. However, the some input parameters are simplified. An example is the archive buffer fill times. The user must estimate the number of inputs which will change for an 86/14 board and specify to the simulation program how often the buffer is to be sent to the VAX. In other words, every aspect of ABACS is not specifically modeled, but the basic operations are generalized and model as closely as possible the activity of ABACS.

The simulation run-time is a parameter entered by the user to allow the bus activity to be observed during a certain time frame. In different situations this parameter will need to be increased or decreased depending on the activity on the bus. If, for example, a station controller is brought on-line at the end of the simulation run time, then the device will not have a chance to contribute to the loading of the bus. This is a case where the run-time must be high enough to allow the bus activity to stabilize so that an accurate report is given. If the operator is unsure of this run-time figure, several runs may need to be performed and compared against each other. Notice in Table 2.1, the same configuration (Run 5) was run ten times with the simulation run-time increasing each time (varied from 10 seconds to 100 seconds). The performance parameters: the offered load, the simulated throughput, and efficiency, remain close for all runs implying that the activity on the bus has stabilized and longer simulation runs would produce about the same results. From many test simulation runs it is apparent that the Ethernet simulation requires a relatively low run-time for most configurations.

TABLE 2.1 SIMULATION RESULTS, RUN 5, TIME 10 - 100 SECONDS

SIMULATION RUN RESULTS

RUN NUM	OFF LOAD	SIM THRPT	THEO THRPT	EFF	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MIN PKT WAIT TIME	MAX PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
1	3.71E+01	3.55E+01	2.71E+01	9.55E+01	1.64E+00	3.53E-01	2735	1429	7507	3723	7507	2.36E-07	3.67E-02	11	7.51E-02
2	4.94E+01	4.63E+01	3.30E+01	9.38E+01	6.46E+00	2.59E+00	9951	4892	17833	8802	17833	4.46E-11	5.22E-02	15	1.38E-01
3	5.24E+01	4.90E+01	3.44E+01	9.35E+01	9.20E+00	4.04E+00	13970	6774	23105	11388	23105	4.46E-11	5.23E-02	15	1.38E-01
4	5.45E+01	5.08E+01	3.53E+01	9.33E+01	1.19E+01	5.49E+00	17982	8655	28399	13985	28399	4.46E-11	5.23E-02	16	1.38E-01
5	5.59E+01	5.20E+01	3.59E+01	9.31E+01	1.47E+01	7.08E+00	22025	10549	33595	16531	33595	4.46E-11	5.23E-02	16	1.78E-01
6	5.70E+01	5.30E+01	3.63E+01	9.30E+01	1.74E+01	8.24E+00	26061	12495	38870	19120	38870	4.46E-11	5.23E-02	16	1.78E-01
7	5.85E+01	5.44E+01	3.69E+01	9.29E+01	2.26E+01	1.08E+01	33721	16135	49379	24275	49379	4.46E-11	5.23E-02	16	1.78E-01
8	5.95E+01	5.52E+01	3.73E+01	9.23E+01	2.81E+01	1.39E+01	41731	19975	59825	29398	59825	4.46E-11	5.23E-02	16	2.03E-01
9	6.03E+01	5.59E+01	3.76E+01	9.27E+01	3.35E+01	1.62E+01	49841	23868	70405	34588	70405	4.46E-11	5.23E-02	16	2.03E-01
10	6.12E+01	5.67E+01	3.80E+01	9.27E+01	4.43E+01	2.18E+01	65518	31319	91385	44878	91385	4.46E-11	5.23E-02	16	2.03E-01

3.0 SIMULATION RUN RESULTS

In this section, the focus will be the on interpretation of the results produced by the Ethernet simulation model. This discussion will include an analysis of a specific run to explain performance parameters and the effect of varying user inputs, a configuration summary of fifteen runs with an intuitive preliminary analysis to predict results, a comparison of fifteen simulation runs to demonstrate the system response in various configurations, and a summary of the configuration limitations encountered when using an Ethernet Local Area Network.

3.1 Analysis of A Simulation Run.

A typical ABACS system configuration is shown in Figure 1.0. The communication scenario as presented for this configuration is an average case within ABACS. It includes one VAX which communicates with six archiving stations and three non-archiving stations and a second VAX which communicates with the Emulator which also archives. This simulation will be discussed throughly. The input parameters with the simulation results are shown in Appendix III, Run 2. The output report of each run is divided into seven sections:

1) Overall System Parameters.
2) Station Controller Startup Parameters.
3) VAX Communication with Stations.
4) Station Responses and Archiving.
5) Summary Table of VAX and Station Activity on the Bus.
6) Overall System Performance Results.
7) VAX Transmit and Receive Summaries.

The first four sections display the operands entered by the user. The overall system parameters include a description of the simulation run, the Ethernet bus I/O rate, the simulation run time, and the random number seed. The information as entered by the user is output for each station attached to the bus. The parameters include the station start time, the VAX which communicates with the station, the number and size of the VAX initialization packets, and the number and size of the initialization archive packets sent to the VAX. This section also includes the TVC and AFT communication parameters: the number of messages per second, the delay time between transmissions, and the packet sizes. For each VAX attached to the bus, the distance from a reference point and number of bytes the VAX can transmit and receive in a second is displayed. For each station which communicates with the VAX, the range is entered by the user which allows random transmissions by the VAX to a station. Then the number of single commands versus the number of blocked commands are displayed with the number of commands per block and the packet size of a command also specified for each station. The station parameters include the distance from some reference point, the number of commands which are issued a response versus those which are not, the average delay time before a response is generated, and the packet size of a response. A user work chart is provided in Appendix I which allows the user to easily define the simulation configuration prior to actually entering parameters into the program.

For the simulation configuration of Appendix III, Run 2, the input parameters allow communication between VAX I and nine stations and between VAX II and the emulator. Each station will be initialized with 10 packets of 1500 bytes from the VAX and with 10 packets of 1500 bytes returned from the station. The TVC and AFT II communicate by sending fifteen 80 byte packets to the other station in a second. The VAX sends commands to each station randomly within an average range of 0.18 to 0.385. The AFT, for example, will receive a maximum of 10 packets per second from the VAX and at least 3.3. The actual number of packets sent to the AFT in a second is generated randomly within the range for the AFT. For this simulation, most of the commands sent from the VAX are single commands and do not require frequent responses from the station. The stations are configured so that all 86/14 boards which perform archiving within ABACS perform archiving in this simulation. The rates at which a board fills is generated from a 40 milli-second scan cycle with 10 percent of the board inputs changing on each cycle.

The report generated by the simulation provides a table which displays the activity of each device attached to the Ethernet bus. The far left column labeled SOURCE defines the device under observation. For each device attached to the bus, several important values are tallied in order to observe the Ethernet activity. The WAIT TIME DEFER column specifies the total amount of time the device had a packet to transmit but had to wait until the bus was free. Similarly, the WAIT TIME COLLISION column records the total amount of time the device was involved in a collision. This total includes the time during packet transmission as the collision occurred, the time spent sending the jam sequence upon detecting a collision and the time the device was required to wait until it could retry the packet. The devices which transmit most frequently will also be the devices most likely to have a high wait time due to collisions. In this case, VAX I and the TVC transmitted the most packets and also had the highest wait times due to packet collisions.

The DEFER COUNT and COLL COUNT columns total, respectively, the number of times that a device had to wait to transmit and the number of times it was involved in a collision. The COLL COUNT column indicates the number of times the device was involved in a collision even if the data packet was a re-try. For each collision, the packets of information are transmitted at a later point in time. The packet transmissions are recorded in the columns PKTS TX or ACKS TX which indicate the total number of packets transmitted successfully and the total number of acknowledgements transmitted successfully. The PKTS RX totals the number of packets received by the device.

The MINIMUM PKT WAIT TIME column indicates the shortest period of time spent by a device when trying to access the ETHERNET cable. Similarly, the MAXIMUM PKT WAIT TIME indicates the longest period of time spent by a device when trying to access the cable. The MAX NUM COLLS indicates the maximum number of times that any one packet collided. The maximum amount of time that any packet waited to transmit is recorded in the MAX PKT COLL TIME column. In this case, the TVC had a packet which was involved in a collision 9 times and had to wait to access the bus 36.9 milli-seconds. Notice that the AFT and TVC were set to communicate 30 messages each second. The 36.9 milli-second wait time delayed a TVC message so that the 30 messages were not sent in one or more of the 20 simulation seconds. Under the columns displaying individual device statistics are column totals, maximum values, and minimum values. For example, the maximum number of collisions experienced by any packet during the simulation was nine and the total number of transmitted packets was 5899.

The overall results of the simulation are then given (bottom of page 166). The total busbusy figure is the total time the bus was active, including the time used transmitting successful packets, the time used on the bus during collisions, and the minimum required dead time of 9.6 micro seconds on the bus. The total usage time is the time spent transmitting packets of information on the bus without collisions. The total idle time is the amount of time the bus was unused or idle. Then the busbusy time plus the idle time equals the simulation run time. The average values are the totals divided by the total number of packets transmitted.

The simulated throughput is calculated using values totaled within the program. This value is calculated by diving the average usage time by the sum of the average busbusy and the average idle time to give the successful utilization of the bus. This value may also be calculated by taking the total number of bits successfully transmitted and dividing it by the Ethernet bus rate times the simulation run time. Thus, the usage time is simply the number of successfully transmitted bits divided by the Ethernet bus rate (or the sum of the slot times for all packets transmitted successfully). For this example, the total number of bits will be estimated. The number of acknowledge packets was 2835 at 640 bits each, the total VAX transmissions was 1643 with 100 packets of 1500 bytes and the majority of the remainder at 640 bits, and the total number of station controller messages was 1421 with a majority of the messages at 12000 bits or 1500 bytes and an estimated 200 messages at 640 bits. Using these estimates, the

$$\text{throughput} = \frac{4578*640 \ + 1321*12000}{10E6 * 20.} = .09391.$$

This value is higher than the throughput calculated from values tallied by the program which indicates that the number of estimated bits is higher than the actual number of transmitted bits. The number of bits which will be transmitted is difficult to estimate since there are many activities within ABACS which contribute to bus traffic. The actual number of successfully transmitted bits is calculated from the usage time of the bus: 1.85520 * 10E6 = 18,552,000 bits over a 20 second time period.

The aggregrate offered load is the total amount of data transmitted on the bus including packets successfully transmitted (18,552,000 bits) and the bits transmitted by each device involved in any collision. If three new or repeated packets collide, then the maximum or worst case offered data to the bus is 512 bits by each device or 1536 bits. The round trip propagation delay of a bit is 51.2 micro seconds and a slot time is defined as 512 bit times. Therefore, the maximum number of bits which may be transmitted on the cable by a device before a collision is detected is 512 bits. The total offered load will be calculated from the estimated traffic load of 18,781,920 bits. This value is added to the 392 collisions with at least 2 colliding devices of 512 bits each. The sum of these estimated values results in

19,183,328 offered bits. The offered load is 19,183,328 / (10E6 * 20) = 0.09592. The offered loading of the bus is actually about 9.59% of the total bus capacity or about one mega bit per second.

The theoretical throughput is calculated using the formula shown in Section 2.0. It displays the theoretical value for the CSMA protocol. The efficiency of the network in the configuration for this simulation example is calculated by dividing the simulated throughput by the offered load. This value specifies how efficient the network will be at transmitting the information across the bus, for example, 97.5% of the time the packet offered to the bus will be transmitted with no collision occurring. Another way to estimate efficiency is to calculate this percentage based on the actual number of packets transmitted and the number involved in collisions. In this case, there were 5899 packets transmitted successfully and 392 collisions. Therefore, the efficiency is about $\frac{5899}{5899 + 392} = 0.9377$, and about 6.3% of the time a packet will collide with another packet.

Two additional performance parameters have been included. The value in parenthesis is another suggested means of calculating offered load. This value will be referred to as alternate load. It is calculated by summing all packets transmitted successfully plus the full packet of information is included in the sum when a collision occurs. Offered load, however, is the sum of all the bits which were transmitted on the bus including successfully transmitted bits and bits transmitted in a collision. The parameter in parenthesis does not indicate the load of the bus since this value includes data which was actually never transmitted on the bus. The efficiency is then calculated using this parameter as if it were offered load. As the plot of simulated throughput versus offered load indicates in Figure 3.0 the curve using alternate load is very close to the CSMA protocol. In fact, the collision detection of the CSMA/CD protocol causes colliding packets to terminate transmission upon detecting a collision (a maximum transmission of 512 bits per colliding packet). If alternate load were actually the same as offered load then the entire packet would have been transmitted on the bus before a collision was discovered to indicate the loading of the cable. For this reason the CSMA curve and the alternate load curve are very similar and the CSMA/CD curve indicates a more efficient use of the cable since the entire packet is not transmitted before a collision is detected.

The total offered data indicates the sum of all the packets which could have been transmitted on the bus. The offered data is the sum of all the bits in packets which attempted transmission. When compared with the simulated throughput, the actual amount of information which has been offered but not transmitted due to collisions may be determined. The efficiency value then shows the amount of information transmitted versus the amount which was offered for transmission. The chart of Figure 3.0 shows that the offered data is close to the actual throughput value except at a very high loading. In the case of run 2, the throughput bits was 18,552,800 in 20 seconds, but the offered bits was 18,637,440 in 20 seconds. Therefore, only 84,640 bits remained to be transmitted when the simulation ended.

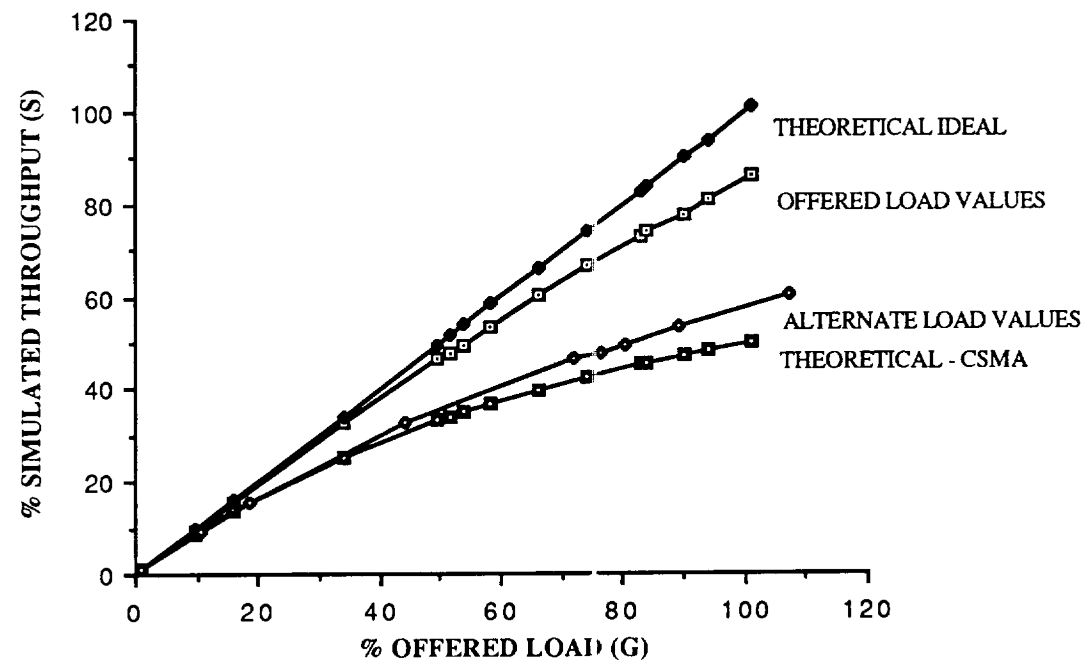


FIGURE 3.0 ALTERNATE LOAD VERSUS THROUGHPUT

A summary of the output information may be included in a special report if the user chooses this option. In this way the information from several runs may be examined from summary charts. The input information for a run is not included on this summary chart, but may be examined for variable data such as packet sizes and transmit times on the individual simulation report. However, the summary charts are very helpful when comparing the results of several simulation runs. (See Section 3.3.)

In order to simulate the operations of the VAX computers, the user enters an estimated number of bytes that the VAX may transmit and receive in any one second. The output report displays a message if these parameters are violated at any time in the simulation. This indicates to the program user that the configuration modeled inherently overloads the VAX device. Realistically, these values show that the VAX computer could not handle the amount of data being sent to it or the VAX computer could not transmit the information at that rate. If either of these parameters are violated, a VAX in an operational system would possibly loose information. In addition, the number of transmitted and received bits per second is displayed in a table at the end of simulation report for each VAX attached to the Ethernet cable. This table helps the user analyze the bus activity to determine any bottlenecks which may have occured in the simulation. Notice that the number of transmitted bits for a VAX is very high (about 400,000 bits) during the periods when a station controller is brought on-line. But, after the station is initialized the number of bits the VAX transmits in a second stabilized at about 10,000 bits.

The simulation under discussion (Run 2) is presented here as an average worst case condition of ABACS. There is only one Ethernet cable, all twelve devices are performing operations over the cable, and the station controllers are transmitting archive messages frequently. This scenario is possible but not likely within ABACS since all checkout tests are being performed simultaneously in this simulation. Actually, the scenario displays a very heavily loaded ABACS system. There were 80,000 bytes received by VAX 1 during the fifteenth second of the simulation run. This indicates that the VAX is very close to its maximum receive total of 90 Kbytes. However, only about 10% of total Ethernet cable capacity was utilized, indicating that the Ethernet cable for this ABACS scenario is lightly loaded. The offered load must be increased significantly before the efficiency of the Ethernet cable is reduced.

In summary, the dual Ethernet Local Area Networks used within ABACS provide the capability for transmission to occur on one network and message reception on the other network. If one Ethernet should fail during checkout operations, the remaining Ethernet is used both for transmission and reception. From the results of this simulation program, the one Ethernet cable can easily carry the load which normally would have been shared. The major operational problem of a failed Ethernet cable is the inability to bring any additional station controllers on-line. For a VAX to initiate communication with a station controller, both Ethernet cables must be functioning properly. Both Ethernet cables are not required for proper operations in ABACS since the simulation results indicate only a 10% bus utilization in an

ABACS worst case scenario (maximum traffic load using one cable). This simulation also indicates that the major bottleneck in the ABACS system will be the I/O limitations of the VAX.

3.2 Simulation Runs - Configuration and Expected Results.

Many simulation runs have been performed to date. The summary information of fifteen different scenarios/ configuration runs are included in Appendix III, page 166. The basic configuration consists of 2 VAX computers which periodically transmit information to station controllers and 10 station controllers which respond to VAX commands and send archive information. The packets sizes vary but primarily VAX commands are 80 bytes, archive messages are 1500 bytes, and acknowledge messages are 80 bytes. The parameters which varied over the fifteen runs are the number of commands issued from the VAX, the frequency archive messages were sent to the VAX, and finally, VAX computers and station controllers were added beyond the ABACS capability to increase network loading. Using this configuration as the basic scenario for observations, the bus performance can be analyzed using this simulation program.

Intuitively, as the amount of information pushed onto the bus increases, there will be a greater probability of collisions between packets since many devices will begin trying to access the bus at the same time and more frequently. Indeed, as the number of packets of information sent on the bus increased, the bus became much more heavily loaded and many more collisions did occur. As the number of collisions became larger and larger, the bus became heavily saturated. More packets were offered but fewer escaped a collision. The system then remained occupied in an attempt to clear the backlog of packets. The efficiency of the system will decrease as packets are offered and the heavily loaded system will not be able to transmit the information because of the load created by many collisions that cause repeated packet transmission.

Metcalfe and Boggs [MB76] of Xerox Palo Alto Research Center used an experimental Ethernet system to analyze performance. Their system was consistently 95% plus efficient when packet sizes were above 4000 bits. They conclude, 'For packets with a size approaching that of a slot, Ethernet efficiency approaches 1/e, the asymptotic efficiency of a slotted Aloha network'. The slot is defined here as 'the maximum time between starting a transmission and detecting a collision, one end-to-end round trip delay'. Since this delay time is 51.2 micro-seconds, then the smallest packet sizes (around 576 bits) will produce the least efficient network. <u>The 10 Mbps Ethernet bus would have an effective data rate of only 3.68 Mbps as packet sizes became small and the number of stations became large</u>.

The ABACS configuration has a maximum of 12 devices attached to the Ethernet. Compared to the maximum number of 1024, the number of stations is very small. In addition, there are both small and large packets within ABACS. For every data packet sent within ABACS there is an automatic acknowledge by the receiving device. This is the primary source of small packets (80 bytes). However, the efficiency of the

ABACS system remained high even under a very heavy load. The efficiency of this network configuration at 100% offered load was 85.2%. The efficiency of this network remained very high since the number of stations was low and the packet size was large on the average. Increasing the number of stations and lowering the packet size would reduce the efficiency of the network as concluded by Metcalfe and Boggs.

3.3 Comparison of Results when Parameters Vary.

In addition to obtaining detailed result from each run, the operator may choose to have the simulation information included in summary tables containing information from previous runs. This provides a concise summary of each simulation and allows easy comparison of the results obtained from each run. In this manner, the system may be readily analyzed to determine the most acceptable configuration for testing.

The results of the fifteen simulation runs are included in Appendix III. The offered load was increased from 8% to 100% by varying the number of packets transmitted on the bus from VAX commands with station responses and station archive data. The number of VAX computers was increased to five and the number of station controllers was increased to fifteen to produce the loading of the fifteen runs shown. In this case, the throughput increased from 8% to 85%, and the bus efficiency dropped from 98% to 85%. The throughput rates were achieved in the 20 second interval as packet transmissions rose from

5,514 to 32,005 with packet sizes of 80 and 1500 bytes. The results of these runs are displayed in Figure 2.1 which shows the throughput rate versus the offered load. Even a heavily loaded Ethernet performs well due to exponential backoff and collision detection features.

The simulation of Appendix III, Run 5 shows the performance of ABACS under the absolute worst case conditions. The only difference between Run 2 and Run 5 are the transmit frequencies of the archive buffers. The rates at which an 86/14 board accumulates archive data is generated from a 40 milli-second scan cycle with all board inputs changing on each cycle. Although this loading is not actually possible within ABACS, the Ethernet cable is busy 9.5 out of 20 seconds. The offered load is 49.4% with a throughput rate of 46.3% which produces a bus efficiency of 93.8%. Only about half the Ethernet capacity is used, but the VAX computer within ABACS is unable to receive the information from the station controllers at that rate.

The simulation runs 6 - 10 gradually increases the number of commands the VAX sends to a station controller and the number of those commands which require a response is increased. The stations which currently do not perform archiving within ABACS were set to send archive data to the VAX. In this way, the bus performance is predicted for possible future configurations. In Run 10, all ten station controllers were archiving data with estimated buffer fill times based on a 40 milli-second scan cycle. The collision rate was high since many packets were offered simultaneously. During the 20 second interval, over half the time (13.3 seconds) was spent transmitting transmitting packets with no collisions. Notice that some data packets were lost in runs nine and ten since they were involved in collisions sixteen times.

In order to bring the offered load to 100% of capacity, additional VAX computers and station controllers beyond the scope of ABACS were added to generate additional traffic on the bus. For the configuration of run fifteen the effective data rate of the Ethernet was about 8.5 Mbits per second. However, there were many collisions which occured so that the system then remained occupied transmitting re-try data. The offered data to the bus was 91.07 compared to a throughput rate of 85.88. The number of bits transmitted was 171,769,594 bits where as 182,135,200 were offered as data to the bus. Therefore, about 10,365,606 bits were offered but remained in a state of collision resolution. The maximum possible would have been about 200 Mbits in 20 seconds. About 5.7% of the offered data remained to be transmitted.

3.4 Limitations on Configuration.

Ethernet specifications limit the cable length to 2.5 kilometers. This allows a maximum of 1024 stations connected 2.5 meters apart. The simulation does not handle this number of stations. The limiting factors are the computer memory and the amount of time required to run a simulation of this magnitude. For the allowable number of stations, see Section 2.2. Another specification of Ethernet is packet size. The maximum allowed size is 1526 bytes, and the minimum packet size is 72 bytes.

From the simulation runs presented, the system configuration may be varied in many ways. However, the performance of the network will vary greatly depending on how it is structured. To ensure that the system performance is high (efficiency above 75%), keep the packet size large. One obvious way to improve ABACS efficiency is to discontinue sending the automatic acknowledge by the receiving station. However, the acknowledge information is important in this system to verify important data reception. If the small packets were reduced, then the number of collisions on the bus will be reduced and thus the amount of repeated data. With very small packet sizes and a high number of stations transmitting, the efficiency is reduced to 36.8% of capacity.

A high performance on the Ethernet is also achieved when the data is divided over time so that transmissions are structured. Currently within ABACS the test sequences are not performed in a structured manner. For example, the archive buffers are set to fill and transmit at a certain time, but no effort was made to insure that packets had transmit times which did not interfere with other attempts. The ABACS tests actually occur on a random basis and VAX commands sometimes have responses and sometimes do not. If the messages could be time structured so that messages do not attempt to transmit in the same time frame, then fewer collisions would occur. However, the ABACS devices must have the flexibility of random access to the Ethernet. The archive buffers are filled as test results are analyzed and VAX commands are issued as needed to perform a test. For these reasons, structuring packet transmission within ABACS is not feasible.

Another limiting factor is the data rate of the devices attached to the bus. The bandwidth of the Ethernet cable is 10 Mbps. However, there is no way to utilize the maximum bandwidth unless there are devices attached which can supply information to be transmitted at that rate. For example, the VAX computers are able to transmit an estimated 100 Kbytes per second and receive only 90 Kbytes per second. In the examples of Appendix III, the initial two simulation runs are configured so that these parameters are not violated. If, however, the number of archive messages are increased as in Run 3, then the VAX computer is realistically unable to handle the load. For the ABACS system, the Ethernet bus is lightly loaded and can accommodate a much heavier load. The remaining simulation runs demonstrate that although the VAX could not realistically transmit or receive at the higher rate the Ethernet can efficiently carry a much heavier load. This illustrates the point that if the VAX units are replaced with higher I/O rate devices, then the system will be able to handle a higher data loading.

4.0 WORST CASE ANALYSIS

A worst case analytical analysis has been developed for the ABACS Local Area Network System. This analysis calculates the amount of information generated for transmission on the Ethernet cable by the ABACS stations and demonstrates that the ABACS system can accommodate a worst case load even on a single Ethernet LAN.

4.1 Analysis

The worst case analysis of the ABACS data communication must involve the Ethernet, the iSBX 186/51 LAN interface boards, the SBC 86/14 boards, which are the interface between the analog data sensors and digital data sources and the iSBX 186/51 boards, and the input/output capabilities of the VAX units.

As stated previously in this report, the VAX units can transmit at approximately 100K bytes per second (100 KBps) rate and receive at approximately a 90 KBps rate to the iSBX 186/15's. The basic stations connected to the Ethernet cables are the FWD and AFT stations (2 of each), the EMU, the TVC, the IEA, the ASA, the VSWR remotes, and the DFI. Figure 1.1 illustrates these distances.

The normal operation of the system will <u>not</u> require the simultaneous checkout of all ten stations. Typically, the SRB testing is done in pieces. Not all Ethernet stations are actively powered and operational at the same time. For this reason, the worst case analysis will consider three cases:

1. All stations except AFT #1, AFT #2 and TVC are operational,
2. All stations except FWD #1 and FWD #2 are operational,
3. All stations are operational.

In all three cases the basic data flow will consist of offered commands from the VAX, command acknowledgements from destination station and archive data transmissions with VAX acknowledgements.

For each case the total amount of expected archival data is estimated using the data in Table 4.0. The estimated worst case input bit rate is derived from the number of analog and digital data sources possible. As may be observed in Table 4.0, there are a number of 86/14 boards with a varying number of analog and digital data sources. Each data source has an equivalent of 200 bits per second data generation rate. For a station, one of the AFT stations for example, the total accumulated worst case input data rate is found by adding the worst case data rates for each board. This total for the AFT station is 4400 Bps (35,200 bps).

One aspect of the archiving of data is that only changes in data sensor outputs are recorded in the stations 86/14 board and transmitted over the Ethernet. Thus, it is likely that only 50% of the 4400 Bps would need to be transmitted. The worst case analysis

TABLE 4.0

86/14 BOARD INPUTS AND FILL TIMES (ESTIMATES)

STATION	86/14 BOARD #	CPU #	ESTIMATED # INPUTS	EQUIVALENT INPUT TOT DIGITAL	WORST CASE ESTIMATED FILL TIME SECONDS	WORST CASE INPUT BIT RATE bps / Bps
FWD & EMU	1	3	76A	76	0.04	15200 / 1900
	2	4	12D	12	0.24	2400 / 300
	3	5	0	0	1.0	------
	4	6	0	0	1.5	------
	5	7	16A	16	0.2	3200 / 400
	6	8	16A + 12D	28	0.1	5600 / 700
	7	9	16A + 24D	40	0.16	8000 / 1000
AFT	1	3	76A	12	0.04	15200 / 1900
	2	4	12D	76	0.24	2400 / 300
	3	5	16A + 16D	32	0.08	6400 / 800
	4	6	16A + 8D	24	0.12	4800 / 600
	5	7	16A + 16A	32	0.08	6400 / 800
	6	8	0	0	-----	------
TVC	1	3	16D+8D+8D+8D +24D*3	112	0.02	22400 / 2800
	2	4	16A * 4	64	0.04	12800 / 1600
	3	5	16A+4D+16A*3	68	0.04	13600 / 1700
	4	6	24D	24	0.12	4800 / 600
IEA	1	3	16A*3 + 24D	72	0.04	14400 / 1800
	2	4	76A	76	0.04	15200 / 1900
	3	5	16A*3	48	0.06	9600 / 1200
ASA & VSWR	1	3	0	0	-----	------
DFI	1	3	?	?	0.5	------
	2	4	?	?	0.7	------
	3	5	?	?	0.9	------

A = Analog data line (Sampled at a 25 sample per second rate and converted to an 8 bit number per sample). Thus, A has 200 bps equivalent Digital rate

D = Digital channel (Number of bits input). Thus, D has 200 bps equivalent Digital rate.

b = bits

B = bytes

Note that there are 2 FWD stations, 1 Emulator, and 2 AFT stations.

will consider both the 50% and 100% situation. In addition, it should be noted that the archived data is buffered into 1500 Byte packets for transmission over Ethernet.

The outlying stations which interface to the SRB transmit archive data to the VAX units. Thus, if a 1500 Byte packet at the TVC station is ready for transmission, it is transmitted via Ethernet to the VAX. The transmission rate over Ethernet is at a 1.25 MByte rate and takes approximately 1.2 milliseconds. Of course, there will also be VAX command transmissions with receiving station acknowledgements on the Ethernet at random times during archival data transfers as well as the VAX automatic acknowledgements.

The archival data gathering takes place on a regular basis. For example, the analog data sources are sampled at a 25-sample per second rate and each sample is converted to an 8-bit number. (The equivalent bit rate of an analog data line is 200 bits per second. Assuming data sources that are digital transducer outputs, the same data rate has been assumed.) Thus, it is conceivable that the archiving data transfer for the busiest 86/14 board could be 1500 Bytes every .535 seconds. However, only changes in data are transmitted. Thus it is probable that say only 50% of the archiving data source data requires transmission. Based on the above description, the analysis for Case 1 is summarized in Table 4.1.

TABLE 4.1

Assuming 20% Overhead for 1500 Byte Packets

Active Stations Providing Source Data	Total Worst Case Data Generation Rate (Bytes Per Second)	Total Archive Data and OVHD to be Transmitted to VAX Assuming 50% Need	Total Archive Data and OVH to be Transmitted to VAX Assuming 100% Need
Case 1 Worst Case Analysis Archival Transfers Only			
FWD #1	4,300		
FWD #2	4,300	10,680 Bytes Per Second	21,360 Bytes Per Second
EMU	4,300		
IEA	4,900	CONCLUSION: THERE IS MORE THAN SUFFICIENT CAPACITY ON BOTH ETHERNET AND VAX I/O, EVEN ON ONE LAN.	
Case 2 Worst Case Analysis			
AFT #1	4,400		
AFT #2	4,400		
EMU	4,300	14,820 Bytes Per Second	29,640 Bytes Per Second
IEA	4,900	CONCLUSION: THERE IS MORE THAN SUFFICIENT CAPACITY ON BOTH ETHERNET AND VAX I/O, EVEN ON ONE LAN.	
Case 3 Worst Case Analysis			
2 FWD			
2 AFT	33,300 aggregate	19,980 Bytes Per Second	39,960 Bytes Per Second
EMU		CONCLUSION: THERE IS MORE THAN SUFFICIENT CAPACITY ON BOTH ETHERNET AND VAX I/O, EVEN ON ONE LAN.	
TVC			
IEA			

One may observe that for only archival data transfers with no commands from VAX, no transfers from TVC to AFT stations, and no acknowledge messages, the single LAN and VAX I/O's have sufficient capacity.

Consider now a series of 60 commands, the required 60 acknowledge messages, and the VAX acknowledgements to archive messages (about 30 Ethernet messages). Since the commands might be buffered into longer packets, an estimate of 500 Byte packets is used for VAX commands. The acknowledgements are assumed to be 100 Byte packets. The data transfer requirements are now increased by an additional 6600 Bytes, still well within the capacity of the VAX machine I/O capability and the Ethernet requirements.

Consider, however, the situation that might arise when all stations want to transmit 1500 Byte packets at the same time due to a quirk in timing. There will be a heavy period of collisions, and then the system should sort itself out. Initially, there will be approximately 30 packets to be transmitted. Assuming all are involved in an average of three collisions, then the time to transmit is upper-bounded by

30 pkts x (3 colls + 1 successful retry) x 2.1 millisecond/pkt

+

30 ack pkts x (3 colls + 1 successful retry) x .7 millisecond/pkt

or a time of 336 milliseconds. (For the equation above, a .20 millisecond wait period is assumed, pkt implies packet, ack implies acknowledge, and coll implies collision.) How long will it be before the next set of archival data is ready to be transmitted? The answer is found in the data of Table 4.1. Each 86/14 board accumulates the archive information into 1500 Byte packets to transmit. The worst case data arrival or generation rate is the 2800 Bytes per second which is shown in the worst case input bit rate column of Table 4.1. Thus, in 530 milliseconds, a second packet will be ready. (This is the first 86/14 board listed for the TVC station.) The first initial burst of simultaneous archival data transmission should be accomplished in a little over half the time it takes to ready just one more packet. Even if 60 commands are transmitted and acknowledgements received by the VAX in this first one second period , the additional time for this would be:

60 pkts x (3 colls + 1 successful retry) x 1.08 millisecond/pkt

+

60 acks x (3 colls + 1 successful retry) x .7 millisecond/pkt

or a total of 427 milliseconds extra time. (The same assumptions as before are made for this equation.) The total time required for the first archival data and acknowledgements plus 60 commands and 60 acknowledgements under 300% collision will require 763 milliseconds. During this period there could be one new packet generated at the 530 millisecond point by the TVC 86/14 board number 1. The next three packets will be generated at the 789 millisecond point, due to a 1900 Byte per second worst case data generation assumption. Thus, there is approximately 100 milliseconds of time to transmit the additional 1500 Byte TVC packet, which would take 8.4 milliseconds even with three collisions and .200 msec average waiting between retries. The system should recover in the first one-second period, and the uneven data generation rates will smooth out the data transfer requests within the end of the first second.

4.2 Analysis Conclusions

The configuration discussed includes all archival data sources simultaneously ready to transmit 100% of the archival data (all newly changed data assumed) from 100% of the possible stations with VAX acknowledgements for each archive message and an assumed 60 commands from the VAX with the 60 required acknowledgements. A 300% collision rate is assumed. The ABACS system, even under the worst case configuration, will be able to accommodate the communication load under the single LAN assumption.

5.0 RECOMMENDATIONS FOR IMPROVING SYSTEM RESPONSIVENESS

Several considerations for improving system responsiveness are presented. The suggestions include equipment upgrading, packet size considerations, and data loading.

5.1 ABACS Environment.

The VAX 11/750 computer systems can receive about 90 Kbytes per second and can transmit about 100 Kbytes per second. Since these values are only about 8% of the Ethernet capability of 10 Mbits per second, upgrading the VAX computers to a system with a faster throughput rate will allow both an increased information transfer and a faster message interpretation and handling cycle.

In order to reduce the possibility of false collision detection, the devices attached to the Ethernet cable must be placed on appropriate 2.5 meter connection locations. Device placement on the cable must be carefully controlled to insure reflections do not add in phase to a significant degree. By using the minimum separation lengths of 2.5 meters, reflection build-up may be avoided.

5.2 System Considerations.

When a station controller is initialized for communication with the VAX control computer, the bus activity is increased significantly. The VAX sends a station about 400 Kbits of information on initialization and the station controller responds with about 240 Kbits. This transfer has been simulated as occurring over a 1 to 2 second interval. There are additional delay times associated with the initialization procedure within ABACS. Since the bus is heavily loaded during initialization, the station controllers should be 'brought on-line' when slack times occur on the bus. <u>Stations should not be initialized at the same time since the VAX may become over loaded by the increased demand.</u>

To reduce the system communication requirements, the VAX commands which are about 20 bytes should be combined to form fewer but larger Ethernet packets. This operation which is called a sequential operation in ABACS is currently performed on a limited basis. Only about 1 in 50 out-going messages from the VAX to a station are sequential operations. If the VAX commands were combined into larger single commands, then the number of acknowledge messages returned by the receiving station would also be reduced. <u>By lowering the number of messages and increasing the packet size of the messages which are transmitted, fewer collisions will occur and the efficiency of the Ethernet will be improved.</u>

If the automatic acknowledge of a packet by a receiving station could be eliminated, then the number of Ethernet transmissions is reduced to almost half the current value. The frequent transmission of small packets greatly reduces the Ethernet efficiency.

6.0 CONCLUSION

The purpose of this research has been to provide a software tool that will aid in the performance analysis of the ABACS Ethernet system in varying configurations. This program is specifically designed to meet the needs of the engineers and scientists at NASA's Huntsville Operation Support Center. The model provides for control computers which issue commands to station controllers also attached to the Ethernet cable and for control computer archival of information received from the station controller.

The report began with a detailed description of the system configuration and operations to be modeled. Figure 1.0 gives an overview of the system configuration. This establishes a basic configuration for test cases in Section 3.0. The devices which serve as the components of the network modeled are characterized in Section 1.1. Section 1.2 analyzes the traffic flow of the ABACS system. A complete description of the Ethernet protocol is then presented in Section 1.3 to provide insight into Ethernet characteristics and operations.

Section 2.0 was provided to detail the ABACS operations modeled. In addition, the performance parameters used to analyze the network were explained and a user interface with a description of how to use the simulation model was included. This section also provides the validation criteria used for this program. Using the results of many simulation runs, the offered load versus throughput were plotted in

Figure 2.1. The graph produced results very close to the research efforts of [ABA77] and [SH80]. For a heavily loaded bus with 20 devices attached and with about half the packet sizes 80 bytes and half 1500 bytes, the bus efficiency was near 85%. The bus utilization then was about 8.5 Mbits per second.

In section 3.0 the results of several simulation run were presented. To gain full appreciation of the ABACS system and performance under varying scenarios, it is necessary to read Section 3.0. In summary, the throughput of the network will approach 98% under favorable conditions. However, adverse conditions such as small packet sizes combined with a high number of stations produces a lower bound of 36.8% channel throughput. The examples within this report average a throughput rate of 85% demonstrating the robust features of Ethernet. The efficiency of this network remained high since the number of stations was low and the packet size was large on the average.

A worst case analysis if the expected Ethernet load was presented in Section 4.0. From the analysis configuration of this section, the ABACS system, even in the worst possible case, will be able to accommodate the communication load under the single LAN assumption. In comparison, the simulation run configuration for Run 2 in Appendix III closely parallels the analysis configuration of Section 4.0. The results of simulation run 2 also demonstrate that the ABACS VAX computer will be able to transmit and receive information without loss of data under this worst case scenario. Since only about 10% of total Ethernet cable capacity was utilized, the offered load must be increased significantly before the efficiency of the Ethernet cable is reduced.

Several suggestion for improving system responsiveness were proposed in Section 5.0 to reduce inefficiency in the network. This included VAX control computer upgrading, station spacing on the Ethernet cable, station initialization, and reducing the number of transmitted messages on the Ethernet cable.

Finally, the stated objectives of this research have been reached. As always, additional features could be added to this model to produce improvements. The current model provides an accurate and valid performance analysis for the system configuration stated.

7.0 REFERENCES AND BIBLIOGRAPHY

[ABA77]
Agrawala, A. K., R. M. Bryant, and J. Agre, 'Analysis of an Ethernet-Like Protocol', Computer Networking Symposium, December 1977, page 104-111.

[FKP85]
Fritz, James S., Charles F. Kaldenbach, and Louis M. Progar, Local Area Networks: Selection Guidelines, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

[HE82]
Heyman, D. P., 'An Analysis of the Carrier-Sense Multiple Access Protocol', The Bell System Technical Journal, October 1982, Vol. 61, No. 8, page 2023-2051.

[IN86]
Ingels, F. M., 'System Analysis for the Huntsville Operation Support Center Distributed Computer System', Contract Final Report 1986, NAS8-34906.

[KI86]
Killen, Harold B., Telecommunications and Data Communication System Design with Troubleshooting, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

[MB76]
Metcalfe, R. M. and D. R. Boggs, 'Ethernet: Distributed Packet Switching for Local Computer Networks', ACM Communications, July 1976, Vol. 19. No. 7, page 395-404.

[SA85]
Stuck, B. W. and E. Arthurs, A Computer and Communications Network Performance Analysis Primer, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

[SH80]
Shoch, John F. and Jon A. Hupp, 'Measured Performance of an Ethernet Local Network', Communications of the ACM, December 1980, Vol. 23, No. 12, page 711-721.

[SM83]
Sauer, Charles H. and Edward A. MacNair, Simulation of Computer Communication Systems, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

[ST84]
Stallings, William, Local Networks: An Introduction, Macmillan Publishing Company, New York, New York, 1984.

APPENDIX I.

ABACS BOARD LEVEL HARDWARE AND DETAILED COMMUNICATIONS

CONTENTS

ATTRIBUTES OF SYSTEM COMPUTER - iSBC 286/10

1) 8 MHz 80286 microprocessor (16/32 bit CPU)

2) 16 Megabyte physical addressability, 1 gigabyte virtual

3) 80287 numeric data coprocessor - IEAA P754 STD

4) Multibus and LBX bus capability

5) 0 wait-state synchronous interface to memory

6) Local and dual port RAM capabilities

7) 8 JEDEC 28 pin sites for 128K (8K) RAM/ 512K (64K) EPROM

8) 2 SBX bus interface connectors

9) 16 levels of vectorized interrupt control

10) Centronics-compatible paralled I/O interface

11) 2 Multiprotocol synchronous/ asynchronous serial I/O

12) Multibus interface for multimaster configuration

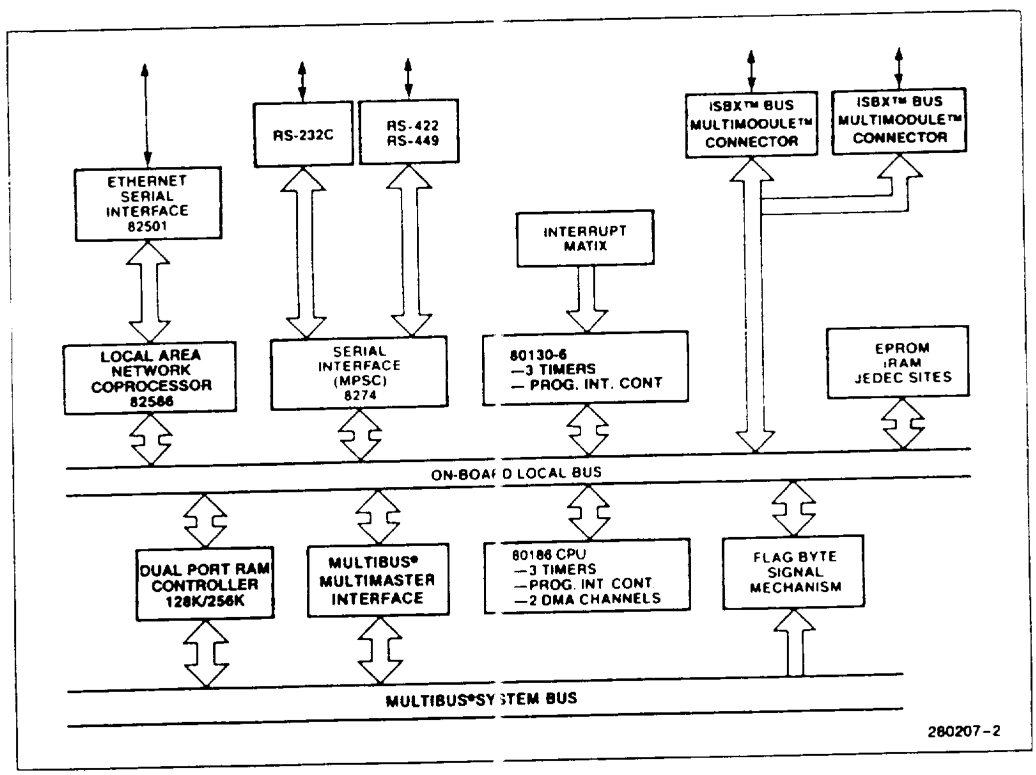
13) 3 independent 16 bit interval timers

14) 64 Kbytes of I/O addressing

ATTRIBUTES OF ETHERNET I/O COMPUTER - iSBC 186/51

1) 6 MHz 80186 microprocessor (16 bit)
2) 128 Kbytes of dual port/ local RAM
3) 6 JEDEC 24/28 pin sockets for RAM/EPROM (32K EPROM)
4) 2 SBX bus interface connectors
5) 2 serial I/O channels
6) 1 Megabyte direct memory addressability
7) 64 Kbytes of I/O addressability
8) an onboard intelligent communications controller capable of gaining access to the Ethernet serial link, formatting data into packets, and targeting the data to its destination with DMA transfer of the data from memory.
9) 3 independent 16 bit interval timers
10) 2 independent high speed DMA channels

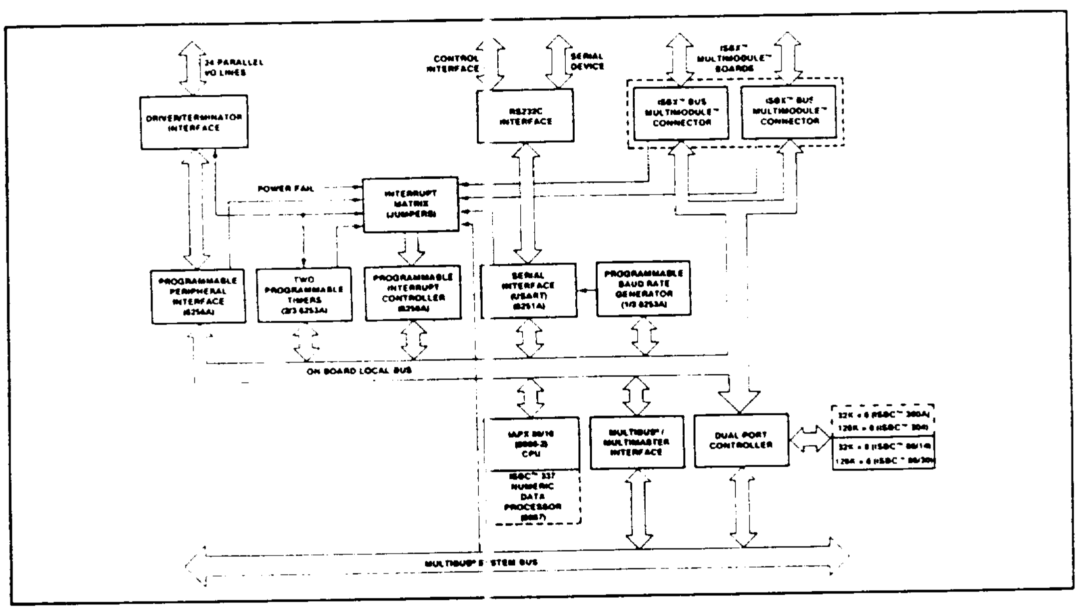
BLOCK DIAGRAM OF ETHERNET iSBC 186/51



ATTRIBUTES OF I/O COMPUTER - iSBC 86/14

1) 16 bit 8086 CPU (16 bit internal and 16 bit data lines)
2) 1 megabyte direct addressability
3) 32 bytes of on-board RAM that is configurable for local and/or dual port to multibus in part or all
4) supports additional 32 Kbytes of RAM with add-on board
5) Multibus arbitration chips
6) On board RS-232 interface
7) On board parallel interface with 24 bits divided into three 8 bit words with each word separately configurable and one word configurable with 4 bits in and 4 bits out
8) 2 SBX interface buses on board
9) 64 Kbytes of I/O addressing
10) 5 or 8 MHz jumper selectable CPU clock
11) 3 independent 16 bit interval timers
12) 9 vectored interrupt levels

BLOCK DIAGRAM OF iSBC 86/14



ORIGINAL PAGE IS
OF POOR QUALITY

STATION I/O IDENTIFICATIONS

SINCE THE SBC286 AND SBC186 BOARDS PERFORM NO UNIT UNDER TEST (UUT) INTERFACE, THE FOLLOWING INFORMATION SHOWS THE SBC86/14 BOARD I/O IDENTIFICATIONS. THE SBX INTERFACES INCLUDE:

CURRENTLY NO ARCHIVING PERFORMED:

488 - IEEE 488 INTERFACE

328 - ANALOG OUTPUT

354 - RS-232 INTERFACE

ARCHIVING MAY BE PERFORMED:

311 - ANALOG INPUT - 16 SINGLE-ENDED INPUT CHANNELS

350 - DISCRETE I/O - 3 CHANNELS WITH 8 BITS EACH
CHANNELS ARE DESIGNATED A, B, & C
A - CAN BE PROGRAMMED AS 8 BITS INPUT OR 8 BITS OUTPUT. ARCHIVED IF INPUT.
B - SAME AS A.
C - CAN BE SAME AS A OR SPLIT INTO TWO 4 BIT CHANNELS WITH 1 INPUT AND THE OTHER OUTPUT. ARCHIVED IF INPUT.

MDM- UUT DEVICE CONNECTED TO ABACS VIA A SERIAL INTERFACE.
4 MODULES EACH HAVING THREE 16 BIT CHANNELS, ALL INPUT.
2 MODULES EACH HAVING 32 ANALOG INPUT CHANNELS.
ALL MDM INPUTS ARE ARCHIVABLE.

SBXO IS NOT USED FOR TESTING. THE ONLY VALID SBX'S ARE 1 UPTO 8.

SBX 3 - 8 ARE ON AN EXTENDER BOARD THAT IS ACCESSED OVER THE MULITBUS.

SBX 1 & 2 ARE ON THE SBC86/14 BOARD AND ARE ACCESSED OVER A LOCAL BUS ON THE BOARD.

STATION I/O IDENTIFICATIONS

SBX350 KEY	PORT ADDRESS	
CH/CL/B/A	00C8	ONBOARD PPI
I-IN 0-OUT	0080	iSBX J4
	00A0	iSBX J3
	1100-11A0	iSBX EXPANDER CARD

STATION	CPU#	SBX0	SBX1	SBX2	SBX3	SBX4	SBX5	SBX6	SBX7	SBX8
FWD	3	0/I/I/0		0/0/0/0	0/0/0/0	0/0/0/0		0/0/0/0	0/0/0/0	0/0/0/0
1 & 2		(00C8)350	(0080)488	(00A0)350	(1100)350	(1120)350	(1140)354	(1160)350	(1180)350	(11A0)350
	4	0/I/I/0		0/0/0/0						
		(00C8)350	(0080)MDM	(00A0)350						
	5	0/I/I/0		0/0/0/0						
		(00C8)350	(0080)328	(00A0)350						
	6	0/I/I/0		0/0/0/0						
		(00C8)350	(0080)328	(00A0)350						
	7	0/I/I/0		0/0/0/0						
		(00C8)350	(0080)311	(00A0)350						
	8	0/I/I/0		I/0/I/0						
		(00C8)350	(0080)311	(00A0)350						
	9	0/I/I/0		I/I/I/I						
		(00C8)350	(0080)311	(00A0)350						

STATION	CPU#	SBX0	SBX1	SBX2	SBX3	SBX4	SBX5	SBX6	SBX7	SBX8
AFT	3	0/I/I/0		I/O/I/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0	0/0/0/0
1 & 2		(00C8)350	(0080)488	(00A0)350	(1100)350	(1120)350	(1140)350	(1160)350	(1180)350	(11A0)350
	4	0/I/I/0		0/0/0/0						
		(00C8)350	(0080)MDM	(00A0)328						
	5	0/I/I/0						0/0/0/0	0/0/0/0	0/0/0/0
		(00C8)350	(0080)311	(00A0)311	(1200)328	(1220)328	(1240)354	(1260)350	(1280)350	(12A0)350
	6	0/I/I/0		0/0/0/I						
		(00C8)350	(0080)311	(00A0)3500						
	7	0/I/I/0								
		(00C8)350	(0080)311	(00A0)311						
	8	0/I/I/0								
		(00C8)350	(0080)328							

STATION I/O IDENTIFICATIONS

SBX350 KEY	PORT ADDRESS	
CH/CL/B/A	00C8	ONBOARD PPI
I-IN 0-OUT	0080	iSBX J4
	00A0	iSBX J3
	1100-11A0	iSBX EXPANDER CARD

STATION	CPU#	SBX0	SBX1	SBX2	SBX3	SBX4	SBX5	SBX6	SBX7	SBX8
TVC	3	0/I/I/0 (00C8)350	(0080)488	I/I/I/0 (00A0)350	0/0/0/I (1100)350	0/0/0/I (1120)350	0/0/0/I (1140)350	I/I/I/I (1160)350	I/I/I/I (1180)350	I/I/I/I (11A0)350
	4	0/I/I/0 (00C8)350	(0080)354	(00A0)311	(1300)311	(1320)311	(1340)311	0/0/0/0 (1260)350	0/0/0/0 (1280)350	0/0/0/0 (12A0)350
	5	0/I/I/0 (00C8)350	(0080)311	0/I/0/0 (00A0)350	0/0/0/0 (1200)350	0/0/0/0 (1220)350	0/0/0/0 (1240)350	(1360)311	(1380)311	(13A0)311
	6	0/I/I/0 (00C8)350	I/I/I/I (0080)350	(00A0)328						

STATION	CPU#	SBX0	SBX1	SBX2	SBX3	SBX4	SBX5	SBX6	SBX7	SBX8
IEA	3	0/I/I/0 (00C8)350	(0080)488	(00A0)311	(1100)311	(1120)311	(1140)328	I/I/I/I (1160)350	0/0/0/0 (1180)350	(11A0)328
	4	0/I/I/0 (00C8)350	(0080)MDM	(00A0)328						
	5	0/I/I/0 (00C8)350	(0080)311	(00A0)354	(1200)311	(1220)311	(1240)328	0/0/0/0 (1360)350	0/0/0/0 (1380)350	

STATION	CPU#	SBX0	SBX1	SBX2	SBX3	SBX4	SBX5	SBX6	SBX7	SBX8
ASA	3	I/I/I/0 (00C8)350	(0080)488	0/0/0/0 (00A0)350	0/0/0/0 (1100)350	(1120)311				

STATION	CPU#	SBX0	SBX1	SBX2	SBX3	SBX4	SBX5	SBX6	SBX7	SBX8
VSWR	3	I/I/I/0 (00C8)350	(0080)488	0/0/0/0 (00A0)350	0/0/0/0					

86/14 BOARD INPUTS AND FILL TIMES (ESTIMATES)

STATION	86/14 BOARD #	CPU #	ESTIMATED # INPUTS	EQUIVALENT INPUT TOT DIGITAL	WORST CASE ESTIMATED FILL TIME SECONDS	WORST CASE INPUT BIT RATE bps / Bps
FWD & EMU	1	3	76A	76	0.04	15200 / 1900
	2	4	12D	12	0.24	2400 / 300
	3	5	0	0	1.0	------
	4	6	0	0	1.5	------
	5	7	16A	16	0.2	3200 / 400
	6	8	16A + 12D	28	0.1	5600 / 700
	7	9	16A + 24D	40	0.16	8000 / 1000
AFT	1	3	76A	12	0.04	15200 / 1900
	2	4	12D	76	0.24	2400 / 300
	3	5	16A + 16D	32	0.08	6400 / 800
	4	6	16A + 8D	24	0.12	4800 / 600
	5	7	16A + 16A	32	0.08	6400 / 800
	6	8	0	0	-----	------
TVC	1	3	16D+8D+8D+8D +24D*3	112	0.02	22400 / 2800
	2	4	16A * 4	64	0.04	12800 / 1600
	3	5	16A+4D+16A*3	68	0.04	13600 / 1700
	4	6	24D	24	0.12	4800 / 600
IEA	1	3	16A*3 + 24D	72	0.04	14400 / 1800
	2	4	76A	76	0.04	15200 / 1900
	3	5	16A*3	48	0.06	9600 / 1200
ASA & VSWR	1	3	0	0	-----	------
DFI	1	3	?	?	0.5	------
	2	4	?	?	0.7	------
	3	5	?	?	0.9	------

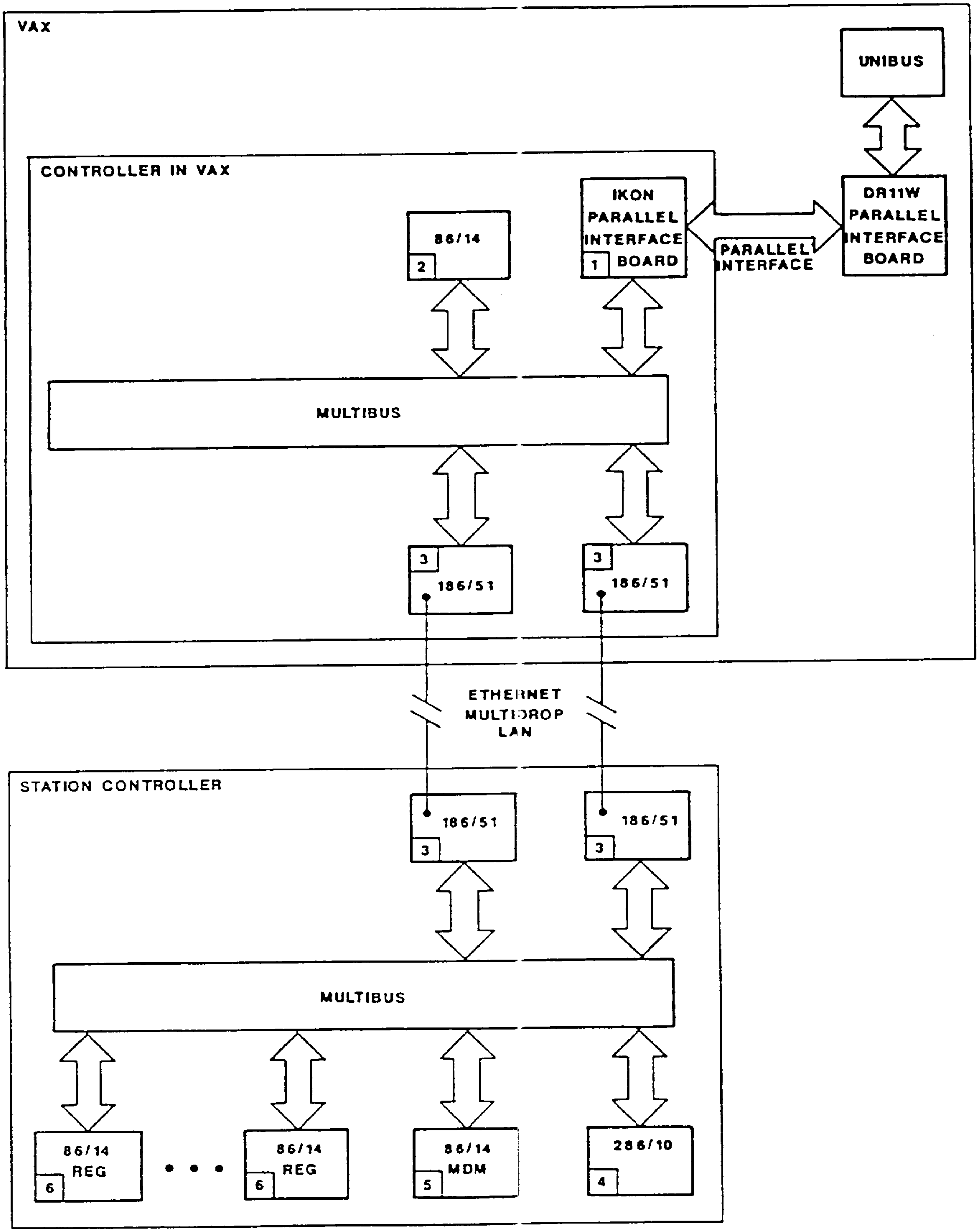
A = Analog data line (Sampled at a 25 sample per second rate and converted to an 8 bit number per sample). Thus, A has 200 bps equivalent Digital rate.

D = Digital channel (Number of bits input). Thus, D has 200 bps equivalent Digital rate.

b = bits

B = bytes

Note that there are 2 FWD stations, 1 Emulator, and 2 AFT stations.



MESSAGE FROM VAX TO STATION CONTROLLER
(HARDWARE)

CONTROLLER IN VAX (SBC 86/14)

- o IKON INTERFACE INITIALIZATION
  - SET ALL STATIONS INACTIVE IN ACTIVE STATION LIST
  - SET UP INTERRUPT FROM IKON RECEIVE BOARD
  - RESET IKON BOARDS STATUS
    - RESET DMA END FLAG
    - RESET ATTN FLAG
    - RESET PARITY ERROR FLAG
  - SET FCN1 LINE TO TELL VAX THAT CONTROLLER IS READY FOR STARTUP
  - WAIT FOR VAX'S DR11W RECEIVE LINE TO BE SET
  - SET UP FOR RECEIVE MESSAGE THROUGH IKON
    - GIVE POINTER TO MESSAGE BUFFER TO THE IKON BOARD FOR INCOMING MESSAGE FROM THE VAX
    - GIVE LENGTH OF MESSAGE TO IKON BOARD
    - RESET IKON BOARDS STATUS
    - ENABLE DMA AND ATTN INTERRUPTS
    - SET FCN1 LINE TO TELL VAX THAT THE CONTROLLER IS READY TO RECEIVE MESSAGE
    - PULSE FCN2 AND GO TO SET ATTN AND GO
  - WAIT FOR "SET STATION ID" MESSAGE FROM VAX. THIS MUST BE FIRST MESSAGE SENT. IF IT IS NOT, THE MESSAGE IS THROWN AWAY AND THE WAIT CONTINUES.
  - SET DUAL PORT MEMORY LOCATION USED BY ALL OTHER CPU BOARDS FOR THE STATION ID.
  - SET UP FOR RECEIVE MESSAGE.
- o RECEIVE MESSAGE FROM VAX - INTERRUPT 3 OCCURS
  - IF IKON DMA COMPLETE BIT IS SET
    - IF MESSAGE IS A CONNECT OR DISCONNECT
      - GET ANOTHER MESSAGE BUFFER
      - COPY MESSAGE INTO NEW MESSAGE BUFFER
      - SEND NEW MESSAGE BUFFER TO SBC186/51 BOARD #2
    - SEND MESSAGE TO SBC186/51 BOARD #1
    - GET NEW IKON RECEIVE MESSAGE BUFFER

CONTROLLER IN VAX SBC 86/14)
(CONTINUED)

- IF IKON DMA COMPLETE BIT IS NOT SET
    - REUSE SAME MESSAGE BUFFER
    - SET UP FOR RETRANSMIT
- SET UP FOR RECEIVE MESSAGE THROUGH IKON
- REENABLE INTERRUPT 3

o SEND MESSAGE TO VAX

- WAIT FOR ATTN FLAG TO BE SET AND FOR DMA END
- GIVE POINTER TO MESSAGE TO THE IKON BOARD
- GIVE THE LENGTH OF THE MESSAGE TO THE IKON BOARD
- RESET DMA END FLAG
- RESET ATTN FLAG
- RESET PARITY ERROR FLAG
- PULSE CYCL TO START DMA AND GO TO RESET REDY
- WAIT FOR REDY TO BE RESET

STATION TO STATION CONNECTION OVER ETHERNET

- RECEIVE A CONNECT MESSAGE OVER THE MULTIBUS
- FOR THE INDICATED STATION:
  - SET ACTIVE IN THE WATCHDOG TIMER LIST
  - SET THE TIME TO SEND A WATCHDOG MESSAGE TO THE STATION
  - SET THE TIME TO RECEIVE A WATCHDOG MESSAGE BEFORE WE CONSIDER THE STATION DOWN
- SEND THE CONNECT MESSAGE OVER ETHERNET TO THE STATION TO BE CONNECTED
- STATION RECEIVES A CONNECT MESSAGE OVER THE ETHERNET
- FOR THE STATION SENDING THE MESSAGE:
  - SET ACTIVE IN CONNECTED STATION LIST
  - SAVE ETHERNET AND TASK ID'S FOR MESSAGE SENDING
  - SET ACTIVE IN THE WATCHDOG TIMER LIST
  - SET THE TIME TO SEND A WATCHDOG MESSAGE TO THE STATION
  - SET THE TIME TO RECEIVE A WATCHDOG MESSAGE BEFORE WE CONSIDER THE STATION DOWN
  - GENERATE A CONNECTED MESSAGE TELLING IF THIS IS THE ONLY CONNECT TO THIS STATION OR A MULTIPLE CONNECT
  - SEND MESSAGE TO CONNECTING STATION VIA ETHERNET
- ORIGINAL STATION RECEIVES MESSAGE OVER ETHERNET FROM THE STATION CONNECTING TO
- FOR INDICATED STATION:
  - SET ACTIVE IN CONNECTED STATION LIST
  - PASS THE SINGLE OR MULTIPLE CONNECT STATUS TO THE SBC86/14
- THE SBC86/14 ONLY SENDS ONE OF THE TWO CONNECTED MESSAGES FROM THE 2 SBC186/51S BOARDS TO THE VAX OVER THE PARALLEL INTERFACE

SENDING MESSAGES OVER ETHERNET

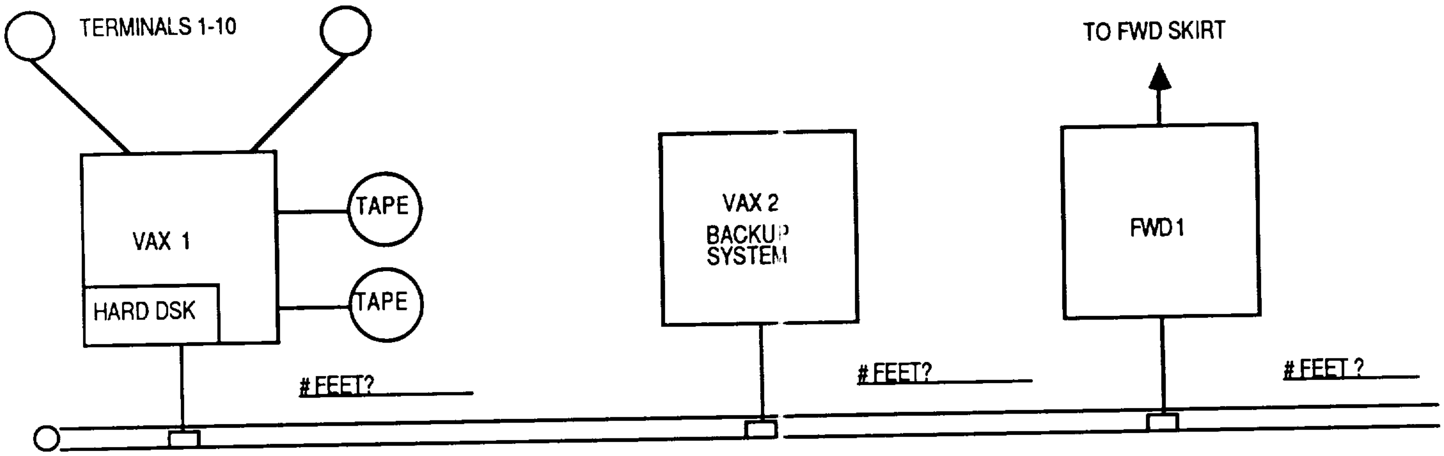
- o MOVE MESSAGE FROM DUAL PORT RAM TO LOCAL RAM
- o SET UP SYNC OF MESSAGE; INCREMENT SYNC
- o FOR STATION WE ARE SENDING TO:
  - SET ACTIVE FLAG TO SHOW WE ARE CURRENTLY SENDING TO THIS STATION
  - SAVE SYNC FOR OF MESSAGE FOR DELIVERY VERIFICATION
  - SAVE ADDRESS OF WHERE MESSAGE IS
  - SET UP A MAX TIME TO WAIT FOR REPLY OF THIS MESSAGE (700 MS)
- o TO START TRANSMISSION OVER ETHERNET, BUILD THE TRANSMIT COMMAND BLOCK:
  - SET FOR TRANSMIT COMMAND WITH SUSPEND AND INTERRUPT AFTER COMPLETION
  - SET THE ETHERNET ADDRESS OF THE RECEIVER
  - STORE MESSAGE SYNC IN TYPE FIELD
  - STORE MESSAGE LENGTH
  - SET EOF FLAG TO INDICATE THERE ARE NO MORE MESSAGES AFTER THIS ONE
- o TELL INTEL 82586 CHIP TO TRANSMIT

RECEIVING MESSAGES OVER ETHERNET

- o INTERRUPT ZERO OCCURS
- o IF TRANSMITTER FINISHED:
  - IF A TRANSMISSION ERROR OCCURRED:
    - INCREMENT RETRY COUNT
    - IF MORE THAN 15 RETRIES HAVE OCCURRED:
      - RESET RETRY TO ZERO
      - TELL 82586 TO CONTINUE NORMAL OPERATION
    - IF LESS THAN 16 RETRIES HAVE OCCURRED:
      - TELL 82586 TO RETRY SENDING MESSAGE
  - IF GOOD TRANSMISSION:
    - RESET RETRY TO ZERO
    - TELL 82586 TO CONTINUE NORMAL OPERATION
- o IF RECEIVED MESSAGE:
  - ORIGINAL MESSAGE:
    - PASS MESSAGE VIA BOARD TO BOARD COMMUNICATIONS TO CPU MESSAGE INDICATES IN MESSAGE HEADER
    - BUILD REPLY MESSAGE WITH THIS MESSAGE SYNC AND SEND IT TO THE ORIGINAL SENDER
  - REPLY MESSAGE:
    - RESET ACTIVE FLAG FOR THAT STATION
  - WATCHDOG MESSAGE:
    - RESET WATCHDOG TIMEOUT TIMER

STATION TO STATION DISCONNECT OVER ETHERNET

- RECEIVE DISCONNECT MESSAGE OVER THE MULTIBUS
- FOR THE INDICATED STATION:
    - SEND THE MESSAGE OVER ETHERNET
    - SET WATCHDOG INACTIVE
    - SET STATION INACTIVE IN ACTIVE STATION LIST
- STATION RECEIVES DISCONNECT OVER ETHERNET
- FOR THE STATION SENDING THE MESSAGE:
    - SET WATCHDOG INACTIVE
    - SET STATION INACTIVE IN ACTIVE STATION LIST



COMMUNICATIONS SUMMARY:

VAX OPERATIONS

ATLAS COMMANDS

X10 - ENTERED FOR EACH CONTROLLER ATTACHED

COMMANDS GENERATED WITHIN RANGE:

1: ______ 2: ______

SINGLE COMMANDS: ______

BLOCKED COMMANDS: ______

CMDS/ BLOCK: ______

BYTES PER COMMANDS: ______

NUM OF CMDS WHICH REQ RESPONSE: ______

NUM OF CMDS WHICH REQ NO RESPONSE: ______

VAX SIMULATION:

TRANSMIT OVERLOAD AT: ______ KB/S

RECEIVE OVERLOAD AT: ______ KB/S

VAX 2

ATLAS COMMANDS

EMULATOR

COMMANDS GENERATED WITHIN RANGE:

1: ______ 2: ______

SINGLE COMMANDS: ______

BLOCKED COMMANDS: ______

CMDS/ BLOCK: ______

BYTES PER COMMANDS: ______

NUM OF CMDS WHICH REQ RESPONSE: ______

NUM OF CMDS WHICH REQ NO RESPONSE: ______

VAX SIMULATION:

TRANSMIT OVERLOAD AT: ______ KB/S

RECEIVE OVERLOAD AT: ______ KB/S

FORWARD STATION 1

RESPONSE TO VAX

PACKET SIZE: ______

AVG DELAY TIME: ______

ARCHIVE DATA

NUM iSBC 86/14: ______

1: BUF SZ: ______
FREQ DATA ACC: ______

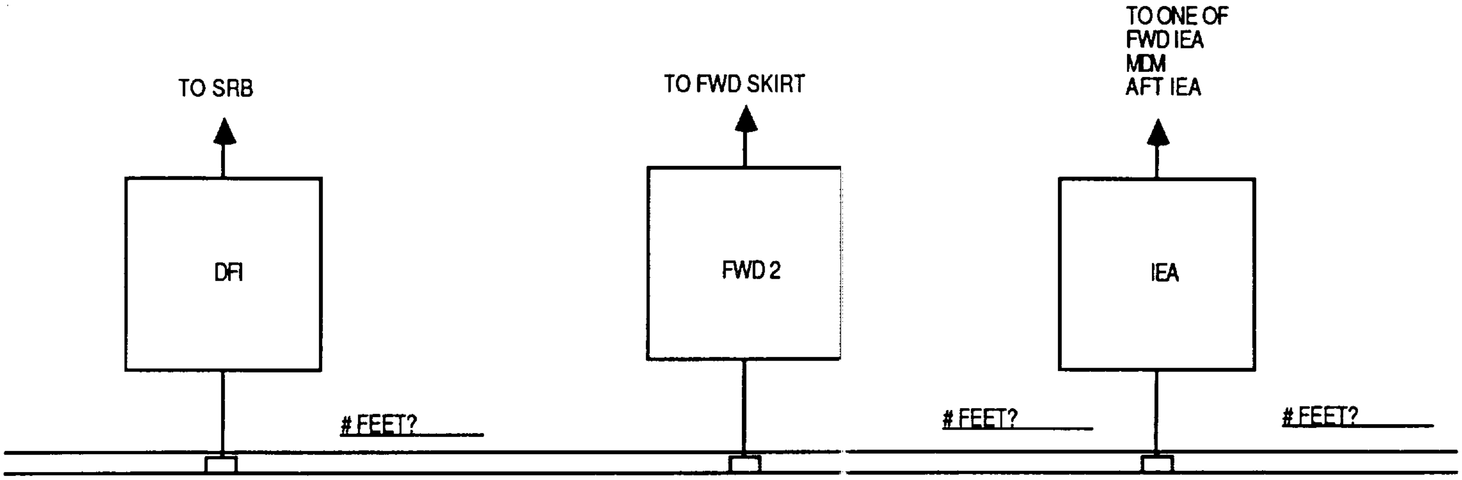
2: BUF SZ: ______
FREQ DATA ACC: ______

3: BUF SZ: ______
FREQ DATA ACC: ______

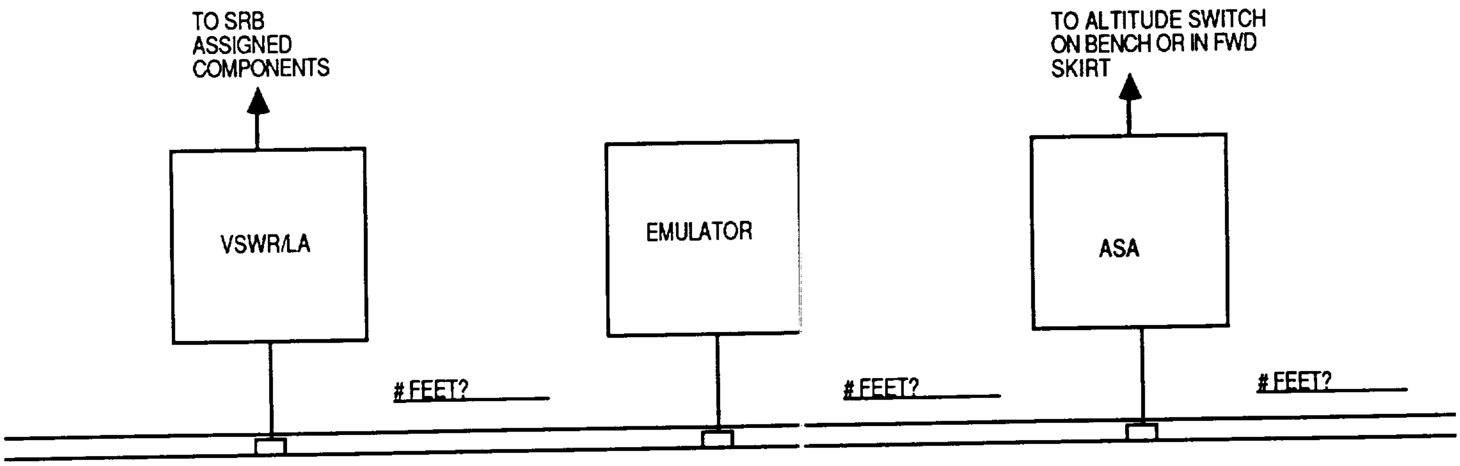
4: BUF SZ: ______
FREQ DATA ACC: ______

5: BUF SZ: ______
FREQ DATA ACC: ______

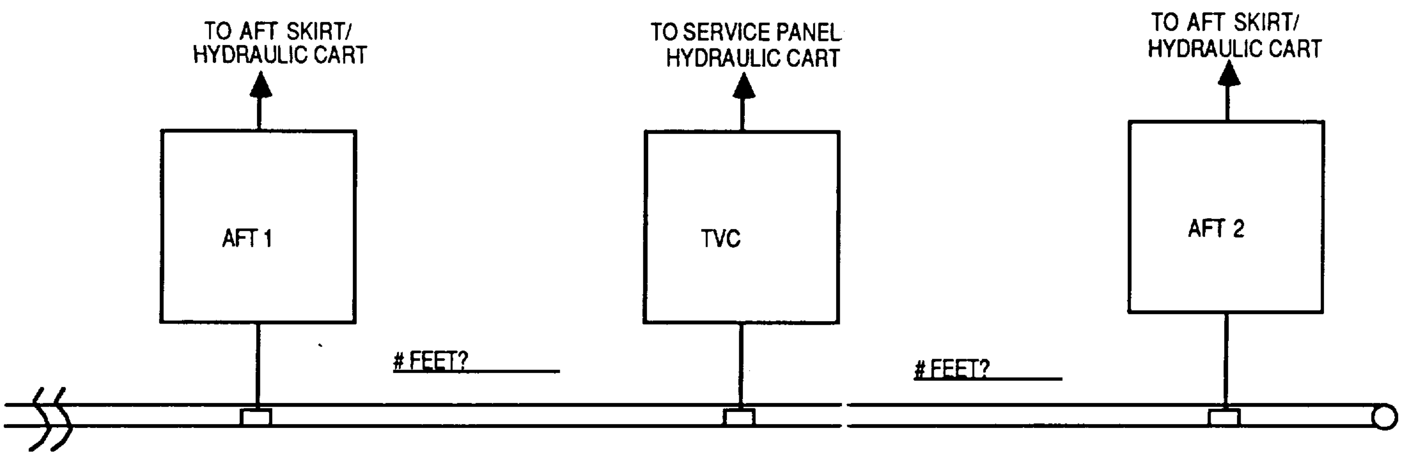
6: BUF SZ: ______
FREQ DATA ACC: ______



DFI OPERATIONS	FWD2	INTEGRATED ELECTRONIC ASSEMBLY OPERATIONS
RESPONSE TO VAX PACKET SIZE: ______ AVG DELAY TIME: ______	RESPONSE TO VAX PACKET SIZE: ______ AVG DELAY TIME: ______	RESPONSE TO VAX PACKET SIZE: ______ AVG DELAY TIME: ______
ARCHIVE DATA NUM iSBC 86/14: ______ 1: BUF SZ: ______ FREQ DATA ACC: ______ 2: BUF SZ: ______ FREQ DATA ACC: ______ 3: BUF SZ: ______ FREQ DATA ACC: ______ 4: BUF SZ: ______ FREQ DATA ACC: ______ 5: BUF SZ: ______ FREQ DATA ACC: ______ 6: BUF SZ: ______ FREQ DATA ACC: ______	ARCHIVE DATA NUM iSBC 86/14: ______ 1: BUF SZ: ______ FREQ DATA ACC: ______ 2: BUF SZ: ______ FREQ DATA ACC: ______ 3: BUF SZ: ______ FREQ DATA ACC: ______ 4: BUF SZ: ______ FREQ DATA ACC: ______ 5: BUF SZ: ______ FREQ DATA ACC: ______ 6: BUF SZ: ______ FREQ DATA ACC: ______	ARCHIVE DATA NUM iSBC 86/14: ______ 1: BUF SZ: ______ FREQ DATA ACC: ______ 2: BUF SZ: ______ FREQ DATA ACC: ______ 3: BUF SZ: ______ FREQ DATA ACC: ______ 4: BUF SZ: ______ FREQ DATA ACC: ______ 5: BUF SZ: ______ FREQ DATA ACC: ______ 6: BUF SZ: ______ FREQ DATA ACC: ______



VSWR/LOCATION AIDS OPERATIONS	EMULATOR	ALTITUDE SWITCH ASSEMBLE/SENSOR
RESPONSE TO VAX PACKET SIZE: ______ AVG DELAY TIME: ______	RESPONSE TO VAX PACKET SIZE: ______ AVG DELAY TIME: ______	RESPONSE TO VAX PACKET SIZE: ______ AVG DELAY TIME: ______
ARCHIVE DATA NUM iSBC 86/14: ______ 1: BUF SZ: ______ FREQ DATA ACC: ______ 2: BUF SZ: ______ FREQ DATA ACC: ______ 3: BUF SZ: ______ FREQ DATA ACC: ______ 4: BUF SZ: ______ FREQ DATA ACC: ______ 5: BUF SZ: ______ FREQ DATA ACC: ______ 6: BUF SZ: ______ FREQ DATA ACC: ______	ARCHIVE DATA NUM iSBC 86/14: ______ 1: BUF SZ: ______ FREQ DATA ACC ______ 2: BUF SZ: ______ FREQ DATA ACC ______ 3: BUF SZ: ______ FREQ DATA ACC ______ 4: BUF SZ: ______ FREQ DATA ACC: ______ 5: BUF SZ: ______ FREQ DATA ACC: ______ 6: BUF SZ: ______ FREQ DATA ACC: ______	ARCHIVE DATA NUM iSBC 86/14: ______ 1: BUF SZ: ______ FREQ DATA ACC: ______ 2: BUF SZ: ______ FREQ DATA ACC: ______ 3: BUF SZ: ______ FREQ DATA ACC: ______ 4: BUF SZ: ______ FREQ DATA ACC: ______ 5: BUF SZ: ______ FREQ DATA ACC: ______ 6: BUF SZ: ______ FREQ DATA ACC: ______



AFT 1 OPERATIONS	THRUST VECTOR CONTROLLER OPERATION	AFT 2
RESPONSE TO VAX PACKET SIZE: ______ AVG DELAY TIME: ______	RESPONSE TO VAX PACKET SIZE: ______ AVG DELAY TIME: ______	RESPONSE TO VAX PACKET SIZE: ______ AVG DELAY TIME: ______
ARCHIVE DATA NUM iSBC 86/14: ______ 1: BUF SZ: ______ FREQ DATA ACC: ______ 2: BUF SZ: ______ FREQ DATA ACC: ______ 3: BUF SZ: ______ FREQ DATA ACC: ______ 4: BUF SZ: ______ FREQ DATA ACC: ______ 5: BUF SZ: ______ FREQ DATA ACC: ______ 6: BUF SZ: ______ FREQ DATA ACC: ______	ARCHIVE DATA NUM iSBC 86/14: ______ 1: BUF SZ: ______ FREQ DATA ACC: ______ 2: BUF SZ: ______ FREQ DATA ACC: ______ 3: BUF SZ: ______ FREQ DATA ACC: ______ 4: BUF SZ: ______ FREQ DATA ACC: ______ 5: BUF SZ: ______ FREQ DATA ACC: ______ 6: BUF SZ: ______ FREQ DATA ACC: ______	ARCHIVE DATA NUM iSBC 86/14: ______ 1: BUF SZ: ______ FREQ DATA ACC: ______ 2: BUF SZ: ______ FREQ DATA ACC: ______ 3: BUF SZ: ______ FREQ DATA ACC: ______ 4: BUF SZ: ______ FREQ DATA ACC: ______ 5: BUF SZ: ______ FREQ DATA ACC: ______ 6: BUF SZ: ______ FREQ DATA ACC: ______

VAX OPERATIONS		FORWARD STATION 1
STARTUP CONDITIONS X 10 - ENTERED FOR EACH CONT ATTACHED CONTROLLER STARTUP TIME: ______ NUM OF PKTS SENT: ______ PACKET SIZE: ______	STARTUP CONDITIONS X 10 - ENTERED FOR EACH CONT ATTACHED CONTROLLER STARTUP TIME: ______ NUM OF PKTS SENT: ______ PACKET SIZE: ______	STARTUP RESPONSE NUM PKTS RET: ______ PACKET SIZE: ______
OTHER COMMUNICATIONS: (NON-OPERATOR ENTERED) 1) WATCHDOG TIMER: VAX: SENDS 1 80B PKT TO CONT EVERY 1 SEC CONT: RESPONDS TO VAX WITH 80B PKT 2) ACKNOWLEDGEMENT: EACH MESSAGE SENT OVER ETHERNET WILL BE AUTOMATICALLY ACKNOWLEDGED BY THE RECEIVING DEVICE.		

DFI OPERATIONS	FWD 2	INTEGRATED ELECTRONIC ASSEMBLY OPERATIONS
STARTUP RESPONSE NUM PKTS RET: ____________ PACKET SIZE: ____________	STARTUP RESPONSE NUM PKTS RET: ____________ PACKET SIZE: ____________	STARTUP RESPONSE NUM PKTS RET: ____________ PACKET SIZE: ____________

VSWR/LOCATION AIDS OPERATIONS		ALTITUDE SWITCH ASSEMBLE/SENSOR
STARTUP RESPONSE NUM PKTS RET: ____________ PACKET SIZE: ____________	STARTUP RESPONSE NUM PKTS RET: ____________ PACKET SIZE: ____________	STARTUP RESPONSE NUM PKTS RET: ____________ PACKET SIZE: ____________

AFT 1 OPERATIONS	THRUST VECTOR CONTROLLER OPERATION	AFT 2
HOTFIRE OPERATION NUM MESSAGE/SEC: ________ AVG DELAY TIME: ________ PACKET SIZE: ________	HOTFIRE OPERATION NUM MESSAGE/SEC: ________ AVG DELAY TIME: ________ PACKET SIZE: ________	HOTFIRE OPERATION NUM MESSAGE/SEC: ________ AVG DELAY TIME: ________ PACKET SIZE: ________
STARTUP RESPONSE NUM PKTS RET: ________ PACKET SIZE: ________	STARTUP RESPONSE NUM PKTS RET: ________ PACKET SIZE: ________	STARTUP RESPONSE NUM PKTS RET: ________ PACKET SIZE: ________

APPENDIX II.

SIMULATION SOURCE LISTING

ORIGINAL PAGE IS
OF POOR QUALITY

```
(*****************************************************************************
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *
 * TITLE:    ETHERNET SIMULATION FOR THE ABACS SYSTEM
 *
 * AUTHOR:   TERESA RIVES
 *
 * DATE:     APRIL 9, 1988
 *
 * PURPOSE:  PROVIDE A SIMULATION OF THE ABACS SOLID ROCKET BOOSTER TEST FACILITY
 *           ETHERNET NETWORKING SYSTEM.
 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *****************************************************************************)

(* SPECIAL INTERNAL CODES  (CALLED PART)  DESCRIBES CURRENT OPERATION        *)
(* CONT:                            -2     AFT-TVC COMMUNICATION              *)
(* VAX AND CONT:                    -1     ACKNOWLEDGEMENT                    *)
(* VAX AND CONT:                     0     WATCHDOG TIMER                     *)
(* CONT:                 1 TO NO8614       ARCHIVE MESSAGE                    *)
(* CONT:                  NO8614 + 1       RESPONSE TO VAX                    *)
(* VAX:                   CONT INDEX       COMMAND TO CONTROLLER              *)
(* VAX AND CONT:   500 + CONT INDEX        STARTUP FOR CONTROLLER             *)


PROGRAM ABACS (INPUT, OUTPUT, ABDATA(LFN=15), ABOUT (LFN=17),
               ABTEMP (LFN=18), ASF (LFN=19));

CONST
        MINDELAY = 9.6E-6; (*BUS DELAY TIME BEFORE ANOTHER PACKET MAY BE TX'D*)
        MAXNOVAX = 5; (* MAXIMUM NUMBER OF VAXS WHICH MAY BE ATTACHED *)
        MAXNOCONT = 15; (* MAXIMUM NUMBER OF CONTROLLERS WHICH MAY BE ATTACHED*)
        MAXDEVTOT = 20; (*TOTAL OF MAXNOVAX AND MAXNOCONT*)
        PLUSONE = 21; (*MAXDEVTOT + 1*)
        MAXNOBOARDS = 7; (*MAX NUMBER OF SBC 186/14 BOARDS ON A CONTROLLER *)
        SMALLPKT = 640; (*80B*8BITS=ABOUT THE SMALLEST ETHERNET PACKET ALLOWED*)
        BIGNUM = 9999.99; (*JUST A BIG NUMBER TO USE CONSISTENTLY!*)
        STARTNUM = 500; (* BASE VALUE TO INDICATE A START UP IS OCCURRING*)
        MAXCOLLS = 1.600000000E+1; (*MAX NUMBER OF COLLISIONS FOR 1 PACKET*)
        TVC = 7; (* INDEX VALUE FOR THE THRUST VECTOR CONTROLLER*)
        AFT1 = 5; (* INDEX VALUE FOR AFT (CONTROLLER 1*)
        AFT2 = 6; (* INDEX VALUE FOR AFT (CONTROLLER 2*)


TYPE
        ARAY = PACKED ARRAY [1..17] OF CHAR; (* SPACE FOR FILENAMES*)
        ARA1 = PACKED ARRAY [1..40] OF CHAR; (* SPACE FOR SIM DESCRIPTION*)
        STRTYP = PACKED ARRAY [1..40] OF CHAR; (* OPERATOR MESSAGES *)
        DEV  = PACKED ARRAY [1..4] OF CHAR;  (* SPACE FOR DEVICE NAMES*)

        VAXSTARTREC = RECORD    (* VAX START UP RECORD*)
                TXTIME : REAL;  (* NEXT TIME TO TX TO CONTROLLER*)
                BITS : INTEGER; (* NUM BITS TO SENT TO CONT *)
                STAT : INTEGER; (* CONTROL VARIABLE FOR VAX OPERATIONS*)
```

```
        COUNT: INTEGER; (* TALLIES NUM OF PACKETS SENT TO CONT*)
        NUMPKTSENT : INTEGER; (*NUMBER OF PACKETS TO SEND TO CONT*)
        PKTSENTSZ : INTEGER; (*PACKET SIZE TO SEND TO CONT*)

        NUM100: INTEGER; (*NUM OF PACKETS TO SEND ON START UP FINISH*)
        PKT100SZ: INTEGER; (* PACKET SIZE ON START UP*)

END;

ATLASRECORD = RECORD
        R1 : REAL;    (* RANGE 1 - GEN NEXT TX TIME*)
        R2 : REAL;    (* RANGE 2 - GEN NEXT TX TIME*)

        SP : INTEGER;(* PERCENTAGE OF SINGLE COMMANDS*)
        BP : INTEGER;(* PERCENTAGE OF BLOCK COMMANDS*)
        BC : INTEGER;(* NUMBER OF COMMANDS/ BLOCK*)
        COMSZ: INTEGER; (*AVERAGE NUMBER OF BYTES/ COMMAND*)
        NUMBCTX: INTEGER; (*SET IF PKT SZ ON BLOCKED CMD TOO LARGE*)
        C: ARRAY [1..2] OF INTEGER; (*TALLIES CMDS SENT SO FAR*)
        NEXT: INTEGER; (*NEXT TYPE COMMAND TO SEND*)

        TXTIME: REAL;(*GENERATED TX TIME FOR NEXT TRANS TO CONT*)
        BITS: INTEGER;(*NUMBER OF BITS TO TRANS IN NEXT PACKET*)
END;

VAXRECORD = RECORD
        VC : ARRAY [1..MAXNOCONT] OF ATLASRECORD;
        VS : ARRAY [1..MAXNOCONT] OF VAXSTARTREC;

        CONNECT: ARRAY [1..MAXNOCONT] OF INTEGER; (* CONT CONNECTED*)
        OKTX: ARRAY [1..MAXNOCONT] OF INTEGER; (* OK TO TX TO CONT*)

        NUMRX: INTEGER; (*USED ON VAX SIM - MAX AMT VAX CAN RECEIVE*)
        NUMTX: INTEGER; (*USED ON VAX SIM - MAX AMT VAX CAN TRANSMIT*)
        ACK: REAL; (*ACKNOWLEDGE TIME*)
        NOACK :INTEGER; (*NUMBER OF QUEUED ACKS*)

        VAXDOG: REAL; (*WATCHDOG TIMER NEXT TIME TO TX *)
        WDTXTM : ARRAY [1..MAXNOCONT] OF REAL; (*NEXT WATCH DOG
                                                TRANSMIT TIME TO CONT*)
        WHODOG: INTEGER; (*CONTROLLER WHICH RECEIVED WD MESSAGE*)
        PKTDOG: INTEGER; (*WATCHDOG PACKET SIZE *)

        TXDATA: INTEGER; (*TOTAL AMT TRANSMITTED IN A SECOND*)
        RXDATA: INTEGER; (*TOTAL AMT TRANSMITTED IN A SECOND*)
END;

BOARDRECORD = RECORD
        FILLTM: REAL; (*FREQ DATA WILL ACCUMULATE IN ARCHIVE BUFFER*)
        ARCSZ: INTEGER; (*ARCHIVE BUFFER SIZE*)
        TXTM: REAL; (*NEXT TX TIME FOR ARCHIVE BOARD*)
END;

CONTSTARTREC = RECORD    (*CONTROLLER START UP RECORD*)
        TXTIME : REAL;   (*NEXT TRANSMIT TIME TO VAX*)
        BITS : INTEGER;  (*NUM BITS SENT TO VAX*)
```

```
        STAT : INTEGER; (*CONTROL VARIABLE FOR CONT OPERATIONS*)

        STARTTIME: REAL; (* TIME CONTROLLER COMES ON-LINE*)
        NUMPKTSRET : INTEGER; (*NUM PKTS TO SEND TO VAX*)
        PKTSZRET : INTEGER; (*PKT SZ TO SEND TO VAX*)
        COUNT : INTEGER; (*TALLIES NUM PKTS SENT TO VAX*)

        ACKSRET: INTEGER; (*SPECIAL ACK RETURNED TO VAX*)
        ACKPKTSZ : INTEGER; (*SPECIAL ACK PKT SZ*)
END;

CONTRECORD = RECORD
        CS : CONTSTARTREC; (*CONT START UP RECORD*)
        INSTART : INTEGER; (*PERFORMING START UP NOW VARIABLE*)
        NUMRET: INTEGER; (*NUMBER OF COMMANDS LEFT TO RESPOND TO*)
        RESPTM: REAL; (*RESPONSE TRANSMIT TIME *)
        BITS: INTEGER; (*NUMBER OF BITS TO SEND TO VAX AS RESPONSE*)
        RSP: ARRAY [1..2] OF INTEGER; (*RATIO OF COMMANDS REQUIRING*)
                                      (*RESPONSE TO THOSE WHICH DON'T*)
        C: ARRAY [1..2] OF INTEGER; (*TALLIES CMDS SENT SO FAR*)
        NEXT: INTEGER; (* NEXT TYPE OF COMMAND TO SEND*)

        CONNTO : INTEGER; (*VAX THIS CONTROLLER CONNECTED TO*)
        ACK: REAL; (*ACKNOWLEDGE TIME*)
        NOACK: INTEGER; (*NUMBER OF QUEUED ACKS*)
        AD: REAL; (*AVERAGE DELAY BEFORE RESPONDING TO VAX*)
        RESPSZ: INTEGER; (*PKT SZ FOR CONT RESPONSE TO VAX CMD*)
        NO8614: INTEGER; (*NUMBER OF ISBC86/14 BOARDS ON CONTROLLER*)

        ARC: ARRAY [1..MAXNOBOARDS] OF BOARDRECORD; (*ARCHIVE OP*)

        CONTDOG: REAL; (* RESPONSE TO VAXDOG TIME*)
        PKTDOG: INTEGER; (*WATCHDOG PACKET SIZE*)

        MSGSEC : INTEGER; (*NUM MSG PER SEC ALLOWED ON HOTFIRE*)
        DLYTM: REAL;  (* DELAY TIME BEFORE RESPONDING WITH HOTFIRE MSG*)
        PKTCOMMSZ : INTEGER; (* PKT SZ ON HOTFIRE COMMUNICATION*)
        TALKTO : INTEGER; (* ON HOTFIRE WHO DOES CONT TALK TO?*)
        NUMSENT : INTEGER; (*TALLIES NUMBER SENT*)
        COMMTM: REAL; (*SPECIFIES NEXT TRANSMIT TIME*)

END;

STATREC = RECORD
        DISTANCE: REAL; (*DISTANCE OF DEVICE FROM SOME REF. POINT*)
        NOCOLS: INTEGER; (*TOTAL NUMBER OF COLLISIONS ON DEVICE*)
        NUMCOLS: INTEGER; (*NUMBER OF PACKET COLLISIONS*)
        NODER: INTEGER; (* NUMBER OF DEFERS *)
        COLTIME: REAL; (* COLLISION WAITING TIME*)
        DERTIME: REAL; (* DEFER WAITING TIME*)
        MINWAIT: REAL; (*MIN WAIT TIME FOR ANY PACKET*)
        MAXWAIT: REAL; (*MAXWAIT TIME FOR ANY PACKET*)
        PKTSTX: INTEGER; (* NUM OF PACKETS TX*)
        PKTSRX: INTEGER; (* NUM OF PACKETS RX*)
        ACKSTX: INTEGER; (* NUM OF ACKS TX*)
        MAXPKTCOLS: INTEGER; (*MAX NUM OF TIMES ANY PKT COLLIDED*)
```

```
        MAXCOLTIME: REAL; (*MAX AMT TIME ANY ONE PKT SPENT IN A COLL*)
        PKTCOLTIME: REAL; (*AMT OF TIME PKT SPENT IN A COLLISION*)
    END;

VAR
    NOVAX: INTEGER; (*NUMBER OF VAX COMPUTERS ATTACHED *)
    NOCONT: INTEGER; (*NUMBER OF CONTROLLERS ATTACHED *)
    VAX: ARRAY [1..MAXNOVAX] OF VAXRECORD;   (* VAX INFORMATION*)
    CONT: ARRAY [1..MAXNOCONT] OF CONTRLCORD;(* CONTROLLER INFORMATION*)
    STAT: ARRAY [1..MAXDEVTOT] OF STATREC; (* KEEPS DEVICE STATISTICS*)
    TX: ARRAY [1..MAXDEVTOT] OF REAL; (* NEXT TRANSMIT TIME OF ALL DEVICES*)
    PART: ARRAY [1..MAXDEVTOT] OF INTEGER;(*OPERATION OF NEXT TRANSMISSION*)
    PDELAY: ARRAY [1..MAXDEVTOT,1..MAXDEVTOT] OF REAL;(*PROPAGATION DELAYS*)
    BIT: ARRAY [1..MAXDEVTOT] OF INTEGER; (*NUMBER OF BITS FOR NEXT TX*)
    DAT: PACKED ARRAY [1..5] OF ARAY; (* FILENAMES *)
    NAM: PACKED ARRAY [1..PLUSONE] OF DEV; (* DEVICE NAMES *)
    RXMAX: ARRAY [1..MAXNOVAX] OF INTEGER; (* AMT RECEIVED FOR VAX*)
    TXMAX: ARRAY [1..MAXNOVAX] OF INTEGER; (* AMT TRANSMITTED FOR VAX*)
    RXSEC: ARRAY [1..MAXNOVAX,1..100] OF INTEGER;
    TXSEC: ARRAY [1..MAXNOVAX,1..100] OF INTEGER;
    TMLINE: INTEGER; (*KEEPS CURRENT SECOND COUNT *)
    HOTFIRE: INTEGER; (*HOTFIRE OPERATION TO BE PERFORMED IF SET *)
    NUMHOT: INTEGER; (*NUMBER OF HOTFIRE MSG TX IN A SECOND *)
    HOTMIN: INTEGER; (*LOWEST NUMBER OF HOTFIRE MSG TX IN A SEC*)
    TOTMSGSEC: INTEGER; (*TOTAL MESSAGES PER SECOND EXPECTED TO BE TX'D*)
    HOTSTART : INTEGER; (*OK TO BEGIN COUNTING HOTFIRE MSGS*)
    HOTFAIL : INTEGER;  (* NUM MESSAGES TX'D IN SEC < TOTMSGSEC*)
    ABDATA, ABOUT, ABTEMP, ASF: TEXT; (* FILENAMES *)
    SIMTIME: REAL; (*NUM SECONDS PROGRAM WILL SIMULATE*)
    DESCRIPT: ARRAY[1..5] OF ARA1; (* USER DATA FILE NAMES*)
    EBUSRATE: REAL;  (*ETHERNET BUS TRANSFER RATE*)
    IDUM, RANNUM: INTEGER; (*USER ENTERED RANSOM NUMBERS*)
    GLIY: INTEGER;  (* VAR FOR RANDOM NUMBER GENERATOR*)
    GLIR: ARRAY[1..97] OF INTEGER; (*VAR FOR RANDOM NUMBER GENERATOR*)
    FN: INTEGER;  (* FILE NUMBER SELECTED BY OPERATOR*)
    SL: INTEGER; (* OPERATOR MENU SELECTION VARIABLE*)
    CLOCK: REAL; (* BUS CURRENT TIME *)
    CURTIME: REAL; (* REFLECTS TIME A NEW TRANSMIT BEGINS - CLOCK UPDATED*)
    TOTBITS: REAL; (*TOTAL BITS OFFERED TO THE BUS *)
    TOTDATA: REAL; (* TOTAL DATA PACKETS OFFERED TO BUS*)
    BADOFF: REAL; (*INCORRECT CALCULATION OF OFFERERD LOAD*)
    TOTCOLS: INTEGER; (*TOTAL COLLISIONS ON BUS*)
    TOTPKTSTX: INTEGER; (*TOTAL PACKETS TRANSMITTED *)
    BUSBUSY: REAL; (*ACTUAL USED TIME OF BUS - INCLUDES JAMS, MIN DLY, ETC*)
    USAGE: REAL; (* BUS USED TIME DUE TO PKTS BEING TX'D*)
    IDLE: REAL; (* BUS NOT OCCUPIED TIME*)
    SIMTHRUPUT: REAL; (*THROUGHPUT RATE OF BUS FROM NUMBERS TALLIED*)
    OFFLOAD: REAL;  (* OFFERED LOAD OF BUS*)
    EFFICIENCY: REAL; (* SIMTHRUOUT/ OFFLOAD *)
    THEORETICAL: REAL; (*THROUGHPUT RATE OF CSMA PROTOCOL USING FORMULA*)
    TXIX : INTEGER; (* CURRENT INDEX FOR DEVICE WHICH IS TRANSMITTING*)
    ISTAT: INTEGER; (* STATUS VARS RETURNED WHEN OPENING A FILE*)
    ACOLL: INTEGER; (* A COLLISION WITH ANOTHER PACKET DID OCCUR*)
    LAST : INTEGER; (* KEEPS TRACK OF LAST TX OPERATION PERFORMED*)
    I : INTEGER;  (* FOR LOOP INDEX VAR*)
```

```
        OPCHAR: CHAR; (* DETERMINES IF SUMMARY DATA IS TO BE ADDED TO CHARTS*)
        NEWPKTCOLS: INTEGER;


(*****************************************************************************)

PROCEDURE INITIALIZE;

(*****************************************************************************)

(* INITIALIZES ALL VARIABLES TO A KNOWN STATE *)

VAR
I, J, K: INTEGER;

BEGIN
        (* USER FILENAMES*)
        DAT[1] := 'ABDATA1             ';
        DAT[2] := 'ABDATA2             ';
        DAT[3] := 'ABDATA3             ';
        DAT[4] := 'ABDATA4             ';
        DAT[5] := 'ABDATA5             ';

        (* DEVICE ABBREVIATIONS *)
        NAM[1] := 'FWD1';
        NAM[2] := 'FWD2';
        NAM[3] := 'DFI ';
        NAM[4] := 'EMU ';
        NAM[5] := 'AFT1';
        NAM[6] := 'AFT2';
        NAM[7] := 'TVC ';
        NAM[8] := 'IEA ';
        NAM[9] := 'ASA ';
        NAM[10] := 'VSWR';
        NAM[11] := 'SPC1';
        NAM[12] := 'SPC2';
        NAM[13] := 'SPC3';
        NAM[14] := 'SPC4';
        NAM[15] := 'SPC5';
        NAM[16] := 'VAX1';
        NAM[17] := 'VAX2';
        NAM[18] := 'SPV1';
        NAM[19] := 'SPV2';
        NAM[20] := 'SPV3';


        (* GENERAL VARIABLES INITIALIZED *)
        TXIX := 1;
        ACOLL := 0;
        TOTCOLS := 0;
        TOTBITS := 0.0;
        TOTDATA := 0.0;
        BADOFF := 0.0;
        TOTPKTSTX := 0;
        BUSBUSY := 0.0;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
USAGE := 0.0;
IDLE := 0.0;
LAST := 0;
NOVAX := 0;
NOCONT := 0;
NEWPKTCOLS := 0;
SIMTIME := 0.0;
EBUSRATE := 0;
HOTFIRE := 0;
NOVAX := 0;
NOCONT := 0;
IDUM := 0;
RANNUM := 0;

(* VAX VARIABLES INITIALIZED*)
FOR I := 1 TO MAXNOVAX DO
    WITH VAX[I] DO
    BEGIN
        FOR J:= 1 TO MAXNOCONT DO
           BEGIN
           CONNECT[J] := 0;
           OKTX[J] := 0;
                WITH VC[J] DO
                   BEGIN
                     R1 := 0.0;
                     R2 := 0.0;
                     SP := 0;
                     BP := 0;
                     BC := 0;
                     COMSZ := 0;
                     NUMBCTX := 0;
                     TXTIME := 0.0;
                     NEXT := 0;
                     C[1] := 0;
                     C[2] := 0;
                     BITS :=0;
                   END;
                WITH VS[J] DO
                   BEGIN
                     STAT := 0;
                     COUNT := 0;
                     NUMPKTSENT := 0;
                     PKTSENTSZ := 0;
                     NUM100 := 20;
                     PKT100SZ := 1500;
                   END;
                WDTXTM[J] := 1.0;
             END;
        NUMRX := 0;
        NUMTX := 0;
        ACK := BIGNUM;
        NOACK := 0;
        VAXDOG := BIGNUM;
        PKTDOG := 0;
        WHODOG := 0;
        RXDATA := 0;
```

```
        TXDATA := 0;
        RXMAX[I] := 0;
        TXMAX[I] := 0;
        TMLINE := 1;
      END;

(* CONTROLLER VARIABLES INITIALIZED *)
FOR I := 1 TO MAXNOCONT DO
   WITH CONT[I] DO
        BEGIN
                NUMRET := 0;
                RESPTM := 0.0;
                BITS := 0;
                RSP[1] := 0;
                RSP[2] := 0;
                C[1] := 0;
                C[2] := 0;
                NEXT := 0;
                CONNTO := 0;
                ACK := BIGNUM;
                NOACK := 0;
                AD := 0.0;
                RESPSZ := 0;
                INSTART := 0;
                WITH CS DO
                   BEGIN
                     STAT := 0;
                     COUNT := 0;
                     STARTIME := 0.0;
                     NUMPKTSRET := 0;
                     PKTSZRET := 0;
                     ACKSRET:= 0;
                     ACKPKTSZ := 0;
                   END;
                NO8614 := 0;
                FOR J:= 1 TO MAXNOBOARDS DO
                     WITH ARC[J] DO
                          BEGIN
                                 FILLTM := 0.0;
                                 ARCSZ := 0;
                                 TXTM := 0.0;
                          END;
                CONTDOG := BIGNUM;
                PKTDOG := 0;
                MSGSEC := 0;
                DLYTM := 0.0;
                PKTCOMMSZ := 0;
                TALKTO := 0;
                NUMSENT := 0;
                COMMTM := BIGNUM;
        END;

        HOTFIRE := 0;
        NUMHOT := 0;
        HOTMIN := STARTNUM;
```

```
            TOTMSGSEC := 0;
            HOTSTART := 0;
            HOTFAIL := 0;

        (* STATISTICS VARIABLES INITIALIZED *)
        FOR I:= 1 TO MAXDEVTOT DO
            BEGIN
               TX[I] := BIGNUM;
               PART[I] := 0;
               BIT[I] := 0;
               WITH STAT[I] DO
                    BEGIN
                      DISTANCE:= 0.0;
                      NOCOLS:= 0;
                      NUMCOLS:= 0;
                      NODER := 0;
                      COLTIME:= 0.0;
                      DERTIME:= 0.0;
                      MINWAIT:= BIGNUM;
                      MAXWAIT:= 0.0;
                      PKTSTX:= 0;
                      PKTSRX:= 0;
                      ACKSTX := 0;
                      MAXPKTCOLS := 0;
                      MAXCOLTIME := 0.0;
                      PKTCOLTIME := 0.0;
                    END;
            END;

        FOR I:= 1 TO MAXNOVAX DO
           FOR J:= 1 TO 100 DO
            BEGIN
               RXSEC[I,J] := 0;
               TXSEC[I,J] := 0;
            END;
END;

(**************************************************************************)

PROCEDURE GETDATA;

(**************************************************************************)

(* READS IN USER SELECTED SIMULATION DATA FILE *)

VAR
        I, J, K: INTEGER;

BEGIN
        (* READ GENERAL INFORMATION FROM DATA FILE SELECTED BY USER *)
        READLN(ABDATA,DESCRIPT[FN]);
        READLN(ABDATA,IDUM);
        RANNUM := IDUM;
        IDUM := 0 - IDUM;
        READLN(ABDATA,SIMTIME);
        READLN(ABDATA,EBUSRATE);
```

```
READLN(ABDATA,HOTFIRE);
READLN(ABDATA,NOVAX);
READLN(ABDATA,NOCONT);

(* READ IN VAX PARAMETERS *)
FOR I:= 1 TO MAXNOVAX DO
    WITH VAX[I] DO
        BEGIN
        FOR J:= 1 TO MAXNOCONT DO
           BEGIN
                READLN(ABDATA, CONNECT[J]);
                IF CONNECT[J] = 1 THEN
                  BEGIN
                    VAX[I].OKTX[J] := 0;
                    CONT[J].CONNTO := I;
                  END;
           END;
        FOR J:=1 TO MAXNOCONT DO
                  BEGIN
                    READLN(ABDATA,VS[J].NUMPKTSENT);
                    READLN(ABDATA,VS[J].PKTSENTSZ);
                    IF CONT[J].CONNTO = I THEN
                      BEGIN
                        CONT[J].CS.ACKSRET := VS[J].NUMPKTSENT;
                        CONT[J].CS.ACKPKTSZ := VS[J].PKTSENTSZ;
                      END;
                    WITH VC[J] DO
                      BEGIN
                        READLN(ABDATA,R1);
                        READLN(ABDATA,R2);
                        READLN(ABDATA,SP);
                        READLN(ABDATA,BP);
                        READLN(ABDATA,BC);
                        READLN(ABDATA,COMSZ);
                      END;
                  END;
         READLN(ABDATA,STAT[I+MAXNOCONT].DISTANCE);
         READLN(ABDATA,NUMRX);
         READLN(ABDATA,NUMTX);
         VAXDOG := BIGNUM;
         PKTDOG := SMALLPKT;
         END;


 (* READ IN CONTROLLER PARAMETERS *)
 FOR I:= 1 TO MAXNOCONT DO
         WITH CONT[I] DO
                 BEGIN
                 READLN(ABDATA,STAT[I].DISTANCE);
                 WITH CS DO
                    BEGIN
                       READLN(ABDATA,STARTIME);
                          IF CONNTO = 0 THEN
                            STARTIME := BIGNUM;
                       READLN(ABDATA,NUMPKTSRET);
                       READLN(ABDATA,PKTSZRET);
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
            END;
         FOR J := 1 TO 2 DO
            READLN(ABDATA,RSP[J]);
         READLN(ABDATA,AD);
         READLN(ABDATA,RESPSZ);
         BITS := RESPSZ * 8;
         READLN(ABDATA,NO8614);
         CONTDOG := BIGNUM;
         PKTDOG := SMALLPKT;
         FOR J:= 1 TO NO8614 DO
                WITH ARC[J] DO
                BEGIN
                READLN(ABDATA,FILLTM);
                READLN(ABDATA,ARCSZ);
                END;
         READLN(ABDATA,MSGSEC);
         TOTMSGSEC := TOTMSGSEC + MSGSEC;
         READLN(ABDATA,DLYTM);
         READLN(ABDATA,PKTCOMMSZ);
         READLN(ABDATA,TALKTO);
         END;

END;

(*****************************************************************************)

PROCEDURE DISPLAY;

(*****************************************************************************)

(* DISPLAY SIMULATION CONFIGURATION DATA TO OPERATOR TERMINAL *)

VAR
        I : INTEGER;
        IN1, IN2, IN3: INTEGER;
        IN4: CHAR;

BEGIN

REPEAT

WRITELN(' ');
WRITELN(' ');
WRITELN('                DATA DISPLAY MENU ');
WRITELN(' ');
WRITELN(' ');
WRITELN('1   -   GENERAL CONFIGURATION INFORMATION (GLOBAL DATA)');
WRITELN('2   -   CONTROLLERS ATTACHED TO VAX');
WRITELN('3   -   VAX SIMULATION INFORMATION ');
WRITELN('4   -   VAX COMMANDS TO CONTROLLER, CONTROLLER RESPONSE');
WRITELN('5   -   CONTROLLER STARTUP');
WRITELN('6   -   CONTROLLER ARCHIVE INFORMATION ');
WRITELN('7   -   TVC/AFT COMMUNICATION');
WRITELN('8   -   RETURN TO PREVIOUS MENU');
WRITELN(' ');
```

```
REPEAT
   WRITELN('ENTER SELECTION: ');
   READLN(IN1);
UNTIL ((IN1>=1) AND (IN1<=9));

IF IN1 IN [2..4] THEN
      BEGIN
         IN2 := 1;
         IF NOVAX > 1 THEN
            BEGIN
                 REPEAT
                    WRITELN(' ');
                    WRITELN('ENTER VAX NUMBER: (1 TO ',NOVAX:1,') ');
                    READLN(IN2);
                 UNTIL ((IN2>=1) AND (IN2<=NOVAX));
            END;
      END;

WRITELN(' ');

IF IN1 IN [4..7] THEN
      BEGIN
         IN3 := 1;
           REPEAT

                REPEAT
                     WRITELN(' ');
                     WRITELN('ENTER CONT NUMBER: (1 TO ',MAXNOCONT:2,') ');
                     READLN(IN3);
                UNTIL ((IN3>=1) AND (IN3<=MAXNOCONT));

                IF IN1 = 4 THEN
                   BEGIN
                     IF VAX[IN2].CONNECT[IN3] <> 1 THEN
                        BEGIN
                           WRITELN('CONTROLLER NOT CONNECTED TO VAX');
                           IN3 := 0;
                        END;
                   END
                 ELSE
                   IF CONT[IN3].CONNTO = 0 THEN
                      BEGIN
                         WRITELN(' ');
                         WRITELN('CONTROLLER NOT ATTACHED');
                         IN3 := 0;
                      END;
           UNTIL ((IN3>=1) AND (IN3<=MAXNOCONT));
      END;

WRITELN(' ');
WRITELN(' ');
CASE IN1 OF
    1: BEGIN
        WRITELN('CONFIGURATION DESCRIPTION:    ',DESCRIPT[FN]);
        WRITELN(' ');
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
    WRITELN('RANDOM NUMBER:                    ',RANNUM);
    WRITELN('SIMULATION RUN TIME:              ',SIMTIME);
    WRITELN('ETHERNET BUS I/O RATE:            ',EBUSRATE);
    IF HOTFIRE = 1 THEN
    WRITELN('TVC/AFT COMMUNICATION:             SELECTED')
    ELSE
    WRITELN('TVC/AFT COMMUNICATION:             NOT SELECTED');
    WRITELN(' ');
    WRITELN('NUMBER OF VAX COMPUTERS:          ',NOVAX);
    WRITELN('NUMBER OF CONTROLLERS:            ',NOCONT);
   END;
2: BEGIN
    WRITELN('DISTANCE OF VAX ',IN2:1,' FROM REFERENCE POINT: ',
            STAT[IN2+MAXNOCONT].DISTANCE);
    WRITELN(' ');
    WRITELN('CONTROLLERS WHICH COMMUNICATE WITH VAX ',IN2:1);
    WRITELN(' ');
    WITH VAX[IN2] DO
      BEGIN
         FOR I := 1 TO MAXNOCONT DO
            IF CONNECT[I] = 1 THEN
               WRITELN('INDEX: ',I:2,'  NAME: ',NAM[I]);
      END;
   END;
3: BEGIN
    WRITELN('SIMULATION INFORMATION: VAX ',IN2:1);
    WRITELN(' ');
    WRITELN('RECEIVE  - MAX NUMBER OF BYTES IN A SECOND: ',
            VAX[IN2].NUMRX);
    WRITELN(' ');
    WRITELN('TRANSMIT - MAX NUMBER OF BYTES IN A SECOND: ',
            VAX[IN2].NUMTX);
   END;
4: BEGIN
    WITH VAX[IN2] DO
      BEGIN
        WRITELN('COMMAND TRANSMISSION FROM VAX: ',IN2:1,' TO CONT: ',IN3:2);
        WRITELN(' ');
        WRITELN('NEXT TRANSMIT TIME GENERATED IN RANGE: ',VC[IN3].R1,
           ' TO: ',VC[IN3].R2);
        WRITELN('NUMBER OF SINGLE COMMANDS:      ',VC[IN3].SP);
        WRITELN('NUMBER OF BLOCKED COMMANDS:     ',VC[IN3].BP);
        WRITELN('NUMBER OF COMMANDS PER BLOCK:   ',VC[IN3].BC);
        WRITELN('NUMBER OF PACKETS PER COMMAND:  ',VC[IN3].COMSZ);
        WRITELN(' ');
      END;
    WITH CONT[IN3] DO
      BEGIN
        WRITELN('CONT RESPONSE GENERATION:');
        WRITELN(' ');
        WRITELN('PERCENTAGE OF COMMANDS WHICH REQUIRE RESPONSE: ',
                RSP[1]:2,' DO,  ',RSP[2]:2,' DO NOT' );
        WRITELN('AVERAGE DELAY TIME TO RESPOND: ',AD);
        WRITELN('RESPONSE PACKET SIZE:          ',RESPSZ);
      END;
   END;
```

```
5: BEGIN
    WITH CONT[IN3] DO
        BEGIN
          WRITELN('CONTROLLER: ',IN3:2,'              START TIME: ',
                    CS.STARTIME);
          WRITELN('DISTANCE FROM REFERENCE POINT:          ',
                    STAT[IN3].DISTANCE);
          WRITELN('CONNECTED TO VAX:                       ',CONNTO);
          WRITELN(' ');
          WRITELN('NUMBER OF PACKETS VAX SENDS TO CONT: ',
                VAX[CONNTO].VS[IN3].NUMPKTSENT);
          WRITELN('NUMBER OF BYTES PER PACKET:             ',
                VAX[CONNTO].VS[IN3].PKTSENTSZ);
          WRITELN(' ');
          WRITELN('CONT RESPONDS WITH SPECIAL ACK TO PACKETS ABOVE');
          WRITELN(' ');
          WRITELN('NUMBER OF PACKETS CONT SENDS TO VAX: ',CS.NUMPKTSRET);
          WRITELN('NUMBER OF BYTES PER PACKET:             ',CS.PKTSZRET);
          WRITELN(' ');
          WRITELN('FINALLY, VAX SENDS CONT 20 PACKETS OF 1500 BYTES ',
                  'THEN 80 PACKETS OF 80 BYTES');
        END;
    END;
 6: BEGIN
     WITH CONT[IN3] DO
       BEGIN
         WRITELN('NUMBER OF ISBC 8614 BOARDS ON CONTROLLER: ',NO8614);
         WRITELN(' ');
         WRITELN('BOARD NO.          FILL TIME       BUFFER SIZE');
         FOR I:= 1 TO NO8614 DO
             WITH ARC[I] DO
                WRITELN('     ',I:1,'          ',FILLTM,'     ',ARCSZ);
       END;
    END;
 7: BEGIN
      IF HOTFIRE = 1 THEN
        BEGIN
          IF IN3 IN [5..7]  THEN
            BEGIN
              IF ((IN3 = TVC) OR (CONT[IN3].TALKTO = TVC)) THEN
                 BEGIN
                   WRITELN('TVC - AFT COMMUNICATION');
                   WRITELN(' ');
                   WRITELN('TVC SENDS: ',CONT[TVC].MSGSEC:2,' MSG/SEC TO ',
                           NAM[CONT[TVC].TALKTO]);
                   WRITELN('DELAY TIME BEFORE TVC RESPONDS:  ',
                           CONT[TVC].DLYTM);
                   WRITELN('PACKET SIZE ON RESPONSES:         ',
                           CONT[TVC].PKTCOMMSZ);
                   WRITELN(' ');
                   WRITELN(NAM[CONT[TVC].TALKTO],' SENDS: ',
                    CONT[CONT[TVC].TALKTO].MSGSEC:2,' MSG/SEC TO TVC');
                   WRITELN('DELAY TIME BEFORE ',NAM[CONT[TVC].TALKTO],
                           ' RESPONDS: ',CONT[CONT[TVC].TALKTO].DLYTM);
                   WRITELN('PACKET SIZE ON RESPONSES:         ',
                           CONT[CONT[TVC].TALKTO].PKTCOMMSZ);
```

```
                  WRITELN(' ');
               END
            ELSE
               WRITELN('AFT SPECIFIED NOT CONNECTED TO TVC');
         END
       ELSE
          WRITELN('TVC OR AFT MUST BE SELECTED');
    END;
  END;
 8:
END;

IF (IN1 <> 8) THEN
   BEGIN
      WRITELN(' ');
      WRITELN('ENTER <RETURN> TO CONTINUE: ');
      WHILE NOT EOLN DO
        READLN(IN4);
   END;

UNTIL ( IN1 = 8);

END;



(**************************************************************************************)

PROCEDURE REALGETOP(VIEW: STRTYP; VAR VALUE: REAL);

(**************************************************************************************)

(* QUERY OPERATOR FOR A REAL NUMBER TO REPLACE VARIABLE DISPLAYED *)

VAR
        INP1: REAL;

BEGIN

        INP1 := BIGNUM;
        WRITELN('PRESS <RETURN> TO RETAIN CURRENT VALUE OR ');
        WRITELN('ENTER ',VIEW,' [CURRENT: ',VALUE,']');

        WHILE NOT EOLN DO
            READ (INP1);
        READLN;

        IF INP1 <> BIGNUM THEN
            VALUE := INP1;

        WRITELN(' ');

END;
```

```
(*****************************************************************************)

PROCEDURE INTGETOP (VIEW: STRTYP; VAR VALUE: INTEGER);

(*****************************************************************************)

(* QUERY OPERATOR FOR A INTEGER NUMBER TO REPLACE VARIABLE DISPLAYED.*)

VAR
        INP1: INTEGER;

BEGIN

        INP1 := 99999;
        WRITELN('PRESS <RETURN> TO RETAIN CURRENT VALUE OR ');
        WRITELN('ENTER ',VIEW,' [CURRENT: ',VALUE,']');

        WHILE NOT EOLN DO
            READ (INP1);
        READLN;

        IF INP1 <> 99999 THEN
            VALUE := INP1;

        WRITELN(' ');

END;

(*****************************************************************************)

PROCEDURE MODIFY;

(*****************************************************************************)

(* ALLOWS OPERATOR TO MODIFY SIMULATION PARAMETERS *)

VAR
        I,J,K  : INTEGER;
        IN1, IN2, IN3, IN5, IN6: INTEGER;
        IN4: CHAR;
        TEMP : REAL;
        OPDISP: STRTYP;
        INDES: ARRAY [1..5] OF ARA1;

BEGIN

REPEAT

WRITELN(' ');
WRITELN(' ');
WRITELN('                  MODIFY PARAMETERS MENU ');
WRITELN(' ');
WRITELN(' ');
WRITELN('1    -   GENERAL CONFIGURATION INFORMATION (GLOBAL DATA)');
```

```
WRITELN('2    -    VAX SIMULATION INFORMATION ');
WRITELN('3    -    VAX COMMANDS TO CONTROLLER, CONTROLLER RESPONSE');
WRITELN('4    -    CONTROLLER STARTUP');
WRITELN('5    -    CONTROLLER ARCHIVE INFORMATION ');
WRITELN('6    -    TVC/AFT COMMUNICATION');
WRITELN('7    -    RETURN TO PREVIOUS MENU');
WRITELN(' ');

REPEAT
   WRITELN('ENTER SELECTION: ');
   READLN(IN1);
UNTIL ((IN1>=1) AND (IN1<=7));

IF IN1 IN [2..3] THEN
      BEGIN
         IN2 := 1;
         REPEAT
              WRITELN(' ');
              WRITELN('ENTER VAX NUMBER: (1 TO ',MAXNOVAX:1,') ');
              READLN(IN2);
         UNTIL ((IN2>=1) AND (IN2<=MAXNOVAX));
      END;

WRITELN(' ');

IN5 := 1;
IF IN1 IN [3..6] THEN
      BEGIN
         IN3 := 1;
         REPEAT
            WRITELN(' ');
            WRITELN('CONTROLLER SELECTION ');
            IF IN1 = 3 THEN
            WRITELN('VAX SELECTED WILL BE SET TO COMMUNICATE WITH THE ',
                  'CONTROLLER SELECTED HERE');
            WRITELN(' ');
            WRITELN('NUMBER    NAME    CONNECTED TO');
            FOR I:=1 TO MAXNOCONT DO
               BEGIN
                 WRITE('    ',I:2,'      ',NAM[I]);
                 IF CONT[I].CONNTO = 0 THEN
                    BEGIN
                          J := MAXNOVAX+1;
                          NAM[J+MAXNOCONT] := 'NONE';
                    END
                 ELSE
                    J := CONT[I].CONNTO;
                 WRITELN('     ',NAM[J+MAXNOCONT]);
              END;
            WRITELN(' ');
            WRITELN('ENTER NUMBER: ');
            READLN(IN3);
         UNTIL ((IN3>=1) AND (IN3<=MAXNOCONT));

         IF IN1 = 3 THEN
             BEGIN
```

```
          WRITELN(' ');
          WRITELN('CONT: ',IN3:2,' COMMUNICATES WITH VAX: ',IN2:2);
          REPEAT
              WRITELN('ENTER <1> CONNECT OR <2> DISCONNECT:' );
              READLN(IN5);
          UNTIL ((IN5=1) OR (IN5=2));
          IF IN5 = 1 THEN
             BEGIN
                CONT[IN3].CONNTO := IN2;
                FOR I := 1 TO MAXNOVAX DO
                  VAX[I].CONNECT[IN3] := 0;
                VAX[IN2].CONNECT[IN3] := 1;
             END
          ELSE
             BEGIN
                CONT[IN3].CONNTO := 0;
                VAX[IN2].CONNECT[IN3] := 0;
                CONT[IN3].CS.STARTIME := BIGNUM;
             END;
     END;

IF IN5 = 1 THEN
   BEGIN
   IF CONT[IN3].CONNTO = 0 THEN
      BEGIN
        REPEAT
           WRITELN(' ' );
           WRITELN('ENTER VAX NUMBER WHICH COMMUNICATES',
                   ' WITH CONTROLLER: ');
           READLN(IN2);
        UNTIL ((IN2>=1) OR (IN2<=MAXNOVAX));
        CONT[IN3].CONNTO := IN2;
        VAX[IN2].CONNECT[IN3] := 1;
      END;

   IF ((IN3 IN [5..7]) AND (IN1 = 6) AND (HOTFIRE = 1)) THEN
           IF IN3 = TVC THEN
                BEGIN
                   REPEAT
                     WRITELN('TVC COMMUNICATES WITH <5> AFT1 OR <6>',
                             ' AFT2: ');
                     READLN(CONT[TVC].TALKTO);
                   UNTIL (CONT[TVC].TALKTO IN [5..6]);
                   CONT[CONT[TVC].TALKTO].TALKTO := TVC;
                END
            ELSE
                BEGIN
                   REPEAT
                      WRITELN('TVC COMMUNICATES WITH ',NAM[IN3]);
                      WRITELN('ENTER <1> YES OR <2> NO:');
                      READLN(IN6);
                   UNTIL (IN6 IN [1..2]);
                   IF IN6 = 1 THEN
                     BEGIN
                        CONT[IN3].TALKTO := TVC;
                        CONT[TVC].TALKTO := IN3;
```

```
                                    END
                                   ELSE
                                       IN5 := 2;
                               END;
                        END;
        END;

WRITELN(' ');
WRITELN(' ');
IF IN5 = 1 THEN
CASE IN1 OF
    1: BEGIN
        INDES[FN] := 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA';
        WRITELN('PRESS RETURN TO RETAIN CURRENT VALUE OR ');
        WRITELN('ENTER NEW DESCRIPTION, CURRENT: ',DESCRIPT[FN]);
        WHILE NOT EOLN DO
           READ(INDES[FN]);
        READLN;
        IF INDES[FN] <> 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA' THEN
            DESCRIPT[FN] := INDES[FN];
        WRITELN(' ');

        OPDISP := 'RANDOM NUMBER                                    ';
        INTGETOP(OPDISP,RANNUM);
        IDUM := 0-RANNUM;

        OPDISP := 'SIMULATION RUN TIME                              ';
        REALGETOP(OPDISP,SIMTIME);

        OPDISP := 'ETHERNET BUS I/O RATE                            ';
        REALGETOP(OPDISP,EBUSRATE);

        REPEAT
        OPDISP := 'TVC/AFT COMMUNICATION: 0 - NO, 1 - YES  ';
        INTGETOP(OPDISP,HOTFIRE);
        UNTIL ((HOTFIRE=0) OR (HOTFIRE=1));

        OPDISP := 'NUMBER OF VAX COMPUTERS                          ';
        INTGETOP(OPDISP,NOVAX);

        OPDISP := 'NUMBER OF CONTROLLERS                            ';
        INTGETOP(OPDISP,NOCONT);
       END;
    2: BEGIN
        WRITELN('SIMULATION INFORMATION: VAX ',IN2:1);
        WRITELN(' ');
        OPDISP := 'DISTANCE OF VAX FROM REFERENCE POINT     ';
        REALGETOP(OPDISP,STAT[IN2+MAXNOCONT].DISTANCE);

        OPDISP := 'MAX NUM BYTES VAX CAN RECEIVE PER SEC    ';
        INTGETOP(OPDISP,VAX[IN2].NUMRX);

        OPDISP := 'MAX NUM BYTES VAX CAN TRANSMIT PER SEC   ';
        INTGETOP(OPDISP,VAX[IN2].NUMTX);
       END;
    3: BEGIN
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
   WITH VAX[IN2] DO
     BEGIN
       WRITELN('COMMAND TRANSMISSION FROM VAX: ',IN2:1,' TO CONT: ',IN3:2);
       WRITELN(' ');
       OPDISP := 'VAX TX TIME GENERATED USING RANGE 1      ';
       REALGETOP(OPDISP,VC[IN3].R1);

       OPDISP := 'VAX TX TIME GENERATED USING RANGE 2      ';
       REALGETOP(OPDISP,VC[IN3].R2);
       IF VC[IN3].R1 > VC[IN3].R2 THEN
        BEGIN
           TEMP := VC[IN3].R1;
           VC[IN3].R1 := VC[IN3].R2;
           VC[IN3].R2 := TEMP;
         END;

       OPDISP := 'NUMBER OF SINGLE COMMANDS                 ';
       INTGETOP(OPDISP,VC[IN3].SP);

       OPDISP := 'NUMBER OF BLOCKED COMMANDS                ';
       INTGETOP(OPDISP,VC[IN3].BP);

       OPDISP := 'NUMBER OF COMMANDS PER BLOCK              ';
       INTGETOP(OPDISP,VC[IN3].BC);

       OPDISP := 'NUMBER OF BYTES PER PACKET (COMMAND)      ';
       INTGETOP(OPDISP,VC[IN3].COMSZ);
       WRITELN(' ');
     END;
    WITH CONT[IN3] DO
      BEGIN
        WRITELN('CONT RESPONSE GENERATION:');
        WRITELN(' ');
        OPDISP := 'NUM CMDS WHICH DO REQUIRE RESPONSE       ';
        INTGETOP(OPDISP,RSP[1]);

        OPDISP := 'NUM CMDS WHICH DO NOT REQUIRE RESPONSE   ';
        INTGETOP(OPDISP,RSP[2]);

        OPDISP := 'AVERAGE DELAY TIME TO RESPOND            ';
        REALGETOP(OPDISP,AD);

        OPDISP := 'RESPONSE PACKET SIZE                     ';
        INTGETOP(OPDISP,RESPSZ);
        BITS := RESPSZ * 8;

      END;
   END;
4: BEGIN
    WITH CONT[IN3] DO
        BEGIN
          WRITELN(' ');
          WRITELN('CONTROLLER: ',IN3:2,' COMMUNICATES WITH VAX: ',
                    NAM[CONNTO+MAXNOCONT]);
          WRITELN(' ');
```

```
          OPDISP := 'START TIME OF CONTROLLER                         ';
          REALGETOP(OPDISP,CS.STARTIME);

          OPDISP := 'DISTANCE FROM REFERENCE POINT                    ';
          REALGETOP(OPDISP,STAT[IN3].DISTANCE);

          OPDISP := 'NUMBER OF PACKETS VAX SENDS TO CONT              ';
          INTGETOP(OPDISP,VAX[CONNTO].VS[IN3].NUMPKTSENT);
          CONT[IN3].CS.ACKSRET := VAX[CONNTO].VS[IN3].NUMPKTSENT;

          OPDISP := 'NUMBER OF BYTES PER PACKET                       ';
          INTGETOP(OPDISP,VAX[CONNTO].VS[IN3].PKTSENTSZ);
          CONT[IN3].CS.ACKPKTSZ := VAX[CONNTO].VS[IN3].PKTSENTSZ;

          WRITELN(' ');
          WRITELN('CONT RESPONDS WITH SPECIAL ACK TO PACKETS ABOVE');
          WRITELN(' ');
          OPDISP := 'NUMBER OF PACKETS CONT SENDS TO VAX              ';
          INTGETOP(OPDISP,CS.NUMPKTSRET);

          OPDISP := 'NUMBER OF BYTES PER PACKET                       ';
          INTGETOP(OPDISP,CS.PKTSZRET);
          WRITELN(' ');
          WRITELN('FINALLY, VAX SENDS CONT 20 PACKETS OF 1500 BYTES ',
                  'THEN 80 PACKETS OF 80 BYTES');
        END;
    END;
 5: BEGIN
     WITH CONT[IN3] DO
       BEGIN
         OPDISP := 'NUMBER OF ISBC 8614 BOARDS ON CONTROLLER';
         INTGETOP(OPDISP,NO8614);

         FOR I:= 1 TO NO8614 DO
           WITH ARC[I] DO
            BEGIN
               WRITELN(' ');
               WRITELN('BOARD NUMBER: ',I:2);
               WRITELN(' ');
               OPDISP := 'BUFFER FILL TIME                               ';
               REALGETOP(OPDISP,FILLTM);
               OPDISP := 'BUFFER BYTE SIZE                               ';
               INTGETOP(OPDISP,ARCSZ);
             END;
       END;
    END;
 6: BEGIN
     IF HOTFIRE = 1 THEN
       BEGIN
          IF IN3 IN [5..7] THEN
             BEGIN
              IF ((IN3 = TVC) OR (CONT[IN3].TALKTO = TVC)) THEN
                 BEGIN
                    WRITELN('TVC - ',NAM[CONT[TVC].TALKTO],' COMMUNICATION');
                    WRITELN(' ');
```

```
                    OPDISP := 'NUM MSG/SEC TVC SENDS TO AFT              ';
                    INTGETOP(OPDISP,CONT[TVC].MSGSEC);

                    OPDISP := 'DELAY TIME BEFORE TVC RESPONDS            ';
                    REALGETOP(OPDISP,CONT[TVC].DLYTM);

                    OPDISP := 'PACKET SIZE ON RESPONSES                  ';
                    INTGETOP(OPDISP,CONT[TVC].PKTCOMMSZ);

                    WRITELN(' ');

                    OPDISP := 'NUM MSG/SEC AFT SENDS TO TVC              ';
                    INTGETOP(OPDISP,CONT[CONT[TVC].TALKTO].MSGSEC);

                    OPDISP := 'DELAY TIME BEFORE AFT RESPONDS            ';
                    REALGETOP(OPDISP,CONT[CONT[TVC].TALKTO].DLYTM);

                    OPDISP := 'PACKET SIZE ON RESPONSES                  ';
                    INTGETOP(OPDISP,CONT[CONT[TVC].TALKTO].PKTCOMMSZ);

                    TOTMSGSEC := CONT[TVC].MSGSEC +
                             CONT[CONT[TVC].TALKTO].MSGSEC;

                    WRITELN(' ');
                 END
              ELSE
                 WRITELN('AFT SPECIFIED NOT CONNECTED TO TVC');
            END
          ELSE
             WRITELN('TVC OR AFT MUST BE SELECTED');
       END;
     END;
   7:
END;

IF (IN1 <> 7) THEN
   BEGIN
      WRITELN(' ');
      WRITELN('ENTER <RETURN> TO CONTINUE: ');
      WHILE NOT EOLN DO
        READLN(IN4);
   END;

UNTIL ( IN1 = 7);

END;


(****************************************************************************)

PROCEDURE STORE;

(****************************************************************************)
```

```
(* WRITE SIMULATION RUN PARAMETERS TO DATA FILE SELECTED BY OPERATOR *)
VAR
        I, J, K: INTEGER;

BEGIN
        (* WRITE GENERAL INFORMATION TO DATA FILE SELECTED BY USER *)
        REWRITE(ABDATA);
        WRITELN(ABDATA,DESCRIPT[FN]);
        WRITELN(ABDATA,RANNUM);
        WRITELN(ABDATA,SIMTIME);
        WRITELN(ABDATA,EBUSRATE);
        WRITELN(ABDATA,HOTFIRE);
        WRITELN(ABDATA,NOVAX);
        WRITELN(ABDATA,NOCONT);

        (* WRITE IN VAX PARAMETERS *)
        FOR I:= 1 TO MAXNOVAX DO
            WITH VAX[I] DO
                BEGIN
                FOR J:= 1 TO MAXNOCONT DO
                        WRITELN(ABDATA, CONNECT[J]);
                FOR J:=1 TO MAXNOCONT DO
                          BEGIN
                            WRITELN(ABDATA,VS[J].NUMPKTSENT);
                            WRITELN(ABDATA,VS[J].PKTSENTSZ);
                            WITH VC[J] DO
                              BEGIN
                                WRITELN(ABDATA,R1);
                                WRITELN(ABDATA,R2);
                                WRITELN(ABDATA,SP);
                                WRITELN(ABDATA,BP);
                                WRITELN(ABDATA,BC);
                                WRITELN(ABDATA,COMSZ);
                              END;
                          END;
                WRITELN(ABDATA,STAT[I+MAXNOCONT].DISTANCE);
                WRITELN(ABDATA,NUMRX);
                WRITELN(ABDATA,NUMTX);
                END;


        (* WRITE OUT CONTROLLER PARAMETERS *)
        FOR I:= 1 TO MAXNOCONT DO
                WITH CONT[I] DO
                        BEGIN
                        WRITELN(ABDATA,STAT[I].DISTANCE);
                        WITH CS DO
                           BEGIN
                              WRITELN(ABDATA,STARTIME);
                              WRITELN(ABDATA,NUMPKTSRET);
                              WRITELN(ABDATA,PKTSZRET);
                           END;
                        FOR J := 1 TO 2 DO
                           WRITELN(ABDATA,RSP[J]);
                        WRITELN(ABDATA,AD);
```

```
                        WRITELN(ABDATA,RESFSZ);
                        WRITELN(ABDATA,NO8614);
                        FOR J:= 1 TO NO8614 DO
                                WITH ARC[J] DO
                                BEGIN
                                   WRITELN(ABDATA,FILLTM);
                                   WRITELN(ABDATA,ARCSZ);
                                END;
                        WRITELN(ABDATA,MSGSEC);
                        WRITELN(ABDATA,DLYTM);
                        WRITELN(ABDATA,PKTCOMMSZ);
                        WRITELN(ABDATA,TALKTO);
                        END;

END;

(***************************************************************************)

PROCEDURE CONFIGURE;

(***************************************************************************)

(* QUERY USER FOR THE DATA FILE OF HIS CHOICE BY DISPLAYING THE DECRIPTION *)
(* FROM WITHIN THE FILE IN A MENU FORMAT *)

VAR
        I, J, K: INTEGER;

BEGIN

FOR I := 1 TO 5 DO
   BEGIN
        BIND(ABDATA,DAT[I],ISTAT);
        RESET(ABDATA);
        IF NOT EOF(ABDATA) THEN
            READLN(ABDATA,DESCRIPT[I]);
        CLOSE(ABDATA);
   END;

WRITELN(' ');
WRITELN(' ');
WRITELN('THE 5 DATA FILES BELOW ARE AVAILABLE FOR OPERATOR USE.');
WRITELN('EACH FILE CONTAINS CONFIGURATION INFORMATION FROM PREVIOUS RUNS.');
WRITELN('THE PARAMETERS MAY BE DISPLAYED AND MODIFIED BEFORE SIMULATION.');
WRITELN('SELECT THE NUMBER BELOW DESCRIBING THE FILE DESIRED.');
WRITELN(' ');
WRITELN(' ');

FOR I:= 1 TO 5 DO
        WRITELN('       ',I:1,'  -  ',DESCRIPT[I]);

WRITELN(' ');
WRITELN(' ');
FN := 0;
REPEAT
        WRITELN('ENTER SELECTION: ');
```

```
        READLN(FN);
UNTIL ((FN>=1) AND (FN<=5));

BIND(ABDATA,DAT[FN],ISTAT);
RESET(ABDATA);

IF NOT EOF(ABDATA) THEN
    GETDATA;

REPEAT

WRITELN(' ');
WRITELN(' ');
WRITELN(' ');
WRITELN(' ');
WRITELN('               ABACS ETHERNET SIMULATION MENU');
WRITELN(' ');
WRITELN(' ');
WRITELN('1  -  DISPLAY CONFIGURATION INFORMATION ON TERMINAL ');
WRITELN('2  -  MODIFY CONFIGURATION INFOFMATION              ');
WRITELN('3  -  STORE MODIFIED INFORMATION TO FILE SELECTED   ');
WRITELN('4  -  RUN SIMULATION                                ');
WRITELN('5  -  EXIT PROGRAM                                  ');
WRITELN(' ');
SL:=0;
REPEAT
        WRITELN('ENTER SELECTION: ');
        READLN(SL);
UNTIL ((SL>=1) AND (SL<=5));

IF SL = 1 THEN DISPLAY;
IF SL = 2 THEN MODIFY;
IF SL = 3 THEN
   BEGIN
        BIND(ABDATA, DAT[FN], ISTAT);
        STORE;
   END;

UNTIL ((SL=4) OR (SL=5));

IF SL = 4 THEN
   BEGIN
        (*CALCULATE PROPAGATION DELAY BETWEEN DEVICES *)
        FOR I:= 1 TO MAXDEVTOT DO
           FOR J:= 1 TO MAXDEVTOT DO
        BEGIN
                PDELAY[I,J] := ABS(STAT[I].DISTANCE-STAT[J].DISTANCE) * 1.27E-9;
        END;
   END;
END;


(****************************************************************************)

PROCEDURE PRTDATA;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
(**************************************************************************)

(* WRITES OUT USER SPECIFIED DATA TO OUTPUT FILE FOR PRINTOUT *)

VAR
        I, J, PRT: INTEGER;

BEGIN
        REWRITE(ABOUT);
        WRITELN(ABOUT,'SIMULATION DESCRIPTION: ',DESCRIPT[FN]);
        WRITELN(ABOUT,'SIMULATION RUN TIME (S): ',SIMTIME:3,' ',
        'ETHERNET BUS RATE (B/S): ',EBUSRATE:3,' ',
        'RANDOM NUMBER SEED: ',RANNUM:5);
        WRITELN(ABOUT,' ');

        WRITELN(ABOUT,'CONT  START TIME(S) VAX TX: # PKTS, PKT SZ(B) CONT RES',
        'P: # PKTS, PKT SZ(B)   AFT/TVC COMM: MSG/SEC   DELAY(S)  PKT SZ(B)');
        WRITELN(ABOUT,'-------------------------------------------------------------',
        '-------------------------------------------------------------');
        FOR I := 1 TO MAXNOCONT DO
          FOR J := 1 TO MAXNOVAX DO
              IF VAX[J].CONNECT[I] = 1 THEN
                BEGIN
                  WITH VAX[J].VS[I] DO
                  WRITELN(ABOUT,NAM[I]:4,' ',CONT[I].CS.STARTIME:10:2,'       ',
                  J:2,'        ',NUMPKTSENT:2,'    ',PKTSENTSZ:5,'
                  '       ',CONT[I].CS.NUMPKTSRET:2,'  ',CONT[I].CS.PKTSZRET:5,
                  '                 ',CONT[I].MSGSEC:4,' ',
                  CONT[I].DLYTM:12:3,'       ',CONT[I].PKTCOMMSZ:4);
                END;
         WRITELN(ABOUT,'-------------------------------------------------------------',
         '-------------------------------------------------------------');
         WRITELN(ABOUT,' ');

         FOR I:= 1 TO MAXNOVAX DO
             WITH VAX[I] DO
                 BEGIN
                 PRT := 0;
                 FOR J := 1 TO MAXNOCONT DO
                   IF CONNECT[J] = 1 THEN
                       PRT := 1;
                 IF PRT = 1 THEN
                       BEGIN
 WRITELN(ABOUT,'VAX NO: ',I:2,' DISTANCE (FT): ',STAT[I+MAXNOCONT].DISTANCE:8:1,
       ' NUMBER RECEIVE (B): ',NUMRX:8,' NUMBER TRANSMIT (B): ',NUMTX:8);
 WRITELN(ABOUT,'CONT    RNG 1(S),RNG 2(S) SINGLE CMD: %,BLOCKED CMD:',
             ' %  NO SZ(B)');
 WRITELN(ABOUT,'-------------------------------------------------------',
             '--------------');
                       END;

                 FOR J:= 1 TO MAXNOCONT DO
                     IF CONNECT[J] = 1 THEN
                         WITH VC[J] DO
                                BEGIN
 WRITELN(ABOUT,NAM[J]:4,'    ',R1:7:3,' ',R2:7:3,'            ',SP:3,'       ',
```

```
  '      ',BP:3,'   ',BC:3,' ',COMSZ:5);
                                  END;
                IF PRT = 1 THEN
                        BEGIN
 WRITELN(ABOUT,'-------------------------------------------------------------',
               '-------------');
 WRITELN(ABOUT,' ');
                        END;
                END;

WRITELN(ABOUT,'CONT  DIST   RESP   PKT   AVG  NUMBER FILL TIME(S), BUF SZ(B)');
WRITELN(ABOUT,'      (FT)   1: 2:  SZ(B) DLY(S) 8614  1:          2:',
        '          3:          4:          5:          6:          7:');
WRITELN(ABOUT,'------------------------------------------------------------',
  '-------------------------------------------------------------');

        FOR I:= 1 TO MAXNOCONT DO
                WITH CONT[I] DO
                        BEGIN
                        IF CONNTO <> 0 THEN
                            BEGIN
WRITE(ABOUT,NAM[I]:4,STAT[I].DISTANCE:8:1,RSP[1]:2,' ',RSP[2]:3,' ',
        RESPSZ:4,' ',AD:2,'  ',NO8614:2,' ');
                        FOR J:= 1 TO NO8614 DO
                                WITH ARC[J] DO
                                BEGIN
                                WRITE(ABOUT,FILLTM:7:3,ARCSZ:5);
                                END;
WRITELN(ABOUT,' ');
                            END;
                        END;
WRITELN(ABOUT,'------------------------------------------------------------',
  '-------------------------------------------------------------');
END;


(*****************************************************************************)

FUNCTION RAN(VAR IDUM: INTEGER): REAL;

(*****************************************************************************)

(* RETURNS SOME RANDOM NUMBER IN RANGE 0 TO 1*)

CONST
        M = 714025;
        IA = 1366;
        IC = 150889;
        RM = 1.400512E-6;

VAR
        J : INTEGER;

BEGIN
        IF (IDUM < 0) THEN
```

```
            BEGIN
                    IDUM := (IC-IDUM) MOD M;
                    FOR J:= 1 TO 97 DO
                            BEGIN
                            IDUM := (IA*(IDUM MOD 6030) + IC) MOD M;
                            GLIR[J] := IDUM;
                            END;
                    IDUM := ((IA*(IDUM MOD 6030)) + IC) MOD M;
                    GLIY := IDUM;
            END;
        J:= 1+(97*(GLIY MOD 86480)) DIV M;
        IF (J>97) OR (J<1) THEN
                WRITELN('SOMETHING IS WRONG WITH THE RANDOM NUMBER GENERATOR');
        GLIY := GLIR[J];
        RAN := GLIY * RM;
        IDUM := ((IA *(IDUM MOD 6030)) + IC) MOD M;
        GLIR[J] := IDUM;

END;

(*************************************************************************)

PROCEDURE GETVAXNXT(I: INTEGER);

(*************************************************************************)

(* FOR A SPECIFIED VAX - I - FIND THE NEXT OPERATION TO BE TRANSMITTED *)
(* I.E. SEND ACK?  SEND WATCH DOG TIMER?  TIME TO SEND MESSAGE TO CONT? ETC *)

VAR
        J : INTEGER;

BEGIN
     WITH VAX[I] DO
        BEGIN
        (* SET NEXT TX TIME AS ACK FOR STARTERS *)
        TX[I+MAXNOCONT] := ACK;
        BIT[I+MAXNOCONT] := SMALLPKT;
        PART[I+MAXNOCONT] := -1;

        IF TX[I+MAXNOCONT] = BIGNUM THEN  (* ACK NOT SET, CHECK OTHER OPS *)
           BEGIN
             IF VAXDOG < ACK THEN  (* WATCHDOG TIMER CHECK *)
               BEGIN
                TX[I+MAXNOCONT] := VAXDOG;
                BIT[I+MAXNOCONT] := PKTDOG*8;
                PART[I+MAXNOCONT] := 0;
               END;

             FOR J:=1 TO MAXNOCONT DO   (* TRANSMIT COMMAND TO CONT CHECK *)
                IF OKTX[J] = 1 THEN
                        WITH VC[J] DO
                        IF TXTIME < TX[I+MAXNOCONT] THEN
                                BEGIN
                                TX[I+MAXNOCONT] := TXTIME;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
                        BIT[I+MAXNOCONT] :=  BITS;
                        PART[I+MAXNOCONT] := J;
                        END;

            FOR J := 1 TO MAXNOCONT DO   (* CONTROLLER STARTUP CHECK *)
               IF ((CONT[J].INSTART = 1) AND (CONT[J].CONNTO = I)) THEN
                     WITH VS[J] DO
                       BEGIN
                        IF TXTIME < TX[I+MAXNOCONT] THEN
                             BEGIN
                             TX[I+MAXNOCONT] := TXTIME;
                             BIT[I+MAXNOCONT] :=  BITS;
                             PART[I+MAXNOCONT] := STARTNUM+J;
                             END;
                        END;
            END;
        END;
END;

(*****************************************************************************)

PROCEDURE GETCONTNXT(I: INTEGER);

(*****************************************************************************)

(* FOR A SPECIFIED CONT - I - FIND THE NEXT OPERATION TO BE TRANSMITTED *)
(* I.E. SEND ACK?  SEND WATCH DOG TIMER?  TIME TO SEND ACRHIVE MSG TO VAX? ETC*)

VAR
        J : INTEGER;

BEGIN
     WITH CONT[I] DO
        BEGIN (* SET NEXT TX TIME AS ACK  OR STARTERS *)
        TX[I] := ACK;
        BIT[I] := SMALLPKT;
        PART[I] := -1;

        IF TX[I] = BIGNUM THEN   (* ACK NOT SET, CHECK OTHER OPS *)
          IF INSTART = 1 THEN    (* CONTROLLER STARTUP - COMING ON-LINE *)
             BEGIN
                  WITH CS DO
                      IF TXTIME < TX[I] THEN
                           BEGIN
                                TX[I] := TXTIME;
                                BIT[I] :=  BITS;
                                PART[I] := STARTNUM+I;
                           END;
             END

           ELSE
          BEGIN   (* HOTFIRE MESSAGE READY TO SEND?*)
            IF COMMTM < TX[I] THEN
              BEGIN
                 TX[I] := COMMTM;
```

```
                BIT[I] := PKTCOMMSZ * 8;
                PART[I] := -2;
              END;

           IF CONTDOG < TX[I] THEN  (* SEND WATCHDOG TIMER MSG BACK TO VAX*)
             BEGIN
                TX[I] := CONTDOG;
                BIT[I] := PKTDOG*8;
                PART[I] := 0;
             END;

           FOR J:=1 TO NO8614 DO  (* ARCHIVE MESSAGE READY FOR TRANSMISSION?*)
             WITH ARC[J] DO
                IF TXTM < TX[I] THEN
                     BEGIN
                       TX[I] := TXTM;
                       BIT[I] :=  ARCSZ*8;
                       PART[I] := J;
                     END;

           IF NUMRET > 0 THEN   (* SEND RESPONSE MSG TO VAX? *)
               IF RESPTM < TX[I] THEN
                 BEGIN
                       TX[I] := RESPTM;
                       BIT[I] := BITS;
                       PART[I] := NO8614 + 1;
                   END;
        END;
    END;
END;

(****************************************************************************)

PROCEDURE SETVAXTIME(J, I: INTEGER);

(****************************************************************************)

(* SET VAX -J- NEXT COMMAND TRANSMIT TIME AND PKT SIZE FOR CONTROLLER - I*)

VAR
      VAL: REAL;
BEGIN
     WITH VAX[J] DO
      WITH VC[I] DO
         BEGIN
           (* GENERATE NEXT TRANSMIT TIME BY ADDING A RANDOM VALUE WHICH *)
           (* IS TAKEN FROM A USER SPECIFIED RANGE *)
           VAL := RAN(IDUM);
           IF ((R1>1) AND (R2<10)) THEN
                VAL := VAL * 10;
            REPEAT
                IF VAL > R2 THEN
                     VAL := VAL - R2;
                IF VAL < R1 THEN
                     VAL := VAL + R1;
            UNTIL ((VAL>=R1) AND (VAL<=R2));
```

```
          TXTIME := TXTIME + VAL;

          (* IF SINGLE COMMAND TO TX NEXT *)
          IF ((C[1]<SP) AND (NEXT=1)) THEN
              BEGIN
                      BITS := COMSZ*8;
                      C[1] := C[1] + 1;
                      IF C[2] < BP THEN
                              NEXT := 2
              END
           ELSE
              (* BLOCKED CMD TO TX NEXT *)
              IF ((C[2]<BP) AND (NEXT=2 ) THEN
                      BEGIN
                              IF COMSZ * BC > 1500 THEN
                                  BEGIN
                                        BITS := 1500 * 8;
                                        NUMBCTX := TRUNC((COMSZ*BC)/1500)+1;
                                  END
                               ELSE
                                  BEGIN
                                        NUMBCTX := 1;
                                        BITS := BC * COMSZ * 8;
                                  END;
                               C[2] := C[2] + 1;
                               IF C[1] < SP THEN
                                        NEXT := 1;
                      END
              ELSE
                 BEGIN  (* RESET TO START SEQUENCE OVER & SEND SINGLE CMD*)
                      C[1] := 1;
                      C[2] := 0;
                      IF C[2] < BP THEN
                              NEXT := 2;
                      BITS := COMSZ * 8;
                    END;
       END;
END;


(*****************************************************************************)

PROCEDURE SETCONTIME(I: INTEGER);

(*****************************************************************************)

(* SET CONTROLLER -I- TO EITHER RESPOND TO NEXT VAX COMMAND OR NOT RESPOND*)

VAR
        J : INTEGER;
        MSG : INTEGER;

BEGIN
      MSG := 0;
      CONT[I].RESPTM := CLOCK;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
WITH CONT[I] DO
   REPEAT
       IF NUMRET = 0 THEN  (* IF ALL CMDS HAVE BEEN RESPONDED TO *)
          BEGIN
              RESPTM := BIGNUM; (*DISALLOW RESPONSE UNTIL ANOTHER VAX CMD*)
              FOR J := 1 TO MAXNOVAX DO
                   WITH VAX[J] DO
                      IF CONNECT[I] = 1 THEN
                       BEGIN
                          VAX[J].OKTX[I] := 1; (*VAX CAN TX AGAIN*)
                          (*IF REQ'D, SET VAX TIME TO CURRENT CLOCK TIME*)
                          IF VAX[J].VC[I].TXTIME < CLOCK THEN
                           VAX[J].VC[I].TXTIME := CLOCK;
                       END;
          END;


   IF RESPTM <> BIGNUM THEN
   BEGIN
      (*RESPOND TO CURRENT CMD? DEPENDS ON NEXT AND RSP VARS*)
      (*IN FIRST CASE WILL SEND RESPONSE TO VAX COMMAND *)
       IF ((C[1]<RSP[1]) AND (NEXT=1)) THEN
           BEGIN
                   MSG := 1;
                   C[1] := C[1] + 1;
                   IF C[2] < RSP[2] THEN
                           NEXT := 2;
           END

        ELSE
           (*WILL NOT SEND RESPONSE TO VAX COMMAND *)
           IF ((C[2]<RSP[2]) AND (NEXT=2)) THEN
                   BEGIN
                        MSG := 2;
                        NUMRET := NUMRET - 1;
                        (* NO MORE COMMANDS TO CONSIDER RESPONDING TO*)
                        IF NUMRET = 0 THEN
                           BEGIN
                              RESPTM := BIGNUM;
                              FOR J := 1 TO MAXNOVAX DO
                                 WITH VAX[J] DO
                                   IF CONNECT[I] = 1 THEN
                                     BEGIN
                                       VAX[J].OKTX[I] := 1;
                                       IF VAX[J].VC[I].TXTIME < CLOCK THEN
                                           VAX[J].VC[I].TXTIME := CLOCK;
                                     END;
                           END;

                           C[2] := C[2] + 1;
                           IF C[1] < RSP[1] THEN
                                   NEXT := 1;
                   END
           ELSE
              BEGIN  (* RESTART SEQUENCE OF EVENTS - SEND RESPONSE*)
                   C[1] := 1;
```

```
                        C[2] := 0;
                        MSG := 1;
                        IF C[2] < RSP[2] THEN
                                NEXT := 2;
                    END;

                (* ADD DLY TO RESPONSE TM - DOESN'T MEAN SEND THE RESPONSE*)
                RESPTM := RESPTM + AD;
        END;

        (* UNTIL NO MORE COMMANDS TO CONSIDER OR A RESPONSE NEEDS TO BE SENT *)
        UNTIL ((NUMRET = 0) OR (MSG = 1));
END;

(****************************************************************************)

PROCEDURE SETXTIMES(VAR IX : INTEGER);

(****************************************************************************)

(* AFTER STARTUP THIS ROUTINES INITIALIZES THE VAX & CONT TO BEGIN COMM *)

VAR
        I, J : INTEGER;
        VAL: REAL;

BEGIN
          (* CONTROLLER -IX- COMING ON-LINE, VAX -J- WILL COMM WITH THIS CONT *)
          J := CONT[IX].CONNTO;
          (* SET VAX INITIAL PARAMETERS FOR THIS PARTICULAR CONT *)
          WITH VAX[J] DO
            BEGIN
                      OKTX[IX] := 1;
                      WITH VC[IX] DO
                        BEGIN
                          C[1] := 0;
                          C[2] := 0;
                          NEXT := 1;
                          TXTIME := CLOCK;
                          SETVAXTIME(J,IX);
                        END;
             END;

            GETVAXNXT(J);

               (* SET CONTROLLER TO BEGIN ARCHIVING TO VAX *)
                WITH CONT[IX] DO
                  BEGIN
                        FOR J:=1 TO NO8614 DO
                           BEGIN
                              WITH ARC[J] DO
                                BEGIN
                                  TXTM := 0;
                                  REPEAT
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
                              TXTM := TXTM + FILLTM;
                             UNTIL TXTM > CLOCK;
                            END;
                       END;
                    C[1] := 0;
                    C[2] := 0;
                    NEXT := 1;
               END;

            GETCONTNXT(IX);

END;

(*************************************************************************)

PROCEDURE UPVAX( I : INTEGER);

(*************************************************************************)

(* VAX IS TRANSMITTING, UPDATE OTHER OPERATIONS OF VAX TO CURRENT CLOCK TIME. *)

VAR
   J : INTEGER;

BEGIN

WITH VAX[I] DO
   BEGIN
       IF ACK < CLOCK THEN
           ACK := CLOCK;
       FOR J := 1 TO MAXNOCONT DO
           IF WDTXTM[J] < CLOCK THEN
              WDTXTM[J] := CLOCK;
       IF VAXDOG < CLOCK THEN
              VAXDOG := CLOCK;
       FOR J := 1 TO MAXNOCONT DO
           IF CONNECT[J] = 1 THEN
              BEGIN
               IF VC[J].TXTIME < CLOCK THEN
                   VC[J].TXTIME := CLOCK;
               IF CONT[J].INSTART = 1 THEN
                   IF VS[J].TXTIME < CLOCK THEN
                        VS[J].TXTIME := CLOCK;
              END;
   END;
END;

(*************************************************************************)

PROCEDURE UPCONT( I : INTEGER);

(*************************************************************************)

(* CONT IS TRANSMITTING, UPDATE OTHER OPERATIONS OF CONT TO CURRENT CLOCK TIME*)
```

```
VAR
   J : INTEGER;

BEGIN

WITH CONT[I] DO
   BEGIN
       IF ACK < CLOCK THEN
           ACK := CLOCK;
       IF CONTDOG < CLOCK THEN
           CONTDOG := CLOCK;
       FOR J := 1 TO NO8614 DO
           IF ARC[J].TXTM < CLOCK THEN
               ARC[J].TXTM := CLOCK;
       IF NUMRET > 0 THEN
           IF RESPTM < CLOCK THEN
               RESPTM := CLOCK;
       IF INSTART = 1 THEN
           IF CS.TXTIME < CLOCK THEN
               CS.TXTIME := CLOCK;
       IF COMMTM < CLOCK THEN
           COMMTM := CLOCK;
   END;
END;


(******************************************************************************)

PROCEDURE FINDNEXT;

(******************************************************************************)

(* FINDS NEXT SMALLEST TRANSMIT TIME. ALTERNATES BETWEEN CONT'S & VAX'S    *)
(* TO ALLOW THE OPPOSITE OF THE ONE THAT JUST TRANSMITTED THE FIRST CHANCE *)
(* AT TRANSMITTING NEXT.                                                   *)

VAR
        I : INTEGER;
        TIME : REAL;

BEGIN
        TIME := BIGNUM;
        IF TXIX > MAXNOCONT THEN
            BEGIN  (* VAX TRANSMITTED LAST, ALLOW CONT TO TX FIRST THIS TIME*)
                FOR I := 1 TO MAXDEVTOT DO
                    IF TX[I] < TIME THEN
                        BEGIN
                            TIME := TX[I];
                            TXIX := I;
                        END;
            END
        ELSE  (* CONT TX'D LAST, ALLOW VAX TO TX FIRST THIS TIME *)
            FOR I := MAXDEVTOT DOWNTO 1 DO
                IF TX[I] < TIME THEN
                  BEGIN
```

```
                        TIME := TX[I];
                        TXIX := I;
                  END;
END;

(****************************************************************************)

PROCEDURE UPTIME(INX : INTEGER; SUM: REAL);

(****************************************************************************)

(* USED DURING A COLLISION TO ADD WAITING TIMES TO THE DEVICE INVOLVED IN COLL*)
(* IF SUM = 16, THEN 16 COLLISIONS OCCURRED, AND THE PACKET IS LOST *)

VAR
        I : INTEGER;

BEGIN

IF INX IN [1..MAXNOCONT] THEN  (* CONTROLLER TRANSMITTING*)
  BEGIN
   IF PART[INX] = -1 THEN
      BEGIN
        IF SUM = MAXCOLLS THEN
           CONT[INX].ACK:=CONT[INX].ACK+MINDELAY+PDELAY[TXIX,INX]+32/EBUSRATE
        ELSE
           CONT[INX].ACK := CURTIME + SUM;
        END

   ELSE  (* RESPONSE TO WATCH DOG TIMER MEG FROM VAX *)
      IF PART[INX] = 0 THEN
        BEGIN
         IF SUM = MAXCOLLS THEN
            CONT[INX].CONTDOG:=CONT[INX].CONTDOG+MINDELAY+PDELAY[TXIX,INX]
                +32/EBUSRATE
         ELSE
            CONT[INX].CONTDOG := CURTIME + SUM;
        END

     ELSE  (* HOTFIRE OPERATION BETWEEN CONT'S*)
        IF PART[INX] = -2 THEN
          BEGIN
             IF SUM = MAXCOLLS THEN
                CONT[INX].COMMTM := CONT[INX].COMMTM +MINDELAY+PDELAY[TXIX,INX]
                        +32/EBUSRATE
             ELSE
                CONT[INX].COMMTM := CURTIME + SUM;
          END

      ELSE   (* CONTROLLER SENDING RESPONSE TO VAX COMMAND *)
            IF PART[INX] = CONT[INX].NO8614+1 THEN
                BEGIN
                IF SUM = MAXCOLLS THEN
                        CONT[INX].RESPTM:=CONT[INX].RESPTM + MINDELAY
                                +PDELAY[TXIX,INX]+32/EBUSRATE
                ELSE
```

```
                    CONT[INX].RESPTM := CURTIME + SUM;
                END

            ELSE  (* CONTROLLER SENDING ARCHIVE MESSAGE *)
                IF PART[INX] IN [1..CONT[INX].NO8614] THEN
                  BEGIN
                     WITH CONT[INX] DO
                       IF SUM = MAXCOLLS THEN
                         ARC[PART[INX]].TXTM:=ARC[PART[INX]].TXTM + MINDELAY +
                              PDELAY[TXIX,INX] + 32/EBUSRATE
                       ELSE
                         ARC[PART[INX]].TXTM:= CURTIME + SUM;
                  END

                ELSE    (* START UP FOR CONT PART[INX] - STARTNUM *)
                  IF PART[INX] >= STARTNUM THEN
                         BEGIN
                         IF SUM = MAXCOLLS THEN
                            CONT[PART[INX]-STARTNUM].CS.TXTIME :=
                                  CONT[PART[INX]-STARTNUM].CS.TXTIME +
                                  MINDELAY + PDELAY[TXIX,INX] + 32/EBUSRATE
                         ELSE
                            CONT[PART[INX]-STARTNUM].CS.TXTIME :=
                                  CURTIME + SUM;
                         END;
  END
ELSE   (* VAX TRANSMITTING*)
  BEGIN
   IF PART[INX] = -1 THEN   (* SENDING ACK MSG BACK TO CONT *)
     BEGIN
        IF SUM = MAXCOLLS THEN
           VAX[INX-MAXNOCONT].ACK := VAX[INX - MAXNOCONT].ACK + MINDELAY
                       + PDELAY[TXIX,INX] + 32/EBUSRATE
        ELSE
        BEGIN
           VAX[INX-MAXNOCONT].ACK := CURTIME + SUM;
        END;
     END

    ELSE  (*WATCH DOG TIMER MSG FROM VAX TO CONT *)
      IF PART[INX] = 0 THEN
        BEGIN
          IF SUM = MAXCOLLS THEN
            VAX[INX-MAXNOCONT].VAXDOG := VAX[INX-MAXNOCONT].VAXDOG + MINDELAY
                       +PDELAY[TXIX,INX] + 32/EBUSRATE
          ELSE
            VAX[INX-MAXNOCONT].VAXDOG := CURTIME + SUM;
        END

      ELSE  (* MESSAGE TO CONTROLLER *)
        IF PART[INX] IN [1..MAXNOCONT] THEN
          BEGIN
             I := INX - MAXNOCONT;
             IF SUM = MAXCOLLS THEN
               VAX[I].VC[PART[INX]].TXTIME := VAX[I].VC[PART[INX]].TXTIME +
                     MINDELAY + PDELAY[TXIX,INX] + 32/EBUSRATE
```

```
            ELSE
        BEGIN
               VAX[I].VC[PART[INX]].TXTIME := CURTIME + SUM;
        END;
         END

        ELSE  (* START UP FOR CONT PART[INX] - STARTNUM *)
           IF PART[INX] >= STARTNUM THEN
              BEGIN
                I := INX - MAXNOCONT;
                IF SUM = MAXCOLLS THEN
                    VAX[I].VS[PART[INX]-STARTNUM].TXTIME :=
                        VAX[I].VS[PART[INX]-STARTNUM].TXTIME +
                        MINDELAY + PDELAY[TXIX,INX] + 32/EBUSRATE
                ELSE
                    VAX[I].VS[PART[INX]-STARTNUM].TXTIME :=
                       CURTIME + SUM;
              END;
     END;
END;

(***************************************************************************)

PROCEDURE COLLISION(INX: INTEGER; VAR SUM1: REAL; VAR SUM2: REAL);

(***************************************************************************)

(* CALCULATES WAIT TIMES FOR THE STATIONS WHICH COLLIDED *)

VAR
  X: REAL;
  I, LP : INTEGER;
  X1, X2: REAL;
BEGIN

        (* THE LEAST AMOUNT OF TIME THAT EITHER DEVICE WILL HAVE TO WAIT IS *)
        (* CALCULATED HERE. MINIMUM BUS DELAY + JAM TIME + TIME USED ON BUS*)
        SUM1 := ABS(TX[INX] - TX[TXIX]) + MINDELAY;
        SUM1 := SUM1 + 32/EBUSRATE;
        SUM2 := SUM1;

        (* USE EXPONENTIAL BACKOFF ALGORITHM TO CALCULATE WAIT TIME FOR *)
        (* DEVICE WHICH COLLIDED WITH TRANSMITTING PACKET *)
        X := 1;
        LP := STAT[INX].NUMCOLS;
        IF LP <> 0 THEN
           BEGIN
                IF LP > 10 THEN
                   LP := 10;
                FOR I:= 1 TO LP DO
                   X := X * 2;
           END;
        X1 := RAN(IDUM);
        SUM1 := SUM1 + X1 * X * 51.2E-6;

        (* USE EXPONENTIAL BACKOFF ALGORITHM TO CALCULATE WAIT TIME FOR *)
```

```
        (* TRANSMITTING DEVICE  *)
        X := 1;
        LP := STAT[TXIX].NUMCOLS;
        IF LP <> 0 THEN
           BEGIN
                IF LP > 10 THEN
                   LP := 10;
                FOR I:= 1 TO LP DO
                   X := X * 2;
           END;
        X2 := RAN(IDUM);
        SUM2 := SUM2 + X2 * X * 51.2E-6;

        (* TALLY NUMBER OF COLLISIONS, CH:CK FOR MAXIMUM ALLOWED *)
        STAT[INX].NUMCOLS := STAT[INX].NUMCOLS + 1;
        IF ACOLL = 0 THEN
           STAT[TXIX].NUMCOLS := STAT[TXIX].NUMCOLS + 1;
        IF STAT[INX].NUMCOLS >= 16 THEN
        BEGIN
           STAT[INX].NUMCOLS := 0;
           SUM1 := 0.0;
        END;
        IF STAT[TXIX].NUMCOLS >= 16 THEN
        BEGIN
           STAT[TXIX].NUMCOLS := 0;
           SUM2 := 0.0;
        END;
END;
(*****************************************************************************)

PROCEDURE COLLIDE(INX : INTEGER; SUM: REAL);
(*****************************************************************************)

(* USED TO UPDATE THE STATISTIC OF DEVICES WHICH COLLIDED *)

BEGIN

        WITH STAT[INX] DO
             BEGIN
                NOCOLS := NOCOLS + 1;
                IF NUMCOLS > MAXPKTCOLS THEN  (*MAX NUM COLLS FOR ANY 1 PKT*)
                        MAXPKTCOLS := NUMCOLS
                ELSE
                    IF NUMCOLS = 0 THEN
                        BEGIN
                        MAXPKTCOLS := 16
                        END;

                COLTIME := COLTIME + SUM;  (* TOTAL DEVICE COLL TIME*)
                PKTCOLTIME := PKTCOLTIME + SUM; (*FIND MAX PKT COLL WAIT TIME*)
                IF PKTCOLTIME > MAXCOLTIME THEN
                        MAXCOLTIME := PKTCOLTIME;

                IF (MINWAIT > SUM) THEN MINWAIT := SUM; (*MIN PKT WAIT TIME *)
```

```
              IF (MAXWAIT < SUM) THEN
                 BEGIN
                    MAXWAIT := SUM; (*MAX PKT WAIT TIME *)
                 END;

           END;
END;
(*****************************************************************************)

PROCEDURE ACOLLISION;

(*****************************************************************************)

(* CHECKS DEVICE WHICH WAS SELECTED TO TRANSMIT AGAINST THOSE WHICH ARE IN    *)
(* LINE TO TRANSMIT. IF COLLISION OCCURS THEN UPDATE TRANSMIT TIMES ACCORDING*)
(* TO ETHERNET PROTOCOL SPECIFICATIONS - EXPONENTIAL BACKOFFF ALGORITHM       *)


VAR
      I, J: INTEGER;
      SUM1, SUM2: REAL;
      PD: REAL;
      BIGBITS: INTEGER;
      SAVE: REAL;
      NEWCOLFLG: INTEGER;

BEGIN

      SUM1 := 0.0; (*WAIT TIME OF COLLIDING DEVICE *)
      SUM2 := 0.0; (*WAIT TIME OF TRANSMITTING DEVICE *)
      ACOLL := 0;  (*FLAGS THAT A COLLISION DID OCCUR FOR THIS PACKET *)
      PD := 0.0;   (*LARGEST PROPAGATION DELAY BETWEEN ANY COLLIDING DEVICES*)
      BIGBITS := 0;(*LARGEST SLOT TIME OFFERED TO BUS *)
      NEWCOLFLG := 0;

      FOR I := 1 TO MAXDEVTOT DO
             IF I <> TXIX THEN
                  (* CHECK FOR A COLLISION BY COMPARING THE DIFFERENCE IN *)
                  (* TX TIMES WITH THE PROPAGATION DLY BETWEEN 2 DEVICES *)
                  IF (TX[I] - CURTIME) < PDELAY[TXIX,I] THEN
                          BEGIN
                             (* CALCULATE DEVICE WAIT TIMES *)
                             COLLISION(I,SUM1,SUM2);
 (*WRITELN('C T=',TXIX:2,' P=',PART[TXIX]:2,' C1=',STAT[TXIX].NUMCOLS:2,
 ' S2=',SUM2:9,' T2=',STAT[TXIX].COLTIME:5:2,' C/W=',I:2,' P=',PART[I]:2,
 ' C2=',STAT[I].NUMCOLS:2,' S1=',SUM1:9,' T1=',STAT[I].COLTIME:5:2);      *)
                             IF NEWCOLFLG = 0 THEN
                                IF STAT[I].NUMCOLS = 1 THEN
                                  BEGIN
                                    NEWPKTCOLS := NEWPKTCOLS + 1;
                                    NEWCOLFLG := 1;
                                  END;
                             IF ACOLL = 0 THEN ACOLL := 1;
                             IF PD < PDELAY[TXIX,I] THEN
                               PD := PDELAY[TXIX,I];
```

```
                    (* UPDATE DEVICE NEXT TRANSMIT TIMES*)
                    IF STAT[I].NUMCOLS = 0 THEN
                       BEGIN
                         UPTIME(I, MAXCOLLS);
                         SUM1 := MINDELAY + PD + 32/EBUSRATE;
                       END
                      ELSE
                         UPTIME(I, SUM1);
                    (* UPDATE DEVICE COLLISION STATISTICS *)
                    COLLIDE(I, SUM1);

                    SAVE := CLOCK;
                    CLOCK := CURTIME + SUM1;
                    IF I <= MAXNOCONT THEN
                            UPCONT(I)
                    ELSE
                            UPMAX(I-MAXNOCONT);
                    TX[I] := CLOCK;
                    CLOCK := SAVE;

                    (* TOTAL DATA OFFERED - NOT OFFERED LOAD *)
                    IF STAT[I] NUMCOLS = 1 THEN
                      BEGIN
                        TOTDATA := TOTDATA + BIT[I];
                        BADOFF := BADOFF + BIT[I];
                      END;
                    TOTBITS := TOTBITS + 512;
                 END;

IF ACOLL = 1 THEN (* UPDATE TRANSMITTER TO REFLECT COLLISION *)
   BEGIN
      TOTBITS := TOTBITS - BIT[TXIX];
      TOTBITS := TOTBITS + 512;
      (* UPDATE TRANSMITTER'S NEXT TRANSMIT TIME *)
      IF STAT[TXIX].NUMCOLS = 0 THEN
        BEGIN
          UPTIME(TXIX, MAXCOLLS);
          SUM2 := MINDELAY + PD + 32/EBUSRATE;
        END
      ELSE
          UPTIME(TXIX,SUM2);
     (* UPDATE TRANSMITTER COLLISION STATISTICS *)
     COLLIDE(TXIX, SUM2);

      IF NEWCOLFLG = 0 THEN
          IF STAT[TXIX].NUMCOLS = 1 THEN
              BEGIN
                NEWPKTCOLS := NEWPKTCOLS + 1;
                NEWCOLFLG := 1;
              END;

     SAVE := CLOCK;
     CLOCK := CURTIME + SUM2;
     IF TXIX <= MAXNOCONT THEN
          UPCONT(TXIX)
     ELSE
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
                UPVAX(TXIX-MAXNOCONT);
            TX[TXIX] := CLOCK;
            CLOCK := SAVE;

            (* UPDATE TIMES AND STATS TO REFLECT COLLISION *)
            TOTCOLS := TOTCOLS + 1;
            BUSBUSY := BUSBUSY + PD + MINDELAY + 32/EBUSRATE;
            CLOCK := CLOCK + PD + MINDELAY + 32/EBUSRATE;
          END;
END;

(*************************************************************************)

PROCEDURE CHECKDEFER;

(*************************************************************************)

(* COMPARE TRANSMITTER WITH OTHER TRANSMITTING DEVICES TO SEE IF THAT DEVICE *)
(* MUST DEFER ITS PACKET TRANSMISSION UNTIL AFTER THE CURRENT TX IS COMPLETE *)

VAR
        I: INTEGER;
        SUM: REAL;
        SAVE: REAL;

BEGIN
        FOR I := 1 TO MAXDEVTOT DO
            IF I <> TXIX THEN
                BEGIN
                   (* CHECK FOR A PACKET DEFER CONDITION BY COMPARING THE *)
                   (* DIFFERENCE IN TRANSMIT TIMES TO DECIDE IF A PACKET  *)
                   (* MUST WAIT UNTIL ANOTHER TRANSMISSION IS COMPLETE *)
                   IF (TX[I]-TX[TXIX]) < (BIT[TXIX]/EBUSRATE +
                        MINDELAY + PDELAY[TXIX,I]) THEN
                      BEGIN
(* WRITELN(ABTEMP,'D T= ',TXIX:2,' P= ',PART[TXIX]:2,' D/W= ',I:2,' P= ',
                PART[I]:2,'TM= ',CURTIME);                              *)
                        SUM :=  BIT[TXIX]/EBUSRATE;
                        UPTIME(I, SUM); (* MAKE DEVICE WAIT SUM TIME *)
                        SUM := SUM + MINDELAY - (TX[I]-TX[TXIX]);
                        IF SUM < 0.0 THEN SUM := 0.0;

                        (* UPDATE DEVICE STATS FOR DEFER CONDITION *)
                        STAT[I].DERTIME := STAT[I].DERTIME + SUM;
                        STAT[I].NODER := STAT[I].NODER + 1;
                        IF ((STAT[I].MINWAIT > SUM) AND (SUM <> 0.0)) THEN
                                STAT[I].MINWAIT := SUM;
                        IF STAT[I].MAXWAIT < SUM THEN
                                STAT[I].MAXWAIT := SUM;
                        IF STAT[I].NUMCOLS <> 0 THEN
                                STAT[I].PKTCOLTIME := STAT[I].PKTCOLTIME + SUM;

                        (* SET OTHER DEVICE TX TIMES TO UPDATED TIME AND *)
                        (* SELECT THE NEXT OPERATION TO BE TRANSMITTED *)
                        SAVE := CLOCK;
```

```
                    CLOCK := TX[I] + SUM - MINDELAY;
                    IF I <= MAXNOCONT THEN
                       BEGIN
                          UPCONT(I);
                          GETCONTNXT(I);
                       END
                    ELSE
                       BEGIN
                          UPVAX(I-MAXNOCONT);
                          GETVAXNXT(I-MAXNOCONT);
                       END;
                    CLOCK := SAVE;
                  END;
            END;
END;

(****************************************************************************)

PROCEDURE SETSTARTUP(I: INTEGER);

(****************************************************************************)

(* INITIALIZES COMM VARS BETWEEN VAX AND CONTROLLER TO BRING ON-LINE*)

VAR
        J: INTEGER;

BEGIN

        (* CONT -I- TALKS TO VAX -J- BRING CONT ON-LINE*)
        J := CONT[I].CONNTO;
        WITH VAX[J].VS[I] DO
                BEGIN
                   TXIX := J + MAXNOCONT; (* SET APPROPRIATE VAX TO TX *)
                   STAT := 1; (* =1 VAX TURN TO TX, =2 CONT RESPONSES*)
                   COUNT := 1; (* NUM PKTS SENT SO FAR VARIABLE *)
                   CLOCK := CONT[I].CS.STARTIME; (* CLOCK = CONT START TIME*)
                   (* SET VAX WATCHDOG TIMER TO BEGIN 1 SEC AFTER STARTUP*)
                   VAX[J].WDTXTM[I] := ROUND(CLOCK);
                   VAX[J].WDTXTM[I] := VAX[J].WDTXTM[I] + 2.0;
                   (* SET NEXT TX TIME FOR VAX WATCHDOG OPERATION*)
                   IF VAX[J].WDTXTM[I] < VAX[J].VAXDOG THEN
                        BEGIN
                          VAX[J].VAXDOG := VAX[J].WDTXTM[I];
                          VAX[J].WHODOG := I;
                        END;
                   (* INITIALIZE VAX TRANSMIT TIME & NUM BITS FOR THIS CONT *)
                   TXTIME := CLOCK;
                   BITS := PKTSENTSZ * 8;
                   CONT[I].INSTART := 1;  (*CONT IN STARTUP MODE*)
                   GETVAXNXT(J); (*FIND NEXT VAX TRANSMISSION *)
                   CONT[I].CS.STARTIME := BIGNUM;(*DON'T STARTUP CONT AGAIN*)
                   CONT[I].CS.COUNT := 0; (*NUM PKTS SENT SO FAR*)
                   CONT[I].CS.STAT := SMALLPKT; (* VAX TURN UNTIL =2*)
                END;
```

```
        IF HOTFIRE = 1 THEN                (*IF HOTFIRE ALLOWED AND TVC&AFT *)
           IF CONT[TVC].CS.STARTIME = BIGNUM THEN (* THEN BEGIN HOTFIRE OP*)
                 IF CONT[CONT[TVC].TALKTO].CS.STARTIME = BIGNUM THEN
                   BEGIN
                      (* START HOTFIRE 2 SECONDS AFTER CONT STARTUP *)
                      CONT[TVC].COMMTM := TRUNC(CLOCK) + 2.0;
                      CONT[TVC].NUMSENT := 0;
                      CONT[CONT[TVC].TALKTO].NUMSENT := 0;
                      (* DON'T RE-START HOTFIRE OPERATION! *)
                      CONT[TVC].CS.STARTIME := BIGNUM * 2.0;
                   END;

END;

(*****************************************************************************)

PROCEDURE UPDATE;

(*****************************************************************************)

(* UPDATE DEVICE WHICH JUST TRANSMITTED TO ALLOW ITS NEXT TRANSMISSION AND   *)
(* SET DEVICES FOR APPROPRIATE RESPONSES TO MESSAGE JUST SENT                *)

VAR
        I, J : INTEGER;
        OKDOG : INTEGER;
        MAXLOOP: INTEGER;

BEGIN

CLOCK := CURTIME + BIT[TXIX] / EBUSRATE;
BUSBUSY := BUSBUSY + BIT[TXIX]/EBUSRATE + MINDELAY;
USAGE := USAGE + BIT[TXIX]/EBUSRATE;

(*IF TXIX <=MAXNOCONT THEN
WRITELN(ABTEMP,'T=',TXIX:2,' P=',PART[TXIX]:3,' B=',BIT[TXIX]:5,
' CL=',CLOCK:9:5,' NK= ',CONT[TXIX].NOACK:2,' NC=',STAT[TXIX].NUMCOLS:2,
' NO=',STAT[TXIX].NOCOLS:4,' CT=',STAT[TXIX].COLTIME:9:5,' PT=',
STAT[TXIX].PKTSRX:4)
ELSE
WRITELN(ABTEMP,'T=',TXIX:2,' P=',PART[TXIX]:3,' B=',BIT[TXIX]:5,
' CL=',CLOCK:9:5,' NK= ',VAX[TXIX-MAXNOCONT].NOACK:2,' NC=',
STAT[TXIX].NUMCOLS:2,' NO=',STAT[TXIX].NOCOLS:4,' CT=',STAT[TXIX].COLTIME:9:5,
' PT=',STAT[TXIX].PKTSRX:4);                                             *)

    IF PART[TXIX] <> -1 THEN
      IF PART[TXIX] <> 0 THEN
        IF PART[TXIX] <> -2 THEN
           IF ((LAST<>-2) AND (PART[TXIX]<>-1)) THEN
              IF TXIX <= MAXNOCONT THEN
          VAX[CONT[TXIX].CONNTO].RXDATA:=VAX[CONT[TXIX].CONNTO].RXDATA+BIT[TXIX]
                                 - 26*8  (*LESS HEADER BITS *)
              ELSE
          VAX[TXIX-MAXNOCONT].TXDATA := VAX[TXIX-MAXNOCONT].TXDATA + BIT[TXIX]
```

```
                              - 26*8; (*LESS HEADER BITS *)

IF TXIX IN [1..MAXNOCONT] THEN  (* CONTROLLER TRANSMITTED*)

   IF PART[TXIX] = -1 THEN (* CONT TX'D ACK TO VAX*)
        BEGIN
         CONT[TXIX].ACK := BIGNUM;
         CONT[TXIX].NOACK := CONT[TXIX].NOACK - 1;
         IF CONT[TXIX].NOACK > 0 THEN
                CONT[TXIX].ACK := CLOCK;
         STAT[TXIX].ACKSTX := STAT[TXIX].ACKSTX + 1;
         STAT[TXIX].PKTSRX := STAT[TXIX].PKTSRX + 1;
(*WRITELN(ABTEMP,' ');*)
        END

   ELSE
      IF PART[TXIX] = 0 THEN  (* WATCHDOG TIMER RESPONSE*)
        BEGIN
            CONT[TXIX].CONTDOG := BIGNUM;
        END

     ELSE
        IF PART[TXIX] = -2 THEN  (* TVC-AFT COMMUNICATION *)
           BEGIN
            CONT[CONT[TXIX].TALKTO].NOACK := CONT[CONT[TXIX].TALKTO].NOACK+1;
                IF CONT[CONT[TXIX].TALKTO].NOACK = 1 THEN
                        CONT[CONT[TXIX].TALKTO].ACK := CLOCK;
                NUMHOT := NUMHOT + 1; (* KEEPS UP WITH NUMBER SENT IN A SECOND*)

                IF TXIX = TVC THEN   (*CAN ONLY BE TVC OF ONE OF THE AFT'S*)
                   BEGIN
                        WITH CONT[TXIX] DO
                          BEGIN
                             NUMSENT := NUMSENT + 1;
                             COMMTM := BIGNUM;
                             IF NUMSENT = MSGSEC THEN
                                   NUMSENT := 0;
                             CONT[TALKTO].COMMTM := CLOCK + CONT[TALKTO].DLYTM;
                             GETCONTNXT(TALKTO);
                          END;
                   END
                ELSE
                   BEGIN
                        WITH CONT[TXIX] DO
                          BEGIN
                                 NUMSENT := NUMSENT + 1;
                                 COMMTM := BIGNUM;
                                 HOTSTART := 1;
                                 IF NUMSENT = MSGSEC THEN
                                    BEGIN
                                      NUMSENT := 0;
                                      CONT[CONT[TXIX].TALKTO].COMMTM :=
                                         TRUNC(CLOCK) + 1.0;
                                      IF HOTFAIL = 1 THEN
                                          HOTSTART := 0;
                                    END
```

```
                        ELSE
                           BEGIN
                              CONT[TALKTO].COMMTM := CLOCK +
                                   CONT[TALKTO].DLYTM;
                              GETCONTNXT(TALKTO);
                           END;
                  END;
            END;
      END

ELSE
   BEGIN
      VAX[CONT[TXIX].CONNTO].NOACK := VAX[CONT[TXIX].CONNTO].NOACK+1;
      IF VAX[CONT[TXIX].CONNTO].NOACK = 1 THEN
          VAX[CONT[TXIX].CONNTO].ACK := CLOCK;
      GETVAXNXT(CONT[TXIX].CONNTO);

      IF PART[TXIX] = CONT[TXIX].NO8614+1 THEN (*RESPONSE TO VAX*)
         BEGIN
            CONT[TXIX].NUMRET := CONT[TXIX].NUMRET - 1;
            SETCONTIME(TXIX);
            FOR J := 1 TO MAXNOVAX DO
                IF VAX[J].CONNECT[TXIX] = 1 THEN
                    GETVAXNXT(J);
         END

      ELSE
         IF PART[TXIX] IN [1..CONT[TXIX].NO8614] THEN
            BEGIN      (* ARCHIVE MESSAGE SENT *)
              WITH CONT[TXIX] DO
                 ARC[PART[TXIX]].TXTM:=CLOCK + ARC[PART[TXIX]].FILLTM;
            END

         ELSE
            IF PART[TXIX] >= STARTNUM THEN
                  BEGIN
                     CONT[PART[TXIX]-STARTNUM].CS.TXTIME := BIGNUM;
                     WITH CONT[PART[TXIX]-STARTNUM].CS DO
                      IF STAT = 1 THEN
                           WITH VAX[CONT[TXIX].CONNTO].VS[TXIX] DO
                                BEGIN
                                   TXTIME := CLOCK + 0.002;
                                   BITS := PKTSENTSZ * 8;
                                   GETVAXNXT(CONT[TXIX].CONNTO);
                                 END
                        ELSE
                             BEGIN
                               COUNT := COUNT + 1;
                               IF COUNT <= NUMPKTSRET THEN
                                 BEGIN
                                    TXTIME := CLOCK + 0.002;
                                    BITS := PKTSZRET * 8;
                                 END
                               ELSE
                                  BEGIN
                                    STAT := SMALLPKT;
```

```
                                                  WITH VAX[CONT[TXIX].CONNTO].VS[TXIX] DO
                                                    BEGIN
                                                      TXTIME := CLOCK+0.002;
                                                      BITS := PKT10GSZ * 8;
                                                      GETVAXNXT(CONT[TXIX].CONNTO);
                                                    END;
                                                END
                                           END;
                                    END;
    END

ELSE   (* VAX TRANSMITTED*)
    IF PART[TXIX] = -1 THEN
         BEGIN
            VAX[TXIX-MAXNOCONT].ACK := BIGNUM;
            VAX[TXIX-MAXNOCONT].NOACK := VAX[TXIX-MAXNOCONT].NOACK - 1;
            IF VAX[TXIX-MAXNOCONT].NOACK > 0 THEN
                VAX[TXIX-MAXNOCONT].ACK := CLOCK;
            STAT[TXIX].ACKSTX := STAT[TXIX].ACKSTX + 1;
            STAT[TXIX].PKTSRX := STAT[TXIX].PKTSRX + 1;
(*WRITELN(ASTEMP,' ');   *)
         END

    ELSE
       IF PART[TXIX] = 0 THEN    (* WATCHDOG TIMER MSG FROM VAX*)
           BEGIN
              J := TXIX-MAXNOCONT;
              CONT[VAX[J].WHODOG].CONTDOG := CLOCK + CONT[VAX[J].WHODOG].AD;
              GETCONTNXT(VAX[J].WHODOG);
          VAX[J].WDTXTM[VAX[J].WHODOG]:=TRUNC(VAX[J].WDTXTM[VAX[J].WHODOG])+1.0;
              OKDOG := 0;
              MAXLOOP := 0;
              REPEAT
                 VAX[J].WHODOG := VAX[J].WHODOG + 1;
                 MAXLOOP := MAXLOOP + 1;
                 IF VAX[J].WHODOG > MAXNOCONT THEN
                    VAX[J].WHODOG := 1;
                 IF VAX[J].CONNECT[VAX[J].WHODOG] <> 0 THEN
                    OKDOG := 1;
                 IF ((VAX[J].OKTX[VAX[J].WHODOG] = 0) AND
                    (CONT[VAX[J].WHODOG].INSTART IN [0..1])) THEN
                        OKDOG := 0;
                 IF MAXLOOP = MAXNOCONT THEN
                        OKDOG := 1;
              UNTIL (OKDOG = 1);
              VAX[J].VAXDOG := VAX[J].WDTXTM[VAX[J].WHODOG];
           END

        ELSE                      (* MESSAGE TO CONTROLLER *)
          IF PART[TXIX] IN [1..MAXNOCONT] THEN
           BEGIN
              CONT[PART[TXIX]].NOACK := CONT[PART[TXIX]].NOACK+1;
              IF CONT[PART[TXIX]].NOACK = 1 THEN
                     CONT[PART[TXIX]].ACK := CLOCK;
              J := TXIX-MAXNOCONT;
              WITH VAX[J].VC[PART[TXIX]] DO
```

```
    IF NUMBCTX>1 THEN
       BEGIN
         NUMBCTX := NUMBCTX - 1;
         TXTIME := CLOCK + 0.002;
       END
    ELSE
       BEGIN
          SETVAXTIME(J, PART[TXIX]);
          VAX[J].CKTX[PART[TXIX]] := 0;
          WITH CONT[PART[TXIX]] DO
             BEGIN
                NUMRET := 1;
                IF BIT[TXIX] > VAX[J].VC[PART[TXIX]].COMSZ * 8 THEN
                      NUMRET := VAX[J].VC[PART[TXIX]].BC;
                SETCONTIME(PART[TXIX]);
              END;
       END;
    GETCONTNXT(PART[TXIX]);
 END

ELSE
   IF PART[TXIX] >= STARTNUM THEN
        BEGIN
CONT[PART[TXIX]-STARTNUM].NOACK := CONT[PART[TXIX]-STARTNUM].NOACK+1;
            IF CONT[PART[TXIX]-STARTNUM].NOACK = 1 THEN
                 CONT[PART[TXIX]-STARTNUM].ACK := CLOCK;
            GETCONTNXT(PART[TXIX]-STARTNUM);
            J := TXIX - MAXNOCONT.
            WITH VAX[J].VS[PART[TXIX]-STARTNUM] DO
            IF STAT = 1 THEN
              BEGIN
                  TXTIME := BIGNUM.
                  COUNT := COUNT + 1;
                  WITH CONT[PART[TXIX]-STARTNUM].CS DO
                    BEGIN
                        TXTIME := CLOCK+0.002;
                        BITS := ACKPKTSZ * 8;
                        GETCONTNXT(PART[TXIX]-STARTNUM);
                        STAT := 1;
                    END;
                  IF COUNT > NUMPKTSENT THEN
                    BEGIN
                     CONT[PART[TXIX]-STARTNUM].CS.STAT := 2;
                     STAT := 2;
                     COUNT := 1;
                    END;
              END
             ELSE
              BEGIN
                  COUNT := COUNT + 1;
                  IF COUNT <= 100 THEN (* PUT BACK TO 100 ON DELIVERABLE*)
                    BEGIN
                     TXTIME := CLOCK+0.002;
                     BITS := PKT100SZ * 8;
                     IF COUNT = 20 THEN
                       BEGIN
```

```
                                        PKT100SZ :  80;
                                        NUM100 := 0;
                                      END;
                                   END
                                 ELSE
                                   BEGIN
                                     CONT[PART[TXIX]-STARTNUM].INSTART := SMALLPKT;
                                     TXTIME := BIGNUM;
                                     I := PART[TXIX] - STARTNUM;
                                     SETXTIMES(I);
                                   END;
                               END;
                         END;

IF PART[TXIX] <> -1 THEN
   BEGIN
        STAT[TXIX].PKTSTX := STAT[TXIX].PKTSTX + 1;
        STAT[TXIX].PKTSRX := STAT[TXIX].PKTSRX + 1;
   END;

STAT[TXIX].NUMCOLS := 0;
STAT[TXIX].PKTCOLTIME := 0.0;
IF TXIX <= MAXNOCONT THEN
     BEGIN
          UPCONT(TXIX);
          GETCONTNXT(TXIX);
     END
ELSE
     BEGIN
          UPVAX(TXIX-MAXNOCONT);
          GETVAXNXT(TXIX-MAXNOCONT);
     END;
LAST := PART[TXIX];
END;

(***************************************************************************)

PROCEDURE PRTRES;
(***************************************************************************)

(* PRINT RESULTS OF SIMULATION RUN TO OUTPUT FILE FOR PRINTOUT *)

VAR
        I,J : INTEGER;
        AVGBB, AVGUS, AVGID : REAL;
        T1, T2: REAL;
        L1,L2,L3,L5,S1 : REAL;
        L4, TOTDER, TOTRX, TOTACK: INTEGER;
        BYTES : INTEGER;
        OFFDATA, BADLOAD : REAL;
        EFF2, EFF3: REAL;
        KT : INTEGER;
        INDAT: CHAR;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
BEGIN
TOTDER := 0;
TOTRX := 0;
TOTACK := 0;
L1 := 0.0;
L2 := 0.0;
L3 := 0.0;
L4 := 0;
L5 := 0.0;
S1 := BIGNUM;


WRITELN(ABOUT,' ');
WRITELN(ABOUT,'SOURCE  WAIT TIME   WAIT TIME   DEFER COLL  ',
' PKTS  ACKS  PKTS  MINIMUM PKT  MAXIMUM PKT  MAX NUM  MAX PKT ');
WRITELN(ABOUT,'         DEFER        COLLISION   COUNT COUNT ',
'  TX    TX    RX   WAIT TIME    WAIT TIME    COLLS    COLL TIME');
WRITELN(ABOUT,'------------------------------------------------------------',
'------------------------------------------------------');

FOR I := 1 TO MAXDEVTOT DO
        IF STAT[I].DISTANCE <> 0 THEN
            WITH STAT[I] DO
                BEGIN
WRITELN(ABOUT,NAM[I]:4,'   ',DERTIME:10,'  ',COLTIME:10,'  ',
NODER:5,' ',NOCOLS:5,'  ',PKTSTX:5,' ',ACKSTX:5,' ',PKTSRX:5,' ',MINWAIT:10,
'  ',MAXWAIT:10,'         ',MAXPKTCOLS:2,'    ',MAXCOLTIME:10);
TOTPKTSTX := TOTPKTSTX + PKTSTX;
L1 := L1 + DERTIME;
L2 := L2 + COLTIME;
TOTDER := TOTDER + NODER;
TOTRX := TOTRX + PKTSRX;
TOTACK := TOTACK + ACKSTX;
IF (MINWAIT < S1) AND (MINWAIT <> 0.0) THEN S1 := MINWAIT;
IF MAXWAIT > L3 THEN L3 := MAXWAIT;
IF MAXPKTCOLS > L4 THEN L4 := MAXPKTCOLS;
IF MAXCOLTIME > L5 THEN L5 := MAXCOLTIME;
                END;

WRITELN(ABOUT,'------------------------------------------------------------',
'------------------------------------------------------');
WRITELN(ABOUT,'TOT ':4,'   ',L1:10,'  ',L2:10,'  ',TOTDER:5,
' ',TOTCOLS:5,'  ',TOTPKTSTX:5,' ',TOTACK:5,' ',TOTRX:5,' ',
S1:10,'  ',L3:10,'         ',L4:2,'    ',L5:10);
WRITELN(ABOUT,'------------------------------------------------------------',
'------------------------------------------------------');


WRITELN(ABOUT,' ');
TOTPKTSTX := TOTPKTSTX + TOTACK;
AVGBB := BUSBUSY / TOTPKTSTX;
AVGUS := USAGE / TOTPKTSTX;
IDLE := SIMTIME - BUSBUSY;
AVGID := IDLE / TOTPKTSTX;
WRITELN(ABOUT,'AVERAGE   BUSBUSY: ',AVGBB:10:6,'  USAGE: ',AVGUS:10:6,
'  IDLE: ',AVGID:10:6);
```

```
WRITELN(ABOUT,'TOTAL     BUSBUSY: ',BUSBUSY:10:6,'  USAGE: ',USAGE:10:6,
'  IDLE: ',IDLE:10:6);

WRITELN(ABOUT,' ');
SIMTHRUPUT := AVGUS / ( AVGBB + AVGID);
OFFLOAD := TOTBITS / (SIMTIME * EBUSRATE);
BADLOAD := BADOFF / (SIMTIME * EBUSRATE);
OFFDATA := TOTDATA / (SIMTIME * EBUSRATE);
EFFICIENCY := SIMTHRUPUT / OFFLOAD;
EFF2 := SIMTHRUPUT / BADLOAD;
EFF3 := SIMTHRUPUT / OFFDATA;
T1 := OFFLOAD * EXP(-(1.27E-9 * OFFLOAD));
T2 := OFFLOAD * (1 + 2 * 1.27E-9) + EXP(-(1.27E-9 * OFFLOAD));
THEORETICAL := T1 / T2;

SIMTHRUPUT := SIMTHRUPUT * 100.0;
WRITELN(ABOUT,'S = SIMULATED THROUGHPUT:                ',SIMTHRUPUT:10:6);
OFFLOAD := OFFLOAD * 100.0;
BADLOAD := BADLOAD * 100.0;
OFFDATA := OFFDATA * 100.0;
WRITELN(ABOUT,'G = OFFERED LOAD AS A % OF BUS CAPACITY: ',OFFLOAD:10:6,
' (',BADLOAD:10:6,')      TOTAL OFFERED DATA:      ',OFFDATA:10:6);
EFFICIENCY := EFFICIENCY * 100.0;
EFF2 := EFF2 * 100.0;
EFF3 := EFF3 * 100.0;
WRITELN(ABOUT,'E = EFFICIENCY (OFF LOAD):               ',EFFICIENCY:10:6,
' (',EFF2:10:6,')      EFFICIENCY: (OFF DATA): ',EFF3:10:6);
THEORETICAL := THEORETICAL * 100.0;
WRITELN(ABOUT,'T = THEORETICAL THROUGHPUT:              ',THEORETICAL:10:6);
WRITELN(ABOUT,' ');

IF HOTFIRE = 1 THEN
IF HOTMIN < TOTMSGSEC THEN
  WRITELN(ABOUT,'TVC-AFT COMM: ',TOTMSGSEC:3,' MSGS NOT TRANSMITTED IN 1 ',
              'SECOND');

FOR I:= 1 TO MAXNOVAX DO
   BEGIN
   IF RXMAX[I] <> 0 THEN
    BEGIN
    BYTES := RXMAX[I] DIV 8;
    WRITELN(ABOUT,'VAX: ',I:2,' RECEIVED ',BYTES:8,' BYTES IN 1 SECOND');
    END;

   IF TXMAX[I] <> 0 THEN
    BEGIN
    BYTES := TXMAX[I] DIV 8;
    WRITELN(ABOUT,'VAX: ',I:2,' TRANSMITTED ',BYTES:8,' BYTES IN 1 SECOND');
    END;
   END;

(*WRITELN(ABOUT,' ');
WRITELN(ABOUT,'NUMBER OF COLLISIONS INVOLVING NON-REPEATED DATA PACKETS ',
               NEWPKTCOLS);                                            *)

WRITELN(ABOUT,' ');
```

```
WRITELN(ABOUT,' ');
WRITELN(ABOUT,'SECOND   VAX    RECEIVED BITS   TRANSMITTED BITS  ');
WRITELN(ABOUT,'-----------------------------------------------------');
FOR I:= 1 TO MAXNOVAX DO
  FOR J := 1 TO TMLINE DO
     BEGIN
        IF ((RXSEC[I,J] <> 0) OR (TXSEC[I,J] <> 0)) THEN
         WRITELN(ABOUT,'    ',J:2,'     ',I:2,'       ',RXSEC[I,J],'            ',
                TXSEC[I,J]);
     END;

IF OPCHAR = 'Y' THEN
    BEGIN

RESET(ASF);
IF EOF(ASF) THEN
        BEGIN
                REWRITE(ASF);
                WRITELN(ASF,'                                                    ',
                '       SIMULATION RUN RESULTS ');
                WRITELN(ASF,' ');
                WRITELN(ASF,' ');

WRITELN(ASF,'RUN  OFF       SIM       THEO                WAIT TIME ',
'WAIT TIME ',
'DEFER COLL  PKTS ACKS PKTS MIN PKT   MAX PKT   MAX NUM  MAX PKT ');
WRITELN(ASF,'NUM  LOAD      THRPT     THRPT     EFF       DEFER     ',
'COLLISION ',
'COUNT COUNT  TX   TX   RX  WAIT TIME WAIT TIME COLLS    COLL',
' TIME');
WRITELN(ASF,'--------------------------------------------------------------------',
'------------------------------------------------------------------------');
        END;

DAT[1] := 'TEMP            ';
BIND(ABDATA,DAT[1],ISTAT);
RESET(ASF);
KT := -5;
REWRITE(ABDATA);
WHILE NOT EOF(ASF) DO
  BEGIN
        KT := KT + 1;
        WHILE NOT EOLN(ASF) DO
           BEGIN
                READ(ASF,INDAT);
                WRITE(ABDATA,INDAT);
           END;
        READLN(ASF);
        WRITELN(ABDATA);
  END;

WRITELN(ABDATA,KT:3,' ',OFFLOAD:9,' ',SIMTHRUPUT:9,' ',THEORETICAL:9,
' ',EFFICIENCY:9,' ',L1:9,' ',L2:9,
' ',TOTDER:5,' ',TOTCCLS:5,' ',TOTPKTST(:5,' ',TOTACK:5,
' ',TOTRX:5,' ',S1:9,' ',L3:9,'  ',L4:2.'  ',L5:9);
```

```
CLOSE(ABDATA);
BIND(ABDATA,DAT[1],ISTAT);
          RESET(ABDATA);
          REWRITE(ASF);
          WHILE NOT EOF(ABDATA) DO
            BEGIN
                  WHILE NOT EOLN(ABDATA) DO
                     BEGIN
                        READ(ABDATA,INDAT);
                        WRITE(ASF,INDAT);
                     END;
                  READLN(ABDATA);
                  WRITELN(ASF);
          END;

          CLOSE(ABDATA);
     END;
END;

(****************************************************************************)

BEGIN (* MAIN ROUTINE *)

(****************************************************************************)

          INITIALIZE;

          CONFIGURE;

IF SL = 4 THEN
   BEGIN
          PRTDATA;
          REWRITE(ABTEMP);
          CLOCK := BIGNUM;
          FOR I := 1 TO MAXNOCONT DO
                  IF CONT[I].CS.STARTIME < CLOCK THEN
                           CLOCK := CONT[I].CS.STARTIME;


REPEAT
          FINDNEXT;

          FOR I := 1 TO MAXNOCONT DO
                  IF CONT[I].CS.STARTIME < TX[TXIX] THEN
                           SETSTARTUP(I);

          CURTIME := TX[TXIX];
          IF CURTIME < (CLOCK + MINDELAY) THEN
                  CURTIME := CLOCK + MINDELAY;

          TOTBITS := TOTBITS + BIT[TXIX];
          BADOFF := BADOFF + BIT[TXIX];
          IF STAT[TXIX].NUMCOLS = 0 THEN
             TOTDATA := TOTDATA + BIT[TXIX];
```

```
         IF CLOCK > TMLINE THEN
            BEGIN
               TMLINE := TMLINE + 1;
               FOR I := 1 TO MAXNOVAX DO
                 BEGIN
                   IF VAX[I].RXDATA > VAX[I].NUMRX*8 THEN
                      IF VAX[I].RXDATA > RXMAX[I] THEN
                         RXMAX[I] := VAX[I].RXDATA;
                   IF VAX[I].TXDATA > VAX[I].NUMTX*8 THEN
                      IF VAX[I].TXDATA > TXMAX[I] THEN
                         TXMAX[I] := VAX[I].TXDATA;
WRITELN(ABTEMP,'TM = ',CLOCK,'  RXDATA = ',VAX[I].RXDATA,'  TXDATA = ',
         VAX[I].TXDATA);
                   RXSEC[I,TMLINE-1] := VAX[I].RXDATA;
                   TXSEC[I,TMLINE-1] := VAX[I].TXDATA;
                   VAX[I].RXDATA := 0;
                   VAX[I].TXDATA := 0;
                 END;

                   IF ((HOTFIRE = 1) AND  HOTSTART = 1)) THEN
                       BEGIN
                         HOTFAIL := 0;
                         IF NUMHOT < TOTMSGSEC THEN
                            BEGIN
                               IF NUMHOT < HOTMIN THEN
                                  HOTMIN := NUMHOT;
                               HOTFAIL := 1;
                            END;
                       END;
                    IF CONT[TVC].NUMSENT <> 1 THEN NUMHOT := 0;
         END;

         ACOLLISION;

         IF ACOLL = 0 THEN
            BEGIN
                 CHECKDEFER;
                 UPDATE;
            END;

UNTIL (CLOCK >= SIMTIME);


FOR I := 1 TO MAXNOVAX DO
    BEGIN
WRITELN(ABTEMP,'TM = ',CLOCK,'  RXDATA = ',VAX[I].RXDATA,'  TXDATA = ',
         VAX[I].TXDATA);
         RXSEC[I,TMLINE] := VAX[I].RXDATA
         TXSEC[I,TMLINE] := VAX[I].TXDATA
    END;

OPCHAR := 'Y';
REPEAT
         WRITELN(' ');
         WRITELN('ADD RUN INFORMATION TO SUMMARY CHARTS? "Y" OR "N": ');
         READLN(OPCHAR);
```

```
UNTIL((OPCHAR = 'Y') OR (OPCHAR = 'N'));

PRTRES;

   END;
END.
```

APPENDIX III.

PARAMETERS AND RESULTS OF VARIOUS SCENARIOS

CONTENTS

SIMULATION DESCRIPTION: SIM RUN 1
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX TX:	# PKTS,	PKT SZ(B)	CONT RESP: # PKTS,	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
IEA	2.10	1	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.200	0.400	100	1	25	80
IEA	0.550	0.750	100	1	50	80

CONT	DIST (FT)	RESP 1:	RESP 2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:		2:		3:	4:	5:	6:	7:
FWD1	1445.0	1	100	80	2.0E-03	2	3.100	1500	3.200	1500					
IEA	1485.0	1	100	80	2.0E-03	1	6.200	1500							

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	1.226E-03	1.170E-04	2	2	46	166	212	3.008E-05	1.196E-03	2	1.313E-03
FWD2	0.000E+00	0.000E+00	0	0	0	0	0	1.000E+04	0.000E+00	0	0.000E+00
DFI	0.000E+00	0.000E+00	0	0	0	0	0	1.000E+04	0.000E+00	0	0.000E+00
EMU	0.000E+00	0.000E+00	0	0	0	0	0	1.000E+04	0.000E+00	0	0.000E+00
AFT1	0.000E+00	0.000E+00	0	0	0	0	0	1.000E+04	0.000E+00	0	0.000E+00
AFT2	0.000E+00	0.000E+00	0	0	0	0	0	1.000E+04	0.000E+00	0	0.000E+00
TVC	0.000E+00	0.000E+00	0	0	0	0	0	1.000E+04	0.000E+00	0	0.000E+00
IEA	8.146E-10	1.696E-05	15	1	40	139	179	1.550E-11	1.696E-05	1	1.696E-05
ASA	0.000E+00	0.000E+00	0	0	0	0	0	1.000E+04	0.000E+00	0	0.000E+00
VSWR	0.000E+00	0.000E+00	0	0	0	0	0	1.000E+04	0.000E+00	0	0.000E+00
VAX1	0.000E+00	4.255E-05	0	1	337	55	392	4.255E-05	4.255E-05	1	4.255E-05
VAX2	0.000E+00	0.000E+00	0	0	0	0	0	1.000E+04	0.000E+00	0	0.000E+00
TOT	1.226E-03	1.765E-04	17	2	423	360	783	1.550E-11	1.196E-03	2	1.313E-03

AVERAGE BUSBUSY: 0.000281 USAGE: 0.000271 IDLE: 0.025262
TOTAL BUSBUSY: 0.219927 USAGE: 0.212384 IDLE: 19.780073

S = SIMULATED THROUGHPUT: 1.061920
G = OFFERED LOAD AS A % OF BUS CAPACITY: 1.062944 (1.074240) TOTAL OFFERED DATA: 1.061920
E = EFFICIENCY (OFF LOAD): [illegible] ([illegible]) EFFICIENCY (OFF DATA): 100.000000
T = THEORETICAL THROUGHPUT: 1.051764

163

ORIGINAL PAGE IS OF POOR QUALITY

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
2	1	0	11792
3	1	235840	376960
4	1	236704	447712
5	1	0	1728
6	1	0	2160
7	1	35376	2160
8	1	0	2160
9	1	0	2160
10	1	23584	2160
11	1	0	2160
12	1	0	1728
13	1	35376	2592
14	1	0	1728
15	1	0	2592
16	1	11792	2160
17	1	11792	1296
18	1	0	2592
19	1	23584	1728
20	1	11792	2592

SIMULATION DESCRIPTION: SIM RUN 2
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX	TX: # PKTS,	PKT SZ(B)	CONT RESP: # PKTS,	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	15	0.034	80
TVC	1.00	1	10	1500	10	1500	15	0.033	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.200	0.400	100	1	25	80
FWD2	0.500	0.800	100	1	50	80
DFI	0.450	0.650	100	1	25	80
AFT1	0.300	0.600	100	1	50	80
AFT2	0.200	0.350	100	1	25	80
TVC	0.100	0.250	100	1	25	80
IEA	0.550	0.750	100	1	50	80
ASA	[illegible]	[illegible]	100	1	25	80
VSWR	0.530	0.680	100	1	25	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
EMU	0.200	0.400	6	1	15	80

CONT	DIST (FT)	RESP 1:	2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	100	80	2.0E-03	7	0.400	1500	2.400	1500	5.000	1500	7.500	1500	2.000	1500	1.000	1500	1.600	1500
FWD2	1465.0	1	100	80	2.0E-03	7	0.410	1500	2.410	1500	5.100	1500	7.510	1500	2.100	1500	1.100	1500	1.610	1500
DFI	1455.0	1	100	80	2.0E-03	3	0.500	1500	0.700	1500	0.900	1500								
EMU	1405.0	1	5	80	2.0E-03	7	0.420	1500	2.420	1500	5.200	1500	7.520	1500	2.200	1500	1.200	1500	1.620	1500
AFT1	20.0	1	100	80	2.0E-03	5	2.430	1500	0.430	1500	0.800	1500	1.200	1500	0.850	1500				
AFT2	30.0	1	100	80	2.0E-03	5	2.440	1500	0.440	1500	0.810	1500	1.210	1500	0.860	1500				
TVC	10.0	1	100	80	2.0E-03	4	0.200	1500	0.450	1500	0.500	1500	1.220	1500						
IEA	1485.0	1	100	80	2.0E-03	3	0.460	1500	0.510	1500	0.600	1500								
ASA	1495.0	1	100	80	2.0E-03	0														
VSWR	1475.0	1	100	80	2.0E-03	0														

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	2.987E-02	6.134E-02	72	74	122	171	293	4.460E-11	2.793E-02	6	3.171E-02
FWD2	7.083E-03	4.513E-03	19	23	55	121	176	6.429E-06	2.649E-03	7	5.942E-03
DFI	1.347E-02	3.077E-02	38	41	88	134	222	4.460E-11	2.779E-02	5	3.116E-02
EMU	1.335E-02	1.009E-03	34	30	126	157	283	7.363E-11	1.207E-03	3	1.817E-03
AFT1	1.168E-02	4.041E-02	42	48	79	132	211	4.460E-11	2.774E-02	7	2.882E-02
AFT2	2.374E-02	9.985E-03	62	76	346	377	723	4.460E-11	2.936E-03	8	7.708E-03
TVC	3.574E-02	1.355E-01	83	131	418	420	838	1.282E-05	2.791E-02	9	3.693E-02
IEA	3.834E-02	1.580E-02	84	73	118	135	253	7.453E-06	4.595E-03	8	1.178E-02
ASA	1.189E-02	3.356E-03	34	33	34	131	165	1.028E-10	1.194E-03	6	4.373E-03
VSWR	1.043E-02	2.170E-03	26	27	35	140	175	1.295E-05	1.191E-03	5	4.389E-03
VAX1	1.049E-01	9.110E-02	206	292	1473	803	2276	1.198E-06	2.771E-02	6	2.771E-02
VAX2	1.941E-02	2.279E-02	50	77	170	114	284	2.435E-08	7.911E-03	9	1.700E-02
TOT	3.199E-01	4.187E-01	755	392	3064	2835	5899	4.460E-11	2.793E-02	9	3.693E-02

AVERAGE	BUSBUSY:	0.000325	USAGE:	0.000315	IDLE:	0.003065
TOTAL	BUSBUSY:	1.917309	USAGE:	1.855280	IDLE:	18.082691

S = SIMULATED THROUGHPUT: 9.276400
G = OFFERED LOAD AS A % OF BUS CAPACITY: 9.513200 (10.791120)
E = EFFICIENCY (OFF LOAD): 97.510827 (85.963273)
T = THEORETICAL THROUGHPUT: 8.686807

TOTAL OFFERED DATA: 9.319720
EFFICIENCY: (OFF DATA): 99.545860

TVC-AFT COMM: 30 MSGS NOT TRANSMITTED IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	283440	400544
2	1	365984	425356
3	1	424512	392208
4	1	354624	401840
5	1	448096	353808
6	1	401792	379552
7	1	330176	41424
8	1	542864	358128
9	1	271216	26176
10	1	400928	9504
11	1	283008	5616
12	1	531072	369920
13	1	412720	30928
14	1	271648	8208
15	1	637200	359856
16	1	377344	41424
17	1	518848	9936
18	1	400928	6912
19	1	412720	17840
20	1	495264	10800
7	2	271648	388752
8	2	72048	10256
9	2	48464	9824
10	2	58960	1296
11	2	47600	432
12	2	48464	10256
13	2	59824	10688
14	2	47168	432
15	2	59392	864
16	2	47600	1296
17	2	47168	1296
18	2	24880	10256
19	2	47600	864
20	2	72048	10688

ORIGINAL PAGE IS
OF POOR QUALITY

SIMULATION DESCRIPTION: SIM RUN 3
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX	TX: # PKTS,	PKT SZ(B)	CONT RESP: # PKTS,	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	15	0.034	80
TVC	1.00	1	10	1500	10	1500	15	0.033	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD:	%	NO SZ(B)
FWD1	0.200	0.400	100	1	25	80
FWD2	0.250	0.450	100	1	50	80
DFI	0.150	0.350	100	1	25	80
AFT1	0.100	0.300	100	1	50	80
AFT2	0.200	0.350	100	1	25	80
TVC	0.100	0.250	100	1	25	80
IEA	0.200	0.500	100	1	50	80
ASA	0.250	0.500	100	1	25	80
VSWR	0.150	0.350	100	1	25	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD:	%	NO SZ(B)
EMU	0.200	0.400	6	1	15	80

CONT	DIST (FT)	RESP 1:	2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	100	80	2.0E-03	7	0.160	1500	0.960	1500	2.000	1500	3.000	1500	0.800	1500	0.400	1500	0.640	1500
FWD2	1465.0	1	100	80	2.0E-03	7	0.804	1500	0.964	1500	2.040	1500	3.004	1500	0.840	1500	0.440	1500	0.644	1500
DFI	1455.0	1	100	80	2.0E-03	3	0.200	1500	0.280	1500	0.360	1500								
EMU	1405.0	1	5	80	2.0E-03	7	0.168	1500	0.968	1500	2.080	1500	3.008	1500	0.880	1500	0.480	1500	0.648	1500
AFT1	20.0	1	100	80	2.0E-03	5	0.972	1500	0.172	1500	0.320	1500	0.480	1500	0.340	1500				
AFT2	30.0	1	100	80	2.0E-03	5	0.976	1500	0.176	1500	0.324	1500	0.484	1500	0.344	1500				
TVC	10.0	1	100	80	2.0E-03	4	0.080	1500	0.180	1500	0.200	1500	0.486	1500						
IEA	1485.0	1	100	80	2.0E-03	3	0.184	1500	0.204	1500	0.240	1500								
ASA	1495.0	1	100	80	2.0E-03	0														
VSWR	1475.0	1	100	80	2.0E-03	0														

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	5.163E-02	5.343E-03	106	99	255	165	420	1.550E-11	1.202E-03	6	3.805E-03
FWD2	1.236E-02	7.069E-03	32	31	70	129	199	6.740E-06	4.539E-03	6	4.539E-03
DFI	2.884E-02	5.600E-03	73	70	173	163	336	1.284E-05	1.301E-03	6	4.074E-03
EMU	3.272E-02	1.148E-02	81	86	224	156	380	8.113E-06	5.535E-03	8	1.027E-02
AFT1	3.223E-02	3.246E-03	64	68	156	162	318	9.601E-06	1.208E-03	4	1.730E-03
AFT2	6.949E-02	1.713E-02	149	151	532	395	927	1.550E-11	5.097E-03	8	1.303E-02
TVC	7.232E-02	1.351E-01	183	216	710	436	1146	1.391E-10	2.545E-02	8	2.597E-02
IEA	2.720E-02	2.938E-02	74	64	243	157	400	2.672E-06	2.544E-02	5	2.799E-02
ASA	1.005E-02	2.777E-02	24	21	34	153	187	4.460E-11	2.544E-02	6	2.544E-02
VSWR	4.731E-03	6.609E-04	15	18	35	186	221	8.294E-06	1.181E-03	2	1.285E-03
VAX1	2.497E-01	1.161E-01	430	564	1630	1686	3316	7.986E-07	2.378E-02	7	2.608E-02
VAX2	5.356E-02	4.015E-02	112	160	168	212	380	1.568E-07	7.873E-03	8	1.342E-02
TOT	6.449E-01	3.990E-01	1343	696	4230	4000	8230	1.550E-11	2.545E-02	8	2.799E-02

AVERAGE BUSBUSY: 0.000390 USAGE: 0.000379 IDLE: 0.002040
TOTAL BUSBUSY: 3.208622 USAGE: 3.120016 IDLE: 16.791378

S = SIMULATED THROUGHPUT: 15.600080
G = OFFERED LOAD AS A % OF BUS CAPACITY: 15.996368 (18.384320)
E = EFFICIENCY (OFF LOAD): 97.522637 (84.855355)
T = THEORETICAL THROUGHPUT: 13.790404

TOTAL OFFERED DATA: 15.644320
EFFICIENCY: (OFF DATA): 99.717213

TVC-AFT COMM: 30 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 168090 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	342400	400544
2	1	613616	426720
3	1	696160	416656
4	1	637200	369056
5	1	766912	393072
6	1	837664	405296
7	1	707520	44016
8	1	990960	408752
9	1	919776	12096
10	1	825440	9504
11	1	801856	11232
12	1	1167840	386464
13	1	1167408	13824
14	1	990960	14688
15	1	1344720	378128
16	1	1108448	47904
17	1	1073072	15120
18	1	1025904	36976
19	1	1214576	14688
20	1	1037696	12528
7	2	296096	351216
8	2	141504	864
9	2	141936	1728
10	2	142800	864
11	2	129712	864
12	2	143232	10688
13	2	176880	1296
14	2	72048	10688
15	2	165520	1296
16	2	118784	10256
17	2	141936	864
18	2	131008	10256
19	2	165520	1728
20	2	130576	10256

SIMULATION DESCRIPTION: SIM RUN 4
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX TX: #	PKTS,	PKT SZ(B)	CONT RESP: # PKTS,	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	15	0.034	80
TVC	1.00	1	10	1500	10	1500	15	0.033	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.200	0.400	100	1	25	80
FWD2	0.250	0.450	100	1	50	80
DFI	0.150	0.350	100	1	25	80
AFT1	0.100	0.300	100	1	50	80
AFT2	0.200	0.350	100	1	25	80
TVC	0.100	0.250	100	1	25	80
IEA	0.200	0.500	100	1	50	80
ASA	0.250	0.500	100	1	25	80
VSWR	0.150	0.350	100	1	25	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
EMU	0.200	0.400	6	1	15	80

CONT	DIST (FT)	RESP 1:	2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:	2:	3:	4:	5:	6:	7:
FWD1	1445.0	1	100	80	2.0E-03	7	0.080 1500	0.480 1500	1.000 1500	1.500 1500	0.400 1500	0.200 1500	0.320 1500
FWD2	1465.0	1	100	80	2.0E-03	7	0.082 1500	0.482 1500	1.020 1500	1.502 1500	0.420 1500	0.220 1500	0.322 1500
DFI	1455.0	1	100	80	2.0E-03	3	0.100 1500	0.140 1500	0.180 1500				
EMU	1405.0	1	5	80	2.0E-03	7	0.084 1500	0.484 1500	1.040 1500	1.504 1500	0.440 1500	0.240 1500	0.324 1500
AFT1	20.0	1	100	80	2.0E-03	5	0.486 1500	0.086 1500	0.160 1500	0.240 1500	0.170 1500		
AFT2	30.0	1	100	80	2.0E-03	5	0.488 1500	0.088 1500	0.016 1500	0.242 1500	0.172 1500		
TVC	10.0	1	100	80	2.0E-03	4	0.040 1500	0.090 1500	0.100 1500	0.244 1500			
IEA	1485.0	1	100	80	2.0E-03	3	0.092 1500	0.102 1500	0.120 1500				
ASA	1495.0	1	100	80	2.0E-03	0							
VSWR	1475.0	1	100	80	2.0E-03	0							

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	1.445E-01	1.831E-02	306	282	474	166	640	4.460E-11	1.209E-03	5	5.325E-03
FWD2	5.670E-02	8.962E-03	120	118	167	128	295	6.201E-06	1.856E-03	7	3.754E-03
DFI	1.122E-01	5.864E-02	219	207	312	163	475	2.310E-06	2.033E-02	10	4.075E-02
EMU	1.217E-01	1.910E-02	253	248	395	157	552	1.362E-06	2.206E-03	7	3.360E-03
AFT1	8.703E-02	1.397E-02	186	194	282	161	443	1.367E-06	1.270E-03	7	4.148E-03
AFT2	3.643E-01	9.054E-02	841	878	1835	423	2258	1.550E-11	6.021E-03	9	1.378E-02
TVC	3.018E-01	1.104E-01	661	693	1188	468	1656	3.543E-07	4.219E-02	11	6.671E-02
IEA	1.293E-01	2.482E-02	284	268	449	156	605	5.973E-07	2.618E-03	8	1.122E-02
ASA	2.286E-02	9.949E-03	48	56	34	154	188	4.460E-11	3.289E-03	8	9.146E-03
VSWR	1.860E-02	1.940E-02	35	45	35	183	218	4.460E-11	8.411E-03	10	2.213E-02
VAX1	1.143E+00	1.650E-01	1417	2089	1624	4193	5817	7.225E-07	6.566E-03	9	1.309E-02
VAX2	1.676E-01	3.487E-01	304	434	169	383	552	1.827E-06	5.051E-02	12	1.028E-01
TOT	2.669E+00	8.877E-01	4674	2451	6964	6735	13699	1.550E-11	5.051E-02	12	1.028E-01

AVERAGE	BUSBUSY:	0.000487	USAGE:	0.000475	IDLE:	0.000973
TOTAL	BUSBUSY:	6.677434	USAGE:	6.511792	IDLE:	13.322566

S = SIMULATED THROUGHPUT: 32.558960
G = OFFERED LOAD AS A % OF BUS CAPACITY: 33.970032 (44.175600)
E = EFFICIENCY (OFF LOAD): 95.846126 (73.703492)
T = THEORETICAL THROUGHPUT: 25.356441

TOTAL OFFERED DATA: 32.630240
EFFICIENCY: (OFF DATA): 99.781552

TVC-AFT COMM: 30 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 383240 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	802288	400544
2	1	1568768	426288
3	1	1757440	417520
4	1	1875360	417520
5	1	2052240	396528
6	1	2193744	419248
7	1	2122560	44016
8	1	2512128	422272
9	1	2358400	11664
10	1	2393776	10800
11	1	2381984	13392
12	1	2783344	435360
13	1	2782912	14256
14	1	2700800	13392
15	1	3042768	390784
16	1	3054128	50064
17	1	3006960	16848
18	1	3030544	14256
19	1	3065920	39568
20	1	3065920	15120
7	2	389568	376528
8	2	295664	10256
9	2	283440	1728
10	2	284304	10256
11	2	271216	1296
12	2	284736	10688
13	2	306592	1296
14	2	296096	10256
15	2	283440	1728
16	2	284304	10688
17	2	307888	10688
18	2	259424	1296
19	2	283440	1296
20	2	296096	10688

SIMULATION DESCRIPTION: SIM RUN 5
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX TX: #	PKTS,	PKT SZ(B)	CONT RESP: # PKTS,	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	15	0.034	80
TVC	1.00	1	10	1500	10	1500	15	0.033	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.200	0.400	100	1	25	80
FWD2	0.250	0.450	100	1	50	80
DFI	0.150	0.350	100	1	25	80
AFT1	0.100	0.300	100	1	50	80
AFT2	0.200	0.350	100	1	25	80
TVC	0.100	0.250	100	1	25	80
IEA	0.200	0.500	100	1	50	80
ASA	0.250	0.500	100	1	25	80
VSWR	0.150	0.350	100	1	25	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
EMU	0.200	0.400	6	1	15	80

CONT	DIST (FT)	RESP 1:	2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	100	80	2.0E-03	7	0.040	1500	0.240	1500	0.500	1500	0.750	1500	0.200	1500	0.100	1500	0.160	1500
FWD2	1465.0	1	100	80	2.0E-03	7	0.041	1500	0.241	1500	0.510	1500	0.751	1500	0.210	1500	0.110	1500	0.161	1500
DFI	1455.0	1	100	80	2.0E-03	3	0.050	1500	0.070	1500	0.090	1500								
EMU	1405.0	1	5	80	2.0E-03	7	0.042	1500	0.242	1500	0.520	1500	0.752	1500	0.220	1500	0.120	1500	0.162	1500
AFT1	20.0	1	100	80	2.0E-03	5	0.243	1500	0.043	1500	0.080	1500	0.120	1500	0.085	1500				
AFT2	30.0	1	100	80	2.0E-03	5	0.244	1500	0.044	1500	0.081	1500	0.121	1500	0.086	1500				
TVC	10.0	1	100	80	2.0E-03	4	0.020	1500	0.045	1500	0.050	1500	0.122	1500						
IEA	1485.0	1	100	80	2.0E-03	3	0.046	1500	0.051	1500	0.060	1500								
ASA	1495.0	1	100	80	2.0E-03	0														
VSWR	1475.0	1	100	80	2.0E-03	0														

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	4.879E-01	1.564E-01	879	846	897	169	1065	4.460E-11	2.085E-02	11	3.857E-02
FWD2	2.302E-01	5.739E-02	382	392	296	127	423	1.166E-06	2.553E-03	8	1.038E-02
DFI	3.455E-01	1.528E-01	621	620	581	162	743	8.234E-07	2.640E-02	12	7.483E-02
EMU	3.829E-01	6.309E-02	692	677	719	156	875	8.326E-07	5.097E-03	8	1.071E-02
AFT1	3.244E-01	1.336E-01	563	611	519	159	678	2.360E-07	2.951E-02	10	4.942E-02
AFT2	5.939E-01	3.273E-01	1100	1117	1334	361	1695	3.635E-07	5.216E-02	12	8.353E-02
TVC	7.633E-01	2.378E-01	1493	1595	1978	409	2387	8.326E-08	3.667E-02	11	7.513E-02
IEA	4.197E-01	8.205E-02	782	755	846	158	1004	2.936E-07	5.247E-03	9	1.502E-02
ASA	3.854E-02	4.425E-02	70	79	34	153	187	1.028E-10	1.910E-02	11	3.584E-02
VSWR	2.402E-02	4.117E-03	48	63	35	183	218	9.454E-06	1.206E-03	7	5.643E-03
VAX1	2.381E+00	7.122E-01	2658	4099	1623	6058	7681	3.294E-07	2.201E-02	12	8.183E-02
VAX2	4.711E-01	6.186E-01	663	1033	169	707	876	2.230E-07	5.129E-02	15	1.377E-01
TOT	6.462E+00	2.590E+00	9951	4892	9031	8802	17833	4.460E-11	5.216E-02	15	1.377E-01

AVERAGE BUSBUSY: 0.000533 USAGE: 0.000519 IDLE: 0.000539
TOTAL BUSBUSY: 9.502487 USAGE: 9.263520 IDLE: 10.497513

S = SIMULATED THROUGHPUT: 46.317599
G = OFFERED LOAD AS A % OF BUS CAPACITY: 49.360672 (72.009840)   TOTAL OFFERED DATA: 46.586960
E = EFFICIENCY (OFF LOAD): 93.835026 (64.321209)   EFFICIENCY: (OFF DATA): 99.421811
T = THEORETICAL THROUGHPUT: 33.047971

TVC-AFT COMM: 30 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 602866 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	648992	400544
2	1	1674896	426720
3	1	2052240	416224
4	1	2323456	395664
5	1	2653632	419248
6	1	2948432	420976
7	1	3042336	45744
8	1	3479072	399120
9	1	3490432	34816
10	1	3514016	12096
11	1	3572976	11232
12	1	4092256	424432
13	1	4304080	15120
14	1	4186592	14256
15	1	4717232	403440
16	1	4763968	50928
17	1	4657840	14688
18	1	4822928	37408
19	1	4728592	15984
20	1	4811136	14688
7	2	554224	376960
8	2	590896	10688
9	2	566016	1296
10	2	579104	10688
11	2	566448	1296
12	2	567312	10688
13	2	554224	1296
14	2	567312	10688
15	2	578240	1296
16	2	567312	10688
17	2	495264	1296
18	2	567312	10256
19	2	578240	1728
20	2	578672	10256

SIMULATION DESCRIPTION: SIM RUN 6
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX	TX: # PKTS	PKT SZ(B)	CONT RESP: # PKTS	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	75	0.007	80
TVC	1.00	1	10	1500	10	1500	75	0.006	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S)	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.100	0.300	10	1	10	80
FWD2	0.150	0.250	10	1	15	80
DFI	0.150	0.200	10	1	15	80
AFT1	0.100	0.250	10	1	20	80
AFT2	0.050	0.200	10	1	15	80
TVC	0.100	0.250	10	1	10	80
IEA	0.080	0.200	10	1	20	80
ASA	0.150	0.300	10	1	15	80
VSWR	0.250	0.500	10	1	10	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S)	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
EMU	0.100	0.300	6	1	15	80

CONT	DIST (FT)	RESP 1:	RESP 2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	5	80	2.0E-03	7	0.040	1500	0.240	1500	0.500	1500	0.750	1500	0.200	1500	0.100	1500	0.160	1500
FWD2	1465.0	1	6	80	2.0E-03	7	0.041	1500	0.241	1500	0.510	1500	0.751	1500	0.210	1500	0.110	1500	0.161	1500
DFI	1455.0	1	7	80	2.0E-03	3	0.050	1500	0.070	1500	0.090	1500								
EMU	1405.0	1	5	80	2.0E-03	7	0.042	1500	0.242	1500	0.520	1500	0.752	1500	0.220	1500	0.120	1500	0.162	1500
AFT1	20.0	1	5	80	2.0E-03	5	0.243	1500	0.043	1500	0.080	1500	0.120	1500	0.085	1500				
AFT2	30.0	1	4	80	2.0E-03	5	0.244	1500	0.044	1500	0.081	1500	0.121	1500	0.086	1500				
TVC	10.0	1	4	80	2.0E-03	4	0.020	1500	0.045	1500	0.050	1500	0.122	1500						
IEA	1485.0	1	6	80	2.0E-03	3	0.046	1500	0.051	1500	0.060	1500								
ASA	1495.0	1	9	80	2.0E-03	0														
VSWR	1475.0	1	4	80	2.0E-03	0														

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	5.364E-01	2.153E-01	1001	1013	924	204	1128	3.183E-07	4.679E-02	11	6.205E-02
FWD2	2.168E-01	2.645E-01	392	430	296	136	432	1.568E-07	4.653E-02	12	9.814E-02
DFI	3.562E-01	2.414E-01	702	720	600	181	781	7.737E-07	3.256E-02	12	7.153E-02
EMU	4.136E-01	1.800E-01	768	805	735	189	924	6.495E-07	4.066E-02	10	6.103E-02
AFT1	3.901E-01	2.391E-01	695	747	535	166	701	1.130E-06	4.219E-02	12	8.769E-02
AFT2	9.182E-01	4.250E-01	1731	2020	1900	949	2849	1.550E-11	4.065E-02	11	5.008E-02
TVC	1.008E+00	4.068E-01	2077	2470	2500	889	3369	3.093E-07	2.565E-02	10	4.123E-02
IEA	5.138E-01	2.495E-01	985	1014	883	229	1112	8.262E-08	2.033E-02	12	6.937E-02
ASA	5.116E-02	3.369E-02	101	117	50	181	231	1.238E-06	1.106E-02	9	1.994E-02
VSWR	4.074E-02	2.954E-02	86	92	53	159	212	1.028E-10	1.312E-02	10	2.569E-02
VAX1	2.572E+00	1.116E+00	3346	5097	1863	6305	8168	1.334E-07	4.123E-02	14	1.242E-01
VAX2	5.287E-01	5.802E-01	811	1211	201	723	924	2.435E-08	4.504E-02	13	1.109E-01
TOT	7.546E+00	3.981E+00	12695	6334	10540	10311	20851	1.550E-11	4.679E-02	14	1.242E-01

AVERAGE	BUSBUSY:	0.000468	USAGE:	0.000454	IDLE:	0.000491
TOTAL	BUSBUSY:	9.763918	USAGE:	9.475136	IDLE:	10.236082

S = SIMULATED THROUGHPUT: 47.375679
G = OFFERED LOAD AS A % OF BUS CAPACITY: 51.404096 (76.111920)
E = EFFICIENCY (OFF LOAD): 92.163238 (62.244757)
T = THEORETICAL THROUGHPUT: 33.951589

TOTAL OFFERED DATA: 47.916320
EFFICIENCY: (OFF DATA): 98.871698

ORIGINAL PAGE IS OF POOR QUALITY

TVC-AFT COMM: 150 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 594886 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	650288	410800
2	1	1688416	387472
3	1	2042608	404192
4	1	1939072	333152
5	1	2645728	417904
6	1	2553120	391264
7	1	3071104	30272
8	1	3130928	439248
9	1	3508272	56016
10	1	3166736	52320
11	1	3530560	28800
12	1	3497344	470832
13	1	4263824	68000
14	1	3603472	58944
15	1	4557328	428512
16	1	4160288	91488
17	1	4688336	46144
18	1	[illegible]	[illegible]
19	1	4759088	60336
20	1	4230176	89520
7	2	555520	398144
8	2	471680	2592
9	2	555088	2160
10	2	438032	11120
11	2	579104	11552
12	2	425376	11552
13	2	544160	11120
14	2	436736	2160
15	2	567312	11552
16	2	460320	11984
17	2	567744	11120
18	2	436736	10688
19	2	567312	11984
20	2	484336	864

SIMULATION DESCRIPTION: SIM RUN 7
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX TX: VAX	# PKTS	PKT SZ(B)	CONT RESP: # PKTS	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	75	0.007	80
TVC	1.00	1	10	1500	10	1500	75	0.006	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S)	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.100	0.300	10	1	10	80
FWD2	0.150	0.250	10	1	15	80
DFI	0.150	0.200	10	1	15	80
AFT1	0.100	0.250	10	1	20	80
AFT2	0.050	0.200	10	1	15	80
TVC	0.100	0.250	10	1	10	80
IEA	0.080	0.200	10	1	20	80
ASA	0.150	0.300	10	1	15	80
VSWR	0.250	0.500	10	1	10	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S)	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
EMU	0.100	0.300	6	1	15	80

CONT	DIST (FT)	RESP 1:	RESP 2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	5	80	2.0E-03	7	0.040	1500	0.240	1500	0.500	1500	0.750	1500	0.200	1500	0.100	1500	0.160	1500
FWD2	1465.0	1	6	80	2.0E-03	7	0.041	1500	0.241	1500	0.510	1500	0.751	1500	0.210	1500	0.110	1500	0.161	1500
DFI	1455.0	1	7	80	2.0E-03	5	0.050	1500	0.070	1500	0.090	1500	0.753	1500	0.088	1500				
EMU	1405.0	1	5	80	2.0E-03	7	0.042	1500	0.242	1500	0.520	1500	0.752	1500	0.220	1500	0.120	1500	0.162	1500
AFT1	20.0	1	5	80	2.0E-03	5	0.243	1500	0.043	1500	0.080	1500	0.120	1500	0.085	1500				
AFT2	30.0	1	4	80	2.0E-03	5	0.244	1500	0.044	1500	0.081	1500	0.121	1500	0.086	1500				
TVC	10.0	1	4	80	2.0E-03	4	0.020	1500	0.045	1500	0.050	1500	0.122	1500						
IEA	1485.0	1	6	80	2.0E-03	5	0.046	1500	0.051	1500	0.060	1500	0.754	1500	0.089	1500				
ASA	1495.0	1	9	80	2.0E-03	0														
VSWR	1475.0	1	4	80	2.0E-03	0														

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	5.334E-01	3.186E-01	999	1001	922	201	1123	4.682E-08	4.504E-02	14	1.297E-01
FWD2	2.195E-01	1.337E-01	377	403	306	138	444	3.259E-06	1.579E-02	12	4.437E-02
DFI	5.176E-01	1.919E-01	968	1007	755	181	936	8.078E-07	2.110E-02	10	4.164E-02
EMU	4.716E-01	1.711E-01	865	875	732	186	918	2.180E-07	2.327E-02	11	5.047E-02
AFT1	3.536E-01	1.922E-01	687	743	540	168	708	2.067E-07	2.609E-02	10	4.205E-02
AFT2	9.682E-01	4.019E-01	1776	2074	1898	948	2846	1.550E-11	2.103E-02	11	4.900E-02
TVC	1.169E+00	4.980E-01	2258	2593	2488	888	3376	2.230E-07	2.527E-02	13	7.903E-02
IEA	5.304E-01	5.365E-01	1101	1150	1045	231	1276	2.963E-07	4.280E-02	14	1.674E-01
ASA	4.085E-02	3.368E-02	81	100	49	177	226	1.238E-06	1.910E-02	10	3.233E-02
VSWR	4.205E-02	3.107E-02	94	94	53	160	213	3.105E-06	7.954E-03	10	2.432E-02
VAX1	2.834E+00	1.122E+00	3602	5495	1870	6629	8499	1.371E-07	2.565E-02	12	4.476E-02
VAX2	5.484E-01	5.903E-01	839	1267	198	720	918	2.315E-07	3.688E-02	15	1.586E-01
TOT	8.278E+00	4.221E+00	13637	6713	10856	10627	21483	1.550E-11	4.504E-02	15	1.674E-01

AVERAGE BUSBUSY: 0.000474 USAGE: 0.000460 IDLE: 0.000457
TOTAL BUSBUSY: 10.183489 USAGE: 9.883440 IDLE: 9.816511

S = SIMULATED THROUGHPUT: 49.417199
G = OFFERED LOAD AS A % OF BUS CAPACITY: 53.718512 (80.505600) TOTAL OFFERED DATA: 50.060400
E = EFFICIENCY (OFF LOAD): 91.992367 (61.383555) EFFICIENCY: (OFF DATA): 98.715151
T = THEORETICAL THROUGHPUT: 34.946027

TVC-AFT COMM: 150 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 630370 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	650288	410800
2	1	1688416	387472
3	1	2042608	404192
4	1	1939072	333152
5	1	2645728	417904
6	1	2494160	396864
7	1	3211312	28976
8	1	3259776	440734
9	1	3803504	56448
10	1	3330528	31312
11	1	3816160	76496
12	1	3591680	394320
13	1	4535040	73760
14	1	3873824	54288
15	1	4891392	463376
16	1	4523680	87456
17	1	5042960	62496
18	1	4363472	72112
19	1	4971776	74528
20	1	4346368	50176
7	2	555520	398576
8	2	448528	2160
9	2	567312	11120
10	2	437600	11120
11	2	555952	11552
12	2	543728	11552
13	2	554656	2592
14	2	484768	11552
15	2	555520	11552
16	2	496560	10688
17	2	544160	11984
18	2	531936	11120
19	2	531936	11120
20	2	495696	1728

SIMULATION DESCRIPTION: SIM RUN 8
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX TX: # PKTS,	PKT SZ(B)		CONT RESP: # PKTS,	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	75	0.007	80
TVC	1.00	1	10	1500	10	1500	75	0.006	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.100	0.300	10	1	10	80
FWD2	0.150	0.250	10	1	15	80
DFI	0.150	0.200	10	1	15	80
AFT1	0.100	0.250	10	1	20	80
AFT2	0.050	0.200	10	1	15	80
TVC	0.100	0.250	10	1	10	80
IEA	0.080	0.200	10	1	20	80
ASA	0.150	0.300	10	1	15	80
VSWR	0.250	0.500	10	1	10	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
EMU	0.100	0.300	6	1	15	80

CONT	DIST (FT)	RESP 1:	RESP 2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:	2:	3:	4:	5:	6:	7:
FWD1	1445.0	1	5	80	2.0E-03	7	0.040 1500	0.240 1500	0.500 1500	0.750 1500	0.200 1500	0.100 1500	0.160 1500
FWD2	1465.0	1	6	80	2.0E-03	7	0.041 1500	0.241 1500	0.510 1500	0.751 1500	0.210 1500	0.110 1500	0.161 1500
DFI	1455.0	1	7	80	2.0E-03	5	0.050 1500	0.070 1500	0.090 1500	0.753 1500	0.088 1500		
EMU	1405.0	1	5	80	2.0E-03	7	0.042 1500	0.242 1500	0.520 1500	0.752 1500	0.220 1500	0.120 1500	0.162 1500
AFT1	20.0	1	5	80	2.0E-03	5	0.243 1500	0.043 1500	0.080 1500	0.120 1500	0.085 1500		
AFT2	30.0	1	4	80	2.0E-03	5	0.244 1500	0.044 1500	0.081 1500	0.121 1500	0.086 1500		
TVC	10.0	1	4	80	2.0E-03	4	0.020 1500	0.045 1500	0.050 1500	0.122 1500			
IEA	1485.0	1	6	80	2.0E-03	5	0.046 1500	0.051 1500	0.060 1500	0.754 1500	0.089 1500		
ASA	1495.0	1	9	80	2.0E-03	4	0.470 1500	0.052 1500	0.066 1500	0.123 1500			
VSWR	1475.0	1	4	80	2.0E-03	0							

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	5.983E-01	2.372E-01	1082	1076	923	200	1127	2.159E-07	2.326E-02	12	6.931E-02
FWD2	3.561E-01	3.329E-01	552	600	297	136	433	2.046E-07	4.491E-02	12	1.195E-01
DFI	5.530E-01	2.187E-01	973	1031	751	180	931	4.460E-11	2.141E-02	10	4.551E-02
EMU	5.036E-01	3.160E-01	893	923	726	186	912	4.460E-11	4.491E-02	13	8.384E-02
AFT1	4.302E-01	3.144E-01	760	533	531	1[illegible]	695	8.503E-07	2.554E-02	12	6.287E-02
AFT2	1.205E+00	7.096E-01	2100	2434	1885	947	2832	2.185E-07	5.228E-02	14	1.019E-01
TVC	1.341E+00	9.100E-01	2411	2659	2455	885	3340	2.435E-06	5.129E-02	14	1.524E-01
IEA	7.209E-01	2.993E-01	1276	1326	1056	231	1287	7.644E-07	1.441E-02	11	3.307E-02
ASA	5.150E-01	4.021E-01	948	935	715	182	897	4.503E-07	4.419E-02	14	1.027E-01
VSWR	3.942E-02	1.675E-02	71	81	53	157	210	3.105E-06	6.325E-03	8	1.542E-02
VAX1	3.226E+00	1.739E+00	3994	6233	1854	7232	9086	1.566E-07	5.129E-02	15	2.162E-01
VAX2	7.474E-01	1.086E+00	1063	1481	199	714	913	2.180E-07	4.219E-02	15	1.350E-01
TOT	1.024E+01	6.581E+00	16125	7653	11445	11214	22659	4.460E-11	5.228E-02	15	2.162E-01

AVERAGE BUSBUSY: 0.000464 USAGE: 0.000470 IDLE: 0.000399
TOTAL BUSBUSY: 10.968316 USAGE: 10.644032 IDLE: 9.031684

S = SIMULATED THROUGHPUT: 53.220159
G = OFFERED LOAD AS A % OF BUS CAPACITY: 58.293312 (89.904480) TOTAL OFFERED DATA: 54.022240
E = EFFICIENCY (OFF LOAD): 91.297196 (59.196337) EFFICIENCY: (OFF DATA): 98.515276
T = THEORETICAL THROUGHPUT: 36.626137

TVC-AFT COMM: 150 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 662690 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	650288	410800
2	1	1688416	387472
3	1	2042608	404192
4	1	1939072	333152
5	1	2916944	417904
6	1	3059312	378176
7	1	3731888	49392
8	1	3638848	388448
9	1	4236112	49392
10	1	3778624	44128
11	1	4274752	54720
12	1	4302224	446144
13	1	4923312	56880
14	1	4310560	75664
15	1	5301520	447792
16	1	4901888	96048
17	1	5182736	45280
18	1	4973504	97752
19	1	5298928	50080
20	1	4984432	91680
7	2	531936	398144
8	2	531072	2160
9	2	531504	11984
10	2	484768	11120
11	2	555520	11120
12	2	508352	11552
13	2	519280	1728
14	2	496560	10688
15	2	520576	11552
16	2	496560	11120
17	2	567312	11552
18	2	460320	10256
19	2	531072	2160
20	2	508352	10688

SIMULATION DESCRIPTION: SIM RUN 9
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX	TX: # PKTS,	PKT SZ(B)	CONT RESP: # PKTS,	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	75	0.007	80
TVC	1.00	1	10	1500	10	1500	75	0.006	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.100	0.300	10	1	10	80
FWD2	0.150	0.250	10	1	15	80
DFI	0.150	0.200	10	1	15	80
AFT1	0.100	0.250	10	1	20	80
AFT2	0.050	0.200	10	1	15	80
TVC	0.100	0.250	10	1	10	80
IEA	0.080	0.200	10	1	20	80
ASA	0.150	0.300	10	1	15	80
VSWR	0.250	0.500	10	1	10	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
EMU	0.100	0.300	6	1	15	80

CONT	DIST (FT)	RESP 1:	2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	5	80	2.0E-03	7	0.040	1500	0.240	1500	0.500	1500	0.750	1500	0.200	1500	0.100	1500	0.160	1500
FWD2	1465.0	1	6	80	2.0E-03	7	0.041	1500	0.241	1500	0.510	1500	0.751	1500	0.210	1500	0.110	1500	0.161	1500
DFI	1455.0	1	7	80	2.0E-03	5	0.050	1500	0.070	1500	0.090	1500	0.753	1500	0.088	1500				
EMU	1405.0	1	5	80	2.0E-03	7	0.042	1500	0.242	1500	0.520	1500	0.752	1500	0.220	1500	0.120	1500	0.162	1500
AFT1	20.0	1	5	80	2.0E-03	5	0.243	1500	0.043	1500	0.080	1500	0.120	1500	0.085	1500				
AFT2	30.0	1	4	80	2.0E-03	5	0.244	1500	0.044	1500	0.081	1500	0.121	1500	0.086	1500				
TVC	10.0	1	4	80	2.0E-03	4	0.020	1500	0.045	1500	0.050	1500	0.122	1500						
IEA	1485.0	1	6	80	2.0E-03	5	0.046	1500	0.051	1500	0.060	1500	0.754	1500	0.089	1500				
ASA	1495.0	1	9	80	2.0E-03	4	0.470	1500	0.052	1500	0.066	1500	0.123	1500						
VSWR	1475.0	1	4	80	2.0E-03	6	0.047	1500	0.053	1500	0.062	1500	0.124	1500	0.087	1500	0.145	1500		

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMJM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	8.076E-01	1.420E+00	1350	1425	876	197	1073	2.091E-07	5.227E-02	16	2.112E-01
FWD2	3.307E-01	5.968E-01	541	583	282	134	416	7.428E-07	4.504E-02	16	1.545E-01
DFI	6.624E-01	8.387E-01	1149	1234	706	176	882	6.674E-07	5.217E-02	14	1.627E-01
EMU	5.995E-01	8.239E-01	1066	1102	701	181	882	1.209E-06	4.504E-02	16	1.823E-01
AFT1	5.271E-01	7.479E-01	834	939	506	163	669	2.127E-06	4.492E-02	16	1.520E-01
AFT2	1.501E+00	1.242E+00	2419	2831	1859	931	2790	1.550E-11	5.051E-02	14	1.571E-01
TVC	1.673E+00	1.642E+00	2805	3336	2376	871	3247	8.234E-07	5.051E-02	16	1.304E-01
IEA	8.909E-01	8.384E-01	1501	1592	1023	223	1246	1.110E-06	5.228E-02	14	1.388E-01
ASA	6.453E-01	4.661E-01	1092	1109	707	177	884	1.028E-10	2.951E-02	16	7.782E-02
VSWR	9.195E-01	1.121E+00	1672	1719	1407	155	1562	9.061E-07	5.051E-02	16	1.883E-01
VAX1	3.790E+00	3.367E+00	4675	7196	1823	8334	10157	6.204E-07	5.129E-02	16	1.292E-01
VAX2	7.150E-01	1.423E+00	1027	1443	194	689	883	2.435E-03	5.227E-02	16	1.675E-01
TOT	1.306E+01	1.453E+01	20131	9236	12460	12231	24691	1.550E-11	5.228E-02	16	2.112E-01

AVERAGE BUSBUSY: 0.000501 USAGE: 0.000486 IDLE: 0.000309
TOTAL BUSBUSY: 12.370635 USAGE: 12.005072 IDLE: 7.629365

S = SIMULATED THROUGHPUT: 60.025359
G = OFFERED LOAD AS A % OF BUS CAPACITY: 66.312464 (107.259200)
E = EFFICIENCY (OFF LOAD): 90.518969 (55.962900)
T = THEORETICAL THROUGHPUT: 39.872215

TOTAL OFFERED DATA: 61.283120
EFFICIENCY: (OFF DATA): 97.947622

TVC-AFT COMM: 150 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 730494 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	650288	410800
2	1	1688416	387472
3	1	2573248	404624
4	1	2693328	347696
5	1	3859872	416608
6	1	3708304	374576
7	1	4601040	21312
8	1	4335008	407568
9	1	5136432	68576
10	1	4697536	32608
11	1	5089264	63680
12	1	4839904	459104
13	1	5703744	56976
14	1	5229040	49824
15	1	5622064	494624
16	1	5598480	67568
17	1	5843952	56272
18	1	5537792	69904
19	1	5737824	61536
20	1	5278800	84096
7	2	496560	398576
8	2	507488	1728
9	2	555520	11120
10	2	544160	11552
11	2	543728	11552
12	2	519712	11120
13	2	531504	2160
14	2	437168	10256
15	2	531936	10688
16	2	543728	11552
17	2	507488	2592
18	2	449392	10688
19	2	496560	10688
20	2	554224	2160

SIMULATION DESCRIPTION: SIM RUN 10
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX TX:	# PKTS,	PKT SZ(B)	CONT RESP: # PKTS,	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	75	0.007	80
TVC	1.00	1	10	1500	10	1500	75	0.006	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.050	0.150	10	1	10	80
FWD2	0.075	0.150	10	1	15	80
DFI	0.075	0.200	10	1	15	80
AFT1	0.050	0.150	10	1	20	80
AFT2	0.030	0.100	10	1	15	80
TVC	0.050	0.200	10	1	10	80
IEA	0.040	0.150	10	1	20	80
ASA	0.075	0.200	10	1	15	80
VSWR	0.080	0.200	10	1	10	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
EMU	0.050	0.100	6	1	15	80

CONT	DIST (FT)	RESP 1:	2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	5	80	2.0E-03	7	0.040	1500	0.240	1500	0.500	1500	0.750	1500	0.200	1500	0.100	1500	0.160	1500
FWD2	1465.0	1	6	80	2.0E-03	7	0.041	1500	0.241	1500	0.510	1500	0.751	1500	0.210	1500	0.110	1500	0.161	1500
DFI	1455.0	1	7	80	2.0E-03	7	0.050	1500	0.070	1500	0.090	1500	0.753	1500	0.088	1500	0.130	1500	0.163	1500
EMU	1405.0	1	5	80	2.0E-03	7	0.042	1500	0.242	1500	0.520	1500	0.752	1500	0.220	1500	0.120	1500	0.162	1500
AFT1	20.0	1	5	80	2.0E-03	7	0.243	1500	0.043	1500	0.080	1500	0.120	1500	0.085	1500	0.140	1500	0.164	1500
AFT2	30.0	1	4	80	2.0E-03	7	0.244	1500	0.044	1500	0.081	1500	0.121	1500	0.086	1500	0.150	1500	0.165	1500
TVC	10.0	1	4	80	2.0E-03	7	0.020	1500	0.045	1500	0.050	1500	0.122	1500	0.091	1500	0.160	1500	0.166	1500
IEA	1485.0	1	6	80	2.0E-03	7	0.046	1500	0.051	1500	0.060	1500	0.754	1500	0.089	1500	0.170	1500	0.167	1500
ASA	1495.0	1	9	80	2.0E-03	7	0.470	1500	0.052	1500	0.066	1500	0.123	1500	0.092	1500	0.171	1500	0.168	1500
VSWR	1475.0	1	4	80	2.0E-03	7	0.047	1500	0.053	1500	0.062	1500	0.124	1500	0.087	1500	0.145	1500	0.168	1500

ORIGINAL PAGE IS OF POOR QUALITY

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	9.321E-01	1.391E+00	1539	1715	902	252	1154	5.765E-08	5.129E-02	16	1.586E-01
FWD2	3.616E-01	9.538E-01	531	616	249	141	390	1.092E-06	4.653E-02	16	1.684E-01
DFI	9.048E-01	2.046E+00	1436	1648	815	186	1001	2.180E-07	4.504E-02	16	1.928E-01
EMU	8.241E-01	7.246E-01	1240	1375	750	269	1019	3.157E-07	5.227E-02	16	1.580E-01
AFT1	7.328E-01	8.395E-01	1157	1327	615	181	796	7.026E-07	5.227E-02	16	1.503E-01
AFT2	1.733E+00	3.770E+00	2806	3476	1879	882	2761	1.166E-08	5.227E-02	16	1.913E-01
TVC	2.017E+00	3.088E+00	3316	4083	2543	778	3321	1.958E-03	5.227E-02	16	1.941E-01
IEA	1.261E+00	2.330E+00	2016	2248	1107	251	1358	2.122E-07	5.228E-02	16	1.942E-01
ASA	1.051E+00	2.062E+00	1663	1831	987	209	1196	3.486E-08	5.228E-02	16	2.393E-01
VSWR	1.360E+00	2.222E+00	2231	2401	1466	230	1696	3.061E-07	5.216E-02	16	2.089E-01
VAX1	3.959E+00	5.973E+00	4962	7577	2163	9409	11572	1.912E-07	5.227E-02	16	1.527E-01
VAX2	1.043E+00	1.998E+00	1371	1825	281	737	1018	8.326E-08	5.227E-02	16	1.440E-01
TOT	1.618E+01	2.740E+01	24268	11034	13757	13525	27282	1.166E-08	5.228E-02	16	2.393E-01

AVERAGE BUSBUSY: 0.000503 USAGE: 0.000488 IDLE: 0.000230
TOTAL BUSBUSY: 13.717772 USAGE: 13.302272 IDLE: 6.282228

S = SIMULATED THROUGHPUT: 66.511359
G = OFFERED LOAD AS A % OF BUS CAPACITY: 74.222592 (126.673520) TOTAL OFFERED DATA: 68.330160
E = EFFICIENCY (OFF LOAD): 89.610665 (52.506126) EFFICIENCY: (OFF DATA): 97.338216
T = THEORETICAL THROUGHPUT: 42.602163

TVC-AFT COMM: 150 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 801516 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	745056	412528
2	1	2009824	408848
3	1	3012576	420896
4	1	3189024	386288
5	1	4488304	444256
6	1	4585232	386160
7	1	5173104	75488
8	1	5244288	472592
9	1	5598912	81248
10	1	5559648	27968
11	1	5810736	87728
12	1	5577920	514960
13	1	6256240	63360
14	1	6308592	105120
15	1	5849136	503840
16	1	6198144	71456
17	1	6354464	94528
18	1	6412128	91872
19	1	6025584	124912
20	1	5761368	32233
7	2	484768	399440
8	2	567744	13280
9	2	545024	23104
10	2	545024	22672
11	2	533664	23536
12	2	544160	13280
13	2	533232	23536
14	2	427104	23104
15	2	569040	23104
16	2	544160	13712
17	2	521872	4320
18	2	521872	22672
19	2	510080	23536
20	2	544160	13280

SIMULATION DESCRIPTION: SIM RUN 11
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX TX: # PKTS,	PKT SZ(B)	CONT RESP: # PKTS,	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	75	0.007	80
TVC	1.00	1	10	1500	10	1500	75	0.006	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0
SPC1	0.55	3	10	1500	10	1500	0	0.000	0
SPC2	1.40	3	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.050	0.150	10	1	10	80
FWD2	0.075	0.150	10	1	15	80
DFI	0.075	0.200	10	1	15	80
AFT1	0.050	0.150	10	1	20	80
AFT2	0.030	0.100	10	1	15	80
TVC	0.050	0.200	10	1	10	80
IEA	0.040	0.150	10	1	20	80
ASA	0.075	0.200	10	1	15	80
VSWR	0.080	0.200	10	1	10	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
EMU	0.050	0.100	6	1	15	80

VAX NO: 3 DISTANCE (FT): 700.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
SPC1	0.100	0.300	10	1	15	80
SPC2	0.100	0.250	10	1	10	80

CONT	DIST (FT)	RESP 1:	RESP 2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	5	80	2.0E-03	7	0.040	1500	0.240	1500	0.500	1500	0.750	1500	0.200	1500	0.100	1500	0.160	1500
FWD2	1465.0	1	6	80	2.0E-03	7	0.041	1500	0.241	1500	0.510	1500	0.751	1500	0.210	1500	0.110	1500	0.161	1500
DFI	1455.0	1	7	80	2.0E-03	7	0.050	1500	0.070	1500	0.090	1500	0.753	1500	0.088	1500	0.130	1500	0.163	1500
EMU	1405.0	1	5	80	2.0E-03	7	0.042	1500	0.242	1500	0.520	1500	0.752	1500	0.220	1500	0.120	1500	0.162	1500

AFT1	20.0	1	5	80	2.0E-03	7	0.243	1500	0.043	1500	0.080	1500	0.120	1500	0.085	1500	0.140	1500	0.164	1500
AFT2	30.0	1	4	80	2.0E-03	7	0.244	1500	0.044	1500	0.081	1500	0.121	1500	0.086	1500	0.150	1500	0.165	1500
TVC	10.0	1	4	80	2.0E-03	7	0.020	1500	0.045	1500	0.050	1500	0.122	1500	0.091	1500	0.160	1500	0.166	1500
IEA	1485.0	1	6	80	2.0E-03	7	0.046	1500	0.051	1500	0.060	1500	0.754	1500	0.089	1500	0.170	1500	0.167	1500
ASA	1495.0	1	9	80	2.0E-03	7	0.470	1500	0.052	1500	0.066	1500	0.123	1500	0.092	1500	0.171	1500	0.168	1500
VSWR	1475.0	1	4	80	2.0E-03	7	0.047	1500	0.053	1500	0.062	1500	0.124	1500	0.087	1500	0.145	1500	0.168	1500
SPC1	800.0	1	10	80	2.0E-03	6	0.245	1500	0.054	1500	0.063	1500	0.125	1500	0.093	1500	0.131	1500		
SPC2	850.0	1	5	80	2.0E-03	6	0.151	1500	0.152	1500	0.153	1500	0.073	1500	0.074	1500	0.172	1500		

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	1.069E+00	2.195E+00	1743	1917	867	237	1104	5.854E-07	5.227E-02	16	2.197E-01
FWD2	3.490E-01	1.182E+00	545	632	204	129	333	5.592E-07	4.123E-02	16	1.542E-01
DFI	9.415E-01	2.319E+00	1448	1689	767	172	939	1.194E-07	5.216E-02	16	1.880E-01
EMU	8.782E-01	3.254E+00	1393	1589	650	238	888	4.405E-07	5.216E-02	16	1.975E-01
AFT1	7.952E-01	2.452E+00	1181	1399	518	164	682	3.401E-07	5.130E-02	16	2.007E-01
AFT2	1.970E+00	5.521E+00	2942	3609	1661	710	2371	1.463E-08	5.227E-02	16	1.940E-01
TVC	2.008E+00	7.272E+00	3258	4089	2021	617	2638	2.180E-07	6.264E-02	16	2.271E-01
IEA	1.362E+00	4.039E+00	2060	2363	995	223	1218	1.807E-07	5.227E-02	16	2.233E-01
ASA	1.243E+00	3.639E+00	1886	2116	914	196	1110	2.179E-07	5.216E-02	16	2.151E-01
VSWR	1.472E+00	3.303E+00	2315	2594	1384	213	1597	2.204E-07	5.227E-02	16	2.124E-01
SPC1	1.164E+00	3.373E+00	1836	2002	1108	198	1306	2.435E-08	5.217E-02	16	2.113E-01
SPC2	1.104E+00	2.230E+00	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	16	2.082E-01
VAX1	3.661E+00	8.137E+00	4777	7529	1999	8442	10441	3.005E-07	5.228E-02	16	2.573E-01
VAX2	9.080E-01	4.209E+00	1268	1621	250	631	881	4.788E-07	5.216E-02	16	2.279E-01
SPV1	1.605E+00	6.367E+00	2193	3211	435	2008	2443	6.242E-08	5.227E-02	16	2.316E-01
TOT	2.053E+01	5.949E+01	30544	13135	14707	14381	29088	1.550E-11	6.264E-02	16	2.[illegible]-01

AVERAGE BUSBUSY: 0.000518 USAGE: 0.000502 IDLE: 0.000169
TOTAL BUSBUSY: 15.074992 USAGE: 14.614160 IDLE: 4.925006

S = SIMULATED THROUGHPUT: 73.070793
G = OFFERED LOAD AS A % OF BUS CAPACITY: 92.652048 (147.031360)
E = EFFICIENCY (OFF LOAD): [illegible].104318 (49.697424)
T = THEORETICAL THROUGHPUT: 45.310976

TOTAL OFFERED DATA: 75.240800
EFFICIENCY: (OFF DATA): 97.115924

ORIGINAL PAGE IS OF POOR QUALITY

TVC-AFT COMM: 150 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 752653 BYTES IN 1 SECOND
VAX: 3 RECEIVED 166724 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	745056	412096
2	1	1998032	410576
3	1	2953616	420032
4	1	3307376	412736
5	1	4088672	431600
6	1	4301792	454320
7	1	4367792	65340
8	1	5148224	403296
9	1	4962576	107056
10	1	5229904	56880
11	1	5136864	56976
12	1	4959984	449952
13	1	5588416	113360
14	1	5386224	78080
15	1	5464016	393536
16	1	5681024	104432
17	1	5869264	[illegible]
18	1	5620768	81328
19	1	6021264	58704
20	1	5678432	78736
7	2	366848	395984
8	2	485632	14576
9	2	521440	22240
10	2	451120	14144
11	2	474704	14144
12	2	461184	12416
13	2	496992	12848
14	2	567744	12416
15	2	533664	22672
16	2	461184	11552
17	2	402224	11934
18	2	485632	22672
19	2	355056	2592
20	2	462480	21376
1	3	330176	388320
2	3	1145120	400304
3	3	1333792	11376
4	3	1204944	9648
5	3	1297552	6048
6	3	1227664	10944
7	3	1322000	13712
8	3	1215872	9648
9	3	1227232	4320
10	3	1109744	19040
11	3	1214576	1296

12	3	1191856	4320
13	3	1086160	18608
14	3	1180496	4320
15	3	1003184	9648
16	3	932864	12848
17	3	1049920	9216
18	3	1121536	11552
19	3	931568	3455
20	3	1168704	8784

SIMULATION DESCRIPTION: SIM RUN 12 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597
SIMULATION RUN TIME (S): 2.0E+01

CONT	START TIME(S)	VAX	TX: # PKTS,	PKT SZ(B)	CONT RESP: # PKTS,	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	75	0.007	80
TVC	1.00	1	10	1500	10	1500	75	0.006	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0
SPC1	0.55	3	10	1500	10	1500	0	0.000	0
SPC2	1.40	3	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.050	0.150	10	1	10	80
FWD2	0.075	0.150	10	1	15	80
DFI	0.075	0.200	10	1	15	80
AFT1	0.050	0.150	10	1	20	80
AFT2	0.030	0.100	10	1	15	80
TVC	0.050	0.200	10	1	10	80
IEA	0.040	0.150	10	1	20	80
ASA	0.075	0.200	10	1	15	80
VSWR	0.080	0.200	10	1	10	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
EMU	0.050	0.100	6	1	15	80

VAX NO: 3 DISTANCE (FT): 700.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
SPC1	0.050	0.250	10	1	15	80
SPC2	0.040	0.120	10	1	10	80

CONT	DIST (FT)	RESP 1:	2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	5	80	2.0E-03	7	0.040	1500	0.240	1500	0.500	1500	0.750	1500	0.200	1500	0.100	1500	0.160	1500
FWD2	1465.0	1	6	30	2.0E-03	7	0.041	1500	0.241	1500	0.510	1500	0.751	1500	0.210	1500	0.110	1500	0.161	1500
DFI	1455.0	1	7	80	2.0E-03	7	0.050	1500	0.070	1500	0.090	1500	0.753	1500	0.088	1500	0.130	1500	0.163	1500
EMU	1405.0	1	5	90	2.0E-03	7	0.042	1500	0.242	1500	0.520	1500	0.752	1500	0.220	1500	0.120	1500	0.162	1500

AFT1	20.0	1	5	80	2.0E-03	7	0.243	1500	0.043	1500	0.080	1500	0.120	1500	0.085	1500	0.140	1500	0.164	1500
AFT2	30.0	1	4	80	2.0E-03	7	0.244	1500	0.044	1500	0.081	1500	0.121	1500	0.086	1500	0.150	1500	0.165	1500
TVC	10.0	1	4	80	2.0E-03	7	0.020	1500	0.045	1500	0.050	1500	0.122	1500	0.091	1500	0.160	1500	0.166	1500
IEA	1485.0	1	6	80	2.0E-03	7	0.046	1500	0.051	1500	0.060	1500	0.754	1500	0.089	1500	0.170	1500	0.167	1500
ASA	1495.0	1	9	80	2.0E-03	7	0.470	1500	0.052	1500	0.066	1500	0.123	1500	0.092	1500	0.171	1500	0.168	1500
VSWR	1475.0	1	4	80	2.0E-03	7	0.047	1500	0.053	1500	0.062	1500	0.124	1500	0.087	1500	0.145	1500	0.168	1500
SPC1	800.0	1	10	80	2.0E-03	7	0.245	1500	0.054	1500	0.063	1500	0.125	1500	0.093	1500	0.146	1500	0.172	1500
SPC2	850.0	1	5	80	2.0E-03	7	0.151	1500	0.152	1500	0.153	1500	0.073	1500	0.074	1500	0.075	1500	0.076	1500

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	1.167E+00	3.390E+00	1818	2031	826	223	1049	3.202E-07	5.228E-02	16	2.162E-01
FWD2	2.943E-01	6.512E-01	436	507	197	123	320	5.132E-07	4.680E-02	16	1.714E-01
DFI	1.018E+00	2.951E+00	1519	1752	741	163	904	1.076E-07	5.227E-02	16	1.909E-01
EMU	9.423E-01	2.252E+00	1391	1558	683	241	924	2.167E-07	5.129E-02	16	2.058E-01
AFT1	6.596E-01	2.649E+00	953	1138	441	152	593	5.625E-08	5.129E-02	16	1.999E-01
AFT2	1.968E+00	5.651E+00	2903	3614	1602	660	2262	4.460E-11	5.227E-02	16	2.450E-01
TVC	2.092E+00	7.141E+00	3239	4010	1964	580	2544	3.722E-07	5.227E-02	16	2.226E-01
IEA	1.298E+00	4.363E+00	1919	2250	959	209	1168	1.188E-06	5.227E-02	16	2.291E-01
ASA	1.160E+00	3.681E+00	1766	2014	889	190	1079	2.033E-07	5.227E-02	16	2.126E-01
VSWR	1.551E+00	3.252E+00	2418	2723	1386	202	1568	6.027E-08	5.216E-02	16	2.485E-01
SPC1	1.420E+00	3.526E+00	2170	2382	1206	227	1433	1.971E-07	5.227E-02	16	1.844E-01
SPC2	1.446E+00	3.445E+00	2207	2525	1218	276	1494	1.550E-11	5.435E-02	16	2.077E-01
VAX1	3.467E+00	8.998E+00	4530	6902	1906	8124	10030	2.438E-08	6.009E-02	16	2.559E-01
VAX2	1.023E+00	4.232E+00	1417	1819	253	671	924	1.405E-07	5.227E-02	16	2.280E-01
SPV1	1.825E+00	8.641E+00	2539	3522	537	2390	2927	1.478E-07	5.227E-02	16	2.241E-01
TOT	2.133E+01	6.482E+01	31230	13075	14808	14431	29239	1.550E-11	6.009E-02	16	2.559E-01

AVERAGE BUSBUSY: 0.000522 USAGE: 0.000506 IDLE: 0.000162
TOTAL BUSBUSY: 15.268323 USAGE: 14.806992 IDLE: 4.731677

S = SIMULATED THROUGHPUT: 74.034958
G = OFFERED LOAD AS A % OF BUS CAPACITY: 83.954192 (150.574880) TOTAL OFFERED DATA: 76.341440
E = EFFICIENCY (OFF LOAD): 88.184945 (49.168200) EFFICIENCY: (OFF DATA): 96.978729
T = THEORETICAL THROUGHPUT: 45.638640

TVC-AFT COMM: 150 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 722962 BYTES IN 1 SECOND
VAX: 3 RECEIVED 205210 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	745056	412096
2	1	1925984	401184
3	1	2942688	429856
4	1	3224832	350576
5	1	4206160	433936
6	1	4416688	446208
7	1	4310560	48880
8	1	4581776	430960
9	1	4902752	89200
10	1	5098896	38048
11	1	5552176	84880
12	1	4733344	388512
13	1	5099328	110880
14	1	5493216	70560
15	1	4948192	423776
16	1	5338192	92160
17	1	5441728	72064
18	1	5783696	51696
19	1	5769744	25952
20	1	5857472	95184
7	2	378640	386784
8	2	544592	13712
9	2	521440	22672
10	2	520576	12416
11	2	438896	22240
12	2	472976	13280
13	2	498288	22672
14	2	449824	13280
15	2	555520	11984
16	2	569040	23536
17	2	532368	12848
18	2	521440	21808
19	2	402656	12416
20	2	496992	12416
1	3	318384	388320
2	3	1251680	406928
3	3	1641680	24224
4	3	1463936	6480
5	3	1582720	22496
6	3	1522464	12672
7	3	1382256	22496
8	3	1346448	16736
9	3	1499312	22496
10	3	1393616	11808
11	3	1439056	4320

12	3	1334656	21200
13	3	1311072	20336
14	3	1179632	4752
15	3	1393616	21200
16	3	1369168	10944
17	3	1392752	15008
18	3	1298416	10080
19	3	1345584	21632
20	3	1074368	5184

ORIGINAL PAGE IS
OF POOR QUALITY

SIMULATION DESCRIPTION: SIM RUN 13
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX	TX: # PKTS	PKT SZ(B)	CONT RESP: # PKTS	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	14.10	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	11.10	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	75	0.007	80
TVC	1.00	1	10	1500	10	1500	75	0.006	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0
SPC1	0.55	3	10	1500	10	1500	0	0.000	0
SPC2	1.40	3	10	1500	10	1500	0	0.000	0
SPC3	2.40	4	10	1500	10	1500	0	0.000	0
SPC4	3.60	5	10	1500	10	1500	0	0.000	0
SPC5	3.90	5	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S)	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.050	0.150	10	1	10	80
FWD2	0.075	0.150	10	1	15	80
DFI	0.075	0.200	10	1	15	80
AFT1	0.050	0.150	10	1	20	80
AFT2	0.030	0.100	10	1	15	80
TVC	0.050	0.200	10	1	10	80
IEA	0.040	0.150	10	1	20	80
ASA	0.075	0.200	10	1	15	80
VSWR	0.030	0.200	10	1	10	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S)	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
EMU	0.050	0.100	6	1	15	80

VAX NO: 3 DISTANCE (FT): 700.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S)	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
SPC1	0.050	0.250	10	1	15	80
SPC2	0.040	0.120	10	1	10	80

VAX NO: 4 DISTANCE (FT): 750.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S)	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
SPC3	0.050	0.250	10	1	15	80

VAX NO: 5 DISTANCE (FT): 650.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S)	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
SPC4	0.060	0.260	10	1	10	80
SPC5	0.600	2.600	10	1	10	80

CONT	DIST (FT)	RESP 1:	RESP 2:	PKT SZ(B)	AVG DLY(S)	NUMBER 8614	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	5	80	2.0E-03	7	0.040	1500	0.240	1500	0.500	1500	0.750	1500	0.200	1500	0.100	1500	0.160	1500
FWD2	1465.0	1	6	80	2.0E-03	7	0.041	1500	0.241	1500	0.510	1500	0.751	1500	0.210	1500	0.110	1500	0.161	1500
DFI	1455.0	1	7	80	2.0E-03	7	0.053	1500	0.070	1500	0.090	1500	0.753	1500	0.088	1500	0.130	1500	0.163	1500
EMU	1405.0	1	5	80	2.0E-03	7	0.042	1500	0.242	1500	0.520	1500	0.752	1500	0.220	1500	0.120	1500	0.162	1500
AFT1	20.0	1	5	80	2.0E-03	7	0.243	1500	0.043	1500	0.080	1500	0.120	1500	0.085	1500	0.140	1500	0.164	1500
AFT2	30.0	1	4	80	2.0E-03	7	0.244	1500	0.044	1500	0.081	1500	0.121	1500	0.086	1500	0.150	1500	0.165	1500
TVC	10.0	1	4	80	2.0E-03	7	0.020	1500	0.045	1500	0.050	1500	0.122	1500	0.091	1500	0.160	1500	0.166	1500
IEA	1485.0	1	6	80	2.0E-03	7	0.046	1500	0.051	1500	0.060	1500	0.754	1500	0.089	1500	0.170	1500	0.167	1500
ASA	1495.0	1	9	80	2.0E-03	7	0.470	1500	0.052	1500	0.066	1500	0.123	1500	0.092	1500	0.171	1500	0.168	1500
VSWR	1475.0	1	4	80	2.0E-03	7	0.047	1500	0.053	1500	0.062	1500	0.124	1500	0.087	1500	0.145	1500	0.168	1500
SPC1	800.0	1	10	80	2.0E-03	7	0.245	1500	0.054	1500	0.063	1500	0.125	1500	0.093	1500	0.146	1500	0.172	1500
SPC2	850.0	1	5	80	2.0E-03	7	0.151	1500	0.152	1500	0.153	1500	0.073	1500	0.074	1500	0.075	1500	0.076	1500
SPC3	900.0	1	5	80	2.0E-03	7	0.131	1500	0.132	1500	0.133	1500	0.134	1500	0.135	1500	0.136	1500	0.137	1500
SPC4	950.0	1	5	80	2.0E-03	7	0.231	1500	0.232	1500	0.234	1500	0.233	1500	0.235	1500	0.236	1500	0.237	1500
SPC5	1000.0	1	9	80	2.0E-03	7	0.251	1500	0.252	1500	0.253	1500	0.254	1500	0.255	1500	0.256	1500	0.257	1500

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	1.209E+00	5.523E+00	1935	2203	737	206	943	2.435E-08	5.216E-02	16	2.099E-01
FWD2	2.420E-01	7.273E-01	336	429	119	121	240	1.358E-06	4.419E-02	16	1.535E-01
DFI	8.923E-01	4.339E+00	1381	1648	614	151	765	2.940E-07	5.228E-02	16	2.414E-01
EMU	9.913E-01	3.582E+00	1456	1693	593	229	822	1.077E-06	5.227E-02	16	2.422E-01
AFT1	6.189E-01	2.395E+00	892	1095	403	144	547	1.748E-06	5.216E-02	16	2.162E-01
AFT2	1.794E+00	8.449E+00	2719	3402	1318	515	1833	1.595E-07	5.227E-02	16	2.450E-01
TVC	2.110E+00	8.501E+00	3294	4100	1699	466	2165	1.595E-07	5.227E-02	16	2.634E-01
IEA	1.230E+00	5.987E+00	1871	2186	833	186	1019	1.808E-07	5.933E-02	16	2.826E-01
ASA	1.369E+00	4.913E+00	2020	2318	839	179	1018	2.307E-08	5.227E-02	16	2.230E-01
VSWR	1.652E+00	6.081E+00	2568	2925	1188	194	1382	9.461E-08	5.227E-02	16	2.424E-01
SPC1	1.477E+00	4.359E+00	2278	2571	1177	219	1396	3.372E-08	5.216E-02	16	1.749E-01
SPC2	1.358E+00	4.669E+00	2136	2560	1145	262	1407	1.550E-11	5.228E-02	16	2.083E-01
SPC3	1.186E+00	4.625E+00	1617	2152	825	222	1047	2.820E-07	5.227E-02	16	1.978E-01
SPC4	7.956E-01	3.121E+00	1172	1401	475	202	677	8.057E-07	5.216E-02	16	2.112E-01
SPC5	7.153E-01	2.995E+00	1088	1247	420	128	543	1.550E-11	5.227E-02	16	2.116E-01
VAX1	3.333E+00	1.035E+01	4542	6859	1773	7101	8874	6.245E-07	5.227E-02	16	2.286E-01
VAX2	1.063E+00	4.752E+00	1442	1800	241	578	819	2.230E-07	5.227E-02	16	1.991E-01
SPV1	1.806E+00	9.111E+00	2543	3616	515	2287	2802	2.304E-08	5.227E-02	16	2.980E-01
SPV2	1.042E+00	5.580E+00	1473	1896	238	805	1043	1.754E-08	5.227E-02	16	2.109E-01
SPV3	1.080E+00	5.540E+00	1579	2061	358	859	1217	2.165E-11	5.217E-02	16	2.352E-01
TOT	2.596E+01	1.056E+02	38542	15202	15510	15054	30564	1.550E-11	5.933E-02	16	2.980E-01

AVERAGE	BUSBUSY:	0.000524	USAGE:	0.000507	IDLE:	0.000130
TOTAL	BUSBUSY:	16.012091	USAGE:	15.509728	IDLE:	3.987909

S = SIMULATED THROUGHPUT: 77.548638
G = OFFERED LOAD AS A % OF BUS CAPACITY: 89.878368 (166.279520)
E = EFFICIENCY (OFF LOAD): 86.281760 (46.637516)
T = THEORETICAL THROUGHPUT: 47.334706

TOTAL OFFERED DATA: 80.628960
EFFICIENCY: (OFF DATA): 96.179633

TVC-AFT COMM: 150 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 665746 BYTES IN 1 SECOND
VAX: 3 RECEIVED 199314 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	745056	412096
2	1	1925984	401134
3	1	2836560	429424
4	1	2941824	424784
5	1	3402144	423056
6	1	3935808	454080
7	1	3779488	56016
8	1	4039776	408944
9	1	4083056	43952
10	1	4299632	72176
11	1	4685312	23216
12	1	4521088	299328
13	1	4651232	161056
14	1	4461264	60304
15	1	4815456	177712
16	1	4710192	254464
17	1	4838176	51872
18	1	4688336	112704
19	1	5325968	51952
20	1	5028576	47232
7	2	235840	370176
8	2	260720	26672
9	2	532368	13712
10	2	545024	22240
11	2	496992	14576
12	2	532368	21376
13	2	403520	13280
14	2	508352	11552
15	2	449824	12848
16	2	462912	23104
17	2	449824	13712
18	2	379072	21376
19	2	379936	12848
20	2	426240	13280
1	3	318384	388320
2	3	1251680	406928
3	3	1594512	22928
4	3	1558272	13536
5	3	1476160	21632
6	3	1309776	13104
7	3	1299280	15872
8	3	1251248	9648
9	3	1381392	22496
10	3	1393184	12240
11	3	1439920	13280

12	3	1369168	11376
13	3	1239024	11552
14	3	1475296	9648
15	3	1074368	21200
16	3	1239456	6048
17	3	1216304	19040
18	3	1370032	13968
19	3	1109312	19904
20	3	1346016	6048
3	4	318384	388320
4	4	555952	11120
5	4	485200	12848
6	4	424944	2592
7	4	590896	11552
8	4	460320	11984
9	4	591328	2592
10	4	543728	12416
11	4	413152	2592
12	4	555952	11984
13	4	461184	11120
14	4	507488	1728
15	4	461184	11552
16	4	507488	2592
17	4	485200	11984
18	4	531504	12348
19	4	496128	2160
20	4	461616	11120
4	5	235840	392032
5	5	496560	367648
6	5	637632	3456
7	5	661648	13248
8	5	648992	3456
9	5	626272	9216
10	5	660784	2592
11	5	590464	9216
12	5	531504	8784
13	5	696160	3024
14	5	496128	9216
15	5	660784	2160
16	5	485200	14976
17	5	578672	3456
18	5	543296	9648
19	5	542864	2592
20	5	625840	7920

SIMULATION DESCRIPTION: SIM RUN 14
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX	TX: # PKTS,	PKT SZ(B)	CONT RESP: # PKTS,	PKT SZ(B)	AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	4.80	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	5.80	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	75	0.007	80
TVC	1.00	1	10	1500	10	1500	75	0.006	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0
SPC1	0.55	3	10	1500	10	1500	0	0.000	0
SPC2	1.40	3	10	1500	10	1500	0	0.000	0
SPC3	2.40	4	10	1500	10	1500	0	0.000	0
SPC4	3.60	5	10	1500	10	1500	0	0.000	0
SPC5	3.90	5	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.050	0.150	10	1	10	80
FWD2	0.075	0.150	10	1	15	80
DFI	0.075	0.200	10	1	15	80
AFT1	0.050	0.150	10	1	20	80
AFT2	0.030	0.100	10	1	15	80
TVC	0.050	0.200	10	1	10	80
IEA	0.040	0.150	10	1	20	80
ASA	0.075	0.200	10	1	15	80
VSWR	0.080	0.200	10	1	10	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
EMU	0.050	0.100	6	1	15	80

VAX NO: 3 DISTANCE (FT): 700.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
SPC1	0.050	0.250	10	1	15	80
SPC2	0.040	0.120	10	1	10	80

VAX NO: 4 DISTANCE (FT): 750.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %,	BLOCKED CMD: %	NO	SZ(B)
SPC3	0.050	0.150	10	1	15	80

VAX NO: 5 DISTANCE (FT): 650.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S)	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
SPC4	0.160	0.400	10	1	10	80
SPC5	0.030	0.100	10	1	10	80

CONT	DIST (FT)	RESP 1:	RESP 2:	PKT SZ(B)	AVG DLY(S)	NUMBER B614	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	5	80	2.0E-03	7	0.030	1500	0.040	1500	0.050	1500	0.060	1500	0.070	1500	0.080	1500	0.090	1500
FWD2	1465.0	1	6	80	2.0E-03	7	0.031	1500	0.041	1500	0.051	1500	0.061	1500	0.071	1500	0.081	1500	0.091	1500
DFI	1455.0	1	7	80	2.0E-03	7	0.032	1500	0.042	1500	0.052	1500	0.062	1500	0.072	1500	0.082	1500	0.092	1500
EMU	1405.0	1	5	80	2.0E-03	7	0.033	1500	0.043	1500	0.053	1500	0.630	1500	0.073	1500	0.083	1500	0.093	1500
AFT1	20.0	1	5	80	2.0E-03	7	0.034	1500	0.044	1500	0.054	1500	0.064	1500	0.074	1500	0.084	1500	0.094	1500
AFT2	30.0	1	4	80	2.0E-03	7	0.036	1500	0.046	1500	0.056	1500	0.066	1500	0.076	1500	0.086	1500	0.096	1500
TVC	10.0	1	4	80	2.0E-03	7	0.037	1500	0.047	1500	0.057	1500	0.067	1500	0.077	1500	0.087	1500	0.097	1500
IEA	1485.0	1	6	80	2.0E-03	7	0.038	1500	0.048	1500	0.058	1500	0.068	1500	0.078	1500	0.088	1500	0.098	1500
ASA	1495.0	1	9	80	2.0E-03	7	0.035	1500	0.045	1500	0.055	1500	0.065	1500	0.075	1500	0.085	1500	0.095	1500
VSWR	1475.0	1	4	80	2.0E-03	7	0.039	1500	0.049	1500	0.059	1500	0.069	1500	0.079	1500	0.089	1500	0.099	1500
SPC1	800.0	1	10	80	2.0E-03	7	0.110	1500	0.120	1500	0.130	1500	0.140	1500	0.150	1500	0.160	1500	0.170	1500
SPC2	850.0	1	5	80	2.0E-03	7	0.111	1500	0.121	1500	0.131	1500	0.141	1500	0.151	1500	0.161	1500	0.171	1500
SPC3	900.0	1	5	80	2.0E-03	7	0.113	1500	0.123	1500	0.133	1500	0.143	1500	0.153	1500	0.163	1500	0.173	1500
SPC4	950.0	1	5	80	2.0E-03	7	0.114	1500	0.124	1500	0.134	1500	0.144	1500	0.154	1500	0.164	1500	0.174	1500
SPC5	1000.0	1	3	80	2.0E-03	7	0.115	1500	0.125	1500	0.135	1500	0.145	1500	0.155	1500	0.165	1500	0.175	1500

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	1.673E+00	9.310E+00	2450	3019	1136	163	1299	2.825E-08	5.228E-02	16	2.245E-01
FWD2	1.474E+00	7.052E+00	2049	2543	862	139	1001	1.664E-07	5.227E-02	16	2.213E-01
DFI	1.950E-01	1.630E+00	312	368	35	110	145	2.234E-06	5.216E-02	16	1.898E-01
EMU	1.404E+00	6.691E+00	2046	2464	779	183	962	4.139E-07	5.216E-02	16	2.278E-01
AFT1	1.155E+00	6.018E+00	1646	2007	618	128	746	2.744E-07	5.227E-02	16	2.434E-01
AFT2	2.096E+00	9.197E+00	3074	3900	1676	425	2101	1.550E-11	5.227E-02	16	2.234E-01
TVC	1.897E+00	1.089E+01	2809	3505	1346	369	1715	3.675E-07	5.227E-02	16	2.445E-01
IEA	1.039E+00	6.834E+00	1555	1881	539	127	666	8.314E-03	5.228E-02	16	2.880E-01
ASA	1.493E+00	7.846E+00	2183	2620	901	139	1040	6.908E-07	5.228E-02	16	2.174E-01
VSWR	1.649E+00	8.240E+00	2454	2937	1190	158	1348	8.590E-08	8.495E-02	16	4.528E-01
SPC1	1.219E+00	6.310E+00	1837	2116	807	206	1013	1.102E-07	5.228E-02	16	2.888E-01
SPC2	1.333E+00	5.644E+00	2006	2284	834	258	1092	1.028E-10	5.228E-02	16	2.545E-01
SPC3	1.358E+00	5.998E+00	1973	2299	765	237	1002	3.166E-07	5.227E-02	16	2.307E-01
SPC4	1.138E+00	5.370E+00	1625	1887	634	156	790	1.903E-07	7.742E-02	16	2.222E-01
SPC5	1.196E+00	5.361E+00	1775	2055	646	211	857	4.460E-11	5.217E-02	16	3.119E-01
VAX1	2.645E+00	1.297E+01	3747	5830	1457	7712	9169	2.051E-08	5.227E-02	16	2.761E-01
VAX2	1.051E+00	7.520E+00	1445	1857	196	767	963	2.936E-07	5.227E-02	16	2.272E-01
SPV1	1.582E+00	8.741E+00	2214	3073	498	1588	2086	2.913E-07	5.227E-02	16	2.721E-01
SPV2	1.074E+00	6.684E+00	1536	2002	253	747	1000	2.143E-07	5.227E-02	16	2.707E-01
SPV3	1.385E+00	9.715E+00	2000	2636	395	1226	1621	6.255E-07	5.228E-02	16	3.427E-01
TOT	2.805E+01	1.480E+02	40736	15702	15547	15049	30616	1.550E-11	8.495E-02	16	4.528E-01

AVERAGE BUSBUSY: 0.000544 USAGE: 0.000527 IDLE: 0.000109
TOTAL BUSBUSY: 16.648416 USAGE: 16.135744 IDLE: 3.351584

S = SIMULATED THROUGHPUT: 80.678718
G = OFFERED LOAD AS A % OF BUS CAPACITY: 93.832768 (180.896480)
E = EFFICIENCY (OFF LOAD): 85.981390 (44.599385)
T = THEORETICAL THROUGHPUT: 48.409136

TOTAL OFFERED DATA: 84.588560
EFFICIENCY: (OFF DATA): 95.377813

TVC-AFT COMM: 150 MSGS NOT TRANSMITTED IN 1 SECOND
VAX: 1 RECEIVED 706100 BYTES IN 1 SECOND
VAX: 2 RECEIVED 125452 BYTES IN 1 SECOND
VAX: 3 RECEIVED 143302 BYTES IN 1 SECOND
VAX: 5 RECEIVED 138934 BYTES IN 1 SECOND

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	1028064	412528
2	1	2409456	408416
3	1	3389920	414704
4	1	3695648	406512
5	1	4096144	514480
6	1	3824928	453808
7	1	4672656	308688
8	1	4109232	401536
9	1	4348672	149456
10	1	4425024	228192
11	1	4933808	59728
12	1	5648800	0
13	1	5144336	38592
14	1	5132976	32000
15	1	5530380	6480
16	1	4919424	26672
17	1	5236944	7920
18	1	5176688	1296
19	1	5253056	80032
20	1	5450064	13712
7	2	235840	363264
8	2	106560	25488
9	2	649856	11552
10	2	1003616	11120
11	2	685664	11984
12	2	484768	11120
13	2	648992	10688
14	2	873904	2160
15	2	485200	12416
16	2	568176	21376
17	2	603120	11984
18	2	873040	11984
19	2	567312	1296
20	2	755984	11120
1	3	306592	388320
2	3	969104	407792
3	3	1146416	24224
4	3	1074368	11808
5	3	1111040	22496
6	3	991824	21200
7	3	920640	3024
8	3	1063008	12240
9	3	862976	21632

10	3	885264	10944
11	3	850752	13712
12	3	838528	10944
13	3	896624	4320
14	3	862976	19904
15	3	803152	9216
16	3	955584	5616
17	3	945520	20768
18	3	815376	10512
19	3	838096	14144
20	3	838528	9216
3	4	307024	388752
4	4	579104	12416
5	4	567744	13712
6	4	449824	11120
7	4	531072	2592
8	4	520576	13280
9	4	555520	12848
10	4	555952	12416
11	4	436736	3024
12	4	331472	11552
13	4	461616	12848
14	4	401792	3024
15	4	343264	10688
16	4	460320	13280
17	4	414448	3456
18	4	402224	11552
19	4	460320	3024
20	4	438032	11120
4	5	294800	443648
5	5	330608	307680
6	5	981328	18000
7	5	933296	10944
8	5	1111472	12672
9	5	862112	7920
10	5	873904	8352
11	5	873040	3024
12	5	804016	10512
13	5	814080	3024
14	5	852048	16704
15	5	827600	11376
16	5	897056	9648
17	5	875200	9648
18	5	874336	10080
19	5	814944	3456
20	5	826736	14976

SIMULATION DESCRIPTION: SIM RUN 15
SIMULATION RUN TIME (S): 2.0E+01 ETHERNET BUS RATE (B/S): 1.0E+07 RANDOM NUMBER SEED: 2597

CONT	START TIME(S)	VAX TX: # PKTS,	PKT SZ(B)		CONT RESP: # PKTS, PKT SZ(B)		AFT/TVC COMM: MSG/SEC	DELAY(S)	PKT SZ(B)
FWD1	3.10	1	10	1500	10	1500	0	0.000	0
FWD2	4.80	1	10	1500	10	1500	0	0.000	0
DFI	7.10	1	10	1500	10	1500	0	0.000	0
EMU	6.10	2	10	1500	10	1500	0	0.000	0
AFT1	5.80	1	10	1500	10	1500	0	0.000	0
AFT2	0.10	1	10	1500	10	1500	75	0.007	80
TVC	1.00	1	10	1500	10	1500	75	0.006	80
IEA	5.10	1	10	1500	10	1500	0	0.000	0
ASA	4.10	1	10	1500	10	1500	0	0.000	0
VSWR	2.10	1	10	1500	10	1500	0	0.000	0
SPC1	0.55	3	10	1500	10	1500	0	0.000	0
SPC2	1.40	3	10	1500	10	1500	0	0.000	0
SPC3	2.40	4	10	1500	10	1500	0	0.000	0
SPC4	3.60	5	10	1500	10	1500	0	0.000	0
SPC5	3.90	5	10	1500	10	1500	0	0.000	0

VAX NO: 1 DISTANCE (FT): 1385.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
FWD1	0.020	0.100	10	1	10	80
FWD2	0.020	0.100	10	1	15	80
DFI	0.020	0.100	10	1	15	80
AFT1	0.020	0.100	10	1	20	80
AFT2	0.020	0.100	10	1	15	80
TVC	0.020	0.100	10	1	10	80
IEA	0.020	0.100	10	1	20	80
ASA	0.020	0.100	10	1	15	80
VSWR	0.020	0.100	10	1	10	80

VAX NO: 2 DISTANCE (FT): 1395.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
EMU	0.020	0.100	6	1	15	80

VAX NO: 3 DISTANCE (FT): 700.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
SPC1	0.020	0.100	10	1	15	80
SPC2	0.020	0.100	10	1	10	80

VAX NO: 4 DISTANCE (FT): 750.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S),	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
SPC3	0.020	0.100	10	1	15	80

ORIGINAL PAGE IS OF POOR QUALITY

10

VAX NO: 5 DISTANCE (FT): 650.0 NUMBER RECEIVE (B): 90000 NUMBER TRANSMIT (B): 100000

CONT	RNG 1(S)	RNG 2(S)	SINGLE CMD: %	BLOCKED CMD: %	NO	SZ(B)
SPC4	0.020	0.100	10	1	10	80
SPC5	0.020	0.100	10	1	10	80

CONT	DIST (FT)	RESP 1:	RESP 2:	PKT SZ(B)	AVG DLY(S) 8614	NUMBER	FILL TIME(S), BUF SZ(B) 1:		2:		3:		4:		5:		6:		7:	
FWD1	1445.0	1	5	80	2.0E-03	7	0.030	1500	0.040	1500	0.050	1500	0.060	1500	0.070	1500	0.080	1500	0.090	1500
FWD2	1465.0	1	6	80	2.0E-03	7	0.031	1500	0.041	1500	0.051	1500	0.061	1500	0.071	1500	0.081	1500	0.091	1500
DFI	1455.0	1	7	80	2.0E-03	7	0.032	1500	0.042	1500	0.052	1500	0.062	1500	0.072	1500	0.082	1500	0.092	1500
EMU	1405.0	1	5	80	2.0E-03	7	0.033	1500	0.043	1500	0.053	1500	0.630	1500	0.073	1500	0.083	1500	0.093	1500
AFT1	20.0	1	5	80	2.0E-03	7	0.034	1500	0.044	1500	0.054	1500	0.064	1500	0.074	1500	0.084	1500	0.094	1500
AFT2	30.0	1	4	80	2.0E-03	7	0.036	1500	0.046	1500	0.056	1500	0.066	1500	0.076	1500	0.086	1500	0.096	1500
TVC	10.0	1	4	80	2.0E-03	7	0.037	1500	0.047	1500	0.057	1500	0.067	1500	0.077	1500	0.087	1500	0.097	1500
IEA	1485.0	1	6	80	2.0E-03	7	0.038	1500	0.048	1500	0.058	1500	0.068	1500	0.078	1500	0.088	1500	0.098	1500
ASA	1495.0	1	9	80	2.0E-03	7	0.035	1500	0.045	1500	0.055	1500	0.065	1500	0.075	1500	0.085	1500	0.095	1500
VSWR	1475.0	1	4	80	2.0E-03	7	0.039	1500	0.049	1500	0.059	1500	0.069	1500	0.079	1500	0.089	1500	0.099	1500
SPC1	800.0	1	10	80	2.0E-03	7	0.011	1500	0.012	1500	0.013	1500	0.014	1500	0.015	1500	0.016	1500	0.017	1500
SPC2	850.0	1	5	80	2.0E-03	7	0.021	1500	0.022	1500	0.023	1500	0.024	1500	0.025	1500	0.026	1500	0.027	1500
SPC3	900.0	1	3	80	2.0E-03	7	0.111	1500	[illegible]	1500	0.131	1500	0.141	1500	0.151	1500	0.161	1500	0.171	1500
SPC4	950.0	1	5	80	2.0E-03	7	0.110	1500	0.120	1500	0.130	1500	0.140	1500	0.150	1500	[illegible]	1500	0.170	1500
SPC5	1000.0	1	3	80	2.0E-03	7	0.112	1500	0.122	1500	0.132	1500	0.142	1500	0.152	1500	0.162	1500	0.172	1500

SOURCE	WAIT TIME DEFER	WAIT TIME COLLISION	DEFER COUNT	COLL COUNT	PKTS TX	ACKS TX	PKTS RX	MINIMUM PKT WAIT TIME	MAXIMUM PKT WAIT TIME	MAX NUM COLLS	MAX PKT COLL TIME
FWD1	1.609E+00	1.059E+01	2292	2877	823	157	980	4.043E-07	5.227E-02	16	2.272E-01
FWD2	1.302E+00	9.194E+00	1853	2317	670	143	813	2.230E-07	5.228E-02	16	2.649E-01
DFI	8.622E-01	6.053E+00	1264	1558	373	118	491	2.046E-07	5.227E-02	16	3.211E-01
EMU	1.373E+00	9.148E+00	1988	2462	651	192	843	3.258E-07	5.227E-02	16	2.428E-01
AFT1	9.239E-01	7.297E+00	1372	1745	455	126	581	2.080E-07	5.227E-02	16	2.861E-01
AFT2	2.019E+00	1.228E+01	2954	3739	1304	357	1661	8.893E-08	5.227E-02	16	3.571E-01
TVC	1.919E+00	1.156E+01	2862	3705	1201	341	1542	3.810E-07	5.227E-02	16	3.844E-01
IEA	1.124E+00	7.160E+00	1643	2027	538	130	668	1.767E-07	5.227E-02	16	3.502E-01
ASA	9.777E-01	8.211E+00	1457	1830	499	133	632	6.174E-07	5.227E-02	16	2.383E-01
VSWR	1.428E+00	1.062E+01	2153	2678	817	162	979	7.761E-08	5.228E-02	16	3.713E-01
SPC1	2.445E+00	1.284E+01	3762	5379	2605	211	2816	1.438E-07	5.228E-02	16	4.166E-01
SPC2	2.179E+00	1.154E+01	3271	4201	1620	189	1809	3.317E-07	5.227E-02	16	2.659E-01
SPC3	1.842E+00	1.005E+01	2611	3105	894	217	1111	2.352E-07	5.228E-02	16	3.718E-01
SPC4	1.192E+00	6.634E+00	1755	2110	579	198	777	8.279E-08	5.227E-02	16	2.085E-01
SPC5	1.120E+00	8.297E+00	1685	2055	466	188	654	3.242E-07	5.229E-02	16	2.772E-01
VAX1	2.538E+00	1.444E+01	3622	5620	1428	6251	7679	3.383E-07	5.677E-02	16	3.363E-01
VAX2	1.102E+00	7.790E+00	1541	1922	204	639	843	5.704E-07	5.229E-02	16	2.262E-01
SPV1	2.041E+00	1.382E+01	2768	4132	434	4156	4590	1.722E-08	5.228E-02	16	2.764E-01
SPV2	1.238E+00	9.449E+00	1736	2221	234	878	1112	1.917E-07	5.228E-02	16	2.911E-01
SPV3	1.429E+00	1.077E+01	2034	2654	413	1011	1424	4.111E-07	5.689E-02	16	2.797E-01
TOT	3.066E+01	1.967E+02	44625	17785	18200	[illegible]	32005	1.722E-08	5.689E-02	16	4.166E-01

AVERAGE	BUSBUSY:	0.000554	USAGE:	0.000537	IDLE:	0.000071
TOTAL	BUSBUSY:	17.727513	USAGE:	17.176959	IDLE:	2.272487

S = SIMULATED THROUGHPUT: 85.884797
G = OFFERED LOAD AS A % OF BUS CAPACITY: 100.831872 (201.326400)
E = EFFICIENCY (OFF LOAD): 85.176240 (42.659481)
T = THEORETICAL THROUGHPUT: 50.207106

TOTAL OFFERED DATA: 91.067600
EFFICIENCY: (OFF DATA): 94.308840

```
TVC-AFT COMM: 150 MSGS NOT TRANSMITTED IN 1 SECOND
VAX:  1 RECEIVED   632346 BYTES IN 1 SECOND
VAX:  2 RECEIVED    97446 BYTES IN 1 SECOND
VAX:  3 RECEIVED   577970 BYTES IN 1 SECOND
VAX:  4 RECEIVED   104870 BYTES IN 1 SECOND
VAX:  5 RECEIVED   115080 BYTES IN 1 SECOND
```

SECOND	VAX	RECEIVED BITS	TRANSMITTED BITS
1	1	1028064	412960
2	1	2044336	400320
3	1	2196336	387056
4	1	2349632	388528
5	1	2172752	188752
6	1	2443536	760336
7	1	2338272	477232
8	1	2941392	323600
9	1	3705280	75952
10	1	3859872	248512
11	1	4323216	76464
12	1	4106208	32224
13	1	4700560	88624
14	1	4729456	0
15	1	4504544	0
16	1	4504976	20336
17	1	4918560	0
18	1	5058768	0
19	1	4908064	20448
20	1	4767856	37984
7	2	235840	370176
8	2	0	12528
9	2	722336	27424
10	2	732832	13280
11	2	343264	12848
12	2	450688	21376
13	2	495696	2160
14	2	685664	11120
15	2	638064	11120
16	2	200464	2160
17	2	449824	11552
18	2	779568	11120
19	2	638496	12848
20	2	638064	11120
1	3	814080	398144
2	3	4281792	394256
3	3	4623760	19152
4	3	3857712	15872
5	3	4128928	19040
6	3	3620576	4752
7	3	3055856	18608
8	3	3113952	4752

9	3	2065328	21200
10	3	1970128	5184
11	3	1864864	16448
12	3	2417360	2160
13	3	1356512	3456
14	3	1722496	11552
15	3	1698048	1296
16	3	1912032	11376
17	3	1733856	10688
18	3	1557408	9648
19	3	1321568	3888
20	3	1380960	18176
3	4	235840	355056
4	4	0	26784
5	4	780000	16304
6	4	803584	14144
7	4	838960	11984
8	4	708816	12848
9	4	508352	13712
10	4	237136	3888
11	4	496992	11552
12	4	530640	2160
13	4	508784	12416
14	4	803152	11552
15	4	696592	3456
16	4	543728	12416
17	4	696160	864
18	4	259856	10256
19	4	543296	1296
20	4	566448	1728
4	5	294800	318384
5	5	141504	401408
6	5	0	18144
7	5	509216	29664
8	5	886560	12240
9	5	898784	10944
10	5	686528	17136
11	5	898352	12672
12	5	744624	9216
13	5	920640	12240
14	5	614912	9216
15	5	757280	10512
16	5	852048	19728
17	5	625840	4320
18	5	722336	17568
19	5	707952	3456
20	5	861680	2160

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle SPACELAB SYSTEM ANALYSIS		5. Report Date September 1988
		6. Performing Organization Code
7. Author(s) F. M. Ingels and T. B. Rives		8. Performing Organization Report No. MSU-EE-FIN-9-88
		10. Work Unit No.
9. Performing Organization Name and Address Department of Electrical Engineering Mississippi State University Mississippi State, MS 39762		11. Contract or Grant No. NAS8-36717
		13. Type of Report and Period Covered INTERIM FINAL REPORT Sept. 87 to Sept. 88
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20516		14. Sponsoring Agency Code

15. Supplementary Notes

Prepared by Mississippi State University for Sherman Jobe, Huntsville Operational Support Center, Huntsville, AL

16. Abstract

An analysis of the Automated Booster Assembly Checkout System (ABACS) has been conducted. A computer simulation of the ETHERNET LAN has been written. The simulation allows one to investigate different structures of the ABACS system. The simulation code is in PASCAL and is VAX compatible.

17. Key Words (Suggested by Author(s)) ETHERNET Simulation LAN-Local Area Network ABACS	18. Distribution Statement

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	214	

NASA FORM 1626 OCT 86