Networking and Communications Technology Laboratory

Design/Development Progress Report Submission #2

J. Thompson
M. Georgiopolous
M. Basiouni

Institute for Simulation and Training
12424 Research Parkway, Suite 300
Orlando FL 32826

University of Central Florida
Division of Sponsored Research

91-11424
DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.
Networking and Communications Technology Laboratory Design/Development Progress Report

Bassiouni, M.; Georgiopolous, M.; Thompson, J.

From 4/89 TO 1/90

Networking and Communications Technology Laboratory, Simulation Network Prototyping Testbed, SIMNET World Access Testbed

This report presents a summary of the progress to date involving the design and development of the Institute For Simulation and Training's Networking and Communications Technology Laboratory. Within this laboratory there are two functional testbeds which house the equipment and capabilities required for carrying out the specific research activities of this project: The Simulation Network Prototyping Testbed and the SIMNET World Access Testbed.

The Simulation Network Prototyping Testbed supports research in several areas pertaining to the use of Local Area Network (LAN) technology for interconnecting Simulation Training Devices. These areas include:

- Carrier Sense Multiple Access with Collision Detection protocol networks (i.e. ETHERNET),
- Token-Ring Networks,
- Fiber Distributed Data Interface (FDDI) Technology,
- Simulations Voice and Data Transmission,
- Non-Homogeneous
Providing access to the SIMNET world is one of the major capabilities IST is developing in the Network and Communications Laboratory. Additional SIMNET modules are being acquired to enhance the existing suite of SIMNET equipment. The addition of this equipment will provide a wide range of SIMNET capabilities to support ongoing research in alternate network implementations, digital voice transmission, network bench marking, and Long Haul Networking.
1. INTRODUCTION
This memo presents a summary of the progress made to date involving the design and development of the Institute for Simulation and Training's Network and Communications Technology Laboratory. Within this laboratory there are two functional testbeds which house the equipment and capabilities required for carrying out the specific research activities of this project. These functional testbeds are the Simulation Network Prototyping Testbed and the SIMNET World Access Testbed.

2. SIMULATION NETWORK PROTOTYPING TESTBED
This testbed supports research in several areas pertaining to the use of Local Area Network (LAN) technology for interconnecting Simulation Training Devices. These research areas include: Carrier Sense Multiple Access with Collision Detection protocol networks (i.e., ETHERNET), Token-Ring Networks, Fiber Distributed Data Interface (FDDI) Technology, Simultaneous Voice and Data Transmission, and Non-Homogeneous Simulator Network Interfacing.

2.1 Testbed Overall Design Approach
A flexible design approach has been developed and adopted for the establishment of the IST Simulation Network Prototyping and Assessment Testbed. The main goal of this approach is to facilitate the investigation and evaluation of alternate network protocols using PC-based platforms. The PC's will provide each SIMNET node with a quasi-contentionless ETHERNET interface. When equipped with appropriate network controller boards, the PC platforms readily provide a gateway capability between networks of different topologies, such as ETHERNET and token-ring. Each PC will also be capable of operating as a controller/protocol translator providing the necessary services for routing SIMNET packets to the alternate network prototypes.

2.2 Testbed Implementation
The Hewlett-Packard Vectra 386 PC/AT Tower System will be used as a data logger, network traffic generator and protocol translator for the Testbed. Our initial tests and evaluation of the intelligent Excelan 205E ETHERNET controller boards have revealed that such intelligent boards would not be able to capture all the broadcast data packets generated in the SIMNET real-time environment. Our data capture prototyping effort will be based, therefore, on dumb ETHERNET controller boards that are optimized for speed of the low-level transmit/receive operations. The high-level TCP/IP processing capability of the intelligent boards, however, will still be used to provide file transfer services for
data analysis, software development, and other applications requiring PC-to-PC ETHERNET communications.

Because of the many features of token-ring protocols, coupled with the commercial availability of token-ring boards for the PC, our alternate network prototyping effort will focus on building a token-ring network configuration for the SIMNET environment. Packets captured off the SIMNET ETHERNET by the PC-platforms will be used to drive the token-ring LAN. Various performance tests to evaluate the token-ring scheme will then be conducted.

2.2.1 Ongoing Activities

The following is a summary of the main activities that have been carried out during the first phase of building the Alternate Network Testbed.

- We have gained considerable experience on using the 3-Com ETHERLINK II dumb ETHERNET boards. With these boards installed in the HP Vectra 20MHz PC's, we are able to transmit packets with data passed from the HP Vectra to the 3-Com board, of length 64, 128 and 256 bytes at rates of 1.8, 2.1 and 2.3 Mbits/sec., respectively. Furthermore, we are able to transmit packets without data passed from the HP Vectra to the 3-Com board, of length 64, 128 and 256 bytes at rates of 3.6, 4.9 and 6.4 Mbits/sec., respectively. The data capture capability of the boards using a single receive buffer is approximately one half of the transmit capability or 1Mbits/sec. These measurements were made over Thin-Net ETHERNET under light traffic loads with minimal collisions.

- We have begun preliminary efforts towards using the HP Vectra's to perform data logging (i.e., to read broadcast packets off the SIMNET ETHERNET, time-stamp and store them to a disk or tape file). These early activities include experimentation with various techniques for time stamping, assessing the impact of missed packets on playback performance, experimentation with optimum precision of time reference used for timestamping.

- We have written a program to generate ETHERNET packets and transmit them out onto the network. Currently, we are working on techniques to provide programmable delay to packet transmissions, as well as generating packets with fixed and jittering interarrival times. Software used to generate simulated packet inter-arrival times in the network simulation software models will be reused to generate actual network traffic. This will allow us to perform more accurate validation experiments on the software models against actual hardware.

- We have written C-language programs to extract and manipulate different fields within a SIMNET protocol data unit (PDU). These programs consist of several header files along with compilable C-routines and have been used in several applications including capturing, manipulating and retransmitting SIMNET M1 data packets, as well as capturing ETHERNET data packets from non-SIMNET simulators and translating them into SIMNET compatible packets.
• We are currently able to pass data packets across the 4Mbits/sec 3-Com TOKENLINK token-ring network boards between two of the HP Vectra's. Experiments are underway to determine the maximum load of SIMNET packets that can be communicated over the ring.

• We are currently performing tests using Concurrent-C simulation models to compare the performance of the early token release protocol of token-ring LAN's with that of the late token release version. These tests will give us an insight into the significance of the improvement in throughput attained through the early release protocol, as well as the amount of network overhead required to support prioritized tokens.

• We are currently building a predictive model to investigate the greedy node problem in Ethernet simulation networks. In our preliminary model, the impact of a greedy node on the transmission of a single non-greedy node is considered and the corresponding channel probabilities are tabulated. It is hoped that this type of modeling will help us evaluate the magnitude of the greedy node problem and its impact on network packet delay and packet loss.

• We are in the process of completing experiments which will allow us to implement ETHERNET-like protocols via the 3-Com Etherlink II boards. Tests have indicated that it may be possible to discard old state update messages from the 3-Com board's transmit buffer and substitute them with new (more recent) update messages. This will allow us to improve the delay performance of the standard ETHERNET protocol.

NOTE: Listings of all software programs mentioned above are included as an attachment.

2.2.2 Planned Activities
The following activities are planned the next phase of the project:

• Improve the data capture capabilities of the 3-Com Etherlink II ETHERNET controller board by implementing a scheme utilizing multiple receive buffers. This will allow us to determine the safe operating range of traffic load for which minimal data loss occurs.

• Design and build C-language software libraries for transmitting and receiving both ETHERNET and token-ring data packets.

• Design and build C-language software programs for performing data logging and artificial packet generation for both the ETHERNET and token-ring LAN's.

• Examine the token-ring priority scheme and evaluate its suitability and potential benefits to optimize packet management in the SIMNET environment.
• Begin using the DURRA software analysis tool developed by Carnegie Mellon University's Software Engineering Institute. This application is written in ADA and will be implemented on a SUN Workstation. Plans are to use DURRA as part of a research task involving the use of intelligent filtering techniques applied at Gateways which interconnect multiple SIMNET type networks via high capacity local area or long haul networks.

• Continue activities involving the use of the 3-Com Etherlink II board to implement ETHERNET-like protocols and investigate the capability of changing some parameters of the standard ETHERNET protocol in an effort to produce priorities on the network. Such parameters include the packet slot-time which directly affects the calculation of the retransmission back-off algorithm, as well as the back-off algorithm itself. We will also focus on the implementation of a modification of the standard ETHERNET protocol that reduces packet transmission delays, only at times when the channel is sensed idle. The final thrust in this effort will be to implement the GBRAM protocol by utilizing the 3-Com ETHERNET board. GBRAM is superior to the ETHERNET protocol for medium to high traffic loads.

2.3 Data Analysis
Data Analysis capabilities in the laboratory will consist of hard and software which will be used to manage and analyze the large amounts of data generated by networked simulators. A variety of test experiments will be conducted in order to evaluate the performance of the various LAN configurations. Different performance measures (e.g., packet transmission delay, distribution of packet inter-arrival times, utilization of transmission medium, LAN throughput, etc.) will be collected and analyzed (using statistical inference) for both ETHERNET and token-ring LAN’s. Some of the statistical tests which will be applied include confidence intervals, analysis of variance, goodness-of-fit tests (e.g., the Kolmogorov-Smirnov test), and regression analysis. A VAX 3100 workstation has been procured and will be used for the performance of the required statistical tests and data analysis services.

2.4.1 Ongoing Activities
The following is a summary of the main activities that have been carried out during the first phase of this research.

• We have gained considerable experience on using the VAX 3100 workstation in both the system administration and user areas.

• Graphics software, the ULTRIX (UNIX for VAX) operating system and some software development tools for the VAX 3100 workstation have been received.

• Chris Pinon has attended the VMS System Management Class I to aid her in administering the VAX 3100 (see Memo for Record from Chris Pinon dated Nov. 20, 1989).
• Local Software and Hardware support has been established through Digital Equipment Computer Users Society (DECUS). Membership has been obtained and a Local User Group meeting was attended (see Memo for Record from Chris Pinon dated Nov. 29, 1989).

• Procurement has begun for statistical packages and data analysis tools.

2.3.2 Planned Activities
The following activities are planned for the next phase of the project:

• Develop a list and a detailed description of the performance measures, statistical experiments and data analysis tests that will be used for evaluating the performance of the ETHERNET interface, as well as the prototype networks to be implemented.

• Procure any statistical software packages found to be suitable for this project.

• Write any necessary software interfaces needed for the invocation of the statistical packages mentioned above.

• Interface VAX DECNET to existing laboratory ETHERNET.

2.4 Simultaneous Voice and Data Transmission Research
Research involving the simultaneous transmission of digital voice and data will be conducted utilizing Digital Signal Processing (DSP) modules interfaced to a networked HP Vectra PC platform. The Ariel DSP56001 DSP modules were chosen and two of the boards were procured for this effort.

2.4.1 Ongoing Activities
The following is a summary of the main activities that have been carried out during the first phase of this research.

• We have received the DSP56001 boards and are gaining experience on using them to manipulate voice data under real-time constraints.

• We have nearly completed the program to packetize the digital voice data that are stored in the memory of DSP56001 Board.

• We are in the process of writing a program to transfer the packetized data from the DSP56001 board to the 3-Com ETHERNET board, and visa versa, for transmission to and reception from the ETHERNET network.

• We are in the process of writing a program to reassemble the packetized data located in the memory of the DSP56001 board into a continuous stream of digital data for subsequent conversion to analog information (voice).
2.4.2 Planned Activities

The following are planned activities which will be performed during the next phase of the project:

- Utilize the aforementioned C-language programs to extract and manipulate different fields within a protocol data unit (PDU) in order to send the voice data over the network in a form that is consistent with the SIMNET communication protocol standard.

- Utilize the capabilities of the DSP56001 board to distort the digitized voice information in a manner that corresponds to the degradation of the analog voice signal in the actual battle environment (RF phenomena).

- Show experimentally, by using the DSP56001 board, the percentage of lost voice packets that we can accommodate without affecting the clarity of the voice signal. This will allow us to find the number of concurrent voice conversations that the network can support in the ETHERNET protocol environment.

- Use the DSP56001 boards to show the effect of certain signal processing techniques on the digitized speech signals (i.e., data compression, coding, voice library tests). By doing so we will expect to accommodate more simultaneous voice conversations on the network.

- Examine the ETHERNET boards carefully to determine the possibility of implementing an alternative protocol (other than ETHERNET) that can support simultaneous voice and data transmission over the network.

2.5 Non-Homogeneous Simulator Network Interfacing

The goal of this research is to provide a proof-of-principle demonstration of interconnecting non-homogeneous simulators via a common network, and provide the means for them to interact with one another.

This activity is on-going in nature and centers on the interconnection of non-SIMNET devices (such as the ASAT’s, the Silicon Graphics’ Networkable Flight Simulator, the SUN Microsystems’ AVIATOR Networkable Simulator, and others) with the existing IST SIMNET devices. Protocol translation/transformation, intelligent filtering techniques for gateways used to interconnect LAN’s of differing topologies, and techniques for handling inconsistencies in data protocol formats between dissimilar simulations are some of the research areas being investigated under this task.

3. SIMNET WORLD ACCESS TESTBED

Providing access to the SIMNET World is one of the major capabilities IST is developing in the Network and Communications Technology Laboratory. Additional SIMNET modules are being acquired to enhance the existing suite of SIMNET equipment. These new modules include a Stealth Vehicle, a Plan View Display, a Data Logger/Playback System and a Long Haul Communications Gateway. The addition of this equipment will provide a wide
range of SIMNET capabilities to support ongoing research efforts in the areas of alternate network implementations, digital voice transmission, network benchmarking, and Long Haul Networking.

3.1 IST SIMNET Network Configuration
As mentioned earlier, the current SIMNET configuration uses an ETHERNET network to provide data communications between simulators. The SIMNET-T site at Ft. Knox uses an interconnect scheme which connects up to eight SIMNET modules together via a multi-port transceiver box, which in turn is attached to the ETHERNET coaxial cable. In the IST Lab, the SIMNET modules are interconnected via a THIN-NET ETHERNET network. THIN-NET uses 50 ohm coaxial cable similar to RG58 to interconnect the nodes on the network. Each node has a small transceiver attached directly to it which provides the required interface to the coaxial cable. This THIN-NET implementation provides a flexible interconnect scheme, without any loss in performance and is more suited to laboratory requirements.

Currently in the IST Laboratory, there are several clusters of computers which are being used for various research activities. By running a series of coaxial cables around the lab we are able to provide a variety of interconnections between the clusters. For example, the SIMNET modules are linked together in one cluster and the networking research equipment (HP LAN Analyzer and PC's with ETHERNET cards) are linked in another. These two clusters can be tied together whenever desired by simply removing two cable termination devices and hooking the two cables together. This scheme allows for the sharing of resources, no matter where they may be physically located in the lab.

3.2 SIMNET Compatible Interconnect Capabilities
This capability in the lab refers specifically to the concept of providing gateways into the SIMNET World. The first gateway to be procured will be a BBN SIMNET Gateway. This gateway is based on the BBN Butterfly computer and most probably will be a closed system, meaning that we will have no way to alter its software and/or hardware to experiment with it. The SIMNET Gateway is being procured, and is expected to be delivered to IST within the next two months.

Commercially available long haul networking hardware is currently being evaluated to determine its suitability for the SIMNET application. Details of this evaluation can be found in the attached memo, Notes on IST Long-haul Interconnectivity, dated 11/29/89. To achieve interconnectivity, we will procure several ETHERNET bridges which will allow for limited dial-up access to the IST SIMNET world, as well as support research being performed in the area of Long Haul Networking.

We have initiated conversations with personnel at Human Engineering Labs (HEL) in Aberdeen Proving Grounds, MD. Preliminary plans are to establish a long haul link between the IST SIMNET Laboratory and HEL's laboratories. There are tentative travel plans for two IST researchers to visit HEL (Aberdeen, MD) during the month of January 1990 to further discuss this project.
3.3 Simulation Network Performance Benchmarks
The functional requirements for a set of benchmarks to be used to evaluate training device network performance and interfacing capabilities will be established. These benchmarks will aid in the validation of interfacing methods between non-homogeneous simulators and compatibility with the current SIMNET communications protocol standard. The benchmarks will consist of a set of software programs which will perform automated analysis of incoming network data, either in real-time or off-line, and will provide an orderly method of evaluating a networked training device's network performance.

Initial benchmark development efforts will employ the use of the VAX 3100 workstation for software development and data analysis. This benchmark work depends highly on the simulation network protocol standards currently under development. Therefore, these activities will be closely monitored and attended to ensure benchmark analysis techniques are valid meaningful measures of performance.

Our initial evaluations indicate a software system called DURRA might be a useful tool to aid in benchmark development. DURRA was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University. IST is the first site to receive DURRA. DURRA is essentially a system for predicting the performance networked computing nodes. DURRA provides a flexible environment for specifying the interconnection of these nodes (i.e. network topology), as well as predicting the system performance under varying loads and usages. DURRA programs can be written which can perform network assessments off-line. On-line assessments will require enhancements which will be pursued by IST and SEI.

4. CONCLUSIONS
This report has presented a summary of the procurements, activities and progress made towards the development of the IST Network and Communications Technology Laboratory. Comments and/or suggestions are encouraged and should be directed to:

Jack Thompson
Institute for Simulation and Training
University of Central Florida
12124 Research Parkway
Orlando, FL 32826
ATTACHMENT A

MEMORANDUMS
MEMORANDUM FOR RECORD

To: Jack Thompson
From: Chris Pinon
Subject: VMS System Management I Class
November 13-17
DEC Education Center
Maitland, Florida
Date: November 20, 1989

Purpose:
The purpose of taking this class was to become more familiar with
the VAXstation's operating system and to learn skills and
commands associated with managing the system. The VAXstation
3100 is an integral part of the Networking laboratory. The
training was necessary to aid in the integration of the VAX onto
the network.

Key Topics:
The class provided an overview of the VMS operating system and
the role of the system manager in maintaining the system. Topics
discussed include:
- Understanding the User Environment
  - Managing System Users
  - Managing Queues
  - Managing Disk and Tape Volumes
  - Customizing the System
  - Starting Up and Shutting Down the System
  - Maintaining System Integrity
  - Monitoring System Performance
  - Installing and Updating System Software

Conclusion:
The class provided an excellent overview of the VMS operating
system and gave the student many valuable tools that can be
implemented immediately. The class fulfilled the purpose
detailed above.

Copy to:
B. Goldiez, S. Smith, J. Cadiz, R. Ouyang, M. Georgiopoulos,
M. Bassiounni
Memorandum

To: Jack Thompson  
From: Chris Pinon  
Subject: Central Florida DECUS LUG November Meeting  
Merritt Island Public Library  
Date: November 29, 1989

Purpose:
The purpose of the meeting was to meet with members of the Central Florida DECUS LUG (DEC users Local Users Group). This group is a valuable resource for help concerning the VAXstation. This is the first meeting attended since joining DECUS. I also sought contacts to help with the transfer of data from one type of tape media to another, an activity essential for the statistical study of the SIMNET data packets and for examining the program from Carnegie-Mel. on University

Key Topics:  
The meeting took place at the Merritt Island Public Library and began at 9:00 am. The meeting proceeded as follows:

1) DECUS business
2) DIGITAL update - an overview of new products on the market
3) "Leveraging PC Applications on the VAX" - a presentation by RECITAL Corporation

LUNCH BREAK

4) "PCSA and 386WARE" - a presentation by Bob Thomson, Computer Operations Supervisor for Martin Marietta Aerospace, KSC
5) General Question and Answer session - A chance for all to discuss problems and solutions. Also a chance to share tips and shortcuts.

The meeting ended at 3:30 pm. I spent some time talking to Mr. Christopher Korson, Software Engineer for Level Five Research, Inc. in Indialantic. He has the means to transfer 8mm, 9mm and TK70 tapes to the TK50 format our computer requires. All IST has to do is provide the tape.

Conclusion:
This meeting provided some valuable information concerning VAX computers in general and some SW products available on the market at this time. It also provided some business contacts that may be valuable in the near future.

Copy to: B. Goldiez, G. Winkler, M. Bassiouni
To: Jack Thompson
From: Jorge Cadiz
Date: 11/29/89
Subject: Notes on IST Long-haul Interconnectivity

It seems that we have the choice to make as far as what type of interface device we would like to use in the Long-haul environment. The three devices that we can use are Bridges, Routers, and Gateways. Following are definitions for these devices. These definitions were extracted from TRW's Unified LAN I Components Guide (July, 1989).

Bridge: A router that connects two or more networks and forwards packets among them. Usually, bridges operate at the physical network level. For example, an ETHERNET bridge connects two physical ETHERNET cables and forwards from one cable to the other exactly those packets that are not local. Bridges differ from repeaters because bridges store and forward complete packets while repeaters forward electrical signals.

Router: Any machine responsible for making decisions about which of several paths network (or Internet) traffic will follow. At the lowest level, a physical network bridge is a router because it chooses whether to pass packets from one physical wire to another. Within a long haul network, each individual packet switch is a router because it chooses routes for individual packets. In the Internet, each IP gateway is a router because it uses IP destination addresses to choose routes.

Gateway: A special purpose, dedicated computer that attaches two or more networks and routes packets from one to the other. In particular, an Internet gateway routes IP datagrams among the networks to which it connects. Gateways route packets to other gateways until they can be delivered to the final destination directly across one physical network. The term is loosely applied to any machine that transfers information from one network to another, as in mail gateway.

After looking at some literature on the three devices, it seems that a bridge may be the type of device that we may want to procure. Bridges are generally faster than routers, and they perform packet filtering in order to prevent some of the "local" traffic from getting onto the long-haul medium.

Routers seem like they may provide more functions than are necessary for our application. In the SIMNET environment a large percentage of the traffic has a broadcast destination address. This means that most of the traffic generated at the different nodes will be looking to be transmitted over
the network. This will require a "dumb" interface which simply passes the traffic to the remote location.

- A gateway will provide a connection between two segments of network that are driven by a different type of protocol. These "protocol translators" are not what we need since the SIMNET units communicate with the same protocols.

- Following is a diagram which is my perception of the long-haul network that will be established by IST.

![Diagram of the network setup]

- I have gathered some product information on some Bridges, Routers, Brouters, etc. Here is a table which summarizes the pricing information.

<table>
<thead>
<tr>
<th>Company</th>
<th>Device</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Computer</td>
<td>ACS 4110 Remote ETHERNET Bridge</td>
<td>$7,500</td>
</tr>
<tr>
<td>Advanced Computer</td>
<td>ACS 4030 Remote ETHERNET Bridge</td>
<td>$4,975-$5,575</td>
</tr>
<tr>
<td>Halley Systems</td>
<td>ConnectLAN 100 Local and Remote Brouter</td>
<td>$?</td>
</tr>
<tr>
<td>Blackbox Corporation</td>
<td>Remote Bridge 56Kbps</td>
<td>$6,600</td>
</tr>
<tr>
<td>Blackbox Corporation</td>
<td>Remote Bridge T1</td>
<td>=12k</td>
</tr>
</tbody>
</table>
ATTACHMENT B

LIST OF ALL SOFTWARE PROGRAMS
/* Description: This file contains the code which calls the functions provide by the CT03L.ASM to receive/transmit packets through 3COM EtherLinkii board. */

#include <stdio.h>
extern cInitAdapters();
extern cInitParameters();
exern cResetAdapter();
exern cWhoAmI();
exern cRdrxFilter();
exern cWrRxFilter();
exern cPutTxData();
exern cGetRxData();
exern cSetLookAhead();
exern cXmit1();
exern cRcvSome();

main()

    int i, j;
    struct init har { 
        char len;
        char non1;
        char non2;
        char non3[21]:
        char non4[41]:
        char non5[41]:
        char non6:
        char cdena[4]:
        char *arg0:
        short args:
        char non7:
    }:

    struct WhoStruct { 
        unsigned char addr[6]:
        char ver major:
        char ver minor:
        char sub_ver:
        char type ds:
        char type adapter:
        char init status:
        char reserved:
        char num_tran buf:
        short size_tran buf:
        long ttl tran cnt:
        long ttl tran err cnt:
        long ttl tran timeout cnt:
        long ttl recv cnt:
        long ttl recv bdr cnt:
        long ttl recv err cnt:
        long ttl retry cnt:
        char xtr mode: 

}
char wait mode:
char hdr_spec data:

struct PktStr {
  char inp[1500];
};

struct WhoStruct far *Who:
struct PktStr far *Pkt:
struct init hd" *parmsdr:
int ttypl, nb, flags, reqid, nreqid:
char far *paddr = "This is a test only":

int rc, rxf=0x00c, rrxf, Adapters=0:
int rs = 0, icnt = 0:
parmsdr->en=0x1d7:
parmsdr->non1=0x00:
parmsdr->non2=0x00:
parmsdr->non3[0]=0x00:
parmsdr->non3[1]=0x00:
parmsdr->non4[0]=0x00:
parmsdr->non4[1]=0x00:
parmsdr->non4[2]=0x00:
parmsdr->non4[3]=0x00:
parmsdr->non5[0]=0x00:
parmsdr->non5[1]=0x00:
parmsdr->non5[2]=0x00:
parmsdr->non5[3]=0x00:
parmsdr->non6=0x00:
parmsdr->cdend[0]=0x00:
parmsdr->cdend[1]=0x00:
parmsdr->cdend[2]=0x00:
parmsdr->cdend[3]=0x00:

/* parmsdr->argo = "c:\3com\ether503.sys /a:2e0/m:4/c:1/d:1/i:1/n": */
parmsdr->argo = "c:\3com\ether503.sys /A:2e0 /D:1 /I:1 /O:0x0a":
parmsdr->args=getds();
parmsdr->non7=0x00:

rc=getds():
printf("getds 0x%x\n",rc);

rc=clnitParameters(parmsdr):
printf("clnitParameters returns %d\n",rc):
rc=clnitAdapters(&Adapters):
printf("clnitAdapters returns %d, Adp=%d\n",rc, Adapters):

rc=cSetLookAhead(32):
printf("cSetLookAhead returns %d\n",rc):

rc=cWhoAmI(&Who):
printf("cWhoAmI returns %d\n",rc):
printf("addr = %02x %02x %02x", Who->addr[0],
  Who->addr[1], Who->addr[2]):
printf("%02x %02x %02x\n", Who->addr[3],
  Who->addr[4], Who->addr[5]):
printf("ver major %02x ver minor %02x\n", Who->ver_major, Who->ver_minor):
printf("transfer mode %x wait mode %x\n", Who->xfr_mode, Who->wait_mode):
printf("ttl recv cnt %d (0x%x)", Who->ttl_recv_cnt, Who->ttl_recv_cnt):
rc=cWrRxFilter(rxf);
printf("cWrRxFilter returns %d\n",rc);
rc=cRdRxFilter(&rrxf);
printf("cRdRxFilter returns %d, filter=%d\n",rc,rrxf);

rs = 'r';
printf("Receiver or Sender ? (r/s)\n");
while ( ((rs = getchar()) != 'r') && (rs != 's') ) {
    printf("Receiver or Sender ? (r/s)\n");
    if (rs == 'r') {
        while (!kbhit())
            rc=cRcvSome(&Pkt);
        if (rc > 0) {
            icnt++;
            printf("cRcvSome returns %d\n",rc);
            for (i=0; i<rc; i++)
                printf("\%02x",Pkt->inp[i]);

            printf("Total input count %d\n",icnt);
        }
    } else {
        ttlp = 0x64;
        nb = 0x64;
        flags = 0x0060;
        reqid = 0x0001;
        nreqid = 0x0011;
        for (i=0; i<10; i++)
            rc=cXmit1(ttlp, nb, flags, reqid, paddr, &nreqid);
    }
rc=ckResetAdapter();
printf("ckResetAdapter returns %d\n",rc);
exit (0);

void myRxProcess(Status, PacketSize, RequestID, PacketHeader)
int Status, PacketSize, RequestID;
char far *PacketHeader;

/* fprintf(stderr,"Called by ASM - myRxProcess\nNot implement yet\n");
fprintf(stderr,"Status=%d, PacketSize=%d, RequestID=%d\n",Status,PacketSize,
RequestID): */

void myTxProcess(Status, RequestID)
int Status, RequestID;

/* printf("Called by ASM - myTxProcess\nNot implement yet\n");
printf("Status=%d, RequestID=%d\n",Status, RequestID): */

void myExitRcvInt()

/* printf("Called by ASM - myExitRcvInt\nNot implement yet\n"); */
title cto3l.asm

*---------------------------------------------------------------------------*
; File: CTO3L.ASM
;
; Description: This file contains subroutines which provide a
; 
; C program with an interface to the 3L 1.0 routines.
*---------------------------------------------------------------------------*

Functions called by C

PUBLIC _getds
PUBLIC _cInitParameters
PUBLIC _cInitAdapters
PUBLIC _cResetAdapter
PUBLIC _cWhoAmI
PUBLIC _cRdRxFilter
PUBLIC _cWrRxFilter
PUBLIC _cPutTxData
PUBLIC _cGetRxData
PUBLIC _cSetLookAhead
PUBLIC _etext
PUBLIC _cRcvSome
PUBLIC _cXmitl

; Need to be written in C
extrn _myExitRcvInt :near
extrn _myRxProcess :near
extrn _myTxProcess :near

; Functions provide by this file
PUBLIC ExitRcvInt
PUBLIC RxProcess
PUBLIC TxProcess

3L functions
extrn InitParameters :near
extrn InitAdapters :near
extrn WhoAmI :near
extrn ResetAdapter :near
extrn RdRxFilter :near
extrn WrRxFilter :near
extrn GetRxData :near
extrn SetLookAhead :near
extrn PutTxData :near

f equ 0ah
cr equ 0dh

print macro strloc ; print string at strloc
local strloc
push ax
push cx
push ds
push dx
mov dx,seg strloc
mov ds,dx
mov dx, offset strloc
mov ah, 09h
int 21h
pop dx
pop ds
pop cx
pop ax
endm

kbdin macro ; get kbd char in al
mov ah, 8
int 21h ; wait for key
endm

@kbdchk macro ; check for kbd char
mov ah, 0bh ; returns al: 0-nokey, ff-keyhit
int 21h
endm

CODE GROUP _TEXT, DATA, ICODE

_TEXT segment byte public 'CODE'

GROUP group _DATA, _BSS
assume cs:_TEXT, ds:GROUP, ss:GROUP

_TEXT ends

DATA segment word public 'CODE'

DATA ends

CODE segment word public 'CODE'

CODE ends

DATA segment

is_ds dw ?
_ etext db ?

ectsv dd 22h dup (0) ; save all vectors so we can cleanup
retsav dw ?
crlf db cr,lf,'$

pklock db 0
pklen dw 0
pkerr dw 0
pkcnt dw 0
pkcount dw 0
pkthd db 32 dup(0)
pktdata db 1500 dup(0)

DATA ends

_DATA segment word public 'DATA'
_d@ label byte
_DATA ends

_BSS segment word public 'BSS'
_b@ label byte
_BSS ends

_DATA segment word public 'DATA'
_s@ label byte
_DATA ends
_TEXT SEGMENT
ASSUME CS:_TEXT, DS:DGROUP, SS:DGROUP

_getds proc near
    mov ax,ds
    mov cs:his_ds,ax
    ret
_getds endp

_cInitAdapters: This procedure provides the glue between a C
program and the 3L 1.0 InitAdapters function.

;Calling Sequence:
    int cInitAdapters(&nAdapters)

;Input Parameters:
    None

;Output Parameters:
    int nAdapters

;Returns:
    The return value of the InitAdapters function
_cInitAdapters proc near
    push bp
    mov bp,sp
    push si
    push di
    push ds
    mov ax,cs
    mov ds,ax
    mov di,offset CODE:RxProcess
    call initAdapters
    pop ds
    mov di,word ptr[bp+4]
    mov word ptr[di],cx
    pop di
    pop si
    pop bp
    ret
_cInitAdapters endp

_cInitParameters: This procedure provides the glue between a C
program and the 3L 1.0 InitAdapters function.

;Calling Sequence:
    int cInitParameters(Parms)

;Input Parameters:
    char *Parms - Pointer to a structure with overrides of default
Output Parameters:
None

Returns:
The return value of the InitParameters function

---

cInitParameters proc near
push bp
mov bp,sp
push si
push di
push ds
mov bx,[bp+4]
mov ax,ds
mov es,ax
mov ax,cs
mov ds,ax
call savvecs
call InitParameters
call savvecs
call InitParameters
call savvecs
pop ds
pop di
pop si
pop bp
ret
cInitParameters endp

---

cResetAdapter: This procedure provides the glue between a C program and the 3L 1.0 ResetAdapters function.

Calling Sequence:
int cResetAdapter()

Input Parameters:
None

Output Parameters:
None

Returns:
The return value of the ResetAdapter function

---
cResetAdapter proc near
push bp
mov bp,sp
push si
push di
push ds
mov dx,0
mov ax,cs
mov ds,ax
ret
cResetAdapter endp
mov    dl,0
call   ResetAdapter
call   fixvecs
pop    ds
pop    di
pop    si
pop    bp
ret
~cResetAdapter endp

$cWhoAmI: This procedure provides the glue between a C program and the 3L 1.0 WhoAmI function.

Calling Sequence:
int cWhoAmI(&WhoPtr)

Input Parameters:
None

Output Parameters:
 struct WhoStruct far *WhoPtr - Far pointer to the WhoAmI structure

Returns:
The return value of the WhoAmI function

cWhoAmI proc near
push   bp
mov    bp,sp
push   si
push   di
push   ds
mov    dx,0
mov    ax,cs
mov    ds,ax
call   WhoAmI
pop    ds
mov    si,[bp+4]
mov    Word ptr [si],di
mov    Word ptr [si+2],es
pop    di
pop    si
pop    bp
ret
cWhoAmI endp

cRdRxFilter: This procedure provides the glue between a C program and the 3L 1.0 RdRxFilter function.
Calling Sequence:
int cRdRxFilter(&RxFilter)

Input Parameters:
None

Output Parameters:
int RxFilter - The receive filter value

Returns:
The return value of the RdRxFilter function

---
cRdRxFilter proc near
push bp
mov bp,sp
push si
push di
push ds
mov ax,cs
mov ds,ax
mov dx,0
call RdRxFilter
pop ds
mov di,(bp-1)
mov dx,di,bx
pop di
pop si
pop bp
ret
cRdRxFilter endp
---
cWrRxFilter: This procedure provides the glue between a C
program and the JL 1.0 WrRxFilter function.

Calling Sequence:
int cWrRxFilter(RxFilter)

Input Parameters:
int RxFilter - The new receive filter value

Output Parameters:
None

Returns:
The return value of the WrRxFilter function

---
cWrRxFilter proc near
push bp
mov bp,sp
push ds
push si
push di
push bp
1

mov    ax,cs
mov    ds,ax
mov    dx,0
mov    ax,[bp+4]
call   WrRxFilter

pop    di
pop    si
pop    ds
pop    bp
ret

_cWrRxFilter endp

---

_cSetLookAhead:  This procedure provides the glue between a C
program and the 3L 1.0 SetLookAhead function.

Calling Sequence:
int cSetLookAhead(NumBytes)

Input Parameters:
int NumBytes - The number of bytes of look ahead data

Output Parameters:
None

Returns:
The return value of the SetLookAhead function

_cSetLookAhead proc near
push   bp
mov    bp,sp
push   si
push   di
push   ds
mov    ax,cs
mov    ds,ax
mov    dx,0
mov    ax,[bp+4]
call   SetLookAhead

pop    ds
pop    di
pop    si
pop    bp
ret
_cSetLookAhead endp

---

_cPutTxData:   This procedure provides the glue between a C
program and the 3L 1.0 PutTxData function.

Calling Sequence:
int cPutTxData(TotalPacketLen, NumBytes, Flags, RequestID, PacketAddr, &NewRequestID)

; Input Parameters:
int TotalPacketLen - The total packet length (first call only)
int NumBytes - The number of bytes to transfer this call
int Flags - The DL flags
int RequestID - Used if not the first call
char far * PacketAddr - A far pointer to the packet

; Output Parameters:
int NewRequestID - Returned after first call

; Returns:
The return value of the PutTxData function

;cPutTxData proc near
push bp
mov bp,sp
push si
push di
push ds
mov ax,ds
mov es,ax
mov bx,[bp+4]
mov cx,[bp+6]
mov dl,byte ptr[bp+8]
mov dh,byte ptr[bp+10]
mov si,[bp+12]
mov di,offset CODE:TxProcess
mov di,0ffffh ; no TxProcess
call PutTxData
pop ds
xchg dh,dl
xor dh,dh
mov di,[bp+16]
mov [di],dx
pop di
pop si
pop bp
ret
cPutTxData endp

;cGetRxData: This procedure provides the glue between a C program and the 3L 1.0 GetRxData function.

; Calling Sequence:
icGetRxData(&NumBytes, Flags, RequestID, PacketAddr)

; Input Parameters:
ic NumBytes - The number of bytes to transfer this call
int Flagt. - The DL flags
int RequestID - The request identifier
char far * PacketAddr - A far pointer to the packet to copy the data

Output Parameters:
int NumBytes - The actual number of bytes transferred

Returns:
The return value of the GetRxData function

---
cGetRxData proc near
push bp
mov bp,sp
push si
push di
push ds
mov di,[bp+4]
mov cx,ss:[di]
mov dl,byte ptr[bp+6]
mov dh,byte ptr[bp+8]
mov di,[bp+10]
mov es,[bp+12]
call GetRxData
pop ds
mov si,[bp+4]
mov ss:[di],cx
pop di
pop si
pop bp
ret
cGetRxData endp

---

TxProcess: This procedure is the protocol-side routine which is called when a packet has finished transmitting (see _cInitAdapters). It provides the glue between the DL 1.0 routines and C routine called myTxProcess.

myTxProcess Calling Sequence:
void myTxProcess(Status, RequestID)

myTxProcess Input Parameters:
int Status - Receive status
int RequestID - The request identifier

myTxProcess Returns:
Nothing

TxProcess proc near
push bp
push si
push di
push ds
push es
; ExitRcvInt: This procedure is the protocol-side routine which is called
; when the 3L has completed a receive interrupt. It provides
; the glue between the 3L 1.0 routines and C routine called
; myExitRcvInt.

; myExitRcvInt Calling Sequence:
; void myExitRcvInt()

; myExitRcvInt Input Parameters:
; None

; myExitRcvInt Returns:
; Nothing

exitRcvInt proc near
    push bp
    push ds
    push es
    push di
    push ax
    mov ax,cs:his_ds
    mov ds,ax
    mov es,ax
    pop ax
    call _myExitRcvInt
    pop di
    pop si
    pop es
exitRcvInt endp
; RxProcess: This procedure is the protocol-side routine which is called
; when a packet has been received (see _cInitAdapters). It provides
; the glue between the 3L 1.0 routines and C routine called
; myRxProcess.

myRxProcess Calling Sequence:
void myRxProcess(Status, PacketSize, RequestID, PacketHeader)

myRxProcess Input Parameters:
int Status - Receive status
int PacketSize - Size of the received packet
int RequestID - The request identifier
char far * PacketHeader - Address of the virtual packet header

myRxProcess Returns:
Nothing

xProcess proc near
; comment #
push bx
push cx
push dx
push si
push di
push bp
push ds
push es
pushf

push es
push di

push ax
mov ax,cs:his_ds
mov ds,ax
mov es,ax
pop ax

xor bx,bx
mov bl,dh
xor dh,dh

push bx
push cx
push ax

call _myRxProcess
add sp,10

popf
pop es
pop ds
pop    bp
pop    di
pop    si
pop    dx
pop    cx
pop    bx
ret

push   bx
push   cx

test   cs:pklock,0ffh
jz     getp

dontget:
  ;inc   pkcount
  inc    cs:pkcount
  mov    cx,0       ;zero length (just discard)
  jmp    goget

goget:
  ; At this point we could check es:di packet header data
  ; to make some decision on packet disposition
  ; lock our buffer and get packet data into it
  mov    cs:pklock,0ffh ;lock buff
  mov    cs:pkerr,0

  mov    ax,CODE
  mov    es,ax
  mov    di,offset cs:pkthd ;buffer
  or     dl,40h           ;release buffer
  call   GetRxData

  jcxz   nolen
  mov    cs:pkerr,ax
  mov    cs:pklen,cx

nolen:
  pop    cx
  pop    bx
  ret

RxProcess endp

; -----------------------------------------------
; _cXmitl proc near
; transmit one packet
_cXmitl proc near

push   bp
mov    bp,sp
push   si
push   di
push   ds

mov    ax,ds
mov    es,ax

;setup for PutTxData
mov    bx,[bp+4]     ;set lengths

B-15
mov cx,[bp+6]
mov dl, byte ptr [bp+8]
mov dh, byte ptr [bp+10]
mov si,[bp+12]
mov di,0ffffh ;no TxProcess

call PutTxData

pop ds
xchg dh,dl
xor dh,dh
mov di,[bp+16]
mov [di],dx

pop di
pop si
pop bp
ret

_cXmit1 endp

;------------------------------------------------------------------------
;_cRcvSome proc near
;following code to dump received packets for a fixed time
;------------------------------------------------------------------------

cRcvSome proc near
push bp
mov bp,sp
push si
push di
push ds

mov ax,cs
mov ds,ax

hkpk:
test cs:pklock,0ffh ;got a pkt?
jnz lstpkt
mov cs:pklen,0 ; No pkt, move 0 to pklen
jmp wedone

lstpkt:
test cs:pkerr,0ffffh ;any error
jz dmpk
jmp wedone

mpk:
cmp cs:pklen,0
jnz pkok
jmp wedone

kok:
cmp cs:pklen,256
jle wedone
mov cs:pklen,256 ;limit dump to 1st 256 bytes

edone:
mov cs:pklock,0
inc cs:pkcnt
mov ax,cs
pop ds
mov si,[bp+4]
mov word ptr [si], offset cs:pkthd
mov word ptr [si+2], ax
mov ax,cs:pklen
pop di
pop si
pop bp
ret
cRcvSome endp

avvecs proc near
push ds
push es
push si
push di
push cx
mov ax,ds
mov es,ax
xor ax,ax
mov ds,ax
mov cx,22h*2 ;vectors 0 - 21h, 2 wds per
mov di,offset CODE:vectsv
xor si,si
cld
cld
cld cli
rep movsw ;save 'em all
sti
pop cx
pop di
pop si
pop es
pop ds
ret savvecs endp

ixvecs proc near
push es
push si
push di
push cx
push ax
xor ax,ax
mov es,ax
mov cx,22h*2 ;vectors 0 - 21h, 2 wds per
mov si,offset CODE:vectsv
xor di,di
cld
cld cli
rep movsw ;restore 'em all
sti
pop ax
pop cx
pop di
pop si
pop es
ret
ixvecs endp
ends
end
/* Description: This file contains the code which calls the functions provide by the CTO3L.ASM to receive/transmit packets through 3COM Token Ring board. */

#include <stdio.h>

extern cInitAdapters();
extern cInitParameters();
extern cResetAdapter();
extern cWhoAmI();
extern cRdRxFilter();
extern cWrRxFilter();
extern cPutTxData();
extern cGetRxData();
extern cSetLookAhead();
extern cXmit();
extern cRcvSome();

main()
{
    int i;
    struct ini_hdr
    {
        char len;
        char non1;
        char non2;
        char non3[2];
        char non4[4];
        char non5[4];
        char non6;
        char cdend[4];
        char *argo;
        short args;
        char non7;
    };

    struct WhoStruct
    {
        unsigned char addr[6];
        char ver_major;
        char ver_minor;
        char sub_ver;
        char type_ds;
        char type_adapter;
        char init_status;
        char reserved;
        char num_trans_buf;
        short size_trans_buf;
        long ttl_trans_cnt;
        long ttl_trans_err_cnt;
        long ttl_trans_timeout_cnt;
        long ttl_recp_cnt;
        long ttl_recv_bdr_cnt;
        long ttl_recv_err_cnt;
        long ttl_retry_cnt;
        char xfr_mode;
    };
}
char wait_mode;
char hdr_spec_data;

struct TokenFrame {
    unsigned char da[6];
    unsigned char sa[6];
    unsigned char info[16];
};

struct PktStr {
    unsigned char inp[1500];
};

struct WhoStruct far *Who;
struct PktStr far *Pkt;
struct ini_hdr ddh;
struct ini_hdr *parmsdr = &ddh;
struct TokenFrame tkbuf;
struct TokenFrame *ptkbuf = &tkbuf;

int ttlpl, nb, flags, reqid, nreqid;

int rc, rxf=0x0005, rrxf, Adapters=0;
int rs = 0, icnt = 0;
 parmsdr->len=0x17;
 parmsdr->non1=0x00;
 parmsdr->non2=0x00;
 parmsdr->non3[0]=0x00;
 parmsdr->non3[1]=0x00;
 parmsdr->non4[0]=0x00;
 parmsdr->non4[1]=0x00;
 parmsdr->non4[2]=0x00;
 parmsdr->non4[3]=0x00;
 parmsdr->non5[0]=0x00;
 parmsdr->non5[1]=0x00;
 parmsdr->non5[2]=0x00;
 parmsdr->non5[3]=0x00;
 parmsdr->non6=0x00;
 parmsdr->cdend[0]=0x00;
 parmsdr->cdend[1]=0x00;
 parmsdr->cdend[2]=0x00;
 parmsdr->cdend[3]=0x00;
 parmsdr->argo = "c:\\3com\\tok603.sys 5,300,0,,\0x0a":
 parmsdr->args=getds();
 parmsdr->non7=0x00;

rc=getds();
printf("getds 0x%x\n",rc);

rc=cInitParameters(parmsdr);
printf("cInitParameters returns %d\n",rc);
rc=cInitAdapters(&Adapters);
printf("cInitAdapters returns %d, Adp=%d\n",rc, Adapters);

rc=cSetLookAhead(32);
printf("cSetLookAhead returns %d\n",rc);

rc=cWhoAmI(&Who);
printf("cWhoAmI returns %d\n",rc);
printf("addr = \%02x \%02x \%02x", Who->addr[0],
    Who->addr[1], Who->addr[2]);
printf(" \%02x \%02x \%02x\n", Who->addr[3],
    Who->addr[4], Who->addr[5]);
printf("ver major \%02x ver minor \%02x\n", Who->ver_major, Who->ver_minor);
printf("adapter type \%02x\n", Who->type_adapter);
printf("xfr_mode \%x wait_mode \%x\n", Who->xfr_mode, Who->wait_mode);
printf("ttl recp cnt \&d (0x\%4x)\n", Who->ttl_recpt, Who->ttl_recpt);
for (i=0; i<=5; i++)
    ptkbuf->da[i] = 0xff;
for (i=0; i<=5; i++)
    ptkbuf->sa[i] = Who->addr[i];
rc=cWwRxFilter(rxf);
printf("cWwRxFilter returns \%d
",rc);
rc=cRdRxFilter(&rrf);
printf("cRdRxFilter returns \%d, filter=%x\n",rc,rrf);
rs = ' ';
printf("Receiver or Sender ? (r/s)\n");
while ( ((rs = getchar()) != 'r' ) && (rs != 's') )
    printf("Receiver or Sender ? (r/s)\n");
rc=crvSome(&Pkt):
    if (rc > 0) {
        printf("length = \%d\n", rc):
        for (i=0; i<rc; i++)
            printf(" \%x", Pkt->inp[i]):
        icnt++:
    }
    printf("Total input count \%d\n",icnt):
} else {
    ttlpl = 0x1c:
    nb = 0x1c:
    flags = 0x0060:
    reqid = 0x0001:
    nreqid = 0x0011:
    for (i=0; i<10; i++)
        rc=cXmittl(ttlpl, nb, flags, reqid, ptkbuf, &nreqid):

rc=cResetAdapter():
printf("cResetAdapter returns \%d\n",rc);
exit (0):

void myRxProcess(Status, PacketSize, RequestID, PacketHeader)
int Status, PacketSize, RequestID:
char far *PacketHeader:
    /* fprintf(stderr,"Called by ASM - myRxProcess\n Not implement yet\n"):
    fprintf(stderr, "Status=%d, PacketSize=%d, RequestID=%d\n", Status, PacketSize, B-21
void myTxProcess(Status, RequestID)
{
    int Status, RequestID;
    /* printf("Called by ASM - myTxProcess\n Not implement yet\n"");
    printf("Status=%d, RequestID=%d\n",Status, RequestID); */
}

void myExitRcvInt()
{
    /* printf("Called by ASM - myExitRcvInt\n Not implement yet\n"); */
}
title cto3l.asm

*---------------------------------------------------------------------*

; File: CTO3L.ASM

; Description: This file contains subroutines which provide a
; C program with an interface to the 3L 1.0 routines.

*---------------------------------------------------------------------*

; Functions called by C

PUBLIC _getds
PUBLIC _cInitParameters
PUBLIC _cInitAdapters
PUBLIC _cResetAdapter
PUBLIC _cWhoAmI
PUBLIC _cRdRxFilter
PUBLIC _cWrRxFilter
PUBLIC _cPutTxData
PUBLIC _cGetRxData
PUBLIC _cSetLookAhead
PUBLIC _etext
PUBLIC _cRcvSome
PUBLIC _cXmitInt

; Need to be written in C
extrn _myExitRcvInt :near
extrn _myRxProcess :near
extrn _myTxProcess :near

; Functions provide by this file
PUBLIC ExitRcvInt
PUBLIC RxProcess
PUBLIC TxProcess

; 3L functions
extrn InitParameters :near
extrn InitAdapters :near
extrn WhoAmI :near
extrn ResetAdapter :near
extrn RdRxFilter :near
extrn WrRxFilter :near
extrn GetRxData :near
extrn SetLookAhead :near
extrn PltTxData :near

; f equ 0ah
; cr equ 0dh

; print macro strloc :print string at strloc
local strloc
push ax
push cx
push ds
push dx
mov dx,word ptr strloc
mov ds,dx
mov     dx, offset strloc
mov     ah, 09h
int     21h
pop     dx
pop     ds
pop     cx
pop     ax
endm

@kbdin macro ; get kbd char in al
    mov     ah, 0h
    int     21h ; wait for key
endm

@kbdchk macro ; check for kbd char
    mov     ah, 0bh
    int     21h ; returns al: 0-nokey, ff-keyhit
endm

CODE GROUP _TEXT, DATA, ICODE

.TEXT segment byte public 'CODE'
GROUP group _DATA, _BSS
assume cs: _TEXT, ds: DGROUP, ss: DGROUP

_TEXT ends

DATA segment word public 'CODE'
DATA ends

ICODE segment word public 'CODE'
ICODE ends

DATA segment
his_ds  dw     ?
e_text db     ?

vectsv dd     22h dup (0) ; save all vectors so we can cleanup
retsav dw     ?
crlf     db     cr, lf, '$'
pklock    db     0
pklen     dw     0
pkerr     dw     0
pkcnt     dw     0
pkcount   dw     0
pkthd     db     32 dup(0)
pktdat    db     1500 dup(0)

DATA ends

_DATA segment word public 'DATA'
d d @ label byte
_DATA ends

_BSS segment word public 'BSS'
_b @ label byte
_BSS ends

_DATA segment word public 'DATA'
s s @ label byte
_DATA ends
.TEXT SEGMENT
ASSUME CS:_TEXT, DS:DGROUP, SS:DGROUP

_getds proc near
mov ax,ds
mov cs:his_ds,ax
ret
_getds endp

;-------------------------------
;_clnitAdapters: This procedure provides the glue between a C
;program and the 3L 1.0 InitAdapters function.

;Calling Sequence:
;   int cInitAdapters(&nAdapters)

;Input Parameters:
;   None

;Output Parameters:
;   int nAdapters

;Returns:
;   The return value of the InitAdapters function

;-------------------------------
_cInitAdapters proc near
push bp
mov bp,sp
push si
push di
push ds
mov ax,cs
mov ds,ax
mov di,offset CODE:RxProcess
call InitAdapters
pop ds
mov di,word ptr[bp+4]
mov word ptr[di],cx
pop di
pop si
pop bp
ret
_cInitAdapters endp

;-------------------------------
;cInitParameters: This procedure provides the glue between a C
;program and the 3L 1.0 InitAdapters function.

;Calling Sequence:
;   int cInitParameters(Parms)

;Input Parameters:
;   char *Parms - Pointer to a structure with overrides of default

B-76
Output Parameters:
  None

Returns:
  The return value of the InitParameters function

_initParameters proc near
  push bp
  mov bp, sp
  push si
  push di
  push ds
  mov bx, [bp+4]
  mov ax, ds
  mov es, ax
  mov ax, cs
  mov ds, ax
  call savvecs
  call InitParameters
  pop ds
  pop di
  pop si
  pop bp
  ret
_initParameters endp

_cResetAdapter: This procedure provides the glue between a C program and the 3L 1.0 ResetAdapters function.

Calling Sequence:
  int cResetAdapter()

Input Parameters:
  None

Output Parameters:
  None

Returns:
  The return value of the ResetAdapter function

_cResetAdapter proc near
  push bp
  mov bp, sp
  push si
  push di
  push ds
  mov dx, 0
  mov ax, cs
  mov ds, ax
  call savvecs
  call InitParameters
  pop ds
  pop di
  pop si
  pop bp
  ret
_cResetAdapter endp
mov     dl,0 ; Ruey Ouyang
call    ResetAdapter
call    fixvecs

pop     ds
pop     di
pop     si
pop     bp

ret

_cResetAddress endp

;cWhoAmI: This procedure provides the glue between a C
;program and the 3L 1.0 WhoAmI function.

;Calling Sequence:
;    int cWhoAmI(&WhoPtr)

;Input Parameters:
;    None

;Output Parameters:
;    struct WhoStruct far *WhoPtr - Far pointer to the WhoAmI structure

;Returns:
;    The return value of the WhoAmI function

_cWhoAmI proc near
push    bp
mov     bp,sp
push    si
push    di
push    ds

mov     dx,0
mov     ax,cs
mov     ds,ax

call    WhoAmI

pop     ds
mov     si,[bp+4]
mov     Word ptr [si],di
mov     Word ptr [si+2],es

pop     di
pop     si
pop     bp
ret

_cWhoAmI endp

;_cRdRxFilter: This procedure provides the glue between a C
;program and the 3L 1.0 RdRxFilter function.

----------------------------------------------------------------------------
Calling Sequence:
    int cRdRxFilter(&RxFilter)

Input Parameters:
    None

Output Parameters:
    int RxFilter - The receive filter value

Returns:
    The return value of the RdRxFilter function

---

cRdRxFilter proc near
    push bp
    mov bp,sp
    push si
    push di
    push ds

    mov ax,cs
    mov ds,ax

    mov dx,0
    call RdRxFilter

    pop ds
    mov di,[bp+4]
    mov [di],bx

    pop di
    pop si
    pop bp
    ret

cRdRxFilter endp

---

_cWrRxFilter: This procedure provides the glue between a C program and the 3L 1.0 WrRxFilter function.

Calling Sequence:
    int cWrRxFilter(RxFilter)

Input Parameters:
    int RxFilter - The new receive filter value

Output Parameters:
    None

Returns:
    The return value of the WrRxFilter function

---

cWrRxFilter proc near
    push bp
    mov bp,sp
    push ds
    push si
    push di

    pop di
    pop si
    pop bp
    ret

cWrRxFilter endp
; cSetLookAhead: This procedure provides the glue between a C program and the 3L 1.0 SetLookAhead function.

; Calling Sequence:
\hspace{1cm} int cSetLookAhead(NumBytes)

; Input Parameters:
\hspace{1cm} int NumBytes - The number of bytes of look ahead data

; Output Parameters:
\hspace{1cm} None

; Returns:
\hspace{1cm} The return value of the SetLookAhead function

_cSetLookAhead proc near
    push bp
    mov bp,sp
    push si
    push di
    push ds
    mov ax,cs
    mov ds,ax
    mov dx,0
    mov ax,[bp+4]
call SetLookAhead
pop ds
pop di
pop si
pop bp
ret
_cSetLookAhead endp

; _cPutTxData: This procedure provides the glue between a C program and the 3L 1.0 PutTxData function.

; Calling Sequence:
int cPutTxData(TotalPacketLen, NumBytes, Flags, RequestID, PacketAddr, &NewRequestID)

; Input Parameters:
; int TotalPacketLen - The total packet length (first call only)
; int NumBytes - The number of bytes to transfer this call
; int Flags - The DL flags
; int RequestID - Used if not the first call
; char far * PacketAddr - A far pointer to the packet

; Output Parameters:
; int NewRequestID - Returned after first call

; Returns:
; The return value of the PutTxData function

_cPutTxData proc near
    push bp
    mov bp,sp
    push si
    push di
    push ds
    mov ax,ds
    mov es,ax
    mov bx,[bp+4]
    mov cx,[bp+6]
    mov dl,byte ptr[bp+8]
    mov dh,byte ptr[bp+10]
    mov si,[bp+12]
    mov di,offset CODE:TxProcess
    mov di,0ffffffh ; no TxProcess
    call PutTxData
    pop ds
    xchg dh,dl
    xor dh,dh
    mov di,[bp+16]
    mov [di],dx
    pop di
    pop si
    pop bp
    ret
_cPutTxData endp

_cGetRxData: This procedure provides the glue between a C program and the 3L 1.0 GetRxData function.

; Calling Sequence:
; int cGetRxData(&NumBytes, Flags, RequestID, PacketAddr)

; Input Parameters:
; int NumBytes - The number of bytes to transfer this call
int Flags - The DL flags
int RequestID - The request identifier
char far * PacketAddr - A far pointer to the packet to copy the data

Output Parameters:
int NumBytes - The actual number of bytes transferred

Returns:
The return value of the GetRxData function

cGetRxData proc near
    push bp
    mov bp,sp
    push si
    push di
    push ds
    mov di,[bp+4]
    mov cx,ss:[di]
    mov di,byte ptr[bp+6]
    mov dh,byte ptr[bp+8]
    mov di,[bp+10]
    mov es,[bp+12]
    call GetRxData
    pop ds
    mov di,[bp+4]
    mov ss:[di],cx
    pop di
    pop si
    pop bp
    ret
_cGetRxData endp

TxProcess: This procedure is the protocol-side routine which is called when a packet has finished transmitting (see _cInitAdapters). It provides the glue between the 3L 1.0 routines and C routine called myTxProcess.

myTxProcess Calling Sequence:
void myTxProcess(Status, RequestID)

myTxProcess Input Parameters:
    int Status - Receive status
    int RequestID - The request identifier

myTxProcess Returns:
    Nothing

TxProcess proc near
; ExitRcvlnt: This procedure is the protocol-side routine which is called when the 3L has completed a receive interrupt. It provides the glue between the 3L 1.0 routines and C routine called myExitRcvlnt.

myExitRcvlnt Calling Sequence:
    void myExitRcvlnt()

myExitRcvlnt Input Parameters:
    None

myExitRcvlnt Returns:
    Nothing

ExitRcvlnt proc near
    push bp
    push ds
    push es
    push si
    push di
    push ax
    mov ax,cs:his_ds
    mov ds,ax
    mov es,ax
    push ax
    call _myExitRcvlnt
    pop di
    pop si
    pop es
    ret

xProcess endp

push    ax
mov     ax,cs:his_ds
mov     ds,ax
mov     es,ax
pop     ax
xor     cx,cx
mov     cl,dh
xor     dh,dh
push    cx
push    ax
call    _myTxProcess
add     sp,4
pop     es
pop     ds
pop     di
pop     si
pop     bp
ret

; ExitRcvlnt: This procedure is the protocol-side routine which is called when the 3L has completed a receive interrupt. It provides the glue between the 3L 1.0 routines and C routine called myExitRcvlnt.

myExitRcvlnt Calling Sequence:
    void myExitRcvlnt()

myExitRcvlnt Input Parameters:
    None

myExitRcvlnt Returns:
    Nothing

ExitRcvlnt proc near
    push    bp
    push    ds
    push    es
    push    si
    push    di
    push    ax
    mov     ax,cs:his_ds
    mov     ds,ax
    mov     es,ax
    pop     ax
    call    _myExitRcvlnt
    pop     di
    pop     si
    pop     es
    ret

xProcess endp
; RxProcess: This procedure is the protocol-side routine which is called
; when a packet has been received (see _cInitAdapters). It provides
; the glue between the 3L 1.0 routines and C routine called
; myRxProcess.

myRxProcess Calling Sequence:
void myRxProcess(Status, PacketSize, RequestID, PacketHeader)

myRxProcess Input Parameters:
int Status - Receive status
tPacketSize - Size of the received packet
int RequestID - The request identifier
cchar far *PacketHeader - Address of the virtual packet header

myRxProcess Returns:
Nothing

RxProcess proc near
;Comment #
push bx
push cx
push dx
push si
push di
push bp
push ds
push es
pushf

push es
push di

push ax
mov ax,cs:his_ds
mov ds,ax
mov es,ax
pop ax
xor bx,bx
mov b1,dh
xor dh,dh

push bx
push cx
push ax

call _myRxProcess
add sp,10

popf
pop es
pop ds
; At this point we could check es:di packet header data
; to make some decision on packet disposition
; lock our buffer and get packet data into it
mov cs:pklock,0ffh ;lock buff
mov cs:pkerr,0

getp:
    inc cs:pkcount
    mov cx,0 ;zero length (just discard)
    jmp goget

jz getp
    test cs:pklock,0ffh
    jz getp

getp:

    ; At this point we could check es:di packet header data
    ; to make some decision on packet disposition
    ; lock our buffer and get packet data into it
    mov cs:pklock,0ffh ;lock buff
    mov cs:pkerr,0
    mov ax,CODE
    mov es,ax
    mov di,offset cs:pkthd ;buffer
    or dl,40h ;release buffer
    call GetRxData

jcxz nolen
    mov cs:pkerr,ax
    mov cs:pklen,cx

jolen:
    POP cx
    POP bx
    RET

RxProcess endp

; cXmitl proc near
transmit one packet
cXmitl proc near

push bp
push si
push di
push ds
mov ax,ds
mov es,ax

mov bx,[bp+4] ;set lengths
mov cx,[bp+6]
mov dl,byte ptr[bp+8]
mov dh, byte ptr[bp+10]
mov si,[bp+12]
mov di,0fffh ;no TxProcess

call PutTxData

pop ds
xchg dh,dl
xor dh,dh
mov di,[bp+16]
mov [di],dx

pop di
pop si
pop bp
ret

_cXmit1 endp

-------------------------------

_cRcvSome proc near
following code to dump received packets for a fixed time
-------------------------------

_cRcvSome proc near
push bp
mov bp,sp
push si
push di
push ds

mov ax,cs
mov ds,ax

chkpk:
test cs:pklock,0ffh ;got a pkt?
jnz lstpkt
mov cs:pklen, 0 ; No pkt, move 0 to pklen
jmp wedone

.stpkt:
test cs:pkerr,0ffffh ;any error
jz dmpk
jmp wedone

dmpk:
cmp cs:pklen,0
jnz pkok
jmp wedone

pkok:
cmp cs:pklen,256
jle wedone
mov cs:pklen,256 ;limit dump to 1st 256 bytes

wedone:
mov cs:pklock,0
inc cs:pkcnt
mov ax,cs
pop ds
mov si,[bp+4]
mov word ptr [si], offset cs:pkthd
mov word ptr [si+2], ax
mov ax,cs:pklen

pop di
pop si
pop bp
ret.

_avvecs proc near
push ds
push es
push si
push di
push cx

mov ax,ds
mov es,ax
xor ax,ax
mov ds,ax
mov cx,22h*2 ;vectors 0 - 21h, 2 wds per
mov di,offset CODE:vectsv
xor si,si
cld
cld
rep movsw ;save 'em all
sti
pop cx
pop di
pop si
pop es
pop ds
ret

_avvecs endp

_ixvecs proc near
push es
push si
push di
push cx
push ax

xor ax,ax
mov es,ax
mov cx,22h*2 ;vectors 0 - 21h, 2 wds per
mov si,offset CODE:vectsv
xor di,di
cld
cld
rep movsw ;restore 'em all
sti
pop ax
pop cx
pop di
pop si
pop es
ret

_ixvecs endp
}

TEXT ends
end
This program displays the airplane controlled by the SiliconGraphics or the simnet.

simnet: Link Level Raw Ethernet Packets / Synchronous Non-blocking
SiliconGraphics: Synchronous-blocking UDP/IP or (disk file)

#include <sys/types.h>
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <sys/exerrno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <signal.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/exosopt.h>
#include <sys/uxos.h>
#include <ex_ioctl.h>
#include <sys/ioctl.h>
#include <sys/dcb.h>
#include "../simnet.h/simnet2.h"
#include "../flight.h/flight.h"

struct sockaddr_link recv_socket = { AF_INET6FILTER);
struct sockaddr_link send_socket = { AF_INET6FILTER);
struct sockaddr_in recv_socket_sg = { AF_INET6);
struct sockaddr_in send_socket_sg = { AF_INET6);

#define FILEOFLAG (O_RDONLY | O_BINARY)
#define FILEPMODE (0)
#define PI 3.14159

extern int errno;
extern int break enabled;
extern int abort_op;

int diskfd = -1; /* disk file */
int netfd = 1; /* simnet file */
int netfdsg = -1; /* udp/ip file */
int timelimit = 30;
char *inputfile;

char SENDIT;
char but[1024];

int break handler()
main (argc, argv)
char **argv:
int an, i, j, pdukind, netcnt;
signal(SIGINT, break_handler);
break_enabled = 1;
inputfile = argv[1];
sginitin();
netinit();

/* Capture a simnet packet first, so we don't have to fill all of the data field */
fprintf(stderr, "wait for simnet\n");
while(1) {
    /* netcnt=netread(inbuf); */
    netcnt=netread();
data_length.p_data_length= ntohs(ether_buf.simnet_data.e_data_length):
    netcnt=data_length.i_data_length.length + HEADER_SIZE;
    memcpy(&pdu_buf, &ether_buf.simnet_data, netcnt - HEADER_SIZE);
    pdukind = ntoh_simnet();
    if (pdukind == vehicleAppearancePDUKind) {
        SENDIT = 'A';
        if (ether_buf.e_host[5] == TANKA)
            SENDIT = 'A';
        if (ether_buf.e_host[5] == TANKB)
            SENDIT = 'B';
    }
    if (SENDIT == 'A') || (SENDIT == 'B') break;
}

fprintf(stderr, "Got a vehicle appearance packet from tank %c\n", SENDIT);
pdu_buf.VAPDU.VADATA.hdr.vehicleID = MYTANKID;
pdu_buf.VAPDU.VADATA.appearance.vehKindMask = A10;
memcpy (ether_buf.e_host, my_addr, sizeof(my_addr));

while (1) {
    netcnt = sgreadin();
    if (netcnt <= 0) break;
    memcpy(&plane, buf, netcnt);
    ntoh_flight();
    pdu_buf.VAPDU.VADATA.location[0] = AIRPORTX + ((plane.x + ADJUSTX)/F2M);
    pdu_buf.VAPDU.VADATA.location[1] = AIRPORTY + ((plane.y/F2M) + ((plane.z + ADJUSTZ)/F2M);
    pdu_buf.VAPDU.VADATA.location[2] = AIRPORTY + (plane.y/F2M);
    calrotation();
    hton_simnet();
    memcpy (&ether_buf.simnet_data, &pdu_buf, netcnt - HEADER_SIZE);
    netwrite();
}

fprintf (stderr, "End of input sq packet\n");
close(diskfd);
sgfiniin();
netfiniin();

errexit(errstring);
char *errstring;
if (errno) experror(errstring);
else fprintf(stderr, "%s
usage: dogdisk filename
", errstring);
close(diskfd);
soclose(netfdsg);
netfini();
exit(1);

break_handler() /* break handler ... control-break or control-c */

static int break_count = 0;

if (++break_count == 1) {
    /* first time, just try to stop current network operation */
    abort_op = 1;
    signal(SIGINT, break_handler); /* reset trap */
    return;
}
else {
    /* second time, try to clean up, then quit */
    errexit("user abort");
}

*///pinfo(opp);
struct exosopt *optp:
/* note that this routine will not return valid results
* if used with a pre-3.3 driver, which interpreted the
* board memory address as absolute, rather than relative
* to the beginning of the data segment
*/
long optaddress = 0; /* location of options */
int id:

if ((id = brdopen(0, 1)) < 0) {
    experror("brdopen");
    return(-1);
}
if (brdioctl(id, BRDADDR, &optaddress) < 0) {
    experror("brdioctl(BRDADDR,...)");
    return(-1);
}
if (brdread(id, optp, sizeof(struct exosopt)) < 0) {
    experror("brdread");
    return(-1);
}

brdclose(id);
return 0;

include ".\simnet.h\simnet.ccd"
include ".\flight.h\flight.ccd"

/* This subroutine computes the rotation matrix (3x3) for the SIMNET PDU's */
/* given the pitch, roll and yaw of the vehicle. */

/*rotation()
int i,j,k=0;
float R,P,Y;
float RC,RS,PC,PS,YC,YS;
float A [3] [3];
float z [3] [3];
float x [3] [3];
float y [3] [3];

/* In Silicon Graphics Dog Fight: Roll=Twist; Pitch=Elevation; Yaw=Azimuth */

R=(plane.twist/10*PI)/180;
P=-(plane.elevation/10*PI)/180;
Y=-(plane.azimuth/10*PI)/180;

RC=cos(R);
RS=sin(R);
PC=cos(P);
PS=sin(P);
YC=cos(Y);
YS=sin(Y);

z[0] [0]=YC;
z[0] [1]=-YS;
z[0] [2]=0;
z[1] [0]=YS;
z[1] [1]=YC;
z[1] [2]=0;
z[2] [0]=0;
z[2] [1]=0;
z[2] [2]=-1;

x[0] [0]=1;
x[0] [1]=0;
x[0] [2]=0;
x[1] [0]=0;
x[1] [1]=PC;
x[1] [2]=-PS;
x[2] [0]=0;
x[2] [1]=PS;
x[2] [2]=PC;

y[0] [0]=RC;
y[0] [1]=0;
y[0] [2]=RS;
y[1] [0]=0;
y[1] [1]=1;
y[1] [2]=0;
y[2] [0]=-RS;
y[2] [1]=0;

for (i=0; i<=2; i++)
    for (j=0; j<=2; j++)
        A [i][j]=0;
    for (k=0; k<=2; k++)
        A[i][j] += x[i][k] * y[k][j];
for (i=0; i<=2; i++) {
    for (j=0; j<=2; j++) {
        pdu_buf.VAPDU.VADATA.rotation[i][j] = 0;
        for (k=0; k<=2; k++)
            pdu_buf.VAPDU.VADATA.rotation[i][j] += A[i][k] * z[k][j];
    }
}
This file is the header file for the airplane running on the SiliconGraphics

---

`#define NAME_LENGTH 15`  
`#define MYPLANEID 16`  
`#define ADJUSTX -350`  
`#define ADJUSTZ 2050`  
`#define AIRPORTX 40000.0`  
`#define AIRPORTY 220.0`  
`#define AIRPORTZ 30000.0`  
`#define P2M 3.281`  
`#define P2M 5.0`

`struct plane {
    long planeid;
    char version; /* flight version */
    char cmd; /* type of packet */
    short type; /* plane type */
    short alive; /* alive */
    char myname[NAME_LENGTH+1];

    unsigned short status; /* for msgs these 2 shorts */
    unsigned short won; /* hold the plane id */
    unsigned short lost;

    float x; /* plane position */
    float y;
    float z;
    short azimuth;
    short elevation;
    short twist;

    short mstatus; /* missile data */
    float mx;
    float my;
    float mz;
    float last_mx;
    float last_my;
    float last_mz;
    long kill;
    float tps;
    int airspeed;
    int thrust;
    short wheels; /* wheel position */
    short elevator; /* elevator position */
    char mtype;
};`

`struct plane plane:`  
`short port=0x140a;`  
`B-42 /* port address for udp/ip connection */`
This file contains the c code to handle the airplane flying on the SG

/* Initialize a synchronous/blocking udp/ip connection for input */
ginitin()

/* Check that the driver is loaded, and get our own ethernet MAC
address from the EXOS board */
if (!loaded()) errexit("driver NOT loaded");
if (ipinfo(&opt) < 0) errexit("could not get own ethernet MAC address");
memcpy(my_addr, opt.xo_eaddr, sizeof(my_addr));

/* Display my address */
fprintf(stderr, "my addr = %02x-%02x-%02x-%02x-%02x-%02x\n",
my_addr[0], my_addr[1], my_addr[2],
my_addr[3], my_addr[4], my_addr[5]);

/* Open input disk file */
diskfd = open(inputfile, FILEOFLAG, FILEPMODE);
if (diskfd < 0) errexit("cannot open diskfile");
fprintf(stderr, "disk file fd = %d\n", diskfd);

/* UDP/IP specification */
send_socket_sg.sin_port = htons(port);
send_socket_sg.sin_addr.s_addr = 0x00000000;
recv_socket_sg.sin_port = htons(port);
recv_socket_sg.sin_addr.s_addr = 0xffffffff;

/* Make a udp socket call */
if (0

/* Read synchronous/blocking udp/ip packet */
greadin()

int cnt;

/* if ((cnt = soreceive(netfdsg, &recv_socket_sg, buf, sizeof(buf))) < 0)
   errexit("soreceive");
fprintf(stderr, "read %d bytes from sg\n", cnt); */
if ((cnt = read(diskfd, buf, 100)) < 0)
   errexit("read");
/* fprintf(stderr, "read %d bytes from disk\n", cnt); */
return(cnt);

/* Close connection */
gfinin()

soclose(netfdsg):
* Network order to host order transform */

ntoh_flight()
{
    int i, j;
    union {
        char *tmpc;
        float *tmpf;
    } tmp;
    union {
        char *tmpc;
        short *tmps;
    } tmps;

tmp.tmpf = &plane.x;
swap4(tmp.tmpc);
tmp.tmpf = &plane.y;
swap4(tmp.tmpc);
tmp.tmpf = &plane.z;
swap4(tmp.tmpc);
tmps.tmps = &plane.azimuth;
swap2(tmps.tmpc);
tmps.tmps = &plane.elevation;
swap2(tmps.tmpc);
tmps.tmps = &plane.twist;
swap2(tmps.tmpc);
}

* Host order to network order transform */

hton_flight()
{
    int i, j;
    union {
        char *tmpc;
        float *tmpf;
    } tmp;
    union {
        char *tmpc;
        short *tmps;
    } tmps;

tmp.tmpf = &plane.x;
swap4(tmp.tmpc);
tmp.tmpf = &plane.y;
swap4(tmp.tmpc);
tmp.tmpf = &plane.z;
swap4(tmp.tmpc);
tmps.tmps = &plane.azimuth;
swap2(tmps.tmpc);
tmps.tmps = &plane.elevation;
swap2(tmps.tmpc);
tmps.tmps = &plane.twist;
swap2(tmps.tmpc);
}

* This subroutine is here for documentation, it is on simnet.ccd */

swap4(char *ptr)
char tmp;

tmp = *ptr;
*ptr = *(ptr+3);
*(ptr+3) = tmp;
tmp = *(ptr+1);
*(ptr+1) = *(ptr+2);
*(ptr+2) = tmp;

/* This subroutine is here for documentation, it is on simnet.ccd */

swap2(char *ptr)

char tmp:

tmp = *ptr;
*ptr = *(ptr+1);
*(ptr+1) = tmp;

display_plane()

fprintf(stderr, "plane id %ld\n", plane.planeid);
fprintf(stderr, "version %c\tcmd %c\stype %d\talive %d\tmname %s\n", plane.version, plane.cmd, plane.type, plane.alive, plane.mname);
fprintf(stderr, "status %ud\twon %ud\tlost %ud\n", plane.x, plane.y, plane.z);
fprintf(stderr, "x %f\ty %f\tz %f\n", plane.x, plane.y, plane.z);
fprintf(stderr, "azimuth %d\televation %d\ttwist %d\n", plane.azimuth, plane.elevation, plane.twist);
fprintf(stderr, "mstatus %d\tx %f\ty %f\tmz %f\n", plane.mstatus, plane.mx, plane.my, plane.mz);
fprintf(stderr, "last_mx %f\tlast_my %f\tlast_mz %f\n", plane.last_mx, plane.last_my, plane.last_mz);
fprintf(stderr, "kill %id\tps %f\n", plane.kill, plane.tps);
fprintf(stderr, "airspeed %d\thrust %d\n", plane.airspeed, plane.thrust);
fprintf(stderr, "wheels %d\televator %d\tmtype %c\n", plane.wheels, plane.elevator, plane.mtype);
/******************************simnet2.h***********************************/

#ifndef SIMNET
/* C2-cf-lf-30-16 */
#define TANKA 0x68 /* 02-cf-lf-30-27-68 */
#define TANKB 0xff95 /* C2-cf-lf-30-27-95 */
#define MCC 0x09 /* 02-cf-lf-30-28-09 */
#define ANZR 0x14 /* 08-00-09-00-ba-14 */
#endif

typedef struct{
  unsigned version :4; /* version of protocol */
  unsigned length :12; /* length of PDU in octets */
  unsigned protocol :8; /* protocol PDU belongs to */
  unsigned kind :8; /* type of PDU within protocol */
} PDUHeader;

/* version field */
#define protocolVersionFeb87 0 /* the Feb. 1987 version of the protocols */
#define protocolVersionNov87 1 /* the Nov. 1987 version of the protocols */

/* protocol field */
#define protocolNone 0 /* no protocol -- PDU used for padding */
#define protocolMgmt 1 /* the Network Management Protocol */
#define protocolSim 2 /* the Simulation Protocol */
#define protocolData 3 /* the Data Collection Protocol */
#define protocolXfer 4 /* the File Transfer Protocol */
#define protocolDiag 5 /* the Diagnosis Protocol */

/* kind field */
#define activatePDUKind 1 /* Activate PDU */
#define activatingPDUKind 2 /* Activating PDU */
#define deactivatePDUKind 3 /* Deactivate PDU */
#define vehicleAppearancePDUKind 4 /* Vehicle Appearance PDU */
#define UNUSED 5 /* Unused PDU */
#define vehicleImpactPDUKind 6 /* Vehicle Impact PDU */
#define groundImpactPDUKind 7 /* Ground Impact PDU */
#define indirectFirePDUKind 8 /* Indirect Fire PDU */
#define serviceRequestPDUKind 9 /* Service Request PDU */
#define resupplyOfferPDUKind 10 /* Resupply Offer PDU */
#define resupplyReceivedPDUKind 11 /* Resupply Received PDU */
#define repairPDUKind 12 /* Repair PDU */
#define repairedPDUKind 13 /* Repaired PDU */
#define collisionPDUKind 14 /* Collision PDU */
#define firePDUKind 15 /* Fire PDU */
#define radiatePDUKind 16 /* Radiate PDU */
#define resupplyCancelPDUKind 17 /* Resupply Cancel PDU */

/* Vehicle Type Identifier Field */
#define vehMainBattleTank 1 /* M1 or T72 main battle tank */
#define vehPersonnelCarrier 2 /* M2, M3 or BMP */
#define vehCommandPost 3 /* M577 Command Post */
#define vehAmmunitionTruck 4 /* M977 Ammo Truck */
#define vehFuelTruck 5 /* M978 Fuel Truck */
#define vehSupplyTruck 6 /* M35-A2 Truck */
#define vehMortarCarrier 7 /* M106 Carrier */
#define vehSPHowitzer 8 /* M109 Howitzer */
#define vehRecoveryVehicle 9 /* M88 Recovery */
#define vehFISTVehicle 10 /* Fire Support */
typedef struct {
PDUHeader pduHdr;  /* version, length, protocol, PDUkind */
unsigned char exerciseID; /* exercise identifier */
unsigned char padding;  /* vehicle identifier */
unsigned short vehicleID; /* vehicle identifier */
} SimPDUHeader;

typedef struct {
unsigned char role;  /* role of vehicle: ammo truck, fuel truck, etc */
unsigned char battalion; /* battalion (task force) vehicle belongs to */
unsigned char company; /* company (team) vehicle belongs to */
unsigned char bumper; /* bumper number within company */
} VehicleRole;

#define roleSimulator 0 /* a vehicle operated by a full crew, simulated by a crewed vehicle simulator */
#define roleOPFOR 1 /* a vehicle simulated by a Semi-automated Forces system */
#define roleGunneryTarget 2 /* a gunnery target, such as that simulated by an MCC system */
#define roleAmmoTruck 3 /* an ammunition truck, such as that simulated by an MCC system */
#define roleFuelTruck 4 /* a fuel truck, such as that simulated by an MCC system */
#define roleMaintTeam 5 /* a maintenance team, such as that simulated by an MCC system */
#define roleS2 6 /* a batallion S2’s vehicle, such as that simulated by an MCC system as part of a tactical operations center (TOC) */
#define roleS3 7 /* a batallion S3’s vehicle, such as that simulated by an MCC system as part of a TOC */
#define roleFSE 8 /* a batallion fire support officer’s vehicle, such as those simulated by an MCC system as part of a TOC */
#define roleTACP 9 /* a batallion tactical air control party vehicle, such as those simulated by an MCC system as part of a TOC */
#define roleAdminLogCenter 10 /* a batallion admin/log center vehicle, such as that simulated by an MCC system */
#define roleOther 99 /* any other vehicle not in one of the above categories */

#define assignedRattalion 1 /* the vehicle is assigned to no unit in particular within the battallion */
#define assignedScoutPlt 2 /* the vehicle belongs to the batallion’s scout platoon */
#define assignedTACP 3 /* the vehicle belongs to the batallion’s tactical air control party */
typedef struct SimPDUHeader hdr; /* include ID of described number */

/* Common to all vehicles */
VehicleRole role; /* include ID of described number */
unsigned char alignment; /* offense, defense, friend, or foe */
unsigned char vehicleClass; /* class of vehicle */
/* unsigned short appearance; /* type of vehicle and appearance */
/* struct */
unsigned vehKindMask : 6;
unsigned un1 : 1;
unsigned vehDestroy : 1;
unsigned vehSmokePlume : 1;
unsigned vehFlaming : 1;
unsigned vehDustCloudMask : 2;
unsigned un2 : 1;
unsigned vehTOWLauncherUp : 1;
unsigned vehEngineSmoke : 1;
unsigned un3 : 1;
appearance; /*
struct {
unsigned vehSmokePlume : 1;
unsigned vehFlaming : 1;
unsigned vehDustCloudMask : 2;
unsigned un2 : 1;
unsigned vehTOWLauncherUp : 1;
unsigned vehEngineSmoke : 1;
unsigned un3 : 1;
appearance;
float rotation [3][3]; /* vehicle rotation */
float location [3]; /* exact vehicle location */
short grid [2]; /* approximate vehicle location */
unsigned short engineSpeed; /* engine speed, in RPM */
/* unsigned short padding; */
unsigned short sequence; /* sequence # for vehicleAppearancePDU */
/* Depending on vehicle class */
union {

/* If a simple moving vehicle, without turret ... */
struct {
float velocity [3]; /* velocity (m/sec/15) */
}
simple;
/* If a tank */
struct {
float velocity [3]; /* velocity (m/sec/15) */
unsigned short turretAzimuth; /* turret/hull orinntation */
unsigned short gunElevation; /* gun/turret elevation */
}
tank;
}
VehicleAppearancePDU;
/* alignment field */
define alignedFoe 0 /* the vehicle appears unfriendly to all participants */

#define alignedOffense 1 /* the vehicle is on the offense team */
#define alignedDefense 2 /* the vehicle is on the defense team */
#define alignedFriend 3 /* the vehicle appears friendly to all participants */

* vehicle class field */
#define vehicleClassStatic 1 /* the vehicle is always stationary when visible, and it has no independently movable parts */
#define vehicleClassSimple 2 /* the vehicle can move, but it has no independently movable parts */
#define vehicleClassTank 3 /* the vehicle can move, and it has a turret and a gun barrel */

typedef struct {
    unsigned char ammunition; /* type of ammunition fired */
    unsigned char fuze; /* type of fuze used */
    unsigned char quantity; /* number of rounds in burst */
    unsigned char rate; /* rate of fire, rounds per second */
} BurstDescriptor;

/* ammunition field */
define ammoHEI25 1 /* 25 mm high explosive incendiary shell */
define ammoHEAT105 2 /* 105 mm high explosive anti-tank shell */
define ammoAPDS25 3 /* 25 mm armor piercing discarding sabot shell */
define ammoAPDS105 4 /* 105 mm armor piercing discarding sabot shell */
define ammoTP25 5 /* 25 mm target practice shell */
define ammoBomb500 6 /* 500 lb. bomb */
define ammoHE107 7 /* 107 mm (4.2in.) high explosive mortar shell */
define ammoHE155 8 /* 155 mm high explosive howitzer shell */
define ammoMissileTOW 9 /* TOW anti-tank missile */

/* fuze field */
define fuzePointDetonating 1 /* point detonating fuze */
define fuzeProximity 2 /* proximity fuze */

typedef struct {
    unsigned char targetType:2; /* what is known about the target */
    unsigned :14;
    unsigned short vehicleID; /* ID of target vehicle, if known */
} TargetDescriptor;

/* targetType field */
define targetUnknown 0 /* the target vehicle is not known */
define targetNotVehicle 1 /* the target is known, but it is not a vehicle */
define targetVehicle 2 /* the target is known and it is not a vehicle */

*/ */
define MYTANKTD 16
define MAXBUF 8192
#define HEADER_SIZE 14 /* ethernet header size including our header */

struct ether { /* first three fields required for any link level packet */
    char e_ghost[6]; /* 00-05 ethernet destination */
    char e_shost[6]; /* 06-11 ethernet source */
    short e_type; /* 12-13 ethernet packet type */
}
struct {
    short e_data_length; /* 14-15 user data length */
    char e_data[1512-HEADER_SIZE]; /* 16-1512 data, max size is 512 */
    simnet_data;
};

Union {
    struct {
        unsigned length :12;
        unsigned version :4;
    } i_data_length;
    short p_data_length;
    data_length;
}

typedef union {
    char DATAONLY [1512-HEADER_SIZE];
} DATAONLYPDU;

struct {
    PDUHeader ANYHDR;
    char data [1512-HEADER_SIZE 4];
} ANYPDU;

struct {
    VehicleAppearancePDU VADATA;
} VAPDU;

#define MAXPKTSIZE 1514 /* total size of largest possible packet */
* char send_addr[6]; /* our ethernet MAC address */
* char recv_addr[6]; /* his ethernet MAC address */
char my_addr[6]; /* my ethernet MAC address */
struct exosopt opt; /* EXOS board options include own address */
#define ETYPES htons(0x5208) /* arbitrary unused ethernet type */
#define HELICOPTER1111
#define HELICOPTER1212
#define AIR 13
DU pdu_buf;
struct ether ether_buf;
This file contains the C code for the simnet M1 tank simulator.

```c
/* Initialize the synchronous/non-blocking link-level socket connection */
jetinit()

int rc, on=1;

/* Check that the driver is loaded, and get our own ethernet MAC */
address from the EXOS board */
if (!(loaded())) errexit("driver NOT loaded");
if (ipinfo(&opt) < 0) errexit("could not get own ethernet MAC address");
memcpy(my_addr, inet.xo_eaddr, sizeof(my_addr));

/* Display my address */
fprintf(stderr, "my_addr = \%x-%02x-%02x-%02x-%02x-%02x\n",
my_addr[0], my_addr[1], my_addr[2],
my_addr[3], my_addr[4], my_addr[5]);

/* Initialize the simnet receiver/sender socket type */
recv_socket.sl_types[0] = ETYPE;

/* Make a link level socket call */
if ((netfd=socket(SOCK_ETH, (struct sockaddr *)0, &recv_socket, 0)) < 0) {
    if (errno == EACCES)
        errexit("link-level access must be enabled with -l option on netlo-
else errexit("cannot create socket");
    fprintf(stderr, "socket fd = ", netfd);
}

/* Synchronous/non-blocking mode */
soioctl(netfd, STOSLINGER, &on);
rc = so ioctl(netfd, FIONBIO, &on);
if (rc < 0) {
    experror("so ioctl...FIONBIO, &on");
    return(-1);
}
return(0);

/* Read synchronous/non-blocking mode packet */
* netread (struct ether buf) */
netread()

int cnt;

cnt = itereceive(netfd, (struct sockaddr *)0, ether_buf, MAXPKTSIZE);
if ((cnt < 0) && (errno == EWOULDBLOCK))
    /* No network data */
else
    if (cnt == 0) experror("soreceive read error"); /* Error condition */
    return (cnt);

/* Write synchronous/non-blocking mode packet */
* netwrite (struct ether *buf) */
```
netwrite()

int cnt, netcnt:

data_length.p_data_length = ntohs(ether_buf.simnet_data.e_data_length);

cnt = data_length.i_data_length.length;

netcnt = sosend(netfd, (struct sockaddr *)0, &ether_buf, cnt + HEADER_SIZE);

if ((netcnt < 0) && (errno == EWOULDBLOCK)) netcnt = 0;

if (netcnt < 0)
    errexit("sosend write error");

else
    if ((netcnt >= 0) && (netcnt < cnt))
        fprintf(stderr, "sosend: some data has been lost
\007")

* Close synchronous/non blocking socket connection */
netfini()

int off = 0;

if (netfd == -1)
    fprintf(stderr, "Please wait up to %d seconds for completion\n",
            timelimit);

select(netfd, FIONBIO, &off);

close(netfd);

netfd = -1;

* Network order to host order transform, not all of the data field are included yet. Add more statements if needed and modify the hton simnet() too */

htoh_simnet(pdu_buf)

htoh_simnet()

int i, j;

union {
    char *tmpu;
    unsigned short *tmpui;
} tmpui;

union {
    char *tmpc;
    float *tmpf;
} tmp;

tmp.tmpf = &pdu_buf.VAPDU.VADATA.location[0];
    swap4(tmp.tmpfc);

tmp.tmpf = &pdu_buf.VAPDU.VADATA.location[1];
    swap4(tmp.tmpfc);

tmp.tmpf = &pdu_buf.VAPDU.VADATA.location[2];
    swap4(tmp.tmpfc);

tmpui.tmpui = &pdu_buf.VAPDU.VADATA.hdr.vehicleID;
    swap2(tmpui.tmpui);

for (i = 0; i <= 2; i++)
    for (j = 0; j <= 2; j++) {
        tmp.tmpf = &pdu_buf.VAPDU.VADATA.rotation[i][j];
    swap4(tmp.tmpfc);

return(pdu_buf.ANYPDU.ANYHDR.kind);

B-5
Host order to network order transform, not all of the data field are included yet. Add more statements if needed and modify the ntoh_simnet() too */

hton_simnet (struct PDU buf) */

hton_simnet ()

int i, j;
union {
    char *tmpc;
    unsigned short *tmpui;
} tmpui;
union {
    char *tmpc;
    float *tmpf;
} tmp;

tmp.tmpf = &pdu.buf.VAPDU.VADATA.location[0];
swap4(tmp.tmpc):
tmp.tmpf = &pdu.buf.VAPDU.VADATA.location[1];
swap4(tmp.tmpc):
tmp.tmpf = &pdu.buf.VAPDU.VADATA.location[2];
swap4(tmp.tmpc):
tmpui.tmpui = &pdu.buf.VAPDU.VADATA.hdr.vehicleID;
swap2(tmpui.tmpc);
for (i=0; i<=2; i++)
    for (j=0; j<=2; j++):
        tmp.tmpf = &pdu.buf.VAPDU.VADATA.rotation[i][j]:
        swap4(tmp.tmpc):

return(0);

* This subroutine does the same work as ntohl(), htonl(). */
swap4/char *ptr;

char tmp;

tmp = *ptr;
*ptr = *(ptr+1):
*(ptr+3) = tmp;
tmp = *(ptr+1):
*(ptr+1) = *(ptr+2):
*(ptr+2) = tmp;

* This subroutine does the same work as ntohs(), htons(). */
swap2/char *ptr;

char tmp;

tmp = *ptr;
*ptr = *(ptr+1):
*(ptr+1) = tmp;

* This subroutine is for debugging purpose only, it will DUMP the content of a
  link level packet in hexadecimal*/
*dump ether (struct ether ether_buf) */
dump ether ()

int i, j, netcnt;
```c
iprintf(stderr,"ETH	R	content\n");

data	len = ntohs (ether_buf.simnet_data.e_datalen);

fprintf(stderr,"Source addr ": ether_buf.e_host[0], ether_buf.e_host[1], ether_buf.e_host[2], ether_buf.e_host[3], ether_buf.e_host[4], ether_buf.e_host[5]);

fprintf(stderr,"Destination addr ": ether_buf.e_dest[0], ether_buf.e_dest[1], ether_buf.e_dest[2], ether_buf.e_dest[3], ether_buf.e_dest[4], ether_buf.e_dest[5]);

netcnt = data	len = ntohs (ether_buf.simnet_data.e_datalen);

for (i=0, j=3; i<(netcnt-HEADER_SIZE-2); i++, j++) {
    fprintf(stderr,"%2x ", ether_buf.simnet_data.e_data[i]);
    if (j >= 17)
        j=0;
    fprintf(stderr,"\n");
}

fprintf(stderr,"\n");

* This subroutine is for debugging purpose only, it will DUMP the content of a
dpu packet in hexadecimal */

dump_pdu ()

int i, j, netcnt;

fprintf(stderr,"PDU content\n");
data	len = ntohs (ether_buf.simnet_data.e_datalen);

for (i=0, j=1; i<(netcnt-HEADER_SIZE-2); i++, j++) {
    fprintf(stderr,"%2x ", pdu_buf.DATAONLYPDU.DAT

for (j=0; j<=27; j++)
    fprintf(stderr, "%d %d
", i, j, pdu_buf.VAPDU.VADATArotation[i][j])

fprintf(stderr, "Location\n");

for (i=0; i<=2; i++)
    fprintf(stderr, "%1f
", i, j, pdu_buf.VAPDU.VADATA.location[i])

fprintf(stderr, "%s\n", pdu_buf.VAPDU.VADATA.hdr.vehicleID);
```

R-54