1-1-1990

ASAT to SIMNET Protocol Translator : Hardware And Software Description

Jorge Cadiz

INSTITUTE FOR SIMULATION AND TRAINING

Contract Number N61139-89-C-0043
PM TRADE
DARPA

June 20, 1990

ASAT to SIMNET Protocol Translator

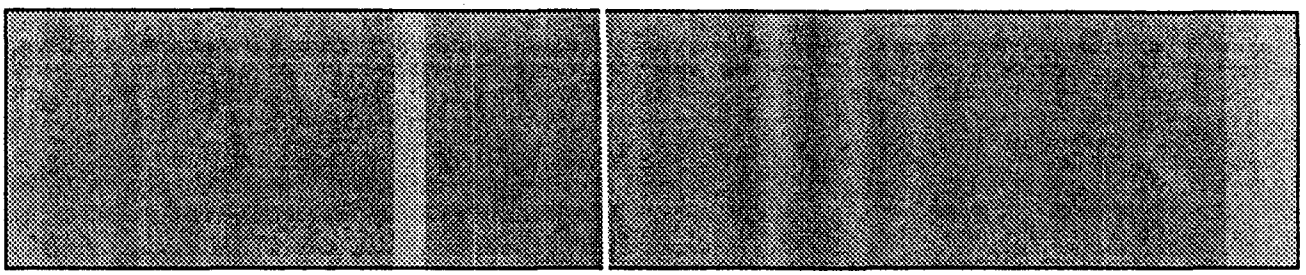
Hardware and Software Description

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

University of Central Florida
Division of Sponsored Research

IST-TR-90-10

INSTITUTE FOR SIMULATION AND TRAINING



Contract Number N 61139-89-C-0043
PM TRADE
DARPA

June 20, 1990

ASAT to SIMNET Protocol Translator

Hardware and Software Description

Jorge Cadiz
Gilbert Gonzalez
Ruey Ouyang
Michael Ruckstuhl

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

University of Central Florida
Division of Sponsored Research

IST-TR-90-10

Table of Contents

Hardware and Software Description of the ASAT to SIMNET Protocol Translator

Technical Report No. IST-TR-90-10

UNIVERSITY OF CENTRAL FLORIDA

INSTITUTE FOR SIMULATION AND TRAINING

NETWORKING AND COMMUNICATIONS TECHNOLOGY LABORATORY

1. Introduction

The Institute for Simulation and Training (IST) has procured two Avionics Situational Awareness Trainers (ASATs) from Perceptronics, Inc. Perceptronics, in conjunction with Sphere, Inc. are the developers of the ASAT flight trainers. The ASATs are low-cost F-16 cockpit trainers that stimulate the trainee with visual cues and beyond visual range (BVR) radar signals. These modules are connected together via an ETHERNET network. They are capable of fighting against one another, or fighting as a team against computerized targets which are generated by the master ASAT system.

Interconnecting these flight trainers to the SIMNET network resident in IST's laboratory became a primary goal for demonstrating the interconnection of dissimilar simulators. The approach to this task was to develop a network protocol translator which would convert ASAT network packets to SIMNET format and vice versa.

The ultimate goal for this task is to design a Generic Protocol Translator (GPT) that will accept a packet from any networkable simulator, including SIMNET, and translate it to the Standard Protocol for Interoperable Simulations. The standard protocol is currently being developed by IST.

The GPT must be able to handle simulations of various fidelities and update rates. It must also be able to interconnect synchronous simulators with asynchronous simulators. All of these tasks must be performed in a propitious manner so as to not adversely affect the time sensitive requirements of the real time distributed simulation environment.

2. APPROACH

In order to translate a simulator's packets, we connected a PC onto the Simulation (SIMNET) ETHERNET network to act as a protocol translator. This PC copies the ASAT packets from the network, performs the necessary protocol translations and places the new packets onto the ETHERNET.

Prior to this exercise, we had performed a similar experiment in interfacing between SIMNET M1's and a networkable flight demo program running on a Silicon Graphics (SG) Workstation [1]. The concepts which were used in the SG—>SIMNET interconnection were adapted and implemented in the ASAT —>SIMNET interconnection exercise.

System	Type of PDU	Function
SIMNET	Vehicle Appearance	Gives the vehicle's appearance, location, attitude, ID, role, etc.
	Activate	Sent from MCC to Simulator. Provides vehicular information for initialization.
	Activating	Broadcast from simulator to the network to announce beginning of activation.
	Fire	Broadcast by firing simulator to announce the firing of his weapon.
	Ground Impact	Broadcast to announce that round has hit the ground.
	Vehicle Impact	Announces that a round has hit a vehicle.
ASAT	Type 0	Performs handshaking between two entities.
	Type 1	Initializes the aircraft.
	Type 8	Broadcasts position, orientation, and status of simulator.
	Type 9	Broadcasts position, orientation, and status of the computer generated targets.

Table 1. SIMNET and ASAT PDUs

The preliminary activities for development of this protocol translator consisted of studying the formats of the SIMNET protocol data units (PDUs) and the ASAT PDUs. A summary of these formats are outlined in Section 5 of this document. The PDUs which were of main interest for the ASAT—>SIMNET interconnection are shown in Table 1.

The PDU of most interest is the Vehicle Appearance PDU (VA PDU). This is the packet which is used to pass the location and appearance information of one node (simulator) to the rest of the nodes on the network.

The following hardware and software descriptions of the interconnection project between the ASAT and the SIMNET units focuses on the translation of the ASAT Appearance PDU (packet type 8) to the SIMNET Vehicle Appearance PDU format.

3. HARDWARE DESCRIPTION

3.1 ASAT Hardware

The ASATs are low-cost cockpit trainers which assist in the training of an F-16A fighter pilot in Air-to-Air Beyond Visual Range tactics.

Perceptronics uses commercial components, supplemented with their own hardware designs. The main modules which comprise the ASAT trainers are:

- Computer system
- Heads-Up Display (HUD)/Out-the-window Graphics System
- Multi-function display
- Hands-on stick and throttle controls
- Cockpit enclosure with inclined seat

- Aural cue system
- Communication system
- Local Area Network (LAN)

We will briefly discuss the computer system hardware, graphics and HUD system, and the networking hardware for the ASAT Trainers.

3.1.1. Computer System

The ASAT computer system consists of high performance, single-board microprocessors. The host computer is a PC-AT chassis with an Intel 20 Mhz 80386 CPU and an 80387-20 co-processor which drives both the HUD and the radar display. An Intel 80286-based PC-AT computer drives the cockpit multi-function display [2].

3.1.2. Graphics and HUD System

The Computer Image Generator in the ASAT is manufactured by XTAR Corporation. It is a four board set which includes a CPU, memory processor, and array processor. The CIG produces high resolution 1024 x 1024 graphics for the HUD symbology and the out-the-window visual. It is installed directly into the 386 PC-AT host computer [2].

3.1.3. Networking Hardware

The ASAT network will support up to 12 simulators in a single exercise. The networking hardware and cabling is based on IEEE 802.3 specifications; however, the protocols used are not standard ETHERNET protocols. This inconsistency will be discussed further in the ASAT PDU description section of this document. The network adapter used in the ASAT system is an Etherlink II ETHERNET board from 3Com Corporation and is installed in the 386 PC-AT.

3L Routines	3L I/F Routines
InitAdapters	cInitAdapters
InitParameters	cInitParameters
ResetAdapter	cResetAdapter
WhoAmI	cWhoAmI
RdRxFilter	cRdRxFilter
WrRxFilter	cWrRxFilter
SetLookAhead	cSetLookAhead
PutTxData	cPutTxData
GetRxData	cGetRxData
Protocol Side Routines	**Protocol Side Interface Routines**
myTxProcess	TxProcess
myExitRcvInt	ExitRcvInt
myRxProcess	RxProcess

Table 2. Software Routines

3.2. Protocol Translator Hardware

We will also cover the hardware specifications of the Protocol Translator. The Protocol Translator consists of a Hewlett Packard Vectra PC/AT compatible with an 80386 processor that operates at 20 Mhz. Installed in the computer is a 3Com Etherlink II Network Adapter. The Etherlink II is a high-performance

network interface which links PCs to IEEE 802.3 ETHERNET networks. Some of the other characteristics of the Etherlink II board are on-board transceiver, 8 Kbytes of dual ported RAM, and a choice of host interfaces: shared memory, DMA, or programmed I/O.

4. SOFTWARE DESCRIPTION

This section describes the software routines developed for the ASAT to SIMNET interconnection project. The software runs on the Protocol Translator, and performs the necessary functions to capture, translate, and transmit an ASAT Appearance PDU (type 8) to a SIMNET Vehicle Appearance PDU.

The translator software is implemented with C programming language and assembly language routines. These routines will serve as software templates for the future development of a generic protocol translator, which will ultimately allow the interconnection of any networkable simulator to SIMNET. The Software Description is divided into three sections. The first section provides a flow chart of the translator software routines. The second section focuses on the network software, including the functions that provide an interface to the 3Com 503 Link Level Library. The third section focuses on the translation routines developed by IST for performing the actual protocol translation.

4.1. Translator Software Flow Diagram

Before going into a detailed description of the individual software modules, we provide a software flow diagram which assists in understanding the internal workings of the translator. Figures 1a and 1b illustrate how the translator software functions. Functional hierarchy and modular hierarchy diagrams are provided in Appendix B.

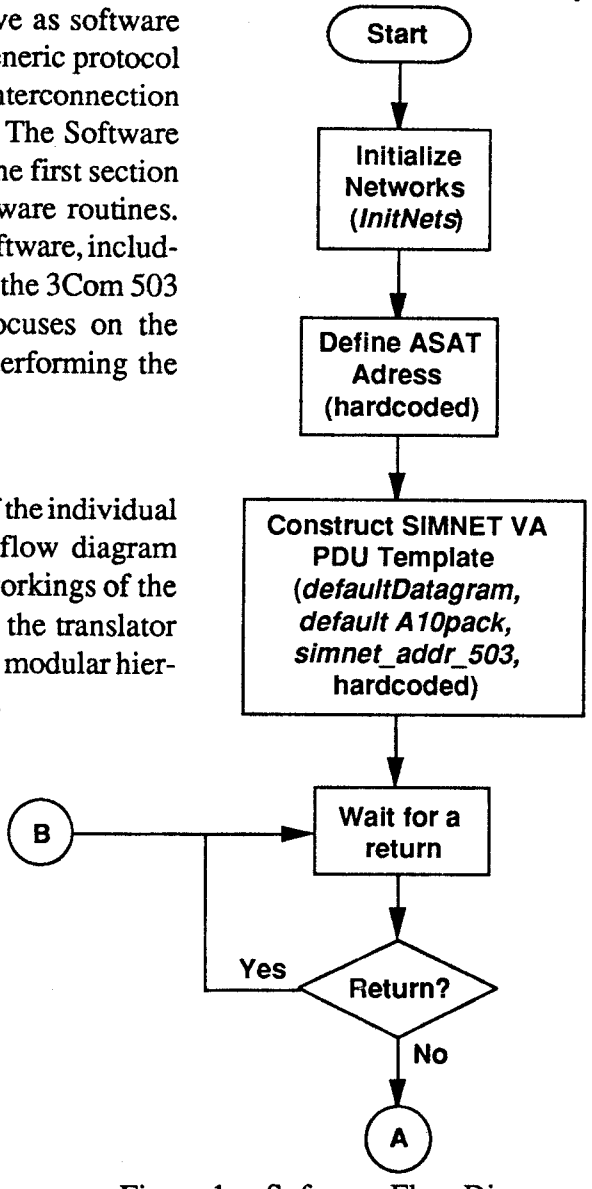


Figure 1a. Software Flow Diagram

4.2. Network Software

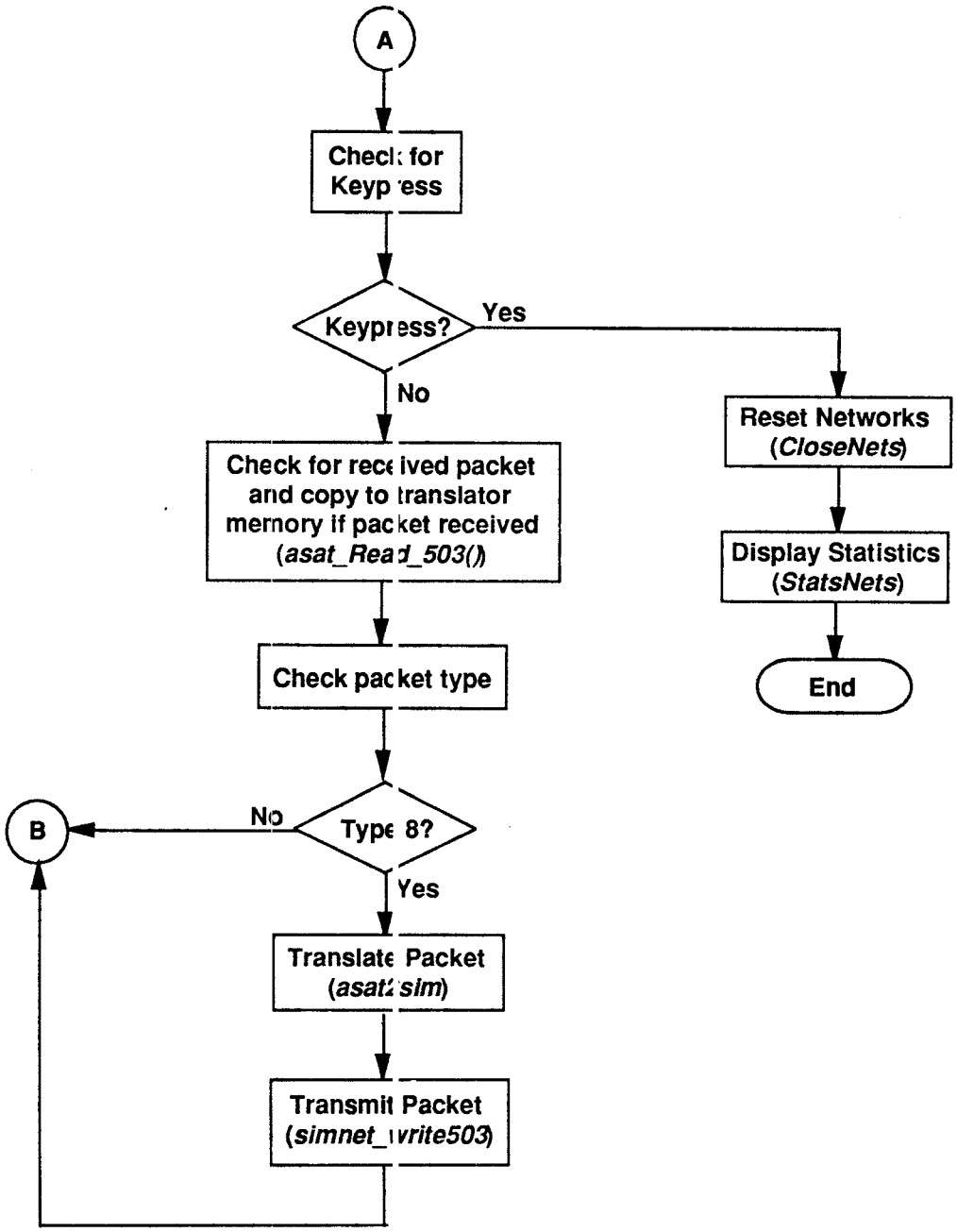
4.2.1. 3Com Library and Low-Level Interface Routines

This section describes the interface to the 3Com Network Adapter, which is being used in the ASAT—>SIMNET Protocol Translator. The library routines were developed by 3Com Corporation and are copyrighted. 3Com also supplies a set of assembly routines which provide a high-level interface to their own library func-

tions. With these interfaces, the 3Com Network Board can be accessed by a high-level language (C). These interfaces are found in the file NETTO3L.ASM; the 3Com routines are in the library file 503.LIB. The relationship between the interface routines and the 3Com code is shown in Table 2. The 3L Assembly Routines and the 3L Interface Routines are Link Level Library (3L) routines.

Figures 2 and 3 were extracted from the Link Level Library (3L) Interface Specification

Figure 1b. Software Flow Diagram (con't)



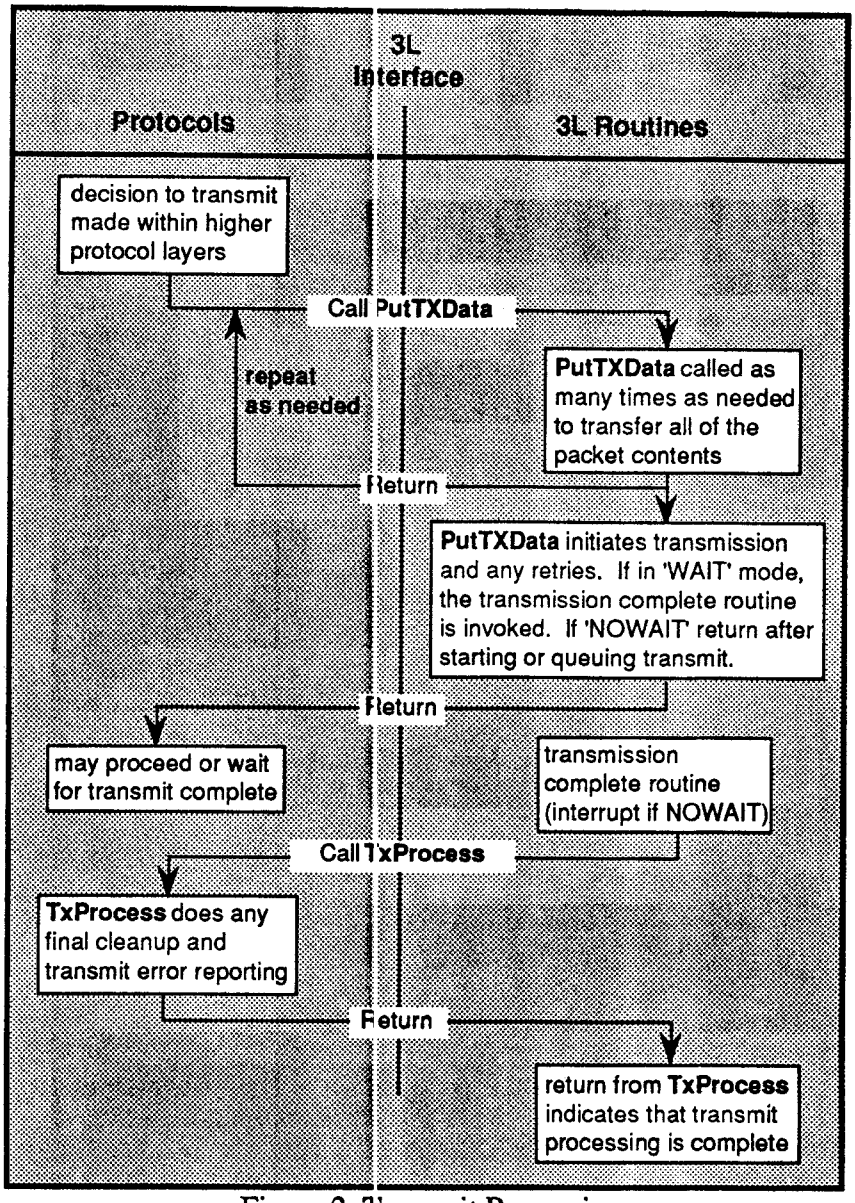


Figure 2. Transmit Processing

[3]. These figures provide a graphical representation of the transmit and receive processing which takes place at the board level and are applicable to both the IST protocol translator and the ASAT ETHERNET interface.

4.2.1.1. cInitParameters()

cInitParameters calls InitParameters to set up the values used to initialize the 3Com Etherlink II adapter hardware. This routine uses the request header provided by DOS (from the CONFIG.SYS file). If a DOS device driver is not being used (as in the current configuration), a DOS INIT header must be simulated. The address of this simulated header is passed

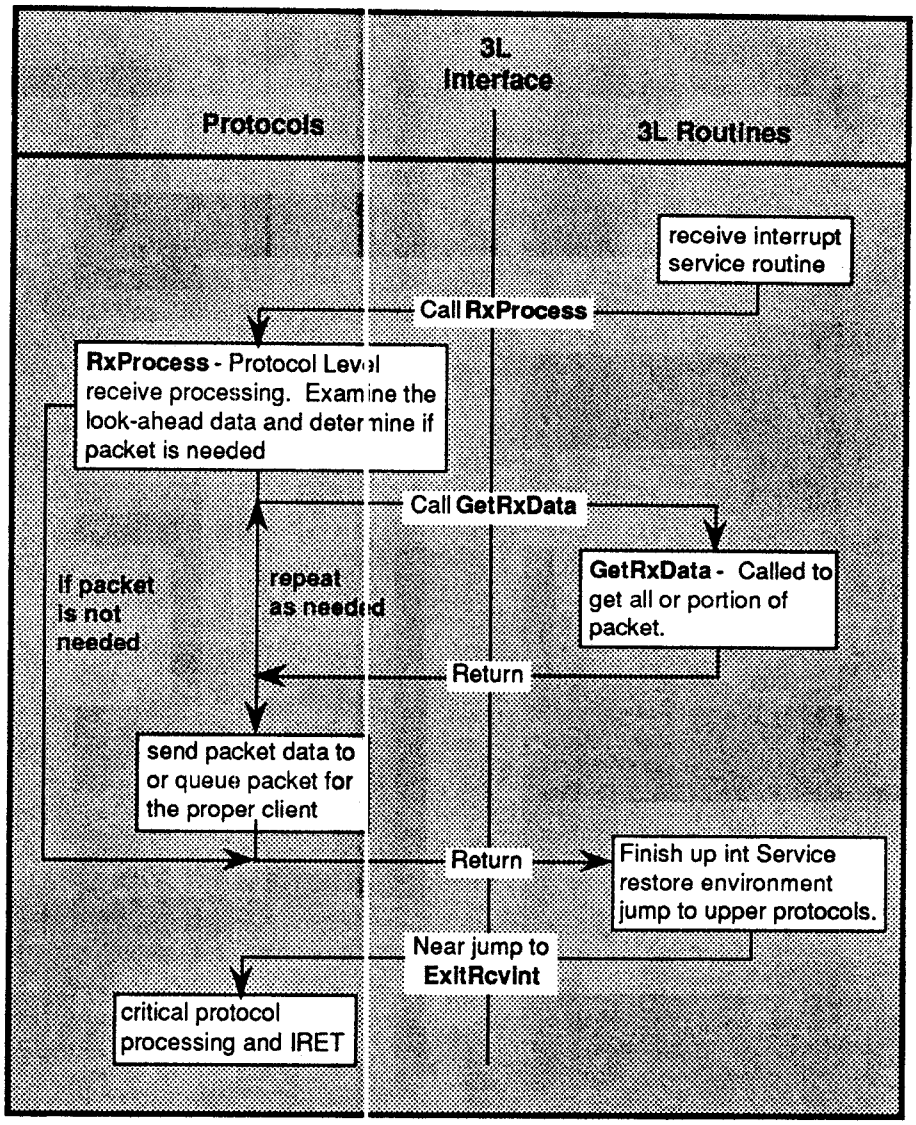


Figure 3. Receive Processing

to the function. This function, like all of the 3Com 3L functions, returns a completion code which specifies any errors that might have occurred.

4.2.1.2. cInitAdapters()

cInitAdapters calls InitAdapters to initialize the network adapter hardware. After this routine is called, the adapter hardware is ready for use.

4.2.1.3. cResetAdapter()

cResetAdapter calls ResetAdapter to reset the adapter in the case of a catastrophic failure. The adapter hardware is reset to the state that normally is set after the InitAdapters call.

4.2.1.4. cWhoAmI()

cWhoAmI calls WhoAmI to provide a hardware status report. This status report includes detailed data about the network-specific identification of the adapter. The routine returns the address of a data structure containing relevant information.

4.2.1.5. cRdRxFilter()

cRdRxFilter calls RdRxFilter to read the current packet address filter. This address filter specifies whether the adapter will accep: all packets, broadcast packets, multi-cast packets, packets addressed to the adapter only, or no packets.

4.2.1.6. cWrRxFilter()

cWrRxFilter calls WrRxFilter to set the packet address filter to the specified value.

4.2.1.7. cGetRxData()

cGetRxData is used within myRxProcess to call GetRxData to transfer the received packet from the 3Com board to a specified location in PC memory. The received packet may be copied in as many transfers as needed, because the amount of data transferred at any one time may be specified. This routine also releases buffer space on the 3Com board.

4.2.1.8. RxProcess()

RxProcess is called to process received packets. When the board receives a packet, an interrupt is generated. The interrupt handler calls RxProcess, which in turn calls myRxProcess to process the packet data. myRxProcess is provided with a limited amount of packet header information in order to decide what to do with the packet. Subsequently, myRxProcess may call cGetRxData to either copy the packet to memory, or to release the board's buffer space. Note that myRxProcess is never called by the protocol code.

4.2.1.9. ExitRcvInt()

ExitRcvInt is called to perform basic packet processing immediately following packet reception. Upon reception of a packet, the 3Com routines call RxProcess to perform the incoming packet processing. Once RxProcess has returned to the 3Com routines, the 3Com routines will call ExitRcvInt, which could call myExitRcvInt for last-minute packet handling. myExitRcvInt may be a simple IRET, a return from interrupt. This routine is never called by the protocol code.

4.2.1.10. cSetLookAhead()

cSetLookAhead calls SetLookAhead, which is used to specify the amount of packet header information (in bytes) that will be provided for myRxProcess.

4.2.1.11. cPutTxData()

cPutTxData calls PutTxData, which is used to copy a section of a packet to the 3Com board. On the last call to PutTxData (the last transfer to complete the packet on the board),

PutTxData specifies (1) whether the PC will wait for the transmission process to be completed, or (2) if an interrupt must be generated to announce that the packet transmission is completed. If the interrupt announce method is chosen, PutTxData returns, and TxProcess will be called as a result of the interrupt (when transmission is completed). Otherwise, the last call to PutTxData does not return control to the calling routine until transmission is complete.

4.2.1.12. TxProcess()

TxProcess is used to process the end of the transmission of packets. After the packet has been successfully transmitted, TxProcess may be called. TxProcess calls myTxProcess for any after-transmission processing. TxProcess is never called by the protocol process.

4.2.1.13. Procedure Call Categories

In this section we categorize the software procedures by in the following manner: initialization, control, transmission, and reception. The time at which the various software functions are called is given in brackets.

Initialization:	InitParameters	[called first]
	InitAdapters	[called second]
	ResetAdapter	[for catastrophic error only]
	WhoAmI	[for current Status information, anytime]
Control:	WrRxFilter	[anytime]
	RdRxFilter	[anytime]
Transmission:	PutTxData	[repeat until packet completely transferred to the board]
	myTxProcess	[automatically called upon completion of transmission]
Reception:	SetLookAhead	[anytime before packet is received]
	myRxProcess	[automatically called upon reception of a packet]
	GetRxData	[called from myRxProcess]
	myExitRcvInt	[automatically called after myRxProcess]

4.2.2. Functions Defined in STAMP.ASM

This file contains miscellaneous assembly functions. The primary purpose of these functions is to establish timestamping capabilities for system performance statistics.

4.2.2.1. savvecs()

This function saves the interrupt vector table. This is important because some of the 3Com functions modify the table and do not restore it.

4.2.2.2. fixvecs()

This function restores the interrupt vector table to the state it was in before savvecs() was called.

4.2.2.3. getstamp()

This function returns the value in the hardware Counter 1. This counter decrements each system clock cycle (every 838 ns). When the hardware counter reaches 0, it rolls over to FFFF (hex) and generates an interrupt. This Real-Time Interrupt is generated each time the counter rolls over (every 55 ms). The RTI also drives the MS-DOS clock. getstamp is used to derive performance statistics for the translator.

4.2.2.4. getclock()

This function gets the lower word of the MS-DOS clock. This information is used in conjunction with the clock cycle count returned from getstamp() to keep track of time segments larger than 55 ms. Note that the MS-DOS clock counts up, while the clock cycle counter counts down.

4.2.3. Functions Defined in 3COM.C

This file provides the interrupt routines that the 3L interface functions call during an interrupt. This file also provides a simple interface to the 3Com adapter object.

4.2.3.1. init3COM()

This function calls several 3L functions to initialize the adapter.

4.2.3.2. addr_3COM()

This function returns a pointer to the Ethernet address of the 3Com adapter.

4.2.3.3. reset3COM()

This function calls cResetAdapter to reset the adapter in the case of a catastrophic error.

4.2.3.4. stats3COM()

This function displays on the screen a limited amount of network and adapter statistics.

4.2.3.5. myRxProcess()

This function is called by RxProcess(), and performs the interrupt packet processing. The current configuration includes one buffer. All new packets are placed in this buffer, overwriting any old packets. In addition, a timestamp is saved with the new packet. Several rudimentary calculations are performed for a later display of network statistics. The length of the packet is also saved.

4.2.3.6. myTxProcess()

This function is called by TxProcess(), and currently performs some rudimentary statistics calculations.

4.2.3.7. myExitRcvInt()

This function is not included in IST's implementation, but it is described here for completeness. This function could provide last-chance processing, and possibly save a significant amount of time from not having to wait until control is passed back from the interrupt.

4.2.3.8. cXmit1()

This function calls cPutTxData in order to transmit the given packet.

4.2.3.9. cRcvSome()

This function is passed the address of a pointer. cRcvSome() sets the pointer to the buffer that myRxProcess uses. The pointer returns the length of the packet in the buffer (0 if no packet), and the length of the packet is reset to zero so that the same packet is not read twice by mistake. It is suggested that all calls to cRcvSome be performed with interrupts disabled, so that the calling routine has a chance to copy the packet in the buffer before another packet can be copied into the buffer by myRxProcess.

4.2.4. Functions Defined in NETS.C

This file provides global network access functions. In the current configuration this file is redundant; however, if the configuration is expanded to a multiple-network configuration, the functions provided by this file may become necessary.

4.2.4.1. InitNets()

This function calls all the network initialization routines (currently there is only one) to initialize all network adapters at once. The function savvecs() is also called to save the interrupt vector table (because the 3Com routines modify it).

4.2.4.2. ResetNets()

This function calls all the network reset routines (currently just one) to reset all the network adapters at once.

4.2.4.3. CloseNets()

This function calls ResetNets() to reset all of the network adapters, and then fixvecs() to restore the previously saved interrupt vector table.

4.2.4.4. StatsNets()

This function calls the network functions that display network statistics.

4.2.5. Functions Defined in ASAT.C

This file provides a basic interface to the ASAT network. In the current configuration,

this file is redundant; however, if the configuration is expanded into a multiple-network configuration, this function structure will become important. Not all of the functions included in this module are called. Those functions which go unused were developed for debugging purposes.

4.2.5.1. asat_init_503()

This function is not implemented. It will be used for future development.

4.2.5.2. asat_read_503()

This function calls cRcvSome() to get the most recent packet received. It then checks the first six bytes of packet data (the source field in the case of the ASATs) and compares it to the passed six-byte string. If the addresses match, the function copies the packet into the buffer specified (passed by reference) and updates two variables (passed by reference) that contain timestamp data (for statistical calculations).

4.2.5.3. asat_write_503()

This function transmits an ASAT packet that has been received. The function expects the address of a structure and the length of the packet. The function calls cXmit1().

4.2.5.4. asat_close_503()

This function is not implemented. It will be used for future development.

4.2.5.5. asat_stats_503()

This function displays some statistics particular to the ASAT.C module.

4.2.5.6. asat_dump()

This function is for debugging purpose only. It will DUMP the content of an ASAT PDU in hexadecimal format. The function expects the address of a structure and does not return a value.

4.2.5.7. asat_display()

This function is for debugging purpose only. It will DISPLAY the content of an ASAT PDU packet. The function expects the address of a structure and does not return a value.

4.2.6. Functions Defined in SIMNET.C

This file is somewhat redundant in the current configuration; however, in an expanded configuration this design will become important. Not all of the functions included in this module are called. Some were developed and used for debugging purposes.

4.2.6.1. simnet_init_503()

This function is not implemented. It will be used for future development.

4.2.6.2. simnet_addr_503()

This function returns the Ethernet address in the SIMNET LAN that refers to the translator.

4.2.6.3. simnet_read_503()

This function calls cRcvSome() to get the most recent packet and copy it into the specified (pass by reference) buffer.

4.2.6.4. simnet_write_503()

This function transmits a packet and does not return a value. The parameters passed to the function are a packet and the length of the packet. The function uses the assembly level function cXmit1() to send a packet to the network. It is called from TEST6.C.

4.2.6.5. simnet_close_503()

This function is not implemented. It will be used for future development.

4.2.6.6. simnet_dump()

This function is for debugging purposes only. It will DUMP the content of a SIMNET PDU in hexadecimal format. The content will be in SIMNET network order. Note that any field that is not exactly one byte long is in reverse order from the IBM 386 format. The function expects the address of a structure and does not return a value.

4.2.6.7. simnet_display()

This function is for debugging purposes only. It will display the content of a SIMNET PDU. The function expects the address of a structure. The content of the buffer must be in host (IBM 386) order.

4.3. Translator Software Routines

There are four C files in the translator software. The files are TEST6.C, PACKDEF.C, ASAT2SIM.C, and MISC.C. The main module is TEST6.C.

4.3.1. Functions Defined in PACKDEF.C

This file provides a SIMNET Vehicle Appearance PDU template along with an Association Protocol Datagram Header so that the entire SIMNET packet does not have to be reconstructed. Note that the translation functions actually update the existing template.

4.3.1.1. defaultA10Pack()

This function constructs a VAPDU template for an A-10 air vehicle in the specified buffer. The Exercise ID is passed into the function. Note that the ASAT is displayed as an A-10 in this translation. The SIMNETs do not support F-16 models, therefore an A-10 was used as the next viable candidate.

4.3.1.2. defaultDatagram()

This function constructs a defaul: datagram header for the Association Protocol. The header is constructed in the specified header field. The length field of the header is passed to the function.

4.3.2. Functions Defined in ASAT2SIM.C

This file provides the actual translation of the ASAT packets to SIMNET PDU's.

4.3.2.1. asat2sim()

This function is passed the ASAT packet and the SIMNET VA PDU template. It then strips specific information out of the ASAT packet, calls routines to translate this information into SIMNET structures, and updates the SIMNET template.

4.3.2.2. Rcalvelocity()

This function calculates the velocity of the ASAT using the roll, pitch, yaw, and velocity information. This function assumes that the ASAT heading is the direction of its motion (this is not always true).

4.3.2.3. Rcallocation()

This function calculates the location of the ASAT in the SIMNET world using the ASAT's x, y, and z coordinates, and a pre-defined displacement between the ASAT and SIMNET worlds. This displacement is implemented to make it easier to visually locate the ASAT (A10) in the SIMNET visuals.

4.3.2.4. Rcalrotation()

This function calculates the SIMNET rotation matrix for the ASAT using information from the roll, pitch, and yaw fields of the ASAT packet.

4.3.3. Functions Defined in MISC.C

4.3.3.1. swap8()

This function expects an address to the beginning of a character string, and swaps eight bytes to invert their sequence. The function does not return a value.

4.3.3.2. swap4()

This function expects an address to the beginning of a character string, and swaps four bytes to invert their sequence. The function does not return a value.

4.3.3.3. swap2()

This function expects an address to the beginning of a character string, and swaps two bytes to invert their sequence. The function does not return a value.

SIMNET Header	Destination Address	Source Address	Type	Data
	Bytes 1-6	Bytes 7-12	Bytes 13-14	Bytes 15-xxx
ASAT Header	does not exist	Source Address	does not exist	Data
		Bytes 1-6		Bytes 7-xxx

Figure 4. SIMNET and ASAT PDU Headers

4.3.4. Functions Defined in TEST6.C

This file contains the main program.

5. PROTOCOL DATA UNITS (PDUs)

PDUs are the elements of data exchanged between simulators and provide the information required for interactive, real-time, networked simulation. Every PDU is divided into individual fields. The specific layout and meaning of the fields depends on the type of PDU being transmitted.

A description of the SIMNET architecture and protocols is given in The BBN SIMNET Network and Protocols manual [4]. The version of the SIMNET Protocols covered in this report is 6.0. This is also the version currently used by the simulators at IST. A pictorial layout of the SIMNET VA PDU and the ASAT packet type 8 is provided in Appendix A.

The format of the ASAT PDUs was found to be incompatible with IEEE 802.3, because it lacked a proper 802.3 header. The standard ETHERNET packet has a destination address, source address, and a type field. In contrast, the ASAT packet simply has a source address followed by vehicle specific data (see Figure 4).

5.1. ASAT Protocol Data Units

There are four types of ASAT packets: type 0, 1, 8, and 9. All ASAT trainers must transmit PDUs of type 0 and type 8. The master system transmits PDUs of types 0, 1, 8, and 9. The ASAT application has a master simulator which performs all of the necessary calculations for the computer generated targets. The master system also initializes the vehicles (computer generated and actual trainers) in their proper location and orientation. Below is a description of the four types of packets.

- Type 0: Performs the handshaking between two entities.
- Type 1: The master system utilizes this packet to initialize all aircraft in their proper position.
- Type 8: This packet is the ASAT's Vehicle Appearance packet. It broadcasts a simulator's position, orientation, status, etc. This is only broadcasted by manned simulators.

- Type 9: This packet is the Vehicle Appearance packet for the computer generated targets. It contains information about the position, orientation, status, etc. of the enemy vehicles. Again, only the master system transmits these packets.

The IST translator deals solely with packets of Type 8. These are the packets which the translator copies from the network to change into SIMNET Vehicle Appearance packets. A description of the Type 8 packet is given in section 5.3.

5.2. ASAT Packet Header

Every ASAT packet has a 12 byte header consisting of the following fields:

- ETHERNET Source Address [6 bytes]
- Packet Length [2 bytes]
- Blank Field (reserved for future extension) [2 bytes]
- Packet Type (0, 1, 8, or 9) [2 bytes]

5.3. Packet Type 8

Besides informing other entities on the network of the simulator's position, orientation, and status, this packet also broadcasts the location, attitude, and speed information of any missiles fired by that simulator, and contains radar information. The packet length is variable since it depends on the number of missiles, if any, that are fired. The packet layout is presented below:

bytes 0-11:	packet header (described above)
byte 12:	simulator ID #
byte 13:	simulator type
bytes 14-25:	simulator XYZ
bytes 26-31:	simulator YPR
bytes 32-35:	simulator speed
bytes 36-37:	simulator hit status (set if damage/destroyed an aircraft)
bytes 38-39:	simulator radar status
bytes 40-50:	order table of other simulators/vehicles
bytes 51-52:	simulator status (reports of any damage, spinning, etc.)

bytes 53-58:	simulator heading rate, climb rate, and bank rate
bytes 59-60:	simulator missiles/flares (0 -> no missiles/flares)
bytes 61-104:	information about missiles that are being decoyed
bytes 105-112:	simulator missile data (whether the missile is tracking the target, just hit target, etc.)
bytes 113 - ...:	missile XYZ (12 bytes)
	missile speed (4 bytes)
	missile YP (4 bytes)
	missile XYZ (12 bytes)
	missile speed (4 bytes)
	missile YP (4 bytes) . . .

Note that the order table (bytes 40-50) is a list of all aircraft ID numbers for those aircraft that are involved in an exercise (including computer generated aircraft). The size of the order table is 11 bytes. The crewed simulators will take precedence in the numbering of the order table. For example if there are five simulators in the exercise they will take on values 1-4 (sorted by ETHERNET address), the computerized aircraft will be numbers 5 thru 11. The ID numbers for the computerized aircraft will be assigned by the master system.

Example:

There are ten vehicles in a simulation, of which five are actual simulators and five are computerized enemy aircraft. Initially, all enemy aircraft will occupy the low address space and all friendly aircraft will take the remaining space. In the master system the order table will look like this:

5, 6, 7, 8, 9, 1, 2, 3, 4

The system which has ID number 2 will have the following order table:

5, 6, 7, 8, 9, 1, 0, 3, 4

The first entry in the order table corresponds to mig0 in the mig data area, the second entry corresponds to mig1, etc. If an F-16 locks on a mig, that mig will become mig0. Accordingly, if the F-16 locks on mig4, the corresponding entries in the order table and mig data need to be swapped.

6. ASAT CHARACTERISTICS

6.1. Terrain Database

The ASAT terrain database is described by a variable of 24 bits in length and width

(800000h x 800000h feet). This translates to 8,388,608 square feet, or approximately 2,557km to a side.

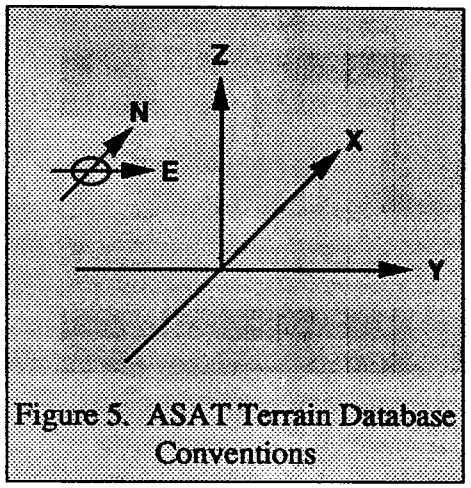


Figure 5. ASAT Terrain Database Conventions

6.2. Coordinate System

The coordinate system has its origin at the center of the database; elevation z=0 is on the ground (see Figure 5).

There are two units used to keeping track of the simulator's position: 32-bit high-resolution units and 24-bit low-resolution units.

1. There are 32-bit high-resolution units which are updated every time the simulator moves. The high-resolution units are used to maintain precision. They are the units that are transmitted over the network. One high-resolution unit equals 1/256 foot.

2. When high precision units are not necessary the high-resolution units are scaled down to 24-bit low-resolution units. The low-resolution units are used to calculate distance, intercept angle, etc.. One low-resolution unit equals 1 foot.

6.3. Attitude Data

The heading, climb, and bank use high and low-resolution units. Low resolution angles range in value from 0 to 360. High resolution units range from 0 to 64,000.

6.4. Velocity

The units used for velocity are the same as those used for the coordinate system: ft./sec and ft x 256/sec.

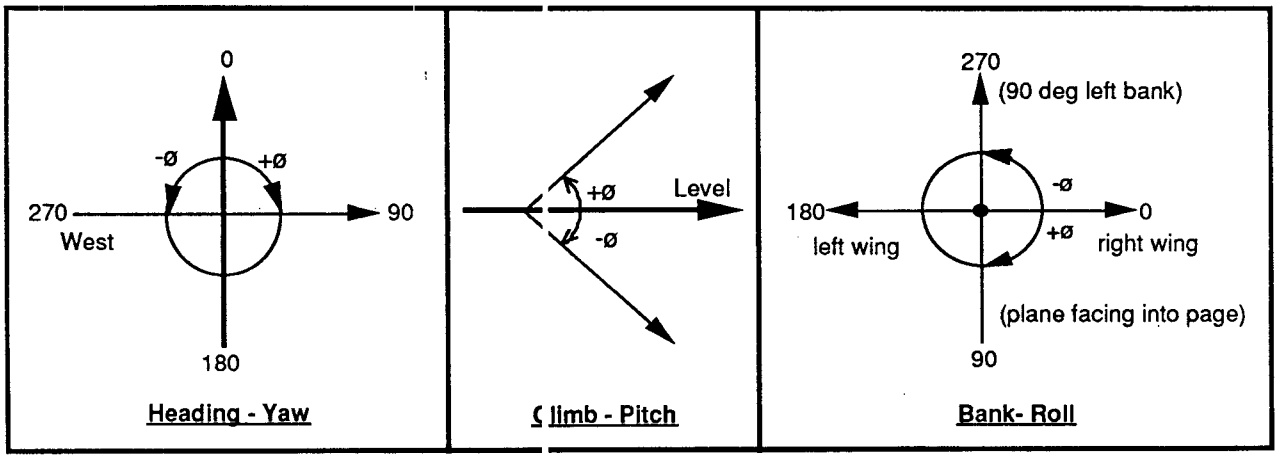
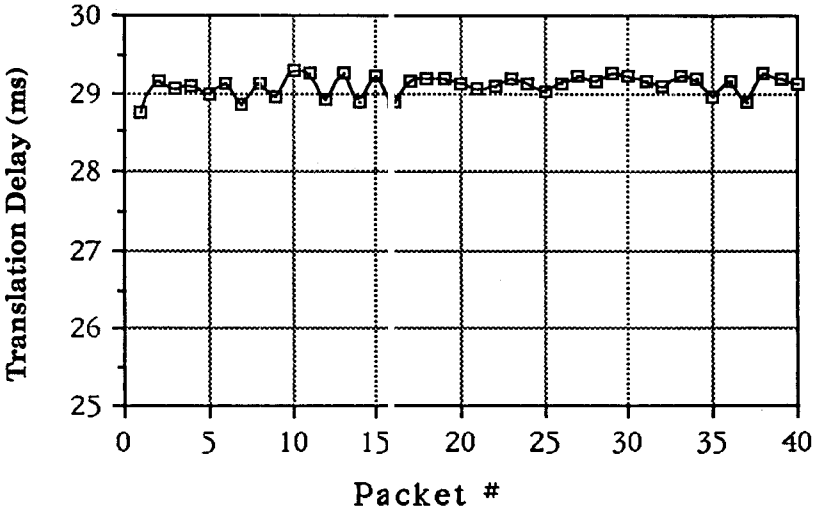


Figure 6. ASAT Attitude Data

7. TRANSLATOR PERFORMANCE TEST AND EVALUATION

The introduction of an intelligent gateway, bridge, or protocol translator will cause the simulation's performance to incur some penalties in the form of network delays. These delays are introduced by the extra processing that must be performed at the translator level. Since the extra time that it takes for a packet to reach its destination is an important measurement, board statistics were accessed and several software routines were developed to test and evaluate the timing performance of the protocol translator. For our initial testing, a Hewlett

Figure 7.



Packard 4972A ETHERNET LAN Analyzer was used to transmit packets of various types at differing rates.

The first test scenario involved sending ASAT packets from the LAN Analyzer at a rate of one packet every 100 ms. These packets were typically 113 bytes in length. The second scenario involved alternately sending an ASAT packet and a non-ASAT packet every 20 ms. The ASAT packet was 113 bytes long, while the non-ASAT packet was 414 bytes long. The non-ASAT packet was introduced so that the 3Com board would have to perform packet rejection while translation was still being performed. For the first scenario, the average translation time was 29.99 ms. As expected, the second scenario produced a larger delay of 32.94 ms.

The final tests were performed using traffic directly from the ASAT simulators. On the average, the translator received one packet every 83 ms for translation purposes. Other traffic on the network included packets from the SIMNET M1 and packets created by the ASAT master system for the appearance of the computer generated targets. Under these conditions the average translation time was 29.10 ms.

Figure 7 is a graph which illustrates the total delay for thirty packets received under the

final test conditions.

8. CONCLUSION

The interconnection between the ASATs and SIMNET was accomplished with several limitations. First, the translation was not bi-directional. It was only performed from ASAT to SIMNET. This was due to certain factors that are inherent in the ASAT design. The nature of the ASAT design does not allow for simple networking of non-ASAT vehicles into the ASAT environment. Also, the ASATs do not provide models for ground vehicles and its software does not handle vehicles which can operate at zero velocity.

Another limitation relates to packet delays that are incurred when the translation process takes place. The time that it takes to perform the operations of the translator may be critical to the realism of a simulation exercise. The delays experienced in this experiment were acceptable for our application; however, in a large scale distributed simulation there are many factors which must be considered. Weapons fire/effects, radars and emitters, environmental effects, and dynamic terrain are just a few of the many considerations which were not part of the focus of this project. If these factors were to be implemented in a similar project as this, the delays may be a critical factor in the feasibility of its use.

The Protocol Translator creates a new packet for every ASAT PDU that is received. The translated ASAT PDU will be added to the network, while the original ASAT packet continues on the network as surplus network traffic. The increase in traffic is given by the simple formula:

$$\text{Increased traffic [one-way translation]} = N_{pt}(R_a)$$

$$\text{Increase in traffic [two-way translation]} = N_{pt}(R_s + R_a)$$

where N_{pt} is the number of protocol translators on the network, and R_a and R_s are the rates of transmission for the ASATs and the SIMNETs, respectively. In our experiment we had a single protocol translator performing a one-way translation. We are assuming that one translator will translate the traffic of a single ASAT simulator. The increase in traffic is equal to adding another ASAT module onto the network. This means that on the average the increase will be:

$$(1 \text{ translator})(113 \text{ bytes/packet})(12 \text{ packets/sec}) = 1.356 \text{ Kbytes/sec}$$

This increase in traffic is merely 0.15% of the usable network bandwidth. However, it must be kept in consideration that the ASATs are selective fidelity simulators and do not have as rapid an update rate as most flight simulators. A gateway approach may be more appropriate for use as a translator because this would not increase the network traffic.

9. Bibliography

[1] J. Cadiz, R. Ouyang, J. Thompson, "Interfacing of the Silicon Graphics Networkable Flight Simulator with SIMNET," Institute for Simulation and Training Publication Number IST-TR-89-2, October 8, 1989.

[2] Perceptronics Manual, "ASAT Functional Description and Operator's Guide," Publication Number 4031-03-0000, September 1, 1989.

[3] 3Com Manual, "Link Level Library (3L) Interface Specification," Manual Part Number 4205-01, January 1989.

[4] Arthur R. Pope, Draft Version "The SIMNET Network and Protocols," BBN Systems and Technologies Report Number 7102, November 1987.

APPENDIX A

GRAPHICAL REPRESENTATION

SIMNET VEHICLE APPEARANCE PDU

&

ASAT PACKET TYPE 8

FIGURE A1: SIMNET VA PDU

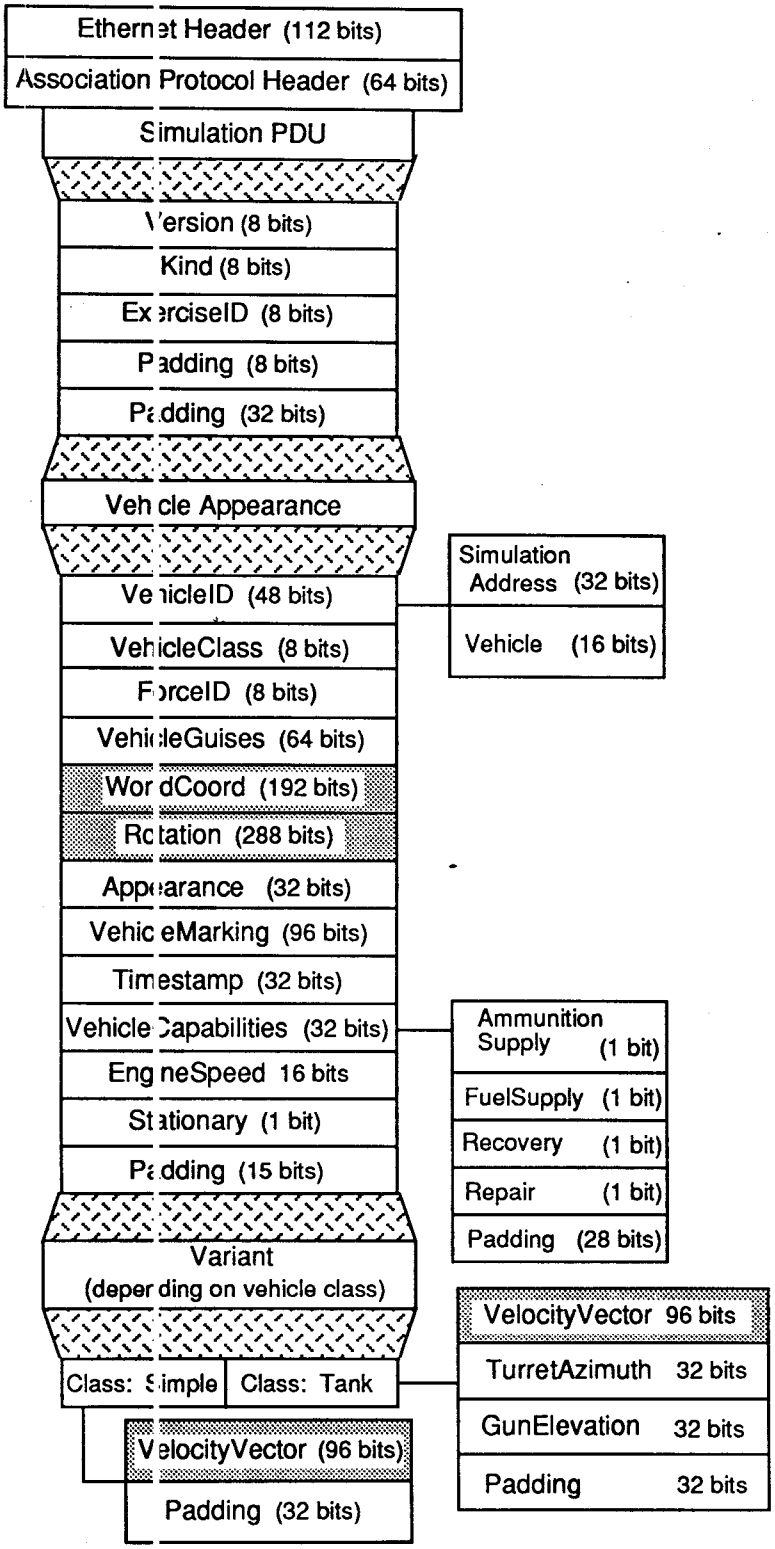
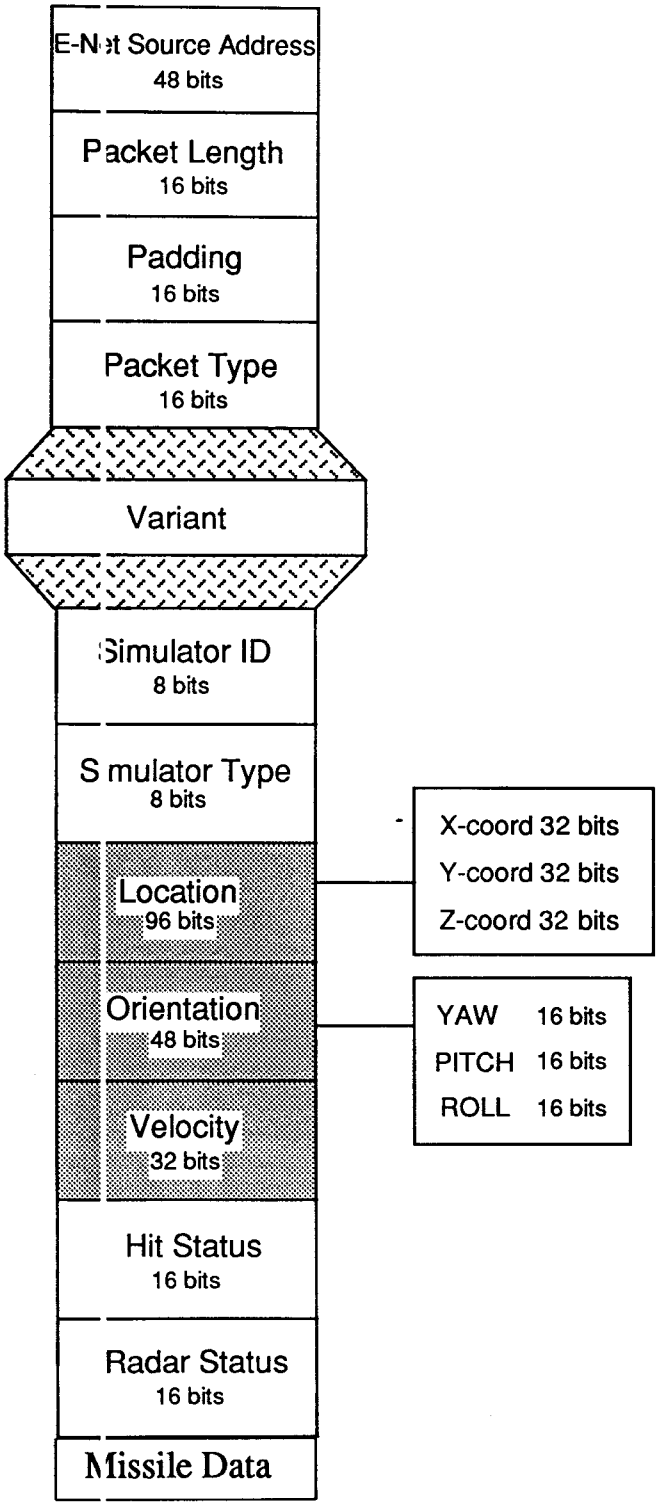


FIGURE A2: ASAT PACKET TYPE 8



APPENDIX B

MODULAR HIERARCHY DIAGRAM

&

FUNCTIONAL HIERARCHY DIAGRAM

APPENDIX B

FIGURE B1: MODULAR HIERARCHY

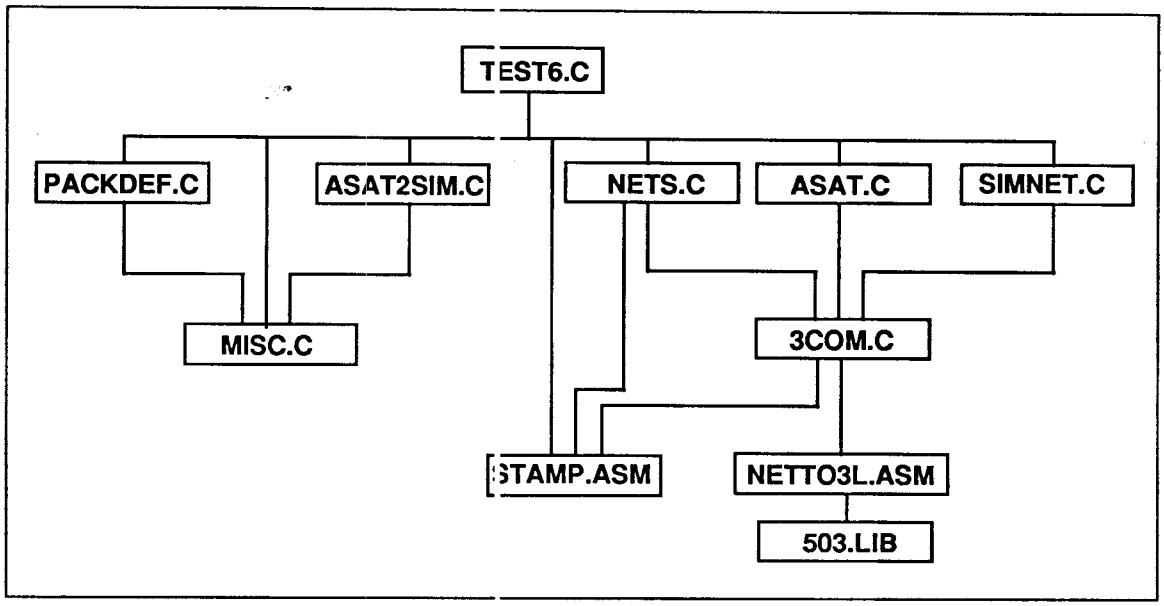
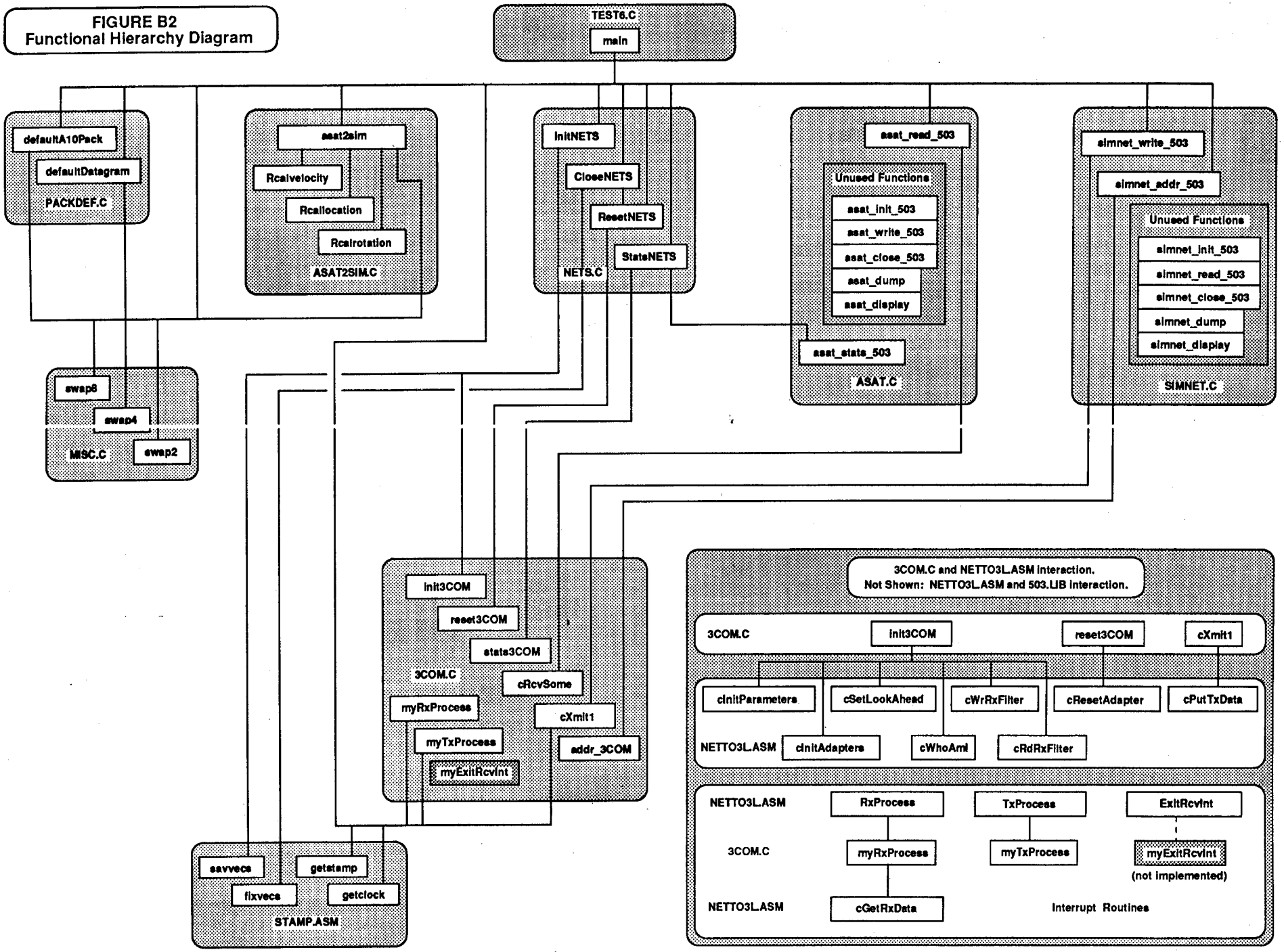


FIGURE B2
Functional Hierarchy Diagram



APPENDIX C

SOFTWARE LISTINGS

ORDER OF SOFTWARE MODULES

Project/Xlate Directory:

TEST6.C

MISC.C

PACKDEF.C

ASAT2SIM.C

Project/Nets Directory:

3COM.C

ASAT.C

3COM.H

STAMP.ASM

NETS.C

SIMNET.C

Project Directory:

ASAT.H

SIMNET.H

```
/*****************************************************************************/
/*                                                                           */
/*  TEST6.C                                                                  */
/*                                                                           */
/*  Description: This is the driver program for the ASAT to SIMNET           */
/*               translator.  Other related files are :                      */
/*                     PACKDEF.C          A: AT.C                            */
/*                     ASAT2SIM.C         S: MNET.C                          */
/*                     MISC.C             3( OM.C                            */
/*                                        N: TS.C                            */
/*                                        N: TTO3L.ASM                       */
/*                                        S: AMP.ASM                         */
/*                                                                           */
/*****************************************************************************/

#include <stdio.h>
#include <time.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <math.h>

#include "..\\sim.h"
#include "..\\asat.h"

#define  VAPDU_LENGTH   158
#define  EXERCISE_ID      1
#define  VEHICLE_ID     256

/*
#define DISPLAY 1
*/

/*  define external functions  */
extern unsigned short getclock();
                      /* gets the low word of MS-DOS clock (count up) */
extern unsigned short getstamp();
                      /* gets CPU clock co nter (counts down) */
extern unsigned long  getATimeStamp();
                      /* performs function of getclock and getstamp */

extern unsigned char far *simnet_addr_503();

/*  Global Variables                        */
/*  these variables are for time-delay stats */

unsigned long  rcv_start_stamp;
unsigned long  xlate_start_stamp;
unsigned long  xlate_end_stamp;
unsigned long  xmit_start_stamp;



/****************************** MAIN ROUTINE ****************************/

main()
{

/*time calcs variables*/
    unsigned long delta0,delta1,delta2,delta3,delta4,delta5,delta6;


/* various variables */
    unsigned char asat_address[6];
    unsigned char far *simnet_addr;

    unsigned short startm, outm;
    int msec;
```

```
/*time calc stats variables*/
    int  inpacketcount    = 0;  /* total number of packets processed */
    int  xlatecount       = 0;  /* total number of packets translated (transmitted) */
    long sumxlatecycles   = 0;  /* sum of all translation delays (clock cycles) */
    long sumtotalcycles   = 0;  /* sum of all total packet delays (clock cycles) */
    long readcount        = 0;  /* number of calls to asat_read_503 */

    struct asat asat_buf;
    simnet simnet_buf;
    int rc = 0, icnt = 0, i, j;


/* initialize all network software */

    InitNets();

/*  the ASAT address changes, so this may have to change */

    asat_address[0] = 0x02;
    asat_address[1] = 0x60;
    asat_address[2] = 0x8c;
    asat_address[3] = 0x0d;
    asat_address[4] = 0x25;
    asat_address[5] = 0xea;

 /* construct default packet and association protocol header */

    defaultDatagram(&simnet_buf.association_header[0],VAPDU_LENGTH);
    defaultA10pack(&simnet_buf.PDU,EXERCISE_ID);

    simnet_buf.PDU.variant.appearance.vehicleID.vehicle = VEHICLE_ID;
    swap2((char *) &simnet_buf.PDU.variant.appearance.vehicleID.vehicle);

    simnet_addr = (unsigned char far *) simnet_addr_503();
    simnet_buf.ether_source [0] = simnet_addr[0];
    simnet_buf.ether_source [1] = simnet_addr[1];
    simnet_buf.ether_source [2] = simnet_addr[2];
    simnet_buf.ether_source [3] = simnet_addr[3];
    simnet_buf.ether_source [4] = simnet_addr[4];
    simnet_buf.ether_source [5] = simnet_addr[5];

    simnet_buf.ether_dest [0] = 0x03;     /* simnet MULTICAST address */
    simnet_buf.ether_dest [1] = 0x00;
    simnet_buf.ether_dest [2] = 0x00;
    simnet_buf.ether_dest [3] = 0x00;
    simnet_buf.ether_dest [4] = 0x01;
    simnet_buf.ether_dest [5] = 0x01;

    simnet_buf.ether_type = 0x5208;
    swap2((char *) &simnet_buf.ether_type);

/* begin translation process */

    printf("Press <return> to begin translation\n");
    getchar();
    printf("Translation begins\n");


/*  This is a quick fix for a new minor problem we haven't solved yet. */
simnet_buf.association_header[5] = 0;

/* translator loop */
    while ( !kbhit() )
      {
      /* see if we have a packet */
       rc = asat_read_503 (&asat_buf, &asat_address[0],
                           &rcv_start_stamp);
       readcount++;    /* we tried to get a packet one more time */
```

```
       if  (rc > 0)    /* do we have a packet ? */
          {            /* yes we have a packet */

           inpacketcount++;   /* one more processed packet */
#ifdef DISPLAY
           printf(".");       /* show on display that we got a packet */
#endif
           icnt++;

           if (asat_buf.hdr.type == 8)  /* is it the right type of packet ? */
               {                        /* yes */
                 xlate_start_stamp = getATimeStamp();
                                        /* get translate start times */
                xlatecount++;           /* one more packet translated */

                asat2sim(&simnet_buf,&asat_buf);  /* translate packet */
                  xlate_end_stamp   = getATimeStamp();
                                        /* get translate end times */

                simnet_write_503(&simnet_buf, VAPDU_LENGTH); /*transmit packet */
                  xmit_start_stamp = getATimeStamp();
                                        /* get transmit end times */


/* calculate delay times (cycles) based on raw time data */

                delta2 = xlate_end_stamp   - xlate_start_stamp;
                delta3 = xmit_start_stamp  - rcv_start_stamp;

/* adjust measured times to account for TimeStamp inaccuracies */
//                if (delta2 < 25000) delta2 = delta2 + 65536;
//                if (delta3 < 25000) delta3 = delta3 + 65536;

                /* this is for the average of delay times */
                sumxlatecycles = sumxlatecycles  + delta2;
                sumtotalcycles = sumtotalcycles  + delta3;

#ifdef DISPLAY
/* display delay times for current packet */
/* NOTE : something screws up the PRINTF call so that the first call prints
          five characters and then a 0 (ending the printf call) */

                printf("\n packet ");
                printf("\nPkt %7d stats:",xlatecount);
                printf("  xlate cyc. = %7ld", delta2);
                printf("  total int. cyc. = %7ld\n",delta3);
#endif

               } /* end if asat_buf.hdr.type part */
          }  /* end if rc > 0 part */
      }  /* end while !kbhit */


/* reset network adapters and display translation statistics */

      CloseNets();

      printf("\nNet Stats:\n");
      StatsNets(); /* display net-specific information */

      printf("\nXlator Stats:\n");
      printf(" Total packet input polling    : %ld\n",readcount);
      printf(" Total translator input count  : %d\n",icnt);
      printf(" Number packets translated     : %d\n",xlatecount);
if (xlatecount > 0)
     {
      printf(" average translation time is  %lu / %d = %lu cycles\n",
              sumxlatecycles,  xlatecount, (sumxlatecycles  / xlatecount));
```

```
        printf(" average total delay time is  %lu / %d = %lu cycles\n",
                sumtotalcycles, xlatecount, (sumtotalcycles / xlatecount));
      }

      exit (0);
}
```

```
/*********************************************************************
misc.c

        This file contains the miscellaneous c code

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

/* This subroutine swaps 8 bytes to reverse order */
swap8(char *ptr)
{
    char tmp;

/* swap bytes 0 and 7 */
    tmp = *ptr;
    *ptr = *(ptr+7);
    *(ptr+7) = tmp;

/* swap bytes 1 and 6 */
    tmp = *(ptr+1);
    *(ptr+1) = *(ptr+6);
    *(ptr+6) = tmp;

/* swap bytes 2 and 5 */
    tmp = *(ptr+2);
    *(ptr+2) = *(ptr+5);
    *(ptr+5) = tmp;

/* swap bytes 3 and 4 */
    tmp = *(ptr+3);
    *(ptr+3) = *(ptr+4);
    *(ptr+4) = tmp;
}


/* This subroutine swaps 4 bytes to reverse order */
swap4(char *ptr)
{
    char tmp;

/* swap bytes 0 and 3 */
    tmp = *ptr;
    *ptr = *(ptr+3);
    *(ptr+3) = tmp;

/* swap bytes 1 and 2 */
    tmp = *(ptr+1);
    *(ptr+1) = *(ptr+2);
    *(ptr+2) = tmp;
}



/* This subroutine swaps 2 bytes to reverse order */
swap2(char *ptr)
{
    char tmp;

/* swap bytes 0 and 1 */
    tmp = *ptr;
    *ptr = *(ptr+1);
    *(ptr+1) = tmp;
}
```

```
/*****************************************************************************/
/*                                                                           */
/*      PACKDEF.C                                                            */
/*                                                                           */
/*      This file contains functions that create a default SIMNET packet.    */
/*      Current functions include :                                          */
/*                                                                           */
/*              defaultA10pack()        creates a default A10 VAPDU          */
/*              defaultDatagram()       creates a Datagram header for PDU    */
/*                                                                           */
/*****************************************************************************/

#include "..\\sim.h"
#include "..\\asat.h"


#define DefaultVeh_ID 1000
#define DefaultSite   256
#define DefaultHost   256

char *default_marking = "                              ";


/*****************************************************************************/
/*                create a Vehicle Appearance PDU for an A10                 */

defaultA10pack(simbuf,ex_ID)
SimulationPDU *simbuf;
unsigned char  ex_ID;

{
unsigned short *temp;
int i;

/*********************************************** common to all packets */

simbuf->version        = protocolVersionCurrent;
simbuf->kind           = vehicleAppearancePDUKind;
simbuf->exercise       = ex_ID;
simbuf->_unused_48_49 = 0;
simbuf->_unused_48     = 0;

/*********************************************** variant part */



/*********************************************** vehicle ID */

simbuf->variant.appearance.vehicleID.simulator.site = (DefaultSite);
swap2((char *) &simbuf->variant.appearance.vehicleID.simulator.site);

simbuf->variant.appearance.vehicleID.simulator.host = (DefaultHost);
swap2((char *) &simbuf->variant.appearance.vehicleID.simulator.host);

simbuf->variant.appearance.vehicleID.vehicle        = (DefaultVeh_ID);
swap2((char *) &simbuf->variant.appearance.vehicleID.vehicle);



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

simbuf->variant.appearance.vehicleClass         =  vehicleClassTank;
simbuf->variant.appearance.force                =  distinguishedForceID;



/*********************************************** description */
```

```
temp = (short *) &simbuf->variant.appearance.guises.distinguished;
temp[0] = ( (1 << (objectDomainShift - 16))        | \
            (1 << (vehicleEnvironmentShift - 16)) | \
            (1 << (vehicleClassShift - 16))       | \
            (countryUS << (vehicleCountryShift - 16)));
temp[1] = ( (1 << vehicleSeriesShift) | (0 << vehicleModelShift) |     \
             vehicleFunctionGroundAttack);

swap2((char *) &temp[0]);
swap2((char *) &temp[1]);


simbuf->variant.appearance.guises.other =
        simbuf->variant.appearance.guises.distinguished;


/*********************************************** location data */

simbuf->variant.appearance.location[0]  = (SIMNET_AIRPORT_X - ASAT_AIRPORT_Y);
swap8((char *) &simbuf->variant.appearance.location[0]);

simbuf->variant.appearance.location[1]  = (SIMNET_AIRPORT_Y - ASAT_AIRPORT_X);
swap8((char *) &simbuf->variant.appearance.location[1]);

simbuf->variant.appearance.location[2]  = (SIMNET_AIRPORT_Z - ASAT_AIRPORT_Z);
swap8((char *) &simbuf->variant.appearance.location[2]);



/*********************************************** rotation matrix */

simbuf->variant.appearance.rotation[0][0] = (0.0);
swap4((char *) &simbuf->variant.appearance.rotation[0][0]);

simbuf->variant.appearance.rotation[0][1] = (0.0);
swap4((char *) &simbuf->variant.appearance.rotation[0][1]);

simbuf->variant.appearance.rotation[0][2] = (0.0);
swap4((char *) &simbuf->variant.appearance.rotation[0][2]);

simbuf->variant.appearance.rotation[1][0] = (0.0);
swap4((char *) &simbuf->variant.appearance.rotation[1][0]);

simbuf->variant.appearance.rotation[1][1] = (0.0);
swap4((char *) &simbuf->variant.appearance.rotation[1][1]);

simbuf->variant.appearance.rotation[1][2] = (0.0);
swap4((char *) &simbuf->variant.appearance.rotation[1][2]);

simbuf->variant.appearance.rotation[2][0] = (0.0);
swap4((char *) &simbuf->variant.appearance.rotation[2][0]);

simbuf->variant.appearance.rotation[2][1] = (0.0);
swap4((char *) &simbuf->variant.appearance.rotation[2][1]);

simbuf->variant.appearance.rotation[2][2] = (0.0);
swap4((char *) &simbuf->variant.appearance.rotation[2][2]);



/*********************************************** appearance */

simbuf->variant.appearance.appearance = (0);
swap4((char *) &simbuf->variant.appearance.appearance);
```

```
/*************************************************** text and time */

simbuf->variant.appearance.marking.characterSet = asciiCharacterSet;
for (i = 0; i < maxVehicleMarkingLength; i++)
            simbuf->variant.appearance.marking.text[i] = default_marking[i];

simbuf->variant.appearance.timestamp = (0);
swap4((char *) &simbuf->variant.appearance.timestamp );



/************************************************** various */

simbuf->variant.appearance.capabilities.ammunitionSupply = 0;
simbuf->variant.appearance.capabilities.fuelSupply = 0;
simbuf->variant.appearance.capabilities.recovery   = 0;
simbuf->variant.appearance.capabilities.repair     = 0;
simbuf->variant.appearance.capabilities._unused_1  = 0;

swap4((char *) &simbuf->variant.appearance.capabilities);

simbuf->variant.appearance.engineSpeed  = (0);
swap2((char *) &simbuf->variant.appearance.engineSpeed);

simbuf->variant.appearance.stationary   = 0;
simbuf->variant.appearance._unused_31   = 0;
/*
swap2((char *) &simbuf->variant.appearance._unused_31);
*/


/************************************************** velocity */

simbuf->variant.appearance.specific.tank.velocity[0] = (0.0);
swap4((char *) &simbuf->variant.appearance.specific.tank.velocity[0]);

simbuf->variant.appearance.specific.tank.velocity[1] = (0.0);
swap4((char *) &simbuf->variant.appearance.specific.tank.velocity[1]);

simbuf->variant.appearance.specific.tank.velocity[2] = (0.0);
swap4((char *) &simbuf->variant.appearance.specific.tank.velocity[2]);



/************************************************** tank-specific part */

simbuf->variant.appearance.specific.tank.turretAzimuth = (0);
swap4((char *) &simbuf->variant.appearance.specific.tank.turretAzimuth);

simbuf->variant.appearance.specific.tank.gunElevation  = (0);
swap4((char *) &simbuf->variant.appearance.specific.tank.gunElevation);

}



/*****************************************************************************/
/*      create Datagram Association Protocol header for SIMNET PDU        */

defaultDatagram(AssocBuf,length)
AssociationPDU *AssocBuf;
{
char *ABUF;

ABUF = (char *) AssocBuf;
```

```
/* had some problems here -  hardcoded due to structure problems */

/*
AssocBuf->version       = protocolVersionCurrent;
AssocBuf->kind          = datagramAPDUKind;
*/
ABUF[0] = 0x21;


/*
AssocBuf->dataLength    = 0x11; //((length-22.))/8.0)
*/
ABUF[1] = 0x11;

ABUF[2] = 0x01;


AssocBuf->group         = 1;
AssocBuf->userProtocol = 1;

AssocBuf->originator.site = DefaultSite;
swap2((char *) &AssocBuf->originator.site);

AssocBuf->originator.host = DefaultHost;
swap2((char *) &AssocBuf->originator.host);

}
```

```
/*****************************************************************************/
/*                                                                           */
/*      ASAT2SIM.C                                                           */
/*                                                                           */
/*              This file provides functions that do the actual protocol     */
/*              translation from ASAT to SIMNET.  Included functions are:    */
/*                                                                           */
/*                    asat2sim()        general translator function          */
/*                    Rcalvelocity()    translates velocity field            */
/*                    Rcallocation()    translates location field            */
/*                    Rcalrotation()    creates SIMNETY rotation matrix      */
/*                                                                           */
/*****************************************************************************/


#include "..\\asat.h"
#include "..\\sim.h"
#include <math.h>
#include <stdio.h>

#define PI 3.14159265358979323
/*
#define DISPLAYALL 1
*/

Rcalvelocity();
Rcalrotation();
Rcallocation();

static float ROT[3][3];
static float VEL[3];
static float LOC[3];


/**** note : update asat2sim function calls to include address of matrix to be
             updated.  reduce multiple copies of matrices */


void asat2sim(char *simbuf, char *asatbuf)
{
int    i, j, k, temp, rc;

static unsigned short  asat_type;
static char            sourceaddr[6];
static long            asatX, asatY, asatZ;
static unsigned short  asatROL, asatPIT, asatYAW;
static long            asatSPEED;

static float           rotation [3] [3];
static float           location [3];
static float           velocity [3];
float vector[3];
float R,P,Y,RC,RS,PC,PS,YC,YS;

void  *pointr;

struct asat *myasat;
simnet *mysimnet;

/* set buffer pointers */
myasat   = (struct asat *) asatbuf;
mysimnet = (simnet *) simbuf;

/* get source address (for future use) */
for (i=0;i<6;i++) sourceaddr[i] = myasat->hdr.ether_addr[i];

/* check the type of packet */
asat_type = myasat->hdr.type;
```

```
switch (asat_type)
{
        case 0  : break;  /* type 0 not yet implemented */
        case 1  : break;  /* type 1 not yet implemented */
        case 8  : asatX = myasat->data.TYPE8.x;
                  asatY = myasat->data.TYPE8.y;
                  asatZ = myasat->data.TYPE8.z;
                  asatYAW = myasat->data.TYPE8.yaw;
                  asatPIT = myasat->data.TYPE8.pitch;
                  asatROL = myasat->data.TYPE8.roll;
                  asatSPEED = myasat->data.TYPE8.speed;

                  Rcalrotation(asatROL,asatPIT,asatYAW);
                  for (i = 0; i<3; i++) {
                      for (j = 0; j<3; j++)
                           {
                            rotation[i][j] = ROT[i][j];
                           }
                      }

                  Rcallocation(asatX,asatY,asatZ);
                  for (i = 0; i<3; i++)
                      {
                        location[i]    = LOC[i];
                      }

                  Rcalvelocity(asatROL,asatPIT,asatYAW,asatSPEED);
                  for (i = 0; i<3; i++)
                      {
                        velocity[i]    = VEL[i];
                      }


                  break;

        case 9  : break;   /* type 9 not yet implemented */
        default : break;
}  /* end switch */

/* update VAPDU rotation matrix */
for (i = 0; i<3; i++)
    {
     for (j=0; j<3; j++)
      {
        mysimnet->PDU.variant.appearance.rotation[i][j] = rotation[i][j];
        swap4((char *) &mysimnet->PDU.variant.appearance.rotation[i][j]);
       }
    }

/* update VAPDU location */
for (i = 0; i<3; i++)
       {
        mysimnet->PDU.variant.appearance.location[i]    = location[i];
        swap8((char *) &mysimnet->PDU.variant.appearance.location[i]);
       }

/* update VAPDU velocity */
for (i = 0; i<3; i++)
       {
        mysimnet->PDU.variant.appearance.specific.tank.velocity[i] = velocity[i];
        swap4((char *) &mysimnet->PDU.variant.appearance.specific.tank.velocity[i]);
       }
#ifdef DISPLAYALL
printf("velocity is : x: %f  y: %f  z: %f\n", velocity[0],velocity[1],velocity[2]);
#endif
} /* end asat2sim */
```

```
/* calculate velocity vector of ASAT */
Rcalvelocity(asatROL,asatPIT, asatYAW, asatSPEED)
unsigned short asatROL,asatPIT,asatYAW;
long  asatSPEED;
{
float R,P,Y;
float RC,RS,PC,PS,YC,YS;
float vector[3];
int i;

#ifdef DISPLAYALL
    printf("asatSPEED = %ld\n",asatSPEED);
#endif

/* convert input angles from ASAT units to radians */
    R = -(asatROL / 65536.0) * 2 * PI;  /* convert roll from ASAT to radians */
    if (R < -PI) R = R+2*PI;
    if (R >  PI) R = R-2*PI;
#ifdef DISPLAYALL
    printf("ROLL = %f\n",R);
#endif

    P = (0.5-(asatPIT / 65536.0)) * 2 * PI;  /* convert pitch to radians */
    if (P < -PI) P = P + 2*PI;
    if (P >  PI) P = P - 2*PI;
#ifdef DISPLAYALL
    printf("PITCH = %f\n",P);
#endif

    Y =  (asatYAW / 65536.0) * 2 * PI;  /* convert yaw to radians */
    if (Y < -PI) Y = Y + 2*PI;
    if (Y >  PI) Y = Y - 2*PI;
#ifdef DISPLAYALL
    printf("YAW = %f \n",Y);
#endif


/* calculate trigonometric functions of input angles */
    RC = cos(R);
    RS = sin(R);

    PC = cos(P);
    PS = sin(P);

    YC = cos(Y);
    YS = sin(Y);


/* calculate unit vector in heading direction,  assume that velocity unit */
/* vector is the same direction */

vector[0] = (     PC * YS);
vector[1] = (     PC * YC);
vector[2] = (     PS );

/* create velocity vector from speed and direction */
for (i=0;i<3;i++) VEL[i] = vector[i] * (asatSPEED / 256.0) / F2M;

#ifdef DISPLAYALL
printf("scaled speed is : %f\n", ((asatSPEED / 256.0) / F2M));
printf("vector   is : x: %f  y: %f  z: %f\n", vector[0],vector[1],vector[2]);
printf("velocity is : x: %f  y: %f  z: %f\n", VEL[0],VEL[1],VEL[2]);
#endif

} /* end calvelocity */
```

```
/* calculate ASAT location from input coordinates */
Rcallocation(asatX,asatY,asatZ)
long asatX,asatY,asatZ;
{

/* convert ASAT coordinates to SIMNET coordinates (coordinate translation) */
LOC[0] = SIMNET_AIRPORT_X + (asatY / F2M) - ASAT_AIRPORT_Y;
LOC[1] = SIMNET_AIRPORT_Y + (asatX / F2M) - ASAT_AIRPORT_X;
LOC[2] = SIMNET_AIRPORT_Z + (asatZ / F2M) - ASAT_AIRPORT_Z;

#ifdef DISPLAYALL
printf("asatLoc  is : x: %ld  y: %ld  z: %ld\n",asatX,asatY,asatZ);
printf("location is : x: %f  y: %f  z: %f\n",LOC[0],LOC[1],LOC[2]);
#endif

} /* end callocation */


/* calculate rotation matrix from input rotation angles */
Rcalrotation(asatROL,asatPIT,asatYAW)
unsigned short asatROL,asatPIT,asatYAW;
{
    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];

/* convert input angles from ASAT units to radians */
    R = -(     asatROL / 65536.0)  * 2 * PI; /* convert roll to radians */
    if (R < -PI) R = R+2*PI;
    if (R >  PI) R = R-2*PI;

    P = (0.5-(asatPIT / 65536.0)) * 2 * PI; /* convert pitch to radians */
    if (P < -PI) P = P + 2*PI;
    if (P >  PI) P = P - 2*PI;

    Y =     (asatYAW / 65536.0)  * 2 * PI; /* convert yaw to radians */
    if (Y < -PI) Y = Y + 2*PI;
    if (Y >  PI) Y = Y - 2*PI;


/* calculate trigonometric functions of input angles */
    RC = cos(R);
    RS = sin(R);

    PC = cos(P);
    PS = sin(P);

    YC = cos(Y);
    YS = sin(Y);

/* calculate rotation matrix as [3 x 1] matrix of [1 x 3] unit vectors */
/*  calculate 'X vector in world coordinates */


    ROT[0][0] =  (RC * YC) - (RS * PS * YS);
    ROT[1][0] =              (     PC * YS);
    ROT[2][0] = -(RS * YC) - (RC * PS * YS);


/*  calculate 'Y vector in world coordinates */

    ROT[0][1] = -(RS * PS * YC) - (RC * YS);
    ROT[1][1] =  (     PC * YC);
```

```
    ROT[2][1] = -(RC * PS * YC) + (RS * YS);


/*  calculate 'Z vector in world coordinates */

    ROT[0][2] = (RS * PC );
    ROT[1][2] = (     PS );
    ROT[2][2] = (RC * PC );

}
```

```
/*****************************************************************************/
/*      asat.c                                                               */
/*                                                                           */
/*      This file contains the c code for ETHERNET access to the             */
/*      ASAT simulator.                                                      */
/*                                                                           */
/*****************************************************************************/

#include "..\\asat.h"

extern int cRcvSome(char **bufpointer);
extern cXmit1();

static long incount = 0, outcount = 0;

/********************** Initialize the 3COM EtherLinkii connection for ASAT */
asat_init_503 ()
{
    printf("asat_init_503 Not Implement Yet\n");
}

/*********************************************** Read a packet from ASAT  */
asat_read_503 (buf,asat_address,stamp)
unsigned char *buf;
unsigned char *asat_address;
unsigned long *stamp;
       /* stamp is a modularly correct way to
          get this data from the 3COM.C module to the TEST6.C module
          while allowing no direct interaction between the two modules */
{
    int cnt, i;
    unsigned char *Pkt;
    int cnt2 = 0;

_disable();
    cnt = cRcvSome(&Pkt);
    if (cnt > 0)
    {
      incount++;
      if  ((Pkt[0] == asat_address[0]) &&
            (Pkt[1] == asat_address[1]) &&
            (Pkt[2] == asat_address[2]) &&
            (Pkt[3] == asat_address[3]) &&
            (Pkt[4] == asat_address[4]) &&
            (Pkt[5] == asat_address[5])
          )
          {
          /* memcpy(ether_buf,&Pkt, cnt); */
          for (i=0; i<cnt; i++)
              buf[i] = Pkt[i];
          cnt2 = cnt;
          *stamp = *((unsigned long far *) (Pkt - 4));
          outcount++;
          }
    }
_enable();
    return (cnt2);
}

/****************************************** Send a packet to the ASAT */
asat_write_503 (Pkt, cnt)
char *Pkt;
int cnt;
{
    int i, flags = 0x0060, reqid = 0x0001, nreqid = 0x0001;

    cXmit1(cnt, cnt, flags, reqid, Pkt, &nreqid);
}
```

```
/*****************************************************************************/
/*                                                                           */
/*  3COM.C                                                                   */
/*                                                                           */
/*  Description: This file augments the access functions for the 3COM        */
/*               ETHERNET board provided by NETTO3L.ASM.                     */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/

#include <stdio.h>

#include "3com.h"

#define GetRxData_FLAGS  0x40

/*  declare NETTO3L.ASM functions */
extern cInitAdapters();
extern cInitParameters();
extern cResetAdapter();
extern cWhoAmI();
extern cRdRxFilter();
extern cWrRxFilter();
extern cPutTxData();
extern cGetRxData();
extern cSetLookAhead();


/* declare STAMP.ASM functions */
extern savvecs();
extern fixvecs();

extern unsigned short getclock();
extern unsigned short getstamp();
extern unsigned long  getATimeStamp();

/* global variables */

static char   Buffer[1024];      /* implement single receive buffer */
static int    BufLength = 0;     /* length of packet */

static struct WhoStruct far *Who;  /* structures used to pass parameters to */
static struct ini_hdr parmsdr;     /*  3COM routines */
static int Adapters = 0;



/* some timestamp global variables */

static unsigned long  ETH_rcv_start_stamp  = 0;
static unsigned long  ETH_rcv_end_stamp    = 0;
static unsigned long  ETH_xmit_start_stamp = 0;
static unsigned long  ETH_xmit_end_stamp   = 0;
static unsigned long  OLD_rcv_start_stamp  = 0;

static unsigned long totalXmit     = 0,  totalRcv     = 0;
static unsigned long totalXmitTime = 0,  totalRcvTime = 0;

static unsigned long totalInterArrival     = 0xffffffff;
static unsigned long totalInterArrivalTime = 0;


/* functions to simulate old cRcvSome and cXmit1 functions from CTO31C.ASM */

/* transmit a Packet *********************************************************/
int cXmit1(cnt1,cnt2,flags,reqid,Pkt,nreqid)
int cnt1,cnt2,flags,reqid,*nreqid;
unsigned char *Pkt;
```

```
{
int rc = 0;
ETH_xmit_start_stamp = getATimeStamp();
rc = cPutTxData(cnt1,cnt2,flags,reqid,Pkt,neqid);
return(rc);
}


/* check buffer for a packet ************************************/
int cRcvSome(bufpointer)
char **bufpointer;
{
int length;
length = BufLength;
BufLength = 0;
*bufpointer = &Buffer[4]; /* point buffer pointer to buffer */
return(length);           /* and return length of packet (0 if no packet) */
}


/* initialize 3COM board ************************************/
void init3com()
{

    char *pointr;
    int rc, rxf=0x000e, rrxf;
    int rs = 0, i = 0;

/* initialize 3COM board for network communications */

    pointr = (char *) &parmsdr.len;
    for (i=0;i<23;i++) pointr[i] = 0x00;

    parmsdr.len=0x17;
    /* parmsdr.argo = "c:\3com\ether503.sys /a:2e0/m:4/t:1/d:1/i:3\n"; */
    parmsdr.argo = "c:\\3com\\ether503.sys /A:2e0 /D:1 /I:3\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]);

    rc=cWrRxFilter(rxf);
    printf("cWrRxFilter returns %d\n",rc);
    rc=cRdRxFilter(&rrxf);
    printf("cRdRxFilter returns %d, filter=%x\n",rc,rrxf);
    printf("\n\n");
}


/* get current Ethernet Address ************************************/
char far *addr_3COM()
{
```

```
        return(&Who->addr[0]);
}

/* reset 3COM adapter ***************************************************/
void reset3COM()
{
      int rc = 0;
      rc=cResetAdapter();
      printf("cResetAdapter returns %d\n",rc);
}

/* display adapter statistics *******************************************/
void stats3COM()
{
if (totalRcv > 0)
     {
      printf("average 3COM receive      time : %ld / %ld = %ld cycles\n",
              totalRcvTime, totalRcv, (totalRcvTime / totalRcv));
     }
if (totalInterArrival > 0)
     {
      printf("average 3COM interarrival time : %ld / %ld = %ld cycles\n",
              totalInterArrivalTime, totalInterArrival,
              (totalInterArrivalTime / totalInterArrival));
     }
if (totalXmit > 0)
     {
      printf("average 3COM transmission time : %ld / %ld = %ld cycles\n",
              totalXmitTime, totalXmit, (totalXmitTime / totalXmit));
     }
      printf("Total 3COM reception count : %d\n", Who->ttl_recp_cnt);
      printf("3COM WhoAmI stats :\n");
      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("  total reception count       %d \n", Who->ttl_recp_cnt);
      printf("  total reception bdr count   %d \n", Who->ttl_recv_bdr_cnt);
      printf("  total reception errors      %d \n", Who->ttl_recv_err_cnt);

      printf("  total transmission count    %d \n", Who->ttl_tran_cnt);
      printf("  total transmission errors   %d \n", Who->ttl_tran_err_cnt);
      printf("  total transmission timeouts %d \n", Who->ttl_tran_timeout_cnt);

      printf("  total retries               %d \n", Who->ttl_retry_cnt);

}


/*****************************************************************************/
/*        Interrupt processing routines required by 3COM package             */
/*                                                                           */
/* note that these routines are performed inside an interrupt, and thus      */
/* have a limited scope - they should be short, and some function calls      */
/* (especially ones that use interrupts) will not work well inside them.     */
/* Note in particular that PutTxData cannot be called successfully inside    */
/* RxProcess,  and that FTIME does not work inside any of them. Note also    */
/* that for some (similar?) reason, myExitRcvInt does not work if written    */
/* in C.                                                                     */
/*****************************************************************************/

void myRxProcess(Status, PacketSize, RequestID, PacketHeader)
int Status, PacketSize, RequestID;
char far *PacketHeader;
{
        int rc, NumBytes, Flags;
        char far *PacketAddr;
        unsigned long tempInterArrivalTime;
```

```
*((unsigned long *) &Buffer[0]) = getATimeStamp();
ETH_rcv_start_stamp = getATimeStamp();

       totalInterArrival++;
       tempInterArrivalTime  =
                   ETH_rcv_start_stamp - OLD_rcv_start_stamp;
//       if (tempInterArrivalTime < 30000)
//                   tempInterArrivalTime = tempInterArrivalTime + 65536;

       totalInterArrivalTime = totalInterArrivalTime + tempInterArrivalTime;

        if (totalInterArrival == 0) totalInterArrivalTime = 0;
        OLD_rcv_start_stamp = ETH_rcv_start_stamp;

        Flags = GetRxData_FLAGS;
        NumBytes = PacketSize;
        PacketAddr = (char *) &Buffer[4];
        BufLength = 0;

    rc = cGetRxData(&NumBytes, Flags, RequestID, PacketAddr);
ETH_rcv_end_stamp = getATimeStamp();
    if (( ! rc) && ( ! Status))
         {
          BufLength = PacketSize;

          totalRcv++;
          totalRcvTime = totalRcvTime +
                       abs(ETH_rcv_end_stamp - ETH_rcv_start_stamp);
         }
}


void myTxProcess(Status, RequestID)
int Status, RequestID;
{
ETH_xmit_end_stamp = getATimeStamp();
if (! Status)
     {
      totalXmit++;
      totalXmitTime = totalXmitTime +
                    abs(ETH_xmit_end_stamp - ETH_xmit_start_stamp);
     }
}

/*  myExitRcvInt does not work in C (stack overflow error) */
/*
void myExitRcvInt()
{
}
*/
```

```
/************************************ Close the connection for the ASAT */
asat_close_503 ()
{
    printf("asat_close_503 Not Implement Yet\n");
}

/************************************* Display ASAT Packet statistics */
asat_stats_503()
{
        printf("total unfiltered packets = %ld\n",incount);
        printf("total ASAT       packets = %ld\n",outcount);
}


/* This subroutine is for debugging purpose only, it will DUMP the content of a
   ASAT pdu in hexdecimal. The content should be in NETWORK ORDER */
asat_dump (buf)
struct asat *buf;
{
    int i, j;
    unsigned short netcnt;

    printf("ASAT content\n");
    netcnt = buf->hdr.length;
    printf("Source addr      : %2x-%2x-%2x-%2x-%2x-%2x\n",
        buf->hdr.ether_addr [0], buf->hdr.ether_addr [1],
        buf->hdr.ether_addr [2], buf->hdr.ether_addr [3],
        buf->hdr.ether_addr [4], buf->hdr.ether_addr [5]);
    printf("%d\n",netcnt);
    printf("%2x %2x\n", buf->data.DATAONLY[0], buf->data.DATAONLY[1]);
    for (i=2, j=3; i<netcnt; i++, j++) {
        printf("%2x ", buf->data.DATAONLY[i]);
        if  (j >= 17) {
            j=0;
            printf("\n");
        }
    }
    printf("\n");
}

/* This subroutine is for debugging purpose only, it will DISPLAY the content of
   a ASAT pdu packet.  This content should be in HOST ORDER */
asat_display (buf)
struct asat *buf;
{
    int i, j;

    printf("ASAT INFORMATION\n");
    printf("x = %ld, y = %ld, z = %ld\n",buf->data.TYPE8.x, buf->data.TYPE8.y,
           buf->data.TYPE8.z);
    printf("(original) yaw = %d, pitch = %d, roll = %d\n",
           buf->data.TYPE8.yaw,
           buf->data.TYPE8.pitch,
           buf->data.TYPE8.roll);
    printf("(degree) yaw = %f, pitch = %f, roll = %f\n",
           (buf->data.TYPE8.yaw)*360.0/65536.0,
           (32768.0 - buf->data.TYPE8.pitch)/180.0,
           (buf->data.TYPE8.roll)*360.0/65536.0);
}
```

```
/*********************************************************************/
/*                                                                   */
/*      3COM.H                                                       */
/*                                                                   */
/*              This file provides two structures used as parameters for  */
/*              the 3Com Adapter 3L routines                         */
/*                                                                   */
/*********************************************************************/


/*   These are structures used only for 3COM board initialization */


    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_tran_buf;
        short size_tran_buf;
        long  ttl_tran_cnt;
        long  ttl_tran_err_cnt;
        long  ttl_tran_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;
    };
```

```
title netto3l.asm

;*****************************************************************************
;
;File: NETTO3L.ASM
;
;Description:  This file contains subroutines which provide a
;              C program with an interface to the 3L 1.0 routines.
;              Based on CTO3l.ASM.
;
;*****************************************************************************

; Functions called by C

PUBLIC  _getds
PUBLIC  _cInitParameters
PUBLIC  _cInitAdapters
PUBLIC  _cResetAdapter
PUBLIC  _cWhoAmI
PUBLIC  _cRdRxFilter
PUBLIC  _cWrRxFilter
PUBLIC  _cPutTxData
PUBLIC  _cGetRxData
PUBLIC  _cSetLookAhead


;Need to be written in C
;extrn   _myExitRcvInt    :near  SOME ERROR when written in C
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

lf      equ     0ah
cr      equ     0dh

; unused macros

@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
        int     21h             ;returns al: 0-nokey, ff-keyhit
        endm

CODE    GROUP   _TEXT, DATA, ICODE

_TEXT   segment byte public 'CODE'
DGROUP  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      ?

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
;                  parameters.
;
;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    InitParameters

        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

        mov     dl,0
        call    ResetAdapter

        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+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

        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 nnumber 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 nnumber 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:
;     int cGetRxData(&NumBytes, Flags, RequestID, PacketAddr)
;
;Input Parameters:
;     int NumBytes - The nnumber 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     dl,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 3.1 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

        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

TxProcess endp

;-------------------------------------------------------------------------
;
;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    si
;        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
;        pop     ds
;        pop     bp
        iret
ExitRcvInt endp

;_myExitRcvInt proc near
;        ret
;_myExitRcvInt 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
;
;-------------------------------------------------------------------------
RxProcess proc near

        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
RxProcess endp

;-------------------------------------------------------------------------
;
_TEXT   ends
        end
```

```
title stamp.asm

;*****************************************************************************
;
;File: STAMP.ASM
;
;Description:  This file contains subroutines which provide a
;              C program with an interface to the 3L 1.0 routines.
;              Based on CTO31.ASM.
;
;*****************************************************************************

; Functions called by C
PUBLIC  _getstamp
PUBLIC  _getclock
PUBLIC  _savvecs
PUBLIC  _fixvecs
PUBLIC  _getATimeStamp

lf      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
        int     21h             ;returns al: 0-nokey, ff-keyhit
        endm

CODE    GROUP   _TEXT, DATA, ICODE

_TEXT   segment byte public 'CODE'
DGROUP  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
;-----------------------------------------------------------------------
;
_etext  db      ?

vectsv  dd      22h dup (0)     ;save all vectors so we can cleanup
```

```
retsav  dw      ?
crlf    db      cr,lf,'$'

;-----------------------------------------------------------------------
;

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

;-----------------------------------------------------------------------
;

temp_hi         db      0
temp_lo         db      0
temp_hi_bit     db      0

_getstamp proc near

        xor     ax,ax
        out     043h,al
        in      al,040h
        mov     cs:temp_lo,al
        in      al,040h
        mov     cs:temp_hi,al
        mov     ah,cs:temp_hi
        mov     al,cs:temp_lo
        ret

_getstamp endp
;-----------------------------------------------------------------------
;


_getclock proc near         ; get lower two bytes from 0040:006c, clock data.
        push ds
        mov  ax,0040h
        mov  ds,ax
        mov  ax,ds:006ch
        pop  ds
        ret
_getclock endp

;-----------------------------------------------------------------------
;
;       This function returns a timestamp constructed of the Timer 0 value
;       and the lowest word of the MS-DOS clock.  The Timer 0 is a count-
;       down timer,  so it is converted to form a coherent timestamp value.
;       The Timer value is returned in the AX register (low word) and the
;       clock value is returned in the DX register (hi word).
;
_getATimeStamp   proc    near

                push    ds              ;set segment pointer for clock read
                mov     ax,0040h        ;
                mov     ds,ax           ;
```

```
                mov     al,0c2h         ;set up for count/status latch

                cli                     ;no ints here
                out     043h,al         ;latch
                mov     dx,ds:006ch     ;get clock lsw
                sti                     ;restore ints

                in      al,040h                 ;get status
                and     al,080h                 ;get msbit
                mov     cs:temp_hi_bit,al       ;store msbit
                in      al,040h                 ;get lsb of count
                mov     cs:temp_lo,al           ;store lsb of count
                in      al,040h                 ;get msb of count
                mov     ah,al
                mov     al,cs:temp_lo           ;get count into ax reg
                ror     ax,1
                or      ah,cs:temp_hi_bit       ;get back bit 16
                not     ax                  ;change from count-down to count-up

                pop     ds                      ;restore segment pointer
                ret

_getATimeStamp   endp

;-----------------------------------------------------------------------
;
;-----------------------------------------------------------------------
;
;  _Savvecs:  This procedure saves the interrupt vector table for restoration
;             after the adapter usage is completed.  The interrupt vector
;             table must be restored BEFORE initparameters may be called
;             again,  else the 55ms RTI handler will (at best) execute an
;             infinite loop.
;
;  _Savvecs returns nothing.
;
;-----------------------------------------------------------------------
;
_savvecs 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
        cli
        rep     movsw                   ;save 'em all
        sti

        pop     cx
        pop     di
        pop     si
        pop     es
        pop     ds
        ret
_savvecs endp

;-----------------------------------------------------------------------
;
```

```
;  _Fixvecs:  This routine restores the interrupt table portion saved
;             by _Savvecs.
;
;  _Fixvecs returns nothing.
;
;-------------------------------------------------------------------
;
_fixvecs 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
        cli
        rep     movsw                   ;restore 'em all
        sti

        pop     ax
        pop     cx
        pop     di
        pop     si
        pop     es
        ret
_fixvecs endp
_TEXT   ends
        end
```

```
/*****************************************************************************/
/*                                                                           */
/*  NETS.C                                                                   */
/*                                                                           */
/*  Description: This file is for network expansion.  If each different     */
/*               type of simulator is on a different network, each net       */
/*               must be separately initialized.  To convert this program    */
/*               into a multi-net system, the routines in SIMNET.C and       */
/*               ASAT.C become more important; these files contain network   */
/*               access routines, some of which are stubs for the most       */
/*               part because the tested configuration has both Asats and    */
/*               Simnet on the same ETHERNET connection.  Note, however,     */
/*               that this file assumes parallel functions for each          */
/*               network.  Also, the individual access functions (and        */
/*               related interrupt routines) are not defined here;           */
/*               this file merely provides global net control in a limited   */
/*               fashion.                                                    */
/*                                                                           */
/*  Routines:    InitNets   calls routines to save the interrupt vector      */
/*                              table and to initialize all networks.        */
/*               ResetNets  calls routines to reset all the networks.        */
/*               CloseNets  calls routines to reset all the networks and     */
/*                              reset the interrupt vector table.            */
/*               StatsNets  calls routines to display Network Statistics     */
/*                              for each network.                            */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/

#include <stdio.h>

/*************************************** declare TIMETO3L.ASM functions */

extern savvecs();   /* saves Interrupt Vector Table */
extern fixvecs();   /* restores IVT */


/************************************************** initialize all networks */
void InitNets()
{
        savvecs();              /* save IVT */
        init3COM();             /* init 3COM board */

/*        simnet_init_503();  */ /* init SIMNET connection (not implemented) */
/*        asat_init_503();    */ /* init ASAT connection (not implemented) */

}

/************************************************ reset all network connections */
void ResetNets()
{
        reset3COM();            /* reset 3COM board */

/*        simnet_reset_503();  */ /*reset SIMNET connection (not implemented)*/
/*        asat_reset_503();    */ /* reset ASAT connection (not implemented) */
}

/************************************************ close all network connections */
void CloseNets()
{
        ResetNets();  /* reset all network connections */
        fixvecs();    /* restore IVT */
}

/************************************************ display network statistics */
void StatsNets()
{
```

```
        stats3COM();              /* display 3COM ETHERNET board statistics */
/*        simnet_stats_503();   */ /* display SIOMNET connection stats (NYI) */
        asat_stats_503();         /* display ASAT connection stats */
}
```

```
/*****************************************************************************/
/*      simnet.c                                                             */
/*                                                                           */
/*                                                                           */
/*      This file contains the c code for the simnet M1 tank simulator.      */
/*                                                                           */
/*****************************************************************************/

#include "..\\sim.h"
/*
#define DISPLAY 1
*/

extern cRcvSome();
extern cXmit1();
extern char far *addr_3COM();

/* Initialize the 3COM EtherLinkii connection */
simnet_init_503 ()
{
    printf("simnet_init_503 Not Implement Yet\n");
}

char far *simnet_addr_503()
{
    return(addr_3COM());
}

/* Read a packet from SIMNET */
simnet_read_503 (buf)
unsigned char *buf;
{
    int cnt, i;
    unsigned char far *Pkt;

_disable();
    cnt = cRcvSome(&Pkt);
    printf("\ncnt = %d\n", cnt);

    if (cnt)
    {
    printf("Pkt = ");
    for (i=0; i<30; i++)
        printf(" %2x", Pkt[i]);
    }

    if  ((((Pkt[6] == 0x02) && (Pkt[7] == 0xc[illegible]) && (Pkt[8] == 0x1f) &&
          (Pkt[9] == 0x30) && (Pkt[10] == 0x2[illegible]) && (Pkt[11] == 0x68)) ||
         ((Pkt[6] == 0x02) && (Pkt[7] == 0xcf  && (Pkt[8] == 0x1f) &&
          (Pkt[9] == 0x30) && (Pkt[10] == 0x2[illegible]) && (Pkt[11] == 0x95)))  &&

          (Pkt[23] == 0x05) && (cnt))
    {
     /*    memcpy(ether_buf,&Pkt, cnt);*/
        for (i=0; i<cnt; i++)
            buf[i] = Pkt[i];
        printf("\ncopied message to buffer\n");
    }


  else
      { cnt=0;}

_enable();
    return (cnt);
}

/* Write a SIMNET pdu */
```

```
simnet_write_503 (Pkt, cnt)
unsigned char *Pkt;
int cnt;
{
    int i,j, flags = 0x0060, reqid = 0x0001,  nreqid = 0x0001;

    cXmit1(cnt, cnt, flags, reqid, Pkt, &nreq.d);

#ifdef DISPLAY
      j = 0;
      for (i = 0; i<cnt; i++)
        {
         j++;
         if (j > 20) { j = 1; printf("\n"); }
         printf("%2x ",Pkt[i]);
        }
        printf("\n");
#endif
}

/* Close SIMNET connection */
simnet_close_503 ()
{
    printf("simnet_close_503 Not Implement Yet\n");
}



/* This subroutine is for debugging purpose only, it will DUMP the content of a
   SIMNET pdu.  The content of the buffer must be in NETWORK ORDER */
simnet_dump (buf)
simnet *buf;
{
    int i, j, netcnt;
    unsigned char *pointr;
    union {
        struct {
            unsigned lengthr :12;
            unsigned version :4;
        } lengths;
        short lengthi;
        char  lengthc[2];
    } length;

    printf("SIMNET content\n");
    /* length does not include source addr. destination addr. and type field */

    printf("Source addr      : %2x-%2x-%2x-%2x-%2x-%2x\n",
        buf->ether_source [0], buf->ether_source [1], buf->ether_source [2],
        buf->ether_source [3], buf->ether_source [4], buf->ether_source [5]);
    printf("Destination addr : %2x-%2x-%2x-%2x-%2x-%2x\n",
        buf->ether_dest [0], buf->ether_dest [1], buf->ether_dest [2],
        buf->ether_dest [3], buf->ether_dest [4], buf->ether_dest [5]);

    pointr = (char *) &buf;
    for (i=0, j=3; i<158; i++, j++) {
        printf("%2x ", pointr[i]);
        if  (j >= 17) {
            j=0;
            printf("\n");
        }
    }
    printf("\n");
}


/* This subroutine is for debugging purpose only, it will DISPLAY the content of
a  SIMNET pdu.  The content of the buffer must be in HOST ORDER */
```

```
simnet_display (buf)
simnet *buf;
{
    int i, j;

    printf("Rotation\n");
    for (i=0; i<=2; i++)
        for (j=0; j<=2; j++)
            printf("%d %d %lf\n",i,j,buf->PDU.variant.appearance.rotation[i][j]);
    printf("Location\n");
    printf("%lf\n",buf->PDU.variant.appearance.location[0]);
    printf("%lf\n",buf->PDU.variant.appearance.location[1]);
    printf("%lf\n",buf->PDU.variant.appearance.location[2]);
    printf("%u\n",buf->PDU.variant.appearance.vehicleID.vehicle);
}
```

```
/*********************************************************************/
/*                                                                   */
/*  asat.h                                                           */
/*  This file contains the data structure for the ASAT               */
/*                                                                   */
/*********************************************************************/
/* #define F2M 5 */
//#define F2M 500.0
#define F2M 100000.0
#define ASAT_AIRPORT_X 160000/F2M
#define ASAT_AIRPORT_Y 0/F2M
#define ASAT_AIRPORT_Z 0/F2M

typedef struct {
    unsigned char ether_addr[6];
    unsigned short length;
    short reserved;
    unsigned short type;
} asat_hdr;

typedef struct {
    short x;
    short y;
    short z;
    unsigned short yaw;
    unsigned short pitch;
    unsigned short roll;
    short speed;
    /* ???? id; */
    short id;
} aircraft_init;

typedef struct {
    short x;
    short y;
    short z;
    unsigned short yaw;
    unsigned short pitch;
    unsigned short roll;
    short heading;
    short climb;
    short bank;
    float speed;
    char got_hit_data [5];
    char plane_type;
    char radar;
    short plane_hit_status;
} target;

typedef struct {
    long  x;
    long  y;
    long  z;
    long  speed;
    unsigned short yaw;
    unsigned short pitch;
} missile_simulator;

typedef struct {
    long  speed;
    long  x;
    long  y;
    long  z;
    unsigned short yaw;
    unsigned short pitch;
} missile_target;

char flare_type [6];
```

```
char flare_data [6];

typedef struct {
    char team;
} type_0;

typedef struct {
    short enemy_type;
    short team_mode;
    short ttl_enemies;
    short ttl_friends;
    char players;
    char skill_level;
    char missile_flag;
    char flare_flag;
    /* aircraft_init aircraft_init [12]; */
    char aircraft_init [200];
} type_1;

typedef struct {
    char sim_id;
    char sim_type;
    long  x;
    long  y;
    long  z;
    unsigned short  yaw;
    unsigned short  pitch;
    unsigned short  roll;
    long  speed;
    short hit_status;
    short radar_status;
    char order_table[11];
    short sim_status;
    short heading_rate;
    short climb_rate;
    short bank_rate;
    short missle_flare;
    char info[44];
    char missl_dat [8];
    /* missile_simulator missile_simulator[???]; */
    char missile_simulator[200];
} type_8;

typedef struct {
        char mig_cnts;
        short friend_enemy;
        /* char data[????]; */
        char data[200];
} type_9;

struct asat {
    asat_hdr hdr;
    union {
        unsigned char DATAONLY [200];
        type_0 TYPE0;
        type_1 TYPE1;
        type_8 TYPE8;
        type_9 TYPE9;
    } data;
}

typedef struct asat ruckasat;
```

```
/*****************************************************************************/
/*                                                                           */
/*      SIM.H                                                                */
/*                                                                           */
/*              This header file provides some basic defines and structures */
/*              used in creating and accessing SIMNET PDUs.                  */
/*                                                                           */
/*****************************************************************************/

/* include all SIMNET v.6 header files (in another directory) */

#include "\\simhdr\\simhdr.all"


/******* This came from SIMNET2.h ********************/

#define MYPLANEID       16
#define MAXBUF        8192
#define HEADER_SIZE     14  /* ethernet header size including our header */
#define MAXPKTSIZE    1514  /* total size of largest possible packet     */
#define HELICOPTER11    11
#define HELICOPTER12    12
#define A10              1

#define SIMNET_AIRPORT_X  40000.0
#define SIMNET_AIRPORT_Y  30000.0
#define SIMNET_AIRPORT_Z    220.0


typedef struct {
        char           ether_dest[6];
        char           ether_source[6];
        short          ether_type;
        char           association_header[8];
        SimulationPDU PDU;
} simnet;
```