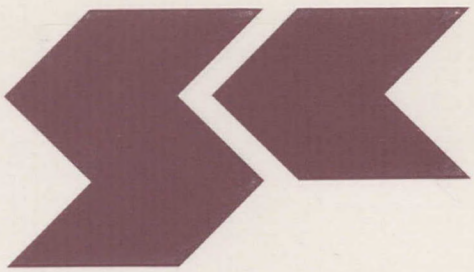
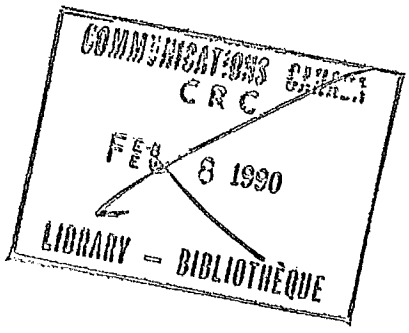


SOFTWARE TOP LEVEL DESIGN DOCUMENT FOR THE INTERNETWORK GATEWAY PROJECT

Submitted to: C.R.C.
Ottawa, Ontario

SKL Document #1500-15-010.03.0
Copy #4 05 May 1988



SOFTWARE TOP LEVEL DESIGN DOCUMENT FOR THE INTERNETWORK GATEWAY PROJECT

Submitted to: C.R.C.
Ottawa, Ontario

SKL Document #1500-15-010.03.0
Copy #4 05 May 1988

Software Kinetics

#1500-15-010.03.0

SOFTWARE TOP LEVEL DESIGN DOCUMENT

FOR THE

INTERNETWORK GATEWAY PROJECT

Contract No. 36001-06-3535/02-ST

05 May 1988

Prepared for:

Communications Research Centre
Ottawa, Ontario

Prepared by:

Software Kinetics Ltd.
65 Iber Road, P.O. Box 680
Stittsville, Ontario Canada
K0A 3G0

SKL Document #1500-15-010.03.0

Software Kinetics

#1500-15-010.03.0

Document Approval Sheet

for the

Internetwork Gateway Project

Document No: 1500-15-010.03.0

Document Name: Software Top Level Design Document for the Internetwork Gateway Project

Approvals	Signature	Date
Project Engineer:	R. D. Bradford	5 May 1988
Project Manager:	T. M. Symchych	May 5/88
Technical Authority:	P. Labbe - CRC	7 June 88

Software Kinetics

Document Revision History

Revision	Description of Changes	Origin Date
01	New Document Issued	22 May 1987
02	Added Software Options	24 August 1987
03	Coding and Integration Revisions	05 May 1988

TABLE OF CONTENTS

1.0 INTRODUCTION

The Top Level Design (TLD) of the Internetwork Gateway (IGW) software is presented in this document. The components of the design are identified and described. Also presented are the functions allocated to each component, the functional control and data flow among the components, global data, important local data, inputs, outputs, interrupts, and processing.

The IGW is a gateway between networks operating under the DARPA Internet protocols. Specifically, the IGW is a gateway between X.25 based TCP/IP networks and Ethernet based TCP/IP networks. The IGW supports the IP, ICMP, EGP, ARP, and X.25 protocols.

2.0 REFERENCED/APPLICABLE DOCUMENTS

1) 1500-15-002.01.0, "Requirements Specification for the Internetwork Gateway", Software Kinetics Ltd., 1987.

2) "VAX Architecture Handbook", Digital Equipment Corporation, 1981.

3) Defense Advanced Research Projects Agency, "Internet Protocol", DARPA Network Working Group Report RFC-791, USC Information Sciences Institute, September 1981.

4) Defense Advance Research Projects Agency, "Internet Control Message Protocol", DARPA Network Working Group Report RFC-792, USC Information Sciences Institute, September 1981.

5) Defense Advanced Research Projects Agency, "Exterior Gateway Protocol Formal Specification", DARPA Network Working Group Report RFC-904, M/A-COM Linkabit, April 1984.

6) Reltek Inc., "Q-Bus X.calibre FEP COMII-Q Technical/Users's Guide", Reltek Inc., 1985

7) Plummer, D., "An Ethernet Address Resolution Protocol", DARPA Network working Group Report RFC-826, Symbolics, September 1982

3.0 DESIGN

3.1 Software Architecture

The IGW consists of software components that are divided into two classifications: Boot Software, and Operating Software. The Boot Software is responsible for loading the Operating Software from the boot device. The Operating Software is responsible for performing all the gateway functions of the IGW, including input and output.

The Boot Software is composed of two Top Level Components (TLCs). The first TLC is the Primary Boot component which performs basic initializations of the gateway hardware, and then loads the second component, the Secondary Boot TLC. The Secondary Boot TLC is responsible for loading the Operating Software into the gateway memory, and then starting the Operating Software.

The Operating Software consists of a set of TLCs each having certain responsibilities to perform with respect to IGW operation. The operating TLCs are listed as follows:

1) IP - This TLC implements the IP protocol, the ICMP protocol, and packet filtering.

2) EGP - This TLC implements the EGP Internet protocol.

3) Operator Interface (OI) - The OI TLC presents a simple command line interface to the IGW operator to allow the operator to extract information and statistics and to modify IGW operating parameters.

4) Statistics (STAT) - STAT gathers statistics from the other TLCs of the IGW. In particular, the STAT TLC tallies packets to and from IP and X25 addresses, error packets sent and received, and data bytes

transmitted between hosts via IGW.

5) X.25 Device Driver (XDD) - This TLC controls the operation of the IXIB X.25 interface. The XDD passes packets to, and receives packets from the IXIB. It also collects status information from the IXIB and provides configuration parameters to the IXIB.

6) Ethernet Device Driver (EDD) - The EDD TLC manages the Ethernet hardware interface for the IGW. The TLC prepares packets for transmission and invokes the transmission facility, invokes the receive facility and collects received packets, and implements the ARP protocol for Ethernet - Internet address resolution.

7) Console Device Driver (CDD) - This TLC supports the VAX console device hardware which is used for the Operator Interface. The TLC supports the use of hardcopy or video terminals which support the ASCII character set.

8) Efficient Real-Time Executive (ERTE) - The ERTE is a small, real-time, multi-tasking executive which supports the preceding TLCs. Each of the above TLCs contains one or more processes whose execution the ERTE manages. The ERTE implements a combination of round-robin and priority based scheduling of processes. This permits processes to reasonably alternate execution while permitting critical processes, such as interrupt servers, to obtain priority service. The ERTE recognizes all interrupts and dispatches the correct interrupt server. The ERTE also supplies a message passing facility to allow processes to communicate with other processes. ERTE also supplies sleep, wake-up, and time-out facilities to processes.

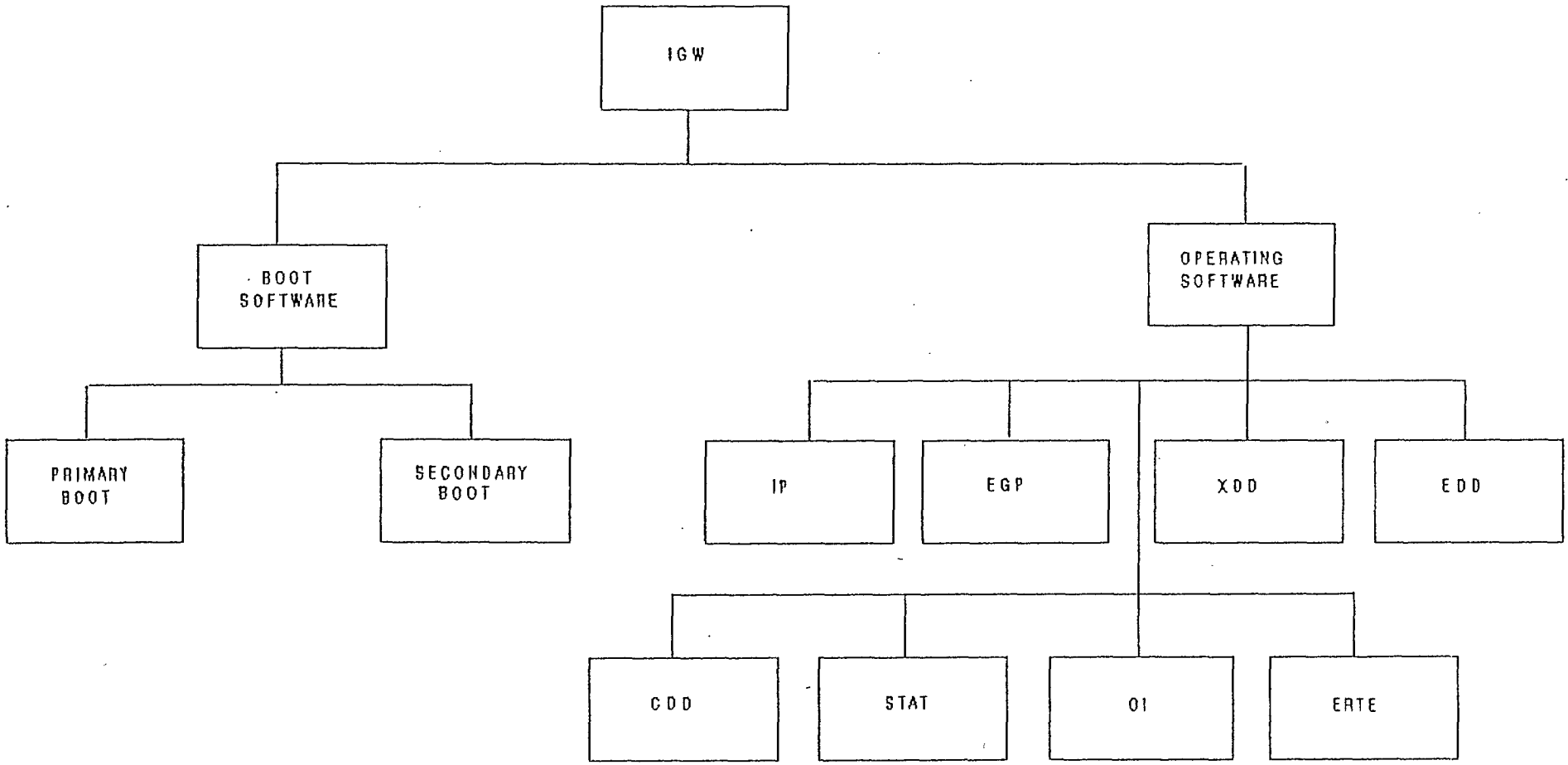


FIGURE 1
IGW SOFTWARE ARCHITECTURE

3.2 Functional Allocation

In this section the allocations of functions to the TLCs is presented. Each function is associated with one or more requirements from the Requirements Specification [1]. The associated requirements are also listed in this section.

3.2.1 Primary Boot

The Primary Boot TLC is a small portion of code which is stored in block number 0 on the boot device. It is automatically loaded from the boot device by VAX system software in read-only memory (ROM) each time the IGW is booted. This TLC's function is to find the secondary boot program on the boot device, load it into memory, and start it executing. The Primary Boot will relocate itself into high memory. The Primary Boot will load the Secondary Boot into low memory.

3.2.2 Secondary Boot

The Secondary Boot TLC will read all the operating software and configuration data and install it into the IGW memory. The Secondary Boot will also read the IXIB software and download this to the IXIB hardware. The Secondary Boot will establish all memory management tables and hardware for the IGW. Finally, the Secondary Boot will transfer control to the operating software, at which time, the memory occupied by the Secondary Boot and the Primary Boot will become available for use by the operating software.

The Secondary Boot TLC consists of two components: A Local Boot component and a Net Boot component. The Local Boot resides

entirely on the IGW local boot device and it in turn expects all software and data required for booting the IGW to reside on the boot device. The Net Boot component resides partly on the IGW boot device, and partly on a cooperating host on the Ethernet to which the IGW is attached. The Net Boot permits the IGW to receive its software and configuration data from the cooperating host via the Ethernet. Only one of these components resides on the boot device at one time.

The Net Boot component consists of two distinct elements: The IGW Net Boot and the Host Net Boot. The IGW Net Boot sends a signal to the Host Net Boot requesting that the Host Net Boot supply the IGW with a loadable image. The Host Net Boot, upon receiving the signal from the IGW Net Boot builds an image of the IGW with all software and configuration data in place, and then sends this image to the IGW. The IGW Net Boot loads this image into memory. Once the image is loaded, the IGW Net Boot transfers control to the operating software.

The requirements listed in the Requirements Specification [1] in Section 3.1 paragraphs 2 and 4, and in Section 3.2.2 paragraph 3 apply to the Secondary Boot TLC. The applicable requirements are shown in Table 1, the Requirements Cross-Reference Table.

Table 1: Requirements Cross-Reference Table

REQUIREMENT		TOP LEVEL COMPONENT											
Section #	Paragraph #	N/A	Pri. BOOT	Sec. BOOT	ERTE	IP	EGP	OI	STAT	ROUTE	Ether Drvr	X.25 Drvr	Cons Drvr
1.0	-	X											
2.0	-	X											
3.0													
3.1	1				X	X	X	X	X	X	X	X	X
	2		X	X	X	X	X	X	X	X	X	X	X
	3				X			X					X
	4		X	X	X								
	5							X	X				X
	6								X				
	7							X		X			
3.2													
3.2.1													
3.2.1.1	1 (1.)					X					X	X	
	1 (2.)					X	X				X		
	1 (3.)						X						
	1 (4.)					X					X	X	
	1 (5.)					X	X						
	1 (6.)					X	X	X	X				
	2					X					X	X	
	3				X	X		X			X	X	X
3.2.1.2	1	X											
3.2.1.2.1	All					X							
3.2.1.2.2	All					X							
3.2.1.2.3	1						X						
3.2.1.2.4	1										X		
3.2.1.3	1					X							
	2					X		X					X
	3					X							
3.2.1.4	1						X						
	2										X	X	
3.2.2	1	X											
	2				X			X					X
	3		X	X	X			X					
3.2.3	1							X	X				
	2 (1.)					X			X				
	2 (2.)					X			X				
	2 (3.)								X			X	
	2 (4.)					X			X				
3.2.4	1 (1.)							X				X	
	1 (2.)						X	X		X			
	2							X		X			
	3							X					
3.2.5	1				X			X					
3.2.6	1							X					X
3.3		X											

Table 1: Requirements Cross-Reference Table — Page 2 of 2

REQUIREMENT		TOP LEVEL COMPONENT											
Section #	Paragraph #	N/A	Pri. BOOT	Sec. BOOT	ERTE	IP	EGP	OI	STAT	ROUTE	Ether Drvr	X.25 Drvr	Cons Drvr
3.4													
3.4.1	1											X	
	2											X	
3.4.2	1										X		
	2	X											
3.4.2.1	1												
3.4.2.2	1					X					X		
3.4.2.3											X		
3.4.2.4	1					X							
	2					X							
	3					X							
4.0		X											

3.2.3 IP

The IP TLC is responsible for the implementation of the IP protocol, the ICMP protocol, and packet filtering. The IP TLC will accept datagrams from any network interface and forward them, as permitted by the packet filter facility, to the interface appropriate for the next hop on the datagram's route. The IP TLC will also refer to connectivity and reachability information gathered by the EGP TLC and it will use the information for routing datagrams.

The IP TLC will be applied to many requirements defined in the Requirements Specification [1]. The applicable requirements are shown in the Requirements Cross-Reference Table, Table 1.

3.2.4 Exterior Gateway Protocol (EGP)

The EGP TLC is responsible for the implementation of the EGP protocol. The TLC will communicate with its neighbour "core" gateways to determine host and network connectivity and reachability information. The TLC will build a table to contain the collected information, and this table will be available to other TLCs as needed.

The requirements to which the EGP TLC will be applied are shown in Table 1, the Requirements Cross-Reference Table.

3.2.5 Operator Interface (OI)

The OI TLC presents a simple command line type interface to the IGW operator. The interface supports commands which: 1) display IGW link, node, and route information; 2) display IGW traffic statistics; and 3) modify IGW parameters for the control of gateway operations.

The Requirements Cross-Reference Table, Table 1, shows the requirements that the OI TLC will address.

3.2.6 Statistics (STAT)

The STAT TLC is responsible for gathering IGW traffic data and calculating traffic statistics. The IGW TLC's responsible for handling datagrams through the gateway will extract traffic information and will pass it to the STAT TLC where it will be stored and processed. Processed data are transferred to the OI TLC when requested. The requirements addressed by the STAT package are listed in Table 1.

3.2.7 X.25 Device Driver (XDD)

The X.25 Device Driver is the TLC which passes IP packets to the IXIB for transmission over an X.25 network. The XDD also receives IP packets from the IXIB. Further, the XDD transfers status information from the IXIB and loads configuration and control information (particularly, the Internet-X.25 address translation table) to the IXIB. Finally, the XDD will extract X.25 traffic data and pass them to the STAT TLC. Table 1 shows the requirements that will be covered by the XDD TLC.

3.2.8 Ethernet Device Driver (EDD)

The IGW Ethernet interface is supported by the Ethernet Device Driver (EDD) TLC. The EDD supports the transfer of IP packets to and from the Ethernet hardware interface. The EDD also implements the Address Resolution Protocol (ARP) and the encapsulation of IP packets in Ethernet packets. Finally, the EDD tracks the status of the interface and extracts information for use in reporting traffic statistics. The complete list of requirements covered by the EDD is presented in Table 1, the Requirements Cross-Reference Table.

3.2.9 Console Device Driver (CDD)

The CDD TLC controls the hardware interface to the operator's console device. The CDD will support a general terminal abstraction which could be either a CRT or a hard-copy terminal. A keyboard is assumed to be the input device on the terminal. The CDD will only support ASCII terminals. The CDD TLC will support full duplex transmissions up to 9600 bits per second and the XON/XOFF flow control protocol. Table 1 presents the requirements which will be covered by the CDD TLC.

3.2.10 Efficient Real-Time Executive (ERTE)

The ERTE TLC is a small real-time executive, or kernel, which implements a multi-tasking abstraction. The ERTE controls the execution of "processes" which are independent tasks that make up the TLCs presented earlier. The ERTE offers a set of services to these processes to assist them in their operation. The main service is a message passing service which processes use to communicate with one another. Other services include sleeping for a period of time, waiting for an event (both external interrupts and internal process events are supported), setting or clearing a process event, reading the clock, and allocating memory.

The ERTE will execute its functions as quickly as possible. The services offered will be as simple as possible to facilitate the need for quick operation. The ERTE will also make extensive use of hardware facilities which support context and task switching, and memory management.

The IGW requirements covered by the ERTE are shown in Table 1, the Requirements Cross-Reference Table.

3.3 Memory Allocation

Within the IGW, memory will be allocated as follows (Figure 2):

1) The lowest pages of memory will be used as the System Control Block (SCB) which contains all the vectors used for dispatching interrupt and exception service routines (Reference [2]). The SCB is allocated in physical memory only.

2) The System Page Table (SPT) (Reference [2]) will follow the System Control Block in physical memory. The size of the SPT will be fixed, and will be allocated enough memory such that the entire portion of the VAX system virtual address space allocated to the IGW software will be mapped to physical memory. The SPT will be allocated physical memory only.

3) Following the SPT in physical memory, the IGW Link Area (ILA) provides the IGW software with references to global data items. The ILA occupies the lowest pages of VAX system virtual address space, a location all IGW processes can reference. The contents of the ILA are described in Section 3.5, Global Data.

4) The ERTE TLC software and data areas are loaded following the ILA. The ERTE is allocated space in physical memory and the system virtual address space immediately after the ILA.

5) The IGW processes follow the ERTE. Each process is allocated physical memory one after the other. Each process is also allocated space in its own process virtual address space, beginning at address 0. The Process Page Table (PPT) for each process follows the process in physical memory. The PPT is also allocated space in system virtual address space following the system virtual address space allocated to the ERTE (Reference [2]). Finally, each process is allocated two stacks which are located in physical memory immediately following the process's PPT memory. Page Table Space in the system virtual address space is allocated for these stacks.

6) The global tables referenced by IGW software reside in physical memory following the IGW processes. These tables are also allocated space in the system

virtual address space following the PPT space discussed in 5 above.

7) The VAX interrupt stack follows the global tables in physical memory. The stack also follows the tables in the system virtual address space.

8) Any physical memory not allocated above becomes free memory which is used for message passing and dynamic memory allocation by processes. Each free physical page is associated with a page in system virtual address space in the area following the interrupt stack.

9) The I/O physical address pages are associated with system virtual address pages in the space immediately following the free memory virtual address space.

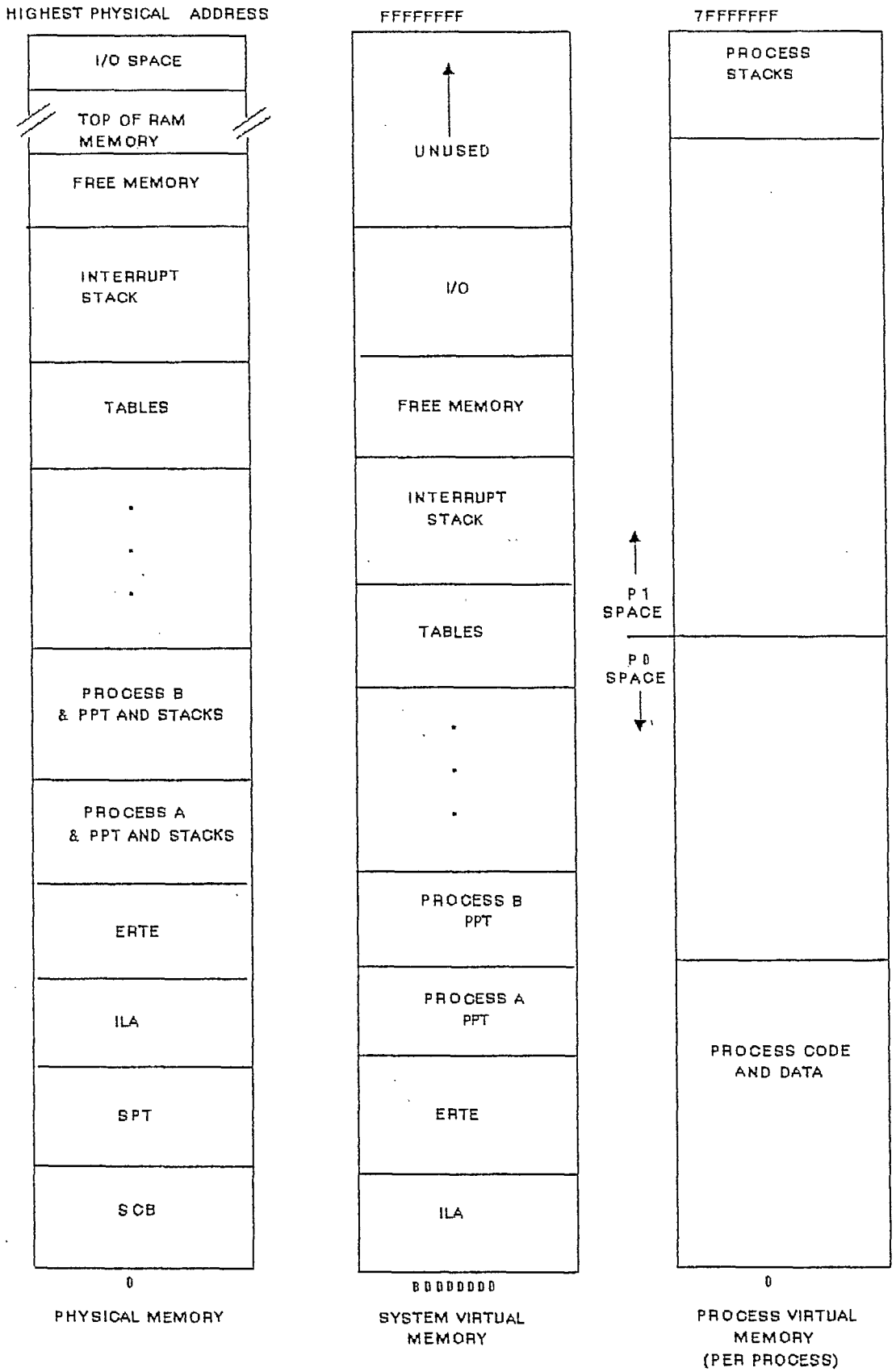


FIGURE 2
MEMORY USAGE

All memory is allocated by pages, where a page is 512 bytes (Reference [2]). IGW programs and data reside in VAX virtual memory, which is mapped to physical memory via the VAX memory management page tables. Some VAX structures, such as the SCB and SPT, do not reside in virtual memory. These structures are not considered part of IGW programs and data. Refer to Reference [2] for a complete description of VAX architecture.

Within the IGW, each process or task will be allocated at boot time all the memory needed for the process' code and static data areas. The ERTE will provide additional memory dynamically to processes upon request. The additional memory is allocated from free memory. There are two ways memory may be allocated. One is via an explicit request for memory, and the other is by allocation of message buffers.

Processes requiring additional memory make explicit requests to the ERTE to supply the required memory. Each process will be supplied the memory requested from the free memory pool. The memory is permanently allocated to the process; there is no function to release memory back to the free memory pool.

Memory can also be obtained from the ERTE through the message system calls. Each message buffer in the system has a fixed amount of memory attached to it. Processes have this memory available to them as long as the message buffer is allocated to them. Processes lose the memory when they free the message buffer or send the message buffer to another process. Processes also obtain memory in message buffers when they receive message buffers from other processes.

When the number of message buffers is defined, care will be taken

to ensure that enough free memory will be available in the system to supply all the message buffers and have some memory left over for processes to allocate. Each message buffer will require enough memory to hold the largest packet allowed by the IGW. A configuration parameter will be used to specify the maximum number of message buffers that are available for use. The number of message buffers defined will be large enough to ensure that throughput is not affected when the IGW must process many small packets.

3.4 Functional Control and Data Flow

3.4.1 Control Flow

Control flow among TLCs in the IGW is controlled through the ERTE TLC (Figure 3). All TLCs (excluding the ERTE and Boot TLCs) are composed of one or more processes to which control is granted by the ERTE. These processes then execute until one of three events occurs:

1) The process issues a "system call" to the ERTE: a system call is a request for a service offered by the ERTE;

2) An external interrupt occurs; or,

3) An exception (a fault in hardware or software, such as Power Failure, Arithmetic Fault, Privileged Instruction, etc) occurs.

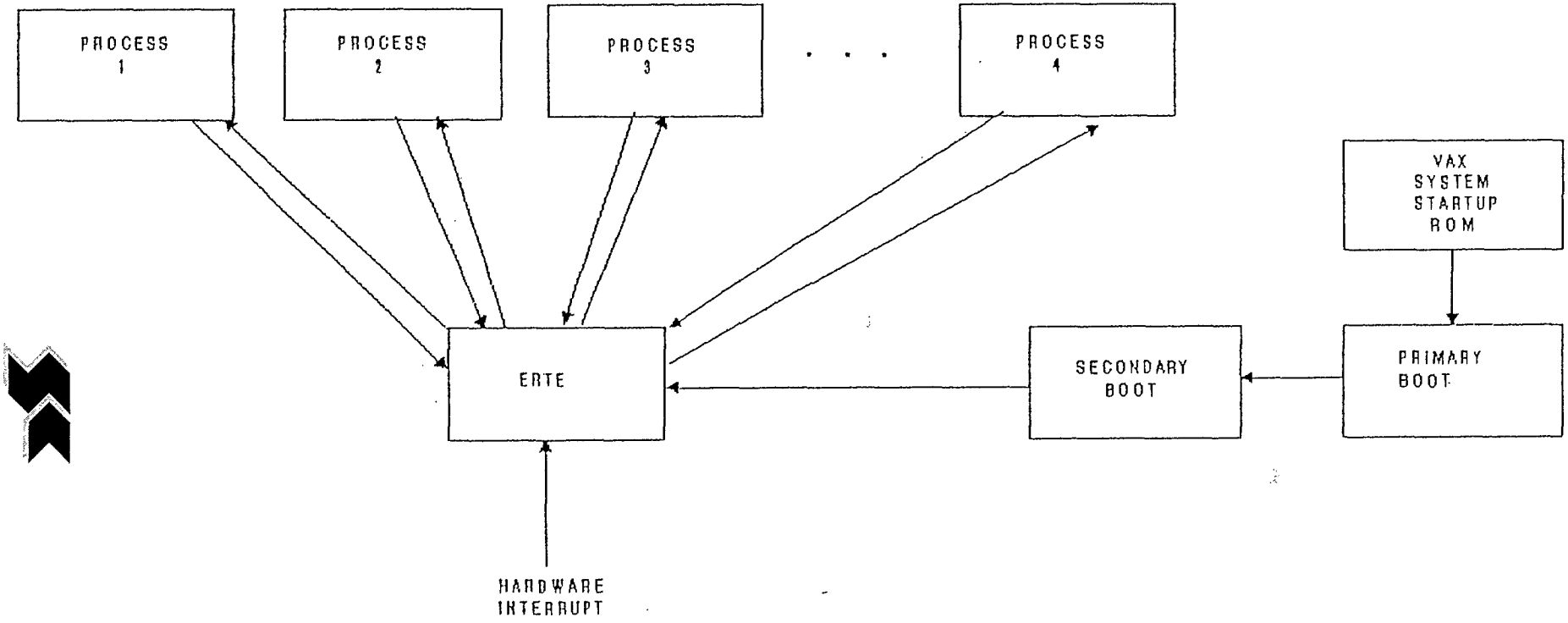


FIGURE 3
IGW FLOW OF EXECUTION CONTROL

Each of these events causes the process execution to be suspended and the ERTE to begin executing to handle the event. In the case of a system call, the calling process generates the equivalent of a software interrupt, which triggers the hardware to suspend process execution and to transfer control to the ERTE. The ERTE then examines and processes the request in the system call. The ERTE then selects the next process to run (which may be the process which issued the system call) and transfers control to it.

In the case of an external interrupt or exception, the event also triggers the hardware to interrupt process execution and to transfer control to the ERTE TLC. The ERTE then processes the event, selects the next process to run, and transfers control to the process. For external interrupts, the ERTE signals that an internal event corresponding to the interrupt has occurred. Any processes waiting on the internal event will receive notification of the event. For exceptions, the ERTE may or may not take action. The action taken depends on the exception. Serious exceptions, like power failure detection, cause the IGW to reboot, while non-fatal exceptions, like Arithmetic Fault, cause only a console message to be displayed.

In all cases when a process is interrupted, enough state information is saved by the ERTE to permit the process to resume executing from exactly the point of interruption. The process will not require any special code to handle unexpected interruptions.

Flow of control at boot time differs from that described above. When a boot is initiated, the VAX system Read-Only Memory (ROM) receives control first. The ROM routines perform memory tests

and other simple hardware verification checks. It then reads the first block on the boot device and loads that block into memory and transfers control to it. The block will contain the Primary Boot TLC; thus, the Primary Boot TLC will receive control from the ROM.

As indicated earlier (Section 3.2.1), the Primary Boot TLC transfers control to the Secondary Boot TLC once the Secondary Boot has been completely loaded. The Secondary Boot then loads the operating software and, finally, transfers control to the ERTE TLC. Once ERTE receives control from the Secondary Boot TLC, ERTE initializes its process queues, selects the first process to execute, and transfers control to that process.

3.4.2 Data Flow

Data flow between the TLCs of IGW is accomplished two ways: by messages, and by global data areas. Messages form the most common method of transferring data between TLCs. Under the message passing system, the sender creates the message privately, and then sends the message to the receiver, who is immediately able to read and process the message. Global Data areas are used to store data that must be available to most or all of the TLCs.

Each TLC exchanges data with one or more TLCs or hardware interfaces as shown in Figure 4. All exchange is accomplished via the message passing system with the following exceptions. Exchanges with Global Tables are accomplished by reading or writing global data areas containing the global tables, while exchanges between drivers and I/O hardware are accomplished via the corresponding I/O interface registers and procedures.

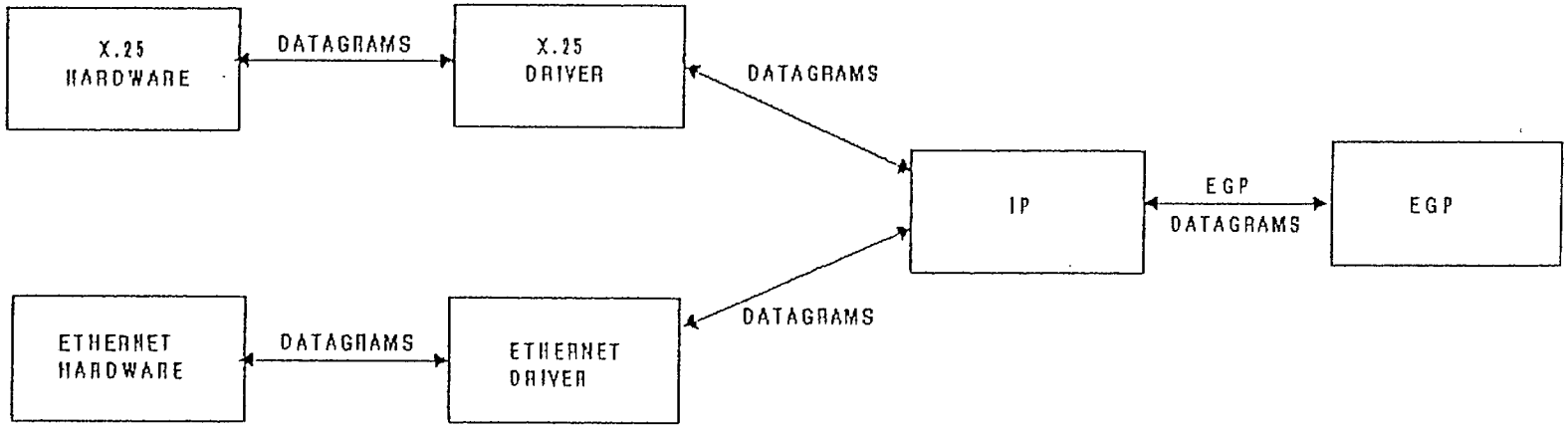


FIGURE 4A
DATAGRAM DATA FLOW

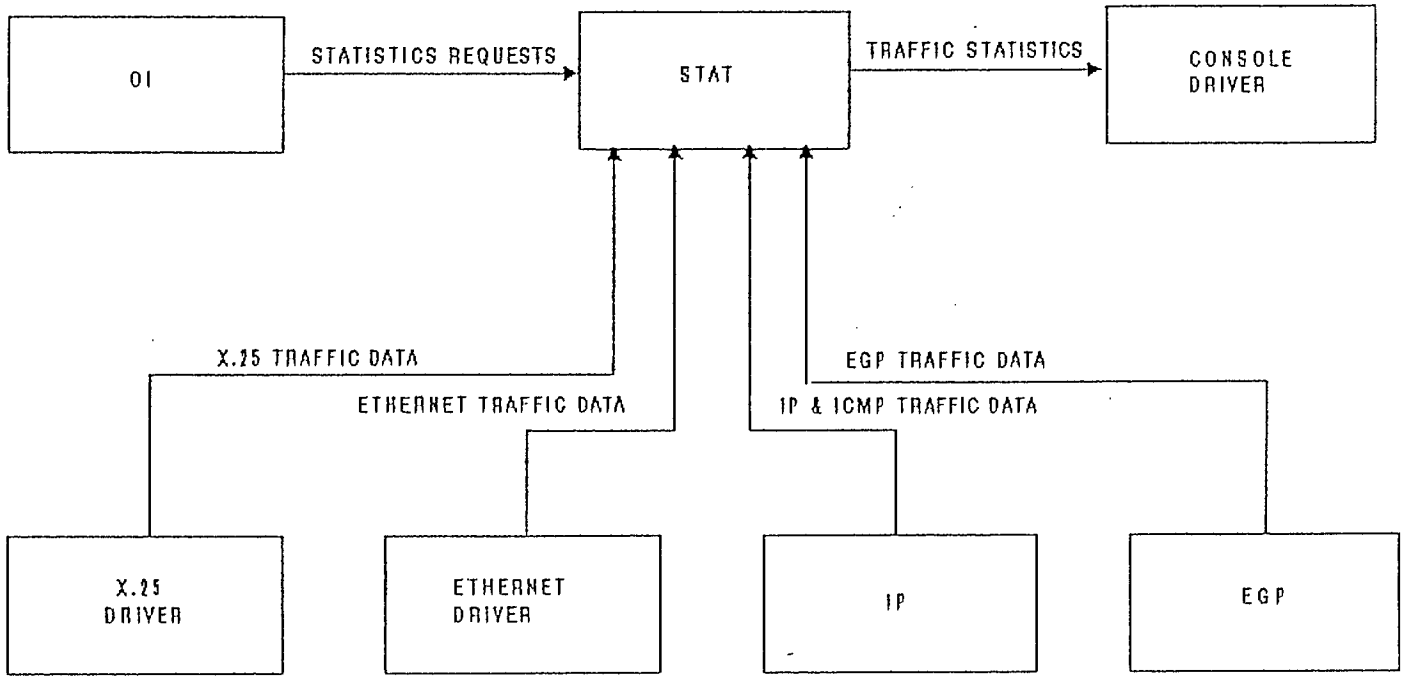


FIGURE 4B
TRAFFIC DATA AND STATISTICS DATA FLOW

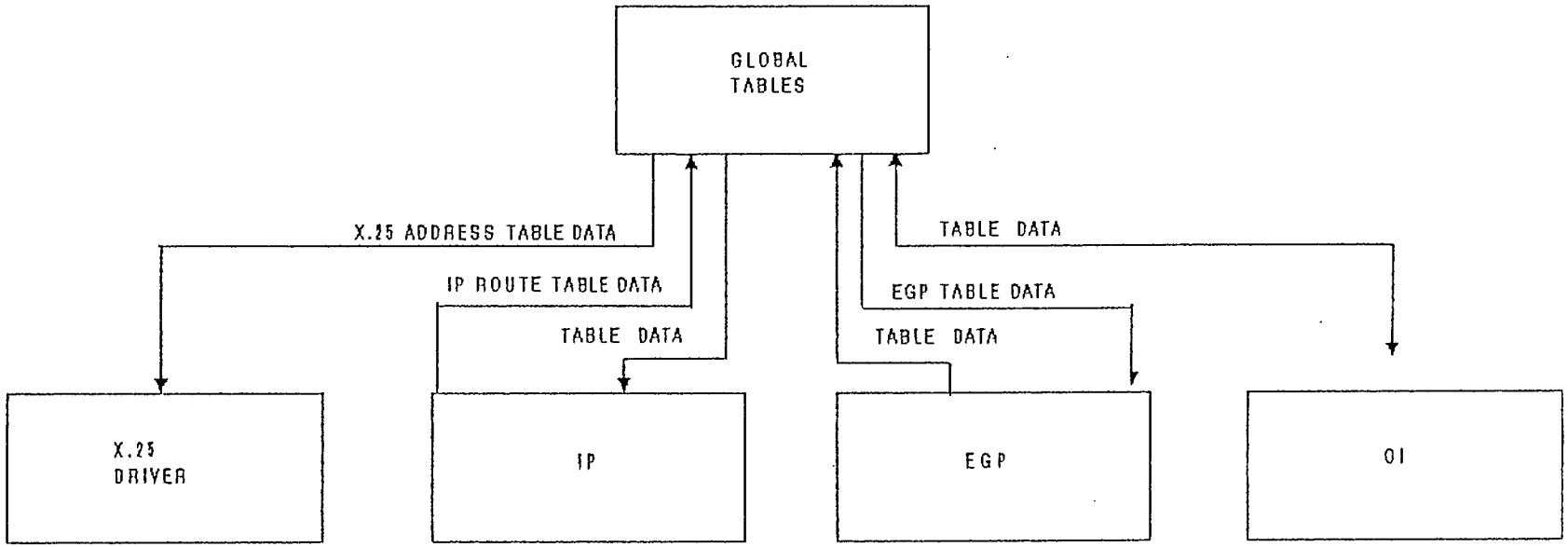


FIGURE 4C
TABLE DATA FLOW

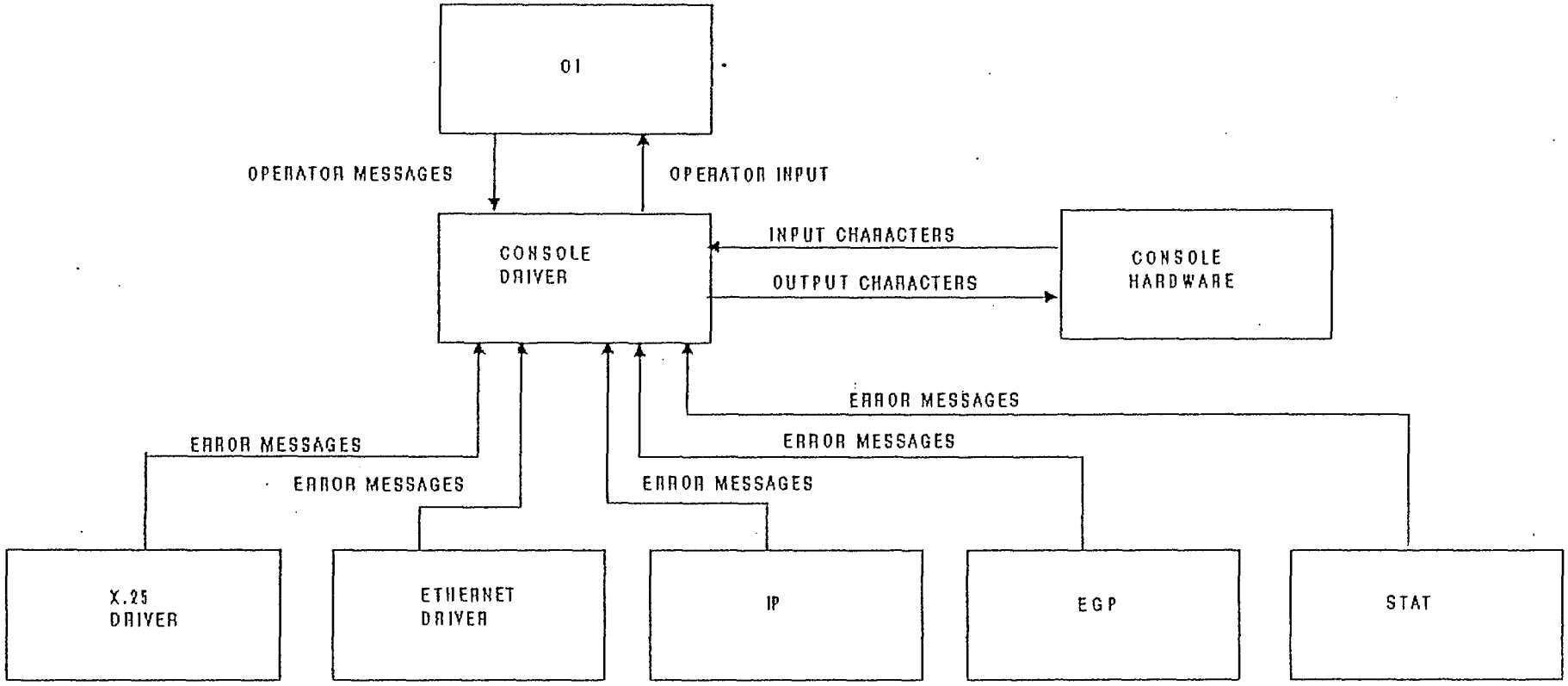


FIGURE 4D
CONSOLE DATA FLOW

3.4.2.1 Datagram and Route Data

Datagrams enter the system via either the Ethernet or X.25 I/O hardware and are collected by the corresponding driver. The driver then passes the datagram to IP through the message passing system. IP determines the appropriate action to be taken and, if a datagram is to be transmitted, IP prepares the datagram and passes it to the appropriate driver in a message buffer. The driver then passes the datagram to the corresponding hardware for transmission via the hardware I/O interface.

If IP receives an EGP datagram, IP passes the datagram on to EGP in a message buffer. EGP may generate datagrams of its own, which it will pass to IP in a message buffer. EGP, based on the data it receives in datagrams, may update tables containing EGP routing and neighbour gateway information. Such updates are accomplished by writing the global data areas that contain the tables. When EGP does update the tables, it sends a message buffer to IP to indicate that it has done so.

IP also updates a route table. It uses information from EGP and other tables (see Section 3.5) to build the IP routing table. Each time IP receives a message from EGP indicating that EGP has updated the EGP tables, IP rebuilds the IP routing table. IP builds its table in the global data area and it also reads other table data from the global area.

3.4.2.2 Traffic Data

IP, EGP, and the X.25 driver will transmit traffic data to the STAT TLC. IP tracks the number of packets and the number of bytes transmitted through IGW in each direction between pairs of Internet addresses. Periodically, IP will send what it has accumulated to STAT where the raw data will be reduced and tabulated. IP sends such data in message buffers.

Similarly, EGP tracks EGP packet types sent and received between itself and other gateways. Periodically, EGP transfers this data to STAT in a message buffer.

The X.25 driver also transmits traffic data to STAT in message buffers. The driver tallies datagrams sent and received to and from each X.121 address. The driver also collects status and data from the IXIB board and transfers the information to STAT. The IXIB data includes calls made and X.25 packets transferred to and from X.121 addresses.

ICMP traffic data is collected by IP and transferred to STAT in message buffers. The ICMP data collected is counts of ICMP message types sent and received. The IP TLC collects this data because the ICMP protocol is implemented in the IP TLC.

3.4.2.3 Operator Interface Data

The data collected and reduced by STAT is available for display on the operator's console. The STAT TLC transfers requested statistics to the Console Device Driver (CDD) TLC when requested to do so by the Operator Interface (OI) TLC. The display request is contained in a message buffer from the OI. The statistics sent by STAT to the CDD are also sent in message buffers.

The OI also manipulates the route tables in the global data areas. The OI may modify, add or delete entries, or may simply read the data for display on the operator's console.

To communicate with the operator, the OI exchanges messages with the console driver. The OI sends messages containing data or text to be displayed, and the driver receives messages containing terminal input data typed at the operator's console. The console driver sends and receives data to and from the console via the console hardware interface.

Each of the other TLC's may also send data (in message buffers) to the console via the console driver. Such data is expected to be error messages and will be displayed on the console exactly as received by the console driver.

3.5 Global Data

Each TLC (and each process within each TLC) will have access to four types of global information. All global information will reside in System Virtual Address (SVA) space which will be readable by all processes and TLCs. Some global information will also be writable by all TLCs, while some information will be write protected (ie. read-only) to all but the ERTE TLC, using the protection features of VAX memory management facilities.

The first type of global data is the route tables. These tables are of fixed size and contain route information required or produced by IGW processes. There are seven tables which are:

1) Network Table - A table containing information on directly connected networks.

2) Gateway Table - A table of known gateways to be used in addition to those acquired by EGP.

3) Neighbour Table - A table of neighbour gateways that implement EGP.

4) EGP Route Table - A table built by EGP which contains a list of network numbers and gateway addresses to use to reach those networks.

5) IP Route Table - A table built by IP to select the next hop for a datagram along a route.

6) IXIB Address Configuration Table - A table used to translate IP addresses to X.121 addresses for datagrams to be sent over the X.25 network.

7) Packet Filter Table - A table containing a list of packet filter entries which are used to administer IP packet flow through the Internet Gateway.

The Network Table is read by EGP to determine networks that the IGW will report to its neighbours. The table will also be read

by IP when it produces the IP Route Table. The table is read by OI to display network interface data on the console. The OI also writes the table to modify the characteristics of one or more interfaces (eg. marking the interface up or down).

The Gateway Table contains a list of gateways available to the IGW, in addition, to those that are acquired by EGP. There is one entry for each network for which a gateway is indicated. Each entry has the following fields:

1) Gateway Address: the address of the gateway to route packets for the network indicated by the Network Address field;

2) Network Address: the network reached by the gateway specified by the Gateway Address field;

3) Mask: an address mask used to indicate network class or to indicate a subnet;

4) Flags: a set of indicators related to the entry, which specify the status of the entry (ie. Valid/Invalid, Report on EGP/Don't Report on EGP);

5) Hops: a count, used by EGP, which indicates the number of gateways that must be crossed to reach the destination; and,

6) Interface: the number of the interface to use to start a packet towards the gateway.

The Network Table contains an entry for each network interface in the gateway. Each entry contains:

1) the Maximum Transmission Unit (MTU, maximum IP packet size for the connected network);

2) the address of the IGW on the network;

3) the address mask for the network (used to indicate the network's address class, or to support subnetting);

4) a list of ten interfaces which access the network (the interface is represented by the message queue number used to send data to the interface's driver,

5) the current interface (the next datagram to the network will be sent by this interface)

6) interface flags used to indicate the state of the interface (eg. UP/DOWN, Report on EGP/Don't report on EGP).

The EGP TLC reads this table when building responses to poll requests from other EGP gateways. IP reads the table when it builds the IP Route Table. The OI reads the table to display the data on the console. The OI also writes the table. The OI can modify an existing table entry, add a new entry, or delete an entry. Deletion is accomplished by marking the entry as Invalid. This avoids the complications of removing an entry completely from the table. The entry can easily be added back by marking the entry valid again.

The Neighbour Table is a list of neighbour gateways that implement EGP that the IGW knows about. Each entry in the table represents a gateway that is a neighbour to the IGW. At startup time, the table contains a list of gateways that the IGW should know about immediately after startup. During operation, the table contains all neighbours encountered by by EGP, plus the entries at startup.

A Neighbour Table entry has the following fields:

1) Gateway Address: the address of the neighbour gateway;

2) Flags: indicators of the state of the entry (ie. Valid/Invalid, UP/Down/Acquiring); and,

3) Two Time Fields: a pair of time values inserted by EGP as part of its neighbour reachability procedures.

The Neighbour Table is read by both EGP and OI. EGP uses the table data as part of its neighbour reachability procedures, while OI displays the table data on the operators console. Both EGP and OI update the table. EGP adds new neighbours to the table, and marks the state of the table entries as they change. The OI may add new entries, modify existing entries, or delete entries (deletion is effected by marking the entry Invalid) as requested by the IGW operator.

The EGP Route Table is a table built by EGP to reflect the network reachability information accumulated by EGP. The table is a list of address pairs, the first being a network address and the second being a gateway address. A given pair indicates that the given network may be accessed via the given gateway.

OI and IP also refer to this table. OI will display the table on the operator's console, while IP uses the table when constructing the IP Route Table. Only EGP modifies the table, depending on information received from its neighbours.

The IP Route Table is a table built by IP for use in determining routes. IP collects information from the above tables and accumulates it in the IP Route Table. Only IP writes to this table, but OI may read it and display it on the console. IP recalculates the table whenever it is notified of change in one of the preceding four tables. The TLC changing one of these tables is responsible for notifying the IP TLC.

The table entries consist of five fields:

1) Flags: indicators of the state and other status information pertinent to the entry. Flags represent items such the validity of the entry (on or off state), whether the indicated network (see below) is directly connected, and whether EGP calculated the route or not;

2) Network Reference: this links the route to one of the IGW network interfaces;

3) Mask: a mask used for extracting address class or subnet information from an address;

4) Gateway Address: the gateway to route packets to reach the network in the Network Address field; and,

5) Network Address: the address of the network for which this table entry indicates a route.

The IXIB Address Configuration Table is used to indicate the mapping of IP addresses to X.121 address for packets entering an X.25 network. The X.25 Driver reads this file and loads it into the IXIB at boot time. The OI also reads this table and displays it on the console. The OI also adds, deletes, and modifies entries in the table as indicated by user requests. After changing the table, the OI signals the X.25 driver who reloads the table into the IXIB.

Each table entry has four fields:

1) X.121 Address: the X.121 address for the host at the given IP address;

2) IP Address: the Internet address of the remote host on the X.25 network;

3) Maximum Packet Size: the maximum size of the X.25 packets to the X.121 destination address; and,

4) Flags: indicators used to control IXIB operations related to the X.121 addresses (ie. request reverse charging, accept reverse charging, reject outgoing calls, reject incoming calls, and remote has IXIB

indicators).

Finally, the Packet Filter Table is read by both IP and OI. IP uses the table data to administer the flow of data through the Internet Gateway, while OI displays the table data at the operator console. The table contains list of address pairs and a mode for each pair. The mode for each pair may be either "allow" or "restrict". The allowance mode specifies that for a packet whose source and destination addresses match the table entry in the Packet Filter Table, the packet is allowed to flow through the IGW. The restrictive mode specifies that for each packet matching the table entry in the Packet Filter Table, the packet is not allowed to flow through the IGW, and is discarded by the IGW. The packet filter table is loaded at system startup. The operator may add new entries, modify existing entries, and delete existing entries in the packet filter table.

An address in the Packet Filter Table may be either a host or network address. This permits filter entries to specify net-to-net, net-to-host, or host-to-host restrictions or permissions. When searching the table, the most specific address match is used. In this way, a packet which matches a net-to-net entry and a host-to-host entry will be processed based on the host-to-host entry. This method allows selective exceptions to be specified to particular rules. For example, if Net A and Net B are restricted, it is possible to give Host X on net A and Host Y on Net B specific permissions over and above the general ban imposed on Net A and Net B by specifying a table entry for Host X and Host Y with allowance mode.

Each table filter entry has three fields:

1) IP Address A: the Internet address of a host or network which together with IP Address B define IP source and destination addresses.

2) IP Address B: the Internet address of a host or network which together with IP Address A define IP source and destination addresses.

3) Mode: specifies whether packets are allowed or restricted between Address A and Address B.

The second type of global data is the IGW Link Area (ILA). This area, created at boot time, contains pointers to other structures and global data. There are pointers to each of the global table, pointers to the IO pages, and pointers to process control information. The global table pointers are used by all TLCs to locate where the global tables have been loaded.

The IO page pointers are pointers to the SVA pages that represent the IO pages. All TLCs will determine the addresses of IO devices from these pointers.

Finally, the pointers to the process control information permit the ERTE to locate all the process related information needed for scheduling tasks. ERTE will use the process information to allocate the CPU to IGW processes as described in Section 3.4. The ERTE TLC will also update the process related data as required to reflect changes of state and status with the IGW.

The ILA area will be available on a read only basis to all TLCs except ERTE. Only ERTE may modify the ILA area. This protects the pointers and process data from accidental destruction by erroneous processes.

The third type of global data is the IO pages. All the physical

IO pages will be mapped into SVA space and will, thus, be accessible to processes. Input/Output processes will be designed to only access those IO addresses for which they have responsibility. Other processes will not access the IO pages.

The fourth and last type of global data is free memory. As free memory is part of SVA space, all processes may access it freely. Processes will be designed to request free memory from ERTE, and then to use only that allocated by ERTE.

3.6 Design

3.6.1 Primary Boot TLC

The Primary Boot TLC contains the software required to load a Secondary Boot Program from disk into the main memory of the MicroVAX, and to start the execution of the Secondary Boot Program. The Primary Boot software is to be written in VAX assembler as compact code and hardware register access are required.

The software contained in the Primary Boot TLC is to reside on the first 512 bytes of the boot device (RX-50 diskette). These first 512 bytes of the diskette are automatically loaded and executed by system software located on boot ROMs on the MicroVAX.

3.6.1.1 Inputs

The following inputs are required by the Primary Boot TLC:

1) Secondary Boot Program - This input is read from the 7.5K bytes on the boot device immediately following the Primary Boot Program.

2) Q-BUS I/O Page Address - This input is provided by the MicroVAX boot ROMs through register R1, and contains the starting address of the I/O page on the Q-BUS.

3) Boot Device CSR - This input is provided by the MicroVAX boot ROMs through register R2, and contains the Q-BUS address of the control status register for the boot device.

4) Boot Device Unit Number - This input is provided by the MicroVAX boot ROMs through register R3, and contains the unit number of the device that the boot is occurring from.

5) ROM Based Disk Driver - This input is provided by the MicroVAX boot ROMs through register R6, and contains the address of the ROM based driver for the boot device.

3.6.1.2 Local Data

No local data is defined for the Primary Boot TLC.

3.6.1.3 Interrupts

No interrupts are processed by the Primary Boot TLC.

3.6.1.4 Timing and Sequencing

The only sequencing constraint placed on the Primary Boot TLC is that it is to be the first software to be executed upon startup of the IGW.

3.6.1.5 Processing

The Primary Boot Program contained in this TLC can be broken down into three main tasks.

The first task of the Primary Boot Program is to find the highest page of memory and to relocate itself there. After this relocation has been performed execution of the relocated code begins at the instruction following the relocation instructions.

The second task of the Primary Boot Program is to load the Secondary Boot Program from the boot device into main memory. This is accomplished by making use of the ROM based disk driver that is provided by the MicroVAX for the boot device. If an error is detected while reading from the boot device, the reading of the Secondary Boot Program is restarted from the beginning.

The third task of the Primary Boot Program is to begin execution of the Secondary Boot Program. Before control is passed to the Secondary Boot Program the following information is loaded into registers to be read by the Secondary Boot Program:

- Boot Device CSR (Register R9)
- Boot Device Type (Register R10)
- Boot Device Unit (Register R10)

If for any reason the secondary boot program returns without successfully loading and executing the IGW software, the Primary Boot Program will attempt to reload the Secondary Boot Program and execute it again.

3.6.1.6 Outputs

The following outputs are produced by the Primary Boot TLC:

1) Secondary Boot Program - This output is written to the first 7.5K bytes of the MicroVAX main memory, after the Primary Boot Program has been relocated.

2) Boot Device CSR - This output is provided to the Secondary Boot Program through register R9, and contains the Q-BUS address of the control status register for the boot device.

3) Boot Device Type - This output is provided to the Secondary Boot Program through the first 8 bits of register R10, and contains a value indicating the type of device that is being booted from.

4) Boot Device Unit - This Output is provided to the Secondary Boot Program through the second 8 bits of register R10, and contains the unit number of the device that is being booted from.

3.6.2 Secondary Boot TLC

The Secondary Boot TLC is responsible for reading all of the IGW software and configuration files into main memory, and for loading the IXIB software to the IXIB interface. In addition, the initialization of memory hardware and tables is performed. After this initialization and loading has been performed, control is transferred to the IGW ERTE that has been loaded into main memory.

The Secondary Boot consists of two components: the Local Boot component which reads software and data from the IGW boot floppies, and the Net Boot component which reads software and configuration data from a host on the Ethernet. The Net Boot is further divided into the following elements:

1) IGW Net Boot - This component receives an executable image from the network and installs it.

2) Host Net Boot - This component resides on the host, builds the executable image, and sends the image to the IGW.

3.6.2.1 Secondary Boot Local Boot

The Local Boot component resides on the IGW boot floppies. It is loaded by and receives control from the Primary Boot. The Local Boot reads all software and data from the boot floppies and loads this software and data into the IGW memory.

3.6.2.1.1 Inputs

The following inputs are required by the Secondary Boot Local Boot:

1) IGW System Files - This input is read from the floppy disk drive, and includes the following:

   - IGW ERTE
   - List of files containing IGW Processes
   - IGW Processes which make up the IGW TLCs
   - IGW Configuration Files
   - IXIB Software

2) Boot Device CSR - This input is provided by the Primary Boot Program through register R9, and contains the Q-BUS address of the control status register for the boot device.

3) Boot Device Type - This input is provided by the Primary Boot Program through the first 8 bits of register R10, and contains a value indicating the type of device that is being booted from.

4) Boot Device Unit - This input is provided by the Primary Boot Program through the second 8 bits of register R10, and contains the unit number of the device that is being booted from.

3.6.2.1.2 Local Data

No local data is defined for the Local Boot component of the Secondary Boot TLC.

3.6.2.1.3 Interrupts

No interrupts are processed by the Local Boot component.

3.6.2.1.4 Timing and Sequencing

The software contained in the Local Boot component is called by the Primary Boot TLC whenever the IGW is booted and the Local Boot component resident on the boot device.

3.6.2.1.5 Processing

To allow the Local Boot to load the IGW software and data, it must first relocate itself from the beginning of memory to the end of memory, and then set up the memory management hardware and tables.

Initial memory initialization involves sizing memory, creating the system vector page, reserving a contiguous area of memory for the system page table, and reserving an area of memory known as the link area to store various tables and addresses of tables. At this point it is required to add entries in the system page table referencing the link area.

After the memory initialization is performed, the Secondary Boot Local Boot component loads the IGW ERTE from a file. The disk accesses required to load ERTE, as with all disk accesses in the

Secondary Boot TLC, are accomplished by making use of the MicroVAX ROM based disk driver. The Local Boot component will find the location of the file on the disk, and read the first block of the file to acquire the header for this file. Once this header is obtained, the text and data portions of the file are loaded into the IGW memory, and the uninitialized data segment is cleared. During the loading of ERTE, system page table entries for ERTE are created as required.

Once the IGW ERTE is loaded in IGW memory, the next step is to to load the IGW TLC processes from disk to memory. The list of processes that are to be loaded into main memory is loaded from a disk file into the link area of memory.

For each process that is required, the Local Boot component determines the location of the file containing the process executable file, and loads this file into the IGW memory. This loading is done sequentially through physical memory, with process stacks being created along with the processes. As each process is being loaded, process page table entries are created for the loaded pages. As the process page table grows, system page table entries are added to reference the pages occupied by the process page table.

After each process is loaded, the hardware process control block contained in the process list in the link area is loaded with the virtual address of the process page table and the length of that table. The physical address of this hardware process control block used during context switching is also stored in the process list at this time.

At this point the IGW software will have been loaded. Next it is

necessary to load the IGW configuration information. This information is read from disk files and loaded into physical memory after the processes and after the process page tables in system virtual address space. During this table loading, system page table entries are added as required. After each table is loaded, its system virtual address is stored in the link area of memory to allow other processes to access it.

The system interrupt stack is now allocated. This stack is allocated immediately following the IGW configuration information in physical memory.

The remainder of the free physical memory is now mapped into the system virtual address space by adding appropriate system page table entries. This memory is free to be allocated by the IGW operating software.

The next task performed by the Local Boot component is to link the I/O physical address pages into the top of the system virtual address space.

The final step before transferring control to the IGW ERTE is to load the IXIB board. This is done by reading three separate files containing:

- IXIB software in Motorola S-Record format
- IXIB software configuration information
- IXIB address configuration information

After the information from these files has been written to the IXIB board, the Secondary Boot Local Boot component transfers control to the IGW ERTE.

3.6.2.1.6 Outputs

The following outputs are produced by the Local Boot:

1) IGW Software - This output is written to the main memory of the IGW, and contains the software for the IGW ERTE as well as for all of the TLC processes that are to be controlled by the IGW ERTE.

2) IGW Configuration Information - This output is written to the main memory of the IGW, and contains configuration information required by the IGW software.

3) IXIB Software - This output is written to the IXIB interface, and contains the software required by the IXIB interface.

3.6.2.2 Secondary Boot IGW Net Boot

The IGW Net Boot component resides on the IGW boot floppies and is loaded and executed by the Primary Boot. The IGW Net Boot reads all software and data from the boot floppies and loads this software and data into the IGW memory.

3.6.2.2.1 Inputs

The following inputs are required by the Secondary Boot IGW Net Boot:

1) IGW Executable Image - This input is an executable image of the IGW software and configuration data which can be directly loaded into the IGW memory and executed.

2) IXIB Executable Image - This input is the image of the IXIB software which is downloaded to the IXIB hardware.

3.6.2.2.2 Local Data

The following local data is defined for the IGW Net Boot component of the Secondary Boot:

1) Ethernet Device Register Addresses - The device registed addresses for the Ethernet interface hardware will be known by the IGW Local Boot component.

2) Host Network Addresses - The Internet and Ethernet addresses of the host which will supply the required software and configuration data will be known by the IGW Net Boot component.

3.6.2.2.3 Interrupts

No interrupts are processed by the IGW Net Boot component.

3.6.2.2.4 Timing and Sequencing

The software contained in the IGW Net Boot component is called by the Primary Boot TLC whenever the IGW is booted with the IGW Net Boot component resident on the boot device.

3.6.2.2.5 Processing

To allow the IGW Net Boot to load the IGW software and data it must first size memory and relocate itself from the beginning of memory to the end of memory.

After the memory initialization is performed, the Secondary Boot IGW Net Boot sends a message to the Host Net Load component on the host which is supplying the executable images and data. This message indicates the size of memory on the IGW and requests an image of the IGW software to be sent.

The IGW Net Boot component then waits for the image to be sent, and stores the image in the IGW memory. After the image is sent, the IGW Net Boot component then sends a message to the host requesting the IXIB software image be sent. As this information is received, the IGW Net Boot component downloads the information to the IXIB board.

Once the software and configuration data is loaded, the IGW Net Boot receives a final message indicating that loading is completed. This message also supplies the values of the internal registers that the Host Net Boot program is able to determine. The IGW Net Boot sets the registers to the supplied values and calculates the values of other required registers. The IGW Net Boot then transfers control to the operating software.

3.6.2.2.6 Outputs

The following outputs are produced by the IGW Net Boot:

1) IGW Software - This output is written to the main memory of the IGW, and contains the software for the IGW ERTE as well as for all of the TLC processes that are to be controlled by the IGW ERTE.

2) IGW Configuration Information - This output is written to the main memory of the IGW, and contains configuration information required by the IGW software.

3) IXIB Software - This output is written to the IXIB interface, and contains the software required by the IXIB interface.

3.6.2.3 Secondary Boot Host Net Boot

The Host Net Boot component resides on the host which is cooperating with the IGW net boot process. The Host Net Boot runs continuously as a network server process which listens for boot requests from the IGW Net Boot component, and then sends the IGW and IXIB software and configuration data to the IGW. The Host Net Boot reads all software and data from a known directory on disk which contains a copy of all the files that would be present on the IGW boot floppies read by the Local Boot component.

3.6.2.3.1 Inputs

The following inputs are required by the Secondary Boot Host Net Boot:

1) IGW System Files - This input is read from a host disk directory and includes the following:

   - IGW ERTE
   - List of files containing IGW Processes
   - IGW Processes which make up the IGW TLCs
   - IGW Configuration Files
   - IXIB Software
   - IXIB Configuration Files

2) Request Messages - This input is received from the IGW and consists of messages requesting load images be downloaded to the IGW.

3.6.2.3.2 Local Data

No local data is defined for the Host Net Boot component of the Secondary Boot TLC.

3.6.2.3.3 Interrupts

No interrupts are processed by the Host Net Boot component.

3.6.2.3.4 Timing and Sequencing

The software contained in the Host Net Boot component runs continuously as a server process on the host system.

3.6.2.3.5 Processing

The Host Net Boot waits for a message from the IGW requesting that the IGW be loaded. This message contains the size of memory on the IGW.

To allow the Host Net Boot to load the IGW software and data it must first create an area of local memory the size of the IGW memory to be used as an area to build the IGW software and data image. In the paragraphs below, references to loading software or data refer to this local memory.

To prepare an image of the IGW system, the Host Net Load component loads the system vector page, reserves a contiguous area of memory for the system page table, and reserves an area of memory known as the link area to store various tables and addresses of tables. At this point it is required to add entries in the system page table referencing the link area.

After the memory initialization is performed, the Secondary Boot Host Net Boot component loads the IGW ERTE from a file. This disk access, as with all disk accesses in the Host Net Boot is accomplished by making use of the standard UNIX file access routines. The Host Net Boot component will open the file on the disk, and read the first block of the file to acquire the header for this file. Once this header is obtained, the text and data portions of the file are loaded into the IGW memory, and the

uninitialized data segment is cleared. During the loading of ERTE, system page table entries for ERTE are created as required.

Once the IGW ERTE is loaded in the local IGW memory image, the next step is to to load the IGW TLC processes from disk to memory. The list of processes that are to be loaded into memory is loaded from a disk file into the link area of memory.

For each process that is required, the Host Net Boot component determines the location of the file containing the process executable file, and loads this file into the IGW memory image. This loading is done sequentially through memory, with process stacks being created along with the processes. As each process is being loaded, process page table entries are created for the loaded pages. As the process page table grows, system page table entries are added to reference the pages occupied by the process page table.

After each process is loaded, the hardware process control block contained in the process list in the link area is loaded with the virtual address of the process page table and the length of that table. The physical address of this hardware process control block used during context switching is also stored in the process list at this time.

At this point the IGW software will have been completely loaded. Next it is necessary to load the IGW configuration information. This information is read from disk files and loaded into physical memory after the processes and after the process page tables in system virtual address space. During this table loading, system page table entries are added as required. After each table is

loaded, its system virtual address is stored in the link area of memory to allow other processes to access it.

The system interrupt stack is now allocated. This stack is allocated immediately following the IGW configuration information in physical memory.

The remainder of the free physical memory is now mapped into the system virtual address space by adding appropriate system page table entries. This memory is free to be allocated by the IGW operating software.

The next task performed by the Host Net Boot component is to link the I/O physical address pages into the top of the system virtual address space.

The final step is to send the completed image of the IGW system to the IGW. The image is sent via Ethernet one page (512 bytes) at a time.

The Host Net Boot component then waits for the IGW Net Boot component to request the IXIB software and configuration data. The Host Net Boot reads the file containing the

IXIB software in Motorola S-Record format.

After the information from this file has been sent to the IGW, the Secondary Boot Host Net Boot component sends an end of boot message to the IGW. This message contains the values of all IGW internal registers that the Host Net Boot component can determine. The message signals that downloading has been completed.

3.6.2.3.6 Outputs

The following outputs are produced by the Host Net Boot:

1) IGW Executable Image - This output is sent to the IGW over the Ethernet.

2) IXIB Executable Image - This output is sent to the IGW over the Ethernet.

3.6.3 IP Protocol TLC

The IP TLC is responsible for the implementation of the IP protocol, the ICMP protocol, and packet filtering. The TLC accepts datagrams received by one of the network hardware interfaces and processes the datagram according to IP and ICMP protocol specifications [3,4], and according to packet filtering specifications outlined in section 3.6.3.5.

3.6.3.1 Inputs

The inputs to IP are as follows:

1) Datagrams: these are received in message buffers transmitted to the IP TLC from one of the interface driver TLCs or from the EGP TLC.

2) Global Table Data: the IP TLC reads table data to determine where to route a datagram, to determine legality of the route, to select a network interface for datagram transmission, and to build the IP route table.

3) Tables Updated Message: a message buffer from EGP indicating that EGP global tables have changed and that IP should recalculate its route table.

3.6.3.2 Local Data

The IP TLC maintains a hash table for entries in the IP Route Table. Every time IP builds the IP Route Table, it creates a hash table based on the network address class of each route table entry. Three hash lists are maintained, one for each of the classes A, B, and C. The hash lists are indexed by the low order eight bits of the network address with collisions handled by linking entries together into a linked list. IP uses the hash table to look up routes quickly.

IP also accumulates traffic data locally. IP examines each non-erroneous packet and stores the source and destination address and packet size in a table. When the table is full, or every five minutes, whichever occurs first, IP sends the table to the STAT TLC for data reduction and begins filling the table again. IP also counts erroneous datagrams. A count of the number of datagrams that have invalid headers is maintained. This count is sent to the STAT TLC whenever the traffic data table is sent to the STAT TLC.

Additionally, IP collects data about ICMP packets. IP fills a table that specifies, for each ICMP datagram sent or received, the IP address, the ICMP type, the ICMP code, and a sent/received indicator. This table is also sent to the STAT TLC when full or at five minute intervals.

3.6.3.3 Interrupts

IP does not process any interrupts.

3.6.3.4 Timing and Sequencing

The IP TLC will run as a process under the IGW. The ERTE TLC will schedule the IP TLC for running based on the ERTE scheduling algorithm (described in Section 3.6.10). The IP TLC will be assigned a relatively high priority; only the network interface device drivers (XDD and EDD TLCs) will have higher priority. This will ensure that datagrams to be forwarded are done so promptly.

3.6.3.5 Processing

IP operates in a continual loop, reading message buffers from its message queue and processing the message buffer. The message buffer will contain an Internet IP datagram, or a table update message from EGP. If the message buffer contains a datagram, the IP TLC will process the datagram in accordance with IP and ICMP protocol specifications [3,4], and in accordance with packet filtering specifications. If the message buffer contains a table update message from EGP, then this message buffer instructs IP to recalculate the IP Route Table because EGP route data used in the IP table has changed. IP will then rebuild the IP Route Table and rebuild its hash tables.

The packet filtering specifications are defined in two parts. First, IP implements fixed packet filtering to prevent the flow of IP packets from one host through the IGW to another host on the same network. Secondly, IP implements variable packet

filtering using a Packet Filtering Table to control the flow of IP packets from a source host or network through the IGW to a destination host or network. At system startup the Packet Filter Table is set to define allowed or restricted IP packet flow between IP address pairs.

When processing a datagram, IP will forward a datagram (on a message buffer) according to the IP protocol. If the datagram requires the generation of an ICMP datagram, then this datagram is created and sent to the appropriate destination. If the datagram received is an ICMP datagram, then it is processed according to the ICMP protocol. EGP datagrams received from a network interface are forwarded to the EGP TLC.

If a datagram must be fragmented, then each fragment is placed in a separate message buffer and each message buffer is forwarded to the destination interface driver. If insufficient message buffers are available for fragmentation, fragmentation will not occur and the datagram will be discarded.

In addition to datagram processing and route table construction, the IP TLC accumulates traffic statistics on datagrams it processes. The TLC stores data showing source and destination and size of IP datagrams received and forwarded. The TLC also stores the address, type, and code of ICMP datagrams sent or received. Lastly, IP counts the number datagrams with errors that invalidate the datagram. Section 3.6.3.2 above provides additional details.

3.6.3.6 Outputs

IP produces the following outputs:

1) IP Route Table in the global data area;

2) Datagrams in message buffers to one of the network interface drivers for transmission on a network interface;

3) EGP datagrams in message buffers to the EGP TLC; and,

4) Traffic data, also in message buffers, to the STAT TLC.

3.6.4 EGP Protocol TLC

The EGP TLC implements the EGP protocol on IGW. The IP TLC receives EGP datagrams from other EGP gateways via the network interfaces and forwards them to EGP. EGP then processes these datagrams according to the EGP protocol specification [5]. Additionally, the EGP TLC generates datagrams which it passes to IP for transmission to other EGP gateways. The EGP TLC also collects traffic data on numbers and types of EGP packets sent and received.

3.6.4.1 Inputs

The EGP TLC has two types of inputs:

1) EGP Datagrams: these inputs arrive in message buffers from the IP TLC; and,

2) Global Tables: the EGP reads data from the network, gateway, and neighbour tables in the global tables data area.

3.6.4.2 Local Data

The EGP TLC gathers EGP traffic data in a local table. The table contains the Internet address, Autonomous System Number, EGP Type, EGP Code, and a sent or received indicator. An entry is added to the table for each EGP datagram sent or received by the EGP TLC. When the table is filled, EGP sends it to the STAT TLC, and then begins filling a new table.

3.6.4.3 Interrupts

No interrupts are processed by the EGP TLC.

3.6.4.4 Timing and Sequencing

The EGP TLC will be run whenever there are received EGP datagrams to be processed. The EGP TLC will also be run on a fixed schedule according to the EGP protocol to send reachability datagrams to other gateways.

The EGP TLC is allocated run time by the ERTE TLC based on a priority and round-robin scheduling system (Section 3.6.10). The EGP will be given priority less than the device driver TLCs and the IP TLC, but greater than the OI and STAT TLCs. Thus, EGP will run ahead of the OI and STAT TLCs but after all drivers and the IP TLC.

3.6.4.5 Processing

The processing performed by the EGP TLC conforms to the procedures required by the EGP protocol. The TLC executes an endless loop of waiting for incoming EGP datagrams, then processing the datagram and generating responses as required. The TLC also generates EGP datagrams to its neighbours based on timers, as specified by the EGP protocol. The timers are set by the EGP TLC.

The EGP TLC also collects EGP traffic data, an operation not part of the EGP protocol. For each EGP datagram sent or received, the EGP TLC stores data giving the sender or recipient address, the type of EGP datagram, and the Autonomous System of the EGP datagram. The data is transmitted to the STAT TLC every five minutes where data reduction is performed.

3.6.4.6 Outputs

the EGP generates the following outputs:

1) EGP datagrams: these are passed to the IP TLC in message buffers for transmission onto a network;

2) EGP Route Table: this table is built in the global area according to the EGP protocol (the global data area is used to permit other processes to examine the table); and,

3) EGP Traffic Data: this data is transmitted to the STAT TLC in a message buffer.

3.6.5 Operator Interface TLC

The Operator Interface TLC serves as an interface to the IGW operator. The Operator Interface will allow the operator to display various tables maintained by the IGW. The OI also allows the operator to modify some of the tables. In addition, the Operator Interface also provides the operator with the ability to retrieve traffic statistics generated by the STAT TLC.

3.6.5.1 Inputs

The following inputs are required by the Operator Interface TLC:

1) Operator Commands - This input is provided by the Console Device Driver, and contains commands by the IGW operator.

2) IGW Tables - This input is provided from the global data storage area, and contains the following tables:

   - Network Interface Table
   - Gateway Table
   - EGP neighbour Table
   - EGP Network Reachability Table
   - IP Routing Table

- IXIB Address Configuration Table
- Packet Filter Table

3.6.5.2 Local Data

No data is defined to be local in the Operator Interface TLC.

3.6.5.3 Interrupts

No interrupts are defined for the Operator Interface TLC.

3.6.5.4 Timing and Sequencing

The Operator Interface TLC is run whenever no other processes can be run and a command is available to be processed or a time scheduled event has occurred which requires OI action. The input command is made available to this TLC by the Console Device Driver TLC.

Time scheduled events are commands which are run at specified intervals. The length of interval is specified by the operator using an operator command.

3.6.5.5 Processing

Commands are presented to the Operator Interface TLC by the Console Device Driver TLC. The following commands are supported by the Operator Interface TLC:

1) Statistics commands:

   a) Display Statistics commands:
      - Display ICMP statistics
      - Display X.25 statistics
      - Display EGP statistics
      - Display all the above statistics

   b) Reset Statistics Commands:
      - Reset ICMP statistics
      - Reset X.25 statistics
      - Reset Ethernet statistics
      - Reset EGP statistics
      - Reset all the above statistics

   c) Set Interval Command
      - Set regular statistics display interval

2) Software Table and Parameter Commands:
   a) Display Commands:
      - Display network interfaces
      - Display gateways
      - Display neighbours
      - Display EGP routes
      - Display IP routes
      - Display X.121 address mapping table
      - Display packet filter table entries

   b) Addition Commands
      - Add gateway table entries
      - Add neighbour table entries
      - Add X.121 address configuration table entries
      - Add packet filter table entries

   c) Deletion Commands
      - Delete gateway table entries
      - Delete neighbour table entries
      - Delete X.121 address configuration table entries
      - Delete packet filter table entries

d) Modification Commands:
   - Modify network interface table entries
   - Modify gateway table entries
   - Modify neighbour table entries
   - Modify X.121 address configuration table entries
   - Modify packet filter table entries

The purpose and usage of these commands is described in detail in the IGW User's Guide.

The Display Statistics Commands are used to retrieve traffic statistics information from the STAT TLC. The operator may specify an individual traffic statistics item or all of the generated traffic statistics. The issuing of a Display Statistics Command to the Operator Interface TLC will cause a request to be sent to the STAT TLC which will process the request and send the requested information to the Console Device Driver TLC to be displayed on the operator's console.

The Set Interval Command permits the operator to specify an optional timer value which specifies a time interval for the statistics to be displayed automatically on a regular basis. Specifying a value of zero for the timer sets the timer to a default value of 12 hours. Twelve hours is also the maximum interval that can be specified.

The Reset Statistics Commands are used to reset statistics information in the STAT TLC. As with the Display Statistics Commands, an individual statistics item may be reset or all of the statistics items may be reset. Once the Operator Interface TLC determines which statistics are to be reset, it sends the appropriate requests to the STAT TLC to perform the resetting of the statistics.

The Display Software Table Commands are used to display various tables maintained by the IGW. When a request to display this information is received from the Console Device Driver TLC, the requested information is obtained from a global data storage area, formated, and sent to the Console Device Driver TLC for display on the operator's console.

It is possible to delete entries from certain tables maintained by the IGW. The applicable tables for entry deletion are:

- Gateway Table
- neighbour Table
- X.121 Address Configuration Table
- Packet Filter Table

When a request is received to delete an entry from one of these tables, the entry is not actually removed from the table, but is marked as being deleted so that it may be reused to hold new entries at a later time if desired.

Entries may be added to certain of the tables maintained by the IGW. The entries that this addition applies to are as follows:

- Gateway Table
- Neighbour Table
- X.121 Address Configuration Table
- Packet Filter Table

To add an entry the operator will be asked for the new entry, and the new entry will be added to the specified table in the global data storage area, unless such an addition exceeds the table size, in which case the addition is refused and a message indicating that the table is full is displayed on the operator console.

With certain IGW tables it is possible to modify existing entries within those tables. The tables that the table modification facility apply to are:

- Network Interface Table
- Gateway Table
- Neighbour Table
- X.121 Address Configuration Table
- Packet Filter Table

Modification of a table entry consists of replacing an existing table entry with a new one. To modify an existing table entry, the operator will be asked to specify which entry of the table is to be modified, along with the contents of the entry that it is to be replaced by. Only certain information contained in a table entry may be modified depending on the table that is being modified.

3.6.5.6 Outputs

The following outputs are produced by the Operator Interface TLC:

1) STAT Commands - This output is sent in memory buffers to the STAT TLC, and contains requests for traffic statistics to be sent to the operator's console.

2) Tables - This output involves the modification of various IGW tables and parameters that are kept in the IGW global data storage area.

3) Console Output - This output is sent in memory buffers to the Console Device Driver TLC, and contains output that is to be displayed on the operator's console.

3.6.6 Statistics (STAT) TLC

The STAT TLC is responsible for gathering traffic data from other IGW TLCs and reducing the data into tables suitable for display on the operator's console. The operations of this TLC include accumulating totals of packet counts and packet sizes, sorting statistical data, and arranging the data into tables.

3.6.6.1 Inputs

The STAT TLC receives the following inputs:

1) X.25 Traffic Data: this data is received in message buffers from the X.25 Device Driver, and contains the number of X.25 packets sent and received to and from each X.25 host or gateway.

2) IP Traffic Data: this data consists of a list which identifies the source and destination internet addresses and the size in bytes of each datagram passing through the IP TLC.

3) ICMP Traffic Data: this data consists of a list which indicates the source or destination address, type, and code of each ICMP datagram processed (sent or received) by the IP TLC.

4) EGP Traffic Data: this data is a list of packets generated and received by the EGP TLC. The list entries indicate, for each EGP datagram sent or received, the source or destination address, the type, and the direction (sent or received) of the datagram.

5) OI Requests: this data is a request for statistics from the OI TLC. The request will indicate that the STAT TLC should:

    1. send X.25, IP, ICMP, EGP, or all (all the preceding) statistics to the OI; or,

    2. clear the statistics and begin recalculating new statistics.

3.6.6.2 Local Data

The STAT TLC maintains a set of tables containing the current statistics for each type of traffic (X.25, IP, ICMP, and EGP). The tables are as follows:

1) Name: X.25 Statistics Table

   Purpose: To maintain activity information to and from X.25 hosts and gateways.

   Content: One entry for each X.25 host or gateway, sorted by X.121 address. Each entry contains:

   - X.121 address
   - number X.25 packets transmitted to the host
   - number of bytes transmitted to the host
   - number of X.25 packets received from the host
   - number of bytes received from the host
   - the number of X.25 calls to the host
   - the number of X.25 calls from the host

2) Name: IP Statistics Table

   Purpose: To monitor activity between pairs of Internet Hosts which pass data through the IGW.

   Content: The table will have one entry for each pair of Internet hosts. The table will be sorted by the lesser of the two addresses of each pair. Each entry will contain:

   - the lesser of the two Internet addresses (A1)
   - the greater of the two addresses (A2)
   - the number of datagrams from A1 to A2
   - the number of bytes from A1 to A2
   - the number of datagrams from A2 to A1
   - the number of bytes from A2 to A1

3) Name: ICMP Statistics Table

Purpose: To track ICMP activity involving the IGW

Content: The table will contain one entry for each Internet address to or from which an ICMP datagram was sent or received. The table is sorted by Internet address. Each entry contains:

- the Internet address of the participating host or gateway
- the number of ICMP packets received for each of the eleven ICMP message types defined in RFC-792 [4].
- the number of ICMP datagrams transmitted for each of the eleven ICMP message types

4) Name: EGP Statistics Table

Purpose: To monitor EGP traffic to and from the IGW

Content: There is one entry for each EGP gateway that interacts with the IGW, sorted by gateway address. Each entry consists of:

- Internet address of the interacting EGP gateway
- the number of EGP datagrams sent to the IGW from the gateway for each of the five EGP program types defined in RFC-904 [5].
- the number of EGP datagrams sent by the IGW to the EGP gateway for each of the five EGP datagram types

Each of the above tables will have a maximum number of entries. If the table is filled, then when a new entry is to be added, the least active entry in the table will be removed and replaced by a new entry. If this occurs, then when STAT is requested to display the statistics from the affected table, the STAT process will append a message indicating that the statistics are

incomplete to the table display.

3.6.6.3 Interrupts

The STAT TLC services no interrupts.

3.6.6.4 Timing and Sequencing

Whenever the STAT TLC has traffic data or OI requests to process, the TLC will be marked as runnable and allocated CPU time by the ERTE. The STAT TLC is allocated CPU time on a priority basis. The STAT TLC is given a priority lower than all other processes except the OI. The ERTE will give the STAT TLC CPU time only when all higher priority processes are unable to run because they are waiting on events.

3.6.6.5 Processing

The STAT TLC operates in an endless loop of waiting for traffic data or OI requests, and processing the data or requests. When traffic data is received, the STAT TLC processes the incoming traffic data tables sequentially, adding new data or modifying data in STAT's local tables, depending on the received data.

When STAT receives an OI request to display statistics, STAT converts the requested table (or possibly all tables) into an ASCII format suitable for display on the operator's console, and puts the table into as many message buffers as required and sends them to the console driver for display on the console.

If the OI requests that statistics be cleared, then the STAT TLC will discard all statistical data currently in its tables and

will begin to accumulate new traffic statistics.

The STAT TLC will use a short queue to accept incoming traffic data messages. If this queue fills and the STAT routine can not get enough time to process drain the queue (which may occur if the gateway is very busy processing datagrams) then other processes will be prevented from adding to the STAT input queue. The sending process will then drop the traffic data message and instead send a message to the operator console indicating that traffic data was lost. This ensures that the IGW does not waste time processing statistics when the IGW is busy with its gateway functions.

3.6.6.6 Outputs

The STAT TLC generates the following outputs:

1) Statistics: STAT produces statistics in ASCII formatted tables which are sent to the Console Device Driver in message buffers.

2) Error Messages: STAT will also generate error messages which it will send to the Console Device Driver in message buffers.

3.6.7 X.25 Device Driver (XDD) TLC

The X.25 Device Driver TLC consists of a device driver responsible for the control of the interface between the IGW Q-BUS and the IXIB. This TLC provides the IGW with the ability to communicate with the IXIB. The communication involves the transmission and reception of packets to be sent over an X.25 network, as well as control and status information.

There is one instance of this driver for each IXIB interface in

the IGW. Each operates identically, but independently of the other.

3.6.7.1 Inputs

The following inputs are received by the X.25 Device Driver TLC:

1) X.25 Input Packet - This input is read from the IXIB, and contains IP and logging packets.

2) IGW Input Packet - This Input packet is obtained from other TLCs residing on the IGW via IGW messages, and contains packets to be transmitted by the IXIB, as well as various control packets.

3) IXIB Address Configuration Table - This input is obtained from a global storage area, and contains the X.25/IP address configuration table required by the IXIB.

3.6.7.2 Local Data

The following data is local to the X.25 Device Driver TLC:

1) Name: IXIB Software Control

   Purpose: Used to maintain software information pertaining to the IXIB.

   Contents: Structure containing device software information for an individual IXIB.

2) Name: IXIB Traffic Statistics

   Purpose: Used to maintain statistics to allow the analysis of X.25 traffic flow on the IGW.

   Contents: A table containing the following information for each X.25 host known by the gateway:

   - Number of input packets

- Number of input bytes
- Number of output packets
- Number of output bytes
- Number of incoming calls
- Number of outgoing calls

3.6.7.3 Interrupts

Two interrupts are serviced by this TLC. One is used to process input commands from the IXIB and the other to process output commands from the IXIB. The IXIB produces a priority 4 (ipl 14) interrupt when the IXIB writes a command to the Mailbox command register of the IXIB. The IXIB produces a priority 4 (ipl 14) interrupt when the IXIB reads a command from the Mailbox command register of the IXIB.

3.6.7.4 Timing and Sequencing

The software residing in the X.25 Device Driver TLC is designated as runnable whenever a packet is available to be read from or written to the IXIB. Once designated as runnable, the ERTE TLC will run the X.25 Device Driver on a priority basis. The X.25 Device Driver TLC is lower in priority than the Ethernet Device Driver TLC, but higher in priority than all other TLCs. Within the X.25 Device Driver TLC the processes are given priority as follows (in decreasing order):

- XDD Packet Input
- XDD Packet Output
- XDD Supervisor Functions

3.6.7.5 Processing

The X.25 Device Driver TLC is composed of three separate processes: an input process, an output process, and a supervisor process. The XDD supervisor process is responsible for communications to other software residing on the IGW, while the XDD input and output processes communicate with the IXIB.

The communication between the XDD supervisor process, and the XDD input and output processes is accomplished by the use of message buffers. To send a packet from the XDD supervisor process to the XDD output process, the XDD supervisor process places the output packet in a message buffer which it sends to the XDD output process. The XDD input process sends received packets to the supervisor in a message buffer, and the XDD supervisor then determines the appropriate action for the received packets.

Other IGW software desiring to communicate with the XDD TLC also makes use of message buffers. To send information to the XDD TLC, message buffers are sent to the XDD supervisor. Any message that is successfully sent to the supervisor is guaranteed to be processed, and sent to the IXIB if required. Information sent to the X.25 Device Driver TLC consists of either full IP packets or full command packets, however these packets may be broken down into smaller pieces by the driver before actual transmission to the IXIB.

The XDD passes information to other IGW processes by sending message buffers to these processes. Such messages contain either full IP packets or full logging packets.

The transmission of a packet from the X.25 Device Driver TLC to

the IXIB is accomplished by the XDD output process making use of the IXIB FIFO and Mailbox. Data transmission is initiated by dividing the packet to be transmitted to the IXIB into a command portion and a data portion. The data portion is loaded into the IXIB FIFO, and then the IXIB Mailbox is loaded with the command portion. Once the IXIB reads the command written by the XDD output process, the X.25 output process is free to send another packet to the IXIB. For further information concerning the IXIB Mailbox and FIFO as well as other IXIB hardware related details, consult Reference [6].

The reception of a packet from the IXIB by the X.25 Device Driver TLC is accomplished by the XDD input process making use of the IXIB FIFO and Mailbox. Data reception is initiated by the receipt of a receive interrupt. Upon receipt of this interrupt the XDD input process reads a command from the IXIB Mailbox. After decoding the command, any data required by this command is extracted from the IXIB FIFO.

X.25 traffic data collection is performed on regular five minute intervals. This procedure involves requesting the IXIB to provide the IGW with a copy of its traffic per X.121 address information. When this information has been obtained, it is sent in a message buffer to the STAT TLC.

3.6.7.6 Outputs

The following outputs are produced by the X.25 Device Driver TLC:

1) X.25 Output Packet - This output is written to the IXIB, and contains IP packets to be transmitted by the IXIB as well as various control packets.

2) IGW Datagram - This output is sent in message buffers to the IP TLC, and contains IP datagrams.

3) Logging Messages - This output, obtained from information produced by the IXIB, contains IXIB logging and error information and is sent to the Console Device Driver for display on the operator's console.

4) X.25 Traffic Data - This output is obtained from a table generated by this TLC, and contains totals of X.25 packets and bytes that have been transmitted and received by the IXIB. This output is sent to the STAT TLC in message buffers.

3.6.8 Ethernet Device Driver (EDD) TLC

The Ethernet Device Driver TLC consists of a device driver responsible for the management of the Digital Equipment DEQNA Ethernet interface. This TLC provides the IGW processor with the ability to communicate with the DEQNA interface. This communication involves the transmission and reception of Ethernet packets, as well as control and status information. This device driver implements the ARP protocol [7].

There is one instance of this driver for each Ethernet interface in the IGW. Each operates identically, but independently of the

other.

3.6.8.1 Inputs

The following inputs are required by the Ethernet Device Driver TLC:

1) Ethernet Input Packet - This Input is read from the control registers and DMA area of the DEQNA, and contains IP and logging packets.

2) IGW Input Packet - This Input packet is obtained from messages sent from other software residing on the IGW, and contains packets to be transmitted by the DEQNA, as well as various control packets.

3.6.8.2 Local Data

The following data is local to the Ethernet Device Driver TLC:

1) Name: Ethernet Software Control

   Purpose: Used to maintain software information pertaining to the DEQNA.

   Contents: Structure containing device software information for an individual DEQNA board.

2) Name: ARP Address Mapping Table

   Purpose: Used to map between IP and Ethernet addresses.

   Contents: Table containing the following information for mapping between IP and Ethernet:

   - IP address
   - Ethernet address
   - Time entry last referenced

- Last packet for incomplete entry
- Flags

3) Name: Ethernet Traffic Data Table

Purpose: Used to maintain statistics to allow the analysis of Ethernet traffic flow on the IGW.

Contents: A table containing the following information for received packets:

- Remote Ethernet Address
- Direction of packet (input/output)
- Number of bytes in packet

3.6.8.3 Interrupts

The DEQNA will produce a priority 4 (ipl 14) interrupt on the host bus when the DEQNA transmits or receives an Ethernet packet.

3.6.8.4 Timing and Sequencing

The software residing in the Ethernet Device Driver TLC is designated as runnable whenever a packet is available to be read from or written to the DEQNA. Once designated as runnable, the ERTE TLC will run the Ethernet Device Driver on a priority basis. The Ethernet Device Driver TLC is higher in priority than all other TLCs. Within the Ethernet Device Driver TLC the processes are given priority as follows (in decreasing order):

- Ethernet Input
- Ethernet Output
- Ethernet Supervisor Functions

3.6.8.5 Processing

The Ethernet Device Driver TLC is composed of two separate processes: an interrupt handling process, and a supervisor process.

The EDD supervisor process is responsible for communications to other software residing on the IGW and Ethernet packet processing, while the EDD interrupt handling process communicates with the EDD supervisor process and the DEQNA board.

The communication between the EDD supervisor process, and the EDD interrupt handling process is accomplished by the use of the IGW message passing facility. To send a packet from the EDD supervisor process to the EDD interrupt handling process, the EDD supervisor process places the output packet in a message buffer queue which it sends to the EDD interrupt handling process. The EDD supervisor process also receives message buffers containing input packets sent from the EDD interrupt handling process.

Other IGW software desiring to communicate with the Ethernet Device Driver TLC also makes use of message buffers. Information for the Ethernet Device Driver TLC is placed in message buffers

and sent to the EDD supervisor. Any packet that is successfully sent is guaranteed to be processed, and sent to the DEQNA if required. Information sent to the Ethernet Device Driver TLC consists of either IP packets or command packets.

The EDD transmits information to the other TLCs by placing the information in a message buffer and then sending that message to the TLC.

To transmit a packet from the Ethernet Device Driver TLC to the DEQNA interface, the EDD supervisor process first extracts the packet to be processed from the incoming message buffer. If the packet is a command packet it is processed appropriately. If the packet is to be an IP packet to be sent over the Ethernet, the EDD supervisor process will determine the correct Ethernet address to send to by performing a table lookup in the ARP address mapping table.

If an address translation entry exists the IP packet is encapsulated into an Ethernet packet and sent in a message buffer to the EDD interrupt handling process. The EDD interrupt handling process is then responsible for the transmission of the packet to the DEQNA interface.

If no entry corresponding to the IP address is found in the ARP

address mapping table, then the correct Ethernet address is determined according to the ARP protocol. If no Ethernet address can be determined via ARP, the packet is dropped.

The reception of a packet from the DEQNA interface by the Ethernet Device Driver TLC is accomplished by the EDD interrupt handling process reading the packet from the DEQNA interface. This packet is then placed in a message buffer and sent to the EDD supervisor process. The EDD supervisor process will determine if the packet is an IP, ARP, or an other protocol type packet. If the packet is not of an IP or ARP protocol type the packet is discarded.

If the input packet is found to be an ARP packet, it is processed appropriately. The ARP table is updated, ARP replies sent, and stored packets sent as required.

If the input packet is found to be an IP packet, the IP portion of the Ethernet packet is extracted and placed in a message buffer and sent to the IP TLC.

When a packet is transmitted or received information from the packet is added to the Ethernet Traffic Data Table. When this table becomes full it is sent to the STAT TLC, and a new table is created. The entries stored in this table are described in

section 3.6.8.2 of this document.

3.6.8.6 Outputs

The following outputs are produced by the Ethernet Device Driver TLC:

1) Ethernet Output Packet - This output is written to the DEQNA control registers, and a DMA area. It contains packets to be transmitted by the DEQNA, as well as control information.

2) IGW Datagram - This output is placed in a message buffer and sent to the IP TLC, and contains IP datagrams.

3) Ethernet Traffic Data - This output is sent in message buffers to the STAT TLC, and contains a list of traffic statistics information for incoming and outgoing Ethernet Traffic.

3.6.9 Console Device Driver (CDD) TLC

The Console Device Driver TLC is a device driver responsible for the control of the interface between the IGW and the console device hardware. This TLC provides the IGW with the ability to communicate with the console interface hardware. The communication involves the transmission and reception of characters to and from the console, as well as control and status information.

3.6.9.1 Inputs

The following inputs are received by the Console Device Driver TLC:

1) Console Input Character - This input is read from the Console Device, and consists of characters typed at the operator's console.

2) IGW Input Message - This input message is sent to the CDD from other TLCs residing on the IGW, and contains character strings to be transmitted by the Console Device to the operator's console.

3.6.9.2 Local Data

The following data is local to the Console Device Driver TLC:

1) Name: Console Device Software Control

   Purpose: Used to maintain software information pertaining to the Console Device.

   Contents: Structure containing device software information for an individual Console Device.

3.6.9.3 Interrupts

Two interrupts are serviced by this TLC. One is used to process input characters from the Console Device and the other to process output characters to the Console Device. The Console Device produces a receive interrupt when the Console Device writes a character into the input data register of the Console Device. The Console Device produces a transmit interrupt when the Console Device completes the transmission of a character from the output data register of the Console Device.

3.6.9.4 Timing and Sequencing

The software residing in the Console Device Driver TLC is run whenever a character is available to be read from Console Device or there are characters or messages available to be transmitted by the Console Device. The ERTE TLC will run the Console Device Driver on a priority basis. The Console Device Driver TLC is lower in priority than the Ethernet Device Driver TLC, X.25 Device Driver TLC, and IP TLC, but higher in priority than all other TLCs. Within the Console Device Driver TLC the processes are given priority as follows (in decreasing order):

- CDD Character Input
- CDD Character Output
- CDD Supervisor Functions

3.6.9.5 Processing

The Console Device Driver TLC is composed of three separate processes: an input process, an output process, and a supervisor process. The CDD supervisor process is responsible for communications to other software residing on the IGW, while the CDD input and output processes communicate with the Console Device and the supervisor process.

The communication between the CDD supervisor process, and the CDD input and output processes is accomplished by the use of message buffers. To send a character stream from the CDD supervisor process to the CDD output process, the CDD supervisor process places the output stream in a message buffer which it sends to the CDD output process. The CDD output process then extracts each character from the stream and transmits it to the operators console. If the character to be transmitted is an ASCII Line Feed character, then the CDD output process will insert a single ASCII Carriage Return character in front of it.

The CDD input process collects received characters a message buffer. When the input process receives an ASCII Carriage Return character, the process will send the message buffer to the CDD supervisor who will then forward it to the OI TLC. The input process will begin collecting newly received characters in a new

message buffer.

Other IGW software desiring to communicate with the CDD TLC also makes use of message buffers. To send information to the CDD TLC, message buffers are sent to the CDD supervisor. Any message that is successfully sent to the supervisor is guaranteed to be processed, and sent to the Console Device. Information sent to the Console Device Driver TLC consists of message buffers containing character streams to be displayed on the operator's console. Such stream are transmitted verbatim, except for the ASCII Line Feed character. This character signals the end of a line, so the CDD inserts a Carriage Return character in front of the Line Feed character.

3.6.10 Efficient Real-Time Executive (ERTE) TLC

The ERTE TLC is responsible for controlling the execution of all processes that make up all the other TLCs of the IGW system. The ERTE TLC also offers a set of facilities to the processes it controls which permit the processes to communicate with one another, wait for events, declare events, suspend execution for a period of time, and give up the CPU so that other processes may use the CPU.

3.6.10.1 Inputs

The inputs to the ERTE are supplied to the ERTE by processes requesting ERTE functions. The calling process uses an ERTE system call to pass information to the ERTE. The system call places the system call parameters into process general registers, and then executes an instruction which generates a VAX exception. This results in a vectored transfer to the ERTE which determines the system call and parameters from the process registers, and performs the function indicated by those parameters. The following lists the system calls available and the ERTE input parameters associated with them:

1) Name: Suspend

   Purpose: A request to the ERTE to allow other processes awaiting access to the CPU to be allowed to run

   Parameters: - Suspend request code

2) Name: Sleep

   Purpose: A request to the ERTE to have the process inactive for a period of time

   Parameters: - Sleep request code

   - Time to sleep

3) Name: Wait On Event

   Purpose: To instruct the ERTE to suspend the process execution until the indicated event has occurred.

Parameters:
- Wait Event request code
- Event number to wait for

4) Name: Wait With Time-out

Purpose: To instruct ERTE to suspend process execution until either the indicated event occurs, or the time-out period expires

Parameters:
- Wait With Time-Out request code
- Event to wait for
- Time-out period

5) Name: Wait For One Of

Purpose: To instruct ERTE to suspend process execution until one of the events in a given list of events occurs, or until the given time-out period expires

Parameters:
- Wait For One Of request code
- Pointer to a list of event codes
- Time-out period

6) Name: Message Get

Purpose: To instruct the ERTE to allocate a free message buffer to the requesting process.

Parameters:
- Message Get request cod
- Pointer to process space where the ERTE will put the message header.

7) Name: Message Discard

Purpose: To tell the ERTE to return the indicated message to the free message list.

Parameters:
- Message Discard request code
- Pointer to the message header of the message to be discarded

8) Name: Open Message Queue

Purpose: To instruct the ERTE to allocate a queue for incoming messages to the process

Parameters:
- Open Message Queue request code
- Identifier of message queue to be opened
- Maximum number of messages in the queue

9) Name: Message Send

Purpose: To send a message to another process

Parameters:
- Message Send request code
- Pointer to message header of message to be sent
- Queue identifier of message queue the message should be placed on

10) Name: Message Receive

Purpose: To receive a message from a message queue

Parameters:
- Message Receive request code
- Identifier of queue to get message from

11) Name: Queue Status

Purpose: To get the status of the indicated message queue

Parameters:
- Queue Status request code
- Queue identifier of queue for which status is desired

12) Name: Get Time

Purpose: To request the ERTE supply its current time value to the requesting process

Parameters: - Get Time request code

13) Name: Set Priority Level

Purpose: To request ERTE to alter the priority level the process is running at (used to prevent interrupts at inappropriate moments)

Parameters: - Set Priority Level request code
- Priority level to set process to

3.6.10.2 Local Data

The following local data is maintained by the ERTE:

1) Name: Free Message List

Purpose: A list of message headers for unallocated message buffers. (A message header contains data pertinent to the allocation and manipulation of message buffers, including a pointer to the buffer itself.)

Content: Each entry in the list contains:

- Length: the length of valid data
- From: the task id of the sending process
- Queue: the queue identifier the message is to be delivered to
- Address: pointer to the message buffer
- Offset: process definable field, usually an offset of special significance into the message buffer
- Next: a pointer to the next message on the message queue
- This message: a pointer to this message header; used when message headers are returned to the ERTE from a process to find

the header quickly in the list of all message headers
- On Queue: a pointer to the queue the message header is currently on

2) Name: Message Queue List

Purpose: A list of queues which processes can use to receive incoming messages

Content: Each entry in the list consist of:

- Process: identifier of the process receiving messages via this queue
- Size: the maximum number of messages allowed on the queue at one time
- Count: number of messages currently on the queue
- First: pointer to the first message on the queue
- Last: pointer to the last message on the queue

3) Name: Process Header List

Purpose: To maintain a list of process information needed to permit processes to be suspended and restarted correctly

Content: The list contains an entry for each process in the system. Each entry contains:

- Id: process identifier
- Name: the name of the process
- Flags: status flags representing the state of the process
- Priority: scheduling priority of the process
- Events: the events, if any, the process is waiting on, maximum of 8
- Next: pointer to next process on the run queue
- Time Left: the time remaining before this process can run again

- Message Count: number of messages allocated to this process
- PCB Address: physical address of hardware Process Control Block (PCB) for process
- PCB: the hardware Process Control Block for the process; the PCB contains:
  - Stack pointers: one for each of four VAX modes (only Kernel mode and User mode stacks are used in IGW)
  - Process Registers: copies of process general registers are kept here (Program Counter is register 15)
  - Process Status Longword: a copy of the Processor Status Longword register for the process [2]
  - Program Base Register: points to start of program page table for P0 address space [2]
  - Program Length Register: points to start of page table for P1 address space [2]
  - Control Base Register: points to start of page table for P1 address space [2]
  - Control Length Register: length of P1 page table [2]

4) Name: Run Queue

Purpose: List of processes ready and wanting to run, in order of priority

Content: - First: pointer to first process on the run queue

3.6.10.3 Interrupts

All interrupts and exceptions are initially vectored to the ERTE TLC. The ERTE TLC examines the type of the exception or interrupt and selects an action based on the type. For interrupts, the ERTE will set an event corresponding to interrupt. The setting of this event will cause the process responsible for servicing the interrupting device to become runnable (if it is waiting for the event). The ERTE will then schedule the process to run based on its priority. The interrupts the ERTE will recognize are:

1) Console Receive,
2) Console Transmit,
3) Ethernet Receive/Transmit (one for each Ethernet interface),
4) IXIB Receive (one for each IXIB interface), and
5) IXIB Transmit (one for each IXIB interface).

The ERTE handles a Clock Tick interrupt within the ERTE alone. This is done to ensure that clock interrupts take as short a time as possible. The interrupt service routine in ERTE increments the IGW time counter maintained by ERTE, and then decrements the Time Left field of each process waiting for a time-out or sleeping.

The ERTE also implements a watchdog timer. At every Clock Tick interrupt the ERTE increments a counter. This counter is cleared every time the ERTE selects a process to run. If no process can be run in five minutes, then the system must be in trouble and the ERTE will initiate a reboot sequence. The ERTE determines that the five minute interval has passed when the counter it is incrementing reaches a large number representing five minutes worth of clock ticks.

All other interrupts should not occur. If one does occur, a message will be displayed on the console and the interrupt will be ignored.

The ERTE TLC will also process one exception condition. The Change Mode to Kernel (CHMK) exception occurs when a process executes a CHMK instruction. Processes use this instruction in system calls to deliberately create an exception which will cause the ERTE to service the call. After processing the call, the ERTE will select the next process to run and will run it.

All other exceptions should not be generated. If one is generated, then the ERTE will display a message on the console indicating the exception that occurred. The ERTE will then either ignore the exception (returning control to the interrupted process), or will reboot the IGW, depending on the seriousness of

the exception. Table 1 lists all unexpected exceptions and the IGW response to them.

Table 2: Unexpected Exception Responses

Exception	Response
Machine Check	Reboot
Kernel Stack Not Valid	Reboot
Power Fail	Reboot
Privileged Instruction	Reboot
Customer Reserved Instruction	Reboot
Reserved Operand	Reboot
Reserved Addressing Mode	Reboot
Access Control Violation	Reboot
Translation Not Valid	Reboot
Trace Pending	Reboot
Breakpoint Instruction	Reboot
Compatibility	Reboot
Arithmetic	Ignore
Change Mode To Execute	Ignore
Change Mode To Supervisor	Ignore
Change Mode to User	Ignore

3.6.10.4 Timing and Sequencing

The ERTE is the first TLC to be executed after booting is completed. The ERTE is then responsible for determining the sequencing for the remaining TLCs. The ERTE regains control only when an interrupt occurs or a process makes a system call where upon it executes only long enough to process the interrupt or system call and to schedule the next process to run.

3.6.10.5 Processing

The ERTE is first invoked by the Secondary Boot TLC, at which point it sets up and initializes its internal structures, selects the first process to run, and then runs it. Thereafter, the ERTE only executes after an interrupt or an exception occurs. When one of these events does occur, the ERTE determines the response to the event, executes the response, selects the next process to run, and runs it.

The ERTE responses to interrupts and exceptions is given above in Section 3.6.10.3. To service system calls, the ERTE determines the request in the call as described in Section 3.6.10.1 above, extracts the parameters, and performs the service. The services offered by ERTE are:

1) Suspend: the requesting process gives up the processor to other processes of higher or equal priority which are ready to run.

2) Sleep: the requesting process is held inactive for a requested period of time.

3) Wait On Event: the requesting process will be held inactive until the requested event is set.

4) Wait With Time-out: the requesting process is held inactive until the requested event occurs or the requested time-out period expires, whichever comes first.

5) Wait For One Of: the requesting process is held inactive until one of the requested events occurs or the requested time-out period expires, whichever comes first.

6) Message Get: a message buffer is allocated to the requesting process and a message header for the buffer is copied into the process space for the requesting process.

7) Message Discard: a message buffer allocated to a process is returned to the free message pool, and the message header for that buffer is removed from the process.

8) Open Message Queue: a message queue is assigned to the process and the queue is given the requested maximum size and queue identifier.

9) Message Send: the requested message header is unallocated from the requesting process and placed on the requested message queue, thus sending the message buffer from the requesting process to the process who has the indicated message queue open.

10) Message Receive: the first message header is removed from the requested message queue and copied to the requested process, thus allocating the message buffer to the requesting process.

11) Queue Status: the status of the requested queue is determined and returned to the requesting process.

12) Get Time: the current ERTE time value is returned to the requesting process.

13) Set Priority Level: the ERTE sets the processing priority level to prevent interrupts of lower than the requested priority.

3.6.10.6 Outputs

The ERTE TLC generates a single output for each system call supported by the TLC. The output is placed into the requesting process's register 0 (R0) when the ERTE completes the system call processing. In general, the output indicates the success or failure of the system call. The following lists the complete set of return values by system call:

1) Suspend: a success code is returned.

2) Sleep: a success code is returned.

3) Wait On Event: a success code is returned, unless the event is invalid when a failure code is returned.

4) Wait With Time-out: a failure code is returned if the requested event is invalid; otherwise, a success code is returned if the event occurred or a time-out code if the time-out occurred.

5) Wait For One Of: the event number of the first event is returned, or a time-out value if the time-out occurred.

6) Message Get: returns a success code if the message was allocated or an error code if no message could be allocated.

7) Message Discard: returns a success code.

8) Open Message Queue: returns a success code if the open succeeds, a busy code if the queue is already

open, or an error code if the queue could not be opened.

9) Message Send: returns a success code if the send succeeds an already queued code if the message is already on a queue, an invalid queue code if the queue is not opened, or a queue full code if the queue is full.

10) Message Receive: returns a success code if the received worked, a queue empty code if the queue was empty, and an invalid queue code if the queue was not opened.

11) Queue Status: returns a code indicating if the queue is open or not, and if open, the number of messages on the queue.

12) Get Time: returns ERTE time value.

13) Set Priority Level: returns the previous priority level.

4.0 GLOSSARY

ARP	- Address Resolution Protocol
ASCII	- American Standard Code for Information Interchange
CDD	- Console Device Driver
CHMK	- Change Mode to Kernel (VAX instruction)
CPU	- Central Processing Unit
CRC	- Communications Research Centre
CRT	- Cathode Ray Tube (a video terminal)
CSR	- Control/Status Register (in input output control register)
DARPA	- Defense Advanced Research Projects Agency
DEQNA	- Digital Equipment Corporation's Ethernet interface for Q-Bus
DMA	- Direct Memory Access
EDD	- Ethernet Device Driver
EGP	- Exterior Gateway Protocol
ERTE	- Efficient Real Time Executive
FIFO	- First In First Out (a queue)
ICMP	- Internet Control Message Protocol
IGW	- Internet Gateway
ILA	- IGW Link Area (an area of IGW memory)
IP	- Internet Protocol
IXIB	- Intelligent X.25 Interface Board
MTU	- Maximum Transmission Unit
OI	- Operator Interface
PCB	- Process Control Block
PPT	- Process Page Table
ROM	- Read-Only Memory
SCB	- System Control Block
SPT	- System Page Table
STAT	- Statistics processing component of the IGW software
SVA	- System Virtual Address
TCP	- Transmission control Protocol
TLC	- Top Level Component
TLD	- Top Level Design
XDD	- X.25 Device Driver
XON/XOFF	- Flow control on a serial line