

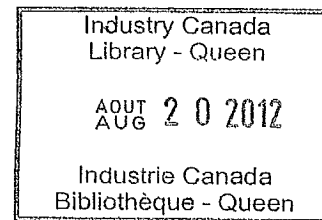
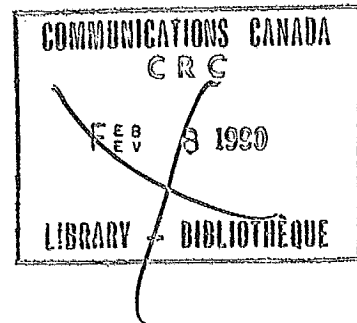
Software Kinetics

VOLUME 5
SOFTWARE DETAILED DESIGN DOCUMENT
FOR THE
INTERNETWORK GATEWAY
Submitted to: C.R.C.
Ottawa, Ontario

SKL Document #1500-15-031.03.0
Copy #3 05 May 1988

QA
76.9
S88
S6474
1988
v.5

IC



VOLUME 5
SOFTWARE DETAILED DESIGN DOCUMENT
FOR THE
INTERNETWORK GATEWAY
Submitted to: C.R.C.
Ottawa, Ontario

SKL Document #1500-15-031.03.0
Copy #3 05 May 1988

SOFTWARE DETAILED DESIGN DOCUMENT
FOR THE
INTERNETWORK GATEWAY PROJECT
VOLUME 5

Contract No. 36001-6-3535/02-ST

05 May 1988

Prepared for:

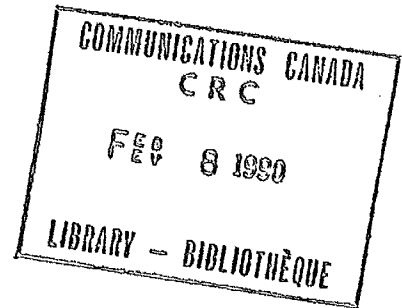
Communications Research Centre
Ottawa, Ontario

Prepared by:

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

SKL Document #1500-15-031.02.0

Q A
769
598
56494
1988
V-5



Document Approval Sheet
for the
Internetwork Gateway Project

Document No: 1500-15-031.02.0

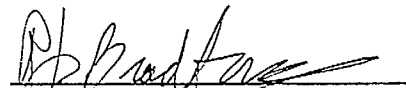
Document Name: Software Detailed Design Document
for the Internetwork Gateway Project

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

Document Revision History

<u>Revision</u>	<u>Description of Changes</u>	<u>Origin Date</u>
01	New Document Issued	23 September 1987
02	Coding and Integration	05 May 1988

TABLE OF VOLUMES

VOLUME 1	1.0 Introduction
	2.0 Referenced/Applicable Documents
	3.0 Design
	3.1 Interface Design
	3.2 Global Data
	3.3 TLC Design
	3.3.1 Efficient Real Time Executive (ERTE)
VOLUME 2	3.3.2 IP TLC
	3.3.3 EGP TLC
VOLUME 3	3.3.4 X.25 Device Driver (XDD) TLC
	3.3.5 Ethernet Device Driver (EDD) TLC
	3.3.6 Console Device Driver (CDD) TLC
VOLUME 4	3.3.7 Operator Interface TLC
	3.3.8 STAT TLC
VOLUME 5	3.3.9 Primary Boot TLC
	3.3.10 Secondary Boot TLC
	3.3.10.1 Local Boot LLC
	3.3.10.2 IGW Net Load LLC
	3.3.10.3 Host Net Load LLC
	3.3.11 Support Software
	4.0 Glossary

TABLE OF CONTENTS

3.3.9	Primary Boot TLC Detailed Design	1
3.3.10	Secondary Boot TLC Detailed Design	1
3.3.10.1	Local Boot LLC	3
3.3.10.1.1	Local Boot Architecture	3
3.3.10.1.2	Global Data	7
3.3.10.1.3	Local Boot LLCs	9
3.3.10.1.4	Secondary Boot Units	9
3.3.10.1.4.1	Add_To_PT Unit	9
3.3.10.1.4.2	Create_Int_Stack Unit	11
3.3.10.1.4.3	Define_Free_Mem Unit	12
3.3.10.1.4.4	Define_ILA Unit	14
3.3.10.1.4.5	File_Open Unit	16
3.3.10.1.4.6	File_Read Unit	18
3.3.10.1.4.7	File_Read_Line Unit	20
3.3.10.1.4.8	File_Seek Unit	22
3.3.10.1.4.9	Get_Flags Unit	24
3.3.10.1.4.10	Inet_Addr Unit	25
3.3.10.1.4.11	Link_IO_Pages Unit	28
3.3.10.1.4.12	Load_ACT_Tbl Unit	30
3.3.10.1.4.13	Load_ERTE Unit	33
3.3.10.1.4.14	Load_GW_Tbl Unit	36
3.3.10.1.4.15	Load_IXIB Unit	40
3.3.10.1.4.16	Load_NB_Tbl Unit	42
3.3.10.1.4.17	Load_Net_Tbl Unit	45
3.3.10.1.4.18	Load_SCB Unit	48
3.3.10.1.4.19	Main Unit	50
3.3.10.1.4.20	Panic Unit	52
3.3.10.1.4.21	Printf Unit	53
3.3.10.1.4.22	Read_Dir Unit	57
3.3.10.1.4.23	Read_Process_ List Unit	58
3.3.10.1.4.24	Read_Processes Unit	60
3.3.10.1.4.25	Reboot Unit	66
3.3.10.1.4.26	Relocate Unit	67
3.3.10.1.4.27	Reserve_SPT Unit	69
3.3.10.1.4.28	Size_Memory Unit	71
3.3.10.1.4.29	Start_ERTE Unit	73
3.3.10.2	IGW Net Load Component	75

3.3.10.2.1	Net Load Component Architecture	75
3.3.10.2.2	Global Data	78
3.3.10.2.3	IGW Net Load LLCs	79
3.3.10.2.4	IGW Net Load Units	79
3.3.10.2.4.1	Calc_Memory_Size Unit	79
3.3.10.2.4.2	Check_Dgram Unit	81
3.3.10.2.4.3	Check_IP Unit	83
3.3.10.2.4.4	Check_UDP_Hdr Unit	86
3.3.10.2.4.5	Chk_Sum Unit	89
3.3.10.2.4.6	Copy_Dgram Unit	90
3.3.10.2.4.7	Create_Dgram Unit	92
3.3.10.2.4.8	Create_Ip_Hdr Unit	94
3.3.10.2.4.9	Create_UDP_Hdr Unit	97
3.3.10.2.4.10	Download_Dgram Unit	100
3.3.10.2.4.11	Init_Ether Unit	102
3.3.10.2.4.12	Install_IXIB Unit	103
3.3.10.2.4.13	Install_SW Unit	106
3.3.10.2.4.14	Main Unit	109
3.3.10.2.4.15	Print_Msg Unit	110
3.3.10.2.4.16	Recv_Data Unit	111
3.3.10.2.4.17	Recv_Dgram Unit	114
3.3.10.2.4.18	Reboot Unit	117
3.3.10.2.4.19	Relocate Unit	118
3.3.10.2.4.20	Send_Dgram Unit	120
3.3.10.2.4.21	Send_Message	123
3.3.10.3	Host Net Load LLC	125
3.3.10.3.1	Host Net Load LLC Architecture	125
3.3.10.3.2	Global Data	128
3.3.10.3.4	Host Net Load Units	130
3.3.10.3.4.1	Add_To_PT Unit	130
3.3.10.3.4.2	Create_Int_Stack Unit	132
3.3.10.3.4.3	Define_Free_Mem Unit	134
3.3.10.3.4.4	Define_ILA Unit	136
3.3.10.3.4.5	Get_Flags Unit	138
3.3.10.3.4.6	Link_IO_Pages Unit	139
3.3.10.3.4.7	Load_ACT_Tbl Unit	141
3.3.10.3.4.8	Load_ERTE Unit	144
3.3.10.3.4.9	Load_GW_Tbl Unit	148
3.3.10.3.4.10	Load_IGW Unit	152
3.3.10.3.4.11	Load_IXIB Unit	154
3.3.10.3.4.12	Load_NB_Tbl Unit	156
3.3.10.3.4.13	Load_Net_Tbl Unit	159
3.3.10.3.4.14	Load_Reg Unit	163

3.3.10.3.4.15	Load_SCB Unit	164
3.3.10.3.4.16	Main Unit	166
3.3.10.3.4.17	Read_Process_ List Unit	169
3.3.10.3.4.18	Read_Processes Unit	171
3.3.10.3.4.19	Reserve_SPT Unit	177
3.3.10.3.4.20	Write_ILA Unit	179
3.3.11	SUPPORT SOFTWARE Detailed Design	180
3.3.11.1	Write_Diskettes Architecture	181
3.3.11.2	Write_Diskettes Global Data	182
3.3.11.3	Write_Diskettes LLC Design	183
3.3.11.4	Write_Diskettes Unit Design	183
3.3.11.4.1	Add_Files	183
3.3.11.4.2	Blocks	187
3.3.11.4.3	Directory_List	188
3.3.11.4.4	Init	192
3.3.11.4.5	Main	194
3.3.11.4.6	Transfer	199
4.0	GLOSSARY	201

#1500-15-031.02.0

3.3.9 Primary Boot TLC Detailed Design

This TLC has been eliminated from the design of the IGW because the Boot programs residing in ROM on the Micro-VAX contain all the functionality of the Primary Boot TLC as described in [5].

3.3.10 Secondary Boot TLC Detailed Design

The Secondary Boot TLC is responsible for booting the IGW from either floppy diskette, or from a host on the Ethernet. To accomplish this the TLC has been divided into three distinct lower level components (Figure 3-9):

- 1) Local Boot LLC - This component is responsible for booting the IGW entirely from floppy disks.
- 2) IGW Net Load LLC - This component is responsible for requesting and receiving boot software and data over the Ethernet network. This component resides on the IGW.
- 3) Host Net Load LLC - This component resides on a host, and is responsible for sending IGW software and data to the IGW when a request to do so is received.

The following sections describe each of these LLCs.

#1500-15-031.02.0

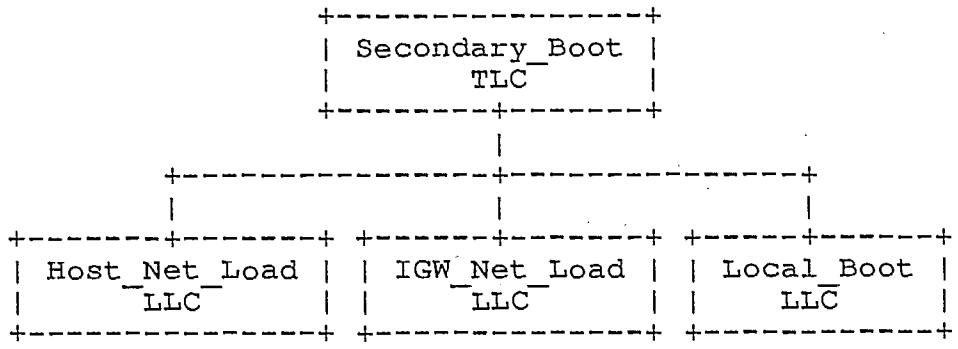


Figure 3-9

#1500-15-031.02.0

3.3.10.1 Local Boot LLC

The Local Boot LLC contains the software that is responsible for the loading of the IGW system from diskette, and for the initialization of memory hardware and tables. After this loading and initialization, the Local Boot software will transfer control to the IGW ERTE that has been loaded into main memory.

3.3.10.1.1 Local Boot LLC Architecture

The Local Boot LLC consists of the following units as shown in Figure 3-10:

- 1) Relocate Unit - This unit is used to relocate the secondary boot program to the end of memory to allow the loading of the IGW software.
- 2) Size_Memory Unit - This unit determines the amount of physical memory in the IGW.
- 3) Load_SCB Unit - This unit loads the System Control Block from a file on the IGW diskette.
- 4) Reserve_SPT Unit - This unit reserves the space required to contain the System Page Table.
- 5) Define_ILA Unit - This unit defines the structure of the IGW Link Area. This area contains information and pointers to information that are used globally by the IGW. As part of the ILA definition procedure SPT entries are added to the System Page Table to reference the pages of the IGW Link Area.
- 6) Load_ERTE Unit - This unit is responsible for the loading of the ERTE from diskette into IGW memory.

#1500-15-031.02.0

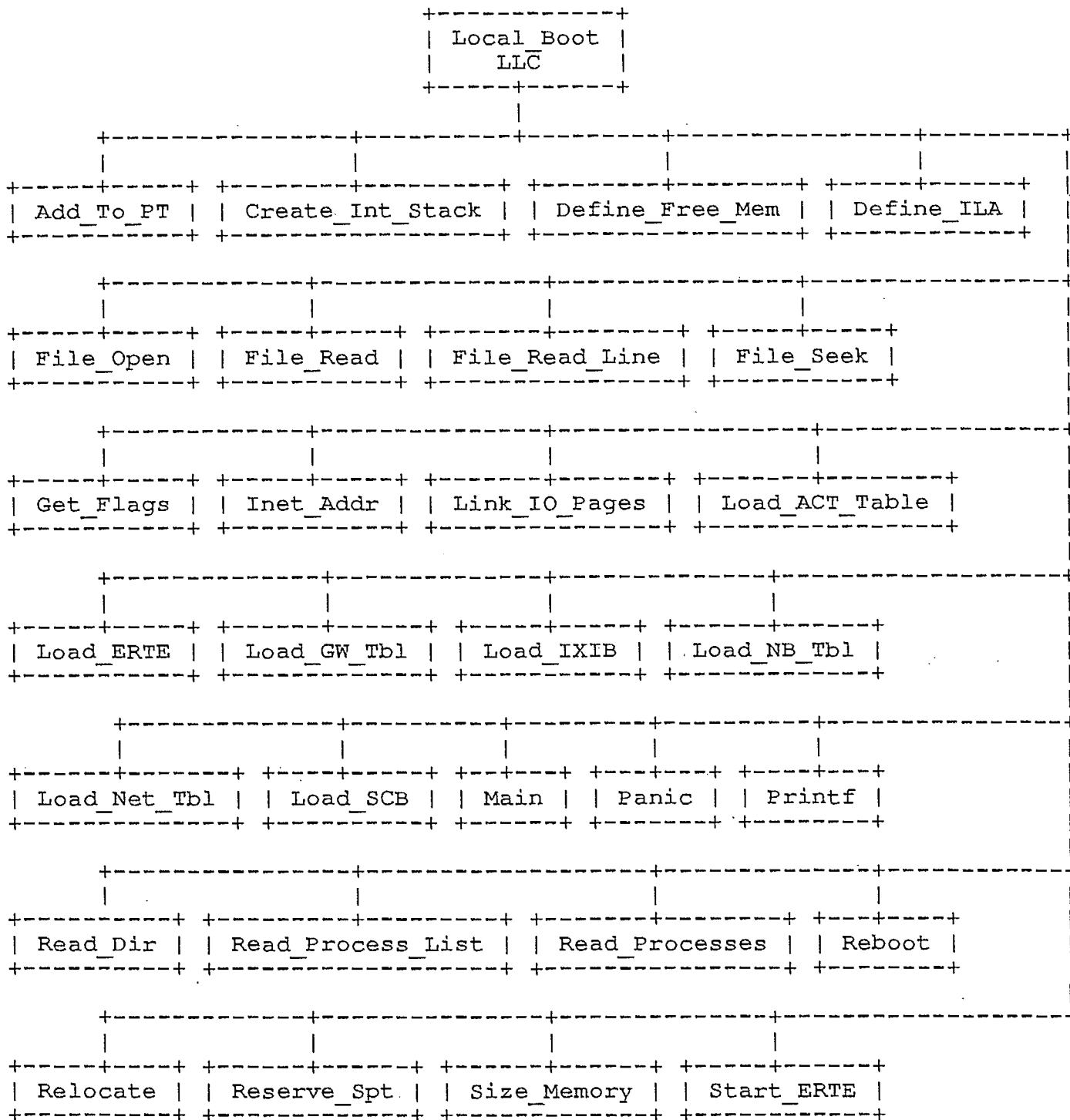


Figure 3-10

- 7) Read_Process_List Unit - This unit reads the list of processes that are to be loaded from diskette.
- 8) Read_Processes Unit - This unit makes use of the list obtained by the Read_Process_List Unit to read in the IGW processes from diskette. This unit also sets up PPT and SPT entries, allocates stack space (by the use of the Allocate_Stacks Unit), and updates the PCB in Process_List.
- 9) Load_ACT_Tbl Unit - This unit reads the X.121 Address Configuration Table from diskette and loads it into system virtual address space. The IGW Link Area is updated to reference the system virtual address of the loaded table.
- 10) Load_Net_Tbl Unit - This unit reads the X.121 Address Configuration Table from diskette and loads it into system virtual address space. The IGW Link Area is updated to reference the system virtual address of the loaded table.
- 11) Load_GW_Tbl Unit - This unit reads the Gateway Table from diskette and loads it into system virtual address space. The IGW Link Area is updated to reference the system virtual address of the loaded table.
- 12) Load_NB_Tbl Unit - This unit reads the Neighbor Table from diskette and loads it into system virtual address space. The IGW Link Area is updated to reference the system virtual address of the loaded table.
- 13) Create_Int_Stack Unit - This unit allocates space for the interrupt stack. System page table entries are added and hardware registers are set during the allocation procedure.
- 14) Define_Free_Mem Unit - This unit sets up the system page table entries required to reference the free memory of the IGW.
- 15) Link_IO_Pages Unit - This unit sets up a pointer in the IGW Link Area to reference the area of memory that is designated for I/O.
- 16) Load_IXIB Unit - This unit reads the IXIB software from diskette and sends it to the IXIB board to be loaded into the memory on that board.

#1500-15-031.02.0

- 17) File_Open Unit - This unit will obtain file description information required by the File_Read and File_Read_Line Units.
- 18) File_Read Unit - This unit will read data from a floppy disk file that has been opened by the File_Open Unit.
- 19) File_Read_Line Unit - This unit will read a single newline terminated record from the floppy disk file that has been opened with the File_Open Unit.
- 20) Add_To_PT Unit - This unit is called to add an entry to a page table.
- 21) Start_ERTE Unit - This unit transfers control from the Local Boot LLC to the ERTE TLC thus starting IGW gateway operations.
- 22) Printf - This unit is called to display formatted messages on the IGW console during the secondary boot procedure.
- 23) Reboot - This unit is used to perform a reboot of the IGW from diskette.
- 24) Panic - This unit displays an error message on the IGW console and causes the IGW to reboot from diskette.
- 25) Inet_Addr - This unit converts an IP address in dot notation to 32 bit representation.
- 26) Get_Flags - This unit converts a flags string into a bit pattern.
- 27) Main - This unit is called after secondary boot relocation and is responsible for calling the secondary boot units required to perform the boot.
- 28) File_Seek - This unit will reposition the file pointer for the currently open file.
- 29) Read_Dir - This unit will read the disk directory for both IGW diskettes from diskette number 0.

#1500-15-031.02.0

3.3.10.1.2 Global Data

This section describes the format and contents of the data which is defined to be globally used between the units contained within the secondary boot procedure.

- 1) RELOC - This constant defines the physical address that the secondary boot image is to be relocated to.
- 2) DIR_SIZE - This constant defines the number of entries that the Disk_Dir table will hold. This value is currently defined as 20.
- 3) Disk_Dir - This global data item contains a table of DIR_SIZE directory entries for the files contained on the IGW diskettes. The end of the directory is detected by either the predefined constant DIR_SIZE or if there are less entries by having the Dir_BN field of the entry after the last valid entry to be 0. Each directory entry contains the following fields:
 - Dir_Name - This field consists of an array of 15 bytes containing the file name stored as a null terminated character string.
 - Dir_Dev - This field consists of a single byte indicating the floppy drive number that the file is stored on.
 - Dir_BN - This field consists of a 16 bit integer containing the starting block number of the file on the diskette.
 - Dir_Size - This field consists of a 32 bit integer containing the size of the file in bytes.
- 4) FICS - This global data item holds the File Input Control Structure. This structure contains information pertaining to the currently opened file. This structure contains the following fields:
 - OI_Start_BN - This field consists of a 16 bit integer containing the block number of the first block in the currently open file.

#1500-15-031.02.0

- OI_Size - This field consists of a 32 bit integer containing the size of the currently open file in bytes.
- OI_Dev - This field consists of a 16 bit integer containing the floppy drive number where the currently open file resides.
- OI_Block_Offset - This field consists of a 16 bit integer containing the block offset from the start of the diskette of the currently open file.
- OI_Byte_Offset - This field consists of a 16 bit value indicating the bytes offset of the file read position in the current block being read in the currently open file.
- OI_Flags - This field consists of a 16 bit integer containing flags indicating information pertaining to the currently open file. Valid flags for this field are:

FILE_OPEN (0x01) - File has been opened.

- 5) Free_Phys - This global data item consists of a 32 bit integer containing the physical address of the start of memory that has not been allocated yet.
- 6) Free_Virt - This global data item consists of a 32 bit integer containing the virtual address of the start of virtual memory that has not been allocated yet.
- 7) Istack_Phys - This global data item consists of a 32 bit integer containing the physical address of the top of the interrupt stack.
- 8) Istack_Virt - This global data item consists of a 32 bit integer containing the virtual address of the top of the interrupt stack.
- 9) Proc_List - This global data item consists of an array of 100 bytes containing the names of the files that processes are to be loaded from. Each file name is separated by a newline character and the list is terminated by a null character.
- 10) Sys_PT - This global data item consists of a 32 bit integer containing the starting physical address of the

#1500-15-031.02.0

system page table.

3.3.10.1.3 Local Boot LLCs

There are no LLCs defined for the Local Boot LLC.

3.3.10.1.4 Local Boot Units

The following subsections contain the unit descriptions for the units comprising the Local Boot LLC.

3.3.10.1.4.1 Add_To_PT Unit

The Add_To_PT Unit adds page table entries to either the system or the process page tables.

3.3.10.1.4.1.1 Inputs

The following inputs are used by the Add_To_PT Unit:

- 1) PT_Addr - This input contains the starting physical address of the page table that page table entries are to be added to.
- 2) Phys_Addr - This input contains the physical address of the page that is to be added to the page table.
- 3) Virt_Addr - This input contains the virtual address of the page that is to be added to the page table.

#1500-15-031.02.0

3.3.10.1.4.1.2 Outputs

The following outputs are produced by the Add_To_PT Unit:

- 1) Page Tables - This output is written to the page table specified by the PT_Addr input. The format of these page tables is given in the global data section.

3.3.10.1.4.1.3 Local Data

No local data is defined for the Add_To_PT Unit.

3.3.10.1.4.1.4 Processing

Move PFN of Phys_Addr to address specified by VPN of Virt_Addr + PT_Addr
Set PT_Valid field in PT entry at VPN of Virt_Addr + PT_Addr
Move PT_UW to PT_Prot field in PT entry at VPN of Virt_Addr + PT_Addr
Return

3.3.10.1.4.1.5 Limitations

This unit performs no checks for incorrect virtual addresses, so specifying invalid virtual address could result in page table entries to be written to incorrect locations outside of the page table in physical memory.

#1500-15-031.02.0

3.3.10.1.4.2 Create_Int_Stack Unit

The Create_Int_Stack Unit Reserves an area in physical memory following the global tables to contain the interrupt stack.

3.3.10.1.4.2.1 Inputs

The following inputs are defined for the Create_Int_Stack Unit:

- 1) Free_Phys - This input is read from global data and contains the free physical memory address where the interrupt stack is located.
- 2) Free_Virt - This input is read from global data and contains the free virtual memory address where the interrupt stack is placed.

3.3.10.1.4.2.2 Outputs

The following outputs are produced by the Create_Int_Stack Unit:

- 1) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 2) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 3) Istack_Phys - This output is written to global data and contains the physical address of the initial interrupt stack pointer.
- 4) Istack_Virt - This output is written to global data and contains the virtual address of the initial interrupt stack pointer.

#1500-15-031.02.0

3.3.10.1.4.2.3 Local Data

No local data is defined for the Create_Int_Stack Unit.

3.3.10.1.4.2.4 Processing

Add ISTACK_SIZE to Free_Phys

Add ISTACK_SIZE to Free_Virt

Adjust Free_Phys and Free_Virt to point to next page boundary if necessary

Move Free_Phys to Istack_Phys

Move Free_Virt to Istack_Virt

Return

3.3.10.1.4.2.5 Limitations

No limitations are defined for the Create_Int_Stack Unit.

3.3.10.1.4.3 Define_Free_Mem Unit

The Define_Free_Mem Unit defines the system virtual addresses for the area in physical memory from the beginning of the tables to the end of the free memory.

#1500-15-031.02.0

3.3.10.1.4.3.1 Inputs

The following inputs are defined for the Define_Free_Mem Unit:

- 1) Table_Phys - This input is read from global data and contains the physical memory address where table storage begins.
- 2) Table_Virt - This input is read from global data and contains the virtual memory address where table storage begins.
- 3) Sys_Pt - This input is read from global data and contains the physical address of the system page table that has been defined by the Reserve_SPT Unit.

3.3.10.1.4.3.2 Outputs

The following outputs are produced by the Define_Free_Mem Unit:

- 1) SPT - This output is written to the system page table by the use of the Add_To_PT Unit, and contains new entries which are added to the system page table.
- 2) Free_Virt - This output is written to global data and contains the updated value for the next free address in physical memory.
- 3) ILA - This output is written to the ILA area and is updated with the virtual address for the start of the IGW free memory.

#1500-15-031.02.0

3.3.10.1.4.3.3 Local Data

No local data is defined for the Define_Free_Mem Unit.

3.3.10.1.4.3.4 Processing

```
Move Free_Virt to ILA entry for free virtual memory
For each page N starting at Table_Phys to end of physical memory
  Call Add_To_PT(Page_Table = Sys_PT,
    Virt_Addr = N * PAGE_SIZE + Table_Virt,
    Phys_Addr = N * PAGE_SIZE + Table_Phys)
Endfor
Return
```

3.3.10.1.4.3.5 Limitations

No limitations are defined for the Define_Free_Mem Unit.

3.3.10.1.4.4 Define_ILA Unit

The Define_ILA Unit Reserves an area in physical memory following the system page table. to contain the IGW Link Area. System page table entries are created for this area referencing system virtual addresses starting at the beginning of the system virtual address space.

#1500-15-031.02.0

3.3.10.1.4.4.1 Inputs

The following inputs are defined for the Define_ILA Unit:

- 1) Free_Phys - This input is read from global data and contains the free physical memory address where the ILA area is to be placed.
- 2) Free_Virt - This input is read from global data and contains the free virtual memory address where the ILA area is to be placed.
- 3) Sys_Pt - This input is read from global data and contains the physical address of the system page table that has been defined by the Reserve_SPT Unit.

3.3.10.1.4.4.2 Outputs

The following outputs are produced by the Define_ILA Unit:

- 1) SPT - This output is written to the system page table by the use of the Add_To_PT Unit, and contains new entries which are added to the system page table.
- 2) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 3) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 4) ILA - This output updates the IGW Link Area with any fields of that area that are to be initialized.

#1500-15-031.02.0

3.3.10.1.4.4.3 Local Data

No local data is defined for the Define_ILA Unit.

3.3.10.1.4.4.4 Processing

```
For each page N in ILA
  Call Add_To_PT(Page_Table = Sys_PT, Free_Virt, Free_Phys)
  Add PAGE_SIZE to Free_Phys
  Add PAGE_SIZE to Free_Virt
Endfor
Move Sys_Pt to SPT_Address field of ILA
Return
```

3.3.10.1.4.4.5 Limitations

No limitations are defined for the Define_ILA Unit.

3.3.10.1.4.5 File_Open Unit

The File_Open Unit searches through the memory resident disk directory for a specified file. If the file is found in the disk directory the File Input Control Structure (FICS) is initialized for that file and the value of NOERROR is returned. If the file isn't in the directory ERROR is returned.

#1500-15-031.02.0

3.3.10.1.4.5.1 Inputs

The following inputs are defined for the File_Open Unit:

- 1) Open_File_Name - This input consists of the name of the diskette file that is to be opened.
- 2) Disk_Dir - This input consists of the disk directory in global data that has been obtained by the File_Get_Dir Unit.

3.3.10.1.4.5.2 Outputs

The following outputs are produced by the File_Open Unit:

- 1) FICS - This output is the File Input Control Structure that has been initialized for the open file. It is defined as global data.

3.3.10.1.4.5.3 Local Data

No local data is defined for the File_Open Unit.

3.3.10.1.4.5.4 Processing

For each directory entry N in Disk_Dir

 If Dir_Name field of directory entry N matches Open_File_Name

 Move Dir_BN field of directory entry N to IO_Start_BN
 field of FICS

 Move Dir_Size field of directory entry N to IO_Size field
 of FICS

 Move Dir_Dev field of directory entry N to IO_Dev field
 of FICS

 Move IO_Start_BN field of FICS to IO_Block_Offset field of
 FICS

 Move 0x7FFFF hexadecimal to IO_Byte_Offset field of FICS

 Move FILE_OPEN to IO_Flags field of FICS

 Return NOERROR

 Endif

Endfor

Return ERROR

#1500-15-031.02.0

3.3.10.1.4.5.5 Limitations

The Read_Dir unit must be called before this unit.

3.3.10.1.4.6 File_Read Unit

The File_Read Unit will read a specified number of bytes from the diskette file that was opened with the File_Open Unit into a Buffer.

3.3.10.1.4.6.1 Inputs

The following inputs are defined for the File_Read Unit:

- 1) FICS - This input contains the File Input Control Structure, and contains the current file input status for the file being read. This input comes from global data.
- 2) Input_Buffer - This input is the address of the Input_Buffer output item.
- 3) Bytes_To_Read - This input contains the number of bytes that are to be read from the diskette file.
- 4) Diskette - This input consists of the file data that is read from the IGW diskette.

#1500-15-031.02.0

3.3.10.1.4.6.2 Outputs

The following outputs are produced by the File_Read Unit:

- 1) Input_Buffer - This output is an array of characters in which the input data is written to.
- 2) FICS - This output is the File Input Control Structure that has been updated to indicate the read data. It is defined as global data.

3.3.10.1.4.6.3 Local Data

The following local data is defined for this unit:

- 1) In_Ptr - This local data item is used to point to the input buffer in which data is being returned to the calling unit.

3.3.10.1.4.6.4 Processing

If FILE_OPEN Flag isn't set in IO_Flags Field of FICS

Return ERROR

Endif

Move Input_Buffer address to In_Ptr

Loop

While Bytes_To_Read is greater than 0 and IO_Byte_Offset field of FICS is less than 512 (1 block)

If (512 * IO_Start_BN + IO_Size < 512 * IO_Block_Offset + IO_Byte_Offset) using the fields in FICS

Return address given in In_Ptr - address of Input_Buffer

Endif

Move byte from IO_In_Buf field of FICS referenced by IO_Byte_Offset field of FICS to address referenced by In_Ptr

Increment IO_Byte_Offset field of FICS

Increment In_Ptr

Decrement Bytes_To_Read

Endwhile

If Bytes_To_Read is 0

Exit Loop

Else

Call ROM based disk driver routine to read the disk block addressed by the IO_Block_Offset field of FICS on the disk specified by the IO_Dev field of FICS to the IO_In_Buf field of FICS

#1500-15-031.02.0

Increment IO_Block_Offset field of FICS
Clear IO_Byte_Offset field of FICS

Endif

Endloop

Return address given in In_Ptr - address of Input_Buffer

3.3.10.1.4.6.5 Limitations

No limitations are defined for this unit.

3.3.10.1.4.7 File_Read_Line Unit

The File_Read_Line Unit will read a line of input from the diskette file that was opened with the File_Open Unit into a buffer.

3.3.10.1.4.7.1 Inputs

The following inputs are defined for the File_Read_Line Unit:

- 1) In_Ptr - This input contains the address of the Input_Buffer output.

#1500-15-031.02.0

3.3.10.1.4.7.2 Outputs

The following outputs are produced by the File_Read_Line Unit:

- 1) Input_Buffer - This output is an array of characters in which the input data is written to.

3.3.10.1.4.7.3 Local Data

The following local data is defined for this unit:

- 1) Bytes_Read - This local data item contains the number of bytes that have been read while performing a single character read.

3.3.10.1.4.7.4 Processing

Loop

```
Bytes_Read = File_Read(In_Ptr, 1)
If Bytes_Read isn't equal 1 or data referenced by In_Ptr
  is a <LF> character
  Exit Loop
Endif
In_Ptr++
```

Endloop

Move NULL character to data referenced by In_Ptr

Return

3.3.10.1.4.7.5 Limitations

No limitations are defined for this unit.

#1500-15-031.02.0

3.3.10.1.4.8 File_Seek Unit

The File_Seek Unit is used to position the file pointer of the open file to the specified byte offset.

3.3.10.1.4.8.1 Inputs

The following inputs are defined for the File_Seek Unit:

- 1) FICS - This input contains the File Input Control Structure, and contains the current file input status for the file being referenced. This input comes from global data.
- 2) Bytes_Offset - This input contains the new value of the byte offset into the file.
- 3) Diskette - This input consists of the file data that is read from the IGW diskette to load the internal data buffer in FICS.

3.3.10.1.4.8.2 Outputs

The following outputs are produced by the File_Seek Unit:

- 2) FICS - This output is the File Input Control Structure that has been updated to new file pointers. It is defined as global data.

#1500-15-031.02.0

3.3.10.1.4.8.3 Local Data

No local data is defined for this unit.

3.3.10.1.4.8.4 Processing

If FILE_OPEN Flag isn't set in IO_Flags Field of FICS

Return ERROR

Endif

If Byte_Offset is greater than or equal to IO_Size field of FICS

Return ERROR

Endif

Move Byte_Offset to IO_Block_Offset field of FICS

Divide IO_Block_Offset field of FICS by 512 placing remainder in the IO_Byte_Offset field of FICS

Add IO_Start_BN field of FICS to IO_Block_Offset field of FICS

Call ROM based disk driver routine to read the disk block

addressed by the IO_Block_Offset field of FICS on the disk

specified by the IO_Dev field of FICS to the IO_In_Buf field of FICS

Return NOERROR

3.3.10.1.4.8.5 Limitations

No limitations are defined for this unit.

#1500-15-031.02.0

3.3.10.1.4.9 Get_Flags Unit

The Get_Flags Unit is used to convert a character string containing flags to a bit representation of those flags.

3.3.10.1.4.9.1 Inputs

The following inputs are required by the Get_Flags Unit:

- 1) Set_Flags - This input consists of a character string containing the flags that are set.
- 2) All_Flags - This input consists of a character string containing all possible flags given in the correct bit order.

3.3.10.1.4.9.2 Outputs

The following outputs are produced by the Get_Flags Unit:

- 1) Flags - This output consists of the flags given in Set_Flags stored in bit positions.

#1500-15-031.02.0

3.3.10.1.4.9.3 Local Data

No local data is defined for the Get_Flags Unit.

3.3.10.1.4.9.4 Processing

Clear Flags

For each Flag from 0 to N - 1 in All_Flags

 If Flag N is in Set_Flags

 Bitwise or (1 left shifted by N) into Flags

 Endif

Endfor

Return Flags

3.3.10.1.4.9.5 Limitations

No limitations are defined for this unit.

3.3.10.1.4.10 Inet_Addr Unit

The Inet_Addr Unit is used to convert a character string containing an IP address in dot notation to a 32 bit value.

#1500-15-031.02.0

3.3.10.1.4.10.1 Inputs

The following inputs are required by the Inet_Addr Unit:

- 1) Addr_Ptr - This inputs points to a character string consisting of the IP address in dot notation.

3.3.10.1.4.10.2 Outputs

The following outputs are produced by the Inet_Addr Unit:

- 1) IP_Addr - This output contains the 32 bit representation of the IP address in the Addr_Ptr input.

3.3.10.1.4.10.3 Local Data

The following local data is used by the Inet_Addr_Unit:

- 1) Addr_Char - This local data item is used to hold the characters of the dot format IP address while stepping through it.
- 2) Addr_Parts - This local data item is an array of 3 unsigned 32 bit integers indexed from 0 to 2. It is used to hold the parts of IP address that have been converted to internal representation.
- 3) Dot_Count - This local data item is used to count the number of "." characters in the dot format IP address.

#1500-15-031.02.0

3.3.10.1.4.10.4 Processing

Loop

Clear IP_Addr

While Addr_Ptr points to non NULL data

Move byte pointed to by Addr_Ptr to Addr_Char

If Addr_Char is a digit

Multiply IP_Addr by 10

Add (Addr_Char - '0') to IP_Addr

Increment Addr_Ptr

Else

Exit Loop

Endif

Endwhile

If Addr_Ptr points to a '.' character

If Dot_Count is greater than or equal to 3

Return ERROR

Endif

Move IP_Addr to Entry indexed by Dot_Count in Addr_Parts

Increment Dot_Count

Increment Addr_Ptr

Else

Exit Loop

Endloop

If Addr_Ptr points to data other than NULL, SPACE, or TAB

Return ERROR

Case Dot_Count

1:

Left shift Addr_Parts entry 0 by 24 bits

And IP_Addr with 0xffffffff

Or Addr_Parts entry 0 into IP_Addr

2:

Left shift Addr_Parts entry 0 by 24 bits

And Addr_Parts entry 1 with 0xff

Left shift Addr_Parts entry 1 by 16 bits

And IP_Addr with 0xffff

Or Addr_Parts entry 0 into IP_Addr

Or Addr_Parts entry 1 into IP_Addr

3:

Left shift Addr_Parts entry 0 by 24 bits

And Addr_Parts entry 1 with 0xff

Left shift Addr_Parts entry 1 by 16 bits

And Addr_Parts entry 2 with 0xff

Left shift Addr_Parts entry 2 by 8 bits

And IP_Addr with 0xff

Or Addr_Parts entry 0 into IP_Addr

Or Addr_Parts entry 1 into IP_Addr

Or Addr_Parts entry 2 into IP_Addr

#1500-15-031.02.0

Endcase

Return IP_Addr in reverse byte order

3.3.10.1.4.10.5 Limitations

No limitations are defined for this unit.

3.3.10.1.4.11 Link_IO_Pages Unit

The Link_IO_Pages Unit is used to create system virtual address for the IO pages and store the starting virtual address of the IO pages in the ILA.

3.3.10.1.4.11.1 Inputs

No inputs are defined for the Link_IO_Pages Unit.

- 1) Free_Virt - This input is read from global data and contains the next free virtual address.
- 2) Sys_Pt - This input is read from global data and contains the address of the system page table.

#1500-15-031.02.0

3.3.10.1.4.11.2 Outputs

The following outputs are produced by the Link_IO_Pages Unit:

- 1) SPT - This output is written to the system page table by the use of the Add_To_PT Unit, and contains new entries which are added to the system page table.
- 2) ILA - This output is written to the IGW link area to indicate the starting system virtual address of the IO pages.

3.3.10.1.4.11.3 Local Data

No local data is defined for the Link_IO_Pages Unit.

3.3.10.1.4.11.4 Processing

```
Move Free_Virt to Link_IO field of ILA
For each page N in the IO Space
  Call Add_To_PT(Page_Table = Sys_PT,
    Virt_Addr = N * PAGE_SIZE + Free_Virt,
    Phys_Addr = N * PAGE_SIZE + IO_PHYS)
Endfor
Return
```

3.3.10.1.4.11.5 Limitations

No limitations are defined for the Link_IO_Pages Unit.

#1500-15-031.02.0

3.3.10.1.4.12 Load_ACT_Tbl Unit

The Load_ACT_Tbl Unit loads X.25 Address Configuration Table from diskette to the system virtual address space following the process page tables. The ILA is updated to indicate the correct address of this table.

3.3.10.1.4.12.1 Inputs

The following inputs are used by the Load_ACT_Tbl Unit:

- 1) X.25_ACT - This input is read from the file "x.25_act" on the IGW diskette and contains a copy of the X.25 Address Configuration Table. This file contains the following fields:
 - X121 - This field contains the X.121 address (1 to 15 bytes) of the table entry.
 - Inet - This field contains the IP address in dot notation for the table entry.
 - Size - This field contains the maximum size for a packet for the host described in the table entry.
 - Flags - This field contains flags describing a table entry.
- 2) Free_Phys - This input is read from global data and contains the free physical memory address where the X.25 Address Configuration Table is to be placed.
- 3) Free_Virt - This input is read from global data and contains the free virtual memory address where the X.25 Address Configuration Table is to be placed.

3.3.10.1.4.12.2 Outputs

The following outputs are produced by the Load_ACT_Tbl Unit:

- 1) ACT_Table - This output is written to the global data area as new entries are added to the X.25 Address Configuration Table. This table contains the following fields:

ACT_X121 - This field consists of a 16 byte character string the X.121 address of the current entry.

ACT_Inet - This field consists of a 32 bit value indicating the IP address for the current entry.

ACT_Size - This field consists of a 16 bit value indicating the maximum size of a packet for the current entry.

ACT_Flags - This field consists of a 32 bit value containing the following flags.

REQ_REV (0x01) - Request reverse charging.
ACC_REV (0x02) - Accept reverse charging.
REJ_IN (0x04) - Reject incoming calls.
REJ_OUT (0x08) - Reject outgoing calls.
IXIB (0x10) - Remote is an IXIB.

- 2) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 3) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 4) ILA - This output is written to the IGW Link Area and is updated with the address of the X.25 Address Configuration Table.

#1500-15-031.02.0

3.3.10.1.4.12.3 Local Data

The following local data is defined for the Load_ACT_Tbl Unit:

- 1) Input_Buffer - This data consists of a buffer of 100 bytes that is used to read records from the x.25_act file.
- 2) Host - This data item is used to hold the dot notation format of the IP addresses as they are read from the x.25_act file.
- 3) Flags - This data item is a character string used to hold the flags field of each entry that is read from x.25_act file.
- 4) ACT_Ptr - This data item is used to step through ACT_Table while adding table entries.

3.3.10.1.4.12.4 Processing

```
If result of File_Open("x.25_act") is less than 0
    Call panic(Error message)
Endif
Clear ACT_Table
Move Free_Virt to ACT_Table pointer in ILA
Move Free_Phys to ACT_Ptr
While more data in x.25_act file
    Call File_Read_Line(Input_Buffer, bytes to read = 100)
    If first character in Input_Buffer is a '#'
        Continue next loop iteration
    Endif
    If ACT_Table is full
        Exit loop
    Endif
    If result of sscanf(Input_Buffer, "%s %s %d %s",
        Act_X121 field of ACT_Table entry pointed to by ACT_Ptr,
        Host,
        Address of ACT_Size field of ACT_Table entry pointed to by ACT_PTR,
        Flags) is -1
        Exit Loop
    Endif
    ACT_Inet field of ACT_Table entry pointed to by ACT_Ptr =
    Inet_Addr(Host)
    If ACT_Inet field of ACT_Table = -1
        Call Printf(error message indicating invalid ACT Entry)
        Continue next loop iteration
```

#1500-15-031.02.0

```
Endif
ACT_Flags field of ACT_Table entry pointed to by ACT_Ptr =
  bitwise or between ACT_VALID flag and Get_Flags(Flags, "RAIOX")
Set ACT_Ptr to point to next entry in ACT_Table
Endwhile
Add size of ACT_Table to Free_Phys
Add size of ACT_Table to Free_Virt
Return
```

3.3.10.1.4.12.5 Limitations

No limitations are defined for the Load_ACT_Tbl Unit.

3.3.10.1.4.13 Load_ERTE Unit

The Load_ERTE Unit is responsible for the loading of the ERTE executable image from diskette to the IGW memory. For each page of ERTE that is loaded a system page table entry is created.

3.3.10.1.4.13.1 Inputs

The following inputs are defined for the Load_ERTE Unit:

- 1) ERTE - This input is read from the file "ERTE" on the IGW diskette and contains the header, text, data, and bss areas of the ERTE executable.
- 2) Free_Phys - This input is read from global data and contains the free physical memory address where ERTE is to be placed.
- 3) Free_Virt - This input is read from global data and contains the free virtual memory address where ERTE is to be placed.

#1500-15-031.02.0

- 4) Sys_Pt - This input is read from global data and contains the physical address of the system page table that has been defined by the Reserve_SPT Unit.

3.3.10.1.4.13.2 Outputs

The following outputs are produced by the Load_ERTE Unit:

- 1) ERTE Memory Image - This output is written to the main memory of the IGW and contains the text, data, and bss areas for ERTE.
- 2) SPT - This output is written to the system page table by calling the Add_To_PT Unit, and contains the new page table entries for the memory occupied by ERTE.
- 3) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 4) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.

3.3.10.1.4.13.3 Local Data

The following local data is defined for the Load_ERTE Unit:

- 1) Counter - This local data item is an integer that is used as a byte counter while clearing the bss area of ERTE.
- 2) Exec_Header - This local data item is a structure that is used to hold the header from the ERTE executable that is loaded from diskette. The fields in this structure are all 32 bit values and are defined as follows:

A_Magic - This field contains the type of the executable image that is being loaded. Valid values for this field are:

#1500-15-031.02.0

OMAGIC (0407) - Old impure format.
NMAGIC (0410) - Read-only text.
ZMAGIC (0413) - Demand load format.

A_Text - This field contains the size of the text segment in bytes.

A_Data - This field contains the size of the initialized data segment in bytes.

A_Bss - This field contains the size of the uninitialized data segment in bytes.

A_Syms - This field contains the size of the symbol field.

A_Entry - This field contains of the address of the entry point of the loaded executable image.

A_Trsize - This field contains the size of the text relocation area.

A_Drsize - This field contains the size of the data relocation area.

3.3.10.1.4.13.4 Processing

```
If result of File_Open("ERTE") is less than 0
    Call Panic(error message)
Endif
Bytes_Read = File_Read(address of Exec_Header,
    size of Exec_Header)
If Bytes_Read not equal size of Exec_Header
    Call Panic(error message)
Endif
If A_Magic field of Exec_Header is ZMAGIC
    If result of File_Seek(Offset = 1024) is less than 0
        Call Panic(error message)
    Endif
Else if A_Magic field of Exec_Header isn't one of OMAGIC or NMAGIC
    Call Panic(error message)
Endif
Bytes_Read = Read_File(Free_Phys,
    A_Text field of Exec_Header + A_Data field of Exec_Header)
If Bytes read not equal (A_Text field of Exec_Header + A_Data
    field of Exec_Header)
```

#1500-15-031.02.0

```
    Call Panic(error message)
Endif
For each page N in ERTE text, data, and bss areas
    Call Add_To_PT(Page_Table = Sys_PT, Free_Virt, Free_Phys)
    Free_Phys += PAGE_SIZE
    Free_Virt += PAGE_SIZE
Endfor
Clear bss area of ERTE
Endfor
Return
```

3.3.10.1.4.13.5 Limitations

No limitations are defined for the Load_ERTE Unit.

3.3.10.1.4.14 Load_GW_Tbl Unit

The Load_GW_Tbl Unit loads the Gateway Table from the file "gateway" on diskette into the IGW main memory. This file is used to define the various gateways that the IGW may access in addition to those determined through EGP.

#1500-15-031.02.0

3.3.10.1.4.14.1 Inputs

The following input is used by the Load_GW_Tbl Unit:

- 1) Gateway - This input is read from the diskette file "gateway" contains a copy of the Gateway Table. This file contains the following fields:

Dst_Net - The destination network that is accessed by a gateway table entry.

GW_Addr - The address of the gateway to route packets for the specified destination network.

Mask - The IP network address mask. This field consists of a hexadecimal constant specifying the IP network address mask for the destination network.

Hop - The number of gateways that must be crossed to reach the destination.

Flags - This field consists of user definable flags. Valid flags are:

E - Report route via EGP.

G - Gatewayed host. Delete route if the gateway goes down.

R - Attempt to reroute datagrams if the gateway goes down.

- 2) Free_Phys - This input is read from global data and contains the free virtual memory address where the gateway table is placed.
- 3) Free_Virt - This input is read from global data and contains the free virtual memory address where the gateway table is to be placed.

#1500-15-031.02.0

3.3.10.1.4.14.2 Outputs

The following outputs are produced by the Load_GW_Tbl Unit:

- 1) GW_Table - This output is written to the global data area as new entries are added to the Gateway Table. This table contains the following fields:
- 2) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 3) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 4) ILA - This output is written to the IGW Link Area and is updated with the address of the gateway table.

3.3.10.1.4.14.3 Local Data

The following local data is defined for the Load_GW_Tbl Unit:

- 1) Input_Buffer - This data consists of a buffer of 100 bytes that is used to read records from the gateway file.
- 2) GW_Addr - This data item consists of a character string used to hold the IP address for each entry that is read from the gateway file.
- 3) Dst_Net - This data item consists of a character string used to hold the destination network number for each entry that is read from the gateway file.
- 4) Flags - This data item is used to hold the flags field of each entry that is read from the gateway file.
- 5) GW_Ptr - This data item is used to step through GW_Table while adding table entries.

#1500-15-031.02.0

3.3.10.1.4.14.4 Processing

```
If result of File_Open("gateway") is less than 0
    Call Panic(error message)
Endif
Clear entries in GW_Table
Move Free_Virt to GW_Table pointer in ILA
Move Free_Phys to GW_Ptr
While more data in gateway file
    Call File_Read_Line(Input_Buffer, bytes to read = 100)
    If first character in input buffer is a '#'
        Continue next loop iteration
    Endif
    If GW_Table is full
        Exit loop
    Endif
    If result of sscanf(Input_Buffer, "%s %s %x %d %s %s",
        Dst_Net, GW_Addr,
        Address of GW_Mask field of GW_Table entry pointed to by GW_Ptr,
        Address of GW_Hop field of GW_Table entry pointed to by GW_Ptr,
        Flags) is -1
        Exit Loop
    Endif
    GW_Dst_Net field of GW_Table entry pointed to by GW_Ptr =
        Inet_Addr(Dst_Net)
    If GW_Dst_Net field of GW_Table entry pointed to by GW_Ptr = -1
        Call Printf(error message indicating invalid gateway entry)
        Continue next loop iteration
    Endif
    GW_GW_Addr field of GW_Table entry pointed to by GW_Ptr =
        Inet_Addr(GW_Addr)
    If GW_GW_Addr field of GW_Table entry pointed to by GW_Ptr = -1
        Call Printf(error message indicating invalid gateway entry)
        Continue next loop iteration
    Endif
    Move index of entry in Net_Table with same network address as
    the network portion of the gateway address to GW_Number
    field of GW_Table entry pointed to by GW_Ptr
    GW_Flags field of GW_Table entry pointed to by GW_Ptr =
    bitwise or between GW_VALID flag and Get_Flags(Flags, "EGR")
    Set GW_Ptr to point to next entry in GW_Table
Endwhile
Add size of GW_Table to Free_Phys
Add size of GW_Table to Free_Virt
Return
```

#1500-15-031.02.0

3.3.10.1.4.14.5 Limitations

The unit Load_Net_Table must be executed before this unit.

3.3.10.1.4.15 Load_IXIB Unit

The Load_IXIB Unit is responsible for the loading of the IXIB communications software from diskette to the IXIBs.

3.3.10.1.4.15.1 Inputs

The following input is defined for the Load_IXIB Unit:

- 1) IXIB File - This input is read from the file "IXIB" on the IGW diskette and contains the IXIB software in Motorola S-record format.

3.3.10.1.4.15.2 Outputs

The following output is produced by the Load_IXIB Unit:

- 1) IXIB - This output is written to each IXIB by way of the IXIB FIFO registers. The output written consists of the IXIB File.

#1500-15-031.02.0

3.3.10.1.4.15.3 Local Data

The following local data is defined for the Load_IXIB Unit:

- 1) Input_Buffer - This local data item consists of an array of 512 bytes and is used to hold data from the IXIB file while loading the IXIBs.
- 2) Bytes_Read - This local data is used to hold the number of bytes that have been read from a File_Read request.
- 3) Ibuf_Index - This local data item used as an index to the Input_Buffer while sending data from that buffer to the IXIBs.

3.3.10.1.4.15.4 Processing

```
If the result of File_Open("IXIB")
  Call Panic(error message)
Endif
While data remains to be read in IXIB File
  Bytes_Read = File_Read(Input_Buffer, 512)
  For each byte in Input_Buffer
    Move byte from input buffer to IXIB FIFO for
    each IXIB
  Endfor
Endwhile
Return
```

#1500-15-031.02.0

3.3.10.1.4.15.5 Limitations

No limitations are defined for the Load_IXIB Unit.

3.3.10.1.4.16 Load_NB_Tbl Unit

The Load_NB_Tbl Unit loads the EGP Neighbour Table from the file "neighbour" on diskette into the IGW main memory. This file is used to define information describing the gateways that the IGW can communicate via EGP with.

3.3.10.1.4.16.1 Inputs

The following inputs are used by the Load_NB_Tbl Unit:

- 1) Neighbour - This input is read from the diskette file "neighbor" and contains a copy of the EGP Neighbour Table. This file contains the following fields:

IP_ADDR - The Internet address of the EGP neighbour gateway in dot notation.

Flags - This field consists of user definable flags. Valid flags are:

- M - Gateway is a main neighbour.
- O - Gateway is an alternate neighbour.
- S - Gateway is a stub gateway.

- 2) Free_Phys - This input is read from global data and contains the free physical memory address where the neighbour table is to be placed.
- 3) Free_Virt - This input is read from global data and

#1500-15-031.02.0

contains the free virtual memory address where the neighbour table is to be placed.

3.3.10.1.4.16.2 Outputs

The following outputs are produced by the Load_NB_Tbl Unit:

- 1) NB_Table - This output is written to the global table area of physical memory as new entries are added to the Neighbour Table.
- 2) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 3) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 4) ILA - This output is written to the IGW Link Area and is updated with the address of the neighbour table.

3.3.10.1.4.16.3 Local Data

The following local data is defined for the Load_NB_Tbl Unit:

- 1) Input_Buffer - This data consists of a buffer of 100 bytes that is used to read records from the neighbour file.
- 2) IP_Addr - This data item consists of a character string used to hold the IP address of each EGP neighbour gateway entry that is read from the neighbour file.
- 3) Flags - This data item is used to hold the flags field of each entry that is read from the neighbour file.
- 4) NB_Ptr - This data item is used to step through NB_Table while adding table entries.

#1500-15-031.02.0

3.3.10.1.4.16.4 Processing

```
If result of File_Open("neighbour") is less than 0
    Call Panic(error message)
Endif
Clear entries in NB_Table
Move Free_Virt to NB_Table pointer in ILA
Move Free_Phys to NB_Ptr
While more data in neighbour file
    Call File_Read_Line(Input_Buffer, bytes to read = 100)
    If first character in input buffer is a '#'
        Continue next loop iteration
    Endif
    If NB_Table is full
        Exit loop
    Endif
    If result of sscanf(Input_Buffer, "%s %s", IP_Addr, Flags) is -1
        Exit Loop
    Endif
    NB_IP_Addr field of NB_Table entry pointed to by NB_Ptr =
        Inet_Addr(IP_Addr)
    If NB_IP_Addr field of NB_Table entry pointed to by NB_Ptr = -1
        Call Printf(error message indicating invalid neighbour entry)
        Continue next loop iteration
    Endif
    NB_Flags field of NB_Table entry pointed to by NB_Ptr =
        bitwise or between NB_VALID flag and Get_Flags(Flags, "MOS")
    Set NB_Ptr to point to next entry in NB_Table
Endwhile
Add size of NB_Table to Free_Phys
Add size of NB_Table to Free_Virt
Return
```

#1500-15-031.02.0

3.3.10.1.4.16.5 Limitations

No limitations are defined for the Load_NB_Tbl Unit.

3.3.10.1.4.17 Load_Net_Tbl Unit

The Load_Net_Tbl Unit Loads the Network Table from the file "network" on diskette into the IGW main memory. This file is used to define the network interface information required for each network that the IGW is connected to.

3.3.10.1.4.17.1 Inputs

The following inputs are used by the Load_Net_Tbl Unit:

- 1) Network - This input is read from the file "network" on the IGW diskette and contains a copy of the Network Table. This file contains the following fields:

IP_Addr - The local Internet address of the IGW on the referenced network. This field is a string containing the IP address in dot notation.

Interface_Id - The interface number of the network interface represented by this entry. Each interface is given a number which is used to direct datagrams to the correct interface for transmission.

Mask - The IP network address mask. This field consists of a hexadecimal constant specifying the IP network address mask.

MTU - The maximum transmission unit for IP datagrams.

#1500-15-031.02.0

This value is specified as an integer.

Flags - This field consists of one user definable flag which is "U" indicating that the interface should be marked as being up.

- 2) Free_Phys - This input is read from global data and contains the free physical memory address where the gateway table is to be placed.
- 3) Free_Virt - This input is read from global data and contains the free virtual memory address where the gateway table is to be placed.

3.3.10.1.4.17.2 Outputs

The following outputs are produced by the Load_Net_Tbl Unit:

- 1) Net_Table - This output is written to the global data area as new entries are added to the Network Table. This table is defined in section 3.2.
- 2) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 3) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 4) ILA - This output is written to the IGW Link Area and is updated with the address of the gateway table.

#1500-15-031.02.0

3.3.10.1.4.17.3 Local Data

The following local data is defined for the Load_Net_Tbl Unit:

- 1) Input_Buffer - This data consists of a buffer of 100 bytes that is used to read records from the network file.
- 2) IP_Addr - This data item is used to hold the IP address for each entry that is read from the network file.
- 3) Flags - This data item is used to hold the flags field of each entry that is read from the network file.
- 4) Net_Ptr - This data item is used to step through Net_Table while adding table entries.
- 5) Interface_Number - This data item is used to hold the interface id number of each entry read from the network file.
- 6) Mask - This data item is used to hold the address mask for each entry read from the network file.
- 7) MTU - This data item is used to hold the network MTU for each entry read from the network file.

3.3.10.1.4.17.4 Processing

```
If result of File_Open("network") is less than 0
    Call Panic(Error message)
Endif
Clear entries in Net_Table
Move Free_Virt to Net_Table pointer in ILA
Move Free_Phys to Net_Ptr
While more data in network file
    Call File_Read_Line(Input_Buffer, bytes to read = 100)
    If first character in input buffer is a '#'
        Continue next loop iteration
    Endif
    If Net_Table is full
        Exit loop
    Endif
    If result of sscanf(Input_Buffer, "%s %s %x %d %s",
        IP_Addr, Interface_Number, Net_Mask, MTU, Flags) is -1
        Exit Loop
    Endif
```

#1500-15-031.02.0

```
Search Net_Table for an entry with Net_IP_Addr field = IP_Addr
If table entry found
    Add Interface_Number to the end of the Net_QID_List for the
    found entry
Else
    Set Net_Ptr to the first empty position in Net_Table
    Net_IP_Addr field of Net_Table entry pointed to by Net_Ptr =
    Inet_Addr(IP_Addr)
    If Net_IP_Addr field Net_Table entry pointed to by Net_Ptr = -1
        Call Printf(error message indicating invalid network Entry)
        Continue next loop iteration
    Endif
    Net_Flags field of Net_Table entry pointed to by Net_Ptr =
    bitwise or between NET_VALID and Get_Flags(Flags, "U")
    Set Net_MTU referenced by Net_Ptr to MTU
    Set Net_Mask referenced by Net_Ptr to Mask
    Set Current_IF field referenced by Net_Ptr to zero.
Endwhile
Add size of Net_Table to Free_Phys
Add size of Net_Table to Free_Virt
Return
```

3.3.10.1.4.17.5 Limitations

No limitations are defined for the Load_Net_Tbl Unit.

3.3.10.1.4.18 Load_SCB Unit

The Load_SCB Unit loads the System Control Block from a file to the SCB area of memory.

#1500-15-031.02.0

3.3.10.1.4.18.1 Inputs

The following inputs are used by the Load_SCB Unit:

- 1) SCB_Init - This input is read from a file on the IGW diskette. This file contains an image of the System Control Block.

3.3.10.1.4.18.2 Outputs

The following outputs are produced by the Load_SCB Unit:

- 1) SCB - This output is written to the address specified by the SCBB processor register (physical address 0), and contains the System Control Block that has been obtained from the SCB_init input.
- 2) SCBB - This output is written to the System Control Block Base Register, and contains the physical address of the SCB.
- 3) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.

3.3.10.1.4.18.3 Local Data

No local data is defined for the Load_SCB Unit.

#1500-15-031.02.0

3.3.10.1.4.18.4 Processing

```
If result of Open_File("SCB_init") is less than 0
    Call printf(Error message)
    Call Reboot()
Endif
Bytes_Read = File_Read(Address of SCB = 0, Size of SCB = 512)
If Bytes_Read not equal Size of SCB
    Call printf(Error message)
    Call Reboot()
Endif
Move 0 to the system control block base register (processor
register SCBB)
Move Size of SCB to Free_Phys
Return
```

3.3.10.1.4.18.5 Limitations

This unit must be called before any other unit which allocates physical memory.

3.3.10.1.4.19 Main Unit

The Main Unit is called after secondary boot relocation. This unit is responsible for calling the other units that make up the secondary boot procedure.

#1500-15-031.02.0

3.3.10.1.4.19.1 Inputs

No inputs are defined for the Main Unit.

3.3.10.1.4.19.2 Outputs

The following outputs are produced by the Main Unit:

- 1) Console - This output is written to the IGW console and contains messages to indicate the start and end of the IGW loading procedure.

3.3.10.1.4.19.3 Local Data

No local data is defined for the Main Unit.

3.3.10.1.4.19.4 Processing

```
Call Printf("Loading IGW...")
Call Read_Dir()
Call Load_SCB()
Call Reserve_SPT()
Call Define_ILA()
Call Size_Memory()
Call Load_ERTE()
Call Read_Process_List()
Call Read_Processes()
Move Free_Phys to Table_Phys
Move Free_Virt to Table_Virt
Call Load_ACT_Tbl()
Call Load_Net_Tbl()
Call Load_GW_Tbl()
Call Load_NB_Tbl()
Reserve space from unloaded tables
Call Create_Int_Stack()
Call Define_Free_Mem()
Call Link_IO_Pages()
Call Load_IXIB()
Call Printf("done.\n")
```

#1500-15-031.02.0

Call Start_ERTE()

3.3.10.1.4.19.5 Limitations

No limitations are defined for the Main Unit.

3.3.10.1.4.20 Panic Unit

The Panic Unit causes the IGW to display a panic message on the console and perform a reboot.

3.3.10.1.4.20.1 Inputs

The following inputs are required by the Panic Unit:

- 1) Panic_Message - This input is a null terminated character string that contains a message that is to be printed on the IGW console.

#1500-15-031.02.0

3.3.10.1.4.20.2 Outputs

The following output is produced by the Panic Unit:

- 1) `Panic_Message` - This output is the same as the Panic Message input except for the fact that it is preceded by the message "panic: " and is followed by a newline.

3.3.10.1.4.20.3 Local Data

No local data is defined for the Panic Unit.

3.3.10.1.4.20.4 Processing

```
Call printf("panic: %s\n", panic message)
Call Reboot()
```

3.3.10.1.4.20.5 Limitations

No limitations are defined for this unit.

3.3.10.1.4.21 Printf Unit

The Printf Unit will send formatted text strings to the operator's console during the boot procedure.

3.3.10.1.4.21.1 Inputs

The following inputs are used by the Printf Unit:

- 1) Format_String - This input consists of a null terminated string that is used to format the output produced by this unit. A '%' character in this string is treated specially. The '%' character indicates to this unit that the following character indicates a data type that is to be printed from the next next item to be formatted. The following special characters may follow a '%':

x, X	- Print argument as a 32 bit hexadecimal value.
d, D, u	- Print argument as a 32 bit decimal value.
s	- Print null terminated string pointed to by argument.
c	- Print 8 bit character representation of argument.
%	- Print a '%' character.

This input is passed as the address of the format string to the Printf Unit.

- 2) Items_To_Be_Formatted - This input consists of the data that is to be formatted.
- 3) TXCS - This input contains the Console Transmitter Control Status Register, and is used to examine the status of the console transmitter.

#1500-15-031.02.0

3.3.10.1.4.21.2 Outputs

The following outputs are produced by the Printf Unit:

- 1) Formatted Message - This output is the formatted version of the message inputs. It is sent to the operator console one character at a time.
- 2) TXCS - This output is written to the Console Transmitter Control Status Register during the process of character transmission.
- 3) TXDB - This output is written to the Console Transmitter Data Buffer Register, and contains the bytes of the data that is to be sent to the console.

3.3.10.1.4.21.3 Local Data

The following local data is defined for the Printf Unit:

- 1) Output_Buffer - This local data item is an array of 100 bytes that is used to hold the string to be displayed after it has been formatted.
- 2) Save_TXCS - This local data item is used to save the value of TXCS during transmission of characters to the console.
- 3) Timeout - This local data item is used for a timeout counter while checking TXCS.

#1500-15-031.02.0

3.3.10.1.4.21.4 Processing

```
sprintf(Output_Buffer, Format_String, Items_To_Be_Formated)
Move 30000 to Timeout
For each character N in Output_Buffer
  While TXCS_RDY bit of Processor Register TXCS is clear
    Decrement Timeout
    If Timeout is less than or equal to 0
      Exit Loop
    Endif
  Endwhile
  If character N of Output_Buffer is a NULL character
    Exit Loop
  Endif
  Move Processor Register TXCS to Save_TXCS
  Clear Processor Register TXCS
  Move character N of Output_Buffer to Processor Register TXDB
  If character N of Output_Buffer is a <LF>
    Call Printf("\r")
  Endif
  Move 30000 to Timeout
  While TXCS_RDY bit of Processor Register TXCS is clear
    Decrement Timeout
    If Timeout is less than or equal to 0
      Exit Loop
    Endif
  Endwhile
  Move Save_TXCS to Processor Register TXCS
Endfor
Return
```

#1500-15-031.02.0

3.3.10.1.4.21.5 Limitations

Only 10 format items can be specified with each call to this unit, and the maximum length of the final formatted string must be less than 100 characters.

3.3.10.1.4.22 Read_Dir Unit

The Read_Dir Unit reads the directory for diskettes 0 and 1 from diskette 0. This directory is used by the File_Open Unit to determine file location and other information for the files that are on the diskettes.

3.3.10.1.4.22.1 Inputs

The following inputs are defined for the Read_Dir Unit:

- 1) Diskette - This input is read from diskette 0 starting at block 16. This input contains the directory information for both diskettes 0 and 1.

#1500-15-031.02.0

3.3.10.1.4.22.2 Outputs

The following outputs are produced by the Read_Dir Unit:

- 1) Disk_Dir - This output is written to global data and contains a copy of the diskette directory obtained from diskette 0.

3.3.10.1.4.22.3 Local Data

No local data is defined for the Read_Dir Unit.

3.3.10.1.4.22.4 Processing

Call ROM routine to read 1K directory at disk address 8K to
Disk_Dir structure
Return

3.3.10.1.4.22.5 Limitations

The maximum size of a directory is predefined to be 1024 bytes and cannot be exceeded.

3.3.10.1.4.23 Read_Process_List Unit

The Read_Process_List Unit reads the list of process from a file on disk and stores it in the Proc_List area that is declared to be global within the Local Boot component.

#1500-15-031.02.0

3.3.10.1.4.23.1 Inputs

The following input is used by the Read_Process_List Unit:

- 1) Proc_List File - This input is obtained from the "Proc_List" diskette file and contains a copy of the names of the files containing the processes (and their priorities) that the Read_Processes Unit is to load into the IGW. Each entry is separated by newlines and process are separated from priorities by spaces.

3.3.10.1.4.23.2 Outputs

The following output is produced by the Read_Process_List Unit:

- 1) Proc_List - This output is written to the Proc_List array and contains the list of process that are to be loaded. The entries in the list are each separated by a newline character and the end of the list is indicated by a Null character following a newline character. Names are separated from priorities by spaces.

3.3.10.1.4.23.3 Local Data

No local data is defined for the Read_Process_List Unit:

3.3.10.1.4.23.4 Processing

```
If result of File_Open("Proc_List") is less than 0
    Call Panic(error message)
Endif
If result of File_Read(address of Proc_List,
    bytes to read = 1024) is less than or equal to 0
    Call Panic(error message)
Endif
Add a Null character to the end of Proc_List
Return
```

#1500-15-031.02.0

3.3.10.1.4.23.5 Limitations

The maximum size of the process list input file that this unit will accept is 1024 bytes.

3.3.10.1.4.24 Read_Processes Unit

The Read_Processes Unit loads the processes specified in the list "Proc_List" that has been created by the Read_Process_List Unit. This involves placing the process text, data, bss, and stack area in physical memory and creating a process page table for them. System page table entries will also be added to reference the process page table.

3.3.10.1.4.24.1 Inputs

The following inputs are used by the Read_Processes Unit:

- 1) Proc_List - This input comes from the global data that has been loaded by the Read_Process_List Unit. This input contains a list of file names to load processes from as well as the priority of each of the processes.
- 2) Process Images From Diskette - This input consists of the binary images of the IGW processes that are to be loaded from diskette.
- 3) Free_Phys - This input is read from global data and contains the free physical memory address where the processes are to be loaded.
- 4) Free_Virt - This input is read from global data and

#1500-15-031.02.0

contains the free virtual memory address where the processes are to be loaded.

- 5) Sys_Pt - This input is read from global data and contains the physical address of the system page table that has been defined by the Reserve_SPT Unit.

3.3.10.1.4.24.2 Outputs

The following outputs are produced by the Read_Processes Unit:

- 1) Process_Header_List - This output is written to the ILA and contains the initialized process headers including PCBs for the processes that have been loaded into memory.
- 2) Processes In Memory - This output is written to the IGW main memory and contains the text, data, bss, and stack areas of the IGW processes that have been loaded.
- 3) Process Page Tables - This output is written to IGW memory and contains the process page tables for P0 and P1 address space for each process that is loaded by this unit.
- 4) System Page Table - This output is updated with the system page table entries required to reference the process page tables that have been created by this unit.
- 5) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 6) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 7) ILA - This output contains the value for the number of processes that have been loaded. This value is written to the Nproc field of the IGW Link Area.

3.3.10.1.4.24.3 Local Data

The following local data is defined for the Read_Processes Unit:

- 1) Process_Name_Pointer - This local data item is used to step through the Proc_List input to obtain each Current_Process_Entry item for the entries in Proc_List.
- 2) Current_Process_Entry - This local data item is used to store the entry in the Proc_List input that contains the filename and priority of the current process.
- 3) Current_Process_Name - The local data item is used to store the name the current process that is being loaded from diskette to main memory.
- 4) Current_Process_Priority - This piece of data is used to store the priority of the processes as they are read from diskette and loaded to the IGW.
- 5) Exec_Header - This local data item is a structure that is used to hold the header from the IGW processes that are loaded from diskette. The fields in this structure are all 32 bit values and are defined as follows:

A_Magic - This field contains the type of the executable image that is being loaded. Valid values for this field are:

- OMAGIC (0407) - Old impure format.
- NMAGIC (0410) - Read-only text.
- ZMAGIC (0413) - Demand load format.

A_Text - This field contains the size of the text segment in bytes.

A_Data - This field contains the size of the initialized data segment in bytes.

A_Bss - This field contains the size of the uninitialized data segment in bytes.

A_Syms - This field contains the size of the symbol field.

A_Entry - This field contains of the address of the entry point of the loaded executable image.

A_Trsize - This field contains the size of the text relocation area.

A_Drsize - This field contains the size of the data relocation area.

- 6) Proc_Phys - This local data item is used to hold the physical address of the loaded process text area.
- 7) Proc_Virt - This local data item is used to hold the virtual address of the loaded process text area.
- 8) Stack_Phys - This local data item is used to hold the physical address of the stack area.
- 9) Stack_Virt - This local data item is used to hold the virtual address of the stack area.
- 10) P0_Phys - This local data item contains the physical address of the P0 page table.
- 11) P0_Virt - This local data item contains the virtual address of the P0 page table.
- 12) P0_Len - This local data item contains the length of the P0 page table.
- 13) P1_Phys - This local data item contains the physical address of the P1 page table.
- 14) P1_Virt - This local data item contains the virtual address of the P1 page table.
- 15) P1_Len - This local data item contains the length of the P1 page table.
- 16) Nproc - This local data item contains the number of process that have been loaded into IGW memory.

#1500-15-031.02.0

3.3.10.1.4.24.4 Processing

Loop

```
Move Process_Name_Pointer to Current_Entry
Move address of next newline character in string pointed to
  by Process_Name_Pointer to Process_Name_Pointer
If no newline was found in the string pointed to by
  Process_Name_Pointer
  Exit Loop
If process header list is full
  Process error condition
Endif
Move Null character to character pointed to by
  Process_Name_Pointer
Increment Process_Name_Pointer
scanf(Current_Entry, "%s %d", Current_Process_Name,
  address of Current_Process_Priority)
If result of File_Open(Current_Process_Name) is less than 0
  Call Panic(error message)
Endif
If result of File_Read(address of Exec_Header,
  size of Exec_Header) isn't equal size of Exec_Header
  Call Panic(error message)
Endif
If A_Magic field of Exec_Header is ZMAGIC
  If result of File_Seek(Offset = 1024) is less than 0
    Call Panic(error message)
  Endif
Else if A_Magic field of Exec_Header isn't one of OMAGIC or NMAGIC
  Call Panic(error message)
Endif
If result of File_Read(Free_Phys,
  A_Text field of Exec_Header + A_Data field of Exec_Header)
  isn't equal (A_Text field of Exec_Header + A_Data field of
  Exec_Header)
  Call Panic(error message)
Endif
Move Free_Phys to Proc_Phys
Move 0 to Proc_Virt
Add A_Text field of Exec_Header to Free_Phys
Add A_Data field of Exec_Header to Free_Phys
For each address in bss area
  Clear memory referenced by Free_Phys
  Increment Free_Phys
Endfor
Adjust Free_Phys to point to page boundary if required
Move Free_Phys to Stack_Phys
Move stack virtual address to Stack_Virt
```

#1500-15-031.02.0

```
For Counter equals 1 to PROC_KERN_STACK_SIZE + PROC_USER_STACK_SIZE
  Clear memory referenced by Free_Phys
  Increment Free_Phys
Endfor
Advance Free_virt to start of next page if necessary
Move Free_Phys to P0_Phys
Move Free_Virt to P0_Virt
For each page in process text, data, and bss areas
  Call Add_To_PT(Page_Table = P0_Phys, Proc_Virt,
    Proc_Phys)
  Add PAGE_SIZE to Proc_Phys
  Add PAGE_SIZE to Proc_Virt
Endfor
Add length of P0 page table to Free_Phys
Add length of P0 page table to Free_Virt
Move P1 PT start physical address to P1_Phys
Move P1 PT start system virtual address to P1_Virt
For each page in stack area
  Call Add_To_PT(Page_Table = P1_Phys, Stack_Virt,
    Stack_Phys)
  Add PAGE_SIZE to Stack_Phys
  Add PAGE_SIZE to Stack_Virt
Endfor
Add length of P1 PT to Free_Phys
Add length of P1 PT to Free_Virt
Adjust Free_Phys to next page if necessary
Adjust Free_Virt to next page if necessary
Set Name field of process entry indexed by Nproc in
  Process_Header_List to name stored in Current_Process_Name
Set Priority field of process entry indexed by Nproc in
  Process_Header_List to priority stored in
  Current_Process_Priority
Set PCB_Address field of process entry indexed by Nproc in
  Process_Header_List to the physical address of the hardware PCB
Initialize kernel and user stack pointer in PCB
Initialize Processor_Status_Longword in PCB
Move P0_Virt to Program_Base_Register in the hardware PCB
Move P0_Len to Program_Length_Register in the hardware PCB
Move P1_Virt to Control_Base_Register in hardware PCB
Move P1_Len to Control_Length_Register in hardware PCB
For each page N in process page tables P0 and P1
  Call Add_To_PT(Page_Table = Sys_PT,
    Virt_addr = P0_Virt,
    Phys_addr = P0_Phys)
  Add PAGE_SIZE to P0_Phys
  Add PAGE_SIZE to P1_Virt
Endfor
Increment Nproc
Endloop
```

#1500-15-031.02.0

Move Nproc to Nproc field of ILA
Return

3.3.10.1.4.24.5 Limitations

The unit Read_Process_List must be called prior to this unit.

3.3.10.1.4.25 Reboot Unit

The Reboot Unit causes the IGW to perform a reboot.

3.3.10.1.4.25.1 Inputs

No inputs are defined for the Reboot Unit.

3.3.10.1.4.25.2 Outputs

The following output is produced by the Reboot Unit:

- 1) MicroVAX II Console Program Mailbox - This output is loaded with the value RB_REBOOT to cause the MicroVAX to perform a reboot.

#1500-15-031.02.0

3.3.10.1.4.25.3 Local Data

No local data is defined for the Reboot Unit.

3.3.10.1.4.25.4 Processing

Set MicroVAX II Console Program Mailbox to RB_REBOOT
Halt Processor

3.3.10.1.4.25.5 Limitations

No limitations are defined for this unit.

3.3.10.1.4.26 Relocate Unit

The Relocate Unit copies the software for the entire Sec_Boot TLC from the beginning of memory to the memory location RELOC. After this relocation control is transferred to the Main Unit of the Sec_Boot TLC.

#1500-15-031.02.0

3.3.10.1.4.26.1 Inputs

The following input is used by the Relocate Unit:

- 1) Original Sec_Boot Program Image - This input is the image in memory of the secondary boot program before relocation.

3.3.10.1.4.26.2 Outputs

The following output is produced by the Relocate Unit:

- 1) Relocated Sec_Boot Program Image - This output is a copy of the original Sec_Boot program that has been relocated to location RELOC in memory.

3.3.10.1.4.26.3 Local Data

No local data is defined for the Relocate Unit.

3.3.10.1.4.26.4 Processing

```
Move relocation address RELOC to stack pointer
For memory locations from the end of the data area to the
beginning of the relocation area
    Clear memory location
```

```
Endfor
```

```
Copy Sec_Boot image to relocation address RELOC
Transfer control to main() Unit in relocated image
```

#1500-15-031.02.0

3.3.10.1.4.26.5 Limitations

No limitations are defined for the Relocate Unit.

3.3.10.1.4.27 Reserve_SPT Unit

The Reserve_SPT Unit reserves a predefined number of pages following the SCB to contain the system page table. This is accomplished by setting the System Base Register (SBR) and System Length Register (SLR) to indicate the start and length of the system page table.

3.3.10.1.4.27.1 Inputs

The following inputs are defined for the Reserve_SPT Unit:

- 1) Free_Phys - This input is read from global data and contains the free physical memory address where the system page table is placed.

#1500-15-031.02.0

3.3.10.1.4.27.2 Outputs

The following outputs are produced by the Reserve_SPT Unit:

- 1) SBR - This output is written to the System Base Register, and contains the base address of the system page table (SPT_BASE).
- 2) SLR - This output is written to the System Length Register, and contains the length of the system page table in long words (SPT_LENGTH / 4).
- 3) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 4) Sys_PT - This output is written to global data and contains the physical address of the system page table.

3.3.10.1.4.27.3 Local Data

No local data is defined for the Reserve_SPT Unit.

3.3.10.1.4.27.4 Processing

```
Move Free_Phys to Sys_PT
Move Sys_PT to processor register SBR
Move SPT_LENGTH to processor register SLR
Add SPT_LENGTH * 4 to Free_Phys
Return
```


#1500-15-031.02.0

3.3.10.1.4.27.5 Limitations

This unit must be called immediately after the Load_SCB unit.

3.3.10.1.4.28 Size_Memory Unit

The Size_Memory Unit counts up the number of bytes of physical memory in the IGW.

3.3.10.1.4.28.1 Inputs

The following inputs are used by the Size_Memory Unit:

- 1) PFN_Map_Addr - This input is found at 0x48 (hex) plus the address stored in register R11 by the boot ROMs. This input contains the starting address of the PFN map.
- 2) PFN_Map_Size - This inputs is found at 0x44 (hex) plus the address stored in register R11 by the boot ROMs. This input contains the size of the PFN map.

#1500-15-031.02.0

3.3.10.1.4.28.2 Outputs

The following output is produced by the Size_Memory Unit:

Memory_Size - This output is written to the ILA to indicate the number of bytes of memory that is in the IGW.

3.3.10.1.4.28.3 Local Data

The following local data is defined for the Size_Memory Unit:

- 1) Good_Page_Counter - This data item is used to count the number of good pages of memory in the IGW.
- 2) PFN_Pointer - This data item is a pointer to the current byte that is being examined in the PFN map.
- 3) PFN_Counter - This data item is used to count through the PFN map while looking for good pages.

3.3.10.1.4.28.4 Processing

```
Clear Good_Page_Counter
Move PFN_Map_Addr to PFN_Pointer
Move PFN_Map_Size to PFN_Counter
While byte pointed to by PFN_Pointer equals 0xff (hex) and
  PFN_Counter is greater than 0
  Add 8 to Good_Page_Counter
  Increment PFN_Pointer
  Decrement PFN_Counter
Endwhile
Move 512 * Good_Page_Counter to Memory_Size field in ILA
Return
```

#1500-15-031.02.0

3.3.10.1.4.28.5 Limitations

The number of pages of memory is calculated in units of 8 pages at a time.

3.3.10.1.4.29 Start_ERTE Unit

The Start_ERTE Unit transfers control from the Sec_Boot TLC to the ERTE TLC.

3.3.10.1.4.29.1 Inputs

The following inputs are required by the Start_ERTE Unit:

- 1) Free_Phys - The starting physical address of the free memory area. This input is obtained from global data.
- 2) Free_Virt - The starting virtual address of the free memory area. This input is obtained from global data.
- 3) Istack_Virt - The system virtual address of the top of the interrupt stack.

#1500-15-031.02.0

3.3.10.1.4.29.2 Outputs

The following outputs are defined for the Start_ERTE Unit.

- 1) PO_PT - This output is written to the free memory area and contains the page table entry for the instruction that transfers control to ERTE.
- 2) POBR - This output is written to the P0 Base Register and contains the base address of the P0 page table used to switch to virtual addressing mode.
- 3) POLR - This output is written to the P0 Length Register and contains the length of the P0 page table used to switch to virtual addressing mode.
- 4) ISP - This output is written to the interrupt stack pointer and contains the starting address of the interrupt stack.

3.3.10.1.4.29.3 Local Data

No local data is defined for the Start_ERTE Unit.

3.3.10.1.4.29.4 Processing

Move Istack_Virt to ISP
Move Free_Virt - (physical address of LABEL_1 shifted right by 9 bit positions) to POBR
Move (physical address of LABEL_1 shifted right by 9 bit positions) + 1 to POLR
Call Add_To_PT(Page_Table = Free_Phys - (physical address of LABEL_1 shifted right by 9 bit positions),
Virt_Addr = Address of LABEL_1,
Phys_Addr = Address of LABEL_1)
Call Add_To_PT(Page_Table = Free_Phys - (physical address of LABEL_1 shifted right by 9 bit positions),
Virt_Addr = Address of LABEL_1 + 512,
Phys_Addr = Address of LABEL_1 + 512)
Clear Translate Buffer Invalidate All Register (TIBA)
Move 1 to Map Enable Register (MAPEN)

LABEL_1:

Transfer control to virtual address of start of ERTE

#1500-15-031.02.0

3.3.10.1.4.29.5 Limitations

No limitations are defined for this unit.

3.3.10.2 IGW Net Load Component

This component provides the Net Load boot operations for the IGW. The component sends requests for software to be downloaded to a known cooperating host on the Ethernet, and then receives and installs the software. The IGW software is loaded into IGW memory, and the IXIB software is loaded onto the IXIB board.

3.3.10.2.1 Net Load Component Architecture

The IGW Net Load Component is composed of the following units (Figure 3-11):

- 1) Calc_Mem_Size - This unit calculates the amount of IGW memory (in bytes).
- 2) Check_Dgram - This unit checks a received datagram to ensure it is from the correct host and contains no errors.
- 3) Check_IP - This unit checks the IP header of the received datagram for errors.
- 4) Check_UDP - This unit checks the UDP header of the received datagram for errors.
- 5) Chk_Sum - This unit adds a value to a ones complement check sum.

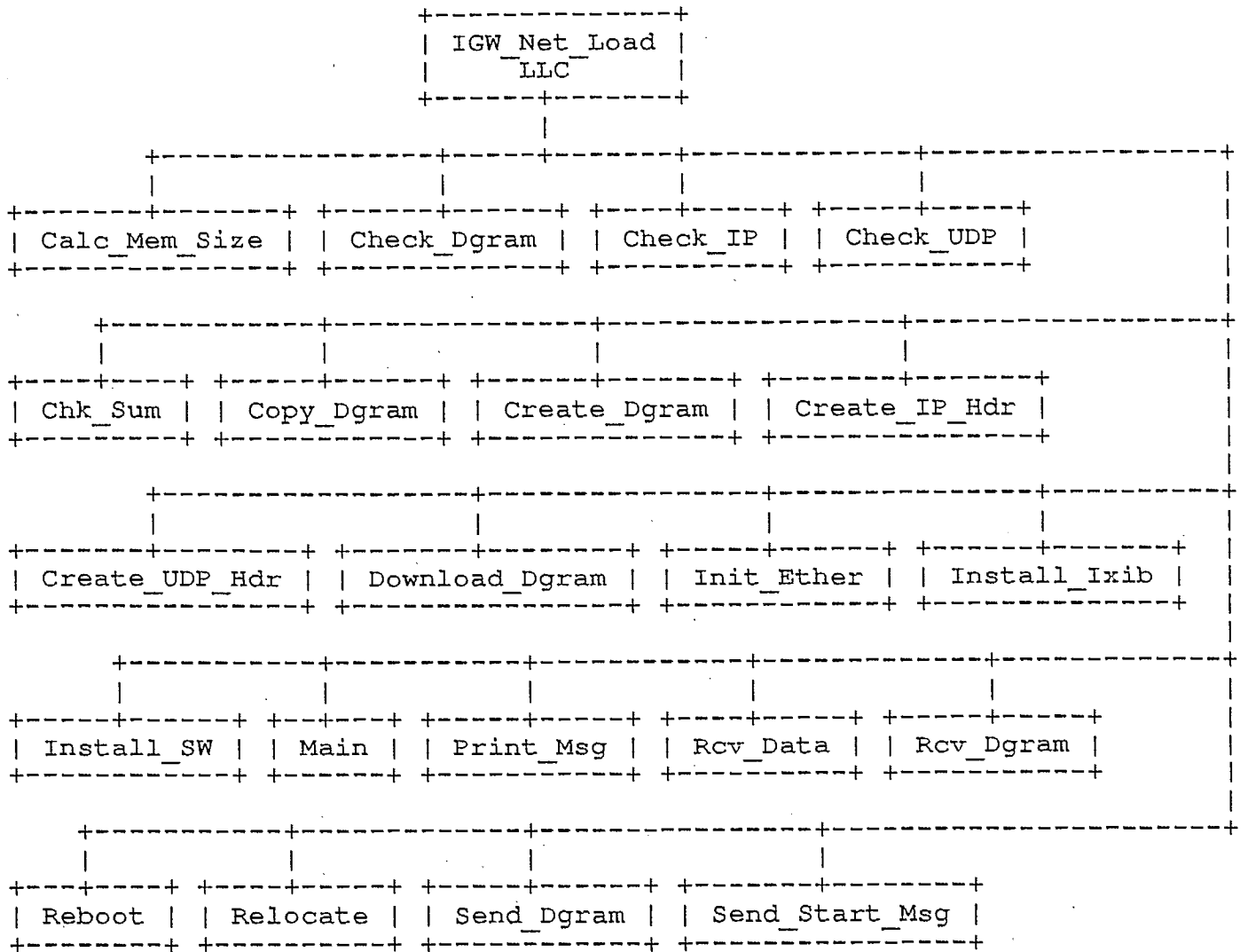


Figure 3-11

#1500-15-031.02.0

- 6) Copy_Dgram - This unit copies a the data of a datagram to IGW memory.
- 7) Create_Dgram - This unit creates a datagram to contain an outgoing message.
- 8) Create_IP_Hdr - This unit creates the IP header for an outgoing datagram.
- 9) Create_UDP_Hdr - This unit creates the UDP header for an outgoing datagram.
- 10) Download_Dgram - This unit downloads the data portion of a datagram to the IXIB board.
- 11) Init_Ether - This unit initializes the Ethernet hardware for receive and transmit operation without interrupts.
- 12) Install_Ixib - This unit controls the receiving and downloading of IXIB software from the cooperating host.
- 13) Install_SW - This unit controls the receiving and loading of IGW software.
- 14) Main - This unit is the starting unit of the component.
- 15) Print_Msg - This unit causes an error message to be displayed on the operator's console.
- 16) Rcv_Data - This unit begins and controls the process of requesting for software to be downloaded, and then receiving the software.
- 17) Rcv_Dgram - This unit receives a datagram from the Ethernet device.
- 18) Reboot - This unit causes the IGW to reboot when a boot failure is detected.
- 19) Relocate - This unit relocates the Net Load component to high memory in the IGW.
- 20) Send_Dgram - This unit controls the Ethernet hardware to send a datagram to the cooperating host.

#1500-15-031.02.0

- 21) Send_Start_Msg - This unit prepares a message to be sent to the cooperating host.

3.3.10.2.2 Global Data

The following constants are defined as global data within this TLC:

- 1) ZERO (0) - A constant to represent a null pointer (zero address).
- 2) SEND_IGW (1) - This constant represents the code used when requesting the cooperating host to download IGW software.
- 3) SEND_IXIB (2) - This constant represents the code used when requesting the cooperating host to download IGW software.
- 4) VALID_ADDRESS (8000 hex) - This constant is used to mark the address of an Ethernet BDL as valid (See the DEQNA User's Guide).
- 5) INITIALIZED (8000 hex) - This constant is used to mark an Ethernet BDL as initialized (See the DEQNA User's Guide).
- 6) END_MSG (2000 hex) - This constant is used to mark an Ethernet BDL as the last for the current packet being transmitted (See the DEQNA User's Guide).

#1500-15-031.02.0

3.3.10.2.3 IGW Net Load LLCs

No LLCs are defined for the Net Load Component.

3.3.10.2.4 IGW Net Load Units

The following sections contain the unit descriptions for all units comprising the IGW Net Load Component.

3.3.10.2.4.1 Calc_Memory_Size Unit

The Calc_Memory_Size unit calculates the size of IGW memory in bytes.

3.3.10.2.4.1.1 Inputs

The following input is used by the unit:

- 1) Boot_Info_Pointer - This thirty-two bit pointer references a table of boot information left by the Boot ROMS of the Micro-VAX.
- 2) Boot_Info - This is a table of information prepared by the Micro-VAX boot ROMS which specifies information useful to the boot procedure. The information used by this unit is:
 - 1) Page_Map - This table is at offset PAGE_MAP (48 hex) from the start of Boot_Info. The page map consists of a list of bytes, one for each set of 8 memory pages (or portion thereof). Each bit in each byte represents one page. If the bit is 1, then the page is

#1500-15-031.02.0

good, otherwise the page is bad.

- 2) Page_Map_Size - The size of Page_Map (in bytes) is located at offset MAP_SIZE (44 hex) from the start of Boot_Info.

3.3.10.2.4.1.1 Outputs

The following outputs are produced by the unit:

- 1) Memory_Size - The size in bytes of good memory from address zero up to, but excluding, the first bad page as indicated by the page map.

3.3.10.2.4.1.3 Local Data

The following local data is defined for this unit:

- 1) Page_Map_Pointer - This thirty-two bit pointer is used to step through the page map table.
- 2) Good_Count - A count of the number of good pages in the Page_Map.

3.3.10.2.4.1.4 Processing

Set Good_Count to 0

Set Page_Map_Pointer to Boot_Info Page_Map start

While (contents of byte referenced by Page_Map_Pointer is all ones)

 Increment Good_Count by one

 Increment Page_Map_Pointer by one

Endwhile

Set Memory_Size to (Good_Count * 8 pages per count * 512 bytes per page)

Return(Memory_Size)

#1500-15-031.02.0

3.3.10.2.4.1.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.2 Check_Dgram Unit

The Check_Dgram unit examines a datagram received from the cooperating host and verifies that it is correct and complete. The unit verifies the IP and UDP headers for the received datagram according to the protocol specification. The unit also ensures that the source and destination addresses are correct.

3.3.10.2.4.2.1 Inputs

The following inputs are used by the unit:

- 1) Buffer - This input parameter is a thirty-two bit pointer to the buffer containing the received datagram.

#1500-15-031.02.0

3.3.10.2.4.2.1 Outputs

The following outputs are produced by the unit:

- 1) `Data_Address` - This return parameter is a thirty-two bit pointer to the address in the Buffer where the datagram data begins (the byte immediately following the UDP header). This parameter is returned as zero if an error is detected in IP or UDP headers.

3.3.10.2.4.2.3 Local Data

The following local data is defined for the unit:

- 1) `UDP_Start` - This item is a thirty-two bit pointer to the start of the UDP header. This item is returned from the `Check_UDP` unit. The value is set to zero if the UDP header contains an error.

3.3.10.2.4.2.4 Processing

```
UDP_Start = Check_IP(Buffer)
If (UDP_Start != ZERO)
    Data_Address = Check_UDP(UDP_Start)
Else
    Return(UDP_Start)
Endif
Return(Data_Address)
```

#1500-15-031.02.0

3.3.10.2.4.2.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.3 Check_IP Unit

The Check_IP unit checks an IP datagram header for the datagram received.

3.3.10.2.4.3.1 Inputs

The following input is used by the unit:

- 1) Buffer - this thirty-two bit pointer parameter is a pointer to the start of the received datagram.

3.3.10.2.4.3.1 Outputs

The following is output by the unit:

- 1) UDP_Start - This thirty-two bit return parameter is a pointer to the start of the UDP header in the datagram.

#1500-15-031.02.0

3.3.10.2.4.3.3 Local Data

The following local is defined for the unit:

1) IP_Header - This structure contains the IP Header required for datagram transmission. It consists of the following fields:

- 1) Version - 4 bits contain the IP version number (4).
- 2) IHL - 4 bits containing the IP header length in 32 bit words. This is held constant at 5 for this application because no options are used.
- 3) Service_Type - 8 bits containing the type of service requested from IP. This field is held at 0, representing routine or normal service.
- 4) Total_Length - 16 bits containing the total length of the datagram, including header and data. It is the sum of IHL field and the Data_Size parameter.
- 5) Time_To_Live - 8 bits which contain the number of seconds the datagram is allowed to live before it is declared undeliverable. It is set to 10 for all transmitted datagrams, which is more than adequate for the application.
- 6) Protocol - 8 bits containing the protocol number for the datagram. The number is 17 for the UDP protocol.
- 7) Header_Checksum - 16 bits containing the checksum for the IP header of the datagram.
- 8) Source_Address - 32 bits containing the source address of the datagram. For this unit, this will be the Internet address of the IGW.
- 9) Destination_Address - 32 bits containing the destination address of the datagram. For this unit, this will be the Internet address

#1500-15-031.02.0

of the cooperating host.

- 2) Csum - This 16 bit integer is used to accumulate the header checksum for the datagram header.
- 3) Error - This 32 bit integer is used to indicate that an error has been detected.

3.3.10.2.4.3.4 Processing

```
Set Error to FALSE
If (Version field of IP_Header != 4)
    Set Error to TRUE
Endif
If (Identification field of IP_Header != 0)
    Set Error to TRUE
Endif
If (Flags field of IP_Header != 0)
    Set Error to TRUE
Endif
If (Fragment_Offset field of IP_Header != 0)
    Set Error to TRUE
Endif
If (Protocol field of IP_Header != 17)
    Set Error to TRUE
Endif
If (Destination_Address field of IP_Header != Internet address of IGW on
    the Ethernet)
    Set Error to TRUE
Endif
If (Source_Address field of IP_Header != Internet address of the
    cooperating host)
    Set Error to TRUE
Endif
Set Csum to 0
For each 16 bit word in IP_Header
    Csum = Chk_Sum(word, Csum)
Endfor
If (Csum != 0 and Csum != -1)
    Set Error to TRUE
Endif
If (Error = TRUE)
    return(0)
Else
    Return(Buffer + IHL field of datagram)
Endif
```

#1500-15-031.02.0

3.3.10.2.4.3.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.4 Check_UDP_Hdr Unit

The Check_UDP_Hdr unit checks an UDP datagram header for the datagram to be transmitted.

3.3.10.2.4.4.1 Inputs

The following input is used by the unit:

- 1) UDP_Hdr - This thirty-two bit parameter is a pointer to the UDP header in the datagram.
- 2) Buffer - This thirty-two bit parameter is a pointer to the start of the datagram, which is assumed to be the start of the IP header.

#1500-15-031.02.0

3.3.10.2.4.4.1 Outputs

The following is output by the unit:

- 1) Data_Start - This thirty-two bit return parameter is a pointer to the the start of the data in the datagram.

3.3.10.2.4.4.3 Local Data

The following local is defined for the unit:

- 1) UDP_Header - This structure contains the UDP Header required for datagram transmission. It consists of the following fields:
 - 1) Source_Port - 16 bits containing the number of the IGW port number used for this application. This number is always zero.
 - 2) Destination_Port - 16 bits containing the number of the port number used by the cooperating host for this application. This number is determined by the host administrator.
 - 3) Length - 16 bits containing the length in bytes of the UDP datagram, including UDP header and data.
 - 4) Check_Sum - 16 bits containing the checksum for the datagram. The checksum is the ones complement of the ones complement sum of all the 16 bit words in the Message, the UDP header (with the Check_Sum field at zero), the IP header source and destination addresses, the IP header protocol field (one byte with a zero byte prepended to make a 16 bit value), and the IP header total length field.
- 2) Csum - This 16 bit integer is used to accumulate the header checksum for the datagram header.

3.3.10.2.4.4.4 Processing

#1500-15-031.02.0

```
Set Csum to 0
Set Data_Start to UDP_Hdr + 8
If (Check_Sum field of UDP_Hdr != 0)
  Set Check_Sum field of UDP_Hdr to 0
  For each 16 bit word in UDP_Header
    Csum = Chk_Sum(word, Csum)
  Endfor
  For each 16 bit word in Data_Start
    Csum = Chk_Sum(word, Csum)
  Endfor
  For each 16 bit word in source address field of IP_Hdr
    Csum = Chk_Sum(word, Csum)
  Endfor
  For each 16 bit word in destination address field of IP_Hdr
    Csum = Chk_Sum(word, Csum)
  Endfor
  Csum = Chk_Sum( protocol field of IP_Hdr prepended with a zero
    byte, Csum)
  Csum = Chk_Sum( Total_Length field of IP_Hdr, Csum)
Endif

If (Csum != 0 and Csum != -1)
  Return(0)
Else
  Return(Data_Start)
Endif
```

3.3.10.2.4.4.5 Limitations

There are no limitations defined for this unit.

#1500-15-031.02.0

3.3.10.2.4.5 Chk_Sum Unit

The Chk_Sum unit adds a value to a checksum. The sum is the one's complement of the 16 one's complement sum of 16 bit words.

3.3.10.2.4.5.1 Inputs

The following input is used by the unit:

- 1) Word - This 16 bit input parameter is the 16 bit word to be added to the checksum.
- 2) Sum - This 16 bit input parameter is the current value of the checksum.

3.3.10.2.4.5.1 Outputs

The following output is produced by the unit:

- 1) New_Sum - This 16 bit return parameter is the new value of the checksum.

#1500-15-031.02.0

3.3.10.2.4.5.3 Local Data

There is no local data defined for this unit.

3.3.10.2.4.5.4 Processing

```
Set Sum to the ones complement of Sum
New_Sum = Sum + Word using 16 bit arithmetic
If a carry occurred
    Add 1 to New_Sum
Endif
Set New_Sum to the ones complement of New_Sum
Return(New_Sum)
```

3.3.10.2.4.5.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.6 Copy_Dgram Unit

The Copy_Dgram unit copies a datagram received from the cooperating host into IGW memory.

#1500-15-031.02.0

3.3.10.2.4.6.1 Inputs

The following input is used by the unit:

- 1) Dgram - This thirty-two bit parameter is a pointer to the data portion of the datagram. The datagram resides in a buffer global to this unit.
- 2) Page - This thirty-two bit integer parameter indicates which page of IGW physical memory to copy the datagram into.

3.3.10.2.4.6.1 Outputs

The following output is produced by the unit:

- 1) Memory_Page - The IGW physical page of memory indicated by Page is written with the datagram data.

3.3.10.2.4.6.3 Local Data

The following local data is defined for this unit

- 1) Memory_Page_Addr - The data in the datagram is copied into the IGW physical memory page specified by Page. The starting address of the memory page is calculated and stored in this local item.

#1500-15-031.02.0

3.3.10.2.4.6.4 Processing

```
Calculate Memory_Page_Addr = Page * 512
For each of the 512 bytes in Dgram
  Copy byte(i) in Dgram to Memory_Page_Addr + i
Endfor
```

3.3.10.2.4.6.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.7 Create_Dgram Unit

The Create_Dgram unit builds the IP datagram header and the UDP datagram header for a message to be sent to the cooperating hosts.

3.3.10.2.4.7.1 Inputs

The following input is used by the unit:

- 1) Message - This thirty-two bit parameter is a pointer to the message to be sent to the cooperating host.

#1500-15-031.02.0

3.3.10.2.4.7.1 Outputs

The following are output by the unit:

- 1) Dgram - This thirty-two bit return parameter is a pointer to the datagram structure prepared by the unit.

3.3.10.2.4.7.3 Local Data

The following local data is defined for the unit:

- 1) Datagram - This item pulls together the headers and data of the datagram so that they can be referenced in a single structure. The structure has the following fields:
 - 1) IP_Hdr - 32 bit pointer to the IP datagram header.
 - 2) UDP_Hdr - 32 bit pointer to the UDP datagram header.
 - 3) Dgram_Msg - 32 bit pointer to the datagram data.
- 2) Msg_Size - This item is a thirty-two bit integer containing the size (in bytes) of the message to be sent. This value is always 8 because the message consists of two 32 bit words.

#1500-15-031.02.0

3.3.10.2.4.7.4 Processing

```
IP_Hdr field of Datagram = Create_IP_Hdr(Msg_Size)
UDP_Hdr field of Datagram = Create_UDP_Hdr(IP_Hdr field of
  Datagram, Message)
Set Dgram_Msg field of Datagram to Message
Set Dgram to the address of Datagram
Return(Dgram)
```

3.3.10.2.4.7.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.8 Create_Ip_Hdr Unit

The Create_Ip_Hdr unit creates an IP datagram header for the datagram to be transmitted.

3.3.10.2.4.8.1 Inputs

The following input is used by the unit:

- 1) Data_Size - this thirty-two bit integer parameter is the length of the datagram data field in bytes.

#1500-15-031.02.0

3.3.10.2.4.8.1 Outputs

The following is output by the unit:

- 1) Header - This thirty-two bit return parameter is a pointer to the the header generated by the unit.

3.3.10.2.4.8.3 Local Data

The following local is defined for the unit:

- 1) IP_Header - This structure contains the IP Header required for datagram transmission. It consists of the following fields:
 - 1) Version - 4 bits contain the IP version number (4).
 - 2) IHL - 4 bits containing the IP header length in 32 bit words. This is held constant at 5 for this application because no options are used.
 - 3) Service_Type - 8 bits containing the type of service requested from IP. This field is held at 0, representing routine or normal service.
 - 4) Total_Length - 16 bits containing the total length of the datagram, including header and data. It is the sum of IHL field and the Data_Size parameter.
 - 5) Time_To_Live - 8 bits which contain the number of seconds the datagram is allowed to live before it is declared undeliverable. It is set to 10 for all transmitted datagrams, which is more than adequate for the application.
 - 6) Protocol - 8 bits containing the protocol number for the datagram. The number is 17

for the UDP protocol.

- 7) Header_Checksum - 16 bits containing the checksum for the IP header of the datagram.
 - 8) Source_Address - 32 bits containing the source address of the datagram. For this unit, this will be the Internet address of the IGW.
 - 9) Destination_Address - 32 bits containing the destination address of the datagram. For this unit, this will be the Internet address of the cooperating host.
- 2) Csum - This 16 bit unsigned integer is used to accumulate the header checksum for the datagram header.

3.3.10.2.4.8.4 Processing

```
Set Version field of IP_Header to 4
Set IHL field of IP_Header to 5
Set Service_Type field of IP_Header to 0
Set Total_Length field of IP_Header to IHL + Data_Size
Set Identification field of IP_Header to 0
Set Flags field of IP_Header to 0
Set Fragment_Offset field of IP_Header to 0
Set Time_To_Live field of IP_Header to 10
Set Protocol field of IP_Header to 17
Set Source_Address field of IP_Header to Internet address of IGW on
the Ethernet
Set Destination_Address field of IP_Header to Internet address of the
cooperating host
Set Header_Checksum field of IP_Header to 0
Set Csum to 0
For each 16 bit word in IP_Header
    Csum = Chk_Sum(word, Csum)
Endfor
Set Header_Checksum field of IP_Header to Csum

Set Header to the address of IP_Header
Return(Header)
```

#1500-15-031.02.0

3.3.10.2.4.8.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.9 Create_UDP_Hdr Unit

The Create_UDP_Hdr unit creates an UDP datagram header for the datagram to be transmitted.

3.3.10.2.4.9.1 Inputs

The following input is used by the unit:

- 1) Data_Size - this thirty-two bit integer parameter is the length of the datagram data field in bytes.
- 2) Message - This thirty-two bit parameter is a pointer to the message to be placed in a UDP datagram.
- 3) IP_Hdr - This thirty-two bit parameter is a pointer to the IP header for the datagram, which is assumed to be 5 thirty-two bit words long.

#1500-15-031.02.0

3.3.10.2.4.9.1 Outputs

The following is output by the unit:

- 1) Header - This thirty-two bit return parameter is a pointer to the the header generated by the unit.

3.3.10.2.4.9.3 Local Data

The following local is defined for the unit:

- 1) UDP_Header - This structure contains the UDP Header required for datagram transmission. It consists of the following fields:
 - 1) Source_Port - 16 bits containing the number of the IGW port number used for this application. This number is always zero.
 - 2) Destination_Port - 16 bits containing the number of the port number used by the cooperating host for this application. This number is determined by the host administrator.
 - 3) Length - 16 bits containing the length in bytes of the UDP datagram, including UDP header and data.
 - 4) Check_Sum - 16 bits containing the checksum for the datagram. The checksum is the ones complement of the ones complement sum of all the 16 bit words in the Message, the UDP header (with the Check_Sum field at zero), the IP header source and destination addresses, the IP header protocol field (one byte with a zero byte prepended to make a 16 bit value), and the IP header total length field.
- 2) Csum - This 16 bit unsigned integer is used to accumulate the header checksum for the datagram header.

#1500-15-031.02.0

3.3.10.2.4.9.4 Processing

```
Set Source_Port field of UDP_Hdr to 0
Set Destination_Port field of UDP_Hdr to DEST_PORT
Set the Length field of UDP_Hdr to 8 + Data_Size

Set Check_Sum field of UDP_Header to 0
Set Csum to 0
For each 16 bit word in UDP_Header
    Csum = Chk_Sum(word, Csum)
Endfor
For each 16 bit word in Message
    Csum = Chk_Sum(word, Csum)
Endfor
For each 16 bit word in source address field of IP_Hdr
    Csum = Chk_Sum(word, Csum)
Endfor
For each 16 bit word in destination address field of IP_Hdr
    Csum = Chk_Sum(word, Csum)
Endfor
Csum = Chk_Sum(protocol field of IP_Hdr prepended with a zero
    byte, Csum)
Csum = Chk_Sum(Total_Length field of IP_Hdr, Csum)
If (Csum = 0)
    Set Csum to the one's complement of 0
Endif
Set Check_Sum field of IP_Header to Csum

Set Header to the address of UDP_Header
Return(Header)
```

#1500-15-031.02.0

3.3.10.2.4.9.5 Limitations

The IP header for the datagram must be created before this unit is called. The IP header length is assumed to be 5 thirty-two bit words in length.

3.3.10.2.4.10 Download_Dgram Unit

The Download_Dgram unit copies a datagram received from the cooperating host into IXIB board.

3.3.10.2.4.10.1 Inputs

The following input is used by the unit:

- 1) Dgram - This thirty-two bit parameter is a pointer to the data portion of the datagram. The datagram resides in a buffer global to this unit.
- 2) IXIB_FIFO - This item is the IXIB device port used by the IXIB as a FIFO queue for transferring data to the IXIB.

#1500-15-031.02.0

3.3.10.2.4.10.2 Outputs

The following output is produced by the unit:

- 1) IXIB_Data - The IXIB is loaded with the data in the datagram.

3.3.10.2.4.10.3 Local Data

No local data is defined for this unit.

3.3.10.2.4.10.4 Processing

For each of the 512 bytes in Dgram

Write the byte in Dgram to IXIB_FIFO register of each IXIB device
Endfor

3.3.10.2.4.10.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.11 Init_Ether Unit

The Init_Ether unit initializes the DEQNA Ethernet interface for operation without interrupts.

#1500-15-031.02.0

3.3.10.2.4.11.1 Inputs

No inputs are used by the unit.

3.3.10.2.4.11.1 Outputs

The following outputs are used by the unit:

- 1) Ether_CSR - This device register is used to provide control information to the DEQNA interface.

3.3.10.2.4.11.3 Local Data

The constant SOFTWARE_RESET (2) is the only local data defined for this unit. This constant is used to create a software reset condition on the DEQNA interface board, which will reset the board into the desired state for use without interrupts.

3.3.10.2.4.11.4 Processing

Write SOFTWARE_RESET to Ether_CSR

#1500-15-031.02.0

3.3.10.2.4.11.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.12 Install_IXIB Unit

The Install_IXIB unit receives IXIB software from the cooperating host and loads the software down to IXIB board.

3.3.10.2.4.12.1 Inputs

No input is used by the unit.

3.3.10.2.4.12.1 Outputs

The unit produces the following output:

- 1) IXIB Software - The IXIB Software collected by the unit is downloaded to the IXIB board.

#1500-15-031.02.0

3.3.10.2.4.12.3 Local Data

The following local data is used by the unit:

- 1) Count - This thirty-two bit integer contains the count of received IXIB software datagrams.
- 2) Time_Out - This item is a thirty-two bit word which contains the maximum time to wait for the next datagram to arrive. The value of this item is 30 seconds.
- 3) Dgram_Buffer - This data item is an area of contiguous memory available to the unit Rcv_Dgram to place a received datagram into.
- 4) Message - This item is a thirty-two bit pointer to the start of the message contained in the received datagram. The message consists of:
 - 1) Message_Type - A eight bit byte containing the type of message. Message types are:
 - A) DATA (1) - The message contains download software.
 - B) END (0) - The message is the last message of the downloading process.
 - 2) Message_Data - 512 eight bit bytes of download data.
- 5) Status - This thirty-two bit word is used to receive status returned by called units.

#1500-15-031.02.0

3.3.10.2.4.12.4 Processing

Set Count to zero

Loop

Status = Rcv_Dgram(Buffer, Time_Out)

If (Status = ERROR)

return(ERROR)

Endif

Message = Check_Dgram(Buffer)

If (Message = ZERO)

return(ERROR)

If (Message_Type field of Message != END)

Increment Count by 1

Call Download_Dgram(Message)

Endif

While (Message_Type field of Message != END)

If (Count != End_Count field in Message)

return(ERROR)

Endif

return(NOERROR)

3.3.10.2.4.12.5 Limitations

There are no limitations defined for this unit.

#1500-15-031.02.0

3.3.10.2.4.13 Install_SW Unit

The Install_SW unit receives IGW software from the cooperating host and loads the software into IGW memory.

3.3.10.2.4.13.1 Inputs

No input is used by the unit.

3.3.10.2.4.13.2 Outputs

The unit produces the following output:

- 1) IGW Software - The IGW Software collected by the unit is written to the IGW memory.
- 2) Special_Registers - A global variable containing:
 - SCBB - SCB base register
 - ISP - interrupt stock pointer
 - SBR - system base register
 - SLR - system length register
 - ERTE_VIRT - ERTE starting virtual address
- 3) Free_Phys - start at free physical memory after software is loaded

#1500-15-031.02.0

3.3.10.2.4.13.3 Local Data

The following local data is used by the unit:

- 1) Page - This thirty-two bit integer contains the memory page number that the next datagram of the IGW software will be written to.
- 2) Count - This thirty-two bit integer contains the count of received IGW software datagrams.
- 3) Time_Out - This item is a thirty-two bit word which contains the maximum time to wait for the next datagram to arrive. The value of this item is 30 seconds.
- 4) Dgram_Buffer - This data item is an area of contiguous memory into available to the unit Rcv_Dgram to place a received datagram into.
- 5) Message - This item is a thirty-two bit pointer to the start of the message contained in the received datagram. The message consists of:
 - 1) Message_Type - A thirty-two bit word containing the type of message. Message types are:
 - A) DATA (1) - The message contains download software.
 - B) END (0) - The message is the last message of the downloading process.
 - 2) Message_Data - 512 eight bit bytes of download data.
- 6) Status - This thirty-two bit word is used to receive status returned by called units.

#1500-15-031.02.0

3.3.10.2.4.13.4 Processing

Set Page to zero
Set Count to zero

Loop

Status = Rcv_Dgram(Buffer, Time_Out)
If (Status = ERROR)
 return(ERROR)

Endif

Message = Check_Dgram(Buffer)

If (Message = ZERO)
 return(ERROR)

If (Message_Type field of Message != END)
 Increment Count by 1
 Call Copy_Data(Message, Page)
 Increment Page by 1

Endif

While (Message_Type field of Message != END)

If (Count != End_Count field in Message)
 return(ERROR)

Endif

For (each Special_Register field in Message)

 Copy the field to the corresponding special register global variable

Endfor

See Free_Phys to Page*PAGE_SIZE
return(NOERROR)

3.3.10.2.4.13.5 Limitations

There are no limitations defined for this unit.

#1500-15-031.02.0

3.3.10.2.4.14 Main Unit

The Main unit of the IGW Net Load component is the unit which first receives control from the IGW Boot ROMS. The unit then directs the process of loading the IGW and IXIB software and data, and then transfers control to the IGW operating software.

3.3.10.2.4.14.1 Inputs

The following input is used by the unit:

- 1) Boot_Info_Pointer - This input is supplied by the Micro-VAX boot ROMS in register R11. It is a pointer to an area of memory where boot information is stored.
- 2) Special_Registers - This input is a global table containing the values for the Micro-VAX internal registers.

3.3.10.2.4.14.2 Outputs

The unit produces the following outputs:

- 1) VAX Registers - The unit will copy the values from the global Special_Registers input to the corresponding VAX internal registers.

#1500-15-031.02.0

3.3.10.2.4.14.3 Local Data

No local data is defined for this unit.

3.3.10.2.4.14.4 Processing

Call Calc_Memory_Size(Boot_Info_Pointer)

Call Relocate()

Call Receive_Data()

Copy fields in Special_Registers input to corresponding VAX internal registers.

Set up a local page table to map the memory containing the "jump to ERTE" instruction into its physical memory location

Set up POBR and POLR internal registers to select the page table just created.

Set the MAPEN internal register to turn on the VAX memory management.

Jump to the start of the IGW ERTE TLC.

3.3.10.2.4.14.5 Limitations

No limitations are defined for this unit

3.3.10.2.4.15 Print_Msg Unit

The Print_Msg unit prints the string passed to it as a parameter to be displayed on the IGW console.

#1500-15-031.02.0

3.3.10.2.4.15.1 Inputs

The following input is used by the unit:

- 1) Message - This thirty-two bit parameter is a pointer to the start of the message to be printed, which is global to this unit.
- 2) Transmit_CSR - This input/output device register contains the status and control information for the console device transmitter.

3.3.10.2.4.15.1 Outputs

The following are output by this unit:

- 1) Transmit_Data - This device register is the output data register for the console device. Characters of the message are written to this register to be displayed on the console.
- 2) Transmit_CSR - This input/output device register contains the status and control information for the console device transmitter. It is written to set up the transmitter for writing characters to the console.

#1500-15-031.02.0

3.3.10.2.4.15.3 Local Data

There is no local data defined for this unit.

3.3.10.2.4.15.4 Processing

Write the Transmit_CSR to set up the transmitter for writing characters without generating interrupts.

For each character in Message

 Write the character to Transmit_Data

 Loop

 Test Transmit_CSR

 While (Transmit_CSR show output is not completed)

Endfor

3.3.10.2.4.15.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.16 Receive_Data Unit

The Receive_Data unit controls the activities and procedures used to receive software and data from the cooperating host and to load it into the IGW memory or the IXIB.

#1500-15-031.02.0

3.3.10.2.4.16.1 Inputs

There are no inputs defined for this unit.

3.3.10.2.4.16.1 Outputs

There are no outputs defined for this unit.

3.3.10.2.4.16.3 Local Data

The following local data is defined for the unit:

- 1) Status - Returned status from called units.

3.3.10.2.4.16.4 Processing

Call Init_Ethernet()

```
Status = Send_Message(SEND_IGW)
If Status indicates an error occurred
    Call Reboot()
Endif
```

```
Status = Install_IGW_Software()
If Status indicates an error occurred
    Call Reboot()
Endif
```

```
Status = Send_Message(SEND_IXIB)
If Status indicates an error occurred
    Call Reboot()
Endif
```

```
Status = Install_IXIB_Software()
If Status indicates an error occurred
    Call Reboot()
Endif
```

#1500-15-031.02.0

3.3.10.2.4.16.5 Limitations

No limitations are defined for this unit.

3.3.10.2.4.17 Recv_Dgram Unit

The Recv_Dgram unit manipulates the Ethernet hardware to allow the receipt of an Ethernet packet, which is expected to contain an IP datagram. The unit implements a time-out so that the attempt to receive a datagram can be aborted if no datagram arrives.

3.3.10.2.4.17.1 Inputs

The following input is used by the unit:

- 1) Buffer_Pointer - This thirty-two bit parameter is a pointer to the input buffer supplied by the calling unit. The received datagram will be placed in this buffer, less the Ethernet header.
- 2) Time_Out - This thirty-two bit unsigned integer is a value used to determine how long to wait for an incoming Ethernet packet before assuming that an error has occurred or no packet is coming.
- 3) Ether_CSR - This input is the Control and Status register of the DEQNA hardware. It supplies status information when read.

#1500-15-031.02.0

3.3.10.2.4.17.1 Outputs

The following are output by this unit:

- 1) Buffer - The received datagram is loaded into the buffer pointed at by Buffer_Pointer. The datagram does not include the Ethernet header.
- 2) Status - This thirty-two bit return parameter indicates whether a successful receive operation occurred. If the receive was successful, then NOERROR is returned, otherwise ERROR is returned.
- 3) Ether_CSR - This input is the Control and Status register of the DEQNA hardware. Control information is passed to the register when it is written.
- 4) Rcv_BDL_Reg - This DEQNA device register is used to load the address of the Receive_BDL into the DEQNA to begin a receive operation.

3.3.10.2.4.17.3 Local Data

The following local data is defined for this unit:

- 1) Receive_BDL - This global item is a list of Buffer Descriptors for DEQNA receive operations. The buffer descriptors are predefined by the Init_Ether unit.

#1500-15-031.02.0

3.3.10.2.4.17.4 Processing

Write Buffer_Pointer into Address_Bits field of second BDL in Receive_BDL

/* first BDL is for Ethernet header */

Set Status to NOERROR

Write address of Receive_BDL into Rcv_BDL_Reg

/* This starts DEQNA receive operation */

Clear Ether_CSR Receive Interrupt Request bit

Loop

If (Ether_CSR Receive Interrupt Request bit is set)

Examine Receive Status Word 1 of second BDL in Receive_BDL

If the ERROR/USED bit of the status word is set

Set Status to ERROR

Endif

Else

Decrement Time_Out

If (Time_Out = 0)

Set Status to ERROR

Endif

Endif

While (Timer != 0)

Return(Status)

3.3.10.2.4.17.5 Limitations

This unit does not attempt to distinguish between types of receive errors. Also, the Time_Out defines a loop count which specifies how many times a loop must be executed before a time-out occurs. Because the time-out should be several seconds, this value must be very large.

#1500-15-031.02.0

3.3.10.2.4.18 Reboot Unit

The Reboot unit issues a message to the console, and then causes the IGW to reboot itself.

3.3.10.2.4.18.1 Inputs

There are no inputs to the unit.

3.3.10.2.4.18.1 Outputs

There are no outputs from the unit.

3.3.10.2.4.18.3 Local Data

There is no local data defined for the unit.

- 1) Halt_Control - This item is a thirty-two bit pointer to the 16 bit Q-Bus register used to direct the operation of the Micro-VAX processor when a Halt occurs. The value of this item is 200B801C(Hex)
- 2) REBOOT - This constant defines the value of Halt_Control to reboot the machine when a halt occurs. The value is 23(Hex).

#1500-15-031.02.0

3.3.10.2.4.18.4 Processing

Call Print_Msg("Boot Failure - IGW Rebooting")
Write REBOOT into the Halt_Control
Execute a Halt instruction

3.3.10.2.4.18.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.19 Relocate Unit

The Relocate unit copies the IGW Net Load boot program from low memory to the top of memory.

3.3.10.2.4.19.1 Inputs

The following input is used by the unit:

- 1) Memory_Size - This 32 bit integer contains the amount of memory in bytes available in the IGW.

#1500-15-031.02.0

3.3.10.2.4.19.1 Outputs

The following output is produced by the unit:

- 1) Relocated_Code - The program code is relocated to the top of IGW memory.

3.3.10.2.4.19.3 Local Data

The following local data is defined for the unit:

- 1) Program_Start - This 32 bit item is the address of the start of the program in memory. This item is determined from a Start symbol defined at compile time.
- 2) Program_End - This 32 bit item is the address of the end of the program in memory. This item is determined from a End symbol defined at compile time.
- 3) Program_Size - This 32 bit item is the size of the program in memory.

3.3.10.2.4.19.4 Processing

Set Program_Start to address of Start symbol

Set Program_End to address of End symbol

Set Program_Size to Program_End - Program_Start

Copy program code from its current start to Memory_Size - Program_Size

Transfer control to relocated code.

#1500-15-031.02.0

3.3.10.2.4.19.5 Limitations

There are no limitations defined for this unit.

3.3.10.2.4.20 Send_Dgram Unit

The Send_Dgram unit takes a datagram with IP and UDP headers and adds an Ethernet header, and then outputs the datagram to the the Ethernet.

3.3.10.2.4.20.1 Inputs

The following input is used by the unit:

- 1) Transmit_BDL - This global item is a list of Buffer Descriptors for DEQNA receive operations. The buffer descriptors are predefined by the Init_Ether unit.
- 2) Ether_CSR - This input is the Control and Status register of the DEQNA hardware. It supplies status information when read.

#1500-15-031.02.0

3.3.10.2.4.20.1 Outputs

The following outputs are produced by the unit:

- 1) Status - This thirty-two bit return parameter indicates whether a successful transmit operation occurred. If the transmit was successful, then NOERROR is returned, otherwise ERROR is returned.
- 2) Ether_CSR - This output is the Control and Status register of the DEQNA hardware. Control information is passed to the device when it is written.
- 3) Xmit_BDL_Reg - This DEQNA device register is used to load the address of the Transmit_BDL into the DEQNA to begin a transmit operation.

3.3.10.2.4.20.3 Local Data

The following local data is defined for the unit:

- 1) Ether_Hdr - This item consists of seven 16 bit words used to hold the header for the received ethernet packet. The item contains the following fields:
 - 1) Dest_Addr - 48 bit (three 16 bit words) Ethernet address for the destination of the packet.
 - 2) Src_Addr - 48 bit (three 16 bit words) Ethernet address for the source of the packet.
 - 3) Ether_Type - 16 bit word containing the type of Ethernet packet. This field is used to identify the higher level protocol carried by the packet.

#1500-15-031.02.0

3.3.10.2.4.20.4 Processing

Set the Dest_Addr field of Ether_Hdr to the destination Ethernet address of the cooperating host

Set the Src_Addr field of Ether_Hdr to the Ethernet address of the IGW

Set the Ether_Type field of Ether_Hdr to IP_TYPE

Set the Address_Bits of the first BDL in Transmit_BDL to the address of Ether_Hdr

Set the Addr_Descriptor_Bits of first BDL in Transmit_BDL to VALID_ADDRESS

Set the Buffer_Length field of the first BDL in Transmit_BDL to 7
/* seven 16 bit words in Ethernet header */

Set the Flag field of first BDL in Transmit_BDL to INITIALIZED

Set the Address_Bits of the second BDL in Transmit_BDL to the IP_Hdr field of Dgram

Set the Addr_Descriptor_Bits of second BDL in Transmit_BDL to VALID_ADDRESS

Set the Buffer_Length field of the second BDL in Transmit_BDL to 10
/* ten 16 bit words in IP header */

Set the Flag field of second BDL in Transmit_BDL to INITIALIZED

Set the Address_Bits of the third BDL in Transmit_BDL to the UDP_Hdr field of Dgram

Set the Addr_Descriptor_Bits of third BDL in Transmit_BDL to VALID_ADDRESS

Set the Buffer_Length field of the third BDL in Transmit_BDL to 4
/* four 16 bit words in UDP header */

Set the Flag field of third BDL in Transmit_BDL to INITIALIZED

Set the Address_Bits of the fourth BDL in Transmit_BDL to the Dgram_Msg field of Dgram

Set the Addr_Descriptor_Bits of fourth BDL in Transmit_BDL to the logical or of VALID_ADDRESS and END_MSG.

Set the Buffer_Length field of the fourth BDL in Transmit_BDL to 4
/* four 16 bit words of datagram data */

Set the Flag field of fourth BDL in Transmit_BDL to INITIALIZED

Set the Address_Bits of the fifth BDL in Transmit_BDL to the zero

Set the Addr_Descriptor_Bits of fifth BDL in Transmit_BDL to INVALID_ADDRESS

Set the Buffer_Length field of the fifth BDL in Transmit_BDL to 0

Set the Flag field of fifth BDL in Transmit_BDL to INITIALIZED

Write 0 to Ether_CSR

Write address of Transmit_BDL to Xmit_BDL_Reg

#1500-15-031.02.0

/* starts transmission */

```
Set Csr to Ether_CSR
While (Csr does not indicate transmission completed)
    Set Csr to Ether_CSR
Endwhile
```

3.3.10.2.4.20.5 Limitations

3.3.10.2.4.21 Send_Message

The Send_Message unit prepares a message to the cooperating host which requests that the host begin downloading either IGW software or IXIB software to the IGW.

3.3.10.2.4.21.1 Inputs

The following input is used by the unit:

- 1) Message_Type - This thirty-two bit parameter specifies the message sent to the host. The value of the item is either SEND_IGW or SEND_IXIB (these values are defined under global data).

#1500-15-031.02.0

3.3.10.2.4.21.1 Outputs

There are no outputs from this unit.

3.3.10.2.4.21.3 Local Data

The unit uses the following local data:

- 1) Message - This item is the message sent to the cooperating host requesting the host begin downloading. The message consists of two thirty-two bit words. The first word indicates the type of data to be returned:

- 1) IGW_SW - IGW software
- 2) IXIB_SW - IXIB software

The second word contains the size of the IGW memory in bytes. This word is only applicable for IGW_SW messages.

- 2) Dgram - This item is a thirty-two bit pointer to a datagram structure created and returned by the Create_Dgram unit.
- 3) Status - This thirty_two bit item contains the status returned by the Send_Dgram unit.

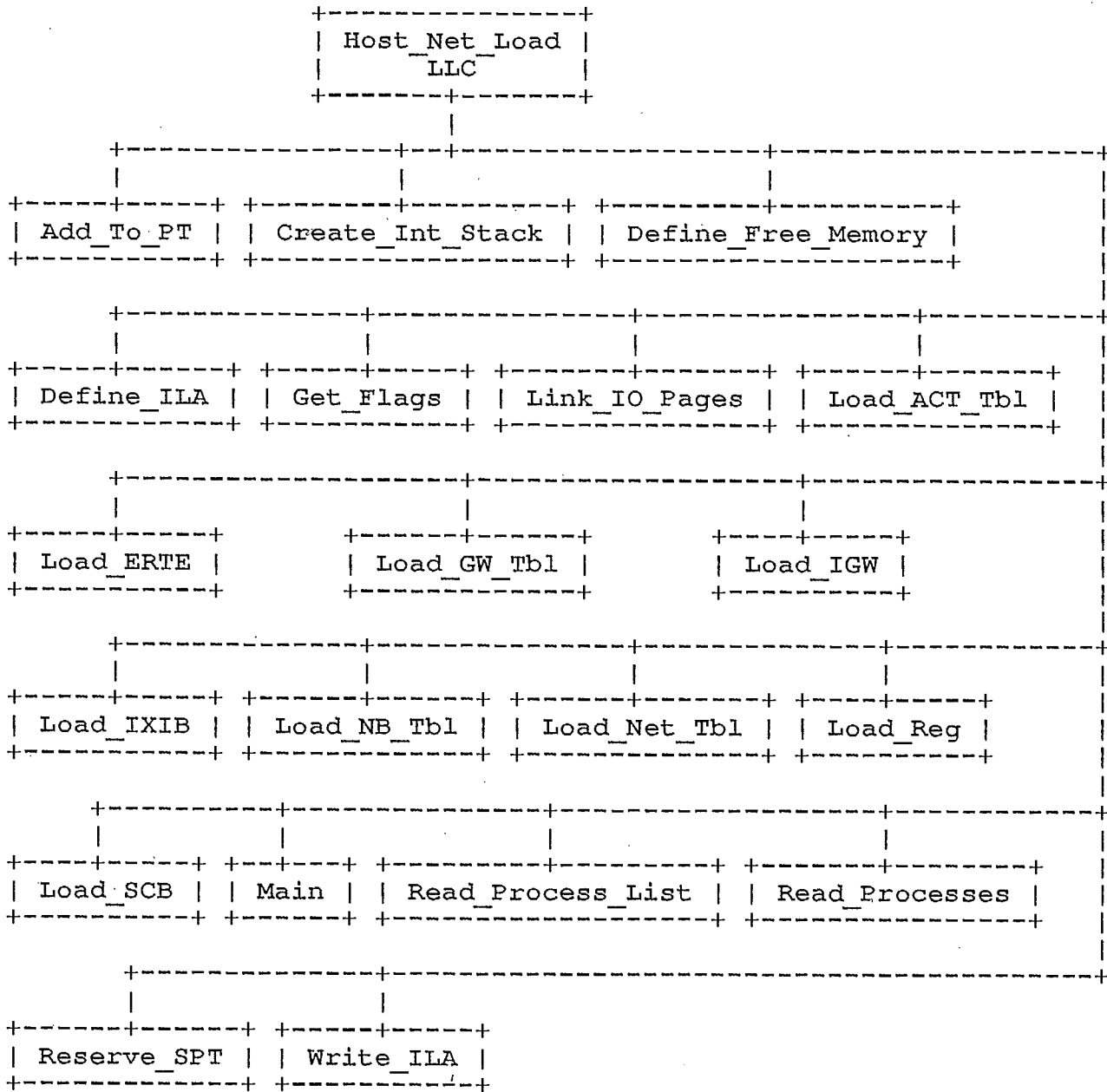


Figure 3-12

#1500-15-031.02.0

3.3.10.2.4.21.4 Processing

```
Copy Message_Type to Message
Dgram = Create_Dgram(Message)
Status = Send_Dgram(Dgram)
If (Status != NOERROR)
    return(ERROR)
Endif
```

3.3.10.2.4.21.5 Limitations

There are no limitations defined for this unit.

3.3.10.3 Host Net Load LLC

The Host Net Load LLC contains the host software that is responsible for the loading of the IGW system from a remote host. To load the IGW software, an image of the IGW memory containing the IGW software is built in a file, and the file is then sent to the IGW.

3.3.10.3.1 Host Net Load LLC Architecture

The Host Net Load LLC consists of the following units as shown in Figure 3-12:

- 1) Load_SCB Unit - This unit loads the System Control Block to the IGW_Image file.
- 2) Reserve_SPT Unit - This unit reserves the space required to contain the System Page Table.
- 3) Define_ILA Unit - This unit defines the structure of the

#1500-15-031.02.0

IGW Link Area. This area contains information and pointers to information that are used globally by the IGW. As part of the ILA definition procedure SPT entries are added to the System Page Table to reference the pages of the IGW Link Area.

- 4) Load_ERTE Unit - This unit is responsible for the loading of the ERTE from diskette into IGW memory.
- 5) Read_Process_List Unit - This unit reads the list of processes that are to be loaded on the IGW.
- 6) Read_Processes Unit - This unit makes use of the list obtained by the Read_Process_List Unit to read in the IGW processes from disk. This unit also sets up PPT and SPT entries, allocates stack space (by the use of the Allocate_Stacks Unit), and updates the PCB in Process_List.
- 7) Load_ACT_Tbl Unit - This unit reads the X.121 Address Configuration Table from disk and loads it into system virtual address space. The IGW Link Area is updated to reference the system virtual address of the loaded table.
- 8) Load_Net_Tbl Unit - This unit reads the X.121 Address Configuration Table from disk and loads it into system virtual address space. The IGW Link Area is updated to reference the system virtual address of the loaded table.
- 9) Load_GW_Tbl Unit - This unit reads the Gateway Table from disk and loads it into system virtual address space. The IGW Link Area is updated to reference the system virtual address of the loaded table.
- 10) Load_NB_Tbl Unit - This unit reads the Neighbor Table from disk and loads it into system virtual address space. The IGW Link Area is updated to reference the system virtual address of the loaded table.
- 11) Create_Int_Stack Unit - This unit allocates space for the interrupt stack. System page table entries are added and hardware registers are set during the allocation procedure.
- 12) Define_Free_Mem Unit - This unit sets up the system page table entries required to reference the free memory of the IGW.

#1500-15-031.02.0

- 13) Link_IO_Pages Unit - This unit sets up a pointer in the IGW Link Area to reference the area of memory that is designated for I/O.
- 14) Load_IXIB Unit - This unit reads the IXIB software from disk and send it to the IGW.
- 15) Add_To_PT Unit - This unit is called to add an entry to a page table.
- 16) Get_Flags - This unit converts a flags string into a bit pattern.
- 17) Main - This unit is responsible for listening for load requests and calling the appropriate units to load the software.
- 18) Load_IGW - This units is called to load the IGW software to the requesting IGW.
- 19) Load_IXIB - This units is called to load the IXIB software to the requesting IGW.
- 20) Write_ILA - This unit is called to write the ILA to the IGW_Image file.
- 21) Load_Reg - This unit is called to load register values required by the IGW into a buffer to be sent to the IGW.

3.3.10.3.2 Global Data

This section describes the format and contents of the data which is defined to be globally used between the units contained within the Host Net Load procedure.

- 1) Free_Phys - This global data item consists of a 32 bit integer containing the physical address of the start of memory that has not been allocated yet.
- 2) Free_Virt - This global data item consists of a 32 bit integer containing the virtual address of the start of memory that has not been allocated yet.

#1500-15-031.02.0

- 3) Istack_Phys - This global data item consists of a 32 bit integer containing the physical address of the top of the interrupt stack.
- 4) Istack_Virt - This global data item consists of a 32 bit integer containing the virtual address of the top of the interrupt stack.
- 5) Proc_List - This global data item consists of an array of 256 bytes containing the names of the files that processes are to be loaded from. Each file name is separated by a newline character and the list is terminated by a null character.
- 6) Sys_PT - This global data item consists of a 32 bit integer containing the starting physical address of the system page table.
- 7) Sys_Len - This global data item contains the length of the system page table.
- 8) ILA - This global data item contains a memory image of the IGW ILA.
- 9) ILA_Phys - This global data item contains the physical address of the start of the IGW ILA.
- 10) ERTE_Virt - This global data item contains the virtual address of the start of ERTE.
- 11) Table_Phys - This global data item contains the physical address of the table area.
- 12) Table_Virt - This global data item contains the virtual address of the table area.
- 13) IGW_Image - This global data item consists of a file used to access the IGW_Image file.

#1500-15-031.02.0

3.3.10.3.4 Host Net Load Units

The following subsections contain the unit descriptions for the units comprising the Host Net Load LLC.

3.3.10.3.4.1 Add_To_PT Unit

The Add_To_PT Unit adds page table entries to either the system or the process page tables.

3.3.10.3.4.1.1 Inputs

The following inputs are used by the Add_To_PT Unit:

- 1) PT_Addr - This input contains the starting physical address of the page table that page table entries are to be added to.
- 2) Phys_Addr - This input contains the physical address of the page that is to be added to the page table.
- 3) Virt_Addr - This input contains the virtual address of the page that is to be added to the page table.

#1500-15-031.02.0

3.3.10.3.4.1.2 Outputs

The following outputs are produced by the Add_To_PT Unit:

- 1) Page Tables - This output is written to the page table specified by the PT_Addr input. The format of these page tables is given in the global data section.

3.3.10.3.4.1.3 Local Data

The following local data is defined for the Add_To_PT Unit:

- 1) Cur_Pos - This local data item is used to store the current position in the IGW_Image file.
- 2) Cur_PTE - This local data item hold a page table entry as described in the VAX Architecture Handbook.

3.3.10.3.4.1.4 Processing

```
Cur_Pos = tell(IGW_Image)
Call lseek(IGW_Image, VPN of Virt_Addr + PT_Addr, L_SET)
Move PFN of Phys_Addr to PFN of Cur_PTE
Set PT_Valid field in Cur_Pte
Move PT_UW to PT_Prot field in Cur_PTE
Call write(IGW_Image, Address of Cur_PTE, 4)
Call lseek(IGW_Image, Cur_Pos, L_SET)
Return
```

#1500-15-031.02.0

3.3.10.3.4.1.5 Limitations

This unit performs no checks for incorrect virtual addresses, so specifying invalid virtual address could result in page table entries to be written to incorrect locations outside of the page table in physical memory.

3.3.10.3.4.2 Create_Int_Stack Unit

The Create_Int_Stack Unit Reserves an area in physical memory following the global tables to contain the interrupt stack.

3.3.10.3.4.2.1 Inputs

The following inputs are defined for the Create_Int_Stack Unit:

- 1) Free_Phys - This input is read from global data and contains the free physical memory address where the interrupt stack is located.
- 2) Free_Virt - This input is read from global data and contains the free virtual memory address where the interrupt stack is placed.

#1500-15-031.02.0

3.3.10.3.4.2.2 Outputs

The following outputs are produced by the Create_Int_Stack Unit:

- 1) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 2) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 3) Istack_Virt - This output is written to global data and contains the virtual address of the initial interrupt stack pointer.

3.3.10.3.4.2.3 Local Data

No local data is defined for the Create_Int_Stack Unit.

3.3.10.3.4.2.4 Processing

Add ISTACK_SIZE to Free_Phys
Add ISTACK_SIZE to Free_Virt
Adjust Free_Phys and Free_Virt to point to next page boundary if
necessary
Move Free_Virt to Istack_Virt
Return

#1500-15-031.02.0

3.3.10.3.4.2.5 Limitations

No limitations are defined for the Create_Int_Stack Unit.

3.3.10.3.4.3 Define_Free_Mem Unit

The Define_Free_Mem Unit defines the system virtual addresses for the area in physical memory from the beginning of the tables to the end of the free memory.

3.3.10.3.4.3.1 Inputs

The following inputs are defined for the Define_Free_Mem Unit:

- 1) Table_Phys - This input is read from global data and contains the physical memory address where table storage begins.
- 2) Table_Virt - This input is read from global data and contains the virtual memory address where table storage begins.
- 3) Sys_Pt - This input is read from global data and contains the physical address of the system page table that has been defined by the Reserve_SPT Unit.

#1500-15-031.02.0

3.3.10.3.4.3.2 Outputs

The following outputs are produced by the Define_Free_Mem Unit:

- 1) SPT - This output is written to the system page table by the use of the Add_To_PT Unit, and contains new entries which are added to the system page table.
- 2) Free_Virt - This output is written to global data and contains the updated value for the next free address in physical memory.
- 3) ILA - This output is written to the ILA area and is updated with the virtual address for the start of the IGW free memory.

3.3.10.3.4.3.3 Local Data

No local data is defined for the Define_Free_Mem Unit.

3.3.10.3.4.3.4 Processing

```
Move Free_Virt to ILA entry for free virtual memory
For each page N starting at Table_Phys to end of physical memory
  Call Add_To_PT(Page_Table = Sys_PT,
    Virt_Addr = N * PAGE_SIZE + Table_Virt,
    Phys_Addr = N * PAGE_SIZE + Table_Phys)
Endfor
Return
```

#1500-15-031.02.0

3.3.10.3.4.3.5 Limitations

No limitations are defined for the Define_Free_Mem Unit.

3.3.10.3.4.4 Define_ILA Unit

The Define_ILA Unit Reserves an area in physical memory following the system page table to contain the IGW Link Area. System page table entries are created for this area referencing system virtual addresses starting at the beginning of the system virtual address space. The ILA data structure isn't written to the IGW_Image disk file until the rest of IGW_Image is completed.

3.3.10.3.4.4.1 Inputs

The following inputs are defined for the Define_ILA Unit:

- 1) Free_Phys - This input is read from global data and contains the free physical memory address where the ILA area is to be placed.
- 2) Free_Virt - This input is read from global data and contains the free virtual memory address where the ILA area is to be placed.
- 3) Sys_Pt - This input is read from global data and contains the physical address of the system page table that has been defined by the Reserve_SPT Unit.

#1500-15-031.02.0

3.3.10.3.4.4.2 Outputs

The following outputs are produced by the Define_ILA Unit:

- 1) SPT - This output is written to the system page table by the use of the Add_To_PT Unit, and contains new entries which are added to the system page table.
- 2) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 3) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 4) ILA - This output updates the IGW Link Area data structure with any fields of that area that are to be initialized.

3.3.10.3.4.4.3 Local Data

No local data is defined for the Define_ILA Unit.

3.3.10.3.4.4.4 Processing

Move Free_Phys to ILA_Phys

For each page in ILA

 Call Add_To_PT(Page_Table = Sys_PT, Free_Virt, Free_Phys)

 Add PAGE_SIZE to Free_Phys

 Add PAGE_SIZE to Free_Virt

Endfor

Move Sys_Pt to SPT_Address field of ILA

Return

#1500-15-031.02.0

3.3.10.3.4.4.5 Limitations

No limitations are defined for the Define_ILA Unit.

3.3.10.3.4.5 Get_Flags Unit

The Get_Flags Unit is used to convert a character string containing flags to a bit representation of those flags.

3.3.10.3.4.5.1 Inputs

The following inputs are required by the Get_Flags Unit:

- 1) Set_Flags - This input consists of a character string containing the flags that are set.
- 2) All_Flags - This input consists of a character string containing all possible flags given in the correct bit order.

#1500-15-031.02.0

3.3.10.3.4.5.2 Outputs

The following outputs are produced by the Get_Flags Unit:

- 1) Flags - This output consists of the flags given in Set_Flags stored in bit positions.

3.3.10.3.4.5.3 Local Data

No local data is defined for the Get_Flags Unit.

3.3.10.3.4.5.4 Processing

Clear Flags

For each Flag from 0 to N - 1 in All_Flags

 If Flag N is in Set_Flags

 Bitwise or (1 left shifted by N) into Flags

 Endif

Endfor

Return Flags

3.3.10.3.4.5.5 Limitations

No limitations are defined for this unit.

3.3.10.3.4.6 Link_IO_Pages Unit

The Link_IO_Pages Unit is used to create system virtual address for the IO pages and store the starting virtual address of the IO pages in the ILA.

#1500-15-031.02.0

3.3.10.3.4.6.1 Inputs

No inputs are defined for the Link_IO_Pages Unit.

- 1) Free_Virt - This input is read from global data and contains the next free virtual address.
- 2) Sys_Pt - This input is read from global data and contains the address of the system page table.

3.3.10.3.4.6.2 Outputs

The following outputs are produced by the Link_IO_Pages Unit:

- 1) SPT - This output is written to the system page table by the use of the Add_To_PT Unit, and contains new entries which are added to the system page table.
- 2) ILA - This output is written to the IGW link area data structure to indicate the starting system virtual address of the IO pages.

3.3.10.3.4.6.3 Local Data

No local data is defined for the Link_IO_Pages Unit.

3.3.10.3.4.6.4 Processing

```
Move Free_Virt to Link_IO field of ILA
For each page N in the IO Space
  Call Add_To_PT(Page_Table = Sys_PT,
    Virt_Addr = N * PAGE_SIZE + Free_Virt,
    Phys_Addr = N * PAGE_SIZE + IO_PHYS)
Endfor
Return
```

#1500-15-031.02.0

3.3.10.3.4.6.5 Limitations

No limitations are defined for the Link_IO_Pages Unit.

3.3.10.3.4.7 Load_ACT_Tbl Unit

The Load_ACT_Tbl Unit loads the X.25 Address Configuration Table from disk to the system virtual address space following the process page tables. The ILA is updated to indicate the correct address of this table.

3.3.10.3.4.7.1 Inputs

The following inputs are used by the Load_ACT_Tbl Unit:

- 1) X.25_ACT - This input is read from the file "x.25_act" and contains a copy of the X.25 Address Configuration Table. This file contains the following fields:
 - X121 - This field contains the X.121 address (1 to 15 bytes) of the table entry.
 - Inet - This field contains the IP address in dot notation for the table entry.
 - Size - This field contains the maximum size for a packet for the host described in the table entry.
 - Flags - This field contains flags describing a table entry.
- 2) Free_Phys - This input is read from global data and contains the free physical memory address where the X.25 Address Configuration Table is to be placed.

#1500-15-031.02.0

- 3) Free_Virt - This input is read from global data and contains the free virtual memory address where the X.25 Address Configuration Table is to be placed.

3.3.10.3.4.7.2 Outputs

The following outputs are produced by the Load_ACT_Tbl Unit:

- 1) ACT_Table - This output is written to the global data area of the IGW_Image file as new entries are added to the X.25 Address Configuration Table. This table contains the following fields:

ACT_X121 - This field consists of a 16 byte character string the X.121 address of the current entry.

ACT_Inet - This field consists of a 32 bit value indicating the IP address for the current entry.

ACT_Size - This field consists of a 16 bit value indicating the maximum size of a packet for the current entry.

ACT_Flags - This field consists of a 32 bit value containing the following flags.

REQ_REV (0x01) - Request reverse charging.
ACC_REV (0x02) - Accept reverse charging.
REJ_IN (0x04) - Reject incoming calls.
REJ_OUT (0x08) - Reject outgoing calls.
IXIB (0x10) - Remote is an IXIB.

- 2) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 3) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 4) ILA - This output is written to the IGW Link Area data structure and is updated with the address of the X.25 Address Configuration Table.

#1500-15-031.02.0

3.3.10.3.4.7.3 Local Data

The following local data is defined for the Load_ACT_Tbl Unit:

- 1) Input_Buffer - This data consists of a buffer of 100 bytes that is used to read records from the x.25_act file.
- 2) Host - This data item is used to hold the dot notation format of the IP addresses as they are read from the x.25_act file.
- 3) Flags - This data item is a character string used to hold the flags field of each entry that is read from x.25_act file.
- 4) ACT_Ptr - This data item is used to step through ACT_Table while adding table entries.
- 5) Fp - This local data item is a pointer to a FILE structure referencing the x.25_act file.
- 6) ACT_Table - This local data item is an image of the ACT_Table output.

3.3.10.3.4.7.4 Processing

```
Fp = fopen("x.25_act", "r")
If Fp is null
    Call printf(Error message)
    Call exit(-1)
Endif
Clear ACT_Table
Move Free_Virt to ACT_Table pointer in ILA
Move start address of ACT_Table to ACT_Ptr
While more data in x.25_act file
    Call fgets(Input_Buffer, bytes to read = 100, Fp)
    If first character in Input_Buffer is a '#'
        Continue next loop iteration
    Endif
    If ACT_Table is full
        Exit loop
    Endif
    If result of sscanf(Input_Buffer, "%s %s %d %s",
        Act_X121 field of ACT_Table entry pointed to by ACT_Ptr,
        Host,
        Address of ACT_Size field of ACT_Table entry pointed to by ACT_PTR,
```

#1500-15-031.02.0

```
    Flags) is -1
      Exit Loop
    Endif
    ACT_Inet field of ACT_Table entry pointed to by ACT_Ptr =
      Inet_Addr(Host)
    If ACT_Inet field of ACT_Table = -1
      Call Printf(error message indicating invalid ACT Entry)
      Continue next loop iteration
    Endif
    ACT_Flags field of ACT_Table entry pointed to by ACT_Ptr =
      bitwise or between ACT_VALID and Get_Flags(Flags, "RAIOX")
    Set ACT_Ptr to point to next entry in ACT_Table
  Endwhile
  Call fclose(Fp)
  Call write(IGW_Image, Address of ACT_Table, size of ACT_Table)
  Add size of ACT_Table to Free_Phys
  Add size of ACT_Table to Free_Virt
  Return
```

3.3.10.3.4.7.5 Limitations

No limitations are defined for the Load_ACT_Tbl Unit.

3.3.10.3.4.8 Load_ERTE Unit

The Load_ERTE Unit is responsible for the loading of the ERTE executable image from a file to the IGW memory image file. For each page of ERTE that is loaded a system page table entry is created.

#1500-15-031.02.0

3.3.10.3.4.8.1 Inputs

The following inputs are defined for the Load_ERTE Unit:

- 1) ERTE - This input is read from the file "ERTE" contains the header, text, data, and bss areas of the ERTE executable.
- 2) Free_Phys - This input is read from global data and contains the free physical memory address where ERTE is to be placed.
- 3) Free_Virt - This input is read from global data and contains the free virtual memory address where ERTE is to be placed.
- 4) Sys_Pt - This input is read from global data and contains the physical address of the system page table that has been defined by the Reserve_SPT Unit.

3.3.10.3.4.8.2 Outputs

The following outputs are produced by the Load_ERTE Unit:

- 1) ERTE Memory Image - This output is written to the disk IGW memory image file and contains the text, data, and bss areas for ERTE.
- 2) SPT - This output is written to the system page table by calling the Add_To_PT Unit, and contains the new page table entries for the memory occupied by ERTE.
- 3) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 4) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.

#1500-15-031.02.0

3.3.10.3.4.8.3 Local Data

The following local data is defined for the Load_ERTE Unit:

- 1) Exec_Header - This local data item is a structure that is used to hold the header from the ERTE executable that is loaded from diskette. The fields in this structure are all 32 bit values and are defined as follows:

A_Magic - This field contains the type of the executable image that is being loaded. Valid values for this field are:

OMAGIC (0407) - Old impure format.
NMAGIC (0410) - Read-only text.
ZMAGIC (0413) - Demand load format.

A_Text - This field contains the size of the text segment in bytes.

A_Data - This field contains the size of the initialized data segment in bytes.

A_Bss - This field contains the size of the uninitialized data segment in bytes.

A_Syms - This field contains the size of the symbol field.

A_Entry - This field contains of the address of the entry point of the loaded executable image.

A_Trsize - This field contains the size of the text relocation area.

A_Drsize - This field contains the size of the data relocation area.

- 2) Buffer - This local data item consists of an array of 512 bytes used to read ERTE from disk.
- 3) Bytes_Read - This local data item contains the number of bytes that have been read in result of a read call.
- 4) Byte_Count - This local data item is used to hold a count for the purpose of reading the ERTE image from disk.

#1500-15-031.02.0

3.3.10.3.4.8.4 Processing

```
Fd = open("ERTE", 0)
If Fd is less than 0
    Call printf(Error message)
    Call exit(-1)
Endif
Bytes_Read = read(Fd, Address of Exec_Header,
    Size of Exec_Header)
If Bytes_Read not equal size of Exec_Header
    Call printf(Error message)
    Call exit(-1)
Endif
If A_Magic field of Exec_Header is ZMAGIC
    If result of lseek(Fd, Offset = 1024, L_SET) is less than 0
        Call printf(Error message)
        Call exit(-1)
    Endif
Else if A_Magic field of Exec_Header isn't one of OMAGIC or NMAGIC ZMAGIC
    Call printf(Error message)
    Call exit(-1)
Endif
Move A_Text field of Exec_Header to Byte_Count
Add A_Data field of Exec_Header to Byte_Count
While Byte_Count is greater than 0
    Subtract 512 from Byte_Count
    If Byte count less than 0
        If Byte_Count equals -512
            Exit loop
        Endif
        Bytes_Read = read(Fd, Buffer, 512 + Byte_Count)
        Call write(IGW_Image, Buffer, Bytes_Read)
    Else
        Bytes_Read = read(Fd, Buffer, 512)
        If Bytes_Read isn't 512
            Call printf(Error message)
            Call exit(-1)
        Endif
        Call write(IGW_Image, Buffer, Bytes_Read)
    Endif
Endwhile
Call close(Fd)
Move Free_Virt to ERTE_Virt
For each page N in ERTE text, data, and bss areas
    Call Add_To_PT(Page_Table = Sys_PT, Free_Virt, Free_Phys)
    Free_Phys += PAGE_SIZE
    Free_Virt += PAGE_SIZE
Endfor
```

#1500-15-031.02.0

Call lseek(IGW_Image, Free_Phys, L_SET)
Return

3.3.10.3.4.8.5 Limitations

No limitations are defined for the Load_ERTE Unit.

3.3.10.3.4.9 Load_GW_Tbl Unit

The Load_GW_Tbl Unit loads the Gateway Table from the file "gateway" into the IGW_Image file. The gateway file is used to define the various gateways that the IGW may access in addition to those determined through EGP.

3.3.10.3.4.9.1 Inputs

The following input is used by the Load_GW_Tbl Unit:

- 1) Gateway - This input is read from the file "gateway" contains a copy of the Gateway Table. This file contains the following fields:

Dst_Net - The destination network that is accessed by a gateway table entry.

GW_Addr - The address of the gateway to route packets for the specified destination network.

Mask - The IP network address mask. This field consists of a hexadecimal constant specifying the IP network address mask for the destination network.

Hop - The number of gateways that must be crossed to reach the destination.

#1500-15-031.02.0

Flags - This field consists of user definable flags.
Valid flags are:

- E - Report route via EGP.
- G - Gatewayed host. Delete route if the gateway goes down.
- R - Attempt to reroute datagrams if the gateway goes down.

- 2) Free_Phys - This input is read from global data and contains the free virtual memory address where the gateway table is placed.
- 3) Free_Virt - This input is read from global data and contains the free virtual memory address where the gateway table is to be placed.

3.3.10.3.4.9.2 Outputs

The following outputs are produced by the Load_GW_Tbl Unit:

- 1) GW_Table - This output is written to the global data area as new entries are added to the Gateway Table. This table contains the following fields:
- 2) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 3) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 4) ILA - This output is written to the IGW Link Area data structure and is updated with the address of the gateway table.

#1500-15-031.02.0

3.3.10.3.4.9.3 Local Data

The following local data is defined for the Load_GW_Tbl Unit:

- 1) Input_Buffer - This data consists of a buffer of 100 bytes that is used to read records from the gateway file.
- 2) GW_Addr - This data item consists of a character string used to hold the IP address for each entry that is read from the gateway file.
- 3) Dst_Net - This data item consists of a character string used to hold the destination network number for each entry that is read from the gateway file.
- 4) Flags - This data item is used to hold the flags field of each entry that is read from the gateway file.
- 5) GW_Ptr - This data item is used to step through GW_Table while adding table entries.
- 6) Fp - This local data item is a pointer to the FILE structure describing the gateway file.
- 7) GW_Table - This local data item is a memory image of the GW_Table output.

3.3.10.3.4.9.4 Processing

```
Fp = fopen("gateway", 0)
If Fp is Null
    Call printf(Error message)
    Call exit(-1)
Endif
Clear entries in GW_Table
Move Free_Virt to GW_Table pointer in ILA
Move start address of GW_Table to GW_Ptr
While more data in gateway file
    Call fgets(Input_Buffer, bytes to read = 100, Fp)
    If first character in input buffer is a '#'
        Continue next loop iteration
    Endif
    If GW_Table is full
        Exit loop
    Endif
    If result of sscanf(Input_Buffer, "%s %s %x %d %s %s",
```

#1500-15-031.02.0

```
    Dst_Net, GW_Addr,
    Address of GW_Mask field of GW_Table entry pointed to by GW_Ptr,
    Address of GW_Hop field of GW_Table entry pointed to by GW_Ptr,
    Flags) is -1
        Exit Loop
    Endif
    GW_Dst_Net field of GW_Table entry pointed to by GW_Ptr =
        Inet_Addr(Dst_Net)
    If GW_Dst_Net field of GW_Table entry pointed to by GW_Ptr = -1
        Call Printf(error message indicating invalid gateway entry)
        Continue next loop iteration
    Endif
    GW_GW_Addr field of GW_Table entry pointed to by GW_Ptr =
        Inet_Addr(GW_Addr)
    If GW_GW_Addr field of GW_Table entry pointed to by GW_Ptr = -1
        Call Printf(error message indicating invalid gateway entry)
        Continue next loop iteration
    Endif
    Move index of entry in Net_Table with same network address as the
    network portion of the gateway address to GW_Number
    field of GW_Table entry pointed to by GW_Ptr
    GW_Flags field of GW_Table entry pointed to by GW_Ptr =
    bitwise or between GW_VALID flag and Get_Flags(Flags, "EGR")
    Set GW_Ptr to point to next entry in GW_Table
Endwhile
Call fclose(Fp)
Add size of GW_Table to Free_Phys
Add size of GW_Table to Free_Virt
Call write(IGW_Image, Address of GW_Table, Size of GW_Table)
Return
```

3.3.10.3.4.9.5 Limitations

The unit Load_Net_Table must be executed before this unit.

#1500-15-031.02.0

3.3.10.3.4.10 Load_IGW Unit

The Load_IGW Unit is called to load the IGW software to the IGW that is requesting to be loaded.

3.3.10.3.4.10.1 Inputs

The following inputs are defined for the IGW_Load Unit:

- 1) IGW Image - This input is a file that contains the created IGW memory image.
- 2) S - This input contains the file descriptor to reference the socket that is used to access the remote IGW.
- 3) Remote - This input consists of a structure of type sockaddr_in and contains the address information for the IGW that is requesting the IGW software.

3.3.10.3.4.10.2 Outputs

The following outputs are produced by the Load_IGW Unit:

- 2) S - This item contains the file descriptor to reference the socket that is used to send data to the remote IGW.

#1500-15-031.02.0

3.3.10.3.4.10.3 Local Data

The following local data is defined for the Main Unit.

- 1) Bytes_Read - This local data item contains the number of bytes have been read from the IGW_Image file on a read request.
- 2) Buffer - This local data item is a buffer of 513 characters used in reading from IGW_Image and sending data to the IGW.

3.3.10.3.4.10.4 Processing

```
IGW_Image = open("IGW_Image", (read only, create, truncate))
If IGW_Image is less than 0
    Call printf(Error message)
    Call exit(-1)
Endif
Call Load_SCB()
Call Reserve_SPT()
Call Define_ILA()
Call Load_ERTE()
Call Read_Process_List()
Call Read_Processes()
Move Free_Phys to Table_Phys
Move Free_Virt to Table_Virt
Call Load_ACT_Tbl()
Call Load_Net_Tbl()
Call Load_GW_Tbl()
Call Load_NB_Tbl()
Reserve space for unloaded tables
Call Create_Int_Stack()
Call Define_Free_Mem()
Call Link_IO_Pages()
Call Write_ILA()
Call lseek(IGW_Image, 0L, L_SET)
Move NL_DATA to first byte in Buffer
Loop
    Bytes_Read = read(IGW_Image, Address of second byte in
        Buffer, 512)
    If Bytes_Read is less than or equal 0
        Exit Loop
    Endif
    Call sendto(S, Buffer, 513, 0, Address of Remote, Size of
        Remote)
Endloop
```

#1500-15-031.02.0

```
Call close(IGW_Image)
Move NL_END to first byte of Buffer
Call Load_Reg(Address of second byte in Buffer)
Call sendto(S, Buffer, 513, 0, Address of Remote, Size of Remote)
Return
```

3.3.10.3.4.10.5 Limitations

No limitations are defined for this unit.

3.3.10.3.4.11 Load_IXIB Unit

The Load_IXIB Unit is called to load the IXIB software to the IGW that is requesting the IXIB software to be loaded.

3.3.10.3.4.11.1 Inputs

The following inputs are defined for the IGW_Load Unit:

- 1) IXIB.S28 - This input is obtained from the disk file "IXIB.S28" and contains the IXIB software in Motorola S-Record format.
- 2) S - This input contains the file descriptor to reference the socket that is used to access the remote IGW.
- 3) Remote - This input consists of a structure of type sockaddr_in and contains the address information for the IGW that is requesting the IXIB software.

#1500-15-031.02.0

3.3.10.3.4.11.2 Outputs

The following outputs are produced by the Load_IGW Unit:

- 2) S - This item contains the file descriptor to reference the socket that is used to send data to the remote IGW.

3.3.10.3.4.11.3 Local Data

The following local data is defined for the Main Unit.

- 1) Bytes_Read - This local data item contains the number of bytes have been read from the IGW_Image file on a read request.
- 2) Buffer - This local data item is a buffer of 513 characters used in reading from IGW_Image and sending data to the IGW.
- 3) IXIB_S28 - This local data item contains the file descriptor to access the IXIB.S28 input.

3.3.10.3.4.11.4 Processing

```
IXIB_S28 = open("IXIB.S28", 0)
If IXIB_S28 is less than 0
    Call printf(Error message)
    Call exit(-1)
Endif
Move NL_DATA to first byte in Buffer
Loop
    Bytes_Read = read(IXIB_S28, Address of second byte in
        Buffer, 512)
    If Bytes_Read is less than or equal 0
        Exit Loop
    Endif
    Call sendto(S, Buffer, 513, 0, Address of Remote, Size of
        Remote)
Endloop
Call close(IXIB_S28)
Move NL_END to first byte of Buffer
Call sendto(S, Buffer, 513, 0, Address of Remote, Size of Remote)
Return
```

#1500-15-031.02.0

3.3.10.3.4.11.5 Limitations

No limitations are defined for this unit.

3.3.10.3.4.12 Load_NB_Tbl Unit

The Load_NB_Tbl Unit loads the EGP Neighbour Table from the file "neighbour" into the IGW_Image file. The neighbour file is used to define information describing the gateways that the IGW can communicate via EGP with.

3.3.10.3.4.12.1 Inputs

The following inputs are used by the Load_NB_Tbl Unit:

- 1) Neighbour - This input is read from the disk file "neighbor" and contains a copy of the EGP Neighbour Table. This file contains the following fields:

IP_ADDR - The Internet address of the EGP neighbour gateway in dot notation.

Flags - This field consists of user definable flags. Valid flags are:

- M - Gateway is a main neighbour.
- O - Gateway is an alternate neighbour.
- S - Gateway is a stub gateway.

- 2) Free_Phys - This input is read from global data and contains the free physical memory address where the neighbour table is to be placed.
- 3) Free_Virt - This input is read from global data and contains the free virtual memory address where the

#1500-15-031.02.0

neighbour table is to be placed.

3.3.10.3.4.12.2 Outputs

The following outputs are produced by the Load_NB_Tbl Unit:

- 1) NB_Table - This output is written to the global table area of physical memory as new entries are added to the Neighbour Table.
- 2) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 3) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 4) ILA - This output is written to the IGW Link Area and is updated with the address of the neighbour table.

3.3.10.3.4.12.3 Local Data

The following local data is defined for the Load_NB_Tbl Unit:

- 1) Input_Buffer - This data consists of a buffer of 100 bytes that is used to read records from the neighbour file.
- 2) IP_Addr - This data item consists of a character string used to hold the IP address of each EGP neighbour gateway entry that is read from the neighbour file.
- 3) Flags - This data item is used to hold the flags field of each entry that is read from the neighbour file.
- 4) NB_Ptr - This data item is used to step through NB_Table while adding table entries.
- 5) Fp - This local data item contains a pointer to the FILE structure that references the neighbour file.
- 6) NB_Table - This local data item contains a memory image of the NB_Table output.

#1500-15-031.02.0

3.3.10.3.4.12.4 Processing

```
Fp = fopen("neighbour", "r")
If Fp is null
    Call printf(Error message)
    Call exit(-1)
Endif
Clear entries in NB_Table
Move Free_Virt to NB_Table pointer in ILA
Move start address of NB_Table to NB_Ptr
While more data in neighbour file
    Call fgets(Input_Buffer, bytes to read = 100, Fp)
    If first character in input buffer is a '#'
        Continue next loop iteration
    Endif
    If NB_table is full
        Exit loop
    Endif
    If result of sscanf(Input_Buffer, "%s %s", IP_Addr, Flags) is -1
        Exit Loop
    Endif
    NB_IP_Addr field of NB_Table entry pointed to by NB_Ptr =
        Inet_Addr(IP_Addr)
    If NB_IP_Addr field of NB_Table entry pointed to by NB_Ptr = -1
        Call Printf(error message indicating invalid neighbour entry)
        Continue next loop iteration
    Endif
    NB_Flags field of NB_Table entry pointed to by NB_Ptr =
        bitwise or between NB_VALID flag and Get_Flags(Flags, "MOS")
    Set NB_Ptr to point to next entry in NB_Table
Endwhile
Call fclose(Fp)
Add size of NB_Table to Free_Phys
Add size of NB_Table to Free_Virt
Call write(IGW_Image, Address of NB_Table, Size of NB_Table)
Return
```

#1500-15-031.02.0

3.3.10.3.4.12.5 Limitations

No limitations are defined for the Load_NB_Tbl Unit.

3.3.10.3.4.13 Load_Net_Tbl Unit

The Load_Net_Tbl Unit Loads the Network Table from the file "network" into the IGW main memory. This file is used to define the network interface information required for each network that the IGW is connected to.

3.3.10.3.4.13.1 Inputs

The following inputs are used by the Load_Net_Tbl Unit:

- 1) Network - This input is read from the file "network" and contains a copy of the Network Table. This file contains the following fields:

IP_Addr - The local Internet address of the IGW on the referenced network. This field is a string containing the IP address in dot notation.

Interface_Id - The interface number of the network interface represented by this entry. Each interface is given a number which is used to direct datagrams to the correct interface for transmission.

Mask - The IP network address mask. This field consists of a hexadecimal constant specifying the IP network address mask.

MTU - The maximum transmission unit for IP datagrams. This value is specified as an integer.

#1500-15-031.02.0

Flags - This field consists of one user definable flag which is "U" indicating that the interface should be marked as being up.

- 2) Free_Phys - This input is read from global data and contains the free physical memory address where the gateway table is to be placed.
- 3) Free_Virt - This input is read from global data and contains the free virtual memory address where the gateway table is to be placed.

3.3.10.3.4.13.2 Outputs

The following outputs are produced by the Load_Net_Tbl Unit:

- 1) Net_Table - This output is written to the global data area as new entries are added to the Network Table. This table contains the following fields:
- 2) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 3) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 4) ILA - This output is written to the IGW Link Area data structure and is updated with the address of the gateway Table.

#1500-15-031.02.0

3.3.10.3.4.13.3 Local Data

The following local data is defined for the Load_Net_Tbl Unit:

- 1) Input_Buffer - This data consists of a buffer of 100 bytes that is used to read records from the network file.
- 2) IP_Addr - This data item is used to hold the IP address for each entry that is read from the network file.
- 3) Flags - This data item is used to hold the flags field of each entry that is read from the network file.
- 4) Net_Ptr - This data item is used to step through Net_Table while adding table entries.
- 5) Ep - This local data item contains a pointer to the FILE structure that references the network file.
- 6) Net_Table - This local data item contains a memory image of the Net_Table Output.
- 7) Interface_Number - This data item is used to hold the interface id number of each entry read from the network file.
- 8) Mask - This data item is used to hold the address mask for each entry read from the network file.
- 9) MTU - This data item is used to hold the network MTU for each entry read from the network file.

#1500-15-031.02.0

3.3.10.3.4.13.4 Processing

```
Fp = fopen("network", "r")
If Fp is null
    Call printf(Error message)
    Call exit(-1)
Endif
Clear entries in Net_Table
Move Free_Virt to Net_Table pointer in ILA
Move start address of Net_Table to Net_Ptr
While more data in network file
    Call fgets(Input_Buffer, bytes to read = 100, Fp)
    If first character in input buffer is a '#'
        Continue next loop iteration
    Endif
    If Net_Table is full
        Exit loop
    Endif
    If result of sscanf(Input_Buffer, "%s %s %x %d %s",
        IP_Addr, Interface_Number, Net_Mask, MTU, Flags) is -1
        Exit Loop
    Endif
    Search Net_Table for an entry with Net_IP_Addr field = IP_Addr
    If table entry found
        Add Interface_Number to the end of the Net_QID_List for the
        found entry
    Else
        Set Net_Ptr to the first empty position in Net_Table
        Net_IP_Addr field of Net_Table entry pointed to by Net_Ptr =
        Inet_Addr(IP_Addr)
        If Net_IP_Addr field Net_Table entry pointed to by Net_Ptr = -1
            Call Printf(error message indicating invalid network Entry)
            Continue next loop iteration
        Endif
        Net_Flags field of Net_Table entry pointed to by Net_Ptr =
        bitwise or between NET_VALID and Get_Flags(Flags, "U")
        Set Net_MTU referenced by Net_Ptr to MTU
        Set Net_Mask referenced by Net_Ptr to Mask
        Set Current_IF field referenced by Net_Ptr to zero.
    Endif
Endwhile
Call fclose(Fp)
Add size of Net_Table to Free_Phys
Add size of Net_Table to Free_Virt
Call write(IGW_Image, Address of Net_Table, size of Net_Table)
Return
```

#1500-15-031.02.0

3.3.10.3.4.13.5 Limitations

No limitations are defined for the Load_Net_Tbl Unit.

3.3.10.3.4.14 Load_Reg Unit

The Load_Reg Unit loads a buffer with initial values for processor registers required by the IGW.

3.3.10.3.4.14.1 Inputs

The following inputs are required by the Load_Reg Unit:

- 1) Buffer - This input contains the address of the output Buffer.

3.3.10.3.4.14.2 Outputs

The following outputs are required by the Load_Reg Unit:

- 1) Buffer - This output consists of an array of 32 bit values used to define the IGW processor registers. The entries of this array are referenced as follows:
 - 1) SCBB - This entry holds the system control block base register.
 - 2) ISP - This entry holds the address of the top of the system interrupt stack.
 - 3) SBR - This is the base register for the system page table.
 - 4) SLR - This is the length register for the system page table.

#1500-15-031.02.0

- 5) ERTE_VIRT - This is the virtual address of the ERTE entry point.

3.3.10.3.4.14.3 Local Data

No local data is defined for the Load_Reg Unit.

3.3.10.3.4.14.4 Processing

Move 0 to SCBB entry in Buffer
Move Istack_Virt to ISP entry in Buffer
Move Sys_PT to SBR entry in Buffer
Move SPT_LENGTH to SLR entry in Buffer
Move ERTE_Virt to ERTE_VIRT entry in Buffer

3.3.10.3.4.14.5 Limitations

No limitations are defined for the Load_Reg Unit.

3.3.10.3.4.15 Load_SCB Unit

The Load_SCB Unit loads the System Control Block from a file to the IGW_Image file.

#1500-15-031.02.0

3.3.10.3.4.15.1 Inputs

The following inputs are used by the Load_SCB Unit:

- 1) SCB_Init - This input is read from a file on disk. This file contains an image of the System Control Block.

3.3.10.3.4.15.2 Outputs

The following outputs are produced by the Load_SCB Unit:

- 1) SCB - This output is written to the address specified by the SCBB processor register (physical address 0), and contains the System Control Block that has been obtained from the SCB_Init input.
- 2) SCBB - This output is written to the System Control Block Base Register, and contains the physical address of the SCB.
- 3) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.

3.3.10.3.4.15.3 Local Data

The following local data is defined for the Load_SCB Unit:

- 1) Fd - This local data item contains a file descriptor for the SCB_Init file.
- 2) Buffer - This local data item is an input buffer for the system control block, it consists of an array of 512 bytes.
- 3) Bytes_Read - This local data item is a count for the number of bytes read by the read function call.

#1500-15-031.02.0

3.3.10.3.4.15.4 Processing

```
Fd = open("SCB_Init", 0)
If Fd is less than 0
    Call perror(Error message)
    Call exit(-1)
Endif
Bytes_Read = Read(Fd, Buffer, Size of SCB = 512)
If Bytes_Read not equal Size of SCB
    Call perror(Error message)
    Call exit(-1)
Endif
Call close(Fd)
Call write(IGW_Image, Buffer, Bytes_Read)
Move Size of SCB to Free_Phys
Return
```

3.3.10.3.4.15.5 Limitations

This unit must be called before any unit that allocates IGW physical memory.

3.3.10.3.4.16 Main Unit

The Main Unit is the first software in the Host_Net_Load LLC. This unit is responsible for listening for load requests from an IGW and calling the appropriate units to load the requested software.

#1500-15-031.02.0

3.3.10.3.4.16.1 Inputs

The following inputs are defined for the Main Unit:

- 1) IGW - This input is read from a socket consisting of UDP datagrams sent by an IGW.

3.3.10.3.4.16.2 Outputs

No outputs are defined for the Main Unit

3.3.10.3.4.16.3 Local Data

The following local data is defined for the Main Unit:

- 1) S - This local data item contains a file descriptor for the socket used to communicate to the IGW with.
- 2) Bytes_Read - This local data item is used to hold the number of bytes read into Buffer.
- 3) Remote - This local data item is a structure of type sockaddr_in, and is used to hold the sockaddr_in structure for the IGW.
- 4) Remote_Len - This local data item holds the length of Remote.
- 5) Buffer - This local data item is an array of 8 bytes used to read UDP packets from the IGW.
- 6) SP - This local data item is a pointer to a servant structure describing the IGW server.

#1500-15-031.02.0

3.3.10.3.4.16.4 Processing

```
SP = getservbyname("igw", "udp")
Run as daemon in background
S = socket(AF_INET, SOCK_DGRAM, 0)
Clear sin_addr field of Remote
Move s_port field of SP to sin_port field of Remote
Remote_len = sizeof(Remote)
Call bind(S, &Remote, Remote_Len)
Loop
    Bytes_Read = recvfrom(S, Buffer, 8, 0, Address of Remote,
        Address of Remote_Len)
    If Bytes_Read isn't 8
        Call printf(Error Message)
        Continue
    Endif
    Run as sub-process
        Case first integer in Buffer
            LOAD_IGW:
                Move second integer in Buffer to
                Memory_Size
                Call Load_IGW(S, Address of
                Remote)
            LOAD_IXIB:
                Call Load_IXIB(S, Address of
                Remote)
            Otherwise
                Call printf(Error message)
        Endcase
    End sub-process
Endloop
```

#1500-15-031.02.0

3.3.10.3.4.16.5 Limitations

No limitations are defined for the Main Unit.

3.3.10.3.4.17 Read_Process_List Unit

The Read_Process_List Unit reads the list of processes from a file on disk and stores it in the Proc_List area that is declared to be global within the Host_Net_Load LLC.

3.3.10.3.4.17.1 Inputs

The following input is used by the Read_Process_List Unit:

- 1) Proc_List File - This input is obtained from the "Proc_List" file and contains a copy of the names of the files containing the processes (and their priorities) that the Host_Net_Load LLC is to load into the IGW. Each entry is separated by newlines and process are separated from priorities by spaces.

#1500-15-031.02.0

3.3.10.3.4.17.2 Outputs

The following output is produced by the Read_Process_List Unit:

- 1) Proc_List - This output is written to the Proc_List array and contains the list of process that are to be loaded. The entries in the list are each separated by a newline character and the end of the list is indicated by a Null character following a newline character. Names are separated from priorities by spaces.

3.3.10.3.4.17.3 Local Data

The following local data is defined for the Read_Process_List Unit:

- 1) Bytes_Read - This local data item contains the number of bytes read while reading the Proc_List file.
- 2) Fd - This local data item contains the file descriptor used to access the Proc_List file.

3.3.10.3.4.17.4 Processing

```
Fd = open("Proc_List", 0)
If Fd is less than 0
    Call printf(Error message)
    Call exit(-1)
Endif
Bytes_Read = read(Fd, Address of Proc_List, bytes to read = 1024)
If Bytes_Read is less than or equal to 0
    Call Printf(Error message)
    Call exit(-1)
Endif
Call close(Fd)
Add a Null character to the end of Proc_List
Return
```

#1500-15-031.02.0

3.3.10.3.4.17.5 Limitations

The maximum size of the process list input file that this unit will accept is 1024 bytes.

3.3.10.3.4.18 Read_Processes Unit

The Read_Processes Unit loads the processes specified in the list "Proc_List" that has been created by the Read_Process_List Unit. This involves placing the process text, data, bss, and stack area in the IGW_Image file, and creating a process page table for them. System page table entries will also be added to reference the process page table.

3.3.10.3.4.18.1 Inputs

The following inputs are used by the Read_Processes Unit:

- 1) Proc_List - This input comes from the global data that has been loaded by the Read_Process_List Unit. This input contains a list of file names to load processes from as well as the priority of each of the processes.
- 2) Process Images From Disk - This input consists of the binary images of the IGW processes that are to be loaded from disk.
- 3) Free_Phys - This input is read from global data and contains the free physical memory address where the processes are to be loaded.
- 4) Free_Virt - This input is read from global data and

#1500-15-031.02.0

contains the free virtual memory address where the processes are to be loaded.

- 5) Sys_Pt - This input is read from global data and contains the physical address of the system page table that has been defined by the Reserve_SPT Unit.

3.3.10.3.4.18.2 Outputs

The following outputs are produced by the Read_Processes Unit:

- 1) Process_Header_List - This output is written to the ILA and contains the initialized process headers including PCBs for the processes that have been loaded into memory.
- 2) Processes In Memory - This output is written to the IGW_Image file and contains the text, data, bss, and stack areas of the IGW processes that have been loaded.
- 3) Process Page Tables - This output is written to the IGW_Image file and contains the process page tables for P0 and P1 address space for each process that is loaded by this unit.
- 4) System Page Table - This output is updated with the system page table entries required to reference the process page tables that have been created by this unit.
- 5) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 6) Free_Virt - This output is written to global data and contains the updated address of the next free virtual memory address.
- 7) ILA - This output contains the value for the number of processes that have been loaded. This value is written to the Nproc field of the IGW Link Area.

3.3.10.3.4.18.3 Local Data

The following local data is defined for the Read_Processes Unit:

- 1) `Process_Name_Pointer` - This local data item is used to step through the `Proc_List` input to obtain each `Current_Process_Entry` item for the entries in `Proc_List`.
- 2) `Current_Process_Entry` - This local data item is used to store the entry in the `Proc_List` input that contains the filename and priority of the current process.
- 3) `Current_Process_Name` - The local data item is used to store the name the current process that is being loaded from disk to the `IGW_Image` file.
- 4) `Current_Process_Priority` - This piece of data is used to store the priority of the processes as they are read from disk and loaded to the `IGW_Image` file.
- 5) `Exec_Header` - This local data item is a structure that is used to hold the header from the `IGW` processes that are loaded from diskette. The fields in this structure are all 32 bit values and are defined as follows:
 - `A_Magic` - This field contains the type of the executable image that is being loaded. Valid values for this field are:
 - `OMAGIC (0407)` - Old impure format.
 - `NMAGIC (0410)` - Read-only text.
 - `ZMAGIC (0413)` - Demand load format.
 - `A_Text` - This field contains the size of the text segment in bytes.
 - `A_Data` - This field contains the size of the initialized data segment `m` is used to hold the virtual address of the stack area.
- 10) `P0_Phys` - This local data item contains the physical address of the `P0` page table.
- 11) `P0_Virt` - This local data item contains the virtual address of the `P0` page table.
- 12) `P0_Len` - This local data item contains the length of the `P0` page table.

#1500-15-031.02.0

- 13) P1_Phys - This local data item contains the physical address of the P1 page table.
- 14) P1_Virt - This local data item contains the virtual address of the P1 page table.
- 15) P1_Len - This local data item contains the length of the P1 page table.
- 16) Nproc - This local data item contains the number of process that have been loaded into IGW memory.
- 17) Fd - This local data item contains the file descriptor for the process file that is being loaded.
- 18) Buffer - This local data item contains an array of 512 bytes used to read in processes.
- 19) Bytes_Read - This local data item contains the number of bytes returned from a read operation.
- 20) Byte_Count - This local data item is used to count the bytes that have been read from a process file.

3.3.10.3.4.18.4 Processing

Loop

```
Move Process_Name_Pointer to Current_Entry
Move address of next newline character in string pointed to
  by Process_Name_Pointer to Process_Name_Pointer
If no newline was found in the string pointed to by
  Process_Name_Pointer
  Break Loop
If process header list is full
  Process error condition
Endif
Move Null character to character pointed to by
  Process_Name_Pointer
Increment Process_Name_Pointer
scanf(Current_Entry, "%s %d", Current_Process_Name,
  address of Current_Process_Priority)
Fd = open(Current_Process_Name, 0)
If is less than 0
  Call printf(Error message)
  Call exit(-1)
Endif
Bytes_Read = read(Fd, address of Exec_Header,
```

#1500-15-031.02.0

```
size of Exec_Header) isn't equal size of Exec_Header
    Call printf(Error message)
    Call exit(-1)
Endif
If A_Magic field of Exec_Header is ZMAGIC
    If result of lseek(Fd, Offset = 1024, L_SET) is less than 0
        Call printf(Error message)
        Call exit(-1)
    Endif
Else if A_Magic field of Exec_Header isn't one of OMAGIC or NMAGIC
    Call printf(Error message)
    Call exit(-1)
Endif
Move A_Text field of Exec_Header to Byte_Count
Add A_Data field of Exec_Header to Byte_Count
While Byte_Count is greater than 0
    Subtract 512 from Byte_Count
    If Byte count less than 0
        If Byte_Count equals -512
            Exit loop
        Endif
        Bytes_Read = read(Fd, Buffer, 512 + Byte_Count)
        Call write(IGW_Image, Buffer, Bytes_Read)
    Else
        Bytes_Read = read(Fd, Buffer, 512)
        If Bytes_Read isn't 512
            Call printf(Error message)
            Call exit(-1)
        Endif
        Call write(IGW_Image, Buffer, Bytes_Read)
    Endif
Endwhile
Call close(Fd)
Move Free_Phys to Proc_Phys
Move 0 to Proc_Virt
Add A_Text field of Exec_Header to Free_Phys
Add A_Data field of Exec_Header to Free_Phys
Add A_Bss field of Exec_Header to Free_Phys
Adjust Free_Phys to point to page boundary if required
Move Free_Phys to Stack_Phys
Move stack virtual address to Stack_Virt
Add PROC_KERN_STACK_SIZE to Free_Phys
Add PROC_USER_STACK_SIZE to Free_Phys
Advance Free_Virt to start of next page if necessary
Move Free_Phys to PO_Phys
Move Free_Virt to PO_Virt
For each page in process text, data, and bss areas
    Call Add_To_PT(Page_Table = PO_Phys, Proc_Virt,
        Proc_Phys)
```

#1500-15-031.02.0

```
        Add PAGE_SIZE to Proc_Phys
        Add PAGE_SIZE to Proc_Virt
    Endfor
    Add length of P0 page table to Free_Phys
    Add length of P0 page table to Free_Virt
    Move P1 PT start physical address to P1_Phys
    Move P1 PT start system virtual address to P1_Virt
    For each page in stack area
        Call Add_To_PT(Page_Table = P1_Phys, Stack_Virt,
            Stack_Phys)
        Add PAGE_SIZE to Stack_Phys
        Add PAGE_SIZE to Stack_Virt
    Endfor
    Add length of P1 PT to Free_Phys
    Add length of P1 PT to Free_Virt
    Adjust Free_Phys to next page if necessary
    Adjust Free_Virt to next page if necessary
    Call lseek(IGW_Image, Free_Phys, L_SET)
    Set Name field of process entry indexed by Nproc in
        Process_Header_List to name stored in Current_Process_Name
    Set Priority field of process entry indexed by Nproc in
        Process_Header_List to priority stored in
        Current_Process_Priority
    Set PCB_Address field of process entry indexed by Nproc in
        Process_Header_List to the physical address of the hardware PCB
    Initialize kernel and user stack pointer in PCB
    Initialize Processor_Status_Longword in PCB
    Move P0_Virt to Program_Base_Register in the hardware PCB
    Move P0_Len to Program_Length_Register in the hardware PCB
    Move P1_Virt to Control_Base_Register in hardware PCB
    Move P1_Len to Control_Length_Register in hardware PCB
    For each page in process page tables P0 and P1
        Call Add_To_PT(Page_Table = Sys_PT,
            Virt_addr = P0_Virt,
            Phys_addr = P0_Phys)
        Add PAGE_SIZE to P0_Phys
        Add PAGE_SIZE to P0_Virt
    Endfor
    Increment Nproc
Endloop
Move Nproc to Nproc field of ILA
Return
```

#1500-15-031.02.0

3.3.10.3.4.18.5 Limitations

The Read_Process_List unit must be called before this unit.

3.3.10.3.4.19 Reserve_SPT Unit

The Reserve_SPT Unit reserves a predefined number of pages following the SCB to contain the system page table. This is accomplished by setting the System Base Register (SBR) and System Length Register (SLR) to indicate the start and length of the system page table.

3.3.10.3.4.19.1 Inputs

The following inputs are defined for the Reserve_SPT Unit:

- 1) Free_Phys - This input is read from global data and contains the free physical memory address where the system page table is placed.

#1500-15-031.02.0

3.3.10.3.4.19.2 Outputs

The following outputs are produced by the Reserve_SPT Unit:

- 1) Free_Phys - This output is written to global data and contains the updated address of the next free physical memory address.
- 2) Sys_PT - This output is written to global data and contains the physical address of the system page table.
- 3) Sys_Len - This output is written to global data and contains the number of PTEs in the system page table.

3.3.10.3.4.19.3 Local Data

No local data is defined for the Reserve_SPT Unit.

3.3.10.3.4.19.4 Processing

```
Move Free_Phys to Sys_PT
Move SPT_LENGTH to Sys_Len
Add SPT_LENGTH * 4 to Free_Phys
Call lseek(IGW_Image, SPT_LENGTH, L_INCR)
Return
```

#1500-15-031.02.0

3.3.10.3.4.19.5 Limitations

This unit must be called immediately after the Load_SCB unit.

3.3.10.3.4.20 Write_ILA Unit

The Write_ILA Unit will write the ILA to the IGW_Image file.

3.3.10.3.4.20.1 Inputs

The following inputs are defined for the Define_ILA Unit:

- 1) ILA - This input is obtained from global data and contains the IGW ILA.
- 2) ILA_Phys - This input is obtained from global data and contains the physical address of the ILA for the IGW.

3.3.10.3.4.20.2 Outputs

The following outputs are produced by the Define_ILA Unit:

- 1) IGW_Image - This output is written to the IGW_Image file and is loaded with the ILA.

#1500-15-031.02.0

3.3.10.3.4.20.3 Local Data

No local data is defined for the Define_ILA Unit.

3.3.10.3.4.20.4 Processing

Call lseek(IGW_Image, ILA_Phys, L_SET)
Call write(IGW_Image, ILA, Size of ILA)

3.3.10.3.4.20.5 Limitations

No limitations are defined for the Define_ILA Unit.

3.3.11 SUPPORT SOFTWARE Detailed Design

The support software which must be created for the IGW consists of Makefile scripts and the program Write_Diskettes. The Makefile scripts will be used to build all executable images for the IGW software. Additionally, the Makefile scripts will also be used to produce listings of software, and to prepare bootable floppies. The preparation of these Makefile scripts will be completed during the coding phase of IGW development.

The Write_Diskettes program is called by the Makefile scripts to produce bootable IGW diskettes. The program produces the diskettes required for booting entirely from floppy disks. The diskettes contain all the IGW and IXIB software required to completely boot the

#1500-15-031.02.0

IGW. This program is detailed in the following subsections.

3.3.11.1 Write_Diskettes Architecture

The Write_Diskettes program writes configuration files and images to two RX50 diskettes which are required to boot an IGW. Write_Diskettes consists of the following units:

- 1) Main - This unit is the control unit which prompts the user to confirm that the diskettes are to be written and then calls the units to write the primary boot information, the secondary boot image, the directory list, and the bootable files and images to the diskettes.
- 2) Add_Files - This unit adds the file name and the file size in bytes of each file to the directory list.
- 3) Blocks - This unit converts a byte count to the number of blocks required on a RX50 diskette.
- 4) Directory_List - This unit calls Add_Files to build the directory list and then assigns the starting block number and the floppy drive number for each entry in the list.
- 5) Init - This unit confirms the intent to write the diskettes. When the write is to occur, the diskettes are opened for writing.
- 6) Transfer - This unit copies a file from a specified source directory to a specified diskette.

#1500-15-031.02.0

3.3.11.2 Write_Diskettes Global Data

The global data for the Write_Diskettes program is listed as follows:

- 1) PRIMARY_BOOT - This global data item is an array of bytes which contain the full pathname of the primary boot block information. This global data item will be defined during coding.
- 2) SECONDARY_BOOT - This global data item is an array of bytes which contains the full pathname of the secondary boot image. This global data item will be defined during coding.
- 3) DIR_NAME - This global data item is an array of bytes which contains the pathname to the directory of configuration files and bootable images as a null terminated character string. This global data item will be defined during coding.
- 4) DIR_SIZE (30) - This global data item is a 32 bit integer which contains the maximum number of directory entries in a directory list.
- 5) BLK_SIZE (512) - This global data item is a 32 bit integer which contains the number of bytes in a block on a RX50 diskette.
- 6) DISK_SIZE (800) - This global data item is a 32 bit integer which contains the total number of blocks on a RX50 diskette.
- 7) DISK1 - This global data item is an array of bytes which contains the full pathname of the first floppy drive as a null terminated character string. The definition of this global data item depends on the device names of the support system.
- 8) DISK2 - This global data item is an array of bytes which contains the full pathname of the second floppy drive as a null terminated character string. The definition of this global data item depends on the device names of the support system.

#1500-15-031.02.0

3.3.11.3 Write_Diskettes LLC Design

The Write Diskettes program does not contain any lower level components.

3.3.11.4 Write_Diskettes Unit Design

The following sections describe the units of the Write_Diskettes program.

3.3.11.4.1 Add_Files

Add_Files adds the file name and the size of the file in bytes of each file in a specified directory into the directory listing. An error is returned when all files cannot be added to the directory list.

3.3.11.4.1.1 Inputs

The following inputs are required by the Add_Files unit:

- 1) Dir_Name - This input parameter is a pointer to a character string which contains the full pathname of the target directory.
- 2) DIR_SIZE - This global data input is a 32 bit integer which contains the maximum number of directory entries in Disk_Dir.

#1500-15-031.02.0

3.3.11.4.1.2 Outputs

The following outputs are returned by the Add_Files unit:

- 1) Disk_Dir - This output parameter is a table of at most DIR_SIZE directory entries for the files which will be contained on the IGW diskettes. Each directory entry contains the following fields:
 - Dir_Name - This field consists of an array of 15 bytes containing the file name stored as a null terminated character string.
 - Dir_Dev - This field consists of a single byte indicating the floppy drive number that the file is stored on.
 - Dir_BN - This field consists of a 16 bit integer containing the starting block number of the file on the diskette.
 - Dir_Size - This field consists of a 32 bit integer containing the size of the file in bytes.

- 2) Added_Entries - This output function value is a 16 bit integer which indicates the number of directory entries added to Disk_Dir.

#1500-15-031.02.0

3.3.11.4.1.3 Local Data

The following local data is defined for the Add_Files unit:

- 1) Dir_Ptr - This local data item is a pointer to a directory which is opened for reading.
- 2) Dir_Struct - This local data item is a structure defined by the DIR.H include file which identifies by name the files in the directory pointed to by Dir_Ptr. The directory structure includes the following fields:
 - D_Namlen - This field consists of a 16 bit integer which defines the number of characters in the directory file name.
 - Dname - This field consists of an array of 255 bytes which contains the name of the file in the directory.
- 3) Stat_Buf - This local data item structure is defined by the STAT.H include file which identifies characteristics of a file. The buffer includes the following fields:
 - St_Mode - This field is a 16 bit integer which contains a set of flags describing the file. S_IFMT is used to determine the type of file, in particular:
 - S_IFDIR (0x4000) - indicates the file is a directory
 - St_Size - This field consists of a 32 bit integer which contains the size of the file.

#1500-15-031.02.0

3.3.11.4.1.4 Processing

```
Set the Added_Entries to zero
Dir_Ptr = Opendir( Dir_Name )
Dir_Struct = Readdir( Dir_Ptr )
While Dir_Struct is not NULL
```

```
    /* Process the filename if the file name is not a directory */
```

```
    Call Stat( Dir_Name, address of Stat_Buf )
    If St_Mode and S_IFMT is not equal to S_IFDIR
        Increment Added_Entries
        If Added_Entries exceeds the maximum entries DIR_SIZE
            Return( ERROR )
        Endif
```

```
    /* Set the file name and file size in bytes for the entry */
```

```
    Parse the file name from the D_Namlen bytes in D_Name
    Set the Dir_Name field in the Disk_Dir entry Added_Entries to
    the parsed file name
    Set the Dir_Size field in the Disk_Dir entry Added_Entries to
    the St_Size field in Stat_Buf
```

```
Endif
```

```
/* Read the next file name in the directory */
```

```
Dir_Struct = Readdir( Dir_Ptr )
```

```
Endwhile
Call Closedir( Dir_Ptr )
Return( Added_Entries )
```

#1500-15-031.02.0

3.3.11.4.1.5 Limitations

No limitations are defined for the Add_Files unit.

3.3.11.4.2 Blocks

Blocks determines the number of blocks required on a diskette given both the size of the file in bytes and the size of a block in bytes.

3.3.11.4.2.1 Inputs

The following inputs are required by the Blocks unit:

- 1) File_Size - This input parameter is a 32 bit integer which contains the size of the file in bytes.
- 2) Blk_Size - This input parameter is a 32 bit integer which contains the size of a diskette block in bytes.

3.3.11.4.2.2 Outputs

The following outputs are returned by the Blocks unit:

- 1) Block_Count - This output function value is a 32 bit integer which contains the number of blocks the file requires on the diskette.

#1500-15-031.02.0

3.3.11.4.2.3 Local Data

No local data is defined for the Blocks unit.

3.3.11.4.2.4 Processing

Set the Block_Count equal to the File_Size divided by the Blk_Size The remainder equals the File_Size less the Block_Count times the Blk_Size If the remainder is not zero Increment the Block_Count Endif Return(Block_Count).

3.3.11.4.2.5 Limitations

No limitations are defined for the Blocks unit.

3.3.11.4.3 Directory_List

Directory_List compiles the list of configuration files and bootable images in a directory listing. For each entry in the list, a floppy drive and the starting block number are assigned. A message is printed to standard output and an error returned when the floppy drive requirements are exceeded.

#1500-15-031.02.0

3.3.11.4.3.1 Inputs

The following inputs are required by the Directory_List unit:

- 1) Block_Count - This input parameter is a 32 bit integer which contains the number of diskette blocks used on the first floppy drive.
- 2) DIR_NAME - This global data input is an array of bytes which contains the pathname to the directory of configuration files and bootable images as a null terminated character string.
- 3) BLK_SIZE - This global data input is a 32 bit integer which contains the number of bytes in block on a RX50 diskette.
- 4) DISK_SIZE - This global data input is a 32 bit integer which contains the total number of blocks on a RX50 diskette.

3.3.11.4.3.2 Outputs

The following outputs are returned by the Directory_List unit:

- 1) Disk_Dir - This output parameter is a table of at most DIR_SIZE directory entries for the files which will be contained on the IGW diskettes. Each directory entry contains the following fields:
 - Dir_Name - This field consists of an array of 15 bytes containing the file name stored as a null terminated character string.
 - Dir_Dev - This field consists of a single byte indicating the floppy drive number that the file is stored on.
 - Dir_BN - This field consists of a 16 bit integer containing the starting block number of the file on the diskette.

#1500-15-031.02.0

- Dir_Size - This field consists of a 32 bit integer containing the size of the file in bytes.

- 2) Entry_Count - This output function value is a 16 bit integer which indicates the success of updating the directory list. The positive integer Added_Entries indicates the number of entries added, otherwise the error indication is given:

ERROR - This negative integer value indicates not all entries were added to the directory list.

3.3.11.4.3.3 Local Data

The following local data is defined for the Directory_List unit:

- 1) Dir_Ptr - This local data item is a pointer to a directory which is opened for reading.
- 2) Format_String - This local data item is a pointer to a character string which contains information prompts to the operator on the number of diskettes either used or required.

3.3.11.4.3.4 Processing

```
/* Compile file names and sizes of all configuration files and
bootable images */
```

```
Entry_Count = Add_Files( address of DIR_NAME, address of Disk_Dir )
If Entry_Count is equal to ERROR
    Return( Entry_Count )
Endif
```

```
/* Linearly sort the file names by size from smallest to largest */
```

```
For Index which indexes the first Entry_Count - 1 Disk_Dir entries
    For Jindex which indexes the Disk_Dir entries subsequent to Index
        If the Dir_Size of entry Index exceeds Dir_Size of entry Jindex
            Temp entry = Disk_Dir entry Index
```

#1500-15-031.02.0

```
        Disk_Dir entry Index = Disk_Dir entry Jindex
        Disk_Dir entry Jindex = temp entry
    Endif
Endfor
Endfor

/* Compute the starting block number and floppy drive components
   for each file name, accounting for the blocks required to store
   the directory list */

Dir list size = Entry_Count * size of a Disk_Dir directory entry
Block_Count = Block_Count + Blocks( dir list size, BLK_SIZE )
Set the floppy drive to the integer one
For each of the Entry_Count directory entries in Disk_Dir
    Entry size = Blocks( Dir_Size, BLK_SIZE )
    If the Block_Count + entry size exceeds the DISK_SIZE
        If the floppy drive has already been set to two
            Call Printf( Format_String )
            Return( ERROR )
        Endif
        Reset the Block_Count to one
        Reset the floppy drive to the integer two
    Endif
    Set Disk_Dir Dir_BN equal to the Block_Count
    Set the Disk_Dir Dir_Dev equal to the floppy drive
    Block_Count = Block_Count + entry size
Endfor
If the floppy drive is still one
    Call Printf( Format_String )
Endif
Return( CONTINUE )
```

#1500-15-031.02.0

3.3.11.4.3.5 Limitations

No limitations are defined for the Directory_List unit.

3.3.11.4.4 Init

Init confirms the intent to write the diskettes. When the write is to occur, the diskettes are opened for writing.

3.3.11.4.4.1 Inputs

The following inputs are required by the Init unit:

- 1) DISK1 - This global data item is an array of bytes which contains the full pathname of the first floppy drive as a null terminated character string.
- 2) DISK2 - This global data item is an array of bytes which contains the full pathname of the second floppy drive as a null terminated character string.

#1500-15-031.02.0

3.3.11.4.4.2 Outputs

The following outputs are returned by the Init unit:

- 1) Disk_Descr_1 - This output parameter is a 32 bit integer which contains a file descriptor to which the source file Filename is copied.
- 2) Disk_Descr_2 - This output parameter is a 32 bit integer which contains a file descriptor to which the source file Filename is copied.
- 3) Status - This output parameter is a 16 bit integer which indicates the intent to write the diskettes:

CONTINUE - The diskettes are written
ERROR - The diskettes are not written

3.3.11.4.4.3 Local Data

The following local data is defined for the Init unit:

- 1) Flags - This local data item is a 32 bit integer which indicates that both diskettes are opened for write only.
- 2) Format_String - This local data item is a pointer to a character string which contains input/output prompts used to confirm the rewrite of the diskettes.
- 3) Response - This local data item is a pointer to a character string which contains the confirmation from the operator.

#1500-15-031.02.0

3.3.11.4.4.4 Processing

Set the first diskette descriptor Disk_Descr_1 to zero
Set the second diskette descriptor Disk_Descr_2 to zero

```
/* Prompt operator for confirmation of writing diskettes */
```

```
Call Printf( Format_String )  
Call Scanf( Format_String, Response )  
If the first character in the Response is not "Y" or "y"  
    Call Printf( Format_String )  
    Status = ERROR  
    Return( Status )  
Endif
```

```
/* Open both diskettes for writing */
```

```
Status = CONTINUE  
Flags = O_WRONLY  
Disk_Descr_1 = Open( address of DISK1, Flags )  
Disk_Descr_2 = Open( address of DISK2, Flags )  
Return( Status )
```

3.3.11.4.4.5 Limitations

No limitations are defined for the Init unit.

3.3.11.4.5 Main

The Main unit initiates the update of the IGW diskettes and, upon confirmation by the operator, copies bootable images and the configuration files to the two available diskettes.

The diskette in floppy drive one is used first. The first block contains primary boot information. Subsequent blocks contain the

#1500-15-031.02.0

secondary boot image, a disk directory of subsequent files and images, configuration files and bootable images. The number of blocks required for each section following the primary boot block is dynamically determined and, as a result, may require the diskette in the second floppy drive.

3.3.11.4.5.1 Inputs

The following inputs are required by the Main unit:

- 1) PRIMARY_BOOT - This global input parameter is an array of bytes which contains the full pathname of the primary boot file information as a null terminated character string.
- 2) SECONDARY_BOOT - This global data item is an array of bytes which contains the full pathname of the secondary boot image as a null terminated character string.
- 3) BLK_SIZE - This global data item is a 32 bit integer which contains the number of bytes in a block on a RX50 diskette.
- 4) DIR_NAME - This global data item is an array of bytes which contains the pathname to the directory of configuration files and bootable images as a null terminated character string.

#1500-15-031.02.0

3.3.11.4.5.2 Outputs

No outputs are returned by the Main unit.

3.3.11.4.5.3 Local Data

The following local data is defined for the Main unit:

- 1) Disk_Descr_1 - This local data item is a 32 bit integer containing a descriptor which identifies floppy drive one.
- 2) Disk_Descr_2 - This local data item is a 32 bit integer containing a descriptor which identifies floppy drive two.
- 3) Stat_Buf - This local data item structure is defined by the STAT.H include file which identifies characteristics of a file. The buffer includes the following fields:
 - St_Size - This field consists of a 32 bit integer which contains the size of the file.
- 4) Disk_Dir - This local data item is a table of at most DIR_SIZE directory entries for the files which will be contained on the IGW diskettes. Each directory entry contains the following fields:
 - Dir_Name - This field consists of an array of 15 bytes containing the file name stored as a null terminated string.
 - Dir_Dev - This field consists of a single byte indicating the floppy drive number that the field is stored on.
 - Dir_BN - This field consists of a 16 bit integer containing the starting block number of the file on the diskette. - Dir_Size - This field consists of a 32 bit integer containing the size of the file in bytes.

#1500-15-031.02.0

- 5) Filename - This local data item is an array of bytes which contains the full pathname of a file to be transferred to a diskette as a null terminated character string.
- 6) Block_Count - This local data item is a 32 bit integer which contains the next diskette block in which data can be written.
- 7) Byte_Count - This local data item is a 32 bit integer which contains the number of diskette bytes to be skipped.
- 8) L_SET (0x00) - This local data item is a 32 bit integer which defines the seek mode to be from the start of the file.
- 9) L_INCR (0x01) - This local data item is a 32 bit integer which defines the seek mode to be from the current seek position.

3.3.11.4.5.4 Processing

```
/* Initialize both floppy drives */
```

```
Call Init( Disk_Descr_1, Disk_Descr_2 )  
Block_Count = 1  
Byte_Count = ( Block_Count - 1 ) * BLK_SIZE
```

```
/* Copy the primary boot information to the first block */
```

```
Call Lseek( Disk_Descr_1, Byte_Count, L_SET )  
Call Stat( address of PRIMARY_BOOT pathname, address of Stat_Buf )  
Call Transfer( address of PRIMARY_BOOT pathname, St_Size, Disk_Descr_1 )  
Block_Count = Blocks( St_Size, BLK_SIZE )  
Byte_Count = ( Block_Count - 1 ) * BLK_SIZE
```

```
/* Copy the secondary boot image */
```

```
Call Lseek( Disk_Descr_1, Byte_Count, L_SET )  
Call Stat( address of SECONDARY_BOOT pathname, address of Stat_Buf )  
Call Transfer( address of SECONDARY_BOOT pathname, St_Size, Disk_Descr_1 )  
Block_Count = Blocks( St_Size, BLK_SIZE )  
Byte_Count = ( Block_Count - 1 ) * BLK_SIZE
```

```
/* Compile the directory listing of configuration files and bootable  
images and write to the diskette. */
```


#1500-15-031.02.0

```
Call Lseek( Disk_Descr_1, Byte_Count, L_SET )
Entry count = Directory_List( address of Disk_Dir, Block_Count )
If the entry count is not ERROR
  Dir list size = entry count * size of a Disk_Dir directory entry
  Call Write( Disk_Descr_1, Disk_Dir, dir list size )
  Byte_Count = Blocks( dir list size, BLK_SIZE ) * BLK_SIZE
  Call Lseek( Disk_Descr_1, Byte_Count - dir list size, L_INCR )

  /* Write each item to the diskette on the appropriate
     floppy drive. */

  For each of the entry count entries in Disk_Dir
    Concatenate DIR_NAME and the Disk_Dir Dir_Name to Filename
    Byte_Count = Blocks( Dir_Size, BLK_SIZE ) * BLK_SIZE - Dir_Size
    If Dir_Dev equals 1
      Call Transfer( address of Filename, Dir_Size, Disk_Descr_1 )
      Call Lseek( Disk_Descr_1, Byte_Count, L_INCR )
    Else
      Call Transfer( address of Filename, Dir_Size, Disk_Descr_2 )
      Call Lseek( Disk_Descr_2, Byte_Count, L_INCR )
    Endif
  Endfor
Endif

/* Release the diskettes on both the floppy drives */

Call Close( Disk_Descr_1 )
Call Close( Disk_Descr_2 )
End
```

3.3.11.4.5.5 Limitations

No limitations are defined for the Main unit.

#1500-15-031.02.0

3.3.11.4.6 Transfer

Transfer copies a file from the specified source directory to the specified diskette.

3.3.11.4.6.1 Inputs

The following inputs are required by the Transfer unit:

- 1) Filename - This input parameter is a pointer to a character string which contains the full pathname of the source file.
- 2) Size - This input parameter is a 32 bit integer which contains the number of bytes in the source file Filename.
- 3) Disk_Descr - This input parameter is a 32 bit integer which containing a descriptor of the floppy drive to which the source file Filename is copied.

3.3.11.4.6.2 Outputs

No outputs are returned by the Transfer unit.

#1500-15-031.02.0

3.3.11.4.6.3 Local Data

The following local data is defined for the Transfer unit:

- 1) Buf - This local data item is an array of bytes into which the source file Filename is read.
- 2) File_Descr - This local data item is a 32 bit integer which is used to reference the source file Filename in the Open, Read, Write, and Close file operations.
- 3) Flags - This local data item is a 32 bit integer which indicates the source file Filename is opened for read only.

3.3.11.4.6.4 Processing

```
Flags = O_RDONLY
File_Descr = Open( Filename, Flags )
Call Read( File_Descr, Buf, Size )
Call Write( Disk_Descr, Buf, Size )
Call Close( File_Descr )
Return
```

3.3.11.4.6.5 Limitations

No limitations are defined for the Transfer unit.

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
LLC	- Lower Level Component
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
UDP	- User Datagram Protocol
XDD	- X.25 Device Driver
XON/XOFF	- Flow control on a serial line

SOFTWARE DETAILED DESIGN
DOCUMENT FOR THE INTER-
NETWORK GATEWAY PROJECT

QA
76.9
S88
S6474
1988
v.5

INDUSTRY CANADA / INDUSTRIE CANADA



208853