TOPS–10/TOPS–20 RSX–20F System Reference Manual

AA–BS94A–TK, AD–BS94A–T1

April 1986

This reference manual describes RSX–20F, the operating system that runs on the PDP–11/40 front-end processor of KL-based computers. RSX–20F loads the KL microcode, configures main and cache memory, loads the boot program, and performs diagnostics. For systems that are running TOPS–20, as well as for 1091 and 1095 systems, RSX–20F also provides device handling for unit record equipment.

This manual updates the *TOPS–10/TOPS–20 RSX–20F System Reference Manual*, order number AA–BS94A–TK.

			1090	1091/1095	2060/2065
OPERATING SYSTEM:		TOPS–10	REL. 7.03	REL. 7.03	
		TOPS–20			REL. 6.1
SOFTWARE:	RSX–20F	(TOPS–10)	VA15-50	VE15-50	
	RSX–20F	(TOPS–20)			VB15-50

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

Northeast/Mid–Atlantic Region

Digital Equipment Corporation
PO Box CS2008
Nashua, New Hampshire 03061
Telephone:(603)884–6660

Central Region

Digital Equipment Corporation
Accessories and Supplies Center
1050 East Remington Road
Schaumburg, Illinois 60195
Telephone:(312)640–5612

Western Region

Digital Equipment Corporation
Accessories and Supplies Center
632 Caribbean Drive
Sunnyvale, California 94086
Telephone:(408)734–4915

digital equipment corporation • marlboro, massachusetts

First Printing, February 1984
Updated, April 1986

Copyright ©1984, 1986 by Digital Equipment Corporation. All Rights Reserved.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

The following are trademarks of Digital Equipment Corporation:

DEC	MASSBUS	RSX
DECmate	PDP	RT
DECsystem-10	P/OS	UNIBUS
DECSYSTEM-20	Professional	VAX
DECUS	Q-BUS	VMS
DECwriter	Rainbow	VT
DIBOL	RSTS	Work Processor

digital

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

CONTENTS

TABLES

PREFACE

The RSX-20F System Reference Manual contains information describing the RSX-20F front-end operating system. RSX-20F runs on a PDP-11/40 front-end processor and communicates with either a TOPS-10 or TOPS-20 operating system running on a KL main processor.

The audience for this manual comprises Software Support Specialists, Field Service personnel, systems programmers, and system operators. It is assumed that the reader is familiar with PDP-11 hardware, RSX-11 operating systems, and either TOPS-10 or TOPS-20.

This manual does not contain everything anyone would like to know about RSX-20F. The information contained here was included because it seemed to be especially important and useful to the largest part of the audience. Hopefully, this information will prevent some users from having to place calls to a central DIGITAL installation when they need help. The information in the manual is organized as follows:

The first three chapters introduce the PDP-11 hardware, the RSX-11-based operating system, and the Files-11 file structure. A short glossary of RSX-20F terms and acronyms is also included.

Chapters 4 through 6 contain a description of the PARSER (the front-end command processor) and of KLINIT (the KL initialization software), as well as operating instructions for the front-end utility programs.

Chapters 7 and 8 contain information about the resident RSX-20F monitor and the nonresident system tasks. Communications between the KL and the PDP-11 using the DTE20 are discussed in detail. The handling of terminals, both direct lines and dial-up lines, is also described.

Chapters 9 and 10 contain information on detecting and debugging errors.

The remainder of the manual, the appendixes, contain several topics. Included is a list of system error messages. The procedure for transferring files between the KL and the PDP-11 is described, and the front-end system tasks are listed. The dialog involved in setting up a KLINIK window for remote diagnostics is discussed. There is also a list of information to include with RSX-20F Software Performance Reports and hot line calls, as well as a table of EIA pin definitions. The following TOPS-10 and TOPS-20 manuals also contain information about RSX-20F:

TOPS-10 Operator's Guide	AA-H283A-TB
TOPS-10 Monitor Installation Guide	AA-5056B-TB
TOPS-20 Operator's Guide	AA-4176D-TM
TOPS-20 Software Installation Guide	AA-4195G-TM
TOPS-10/TOPS-20 DDT11 Manual	AA-M494A-TK
DECsystem-10/DECSYSTEM-20 Processor Reference Manual	AA-H391A-TK

Readers who wish to become more familiar with PDP-11 hardware and software can find additional material in the following handbooks, which contain both tutorial and reference information:

PDP-11 Processor Handbook	EB 05138 76
PDP-11 Software Handbook	EB 09798 78
PDP-11 Peripherals Handbook	EB 05961 76
Terminals and Communications Handbook	EB 15486 79

CHAPTER 1

INTRODUCTION

Two PDP-11 operating systems, RSX-11M and RSX-11D, provided the base upon which RSX-20F was built. These operating systems were chosen because they offered the best base for building the required front end. The KL requires a front end that:

- Is small and efficient
- Can handle special cases such as booting the KL and diagnosing and recovering KL errors
- Can handle the unit record devices of TOPS-20 and TOPS-10

The purpose of the KL front end is to reduce the load on the KL. Specifically, the front end handles booting, configuring and loading the KL, and drives the unit-record and terminal hardware. Figure 1-1 shows a diagram of a KL-based computer system with a PDP-11 front end, including the connections that are present between the front end and the various peripherals that it drives.

This chapter contains important concepts of PDP-11 software, explains the needs of RSX-20F, and describes how RSX-11M and RSX-11D were modified to produce RSX-20F.

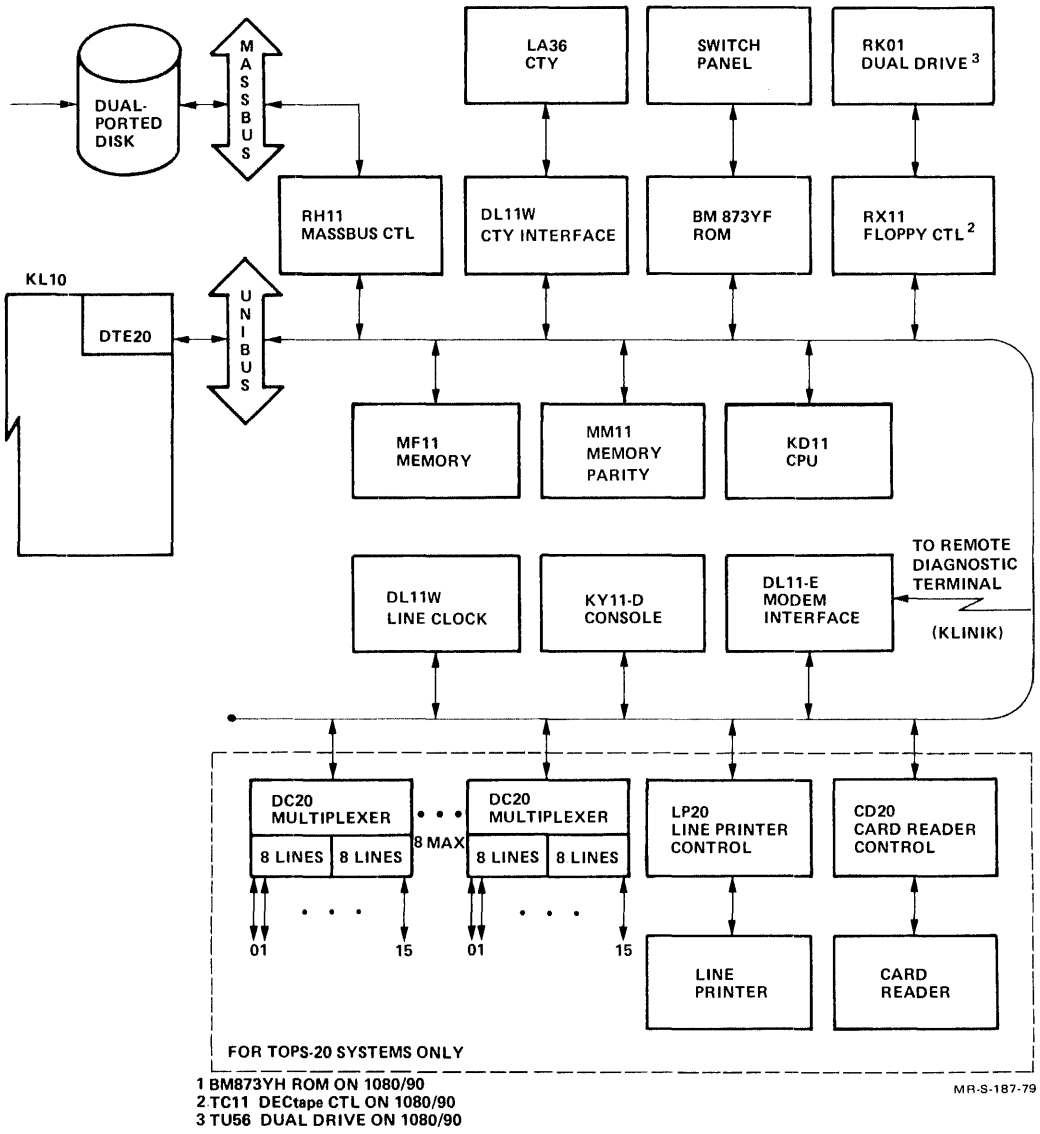


Figure 1-1: The Front End for a KL-based Computer System

1.1 THE PDP-11

The PDP-11 has several unique features that make it an easy machine to program and use. This section describes some of the most important of these features.

1.1.1 The UNIBUS

The UNIBUS is a 56-line bus used to send addresses, data, and control information to the system components and peripherals. The method of communication is the same for every device on the UNIBUS, including memory and the central processor. Each device, including memory locations, processor registers, and peripheral-device registers, is assigned an address on the UNIBUS. Thus, peripheral-device registers can be manipulated as easily as main memory by the central processor. The UNIBUS is both bidirectional and asynchronous; this allows devices of varying speeds to be connected to it.

1.1.2 The I/O Page

The I/O page is an area at the high end of memory in which all the physical device registers are assigned an address. The UNIBUS concept makes this I/O page easy to access and easy to keep current, thus making it useful to those who wish to find out about the physical state of the system.

1.1.3 Vector Interrupts

Vector interrupts allow the user to control interrupt handling as easily as depositing data into memory locations. Each device on the UNIBUS has two words assigned to it in low memory to handle its interrupts. The first word is the address of the routine to which control is to be relinquished in the event of an interrupt. The second word contains the processor status word to be installed when control is transferred to the interrupt routine.

1.1.4 Priorities

Interrupt priorities can be set individually for devices on the UNIBUS. Each device has a priority level on which it can interrupt. In the processor status word, the priority level field (bits 5-7) can be set to a value of 0 through 7. Only devices with a priority higher than the priority in the status word can interrupt. The user, therefore, can control interrupt priorities by depositing data into memory.

1.1.5 Traps

Both synchronous and asynchronous traps can be handled by the PDP-11. Synchronous traps occur immediately upon the issuance of an illegal instruction or general trap. They are dealt with by means of the vectors in low memory and provide user flexibility. Asynchronous traps occur independently of user instructions, usually as the result of I/O completion.

1.1.6 Data Transfers

Data can be transferred in two ways via the UNIBUS: a BR (Bus Request) or an NPR (Non-Processor Request). The method normally used is the BR, in which the device wanting to use the UNIBUS must first request the use of the bus from the bus master. An NPR parallels DMA (Direct Memory Access) on the KL. An NPR steals UNIBUS cycles without directly gaining control of the processor. Since it does not need to access the processor, an NPR is much faster than a BR.

1.1.7 General Registers

The central processor has eight registers (0-7) for general use. The registers can be used as accumulators, index registers, or stack pointers. Register 6 is used as the system stack pointer (SP), and Register 7 is used as the hardware program counter (PC). These last two registers can be manipulated in exactly the same way as the other registers, but depositing data in them destroys the state of the PDP-11, because the PDP-11 is not able to find the next instruction or the hardware stack.

1.1.8 Stacks

The PDP-11 is a stack-oriented machine. It contains built-in addressing modes designed to manipulate stacks of data. The PDP-11 also has its own system stack, and it uses R6 as the hardware stack pointer.

1.1.9 Instruction Set

The PDP-11 instruction set operates on single- or double-byte operands. Addressing on the PDP-11 is by eight-bit bytes. The word size is sixteen bits. Addressing includes a variety of addressing modes which, when combined with the instruction set, allow the programmer great flexibility in programming.

1.2 RSX-11M OPERATING SYSTEM

RSX-11M is a PDP-11 operating system. It controls I/O, schedules tasks to be run, and provides common subroutines. The resident operating system is referred to as the Executive.

1.2.1 Directives

A directive is a request to the Executive to perform a function. Directives can perform I/O functions, obtain task and system information, suspend and resume task execution, and cause a task to exit. Directives are called EMTs (EMulator Traps), and are equivalent to JSYSs in TOPS-20 and UUOs in TOPS-10.

1.2.2 Device Drivers

A device driver is a program that controls physical hardware activities on a peripheral device. The device driver is generally the device-dependent interface between a device and the common device-independent I/O code in an operating system.

1.2.3 Significant Events

A significant event is an event or condition that indicates a change in the system status of an event-driven system. A significant event is declared, for example, when an I/O operation completes. A declaration of a significant event indicates that the Executive should review the eligibility of task execution, because the event might have unblocked the execution of a higher priority task. The following are considered to be significant events:

- Queueing of I/O requests
- Completing of I/O requests
- Requesting a task
- Scheduling a task
- Waking up a task
- Exiting a task

There are 64 significant event flags, most of them directly related to servicing directives. These flags can also be used by tasks to communicate with other tasks.

1.2.4 Mapped and Unmapped Systems

A mapped system uses hardware memory management to relocate virtual memory addresses. An unmapped system has no hardware to relocate virtual addresses into physical addresses. An unmapped system must therefore be assembled with the correct physical addresses. RSX-20F is an unmapped system.

1.3 TASKS

A task is the fundamental executable unit in PDP-11 operating systems. Some tasks are self-sufficient and can be thought of in much the same way as programs in TOPS-10 or TOPS-20. Some tasks must call other tasks to complete their function. Some tasks are considered subroutines to be called by still other tasks.

Tasks can reside in one of three places: the resident Executive (EXEC) partition, the general (GEN) partition, or the Files-11 partition (F11TPD). A partition is an area of memory reserved for the execution of tasks. In the simplest case, a task uses all of the partition. If the task is smaller than the partition, the unused space is unavailable to other tasks. If a task is larger than the partition, it must be written to use overlays. Overlays are sections of code that are brought into memory as needed and are written over existing code that is no longer required.

Most tasks that run in the EXEC partition handle specific devices and system functions. These tasks are called resident tasks; that is, they are always core resident and are not swapped out. This is important because system functions and devices demand instantaneous service and should not have to wait for code to be read in from a peripheral device. Occasionally a task is larger than the partition into which it must fit. This situation can be handled by overlaying code. Overlaying consists of replacing unneeded sections of code in core with sections that are needed but are not currently in main memory.

There are two general classes of tasks: privileged and nonprivileged. A privileged task can access its own partition, the Executive partition, and the I/O page. A nonprivileged task can access only its own partition and shared regions.

When a task has been compiled, it is still not ready to be loaded and executed. It must be put through the Task Builder. A compiler produces an output file called an object module. The Task Builder accepts object modules as input, links them together, resolves references to global symbols and library files, and produces an output file called a task image. In the task image file, all relocatable expressions and external references have been converted to absolute addresses. The task image file can then be loaded into a partition and executed. The Task Builder can also produce a memory map file. A memory map describes the allocation of storage, itemizes the separate modules that the task comprises, and lists all global-symbol values.

1.4 RSX-20F REQUIREMENTS

The PDP-11/40 fulfills the normal functions of a front-end computer. It acts as a peripheral handler and data concentrator/router in its relation with the KL. The devices that it handles are the slower, unit record devices (TTY, CDR, and LPT). This allows the KL to concentrate on computing rather than servicing interrupts from the slower devices.

The front end can also be used for other special functions. For example, it can perform all the following steps necessary to get the KL up and running:

- Load the microcode
- Configure memory
- Configure cache
- Load a bootstrap program

The front end can also perform diagnostics on the KL when hardware problems develop.

1.5 THE DERIVATION OF RSX-20F FROM RSX-11M

RSX-11M is geared toward multiprogramming and quick response to real-time events. The multiprogramming capability allows the development and use of utility programs that can perform special tasks. The real-time response allows any attached devices to be serviced quickly. For these reasons, RSX-11M was chosen as the basis for RSX-20F.

RSX-20F utility programs can be run only in the GEN partition. Nonresident Exec routines (for example, Files-11, KLRING, KLDISC, SETSPD, TKTN, and MIDNIT) run in the F11TPD partition. Only one utility task can run at any one time in the GEN partition and that task runs until completion. Some tasks use overlays. These tasks must control their own overlaying, however, since the Executive makes no attempt to do so.

The significant event scheme of RSX-11M was kept in RSX-20F to handle changes in system states and to provide directives with information. The directives that were kept provide I/O service, task information and task control. The scheduling algorithm used to decide which task runs next is round robin within priority value.

Specific programs are brought into core to do special tasks. Some of the RSX-20F utility programs are MOUNT and DISMOUNT to control access to Files-11 devices, PIP to transfer files from one Files-11 device to another, UFD to create User File Directories on Files-11 devices, and PARSER to provide communication and diagnostic functions. All these tasks run in the GEN partition.

The biggest change in the structure of RSX-11M had to do with driving the DTE20 interface. The DTE20 is the only link between the front end and the KL, and provides the interface between the KL and the terminals, printers, and so forth. In order to deal with all the purposes to which the DTE20 would be put, the operating system needed a device driver. A queue mechanism had to be set up to handle all the requests for the devices that the KL receives and transmits to the front end. Consequently, the queued protocol task was added to handle the communication between TOPS-20 and the device drivers in the front end.

Although no inter-CPU communication can take place over the disk, the PDP-11 and the KL can access the dual-ported RP04/06 drive independent of each other. However, RSX-20F does not have access to the entire dual-ported disk; RSX-20F is limited to 950 pages by default (the value can be made larger by reinitializing the disk). Logical block number 400 is the home block for the Files-11 system. TOPS-20 views the front-end file system as one big file, <ROOT-DIRECTORY>FRONT-END-FILE-SYSTEM.BIN. TOPS-10 also views the front-end file system as one big file, SYS:FE.SYS.

System access to front-end files is usually done with file ID's. Because the front-end file system contains relatively few files, this access method can find the files quickly. The directory structure is kept for those few situations when users must interact with a Files-11 area on floppy disk, DECtape, or dual-ported RP04/06. No protection checking is enforced with the file systems.

Real PDP-11 formatted disks have 16-bit words, and disk addressing and accessing is consistent with this scheme. However, disks supported by TOPS-10 and TOPS-20 must be formatted in 18-bit words to make them compatible with the 36-bit word size expected by the KL processor. Therefore, the RSX-20F disk driver is a modified RSX-11M routine. Each PDP-11 word of data in the Files-11 area is written right-justified in the 18-bit space available. The two left-hand (high-order) bits are ignored by RSX-20F's disk driver.

CHAPTER 2

FILES-11 SYSTEM

All RSX-based operating systems have a standard file system called Files-11. Users who access files in an RSX-20F Files-11 system use a syntax that is similar to TOPS-20 and TOPS-10. This chapter defines some terms used by Files-11, and describes the file structure and directory structure used by the system.

2.1 GENERAL DEFINITIONS

The Files-11 system imposes a structure on a medium. The medium Files-11 uses is any block-addressable storage device. This includes such media as disks and DECtapes. Since the method of access to all Files-11 media is similar, all types of Files-11 media are referred to as disks.

A Files-11 volume is a logical file structure that includes one or more devices of the same type. A Files-11 volume can be compared to a file structure under TOPS-10 and TOPS-20.

When Files-11 devices are used by a task, each device is assigned a number called a Logical Unit Number (LUN). LUNs are associated with physical devices during a task's I/O operations. The Executive can also assign LUNs for its own use.

2.2 FILES-11 FILE SPECIFICATION

The file specification for Files-11 is:

```
dev:[g,m]filename.ext;version
```

where:

dev:	is the name of a physical or logical device on which the desired file is located. The device name consists of two ASCII characters followed by an optional one-digit unit number and a colon.
[g,m]	is the group number and member number associated with the User File Directory (UFD). These numbers are octal and are in the range of 1 to 777. This section of the file specification is also referred to as the User Identification Code (UIC).
filename	is the name of the file which can be from 1 to 9 alphanumeric characters.

ext	is the extension of the file which can be from 1 to 3 alphanumeric characters or null.
version	is the version number of the file which can range from 1 to 77777. If no version number is specified, the number defaults to the most recent version on a read operation and the next version number on a write operation.

By comparison, the TOPS-20 file specification format is:

```
dev:<directory>filename.type.gen
```

The TOPS-10 file specification format is:

```
dev:filename.ext[p,pn]
```

The quantity [g,m] is the directory number and corresponds to the directory name in TOPS-20 and the project-programmer number in TOPS-10.

Two examples of valid RSX-20F Files-11 file specifications are:

```
DB0:[5,5]KLINIK.TSK

DX1:[5,5]MIDNIT.TSK;1
```

2.2.1 Files-11 File Structure

Any data of interest on a Files-11 volume is contained in a file. A file is an ordered set of virtual blocks, a virtual block being an array of 512 eight-bit bytes. A file's virtual blocks are numbered from 1 to n, where n blocks have been allocated to the file. The number assigned to a virtual block is called a Virtual Block Number, or VBN. Each virtual block is mapped to a unique logical block on the volume. Virtual blocks can be processed in the same manner as logical blocks. Any array of bytes that is less than 65K in length ca be read or written, provided that the transfer starts on a virtual block boundary and that its length is a multiple of four.

Each file in a volume is uniquely identified by a file ID. A file ID is a binary value consisting of three PDP-11 words (48 bits). It is supplied by Files-11 when the file is created and used whenever the file is referenced. The three words contain:

1. File number - This number uniquely identifies the file on the volume.

2. File sequence number - This number identifies the current use of an individual file number on a volume. The file numbers are reused. Since the file number of a deleted file is available to be used again, the file sequence number is attached to distinguish the uses of the file number.

3. Relative Volume Number - This number must be zero. The location is reserved for the implementation of volume sets.

Each file on a Files-11 volume is described by a file header. The file header is a block that contains all the information necessary to access the file. The file header is contained in the volume's index file, not in the file itself. The file header is divided into four distinct areas:

1. Header Area - This area contains the file number and the file sequence number as well as the file's ownership and protection codes. This area also contains offsets to the other areas of the file header, thereby defining their size. Finally, the header area contains a user attribute area, in which the user can store a limited amount of data describing the file.

2. Ident Area - This area contains identification and accounting data about the file, including the primary name of the file, its creation date and time, its expiration date, and its revision count, date and time.

3. Map Area - This area describes the mapping of virtual blocks of the file to logical blocks of the volume. The area contains a list of retrieval pointers, each of which describes one logically contiguous segment of the file. The map area also contains the linkage to the next extension header of the file, if one exists.

4. End Checksum - This area, the last two bytes of the file header, contain a sixteen-bit additive checksum of the preceding 255 words of the file header. The checksum is used in the process of verifying that this block is a file header.

Since the file header has a fixed size while the file itself does not, a large file could require more space for its mapping information than is available. To provide for this contingency, Files-11 uses extension headers. An extension header is used to chain together file headers to provide enough space for the mapping information. The map areas link the headers together in order of ascending virtual block numbers.

2.3 FILES-11 DIRECTORIES

Directories are Files-11 files whose sole function is to associate file-name strings with file ID's. Since the file ID is unique to the file, the file ID can be used to locate the file directly in the Files-11 system. However, most users find it easier to deal with a group of files if the files can be named. This ease of use is the goal of the directory file. A directory file is an FCS (File Control Services) file consisting of fixed sixteen-byte records (see Section 2.5 for a description of FCS files). Each record is a directory entry describing a single file. Each entry contains the following data:

- File ID - The three-word binary ID of the file this entry represents (The file number portion of the file ID is zero when the entry is empty.)

- Name - The name of the file, stored as three words of three Radix-50 characters

- Type - The file type (known also as the extension), stored as one word of three Radix-50 characters

- Version - The file version number, stored in binary in one word

2.4 FIXED FILE ID'S

As with any file system, Files-11 maintains a data structure that it uses to control the file organization. The information that Files-11 needs are kept in files called known files because the system always knows about them. These files are created when a new volume is initialized. The files have fixed file ID numbers so that Files-11 can always find its own data. The files and their uses are described below.

1. Index File - The index file is file ID 1,1,0. It is listed in the Master File Directory (MFD) as INDEXF.SYS;1. The index file provides the means for identification of and initial access to a Files-11 volume. It also contains the access data for all files on the volume, including itself.

2. Storage Bitmap File - The storage bitmap file is file ID 2,2,0. It is listed in the MFD as BITMAP.SYS;1. This file is used to control the available space on a volume. It contains a storage control block with summary information about the volume, and the bitmap itself, which lists the availability of individual blocks.

3. Bad Block File - The bad block file is file ID 3,3,0. It is listed in the MFD as BADBLK.SYS;1. The bad block file is simply a file containing a list of all the known bad blocks on the volume.

4. Master File Directory - The master file directory is file ID 4,4,0. It is listed in the MFD (itself) as 000000.DIR;1. It lists the five known files, and all the user file directories for the volume.

5. Core Image File - The core image file is file ID 5,5,0. It is listed in the MFD as CORIMG.SYS;1. This file is the bootable RSX-20F system image file.

2.5 FCS FILE STRUCTURE

FCS stands for File Control Services, which is a user-level interface to Files-11. Its principal feature is a record control facility that allows sequential processing of variable-length records as well as sequential and random processing of fixed-length records. FCS uses the virtual block system provided by the basic Files-11 structure.

FCS treats every disk file as a sequentially numbered array of bytes. Records are given ordinal numbers starting with 1 for the first record in the file. A file consisting of fixed-length records can have records crossing block boundaries or not, depending on the setting of a flag in the file header area. If records do cross block boundaries, the records are simply packed end to end. Records of an odd length are padded with a byte of indeterminate contents. If records do not cross block boundaries, their size is limited to 512 bytes.

Variable-length records can be 32,767 bytes, unless records do not cross block boundaries, in which case the limit is 510 bytes. Each record is preceded by a two-byte binary count of the bytes in the record (the count does not include these two bytes). This byte count is always word-aligned, and padded with a null byte if necessary. A byte count of -1 signals the end of live data in a particular block. The next record in the file begins at the next block.

CHAPTER 3

RSX-20F GLOSSARY OF TERMS

This chapter includes definitions and expansions of several words, phrases, and acronyms used in the manual.

ACL
: Access Control List

ACP
: Ancillary Control Processor

ACK
: Affirmative ACKnowledgement - the reply that indicates that the receiver accepted the previous data block and that the receiver is ready to accept the next block of the transmission.

APR
: Arithmetic PRocessor

AST
: Asynchronous System Trap

ATL
: Active Task List

Auto-bauding
: The process by which the terminal software determines the line speed on a dial-up line.

Carrier
: The analog signal that carries data over telephone lines.

Carrier Transition
: A transition in the state of the carrier signal, either from "On" to "Off", or vice versa.

CC
: Condition Code

CKL
: ClocK List

Communications Region
: An area in KL memory that is used for coordinating status, preparing for byte transfer operations, and passing limited amounts of data. Both the KL and the PDP-11 have an Owned Communications Region in which they alone can write.

CUSP
: Commonly Used System Program

DEQUE
: Double Ended QUEue

DIC
: Directive Identification Code (0-127)

Deposit Region
: A region in KL memory that is accessed by the PDP-11 using Protected Deposits.

DH11
: Communications interface between the PDP-11 front end and up to sixteen terminals and sixteen modems. Specifically a DH11AD.

DM11
: Communications interface between the PDP-11 front end and the EIA modem control lines. The DM11 is used in the DH11AD to handle modem control for asynchronous terminal lines connected to common carrier facilities.

DPB
: Directive Parameter Block

DSW
: Directive Status Word

DTE20
: The hardware interface between the PDP-11 and the KL. DTE stands for Data Ten to Eleven.

DTL
: DTE20 List

DTR
: The signal used by a computer system to answer the phone ring from a remote user. DTR stands for Data Terminal Ready.

EBOX
: Part of the KL hardware that performs arithmetic and logical operations.

EMT
: EMulator Trap Instruction

EPT
: The area in KL memory that is reserved for use in transmission of data between processors. EPT stands for Executive Process Table.

Examine Region
: A region in KL memory that is accessed by the PDP-11 using Protected Examines.

External Page
: An area (4K) of real memory space (760000-777777) containing CPU and peripheral device control and status registers (also known as the I/O page).

FCP
: File Control Primitives

IRQ
: I/O Request Queue

ISR
: Interrupt Service Routine

KT11
: Hardware Memory Management Option

LBN
: Logical Block Number

Login
: The process of getting a KL to recognize a potential user (see also Logon)

Logon
: The process of getting a PDP-11 to recognize a potential user

LUN
: Logical Unit Number

MCR
: Monitor Console Routines

MFD
: Master File Directory

MRL
: Memory Request List

Normal Termination
: An error-free completion of a given task. The term Done is not used because, unlike a Done flag, a Normal Termination flag is not set if an error occurs. An error causes the Error Termination flag to be set.

NXM
: Non-eXistent Memory

Owned Area
: An area in the Communications Region that is for the use of the related processor. The related processor can read and write to and from this area.

Packet
: A group of bytes including data and control elements that is switched and transmitted as a composite whole.

Privileged Front End
: A PDP-11 attached to a KL by means of a DTE20 that can use the diagnostic bus and do unprotected deposits. A privileged front end can crash the KL.

Protected Examines/Deposits
: An Examine or Deposit that is range-checked by the KL. The relocation and protection for the Examine operation is separate from that for the Deposit operation. A privileged front end can override the protection checks; a restricted front end cannot override the protection checks. (See also Relative Address)

PUD
: Physical Unit Device Table

Relative Address
: An address specified by the PDP-11 software on a Protected Examine or Deposit. The address specified by the PDP-11 is relative to the Examine or Deposit Region, and runs from 0 to the maximum relative address (which is kept by the KL in the EPT). (See also Examine Region, Deposit Region, EPT)

Restricted Front End
: A PDP-11 that is attached to a KL by means of a DTE20 and cannot crash the KL if the KL hardware and software are working correctly. A restricted front end cannot use the diagnostic bus, and cannot read KL memory unless the KL has first set up the interrupt system to allow it.

RTS
: A signal sent from the Data Terminal Equipment (in this case the DTE20) to the Data Communications Equipment (DCE) to condition the DCE for transmission. Since all terminal communication is full-duplex, the local modem should always be ready to transmit when a user is dialed in. Thus, RTS should always be asserted by the PDP-11 for active dial-up lines. RTS stands for Request to Send.

Send-All
: Data that is sent to every active line on the system that has not refused it. An obvious example is a system message.

SPR
: Software Performance Report

SST
: Synchronous System Traps

STD
: System Task Directory

Thread
: The link word in a node.

TPD
: Task Partition Directory

UIC
: User Identification Code

UFD
: User File Directory

VCB
: Volume Control Block

VBN
: Virtual Block Number

CHAPTER 4

PARSER

The command language processor for the front-end operating system is called the PARSER. It is a nonresident system task and executes in the GEN partition when it is invoked. The PARSER is the primary means of communications between the system operator and the front-end programs. It also provides access to the KL's memory and diagnostic registers. The PARSER accepts input in the form of ASCII strings entered at the console terminal (CTY).

4.1 ENTERING AND EXITING THE PARSER

If you are currently communicating with the TOPS-10 or TOPS-20 monitor, or a TOPS-10 or TOPS-20 job, type a control backslash (CTRL/\) to enter the PARSER.

If you are currently communicating with another RSX-20F task or utility such as KLINIT or PIP, type a CTRL/Z to exit the current task and then a CTRL/\ to enter the PARSER.

When you enter the PARSER, you receive one of the following prompts:

PAR> This indicates that the PARSER is ready to accept commands and the KL is running (that is, the KL clock is running and the KL run flop is on).

PAR% This indicates that the PARSER is ready to accept commands but the KL microcode is in the HALT loop. (The KL clock is running but the KL run flop is off.)

PAR# This indicates that the PARSER is ready to accept commands but the KL clock is stopped and the KL is not running.

NOTE

If you see the PAR# prompt displayed during timesharing, you should reload the system.

If the PARSER encounters an error during its initialization, an error message precede the prompt.

In order to exit the PARSER, type QUIT or a CTRL/Z to return to the TOPS-10 or TOPS-20 monitor or use the PARSER command, MCR, to load and start another program.

4.2 PARSER COMMAND SYNTAX

Commands to the PARSER are typed one or more to a line in response to a PAR>, PAR%, or PAR# prompt. The rules that follow apply to all commands you wish to type unless you are explicitly told otherwise in the description of the command.

1. Multiple commands can be entered on a single line. To do this, separate each command from the following one by a semicolon. For example:

```
PAR>EXAMINE PC;EXAMINE 20;SHUTDOWN<RET>
```

2. Command lines can be continued on the following line. To continue a command line on the next line, end the line to be continued with a hyphen (-) and a carriage return. The PARSER will prompt you for the continuation line with another hyphen. For example:

```
PAR>EXAMINE PC;EXAMINE 20;-<RET>
-EXAMINE NEXT<RET>
```

   The maximum number of characters in a command line is 280.

3. A comment can be added to the end of a command line or can be an entry in itself. To insert a comment, begin the text with an exclamation mark (and end it with a carriage return). For example:

```
PAR>CLEAR CONSOLE!RESET TO OPERATOR MODE<RET>
PAR>!THIS IS A COMMENT LINE<RET>
```

4. Terminal output can be suppressed. To do this, type CTRL/O.

5. Keywords in a command can be truncated to their shortest unique abbreviation. For example:

```
PAR>H!HALT THE KL CPU<RET>
```

   If the truncation is not unique, you receive an error message. For example:

```
PAR>RE 5<RET>
PAR -- [PARSER] AMB - AMBIGUOUS KEYWORD "RE"
```

   In this example, the PARSER found two commands that started with RE: REPEAT and RESET.

6. The default radix of integers is octal if an address or 36-bit value is expected; otherwise the default radix is decimal.

7. Numbers can be shifted a specified number of binary places in either direction. To shift to the left, use an underscore (_) between the number you wish to shift and the number of binary places you wish it to be shifted. This causes the left-hand number to be shifted to the left by the number of binary bits indicated by the right-hand number, assuming that the right-hand number is positive. If the right-hand number is negative, the left-hand number is shifted to the right that many binary places. Thus, in order to specify a number in octal that ends in several zeros, you could write the

nonzero part, then an underscore, then the number of trailing (binary) zeros in the number. For example:

```
PAR>EXAMINE 2_3<RET>
```

results in

```
20/ xxxxxx xxxxxx
```

Note that rule #6 applies to both the left- and right-hand numbers.

8. Negative numbers can be specified through the use of a unary minus (-) preceding the number. For example:

```
PAR>DEPOSIT TEN 30=-1<RET>      (deposit -1 in
                                  loc. 30)
30/ xxxxxx xxxxxx                (previous
                                  contents)
PAR>EXAMINE TEN 30<RET>          (examine
                                  loc. 30)
30/ 777777 777777                (new contents)
```

9. Numeric values can be entered as arithmetic expressions using addition (+), subtraction (-), multiplication (*), and division (/). For example:

```
PAR>EXAMINE 123654+32<RET>
123706/ xxxxxx xxxxxx

PAR>DEPOSIT TEN 408-6=100<RET>

PAR>SET INCREMENT 2*3<RET>
KL INCREMENT: 6

PAR>REPEAT 8/4; EXAMINE PC<RET>
PC/ xxxxxx xxxxxx
PC/ xxxxxx xxxxxx
```

Note that in the evaluation of arithmetic expressions, multiplication, division, and binary shifts take precedence over addition and subtraction.

10. Relocation factors can be added or subtracted from a number. To do this, use a single quote (') following a number to add the PDP-11 relocation factor (offset) to the number. Use a double quote (") to subtract the PDP-11 relocation factor. For example:

```
PAR>SET OFFSET 101204<RET>
PDP-11 OFFSET: 101204
PAR>EXAMINE ELEVEN 32'<RET>
101236\ xxxxxx
PAR>EXAMINE ELEVEN 101236"<RET>
32\ xxxxxx
```

You can use the PDP-11 relocation factor to modify KL memory addresses as well as PDP-11 memory addresses.

When you close a command line (carriage return without a preceding hyphen), the PARSER first scans the command line buffer for illegal

characters. If it finds any, the entire command line is discarded and the following message is issued:

```
PAR -- [PARSER] ILC - ILLEGAL CHARACTER "c"
```

where "c" is the first illegal character found.

If the command line passes the character scan, the PARSER begins to execute the individual commands. If the PARSER encounters an invalid command, that command and any others remaining in the command line are not executed. The invalid command also generates an error message (refer to Section 4.5, PARSER Error Messages).

4.3 PARSER CONSOLE MODES

The PARSER command set differs according to the current console mode. There are three basic console modes:

OPERATOR MODE	This mode allows only those commands that will not crash the TOPS-10 or TOPS-20 monitor.
PROGRAMMER MODE	This mode allows all PARSER commands except diagnostic functions.
MAINTENANCE MODE	This mode allows the full set of PARSER commands.

In addition, there is a mode called USER MODE. Entering this mode has the effect of exiting the PARSER and is equivalent to a QUIT command.

When RSX-20F is initially loaded, the console mode is the mode that was in effect in the PARSER when the RSX-20F front-end module was saved. (Refer to Chapter 6 for a description of the SAV utility.) There is a SET CONSOLE command to change the console mode, a CLEAR CONSOLE command to reset the mode to OPERATOR, and a WHAT CONSOLE command to determine the current mode. These commands are explained in detail in Section 4.4.

4.3.1 PARSER Help Facility

The PARSER has a built-in help facility that prints out the available list of commands for the console mode you are in.

For an example, assume you are in OPERATOR mode and type:

```
PAR>?<RET>
```

The PARSER responds:

```
PARSER COMMANDS ARE:

ABORT
CLEAR
DISCONNECT
EXAMINE
JUMP
MCR
REPEAT
RUN
```

```
SET
SHOW
SHUTDOWN
TAKE
QUIT
WHAT
```

If, on the other hand, you are in PROGRAMMER mode, the response is:

```
PARSER COMMANDS ARE:

ABORT
CLEAR
CONTINUE
DEPOSIT
DISCONNECT
EXAMINE
HALT
INITIALIZE
JUMP
MCR
REPEAT
RESET
RUN
SET
SHOW
SHUTDOWN
START
TAKE
QUIT
WHAT
XCT
ZERO
```

This help facility extends to the argument level. If you are not sure of the arguments for a particular command, type the command followed by a space and a question mark.

For instance, assume you are in OPERATOR mode and type:

```
PAR>CLEAR ?<RET>
```

The PARSER responds:

```
CLEAR COMMANDS ARE:

CONSOLE
INCREMENT
KLINIK
MEMORY
NOT
OUTPUT
REPEAT
```

If instead you are in PROGRAMMER mode, the response is:

```
CLEAR COMMANDS ARE:

CONSOLE
DATE
FAULT-CONTINUATION
INCREMENT
```

```
KLINIK
MEMORY
NOT
OFFSET
OUTPUT
RELOAD
REPEAT
RETRY
TRACKS
```

Subarguments can also be determined in this manner. For example, if you type:

```
PAR>SET CONSOLE ?<RET>
```

The PARSER responds:

```
SET COMMANDS ARE:

MAINTENANCE
OPERATOR
PROGRAMMER
USER
```

4.4 PARSER COMMANDS

This section lists all PARSER commands. The console mode associated with each command specifies the minimal console mode at which the command is available. The following notational conventions apply to the command format:

- Any single argument not in brackets must be specified.
- Uppercase arguments are keywords and must be entered as shown or truncated according to rule 5 in Section 4.2.
- A multiple choice list enclosed in square brackets [ ] means that an entry is optional. If there is a default entry, it is specified.
- A multiple choice list enclosed in braces { } means that one of the entries must be specified.

In the following list of commands, those specified as requiring MAINTENANCE console mode should be restricted to Field Service personnel. Also, some commands require that the KL be stopped; this can be done with a HALT or ABORT command.

ABORT **OPERATOR**

The ABORT command stops the KL by trying to force it into the HALT loop. If this fails after a reasonable number of EBOX clock ticks, the command tries to START MICROCODE, which implies a MASTER RESET of the KL processor. This is one way to get the KL into a known state when a previous state left it in a hung condition.

CLEAR CLOCK { **NORMAL** | **CRAM** | **DATA-PATH** | **CONTROL** | **EXTERNAL** | **INTERNAL** | **MARGIN** | **FULL** | **HALF** | **QUARTER** | **SLOW** } **MAINTENANCE**

The CLEAR CLOCK command selectively resets the KL clock parameters. The CLEAR CLOCK arguments function as follows:

CLEAR CLOCK NORMAL	=	SET CLOCK NORMAL
CLEAR CLOCK CRAM		Disables the control-RAM clock
CLEAR CLOCK DATA-PATH		Disables the data-path clock
CLEAR CLOCK CONTROL		Disables the control logic clock
CLEAR CLOCK { EXTERNAL | INTERNAL | MARGIN | FULL | HALF | QUARTER | SLOW }	=	SET CLOCK NORMAL

CLEAR CONSOLE **OPERATOR**

The CLEAR CONSOLE command forces the PARSER into OPERATOR console mode. It is the equivalent of SET CONSOLE OPERATOR.

CLEAR DATE **PROGRAMMER**

The CLEAR DATE command clears the validity bit and prompts you for a new date and time (see SET DATE command). This command is not valid if RSX-20F is in primary protocol.

CLEAR FAULT-CONTINUATION **PROGRAMMER**

The CLEAR FAULT-CONTINUATION command disables the automatic KL fault continuation reload. This command prevents the host from recovering from errors detected by the front-end.

CLEAR FS-STOP **MAINTENANCE**

The CLEAR FS-STOP command disables the Field Service stop facility.

CLEAR INCREMENT **OPERATOR**

The CLEAR INCREMENT command resets the KL increment factor to zero. (See EXAMINE INCREMENT command.)

CLEAR KLINIK **OPERATOR**

The CLEAR KLINIK command closes the KLINIK access window and terminates the KLINIK link. (refer to Appendix D for a discussion on KLINIK access.)

CLEAR MEMORY **OPERATOR**

The CLEAR MEMORY command forces all subsequent EXAMINEs and DEPOSITs to reference KL memory. This command is the equivalent of the SET MEMORY TEN command. Note that this command does not set memory to zeros, or in fact to anything at all; it simply specifies which memory, the KL or the PDP-11, is being referenced.

CLEAR NOT **OPERATOR**

The CLEAR NOT command is the equivalent of the SET command.

CLEAR OFFSET **PROGRAMMER**

The CLEAR OFFSET command sets the relocation factor to zero. (refer to rule ten in Section 4.2.)

CLEAR OUTPUT [**LOG** | **LPT** | **TTY**] **OPERATOR**

This command stops the CTY output from going to the specified device. When the CLEAR OUTPUT [device] command is given, the PARSER checks for an active recording device. If as a result of the command, no recording device is active, then the console terminal is made the active recording device.

The default devices for the CLEAR OUTPUT command are LOG and LPT. If the default command is used the active recording device becomes TTY.

CLEAR PARITY STOP { **ALL** | **AR** | **CRAM** | **DRAM** | **ENABLE** | **FM** | **FS-STOP** } **MAINTENANCE**

The CLEAR PARITY-STOP command selectively disables parity stops for AR, CRAM, DRAM, Fast Memory, and Field Service.

CLEAR RELOAD **PROGRAMMER**

The CLEAR RELOAD command disables the automatic reload of the KL following a fatal error.

CLEAR REPEAT **OPERATOR**

The CLEAR REPEAT command resets the command line repeat factor to zero. A repeat factor of zero is the same as a repeat factor of one; subsequent command lines are executed once.

CLEAR RETRY **PROGRAMMER**

The CLEAR RETRY command resets the RETRY flag in the PARSER. When this flag is off, a Keep-Alive-Cease error causes the KLERR routine to take a system snapshot and then call KLINIT to perform a system reload of the KL. (See SET RETRY.)

CLEAR TRACKS **PROGRAMMER**

The CLEAR TRACKS command stops RSX-20F from typing all KL operations and results on the controlling terminal.

CONTINUE **PROGRAMMER**

The CONTINUE command takes the KL out of the HALT loop and starts execution at the instruction pointed to by the PC.

DEPOSIT AR=newdata **PROGRAMMER**

NOTE

The DEPOSIT AR command is now separated from the new data by an equal sign (=).

The DEPOSIT AR command sets the contents of the arithmetic register to new data.

DEPOSIT [**ELEVEN** / **TEN**] (**addr** / **DECREMENT** / **INCREMENT** / **NEXT** / **PREVIOUS** / **THIS**) **=newdata** **PROGRAMMER**

The DEPOSIT memory address command displays the contents of the specified or implied memory address and then replaces the contents with newdata.

ELEVEN specifies that the command is referencing an address in the PDP-11 memory.

TEN specifies that the command is referencing an address in the KL memory.

If neither ELEVEN or TEN is specified, the memory to be referenced is determined by the most recent SET MEMORY command.

If no SET MEMORY command has been issued, KL memory is referenced.

The following six arguments determine the specific memory address into which you wish to deposit the data; one of them must be entered.

addr is the actual memory address in octal notation. When referencing PDP-11 memory, this must be an even number.

INCREMENT means add the KL increment factor to the address last referenced to arrive at the deposit address. If PDP-11 memory is being referenced, this command is the equivalent of DEPOSIT NEXT.

DECREMENT means subtract the KL increment factor from the address last referenced to arrive at the deposit address. If PDP-11 memory is being referenced, this command is the equivalent of DEPOSIT PREVIOUS.

NEXT means add one (for a KL) or two (for a PDP-11) to the address last referenced to arrive at the deposit address.

PREVIOUS — means subtract one (for a KL) or two (for a PDP-11) from the address last referenced to arrive at the deposit address.

THIS — means use the address last referenced as the deposit address.

DISCONNECT — OPERATOR

The DISCONNECT command disconnects the KLINIK link by running the KLDISC task. This command does not clear any KLINIK parameters. (refer to Appendix D for a discussion of KLINIK.)

EXAMINE — OPERATOR

```
EXAMINE [ELEVEN]  {addr          }
        [TEN   ]  {addr1:addr2   }
                  {DECREMENT     }
                  {INCREMENT     }
                  {NEXT          }
                  {PREVIOUS      }
                  {THIS          }
```

The EXAMINE memory address command displays the contents of the specified or implied physical memory address in octal, on the CTY.

ELEVEN — specifies that the command is referencing an address in the PDP-11 memory.

TEN — specifies that the command is referencing a physical address in the KL memory.

If neither ELEVEN or TEN is specified, the memory to be referenced is determined by the most recent SET MEMORY command. If no SET MEMORY command has been issued, KL memory is referenced.

The following six arguments determine the specific memory address to be examined; one of them must be entered.

addr — is the actual memory address in octal notation. If you are referencing PDP-11 memory, this must be an even number.

addr1:addr2 — examines the memory addresses from addr1 through and including addr2.

INCREMENT — means add the KL increment factor to the address last referenced to arrive at the examine address. If PDP-11 memory is being referenced, this command is the equivalent of EXAMINE NEXT.

DECREMENT — means subtract the KL increment factor from the address last referenced to arrive at the examine address. If PDP-11 memory is being referenced, this command is the equivalent of EXAMINE PREVIOUS.

NEXT	means add one (for a KL) or two (for a PDP-11) to the address last referenced to arrive at the examine address.
PREVIOUS	means subtract one (for a KL) or two (for a PDP-11) from the address last referenced to arrive at the examine address.
THIS	means use the address last referenced as the examine address.

EXAMINE KL **OPERATOR**

The EXAMINE KL command performs the EXAMINE PC, EXAMINE VMA, EXAMINE PI, and the EXAMINE FLAGS commands, in that order.

EXAMINE PC **OPERATOR**

The EXAMINE PC command prints the contents of the KL program counter (22-bit PC) in octal, on the CTY.

EXAMINE AB **PROGRAMMER**

The EXAMINE AB command displays the contents of the KL address break register.

EXAMINE AD **PROGRAMMER**

The EXAMINE AD command displays the contents of the KL adder register.

EXAMINE ADX **PROGRAMMER**

The EXAMINE ADX command displays the contents of the KL adder extension.

EXAMINE AR **PROGRAMMER**

The EXAMINE AR command displays the contents of the KL arithmetic register.

EXAMINE ARX **PROGRAMMER**

The EXAMINE ARX command displays the contents of the KL arithmetic register extension.

EXAMINE BR **PROGRAMMER**

The EXAMINE BR command displays the contents of the KL buffer register.

EXAMINE BRX **PROGRAMMER**

The EXAMINE BRX command displays the contents of the KL buffer register extension.

EXAMINE CRADDR **PROGRAMMER**

The EXAMINE CRADDR command displays the contents of the KL CRAM address register.

EXAMINE CRLOC **PROGRAMMER**

The EXAMINE CRLOC command displays the contents of the KL CRAM location register.

EXAMINE DRADDR **PROGRAMMER**

The EXAMINE DRADDR command displays the contents of the KL DRAM address register.

EXAMINE DTE-20 **PROGRAMMER**

The EXAMINE DTE-20 command displays the contents of the DTE20 registers. These registers are displayed as in the following example:

```
DLYCNT: 000000
DEXWD3: 000447
DEXWD2: 000000
DEXWD1: 000000
   KL10 DATA=000000,,000447
TENAD1: 000000  TENAD2: 000007
   ADDRESS SPACE=EPT
   OPERATION=EXAMINE
   PROTECTION-RELOCATION IS ON
   KL10 ADDRESS=7
TO10BC: 010000  TO11BC: 130000
TO10AD: 066652  TO11AD: 066075
TO10DT: 000000  TO11DT: 000012
DIAG1 : 002400
   KL IN RUN MODE
   MAJOR STATE IS DEPOSIT-EXAMINE
DIAG2 : 040000
STATUS: 012104
   RAM IS ZEROS
   DEX WORD 1
   E BUFFER SELECT
   DEPOSIT-EXAMINE DONE
DIAG3 : 000000
```

EXAMINE EBR nnn **MAINTENANCE**

The EXAMINE EBR command reads Executive Process Table (EPT) entry nnn and examines the KL location given in that EPT entry.

EXAMINE EBUS **PROGRAMMER**

The EXAMINE EBUS command displays the contents of the KL EBUS register.

EXAMINE FE **PROGRAMMER**

The EXAMINE FE command displays the contents of the KL Floating Exponent register.

EXAMINE FLAGS **PROGRAMMER**

The EXAMINE FLAGS command displays the current state of the flag bits (0-12) in the left half of the PC word. Those flags are OVF, CY0, CY1, FOV, BIS, USR, UIO, LIP, AFI, AT1, AT0, FUF, and NDV.

EXAMINE FM **PROGRAMMER**

The EXAMINE FM command displays the contents of the KL Fast Memory register.

EXAMINE MQ **PROGRAMMER**

The EXAMINE MQ command displays the contents of the KL Multiplier Quotient register.

EXAMINE PI **PROGRAMMER**

The EXAMINE PI command displays the current state of the KL Priority Interrupt system. Each field displayed (PI HOLD, PI GEN, and PI ACTIVE) indicates the current state of PI 1 (leftmost of the 7-bits) to PI 7 (rightmost of the 7-bits) for that field.

EXAMINE REGISTERS **PROGRAMMER**

The EXAMINE REGISTERS command displays the contents of the following registers (see also the EXAMINE command for the individual registers):

AD, ADX, AR, ARX, BR, BRX, EBUS, FM, MQ, and PC.

EXAMINE SBR **PROGRAMMER**

The EXAMINE SBR command displays the contents of the KL Subroutine Return register.

EXAMINE SC **PROGRAMMER**

The EXAMINE SC command displays the contents of the KL Shift Count register.

EXAMINE UBR nnn **MAINTENANCE**

The EXAMINE UBR command reads User Process Table (UPT) entry nnn and examines the KL location given in that UPT entry.

EXAMINE VMA **PROGRAMMER**

The EXAMINE VMA command displays the contents of the KL Virtual Memory Address register.

EXAMINE VMAH **PROGRAMMER**

The EXAMINE VMAH command displays the contents of the KL Virtual Memory Address Held register.

FREAD nnn **MAINTENANCE**
nnn:nnn

The FREAD command performs a diagnostic function read of the KL CPU. The valid range of function codes (nnn) is 100 through 177 octal.

FWRITE nn=data **MAINTENANCE**

The FWRITE command performs a diagnostic function write to the KL CPU. The valid range of function codes (nn) is 40 through 77 octal. The data must be a 36-bit integer.

FXCT nn **MAINTENANCE**

The FXCT command performs a diagnostic function execute on the KL CPU. The valid range of function codes (nn) is 0 through 37 octal.

HALT **PROGRAMMER**

The HALT command tries to put the KL into the HALT loop by clearing the RUN flop (FXCT 10) and waiting. If the KL refuses to go into the HALT loop, the front end tries to force it in by using BURST mode. If this does not work, the following error message is issued:

```
PAR -- [HALT] CFH - CAN'T FIND KL HALT LOOP
```

INITIALIZE **PROGRAMMER**

The INITIALIZE command sets up the KL state flag word with default values and restarts the KL based on those values.

JUMP addr **OPERATOR**

The JUMP command starts the KL at the specified address and exits from the PARSER. At this point, the CTY is connected to the TOPS-10 or TOPS-20 operating system. The argument addr must be an octal, positive, nonzero address with a maximum value of 17777777.

MARK-MICROCODE n **MAINTENANCE**

The MARK-MICROCODE command sets the mark bit in the specified CRAM location. The n is the CRAM address in the range of Ø to 3777. The bit can be cleared by using the UNMARK-MICROCODE command.

MCR taskname **OPERATOR**

The MCR command loads and starts the specified task file.

QREST n=addr **MAINTENANCE**

The QREST command writes the contents of one of four reserved PDP-11 locations (n can be Ø,1,2 or 3) to the KL address given by addr.

QSAVE n=addr **MAINTENANCE**

The QSAVE command saves the contents of the KL address addr in one of four reserved PDP-11 locations (n can be Ø,1,2 or 3).

QUIT **OPERATOR**

The QUIT command causes the PARSER to be exited. At this point, the CTY is connected to the TOPS-1Ø or TOPS-2Ø operating system. This command is equivalent to SET CONSOLE USER or CTRL/Z.

THIS PAGE INTENTIONALLY LEFT BLANK

REPEAT nnn;[command1;command2;...] **OPERATOR**

The REPEAT command causes the subsequent commands in the current command line to be repeated the number of times specified by nnn. The argument nnn must be a positive, decimal, nonzero integer.

The command line can contain as many commands as will fit within the 280 character buffer limitation. You can nest REPEATs within the command line. Also, if a SET REPEAT command is in effect, the two repeat factors are multiplied to arrive at the actual number of times commands are repeated.

For example, the following command examines the PC ten times:

```
REPEAT 10;EXAMINE PC
```

A more complex example is shown below, along with the sequence of single commands that would duplicate the action of the single command line.

```
REPEAT 3;EXAMINE PC;REPEAT 2;EXAMINE NEXT

EXAMINE PC
EXAMINE NEXT
EXAMINE NEXT
EXAMINE PC
EXAMINE NEXT
EXAMINE NEXT
EXAMINE PC
EXAMINE NEXT
EXAMINE NEXT
```

If SET REPEAT 4 had been previously entered, the above sequence would be repeated four times.

If no commands are specified, the effect is that of a null command.

RESET **PROGRAMMER**

The RESET command performs a MASTER RESET of the KL and retains the clock and parity-stop enables that existed before the reset. This command is not allowed while the KL is running.

RESET ALL **PROGRAMMER**

The RESET ALL command executes the RESET APR, RESET DTE-20, RESET PAG, and RESET PI commands. This command is not allowed while the KL is running.

RESET APR **PROGRAMMER**

The RESET APR command executes a CONO APR,,267760 instruction to clear the KL arithmetic processor. This command is not allowed while the KL is running.

RESET DTE-20 **PROGRAMMER**

The RESET DTE-20 command resets the DTE20 by depositing a 1 in bit 6 of the DTE20 diagnostic word 2. Bit 0 in diagnostic word 3 is set to 1 to indicate word-mode transfers.

RESET ERROR **PROGRAMMER**

The RESET ERROR command executes a CONO APR,,27760 instruction to reset the KL error flags.

RESET INITIALIZE **PROGRAMMER**

The RESET INITIALIZE command performs a MASTER RESET of the KL and sets up normal clock and parity-stop enables. This command is not allowed while the KL is running.

RESET IO **PROGRAMMER**

The RESET IO command executes a CONO APR,,200000 instruction to perform an I/O reset of the KL.

RESET PAG **PROGRAMMER**

The RESET PAG command executes a CONO PAG,,0 instruction followed by a DATAO PAG,,100 instruction to reset the KL PAGing box. This command requires that the KL clock be running.

RESET PI **PROGRAMMER**

The RESET PI command executes a CONO PI,,10000 instruction to reset the KL Priority Interrupt system.

RESTORE AC-BLOCK

The RESTORE AC-BLOCK command resets the current KL AC block number to the KL AC block number that was saved by the SAVE AC-BLOCK command.

RUN taskname **OPERATOR**

The RUN command loads and starts the specified task file. This command is an alias for the MCR command.

SAVE AC-BLOCK

The SAVE AC-BLOCK command saves the current KL AC block number so that it can later be restored with the RESTORE AC-BLOCK command. SAVE AC-BLOCK is used in KLERR command files that execute the SET AC-BLOCK command. The AC BLOCK commands normally are executed in the following order: SAVE AC-BLOCK; SET AC-BLOCK n; RESTORE AC-BLOCK.

NOTE

SAVE AC-BLOCK only saves the AC block number, not the contents of the AC block.

SAVE PC-FLAGS

The SAVE PC-FLAGS command saves the current KL PC and flags for later use by fault continuation. This command is used by the KLERR command files and is not intended for any other use.

SET AC-BLOCK n **PROGRAMMER**

The SET AC-BLOCK command changes the current KL AC block number to the one specified by the argument n in the command. This argument can have a value in the range 0 to 7.

SET CLOCK NORMAL **MAINTENANCE**

The SET CLOCK NORMAL command sets the KL's clock parameters to internal source, full rate, and enables the CRAM, data path, and control logic clocks.

SET CLOCK {**CRAM** / **DATA-PATH** / **CONTROL**} **MAINTENANCE**

This SET CLOCK command enables the specified clock as follows:

CRAM enables the control-RAM clock.

DATA-PATH enables the data-path clock.

CONTROL enables the control logic clock.

SET CLOCK {**EXTERNAL** / **INTERNAL** / **MARGIN**} **MAINTENANCE**

This SET CLOCK command sets the source of the clock pulses: external, internal, or margin. The margin clock is slightly faster than the normal internal clock and is used for diagnosing rate-sensitive problems. There may not be an external clock attached to the KL. Therefore, after you type the SET CLOCK EXTERNAL command, the PARSER will print:

CONFIRM EXTERNAL CLOCK SOURCE (YES OR NO)?

If you answer YES, the operation is performed. If you answer YES and there is no external clock attached, the KL hangs and has to be reset.

SET CLOCK {**FULL** / **HALF** / **QUARTER** / **SLOW**} **MAINTENANCE**

This SET CLOCK command determines the speed of the KL clock: full speed, one half speed, one quarter speed, or slow speed which is equivalent to one eighth speed.

SET CONSOLE [**MAINTENANCE** | **OPERATOR** | **PROGRAMMER** | **USER**] **OPERATOR**

The SET CONSOLE command sets the console mode of operation and, therefore, the allowable subset of PARSER commands.

MAINTENANCE allows the full set of PARSER commands.

PROGRAMMER allows all PARSER commands except diagnostic functions.

OPERATOR allows only those PARSER commands that will not crash the TOPS-10 or TOPS-20 monitor.

USER exits the PARSER.

If no subargument is entered, the console is set to PROGRAMMER mode.

NOTE

If KLINIK is enabled and active, the PARSER does not let you set the console mode any higher than that specified when the KLINIK window was defined.

SET DATE **PROGRAMMER**

The SET DATE command sets RSX-20F's internal date. This date is used in setting up and accessing KLINIK. This command is not available if RSX-20F thinks that it already has a valid date (validity flag is ON). In response to the SET DATE command, the PARSER prompts you as follows:

```
PAR>SET DATE<RET>
DATE: 19 AUG 83
TIME: 1211
CURRENT SYSTEM DATE:
FRIDAY, 19-AUGUST-83 12:11
 VALIDITY FLAG IS:ON
PAR>
```

SET FAULT-CONTINUATION **PROGRAMMER**

The SET FAULT-CONTINUATION command enables the KL's automatic fault continuation reload mechanism. This allows the host to recover from errors detected by the front-end.

SET FS-STOP **MAINTENANCE**

The SET FS-STOP command enables the Field Service stop facility.

SET INCREMENT n **OPERATOR**

The SET INCREMENT command sets the KL increment counter to the value specified by the octal integer, n. The increment counter is used by the INCREMENT and DECREMENT arguments of the EXAMINE and DEPOSIT commands. Also, only KL memory addresses are modified by the increment counter. PDP-11 addresses that are incremented or decremented default to NEXT and PREVIOUS, respectively.

SET KLINIK **OPERATOR**

The SET KLINIK command is used to enable access to the KLINIK link. The command initiates a dialog in which a KLINIK access window and security parameters are established. (refer to Appendix D for the KLINIK dialog.)

SET MEMORY [**ELEVEN** / **TEN**] **OPERATOR**

The SET MEMORY command establishes the default memory for EXAMINEs and DEPOSITs.

ELEVEN means default to the PDP-11 memory.

TEN means default to the KL memory.

The command itself has no default; an argument must be entered. When RSX-20F is first loaded, the default memory is TEN.

SET NOT argument **OPERATOR**

The SET NOT command is the equivalent of the CLEAR command and requires an argument. (See the CLEAR commands.)

SET NOT OUTPUT [device] **OPERATOR**

The SET NOT OUTPUT command is equivalent to the CLEAR OUTPUT command. (See CLEAR OUTPUT command.)

The default devices for the SET NO OUTPUT command are LOG and LPT.

SET OFFSET nnnnnn **PROGRAMMER**

The SET OFFSET command sets the PDP-11 relocation factor to the value specified by nnnnnn, an octal number in the range 77777 (+32,767) through 100000 (-32,768). The relocation factor when RSX-20F is first loaded is the address of the PARSER root overlay.

SET OUTPUT [**LOG** / **LPT** / **TTY**] **OPERATOR**

This command directs the CTY output to the specified device. The available devices are the log file PARSER.LOG, the lineprinter, and the console terminal. The output is directed to all activated devices

If the output is directed to a log file and the file PARSER.LOG does not exist, the file is created and the data written to the new file. If the file already exists, the new data is appended to the file. In either case a header line is output to all active output devices when a logging file is opened or closed.

The default argument is TTY.

SET PARITY-STOP {**ALL** / **AR** / **CRAM** / **DRAM** / **ENABLE** / **FM** / **FS-STOP**} **MAINTENANCE**

The SET PARITY-STOP command allows you to selectively enable parity stops for the Arithmetic Register and extension, CRAM, DRAM, Fast Memory, and Field Service. The parity stops when RSX-20F is first loaded are AR, CRAM, DRAM, and FM with ENABLE ON.

SET RELOAD **PROGRAMMER**

The SET RELOAD command enables the automatic reload of the KL by the PDP-11 front end in situations such as Keep-Alive-Cease or CPU errors.

SET REPEAT n **OPERATOR**

The SET REPEAT command sets the command line repeat factor to n. The value n must be specified as a positive decimal number. Each subsequent command line is repeated n number of times.

SET RETRY **PROGRAMMER**

The SET RETRY command sets the RETRY flag in RSX-20F. When this flag is set, the first occurrence of a Keep-Alive-Cease error results in the execution of the instruction in location 71. This instruction usually branches to a routine that causes the KL monitor to dump memory and request a reload (BUGHLT in TOPS-20, STOPCD in TOPS-10). If the KL cannot accomplish this task before the end of the Keep-Alive period (5 seconds), RSX-20F assumes that the KL is incapacitated. In this case, KLERR is called to take a KL hardware snapshot and then reload the KL.

If the RETRY flag is reset (see CLEAR RETRY), every occurrence of a Keep-Alive-Cease error results in a KLERR snapshot/reload of the KL.

SET TRACKS **PROGRAMMER**

The SET TRACKS command causes RSX-20F to type out, on the console terminal, all KL operations and their results.

SHOW [parameter] **OPERATOR, PROGRAMMER**

The SHOW command is the same as the WHAT command, and accepts the same values as parameters.

SHUTDOWN **OPERATOR**

The SHUTDOWN command DEPOSITs a minus one into the KL EXEC, virtual location 30 (octal). This command is used to bring down a running system gracefully.

Example:

```
PAR>SHUTDOWN
**HALTED**

%DECSYSTEM-20 not running
```

START TEN addr **PROGRAMMER**

The START TEN command starts the KL at the address specified. Control then returns to the PARSER. The starting address, addr, is a required argument and must not be zero.

START MICROCODE [addr] **PROGRAMMER**

The START MICROCODE command performs a MASTER RESET of the KL and then starts the microcode at the specified address. If addr is omitted, the default address is zero. Starting the microcode at an address other than zero is not recommended.

SWEEP [n] **MAINTENANCE**

The SWEEP command performs a sweep of the specified KL AC block or, if no argument is given, all 8 AC blocks are swept. The argument can be 0 to 7. The sweep consists of reading the contents of all the registers in a block and checking for a parity error after each read.

The SWEEP command first executes a SET AC-BLOCK for the specified block, then examines each location. If an FM parity error is detected, a message is output and the KL parity is reset. Then the next location is examined. When no argument is specified and a Clock Error Stop (CES) exists, then all eight blocks are swept. When the sweep is complete, the block is reset to the original AC block. If an error is detected, an attempt is made to write the contents of the registers with good parity. If this fails, an error is generated.

Each time an FM parity error is detected, a message is printed. This message is of the form:

SWEEP PASS number> n:aa/dddddd dddddd

where n is the AC block number, aa is the AC address, and dddddd dddddd is the AC contents.

If this command is issued during a KL clock error stop caused by an FM error, then the current content of the FM output register is output in the form:

FM PARITY ERROR-(BLOCK:ADDR/DAT) n:aa/dddddd dddddd

where n is the AC block number, aa is the AC address, and dddddd dddddd is the AC contents.

TAKE [file] **OPERATOR**

The TAKE command executes the specified file as an indirect command file. All legal PARSER commands, except another TAKE command, are allowed in the command file. During execution of the file, errors are handled exactly as if the commands were input from the CTY.

The commands in the file are executed until an end-of-file condition is reached. At that point the message <EOF> is printed on the CTY and input is accepted from the CTY.

The file specified can be any 6 character alphanumeric file name. The system device should have a file of the form 'file'.CMD, where 'file' is the filename that appeared in the command.

NOTE

Only the filename can be used with the TAKE command. The file extention cannot be entered on the command line. The file is assumed to have the extention .CMD.

Example:

Executing a command file named TEST.CMD.

```
PAR>TAKE TEST.CMD
PAR>!THIS IS AN INDIRECT COMMAND FILE
PAR>!ALL PARSER COMMANDS ARE COMING FROM IT
PAR>WH VER!FIRST COMMAND
 PARSER VERSION:  V07-03
 RSX-20F VERSION:  VB1506
PAR>WH DAT!  SECOND COMMAND
CURRENT SYSTEM DATE:
WEDNESDAY, 10-AUGUST-83 14:10
 VALIDITY FLAG IS:  OFF
PAR>!  THIS IS THE END OF THE FILE
PAR> <EOF>
```

Attempting to TAKE a nonexistent command file.

```
PAR>TAKE TEST2

PAR> -- [TAKE] NSF - NO SUCH FILE
```

UNMARK-MICROCODE n **MAINTENANCE**

The UNMARK-MICROCODE command clears the mark bit in the specified CRAM location. The n is the CRAM address in the range 0 to 2377. The bit is marked with the MARK-MICROCODE command.

WHAT CLOCK **PROGRAMMER**

The WHAT CLOCK command displays the current source, rate, and control of the KL's clocks.

WHAT CONSOLE **OPERATOR**

The WHAT CONSOLE command displays the current console mode: OPERATOR, PROGRAMMER, or MAINTENANCE.

WHAT DATE **OPERATOR**

The WHAT DATE command displays the day, date, and time that are currently stored in RSX-20F. The status of the date validity flag is also displayed.

WHAT FAULT-CONTINUATION **PROGRAMMER**

The WHAT FAULT-CONTINUATION command reports the status of FAULT-CONTINUATION. It displays either of the following:

```
FAULT-CONTINUATION:  ON
or
FAULT-CONTINUATION:  OFF
```

WHAT HARDWARE **OPERATOR**

The WHAT HARDWARE command displays the environmental report that KLINIT generates. The report contains the KL serial number, model type, line frequency, and hardware options. (refer to section 5.3.1)

WHAT INCREMENT **OPERATOR**

The WHAT INCREMENT command displays the current value of the KL increment counter used in EXAMINEs and DEPOSITs.

WHAT KLINIK **OPERATOR**

The WHAT KLINIK command displays the current access status of the KLINIK link (refer to the SET KLINIK command in Appendix D). If no access window has been set up the reply is:

```
KLINIK DISABLED
```

If an access window has been set up and the link is in use, the reply is:

```
KLINIK ACTIVE
     or
KLINIK ACTIVE AFTER REBOOT
```

If an access window has been set up and the link is not in use, the reply is:

```
KLINIK INACTIVE
```

In either of the last two instances, the status is followed by a display of the KLINIK window parameters.

WHAT MEMORY **OPERATOR**

The WHAT MEMORY command displays the default memory for DEPOSITs and EXAMINEs.

WHAT OFFSET **PROGRAMMER**

The WHAT OFFSET command displays the current PDP-11 relocation factor.

WHAT OUTPUT **OPERATOR**

The WHAT OUTPUT command displays the current devices that are logging output.

WHAT PARITY-STOP **PROGRAMMER**

The WHAT PARITY-STOP command displays the current status of the parity stop enable bit, as well as the parity stops that are currently enabled.

WHAT RELOAD **PROGRAMMER**

The WHAT RELOAD command displays the current status of the automatic reload function.

WHAT REPEAT **OPERATOR**

The WHAT REPEAT command displays the current value of the PARSER repeat factor.

WHAT RETRY **PROGRAMMER**

The WHAT RETRY command displays the current status of the RETRY flag in the front end.

WHAT TRACKS **PROGRAMMER**

The WHAT TRACKS command displays the current KL tracking status.

WHAT VERSION **OPERATOR**

The WHAT VERSION command displays the current versions of RSX-20F and the PARSER.

XCT argument **PROGRAMMER**

The XCT command takes a 36-bit numerical expression as an argument and executes this expression as a KL instruction. It also accepts input in the form:

```
func dev, addr
```

where:

func is one of the following:

CONI
CONO
DATAI
DATAO
BLKI
BLKO
CONSO
CONSZO

dev is the octal device code.

addr is the I/O instruction right half.

This input is decoded to create a 36-bit KL I/O instruction that is then executed. This form allows the user to execute I/O instructions to obtain device status information without knowing the opcodes. The user need only know the device code of a few standard devices.

Note that executing an instruction with an opcode (bits 0 through 8) of zero is not allowed. If you attempt to do this, you will receive an ILLEGAL KL OPCODE error message.

ZERO loaddr>hiaddr **PROGRAMMER**

The ZERO command zeroes a specified area of KL memory. ZERO accepts as an argument the boundary addresses of the area to be zeroed: loaddr and hiaddr.

4.5 PARSER ERROR MESSAGES

The following list contains all the error messages that can be issued by the PARSER while in any of the three console modes. The format of each message is:

```
PAR -- [command name] code - message
```

The command name is the name of the command that caused the error.

However, this command name can be PARSER if you typed a string that caused an error in the command parser rather than in a specific command routine. For example, assume that you type an invalid command such as:

```
PAR>KLEAR CONSOLE
```

You will receive the error message:

```
PAR -- [PARSER] NSK - NO SUCH KEYWORD "KLEAR"
```

On the other hand, assume that you type in an invalid argument:

```
PAR>CLEAR KONSOLE
```

You will receive the error message:

```
PAR -- [CLEAR] NSK - NO SUCH KEYWORD "KONSOLE"
```

The various error codes, messages, and explanations are given below.

AMB AMBIGUOUS KEYWORD "xxx"

where "xxx" is the ambiguous keyword. The PARSER found more than one keyword that matched the abbreviation you typed.

NOTE

The PARSER matches your abbreviation against the complete set of commands and arguments, regardless of the subset allowed by the console mode.

APE KL APR ERROR

The PARSER encountered a CPU error (nonexistent memory, parity error, or a similar condition). Call your Field Service Representative.

BAE BURST ARGUMENT ERROR

This is an internal programming failure. Call your Software Support Specialist.

CAE KL CRAM ADDRESS ERROR

This is an internal programming failure. Call either your Field Service Representative or your Software Support Specialist.

CAL CAN'T ASSIGN LUN

RSX-20F is unable to assign a Logical Unit Number. The problem is in the file structure. Retry the operation. If the error continues, call your Field Service Representative.

CAP CAN'T ATTACH LINEPRINTER

The PARSER is unable to attach the line printer. Check to make sure that the line printer is available to the system and that the command format is correct. If the error persists, call your Field Service Representative.

CBO COMMAND BUFFER OVERFLOW

You typed a command line that was more than 280 characters in length. Reenter the command as two or more lines.

CDI CLEAR DATE ILLEGAL

You tried to clear the internal date while the KL was in primary protocol.

CES CLOCK ERROR STOP - code ERROR STOP

The variable, code, is either CRAM, DRAM, FM, or FS-STOP. This message is displayed when the CPU encounters a fatal internal hardware error. Note the code received and call your Field Service Representative. Also, try to reload the system using DISK, DECtape, floppy or switch register. If you use the switch register, make sure that you reload the microcode.

CFH CAN'T FIND KL HALT LOOP

The PARSER tried to halt the KL but failed. Call your Field Service Representative.

CLE CONSOLE LIMIT EXCEEDED

You tried to set a console mode that was higher than the console mode specified in the SET KLINIK command dialog. This is not allowed while the KLINIK link is active in remote mode.

CNR COMMAND IS NOT REPEATABLE

You tried to repeat a command that cannot be repeated. However, the command has been executed once.

CRH CAN'T READ HARDWARE OPTIONS

The PARSER is unable to read the system hardware options. Call your Field Service Representative.

DAV DATE ALREADY VALID

You tried to set a new internal date and the date validity flag was on.

DBT DATE BEFORE TODAY

While in the SET KLINIK command dialog, you tried to specify an open or close date that was prior to the current date.

DCK DIVIDE CHECK

This is an internal programming error. Call your Software Support Specialist.

DMF DEPOSIT KL MEMORY FAILED

This is an internal programming failure. RSX-20F did not accept a deposit directive. Call your Software Support Specialist.

DNF DIRECTORY FILE NOT FOUND

The directory file cannot be found. The problem is in the file structure. Retry the operation, and if the error continues, call your Field Service Representative.

DNP DTE-20 IS NOT PRIVILEGED

This is a fatal error. The DTE20 mode switch is in the wrong position. Call either your Field Service Representative or your Software Support Specialist.

DOR DAY OUT OF RANGE

You specified a day that does not exist in the month you entered.

DSF DTE-20 STATUS FAILURE

A read or write to one of the DTE20 status registers failed. Call your Software Support Specialist.

DTC DTE-20 CONFUSED - RUN AND HALT LOOP

This is a fatal error. The run and halt loop flags were set simultaneously, an impossible situation. Call your Field Service Representative.

ECT EBOX CLOCK TIMEOUT

While the PARSER was doing an execute function, the KL failed to reenter the halt loop within the allotted time. Call your Software Specialist.

EMF EXAMINE KL MEMORY FAILED

This is an internal programming failure. RSX-20F did not accept an examine directive. Call your Software Support Specialist.

EOC END OF COMMAND REQUIRED

The command was ended with a ? and no additional arguments are required. Retype the command and press the RETURN key.

EPE EBUS PARITY ERROR

This is a fatal error. The PARSER encountered an EBUS parity error. Call your Field Service Representative.

ESD EBOX STOPPED - DEPOSIT

The PARSER executed a deposit directive and found that the KL clock was stopped.

ESE EBOX STOPPED - EXAMINE

The PARSER executed an examine directive and found that the KL clock was stopped.

FCF FILE CLOSE FAILURE

The PARSER is unable to close the file. Retry the operation. If the error continues, call your Field Service Representative.

FEF FILE EXTEND FAILURE

The PARSER failed to extend the file. Retry the command. If the error persists, call your Field Service Representative.

FEN FILE ENTER FAILURE

The PARSER is unable to enter the specified file. Retry the command. If the error persists, call your Field Service Representative.

FLF FILE LOOKUP FAILURE

The PARSER failed in its attempt to look up the specified file. Retry the operation. If the error continues, call your Field Service Representative.

FOF FILE OPEN FAILURE

The PARSER failed to open the specified file. Retry the operation. If the error persists, call your Field Service Representative.

FRD FILE READ FAILURE

The PARSER is unable to read the file. Retry the command. If the error persists call your Software Support Specialist.

FRF FUNCTION READ nnn FAILED

A diagnostic function read with function code nnn has failed. This is a fatal error. Call your Field Service Representative and your Software Support Specialist. If the system crashes, try to reload it.

FSW FM SWEEP FAILED

The FM sweep failed. Retry the sweep. If the problem persists, call your Field Service Representative.

FWF FUNCTION WRITE nn FAILED

A diagnostic function write with function code nn has failed. This is a fatal error. Call your Field Service Representative and your Software Support Specialist. If the system crashes, try to reload it.

FWT FILE WRITE FAILURE

The PARSER is unable to write to the specified file. Retry the command. If the error persists call your Field Service Representative.

FXF FUNCTION XCT nn FAILED

A diagnostic function execute with function code nn has failed. This is a fatal error. Call your Field Service Representative and your Software Support Specialist. If the system crashes, try to reload it.

IDF ILLEGAL DATE FORMAT

You entered a date in the wrong format. The correct format is:

```
dd-mmm-yy
```

where the hyphens can be replaced by spaces and the year can be entered as four digits. The day and year must be numeric and the month must be alphabetic. The month can be abbreviated as long as it remains unique.

IFC ILLEGAL FUNCTION CODE

This is either an internal programming error or the result of entering a diagnostic command with an invalid function code. The valid function codes are as follows:

```
FREAD  command takes codes 100-177
FWRITE command takes codes 40-77
FXCT   command takes codes 0-37
```

If the message is not a result of entering a diagnostic command, call your Software Support Specialist.

IFN ILLEGAL FILE NAME

The PARSER cannot accept the filename as specified. Reissue the command, making sure the filename is correct.

ILC ILLEGAL CHARACTER "c"

The PARSER found an illegal character in the command line and "c" is the character's printing equivalent. Nonprinting characters are preceded by a circumflex (^) and converted to their printing equivalent for output.

ILI ILLEGAL INSTRUCTION

CLEAR AC-BLOCK is not allowed.

ILS ILLEGAL SEPARATOR CHARACTER "s"

The PARSER found an illegal separator character in the command line and "s" is the illegal character. Nonprinting characters are preceded by a circumflex (^) and converted to their printing equivalent for output. Note that a tab is converted to one space.

IOC ILLEGAL KL OPCODE

Either you or the PARSER tried to execute a KL instruction with an illegal op-code. If this is not the result of an XCT command, call your Software Support Specialist.

IPC ILLEGAL PASSWORD CHARACTER "c"

During the SET KLINIK dialog, you typed a password containing "c," an illegal character. You must use only numeric or uppercase alphabetic characters in the password.

IRC ILLEGAL REPEAT COUNT

You typed a zero or negative argument to either the REPEAT or SET REPEAT command.

IRE ILLEGAL RECURSION

A TAKE command cannot be used in an indirect command file.

ITF ILLEGAL TIME FORMAT

You entered a time of day that was not in the proper format. The PARSER expects a numeric value of the form hh:mm or hhmm.

ITN ILLEGAL TASK NAME

The RUN or MCR command was entered with no task name.

KCN KL CLOCK IS OFF

The KL clock is off and you tried to execute a command that requires the clock to be on.

KLA KL ADDRESS ERROR

You specified a KL address that was out of range (over 22 bits), negative, or not in octal radix.

KLR ILLEGAL WHILE KL RUNNING

You tried to execute an illegal command while the KL was running.

KNC KL IS NOT CONTINUABLE

You tried to resume processing with the CONTINUE command, but the KL was not in a continuable state. For example, you cannot CONTINUE after a RESET command.

KWE KLINIK WINDOW ERROR

During the SET KLINIK dialog, you specified a window close date and time that is prior to the window open date and time.

MRA MISSING REQUIRED ARGUMENT

You did not specify all the necessary arguments for the command.

MRH HARDWARE OPTIONS MUST YET BE READ

KLINIT has not yet read and stored the hardware options because the KL is not running. Retry the command when the KL is running.

NDI NULL DATE ILLEGAL

During the SET DATE dialog, you answered the DATE: prompt with a carriage return. You must supply a date.

NER NUMERIC EXPRESSION REQUIRED

You entered a command that expects a numeric expression as an argument and something else was entered.

NOR INPUT NUMBER OUT OF RANGE

You specified a number that was out of range or in the wrong radix.

NPI NULL PASSWORD ILLEGAL

During the SET KLINIK dialog, you answered the PASSWORD: prompt with a carriage return. You must supply a password if one is requested.

NSF NO SUCH FILE

The file as specified does not exist. Check the filename and reenter the command if necessary.

NSK NO SUCH KEYWORD "xxx"

You entered a command containing the invalid keyword "xxx".

NST NO SUCH TASK

You specified a nonexistent task in an MCR or RUN command.

NTI NULL TIME ILLEGAL

During the SET DATE dialog, you answered the TIME: prompt with a carriage return. You must specify a time.

OAI ODD ADDRESS ILLEGAL

You tried to examine an odd-numbered PDP-11 address.

OFC ODD FUNCTION CODE

This is an internal programming error. Call your Software Support Specialist.

PTL PASSWORD TOO LONG

During the SET KLINIK dialog, you specified a password that was more than six characters in length.

RPM RIGHT PARENTHESIS MISSING

You omitted a right parenthesis in a numeric expression.

SCF SET CLOCK FAILED

The PARSER cannot validate the clock enable parameters it has just set. This is a hardware error. Call your Field Service Representative.

SKI SET KLINIK ILLEGAL WHILE KLINIK ACTIVE

You tried to set new KLINIK parameters while the KLINIK link was active. If you want to change the parameters, you must first disconnect the KLINIK link by typing DISCONNECT or CLEAR KLINIK.

SPF SET PARITY FAILED

The PARSER cannot validate the parity stop parameters it has just set. This is a hardware error. Call your Field Service Representative.

SZI START AT ZERO ILLEGAL

You tried to start the KL at location zero; this is illegal.

TAA TASK ALREADY ACTIVE

You issued a RUN or MCR command for a task that was already active.

TOR TIME OUT OF RANGE

You specified a time in which the hours were greater than 23 or the minutes were greater than 59.

UNL KL MICROCODE NOT LOADED

The system tried to start the KL microcode and found that it was not loaded or was not functioning. Use DISK, DECtape, FLOPPY, or the switch register to reload the microcode and the system.

VFY VERIFY FAILED

The PARSER cannot verify the correct execution of a DEPOSIT command. Call your Software Support Specialist.

WRM COMMAND NOT AVAILABLE IN THIS CONSOLE MODE

You entered a command that is not available in the current console mode. Use the SET CONSOLE command to change mode.

XTO KL EXECUTE TIMED OUT

The KL failed to reenter the halt loop within the allotted time while performing a fast internal execute function.

YOR YEAR OUT OF RANGE

You specified the year incorrectly.

CHAPTER 5

KLINIT

KLINIT is the KL initialization program. You can run KLINIT in default mode where it performs a fixed series of operations or you can run it in dialog mode and specify selected operations.

When you load the system using the DISK, DECTAPE, or FLOPPY load switch, (Figures 5-1 and 5-2), KLINIT performs the following steps automatically without operator intervention.

1. Loads the KL processor microcode from the appropriate microcode file on the front-end load device.

2. Configures and enables cache memory according to the KLINIT configuration file, KL.CFG. If this file is not present on the front-end load device, all cache is enabled.

3. Configures and interleaves KL memory according to the KLINIT configuration file, KL.CFG. If this file is not present on the front-end load device, all available memory is configured with the highest possible interleaving.

4. If the KL.CFG file does not exist, KLINIT creates a file by that name and stores it on the front-end load device. The file contains the cache and memory configuration in effect at the time.

5. Loads and starts the default KL bootstrap program from the file BOOT.EXB located on the disk, DECtape, or floppy disk device. The bootstrap program then loads and starts the default monitor. The default monitor is found in:

```
SYS:SYSTEM.EXE          for TOPS-10
PS:<SYSTEM>MONITR.EXE   for TOPS-20
```

If you do not want KLINIT to perform the above series of operations, you must enter the dialog mode of KLINIT. Then, you can do any one or more of the following:

- Load and/or verify the KL microcode.
- Configure cache memory as you want it.
- Configure KL memory as you want it.
- Load and start any bootstrap program.

- Specify switches to the bootstrap program.
- Load and start any monitor from disk or magnetic tape.

NOTE

The default bootstrap program BOOT.EXB does not understand TOPS-20 subdirectories. Therefore, for example, you can load <SYSTEM>MONITOR.EXE, but you cannot load <SYSTEM.NEW>MONITOR.EXE.

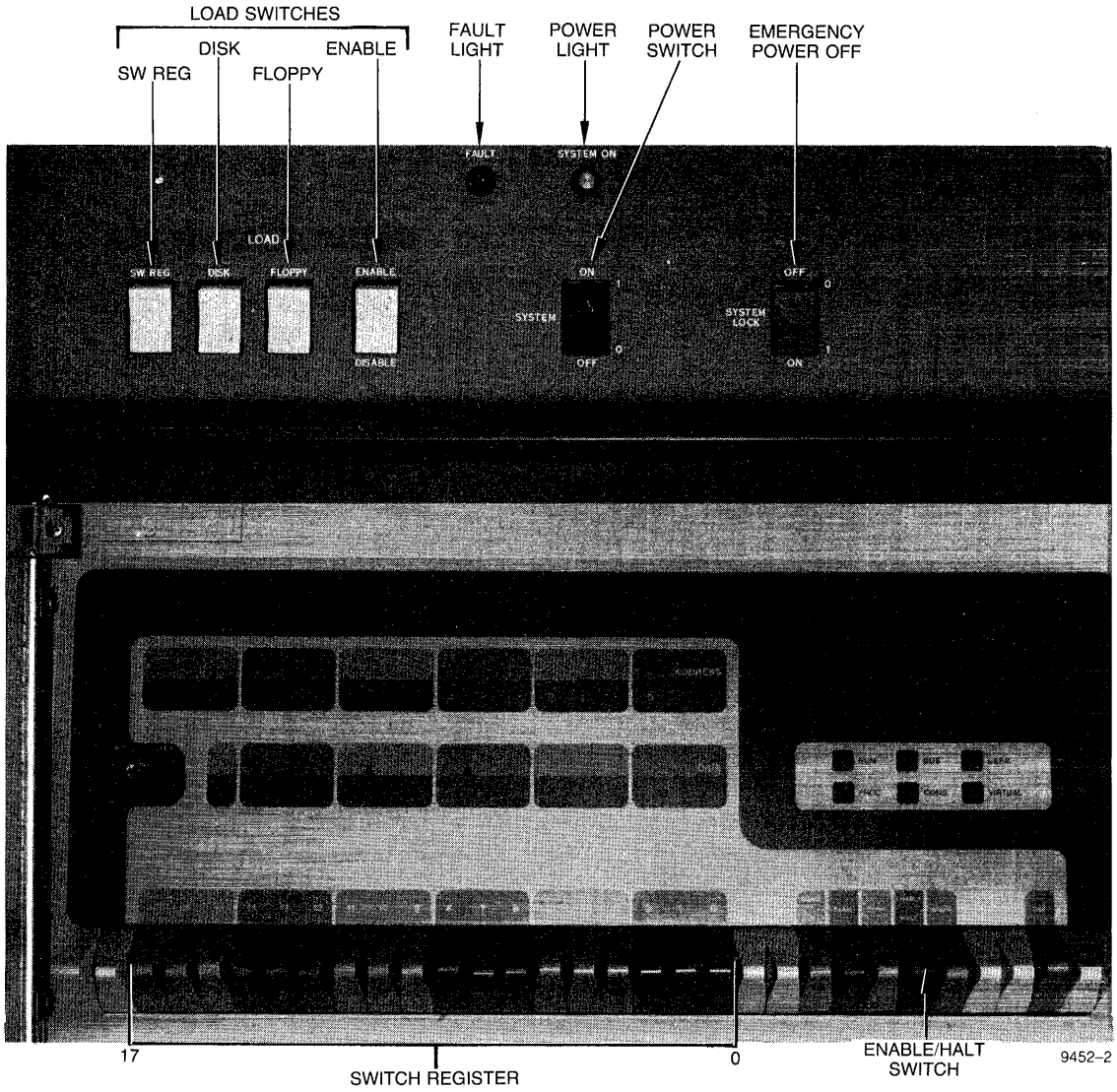


Figure 5-1: Load Switches and Switch Register for KL with Floppy Disks

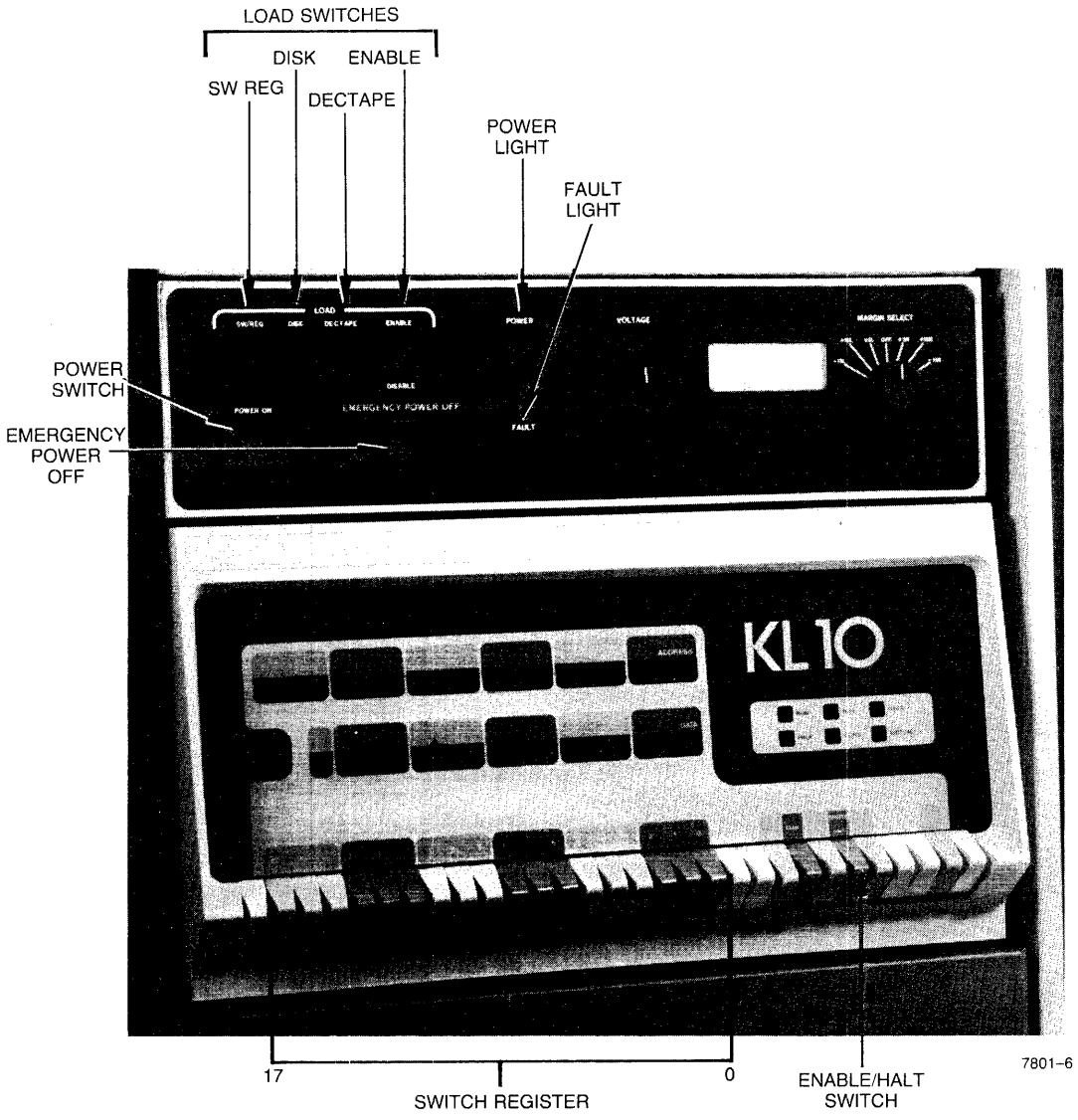


Figure 5-2: Load Switches and Switch Register for KL with DECtapes

5.1 KLINIT LOAD AND START

When you load and start the KL using the SW REG load switch, you usually enter the KLINIT dialog. (See Figures 5-1 and 5-2.) Set the switch register bits 0, 1, and 2 on (in the up position). Refer to Table 5-1 to determine if bits 7 through 10 should be set. Press the load switches SW REG and ENABLE simultaneously. RSX-20F loads and starts and, in turn, loads and starts KLINIT. KLINIT then prompts you with the first question:

```
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
```

You may also enter the KLINIT dialog from the PARSER. Assuming that RSX-20F is running, type the following:

```
CTRL/\ (does not echo)    ;to enter the PARSER
PAR>MCR KLINIT            ;to load KLINIT
KLI -- ENTER DIALOG [NO, YES, EXIT, BOOT]?
```

During the dialog, the following conventions hold:

- A carriage return terminates the answer to a question.
- A RUBOUT or DELETE deletes a character.
- A carriage return by itself in answer to a question selects the default answer to the question. The default answer is the first answer listed.
- CTRL/Z terminates the operator dialog and exits to the PARSER without rewriting the KL.CFG file. If the KLINIT dialog is terminated in this manner, the KL hardware may not be fully or completely initialized.
- CTRL/U deletes the current input line.
- An answer of NO to the ENTER DIALOG question skips the rest of the dialog and assumes all the default answers.
- An answer of BACK to any question returns you to the previous question unless stated otherwise.
- An answer of RESTART to the EXIT question returns you to the first question in the dialog.
- An ESCape typed at any point in a reply before the carriage return restarts the dialog. Note that ESCape does not echo on your terminal.
- An unacceptable answer results in an error message and causes the question to be repeated.
- The minimum size of an abbreviation for any answer other than filename is the first two characters.

Table 5-1: Switch Register Bit Definitions

```
+-----------------------------------------------+
|17|16|15|14| | |11|10| | 8| 7| 6| | | 3| 2| 1| 0|
+-----------------------------------------------+
```

Bit	Meaning
0	If this bit is set, the remaining bits are interpreted. You must set this to load the system using the switch register.
2,1	If both bits 1 and 2 are set, RSX-20F is loaded and the KL initialization operator dialog (KLINIT) is loaded and started. This is what should normally be used when loading the system from the switch register. If bit 1 is set and bit 2 is not set, the RSX-20F monitor is loaded and started; no communication is initiated between the KL and PDP-11 processors at this time. If bit 1 is not set and bit 2 is set, RSX-20F is loaded and started. However, the front end tries to communicate with the KL using secondary and then primary protocol. If the KL is not running, a TBT 11-halt occurs. If both 1 and 2 are not set, the system is loaded much as it is using the DISK, DECtape, or FLOPPY load switch. However, before KLINIT is run, a diagnostic program (CHK11) that checks out the front end CPU and front-end devices is executed.
6-3	Used to set the speed of the DH11 being used as the CTY (see bits 14-11), these bits correspond to the speed selection in the DH11 line parameter register (bits 6, 5 and 3 set indicate 9600 baud). If these bits are not set, the console DL11 is assumed to be the CTY and bits 10-8 are read as the unit number of the RP04/RP06 disk to boot from. If any of these bits are set, then bits 10-8 are read as the DH11 unit number and bits 14-11 are read as the number of the DH11 line that is to be the CTY. Note that you cannot boot the front-end from a disk other than unit 0 if you want to redirect the CTY to a DH11 line using the switch register.
7	If this bit is set, the bootstrap device is a disk pack on a dual-ported drive. If the bit is not set, the bootstrap device is a DECtape drive or floppy disk on the front-end processor.

Table 5-1: Switch Register Bit Definitions (Cont.)

Bit	Meaning
10-8	These bits allows you to specify the unit number of the disk to boot from in binary. No bits set indicate unit 0, bits 9 and 8 set indicate unit 3. If any of bits 6-3 are set, then this field is interpreted as a DH11 unit number instead.
14-11	These four bits allow you to specify the DH11 line number within the selected DH11 unit to which you redirect the CTY. These bits are only valid if any of bits 6-3 are set to indicate the speed of the DH11 line selected.
15	This bit indicates the action taken when an I/O error occurs during the bootstrapping. If the bit is set, the operation is retried indefinitely if an error occurs. If it is not set (the normal case), a halt occurs after ten unsuccessful retries.
17,16	Currently not used, and must not be set.

A bit is set when the corresponding switch is in the upward position.

5.2 KLINIT OPERATOR DIALOG

The following KLINIT dialog includes all the possible questions and all the acceptable answers. The questions are presented in the order in which KLINIT asks them, unless it is specifically stated otherwise in the description of the particular question. In practice, however, only a subset of the dialog is encountered on any one system. The KLINIT program automatically bypasses any questions that are not applicable to the system configuration. In addition, a particular response to one question can result in the bypassing of subsequent questions. This behavior is documented wherever it occurs.

There are two commands that are not used in response to any particular question, but can be used at almost any time. One of these is BACK, which causes the dialog to return to the previous question. This command can be used at any time except on the first question of the dialog, when of course there is no previous question. The other command has four forms that are used to toggle on and off the tracking capability. These forms are T+, T-, and L+, L-. If you wish to see a report on each operation of the initialization procedure, you can give the L+ command, and the complete listing is printed on the line printer. The T+ command prints the listings on the CTY. You should be aware that the T+ command causes a great deal of information to be dumped to the CTY, and uses a lot of time and paper. The L- and T- commands turn off this reporting.

Each of the following questions is followed by the KLINIT prompt, KLI>.

```
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
```

An answer of YES or NO to the question above causes KLINIT to print a hardware environment report containing the KL serial number, machine type, power line frequency, and the system's hardware options. (Refer to Section 5.3.1, Informational Messages.)

NO displays the hardware environment and assumes the default answers for all the remaining questions. This is the last chance to bypass the dialog and take the default path.

YES displays the hardware environment, continues the dialog and asks the next question.

EXIT discontinues the dialog and returns to the RSX-20F monitor.

BOOT skips the rest of the dialog, enables cache memory as directed by KL.CFG, and immediately loads and starts the KL bootstrap program whose name is found in the configuration file. If none is found, the standard KL bootstrap program found in BOOT.EXB is loaded. No defaults are taken when this option is selected.

KLI -- RELOAD MICROCODE [YES,VERIFY,FIX,NO]?

YES loads the KL microcode from the bootstrap device into the KL processor. Should you wish to load the microcode from a file that does not have the default file name, you can respond with YES and, before typing the carriage return, include the actual file name.

VERIFY verifies that the microcode in the KL processor matches the microcode on the bootstrap device. An error report is printed for each location found in error and an error count is incremented. (Refer to Section 5.4.3 for the format and contents of this error report.) Whenever the error count exceeds five, verification is discontinued and the message VERIFY FAILED is issued. If verification continues through all the microcode and the final error count is greater than zero, the VERIFY FAILED message is issued. In both cases, KLINIT returns to the beginning of the dialog. You can then reload the microcode and try again.

FIX verifies the microcode as in the VERIFY option. In addition, whenever an error is detected, KLINIK attempts to reload that location. If the reload operation is successful, the error count is decremented. If the reload fails, the MICROCODE FIX FAILED message is issued. In either case verification continues with the next location. Whenever the error count exceeds five, verification is discontinued and the VERIFY FAILED message is issued. If verification continues through all the microcode and the final error count is greater than zero, the VERIFY FAILED message is issued. In both cases, KLINIT returns to the beginning of the dialog. You can then reload the microcode and try again.

NO neither loads nor verifies the microcode.

KLI -- SELECT PAGE TABLE [FILE,Ø,1,BOTH]?

FILE — selects the MCA25 cache page table as specified in the configuration file KL.CFG. If KL.CFG does not exist, both MCA25 cache pages are selected.

Ø — selects page table Ø. Only half of the MCA25 is used.

1 — selects page table 1. Only half of the MCA25 is used.

BOTH — selects both halves of the MCA25.

KLI -- RECONFIGURE CACHE [FILE,ALL,YES,NO]?

FILE — configures cache memory as specified in the configuration file, KL.CFG. If this file does not exist, all cache memory is enabled. The dialog continues with the CONFIGURE KL MEMORY question.

ALL — enables all cache memory. The dialog continues with the CONFIGURE KL MEMORY question.

YES — configures cache memory under dialog control.

THIS PAGE INTENTIONALLY LEFT BLANK

NO does not reconfigure cache memory; the existing configuration is left unchanged. The dialog continues with the CONFIGURE KL MEMORY question.

KLI -- ENABLE WHICH CACHES [ALL,NONE,0-3]

ALL enables all cache memory.

NONE disables all cache memory.

0-3 enables only the caches specified. For example, to enable caches 0, 1, and 3 reply with:

KLI>0,1,3<cr>

KLI -- CONFIGURE KL MEMORY [FILE,ALL,REVERSE,FORCE,YES,NO]?

NOTE 1

A reply of BACK to this question returns you to the RECONFIGURE CACHE question.

NOTE 2

The FORCE option appears only in systems that have MOS memory. In systems that do not have MOS memory the FORCE option does not appear in the CONFIGURE KL MEMORY question.

FILE configures KL memory as specified in the configuration file, KL.CFG. If this file does not exist, ALL is assumed. KLINIT then prints the logical memory map and the dialog continues with the LOAD KL BOOTSTRAP question.

If the configuration in the KL.CFG file is not consistent with the actual configuration an error message is issued and the dialog restarts from the beginning.

ALL configures KL memory in the normal (forward) direction with as much memory as possible. KLINIT then prints the logical memory map and the dialog continues with the LOAD KL BOOTSTRAP question.

REVERSE configures memory under dialog control; however, the memory configuration is reversed. Before the next question is asked, KLINIT examines memory and prints a physical memory map. This feature has been included for maintenance purposes.

FORCE appears ONLY in systems in which KLINIT can detect the presence of a KW-20 MOS Master Oscillator. The FORCE memory configuration option allows the operator to force KLINIT into a Double-Bit-Error (DBE) scan of the MF-20 MOS memory controllers. This enables KLINIT to attempt to recover "lost" MF-20 blocks. The scan requires approximately twenty-five seconds for each 256K of memory to be scanned.

YES configures memory under dialog control, in the normal (forward) direction. Before the next question is asked, KLINIT examines memory and prints out a physical memory map.

NO does not configure memory at all. The previous memory configuration remains, and the dialog continues with the LOAD KL BOOTSTRAP question.

NOTE

The forward/reverse configuration indicator is saved in the KL.CFG file to allow restoration of the reverse configuration over reloads. If the KL.CFG file does not exist, the default is normal (forward) configuration.

KLI -- CONFIGURE INTERNAL CORE MEMORY [ALL,YES,NO]?

ALL configures all internal core memory. The dialog continues with the INTERNAL CORE MEMORY INTERLEAVE UPPER LIMIT question.

YES configures internal core memory under dialog control.

NO deletes all internal core memory. The dialog continues with questions on other types of memory, if any. See Figure 5-3.

KLI -- MODULES/BLOCKS WITHIN CONTROLLER n [ALL,NONE,SPECIFY]?

NOTE

This question is repeated for each controller. In each iteration, the number n is the current controller number.

ALL configures all the memory modules for controller n.

NONE deletes all the memory modules for controller n.

SPECIFY configures the modules specified. DO NOT TYPE SPECIFY! Valid module numbers are 0 through 3 and the entries are separated by commas. For example, to configure modules 0 and 1, type the following:

```
KLI>0,1<CR>
```

KLI -- INTERNAL CORE MEMORY INTERLEAVE UPPER LIMIT [4,2,1]?

4 allows up to 4-way interleaving.

2 allows up to 2-way interleaving.

1 allows no interleaving

The dialog continues with questions on other types of memory, if any. (See Figure 5-3.) If none, KLINIT prints the logical memory map and the dialog continues with the LOAD KL BOOTSTRAP question.

KLI -- CONFIGURE EXTERNAL CORE MEMORY [YES,NO]?

YES allows you to set the bus-mode for external memory.

NO deletes all external core memory. The dialog continues with questions on other types of memory, if any. (See Figure 5-3.)

KLI -- EXTERNAL CORE MEMORY BUS-MODE [OPTIMAL,1,2,4]?

OPTIMAL sets the bus-mode for optimal performance.

1 sets the bus-mode to 1.

2 sets the bus-mode to 2.

4 sets the bus-mode to 4.

The dialog continues with questions on other types of memory, if any. (See Figure 5-3.) If none, KLINIT prints the logical memory map and the dialog continues with the LOAD KL BOOTSTRAP question.

KLI -- CONFIGURE MOS MEMORY [ALL,YES,NO]?

ALL configures all MOS memory. The dialog continues with the printing of the logical memory map and the LOAD KL BOOTSTRAP question.

YES configures MOS memory under dialog control.

NO deletes all MOS memory. The dialog continues with the printing of the logical memory map and the LOAD KL BOOTSTRAP question.

KLI -- MODULES/BLOCKS WITHIN CONTROLLER n [ALL,NONE,SPECIFY]?

NOTE

This question is repeated as many times as there are controllers. In each iteration, the n is the current controller number.

ALL configures all memory blocks for controller n.

NONE deletes all memory blocks for controller n.

SPECIFY configures the blocks specified. DO NOT TYPE SPECIFY! Type a list of block numbers (0 through 13 octal) separated by commas. For example, to configure blocks 0, 1, 2, 7, 10 and 11 reply with:

KLI>0,1,2,7,10,11<CR>

KLI -- LOAD KL BOOTSTRAP [FILE,YES,NO,FILENAME]?

FILE notifies KLINIT to load the bootstrap specified in the KL.CFG file. If no KL.CFG file exists, KLINIT will use the default bootstrap.

YES notifies KLINIT to load the default bootstrap.

NO notifies KLINIT not to load a bootstrap.

FILENAME notifies KLINIT to load the specified file as the bootstrap.

KLI -- WRITE CONFIGURATION FILE [YES,NO]?

YES notifies KLINIT to write a new KL.CFG file containing the current configuration and load parameters.

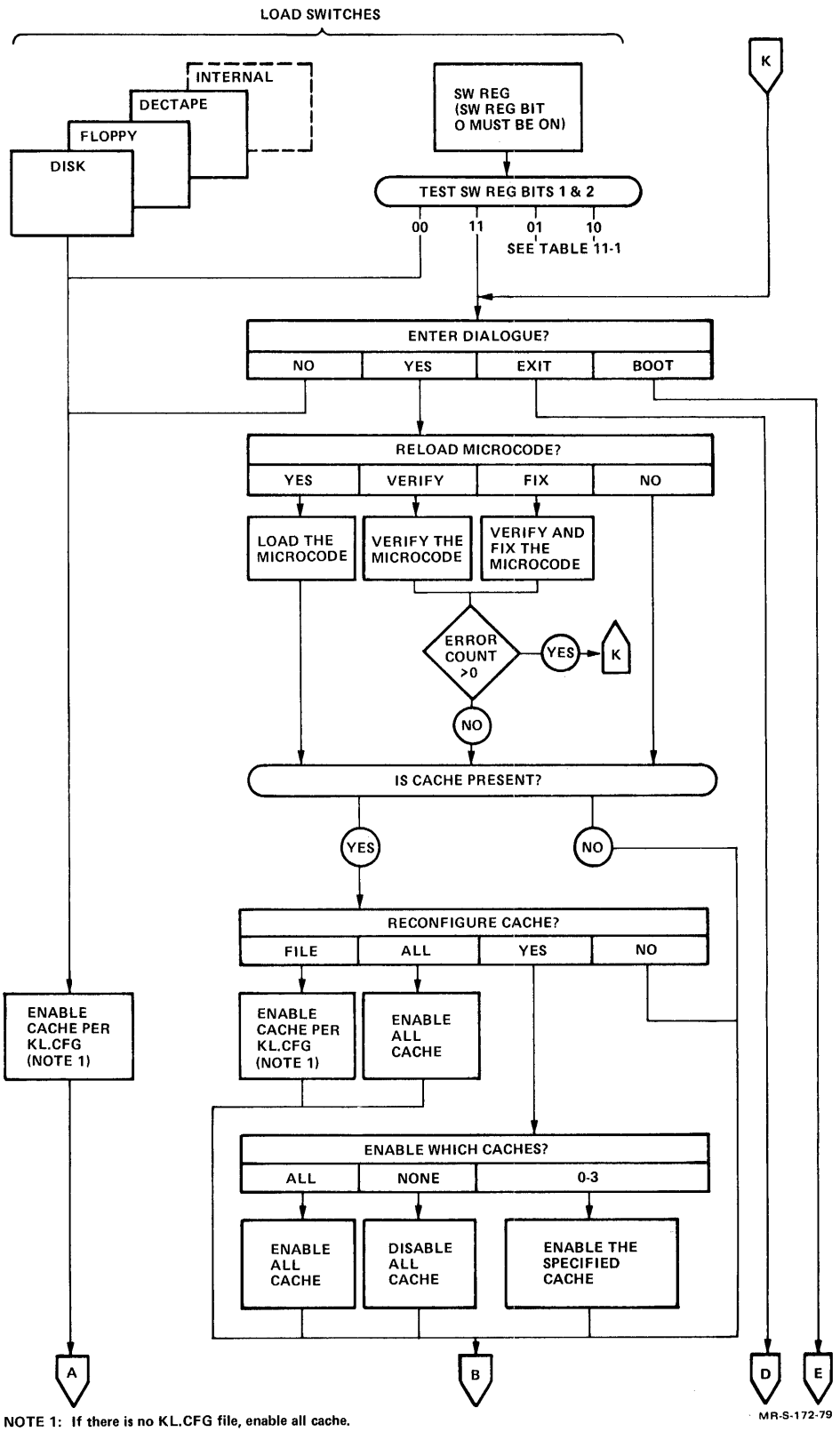
NO notifies KLINIT not to change the existing KL.CFG file.

At this point if a bootstrap was requested, the bootstrap program is loaded into the KL and started. If the answer to the LOAD KL BOOTSTRAP question was NO, the following question is asked:

KLI -- EXIT [YES,RESTART]?

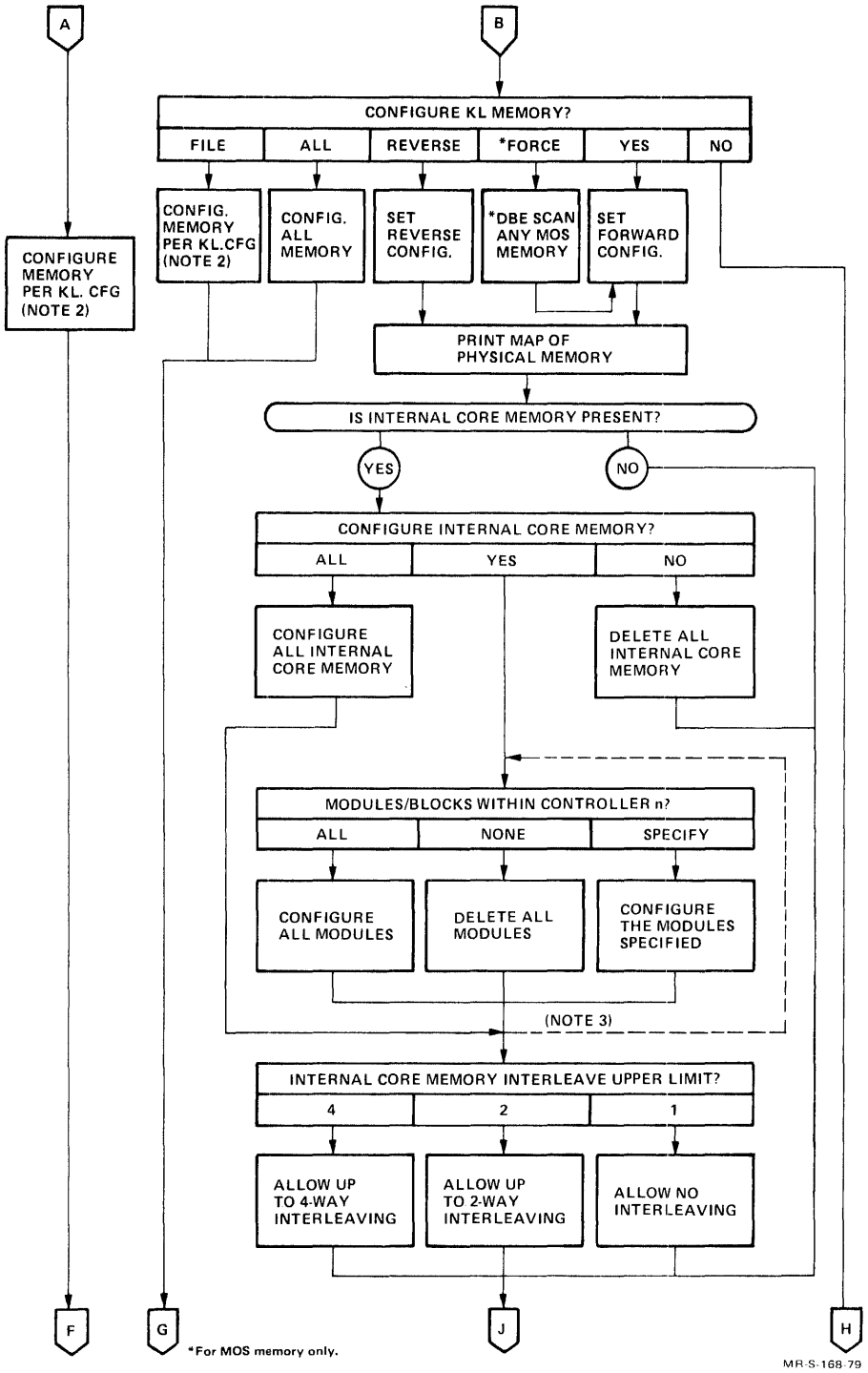
YES exits KLINIT after optionally writing a new KL.CFG file (see previous question).

RESTART restarts the dialog with the ENTER DIALOG question.



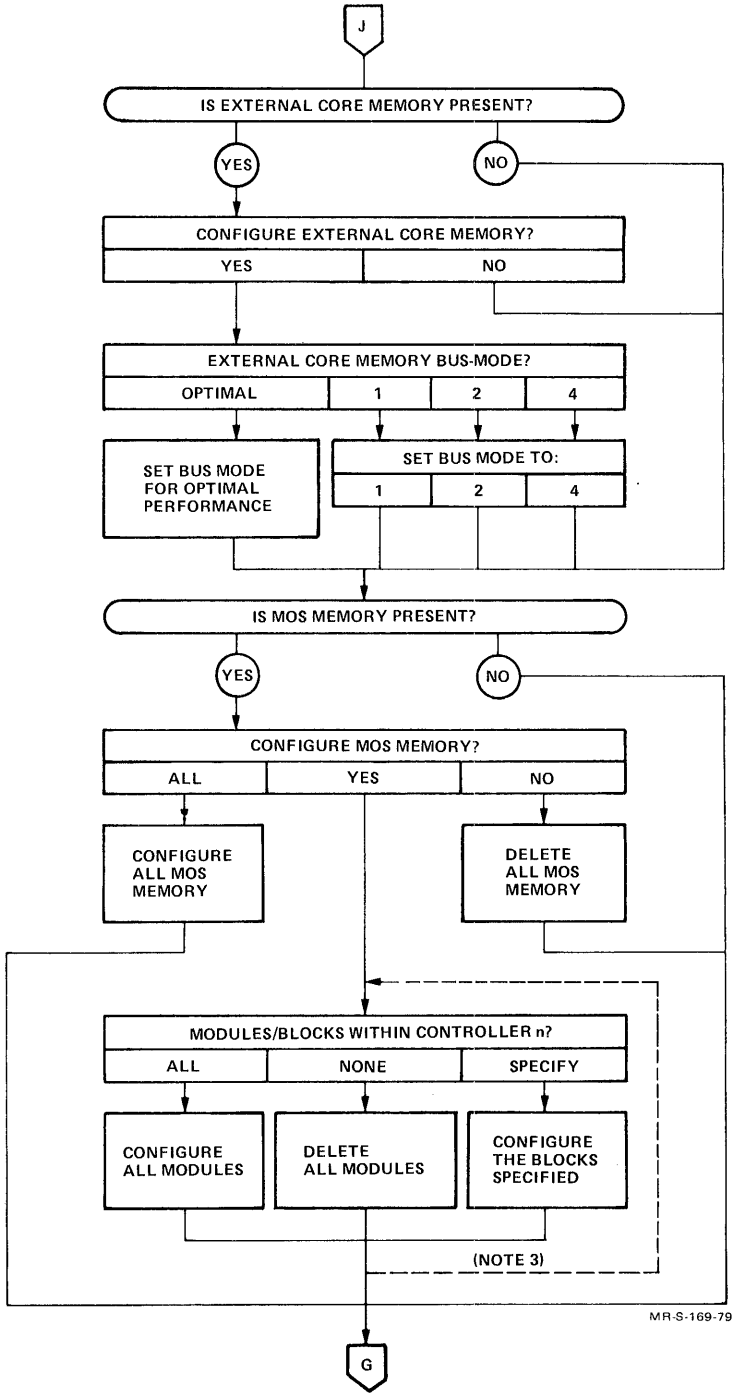
NOTE 1: If there is no KL.CFG file, enable all cache.

Figure 5-3: KLINIT Operator Dialog



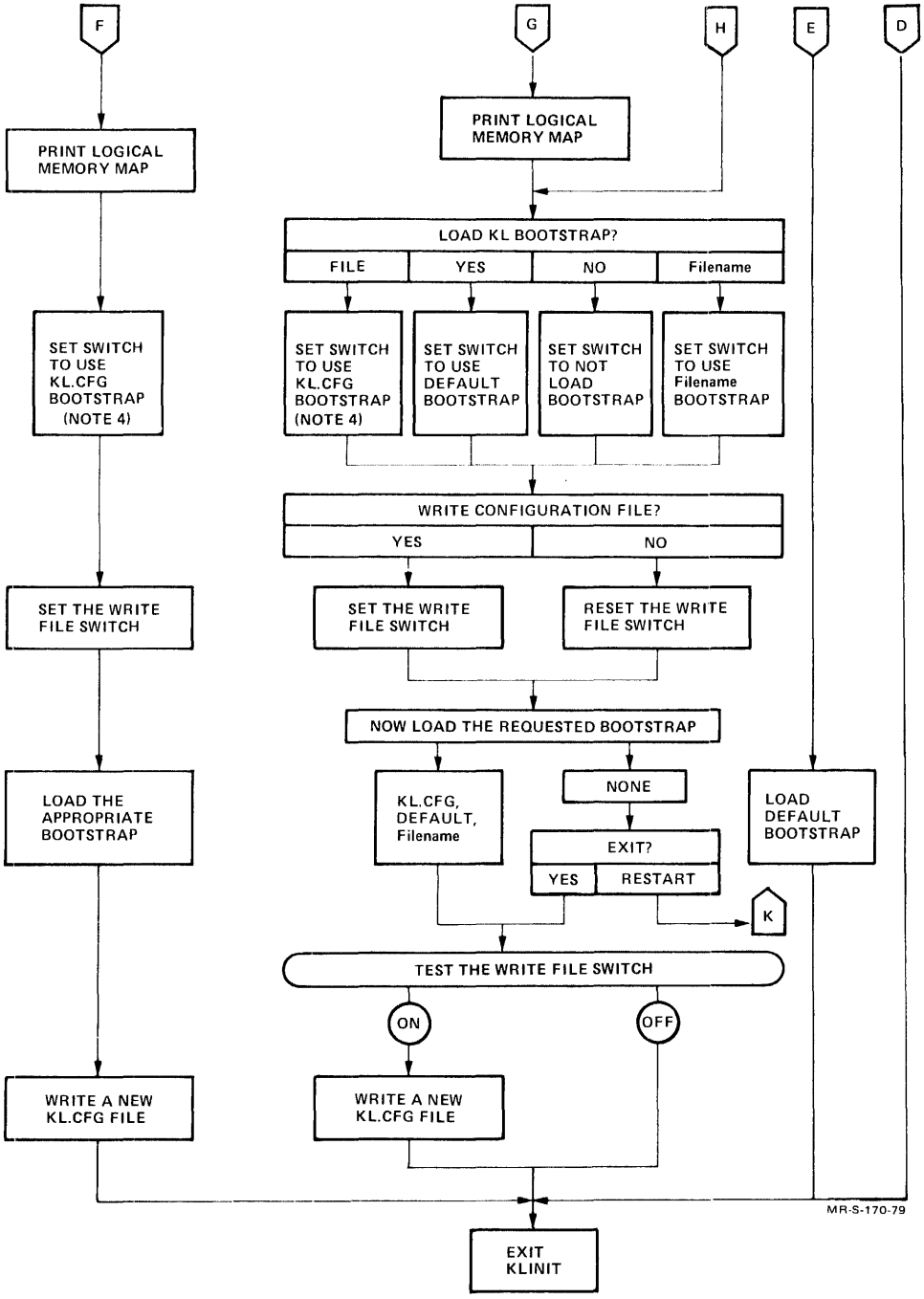
NOTE 2: If there is no KL.CFG file, configure all memory.
NOTE 3: This question is repeated as many times as there are controllers. The controller currently being configured is denoted by n.

Figure 5-3: KLINIT Operator Dialog (Cont.)



NOTE 3: This question is repeated as many times as there are controllers. The controller currently being configured is denoted by n.

Figure 5-3: KLINIT Operator Dialog (Cont.)



NOTE 4: If there is no KL.CFG file, set switch to use the default bootstrap program in BOOT.EXB.

Figure 5-3: KLINIT Operator Dialog (Cont.)

5.3 KLINIT MESSAGES

KLINIT issues four classes of messages: informational, warning, dialog error, and system error messages. These messages are listed in sections 5.3.1 through 5.3.4 according to class.

5.3.1 Informational Messages

KLINIT prints a hardware environment message for each invocation of the program. If KLINIT is activated using the ENABLE and DISK switches, the environment report appears immediately after KLINIT prints its heading and version number. If KLINIT is activated using the ENABLE and SW/REGISTER switches and if the question KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]? is answered with YES or NO, the hardware environment report follows immediately. If this question is answered with EXIT or BOOT, the hardware environment does not appear.

The hardware environment report contains the following information:

- The KL processor serial number
- The KL processor model type
- The power line frequency
- The hardware options available on the system

The serial number is that of the KL processor. The model type can be either A or B. The power line frequency can be either 50 or 60 Hz. The hardware options can include the following:

- MOS Master Oscillator
- Extended Addressing
- Internal Channels
- Cache

Example:

```
KLI -- VERSION VB13-06 RUNNING
KLI -- KL10 S/N:  2136., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT
        MOS MASTER OSCILLATOR
        EXTENDED ADDRESSING
        INTERNAL CHANNELS
        CACHE
```

NOTE

The hardware environment report is not displayed during automatic reloads or during Keep-Alive-Cease processing.

KLINIT also prints informational messages to indicate the normal completion of a KLINIT function. The message text is preceded by "KLI --".

The informational messages include:

KLI -- ALL CACHES ENABLED

All four of the KL processor caches have been enabled.

KLI -- BOOTSTRAP LOADED AND STARTED

A KL bootstrap program has been loaded into KL memory and started. Any messages that follow are a function of the particular bootstrap program being used.

KLI -- CACHES DISABLED

All cache memory has been disabled.

KLI -- CACHES n,n... ENABLED

The specified caches have been enabled.

KLI -- CONFIGURATION FILE WRITTEN

The KL.CFG file has been updated with a new cache and/or memory configuration. This message is issued whenever you set up a nondefault configuration or if the KL.CFG file did not previously exist.

KLI -- KL RESTARTED

The KL processor has been restarted following a power failure or a hardware or software crash.

KLI -- MICROCODE VERSION x[yyy] LOADED

The KL microcode, version x, edit yyy, has been loaded into the KL system from the appropriate microcode file on the front-end bootstrap device.

KLI -- MICROCODE VERSION x[yyy] VERIFIED

The KL microcode, version x, edit yyy, currently residing in the system has been compared correctly with the code in the appropriate microcode file on the front-end bootstrap device.

KLI -- PAGE TABLE SELECTED: BOTH

The entire MCA25 has been initialized and will be used.

5.3.2 Warning Messages

Warning messages inform the operator of some unusual condition. After the message is printed, the KLINIT dialog continues. These messages are preceded by "KLI -- % ".

The warning messages include:

KLI -- % EXTERNAL CORE MEMORY IS OFFLINE

KLINIT found that a DMA2Ø external memory controller was offline.

SYSTEM ACTION:

KLINIT attempts to configure the system without the controller in question.

THIS PAGE INTENTIONALLY LEFT BLANK

KLI -- % EXTERNAL CORE MEMORY RESOURCES DO NOT MATCH FILE

The external memory resources found by KLINIT do not match the KL.CFG file. The KL.CFG file contains more resources than KLINIT can find on the current system. This usually means a controller has dropped off line.

SYSTEM ACTION:

KLINIT attempts to configure the memory it can find in the way closest to that specified in the KL.CFG file.

KLI -- % INTERNAL CORE MEMORY RESOURCES DO NOT MATCH FILE

The internal memory resources found by KLINIT do not match the KL.CFG file. The KL.CFG file contains more resources than KLINIT can find on the current system. This usually means a controller has dropped off line.

SYSTEM ACTION:

KLINIT attempts to configure the memory it can find in the way closest to that specified in the KL.CFG file.

KLI -- % MOS MEMORY IS ALREADY CONFIGURED

KLINIT found that the MOS memory was already configured. It does not attempt to reconfigure MOS memory unless specifically told to do so.

SYSTEM ACTION:

KLINIT proceeds with the initialization.

KLI -- % MOS MEMORY RESOURCES DO NOT MATCH FILE

The MOS memory resources found by KLINIT do not match the KL.CFG file. The KL.CFG file contains more resources than KLINIT can find on the current system. This usually means a controller has dropped off line or some MOS blocks have been deallocated by TGHA.

SYSTEM ACTION:

KLINIT attempts to configure the memory it can find in the way closest to that specified in the KL.CFG file.

KLI -- % NO FILE - ALL CACHE BEING CONFIGURED

The default to the RECONFIGURE CACHE question was taken, and KLINIT could not find the KL.CFG file in the directory.

SYSTEM ACTION:

KLINIT enables all caches.

KLI -- % NO FILE - ALL MEMORY BEING CONFIGURED

The default to the CONFIGURE KL MEMORY question was taken, and KLINIT could not find the KL.CFG file in the directory.

SYSTEM ACTION:

KLINIT configures all available memory and sets the interleaving at the highest level consistent with the setting of the interleave switches on the memory units.

KLI -- % NO FILE - LOADING BOOTSTRAP

KLINIT could not find the KL.CFG file or could not find the bootstrap record in the KL.CFG file.

SYSTEM ACTION:

KLINIT loads the default bootstrap.

KLI -- % PHYSICAL MEMORY CONFIGURATION ALTERED - DUMP OR RESTART SUPPRESSED

During an automatic reload, KLINIT found that the physical configuration of the system does not match the configuration described in the KL.CFG file.

SYSTEM ACTION:

KLINIT suppresses the dump or restart, and proceeds to reload the KL monitor.

5.3.3 Dialog Error Messages

Dialog error messages indicate that your answer to the current KLINIT question is unacceptable. The message text is preceded by "KLI -- ".

The system action for dialog error messages is to repeat the question and the prompt.

Currently, the only dialog error message is:

KLI -- COMMAND SYNTAX ERROR

Your reply is not one of the acceptable answers as specified in the question.

OPERATOR ACTION:

Reply with one of the acceptable answers, correctly spelled, or use carriage return to take the default answer.

5.3.4 System Error Messages

System error messages indicate conditions in which KLINIT cannot continue. These conditions can be brought about by software, hardware, or environmental failures. Sometimes are try is successful other times you may require the assistance of your Field Service Representative or Software Support Specialist. For any system error, it is important to save all console log data and memory dump listings; this material is of prime importance when attempting to determine the cause of the error. System error messages are preceded by KLI -- ? .

Unless noted otherwise, the system action for all system error messages is to restart the KLINIT dialog and repeat the prompt. Whenever a file is specified in a message text, the file is identified in the following format:

```
dev:filename.ext;ver
```

The system messages include:

KLI -- ? BOOTSTRAP LOAD FAILED

A software or hardware error occurred while the KL bootstrap program was being loaded. (See accompanying messages for additional information.)

OPERATOR ACTION:

Reload the bootstrap program by replying:

```
KLI>BOOT
```

If the trouble persists, call your Field Service Representative.

```
KLI -- ? C-RAM DIFFERS AT xxxxxx
KLI -- BAD  xxxxxx xxxxxx xxxxxx xxxxxx xxxxxx xx
KLI -- GOOD xxxxxx xxxxxx xxxxxx xxxxxx xxxxxx xx
KLI -- XOR  xxxxxx xxxxxx xxxxxx xxxxxx xxxxxx xx
```

During the microcode verify operation, the contents of octal location xxxxxx in the KL Control RAM did not match the corresponding code in the appropriate microcode file. The actual contents of the location are printed, followed by the expected contents, and the last line is the result of a bit-by-bit exclusive or (XOR) of the actual and expected values.

OPERATOR ACTION:

Reload the KL microcode and reverify it by means of the KLINIT dialog. If the trouble persists, call your Field Service Representative.

KLI -- ? CACHE ENABLE FAILED

A hardware error has probably occurred while KLINIT was trying to configure the cache memory. (See accompanying messages for additional information.)

OPERATOR ACTION:

Retry the operation; if the trouble persists, call your Field Service Representative. You can also temporarily reconfigure with no cache memory.

KLI -- ? CANNOT FIND [5,5] DIRECTORY

KLINIT cannot locate the PDP-11 system file directory; a software error may have overlaid it.

OPERATOR ACTION:

Reload the system; if the trouble persists, call your Software Support Specialist.

KLI -- ? CANNOT FIND HALT LOOP

KLINIT tried to start the microcode, but it failed to run properly.

OPERATOR ACTION:

Reload the microcode; if the trouble persists, call your Field Service Representative.

KLI -- ? CANNOT GET DEVICES

KLINIT cannot open a system device for communications. This is probably a software error in RSX-20F.

OPERATOR ACTION:

Reload the system; if the trouble persists, call your Software Support Specialist.

KLI -- ? CANNOT RUN KLINIT WHILE KL IS IN PRIMARY PROTOCOL

An attempt was made to run the KLINIT program while the KL processor was running. This condition can arise only if KLINIT is loaded by means of the PARSER command language instruction:

```
PAR>RUN KLINIT
```

OPERATOR ACTION:

If the intent was to rerun KLINIT, follow the appropriate procedures to shut down TOPS-10 or TOPS-20; then reload the system and enter the KLINIT program. If TOPS-10 or TOPS-20 does not shut down properly, set the console mode to PROGRAMMER and reload KLINIT.

KLI -- ? CANNOT START KL

A hardware or software failure occurred while trying to restart from a power failure or system crash during memory determination. (See accompanying messages for additional information.)

OPERATOR ACTION:

Reload the microcode and retry the operation. If the trouble persists, call your Field Service Representative.

KLI -- ? CAN'T DETERMINE KL10 HARDWARE ENVIRONMENT

This message is followed immediately by one of the following messages. If the error occurred in the KLINIT dialog, you will get

KLI -- % PROCEED AT YOUR OWN RISK

while if the error occurred during an automatic reload, you will get

KLI -- % AUTOMATIC RELOAD ABORTED

OPERATOR ACTION:

Contact your Field Service Representative.

KLI -- ? CAN'T SUPPORT MOS MEMORY ON A MODEL "A" CPU

KLINIT has conflicting information on the hardware available to it.

OPERATOR ACTION:

Contact your Field Service Representative.

KL -- ? CLOCK ERROR STOP DURING FAULT CONTINUATION
KLI -- ? CLOCK ERROR STOP DURING KL RESTART

The KL processor clock stopped while KLINIT was monitoring a restart operation. (See accompanying messages for additional information.)

OPERATOR ACTION:

Retry loading the KL bootstrap and monitor. If the trouble persists, call your Software Support Specialist.

KLI -- ? CONFIGURATION FILE NOT CHANGED

The KL.CFG configuration file cannot be updated because the old file cannot be read, the new file cannot be written, or some other error has occurred. (See accompanying messages for additional information.)

OPERATOR ACTION:

Delete the old configuration file and retry the operation. If the trouble persists, call your Software Support Specialist.

KLI -- ? D-RAM DIFFERS AT xxxxxx
KLI -- BAD A:x B:x P:x J:xxxx A:x B:x P:x J:xxxx
KLI -- GOOD A:x B:x P:x J:xxxx A:x B:x P:x J:xxxx
KLI -- XOR A:x B:x P:x J:xxxx A:x B:x P:x J:xxxx

During the microcode verify operation, the contents of octal location xxxxxx in the KL Dispatch RAM did not match the corresponding code in the appropriate microcode file. The actual contents of the locations are printed, the even location first, and the odd location next. This line is followed by the expected contents of the two locations. The last line is the result of a bit-by-bit exclusive or (XOR) of the actual and expected values.

OPERATOR ACTION:

Reload the KL microcode and reverify it by means of the KLINIT dialog. If the trouble persists, call your Field Service Representative.

KLI -- ? DEPOSIT FAILED

KLINIT could not store information into KL memory.

OPERATOR ACTION:

Reload the system and retry the operation. If the trouble persists, call your Field Service Representative.

KLI -- ? DEVICE device FULL

KLINIT cannot find room on the specified front-end load device for an updated copy of the configuration file KL.CFG.

OPERATOR ACTION:

Exit from KLINIT and use a front-end system program such as PIP to delete some files and make room for the updated KL.CFG file. (Make sure that you do not delete any files that contain RSX-20F software. You may wish to consult a system programmer or the system administrator to determine which files can be deleted.) Then reenter KLINIT and retry the operation.

KLI -- ? DF EXECUTE FAILED

A diagnostic function execute failed while KLINIT was initializing the KL processor.

OPERATOR ACTION:

Reload the system and retry the operation. If the trouble persists, call your Field Service Representative.

KLI -- ? DF READ FAILED

A diagnostic function read failed while KLINIT was initializing the KL processor.

OPERATOR ACTION:

Reload the system and retry the operation. If the trouble persists, call your Field Service Representative.

KLI -- ? DF WRITE FAILED

A diagnostic function write failed while KLINIT was initializing the KL processor.

OPERATOR ACTION:

Reload the system and retry the operation. If the trouble persists, call your Field Service Representative.

KLI -- ? DIRECTIVE ERROR -n ON FILE filename

A system error occurred while KLINIT was trying to access the file "filename." The "n" is an octal error code for use by software support.

OPERATOR ACTION:

Reload the system and retry the operation. If the trouble persists, call your Software Support Specialist.

KLI -- ? EXAMINE FAILED

KLINIT could not examine contents of KL memory.

OPERATOR ACTION:

Reload the system and retry the operation. If the trouble persists, call your Field Service Representative.

KLI -- ? FATAL MEMORY CONFIGURATION ERROR - CODE xxx

KLINIT encountered an error in attempting to configure memory. The type of error encountered is specified by the code xxx. Most of these errors are hardware problems or software bugs. The possible codes are listed below, along with the corrective action that can be tried, if any exists.

Code	Corrective Action
3BB	No corrective action is possible. This error code was inserted as a debugging aid, and is not expected to occur in normal operation. A CPU fault could be responsible. Run diagnostics on the CPU and call Field Service.
ABS	No corrective action is possible. The CPU has made an error. Run diagnostics on it and call Field Service.
APL	Make sure the microcode is loaded and retry. If the problem recurs, the CPU has most likely failed. Call Field Service.
B4M	No corrective action is possible. If the hardware environment has not changed, and you have been able to boot memory successfully in the past, the problem is likely to be in the hardware. If, on the other hand, you have an odd hardware configuration, you may have come across a software bug. If Field Service is unable to find the problem, contact a Software Support Specialist.
BCM	No corrective action is possible. If the hardware environment has not changed, and you have been able to boot memory successfully in the past, the problem is likely to be in the hardware. If, on the other hand, you have an odd hardware configuration, you may have come across a software bug. If Field Service is unable to find the problem, contact a Software Support Specialist.
BTL	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
CES	No corrective action is possible. This is almost certainly a hardware fault. Contact Field Service.

Code	Corrective Action
CFT	No corrective action is possible. If the hardware environment has not changed, and you have been able to boot memory successfully in the past, the problem is likely to be in the hardware. If, on the other hand, you have an odd hardware configuration, you may have come across a software bug. If Field Service is unable to find the problem, contact a Software Support Specialist.
CTF	Set all MF20 controllers to software state 0 and retry. If the problem persists, call Field Service. This condition could not be created by MF20 software. It could happen as a result of user setting of the function 1 software state bits.
DCB	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
EDE	Make sure the microcode is loaded and retry. If the failure recurs, call Field Service.
FOE	No corrective action is possible. Contact a Software Support Specialist.
GOO	No corrective action is possible. If the hardware environment has not changed, and you have been able to boot memory successfully in the past, the problem is likely to be in the hardware. If, on the other hand, you have an odd hardware configuration, you may have come across a software bug. If Field Service is unable to find the problem, contact a Software Support Specialist.
HOV	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
IEE	Make sure the microcode is loaded and retry. If the failure recurs, call Field Service.
LDE	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
MAB	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
MFE	No corrective action is possible. This halt often indicates a memory controller failure, especially if the hardware environment has not changed and you have been able to boot memory in the past. You may also have uncovered a software bug. If Field Service cannot find the problem, contact a Software Support Specialist.
MMR	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
MNA	No corrective action is possible. If the hardware environment has not changed, and you have been able to boot memory successfully in the past, the problem is likely to be in the hardware. If, on the other hand, you have an odd hardware configuration, you may have come across a software bug. If Field Service is unable to find the problem, contact a Software Support Specialist.

Code	Corrective Action
NBS	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
NHA	No corrective action is possible. If the hardware environment has not changed, and you have been able to boot memory successfully in the past, the problem is likely to be in the hardware. If, on the other hand, you have an odd hardware configuration, you may have come across a software bug. If Field Service is unable to find the problem, contact a Software Support Specialist.
NMS	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
ODL	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
OO2	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
PDH	Make sure the microcode is loaded and retry. If the failure recurs, call Field Service.
SB4	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
SIH	No corrective action is possible. This is most likely to be a hardware failure. Contact Field Service.
SNR	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
SS0	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
TMD	No corrective action is possible. This is a pure software bug. Contact a Software Support Specialist.
UMB	No corrective action is possible. If the hardware environment has not changed, and you have been able to boot memory successfully in the past, the problem is likely to be in the hardware. If, on the other hand, you have an odd hardware configuration, you may have come across a software bug. If Field Service is unable to find the problem, contact a Software Support Specialist.
XOO	No corrective action is possible. If the hardware environment has not changed, and you have been able to boot memory successfully in the past, the problem is likely to be in the hardware. If, on the other hand, you have an odd hardware configuration, you may have come across a software bug. If Field Service is unable to find the problem, contact a Software Support Specialist.

KL -- ? FAULT CONTINUATION FAILED

Either a DEPOSIT or XCT 72 failed when KLINIT attempted fault continuation. KLINIT aborts fault continuation and reloads the KL.

KLI -- ? FILE filename NOT FOUND

KLINIT cannot find BOOT.EXB, the appropriate microcode file, or the alternate KL bootstrap file in the PDP-11 file directory [5,5] on SY0:.

OPERATOR ACTION:

Ensure that the file being requested resides on the front-end load device and retry the operation.

KLI -- ? I/O ERROR -n ON FILE filename

An I/O error occurred while KLINIT was trying to access the file "filename." The "n" is an RSX-11 octal error code for use by software support. Refer to Appendix A for a list of these error codes and their meanings.

OPERATOR ACTION:

Reload the system and retry the operation. If the trouble persists, call your Software Support Specialist.

KLI -- ? ILLEGAL BUS-MODE

You specified a bus-mode under which the current DMA20 configuration cannot operate.

OPERATOR ACTION:

Retry the operation without the illegal bus-mode setting. If you still have problems, or if you believe that you did not specify an illegal bus-mode, contact your Software Support specialist.

KLI -- ? ILLEGAL MF20 TIMING FILE FORMAT

KLINIT found the MF20 timing file, but it was not in the correct format.

OPERATOR ACTION:

Obtain a new copy of the timing file and retry the operation. The current MF20 timing file name is BF16N1.A11.

KLI -- ? INPUT RECORD LENGTH ERROR

An error occurred while KLINIT was trying to read KL.CFG, the appropriate microcode file, or the KL bootstrap file. This error could be caused by software or hardware failure.

OPERATOR ACTION:

If possible, try other copies of the files. If the trouble persists, call your Software Support Specialist. If the file in question is KL.CFG, you can get around the error by renaming or deleting the file. KLINIT will then write a new KL.CFG file by default.

KLI -- ? INSUFFICIENT MEMORY FOR BOOTSTRAP

KLINIT was unable to find enough memory in the area where it wished to load the bootstrap program. (See any accompanying messages for additional information.) Memory selection switches on the memory units may be set in error.

OPERATOR ACTION:

Check memory selection switches on the memory units and retry the operation. If trouble persists, call your Field Service Representative.

KLI -- ? KL HALT DURING FAULT CONTINUATION
KLI -- ? KL HALT DURING RESTART

The KL processor stopped on a HALT instruction while KLINIT was monitoring a restart operation.

OPERATOR ACTION:

Reboot and load the KL monitor; if the trouble persists, call your Software Support Specialist.

KLI -- ? MASTER RESET FAILED

A MASTER RESET function to the KL failed. This is a hardware error.

OPERATOR ACTION:

Reload the system and retry the operation. If the trouble persists, call your Field Service Representative.

KLI -- ? MEMORY CONFIGURATION FAILED

A hardware or software error occurred while KLINIT was configuring memory. (See accompanying messages for additional information.)

OPERATOR ACTION:

Reload the system and retry the operation. If the trouble persists, call your Field Service Representative.

KLI -- ? MF20 TIMING FILE CHECKSUM ERROR

KLINIT got a checksum error while accessing the MF20 timing file during memory configuration.

OPERATOR ACTION:

Retry the operation. If this still fails, obtain a new copy of the timing file from the distribution media and retry again. The current MF20 timing file name is BF16N1.A11.

KLI -- ? MF20 TIMING FILE READ ERROR

KLINIT got a read error while accessing the MF20 timing file.

OPERATOR ACTION:

Retry the operation. If you still get the read error, obtain a new copy of the timing file and retry. The current MF20 timing file name is BF16N1.A11. If this does not solve the problem, contact your Software Support specialist.

KLI -- ? MICROCODE FIX FAILED

KLINIT found more than five hard (irreparable) errors while trying to fix the microcode.

OPERATOR ACTION:

Retry loading the microcode; if the trouble persists, call your Field Service Representative.

KLI -- ? MICROCODE LOAD FAILED

A hardware or software error occurred while KLINIT was loading the KL microcode. (See accompanying messages for additional information.)

OPERATOR ACTION:

Retry loading the microcode; if the trouble persists, call your Field Service Representative.

KLI -- ? MICROCODE VERIFY FAILED

The verification of the KL microcode discovered errors that are itemized in preceding error messages.

OPERATOR ACTION:

Reload the microcode and verify it. If the trouble persists, call your Field Service Representative.

KLI -- ? NO MEMORY AT LOCATION ZERO

When KLINIT was configuring memory it could not locate any memory unit with address switches set at zero.

OPERATOR ACTION:

Check the memory units and ensure that one of the units has its address switches set at zero; then retry loading.

KLI -- ? NO MF20 TIMING FILE

KLINIT did not find an MF20 timing file.

OPERATOR ACTION:

Obtain a timing file from the release media and retry the operation. The current MF20 timing file name is BF16N1.A11.

KLI -- ? NONEXISTENT CONTROLLER

KLINIT attempted to configure a controller and found that it was not there.

OPERATOR ACTION:

Retry the operation; if the problem persists, call your Field Service Representative.

KLI -- ? NONEXISTENT MODULE/BLOCK

KLINIT attempted to configure a module or block that does not exist in the controller.

OPERATOR ACTION:

Retry the operation; if the problem persists, call your Field Service Representative.

KLI -- ? OUTPUT RECORD LENGTH ERROR

An error occurred while KLINIT was trying to write an updated configuration file, KL.CFG.

OPERATOR ACTION:

Retry the operation and if the problem persists, call your Software Support Specialist.

KLI -- ? POWER-FAIL RESTART FAILED

KLINIT could not restart the KL processor during a power-fail recovery. (See accompanying message for additional information.)

OPERATOR ACTION:

Reload the system using one of the load switch procedures. If the system still does not come up, call your Field Service Representative.

KLI -- ? READ ERROR

A hardware or software error occurred while KLINIT was accessing KL.CFG, the appropriate microcode file, or the KL bootstrap file. (See any accompanying messages for additional information.)

OPERATOR ACTION:

Retry the operation; if the trouble persists, call your Software Support Specialist. If the file in question is KL.CFG, you can get around the read operation by renaming or deleting the file. KLINIT writes a new KL.CFG file by default.

KLI -- ? READ PC FAILED

KLINIT could not read the KL's PC during memory configuration.

OPERATOR ACTION:

Try the operation again. If it still fails, contact your Software Support specialist.

KLI -- ? SYSTEM ERROR DURING KL RESTART

A KLINIT software error occurred during a KL restart operation.

OPERATOR ACTION:

Reload the system using one of the load switch procedures. If you still have problems bringing up the system, call your Software Support Specialist.

KLI -- ? TIMEOUT DURING KL RESTART

While KLINIT was monitoring a KL restart, the 30-second allowable time limit was exceeded.

OPERATOR ACTION:

Try reloading the KL processor using the dialog default. If you still cannot bring the system up, call your Software Support Specialist.

KLI -- ? WRITE ERROR

A hardware or software error occurred while KLINIT was writing an updated copy of the configuration file, KL.CFG. (See any accompanying messages for additional information.)

OPERATOR ACTION:

Retry the write operation; if the trouble persists, call your Software Support Specialist.

5.4 REPORTS RELATING TO THE KLINIT DIALOG

KLINIT prints logical memory configuration maps and physical memory configuration maps for external memory (DMA20) and internal memory (MA20, MB20, MF20, and MG20). KLINIT also prints error reports whenever failures occur during microcode verification. Sections 5.4.1 through 5.4.3.2 describes the information contained in these maps and reports.

5.4.1 External Memory Maps

If you answer YES to the CONFIGURE KL MEMORY question in the KLINIT dialog, KLINIT prints a physical memory configuration map, as shown below:

```
MEMORY RESOURCES:
CONTROLLER ADDRESS  TYPE  MODULES/GROUPS
                          7 6 5 4 3 2 1 0

        4           DMA20  1024K   4 BUS MODE

EXTERNAL MEMORY RESPONSE
  ADDRESS   SIZE
 00000000  1024K
```

This map represents the physical memory allocation, where:

CONTROLLER ADDRESS = memory controller number; this is always 4 for a DMA20

TYPE = memory controller type

MODULES/GROUPS = memory storage module

Under EXTERNAL MEMORY RESPONSE, the total storage is broken down by contiguous blocks and their beginning addresses, where:

ADDRESS = beginning address of memory block

SIZE = size of the memory block

Whenever KLINIT configures KL memory, either by default or through the dialog, it prints a logical memory configuration map on your console terminal. If you answer NO to the CONFIGURE KL MEMORY question, the map is not printed. The format of the map is as follows:

```
LOGICAL MEMORY CONFIGURATION.
  ADDRESS    SIZE   INT    TYPE  CONTROLLER
 00000000   1024K    4    DMA20   4
```

This map tells you how KL memory has been configured, where:

ADDRESS = KL memory address

SIZE = KL memory size in K

INT = KL memory interleave mode

TYPE = memory controller type

CONTROLLER = memory controller number

5.4.2 Internal Memory Maps

If you attempt to configure memory yourself using the dialog, KLINIT prints a physical memory configuration map after you answer YES to CONFIGURE KL MEMORY. The map looks like the following example:

```
MEMORY RESOURCES:
CONTROLLER ADDRESS   TYPE  MODULES/GROUPS
                           7 6 5 4 3 2 1 0

        0            MA20  0 0 0 0 1 1 1 1
        1            MA20  0 0 0 0 1 1 1 1
        11           MF20  0 0 0 0 0 4 4 3
```

NOTE

For the MF20 map each group can contain four possible modules. Group 0 can contain modules 0,1,2,3 or any subset of those modules. Group 1 can contain the modules 4,5,6,7 or any subset of those modules and so on for the remaining groups. The number shown in the map indicates the number of modules in that group but does not indicate which modules are present. The number 3 in the example above shows that 3 modules of the set containing modules 0,1,2,3 are present. It does not, however, show which three modules are present.

This map represents the physical memory allocation, where:

CONTROLLER ADDRESS = memory controller number

TYPE = memory type

MODULES/GROUPS = memory storage module

Some of the rules that the memory configuration algorithm follows are:

1. 2-way or 4-way interleaving can only be done between controllers 0 and 1 or between controllers 2 and 3.

2. To use any memory, module 0 of some controller must be available.

Whenever KLINIT configures KL memory, either by default or through the dialog, KLINIT prints a logical memory configuration map on your console terminal. If you answer NO to the CONFIGURE KL MEMORY question, the map is not printed. The format of the map is as follows:

```
LOGICAL MEMORY CONFIGURATION.
  ADDRESS    SIZE    INT    TYPE   CONTROLLER
 00000000    128K     2     MA20    0 & 1
 00400000    768K     4     MF20    11
```

This map tells you how KL memory has been configured, where:

ADDRESS = KL memory address

SIZE = KL memory size in K

INT = KL memory interleave mode

TYPE = memory controller type

CONTROLLER = memory controller number

5.4.3 Microcode Verification Error Reports

Whenever you reply VERIFY or FIX to the RELOAD MICROCODE question in the KLINIT dialog, the contents of the control and dispatch storage (CRAM and DRAM, respectively) are compared to the corresponding files on disk. Whenever a mismatch is detected, an error report is typed on the CTY.

NOTE

The MARK bit is ignored.

The general format of the error report is:

```
KLI -- ? x-RAM DIFFERS AT location
KLI -- BAD  (contents of n-RAM)
KLI -- GOOD (contents of disk file)
KLI -- XOR  (bit positions that differ)
```

where:

x is C for control storage or D for dispatch storage

location is the RAM address of the error

The XOR line is the result of an exclusive OR of the BAD and GOOD lines and represents the bit positions that differed.

5.4.3.1 CRAM Error Report - The CRAM error report displays the 86 bits of information from left to right for each CRAM location in error. Each line consists of six groups of octal numbers. Each of the first five groups represents a 16-bit quantity; the sixth group represents a six-bit quantity. The bit correspondence is shown below.

Group	Bit Positions
1	0-15
2	16-31
3	32-47
4	48-63
5	64-79
6	80-85 (SPEC field)

The following is an example of a CRAM error report:

```
KLI -- ? C-RAM DIFFERS AT 43
KLI -- BAD  002556 012600 002000 002640 100002 10
KLI -- GOOD 002575 012700 002000 002640 100002 10
KLI -- XOR  000023 000100 000000 000000 000000 00
```

5.4.3.2 DRAM Error Report - The DRAM error report displays the contents of a pair of DRAM locations as two sets of labeled fields. The first set represents the even DRAM location and the second set represents the following odd DRAM location. Each set consists of four fields, as shown below.

Field	Size
A	3 bits
B	3 bits
P	1 bit
J	10 bits (See Note)

NOTE

Although the J field is a 10 bit quantity, bits 5 and 6 are always zero and bits 1 through 4 are common to the even and odd locations. Bits 7 through 10 vary by location.

The following is an example of a DRAM error report:

```
vKLI -- ? D-RAM DIFFERS AT 106
KLI -- BAD  A:2 B:0 P:0 J:1002 A:2 B:0 P:0 J:1002
KLI -- GOOD A:4 B:0 P:0 J:1412 A:2 B:0 P:1 J:1412
KLI -- XOR  A:6 B:0 P:0 J:0410 A:0 B:0 P:1 J:0410

            Contents of       Contents of
            location 106      location 107
```

5.5 KLINIT DIALOG EXAMPLES

1. This example shows the output at your console terminal when you load a TOPS-10 system using the DISK load switch. KLINIT automatically takes the default values without asking you any questions. However, KLINIT tells you that the RAMs (random access memories) have been loaded with the microcode. KLINIT prints the logical memory map and then loads and starts the KL bootstrap program.

```
RSX-20F VE15-06 8:00 10-AUG-83

[SY0:REDIRECTED TO DB0:]
[DB0:MOUNTED]
KLI -- VERSION VA13-06 RUNNING
KLI -- KL10 S/N: 2136., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT
       MOS MASTER OSCILLATOR
       EXTENDED ADDRESSING
       INTERNAL CHANNELS
       CACHE

KLI -- MICROCODE VERSION 324 LOADED
KLI -- ALL CACHES ENABLED
KLI -- % MOS MEMORY IS ALREADY CONFIGURED
```

```
LOGICAL MEMORY CONFIGURATION.
  ADDRESS SIZE  INT   TYPE CONTROLLER
 00000000 128K   4    MA20  0 & 1
 00400000 768K   4    MF20  11

KLI -- CONFIGURATION FILE WRITTEN
KLI -- BOOTSTRAP LOADED AND STARTED
BOOT V2(14)

BOOT>
```

2. This example shows the output at your console terminal when you load a TOPS-20 system using the switch register with switches 0, 1 and 2 set. The KLINIT dialog is entered only to load and start the KL bootstrap. This allows you to leave the microcode and memory configuration as they were.

```
RSX-20F VB15-06 8:00 10-AUG-83

[SY0:REDIRECTED TO DB0:]
[DB0:MOUNTED]
KLI -- VERSION VB13-06 RUNNING
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>BOOT
KLI -- ALL CACHES ENABLED
KLI -- BOOTSTRAP LOADED AND STARTED
BOOT V10(152)

BOOT>
```

3. This example shows the KLINIT dialog being used to reconfigure TOPS-20 memory. KLINIT prints both the physical memory configuration and the logical memory map. These maps indicate that 128K of memory is 2-way interleaved, and 768K is 4-way interleaved.

```
RSX-20F VB15-06 8:00 10-AUG-83

[SY0:REDIRECTED TO DB0:]
[DB0:MOUNTED]
KLI -- VERSION VB13-06 RUNNING
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>YES
KLI -- KL10 S/N: 2136., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT
   MOS MASTER OSCILLATOR
   EXTENDED ADDRESSING
   INTERNAL CHANNELS
   CACHE
KLI -- RELOAD MICROCODE [YES,VERIFY,FIX,NO]?
KLI>YES
KLI -- MICROCODE VERSION 275 LOADED
KLI -- RECONFIGURE CACHE [FILE,ALL,YES,NO]?
KLI>ALL
KLI -- ALL CACHES ENABLED
KLI -- CONFIGURE KL MEMORY [FILE,ALL,REVERSE,FORCE,YES,NO]?
KLI>YES
MEMORY RESOURCES:
CONTROLLER ADDRESS   TYPE  MODULES/GROUPS
                          7 6 5 4 3 2 1 0

             0      MA20 0 0 0 0 1 1 1 1
             1      MA20 0 0 0 0 1 1 1 1
             11     MF20 0 0 0 0 0 4 4 4
```

```
KLI -- CONFIGURE INTERNAL CORE MEMORY [ALL,YES,NO]?
KLI>ALL
KLI -- INTERNAL CORE MEMORY INTERLEAVE UPPER LIMIT [4,2,1]?
KLI>2
KLI -- CONFIGURE MOS MEMORY [ALL,YES,NO]?
KLI>ALL

LOGICAL MEMORY CONFIGURATION.
  ADDRESS  SIZE  INT  TYPE CONTROLLER
 00000000  128K   2   MA20  0 & 1
 00400000  768K   4   MF20  11

KLI -- LOAD KL BOOTSTRAP [FILE,YES,NO,FILENAME]?
KLI>YES
KLI -- WRITE CONFIGURATION FILE [YES,NO]?
KLI>YES
KLI -- CONFIGURATION FILE WRITTEN
KLI -- BOOTSTRAP LOADED AND STARTED
BOOT V10(152)

BOOT>
```

4. This example shows the dialog being used to enable all caches and to reconfigure MB20 memory on a 1091 system. Controllers 0 and 1 are specified with modules 0, 1, and 2 on each controller. KLINIT prints both physical and logical memory maps. The maps indicate that there is 256K of memory, 2-way interleaved.

```
RSX-20F VE15-06 8:00 10-AUG-83

[SY0: REDIRECTED TO DB0:]
[DB0: MOUNTED]
KLI -- VERSION VA13-06
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>YES
KLI -- KL10 S/N: 2136., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT
            MOS MASTER OSCILLATOR
            EXTENDED ADDRESSING
            INTERNAL CHANNELS
            CACHE
KLI -- RELOAD MICROCODE [YES,VERIFY,FIX,NO]?
KLI>NO
KLI -- RECONFIGURE CACHE [FILE,ALL,YES,NO]?
KLI>ALL
KLI -- ALL CACHES ENABLED
KLI -- CONFIGURE KL MEMORY [FILE,ALL,REVERSE,FORCE,YES,NO]?
KLI>YES

MEMORY RESOURCES:
CONTROLLER ADDRESS       TYPE  MODULES/GROUPS
                               7 6 5 4 3 2 1 0

             0           MB20  0 0 0 0 1 1 1 1
             1           MB20  0 0 0 0 1 1 1 1

KLI -- CONFIGURE INTERNAL CORE MEMORY [ALL,YES,NO]?
KLI>YES
KLI -- MODULES/BLOCKS WITHIN CONTROLLER 0 [ALL,NONE,SPECIFY]?
KLI>0,1,2
KLI -- MODULES/BLOCKS WITHIN CONTROLLER 1 [ALL,NONE,SPECIFY]?
KLI>0,1,2
KLI -- INTERNAL CORE MEMORY INTERLEAVE UPPER LIMIT [4,2,1]?
KLI>2
```

```
LOGICAL MEMORY CONFIGURATION.
  ADDRESS SIZE  INT   TYPE CONTROLLER
 00000000 256K   2    MB20  0 & 1

KLI -- LOAD KL BOOTSTRAP [FILE,YES,NO,FILENAME]?
KLI>YES
KLI -- CONFIGURATION FILE WRITTEN
KLI -- BOOTSTRAP LOADED AND STARTED
BOOT V2(14)

BOOT>
```

5. This example shows the console terminal output when the system is loaded using the DECtape load switch. KLINIT did not find a KL.CFG file on the DECtape (%NO FILE messages) therefore, it configured all cache and all available memory. Note that KLINIT informs you whenever it writes a new KL.CFG file; it does so whenever you answer ALL or YES to the CONFIGURE KL MEMORY question or if no previous KL.CFG file exists.

```
RSX-20F VA15-06 8:00 10-AUG-83

[SY0:REDIRECTED TO DT0:]
[DT0:MOUNTED]
KLI -- VERSION VA13-06 RUNNING
KLI -- KL10 S/N: 1026., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT
   EXTENDED ADDRESSING
   INTERNAL CHANNELS
   CACHE

KLI -- MICROCODE VERSION 231 LOADED
KLI -- % NO FILE - ALL CACHE BEING CONFIGURED
KLI -- ALL CACHES ENABLED
KLI -- % NO FILE - ALL MEMORY BEING CONFIGURED

LOGICAL MEMORY CONFIGURATION.
  ADDRESS  SIZE INT    TYPE CONTROLLER
 00000000 1024K  4    DMA20  4

KLI -- CONFIGURATION FILE WRITTEN
KLI -- BOOTSTRAP LOADED AND STARTED
BOOT V2(14)

BOOT>
```

6. This example shows that the specified bootstrap file XXBOOT.EXB was not found. Therefore, after the fatal error messages, the KLINIT dialog restarts.

```
RSX-20F VB15-06 8:00 10-AUG-83

[SY0:REDIRECTED TO DB0:]
[DB0:MOUNTED]
KLI -- VERSION VB13-06 RUNNING
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>YES
KLI -- KL10 S/N: 2136., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT
   MOS MASTER OSCILLATOR
   EXTENDED ADDRESSING
   INTERNAL CHANNELS
   CACHE
```

```
KLI -- RELOAD MICROCODE [YES,VERIFY,FIX,NO]?
KLI>NO
KLI -- RECONFIGURE CACHE [FILE,ALL,YES,NO]?
KLI>NO
KLI -- CONFIGURE KL MEMORY [FILE,ALL,REVERSE,FORCE,YES,NO]?
KLI>NO
KLI -- LOAD KL BOOTSTRAP [YES,NO,FILENAME]?
KLI>XXBOOT
KLI -- WRITE CONFIGURATION FILE [YES,NO]?
KLI>YES
KLI -- ALL CACHES ENABLED
KLI -- ? FILE "SY0:XXBOOT.EXB;0" NOT FOUND
KLI -- ? BOOTSTRAP LOAD FAILED
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>
```

7. This example shows that a <CR> defaults to the first reply listed. (NO to ENTER DIALOG) In this case the default signals KLINIT to bypass any further dialog and assume the default answers to all the remaining questions.

```
RSX-20F VB15-06 8:00 10-AUG-83

[SY0:REDIRECTED TO DB0:]
[DB0:MOUNTED]
KLI -- VERSION VB13-06 RUNNING
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>   (carriage return was pressed here)
KLI -- KL10 S/N: 2136., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT
   MOS MASTER OSCILLATOR
   EXTENDED ADDRESSING
   INTERNAL CHANNELS
   CACHE

KLI -- MICROCODE VERSION 275 LOADED
KLI -- ALL CACHES ENABLED
KLI -- % MOS MEMORY IS ALREADY CONFIGURED

LOGICAL MEMORY CONFIGURATION.
  ADDRESS SIZE  INT   TYPE CONTROLLER
 00000000 128K   2    MA20  0 & 1
 00400000 768K   4    MF20  11

KLI -- CONFIGURATION FILE WRITTEN
KLI -- BOOTSTRAP LOADED AND STARTED
BOOT V10(152)

BOOT>
```

8. This example shows the dialog first being used to load and verify the microcode. Then it shows the cache memory being configured. The TOPS-10 monitor is to be loaded from a magnetic tape so the program BOOTM must be loaded in place of the default program, BOOT. BOOTM is contained in the file MTBOOT.EXB. KLINIT accepts the file name, appends the default file type of .EXB, and loads and starts the magnetic tape bootstrap program.

```
RSX-20F VA15-06 8:00 10-AUG-83

[SY0:REDIRECTED TO DB0:]
[DB0:MOUNTED]
KLI -- VERSION VA13-06 RUNNING
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>YES
KLI -- KL10 S/N: 1026., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT
   EXTENDED ADDRESSING
   INTERNAL CHANNELS
   CACHE

KLI -- RELOAD MICROCODE [YES,VERIFY,FIX,NO]?
KLI>YES
KLI -- MICROCODE VERSION 324 LOADED
KLI -- RECONFIGURE CACHE [FILE,ALL,YES,NO]?
KLI>BACK
KLI -- RELOAD MICROCODE [YES,VERIFY,FIX,NO]?
KLI>VERIFY
KLI -- MICROCODE VERSION 324 VERIFIED
KLI -- RECONFIGURE CACHE [FILE,ALL,YES,NO]?
KLI>YES

KLI -- ENABLE WHICH CACHES [ALL,NONE,0-3]?
KLI>0,1,3
KLI -- CACHES 0,1,3 ENABLED
KLI -- CONFIGURE KL MEMORY [FILE,ALL,REVERSE,FORCE,YES,NO]?
KLI>YES

MEMORY RESOURCES:
CONTROLLER ADDRESS         TYPE MODULES/GROUPS
                                7 6 5 4 3 2 1 0

             4             DMA20  1024K   4 BUS MODE

KLI -- CONFIGURE EXTERNAL CORE MEMORY [YES,NO]?
KLI>YES
KLI -- EXTERNAL CORE MEMORY BUS-MODE [OPTIMAL,1,2,4]?
KLI>4
LOGICAL MEMORY CONFIGURATION.
  ADDRESS SIZE  INT    TYPE CONTROLLER
 00000000 256K   4    DMA20  4

KLI -- LOAD KL BOOTSTRAP [YES,NO,FILENAME]?
KLI>MTBOOT
KLI -- WRITE CONFIGURATION FILE [YES,NO]?
KLI>YES
KLI -- CONFIGURATION FILE WRITTEN
KLI -- BOOTSTRAP LOADED AND STARTED
BOOTM V6(32)

BTM>
```

9. This example shows that an error occurred in verifying the existing microcode. Because the dialog is restarted after a fatal error, the solution you should try is answering YES to the RELOAD MICROCODE question the next time.

```
RSX-20F VB15-06 8:00 10-AUG-83

[SY0:REDIRECTED TO DB0:]
[DB0:MOUNTED]
KLI -- VERSION VB13-06 RUNNING
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>YES
KLI -- KL10 S/N: 2136., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT
   MOS MASTER OSCILLATOR
   EXTENDED ADDRESSING
   INTERNAL CHANNELS
   CACHE

KLI -- RELOAD MICROCODE [YES,VERIFY,FIX,NO]?
KLI>VERIFY
KLI -- ? C-RAM DIFFERS AT 1
KLI -- BAD  002556 012600 002000 002640 100002 10
KLI -- GOOD 002575 012700 002000 002640 100002 10
KLI -- XOR  000023 000100 000000 000000 000000 00
KLI -- ? MICROCODE VERIFY FAILED
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>
```

10. This example shows ESCape being used during the reply to the RELOAD MICROCODE question in order to restart the dialog. It also shows unacceptable answers causing questions to be repeated. Finally, a CTRL/Z causes the dialog to exit to the console processor command language (the PARSER).

```
RSX-20F VB15-06 8:00 10-AUG-83

[SY0:REDIRECTED TO DB0:]
[DB0:MOUNTED]
KLI -- VERSION VB13-06 RUNNING
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>YES
KLI -- KL10 S/N: 2136., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT
   MOS MASTER OSCILLATOR
   EXTENDED ADDRESSING
   INTERNAL CHANNELS
   CACHE
```

```
KLI -- RELOAD MICROCODE [YES,VERIFY,FIX,NO]?
KLI>VER<ESC>
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>YNOT
KLI -- COMMAND SYNTAX ERROR
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>YES
KLI -- RELOAD MICROCODE [YES,VERIFY,FIX,NO]?
KLI>NO
KLI -- RECONFIGURE CACHE [FILE,ALL,YES,NO]?
KLI>MAYBE
KLI -- COMMAND SYNTAX ERROR
KLI -- RECONFIGURE CACHE [FILE,ALL,YES,NO]?
KLI>NO
KLI -- CONFIGURE KL MEMORY [FILE,ALL,REVERSE,FORCE,YES,NO]?
KLI>NO
KLI -- LOAD KL BOOTSTRAP [YES,NO,FILENAME]?
KLI>^Z
```

11. This example shows the KLINIT dialogue being used to load a TOPS-10 system that includes MCA25 memory.

```
RSX-20F VE15-50 00:01 1-Jan-86

[SY0: redirected to DB0:]
[DB0: mounted]
KLI -- VERSION VA15-50 RUNNING
KLI -- ENTER DIALOG [NO,YES,EXIT,BOOT]?
KLI>YES
KLI -- KL10 S/N: 3500., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT:
        MCA25 CACHE PAGER
        MOS MASTER OSCILLATOR
        EXTENDED ADDRESSING
        INTERNAL CHANNELS
        CACHE

KLI -- SELECT PAGE TABLE [FILE,BOTH,0,1]?
KLI>BOTH
KLI -- PAGE TABLE SELECTED: BOTH
KLI -- RELOAD MICROCODE [YES,VERIFY,FIX,NO]?
KLI>YES
KLI -- MICROCODE VERSION 2.0[406] LOADED
KLI -- RECONFIGURE CACHE [FILE,ALL,YES,NO]?
KLI>ALL
KLI -- ALL CACHES ENABLED
KLI -- CONFIGURE KL MEMORY [FILE,ALL,REVERSE,FORCE,YES,NO]?
KLI>FORCE
STARTING MF20 DBE SCAN.  WAIT 25 SEC/256K.

MEMORY RESOURCES:
CONTROLLER ADDRESS  TYPE  MODULES/GROUPS
                          7 6 5 4 3 2 1 0

        10          MG20  0 0 0 0 0 0 4 4
        11          MG20  0 0 0 0 0 0 4 4

KLI -- CONFIGURE MOS MEMORY [ALL,YES,NO]?
KLI>
```

```
LOGICAL MEMORY CONFIGURATION.
  ADDRESS   SIZE   INT   TYPE CONTROLLER
 00000000 2048K    4    MG20  10
 10000000 2048K    4    MG20  11

KLI -- LOAD KL BOOTSTRAP [FILE,YES,NO,FILENAME]?
KLI>YES
KLI -- WRITE CONFIGURATION FILE [YES,NO]?
KLI>YES
KLI -- CONFIGURATION FILE WRITTEN
KLI -- BOOTSTRAP LOADED AND STARTED
BOOT V2(17)

BOOT>
```

CHAPTER 6

RSX-20F UTILITIES

RSX-20F is designed to run with a minimum amount of human interaction. However, occasions arise when some flexibility is needed, for example when the front-end file system must be used or a failing KL must be diagnosed. RSX-20F provides a set of utility programs that gives the operator, system programmer, or Field Service Representative this needed flexibility.

Since RSX-20F was derived from the RSX-11M operating system, many of the RSX-11M utilities were adapted to run on RSX-20F. This chapter describes the following RSX-20F utilities:

- COP - Copies the contents of one floppy disk on to another floppy disk.
- DMO - Dismounts a device.
- INI - Initializes a volume.
- MOU - Mounts a device.
- PIP - Manipulates files. It is used to copy, rename, append, and delete files.
- RED - Redirects I/O requests from one device to another.
- SAV - Saves the system task image.
- UFD - Builds a User File Directory (UFD).
- ZAP - Patches task images.

6.1 COP UTILITY

Sections 6.1.1 through 6.1.4 describe the function and format of the COP utility, provide examples and show error messages.

6.1.1 Function

The COP utility copies and verifies the contents of one floppy disk to another floppy disk, error-checks a single floppy disk, or zeroes a floppy disk. The COP utility is frequently used to make backup copies

of the installation floppies. To use the COP utility, type the following:

```
CTRL/\ (control backslash)

PAR> MCR COP

COP>
```

WARNING

COP is for use only with floppy disks or DECtapes. Do NOT attempt to use COP with RP04 or RP06 disks. COP destroys the file structures of those disks.

6.1.2 Format

The format of the COP command line is as follows:

```
COP>DXn:=DXm:/switch
```

The following switches can be used with COP:

/BL:n,m	Copies starting at (extended) block n,m until the last block on the device. If COP is interrupted (by a CTRL/C for example), it will print out the last block copied.
/CP	Copies the contents of one floppy to another.
/HE	Prints a list of the available switches.
/RD	Reads the device and checks for errors.
/VF	Verifies that First Device = Second Device.
/ZE	Writes zeros onto a device (deleting all files).

6.1.3 Examples

The following command copies and verifies the contents of the floppy disk in DX1: to the floppy disk in DX0:.

```
COP>DX0:=DX1:
```

The following command zeroes the floppy in DX0:; all files are deleted.

```
COP>DX0:/ZE
```

The following command reads the floppy in DX0: and checks for errors.

```
COP>DX0:/RD
```

The following command copies the contents of the floppy in DX0: onto the floppy in DX1: and verifies the copy.

```
COP>DX1:=DX0:/VF
```

The following command prints the format of the COP command line and lists the available switches.

```
COP>/HE
COP>DEST-DEV:=SRC-DEV:/SW-- SWITCHES: /BL:N,M /CP /HE /RD /VF /ZE
```

6.1.4 Error Messages

COP -- *DIAG* -- ABORTED BY ^C AT BLOCK xxx,xxxxxx

COP was interrupted by a CTRL/C at block number xxx,xxxxxx.

COP -- *DIAG* -- I/O ERROR ON Dxx AT BLOCK xxx,xxxxxx

An I/O error was detected on the specified device at block number xxx,xxxxxx.

COP -- *DIAG* -- VERIFY ERROR AT BLOCK xxx,xxxxxx

An error was detected during verification at block number xxx,xxxxxx.

COP -- *FATAL* -- CONFLICTING SWITCHES

A /CP or /VF was used with a /RE or /ZE.

COP -- *FATAL* -- DEVICE Dxx NOT IN SYSTEM

The device Dxx specified in the command string is not in the system or is off-line.

COP -- *FATAL* -- DEVICE Dxx IS WRITE LOCKED

The user does not have access to device Dxx.

COP -- *FATAL* -- DESTINATION DEVICE SPECIFIED FOR /RD

You specified a destination device with the /RD switch. This is an illegal combination of commands.

COP -- *FATAL* -- FATAL I/O ERROR xxx ON Dxx AT BLOCK xxx,xxxxxx

An I/O error xxx (refer to Appendix A) has been detected on device Dxx at block number xxx,xxxxxx.

COP -- *FATAL* -- ILLEGAL SWITCH /xx

The switch /xx is not a legal COP switch.

COP -- *FATAL* -- NO DESTINATION DEVICE SPECIFIED FOR /CP OR /VF OR /ZE

This message is self-explanatory.

COP -- *FATAL* -- NO SOURCE DEVICE SPECIFIED FOR /CP OR /VF OR /RD

This message is self-explanatory.

COP -- *FATAL* -- SOURCE DEVICE SPECIFIED FOR /ZE

You specified a source device with the /ZE switch. This is an illegal combination of commands.

COP -- *FATAL* -- SYNTAX ERROR: xxxxx...

A syntax error was detected in the command string xxxxxx... Perhaps the switch was not preceded by a slash (/) or the input string was too long, or the block number was not separated from the switch by a colon (:).

6.2 INI UTILITY

Sections 6.2.1 through 6.2.4 describe the function and format of the INI utility, provide examples and show error messages.

6.2.1 Function

The INI utility produces a Files-11 volume. The utility initializes the volume (destroys all existing files), writes a dummy bootstrap and a home block, and builds the directory structure. INI must first perform the following steps:

1. Read the TOPS home block (if dual-ported RP04/RP06).
2. Process bad blocks.
3. Allocate space for system files.
4. Write out the storage bit map file.
5. Mark the bit map for TOPS file system (if dual-ported RP04/RP06).
6. Build the boot and home blocks.
7. Write the index and file headers.
8. Write the Master File Directory (MFD).

To initiate the INI utility, type the following:

```
CTRL/\ (control backslash)

PAR>MCR INI

INI>
```

6.2.2 Format

```
INI>dev:/switch /switch ....
```

where:

dev: is the device of the volume to be initialized.

The INI utility accepts a string of switches. A hyphen (-) can be used as a line terminator to extend the INI command line when the selected switches cause the command line to exceed the buffer size that has been specified for the entering terminal. Any number of continuous lines is permitted, but the total command line cannot exceed 512 characters. Switches can be used with the INI utility to modify the home parameters and the INI execution.

The following switch can be used to modify INI execution.

/FULL Declares that the volume being initialized is to be used only as a Files-11 structure (no TOPS-10/TOPS-20 files allowed).

The following switch can be used to modify the home block parameters.

/INDX=index-file-position

The INDX switch specifies the index file logical block number. This switch can be used to force the Master File Directory (MFD), the index file, and the storage allocation file to a specific volume location, usually for minimizing access time. Four possibilities are available.

BEG Places the index file at the beginning of the volume.

MID Places the index file at the middle of the volume. This option must be used for DECtape.

END Places the index file at the end of the volume.

BLK:nnn Places the index file at the specified block number.

Default: /INDX=BEG

NOTES

The INI utility does not indicate when the initialization process is complete. Wait 10 seconds after issuing the command, then type control backslash to return to the PARSER.

Any error that occurs while you are using the INI utility causes an exit from the program. The INI utility must then be reentered from the PARSER.

6.2.3 Examples

The following command sets the location of the index block in the middle of the volume.

```
INI>DB0:/INDX:MID
```

The following command declares the structure to be entirely Files-11.

```
INI>DB0:/FULL
```

6.2.4 Error Messages

INI -- ALLOCATION FOR SYS FILE EXCEEDS VOLUME LIMIT

The system is unable to allocate a system file from the specified block because of intermediate bad blocks or the end of the volume.

INI -- BAD BLOCK FILE FULL

The disk has more than 102 bad regions on it.

INI -- BAD BLOCK HEADER I/O ERROR

INI detected an error while writing out the bad-block file header.

INI -- BLOCKS EXCEED VOLUME LIMIT

The specified block or blocks exceeded the physical size of the volume.

INI -- BOOT BLOCK WRITE ERROR

INI detected an error while writing out the volume boot block.

INI -- CHECKING DDnn

This is not an error message. An automatic bad block specification was proceeding, using the bad block file provided by the Bad Block Locator utility program.

INI -- CHECKPOINT FILE HEADER I/O ERROR

An error was detected in writing out the checkpoint file header.

INI -- COMMAND I/O ERROR

INI encountered an I/O error while attempting to execute the command.

INI -- COMMAND TOO LONG

The command, including continuation lines, exceeded the maximum length of 512 characters.

INI -- DATA ERROR

The command specified a bad block number or contiguous region that is too large.

INI -- DEVICE NOT IN SYSTEM

INI was unable to find the device on which it was supposed to act.

INI -- DEVICE NOT READY

The device on which INI expects to operate is not in the ready state.

INI -- DEVICE WRITE-LOCKED

INI attempted to operate on a device and found the device write-locked. INI could, therefore, do nothing with the device.

INI -- DUPLICATE BLOCK(S) FOUND

A block that was defined as bad is being defined as bad a second time.

INI -- FILE CORRUPT - DATA IGNORED

Although automatic bad block recognition was selected, the bad block data on the disk was not in the correct format and was ignored.

INI -- HANDLER NOT RESIDENT

You should never get this message. If it appears, contact your Software Support Specialist; you may have corrupted software.

INI -- HOME BLOCK ALLOCATE WRITE ERROR

INI detected an error while overwriting a bad home block area.

INI -- HOME BLOCK WRITE ERROR

INI detected an error while overwriting a bad home block area.

INI -- ILLEGAL KEYWORD

The command string contained an illegal keyword.

INI -- ILLEGAL UIC

You should never get this message. If it appears, contact your Software Support Specialist; you may have corrupted software.

INI -- INDEX FILE BITMAP I/O ERROR

INI encountered an I/O error while attempting to read the index file bitmap.

INI -- INDEX FILE HEADER I/O ERROR

INI encountered an I/O error while attempting to read the index file header.

INI -- MFD FILE HEADER I/O ERROR

INI encountered an I/O error while attempting to read the MFD file header.

INI -- MFD WRITE ERROR

INI encountered an I/O error while making an entry in the MFD.

INI -- NO BAD BLOCK DATA FOUND

INI is unable to read the bad block information from the last track of the disk.

INI -- RSX-20F FILE SYSTEM ALLOCATED LESS THAN 256. BLOCKS

INI allocated less than 256 blocks of space to the front end file system.

INI -- RSX-20F FILE SYSTEM OUTSIDE OF VOLUME RANGE

When INI went to the home block to find out where the front-end file system is located, it was given an address that is not within the bounds of the volume in question. In other words, INI could not find the front-end file system.

INI -- STORAGE BITMAP FILE HEADER I/O ERROR

INI got an I/O error while attempting to read the file header for the storage bitmap.

INI -- STORAGE BITMAP I/O ERROR

INI got an I/O error while reading the storage bitmap.

INI -- SYNTAX ERROR

There was a syntax error in the command string to INI.

INI -- TOPS HOME BLOCK I/O ERROR

INI received an I/O error while trying to read the TOPS home block.

INI -- TOPS HOME BLOCK NOT FOUND

INI was unable to find the TOPS copy of the home block.

INI -- WRONG PACK - NO RSX-20F FILE SYSTEM ALLOCATED

INI was not able to find any front-end file system on the specified pack.

6.3 MOU AND DMO

Sections 6.3.1 through 6.3.4 describe the function and format of the MOU and DMO utilities, provide examples and show error messages.

6.3.1 Function

The MOU and DMO utilities give information to RSX-20F that makes a device available to another user utility or takes away a device. RSX-20F then allocates or deallocates buffer space and Logical Unit Number (LUN) information as required.

The MOU utility creates the Volume Control Block (VCB) and declares that the volume is logically on-line for access by the file system. The VCB is allocated in the dynamic memory and controls access to the volume.

The DMO utility declares that the volume or communication channel specified is logically off-line. After a DMO operation, the device cannot be accessed by the associated Ancillary Control Processor. A request is placed in the file system queue to delete the Volume Control Block, and the volume is marked for dismount so that no additional files can be accessed on the volume. The command is completed when the VCB is deleted; the VCB deletion does not occur until all accessed files on the volume have been deaccessed. The system indicates that the process is completed by issuing the following message:

```
dev:  DISMOUNT COMPLETE
```

A delay can occur between the issuance of the command and the printing of this message if a number of I/O requests are pending or a number of files are accessed on the volume.

No switches are available to either the MOU or DMO utility. The only devices available that can be mounted are as follows:

```
DB0: disk drive 0 (RP04/06)
.
.
.
DBn: disk drive n (RP04/06)

DX0: floppy drive 0
DX1: floppy drive 1
DT0: DECtape drive 0
DT1: DECtape drive 1

FE0: pseudodevice FE0: to talk to the KL
.
.
.
FEn: pseudodevice FEn: to talk to the KL
```

To initiate the MOU utility, type the following:

```
CTRL/\ (control backslash)

PAR>MOU

MOU>
```

To initiate the DMO utility, type:

```
CTRL/\ (control backslash)

PAR>DMO

DMO>
```

6.3.2 Format

```
MOU>dev:
```

where:

dev: is the device on which the volume is to be mounted. Only devices that have an Ancillary Control Processor can be mounted. Devices that meet this criterion are rigid disk, floppy disk, DECtape, and the FE: device.

```
DMO>dev:
```

where:

dev: is the device that holds the volume to be dismounted.

6.3.3 Examples

The following command mounts the volume on device DX1:.

```
MOU>DX1:
```

The following command dismounts the volume on device DX1:.

```
DMO>DX1:
```

The following command deletes the volume control block (VCB) for DX1:.

```
DMO>DX1:
```

6.3.4 Error Messages

This section lists the error messages that can arise during the execution of MOU and DMO command strings. Since they are presented in alphabetic order, the messages that come only from DMO are listed first, followed by those that come only from MOU. Finally, there are a few messages that can come from either utility. These are listed with the string "xxx" substituted for one of the strings "DMO" or "MOU".

DMO -- DEVICE CANNOT BE DISMOUNTED

The device dismount command could not be executed. DMO was unable to determine the specific reason for the problem. You should never get this message from DMO; if you do, contact your Software Support Specialist.

DMO -- DEVICE NOT MOUNTED

The specified device is not mounted.

DMO -- DISMOUNT ERROR xx

The attempt to dismount a device failed because an I/O error was received. Refer to Appendix A for a list of the I/O error codes.

MOU -- DEVICE ALREADY MOUNTED

The specified device is already mounted.

MOU -- MOUNT ERROR xx

The attempt to mount a device failed because an I/O error was received. Refer to Appendix A for a list of the I/O error codes.

MOU -- NO ACP FOR DEVICE

The task specified as ACP or the default ACP is not installed in the system.

xxx -- ACP REQUEST ERROR

The ACP for the specified device could not be removed.

xxx -- DEVICE NOT IN SYSTEM

The device specified in the command to MOU or DMO was not found to be in the system resources.

xxx -- SYNTAX ERROR

The command you typed contained an error in syntax.

6.4 PIP - PERIPHERAL INTERCHANGE PROGRAM

Sections 6.4.1 through 6.4.5 describe the function, format, switches and subswitches of PIP, present examples, and show error messages.

6.4.1 Function

The Peripheral Interchange Program (PIP) is an RSX-20F utility for the manipulation of files. PIP performs the following major functions:

- Copying files from one device or area to another
- Deleting files
- Renaming files
- Listing file directories
- Concatenating or appending files

6.4.2 Initiating PIP

PIP can be initiated by typing the following:

```
CTRL/\ (Control Backslash)

PAR>MCR PIP

PIP>
```

6.4.3 PIP Command String Format

The general PIP command string is:

```
PIP>outfile=infile1[,infile2,infile3,...infileN][/switch][/subswitch]
```

where:

outfile	The output file specifier in the form device:[UIC]filename.filetype;version. If the output filename, file type, and version are the null set or *.*,*, the input filename, file type, and version are preserved. If any part of the output file specifier (filename, file type, or version) is entered, wildcards cannot be used for the remaining file specifiers.
infile	The input file specifier in the form device:[UIC]filename.file type;version. If the filename, file type, and version are null, then *.*;* is the default.
switch	Any one of the switches that can be entered.
subswitch	Any one of the subswitches that can be entered immediately after a switch.

6.4.4 PIP Switches and Subswitches

A switch consists of a slash (/) followed by a two-character switch name and optionally followed by a subswitch name separated from the switch by a slash. The subswitch can have arguments, which are separated from the subswitch by a colon (:).

Switches are global; that is, they can be specified once for an entire list of file specifiers. Subswitches are local; that is, they apply only to the file specifier that immediately precedes them.

NOTE

If a subswitch is applied to the first file specifier in a collection of file specifiers, and no command switch has been specified, PIP assumes that the command associated with the subswitch is the one requested. This switch is then applied to the entire collection.

PIP switches and their functions are listed below.

Switch	Name	Function
/AP	APPEND	Adds files to the end of an existing file.
No Switch	COPY	Copies a file.
/DE	DELETE	Deletes one or more files.
/FR	FREE	Prints out available space on specified volume.
/LI	LIST	Lists a directory file.
/ME	MERGE	Concatenates two or more files.
/PU	PURGE	Deletes obsolete version(s) of a file.
/RE	RENAME	Changes the name of a file.

APPEND Switch (/AP)

Function

The /AP switch opens an existing file and appends the input file(s) to the end of it.

Format

```
PIP>outfile=infile[ [,infile2,...,infileN]/AP
```

Examples

The following command opens FILE.DAT;1 on DX1: and appends the contents of files TEST.DAT;2, FILE2.TXT;3, and PRACT.DAT;1 to it.

```
PIP>DX1:FILE1.DAT;1=TEST.DAT;2,FILE2.TXT;3,PRECT.DAT;1/AP
```

COPY (no switch)

Function

The COPY switch is used to create a copy of a file on the same or another device. COPY is the PIP default switch when only one output file specifier and one input file specifier are contained in a command line.

Format

```
PIP>outfile=infile
```

NOTE

If the output filename, file type, and version are either null or *.*;*, the input filename, file type, and version are preserved.

The following subswitch can be used with the COPY switch:

/NV	New Version - This switch allows the user to force the output version number of the file being copied to be one greater than the greatest version of the file already in the output directory.

Examples

The following command copies TEST.DAT;1 on DX2: to DX1: as SAMP.DAT;1.

```
PIP>DX1:SAMP.DAT;1=DX2:TEST.DAT;1
```

The following command copies all versions of all files of type .DAT from DX0: to DX1:.

```
PIP>DX1:=DX0:*.DAT;*
```

DELETE Switch (/DE)

Function

The /DE switch allows the user to delete files from a directory.

Format

```
PIP>infile1[, infile2,...infileN]/DE
```

NOTES

1. A version number must always be specified with the /DE switch.

2. A version number of -1 can be used to delete the oldest version of a file. An explicit version of ;0 or ; can be used to delete the most recent version of a file.

Examples

The following command deletes version one of the file TEST.DAT in the default directory on the default device.

```
PIP>TEST.DAT;1/DE
```

The following command deletes all files of type DAT or TMP from the default directory on the default device.

```
PIP>*.DAT;*,*.TMP;*/DE
```

FREE Switch (/FR)

The /FR switch allows the user to print the available space on a specific device.

Format

```
PIP>dev:/FR
```

The output from the /FR switch is shown below.

```
dev:  HAS nnnn.  BLOCKS FREE, nnnn BLOCKS USED OUT OF nnnn.
```

Example

```
PIP>DB0:/FR
DB0: HAS 2813.  BLOCKS FREE, 1416.  BLOCKS USED OUT OF 4229.
```

LIST Switch (/LI)

Function

The /LI switch allows the user to list one or more directories. PIP also provides the following three alternate mode switches, which allow the user to specify different directory formats.

```
/BR
/FU
/TB
```

/LI lists the following information:

Filename.file type;version

Number of blocks used (decimal)

File code:
- null = noncontiguous
- C = contiguous
- L = locked

Creation date and time

Example

```
PIP>DT1:/LI

DIRECTORY DT1:[5,5]
9-AUG-83 13:27

T20ACP.TSK;1506          8.      C  09-AUG-83 13:10
BOO.TSK;1506             19.     C  09-AUG-83 13:11
COP.TSK;1506             8.      C  09-AUG-83 13:12
DMO.TSK;1506             5.      C  09-AUG-83 13:13
INI.TSK;1506             23.     C  09-AUG-83 13:15
PIP.TSK;1506             56.     C  09-AUG-83 13:16
RED.TSK;1506             6.      C  09-AUG-83 13:17
SAV.TSK;1506             23.     C  09-AUG-83 13:18
UFD.TSK;1506             9.      C  09-AUG-83 13:20
ZAP.TSK;1506             38.     C  09-AUG-83 13:21
RSX-20F.SYS;1506         59.     C  09-AUG-83 13:22
RSX-20F.MAP;1            153.       09-AUG-83 13:23

TOTAL OF 407./407. BLOCKS IN 12. FILES
```

/BR displays the following brief form of the directory listing:

```
Filename.  file type;  version
```

Example

```
PIP>DT1:/BR

DIRECTORY DT1:[5,5]

T20ACP.TSK;1506
BOO.TSK;1506
COP.TSK;1506
DMO.TSK;1506
INI.TSK;1506
PIP.TSK;1506
RED.TSK;1506
SAV.TSK;1506
UFD.TSK;1506
ZAP.TSK;1506
RSX-20F.SYS;1506
RSX-20F.MAP;1
```

/FU displays the full directory listing containing the following information:

```
Filename.  file type;  version

File identification number in the format (file number, file
sequence number)

Number of blocks used/allocated (decimal)

File code
   null = noncontiguous
   C = contiguous
   L = locked

Creation date and time

Owner UIC and file protection in the format:
   [group, member]
   [system, owner, group, world]
```

These protection fields can contain the values R,W,E,D,

where:

```
   R = read access permitted
   W = write access permitted
   E = extend privilege
   D = delete privilege permitted

Date and time of the last update plus the number of
revisions.

Summary line containing the following:

  Number of blocks used

  Number of blocks allocated

  Number of files printed
```

Example

```
PIP>DT1:/FU

DIRECTORY DT1:[5,5]
10-AUG-83 10:51

T20ACP.TSK;1506      (7,1)       8./8.          C  09-AUG-83 13:10
  [5,5]              [RWED,RWED,RWED,R]
BOO.TSK;1506         (10,1)      19./19.        C  09-AUG-83 13:11
  [5,5]              [RWED,RWED,RWED,R]
COP.TSK;1506         (11,1)      8./8.          C  09-AUG-83 13:12
  [5,5]              [RWED,RWED,RWED,R]
DMO.TSK;1506         (12,1)      5./5.          C  09-AUG-83 13:13
  [5,5]              [RWED,RWED,RWED,R]
INI.TSK;1506         (13,1)      23./23.        C  09-AUG-83 13:15
  [5,5]              [RWED,RWED,RWED,R]
PIP.TSK;1506         (14,1)      56./56.        C  09-AUG-83 13:16
  [5,5]              [RWED,RWED,RWED,R]
RED.TSK;1506         (15,1)      6./6.          C  09-AUG-83 13:17
  [5,5]              [RWED,RWED,RWED,R]
SAV.TSK;1506         (16,1)      23./23.        C  09-AUG-83 13:18
  [5,5]              [RWED,RWED,RWED,R]
UFD.TSK;1506         (17,1)      9./9.          C  09AUG-83  13:20
  [5,5]              [RWED,RWED,RWED,R]
ZAP.TSK;1506         (20,1)      38./38.        C  09-AUG-83 13:21
  [5,5]              [RWED,RWED,RWED,R]
RSX-20F.SYS;1506     (21,1)      59./59.        C  09-AUG-83 13:22
  [5,5]              [RWED,RWED,RWED,R]
RSX-20F.MAP;1        (22,1)      153./153.         09-AUG-83 13:23
  [5,5]              [RWED,RWED,RWED,R]

TOTAL OF 407./407. BLOCKS IN 12. FILES
```

/TB displays only the summary line in the following format:

TOTAL OF nnnn./mmmm. BLOCKS IN xxxx. FILES

where:

nnnn = blocks used

mmmm = blocks allocated

xxxx = number of files

Example

```
PIP>DT1:/TB

TOTAL OF 407./407.  BLOCKS IN 12.  FILES
```

MERGE Switch (/ME)

Function

The /ME switch is used to create a new file from two or more existing files. If an explicit output file specifier is used and more than one input file is named without an appended switch, the /ME switch becomes the default switch.

Format

```
PIP>outfile = infile1, infile2,...infileN[/ME]
```

Example

The following command concatenates version 1 of the file TEST.DAT and version 2 of NEW.DAT from DK2: and generates the file SAMP.DAT on DK1:.

```
PIP>DK1:SAMP.DAT=DK2:TEST.DAT;1,NEW.DAT;2/ME
```

PURGE Switch (/PU)

The /PU switch allows you to delete obsolete versions of a file.

Format

```
PIP>infile1[,infile2,...,infileN]/PU
```

The /PU switch provides you with a convenient way to delete old versions of files. The /PU switch deletes all but the latest version of a file.

Example

Before issuing the /PU switch the following files are in a directory.

```
TEST.DAT;1,TEST.DAT;2,TEST.DAT;5
```

Then the following command and switch are issued:

```
PIP>TEST.DAT/PU
```

After PIP has completed the purging action, the directory contains the following file.

```
TEST.DAT;5
```

RENAME Switch (/RE)

Function

The /RE switch allows you to change the name of a file. The subswitch /NV allows you to force the version number of the renamed file to be one greater than the latest version number of the previously existing file.

Format

```
PIP>outfile = infile/RE[/NV]
```

Example

```
PIP>TEST.DAT;1=TRIAL.DAT;5/RE
```

File TRIAL.DAT;5 is renamed TEST.DAT;1.

NOTE

Renaming files across devices is not allowed. However, renaming across directories on the same device is allowed.

6.4.5 PIP Error Messages

PIP -- ALLOCATION FAILURE -- NO CONTIGUOUS SPACE

The contiguous space available on the output volume is insufficient for the file being copied.

PIP -- ALLOCATION FAILURE ON OUTPUT FILE

Space available on the output volume is insufficient for the file being copied.

PIP -- ALLOCATION FAILURE - NO SPACE AVAILABLE

Space available on the output volume is insufficient for the file being copied.

PIP -- BAD USE OF WILD CARDS IN DESTINATION FILE NAME

The user specified a wildcard "*" for an output filename where the use of a wildcard is explicitly disallowed.

PIP -- CANNOT FIND DIRECTORY FILE

You specified a UFD that does not exist on the specified volume.

PIP -- CANNOT FIND FILE(S)

The file(s) specified in the command was not found in the specified directory.

PIP -- CANNOT RENAME FROM ONE DEVICE TO ANOTHER

You attempted to rename a file across devices.

PIP -- CLOSE FAILURE ON INPUT FILE

The input file cannot be properly closed. The file is locked to indicate possible corruption.

PIP -- CLOSE FAILURE ON OUTPUT FILE

The output file cannot be properly closed. The file is locked to indicate possible corruption.

PIP -- COMMAND SYNTAX ERROR

You entered a command that does not conform to the syntax rules.

PIP -- DEVICE NOT MOUNTED

The specified device is not mounted.

PIP -- DIRECTORY WRITE PROTECTED

PIP cannot remove an entry from a directory because the directory is write-protected or because of a privilege violation.

PIP -- ERROR FROM PARSE

The specified directory file does not exist.

PIP -- FAILED TO ATTACH OUTPUT DEVICE

An attempt to attach a record-oriented output device failed. This is usually caused by the device being off line or not being resident.

PIP -- FAILED TO DETACH OUTPUT DEVICE

An attempt to detach a record-oriented device failed.

PIP -- FAILED TO DELETE FILE

You attempted to delete a protected file.

PIP -- FAILED TO ENTER NEW FILE NAME

You specified a file that already exists in the directory file, or you do not have the necessary privileges to make entries in the specified directory file.

PIP -- FAILED TO FIND FILE(S)

The file(s) specified in the command line was not found in the specified directory.

PIP -- FAILED TO GET TIME PARAMETERS

An internal system failure occurred while PIP was trying to obtain the current date and time.

PIP -- FAILED TO OPEN STORAGE BITMAP FILE

PIP cannot read the specified volume's storage bit map, usually because of a privilege violation.

PIP -- FAILED TO READ ATTRIBUTES

Your volume is corrupted or you do not have the necessary privileges to access the file.

PIP -- FAILED TO REMOVE DIRECTORY ENTRY

PIP cannot remove an entry from a directory because the entry is write-protected, or a privilege violation was detected.

PIP -- FILE IS LOST

PIP removed a file from its directory, failed to delete it, and failed to restore the directory entry.

PIP -- FAILED TO WRITE ATTRIBUTES

Your volume is corrupted or you do not have the necessary privileges to write the file attributes.

PIP -- ILLEGAL COMMAND

You entered a command not recognized by PIP.

PIP -- ILLEGAL SWITCH

You specified an illegal PIP switch or used a legal switch in an illegal manner.

PIP -- ILLEGAL "*" COPY TO SAME DEVICE AND DIRECTORY

You attempted to copy all versions of a file into the same directory that is being scanned for input files. This results in an infinite number of copies of the same file.

PIP -- ILLEGAL USE OF WILD CARD VERSION

The use of a wildcard version number in the attempted operation results in inconsistent or unpredictable output.

PIP -- I/O ERROR ON INPUT FILE

or

PIP -- I/O ERROR ON OUTPUT FILE

One of the following conditions exist:

- The device is off-line.
- The device is not mounted.
- The hardware failed.
- The volume is full (output only).
- The input file is corrupted.

PIP -- EXPLICIT OUTPUT FILENAME REQUIRED

You failed to specify the output filename.

PIP -- NO DIRECTORY DEVICE

You issued a directory-oriented command to a device that does not have directories.

PIP -- NOT ENOUGH BUFFER SPACE AVAILABLE

PIP has insufficient I/O buffer space to perform the requested command.

PIP -- NO SUCH FILE(S)

The file(s) specified in the command are not in the designated directory.

PIP -- ONLY [*,*] IS LEGAL AS DESTINATION UIC

You specified a UIC other than [*,*] as the output file UIC for a copy.

PIP -- OPEN FAILURE ON INPUT FILE

or

PIP -- OPEN FAILURE ON OUTPUT FILE

The specified file cannot be opened. On EOF one or more of the following conditions can exist:

- The file is protected against access.
- A problem exists on the physical device.
- The volume is not mounted.
- The specified file directory does not exist.
- The named file does not exist in the specified directory.

PIP -- OUTPUT FILE ALREADY EXISTS - NOT SUPERSEDED

An output file of the same name, type, and version as the specified file already exists.

PIP -- TOO MANY COMMAND SWITCHES - AMBIGUOUS

You specified too many switches or conflicting switches.

PIP -- VERSION MUST BE EXPLICIT OR "*"

The version number of the specified file must be expressed explicitly or as a wildcard.

6.5 RED

Sections 6.5.1 through 6.5.4 describe the function and format of RED, provide examples and show error messages.

6.5.1 Function

The RED utility allows the operator to redirect all I/O requests previously directed to one system device to another system device. The utility does not affect any I/O requests already in the I/O queue.

To initiate the RED utility, type the following:

```
CTRL/\ (Control backslash)

PAR>MCR RED

RED>
```

6.5.2 Format

```
RED>nud:=SY:
```

where:

nud — the new device to which subsequent requests are to be redirected.

SY — the system device from which requests have been directed.

NOTE

The device nud must be mounted before the RED command is given

6.5.3 Examples

The following command redirects all I/O requests for SY: to DB1:.

```
RED>DB1:=SY:
```

6.5.4 Error Messages

RED -- DEVICE NOT KNOWN TO SYSTEM

An attempt was made to redirect a device that does not exist in the device tables.

RED -- F11ACP NOT FOUND ON SYSTEM
SYSTEM MUST BE RELOADED

RED could not find the Files-11 ACP on the new system device. This situation forces the front end to crash.

RED -- PRIMARY PROTOCOL RUNNING

You attempted to redirect the system device while primary protocol was running. This is not allowed.

RED -- NEW SYSTEM NOT MOUNTED
F11ACP NOT FOUND ON SYSTEM
SYSTEM MUST BE RELOADED

RED -- SYNTAX ERROR

The command string contained a syntax error.

6.6 SAV

Sections 6.6.1 through 6.6.5 describe the function and format of the SAV utility, provide examples, and show error messages.

6.6.1 Function

The SAV utility writes into a contiguous task image file the image of an RSX-20F system that has been resident in main memory. The utility saves the image so that you can later use a hardware bootstrap or the BOOT command to reload and restart the system. The saved system is written into the file from which it was originally booted. This utility provides a way to save a patched system image.

The SAV utility removes any installed tasks that were not loaded from LB: and verifies that the system is inactive by making the following checks:

- No tasks have outstanding I/O.
- No devices are mounted.
- No checkpoint files are active.
- Error logging has been turned off.

An error is reported if any of these checks fail.

All RSX-20F system images reside on a file structure volume as a special format of task image. This special image is a task image without a task header.

A system can either be booted by using the hardware bootstrap, or by using the BOOT command. A system saved on one controller cannot be booted from another controller.

When a user installs a task, the system stores the task's file identification in the task header. When a system is saved, it places the file identification rather than the files's logical block number in the task control block. When the system is rebooted, it reopens the task file and stores the new logical block number of the task in the task control block. If a task has been deleted, the system cannot open the task file when the system is rebooted. In this case the system automatically removes the task's control block from the system task directory.

A saved system does not retain the physical disk addresses of installed tasks. However, the task control block entries contain task file identifications, rather than logical block numbers after a system save. Thus, the system can function normally when it is rebooted.

When the bootstrap block is written, the physical disk-block address of the system-image file is stored with it. However, the file can be deleted. If file system activity occurs, the blocks previously allocated to the system image can be reallocated to another file. A subsequent bootstrap that uses the boot block can cause random data to be loaded.

Since SAV is active when the memory-resident system image is copied to disk, SAV appears in this image. In fact, SAV is the program that starts up the saved system after a disk boot.

To run SAVE, type:

```
CTRL/\ (control backslash)

PAR>MCR SAV

SAV>
```

6.6.2 Format

```
SAV> [/switch1/switch2.../switchN]
```

where:

switch is one of the following:

/DM:dev causes the specified device to be dismounted when the save file has been written. This switch is not widely used.

/EX causes SAV to exit after writing the save file. This switch is the default condition.

/MO:dev tells SAV that the specified device must be mounted before the save file is written. This switch is not widely used.

/RH indicates that SAV should read the home block to find the front-end file system. This switch is the default condition.

/WB indicates that a boot block pointing to the system image is to be written out to the system device. The new boot block points to the file that is saved by the execution of this command. Thus, on the next hardware bootstrap, this saved file will be loaded. If the command omits the /WB switch, the file previously pointed to by the boot block remains in effect; that is, the file is not overwritten.

/WS causes SAV to write a save file. This switch is the default condition.

6.6.3 Example

```
SAV>
```

The current status of the system is saved on the system disk (because /WS is the default switch). System changes made by the RED utility or another utility are also saved with the system image that is resident in main memory.

6.6.4 Error Messages

SAV -- *DIAG* -- CANNOT FIND SECOND DX:

In attempting to boot the system from floppy disks you must have both floppies mounted. SAV was unable to find the second floppy disk it expected.

SAV -- *DIAG* -- DBn: NOT IN PROGRAMMABLE (A/B) MODE

SAV attempted to access disk DBn: and found that it was not in the correct mode.

SAV -- *DIAG* -- DEVICE ALREADY MOUNTED

You have used the /MO switch in the command string to SAV and the device you requested is already mounted.

SAV -- *DIAG* -- dev NOT READY

SAV attempted to use the specified device and found that it was not ready.

SAV -- *DIAG* -- KLINIK LINE ACTIVE IN REMOTE MODE

SAV discovered an active KLINIK line. You may receive this message when the KLINIK line was actually in REMOTE mode previous to the system load. You may also receive it when the KLINIK parameters were not saved due to some condition at the time of the crash. In this case, SAV resets the KLINIK line to REMOTE mode by default, since it has no way to know what mode the KLINIK line was in without the parameters.

SAV -- *DIAG* -- KLINIK LINE ACTIVE IN USER MODE

SAV discovered an active KLINIK line during the initialization procedure. SAV checked for saved KLINIK parameters, and found that the KLINIK line had been in USER mode before the crash.

SAV -- *DIAG* -- KLINIK LINE CONNECTED TO SYSTEM CONSOLE

SAV found an active KLINIK line while bringing up the system. Either the KLINIK line was in REMOTE mode when the system went down, or the KLINIK parameters were lost during the crash and SAV has restored the KLINIK line to REMOTE mode by default. REMOTE mode means that the KLINIK line user has the use of a remote CTY (refer to Appendix D for more information on KLINIK). This message will immediately follow the KLINIK LINE ACTIVE IN REMOTE MODE message.

SAV -- *DIAG* -- MOUNT dev ERROR nn

SAV got an I/O error when attempting to mount the specified device. Refer to Appendix A for a list of the I/O error codes.

SAV -- *DIAG* -- NO TOPS FILE SYSTEM ON dev

SAV could not find the file system owned by the KL's operating system on the specified boot device.

SAV -- *DIAG* -- TOPS HOM BLOCK CONSISTENCY ERROR OR DBn:

The two home blocks which are read and compared by SAV are not consistent with each other.

SAV -- *DIAG* -- TOPS HOM BLOCK READ ERROR nn ON DBm:

SAV got a read error while attempting to read and compare the home blocks. Refer to Appendix A for a list of the I/O error codes.

SAV -- *FATAL* -- ACP FOR devl NOT ON dev2

You have used the /DM switch, and SAV was not able to find the ACP it expected for device devl on device dev2.

SAV -- *FATAL* -- CREATE SAVE FILE (5,5) ERROR xx

SAV attempted to create a system image file and received an I/O error. Refer to Appendix A for a list of I/O error codes.

SAV -- *FATAL* -- dev CANNOT BE DISMOUNTED

You gave SAV the /DM switch and SAV was unable to dismount the specified device.

SAV -- *FATAL* -- dev DISMOUNT ERROR xx

SAV got an I/O error while attempting to dismount the specified device. Refer to Appendix A for a list of I/O error codes.

SAV -- *FATAL* -- DEVICE dev NOT IN SYSTEM

The specified device does not exist as part of the system resources.

SAV -- *FATAL* -- DTE-20 #n NOT AT PRIORITY LEVEL 6

SAV accessed DTE20 #n and found that it was not at the correct priority level. All DTE20s should be at level 6.

SAV -- *FATAL* -- DTE-20 PROTOCOL RUNNING

Some type of DTE20 protocol is running. SAV cannot run while protocol of any type is in force.

SAV -- *FATAL* -- ILLEGAL DEVICE dev

SAV does not recognize the device identifier.

SAV -- *FATAL* -- ILLEGAL MODIFIER /xx

SAV discovered an illegal switch value: /xx.

SAV -- *FATAL* -- KLI TASK REQUEST ERROR nn

There was an I/O error on the request for KLINIT to run. Refer to Appendix A for a list of the I/O error codes.

SAV -- *FATAL* -- MOUNT ERROR

You specified the /MO switch in the SAV command string and SAV was unable to mount the specified device.

SAV -- *FATAL* -- NO DTE-20

SAV could not find the DTE20 it expected.

SAV -- *FATAL* -- PROTOCOLS NOT RUNNING

SAV expected to find one of the protocols running, but none were there. No KLINIK parameters can be passed to the KL if there is no protocol running.

SAV -- *FATAL* -- SAVE FILE (5,5) NOT CONTIGUOUS

SAV was unable to find enough contiguous space to write the requested save file.

SAV -- *FATAL* -- SYNTAX ERROR xx

SAV discovered a syntax error in your command string, namely xx.

SAV -- *FATAL* -- WRITE ERROR

While you were trying to boot from the save file, SAV discovered that the save file was not written correctly.

6.7 UFD - USER FILE DIRECTORY

Sections 6.7.1 through 6.7.4 describe the function and format of the UFD utility, provide examples and show error messages.

6.7.1 Function

The UFD utility creates a User File Directory on a Files-11 volume and enters its name into the Master File Directory. Before the User File Directory can be defined, the volume must be mounted with the MOU utility and initialized with the INI utility. Once the volume has been mounted and initialized, User File Directories can be added at any time.

To run UFD, type:

```
CTRL/\ (control backslash)

PAR>MCR UFD

UFD>
```

6.7.2 Format

```
UFD>dev:[group,member][/switch][/switch]
```

where:

dev:	The device containing the volume on which the UFD being created will reside. Default: none; must be specified.
[group,member]	The owning UIC for the UFD. The brackets are required syntax.
switch	The UFD utility accepts the following switch.
/ALL	The /ALL switch allocates space for the UFD and takes as an argument the number of blocks to allocate. Default: /ALL:32

6.7.3 Examples

```
PAR>MCR UFD
UFD>DB2:[10,10]/ALL:900
```

6.7.4 Error Messages

UFD -- CAN'T READ MCR COMMAND BUFFER

This message indicates that UFD was unable to parse your command string.

UFD -- DEVICE NOT IN SYSTEM

UFD was unable to find the specified device on the system.

UFD -- DIRECTORY ALREADY EXISTS

The requested UFD already exists on the volume.

UFD -- FAILED TO CREATE DIRECTORY

No space exists on the volume, or an I/O error occurred.

UFD -- FAILED TO ENTER IN MFD

No space exists in the MFD or on the volume, or an I/O error occurred on the volume.

UFD -- NOT FILES-11 DEVICE

The device specified for the UFD was not a Files-11 device, and therefore could not support a UFD.

UFD -- SYNTAX ERROR

UFD found a syntax error in your command string.

UFD -- VOLUME NOT MOUNTED

The volume was not mounted prior to the attempt to create the UFD.

UFD -- WRITE ATTRIBUTES FAILURE

UFD encountered an error while writing the attributes of either the MFD or the newly created UFD.

6.8 ZAP

Sections 6.8.1 through 6.8.8 describe the ZAP utility.

6.8.1 Function

The ZAP utility allows you to examine files on a Files-11 volume and to patch task images and data files in an interactive environment without reassembling the files.

ZAP provides the following features:

- Command line switches that allow access to specific words and bytes in a file, modify locations in a task image, list the disk block and address boundaries for each overlay segment in a task image, and open a file in read-only mode.
- A set of internal registers that includes eight relocation registers.
- Single character commands that, in combination with other command line elements, display, open, close, and manipulate the values in task images and data files.

NOTES

The results of ZAP are permanent. The most convenient way to use ZAP is with a hard-copy terminal. Hard copy provides a record of the changes made with ZAP commands.

Although using the ZAP utility is relatively uncomplicated, patching locations into the task image requires that you know how to use the map generated by the task builder along with the listings generated by MACRO-11. These maps and listings provide the information needed to access the locations to be changed.

6.8.2 Invoking and Terminating ZAP

The method for invoking ZAP is as follows:

```
CTRL/\ (control backslash)
PAR>MCR ZAP

ZAP>filespec[/sw...] <cr>
 command line
```

To exit from ZAP, type either of the following:

```
X

or

CTRL/Z
```

6.8.3 ZAP Switches

ZAP switches set the mode in which ZAP operates: task image mode, absolute mode, or read-only mode. For example, you can select task image mode by omitting the /AB switch. The three ZAP switches are presented below.

/AB — Processes the addresses entered in the ZAP command lines as absolute byte addresses within the file RSX-20F.SYS (not within the program). You must add 2000 (octal) to the absolute address inside the code to compensate for the header information at the top of the file.

If /AB is not specified, addresses in ZAP command lines refer to addresses in a task image file, as shown in the task-builder task image map for the file.

/LI Displays the starting disk block and address boundaries for each overlay segment in the file in the following form:

ssssss: aaaaaa-bbbbbb

where:

ssssss: specifies the starting block in octal

aaaaaa specifies the lower address boundary in octal

bbbbbb specifies the upper address boundary in octal

/RO Opens a file in read-only mode. When /RO is specified, ZAP functions that change the contents of locations can be executed, but the changes are not permanent. When ZAP exits, the original values in the task image file are restored.

6.8.4 Addressing Locations in a Task Image

To make addressing a task image more convenient, ZAP provides two modes of addressing a task image and a set of internal relocation registers. The two modes of addressing are absolute mode and task mode. These two modes aid in the figuring of relocation biases.

When MACRO-11 generates a relocatable object module, the base address of each program section in the module is 000000. In the assembly listing, all locations in the program section are shown relative to this base address.

The task builder links program sections to other program sections by mapping the relative addresses applied by the assembler to the physical addresses in memory (for unmapped systems), or to virtual memory locations (for mapped systems).

Many values within the resulting task image are biased by a constant whose value is the absolute base address of the program section after it has been relocated. This bias is called the relocation bias for the program section.

ZAP's eight relocation registers, 0R through 7R, are generally set to the relocation biases of the modules that will be examined. Thus, you can reference a location in a module by the same relative address that appears in the MACRO-11 listing. ZAP provides two addressing modes that simplify the calculation of relocation biases.

6.8.4.1 ZAP Addressing Modes: Absolute and Task Image - ZAP provides two modes of addressing locations in a task image: absolute mode and task image mode.

To use ZAP in absolute mode, enter the /AB switch with the file specifier when ZAP is invoked.

In absolute mode, ZAP interprets the first address in the file being changed as segment 1, location 000000. All other addresses entered are interpreted using this address as a base location. This mode allows access to all the bytes in a file, as well as the label and header blocks of the task image. However, to modify a task image in absolute mode, the layout of the task image on disk must be known. Generally, this is practical only for task image files that are not overlaid. In absolute mode the task header is 2000 bytes. Therefore, to access location 0 of a nonoverlaid task in absolute mode, open address 2000.

In task image mode, ZAP uses the block number and relative offset listed in the task builder's memory allocation map to address locations. This mode is useful for changing locations in a file constructed of overlay segments because the task builder and ZAP perform the calculations necessary to relate the task's disk structure to its run-time memory structure.

The task builder adds blocks that contain system information to the beginning of the task image file. The memory allocation map generated by the task builder gives the starting block and byte offset of the file to be changed.

Task image mode is the default mode for ZAP. You may also put ZAP in task image mode by entering the /LI switch to display block/segment information. This puts ZAP in task image mode after the information is displayed.

Locations in a file can be examined in either absolute mode or task image mode by using the /RO switch. This switch allows locations to be opened and the contents temporarily changed. When ZAP exits, the original file remains intact.

6.8.4.2 Addressing Locations in Task Image Mode - In task image mode, ZAP uses the block number and byte offset listed in the task builder memory allocations map, and addresses that MACRO-11 prints in an object module listing to access a location in a task image. The following excerpts from a MACRO-11 listing and a task image memory allocation map generated by the task builder show how to use ZAP in task image mode.

The following lines represent assembled instructions from a MACRO-11 source listing:

```
71 000574  032767  000000G 000000G     BIT  #FE.MUP,$FMASK
72 000602  001002                     BNE  2$
73 000604  000167  000406             JMP  30$
74 000610  061700  000000G         2$  MOV  $TKTCB,R0
75 000614  016000  000000G             MOV  T.UCB(R0),R0
76 000620  010067  177534              MOV  R0,UCB
```

The following excerpt from the task builder memory allocation map gives the information needed to address locations in the task image file as they appear in the above MACRO-11 listing:

```
R/W MEM   LIMITS: 120000 123023 003024 01556.
DISK BLK  LIMITS: 000002 000005 000004 00004.

MEMORY ALLOCATION SYNOPSIS:

SECTION
. BLK.:(RW,I,LCL,REL,CON0  120232 002546 01382.
                           120232 002244 01188.

(TITLE: MYFILE, IDENT: 01, FILE: MCR.OLD;1)

                           122476 000064 00052.
$$RESL:(RE,I,LCL,REL,CON)  123000 000024 00020.

(TITLE: FMTDV, IDNET: 01, FILE: MCR.OLB;1)
```

With the information in the memory allocation map above, the user can determine the block number and byte offset for the beginning of the file to be changed. The disk-block-limits line lists block 2 as the block where the file begins. The memory allocation synopsis lists byte offset 120232 as the beginning of the file MYFILE. To address location 574 in the MACRO-11 listing in the task image mode, specify the command:

```
002:120232+574/<cr>
```

ZAP responds by opening the location and displaying its contents:

```
002:121026/032767
```

6.8.5 The ZAP Command Line

ZAP commands allow you to examine and modify the contents of locations in a task image file. Command lines comprise combinations of the following elements:

- Commands
- Internal registers
- Arithmetic operators
- Command line element separators
- The current location symbol
- Addresses of location in storage

These command elements can be combined to perform multiple functions. The functions of a given command line depend on the positional relationship of one command line element to the rest. In other words, the function specified on a ZAP command line depends on both the elements specified and on the form in which those elements are specified.

ZAP commands take effect only after a carriage return is pressed. Corrections to the command line can be made prior to a carriage return by using the delete key. The line currently being typed can be deleted using the CTRL/U.

ZAP commands are grouped into three categories, as follows:

- Open/close location commands
- General purpose commands
- Carriage return command

6.8.5.1 Open/Close Location Commands - Open/close location commands are nonalphanumeric ASCII characters that direct ZAP to perform two general types of operations, as follows:

- Open a location, display its contents, and store the contents in the quantity register.
- Close the open location after optionally modifying the contents and open another location as specified by a command.

ZAP Open/Close Commands

/	Open a location, display its contents in octal, and store the contents of the location in the quantity register (Q). If the location is odd, it is opened as a byte.
"	Open a location, display the contents of the location as two ASCII characters, and store the contents of the location in the quantity register (Q).
%	Open a location, display the contents of the location in RADIX-50 format, and store the contents of the location in the quantity register (Q).
\	Open a location as a byte, display the contents of the location in octal, and store the contents of the location in the quantity register (Q).
'	Open a location, display the contents as one ASCII character, and store the contents of the location in the quantity register (Q).
-	Close the currently open location as modified, use the contents of the location as an offset from the current location value, and open that location.
@	Close the currently open location as modified, use the contents of the location as an absolute address, and open that location.

ZAP Open/Close Commands

>	Close the currently open location as modified, interpret the low-order byte of the location as the relative branch offset, and open the target location of the branch.
<	Close the currently open location as modified, return to the location from which the last series of _, @, or > commands began, and open the next sequential location.

6.8.5.2 General Purpose Commands - ZAP provides six single-character general-purpose commands. Some can be entered on the command line with no other parameters; others must be entered with parameters. The following table summarizes the commands and their functions.

X	Exit from ZAP; return to MCR.
K	Compute the offset between the value of the nearest (less than or equal to) relocation register and the currently open location, display the offset value, and store it in the quantity register.
O	Display the jump and branch displacements from the current locations to a target location.
=	Display in octal, the value of the expression to the left of the equal sign.
R	Set the value of a relocation register.

6.8.5.3 Using the Carriage Return - The carriage return causes ZAP to close the current location as modified. Two sequential carriage returns open the next sequential location in the file.

6.8.5.4 ZAP Internal Registers - ZAP internal registers are fixed storage locations that are used as registers. The internal registers contain values set by ZAP and the user to make it easier to modify locations in a task image. ZAP provides the following internal registers:

0R - 7R	Relocation registers 0 through 7. These registers can be loaded with the base address of modules relocated by the task builder. They provide a convenient means for indexing into a module to change the contents of locations in the modules.
C	The constant register. Set this register to contain a 16-bit value, which can be specified as an expression.

F — The format register. This register controls the format of the displayed address. If the value of the F register is 0, ZAP displays the addresses relative to the largest value of any relocation register whose value is less than or equal to the address to be displayed. If the value of the F register is nonzero, ZAP displays addresses in absolute format. Zero is the initial value of the F register.

Q — The quantity register. The value in the quantity register is set by ZAP to contain the last value displayed on the terminal.

To access the contents of a register, specify a dollar sign ($) before the register when you enter a command, as shown below:

```
$C/
```

This example directs ZAP to display the contents of the constant register.

6.8.5.5 ZAP Arithmetic Operators - The arithmetic operators are single-character command-line elements that define an arithmetic operation in the command-line expression. In general, ZAP evaluates such expressions as addresses. ZAP provides the following arithmetic operators:

\+ — Add a value to another value.

\- — Subtract a value from another value.

* — Multiply a value by 50 (octal) and add it to another value. Used to form a RADIX-50 string.

A*B means A x 50 (base 8) + B.

These operators are used in expressions on the command lines. For example, rather than adding by hand all the displacements listed in the task builder memory allocation map, the following notation could be used.

```
002:120000+170/
```

This method for calculating such a displacement is faster and more accurate than calculating it by hand.

6.8.5.6 ZAP Command Line Element Separators - ZAP provides separators to delimit one command line element from another. Different separators are required for the type of ZAP command being executed. ZAP uses the following separators:

, — Separate a relocation register specification from another command line element.

; — Separate an address from an internal register specification. (used in expressions that are values for relocation registers)

: — Separate a block number base value from an offset into the block. (used in most references to locations in a file)

6.8.5.7 The Current Location Symbol - In expressions that evaluate to an address on a ZAP command line, a period (.) represents the last open location.

6.8.5.8 Formats for Specifying Locations in ZAP Command Lines - There are three formats for specifying locations in a ZAP command line. Each provides a means of indexing into the task image file, but the methods of indexing differ. The three formats are:

- Byte offset format
- Block number/byte offset format
- Relocation register, byte offset format

Byte Offset Format

The byte offset format specifies a location in the task image file as follows:

```
location
```

If ZAP is being used in absolute mode, ZAP interprets this specification as a byte offset from block 1, location 000000. If ZAP is being used in task image mode, ZAP interprets this specification as a byte offset from block 0, location 000000.

This format is useful only when ZAP is being used in absolute mode. For example, the following ZAP command opens absolute location 664:

```
664/
```

Block Number/Byte Offset Format

Block number/byte offset format allows you to specify a byte offset from a specific block. Enter this format as follows:

```
blocknum:byteoffset
```

This form for addressing locations can be used regardless of whether /AB has been used with the ZAP file specification.

The task builder prints a map that gives information on the overlay segments. For example:

```
 R/W MEM   LIMITS:  120000 123023 033024 01556.
 DISK BLK  LIMITS:  000002 000005 000004 00004.

 MEMORY ALLOCATION SYNOPSIS:

 SECTION                                          TITLE  INDENT  FILE

 .BLK.:(RW,I,LCL,REL,CON)  120232 002546 01382.
                           120232 002244 01188. MYFILE  01  MCR.OLB;1
 122476 000064 00052.                           FMTDV   01  MCR.OLB;1
 $$RESL:(RW,I,LCL,REL,CON)  123000 000024 00020.
```

In task image mode, ZAP allows you to enter the block number and byte offset displayed in the task builder memory allocation map. In this case, the disk-block-limits line shows MYFILE beginning on block 2; the memory allocation synopsis shows that MYFILE has an offset of 120232.

Relocation Register, Byte Offset Format

This format allows you to load a relocation register with the value of a location to be used as a relocation bias. This mode of addressing locations in a task image is as follows:

```
relocreg,byteoffset
```

Specify relocreg in the form nR, where n is the number of the relocation register. Byte offset can then be addressed from the value loaded in the relocation register as follows:

2:12032;3R	Set the value of relocation register 3.
3,574/	Open the location 574 bytes offset from block (segment) 3, location 12032.

6.8.6 Using ZAP Open/Close Commands

This section explains how to use ZAP open/close commands. It contains information on how to open locations in a task image file, modify those locations, and close the locations.

6.8.6.1 Opening Locations in a Task Image File - Any of the five ZAP commands (/, ", %, \, or ') can open a location in a task image file. Once the location is open, its contents can be changed.

The value ZAP displays depends on the format in which the value was stored. For example, the word value 001002 takes the following binary form:

```
00000010000010
```

If the location is opened in byte format, the value contained in both locations is 002.

Once a location is opened in a given format, ZAP continues to display any subsequently opened locations in that format until the format is changed by entering another special-character open command. For example, if the percent (%) command is used, the contents are displayed in RADIX-50 format. If consecutive carriage returns are entered, consecutive locations will be in RADIX-50 format.

6.8.6.2 Changing the Contents of a Location - When a location is opened using a special-character command, the contents can be changed by entering the new value and a carriage return. The example below shows how to open a location, change the location, and close the location.

002:120000/ 000000	Display the contents of a word location.
44444	Change the contents of the location by entering a value (44444) and close the location by a carriage return.
/	Display the new contents of the location by entering a slash (/) and a carriage return.

```
002:120000/ 44444
```

When ZAP displays the contents of the opened location, the format in which the value is displayed is indicated by the special command character immediately following the address portion of the location. In this example the slash (/) indicates that word locations are being opened and the contents displayed in octal.

6.8.6.3 Closing Task Image Locations - There are five ZAP special-character commands for closing a location in a task image. The carriage return also closes a location. All the ZAP close commands perform the following three functions:

- Close the current location.
- Direct ZAP to another location (such as the preceding location or a location referenced by the current location).
- Open the new location.

The examples below show how these commands work.

Close a Location -- Open the Preceding Location

The circumflex (^) command is used to close the current location, to direct ZAP to the preceding location, and to open that location.

```
002:120100/
002:120100/ 000000

002:120102/ 000111

002:120104/ 000222

002:120106/ 000333
^

002:120104/ 000222
```

The carriage return is used to close the first three open locations and open the next location. The circumflex (^) closes location 120106 and directs ZAP to open the preceding location, 120104.

Close a Location -- Open a Location at an Offset from the Location Counter

The underscore (_) command is used as follows:

- Closes the current location
- Directs ZAP to use the contents of this just-closed location as an offset
- Adds this offset to the next sequential location
- Opens that location

The following example illustrates the use of the underscore command:

```
002:120100/
002:120100/ 000000

002:120102/ 000121

002:120104/ 000222

002:120106/ 000022
002:120132/ 234102
```

The first locations are closed by carriage returns. Location 120106 is closed using the underscore command, which directs ZAP to use the contents of the just-closed location (22) as an offset to the next sequential location (120110), and to open that location (120132).

Close a Location -- Open a Location Offset from the Value of the Just-Closed Location

The @ command is used to close a location, to direct ZAP to use the contents of the just-closed location as the absolute address of a location, and to open that location. The following example illustrates the use of the @ command:

```
002:120100/
002:120100/ 005000

002:120102/ 005301

002:120104/ 120114
@
002:120114/ 124104
```

The first locations are closed using the carriage return. Location 120104 is closed using the @ command, which directs ZAP to use the value in that location (120114) as the absolute address of the next location to open, and to open that location (8120114).

Close a Location -- Open the Target Location of a Branch

The greater-than (>) command is used to close the current location, to direct ZAP to use the low-order byte of the just-closed location as a branch offset to the next sequential location, and to open that location. The example below illustrates the use of the > command:

```
002:120100/
002:120100/ 005000

002:120100/ 005301

002:120104/ 001020
>
002:120146/ 052712
```

The first locations are closed using the carriage return. Location 120104 is closed using the > command. ZAP takes the low-order byte (020) of this just-closed location, uses it as a branch offset to the next sequential location, and opens that location. Since the low-order byte refers to a word, and the machine is byte-addressable, the offset value (020) is multiplied by 2 and added to the next sequential address (120106). This yields the new address (120146) that ZAP then opens.

Close a Location -- Open the Location Where the Current Series of Commands Began

The less-than (<) command is used to close the current location, direct ZAP to the next sequential location from the location where the series of , @, or > commands was first issued, and to open that location. The example below illustrates the use of the < command.

```
002:120002/LI
002:120002/000212

002:120004/000002
>
002:120012/004001
>
002:120016/120412
<
002:120006/140236
```

The < command closes location (120016), returns to the location where the sequence of > commands began (120004), and opens the next sequential location (120006).

6.8.7 Using Zap General Purpose Commands

This section describes the functions of the three ZAP general purpose commands K, O, and =.

6.8.7.1 K Command: Compute Offset, Store in Quantity Register - The K command is used to compute the offset between the value of the nearest (less than or equal to) relocation register and the currently open location, to display the offset value, and to store it in the quantity register.

K can be entered in the following formats:

K	Computes the displacement in bytes from the address of the last open location and the value of the relocation register whose contents are closest to but less than the value of that address.
nK	Calculates the displacement in bytes from the last open location and relocation register n.
a;nK	Calculates the displacement between address a and relocation register n.

The following example illustrates the use of the K command:

```
  2:1172,0R
  2:1232;1R
  2:1202/
002:000020/000111
  K
 =0,000010
  0,100;1K
 01,000040
```

6.8.7.2 O Command: Display Branch, Jump Displacement from Current Location - The O command is used to display the branch and jump displacements from the current location to a target location. A branch displacement is the low-order eight bits of a branch instruction which, when executed, would branch to the target location. A jump displacement is the offset between the open location and the target location. This displacement is used in the second word of a jump instruction when such an instruction uses relative addressing. The O command can be entered in the following formats:

aO	Displays the jump and branch displacements from the current location to the target of the branch.
a;KO	Displays the jump and branch displacements from location a to target location K.

The following example illustrates the use of the O command:

```
 2:1172;0R
 0,101
002:000010/000005
 0,200
 00006>000003
 0,30;2:12020
 177756>17777767
```

6.8.7.3 = Command: Display Value of Expression - The equal sign (=) command is used to display (in octal) the value of an expression to the left of the equal sign.

The format for specifying the equal sign command is as follows:

```
expression=
```

The following example illustrates the use of the equal sign command:

```
 2:30/
002:000030/000000
 .+177756=
000006
```

6.8.8 ZAP Error Messages

ZAP -- ADDRESS NOT WITHIN SEGMENT

The address specified was not within the overlay segment specified.

ZAP -- CANNOT BE USED IN BYTE MODE

The commands @, #, and > cannot be used when a location is open as a byte.

ZAP -- ERROR IN FILE SPECIFICATION

The file specification was entered incorrectly.

ZAP -- ERROR ON COMMAND INPUT

An I/O error occurred while a command was being read; this could be a hardware error.

ZAP -- I/O ERROR ON TASK IMAGE FILE

An I/O error occurred while reading or writing to the file being modified; this could be a hardware error.

ZAP -- NO OPEN LOCATION

An attempt was made to modify data in a closed location.

ZAP -- NO SUCH INTERNAL REGISTER

The character following the dollar sign was not a valid specification for an internal register.

ZAP -- NO SUCH RELOCATION REGISTER

An invalid number for a relocation register was specified.

ZAP -- NO SUCH SEGMENT

The starting disk block was not the start of any segment in the task image.

ZAP -- NOT A TASK IMAGE OR NO TASK HEADER

An error occurred during the construction of the segment description tables. The problem could be that the file is not a task image, /AB was not specified, or the task image is defective.

ZAP -- NOT IMPLEMENTED

The command entered is recognized but not implemented by ZAP.

ZAP -- OPEN IMAGE FAILURE FOR TASK IMAGE FILE

The file to be modified could not be opened. Possibly the file does not exist, the file is locked, the device is not mounted, or the file is protected from write access.

ZAP -- SEGMENT TABLE OVERFLOW

ZAP does not have enough room in its partition to construct a segment table.

ZAP -- TOO MANY ARGUMENTS

More arguments were entered on a command line than are allowed.

ZAP -- UNRECOGNIZED COMMAND

ZAP did not recognize the command as entered.

CHAPTER 7

RSX-20F MONITOR

Before examining the internals of the RSX-20F monitor, let us recall the functions of an operating system. These functions are:

- To provide service to the I/O devices in the form of device drivers
- To control the scheduling of the device drivers via some monitor call and queue mechanism
- To control the scheduling of tasks
- To provide common routines that any program can use

With these functions in mind, we will proceed to discuss the structure of the RSX-20F operating system. This chapter details the RSX-20F Executive. It then describes RSX-20F tasks and the scheduling of these tasks, along with the actions performed by system traps. Finally, the terminal service routines are presented.

7.1 RSX-20F EXECUTIVE

RSX-20F differs from TOPS-10 and TOPS-20 in that it is not a paging or swapping system. All of the RSX-20F Executive is in memory all the time. RSX-20F also differs from TOPS-10 and TOPS-20 in that it uses the same location in memory all the time, instead of bringing in only part of the monitor and placing it wherever space can be found. RSX-20F does use overlays, but overlays are handled by the individual tasks, rather than by the Executive. The only tasks that use overlays are tasks such as the PARSER or KLINIT, that are too large to fit into the GEN partition.

The components of the RSX-20F Executive are shown in Figure 7-1. The location .EXEND always points to the bottom of Lower Core, while the location .EXEND+2 points to the bottom of the Free Pool. Using this information you can find the boundaries of the Executive. The file RSX-20F.MAP contains a current map of the Executive, along with the correct addresses at which to find selected portions of it.

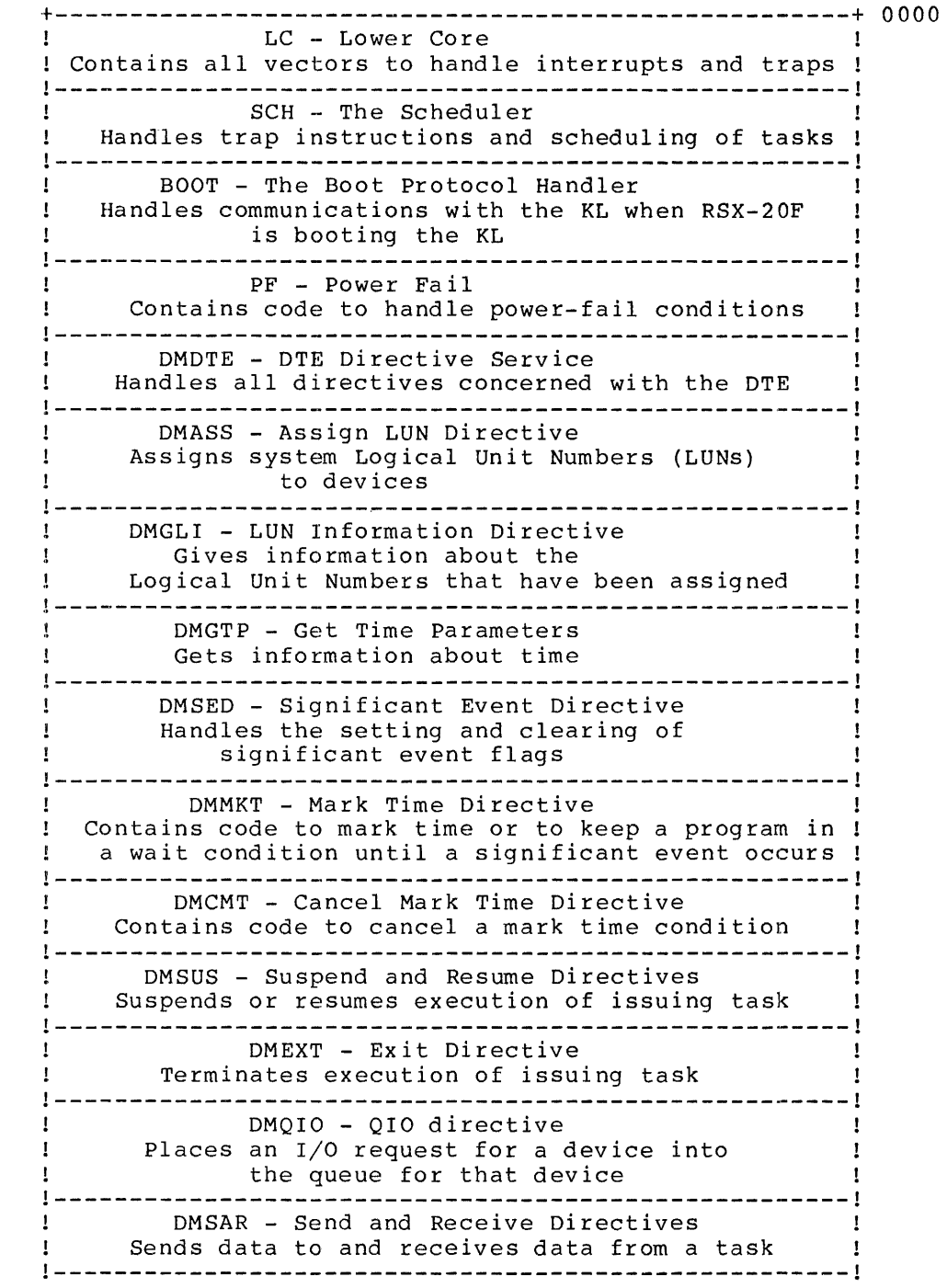


Figure 7-1: RSX-20F Executive

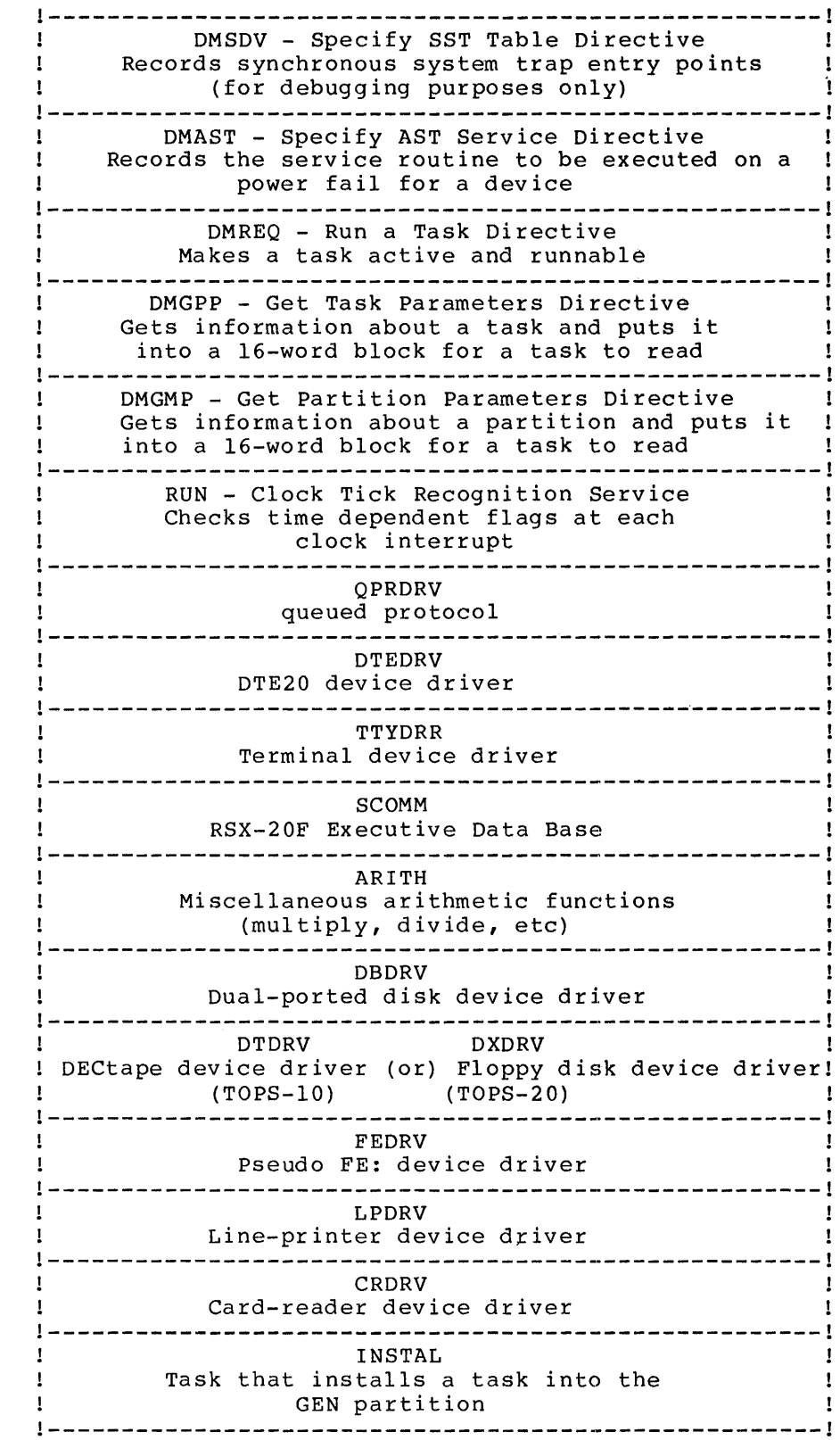


Figure 7-1: RSX-20F Executive (Cont.)

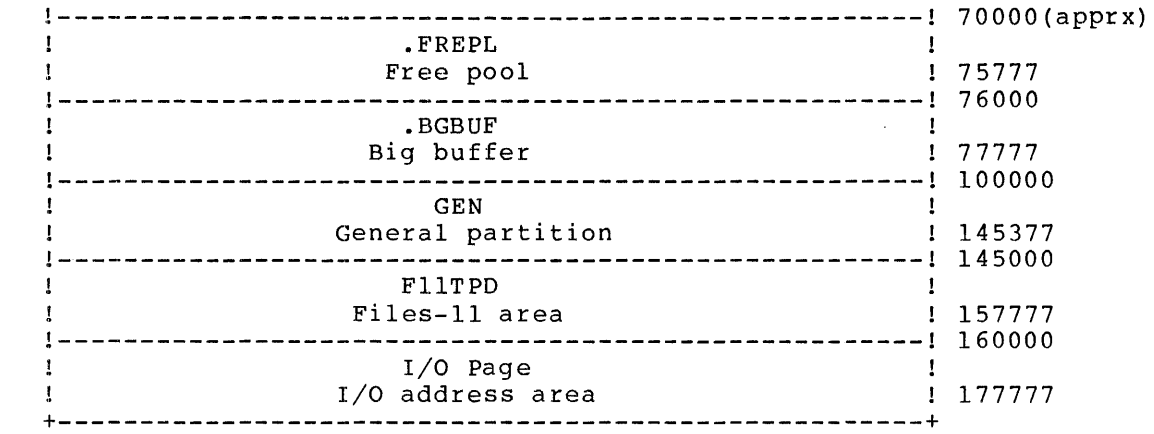


Figure 7-1: RSX-20F Executive (Cont.)

The bulk of the RSX-20F Executive is taken up by the directive service routines and the device drivers. The scheduler is small and not as involved as the scheduler in TOPS-20 or TOPS-10, because there are fewer tasks to schedule and they run quickly. (Scheduling is discussed in Section 7.2.) A representation of the entire memory is shown in Figure 7-2. The RSX-20F Executive takes up memory locations 000000 to 070000. The addresses below 070000 are not fixed; consult the file RSX-20F.MAP to find the actual addresses for the version of RSX-20F you have. Addresses above 070000 are fixed as pictured above. The area labeled .FREPL is a Free Pool of space for general use by the Executive. The TTY thread lists, task information, and LPT thread lists are stored in the Free Pool. The area labeled .BGBUF is a big buffer used to store LPT RAM data or task information when a task is being installed. The GEN partition is where the RSX-20F utility programs are executed. It is sometimes referred to as the user area. The F11TPD partition is a system partition and usually hosts the Files-11 Ancillary Control Processor (F11ACP). Other tasks that also use this partition are SETSPD, KLRING, KLDISC, and MIDNIT. The I/O page (also referred to as the external page) resides in upper memory and contains the input and output device registers.

With the aid of the Task Builder map for RSX-20F and the PDP-11 Peripherals Handbook it is possible to determine the contents of any location in memory. This data can be useful when using the switch registers to look at a crashed system. Not only are the locations of the hardware registers known, but also, many key software locations can be examined.

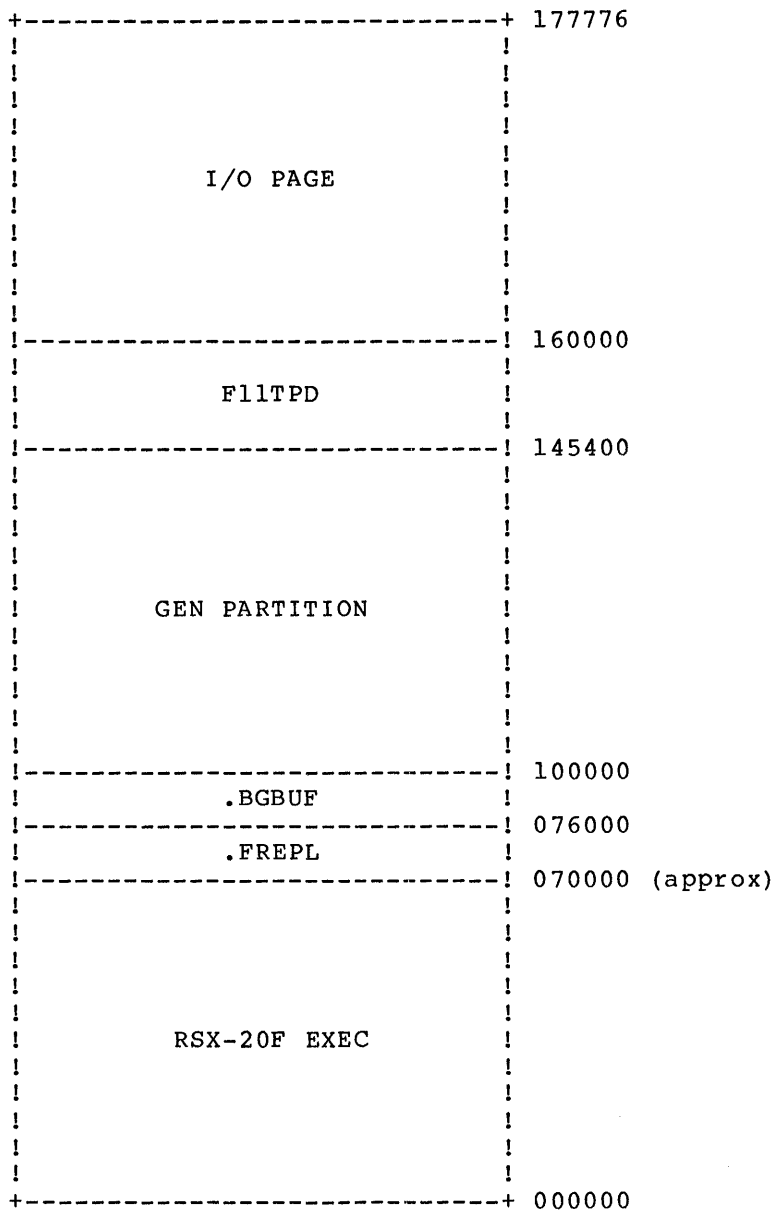


Figure 7-2: RSX-20F Memory Layout

7.2 TASKS AND SCHEDULING

The tasks that run in the front end are either part of the RSX-20F Executive or are utility programs. Executive tasks are resident in memory while the utilities are brought in from auxiliaries storage as needed. The following parts of the Executive are considered tasks and must be scheduled:

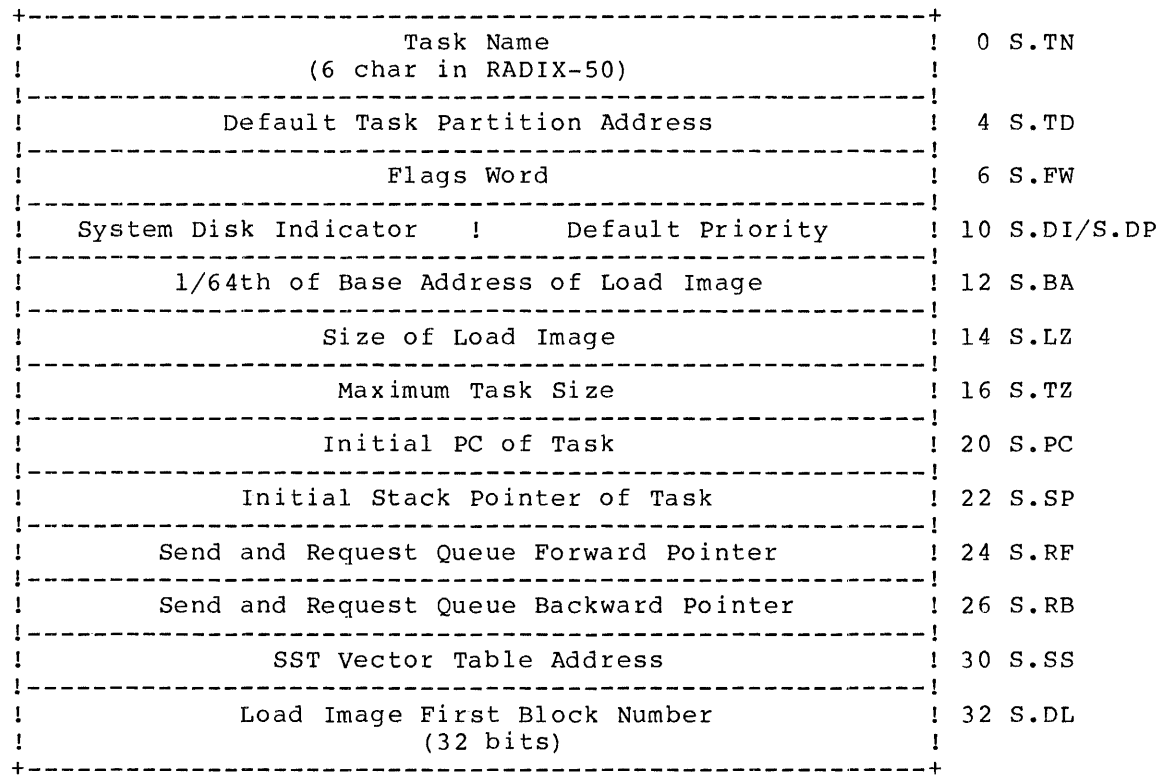
DTEDRV	DTE device driver
FEDRV	FE device driver
DBDRV	RP04 device driver
DXDRV	Floppy disk device driver
DTDRV	DECtape driver
TTYDRV	Terminal device driver
LPTDRV	LPT device driver

CDRDRV	Card-reader device driver
F11ACP	Files-11 Ancillary Control Processor
QPRDTE	Queued Protocol
INSTAL	Installs task into GEN partition
<optional>	Task chosen to run in GEN partition
NULL	Null task

Notice that it is a task (INSTAL) that installs the task in the GEN partition. The Executive has a system partition for its own use.

F11ACP stands for Files-11 Ancillary Control Processor. An Ancillary Control Processor (ACP) is a privileged task that extends the function of the Executive. F11ACP receives and processes file-related I/O requests on behalf of the Executive.

RSX-20F keeps several lists about tasks so that it knows what the tasks are doing. The System Task Directory (STD) is a list of all tasks installed into the system. Each task on the list has a 15-word block (referred to as an STD node) that contains the information shown in Figure 7-3.



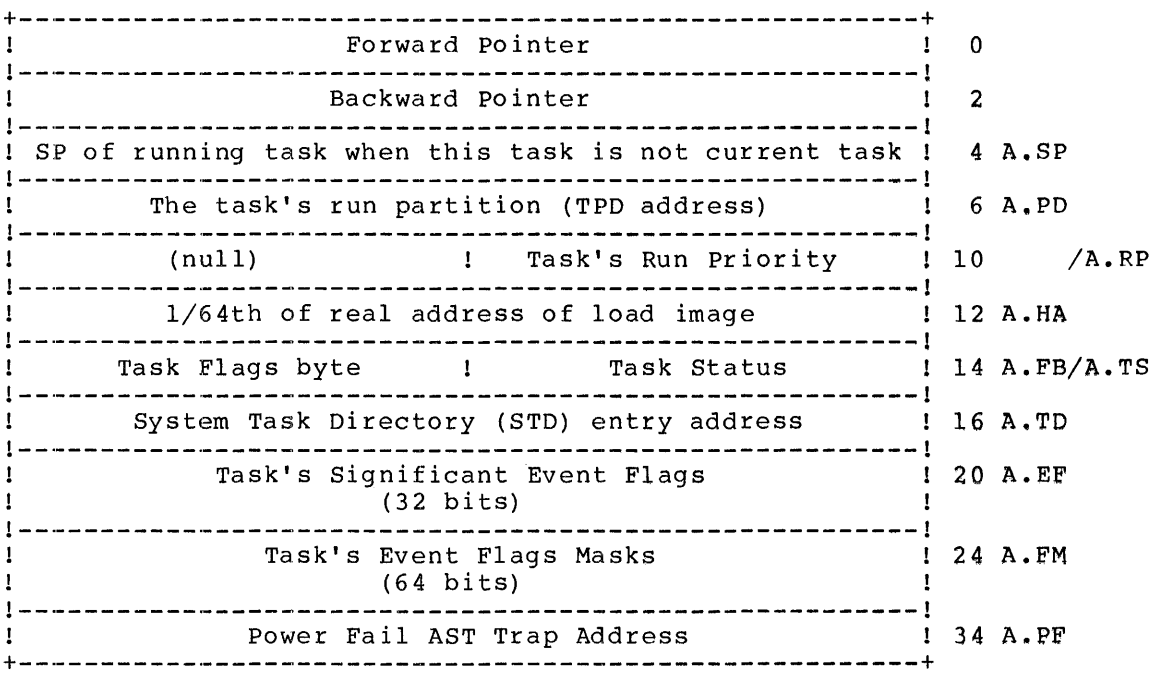
Task Flags (Bytes 6 and 7)

```
SF.TA==000001   Bit  0 - Set when task is active
SF.FX==000002   Bit  1 - Set when task is fixed in memory
SF.EX==000004   Bit  2 - Set when task to be removed on exit
SF.IR==040000   Bit 14 - Set when install is requested
SF.ST==100000   Bit 15 - Set when task is system task
```

Figure 7-3: System Task Directory (STD) Node

The 15-word blocks in the STD are pointed to by entries in the table at location .STDTB. This table has an entry for every installed task in the system.

RSX-20F keeps another list of those tasks wanting to run. This list is called the Active Task List (ATL). The ATL is a doubly linked list of nodes (entries) for active tasks that have memory allocated for their execution. The list is in priority order. Tasks with an entry in the ATL are either in memory or are being loaded into memory. A node in RSX-20F is a block of data that concerns a task. "Doubly linked" means that each node is linked to both the previous node and the following node. The ATL nodes have the format shown in Figure 7-4.



Status Bits (Byte 14)

TS.LRQ==02 Bit 1 - Task load request queued
TS.TKN==04 Bit 2 - Task waiting for termination notice
TS.LRF==06 Task load request failed
TS.RUN==10 Task is running
TS.SUS==12 Task is suspended
TS.WF0==14 Task is waiting for an event 1-14
TS.WF1==16 Task is waiting for an event 17-32
TS.WF2==20 Task is waiting for an event 33-48
TS.WF3==22 Task is waiting for an event 49-64
TS.WF4==24 Task is waiting for an event flag 1-64
TS.EXT==26 Task exited

Flag Bits (Byte 15)

AF.PP==200 Bit 7 set when task is primary protocol task

Figure 7-4: Active Task List (ATL) Node

When you look at dumps of RSX-20F, you can find the location of the ATL node of the current task by examining the location .CRTSK.

Installing a task into the GEN partition consists of reading it into memory from the system file area, putting the task into the STD and ATL, and setting the appropriate flags. The STD and ATL entries are located in the Executive Data Base.

One of the tasks in the Active Task List is the Null Task. The Null Task is the task that runs when no other task wants to run (a very quiet system) or no other task can run (tasks are blocked waiting for pending I/O).

Scheduling for all tasks is by a priority system. When a task is installed it has a priority that is reflected in its position in the ATL. The task with the highest priority goes first in the list, the next highest goes second, and so on. Scheduling occurs when a task has declared itself waiting for some significant event to occur, or when a directive service routine exits. Two separate entry points to the ATL scan routine provide for these two situations. Control is passed to the first, ASXE1, when a significant event is declared. The ATL is scanned from the beginning to the end to find the first runnable job. Control is passed to the second entry point, ASXE2, when a directive service routine exits. In this case, one of three things can happen:

- Control can be returned to the task that issued the directive.
- The ATL can be scanned for the next runnable task beginning with the task that issued the directive down the ATL through the lower priority tasks.
- The ATL can be scanned from the beginning.

7.3 SYSTEM TRAPS

A trap is a CPU-initiated interrupt that is automatically generated when a predetermined condition is detected. Two vector locations in low memory are dedicated for each trap type. The vector locations contain the PC and PS for the trap service routine. When the trap occurs, the current PC and PS are put on the stack and the contents of two vector locations are loaded as the new PC and PS. Traps can occur as the result of the following conditions:

Location	Trap
004	CPU errors
010	Illegal and reserved instructions
014	BPT
020	IOT
024	Power Fail
030	EMT
034	TRAP
114	MPE

Traps can be either asynchronous or synchronous. An asynchronous trap occurs as the result of an external event such as the completion of an I/O request. In this case, the task is doing other work or waiting for the I/O to be done. A synchronous trap occurs immediately upon the issue of the instruction that causes the trap.

The PDP-11 instruction set contains several instructions that cause traps. The EMT instruction, generally reserved for system software, causes a trap to an emulator routine. This instruction is used by RSX-20F to perform directives. Whenever a directive must be performed, the necessary information is loaded into the registers and an EMT is issued. The EMT instruction traps to a routine that decides which directive is to be performed. A TRAP instruction is like the EMT instruction, except that it is used by user tasks. The only difference between TRAP and EMT is a different vector location. IOT is used by RSX-20F for error reporting. When RSX-20F detects an error that is considered serious enough to crash the system, an IOT instruction is issued.

Power fail conditions cause an automatic trap independent of the software operations mentioned above.

There are two places in RSX-20F where traps are handled. The following events cause a trap to location COMTRP:

- IOT instruction
- TRAP instruction
- BPT (break point trap)
- Trap to 10 (illegal instruction)
- Trap to 4 (device or memory timeout)
- Illegal Interrupts
- Parity Error

COMTRP has to sort out the type of error it gets. If possible, only the offending task is terminated. If COMTRP concludes that this error is serious enough to crash the system, the COMTRP routine issues a .CRASH macro itself. This causes control to come right back to COMTRP. COMTRP sees that it was an IOT instruction that occurred and dispatches to IOTTRP. During this process, the COMTRP routine performs the following functions:

1. Tries to restore the user task that had the problem
2. Issues an IOT error instruction
3. Dispatches to the IOT handling routine

When it is called, the IOTTRP routine performs the following functions:

1. Sets up the emergency stack pointer
2. Sets up the emergency message pointer
3. Saves the registers
4. Saves the crash code
5. Saves the parity-error data
6. Prints the 11-Halt message on the CTY (and KLINIK)

7. Requests the KL to reload the PDP-11 via the DTE20

8. Rings the KL doorbell

9. Loops through the previous step until the PDP-11 is reloaded

The routine to handle EMT instructions is comparable in a way to JSYS's under TOPS-20. Since the PDP-11 is a smaller system, it cannot have one instruction for every directive it wants to run. Therefore, that which is handled by hardware on the KL is handled by a combination of hardware and software on the PDP-11. The KL handles the instruction by dispatching to the right routine. The PDP-11 issues the trap and then the software checks the stored argument to decide which routine is to be called.

7.4 TERMINAL SERVICE ROUTINES

RSX-20F handles terminals that access the system over dial-up lines in a different manner from those with local lines. The signals and algorithms used in determining line speeds and maintaining a stable link are described in Section 7.4.1.

When RSX-20F receives a character from a terminal, it must determine a number of things about the character before deciding what to do with the character. For example, the character would probably have a special meaning if it came from the CTY. It could also be a special character used in the buffering of data by both the terminal and the computer system. The algorithm that RSX-20F uses to decide what to do with an input character is described in Section 7.4.2.

7.4.1 Modem Handling

This section describes the RSX-20F algorithms for dealing with terminals that are accessing the system by way of dial-up lines and modems. Section 7.4.1.1 discusses modem-handling concepts. Section 7.4.1.2 describes the line service that is provided when an event on a dial-up line requires some action by RSX-20F. The rest of the section describes timeout routines.

7.4.1.1 Modem Handling Concepts - Each computer system has its own method for handling modems. The modem's "strapping options" must be set up to deal with the computer system's modem-handling techniques to establish a clean link. Attempts to hook up a nonstandard modem without taking into account the system's modem-handling techniques can cause significant problems for the computer system.

The DTR (Data Terminal Ready) signal is used by the host computer system to answer the phone ring. DTR allows the modem to answer the phone. At this point, if all is going well, the remote modem returns a carrier pulse. The local modem receives this pulse, and the modem control asserts to the computer system that carrier is on. Finally, the computer system raises RTS (Request to Send), which allows the local modem to give data to the system.

A "carrier transition" is a change in the state of the carrier signal. This change in state may be in either direction, from on to off or from off to on. The term "transition on" refers to a change from carrier off to carrier on, and the term "transition off" signifies a change in the other direction.

7.4.1.2 Terminal Driver Routine - This section describes the sequence of events when RSX-20F receives an interrupt requesting some type of modem handling on a certain line. The code that provides this service is the terminal driver routine, called $DMINT.

When a remote user dials up the local computer system, the modem raises the Ring Indicate signal (RI) which causes the DM11 to generate an interrupt requesting line service. When $DMINT, the terminal service routine, receives the interrupt, it must find out what type of service is being requested, because the interrupt is not specific about this. Therefore, when the interrupt is received, $DMINT checks to see if the interrupt is a ring or a carrier transition.

If the interrupt is a ring, $DMINT tells the KL to detach any job currently associated with this line. This prevents a user from dialing into another user's job. $DMINT then raises DTR (Data Terminal Ready) and RTS (Request to Send), which causes the local modem to answer the ring. $DMINT proceeds to check for carrier transition.

If the interrupt is not a ring, $DMINT determines whether DTR is high (meaning that a ring has been seen). If DTR is low, the interrupt is dismissed without any further checks. If DTR is high, $DMINT checks for carrier transition.

$DMINT now checks for carrier transition. If carrier is low, $DMINT assumes that carrier has been lost and sets TT.CRW (Carrier Wait) and TT.RIP (Remote In Progress). The loss of carrier may mean that the user has hung up the phone, or it may mean that atmospheric or other environmental conditions have caused a temporary signal loss. To determine if the user has hung up, $DMINT sets the Carrier Wait flag, which indicates that the line is waiting for a clean carrier signal to be re-established. Then the routine dismisses the interrupt. The modem timeout code, upon seeing the Carrier Wait flag, tests the state of carrier, and if the signal has returned, the timeout code returns the line to normal operation. (Refer to Section 7.4.1.3 for further information on the modem timeout code.)

If carrier is high, $DMINT assumes carrier has just come up and clears TT.CRW and TT.RIP. Then the routine checks to see if the line in question was previously connected to the computer system. If so, the interrupt is dismissed. These actions prevent a noisy carrier signal (one that comes and goes frequently) from detaching the user's job.

If $DMINT finds that the line for which it received an determines if the line needs to be checked for the correct baud rate (a software flag is set if the line is waiting to be checked). If the line needs to be auto-bauded, $DMINT sets the Auto-baud Wait flag (which is noticed by $DHINP, the character input routine - refer to Section 7.4.2 for an explanation of $DHINP). The routine then dismisses the interrupt.

If the line does not need to be auto-bauded, $DMINT assumes that the carrier transition to the ON state is the result of the local modem receiving the carrier signal for the first time. This assumption is warranted because the line has already been checked to see if it is connected to the system before this point in the algorithm is reached. At this point, $DMINT notifies the KL that a new line has been connected to the system. The KL sends the system banner to that line, and the connection is complete. The modem control routine then dismisses the interrupt.

7.4.1.3 Modem Timeout Routine - This section describes the modem timeout routine, which is labeled .DMTMO. Every 22 seconds, the terminal task routine calls .DMTMO. .DMTMO is also called at startup time and after a power-fail.

At startup time .DMTMO alters the nonexistent-device trap to handle a missing DM11. Since for each DM11 device on the system there should be a corresponding DH11, .DMTMO first checks the DH11 associated with a given DM11 to see if it exists. If the DH11 does not exist, .DMTMO marks the DM11 as nonexistent and proceeds to the next DH11/DM11 pair. If the DH11 does exist, .DMTMO tries to initialize the associated DM11. If the DM11 does not exist, it causes a nonexistent-device trap, and .DMTMO marks the DM11 as nonexistent and then proceeds to the next pair. If both the DH11 and the DM11 exist, .DMTMO initializes the device. This process is repeated until all 16 possible pairs have been checked and initialized or marked as nonexistent.

When .DMTMO is called during normal operation, it already knows which DH11s and DM11s exist. It checks the first DH11 and if it is nonexistent it checks the next DH11. When the routine finds a DH11 that exists, it checks the associated DM11. If the DH11 does not exist, the routine proceeds to the next pair. If both the DH11 and the DM11 exist, .DMTMO checks to see if any lines are in Carrier Wait. A line may be in the Carrier Wait state as the result of either a ring or a carrier loss. There are two bits associated with Carrier Wait, TT.CRW (Carrier Wait) and TT.RIP (Remote in Progress). Both these bits are set when a line enters Carrier Wait. If these bits are set as the result of a ring, they are both cleared when a carrier signal is asserted by the calling modem. If they are the result of a lost carrier signal, then .DMTMO determines if the carrier wait is the result of a noisy carrier signal or if the line has been hung up. The first time that .DMTMO finds a line in carrier wait, that is with both

the TT.CRW and TT.RIP bits set, it clears the TT.RIP bit and continues examining the other lines. The next time .DMTMO is called, it looks at the bits. If the TT.CRW is set and the TT.RIP bit is cleared, then the line has not reasserted carrier for at least twenty-two seconds. The connection is assumed to be lost and .DMTMO hangs up the line and clears the TT.CRW bit.

.DMTMO then examines the Interrupt Enable bit for each DM11. If .DMTMO does not find Interrupt Enable bit set for a given DM11, it logs this as an error to SPEAR and sets the Enable Interrupt bit. Then, on a line-by-line basis, it looks at the Line Enable Interrupt bit. If this bit is not set for a remote (dial-up) line, an error entry is sent to ERROR.SYS, the line is reset, and the Interrupt Enable bit is set to maintain the existing connection.

.DMTMO examines the Request To Send bit for each line. If the bit is set, .DMTMO sets the Set Data Terminal Ready bit.

7.4.1.4 The CTY and DL-11E Timeout Routine - The CTY and DL-11E timeout routine, .DLTMO, is called every 22 seconds. It is also called at startup and after a power-fail.

If .DLTMO is called at startup, it always assumes that the CTY exists. If the CTY does not exist the system crashes. Also, at startup .DLMTO alters the nonexistent-device trap to handle missing DL-11E's. .DLTMO checks each possible DL-11E to see if it exists. If it encounters a nonexistent DL-11E, it marks it as nonexistent and checks the next DL-11E. If the DL-11E exists, .DMTMO enables the line and initializes the software state of the line. This is done for all four possible DL-11E interfaces.

When .DLTMO is called during normal operation, it already knows which lines exist. It checks existing DL-11E's to see if the Hardware Carrier and Data Terminal Ready bits are set, and if the Interrupt Enable bits are set. If any one of the Interrupt Enable bits is not set, this condition is logged to ERROR.SYS and the bit is set. If both the Hardware Carrier and Data Terminal Ready bits are set, the line is actively connected. .DLTMO then asserts Data Terminal Ready and proceeds to the next DL-11E interface. If neither of these bits are set the line is disconnected.

.DLTMO checks the Carrier Wait bit (TT.CRW) and the Remote In Progress bit (TT.RIP). If the line has lost the carrier signal as a result of either a noisy line or the line being hung up, then both these bits are set. To determine which condition caused the loss of the carrier signal .DLTMO first clears the TT.RIP bit and then proceeds to the next line. The next time .DLTMO is called, if it still finds the TT.CRW bit set, it hangs up the line.

7.4.2 Terminal Handling

This section describes the $DHINP routine and the $DHOUT routine, which deal with input from and output to terminals, respectively (whether the terminals are remote or local). $DHINP is comprised of two routines: the character input routine and the terminal timeout routine. These are presented separately, while $DHOUT is described as a unit.

7.4.2.1 Character Input Routine - When a DH11 communications interface causes an interrupt, the first action the $DHINP routine performs is to check the line in question to see if it is an auto-baud line in auto-baud wait. If the auto-baud wait flag (TT.ABW) is on, $DHINP attempts to set the line speed. The routine then checks the low-speed auto-baud flag (TT.LSP) to determine whether it is attempting to do low-speed (110, 150, or 300) or high-speed (1200, 1800, 2400, 4800, or 9600) auto-baud. If it is doing high-speed auto-baud, the character received was detected at 2400 baud and is checked to see if it is a NUL. If the character is a NUL, the low-speed auto-baud and the ignore-next-character (TT.IGN) flags are set, and the line speed is set to 300 baud. If the character is not a NUL, the character is then pattern-matched against a table (HSPTAB) of bit patterns for both CTRL/C and carriage return at all applicable baud rates (1200, 1800, 2400, 4800, 9600). If a match is found, the table index is used to index another table (HSPSPD) of speed indices. If the sign bit (LSP.IG) is set, the ignore-next-character flag is set. The low-speed auto-baud code uses different tables (LSPTAB, LSPSPD) that check for 110, 150, and 300 baud, but it uses common code for checking for pattern matching. If no match is found in low speed auto-baud, the line is switched into high speed auto-baud detect (2400 baud), and the low speed auto-baud flag is cleared.

In either low or high speed auto-baud, if no match is found, the character is flushed. If a match is found, the auto-baud wait flag is cleared, the speed index is used to index a table of DH11 hardware speeds (SPDPAR) and the speed of the line is then set to the new speed. $DHINP then sends a message to the KL informing it of the connection and the new speed. $DHINP also places a <CR> in the line's input buffer if the line was not previously connected; this causes the system banner to be generated.

Note that the ignore-next-character flag (TT.IGN) serves two purposes: it is used to discard the second NUL that is received when the switch is made from high to low speed auto-baud, and it is used to discard junk characters that are sometimes generated when detecting the lower speeds in each table based on LSP.IG.

If the line in question is an auto-baud line not in auto-baud wait, and the character received has a framing error, a check is made for a BREAK (NUL accompanied by a framing error). The flag that indicates the last character was a BREAK (TT.BRK) is checked to see if two consecutive breaks have been seen. If two consecutive breaks have been seen, the line is put into high speed auto-baud wait and the TT.BRK is cleared. If this is the first BREAK, TT.BRK is set. If the character is not a BREAK, TT.BRK is cleared, and the character is passed to TTSTCH.

If the line is not an auto-baud line, $DHINP checks the character to see if there is a framing error. If the character has a framing error, there may be noise on the line or the line may be set to the wrong speed. In either case, the front end cannot determine what character -- if any -- was sent. $DHINP keeps track of how many framing errors it has received, and upon getting four consecutive framing errors, it sets the line's input speed to zero, which causes the DH11 to ignore input interrupts on that line. $DHINP creates a SPEAR entry that names the DH11 and the shut-down line within that DH11. If $DHINP gets a character cleanly (without a framing error) on a non auto-baud line, it first clears the count of framing errors, then passes the character to TTSTCH.

TTSTCH determines whether the character came from the CTY and, if so, whether it was a control backslash (CTRL/\). If the character was a CTRL/\, PARSER is requested. If the character from the CTY is not

meant for the PDP-11, the KL must be the intended recipient. In this case, TTSTCH checks the state of the protocol between the processors to determine the next action to be taken. If secondary protocol is running, the character is sent to the KL using secondary protocol.

For any line, if primary protocol is running, a check is then made for protocol pause. If the protocol is in the "pause" state, an additional check is made to see how much buffer space is left. If there is not a reasonable margin of buffers available, the character is discarded and a bell is sent to the user, otherwise a check is made to see if the line is enabled for XOFF/XON processing. If XON/XOFF is enabled and the character is an XOFF, the routine calls the XOFF processing code that will stop the output to the line quickly enough so that only a few more characters will get output to the line. If the character is an XON, and the front-end is enabled to handle XONs, and the line has been stopped previously by an XOFF, then output is restarted on the line.

If primary protocol is currently in force, $DHINP checks the character to see if it is an XOFF character. XOFF is the only input character to which RSX-20F reacts. The XOFF character requests the front end to stop sending data temporarily because the requesting device's buffer is full. The timing of XOFF processing is therefore critical, because any data sent by the front end after an XOFF will be lost. Thus, $DHINP checks for an XOFF before dispatching the character to the KL. If the character is indeed an XOFF, $DHINP calls the output XOFF processing code, thereby ending $DHINP's responsibility for the character. If the character is an XON, the front-end is enabled to handle XONs, and the line has been stopped previously by an XOFF, then output is restarted on the line. If the character is not an XOFF or XON, $DHINP simply passes the character, whatever it is, to the KL.

Finally, $DHINP checks the number of bytes available in the free pool. If the number of bytes left is less than the INPUT-THRESHOLD-LOW value stored in .IBFLO and currently set at 1600 bytes, the line is sent an XOFF. If the line sends another character or on the next clock tick $DHINP sets the line speed to zero and echoes a bell to the line. A table of lines that have been shut down is stored in STSW2. If the line was shut down by RSX-20F bit 11 of this word is set.

On each clock tick, RSX-20F checks to see if the number of bytes in the free pool is greater than or equal to the value of .IBFOK, currently set at 2000. If the number of bytes in the free pool is greater than or equal to this threshold, RSX-20F scans the table to see if there are any lines that it has shut down. If it finds any such lines it resets the line speeds of these lines, one line per clock tick.

7.4.2.2 Terminal Timeout Routine - This section describes the terminal timeout routine, which is labeled .DHTMO.

NOTE

The .DHTMO routine deals only with user terminals. Therefore, this section does not describe how the CTY is handled.

Every 10 seconds, the terminal task routine calls the DH11 timeout routine, .DHTMO. .DHTMO is also called at startup, and on a power-fail restart. When .DHTMO is called at startup, it does not know how many communications interfaces exist on the system, therefore .DHTMO assumes that all sixteen possible interfaces exist. At startup

it also alters the nonexistent-device trap vector,for initialization only. When .DHTMO finds an interface that does not exist, it marks the device as nonexistent and sets the external page pointer for that interface to 0. It then proceeds to the next interface. This is repeated for all sixteen possible interfaces. If, on the other hand, .DHTMO is called at some time other than startup, its internal table will have recorded the actual interfaces in use by the system. .DHTMO therefore does not attempt to do anything with a nonexistent interface.

Once .DHTMO knows whether it is startup, and has taken the appropriate action, the routine clears the reset flag. This flag allows .DHTMO to keep track of the hardware state of the communications interface. .DHTMO then gets the UNIBUS address of the DH11 communications interface and checks all the possible lines from bottom to top (0-17).

At this point in the startup execution of .DHTMO, it may attempt to reference a nonexistent DH11. If this happens it marks all sixteen lines in that controller as nonexistent and proceeds to the next DH11. If it is not startup, the number of DH11s connected to the system is already known. When the routine has scanned all possible DH11s, it proceeds to the exit code. Control is then returned to the calling routine.

Assuming that .DHTMO has not finished processing all the system's lines, it proceeds next to check for nonexistent memory locations and lost interrupt enabled conditions. If either of these hardware errors has occurred, .DHTMO logs the error in ERROR.SYS and performs a master clear on the DH11. .DHTMO then sets the reset flag to show that this DH11 has been cleared. If the call to .DHTMO comes at startup, .DHTMO next sets up the software state of the line. If it is not startup time, the software state of the line has already been set and does not require further attention.

If it is startup, or if the DH11 has been reset, .DHTMO's next action is to set the line speed for all sixteen lines. .DHTMO may also find that it needs to restart a single line. This occurs when a nonauto-baud line has had four consecutive framing errors (Refer to Section 7.4.2.1 for a description of framing error processing). After four consecutive framing errors occur, the $DHINP routine sets the line speed to zero. .DHTMO finds this line, sees that the line is marked to be reenabled, and resets it to the correct speed. .DHTMO also times out each auto-baud line that is in low-speed auto-baud wait. Upon finding such a line, .DHTMO clears the low speed auto-baud flag (TT.LSP) and puts the line in high-speed auto-baud detect. This prevents the line from becoming wedged in low-speed auto-baud wait.

This series of checks is run on each line. When .DHTMO completes the checks for one line, it checks to see if it is finished with the DH11 to which the line is connected. If not, .DHTMO increments its line-number counter and proceeds to check the next line on the DH11. If .DHTMO is finished with the DH11, it checks to see if the DH11 was reset (by checking the reset flag). If the DH11 was reset, .DHTMO hardware enables the DH11 and increments the DH11 counter. If it was not reset, .DHTMO simply steps to the next controller. In either case, .DHTMO returns to clear the reset flag, and proceeds to the next controller until all possible DH11s have been checked.

7.4.2.3 Character Output Routine - RSX-20F allows output to the terminals to take place independent of any intervention by the front end. The DH11s are capable of taking the address of the data to be sent and a line number to which to send it, and putting the data out to that line, without any prodding by RSX-20F. When output to a line is finished, an interrupt is generated. Since the interrupt may be generated by a variety of conditions other than a DH11 finishing output, RSX-20F must decide which type of interrupt occurred and determine what to do about it.

To determine the condition that generated the interrupt, RSX-20F examines each line connected to the DH11, starting with line 0. Having chosen the line to be examined, the $DHOUT routine checks to see if an output interrupt was expected from this line. If no interrupt was expected, the line was not considered to be active when the interrupt was generated. Receiving an interrupt on an inactive line is not necessarily an error condition, but $DHOUT does not attempt to deal with the interrupt. If all lines on the DH11 have been checked, the interrupt is dismissed. If there are still lines to be checked on this DH11, $DHOUT returns to check them.

If an output interrupt was expected from the line being examined, $DHOUT checks to see if the current line generated the interrupt. If it did, the line has just completed its output. If not, $DHOUT returns to check for other lines in this DH11, with the same results as in the preceding paragraph.

If the current line has completed output, $DHOUT first sees if the terminal concerned is the CTY. If it is, $DHOUT must decide whether the output was PDP-11 I/O or KL I/O. PDP-11 I/O is handled by RSX-20F's I/O routines, which $DHOUT calls. KL I/O, on the other hand, must be handled by $DHOUT.

Assuming that the output is from the KL, the STTYDN routine checks to see if there is a Send-All in progress. (The term Send-All refers to data that is sent to all active lines that have not refused it.) If a Send-All is in progress, the interrupt was sent to notify $DHOUT that this line finished the Send-All transmission. In this case, $DHOUT counts the Send-All done for this line by decrementing the Send-All line counter. The Send-All is done for the entire system, as far as the software is concerned, when the count goes to zero. If this counter ever goes negative, RSX-20F crashes, because it has received an interrupt when it has no reason to expect one. This can be due to either a hardware or a software error.

If no Send-All is in progress, the interrupt was sent to notify $DHOUT that the line in question has completed the transmission of a normal output packet. Therefore, $DHOUT checks to see if the output queue for the line is empty. If so, all the data for the line has been sent. If not, $DHOUT computes the remaining bytes in the current packet. Should it find that the packet has been transmitted completely, $DHOUT deallocates the packet's node (so as to keep as much free space available as possible).

$DHOUT next checks to see if there is a Send-All pending for this line. The routine that does this check is called STNXT. (Note that it is possible for a Send-All to have just finished and produced an interrupt, and to have another Send-All waiting to be transmitted.) If a Send-All is waiting, STNXT asks if the line is suppressing Send-Alls. If not, STNXT starts the Send-All transmission, and calls .DHSTO (for a description of .DHSTO see below).

If the line is suppressing Send-Alls, the next check STNXT performs is to see if the output queue for this line is empty. If the queue is empty, STNXT calls the routine .TTACK to determine if an acknowledge signal (ACK) can be sent (refer to section 7.4.2.4). $DHOUT then dismisses the interrupt and returns to choose a new line to check. If the queue is not empty, $DHOUT checks if the line is XOFF'd. The terminal sends an XOFF signal if its input buffer is full and it has no more space to store characters. If the line is XOFF'd, $DHOUT proceeds to choose another line to check. If it is not XOFF'd, control is passed to .DHSTO.

.DHSTO is a simple routine that starts the transmission of the next packet of data, and flags the line as one that will be generating an output interrupt sometime in the future. This is the flag that is checked immediately after choosing a new line. Thus, when this packet has finished transmitting, the interrupt generated can be recognized because this line expects an interrupt. At this point, .DHSTO relinquishes control to $DHOUT so that $DHOUT can choose another line to be checked.

CHAPTER 8

DTE20 OPERATION

The DTE20 (DTE stands for Data Ten-to-Eleven) is the hardware interface between the KL and the PDP-11 processors. The DTE20 is used for many different things in the operation of the computer system, since it is the only (nondiagnostic) method of communication between the KL and the front end. All the uses of the DTE20 are extensions of its four basic hardware operations:

- Deposit/Examine
- TO-11 Transfer/TO-10 Transfer
- Doorbell Function
- Diagnostic Operations

8.1 DTE20 COMMUNICATIONS REGION

The Communications Region is an area in KL memory that is readable by all processors. It is composed of sequential areas, one for each processor connected to the network. Both KL's and PDP-11's are represented, and each processor owns one area. This owned area is the only part of the Communications Region into which the processor can write. There is an exception to this rule, however, to cover the situation of Communications Region initialization. In this case, the KL processor that initializes the Communications Region is allowed access to the entire region. When the region has been initialized, the rules for access to areas hold until the region is initialized again.

There is a negative extension to the Communications Region called the header. This header allows each PDP-11 processor represented in the Communications Region to determine its protocol processor number, because each PDP-11 has a space which it can examine in the header. By examining the first word of its relocated examine space, the PDP-11 can determine its protocol processor number and thereby locate its area in the Communications Region. The PDP-11 also knows that the first word of the Communications Region itself is at location N+1, where N is the PDP-11's protocol processor number. Thus, when the PDP-11 wishes to communicate with another processor, the PDP-11 can scan the areas in the Communications Region and find the areas owned by any other processors.

Each processor has its own area in the Communications Region that is made up of a number of sections. A processor has one section in its area for itself, and one for each processor with which it will communicate. Thus, each pair of processors that communicate with each other uses four sections of the Communications Region, two in each of two different areas.

Figure 8-1 illustrates the Communications Region.

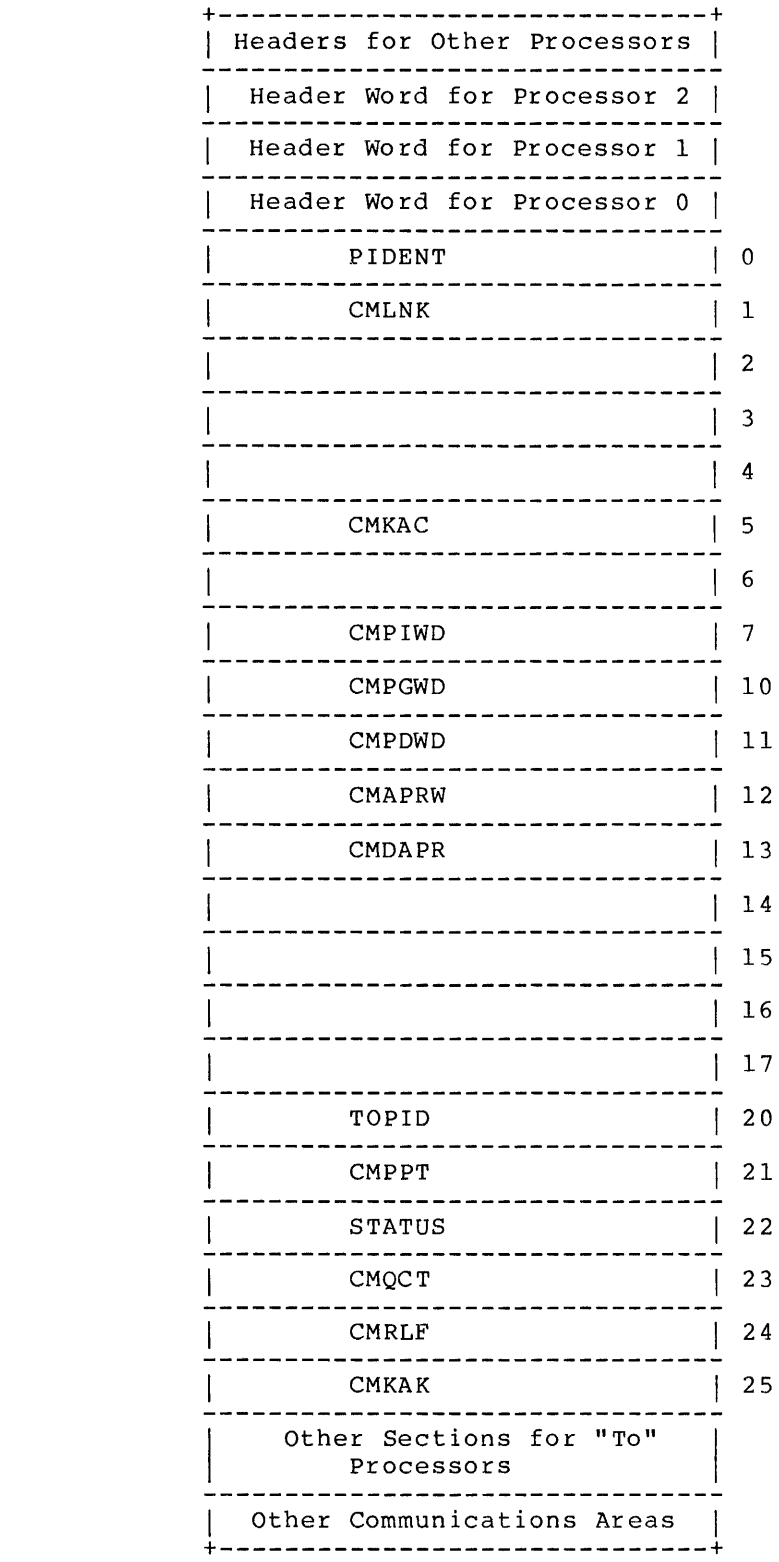
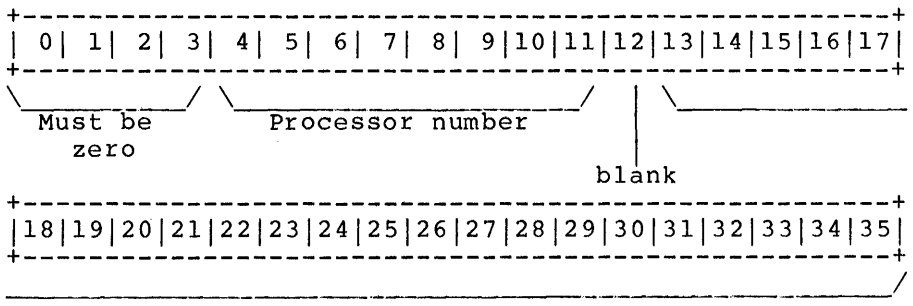


Figure 8-1: KL Communications Region

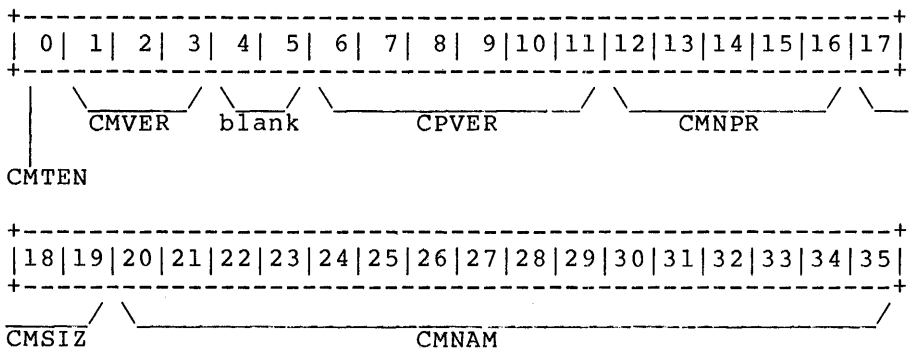
The information contained in each word of the Communications Region is described below.

Processor Header Word:



This word is part of the negative extension to the Communications Region. It specifies the location of the PDP-11's owned area by means of an offset from word 0 of the Communications Region.

PIDENT Word:



The PIDENT word provides information about the owning processor and its area. The separate fields in PIDENT are described below.

CMTEN This bit is one if this area belongs to a KL; otherwise, it is zero.

CMVER This area contains the communications area version number.

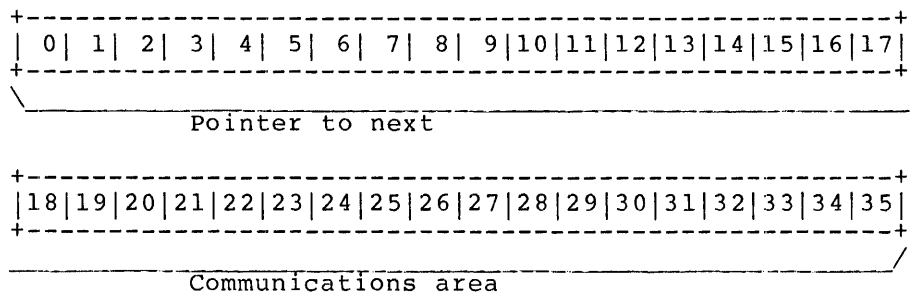
CPVER This area contains the protocol version number.

CMNPR This area contains the number of processors represented in this area, including the owning processor.

CMSIZ This area contains the size of the entire owning processor's area in eight-word blocks.

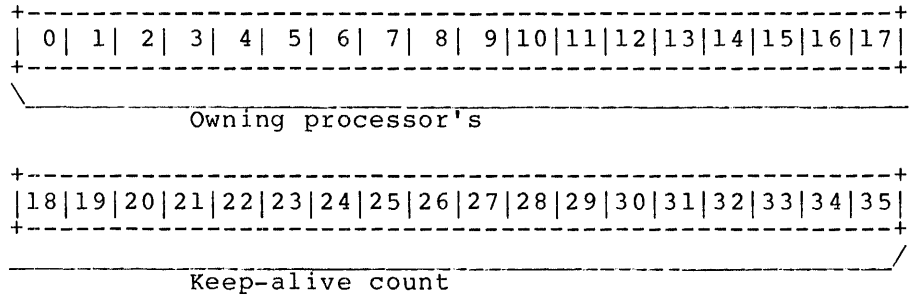
CMNAM This area contains the name (serial number) of the processor that owns this area.

CMLNK Word:



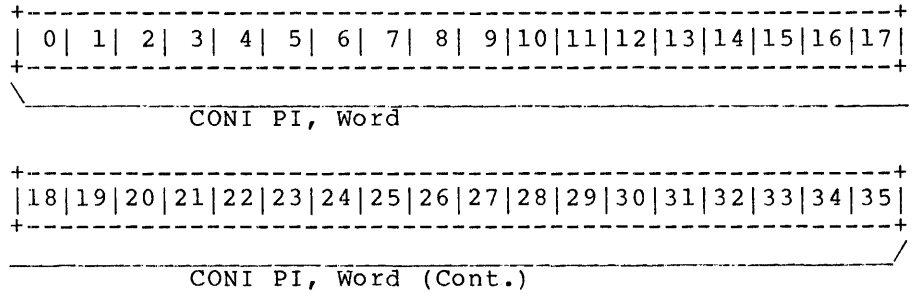
The CMLNK word contains a pointer to the next communications area, relative to word 0 of the entire Communications Region. All the CMLNK words in the entire Communications Region form a circular list.

CMKAC Word:



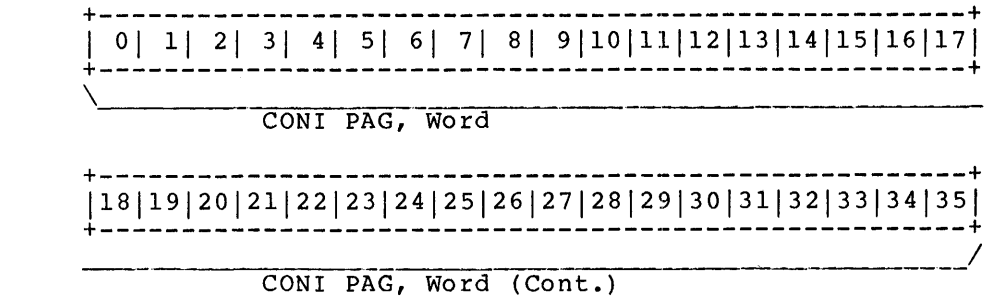
The CMKAC word contains the owning processor's Keep-Alive count. This word is incremented periodically, and is also checked periodically to make sure that it has changed. The Keep-Alive count should be incremented at least once a second by the owning processor, and the monitoring processor should allow the count to remain unchanged for at least six seconds before declaring the owning processor to be dead.

CMPIWD Word:



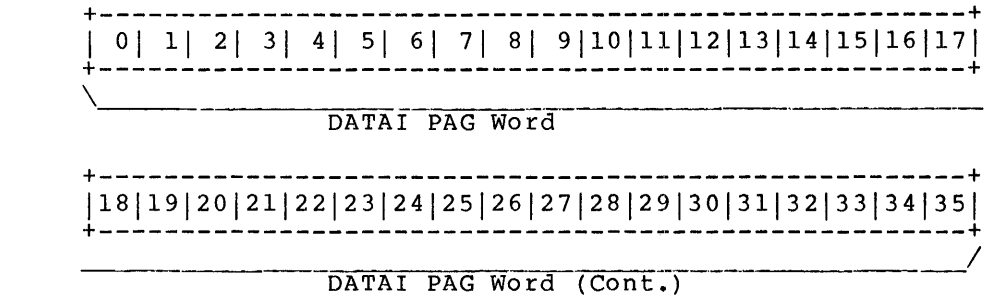
The CMPIWD word is the storage area provided for reading the Priority Interrupt system conditions. The CONI PI, instruction will place the PI status information here. (For more information on the Priority Interrupt system refer to the DECsystem-10/DECSYSTEM-20 Processor Reference Manual.)

CMPGWD Word:



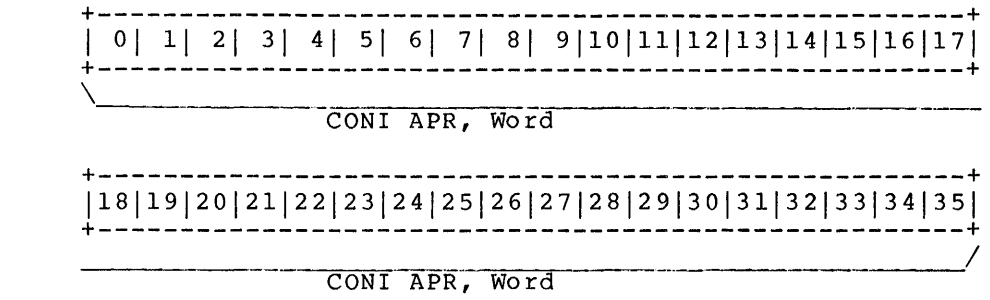
The CMPGWD word is the storage area provided for reading the system status of the Pager. The CONI PAG, instruction puts the status information here. (For more information on the Pager refer to the DECsystem-10/DECSYSTEM-20 Processor Reference Manual.)

CMPDWD Word:



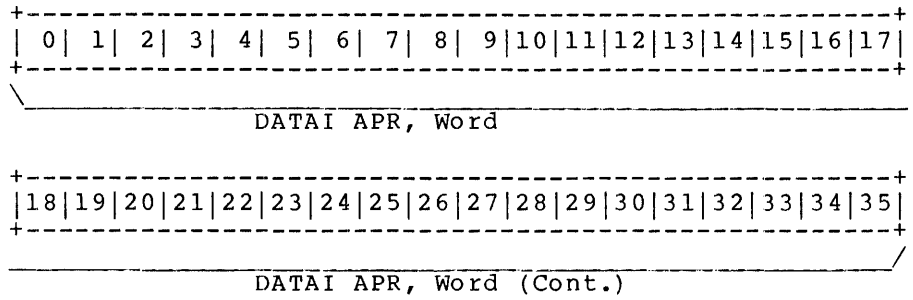
The CMPDWD word is the storage area provided for reading the process status of the Pager. The DATAI PAG, instruction puts the status information here. (For more information on the Pager refer to the DECsystem-10/DECSYSTEM-20 Processor Reference Manual.)

CMAPRW Word:



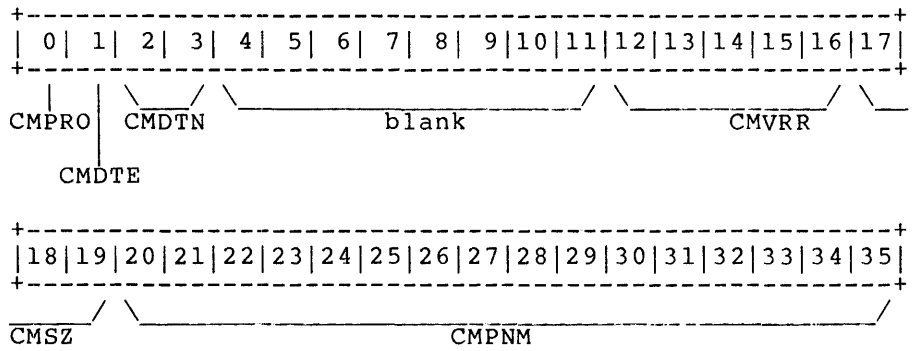
The CMAPRW word is the storage area provided for reading the Arithmetic Processor flags. The CONI APR, instruction puts the status of the error and sweep flags here. (For more information on the Arithmetic Processor, refer to the DECsystem-10/DECSYSTEM-20 Processor Reference Manual.)

CMDAPR Word:



The CMDAPR word is the storage area provided for reading the current break conditions from the Arithmetic Processor. The DATAI APR, instruction puts the information here. (For more information on break conditions and the Arithmetic Processor, refer to the DECsystem-10/DECSYSTEM-20 Processor Reference Manual.)

TOPID Word:



The TOPID word is the first location in each "To" processor area. This word contains information on the connections between the owning processor and the "To" processor.

CMPRO This bit is one if this block is used to communicate with a KL.

CMDTE This bit is one if a DTE20 connection exists between this "To" processor and the owning processor.

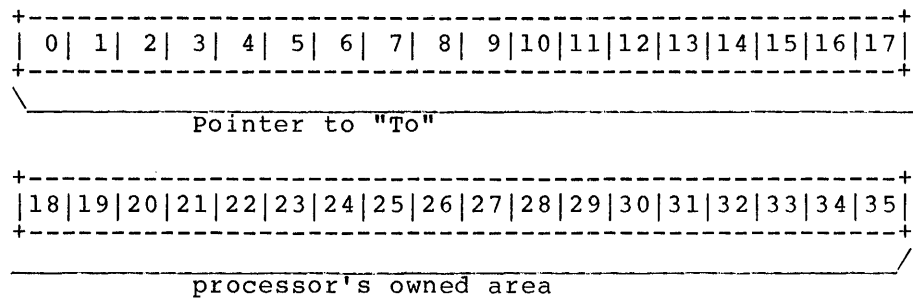
CMDTN If CMDTE is one, this area contains the number of the DTE20 that connects the two processors.

CMVRR This area designates the version of protocol in use between the two processors.

CMSZ This area contains the size of this block in eight-word blocks.

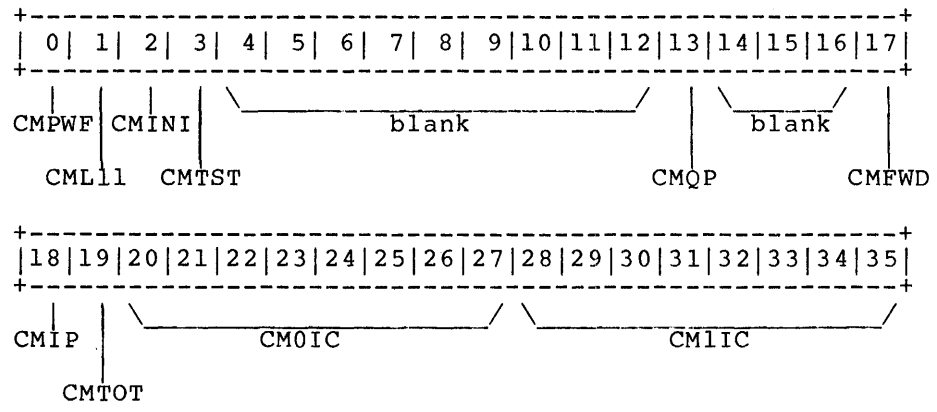
CMPNM This area contains the processor number of the "To" processor.

CMPPT Word:



The CMPPT word contains a pointer to the area in the Communications Region that is owned by the "To" processor.

STATUS Word:



The STATUS word is intended to be the only word a processor has to examine when it receives a doorbell interrupt. The STATUS word contains the following status bits and counters:

CMPWF This bit is the power fail indicator. The PDP-11 sets this bit to one (in its own area) to notify the KL that the PDP-11 has lost power.

CML11 This bit is one when the owning processor knows that it has crashed and wishes to be reloaded. The reload happens when the owning processor sets CML11 (in the "To" processor's section of the owned processor's area) and rings the "To" processor's doorbell. The "To" processor then examines the STATUS word, sees CML11 set, and performs a reload operation for the owning processor.

CMINI This bit is the initialization bit for MCB protocol only. MCB protocol applies to front ends other than the console front end; it is therefore not discussed further in this manual.

CMTST This bit provides the PDP-11 with the ability to determine if the Deposit/Examine operation that just finished was a valid operation. This ability is useful when the examine protection word of the DTE20 is zero, and PI 0 has been enabled, since in this situation any Examine done by the PDP-11 appears to succeed and returns a value of zero. The CMTST bit provides a check on the operation because it is always guaranteed to be one. If the PDP-11 finds this bit to be zero after an Examine or a Deposit, the PDP-11 leaves primary protocol.

CMQP This bit is one if queued protocol is in use. This bit is originally set in all areas by the KL that initializes the Communications Region.

CMFWD This bit is a flag, set by the sending processor to indicate that the transfer is to be done in full word mode.

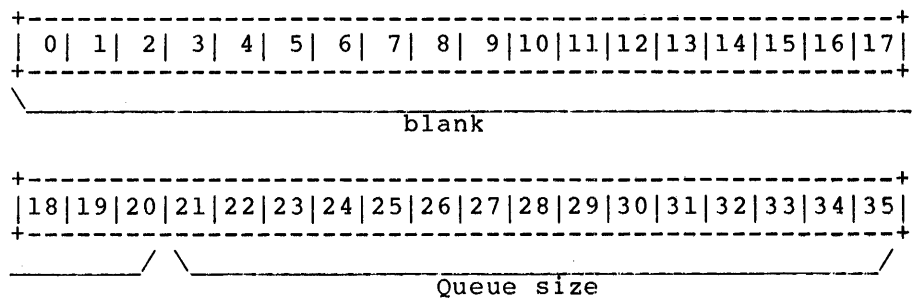
CMIP This bit is set if an indirect packet is being transferred. The bit is set in the sending processor's section of the receiving processor's area. If this bit is set by the sending processor, it should not increment the queue count. If the bit is set, the receiving processor reads it and realizes that the doorbell interrupt it received signals the beginning of the transfer of the indirect portion of an indirect message transfer.

CMTOT This bit is set to one by the receiving processor in the sending processor's section of the receiving processor's area. The bit is set when the sending processor sets the CMIP bit or when the sender increments the queue count. The receiving processor clears this bit when it gets a To-receiver Done interrupt. The purpose of this procedure is to assure the sending processor that the receiver has not lost a Done interrupt.

CM0IC This area contains a wrap-around counter that is incremented in the PDP-11's area each time a direct transfer request is initiated by the PDP-11. The KL keeps the last value of CM0IC in the TO-11 section of its own area. If the KL's saved value differs from the current copy in the PDP-11's area, the KL starts a TO-10 packet transfer. The difference between the KL's copy and the copy the PDP-11 increments should be either zero or one. If the difference is greater than one, the PDP-11 has tried to send a packet before the previous packet transfer was finished. This count is not incremented when the PDP-11 sends a TO-10 indirect packet; the CMIP bit is used to indicate doorbells for indirect packets. This counter is also useful in the situation where a doorbell has been missed by the KL. The next doorbell causes the KL to check this counter, during which operation it discovers the missed doorbell because of the difference in the counter's value.

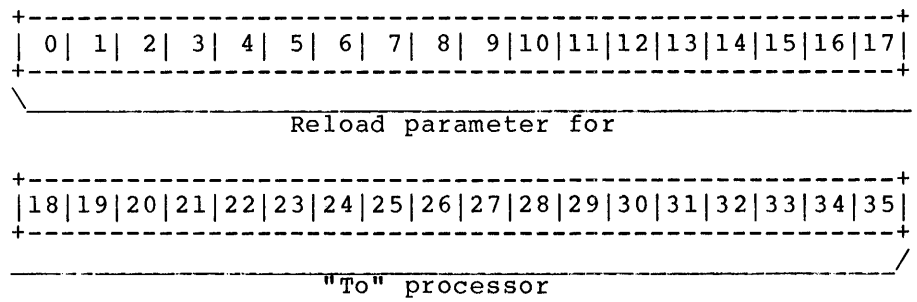
CM1IC This counter has the same function for the PDP-11 that the CM0IC area performs for the KL.

CMQCT Word:



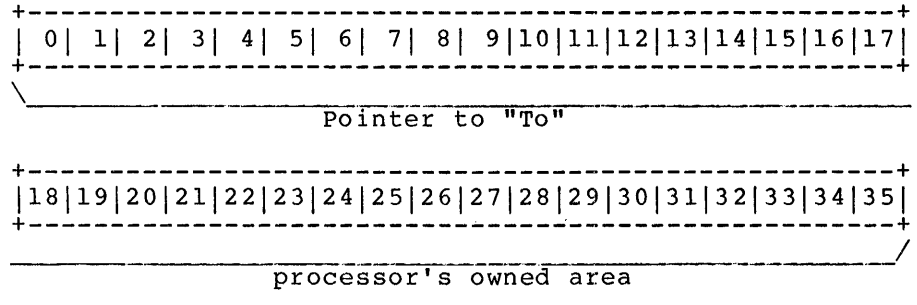
This word contains the number of eight-bit bytes written into the current packet by the transmitting processor. The packet can contain more than one message; this word does not contain a count of the number of messages, only of the number of bytes.

CMRLF Word:



This word contains a copy of the "To" processor's reload word. The copy is saved by the owning processor in case the "To" processor crashes.

CMKAK Word:



This word contains the owning processor's copy of the "To" processor's Keep-Alive count for purposes of comparison with the continuously updated copy kept by the "To" processor.

8.2 DTE REGISTERS

The DTE20 has sixteen registers, which are used in various types of transfer operations. Figure 8-2 shows the registers along with their addresses, and offers an explanation of the function of each register. The addresses shown are for DTE0; DTE1, DTE2 and so on, use the succeeding locations. The precise location can be figured by adding 40*N to the location of the register that you wish to access, where N stands for the DTE20 number.

DLYCNT	774400
DEXWD3	774402
DEXWD2	774404
DEXWD1	774406
TENAD1	774410
TENAD2	774412
TO10BC	774414*
TO11BC	774416
TO10AD	774420
TO11AD	774422
TO10DT	774424
TO11DT	774426
DIAG1	774430
DIAG2	774432
STATUS	774434*
DIAG3	774436

Figure 8-2: DTE20 Registers

The two registers that have asterisks beside their addresses, TO10BC and STATUS, are the only registers that are available to both processors.

8.2.1 DTE20 Status Word

The most important of these registers is STATUS, the Status Word. This register is the only one read when the KL receives a doorbell interrupt, so it must store a good deal of information. The Status Word has two different states: one interpretation is valid if the PDP-11 is writing into the Status Word, and another is valid if the PDP-11 is reading the register. If you are examining the Status Word

after a crash in an attempt to tell why the crash occurred, you should assume that the PDP-11 was reading the Status Word. This is logical because the write state lasts only as long as the hardware takes to do the physical write, which, of course, is a very short time. Thus the chances are that the Status Word was in a read state.

Figure 8-3 illustrates the form of the Status Word in the read state.

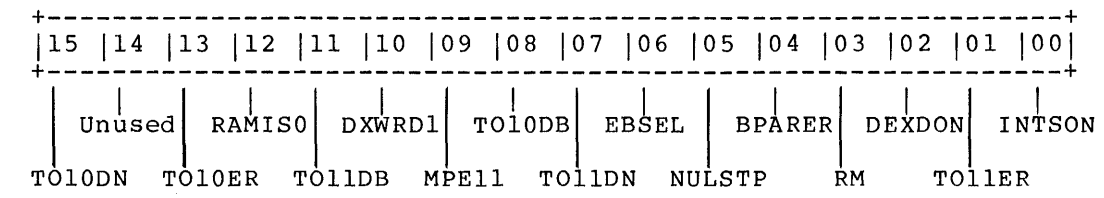


Figure 8-3: DTE20 Status Word in Read State

The bits in the read version of the Status Word have the following names and functions:

Bit	Symbol	Function
15	TO10DN	TO-10 NORMAL TERMINATION If this bit is set, the TO-10 byte count went to zero or the PDP-11 program set the DON10S bit in the write version of the status word. TO10DN is not set if an error termination occurred. See TO10ER if you believe an error has occurred.
14		UNUSED
13	TO10ER	TO-10 ERROR TERMINATION If this bit is set, one of a number of errors has occurred. To determine which one, you must check a number of different bits in different words. If there was an NPR UNIBUS parity error, the NUPE bit of the DIAG3 word is set. A PDP-11 memory memory parity error is indicated by the MPE11 bit of the Status Word being set. The PDP-11 program may have set the error status bit in the write version of the Status Word. Or there may have been a UNIBUS timeout, in which case no bit is set. If this bit (TO10ER) is set, TO10DN is not set.
12	RAMIS0	RAM IS ZEROS This bit is used in single-stepping the DTE20, and has no meaning in other uses. It is set if the data read from a RAM location is all zeros. This bit is provided for diagnostic purposes only.
11	TO11DB	KL REQUESTED PDP-11 INTERRUPT When this bit is set, the KL processor has requested a PDP-11 doorbell interrupt by means of a CONO DTEn, TO11DB instruction.
10	DXWRD1	DEXWORD 1 This bit is provided on a read for diagnostic purposes only, and has no meaning except when the DTE20 is being single-stepped.

Bit	Symbol	Function
09	MPE11	PDP-11 MEMORY PARITY ERROR If this bit is set, the PDP-11 memory had a parity error during a data fetch for a TO-10 byte transfer. Parity errors can be detected only if the PDP-11 has one of the MF11UP or MF11LP memory parity options.
08	TO10DB	PDP-11 REQUESTED KL INTERRUPT When this bit is set, the PDP-11 has requested a KL doorbell interrupt by writing the INT10S bit of the write version of the Status Word, and the KL has not yet cleared the bit.
07	TO11DN	TO-11 TRANSFER DONE This bit is set when the TO-11 byte count equals zero, when a transfer stops on a null character, or when a PDP-11 program has set the error status bit in the write version of the Status Word (DON11S).
06	EBSEL	BUFFER SELECT This bit is provided on a read for diagnostic purposes only, and has no meaning except when the DTE20 is being single-stepped.
05	NULSTP	NULL STOP If this bit is on, the TO-11 transfer stopped because the stop bit was set (the stop bit is the ZSTOP bit of the TO11BC register).
04	BPARER	EBUS PARITY ERROR This bit is set if the DTE20 detects an EBUS parity error during a TO-11 DTE20 byte transfer or examine transfer.
03	RM	RESTRICTED MODE If this bit is set, the attached PDP-11 is in restricted mode. Otherwise, the PDP-11 is in privileged mode. The value of this bit is determined by the setting of the privileged switch on the DTE20.
02	DEXDON	DEPOSIT/EXAMINE DONE This bit is set when the last deposit or examine operation is finished. No interrupt occurs when this operation finishes; the PDP-11 must watch for this bit to be set after every deposit or examine operation. The DTE20 clears this bit whenever a deposit or examine is started by writing into the TENAD2 register.
01	TO11ER	TO-11 BYTE ERROR TERMINATION If this bit is set, an error occurred during a TO-11 byte transfer, or the PDP-11 program set the error bit ERR11S in the write version of the Status Word. The TO11DN bit in the read version is not set if there is actually an error termination.

Bit	Symbol	Function
00	INTSON	INTERRUPTS ON If this bit is set, the DTE20 is enabled to generate PDP-11 BR requests. If the bit is off, the DTE20 does not have this capability. The bit can be set by writing a one into bit 5 (INTRON) of the read version of the Status Word, and cleared by writing a one into bit 3 (INTROF).

Figure 8-4 illustrates the form of the Status Word in the write state.

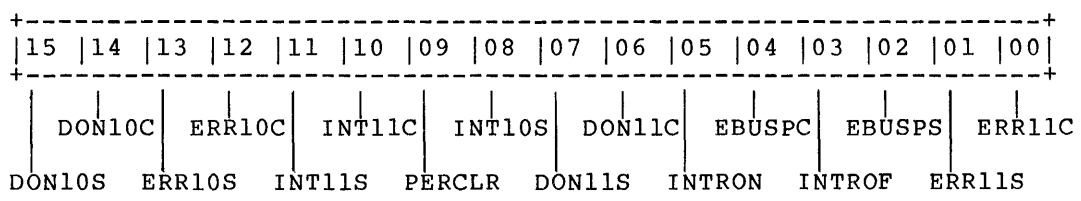


Figure 8-4: DTE20 Status Word in Write State

The bits in the write version of the Status Word have the following names and functions:

Bit	Symbol	Function
15	DON10S	SET NORMAL TERMINATION STATUS This bit on a write is provided for diagnostic purposes only. Writing a one to this bit sets the TO10DN bit in the read version of the Status Word. Writing a one does not terminate a transfer in progress.
14	DON10C	CLEAR NORMAL TERMINATION STATUS Writing a one to this bit clears the TO10DN bit in the read version of the Status Word.
13	ERR10S	SET ERROR TERMINATION STATUS Writing a one to this bit sets the TO10ER bit in the read version of the Status Word.
12	ERR10C	CLEAR ERROR TERMINATION STATUS Writing a one to this bit clears the TO10ER bit in the read version of the Status Word.
11	INT11S	SET PDP-11 INTERRUPT STATUS Writing a one to this bit sets the TO11DB bit in the read version of the Status Word, resulting in a doorbell interrupt to the PDP-11.
10	INT11C	CLEAR PDP-11 INTERRUPT STATUS Writing a one to this bit clears the TO11DB bit in the read version of the Status Word. This action enables the next doorbell interrupt to be generated.
09	PERCLR	CLEAR PARITY ERROR Writing a one to this bit clears the PDP-11 memory parity error flag, the MPE11 bit in the read version of the Status Word.

Bit	Symbol	Function
08	INT10S	SET KL INTERRUPT STATUS Writing a one to this bit sets the TO10DB bit and does a CONI [TO10DB]. This results in a vectored interrupt to location 104+B*N in the KL EPT.
07	DON11S	SET TO-11 TERMINATION STATUS Writing a one to this bit sets the TO-11 normal termination flag, which is the TO11DN bit in the read version of the Status Word. This bit on a write operation is provided for diagnostic purposes only, since writing a one here does not terminate a transfer already in progress.
06	DON11C	CLEAR TO-11 TERMINATION STATUS Writing a one to this bit clears the TO-11 normal termination status flag, TO11DN.
05	INTRON	INTERRUPTS ON Writing a one to this bit enables the DTE20 to generate PDP-11 BR requests. Writing into this bit, whether a zero or a one, does not clear any interrupts that are waiting. The DTE20 interrupt capability can be disabled by writing a one into bit 3, INTROF. The current setting of the interrupt capability can be checked by reading bit 0, INTSON.
04	EBUSPC	CLEAR EBUS PARITY ERROR Writing a one to this bit clears the EBUS parity error flag, BPARER, which is bit 4 in the read version of the Status Word.
03	INTROF	INTERRUPTS OFF Writing a one to this bit disables the DTE20 interrupt capability. Writing a one or a zero to this bit does not clear any interrupts that are waiting.
02	EBUSPS	SET EBUS PARITY ERROR Writing a one to this bit sets the EBUS parity error flag, BPARER, bit 4 in the read version of the Status Word.
01	ERR11S	SET TO-11 ERROR TERMINATION STATUS Writing a one to this bit sets the TO-11 error termination flag, which is bit 1, TO11ER, in the read version of the Status Word. This bit on a write is provided for diagnostic purposes only. Writing a one does not terminate a transfer in progress.
00	ERR11C	CLEAR TO-11 ERROR TERMINATION STATUS Writing a one to this bit clears the TO-11 error termination flag, TO11ER.

8.2.2 Diagnostic Words

The three diagnostic words are used to communicate by way of the diagnostic bus, which is electronically isolated from the other EBUS communications. This means of communication is not normally used except for diagnostic checks, or when other means have broken down. For example, the PARSER can on occasion use the diagnostic bus, and of course the diagnostic programs use it when necessary.

Diagnostic Word 1 has the following form when being written:

Bit	Symbol	Function
15-09	DS00-DS06	These bits specify the diagnostic selection code. (For the meanings of these bits see the read form of Diagnostic Word 1.) If DS00 and DS01 are both zero, write functions can be done while the system is running without being in diagnostic mode. Thus the PDP-11 can sample KL status without danger of corrupting data on the EBUS.
08	DEX	This bit must be zero.
07	DFUNC	Setting this bit to a one causes the KL processor to stop sending basic status information on the DS lines. This allows a loop-back test to be performed on the DS lines. If any of the DS lines are set (by the DTE20) the result is an "or" of the bits set in the DTE20 and the KL status.
06		This bit must be zero.
05	D1011	Setting this bit to a one sets the DTE20 to 10/11 diagnostic mode. This mode is used to diagnose the DTE20 itself.
04	PULSE	Writing a one to this bit generates a single clock cycle if 10/11 diagnostic mode is set (that is, the D1011 bit is on).
03	DIKL10	Writing a one to this bit puts the DTE20 into diagnostic data transfer mode if the DTE20 is privileged. Any subsequent examines and deposits become diagnostic functions instead of accessing KL memory. Writing a zero to this bit returns the DTE20 to normal data transfer mode. All subsequent examines and deposits refer to KL memory.
02	DSEND	Setting this bit to a one causes the data in a diagnostic bus transfer to be sent (TO-10). Setting the bit to a zero causes the data to be received (TO-11).
01		Unused
00	DCOMST	If this bit is set to a one while the DTE20 is switched to privileged, the effect is to set diagnostic command start. Setting the bit to a zero clears diagnostic command start.

Diagnostic Word 1 has the following form when being read:

Bit	Symbol	Function
15-12	DS00-DS03	Unused
11	DS04	If this bit is a one, the KL internal clock has frozen because of one of the following hardware malfunctions: CRAM, DRAM, fast memory parity error, or Field Service test condition.
10	DS05	If this bit is a one, the KL is running. The microcode checks this flag between PDP-10 instructions, and enters the halt loop if the flag is off. This flag is under control of the PDP-11 using two diagnostic functions. The KL cannot affect it.
09	DS06	This bit is set to one when the microcode enters the halt loop and is cleared when the microcode leaves the halt loop.
08	DEX	If this bit is set, the interface major state is deposit or examine.
07	TO10	If this bit is set, the interface major state is a TO-10 transfer.
06	TO11	If this bit is set, the interface major state is a TO-11 transfer.
05	D1011	If this bit is a one, the DTE20 is in 10/11 diagnostic mode, that is, it diagnoses itself.
04	VEC04	This bit is set to vector interrupt address bit 4.
03-01		Unused or zero.
00	DCOMST	If this bit is set, a diagnostic command is in progress.

Diagnostic Words 2 and 3 have very similar forms in the read and the write state; thus, the two states are illustrated together, rather than separately, as with Diagnostic Word 1.

Diagnostic Word 2 has the following form when being read or written:

Bit	Symbol	Function
15		RAM FILE MIXER (RFM) ADDRESS BIT 0
	RFMAD0	Read: This bit is set to the contents of RFM address bit 0.
	RFMAD0	Write: This bit must be zero.
14		RFM ADDRESS BIT 1
	RFMAD1	Read: This bit is set to the contents of RFM address bit 1.
	EDONES	Write: If a one is written to this bit, the EBUS done status is set. If a zero is written here, the EBUS done status is cleared.

Bit	Symbol	Function
13		RFM ADDRESS BIT 2
	RFMAD2	Read: This bit is set to the contents of RFM address bit 2.
	RFMAD2	Write: This bit must be zero.
12		RFM ADDRESS BIT 3
	RFMAD3	Read: This bit is set to the contents of RFM address bit 3.
	RFMAD3	Write: This bit must be zero.
11-07		Unused Read: These bits are always zeros. Write: These bits must be zeros.
06		DTE20 RESET
	DRESET	Read: This bit is zero.
	DRESET	Write: If a one is written to this bit, the DTE20 is reset.
05		Unused Read: This bit is zero. Write: This bit must be zero.
04-01		Read: These bits are zero. Write: Loads 04, 03, 02, 01 into minor state counter 8, 4, 12, 1 for diagnostic use only. During normal operation this bit must be zero.
00		Unused

Diagnostic Word 3 has the following form when being read or written:

Bit	Symbol	Function
15	SWSLLT	SWAP SELECT LEFT Read: CNTl[N] SWAP DEL LT Write: This bit must be zero.
14	DPS4[N]	PARITY (1) H Read: If this bit is set, the DPS4 [N] parity flop is on. This bit is for diagnostic use only. Write: This bit must be zero.
13-08		CAPTURED UNIBUS PARITY ERROR INFORMATION Read: When a UNIBUS parity error is detected, Ann means UNIBUS register address bit, and Dnn means UNIBUS data bit.

UNIBUS Data Bits

INITIAL	D15	D14	D13	D12	D11	A00
1st Shift	D10	D09	D08	D07	D06	A00
2nd Shift	D05	D04	D03	D02	D01	A00
3rd Shift	D00	A04	A03	A02	A01	A00
4th Shift	D15	D14	D13	D12	D11	A00

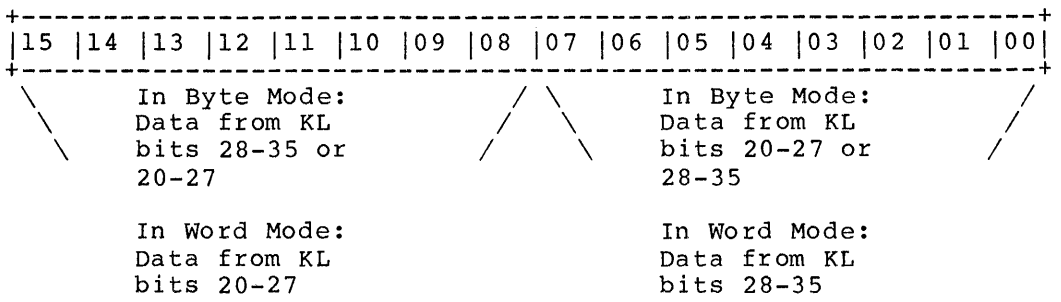
Write: This bit must be zero.

Bit	Symbol	Function
07-06		Unused Read: This bit is zero. Write: This bit must be zero.
05	SCD	SHIFT CAPTURED DATA Read: This bit is zero. Write: Writing a one to this bit shifts captured data so that the next read of DIAG3 changes bits 13-08.
04	DUPE CDD	DATO UNIBUS PARITY ERROR Read: If this bit is a one, a DATO UNIBUS parity error has been detected by the DTE20. Write: Writing a one to this bit clears the DUPE and DURE error flags.
03	WEP	WRITE EVEN (BAD) PARITY Read: This bit specifies the read status of the write even UNIBUS parity flip-flop. Write: Writing a one to this bit causes the DTE20 to generate even (bad) parity on all UNIBUS transfers that have parity. Writing a zero to this bit makes the DTE20 generate odd (good) parity on all subsequent transfers that have parity. This bit is provided for diagnostic purposes to check the parity network.
02	DURE	DATO UNIBUS RECEIVE ERROR Read: If this bit is set, a UNIBUS receiver error has occurred. Write: This bit must be zero.
01	NUPE CNUPE	NPR UNIBUS PARITY ERROR Read: If this bit is a one, a UNIBUS parity error has occurred on an NPR (byte) transfer. Write: Writing a one to this bit clears the NUPE flag.
00	TO10BM	TO-10 BYTE TRANSFER MODE Read: This bit is zero. Write: Writing a one to this bit causes TO-10 byte transfers to be done in byte mode from PDP-11 memory. Writing a zero to the bit causes the transfers to be in word mode.

8.2.3 DTE20 Data Transfer Registers

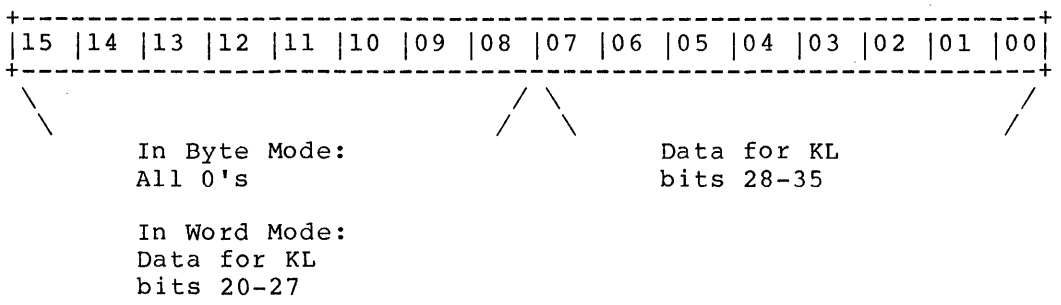
The remaining twelve DTE20 registers are used in data transfer operations. This section briefly describes the function of each of these registers, and illustrates their format.

TO11DT



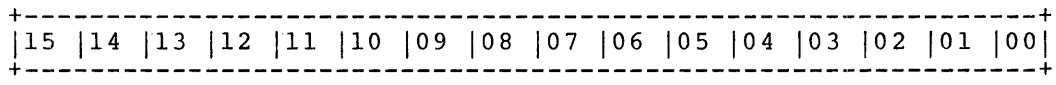
TO11DT This register contains the last byte or word transferred across the DTE20 to the PDP-11. Since it is not clear from the data in this register which bits of the KL word are represented, the following method is used to resolve the ambiguity. If the address in TO11AD is even, the left byte of TO11DT will contain bits 28-35; if TO11AD is odd, the left byte of TO11DT will contain bits 20-27. The right byte will contain the complementary set of bits. This register makes it possible to identify the last data that successfully transferred across the DTE20.

TO10DT



TO10DT This register contains the remainder of the data sent to the KL during a TOKL10 byte transfer. Its use is similar to that of TO11DT.

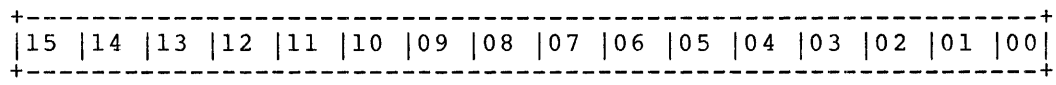
TO11AD



Byte address in PDP-11 memory where next byte received will be stored

TO11AD This register is used by the PDP-11 during byte transfers. TO11AD contains the address of the area in PDP-11 memory where the data from the KL is to be written. The DTE20 keeps this register updated with the currently correct address as the TO-11 byte transfer progresses.

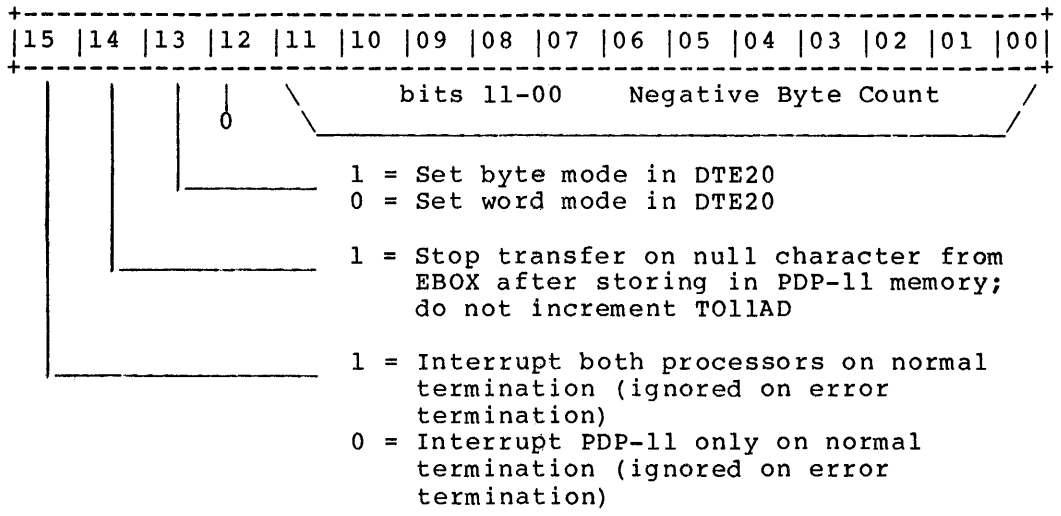
TO10AD



Byte address in PDP-11 memory from which next byte to be transferred is to be taken

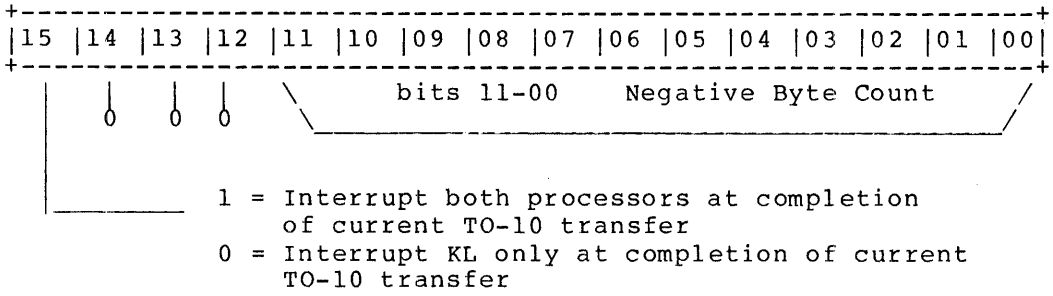
TO10AD This register is also used by the PDP-11 during byte transfer operations. TO10AD contains the address of the area in PDP-11 memory where the data to be transferred resides.

TO11BC



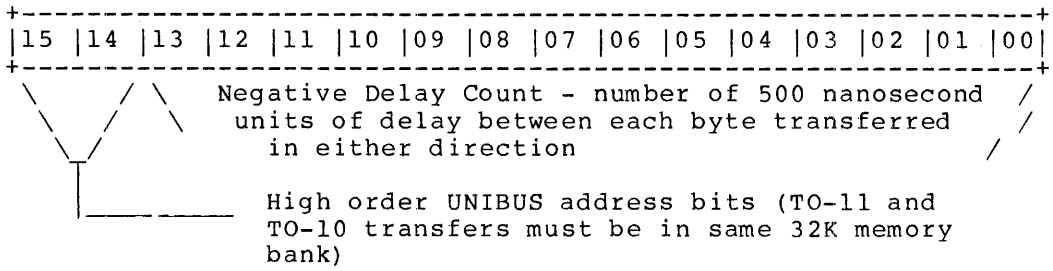
TO11BC When the PDP-11 loads this register (the TO-11 byte count) with the number of bytes to be transferred, the DTE20 begins the TO-11 transfer operation.

TO10BC



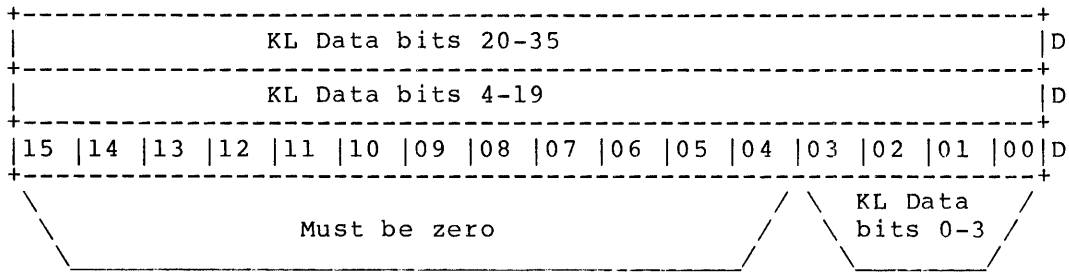
TO10BC When the KL loads this register the DTE20 initiates the TO-10 transfer. The PDP-11 never writes to this register. The KL sets the register by writing into its EPT. The count may not include everything the PDP-11 wishes to send (in the case of a "scatter write").

DLYCNT



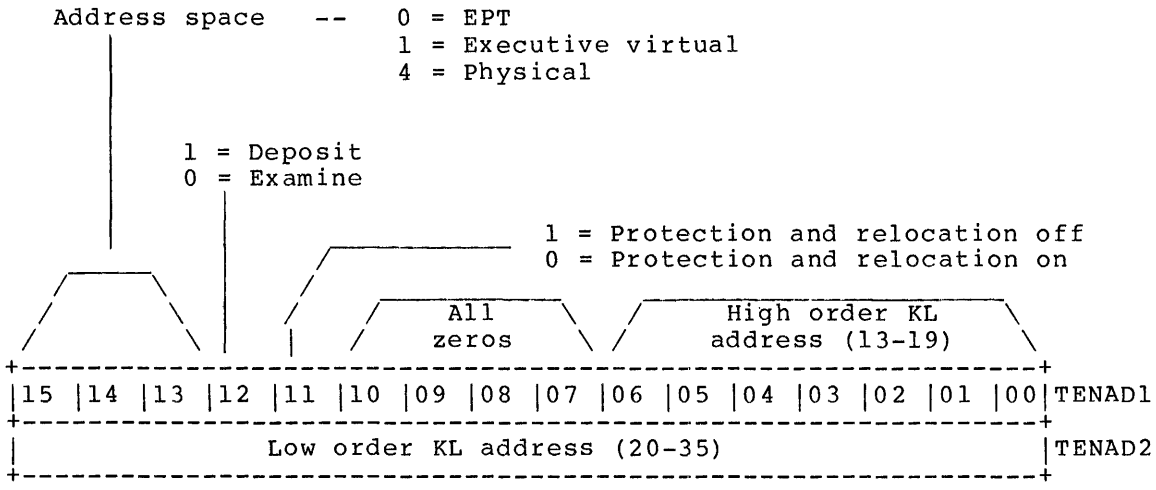
DLYCNT Since the DTE20 is clocked from the EBUS clock module, which runs at a different rate from the PDP-11 clock, a compromise in timing must be effected in order to transfer data over the DTE20. Therefore, the PDP-11 sets this register to notify the DTE20 of the speed at which to carry out byte transfer operations. The register contains the number of 500 nanosecond units of delay that should come between two consecutive byte transfers.

DEXWD1-3



DEXWD1-3 During a deposit or examine operation, the data being deposited or examined appear in these three registers. However, DEXWD1 must always contain zeros in the high-order bits.

TENAD1-2



TENAD1-2 The PDP-11 uses these two registers in examine and deposit operations to specify the KL address. When the PDP-11 writes TENAD2 the deposit/examine operation is initiated; thus, all necessary data must be written to DEXWD1-3 before TENAD2 is written.

8.3 USING THE DTE20 REGISTERS

Each of the four DTE20 operations (Deposit/Examine, TO-10/TO-11 transfers, Doorbell functions, and Diagnostic operations) has a preliminary phase where control information and/or data is loaded into the DTE registers. This is always done before the interface begins any operation. Sections 8.3.1 through 8.3.4 describe, operation by operation, the information and data that must be loaded into the registers.

8.3.1 Deposit and Examine

For the Deposit operation, the following information is always loaded into the indicated registers in the RAM by the PDP-11 processor:

Register	Data Loaded
DEXWD3	Data Word 3
DEXWD2	Data Word 2
DEXWD1	Data Word 1
TENAD1	Address Word 1
TENAD2	Address Word 2

For an Examine operation, the KL address is loaded into TENAD1 and TENAD2. The result of the Examine is put into DEXWD1 and DEXWD2. A Deposit operation loads the contents of the three data words into the KL address specified by the address words. Bit 12 of Address Word 1 (TENAD1) specifies whether the operation is to be Examine or Deposit. If bit 12 is set (=1), the operation is a Deposit; if bit 12 is clear (=0), the operation is an Examine. For a privileged front end, the protection bit (bit 11 of TENAD1) can be set to one by the software to

perform an unprotected Deposit or Examine. For unprotected Deposits and Examines, the address space field (bits 15-13) specifies the type of address. Currently, three types of address space can be specified:

Bits 15-13	Space addressed
0	Executive Process Table
1	Executive Virtual Address Space
4	Physical Address Space

When Address Word 2 is loaded, the operation begins.

8.3.2 Transfer Operations

The TO-10 and TO-11 transfer operations pass variable length data between the two processors. The sender of the data must specify the address of the source string. The KL controls the address either to or from the KL by byte pointers in the Executive Process Table (EPT). The PDP-11 controls the address to or from the PDP-11 by two locations in the DTE (one word for each direction of transfer). It is the responsibility of the receiver to control scatter writes. The PDP-11 specifies the transfer rate (with the delay count) and the type of transfer. Bit 13 in the TO-11 Byte Count word controls whether the DTE is in byte mode or word mode (1=byte mode, 0=word mode). Byte mode transfers 8-bit bytes while word mode moves 16-bit bytes.

When transferring string data from the KL to the PDP-11, the following DTE registers must be loaded by the PDP-11:

Register	Data Loaded
TO11BC	TO-11 Byte Count
TO11AD	TO-11 PDP-11 Memory Address

The TO-11 Byte Count register holds a negative number whose absolute value is equal to the number of bytes to be transfered. As each byte is transferred, this register is incremented by one. When the byte count reaches zero, the transfer is over. A special provision in the byte count word allows for gather reads. This provision allows the receiver of data (and only the receiver) the option of being interrupted before the transfer is complete. At this point, another transfer can be started (without reloading all of the parameters) just by changing the address. The transfer in progress continues from the new address. On termination of the transfer, the PDP-11 is interrupted. (See Figure 8-5.) The KL can be interrupted also if this is desired.

The TO-10 transfer process is very similar to the TO-11 process. The PDP-11 loads the following registers:

Register	Data loaded
TO10AD	TO-10 PDP-11 Address

The KL loads the following register:

TO10BC	TO-10 Byte Count

The transfer starts when the TO10BC register is loaded. The TO10DT register is used in the same manner as TO11DT; that is, TO11DT is a temporary buffer used in conjunction with the delay count. The KL is interrupted when the transfer is finished. (See Figure 8-5.)

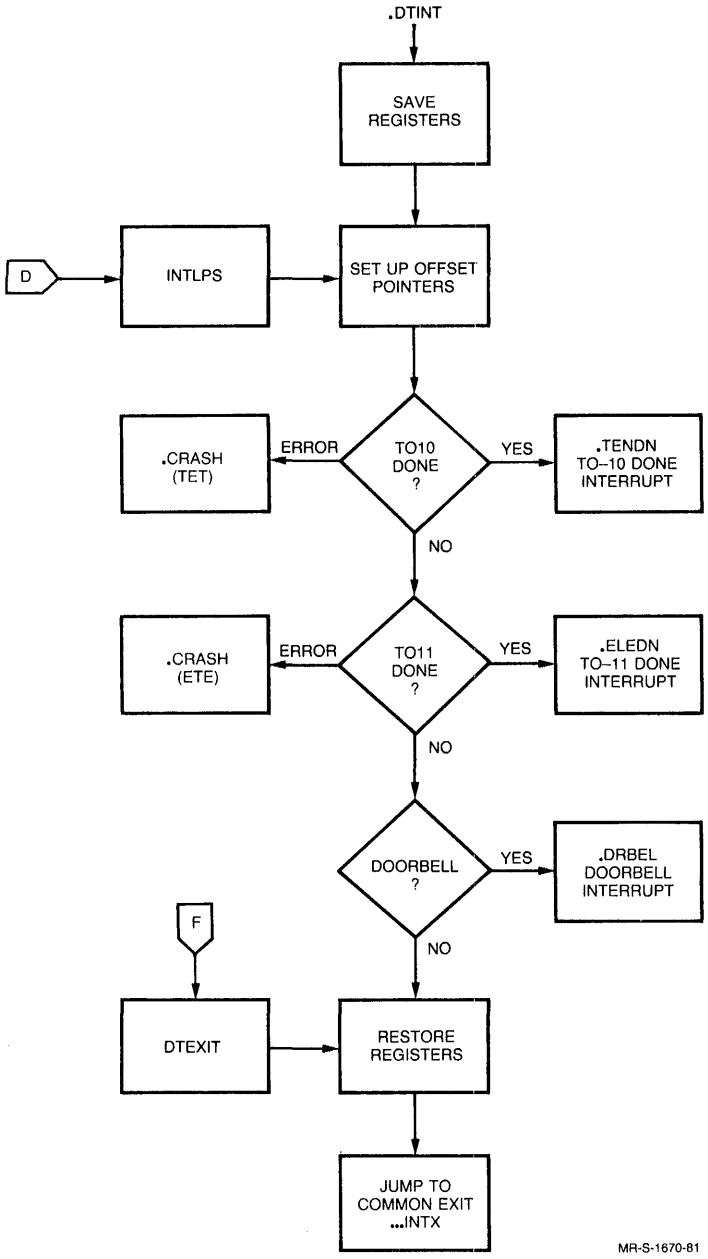


Figure 8-5: DTE Interrupt Handler (part 1 of 5)

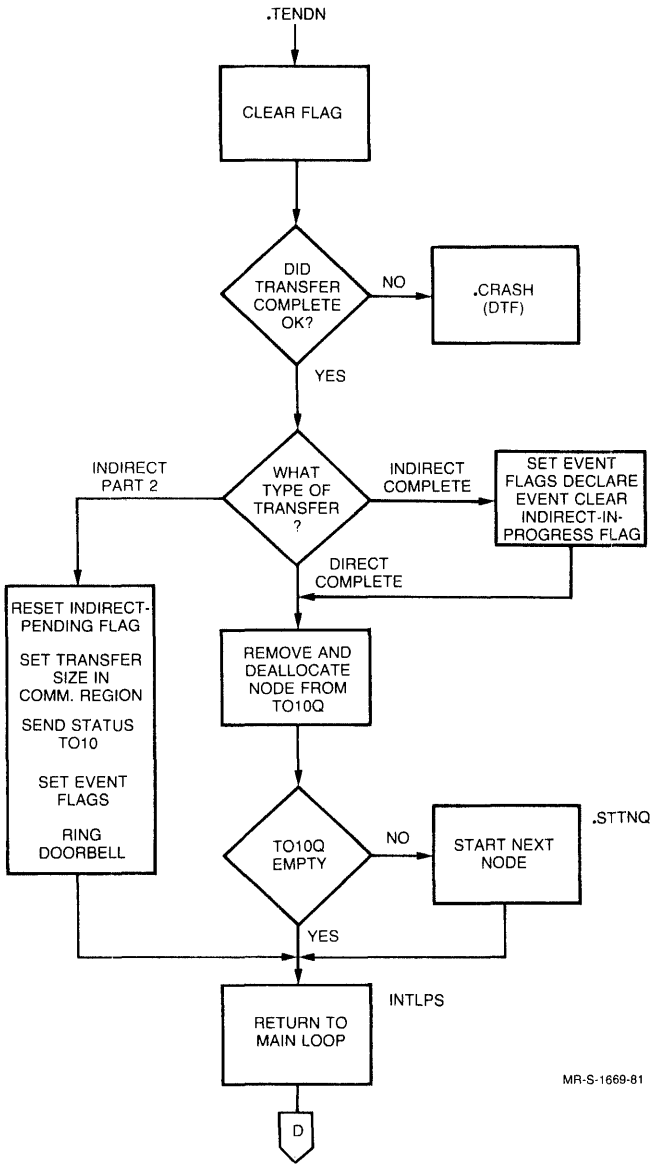


Figure 8-5: DTE Interrupt Handler (part 2 of 5)

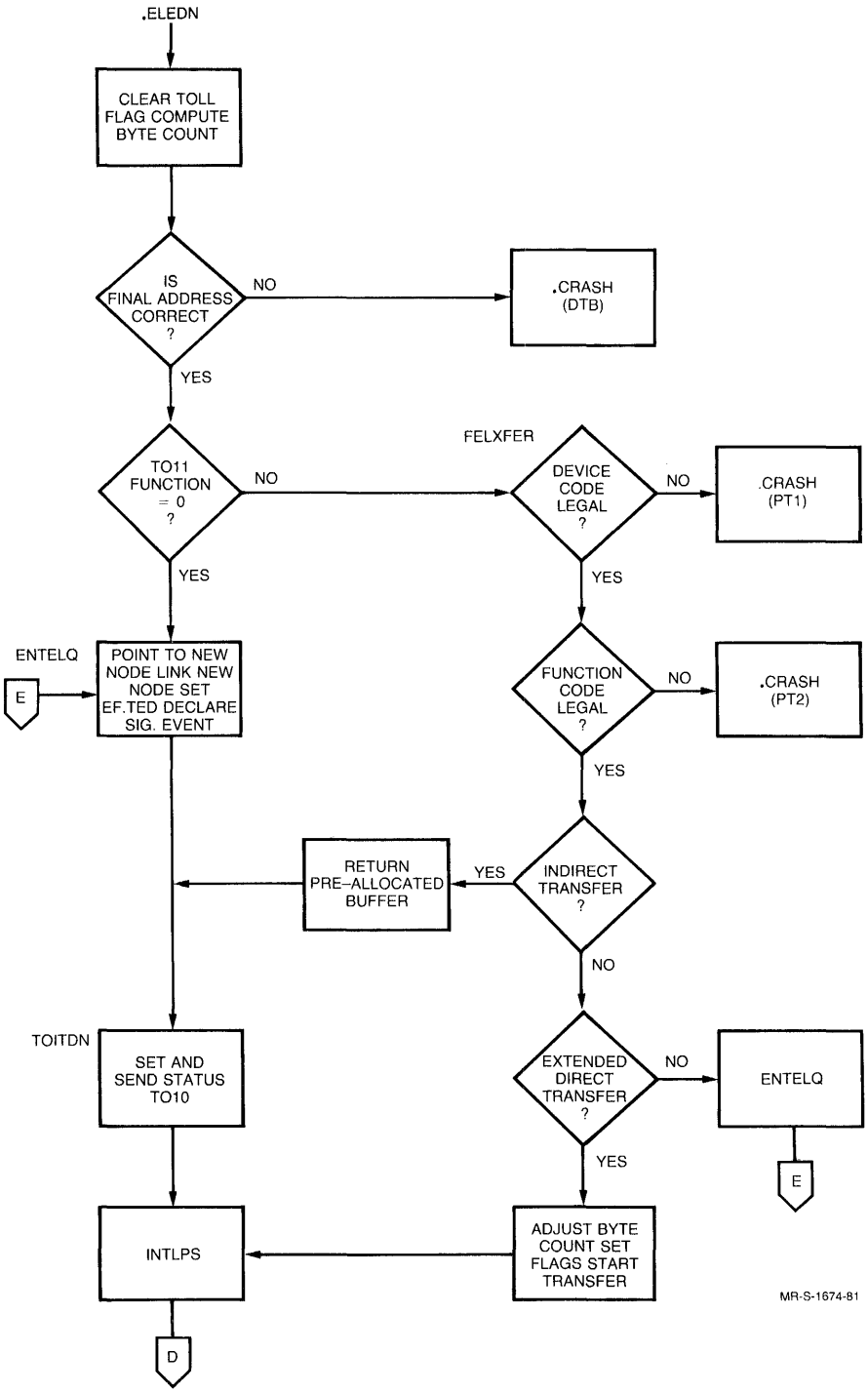


Figure 8-5: DTE Interrupt Handler (part 3 of 5)

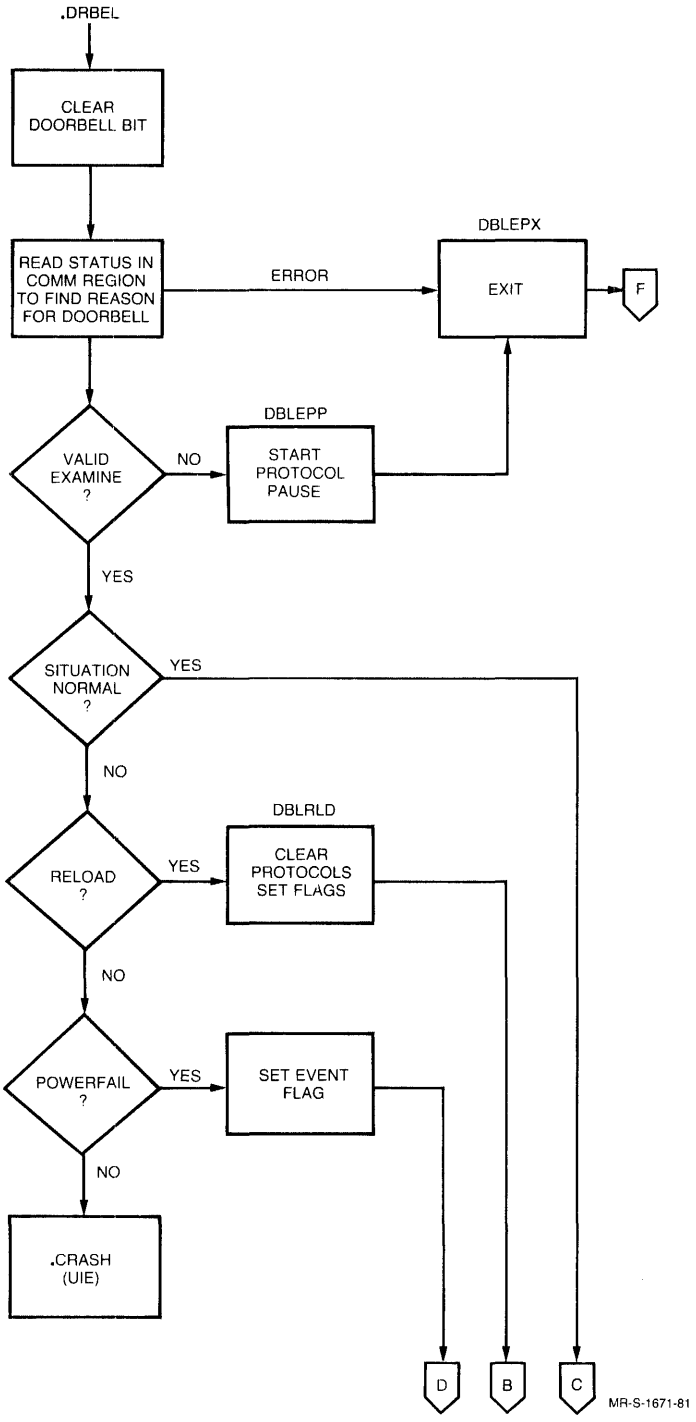


Figure 8-5: DTE Interrupt Handler (part 4 of 5)

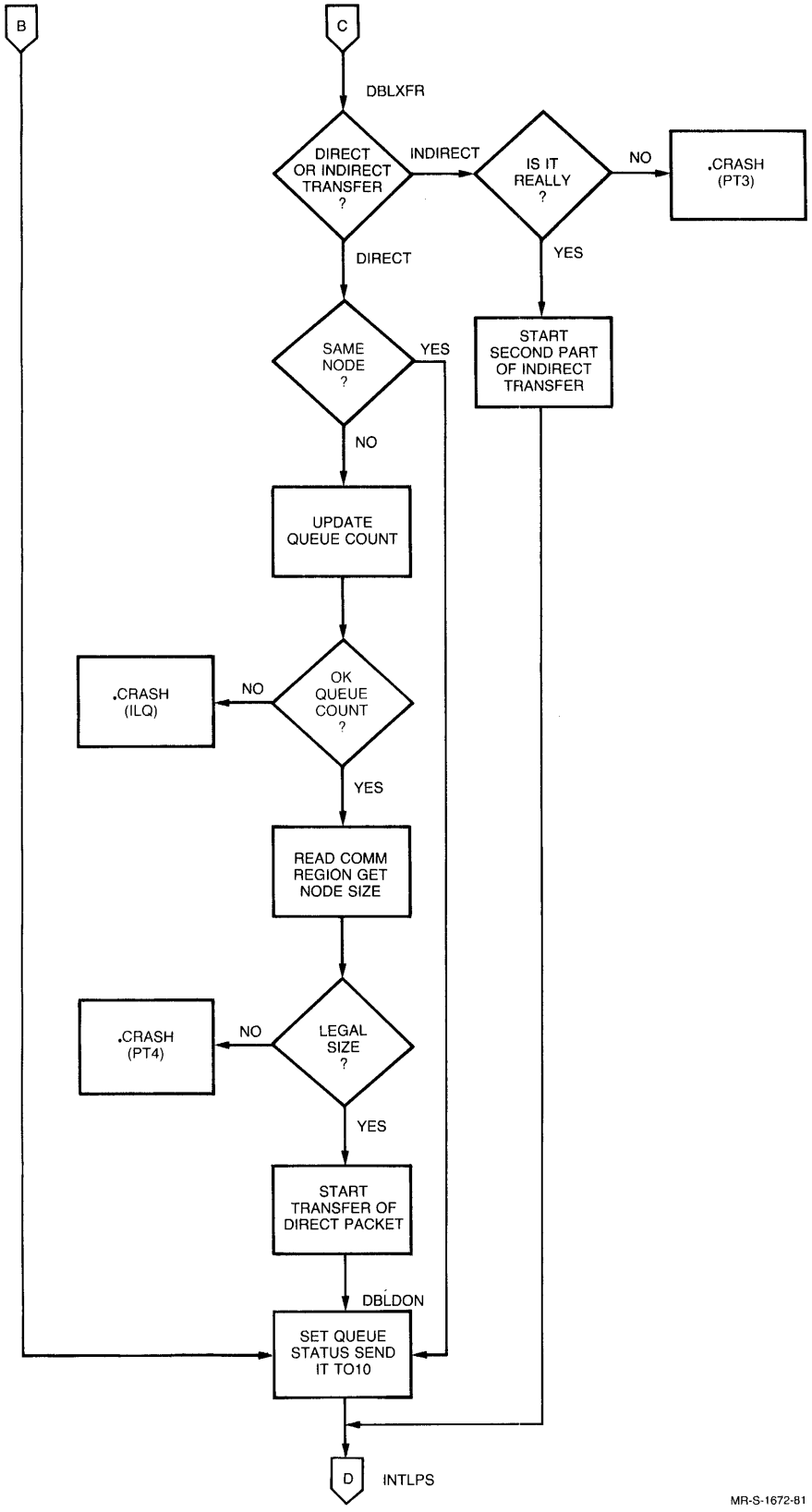


Figure 8-5: DTE Interrupt Handler (part 5 of 5)

8.3.3 Doorbell Function

The doorbell function allows the KL to interrupt each PDP-11 connected to it by a DTE20, and vice versa. The doorbell consists of a programmable interrupt and a status bit. In order for the PDP-11 to interrupt the KL, the PDP-11 sets the Request-10 interrupt flip-flop (bit 8 in the DTE Status Word). When this bit is set, the DTE20 generates an interrupt in the KL with a status bit set in the CONI word (bit 26 in TO10DB) to indicate that the PDP-11 CPU has requested an interrupt of the KL. (See Figure 8-5.)

This procedure works in a reversed but identical manner for the KL interrupting the PDP-11. The KL sets the Request-11 interrupt by doing a CONO to the DTE20. The PDP-11 discovers the cause for the interrupt by looking at TO11DB (bit 11 in the DTE status word). Communication is then performed by loading one or more words in the Communications Region of KL memory. The Deposit and Examine features are used by the PDP-11 to gain access to these words.

8.3.4 Diagnostic Functions

The PDP-11 front end can diagnose problems in the KL using the diagnostic functions of the DTE20. The diagnostic functions are performed over an electronically isolated portion of the EBUS known as the diagnostic bus. This bus contains Diagnostic Select lines to tell the KL which diagnostic function the front end wishes to perform. The bus also has lines to carry data that helps the KL hardware interpret the Diagnostic Select lines.

To perform the diagnostic functions, set the bits in register DIAG1 that correspond to the code for the function you wish to perform. When you set bit 0 (DCOMST) in DIAG1, the function code is sent to the KL. When the DCOMST bit is zero, the DTE20 has sent the diagnostic function to the KL. Bit 2 of DIAG1 (DSEND) has no effect on the transfer of the function code; DSEND deals with diagnostic data transfer only. Diagnostic data transfer takes place only when bit 3 (DIKL10) is set to one.

8.4 PROTOCOLS

The protocol used between the KL and the PDP-11 front end is a tightly coupled communications protocol designed for use in exactly this environment. Two levels are included in this protocol: Secondary Protocol and Primary Protocol.

8.4.1 Secondary Protocol

Secondary Protocol uses the Doorbell and Deposit/Examine features of the DTE20. This protocol is only used in special situations such as booting the KL, or in emergencies where normal communications with the KL are not available. The code to support Secondary Protocol is in module BOOT of RSX-20F.

Only the privileged PDP-11 can run Secondary Protocol because it requires privileged Deposits and Examines.

8.4.2 Primary Protocol

Primary Protocol is the main protocol used for communications and uses the Deposit/Examine, Byte Transfer, and Doorbell features of the DTE20. The switch to Primary Protocol is made just before the [PS MOUNTED] message appears at the CTY.

Primary Protocol is a queued protocol because the DTE20 is used by several tasks. Multiple tasks cannot be allowed to use the DTE20 whenever they desire, or confusion would result. Therefore, tasks that want to use the DTE must line up the data in a queue and wait their turn.

8.5 QUEUED PROTOCOL

The queued protocol driver is responsible for many things: controlling the exchange of data between the KL and the PDP-11, scheduling the transmission of information packets sent across the DTE, and interfacing between the KL and the PDP-11 device drivers that must communicate with it (terminals, line printers and card readers). The queued protocol driver places output data in the thread packets for terminals and line printers. The queued protocol driver also takes data from card readers and terminals, bundles them into packets and sends them off to the KL. When device status information is needed, it is the queued protocol driver that gathers the information for those devices that must report to the KL.

The following list includes all the functions of the queued protocol driver. Each function is listed along with its associated function code, which is used by both the KL and the front end to recognize the type of request just received.

Code	Function
1	Request for Initial Status
2	Here is CTY Alias
3	String Data This function is the general data-transferring mechanism of the protocol.
4	Line/Character Data This function allows the protocol to handle data transfers for several lines with a single function, which cuts down on overhead by reducing the number of messages transferred.
5	Return Device Status This function requests the status of the device from the other processor.
6	Set Device Status This function requests the device status to be set to the specified values.
7	Here is Device Status This function is the response to function 5 (Request Device Status).
10	Unused

Code	Function
11	Return Time of Day This function is used to determine the other processor's current system date and time.
12	Here is Time of Day This function is the response to function 11 (Return Time of Day).
13	Flush Output Device Queue This function provides the ability to deal with CTRL/O by flushing all output waiting for output to the specified device.
14	Send All This function causes a specified string to be typed on all TTY-type devices connected to the front end.
15	Device Dial-up This function causes the PDP-11 to raise DTR for the specified dataset line.
16	Device Hang-up This function causes the PDP-11 to hang up (drop DTR for) the specified dataset line.
17	Acknowledge Device Done This function is used to notify the other processor that a data transfer operation has been completed. The passing of this signal allows the buffer space that was taken up with the data just transferred to be freed.
20	X-OFF (TTY only) This function is used to produce the effect of CTRL/S. This causes the data transfer in progress to be suspended until further notice (the notice will be in the form of a CTRL/Q).
21	X-ON (TTY only) This function is used to produce the effect of CTRL/Q. This causes the data transfer currently suspended (if there is one) to be continued.
22	Set TTY Speed This function is used to inform the other processor of the speed of a given TTY line. The front end will use this information to set the line speed. The KL will need the information when the KL has been reloaded and is reentering Primary Protocol.
23	Set Line Allocation This function sets the maximum amount of data that a device can accept between acks.
24	PDP-11 Reboot Word This function provides the KL with the settings of the PDP-11 switch register.
25	Acknowledge All This function is used to restart a data transfer operation. The KL, for example, may use it when the KL has temporarily left Primary Protocol.

Code Function

26 Start/Stop Line
This function is used to enable and disable input processing for the line specified.

27 Enable/Disable Remotes
This function is used to enable and disable the modem control; when the current state is enabled, the telephone may be answered by the front end.

30 Load Line Printer RAM
This function is used by the KL to notify the front end that the line printer RAM needs reloading. The front end will, upon receipt of this message, proceed to load the RAM.

31 Load Line Printer VFU
This function is used by the KL to notify the front end that the line printer VFU needs to be reloaded. Upon receipt of this message, the front end will load the RAM.

32 Suppress Send-All
This function is used to suppress the system messages for a specified line. It is the equivalent of the TOPS-20 command REFUSE SYSTEM-MESSAGES.

33 Send KLINIK Parameters
This function is used to notify the other processor that the KLINIK parameters are about to be sent.

34 Enable/Disable Local X-OFF
This function is used to allow (or disallow) the front end to process the X-OFF character itself, rather than waiting for the KL to process the character

35 Break Through Write
This function sends the associated packet to the front of the output queue.

36 Set Host Debug
This function set the KL to debug mode. No message is printed during a KL halt and no reloads are performed.

37 Clear Host Debug
This function causes the KL to exit debug mode. If the KL halts the system message is sent and the KL is reloaded.

When two processors communicate, they require an area that both can access to exchange information. This area is in KL memory because the KL can not access PDP-11 memory through the DTE. The area where this common data is stored is the Communications Region.

Recall that the first part of the region is a header area. Following this, each processor that is connected to the KL has its own communications area. A minimal configuration has two communications areas, one for the KL and one for the PDP-11 front end. If another PDP-11 is attached for data communications, there will be three communications areas. Each communications area has one section of data about itself and one or more sections for the other processors to which it is connected.

The KL and the PDP-11 also use the Executive Process Table (EPT) to communicate. The EPT occupies a page in KL memory in which several words are reserved for DTE communications. The location of the EPT is always known because a hardware register points to it.

The EPT stores the KL addresses for byte-transfer operations and tells the PDP-11 where in KL memory it may Deposit and Examine. The relocation and protection words are set once by the KL. The PDP-11 reads these words, and stores them so that it will not have to read them again. All PDP-11 Deposits and Examines of KL memory are checked by the hardware using the relocation and protection words.

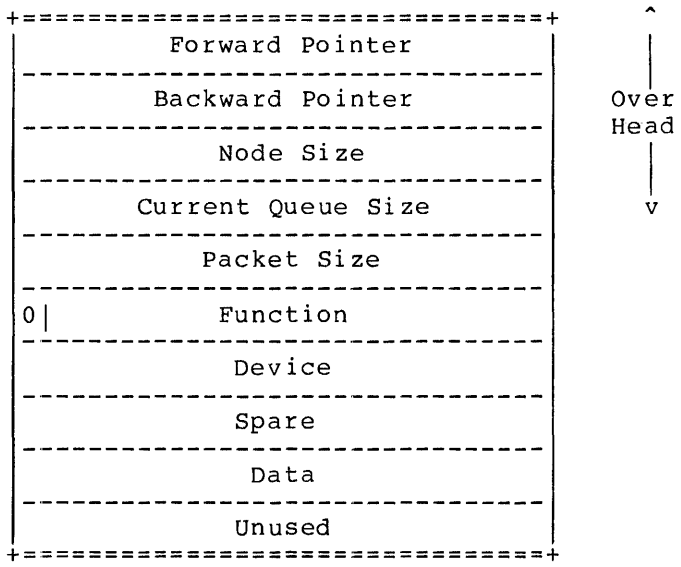
8.6 DTE20 DRIVER LOGIC

The DTE20 driver is responsible for the exchange of data between the front end and the KL. It can transfer data in both directions simultaneously. There are three types of data packets transferred across the DTE20:

- Direct packets
- Extended direct packets
- Indirect packets.

8.6.1 TO-10 Direct Packets

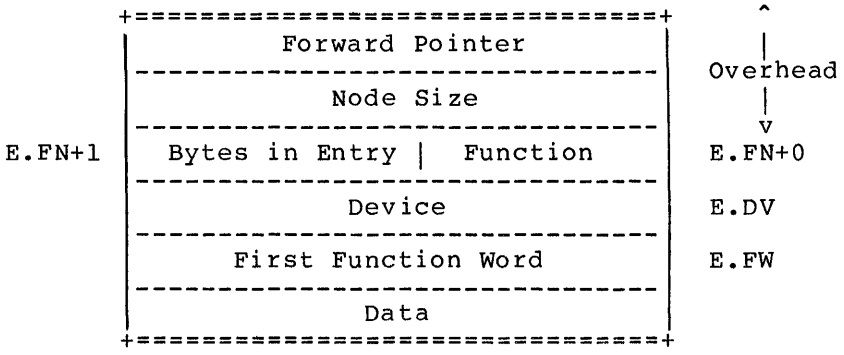
A TO-10 direct packet always contains at least 12 (octal) contiguous bytes and consists of a Header and one or more words of data. A Header is always sent across the DTE20 as a single transfer.



The first four words of the packet are not transfered to the KL. They contain information necessary to manage the queue in PDP-11 memory. The packet that is actually transfered across the DTE20 during a direct transfer is always at least 12 (octal) bytes and consists of words beginning with Packet Size and ending with the Data word.

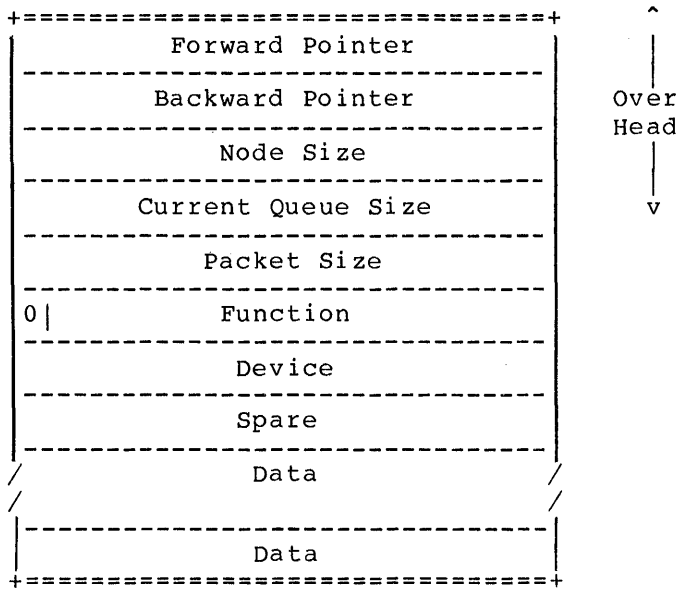
8.6.2 TO-11 Direct Packets

TO-11 direct packets are essentially the same as TO-10 direct packets. However, during a TO-11 direct transfer the spare word is discarded by the DTE20 device driver when a packet is received by the front end, also the byte size and function code are compressed into one word. Direct transfers are used to transfer a single data word. The sender always considers the direct transfer to be a single transfer. The receiver may fragment the transfer and hide the TO-11 Done interrupt from the sender until the entire transfer has been completed.



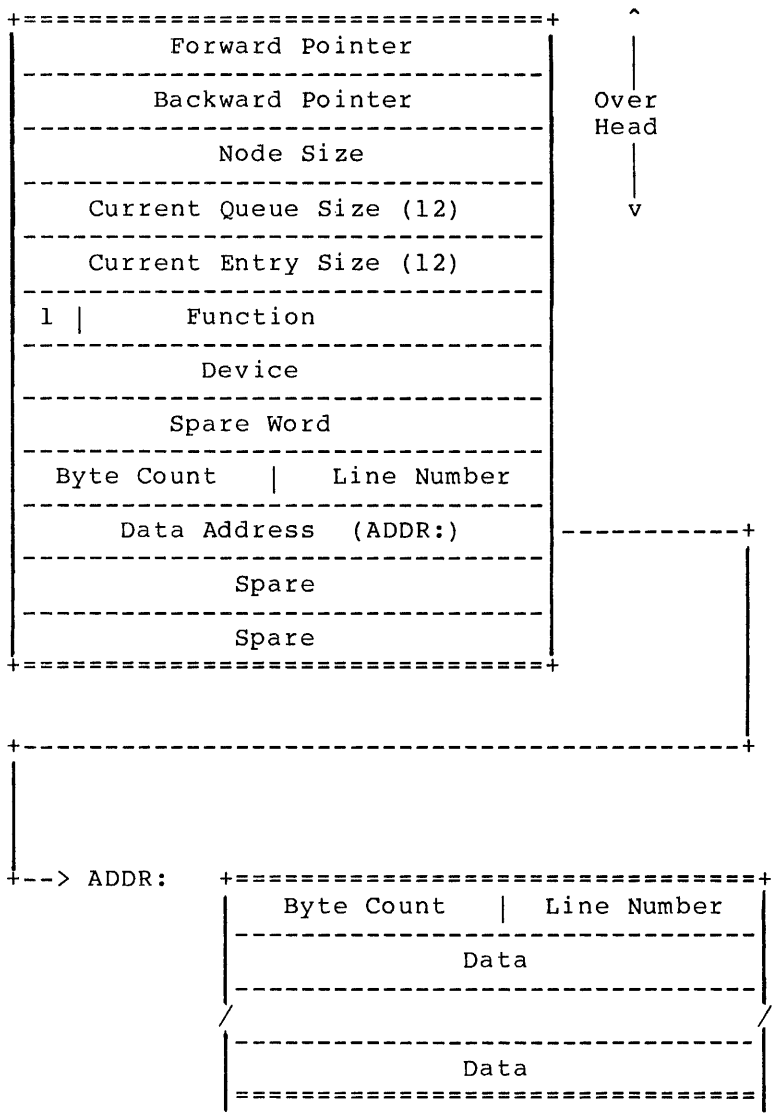
8.6.3 TO-10 Extended Direct Packets

A TO-10 extended direct packet consists of a Header and more than one data word. The Header portion contains a count of the number of bytes in the extended portion of the packet. Direct packets can be sent only from a contiguous buffer. The extended direct data packet has the following format.



8.6.4 TO-10 Indirect Packets

A TO-10 indirect packet consists of a Header portion and a data portion. The header and data portions do not necessarily occupy contiguous locations in PDP-11 memory. The header portion contains the total size of the entire transfer and a pointer to the indirect or data portion of the packet. The Header is always 12 octal bytes long. TO-10 Indirect packets have the following format:

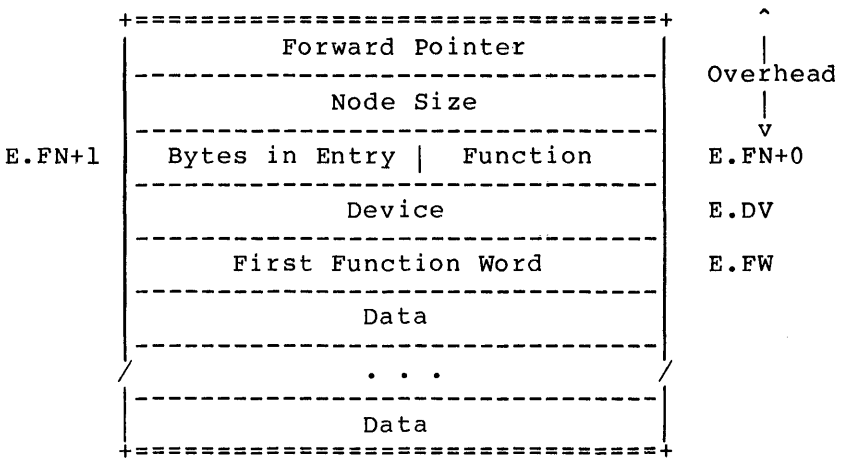


An indirect transfer takes place in two parts; first the header portion and then the data. Not all of the header shown above is transfered across the DTE20. The Header transfer always consists of 12 (octal) byte starting with the function word and ending with the data address word.

The Indirect Flag set in the function word indicates that the transfer is an indirect transfer.

8.6.5 TO-11 Indirect Packets

The To-eleven Indirect packet is essentially the same as the TO-10 indirect packet. However the TO-11 indirect packet occupies contiguous memory locations in PDP-11 memory.



8.6.6 Register Conventions

The DTE20 driver uses the following register conventions:

R0 Points to the base address of the DTE20 in the I/O page.

R1 Points to the DTE20 Status Register in the I/O page.

R3 Points to the Processor Table in the data base.

8.6.7 DTE20 Device Driver Functions

The DTE20 Device driver performs the following functions:

- .DTINT - Interrupt Head and service dispatch
- .TENDN - Services the 10-Done interrupt
- .ELEDN - Services the 11-Done interrupt
- .DRBEL - Services the Doorbell interrupt
- .STELD - Starts a TO-11 Direct Transfer
- .STELI - Starts a TO-11 Indirect Transfer
- .KPALV - Does Keep-Alive processing
- .STTNQ - Starts a TO-10 Transfer

Whenever a request for a transfer of data between the KL and the front end processor is made .DTINT handles the request. .DTINT is responsible for the following:

- Saving all registers on the stack
- Restoring all registers from the stack
- Setting up register R0 with a pointer to the base of the DTE20
- Setting up register R1 with a pointer to the DTE20 Status Register
- Setting up register R3 with a pointer to the Processor Table
- Checking for TO-10 or TO-11 termination errors (TET)
- Checking for TO-11 Memory Parity errors (ETE)

A TO-11 transfer starts when the KL rings the doorbell indicating that a transfer is ready. The KL has already set up pointers to the packet to be transferred in the TO-11 Communications Region. It has also set up its pointer in the DTE20 hardware. RSX-20F reads the size of the transfer in .QSIZE word in the Communications Region.

A .QSIZE word containing a count greater than 12 octal indicates that the transfer is an extended direct. The RSX-20F DTE20 Driver then attempts to allocate a buffer from the Free Pool to accept the transfer. If the allocation fails RSX-20F blocks the DTE20, inhibiting any further transfers until the buffer can be allocated. Each time a buffer is returned to the Free Pool an attempt is made to unblock the DTE20. Once the buffer has been allocated, RSX-20F conditions the DTE20 not to show the next TO-11 Done interrupt to the sending processor (the KL). RSX-20F then reads the Header portion into its fixed buffer in low core. A TO-11 Done interrupt signals that the Header transfer is complete. This interrupt is not seen by the KL, but indicates that the DTE20 has completed its transfer of the Header portion. The Header portion is then copied from the fixed buffer to the allocated buffer and the DTE20 is set to point to the allocated buffer. The DTE20 is then conditioned to show the next TO-11 done interrupt to both processors. This decision is based on the the transfer size in the .QSIZE word. If the size of the transfer is exactly 12 octal bytes, both processors are interrupted at the completion of the transfer. Otherwise, only the the PDP-11 is interrupted. The Function Word in the fixed buffer is cleared to indicate that the packet may be entered into the TO-11 Queue on the next TO-11 done interrupt. The remainder of the message is transferred across the DTE20 to the allocated buffer. When the extended portion has been transferred to the allocated buffer, a TO-11 Done interrupt is seen by both processors.

In other words, the KL does not see a TO-11 Done interrupt until the entire transfer is complete. RSX-20F, on the other hand, sees two TO-11 Done interrupts. The first indicates that the Header portion of the transfer has been completed; the second indicates that the entire transfer has been completed. At this point, the packet is entered into the TO-11 Queue.

If RSX-20F receives a Doorbell interrupt and the .QSIZE word indicates a transfer of exactly 12 octal bytes, it sets up for a direct transfer. First it attempts to allocate a buffer from the Free Pool to accept the transfer. If the allocation fails, RSX-20F blocks the DTE20 until a buffer is available. Once a buffer becomes available, RSX-20F transfers the Header into its fixed buffer in low core. Note that since the transfer is exactly 12 octal bytes, both processors are interrupted. The DTE20 driver then examines the Indirect semaphore in the Function Word. If this semaphore is not set, the transfer is a simple direct transfer, and RSX-20F copies the Header into the allocated buffer and immediately enters that buffer into the TO-11 Queue.

If the Indirect semaphore is set in the Function Word the transfer is an indirect transfer and only the Header has been read. The size of the entire transfer is contained in this Header. Since the allocated buffer contains only enough space for the header portion, RSX-20F returns it to the Free Pool. The Header portion remains in the fixed buffer in low core in anticipation of the second part of the indirect transfer. The KL has seen the TO-11 done interrupt indicating that the Header has been successfully transfered.

The KL will later set the Indirect-in-Progress semaphore and ring the Doorbell to indicate that the indirect portion of the transfer is ready. RSX-20F answers the Doorbell and sees that the Indirect-in-Progress semaphore is set. It then checks to see if the Header in the fixed buffer contains an indirect Header (the Indirect semaphore in the function word is set). RSX-20F now attempts to allocate a buffer to accept the entire transfer. That is, both the Header and the data portion of the transfer will be stored in contiguous locations in the PDP-11's memory. If the allocation fails, RSX-20F blocks the DTE20 until the buffer becomes available. Once the buffer is allocated, RSX-20F copies the Header from the fixed buffer into the allocated buffer, then starts the transfer of the indirect portion across the DTE20 into the allocated buffer. When the transfer is complete, both processors see a TO-11 Done interrupt signaling the completion of the transfer.

An indirect transfer is essentially two separate transfers. First the Header portion (12 octal bytes) is transferred. This is treated as a direct transfer until RSX-20F reads the Indirect Flag in the Function word. It then expects that the next transfer will be the indirect portion of the message. The Header contains the byte count of the entire transfer. When the KL signals that the indirect portion is ready by ringing the Doorbell, RSX-20F reads the indirect portion and stores the entire message in the contiguous buffer that has been allocated.

CHAPTER 9

ERROR DETECTION AND LOGGING

9.1 THE KEEP-ALIVE COUNT

The KL and the PDP-11 watch each other to make sure that the other does not crash. The mechanism they use is called the "Keep-Alive Count." The Keep-Alive Count for each processor is a word in the Communications Region of KL memory. Both processors have clock interrupts regularly. During the servicing of those clock interrupts, the processors increment their own Keep-Alive Counts and check the other processor's Keep-Alive Count. RSX-20F also transmits its Keep-Alive count once every 256 DTE20 transfers. If the count has not changed after a certain number of interrupts, then that processor is assumed to be hung or crashed and must be reloaded. If the KL goes down, RSX-20F schedules the TKTN task to run. TKTN shuts down the protocol, prints a message, and schedules the KLERR. Refer to Section 9.3.3 for TKTN messages.

The front end must go through the DTE20 in order to update and examine the Keep-Alive Counts, because both copies of the Keep-Alive Count are located in the Communications Region of KL memory. The KL's Keep-Alive Count is kept in CMKAC, the sixth word in the KL's area of the Communications Region. The PDP-11's Keep Alive Count is stored in the sixth word of the PDP-11's area of the Communications Region.

9.2 KLERR

When the front end notices that the KL has not responded by the Keep-Alive Count mechanism, KLERR checks the status of the Retry flag (refer to the SET RETRY command in Section 4.4 for more information). If the Retry flag is not set, KLERR checks the setting of the Reload flag. If the Reload flag is set, KLERR reloads the KL. If the Reload flag is not set, the front end ignores the fact that the KL is not running and continues its processing.

If, on the other hand, the Retry flag is set, the front end gives the KL a chance to save its context (if possible) by executing the instruction at location 71, which transfers control to a KL routine that attempts to save the context information. When the context has been saved, KLERR determines whether to reload the KL depending on the state of the Reload flag. (It is usual for the KL to request a reload during the execution of the context-saving subroutine.)

KLERR has several important functions. When it realizes that the KL has not responded, KLERR tries to read as much information as it can from registers in the KL by performing function reads over the diagnostic portion of the EBUS. KLERR stores the information it retrieves in the PARSER.LOG file in the front-end file system. When KLERR has completed its function, TKTN schedules the KLINIT task to restart the KL.

9.3 ERROR LOGGING

There are two types of errors for which some information is logged on the CTY. KLERR logs information for KL errors and describes the state of the KL, as well as it could be determined by the front end. RSX-20F logs information about PDP-11 errors. These error types are described in Sections 9.3.1 through 9.3.3.

9.3.1 KL Error Logging

KLERR is alternate name for routine .KLE in PARSER. This routine provides SPEAR with useful information about the state of the KL when a hardware or software error occurs. Because KLERR "takes a picture" of the KL state when an error occurs, KLERR is said to produce a snapshot.

When KLERR runs, the RSX-20F executive determines what type of error has occurred and invokes TKTN (Task Termination Task), which prints an error message (for example, EBUS PARITY ERROR) on CTY and generates an error code for KLERR. TKTN then schedules KLERR to run. If the error is a CLOCK ERROR STOP, KLERR determines if it is a Fast Memory Parity Error, a CRAM Parity Error, a DRAM Parity Error, or a Field Service Probe and modifies the error code appropriately. KLERR then TAKEs a command file based on the error code.

Following is a list of command files and the errors that cause them to be invoked:

1. CLOCK.CMD - Clock Error Stop - Field Service induced for debugging
2. EBUS.CMD - EBUS parity error
3. DEX.CMD - Deposit Examine Failure (PI level 0 interrupt failure)
4. KPALV.CMD - Keep-Alive Ceased
5. TIMEO.CMD - Protocol Timeout
6. FMPAR.CMD - Clock Error Stop - Fast Memory Parity Error
7. CRAM.CMD - Clock Error Stop - CRAM Parity Error
8. DRAM.CMD - Clock Error Stop - DRAM Parity Error
9. HALT.CMD - KL Halted

The information gathered by these command files is logged to the file PARSER.LOG and, when the KL returns to a running state, the information is transferred to ERROR.SYS by the task LOGXFR.

The following examples show the format of a few samples of KLERR output. See Sections 8.2 and 8.3 for descriptions of the functions and bit assignments of the various DTE20 registers.

9.3.1.1 FMPAR Example - The following report was generated by FMPAR.CMD.

```
**********************************************
FRONT END DEVICE REPORT "KLERR" TYPE 205
 LOGGED ON 29-JUN-83 21:17:17          MONITOR UPTIME WAS 0 DAY(S) 0:0:44
        DETECTED ON SYSTEM # 1042
        RECORD SEQUENCE NUMBER: 27398
**********************************************

        Active error bits:
                CLK1-M8526-CLK ERROR STOP H
                CLK3-M8526-CLK ERROR L
                CLK3-M8526-CLK FM PAR ERR H
                CON5-M8525-CON MBOX WAIT L
                MBZ1-M8537-MEM BUSY H

******* LOGGING STARTED  29-JUNE-83 20:58 ,RSX-20F YE15-04
 OUTPUT DEVICES: TTY,LOG
KLE>! FAST MEMORY PARITY ERROR
KLE>CLEAR OUT TTY
 OUTPUT DEVICES: LOG
KLE>WHAT HARDWARE
 KL10 S/N: 1042., MODEL B, 60. HERTZ
        EXTENDED ADDRESSING
        INTERNAL CHANNELS
        CACHE

KLE>FREAD 100:177
FR 100/ 004377 012664
FR 101/ 000000 002600
FR 102/ 000013 410042
FR 103/ 000030 212020
FR 104/ 000000 022424
FR 105/ 000000 401461
FR 106/ 000000 200000
FR 107/ 000000 655602
FR 110/ 000003 021437
FR 111/ 002100 000000
FR 112/ 007760 050414
FR 113/ 000000 026607
FR 114/ 000426 001600
FR 115/ 001107 043202
FR 116/ 001400 032003
FR 117/ 001100 002000
FR 120/ 000000 066102
FR 121/ 000000 066061
FR 122/ 022200 000000
FR 123/ 000100 055132
FR 124/ 256004 000035
FR 125/ 256004 000035
FR 126/ 000000 000000
FR 127/ 000100 055132
FR 130/ 000072 000000
FR 131/ 010056 660000
FR 132/ 014060 360000
FR 133/ 000040 450000
FR 134/ 130064 444000
FR 135/ 120022 240000
FR 136/ 110050 604000
FR 137/ 002070 244000
FR 140/ 020405 010407
FR 141/ 010000 021006
FR 142/ 101210 000000
```

```
FR 143/ 470000 000000
FR 144/ 020405 010407
FR 145/ 440000 021006
FR 146/ 041210 000000
FR 147/ 000000 000000
FR 150/ 000000 000000
FR 151/ 000001 000000
FR 152/ 000000 042104
FR 153/ 000024 042104
FR 154/ 000000 000000
FR 155/ 000001 010001
FR 156/ 000000 052524
FR 157/ 000024 042504
FR 160/ 211007 417402
FR 161/ 211005 276702
FR 162/ 211004 344036
FR 163/ 211000 000462
FR 164/ 211007 417276
FR 165/ 211004 276172
FR 166/ 211004 344766
FR 167/ 211000 000302
FR 170/ 360002 136372
FR 171/ 000040 625522
FR 172/ 105600 074634
FR 173/ 214502 275327
FR 174/ 103000 137164
FR 175/ 235600 167365
FR 176/ 000400 377347
FR 177/ 760200 523715
KLE>SAVE PC
KLE>E REG;E KL

PAR -- [EXAMINE] CFH - CAN'T FIND KL HALT LOOP
KLE>SWEEP
 FM PARITY ERROR-(BLOCK:ADDR/DATA) 2:2/ 000100 055132

PAR -- [SWEEP] XTO - KL EXECUTE TIMED OUT
KLE>CLE OUT LOG

******* LOGGING FINISHED  29-JUNE-83 20:58
```

9.3.1.2 DEX Example - The following report was generated by DEX.CMD.

```
**********************************************
FRONT END DEVICE REPORT "KLERR" TYPE 205
 LOGGED ON 06-JUL-83 18:13:05          MONITOR UPTIME WAS 0 DAY(S) 0:0:57
        DETECTED ON SYSTEM # 1042
        RECORD SEQUENCE NUMBER: 35534
**********************************************

        Active error bits:
                APR1-M8539-APR MB PAR ERR IN H
                APR1-M8539-APR SBUS ERR IN H
                APR2-M8539-APR ANY EBOX ERR FLG H

******* LOGGING STARTED  6-JULY-83 18:06 ,RSX-20F YE15-04
 OUTPUT DEVICES: TTY,LOG
KLE>! DEPOSIT/EXAMINE FAILURE
KLE>CLEAR OUT TTY
 OUTPUT DEVICES: LOG
KLE>WHAT HARDWARE
 KL10 S/N: 1042., MODEL B, 60. HERTZ
        EXTENDED ADDRESSING
        INTERNAL CHANNELS
        CACHE

KLE>E DTE
 DLYCNT: 000000
 DEXWD3: 177777
 DEXWD2: 000000
 DEXWD1: 000000
   KL10 DATA=000000 177777
 TENAD1: 014000  TENAD2: 000455
   ADDRESS SPACE=EPT
   OPERATION=DEPOSIT
   PROTECTION-RELOCATION IS OFF
   KL10 ADDRESS=455
 TO10BC: 010000  TO11BC: 130000
 TO10AD: 070434  TO11AD: 001314
 TO10DT: 000000  TO11DT: 000400
 DIAG1 : 002600
   KL IN RUN MODE
   MAJOR STATE IS DEPOSIT-EXAMINE
   MAJOR STATE IS TO-10 TRANSFER
 DIAG2 : 000000
 STATUS: 012504
   RAM IS ZEROS
   DEX WORD 1
   11 REQUESTED 10 INTERRUPT
   E BUFFER SELECT
   DEPOSIT-EXAMINE DONE
 DIAG3 : 030000
KLE>DE ELE 174432=100
174432\ 040000
KLE>FX 0
KLE>FREAD 100:177
FR 100/ 010000 013044
FR 101/ 000000 003000
FR 102/ 000104 011226
FR 103/ 000020 213224
FR 104/ 000000 023020
FR 105/ 000000 000001
FR 106/ 000000 205420
FR 107/ 000000 654642
FR 110/ 004433 003476
```

```
FR 111/ 000101 055263
FR 112/ 007760 014451
FR 113/ 000000 056411
FR 114/ 000017 001434
FR 115/ 001107 063202
FR 116/ 001500 072003
FR 117/ 001100 002000
FR 120/ 777776 300473
FR 121/ 777630 047316
FR 122/ 000000 000000
FR 123/ 777630 047316
FR 124/ 000000 000000
FR 125/ 160000 000000
FR 126/ 160000 000000
FR 127/ 777776 300473
FR 130/ 000012 000000
FR 131/ 000021 660000
FR 132/ 012261 360000
FR 133/ 023705 454000
FR 134/ 130061 444000
FR 135/ 122032 244000
FR 136/ 110253 604000
FR 137/ 003774 244000
FR 140/ 640513 160501
FR 141/ 300004 000000
FR 142/ 121710 101010
FR 143/ 010410 130010
FR 144/ 640513 160501
FR 145/ 020004 000000
FR 146/ 641710 101010
FR 147/ 020410 130010
FR 150/ 000000 000000
FR 151/ 000001 042000
FR 152/ 000000 040000
FR 153/ 000004 042100
FR 154/ 000000 000001
FR 155/ 000001 042000
FR 156/ 000000 040000
FR 157/ 000024 042100
FR 160/ 401002 016024
FR 161/ 401006 276700
FR 162/ 401006 206004
FR 163/ 401000 000520
FR 164/ 401002 016120
FR 165/ 401006 276564
FR 166/ 401006 206014
FR 167/ 401000 000540
FR 170/ 100102 126722
FR 171/ 000040 735722
FR 172/ 003600 035220
FR 173/ 340002 237322
FR 174/ 112400 177664
FR 175/ 327620 167375
FR 176/ 000000 337365
FR 177/ 760000 573305
KLE>SAVE PC
KLE>E REG;E KL
AD/ 000000 000001
ADX/ 000000 000001
AR/ 000000 000000
ARX/ 000000 000000
BR/ 000000 000000
BRX/ 000000 000000
EBUS/ 401000 000540
```

```
FM/ 000000 000000
MQ/ 046200 000000
PC/ 565201
PC/ 565201
VMA/ 565201
PI ACTIVE: OFF, PI ON: 000, PI HOLD: 020, PI GEN: 000
 OVF  CY0  CY1  FOV  BIS  USR  UIO  LIP  AFI  AT1  AT0  FUF  NDV
                                X
KLE>E VMA;E VMAH
VMA/ 565201
VMAH/ 565201
KLE>SAVE AC-BLOCK
KLE>ST MIC
KLE>SET AC-BLOCK 5
 AC-BLOCK: 5
KLE>XCT 700400 0;E 0
0/ 000000 000100
KLE>XCT 701240 0;E 0
0/ 000000 000003
KLE>XCT 701040 0;E 0
0/ 705000 000005
KLE>!
KLE>!ALL THE POSSIBLE SBUS DIAGS (CAN BE MADE SITE SPECIFIC)
KLE>!
KLE>!FOR DMA-20
KLE>!
KLE>DE 0=100000 0
0/ 705000 000005
KLE>XCT 700500 0;E 1
1/ 046440 005505
KLE>!
KLE>!FOR INTERNAL CORE
KLE>!
KLE>DE 0=0 0
0/ 100000 000000
KLE>DE 2=020000 0
2/ 000000 000000
KLE>DE 4=040000 0
4/ 000000 000000
KLE>DE 6=060000 0
6/ 000000 000000
KLE>XCT 700500 0
KLE>XCT 700500 2
KLE>XCT 700500 4
KLE>XCT 700500 6
KLE>E 1;E 3;E 5;E 7
1/ 000000 000000
3/ 000000 000000
5/ 000000 000000
7/ 000000 000000
KLE>!
KLE>!FOR MOS
KLE>!
KLE>DE 02=200000 0
2/ 020000 000000
KLE>DE 04=200000 1
4/ 040000 000000
KLE>DE 06=200000 2
6/ 060000 000000
KLE>XCT 700500 2
KLE>XCT 700500 4
KLE>XCT 700500 6
KLE>E 3;E 5;E 7
3/ 000000 000000
```

```
5/ 000000 000000
7/ 000000 000000
KLE>DE 10=220000 0
10/ 000000 000000
KLE>DE 12=220000 1
12/ 000000 000000
KLE>DE 14=220000 2
14/ 000000 000000
KLE>XCT 700500 10
KLE>XCT 700500 12
KLE>XCT 700500 14
KLE>E 11;E 13;E 15
11/ 000000 000000
13/ 000000 000000
15/ 000000 000000
KLE>RESTORE AC-BLOCK
KLE>CLE OUT LOG

******* LOGGING FINISHED  6-JULY-83 18:07


KLI -- ? C-RAM DIFFERS AT 2241
KLI -- BAD  052164 030002 100003 122400 000040 22
KLI -- GOOD 052164 030002 100003 122410 000040 22
KLI -- XOR  000000 000000 000000 000010 000000 00
```

9.3.1.3 HALT Example - The following report was generated by HALT.CMD.

```
**********************************************
FRONT END DEVICE REPORT "KLERR" TYPE 205
 LOGGED ON  5-Jul-83 15:08:21          MONITOR UPTIME WAS 0 DAY(S) 0:0:15
        DETECTED ON SYSTEM # 2102
        RECORD SEQUENCE NUMBER: 8
**********************************************

        No error bits are active

******* LOGGING STARTED  5-JULY-83 15:05 ,RSX-20F YB15-04
 OUTPUT DEVICES: TTY,LOG
KLE>! KL HALTED
KLE>CLEAR OUT TTY
 OUTPUT DEVICES: LOG
KLE>WHAT HARDWARE
 KL10 S/N: 2102., MODEL B, 60. HERTZ
        MOS MASTER OSCILLATOR
        EXTENDED ADDRESSING
        INTERNAL CHANNELS
        CACHE

KLE>FX 0
KLE>FREAD 100:177
FR 100/ 000177 602644
FR 101/ 000000 002600
FR 102/ 000013 410222
FR 103/ 000020 212020
FR 104/ 000000 032434
FR 105/ 000000 001421
FR 106/ 000000 640000
FR 107/ 000000 715642
FR 110/ 000003 064607
FR 111/ 000104 000000
FR 112/ 007740 000000
FR 113/ 000000 000000
FR 114/ 000041 000051
FR 115/ 001107 060144
FR 116/ 001400 012003
FR 117/ 001100 002000
FR 120/ 000000 000000
FR 121/ 000000 000000
FR 122/ 001100 002000
FR 123/ 000000 271232
FR 124/ 002000 020000
FR 125/ 000000 000000
FR 126/ 000000 000000
FR 127/ 000000 000000
FR 130/ 000002 000000
FR 131/ 070054 060000
FR 132/ 014064 760000
FR 133/ 000020 414000
FR 134/ 130066 404003
FR 135/ 120024 224003
FR 136/ 104010 604003
FR 137/ 002000 264003
FR 140/ 760505 000103
FR 141/ 100201 000001
FR 142/ 310000 001010
FR 143/ 501212 030407
FR 144/ 650505 000103
FR 145/ 500201 000001
```

```
FR 146/ 540000 001010
FR 147/ 101212 030407
FR 150/ 000000 002000
FR 151/ 000000 042000
FR 152/ 000100 000104
FR 153/ 000124 002004
FR 154/ 000000 002400
FR 155/ 000000 052400
FR 156/ 000000 000125
FR 157/ 000024 002405
FR 160/ 001003 016025
FR 161/ 001006 276701
FR 162/ 001006 204005
FR 163/ 001000 000521
FR 164/ 001003 016321
FR 165/ 001006 276761
FR 166/ 001006 204015
FR 167/ 001000 000241
FR 170/ 360100 126722
FR 171/ 000000 735722
FR 172/ 111600 137230
FR 173/ 200002 377322
FR 174/ 176000 177664
FR 175/ 273420 127365
FR 176/ 000400 337375
FR 177/ 760040 533305
KLE>SAVE PC
KLE>E REG;E KL
AD/ 000000 000000
ADX/ 000000 000000
AR/ 000000 000000
ARX/ 000000 000000
BR/ 000000 000000
BRX/ 002000 020000
EBUS/ 001000 000241
FM/ 000000 271232
MQ/ 001100 002000
PC/ 25514
PC/ 25514
VMA/ 25514
PI ACTIVE: OFF, PI ON: 177, PI HOLD: 000, PI GEN: 000
 OVF  CY0  CY1  FOV  BIS  USR  UIO  LIP  AFI  AT1  AT0  FUF  NDV
       X    X
KLE>E VMA;E VMAH
VMA/ 25514
VMAH/ 25514
KLE>FX 1
KLE>E 0;17
0/ 000000 000000
1/ 000000 000000
2/ 720200 020000
3/ 000000 271152
4/ 000000 271232
5/ 000000 052636
6/ 000000 000000
7/ 000000 000216
10/ 000000 000003
11/ 000000 000140
12/ 000000 613316
13/ 000000 613316
14/ 000000 000000
15/ 777650 401226
16/ 200000 000000
17/ 777731 245527
```

```
KLE>SAVE AC-BLOCK
KLE>SET AC-BLOCK 5
 AC-BLOCK: 5
KLE>XCT 700400 0;E 0
0/ 002000 025514
KLE>XCT 701240 0;E 0
0/ 000000 060166
KLE>XCT 701040 0;E 0
0/ 705001 002666
KLE>!
KLE>!ALL POSSIBLE SBUS DIAGS(CAN BE MADE SITE SPECIFIC)
KLE>!
KLE>!FOR DMA
KLE>!
KLE>DE 0=100000 0
0/ 705001 002666
KLE>XCT 700500 0;E 1
1/ 000000 000000
KLE>!
KLE>!INTERNAL CORE
KLE>!
KLE>DE 0=0 0
0/ 100000 000000
KLE>DE 2=020000 0
2/ 000000 000000
KLE>DE 4=040000 0
4/ 000000 000000
KLE>DE 6=060000 0
6/ 000000 000000
KLE>XCT 700500 0
KLE>XCT 700500 2
KLE>XCT 700500 4
KLE>XCT 700500 6
KLE>E 1;E 3;E 5;E 7
1/ 000000 000000
3/ 000000 000000
5/ 000000 000000
7/ 000000 000000
KLE>!
KLE>!MOS CONTROLLER 10
KLE>!
KLE>DE 2=200000 0
2/ 020000 000000
KLE>DE 4=200000 1
4/ 040000 000000
KLE>DE 6=200000 2
6/ 060000 000000
KLE>XCT 700500 2
KLE>XCT 700500 4
KLE>XCT 700500 6
KLE>E 3;E 5;E 7
3/ 007040 025514
5/ 000500 001000
7/ 002100 000400
KLE>!MOS CONTROLLER 11
KLE>DE 10=220000 0
10/ 000000 000000
KLE>DE 12=220000 1
12/ 000000 000000
KLE>DE 14=220000 2
14/ 000000 000000
KLE>XCT 700500 10
KLE>XCT 700500 12
KLE>XCT 700500 14
```

```
KLE>E 11;E 13;E 15
11/ 006144 436367
13/ 000500 001000
15/ 016120 001400
KLE>SET AC-BLOCK 0
 AC-BLOCK: 0
KLE>E 0;17
0/ 000000 000000
1/ 000000 000000
2/ 720200 020000
3/ 000000 271152
4/ 000000 271232
5/ 000000 052636
6/ 000000 000000
7/ 000000 000216
10/ 000000 000003
11/ 000000 000140
12/ 000000 613316
13/ 000000 613316
14/ 000000 000000
15/ 777650 401226
16/ 200000 000000
17/ 777731 245527
KLE>SET AC-BLOCK 1
 AC-BLOCK: 1
KLE>E 0;17
0/ 000000 000000
1/ 000700 011414
2/ 540000 000006
3/ 000000 000000
4/ 000000 234622
5/ 000000 000140
6/ 000000 000000
7/ 000000 000025
10/ 331200 410000
11/ 000000 000140
12/ 000000 613316
13/ 000000 613316
14/ 000000 000000
15/ 777405 402404
16/ 000000 000000
17/ 777610 007532
KLE>SET AC-BLOCK 2
 AC-BLOCK: 2
KLE>E 0;17
0/ 000000 000000
1/ 000000 000000
2/ 000000 000000
3/ 000000 000000
4/ 000000 000000
5/ 000000 000000
6/ 000000 000000
7/ 000000 000000
10/ 000000 000000
11/ 000000 000000
12/ 000000 000000
13/ 000000 000000
14/ 000000 000000
15/ 000000 000000
16/ 000000 000000
17/ 000000 000000
KLE>SET AC-BLOCK 3
 AC-BLOCK: 3
KLE>E 0;17
```

```
0/ 000000 000000
1/ 000000 000000
2/ 000000 000000
3/ 000000 000000
4/ 000000 000000
5/ 000000 000000
6/ 000000 000000
7/ 000000 000000
10/ 000000 000000
11/ 000000 000000
12/ 000000 000000
13/ 000000 000000
14/ 000000 000000
15/ 000000 000000
16/ 000000 000000
17/ 000000 000000
KLE>SET AC-BLOCK 4
 AC-BLOCK: 4
KLE>E 0;17
0/ 000000 000000
1/ 000000 000000
2/ 000000 000000
3/ 000000 000000
4/ 000000 000000
5/ 000000 000000
6/ 000000 000000
7/ 000000 000000
10/ 000000 000000
11/ 000000 000000
12/ 000000 000000
13/ 000000 000000
14/ 000000 000000
15/ 000000 000000
16/ 000000 000000
17/ 000000 000000
KLE>SET AC-BLOCK 6
 AC-BLOCK: 6
KLE>E 0;17
0/ 000770 400007
1/ 234000 002160
2/ 000000 531000
3/ 000000 205702
4/ 000000 000000
5/ 101500 126241
6/ 000000 000000
7/ 000000 000000
10/ 000000 126241
11/ 000000 000000
12/ 001000 000241
13/ 000000 000000
14/ 000000 000000
15/ 000000 000000
16/ 777733 245531
17/ 777733 245531
KLE>SET AC-BLOCK 7
 AC-BLOCK: 7
KLE>E 0;E 1;E 2
0/ 000000 000000
1/ 000000 000000
2/ 000000 000000
KLE>SET AC-BLOCK 5
 AC-BLOCK: 5
KLE>!
KLE>!NOW TO GET THE PAGE FAIL AND UUO STUFF
```

```
KLE>!
KLE>DE 10=0 370000
10/ 220000 000000
KLE>XCT 200610 500;E 14
14/ 220000 000002
KLE>XCT 200610 501;E 14
14/ 220000 000002
KLE>XCT 200610 502;E 14
14/ 220000 000002
KLE>XCT 200610 424;E 14
14/ 220000 000002
KLE>XCT 200610 425;E 14
14/ 220000 000002
KLE>XCT 200610 426;E 14
14/ 220000 000002
KLE>XCT 200610 427;E 14
14/ 220000 000002
KLE>!CHANNEL LOGOUT
KLE>XCT 701240 17;E 17
17/ 000000 060166
KLE>XCT 242740 000011!LSH 17,11
KLE>XCT 550040 17!HRRZ 1,17
KLE>XCT 200001 0000;E 0
0/ 000000 030366
KLE>XCT 200001 0001;E 0
0/ 000000 025375
KLE>XCT 200001 0002;E 0
0/ 000500 001000
KLE>XCT 200001 0003;E 0
0/ 200000 000002
KLE>XCT 200001 0004;E 0
0/ 002100 000400
KLE>XCT 200001 0005;E 0
0/ 000000 023362
KLE>XCT 200001 0006;E 0
0/ 000000 066041
KLE>XCT 200001 0007;E 0
0/ 200000 164375
KLE>XCT 200001 0010;E 0
0/ 500000 164376
KLE>XCT 200001 0011;E 0
0/ 600007 725000
KLE>XCT 200001 0012;E 0
0/ 254340 720312
KLE>XCT 200001 0013;E 0
0/ 200000 165375
KLE>XCT 200001 0014;E 0
0/ 500000 165376
KLE>XCT 200001 0015;E 0
0/ 600002 567000
KLE>XCT 200001 0016;E 0
0/ 254340 721312
KLE>XCT 200001 0017;E 0
0/ 200000 273322
KLE>XCT 200001 0020;E 0
0/ 500000 273323
KLE>XCT 200001 0021;E 0
0/ 600010 254000
KLE>XCT 200001 0022;E 0
0/ 254340 721746
KLE>XCT 200001 0023;E 0
0/ 200000 276573
KLE>XCT 200001 0024;E 0
0/ 500000 276574
```

```
KLE>XCT 200001 0025;E 0
0/ 600004 446400
KLE>XCT 200001 0026;E 0
0/ 254340 722642
KLE>XCT 200001 0027;E 0
0/ 000000 000000
KLE>XCT 200001 0030;E 0
0/ 000000 000000
KLE>XCT 200001 0031;E 0
0/ 000000 000000
KLE>XCT 200001 0032;E 0
0/ 000000 000000
KLE>XCT 200001 0033;E 0
0/ 200000 303642
KLE>XCT 200001 0034;E 0
0/ 500000 303643
KLE>XCT 200001 0035;E 0
0/ 600000 303650
KLE>XCT 200001 0036;E 0
0/ 254340 723056
KLE>XCT 200001 0037;E 0
0/ 000000 000000
KLE>!
KLE>RESTORE AC-BLOCK
KLE>CLE OUT LOG

******* LOGGING FINISHED  5-JULY-83 15:06
```

9.3.2 PDP-11 Error Logging

Upon encountering a serious PDP-11 error, the front end stops and waits for the KL to reload it. Before the front end dies, however, it prints the following message on the CTY:

```
11-HALT
<code>
```

where <code> is a 3-character error code indicating why the front end crashed. Refer to Appendix A for a list of RSX-20F stop codes.

Whenever RSX-20F discovers a condition that it considers serious enough to cause a crash, it executes an IOT instruction. The 3-character crash code following the IOT is picked up by the crash routine.

For a disk-based PDP-11 operating system, the IOT instruction is used for error reporting. The instruction first executes a trap vector at location 20. From there it dispatches to COMTRP, then to IOTTRP which halts the PDP-11, trying to save as much information from the registers as it can.

The IOT routine stores the crash code and parity-error registers in locations 0 to 3 of PDP-11 memory. This information is readily available in a dump listing. For example, an IOT instruction followed by ASCIZ /DTB/ would result in the following:

```
+---------------------+
!     T    !     D    ! 0
!---------------------!
! Par.Error!     B    ! 2
+---------------------+
```

When looking at the source listings of RSX-20F, notice that a macro is used instead of the IOT instruction. The macro is .CRASH and expands to the IOT plus ASCIZ crash code as stated above.

9.3.3 TKTN MESSAGES

TKTN prints a message before scheduling the KLERR task. A list of the possible messages and their causes follows.

CLOCK ERROR STOP

This is a hardware error. It could be caused by an FM parity error. If the error persists, call your Field Service Representative.

E-BUS PARITY ERROR

This is a hardware error. If it persists, call your Field Service Representative.

FAULT CONTINUATION TIMEOUT

The KL failed to ring the 11's doorbell within 5 seconds after performing the XCT 72 for fault continuation.

KEEP ALIVE CEASED

This can be either a hardware or software problem. Reload the system. If the problem persists, call your Software Support Specialist.

KL HALTED

The KL is in a halt loop. If the reload flag is set, the front end reloads the KL.

PI LEVEL 0 INTERRUPT FAILURE (DEX)

This is a hardware problem. If it persists, call your Field Service Representative.

POWER-FAIL-RESTART

TKTN prints this message on a reload following a power fail.

PROTOCOL TIME OUT

TKTN prints this message when the protocol being used switches from primary protocol to protocol pause. If primary protocol does not return within 30 seconds, the system crashes and is reloaded.

RE-BOOT REQUESTED

The TOPS-10 or TOPS-20 monitor has requested a reload. The front end reloads the system.

9.4 LOGXFR

LOGXFR is not run immediately after KLERR or KLINIT. The front end must wait until TOPS-10/TOPS-20 has been loaded and is running so that it can transfer the PARSER.LOG file. LOGXFR runs after the SETSPD task in the front end runs. SETSPD runs when the KL has just been reloaded and wants information about line speeds from the front end. (Note that this SETSPD is not the SETSPD that runs on TOPS-20.) The last thing that SETSPD does is to make a request for LOGXFR to run. LOGXFR transfers the log file, if it exists, to TOPS-10/TOPS-20 through the DTE20. TOPS-10 or TOPS-20 then appends the information to ERROR.SYS, the master error file. Finally, PARSER.LOG is deleted from the front-end file area. Any errors detected by LOGXFR will be preceded by

XFR --

and a one-line explanation of the error.

CHAPTER 10

ERROR DEBUGGING

When the front end crashes, the bootstrap ROM passes a dump file to the KL (assuming the KL is running at the time). Much information can be extracted from this file, given the right tools. This chapter explains what data you can get from a dump file using DDT11.

Section 10.1 describes DDT11. Once you are familiar with DDT11, you can proceed to examine locations in the dump file and attempt to determine the cause of the crash. Although much of the data in a dump file is rather obscure, several locations contain data that is almost always useful. This chapter identifies these locations and explains the meaning of the data they contain. The chapter also lists all information in the Front End Status Block.

10.1 USING DDT11

DDT11, which runs on both TOPS-10 and TOPS-20, is a tool for symbolically debugging dumps taken of front-end crashes. The program can also be used to deposit and examine data (with symbols) in the physical front-end memory, using the DTE20 and the Primary Protocol deposit and examine functions. DDT11 can display PDP-11 memory locations as instructions, numbers in a given radix, or bytes of a given size. DTT11 can accept user-defined symbols that are either defined at a terminal or read from a .MAP or .CRF listing. In addition, DDT11 has an initial symbol table of PDP-11 instructions.

DDT11 reads a binary dump file of PDP-11 core. Normally, the PDP-11's bootstrap ROM that sends this file across the DTE20.

TOPS-10 writes the dump file onto its own disk as XPN:DTEDc0.ext, where "c" is the CPU number of the RSX-20F front end being dumped by DTELDR (running with the /AUTO switch) and ext is either BIN or Bnn, where nn is a sequentially assigned number in the range 00 to 99. TOPS-20 writes the dump file onto its own disk as PS:<SYSTEM>0DMP11.BIN. The KL must be running for the front-end to be dumped.

Once TOPS-10 or TOPS-20 has written the dump file, you need not transfer the file to another directory to save it. This is because TOPS-10 increments the file's extension from .BIN to .B99, and TOPS-20's directory PS:<SYSTEM> has an infinite generation-retention count: TOPS-20 never deletes old copies of files in PS:<SYSTEM> without an explicit command to do so.

To use DDT11 to read the dump file, enter the name of the file in response to the DDT11 Input: prompt and use the /DTELDR switch.

The following example demonstrates loading the RSX-20F symbol file and saving a copy of DDT11 that includes the symbols. By using this technique you can later start DDT with the symbols already loaded. Lower-case letters denote information typed by the user.

Example:

```
.r ddt11

DDT11 7E(114)

Input:  RSX-20F.map/fesym          ;name of symbol file
[56p core]                         ;symbols now get loaded
[57p core]
[58p core]
[59p core]
[60p core]
[61p core]
[62p core]
[63p core]
[64p core]
[65p core]
[66p core]
%Loaded 1243 symbols.

Input:  ^Z                         ;back to the monitor

EXIT

.save ve1506                       ;save a copy of DDT11 with
                                   ;symbols using the version
                                   ;of RSX-20F as the file name
.run ve1506                        ;now run the RSX-20F version of
                                   ;DDT11

DDT11 7E(114) = VE1506 /SYMBOLS=DSK:RSX-20F.MAP 11:43 13-July-83

Input:  xpn:dted00.bin/dteldr      ;load dump file
[59p core]
[60p core]
[96p core]
 highest location is 157777
                                   ;user performs task
                      .
                      .
                      .
```

The following is a brief description of some of the commands available in DDT11. See the TOPS-10/TOPS-20 DDT11 Manual for a more detailed description of the commands.

Command	**Effect**
TAB	Causes the current location to be closed. The current location is then set to the current value, and the new location is opened, as with a slash command.
LF	Examines the next location.
CR	Closes the current location.
CTRL-U	Aborts the current expression.

Command	Effect
SPACE	Ends the current expression, and adds it to the current value.
!	Ends the current expression, and sets the current value to the logical OR of the current value and the current expression.
^	Examines the previous location.
*	Ends the current expression, and multiplies it by the current value.
+	Ends the current expression, and adds it to the current value.
-	Ends the current expression, and subtracts it from the current value.
.	Contains the value of the current location.
/	Makes the current location the current value, opens it, and prints the contents.
:	Following an ASCII string of six or fewer characters, defines the ASCII string as a symbol whose value is to be made the current value.
=	Ends the current expression, and types its value as a number in the current radix.
\	Ends the current expression, and examines its value as a symbolic expression. The location counter is not changed.
DELETE	Deletes the most recently typed character.

ALT-MODE Commands

Command	Effect
$A	Temporarily sets address mode.
$nB	Temporarily sets byte mode.
$nC	Temporarily sets constant mode.
$D	Dumps PDP-11 memory to output device. The correct format is [start-address]<[end address]>$D. Asks for the output file specification.
$K	Suppresses the previous typed symbol.
$nR	Temporarily sets radix (values can be 2 through 16).
$S	Temporarily sets output mode to symbolic.
$nT	Temporarily sets output mode to bytes. The byte size is specified by the number that precedes the escape (or Altmode).

Command	Effect
$V	Monitors a location and redisplays it if it changes.
$Y	Asks for a log file and a command file, then executes the command file.
$$A	Permanently sets address mode.
$$nB	Permanently sets byte mode.
$$nC	Permanently sets constant mode.
$$K	Removes the previous symbol from the symbol table.
$$nR	Permanently sets the radix.
$$S	Permanently sets symbolic mode.
$$nT	Permanently sets ASCII mode.

10.2 INTERPRETING AN RSX-20F DUMP

The RSX-20F dump file, <SYSTEM>0DMP11.BIN or XPN:DTED00.BIN, contains useful information for those investigating a front-end crash. You can read either of these files with the symbolic debugger DDT11. By examining various locations in the dump file and comparing them with expected values and with other locations, you can often determine why the front-end crash occurred.

However, crashes occur for innumerable reasons and in many different environments. Thus, it is not possible to give a simple formula that will take the dump file as input and give as output the answer to the question "Why did the front end crash?". You must examine all aspects of the situation, many of them not symbolized in the dump file. For example, some installations have had problems that, when investigated, were found to be caused by poor wiring schema - lines connecting vital pieces of hardware were longer than they should have been, and the noise on the line confused all concerned. This problem is just one of many that could cause RSX-20F to crash without having anything to do with the software itself. It emphasizes the fact that all aspects of the environment must be considered in attempting to determine the root of the problem.

The following points should be kept in mind while you examine crash dumps from RSX-20F.

1. If the problem was severe, random locations in memory may have been erased or overwritten.

2. Not all situations that are seen by humans as problems result in RSX-20F crashing. Therefore, RSX-20F may not produce the dump you need to determine the problem.

3. Because PDP-11 stacks use autodecrement mode, the stack grows toward lower core, not higher.

4. The PDP-11 low-order byte is the right-hand byte, not the left-hand one.

5. Many times hardware problems masquerade as software problems. Something that seems to have been caused by software is often found to be a subtle manifestation of a hardware difficulty.

10.2.1 Useful Data in Dump Files

Although each crash is different from another, if only because the environment is different, there are some data that you will always wish to have before attempting to fix blame for the crash. This includes such data as the crash code, the task that was running at the time of the crash, and the last instruction to execute before the crash. This section explains how to obtain this useful data.

When you examine a dump file, you should first find out what the crash code was. The crash code is a three-letter code that identifies the type of error RSX-20F detected when it crashed. The code is always at locations 0 and 2 in the dump file. If no readable code is in these locations, RSX-20F did not have control over the crash; the PDP-11 may have been halted with the HALT switch, or there may have been a Keep-Alive-Cease error, for example.

At this point you may wish to make sure that the version of RSX-20F you are using is consistent with the version of the symbol file you loaded into DDT11. You can verify this by checking the locations .VERNO, .VERNO+2, .VERNO+4, and .VERNO+6. These locations contain data of the form Vxyy-zz, where x is either A for 1080/1090s, E for 1091/1095s, or B for TOPS-20, and yy and zz are the version and edit numbers, respectively.

You can find out which task was running at the time of the crash by examining the location .CRTSK. This location points to the ATL node of the current task. When you have opened .CRTSK, you can use the <TAB> command in DDT11 to open the location to which .CRTSK points. The response from DDT11 includes a symbolic address that is the symbol used internally by RSX-20F to name a task.

You can determine the last instruction to execute before the crash by examining the location SPSAV. This location contains a copy of the stack pointer at the time of the crash. Since the PS (Processor Status word) and PC (Program Counter) are stored on the stack at the time of a crash, you can find out to which instruction the PC pointed. (Of course, the last instruction to execute would be the one previous to that pointed to by the stacked PC.) Once you have opened SPSAV, you can use the <TAB> command to open the address pointed to by SPSAV. This address will be the top word in the stack. If SPSAV is zero, the crash was caused by a Keep-Alive-Cease. If it is not zero, and the crash code is not a T04, FTA, RES, BPT, or DTD, the top three words on the stack will be R5, the PC, and the PS, in that order. If the crash code is a T04, FTA, RES, BPT or DTD, and SPSAV is not zero, the fourth, fifth, and sixth words will be R5, the PC, and the PS, respectively. Subtracting two from the address contained in the PC gives you the address of the last instruction to execute. You can also find other PS's and PC's further down the stack that were saved earlier. This data can help you determine the environment prior to the crash; you must be careful, however, in using the data, because random data can sometimes appear similar to a saved copy of the PS and PC.

You may wish to examine the PDP-11 registers and the DTE20 registers. The PDP-11 registers are stored in locations 40 through 56 (R0 at 40, R7 at 56). Since R6 (at location 54) is the hardware stack pointer, it points to the top of the stack at all times. Note that R7 almost always contains the same address, because it is pointing to ROM code. The DTE20 registers are stored in locations 130 through 156.

One of the most frequent reasons for RSX-20F crashing is the lack of sufficient buffer space. This causes crashes of the BO3 type. RSX-20F uses three areas for storage: the Free Pool, the Big Buffer, and the Node Pool. The Free Pool runs out of space the fastest because it is used the most frequently (it holds TTY thread lists and LPT thread lists). When looking at any of these areas, the questions you should try to answer are:

1. How much space is left in the buffer?
2. How fragmented is the space that is left?
3. Are all the pointers pointing to the correct places?
4. Is the count of free space an accurate representation of the state of the buffer?

The initial pointers to each of these areas follows.

Free Pool — .FREPL is the pointer to the first free chunk of storage space.
.FREPL+2 is the tally of the free space remaining in this area.

Big Buffer — .BGBUF is the pointer to the first free chunk of storage space.
.BGBUF+2 is the tally of the free space remaining in this area.

Node Pool — .POLLH is the pointer to the first free node in the doubly-linked queue.
.POLLH+2 is the pointer to the last node in the queue.

The queues, both TO-10 and TO-11, can yield some hints on the cause of the crash, especially in cases such as buffer overflows where the queues may have used up all the buffer space. The TO-10 queue pointer (TO10Q) points to itself when the TO-10 queue is empty, whereas the TO-11 queue pointer (TO11Q) contains zero if the queue is empty. Thus, in order to examine the entire TO-10 queue, you open the location TO10Q and use the <TAB> command until the contents of the location you open is TO10Q. To examine the entire TO-11 queue, you open the location TO11Q and use the <TAB> command until the contents of the location you open is zero.

10.2.2 Sample Dump Analysis

In the following dump analysis, a sample RSX-20F dump is examined to determine what caused the crash. The dump has been produced by aggravating a known RSX-20F weakness, which is the lack of free space. A privileged program doing repeated Send-alls can easily fill up all the available space and cause buffer overflows, because the Send-all message must be put into the thread list of every active terminal.

The sample analysis below assumes that a copy of DDT11 (called VA1341) has been saved with the symbol file already loaded into it. The first thing to do after starting the DDT11 program is to determine the crash code.

```
@run va1341

DDT11 7E(114) = VA1341 /SYMBOLS=DSK:RSX-20F.MAP 11:43 13-July-83

Input:  xpn:dted00.bin/dteldr
[59p core]
[60p core]
[96p core]
 highest location is 157777

$$3T   0/   B02
```

The stopcode is BO2, buffer overflow, which is produced by the DTE20 device driver when it cannot find Free Pool space for a TO-11 indirect transfer. Since we know this much about the cause of the crash, there is no reason to find out the task that was running or the last instruction executed. Therefore, the next step in the dump analysis is to examine the Free Pool.

```
$$A
.FREPL/ 65664
.FREPL+2/       QI.VER  =300
.FREPL/ 65664
65664/  70674
70674/  72234
72234/  72474
72474/  73034
73034/  75074
75074/  0
65664/  70674
65666/  PATSIZ  =40
70676/  PATSIZ  =40
72236/  PATSIZ  =40
72476/  PATSIZ  =40
73036/  PATSIZ  =40
75076/  PATSIZ  =40
```

Thus we can see that the Free Pool has only 300 remaining bytes in six sections, each section being forty bytes long (or 20 words). Indirect transfers need more contiguous space than is available in the Free Pool. Since we now know that the Free Pool has run out of space while the system was attempting to do an indirect transfer, we can surmise that the indirect transfer may well have been a Send-all (especially since we know that Send-alls can create problems for RSX-20F). Thus, we proceed to examine the Send-all buffers. The location .SNDLP points to the Send-all buffer in use, .SNDBF is the Send-all ring buffer pointer, .SNDCN is the terminal count for a pending Send-all, and .CRSND is the pointer to the current Send-all node.

```
.SNDLP/ DR.03   =3
.SNDBF/ 66574   .SNDBF+2/       66774
.SNDBF+4/       67074   .SNDBF+6/       0
.SNDCN/ DR.03   =3      .SNDCN+2/       DR.03   =3
.SNDCN+4/       DR.03   =3      .SNDCN+6/       0
.CRSND/ 0
```

As we can tell from the three words of nonzero data at location .SNDBF, there are three ring buffers in use, which is the maximum. .SNDCN (and the following locations) tell us that three terminals have yet to empty the buffers. The queued protocol task is therefore unable to accept further Send-all messages from the KL. Thus, a logical next step would be to check the state of the TO-11 queue.

```
TO11Q/  67274
67274/  67374
67374/  67474
67474/  67574
67574/  67674
67674/  67774
67774/  70074
70074/  70174
70174/  70274
70274/  70374
70374/  70474
70474/  70734
70734/  66274
66274/  70574
70574/  71034
71034/  71134
71134/  71234
71234/  71334
71334/  66474
66474/  71434
71434/  71534
71534/  71634
71634/  71734
71734/  72034
72034/  66674
66674/  72274
72274/  72134
72134/  72534
72534/  72634
72634/  65464
65464/  73074
73074/  72734
72734/  73174
73174/  73274
73274/  72374
72374/  73374
73374/  73474
73474/  73574
73574/  73674
73674/  73774
73774/  74074
74074/  74174
74174/  74274
74274/  74374
74374/  74474
74474/  74574
74574/  65564
65564/  74674
74674/  66374
66374/  75134
75134/  75234
75234/  75474
75474/  75574
75574/  74774
74774/  75674
75674/  0
TO10Q/  TO10Q
```

The TO-11 queue is quite full (since all the Free Pool space is being used for this queue). The TO-10 queue is empty.

We know that the Send-all service waits for a significant event when it has filled the ring buffer. While the Send-all service waited, the Free Pool ran out of space. However, the line printer and terminal thread lists may be contributing to the problem if they are also taking space from the Free Pool. Therefore, it would probably be a good idea to check the state of these thread lists.

```
LPTBL/  AF.PP   =200
LPTBL+2/        175400
LPTBL+4/        0
DHTBL/  0
DHTBL+2/        160020
DHTBL+4/        TTYSP+161
DHTBL+6/        TT.SND
DHTBL+10/       0
DHTBL+12/       160020
DHTBL+14/       TTYSP+161
DHTBL+16/       TT.SND
DHTBL+20/       0
DHTBL+22/       160020
DHTBL+24/       TTYSP+161
DHTBL+26/       TT.SND
DHTBL+30/       0
DHTBL+32/       160020
DHTBL+34/       TTYSP+161
DHTBL+36/       TT.SND
DHTBL+40/       0
DHTBL+42/       160020
DHTBL+44/       TTYSP+161
DHTBL+46/       TT.SND
DHTBL+50/       0
DHTBL+52/       160020
DHTBL+54/       TTYSP+161
DHTBL+56/       TT.SND
DHTBL+60/       0
DHTBL+62/       160020
DHTBL+64/       TTYSP+161
DHTBL+66/       TT.SND
DHTBL+70/       0
DHTBL+72/       160020
DHTBL+74/       TTYSP+161
DHTBL+76/       TT.SND
        Etc.
         .
         .
         .
```

All the terminals and line printers have empty thread lists. Thus, the Send-all service alone must be the cause of the crash. The final step is to read the actual Send-all messages.

```
.SNDBF/ 66574
66574/  0
66576/  CH.FOR  =100
66600/  66606   $$6T    ./      <206>M3<0><377>3
66606/  THIS I
66614/  S A DA
66622/  TA COL
66630/  LECTIO
66636/  N TEST
66644/  .  PLE
```

```
66652/  ASE BE
66660/   PATIE
66666/  NT.<0><17><0>    66774/  <0><0>@<0><6>N
67002/  3<0><377>3TH
67010/  IS IS
67016/  A DATA
67024/   COLLE
67032/  CTION
67040/  TEST.
67046/   PLEAS
67054/  E BE P
67062/  ATIENT
67070/  .<0>D<15>\N
67076/  @<0>FN3<0>       67074/  \N@<0>FN
67102/  3<0><377>3TH
67110/  IS IS
67116/  A DATA
67124/   COLLE
67132/  CTION
67140/  TEST.
67146/   PLEAS
67154/  E BE P
67162/  ATIENT
```

The message that caused the crash when sent to all users was "This is a data collection test. Please be patient".

10.2.3 Front End Status Block

The Front End Status Block contains all the data and status information used by the Executive while operating. Thus, the block brings together most of the information you need to determine what the data in a crash dump file means. The following lists all information contained in the Front End Status Block.

FE STATUS BLOCK
RSX-20F VERSION 15-06

COMMON GLOBAL DATA

ADDRESS	SIZE	NAME	USE
001000	2	.FESTB	length of FE status block in words
001002	4	.EXEND	limits of front-end word 1- base address of executive word 2- high address of executive
001006	2	.CRTSK	pointer to ATL node of current task
001010	4	.COMEF	global common event flags (FLAGS 33-64) word 2- flags 49 to 64 bit 9- (EF.FCP)fault continuation in progress bit 10- (EF.CRI)comm region is invalid bit 11- (EF.PFR)powerfail restart in progress bit 12- (EF.RKP)KLINIK parameters received bit 13- (EF.PR2)secondary protocol running bit 14- (EF.CTC)control C bit bit 15- (EF.PR1)primary protocol running
001014	2	.SERFG	Significant event flag bit 0- (EV.SE)sig event to be recognized bit 1- (EV.AS)power fail is required bit 7- (EV.PF)power down has occured
001016	2	.SEWFL	Significant event wait flag bit 0- (EV.SE)sig event to be recognized bit 1- (EV.AS)power fail is required bit 7- (EV.PF)power down has occured
001020	2	SPSAV	Save area for stack pointer during crash
001022	4	PARSAV	Save area for parity registers when parity error
001026	2	.PFAIL	Indicates power fail in progress if non-zero
001030	2	.PFIOW	Power fail recovery flag, set during power up
001032	2	PWRXSP	Buffer for stack pointer during power up
001034	2	CROBAR	Power-up crobar timer, power up attempts=6
001036	12	.VERNO	RSX-20F ASCII version number
001050	2	.CKASS	Clock AST address for current task
001052	2	.PFASS	Power fail AST address for current task
001054	40		
001114	2	.MSIZE	Memory size in 64 byte blocks (1600)

ADDRESS	SIZE	NAME	USE
001116	2	EMTSTK	SP saved during EMT execution
001120	2	TRPASV	Saved PS during EMT/trap execution
001122	1	.NOERR	Don't recognize KL errors if non-zero
001123	1	.NOHLT	Don't recognize KL halts if non-zero
001124	2	.TKTN	TKTN required if non-zero, checked by null task
001126	2	.KLITK	KLI requested, read by TKTN bit 0- (KS.TSP)ten halted bit 1- (KS.CES)clock error stop bit 2- (KS.EPE)E box parity error bit 3- (KS.DEX)deposit/examine error bit 4- (KS.CST)keep alive stopped bit 5- (KS.TRR)ten request's re-boot bit 6- (KS.PFT)power fail restart bit 7- (KS.PTO)protocol timeout
001130	2	.KLERQ	KL crash snapshot flag for PARSER 1=take snapshot 0=no snapshot
001132	2	.KLIWD	KLI word to determine boot parameters bit 0- (KL.LRM)load RAMS bit 1- (KL.CFM)configure memory bit 2- (KL.LVB)load VBOOT bit 3- (KL.VBN)VBOOT start at START+1 bit 4- (KL.VBD)dump monitor bit 5- (KL.SPF)start at loc 70 (power fail) bit 6- (KL.LCA)load cache bit 7- (KL.SSC)start at loc 407 (system crash) bit 8- (KL.CFL)if 0, configure from file bit 9- (KL.KAC)keep alive ceased error bit 10- (KL.DEF)operator reboot from switches bit 11- (KL.REQ)KLINIT requested bit 12- (KL.ABO)KLINIT canceled
001134	2	.TICKS	Unrecognized clock tick counter
001136	2	.CLKSW	Clock overflow switch
001140	2	.DATE	Front-End date valid flag
001142	2	.YEAR	Decimal year (1979)
001144	1	.DAY	Day of month
001145	1	.MON	Month of year
001146	1	.DST	Daylight savings time flag
001147	1	.DOW	Day of week index

FE Status Block
COMMON GLOBAL DATA

ADDRESS	SIZE	NAME	USE
001150	4	.SSM	Elapsed time in seconds since midnight
001154	2	.TKPS	Clock rate in jiffies
001156	2	.SYUIC	System UIC ([5,5])
001160	2	.BTPRM	Boot parameter from switch register bit 0- (BP.SWR)switch register button pushed bit 1-2- (BP.LD0,BP.LD1)boot options 0=auto deadstart entire system 1=deadstart RSX-20F only 2=reboot RSX-20F only 3=operator controlled start bit 3-6- (BP.CSP)CTY speed if DH,1=DL,0=default bit 7- (BP.RP4)load from RP04/rp06 bit 8-10- (BP.UNT)boot unit or DH unit bit 11-14- (BP.CLN)CTY line no.(within DH/DL) bit 15- (BP.ERR)indefinite error load retry
001162	2	.BTSCH	Character saved for secondary protocol
001164	2	.ACKAL	Send acknowledge-all protocol message
001166	2	.KLERW	KLI word for error reporting by SETSPD
001170	2	.FEMOD	Front-End console mode flag for PARSER 1= (LG.OPR)operator 3= (LG.PRM)programmer 7= (LG.ALL)maintenance
001172	1	.KLRLD	KL automatic reload flag
001173	1	.KLFCF	KL Fault Continuation Flag
001174	2	.KLFLG	Flag for PARSER to indicate state of KL bit 6- (KF.CES)clock error stop bit 7- (KF.CON)KL continuable bit 8- (KF.KLO)instruction mode bit 9- (KF.BRM)burst mode bit 10- (KF.SPM)single pulse EBOX mode bit 11- (KF.SMC)single pulse MBOX mode bit 12- (KF.SIM)single instruction mode bit 13- (KF.MRS)master reset flop set bit 14- (KF.RUN)run flop on bit 15- (KF.CLK)clock running
001176	1	.KLCPU	CPU Number +1 for warm restart 0= KL cannot do warm restart
001177	1	.KLMON	Code for monitor that is running (not currently used) 1=TOPS-10 2=ITS 3=TENEX 4=TOPS-20
001200	4	.KLMF2	Retry count for MF20 controllers (default is 1)

KLINIK DATA BASE

ADDRESS	SIZE	NAME	USE
001204	2	.KLNPB	KLINIK parameter block length (26)
001206	2	.KLNBC	Byte count for transfer (24)
001210	2	.KLNFT	KLINIK enable start time
001212	4	.KLNFD	KLINIK enable start date
001216	2	.KLNTT	KLINIK enable end time
001220	4	.KLNTD	KLINIK enable end date
001224	2	.KLNMD	KLINIK console mode byte 0- console mode 1= (LG.OPR)operator 3= (LG.PRM)programmer 7= (LG.ALL)maintenance byte 1- status 0=disabled 1=remote -1=user
001226	6	.KLNPW	ASCII password for KLINIK
001234	2	.KLNSW	KLINIK line status byte 0- line use 0=disabled 1=remote -1=user byte 1- current status 1=clear KLINIK, recall PARSER 2=report carrier loss 3=disconnect, recall PARSER 4=disconnect and exit

COMMUNICATIONS REGION DATA BASE

ADDRESS	SIZE	NAME	USE
001236	2	COMBSE	Base of communication area
001240	2	PRMEMN	My processor number
001242	2	DEPOF	Deposit offset from examine
001244	12	PROTBL	Processor identification table word 1- (DTENM)DTE addr to access this proc word 2- (EMYN)addr to read from proc 0 word 3- (DMYN)addr to write to proc 0 word 4- (EHSG)addr from general word 5- (EHSM)addr from specific

QUEUED PROTOCOL DATA BASE

ADDRESS	SIZE	NAME	USE
001256	2	.CRQZ	Size of current TO-10 buffer
001260	2	.CPFN	Function in current TO-10 buffer
001262	2	.CPDV	Device in current TO-10 buffer
001264	2	.CRSZ	Size left in current TO-10 buffer
001266	2	.CRPB	Pointer to open word in current TO-10 buffer
001270	2	.CRHD	Head of current TO-10 queue
001272	2	.CRSB	Pointer to current function/size in TO-10 buffer
001274	2	DTEMSK	DTE device event flag mask
001276	2	DTEADR	DTE device indirect flag address
001300	2	TO11NP	Pointer to current received node
001302	2	TO11HD	Count of bytes in this queue
001304	2	TO11FN	Current received TO-11 function code
001306	2	TO11DV	Current received TO-11 device number
001310	2	TO11SP	Space
001312	2	TO11FW	First word of function
001314	2	TO11GW	Guard word for DTE20 (-1)
001316	2	TO11AS	Address save
001320	2	TO11BS	Byte count of TO-11 transfer saved
001322	2	TO10SZ	Byte count of transfer
001324	2	TO10AS	TO-10 transfer address saved
001326	6	STSTT	TO-10 status
001334	4	TO10Q	Listhead for TO-10 queue

QUEUED PROTOCOL DATA BASE

ADDRESS	SIZE	NAME	USE
001340	2	EQSZ	TO-11 queue size
001342	2	TO11Q	Head of TO-11 queue
001344	6	STATI	Status/scratch word for examine/deposit
001352	2	DEXST	DEX done timeout
001354	2	DEXTM3	Deposit/examine word 3 for retry
001356	2	DEXTM2	Deposit/examine word 2 for retry
001360	2	DEXTM1	Deposit/examine word 1 for retry
001362	2	.PRADR	Address of privileged offset table entry
001364	2	.PRSTA	Address of privileged DTE20 status (174434)
001366	2	.PRDTE	Address of privileged DTE20 (174400)
001370	2	.PRDCT	Doorbell counter for KLINIT
001372	1	.DXRTY	DEX error processing flag 1=no error 0=retry succeeded -1=retry failed
001373	1	.EBRTY	EBUS parity error processing flag 1=no error 0=retry succeeded -1=retry failed
001374	2	.EBPEQ	Pointer to EBUS/DEX error snapshot queue
001376	2	.EBPEC	Count of nodes in EBUS/DEX error queue
001400	1	.PRPSE	Protocol pause flag -1=pause state 0=no pause
001401	1	TOXQIP	TO-10 queue in progress
001402	2	.DTBLK	Holds re-entry point to start blocked DTE transfer

KEEP ALIVE DATA BASE

ADDRESS	SIZE	NAME	USE
001404	6	KPAL0	Current KL10 keep alive value
001412	2	OKPAL0	KL10 saved keep alive value
001414	2	KPAL1	Current RSX-20F keep alive value
001416	2	.KPAC	Counter of keep alive for XCT 71 byte 0- keep alive counter byte 1- XCT 71 counter
001420	2	.KACFL	XCT 71 retry flag

CORE MANAGER DATA BASE

ADDRESS	SIZE	NAME	USE
001422	4	.BGBUF	Big buffer space word 1- pointer to first node in free space word 2- current total size of space
001426	4	.FREPL	Free pool list word 1- address of first node in free pool word 2- current total size of free pool
001432	4	.POLLH	Pool header for ATL and send nodes word 1- pointer to start of list word 2- pointer to end of list
001436	700	.POLST	Pool list (14 entries) 16 word entries word 1- pointer to next block word 2- pointer to previous block
002336	40	.POLND	Pool end 16 word entry word 1- pointer to next block word 2- pointer to previous block

CLOCK REQUEST LIST

ADDRESS	SIZE	NAME	USE
002376	204	.CLKBA	Clock list 6 word entries word 1- (C.AT)ATL node address of requestor word 2- (C.AS)AST trap address of requestor word 3- (C.SD)schedule delta in ticks word 4- (C.RS)reschedule delta in ticks word 5- (C.FM)flag mask word 6- (C.FA)flags word address
002602	2	.CLKEA	End of clock list guard word

TERMINAL SERVICE DATA BASE

ADDRESS	SIZE	NAME	USE
002604	2	.INHDM	Inhibit/enable remote lines (0=enable)
002606	2	.ABCNT	Count of auto-bauded lines
002610	2	.ABFLG	Interlock flag for SETSPD
002612	2	.SNDLP	Pointer to send-all buffer in use
002614	10	.SNDBF	Send-all ring buffer pointer
002624	10	.SNDCN	Send-all TTY count for send-all pending
002634	2	.CRSND	Current send-all node pointer
002636	2	.BRKCH	Break character (control \)
002640	2	.TTP11	TTY PDP11 input in progress flag
002642	2	.CTYPT	CTY line pointer
002644	2	.KLNPT	KLINIK line pointer
002646	2	$UNIT	DH unit number if CTY
002650	2	$BTMSK	Mask to start CTY if DH
002652	2	DMTMP	Saved DM11/BB controller number
002654	2	DHTMP	Saved DH11 controller number
002656	2	DLTMP	Saved DL11 controller number
002660	2	DHSTSV	Saved DH11 table pointer from PS
002662	2	.TTELQ	Terminal error logging queue ptr
002664	2	.TTELC	Count of nodes in error logging queue
002666	2	.TTELB	Temp buffer ptr for error logging
002670	2	TMOCNT	Timeout counter byte 0- terminal(10) byte 1- modem(22)

PDP-11 CTY SERVICE DATA BASE

ADDRESS	SIZE	NAME	USE
002672	40	CTYSTS	CTY status block word 1- (STATS)status word bit 0- (FLBT)unprocessed fill count bit bit 0-3- (FLCT)unprocessed fill count field bit 4- (RUBP)rubout sequence in progress bit 5- (CTLO)output disabled bit 8- (EOLS)end of line seen bit 9- (CRJT)CR typed just typed bit 10- (CRTY)carriage control at EOL bit 11- (LFBT)unprocessed LF add/sub bit bit 11-14- (LFCT)unprocessed LF count field bit 15- (MODE)terminal busy(1=output,0=input) word 2- (STRBF)current input buffer address word 3- byte 0- (RMBYT)remaining bytes in buffer byte 1- (FNBYT)terminal byte word 4- (CURBF)starting buffer address word 5- byte 0- (MECNT)multi-echo byte count byte 1- (FLBYT)fill byte word 6- (MEBUF)multi-echo buffer address word 7- (MBUFR)dynamic multi-echo buffer word 8- (HORPS)horizontal position of carriage word 9- (DHBUF)DH character buffer if CTY a DH
002732	2	CNT	I/O packet size
002734	2	BYCNT	I/O packet size
002736	2	CRADR	Current I/O address
002740	2	TTPKT	I/O packet address
002742	40	DMTBL	DM11/BB table 2 word entries (8 entries) word 1- DM11/BB base address word 2- pointer to DH11 table entry
003002	2	DMTBE	End of table marker(0)

DATA LINE SCANNER DATA BASE

ADDRESS	SIZE	NAME	USE
003004	10	DLTBL	DL11/C table 4 word entry for each DL11/C line (1 entry) word 1- (THRED)output thread word pointer word 2- (TTYEXP)device base address word 3- (STSW0)status word 0 DL11- input flag DH11- line speed bit 6-9- (S0.ISP)input speed bit 10-13- (S0.OSP)output speed bit 14- (S0.CON)remote line connected bit 15- (S0.ABR)autobaud report pending word 4- (STSW1)status word 1 bit 0- (TT.OUT)TTY output flag bit 1- (TT.CTY)console CTY bit 2- (TT.CRW)waiting for carrier bit 3- (TT.ABW)auto-baud wait bit 4- (TT.XEN)XON/XOFF enabled bit 5- (TT.ABL)auto-baud line bit 6- (TT.RMT)remote line bit 7- (TT.XOF)line is XOFF'D bit 8- (TT.NSA)suppress send-alls bit 9- (TT.SIP)send-all in progress bit 10- (TT.RIP)remote in progress bit 11-12- (TT.FEC)framing error count bit 13- (TT.RSI)restart tty on timeout bit 14- (TT.SNI)increment send-all index bit 14-15- (TT.SND)index of next send-all
003014	40	DLETBL	DL11/E table 4 word entry for each DL11/E line (4 entries) see above

DH11 DATA BASE

ADDRESS	SIZE	NAME	USE
003054	2000	DHTBL	DH11 table 4 word entry for each DH11 line (128 entries) see above
005054	2	.TTS2F	Flag for clock service
005056	2	.S2IDC	Current count of locally disabled lines
005060	2	.S2ITP	Table position of last locally enabled line
005062	2	.IBFLO	Input buffer low threshold
005064	2	.IBFOK	Input buffer ok threshold
005066	412	STSW2	Terminal input control table (133 entries) bit 0-7- (S2.CHR)deferred write char bit 8-9- (S2.CNT)wait count field bit 9- (S2.SSZ)set input speed to zero bit 10- (S2.ACK)owe this line an ack bit 11- (S2.LCL)input XOFF'd,local request bit 12- (S2.ENB)deferred input XON request bit 13- (S2.DIS)deferred input XOFF request bit 14- (S2.DIP)input XOFF in progress bit 15- (S2.DDN)input is XOFF'd

FLOPPY DRIVER DATA BASE

(2040S, 2060, 2065, 1091, 1095 ONLY)

ADDRESS	SIZE	NAME	USE
005500	2	DXRTC	Error retry count (8)
005502	2	DXCNT	Byte count of transfer
005504	2	DXBUF	Address of buffer
005506	14	DXVCB	word 1- logical or physical sector number word 2- bytes to transfer on current sector word 3- current function code word 4- physical sector number(1-26.) word 5- physical track number(0-77.) word 6- status register after interrupt
005522	2	DXUNIT	Current unit number
005524	2	DXPKT	I/O packet address

DECTAPE DRIVER DATA BASE

(1090 ONLY)

ADDRESS	SIZE	NAME	USE
005500	2	DTRTC	Error retry count and reset flag
005502	2	DTRNA	Request node address
005504	4	DTBUF	DECtape buffer
005510	2	DTCNT	Buffer size
005512	2	DTCW2	
005514	2	DTCW3	
005516	10		Pad to floppy driver size

DISK DRIVER DATA BASE

ADDRESS	SIZE	NAME	USE
005526	2	RPRTC	Error retry count (8)
005530	2	RPRNA	Address of request node
005532	4	RPBUF	Address of buffer word 1- high order of address word 2- low order of address
005536	2	RPCNT	Transfer size
005540	2	RPUNIT	Current unit number
005542	2	RPCW2	
005544	2	.RPELQ	RH11 error logging queue
005546	2	.RPELC	Count of nodes in RH11 error queue
005550	2	.RHPB	Serial numbers packet size
005552	20	.RHSN	RP04/RP06 serial numbers

FE DRIVER DATA BASE

ADDRESS	SIZE	NAME	USE
005572	10	FETBL	Table of FE device states 1 word per FE device (4) bit 10- (FE.DET)more data 11 to 10 to be sent bit 11- (FE.DTE)more data 10 to 11 expected bit 13- (FE.SER)servicing 11 transfer request bit 14- (FE.STR)servicing 10 transfer request
005602	2	NODADR	Current request node address
005604	2	ADRSAV	Address saved
005606	2	BYTESA	Byte count of transfer
005610	4	STSWD	I/O status words
005614	22	TO10PK	11 request to 10 packet address
005636	2	DNBLK	Response to 10 request
005640	6	DNFCN	Function
005646	6	DNSTS	Status
005654	40	BLKTT	Data buffer
005714	2	.RPUNT	RP unit number
005716	2	.FEACT	FE device available for DB access
005720	4	.RPADR	
005724	4	.RPSIZ	

FE Status Block

CD-11 DRIVER DATA BASE

ADDRESS	SIZE	NAME	USE
005730	2	CREVFG	Address of CR task's event flags
005732	2	CRCEVF	Current event flags
005734	2	CRHUNG	Count of times CR found hung
005736	2	.CRPFL	Power fail flag for card reader
005740	2	CRSTBH	Header word of status block
005742	16	CRSTBK	Status return block to 10 word 1- 1st status word bit 0- (DV.NXD)non-existent device bit 1- (DV.OFL)device off-line bit 2- (DV.OIR)hardware error, opr required bit 3- (DV.SCN)software error, ACK required bit 4- (DV.IOP)I/O in progress bit 5- (DV.EOF)end of file encountered bit 6- (DV.LOG)error logging required bit 7- (DV.URE)un-recoverable error bit 8- (DV.F11)error on from 11 request bit 9- (DV.HNG)device hung word 2- 2nd status word, device dependent bit 0- (DD.RCK)read check bit 1- (DD.PCK)pick check bit 2- (DD.SCK)stack check bit 3- (DD.HEM)hopper empty bit 4- (DD.SFL)stacker full word 3- control and status register word 4- column count register word 5- bus address register word 6- data buffer register
005760	2	CRBUFH	Header word of data buffer
005762	240	CRBUFF	Data buffer from CD11
006222	2		Data buffer overrun area
006224	2	CRTHD	Threaded list pointer
006226	2	CREXP	Device external page address
006230	2	CRSTS	Status bits bit 8- (CR.NSF)not stacker full bit 9- (CR.NXD)non-existent CD-11 bit 10- (CR.RHN)reader hung during read bit 11- (CR.ACK)acknowledge received bit 12- (CR.IOD)I/O done bit 13- (CR.IOP)I/O in progress bit 14- (CR.SST)device status changed bit 15- (CR.HNG)CR hung
006232	2		Unused

FE Status Block

LP-2Ø DRIVER DATA BASE

ADDRESS	SIZE	NAME	USE
ØØ6234	2	LPUNIT	LP unit number from PS on interrupt
ØØ6236	2	LPEVFG	Address of where to set event flags for LP task
ØØ624Ø	2	LPCEVF	Current event flags
ØØ6242	2	LPHUNG	Count of times LP was hung
ØØ6244	2	.LPPFL	Power fail flag
ØØ6246	2	LPSTBH	Header word of status block
ØØ625Ø	3Ø	LPSTBK	Status return block to 1Ø word 1- 1st status word bit Ø- (DV.NXD)non-existent device bit 1- (DV.OFL)device off-line bit 2- (DV.OIR)hardware error, opr required bit 3- (DV.SCN)software error, ACK required bit 4- (DV.IOP)I/O in progress bit 5- (DV.EOF)end of file encountered bit 6- (DV.LOG)error logging required bit 7- (DV.URE)un-recoverable error bit 8- (DV.F11)error on from 11 request bit 9- (DV.HNG)device hung word 2- 2nd status word, device dependent bit Ø- (DD.PGZ)page counter passed zero bit 1- (DD.CHI)character interrupt from RAM bit 2- (DD.VFE)VFU error bit 3- (DD.LER)error with VF/RAM file bit 4- (DD.OVF)printer has optical VFU bit 5- (DD.RME)RAM parity error word 3- byte Ø- no. bytes device dependent info (2.) byte 1- no. bytes device registers (16.) word 4- byte Ø- accumulated checksum byte 1- retry count word 5- control and status register A word 6- control and status register B word 7- bus address register word 8- byte count register(2's complement) word 9- page counter register word 1Ø- RAM data register word 11- byte Ø- character buffer register byte 1- column count register word 12- byte Ø- printer data register byte 1- checksum register

FE Status Block
LP-20 Driver Data Base

ADDRESS	SIZE	NAME	USE
006300	20	LPTBL	LP first device table 4 word entry for unit 0 word 1- (LPSTS) status bits bit 0-1- (LP.UNT) unit number bit 7- (LP.EOF) end of file encountered bit 8- (LP.F10) from 10 request queued bit 9- (LP.LIP) load VFU in progress bit 10- (LP.CLR) clear RAM required bit 11- (LP.WAT) LP waiting for response bit 12- (LP.MCH) multi-char printing bit 13- (LP.PZI) page zero interrupt enabled bit 14- (LP.SST) send status to 10 bit 15- (LP.HNG) device hung word 2- (LPCSA) external page address word 3- (LPTHD) thread list pointer word 4- (LPITH) current buffer pointer 4 word entry for unit 1
006320	20	LPTBL2	LP second device table 4 word entry for unit 0 word 1- (LPMCB) multi-character buffer word 2- (LPCSM) accumulated checksum word 3- (LPRTY) retry counter word 4- 4 word entry for unit 1
006340	20	LPTBL3	LP third device table 4 word entry for unit 0 word 1- (LPRMA) VFU data address word 2- (LPRMZ) VFU data buffer size word 3- (LPRMC) current ptr into VFU data word 4- 4 word entry for unit 1
006360	4	LPUTBL	Unit table pointer word 1- unit 0 pointer in LPTBL word 2- unit 1 pointer in LPTBL

SYSTEM TASK DIRECTORY

ADDRESS	SIZE	NAME	USE
006364	2	.STDTA	Pointer to STD list
006366	2	.STDTC	Maximum STD list size (18 entries)
006370	2	.STDTZ	Current size of STD list
006372	44	.STDTB	STD table 18 pointers to task's STD entries word 1- card reader driver word 2- DTE driver word 3- FE driver word 4- floppy (2040S,2060,2065,1091,1095) DECtape (1090) word 5- F11ACP task word 6- line printer driver word 7- queued protocol task word 8- disk driver word 9- terminal driver word 10- install task
006436	40	STDDTE	DTE driver STD entry 16 word task STD entry word 1- (S.TN)task name (1st 3 chars) word 2- task name (2nd 3 chars) word 3- (S.TD)default task partition word 4- (S.FW)flags word bit 0- (SF.TA)task active bit 1- (SF.FX)task fixed bit 2- (SF.EX)task to be removed bit 14- (SF.IR)install requested bit 15- (SF.ST)system task word 5- byte 0- (S.DP)default priority byte 1- (S.DI)system disk indicator word 6- (S.BA)1/64th of base address word 7- (S.LZ)size of load image word 8- (S.TZ)max task size word 9- (S.PC)initial PC word 10- (S.SP)initial SP word 11- (S.RF)send/req queue forward ptr word 12- (S.RB)send/req queue backward ptr word 13- (S.SS)SST vector table address word 14- (S.DL)load image low disk address word 15- load image high disk address word 16- zero see above

FE Status Block
System Task Directory

ADDRESS	SIZE	NAME	USE
006476	40	STDFED	FE driver STD entry see above
006536	40	STDDX	Floppy driver STD entry (2040S,2060,2065,1091,1095)
006536	40	STDDTP	DECtape driver STD entry (1090) see above
006576	40	STDF11	F11ACP STD entry see above
006636	40	STDRPT	RP device STD entry see above
006676	40	STDINS	Install STD entry see above
006736	40	STDLPT	LP driver STD entry see above
006776	40	STDCDR	CR driver STD entry see above
007036	40	STDTTY	TTY driver STD entry see above
007076	40	STDQPR	Queued protocol STD entry

ACTIVE TASK LIST

ADDRESS	SIZE	NAME	USE
007136	4	.ATLLH	ATL header word 1- forward pointer (DTE) word 2- backward pointer (null task)
007142	40	DTETSK	DTE task ATL entry 16 word ATL entry word 1- forward linkage word 2- backward linkage word 3- (A.SP)SP of running task word 4- (A.PD)run partition word 5- (A.RP)run priority word 6- (A.HA)1/64th of base address word 7- byte 0- (A.TS)task status 2= (TS.LRQ)load request queued 4= (TS.TKN)waiting for TKTN 6= (TS.LRF)load request failed 10= (TS.RUN)task running 12= (TS.SUS)task suspended 14= (TS.WF0)waiting for flag 1-16 16= (TS.WF1)waiting for flag 17-32 20= (TS.WF2)waiting for flag 33-48 22= (TS.WF3)waiting for flag 49-64 24= (TS.WF4)waiting for flag 1-64 26= (TS.EXT)task exited byte 1- (A.FB)task flags byte bit 7- (AF.PP)primary protocol task word 8- (A.TD)STD entry address word 9- (A.EF)task event flags 1-16 word 10- task event flags 17-32 word 11- (A.FM)task event flags mask 1-16 word 12- task event flags mask 17-32 word 13- task event flags mask 33-48 word 14- task event flags mask 49-64 word 15- (A.PF)power fail AST trap address word 16- zero

FE Status Block
Active Task List

ADDRESS	SIZE	NAME	USE
007202	40	TTYTSK	TTY task ATL entry see above
007242	40	RPTSK	RP task ATL entry see above
007302	40	LPTSK	LP task ATL entry see above
007352	40	CDTSK	CD task ATL entry see above
007402	40	FETSK	FE task ATL entry (2040S,2060,2065,1091,1095)
007402	40	DTTSK	DECtape task ATL entry (1090) see above
007442	40	DXTSK	Floppy task ATL entry (2040S,2060,2065,1091,1095)
007440	40	FETSK	FE task ATL entry (1090) see above
007502	40	QPRTSK	Queued protocol task ATL entry see above
007542	40	NULTSK	Null task ATL entry see above

TASK PARTITION DIRECTORY

ADDRESS	SIZE	NAME	USE
007602	20	INSTPD	Install TPD entry 8 word TPD entry word 1- (T.PN)partition name (1st 3 chars) word 2- partition name (2nd 3 chars) word 3- (T.BA)base address of partition word 4- (T.PZ)size of partition word 5- (T.FW)partition flags word bit 1- (TF.OU)partition occupied word 6- (T.HP)1/64th base addr of 1st hole word 7- (T.RF)MRL forward linkage word 8- (T.RB)MRL backward linkage
007622	20	DTETPD	DTE TPD entry see above
007642	20	FETPD	FE TPD entry see above
007662	20	TTYTPD	TTY TPD entry see above
007702	20	LPTPD	LP TPD entry see above
007722	20	CDRTPD	CR TPD entry see above
007742	20	QPRTPD	Queued protocol TPD entry see above
007762	20	DXTPD	Floppy TPD entry (2040S,2060,2065,1091,1095)
007762	20	DTTPD	DECtape TPD entry (1090) see above
007802	20	RPTPD	RP TPD entry see above
010022	20	F11TPD	F11ACP TPD entry see above
010042	20	GENTPD	GEN partition TPD entry see above

DEVICE QUEUE POINTERS

ADDRESS	SIZE	NAME	USE
010062	40	.DQPBA	CTY and DL11 queue Entry for all terminals Entry for DL11 lines 8 words per entry word 1- address of device table list word 2- size of entry in device table word 3- address of device start routine word 4- address of device stop routine word 5- spare word 6- address of acknowledge routine word 7- spare word 8- device count
010122	20	.DQDH0	DH11 queue Entry for DH11 lines
010142	120	.DQDLS	Data line scanner queue entry for all terminals entry for line printer entry for card reader entry for clock entry for FE device

LOGICAL UNIT TABLES

ADDRESS	SIZE	NAME	USE
010262	50	TTPEN	Terminal PUD entry 20 word entry word 1- (U.DN)ASCII device name word 2- byte 0- (U.UN)unit number byte 1- (U.FB)flags byte bit 5- (UF.OFL)device offline bit 6- (UF.TL)recognizes load/record bit 7- (UF.RH)handler resident word 3- (U.C1)characteristics word bit 0- (UC.REC)record oriented device bit 1- (UC.CCL)carriage control device bit 2- (UC.TTY)TTY device bit 3- (UC.DIR)directory device bit 4- (UC.SDI)single directory device bit 5- (UC.SQD)sequential device bit 6- (UC.ETB)18 bit mode bit 8- (UC.NB)intermediate buffered bit 9- (UC.SWL)software write locked bit 10- (UC.ISP)input spooled bit 11- (UC.0SP)output spooled bit 12- (UC.PSE)pseudo device bit 13- (UC.COM)communications channel bit 14- (UC.F11)files 11 device bit 15- (UC.MNT)mountable device word 4- (U.C2)characteristics word bit 0-(CH.LAB)labeled tape bit 3-(CH.NDC)no control functions bit 4-(CH.NAT)no attaching bit 5-(CH.UNL)dismount pending bit 6-(CH.FOR)foreign volume bit 7-(CH.OFF)volume offline word 5- (U.C3)characteristics word word 6- (U.C4)characteristics word word 7- (U.AF)ATL node of task word 8- (U.RP)redirect pointer word 9- (U.HA)handler task ATL node word 10- (U.RF)request forward linkage word 11- (U.RB)request backward linkage word 12- (U.VA)address of control block word 13- (U.UI)owner UIC byte 0- (U.PC)programmer code byte 1- (U.GC)group code word 14- (U.VP)volume protection word word 15- (U.AR)access rights word 16- (U.DACP)default ACP name word 17- (U.ACP)STD address of ACP word 18- (U.TF)terminal privilege word bit 0- (UT.PR)terminal privileged bit 1- (UT.SL)TTY slaved bit 2- (UT.LG)TTY logged on word 19- (U.LBH)high order no. of blocks word 20- (U.LBN)low order no. of blocks

FE Status Block
Logical Unit Tables

ADDRESS	SIZE	NAME	USE
010332	50	RPPEN	1st disk PUD entry see above
010402	50	.RP1PE	2nd disk PUD entry see above
010452	50	.RP2PE	3rd disk PUD entry see above
010522	50	.RP3PE	4th disk PUD entry see above
010572	50	.RP4PE	5th disk PUD entry see above
010642	50	.RP5PE	6th disk PUD entry see above
010712	50	.RP6PE	7th disk PUD entry see above
010762	50	.RP7PE	8th disk PUD entry see above
011032	50	DX0PEN	1st floppy PUD entry (2040S,2060,2065,1091,1095)
011032	50	DT0PEN	1st DECtape PUD entry (1090) see above
011102	50	DX1PEN	2nd floppy PUD entry (2040S,2060,2065,1091,1095)
011102	50	DT1PEN	2nd DECtape PUD entry (1090) see above
011152	50	LP0PUD	Line printer PUD entry see above
011222	50	FE0PUD	FE PUD entry see above
011272	50	SY0PUD	System PUD entry see above

HARDWARE OPTIONS

ADDRESS	SIZE	NAME	USE
011342	2	.CPUSN	KL10 CPU serial number 0=not read <1=can't be read >1=valid serial number
011344	2	.HRDWR	Hardware options bit 0- MCA25 cache pager bit 1- MOS master oscillator bit 2- extended addressing bit 3- internal channels bit 4- cache bit 5- line frequency 0=60 hertz 1=50 hertz
011346	6	.ERRPC	KL PC register
011354	6	.ERRCD	Flags and warm restart (fault continuation) error codes 1=clock error stop 2=EBUS parity error 3=deposit examine error 4=keep alive stopped 5=protocol timeout 6=fast memory parity error 7=CRAM parity error 10=DRAM parity error 11=KL halted 12=KL requested reboot
0011362	2	.MISC	Miscellaneous bit bit 0- used by PARSER to see if it is time to do the TAKE command bit 1- PARSER has been requested
011364	2	ABCHAR	Last Auto-baud character

FE Status Block

EMERGENCY STACK

ADDRESS	SIZE	NAME	USE
011446	106	INITLM	Once only initialization code
011554	0	EMGSTK	Emergency stack base address

I/O PAGE DUMP

BLOCK ADDRESS	UNIBUS ADDRESS	SIZE	DEVICE NAME	USE
100042	760020	20	DH11	DH terminal controller #1
100062	760040	20	DH11	DH terminal controller #2
100102	760060	20	DH11	DH terminal controller #3
100122	760100	20	DH11	DH terminal controller #4
100142	760120	20	DH11	DH terminal controller #5
100162	760140	20	DH11	DH terminal controller #6
100202	760160	20	DH11	DH terminal controller #7
100222	760200	20	DH11	DH terminal controller #8
110522	770500	10	DM11-BB	Modem controller #1
110532	770510	10	DM11-BB	Modem controller #2
110542	770520	10	DM11-BB	Modem controller #3
110552	770530	10	DM11-BB	Modem controller #4
110562	770540	10	DM11-BB	Modem controller #5
110572	770550	10	DM11-BB	Modem controller #6
110602	770560	10	DM11-BB	Modem controller #7
110612	770570	10	DM11-BB	Modem controller #8
113022	773000	1000	BM873-YH	Bootstrap ROM
114422	774400	40	DTE20	KL10 interface device
115422	775400	20	LP20	Line printer #1 interface
115442	775420	20	LP20	Line printer #2 interface
115632	775610	10	DL11-E	DL terminal interface #1
115652	775630	10	DL11-C	DL terminal interface #2
115662	775640	10	DL11-C	DL terminal interface #3
115672	775650	10	DL11-C	DL terminal interface #4
116722	776700	50	RH11	RP04/06 disk interface
117202	777160	10	CD11	Card reader interface

FE Status Block
I/O Page Dump

BLOCK ADDRESS	UNIBUS ADDRESS	SIZE	DEVICE NAME	USE
117212	777170	10	RX11	Floppy disk interface (2040S,2060,2065,1091,1095)
117362	777340	20	TC11	DECtape interface (1090)
117562	777540	2	DL11-W	Line clock status register
117602	777560	10	DL11	CTY interface
117612	777570	2	SW	Switch register value
117802	777760	2	PSW	Processor status word

APPENDIX A

RSX-20F STOP CODES AND I/O ERROR CODES

This appendix contains two lists of error codes. The first list contains RSX-20F stop codes. Associated with each code is the name of the module that issued the stop code, a short explanation of the error, and a possible cause of the error. The second is a list of I/O error codes that are produced by the device handlers and file control primitives. These error codes have associated messages that are listed along with them; however, due to the many different situations in which these errors can arise, no attempt is made to describe recovery algorithms for these errors.

Code	Module	Meaning
BO3	SCOMM	BUFFER OVERFLOW 3
		The PDP-11 was not able to obtain the buffer space necessary for data it wanted to send to the KL.
		Possible Cause:
		Buffer pool space became exhausted or highly fragmented. R1 contains the node (buffer) size requested. FREPL points to the list of free nodes. .FREPL+2 contains the number of free bytes in the pool. Nodes are linked together in the forward direction through the first word of the node. The second word of each node contains the node size.
BO5	TTYDRR	BUFFER OVERFLOW 5
		The Front-End does not have the buffer space to to send an XON or an XOFF to a line.
CBR	PF	CROBAR ERROR
		DTE20 power did not return after a power-fail restart. RSX-20F allows DTE20 power 30 seconds to reappear.
		Possible Cause:
		Malfunctioning hardware in the KL.

Code	Module	Meaning
DTB	QPRDTE	TO-11 DTE TRANSFER FAILURE

A TO-11-done interrupt has occurred, but the TO-11 address in the DTE TO11AD register (register 22) did not have the expected value. Since TO11AD is incremented for each byte transferred, it should point to the first word following the buffer into which the TO-11 data was written.

Possible Cause:

The PDP-11 received the wrong byte count or, more likely, the DTE has a hardware malfunction. TO11BC contains the negative count of data that was actually transferred. TO11AS contains address of data node. R1 contains expected termination address, and CR$DTB-2 contains the actual termination address for transfer.

DTD	COMTRP	UNIBUS TIMEOUT

Reference to the DTE20 caused a UNIBUS timeout.

Possible Cause:

Malfunction of hardware in the KL.

DTF	QPRDTE	TO-10 DTE TRANSFER FAILURE

A TO-10-done interrupt has occurred, but the TO-10 address in the DTE TO10AD register (register 20) did not have the expected value. Since TO10AD gets incremented for each byte transferred, it should point to the first word following the packet that was sent to the KL.

Possible Cause:

The PDP-11 gave the KL the wrong byte count or, more likely, the DTE has a hardware malfunction. TO10SZ contains the size of the transfer and TO10AS the start address. The expected termination address is in R4.

ETE	QPRDTE	TO-11 TRANSFER ERROR

A DTE interrupt occurred with the TO11ER bit set in the DTE status register (register 34).

Possible Cause:

Hardware malfunction along the data path between the KL and PDP-11 (MBOX, EBOX, EBUS, DTE20, through to PDP-11 memory).

Code	Module	Meaning
FTA	LC	FILES-11 TASK ABORTED

A task occupying F11TPD partition has aborted and the task termination notification task (TKTN) cannot be started since it too runs in the F11TPD partition.

Possible Cause:

.TKTN may have aborted. R5 and .CRTSK point to the Active Task List (ATL) node of the aborted task.

IAS	SCH	UNKNOWN SIGNIFICANT EVENT

An unused bit in .SERFG has been set.

Possible Cause:

PDP-11 hardware malfunctioned or PDP-11 software is corrupt. .SERFG has the bit set.

ILF	QPRDTE	ILLEGAL PROTOCOL FUNCTION

The function code in a TO-11 protocol header specified a function that is outside the legal range or that is currently not implemented.

Possible Cause:

KL software is corrupted or hardware malfunctioned along the data path between the KL and the PDP-11. R1 contains the function code times two. R4 contains the address of the protocol header.

ILQ	QPRDTE	ILLEGAL QUEUE COUNT

The KL and the PDP-11 disagree on the number of direct transfers that have thus far taken place from the KL to the PDP-11. You should take into account that indirect headers are sent across the DTE20 as direct packets.

Possible Cause:

The PDP-11 is missing TO-11 doorbell interrupts, or the software of either the KL or the PDP-11 is corrupted. STATI+0 to STATI+2 contain the KL's TO-11 status word as read by RSX-20F at the last examine. STATI+4 is the count the KL expects, and TO10QC is the count the PDP-11 expects.

LRF	SCH	LOAD REQUEST FAILURE

An attempt to load a nonresident monitor routine into the F11TPD partition failed.

Possible Cause:

The Files-11 system is incomplete or damaged.

Code	Module	Meaning
MPE	LC	MEMORY PARITY ERROR
		A memory parity error has occurred in the PDP-11 (trap to location 114). The memory status registers are stored starting at location PARSAVE. (Refer to the PDP-11 Processor Handbook for details.)
PT1	QPRDTE	PROTOCOL BROKEN
		An illegal protocol device number was specified in a TO-11 request. The number was found to be greater than the maximum allowed device number .DQPSZ (currently 10).
		Possible Cause:
		KL software is corrupted or hardware malfunctioned along the data path between the KL and the PDP-11. The device number from the protocol header is in TO11DV.
PT2	QPRDTE	PROTOCOL ERROR 2
		An illegal protocol function was specified in a TO-11 request. The function was found to be greater than the allowed maximum BC.FNM (currently 34).
		Possible Cause:
		Same as PT1 above. The function code from the protocol header is in TO11FN.
PT3	QPRDTE	PROTOCOL ERROR 3
		The PDP-11 has received a doorbell interrupt from the KL. The indirect bit in the KL's TO-11 status word indicates that an indirect transfer is to be initiated. The function code, however, sent in the last protocol header, does not indicate that an indirect request is in progress (the most significant bit of the function code was not set).
		Possible Cause:
		Same as PT1 above. TO11FN contains the function code and STATI contains the TO-11 protocol status word.
PT4	QPRDTE	PROTOCOL ERROR 4
		The KL wants to send a packet to the PDP-11, but the packet size is greater than 100, the maximum allowed.
		Possible Cause:
		Same as PT1 above. The size is in EQSZ.

Code	Module	Meaning
PxxP	IOTTRP	POWER-FAIL-TRIGGERED CRASH xxx is any crash code. The crash was probably triggered by a problem related to a power failure.
RED	RED	REDIRECT ERROR A fatal error has occurred during an MCR REDIRECT command. The file control service is corrupted. Call your Software Support Specialist.
RES	LC	RESERVED INSTRUCTION TRAP This is the PDP-11 trap to location 10. An attempt was made to execute an illegal or reserved instruction. Refer to the PDP-11 Processor Handbook for further details. Possible Cause: PDP-11 software is corrupted or a PDP-11 hardware malfunction occurred.
TBT	LC	T-BIT TRAP This PDP-11 trap to location 14 occurs when the BPT instruction (not used by RSX-20F) is executed or when the T-bit is set. (See the PDP-11 Processor Handbook for further details.) Possible Cause: Corrupted PDP-11 software or PDP-11 hardware malfunction.
TET	QPRDTE	TO-10-TRANSFER ERROR A DTE20 interrupt has occurred with either TO10ER (TO-10 error) or MPE11 (PDP-11 parity error) bit set in the DTE20 status register (register 34). Possible Cause: DTE20 hardware error, PDP-11 memory parity error, or hardware malfunction along the data path between the PDP-11 and KL.
T04	LC	TRAP AT LOCATION 4 The PDP-11 traps to location 4 when it makes a word reference to an odd address or when a bus timeout occurs. (See the PDP-11 Processor Handbook for further details.) Possible Cause: PDP-11 software is corrupted, or a PDP-11 peripheral device is malfunctioning or has gone away.

Code	Module	Meaning
TTT	IOTTRP	RECURSIVE TRAP ERROR While RSX-20F attempted to bring down the system after a trap, a second trap serious enough to crash the system occurred. Possible Cause: PDP-11 software is corrupted, or PDP-11 hardware is malfunctioning.
UIE	QPRDTE	UNIMPLEMENTED PROTOCOL FUNCTION The KL uses bits 0-2 of its TO-11 status word in the communications region to inform the front end of any disaster occurring in the KL. These bits are read by the front end on receipt of a TO-11 doorbell. The currently implemented functions are KL-RELOAD REQUEST and KL POWER FAIL. Any other bits that are set cause this halt. Possible Cause: Corrupted KL software, a KL hardware malfunction or any hardware malfunction along the data path between the KL and the PDP-11 could be the cause of this error.

The following is a list of possible I/O error codes that RSX-20F can produce. Since these codes are returned by the device handlers and file control primitives in RSX-20F, they are global in the sense that they can come from any utility in the system. That is, a code of -33 means the same thing when it comes from PIP that it means when it comes from SAV. Because of the global nature of the error codes, it is not possible to describe the exact problem; the situation is different with different utilities. Therefore, the following list does not attempt to explain the error code other than to list the message associated with it.

Note that there are two messages associated with the code -2. This is legitimate; a message code of -2 is produced in two types of situations.

Code	Message
-1	Bad parameters
-2	Invalid function code
-2	EBOX stopped
-3	Device not ready
-4	Parity error on device
-5	Hardware option not present
-6	Illegal user buffer
-7	Device not attached
-8	Device already attached
-9	Device not attachable
-10	End of file detected
-11	End of volume detected
-12	Write attempted to locked unit
-13	Data overrun
-14	Send/receive failure

Code	Message
-15	Request terminated
-16	Privilege violation
-17	Sharable resource in use
-18	Illegal overlay request
-19	Odd byte count or virtual address
-20	Logical block number too large
-21	Invalid UDC module
-22	UDC connect error
-23	Caller's nodes exhausted
-24	Device full
-25	Index file full
-26	No such file
-27	Locked from write access
-28	File header full
-29	Accessed for write
-30	File header checksum failure
-31	Attribute control list format error
-32	File processor device read error
-33	File processor device write error
-34	File already accessed on LUN
-35	File ID, file number check
-36	File ID, sequence number check
-37	No file accessed on LUN
-38	File was not properly closed
-39	Open - no buffer space available for file
-40	Illegal record size
-41	File exceeds space allocated, no blocks
-42	Illegal operation on file descriptor block
-43	Bad record type
-44	Illegal record access bits set
-45	Illegal record attributes bits set
-46	Illegal record number - too large
-47	Multiple block read/write - not implemented
-48	Rename - two different devices
-49	Rename - new file name already in use
-50	Bad directory file
-51	Cannot rename old file system
-52	Bad directory syntax
-53	File already open
-54	Bad file name
-55	Bad device name
-56	Bad block on device
-57	Enter duplicate entry in directory
-58	Not enough stack space (FCS or FCP)
-59	Fatal hardware error on device
-60	File ID was not specified
-61	Illegal sequential operation
-62	End of tape detected
-63	Bad version number
-64	Bad file header
-65	Device off-line
-66	File expiration date not reached
-67	Bad tape format
-68	Not ANSI "D" format byte count

APPENDIX B

FILE TRANSFERS BETWEEN TOPS-10/TOPS-20 AND RSX-20F

Normally the KL and the PDP-11 transfer any data that needs to be passed between them without any human intervention. However, occasionally, you may want to move a file from the front-end area to an area that is readable by the KL. This could happen if, for example, you could not find a KL-readable copy of the front-end map file and wanted to transfer a copy from the front-end release media. However, since the file systems for the two processors do not use the same format, the transfer must include a reformatting as well. The software that allows you to reformat the file and transfer it from the PDP-11's area to the KL's area (or vice versa) is described in this appendix.

A TOPS-10 program called RSXT10 is used to make TOPS-10 files readable to the front-end file system. A TOPS-20 program called RSXFMT has a similar function. The programs that are used to transfer files between TOPS-10/TOPS-20 and RSX-20F are FE (under both TOPS-10 and TOPS-20), and PIP (under RSX-20F). All of these programs execute in a normal timesharing environment, but some may be restricted to privileged users.

B.1 REFORMATTING FILES

RSXT10 and RSXFMT, the reformatting programs, are available to all users and do not require any special privileges to execute.

You can invoke RSXT10 by typing:

```
.R RSXT10<CR>
```

RSXT10 responds with the prompt:

```
RSXFMT>
```

You can invoke RSXFMT by typing:

```
@RSXFMT<CR>
```

RSXFMT responds with the same prompt:

```
RSXFMT>
```

At this point, you can give commands to the reformatting program. A description of the available commands is presented in Section B.1.2.

B.1.1 Restrictions

Files that are to be transferred must be reformatted on the KL processor, regardless of the direction of the transfer. Thus, if you wish to transfer files from the front end to the KL, you must do the transfer before the reformatting. If, on the other hand, you wish to transfer files from the KL to the front end, you must reformat the files before the transfer.

Some features are not available in one of the versions of the software. For example, temporary files are not supported by RSXT10. Nor can RSXT10 write to a log file when taking commands from a command file. RSXFMT, on the other hand, does not support MICRO-CODE or SAVE modes. Thus, you should check that the feature you wish to use is supported by the version of the program that you can access.

B.1.2 RSXT10/RSXFMT Commands

The following list describes the commands available to users of RSXT10 and RSXFMT. The parts of the commands enclosed in parentheses do not appear in the dialog with RSXT10; they are part of the TOPS-20 version only. The word NO in square brackets - [NO] - indicates that the command can be negated by preceding the command with NO.

[NO] ADDRESS (WORDS EXIST IN IMAGE FILES)

When you are converting to IMAGE-BINARY files, the program ignores the first two bytes of each record. When you are converting from IMAGE-BINARY, the program inserts two bytes of address at the beginning of each record. The default is NO ADDRESS (WORDS EXIST IN IMAGE FILES).

CONVERT (FILE) <input-file-spec> (OUTPUT AS) <output-file-spec>

This command converts the specified input file group to the output file group, in the mode determined by the MODE command or the input file. The default output file specification is the same as the input file specification, with the next highest generation number.

CRLF (IN ASCII FILES IS) [DEFAULT, IMBEDDED, IMPLIED, CARRIAGE-RETURN-SINGLE-SPACE]

This command selects whether <CR><LF> should be inserted or removed at the end of formatted ASCII records. RSXT10 also converts <CR><LF> to <CR><DC3> if you specify the final option. This option is available only to users of RSXT10. If you are using RSXT10, the default for .MAP and .DIR file types is IMBEDDED, whereas the default for .LST files is CARRIAGE-RETURN-SINGLE-SPACE. Other file types default to IMPLIED. If you are using RSXFMT, the default for all files is DEFAULT.

EXIT (FROM RSXFMT)

This command returns control to the TOPS-10 Monitor or the TOPS-20 Executive.

[NO] IGNORE (FILE FORMAT ERRORS)

File format errors produce warning messages only if this command has previously been issued.

HELP (WITH RSXFMT)

This command types this text.

INFORMATION (ABOUT) [ADDRESS, ALL, CRLF, IGNORE, MODE, RECORD-SIZE, TEMPORARY]

This command displays the settings of the various status commands. INFORMATION TEMPORARY does not work under RSXT10, since temporary files are not supported by the TOPS-10 version of the reformatting program.

MODE (OF INPUT) <mode-type> (AND OUTPUT) <mode-type>

This command selects input and output modes, where <mode-type> is one of the following:

- 7-BIT-ASCII
- DOS-BINARY
- DEFAULT
- IMAGE-BINARY
- MICRO-CODE
- RSX-ASCII
- RSX-BINARY
- SAVE

DEFAULT input mode is selected by the file type and the first word of the current input file. DEFAULT output mode is selected by a mapping from the mode of the current input file. The MICRO-CODE and SAVE modes are available exclusively to users of RSXT10.

RECORD-SIZE (FOR IMAGE FILES IS) <decimal number>

This command selects the record size for files being converted from IMAGE-BINARY format. Default is 256 bytes.

TAKE (COMMANDS FROM FILE) <command-file-spec> (LOGGING OUTPUT ON) <log-file-spec>

This command takes RSXFMT commands from the specified file. RSXT10 does not support the output to a log file.

[NO] TEMPORARY (OUTPUT FILES)

If you specify TEMPORARY, all output files (see CONVERT command) are written as temporary files. You might wish to use this command if you want to maintain a copy of the file in TOPS-20-readable format after the file is transferred to the front end. This feature is not supported in RSXT10.

B.2 TRANSFERRING FILES

The act of transferring files is logically separate from the reformatting process, since the reformatting can occur at different points depending on the direction of the transfer. To accomplish the actual transfer, the FE program must be running, the FE: device must be assigned, and the user must invoke several tasks with the PARSER. These actions are discussed more fully in the following sections.

B.2.1 Running FE

The FE program must be executed by a privileged user. It can run detached if this is desirable. FE does not have any commands. It simply runs while the user transfers files.

For users of TOPS-10, FE can be invoked and detached by either of the following two command sequences:

```
.R FE<CR>
^C

.CCONT
```

or

```
.GET SYS:FE<CR>
JOB SETUP
.CSTART<CR>
.DETACH<CR>
```

Users of TOPS-20 can invoke and detach FE by typing:

```
@ENABLE (CAPABILITIES)
$FE<CR>
^C

@DETACH (AND) CONTINUE
```

When running under TOPS-10, FE requires access to the UIC/[p,pn] mapping file: SYS:FEUIC.TXT. Each invocation of FE causes this file to be read. FEUIC.TXT is an ASCII file that is created and maintained with standard TOPS-10 Text Editors (like TECO or SOS). The format of the UIC-to-PPN mapping descriptor is:

```
[uic]=STR:[p,pn]
```

In this descriptor, [uic] must already exist in the front-end file system. STR: must be a valid TOPS-10 structure name, and [p,pn] must be a valid TOPS-10 directory. The default for STR: is DSK:. FEUIC.TXT can contain as many UIC to [p,pn] mappings as required. Furthermore, these mappings can be internally documented by the insertion of comments, which must begin with a semicolon or an exclamation mark.

TOPS-20 uses a different approach to UIC/[p,pn] mapping. When running under TOPS-20, FE does not look for any file containing the mapping from directory to UIC. Instead, FE contains its own algorithm to find the UIC. The algorithm it uses is the following:

```
UIC = [(340+(D/400)),(Dmod400)]
```

where D is the TOPS-20 directory number. (The directory number can be printed in response to an INFORMATION (ABOUT) DIRECTORY command.) Thus, if your TOPS-20 directory number is 164, your UIC would be

[(340+(164/400)),(164mod400)] = [340,164]

Since 164/400 is less than 1, the quantity is dropped. The quantity 164mod400 remains after 164 is divided by 400 as many times as will go evenly; in this case, since 400 does not go into 164, the remainder is 164.

B.2.2 The FE: Device

The FE: device exists under both TOPS-10/TOPS-20 and RSX-20F. The FE: device is often referred to as a pseudodevice because it is not a physical device, but a logical one. You can think of the FE: device as the other file system, regardless of which system - the KL or the PDP-11 - you are presently using. When you assign and use the FE: device, you notify the two processors of the link between the file systems.

B.2.3 RSX-20F Tasks

In order to transfer files between the file systems, three RSX-20F tasks must be invoked and released. These tasks are MOU (MOUNT), PIP (file transfer), and DMO (DMOUNT). To invoke these RSX-20F tasks, type CTRL/\ on the CTY. The CTRL/\, which is not echoed on the terminal, invokes the PARSER. Type:

CTRL/\

The system responds with the PARSER prompt:

PAR>

The tasks themselves are invoked by the MCR command. For example:

PAR>MCR MOU<CR>

This command invokes the MOU (MOUNT) task. All RSX-20F tasks prompt by typing their three-character task name and a right bracket. All RSX-20F tasks are released by typing a CTRL/Z.

B.2.4 File Transfer Dialog

The following sequence of steps is used to transfer files both to and from the front-end file system:

1. Run (and detach) FE.
2. Use the MOUNT task to mount the RSX-20F FE: device.
3. Use PIP to transfer the file(s).
4. Dismount the RSX-20F FE: device using DMOUNT.

5. Go back to monitor level and stop FE.

6. Deassign the FE: device from your job.

This basic sequence is used for all file transfers. However, reformatting always takes place on the KL, regardless of which operating system is running there, and regardless of the direction of the transfer.

To transfer files from TOPS-10/TOPS-20 to RSX-20F, invoke the RSX-20F PIP task and type the following command string:

```
PIP>[uic]filename.ext=FE:[uic]filename.ext<CR>
```

To transfer files from RSX-20F to TOPS-10/TOPS-20, invoke the RSX-20F PIP task and type the following command string:

```
PIP>FE:[uic]filename.ext=[uic]filename.ext<CR>
```

Refer to Section 6.4 for details on the RSX-20F utility, PIP.

The following example shows an operator copying the file TEST.TXT from TOPS-20 to RSX-20F. The copy in the opposite direction can be effected by switching the file specifications to the opposite sides of the equal sign.

```
@LOG OPERATOR (PASSWORD)
 Job 7 on TTY205 10-AUG-83 12:48:00
@RSXFMT
RSXFMT>CONVERT (FILE) TEST.RNO.2 (OUTPUT AS) TEST.TXT
TEST.RNO.2 [7-BIT-ASCII] ==> TEST.TXT.1 [RSX-ASCII]
RSXFMT>EXIT (FROM RSXFMT)

@ENABLE (CAPABILITIES)
$ASSIGN FE0:
$FE
^C
$DETACH (AND) CONTINUE
Detaching job #7

PAR>MCR MOU
MOU>FE:
MOU -- MOUNT COMPLETE
MOU>^Z

PAR>MCR PIP

PIP>TEST.TXT/LI
PIP -- NO SUCH FILE(S)

PIP>SY:TEST.TXT=FE:[340,5]TEST.TXT
PIP>TEST.TXT/LI


DIRECTORY DB0:[5,5]
10-AUG-83 12:50

TEST.TXT;1 1.       10-AUG-83 12:49

        TOTAL OF 1.  BLOCKS IN 1.  FILE
```

```
PIP>^Z

PAR>MCR DMO
DMO>FE:
DMO -- DISMOUNT COMPLETE
DMO>^Z

 TOPS-20 BIG SYSTEM, T2 Monitor 5.1(5101)
@ATT OPERATOR (JOB #) 7
Password:
^C
$INFORMATION (ABOUT) FILE-STATUS (OF JFN)
 Connected to PS:<OPERATOR>.  JFNS:
 4    TEST.TXT.2  Not opened Read, EOF
 3    FE0:        Read, Append, 0.(16)
 2    <SUBSYS>FE.EXE.2    Read, Execute
 1    <SYSTEM>EXEC.EXE.51         Read, Execute

Devices assigned to/Opened by this job: FE0, TTY205

$CLOSE (JFN) 3,4
 4   TEST.TXT.2   [OK]
 3   FE0:   [OK]
$DEASSIGN FE0:
$LOGOUT
```

APPENDIX C

FRONT-END TASKS

The tasks that are listed here are those tasks that exist separately from the RSX-20F Executive. These tasks reside in the front-end file area from which they can be loaded into core and executed in either the GEN user partition or the F11TPD system partition.

F11ACP.TSK — Files-11 Ancillary Control Processor
An Ancillary Control Processor (ACP) is an extension of the monitor. F11ACP handles the front-end disk files, and performs file access, management, and control functions. F11ACP runs in the F11TPD partition.

PARSER.TSK — The Command Parser
PARSER is the primary means of access to the front-end programs. It also controls the KLINIK link and provides KL diagnostic tools. PARSER runs in the GEN partition.

KLI.TSK — KLINIT
KLINIT initializes the KL processor by loading the microcode, configuring memory, configuring cache, and then loading and starting the KL bootstrap program. KLINIT runs in the GEN partition.

LOGXFR.TSK — LOGXFR
LOGXFR transfers PARSER.LOG (the snapshot taken by KLERR) across the DTE to the KL, where it is placed in the ERROR.SYS file. LOGXFR runs in the GEN partition.

MOU.TSK — Mount a Device
MOUNT makes a device known to F11ACP so that it can be accessed by a given user. MOUNT runs in the GEN partition.

PIP.TSK — Peripheral Interchange Program
PIP performs general file transfers and some maintenance functions between Files-11 devices and other peripherals. PIP runs in the GEN partition.

TKTN.TSK — Task Termination Program
TKTN outputs task termination notification and provides for the orderly termination of front-end tasks. It also acts as an interface between KLINIT and KLERR. TKTN runs in the GEN partition.

COP.TSK — Copy from device to device
COPY is a device copy utility that allows verification of the physical state of the device. COPY supports both floppy disks and DECtapes. COPY runs in the GEN partition.

RED.TSK — Redirect the system device
REDIRECT moves the front-end system device from one Files-11 device to another and informs the system of its new location. REDIRECT runs in the GEN partition.

INI.TSK — Initialize volumes
INI initializes Files-11 devices to be recognizable Files-11 volumes and sets up Master Directory space, index, home block, and so forth. INI runs in the GEN partition.

UFD.TSK — User File Directory
UFD creates User File Directories on Files-11 volumes. User File Directories are used to store file identifiers. UFD runs in the GEN partition.

T20ACP.TSK — TOPS-20 Ancillary Control Processor
T20ACP is the file handler for files to be transferred to and from the KL's disk file area. It interacts with the TOPS-10 and TOPS-20 device FE:. T20ACP provides access to the TOPS-10 and TOPS-20 disk file areas in terms compatible with Files-11 operations. T20ACP runs in the GEN partition.

SAV.TSK — Save system image
SAV creates a task-image file of the current RSX-20F monitor and saves it in the Files-11 area. SAV runs in the GEN partition.

DMO.TSK — Dismount a Device
DMOUNT declares a device off-line to F11ACP and therefore inaccessible to a user. DMOUNT runs in the GEN partition.

SETSPD.TSK — Set Line Speeds
SETSPD sets the line-speed table in the KL after a restart. It also sets the time in the KL processor. SETSPD.TSK is a front-end task and is not to be confused with the TOPS-20 program, SETSPD.EXE. SETSPD runs in the F11TPD partition.

KLRING.TSK — KLINIK Request
KLRING checks the KLINIK time window and password whenever the KLINIK line rings. If the time and security checks are verified, KLINIK is enabled. KLRING runs in the F11TPD partition.

KLDISC.TSK — KLINIK Disconnect
KLDISC performs system functions associated with disconnecting the KLINIK line. KLDISC also logs significant KLINIK events across the DTE into the KL ERROR.SYS file. KLDISC runs in the F11TPD partition.

MIDNIT.TSK — Update the clock
Each time the clock passes midnight, MIDNIT updates the time and date on the PDP-11. Then, if the KL is running, MIDNIT obtains the KL's time and date and resets its own to match. MIDNIT runs in the F11TPD partition.

APPENDIX D

KLINIK ACCESS DIALOG

The RSX-20F KLINIK link allows DIGITAL Field Service or Software Support personnel at remote locations to access a KL-based computer as a timesharing user or as a remote console terminal operator. The computer may or may not be up for timesharing, but the front end must have RSX-20F running. The link is controlled by the operator of the computer, who can allow or disallow access, and can also terminate the KLINIK link. If the KL monitor supports error logging, the RSX-20F Executive records significant events and errors for analysis with the SPEAR mechanism.

This appendix lists the events that are logged by RSX-20F, and describes the KLINIK access parameters. It also documents the commands used in the access dialog from the point of view of both the computer operator and the Field Service or Software Support person who wishes to access a remote KL.

D.1 SIGNIFICANT KLINIK EVENTS

The KLINIK events that RSX-20F considers significant are logged in the ERROR.SYS file, and can be read with SPEAR. The significant events are:

- Each occurrence of a SET KLINIK command (the parameters given in the command are also saved)
- Each occurrence of a CLEAR KLINIK command
- Each occurrence of a DISCONNECT command or a DL11E hang-up
- Each occurrence of a successful LOGON (the mode selected is also saved)
- Each occurrence of an unsuccessful LOGON (the number of attempts is also saved)

D.2 KLINIK ACCESS PARAMETERS

The computer operator and the person who wishes to access the computer from a remote location must agree on certain parameters regarding the

time at which the link will take place and the type of access the remote user will have. Specifically, they must agree on:

- Whether the remote user of the link wishes to have a remote CTY or simply a timesharing terminal
- Which password will allow the remote user access to the system, if the user has requested a remote CTY
- The date and time the remote user will dial up to request access by way of the KLINIK link
- The highest console mode the remote user will be allowed

Once this information has been verified, the computer operator must notify RSX-20F of the arrangements by using the SET KLINIK command, described in Section D.3.1.

D.2.1 Usage of the Remote Terminal

The remote user can access the system in two ways: as a normal timesharing user, or as a remote operator. If the remote terminal is set up for timesharing, the remote user can deal with the system just as any other timesharing user; thus, the KLINIK link could be used as a special dial-up line. Alternatively, the remote terminal can be declared to be the system's CTY, thereby allowing the remote user to access the system as if the user were present at the local CTY. In this case, both the remote user and the system operator have the ability to enter commands to RSX-20F and to see all output. In fact, it is possible to execute PARSER commands that are entered by two people typing alternate characters from the two consoles.

The system operator declares the usage of the remote terminal when the PARSER requests the usage mode in the dialog following the SET KLINIK command. The legal replies are REMOTE and USER; no defaults exist.

D.2.2 Access Password for Remote CTY's

The system operator must declare to RSX-20F that the remote user wishes to have a terminal of the agreed-upon type. If the remote user is to have a CTY, the operator must also give RSX-20F a password that the remote user must repeat to establish the KLINIK link.

The operator declares the password when the PARSER requests it in the dialog following the SET KLINIK command. This password must be one to six numeric or uppercase alphabetic characters with no embedded or trailing blanks.

D.2.3 KLINIK Access Window

The date and time at which the remote user plans to establish the link is specified by opening a window, that is, defining two times between which the remote user can access the computer. This window has no effect once the remote user has gained access to the system; the KLINIK link is not terminated when the end of the window is reached. However, access to the system is not allowed unless it is requested between the specified times.

The operator specifies the access window dates and times in response to the PARSER's prompt in the SET KLINIK dialog. The access window dates are specified in the following format:

```
DD-MMM-YY
DD-MMM-YYYY
DD MMM YY
DD MMM YYYY
```

where DD is the day, MMM is the alphabetic representation of the month, and YY or YYYY is the year. The year can be specified either by using the entire Gregorian year, or by using only the last two digits of the year number. If only the last two digits are specified, the PARSER assumes that the first two digits should be 19.

The access window times are specified in the following format:

```
HHMM
HH:MM
```

where HH is the hour in 24-hour format and MM is the minute. HH must be in the range 00 to 24, and MM must be in the range 00 to 60.

The default condition for both dates and times can be accepted by replying to the relevant prompts with a carriage return. The default date and time for the opening of the access window are the current system date and time. The default date for the closing of the access window is the current system date plus one day. The default time for the closing of the access window is the current system time. It is possible to specify the date on which the access window will open and still allow the time at which the window will open to default to the current time of day. It is also possible to allow the date to default while specifying the time. A similar situation exists for the closing of the access window.

D.2.4 Console Mode of the Remote Terminal

If the remote terminal is being used for simple timesharing, the security systems of the operating system are assumed to be in control. However, when the remote user requests the use of a remote CTY, the computer operator must maintain the security of the system by declaring which type of access the remote user is to have. (Refer to Section 4.3, PARSER Console Modes, for a discussion of the capabilities of the various modes.)

The operator specifies the console mode of the remote terminal in the dialog following the SET KLINIK command. The legal replies to the PARSER's request for the console mode are MAINTENANCE, PROGRAMMER, and OPERATOR. There are no default replies to this question. The local operator should make sure that sufficient capabilities are supplied to the remote user initially, since it is not possible for either the operator or the remote user to raise the console mode while the KLINIK link is active.

D.3 OPERATOR DIALOG WITH KLINIK

The system operator, through dialog with the PARSER, sets parameters for establishing the KLINIK link, checks those parameters, terminates the KLINIK link, and disconnects the remote modem. The dialog that the operator uses to accomplish these tasks is presented below.

D.3.1 Setting Access Parameters

The system operator must declare to RSX-20F that a remote user will be accessing the system before the remote user can actually attempt to establish the link. When the parameters discussed in Section D.1 have been set using the SET KLINIK command, the remote user can dial up and attempt to establish the KLINIK link, assuming that the user does so during the agreed-upon time window.

You should use the following dialog to set the access parameters.

1. Type CTRL/\ (Control-Backslash) to enter the PARSER.

2. When you receive the PARSER's prompt, enter the SET KLINIK command to tell the PARSER you wish to set KLINIK parameters.

3. Answer the PARSER's prompt (KLINIK MODE:) with the console mode you wish to allow the remote user: REMOTE to give the user a remote CTY, or USER to make the remote terminal into a normal timesharing terminal. If you respond with something other than REMOTE or USER, one of the following error messages appears:

```
PAR -- [SET] NSK - NO SUCH KEYWORD "xxx"
PAR -- [SET] ILC - ILLEGAL CHARACTER "c"
```

   where "xxx" and "c" are the offending keyword and character, respectively.

   After printing the relevant error message, the PARSER aborts the SET KLINIK command.

4. If you answered the previous prompt with REMOTE to allow a remote CTY, you must answer the PARSER's prompt (PASSWORD:) with the password that the remote user must give to be allowed access to the computer. If you do not provide a legal password, you receive one of the following error messages.

   If you specified no password, you get:

```
PAR -- [SET] NPI - NULL PASSWORD ILLEGAL
```

   If you typed more than six characters, you get:

```
PAR -- [SET] PTL - PASSWORD TOO LONG
```

   If you included a character that was not an alphanumeric, you get:

```
PAR -- [SET] IPC - ILLEGAL PASSWORD CHARACTER "c"
```

   where "c" is the offending character.

   After printing the relevant error message, the PARSER aborts the SET KLINIK command.

5. Answer the PARSER's access window prompts (ACCESS WINDOW OPEN DATE:, ACCESS WINDOW OPEN TIME:, ACCESS WINDOW CLOSE DATE:, ACCESS WINDOW CLOSE TIME:) with dates and times in the format explained above (in Section D.1.3).

   If the PARSER cannot recognize the date in the format you have used, you receive one of the following error messages.

   If the day specified does not exist in the month specified, you get:

   ```
   PAR -- [SET] DOR - DATE OUT OF RANGE
   ```

   If the month you specified cannot be matched, you get:

   ```
   PAR -- [SET] NSK - NO SUCH KEYWORD "xxx"
   ```

   If the keyword you specified for the month is ambiguous, you get:

   ```
   PAR -- [SET] AMB - AMBIGUOUS KEYWORD "xxx"
   ```

   In both of the previous cases, "xxx" is the offending keyword.

   If the year is not recognizable, you get:

   ```
   PAR -- [SET] YOR - YEAR OUT OF RANGE
   ```

   If the access window open or close date is prior to the current system date, you get:

   ```
   PAR -- [SET] DBT - DATE BEFORE TODAY
   ```

   If the access window open or close time does not conform to the required format, you get:

   ```
   PAR -- [SET] TOR - TIME OUT OF RANGE
   ```

   If the open or close time is not numeric, you get:

   ```
   PAR -- [SET] ITF - ILLEGAL TIME FORMAT
   ```

   Finally, when you have answered all four prompts (or allowed the default condition to hold), the PARSER checks that the opening date and time you specified are before the closing date and time. If this is not the case, you get:

   ```
   PAR -- [SET] KWE - KLINIK WINDOW ERROR
   ```

   If you made an error in typing a command, the PARSER aborts the SET KLINIK command when it finishes printing the relevant error message.

6. If you specified REMOTE in response to the KLINIK MODE: prompt, you must now reply to the PARSER's prompt (HIGHEST CONSOLE MODE:) with the highest PARSER console mode you wish to allow the remote user. The legal replies are MAINTENANCE, PROGRAMMER, and OPERATOR. There is no default reply to this question. (Refer to Section 4.3, PARSER Console Modes, for a discussion of the capabilities of the various modes.) If the

PARSER does not recognize the console mode you specify, you get the following error message:

```
PAR -- [SET] NSK - NO SUCH KEYWORD "xxx"
```

where "xxx" is the offending keyword.

If you enter only a carriage return in response to the prompt, you get:

```
PAR -- [SET] MRA - MISSING REQUIRED ARGUMENT
```

After printing the relevant error message, the PARSER aborts the SET KLINIK command.

7. If you have specified all of the parameters correctly, the PARSER returns to command level after displaying the KLINIK parameters in the following format:

```
KLINIK [<state>]
ACCESS WINDOW OPEN:  DD-MMM-YY HH:MM
ACCESS WINDOW CLOSED:  DD-MMM-YY HH:MM
KLINIK MODE:  [<mode>]
```

where <state> can be ACTIVE, INACTIVE, or DISABLED, and <mode> may be REMOTE or USER. If the KLINIK MODE is REMOTE, one more line is displayed:

```
HIGHEST CONSOLE MODE:  [<mode>]
```

where <mode> can be MAINTENANCE, PROGRAMMER, or OPERATOR.

The state of the KLINIK link is described by the first line, which tells whether the link is ACTIVE, INACTIVE, or DISABLED. ACTIVE means the KLINIK parameters have been set and the remote user is currently accessing the system. INACTIVE means that the parameters have been set, but the remote user is not currently accessing the computer. DISABLED means that the parameters have not been set.

The use of the SET KLINIK command is illustrated by the following example:

```
PAR>SET KLINIK
KLINIK MODE:  REMOTE
PASSWORD:  ASDF
ACCESS WINDOW OPEN DATE:
ACCESS WINDOW OPEN TIME:
ACCESS WINDOW CLOSE DATE:
ACCESS WINDOW CLOSE TIME:
HIGHEST CONSOLE MODE:  OPERATOR
 KLINIK INACTIVE
 ACCESS WINDOW OPEN:  10-AUGUST-1983 13:04
 ACCESS WINDOW CLOSE:  10-AUGUST-1983 13:04
 KLINIK MODE:  REMOTE
 HIGHEST CONSOLE MODE:  OPERATOR
PAR>
```

D.3.2 Examining the Current KLINIK Parameters

The KLINIK parameters that are displayed at the end of the SET KLINIK dialog can be displayed at will by the use of the WHAT KLINIK command. The format of the information is exactly the same as the display following the SET KLINIK dialog when the KLINIK parameters have been set. In two cases, however, the format is different. One is the time when no KLINIK parameters have been set. In this case, the following line (only) will be displayed in response to the WHAT KLINIK command:

```
KLINIK DISABLED
```

The other exceptional case is when the KLINIK link is active after a reboot of the system software. (This situation is described more fully in Section D.4.) In this case the response to the WHAT KLINIK command is in the form shown below:

```
KLINIK ACTIVE FROM REBOOT
KLINIK MODE:  REMOTE
HIGHEST CONSOLE MODE:  MAINTENANCE
```

Normal use of the WHAT KLINIK command is illustrated by the following example:

```
PAR>WHAT KLINIK
 KLINIK INACTIVE
 ACCESS WINDOW OPEN:  10-AUGUST-1983 13:04
 ACCESS WINDOW CLOSED:  10-AUGUST-1983 13:04
 KLINIK MODE:  REMOTE
 HIGHEST CONSOLE MODE:  MAINTENANCE
PAR>
```

D.3.3 Terminating the KLINIK Link

The system operator can terminate the KLINIK link at any time by the CLEAR KLINIK command. The CLEAR KLINIK command clears the KLINIK parameters, but does not hang up the modem. If the remote user has a remote CTY, the user can also terminate the link by the same method. In either case, the link is not completely cleared until the operator issues the DISCONNECT command. This command hangs up the modem, thus ending the link. Breaking up the termination into two commands has the advantage of allowing the link to be terminated (by use of the DISCONNECT command) without clearing the parameters. Thus, the remote user can try again to establish the link, but the operator does not need to reenter the parameters.

When the operator issues the CLEAR KLINIK command during the time the link is in use, the following messages are printed on both terminals:

```
 KLINIK DISABLED

KLD -- KLINIK ACCESS TERMINATED BY OPERATOR
```

When the operator issues the DISCONNECT command, the following message is printed on both terminals:

```
KLD -- KLINIK LINE DISCONNECTED
```

This message signals the end of the link, since the modem is hung up by the DISCONNECT command.

If the remote user tries again to gain access to the system before the parameters are reset, access is denied, and the following messages appear on both the remote and local terminals:

```
KLR -- KLINIK RING - WINDOW CLOSED

KLD -- KLINIK LINE DISCONNECTED
```

The termination dialog is illustrated by the following example:

```
PAR>CLEAR KLINIK
 KLINIK DISABLED

KLD -- KLINIK ACCESS TERMINATED BY OPERATOR

PAR>DISCONNECT

KLD -- KLINIK LINE DISCONNECTED

PAR>
```

D.4 REMOTE USER DIALOG WITH KLINIK

The remote user of the KLINIK link can be accessing the computer system for a variety of reasons. For example, Software Support and Field Service personnel may wish to run software and hardware diagnostics. It can also be useful to watch some problem-causing event as it occurs. Other DIGITAL personnel may use the link to gather statistics on local system usage.

The link can be set up in two different ways to accommodate this variety. As discussed in Section D.1.1, the remote terminal can be either a normal timesharing user or a remote CTY.

D.4.1 Logging In as a Remote Operator

RSX-20F answers the DL11E when it rings and decides what to do based on the current setting of the KLINIK MODE parameter. If the KLINIK MODE is REMOTE, RSX-20F prints the following message on both the remote and local terminals:

```
KLR -- KLINIK RING - VALIDATING ACCESS
```

This message is followed on the remote terminal by:

```
PASSWORD:
```

At this time you should type in the password that was previously agreed upon. You have five tries to give the correct password. During the time the KLINIK link is being validated, the PARSER is unavailable to either the local or the remote terminals. If either the local operator or the remote user attempts to invoke the PARSER, the request is queued and the validation process is continued.

If you give an incorrect password, RSX-20F prints the following message:

```
KLR -- INCORRECT PASSWORD
```

RSX-20F then waits ten seconds before allowing any further attempt to enter the password. During this ten-second wait, anything you type is ignored. After ten seconds, the following prompt is printed again:

```
PASSWORD:
```

If after five tries you are unable to give the correct password, the following messages are printed on both terminals:

```
KLR -- KLINIK LOGON TIMEOUT -
KLR -- LOGON ABORTED

KLD -- KLINIK LINE DISCONNECTED
```

At this time, RSX-20F hangs up the modem. The remote user can dial up and try again to gain access to the system if the local operator does not issue the CLEAR KLINIK command.

When you type the correct password, RSX-20F prints the following message on the remote terminal:

```
KLINIK MODE:
```

In response, you must type either REMOTE or USER. If you wish to be connected to RSX-20F, you must type REMOTE. You then receive the following notification from RSX-20F:

```
KLR -- KLINIK LINE CONNECTED TO RSX-20F
KLR -- CONSOLE MODE LIMIT:  [<mode>]
```

where <mode> can be MAINTENANCE, PROGRAMMER, or OPERATOR.

You can login to the local system as a timesharing user even though the KLINIK MODE was declared by the operator to be REMOTE. If you wish to do this, simply reply to the KLINIK MODE: prompt with USER. If you do this, you receive the following message before RSX-20F routes the line to the KL monitor:

```
KLR -- KLINIK LINE CONNECTED TO TOPS-xx
```

where "xx" is either 10 or 20. The next line printed is the system herald, just as a normal timesharing user would receive.

The following examples illustrate the dialog between RSX-20F and the remote user. The first example shows a remote user answering the KLINIK MODE: prompt with REMOTE to get a remote CTY. The second example shows how the same user could decide to login to the system as a timesharing user.

1.
```
KLR -- KLINIK RING - VALIDATING ACCESS

PASSWORD:  [the password will not echo on the terminal]

KLR -- INCORRECT PASSWORD
PASSWORD:  [this time it is correct]

KLINIK MODE:  REMOTE

KLR -- KLINIK LINE CONNECTED TO RSX-20F
KLR -- CONSOLE MODE LIMIT:  [MAINTENANCE]
```

2. KLR -- KLINIK RING - VALIDATING ACCESS

   PASSWORD: [the password will not echo on the terminal]

   KLINIK MODE: USER

   KLR -- KLINIK LINE CONNECTED TO TOPS-20

   SYSTEM 2116 THE BIG ORANGE, TOPS-20 MONITOR 5.1(5101)
   @

D.4.2 Logging In as a Timesharing User

If the local system operator has declared the usage of the remote terminal to be USER, the link is routed to the KL monitor after RSX-20F prints

```
KLR -- KLINIK LINE CONNECTED TO TOPS-xx
```

where "xx" is either 10 or 20. The next line printed is the system message from the particular system, just as a normal timesharing user would receive.

The following example shows the messages printed when a remote user logs in to the system as a timesharing user.

Example D-6

```
KLR -- KLINIK LINE CONNECTED TO TOPS-20

SYSTEM 2116 THE BIG ORANGE, TOPS-20 MONITOR 5.1(5101)
@
```

D.5 KLINIK INTEGRITY OVER A REBOOT

The computer system attempts to maintain the integrity of a KLINIK link over a reload of system software. Both the KL processor and the front-end processor are aware of the KLINIK link, and both store the current KLINIK parameters. This allows one processor to remind the other of the current state of the KLINIK link should one of the two processors be reloaded. If the link was set up to be REMOTE MODE, the following messages are printed on both terminals:

```
SAV -- *DIAG* -- KLINIK LINE ACTIVE IN REMOTE MODE
SAV -- *DIAG* -- KLINIK LINE CONNECTED TO SYSTEM CONSOLE
```

If the link was set up to be USER MODE, the following message is printed on both terminals:

```
SAV -- *DIAG* -- KLINIK LINE ACTIVE IN USER MODE
```

The KLINIK link is also maintained over a reload of the entire system. That is, if both the KL and the front end are reloaded, RSX-20F detects the carrier signal when it comes up and realizes that a KLINIK link is in progress. At this point, RSX-20F waits 45 seconds for the

KL to provide the correct KLINIK parameters. Since in this situation there is no way for the KL to know the original KLINIK parameters, it is unable to supply the parameters. Thus, when the 45-second wait is over, RSX-20F sets up default parameters and continues the link. The parameters are set up for REMOTE MODE with the highest console mode being MAINTENANCE. The messages printed are the same as those printed on reloading only one of the processors.

In the event that the KL monitor has difficulty starting or restarting the Primary or Secondary Protocols, the following message is printed on the local console:

```
SAV -- *FATAL* -- PROTOCOLS NOT RUNNING
```

This problem usually requires a reload of the TOPS-10 or TOPS-20 system.

APPENDIX E

GETTING HELP ON RSX-20F

At times it becomes necessary for users of KL-based computers to get help on some aspect of RSX-20F. There may be some problem with the RSX-20F software, or the user may have some hardware problem that RSX-20F detects but cannot deal with. If you find that your installation is having a problem of this sort and you wish to submit a Software Performance Report or place a hot line call to Software Services, consult this appendix before calling for help. This appendix provides assistance in making sure you supply all the needed information to allow DIGITAL personnel to determine what the problem is.

The items you should include with an SPR, or have ready when you make a hot line call, are listed below. Providing this information to the Software Support personnel speeds up the answering of your question and helps insure that you receive a complete and useful answer.

1. Dump File(s) - Include the dump file(s) that were taken at the time of the problem. The filename for every dump file is ODMP11.BIN. One of these files is generated for every crash of RSX-20F, as long as the KL is running when RSX-20F dies. You can also produce a dump manually, if the situation calls for it. You should be aware, however, that the manual production of a dump file defeats any attempt by RSX-20F to save the state of the processor as RSX-20F sees it. Specifically, the stack pointer (SPSAV) will not contain the address of the next instruction to be executed. If you feel that it would be helpful to produce a dump, press the HALT switch on the front end, then raise it again immediately. Make sure that you record the circumstances of the crash and correlate the particular circumstances with the particular dump, especially if you are submitting more than one dump file. Also, if you have produced the dump file by hand, be sure to make that fact known, because it will definitely influence the method of extracting information from the dump.

2. Console Log - Include the console log (or a copy of it) from the time of the crash (or other problem). The copy you include should cover any recent odd occurrences, as well as a running commentary. This commentary is useful for determining, not only the sequence of events, but the timing of the events as well. Thus, if you try one method of recovering from the problem, then think about the problem for half an hour, then try another approach, make sure that your commentary notes the half-hour delay, since the delay cannot be inferred from reading the console log itself.

3. ERROR.SYS Entries - Include any ERROR.SYS entries generated by the problem. You can also include any entries generated around the time the problem occurred, since they may have a bearing on the problem that you do not realize at the time.

4. Description of Problem - Include a description of the problem. Writing down exactly what seemed to be happening on the system, what signals told you a problem existed, and what attempts you made to recover from the problem, can save a good deal of time in getting your answer. If the Software Support personnel do not have this information, they may have to try to get in touch with you to get it, thereby delaying your receiving an answer.

5. Device Descriptions - Include a description of any device involved in the problem. It would be wise, also, to include a description of any nonstandard device that you have hooked to your system, since these are often the cause of unusual problems.

APPENDIX F

EIA PIN DEFINITIONS

The following table lists the pin definitions that are part of the EIA standards. DTE here refers not to the DTE20 device, but to the Data Terminal Equipment - in other words, the terminal. DCE refers to Data Communications Equipment - in other words, whatever hardware interface you are using between the terminal and the host computer.

Pin	Name	To DTE	To DCE	Function	Circuit (CCITT)	Circuit (EIA)
1	FD			Frame Ground	101	(AA)
2	TD		>	Transmitted Data	103	(BA)
3	RD	<		Received Data	104	(BB)
4	RTS		>	Request To Send	105	(CA)
5	CTS	<		Clear To Send	106	(CB)
6	DSR	<		Data Set Ready	107	(CC)
7	SG			Signal Ground	102	(AB)
8	DCD	<		Data Carrier Detect	109	(CF)
9		<		Positive Dc Test Volt		
10		<		Negative Dc Test Voltage		
11				Unassigned		
12	SDCD	<		Sec. Data Carrier Detect	122	(SCF)
13	SCTS	<		Sec. Clear To Send	121	(SCB)
14	STD		>	Sec. Transmitted Data	118	(SBA)
15	TC	<		Transmitter Clock	114	(DB)
16	SRD	<		Sec. Received Data	119	(SBB)
17	RC	<		Receiver Clock	115	(DD)
18			>	Receiver Dibit Clock		
19	SRTS		>	Sec. Request To Send	120	(SCA)
20	DTR		>	Data Terminal Ready	108.2	(CD)
21	SQ	<		Signal Quality Detect	110	(CG)
22	RI	<		Ring Indicator	125	(CE)
23			>	Data Rate Select	111/112	(CH/CI)
24	(TC)		>	External Transmitter Clock	113	(DA)
25			>	Busy		

TOPS–10/TOPS–20
RSX–20F System
Reference Manual
AA–BS94A–TK

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____ Telephone _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear – Fold Here and Tape

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MRO1–2/L12
MARLBOROUGH, MA 01752

Do Not Tear – Fold Here and Tape

Cut Along Dotted Line

UPDATE NOTICE

TOPS–10/TOPS–20 RSX–20F System Reference Manual AD–BS94A–T1

April 1986

Insert this Update Notice in the *TOPS–10/TOPS-20 RSX–20F System Reference Manual* to maintain an up-to-date record of changes to the manual.

Changed Information

The changed pages contained in this update package reflect the changes to the new version of RSX–20F.

The instructions for inserting this update start on the next page.

© Digital Equipment Corporation 1986. All Rights Reserved.

digital software

INSTRUCTIONS
AD–BS94A–T1

The following list of page numbers specifies which pages are to be placed in the *TOPS–10/TOPS–20 RSX–20F System Reference Manual* as replacements for, or additions to, current pages.

Title Page	5–17	10–5
Copyright Page	5–18.2	10–6
Entire	5–31	10–25
Contents	5–32	10–43
3–3	5–43	A–5
3–4	5–44	A–7
4–13	6–3	Entire
4–18	6–4	Appendix C
5–1	6–13	Entire
5–2	6–14	Index
5–7	7–11	
5–8.2	7–16	

KEEP THIS UPDATE NOTICE IN YOUR MANUAL TO MAINTAIN AN UP–TO–DATE RECORD OF CHANGES.

TYPE AND IDENTIFICATION OF DOCUMENTATION CHANGES.

Five types of changes are used to update documents contained in the TOPS–10/TOPS–20 software manuals. Change symbols and notations are used to specify where, when, and why alterations were made to each update page. The five types of update changes and the manner in which each is identified are described in the following table.

The Following Symbols and/or Notations	Identify the Following Types of Update Changes
1. Change bar in outside margin; version number and change date printed at bottom of page.	1. Changes were required by a new version of the software being described.
2. Change bar in outside margin; change date printed at bottom of page.	2. Changes were required to either clarify or correct the existing material.
3. Change date printed at bottom of page.	3. Changes were made for editorial purposes but use of the software is not affected.
4. Bullet (•) in outside margin; version number and change date printed at bottom of page.	4. Data was deleted to comply with a new version of the software being described.
5. Bullet (•) in outside margin; change date printed at bottom of page.	5. Data was deleted to either clarify or correct the existing material.

April 1986