decsystem10

MONITOR CALLS

digital

DEC-10-OMCMA-B-D
DEC-10-OMCMA-B-DN1

decsystem10

MONITOR CALLS

This manual reflects the software of the
6.02 monitor.

digital equipment corporation · maynard, massachusetts

First Printing, June 1971
Revised: January 1972
June 1972
March 1973
May 1974
November 1975
March 1976

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.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by DIGITAL.

Copyright © 1971, 1972, 1973, 1974, 1975, 1976 by Digital Equipment Corporation

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

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM	DECsystem-20	TYPESET-11

CONTENTS

			Page
CHAPTER	1	USER PROGRAMMING	1-1
	1.1	PROCESSOR MODES	1-1
	1.1.1	User Mode Processing	1-1
	1.1.2	User I/O Mode Processing	1-2
	1.1.3	Executive Mode Processing	1-2
	1.2	MONITOR CALLS (PROGRAMMED OPERATORS)	1-2
	1.2.1	Op Codes 001-037	1-2
	1.2.2	Op Codes 040-100 and 000	1-2
	1.2.2.1	Physical Only	1-19
	1.2.2.2	Restriction on Monitor Calls in Programs	1-19
	1.2.3	Operation Codes 100-127	1-19
	1.2.4	Illegal Operation Codes	1-19
	1.2.5	Naming Convention for Monitor Symbols	1-19
	1.2.5.1	Symbols for Numbers	1-19
	1.2.5.2	Symbols for Masks	1-20
	1.2.5.3	Symbols for Monitor Calls	1-20
	1.2.5.4	Symbols for GETTAB Tables	1-20
	1.2.5.5	Symbols for Error Codes	1-20
CHAPTER	2	MEMORY FORMAT	2-1
	2.1	USER PROGRAMS	2-1
	2.2	MEMORY PROTECTION AND RELOCATION	2-1
	2.2.1	The KA10 Processor	2-2
	2.2.2	The KI10 and KL10 Processors (Without Virtual Memory)	2-3
	2.2.3	KI10 and KL10 Processors With the Virtual Memory Option	2-4
	2.2.3.1	Virtual Memory Organization	2-4
CHAPTER	3	JOB DATA AREA	3-1
	3.1	JOBDAT (JOB DATA AREA)	3-1
	3.2	VESTIGIAL JOB DATA AREA	3-5
CHAPTER	4	JOB CONTROL AND INFORMATION	4-1
	4.1	JOB CONTROL	4-1
	4.1.1	Start Program Execution	4-1
	4.1.2	Stop Program Execution	4-1
	4.1.2.1	The HALT Instruction	4-1
	4.1.2.2	The EXIT Monitor Call (CALLI 12)	4-2
	4.1.2.3	The LOGOUT Monitor Call (CALLI 17)	4-2
	4.1.3	Suspend the Execution Of A Job	4-2
	4.1.3.1	The SLEEP Monitor Call (CALLI 31)	4-2
	4.1.3.2	The HIBERnate Monitor Call (CALLI 72)	4-3
	4.1.3.3	The WAKE Monitor Call (CALLI 73)	4-3
	4.2	SET OR OBTAIN JOB INFORMATION	4-4
	4.2.1	Set the Program Name	4-4
	4.2.2	Set System/Job Parameters	4-5
	4.2.3	Set the Logical Node	4-8
	4.2.4	Obtain Run Time	4-9
	4.2.5	Obtain the Job Number of the Calling Job	4-9

CONTENTS (Cont.)

		Page
4.2.6	Obtain the Project-Programmer Number of the Calling Job	4-9
4.2.7	The OTHUSR Monitor Call (CALLI 77)	4-10
4.3	TIMING INFORMATION	4-10
4.3.1	The DATE Monitor Call (CALLI 14)	4-11
4.3.2	The MSTIME Monitor Call (CALLI 23)	4-11
4.3.3	The TIMER Monitor Call (CALLI 22)	4-11
4.4	CONFIGURATION INFORMATION	4-11
4.4.1	The SWITCH Monitor Call (CALLI 20)	4-11
4.4.2	The LIGHTS Monitor Call (CALLI -1)	4-11
4.4.3	The DAEMON Monitor Call (CALLI 102)	4-12
4.5	MONITOR EXAMINATION	4-13
4.5.1	The PEEK Monitor Call (CALLI 33)	4-13
4.5.2	The SPY Monitor Call (CALLI 42)	4-14
4.5.3	The POKE Monitor Call (CALLI 114)	4-14
4.6	INTER-PROGRAM COMMUNICATION	4-14
4.6.1	The TMPCOR Monitor Call (CALLI 44)	4-14
CHAPTER 5	TRAPPING, INTERCEPTION, AND INTERRUPTION	5-1
5.1	USER TRAP SERVICING	5-1
5.2	ERROR INTERCEPTING	5-2
5.3	SOFTWARE INTERRUPT SYSTEM	5-5
5.3.1	Interrupt Conditions	5-6
5.3.2	Interrupt Control Block	5-8
5.3.3	Initialize the Software Interrupt System	5-8.1
5.3.4	Control the Software Interrupt System	5-9
5.3.5	Save the Interrupt Blocks	5-11
5.3.6	Reload the Saved State of the Interrupt System	5-12
5.3.7	Dismiss an Interrupt	5-12
5.3.8	An Example of the Software Interrupt System	5-12
CHAPTER 6	CORE AND SEGMENT CONTROL	6-1
6.1	CORE CONTROL	6-1
6.1.1	Definitions	6-1
6.1.2	The LOCK Monitor Call (CALLI 60)	6-1
6.1.2.1	The LOCK Monitor Call Extension	6-4.1
6.1.2.2	Minimizing Fragmentation	6-7
6.1.3	The UNLOK.Monitor Call (CALLI 120)	6-8
6.1.4	The CORE Monitor Call (CALLI 11)	6-8
6.1.5	The SETUWP Monitor Call (CALLI 36)	6-10
6.1.6	The PAGE. Monitor Call (CALLI 145)	6-11
6.1.6.1	Page Fault Handling	6-14
6.1.6.2	Format of the Page Fault Handler	6-15
6.2	SEGMENT CONTROL	6-16
6.2.1	The RUN Monitor Call (CALLI 36)	6-16
6.2.1.1	Programming with the RUN Monitor Call	6-18
6.2.2	The GETSEG Monitor Call (CALLI 40)	6-18
6.2.3	The REMAP Monitor Call (CALLI 37)	6-19
6.2.4	Testing for a Sharable High Segment	6-20
6.2.5	Determining the High Segment Origin	6-21
6.2.6	Modifying Shared Segments and Meddling	6-21

CONTENTS (Cont.)

			Page
CHAPTER	7	I/O PROGRAMMING	7-1
	7.1	JOB INITIALIZATION	7-1
	7.2	DEVICE SELECTION	7-2
	7.2.1	Device Initialization	7-3
	7.2.2	Device Names	7-4
	7.2.2.1	File Structure Names	7-5
	7.2.2.2	Logical Unit Names	7-5
	7.2.2.3	Physical Controller Class Names	7-5
	7.2.2.4	Physical Controller Names	7-5
	7.2.2.5	Physical Unit Names	7-5
	7.2.2.6	Name Abbreviations	7-5
	7.3	DATA MODES	7-6
	7.3.1	Unbuffered Data Modes	7-6
	7.3.2	Buffered Data Modes	7-9
	7.3.2.1	Buffered Input	7-9
	7.3.2.2	Buffered Output	7-9
	7.3.2.3	Synchronization of Buffered I/O	7-10
	7.3.3	Buffer Structure	7-10
	7.3.3.1	Buffer Ring Header Block	7-10
	7.3.3.2	Buffer Ring	7-11
	7.3.3.3	Monitor Generated Buffers	7-12
	7.3.3.4	User Generated Buffers	7-12
	7.3.4	Non-Blocking I/O	7-13
	7.4	DEVICE TERMINATION AND REASSIGNMENT	7-13
	7.4.1	RELEASE a Device	7-13
	7.4.2	The RESDV. Monitor Call (CALLI 117)	7-13
	7.4.3	The REASSIGN Monitor Call (CALLI 21)	7-14.1
	7.4.4	The DEVLNM Monitor Call (CALLI 107)	7-14.1
	7.5	DEVICE INFORMATION	7-15
	7.5.1	The DEVSTS Monitor Call (CALLI 54)	7-15
	7.5.2	The DEVCHR Monitor Call (CALLI 4)	7-15
	7.5.3	The DEVTYP Monitor Call (CALLI 53)	7-16
	7.5.4	The DEVSIZ Monitor Call (CALLI 101)	7-16
	7.5.5	The WHERE Monitor Call (CALLI 63)	7-19
	7.5.6	The NODE Monitor Call (CALLI 157)	7-19
	7.5.7	The DEVNAM Monitor Call (CALLI 64)	7-21
CHAPTER	8	FILES	8-1
	8.1	FILE DEFINITION	8-1
	8.2	STRUCTURE OF DISK FILES	8-1
	8.2.1	File Directories	8-1
	8.2.2	Job Search List	8-4
	8.2.3	Storage Allocation Table (SAT) Blocks	8-5
	8.3	DISK FILE FORMAT	8-5
	8.4	ACCESS PROTECTION	8-6
	8.4.1	File Access Privileges	8-6
	8.4.2	Directory Privileges	8-7
	8.5	FILE NAMES	8-8
	8.5.1	The RENAME Operator	8-8
	8.6	FILE SELECTION	8-10
	8.6.1	LOOKUP/ENTER A File	8-10
	8.6.1.1	The LOOKUP Operator	8-10

CONTENTS (Cont.)

			Page
	8.6.1.2	The ENTER Operator	8-11
	8.6.1.3	Extended Arguments to LOOKUP, ENTER and RENAME	8-13
	8.6.1.4	Error Recovery for ENTER and RENAME Monitor Calls	8-21
	8.6.1.5	Non-Superseding ENTER	8-21
	8.6.2	Data Transmission	8-21
	8.6.2.1	The FILOP. Monitor Call (CALLI 155)	8-22
	8.7	THE PATH. MONITOR CALL (CALLI 110)	8-22.2
	8.7.1	PATH. Examples	8-25
	8.8	USETI AND USETO Operators	8-27
	8.9	THE SEEK MONITOR CALL (CALLI 56)	8-29
	8.10	THE CHKACC MONITOR CALL (CALLI 100)	8-30
	8.11	THE STRUUO MONITOR CALL (CALLI 50)	8-32
	8.12	THE JOBSTR MONITOR CALL (CALLI 47)	8-32
	8.13	THE GOBSTR MONITOR CALL (CALLI 66)[1]	8-33
	8.14	THE SYSSTR MONITOR CALL (CALLI 46)	8-33
	8.15	THE SYSPHY MONITOR CALL (CALLI 51)	8-34
	8.16	THE DEVPPN MONITOR CALL (CALLI 55)	8-34
	8.17	THE DSKCHR MONITOR CALL (CALLI 45)	8-36
	8.18	THE DISK. MONITOR CALL (CALLI 121)	8-39
	8.19	FILE STATUS	8-41
	8.19.1	The GETSTS Operator	8-42
	8.19.2	The STATO/STATZ Operators	8-43
	8.19.3	The SETSTS Operator	8-43
	8.20	TERMINATE A FILE	8-44
CHAPTER	9	I/O PROGRAMMING FOR DECTAPE	9-1
	9.1	DECTAPE	9-1
	9.2	DATA MODES	9-1
	9.2.1	Buffered Data Modes	9-1
	9.2.2	Unbuffered Data Modes	9-2
	9.3	DECTAPE FORMAT	9-2
	9.3.1	DECtape Directory Format	9-2
	9.3.2	DECtape File Format	9-4
	9.3.3	Block Allocation	9-5
	9.4	I/O PROGRAMMING	9-5
	9.4.1	The LOOKUP Operator	9-5
	9.4.2	The ENTER Operator	9-7
	9.4.3	The RENAME Operator	9-7
	9.4.4	The INPUT, OUTPUT, CLOSE, RELEASE Operators	9-8
	9.5	SPECIAL MONITOR CALLS	9-9
	9.5.1	The USETI Operator	9-9
	9.5.2	The USETO Operator	9-9
	9.5.3	The UGETF Operator	9-9
	9.5.4	The UTPCLR Monitor Call	9-9
	9.5.5	The MTAPE Operator	9-9
	9.5.6	The DEVSTS Monitor Call After Each Interrupt	9-9
	9.6	FILE STATUS	9-9
	9.7	IMPORTANT CONSIDERATIONS	9-11
CHAPTER	10	I/O PROGRAMMING FOR MAGNETIC TAPE	10-1
	10.1	DATA MODES	10-2

CONTENTS (Cont.)

			Page
	10.2	SPECIAL MONITOR CALLS FOR MAGNETIC TAPE	10-3
	10.2.1	The MTAPE Operator	10-3
	10.2.1.1	Function 11, Rewind and Unload	10-4
	10.2.2	The MTCHR. Monitor Call (CALLI 112)	10-5
	10.2.2.1	Nine-Channel Tapes	10-6.1
	10.2.3	The TAPOP. Monitor Call (CALLI 154)	10-6.1
	10.2.3.1	Function .TFMOD (Data Modes)	10-11
	10.2.3.2	Read Backwards (TX01 Only)	10-12
	10.2.4	The MTAID. Monitor Call (CALLI 126)	10-12
	10.3	FILE STATUS	10-12
CHAPTER	11	I/O PROGRAMMING WITH TERMINALS	11-1
	11.1	INTRODUCTION	11-1
	11.2	TERMINAL MONITOR CALLS	11-1
	11.2.1	The TTCALL Operator	11-1
	11.2.1.1	TTCALL Examples	11-3
	11.2.2	The GETLIN Monitor Call (CALLI 34)	11-4
	11.2.3	The TRMNO. Monitor Call (CALLI 115)	11-5
	11.2.4	The TRMOP. Monitor Call (CALLI 116)	11-5
	11.3	DATA MODES	11-9
	11.4	FILE STATUS	11-12
	11.5	PAPER-TAPE INPUT FROM THE TERMINAL (FULL-DUPLEX SOFTWARE)	11-13
	11.6	PAPER–TAPE OUTPUT AT THE TERMINAL (FULL-DUPLEX SOFTWARE)	11-13
	11.7	PSEUDO-TTYS (PTYs)	11-14
	11.7.1	Concepts	11-14
	11.7.2	The HIBER Monitor Call (CALLI 72)	11-15
	11.7.3	File Status	11-15
	11.7.4	Special Monitor Calls	11-16
	11.7.4.1	The OUT, OUTPUT Operators (Op Codes 57 and 67)	11-16
	11.7.4.2	The IN, INPUT Operators (Op Codes 56 and 66)	11-16
	11.7.4.3	The RELEASE Monitor Call (Op Code 71)	11-16
	11.7.4.4	The JOBSTS Monitor Call (Op Code 61)	11-16
	11.7.4.5	The CTLJOB Monitor Call (CALLI 65)	11-17
CHAPTER	12	I/O PROGRAMMING WITH UNIT RECORD DEVICES	12-1
	12.1	THE CARD PUNCH (CDP)	12-2
	12.1.1	Data Modes and Buffer Zones	12-2
	12.1.2	Monitor Calls	12-2
	12.1.3	File Status	12-3
	12.2	THE CARD READER (CDR)	12-4
	12.2.1	Data Modes	12-4
	12.2.1.1	ASCII, Octal Code 0	12-4
	12.2.1.2	ASCII Line, Octal Code 1	12-4
	12.2.1.3	Image, Octal Code 10	12-4
	12.2.1.4	Image Binary, Octal Code 13	12-4
	12.2.1.5	Binary, Octal Code 14	12-4
	12.2.1.6	Super-Image, Code 110_2	12-4
	12.2.2	Monitor Calls	12-4
	12.2.3	File Status	12-5
	12.3	DISPLAY WITH LIGHT PEN	12-6
	12.3.1	Data Modes	12-6
	12.3.2	Background	12-6

CONTENTS (Cont.)

		Page
12.3.3	Display Monitor Calls	12-6
12.3.3.1	The INPUT Operator	12-6
12.3.3.2	The OUTPUT Operator	12-6
12.3.4	File Status	12-8
12.4	LINE PRINTER	12-9
12.4.1	Data Modes	12-9
12.4.1.1	ASCII, Octal Code 0	12-9
12.4.1.2	ASCII Line, Octal Code 1	12-9
12.4.1.3	Image, Octal Code 10	12-9
12.4.2	Monitor Calls	12-9
12.4.3	File Status	12-9
12.5	THE PAPER-TAPE PUNCH	12-10
12.5.1	Data Modes	12-10
12.5.1.1	ASCII, Octal Code 0	12-10
12.5.1.2	ASCII Line, Octal Code 1	12-10
12.5.1.3	Image, Octal Code 10	12-10
12.5.1.4	Image Binary, Octal Code 13	12-10
12.5.1.5	Binary, Octal Code 14	12-10
12.5.2	Monitor Calls	12-10
12.5.3	File Status	12-10
12.6	THE PAPER-TAPE READER	12-11
12.6.1	Data Modes	12-11
12.6.1.1	ASCII, Octal Code 0	12-11
12.6.1.2	ASCII Line, Octal Code 1	12-11
12.6.1.3	Image, Octal Code 10	12-11
12.6.1.4	Image Binary, Octal Code 13	12-11
12.6.1.5	Binary, Octal Code 14	12-11
12.6.2	Monitor Calls	12-11
12.6.3	File Status	12-11
12.7	PLOTTER	12-13
12.7.1	Data Modes	12-13
12.7.1.1	ASCII, Octal Code 0	12-13
12.7.1.2	ASCII Line, Octal Code 1	12-13
12.7.1.3	Image, Octal Code 10	12-13
12.7.1.4	Image Binary, Octal Code 13	12-13
12.7.1.5	Binary, Octal Code 14	12-13
12.7.2	Monitor Calls	12-13
12.7.3	File Status	12-13
CHAPTER 13	THE MULTIPLEX CHANNEL FEATURE	13-1
13.1	BUFFER RING EXTENSION	13-1
13.1.1	Device Chains	13-2
13.2	DATA MODES	13-2
13.3	DEVICE IDENTIFICATION	13-2
13.4	MPX MONITOR CALLS	13-2
13.4.1	The CNECT. Monitor Call (CALLI 130)	13-2
13.4.2	The ERLST. Monitor Call (CALLI 132)	13-3
13.4.3	The SENSE. Monitor Call (CALLI 133)	13-4
13.4.4	The CLRST. Monitor Call (CALLI 134)	13-5
13.4.5	The IONDX. Monitor Call (CALLI 127)	13-6
13.4.6	The MVHDR. Monitor Call (CALLI 131)	13-6

CONTENTS (Cont.)

			Page
CHAPTER	14	REAL-TIME PROGRAMMING	14-1
	14.1	THE RTTRP MONITOR CALL (CALLI 57)	14-1
	14.1.1	Interrupt Level Use of RTTRP	14-4
	14.1.2	Restrictions	14-4
	14.1.3	Removing Devices from a Priority Interrupt Channel	14-5
	14.1.4	Dismissing the Interrupt	14-5
	14.2	THE TRPSET MONITOR CALL (CALLI 25)	14-5
	14.2.1	The UJEN Operator (Op Code 100)	14-7
	14.2.2	The HPQ Monitor Call (CALLI 71)	14-7
CHAPTER	15	INTER-PROCESS COMMUNICATION FACILITY	15-1
	15.1	PACKETS	15-1
	15.1.1	Flags	15-1
	15.1.2	PIDs	15-1
	15.1.3	Length of the Packet Data Block	15-4
	15.1.4	Address of the Packet Data Block	15-4
	15.1.5	Sender's Project-Programmer Number	15-5
	15.1.6	Capabilities of Sender	15-5
	15.1.7	Packet Data Block	15-5
	15.2	SENDING A PACKET	15-5
	15.3	RECEIVING A PACKET	15-6
	15.4	[SYSTEM] INFO	15-6
	15.5	[SYSTEM] ICPP	15-7
	15.6	STATUS OF AN INPUT QUEUE	15-10
	15.7	RETRIEVE AN IPCF PACKET	15-11
	15.8	SEND AN IPCF PACKET	15-11
	15.9	IPCF EXAMPLE	15-13
CHAPTER	16	ENQUEUE/DEQUEUE FACILITY	16-1
	16.1	OVERVIEW OF ENQUEUE/DEQUEUE	16-1
	16.1.1	Shared Ownership and Exclusive Ownership	16-1
	16.1.2	Pooled Resources	16-2
	16.1.3	Sharer's Group	16-4
	16.2	ENQUEUE/DEQUEUE MONITOR CALLS	16-5
	16.2.1	The ENQ. Monitor Call (CALLI 151)	16-5
	16.2.2	The DEQ. Monitor Call (CALLI 152)	16-7
	16.2.3	The ENQC. Monitor Call (CALLI 153)	16-9
	16.2.3.1	Status Information	16-9
	16.2.3.2	Modifying the Queue Structure	16-11
CHAPTER	17	METERING	17-1
	17.1	OVERVIEW OF METERING	17-1
	17.2	METER POINT ROUTINES	17-2
	17.3	USING THE TRACE CHANNEL	17-2
	17.4	ADDING A METER POINT	17-3
	17.5	THE METER. MONITOR CALL (CALLI 111)	17-4
CHAPTER	18	GETTABS	18-1
APPENDIX A		COMPARISON OF DISK DEVICES	A-1
APPENDIX B		COMPARISON OF MAGNETIC TAPE SYSTEMS	B-1

CONTENTS (Cont.)

			Page
APPENDIX C		CARD AND TAPE CODES	C-1
APPENDIX D		COMPARISON OF TERMINALS	D-1
APPENDIX E		ERROR CODES	E-1
APPENDIX F		DECSYSTEM-10 AT-A-GLANCE	F-1
APPENDIX G		MEMORIES	G-1
APPENDIX H		FILE RETRIEVAL POINTERS	H-1
	H.1	A GROUP POINTER	H-1
	H.1.1	Folded Checksum Algorithm	H-1
	H.2	END-OF-FILE POINTER	H-1
	H.3	CHANGE OF UNIT POINTER	H-2
	H.4	DEVICE DATA BLOCK	H-2
	H.5	ACCESS BLOCK	H-2
APPENDIX I		DATA COMMUNICATIONS	I-1
APPENDIX J		UUOSYM.MAC	J-1
APPENDIX K		2741 TERMINALS	K-1
INDEX			Index-1

FIGURES

			Page
FIGURE	2-1	KA10 User Address Relocation	2-2
	2-2	KI10/KL10 Paging Configuration (Without Virtual Memory)	2-3
	2-3	Physical and Virtual Page Limits	2-5
	5-1	Software Interrupt Process	5-6
	5-2	Interrupt Control Block	5-8
	5-3	Saved Status Block Structure	5-11
	6-1	Locking Jobs in Core	6-4.1
	7-1	Buffer Ring Header Block	7-11
	7-2	Buffer Structure	7-11
	8-1	Basic Disk File Organization for Each File Structure	8-2
	8-2	Disk File Organization	8-3
	8-3	Access Protection Code	8-7
	8-4	ENTER Argument Block	8-12
	8-5	FILOP. Argument Block	8-22
	8-6	PATH. Argument Block	8-22.2
	8-7	Directory Paths on a Single File Structure	8-26
	8-8	Directory Paths on Multiple File Structures	8-26
	8-9	DISK. Priority Level	8-40
	9-1	DECtape Directory Format	9-3
	9-2	Format of a File on a DECtape	9-5
	9-3	Format of a DECtape Block	9-5
	9-4	LOOKUP/ENTER/RENAME Argument Block	9-6
	10-1	Data Word on Tape	10-6.1
	11-1	Pseudo-TTY	11-14
	13-1	MPX Buffer Ring Header Block	13-1

FIGURES (Cont.)

			Page
FIGURE	13-2	Status Block Entry	13-5
	15-1	Representation of an IPCF Packet	15-2
	15-2	Request to [SYSTEM] INFO	15-6
	15-3	[SYSTEM] INFO Response Format	15-8
	15-4	Request to [SYSTEM] IPCC	15-8
	15-5	Response from [SYSTEM] IPCC	15-10
	15-6	An Associated Variable	15-11
	16-1	Shared Ownership	16-2
	16-2	Shared and Exclusive Ownership Requests	16-3
	16-3	Exclusive Ownership	16-3
	16-4	Pooled Resources	16-4
	16-5	ENQ. Argument Block	16-5
	16-6	Lock-Block Dump	16-12
	16-7	Queue-Block Dump	16-12
	17-1	Metering	17-1
	K-1	Model 2741 Keyboard	K-1

TABLES

			Page
TABLE	1-1	Op Codes 040-100	1-3
	1-2	CALLIs/Monitor Calls	1-6
	2-1	VM Abbreviations	2-5
	3-1	Job Data Area Locations	3-1
	3-2	Vestigial Job Data Area Locations	3-6
	4-1	HIBERnate Conditions	4-4
	4-2	SETUUO Functions	4-5
	4-3	DAEMON Functions	4-12
	4-4	DAEMON Error Codes	4-13
	5-1	APRENB Flags	5-2
	5-2	Error Intercepting Class Bits	5-3
	5-3	I/O Interrupt Conditions	5-7
	5-4	Non-I/O Interrupt Conditions	5-8.1
	5-5	Control Flags	5-9
	5-6	Argument Block Flags	5-10
	5-7	PISYS. Error Codes	5-10
	5-8	PISAV. Error Codes	5-11
	5-9	PIRST. Error Codes	5-12
	6-1	LOCK Bits	6-2
	6-2	LOCK Error Codes	6-3
	6-3	LOCK Extension Functions	6-7
	6-4	Values Returned from CORE	6-9
	6-5	PAGE. Functions	6-11
	6-6	Bits Returned from Function .PAGCA	6-13
	6-7	PAGE. Error Codes	6-14
	6-8	Page Fault Word	6-15
	7-1	Non-Directory Devices	7-2
	7-2	OPEN Status Bits	7-4
	7-3	Format of Device Names	7-4

TABLES (Cont.)

			Page
TABLE	7-4	Physical Disk Unit Names	7-6
	7-5	Data Modes	7-7
	7-6	DEVLNM Error Codes	7-15
	7-7	Device Characteristics	7-17
	7-8	Device Type Bits	7-18
	7-9	DEVSIZ Error Codes	7-19
	7-10	NODE Function Codes	7-19
	7-11	NODE Error Codes	7-21
	8-1	Access Protection Codes	8-6
	8-2	Access Privileges to UFDs and SFDs	8-8
	8-3	LOOKUP Argument Block	8-11
	8-4	Extended Arguments to ENTER, LOOKUP, and RENAME	8-14
	8-5	.RBSTS Bit Definitions	8-20
	8-6	FILOP. Function Codes	8-22.1
	8-7	PATH. Function Codes	8-23
	8-8	PATH. Switches and Flags	8-23
	8-9	USETI/USETO Function Codes	8-28
	8-10	CHKACC Access Codes	8-31
	8-11	JOB/GOBSTR Status Bits	8-32
	8-12	GOBSTR Error Codes	8-34
	8-13	Ersatz Devices	8-35
	8-14	DSKCHR Argument Block	8-37
	8-15	DSKCHR Status Bits	8-38
	8-16	DISK. Function Codes	8-40
	8-17	DISK. Error Codes	8-41
	8-18	Disk File Status Bits	8-42
	9-1	DECtape Devices	9-1
	9-2	Format of Words 105 - 126 in the DECtape Directory Block	9-3
	9-3	LOOKUP Parameters	9-6
	9-4	ENTER Parameters	9-7
	9-5	RENAME Parameters	9-8
	10-1	Magnetic Tape Data Modes	10-2
	10-2	MTAPE Functions	10-3
	10-3	MTCHR. Returned Values	10-5
	10-4	Values Returned to the AC After MTCHR.	10-6
	10-5	TAPOP. Function Codes	10-7
	10-6	TAPOP. Error Codes	10-10
	11-1	Terminals	11-1
	11-2	TTCALL Functions	11-2
	11-3	Terminal Line Characteristics	11-4
	11-4	TRMOP. Function Codes	11-6
	11-5	Transmit/Receiving Speeds	11-9
	11-6	TRMOP. Error Codes	11-9
	11-7	JOB Status Bits	11-17
	12-1	Summary of Some Non-Directory Devices	12-1
	13-1	CNECT. Operation Codes	13-3
	13-2	CNECT. Error Codes	13-3
	13-3	ERLST. Error Codes	13-4
	13-4	SENSE. Error Codes	13-5
	13-5	CLRST. Error Codes	13-6
	14-1	RTTRP Error Codes	14-3

TABLES (Cont.)

			Page
TABLE	15-1	Packet Flags	15-3
	15-2	IPCF Capabilities of a Sender	15-5
	15-3	[SYSTEM] INFO Functions	15-7
	15-4	[SYSTEM] IPCC Functions	15-8
	15-5	IPCF Error Codes	15-12
	16-1	ENQ. Function Codes	16-7
	16-2	DEQ. Function Codes	16-8
	16-3	Current State of Resource	16-10
	16-4	ENQC. Function Codes	16-11
	16-5	ENQC. Flags	16-12
	16-6	Enqueue/Dequeue Error Codes	16-13
	17-1	METER. Function Codes	17-4
	17-2	METER. Error Codes	17-5
	18-1	GETTAB Tables	18-2
	18-2	Privilege Table (.GTPRV, GETTAB Table Number 6)	18-6
	18-3	Configuration Table (.GTCNF, GETTAB Table Number 11)	18-6
	18-4	System State Bit Settings (Item Number 17 in GETTAB Table Number 11)	18-11
	18-5	Non-swapping Data Table (.GTNSW, GETTAB Table Number 12)	18-12
	18-6	Swapping Data Table (.GTSDT, GETTAB Table Number 13)	18-13
	18-7	ONCE-ONLY Disk Parameters (.GTODP, GETTAB Table Number 15)	18-14
	18-8	LEVEL-D Parameters (.GTLVD, GETTAB Table Number 16)	18-14
	18-9	GETTAB Immediate Word Entries (.GTSLF, GETTAB Table Number 23)	18-16
	18-10	Spooling Control Table (.GTSPL, GETTAB Table Number 36)	18-17
	18-11	Time and Batch Status Table (.GTLIM, GETTAB Table Number 40)	18-17
	18-12	Hardware Status After a Crash (.GTCRS, GETTAB Table Number 44)	18-18
	18-13	System Wide Data Table (.GTSYS, GETTAB Table Number 51)	18-18
	18-14	CPU0 Control Data Block Constants Table (.GTC0C, GETTAB Table Number 55)	18-19
	18-15	CPU0 Control Data Block Variable Table (.GTC0V, GETTAB Table Number 56)	18-20
	18-16	Response Subtable	18-21
	18-17	Parity Subtable	18-22
	18-18	Feature Table (.GTFET, GETTAB Table Number 71)	18-23
	18-19	Scanner Table (.GTSCN, GETTAB Table Number 73)	18-26
	18-20	SEND-ALL Text (.GTSND, GETTAB Table Number 74)	18-26
	18-21	IPCF Miscellaneous Data (.GTIPC, GETTAB Table Number 77)	18-26
	18-22	IPCF Statistics Per Job (.GTIPA, GETTAB Table Number 104)	18-26
	18-23	IPCF Flags and Quotas (.GTIPQ, GETTAB Table Number 107)	18-27
	18-24	General Virtual Memory Data (.GTVM, GETTAB Table Number 113)	18-27
	18-25	Scheduler Statistics (.GTSST, GETTAB Table Number 115)	18-28
	18-26	Special PID Table (.GTSID, GETTAB Table Number 126)	18-28
	18-27	ENQ./DEQ. Statistics (.GTENQ, GETTAB Table Number 127)	18-29
	A-1	Disk Devices	A-1
	B-1	Magnetic Tape Systems	B-1
	C-1	ASCII Card Codes	C-1
	C-2	ASCII Codes and BCD Equivalents	C-3
	D-1	Terminals	D-1
	E-1	Error Codes	E-1
	F-1	DECsystem-10	F-1
	G-1	Core Memories	G-1
	I-1	Communication Systems	I-1
	K-1	Conversion of Characters	K-2
	K-2	Model 2741 Typing Elements	K-3

ALPHABETICAL LIST OF MONITOR CALLS

APRENB, 5-1
ATTACH, UUOPRV

CAL11., UUOPRV
CHKACC, 8-30
CHGPPN, UUOPRV
CLOSE, 8-44
  for DECtape, 9-8
  for CDP, 12-2
  for LPT, 12-9
  for Display, 12-6
  for PTP, 12-10
  for Plotter, 12-13
CLRST., 13-5
CNECT., 13-2
CORE, 6-8
CTLJOB, 11-17

DAEFIN, UUOPRV
DAEMON, 4-12, UUOPRV
DATE, 4-11
DEBRK., 5-12
DEQ., 16-7
DEVCHR, 7-15
DEVLNM, 7-14.1
DEVNAM, 7-21
DEVPPN, 8-34
DEVSIZ, 7-16
DEVSTS, 7-15
DEVTYP, 7-16
DISK., 8-39
DSKCHR, 8-36
DVRST., UUOPRV
DVURS., UUOPRV

ENQ., 16-5
ENQC., 16-9
ENTER, 8-11
  for DECtape, 9-7
ERLST., 13-3
ERRPT., UUOPRV
EXIT, 4-2

FILOP., 8-22
FRUCCO, UUOPRV

GETLIN, 11-4
GETPPN, 4-9
GETSEG, 4-1
GETSTS, 8-41
GETTAB, 18-1
GOBSTR, 8-33

HALT, 4-1
HIBER, 4-2
  for pseudo-TTYs, 11-15
HPQ, 14-7

IN, 8-21
  for Display, 12-6
  for TTY, 11-16
INBUF, 7-11
INIT, 7-3
INPUT, 8-21
  for DECtape, 9-8
  for TTY, 11-16
IONDX., 13-6
IPCFQ., 15-10
IPCFR., 15-11
IPCFS., 15-11

JBSET, UUOPRV
JOBPEK, UUOPRV
JOBSTR, 8-32
JOBSTS, 11-16

LIGHTS, 4-11
LOCATE, 4-8
LOCK, 6-1
LOGIN, UUOPRV
LOGOUT, 4-2, UUOPRV
LOOKUP, 8-10
  for DECtape, 9-5

METER, 17-1
MSTIME, 4-11
MTAID., 10-12, UUOPRV
MTAPE, 10-3
  for DECtape, 9-9
MTCHR., 10-5
MVHDR., 13-6

NODE, 7-19

OPEN, 7-3
OTHUSR, 4-10
OUT, 8-21
  for TTY, 11-16
OUTBUF, 7-11
OUTPUT, 8-21
  for TTY, 11-16
  for display, 12-6
  for DECtape, 9-8

PAGE., 6-11
PATH., 8-22.2

PEEK, 4-13
PIINI., 5-8
PIRST., 5-12
PISAV., 5-11
PISYS., 5-9
PJOB, 4-9
POKE, 4-14

REASSIGN, 7-14.1
RELEASE, 7-13
  for TTY, 11-16
  for Display, 12-6
  for DECtape, 9-8
RENAME, 8-8
  for DECtape, 9-7
REMAP, 6-19
RESDV., 7-13
RESET, 7-1
RTTRP., 14-1
RUN, 6-16
RUNTIME, 4-9

SCHED., UUOPRV
SEEK, 8-29
SENSE., 13-4
SETNAM, 4-4
SETSTS, 8-43
SETUUO, 4-5, UUOPRV
SETUWP, 6-10
SLEEP, 4-2
SPY, 4-14
STATO, 8-42
STATZ, 8-43
STRUUUO, UUOPRV
SUSET., UUOPRV
SWITCH, 4-11
SYSPHY, 8-34
SYSSTR, 8-33

TAPOP., 10-6.1
TIMER, 4-11
TMPCOR, 4-14
TRMNO, 11-5
TRMOP., 11-5
TRPSET, 14-5
TTCALL, 11-1

UGETF, 9-9
UJEN, 14-7
UNLOK., 6-8

ALPHABETICAL LIST OF MONITOR CALLS (Cont.)

USETI, 8-27
for DECtape, 9-9
USETO, 8-27
for DECtape, 9-9
UTPCLR, 9-9

WAIT, 7-10
WAKE, 4-3
WHERE, 7-19

CHAPTER 1
USER PROGRAMMING

1.1 PROCESSOR MODES

In a single-user, non-timesharing environment, a user's program is limited only by those conditions inherent in the hardware. The program must

1. stay within the memory capacity,
2. observe the hardware restrictions placed on the use of certain memory locations, and
3. observe the restrictions on interrupt conditions.

In a timesharing environment, the hardware limits the central processor to one of three modes:

1. user mode, which on a KI10/KL10 is divided into concealed and public modes,
2. user I/O mode, and
3. executive mode, which on the KI10/KL10 is divided into kernal mode and supervisor mode.

1.1.1 User Mode Processing

The processor is usually in *user mode* when user programs are executed. In this mode, user programs must operate within an assigned area of core and certain instructions are illegal. The illegal instructions are:

the op codes 700 through 777[1],
a JRST 10, instruction,
a JRST 4, instruction,
unimplemented op codes,
and, on the KL10, any JRST except JRST 0, JRST 1, or JRST 2.

All illegal instructions trap to the monitor, stop the program, and print one of the following messages on the user's terminal.

?HALT AT USER PC *addr*
?ILLEGAL INSTRUCTION AT USER PC *addr*
?KI10 or KL10 INSTRUCTION AT USER PC *addr*
?KL10 ONLY INSTRUCTION AT USER PC *addr*

The CONT and CCONT commands can be used to continue the execution of the user program only after a HALT.

User mode processing is used to guarantee the integrity of the monitor and each user program. The user mode of the processor is characterized by the following conditions.

1. Automatic memory protection and mapping, refer to Chapter 3.
2. Trap to the monitor on any of the following:

   op codes 040 through 077 and op code 00,
   I/O instructions (DATAI, DATAO, BLKI, BLKO, CONI, CONO, CONSZ, and CONSO),[1]
   HALT (i.e., JRST 4,)
3. Trap to location 40 in the user area on the execution of op codes 001 through 037.

[1]except I/O instructions using a device code greater then 734 on a KI10/KL10.

1.1.2 User I/O Mode Processing

The user I/O mode (i.e., bits 5 and 6 of the PC word equal to 11_2) of the central processor allows privileged user programs to be executed with automatic protection and mapping in effect, as well as the normal execution of all defined operation codes (except the HALT instruction on the KI10/KL10 processors.) User I/O mode provides some protection against partially debugged monitor routines and it provides infrequently used device service routines to be executed as user jobs. Direct control of special devices, which is particularly important in real-time applications, may be obtained by a user program.

To utilize user I/O mode, the system administrator must have set bit 15 (JP.TRP) in the privilege word (refer to .GTPRV, GETTAB Table Number 6). All user I/O mode activities are terminated by the execution of the RESET monitor call. User I/O mode is not used by the monitor, and it is not, normally, available to the unprivileged time-sharing user.

1.1.3 Executive Mode Processing

The monitor operates with the processor in executive mode, which is characterized by special memory protection and mapping (refer to Chapter 3) and by the normal execution of all defined op codes.

When user programs execute in user mode, the monitor schedules user programs, services the interrupts, performs all input/output operations, takes action when control returns from a user program, and performs any other legal user-requested operations not available in user mode. Monitor services and how a user program obtains these services are described in Chapters 4 and 5.

1.2 MONITOR CALLS (PROGRAMMED OPERATORS)

Operation codes 000 through 100 on the DECsystem-10 are monitor calls. These are sometimes referred to as programmed operators (UUOs). Monitor calls are software-implemented instructions. Their functions, from a hardware point of view, are not prespecified. Some monitor calls trap to the monitor; others trap to the user program.

When a monitor call is executed:

the effective address is calculated,
the contents of the instruction register (along with the effective address) is stored,
and an instruction is executed out of the normal sequence of operations.

Refer to the *DECsystem-10 System Reference Manual* for further details on Monitor Call handling by the central processor.

1.2.1 Op Codes 000-037

Op codes 000 through 037 *do not* affect the mode of the central processor. When these op codes are executed in user mode, they trap to location 40. This allows the user program complete freedom when using these operators.

If an undebugged program executes one of these op codes accidentally, the following message is printed:

HALT AT USER PC *addr*

where: *addr* is the location of the user's monitor call.

The message is typed because the users *relative* location 41 contains a HALT instruction, unless the user program overtly changed it. This HALT instruction is provided by LINK-10.

1.2.2 Op Codes 040-100 and 000

Op codes 040 through 100 trap to *absolute* location 40 on KA10-based systems. On KI10/KL10-based systems, the call is stored at location 424, and the new pc is loaded from location 436 of the user's process table (the central processor operates in executive mode). These monitor calls are interpreted by the monitor to perform I/O operations and other control functions for the user program.

Op code 000 always returns to monitor mode, and the following error message is printed:

?ILLEGAL UUO AT USER PC *addr*

Table 1-1 lists the op codes 040 through 100, their mnemonics, and their meanings.

Table 1-1
Op Codes 040-100

Op Code	Call	Meaning
040	{**CALL** *ac,* [SIXBIT/*name*/] {*name ac,* }	Programmed operator extension. (Refer to Table 1-2).
041	**INIT** *channel, status* SIXBIT/*device*/ XWD *obuf, ibuf* *error return* *normal return*	Initialize a device and associate it with an I/O channel.
042-046		Reserved for installation dependent definition.
047	**CALLI** *ac, n*	Programmed operator extension.
050	**OPEN** *channel, spec* *error return* *normal return* . . . *spec:* EXP *status* SIXBIT/*device*/ XWD *obuf, ibuf*	Initialize a device, and associate it with an I/O channel.
051	**TTCALL** *ac, addr*	Extended operations on job-controlling terminals.
052-054		Reserved to Digital for future expansion.
055	**RENAME** *channel, addr* *error return* *normal return* . . . *addr:* SIXBIT/*filename*/ SIXBIT/*extension*/	Rename or delete a file on a non-directory device.
	addr: SIXBIT/*filename*/ SIXBIT/*extension*/,,*hi-date2* 0,,*low-date*	DECtape.
	addr: SIXBIT/*filename*/ SIXBIT/*ext*/,*hi-date, date1* *prot, mode, time, low-date2* *ppn*	disk unit.
056	**IN** *channel, addr* *normal return* *error return*	Transmit data from a file to a user's core area, skip on an error or an EOF.

Table 1-1 (Cont.)
Op Codes 040-100

Op Code	Call	Meaning
057	**OUT** *channel, addr* *normal return* *error return*	Transmit data from a user's core area to a file, skip in an error or an EOF.
060	**SETSTS** *channel, addr* *return*	Change the file status.
061	**STATO** *channel, status* *normal return* *alternate return*	Skip if any file status bits are equal to one.
062	**GETSTS** *channel, addr* *return*	Copy file status to *addr*.
063	**STATZ** *channel, status* *normal return* *alternate return*	Skips if all status bits are zero.
064	**INBUF** *channel, n* *return*	Set up input buffer ring with *n* buffers.
065	**OUTBUF** *channel, n* *return*	Set up output buffer ring with *n* buffers.
066	**INPUT** *channel, addr* *return*	Transmit data from a file to the user's core area.
067	**OUTPUT** *channel, addr* *return*	Transmit data from a file to the user's core area.
070	**CLOSE** *channel, option* *return*	Terminate file operations.
071	**RELEASE** *channel,* *return*	Release a device.
072	**MTAPE** *channel, function* *return*	Perform tape positioning operations.
073	**UGETF** *channel, addr* *return*	Get next free block number on DECtape.
074	**USETI** *channel, block* *return*	Set next input block number on disk or DECtape.
075	**USETO** *channel, block* *return*	Set next output block number on disk or DECtape.
076	**LOOKUP** *channel, addr* *error return* *normal return* . . . *addr:* SIXBIT/*filename*/ SIXBIT/*ext*/	Select a file for input on a non-directory device
	addr: SIXBIT/*filename*/ SIXBIT/*ext*/, *high date2* *prot, mode, time, date* XWD *proj, prog*	directory device.

Table 1-1 (Cont.)
Op Codes 040-100

Op Code	Call	Meaning
077	**ENTER** *channel, addr* *error return* *normal return* . . . *addr:* SIXBIT/*filename*/ SIXBIT/*extension*/	Select a file for output on a non-directory device.
	addr: SIXBIT/*filename*/ SIXBIT/*ext*/, *date* 0,,*date*	DECtape.
	addr: SIXBIT/*filename*/ SIXBIT/*ext*/, *date, date* *prot, mode, time, date* XWD *proj, prog*	disk unit.
100	**UJEN** *return*	Dismiss a real-time interrupt.

Table1 -2
CALLIs/Monitor Calls

CALLI #	Call	Function
-2, . . .	Customer defined.	Reserved for customer definition.
-1	**LIGHTS** *ac,* *only return*	Display the contents of the AC in the lights.
1	**RESET** *ac,* *return*	Reset an I/O device.
2	MOVEI *ac, start-addr* **SETDDT** *ac,* *return*	Set the protected DDT starting address.

4	{MOVE *ac,* [SIXBIT/*device*/]
	MOVE *ac, channel*
	MOVEI *ac, udx*}
	DEVCHR *ac,*
	return
		Get device characteristics.

10	{MOVEI *ac, channel* MOVEI *ac, udx*} **WAIT** *channel* *return*	Wait until device is inactive.
11	MOVE *ac,* [XWD *hi, lo*] **CORE** *ac,* *error return* *normal return*	Allocate core.
12	**EXIT** *ac,* *return here on continue*	Reset is performed when AC = 0, job is stopped when AC not equal to 0.
13	{MOVEI *ac, channel* MOVEI *ac, udx*} **UTPCLR** *ac,* *only return*	Clear a DECtape directory.
14	**DATE** *ac,* *only return*	Return the date.
15[1]	MOVE *ac,* [XWD *-n, loc*] **LOGIN** *ac,* *only return*	Privileged call available only to system privileged programs. It is a no-op if executed by a job already logged in.
16	MOVEI *ac, bits* **APRENB** *ac,* *return*	Enable central processor traps.
17	**LOGOUT** *ac,* *no return*	Privilege monitor call available only to system privileged programs. It is treated as an EXIT monitor call if executed by a non-system privileged program.
20	**SWITCH** *ac,* *return*	Read the console data switches.

Table 1-2 (Cont.)
CALLIs/Monitor Calls

CALLI #	Call	Function

21	MOVEI *ac,job-number*
	{MOVE *ac+1,* [SIXBIT/*device*/]
	MOVEI *ac+1,channel*
	MOVEI *ac+1,udx*}
	REASSIGN *ac,*
	only return
		Reassign a device.

22	**TIMER** *ac,* *only return*	Return the time of day in jiffies.
23	**MSTIME** *ac,* *only return*	Return the time of day in milliseconds.
24	**GETPPN** *ac,* *normal return* *alternate return*	Return the project-programmer number of the current job.
25	MOVE *ac,* [XWD *n,loc*] **TRPSET** *ac,* *error return* *normal return* . . . *loc:* JSR TRAP	Set a trap for user I/O mode.
26		Illegal monitor call.
27	{MOVEI *ac,job number* MOVEI *ac, 0*} **RUNTIM** *ac,* *only return*	Return the job's running time in milliseconds. (0 indicates the current job).
30	**PJOB** *ac,* *only return*	Return the job number.
31	MOVEI *ac, secs. to sleep* **SLEEP** *ac,* *only return*	Stop a job for a specified number of seconds (68 seconds is the maximum).
32		Historical monitor call.
33	MOVEI *ac,addr* **PEEK** *ac,* *only return*	Return the contents of a specified exec address.
34	**GETLIN** *ac,* *only return*	Return the SIXBIT physical name of the terminal that the current job is attached to.
35	MOVSI *ac, start-addr-inc* HRRI *ac,loc* **RUN** *ac,* *error return* *normal return* ; @.JBSA . . .	Allow programs to transfer control to one another. Both the low and the high segments of the user's addressing space are replaced with the program being called.

Table 1-2 (Cont.)
CALLIs/Monitor Calls

CALLI #	Call	Function
	loc: SIXBIT/*device*/ SIXBIT/*extension*/ SIXBIT/*filename*/ 0 XWD *proj,prog* XWD *hi-addr, core*	
36	MOVEI *ac, bit* **SETUWP** *ac,* *error return* *normal return*	Set or clear user mode write protect for high segment.
37	{MOVEI *ac, hi in low seg* MOVE *ac,* [XWD *hiorigin, lo*]} **REMAP** ac, *error return* *normal return*	Remap top of low segment into the high segment.
40	MOVEI *ac, addr* **GETSEG** *ac,* *error return* *normal return* . . . *addr:* SIXBIT/*device*/ SIXBIT/*filename*/ SIXBIT/*ext*/ 0 XWD *proj,prog* 0	Replace the high segment in user's addressing space.
41	{MOVSI *ac, job-number* MOVSI *ac, index-number*} HRRI *ac, table-number* **GETTAB** *ac,* *error return* *normal return*	Return contents of monitor table or location.
42	MOVEI *ac, hi-phys-addr* **SPY** *ac,* *error return* *normal return*	Make physical core assignment for examination of monitor.
43	MOVE *ac,* [SIXBIT/*name*/] **SETNAM** *ac,* *only return*	Set program name in monitor table.
44	MOVE *ac,* [XWD *code, block*] **TMPCOR** *ac,* *error return* *normal return* . . .	Allow temporary in-core file storage for the job.

Table 1-2 (Cont.)
CALLIs/Monitor Calls

CALLI #	Call	Function
	block: XWD *name, 0* IOWD *bulen, buffer*	
45	MOVEI *ac,* [XWD *+n, loc*] **DSKCHR** *ac,* *error return* *normal return* . . . *loc:* SIXBIT/*diskname*/	Return disk characteristics.
46	MOVEI *ac,* 0 MOVE *ac,* [SIXBIT/*fsname*/] **SYSSTR** *ac,* *error return* *normal return*	Return all of the file structures names in the system.
47	MOVE *ac,* [XWD *n, loc*] **JOBSTR** *ac,* *error return* *normal return*	Return next file structure name in job search list.
50	MOVE *ac,* [XWD *n, loc*] **STRUUO** *ac,* *error return* *normal return* . . . *loc:* *function* . . . *argn*-1	Manipulate the structures.
51	MOVEI *ac,* 0 MOVE *ac,* [SIXBIT/*name*/] **SYSPHY** *ac,* *error return* *normal return*	Return all physical disk units.
52		Reserved for future use.
53	{ MOVE *ac,* [SIXBIT/*device*/] MOVEI *ac, channel* MOVEI *ac, udx* } **DEVTYP** *ac,* *error return* *normal return*	Return properties of a device.
54	{ MOVEI *ac, channel* MOVE *ac,* [SIXBIT/*device*/] MOVEI *ac, udx* } **DEVSTS** *ac,* *error return* *normal return*	Remove hardware device status word.

Table 1-2 (Cont.)
CALLIs/Monitor Calls

CALLI #	Call	Function
55	{MOVE *ac*, [SIXBIT/*device*/] {MOVEI *ac*, *channel* **DEVPPN** *ac*, *error return* *normal return*	Return the project-programmer word.
56	**SEEK** *channel*, *return*	Perform a SEEK to current selected block for software channel AC.
57	MOVEI *ac*, *loc* **RTTRP** *ac*, *error return* *normal return*	Connect real-time device to the PI system.
60	{MOVE *ac*, [XWD *hi*, *lo*] {MOVE *ac*, [XWD -*n*, *addr*] **LOCK** *ac*, *error return* *normal return*	Lock job in core.
61	{MOVEI *ac*, *channel* {MOVEI *ac*, *job-number* {MOVEI *ac*, *udx* **JOBSTS** *ac*, *error return* *normal return*	Return status information about device TTY and/or controlled job.
62	MOVEI *ac*, *station-number* **LOCATE** *ac*, *error return* *normal return*	Change the job's logical station/node number.
63	{MOVEI *ac*, *channel* {MOVE *ac*, [SIXBIT/*device*/] {MOVEI *ac*, *udx* **DEVNAM** *ac*, *error return* *normal return*	Return physical name of device obtained through generic INIT/OPEN/FILOP. or logical device assignment.
65	MOVEI *ac*, *job-number* **CTLJOB** *ac*, *error return* *normal return*	Return job number of controlling job.
66	MOVE *ac*, [XWD *n*, *loc*] **GOBSTR** *ac*, *error return* *normal return* . . . *loc:* *job-number* XWD *proj*, *prog* SIXBIT/*name*/ -1 0 *status-bits*	Return next file structure name in an arbitrary job's search list.

Table 1-2 (Cont.)
CALLIs/Monitor Calls

CALLI #	Call	Function
67		Reserved to Digital.
70		Reserved to Digital.
71	MOVEI *ac, hpq* **HPQ** *ac,* *error return* *normal return*	Place job in high priority scheduler's run queue.
72	MOVSI *ac, enable bits* HRRI *ac, sleep time* **HIBER** *ac,* *error return* *normal return*	Allow job to become dormant until the specified event occurs.
73	MOVEI *ac, job-number* **WAKE** *ac,* *error return* *normal return*	Activate the specified dormant job.
74	MOVE *ac,* [XWD *proj, prog*] **CHGPPN** *ac,* *error return* *normal return*	Change project-programmer number.
75	MOVE *ac,* [XWD *function, arg*] **SETUUO** *ac,* *error return* *normal return*	Set system and job parameters.
76		Reserved for Digital.
77	**OTHUSR** *ac,* *non-skip return* *skip return*	Determine if another job is logged into the same project-programmer number. If one is not, take the non-skip return.
100	MOVEI *ac, loc* **CHKACC** *ac,* *error return* *normal return* . . . *loc:* *access, dir-prot, file-prot* *directory ppn* *user ppn*	Check user's access to the file specified.

Table 1-2 (Cont.)
CALLIs/Monitor Calls

CALLI #	Call	Function
101	MOVEI *ac, loc* **DEVSIZ** *ac,* *error return* *normal return* *loc:* EXP *data mode* {SIXBIT/*device*/ / channel / udx}	Determine buffer size.
102	MOVE *ac,* [XWD *length, addr*] **DAEMON** *ac,* *error return* *normal return* . . . *addr:* *function* *arg1* . . . *argn*	Request DAEMON to perform a specified function.
103	MOVEI *ac,addr* **JOBPEK** *ac,* *error return* *normal return*	Read or write another job's core.
104	MOVE *ac,* [XWD *line# ,job #]* **ATTACH** *ac,* *error return* *normal return*	Attach the job to the specified TTY line number.
105	MOVE *ac,* [XWD *length,addr*] **DAEFIN** *ac,* *error return* *normal return*	Indicate that the request to the DAEMON program has been completed.
106	MOVE *ac,* [XWD *length, addr*] **FRCUUO** *ac,* *error return* *normal return*	Force a command for a job.

Table 1-2 (Cont.)
CALLIs/Monitor Calls

CALLI #	Call	Function

107	MOVE *ac*, [SIXBIT/*device*/]
	MOVEI *ac, channel*
	MOVEI *ac, udx*
	MOVE *ac+1*, [SIXBIT/*logical*/]
	DEVLNM *ac*,
	error return
	normal return
		Set a logical name for this device.

110	MOVE *ac*, [XWD *length, addr*] **PATH** *ac*, *error return* *normal return* . . . *addr:* SIXBIT/*name*/ *scan switch* XWD *proj, prog* *sfd name* *sfd name* . . .	Read or modify the default directory path or read the current path of a file OPEN on a channel, or set and/ or test the additional path.

Table 1-2 (Cont.)
CALLIs/Monitor Calls

CALLI #	Call	Functiom
111	MOVE *ac*, [XWD *n+1, loc*] **METER.** *ac*, *error return* *normal return* . . . *loc:* *function* *arg1* . . . *argn*	Provide performance analysis and metering of dynamic system variables.

112	MOVE *ac*, [XWD *n, loc*]
	MOVEI *ac, channel*
	MOVEI *ac, udx*
	MOVE *ac*, [SIXBIT/*device*/]
	MTCHR. *ac*,
	error return
	normal return
		Return characteristics of a magnetic tape.

113	MOVE *ac*, [XWD 2, *block*] **JBSET.** *ac*, *error return* *normal return* . . . *block:* XWD 0, *job #* XWD function, value	Execute the specified function of SETUUO for a job.
114	MOVE *ac*, [XWD 3, *loc*] **POKE.** *ac*, *error return* *normal return* . . . *loc:* *location* *old value* *new value*	Alter the specified location in the monitor.
115	MOVEI *ac, job#* **TRMNO.** *ac*, *error return* *normal return*	Return the number of the terminal currently controlling the specified job.
116	MOVE *ac*, [XWD *n, addr*] **TRMOP.** *ac*, *error return* *normal return*	Perform miscellaneous terminal functions.

Table 1-2 (Cont.)
CALLIs/Monitor Calls

CALLI#	Call	Function

117	MOVEI *ac, channel*
	MOVEI *ac, udx*
	RESDV. *ac,*
	error return
	normal return
		Reset the specified channel.

120	MOVSI *ac*, 1
	MOVSI *ac*, 0
	HRRI *ac*, 1
	HRRI *ac*, 0
	UNLOK. *ac,*
	error return
	normal return
		Unlock a locked job in core.

121	MOVE *ac*, [XWD *function,addr*] **DISK.** *ac,* *error return* *normal return*	Set or read a disk or file system parameter.

122	MOVE *ac*, [SIXBIT/*device*/]
	MOVEI *ac, channel*
	MOVEI *ac, udx*
	DVRST. *ac,*
	error return
	normal return
		Restrict the specified device to a privileged job.

123	MOVE *ac*, [SIXBIT/*device*/]
	MOVEI *ac, channel*
	MOVEI *ac, udx*
	DVURS. *ac,*
	error return
	normal return
		Remove the restricted status of a specified device.

124		Reserved to Digital.
125	MOVE *ac*, [XWD *n,addr*] **CALL11.** *ac,* *error return* *normal return*	Front-end debug monitor call.

126	MOVE *ac*, [SIXBIT/*device*/]
	MOVEI *ac, udx*
	MOVEI *ac, channel*
	MOVE *ac+1*, [SIXBIT/*reelid*/]
	MTAID. *ac,*
	error return
	normal return
		Associate a visual identification with a magnetic tape drive during a mount.

127	MOVEI *ac, channel*
	MOVE *ac*, [SIXBIT/*device*/]
	IONDX. *ac,*
	error return
	normal return
		Return the universal I/O index for a device.

Table 1-2 (Cont.)
CALLIs/Monitor Calls

CALLI#	Call	Function
130	MOVEI *ac, list* **CNECT.** *ac,* *error return* *normal return*	Connect/disconnect devices to/from an MPX channel.
131	{MOVEI *ac, channel* {MOVE *ac,* [*outaddr, inaddr*]} **MVHDR.** *ac,* *error return* *normal return*	Move a buffer ring header between core locations.
132	MOVEI *ac, addr* **ERLST** *ac,* *error return* *normal return* . . . *addr:* *#words, channel* *#devices* *UDX,, GETSTS*	Provide a list of non-operational devices connected to an MPX channel.
133	MOVE *ac,* [XWD *length, addr*] **CLRST.** *ac,* *error return* *normal return* . . . *addr:* *udx* *channel* SIXBIT/*device*/ 0,*SETSTS value* . . .	Allow a device to continue after a device error condition has occurred.
135	MOVE *ac, base-addr* **PIINI.** *ac,* *error return* *normal return*	Initialize the software interrupt system.
136	MOVE *ac,* [XWD *flags, addr*] **PISYS.** *ac,* *error return* *normal return*	Control the software interrupt system.
137	**DEBRK.** *return*	Dismiss a software interrupt.
140	MOVE *ac,* [XWD *size, addr*] **PISAV.** *ac,* *error return* *normal return*	Save the state of the software interrupt system.

Table 1-2 (Cont.)
CALLIs/Monitor Calls

CALLI#	Call	Function
141	MOVEI *ac,addr* **PIRST.** *ac,* *error return* *normal return*	Restore the state of the software interrupt system.
142	MOVE *ac,* [XWD *n,loc*] **IPCFR.** *ac,* *error return* *normal return* . . . *loc:* *flags* *sender's PID* *receiver's PID* *message length,,addr*	Receive an IPCF packet.
143	MOVE *ac,* [XWD *n,loc*] **IPCFS.** *ac,* *error return* *normal return* . . . *loc:* *flags* *sender's PID* *receiver's PID* *message length,,addr*	Send an IPCF packet.
144	MOVE *ac,* [XWD *n,loc*] **IPCFQ.** *ac,* *error return* *normal return* . . . *loc:* *flags* *sender's PID* *receiver's PID* *message length,,addr*	Obtain information about an IPCF input queue.
145	MOVE *ac,* [XWD *function,loc*] **PAGE.** *ac,* *error return* *normal return*	Manipulate pages and the data associated with these pages.
146	MOVE *ac,* [XWD *n,loc*] **SUSET.** *ac,* *error return* *normal return*	Set the next I/O block number.

1.2.2.1 Symbols for Numbers – Some system programs, e.g., LOGOUT, require I/O to specific physical devices regardless of the logical name assignments. Therefore, for any monitor call, if bit 19 (UU.PHS) in the effective address of the call is not equal to bit 18, *only* physical device for any names will be used; logical device assignments will be ignored. This suppression of logical device names is helpful, for example, when using the results of the DEVNAM monitor call where the physical name corresponding to a logical name is returned.

1.2.2.2 Restriction on Monitor Calls in Programs – A number of restrictions on monitor calls that involve a high segment prevent naive or malicious users from interfering with other users while sharing segments and minimize monitor overhead in handling two-segment programs. The basic rules are as follows:

1. All monitor calls can be executed from *either* the low or high segment although some of their arguments cannot be in or refer to the high segment.
2. No buffers or buffer headers may exist in the high segment for reading from or writing to any I/O device.
3. No I/O is processed into or out of the high segment except via the SAVE, OSAVE, NSAVE and SSAVE, OSSAVE, NSSAVE commands.
4. As a convenience in writing user programs, the monitor makes a special check so that the INIT monitor call can be executed from the high segment, although the calling sequence is in the low segment. The monitor also allows the effective address of the OPEN monitor call (which contains the status bits, device name, and buffer header addresses) in the high segment. The address of TTCALL 1, and TTCALL 3, may be in the high segment for convenience in typing messages.

1.2.3 Operation Codes 100-127

OP Code 100 (UJEN)	Dismiss real-time interrupt from user mode.
OP Codes 101-106	Monitor prints ?ILLEGAL INSTRUCTION AT USER *n* and stops the job.
107,114-117	These op codes are valid only on the KL10. The message ?KL10 ONLY INSTRUCTION will be printed if an attempt is made to execute them on any other system.
OP Codes 110-113	These op codes are valid on the KI10 or the KL10. If used on the KA10, the monitor prints ?KI10 or KL10 INSTRUCTION AT USER *n* and stops the job.

1.2.4 Illegal Operation Codes

The eight I/O instructions (e.g., DATAI) and JRST instructions with bit 9 or 10 equal to 1 (e.g., HALT, JEN) are interpreted by the monitor as illegal instructions (refer to the *DECsystem-20 SYSTEM Reference Manual*). The job is stopped and a question mark is printed immediately. A carriage-return/line-feed sequence is performed, followed by an error message. For example, a DATAI instruction would produce the following:

```
?
?ILLEGAL INSTRUCTION AT USER addr
```

1.2.5 Naming Convention for Monitor Symbols

The names of the monitor's data base symbols contain dots or percent signs so that they can be made user-mode symbols without conflicting with previously-coded user programs. Data symbols can be divided into five classes:

1. numbers
2. masks
3. monitor call names
4. GETTAB arguments
5. error codes

1.2.5.1 Symbols for Numbers – Symbols defining numbers begin with a dot, followed by a 2-letter prefix indicating the type of number, and end with a 3-character abbreviation representing the specific number. Numbers are 18-bit quantities and include core addresses and function codes. The following are examples of various numbers:

.JB*xxx*	Job Data Area
.GT*xxx*	GETTAB Table Numbers
.RB*xxx*	Extended arguments for LOOKUP, ENTER, RENAME monitor calls

1.2.5.2 Symbols for Masks – Names for masks start with a 2-letter prefix indicating the individual word, followed by a dot, and end with 3 characters representing the specific mask. Masks are 36-bit quantities that include bits and fields. The following are examples of names of masks:

JP.*xxx*	Privilege word bits
JW.*xxx*	WATCH word bits
PC.*xxx*	PC word bits

1.2.5.3 Symbols for Monitor Calls – Names for monitor calls implemented after the 5.03 release of the monitor are 5 or fewer characters followed by a dot. For example:

PATH.	The monitor call to modify a directory path
TRMOP.	The monitor call to perform terminal operations

1.2.5.4 Symbols for GETTAB Tables – Individual words within a GETTAB table start with a percent sign, followed by 2 characters representing the generic name of the table, and end with 3 characters identifying the specific word. For example:

%NSCMX	CORMAX word in the nonswapping data table
%CNSTS	The states word in the configuration table

Names of bytes and bits within a GETTAB table begin with 2 characters representing the word, followed by a percent sign, and end with 3 characters designating the specific byte.

ST%DSK	Byte representing the disk system; this is contained in the states word
ST%SWP	Byte indicating a swapping system; this is contained in the states word

1.2.5.5 Symbols for Error Codes – Error codes returned because of an monitor call error have names following the pattern: 2 characters indicating the call name, three characters designating the failure type, and a terminating percent sign. The following are examples of error code symbols:

DMILF%	DAEMON error code
RTDIU%	RTTRP error code
LKNLPL%	LOCK error code

Many of the values useful in user programming are encoded in the parameter file UUOSYM.MAC (refer to Appendix J) for the convenience of writing and modifying programs.

CHAPTER 2
MEMORY FORMAT

2.1 USER PROGRAMS

User programs must be loaded into core memory before they can be executed. There are two methods of loading a user program into core. The simpler method is to load a core image stored on a retrievable device (refer to the RUN and GET commands in the *DECsystem-10 Operating System Commands Manual).* The other method is to load a collection of relocatable binary (.REL) files using LINK-10.

The address space of a user program can be divided into two segments, high and low; all user programs must have at least a low segment. A user program can have one of several file name extensions, depending on its file type and how it was stored.

.EXE	The file is an exact copy of the user's core image.
.SAV	The file is a low segment file.
.LOW .HGH	The file has two segments: a low segment (.LOW) and a high segment (.HGH).
.LOW .SHR	The file has two segments: a low segment and a sharable high segment (.SHR).

The low segment is always used by one individual program, and each user program has its own low segment.

By default, the monitor will write-protect the high segment, preventing a user from altering it. Any suitably privileged program can clear the write-protect bit of a high segment. This clearing to modify the write-protect bit is necessary for debugging the high segment.

A high segment can be shared by any number of jobs that have unique low segments. For instance, if five users have low segments containing their own FORTRAN user programs, they may share a high segment containing the FORTRAN compiler. The monitor performs this sharing function automatically; each user believes that he has his own high segment containing the FORTRAN compiler, and is unaware of sharing the high segment with the others.

Any user job attempting to modify a write-protected high segment will be aborted and will receive an error message. If the user's job has two segments and he has asked the monitor to clear the high segment's write-protection, his job will be a two-segment writable job.

All user programs are assembled and loaded as if to execute in an address space starting at location zero in core memory. However, user programs are never started at location zero, but at the most convenient location in core. All references to locations made by the user program during its execution are relocated to actual physical core memory addresses. Relocation is accomplished in different ways, depending on the type of processor in the system.

2.2 MEMORY PROTECTION AND RELOCATION

When a user program is executing, the processor operates in user mode. In this mode, certain operations are illegal (e.g., I/O instructions), and all address references are relocated. The relocation hardware prevents a user from accessing any location not assigned to him by the monitor; conversely, the relocation hardware prevents access to his assigned area by another user.

The user specifies the size of his program; from that information the DECsystem-10 monitor determines the position in core memory where his program will reside.

There are three types of processors available with the DECSystem-10 – the KA10, the KI10, and the KL10. Monitors for the KI10 and KL10 processors are available with the virtual memory option. The methods for relocating user programs into core memory are:

- the KA10 method,
- the KI10 /KL10 method, or
- the KI10/KL10 with virtual memory method.

Each method determines the core resident size and position of each user's area in a different way. A program running on a KI10-based system (or KL10) with the virtual memory option is upwards compatible with those running on the KI10 without virtual memory and with those running on the KA10.

2.2.1 The KA10 Processor

With a KA10 processor, the monitor relocates each user program on a per segment basis. The segments of a user program (i.e., just a low segment, or both high and low segments) are relocated into core memory, occupying contiguous blocks of 1024 words each. The relocation is accomplished by protection and relocation registers. Besides relocation, segment protection is performed by these registers, which prevents one user job from accessing memory assigned to the monitor or to another user job.

Each segment of a user program has a relocation and a protection address. The relocation address is the absolute core address of the first location in the segment, as seen by the hardware. The protection address of each segment is the maximum relative address that the user may reference. The hardware defines both these addresses in units of 1024-word blocks. Relocation is accomplished dynamically by adding the contents of the appropriate relocation register to every user address reference.

All physical address locations are invisible to the user, as is the process of relocation. The relative user address and relocated address configuration on the KA10 are shown in Figure 2-1, where PL, RL, PH, and RH are the protection and relocation addresses of the low and high segments, respectively. If the low segment is more than half the maximum capacity (i.e., PL is greater than or equal to 400000), the high segment will start at the first location after the low segment ends (i.e., at PL+200000). The high segment is limited to 128K.

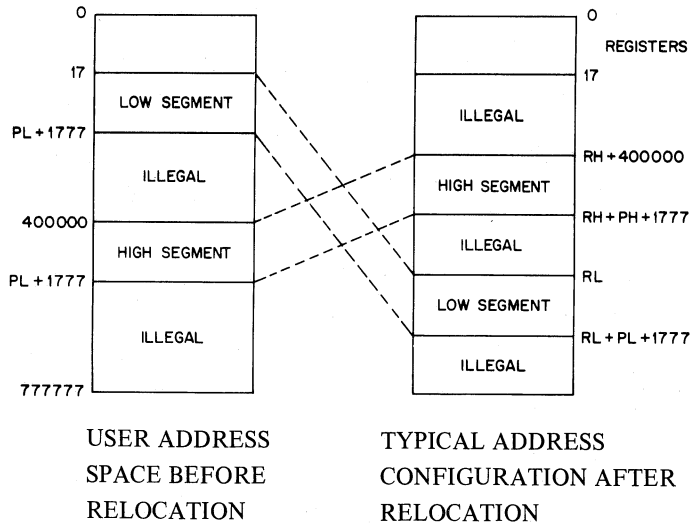


Figure 2-1 KA10 User Address Relocation

In summary, the KA10 relocates each segment of a user program into contiguous blocks of core memory. Relocation and protection are accomplished via the relocation and protection registers. At any one time during a program's execution, the entire program is core resident.

2.2.2 The KI10 and KL10 Processors (Without Virtual Memory)

KI10 and KL10 based programs are relocated and protected just as KA10 based programs are; relocation is accomplished by paging hardware. A KI10 or KL10 processor relocates user programs into core memory in the form of pages. A page consists of 512_{10} words, and the maximum possible user address space is 512_{10} pages (or 256K) of core. A user program of more than 512 words, when relocated, will consist of several pages.

The pages composing a user program are relocated into core individually. One page's physical placement has no connection with that of any other page. The monitor maintains a map for translating user addresses into their actual physical addresses. The map, which is invisible to the user, is kept in a page known as the user process table or the user map page. The paging hardware in the KI10 and the KL10 employs the user process table to relocate all user address references. Since all address references must be mapped through the process table, a user program can access only those physical pages contained in that program's process table. Therefore, the paging hardware provides protection and relocation capabilities compatible with the KA10's protection and relocation registers.

The important difference between the KA10 and the KI10/KL10 (without virtual memory) is that the pages of a segment do not have to be contiguous on the KI10/KL10 as they do on the KA10. All pages forming a program, however, must be in core whenever that program is executed.

Figure 2-2 illustrates the KI10/KL10 paging method (without virtual memory).

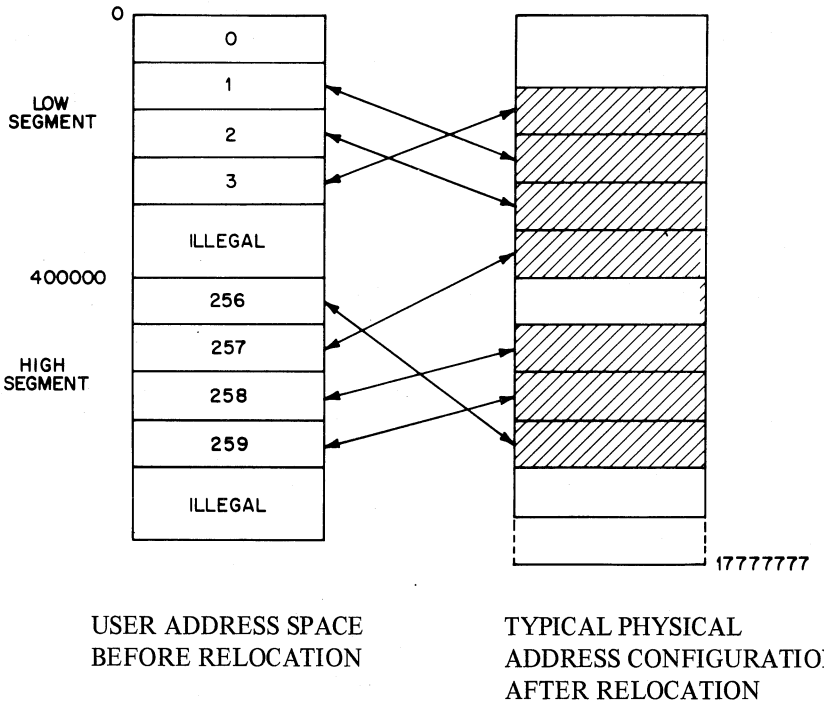


Figure 2-2 KI10/KL10 Paging Configuration (Without Virtual Memory)

2.2.3 KI10 and KL10 Processors With the Virtual Memory Option

The virtual memory option of the 6.01 and 6.02 monitors makes further use of the KI10/KL10's paging hardware. The pages of a user program are relocated individually, but not all pages need reside in core memory during execution. Some pages may be in core and the rest in secondary storage (i.e., disk or drum). Therefore, the virtual memory option makes it possible to run programs that are significantly larger than the physical core memory available for their execution.

Assume that user A has a 50-page program but core memory has only 20 pages blank. With the virtual memory option, the monitor can swap into core several of user A's pages, while keeping the rest on a secondary storage device. When one of the stored pages is referenced, it is brought into core while another page is swapped out to make room for it.

Deciding which pages are initially swapped to core and which pages are swapped into secondary storage is a function of the page fault handler. The page fault handler also decides which page will be swapped into secondary storage to make room in core for a new page. A user may create his own page fault handler; if he does not, the system default page fault handler will be used.

For more information concerning the page fault handler and virtual memory, refer to Chapter 4 and to *GUIDES TO VIRTUAL MEMORY.*

Using virtual memory is a privilege granted (or denied) by the system administrator at each installation. Therefore, not all users at an installation may utilize the virtual memory features, if the system administrator so chooses.

2.2.3.1 Virtual Memory Organization – Virtual memory permits a program to reference an address space that is larger than the actual physical core occupied during that program's execution. Programs running on a virtual memory system need no modifications from the same programs running on non-virtual memory systems. It is possible with the virtual memory option to execute very large programs (e.g., BLISS-10 programs) on small systems.

For efficiency and rapid response, the monitor is always core resident. High segments can be paged or shared, but not both. A sharable high segment must be completely core resident during its execution.

Not all programs utilize virtual memory during their execution. If a user is authorized to employ the virtual memory feature, his program will go virtual only when one of the following events occurs:

1. The program exceeds the user's physical core limit when he issues a GET or RUN command.
2. The program uses the CORE monitor call (or command) to expand the program's memory beyond the user's physical core limit; subsequently, the program references one of the newly created pages.
3. The program assumes direct control of its memory management with the PAGE. monitor call.

Figure 2-3 illustrates the limits imposed on a virtual memory system.

Figure 2-3 uses several abbreviations, which are described in Table 2-1.

At the moment a job's current physical page count becomes greater than its physical page limit, i.e.,

$$CPPC > CPPL$$

that job will go virtual. A user with virtual memory privileges can control how much of his job will be core resident at any time. If a user lowers a program's physical page limit to fewer than the number of pages within the program, he will force the program to use virtual memory. A user controls the current physical page limit via the SET PHYSICAL LIMIT command and the current virtual page limit via the SET VIRTUAL LIMIT command.

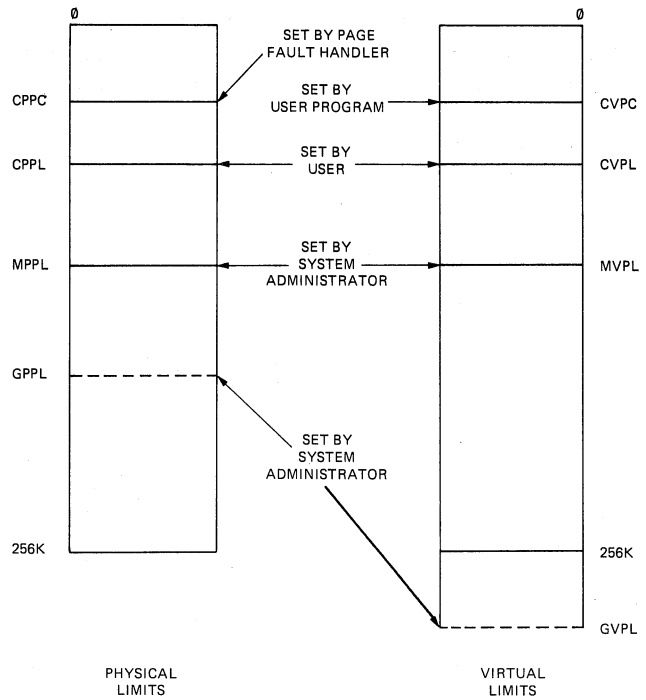


Figure 2-3 Physical and Virtual Page Limits

Table 2-1
VM Abbreviations

Abbreviation	Meaning
GPPL	The global physical page limit set by the system administrator using a privileged SETUUO.
GVPL	The global virtual page limit set by the system administrator using a privileged SETUUO.
MPPL	The maximum physical page limit set by the system administrator using a privileged SETUUO.
CPPL	The current physical page limit which can be set by each user via the SET PHYSICAL LIMIT command or an unprivileged SETUUO.
CPPC	The current physical page count established by the user program or by the page fault handler.
CVPL	The current virtual page limit set by the user via the SET VIRTUAL LIMIT command or an unprivileged SETUUO.
CVPC	The current virtual page count established by the user program.

The system administrator establishes a maximum virtual page limit (MVPL) for each user, a maximum physical limit for all users (GPPL), and a combined virtual limit that applies to the total amount of virtual memory (i.e., secondary storage) in use by all virtual memory users (GVPL). These limits are set by LOGIN using the privileged functions of the SETUUO.

CHAPTER 3
JOB DATA AREA

3.1 JOBDAT (JOB DATA AREA)

The first 140_8 locations in the user's core area are allocated to the Job Data Area. The locations within this area are given mnemonic assignments, where the first three characters are always .JB.

The function of the Job Data Area is to store information of interest to the monitor and to the user. Some locations within the Job Data Area, such as .JBPFH and .JBINT, are set by the user's program for use by the monitor. Other locations, such as .JBREL, are set by the monitor for use by the user's program.

The JOBDAT locations that are of significant importance to the user are listed in Table 3-1. The JOBDAT locations that are not listed within this table are those used by the monitor and those unused at the present time. User programs should not be written to reference any location in the Job Data Area that is not described in Table 3-1.

JOBDAT is loaded automatically (when needed) during LINK-10's library search for undefined global references; the values are assigned to the mnemonics at this time also. JOBDAT exists as a .REL file on device SYS: so that it may be loaded with the user programs that symbolically refer to locations within it. When a user program refers to a JOBDAT location, the location's assigned mnemonic should be used. The mnemonic must first be declared as an EXTERNal reference to the assembler (refer to the MACRO-10 Programmer's Reference Manual). All mnemonics beginning with the characters .JB within this manual refer to locations within the Job Data Area (JOBDAT).

Table 3-1
Job Data Area Locations
(for user program reference)

Mnemonic	$Location_8$	Description
.JBUUO	40	User's location 40 which is used by the hardware, when processing user monitor calls (001 through 037), for storing the op code and the effective address.
.JB41	41	This instruction is executed to start the user's programmed operator service routine (usually this is a JSR or a PUSHJ).
.JBERR	42	Left Half: Reserved. Right Half: The accumulated error count from one system program to the next. System programs should be written to look at the right half of this location only.
.JBREL	44	Left Half: Reserved. Right Half: The highest relative core location available to the user (i.e., the contents of the memory protection register when this user program is executing).

Table 3-1 (Cont.)
Job Data Area Locations
(for user program reference)

Mnemonic	$Location_8$	Description
.JBDDT	74	Left Half: The last address of DDT. Right Half: The starting address of DDT, which can be set with the SET DDT monitor call. If the contents of .JBDDT are zero, it indicates that DDT has not been loaded. If the monitor includes the virtual memory option and this location contains zero, the monitor will attempt to read into core SYS:DDT.VMX when the user executes a DDT command (refer to the DECsystem-10 Operating System Commands Manual). If successful, SYS:DDT.VMX is brought into the program's virtual address space, starting at the user virtual address 700000. The left and right halves of .JBDDT will then be set to the appropriate values.
.JBPFI	114	All user I/O references must be to locations greater than 114.
.JBHRL	115	Left Half: The first relative free location in the high segment (relative to the high segment origin, so it is the same as the high segment length). This location is set by LINK-10 and on subsequent GETs, even if there is no file to initialize the low segment. The left half is a relative quantity because the high segment can appear at different user origins at the same time. The SAVE command uses this value to determine how much to write from the high segment. Right Half: The highest legal address in the high segment. This value is set by the monitor each time a user program begins execution or executes a CORE or REMAP monitor call. .JBHGA allows this value to be any non-zero value. The proper way to test if a high segment exists is to test this half-word for a non-zero value.
.JBSYM	116	A pointer to the symbol table created by LINK-10. Left Half: The negative length of the symbol table. Right Half: The lowest address used by the symbol table.
.JBUSY	117	A pointer to the table of undefined symbols created by LINK-10 or defined by DDT. If .JBUSY contains a value greater than or equal to zero, there are no undefined symbols. Left Half: The negative length of the undefined symbol table.

Table 3-1 (Cont.)
Job Data Area Locations
(for user program reference)

Mnemonic	$Location_8$	Description	
		Right Half:	The lowest address used by the undefined symbol table.
.JBSA	120	Left Half:	The first free location in the low segment, which is set by LINK-10.
		Right Half:	The starting address of the user's program.
.JBFF	121	Left Half:	Zero.
		Right Half:	The address of the first free location following the low segment. This value is set to the contents of .JBSA (left half) each time a RESET monitor call is executed.
.JBPFH	123	Left Half:	The last address of the page fault handler.
		Right Half:	The starting address of the page fault handler. If the contents of .JBPFH are zero, the user program does not currently have its own page fault handler. When a page fault occurs, the monitor will read in SYS:PFH.VMX to the top of the user program's virtual address space; the left and right halves of .JBPFH will be set accordingly.
.JBREN	124	Left Half:	Unused.
		Right Half:	The REENTER start address. This value is set by the user program or by LINK-10, and it is used when the REENTER command is executed as an alternate entry point.
.JBAPR	125	Left Half:	Zero.
		Right Half:	This value is set by the user program as a trap address when the user is enabled to handle APR traps (such as illegal memory references, push-down list overflows). Refer to the APRENB monitor call description, Chapter 5.
.JBCNI	126	The state of the APR as stored by a CONI APR when a user-enabled APR trap occurs.	
.JBTPC	127	The PC of the next instruction to be executed after a user-enabled APR trap occurs. This value is set by the monitor.	

Table 3-1 (Cont.)
Job Data Area Locations
(for user program reference)

Mnemonic	$Location_8$	Description
.JBOPC	130	The previous contents of the job's last user mode program counter (PC). This value is set by the monitor each time a DDT, REENTER, START, or CSTART command is executed. Location .JBOPC contains the address of the HALT instruction, when the user program contains a HALT instruction followed by the execution of a START, DDT, CSTART, or REENTER command. In order to process at the address specified by the effective address, the user program must recompute the effective address of the HALT instruction and use that address to start. Similarly, after an error has occurred during execution of a monitor call followed by a START, DDT, CSTART, or REENTER command, .JBOPC will point to the address of the monitor call.
.JBOVL	131	Left Half: Zero. Right Half: A pointer to the header block for the root link.
.JBCOR	133	Left Half: The highest location in the low segment loaded with a non-zero. If .JBCOR is less than 140, a low segment file will not be written when a SAVE or SSAVE command is executed. This value is set by LINK-10. Right Half: The user specified argument on the last execution of a SAVE or GET command. This value is set by the monitor.
.JBINT	134	Left Half: Reserved for the future. Right Half: Zero or the address of the error-intercepting block.
.JBOPS	135	Reserved for all object-time systems.
.JBCST	136	Reserved for customers.
.JBVER	137	The program version number; its bit definitions are: Bits 0-2 =0 indicates that the Digital development group last modified the program. =1 indicates that other Digital employees last modified the program. =2, =3, =4 indicates that a customer last modified the program.

Table 3-1 (Cont.)
Job Data Area Locations
(for user program reference)

Mnemonic	Location[8]	Description
		=5, =6, =7 indicates that a customer's user last modified the program. Bits 3-11 Digital's latest major revision number, which is usually incremented by 1 after a release. Bits 12-17 Digital's minor version number, which is usually 0, but may be >0 if an update is needed to a program after work has started on the next major release. Bits 18-35 The edit number, which is increased by 1 after each edit. This value not reset. The VERSION and the SET WATCH VERSION commands output the version number in the standard format, refer to the DECsystem-10 Operating System Commands Manual.
.JBDA	140	The first location available to the user.

3.2 VESTIGIAL JOB DATA AREA

Some constants in the job data area may be loaded by a two-segment, one-file program without executing a GET command, and some locations are loaded by the monitor when a GET command is executed. The vestigial Job Data Area (the first ten (octal) locations of the high segment) is reserved for these low-segment constants. Therefore, a high segment program is loaded at the high segment origin + 10 (refer to .JBHGA in Table 3-2) instead of at the high segment origin (refer to Table 3-1). By placing the vestigial job data area within the high segment, LINK-10 automatically loads the constant data into the Job Data Area without requiring a low segment file when executing a GET or RUN command, or a RUN monitor call. The SAVE command writes a .LOW file for a two-segment program only if the left half of .JBCOR is 140(8) or greater (refer to Table 3-1).

Table 3-2
Vestigial Job Data Area Locations

Mnemonic	Location^8	Description
.JBHSA	0	A copy of .JBSA (location 120^8) in Table 3-1).
.JBH41	1	A copy of .JB41 (location 41^8) in Table 3-1).
.JBHCR	2	A copy of .JBCOR (location 133^8 in Table 3-1).
.JBHRN	3	Left Half: Restores the left half of .JBHRL (location 115^8 in Table 3-1). Right Half: Restores the right half of .JBREN (location 124^8 in Table 3-1).
.JBHVR	4	A copy of .JBVER (refer to Table 3-1, location 137 for the bit definitions).
.JBHNM	5	The high segment name which was set on the execution of a SAVE command.
.JBHSM	6	A pointer to the high segment symbols, if there are any.
.JBHGA	7	Bits 0-9 indicate the high segment origin. The monitor places the high segment at location 400000^8 or if the segment is larger than 128K, at the first available page boundary (1K on KA10 systems) above the low segment. This nine-bit byte should always be zero on KA10 systems. However, if the field is non-zero on KI10 or KL10 systems, it is taken as the page where the high segment is to start. This value is set by LINK-10. Bits 10-35 are unused fields that are reserved for future expansion and must contain zero.
.JBHDA	10	The first location not used by the vestigial Job Data Area.

[1] Relative to the origin of the high segment, usually .JBHGH = 400000^8.

CHAPTER 4
JOB CONTROL AND INFORMATION

4.1 JOB CONTROL

4.1.1 Start Program Execution

One user program may start the execution of another by either the RUN or the GETSEG monitor call (refer to sections 6.2.1 and 6.2.2. You may start or continue the execution of a user program with any of the following operating system commands.

1. R
2. RUN
3. START
4. CSTART
5. CONT
6. CCONT
7. REENTER

Refer to the *DECsystem-10 Operating System Commands Manual* for specific information concerning these commands. The user program's start address can be the argument of a command, or it can be stored in the program's Job Data Area.

4.1.2 Stop Program Execution

A running program may be stopped in several ways:

1. You can type one CTRL/C from your terminal if your program is in TTY input wait state (TI)[1]; otherwise, you need two CTRL/Cs to stop your program from the terminal.
2. The program contains a monitor detected error.
3. The program can execute one of the following monitor calls:

HALT	see paragraph 4.1.2.1
EXIT function,	see paragraph 4.1.2.2
LOGOUT ac,	see paragraph 4.1.2.3

4.1.2.1 The HALT Instruction – The HALT instruction is an exception to the illegal instructions rule (refer to section 1.1.1). HALT traps to the monitor, stops the current job, and prints the following message at your terminal.

?HALT AT USER PC *addr*

where: *addr* is the location of the HALT instruction.

If the HALT instruction is in location 41 and the program executes a user call (op-codes 001 through 037), the address in the error message will be that of the user call instead of address 41.

To continue the program at the effective address of the HALT instruction (*provided that the effective address does not equal zero or the address of the HALT*), you can issue either the CONT or CCONT command from your terminal. After a user program executes a HALT instruction followed by your typing either a START, DDT, CSTART, or

[1] The state of a program can be determined by using the USESTAT command or CTRL/T (refer to the Operating System Commands Manual).

REENTER command, location .JBOPC in the Job Data Area will contain the address of the HALT instruction. To continue from the effective address specified, you or your program must recompute the effective address of the HALT instruction. HALT should not be used to terminate the execution of a program; its main use is to indicate possible error conditions.

4.1.2.2 The EXIT Monitor Call (CALLI 12) – The EXIT monitor call will stop a job (with optional RESET). Its calling sequence is

```
EXIT function,
return on a continue     ;if function =1
```

where: *function* is either 1 or 0. A 1 stops the job; a 0 performs a RESET and stops the job.

The EXIT monitor call performs the following functions:

1. All I/O devices (including real-time devices) are RELEASED.
2. The job is unlocked from core.
3. The user mode write-protect bit for the high segment is set.
4. The APR traps are reset to zero.
5. The PC flags are cleared.
6. The job is stopped.

(1–5: function 0; 1–6: functions 0 and 1)

If timesharing was stopped by a TRPSET monitor call, it is resumed by the monitor. After releasing all devices that close files, a RESET is performed. A carriage return/line feed sequence is performed, and the word EXIT is printed at the user's terminal, which is left in monitor mode. You cannot type the CONT and CCONT commands to continue the recently stopped program.

When the function specified is 1, the job is stopped, and a return is made to monitor mode (EXIT is not printed on the user terminal). You can type the CONT or CCONT commands to continue the program's execution. In other words, this form of the EXIT monitor call does not affect the state of your job except to stop it and return your terminal to monitor mode. Note that programs using EXIT with a function code of 1 (MONRT.) should RELEASE all devices first.

4.1.2.3 The LOGOUT Monitor Call (CALLI 17) – The LOGOUT monitor call is used by the LOGOUT program to release all I/O devices associated with the job and to return them to the monitor pool along with the job's allocated core and job number. If a program other than LOGOUT uses this monitor call, the LOGOUT call functions identically to EXIT (function 0).

4.1.3 Suspend the Execution Of A Job

There are two monitor calls that can suspend the execution of a job. The SLEEP monitor call suspends execution for a specified amount of time; the HIBERnate monitor call suspends execution until a specified event occurs.

4.1.3.1 The SLEEP Monitor Call (CALLI 31) – The SLEEP monitor call allows a user program to stop or "sleep" for a specified number of seconds. Its calling sequence is

```
MOVEI  ac, seconds
SLEEP  ac,
only return
```

where: *seconds* is the number of seconds the job is to sleep.

SLEEP will temporarily stop a job and continue it automatically after the specified number of real-time seconds has elapsed. If the SLEEP monitor call is issued with a zero argument, the job will sleep for one clock tick. The explicit maximum is approximately 68 seconds (82 seconds in 50-Hz. countries and on the KL10 using 50-Hz.). A program

that requires a SLEEP time longer than 68 seconds should call DAEMON (via the .CLOCK function) to put it to sleep (refer to DAEMON in section 4.4.3). After invoking DAEMON, the program can use the HIBERnate monitor call with no clock request and DAEMON will awaken the job.

4.1.3.2 The HIBERnate Monitor Call (CALLI 72) – The HIBERnate monitor call stops the current job until a specified event occurs. Its calling sequence is

```
MOVSI  ac, bits
HRRI   ac, sleep time     ;or  IORI ac, sleep time
HIBER  ac,
   error return
normal return
```

where: *bits* specifies one or more HIBERnate conditions, as listed in Table 4-1.

sleep time is the number of milliseconds for which the current job is to sleep. This value is rounded up to an even multiple of jiffies (with a maximum value of 68 seconds). A 0 specifies an infinite sleep time (i.e., no clock request).

If the HIBERnate monitor call has not been implemented in the current system, the error return is taken; under this circumstance, the SLEEP monitor call can be used.

The normal return is taken after an enabled HIBERnate condition occurs.

To prevent a job from oversleeping and missing an event, the monitor sets the wakeup bit (even if the event occurs while the job is awake). Another HIBERnate monitor call will clear the wakeup bit. A job issuing a monitor call should test all events that may have caused it to wakeup; however, the job cannot assume that any one of the events actually happened.

Privileged jobs (those running with the JACCT bit set or those logged in under [1,2]) can be written to wake any HIBERnating job.

A RESET monitor call always clears the job's protection code and wake-enable bit. Until the first HIBERnate monitor call, therefore, there is no protection against wakeup commands from other jobs. To guarantee such protection, the calling job should execute a WAKE monitor call on itself followed by a HIBERnate monitor call with the desired protection codes. The WAKE call ensures that the first HIBERnate call takes the normal return immediately, which leaves the job with the correct protection code. Note that a correctly written program will not fail if it was aWAKEned for no apparent reason.

4.1.3.3 The WAKE Monitor Call (CALLI 73) – The WAKE monitor call allows one job to activate a dormant job when some event occurs. Its calling sequence is

```
MOVEI  ac, job number
WAKE   ac,
   error return
normal return
```

where: *job number* is the number of the job to be awakened. (The number -1 indicates the calling job.)

Real-time process control jobs can cause other process control jobs to run in response to a specific alarm condition. WAKE can be called from an RTTRP job running at interrupt level, thereby allowing a real-time job to wake its background portion quickly to respond to some real-time condition. (Refer to the RTTRP monitor call for restrictions on accumulators when using the call at interrupt level.)

Table 4-1
HIBERnate Conditions

Bit	Mnemonic	Meaning
0	HB.SWP	Job will be swapped out immediately.
10	HB.IPC	The job will be awakened on the delivery of an IPCF packet to an input queue.
11	HB.RIO	The job will be awakened when asynchronous I/O is finished.
12	HB.RPT	The job will be awakened when PTY activity occurs since the last HIBERnate.
13	HB.RTL	The job will be awakened when a line of input is ready.
14	HB.RTC	The job will be awakened when a character is ready.
15	HB.RWJ	The HIBERnating job can be awakened only by itself.
16	HB.RWP	The calling job and the HIBERnating job must have the same programmer number (a WAKE protection code).
17	HB.RWT	The calling job and the HIBERnating job must have the same project number (a WAKE protection code).

If the calling job does not have the appropriate privileges, the error return will be taken. If any of the enabled conditions specified in the last HIBERnate monitor call occur, the wake bit is set for that job. A wake bit is associated with each job. At the next HIBERnate call, the wake bit is cleared and the call returns immediately. The wake bit eliminates the problem of a job's oversleeping a WAKE condition.

On a normal return, the referenced job will be awake and start at the return location of the HIBERnate that caused the job to become dormant.

4.2 SET OR OBTAIN JOB INFORMATION

There are several monitor calls that can obtain and set system/job information. The user can set the current program's name and set various system/job parameters (like the time between virtual time traps). He can change a job's current logical node/station number, or obtain its accumulated run time, job number, or project-programmer number.

4.2.1 Set The Program Name

The SETNAM monitor call (CALLI 43) allows the user program to set the program name. The calling sequence for SETNAM is

```
MOVE    ac, [SIXBIT/name/]
SETNAM  ac,
only return
```

where: *name* is a left-justified SIXBIT program name.

SETNAM stores the specified program name in the monitor job table, which is used by the SYSTAT system program. SETNAM performs several addition functions: it clears the SYS: program bit (JB.LSY, which is used by Batch); it clears the execute-only and JACCT bits; and it causes a version typeout to occur on the user's terminal if version watching has been enabled by the SET WATCH VERSION command (refer to the *DECsystem-10 Operating System Commands Manual*).

4.2.2 Set System/Job Parameters

The SETUUO monitor call (CALLI 75) allows the user program to set various system and/or job parameters. To set system parameters, the calling program must be running with the JACCT bit set in the privilege word, or the job must be running under [1,2]. The calling sequence for SETUUO is

```
MOVE    ac, [XWD function, arg]
SETUUO  ac,
   error return
normal return
```

where: *function* is one of the SETUUO function codes listed in Table 4-2.

arg is the argument needed for the specified function. The arguments are also listed in Table 4-2.

The error return is taken if one of the following conditions occurs:

1. The monitor call has not been implemented.
2. The user is not suitably privileged for the function specified.
3. The argument is invalid.

Table 4-2
SETUUO Functions

Code	Mnemonic	Argument
0	.STCMX[1]	The largest size that any job may assume (i.e., the sum of the high segment and low segment). The minimum value is 10K unless changed via MONGEN at monitor generation time. The maximum value is the size of user (non-monitor) core. (This value is known as *CORMAX*.)
1	.STCMN[1]	The guaranteed amount of contiguous core that a single unlocked job may assume. The minimum value is 0; the maximum value is the value of CORMAX. (This value is known as *CORMIN*.)
2	.STDAY[1]	A decimal number (in the range 0-2359) representing the DAYTIME (i.e., time is computed using a 24-hour clock, *hours* * 100 + *minutes*).
3	.STSCH[1]	An octal argument stored in the right half of the *STATES word* in COMMON. 0 regular timesharing 1 no further LOGINS are allowed (except from CTY) 2 no further LOGINs from remote terminals, data sets are not answered 4 no further LOGINs are allowed (except for Batch jobs) 10 the system is running stand-alone 100 device mounts without operator intervention are allowed 200 unspooling is allowed 400 no operator coverage
4	.STCDR	The input name for this job. This value (3-SIXBIT characters) will be stored in the left half of .STSPL.

[1]This is a privileged function.

Table 4-2 (Cont.)
SETUUO Functions

Code	Mnemonic	Argument
5	.STSPL[2]	The bits are 31-35 of .STSPL: Bit — Mnemonic — Meaning 35 — JS.PLP — LPT spooling 34 — JS.PPL — PLT spooling 33 — JS.PPT — PTP spooling 32 — JS.PCP — CDP spooling 31 — JS.PCR — CDR spooling
6	.STWTC	Bits are 18-23 of .STWCH: Bit — Mnemonic — Meaning 19 — ST.WDY — Watch daytime at start. 20 — ST.WRN — Watch run time. 21 — ST.WWT — Watch wait time. 22 — ST.WDR — Watch disk reads. 23 — ST.WDW — Watch disk writes 24 — ST.WVR — Watch versions. 25 — ST.WMT — Watch MTA Statistics
7	.STDAT[1]	The decimal number of days since January 1, 1964. (DATE = ((*year* – 1964) * 12 + (*month* – 1) * 31 + *day* = 1))
10	.STOPR[1]	The address of the word which contains the SIXBIT physical name of the TTY to be used as the Operator terminal.
11	.STKSY[1]	The decimal number of minutes until timesharing ceases. If 0, time-sharing is not to be stopped (This value is known as *KSYS*).
12	.STCLM[1]	The maximum amount of core (in words) for the job. This value is stored in bits 1-9 (JB.LCR) of .GTLIM (units of 512-word pages).
13	.STTLM[1]	The job's time limit in seconds.
14	.STCPU	The CPU specification for this job. Bit — Mnemonic — CPU 35 — SP.CR0 — CPU0 34 — SP.CR1 — CPU1 33 — SP.CR2 — CPU2 32 — SP.CR3 — CPU3 31 — SP.CR4 — CPU4 30 — SP.CR5 — CPU5
15	.STCRN[1]	Bits which indicate a CPU's runnability. If bit 35 = 1, CPU0 is runnable. If bit 34 = 1, CPU1 is runnable, and so on.

[1]This is a privileged function.
[2]This is not a privileged function unless the user is unspooling devices.

Table 4-2 (Cont.)
SETUUO Functions

Code	Mnemonic	Argument
16	.STLMX[1]	The maximum number of jobs allowed to be logged in at any one time. The maximum value for this symbol is JOBN, which is the system limit specified at monitor generation time via MONGEN. The minimum value is 1. If LOGMAX is a value smaller than the number of jobs currently logged in, no new jobs are allowed to LOGIN until enough current jobs LOGOUT to get the system below LOGMAX. The number of jobs currently logged in (referred to as LOGNUM) can be obtained from the GETTAB table .GTCNF, item number 54 (%CNLOG). (This value is referred to as *LOGMAX.*)
17	.STBMX[1]	The maximum number of Batch jobs that are allowed to be logged-in concurrently (BATMAX). The maximum value is the smaller of either the BATCON limit of 14 or the system limit of JOBN. The initial value is 14. The number of jobs currently logged-in (BATNUM) can be obtained from the GETTAB table .GTCNF, Item Number 55 (%CNBAT).
20	.STBMN[1]	The number of jobs guaranteed for batch jobs (BATMIN). The maximum value is either the BATCON limit of 14 or the value of JOBN-1 (one job must be reserved for BATCON), whichever is smaller. The initial value is 0. Therefore, the maximum number of non-Batch jobs is JOBN minus BATMIN.
21	.STDFL[2]	DSKFUL for this job. An argument of 0 (.DFPSE) causes a pause and an argument of 1 (.DFERR) causes an error when the disk is full or the user's quota is exceeded. The current setting can be obtained by using an argument other than 0 or 1. The value returned in the AC is either 0 or 1 depending on whether PAUSE or ERROR is set. The initial setting is ERROR.
22	.STMVM[1]	The system-wide virtual memory limit. The value returned on a normal return is dependent on the value of the call's argument. The returned value will be either: 1. The total amount of virtual memory in use by VM users, if the current argument is less than the current virtual memory page count. 2. The total amount of available swapping space, if the argument is greater than the current available swapping space. 3. The call's argument, if the argument is greater than the total amount of virtual memory currently in use.
23	.STMVR[1]	The address of the word that contains either the SYSTEM or the JOB designation and the virtual memory page fault rate. Left half = 0: Set the system-wide page fault rate to RATE. Left Half = 1: Set the per-job page fault rate for this job to RATE. Right Half = RATE: The number of page faults per CPU second.

[1]This is a privileged function.
[2]This is not a privileged function unless the user is unspooling devices.

Table 4-2 (Cont.)
SETUUO Functions

Code	Mnemonic	Argument
24	.STUVM[1]	The address of the word that contains the maximum virtual memory page limit and the maximum physical page limit. Left half = maximum virtual page limit. Right half = maximum physical page limit. If the left half of the word contains zero, the user cannot utilize the virtual memory option. If the right half of the word contains zero, all of core is indicated.
25	.STCVM[2]	The user's current virtual memory maximum, which is stored in a word with the format: Right half = *current virtual page guideline* Left half = *current virtual page limit* If either the left half or the right half is zero, the current value is unchanged. If bit 18 = 1, bits 19-25 contain the guideline.
26	.STTVM	The time interval between virtual time traps in milliseconds. This type of trap causes a code 5 page fault to the page fault handler each time the time interval has elapsed.
27	.STABK	The address break condition. On a normal return, the new address break condition and the break address will have been set. Address conditions are: 0 Break on EXECUTE 1 Break on READ 2 Break on WRITE 3 Break on the execution of a monitor call Bits 9-17 contain a quantity that specifies the number of times the break address is to be referenced before the interrupt occurs. Bits 18-35 contain the break address. Note that if bits 0, 1, 2, and 3 are equal to 0, the address break is cleared. If the user is enabled for address break interrupts, the software interrupt system will interrupt when an address break occurs.
30	.STPGM	Set the program to run.
31	.STDER	Set deferred spooling.

[1]This is a privileged function.
[2]This is not a privileged function unless the user is unspooling devices.

4.2.3 Set The Logical Node

A user program (via LOCATE, CALLI 62) can change a current job's logical node number. The calling sequence for the LOCATE monitor call is

```
MOVE    ac, [SIXBIT/node-name/]
MOVEI   ac, station-number
LOCATE  ac,
   error return
normal return
```

where: *node-number* can be one of the following:

-1 Changes the job's location to the physical node of the job's terminal.

0 changes the job's location to that of the central station.

n changes the job's location to node number n.

The normal return is taken when the node number or name has been defined and the node is in contact with the central site. Subsequent generic device specifications are associated with the new node number/name.

The error return is taken if

1. the monitor call has not been implemented,
2. an illegal node number was specified,
3. or, the node specified was not in contact.

The following is an example to logically locate a job at node number 3.

```
MOVEI    AC1, 3
LOCATE   AC1,
   JRST  ERROR
JRST     WINNER
```

For more information about remote operation, refer to future editions of the *REMOTE STATION USER'S GUIDE.*

4.2.4 Obtain Run Time

The RUNTIM monitor call (CALLI 27) can be used to obtain a specified job's accumulated run time. The calling sequence for RUNTIM is

```
MOVEI    ac, job-number
RUNTIM   ac,
only return
```

where: *job-number* is the number of the job whose accumulated run time is requested. A job-number of 0 returns the current job's run time.

On a return, the accumulated run time (in milliseconds) is returned as a right-justified quantity in the AC. If the specified job number does not exist, 0 is returned in the AC.

4.2.5 Obtain The Job Number Of The Calling Job

The PJOB monitor call (CALLI 30) obtains the number of the calling job. PJOB's calling sequence is

```
PJOB   ac,
only return
```

On a return, the job number of the calling job is returned right-justified in the AC.

4.2.6 Obtain The Project-programmer Number Of The Calling Job

The GETPPN monitor call (CALLI 24) obtains the project-programmer number associated with the calling job. Its calling sequence is

```
GETPPN   ac,
normal return
   alternate return
```

On both a normal return and an alternate return, the project number is returned in the left half of the AC; the programmer number is returned in the right half.

The alternate (skip) return is taken if the calling program is running with the JACCT bit set, and the project-programmer number is associated with another logged-in job.

4.2.7 The OTHUSR Monitor Call (CALLI 77)

The OTHUSR monitor call helps the user program determine if its project-programmer number is logged-in for one or more other jobs, OTHUSR's calling sequence is

OTHUSR *ac,*
normal return
alternate return

On a return, the specified AC will contain the project-programmer number of the job executing the OTHUSR monitor call. The call has two return points: normal return (non-skip) and alternate (skip) return. The normal return is taken if one of the following occurs.

1. The monitor call has not been implemented, in which case the AC remains unchanged.
2. The monitor call is implemented and no other job is logged-in with the same project-programmer number, in which case the AC contains the calling job's project-programmer number.

The skip return is taken only if the monitor call is implemented, and one or more other jobs are logged-in with the same project-programmer number as that of the calling job. OTHUSR is utilized by KJOB.

4.3 TIMING INFORMATION

There are three methods of generating timing information: the *APR clock* (KA10/KI10), the *DK10* (internal accounting clocks) (KA10/KI10), and *internal clocks* (EBOX, MBOX on the KL10).

The *APR clock,* driven by the power source frequency (60 Hz. in North America, 50 Hz. in most other countries), is accurate over long periods of time. For this reason, it is used to keep the time of day (e.g., for the TIMER call). This clock can also account for the processor time used by each job. In this application, however, the APR clock loses some accuracy, because its tick is often longer than a job's run time period.

The *DK10 clock* (100K Hz.) is accurate over short periods of time. If it is present in the system, it can be used to account for processor run time instead of the APR clock. The DK10 clock, however, is more accurate than the APR clock.

KL10 processor systems have two types of system runtime, called *high precision runtime* and *EBOX/MBOX runtime.* All runtimes that are reported by the system (e.g., RUNTIME monitor call, TIME command, USESTAT command, CTRL/T, and GETTAB table .GTTIM) can be selected to be of either type. The type of runtime reported is determined by the value of bit ST%EMR (bit 20) in item %CNST2 of GETTAB table .GTCNF. ST%EMR equal to 1 selects EBOX/MBOX runtime.

EBOX/MBOX runtime is computed from the KL10 accounting meters, and is designed to be a highly reproducible measure of the amount of work a user job has accomplished. High precision runtime, which is the same as that used on KI10 and KA10 systems with DK10s, uses 10 microseconds resolution elapsed time on the CPU.

Note that EBOX/MBOX runtime cannot be compared with any other type of runtime. The relationship between EBOX/MBOX runtime and elapsed time on the CPU is not direct; it depends on the cache hit rate, memory speed, and other factors.

With all of these clocks, monitor overhead can optionally be included in the run time statistics for the current job. This parameter is set during Monitor Generation Time via MONGEN. Monitor overhead is the CPU time spent in tasks like clock queue processing, command decoding, core shuffling, swapping, and scheduling.

The traditional DECsystem-10 date (which can be obtained via the DATE monitor call) is a 15-bit integer. This integer is incremented by 1 each day, by 31 each month (regardless of the actual number of days in the month), and by 12 * 31 each year (regardless of the actual number of days in the year). The date format is easily resolved into the year-month-day format; however, the difference between two dates in this format is not necessarily the actual number of days between the two.

For convenience, the local (host computer) time can be converted to a *universal date-time standard,* where the word's left half is the day, and its right half is the time. This day is uniformly incremented at midnight, with 1 equal to November 18, 1858. The November date is used for consistency with the Smithsonian Astronomical Date Standard and other computer installations and systems. (Refer to GETTAB Table Number 11, Item Number 64.)

The time is specified as a fraction of the day, allowing the 36-bit value to be in units of days, with a binary point between the right and left halves of the word. The resolution is approximately one-third of a second (i.e., the least significant bit (bit 35) represents approximately one-third of a second).

The monitor maintains a set of GETTAB values which give the local date and time in terms of year, month, day, hours, minutes, and seconds. (Refer to GETTAB Table Number 11, Items 56 through 63.)

4.3.1 The DATE Monitor Call (CALLI 14)

The DATE monitor call returns the current date. Its calling sequence is

DATE *ac,*
only return

The current date will be returned in the AC as a right-justified 15-bit integer. This date is computed using the formula

$$((((year - 1964) * 12) + (month - 1) * 31 + (day - 1))) = date$$

where: January 1, 1964 is the base date.

4.3.2 The MSTIME Monitor Call (CALLI 23)

The MSTIME monitor call returns the current time of day. Its calling sequence is

MSTIME *ac,*
only return

The time is returned in the AC (in milliseconds) as a right-justified binary integer.

4.3.3 The TIMER Monitor Call (CALLI 22)

The TIMER monitor call returns the day time in jiffies, right-justified in the AC. Its calling sequence is

TIMER *ac,*
only return

A jiffy is 1/60 of a second (16.2 milliseconds) for 60-cycle power and 1/50 of a second (20 milliseconds) for 50-cycle power. MSTIME should normally be used so that the time is not a function of the power.

4.4 CONFIGURATION INFORMATION

4.4.1 The SWITCH Monitor Call (CALLI 20)

The SWITCH monitor call returns the contents of the central processor data switches in the specified AC; its calling sequence is

SWITCH *ac,*
only return

Users should issue this call with caution because the data switches are an allocated resource and are always available to all users. This call is meaningful only for KA10 and KI10 systems, since the KL10 has no data switches.

4.4.2 The LIGHTS Monitor Call (CALLI -1)

The LIGHTS monitor call displays the contents of the AC in the console lights. The calling sequence is

LIGHTS *ac,*
only return

This monitor call is meaningful only for KA10 and KI10 systems, because the KL10 has no console lights.

4.4.3 The DAEMON Monitor Call (CALLI 102)

The DAEMON monitor call invokes the DAEMON system program to perform a specified function. The calling sequence for DAEMON is

```
        MOVE    ac, [XWD length,addr]
        DAEMON  ac,
          error return
        normal return

addr:   function
        argument list
```

where: *length* is the number of arguments in the list plus one. If this number is fixed, length is zero.

addr points to the first word of the argument list.

function is the code of the DAEMON function desired. All DAEMON functions are described in Table 4-3. Some function codes are privileged; to use these, the user program must be running with the JACCT bit set or the user must be logged-in under [1,2].

argument list varies with the function code specified; the possible arguments are described along with the DAEMON functions codes in Table 4-3.

Table 4-3
DAEMON Functions

Code	Mnemonic	Arg Block	Meaning
1	.DCORE	1 SIXBIT/*dev*/ SIXBIT/*file*/ SIXBIT/*ext*/ 0,,0	Writes a dump file of the current job's core area. If argument list is omitted, the default is the same as default for DCORE. Refer to the *Operating System Commands Manual.*
2	.CLOCK	2 *seconds*	Enters a request in the clock queue awaken the current job after a specified number of seconds have passed. As soon as the request has been entered into the queue, the HIBER call should be used with no clock request. An argument of 0 clears the job's entry in the clock queue and wakes the job.
3[1]	.FACT	3 arg1 . . . arg208	Makes entry in the system accounting file (FACT). The FACT file entry must be less than 20 octal words in length. The entry will not be instantly written in FACT.SYS, but may be stored in core for as long as ten minutes.
5[1]	.DMERR	5 code word1 . . . word2	Makes an entry in the error file; the *block-2* is put into the error file.

[1]This is a privileged function.

CHAPTER 5
TRAPPING, INTERCEPTION, AND INTERRUPTION

The execution of a program is normally performed in a sequential manner, whereby one instruction is executed followed immediately by the next and so on. By using skip and branch instructions, it is possible for a program to deviate from the normal sequential method of execution. Deviation from normal program flow may also be accomplished by trapping to user trap-servicing routines (APRENB monitor call), enabling for error interception (utilizing .JBINT), or using the software interrupt system.

Using trap-servicing routines and error interception are simple methods for controlling error conditions; the software interrupt system is much more general and complex.

APR trapping allows a user program to handle traps that occur while the job is running, including illegal memory references and push-down list overflows. Error interception may be used when certain conditions occur in a user program. The monitor will intercept, when the condition occurs, and will examine location .JBINT in the Job Data Area. The monitor does this to find out whether or not an error interception routine has been provided. In addition, the monitor provides a generalized software interrupt mechanism for interrupting the sequential flow of operation under a wide variety of special conditions.

Two important reasons for wishing a program to deviate from simple sequential operation are as follows:

1. The program may wish to respond to special conditions without testing for them wherever they may occur. For example, it is possible for a program to test for an arithmetic overflow condition after every instruction that might cause the condition to occur. However, it is easier to have normal sequential operation interrupted each time an overflow occurs and then have control transferred to an error routine.
2. A program could respond to asynchronous events without testing for those events repeatedly. For example, some programs react to CTRL/C by taking a special action rather than permitting control to be returned immediately to monitor level.

   If a program were to test regularly and frequently for the possibility of a user typing a CTRL/C, an unreasonable constraint would be placed on the program's design. The program instead could be interrupted each time a CTRL/C was typed; control would be transferred to a special processing routine before control would be returned to the monitor. The interrupt would eliminate the need for repeated testing.

5.1 USER TRAP SERVICING

The APRENB monitor call (CALLI 16) enables a user program for trap servicing. The calling sequence for APRENB is

```
MOVEI    ac, flags
APRENB   ac,
only return
```

where: *flags* specifies the central processor flags to be tested on a trap condition. The sequence for APRENB flags that may be set are listed in Table 5-1.

When one of the specified conditions occurs while the central processor is in user mode, the state of the central processor is conditioned (CONI) into the Job Data Area location .JBCNI, and the PC is stored in location .JBTPC

Table 5-1
APRENB Flags

Mnemonic	Value	Bit	Trap Condition
AP.REN	400000	18	Repetitive enable.
AP.POV	200000	19	Pushdown list overflow.
AP.ILM	20000	22	Memory protection violation.
AP.NXM	10000	23	Non-existent memory flag.
AP.PAR	4000	24	Parity error.
AP.CLK	1000	26	Clock tick flag.
AP.FOV	100	29	Floating point overflow.
AP.AOV	10	32	Arithmetic overflow.

in the Job Data Area. After the arithmetic and floating point flags have been cleared, control is transferred to the user trap-servicing routine specified by the contents of the right half of .JBAPR in the Job Data Area. The job is stopped, however, if the PC is equal to the first or second instruction in the user's trap-servicing routine.

The user program must set up location .JBAPR before executing the APRENB monitor call. To return control to the interrupted program, the user's trap-servicing routine must execute a JRSTF .JBTPC, which clears the bits that have been processed and restores the state of the processor.

The APRENB monitor call normally clears traps for only one occurrence of any selected condition, and it must be reissued after each trap. To disable this feature, the user program can set bit 18 to a 1 when executing the monitor call. However, even when bit 18 equals 1, clock interrupts must be reenabled after each trap.

If the user program does not enable for traps, the monitor sets the processor to ignore arithmetic and floating-point overflows, but the monitor enables interrupts for other error conditions in the list above. If the user program produces such an error condition, the monitor will stop the user job and print one of the following messages:

?ILL MEM REF AT USER PC *addr*
?MEM PAR ERROR AT USER PC *addr*
?NON-EX MEM AT USER PC *addr*
?PC OUT OF BOUNDS AT USER PC *addr*
?PDL OV AT USER PC *addr*

After one of the above messages, the CONT and CCONT commands will not succeed.

5.2 ERROR INTERCEPTING

When certain conditions occur in a user program, the monitor intercepts them and examines location .JBINT in the Job Data Area. Depending on the contents of this location, control is either retained by the user program or it is transferred to the monitor. If this location contains zero, the job will be stopped, and the user (and possibly the operator) will be notified with an appropriate message, if any. If location .JBINT contains a non-zero value, the contents of the location will be interpreted as the address of a block that has the following format:

```
loc:     XWD   n, intloc
loc+1:   XWD   bits, class
loc+2:   0
loc+3:   0
```

where: *n* is the number of words in the block which must be at least 3.
intloc is the location at which the program is to be restarted.
bits may be 0 or 1 as described below:

bit 0 = 1 causes an error message, if any, to be printed on the user's terminal and/or the operator's terminal.

class contains bits which may be set to determine the action taken on a given error condition; the class bits are listed in Table 5-2.

For each type of error condition, there is an associated class bit. The job is interrupted for an error only if the appropriate bit in the class word is 1 and loc+2 contains zero. The job will be stopped if the appropriate bit is 0 or if the contents of loc+2 is not zero. The requirement that loc+2 contains zero precludes a loop.

Therefore, the monitor examines the class bits and loc+2 to determine whether to stop or interrupt the job after an error occurs. If the monitor interrupts, it stores the following information in loc+2 and loc+3:

loc+2: the last user PC word

loc+3: Right Half: the channel number (if applicable)
Left Half: the error bit as defined in class

The job is then restarted from the location specified in intloc.

Table 5-2
Error Intercepting Class Bits

Bit	Mnemonic	Error Condition
35	ER.IDV	A device error occurred that can be corrected by human intervention. The following message is printed DEVICE *xxx* OPR *zz* ACTION REQUESTED where *xxx* is the device name, and *zz* is the operator's station number. The following message is printed on the operator's terminal: %PROBLEM ON DEVICE *xxx* FOR JOB *n* where *xxx* is the device name, and *n* is the job number of the job stopped. When the operator has corrected the error and continued the job with the JCONT command, the message CONT BY OPR appears on the user's terminal to indicate that the error has been corrected.
34	ER.ICC	A user has typed a CTRL/C, interrupting the program. This interrupt allows the user program to process a CTRL/C instead of returning the job automatically to monitor level when a CTRL/C is typed, but it will instead trap to the user's interrupt routine. No messages will be printed on either the user's terminal or the operator's terminal. When this bit is set, the job should normally exit immediately by releasing any special resources and issuing an EXIT monitor call. The CONT command can be issued by the user to continue the job.
33	ER.OFL	A disk unit has dropped off-line. The following message is printed on the user's terminal: DSK IS OFF-LINE. WAITING FOR OPERATOR ACTION. TYPE ¯C TO GET A HUNG MESSAGE (IN 15 SECONDS). DON'T TYPE ANYTHING WAIT FOR THE OPERATOR TO FIX THE DEVICE. If the user has a system resource, this additional message will be printed on the user's terminal: THE SYSTEM WILL DO NO USEFUL WORK UNTIL THE DRIVE IS FIXED OR YOU TYPE A ¯C

Table 5-2 (Cont.)
Error Intercepting Class Bits

Bit	Mnemonic	Error Condition
		The following message is printed on the operator's terminal: UNIT *xxx* WENT OFF-LINE (FILE UNSAFE)[1] PLEASE POWER DOWN AND THEN TURN IT ON AGAIN
32	ER.FUL	A file structure has been filled with data (i.e., there are no free blocks). There will be no messages printed.
31	ER.QEX	The user's disk quota has been exhausted. The following message is printed on his terminal: [EXCEEDING QUOTA *file-structure name*]
30	ER.TLX	The user's run time limit, as set by the SET TIME command, has been exhausted. This error condition applies only to non-BATCH jobs. The following message is printed on the user's terminal: ?TIME LIMIT EXCEEDED
29	ER.EIJ	The job has a fatal error. The following are a few of the possible error messages which may be printed on the user's terminal: ?ILLEGAL UUO AT USER PC *addr* ?ADDRESS CHECK AT USER PC *addr* ?PC OUT OF BOUNDS AT USER PC *addr*

[1] only if the disk went unsafe

The following example shows how to enable and handle a CTRL/C intercept. Note that the user is returned to monitor level as quickly as possible.

```
        LOC     134             ;SET POINTER IN .JBINT
        EXP     INTBLK          ;TO THE INTERRUPT BLOCK
        RELOC

INTBLK: XWD     4,INTLOC        ;4 WORDS LONG,,PLACE TO START
        XWD     0,2             ;NO MESSAGE CONTROL,,TYPE 2 (^C)
        Z                       ;GETS LAST USER PC
        Z                       ;LH GETS INTERRUPT TYPE

;THE INTERRUPT ROUTINE STARTS HERE

INTLOC: MOVEM   1,TEMP1         ;SAVE AC 1
        HLRZ    1,INTBLK+3      ;GET REASON FOR INTERRUPT
        CAIE    1,2             ;SEE IF CONTROL-C
        HALT    .               ;ERROR IF NOT
                                ;RELEASE ANY SPECIAL RESOURCES HERE
                                ;BUT BE CAREFUL THAT THIS DOES NOT
                                ;TAKE VERY LONG OR CAUSE A LOOP.
        EXIT    1,              ;RETURN TO MONITOR
        MOVE    1,INTBLK+2      ;GET RETURN PC
        EXCH    1,TEMP1         ;RESTORE AC
        PUSH    P,INTBLK+2      ;SAVE RETURN ADDRESS
        SETZM   INTBLK+2        ;CLEAR INTERRUPT TO ALLOW ANOTHER ONE
        POPJ    P,              ;RETURN TO WHERE PROGRAM STOPPED

TEMP1:  Z                       ;TEMPORARY
```

The following example shows processing of a user CTRL/C by a program that prevents return to the monitor by means of a CTRL/C.

```
        LOC     134             ;SET UP .JBINT TO POINT TO
        EXP     INTBLK          ;THE INTERRUPT BLOCK
        RELOC

INTBLK: 3,INTLOC                ;3 WORDS LONG,,PLACE TO START
        XWD     0,2             ;NO MESSAGE CONTROL,,TYPE 2 (^C)
        Z                       ;GETS LAST USER PC
        Z                       ;LH GETS INTERRUPT TYPE
;THE INTERRUPT ROUTINE

INTLOC: SKIPL   RENFLA          ;OK TO FAKE A REENTER?
        JRST    .+3             ;NO, CURRENT ROUTINE CANNOT BE
                                ;INTERRUPTED.
        SETZM   INTBLK+2        ;YES, RE-ENABLE INTERRUPT AND GO
        JRST    REENRT          ;TO INTERRUPT ROUTINE

        SETOM   RENSWH          ;SET FLAG TO SAY "REENTER AS SOON AS
                                ;YOU CAN"
        PUSH    P,INTBLK+2      ;GET LAST PC, PUSH/POP
        SETZM   INTBLK+2        ;RE-ENABLE INTERRUPT
        POPJ    P,              ;GO BACK TO INTERRUPTED ROUTINE
                                ;NOTE THAT IF A CONTROL-C IS
                                ;TYPED AFTER THE SETZM, THE
                                ;INTERRUPTS NEST.
```

5.3 SOFTWARE INTERRUPT SYSTEM

The Software Interrupt System is a generalized mechanism for interrupting sequential program execution under a wide variety of conditions. An interrupt allows the system to respond dynamically to external conditions, and to requests for servicing error conditions. The monitor transfers control to a specified routine that services the interrupt. After interrupt servicing is complete, a transfer of control is made to the point of interruption (from which point normal execution will proceed).

The Software Interrupt System is initialized by the PIINI. monitor call. PIINI. allows the user program to specify the base address of an interrupt vector containing one or more four-word interrupt control blocks that control the operation of the Software Interrupt System. Note that the interrupt vector block must be in page 0. After initializing the Software Interrupt System, the user program must turn the system on with the PISYS. monitor call. This call also specifies

the conditions on which the user wishes control to be passed to an interrupt servicing routine, and

the location of the appropriate interrupt control block (specified as an offset from the base of the interrupt vector).

Interrupts occur after the execution of one instruction and before the execution of the next. When an interrupt condition occurs, the monitor first determines if this type of condition is to cause a transfer of control to an interrupt servicing routine. If a transfer is to take place, control will be transferred to the location specified in the appropriate interrupt control block. If not, the condition's default action will occur. Figure 5-1 charts the software interrupt process.

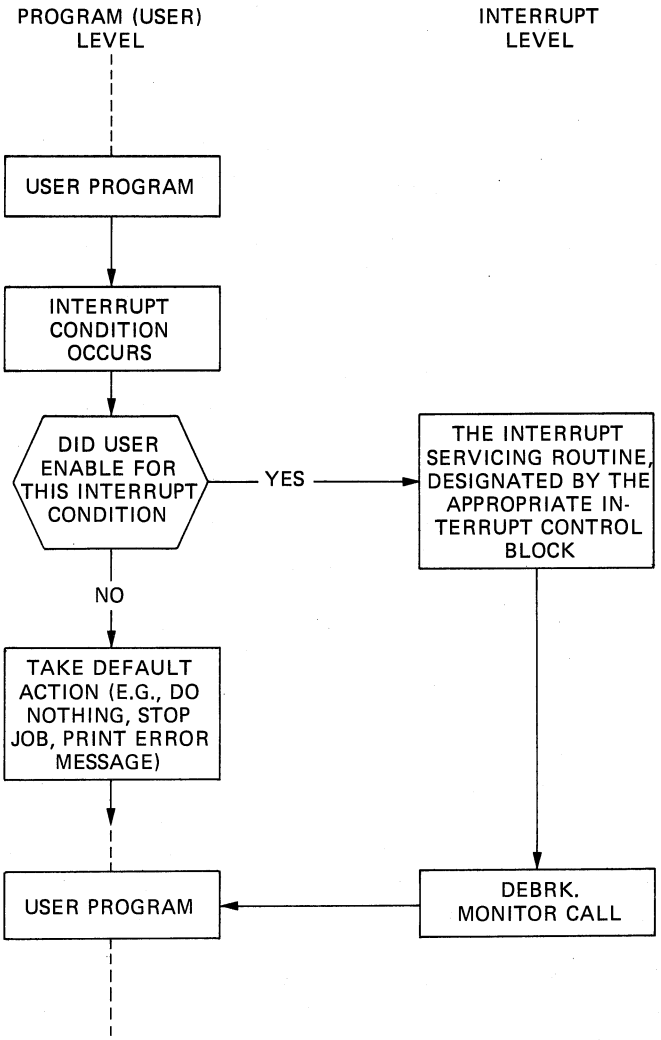


Figure 5-1. Software Interrupt Process

After an interrupt request has been granted, the program operates at interrupt level until the user issues a DEBRK. monitor call. DEBRK. dismisses the interrupt, re-enables the interrupt control block (if it was disabled), and causes any pending interrupt requests to be granted. If there are no pending interrupt requests, the user program will be restarted as though no interrupt had occurred.

The granting of an interrupt request does not change any of the conditions causing the interrupt. If a user program issues a DEBRK. monitor call without doing anything else, the result will be the same as if the interrupt condition was never enabled. However, any special action (e.g., stopping the job on a CTRL/C) is not taken. The monitor does not clear any reason bits on the DEBRK. monitor call; the user program must clear these bits. EXCEPTION: If the interrupt occurs while the monitor is executing a call for the user program, that call will be aborted. The only conditions which can cause interrupts during the processing of monitor calls are error conditions in the calls themselves. All other interrupt conditions are deferred until the monitor call exits.

5.3.1 Interrupt Conditions

The interrupt conditions that can be requested by a user program are divided into two categories: I/O interrupts and non-I/O interrupts. For any device, the user program can specify interrupt processing for one or more of the I/O conditions listed in Table 5-3.

Table 5-3
I/O Interrupt Conditions

Bit	Mnemonic	Meaning	Bit	Mnemonic	Meaning
19	PS.RID	Input done.	24	PS.RDO	Device off-line.
20	PS.ROD	Output done.	25	PS.RDF	Device full.
21	PS.REF	End-of-file.	26	PS.RQE	Quota exceeded.
22	PS.RIE	Input error.	27	PS.RWT	I/O wait.
23	PS.ROE	Output error.			

The non-I/O interrupt conditions are listed in Table 5-4.

5.3.2 Interrupt Control Block

The Interrupt Control Block is the controller of the Software Interrupt System. It keeps track of

the instruction that was last executed when an interrupt occurred,

The location of the interrupt servicing routine for processing the current interrupt, and

the reason for the current interrupt.

There may be more than one interrupt condition associated with the same interrupt control block, but the preferred usage is to associate one interrupt condition with one interrupt control block. An interrupt control block is represented in Figure 5-2.

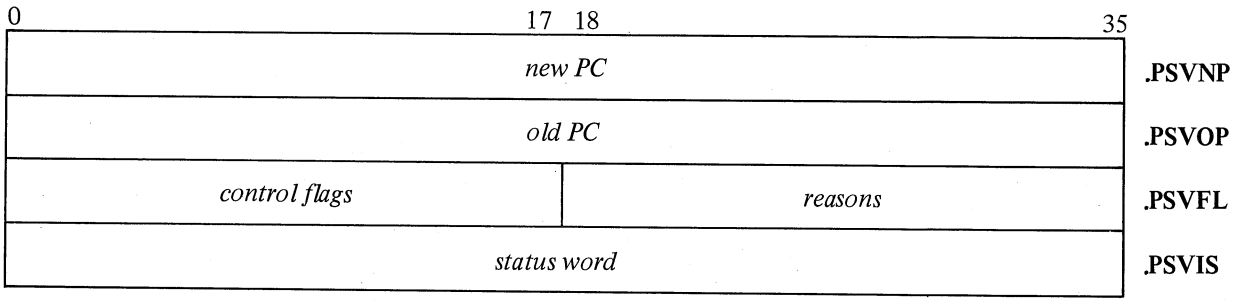


Figure 5-2. Interrupt Control Block

where: *new pc* is the location of the routine that will service the interrupt.

old pc is the current contents of the program counter (PC) at the time of the interrupt. If a monitor call is executed, old PC will contain the address of the call's return location (either error return or normal return). If an attempted monitor call is aborted, old pc will contain the address of the monitor call.

control flags are used to indicate the circumstances under which an interrupt is to occur (refer to Table 5-5).

reason is the type of interrupt condition that has occurred (refer to Table 5-3).

status word contains status information pertinent to the type of interrupt detected. This information is listed in Table 5-4 for those conditions that cause information to be returned in the status word.

Table 5-4
Non-I/O Interrupt Conditions

Code	Mnemonic	Interrupt Condition
-1	.PCTLE	The time limit for a job has been exhausted. (Applicable only for non-Batch jobs). The run time (in milliseconds) for the job is returned in the status word.
-2		Reserved for Digital.
-3	.PCSTP	A CTRL/C has been issued from a user terminal. If the terminal is in input-wait state when this interrupt occurred, bit 0 in the status word will contain a 1.
-4	.PCUUO	A monitor call is about to be processed. The status word contains the monitor call that was executed.
-5	.PCIUU	An illegal monitor call has been executed. The status word contains the illegal monitor call.
-6	.PCIMR	An illegal memory location has been referenced.
-7	.PCACK	An address check has occurred. The status word will contain the device name.
-10	.PCARI	An arithmetic exception has occurred.
-11	.PCPDL	A push-down list overflow has occurred.
-12		Reserved for Digital.
-13	.PCNXM	A non-existent memory location has been referenced.
-14	.PCAPC	The line frequency clock has ticked. The status word will contain the universal date/time word.
-15	.PCUEJ	A fatal error has occurred in the user's job.
-16	.PCXEJ	An external condition has caused a fatal error in the job.
-17	.PSKSY	A KSYS warning has occurred. The status word will contain the minutes left until KSYS.
-20	.PCDSC	The dataset status has changed. The status word will contain the new dataset status.
-21	.PCDAT	Either an ATTACH or a DETACH monitor call has been executed. If DETACH, the status word will contain a -1; if ATTACH, the status word will contain the TTY's UDX (universal device index).
-22	.PCWAK	A WAKE monitor call has been executed. The status word will contain the job number of the waker.
-23	.PCABK	An address break condition has occurred.
-24	.PCIPC	The job has received an IPCF packet in its input queue. The status word will contain the length of the packet in its right half, and the right half of the flags word in its left half.
-25	.PCRMC	Reserved.
-26	.PCQUE	An ENQ/DEQ resource is available for ownership. The request ID is returned in the status word.

5.3.3 Initialize the Software Interrupt System

The PIINI. monitor call (CALLI 136) initializes the software interrupt system. Its calling sequence is

```
MOVEI    ac,addr
PIINI.   ac,
    error return
normal return
```

where: *addr* is the base address of the interrupt vector block. Note that the interrupt vector block must be in page 0 on a VM system.

The PIINI. monitor call performs the following functions:

1. It turns off the software interrupt system.
2. It unlinks any devices with which enabled interrupt conditions are associated.
3. It stores the base address of the interrupt vector block.

Table 5-5
Control Flags

Bit	Mnemonic	Meaning
0		Reserved to Digital.
1	PS.VPO	Disable all interrupts until a PISYS. monitor call re-enables them.
2	PS.VTO	Disable all interrupts until a DEBRK. monitor call is executed.
3	PS.VAI	Allow additional interrupts to be received by this interrupt block. Normally, no other interrupts for the current block are permitted until a DEBRK. monitor call is executed. The use of this bit is not recommended since it could interrupt the service routine and therefore lose information.
4	PS.VDS	Dismiss any additional interrupt requests for this control block that are received while an interrupt is in progress. This bit is useful if the interrupt service routine wants to perform functions that would cause another interrupt.
5	PS.VPM	Print the standard message (if any relevant to this interrupt condition).
6	PS.VIP	This bit indicates that an interrupt is in progress for this block. The user should clear this bit at the start of the program. It is set and cleared by the monitor as interrupts are processed, and it should not be altered by the user.

5.3.4 Control the Software Interrupt System

The PISYS. monitor call (CALLI 137) allows a user program to control the Software Interrupt System. Its calling sequence is

```
        MOVE  ac,[flags,,addr]
        PISYS. ac,
           error return
        normal return

addr:   type
        vector-offset,,enabled reasons
        0,,0
```

where: *flags* may be set which control the software interrupt system (see Table 5-6).

addr points to the first word of the three-word interrupt argument block.

type (.PSECN) specifies a device or a condition to be associated with the interrupt. The type can be one of the following:

SIXBIT/*device-name*/
channel-number
UDX (universal device index)
a negative number specifying a non-I/O condition (see Table 5-3)

vector offset (left half of .PSEOR) is the offset from the base address of the four-word interrupt control block to be associated with this interrupt condition.

enabled reasons (right half of .PSEOR) specifies the type of interrupt desired. This half-word should be zero, if a device is not specified by type (refer to Table 5-3 and Table 5-4).

The PISYS. monitor call is the primary means by which the user program can control the software interrupt system. The call accepts a three-word argument block that specifies the type of condition the user wishes to service with an interrupt servicing routine. It also specifies the offset from the interrupt vector base address that points to the appropriate interrupt control block. Since each interrupt control block is four words long, the offset is always specified in multiples of four words.

Table 5-6
Argument Block Flags

Bit	Mnemonic	Meaning
1	PS.FOF	Turn off the interrupt system.
2	PS.FON	Turn on the interrupt system.
3	PS.FCP	Clear all pending interrupts.
4	PS.FCS	Clear all pending interrupts for a specified device.
5	PS.FRC	Remove the specified device or condition.
6	PS.FAC	Add the specified device or condition.

The possible error codes resulting from a PISYS. monitor call are listed in Table 5-7.

Table 5-7
PISYS. Error Codes

Code	Mnemonic	Meaning
0	PSTMA%	The right half of the AC is non-zero; no bits in the left half require an argument block.
1	PSNFS%	The left half of the AC does not have any function bits set.
2	PSUKF%	The left half of the AC contains function bits which have been set but have no defined meaning.
3	PSOOF%	The bits in the left half of the AC that turn the system on and off have been set.
4	PSUKC%	The contents of addr do not specify a valid address.
5	PSDNO%	The device specified by the contents of addr has not been INITed for this job.
6	PSPRV%	A restricted (illegal) condition has been specified.
7	PSIVO%	The vector table offset is too large or not a multiple of four words. A GETTAB table (Table number 11, item number 76) provides the maximum value that the vector offset may assume.
10	PSUKR%	An invalid bit was set in word 3 of the argument block; word 3 should be all zeroes.
11		Reserved.
12	PSNRW%	The reserved half-word (the right half of word three) is non-zero.
13	PSPND%	A PIINI. monitor call was not executed.
14	PSARF%	Both the 'add the device' bit and the 'remove the device' bit have been set.

5.3.5 Save the Interrupt Blocks

The PISAV. monitor call (CALLI 140) returns the entire monitor base related to the software interrupt system. The call can be used by modules such as QMANGR to save and reload (via PIRST.) the complete interrupt system. It can also be used to provide detailed error message reporting. The calling sequence for PISAV. is

```
MOVE  ac,[size,,addr]
PISAV. ac,
   error return
normal return
```

where: *size* is the length (in words) of the block pointed to be addr. The size of this block can be determined by the algorithm

(3 **number-of-argument-blocks*) + 2 = *size-in-words*

addr points to a block of three words. This block is represented in Figure 5-3.

0	1 … 17	18 … 35	
x	00000000000000000	*count*	.PSSFC
base address			.PSSIV
start-of-the-three-word-argument-blocks			.PSSBL

Figure 5-3. Saved Status Block Structure

where: *x* (PS.SON, bit 0) can be 1 or 0; 1 indicates that the software interrupt system is turned on; 0 indicates that it is off.

bits 1 through 17 must contain zero.

count is the number of words that the monitor actually returned with the saved status block.

base address is the address of the interrupt vector block which contains one or more four-word interrupt control blocks.

start points to the first location of the three-word argument blocks.

The location pointed to by the third word of the block represented in Figure 5-3 is the beginning of one or more argument blocks. The interrupt argument blocks are those that the user has set up by means of the PISYS. monitor call.

The possible errors resulting from the PISAV. monitor call are listed in Table 5-8.

Table 5-8
PISAV. Error Codes

Code	Mnemonic	Meaning
0	PSBTS%	The block is too small to hold the data. The right half of the first word contains the count of the number of words which would have been returned, if the block had been long enough.

5.3.6 Reload the Saved State of the Interrupt System

The PIRST. monitor call reloads the saved state of the software interrupt system. This call does not, however, remember any pending interruptions. If the interrupt control block has not been cleared of its condition, the interrupt will be granted. The PIRST. monitor call should not be used to load the interrupt system of program initialization time; this function is performed by the PIINI. monitor call. The calling sequence for PIRST. is

```
MOVEI  ac,addr
PIRST. ac,
   error return
normal return
```

where: *addr* is the address of the saved status block specified in the PISAV. monitor call.

The possible error codes resulting from a PIRST. monitor call are listed in Table 5-9.

Table 5-9
PIRST. Error Codes

Code	Mnemonic	Meaning
0	PSNRS%	The user program has been modified to prevent the PIRST. monitor call from performing its specified task.

5.3.7 Dismiss an Interrupt

The DEBRK. monitor call dismisses a software interrupt, re-enabling anything which may have been disabled by the occurrence of the interrupt. The calling sequence for the DEBRK. call is

```
DEBRK.
   return 1
   return 2
```

The DEBRK. call normally returns to old PC. *Return 1* is taken if DEBRK. is not implemented; *return 2* is taken if there was no interrupt in progress. The DEBRK. call scans the pending interrupt queue, looking for any conditions which may require servicing by an interrupt servicing routine. If such a condition exists, its interrupt request will be granted, and a transfer will be made to the interrupt servicing routine. If there are no pending interrupts, DEBRK. will restart the interrupt process beginning at the point within the user job where interruption occurred (e.g., the instruction after the last instruction executed).

5.3.8 An Example of the Software Interrupt System

```
TITLE PISAMP -- SAMPLE PROGRAM TO SHOW PSISER USE WITH NON-BLOCKING I/O

;THIS PROGRAM WRITES A FILE CONTAINING THE NUMBERS 1 TO 100,000
; WHILE DOING A COMPUTE-BOUND BACKGROUND COMPUTATION. BECAUSE THE PROGRAM
; NEVER BLOCKS FOR I/O, IT CAN USE 100% OF THE AVAILABLE CPU TIME.  BY USING
; THE PI SYSTEM IT CAN DRIVE THE DISK AT FULL SPEED.

;AC USEAGE
        T1=1                    ;TEMPORARY
        N=2                     ;NUMBER TO WRITE ON THE DISK

;I/O CHANS.
        DSK=1                   ;THE DISK FILE

        SEARCH  C               ;SYMBOL DEFS.

;INITIALIZATION
```

```
START:  RESET                       ;RESET THE WORLD
        MOVEI   T1,VECTOR           ;BASE OF INTERRUPT VECTOR
        PIINI.  T1,                 ;INIT PI SYSTEM
          HALT                      ;NOT IMPLEMENTED
        OPEN    DSK,[UU.AIO+.IOBIN;OPEN DISK FOR ASYNCHRONOUS BINARY
                OUTPUT
                   SIXBIT /DSK/
                   OB,,0]
          HALT                      ;DISK NOT AVAILABLE
        ENTER   DSK,[SIXBIT "SAMPLE" ;ENTER THE OUTPUT FILE
                   SIXBIT "BIN" ;  ON THE DISK
                   EXP 0,0]         ;
          HALT                      ;CAN'T WRITE

        MOVE    T1,[PS.FAC+[ EXP DSK
                    4,,PS.ROD       ;OFFSET,,OUTPUT DONE
                    0]]             ;PRIORITY,, RESERVED
        PISYS.  T1,                 ;CALL MONITOR TO TURN ON SYSTEM AND
                                    ;  ENABLE FOR OUTPUT DONE ON CHAN DSK
          HALT                      ;PISYS. UUO FAILED
        MOVEI   N,0                 ;PRESET N

;HERE ON AN OUTPUT DONE INTERRUPT OR AT THE START OF THE PROGRAM

OUTDON: SOSGE   BYTECT              ;ROOM IN THIS BUFFER?
        JRST    DUMPBF              ;NO--GO OUTPUT BUFFER
        IDPB    N,BYTEPT            ;STORE IN BUFFER
        CAME    N,[^D100000]        ;DONE?
        AOJA    N,OUTDON            ;NO--WRITE NEXT NUMBER
        CLOSE   1,
        EXIT                        ;ALL DONE

DUMPBF: OUT     1,                  ;WRITE OUT THE BUFFER
          JRST  OUTDON              ;NO ERRORS AND MORE BUFFERS
        STATZ   1,IO.ERR            ;ANY ERRORS?
          HALT                      ;FATAL I/O ERROR
;AT THIS POINT WE FILLED ALL AVAILABLE BUFFERS AND WANT TO GO BACK TO THE
;  BACKGROUND TASK.
        DEBRK.                      ;DISMISS THE INTERRUPT
          HALT                      ;CAN NEVER GET HERE
;IF WE GET HERE THERE WAS NO INTERRUPT IN PROGRESS. THAT MEANS WE WERE
;  CALLED BY INITIALIZATION AND NOW MUST START THE BACKGROUND TASK.
        MOVSI   T1,(PS.FON)         ;TURN ON THE PI SYSTEM SO WE CAN GET
        PISYS.  T1,                 ;  TRAPS OUT OF THE BACKGROUND TASK.
          HALT                      ;CAN'T TURN ON SYSTEM
        MOVEI   T1,0
        AOJA    T1,.                ;SUPER SIMPLE BACKGROUND TASK

;BUFFER RING HEADER
OB:     BLOCK   1
BYTEPT: BLOCK   1                   ;BYTE POINTER
BYTECT: BLOCK   1                   ;BYTE COUNT

;INTERRUPT VECTOR
VECTOR: BLOCK   4                   ;FIRST SLOT IS UNUSED
        EXP     OUTDON              ;NEW PC
        EXP     0                   ;OLD PC STORED HERE
        EXP     0                   ;FLAGS
        EXP     0                   ;STATUS

        END     START
```

CHAPTER 6

CORE AND SEGMENT CONTROL

6.1 CORE CONTROL

For various reasons, users may wish to lock privileged jobs into core, so that they are never considered for swapping and shuffling. Some examples of these jobs follow.

Real time jobs	which require immediate access to the processor in response to an interrupt from an I/O device.
Display jobs	which must be refreshed from a display buffer in the user's core area in order to keep the display picture flicker-free.
Performance Analysis Jobs	so that they can be invoked quickly with low overhead in order to record the activities of the monitor.

6.1.1 Definitions

Unlocked jobs occupy only those physical core locations not occupied by locked jobs. Therefore, locked jobs and timesharing jobs contend with one another for physical core memory. In order to control this contention, the system administrator is provided with a number of system parameters which are described below.

Total User Core	is the physical core that can be used for both locked and unlocked jobs. This value equals the total physical core minus the monitor size.
CORMIN	is the amount of contiguous (on a KA10) core guaranteed a single unlocked job. This value is a constant system parameter defined by the system administrator at monitor generation time (via MONGEN). This value can range from 0 to *TOTAL USER CORE*. CORMIN may be changed by the system administrator through the use of the SET CORMIN command (refer to *DECsystem-10 Operating System Commands*) or the SETUUO monitor call.
CORMAX	is the largest (on a KA10) contiguous block allowed an unlocked job. It is a time-varying system parameter that is reduced from its initial setting as jobs are locked in core. In order to satisfy the guaranteed size of CORMIN, the monitor never allows a job to be locked in core if it would cause CORMAX to be less than CORMIN. The initial setting of CORMAX is defined at monitor generation time (via MONGEN), and can be changed with the SET CORMAX command (refer to *DECsystem-10 Operating System Commands*) or the SETUUO monitor call.

6.1.2 The LOCK Monitor Call (CALLI 60)

The LOCK monitor call provides a mechanism for locking jobs into user memory. The user may specify that the high, the low, or both segments are to be locked, and if the core is to be physically contiguous or not. Note that on KA10-based systems, core is always allocated contiguously, and that the job will be moved to an extremity of user core before it is locked. The calling sequence for the LOCK monitor call is:

```
MOVE  ac, [XWD hi-code, lo-code]
LOCK  ac,
   error return
normal return
```

where: *hi-code* and *lo-code* are the high and low segment codes – a series of bits that specify the way in which the high segment (left half code) and the low segment (right half code) are to be locked. The order and the position of the bits in the left half correspond to the order and the position of the bits in the right half (i.e., to obtain the bit number for the high segment, subtract 18 from the corresponding bit for the low segment). The possible bits that may be set with the LOCK monitor call are listed in Table 6-1.

Table 6-1
LOCK Bits

Bit	Mnemonic	Meaning
17 35	LK.HLS LK.LLS	If the bit contains 1, lock the high (or low) segment as indicated by bits 15 or 16 (33 and 34).
16 34	LK.HNE LK.LNE	If the bit contains 0, map the high (or low) segment contiguously in the exec virtual memory (always implied on a KA10-based system). This action causes the segment to be added to the exec virtual address space so that it can be executed in exec mode. For example, this would be required when exec mode real-time trapping is used. On KI10/KL10-based systems, the amount of exec virtual address space used by locked jobs is a limited resource with a defined maximum limit per processor. If mapping of the segment would cause the maximum to be exceeded, the LKMEM% error code will be returned in the AC, and the error return will be taken. The maximum exec virtual memory available for the LOCK monitor call can be obtained from GETTAB table .GTC*n*V, item number 43 (%CEVM), where *n* is the number of the CPU. The current amount used can be obtained from .GTC*n*V, item number 44 (%CVEVU).
15 33	LK.HNP LK.LNP	If the bit contains 1, do not attempt physical continuity. If the bit contains 0, lock the high segment (low segment) into contiguous physical memory locations (always implied on KA10 based systems). This action causes the high segment (low segment) to be moved and remapped (if necessary) so that its physical core is contiguous. On KA10-based systems, the high segment (low segment) is also moved to one end of user core, in order to minimize fragmentation of memory.

On a normal return, the job is locked into core. If there is a high segment, the left half of the AC will contain its physical core address, or virtual address if meaningful (in units of 518_8 word pages). This value can be converted to a word address by shifting it left nine bits. If there is no high segment, the left half of the AC will contain zero. The right half of the AC will contain the page number of the low segment, which can be shifted nine bits. If the job is locked in place so that no physical or virtual contiguity is implied (KI10 and KL10 only), the contents of the AC on a normal return will be zero.

On an error return, the job is not locked into core, and the AC is either unchanged or contains an error code. An error code indicates the condition that prevented the job from being LOCKed. The possible error codes are listed in Table 6-2.

A job may be locked into core if all of the following conditions are satisfied:

1. The job has the LOCK privilege (bit 14, JB.LCK) in the privilege word which is set from the accounting file, ACCT.SYS, by LOGIN.
2. The job, when locked, would not prevent another job from expanding to the guaranteed limit (CORMIN).

3. The job, when locked, would not prevent an existing job from running. Note that unlocked jobs can exceed the value of CORMIN.
4. The job, when mapped, if specifying exec mapping, would not exceed the maximum amount of exec virtual address space available for locking (KI10/KL10-based systems only).
5. The job has a non-sharable high segment when running virtual on a KI10/KL10-based system.

Table 6-2
LOCK Monitor Call Error Codes

Bit	Mnemonic	Meaning
0	LKNIS%	The monitor call is not included in this system, or the requested function is not implemented because it has not been defined with MONGEN or because the appropriate feature test switch has been turned off.
1	LKNLP%%	The job requires locking privileges, but does not have them.
2	LKNCA%	If the job were locked into core, it would be impossible to run the largest existing non-locked job.
3	LKNCM%	If the job were locked into core, it would be impossible to sustain the guaranteed maximum for unlocked jobs.
4	LKNEM%	The allowable amount of exec virtual memory has been exhausted.
5	LKNIA%	An illegal subfunction has been specified.
6	LKNPU%	The specified page is unavailable.

If a user program requests a segment to be locked into contiguous physical memory, the monitor will attempt to lock it into physical memory at the lowest location possible. When the segment is locked below 128K, physical and virtual contiguity are equivalent; therefore, virtual contiguity does not always require the exec virtual memory resource to achieve contiguity.

On KA10-based systems, physical memory is always allocated contiguously, and user segments are directly addressable in exec mode (bits 15, 16, 33, and 34 are ignored).

Clearing bits 33 and 34 (bits 15 and 16 for the high segment) in KI10/KL10-based systems is compatible with the implementation of the LOCK monitor call for KA10-based systems. Code 1 is the most restrictive code, allowing a program coded for a KA10-based system that implicitly uses these properties to also be executed on a KI10/KL10-based system. Applications that do not require all properties can add the appropriate bits the LOCK monitor call's calling sequence.

Although memory fragmentation is minimized by the LOCK monitor call and the shuffler, the locking algorithm always allows job-locking even though severe fragmentation may take place, as long as

1. all existing jobs can continue to run, and
2. CORMIN, at least, is available (refer to Figure 6-1).

Since memory fragmentation can degrade system throughput, system administrators must use caution when granting locking privileges. Section 6.1.2.1 lists guidelines for minimizing fragmentation when using the LOCK monitor call on KA10-based systems.

NOTE

The CORE monitor call may be given for the high segment of a locked job only to remove the high segment from the addressing space. When any segment is locked into core, neither the CORE monitor call nor the CORE

command with a non-zero argument can be satisfied because they will cause an error. Before executing the LOCK monitor call, the program should determine and request the amount of core needed for execution.

6.1.2.1 The LOCK Monitor Call Extension – (KI10/KL10-based systems only) the extension to the LOCK monitor call locks a segment into a specified page of physical core memory. Its calling sequence is

```
        MOVE  ac, [XWD -n,addr]
        LOCK  ac,
           error return
        normal return

addr:   function
        argument 1
        .
        .
        .
        argument i
```

where: *n* is the number of arguments plus one (i.e., i + 1 = n).
addr points to the first word of the argument block.
function is one of the function codes described in Table 6-3 (currently, one function is implemented).
argument 1 ... argument i is different depending on the function used; all possible arguments are listed in Table 6-6.

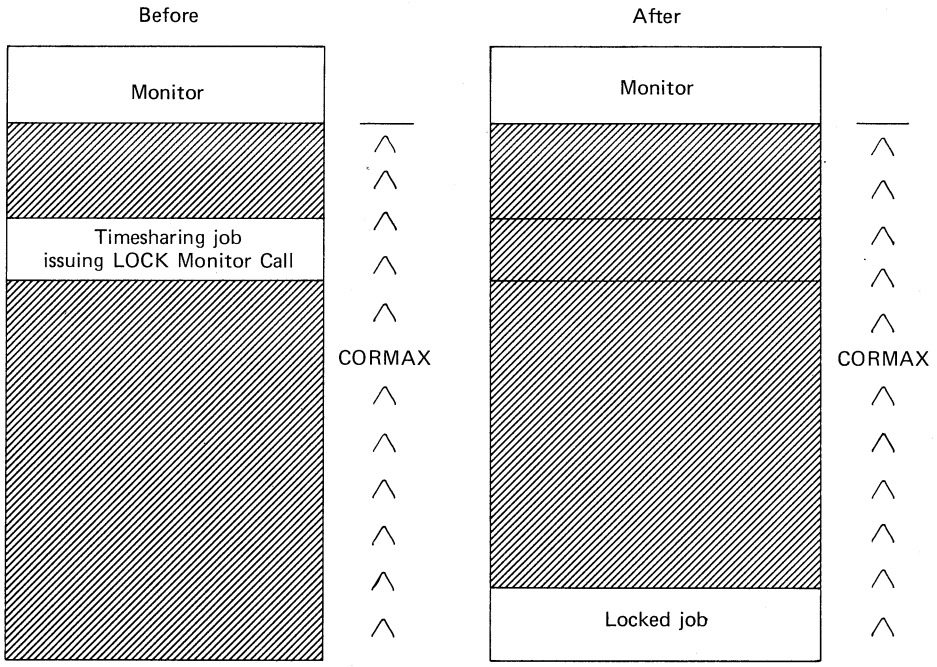


Figure 6-1 Locking Jobs in Core on KA10 Systems

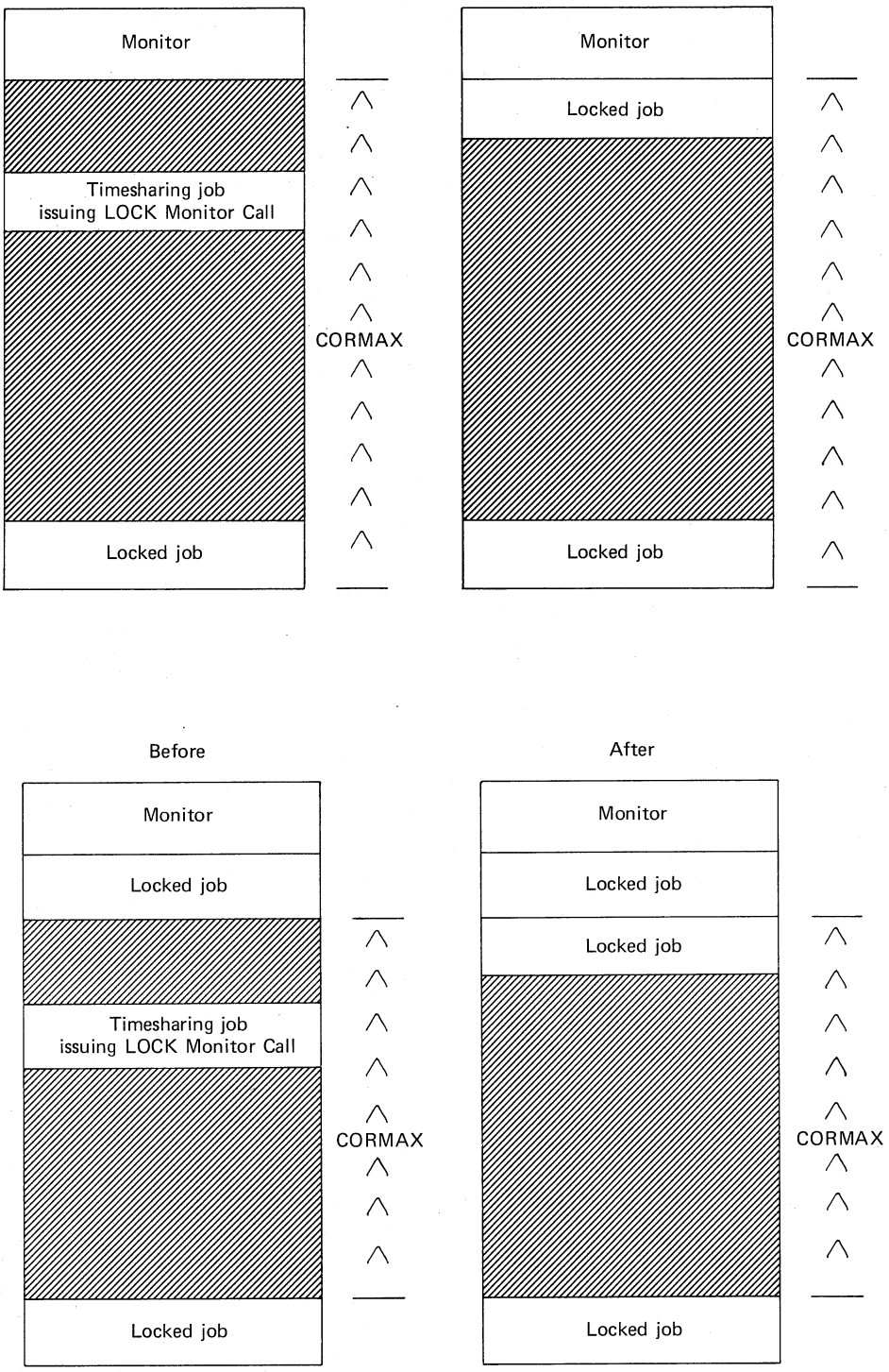


Figure 6-1 (Cont.) Locking Jobs in Core on KA10 Systems

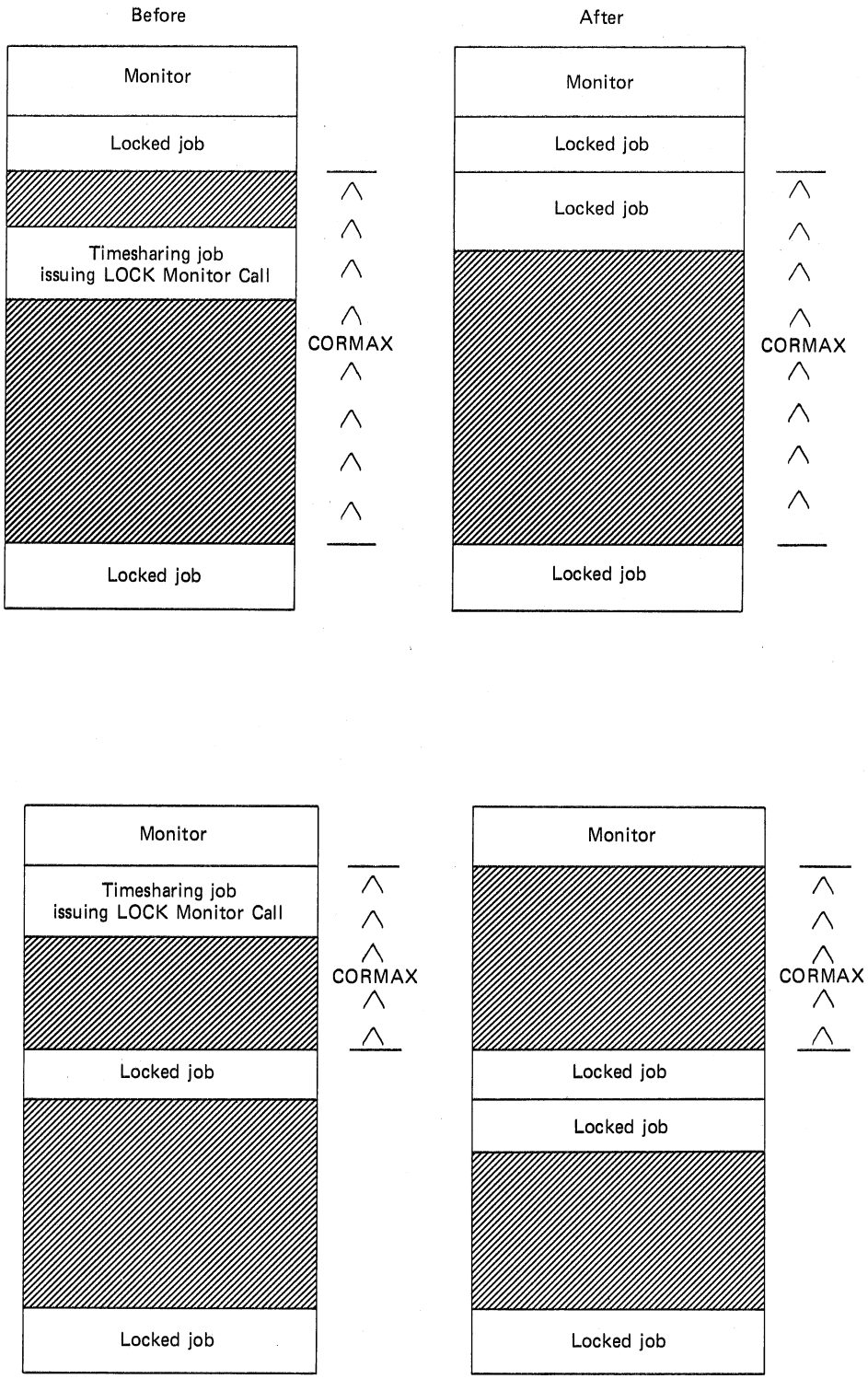


Figure 6-1 (Cont.) Locking Jobs in Core on KA10 Systems

Table 6-3
LOCK Extension Functions

Function	Mnemonic	Meaning	Argument
0	.LKPPN	Lock the high segment and/or low segment into contiguous physical pages, starting at the physical page number specified in the argument.	LH = 0 Do not lock the high segment LH = *n* lock the high segment starting with page number n RH = 0 Do not lock the low segment. RH = *n* lock the low segment starting with page number n.

On a normal return, the segment is locked (physically and contiguously) starting at the specified physical address. On an error return, error code 6 (LKNPU%) is returned in the AC (refer to Table 6-2). The error return will be taken when locking the segment at the pages specified in the argument would cause any of the following conditions to be true.

1. when locked, the two segments would overlap.
2. when locked, one or both segments would overlap another locked job.
3. when locked, one or both segments would overlap the monitor.
4. when locked, one or both segments would be outside the range of on-line memory.

Note that if the monitor call indicates that the low segment is to be locked, the physical page number specified in the right half of argument i is where the low segment is locked into the next higher physical page location, an example of the extended LOCK monitor call is

```
        MOVE    AC, [XWD -2,ADDR]
        LOCK    AC,
          JRST ERROR
        .
        .
        .
ADDR:   0
        230,,224
```

In the example above, the high segment will be locked into core starting with page 230 and the low segment will be locked into core starting with page 224.

6.1.2.2 Minimizing Fragmentation – For KA10-based systems, the guidelines for minimizing fragmentation when using the LOCK monitor call are listed below.

1. If two or fewer segments are locked into core, there will be no memory fragmentation.
2. If the locked jobs do not relinquish their locked status (i.e., no job that has issued the LOCK monitor call terminates), there will be no memory fragmentation. In general, only production jobs should be granted locked privileges.
3. If a job issuing a LOCK monitor call is to be debugged, and production jobs with locking privileges are to be executed, the job to be debugged should be initiated and locked into core first. Locking this job first will place it at the top of core, reserving this area for it and guaranteeing that as the job locks and unlocks there will be no fragmentation.

4. By appropriately setting CORMIN and the initial setting of CORMAX in relationship to TOTAL USER CORE, the system administrator can establish a policy that guarantees the following.
   a. A maximum size for any unlocked job (CORMIN).
   b. A minimum amount of total lockable core for all jobs (TOTAL USER CORE minus CORMAX).
   c. The amount of core for which locked and unlocked jobs can contend on a first-come/first-serve basis (TOTAL USER CORE equals the initial setting for CORMAX plus CORMIN).

6.1.3 The UNLOK. Monitor Call (CALLI 120)

The UNLOK. monitor call provides a mechanism for a job to unlock itself without having to perform a RESET monitor call. The user program specifies whether the high, the low, or both segments are to be unlocked from core. The calling sequence for UNLOK. is

```
MOVSI   ac,n
HRRI    ac,m
UNLOK.  ac,
   error return
normal return
```

where: *n* is specified if the high segment is to be unlocked (LH not equal to 0).
m is specified if the low segment is to be unlocked (RH not equal to 0).
n and *m* are specified if both the high and low segments are to be unlocked.

The error return is taken if the monitor call has not been implemented. Under this circumstance, a job can relinquish its locked status by executing a RESET or EXIT monitor call; locked status is relinquished also when the monitor implicitly executes a RESET for the user program. An implicit RESET will occur when

1. The user program issues a RUN monitor call, or
2. The user types any one of the following operating system commands: R, RUN, GET, SAVE, SSAVE, CORE, NSAVE, NSSAVE, OSAVE, OSSAVE, or a system program-invoking command.

On a normal return, the segment (or job) is unlocked and becomes eligible for swapping and/or shuffling. Any meter point (METER monitor call) is deactivated and, if the low segment is unlocked, any real-time device will be reset. CORMAX is increased to reflect the new size of the largest contiguous region available to unlocked jobs. CORMAX, however, will never be set higher than its initial value.

NOTE

A locked high segment shared by several jobs is unlocked only when the SN%LOK bit is turned off for all of those jobs (i.e., when all jobs which executed the LOCK monitor call have performed the UNLOK. monitor call). Refer to GETTAB Table Number 14.

6.1.4 THE CORE MONITOR CALL (CALLI 11)

The CORE monitor call allows a user program to expand and contract its core allocation in either one or both segments, as its memory requirements change. Its calling sequence is

```
MOVE   ac,[XWD hi-seg-addr, low-seg-addr]
{CORE  ac,        }
{CORE  ac,200000  }
    error return
normal return
```

where: *hi-seg-addr* is the highest user address (end point) to be assigned to the high segment. If hi-seg-addr is zero, the current core assignment for the high segment will be unchanged.
low-seg-addr is the highest user address (end point) to be assigned to the low segment. If low-seg-addr is zero, the current core assignment for the low segment will be unchanged.
The argument 200000 sets bit 19 (UU.PHY) to indicate that the core assignment is for physical core. (This argument is meaningful only for KI10/KL10-based systems with virtual memory.)

The monitor will always assign the smallest amount of core that will satisfy the request. To ensure privacy of all information, all of core is cleared before assignment to the user program.

On KA10-based systems, the high segment address is relative to location 400000 or page 400. If the address specified in the left half of the AC is smaller than either of these values, the high segment will be destroyed. If the value is greater than either of these values, the new high segment end address will be that specified.

On KI10/KL10-based systems the user may start the high segment at any location by using the REMAP monitor call (a LINK-10 switch may also be used, refer to the *LINK-10 Programmer's Reference Manual*). When the CORE monitor call is issued, the left half of the AC is always relative to the starting address of the high segment specified in the REMAP monitor call.

On a normal return, the information returned in the AC is listed in Table 6-4.

Table 6-4
Values Returned from a CORE Monitor Call

Type of System	Information Returned
Non-VM	The maximum amount of 1K core blocks (all of core minus the monitor, unless an installation chooses to restrict the amount of core available to the user. (CORMAX)
VM Systems	The current virtual memory limit in 1K core blocks assigned to the user.

On KI10/KL10-based systems, if the left half of the AC is non-zero, and it is either less than 400000 or the length of the low segment (whichever is greater), the high segment will be eliminated. If this call is executed from the high segment, an illegal memory error message will be printed when the monitor attempts to return control to the new illegal address.

To increase the low segment and decrease the high segment simultaneously, two CORE monitor calls should be issued to reduce the chances of exceeding the maximum size allowed to a user.

The error return is taken on the following conditions:

1. The left half of the AC is greater than or equal to 400000, and the user has been meddling without write-access privileges.
2. The left and the right halves of the AC are both zero.
3. The sum of the sizes of the new low segment and the old high segment is greater than the maximum amount of core allowed to a user. (The core assignment will be unchanged, and the maximum core available for high and low segments is returned in the AC for swapping systems.
4. The sum of the new low segment and the new high segment is greater than the maximum amount of core allowed to a user. (The core assignment will be unchanged.) The maximum core available for the user is returned in the AC in 1K core blocks.
5. The right half of the AC specifies an argument which would cause the low segment to overlay (expand into) the high segment.

If the high segment is eliminated by the CORE monitor call, a subsequent CORE with the left half of the AC greater than 400000 will create a new, non-sharable segment instead of re-establishing the old high segment. This segment will become sharable if it has been

1. given the filename extension .SHR,
2. written onto a storage device,
3. closed so that a directory entry will be made, and
4. initialized from the storage device by either of the following commands: GET, R, RUN, or the RUN or GETSEG monitor calls.

LINK-10 and the SSAVE and GET commands use the above sequence to create and initialize new sharable segments.

A user program that expands core should compare its highest desired address with its highest legal address, which can be obtained from location .JBREL in the Job Data Area (refer to Chapter 3). If the desired address is greater than the highest legal address, the program should execute a CORE monitor call for the new desired address (i.e., not for the highest old legal address plus 512 or 1024).

The monitor will update .JBREL by a basic allocation unit (i.e., 1024 words for KA10-based systems, and 512 words for KI10/KL10-based systems). Subsequent comparing of the desired address and the highest legal address do not cause a CORE monitor call until the next increase of core is required. If used in this manner, the CORE monitor call will execute the same on all three processors and will require less monitor CPU time, because of the number of COREs needed will be minimized.

The following example illustrates the method of obtaining core only when it is needed.

```
;SUBROUTINE TO GET CORE ONLY WHEN NEEDED
;CALL:   MOVE    T1,HIGHEST DESIRED ADDRESS
;        PUSHJ   P,CHKCOR
;        RETURN HERE UNLESS NO MORE CORE
CHKCOR:  CAMG    T1,.JBREL##        ;GREATER THAN HIGHEST LEGAL ADDRESS?
         POPJ    P,                 ;NO, PRESET CORE BIG ENOUGH.
         CORE    T1,                ;YES, GET NEXT INCREMENT OF CORE.
           JRST  ERROR              ;NOT AVAILABLE.
         POPJ    P,                 ;NEXT INCREMENT ASSIGNED.
```

6.1.5 The SETUWP Monitor Call

The SETUWP monitor call allows a user program to set or clear the hardware user-mode write-protect bit and to obtain the previous setting for this bit. The call must be used if the program is to modify its high segment; its calling sequence is

```
MOVEI  ac,bit
SETUWP ac,
   error  return
normal return
```

where: *bits* is used to specify the setting of the user-mode write-protect bit in bit 35 of the AC (write-protect = 1 and write-privileges = 0).

The previous setting of the user-mode write-protect bit is returned in bit 35 of the AC, so that any user subroutine can preserve the previous setting before changing it. Nested subroutines that either set or clear this bit can be written, providing that the subroutines save the previous value of the bit and restore it on returning to the caller.

6.1.6 The PAGE. Monitor Call

The PAGE. monitor call (only available on KI1-/KL10-based systems with virtual memory) manipulates pages and the data associated with those pages. Its general calling sequence is

```
        MOVE  ac, [XWD function,addr]
        PAGE. ac,
           error return
        normal return

addr:   number-of-words
        argument 1
        .
        .
        .
        argument n
```

where: *function* is a PAGE. function code in the range 0 to 7; the possible function codes for PAGE. are listed in Table 6-5.
addr points to the first word in the argument block.
number-of-words is the number of arguments in the argument block.
argument is different depending on the function code specified; all arguments are described in Table 6-5 pertinent to the given function code.

On an error return, an error code will be returned in the AC. The possible error codes are listed in Table 6-7. On a normal return, the function specified has occurred. There are two limitations in regard to what pages may be paged out: page zero may never be paged out; and, if the high segment is sharable, none of the high segment pages may be paged out.

Table 6-5
PAGE. Monitor Call Functions

Function	Mnemonic	Meaning/Argument Format
0	.PAGIO	Swaps a page in or out. Pages already allocated are swapped in and added to the working set. Pages are deleted from the working set in core and moved to secondary storage. The argument words are set up as follows. Bit 0 = 0 Swap-in = 1 Swap-out Bit 1 = 0 Transfer to fast secondary storage = 1 Create page on the disk Bits 2-26 = 0 Bits 27-35 = Page number Multiple entries in the argument block must specify page numbers in increasing numeric order.

Table 6-5 (Cont.)
PAGE. Monitor Call Functions

Function	Mnemonic	Meaning/Argument Format
1	.PAGCD	Creates or destroys a page. The argument word format takes the following format. Bit 0 = 0 Create a page = 1 Destroy a page Bit 1 = 0 Create a page in the working set = 1 Create a page on disk Bits 2-26 = 0 Bits 27-35 = Page number Multiple entries in the argument block must specify page numbers in increasing numeric order.
2	.PAGEM	Moves or exchanges a page. A page is moved from one location to another location in user virtual address space, or two pages exchange locations. Note that a page cannot be moved to a location already allocated to another page. The argument word format is in the following format. Bit 0 = 0 Page is moved = 1 Page is exchanged Bits 1-8 = 0 Bits 9-17 = Source page location Bits 18-26 = Zero Bits 27-35 = Destination page location
3	.PAGAA	Sets or clears the access-allowed bit. If a page is a part of the working set, its access bit can be turned off. When the access bit is off, a page fault will occur the next time that page is accessed. The page will remain in core, the access bit may be turned on at time, therefore. The argument word format is Bit 0 = 0 The bit is set = 1 The bit is cleared Bits 1-26 = 0 Bits 27-35 = Page number Multiple entries in the argument block must specify page numbers in increasing numeric order.

Table 6-5 (Cont.)
PAGE. Monitor Call Functions

Function	Mnemonic	Meaning/Argument Format
4	.PAGWS	Returns a bit map indicating those pages in the current working set. In the PAGE. call, number-of-words specifies the number of words in the bit map that are to be returned (normally 17 for the entire map. There is one bit for each possible page (0-511). If a bit is set, the page associated with that bit is a part of the working set. For example, word 1 contains the bits associated with pages 0 through 35; word 2 contains the bits associated with pages 36 through 71, etc.
5	.PAGGA	Returns a bit map indicating which pages have their access-allowed bits set. This bit map has the same format as the one returned for function code 4. If a bit in the map is set, the page associated with that bit is accessible. In the PAGE. monitor call, number-of-words specifies the number of the words in the bit map that are to be returned (normally 17 for the entire map).
6	.PAGCA	Determines the type of access allowed for a given page. There is no argument block; instead, the AC (bits 0-17) contain the function code and bits 18-35 contain the page number. On a normal return, one or more of the bits described in Table 6-6 will be set.
7	.PAGCH	Create a high segment.

Table 6-6
Bits Returned from Function .PAGCA

Bit	Mnemonic	Meaning
0	PA.GNE	The page specified does not exist.
1	PA.GWR	The page specified is writable.
2	PA.GRD	The page specified is readable.
3	PA.GAA	The page specified is accessible.
4	PA.GAZ	The page specified is allocated but zero.
5	PA.GCP	The page specified cannot be paged out.
6	PA.GPO	The page specified has already been paged out.

Table 6-7
PAGE. Monitor Call Error Codes

Error Code	Mnemonic	Meaning
1	PAGUF%	An unimplemented function has been specified.
2	PAGIA%	An illegal argument has been specified.
3	PAGIP%	An illegal page number has been specified.
4	PAGMF%	A page must exist, but it does not.
5	PAGMI%	A page must be in core, but it is not.
6	PAGCI%	A page cannot be in core, but it is.
7	PAGSH%	A page is in a sharable high segment.
10	PAGIO%	A paging I/O error has occurred.
11	PAGNS%	No swapping space is available.
12	PAGLE%	Specified function is illegal when the job is locked in core.
13	PAGIL%	Cannot create the specified page (an attempt was made to create a page on the disk).
24	PAGNX%	Cannot create the specified page (an attempt was made to create a page on the disk) with virtual memory limit to zero.

6.1.6.1 Page Fault Handling – When an executing program (on KI10/KL10-based systems with virtual memory) references a location in a noncore-resident page, the system transfers control to a page fault handler. The page fault handler determines which pages are to be placed in core during program execution. A user program can utilize its own own page fault handler, or, if the program does not include its own, the default page fault hander will be used.

The default page fault handler utilizes a modified FIFO technique in swapping pages in and out of core. An ordered list of core-resident pages is maintained by age of page in physical core. Each page has an access-allowed bit associated with it; the access-allowed bit can be set to 1 indicating that the page is accessible, or to 0 indicating that the page is not accessible. Periodically, the page fault handler will set every physical page's access-allowed bit to 0. A page fault will occur the next time one of these inaccessible pages is referenced; after the reference is made, the access-allowed bit is set to 1, and execution continues.

By use of the age-ordered list, along with a periodic check on the access-allowed bit, pages are swapped on a modified first-in/first-out basis.

The page fault handler controls the entire working set of pages, and is a part of the user's core image. If the user program does not supply its own page fault handler, the default page fault handler will be read into the top of the user's address space from SYS:PFH.VMX. If a user-supplied handler is to be used, .JBPFH in the Job Data Area must point to it. In .JBPFH, bits 9-17 contain the page fault handler's and address; bits 18-35 contain its start address. Before the occurrence of the first page fault, the user program must have ensured that .JBPFH contains these two addresses.

Alternatively, the user program may cause a page fault handler to be obtained from its directory by ASSIGning DSK: as SYS:, in which case, PFH.VMX will be loaded from the user's disk area.

If the user supplied page fault handler is deleted because of a CORE or PAGE, monitor call, the default page fault handler (i.e., SYS:PFH.VMX) will be brought into core on the next occurrence of a page fault.

6.1.6.2 Format of the Page Fault Handler – The page fault handler must be in the form

```
PFH: JRST START
     pc for fault
     page fault word
     virtual time
     current page rate
     0
     0
     0

START:
```

where: *PC for fault* contains the program counter location when the fault occurred. This value is filled in by the monitor when a page fault occurs.

page fault word is filled in by the monitor when a page fault occurs; refer to Table 6-8.

virtual time since the page fault handler was first brought into core is filled in by the monitor when a page fault occurs.

current page rate is filled in by the monitor when a page fault occurs.

Table 6-8
Page Fault Word

Bit(s)	Mnemonic	Meaning
0	**PF.HCB**	Bit 0 = 1 indicates that the working set of pages has been changed by the monitor or the user program without the page fault handler's control.
1-17	**PF.HPN**	These bits contain the number of the page causing the page fault.
18-35	**PF.HFC**	These bits contain a code indicating the type of page fault that has occurred.

Code	Mnemonic	Meaning
1	.PFHNA	The access-allowed bit has been turned off; the page is not accessible.
2	.PFHNI	The referenced page has been swapped out; the page is not in core.

Table 6-8 (Cont.)
Page Fault Word

Bit(s)	Mnemonic	Meaning		
		Code	**Mnemonic**	**Meaning**
		3	.PFHUU	A page containing a monitor call argument has been swapped out; a monitor detected fault FRAG 11.
		4	.PFHTI	A trap occurs every n units of virtual time as a result of requesting virtual time traps; refer to the description of the .STTVM option of the SETUUO).
		5	.PFHZI	The page has been allocated, but it is a zero page.
		6	.PFHZU	The page has been allocated, but it is zero after a call's execution.

6.2 SEGMENT CONTROL

6.2.1 The RUN Monitor CALL (CALLI 35)

The RUN Monitor call allows program to transfer control to one another. Both the low and the high segments of a user's addressing space are replaced with those of the program being called. RUN's calling sequence is

```
        MOVSI   ac, start-addr-increment
        HRRI ac, addr
        RUN ac,
          error return
                ; normal return is to the start address
                ; plus the start-addr-increment of the
                ; new program
addr:   SIXBIT/device-name/
        SIXBIT/filename/
        SIXBIT/extension/                       ; or 0
        0,,0
        XWD project-number, programmer-number   ; or 0
        XWD 0, core assignemnt                  ; or 0
```

where: *start-addr-increment* is an increment to the starting address.

addr points to the first word of the six-word argument block.

logical-device-name is the name of the device storing the called program.

filename specifies the name of the file for either or both of the high and low segments.

extension is the file name extension for the low segment file. If the left half = 0, .LOW is assumed when there is a high segment; .SAV is assumed when there is not a high segment. When the program has been saved by the NSAVE or NSSAVE command, the extension will be .EXE.

project-programmer number specifies the owner of the called program; if 0, the project-programmer number for the calling program is assumed.

core assignment is an optional argument. If it is present the value specified is assigned to the low segment. The left half of this word is always ignored, and it should be zero.

Normally, the calling program will set up only the first two words of the argument block leaving the rest of the words zero. The error return is taken if any errors are detected; an error code will be returned in the AC. Possible error codes are listed in Appendix E. The user program can attempt to recover from the error and/or give the user more information pertaining to program continuation. If the left half of the error return location contains a HALT instruction, the monitor will return an error code in the AC, and it will print

?HALT AT USER PC *addr*

on the user's terminal, which will be returned to monitor mode. By storing the HALT instruction in the left half of the error return location, a user program can analyze the error code returned in the AC. If the error code indicates an error from which the user can recover, the program can issue a second RUN, including another HALT instruction stored in the error return location. The possible error codes that can be returned in the AC as a result of the RUN monitor call are listed in Appendix E. Note that the monitor will not attempt to return to a user program if the high or low segment containing the RUN call has been overlaid. Therefore, the RUN call should be placed in the low segment in case the error is discovered after the high segment has been released.

For certain system programs (e.g., LOGIN and LOGOUT), the RUN monitor call will automatically set the appropriate bits (e.g., JACCT and JLOG). These bits are not set (or are turned off if they were set) for unprivileged programs from device SYS: or for programs whose starting address offset is greater than I. These bits cannot be set by ASSIGNING a device to be SYS:.

The execution of the RUN monitor call clears all of core. User programs, however, should not assume that this action will occur; they must initialize core to the desired value in order to allow programs to be restarted by the CTRL/C, START sequence without having to perform I/O.

If a user program wants to call programs from the system library, it should call them by device SYS: and the null project-programmer number (instead of the device DSK: and a 1, 4 project-programmer number). The extension specified for these programs should also be null, so that the calling program will not have to determine if the called program is reentrant.

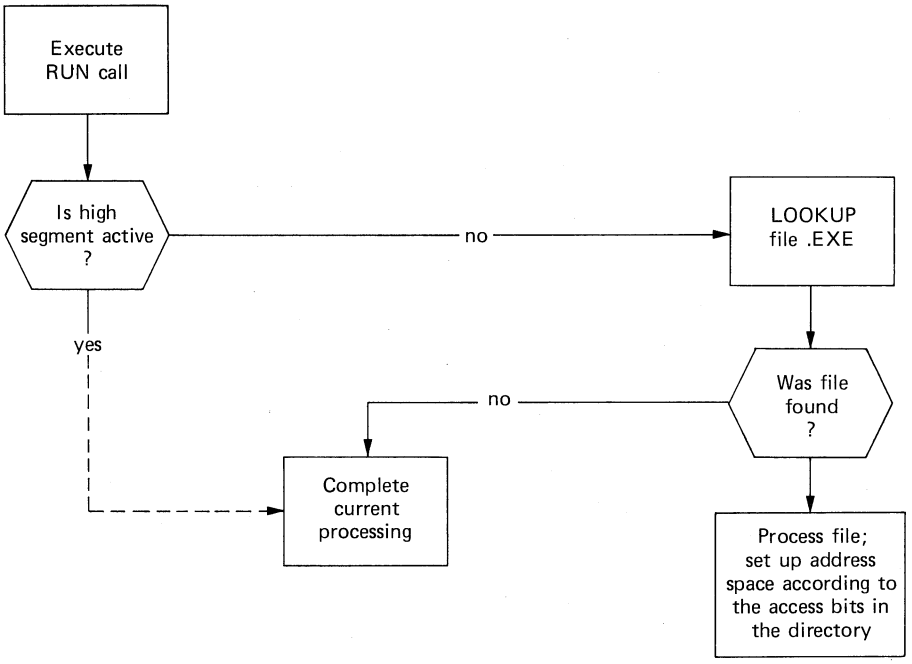
Before control is transferred to the called program, the left half of the AC is added to and stored in .JBSA of the Data Area. A CTRL/C followed by a START command will restart the program at the location specified by the RUN monitor call; therefore the current system program may be restarted.

If the left half of the AC does not contain either 0 or 1, the user is considered to be meddling with the program unless the program being executed is execute-only for this job.

6.2.1.1 Programming with the RUN Monitor Call — To successfully program the RUN monitor call for systems of all sizes and for all system programs whose size is not known at the time of RUN execution, it is necessary to understand the sequence of operations initiated by RUN. (The RUN could be executed from either the high or the low segment; fewer errors are returned to the user, however, if it is executed from the high segment.)

To be guaranteed of handling the largest number of errors, the cautious user should remove his high segment from high logical addressing space (via CORE with I in the left half of the AC). The error handling core should be put in the low segment with the RUN call and the size of the low segment should be reduced to IK. A better idea would be to have to error handling code written once and put into a seldom used (probably non-sharable) high-segment, which could be called via GETSEG when an error return is taken on a RUN monitor call.

The following sequence occurs when a RUN monitor call is utilized for an .EXE file:



Note that if the user specifies an explicit extension to the file name specified in the RUN call, that extension will be used when the file is LOOKedUP. If the extension specified, though, is .SAV, filename.EXE is searched for before filename .SAV.

6.2.2 The GETSEG Monitor Call (CALLI 40)

The GETSEG monitor call replaces the high segment in the user's addressing space. Its calling sequence is

MOVEI *ac, addr*

GETSEG *ac,*

error return

normal return

addr: SIXBIT/*device-name*/

SIXBIT/*filename*/

SIXBIT/*extension*/

0,0

XWD *project-number, programmer-number*

0,0

where: *addr* points to a six-word argument block.

device-name is the name of the device on which the high segment resides.

filename is a 1 to 6 character file name, left-justified, specifying the file containing the new high segment.

project-programmer number are the project-programmer numbers representing the directory in which the file is stored. If this word contains zero, the project-programmer numbers under which the job is logged in are assumed.

The GETSEG monitor call allows a user program to initialize a high segment from a file or shared segment without affecting the low segment. This facility can be used for shared data segments, shared program overlays, and run-time routines (e.g., the FORTRAN or COBOL object-time systems).

The GETSEG monitor call works like the RUN monitor call, except for the following differences:

1. No attempt is made to read the low segment file. If an .EXE file is found, only the pages representing the high segment will be merged into the user's address space.
2. The contents of the ACs are not preserved.
3. The only changes made to JOBDAT are (1) to set the left half of .JBHRL to 0 and (2) to set the right half of the highest legal user address. (Note that a SAVE/OSAVE command can be used to save all of the high segment.)
4. If an error occurs, control will be returned to the error return location, unless its left half contains a HALT instruction.
5. On a normal return, control is returned to one of two locations following the monitor call, depending on whether the high or the low segment made the request. The call should be requested from the low segment unless the normal return coincides with the starting address of the new high segment.
6. User channels 1 through 17 are not released. Channel 0, however, is released because it is used by the GETSEG call.
7. .JBSA and .JBREN will be set to 0 by the monitor, if they point to a high segment that is being removed. If this happens, the following message will be printed on the user's terminal:

   ?NO START ADDRESS

   when a START or REENTER command is issued.

6.2.3 The REMAP Monitor Call (CALLI 37)

The REMAP monitor call remaps the top part of a specified low segment into a high segment. The previous high segment (if any) is removed from the user's addressing space. The new low segment will be the previous low segment minus the amount remapped. The calling sequence for the REMAP monitor call is

MOVEI *ac, addrl*

MOVE *ac,* XWD *origin, addrl*

REMAP *ac,*

error return

normal return

where: *addrl* is the highest address in the low segment.

origin is the origin of the high segment (KI10/KL10 only).

The monitor will round up the address specified to the nearest core allocation unit (i. e., on KA10-based systems the unit is 1024_{10} words; on KI10/KL10-based systems the units is 512_{10} words).

On a normal return, the following will occur:

1. the contents of the AC are preserved.
2. .JBREL is set to the value of *addrl.*
3. The left half of .JBHRL is set to 0.
4. The right half of .JBHRL is set to the highest legal user address in the high segment (greater than or equal to 401777 or 0).
5. The hardware relocation is changed.
6. The user-mode, write-protect bit is set.

The error return is taken under the following conditions:

1. When a high segment origin is specified on a KA10-based system.
2. When the requested remapping would cause the high and the low segments to overlap (KI10/KL10 only).
3. When the sum of the high segment origin plus its length would cause the high segment to start (or end) at an address outside the program's virtual address space (i.e., greater than or equal to 256K).
4. When the specified argument *addrl* exceeds the length of the low segment. (Also, remapping will not occur, and the high segment will remain unchanged in the user's address space).
5. When the system on which the program is running does not have two-register capability.

6.2.4 Testing for a Sharable High Segment

It is occasionally desirable for a program to determine whether or not its high segment is sharable. If the high segment is sharable, the program may decide not to modify itself. The following code determines

- whether or not the system has a high segment capability (i.e., two-register capability), and
- whether or not the job has a high segment.

```
HRROI T, .GTSGN          ; see if high segment is sharable
GETTAB T,                ; look at monitor .GETSGN table
JRST    .+2              ; table or call not present
```

```
TLNN   T, (SN%SHR)        ; is sharable bit on?
JRST   NOTSHR             ; no, go ahead and modify here if
                          ; high segment is sharable
```

6.2.5 Determining the High Segment Origin

It is occasionally desirable for a program to determine the origin of its high segment (i.e., the starting virtual address of the high segment within the program's address space.) This information would be useful, for example, to a program that was to access information in the vestigial Job Data Area or to transfer control to an entry point in a high segment which has been GETSEGed. Before the 5.07/6.01 release of the monitor, the high segment origin was normally 400000_8 or the first available core allocation boundary above the low segment, if the low segment was larger than 129K. User programs should not assume this location for their high segment, but should find its location by using the following procedure:

```
HRRZ    T1, .JBHRL          ; highest relative address in the high segment
JUMPE   T1,NOHIGH           ; jump if there is no high segment
HRRZ    T1,.JBREL           ; the highest address in the low segment
TRNN    T1,400000           ; is low segment larger than 128K?
MOVEI   T1,377777           ; no, assume high segment starts at 400000
MOVE    T2, [XWD -2, .GTUPM] ; get high segment origin from monitor table
GETTAB  T2,                 ; .GTUPM indexed by current high segment number
  HRLI  T2,1(T1)            ; table or call not present, use assumed
LSH     T2, -↑D18           ; value. Convert to address of high segment
ANDI    T2,777000           ; clear any low bits
MOVEM   T2,HIORGN           ; store as the origin of the high segment
```

6.2.6 Modifying Shared Segments and Meddling

A high segment is usually write-protected, but it is possible for a user program to clear the write-protect bit or to increase/decrease a shared segment's core assignment by using the SETUWP monitor call or the CORE monitor call. These calls are legal from either the low segment or the high segment only if the sharable segment has not been meddled with (unless the user has write-privileges for the file that initialized the high segment). Even the malicious user can have the privileges of running such a program, although he does not have the access rights to modify the files used to initialize the shared segment.

Meddling is defined as any of the following conditions (even if the meddling user has the privileges to write the file which initialized the sharable segment):

1. When a START or CSTART command is executed with an argument.
2. When the DEPOSIT command is issued for either the low or the high segment.
3. When the RUN monitor call is called with anything other than a 0 or a 1 in the left half of the AC (i.e., a starting address increment).
4. When the GETSEG monitor call is called.

It is not considered to be meddling to perform any of the above commands or calls with a non-sharable program. It is never considered to be meddling to type a CTRL/C followed by a CONT, CCONT, CSTART (with an argument), START (without an argument), REENTER, DDT, SAVE, NSAVE, OSAVE, or E command (refer to the *DECsystem-10 Operating System Commands Manual).*

When a sharable program is meddled with, the monitor will set the meddle bit for the meddling user. The error return is taken when the user attempts to clear the write-protect bit via the SETUWP monitor call. The error return is taken, also, when the reassignment of core for the high segment (except to remove it completely) is attempted via the CORE monitor call.

An attempt to modify the high segment via the DEPOSIT command causes the following message to be printed on the user's terminal:

?OUT OF BOUNDS

If the user-mode, write-protect bit was not set when the user meddled, it will be set to protect the high segment in case it is being shared. The DEPOSIT command, the SETUWP call, and the CORE call are allowed (in spite of meddling), if the user has the access privileges to write the file which initialized the high segment. Users with access privileges can write programs that access sharable data segments via the GETSEG call and then turn off the write-protect bit using SETUWP.

A privileged user can specify that a sharable program is to be superseded. When a CLOSE, OUTPUT, or RENAME monitor call is executed for a file with the same UFD and file name as the segment being shared, the name of the segment will be set to 0. New users will not be able to share the older version of the program, but they will be able to share the newer version. The monitor, therefore, is required to read a newly created file only once before it initializes the file. The monitor will delete the older version of the file when all users are finished with it.

When control can be transferred only to a small number of entry points (e.g., two), the shared program can do anything that it has the privileges to do. (The program, not the person running it, has these privileges.)

CHAPTER 7
I/O PROGRAMMING

Within this Chapter I/O programming is generally described, specifics deal with the disk. For programming considerations concerning the other devices refer to the applicable chapter. (For example, DECtape I/O programming is described in Chapter 9).

All user I/O programming is controlled by monitor calls. I/O is directed by

1. associating a device and a ring of buffers with one of the user's I/O channels (via INIT, OPEN, or FILOP.),
2. optionally selecting a file (via LOOKUP, ENTER, and FILOP), and
3. passing buffers of data to and from the user program (via IN, INPUT, OUT, or OUTPUT).

Device specifications may be delayed from program generation time until program run-time because the monitor

1. allows a logical device name to be associated with a physical device (via the ASSIGN or MOUNT command), and
2. treats operations that are not pertinent to a given device as no-ops.

7.1 JOB INITIALIZATION

The RESET monitor call should normally be the first instruction in every user program; its calling sequence is

RESET *ac,*
only return

The call immediately stops all I/O on all devices without waiting for the devices to become inactive. All device allocations made by the INIT/OPEN/FILOP. calls are cleared and, unless the devices have been assigned via ASSIGN or MOUNT, the devices are returned to the monitor pool of available devices. The contents of the left half of .JBSA in the Job Data Area is stored in the right half of .JBFF in the Job Data Area, so that the user buffer is reclaimed if the program is restarted. The left half of .JBFF is cleared. Any files being written that have not been closed are deleted on the disk. The user-mode write-project bit is automatically set if a high segment exists (whether or not the high segment is sharable); therefore, a program cannot inadvertently store into the high segment. Additional functions of the RESET call include the following.

1. It unlocks a job if it was locked.
2. It releases any real-time devices.
3. It resets any high-priority queues set by the HPQ monitor call to a value set by the HPQ command (refer to the *DECsystem-10 Operating System Commands* Manual).
4. It resumes timesharing, if timesharing had ceased as a result of a TRPSET monitor call with a non-zero argument.
5. It resets the action of the HIBERnate and APRENB monitor calls.
6. It clears all PC flags except USRMODE (KL10/KI10 only).
7. It drops all PIDs (IPCF) that were to be dropped on the execution of a RESET.
8. It clears the Software Interrupt System.
9. It dequeues anything locked by ENQ.

7.2 DEVICE SELECTION

A specific device must be associated with a software I/O channel for every I/O operation. This specification is made via the INIT, OPEN, or FILOP. monitor calls, which may specify a device by its logical or physical name. Some system programs (e.g., LOGOUT) require I/O to specific physical devices regardless of any logical name assignment.

Therefore, when an OPEN monitor call is executed, if the sign bit of word 0 in the OPEN block is set to 1 (UU.PHS), the device name is regarded as a physical name only, and logical names will not be searched. When an association is made between a device and a software I/O channel via INIT, OPEN, or FILOP., the association remains in effect until the channel is RELEASEd or until another INIT, OPEN, or FILOP., call is issued for that software I/O channel.

Non-disk and non-spooled devices may be assigned to a particular job by use of the ASSIGN or MOUNT commands. Assignable devices are designated as either unrestricted devices or restricted devices. An unrestricted device can be assigned directly by any job via the ASSIGN command, the INIT call, the OPEN call or the FILOP. call. A restricted device can be assigned directly only by a privileged job. However, any unprivileged job can have a restricted device assigned to it indirectly through the use of the MOUNT command.

The MOUNT command allows operator intervention for the selection or denial of a particular device; therefore, the operator can control the use of assignable devices for the non-privileged user. This function is useful when there are multiprogramming batch and interactive jobs competing for the same device. The restricted status of a device is set or removed by the operator with the OPSER commands :RESTRICT and :UNRESTRICT. (Refer to the *DECsystem-10 Operators Guide*.)

The non-directory devices are listed in Table 7-1. The selection of a device is sufficient to allow I/O operations to take place over the associated software channel.

Table 7-1
Non-Directory Devices

Device	3-Letter Generic Name	2-Letter Generic Name
card reader	CDR:	CR:
card punch	CDP:	CP:
line printer	LPT:	LP:
		LL:
		LU:
display unit	DIS:	
paper-tape reader	PTR:	PR:
paper-tape punch	PTP:	PP:
user terminal	TTY:	TT:
pseudo-TTY	PTY:	
magnetic tape unit	MTa:	MT:
		M7:
		M9:
plotter	PLT:	
console TTY	CTY:	

All other file specifiers directed to a non-directory device are ignored. Whether or not the program will use a directory device or a non-directory device for I/O operations, it is advisable that the program select a file, so that a directory device can be substituted for a non-directory device at run-time without notifying the program.

For directory devices (e.g., disk and DECtape), files are addressable by file name. If a directory device has a single file directory (e.g., DECtape, refer to Chapter 9), the device and the file name are sufficient information in determining which file is desired. If the device has a multiple file directory (e.g., disk), the name of the file directory must also be specified in determining the correct file. The file, device, and directory names are specified as arguments to the LOOKUP, ENTER, RENAME, and FILOP. monitor calls.

7.2.1 Device Initialization

The OPEN monitor call (op code 50) and the INIT monitor call (op code 41) initialize a device and associate it with a software I/O channel number. These calls perform almost identical functions; the OPEN call is a reentrant form of INIT and is preferred for this reason. Their calling sequences are

```
        OPEN    channel, addr        INIT    channel, status
          error return               SIXBIT/devicename/
        normal return                XWD obuf, ibuf
        .                              error return
        .                            normal return
        .
addr:   EXP status
        SIXBIT/devicename/
        XWD obuf, ibuf
```

where: *channel* is the 4-bit channel number (0 to 17) to be associated with the device.

addr is the address of a 3-word OPEN argument block.

status contains for OPEN the OPEN status bits in its left half (refer to Table 7-2) and the SETSTS bits in its right half. *status* for INIT contains the SETSTS bits in its right half.

device is the logical or physical name of the device. More information concerning this argument is given in paragraph 7.2.2.

obuf and *ibuf* (used only for buffered modes), if non-zero, specify the location of the first word of the 3-word buffer ring header block for output and input, respectively.

The normal return is taken if a device is selected and if the device is associated with a software I/O channel. The error return is taken if the requested device is in use, if the requested device does not exist, or if the device is restricted and has not been assigned to the job via the MOUNT command. (The MOUNT command is described in *DECsystem-10 Operating System Commands*.) If a device is already associated with the specified channel, the device is released.

The symbols *obuf* and *ibuf* specify the location of the first word of the 3-word buffer ring header blocks for output and input, respectively. Buffered data mode utilizes a ring of buffers in the user area and the priority interrupt system to permit the user to overlap computation with his data transmission. Core memory in the user's area serves as an intermediate buffer between the user's program and the device. The buffer storage mechanism consists of a 3-word buffer ring header block for bookkeeping and a data storage area subdivided into one or more individual buffers linked together to form a ring. During input operations, the monitor fills a buffer, makes that buffer available to the user's program, advances to the next buffer in the ring, and fills that buffer if it is free. The user's program follows the monitor, either emptying the next buffer if it is full or waiting for it to fill.

During output operations, the user's program and the monitor exchange roles; the program fills the buffers and the monitor empties them. Only the headers that will be used need to be specified. For instance, the output header need not be specified, if only input is to be done. Also, data modes 15, 16, and 17 require no buffer ring header block.

If either the buffer headers or the 3-word block lies outside the user's allocated core area, the monitor will stop the job and will print one of the following messages on the user terminal:

?ILLEGAL ADDRESS IN UUO AT USER *addr*
?ADDRESS CHECK FOR DEVICE *device* AT USER *addr*
?ILLEGAL ADDRESS AT *xxxx*

Note that the buffer headers cannot be in the user's AC; however, buffer headers may be in locations above .JBPFI.

The first and the third words of the buffer header are set to zero. The left half of the second word is set up with the byte pointer size field in bits 6 through 11 for the selected device-data mode combination.

If the same device is INITed on two or more channels, the monitor will retain only the buffer headers mentioned in the last INIT, OPEN or FILOP. (A zero specification does not override a previous buffer header specification, though.) Other I/O operations to any of the channels involved act on the buffer mentioned in the INIT/OPEN/FILOP. previous to the I/O operations.

Table 7-2
OPEN Status Bits

Bit	Mnemonic	Meaning
0	UU.PHS	A search is made of physical device names only.
1	UU.DEL	Error logging is disabled. Normally, this bit should not be set by the user; it is used for user-mode diagnostics.
2	UU.DER	Error retry is disabled. Normally, this bit should not be set by the user; it is used for user-mode diagnostics.
3	UU.AIO	Non-blocking I/O will be performed. Refer to Section 7.3.4
4	UU.IBC	The monitor is prevented from zeroing buffers after each output.
5	UU.SIE	Synchronize on an I/O error. The monitor will not perform any more I/O for the user program, until the error bits have been cleared.

7.2.2 Device Names

The device name specified can be either a logical, physical, or generic name. Logical names take precedence over physical device names. The method of device selection depends on the format of the specified SIXBIT device name; it can be in one of the formats listed in Table 7-3.

Table 7-3
Format of Device Names

Format	Example	Meaning
dddn	LPT6	The monitor will attempt to select device LPT6 specifically requested at the user's current node/station.
dddnnu	LPT132	The monitor will attempt to select device LPT2 at node/station number 13.
ddd	LPT	The monitor will attempt to select a device of the desired type (e.g., a line printer) at the user's current node/station. If all devices of this type are in use, the error return will be taken. If no device of the desired type exists at the user's current node/station, the monitor will attempt to select a device of the same type from the central station. If the desired type of device has already been assigned to the job at the appropriate station and it has not been INITed on another channel, it will be selected instead of an unassigned device.
dddnn	LPT01	The monitor will attempt to select any line printer at node/station number 01.

7.2.2.1 File Structure Names – Each file structure has a SIXBIT name associated with it that is specified by the operator at system initialization time. This name can consist of four or less alphanumeric characters, and it must not duplicate any device name, unit name, existing file structure name or ersatz device name (or their abbreviations, refer to section 7.2.2.6). The recommended names for the file structures in the public pool are

DSKA
DSKB
.
.
.
DSKn (in order of decreasing speed)

When a specific file structure is INITed (e.g., DSKA) subsequent LOOKUP and ENTER searches are restricted only to that file structure. Usually a channel is INITed with the generic DSK, in which case all file structures in the active search list of the job are searched (refer to Job Search List, section 8.2.2).

7.2.2.2 Logical Unit Names – When a single file structure name is specified, the set of all units in that file structure is implied; however, it is possible to specify a particular logical unit within a file structure (e.g., DSKA0, DSKA1, and DSKA2 are three logical units in file structure DSKA).

The monitor deals with file structures rather than with individual units; therefore, when reading files, specifying a logical unit within a file structure is equivalent to specifying the file structure itself. The monitor locates the file regardless of which unit it is on within a file structure.

However, when writing a file, the monitor uses the logical unit name as a guide in allocating space and will, if possible, write the file on the unit specified. In this way, a user can apportion files among different units for increased throughput.

7.2.2.3 Physical Controller Class Names – In addition to DSK, single file structure names (e.g., DSKA) and logical unit names (e.g., DSKA0), it is possible to specify a class of controller. If the system has one controller of the type specified, the result is the same as if the user had specified the physical controller name. The controller classes supported by Digital are

FH ;an RC10 controller
FS ;an RH10 controller
DP ;an RP10 controller
RP ;an RH10 controller

7.2.2.4 Physical Controller Names – It is possible to specify any of the units on a particular controller. The monitor relates that name to the file structure which contains at least one unit on the specified controller. More than one file structure may be specified when a physical controller name is used. The controller names that Digital supports are

FHA	FHB	FHC	FHD
DPA	DPB	DPC	DPD
FSA	FSB	FSC	FSD
RPA	RPB	RPC	RPD

7.2.2.5 Physical Unit Names – When a physical controller name is specified, all units on that controller are implied. It is possible to specify a physical unit name on a particular controller. The physical unit names that Digital supports are listed in Table 7-4.

7.2.2.6 Name Abbreviations – Abbreviations may be used as arguments to the ASSIGN command, the INIT monitor call, the OPEN monitor call, and the FILOP. monitor call. The name abbreviation is checked for a match when the

Table 7-4
Physical Disk Unit Names

Name	Meaning
FHA0,...FHA3 FHB0,...,FHB3 FHC0,...,FHC3 FHD0,...,FHD3	A mixture of fixed head disk units (RD10s) and drum units (RM10B) on RC10 controllers.
FSA0,...,FSA3 FSB0,...,FSB3 FSC0,...,FSC3 FSD0,...,FSD3	RS04 disk units on the first, second, third and fourth RH10 controllers.
DPA0,...,DPA3 DPB0,...,DPB3 DPC0,...,DPC3 DPD0,...,DPD3	A mixture of RP02 and RP03 disk packs on the first, second, third, and fourth controllers (RP10).
RPA0,...,RPA3 RPB0,...,RPB3 RPC0,...,RPC3 RPD0,...,RPD3	RP04 disk units on the first, second, third, and fourth RH10 controllers.

ASSIGN, INIT, OPEN, and/or FILOP. is executed. The file structure or device eventually represented by a particular abbreviation depends on whether a LOOKUP, ENTER, or FILOP. follows. A LOOKUP applies to as wide a class of units as possible, and an ENTER applies to as restricted a class of units as possible to allow files to be written on a particular unit at your option.

7.3 DATA MODES

Data transmissions are either unbuffered or buffered. (Note that unbuffered mode is sometimes referred to as dump mode.) The mode of transmission is specified by a 4-bit argument to the INIT, OPEN, FILOP. or SETSTS monitor calls. Buffered mode transmits data to the disk exactly as it appears in the buffer. Data consists of 36-bit words. The data modes are listed in Table 7-5.

All disk buffered mode operations utilize a 200_8 word buffer. Attempts to set up a non-standard buffer size are ignored on the disk. In particular, attempting to use buffer size smaller than 200_8 words for input will result in data being read in past the end of the buffer. This destroys that information that was beyond the buffer (e.g., the buffer header of the next buffer).

With unbuffered mode on the disk, data is read into or written from anywhere in the user's core area without regard to the normal buffering schemes. Control for read or write operations must be via a command list in core memory. The command list format is described in section 7.3.1. The disk control automatically measures dump data into standard-length disk blocks of 200_8 words each. The remainder of a disk block is wasted, unless the number of words is an exact multiple of the standard-disk block length after each command word in the command list.

7.3.1. Unbuffered Data Modes

Data modes 15, 16, and 17 (refer to Table 7-5) are unbuffered data modes that utilize a command list to specify areas in the user's allocated core area to be read or written. The address specified in an IN, INPUT, OUTPUT, or OUT monitor call points to the first word of this command list. There are three types of entries that may be present within the command list:

1. IOWD *n,loc*
   which causes *n* words to be transmitted from *loc* through *loc* + *n* - 1. The next command is obtained from the location following the IOWD. The assembler pseudo-op generates the instruction: XWD -*n*, *loc-1*.

2. XWD *0,y*
   which causes the next command to be taken from location *y*. This instruction is referred to as a GOTO. A maximum of three consecutive GOTOs are allowed in the command list; after which an IOWD that transfers data must be written.
3. 0
   which terminates the command list.

Table 7-5
Data Modes

Code	Mnemonic	Meaning
0	.IOASC	ASCII MODE. Seven-bit bytes are packed left-justified, five characters per word.
1	.IOASL	ASCII-LINE MODE. ASCII-line mode is equivalent to ASCII mode, except that the buffer is terminated by a form-feed, vertical tab, line feed, or ALT MODE character. This difference is pertinent only to terminals (half-duplex software).
2	.IOPIM	Packed image mode.
3-5		Reserved for Digital.
6-7		Reserved for customer definition.
10	.IOIMG	IMAGE MODE. This is a device-dependent mode. The buffer is filled with data exactly as it is supplied by the device, in 36-bit bytes.
11-12		Reserved for Digital.
13	.IOIBN	IMAGE BINARY MODE. This mode is similar to binary mode, except that no automatic formatting or checksumming is performed by the monitor, data is transmitted in 36-bit bytes.
14	.IOBIN	BINARY MODE. This is block format consisting of a word count, *n* (i.e., the right half of the first data word of the buffer), followed by *n* 36-bit bytes.
15	.IOIDP	IMAGE DUMP MODE. A device-dependent mode where data is transmitted in 36-bit bytes.
16	.IODPR	DUMP MODE. Dumps data as records without core buffering. Data is transmitted between any contiguous blocks of core, and one or more standard-length records on the device for each command word in the command list. Data is transmitted in 36-bit bytes.
17	.IODMP	DUMP MODE. Dumps one record of data without core buffering. Data is transmitted between any contiguous block of core, and one record of arbitrary length on the device for each command word in the command list. Data is transmitted in 36-bit bytes.

On disk, every IOWD that causes data to be transferred writes a separate record. Therefore, for all devices (except DECtape) the following two examples will produce the same result:

1.
```
OUTPUT D,[IOWD 70,  BUF1
          IOWD 70,  BUF2
          Z]
```

2.
```
OUTPUT D,[IOWD 70, BUF1
          Z]
OUTPUT D,[IOWD 70, BUF2
          Z]
```

For DECtape, the first example writes one block, and the second examples writes two blocks.

When the command list has been completely processed, the monitor returns control to the user. If an illegal address is encountered while processing the command list, the monitor will stop the job and print the following message on the user's terminal:

?ADDRESS CHECK AT USER *addr*

DUMP input is similar to dump output; below is an example of dump output of fixed-length records.

```
;SUBROUTINE TO OPEN DISK IN DUMP MODE
;CALL WITH:
;          PUSHJ    P,DMPINI
;          RETURN   HERE
;
DMPINI:    OPEN     DSK,OPNBLK          ;OPEN THE DISK ON CHANNEL "DSK"
             JRST   NOTAVL              ;DEVICE IS BUSY
           ENTER    DSK, FILE           ;CREATE A NEW FILE
             JRST   FILBAD              ;CANNOT ENTER FILE NAME IN
                                        ;DISK DIRECTORY.
           POPJ     P,0                 ;RETURN–FILE IS NOW OPEN FOR
                                        ; DUMP MODE OUTPUT

;HERE IF DEVICE DSK:  CANNOT BE OPENED
NOTAVL:    OUTSTR   [ASCIZ "?CANNOT OPEN DSK:
"]                                      ;PRINT AN ERROR MESSAGE
           EXIT                ;RETURN TO THE MONITOR

;HERE IF CANNOT ENTER DUMP.BIN
FILBAD:    OUTSTR   [ASCIZ "?CANNOT CREATE DSK:DUMP.BIN
"]                                      ;PRINT ERROR MESSAGE
           EXIT                         ;RETURN TO THE MONITOR

OPNBLK:    EXP      .IODMP              ;SELECT DUMP MODE
           SIXBIT   /DSK/               ;DEVICE NAME
           EXP      0                   ;NO BUFFERS

FILE:      SIXBIT   /DUMP/              ;FILE NAME
           SIXBIT   /BIN/               ;EXTENSION
           EXP      0                   ;DEFAULT PROTECTION
           EXP      0                   ;DEFAULT DIRECTORY

;SUBROUTINE TO WRITE DATA IN BUFFER
;CALL WITH:
;          PUSHJ    P,DMPOUT
;          RETURN HERE
;
DMPOUT:    OUT      DSK,OUTLST          ;WRITE THE DATA
             POPJ   P,0                 ;NO ERRORS – RETURN TO CALLER
           OUTSTR   [ASCIZ "?OUTPUT ERROR FOR DSK:DUMP.BIN
"]                                      ;PRINT AN ERROR MESSAGE
           EXIT                         ;QUIT

;I/O LIST FOR THE OUTPUT
OUTLST:    IOWD     BUFSIZ,BUFFER       ;WRITE BUFSIZ WORDS FROM BUFFER
           EXP      0                   ;END OF I/O LIST
BUFFER:    BLOCK    BUFSIZ              ;OUTPUT BUFFER
```

```
;SUBROUTINE TO CLOSE OUT FILE
;CALL WITH:
;          PUSHJ    P,DMPDON
;          RETURN   HWERE
;
DMPDON:    CLOSE    DSK,                  ;WRITE THE END OF FILE
           STATO    DSK,IO.ERR            ;ANY ERRORS?
           POPJ     P,0                   ;NO–RETURN
           OUTSTR   ;ASCIZ "?ERROR CLOSING DSK:DUMP.BIN
"]                                        ;PRINT ERROR MESSAGE
           EXIT                          ;RETURN TO THE MONITOR
```

7.3.2 Buffered Data Modes

Data modes 0, 1, 2, 10, 13, and 14 are buffered data modes (refer to Table 7-5). The address specified in an INPUT, IN, OUTPUT, or OUT monitor call may be used to alter the normal sequence of buffer reference. If the specified address is 0, the address of the next buffer is obtained from the right half of the second word of the current buffer. If the specified address is non-zero, it is the address of the second word of the next buffer to be referenced.

The buffer pointed to by the specified address can be in a separate ring from the present buffer. Once a new buffer location has been established, the following buffers are taken from the ring specified by address. Since buffer rings are not changed if I/O activity is pending, it is not necessary to issue a WAIT monitor call.

7.3.2.1 Buffered Input – If no input buffer ring was established (devices other than disk) when the first INPUT or IN monitor call was executed, on buffered input a 2-ring buffer ring is set up. If no buffer ring was established for a disk device when the first INPUT or IN monitor call was executed, the default number of buffers will be set up (this default value is specified at monitor generation time via MONGEN). Buffered input may be performed either synchronously or asynchronously. If bit 30 of the status word is set, each INPUT and IN monitor call will perform the following:

1. The use bit in the second word of the buffer is cleared, thereby making the buffer available for refilling by the monitor.
2. An advance is made to the next buffer (by moving the contents of the second word of the current buffer to the right half of the first word of the 3-word buffer header).
3. Control is returned to the user's program if an end-of-file or error condition is encountered. Otherwise, the monitor will start the device, which fills the buffer and stops data transmission.
4. The number of bytes in the buffer is computed from the number of words in the buffer (right half of the first data word in the buffer) and the byte size, and the result is stored in the third word of the buffer header.
5. The position and address fields of the byte pointer are set in the second word of the buffer header, so that the first data byte is obtained by an ILDB instruction.
6. Control is returned to the user's program.

In synchronous buffered mode, the position of a device, relative to the current data, is easily determined. The asynchronous input mode differs in that once a device has been started, successive buffers in the ring may be filled at the interrupt level without stopping transmission until a buffer whose bit is 1 is encounted. Control is returned to the user after the first buffer is filled. The position of the device, relative to the data currently being processed by the user's program, depends on the number of buffers in the ring and when the device was last stopped.

7.3.2.2 Buffered Output – If the first word of the buffer header is 0, (i.e., no output buffer ring has been established, the following will occur when the first OUT or OUTPUT monitor call is executed the default number of buffers will be set up. The default is established at monitor generation time via *MONGEN*.

NOTE

For non-disk devices, a 2-buffer ring will be set up.

If the ring use bit (bit 0 of the first word of the buffer header) is 1, it will be set to 0. The current buffer will be cleared to zeroes, and the position and address field of the buffer byte pointer (i.e., the second word of the buffer header) will be set, so that the first byte will be properly stored in an IDPB instruction. The byte count (i.e., the third word in the buffer header) will be set to the maximum number of bytes that can be stored in the buffer, and control will be returned to the user's program.

The first execution of an OUT or OUTPUT monitor call initializes the buffer header and the first buffer, but it does not transmit data. Note that a dummy OUT or OUTPUT is required in buffered mode I/O. If the ring use bit contains 0 and bit 31 of the file status word is 0, the number of words to be in the buffer is computed from the address field of the buffer byte pointer (i.e., the second word of the buffer header) and the buffer pointer (i.e., the first word of the buffer header). The result will be stored in the right half of the third word of the buffer.

If bit 31 of the file status word contains 1, the monitor assumes that the user has already set the word count in the right half of the third word. The buffer use bit (bit 0 of the second word of the buffer) is set to 1, which indicates that the buffer contains data to be transmitted to a device.

If the device is not currently active (i.e., not receiving data), the device is started. The buffer header is then advanced to the next buffer by setting the buffer pointer in the first word of the buffer header. If the buffer use bit of the new buffer contains 1, the monitor places the job into a wait state until the buffer is emptied at the interrupt level. The buffer is then cleared to zeroes (provided that the OPEN bit was not specified), the buffer byte pointer and the byte count are initialized in the buffer header, and control is returned to the user's program.

7.3.2.3 Synchronization of Buffered I/O – Sometimes it is desirable to delay a device until it completes its I/O activities. An example of this would be when trying to recover from a transmission error. The WAIT monitor call returns control to a user's program when all data transfers on a specific channel have been satisfied. The WAIT calling sequence is

```
WAIT   channel,
       only return
```

If no device has been associated with the specified channel number control will be returned immediately. After the device has been stopped, the position of the device, relative to the data currently being processed by the user's program, can be determined by the buffer use bit.

7.3.3 Buffer Structure

Buffers are used when transmitting data to account for the different speeds between a sending device and a receiving device. Before performing any I/O, the device to be used must be associated with one of the 16 software I/O channels. All software I/O channels are bi-directional (i.e., they may be used to perform both input and output).

The size of the buffer is different depending on the type of device used. A buffer ring header must be declared in a user program; it is a three words in length and is described in further detail in section 7.3.3.1. Buffers are created at the location specified by .JBFF in the Job Data Area; when buffers are filled, .JBFF will contain a value equal to the number of words in the buffer plus one.

7.3.3.1 Buffer Ring Header Block – The location of the buffer ring header block is specified by an argument to either of the INIT, OPEN, or FILOP. monitor call. The information stored in the buffer ring header block is placed there by the monitor in response to user execution of certain monitor calls. The user program has access to the information within the buffer ring header block, which is necessary information that is required to fill and empty buffers.

The buffer ring header block is represented in Figure 7-1.

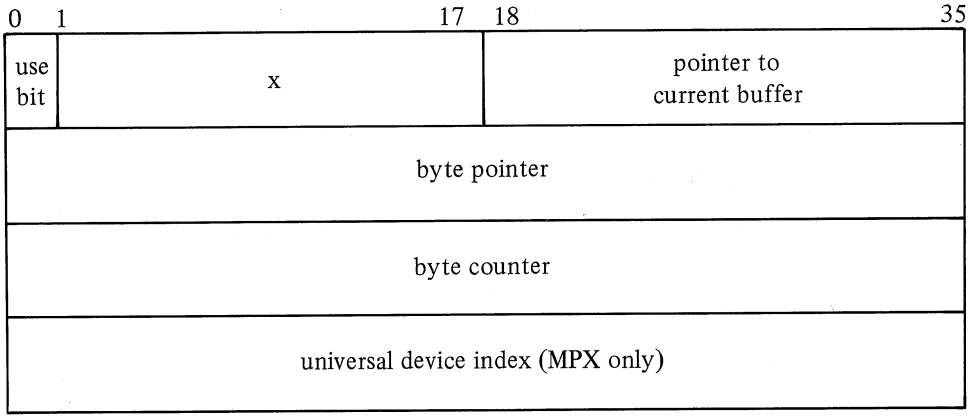


Figure 7-1. Buffer Ring Header Block

where: *the use bit* (bit 0, BF.VBR) indicates whether or not there has been input into or output from the buffer pointed to by pointer to current buffer. (If bit 0 equals 1, the buffer is available to the filler).

x if set (bit 1 = 1, BF.IBC) inhibits the buffer from being cleared.

pointer to current buffer (.BFADR) points to the second word of the current buffer.

byte pointer (.BFPTR) points to the byte within the current buffer which is the next input/output data. (The byte size is determined by the data mode.)

byte counter (.BFCTR) is the count of the number of bytes remaining in the current buffer.

A user program cannot use a single buffer ring header block for both input and output. A single buffer ring header cannot be used for more than one I/O function at the same time. Users cannot use the same buffer ring header block for simultaneous input and output; only one buffer ring is associated with each buffer ring header block. Note that the first word of the buffer ring header block (.BFADR) is zeroed on an OPEN/INIT/FILOP. monitor call.

7.3.3.2 Buffer Ring – The buffer ring is established by the INBUF and OUTBUF monitor calls, or if neither of these exists, when the first IN, INPUT, OUT, or OUTPUT call is executed a 2-buffer ring will be set up. The address specified in a INBUF or OUTBUF call specifies the number of buffers in the ring. The location of the buffer ring is specified by the contents of the right half of .JBFF in the Job Data Area. The monitor then updates .JBFF to point to the first location past the storage area of the buffers.

All buffers in a ring are identical in structure, refer to Figure 7-2.

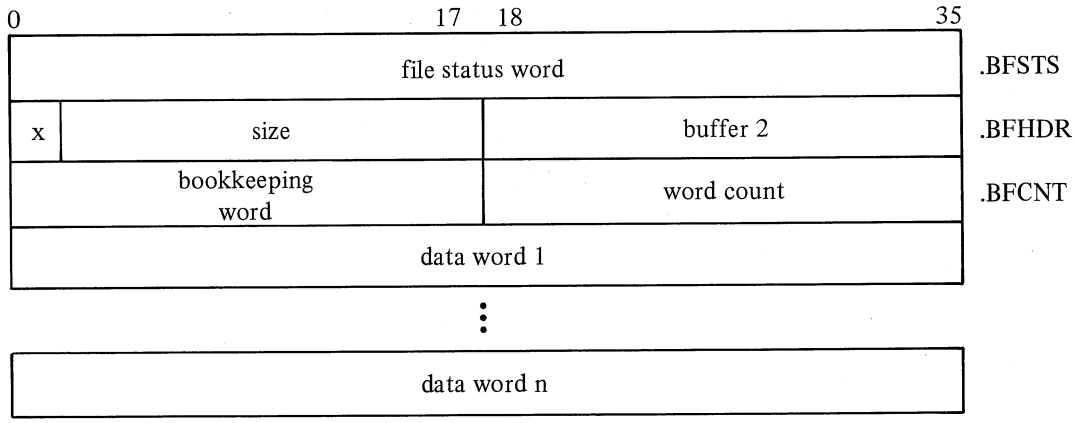


Figure 7-2. Buffer Structure

where: *file status word* (.BFSTS) contains the status of the file when the monitor advances to the next buffer in the ring. This word contains the errors received while working with buffered mode, they are not recorded in the GETSTS word.

x (BF.IOU) is the use bit for the buffer which is a flag that indicates whether or not the buffer contains active data. This bit is set to a 1 by the monitor when the buffer is full on output or it is being filled on input (i.e., if the use bit = 0, the buffer is available to the filler; if the use bit = 1, the buffer is available to the emptier.) The use bit prevents the monitor and the user's program from interfering with each other by attempting to use some of the same buffer simultaneously. Buffers are advanced by the monitor calls and not by the user's program. The use bit should never be changed by a user program.

size (BF.SIZ) is the size of the data area of the buffer plus one. The size of this data area is dependent on the type of device being used.

word count (BF.CNT) is reserved for a count of the number of words that actually contain data.

bookkeeping word is reserved for bookkeeping purposes, depending on the particular device and the data mode. This word contains the Universal Device Index (UDX) for MPX devices.

A user program can check the contents of the use bit in the current buffer by either of the following procedures:

```
MOVE   ac, @RING        SKIPL   @ RING
TLNE   ac, 400000
```

if the user program labeled the buffer ring header block RING.

which skips if the use bit contains 1.

7.3.3.3 Monitor Generated Buffers – Each device has a specified buffer size associated with it. The INBUF and OUTBUF calls set up a ring of n standard-size buffers associated with the input and output buffer headers, respectively, which are specified by the last OPEN/INIT call of the specified data channel. If the number of buffers is zero for either an INBUF or an OUTBUF call, the default number of buffers for the specified device will be set up. If the specified device has not been initialized by either an OPEN, INIT or FILOP. monitor call, the monitor will stop the job and print the following message on the user's terminal;

?I/O TO UNASSIGNED CHANNEL AT USER *addr*

The storage space for a buffer ring is taken from successive locations, beginning with the location specified in the right half of .JBFF of the Job Data Area. This location is set to the program break, which is the first free location above the program area, by a RESET monitor call. If there is insufficient space to set up the buffer ring the monitor will automatically attempt to expand the user's core allocation. If this core reallocation fails, the monitor will stop the job and print the following message on the user's terminal:

?ADDRESS CHECK FOR DEVICE *device* AT USER *addr*

The above message is also printed when an INBUF (or OUTBUF) is attempted when the last INIT (OPEN or FILOP.) call on the specified channel did not specify an input (or output) buffer ring header block.

7.3.3.4 User Generated Buffers – The following code illustrates an alternative to the use of the INBUF operator. Analogous code may replace OUTBUF. This code operates similarly to INBUF, size must be set equal to the greatest number of data words expected in one physical record.

```
GO:     OPEN    I,OPNBLK        ;INITIALIZE ASCII MODE
          JRST  NOTAVL
        MOVE    0, [XWD 400000,BUF1+1]
```

```
                                ;THE 400000 IN THE LEFT HALF
                                ;MEANS THE BUFFER WAS
                                ;NEVER REFERENCED>
        MOVEM   0,MAGBUF        ;SET UP NONSTANDARD
        MOVE    0, [POINT BYTSIZ,0,35]    ;BYTE SIZE
        MOVEM   0,MAGBUF+1
        JRST    CONTIN          ;GO BACK TO MAIN SEQUENCE

OPNBLK: 0
        SIXBIT/MAT0/            ;MAGNETIC TAPE UNIT 0
        XWD 0,MAGBUF            ;INPUT ONLY
MAGBUF: BLOCK 3                 ;SPACE FOR BUFFER RING
                                ;HEADER
BUF1:   0                       ;BUFFER 1, FIRST WORD UNUSED
        XWD     SIZE+1,BUF2+1   ;LEFT HALF CONTAINS
                                ;DATA AREA
                                ;SIZE+1, RIGHT HALF HAS
                                ;ADDRESS OF NEXT BUFFER
        BLOCK SIZE + 1          ;LEFT HALF CONTAINS DATA
                                ;AREA.
```

7.4 DEVICE TERMINATION AND REASSIGNMENT

7.4.1 RELEASE A Device (Op Code 071)

When all transmission between the user's program and a device is finished, the program may relinquish the device by performing a RELEASE monitor call. Its calling sequence is

RELEASE *channel,*

If no device is associated with channel, RELEASE (op code 071) returns control immediately. Otherwise both input and output sides of channel are CLOSEd, all locks created by ENQ/ENQ are released, and the correspondence between channel and the device, which was established by an INIT/OPEN/FILOP., is terminated.

If the device is neither associated with another data channel nor assigned by the ASSIGN or MOUNT commands, it is returned to the monitor's pool of available facilities. Control is returned to the user's program.

7.4.2 The RESDV. Monitor Call (CALLI 117)

The RESDV. monitor call allows a program to reset a single channel. It is similar to the RELEASE call, except that no files or buffers are closed. Its calling sequence is

MOVEI *ac, channel*
RESDV. *ac,*
error return
normal return

On a normal return, the channel is reset. Files that are being created on the channel are deleted; any older generation with the same file name remains. All I/O transmissions on the channel are stopped; and device allocations made by INIT or OPEN or FILOP. on the specified channel are closed. The device is returned to the monitor's pool of available devices unless it has been ASSIGNed by either the ASSIGN, ALLOCATE, or MOUNT commands (refer to *DECsystem-10 Operating System Commands*).

On an error return, the AC will contain -1 if no device was associated with the specified channel. If the call has not been implemented, the AC will be unchanged.

7.4.3 The REASSIGN Monitor Call (CALLI 21)

The REASSIGN monitor call will reassign a device under program control to a specified job. The device's logical name assignment is not cleared, if the program doing the REASSIGN has JACCT or [1,2] privileges. REASSIGN's calling sequence is

```
 MOVEI    ac, job-number
{MOVE     ac+1, [SIXBIT/devicename/]}
{MOVEI    ac+1, channel              }
{MOVEI    ac+1, udx                  }
 REASSIGN ac,
 only return
```

where: *job number* is the number of the job for which a device is to be assigned.
device is the physical or logical name of the device.
channel is the number of the channel on which the reassigned device has been initialized.
udx is the universal device index for the device.

A device can be reassigned if it is assigned to the calling job, or if it is not assigned to any job and it is not detected. A RELEASE is performed on the device unless

1. the job issuing the call is reassigning the device to itself (-1 in the AC), or
2. the job is reassigning the device by specifying 0 in the AC.

If the device is a restricted device and 0 is in the AC when the REASSIGN call is made, the device is returned to the restricted pool of devices and it can then be reassigned to a non-privileged job by a privileged job.

On a return, if the contents of the AC is 0, it indicates that the specified job does not exist. If the contents of the AC is -1 it indicates that

1. the device has not been assigned to the new job.
2. the device is the job's controlling terminal.
3. the logical name is a duplicate logical name, or
4. the device specified is connected to an MPX channel, and no UDX was specified.

7.4.4 The DEVLNM Monitor Call (CALLI 107)

The DEVLNM monitor call sets the logical name for a specified device. Its calling sequence is

```
{MOVEI    ac, channel                }
{MOVEI    ac, udx                    }
{MOVE     ac, [SIXBIT/devicename/]   }
 MOVE     ac+1, [SIXBIT/logicalname/]
 DEVLNM   ac,
    error return
 normal return
```

where: *devicename* is the name of the device for which a logical name is desired.
channel is the channel number on which the specified device has been initialized.
udx is the universal device index for the specified device.
logical name is the logical name to be assigned to the device.

On a normal return, the AC and AC+1 are unchanged. On an error return, the AC will contain one of the error codes listed in Table 7-6.

Table 7-6
DEVLNM Error Codes

Code	Mnemonic	Meaning
-1	DVLNBX%	A non-existent device or channel number has been specified.
-2	DVLIU%	The logical name specified is already in use.
-3	DVLBA%	The device specified has not been ASSIGNed or INITed.

7.5 DEVICE INFORMATION

7.5.1 The DEVSTS Monitor Call (CALLI 54)

The DEVSTS monitor call retrieves the device status word of the device data block (DDB) for a specified device. Its calling sequence is

```
{MOVEI   ac, channel          }
{MOVE    ac, [SIXBIT/device/] }
 DEVSTS  ac,
    error return
  normal return
```

where: *channel* is the number of the software I/O channel associated with the device.
device can be one of the following:

CDR
CDP
DTA
PTR
PTP
DSK
LPT
PLT

The device status word is used by a device service routine to save the results of a CONI after each interrupt from the device.

The DEVSTS monitor call is not meaningful and returns a word of zeroes under the following conditions:

1. When an MPX channel/device has been specified.
2. When a device controlled by a front-end has been specified.
3. When a magnetic tape or disk device has been specified.
4. When used in asynchronous buffered I/O mode. Unless a WAIT monitor call has first been issued to ensure synchronization of the actual data transferred with the device status returned.

On a normal return, the contents of the device status word is returned in the AC. If the device service routine does not store a CONI, useless information may be returned.

Note that the error return is not taken when the device service routine does not use the device status word for its intended purpose. Devices with both a control and data interrupt will store the controller CONI (DTA, DSK, DSK2, DPC, DPC2).

7.5.2 The DEVCHR Monitor Call (CALLI 4)

The DEVCHR Monitor Call will return the physical characteristics associated with a specified device. Its calling sequence is

```
{MOVE    ac, [SIXBIT/device/]}
{MOVEI   ac, channel         }
{MOVEI   ac, udx             }
 DEVCHR  ac,
 normal return,
```

where: *device* is the physical/logical name of the desired device.
channel is the right-justified software I/O channel associated with the desired device.
udx is the universal device index associated with an MPX device.

If the specified device is not found or the specified channel has not been initialized, the AC will contain a zero on the return from the call. If the device specified is found, the AC will contain the device's characteristics (refer to Table 7-7).

7.5.3 The DEVTYP Monitor Call (CALLI 53)

The DEVTYP monitor call returns the physical properties associated with a specified device. Its calling sequence is

```
{MOVE    ac, [SIXBIT/device/]}
{MOVEI   ac, channel         }
{MOVEI   ac, udx             }
 DEVTYP  ac,
    error return
 normal return
```

On a normal return, the device type bits will be returned in the AC, refer to Table 7-8. If the AC contains 0 on a normal return, it is an indication that either the c device does not exist or the channel has not been initialized.

On an error return, the AC remains unchanged, indicating that the DEVTYP monitor call has not been implemented, in which case, the DEVCHR monitor call may be used.

7.5.4 The DEVSIZ Monitor Call (CALLI 101)

The DEVSIZ monitor call is used to determine the buffer size for a device, if the user wants to allocate core. Its calling sequence is

```
          MOVEI   ac, addr
          DEVSIZ  ac,
             error return
          normal return

addr:     EXP  data mode
addr+1:  {SIXBIT/device/}
         {channel       }
         {udx           }
```

where: *addr* points to a 2-word argument block.

data mode contains the bit settings the user set on an OPEN or INIT.

device is the physical/logical name of the device.

channel is the number of the software I/O channel associated with the device.

udx is the universal device index.

If the device exists and the data mode is legal, the AC will contain in bits 0-17, the default number of buffers, and in bits 18-35 the default buffer size (including the first three words of the buffer).

Table 7-7
Device Characteristics

Bit	Mnemonic	Meaning
0	DV.DRI	The DECtape directory is in core. This bit is cleared by an ASSIGN call or a DEASSIGN call to the specified DECtape unit.
1	DV.DSK	The device is a disk unit.
2	DV.CDR	The device is either a card reader (DVIN = 1) or a card punch (DVOUT = 1).
3	DV.LPT	The device is a line printer.
4	DV.TTA	The device is a terminal that is controlling a job.
5	DV.TTU	TTY DDB is in use.
6		Reserved.
7	DV.DIS	The device is a display unit.
8	DV.LNG	The device has a long dispatch table (monitor calls other than INPUT, OUTPUT, CLOSE, and RELEASE perform real functions).
9	DV.PTP	The device is a paper tape punch.
10	DV.PTR	The device is a paper tape reader.
11	DV.DTA	The device is a DECtape unit.
12	DV.AVL	The device is available to this job, or this device is already assigned to this job.
13	DV.MTA	The device is a magnetic tape unit.
14	DV.TTY	The device is a terminal.
15	DV.DIR	The device is a directory device (disk or DECtape).
16	DV.IN	The device can perform input.
17	DV.OUT	The device can perform output.
18	DV.ASC	The device has been ASSIGNED.
19	DV.ASP	The device has been assigned via INIT/OPEN.
20	DV.M17	Mode 17 - dump - .IODMP
21	DV.M16	Mode 16 - dump records - .IODPR
22	DV.M15	Mode 15 - Image dump - .IOIDP
23	DV.M14	Mode 14 - Binary - .IOBIN
24	DV.M13	Mode 13 - Image Binary - .IOBIN
25	DV.M12	Mode 12
26	DV.M11	Mode 11
27	DV.M10	Mode 10 - image - .IOIMG
28	DV.M7	Mode 7
29	DV.M6	Mode 6
30	DV.M5	Mode 5
31	DV.M4	Mode 4
32	DV.M3	Mode 3
33	DV.M2	Mode 2 - packed image - .IPOIM
34	DV.M1	Mode 1 - ASCII line - .IOASL
35	DV.M0	Mode 0 - ASCII - .IOASC

Table 7-8
Device Type Bits

Bit	Mnemonic	Meaning
0 = 1	TY.MAN	LOOKUP/ENTER mandatory.
1 - 11		Reserved for the future.
12 = 1	TY.AVL	Device is available to this job.
13 = 1	TY.SPL	Spooled on disk. (Other bits reflect properties of real device, except variable buffer size.)
14 = 1	TY.INT	Interactive device (output after each break character).
15 = 1	TY.VAR	Capable of variable buffer size (user can get his own buffer lengths).
16 = 1	TY.IN	Capable of input.
17 = 1	TY.OUT	Capable of output.
18 - 26	TY.JOB	Job number that currently has device INITed or ASSIGNed.
27 - 28		Reserved for the future.
29	TY.RAS	Device is a restricted device (i.e., can be assigned only by a privileged job or the MOUNT command).
30 - 35	TY.DEV	Device type code.

Code	Mnemonic	Meaning
0	.TYDSK	Disk of some sort
1	.TYDTA	DECtape
2	.TYMTA	Magnetic tape
3	.TYTTY	TTY or equivalent
4	.TYPTR	Paper-tape reader
5	.TYPTP	Paper-tape punch
6	.TYDIS	Display
7	.TYLPT	Line printer
10	.TYCDR	Card reader
11	.TYCDP	Card punch
12	.TYPTY	Pseudo-TTY
13	.TYPLT	Plotter
14	.TYEXT	External task
15	.TYMPX	Software MPX
16	.TYPAR	PA611R on DC44
17	.TYYCR	PC11(R) on DC44
20	.TYPAP	PA611P on DC44
21	.TYLPC	LPC-11 on DC44
22	.TYPCP	PC-11 (P) on DC44
23-57		Reserved for Digital
60-77		Reserved for customer

The error return is taken if the call is not implemented, in which case, the AC is unchanged. On an error return, one of the error codes listed in Table 7-9 will be returned in the AC.

Table 7-9
DEVSIZ Error Codes

Error Code	Mnemonic	Meaning
0	DVSDM%	Device exists, but the data mode is dump mode.
-1	DVSNX%	Nonexistent device.
-2	DVSIM%	Illegal mode.

7.5.5 The WHERE Monitor Call (CALLI 63)

The WHERE monitor call returns the physical node/station number of the specified device. Its calling sequence is

```
{MOVE    ac, [SIXBIT/device/]}
{MOVEI   ac, channel         }
 WHERE   ac,
    error return
 normal return
```

where: *device* is a SIXBIT device name.
channel is a right-justified software I/O channel number associated with the device.

If OPR is specified as the device name, the node/station number at which the job is logically located is returned. If CTY is specified, the node/station number of the job's command decoder is returned. If TTY is specified, the node/station number at which the job's terminal is physically located is returned.

On a normal return, the left half of AC contains the status of the node, and the right half of AC contains the node number associated with the device. The status of the node is represented by the following bits:

BIT 13 = 1 if the node is dial-up (.RMSDU).
BIT 14 = 1 if the node is loaded (.RMSUL).
BIT 15 = 1 if the node is in the loading procedure (.RMSUG).
BIT 16 = 1 if the node is down (.RMSUD).
BIT 17 = 1 if the station is not in contact (.RMSUN).

The error return is taken if the monitor call is not implemented, the channel specified is not INITed, or the requested device does not exist.

7.5.6 The NODE Monitor Call (CALLI 157)

The NODE monitor call performs different functions depending on the function code specified in the left-half of the AC. The functions are described in Table 7-10.

Table 7-10
NODE Monitor Call Function Codes

Function Code	Meaning
1	Assigns the logical device name specified in the argument list to the physical device specified.
2	Returns the node number in the specified AC, given the node name. Returns the node name in the specified AC, given the node number.

Table 7-10 (Cont.)
NODE Monitor Call Function Codes

Function Code	Meaning
3	Performs special station control message transfers (privileged monitor call).
4	Requests bootstrap messages from the remote node (privileged monitor call).

The calling sequence for the NODE monitor call is

```
        MOVE  ac, [XWD func-code,addr]
        NODE  ac,
           error return
        normal return

addr:   argument-list
```

Function code 1 assigns a logical device name to a physical device. Its argument list is:

```
XWD 0, count
SIXBIT/node-id/
SIXBIT/dev-name/
SIXBIT/log-name/
```

where: *count* is the number of arguments.
node-id is the identification of the node on which the physical device exists.
dev-name is the physical device name.
log-name is a logical device name assigned by the user.

If *log-name* is omitted from the argument-list, the logical name is the same as the physical device name specified.

Function code 2 returns either the node name or node number; its argument list is

```
XWD 0,2
SIXBIT/node-name/
XWD 0, node-num
```

where: *node-name* is the name of the node, and
node-num is a 2-digit node number.

For function code = 3, the job must be privileged and locked in core. This function performs special station control message transfers. The argument list is

```
XWD 0,count
SIXBIT/node-name/
XWD 0,node-num
XWD  #bytes,addr
XWD  #bytes,addr
```

where: *count* is the number of arguments.
 # bytes is the number of 8-bit bytes in the transmit message
addr is the address of the block containing the message.
 # bytes is the number of bytes that can be received.
addr is the address of the block into which the message is to be placed.

For function code = 4, the job must be privileged and locked in core. This function requests bootstrap messages from the remote node. The argument list is

```
XWD 0,count
0                    ; location function uses to return node number
0                    ; unused
XWD #bytes,addr
```

where: *count* is the number of arguments.
#bytes is the number of bytes to be received.
addr is the address of the block into which the message is to be placed.

If the monitor call is successful, the normal return is taken and the AC remains unchanged. If the block for the device does not yet exist, a device block is built and communication is established with the remote device.

If the monitor cannot perform the logical assignment, the error return is taken and the AC contains an error code. Refer to Table 7-11 for a list of error codes and their meanings.

The following example, upon normal return, causes all references to logical device LPTA to be associated with physical device LPT on node NODEA.

```
        MOVE AC,[XWD 1,ADDR]
        NODE AC,
           JRST NOGOOD
        JRST GOOD
        .
        .
        .
ADDR:   XWD 0,4              ; four arguments in list
        SIXBIT/NODEA/        ; 6-bit node-ID
        SIXBIT/LPT/          ; 6-bit device name
        SIXBIT/LPTA/         ; 6-bit logical name
```

Table 7-11
NODE Monitor Call Error Codes

Error Code	Meaning
1	Illegal argument list. Either an address error occurred or the value is not within an acceptable range.
2	Illegal node number or illegal node name.
3	The user is not a privileged job.
4	Remote node control is not available.
5	The job is not locked in core.

7.5.7 The DEVNAM Monitor Call (CALLI 64)

The DEVNAM monitor call returns the SIXBIT physical name of a device obtained through either a generic INIT/OPEN or a logical device assignment. Its calling sequence is

```
{ MOVE    ac, [SIXBIT/device/] }
{ MOVEI   ac, channel          }
{ MOVEI   ac, udx              }
  DEVNAM  ac,
  error return
normal return
```

where: *device* is the logical/physical name of the device for which the physical device name is to be returned.
channel is the channel number on which the device has been INITed.
udx is the physical device index for a device.

The normal return is taken if the specified device is found, and AC will contain the SIXBIT physical name. The error return is taken if the call has not been implemented, in which case the AC is unchanged. The error return is also taken if the specified channel has not been INITed or the specified device does not exist.

CHAPTER 8
FILES

8.1 FILE DEFINITION

A file is an ordered set of data stored on a peripheral device. The extent of a file on input is determined by an end-of-file condition, which is different depending on the device used. The extent of a file on output is determined by the amount of information written by an OUT or OUTPUT monitor call, up through and including the next CLOSE and RELEASE monitor call.

8.2 STRUCTURE OF DISK FILES

The file structure of a disk system minimizes the number of disk seeks needed for sequential and/or random access to a file during I/O. The monitor automatically assigns physical space for file data, when a user program writes or deletes a logical file.

Disk files may be any length, and each user can have as many disk files as disk space for that user will allow. Files can be simultaneously read and updated, allowing shared data bases (refer to Enqueue/Dequeue, Chapter 16). A new generation of a file may be recreated by one user while other users continue to read the old generation, allowing a smooth replacement of shared data files and programs. All users may selectively update portions of the file, rather than creating new generations.

NOTE

The file structure described in this section is generally transparent to the user; therefore, detailed knowledge of this material is not essential for effective user-mode utilization of the disk.

In the monitor there is one set of disk-independent routines for handling servicing of all disk and drum units. The set of routines interprets and operates on file structures, processes disk monitor calls, queues disk requests, and makes optimization decisions.

All queues, statuses, and flags are organized by logical disk unit rather than physical disk unit. The monitor primarily deals witn logical units within file structures, and it converts to physical units in the small device-dependent routines just before issuing I/O commands. The device-dependent routines perform the I/O for specific storage devices and translate logical block numbers into physical disk addresses. All references made to disk addresses refer to the logical or relative addresses used by the system, and they do not refer to a physical addressing scheme (which records sectors and tracks of a particular device). The basic addressable logical disk block is 200 (octal) 36-bit words.

8.2.1 File Directories

A directory is a file which contains pointers to other files on the disk. There are three levels of directories in each file structure:

1. the master file directory (MFD),
2. the user file directory (UFD), and
3. sub-file directories (SFDs).

The master file directory consists of two-word entries, which are the names of the user file directories on the file structure. The first word of the entry contains the project-programmer number of the user. The left half of the

second word of the entry contains SIXBIT/UFD/; and the right half contains a pointer to the first cluster of the user file directory (refer to Figure 8-1). The main function of the master file directory is to serve as a directory of individual user file directories. A continued MFD is the MFDs of all file structures in the job's search list.

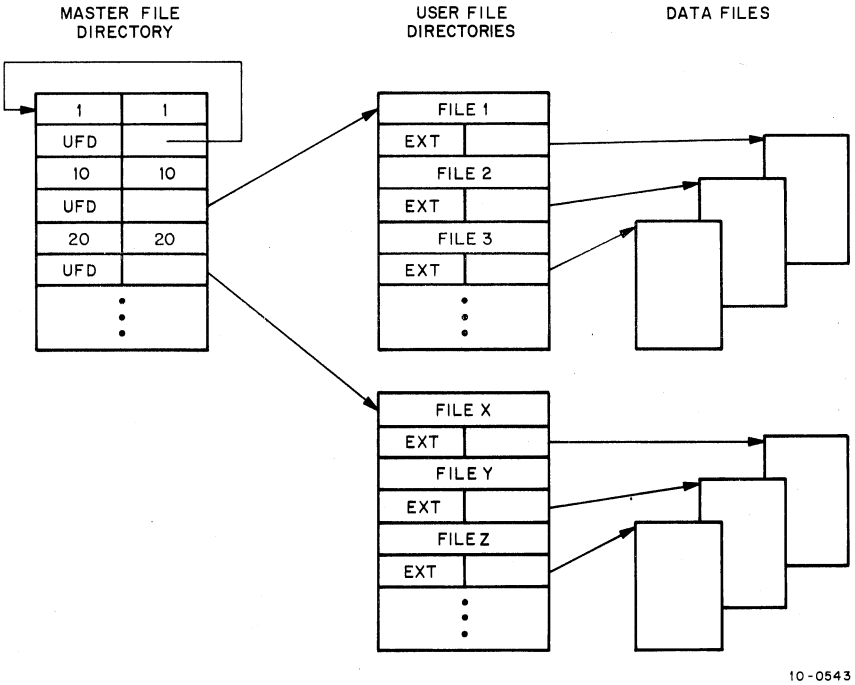


Figure 8-1. Basic Disk File Organization for Each File Structure

The entries within a user file directory are the names of files existing in a given project-programmer number area within the file structure. The first word of each entry contains the file name in SIXBIT. The left half of the second word contains the file name extension in SIXBIT, and the right half contains a pointer to the first cluster of the file (see Figure 8-1). This pointer specifies both the unit and the block number of the file structure in which the file appears. The right half of the directory entry is referred to as a compressed file pointer (CPF). A continued UFD is all the UFDs for the same project-programmer number on all file structures on the job's search list.

When the user logged-in, each file structure for which he has a quota contains a UFD for his project-programmer number. Each UFD contains the names of all the user's files for that file structure only. UFDs are created only by privileged programs (i.e., LOGIN in response to a LOGIN command, and OMOUNT/UMOUNT in response to a MOUNT command). A user is not prevented from attempting to read a file in another user's UFD on a file structure for which he does not have a UFD. Whether or not the user is successful depends on the protection specified for the file being referenced.

As an entry in the user file directory, the user can include a sub-file directory (SFD), The sub-file directory is similar to the other types of directories in that it contains as data all of the names of files within its directory. This directory is pointed to by a UFD or a higher-level SFD nested in any arbitrary tree structure. The maximum number of nested SFDs allowed is defined via a MONGEN question and can be obtained from a GETTAB table (GETTAB Table .GTLVD, Item Number 17). Files can be written or read in SFDs nested deeper than the maximum, but SFDs cannot be created. (There is an absolute maximum of six, including the UFD.) Unlike UFDs, a sub-file directory can be created by any program. A continued SFD, or sub-directory, is all of the SFDs on all file structures in the job's search list with the same name and path.

This third level of directory allows groups of files belonging to the same user to be separate from each other. This is useful when organizing a large number of files according to function. In addition, simultaneous batch runs of the same program for a single user can use the same file names without conflicting with each other. As long as the files are in different sub-file directories, they are unique. Refer to Figure 8-2.

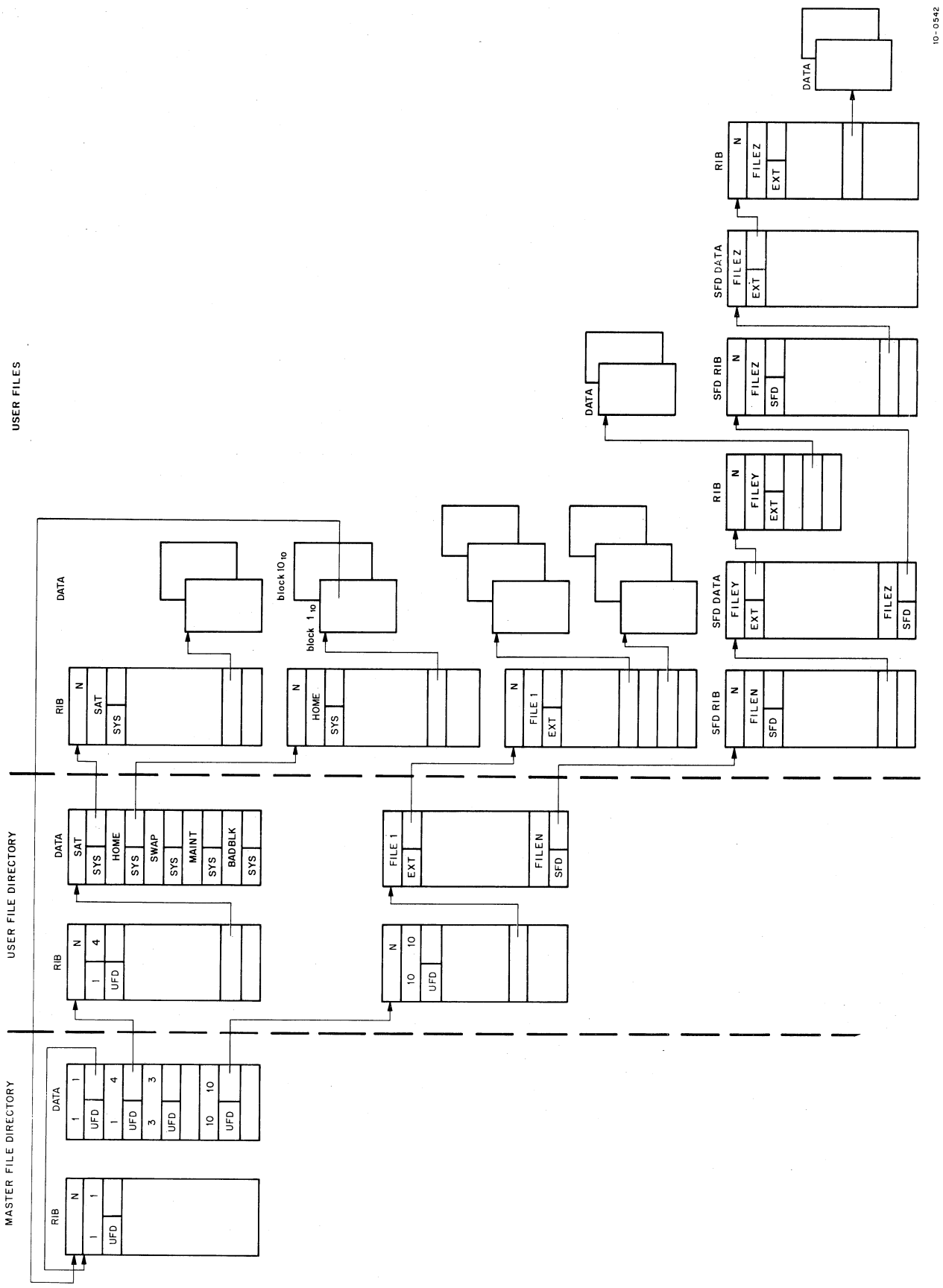


Figure 8-2. Disk File Organization

A file is uniquely identified in the system by a file structure name, a directory path, a file name and an extension. The directory path is an ordered list of directory names, starting with a UFD, which uniquely specifies a directory without regard to a file structure. The PATH monitor call is used to set or read the default directory path for a job (refer to section 6.7). Default paths can be a job's UFD, an SFD in a job's UFD, a UFD different from the job's UFD, or an SFD in another UFD. If a default path is not specified, it is the job's UFD. The notation file.extension [ppn,A,B,...,n] designates the file named file.extension in the UFD ppn in the SFDn, which is in the SFD..., which is in the SFD A. A path to the file name file.extension is [ppn,A,B,...,n].

To improve disk access and core searching times, only UFD names are kept in the MFD (project-programmer number 1,1). All system programs are contained in another project-programmer number directory called the system library (SYS: [1,4]). For convenience, both to users typing commands and to user programs, device name SYS: is interpreted as the system library; therefore, no special programming is required to read a specified file from device SYS:.

8.2.2 Job Search List

To a user, a file structure is like a device, that is, a file structure or a set of file structures may be specified by an INIT/OPEN/FILOP. monitor call or by the first argument of the ASSIGN or MOUNT operating system command. A console user specifies a file structure by naming the file structure and following it with a colon.

There is a flexible naming scheme that applies to file structures; however, most user programs INIT device DSK:, which selects the appropriate file structure, unless directed to do otherwise by the user. The appropriate file structure is determined by a job search list. A job search list is divided into two parts:

1. an active search list (usually referred to as the job search list), and
2. a passive search list.

The active search list is an order list of the file structures that are to be searched on a LOOKUP or ENTER when device DSK: is used. The passive search list is an unordered list of file structures maintained by the monitor for LOGOUT time. At this time, LOGOUT requires that the total allocated blocks on each UFD in both the active and passive search lists be below the logged-out quota. Each job has its own active search list (established by LOGIN) with file structures in the order that they appear in the administrative control file AUXACC.SYS. Therefore, a user has a UFD for his project-programmer number in each file structure in which LOGIN allows him to have files. With the MOUNT command, mounted file structures may be added to the active search list. The following is an example of a search list.

DSKB, DSKA, FENCE, DSKC

DSKB and DSKA comprise the active search list. These file structures are represented by generic name DSK for this job. DSKC is the name of a file structure that was previously in the active search list. FENCE represents the boundary between the active and the passive search lists.

Each file structure in a job search list may be modified by setting one or two flags with the JOBSTR monitor call:

1. Do not create in this structure if just generic DSK is specified.
2. Do not write in this file structure.

Setting the "do not create" flag indicates that no new files are to be created on this file structure unless explicitly stated, For example, if the "do not create" flag is set

DSKA:FOO=

allows FOO to be created on DSKA, but

DSK:FOO=

does not. For LOOKUPs on device DSK, the monitor searches the structures in the order specified by the job's search list. For ENTERs where the file name does not exist (creating, see below), the file is placed on the first file structure in the search list that has space and does not have the "do not create" flag set. For ENTERs when the file name already exists on any file structure in the search list, the file is placed on the same structure that contains the older file. If the write-lock bit is set for the file structure, a write-lock error (ERWLK%) is given. Because superseding is treated differently from creating, a user may explicitly place a file on a particular file structure, for example, a fast one with the "do not create" flag set, so that subsequent supersedes will remain on that file structure even though generic DSK is used.

8.2.3 Storage Allocation Table (SAT) Blocks

Unique to each disk structure is a file named SAT.SYS. This file reflects the current status of every addressable block on the disk. Only the monitor can modify the contents of SAT.SYS as a result of file creation, deletion, or space allocation, although this file may be read by any user. The SAT file consists of bits indicating both the portion of file storage in use and the portion that is available. To reduce the size of SAT.SYS, each bit can be used to represent a contiguous set of blocks called a cluster. Monitor overhead is decreased by assigning and releasing file storage in clusters of blocks rather than in single blocks.

If a particular bit is set, it indicates that the corresponding cluster is bad, nonexistent, or allocated to a file. It may or may not contain data (i.e., files may contain allocated but unwritten clusters). If the bit is cleared, it indicates that the corresponding cluster is empty, or available to be written on.

It is recommended that cluster sizes should evenly divide blocks on a unit. The refresher truncates to the largest number of full clusters. With truncation, the last few blocks are not included in the addressing space, but may be used for swapping; therefore, they are not part of SWAP.SYS even though they are in the swapping space. In addition, any bad blocks in the extra blocks are not included in SWAP.SYS.

8.3 DISK FILE FORMAT

All disk files consist of two parts

1. pure data, and
2. information needed by the system to retrieve the data.

Each disk data block is 200_8 36-bit words in length. If a user outputs a partially-filled buffer to a disk unit, a full block of data will be written to the disk (i.e., trailing zeroes will be appended at the end to fill the 200_8 words). On input at a later time, the block will appear to have 200_8 words. The word counts associated with the individual disk blocks are not retained by the monitor, except for the last block in the file.

The monitor references data on the disk by examining a chain composed of three links. The chain is transparent to the user, but the chain may be thought of as being analogous to a DECtape directory. The first link in the chain is a two word directory entry that points to a second link, which is the retrieval information block (i.e., RIB). The RIB points to the third link, which is the first data block in the file.

The RIB contains pointers to the entire file. The information in the RIB is stored and accessed separately from the file data, increasing system reliability because of the decreasing probability of destroying RIB information. The number of positionings needed for random access is reduced, improving system performance.

When a file is CLOSEd, the monitor writes a copy of the RIB immediately after the last data block in the file. If the first RIB is bad or lost, the monitor allows a recovery program to use the copied RIB. Consequently, a data file of *n* blocks has two overhead blocks:

1. relative block number 0, which contains the primary RIB, and
2. relative block number n + 1, which contains the copied RIB.

8.4 ACCESS PROTECTION

8.4.1 File Access Privileges

Every file has a protection code associated with it indicating who may or may not access the file. The protection code is stored in 9-bits of the RIB. The 9-bits are divided into 3 classes:

1. the first three bits refer to the owner of the file,
2. the second three bits refer to users with the same project number as the owner of the file (i.e., any user logged-in under the same project number as the owner, regardless of programmer number), and
3. the last three bits refer to all other users.

Ordinarily, the owner of a file is the user whose programmer number matches the directory containing the file, regardless of project numbers.

NOTE

Installations should not assign the same programmer number to different users, even when the users are on separate projects. A user who works on several projects may retain his programmer number and retain ownership of all of his files (regardless of their associated project number).

The definition of the file owner is determined at monitor generation time (via MONGEN). No matter what an installation defines as the file owner, project numbers 0 through 7 are always independent of the project-programmer number. (I.e., a user having a programmer number of 1234,4 is not considered the owner of files in 1,4).

The access protection codes and their meanings are listed in Table 8-1.

Table 8-1
Access Protection Codes

Code	Mnemonic	Meaning
7	.PTNON	The greatest protection of file access, which means there are no access privileges. The owner may LOOKUP the file to change the access code (via RENAME). For the file owner, an access protection code of 7 is equal to codes 6 and 5.
6	.PTEXO	The file can only be executed. If any user, other than the file owner, issues a LOOKUP, the error return will be taken. The only way to access the file is via the RUN or GET operating system commands, and the RUN monitor call.
5	.PTRED	The class of users can read and/or execute the file.
4	.PTAPP	The class of users can append to the file, read the file, and/or execute it.
3	.PTOPD	The class of users can update, read and execute the file, and/or append to the file.
2	.PTWRI	The class of users can write, update, append to, read and/or execute the file.
1	.PTREN	The class of users can rename the file, write, update, append to, read and/or execute the file.
0	.PTCPR	The class of users can change the protection code of the file, rename the file, write, update, append to, read and/or execute the file.

Figure 8-3 illustrates the 9-bit protection field of a file that has a protection code of 057.

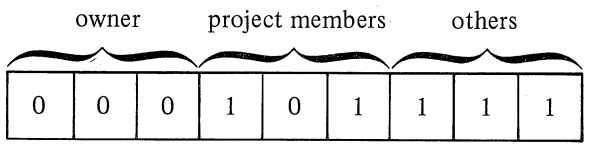


Figure 8-3. Access Protection Code

The access protection code illustrated in Figure 8-3 indicates

1. the owner of the file has complete file privileges (code 0),
2. the project members have read and execute privileges (code 5), and
3. all other users have no access privileges (code 7).

The greatest protection that a file can have is code 7, and the least protection is code 0. Usually the owner's field is 0 or 1. However, it is always possible for the owner of a file to change the access protection code associated with the file, even if the owner's protection is not set to 0. Therefore, codes 0 and 1 are equivalent when they appear in the owner's field. Access protection can be changed by executing the RENAME monitor call or by using the PROTECT operating system command as follows:

PROTECT *filename.extension* ⟨*nnn*⟩

where: *filename.extension* is the name and extension of the file for which its protection code is to be changed. ⟨*nnn*⟩ is the desired protection code for the file.

When an ENTER call is executed that specifies a protection code and the file does not exist, the monitor will substitute the standard protection code as defined by the installation. The normal system standard is 057. This protection prevents users in different projects from accessing another user's files; however, a standard protection of 055 is recommended for in-house systems where privacy is not as important as the capability of sharing files among projects. No program should be coded to assume knowledge of the standard protection code. If it is necessary to use this standard, it should be obtained through a GETTAB monitor call, refer to Chapter 19.

To preserve files with KJOB, a protection code of 1 in the owner's field should be associated with the files. KJOB preserves all files in a UFD for which the protection code for the owner is greater than zero. The PRESERVE operating system command can be used to obtain a protection code of 1 in the owner's field.

8.4.2 Directory Privileges

The protection code associated with each file completely describes the access rights to that file, independent of the protection code for the UFD. UFDs and SFDs can be read in the same manner as files, but they cannot be written explicitly. For UFD and SFD privileges, users are divided into the same three classes as for files. Each class has three independent bits which are listed in Table 8-2.

The owner is permitted to control the access to his own UFD and/or SFD. It is always legal for the owner to issue a RENAME to change the protection of his directories. Any program can create or delete SFDs; however, only privileged programs are allowed to create, supersede, or delete an UFD. The monitor checks for the following types of privileged programs:

1. jobs logged-in under project-programmer number 1,2 (e.g., FAILSAFE), and
2. jobs running with the JACCT bit set in JBTSTS (e.g., LOGIN and LOGOUT).

Table 8-2
Access Privileges to UFDs and SFDs

Bit	Access Privileges
4	Allow LOOKUPs in the UFD or SFD.
2	Allow CREATEs in the UFD or SFD.
1	Allow the UFD or SFD to be read as a file.

Privileged programs are allowed to

1. create UFDs (and SFDs),
2. delete empty UFDs (and SFDs),
3. set privileged arguments to LOOKUP, ENTER, and RENAME, and
4. ignore file protection codes.

UFD and SFD privileges are similar with the exception that SFDs can be RENAMEd and DELETEd by both privileged programs and the owner of the SFD, if the owner's protection field is 7.

8.5 FILE NAMES

Every file has a file name and, optionally, a file name extension written in the following format:

filename.extension

where: *filename* can be 1 to 6 characters in length.
extension can be 1 to 3 characters in length.
. must always separate the file name from the file name extension, when the file name extension is specified.

The name of the file is associated with a particular file when that file is created. At any time after that the name of the file may be changed via the RENAME command (refer to DECsystem-10 OPERATING SYSTEM COMMANDS) or the RENAME monitor call.

8.5.1 The RENAME Monitor Call (Op Code 50)

The RENAME monitor call performs the three function listed below:

1. It alters the file's name, the file name extension, and/or the file access privileges associated with a particular file.
2. RENAME can be used to change an SFD name, but an attempt to change the extension or project-programmer number associated with an SFD results in a protection error. An error also results if an attempt is made to alter the name, extension, or project-programmer number associated with a UFD or the project-programmer number of an ersatz device.
3. It deletes a file associated with a specified channel on a directory device.

a LOOKUP, an ENTER, or a FILOP. must be performed to identify the file for RENAME. The calling sequence for the RENAME operator is

```
        RENAME  channel,addr
          error return
        normal return

addr:   SIXBIT/filename/
        SIXBIT/extension/
        mode,time,date
        XWD project-no,programmer-no.
```

where: *channel* specifies the channel number associated with the device on which the file resides.

addr points to a two-word argument block (unless using the extended argument block, which is described in section 8.6.1.2 and Table 8-4.

filename is a 1 to 6 character file name (specified in SIXBIT, left-justified).

extension (bits 0-17) is 1 to 3 SIXBIT characters, left-justified.

project-programmer number can be either 0, XWD 0,*addr*, the default *ppn,* or not equal to the default *ppn.* If 0 is specified, no change will be made in the directory of the file. If XWD 0,*addr* is specified, the file is renamed into a new SFD/UFD according to the path specified by *addr*. If the default *ppn* is specified, the file will be renamed in that UFD. If something other than the above three alternatives is specified, the monitor will delete the directory entry from the old directory (UFD/SFD) and will insert the entry into the specified UFD.

The only way to RENAME a file into an SFD different from the one in which it currently resides is to give an explicit path via an argument block.

To delete a file only after all read references have been made, specify 0 in addr+1 of the argument block.

To set the access protection code, bits 0 through 8 are used in addr+2 in the argument block.

If the file name and the file name extension specified in the argument block differ from the current name and extension, a search is made for the specified name and extension. If a match is not found, the following occurs:

1. the current file name is changed to the file name specified in the argument block.
2. the current file name extension is changed to the extension specified in the left half of addr+1 in the argument block.
3. the access protection code is set to the contents of bits 0-8 in addr+2 of the argument block.
4. the access date is unchanged.

The error return is taken under the following conditions. An error code will be returned in the right half of .RBEXT.

1. When no file has been selected on the channel specified.
2. When the specified file cannot be found.
3. When the file specified is currently in the process of writing, superseding, or renaming.
4. When the user does not have the appropriate privileges to RENAME the file.
5. When the file name and extension specified differ from the current file name and extension, a search is made for the name and extension specified. If a match is found, the error return is taken.
6. The User File Directory (UFD) has been deleted.

If no device has been previously associated with the channel specified, the monitor stops the job and prints the following message on the user's terminal:

?I/O TO UNASSIGNED CHANNEL AT USER PC LOC *addr*

A CLOSE is optional because RENAME performs a CLOSE. However, a CLOSE should not be issued between a LOOKUP and a RENAME if the file is not in the default directory path or cannot be obtained from the default path by scanning because CLOSE erases all memory of the path of the file. If a CLOSE is performed and the file is not in the default path, the RENAME returns the FILE NOT FOUND error. In addition, disk accesses are minimized if a CLOSE does not precede a RENAME.

RENAME enters the information specified in addr through addr+3 (except for mode and access date) into the retrieval information and proper directory. If addr contains zero, RENAME has the effect of deleting the file.

Although only a privileged job can delete an empty UFD, any job can delete an empty SFD. If the directory is not empty or if a job is currently using the directory, the RENAME returns the DIRECTORY NOT EMPTY error. (Refer to Appendix E for the list of error codes.) Refer to section 8.6.1.4 for a note on error recovery procedures.

8.6 FILE SELECTION

8.6.1 LOOKUP/ENTER A File

There are several methods for writing on the disk, If a user does an ENTER with a file name not previously existent in his UFD, he is creating a file. If the file name was previously existent in his UFD, the user is superseding the file (i.e., the old generation of the file stays on the disk and is available to anyone who wants to read it until the user does the output CLOSE). At the time of a CLOSE, the user's UFD is changed to point to the new generation of the file, and the old generation is either deleted immediately or marked for deletion at a later time if someone is currently reading it. The space occupied by deleted files is always reclaimed in the SAT tables. Finally, if a user does a LOOKUP call followed by an ENTER using the same file name on the same channel, the user is able to modify selected blocks of that file, using USETI and USETO (refer to section 8.8) without creating an entirely new generation; this third method of writing, called updating, eliminates the need to copy a file when making a small number of changes. a LOOKUP followed by an ENTER and OUTPUT (in that order) writes the output to the beginning of the file. To append information to a file, a USETI -1 should be used after the ENTER call.

A standard practice, user programs should read, create, and supersede (new file with the same file name) files on different software I/O (i.e., user channels). However, for compatibility with DECtapes, it is possible to read and create, or read and supersede, two files on the same software I/O channel as long as a CLOSE monitor call is executed before successive LOOKUPs and ENTERs unless updating is intended. Updating, superseding, creating, and writing a file may be performed by one monitor call, FILOP., (instead of the LOOKUP, ENTER, USETI/USETO, INPUT/OUTPUT sequence). Refer to paragraph 8.6.2.1 for a description of the FILOP. monitor call.

To select a file for input the LOOKUP operator (op code 076) is used; to select a file for output the ENTER operator can be used (op code 077). The FILOP. monitor call can be used for selecting a file for either input, output, or update.

8.6.1.1 The LOOKUP Operator – The LOOKUP operator selects a file for input on a specified channel; its calling sequence is

```
        LOOKUP  channel, addr
          error return
        normal return
        .
        .
        .
addr:   SIXBIT/filename/
        SIXBIT/extension/
        0,,0
        XWD project-no,programmer-no.
```

where: *channel* specifies the channel number associated (via OPEN or INIT) with the device storing the specified file.

addr points top a four-word argument block described in Table 8-3.

filename is a 1 to 6 character, left-justified file name. (or, if a user file directory, it is the project number in the left half and the programmer number in the right half of the first argument word.)

extension is a 1 to 3 character file name extension, left-justified.

project-no.,programmer-no. can be a specified project,programmer number or it can be 0. If 0, the specified file will be searched for from the user's default directory (i.e., it is the project-programmer

number under which the user is logged-in). If a project-programmer number is specified, the directory specified will be searched. An entire path will be searched only if the SCAN switch has been set via the PATH. monitor call. Therefore, if a file's protection code permits reading and the UFD permits LOOK-UPs, other user's files may be read. If *project-programmer number* contains XWD 0,*addr* the file will be read according to the path specified at *addr*. A path specification specified in the LOOKUP argument block supersedes any default path specification made via the PATH. monitor call, refer to section 8.7.

On a normal return, the file has been found and the monitor returns information in the argument block as listed in Table 8-3.

Table 8-3
LOOKUP Argument Block

Word	Bits	Contents
addr+1	18-20	the high order 3 bits of the date on which the file was originally created.
addr+1	21-35	the date the file was last accessed in the format used by the DATE monitor call.
addr+2	0-8	the protection code for the file.
addr+2	9-12	the date mode of the file (e.g., ASCII, binary dump).
addr+2	13-23	the time that the file was originally created (represented as the number of minutes past midnight on the creation date).
addr+2	24-35	the low order 12 bits of the date on which the file was originally created.
addr+3	0-35	the length of the file.

On an error return, an error code is returned in the right half of *addr+1* (refer to Appendix E for a list of the possible error codes).

If a device was not previously associated with the specified channel (via an INIT, OPEN, or FILOP.), the monitor will stop the job and print the following message on the user's terminal.

?I/O TO UNASSIGNED CHANNEL AT USER LOC *addr*

8.6.1.2 The ENTER Operator – The ENTER operator causes the monitor to store a directory entry for later entry into the proper UFD or SFD, when the specified channel is closed or released. ENTER's calling sequence is

ENTER *channel,addr*
error return
normal return

where: *channel* is the channel number associated with the device storing the specified file.
addr is the address of a four-word argument block which is illustrated in Figure 8-4.

<table>
<tr><td colspan="4">filename</td></tr>
<tr><td colspan="2">extension</td><td>date1</td><td>date2</td></tr>
<tr><td>code</td><td>mode</td><td>time</td><td>date3</td></tr>
<tr><td colspan="2">project-number</td><td colspan="2">programmer-number</td></tr>
</table>

Figure 8-4 ENTER Argument Block

where: *filename* is a 1 to 6 character filename (specified as SIXBIT, left-justified).

extension (bits 0-17) is a 1 to 3 SIXBIT characters, left-justified.

date1 (bits 13-20) contains the high order three bits of the creation date. If a nonzero date is obtained by concatenating the high order three bits with the low order twelve bits in *date3*, the monitor will use that value as the creation date of the file. If the date is zero, the monitor will supply the high order three bits from the 15-bit value representing the current date.

date2 (bits 21-35) is the date that the file was last referenced. This date will be returned by the monitor; it is ignored if changed by the user.

code (bits 0-8) is the protection code for the file. If the protection code is 0, the monitor will substitute the installation standard as specified at monitor generation time (via MONGEN). If the code is 0 and this ENTER is superseding a file, the protection code for the new file is copied from the old file. The RENAME monitor call may be used to change the protection code of a file, after the file has been completely written and when the file is being closed.

mode (bits 9-12) is the data mode of the file, which is supplied by the monitor. The monitor obtains the information for mode by what the user set via the last INIT or SETSTS monitor call on the channel specified.

time (bits 13-23) is the time at which the file was originally created, represented as the number of minutes past midnight on the creation date. If time is 0, the monitor will supply the current time as the creation time.

date3 (bits 24-35) is the low order 12 bits of the creation date. If date3 and date1 are 0, the monitor will supply the current date as the creation date.

project-programmer number can be 0 or a specified project-programmer number.

If this word is 0, the file will be written in the default directory. For example, if the default path is [10,10,A], the file will be written in SFDA which is on [10,10]. The default path is determined by the PATH. monitor call. If the default path is not specified, the job's UFD will be used.

If this word contains a project-programmer number, the file will be written in the specified UFD (i.e., sub-directories will not be scanned). This facility allows the user program to write in the disk area under which the job is logged-in, although the default directory is different. Note that it is generally not possible to create (i.e., ENTER) files in another user's area of the disk, because UFDs are usually protected from all but the owner when creating files.

If this word contains XWD 0,*addr,* the file will be written according to the path specified by *addr.* The argument block beginning at *addr* is the same as that specified in the PATH. monitor call, except that the first two arguments are ignored. The scan switch (*addr+1*) is not needed since if the file is found in the specified directory, it will be superseded; and if it is not found, it will be created at the end of the path of the specified directory (even if the same name appears in an upper-level directory). A path specification in the ENTER block overrides any default path specification given in the PATH. monitor call.

On an error return, an error code will be returned in the right half of the extension word of the argument block. Appendix E lists the possible error codes that may be returned.

When issuing an update ENTER (i.e., an ENTER after a LOOKUP on the same channel), the user should check that locations *addr* through *addr+3* contain the appropriate information.

When an ENTER is executed on an existent file, a new file with the same name is written, the old file is deleted when the CLOSE (or RELEASE) monitor call is executed (providing that bit 30 of the CLOSE is 0). I.e., the deletion of the previous version of the file does not occur until output CLOSE time. Consequently, if the new file is aborted while only partially written, the old generation remains. The normal return is taken; the monitor makes the file entry; and the monitor records the file information.

Unless a specific unit is INITed on the specified channel, the monitor will maximize the job's throughput by selecting some unit in the file structure for which the job has no opened files, if such exists. Therefore, programs should LOOKUP all input files before ENTERing output files.

If no device has been associated with the specified channel (via INIT or OPEN), the monitor will stop the job, and it will print the following message on the user's terminal:

?I/O TO UNASSIGNED CHANNEL AT USER PC *addr*

8.6.1.3 Extended Arguments to LOOKUP, ENTER and RENAME – As an extension to the four-word argument block., the user can specify the length of the argument block. To do so, if *addr* contains a 0 in its left half and a value greater than or equal to 3 in the right half, the right half will indicate the count of the words in the argument block. If the right half of *addr* is less than 3, the error return will be taken (FILE NOT FOUND). The arguments that the user program is allowed to supply are returned to the monitor as values. If a user program supplies an illegal argument, the monitor will ignore the user-supplied argument, and it will supply default values on the return. Table 8-4 lists the arguments that can be supplied by a user program.

Table 8-4
Extended Arguments to ENTER, LOOKUP, and RENAME

Rel.	Mnemonic	LOOKUP	CREATE	SUPERSEDE	UPDATE	RENAME	Arguments and Values
0	.RBCNT	A		A	A		The count of the number of arguments which follow. Left half: unused, must be zero. Right half: number of arguments following. If bit 18 is set on an ENTER, that ENTER is a non-superseding ENTER. (Refer to Section 8.6.1.5)
1	.RBPPN	A0		A0	A0		A directory name (i.e., a programmer number) or a pointer. Left half: Project Number Right half: Programmer Number or path The project-programmer number is for the user file directory in which the file is to be LOOKed UP, ENTERed, or RENAMEd. To LOOKUP a Master File Directory, .RBPPN must contain a 1 in the left half and a 1 in the right half, indicating that the filename (i.e., .RBNAM) is to be LOOKed UP in [1,1]'s User File Directory (i.e., the Master File Directory).
2	.RBNAM	A		A	A		The SIXBIT filename, left-justified with trailing nulls. If the Master File Directory or the User File Directory is being LOOKed UP, ENTERed, or RENAMEd, this location contains the directory name. The argument can be 0 only on a RENAME, in which case the file will be deleted.
3	.RBEXT	A		A	A		Bits 0-17: The SIXBIT filename extension, left-justified with trailing nulls. Null extensions are discouraged because they convey no information.
		V		A0	A		Bits 18-20: The high order three bits of the 15-bit creation date (i.e., RB.CRX).
		V		V	V		Bits 21-35: The access date. If an error return is taken, bits 18-35 will contain the error code. Refer to section 8.6.1.4 for a special note on error recovery.

Table 8-4 (Cont.)
Extended Arguments to ENTER, LOOKUP, and RENAME

Rel.	Mnemonic	LOOKUP	CREATE	SUPERSEDE	UPDATE	RENAME	Arguments and Values
4	.RBPRV	V	A0		A		Bits 0-8: protection codes.
		V	V		A		Bits 9-12: Data mode in which the file was created (RB.MOD).
		V	A0		V		Bits 13-23: creation time in minutes since midnight (RB.CRT).
		V	A0		A		Bits 24-35: the low order 12 bits of the 15-bit creation date in standard format (RB.CRD).
5	.RBSIZ	V	V		V		The written length of the words written into the file. This argument is ignored, and a value is always returned.
6	.RBVER	V	A		A		The octal version number as in .JBVER.
7	.RBSPL	V	A		A		The filename to be used to label the output from a spooled device. The filename is specified on an ENTER to the spooled device, or it is 0 if an ENTER has not been performed.
10	.RBEST	V	A		A		The estimated length of the file, in positive number of blocks. On the execution of an ENTER call, the monitor uses this value as the number of blocks to allocate for the file. If the requested number of blocks cannot be allocated, partial allocation is performed, and the normal return is taken. .RBALC always contains the actual number of blocks allocated.
11	.RBALC	V	A		A		The number of contiguous 128-word blocks allocated to a file when an ENTER or RENAME call is performed. The number of blocks includes the RIBs of the file also, and it is equivalent to the last block number of the file. .RBALC equal to 0 does not change the allocation of the file. All of the data blocks can be deallocated by superseding the file and performing no output before the CLOSE. This argument can be used to allocate additional space onto the end of the file,

Table 8-4 (Cont.)
Extended Arguments to ENTER, LOOKUP, and RENAME

Rel.	Mnemonic	LOOKUP	CREATE	SUPERSEDE	UPDATE	RENAME	Arguments and Values
							deallocate previously allocated but unwritten space, or truncate written blocks. The smallest unit of disk space that the monitor can allocate is a cluster of 128-word blocks. Typically, small devices use a cluster size of 1-block. If the number of blocks allocated is not equal to the last block of a cluster, the monitor will round up, thereby adding a few more blocks than the user requested. The partial allocation error (error code 17) will be returned; however, a user may still write the file. **NOTE** To create a file of prespecified length, perform an extended ENTER with .RBEST set and .RBALC cleared. To create a file of prespecified length with contiguous blocks, perform an extended ENTER with .RBALC set and .RBEST cleared. After an ENTER, .RBALC will contain the accurate allocated file length.
12	.RBPOS	V	A		A		The logical block number of the first allocated block for a new group of clusters appended to the file. The logical block number is specified with respect to the entire file structure, beginning with block number 0. Combined with the DSKCHR call, this feature allows a program to allocate a file with respect to tracks and cylinders for maximum efficiency when the program is executed.
13	.RBTF1	V	A		A		Reserved for Digital.
14	.RBNCA	A	A		A		Reserved for customer definition.
15	.RBMTA	V	A1		A1		A 36-bit tape label, if the file has been put onto magnetic tape. If the allocated space is zero, the file was deleted from the disk when it was copied onto the magtape. The argument is accepted only from privileged programs; otherwise, it is ignored.

Table 8-4 (Cont.)
Extended Arguments to ENTER, LOOKUP, and RENAME

Rel.	Mnemonic	LOOKUP	CREATE	SUPERSEDE	UPDATE	RENAME	Arguments and Values
16	.RBDEV	V	V		V		The logical name of the unit on which the file is located. This value is ignored as an argument, but it is returned as a value.
17	.RBSTS	V	A1		A1		The File Status Word. Left Half: The status of the UFD. Right Half: The status of the file. Refer to Table 8-6, for the bit definitions of this word.
20	.RBELB	V	V		V		The logical block number within the unit on which the last data error or search error (IO.DTE) occurred, as opposed to the block within the file structure. This value is set in the RIB by the monitor when a CLOSE is executed and the hardware either detects a hard parity error or a search error while reading or writing the file. Device errors, checksum errors, and redundancy errors are not stored in this location. This argument is ignored, but a value is returned.
21	.RBEUN	V	V		V		Left half: The logical unit number within the file structure on which the last bad region was detected. Right half: The number of bad blocks in the last detected bad region. The bad region may extend beyond the file. This argument is ignored, and a value is returned.
22	.RBQTF	V	A1		A1		Meaningful for the UFD only. .RBQTF contains the logged-in quota. This quota is the maximum number of data and RIB blocks that can be in this structure's directory while the user is logged-in. The UFD and the UFD's RIB are not included in this count. The argument is ignored unless it is from a privileged program.

Table 8-4 (Cont.)
Extended Arguments to ENTER, LOOKUP, and RENAME

Rel.	Mnemonic	LOOKUP	CREATE	SUPERSEDE	UPDATE	RENAME	Arguments and Values
23	.RBQTO	V	A1		A1		Meaningful for the UFD only. .RBQTO contains the logged-out quota. This quota is the maximum number of data and RIB blocks that can be left in this structure's directory after the user logs-off. LOGOUT requires that the user be below this quota to log-off. The argument is ignored unless it is from a privileged program.
24							Reserved.
25	.RBUSD	V	A1		A1		Meaningful for the UFD only. .RBUSD contains the number of data and RIB blocks in this structure's directory when the user last logged-off. LOGIN reads this word so that it does not have to LOOKUP all files to set up the number of written blocks. LOGIN sets bit 0 of the file status word (see Table 8-5), and LOGOUT clears it in order to indicate whether LOGOUT has stored the quantity. The argument is ignored unless it is from a privileged program.
26	.RBAUT	V	A1		A1		The project-programmer number of the creator or superseder of the file, as opposed to the owner of the file. Usually the author and the owner are the same. Only when a file is created in a different directory are these different. This argument is used by MPB I to validate queue entries in other directories. The argument is ignored unless it is from a privileged program.
27							Reserved.
30							Reserved.
31	.RBPCA	V	A1		A1		Privileged argument reserved for customer definition.
32	.RBUFD	V	V		V		The logical block number (not the cluster number) in the file structure of the RIB for the UFD (in which the filename appears).

Table 8-4 (Cont.)
Extended Arguments to ENTER, LOOKUP, and RENAME

Rel.	Mnemonic	LOOKUP	CREATE	SUPERSEDE	UPDATE	RENAME	Arguments and Values
33	.RBFLR	V	V		V		The relative block number of the file to which the first pointer of this RIB points; this is used for multiple RIBs (e.g., 0 = prime RIB).
34	.RBXRA	V	V		V		The extended RIB address (i.e., the logical unit number and the cluster address of the next RIB in a multiple RIB file).
35	.RBTIM	V	V		V		The date and the time of creation of the file, in the universal date-time standard. Left half: the date of the creation. Right half: the time of the creation.

A = an argument (supplied by either a privileged or unprivileged program) and returned by the monitor as a value.

A0 = an argument like A, except that a 0 argument causes the monitor to substitute a default value.

A1 = an argument if supplied by a privileged program; if supplied by an unprivileged program, it is ignored.

V = the value returned by the monitor cannot be set even by a privileged program; the monitor will ignore the argument.

Table 8-5
.RBSTS Bit Definitions

Bit	Mnemonic	Meaning
0	RB.LOG	RB.LOG = 1 if the user is logged in; LOGIN sets this bit; LOGOUT clears it.
9	RB.UCE	RB.UCE = 1 if any file in this UFD has had a software checksum error or a redundancy check error.
10	RB.UWE	RB.UWE = 1 if any file in this UFD has had a hard data error while writing.
11	RB.URE	RB.URE = 1 if any file in this UFD has had a hard data error while reading.
18	RB.DIR	The file is a directory file; this protects the system from a user trying to modify a directory file. The protection error is given if the extension UFD is specified on an ENTER or RENAME and this bit is not set.
19	RB.NDL	The file cannot be deleted, renamed, or superseded, even by a privileged program.
21	RB.NFS	The file should not be dumped by disk backup programs because certain files (e.g., SWAP.SYS, SAT.SYS) contain no useful data to write on the tape.
22	RB.ABC	The file always has bad checksums (because the monitor never recomputes a checksum) e.g., SWAP.SYS, SAT.SYS.
25	RB.NQC	The file is a non-quota checked file.
26	RB.CMP	The UFD is being compressed.
27	RB.FCE	The file has a software checksum error or a redundancy check error (i.e., the IO.IMP bit has been set).
28	RB.FWE	The file has had a hard data error while writing. An entry is made in the BAT block so that the bad region is not reused.
29	RB.FRE	The file has had a hard data error while reading. An entry is made in the BAT block so that the bad region is not reused.
32	RB.BFA	The file is bad because of a tape read error during a restore.
33	RB.CRH	The file was closed after a crash.
35	RB.BDA	The file has been marked as bad by a damage assessment program.

8.6.1.4 Error Recovery for ENTER and RENAME Monitor Calls – Error codes for the LOOKUP, ENTER, and RENAME monitor calls are returned in the right half of location *addr+1* of the four-word argument block, and in the right half of location *addr+3* (.RBEXT) in the extended argument block. The error code overwrites the high order three bits of the creation date and the entire access date. Since most programs recover from these errors either by aborting or by reinitializing the entire argument block, this overwriting of data does not cause any problems. However, a small number of programs may attempt to recover from an error by fixing just the incorrect part of the argument and then reexecuting the monitor call. These programs should always restore the right half of *addr+1* before reexecuting the ENTER or RENAME monitor call. (To eliminate problems when recovering from an error in a file with a zero creation date, error codes are restricted to a maximum of 15 bits). The access date is forced to be greater than or equal to the creation date, but never greater than the current date.

8.6.1.5 Non-Superseding ENTER – If bit 18 in the .RBCNT word is set (refer to Table 8-4) on an ENTER, that ENTER is a non-superseding ENTER. In other words, if a file exists with the same filename as that filename specified in the current ENTER, that file cannot be superseded; and the ENTER will fail. The failure of the ENTER monitor call causes an error code to be returned. Error code 4 (ERAEF%) will be returned in location *addr+1* indicating that an already existent filename was specified.

8.6.2 Data Transmission

The IN and INPUT operators transmit data from the file selected on a specified channel to the user's core area. Their calling sequences are

```
INPUT   channel,addr
```

and

```
IN channel,addr
normal return
   error return
```

where: *channel* specifies a software I/O channel.
addr, if specified, is the effective address of the next buffer to be used. If not specified, the next buffer in sequence is implied.

The OUT and OUTPUT operators transmit data from the user's core area to the file selected on the specified channel. Their calling sequences are

```
OUTPUT channel,addr
```

and

```
OUT  channel,addr
normal return
   error return
```

If the specified channel has not been initialized, the monitor will stop the job, and print the following message:

?I/O TO UNASSIGNED CHANNEL AT USER *addr*

If the device is a multiple-directory device, and no file has been selected on *channel*, bit 18 of the file status word is set, and control will be returned to the user's program. Control always returns to the location immediately following an INPUT (op code 066) and an OUTPUT (op code 067). A check of the file status for end-of-file and error conditions must then be made by another monitor call.

Following an INPUT, the user program should check the word count of the next buffer to determine if it contains data. Control returns to the location immediately following an IN (op code 056) if no end-of-file or error condition exists (i.e., if bits 18 through 22 of the file status word are 0). Control returns to the location immediately following an OUT (op code 057) if no error condition or end-of-file exists (i.e., if bits 18 through 21 and bit 25 are 0). Otherwise, control returns immediately to the second location following the IN/OUT. Note that IN and OUT are the only calls where the error return is a skip and the normal return is not a skip.

8.6.2.1 The FILOP. Monitor Call (CALLI 155) – The FILOP. monitor call allows a user program to create, read, write, update, append to, or close a file, or the user program may optionally request the monitor to update a RIB of a file. FILOP.'s calling sequence is

```
MOVE   ac, [XWD length,addr]
FILOP. ac,
   error return
normal return
```

where: *length* is the length of the argument block pointed to by addr.
addr is the address of an argument block represented in Figure 8-5.

	0 – 17	18 – 35
.FOFNC	*X* / *channel*	*function code*
.FOIOS	*I/O Mode*	
.FODEV	*device name (or UDX)*	
.FOBRH	*address of output buffer header*	*address of input buffer header*
.FONBF	*number of output buffers*	*number of input buffers*
.FOLEB	*0*	*address of LOOKUP/ENTER Block*
.FOPAT	*length of PATH.*	*address of PATH. Block*

Figure 8-5 FILOP. Argument Block

where: *X* can be 0 or 1; if 1 and the job is running with the JACCT bit set or the job is running under [1,2] then file protection codes will not be checked. If 0 then protection will be checked even for [1,2] and JACCT programs.

function code is one of the function codes listed in Table 8-6.

I/O mode is one of the data modes (refer to Table 7-5).

device name is either the physical or the logical name of the device (or its UDX).

output buffer header is a pointer to the output buffer ring header.

input buffer header is a pointer to the input buffer ring header.

output buffers is the number of output buffers which are to be built. If this is 0, no buffers will be built. If -1 the default number of buffers will be built.

input buffers is the number of input buffers which are to be built. If this is 0, no buffers will be built. If -1 the default number of buffers will be built.

LOOKUP/ENTER Block is a pointer to the LOOKUP and ENTER blocks, refer to Table 8-3.

length is the length of the PATH. block, refer to section 8.7.

PATH. Block is a pointer to the PATH. block to be returned, refer to section 8.7.

Table 8-6
FILOP. Function Codes

Code	Name	Meaning
1	.FORED	The file is to be read only. No output will be done.
2	.FOCRE	A new file is to be created. If the file already exists then the error return will be taken with error code 4 (ERAEF%) returned in the AC.
3	.FOWRT	The file is to be written. A new file is created or an old file is superseded.
4	.FOSAU	The file is to be updated in single access mode. Only one job may open a file for single access update at any one time.
5	.FOMAU	The file is to be updated in multi-access mode. Any number of jobs may read or write the file at any one time.
6	.FOAPP	The file is to be appended to. **NOTE** If the file is to be written in buffered mode and FILOP. is used to build the output buffers, the last block of the file will be read into the buffer and the byte-count and byte pointer set so that data will be written immediately after the last word of the file.
7	.FOCLS	The file is to be closed. After the CLOSE a GETSTS is performed to read the file status into the AC. If any error bits are set, the non-skip return is taken.
10	.FOURB	Checkpoint the file by writing all output buffers on the disk, updating directories and the end-of-file pointer. The file is still open and more I/O may be done. This function is meaningful only for files that are being written.

On an error return, an error code will be returned in the AC (refer to Appendix E for a list of error codes). If the error was encountered in the LOOKUP/ENTER block, the error code is also returned in the LOOKUP/ENTER block.

8.7 THE PATH. MONITOR CALL (CALLI 101)

The PATH. monitor call sets or reads the default directory path or reads the current directory path on a channel. The calling sequence for the PATH. monitor call is

```
MOVE  ac, [XWD length,addr]
PATH. ac,
   error return
normal return
```

where: *length* is the length of the argument block.
addr is the address of an argument block, which is represented in Figure 8-6.

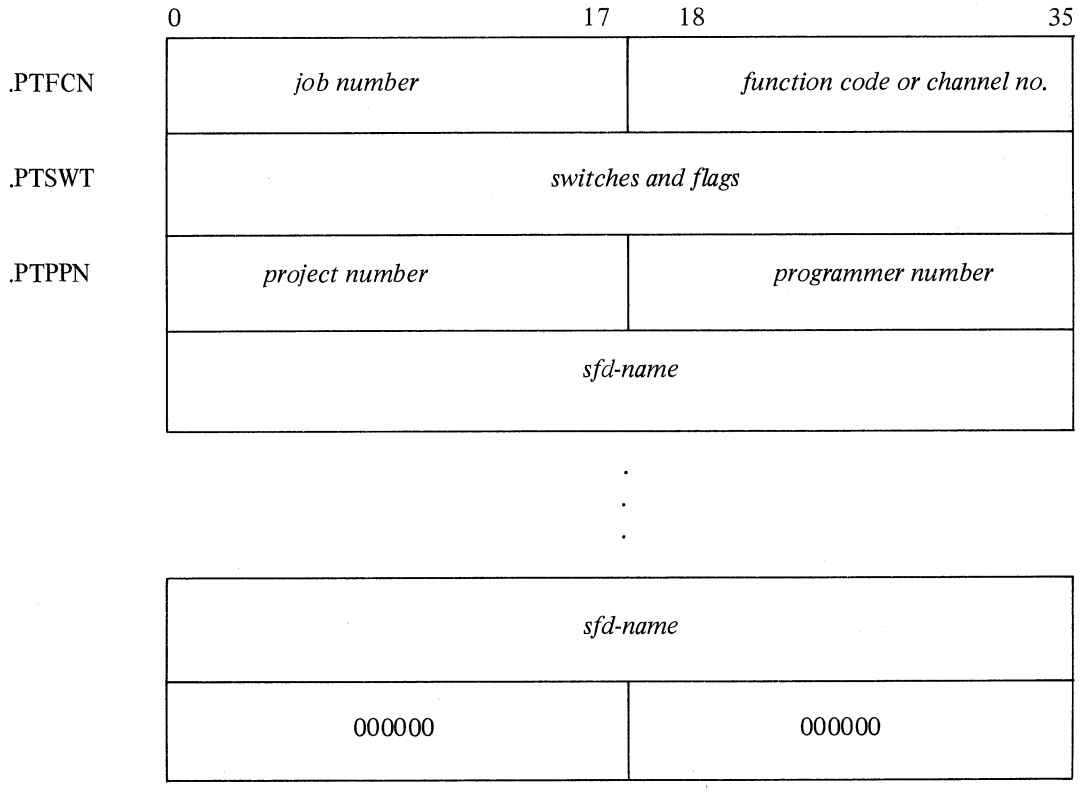


Figure 8-6 PATH. Argument Block

where: *job number* is the number of the specified job (in the range 1 to the highest legal job number, or the current job, if the job number is not in this range.)

function code is one of the functions listed in Table 8-7. (Optionally, this half-word can contain the channel number of the desired device.)

switches and flags listed in Table 8-8.

project and programmer numbers specify the user file directory used in the PATH.

Table 8-7
PATH. Function Codes

Function Code	Mnemonic	Meaning
-1	.PTFRD	Read the default directory path.
-2	.PTFSD	Define the default directory path.
-3	.PTFSL	Define the additional path to be searched when the file is not found in the user's directory path (i.e., set LIB:, NEW:, SYS:).
-4	.PTFRL	Return the additional path to be searched when a file is not found in the user's directory path (i.e., read LIB:, NEW, SYS:).

Table 8-8
PATH. Switches and Flags

Bit(s)	Mnemonic	Meaning
18-29	PT.SLT	The type of search list. code — mnemonic — meaning 1 — .PTSLJ — JOB search 2 — .PTSLA — ALL 3 — .PTSLS — SYS: search
30	PT.IPP	The implied project-programmer number. A user-supplied project-programmer number is ignored on an ENTER or LOOKUP call, and the implied project-programmer number of the device is used (e.g., [1,4] for SYS:). The implied project-programmer number is returned in addr+2.
31	PT.LIB	If there is a library, it is searched.
32	PT.SYS	The system directory (SYS:) is searched.
33	PT.NEW	The experimental system directory is searched (i.e., NEW:).
34-35	PT.SCN	The scan switch. code — mnemonic — meaning 1 — .PTSCN — The switch is off. 2 — .PTSCY — The switch is on.
34	PT.SNW	Search /NEW on function codes -3 and -4.
35	PT.SSY	Search /SYS on function codes -3 and -4.

When defining a path with a User File Directory (i.e., function code = -2), addr+1 is the scan switch, addr+2 is the default project-programmer number, and the rest of the argument (up to the first zero word) defines the default directory path. The scan switch determines whether or not the monitor will scan for the file on a LOOKUP.

If bit 34 contains 1 (.PTSCN), the monitor will examine the specified directory only; higher level directories will not be searched. If bits 34 and 35 contain 2 (.PTSCY), the following occurs:

1. The monitor will search the User File Directory or the Sub-File Directory (either by the explicit path or the default path). If the file is found, the scan will be terminated.
2. If the file is not found, the monitor will back up one directory along the path and will continue the scan (i.e., it will scan the directory in which the current Sub-File Directory appears). The scan will be terminated when the User File Directory is searched or when the file is found.

Since any file not found in the current sub-file directory may be obtained automatically from a higher level directory scanning allows directories to be nested. Nesting of directories is useful when a user has a default directory in use containing currently worked-on files and a higher level directory containing checked-out routines. Since sub-file directories are continued across file structures but the depth of directory nesting is not always the same on each structure, every scan searches the file structures that are

1. in the job's search list, and
2. have sub-file directories to the depth specified in the path.

The file structures are searched in the order that they appear in the search list.

When an ENTER is executed, the scan switch is ignored and if the file is found in the specified directory, the file will be superseded. If the file is not found, the file will be created at the end of the path in the specified directory, whether or not a file with the same name appears in a high level directory.

When an additional path is to be searched after the user's directory path is searches, addr+1 indicates whether SYS; or NEW: is to be searched (bits 34 and 35); addr+2 contains the project-programmer number to be used for a user library. These locations are used as follows:

1. If the file is not found in the user's directory path when a LOOKUP DSK: is executed, the directory specified in addr+2 is searched for the specified file. (The contents of addr+2 must specify a User File Directory, allowing users with different directory paths to share a common directory of files.)
2. If the file is not found in the library and bit 35 of addr+1 contains 1, SYS: is searched. When a LOOKUP SYS: is searched, executed and bit 34 of addr+1 is set, NEW: is searched before SYS:.

When running a path, addr+1 contains the information listed in Table 8-8 and addr+2 through addr+n-1 contains the path. If the path is less than n-1 words in length, a zero is stored at the end of the path. If addr contains a device name or a channel number when PATH. is called, the file structure name or ersatz device name is returned in addr, depending on the name specified (e.g., SYS is returned only if addr contains SYS and the job number does not have a device with the logical name SYS). If a LOOKUP or ENTER is performed on the specified device or channel number, the following information is returned in the argument block:

```
addr:    SIXBIT/file-structure-name
         SIXBIT/ersatz-device-name
addr+1:  scan switch
addr+2:  project-programmer number
addr+3:  actual path of the file
.
.
.
addr+m:  0      ;indicating the end of the path
                ;if m is less than n-1.
```

If a LOOKUP or an ENTER has not been performed on the specified device or channel number, the following information is returned in the argument block:

addr: SIXBIT/DSK/
SIXBIT/*ersatz-device-name*/
addr+1: scan switch
addr+2: job's default project-programmer number
project-programmer number of ersatz device
addr+3: the default path to the file
.
.
.
addr+m: 0 ;indicating the end of the path
;if m is less than n-1.

On an error return the following occurs:

1. The AC is unchanged if the PATH. monitor call has not been implemented on the system. (The sub-file directory will remain as a reserved extension, but all sub-file directory code will disappear). The GETTAB returning the maximum number of SFDs will return 0 or it will fail when executed. The default path is the user's project-programmer number.
2. The AC contains 0 if the device or the channel number is not a disk unit or associated with a disk unit, respectively.
3. The AC contains -1 if a sub-file directory in the path specification is not found.

8.7.1 PATH. Examples

Example A
The following example sets the default path to [27,4072,SUB] with no scanning in effect.

```
A:      -2
        1
        XWD 27,4072
        SIXBIT/SUB/
        0,,0
```

Example B
Refer to Figure 8-7. The path plus the file name for file A is X.MAC[10,63]. The path plus the filename for file B is Y.CBL[14,5]. The path plus the file name for file C is Z.ALG[14,5M].

Example C
Refer to Figure 8-8. The job's search list is DSKA/N, DSKB, DSKC, and the default path is [PPN,A,B,C,D].

1. LOOKUP DSK: with no matches scans in order: DKSA:D(.SFD), DSKA:C, DSKB:C, DSKA:B, DSKB:B, DSKA:A, DSKB:A, DSKA:PPN(.UFD), DSKB:PPN, DSKC:PPN.
2. LOOKUP DSK:file2 finds DSKA:file2[ppn,a,b,c].
3. LOOKUP DSKB:file2 or LOOKUP DSKC:file2 fails.
4. ENTER DSK:file9 receives an error since no file structure has both the no-create bit off and the directory structure [ppn,a,b,c,d].
5. ENTER DSKA:file1 creates the file at the end of the path on DSKA (the file designated by file1 in the diagram.

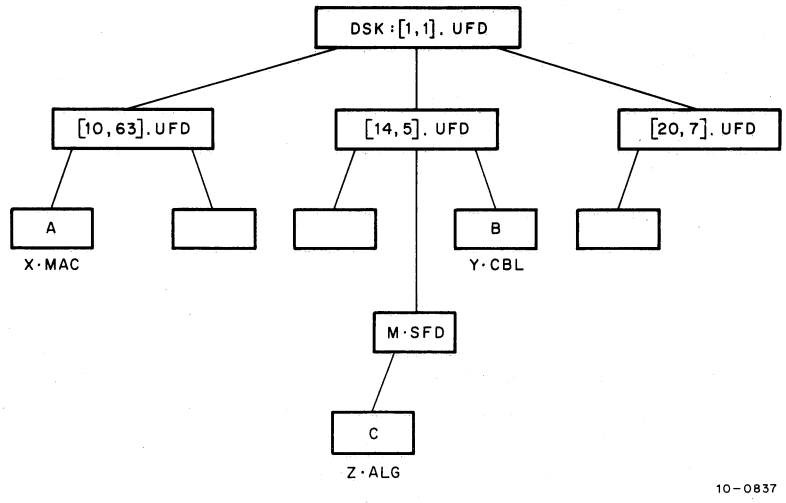


Figure 8-7. Directory Paths on a Single File Structure

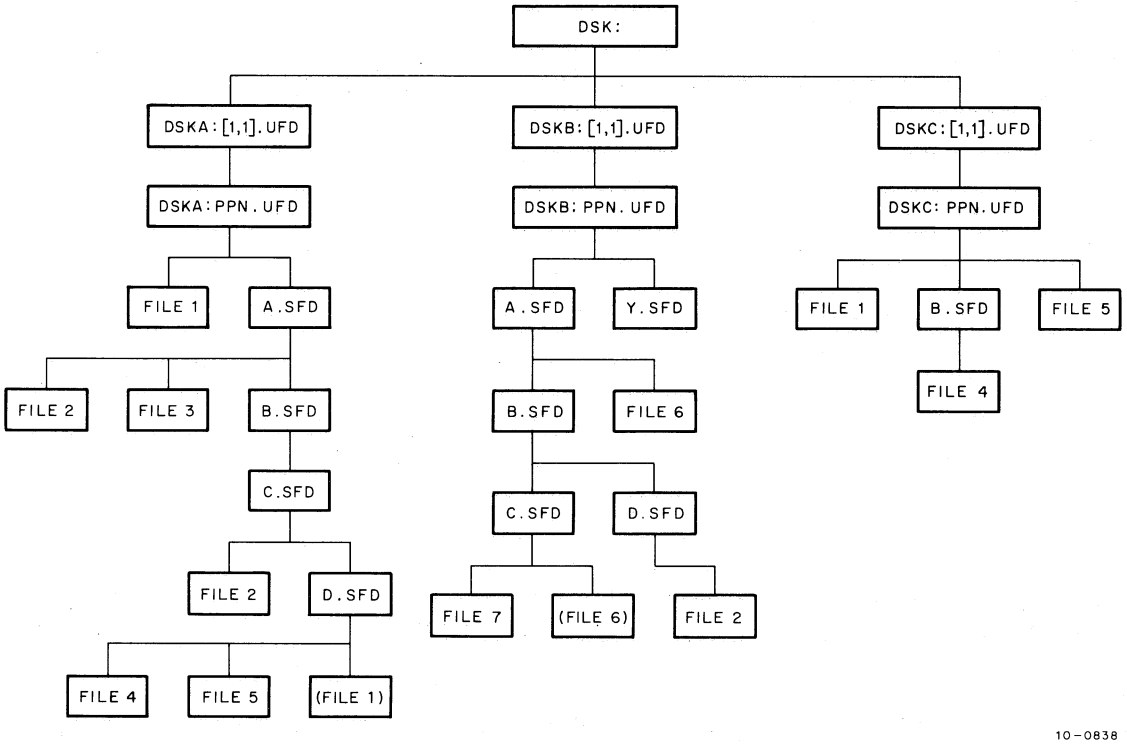


Figure 8-8. Directory Paths on Multiple File Structures

The default path is [ppn,a,b,c] :

1. ENTER DSK:file6 creates DSKB:file6[ppn,a,b,c] (the file designated by file6).
2. ENTER DSK:file2 supersedes file2 in DSKA:[ppn,a,b,c] .
3. LOOKUP DSK:file4 fails.
4. ENTER DSK:file7 supersedes file7 in DSKB:[ppn,a,b,c] .

Example D
The user defines the following path:

```
MOVE    1,[XWD5,A]
PATH.   1,
  HALT  .
MOVE    1,[XWD 3,B]
PATH.   1,
  HALT  .
A:      -2              ;define the default directory path
        2               ;scanning is in effect
        10,,63          ;THE UFD [10,63]
        SIXBIT/NAME/    ;the SFD [NAME]
        0               ;the default path is [10,63,NAME]
B:      -3              ;define an additional path
        3               ;both NEW: and SYS: are searched
        10,,7           ;the user library is [10,7]
```

If the user is logged in as [10,10] and does a LOOKUP DSK: FILTST, the following directories are searched in order:

```
NAME.SFD
[10,63].UFD    job's search list
[10.7].UFD
[1,5].UFD      system's search list
[1,4].UFD
```

If the user is logged in as [10,10] and does a LOOKUP DSK: PRJFIL [10,155], the following directories are searched:

```
[10,155].UFD   job's search list
[10,7].UFD
[1.5].UFD      system's search list
[1,4].UFD
```

8.8 USETI AND USETO MONITOR CALLS

The USETI/USETO monitor calls select a relative block to be either read or written in subsequent INPUT/OUTPUT calls to the specified channel number. The calling sequences are

USETI *channel, block*
USETO *channel, block*

where: *channel* is the number of the software I/O channel associated with the device storing the file.

block is a block number relative to the beginning of the file; the action performed by a USETI (USETO) depends on the value of block number, refer to Table 8-9.

The instruction sequence for reading a file is

```
LOOKUP
USETI
INPUT
```

The instruction sequence for writing a file is:

```
ENTER
USETO
OUTPUT
```

The instruction sequence for updating a file is

```
LOOKUP
ENTER
USETO          ;or USETI
OUTPUT         ;or INPUT
```

If an ENTER is not performed before a USETO, or a LOOKUP before a USETI, an illegal instruction trap will result. The monitor will stop the job and will print the following message on the user's terminal:

?ILLEGAL INST. AT USER *addr*

Table 8-9
USETI/USETO Function Codes

When the Block # is	LOOKUP USETI	ENTER USETO
<current file size 1 to 777777	the block is read on the next INPUT.	the block is written on the next OUTPUT.
>current file size 1 to 777777	IO.EOF is set in the file status word, causing the end-of-file return on the next INPUT.	data is appended to the end of the file. Monitor allocates the intervening blocks, writes zeroes in the 1st new block number -1, and writes the block.
0	the prime RIB is read.	the IO.BKT error bit is set in the file status word.
-2 to -10(8)	the specified block of the RIB is read.	an attempt is made to allocate a large number of blocks.
-1	IO.EOF is set in the file status word, causing the end-of-file return on the next INPUT.	the most recently input or output block re-written on the next OUTPUT.

The USETI/USETO calls do not actually perform any input/output; they change the pointers to the current position of a file. Each input/output instruction also logically advances the files. A program can rewrite/reread the same block by issuing a USETI/USETO before issuing an INPUT/OUTPUT. When a USETI/USETO is executed, the monitor will write all output buffers that the program filled before it changes the pointer to the files position.

Since the monitor reads/writes as many buffers as it can when an INPUT/OUTPUT is executed, it is difficult to determine which buffer the monitor is processing when a USETI/USETO is executed. Therefore, the INPUT/OUTPUT following the USETI/USETO might not be reading/writing the buffer that contains block number. A single buffer ring will read/write the desired block number since the device must stop after each INPUT/OUTPUT. A device with a multiple buffer ring can be stopped after each bufferful of data by setting bit 30 (IO.SYN) in the file status word via INIT., OPEN or SETSTS. USETI/USETO will then specify the buffer supplied on the next INPUT/OUTPUT.

Data can be appended to the last block of an append-only file provided that the protection does not prevent the job from accessing the file, and the feature test switch FTDAIR is set equal to 0 at MONGEN-time, by using a USETO to the last block followed by an OUTPUT. The monitor will read the block into the monitor buffer, copy words n + 1 minus 200 from the user's buffer into the monitor buffer, and rewrite the block. On an INBUF, the byte count size is set appropriately. The current length of the block can be obtained by examining the LOOKUP/ENTER block. It is not necessary that a user program read the last block of a file before appending to it; any data in the file is not changed.

When appending to the end of a file with APPEND-only protection, the IO.BKT error bit is set in the file status word and no output is performed when

1. any block before the last block is written,
2. the last block already contains 200 words, or
3. fewer words are written than the current size of the block.

If the last block of the file is OUTPUT, the size of the last block becomes 200 and cannot be appended to.

Multiple channels of a single job and/or multiple jobs can update a single file simultaneously. The monitor imposes no restrictions or interlocks when a file is being simultaneously updated; therefore, the user must insure that separate jobs do not update the same block of the same file at the same time. The Enqueue/Dequeue Facility may be used to insure that such interference does not occur, but it does not have to be used when simultaneously updating files.

To update a file simultaneously, perform a FILOP. monitor call with a function code of 5. A file can be updated in this manner when the file is idle, it is being read, or it is being simultaneously updated by other jobs. A file cannot be simultaneously updated if it is in the single-access update mode (i.e., a LOOKUP and an ENTER have been performed, or a FILOP. has been performed with a function code of 4 or 6.)

NOTE

For simultaneous update, even though an extended LOOKUP/ENTER block can be specified via FILOP., the user cannot change the attributes of the file; the current attributes will be returned in the extended argument block on a LOOKUP, and those same values will be used for the ENTER. In other words, FILOP., will use the values on the ENTER.

Files that are to be simultaneously updated should be pre-allocated into contiguous blocks, if possible, To prevent checksums from being changed while updating, allocate the file into one contiguous section and do not write block number 1; or do not change the first word of any block in the file.

8.9 THE SEEK MONITOR CALL (CALLI 56)

The SEEK monitor call, used in conjunction with USETI/USETO, controls the time at which positioning operations occur on an idle disk unit. The SEEK will position the disk to the cylinder containing the specified block number within the previous USETI/USETO. SEEK's calling sequence is

SEEK *channel,*
only return

The SEEK monitor call performs as a no-op if the disk is in any other state than an idle state.

SEEK may be issued for public and private file structures, allowing users to debug programs on a public pack and later run the same program on a private pack. The following is the proper instruction sequence for issuing a SEEK monitor call:

For output:
a. perform a USETO to select a block,
b. perform a SEEK to request positioning,
c. perform computations, and
d. perform an OUTPUT to request actual output.

For input:
a. perform a USETI to select a block,
b. perform a SEEK to request positioning,
c. perform computations, and
d. perform an INPUT to request actual input.

8.10 THE CHKACC MONITOR CALL (CALLI 100)

The CHKACC monitor call provides a consistent and uniform method for determining whether or not a file may be accessed. User programs should not make assumptions about the access rights of a file, but should employ CHKACC to insure that access is permitted. This is especially true for privileged programs that are constrained by the access privileges on a non-privileged project-programmer number for which they may be performing a task. For instance, QUEUE must check the access right of the user issuing a PRINT command to verify that the user is actually allowed to read the file.

CHKACC's calling sequence is

```
        MOVEI    ac,addr
        CHKACC   ac,
           error return
        normal return

addr:   access code,,protection
        project-programmer number A
        project-programmer number B
```

where: *access code* is the code for the type of access desired; the codes are listed in Table 8-10.

protection contains the protection code in bits 27-35. Refer to Table 8-2 for UFD protection (functions 7 and 10) codes and Table 8-1 for file protection codes.

project-programmer number A is the project-programmer number of the directory (UFD) containing the file.

project-programmer number B is the project-programmer number of the user who desires access to the specified file.

NOTE

For access codes 0 through 6, the monitor will ignore the directory protection. For codes 7 and 10, the monitor will ignore the file protection.

The error return is taken if the monitor call has not been implemented. On a normal return, the AC is set to zeroes if access is allowed to the specified file, and it is set to -1 otherwise.

The right to access a file is determined by:

1. the type of access desired to the file (e.g., read).
2. the project-programmer number of the user desiring access to the file,
3. the project-programmer number of the directory in which the file resides, and
4. the protection field of the file or the protection field of the directory.

Note that access to a file is not dependent on the file name. However, the file name is needed if a LOOKUP is to be performed (e.g., to obtain the protection field of the file).

Table 8-10
CHKACC Access Codes

Access Codes	Mnemonic	Meaning
0	.ACCPR	Caller would like to change the file's protection code.
1	.ACREN	Caller would like to RENAME the specified file.
2	.ACWRI	Caller would like to write the specified file.
3	.ACUPD	Caller would like to update the file (in old-style update mode).
4	.ACAPP	Caller would like to append to the end of the file.
5	.ACRED	Caller would like to read the specified file,
6	.ACEXO	Caller would like to execute the file.
7	.ACCRE	Caller would like to create the file in his UFD.
10	.ACSRE	Caller would like to read the directory as a file.

The following sample code checks to see if user [36,402] has access rights to the file specified as PRIVAT.TXT [1206,124].

```
        MOVEI   AC,ACRED        ;GET CODE FOR "READ FILE"
        HRLM    AC,CHKLOC       ;STORE TYPE OF ACCESS DESIRED
        LOOKUP  CH,LKPBLK       ;LOOKUP PRIVAT.TXT
           JRST ERROR           ;ERROR RETURN HERE IF
                                ;FILE CANNOT BE FOUND
        LDB     AC,[POINT 9,FILPRO,8]  ;GET FILES PROTECTION
                                ;FIELD
        HRRM    AC,CHKLOC       ;STORE PROTECTION CODE
        MOVE    AC,FILPPN       ;GET PPN OF DIRECTORY
        MOVEM   AC,CHKLOC+1     ;STORE DIRECTORY PPN
        MOVE    AC,[XWD 36,402] ;GET USERS PPN
        MOVEM   AC,CHKLOC+2     ;STORE USERS PPN
        MOVEI   AC,CHKLOC       ;SET UP FOR CHKACC
        CHKACC  AC,             ;GET ACCESS RIGHTS
                                ;FROM MONITOR
           JRST NOTIMP          ;ERROR RETURN HERE WHEN
                                ;CALL NOT IMPLEMENTED
        JUMPE   AC,ALLOWD       ;ACCESS IS ALLOWED
                                ;IF AC CONTAINS 0, ELSE
                                ;IF AC CONTAINS -1
                                ;ACCESS IS NOT PERMITTED
NOACCESS:       .
                .
                .
LKPBLK:
FILNAM: SIXBIT/PRIVAT/
FILEXT: SIXBIT/TXT/
FILPRO: 0
FILPPN: XWD 1206,124
CHKLOC: BLOCK 3
```

8.11 THE STRUUO MONITOR CALL (CALLI 50)

The STRUUO monitor call manipulates file structures, and it is intended primarily for monitor support programs. The call is described within the STRUUO Specification in the DEC-system-10 Notebooks.

8.12 THE JOBSTR MONITOR CALL (CALLI 47)

The JOBSTR monitor call returns the name of the next file structure in the job's search list, along with other information about the file structure. This call is used by DIRECT to list a user's directory correctly and specify in which file structure the files reside, as well as the order in which they are scanned. JOBSTR's calling sequence is

```
        MOVE    ac, [XWD length,addr]
        JOBSTR  ac,
          error return
        normal return

addr:   SIXBIT/file-structure-name/
        directory-name
        status-bits
```

where: *length* specifies the length of the argument block.

addr points to the first word of the argument block.

file structure name (.DFJNM) can be a file structure name in SIXBIT, which will return the next file structure name after the specified file structure name; or it can be 0 which will return the first file structure name in the job's search list; or it can be -1 which will return the file structure name immediately following the FENCE (see section 8.2.2)

directory-name (.DFJDR) contains the directory name.

status bits specifies one of the status bits listed in Table 8-11.

Table 8-11
JOB/GOBSTR Status Bits

Bit	Mnemonic	Meaning
0	DF.SWL	Software write protection is in effect for this job.
1	DF.SNC	Files are not to be created on this file structure, when a multiple file structure name is specified in an INIT or OPEN monitor call. Files can be created if a specific file structure or physical unit is specified.

On a normal return, the first word of the argument list contains either

1. the first file structure name in the search list (if 0 was specified),
2. the next file structure name appearing after the specified name or after the FENCE (if -1 is specified),
3. 0 if the item after the specified name is FENCE, or
4. -1 is there are no more file structure names in the search list, or if the search list is empty.

The following is an example of reading a job's search list.

```
        SETOM   LOC             ;PLACE -1 IN LOC TO GET FIRST
                                ;NAME IN SEARCH LIST
LOOP:   MOVEI   AC,LOC          ;SET UP AC
        JOBSTR  AC,             ;PERFORM THE MONITOR CALL
          JRST  ERROR           ;ERROR RETURN LOCATION
```

```
        MOVE    AC,LOC          ;GET FILE STRUCTURE NAME RETURNED
        JUMPE   AC,FENCE        ;JUMP IF IT IS THE FENCE
        AOJE    AC,END          :JUMP IF END OF SEARCH LIST (-1)
        .                       ;LOC HAS NEXT FILE STRUCTURE NAME
        .
        .
        JRST    LOOP            ;REPEAT WITH NEXT FILE STRUCTURE NAME.
LOC:    -1                      ;FILE STRUCTURE NAME
        0                       ;RESERVED FOR FUTURE USE.
        0                       ;STATUS BITS
```

8.13 THE GOBSTR MONITOR CALL (CALLI 66)[1]

The GOBSTR monitor call returns successive file structure names in the job's or system's search list. Its calling sequence is

```
        MOVE    ac, [XWD length,addr]
        GOBSTR  ac,
          error return
        normal return

addr:   job-number
        XWD project-no.,programmer-no.
        SIXBIT/file-structure-name/
        directory-name
        status-bits
```

where: *length* is the length of the argument list.

addr points to the first word of the argument list.

job-number is the number of the job whose search list is desired; if the calling job's search list is desired. job number should be -1. If job number is 0, the specified project-programmer number is ignored and the system search list is used.

project-no.,programmer-no. is the project-programmer number under which the specified job number is running. If the project-programmer number is -1, the caller's project-programmer number is used.

file-structure-name can be a file-structure-name in SIXBIT which will return the search list beginning with the file structure name after the name specified; it can be -1 which will return the search list beginning with the first file structure name in the list; or it can be 0 which will return the search list beginning with the file structure name immediately following the FENCE.

directory-name is the directory name in which the file structure is found.

status bits which may be set are listed in Table 8-11.

On an error return, one of the error codes listed in Table 8-12 will be returned in the AC.

8.14 THE SYSSTR MONITOR CALL (CALLI 46)

The SYSSTR monitor call provides a mechanism to obtain all the file structure names in the system. To access all files in all User File Directories, the program should access the Master File Directory on each file structure separately. Monitor support programs use this call to access all of the files in the system. SYSSTR's calling sequence is

[1] GOBSTR is a privileged monitor call unless information being requested is either about the system search list or the jobs logged-in under the same project-programmer number as the calling job's number. The privilege bits required are either JP.SPA or JP.SPM in the privilege word (.GTPRV, refer to Chapter 18).

```
MOVEI    ac,n
SYSSTR   ac,
   error return
normal return
```

where: *n* is either 0 or the last value returned by a previous SYSSTR; it cannot be a physical disk unit name or a logical name.

Table 8-12
GOBSTR Error Codes

Error Code	Mnemonic	Meaning
3	DFGIF%	file-structure-name is not either 0,-1, or a file structure name in SIXBIT.
6	DFGPP%	The specified job number and project-programmer number do not correspond to each other.
10	DFGNP%	The calling job is not a privileged job.
12	DFGLN%	The specified length of the argument list is invalid.

The error return is taken if

1. the monitor call has not been implemented or
2. the argument is invalid.

On a normal return, the next file structure in the specified search list is returned in the AC. If there are no more file structures in the search list, -1 is returned in the AC. A return of 0 in the AC indicates that the list of file structure names has been exhausted. If 0 is specified as an argument, the first file structure name is returned in the AC.

8.15 THE SYSPHY MONITOR CALL (CALLI 51)

The SYSPHY monitor call returns all physical disk units in the system. The SYSPHY call is similar to the SYSSTR monitor call. SYSPHY's calling sequence is

```
MOVEI    ac,n
SYSPHY   ac,
   error return
normal return
```

where: *n* is 0 or the last unit name returned by the previous SYSPHY.

The first time a call to SYSPHY is made, the AC should contain 0 requesting that the monitor is to return the first physical unit name. On subsequent calls, the AC should contain the previously returned unit name.

The error return is taken if the AC does not contain a physical disk unit name or zero. On a normal return, the next physical unit name in the system is returned in the AC. If the monitor returns a 0 in the AC, the list of physical disk units has been exhausted.

8.16 THE DEVPPN MONITOR CALL (CALLI 55)

The DEVPPN monitor call allows a user program to obtain the project-programmer number associated with a disk device. DEVPPN's calling sequence is

```
{MOVE    ac, [SIXBIT/device-name/]}
{MOVEI   ac, channel              }
 DEVPPN  ac,
    error return
 normal return
```

where: *device-name* can be either a logical, a physical, or an ersatz device name.

channel may optionally be specified instead of device-name. It specifies the channel on which the specified device has been initialized.

If a legal device is specified, the normal return will be taken and the project-programmer number associated with the device will be returned in the AC. However, if the user has specified SYS: and has NEW: enabled in his search list, the project-programmer number returned will be [1,5].

The error return is taken under the following conditions:

1. The monitor call has not been implemented; therefore, the contents of the AC are unchanged. In this case, to obtain the appropriate project-programmer number follow one of the procedures listed below:
   a. Use the GETPPN monitor call for the user's area.
   b. Use the default project-programmer numbers appearing in Table 8-13 for the special ersatz devices.
2. If the device does not exist or the specified channel has not been initialized, 0 will be returned in the AC.
3. If the device specified is not a disk device, the user's project-programmer number will be returned in the AC.

Table 8-13
Ersatz Devices

Device Name	Project-Programmer Number	Device Name	Project-Programmer Number
ALG:	[5,4]	MUS:	[5,16]
ALL:	user's	MXI:	[5,3]
BAS:	[5,1]	NEL:	[5,20]
BLI:	[5,5]	NEW:	[1,5]
COB:	[5,2]	OLD:	[1,3]
CRP:	[10,1]	POP:	[5,22]
DEC:	[10,7]	PUB:	[1,6]
DMP:	[5,21]	REL:	[5,11]
DOC:	[5,14]	RNO:	[5,12]
DSK:	user's	SNO:	[5,13]
FAI:	[5,15]	SYS:	[1,4]
FOR:	[5,6]	TED:	[5,10]
HLP:	[2,5]	TST:	[5,23]
LIB:	set by user	UMD:	[6,6]
MAC:	[5,7]	UNV:	[5,17]
MFD:	[1,1]		

The following is an example for reading an UFD if either device SYS: or the user's area is specified.

```
        MOVEI   A,16            ;GET MFD PPN
        GETTAB  A,              ;NO CHANGE IF NO GETTAB
          MOVE  A,[1,1]         ; IN CASE OF LEVEL C
        MOVEM   A,MFDPPN        ;STORE MFD DIRECTORY NUMBER

        MOVE    A,DEVNAM        ;GET DEVICE NAME TYPED BY USER
        MOVEM   A,MODE+1        ;STORE FOR OPEN
        DEVPPN  A,
           JRST GETPPX          ;NOT IMPLEMENTED

;BACK HERE WITH IMPLIED PPN IN A

GOTPPN: MOVEM   A,PPN           ;STORE PPN IMPLIED BY DEVICE NAME

        OPEN    I,MODE          ; TRY TO OPEN DEVICE
           JRST ERROR           ;NOT AVAILABLE
        LOOKUP  I,PPN           ;TRY TO LOOKUP UFD
           JRST ERROR           ;NOT THERE
        IN                      ;READ FIRST BLOCK
           JRST USEIT           ;GO DO USEFUL WORK
        JRST    ERROR           ;ERROR OR END OF FILE

;HERE IF DEVPPN FAILS

GETPPX: CAMN    A,[SIXBIT/SYS/] ;SEE IF DEVICE NAME SYS:
        JRST    GETPPS          ;YES–GO HANDLE SYS:
        GETPPN  A,              ;NO–GET OWN PPN
           JFCL                 ;(IN CASE OF JACCT)
        JRST    GOTPPN          ;OK–PROCEED AVOVE

GETPPS: MOVE    A,[1,,16]       ;FIND SYS: PPN
        GETTAB  A,              ;FROM MONITOR TABLES
          MOVE  A,[1,,1]        ;IN CASE OF LEVEL C
        JRST     GOTPPN         ;OK–PROCEED ABOVE

MODE:   14                      ;BINARY READ
        0                       ;DEVICE NAME
        0,,INBUFH               ;BUFFER HEADER

PPN:    0                       ;DIRECTORY NAME
        SIXBIT/UFD/             ;EXTENSION
        0

MFDPPN: 1,,1                    ;LOOKUP UFD IN MFD
```

8.17 THE DSKCHR MONITOR CALL (CALLI 45)

The DSKCHR monitor call provides disk characteristics, which are necessary information for allocating storage efficiently on different types of disk units. DSKCHR's calling sequence is

```
        MOVE    ac, [XWD length,addr]
        DSKCHR  ac,
           error return
        normal return

addr:   argument
```

where: *addr* points to the first word of the argument block, which is represented in Table 8-14.

length is the length of the argument block.

argument can be the name of a file structure (e.g., DSKA), a type of controller (e.g., DP), a controller name (e.g., DPA), a logical unit name (e.g., DSKA3), a physical unit name (e.g., DPA3), or a logical device name (e.g., ALPHA). If more than one unit is specified, the values returned in the AC are for the first unit specified. If more than one file structure (i.e., DSK of logical device name disk) is specified, the first unit of the first file structure is returned in the AC.

On a normal return, the monitor returns the information listed in Table 8-14 in the argument block. All but the first word are set up by the monitor; the user sets up the first word. Status information (see Table 7-2) is returned in the AC.

Table 8-14
DSKCHR Argument Block

Location	Mnemonic	Meaning
addr	.DCNAM	The argument name in SIXBIT that the user program supplies.
addr+1	.DCUFT	The number of blocks left of the logged-in job's quota before the User File Directory of the job is exhausted on the unit specified in addr. If .DCUFT contains a negative value, the user file directory contains 400000,000000), the user file directory has not been accessed since LOGIN-time; therefore, the monitor is not aware of the user's quota.
addr+2	.DCFCT	The number of blocks on a first-come-first-serve basis left for all users on the specified file structure.
addr+3	.DCUNT	The number of blocks left for all users on the specified unit.
addr+4	.DCSNM	The file structure name to which this unit belongs.
addr+5	.DCUCH	The characteristic sizes as follows bits — mnemonic — meaning 0-8 — DC.UCC — blocks per cluster 9-17 — DC.UCT — blocks per track 18-35 — DC.UCY — blocks per cylinder
addr+6	.DCUSZ	The number of 128-word blocks on the specified unit.
addr+7	.DCSMT	The mount count for the file structure, which is the number of jobs that have performed a MOUNT command for this file structure without executing a DISMOUNT command. (LOGIN performs an implied MOUNT of DSK: for all logged-in users.)
addr+10	.DCWPS	The number of words containing data bits per SAT block on this unit.
addr+11	.DCSPU	The number of SAT blocks per unit.
addr+12	.DCK4S	The number of K allocated for swapping.
addr+13	.DCSAJ	The file structure mount word, which will contain either 0,0 if more than one job has this file structure mounted. -1,,n if only job has this file structure mounted, but the file structure is not single-access. 0,,n if job has this file structure mounted, and the file structure is single-access.
addr+14	.DCULN	The unit's logical name.

Table 8-14 (Cont.)
DSKCHR Argument Block

Location	Mnemonic	Meaning
addr+15	.DCUPN	The unit's physical name.
addr+16	.DCUID	The unit's ID.
addr+17	.DCUFS	The first logical block used for swapping on this unit.
addr+20	.DCBUM	The number of blocks per unit (including maintenance cylinders).
addr+21	.DCCYL	The current cylinder number.
addr+22	.DCBUC	The number of blocks per unit in PDP-11 compatibility mode.
addr+23	.DCLPQ	The length of the position wait queue.
addr+24	.DCLTQ	The length of the transfer wait queue.

Table 8-15
DSKCHR Status Bits

Bit	Mnemonic	Meaning
0	DC.RHB	The monitor must reread the home block before the next operation, to ensure that the pack ID is correct. The monitor will set this bit when a disk pack goes off-line.
1	DC.OFL	The unit is off-line.
2	DC.HWP	The unit is hardware write-protected.
3	DC.SWP	The unit belongs to a file structure that is write-protected by the software for this job.
4	DC.SAF	The unit belongs to a single-access file structure.
5	DC.ZMT	The unit belongs to a file structure with a mount count that has gone to zero (i.e., no one is using the file structure).
6		Reserved.
7-8	DC.STS	The unit status, either code — mnemonic — meaning 0 — .DCSTP — A pack is mounted. 2 — .DCSTN — No pack is mounted.
3	DC.STD	The unit is down.
9	DC.MSB	The unit has more than one DAT block.
10	DC.NNA	The unit belongs to a file structure for which the operator has requested no new INITs, LOOKUPs, ENTERs, or FILOP.'s; set by a privileged STRUUO function, see the STRUUO Specification in the DECsystem-10 Software Notebooks.
11	DC.AWL	The file structure is write-locked for all jobs.
12-14		Reserved.
15-17	DC.TYP	The type of argument passed to monitor in addr, either

Table 8-15 (Cont.)
DSKCHR Status Bits

Bit	Mnemonic	Meaning
		code / mnemonic / meaning 0 / .DCTDS / a generic name 1 / .DCTAB / a subset of file structures because of file structure abbreviation 2 / .DCTFS / a file structure name 3 / .DCTUF / a unit within a file structure 4 / .DCTCN / controller class name 5 / .DCTCC / controller class 6 / .DCTPU / physical unit
18-20	DC.DCN	The data channel number that the software believes the hardware is connected to; the first data channel is 0.
21-26	DC.CNT	The controller type: code / mnemonic / meaning 1 / .DCCFH / RC10 controller 2 / .DCCDP / RP10 controller
27-29	DC.CNN	The controller number; first controller of each type starts with 0.
30-32	DC.UNT	The unit type; a controller-dependent field used to distinguish various options of a unit on its controller. code / mnemonic / meaning 0 / .DCUFD / RD10 Burroughs disk (if bits 21-26=1). 1 / .DCUFM / RM10B Bryant disk (if bits 21-16=1). 2 / .DCUD2 / RP02 disk pack (if bits 21-26=2). 3 / .DCUD3 / RP03 disk pack (if bits 21-26=2).
33-35	DC.UNN	The physical unit number within the controller; first unit number is 0.

8.18 THE DISK. MONITOR CALL (CALLI 121)

The DISK. monitor call sets and examines the parameters associated with disk and file systems. It allows the user to assign a priority level for disk operations (e.g., data transfers and head positionings) either for a user I/O channel or for a job (i.e., all user channels). When this priority level has been set, the disk operation request with the highest priority level, the request most satisfying disk optimization is chosen. The DISK. monitor call's calling sequence is

```
MOVE   ac, [XWD function,addr]
DISK.  ac,
   error return
normal return
```

where: *function* is one of the function codes listed in Table 8-16.

addr is appplicable only when specifying function code 0; it points to a word that contains the priority level desired and the channel(s) for which this priority level is to be associated with.

Table 8-16
DISK. Function Codes

Function Code	Mnemonic	Meaning
0	.DUPRI	Set the priority level to that specified at addr, refer to Figure 8-8.
1	.DUSEM[1]	Set the PDP-11 (i.e., 22-sector) mode on the RP04.
2	.DUSTM[1]	Set the PDP-10 (i.e., 20-sector) mode on the RP04.
3	.DUUNL[1]	Unload the RP04.
4	.DUOLS[1]	The specified channel/controller will be set off-line.
5	.DUOLN[1]	The specified channel/controller is off-line now.
6	.DUONL[1]	The specified channel/controller is back on-line.

[1]This is a privileged function.

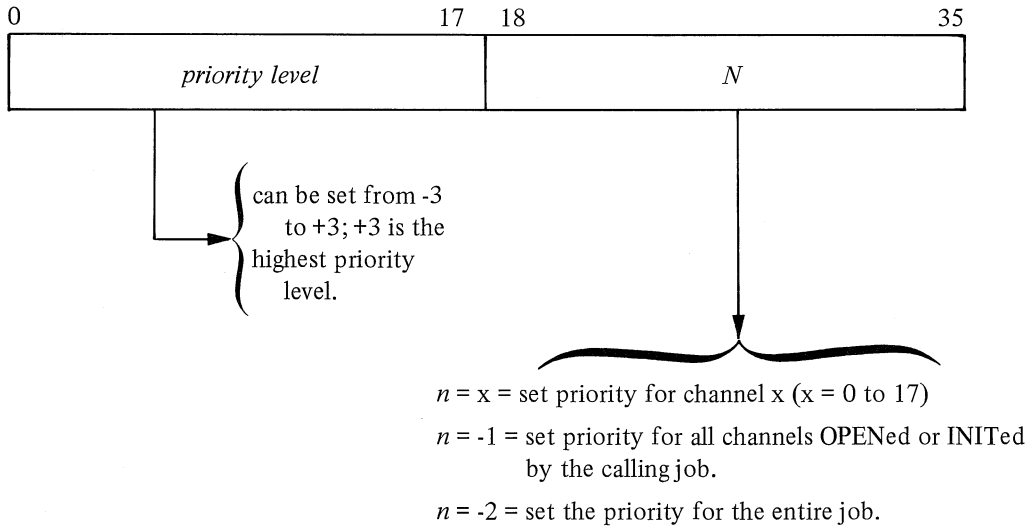


Figure 8-9. DISK. Priority Level

The range of priority levels may be specified from -3 to +3, with 0 being the normal timesharing level of priority. A job can request a priority that is lower than the normal priority; this lower priority would be used when the associated job is a background job.

The maximum priority level that a job may assume is set in bits 1 and 2 (JP.PRI) of the privilege word, .GTPRV.

When a priority level is set for a channel, that priority level overrides any other priority level set for any job associated with that channel. That priority level remains in effect until it is changed or until that channel is RELEASEd.

When a priority level is set for a job, that priority level remains in effect until reset by another DISK. monitor call or until the DSKPRI command is executed, refer to *DECsystem-10 Operating System Commands.*

The possible error codes that may be returned in the AC are listed in Table 8-17.

Table 8-17
DISK. Error Codes

Code	Mnemonic	Meaning
0		DISK. monitor call has not been implemented.
-1	DUILF%	Illegal function has been specified.
-2	DUILP%	An illegal priority level has been specified.

8.19 FILE STATUS

This section describes the file status word, its initial setting, and how a user can examine, test, and change it.

The status of a file, including its data mode, is set initially to the contents of the file status word. After these bits are initially set by the monitor, the user can test and reset the bits by using the STATZ, STATO, and SETSTS monitor calls. Bits 30-35 of the file status word are normally set by the monitor on the execution of an INIT, OPEN, or FILOP. monitor call. If the indicated data mode is not legal for the specified device, the monitor will stop the job and print the following message on the user terminal:

?ILLEGAL DEVICE DATA MODE FOR DEVICE *devicename* AT USER *addr*

where: *devicename* is the physical name of the device.
addr is the location of the INIT, OPEN or FILOP. monitor call.

The right half of the file status word (i.e., bits 18-35) reflects the current state of a file transmission. Table 8-2 lists the file status bits for disk files; refer to Chapters 9, 10, 11, and 12 for the file status bits for files on other devices.

All bits, except the end-of-file bit, are set by the monitor as a condition occurs, rather than being associated with the buffer currently used. However, the file status bits are stored with each buffer, enabling the user to determine which buffer produced the error.

The end-of-file bit is set when the user attempts to read past the last block of data. Therefore, when this bit is set, no data has been placed in the input buffer.

The file status can be returned to the user via the GETSTS monitor call; it can be tested via the STATZ monitor call, or the STATO monitor call; and the file status may be changed via the SETSTS monitor call.

8.19.1 The GETSTS Monitor Call (Op Code 62)

The GETSTS monitor call returns the file status bits associated with a specified device, its calling sequence is

GETSTS *channel,addr*
only return

where: *channel* is the channel number associated with the desired device.
addr is the location in which the file status bits will be stored on a return (i.e., 0,,*bits*)

If a device has not been associated with the specified data channel, the monitor will stop the job and print the following message on the user's terminal:

?I/O TO UNASSIGNED CHANNEL AT USER *addr*

Table 8-18
Disk File Status Bits

Bit	Mnemonic	Meaning
18	IO.IMP	Improper Mode. Attempt to write on a software write-locked file structure, or a software redundancy failure occurred. This bit is usually set by the monitor. The user cannot set this bit.
19	IO.DER	Hardware device error. The disk unit is in error, rather than the data on the disk. However, data read into core or written on the disk is probably incorrect. The user does not usually set this bit.
20	IO.DTE	Hard data error. The data read or written has incorrect parity as detected by the hardware. The user's data is probably unrecoverable even after the device has been fixed. This bit is usually not set by the user.
21	IO.BKT	Block too large. A disk data block is too large to fit into the buffer; or a block number is too large for the disk unit; or DSK has been filled; or the user's quota on the file structure has been exceeded. This bit is usually not set by the user. This error is also returned when the user tries to close a file that has open locks associated with it (via Enqueue/Dequeue).
22	IO.EOF	End-of-file. The user program has requested data beyond the last block of the file with an IN or INPUT call; or USETI has specified a block beyond the last data block of the file. When IO.EOF is set, no data has been read into the buffer. This bit is usually not set by the user.
23	IO.ACT	I/O Active. The disk is actively transmitting or receiving data. This bit is always set by the monitor.
29	IO.WHD	Write disk pack headers. This is used in conjunction with the SUSET monitor call to format a disk pack.
30	IO.SYN	Synchronous mode I/O. Stop disk after every buffer is read or written.
31	IO.UWC	User word count, supplied by the user in each buffer.
32-35	IO.MOD	Data mode of the device.

8.19.2 STATO/STATZ (Op Codes 61 and 63)

The STATO/STATZ calls test the bits in the file status word. STATO will skip if any file status bits specified are 1; STATZ will skip if all file status bits specified are 0. Their calling sequences are:

STATO *channel,bits*	STATZ *channel,bits*
normal return	*normal return*
skip return	*skip return*

where: *channel* is the channel number associated with the desired device.
bits are the bits in the file status word that are to be tested.

If a device has not been associated with the specified data channel, the monitor will stop the job and print the following message on the user's terminal:

?I/O TO UNASSIGNED CHANNEL AT USER *addr*

8.19.3 The SETSTS Monitor Call (Op Code 060)

The SETSTS monitor call allows a user to change the bit setting in the file status word. Its calling sequence is:

SETSTS *channel,bits*
return

where: *channel* is the channel number associated with the desired device.
bits specifies those bits which are to be changed in the file status word.

If no device has been associated with the specified channel number, the monitor will stop the job and print the following message on the user's terminal:

?I/O TO UNASSIGNED CHANNEL AT USER *addr*

If the user is attempting error recovery, bits 18, 19, 20, and 21 must be cleared by this call. SETSTS can be called to clear the end-of-file bit (bit 22, IO.EOF); but this alone is not enough to clear the end-of-file condition. Further input to the file cannot take place until the end-of-file has been cleared by a CLOSE, USETI, MTAPE., TAPOP., LOOKUP, or INIT monitor call.

SETSTS will wait until the specified device is inactive before changing the file status bits. If a new data mode is specified that is not legal for the device, the monitor will stop the job and print the following message on the user's terminal:

?ILL DEVICE DATA MODE FOR DEVICE *devicename* AT USER *addr*

If a user changes the data mode (bits 32-35), he will also have to change the byte size for the byte pointer in the input buffer header (if any) and the byte size and item count in the output buffer header (if any). The output item count should be changed by using *current count* and dividing/multiplying by the appropriate conversion factor (rather than assuming the length of the buffer). If the data mode change requires a different buffer size, but the size is not changed, incorrect I/O will probably result. The mode specified in an INIT call determines the buffer size, even though the buffer ring has not yet been created.

8.20 TERMINATE A FILE

The CLOSE monitor call (op code 070) terminates file transmission. Its calling sequence is

CLOSE *channel,option*
return

where: *channel* is the channel number on which the specified file is to be closed.
option is usually zero, but individual options may be selected independently to control the effect of the CLOSE. The possible options are:

Option	Meaning
0	The output side of the specified channel is closed (bit 35 = 0). In unbuffered data modes, the effect is to execute a device-dependent function. In buffered data modes, the following operations are performed: 1. All data in the buffers that has not been transmitted to the device is written. 2. Device dependent functions are performed. 3. The ring use bit (bit 0 of the first word in the buffer header) is set to a 1 indicating that the buffer ring is available. 4. The buffer byte count (the third word in the buffer header) is set to 0. 5. Control returns to the user program when transmission is complete. The input side of the specified channel is also closed (bit 34 = 0). The end-of-file flag is always cleared. Further action depends on the data mode in unbuffered data modes. The effect is to execute a device dependent function. In buffered data modes, if a ring buffer exists, the following operations are performed: 1. Wait until the device is inactive. 2. The use bit of each buffer is cleared indicating that the buffer is empty. 3. The ring use bit of the buffer header is set to 1 indicating that the buffer ring is available. 4. The buffer byte count is set to 0. 5. Control returns to the user program. On output CLOSE, the unwritten blocks at the end of the disk file are automatically de-allocated (bit 33 = 0). On input CLOSE, the access date of a disk file is updated (bit 32 = 0).
1	The closing of the output side of the specified channel is inhibited. Other actions of CLOSE are unaffected. (Bit 35 = 1, CL.OUT).
2	The closing of the input side of the specified channel is inhibited; other actions of CLOSE are unaffected. (Bit 34 = 1, CL.IN).
4[1]	The unwritten blocks at the end of a disk file are not deallocated. This capability is provided for users who specifically allocate disk space and wish to retain it. (Bit 33 = 1, CD.DLL).

[1]Use of this option is meaningful only with disk files, and it is ignored with non-disk files.

Option	Meaning
10[1]	The updating of the access date on CLOSE input is inhibited (bit 32=1, CL.ACS). This capability is intended for use with FAILSAFE, so that files can be saved on magnetic tape without causing the disk copy to appear as if it had been accessed.
20[1]	The deleting of the NAME block and the access tables in monitor core on CLOSE input is inhibited, if a LOOKUP was done without a subsequent INPUT (bit 31=1, CL.NMB).
40[1]	The deleting of the original file, if any, is inhibited, if an ENTER which creates or supersedes was done (bit 30=1, CL.RST). The new copy of the file is discarded.
100[1]	The NAME block and access tables are deleted from the disk data base, and the space is returned to free core (bit 29=1, CL.DAT).

[1] Use of this option is meaningful only with disk files, and it is ignored with non-disk files.

CHAPTER 9
I/O PROGRAMMING FOR DECTAPE

This chapter explains the unique features of I/O programming using a DECtape unit. DECtape devices accept the calls described in Chapter 7, unless indicated otherwise. Table 9-1 summarizes the characteristics of a DECtape unit. Buffer sizes are given in octal and include 3 bookkeeping words. The physical characteristics of a device may be obtained by issuing the DEVCHR call (refer to paragraph 7.8.2).

Table 9-1
DECtape Devices

Device	Physical Name	Controller Number	Unit Number	Special Monitor Calls	Data Modes for DECtape	Buffer Sizes (Octal)
DECtape	DTA0, DTA1, ...,DTA7 DTB0,DTB1, ...,DTB8	TD10 551 (PDP-6)	TU55 555 (PDP-6)	MTAPE, UGETF, USET0, USET1, UTPCLR	Ascii Ascii Line Image Binary Image Binary Dump Record Dump	202

9.1 DECTAPE

The device mnemonics for DECtape devices are:

DTA0
DTA1
DTA2
DTA3
.
.
.
DTA7

The buffer size is 202 octal words: 177 octal user data and 200 octal transferred. On systems with dual DECtape controllers, the drives on the second controller have the mnemonics DTB0, ...,DTB7 where the B indicates the second controller.

9.2 DATA MODES

Two hundred words are written. The first word is the link plus word count. The following 177 octal words are data supplied to and from the user program.

9.2.1 Buffered Data Modes

Data is written on DECtape exactly as it appears in the buffer and it consists of 36-bit words. No processing or checksumming of any kind is performed by the service routine. The self-checking of the DECtape system is sufficient assurance that the data is correct. Refer to paragraph 9.1.2 for further information concerning the blocking of information.

9.2.2 Unbuffered Data Modes

Data is read or written from anywhere in the user's core area without regard to the standard buffering scheme. Control for read or write operations must be via a command list in core memory. The command list format is described in paragraph 7.3.1. On the KI10/KL10, if the IOWD list is modified as the result of I/O performed (i.e., an INPUT reads into the IOWD list) and the word count of any of the IOWDs read into the list is greater than the following value:

(*maximum word count specified in original list* – 2) / 512+2

the monitor will stop the job and print the following message on the user's terminal:

?ADDRESS CHECK AT USER *addr*

File-structured dump mode data is automatically blocked into standard-length DECtape blocks by the DECtape service routine. Each block read or written contains link word plus 1 to 177 octal data words. Unless the number of data words is an exact multiple of the data portion of a DECtape block (177 octal), the remainder of the last block written after each OUTPUT call is wasted. The INPUT call must specify the same number of words that the corresponding OUTPUT call specified to skip over the wasted fractions of blocks.

9.3 DECTAPE FORMAT

A standard reel of DECtape consists of 1102 octal pre-recorded blocks each capable of storing 200 octal 36-bit words of data. Block numbers that label the blocks for addressing purposes are recorded between blocks. These block numbers run from 0 to 1101 octal. Blocks 0, 1, and 2 are normally not used during timesharing and are reserved for a bootstrap loader. Block 100_{10} is the directory block, which contains the names of all files on the tape and information relating to each file. Blocks 1-143 octal and 145-1101 octal are usable for data.

If, in the process of DECtape I/O, the I/O service routine is requested to use a block number larger than 1101 octal or smaller than 0, the monitor will set the IO.BKT bit in the file status word (refer to paragraph 7.6.1).

9.3.1 DECtape Directory Format

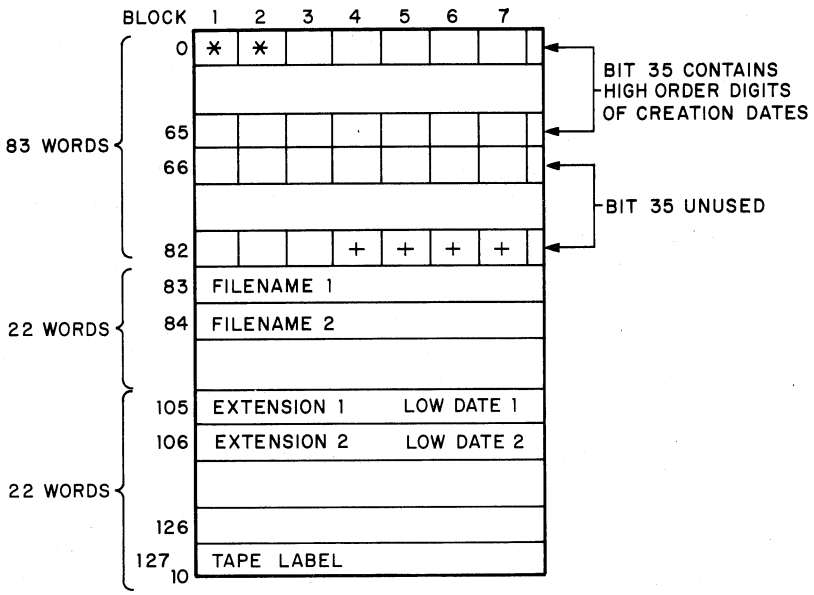
The directory block (block 144 octal) of a DECtape contains directory information for all files on that tape; a maximum of 22 files can be stored on any one DECtape (refer to Figure 9-1).

Words 0 through 83_{10} of the directory block contain slots for blocks 1 through 577 on a DECtape. Each slot occupies five bits (seven slots per word) and represents a given block on the DECtape. Each slot contains the number of the file (1-26 octal) occupying the given block. This allows for 581 slots (83 words x 7 slots per word). The four extra slots represent non-existent blocks 1102 octal through 1105 octal.

Bit 35 of words $0\text{-}65_{10}$ of the directory block contains the high order three bits of the 15-bit creation date of each file on the DECtape. (Note that the low order 12 bits of the creation date of each file are contained in words 105 through 126_{10}. This split format allows for compatibility among monitors and media as old as 1964. The high order 3 bits of the 15-bit creation data for file 1 are contained in bit 35 of words 0, 22, and 44. Word 44 contains the first (i.e., most significant) bit; word 22 contains the second and word 0 contains the third. The high order digits for file 2 are contained in bit 35 of words 1, 23, and 45 with the digits in the same order as described for file 1. The high order digits for the remaining files are organized in the same fashion.

Words 83_{10} through word 104_{10} of the directory block contain the file names of the 22 files that reside on the DECtape. Word 83 contains the file name for file 1, word 84 contains the file name for file 2; file names are stored in SIXBIT.

Words 105_{10} through 126_{10} contain the file name extensions and the low order part of the creation date of the 22 files that reside on the DECtape, in the same relative order as their file names. The bits for each word are described in Table 9-2.



NOTES:

* Reserved for system, contains 36 as does block 144_8 for the directory.

\+ Represents blocks 1102 through 1105, which are not available contains 37_8.

10-0572

Figure 9-1. DECtape Directory Format

Table 9-2
Format of Words 105 - 126 in the DECtape Directory Block

Bit	Meaning
0 - 17	The file name extension in SIXBIT.
18 - 23	Zero.
24 - 35	The low-order 12 bits of the creation date. (Note that the high order digits are encoded in bit 35 of words 0 through 65_{10}.) The creation date is computed using the following formula: $((((year-1964)\text{x}12)+(month-1)\text{x}31+(day-1)))=date$

Word 127_{10} of the directory block is the tape label.

The message

?BAD DIRECTORY FOR DEVICE DTA*n*:EXEC CALLED FROM USER LOC *addr*

is typed when one of the following conditions is detected:

1. A parity error occurred while reading the directory block.
2. No slots are assigned to the file number of the file.
3. The tape block, which may be the first block of the file (i.e., the first block for the file encountered while searching backwards from the directory block), cannot be read.

User programs should never manipulate DECtape directories explicitly since the LOOKUP and ENTER calls (refer to paragraphs 7.5.1.1 and 7.5.1.2) automatically record all of the necessary entries in the directory for the user. These calls have all of the capabilities needed to process the name and the creation date of a file. However, a small number of special purpose programs do process directories by explicit action, rather than using the LOOKUP and ENTER calls. For such programs, the following examples illustrate methods for

1. assembling the 15-bit creation date, and
2. storing the 15-bit creation date. The number of the file (1 to 22) is in register P1 and the directory block begins at the location DIRECT.

The example below shows the special assembly for the creation date.

```
DPB      T1,[POINT 12,DIRECT+-D104(P1),35]   ;SAVE LOW PART
MOVEI    T2,1                                ;SET UP TO MARK LOW BIT
ANDCAM   T2,DIRECT-1(P1)                     ;CLEAR DIRECTORY BIT
TRNE     T1,1B23                             ;IF BIT IN DATE SET,
IORM     T2,DIRECT-1(P1)                     ;SET DIRECTORY BIT
ANDCAM   T2,DIRECT+-D21(P1)                  ;
TRNE     T1,1B22                             ;REPEAT FOR EACH BIT IN
IORM     T2,DIRECT+-D21(P1)                  ;HIGH PART OF DATE
ANDCAM   T2,DIRECT+-D43(P1)                  ;
TRNE     T1,1B21                             ;
IORM     T2,DIRECT+-D43(P1)                  ;
```

The example below shows special purpose storage of the creation date.

```
LDB      T1,[POINT 12,DIRECT+-D104(P1),35]   ;GET LOW PART
MOVEI    T2,1                                ;SET UP TO TEST LOW BIT
TDNE     T2,DIRECT-1(P1)                     ;IF SET IN DIRECTORY
TRO      T1,1B23                             ;THEN SET BIT IN DATE
TDNE     T2,DIRECT+-D21(P1)                  ;REPEAT FOR EACH BIT IN
TRO      T1,1B22                             ;HIGH PART OF DATE
TDNE     T2,DIRECT+-D43(P1)                  ;
TRO      T1,1B21                             ;
```

9.3.2 DECtape File Format

A file consists of any number of DECtape blocks. Figure 9-2 represents the format of a file on a DECtape.

Each block contains the following information:

Word 0	Left Half:	The link, which is the block number of the next block in the file. If the link is zero, this block is the last block in the file.

Right Half:	Bits 18 through 27 contain the block number of the first block in the file. Bits 28 through 35 contain a count of the number of words in this block that are used (maximum is 177 octal).
Words 1 - 177 octal	The data packed exactly as the user placed it in this buffer, or in dump mode files, the next 177 words of memory.

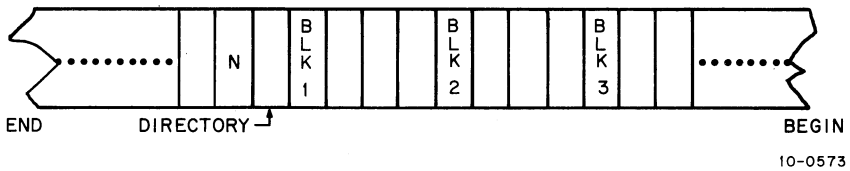


Figure 9-2. Format of a File on a DECtape

Figure 9-3 illustrates the format of a DECtape block.

link	1st block number	word count
data		

Figure 9-3. Format of a DECtape Block

9.3.3 Block Allocation

Normally, blocks are allocated by starting with the first free block nearest the directory and going backwards to the front of the tape (block 0). When the end of the tape is reached, the direction of the scan is reversed. Blocks are not written contiguously; rather, they are separated by a spacing factor. This allows the drive to stop and restart to read the next block of the file without having to back up the tape. The spacing factor is normally four, but for dump mode and UGETF followed by an ENTER, the spacing factor is two.

9.4 I/O PROGRAMMING

DECtape is a directory device; therefore, file selection must be performed by the user before data can be transmitted to or from the file. File selection is accomplished via the LOOKUP, ENTER, or FILOP. calls. Refer to paragraph 7.5.2 for a description of FILOP.

9.4.1 The Lookup Operator

On DECtape the calling sequence for LOOKUP is

```
LOOKUP   channel,addr
   error return
normal return
```

where: *channel* specifies the software I/O channel associated with the device on which the file resides.
addr points to a four-word argument block with the format illustrated in Figure 9-4.

A detailed description of the argument block is given in Table 9-3.

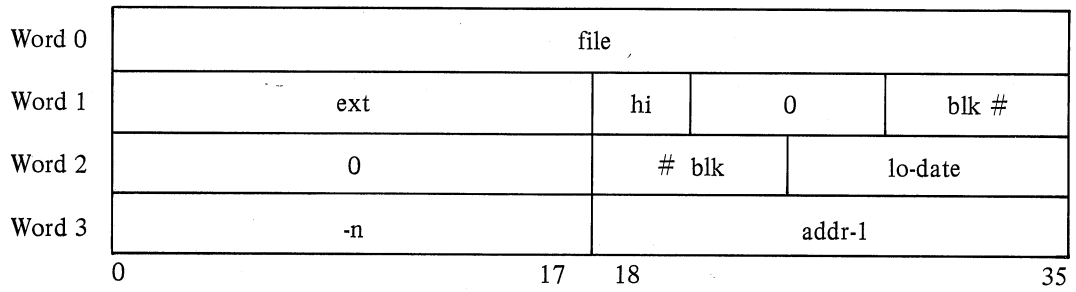


Figure 9-4. LOOKUP/ENTER/RENAME Argument Block

Table 9-3
LOOKUP Parameters

On Call				On Return			
Word	**Bits**	**Use**	**Contents**	**Word**	**Bits**	**Use**	**Contents**
0	0-35	A	The file name in SIXBIT.	0	0-35	V	The file name in SIXBIT.
1	0-17	A	The file name extension in SIXBIT.	1	0-17	V	The file name extension in SIXBIT.
1	18-20	I		1	18-20	V	The high order 3 bits of the creation date.
1	21-25	I		1	21-25	V	Zero.
1	26-35	I		1	26-35	V	The first block number.
2	0-35	I		2	0-17	V	Zero.
				2	18-23	V	Zero.
				2	24-35	V	The low order 12 bits of the creation date.
3	0-35	I		3	0-17	V	The negative word length of the zero-compressed file.
				3	18-35	V	The core address of the first word of the file minus 1.

A = argument for user program, V = value returned from monitor, I = ignored.

LOOKUP sets up an input file on the specified channel. The contents of the argument block are matched against the filenames and file names extensions in the DECtape directory. (Words 1 and 2) if no match is found, the error return is taken, and an error code is returned in the RH of word 1, refer to Appendix E. If a match is found, the 4-word argument is filled in by the monitor, and the normal return is taken.

On a normal return, the first block of the file is as follows:

1. The first 83 words of the DECtape are searched backwards, beginning with the slot immediately before the directory block, until the slot containing the desired file number is found.

2. The block associated with this slot is read in and bits 18-27 of the first word of the block are checked (i.e., the bits containing the block number of the first block of the file). If the bits are equal to the block number of this block, then this block is the first block; if not, then the block with that block number is read as the first block of the file.

9.4.2 The ENTER Operator

The ENTER operator sets up an output file on the specified channel, its calling sequence is

ENTER *channel, addr*
error return
normal return

where: *channel* specifies the software I/O channel associated with the device containing the desired file.

addr points to a four-word argument block (with two 0 words) shown in Figure 10-4 (word 0 and 1).

Table 9-4 describes the ENTER parameters needed for the call and those returned from the monitor on a normal return from the call.

Table 9-4
ENTER Parameters

On Call				On Return			
Word	**Bits**	**Use**	**Contents**	**Word**	**Bits**	**Use**	**Contents**
0	0-35	A	The file name in SIXBIT.	0	0-35	V	The file name in SIXBIT.
1	0-17	A	The file name extension in SIXBIT.	1	0-17	V	The file name extension in SIXBIT.
1	18-35	I		1	18-20	V	The high order 3 bits of the creation date.
				1	21-35	I	
2	0-35	I		2	0-35	I	

A = argument from user program, V = values returned from monitor, I = ignored.

The DECtape directory is searched for a file name and a file name extension matching the argument supplied in word 0 and word 1 (LH) of the argument block. If no match is found, and there is room in the directory, the monitor records the information in the first three words of the DECtape directory. If a match is found, the new entry replaces the old entry, the old file is reclaimed immediately, and the monitor records the file information. This process is called superseding and differs from the process on the disk, in that because of the small size of DECtapes, the space is reclaimed before the file is written rather than after.

9.4.3 The RENAME Operator

The RENAME operator alters the file name or the file name extension of an existing file, or deletes the file directory from the DECtape associated with the specified channel. The calling sequence for RENAME is

RENAME *channel, addr*
error return
normal return

Refer to Figure 9-4 for a description of the argument block; refer to Table 9-5 for a detailed description of the RENAME parameters.

Table 9-5
RENAME Parameters

On Call			On Return		
Parameter	**Use**	**Contents**	**Parameter**	**Use**	**Contents**
word 0	A	SIXBIT/FILE/ or 0	E	V	SIXBIT/FILE/
word 1	A	LH = SIXBIT/EXT/ RH = high order 3 bits of 15-bit creation date (bits 18-20).	E + 1	V	LH = SIXBIT/EXT/ RH = high order 3 bits of 15-bit creation date (bits 18-20).
word 2	A	RH = low order 12 bits of 15-bit creation date or 0 (0 implies current date).	E + 2	V	RH = low order 12 bits of 15-bit creation date (bits 24-35).
word 3	I		E + 3	I	
A = argument from user program, V = value from monitor, I = ignored.					

Unlike on a disk RENAME, a DECtape RENAME works on the last file LOOKUPed or ENTERed for the device, not the last file for the specified channel. The calling sequence required to successfully RENAME a file on DECtape is as follows:

LOOKUP *channel, addr*
CLOSE *channel,*
RENAME *channel, addr 1*

or

ENTER *channel, addr*
CLOSE *channel,*
RENAME *channel, addr 1*

9.4.4 INPUT, OUTPUT, CLOSE, RELEASE

When performing input operations, the DECtape service routine reads the links in each block to determine what block to read next and when to raise the EOF flag.

When an OUTPUT is given, the DECtape service routine examines the left half of the third word in the output buffer (the word containing the word count in its right half). If this word contains -1, it is replaced with a 0 before being written out, and the file is terminated. If this half word is greater than 0, it is not changed and the service routine uses it as the block number for the next OUTPUT. If this half word is 0, the DECtape service routine assigns the block number of the next block for the next OUTPUT.

For both INPUT and OUTPUT, block 100 decimal (the directory block) is treated as an exceptional case. If the user program issues

USETI *channel,* -D 100

to read block 100 decimal, it is treated as a 1-block file.

The CLOSE operator places a -1 in the left half of the third word in the last output buffer, thus terminating the file.

The RELEASE operator writes the copy of the directory, which is normally kept in core onto block 100 decimal, but only if any changes have been made. Certain console commands, such as KJOB or CORE 0, perform an implicit RELEASE of all devices and, thus, write out a changed directory even though the user's program failed to give a RELEASE.

9.5 SPECIAL MONITOR CALLS

Several monitor calls are provided for manipulating DECtape. These calls allow the user to manipulate block numbers and to handle directories.

9.5.1 USETI Channel, Addr

USETI sets the tape on channel to input block *addr* next. Since the monitor reads as many buffers as it can on input, it is difficult to determine which buffer the monitor is processing when the USETI is given. Therefore, the INPUT following the USETI may not obtain the buffer containing the block specified. However, if a single buffer ring is used, the desired block is retrieved since the device must stop after each INPUT. Alternatively, if bit 30 (IO.SYN) of the file status word is set via an INIT, OPEN, or SETSTS monitor call, the device stops after each bufferful of data on an INPUT so that the USETI will apply to the buffer supplied by the next INPUT.

9.5.2 USETO Channel, Addr

USETO sets the DECtape on channel to output block addr next.

9.5.3 UGETF Channel, Addr

UGETF places the number of the next free block of the file in addr.

If UGETF is not preceded by an ENTER, the monitor modifies its algorithm in the following manner:

1. The first block is written nearest the front of the tape instead of nearest the directory.
2. The spacing factor is changed to 2 instead of 4 so that very large programs can fit almost entirely in a forward direction.

If no LOOKUP or ENTER has been done, UGETF returns a -1. If a LOOKUP has been done, UGETF gives the same results as if an ENTER has been done. UGETF returns a block number; it neither marks the directory nor sets a particular block to be written, and is a no-op for any device except DECtape.

9.5.4 UTPCLR AC, (CALLI 13)

UTPCLR clears the directory of the DECtape on the device channel specified in the AC field. A cleared directory has zeroes in the first 83 words except in the slots related to blocks 1, 2, and 100 decimal and nonexistent blocks 1102 through 1105 octal. Only the directory block is affected by UTPCLR. This programmed operator is a no-op if the device on the channel is not a DECtape.

9.5.5 MTAPE *Channel,* 1 And MTAPE *Channel,* 11

MTAPE *channel*, 1 (MTREW.) rewinds the DECtape and moves it into the end zone at the front of the tape. MTAPE channel, 11 (MTUNL.) rewinds and unloads the tape, pulling the tape completely onto the left-hand reel, and clears the directory-in-core bit. These commands affect only the physical position of the tape, not the logical position. When either is used, the user's job can be swapped out while the DECtape is rewinding; however, the job cannot be swapped out if an INPUT or OUTPUT is done while the tape is rewinding.

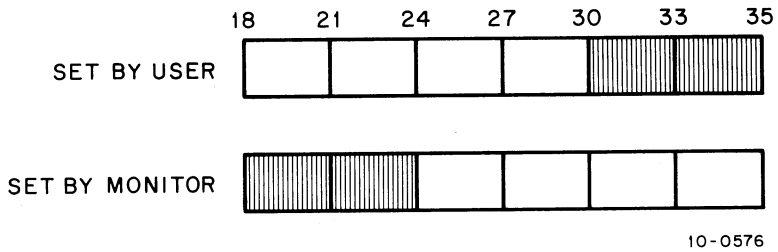
9.5.6 DEVSTS Monitor Call After Each Interrupt

The DECtape service routine stores the results of a CONI in the DEVSTS word of the device data block. The DEVSTS call is used to return the contents of the DEVSTS word to the user (refer to paragraph 7.8.1).

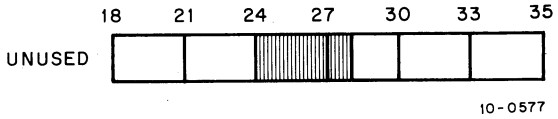
9.6 FILE STATUS

The file status of the DECtape is shown on the next page.

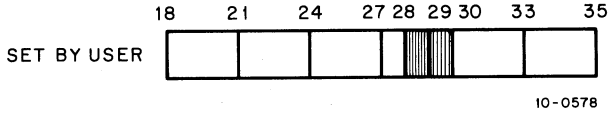
Standard Bits



Bit 18 = IO.IMP	An attempt was made to read block 0 in nonstandard dump mode.
Bit 19 = IO.DER	Data was missed.
Bit 20 = IO.DTE	Parity error.
Bit 21 = IO.BKT	Block number is too large or tape is full on OUTPUT.
Bit 22 = IO.EOF	EOF mark encountered on input. No special character appears in buffer.
Bit 23 = IO.ACT	Device is active.



Device Dependent Bits



Bit 28 = IO.SSD

DECtape is in semi-standard I/O mode. The setting of this bit is recognized only if bit 29 (nonstandard I/O mode) is on. Semi-standard mode is similar to nonstandard mode except

1. Block numbers are checked for legality, and
2. The tape is started in the same direction as it was previously going.
3. Dead reckoning is done.

Bit 29 = IO.NSD

DECtape is in a nonstandard I/O mode format as opposed to standard-I/O mode. No file-structured operations are performed on the tape. Blocks are read or written sequentially; no links are generated (output) or recognized (input). The first block to be read or written must be set by a USETI or USETO. In nonstandard I/O mode, up to 200 octal words per block are read or written as user data (as opposed to the standard mode of 1 link plus word count followed by 177 octal words). No dead reckoning is used on a search for a block number as the tape may be composed of blocks shorter than 200 words. The ENTER, LOOKUP, and UPTCLR calls are treated as no-ops. Block 0 of the tape cannot be read or written in dump mode if bit 29 is on, because the data must be read in a forward direction and block 0 normally cannot be read forward.

9.7 IMPORTANT CONSIDERATIONS

The DECtape service routine reads the directory from a tape the first time it is required to perform a LOOKUP, ENTER, or UGETF; the directory image remains in core until a new ASSIGN command is executed from the console. To inform the DECtape service routine that a new tape has been mounted on an assigned unit, the user uses an ASSIGN command. The directory from the old tape can be transferred to the new tape, thus destroying the information on that tape unless the user reassigns the DECtape transport every time he mounts a new reel.

When positioning to a desired block on DECtape, the technique of dead reckoning is used. This means that the DECtape service routine starts the DECtape spinning and computes the time it should take to reach the desired block. Meanwhile, the service routine performs a service for another user, if any, and then returns just before the computed time has elapsed. If the desired block has not been reached, this process is repeated until it is successful. This technique is used to keep the controller free for other uses while the DECtape is spinning.

When an attempt is made to write on a write-locked tape or to access a drive that has no tape mounted, the message

DEVICE DTA*n* OPERATOR *zz* ACTION REQUESTED

is given to the user. When the situation has been rectified, CCONT may be typed to proceed. However, if this message is output because of an attempt to write on a write-locked tape and any operation that causes a RESET to be performed (e.g., a GET or RUN command) is then executed, a RELEASE will be done on the DECtape. This RELEASE causes any attempt to write the directory to output the same message. To avoid the second output of the message, the user should ASSIGN the DECtape again thus causing the DECtape service routine not to write the directory on the RELEASE.

Although DECtape is a file-structured block device, there is a limit to the number of files that may be opened simultaneously on a single DECtape. A given DECtape may be OPENed or INITed on two software channels (maximum) at the same time, once for INPUT and once for OUTPUT. An attempt to INIT on two channels for INPUT or two channels for OUTPUT generates no error indication, and only the most recent INIT is effective.

CHAPTER 10
I/O PROGRAMMING FOR MAGNETIC TAPE

Magnetic tape format for the DECsystem-10 is industry compatible. The tapes are unlabeled, 7- or 9-track; 200, 556, 800, or 1600 bpi. The device mnemonic is MTAx (MTA0, MTA1, MTB0, etc.), and the buffer size is 203_8 words (including 3 bookkeeping words). (Refer to the *DECsystem-10 System Reference Manual* for further information on -10 magnetic tape systems.)

The user may change the density and/or block size of a magnetic tape by using the SET DENSITY and SET BLOCK-SIZE commands. (Refer to the *DECsystem-10 Operating System Commands Manual).*

As far as the user is concerned, the tape contains only records and tape marks signaling the end of the record or the end of the file. A file consists of an integral number of physical records, separate from each other by inter-record gaps (an area on tape where no data is written). There may or may not be more than one logical record in each physical record. Write and read operations on files are performed sequentially. Tape marks are used in the following manner:

1. A tape mark follows every file.
2. Two tape marks follow a file if that file is the last or only file on the tape. (A double tape mark is also known as the logical end-of-tape.)
3. No tape mark precedes the first file on a tape.

When an output file is closed the I/O service routine automatically writes two tape marks and backspaces over one of them. If another file is opened, the second tape mark is written over leaving one tape mark between files. At the end of the used portion of the tape, a double tape mark appears (defined as the logical end of the tape).

Normally, all data is written with odd parity at 800 bpi (1600 bpi for TU70/TU43 magnetic tape systems); the default format can be changed by the system administrator at MONGEN-time. A maximum of 200_8 words per record is read or written if the monitor has set up the buffer ring. If the user specifies a buffer size (via SET BLOCK-SIZE), a maximum of 4094 words may be realized. If the user builds his own buffers, the 4094 limit may be bypassed.

The word count is not written on the tape. If an I/O error occurs, reading ahead ceases on input and output is terminated.

Below are some statistics concerning DECsystem-10 magnetic tape units:

Physical Name:	MTA0,MTA1, ... MTA7			
	MTB0,MTB1, ... MTB7			
	MTC0,MTC1, ... MTB7			
	MTD0,MTD1, ... MTD7			
Controller Name:	TM10A/B,TX01			
	TC10C			
Unit Name:	TU10/20/30/40/41/43/70/71			
Programmed Operators:	INPUT	OUTPUT	MTAPE	MTCHR.
	IN	OUT	TAPOP.	MTA1D.

Data Modes:	ASCII ASCII Line	Image Image Binary	Binary Dump Record Dump
Buffer Size:	203_8*		

10.1 DATA MODES

Table 1-1 lists the data modes available to magnetic tape users.

Table 10-1
Magnetic Tape Data Modes

Mode	Code	Meaning
ASCII	0	Data is written on the magtape exactly as it appears in the buffer. No processing of any kind is performed by the service routine. Parity checking by the magnetic tape system is sufficient assurance that the data is correct.
ASCII LINE	1	Same as ASCII.
IMAGE	10	This mode is the same as ASCII, but the data consists of 36-bit words.
IMAGE BINARY	13	Same as IMAGE.
BINARY	14	Same as IMAGE.
DUMP RECORDS (DR)	16	Data is in the form of standard, fixed-length records (128 words is the standard unless changed by the installation when generating its monitor or specified differently by the user with the SET BLOCKSIZE commands). Records read into or written from the user's core area are unbuffered. Control for read or write operations must be via a command list (described in paragraph 7.3.1) in core memory. For input operations, a new record is read for each word in the command list (except GOTO words); if the record terminates before the command word is satisfied, the service routine reads the next record. If the command word runs out before the record terminates, the remainder of the record is ignored. For each output command word, exactly enough standard-length records are followed by one short record to write all of the words on the tape. If an I/O error occurs or the EOT is reached, no additional commands are retrieved from a dump mode command list, and I/O is terminated. When the file mark is read, the user receives the standard end-of-file return (the error return from the IN call) and the IO.EOF bit is set in the file status word. (This bit can be retrieved with GETSTS monitor call, refer to paragraph 7.6.1). The EOF character is read into the user's buffer. The next INPUT or IN call will read the next record on tape. Must not use READ backward.
DUMP (D)	17	Variable-length records are read into or written from anywhere in the user's core area without regard to the buffering schemes. Control for read or write operations must be via a command list (described in paragraph 7.3.1) in core memory. For input operations a new record is read for each word in the command list (except for GOTO words); if the record terminates before the command word is satisfied, the service routine skips to the next command word. If the command word runs out before the record terminates, the remainder of the record is ignored. For each output command word, one record is written. Handling of EOF is the same as for DUMP RECORD (DR) as described above.

*The buffer size may be changed by using the SET BLOCKSIZE command.

10.2 SPECIAL MONITOR CALLS FOR MAGNETIC TAPE

There are several monitor calls that are available for magnetic tape users to perform certain functions. They are discussed in the following paragraphs.

10.2.1 The MTAPE Monitor Call

The MTAPE monitor call provides functions such as rewind, backspace a record, etc. Its calling sequence is

MTAPE *channel, function*
return

where: *channel* is the number of the software I/O channel on which the tape unit is initialized.
function is one of the function codes listed in Table 10-2.

Table 10-2
MTAPE Functions

Function Code	Mnemonic	Meaning
0	MTWAT.	A no-op which waits for spacing and I/O operations to finish.
1	MTREW.	Rewinds the magnetic tape to the load point.
3	MTEOF.	Writes a tape mark on the magnetic tape.
6	MTSKR.	Skips one record on the magnetic tape.
7	MTBSR.	Backspaces one record on the magnetic tape.
10	MTEOT.	Spaces to the logical end of the tape. Terminates two consecutive tape marks.
11	MTUNL.	Rewinds and unloads the tape. (Refer to paragraph 10.2.11).
13	MTBLK.	Writes 3 inches of blank tape.
16	MTSKF.	Skips one file, causing a series of skip record operations.
17	MTBSF.	Backspaces files, causing a series of backspace record operations.
100	MTDEC.	Initializes for DIGITAL-Compatible 9-track tape.[1]
101	MTIND.	Initializes for industry-compatible, 9-track tape.[2]
200	MTLTH.	Read next record at low threshold (TM10 Only).

[1] DIGITAL-Compatible mode writes (or reads) 36 bits in five frames of a 9-track magnetic tape. The tape can be any density or parity and is not industry-compatible. This mode is in effect until a RELEASE channel, or a MTIND. channel is executed.
[2] Industry-compatible mode writes (or reads) 32 bits in four frames of a 9-track tape and ignores the low-order four bits of a word. It must be 800 bpi density and odd parity.

MTAPE waits for the magnetic tape to complete the action in progress; bits 18-25 of the status word are then cleared, the indicated function is initiated (including the no-op) and control is immediately returned to the user program.

It is important to remember that the I/O service routine can be reading several blocks ahead of the user's program when performing buffered I/O. MTAPE affects only the physical position of the tape and does not change the data that has already been read into the buffers. Therefore, an INPUT or OUTPUT following an MTAPE call may not retrieve the buffer containing the block requested. However, a single buffer ring retrieves the expected block, since the device must stop after each INPUT/OUTPUT call. Alternatively, if bit 30 (IO.SYN) of the file status word is set via the INIT call or the SETSTS call, the device will stop after each buffer is filled on an INPUT or OUTPUT. The MTAPE will then apply to the buffer supplied on the next INPUT/OUTPUT.

Issuing a backspace file command to a magnetic tape unit will move the tape in the reverse direction until the tape has

1. passed a tape mark, or
2. reached the beginning of the tape.

The end of the backspace file operation positions the tape heads either immediately in front of a tape mark or at the beginning of the tape. In most cases, it is desirable to skip forward over this file mark up to the beginning of the file. In this case, giving a skip file command would skip the entire first file on the tape, stopping at the beginning of the second file rather than leaving the tape positioned at the beginning of the first file. Therefore, a correct sequence for "backspace file" would be:

```
MTBSF.   MT,       ;BACKSPACE FILE
MTWAT.   MT,       ;WAIT FOR COMPLETION
STATO    MT,4000   ;BEGINNING OF TAPE?
MTSKF.   MT,       ;NO, SKIP OVER FILE MARK
```

Since it is necessary to wait after the MTBSF, (i.e., backspace file) instruction to ensure that the move is completed before testing to see whether or not this is the beginning of the tape, the instruction WAIT MT, cannot be used for this purpose; it waits only for the completion of I/O transfer operations and backspace file is a spacing operation not an I/O operation.

The device service routine must wait until the magnetic tape control is free before processing the MTWAT., which tells the tape control to do nothing. The service routine achieves the waiting period necessary for the completion of the previous operation and the proper positioning of the tape.

10.2.1.1 Function 11, Rewind and Unload – MTAPE Channel, 11 (MTUNL.) initializes all automatic error reporting. Therefore, reel-specific errors can be summarized regardless of the method used to change reels. An entry into the system error log file (refer to the SYSERR Specification) will be written as follows:

drive number (e.g., MTxn)
SIXBIT/*reelid*/
number of characters read since last MTUNL.
number of characters written since the last MTUNL.
number of soft-read errors since the last MTUNL.
number of hard-read errors since the last MTUNL.
number of soft-write errors since the last MTUNL.
number of hard-write errors since the last MTUNL.

These numbers will be output on both the operator's terminal and the user's terminal in the following format:

[MT*xn*:*reelid* READ (*c*/*h*/*s*) = *a*/*b*/*c* WRITE (*c*/*h*/*s*) = *d*/*e*/*f*]

where: *x* is an alphabetic representing the tape controller,
n is a number representing the drive number.
reelid is the reel identification.
a is the number of characters read.
b is the number of hard-read errors.
c is the number of soft-read errors.
d is the number of characters written.
e is the number of hard-write errors.
f is the number of soft-write errors.

When *a*, *b*, and *c* are 0, the information pertaining to READ will not be printed.

When *d*, *e*, and *f* are 0, the information pertaining to WRITE will not be printed.

To prevent this message from being printed, the user can type the following command:

.SET WATCH NO MTA

10.2.2 The MTCHR. Monitor Call (CALLI 112)[1]

MTCHR. enables the user to obtain a set of data from which the current state of a specified magnetic tape drive can be determined. The calling sequence for MTCHR. is

```
{MOVE ac, [XWD n, addr]   }
{MOVE ac, [SIXBIT/device/]}
 MOVEI ac, channel
 MTCHR. ac,
    error return
 normal return
```

where: *n* is the number of words in the optional argument block pointed to by *addr*.
addr contains a left-justified SIXBIT physical/logical device name.
device is a physical/logical device name.
channel is the software I/O channel on which the *device* has been initialized.

On a normal return, the monitor will return values in the first n locations of the argument block as described in Table 10-3. The monitor will also return the information listed in Table 10-4 in the AC.

Table 10-3
MTCHR. Returned Values

Word	Mnemonic	Meaning
1	.MTRID	SIXBIT/reelid/
2	.MTFIL	The number of files from the beginning of the tape.
3	.MTREC	The number of records from last end-of-file.
4	.MTCRD	The number of characters read since last reload.
5	.MTCWR	The number of characters written since last reload.
6	.MTSRE	The number of soft-read errors since last reload.
7	.MTHRE	The number of hard-read errors since last reload.
10	.MTSWE	The number of soft-write errors since last reload.
11	.MTHWE	The number of hard-write errors since last reload.
12	.MTTME	The number of total media errors since last unload.
13	.MTTDE	The number of device errors since system load.
14	.MTTUN	The number of unloads since system load.
15	.MTRTY	The number of retries as a result of last error.
16	.MTCCR	Character count of last record read or written.
17	.MTPBE	The right half contains the record number; and the left half contains the position before the last error in the file.
20	.MTFES	The final error state (refer to the SYSERR specification).
21	.MTTRY	The number of retries to resolve the last error, bit 1=1 hard error.

[1] On a multi-processor system, MTCHR. runs only on CPU0.

The error return is taken if

1. the specified device is not a magnetic tape unit,
2. the specified device is not present, or
3. the MTCHR. monitor call has not been implemented.

If the specified device is not a magnetic tape unit or is not present, a -1 is returned in the AC. If the call has not been implemented, the contents of the AC are unchanged.

In determining the value of the density identifier to be returned to bits 33-35 of the AC (see Table 10-4), the monitor will examine the file status bit initialized by the INIT call and will return any INIT-specified density identifier. If density was not specified by an INIT, the monitor will then determine if the user has specified density using the SET DENSITY command, and it will return any user-supplied density identifier in the AC. If the SET DENSITY command has not been issued, the monitor will return the system default identifier in the AC. If no density is specified by an INIT, the GETSTS call will return a 0 in bits 33-35 of the AC. If GETSTS is used, the system default density identifier is not returned.

Table 10-4
Values Returned to the AC After MTCHR.

Bit(s)	Mnemonic	Meaning
0-17	MT.AWC	The actual word count for the last record read or written.
18-26	MT.CRC	The last cyclic redundancy character (CRC), if a 9-track NRZI tape is being used; otherwise, 0 (TU70s) return 0.
27-29	MT.NCR	The number of characters read from the tape into the last addressed location during the last read operation.
30		Zero.
31	MT.7TK	The unit is a 7-track unit.
32	MT.WLK	The transport is write-locked.
33-35	MT.DEN	The density identifier:
		code — mnemonic — meaning
		1 — .MTDN2 — 200 bits per inch (bpi)
		2 — .MTDN5 — 556 bpi
		3 — .MTDN8 — 800 bpi
		4 — .MTD16 — 1600 bpi

10.2.3 The TAPOP. Monitor Call (CALLI 154)

The TAPOP. monitor call allows a user program to control, examine, and modify information concerning any tape unit connected to the system. Many of TAPOP.'s functions are duplicates or extensions to the MTAPE and MTCHR. monitor calls. TAPOP.'s calling sequence is

```
        MOVE ac, [XWD n, addr]
        TAPOP. ac,
           error return
        normal return
        .
        .
        .
addr: function
      { SIXBIT/device/ }
      { channel        }
      { udx            }
```

channel
udx
argument0
.
.
.
argument$_n$

where: *n* is the number of words in the argument block.
addr points to the first word of the argument block.
function is one or more of the function codes listed in Table 10-5.
device, channel, and *udx* are alternative arguments specifying which device is to be used.
argument is different depending on the function code specified, refer to Table 10-5.

Table 10-5
TAPOP. Function Codes

Function Code	Mnemonic	Meaning
1	.TFWAT	Wait for I/O to stop.
2	.TFREW	Rewind the tape to the load point.
3	.TFUNL	Rewind and unload the tape.
4	.TFFSB	Skip forward 1 block.
5	.TFFSF	Skip forward 1 file.
6	.TFSLE	Skip to the logical end of the tape.
7	.TFBSB	Skip backward 1 block.
10	.TFBSF	Skip backward 1 file.
11	.TFWTM	Write a tape mark.
12	.TFWLG	Write three inches of blank tape.
13	.TFDSE	Data security erase (blank the entire tape) TX01 only.
14	.TFWLE	Write the logical end of the tape.
15[1]	.TFLBG	Get the tape label device data block.
16[1]	.TFLRL	Release the label device data block.
17[1]	.TFLSU	Swap units.
20[1]	.TFLDD	Destroy the tape label data base.
21	.TFFEV	Force the end of volume processing.
22	.TFURQ	User request for label processing.
1000	.TFTRY	Return in the AC the number of retries on the last error.
1001 2001	.TFDEN	Obtain (or set) the density indicator, either: code mnemonic meaning 0 .TFD00 unit default bpi 1 .TFD20 200 bpi 2 .TFD55 556 bpi

[1]Performed by tape label manager, privileged for use by label processor.

Table 10-5 (Cont.)
TAPOP. Function Codes

Function Code	Mnemonic	Meaning
		code mnemonic meaning 3 .TFD80 800 bpi 4 .TFD16 1600 bpi (TU70/43 only) 5-17 Reserved.
1002	.TFKTP	The controller type, either: code mnemonic meaning 0 .TFKTA TM10A(TU10/20/30/40/41) 1 .TFKTB TM10B(TU10/20/30/40/41) 2 .TFKTC TC10C(TU43) 3 .TFKTX TX01(TU70/71)
1003 2003	.TFRDB	Read backward (TX01 only). Refer to paragraph 10.2.3.2.
1004 2004	.TFLTH	Read next record at low threshold (TM10 only).
1005 2005	.TFPAR	Set or obtain status of the even parity bit (7-track only).
1006 2006	.TFBSZ	Set or obtain the block size.
1007 2007	.TFMOD	Set or obtain the data mode, either: code mnemonic meaning 0 .TFMDD DEC-compatible core dump (7-track and 9-track). 1 .TFM9T Core dump format (9-track). 2 .TFM8B Industry-compatible, 8-bit mode (4 bytes/word). 3 .TFM6B 6-bit mode (9-track, TU70 only). 4 .TFM7B 7-bit mode (TU70 only). 5 .TFM7T 7-track core dump (SIXBIT). Refer to paragraph 10.2.3.1.
1010 2010	.TFTRK	Set or obtain the track status bit (1=7-track, 0=9-track). It is a privileged function to set this bit.
1011	.TFWLK	The write-lock bit (set =1, off = 0).
1012	.TFCNT	The character count of the last record (the actual record length).
1013 2013	.TFRID	Set or obtain the reel I.D. (bits 0-35). It is a privileged function to set this word.
1014	.TRCRC	The last Cyclic Redundancy Character (CRC) (9-track NRZI only).
1015	.TFSTS	The unit status word:

Table 10-5 (Cont.)
TAPOP. Function Codes

Function Code	Mnemonic	Meaning
		bit / mnemonic / meaning 18 / TF.UNS / Unit is not to be scheduled. 19 / TF.BOT / Beginning-of-tape mark. 20 / TF.WLK / Unit is write-locked. 21 / TF.REW / Unit is rewinding. 22-32 / / Reserved. 33 / TF.STA / Unit is started. 34 / TF.SEL / Unit is selected. 35 / TF.OFL / Unit is off-line.
1016	.TFSTA	The unit statistics to arguments 0 through 12: code / mnemonic / meaning 0 / .TSFIL / Number of files since BOT. 1 / .TSREC / Number of records since EOF. 2 / .TSTCR / Number of characters read. 3 / .TSTCW / Number of characters written. 4 / .TSSRE / Number of soft-read errors. 5 / .TSHRE / Number of hard-read errors. 6 / .TSSWE / Number of soft-write errors. 7 / .TSHWE / Number of hard-write errors. 10 / .TSESU / Total errors since unload (MOUNT). 11 / .TSTDE / Total device errors since system startup. 12 / .TSUNL / Total number of unloads.
1017	.TFIEP	Initial error pointer.
1020	.TFFEP	Final error pointer.
1021	.TFIES	Initial error status.
1022	.TFFES	Final error status.
1023	.TFFED	Final error disposition.
1024	.TFLBL	The type of label processing. code / mnemonic / meaning 0 / .TFBLP / Bypass labeled processing. 1 / .TFLAL / ANSI labels. 2 / .TFLAU / ANSI labels with user labels. 3 / .TFLIL / IBM labels. 4 / .TFLIU / IBM labels with user labels. 5 / .TFLTM / Leading tape mark. 6 / .TFLNS / Non-standard labels. 7 / .TFLNL / No labels.
1025	.TFPLT	The same as function code 1024 (.TFLBL) except that .TFPLT is privileged function to set 0 (.TFBLP.)
1026	.TFLTC	Label termination code.

Table 10-5 (Cont.)
TAPOP. Function Codes

Function Code	Mnemonic	Meaning
		code mnemonic meaning 1 .TFTCP Continue processing. 2 .TFTRE Return EOF. 3 .TFTLT Label type error. 4 .TFTHL Header label error. 5 .TFTTL Trailer label error. 6 .TFTVL Volume label error. 7 .TFTDV Device error. 10 .TFTDE Data error. 11 .TFTWL Write-lock error.

On an error return, one of the error codes listed in Table 10-6 will be returned in the AC.

Table 10-6
TAPOP. Error Codes

Error Code	Mnemonic	Error Condition
-1	TPACS%	Address check storing answer.
0	TPIFC%	An illegal function code was specified.
1	TPPRV%	Function code specified requires special privileges.
2	TPNMT%	Device is not a magnetic tape unit.
3	TPVOR%	Value specified is not in the legal range.
4	TPACR%	Address check while reading arguments.
5	TPCBS%	Specified parameter cannot be set.
6	TPNIA%	The tape unit has not been INITed or ASSIGNed.

The function codes are defined within the following ranges:

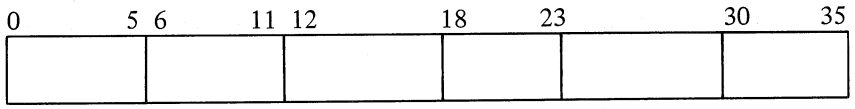
0000-0777	perform a specific action
1000-1777	read a parameter
2000-2777	set a parameter
3000-3777	reserved for DEC customers

The READ functions (codes 1000-1777) and the SET functions (codes 2000-2777) are parallel; i.e., if function 1002 reads a particular parameter, function 2002 sets the same parameter. The values for the READ functions are returned in the AC; arguments to the SET functions are supplied by the user program in addr+2. One bit quantities are not range-checked; instead, bit 35 of addr+2 is stored. Those functions in Table 10-5 that have codes in both the 1000-1777 and the 2000-2777 ranges have parallel SET and READ functions.

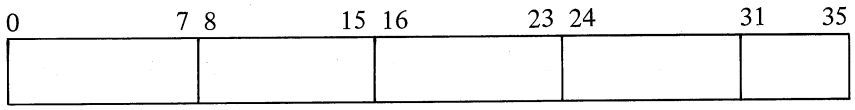
10.2.3.1 Function .TFMOD (Data Modes) – The legal data modes are described below.

0 (.TFMDD) – DEC-compatibility core dump format for 7-track and 9-track (default).

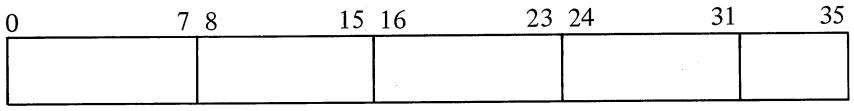
7-track
(6-bit bytes)



9-track

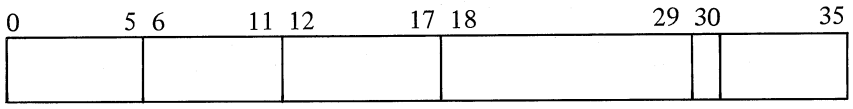


1 (.TFM9T) – Core dump for 9-track (see 9-track code 0).
2 (.TFM8B) – Byte Mode (9-track only) Industry-compatible.

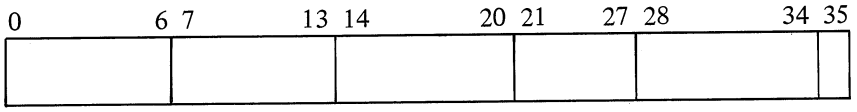


4 8-bit bytes/word, bits 32-35 are zero on a read from TU70.

3 (.TFM6B) – SIXBIT Mode (9-track, TU70 only).

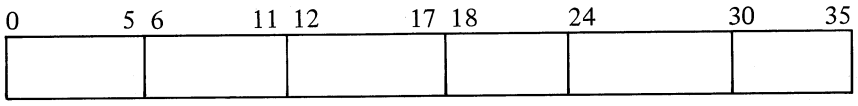


4 (.TFM7B) – 7-bit mode (TU70 only) Industry-compatible mode.



4 7-bit characters
1 8-bit character folded (bit 35)

5 (.TFM7T) – 7-track core dump mode.



10.2.3.2 Read Backwards (TX01 only) – In reading a tape backwards, data is read forwards (i.e., inverted) into the buffer. If the buffer is larger than the record, a zero-fill will result at the beginning of the buffer. For example:

BUFSIZ – WRDCNT = first word of data

If the user program contains a multiple IOWD I/O list, the I/O bit must be reversed by the user program, in order to enable read forward. Reading backwards into BOT will set EOF or the BOT bit. Note that a tape cannot be read backwards if mode 16 is used.

Reads backwards must be set after the OPEN, as the read backwards function is cleared on an OPEN or a RELEASE monitor call.

10.2.4 The MTAID. Monitor Call (CALLI 126)[1]

The MTAID. monitor call is used by OMOUNT to associate a SIXBIT tape reel identifier with a specific magnetic tape drive. The calling sequence for MTAID. is

```
{MOVE ac, [SIXBIT/device/]}
{MOVEI ac, channel        }
{MOVEI ac, udx            }
 MOVE ac+1, [SIXBIT/reelid/]
 MTAID. ac,
    error return
 normal return
```

where: *device* is the physical/logical name of the device.
channel is the software I/O channel number associated with device.
udx is the universal device index associated with the device.
reelid is the tape reel identifier.

The error return is taken if the caller does not have the appropriate privileges, if the device specified does not exist or is not a magtape device, or if the call has not been implemented. A -1 is returned in the AC if the error was a privilege or device-type error, otherwise the AC is unchanged.

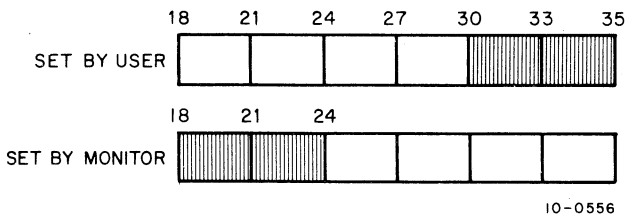
On a normal return, the reel identification (reelid) will be stored in the monitor, and it will be included in media error reports.

REELID may be cleared by using the MTAPE function MTAPE II (MTUNL.) UNLOAD. All reel-specific error counts will be cleared by MTAID. in order that all accumulated data for the specific reel will be accurate, which is important when the previous user has forgotten to issue the UNLOAD command.

10.3 FILE STATUS

The file status of the magnetic tape is shown below.

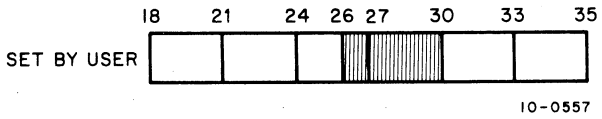
Standard Bits



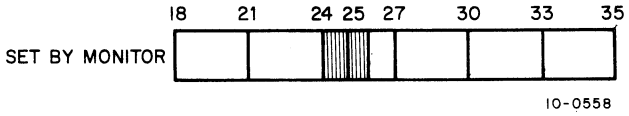
[1]This is a privileged monitor call.

Bit 18 – IO.IMP	Unit was write-locked when output was attempted, or illegal operation was specified to the magnetic-tape control.
Bit 19 – IO.DER	Data was missed, tape is bad, or transport is hung.
Bit 20 – IO.DTE	Parity error.
Bit 21 – IO.BKT	Record read from tape exceeds buffer size.
Bit 22 – IO.EOF	Tape mark encountered.
Bit 23 – IO.ACT	Device active.

Device Dependent Bits



Bit 26 – IO.PAR	I/O parity; 0 for odd parity, 1 for even parity. Odd parity is preferred. Even parity should be used only when creating a tape to be read in binary coded decimal (BCD) on another computer.
Bit 27-28 – IO.DEN	I/O density, 00=System standard. Defined at MONGEN-time and can be changed with the SET DENSITY command. 01=200 bpi 10=556 bpi 11=800 bpi If any density other than 200/556/800 bpi need be specified, use the appropriate TAPOP. (.TFDEN). A code of 00 will select the system standard, which may be other than 200/556/800 bpi.
Bit 29 – IO.NRC	I/O no read check. Suppress automatic error correction if bit 29 is 1.



Bit 24 – IO.BOT	I/O beginning of tape. Unit is at beginning of tape mark.
Bit 25 – IO.EOT	I/O tape end. Physical end of tape mark encountered.

NOTE

If bits 18, 19, 20, and 21 contain one, an error was detected by the tape labeling process and a TAPOP. can be used to determine the actual error (.TFLTC).

CHAPTER 11
I/O PROGRAMMING WITH TERMINALS

11.1 INTRODUCTION

Asynchronous communications equipment and terminals cover a broad range of applications within the DECsystem-10. Asynchronous equipment is used for interactive program development, for operator control of the system, for program production and control, and for running data entry, program preparation, interactive problem-solving, student instruction, and information storage and retrieval. Asynchronous communications terminal equipment is of two types: hard copy, such as the LA30 and LA36 DECwriters, and CRT terminals such as the VT05 and VT50 DECscopes.

Local terminals within 1500 feet of the computer site can be connected or dedicated direct electrical connections. Terminals located beyond this distance are designated as remote terminals. Remote terminals are connected via dial-up telephone lines to the central computer site.

Table 11-1 lists some of the characteristics of Digital-supported terminals.

Table 11-1
Terminals

Name	Controller Number	Unit Number	Monitor Calls	Data Modes	Buffer Size
TTY0 TTY1 . . . TTY777	DC72-L DC10-A DC10-B DC10-C DC10-D DC76	LA36 LA30 LT33A LT33B LT35A LT37AC VT06 VT50 VT05 GT40	INPUT IN OUTPUT OUT TTCALL TRMNO. GETLIN. TRMOP.	ASCII ASCII Line Image PIM	23_8

11.2 TERMINAL MONITOR CALLS

11.2.1 The TTCALL Monitor Call (Op Code 051)

TTCALL is used for terminal operations. TTCALL operations are performed for a physical terminal (not a TTY with a logical name), and most operations reference the terminal controlling the job that executed the TTCALL. (There are exceptions, such as in using GETLCH). TTCALL's calling sequence is

TTCALL *Code, Addr*

where: *Code* is one of the function codes listed in Table 11-2.

Addr points to an argument, if one is desired.

Table 11-2
TTCALL Functions

Function Code	Mnemonic[1]	Meaning
0	INCHRW	Input a character into the low-order seven bits of addr. If the user has not yet typed a character, the program will wait until one is typed.
1	OUTCHR	Output the character stored in addr to the user's terminal. Only the low-order seven bits are used, but the remaining bits do not have to contain zeroes.
2	INCHRS[2]	Input a character into the low-order seven bits of addr. If the user has not yet typed a character, the skip return is taken. INCHRS will not wait until the user types a character, as will INCHRW.
3	OUTSTR	Output a string of ASCIZ characters to the user's terminal.
4	INCHWL[2]	INCHWL is the same as INCHRW, except that INCHWL determines whether or not to wait on the basis of lines rather than characters. INCHRW causes a swap to occur for each character rather than each line. Therefore, INCHWL returns the next line if a break character is typed. If a break character has not been typed, INCHWL waits. Repeated uses of INCHWL return each of the successive characters of the line. Note that a CTRL/C in the input buffer satisfies the condition of a pending line. When the input is finished, the CTRL/C is interpreted and the job is stopped.
5	INCHSL[2]	Input a line to addr. If the line has not yet been typed, a skip is made.
6	GETLCH	Get the line characteristics of the specified line. The user supplies a line number in the right half of addr. If the line number is greater than those defined in the system, a zero answer is returned. The monitor will return, in the left half of addr, the line characteristics associated with the specified line number. The possible line characteristics are listed in Table 11-3.
7	SETLCH	Set the line characteristics for a specified line. The user supplies a line number in the right half of addr, and the line characteristics to be set in the right half of addr. Bits may be changed only when issued for the job's controlling TTY. Table 11-3 lists the possible line characters, note that only bits 13, 14, 15, and 16 can be modified.

Table 11-2 (Cont.)
TTCALL Functions

Function Code	Mnemonic[1]	Meaning
10	RESCAN	The input buffer is scanned from the point where the last command began. If bit 35 of addr contains 1, the error return is taken if there is a command in the input buffer. If the input buffer is empty, the skip return is taken. If RESCAN is issued after the first input, it may no longer be in the buffer. The address of addr is checked, but it is not used. RESCAN is intended for use only by the COMPIL program.
11	CLRBFI	Clear the input buffer in the same manner as though the user had typed CTRL/Us. CLRBFI can be used when the user program detects an error.
12	CLRBFO	Clear the output buffer in the same manner as though the user had typed a CTRL/O. This TTCALL is used infrequently, since most users would like to see all output to the terminal up to the point of an error.
13	SKPINC[2]	Skip if the user has typed at least one character. SKPINC does not skip if the user has not yet typed any characters; however, it never inputs a character. SKPINC is useful for a compute-bound program that wants to occasionally check for input and, if any, go off to another routine to perform the input. Note that the skip occurs whether or not the character has been echoed (examined for echoing if NO ECHO has been set). But, an attempt to input will input only those characters that have echoed, and it will wait if no characters have yet echoed.
14	SKPINL[2]	Skip if the user has typed at least one line. Note that SKPINL skips only if the line has been echoed (examined for echoing if NO ECHO has been set).
15	IONEOU	Output the low-order eight bits of addr as an image character.

[1] The TTCALL mnemonics are defined in a separate MACRO assembler table, which is scanned if an undefined Op Code is found. If the symbol is found in the TTCALL table, it will be defined as if it had appeared in an appropriate OPDEF statement.
[2] This function clears the effect of the last ↑O.

11.2.1.1 TTCALL Examples – Read One Character: The INCHRW TTCALL will input a character into the low-order seven bits of the specified location, addr. If no character has been typed, the program will wait until a character is typed.

```
INCHRW CHAR
JRST DONE
```

Type One Character: The OUTCHR TTCALL outputs the character stored in addr to the user's terminal. The low-order seven bits of addr are used; the remaining bits, though, do not have to contain zeroes.

```
OUTCHR OUTADR
JRST DONE
```

Type a String: The OUTSTR TTCALL outputs a string of characters in ASCIZ format to the terminal;

```
OUTSTR TXT
JRST DONE
```

TYPE THIS OUT would be printed at the terminal if the following was stored in location TXT:

```
TXT:  ASCIZ/TYPE THIS OUT/
```

Table 11-3
Terminal Line Characteristics

Bit	Mnemonic	Meaning
0	GL.ITY	The line is a pseudo-TTY.
1	GL.CTY	The line is a CTY.
2	GL.DSP	The line is a display console.
3	GL.DSL	The line is a dataset data line.
5	GL.HDP	The line is a half-duplex line.
6	GL.REM	The line is a remote TTY.
7	GL.RBS	The line is a remote Batch TTY.
11	GL.LIN	A line has been typed-in by the user.
13	GL.LCM	Lower case input mode is ON.
14	GL.TAB	The terminal has TAB capabilities.
15	GL.LCP	The terminal input is not echoed because the device is local-copy only.
16	GL.PTM	The CTRL/Q (paper-tape) switch is on.

11.2.2 The GETLIN Monitor Call (CALLI 34)

The GETLIN monitor call will return the SIXBIT physical name of the terminal to which the calling job is attached. Its calling sequence is

GETLIN *ac,*

The physical name of the terminal is returned left-justified in the AC. If the calling job is currently detached, the left half of the AC will contain zeroes. The right half of the AC contains the right half of the physical name of the terminal to which the calling job was most recently attached. A job can determine whether or not it is attached to a terminal by examining the left half of the AC.

11.2.3 The TRMNO. Monitor Call (CALLI 115)

The TRMNO. monitor call is used to obtain the number of the terminal currently controlling a specified job. Its calling sequence is

```
MOVE     ac, job-number
TRMNO,   ac
   error return
normal return
```

where: *job-number* is the number of the job for which its controlling terminal's number is desired.

On a normal return, the right half of the AC contains the universal I/O index for the terminal (in the format .UXxxx). The range of values of 200000_8 to 200777_8. The symbol .UXTRM (200000_8) is the offset for the terminal indices.

On an error return, the AC is unchanged if the monitor call has not been implemented. If the AC contains a zero, one of several error conditions occurred. The possible error conditions are

1. The job is currently detached (i.e., there is no controlling terminal).
2. The job number specified is unassigned (i.e., there is no such job), or
3. The job number specified is illegal (i.e., out of range).

The particular condition which caused the error may be obtained by using the JOBSTS monitor call. An example of this follows.

```
MOVEI  ac, job-number
TRMNO, ac,
   JRST  .42
JRST  OK
JUMPN  ac,        ;CALL NOT IMPLEMENTED
MOVNI  ac, job-number
JOBSTS  ac,
   JRST  ILLNUM
JUMPL  ac, DETJOB
JRST  NOJOB
```

11.2.4 The TRMOP. Monitor Call (CALLI 116)

The TRMOP. monitor call allows the user to control, examine, and modify information regarding any terminal connected to the system. Many TRMOP. functions are extensions to the TTCALL input and output functions (refer to Paragraph 11.2.1). TRMOP.'s calling sequence is

```
                MOVE  ac, [XWD n, addr]
                TRMOP, ac,
                   error return
                normal return

addr:           function code
addr+1:         udx
```

where: *n* is the length of the argument block, which must be at least 2 words.

addr is the address of the argument block.

function code is the desired TRMOP. function; all function codes are listed in Table 11-4.

udx is the universal device index of the terminal (.UXTRM + line number).

Table 11-4
TRMOP. Function Codes

Code	Mnemonic	Effect
1	.TOSIP	Skip if the terminal's input buffer is not empty.
2	.TOSOP	Skip if the terminal's output buffer is not empty.
3	.TOCIB	Clear the terminal's input buffer.
4	.TOCOB	Clear the terminal's output buffer.
5	.TOOUC	Output the normal mode character in addr+2 to the terminal.
6	.TOOIC	Output the image mode character (8 bits) in addr+2 to the terminal.
7	.TOOUS	Output the ASCIZ string in addr+2 to the terminal.
10	.TOINC	Input the the normal mode character from the terminal to the AC.
11	.TOIIC	Input the image mode character from the terminal to the AC.
12	.TODSE	Enable the modem for outgoing calls.
13	.TODSC	Enable and place outgoing call on a modem with a dialer. A phone number of up to 17 digits is stored in 4-bit bytes in addr+2 and addr+3. The phone number is terminated by a 17 byte. If the caller must wait for a second dial tone (e.g., after dialing 9), a 16 byte results in a 15 second wait.
14	.TODSF	Disconnect a call (i.e., hang up modem).
15	.TORSC	Set the terminal element to the element number stored in addr+2.
16	.TOELE	Perform a rescan.
17	.TOEAB	Enable automatic baud direction.
1000	.TOOIP	Output is in progress if bit 35 = 1.
1001	.TOCOM	Terminal is in monitor mode if bit 35 = 1.
1002 2002	.TOXON	Set or obtain the status of the paper-tape bit (if bit 35 = 1, the terminal is in paper-tape mode).
1003 2003	.TOLCT	Set or obtain the lower-case capabilities of the terminal (If bit 35 = 1, the terminal has no lower case capabilities.)
1004 2004	.TOSLV	Set or obtain the slave characteristic of this terminal (if bit 35 = 1, the terminal is slaved).

Table 11-4 (Cont.)
TRMOP. Function Codes

Code	Mnemonic	Effect
1005 2005	.TOTAB	Set or obtain the TAB capabilities of the terminal (if bit 35 = 1, the terminal performs TABs).
1006 2006	.TOFRM	Set or obtain the value of the FORM switch (if bit 35 = 1, the terminal performs formfeeds; if bit 35 = 0, the terminal performs linefeeds only).
1007 2007	.TOLCP	Set or obtain the value of the local copy switch (if bit 35 contains 1, characters will not be echoed).
1010 2010	.TONFC	Set or obtain the value of the CR/LF switch (if bit 35 = 1, free carriage returns/linefeeds will not be performed).
1011	.TOHPS	Read the value of the horizontal position of the carriage (a value from 0 to 377 is returned in the AC).
1012 2012	.TOWID	Set or obtain the carriage width (this value may be set from 16 to 200).
1013 2013	.TOSND	Set or obtain the TTY GAG switch (if bit 35 = 1, NOGAG).
1014 2014[1]	.TOHLF	Set or obtain the half-duplex characteristics of this terminal (if bit 35 = 1, the terminal is in half-duplex mode).
1015 2015[1]	.TORMT	Set or obtain the remote status of this terminal (if bit 35 = 1, the terminal is remote).
1016 2016[1]	.TODIS	Set or obtain the display characteristic of this terminal (if bit 35 = 1, the terminal is a display device).
1017 2017	.TOFLC	Set or obtain the filler class code associated with this terminal. (The filler class code may be set from 0 to 3).
1020 2020	.TOTAP	Set or obtain the status of the paper-tape (if bit 35 = 1, paper-tape has been enabled).
1021 2021	.TOPAG	Set or clear the paged display mode (if bit 35 = 1, paged display mode is cleared). Paged display mode can also be set and cleared by the SET TTY PAGE command).
1022		Reserved to Digital.
1023 2023	.TOPSZ	Set or obtain the number of lines to a page in the range 0 to 63. The size of a page may also be changed by the SET TTY PAGE command.
1024	.TOPCT	Set the value of the page counter in the range 0 to 63.

Table 11-4 (Cont.)
TRMOP. Function Codes

Code	Mnemonic	Effect
1025	.TOBLK	Set or clear the capability to suppress blank lines on output (if bit 35 = 1, normal output is performed, if bit 35 = 0, multiple linefeeds are suppressed and vertical tabs are changed to linefeeds).
1026 2026	.TOALT	Set or clear the capability of converting ALTmodes on input (if bit 35 = 0, 175 and 176 are converted to 033; if bit 35 = 1, no conversion is performed).
1027 2027	.TOAPL	Set or clear APL mode (if bit 35 = 1, APL mode is in effect).
1030 2030	.TORSP	Set or obtain the received speed; refer to Table 11-5.
1031 2031	.TOTSP	Set or obtain the transmit speed; refer to Table 11-5.
1032 2032	.TODBK	Set or clear debreak capabilities (if bit 35 = 1, debreak is enabled).
1033 2033	.TO274	Set or clear 2741 terminal characteristics (if bit 35 = 1, the terminal is a 2741). Refer to Appendix K for more information on 2741s.
1034	.TOTDY	Obtain the status of the TIDY word.
1035 2035	.TOACR	Set or clear the automatic carriage return facility. If bit 35 = 1, the first space after the specified column is automatically converted to a carriage return.
1036 2036	.TORTC	Obtain or set the status of CTRL/R and CTRL/T compatibility mode.
1037	.TOPBS	PIM mode break set (4 9-bit bytes).

[1] This is a privileged function.

On an error return, the AC will either contain an error code or will be unchanged if the call has not been implemented. The possible error codes are listed in Table 11-6. Some TRMOP. functions are privileged functions or they require that the user have the specified terminal ASSIGNed. Generally, any function is legal for the calling job's terminal. In addition, any READ or SKIP function is legal for any terminal if the calling job

1. has the privilege bit JP.SPM set.
2. is running with the JACCT bit set, or
3. is logged in with the project/programmer number 1,2.

A SET or output function is legal for any terminal if the job

1. has the privilege bit JP.POK set,
2. is running with the JACCT bit set, or
3. is logged in with the project/programmer number 1,2.

The function codes are defined in the following manner:

function codes 0000-0777	perform a specific action; refer to Table 11-4
function codes 1000-1777	read a parameter; refer to Table 11-4
function codes 2000-2777	set a parameter; refer to Table 11-4
function codes 3000-3777	are reserved for DEC customers

The READ (1000-1777) and SET (2000-2777) functions are parallel; i.e., if function 1002 reads a particular parameters, then function 2002 sets the same parameter. Values for the READ functions are returned in the AC; arguments to the SET functions are passed in addr+2. One-bit quantities are not range-checked; instead, bit 35 of addr+2 is stored. If a 1nnn and 2nnn code appear in the first column of Table 11-4 within braces, the Read function has a corresponding SET function.

Table 11-5
Transmit/Receiving Speeds

Code	Speed	Code	Speed	Code	Speed	Code	Speed
1	50	5	150	11	1200	15	9600
2	75	6	200	12	1800	16	External A
3	110	7	300	13	2400	17	External B
4	134.5	10	600	14	4800		

Table 11-6
TRMOP. Error Codes

Code	Name	Meaning
0		The call has not been implemented.
1	TOPRC%	User is not privileged to perform this function.
2	TORGB%	Argument is out of range.
3	TOADB%	Argument list length or address is illegal.
4	TOIMP%	Dataset activity to a non-dataset terminal.
6	TODIL%	Subfunction failed (e.g., call not properly completed from dialer).

11.3 DATA MODES

ASCII (American Standard Code for Information Interchange) is a standard character set encoded in 7 bits (8 bits including a parity bit). The ASCII set consists of 128 characters, 33 of which are non-printing control characters. The following table describes how the characters are handled.

000	NULL	Ignored on input; suppressed on output.
001	↑A	No special action.
002	↑B	No special action.
003	↑C	Not passed to program. The user's terminal is switched to monitor mode the next time input is requested by the program. Two successive ↑Cs cause the terminal to be immediately switched to monitor mode. Performs a ↑U and a ↑O. For user program control of ↑C, refer to Paragraph 3.1.3.2.
004	↑D	Not echoed; therefore, typing in a control-D (EOT) does not cause a full-duplex data phone to hang up.
005	↑E (WRU)	No special action.
006	↑F	No special action.
007	↑G (Bell)	Echoes as Bell and is a break character.
010	↑H (Backspace)	Echoes as backspace. This is not passed to the program unless APL or special editor mode has been set, instead it functions like RUBOUT.
011	↑I (TAB)	Echoes as a TAB or an equivalent number of spaces. Refer to the SET TTY TAB command.
012	↑J (Linefeed)	Echoes as a linefeed and is a break character.
013	↑K (Vertical Tab)	Echoes as a vertical tab or four linefeeds. Refer to the SET TTY FORM command.
014	↑L (Form)	Echoes as a FORMFEED or eight linefeeds. Refer to the SET TTY FORM command.
015	↑M	Passed to program if terminal is in a paper-tape input mode; otherwise, supplied a linefeed echo, is passed to program as a CR and LF, and is a break character due to the LF.
016	↑N	No special action.
017	↑O	Not passed to program. Complements output suppression bit allowing users to turn output on or off. INPUT, INIT, and OPEN clear the output suppression bit. This bit is also cleared by any other INPUT-class operation, such as DDTIN and TTCALLS 0, 2, 4, and 5, by input test TTCALLS 13 and 14, and by returning to monitor command level via ↑C or EXIT operations. Echoed as ↑O followed by carriage return/linefeed.
020	↑P	No special action.
021	↑Q (XON)	Starts paper-tape mode if TTY TAPE command has been given.
022	↑R	Retype the line currently being input, including the effect of any edits to the line.
023	↑S (XOFF)	Ends paper-tape mode.

024	T	Gives job status and timing information.
025	U	Deletes input line back to last wakeup character. Echoed as U followed by a carriage return/linefeed; is a break character. Passed to program if special editor mode is true.
026	V	No special action.
027	W	No special action.
030	X	No special action.
031	Y	No special action.
032	Z	Acts as EOF of TTY input. Echoes as Z followed by carriage return/linefeed. Is a break character.
033	[(ESC)	The standard ASCII escape. Echoed as $; is a break character.
034		No special action.
035	]	No special action.
036		No special action.
037		No special action.
040-137		Printing characters, no special action.
140-174		Lower case ASCII; translated to upper case, unless lower case mode is on. Echoes as upper case if translated to upper case.
175 and 176		Old versions of altmode; converted to the standard escape (033) unless in special editor mode (INIT or TRMOP.UUO) or no altmode conversion is specified (TRMOP.UUO or SET TTY NO ALT command).
177		RUBOUT or DELETE: 1. Completely ignored if in paper-tape mode (XON). 2. Break character, passed to program if either DDT mode or special editor mode is true. 3. Otherwise (ordinary case) causes a character to be deleted for each rubout typed. All the characters deleted are echoed between a single pair of backslashes. If no characters remain to be deleted, echoes as a carriage return/linefeed.

On output, all characters are typed just as they appear in the output buffer with the exception of TAB, VT, and FORM, which are processed the same as on type-in. Programs should avoid sending D, because it may hang up certain data sets.

Image mode (octal code 10) is legal for TTY input and output except for pseudo-TTY's (refer to Paragraph 5.9). Image mode is especially convenient for users of display devices, light pens, etc., since any sequence of input characters is allowed. The user must use the ASSIGN command before the INIT command can be used in image mode. (The user's own TTY is always assigned by logging in.) An attempt to do input to an unassigned terminal

results in an error return. Since any sequence of characters must be allowed, Control-C and Control-Z will not cause their usual functions. If the user program accepts all characters, and does not release the terminal from image mode, no user input will release the user from this state. The terminal would effectively become dead to the system. Because of this situation, an image input state is defined. The image input state begins when the program starts waiting as a result of an INPUT UUO in image mode. It ends when the program executes any non-image mode terminal output operation. If no output is desired, a TTCALL UUO can be executed to output a null character. If no input characters are received for 10 seconds the EOF is forced. After another 10 seconds, the image input state is terminated by the monitor and a Control-C is simulated. If the user should be in this situation, he should stop typing until the Control-C appears.

Packed Image Mode (octal code 2) is legal for TTY input and output (.PIM is not a legal mode for pseudo-TTYs). PIM is designed for high efficiency character throughput between programs and external devices. This is accomplished by minimizing the monitor's character manipulation and testing.

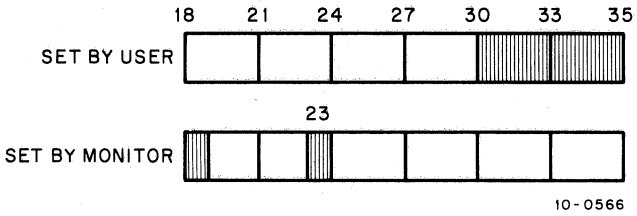
In Packed Image Mode, characters are maintained as 8-bit quantities (i.e., 7 data bits and 1 parity bit), which are stored in buffers at the rate of 4 characters per word. The user program may set a "break" consisting of one to four break characters for each line INITed in Packed Image Mode. The break set is defined via the TRMOP. monitor call.

When the monitor receives a character from a Packed Image Mode line with a pending IN or INPUT, the character is compared to each field in the break set. If no match occurs, the character is put in the buffer and the interrupt is dismissed. In the case where a match does occur, the character is put into the buffer, the buffer is then terminated and the controlling program is awakened. To avoid the possibility of a terminal setting stuck in PIM mode, and to allow for the case where the user wishes to be awakened on each character, the user program specifies an empty (0,,0) break set. In general operation, ALL characters, including control characters are passed by PIM with no monitor intervention (e.g., fill, CR/LF, etc.) with the following exceptions: if PAGE mode is set, the characters XON and XOFF react in the normal PAGE mode sense.

11.4 FILE STATUS

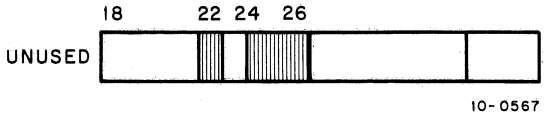
The file status of the terminal is shown below.

Standard Bits

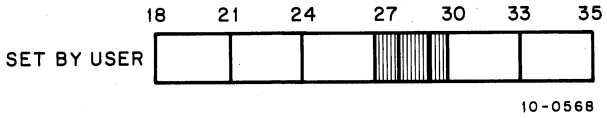


Bit 18 = IO.IMP TTY is not assigned to a job (for image mode input processing).

Bit 23 = IO.ACT Device is active



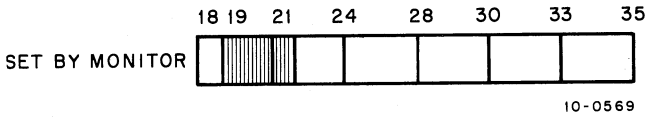
Device Dependent Bits



Bit 27 - IO.TEC — This bit causes 001 through 037,175, and 176 (octal) to echo the character exactly as received by the monitor. THERE IS NO SPECIAL ECHO (E'G" $ OR X).

Bit 28 - IO.SUP — Suppresses echoing on the terminal.

Bit 29 - IO.SEM — Special editor mode. Pass all characters except lower case and C. Lower case is controlled by the SET TTY LC command and corresponding TRMOP. function.



Bit 19 - IO.DER — Ignore interrupts for three-fourths of a second.

Bit 20 - IO.DTE — Echo failure has occurred on output.

Bit 21 - IO.BKT — Character was lost on typein.

11.5 PAPER-TAPE INPUT FROM THE TERMINAL (FULL-DUPLEX SOFTWARE)

Paper-tape input is possible from a terminal equipped with a paper-tape reader that is controlled by the XON (Q) and XOFF (S) characters. When commanded by the XON character, the terminal service reads paper tapes, starting and stopping the paper tape as needed, and continuing until the XOFF character is read or typed in. While in this mode of operation, any RUBOUTS will be discarded and no free line feeds will be inserted after carriage return. Also, TABS and FORMFEEDS will not be simulated on a Model 33 to ensure output of the reader control character–characters. To use paper-tape processing, the terminal with a paper-tape reader must be connected by a fullduplex connection and only ASCII paper tapes should be used.

The correct operating sequence for reading a paper tape in this way is as follows:

```
.SET TTY TAPE↵
.R PIP↵
*DSK:FILE←TTY:↵
^QTHIS IS WHAT IS ON TAPE↵
MORE OF THE SAME↵
LAST LINE↵
^Z
*^S^C
```

11.6 PAPER–TAPE OUTPUT AT THE TERMINAL (FULL-DUPLEX SOFTWARE)

Paper-tape output is possible on any terminal-mounted paper-tape punch, which is controlled by the TAPE, AUX ON (R) AND AUX OFF (T) characters. The punch is connectedin parallel with the keyboard printer and, therefore, when the punch is on, all characters on the keyboard are punched on tape.

LT33B or LT33H terminals can have the reader and punch turned off and on under program control. When commanded by the AUX ON character, the TTY service punches paptapes until the AUX OFF character is read or typed in. The AUX OFF character is the last character punched on tape.

When writing programs to output to the terminal paper-tape punch, the user should punch several inches of blank tape before the AUX OFF character is transmitted. This last character may then be torn off and discarded.

11.7 PSEUDO-TTYS (PTYs)

The device mnemonic is PTY0,PTY1,...,PTYn. (The number of pseudo-TTYs is specified when the monitor is generated for a specific installation.) The buffer size is 238 (208 data) words.

11.7.1 Concepts

Each job in the DECsystem-10 is usually initiated by a user at a physical terminal. Except in the case of a DETACH operation, the job remains under the control of the user's terminal until it is terminated by either the KJOB command or the LOGOUT monitor call. For each physical terminal there is a block of core in the monitor, containing information about the physical terminal and including two buffers as the link between the physical terminal and the job. It is through these buffers that the terminal sends input to the job, and the job returns output to the terminal.

Sometimes it is desirable to allow a job in the DECsystem-10 to be initiated by a program instead of by a user. Since a program cannot use a physical terminal in the way a user can, some means must be provided in the monitor for the program to send input to and accept output from the job it is controlling. The monitor provides this capability via the pseudo-TTY (PTY). The PTY is a simulated terminal and is not defined by hardware. Like hardware-defined terminals, each PTY has a block of core associated with it. This block of core is used by the PTY in the same manner as a hardware-defined terminal uses its block of core. Figure 11-1 shows the parallel between a hardware-defined terminal and a software-defined PTY.

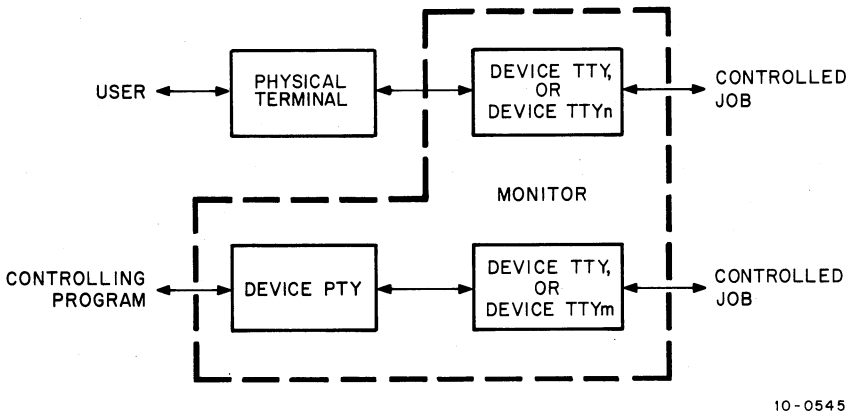


Figure 11-1 Pseudo-TTY

The controlling program, most commonly the batch processor, uses the PTY in the same way a user uses a physical device. It initiates the PTY, inputs characters to and waits for output from the PTY, and closes the PTY using the appropriate programmed operators. The job controlled by the program performs I/O to the PTY as though the PTY were a physical terminal.

A controlled job may go into a loop and not accept any input from its associated buffer; therefore, it is not possible for the controlling program to simply rely on waiting for activity in the controlled job. A controlling program may also wish to drive more than one controlled job, and be able to respond to any of these jobs; therefore, the controlling program cannot wait for any particular PTY. For these two reasons, the PTY differs from other devices in that it is never in an I/O wait state. Timing is accomplished by the HIBER monitor call and the status bits of the PTY.

11.7.2 The HIBER Monitor Call

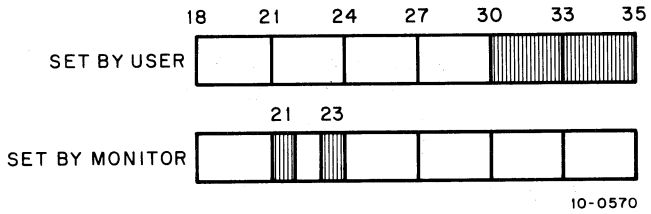
The HIBER call allows the controlling program to temporarily suspend its operation until either there is activity in the controlled job or the specified amount of sleep time runs out, whichever occurs first. If bit 12 in the AC is set in the HIBER call, any PTY activity since the last HIBER causes the controlling program to be awakened. If no PTY activity occurs before the limit of sleep time is reached, the controlling program is activated, and it checks the controlled job's run time or other criteria to determine whether the job should be interrupted. If the job should be interrupted, the controlling program may output two control-C characters to stop the job. (A timesharing user stops a running job in the same way). If the job should not be interrupted, the controlling program should repeat the HIBER. Refer to Section 4.1.3.2.

If bit 12 in AC is not set, unnecessary delays might result if activity occurred on a PTY while the controlling job was sleeping. To avoid these delays, a check is made when a PTY status bit changes to determine if the controlling program is in a sleep. If it is, the sleep time is cleared so the controlling program can service the PTY.

11.7.3 File Status

The file status of the pseudo-TTY is shown below.

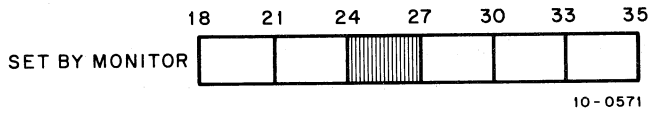
Standard Bits



Bit 21 - IO.BKT

Bit 23 - IO.ACT Device is active

Device Dependent Bits



Bit 24 - IO.PTY Job is in a TTY input wait. The controlling job should perform an OUTPUT to the PTY.

Bit 25 - IO.PTO The TTY buffer has output to be read by an INPUT from the PTY.

Bit 26 - IO.PTM Any characters typed into the TTY buffer (by OUTPUT to the PTY) are read by the monitor command decoder instead of by the controlled job (i.e., the controlled job is in monitor code

11.7.4 Special Monitor Calls

11.7.4.1 OUT, OUTPUT – The first OUTPUT operation after an INIT or OPEN causes the special actions of the RELEASE monitor call.

1. Characters from the controlling program's buffer ring are placed in the input buffer of the TTY linked to the PTY.

2. The IO.PTI bit is cleared.

3. The IO.PTM ibit is set or cleared as determined by the state of the TTY.

The following are exceptions to the normal output actions;

1. NULLS (ASCII 000) are discarded.

2. If more OUTPUTs are performed than are accepted by the controlled job and if the limit on this excess is exceeded, the IO.BKT bit is set and the remainder of the controlling program's buffer is discarded.

3. Lower case characters sent to the controlling job are translated to upper case if the appropriate bit in the TTY is set.

11.7.4.2 IN, INPUT – Characters are read from the output buffer of the TTY and are placed in the buffer ring of the controlling program. If there are no characters to read, an empty buffer is returned. INPUT does not cause a WAIT.

All the available characters are passed to the controlling program. If there are more characters to read than can fit in the buffer of the controlling program, the IO.PTO bit remains set and another INPUT should be done. If the output buffer of the TTY is exhausted by INPUT the IO.PTO bit is cleared.

11.7.4.3 RELEASE Monitor Call – RELEASE causes the following special actions:

1. Any characters in the output buffer of TTY are discarded.

2. If the controlled job is still attached to TTY, it is detached.

3. The PTY is disassociated from the software channel.

NOTE

Haphazard use of the PTY and subsequent RELEASE operations may leave detached jobs tying up core and other system resources.

11.7.4.4 JOBSTS Monitor Call – JOBSTS provides status information about device TTY and/or the controlled job in order to allow complete and accurate checking of a controlled job. The calling sequence is

```
{MOVEI ac, channel    }
{MOVNI ac, job-number }
 JOBSTS ac,
    error return
 normal return
```

AC contains a number *n* specifying the job and/or the TTY to be checked. If n is from 0 to 17, the specified TTY and job are those currently INITed on the user's channel *n*. If *n* is negative, the job to be checked is job number (-*n*).

The error return is given if one of the following is true:

1. the call has not been implemented. If this is the case, check the I/O status word,
2. *n* is out of range, or
3. there is no PTY INITed on channel *n*.

Otherwise, the normal return is given and AC contains the following status information.

Table 11-7
JOB Status Bits

Name	Bit	Explanation
JB.UJA	Bit 0 = 1	Job number is assigned.
JB.ULI	Bit 1 = 1	Job is logged in.
JB.UML	Bit 2 = 1	TTY is at monitor level.
JB.UOA	Bit 3 = 1	TTY output is available.
JB.UDI	Bit 4 = 1	TTY is at user level and in input wait, or TTY is at monitor level and can accept a command. In other words, there is no command awaiting decoding or being delayed, the job is not running, and the job is not stopped waiting for operator device action.
JB.UJC	Bit 5 = 1	JACCT is set. In particular, ↑C ↑C will not work.
JB.UJN	Bits 18 - 35	Job number being checked or 0 if no job number is assigned.

11.7.4.5 CTLJOB Monitor Call – CTLJOB (CALLI 65) is used to determine the job number of the program (job) that is controlling the specified job, if any. Its calling sequence is:

```
MOVE   ac, job number
CTLJOB ac,
    error return
normal return
```

On a normal return, AC contains the job number of the program (job) that is controlling the controlled job. If AC = 1, the specified job is not being controlled via a PTY.

An error return is given if the call has not be implemented or the job number is too large.

CHAPTER 12

I/O PROGRAMMING WITH UNIT RECORD DEVICES

This chapter explains unique features of each of the following standard non-directory I/O devices.

the card punch	refer to 12.1
the card reader	refer to 12.2
the display unit	refer to 12.3
the line printer	refer to 12.4
the paper tape punch	refer to 12.5
the paper tape reader	refer to 12.6
the plotter	refer to 12.7

Each device accepts the monitor calls described in Chapter 7, unless otherwise indicated. Table 12-1 summarizes the characteristics of these devices. Buffer sizes are given in octal and include three bookkeeping words. The user program may obtain the physical characteristics associated with any device by using the DEVCHR monitor call (refer to Section 7.5.2).

Table 12-1
Summary of Some Non-Directory Devices

Device Name	Physical Name	Controller Number	Unit Number	Monitor Calls	Data Modes	Buffer Sizes[1]
Card Punch	CDP	–	CP10A	OUTPUT OUT	A,AL,I, IB,B	35
Card Reader	CDR CDR1	–	CR10-D CR10-E CR10-F	INPUT IN	A,AL,I, IB,B,SI	36
Console TTY	CTY	–	LA36 LA30 LT33A/B LT37AC LT35A VT06/05/50 GT40	INPUT IN OUTPUT OUT	A,AL,I	23
Display	DIS	–	VR30 VP10 340B 30	INPUT OUTPUT IN OUT	ID	Dump Only
Line Printer	LPT LPT1 LPT2	LP10-F LP10-H	LSP10LA LSP10-LB LP10-FA LP10-FB LP10-FC LP10-FD LP10-HA LP10-HB LP10-HC LP10-HD	OUTPUT OUT	A,AL,I	37
Paper-Tape Punch	PTP	–	PC09	OUTPUT OUT	A,AL,I IB,B	43
Paper-Tape Reader	PTR	–	PC04	INPUT IN	A,AL,I IB,B	43
Plotter	PLT PLT1	XY10	XY10-A XY10-B	OUTPUT OUT	A,AL,I IB,B	46

[1] Buffer sizes are subject to change and should be obtained via the DEVSIZ monitor call.

12.1 THE CARD PUNCH (CDP)

The header card is the first card of an ASCII file and identifies the card code used (refer to Appendix C). This card is not punched for data modes other than ASCII. The header card has the same punches in all columns. This punch depends on the card code used; for example, in 026, the header card has 12-2-4-8 punched in columns 1 through 80.

The end-of-file (EOF) card is the last card of each output file. This card is punched for all data modes; it has a 12-11-0-1-6-7-8-9 punch in columns 1 through 80.

12.1.1 Data Modes and Buffer Zones

The buffer sizes for the card punch are data-mode dependent.

ASCII code 0 Buffer Size:23_8 (80 7-bit ASCII characters)	ASCII characters are converted to card codes and punched (up to 80 characters per card). Tabs are simulated by punching from 1 to 8 blank columns; form-feeds and carriage returns are ignored. Line feeds cause a card to be punched. All other non-translatable ASCII characters cause a question mark to be punched. Card can be split between buffers. Attempting to punch more than 80 columns/card causes the error bit (IO.BKT) to set in the file status word. The CLOSE Call will punch the last partial card and punch an EOF card. Cards are normally punched with DEC026 card codes. If bit 29(100_8) of the status word is on (from an INIT, OPEN, or SETSTS), cards will be punched with DEC029 codes (refer to Appendix C). The first card of any file (the header card) indicates the card code used (12-0-2-4-6-8 punched in column 1 for ANSI card codes; 12-2-4-8 punched in column 1 for DEC026 card codes).
ASCII Line Code 1 Buffer Size:23_8 (80 7-bit ASCII characters)	Same as ASCII mode.
IMAGE Code 10 Buffer Size:36_8 (80 12-bit bytes) IMAGE BINARY Code 13 Buffer Size:36_8	Each buffer contains 27_{10} words, each of which contain three 12-bit bytes. Each byte corresponds to one card column. Since there is room for 81 columns in the buffer (3 x 27) and there are only 80 columns on a card, the last word contains 2 bytes of data; the third byte is thrown away. If the byte size is set by the user program to be 12-bit bytes (the monitor normally set up 36-bit bytes), the program must skip the last byte in the buffer. Image binary causes exactly one card to be punched for each buffer output. A program should not force an output every 80 columns since, if the program is in spooled mode, it will waste a large amount of disk space. The CLOSE will punch the last partial card and punch an EOF card.
BINARY Code 14 Buffer Size:35_8	Column 1 contains the word count in rows 12 through 3. A 7-9 punch is in column 1. Column 2 contains a checksum as described for the paper-tape reader (refer to paragraph 11.7.1.5); columns 3 through 80 contain up to 26 data words, 3 columns per word. Binary causes exactly one card to be punched for each output. The CLOSE call punches the last partial card and then punches an EOF card.

12.1.2 Monitor Calls

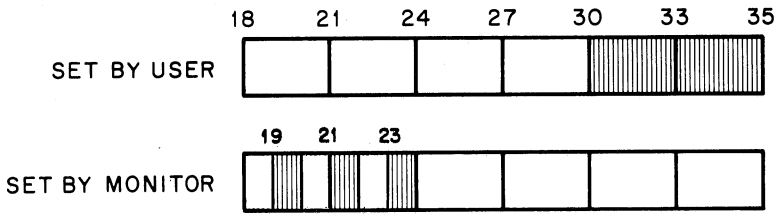
Following a CLOSE, an EOF card is punched. Columns 2 through 80 of the header card and the EOF card contains the same punches that appear in column 1 for each file identification. These punches are ignored by the card reader service routine.

After each interrupt, the card punch stores the result of a CONI in the DEVSTS word of the device data block. The DEVSTS call is used to return the contents of the DEVSTS word to the user.

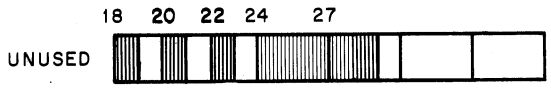
12.1.3 File Status

The file status of the card punch is as follows.

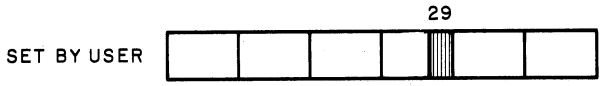
Standard Bits



Bit 19 – IO.DER	Punch error
Bit 21 – IO.BKT	Reached end-of-card with data remaining in buffer
Bit 23 – IO.ACT	Device is active



Device Dependent Bits



Bit 29 – IO.D29	If 1, punch DEC029 card codes in ASCII mode If 0, punch DEC026 codes

12.2 THE CARD READER (CDR)

For ASCII input, a header card can be the first card of the file and identifies the card code used (026 or ANSI standard). The header card is used only when changing from (or back to) installation standard on ASCII input. The header card must not be present with any other data modes; if present, the header card is treated as an incorrect format or read as data. Refer to Appendix C for the card codes.

An EOF card (end-of-file) has a 12-11-0-1-6-7-8-9 punched in columns 1 through 80. Columns 2 through 80 are ignored. The EOF card has the same effect as the EOF key on the card reader. This key must be depressed or the end-of-file card must be present at the end of each input file for all data modes.

To be compatible with PDP-11 operating systems, the DECsystem-10 card reader service accepts several other header card codes and EOF cards.

Only column 1 is looked at; columns 2 through 80 are ignored.

	Punched by DECsystem-10	Also Accepted
ANSI	12-0-2-4-6-8	12-11-8-9

12.2.1 Data Modes

12.2.1.1 ASCII, Octal Code 0 – All 80 columns of each card are read and translated to 7-bit ASCII code. Blank columns are translated to spaces. At the end of each card a carriage return/line feed is appended. As many complete cards as can fit are placed in the input buffer, but cards are not split between the buffers. Using the standard-sized buffer, only one card is placed in each buffer.

Cards are normally translated as ANSI card codes. If an 026 header card is encountered, any following cards are translated as 026 codes (refer to Appendix C) until the 026 conversion mode is turned off. The 026 is turned off by a RELEASE command or by an ANSI header card. Columns 2 through 80 of both of these cards are ignored.

12.2.1.2 ASCII Line, Octal Code 1 – This mode is the same as ASCII mode.

12.2.1.3 Image, Octal Code 10 – All 12 punches in all 80 columns are packed into the buffer as 12-bit bytes. The first byte is in column 1. The last word of the buffer contains columns 79 and 80 as the left and middle bytes, respectively. The EOF button is processed as in ASCII mode. Cards are not split between two buffers.

12.2.1.4 Image Binary, Octal Code 13 – This mode is the same as Image.

12.2.1.5 Binary, Octal Code 14 – Card column 1 must contain a 7-9 punch to verify that the card is in binary format. Column 1 also contains the word count in rows 12 through 3. The absence of the 7-9 punch results in setting the IO.IMP (bit 18 of the status word) flag in the card reader status word. Card column 2 must contain a 12-bit checksum as described for the paper-tape binary format. Columns 3 through 80 contain binary data, 3 columns per word for up to 26 words. Cards are not split between two buffers. The EOF button is processed the same as in ASCII mode.

12.2.1.6 Super-Image, Octal Code 110_2 – Super-image mode may be initialized by setting bit 29 of the card reader's IOS word. This mode causes the 36 bits read from the I/O bus to the BLKI'd directly to the user's buffer. For this mode, the default size of the input buffer is 81_{10} words (80_{10} data words).

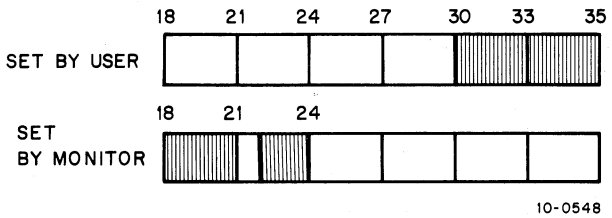
12.2.2 Monitor Calls

The card reader, after each interrupt, stores the result of a CONI in the DEVSTS word in the device data block. The DEVSTS call is used to return the contents of the DEVSTS word to the user.

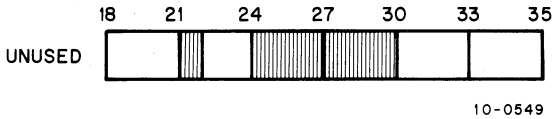
12.2.3 File Status

The file status of the card reader is shown below.

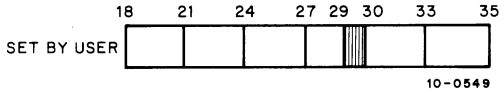
Standard Bits



Bit 18 = IO.IMP	7-9 punch absent in column 1 of a presumed binary card. The card reader is stopped.
Bit 19 = IO.DER	Photocell error, card motion error, data missed. The card reader is stopped.
Bit 20 = IO.DTE	Computed checksum is not equal to checksum read on binary card. The card reader is stopped.
Bit 22 = IO.EOF	EOF card read or EOF button pressed.
Bit 23 = IO.ACT	Device is active.



Device-Dependent Bits



Bit 29 = IO.SIM	Super-image mode.

12.3 DISPLAY WITH LIGHT PEN

The device mnemonic is DIS; there is no buffer because the display uses device-dependent dump mode only.

12.3.1 Data Modes

For IMAGE DUMP, Octal Code 15, an arbitrary length in the user area may be displayed on the scope. The command list format is as described in Chapter 7 with the addition for the Type 30, VR30 and VP10 display, that, if RH = 0, and LH = 0, then LH specifies the intensity for the following data (4 to 13).

12.3.2 Background

During timesharing on a heavily-loaded system, the monitor service routine for the Type 30, VR30, and VP10 guarantees a flicker-free picture on the display if the job is locked in core. To maintain this picture, the picture data must be available for the display at least every two jiffies. If the system is lightly loaded, it is not necessary to keep the job in core. When the job is swapped, a minimum amount of flicker may occur, but the job has high priority to the swapped-in again.

12.3.3 Display Monitor Calls

The I/O Monitor calls for both displays operate as follows:

```
INIT channel, 15          ;MODE 15 ONLY
SIXBIT/device/            ;DEVICE NAME
  0                       ;NO BUFFERS USED
      error return        ;DISPLAY NOT AVAILABLE
normal return
CLOSE channel,            ;STOPS DISPLAY AND
      or
RELEASE channel,          ;RELEASES DEVICE AS
                          ;DESCRIBED IN CHAPTER 7
```

12.3.3.1 INPUT *channel, addr* – If a light pen hit has been detected since the last INPUT, *addr* is set to the location of last light pen hit. If no light pen hit has been detected since last INPUT command, then *addr* is set to -1.

12.3.3.2 OUTPUT *channel, addr* – *addr* specifies the first location of a table of pointers. This table is composed of pointers with the following format:

0 – 17	18 – 35
LH	RH

For the Type 30, VR30 and VP10 Display:

If LH = 0 and RH = 0,	then this is the end of the command list.
If LH = 0 and RH = 0,	then LH is the desired intensity for the following data or commands. The intensity ranges from 4 to 13, where 4 is the dimmest and 13 is the brightest.
If LH = 0 and RH = 0,	then RH is the address of the next pointer. Successive pointers are interpreted beginning at RH.
If LH = 0 and RH = 0,	then -LH words beginning at address RH+1 are output as data to the display. The format of the data word is the following:

0 – 7	8 – 17	18 – 25	26 – 35
	Y-COORD		X-COORD

For the Type 340B Display:

If RH = 0	then this is the end of the command list.
If LH = 0 and RH = 0,	then RH is the address of the next pointer. Successive pointers are interpreted beginning at RH.
If LH = 0 and RH = 0,	then -LH words beginning at address RH+1 are output as data to the display. The format of the data word is described in the Precision Incremental CRT Display Type 340 Maintenance Manual.

An example of a valid pointer list for the VR-30 display is:

```
            OUTPUT  D,LIST      ;OUTPUT DATA
                                ;POINTED TO BY LIST
LIST;       XWD     5,0         ;INTENSITY 5 (DIM)
            IOWD    1,A         ;PLOT A
            IOWD    5,SUBP1     ;PLOT SUBPICTURE 1
            XWD     13,0        ;INTENSITY 13 (BRIGHT)
            IOWD    1,C         ;PLOT C
            IOWD    2,SUBP2     ;PLOT SUBPICTURE 2
            XWD     0,LIST1     ;TRANSFER TO LIST 1

LIST1:      XWD     10,0        ;INTENSITY 10 (NORMAL)
            IOWD    1,B         ;PLOT B
            IOWD    1,D         ;PLOT D
            XWD     0,0         ;END OF COMMAND LIST
            OUTPUT  D,LIST      ;OUTPUT DATA
                                ;POINTED TO BY LIST
A:          XWD     6,6         ;Y=6,     X=6
B:          XWD     70,105      ;Y=70,    X=105
C:          XWD     105,70      ;Y-105,   X=70
D:          XWD     1000,200    ;Y=1000,  X=200

SUBP1:      BLOCK   5           ;SUBPICTURE 1
SUBP2:      BLOCK   2           ;SUBPICTURE 2
```

An example of a valid pointer list for the Type 340B Display is:

```
            OUTPUT  D, LIST     ;OUTPUT DATA POINTER
                                ;TO BY POINTER IN LIST

LIST:       IOWD      1,A       ;SET STARTING POINT TO (6,6)
            IOWD    5,SUBP1     ;DRAW A CIRCLE
            IOWD    1,C         ;SET STARTING POINT TO
                                (70,105)
            IOWD    5,SUBP1     ;DRAW A CIRCLE
            IOWD    1,B         ;SET STARTING POINT TO
                                (105,70)
            IOWD    2,SUBP2     ;DRAW A TRIANGLE
            IOWD    0,LIST1     ;TRANSFER TO LIST1

LIST1:      IOWD    1,D         ;SET STARTING POINT TO
                                ;(100,-200)
            IOWD    5,SUBP1     ;DRAW A CIRCLE
            IOWD    1,A         ;SET STARTING POINT TO (6,6)
            IOWD    2,SUBP2     ;DRAW A TRIANGLE
            XWD     0,0         ;STOP
```

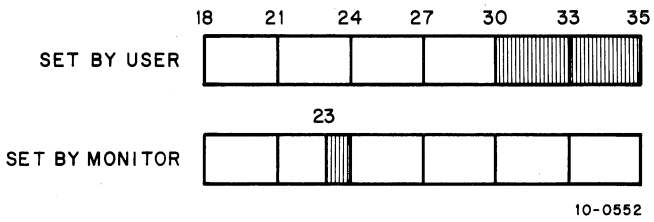
```
A:          X=6       Y=6
B:          X=105     Y=70
C:          X=70      Y=105
D:          X=1000    Y=-200

SUBP1;      BLOCK     5          ;DRAW A CIRCLE
SUBP2:      BLOCK     2          ;DRAW A TRIANGLE
```

The example shows the flexibility of this format. The user can display a subpicture by setting up a pointer. He can also display the same subpicture in many different places by setting up pointers to the subpicture, each preceded by a pointer to commands for the display to reset its coordinates.

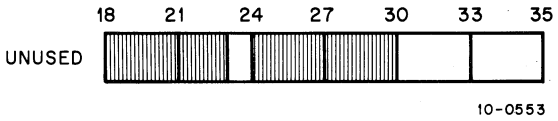
12.3.4 File Status

The file status of the display is shown below.

Standard Bits



Bit 23 = IO.ACT Device is active



Device-Dependent Bits – None

12.4 LINE PRINTER

The device mnemonic is LPT;the buffer size is 37_8 (36_8 data) words.

12.4.1 Data Modes

12.4.1.1 ASCII. Octal Code 0 – ASCII characters are transmitted to the line printer exactly as they appear in the buffer. Refer to the DECsystem-10 System Reference Manual for a list of the vertical spacing characters.

12.4.1.2 ASCII Line, Octal Code 1 – This mode is exactly the same as ASCII and is included for programming convenience. All format control must be performed by the user's program; this includes placing a LINE-FEED at the end of each line (note that a carriage return is not necessary).

12.4.1.3 Image, Octal Code 10 – This mode is the same as ASCII mode.

12.4.2 Monitor Calls

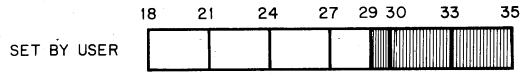
The first output programmed operator of a file and the CLOSE at the end of a file cause an extra form-feed to be printed to keep files separated if IO.SFF is not set.

After each interrupt, the line printer stores the results of a CONI in the DEVSTS word of the device data block. The DEVSTS monitor call is used to return the contents of the DEVSTS word to the user.

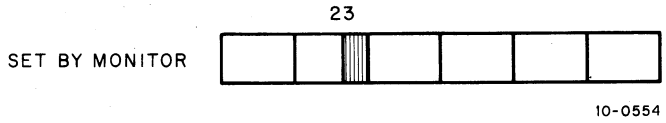
12.4.3 File Status

The file status of the line printer is shown below.

Standard Bits



Bit 29 = IO.SFF Suppress FORM FEEDS on an OPEN or CLOSE.



Bit 23 = IO.ACT Device is active.

Device-Dependent Bits – None

12.5 THE PAPER-TAPE PUNCH

The device mnemonic is PTP; the buffer size is 43_8 (40_8 data) words.

12.5.1 Data Modes

12.5.1.1 ASCII, Octal Code 0 – The eighth hole is punched when necessary in order to make even parity. Tape-feed without the eights hole (000) is inserted after form-feed. A rubout is inserted after each vertical or horizontal tab. Null characters (000) appearing in the buffer are not punched.

12.5.1.2 ASCII Line, Octal Code 1 – The mode is the same as ASCII mode. Format control must be performed by the user's program.

12.5.1.3 Image, Octal Code 10 – Eight-bit characters are punched exactly as they appear in the buffer with no additional processing.

12.5.1.4 Image Binary, Octal Code 13 – Binary words taken from the output buffer are split into six 6-bit bytes and punched with the eighth hole punched in each line. There is no format control or checksumming performed by the I/O routine. Data punched in this mode is read back by the paper-tape reader in the IB mode.

12.5.1.5 Binary, Octal Code 14 – Each bufferful of data is punched as one checksummed binary block as described for the paper-tape reader. Several blank lines are punched after each bufferful for visual clarity.

12.5.2 Monitor Calls

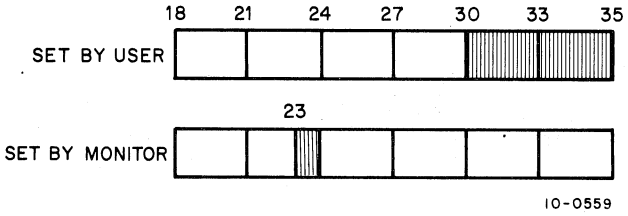
The first output programmed operator of a file causes approximately two fanfolds of blank tape to be punched as leader. Following a CLOSE, an additional fanfold of blank tape is punched as trailer. No EOF character is punched automatically.

After each interrupt, the paper-tape punch stores the results of a CONI in the DEVSTS word of the device data block. The DEVSTS monitor call is used to return the contents of the DEVSTS word to the user.

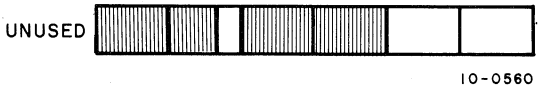
12.5.3 File Status

The file status for the paper-tape punch is shown below.

Standard Bits



Bit 23 – IO.ACT Device is active.



Device Dependent Bits – None.

12.6 THE PAPER-TAPE READER

The device mnemonic is PTR; the buffer size is 43_8 (40_8 data) words.

12.6.1 Data Modes (Input Only)

NOTE

To initialize the paper-tape reader, the input tape must be threaded through the reading mechanism and the FEED button must be depressed momentarily.

12.6.1.1 ASCII, Octal Code 0 – Blank tape (000), RUBOUT (377), and null characters (200) are ignored. All other characters are truncated to seven bits and appear in the buffer. The physical end of the paper tape serves as an EOF, but does not cause a character to appear in the buffer.

12.6.1.2 ASCII Line, Octal Code 1 – Character processing is the same as for ASCII mode. The buffer is terminated by LINE FEED, FORM, or VT.

12.6.1.3 Image, Octal Code 10 – There is no character processing. The buffer is packed with 8-bit characters exactly as read from the input tape. Physical end-of-tape is the EOF indication but does not cause a character to appear in the buffer.

12.6.1.4 Image Binary, Octal Code 13 – Characters not having the eighth hole punched are ignored. Characters are truncated to six bits and packed six to the word without further processing. This mode is useful for reading binary tapes having arbitrary blocking format.

12.6.1.5 Binary, Octal Code 14 – Checksummed binary data is read in the following format. The right half of the first word of each physical block contains the number of data words that follow and the left half contains half a folded checksum. The checksum is formed by adding the data words using 2's complement arithmetic, then splitting the sum into three 12-bit bytes and adding these using 1's complement arithmetic to form a 12-bit checksum. The data eror status flag is set in the status word if the checksum miscompares. Because the checksum and word count appear in the input buffer, the maximum block length is 40. The byte pointer, however, is initialized so as not to pick up the word count and checksum word.

Again, physical end of tape is the EOF indication, but does not result in putting a character in the buffer.

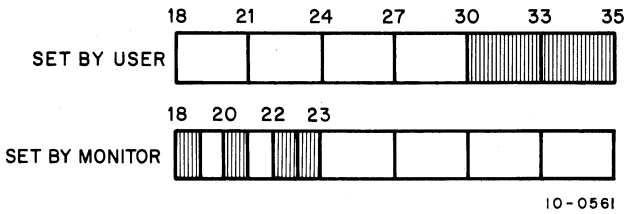
12.6.2 Monitor Calls

After each interrupt, the paper-tape reader stores the results of a CONI in the DEVSTS word of the device data block. The DEVSTS call is used to return the contents of the DEVSTS word to the user.

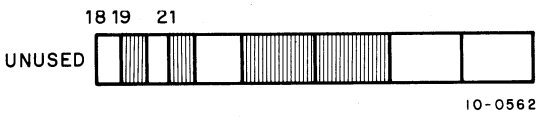
12.6.3 File Status

The file status of the paper-tape reader is shown below.

Standard Bits



Bit 18 – IO.IMP	Binary block is incomplete.
Bit 20 – IO.DTE	Bad checksum in binary mode.
Bit 22 – IO.EOF	Physical end-of-tape is encountered. No character is stored in the buffer.
Bit 23 – IO.ACT	Device is active.



Device Dependent Bits – None.

12.7 PLOTTER

The device mnemonic is PLT; the buffer size is 43_8 (40_8 data) words. The plotter takes 6-bit characters with the bits of each character decoded as follows:

PEN RAISE	PEN LOWER	-X DRUM UP	+X DRUM DOWN	+Y CARRIAGE LEFT	-Y CARRIAGE RIGHT

10-0563

Do not combine PEN RAISE or LOWER with any of the position functions. (For more details on the incremental plotter, refer to the DECsystem-10 System Reference Manual.)

12.7.1 Data Modes

12.7.1.1 ASCII, Octal Code 0 – Five 7-bit characters per word are transmitted to the plotter exactly as they appear in the buffer. The plotter is a 6-bit device; therefore, the leftmost bit of each character is ignored.

12.7.1.2 ASCII Line, Octal Code 1 – This mode is identical to ASCII mode.

12.7.1.3 Image, Octal Code 10 – Six 6-bit characters per word are transmitted to the plotter exactly as they appear in the buffer.

12.7.1.4 Image Binary, Octal Code 13 – This mode is identical to Image mode.

12.7.1.5 Binary, Octal Code 14 – This mode is identical to Image mode.

12.7.2 Monitor Calls

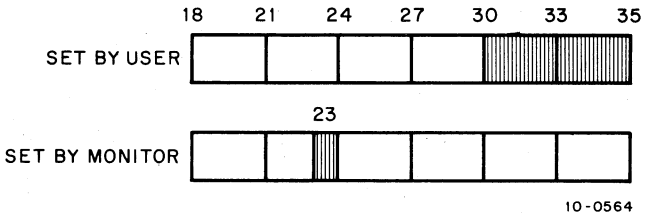
The first OUTPUT operator causes the plotter pen to be lifted from the paper before any user data is sent to the plotter. The CLOSE operator causes the plotter pen to be lifter after all user data is sent to the plotter. These two pen-up commands are the only modifications the monitor makes to the user output file.

After each interrupt, the plotter stores the results of a CONI in the DEVSTS word of the device data block. DEVSTS is used to return the contents of the DEVSTS word to the user.

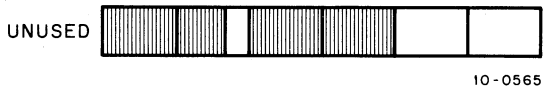
12.7.3 File Status

The file status of the plotter is shown below.

Standard Bits



Bit 23 – IO.ACT Device is active.



Device Dependent Bits – None.

CHAPTER 13
THE MULTIPLEX CHANNEL FEATURE

The MPX channel is a software I/O channel on which an INIT/OPEN has been performed for device MPX:; INIT defines to the monitor a multiplexed channel. Without the MPX channel feature, a program is restricted to referencing sixteen (one for each software I/O channel) simultaneously active devices. An MPX channel connects a large number of devices to one software I/O channel, allowing a program to support a large number of devices simultaneously on one channel. Any job can create an MPX channel, and each job may create as many MPX channels as required within the normal constraints on the maximum number of channels per job.

To each INITed MPX channel the user connect devices that he wishes to control via the MPX channel. He may connect as many devices (in any order and in an arbitrary mix of device types) to a single MPX channel as long as each device connected has the predefined characteristic of being controllable via an MPX channel. (The DEVTYP monitor call may be used to determine whether or not a device can be controlled by an MPX channel.) From a user's point of view, I/O is performed into and out of buffers similar to the buffer ring described in paragraph 7.3.3.1. The buffer ring concept is slightly extended allowing all devices connected to one MPX channel to share the same buffer ring.

The IN and OUT operators are utilized in about the same way as required for devices other than MPX devices. However, the format of buffers and buffer ring headers is modified to provide a unique device I. D. (a universal device index), designating the source/destination of the data in each buffer.

13.1 BUFFER RING EXTENSION

For each MPX channel, one input and one output buffer ring can be defined via INIT/OPEN and INBUF/OUTBUF as described in paragraphs 7.1 and 7.3.2. However, the buffer ring header block is 4-words in length for each device rather than 3 words. INIT/OPEN defines the four-word headers for input and output. INBUF and OUTBUF are then used by the user to create an arbitrary number of buffers for each ring with the buffer header in each buffer appropriately initialized. Figure 13-1 illustrates a 4-word buffer ring header block.

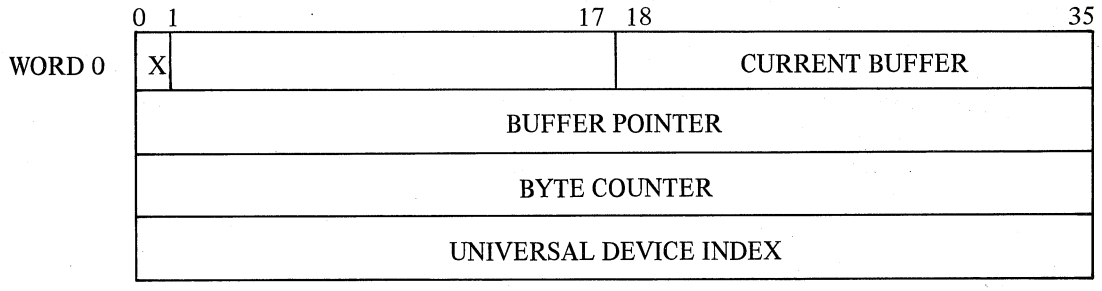


Figure 13-1. MPX Buffer Ring Header Block

The input ring contains buffers chained together in an endless fashion with pointers in each buffer ring header block to the next buffer in the ring. As in the conventional input buffer ring, input data is stored in consecutive buffers in the ring, and data is retrieved in the same order for the user via the IN operator. The MPX channel stores additional information pertaining to the data in each buffer's 4-word header area. The fourth word of the ring header is loaded with a value identifying the specific device which stored the data: the universal device index.

The user must store a value identifying the device for which the data is destined. This value is stored in the fourth word of the buffer ring header before the OUT operator is issued to output the data. With the exception of the fourth word in the buffer ring header block, the MPX channel user can perform I/O operations in a way that is virtually identical to the buffered I/O described in paragraph 7.3.2.

MPX utilizes buffer rings for input; for output the format is slightly modified to form one or more device chains and a free chain.

13.1.1 Device Chains

Device chains are created when an OUT operator causes a buffer of data to be output on a particular device. A control block in the monitor address space maintains pointers to the beginning of the device chain for each device. The chains are linked via the normal buffer link pointer (the right half of the second word of each buffer) and are terminated by a zero pointer value. Using the chain for output allows the monitor to treat each device connected to an MPX channel separately.

After the device service routine empties the buffer it is placed on a free chain available for reuse. The free chain begins with a pointer in the right half of the first word of the ring header. Buffers are linked via the normal pointer in each buffer header area, and the chain is again terminated with a pointer value of zero.

As OUT operators are issued, buffers are removed from the free chain and added to a device chain. The ring header is updated to the next buffer in the free chain for return to the user from the OUT. Buffers on a device chain are returned to the free chain by the monitor when output has been accomplished. A facility exists to allow the user to force an output buffer off a device chain and back to the free chain, when the user wishes to abort the output.

13.2 DATA MODES

All I/O performed on the MPX channel is in buffer I/O mode; that is, I/O is performed to and from buffer rings only. All buffered I/O modes are legal for MPX as long as they are legal for all of the devices connected to MPX at execution time (refer to paragraph 7.3).

13.3 DEVICE IDENTIFICATION

Devices that can be controlled by an MPX channel are identified by the user in one of two ways. Devices are identified by name (logical or physical) or by their universal device index (UDX). An UDX is associated with a device via the CNECT. or IONDX. monitor call.

13.4 MPX MONITOR CALLS

13.4.1 The CNECT. Monitor Call (CALLI 130)

The CNECT. monitor call connects and disconnects individual devices from a particular MPX channel. CNECT. can only be used for devices which can be controlled by an MPX channel, such as terminals, line printers, etc. Its calling sequence is

```
        MOVEI  ac,addr
        CNECT.   ac,
           error return
        normal return

addr:   XWD operation, channel
           SIXBIT/devicename/          ;or UDX
```

where: *addr* points to the two-word argument block.
operation is one of the operation codes listed in Table 13-1.
channel is an INITed MPX channel number.
devicename is the SIXBIT physical, logical or generic name of the device to be connected or disconnected.
UDX is the universal device index for the device to be connected or disconnected.

Table 13-1
CNECT. Operation Codes

Code	Mnemonic	Meaning
1	.CNCCN	Connect the device to an MPX channel.
2	.CNCDC	Equivalent to a CLOSE and disconnect.
3	.CNCDR	Equivalent to a RESET and disconnect.

A device must be connected to an MPX channel before input or output can occur for that device on that MPX channel. One CNECT. call must be issued for each device to be controlled by an MPX channel, and the CNECT. call should be issued after the channel has been INITed and after any desired INBUF or OUTBUF monitor calls have been issued. On a normal return, if a generic name was specified in the argument block, its UDX will be returned in the AC.

On an error return, an error code will be returned in the AC. The possible error codes are listed in Table 13-2.

Table 13-2
CNECT. Error Codes

Code	Name	Meaning
1	CNCNM%	Channel is not OPEN on device MPX:
2	CNCUD%	Device does not exist in the system.
3	CNCCM%	Device cannot be connected to device MPX.
4	CNCNF%	The monitor ran out of core to build the control blocks.
5	CNCNC%	Device is not connected and the requested operation is conditional or unconditional disconnect.
6	CNCNO%	Channel is in some way illegal or not open.
7	CNCII%	An invalid I/O index was specified (UDX).
11	CNCDU%	Device is already assigned, INITed, or connected by this or some other job.
10	CNCUF%	Operation code is invalid (less than 1 or greater than 3).
12	CNCSD%	Device is a spooled device.

13.4.2 The ERLST. Monitor Call (CALLI 132)

The ERLST. monitor call may be used to find out if any devices have encountered a device error, how many devices have encountered errors, and what these device errors are. The user may optionally specify that ERLST. only return the above data for those devices which have not been previously reported during an earlier ERLST. monitor call. Its calling sequence is

```
        MOVEI  ac, addr
        ERLST. ac,
           error return
       normal return

addr:  words , , channel
       number of devices
       UDX1 , , GETSTS-word
       .
       .
       .
```

where: *addr* points to the first word of the device error block.

words is the number of words contained within this block.

channel is the MPX channel associated with the devices which have encountered error conditions.

number of devices is the number of devices associated with the channel which have encountered device error conditions.

UDX1 is the universal device index associated with the first device reported on in this block.

GETSTS-word is the status bits associated with the device identified in the left half of this word.

The monitor performs this device error reporting by returning data in the device error block. The user must first have declared space for this block by using the BLOCK pseudo-op. The ERLST. monitor call will fill this block with data until all of the space allocated has been filled. If ERLST. does need more space, a flag will be set on return to indicate to the user that the device error list is incomplete because of lack of space.

On an error return, an error code will be returned in the AC. The possible error codes are listed in Table 13-3.

Table 13-3
ERLST. Error Codes

Code	Mnemonic	Meaning
1	ERLBC%	Bad channel; not in the range 0-17(8).
2	ERLNM%	Channel specified is not an MPX channel.

13.4.3 The SENSE. Monitor Call (CALLI 133)

The SENSE. monitor call will return the status bit settings associated with a device connected to an MPX channel. Its calling sequence is

```
        MOVE    ac, [XWD length,addr1]
        SENSE.  ac,
           error return
        normal return
addr1:  udx
        channel
        SIXBIT/device/
        block,,addr2
addr2:  SIXBIT/name/
        0,,GETSTS bits
        DEVSTS word
```

where: *length* is the length of the argument block pointed to by *addr1*.

addr1 points to an argument block.

addr1: contains the device identification, which can be either the Universal Device Index, the channel number associated with the device, or the SIXBIT name of the device.

block is the length of the status block which will be filled in by the monitor. The size of this block is determined by:

((*length*)/2) * 3=*words in block*

addr2 points to the first word of the status block.

The monitor will return the status bits associated with each specified device. These status bits will be returned in the status block, beginning with *addr2.*

On an error return, an error code will be returned in the AC. The possible error codes are listed in Table 13-4.

Table 13-4
SENSE. Error Codes

Code	Mnemonic	Meaning
1	SNSBD%	A bad device has been specified.

13.4.4 The CLRST. Monitor Call (CALLI 134)

The CLRST. monitor call allows the user to clear the status bits which were returned to the monitor as a result of the SENSE. monitor call. By clearing the status bits associated with a device, that device may be continued after a device error condition has occurred. The calling sequence for CLRST. is

```
        MOVE    ac, [XWD n, addr]
        CLRST.  ac,
           error return
        normal return

addr:   { SIXBIT/device/ }
        { channel        }
        { udx            }
addr+1: 0,,setsts value
```

where: *n* is the length of the argument block pointed to by addr.
addr points to an argument block containing one or more two-word entries.
device is the logical/physical name of a device.
channel is the software I/O channel number associated with the device.
udx is the universal device index associated with an MPX device.
setsts value specifies those bits which are to be cleared.

On an error return, an error code will be returned in the AC. The possible error codes are listed in Table 13-5.

Table 13-5
CLRST. Error Codes

Code	Mnemonic	Meaning
1	CLRID%	An illegal device has been specified.
2	CLRNO%	The device specified belongs to another job.

13.4.5 The IONDX. Monitor Call (CALLI 127)

The IONDX. monitor call return the Universal Device Index associated with an MPX device. Its calling sequence is

```
{ MOVE    ac, [SIXBIT/device/] }
{ MOVEI   ac, channel          }
  IONDX.  ac,
     error return
  normal return
```

where: *device* is the logical/physical name of the device for which its UDX is desired.
channel is the software I/O channel number associated with the device.

The error return is taken on the following conditions:

1. When the call has not been implemented,
2. When the device specified does not exist, and
3. When SIXBIT/MPX/ is specified as the device name.

CHAPTER 14
REAL-TIME PROGRAMMING

14.1 THE RTTRP MONITOR CALL (CALLI 57)

The real-time trapping monitor call (RTTRP) can be set by a timesharing user

1. to dynamically connect real-time devices to the priority interrupt system,
2. to respond to these devices at interrupt level,
3. to remove the devices from the interrupt system,
4. and to remove the devices from the interrupt system,
5. to change the priority interrupt level on which the devices are associated.

The RTTRP monitor call can be called from UUO level or from interrupt level. This is a privileged monitor call that requires the calling job to have real-time privileges (granted by LOGIN) and the job be locked in core (accomplished via the LOCK monitor call). These real-time privileges are assigned by the system administrator and are obtained by the monitor from ACCT.SYS. The privilege bits required for using the RTTRP monitor call are

JP.LCK (bit 14) which allows the job to be locked in core, and
JP.RTT (bit 13) which allows the RTTRP monitor call to be executed.

NOTE

Improper use of features of the RTTRP monitor call can cause the system to fail in a number of ways. Since design goals of this monitor call were to give the user as much flexibility as possible, some system integrity had to be sacrificed. The most common errors are protected against since user programs run in user mode with all ACs saved. It is recommended that no debugging of real-time programs be done when system integrity is important. However, once these programs are debugged, they can run simultaneously with batch and timesharing programs.

Real-time jobs control devices one of two ways:

1. block mode, or
2. single mode.

In block mode, an entire block of data is read before the user's interrupt program is run. In single mode, the user's interrupt program is run every time the device interrupts.

Furthermore, there are two types of block modes: fast block mode and normal block mode. These differ in response time only. The response time provided to read a block of data in fast block mode (assuming that nothing else is running on the system) is 6.5 microseconds per word, and in normal block mode it is 14.6 microseconds per word. (The CPU time to complete each data transfer.)

In all modes, the response time measured from the receipt of the real-time device interrupt to the start off the user control program is 100 microseconds.

The RTTRP monitor call allows a real-time job to either put a BLKI or BLKO instruction directly on a priority interrupt level (block mode) or add a device to the front of the monitor priority interrupt channel CONSO skip chain (single mode). Since the BLKI and BLKO instructions are executed in exec mode, a KI-10/KL-10 system requires that the job be mapped in exec virtual memory, in addition to being locked (via the LOCK monitor call).

When an interrupt occurs from the real-time device in single mode or at the end of a block of data in block mode, the monitor will save the current state of the machine, set the new user virtual memory and APR clock flags, and trap to the user's interrupt routine. The user services his device and returns control to the monitor to restore the previous state of the machine and to dismiss the interrupt.

In fast block mode, the monitor places the BLKI or BLKO instruction directly in the priority interrupt trap location, followed by a JSR to the context switcher. This action requires that the priority interrupt channel be dedicated to the real-time job during any transfers. In normal block mode, the monitor places the BLKI or BLKO instruction in the CONSO skip chain.

Any number of real-time devices using either single mode or normal mode can be placed on any available priority interrupt channel. The average extra overhead for each real-time device on the same channel is 5.5 microseconds per interrupt.

The calling sequence for the RTTRP monitor call is

```
MOVEI ac,block
RTTRP ac,
   error return
normal return
```

where: *block* points to the RTTRP data block, which is different depending on whether single mode or block mode is used.

On an error return, an error code is returned in the AC. The possible error codes are listed in Table 14-1. The error return will not be taken until RTTRP scans the entire data block to find as many errors as possible.

On a normal return, the job is given user IOT privileges. These privileges allow the user to execute all restricted instructions including the necessary I/O instructions to control his device. The IOT privileges must be used with caution because improper use of the I/O instructions could halt the system (i.e., HALT on the KA10; CONO APR, 0; DATAO APR, 0; CONO PI, 0 on the KA10, KI10, and KL-10; and CONO PAG, 0 or DATA PAG, 0 on the KI10 and KL10). Note that a user can obtain just the user IOT privileges by issuing the RTTRP monitor call with PICHL equal to 0.

In single mode, the data block appears as follows:

```
RTBLK:  XWD     PICHL,TRPADR    ;PI CHANNEL (1-6) AND TRAP ADDRESS.
        EXP     APRTRP          ;APR TRAP ADDRESS.
        CONSO   DEV,BITS        ;CONSO CHAIN INSTRUCTION.
        0                       ;NO BLKI/BLKO INSTRUCTION.
```

where: *trpadr* is the location trapped to by the real-time interrupt (JRST TRPADDR). Before the trap occurs, all ACs are saved by the monitor and can be overridden without concern for their contents. *pichl* is the priority interrupt level on which the device is to be placed. Levels 1 through 6 are legal depending on the system configuration. If pichl is 0, all occurrences of the device whose device code is specified in the CONSO instruction are removed from all levels. When a device is placed on a priority interrupt level, normally all other occurrences of the device on any priority interrupt level are removed. If the user desired the same device on more than one priority interrupt level simultaneously

Table 14-1
RTTRP Monitor Call Error Codes

Code	Value	Mnemonic	Meaning
Bit 24=1	4000	RTJNP%	Job is not privileged.
Bit 25=1	2000	RTNC0%	This call is not runnable on CPU0.
Bit 26=1	1000	RTDIU%	Specified device is already in use by another job.
Bit 27=1	400	RTIAU%	An illegal AC was used during the RTTRP monitor call at interrupt level.
Bit 28=1	200	RTJNL%	Job was not locked in core when RTTRP call was executed.
Bit 29=1	100	RTSLE%	System limit for real-time devices has been exceeded.
Bit 30=1	40	RTILF%	Illegal format for CONSO, BLKO, or BLKI instructions.
Bit 31=1	20	RTPWI%	The block address or pointer word was specified in an incorrect format.
Bit 32=1	10	RTEAB%	The specified error address is out of bounds.
Bit 33=1	4	RTTAB%	Specified trap address is out of bounds.
Bit 34=1	2	RTPNB%	Specified PI channel is not currently available for BLKIs/BLKOs.
Bit 35=1	1	RTPNA%	PI channel not available.

(i.e., a data level and an error level), he can issue the RTTRP call with pichl negative. This indicates to the system that any other occurrence of this device (on any priority level) is not to be removed. Note that this addition to a priority interrupt level counts as a real-time device, occupying one of the possible real-time device slots.

aprtrp is the trap location for all APR traps. When an APR trap occurs, the monitor simulates a JSR APRTRP. The user gains control from an APR trap on the same priority level that his real-time device is on. The monitor always traps to the user program on illegal memory references, non-existent memory references, and push-down list overflows. This allows the user to properly turn off his real-time device if needed. The monitor also traps on the conditions specified in the APRENB monitor call (refer to Chapter 5). No APR errors that cause an interrupt are detected if the interrupt routine is on a priority interrupt level higher than or equal to the APR interrupt level.

dev is the real-time device code.

bits is the bit mask of all interrupt bits of the real-time device and must not contain any other bits. If the user desires control of this bit mask from his user area, he may specify one level of indirection in the CONSO instruction (no indexing is allowed) i.e., CONSO dev, @mask (where mask is the location in the user area of the bit mask). The mask must not have any bits set in the indirect or index fields on the instruction.

The data block for fast block mode is as follows:

```
RTBLK:  XWD PIXHL,TRADR     ;PI AND TRAP ADDRESS WHEN BLKO DONE.
        EXP APRTRP          ;APR TRAP ADDRESS.
        BLKO DEV,BLKADR     ;BLKI OR BLKO INSTRUCTION
        0                   ;BLKADR POINTS TO THE IOWD OF BLOCK TO BE SENT.
```

where: *pichl, trpadr,* and *aprtp* are the same as above.
blkadr is the address in the user's area of the BLKI/BLKO pointer word. Before returning to the user, the monitor adds the proper relocation factor to the right half of the pointer word. Data can only be read into the low segment above the protected job area, i.e., above location 114.

Since the pointer word is in the user's area, the user can set up a new pointer word when the word count goes to 0 at interrupt level. This allows fast switching between buffers. When the user desires to set up his own pointer word, the right half of the word must be set as an exec virtual instead of a user virtual address. The jobs relocation value is returned from both the LOCK monitor call and the first RTTRP monitor call executed for setting the BLKI/BLKO instruction.

If this pointer word does not contain a legal address, a portion of the system might be overwritten. A check should be made to determine if the negative word count in the left half of the pointer word is too large. If the word count extends beyond the user's own area, the device may cause a non-existent memory interrupt, or may overwrite a timehsaring job. If all of the above precautions are taken, this method of setting up the pointer word is much faster and more flexible than issuing the RTTRP monitor call at interrupt level.

The data block for normal block mode is as follows:

```
RTBLK:  XWD PICHL,TRPADR    ;CHANNEL AND TRAP ADDRESS.
        EXP APRTRP          ;APR TRAP ADDRESS
        CONSO DEV,BITS      ;CONTROL BIT MASK FROM USER AREA
        BLKI DEV,BLKADR     ;BLKI INSTRUCTION
```

On a multiprocessor system, the real-time trap monitor call applies only to the processor specified by the job's CPU specification (refer to the SET CPU command description in DECsystem-10 OPERATING SYSTEM COMMANDS, or the SETUUO monitor call description in Chapter 4). If the specification indicates more than one processor, the specification is changed to indicate CPU0. Note that the priority interrupt channel (pichl) and processor traps (aprtrp) are only for the indicated CPU.

14.1.1 Interrupt Level Use Of RTTRP

The format of the RTTRP monitor call at interrupt level is similar to the format at user level except for two restrictions:

1. AC16 and AC17 cannot be used in the monitor call (i.e., CALLI 16,57 is illegal at interrupt level).
2. All ACs are overwritten when the monitor call is executed at interrupt level. Therefore, the user must save any desired ACs before issuing the RTTRP monitor call. This restriction is used to save time at interrupt level.

NOTE

If an interrupt level routine executes a RTTRP monitor call that affects the device currently being serviced, no additional calls of any kind (including RTTRP and WAKE monitor calls) can be executed during the reaminder of the interrupt. At this point, any subsequent call dismisses the interrupt.

14.1.2 Restrictions

Devices may be chained onto any priority interrupt channel that is not used for BLKI/BLKO instructions by the system or by other real-time users using fast block mode. This includes the APR channel. Normally, priority interrupt levels 1 and 2 are reserved by the system for magnetic tape and DECtapes. Priority interrupt level 7 is always reserved for the system. Each device must be chained onto a priority interrupt level before the user program issues the CONO device, PIA to set the device onto the interrupt level. Failure to observe this rule or failure to set the device on the same priority interrupt level that was specified in the RTTRP monitor call could hang the system.

If CONSO bit mask is set up and one of the corresponding flags in a device is on, but the device has not been physically put on its proper priority level, a trap may occur to the user's interrupt service routine. This occurs because there is a CONSO skip chain for each priority level, and if another device interrupts whose CONSO instruction is further down the chain than that of the real-time device, the CONSO associated with the real-time device is executed.

If one of the hardware device flags is set and the corresponding bit in the CONSO bit mask is set, the CONSO skips and a trap occurs to the user's program even though the real-time device was not causing the interrupt on that channel. To avoid this situation that user can keep the CONSO bit mask in his user area. This procedure allows the user to chain a device onto the interrupt level, keeping the CONSO bit mask zero until the device is actually put on the proper priority interrupt level with CONSO instruction. This situation never arises if the device flags are turned off until the CONSO device, PIA can be executed.

The user should guard against putting programs on high priority interrupt levels which execute for long periods of time. These programs could cause real-time programs at lower levels to lose data.

The user program must not change any locations in the protected job data area (locations 20-114, refer to Chapter 3), because the user is running at interrupt level and full context switching is not performed.

If the user is using the BLKI/BLKO feature, he must restore the BLKI/BLKO pointer word before dismissing any end-of-flck interrupts. This is accomplished with another RTTRP monitor call or by directly modifying the absolute pointer word supplied by the first RTTRP monitor call. Failure to reset the pointer word could cause the device to overwrite all of memory.

14.1.3 Removing Devices from a Priority Interrupt Channel

When pichl equals 0 in the data block, all occurrences of the device specified in the CONSO instruction are removed from the interrupt system. If the user removes a device from a priority interrupt chain, he must also remove the device from the priority interrupt level (CONO dev, 0).

A RESET, EXIT, or RUN monitor call from the timesharing levels removes all devices from the interrupt levels. These calls cause a CONO dev, 0 to be executed before the device is removed. Monitor commands that issue implicit RESETs also remove real-time devices (e.g., R, RUN, GET, CORE 0, SAVE, SSAVE, OSAVE, OSSAVE, NSAVE, and NSSAVE).

14.1.4 Dismissing the Interrupt

The user program always dismisses the interrupt in order to allow the monitor to properly restore the state of the machine. The interrupt may be dismissed with any monitor call other than the RTTRP or WAKE monitor call, or on the KA10, any instruction that traps to absolute location 60. The standard method of dismissing the interrupt is with a UJEWN instruction (op code 100). This instruction gives the fastest possible dismissal.

14.2 THE TRPSET MONITOR CALL (CALLI 25)

The TRPSET feature may be used to guarantee some of the fast response requirements of real-time users. In order to achieve fast response to interrupts, this feature temporarily suspends the running of other jobs during its use. This limits the class of problems to be solved to cases where the user wants to transfer data in short bursts at pre-defined times. Therefore, because data transfers are short, the time during which timesharing is stopped is also short, and the pause probably will not be noticed by the timesharing users.

The TRPSET monitor call allows the user program to gain control of the interrupt location. If the user does not have the TRPSET privileges (JP.TRP, bit 15), the error return will be taken, and the user program will remain in user mode. Timesharing is turned back on. If the user has the TRPSET privileges, the central processor is placed in user I/O mode.

If the AC contains zero, timesharing is turned on if it was turned off. If the left half of the AC is within the range 40 to 57 of the central processor, all other jobs are stopped from being scheduled and the specified PI location (40-57) is patched to trap directly to the user. In this case, the monitor moves the contents of the relative location specified in the right half of the AC, converts the user virtual address to the equivalent exec virtual address, and stores the address in the specified executive PI location. On a KI-10 based system, this requires the user segment access during the interrupt to be locked and mapped contiguously in the exec virtual memory (refer to the LOCK monitor call). If the segment does not meet these requirements the error return is taken.

On a multiprocessor system, the TRPSET monitor call applies to the processor specified by the job's CPU specification (refer to the SET CPU command or the SETUUO monitor call). If the specification indicates only CPU1, the error return is taken if the job is not locked in core. When the specification indicates more than one processor, the specification is changed to indicate CPU0 (the master processor).

The user can set up a priority interrupt trap into his relocated core area. On a normal return, the AC contains the previous contents of the address specified by the left half of the AC, so that the user program may restore the original contents of the PI location when the user is through using this monitor call. If the left half of the AC is not within the range 40 to 57, the error return is taken just as if the user did have the proper privileges.

The calling sequence for the TRPSET monitor call is

```
        MOVE    ac, [XWD n,addr]
        TRPSET  ac,
                error return
        normal return
        .
        .
        .
addr:   JSR     TRAP    ;INSTRUCTION TO BE STORED IN EXEC PI LOCATION AFTER
                        ;RELOCATION ADDRED TO IT.
        TRAP:   0               ;HERE ON INTERRUPT FROM EXEC.
```

The monitor assumes that address contains either a JSR U or BLKI U, where U is a user virtual address. Consequently, the monitor adds a relocation to the contents of location U to make it an absolute IOWD (i.e., an exec virtual address). Therefore, a user should reset the contents of U before every TRPSET call. A RESET monitor call returns the user to normal user mode. To maintain compatibility between KA10-based systems and KI10-/KL10-based systems, the interrupt routine should be executed in exec mode. However, for convenience, the routine can be executed in user mode in order to avoid relocation to exec virtual memory. This is possible on KA10-based systems if care is taken when dismissing the interrupt. On KI10-/KL-10-based systems, if there is a possibility that the interrupt may occur during the job's background processing, the interrupt routine must be executed in exec mode (and therefore must be locked and exec-mapped with the LOCK monitor call). In particular, if the job is executing a monitor call at background level, the user of UJEN at interrupt level may cause an error. On KI10-based systems and KL10-based systems, it is recommended that the TRPSET interrupt routines always be coded to run in exec mode (refer to the RTTRP monitor call for programming techniques).

On KA10-based systems, the interrupt routine can be coded to run in user mode if the following procedure is observed. If the interrupt occurs while some other part of the user's program is running, the user may dismiss from the interrupt routine with a JEN XITINT. However, if the machine is in exec mode, a JEN instruction issued in user mode does not work because of memory relocation. This problem is solved by a call to UJEN (op code 100). This monitor call causes the monitor to dismiss the interrupt from exec mode. In this case, the address field of the UJEN instruction is the user location when the return PC is stored (i.e., UJEN XITINT).

14.2.1 UJEN (Op Code 100)

This op code dismisses a user I/O mode interrupt of one is in progress. If the interrupt is from user mode, a JRST 12 instruction dismisses the interrupt. If the interrupt came from executive mode, however, this operator is used to dismiss the interrupt. The monitor restores all accumulators, and executes JEN @U where user location U contains the program counter as stored by a JSR instruction when the interrupt occurred.

14.2.2 The HPQ Monitor Call (CALLI 71)

The HPQ monitor call is used by privileged users to place their jobs in a high priority scheduler run queue. These queues are always scanned by the scheduler before the normal run queues, and any runnable job in one of these queues is executed before all other jobs in the system.

In addition, these jobs are given preferential access to sharable resources (e.g., shared device controllers). Therefore, real-time associated jobs can receive fast response from the timesharing scheduler.

Jobs in high-priority queues are not examined for swap-out until all other queues have been scanned. If a job in a high-priority queue must be swapped, the lowest priority job is transferred first, and the highest priority job last. If the highest priority job is swapped, then that job is the first to be swapped in for immediate execution. Therefore, in addition to being scanned before all other queues for job execution, the high-priority queues are examined after all other queues for swap-out and before all queues for swap-in.

The HPQ monitor call requires as an argument the high-priority queue number of the queue to be entered. The lowest high-priority queue is 1, and the highest-priority queue is equivalent to the number of queues that the system is built for. The calling sequence for HPQ is

```
MOVE  ac,hpq#
HPQ   ac,
   error return
normal return
```

If the user does not have HPQ privileges, the error return is taken and a -1 is returned in the AC. The privilege bits 6 through 9 in the privilege word (.GTPRV). These four bits specify a number from 0 to 17 octal, which is the highest priority queue attainable by a user.

On a normal return, the job is in the desired high-priority queue. A RESET or EXIT monitor call returns the job to the high-priority queue specified in the last SET HPQ command. A queue number of 0 as an argument places the job back to the timesharing level.

CHAPTER 15
INTER-PROCESS COMMUNICATION FACILITY

The Inter-Process Communication Facility (IPCF) makes it possible for jobs to communicate with one another and with system processes. This communication takes place by sending and receiving packets of information. Each sender and receiver has a Process I.D. (PID) assigned to it for communication identification.

When a process sends a packet to another process, the packet goes into the receiver's input queue, and it remains there until the receiver checks his input queue and retrieves the packet. If the receiver is enabled via the Software Interrupt System (refer to Chapter 5, Section 5.1), an interrupt will occur when a packet is received in the job's input queue.

There is a send packet quota and receive packet quota per job which can be set by each installation per user. The default send packet quota is two per job; the default receive packet quota is five per job. These quotas pertain to outstanding sends and receives only. For example (assuming the default limit of two), if a job sends two packets, no more packets can be sent by that job until one or both of its previously sent packets have first been retrieved by the intended receiver. If the receiver has five (assuming the default limit) packets in its receive queue that it has not yet retrieved, it cannot receive any more packets until it has retrieved at least one of the five waiting packets.

There are two system processes utilized by IPCF: [SYSTEM] INFO and [SYSTEM] IPCC. SYSTEM [INFO] acts as the information center for IPCF and performs several functions related to PIDs and names. [SYSTEM] IPCC is the IPCF controller, and it performs many packet controlling functions. A job can send a packet to both of these system processes.

15.1 PACKETS

A packet is divided into two blocks:

- the packet descriptor block, which is four to six words long, and
- the packet data block, which is the message portion of the packet.

The general format of a packet is represented in Figure 15-1.

Each of the words in the figure is described within the following paragraphs.

15.1.1 Flags

The flags that can be set within the packet descriptor block are divided into two categories. Those that may be set in the left half of word 0 (.IPCFL) are instructions to the monitor concerning the packet communication. Those that may be set in the right half of word 0 describe the data message. The bits in the right half of the flag word are those returned within the associative variable, and those returned in the status word when a software interrupt occurs (refer to Chapter 5, Table 5-4, entry -24). The possible flags that can be set in the flag word are listed in Table 15-1.

15.1.2 PIDs

A PID must be obtained for any job that wants to send or receive a packet. PIDs can be obtained by

1. making a request for a PID to [SYSTEM] INFO, or
2. making a request for a PID to [SYSTEM] IPCC (only if the requester has the IPCF privilege).
3. using its job number for a PID.

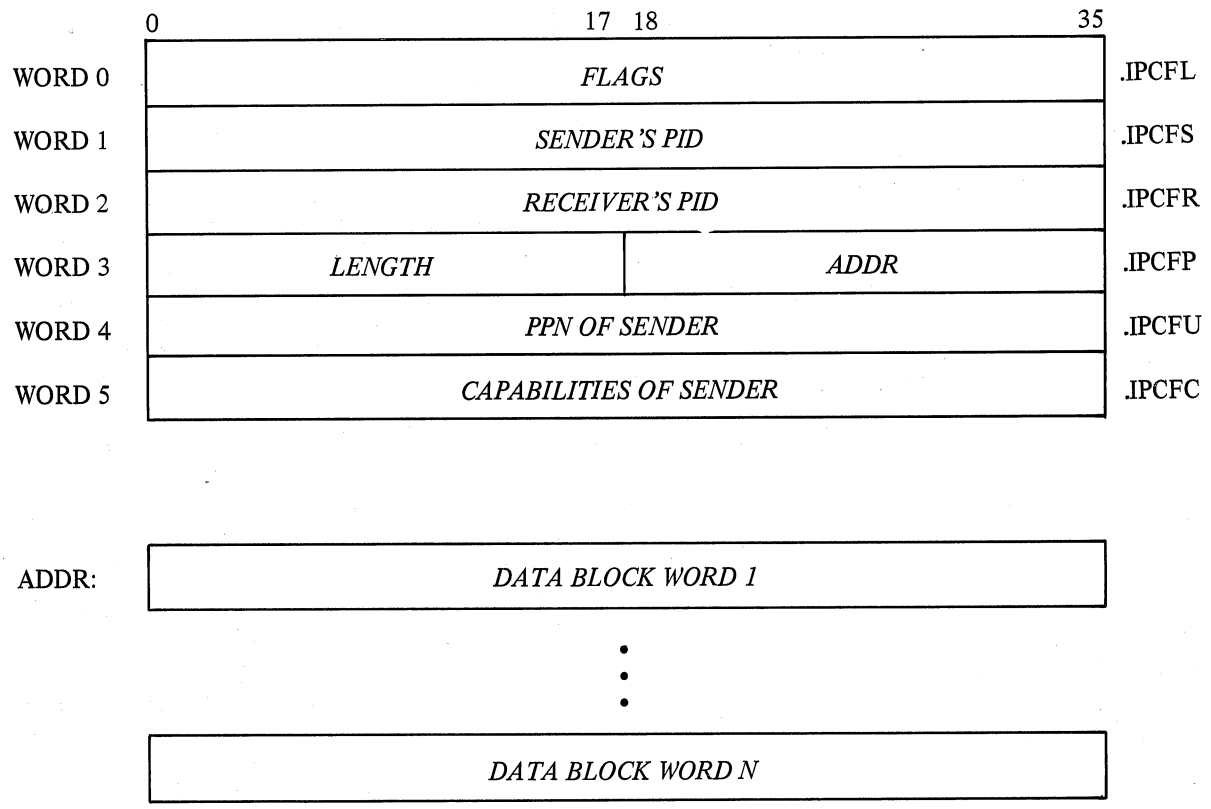


Figure 15-1. Representation of an IPCF Packet

Normally, a job will request its PID from [SYSTEM] INFO, obtaining a name associated with the new PID.

In summary, a job should send a packet to [SYSTEM] INFO (whose PID is 0) requesting that a PID be assigned to that job.

The job must also include, in the PID assignment request, a name that will be associated with the newly assigned PID. The PID will then be associated with both the job number and the symbolic name.

The association between a PID and a job number and name is released either 1) on a RESET, 2) when the job logs off the system, or 3) when a release is requested from [SYSTEM] INFO (depending on the type of request). [SYSTEM] INFO will not allow the assignment of a name which is already associated with another PID. (Unless the owner of the name makes the request.)

A PID can have only one name associated with it, but a job may have several PIDs (and therefore, several names) associated with it. After a PID has been used, it will never be used again until monitor is reloaded. This action protects against messages being sent to the wrong job by accident.

The symbolic name is limited to 29 characters, and it can contain any characters as long as the name is terminated by a zero word. In order to initiate communication between jobs, there should be a mutual understanding between the jobs as to the symbolic name(s) to be used. This mutual agreement frees the communication procedure from

Table 15-1
Packet Flags

Bit	Mnemonic	Meaning
0	IP.CFB	Do not wait for message if the queue is zero. If bit 0 is set on a receive and there is no packet in the queue, the error return is taken with error code 3 returned in the AC.
1	IP.CFS	Use the indirect sender's PID. (PID is found at the address specified in the word 1, .IPCFS).
2	IP.CFR	Use the indirect receiver's PID. (PID is found at the address specified in word 2, .IPCFR).
3	IP.CFO	Allow one send request above the send quota. (The default send quota is two.)
4	IP.CFT	Truncate the message, if it is larger than the reserved space.
5-17		Reserved.
18	IP.CFP	The packet is privileged. (This bit can be set only if the sending/receiving job is a privileged job.) If a privileged sender sets this bit, IPCFR. and IPCFQ. will return this. Bit = 1 in any reply. If this bit was not set, the bit will contain 0 when the packet is placed in the receiver's queue. If this job is not privileged and this bit is set, an error code will be returned.
19	IP.CFV	The packet is a page of data (512 decimal words). Both the sender and the receiver must have virtual memory capabilities. A page must have been reserved (by using the PAGE. monitor call) before a page can be passed using IPCF.
20-23		Reserved.
24-29	IP.CFE	The error code is returned in these bits, if an error is encountered on a receive or a send. The possible error codes are codes 70 to 77 listed in Table 15-5.
30-32	IP.CFC	The system and sender code; this code can be set only by a privileged job. Code — Mnemonic — Meaning 1 — .IPCCC — Sent by [SYSTEM] IPCC. 2 — .IPCCF — Sent by system-wide [SYSTEM] INFO. 3 — .IPCCP — Sent by receiver's local [SYSTEM] INFO.
33-35	IP.CFM	Return the packet to the sender. Code — Mnemonic — Meaning 1 — .IPCFN — Packet in the job's input queue is one that was sent to another PID but returned to the sender due to the receiver's PID being dropped before the packet was received. This bit cannot be set by an unprivileged job; the monitor will fill it in for examination by an unprivileged job.

any dependencies on system characteristics (such as job numbers) that might change between job executions. Four examples of symbolic names used in an IPCF communication are listed below.

[SYSTEM] PHOTOCOMP
FILEPROCESSOR [10,1521]
PHOTOCOMP [SYSTEM]
TESTPROGRAM ['ANY',151]

The symbolic name within square brackets may be specified in one of the following forms:

project number,, programmer number
string,, programmer number
string,, string
project number,, string
string (where *string* is restricted)

If a project-programmer number is used, it must be the project-programmer number under which the job is currently running. A job may specify [SYSTEM] as part of its symbolic name, only if the job is privileged (i.e., the JACCT bit has been set, or the job is running under project-programmer number [1,2]). When specifying the name associated with any PID or job number, you must be sure that the name is specified exactly as it appears, character for character.

Before a job can send a packet to another job, the sender must know either the intended receiver's name or its PID. If only the receiver's name is known, the sender must ask [SYSTEM] INFO for the PID associated with that name – since all communication takes place on a PID-basis only.

[SYSTEM] INFO keeps a list of all PIDs and their associated names currently assigned to each job.

15.1.3 Length of the Packet Data Block

The length of the packet data block is specified in word 3 (.IPCFP). The value specified will be one of two types, depending on the type of message contained in the packet data block. The message can be either a short form message or a long form message.

The short form message can be one to n words in length, where n is defined by the individual system installation (the default is ten words). The maximum length of the message can be obtained by examining .GTIPC (GETTAB Table Number 77, item number 0). The length specified in .IPCFP must be the actual word length of the message to be sent.

The long form message indicates that the length of the packet data block is one page (1000_8 words). When using the long form message, the left half of word 3 in the packet descriptor block must contain 1000_8. Note that in order to send or receive a long form message, the sending and receiving jobs must be running on a system with the virtual memory option, also bit 19 (IP.CFV) of the flag word in the packet descriptor block must be set, or an error code will be returned. If large amounts of data are to be set, the message portion of the packet may be used as a pointer to a file containing the data.

15.1.4 Address of the Packet Data Block

Word 0 of the packet data block is pointed to by the right half of word 3 (.IPCFP) in the packet descriptor block. The value specified within the right half of .IPCFP is either an address or a page number, depending on the type of message contained in the packet data block. If the message is a short form message, the address of word 0 of the message should be specified; if the message is a long form message, the page number of the message should be specified.

15.1.5 Sender's Project-Programmer Number

Word 4 (.IPCFU) is an optional portion of the packet descriptor block. If .IPCFU is present within the block on a receive, the monitor will fill in the sender's project-programmer number. .IPCFU is ignored on a send request.

15.1.6 Capabilities of Sender

Word 5 (.IPCFC) is an optional portion of the packet descriptor block. If .IPCFC is present within the block on a receive, the monitor will fill in the capabilities of the packet sender. This word is ignored on a send request. The possible capabilities of a sender are listed in Table 15-2.

Table 15-2
IPCF Capabilities of a Sender

Bit	Mnemonic	Meaning
0	IP.JAC	The sender has the JACCT bit turned on.
1	IP.JLG	The sender is logged-in.
2	IP.SXO	The sender is an execute-only job.
3	IP.POK	The sender has POKE privileges.
4	IP.IPC	The sender is an IPCF privileged job.

15.1.7 Packet Data Block

The packet data block contains the message portion of the packet. This portion can be in one of two forms: short form or long form. The short form message can be one to *n* words in length, where *n* is defined by the individual system installation (the default is ten). When a job sends a message to either [SYSTEM] IPCC or [SYSTEM] INFO, it always uses the short form. The long form message is one page in length (512_{10} words). When transmitting a long form message, both the sending and receiving jobs must be running on a system with the virtual memory option. Also, bit 19 (IP.CFV) of the flag word in the packet descriptor block must be set, or an error code will be returned.

15.2 SENDING A PACKET

To send a packet, the sending job must set up the packet descriptor block. When the IPCFS. monitor call is executed by the job, the message is sent to the intended receiver. When sending a packet with the sender's PID equal to 0, it indicates that the sender is the originator of the request. If sending a request to [SYSTEM] INFO and the second word of the packet data block contains another job's PID.

For example, to send a message to [SYSTEM] INFO requesting a PID, the following could be written:

```
MOVE  T1,[4,,[0          ;length,,noflags
          0              ;sender did not specify its PID
          0              ;default PID for [SYSTEM] INFO
          3,,NAME]]      ;length,,addr

IPCFS.  T1,              ;packet will be sent

 JRST   ERROR
          .
          .
          .
NAME:   0,,.IPCII        ;addr:code,,function name
        0                ;no duplicate copy to be sent

        ASCIZ/CORPWORD/  ;specified name to be associated
                         ;with PID
```

15.3 RECEIVING A PACKET

To receive a packet, the receiving job must set up a packet descriptor block. When the IPCFR. monitor call is executed by the receiving job, a packet will be retrieved from the input queue, if there is one. If there are no packets in the queue, the job will block until one arrives, unless IP.CIB was set in the flag word. The receive queue is emptied on a first-in, first-out basis. An example to receive a packet is

```
SETTI:  MOVE  T2,[6,,[0          ;set up descriptor block
                      0          ;monitor fills in sender's PID
                      0          ;monitor fills in receiver's PID
                      4,,DATA    ;length,,addr
                      0          ;monitor fills in sender's ppn
                      0]]        ;capabilities of sender
        IPCFR. T2,               ;packet is retrieved from input queue
         JRST  ERROR2
                      .
                      .
                      .
DATA:   BLOCK 4                  ;packet data block
```

15.4 [SYSTEM] INFO

The [SYSTEM] INFO facility acts as the central information utility for IPCF and performs several functions connected with names and PIDs. [SYSTEM] INFO will assign a PID, find a name associated with a PID, assign a name, or drop all PIDs associated with names. The IPCFS. monitor call is used to send packets to [SYSTEM] INFO with the message portion of the packet containing the request. The request must be in the general format shown in Figure 15-2.

	0 – 17	18 – 35
WORD 0 (**.IPCI0**)	*CODE*	*FUNCTION*
WORD 1 (**.IPCI1**)	*PID*	
WORD 2 (**.IPCI2**)	*FUNCTION ARGUMENT*	

Figure 15-2. Request to [SYSTEM] INFO

where: *code* is a user-declared quantity associating the answer with the appropriate request. If the user does not expect an answer, it is not necessary to specify a code; for consistency, the code field should contain zero if no answer is expected.

function is one of eight operations that a user may request [SYSTEM] INFO to perform. These functions are described in Table 15-3.

PID is the PID or job number of the job that is to receive a duplicate of the response from [SYSTEM]-INFO. If word 1 contains zero, the response is sent only to the originator of the request.

function argument is different depending on the function specified. Function arguments are described in Table 15-3 pertinent to each function.

All responses from [SYSTEM] INFO will be in the form of a packet, which is sent to the function requester. The message portions of the responses packets will be in the general format shown in Figure 15-3.

Table 15-3
[SYSTEM]INFO Functions

Function Code	Mnemonic	Request Format	Meaning
1	.IPCIW	*code*,,.IPCIW *PID for copy* *name*	INFO will find the PID associated with the name specified in the request.
2	.IPCIG	*code*,,.IPCIG *PID for copy* *PID*	INFO will find the name associated with the PID specified in the request.
3	.IPCII	*code*,,.IPCII *PID for copy* *name in ASCIZ*	INFO will assign the name specified in the function argument to the job number making the request. (This name will be disassociated from the job number when the job performs a RESET call.)
4	.IPCIJ	*code*,,.IPCIJ *PID for copy* *name in ASCIZ*	INFO will assign the name specified in the function argument to the job number making the request. (This name will be disassociated from the job number when the job logs off the system.)
5	.IPCID	*code*,,.IPCID *PID for copy* *PID to be* *dropped*	INFO will disassociate the specified PID from its job number. This function can only be requested by the owner of the specified PID, unless the requester is an IPCF privileged user.
6	.IPCIR	*code*,,.IPCIR *PID for copy* *job number*	INFO will disassociate all PIDs that were created by function 3 and are associated with the specified job number. This function can be requested only by the owner of the specified job, unless the requester is an IPCF privileged user.
7	.IPCIL	*code*,,.IPCLQ *PID for copy* *job number*	INFO will disassociate all PIDs that were created by function 4 and are associated with the specified job number. This function can only be requested by the owner of the specified job, unless the requester is an IPCF privileged user.
15	.IPCIS		Function 15 is used only by [SYSTEM]IPCC on the execution of the LOGOUT or RESET monitor call.

15.5 [SYSTEM]ICPP

The IPCF controller ([SYSTEM]IPCC) supports a number of functions, including enabling (disabling) jobs to receive packets, assigning a PID to [SYSTEM]INFO and destroying a PID. The IPCFS. monitor call is used to send a packet to [SYSTEM]IPCC with the message portion of the packet containing the request. The request must be in the general format shown in Figure 15-4.

	0 – 17	18 – 35
WORD 0 (**.IPCI0**)	*CODE*	*FUNCTION*
WORD 1 (**.IPCI1**)	*RESPONSE*	
WORD 2 (**.IPCI2**)	*RESPONSE*	

Figure 15-3. [SYSTEM] INFO Response Format

	0 – 17	18 – 35
WORD 0 (**.IPCS0**)	*CODE*	*FUNCTION*
WORD 1 (**.IPCS1**)	*FUNCTION ARGUMENT*	
WORD 2 (**.IPCS2**)	*FUNCTION ARGUMENT*	
WORD 3 (**.IPCS3**)	*FUNCTION ARGUMENT*	

Figure 15-4. Request to [SYSTEM] IPCC

where: *code* is a user-declared quantity that will associate the answer with the appropriate request.

function is one of the functions that a user may request [SYSTEM] IPCC to perform. The functions are listed in Table 15-4.

function argument is different depending on the function requested. Function arguments are described in Table 15-4 along with the functions.

All responses from [SYSTEM] IPCC will be in the form of a packet, which is sent to the function requester. The message portion of the response packet will be in the general format shown in Figure 15-5.

Table 15-4
[SYSTEM] IPCC Functions

Function Code	Mnemonic	Request Format	Meaning
1	.IPCSE	*code*,,.IPCSE *job number*	IPCC will allow the specified job number to receive packets. If the requester is not the owner of the job specified, it must have IPCF privileges.
2	.IPCSD	*code*,,.IPCSD *job number*	IPCC will disable the specified job's ability to receive packets. If the requesting job is not the owner of the specified job number, it must have IPCF privileges.
3	.IPCSI	*code*,,.IPCSI *PID or job number*	IPCC will return the PID associated with [SYSTEM] INFO. The value returned will be for the local [SYSTEM] INFO (if there is one) or secondly, the global [SYSTEM] INFO. (GETTAB Table Number 77, item number 1 contains the PID for the system-wide [SYSTEM] INFO.)

Table 15-4 (Cont.)
[SYSTEM] IPCC Functions

Function Code	Mnemonic	Request Format	Meaning
4	.IPCSF	*code*,,.IPCSF *m* *n*	IPCC will create a PID for [SYSTEM] INFO. The requesting job must have IPCF privileges to request this function. The n should be equal to 0 to create a global [SYSTEM] INFO, and not equal to 0 to create a local [SYSTEM] INFO. Local [SYSTEM] INFOs are valid only if another local or a global does not exist. m is the PID which is to become SYSTEM [INFO], 0 will delete [INFO]. n=0 indicates global [INFO] n=j make local [INFO] for job number j. Local [INFO] can be made only by local or global INFO, if either exists (sending PID=local on global). If no [INFO] then any privileged job can create it. Global INFO can be changed or destroyed only if sender PID is [INFO].
5	.IPCSZ	*code*,,.IPCSZ *PID to be destroyed*	IPCC will destroy a PID. The requesting job must have the IPCF privilege to request this function.
6	.IPCSC	*code*,,.IPCSC *type,,job number*	IPCC will create a PID for the specified job number. The requesting job must have IPCF privileges to request this function. Type can be 0 or 1; 1 indicates that the PID is valid until RESET; 1 indicates that the PID is valid until LOGOUT.
7	.IPCSQ	*code*,,.IPCSQ *PID or job number*	IPCC will set a send and receive quota for the specified job number. The requesting job must have the IPCF privilege to request this function. The send quota will be returned in bits 18-26 of the response word; the receive quota in bits 27-35.
10	.IPCSO	*code*,,.IPCSO *PID* *new job number*	IPCC will change the job number associated with the specified PID. The requesting job must have the IPCF privilege to request this function.
11	.IPCSJ	*code*,,.IPCSJ *PID*	IPCC will return the job number associated with the specified PID. The job number is returned in .IPCS2.
12	.IPCSP	*code*,,.IPCSP *job number*	IPCC will return all of the PIDs associated with the specified job number. Starting with .IPCS1, IPCC will return the PIDs. The number of PIDs returned depends on the length of the block.
13	.IPCSR	*code*,,.IPCSR *job number PID*	IPCC will respond with the send and receive quotas associated with specified job number or PID. The send quota will be returned in bits 18-26 of .IPCSI; the receive quota will be returned in bits 27-35 of .IPCS1.
14	.IPCSW	*code*,,.IPCSW *job number*	Wake a job sleeping from .IPCSS.
15	.IPCSS	*code*,,.IPCSS *job number*	If equal to 1, job is resetting. If equal to -1, job is logging out.

Table 15-4 (Cont.)
[SYSTEM] IPCC Functions

Function Code	Mnemonic	Request Format	Meaning
16-23			Reserved to Digital.
24	.IPCWP	*code*,,.IPCWP	IPCC will write the SYSTEM PID table.
25	.IPCRP	*code*,,.IPCRP	IPCC will read the SYSTEM PID table.
26	.IPCSU	*code*,,.IPCSU	IPCC sends a message to [SYSTEM] QUASAR.
27	.IPCSL	*code*,,.IPCSL *job*	Logout message sent to [SYSTEM] QUASAR.

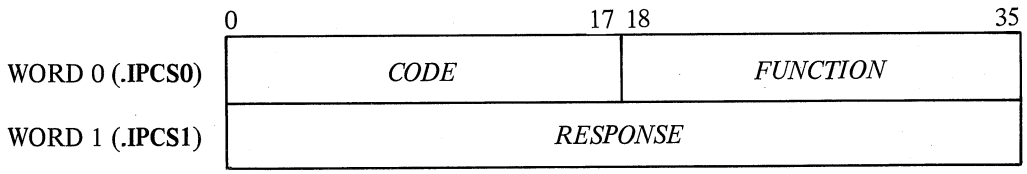


Figure 15-5. Response from [SYSTEM] IPCC

15.6 STATUS OF AN INPUT QUEUE

The IPCFQ. monitor call will return information regarding the status of a job's IPCF input queue. Its calling sequence is

```
MOVE  ac,[length,,addr]
IPCFQ. ac,
   error return
normal return
```

where: *length* is the length in words of the packet descriptor block (in the range 4 to 6).

addr is the first address of the block to store the packet descriptor block (i.e., IPCFQ. returns the same block as IPCFR. minus the packet data block.

On an error return, an error code will be returned in the AC indicating that the query failed; the error codes are listed in Table 15-5. On a normal return, the status of the input queue is returned in addr.

The IPCFQ. monitor call will check the status of the next packet in the receiver's queue. It is wise to query IPCF (with IPCFQ.) before a job tries to receive its first packet, in order to determine whether or not the first packet contains a long form message. (The job could also check bit 19 of word 0 in the packet descriptor block.) If the message in the queue is a long form message, the receiver must set bit 19 in word 0 of the receiving packet descriptor block.

After the job receives its first packet, it is not necessary to use IPCFQ., as IPCFR. will return an associated variable describing the next packet (if there is one) in the input queue. The associated variable (which is also the word returned when a software interrupt occurs) is represented in Figure 15-6.

0 — 17	18 — 35
LENGTH OF THE MESSAGE	*RH OF THE FLAG WORD*

Figure 15-6. An Associated Variable

15.7 RETRIEVE AN IPCF PACKET

The IPCFR. monitor call retrieves an IPCF packet from the job's receive queue. Its calling sequence is

```
MOVE  ac,[length,,addr]
IPCFR. ac,
   error return
normal return
```

where: *length* is the length in words of the packet descriptor block (in the range 4 to 6).
addr is the address of the packet descriptor block.

On an error return, the error code will be returned in the AC; the error codes are listed in Table 15-5. On a normal return, the packet is retrieved.

Before a job retrieves a packet, the job must

1. know whether or not the packet data block will contain a long form message, and
2. set up a packet descriptor block.

The packet descriptor block is shown in Figure 15-1. Bit 19 of Word 0 (IP.CFV) must be set if the expected packet data block contains a page of data. The length to be specified in bits 0-17 of Word 3 (.IPCFP) can be *n* or 1000_8. If the packet data block is a short form message, set the length to *n*; if the packet data block is a long form message, set the length to 1000. The length specified for *n* (short form message) may be greater than the length of the message so that the maximum limit may always be specified. If the specified length is less than the actual message length, the monitor call will take the error return unless the IP.CFT but was set in the flag word. If IP.CFT (bit 4) is set the message will be received correctly, but it will be truncated.

On a normal return from the IPCFR. monitor call, the AC contains the associated variable relating to the next packet in the receive queue (refer to Figure 15-6). If the AC is 0, there is no packet in the queue. The packet data block will be filled in as follows:

Word 0 – the left half remains the same, the right half contains the flags
Word 1 – the sender's PID is filled in by the monitor
Word 2 – the receiver's PID is filled in by the monitor
Word 3 – remains the same
Word 4 – filled in by the monitor
Word 5 – filled in by the monitor

15.8 SEND AN IPCF PACKET

The IPCFS. monitor call is used to send an IPCF packet. Its calling sequence is

```
MOVE  ac,[length,,addr]
IPCFS. ac,
   error return
normal return
```

where: *length* is the length in words of the packet descriptor block (in the range 4 to 6).
addr is the address of the packet descriptor block.

On an error return, an error code will be returned in the AC; all error codes are listed in Table 15-5. On a normal return, the packet is sent to the intended receiver. Refer to section 15.1 and its subsections for information regarding the packet descriptor block.

Table 15-5
IPCF Error Codes

Error Code	Mnemonic	Meaning
1	IPCAC%	An address check was encountered.
2	IPCNL%	The block is not long enough.
3	IPCNP%	There is no packet in the receive queue.
4	IPCIU%	Reserved.
5	IPCTL%	The data is too long for the user's buffer.
6	IPCDU%	Destination unknown (the receiver's PID is unrecognized).
7	IPCDD%	The destination has been disabled.
10	IPCRS%	The sender's quota has been exceeded.
11	IPCRR%	The receiver's quota has been exceeded.
12	IPCRY%	There is no room in system storage.
13	IPCUP%	Sender specified invalid page number, or receiver specified an existent page.
14	IPCIS%	Sender's PID is invalid as specified.
15	IPCPI%	Cannot perform function, because privileges are insufficient.
16	IPCUF%	Unknown function was specified.
17	IPCBJ%	Bad job number has been specified.
20	IPCPF%	PID table is full; new PIDs cannot be assigned at this time.
21	IPCPR%	Bit 19 was set in the flag word, but first packet was not a page or vice versa.
22	IPCIE%	Paging I/O error.
23	IPCBI%	A bad index has been specified for system PID table.
24	IPCUI%	Undefined ID in system PID table.
25-67		Reserved.
70	IPCFU%	[SYSTEM] INFO has an unknown, internal error.
71	IPCCF%	[SYSTEM] IPCC request from [SYSTEM] INFO failed.
72	IPCFF%	[SYSTEM] INFO failed to complete a request to assign a PID or a name.
73	IPCQP%	The PID quota has been exceeded.
74	IPCBP%	Unknown PID has been specified; if [SYSTEM] INFO should crash and restart, all existing PIDs will be lost.
75	IPCDN%	A duplicate name has been specified.
76	IPCNN%	An unknown name has been specified.
77	IPCBN%	The name specified has illegal characters. (Square brackets perhaps were not used properly.)

15.9 IPCF EXAMPLE

```
;EXAMPLE INITIALIZATION OF AY IPCF USER
;CALLED WITH A BLOCK OF SIX WORDS FROM NAME

;       CONTAINING AN ASCIZ STRING TO BE
;       SIGNED OUT BY THIS PROCESS
;       IF ERROR, NON-SKIP WITH T1=ERROR CODE
;       SKIP RETURN IF SUCCESSFUL

IPCINI: PUSHJ   P,,SAVE4##      ;PRESERVE P1-4
        MOVEI   T1,,IPCII       ;SIGN OUT UNTIL RESET
        MOVEM   T1,,NAME-2      ;SAVE AS FUNCTION
        SETZM   NAME-1          ;CLEAR WHO WANTS ANSWER
        MOVE    T1,[4,,[0               ;NO FLAGS
                        0               ;SEND FROM ME
                        0               ;SEND TO [SYSTEM]INFO
                        "D8,,NAME-2]] ;POINT TO ARGUMENT
        IPCFS,  T1,             ;SEND REQUEST
          POPJ  P,              ;ERROR, GIVE UP WITH CODE IN T1

;NOW BLOCK AND WAIT FOR AN ANSWER
;NOTE, THAT JUNK MUST BE DISCARDED

INILP2: MOVE    T1,[4,,P1]      ;POINT TO ARGUMENT BLOCK
        MOVEI   P1,0            ;CLEAR FLAGS
        MOVE    P4,[~D8,,DBLK]  ;POINT TO DATA
        IPCFR,  T1,             ;RECEIVE
          POPJ  P,              ;GIVE UP IF FAIL RETURNING ERROR IN T1
        MOVE    T1,DBLK         ;GET FUNCTION CODE
        LDB     T2,[POINT 3,P1,32]      ;GET SENDER'S CODE
        CAIE    T2,,IPCCF ;SEE IF FROM SYSTEM [SYSTEM]INFO
        CAIN    T2,,IPCCP ;OR IF FROM LOCAL [SYSTEM]INFO
        CAIE    T1,,IPCII ;YES--SEE IF INFO NAME MATCH
        JRST    INILP2          ;NO--TRY AGAIN
        LDB     T1,[POINT 6,P1,29]      ;GET MESSAGE ERROR CODE
        JUMPN   T1,,POPJ##      ;ERROR IF SET, RETURN TO CALLER IN T1
        MOVE    T1,DBLK+1 ;GET PID ASSIGNED
        MOVEM   T1,PIDUS  ;SAVE FOR TRANSACTIONS
        JRST    ,POPJ1##  ;GIVE OK RETURN

;ROUTINE TO IDENTIFY A RECEIVER
;CALLED WITH NAME OF RECEIVER IN SIX WORDS
;       FROM NAME IN ASCIZ
;       NON-SKIPS IF ERROR WITH CODE IN T1
;               .GT,0 IS IPCSER ERROR CODE
;               -1 IF RECEIVER UNKNOWN
;SKIP RETURNS WITH PID IN PIDHIM
IDRCVR: PUSHJ   P,,SAVE4##      ;PRESERVE P1-4
        MOVEI   T1,,IPCIW ;ASK WHO IS
        MOVEM   T1,NAME-2 ;SAVE AS FUNCTION
        SETZM   NAME-1          ;CLEAR WHO WANTS ANSWER
        MOVE    T1,[4,,[0 ;LENGTH, NO FLAGS
                        0       ;SEND FROM US
                        0       ;SEND TO [SYSTEM]INFO
                        ~D8,,NAME-2]] ;POINT TO ARGUMENT
        IPCFS,  T1,             ;SEND REQUEST
        POPJ,                   ;CAN'T SEND MESSAGE

IDRLP2: MOVE    T1,[4,,P1]      ;POINT TO ARGUMENT BLOCK
        MOVEI   P1,0            ;CLEAR FLAGS
        MOVE    P4,[~D8,,DBLK]  ;POINT TO DATA
        IPCFR,  T1,             ;RECEIVE
          POPJ  P,              ;GIVE UP IF FAIL WITH ERROR IN T1
```

```
        MOVE    T1,DBLK          ;GET FUNCTION CODE
        LDB     T2,[POINT 3,P1,32]       ;GET SENDER'S CODE
        CAIE    T2,,IPCCF ;SEE IF FROM SYSTEM [SYSTEM]INFO
        CAIN    T2,,IPCCP ;OR IF FROM LOCAL [SYSTEM]INFO
        CAIE    T1,,IPCIW ;SEE IF INFO NAME MATCH
        JRST    IDRLP2           ;NO--TRY AGAIN
        LDB     T1,[POINT 6,P1,29]       ;ISOLATE ERROR CODE
        JUMPN   T1,,POPJ##       ;IF SET, ERROR--RETURN IN T1
        SKIPN   T1,DBLK+1 ;GET PID ASSIGNED
        GVERR$  -1               ;IF NOT SET, ERROR
        MOVEM   T1,PIDHIM ;SAVE FOR TRANSACTIONS
        JRST    .POPJ1##  ;GIVE OK RETURN

;ROUTINE TO SEND A MESSAGE OF 8 WORDS IN MSG
;IT IS SENT TO THE PID IN PIDHIM
;NON-SKIP ON ERROR WITH CODE IN T1
;SKIP IF OK

SNDMSG: MOVE    T1,[4,,[IP.CFR           ;INDIRECR RECEIVER
                        0                ;SEND FROM US
                        PIDHIM            ;SEND TO HIM
                        ^D8,,MSB]]       ;POINT TO DATA
        IPCFS,  T1,              ;SEND
          POPJ  P,               ;ERROR--RETURN WITH CODE IN T1
        JRST    .POPJ1##  ;GOOD

;ROUTINE TO BLOCK UNTIL A MESSAGE IS RECEIVED
;IT CAN HANDLE UP TO 8 WORDS, STORED IN MSG
;NON-SKIP RETURNS IF ERROR WITH ERROR IN T1
;SKIPS WITH T1=LENGTH AND WITH SENDER IN PIDHIM

RCVMSG: PUSHJ   P,,SAVE4##       ;PRESERVE P1-4
        MOVE    T1,[4,,P1]       ;POINT TO ARGUMENT BLOCK
        MOVEI   P1,0             ;CLEAR FLAGS
        MOVE    P4,[^D8,,MSG]    ;POINT TO DATA AREA
        IPCFR,  T1,              ;BLOCK WAITING
          POPJ  P,               ;ERROR, GIVE UP WITH ERROR CODE IN T1
        MOVEM   P2,PIDHIM ;STORE SENDER
        HLRZ    T1,P4            ;GET ACTUAL LENGTH
        JRST    .POPJ1##  ;GOOD RETURN

;ROUTINE TO ISSUE A FATAL IPCF ERROR MESSAGE AND START OVER
;ENTERED WITH ERROR CODE IN T1

IPCERR: MOVEM   T2,T1            ;GET POSITIVE FORMAT
        TLNE    T2,-1            ;SEE IF STUFF IN LEFT HALF
        MOVEI   T1,0             ;YES--CALL MUST NOT BE IMPLEMENTED
        CAIL    T1,INFERR ;SEE IF INFO ERROR
        CAILE   T1,77            ;(RANGE INFERR TO 77)
        JRST    IPCER1           ;NO--TRY NORMAL IPCF ERRORS
        SUBI    T1,INFERR-MAXERR-1       ;YES--REMOVE TABLE OFFSET
        JRST    IPCER2           ;AND ISSUE MESSAGE
IPCER1: CAILE   T1,MAXERR ;SEE IF ERROR WE UNDERSTAND
        JRST    UNKERR           ;NO--GO HANDLE SEPARATELY
IPCER2: OUTSTR  @ERRTBL(T1)      ;OUTPUT TEXT
        JRST    FINERR           ;FINISH UP

UNKERR: OUTSTR  [ASCIZ \? UNKNOWN IPC ERROR CODE \]
        PUSHJ   P,,TOCTW##       ;ISSUE IN OCTAL

FINERR: OUTSTR  [ASCIZ \
\]
        EXIT
```

```
;TABLE OF ERROR MESSAGES

        DEFINE M($MES),<
        [ASCIZ \? $MES\]
>
        M       UNKNOWN RECEIVER
ERRTBL: M       IPCF NOT IMPLEMENTED
        M       ADDRESS CHECK
        M       BLOCK NOT LONG ENOUGH
        M       NO PACKET IN QUEUE
        M       PAGE IN USE
        M       DATA TOO LONG FOR BUFFER
        M       DESTINATION UNKNOWN
        M       DESTINATION DISABLED
        M       SENDING QUOTA EXCEEDED
        M       RECEIVING QUOTA EXCEEDED
        M       SYSTEM STORAGE EXCEEDED
        M       UNKNOWN PAGE (SEND), EXISTING PAGE (RECEIVE)
        M       INVALID SENDER
        M       INSUFFICIENT PRIVILEGES
        M       UNKNOWN FUNCTION
        M       BAD JOB NUMBER
        M       PID TABLE FULL
        M       PAGE REQUESTED WITH NON-PAGE PACKET NEXT
        M       PAGING I/O ERROR
MAXERR==,<ERRTBL-1          ;HIGHEST KNOWN ERROR CODE
        M       INFO HAD INTERNAL ERROR
        M       INFO RAN INTO AN IPCF REJECTION
        M       INFO FAILED TO COMPLETE AN ASSIGN
        M       INFO RAN OUT OF PID'S
        M       INFO COULD NOT IDENTIFY THE PID
        M       INFO FOUND A DUPLICATE NAME
        M       INFO KNEW OF NO SUCH NAME
        M       INFO DETERMINED THAT NAME HAS ILLEGAL CHARACTERS
INFERR==100-<,-<ERRTBL+MAXERR+1>>  ;FIRST INFO ERROR

;LITERALS

        XLIST
        LIST
        RELOC

;INPURE STORAGE

OFFSET: BLOCK   1           ;CCL START CODE
ORGFF:  BLOCK   1           ;ORIGINAL .JBFF,,.JBREL

ZCOR:!                      ;START OF AREA TO CLEAR
PDLST:  BLOCK   LN$PDL+2 ;PUSH-DOWN LIST
        BLOCK 2
NAME:   BLOCK   6           ;TRANSMISSION NAME
ENAME==,-1
FLSR:   BLOCK   1           ;1=SEND, 2=RECEIVE

PIDUS:  BLOCK   1           ;PID OF US
PIDHIM; BLOCK   1           ;PID OF OTHER GUY

DBLK:   BLOCK   8
MSG:    BLOCK   8
EMSG==,-1
EZCOR==,-1

        END     IPCFEX
```

CHAPTER 16
ENQUEUE/DEQUEUE FACILITY

16.1 OVERVIEW OF ENQUEUE/DEQUEUE

Many times users are placed in situations where they must share files with other users. Each user wants to be guaranteed that while he is reading a file, no other users are reading the same data and while he is writing a file, no other users are writing, or reading the same portion of the file.

By using the Enqueue/Dequeue facility, cooperating users can insure that resources are shared correctly and that one user's modifications do not interfere with another user's. Examples of resources that can be controlled by this facility are files, operations on files (e.g., READ, WRITE), records, devices, and memory pages. This facility can be used for dynamic resource allocation, computer networks, and internal monitor queuing. However, control of simultaneous updating of files by multiple users is its most common application.

The Enqueue/Dequeue facility insures data integrity among jobs only when the jobs cooperate in their use of both the facility and the physical resource. Use of the facility does not prevent non-cooperating jobs from accessing a resource without first enqueuing it. Nor does the facility provide protection from jobs using it in any manner. In order to enqueue a file (resource) the enqueuing user must have access to the concerned file (resource).

Jobs obtain access to a specific resource by placing a request in a queue associated with that resource. A resource is an entity within a system for whose use there is competition among jobs. The actual definition of the term resource (files, devices, records, fields, etc.) is defined by the job using it, not by the system.

A request for a resource is generated by the ENQ. monitor call. Each request enters a queue associated with the specified resource. A queue is merely an ordering of requests for a given resource.

When a request for a resource is granted, a lock is formed between the requesting job and the resource. While the lock is in effect, the requester is the owner of the resource. All other jobs which have requested access to the owned resource wait in the queue until the owner relinquishes ownership of the resource. There can, however, be more than one owner of a resource at one time; this is called shared ownership.

The owner of a resource relinquishes ownership via the DEQ. monitor call. DEQ. also can be used to remove a request for ownership from the queue of waiting requests.

The cycle of enqueuing and dequeuing requests for resources continues until all requests have been granted. After they have been granted, the Enqueue/Dequeue facility deletes the queue associated with that resource. When a new request is generated for that resource, Enqueue/Dequeue will create another queue.

16.1.1 Shared Ownership and Exclusive Ownership

Ownership of a resource can be requested as being shared or exclusive. Shared ownership occurs when two or more jobs request ownership of the same resource, each specifying that it will share ownership with one or more jobs. When the requests are granted, a lock is formed between the requesting jobs and the specified resource. If one job relinquishes ownership of the resource, the remaining jobs retain shared ownership of the resource until each of them relinquishes ownership also. Figure 16-1 illustrates shared ownership. Other requests for shared access are added to the lock as the requests are received.

Requests are enqueued in the order the requests are received, and requests for dequeuing are granted in the order they are received. When all owners of a resource have relinquished ownership, the resource is free for another job(s) to gain ownership of it.

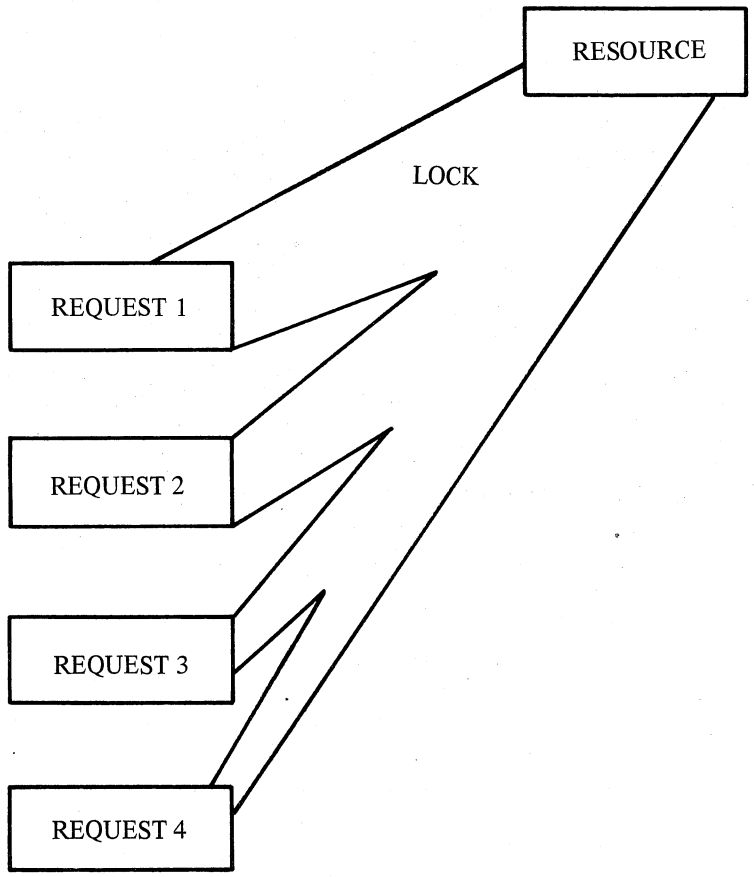


Figure 16-1. Shared Ownership

Exclusive ownership occurs when one job requests that it exclusively gain ownership of the specified resource. When the resource is exclusively owned, all jobs requesting ownership of the device must wait until the owner has relinquished ownership of the resource.

Each specified resource has one queue associated with it. Therefore, requests for shared and exclusive ownership of the same resource are placed in the same queue. Requests are placed in the queue in the order in which they are received.

In Figure 16-2, request 1, 2, and 3 are sharing ownership of the resource. When these 3 jobs have relinquished ownership of the resource, the next request in the queue will be granted. Since the next request in the queue is for exclusive ownership, a lock will be formed between request 4 and the resource as shown in Figure 16-3.

When this exclusive ownership lock is dissolved, request 5 can be granted. Since request 5 is for shared ownership, all other requests for shared ownership following it up to the next request for exclusive ownership can be granted at the same time. But request 5 is immediately followed by an exclusive ownership request, therefore request 5 will not share the resource with another job at this time. If, however, request 6 should be removed from the waiting queue while request 5 is the owner of the resource, requests 7 and 8 will be granted shared ownership of the resource with request 5.

16.1.2 Pooled Resources

A pooled resource occurs when multiple copies exist of a resource. Each resource in the pool can have one exclusive owner at a time. Resources in a pool cannot have shared ownership. Figure 16-4 illustrates a pooled resource.

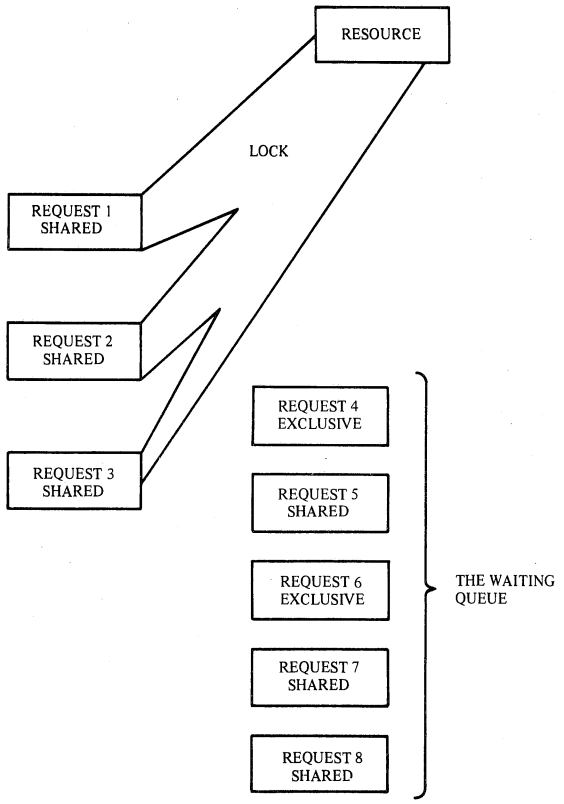


Figure 16-2. Shared and Exclusive Ownership Requests

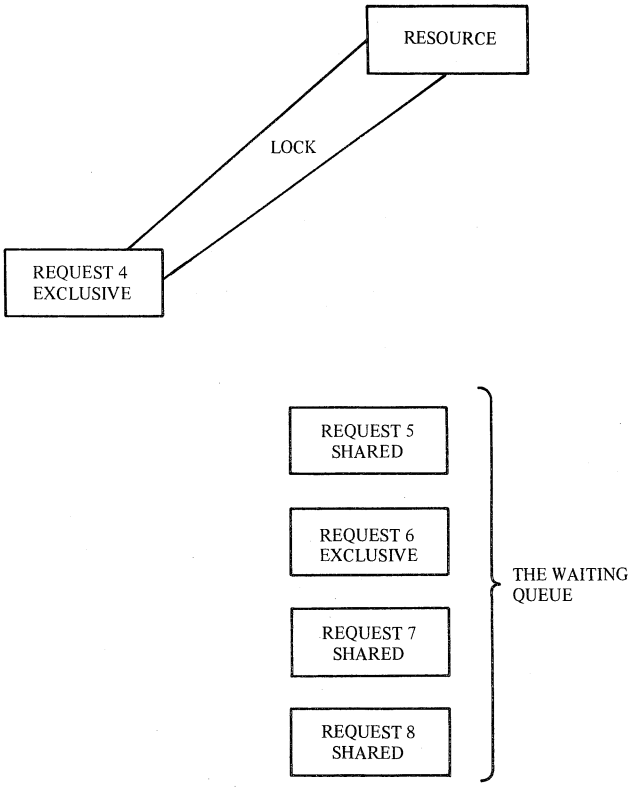


Figure 16-3. Exclusive Ownership

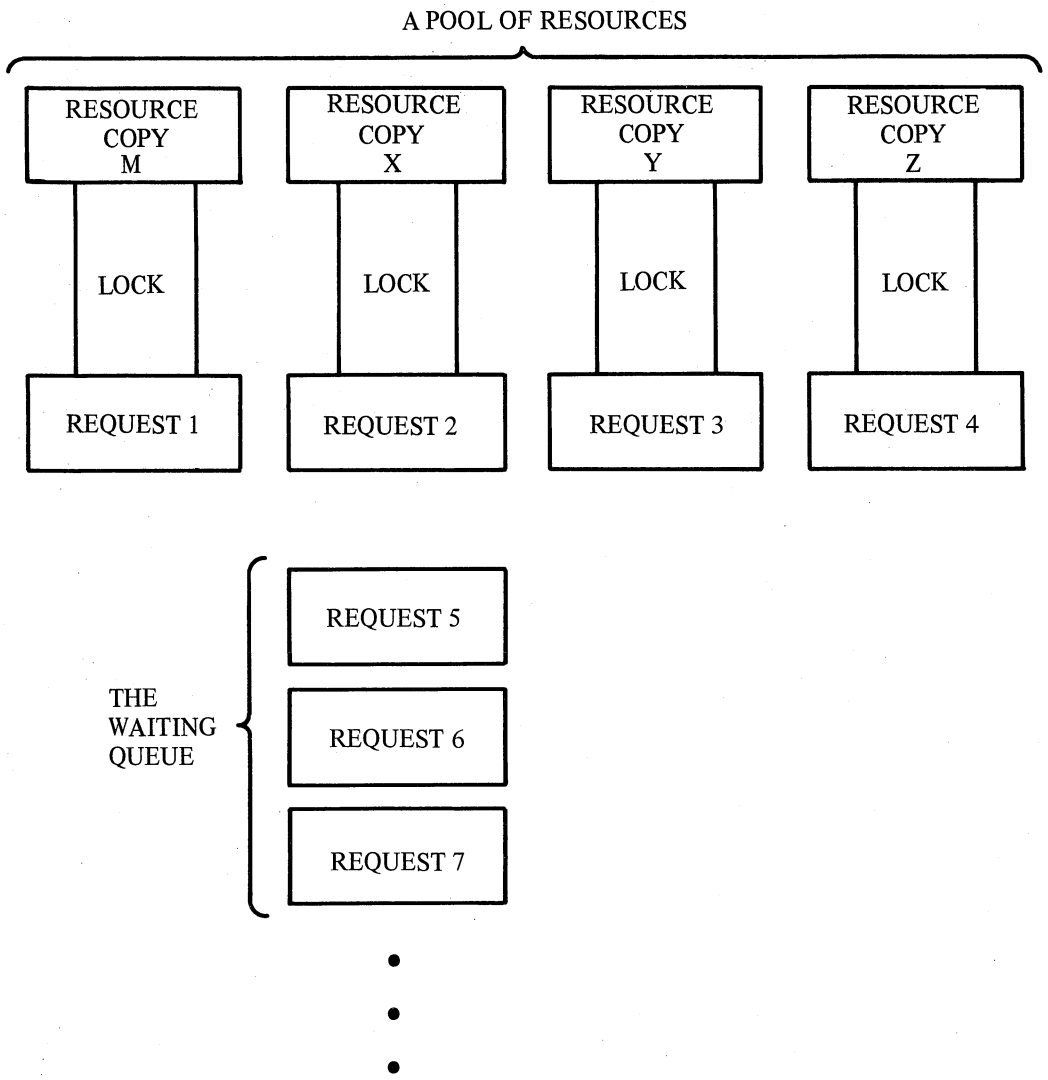


Figure 16-4. Pooled Resources

One job may issue a request asking for ownership of more than one resource in the pool. In other words, the owner can exclusively own one or more resources in a set of pooled resources. To illustrate this, look at Figure 16-4. Requests 1, 2, and 3 could belong to the same job. Then, that job is the exclusive owner of 3 of the 4 resources in the pool. There is no limit to the number of resources within a pool. The first job to specify a pooled resource specifies the number of resources within that pool.

At no time will Enqueue/Dequeue grant a request for ownership for more devices than there are in the pool or than there are unowned in the pool.

16.1.3 Sharer's Group

A sharer's group is a subset of the user jobs that desire ownership of a given resource. A sharer's group prohibits ownership of that resource by any job outside of that particular group. Whenever a member of a sharer's group has

been granted ownership of a resource, no other jobs except other jobs belonging to that sharer's group are allowed joint ownership of that resource.

Jobs specify the sharing of resources in this manner by specifying a sharer group number. The use of sharer groups restricts the access of a resource to a set of jobs smaller than the set of shared ownership (which by default is sharer's group 0) but larger than the set of exclusive ownerships.

16.2 ENQUEUE/DEQUEUE MONITOR CALLS

There are three Enqueue/Dequeue monitor calls. The ENQ. monitor call generates an ownership request for a specified resource. The DEQ. monitor call dequeues a request from the waiting queue or dissolves the lock between an owner (job) and the resource. The ENQC. monitor call returns the status of the resource owner(s) and the waiting queue associated with the specified resource.

16.2.1 The ENQ. Monitor Call (CALLI 151)

The ENQ. monitor call places a request in the queue associated with a specified resource. Its calling sequence is

```
MOVE  ac, [XWD function, loc]
ENQ.  ac,
  error return
normal return
```

where: *function* is one of the function codes listed in Table 16-1.

loc points to an argument block, which consists of a 2-word header followed by 1 or more 3-word request blocks. An argument block is represented in Figure 16-5.

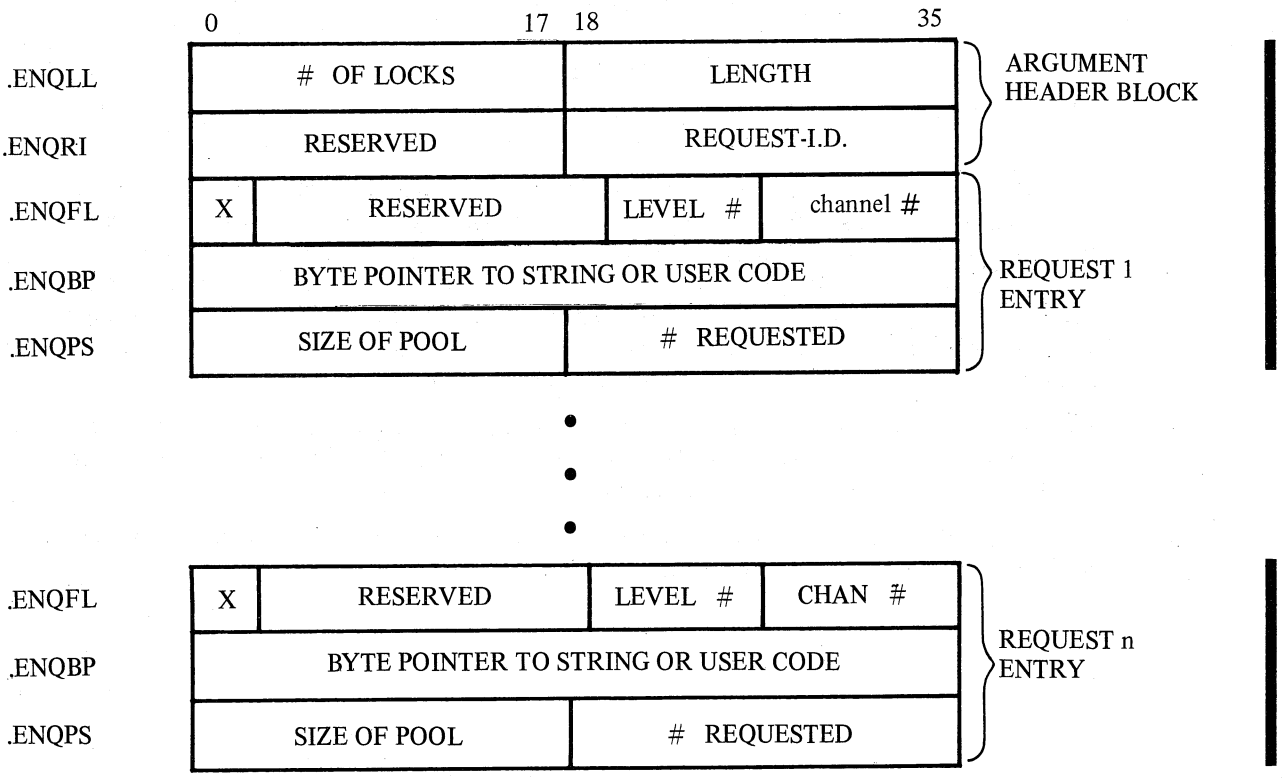


Figure 16-5. ENQ. Argument Block

where: *# of locks* (EQ.LNL) is the number of 3-word request entries in this argument block.

length (EQ.LLB) is the length in words of this argument block.

request-I.D. is an 18-bit quantity identifying an ENQ. request. This quantity uniquely identifies each request, enabling the user job to identify which request caused a software interrupt, if one occurred.

x is flags which may be set for this request entry. The flags that may be set are

Bit	Mnemonic	Meaning
0	EQ.FSR	This request is for shared access.
1	EQ.FBL	The system will bypass level numbers when processing this request.

level number (EQ.FLV) is a 9-bit quantity assigned to each resource class to which the job desires access. The monitor uses this value as a mechanism to reduce the probability of a resource deadlock. The actual association between a physical resource and its level number is known only by the user job.

channel number (EQ.FCC) is the number of the disk channel on which the file is being accessed. The user job must already have issued the appropriate LOOKUP/ENTER/FILOP. monitor call to initialize the file. Several users may specify the same channel number, yet each may be accessing different files. (Likewise, different channel numbers specified among a group of users could indicate the same file.)

user code can be set to one of the following instead of specifying a channel number.

Code	Mnemonic	Meaning
-1		Reserved.
-2	.EQFGL	The lock is a global lock. Jobs that are not accessing a file are able to operate without setting up a dummy file. If two users specify -2 and the same quantity in .ENQBP (refer to Figure 16-2), the locks are treated as identical locks.
-3	.EQFPL	The lock is a global lock and available only to the monitor. This code allows the monitor to define a set of locks with protection from other jobs.

byte pointer or user code is either the address of an ASCIZ string or a 33-bit user code. The string address can either be a standard byte pointer or in the form:1, ,addr (where addr is the address of a left-justified ASCIZ string). This quantity is the second part of the resource name (channel number, -2, or -3 are the alternate first portions of the resource name). The monitor makes no association between the resource and the resource name. The names are merely identifiers whose meaning is agreed upon in advance by all concerned parties. The ASCIZ string denotes a resource to all users, and the user code can represent a block number, record number, etc.

total size of pool (EQ.PPS) is the total number of resources in this pool. If EQ.PPS is zero, the resource is not a pooled resource. If EQ.PPS is nonzero, bit 0 in Word 0 of the request entry must be 0 (indicating an exclusive ownership request).

 # requested (EQ.PPR) is the number of resources in the pool that the calling job requests ownership of. This value must be less than or equal to EQ.PPS.

Table 16-1
ENQ. Function Codes

Code	Mnemonic	Meaning
0	.ENQBL	The request(s) for ownership are placed in the waiting queue. The job blocks until all requests for ownership have been granted. After all requests are granted (i.e., the locks have been formed), the normal return is taken and 0 is returned in the AC. If, when the call was issued, bit 1 of Word 0 of the request entry contained 1 (i.e., bypass level numbers), the AC could contain a non-zero value instead of 0. A non-zero value indicates that a level number sequencing error occurred, but it was ignored. If the call was in an incorrect format, the error return is taken and an error code is returned in the AC.
1	.ENQAA	If all requests for ownership specified in this argument block cannot be granted immediately, no requests are entered in waiting queues, the error return is taken and an error code is returned in the AC. If all requests can be granted immediately (i.e., all resources are currently unowned[1]), all desired locks are formed, the normal return is taken, and the AC is unchanged.
2	.ENQSI	Request for a Software Interrupt. If all requests for ownership can be granted immediately, this function code is identical to function 0. If all requests cannot be granted immediately, the error return is taken and error code 1 (ENQRU%) is returned in the AC. The interrupt when all resources available condition is enabled within the software interrupt system. The calling job continues processing until all resources become available, at which time the job receives a software interrupt. (Note that the job must first have properly set up the software interrupt system.) When an interrupt is received, the Enqueue/Dequeue request-I.D. is returned in the status word of the interrupt control block.
3	.ENQMA	The calling jobs want to modify the access specification of a previously given ownership request. The access specification can be changed from exclusive access to shared access, but an error code will result when trying to modify the access specification from shared to exclusive. If the modification specified is identical to the existing access specification, no action is performed and the normal return is taken. If the caller has no request in the specified queue, the error return is taken. If one call consists of more than one access modification request and the error return is taken, the caller must issue the ENQC. monitor call to determine which (if any) modification request was granted. The error code returned in the AC reflects the last error which occurred as a result of this ENQ. call.

The possible error codes resulting from the DEQ.. monitor call are listed in Table 16-6 at the end of this chapter.

16.2.2 The DEQ. Monitor Call (CALLI 152)

The DEQ. monitor call removes one or more requests from the waiting queue for a specified resource and/or it dissolves a lock between a job and the specified resource. Its calling sequence is

```
MOVE  ac, { [XWD function, loc]          }
          { [XWD function, request-i.d.] }
DEQ.  ac,
   error return
normal return
```

[1] or if the resource is owned by a shared job, and the calling request is for shared ownership and no one is ahead of caller in the queue.

where: *function* is one of the function codes listed in Table 16-2.

loc points to an argument block, which is represented in Figure 16-5. (DEQ.'s argument block is identical to the one used by ENQ.)

Table 16-2
DEQ. Function Codes

Code	Mnemonic	Meaning
0	.DEQDR	Dequeue the request(s) specified. On a normal return, the request will be removed from the specified waiting queue or the lock will be dissolved between the job and the specified resource. The error return is taken if the call was in an incorrect format or if the caller had no pending requests and was not an owner of the specified resource. The AC will contain an error code when the error return is taken.
1	.DEQDA	All requests from this caller are removed from waiting queues and all locks associated with this caller and a resource are dissolved. The error return is taken if the call was in an incorrect format or if the caller has no pending requests and is not the owner of a resource. When the error return is taken, the AC will contain an error code. The job should perform this function before EXITing; otherwise, a CLOSE will fail but the nature of the failure will be difficult to determine. This function is automatically performed on a RESET or a LOGOUT.
2	.DEQID	This function utilizes the alternate contents of the AC (i.e., function,,request-i.d.). All requests or locks associated with the specified request-i.d. are removed from the waiting queue or dissolved. This function should be used when requests are being enqueued and dequeued in the same block. When the request-i.d. is identical to the channel number specified in the ENQ. request, the caller can dequeue all requests/locks on a given file at once, without individually specifying the locks. The error return is taken if the call was in an incorrect format or if the caller has no pending requests and is not the owner of a resource. When the error return is taken, an error code is returned in the AC.

DEQ. calls that specify multiple requests are treated as multiple DEQ. calls specifying a single request. This is not true for the ENQ. monitor call. For example,

```
        MOVE    AC1, [XWD 0,DEQBLK]
        DEQ.    AC1
          HALT
        JRST    SUBR

DEQBLK: 2, , ^D8
        0, , 23
        0, , 2
        POINT 7, [ASCIZ/TEST/]
        ^ D10, , 1
        0, , 4
        POINT 7, [ASCIZ/TESER/]
        ^ D10, , 1
```

is identical to

```
        MOVE    AC1, [XWD 0,DEQBLK]
        DEQ.    AC1,
          HALT  .
        JRST    DEQ
DEQBLK: 1, , ^D8
        0, , 23
        0, , 2
        POINT 7, [ASCIZ/TEST/]
        ^D10, , 1
DEQ:    MOVE    AC1, [XWD 0,DEQ2]
        DEQ.    AC1,
          HALT  .
        JRST    SUBR
DEQ2:   1, , ^D5
        0, , 20
        0, , 4
        POINT 7, [ASCIZ/TESER/]
        ^D10, , 1
```

If an error is found in one request of a multiple request DEQ. call, the error return will be taken and an error code will be returned in the AC. However, Enqueue/Dequeue will continue processing until all dequeue requests have been performed. Therefore, all valid requests will have been dequeued whether or not an error resulted from another request in the same call. If errors are found in several requests, the error code in the AC reflects the last error found.

If the request/lock being dequeued/dissolved is associated with a pooled resource, an error will result if the caller attempts to dequeue more resources than were originally owned by the caller. However, the caller can dequeue a subset of those resources owned by him in a pool, still retaining ownership of those not dequeued.

The possible error codes resulting from the DEQ. monitor call are listed in Table 16-6 at the end of this chapter.

16.2.3 The ENQC. Monitor Call (CALLI 153)

The ENQC. monitor call is used to obtain information about the current state of the queues and to enable privileged programs to manipulate access rights to these queues. This capability may be useful to programs that need to determine such things as who is the owner of a resource.

The ENQC. monitor call has two formats. The first is identical to the ENQ. and DEQ. monitor calls and is used to obtain information about the state of a given resource. The second format is slightly different and is used to perform various utility functions on the queue structure. These calling sequences are described in the following sections.

16.2.3.1 Status Information – The calling sequence is

```
MOVEI   ac+1, status-block
MOVE    ac, [function, , loc]
ENQC.   ac,
  error return
normal return
```

loc: *# of locks, , length*
0, , *request-i.d.*
flags, , channel #
string-pointer
# of resources in pool, , # requested
.
.
.

status-block: BLOCK 3**n*

where: function code is 0 meaning that the status of the specified resource is to be returned.
n is the number of locks

The error return is taken if any error is found in the calling sequence.

On a normal return, the user's status block will have been modified so that each resource in his argument block will be represented by a 3-word entry in the status block. The first word of each entry describes the current state of the resource, the second word is a time-stamp that indicates how long someone has been waiting for that resource, and the third word is the Request I.D. for that resource.

The first word in each entry has the format described in Table 16-3.

Table 16-3
Current State of Resource

Bit	Meaning
0	Lock is invalid.
1	Calling user is owner.
2	Calling user is in a queue.
3	Owner is exclusive owner. If this bit is not on, there is no way of determining number of resource sharers.
9-17	Level number of resource.
18-35	Job number of owner or error code.

If bit 0 is set in the status word, an error was found in the corresponding lock specification. In such a case, bits 18-35 of this word will contain the pertinent error code.

Bits 18-35 will contain the job number of the owner of the lock (unless bit 0 has been set). If the lock is shared, the job number will indicate only one of the sharers. However, if the current job is one of the owners, this field will always contain his job number. Note that due to internal implementation algorithms, it is possible that there be no owner of a lock (even though several users may have requested ownership of that lock). In this case a -1 will be returned in bits 18-35.

The second word of each entry in the status block is a 36-bit value representing the last time at which someone was granted access to the resource. This value is the universal date-time standard, which can be obtained from

GETTAB Table Number 11, item number 53. If there is currently no owner of the resource, this word will contain zero.

This second word is useful if some type of "watch-dog" timer would be implemented. Such a timer could periodically query the status of any queue in which thru-put was of maximum importance. If it became obvious that the time-stamp of the queue had not changed for a long time, the timer process could signal the operator that someone had held the resource for longer than the allowable time interval. The operator would then have to take whatever action was deemed appropriate.

The third word of each entry contains a Request-I.D. in its right half (the left half is reserved for future use). If the user issuing the ENQC. monitor call is in the queue for this resource, this value will be the request I.D. for his request (even if a lock has not yet been formed). If this user is not in the queue for the resource, this field will contain the Request-I.D. of the owner of the resource.

The primary use of the Request-I.D. is to allow the use of Enqueue/Dequeue for limited communication of small amounts of data.

16.2.3.2 Modifying the Queue Structure – The calling sequence is

```
        MOVE  ac, [function, , loc]
        ENQC. ac,
           error return
        normal return

loc:    new quota, , job # ;for function
                           ;codes 1 and 2

loc:    length of block    ;for function
        Block n-1          ;code 3
```

where: *function* may be 1, 2, or 3. Refer to Table 16-4.

Table 16-4
ENQC. Function Codes

Code	Meaning
1	Return user's quotas in the AC.
2	Set user's quotas. The ENQ. quota of the specified user will be modified to the value in the left half of *loc*. In order to perform this function, the caller must have POKE. privileges. If he does not have these privileges, the request is rejected and an error code is returned. If the request is successful, a normal return is taken and a zero will be returned in the AC.
3	Dump the data-base. The entire data-base will be placed in the user's area, beginning with *loc + 1*. The contents of *loc* should be the length of the to-be-returned block. If this block is not large enough to hold the entire data-base, as much as possible will be returned. The end of the data-base dump is indicated by a word of ones. If the length specified is not a positive value, the error return will be taken. This is a privileged function; the calling user must have SPY privileges. All attempts to return the data-base by un-privileged callers, result in an error return and an error code being returned in the AC.

The format of a lock-block dump is shown in Figure 16-6. The format of a queue-block dump is shown in Figure 16-7.

FLAGS	LEVEL #	LOCK I.D.
# IN POOL		# REMAINING
TIME-STAMP		
ASCIZ STRING		

Figure 16.6. Lock-Block Dump

FLAGS		JOB #
GROUP #	# REQUESTED	REQUEST-I.D.

Figure 16-7. Queue-Block Dump

The flags (found in both types of dumps) are described in Table 16-5.

Table 16-5
ENQC. Flags

Bit	Meaning
0	This is a lock-block.
1	This is the lock owner.
2	This lock has text.
3	Exclusive access.
4	This job is blocked.

In the lock-block dump, the lock-I.D. corresponds to the value specified by the user in the channel number field of his argument block. However, since the channel numbers of several users will probably not be identical for a given file, a more suitable identifier is necessary. Therefore, if the lock is not file-related, the lock-I.D. will contain the exact number of the channel number field. If the lock is a file lock, this value will be an 18-bit random number that uniquely determines the file to which the lock refers. This quantity has absolutely no meaning to the user and is used by Enqueue/Dequeue solely for internal file identification.

The left half of word 1 indicates how many resources are in the pool. The right half of the same word indicates how many of those resources are still available.

The contents of word 3 (ASCIZ string or user code) can be determined from flag bit 2. If this bit is set, then word 3 contains a text string that is left-justified and continues until a null is encountered. If the null byte is not the right-most byte in the last word, then the contents of any bytes in that word to the right of the null byte are unpredictable.

In the dump of a queue entry, the right half of word 0 contains the job number of the user who issued the request for this lock. Flag bits 1, 3, and 4 determine whether this user is the owner, what his access is, and whether he is blocked in waiting for the resource.

If the current lock represents a pooled resource, the left half of word 1 of the queue-block dump will specify the number of resources requested by this user. If the resource is not pooled, this value will be the user's group number. The right half of this word is the user's request I.D. which corresponds to the specified resource.

Table 16-6
Enqueue/Dequeue Error Codes

Code	Mnemonic	Meaning
1	ENQRU%	One or more of the resources this job requested ownership of is available. This error code is returned in the AC if function code 2 is specified, after which a software interrupt will be generated when all resources are available.
2	ENQBP%	An illegal number of resources within a pool of resources has been specified. For example, this error would occur when a job requested ownership of seven resources from a pool containing only five resources.
3	ENQBO%	An illegal job number was specified. This error code is returned from the ENQC. monitor call.
4	ENQBB%	An incorrect or invalid byte size was specified in a text string. Byte size cannot be larger than 36 decimal.
5	ENQST%	The specified string is too long. The string can be no longer than
6	ENQBF%	An illegal function code was specified.
7	ENQBL%	An illegal argument block length was specified. The minimum block length is 5.
10	ENQIC%	The number of locks requested has been incorrectly specified.
11	ENQBC%	An illegal channel number has been specified.
12	ENQPI%	The function code specified requires that the caller be running under [1, 2] or have JACCT privileges.
13	ENQNC%	There is no available core.
14	ENQFN%	A channel number has been specified, but no file has been opened on that channel.
15	ENQIN%	An indirect or indexed byte pointer has been specified, but it is not allowed.
16	ENQNO%	The caler has no requests in a waiting queue and/or the caler is not the owner of a resource.

Table 16-6 (Cont.)
Enqueue/Dequeue Error Codes

Code	Mnemonic	Meaning
17	ENQLS%	A level number sequencing error has occurred. Level numbers must be specified in increasing order; the level number specified in this call was lower than the level number previously number specified.
20	ENQCC%	The access specification alteration requested cannot be performed. Access specifications can be changed from exclusive ownership to shared, but it cannot be changed from shared ownership to exclusive ownership.
21	ENQQE%	The Enqueue/Dequeue quota has been exceeded. This quota is set by the system administrator.
22	ENQPD%	The number of resources specified in this request as being in the pool is not equal to the actual number of resources in the pool. The actual number of resources in a pool is determined by the first job specifying this pooled resource. Those jobs wishing to use the same pooled resource after the pool has been defined must specify the same total number of resources in the pool.
23	ENQDR%	A duplicate of this request has already been received.
24	ENQNE%	A request was made to dequeue a request which was never enqueued.
25	ENQLD%	The level number specified in the request does not match the level number in the lock.
26	ENQED%	This function requires Enqueue/Dequeue privileges.

CHAPTER 17
METERING

17.1 OVERVIEW OF METERING

The meter facility of the DECsystem-10 permits users interested in performance measurement to implement measurement software cleanly and efficiently and with a minimum of interference with the rest of the system. The meter facility provides the interface necessary for measurement modules to be added to the monitor as needed, in a manner consistent with the monitor's overall structure. Ordinary user mode programs can access the data provided by these modules, and process it in any way that the user chooses.

The meter facility encompasses three types of modules:

meter points
meter point routines
meter channel routines

Meter point routines and meter channel routines are included in a monitor module called METCON. There are several varieties of each, but there is only one routine of each variety. They provide data processing and communications type functions for performance data. The objects called *meter points* are very small sections of code that act as sources of data when enabled, and there can be an arbitrary number of them scattered throughout the monitor. Each time the flow of control reaches an enabled meter point, the point supplies one word of data. The meter facility is designed to allow convenient addition of meter points whenever they might be needed. In fact, this is the overall purpose of the meter facility – to allow easy addition of meter points to the monitor and to provide the overhead and support functions that they require.

We have said that meter points act as sources of data. Meter point routines provide commonly needed data processing functions for the data supplied by a meter point. Some examples are addition of a point identification and addition of a time stamp field. Meter channel routines have the task of delivering the data to the user in some useful form. There are presently two types of meter channels – the trace channel and the display channel. The trace channel deposits each word into a buffer within a user program. The program can then process the data as the user chooses. The display channel keeps a running average of data values in a specified fixed location. The contents of that location can then be displayed in the console lights or recorded by a hardware monitor. The structure of the meter facility routines, as set up to the monitor one single meter point, is illustrated in Figure 17-1.

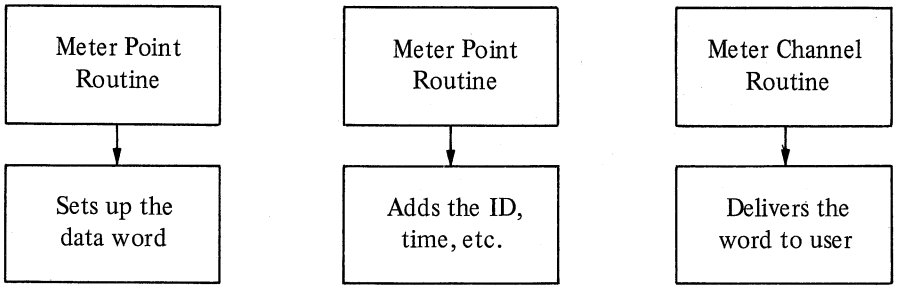


Figure 17-1. Metering

The linkages between meter point, meter point routine, and meter channel are set up dynamically as requested by a user program's executing the METER monitor call. The meter point routines and meter channel routines are

sharable, in the sense that the same routines can be used for different meter points by several user programs at the same time. A meter point, however, can be used by only one user job at any given time. It is possible for a single user program to use any number of different meter points on either the same or different meter point routines and meter channel routines.

As an example of how a program might use the meter facility, the program would first LOCK itself in core. It would then initialize the trace channel by executing the METER monitor call, the appropriate arguments. Included among the arguments would be the address of a buffer area within the program. Next the program would enable a specific set of points, giving the meter point routine and channel to be used with each. The program would then HIBERnate, and wait for data to be supplied in the buffer. When a specified amount of data has been supplied, the program will be awakened. It can then examine each word of data, using pointers and counters supplied by the meter facility, and it can process the data in any way the programmer chooses. Each time the program exhausts the data in the buffer, it will HIBERnate in order to wait for more data. When the program is finished, it will release the meter points and channel, unlock its core, and EXIT.

17.2 POINT ROUTINES

There are four different meter point routines that the user can choose from according to his needs. The 'value' routine, .MPRV, simply passes the data supplied by the meter point onto the channel. The 'time interval' routine, .MPRT, replaces the value supplied by the meter point with the time interval (in microsecond units) since that point was last executed. Note that neither of these routines supplies any identification of the point. Hence they generally cannot be used if the program initializes several points on the same trace channel.

The 'value plus ID' routine, .MPRVI, replaces bits 1 through 5 of the value supplied by the meter point with the point ID assigned to the point by the user when he initialized it. The 'time plus ID' routine, .MPRTI, replaces bits 1 through 5 with the point ID, and bits 6 through 35 with the time of day (in microsecond units). In each of the routines that supply the point ID, bit 0 of the data word will always be turned on.

17.3 USING THE TRACE CHANNEL

The following is what happens when a program uses the trace channel. The program must be LOCKed in core and it must stay locked as long as it wants to use the trace channel. The program then initializes the trace channel via the METER monitor call, supplying the addresses of the following items:

1. A buffer, whose length must be a power of 2.
2. A word to hold the buffer index.
3. A word containing a negative count equal to the same fraction of the buffer length. The job will be awakened after this number of words have been put into the buffer.
4. A flag word. Only one bit is used for the flag word. This bit indicates that the job does want to be awakened when the specified number of words have been added to the buffer.
5. A channel ID word. (This is normally 0 if a program initializes only one meter channel.)

Next the job must initialize the meter points which it wants to read. The program supplies a block of arguments for each meter point that the program wants to initialize. These blocks include the following information:

1. The name of the point. (The number assigned to the point when it was added to the monitor.)
2. The user's point ID. A 5-bit tag by which data from each point can be identified to distinguish it from data supplied by other points on the same channel. (The only reason this is used rather than the name is to save bits in the data word. The name would require 10 bits, while the user ID requires 5 bits.)
3. A point parameter (if one is required) by the point being initialized.
4. The point routine to be used data from this meter point.
5. The channel ID given to the channel to be used for data from this point. (This value is normally zero.)

Using this information, the point initialization routine links together the data structures required for meter point processing and enables the points.

Each time that the monitor's flow of control reaches one of the enabled points, a word of data is produced. The meter point sets up the word and dispatches to the point routine (whose address was obtained from the monitor's Meter Point Table at the same time that the test was made to determine if the point was enabled). The point routine performs its specified functions and passes the word on to the channel routine. The channel routine increments the buffer index (held in the user's core), truncates it to a size determined by the buffer length, and deposits the word into the designated location within the buffer.

The counter (also in the user's core) is incremented, and if it goes from negative to zero, a call is made to the WAKE routine. Control is then returned to the instruction following the meter point. Note that the meter code makes no check for data overrun in the buffer. It automatically cycles back to the beginning of the buffer each time it reaches the end as a result of truncating the buffer pointer. Checking for overrun is left up to the individual program. It is literally a count of how many words have been put into the buffer since the channel was initialized. By comparing the current value to the last value used, the program can determine how many words have not been processed. If this number is greater than the buffer length, an overrun has occurred.

The program initializes the count to a negative value that is some fraction of the buffer length. (Two thirds is a recommended value.) It then does a HIBERnate monitor call. When the count goes from negative to positive, the meter channel routine does a WAKE for the program. At some later time the program will get to run, and it will start processing the data point from the buffer. The program always remembers the last index value that it has used. It checks for overrun by looking at the difference between the current value of the index and the last value that it used. Note that more items can be added to the buffer even while the program is running. The program should then never 'remember' the current index value. Also, it is wise to check for overrun before processing each word. It is possible for overrun to occur while the program is running, even though everything was fine when the program was awakened. When the program has processed all data in the buffer, as indicated by its old index value being equal to the current index value, it resets the count word and does another HIBERnate.

If the program uses several meter points, it must assign IDs to each point in order to tell which is which in the buffer. Normally, it would contain a common routine to get the next word from the buffer. That routine would use the point ID as a pointer into a dispatch table, and call a specific routine for each point's data.

17.4 ADDING A METER POINT

A major purpose of giving the meter facility such as elaborate and general structure is to make it convenient to add and delete meter points as the need arises. To add a new meter point, the programmer must supply three very simple pieces of code:

1. An entry in the Meter Point Table, MPTAB.
2. An entry in the dummy Meter Point Table in COMMON. (Necessary only if the monitor is to be assembled at times without METCON. This table prevents undefined globals at the meter points when METCON is omitted, by providing table entries that are permanently turned off.)
3. The meter point.

The Meter Point Table entries are set up by adding a macro to the list of macros used to generate the table. The meter point 'name' is the only argument. Note that 'names' 1 through 499 are reserved for use in standard DEC software, and names 500 through 1023 are reserved for customer use. The names need not be assigned in order and need not be continuous. They should be thought of as simply as names, not as numbers.

The meter point must set up the word that it is to supply in accumulator T1. The meter point should test if it has been enabled and, at the same time, pick up the address of the Meter Point Data Block in accumulator T2, with a SKIPL or SKIPGE. A negative value of the MPTAB entry indicates that the point is enabled. The point always references its MPTAB entry by its individual label MPx, where x is the meter point name. The code for the Queue Transfer meter point is given below as an example that shows the general form that meter points should take. In the case of this particular point, the point parameter, found in the word MPDPAR of the Meter Point Data Block, is a job number. If a positive parameter value is supplied, only queue transfers for that job number will be reported. If the point parameter is negative, all queue transfers will be reported. On other points, that point parameter might have an entirely different meaning, or it might not be used at all.

Queue Transfer Meter Point
Job Number is in Accumulator J
Destination Queue Number is in Accumulator R

17.5 THE METER MONITOR CALL (CALLI 111)

The METER monitor call provides a mechanism for system performance metering by allowing privileged users to dynamically select and collect performance statistics from the monitor. The multifunction call controls are aspects of the metering facility in order that the user can collect, present, or reduce data for performance analysis or can tune individual jobs or the entire system. In order to use the METER monitor call, JP.MET (bit 3) must be set in the privilege word .GTPRV. The general calling sequence for METER is

```
MOVE     ac, [XWD n, addr]
METER .  ac,
   error return
normal return
```

where *n* is the number of arguments in the argument list.
addr points to the first word of the argument list.

If n is 0, the default number of arguments used depends on the particular function specified. Arguments in the argument list can be

arguments for the monitor,
values returned from the monitor, or
combinations of the two.

The first word of the argument block is the code for a particular function. The detailed descriptions of the various functions of the METER Monitor call are described within the METER. specification in the DECsystem-10 Software Notebooks, but they are summarized in Table 17-1.

Table 17-1
METER. Function Codes

Function Code	Mnemonic	Description
0	.MEFCI	Initialize the meter channel
1	.MEFCS	Obtain the meter channel status
2	.MEFCR	Release the meter channel
3	.MEFPI	Initialize meter points
4	.MEFPS	Obtain meter point status
5	.MEFPR	Release meter points

On an error return, an error code will be returned in the AC. The possible error codes are listed in Table 17-2.

Table 17-2
METER. Error Codes

Error Code	Mnemonic	Meaning
1	MEILF%	Illegal function code has been specified.
2	MENPV%	Caller is not a privileged user.
3	MEIMA%	An illegal memory address has been specified.
4	MEPDL%	A push-down list overflow has occurred.
5	MEIAL%	An illegal argument list has been specified.
6	MEIAV%	An illegal argument value has been specified.
7	MENFC%	There is not enough free core.
10	MEICT%	An illegal channel type has been specified.
11	MEIPT%	An illegal point type has been specified.
12	MENXP%	A non-existent point name has been specified.
13	MENXC%	A non-existent channel has been specified.
14	MEPNA%	The point type specified is not available.

CHAPTER 18
GETTABS

The monitor maintains many tables that contain various system and job related information. The tables are divided into two categories: those relating to jobs and high segments, and those relating to system statistics. The calling sequence for obtaining the contents of these tables is

```
MOVE    ac, [XWD index, table-number]
GETTAB ac,
   error return
normal return
```

where: *index* is either a job number, the number of a specific item number within the table, a high segment number, a channel number, or a class code pertaining to the scheduler. When *index* is greater than the highest job number in the system, the first high segment is indicated. When *index* equals -1, the current job is indicated. When *index* equals -2, the current job's high segment is indicated.

table-number is the number of the GETTAB table containing the desired information. All GETTAB tables are summarized in Table 18-1. Tables 18-2 through 18-27 list the entries of most GETTAB tables.

The entries in the GETTAB tables are global symbols defined in COMMON. The actual values of the core addresses of these locations are subject to change, but they can be found in the LINK-10 storage map for the monitor. A complete description of these globals can be found in the COMMON listing. The GETTABS are listed in UUOSYM.MAC for easy reference (refer to Appendix J).

Each customer installation may define its own GETTAB tables to the monitor. When doing so, a negative table number should be used so no confusion will arise between customer-generated tables and Digital-generated tables.

The error return is taken if there is no high segment when one is expected. The error return is also taken if an invalid job number, index number, or table number is specified.

On a normal return, the AC contains the contents of the specified table entry. If the specified table is not defined in the current running monitor, the AC will contain zero.

NOTE

Many GETTAB tables contain information in their undescribed bits. This information is likely to change and should be ignored. Although the field may currently be zero, there is no reason to believe that it will always be zero.

If a table is defined but does not exist in a given configuration, GETTAB will take the normal return with 0 being returned in the AC.

GETTAB subtables, via the GETTAB monitor call, make monitor-selected data available to user programs. These subtables allow installations to determine whether or not they want to use more monitor table space without invalidating any user programs. These subtables are included in all system configurations except the DECsystem-1040.

However, they may be excluded by changing the appropriate conditional assembly switches at Monitor Generation Time (via MONGEN). It is anticipated that only installations that need the necessary core space for uses other than the subtables will decide to exclude the subtables.

To reference a subtable, the user program first must issue a GETTAB monitor call to obtain the pointer to the subtable. Then the program issues a second GETTAB monitor call to get the desired item in the subtable. If the pointer is zero, the desired subtable is not included in the system.

The following example illustrates the method for obtaining the accumulated response times for CPUn for all users that waited for their jobs to initially run after TTY input was given.

```
%CCRSP== XWD 13,55          ;WORD and table number for response
                            ;subtable
%CVRAI== 3                  ;subtable index for accumulated TTY
                            ;INPUT monitor call response.
.GTC0V == 56                ;GETTAB table for CPU0 variables
        MOVEI   T1,N        ;CPU number (0, 1, ..., 5)
        LSH     T1,N        ;constants table GETTAB index moves
                            ;up by twos.
        MOVE    T2, [%CCRSP]    ;relative GETTAB pointer
                                ;word for response subtable
                                ;for CPU0.
        ADD     T2,T1       ;form GETTAB argument for CPU n.
        GETTAB  T2,         ;get relative pointer to response
                            ;subtable.
          JRST  NONE        ;not there (monitor is before 5.05)
```

Table 18-1
GETTAB Tables

Table No.	Table Name	Indexed By	Contents
00	.GTSTS[1]	Job/Segment	Job status.
01	.GTADR[1]	Job/Segment	Job reloaction and protection.
02	.GTPPN	Job/Segment	Project and programmer number.
03	.GTPRG	Job/Segment	User program name.
04	.GTTIM	Job Number	Total run time used, in unit of jiffies.
05	.GTKCT	Job Number	Kilo-core ticks for a job.
06	.GTPRV	Job Number	Privilege bits for a job.
07	.GTSWP[1]	Job Number	Swapping parameters for a job.
10	.GTTTY[1]	Job Number	Terminal-to-job translation.
11	.GTCNF	Item Number	Configuration table.
12	.GTNSW	Item Number	Non-swapping data.
13	.GTSDT	Item Number	Swapping data.

[1]Bits in these tables are explicitly not documented in later tables. These bits vary from version to version and are provided only for certain system programs, such as SYSTAT and DAEMON. User programs should not reference these locations if they are to be monitor version independent.

Table 18-1 (Cont.)
GETTAB Tables

Table No.	Table Name	Indexed By	Contents
14	.GTSGN	Job Number	High segment table.
15	.GTODP	Item Number	ONCE-Only disk parameters (refer to Table 18-7).
16	.GTLVD	Item Number	5-series monitor disk parameters (refer to Table 18-8).
17	.GTRCT	Job Number	Disk blocks read by job.
20	.GTWCT	Job Number	Disk blocks written by job.
21	.GTDBS	Job Number	Disk block seconds used by job.
22	.GTTDB	Job Number	Time of last allocate and size.
23	.GTSLF	GETTAB Tbl No.	GETTAB addresses (refer to Table 18-9).
24	.GTDEV	Segment Number	Device or file structure name of sharable high segment.
25	.GTWSN[1]	Item Number	Two-character SIXBIT name for job queues.
26	.GTLOC	Job Number	Remote Station Number.
27	.GTCOR[1]		Physical core allocation. There is one bit for each K of core in this word, if the system does not include the LOCK monitor call. There are two bits per K of core in this word, if the system does include the LOCK monitor call. A non-zero entry indicates that core is in use. (KA10 only)
30	.GTCOM		Monitor command names in SIXBIT.
31	.GTNM1	Job Number	User name in SIXBIT (first half).
32	.GTNM2	Job Number	User name in SIXBIT (second half).
33	.GTCNO	Job Number	Charge number for the job.
34	.GTTMP[1]	Job Number	TMPCOR pointers for job.
35	.GTWCH[1]	Job Number	WATCH bits for the job.
36	.GTSPL	Job Number	Spooling control bits for the job (refer to Table 18-10).
37	.GTRTD[1]	Job Number	Real-time status word for the job.
40	.GTLIM	Job Number	Time limit for job and Batch status for job (refer to Table 18-11).
41	.GTQQQ[1]		Timesharing scheduler's queue headers.
42	.GTQJB[1]	Job Number	Timesharing scheduler's queue table.
43	.GTCM2		SET command names in SIXBIT.
44	.GTCRS	Item Number	Status of hardware after a crash; status of devices (refer to Table 18-12).
45	.GTISC[1]		Swapper's input scan list of queues.
46	.GTOSC[1]		Swapper's output scan list of queues.
47	.GTSSC[1]		Scheduler's scan list of queues.

[1]Bits in these tables are explicitly not documented in later tables. These bits vary from version to version and are provided only for certain system programs, such as SYSTAT and DAEMON. User programs should not reference these locations if they are to be monitor version independent.

Table 18-1 (Cont.)
GETTAB Tables

Table No.	Table Name	Indexed By	Contents
50	.GTRSP	Job Number	Response counter table. An entry contains the time (in jiffies) when the user started to wait for his job to run. This time is cleared when the job is first given to the processor for run time by the scheduler.
51	.GTSYS	Item Number	System variables (independent of CPU). Refer to Table 18-13.
52	.GTWHY		Operator WHY RELOAD comments in ASCIZ.
53	.GTTRQ	Job Number	Total elapsed time a job was in a run queue whether or not job was running.
54	.GTSPS[1]		Job status word for second processor.
55	.GTC0C	Item Number	CPU0 control data block constants (refer to Table 18-14).
56	.GTC0V	Item Number	CPU0 control data block variables.
57	.GTC1C	Item Number	CPU1 control data block constants.
60	.GTC1V	Item Number	CPU1 control data block variables.
.			
.			
.			
67	.GTC5C	Item Number	CPU5 control data block constants.
70	.GTC5V	Item Number	CPU5 control data block variables.
71	.GTFET	Item Number	Feature test switches used in building the monitor (refer to Table 18-18).
72	.GTEDN		ERSATZ device names. The search lists for these devices can be obtained via the PATH. monitor call.
73	.GTSCN	Item Number	Scanner response data (refer to Table 18-19).
74	.GTSNA	Item Number	Last send-all message (refer to Table 18-20).
75	.GTCMT		SET TTY command names in SIXBIT.
76	.GTPID	Item Number	PIDs known by IPCF.
77	.GTIPC	Item Number	IPCF miscellaneous data.
100	.GTUPM[1]	Job/Segment	Physical page number of the user page map page.
101	.GTCMW	Item Number	SET WATCH command names in SIXBIT.
102	.GTCVL	Job Number	Current virtual page limit in the left half, and current physical page limit in the right half of the word (i.e., XWD cvpl, cppl).

[1]Bits in these tables are explicitly not documented in later tables. These bits vary from version to version and are provided only for certain system programs, such as SYSTAT and DAEMON. User programs should not reference these locations if they are to be monitor version independent.

Table 18-1 (Cont.)
GETTAB Tables

Table No.	Table Name	Indexed By	Contents
103	.GTMVL	Job Number	Maximum virtual page limit in the left half, maximum physical page limit in the right half of the word (i.e., XWD mvpl, mppl).
104	.GTIPA	Job Number	IPCF statistics (refer to Table 18-22).
105	.GTIPP[1]		IPCF pointers and counts.
106	.GTIPI[1]	Job Number	PID for the job's [SYSTEM] INFO.
107	.GTIPQ	Job Number	IPCF flags and quotas (refer to Table 18-23).
110	.GTDVL[1]	Job Number	Pointer to job's logical name table.
111	.GTABS[1]		Address break word.
112			Reserved.
113	.GTVM[1]		General virtual memory data (refer to Table 18-24).
114	.GTVRT	Job Number	Paging rate for job.
115	.GTSST	Item Number	Scheduler statistics (refer to Table 18-25).
116	.GTDCF[1]	Channel Number	Desired channel use fraction.
117	.GTST2[1]	Job Number	Second job status word.
120	.GTJTC[1]	Job Number	Job type and scheduler class.
121	.GTCQP	Class	Scheduler class quota in percent.
122	.GTCQJ	Class	Scheduler class quota in jiffies.
123	.GTCRT	Class	Scheduler class run time since the quotas were set.
124	.GTSQH[1]		Sub-queue headers.
125	.GTSQ[1]	Job Number	Sub-queue word for each job.
126	.GTSID	Item Number	Special PID table (refer to Table 18-26).
127	.GTENQ	Job Number	ENQ/DEQ statistics (refer to Table 18-27).
130	.GTJLT	Job Number	The time the job was logged in (in Universal Date/Time Format).
131	.GTEBT[1]		Jiffies of KL10 EBOX Time.
132	.GTEBR[1]		Jiffy remainder MOD RTUPS of 131.
133	.GTMBT[1]		Jiffies of KL10 MBOX Time.
134	.GTMBR[1]		Jiffy remainder MOD RTUPS of 133.

[1] Bits in these tables are explicitly not documented in later tables. These bits vary from version to version and are provided only for certain system programs, such as SYSTAT and DAEMON. User programs should not reference these locations if they are to be monitor version independent.

Table 18-2
Privilege Table
(.GTPRV, GETTAB Table Number 6)

Bit	Mnemonic	Meaning
0 = 1	JB.IPC	Job is allowed to perform IPCF privileged functions.
1-2 = n	JB.DPR	Highest disk priority for the job.
3 = 1	JB.MET	Job is allowed to execute the METER monitor call.
4 = 1	JB.POK	Job is allowed to execute the POKE monitor call.
5 = 1	JB.CCC	Job is allowed to change its CPU specification via a command or a monitor call.
6-9 = n	JB.HPQ	Highest high-priority queue available to this job.
10 = 1	JB.NSP	Job is allowed to unspool devices.
11 = 1	JB.ENQ	Job is allowed to perform ENQ/DEQ privileged functions.
12		Reserved.
13 = 1	JB.RTT	Job is allowed to execute the RTTRP monitor call.
14 = 1	JB.LCK	Job is allowed to execute the LOCK monitor call.
15 = 1	JB.TRP	Job is allowed to execute the TRPSET monitor call.
16 = 1	JB.SPA	Job is allowed to PEEK and SPY on all of core.
17 = 1	JP.SPM	Job is allowed to PEEK and SPY on the monitor.

Table 18-3
Configuration Table
(.GTCNF, GETTAB Table Number 11)

Item No.	Mnemonic	Meaning
0	%CNFG0	Name of system in ASCIZ.
1	%CNFG1	
2	%CNFG2	
3	%CNFG3	
4	%CNFG4	
5	%CNDT0	Date system created in ASCIZ.
6	%CNDT1	
7	%CNTAP	Name of system device in SIXBIT.
10	%CNTIM	Time of day in jiffies.
11	%CNDAT	Today's date (in 15-bit binary format)
12	%CNSIZ	Highest location in the monitor in low segment plus one. (System memory size).
13	%CNOPR	Name of OPR TTY in SIXBIT.
14	%CNDEV	Left Half: Start of the device data block (DDB) chain. Right Half: Unused.
15	%CNSJN	Left Half: Number of high segments. Right Half: Number of current jobs (including the null job).

Table 18-3 (Cont.)
Configuration Table
(.GTCNF, GETTAB Table Number 11)

Item No.	Mnemonic	Meaning
16	%CNTWR	If non-zero, system has two-register hardware and software.
17	%CNSTS	Left Half: Location describing feature test switches for this system. Right Half: Current state of switches. Refer to Table 18-4 for a description of the bit settings for the possible system states.
20	%CNSER	APR serial number.
21	%CNNSM	The number of nano-seconds per memory cycle for the memory system. If this GETTAB fails, the number of nano-seconds per memory cycle is↑D1000; this value is used by SYSTAT to compute the shuffling time.
22	%CNPTY	Left Half: The number of the first PTY (which is one greater than the number of the CTY). Right Half: The number of PTYs in the system configuration.
23	%CNFRE	Pointer to the bit map of core blocks.
24	%CNLOC	Left Half: Zero. Right Half: Address in the monitor for free four-word core block areas.
25	%CNSTB	Link to the station block chain (STB) for remote BATCH.
26	%CNOPL	Address of the line data block (LDB) of the operator's terminal.
27	%CNTTF	Pointer to TTY free chunks.
30	%CNTTC	Left Half: Number of TTY chunks. Right Half: Address of first TTY chunk.
31	%CNTTN	Number of free TTY chunks.
32	%CNLNS	Pointer to the current TTY as seen by the command decoder.
33	%CNLNP	Left Half: Total number of TTY lines. Right Half: Beginning of the line table.
34	%CNVER	Version number of the monitor; this value is stored in location 137 of the monitor as a save file when the monitor is not running. The bit definitions of this word entry are: Bit 0-17 are reserved for the customer. Bits 18-23 equal the monitor level. Bits 24-29 equal the monitor release number. Bits 30-35 are used for internal monitor development. If this GETTAB fails, the monitor currently running is a version previous to 5.03.
35	%CNDSC	Left Half: Length of the data set control table. Right Half: Beginning address of the data set control table.
36		Obsolete.
37		Obsolete.

Table 18-3 (Cont.)
Configuration Table
(.GTCNF, GETTAB Table Number 11)

Item No.	Mnemonic	Meaning
40	%CNSGT	Last dormant segment which was deleted to free a segment number.
41	%CNPOK	Address of the last location changed in the monitor via the POKE monitor call.
42	%CNPUC	Left Half: The job number of the job which most recently successfully executed the POKE monitor call. Right Half: The number of successful POKEs which have been executed.
43	%CNWHY	The reason for the last reload, stored as a SIXBIT unabbreviated operator answer. For more information, refer to the ONCE Specification within the DECsystem-10 Software Notebooks.
44	%CNTIC	The number of clock ticks per second. This refers to the time-of-day clock; the number is obtained by conducting an experiment at monitor load time. A different clock can be used for incremental run time accounting (refer to Item number 46, %CNRTC).
45	%CNPDB	The pointer to the process data block (PDB) pointer tables.
46	%CNRTC	The run time clock rate (in jiffies per second). This value is the rate of the clock used to measure the run time of a job and system statistics (null, lost, and overhead times). This rate is the precision of measurement, not the units of measurement.
47	%CNCHN	Left Half: The address of the first channel (DF10) data block. Right Half: Unused.
50	%CNLMX	The maximum number of jobs which may be logged in at the same time (LOGMAX).
51	%CNBMX	The maximum number of BATCH jobs which may be logged in at the same time (BATMAX).
52	%CNBMN	The minimum number of jobs reserved to BATCH (BATMIN).
53	%CNDTM	The host computer time in universal date/time format.
54	%CNLNM	The number of jobs currently logged (LOGNUM).
55	%CNBNM	The number of BATCH jobs currently logged in (BATNUM).
56	%CNYER	The year (LOCYER).
57	%CNMON	The current month (LOCMON).
60	%CNDAY	The current day of the month (LOCDAY).
61	%CNHOR	The local hour in 24-hour format (LOCHOR).
62	%CNMIN	The local minutes (LOCMIN).
63	%CNSEC	The local seconds (LOCSEC).
64	%CNGMT	Time offset from Greenwich Mean Time in internal format (i.e., offset + %CNDTM=GMT).
65	%CNDBG	The debugging status word, its bit definitions are: Bit 0=1 ST%DBG System debugging currently. Bit 1=1 ST%RDC Reload the system on the debug stop code. Bit 2=1 ST%RJE Reload the system when a job stop code occurs.

Table 18-3 (Cont.)
Configuration Table
(.GTCNF, GETTAB Table Number 11)

Item No.	Mnemonic	Meaning
		Bit 3=1 ST%NAR No automatic reloads are permitted. Bit 4=1 ST%CP1 If CPU1 stops (the second processor), stop CPU0.
66	%CNFRU	Number of free core blocks currently in use by the monitor.
67	%CNTCM	Number of nine-bit bytes in TTY chunks.
70	%CNCVM	Customer version number.
71	%CNDVM	Digital version number.
72	%CNDFC	Number of DF10 data channels.
73	%CNRTD	Number of real-time devices.
74	%CNHPQ	Number of high-priority queues.
75	%CNLDB	TTY device data block word pointing to the line data block.
76	%CNMVO	Maximum vector offset for PISYS (currently zero).
77		Reserved.
100	%CNMER	Left Half: Pointer to first magtape DDB. Right Half: Offset for MTA error reporting word. REELID in UDB.
101	%CNET1	User address of EXEC's AC T1 (for DAEMON).
102	%CNLSD	Length of the short device data block.
103	%CNLLD	The length of the long device data block.
104	%CNLDD	The length of the disk device data block.
105	%CNEXM	Address in JOBDAT of the most recently executed E(xamine) or D(eposit) command.
106	%CNST2	Software configuration indicators, the bit defintions are: Bit 18=1 ST%NDN Network device names are used of the form gggnnu. Bit 19=1 ST%XPI Exclude priority interrupt time from run time. Bit 20=1 ST%ERT EBOX/MBOX runtime (KL10 only). Bit 21=1 ST%EXE SAVE and SSAVE commands write .EXE files. Bit 22=1 ST%NJN System uses 9-bit job numbers. Bit 23=1 Extended error reporting. Bit 24=1 ST%TAP TAPSER included in system. Bit 25=1 ST%MBE Mass buss error reporting. Bit 26=1 ST%GAL GALAXY-10 support included. Bit 27=1 ST%ENQ ENQ/DEQ included in system. Bit 28=1 ST%SHC Scheduler has classes. Bit 29=1 ST%NSE Non-superseding ENTER. Bit 30=1 ST%MSG MPX channel feature is included in system.

Table 18-3 (Cont.)
Configuration Table
(.GTCNF, GETTAB Table Number 11)

Item No.	Mnemonic	Meaning
		Bit 31=1 ST%PSI Software Interrupt System is included in the system.
		Bit 32=1 ST%IPC IPCF is included in the system.
		Bit 33=1 ST%VMS The virtual memory option is included in the system.
		Bit 34=1 ST%MER MTA error reporting is included in the system.
		Bit 36=1 ST%SSP Swapping takes place on a page basis.
107	%CNPIM	Minimum condition in PISYS.
110	%CNPIL	Length of internal pits.
111	%CNPIA	Address of JBTPIA.
112	%CNMNT	Monitor type.
113	%CNOCR	Left Half: First card reader device data block.
		Right Half: Offset to card count.
114	%CNOCP	Left Half: First card reader device data block.
		Right Half: Offset to card count.
115	%CNPGS	Unit of core allocation.
116	%CNMMX	Minimum legal cormax.
117	%CNNSC	Number of scheduler classes.
120	%CNUTF	Exponential user time factor.
121	%CNHSO	Start of monitor high segment.
122	%CNHSL	Length of the monitor's high segment.
123	%CNNWC	The number of words of core.

Table 18-4
System State Bit Settings
(Item Number 17 in GETTAB Table Number 11)

Bit	Mnemonic	Meaning
0 =1	ST%DSK	The system is a disk system.
1 =1	ST%SWP	The system is a swapping system.
2 =1	ST%LOG	The system is a LOGIN system.
3 =1	ST%FTT	The system has full duplex TTY software.
4 =1	ST%PRV	The system contains privileged features.
5 =1	ST%TWR	Software is dual segment (reentrant).
6 =1	ST%CYC	System clock is set to 50 hz.
7-9	ST%TDS	Type of disk system, either:
=0		4-series disk system
=1		5-series disk system
=2		a spooled disk
10 =1	ST%IND	Project-programmer numbers are independent on the disk.
11 =1	ST%IMG	Image mode is supported by terminals (8-bit senser).
12 =1	ST%DUL	System is a dual-processor system.
13 =1	ST%MRB	Multiple ribs are supported.
14 =1	ST%HPT	High precision time accounting will be performed.
15 =1	ST%EMO	Monitor overhead will be excluded from time accounting measurements.
16 =1	ST%RTC	System has a real-time clock. (DK10)
17 =1	ST%MBF	System supports forots.
18-26		Reserved.
27 =1	ST%NOP	No operator is present at the central site.
28 =1	ST%MSP	Devices may be unspooled by an unprivileged user.
29 =1	ST%ASS	System is assigning/initializing on restricted devices.
30-31		Reserved.
32 =1	ST%NRT	There are not remote TTYs.
33 =1	ST%BON	Only batch jobs may login.
34 =1	ST%NRL	No remote logins will be accepted.
35 =1	ST%NLG	No LOGINS will be accepted from the CTY or OPR.

Table 18-5
Non-swapping Data Table
(.GTNSW, GETTAB Table Number 12)

Item No.	Mnemonic	Contents
0		Obsolete.
1-7		Unspecified data.
10	%NSCMX	Size (in words) of the largest legal user job which is valve equal to the low segment plus the high segment (CORMAX).
11	%NSCLS	Byte pointer to the last free block of core.
12	%NSCTL	The total amount of free core plus dormant core plus idle core left (virtual core).
13	%NSSHW	The number of the job stopped by the shuffler.
14	%NSHLF	The absolute address of the job above the lowest hole; this value will be zero if there is no job.
15	%NSUPT	The time the system has been up (in jiffies).
16	%NSSHF	The total number of words shuffled by the system.
17	%NSSTU	The number of the job using SYS: if not a disk.
20	%NSHJB	The highest job number currently assigned.
21	%NSCLW	The total number of words cleared by the system.
22	%NSLST	The total number of clock ticks when the null job ran and other jobs wanted to but could not because 1) job was swapped out or on its way in or out, 2) the monitor was waiting for I/O to stop so that it could shuffle or swap, or 3) the job was being swapped out because of core expansion.
23	%NSMMS	The size of physical memory in words.
24	%NSTPE	The total number of user parity errors since the system was loaded.
25	%NSSPE	The total number of spurious memory parity errors.
26	%NSMPC	The total number of multiple memory parity errors.
27	%NSMPA	The absolute location of the last user mode memory parity error.
30	%NSMPW	The contents of the last user mode memory parity error.
31	%NSMPP	The user PC of the last user mode memory parity error.
32	%NSEPO	The total number of push-down list overflows at the monitor call level in executive mode which were not recovered from.
33	%NSEPR	The number of push-down list overflows at monitor call level which were recovered by assigning an extended list.
34	%NSNXM	Highest legal value of CORMAX. (CORMAX is the size of the largest legal job.)
35	%NSKTM	Count-down time for the set KSYS monitor call.
36	%NSCMN	The amount of core guaranteed to be available after jobs have been locked in core (CORMIN).
37	%NSABC	The count of the number of address breaks handled since the system was loaded.
40	%NSABA	The contents of the data switches on the last address break.

Table 18-5 (Cont.)
Non-swapping Data Table
(.GTNSW, GETTAB Table Number 12)

Item No.	Mnemonic	Contents
41	%NSLJR	The number of the last job that ran, if this job number is different from the current job number.
42	%NSACR	The accumulated CPU response time. This is the total number of jiffies that all users waited for their jobs to initially run after either a command was issued which ran a job (or program) or timinal input was given that removed the job from a TTY input wait state.
43	%NSNCR	The number of CPU responses for all user waiting for jobs to run. Dividing the value of item number 42 (%NSACR) by the value of item number 43 (%NSNCR) gives the average response time since system startup.
44	%NSSCR	The accumulated squares of the CPU response time obtained from item number 42 (%NSACR).

Table 18-6
Swapping Data Table
(.GTSDT, GETTAB Table Number 13)

Item No.	Mnemonic	Contents
0	%SWBGH	The number of 1k core blocks which form the biggest hole in core.
1	%SWFIN	A minus number indicating the job number of the job being swapped out. —or— A positive number indicating the job number of the job being swapped in.
2	%SWFRC	The number of the job which is being forced to swap out.
3	%SWFIT	The number of the job which is waiting to be fit into core.
4	%SWVRT	The amount of virtual core left in the system (in 1k blocks). This value is initially set to the number of k of swapping space.
5	%SWERC	Left Half: The number of swap read or write errors. Right Half: Bits 18-21 are the error bits which are the same as the status bits returned from a GETSTS on the disk. Bits 22-35 contain the number of k marked as bad.
6	%SWPIN	The value of item number 6 is -1 if the job has been swapped-in monitors which swap process data blocks and FTPDBS = 1; PDBS only).

Table 18-7
ONCE-ONLY Disk Parameters
(.GTODP, GETTAB Table Number 15)

Item No.	Mnemonic	Contents
0	%ODSWP	Contains zero in 5- and 6-series monitors.
1	%ODK4S	The number of 1k disk words set aside for swapping on all units in the active swapping list.
2	%ODPRT	The in-core protect time[1] multiplied by the size of the job (in 1K core blocks).
3	%ODPRA	The in-core protect time added to the above result after multiplication.

[1] In-core protect time = %ODPRT *<jobsize> + %ODPRA

Table 18-8
LEVEL-D Parameters
(.GTLVD, GETTAB Table Number 16)

Item No.	Mnemonic	Contents
0	%LDMFD	The project-programmer number for UFDS only [1,1].
1	%LDSYS	The project-programmer number for device SYS: [1,4]. (In 4-series monitors [1,1].
2	%LDFFA	The project-programmer number for FAILSAFE [1,2].
3	%LDHLP	The project-programmer number of SYSTAT and HELP [2,5].
4	%LDQUE	The project-programmer number for spooling programs [3,3].
5	%LDSPB	Left Half: The address of the first PPB. Right Half: The address of the next PPB to be scanned.
6	%LDSTR	Left Half: The address of the first file structure data block. Right Half: The relative address of the next file structure data blocks, i.e., the address within the data block which points to the actual address of the next data block.
7	%LDUNI	Left Half: The address of the data block for the first unit in the system. Right Half: The relative address of the data block for the next unit in the system.
10	%LDSWP	Left Half: The address of the first unit for swapping in the system. Right Half: The relative address of the next unit for swapping in the system.
11	%LDCRN	The number of 8-word core blocks for disk systems which were allocated at ONCE-ONLY time.
12	%LDSTP	The standard file protection code (057), which can be changed by individual installations. In 4-series monitors the standard protection code is 055.
13	%LDUFP	The standard UFD protection code (775), which can be changed by individual installations. In 4-series monitors, the standard UFD protection code is 055.

Table 18-8 (Cont.)
LEVEL-D Parameters
(.GTLVD, GETTAB Table Number 16)

Item No.	Mnemonic	Contents
14	%LDMBN	The number of monitor buffers allocated at ONCE-ONLY time. In 4-series monitors, the number of monitor buffers is one.
15	%LDQUS	The SIXBIT name of the file structure containing the UFD for spooling and OMOUNT queues [3,3]. In 4-series monitors, this is DSK:.
16	%LDCRP	The UFD used for storage of system crashes [10,1].
17	%LDSFD	The maximum number of nested SFDs that the monitor allows to be created at one time.
20	%LDSPP	The protection code for spooled output files (stored in bits 0-7).
21	%LDSYP	The standard protection code for files on SYS: (155) for those files with an extension of .SYS.
22	%LDSSP	The standard protection code for files on SYS: with an extension of .SYS (157).
23	%LDMNU	The maximum negative argument that can be supplied for USETI, which reads extended RIBS.
24	%LDMXT	The maximum number of blocks transferred with one I/O operation (i.e., one IOWD). Normally this value is 100000, but it can be changed at MONGEN to be smaller so that a job doing high priority disk I/O will be locked out for a shorter period of time (since it can be locked out for as long as the channel is busy).
25	%LDNEW	The project-programmer number for experimental SYS:, [1,5].
26	%LDOLD	The project-programmer number for the library of superseded programs, [1,3].
27	%LDUMD	The project-programmer number for user mode diagnostics, [6,6].
30	%LDNDB	The default number of disk buffers in a buffer ring.
31	%LDMSL	The maximum units in the active swapping list.
32	%LDALG	The project-programmer number for the ALGOL library, [5,4].
33	%LDBLI	The project-programmer number for the BLISS-10 library, [5,5].
34	%LDFOR	The project-programmer number for the FORTRAN library, [5,6].
35	%LDMAC	The project-programmer number for the MACRO source library, [5,7].
36	%LDUNV	The project-programmer number for the universal library, [5,17].
37	%LDPUB	The project-programmer number for the public user software library, [1,6].
40	%LDTED	The project-programmer number for the text editor library, [5,10].
41	%LDREL	The project-programmer number for the REL file library, [5,11].
42	%LDRNO	The project-programmer number for the RUNOFF library, [5,12].
43	%LDSNO	The project-programmer number for the SNOBOL library, [5,13].
44	%LDDOC	The project-programmer number for the DOC file library, [5,14].
45	%LDFAI	The project-programmer number for the FAIL library, [5,15].
46	%LDMUS	The project-programmer number for the music library, [5,16].

Table 18-8 (Cont.)
LEVEL-D Parameters
(.GTLVD, GETTAB Table Number 16)

Item No.	Mnemonic	Contents
47	%LDDEC	The project-programmer number for standard DEC software, [10,7].
50	%LDSWP	The pointer to the active swap list.
51	%LDBAS	The project-programmer number for the BASIC library, [5,1].
52	%LDCOB	The project-programmer number for the COBOL library, [5,2].
53	%LDMXI	The project-programmer number for the PDP-11 library, [5,3].
54	%LDNEL	The project-programmer number for the NELIAC library, [5,20].
55	%LDDMP	The project-programmer number for DUMP, [5,21].
56	%LDPOP	The project-programmer number for POP2, [5,22].
57	%LDTDT	The project-programmer number for the TEST library, [5,23].

Table 18-9
GETTAB Immediate Word Entries
(.GTSLF, GETTAB Table Number 23)

Bit(s)	Meaning
0-8	The maximum item number in the table, if the table is indexed by item number.
9-11	A code which identifies the type of table. code — meaning 0 — non included in this configuration (e.g., .GTVM in a non-vm system) 1 — index by item number 2 — index by job number (maximum is given in the right half of %CNSJN) 3 — index by job number or segment number (maximum is given in the left and right halves of %CNSJN) 4 — index by job number (data in PDB)
14-17	A monitor AC number.
18-35	The executive mode address of the table if the code (bits 9-11) is 1, 2, or 3, offset to PDB if code (in bits 9-11) is 4.

Table 18-10
Spooling Control Table
(.GTSPL, GETTAB Table Number 36)

Bit	Mnemonic	Meaning
0-17		Input spooling file name (three characters).
24-26	JS.PRI	Disk priority.
27	JS.DFR	Deferred spooling (meaningful only in GALAXY-10 systems, refer to the SET SPOOL command).
28-30		Unused.
31	JS.PCR	Card reader spooling.
32	JS.PCP	Card punch spooling.
33	JS.PPT	Paper-tape punch spooling.
34	JS.PPL	Plotter spooling.
35	JS.PLP	Line printer spooling.

Table 18-11
Time and Batch Status Table
(.GTLIM, GETTAB Table Number 40)

Bits	Mnemonic	Meaning
0-9	JB.LCR	The core limit for the job.
10	JB.LBT	The job is a batch job.
11	JB.LSY	The job was run from SYS:.
12-35	JB.LTM	The time limit to go in jiffies.

Table 18-12
Hardware Status After A Crash
(.GTCRS, GETTAB Table Number 44)

Item No.	Mnemonic	Contents
0	CR.SAP	The APR CONI.
1	CR.SPI	The PI CONI.
2	CR.SSW	The APR DATAI switches.
3n		Reserved to Digital.

Table 18-13
System Wide Data Table
(.GTSYS, GETTAB Table Number 51)

Item No.	Mnemonic	Meaning
0	%SYERR	The system wide hardware error count.
1	%SYCCO	Obsolete.
2	%SYDEL	The count of unlogged hardware errors (error reporting disabled).
3	%SYSPC	Left Half: A three letter name for the last STOPCD. Right Half: The address plus one of the last STOPCD.
4	%SYNDS	The number of DEBUG STOPCDS.
5	%SYNJS	The number of job related STOPCDS.
6	%SYNCP	The number of commands processed.
7	%SYSJN	The job number related to the last STOPCD.
10	%SYSTN	The TTY name related to the last STOPCD.
11	%SYSPN	The program name related to the last STOPCD.
12	%SYSUU	The monitor call related to the last STOPCD.
13	%SYSUP	The user pc related to the last STOPCD.

Table 18-14
CPU0 Control Data Block Constants Table[1]
(.GTC0C, GETTAB Table Number 55)

Item No.	Mnemonic	Meaning
0	%CCPTR	Left Half: Pointer to the next control data block. If this is the last CDB, the value of item number 0 is zero. Right Half: Unused.
1	%CCSER	The APR serial number.
2	%CCOKP	The number of jiffies equal to the amount of time this CPU has been stopped. If this value is less than or equal to zero, the CPU is running o.k. If this value is greater than zero, this CPU has stopped running correctly.
3	%CCTOS	The trap offset for KA10 interrupt locations (this value will be either 0 or 100).
4	%CCLOG	The logical CPU name in SIXBIT (e.g., CPUN).
5	%CCPHY	The physical CPU name in SIXBIT (e.g., CPAN, CPIN, CP6N).
6	%CCTYP	The type of processor (1H is for customers; RH is for Digital). Value / Mnemonic / Meaning 1 / .CC166 / PDP-6 2 / .CCKAX / KA10 processor 3 / .CCKIX / KI10 processor 4 / .CCKLX / KL10 processor
7	%CCMPT	The relative GETTAB pointer to the memory parity bad address subtable. bits 0-8 are the maximum relative entry in the subtable. bits 18-35 are the relative address of the first word in the subtable.
10	%CCRTC	The device data block for the real-time clock (DK10). If the value of this word is zero, there is no real-time clock.
11	%CCRTD	The device data block for the real-time clock if there is to be high precision run time accounting. If the value of this word is zero, there is to be no high precision run time accounting on this CPU.
12	%CCPAR	The relative GETTAB pointer to the memory parity subtable. The bit definitions for this word entry are the same as for item number 7 (%CCMPT).
13	%CCRSP	The relative GETTAB pointer to the response subtable. The bit definitions for this word are the same as for item numbers 7 and 12.

[1]The items within the table correspond to the items in the constants table for each processor (e.g., entries for CPU1 CDB constants are found in GETTAB table .GTCIC, table 57).

Table 18-15
CPU0 Control Data Block Variable Table
(.GTC0V, GETTAB Table Number 56)

Item	Mnemonic	Meaning
5	%CVUPT	Uptime in jiffies for this CPU.
12	%CVLST	Last time in jiffies for this CPU.
14	%CVTPE	Total memory parity error words detected during all CPU sweeps on this CPU while the processor was in exec or user mode. If the system halts, this location has already been updated.
15	%CVSPE	The total number of spurious memory parity errors detected on this CPU (i.e., errors which did not reoccur when the CPU swept through core). This can occur on a read-pause-write which rewrites memory or on a channel-detected parity error not found on the sweep (refer to item number 11 in table number 19-17).
16	%CVMPC	Multiple memory parity errors for this CPU. That is, the number of times the operator pushed continue after a serious memory parity halt. The left half of the word contains 1 if a serious error on this bad parity (must halt). The left halt is cleared on a continue or STARTUP.
17	%CVMPA	Memory parity address for this CPU. That is, the first bad physical memory address found when the monitor swept through core after the processor or channel detected the first parity error.
20	%CVMPW	Memory parity word for this CPU. That is, the contents of the first bad word found by the monitor when it swept through core after the processor channel detected the first bad parity error.
21	%CVMPP	Memory parity PC for this CPU. It is the PC of the last memory parity (not counting the sweep through core).
27	%CVABC	Address break count on this CPU.
30	%CVABA	Address break address on this CPU.
31	%CVLJR	The last job run on this CPU including the null job.
32-34		Obsolete. Refer to item numbers 20-23 in table number 19-16.
35	%CVSTS	Stop timesharing on this CPU. This item number contains the number of the job that last performed the TRPSET monitor call.
36	%CVRUN	Operator-controlled scheduling for this CPU (OPSER:SET RUN command). bit 0 (CV%RUN) = 1 indicates that jobs are not to be run on this CPU.
37	%CVNUL	Null time (in jiffies) for this CPU.
40	%CVEDI	LH = exec PC so that offending instruction can be corrected. RH = number of exec "do not care" interrupts (i.e., user enabled apr interrupts which monitor causes (AOV, FOV).
41	%CVJOB	Current number of the job running on this CPU (0 is the null job).
42	%CVOHT	Overhead time (in jiffies) for this CPU. This value includes clock queue processing, short command processing, swapping and scheduling decisions, and software context switching. This value does not

Table 18-15 (Cont.)
CPU0 Control Data Block Variable Table
(.GTC0V, GETTAB Table Number 56)

Item	Mnemonic	Meaning
		include monitor call execution or I/O interrupt times, since these times are not considered overhead.
43	%CVEVM	KI10/KL10 only. The maximum amount of exec virtual address space to be used for mapping user segments on a LOCK monitor call.
44	%CVEVU	KL10/KI10 only. The current amount of exec virtual address space being used for mapping user segments on a LOCK monitor call.
45	%CVLLC	On a dual-processor system, the count of the number of times a CPU has looped in the CPU interlock while waiting for it to be relinquished by the second CPU.
46	%CVTUC	The total number of monitor calls executed on this CPU from exec and user mode.
47	%CVTJC	The total number of job context switching switches from one job to a a different job, including the null job, on this CPU.

Table 18-16
Response Subtable

Item No.	Mnemonic	Use
0	%CVRSO	Accumulated TTY output for monitor call responses. This is the total number of jiffies users have spent waiting for their jobs to do a TTY output (on CPU0) after either a command was issued which ran a job or terminal input was given that removed the job from a TTY input wait state.
1	%CVRNO	The number of TTY output monitor call responses for this CPU.
2	%CVRHO	The high-order sum of the squares of TTY output for monitor call responses. This value is used for computing the standard deviation.
3	%CVRLO	The low-order part of the sum of the squares of TTY output for monitor call responses.
4	%CVRSI	The accumulated TTY input monitor call responses for this CPU. This is the total number of jiffies users have spent waiting for their jobs to do a TTY input monitor call (on CPU0) after either a command was issued that ran a job or terminal input was given that removed the job from a TTY input wait state.
5	%CVRNI	The number of TTY input monitor call responses for this CPU.
6	%CVRHI	The high-order sum of the squares of TTY input monitor call responses. This value is used for computing the standard deviation.
7	%CVRLI	The low-order part of the sum of the squares of TTY input monitor call responses.
10	%CVRSR	The accumulated CPU quantum requeue responses. This is the total number of jiffies users have spent waiting for their jobs to exceed the CPU quantum on this CPU after either a command was issued that run a job or terminal input was received that removed the job from a TTY input wait state.

Table 18-16 (Cont.)
Response Subtable

Item No.	Mnemonic	Use
11	%CVRNR	The number of CPU quantum requeue responses for this CPU.
12	%CVRHR	The high-order sum of the squares of CPU quantum requeue response. Used for computing standard deviation.
13	%CVRLR	The low-order part of the sum of the squares of CPU quantum requeue response.
14	%CVRSX	The accumulated response terminated by the first occurrence of one of the above three events (TTY output, TTY input, or CPU quantum requeue).
15	%CVRNX	The number of such responses in %CVRSX.
16	%CVRHX	The high-order sum of the squares of responses in %CVRSX. This is used for computing the standard deviation.
17	%CVRLX	The low-order part of the sum of squares of responses in %CVRSX.
20	%CVRSC	The accumulated response on this CPU. The total number of jiffies that users spent waiting for their jobs to run after either a command was issued which ran a job or terminal input was supplied that removed the job from a TTY input wait state.
21	%CVRNC	The number of CPU responses for all users waiting for their jobs to run. Dividing this value into the value of %CVRSC gives the average response time since the system was started.
22	%CVRHC	The high-order part of the sum of the squares of CPU responses on this CPU.
23	%CVRLC	The low-order part of the sum of the squares of CPU responses of this CPU.

Table 18-17
Parity Subtable

Item No.	Mnemonic	Use
00	%CVPLA	Highest bad memory parity address on last sweep of memory. Used to tell operator the range of bad addresses.
1	%CVPMR	Relative address (not virtual address) in the high or low segment of the last memory parity error.
2	%CVPTS	Number of parity errors on the last sweep of core. Set to 0 at beginning of the sweep.
3	%CVPSC	Number of parity sweeps by the monitor.
4	%CVPUE	Number of user-enabled parity errors.
5	%CVPAA	The and of bad addresses on the last memory parity sweep.
6	%CVPAC	The and of bad contents on the last memory parity sweep.
7	%CVPOA	The or of bad addresses on the last memory parity sweep.
10	%CVPOC	The or of bad contents on the last memory parity sweep.
11	%CVPCS	Number of spurious parity errors. (The APR sweep found no bad parity but the channel had requested the sweep rather than the processor). This indicates a channel memory port problem.

Table 18-18
Feature Table
(.GTFET, GETTAB Table Number 71)

Item No.	Mnemonic	Bit	Meaning
0	%FTUUO	22	Enqueue/DEQUEUE is implemented (F%EGDG).
		23	Galaxy-10 features are implemented (F%GALA).
		24	PSISER is implemented (F%PI).
		25	IPCF is implemented (F%IPCF).
		26	CTRL/C intercept (F%CCIN).
		27	JBTSTS and CTLJOB monitor calls are implemented (F%PTYU).
		28	PEEK monitor call is implemented (F%PEEK).
		29	POKE. monitor call is implemented (F%POKE).
		30	Job continue (F%JCON).
		31	Spooling is supported (F%SPL).
		32	Job privileges are supported (F%PRV).
		33	DAEMON is supported (F%DAEM).
		34	GETTAB exists (F%GETT).
		35	System has 2-register relocation (F%2REL).
1	%FTRTS	Real time and scheduler features	
		25	6.02 (and later) scheduler is implemented (F%NSCH).
		26	System has virtual memory (F%VM).
		27	Swapper is implemented (F%SWAP).
		28	Shuffler is implemented (F%SHFL).
		29	DK10 service is implemented (F%RTC).
		30	The LOCK monitor call is implemented (F%LOCK).
		31	The TRPSET monitor call is implemented (F%SLEE).
		32	Real-time traps are implemented (F%RTTR).
		33	The sleep monitor call is implemented (F%SLEE).
		34	The HIBER and WAKE monitor calls are supported (F%HIBW).
		35	High priority queues are supported (F%HPQ).
2	%FTCOM	20	SAVE and SSAVE commands will create .EXE files (F%EXE).
		21	Memory can be set off-line (F%MOFF).
		22	Memory can be set on-line (F%MONL).
		23	COMPIL commands are implemented (F%CCL).
		24	COMPIL-class command are implemented (F%CCLX).
		25	Queue is implemented (F%GCOM).
		26	SETUUO and SET command are implemented (F%SET).
		27	VERSION command is implemented (F%VERS).
		28	Batch control file commands are implemented (F%BCOM).
		29	SET DAYTIME and SET DATE commands are implemented (F%SEDA).
		30	WATCH command has been implemented (F%WATC).
		31	FINISH and CLOSE commands have been implemented (F%FINI).
		32	REASSIGN command has been implemented (F%REAS).
		33	E and D commands have been implemented (F%EXAM).
		34	SEND has been implemented (F%TALK).
		35	ATTACH has been implemented (F%ATTA).
3	%FTACC	Accounting information	
		31	Time and core limits (F%TLIM).
		32	Charge number (F%CNO).

Table 18-18 (Cont.)
Feature Table
(.GTFET, GETTAB Table Number 71)

Item No.	Mnemonic	Bit	Meaning
		33	User name (F%UNAM).
		34	Kilo-core ticks accumulation (F%KCT).
		35	Run-time computation (F%TIME).
4	%FTERR	Error Control and internal options	
		22	NXM error recovery codes (F%MNXM).
		23	System is a KL10 (F%KL10).
		24	System is a KA10 (F%KA10).
		25	22 bit channel, DF10C (F%22BI).
		26	Swapping process data block (F%PDBS).
		27	KI10 features at startup time (F%KI10; always since 5.06).
		28	METER. monitor call is supported (F%METR).
		29	Execute-only files (F%EXON).
5	%FTDEB	Debugging features.	
		27	System has a two segment monitor (F%2SEG).
		28	Response time measurement (F%RSP).
		29	Why reload code (F%WHY).
		30	Patch space left in table (F%PATT).
		31	Back-tracking information left in common (F%TRAC).
		32	Monitor halts on error (F%HALT).
		33	Redundancy checking for internal errors (F%CHK).
		34	Monitor is write-protected (F%MONP).
		35	Monitor check summing (F%CHEC).
6	%FTSTR	File structure parameters.	
		19	System has high availability features (F%DHIA).
		20	System has multiple access update feature (F%DSIM).
		21	NUL device (F%NUL).
		22	LIB/SYS/NEW (F%LIB).
		23	Disk priority transfers (F%DPRI).
		24	Append to last block (F%APLB).
		25	Append implies read (F%AIR).
		26	Generic device search (F%GRSC).
		27	Rename cross directories (F%DRDR).
		28	SEEK monitor call is supported (F%DSEK).
		29	Super useti/useto are supported (F%DSUP).
		30	Disk quotas (F%DQTA).
		31	Multiple file structures (F%STR).
		32	5-series monitor calls (F%5UUO).
		33	Physical-only i/o (F%PHYO).
		34	Sub-file directories (F%SFD).
		35	STRUUO functions (F%MOUN).
7	%FTDSK	Internal disk parameters	
		19	Debug search list code (F%SLCK).
		20	Two-part access blocks (F%2ATB).
		21	DEBUG CB interlock (F%CBDB).
		22	LOGIN system (F%LOGI).

Table 18-18 (Cont.)
Feature Table
(.GTFET, GETTAB Table Number 71)

Item No.	Mnemonic	Bit	Meaning
		23	Disk system (F%DISK).
		24	Race-condition prevention in filfnd (F%FREE).
		25	Swap read error recovery (F%SWPE).
		26	Bad block marking (F%DBBK).
		27	UFD compressor (F%DUFC).
		28	Disk error simulation (F%DETS).
		29	Extended ribs supported (F%DMRB).
		30	Smaller allocation for disk core blocks (F%DSMC).
		31	Allocation optimization (F%DALC).
		32	Disk usage statistics (F%DSTT).
		33	Hung disk recovery (F%DHNG).
		34	Disk off-line recovery (F%DBAD).
		35	Latency optimization (F%DOPT).
10	%FTSCN	Scanner options.	
		22	2741 on a DC10 is supported (F%DCXH).
		23	Unusual vertical positioning is allowed (F%TVP).
		24	TYPESET-10 features in DC76 (F%TYPE).
		25	2741-like terminals supported (F%2741).
		26	DC76 (F%CAFE).
		27	TTY BLANK command is supported (F%TBLK).
		28	Page and display knowledge (F%TPAG).
		29	Automatic dialer supported (F%DTAL).
		30	Special line control (F%SCLC).
		31	Hardware scanner (DC10 or DC8 (F%SCNR).
		32	Modem control (F%MODM).
		33	Single scanner (F%630H).
		34	U.K. modem supported (F%GPO2).
		35	Real half-duplex terminals (F%HDPX).
11	%FTPER	I/O parameters	
		23	System supports DAS 78.
		24	System supports DA28-C networks.
		25	MSGSER implemented mpx device (F%MSGS).
		26	High-speed logical device search (F%HSLN)./
		27	CDP trouble intercept (F%CPTR).
		28	CDR trouble intercept (F%CRTR).
		29	CTYL supported (F%CTYL).
		30	Remote station supported (F%REM).
		31	LPT error recovery (F%LPTR).
		32	Device errors to operator (F%OPRE).
		33	CDR super-image mode (F%CDRS).
		34	MTA density and buffer size (F%MTSE).
		35	TMPCOR area (F%TMP).

Table 18-19
Scanner Table
(.GTSCN, GETTAB Table Number 73)

Item No.	Mnemonic	Use
0	%SCNRI	Number of RCV interrupts.
1	%SCNXI	Number of XMT interrupts.
2	%SCNEI	Number of echo interrupts. (subset of %SCNXI).
3	%SCNMB	Maximum buffer size.
4	%SCNAL	Number of active lines.

Table 18-20
SEND-ALL Text
(.GTSND, GETTAB Table Number 74)

Item No.	Mnemonic	Use
0	%SCNAE	Byte pointer to the end byte in the message.
1	%SCNAS	Byte pointer to the first byte in the message.
2	%SCBAN	First word of data in the message.

Table 18-21
IPCF Miscellaneous Data
(.GTIPC, GETTAB Table Number 77)

Item No.	Mnemonic	Use
0	%IPCML	Maximum packet length.
1	%IPCSI	PID of system-wide [SYSTEM] INFO.
2	%IPCDQ	Default quota.
3	%IPCTS	Total number of packets sent.
4	%IPCTO	Total number of packets outstanding.
5	%IPCCP	PID of [SYSTEM] IPCC.
6	%IPCPM	PID mask.
7	%IPCMP	Length of PID table.
10	%IPCNP	Number of PID's now defined.
11	%IPCTP	Total number of PID's defined since reload.

Table 18-22
IPCF Statistics Per Job
(.GTIPA, GETTAB Table Number 104)

Bit(s)	Mnemonic	Use
18-35	IP.CGC	Count of receives since LOGIN.
1-17	IP.CQD	Count of sends since LOGIN.

Table 18-23
IPCF Flags and Quotas
(.GTIPQ, GETTAB Table Number 107)

Bit(s)	Mnemonic	Use
0	IP.CQX	Quotas disabled.
1	IP.CQQ	Quota has been set.
18-26	IP.CQS	Send quota.
27-35	IP.CQR	Receive quota.

Table 18-24
General Virtual Memory Data
(.GTVM, GETTAB Table Number 113)

Item No.	Mnemonic	Use
0	%VMSWP	Swap count.
1	%VMSCN	Scan count.
2	%VMSIP	Count of swaps in progress.
3	%VMSLE	Count of swap list entries.
4	%VMTTL	Total virtual memory in use.
5	%VMCMX	Maximum value of %VMTTL allowed.
6	%VMRMX	Paging rate max for system.
7	%VMCON	Constant used in swap rate computation.
10	%VMQJB	Job to requeue to PQV (-1 if all).
11	%VMRMJ	Paging rate maximum per job.
12	%VMTLF	Time of last fault.
13	%VMSPF	System page fault counts: lh=not in working set. rh=in working set.
14	%VMSW1	Address of SWP1ST.
15	%VMSW2	Address of SW21ST.
16	%VMSW3	Address of SW31ST.

Table 18-25
Scheduler Statistics
(.GTSST, GETTAB Table Number 115)

Item No.	Mnemonic	Use
0	%SSOSO	The number of the job which was run out of order in order to allow that job to give up a resource for swap out.
1	%SSORJ	The number of jobs which were run out of order in order to have them give up a resource required to run another job.
2	%SSNUL	The amount of swapper null time.
3	%SSLOS	The amount of swapper lost time.
4	%SSRQC	The total number of requeues.
5	%SSICM	The interval of time required to compute minimum core utilization.
6	%SSMSI	The medium term scheduling interval.
7	%SSAJS	The average job size.
10	%SSTQT	The total quota time.
11	%SSEAF	The exponential averaging factor.
12	%SSEAT	Exponentially averaged user time.
13	%SSRSS	The total user run time since the last SCHED. monitor call.

Table 18-26
Special PID Table
(.GTSID, GETTAB Table Number 126)

Item No.	Mnemonic	Use
0	%SIIPC	PID for [SYSTEM] IPCC.
1	%SIINF	PID for [SYSTEM] INFO.
2	%SIQSR	PID for [SYSTEM] QUASAR.
3	%SIMDA	PID for the mountable device allocator.
4	%SITLP	PID for the magtape labeling process.

Table 18-27
ENG./DEQ. Statistics
(.GTENQ, GETTAB Table Number 127)

Item No.	Mnemonic	Use
0	%EQMSS	The maximum string size.
1	%EQNAQ	The number of active queues.
2	%EQESR	The total number of ENQ.S since reload.
3	%EQDSR	The total number of DEQ.S since reload.
4	%EQAPR	Active pooled resources.
5	%EQDEQ	The default ENQ. quota.

APPENDIX A
COMPARISON OF DISK DEVICES

Table A-1
Disk Devices

	RP02	RP03	RP04	RHS04
Disk Drive Capacity	5.12 million words	10.24 million words	20.48 million words	256K words
Transfer Rate	15 μs/word	15 μs/word	5.6 μs/word	4.0 μs/word
Access Time:				
Track-to-track	12 msec	7.5 msec	7 msec	0
Average	35 msec	29 msec	28 msec	8.5 msec
Maximum	60 msec	55 msec	50 msec	8.5 msec
Organization:				
	128 words/sector	128 words/sector	128 words/sector	128 words/sector
	10 sectors/track	10 sectors/track	20 sectors/track	64 sectors/track
	20 tracks/cylinder	20 tracks/cylinder	19 tracks/cylinder	64 tracks/drive
	200 cylinders/pack	400 cylinders/pack	411 cylinders/pack	
Number of Heads	20	20	19	64
Number of Recording Surfaces	20	20	19	1
Number of Disks	11	11	12	1
Number of Drives/Controller	8	8	8	8
Number of Drives/System	32	32	32	16
Maximum Storage/System	0.98 billion characters	1.96 billion characters	3.92 billion characters	12 million characters

APPENDIX B

COMPARISON OF MAGNETIC TAPE SYSTEMS

Table B-1
Magnetic Tape Systems

	TU10A-E	TU10A-F	TU40	TU41	TU70	TU71
Tape Speed	45 ips	45 ips	150 ips	150 ips	200 ips	200 ips
Transfer Rate at:						
220 bpi	9 K char/ sec	9 K char/ sec	30 K char/ sec	30 K char/ sec	–	40 K char/ sec
556 bpi	25 K char/ sec	25 K char/ sec	83.4 K char/ sec	83.4 K char/ sec	–	111.2 K char/ sec
800 bpi	36 K char/ sec	36 K char/ sec	120 K char/ sec	120 K char/ sec	160 K char/ sec	160 K char/ sec
1600 bpi	–	–	–	–	320 K char/ sec	–
Recording Technique	NRZI	NRZI	NRZI	NRZI	PE/NRZI	NRZI
Nominal Inter-Record Gap:						
9-track	0.6 inches	–	0.6 inches	–	0.6 inches	–
7-track	–	0.75 inches	–	0.75 inches	–	0.75 inches
Rewind Time (2400 ft)	195 seconds	195 seconds	66 seconds	66 seconds	45 seconds	45 seconds

APPENDIX C
CARD AND TAPE CODES

Table C-1
ASCII Card Codes

ASCII Character	Octal Code	Card Punches	ASCII Character	Octal Code	Card Punches
NULL	00	12-0-9-8-1	@	100	8-4
CTRL-A	01	12-9-1	A	101	12-1
CTRL-B	02	12-9-2	B	102	12-2
CTRL-C	03	12-9-3	C	103	12-3
CTRL-D	04	9-7	D	104	12-4
CTRL-E	05	0-9-8-5	E	105	12-5
CTRL-F	06	0-9-8-6	F	106	12-6
CTRL-G	07	0-9-8-7	G	107	12-7
CTRL-H	10	11-9-6	H	110	12-8
TAB	11	12-9-5	I	111	12-9
LF	12	0-9-5	J	112	11-1
VT	13	12-9-8-3	K	113	11-2
FF	14	12-9-8-4	L	114	11-3
CR	15	12-9-8-5	M	115	11-4
CTRL-N	16	12-9-8-6	N	116	11-5
CTRL-O	17	12-9-8-7	O	117	11-6
CTRL-P	20	12-11-9-8-1	P	120	11-7
CTRL-Q	21	11-9-1	Q	121	11-8
CTRL-R	22	11-9-2	R	122	11-9
CTRL-S	23	11-9-3	S	123	0-2
CTRL-T	24	9-8-4	T	124	0-3
CTRL-U	25	9-8-5	U	125	0-4
CTRL-V	26	9-2	V	126	0-5
CTRL-W	27	0-9-6	W	127	0-6
CTRL-X	30	11-9-8	X	130	0-7
CTRL-Y	31	11-9-8-1	Y	131	0-8
CTRL-Z	32	9-8-7	Z	132	0-9
ESCAPE	33	0-9-7	]	133	12-8-2
CTRL-\	34	11-9-8-4	\	134	0-8-2

Table C-1
ASCII Card Codes (Cont.)

ASCII Character	Octal Code	Card Punches	ASCII Character	Octal Code	Card Punches
CTRL-]	35	11-9-8-5	]	135	11-8-2
CTRL-¯	36	11-9-8-6	¯	136	11-8-7
CTRL- _	37	11-9-8-7	_	137	0-8-5
SPACE	40		\ -	140	8-1
!	41	12-8-7	a	141	12-0-1
”	42	8-7	b	142	12-0-2
#	43	8-3	c	143	12-0-3
$	44	11-8-3	d	144	12-0-4
%	45	0-8-4	e	145	12-0-5
&	46	12	f	146	12-0-6
’	47	8-5	g	147	12-0-7
(	50	12-8-5	h	150	12-0-8
)	51	11-8-5	i	151	12-0-9
*	52	11-8-4	j	152	12-11-1
+	53	12-8-6	k	153	12-11-2
’	54	0-8-3	l	154	12-11-3
–	55	11	m	155	12-11-4
.	56	12-8-3	n	156	12-11-5
/	57	0-1	o	157	12-11-6
0	60	0	p	160	12-11-7
1	61	1	q	161	12-11-8
2	62	2	r	162	12-11-9
3	63	3	s	163	11-0-2
4	64	4	t	164	11-0-3
5	65	5	u	165	11-0-4
6	66	6	v	166	11-0-5
7	67	7	w	167	11-0-6
8	70	8	x	170	11-0-7
9	71	9	y	171	11-0-8
:	72	8-2	z	172	11-0-9
;	73	11-8-6	[	173	12-0
<	74	12-8-4	|	174	12-11
=	75	8-6	}	175	11-0
>	76	0-8-6	{	176	11-0-1
?	77	0-8-7	DEL	177	12-9-7

NOTE

The ASCII character ESCAPE (octal 33) is also CTRL-[on a terminal.

The ASCII characters and (octal 175 and 176) are treated by the monitor as ALT-MODE and are often considered the same as ESCAPE.

Table C-2
ASCII Codes and BCD Equivalents

ASCII	Character Symbol	BCD	ASCII	Character Symbol	BCD
040	blank	20	074	<	76
041	!	52	075	=	13
042	"	17	076	>	16
043	#	32	077	?	72
044	$	53	100	@	57
045	%	77	101	A	61
046	&	35	102	B	62
047	'	14	103	C	63
050	(	34	104	D	64
051	)	74	105	E	65
052	*	54	106	F	66
053	+	60	107	G	67
054	,	33	110	H	70
055	-	40	111	I	71
056	.	73	112	J	41
057	/	21	113	K	42
060	0	12	114	L	43
061	1	01	115	M	44
062	2	02	116	N	45
063	3	03	117	O	46
064	4	04	120	P	47
065	5	05	121	Q	50
066	6	06	122	R	51
067	7	07	123	S	22
070	8	10	124	T	23
071	9	11	125	U	24
072	:	15	126	V	25
073	;	56	127	W	26

APPENDIX D
COMPARISON OF TERMINALS

Each DECsystem-10 is capable of supporting 511 interactive terminals. A wide variety of EIA and 20 mA Current Loop compatible terminals is supported by the DECsystem. The principle characteristics of some of these terminals is shown in Table D-1.

Table D-1
Terminals

Hard Copy	LA30	LA36
Speed	110–300 baud	110–300 baud
Form width	80 columns	132 columns
	9 7/8 inches wide	3 to 14 7/8 inches wide
Form types	1 part only	Up to 6 part
Character set	128–character keyboard	128–character keyboard
	64–character output	96–character output
	Upper case only	Upper & lower case
Character generation	5x7 dot matrix	7x7 dot matrix
Options	None	Work Surface
		Paper stacking tray
CRT	**VT05**	**VT50**
Speed	110–2400 baud	75–9600 baud
Character/Line	72	80
Lines/Screen	20	12
Character Set	128–character keyboard	64–character keyboard
	64–character output	64–character output
	Upper case only	Upper case only
Character Generation	5x7 dot matrix	5x7 dot matrix

APPENDIX E
ERROR CODES

The error codes in Table E-1 are returned in AC on RUN and GETSEG monitor calls, in the right half of location E + 1 on 4-word argument blocks of LOOKUP, ENTER, and RENAME monitor calls, and in the right half of location E + 3 on extended LOOKUP, ENTER, and RENAME monitor calls.

Table E-1
Error Codes

Symbol	Code	Explanation
ERFNF%	0	File not found, illegal filename (0,*), filenames do not match (UPDATE), or RENAME after a LOOKUP failed.
ERIPP%	1	UFD does not exist on specified file structures. (Incorrect project-programmer number.)
ERPRT%	2	Protection failure or directory full on DTA.
ERFBM%	3	File being modified (ENTER, RENAME).
ERAEF%	4	Already existing filename (RENAME), different filename (ENTER after LOOKUP) or supersede (on a non-superseding ENTER).
ERISU%	5	Illegal sequence of monitor calls (RENAME with neither LOOKUP nor ENTER, or LOOKUP after ENTER).
ERTRN%	6	1. Transmission, device, or data error (RUN, GETSEG only). 2. Hardware-detected device or data error detected while reading the UFD RIB or UFD data block. 3. Software-detected data inconsistency error detected while reading the UFD RIB or file RIB.
ERNSF%	7	Not a saved file (RUN, GETSEG only).
ERNEC%	10	Not enough core (RUN, GETSEG only).
ERDNA%	11	Device not available (RUN, GETSEG only).
ERNSD%	12	No such device (RUN, GETSEG only).
ERILU%	13	Illegal monitor call (GETSEG only). No 2-register relocation capability.
ERNRM%	14	No room on this file structure or quota exceeded (overdrawn quota not considered).
ERWLK%	15	Write-lock error. Cannot write on file structure.
ERNET%	16	Not enough table space in free core of monitor.
ERPOA%	17	Partial allocation only.
ERBNF%	20	Block not free on allocated position.
ERCSD%	21	Cannot supersede an existing directory (ENTER).
ERDNE%	22	Cannot delete a non-empty directory (RENAME).
ERSNF%	23	Sub-directory not found (some SFD in the specified path was not found).

Table E-1 (Cont.)
Error Codes

Symbol	Code	Explanation
ERSLE%	24	Search list empty (LOOKUP or ENTER was performed on generic device DSK and the search list is empty).
ERLVL%	25	Cannot create a SFD nested deeper than the maximum allowed level of nesting.
ERNCE%	26	No file structure in the job's search list has both the no-create bit and the write-lock bit equal to zero and has the UFD or SFD specified by the default or explicit path (ENTER on generic device DSK only).
ERSNS%	27	GETSEG from a locked low segment to a high segment which is not a dormant, active, or idle segment. (Segment not on the swapping space.)
ERFCU%	30	The file cannot be updated.
ERLOH%	31	The low segment overlaps the high segment (GETSEG).
ERNLI%	32	The user is not logged in (RUN).

APPENDIX F

DECSYSTEM-10 AT-A-GLANCE

Table F-1
DECsystem-10

	1040	1050	1055*	1060	1070	1077*	1080
Relative Performance	1	1.5	2.8	2.5	3.5	6.5	5.0
Avg. Number of Users	5-15	10-40	20-70	20-60	30-80	40-100	40-100
No. of CPUs	1	1	2	1	1	2	1
Memory Size in K words (min.-max) (K-1024)	64-256	64-256	128-256	128-4096	128-4096	128-4096	128-4096
No. of Instructions:	366	366	366	378	378	378	386
Instruction Look-ahead	No	No	No	Yes	Yes	Yes	Yes
Virtual Memory	No	No	No	Yes	Yes	Yes	Yes
Memory Interleaving	2 or 4 way	2 or 4 way	2 or 4 way	2 or 4 way	2 or 4 way	2 or 4 way	2 or 4 way
Index Registers	15	15	15 each CPU	4 x 15	4 x 15	4 x 15 each CPU	8 x 15
Accumulators	16	16	16 each CPU	4 x 16	4 x 16	4 x 16 each CPU	8 x 16
Instruction Times (microseconds)							
Fixed Point Add	2.8	2.8	2.8	1.5	1.5	1.5	0.7
Fixed Point Multiply	9.8	9.8	9.8	4.1	4.1	4.1	2.4
Jump	1.5	1.5	1.5	1.1	1.1	1.1	0.5
Single Precision Floating Point Add	9.8	9.8	9.8	3.6	3.6	3.6	1.9
Double Precision Floating Point Add	59.4	59.4	59.4	7.6	7.6	7.6	5.0
I/O Bus Band width (words/second)	200K	200K	200K	370K	370K	370K	370K
Memory Bus Band width (words/second)	4000K	4000K	4000K	4000K	4000K	4000K	4000K

*Dual-processor systems execute two instructions simultaneously.

APPENDIX G
MEMORIES

Table G-1
Core Memories

	MD10	ME10	MF10
Word Size	36 bits plus parity	36 bits plus parity	36 bits plus parity
Minimum Memory Size	64K	16K	32K
Maximum Memory Size			
KA10	256K	256K	256K
KI10	256K	256K	1024K
KL10	–	–	1024K
Module Sizes			
16K words	No	Yes	No
32K words	Yes	No	Yes
64K words	Yes	No	Yes
Read Access Time			
Typical	800 ns	550 ns	550 ns
Maximum	880 ns	610 ns	610 ns
Cycle Time	1.8 μs	1.0 μs	950 ns
Interleaving	2 or 4 way	2 or 4 way	2 or 4 way
	External Only	External Only	External Only

APPENDIX H
FILE RETRIEVAL POINTERS

Sequential and random file access are handled more efficiently by the monitor if all the information describing the file can be kept in core at once. To understand this effect, it is necessary to know how the monitor accesses files.

With each named file, UFD, and MFD, the monitor writes a special block containing necessary information needed to retrieve the data blocks that constitute the file. This block is called a retrieval information block, or RIB.

Retrieval pointers in the RIB describe contiguous blocks of file storage space called groups. Each pointer occupies one word and has one of three forms:

1. A group pointer
2. An EOF pointer
3. A change of unit pointer.

H.1 A GROUP POINTER

A group pointer has three fields:

1. A cluster count
2. A folded checksum
3. A cluster address within a unit. The width of each field may be specified at ONCE-only time; therefore, the same code can handle a wider variety of sizes of devices.

The cluster count determines the number of consecutive clusters that can be described by one pointer. The folded checksum is computed for the first word of the first block of the group. Its main purpose is to catch hardware or software errors when the wrong block is read. The folded checksum is not a check on the hardware parity circuitry. The size of the cluster address field depends on the largest unit size in the file structure and on the cluster size. A cluster address is converted to a logical block address by multiplying the number of blocks per cluster.

H.1.1 Folded Checksum Algorithm

This algorithm takes the low order n-bit byte, repeatedly adds it to the upper part of the word, and then shifts. The code is:

```
LOOP:  ADD     T1,T
       LDB     T,LOW ORDER N BITS OF T1
       LSH     T1,-N                    ,RIGHT SHIFT BY N BITS
       JUMPN   T1,LOOP
       DONE                             ,ANSWER IN T
```

This scheme eliminates the usual overflow problem assosicated with folded checksums and terminates as soon as there are no more bits to add.

H.2 END-OF-FILE POINTER

The EOF is indicated by a zero word.

H.3 CHANGE OF UNIT POINTER

A file structure may comprise more than one unit; therefore, the retrieval information block must indicate which unit the logical block is on. Because a file can start on one device and move to another, a method of indicating a change from one unit to another in the middle of the file is necessary. To show this movement, a zero count field indicates that the right half of the word specifies a change in unit. A zero count field contains a unit number with respect to the file structure. The first retrieval pointer, with respect to the RIB, always specifies a unit number. Bit 18 is 1 to guarantee that the word is non-zero; otherwise, it might be confused with an EOF pointer.

H.4 DEVICE DATA BLOCK

The monitor keeps a copy of up to six retrieval pointers in core at once. Therefore, if a file is allocated in six or less contiguous blocks (i.e., described in six or less pointers), all of the retrieval information can be kept in core and no additional accesses to the RIB are necessary.

H.5 ACCESS BLOCK

For each active file, the monitor keeps eight words of storage called an access block. These access blocks remain dormant in monitor core after a file is closed and are reclaimed only when the core space is needed. Therefore, if a 4-word LOOKUP is done after a file has been active, access to the UFD and RIB blocks will not require I/O.

APPENDIX I
DATA COMMUNICATIONS

Table I-1
Communication Systems

	Number supported by standard software	Maximum number of lines per DECsystem-10	Maximum throughput	Transmission speeds	Data modes	Line types		Attachment to DECsystem-10 via	Comments
						Local	Modem control		
DC10 Data Line Scanner	1	128	3000 chars/sec	up to 2400 baud	full duplex; full duplex with local copy; half duplex	yes	yes	I/O bus	DC10 will also support Telegraph (long distance lines); 8-bit character
DS10 Single-line Synchronous Interface	2	2	9600 bits/sec	up to 9600 bps	full duplex	yes	yes	I/O bus	7-, 8-bit character
DC72 Remote Station	32	16 each station plus operator's console	9600 bits/sec	up to 2400 baud	full duplex; full duplex with local copy	yes	no	DS10 or DC75	Handles 300-card/minute reader and up to 245-line/minute printer; PDP-8/E processor; 8-bit character
DC75 Synchronous Communications Multiplexer System	1	64	160K bits/sec	up to 9600 bps	full duplex	yes	no	I/O bus and memory bus	Up to four PDP-11 processors; will also support up to three DC76-D asynchronous multiplexers
DC76 Asynchronous Communications Multiplexer System	1	512	6000 chars/sec	up to 9600 baud	full duplex; full duplex with local copy	yes	yes	I/O bus and memory bus	Programmable character of 7 or 8 bits; auto baud rate detection; up to four PDP-11 processors; will also support up to three DC75-D synchronous multiplexers

APPENDIX J
UUOSYM.MAC

A complete copy of UUOSYM.MAC is in this chapter.

```
        SUBTTL  UUO PARAMETERS /DAL

        .XCREF
IFDEF %..C,<IFE %..C,<  .CREF
                        TAPE    >>

IFNDEF %..C,<
        UNIVERSAL UUOSYM -- UUO SYMBOLS FOR USER PROGRAMS
        .DIRECTIVE      .NOBIN
        SEARCH          MACTEN
%%MACT==:%%MACT         ;SHOW VERSION
>

;***Copyright (C) 1971,1972,1973,1974,1975
            Digital Equipment Corp., Maynard, Mass.***
;THIS IS THE DEFINITION FILE OF ALL PUBLISHED  MONITOR
;UUO PARAMETERS WHICH DO NOT REQUIRE THE JACCT PRIVILEGE. IT
;EXCLUDES VARIOUS PARAMETERS WHICH CAN BE "SPYED" IF THE USER
;HAS SPY PRIVILEGE SINCE THOSE LOCATIONS CHANGE WITH MONITOR
;DEVELOPMENT. THESE SYMBOLS ARE ALL DEFINED IN THE
;MONITOR CALLS MANUAL.

;THIS DERIVES FROM THE OLD C.MAC FILE. IT IS ONLY A UNIVERSAL


;VERSION INFORMATION
        UUOWHO==0               ;LAST MODIFIER
        UUOVER==11              ;MAJOR VERSION
        UUOMIN==0               ;MINOR VERSION
        UUOEDT==225             ;EDIT LEVEL

        .CREF
        SALL
```

```
        SUBTTL  TABLE OF CONTENTS

;                TABLE OF CONTENTS FOR UUOSYM
;
;
;                          SECTION                                  PAGE
;    1. TABLE OF CONTENTS.....................................      2
;    2. REVISION HISTORY......................................      4
;    3. NAMING CONVENTIONS....................................      6
;    4. UPDATE AND CHECKOUT INSTRUCTIONS......................      7
;    5. ALL JOBDAT SYMBOLS....................................      8
;    6. GTMSG. MACRO..........................................      9
;    7. OPDEFS
;          7.1   BASIC UUOS...................................     10
;          7.2   MTAPE FUNCTIONS..............................     11
;          7.3   TTCALL FUNCTIONS............................     11
;          7.4   CALLI FUNCTIONS.............................     12
;    8. GETTAB CONSTITUENTS...................................     15
;    9. MISC. NON-I/O
;          9.1   TMPCOR.......................................     33
;          9.2   LOCK.........................................     33
;          9.3   RTTRP........................................     33
;          9.4   JOBSTS.......................................     34
;          9.5   HIBER........................................     34
;          9.6   APRENB.......................................     34
;          9.7   SAVE/GET LOCATIONS...........................     35
;          9.8   SETUUO.......................................     36
;          9.9   SCHED........................................     37
;          9.10  ATTACH.......................................     40
;   10. UNIVERSAL DEVICE INDEX...............................     41
;   11. .JBINT INTERCEPT BLOCK...............................     42
;   12. PSI SOFTWARE INTERRUPT SYSTEM........................     43
;   13. IPCF INTERPROCESS COMMUNICATION FACILITY.............     46
;   14. PAGE AND VM  VIRTUAL MEMORY FACILITY.................     49
;   15. DAEMON CALLS.........................................     51
;   16. METER UUO............................................     53
;   17. ENQUEUE AND DEQUEUE SYMBOLS..........................     55
```

```
;    18. MISC. I/O
;         18.1    DEVCHR..........................................  61
;         18.2    DEVTYP..........................................  62
;         18.3    MTCHR...........................................  63
;         18.4    TAPOP...........................................  64
;         18.5    WHERE...........................................  68
;         18.6    CAL11...........................................  68
;         18.7    GETLCH AND TRMOP................................  69
;         18.8    GETSTS AND SETSTS...............................  71
;         18.9    OPEN AND CLOSE..................................  72
;         18.10   FILOP...........................................  73
;         18.11   BUFFER HEADER FORMATS...........................  74
;         18.12   MVHDR...........................................  74
;         18.13   CNECT.,SENSE., CLRST............................  75
;         18.14   DEVLNM..........................................  76
;         18.15   DEVSIZ..........................................  76
;         18.16   MTAID...........................................  76
;    19. DISK UUOS
;         19.1    DSKCHR..........................................  77
;         19.2    CHKACC..........................................  78
;         19.3    DISK............................................  79
;         19.4    JOBSTR..........................................  80
;         19.5    GOBSTR..........................................  80
;         19.6    SUSET...........................................  80
;         19.7    PATH............................................  81
;         19.8    STRUUO..........................................  82
;    20. LOOKUP/ENTER/RENAME.......................................  84
```

```
        SUBTTL  REVISION HISTORY

;%3(67) MAY, 1972

;70     CORRECT MOVX TO INCLUDE <>
;71     ADD MACRO STORE TO GENERATE BLT
;72     MAKE USEABLE AS A UNIVERSAL FILE
;73     CORRECT MOVX, TXYY TO HANDLE RELOCATABLE MASKS
;74     HAVE MOVX GENERATE HRLOI, HRROI
;75     HAVE TXY GENERATE ORCMI, ANDI, EQVI
;76     ADD CAXYY, ADDX, ETC.
;77     ADD PJRSTF
;100    UPDATE TO 50434 (505) BY ADDING ADDITIONAL
;       SYMBOLS; CHANGE %LDSXS TO %LDNEW, F%ABLB TO F%APLB
;101    ENHANCE ADDX, ETC., TO NOTICE SMALL NEGATIVES
;102    ADD PATH AREA
;103    ADD METER, BITS AND PIECES
;104    ADD ALL OLD CALLI MNEMONICS FOR DDT.SAV
;105    ADD SUBTITLES AND INDEX
;106    UPDATE RESPONSE SUBTABLE
;107    CORRECT BUGS IN UNIVERSAL SETUP
;110    ADD OPDEF FOR PORTAL
;%4(110) JULY, 1972

;111    CHANGE GL.NEC TO GL.LCP (SPR 10-7553)
;112    CHANGE SY.ERR AND SY.CCO TO %SYERR AND %SYCCO.
;113    CORRECT BUG IN STORE MACRO TO ALLOW MORE GENERAL USE
;114    ADD 5.06 DEFINITIONS
;115    COMPLETE .RBSTS BITS.
;%5(115) NOV 72

;116    SUPPORT DATE75 BY CHANGING RB.ACD AND ADDING RB.CRX
;117    CLEAN UP PAGE 1 LISTING
;120    ADD .STDFL
;121    ADD LKNEM%
;122    (10-9627) ALLOW FLAG=0 IN TX?? MACROS
;123    (10-9725) CHANGE CAXNE TO CAXN
;124    ADD OPEN BLOCK
;125    ADD MACROS MASK. RGHBT. LFTBT. FILIN. ALIGN. TXND. TXNI. JUMPI. JUMPN.
;126    ADD MACRO BTSWP.
;127    ADD INFO-REDEF.
;130    ADD SN%LOK, CORRECT .BFSTS
;131    (10-11609) FIX STORE MACRO FOR RELOC. 0
;%6(131) DEC 73
```

```
;132     UPDATE TO 50644 (6.01/5.07) MCO 4072
;133     CORRECT BUG IN %FT??? DEFINITIONS
;134     IF UNIVERSAL, MAKE .JB41, ETC., BE EXTERNAL
;135     ADD ALL 6.01 JOBDAT SYMBOLS
;136     ADD .SGDDT
;137     ADD GTMSG.
;140     UPDATE TO 50645
;141     UPDATE TO 50646
;142     UPDATE TO 50650
;143     CORRECT MISSING .CREF IN TX? MACROS
;144     UPDATE TO 50657
;145     RESERVE Q SYMBOLS TO QPRM.UNV; %DIGITS AND U..??? TO E.UNV
;146     UPDATE TO 50660
;147     ADD LOCK UUO BITS, DEVLNM+DEVSIZ+DISK. ERRORS
;150     UPDATE TO 50662
;151     RESERVE SYMBOLS OF THE FORM ?.???? TO OTHER FILES
;152     RESERVE FS.M??, FX.???, TS.???, AND .FX??? TO SCNMAC.UNV
;153     ADD ST.W??, AND "ALL" BITS SETS: JW.WAL, ST.WAL, JS.PAL, RB.ERR
;154     ADD .SG41
;155     ADD BOXES FOR FUNNY FORMATS; RE-ORDER CL.??? FOR CONSISTENCY
;156     CORRECT DEFINITIONS OF .BFSTS, .BFHDR, AND .BFCNT
;157     OBSOLETE IO.FCS. ADD IO.LEM
;160     ADD .INFIN AND .MINFI
;161     UPDATE TO 50664
;162     UPDATE TO 50666
;163     DEFINE INSVL.
;%7(163) MAY 74

;201     SPLIT INTO MACTEN.MAC AND UUOSYM.MAC
;202     ADD REMAINING 5.07/6.01 SYMBOLS
;203     ADD WORDS IN .FSDEF PRIVILEGED FUNCTION OF STRUUO
;204     UPDATE TO 5.07A/6.01A
;205     CHANGE WAY UUOS ARE DEFINED
;206     CORRECT BUG IN REFERENCE TO VRSN.
;%10(206) MARCH 1975

;207     FIX UP SUBTTL STATEMENTS SO TOC OUTPUT CAN BE USED FOR
;         TABLE OF CONTENTS.
;210     DELETE SYMBOLS FOR CHANNEL DATA BLOCK SINCE CDB IS ONLY
;        AVAIL. VIA SPY UUO
;211-225 ADD 6.02 SYMBOLS TO MCO 5478
```

```
        SUBTTL  NAMING CONVENTIONS

;PATTERN        USAGE

; .GGSSS        NUMBER OF GENERAL CATEGORY GG, SPECIFIC USE SSS
; GG.SSS        BYTE OF GENERAL CATEGORY GG, SPECIFIC USE SSS
; UUUUU.        UUO OR FUNCTION OR MACRO

; %GGSSS        GETTAB INDEX (WORD,,TABLE)
; GG%SSS        BYTE IN A SPECIFIC GETTAB
; GGEEE%        ERROR CODE OF CATEGORY GG, SPECIFIC ERROR EEE


;SPECIAL CASES--

; F%AAAA        RH=BYTE OF FEATURE TEST NAMED FTAAAA
;                       IN LH=FEATURE DEFINED
;                       IN RH=FEATURE TURNED ON
;               LH=LH OF GETTAB IN .GTFET CONTAINING INFO


;RESERVED FOR OTHER THAN C.MAC, C.UNV--

;ALL SYMBOLS CONTAING $ ARE RESERVED TO THE USER
;       (CUSTOMER, HIS USER, OR SPECIFIC PROGRAMS)

; ?.????        RESERVED TO OTHER PARAMETER FILES

;ALL SYMBOLS OF THE FORMS:  QABCDE, .QABCD, %QABCD
;       (I.E, WITH FIRST ALPHABETIC "Q") ARE RESERVED TO
;       QPRM.MAC, QPRM.UNV FOR THE QMANGR PARAMETER AREA, ETC.

;ALL SYMBOLS OF THE FORMS: %NNNNN, U..NNN, E..AAA
;       ARE RESERVED TO E.MAC, E.UNV FOR THE ERROR HANDLER

;ALL SYMBOLS OF THE FORMS: FS.M??, FX.???, TS.???, AND .FX???
;       ARE RESERVED TO SCNMAC.MAC, SCNMAC.UNV FOR SCAN AND WILD
```

```
        SUBTTL  UPDATE AND CHECKOUT INSTRUCTIONS

;1.  COMPARE CREFS OF F% IN C VS. FT IN DATMAN
;2.  VERIFY NO $ IN CREF IN C AND THAT ONLY LEGAL PATTERNS EXIST
;3.  UPDATE CALLI TABLE FROM UUOCON
;4.  UPDATE GETTAB LIST OF TABLES (.GTABC) FROM UUOCON. DEFINE
;       ENTRIES/BYTES IN NEW TABLES
;5.  FIND NEW ENTRIES IN OLD TABLES (ESP. .GTCNF, .GTLVD, .GTSYS,
;       .GTCOC, .GTCOV)
;6.  FIND NEW BYTES IN OLD WORDS (ESP. .GTPRV, %CNDBG, .GTWCH,
;       %CNST2)
;7.  FIND NEW ERRORS AND FUNCTIONS FOR UUOS (ESP. DEVTYP, OPEN/CLOSE,
;       PATH., LOOKUP)
;8.  FIND NEW DAEMON FUNCTIONS, ERRORS, FORMATS.
;9.  VERIFY NO DUPLICATES BY SCANNING CREF FOR ONLY SINGLE REFERENCES
;10. VERIFY THAT ALL USER JOBDAT SYMBOLS APPEAR
;11. VERIFY THAT SYMBOLS RESERVED TO OTHER FILES DO NOT APPEAR:
;       ?.????
;       Q?????, %Q????, .Q????
;       %NNNNN, U..NNN, E..???
;       FS.M??, FX.???, TS.???, .FX???
```

```
SUBTTL  ALL JOBDAT SYMBOLS

EXTERN  .JBAPR,.JBBLT,.JBCHN,.JBCNI,.JBCOR
EXTERN  .JBDA,.JBDDT,.JBERR,.JBFF,.JBH41,.JBHCR,.JBHDA,.JBHGA
EXTERN  .JBHGH,.JBHNM,.JBHRL,.JBHRN,.JBHSA,.JBHSM,.JBHVR,.JBINT,.JBOPC
EXTERN  .JBOVL,.JBPFH,.JBPFI,.JBREL,.JBSA,.JBSYM,.JBTPC,.JBUSY,.JBUUO
EXTERN  .JB41,.JBCST,.JBOPS,.JBREN,.JBVER
```

```
        SUBTTL  GTMSG. MACRO

;MACRO TO GET ERROR MESSAGE CODE AND CLEAN IT UP

;CALL:  GTMSG.  AC
;       WHERE AC WILL END UP WITH IT IN BYTE JW.WMS
;       DEFAULT IS /MESSAGE:(PREFIX,FIRST)
;       IF /MESSAGE:CONTINUATION, THEN /MESSAGE:FIRST
;         IS ASSUMED

DEFINE  GTMSG.  (AC),<
        .XCREF                          ;;SUPPRESS REDUNDANT CREF
        HRROI   AC,.GTWCH               ;;IT'S IN THE WATCH TABLE
        .CREF
        GETTAB  AC,                     ;;GET FROM MONITOR
        .XCREF
          MOVEI AC,0                    ;;DEFAULT TO 0
        TXNN    AC,JW.WMS               ;;IF 0,
        TXO     AC,.JWWPO_<ALIGN. (JW.WMS)>  ;; DEFAULT TO PREFIX,FIRST
        TXNE    AC,JW.WCN               ;;IF /MESSAGE:CONTINUATION
        TXO     AC,JW.WFL               ;; DEFAULT TO /MESSAGE:FIRST
        .CREF
```

```
;OPDEF THE UUOS SO THEY APPEAR IN THE OPCODE LISTING

        OPDEF   HALT    [JRST 4,]
                        ;40B8 IS OBSOLETE (CALL)
        OPDEF   INIT    [41B8]
                        ;42-46B8 ARE RESERVED TO CUSTOMERS
        OPDEF   CALLI   [47B8]  ;(PURGED LATER)
        OPDEF   OPEN    [50B8]
        OPDEF   TTCALL  [51B8]  ;(PURGED LATER)
                        ;52-54B8 ARE RESERVED TO DEC
        OPDEF   RENAME  [55B8]
        OPDEF   IN      [56B8]
        OPDEF   OUT     [57B8]
        OPDEF   SETSTS  [60B8]
        OPDEF   STATO   [61B8]
        OPDEF   GETSTS  [62B8]
        OPDEF   STATZ   [63B8]
        OPDEF   INBUF   [64B8]
        OPDEF   OUTBUF  [65B8]
        OPDEF   INPUT   [66B8]
        OPDEF   OUTPUT  [67B8]
        OPDEF   CLOSE   [70B8]
        OPDEF   RELEAS  [71B8]
        OPDEF   MTAPE   [72B8]  ;(PURGED LATER)
        OPDEF   UGETF   [73B8]
        OPDEF   USETI   [74B8]
        OPDEF   USETO   [75B8]
        OPDEF   LOOKUP  [76B8]
        OPDEF   ENTER   [77B8]
        OPDEF   UJEN    [100B8]
```

```
SUBTTL  OPDEFS -- MTAPE FUNCTIONS

OPDEF   MTWAT,  [MTAPE 0]       ;WAIT FOR POSITIONING
OPDEF   MTREW,  [MTAPE 1]       ;REWIND
OPDEF   MTEOF,  [MTAPE 3]       ;WRITE END OF FILE
OPDEF   MTSKR,  [MTAPE 6]       ;SKIP RECORD
OPDEF   MTBSR,  [MTAPE 7]       ;BACKSPACE RECORD
OPDEF   MTEOT,  [MTAPE 10]      ;SKIP TO END OF TAPE
OPDEF   MTUNL,  [MTAPE 11]      ;REWIND AND UNLOAD
OPDEF   MTBLK,  [MTAPE 13]      ;BLANK TAPE
OPDEF   MTSKF,  [MTAPE 16]      ;SKIP FILE
OPDEF   MTBSF,  [MTAPE 17]      ;BACKSPACE FILE
OPDEF   MTDEC,  [MTAPE 100]     ;DEC 9-CHANNEL
OPDEF   MTIND,  [MTAPE 101]     ;INDUSTRY STANDARD 9-CHANNEL
OPDEF   MTLTH,  [MTAPE 200]     ;LOW THRESHOULD

SUBTTL  OPDEFS -- TTCALL FUNCTIONS

OPDEF   INCHRW  [TTCALL 0,]     ;INPUT CHAR AND WAIT
OPDEF   OUTCHR  [TTCALL 1,]     ;OUTPUT CHAR
OPDEF   INCHRS  [TTCALL 2,]     ;INPUT CHAR AND SKIP
OPDEF   OUTSTR  [TTCALL 3,]     ;OUTPUT STRING
OPDEF   INCHWL  [TTCALL 4,]     ;INPUT CHAR WAIT, LINE
OPDEF   INCHSL  [TTCALL 5,]     ;INPUT CHAR SKIP, LINE
OPDEF   GETLCH  [TTCALL 6,]     ;GET LINE CHARS
OPDEF   SETLCH  [TTCALL 7,]     ;SET LINE CHARS
OPDEF   RESCAN  [TTCALL 10,]    ;RESET INPUT LINE
OPDEF   CLRBFI  [TTCALL 11,]    ;CLEAR INPUT BUFFER
OPDEF   CLRBFO  [TTCALL 12,]    ;CLEAR OUTPUT BUFFER
OPDEF   SKPINC  [TTCALL 13,]    ;SKIP IF CHAR IN INPUT
OPDEF   SKPINL  [TTCALL 14,]    ;SKIP IF LINE IN INPUT
OPDEF   IONEOU  [TTCALL 15,]    ;OUTPUT IMAGE CHAR
```

```
SUBTTL  OPDEFS -- CALLI FUNCTIONS

OPDEF   LIGHTS  [CALLI -1]      ;DISPLAY IN LIGHTS
OPDEF   RESET   [CALLI 0]       ;RESET PROGRAM
OPDEF   DDTIN   [CALLI 1]       ;DDT MODE CONSOLE INPUT
OPDEF   SETDDT  [CALLI 2]       ;SET .JBDDT
OPDEF   DDTOUT  [CALLI 3]       ;DDT MODE CONSOLE OUTPUT
OPDEF   DEVCHR  [CALLI 4]       ;GET DEVICE CHARACTERISTICS
OPDEF   DDTGT   [CALLI 5]       ;(HISTORICAL)
OPDEF   GETCHR  [CALLI 6]       ;SAVE AS 4
OPDEF   DDTRL   [CALLI 7]       ;(HISTORICAL)
OPDEF   WAIT    [CALLI 10]      ;WAIT FOR DEVICE INACTIVE
OPDEF   CORE    [CALLI 11]      ;ALLOCATE CORE
OPDEF   EXIT    [CALLI 12]      ;STOP JOB
OPDEF   MONRT.  [CALLI 1,12]    ;MONITOR RETURN
OPDEF   UTPCLR  [CALLI 13]      ;CLEAR DECTAPE DIRECTORY
OPDEF   DATE    [CALLI 14]      ;GET DATE
OPDEF   LOGIN   [CALLI 15]      ;LOGIN
OPDEF   APRENB  [CALLI 16]      ;ENABLE TRAPS
OPDEF   LOGOUT  [CALLI 17]      ;LOGOUT OR EXIT
OPDEF   SWITCH  [CALLI 20]      ;READ CONSOLE SWITCHES
OPDEF   REASSI  [CALLI 21]      ;REASSIGN DEVICES
OPDEF   TIMER   [CALLI 22]      ;READ TIME OF DAY IN TICKS
OPDEF   MSTIME  [CALLI 23]      ;READ TIME OF DAY IN MSEC.
OPDEF   GETPPN  [CALLI 24]      ;RETURN PPN OF THIS JOB
OPDEF   TRPSET  [CALLI 25]      ;ENABLE I/O MODE
OPDEF   TRPJEN  [CALLI 26]      ;(ILLEGAL)
OPDEF   RUNTIM  [CALLI 27]      ;RETURN MSEC TIME THIS JOB
OPDEF   PJOB    [CALLI 30]      ;RETURN JOB NUMBER
OPDEF   SLEEP   [CALLI 31]      ;SLEEP
OPDEF   SETPOV  [CALLI 32]      ;(HISTORICAL)
OPDEF   PEEK    [CALLI 33]      ;READ ABSOL. CORE ADDRESS
OPDEF   GETLIN  [CALLI 34]      ;GET NAME OF TERMINAL
OPDEF   RUN     [CALLI 35]      ;RUN PROGRAM
OPDEF   SETUWP  [CALLI 36]      ;DIDDLE USER WRITE PROTECT
OPDEF   REMAP   [CALLI 37]      ;REMAP LOW TO HIGH SEG
OPDEF   GETSEG  [CALLI 40]      ;GET NEW HIGH SEG
OPDEF   GETTAB  [CALLI 41]      ;READ MONITOR TABLE
OPDEF   SPY     [CALLI 42]      ;SPY ON MONITOR
OPDEF   SETNAM  [CALLI 43]      ;CHANGE NAME OF PROGRAM
OPDEF   TMPCOR  [CALLI 44]      ;ACCESS TMPCOR
OPDEF   DSKCHR  [CALLI 45]      ;RETURN DISK CHARACTERISTICS
OPDEF   SYSSTR  [CALLI 46]      ;RETURN ALL S/L
OPDEF   JOBSTR  [CALLI 47]      ;RETURN JOB S/L
```

```
OPDEF   STRUUO  [CALLI 50]      ;DIDDLE STRS
OPDEF   SYSPHY  [CALLI 51]      ;RETURN ALL DISK UNITS
OPDEF   FRECHN  [CALLI 52]      ;(FUTURE)
OPDEF   DEVTYP  [CALLI 53]      ;RETURN DEVICE PROPERTIES
OPDEF   DEVSTS  [CALLI 54]      ;RETURN LAST CONI
OPDEF   DEVPPN  [CALLI 55]      ;RETURN PPN OF ERSATZ DEVICE
OPDEF   SEEK    [CALLI 56]      ;SEEK DISK
OPDEF   RTTRP   [CALLI 57]      ;CONNECT RT DEVICE
OPDEF   LOCK    [CALLI 60]      ;LOCK IN CORE
OPDEF   JOBSTS  [CALLI 61]      ;RETURN JOB STATUS
OPDEF   LOCATE  [CALLI 62]      ;CHANGE LOGICAL STATION
OPDEF   WHERE   [CALLI 63]      ;RETURN PHYSICAL STATION
OPDEF   DEVNAM  [CALLI 64]      ;RETURN PHYSICAL NAME
OPDEF   CTLJOB  [CALLI 65]      ;RETURN CONTROLLING JOB
OPDEF   GOBSTR  [CALLI 66]      ;RETURN NEXT JOB S/L
OPDEF   ACTIVAT [CALLI 67]      ;(FUTURE)
OPDEF   DEACTI  [CALLI 70]      ;(FUTURE)
OPDEF   HPQ     [CALLI 71]      ;SET HPQ RUN
OPDEF   HIBER   [CALLI 72]      ;SLEEP ON EVENT
OPDEF   WAKE    [CALLI 73]      ;WAKE SOME JOB
OPDEF   CHGPPN  [CALLI 74]      ;CHANGE PPN
OPDEF   SETUUO  [CALLI 75]      ;GENERAL SET SYS PARAMS
OPDEF   DEVGEN  [CALLI 76]      ;(FUTURE)
OPDEF   OTHUSR  [CALLI 77]      ;CHECK FOR ANOTHER USER
OPDEF   CHKACC  [CALLI 100]     ;VALIDATE FILE ACCESS
OPDEF   DEVSIZ  [CALLI 101]     ;GET BUFFER SIZES
OPDEF   DAEMON  [CALLI 102]     ;REQUEST DAEMON FUNCTION
OPDEF   JOBPEK  [CALLI 103]     ;READ/WRITE ANOTHER JOB
OPDEF   ATTACH  [CALLI 104]     ;ATTACH TTY/JOB
OPDEF   DAEFIN  [CALLI 105]     ;DAEMON INDICATES DONE
OPDEF   FRCUUO  [CALLI 106]     ;FORCE COMMAND ON JOB
OPDEF   DEVLNM  [CALLI 107]     ;SET LOGICAL NAME
OPDEF   PATH.   [CALLI 110]     ;DEAL WITH DIRECTORY PATHS
OPDEF   METER.  [CALLI 111]     ;PERFORMANCE METERING
OPDEF   MTCHR.  [CALLI 112]     ;GET MAG TAPE CHARACTERISTICS
OPDEF   JBSET.  [CALLI 113]     ;SETUUO FOR ARBITRARY JOB
OPDEF   POKE.   [CALLI 114]     ;CHANGE MONITOR
OPDEF   TRMNO.  [CALLI 115]     ;JOB'S TERMINAL NUMBER
OPDEF   TRMOP.  [CALLI 116]     ;TERMINAL OPERATION
OPDEF   RESDV.  [CALLI 117]     ;RESET CHANNEL
```

```
OPDEF   UNLOK.  [CALLI 120]     ;UNLOCK A LOCKED JOB
OPDEF   DISK.   [CALLI 121]     ;MISC. DISK FUNCTIONS
OPDEF   DVRST.  [CALLI 122]     ;RESTRICT DEVICE TO OPER
OPDEF   DVURS.  [CALLI 123]     ;UNRESTRICT DEVICE
OPDEF   XTTSK.  [CALLI 124]     ;DA28C FUNCTIONS
OPDEF   CAL11.  [CALLI 125]     ;DL10 MULTI-FUNCTION
OPDEF   MTAID.  [CALLI 126]     ;SET MAG TAPE ID
OPDEF   IONDX.  [CALLI 127]     ;RETURN UNIVERSAL DEVICE INDEX
OPDEF   CNECT.  [CALLI 130]     ;CONNECT TO MPX
OPDEF   MVHDR.  [CALLI 131]     ;MOVE BUFFER HEADER
OPDEF   ERLST.  [CALLI 132]     ;ERROR LIST
OPDEF   SENSE.  [CALLI 133]     ;SENSE
OPDEF   CLRST.  [CALLI 134]     ;CLEAR STATUS
OPDEF   PIINI.  [CALLI 135]     ;INITIALIZE SOFT. PI SYS
OPDEF   PISYS.  [CALLI 136]     ;MANIPULATE SOFT. PI SYS
OPDEF   DEBRK.  [CALLI 137]     ;DISMISS SOFT. PI INTER.
OPDEF   PISAV.  [CALLI 140]     ;SAVE SOFT. PI SYS
OPDEF   PIRST.  [CALLI 141]     ;RESTORE SOFT. PI SYS
OPDEF   IPCFR.  [CALLI 142]     ;IPCF READ
OPDEF   IPCFS.  [CALLI 143]     ;IPCF SEND
OPDEF   IPCFQ.  [CALLI 144]     ;IPCF QUERY
OPDEF   PAGE.   [CALLI 145]     ;PAGING UUO
OPDEF   SUSET.  [CALLI 146]     ;SUPER USETI/O
OPDEF   COMPT.  [CALLI 147]     ;CALL COMPATABILITY PACKAGE
OPDEF   SCHED.  [CALLI 150]     ;SCHEDULING UUO
OPDEF   ENQ.    [CALLI 151]     ;ENQUEUE
OPDEF   DEQ.    [CALLI 152]     ;DEQUEUE
OPDEF   ENQC.   [CALLI 153]     ;ENQ/DEQ CONTROL
OPDEF   TAPOP.  [CALLI 154]     ;MAG TAPE OPERATIONS
OPDEF   FILOP.  [CALLI 155]     ;FILE OPEN/CLOSE
OPDEF   CAL78.  [CALLI 156]     ;DAS-78 DIAGNOSTICS
OPDEF   NODE.   [CALLI 157]     ;RESERVED
OPDEF   ERRPT.  [CALLI 160]     ;FOR DAEMON ERROR REPORTING
OPDEF   ALLOC.  [CALLI 161]     ;ALLOCATE A DEVICE
OPDEF   PERF.   [CALLI 162]     ;KL10 PERFORMANCE ANALYSIS
```

```
        SUBTTL  GETTAB CONSTITUENTS

.GTSTS==0           ;JOB STATUS
.GTADR==1           ;JOB RELOCATION AND PROTECTION
.GTPPN==2           ;PROJ-PROG NUMBER
.GTPRG==3           ;PROGRAM NAME
.GTTIM==4           ;TOTAL RUN TIME IN TICKS
.GTKCT==5           ;KILO-CORE TICKS
.GTPRV==6           ;PRIV WORD
        JP.IPC==1B0       ;IPCF PRIVILEGED FUNCTIONS
        JP.DPR==3B2       ;HIGHEST DISK PRIORITY
        JP.MET==1B3       ;METER UUO
        JP.POK==1B4       ;POKE MONITOR
        JP.CCC==1B5       ;CHANGE CPU SPECIFICATION
        JP.HPQ==17B9      ;HI PRIORITY QUEUE
        JP.NSP==1B10      ;UNSPOOL DEVICES
        JP.ENQ==1B11      ;ENQ./DEQ. PRIVS
        JP.RTT==1B13      ;RTTRP UUO
        JP.LCK==1B14      ;LOCK UUO
        JP.TRP==1B15      ;TRAPSET UUO
        JP.SPA==1B16      ;SPY ON ALL CORE
        JP.SPM==1B17      ;SPY ON MONITOR
.GTSWP==7           ;SWAPPING POINTERS
.GTTTY==10          ;TTY TABLE
.GTCNF==11          ;CONFIGURATION
        %CNFG0==0,,11     ;NAME OF SYSTEM
        %CNFG1==1,,11     ; ..
        %CNFG2==2,,11     ; ..
        %CNFG3==3,,11     ; ..
        %CNFG4==4,,11     ; ..
        %CNDT0==5,,11     ;DATE OF SYSTEM
        %CNDT1==6,,11     ; ..
        %CNTAP==7,,11     ;NAME OF SYSTEM DEVICE
        %CNTIM==10,,11    ;TIME OF DAY
        %CNDAT==11,,11    ;DATE IN BINARY
        %CNSIZ==12,,11    ;SYSTEM MEMORY SIZE
        %CNOPR==13,,11    ;NAME OF OPR TTY
        %CNDEV==14,,11    ;LH = DDB CHAIN
        %CNSJN==15,,11    ;LH=-SEGN, RH=JOBN NUMBERS
        %CNTWR==16,,11    ;NON-ZERO IS DUAL SEGMENTS
```

```
%CNSTS==17,,11   ;SYSTEM STATES
        ST%DSK==1B0      ;DISK SYSTEM
        ST%SWP==1B1      ;SWAPPING SYSTEM
        ST%LOG==1B2      ;LOGIN
        ST%FTT==1B3      ;FULL DUPLEX TTY SOFTWARE
        ST%PRV==1B4      ;PRIVILEGES
        ST%TWR==1B5      ;DUAL SEGMENT SOFTWARE
        ST%CYC==1B6      ;50 HERTZ CLOCK
        ST%TDS==7B9      ;TYPE OF DISK SYSTEM
        ST%IND==1B10     ;IND. PPNS ON DISK
        ST%IMG==1B11     ;IMAGE MODE TTYS
        ST%DUL==1B12     ;DUAL PROCESSOR SYSTEM
        ST%MRB==1B13     ;MULTIPLE RIBS SUPPORTED
        ST%HPT==1B14     ;HIGH PRECISION TIME ACCOUNTING
        ST%EMO==1B15     ;EXCLUDE OVERHEAD FROM TIME ACCOUNTING
        ST%RTC==1B16     ;REAL TIME CLOCK
        ST%MBF==1B17     ;MADE FOR FOROTS
        ST%NOP==1B27     ;NO OPERATOR IN ATTENDANCE
        ST%NSP==1B28     ;UNSPOOL DEVICES
        ST%ASS==1B29     ;ASSIGN/INIT DEVICES
        ST%NRT==1B32     ;NO REMOTE TTY'S
        ST%BON==1B33     ;BATCH ONLY
        ST%NRL==1B34     ;NO REMOTE LOGINS
        ST%NLG==1B35     ;NO LOGINS EXCEPT CTY/OPR
%CNSER==20,,11   ;APR SERIAL NUMBER
%CNNSM==21,,11   ;NANO-SECS PER MEMORY CYCLE
%CNPTY==22,,11   ;LH=NUMBER FIRST INV. TTY, RH=NUMBER PTYS
%CNFRE==23,,11   ;POINTER FOR BIT MAP OF CORE BLOCKS
%CNLOC==24,,11   ;LOCATION OF LOW CORE CORE BLOCKS
%CNSTB==25,,11   ;POINTER TO STATION BLOCK CHAIN
%CNOPL==26,,11   ;OPR LDB ADDRESS
%CNTTF==27,,11   ;POINTER TO TTY FREE CHUNKS
%CNTTC==30,,11   ;LH=NUMBER OF TTY CHUNKS, RH=ADDR OF FIRST
%CNTTN==31,,11   ;NUMBER OF FREE CHUNKS
%CNLNS==32,,11   ;POINTER TO CURRENT COMMAND TTY
%CNLNP==33,,11   ;POINTER TO TTY LINE TABLE
%CNVER==34,,11   ;MONITOR VERSION
%CNDSC==35,,11   ;POINTER TO DATA SET CONTROL TABLE
%CNDLS==36,,11   ;LAST RECIEVE INT. FROM DC10 (PRE 5.07)
%CNCCI==37,,11   ;LAST RECIEVE INT. FROM 680I (PRE 5.07)
%CNSGI==40,,11   ;LAST DORM. SEG THROWN AWAY
%CNPOK==41,,11   ;ADDRESS OF LAST POKED LOCATION
%CNPUC==42,,11   ;LH=JOB, RH=COUNT OF POKES
%CNWHY==43,,11   ;REASON FOR LAST RELOAD
%CNTIC==44,,11   ;NUMBER OF TICKS PER SECOND
%CNPDB==45,,11   ;POINTER TO PDB POINTER TABLES
%CNRTC==46,,11   ;RESOLUTION OF RUNTIME CLOCK (UNITS/SEC)
%CNCHN==47,,11   ;LH=PTR TO CHANNEL D.B. LIST,RH=UNUSED
```

```
%CNLMX==50,,11   ;LOGMAX (MAX JOBS TO BE LOGGED IN)
%CNBMX==51,,11   ;BATMAX (MAX BATCH JOBS)
%CNBMN==52,,11   ;BATMIN (MIN JOBS RESERVED FOR BATCH)
%CNDTM==53,,11   ;INTERNAL FORMAT DATE,,TIME
%CNLNM==54,,11   ;NUMBER OF JOBS LOGGED IN
%CNBNM==55,,11   ;NUMBER OF BATCH JOBS LOGGED IN
%CNYER==56,,11   ;LOCAL YEAR
%CNMON==57,,11   ;LOCAL MONTH
%CNDAY==60,,11   ;LOCAL DAY OF MONTH
%CNHOR==61,,11   ;LOCAL HOUR
%CNMIN==62,,11   ;LOCAL MINUTES
%CNSEC==63,,11   ;LOCAL SECONDS
%CNGMT==64,,11   ;TIME FROM GMT IN INTERNAL FORMAT
%CNDBG==65,,11   ;DEBUGGING STATUS WORD
         ST%DBG==1B0      ;SYSTEM DEBUGGING
         ST%RDC==1B1      ;RELOAD ON DEBUF STOPCD
         ST%RJE==1B2      ;RELOAD ON JOB STOPCD
         ST%NAR==1B3      ;NO AUTO RELOADS
         ST%CP1==1B4      ;IF SECOND CPU STOPS, STOP CPU0
%CNFRU==66,,11   ;MONITOR FREE CORE USED
%CNTCM==67,,11   ;MAX TTY CHUNKS
%CNCVN==70,,11   ;CUSTOMER VERSION (136)
%CNDVN==71,,11   ;DEC VERSION (137)
%CNDFC==72,,11   ;NUMBER OF DF10 DATA CHANS
%CNRTD==73,,11   ;NUMBER OF RT DEVICES
%CNHPQ==74,,11   ;NUMBER OF HPQ'S
%CNLDB==75,,11   ;TTY DDB WORD POINTING TO LDB
%CNMVO==76,,11   ;MAX VECTOR OFFSET FOR PISYS.
%CNMIP==77,,11   ;MAX PRIORITY FOR PISYS.
%CNMER==100,,11  ;ADDR OF MTA0,,OFFSET OF MTA ERR RPT WORD
%CNET1==101,,11  ;USER ADDRESS OF EXEC'S AC T1
%CNLSD==102,,11  ;LENGTH OF SHORT DDB
%CNLLD==103,,11  ;LENGTH OF LONG DDB
%CNLDD==104,,11  ;LENGTH OF DISK DDB
%CNEXM==105,,11  ;ADDRESS IN JOBDAT OF LAST E/D COMMAND
```

```
%CNST2==106,,11 ;MORE CONFIGURATION FEATURE INDICATORS
        ST%NDN==1B18     ;NETWORK DEVICE NAMES (GGGNNU)
        ST%XPI==1B19     ;EXCLUDE PI TIE FROM RUNTIME
        ST%ERT==1B20     ;EBOX/MBOX RUNTIME (KL10 ONLY)
        ST%EXE==1B21     ;SAVE AND SSAVE WRITE .EXE FILES
        ST%NJN==1B22     ;SYSTEM USES 9 BIT JOB NUMBERS
        ST%EER==1B23     ;EXTENDED ERROR REPORTING
        ST%TAP==1B24     ;TAPSER INCLUDED
        ST%MBE==1B25     ;MASS BUS ERROR REPORTS
        ST%GAL==1B26     ;GALAXY-10 SUPPORT INCLUDED
        ST%ENQ==1B27     ;ENQ./DEQ. IS INCLUDED
        ST%SHC==1B28     ;SCHEDULER HAS CLASSES
        ST%NSE==1B29     ;NON-SUPERSEDING ENTER
        ST%MSG==1B30     ;MSGSER INCLUDED
        ST%PSI==1B31     ;PSISER INCLUDED
        ST%IPC==1B32     ;IPCF INCLUDED
        ST%VMS==1B33     ;VMSER INCLUDED
        ST%MFR==1B34     ;MTA ERROR REPORTING
        ST%SSP==1B35     ;SWAP SPACE IN PAGES
%CNPIM==107,,11 ;MINIMUM CONDITION IN PISYS
%CNPIL==110,,11 ;LENGTH OF INTERNAL PIT'S
%CNPIA==111,,11 ;ADDRESS OF JBTPIA
%CNMNT==112,,11 ;MONITOR TYPE
        CN%MNX==1B0      ;STRANGE MONITOR
        CN%MNT==77B23    ;DEC-KNOWN TYPE
                         ;1=TOPS 2=ITS 3=TENEX
        CN%MNS==77B29    ;DEC SUB TYPE
        CN%MNC==77       ;CUSTOMER SUBSUB TYPE
```

```
        %CNOCR==113,,11 ;FIRST CDR DDB,,OFFSET TO CARD COUNT
        %CNOCP==114,,11 ;DITTO FOR CDP
        %CNPGS==115,,11 ;UNIT OF CORE ALLOCATION
        %CNMMX==116,,11 ;MINIMUM LEGAL CORMAX
        %CNNSC==117,,11 ;NUMBER OF SCHEDULER CLASSES
        %CNUTF==120,,11 ;EXPONENTIAL USER TIME FACTOR
        %CNHSO==121,,11 ;START OF MONITORS HISEG
        %CNHSL==122,,11 ;LENGTH OF MONITORS HISEG
        %CNNWC==123,,11 ;NUMBER OF WORDS OF CORE

.GTNSW==12      ;NON-SWAPPING DATA TABLE
        %NSCMX==10,,12  ;SYSTEM CORMAX (LARGEST USER JOB+1)
        %NSCLS==11,,12  ;BYTE POINTER TO LAST FREE CORE AREA
        %NSCTL==12,,12  ;VIRTUAL CORE TALLY
        %NSSHW==13,,12  ;JOB NUMBER STOPPED BY SHUFFLER
        %NSHLF==14,,12  ;ADDRESS OF LOWEST HOLE IN SYSTEM
        %NSUPT==15,,12  ;UPTIME (TICKS)
        %NSSHF==16,,12  ;WORDS SHUFFLED BY SYSTEM
        %NSSTU==17,,12  ;SYSTEM TAPE USER
        %NSHJB==20,,12  ;HIGHEST JOB NUMBER IN USE
        %NSCLW==21,,12  ;WORDS CLEARED BY SYSTEM
        %NSLST==22,,12  ;LOST TIME
        %NSMMS==23,,12  ;MEMORY SIZE
        %NSTPE==24,,12  ;TOTAL MEMEORY PARITY ERRORS
        %NSSPE==25,,12  ;SPURIOUS MEMORY PARITY ERRORS
        %NSMPC==26,,12  ;MULTIPLE MEMORY PARITY ERRORS
        %NSMPA==27,,12  ;LAST MEMORY PARITY ADDRESS
        %NSMPW==30,,12  ;LAST MEMORY PARITY WORD
        %NSMPP==31,,12  ;LAST MEMORY PARITY PC
        %NSEPO==32,,12  ;NUMBER OF EXEC PDL OVERFLOWS NOT RECOVERED
        %NSEPR==33,,12  ;NUMBER OF EXEC PDL OVERFLOWS RECOVERED
        %NSMXM==34,,12  ;MAX VALUE OF CORMAX
        %NSKTM==35,,12  ;KSYS TIMER
        %NSCMN==36,,12  ;CORMIN
        %NSABC==37,,12  ;COUNT OF ADDRESS BREAKS
        %NSABA==40,,12  ;ADDRESS OF ADDRESS BREAKS
        %NSLJR==41,,12  ;LAST JOB RUN
        %NSACR==42,,12  ;ACCUMULATED CPU RESPONSE
        %NSNCR==43,,12  ;NUMBER OF CPU RESPONSES
        %NSSCR==44,,12  ;ACCUMULATED SQUARE OF CPU RESPONSE
                ;*** NO MORE GROWTH--SEE .GTCOV
```

```
.GTSDT==13        ;SWAPPING DATA TABLE
        %SWBGH==0,,13    ;BIG HOLE
        %SWFIN==1,,13    ;FINISH
        %SWFRC==2,,13    ;FORCE
        %SWFIT==3,,13    ;FIT
        %SWVRT==4,,13    ;VIRTAL
        %SWERC==5,,13    ;SWAP ERROR COUNT
        %SWPIN==6,,13    ;-1 IF SWAP IN AND FTPDBS=1
.GTSGN==14        ;SEGMENT NUMBERS
        SN%SHR==1B1      ;SHARABLE SEGMENT
        SN%LOK==1B5      ;HIGH SEGMENT IS LOCKED
.GTODP==15        ;ONCE ONLY DISK PARAMETERS
        %ODSWP==0,,15    ;HIGHEST SWAPPING IN 4-SERIES
        %ODK4S==1,,15    ;K FOR SWAPPING
        %ODPRT==2,,15    ;IN CORE PROTECT TIME MULTIPLIER
        %ODPRA==3,,15    ;IN CORE PROTECT TIME OFFSET
.GTLVD==16        ;LEVEL-D PARAMETERS
        %LDMFD==0,,16    ;MFD PPN [1,1]
        %LDSYS==1,,16    ;SYS PPN [1,4]
        %LDFFA==2,,16    ;FULL FILE ACCESS PPN [1,2]
        %LDHLP==3,,16    ;UNLOGGED IN PPN [2,5]
        %LDQUE==4,,16    ;QUE AREA PPN [3,3]
        %LDSPB==5,,16    ;FIRST PPB,,NEXT PPB TO SCAN
        %LDSTR==6,,16    ;FIRST STR DATA BLOCK,,OFFSET TO NEXT
        %LDUNI==7,,16    ;FIRST UNIT DATA BLOCK,,OFFSET TO NEXT
        %LDSWP==10,,16   ;FIRST SWAP UNIT,,OFFSET TO NEXT
        %LDCRN==11,,16   ;NUMBER OF CORE BLOCKS
        %LDSTP==12,,16   ;STANDARD FILE PROTECTION
        %LDUFP==13,,16   ;STANDARD UFD PROTECTION
        %LDMBN==14,,16   ;NUMBER OF MONITOR BUFFERS
        %LDQUS==15,,16   ;QUE STRUCTURE NAME
        %LDCRP==16,,16   ;CRASH PPN [10,1]
        %LDSFD==17,,16   ;MAX DEPTH OF SFDS TO WRITE
        %LDSPP==20,,16   ;SPOOLED FILE PROTECTION
        %LDSYP==21,,16   ;STANDARD SYS: PROTECTION
        %LDSSP==22,,16   ;STANDARD SYS:.SYS PROTECTION
        %LDMNU==23,,16   ;MAX. NEGATIVE USETI WHICH READS EXTENDED RIBS
        %LDMXT==24,,16   ;MAX. BLOCKS TO TRANSFER
        %LDNEW==25,,16   ;EXPERIMENTAL SYS PPN [1,5]
        %LDOLD==26,,16   ;OLD SYS PPN [1,3]
        %LDUMD==27,,16   ;USER MODE DIAGNOSTICS PPN [6,6]
        %LDNDB==30,,16   ;DEFAULT DISK BUFFERS IN RING
        %LDMSL==31,,16   ;MAX UNITS IN A.S.L.
        %LDALG==32,,16   ;ALGOL LIBRARY PPN [5,4]
        %LDBLI==33,,16   ;BLISS LIBRARY PPN [5,5]
        %LDFOR==34,,16   ;FORTRAN LIBRARY PPN [5,6]
        %LDMAC==35,,16   ;MACRO LIBRARY PPN (SOURCE NOT UNIVERSALS) [5,7]
        %LDUNV==36,,16   ;UNIVERSAL LIBRARY PPN [5,17]
        %LDPUB==37,,16   ;PUBLIC USER SOFTWARE LIBRARY PPN [1,6]
```

```
        ;CONTINUE .GTLVD
%LDTED==40,,16   ;TEXT EDITOR LIBRARY PPN [5,10]
%LDREL==41,,16   ;REL FILE LIBRARY PPN [5,11]
%LDRNO==42,,16   ;RUNOFF LIBRARY PPN [5,12]
%LDSNO==43,,16   ;SNOBOL LIBRARY PPN [5,13]
%LDDOC==44,,16   ;DOC FILE LIBRARY PPN [5,14]
%LDFAI==45,,16   ;FAIL LIBRARY PPN [5,15]
%LDMUS==46,,16   ;MUSIC LIBRARY PPN [5,16]
%LDDEC==47,,16   ;STANDARD DEC SOFTWARE [10,7]
%LDSLP==50,,16   ;POINTER TO ACTIVE SWAP LIST
%LDBAS==51,,16   ;BASIC LIB PPN [5,1]
%LDCOB==52,,16   ;COBOL LIB PPN [5,2]
%LDMXI==53,,16   ;PDP-11 LIB PPN [5,3]
%LDNEL==54,,16   ;NELIAC LIB PPN [5,20]
%LDDMP==55,,16   ;DUMP PPN [5,21]
%LDPOP==56,,16   ;POP2 LIB PPN [5,22]
%LDTST==57,,16   ;TEST LIB PPN [5,23]
%LDLSO==60,,16   ;LOG SOFT OVERRUNS (CALL DAEMON) IF OVERRUN
                 ; IS RECOVERED ON 1 RETRY AND %LDLSO .NE. 0
%LDMBR==61,,16   ;MASS-BUSS REG. LH=OFFSET INTO KDB OF # OF
                 ; REGISTER, RH=OFFSET INTO UDB OF REGS.
%LDBBP==62,,16   ;LH=POINTER TO BYTE POINTER TO # LEFT IN BAT
                 ;RH=OFFSET (IN UDB) OF CHAN TERM FAIL CNT
```

```
.GTRCT==17          ;DISK BLOCKS READ
.GTWCT==20          ;DISK BLOCKS WRITTEN
.GTDBS==21          ;DISK BLOCK SECONDS
.GTTDB==22          ;TIME OF LAST ALLOCATE AND SIZE
.GTSLF==23          ;GETTAB IMMEDIATE (SELF)
.GTDEV==24          ;DEVICE OR STRUCTURE (SEGMENTS ONLY)
.GTWSN==25          ;NAMES OF WAIT STATES
.GTLOC==26          ;REMOTE STATION NUMBER
.GTCOR==27          ;CORE TABLE
.GTCOM==30          ;MONITOR COMMAND NAMES
.GTNM1==31          ;USER NAME
.GTNM2==32          ; ..
.GTCNO==33          ;CHARGE NUMBER
.GTIMP==34          ;IMPCOR POINTERS
.GTWCH==35          ;WATCH BITS
        JW.WDY==1B1       ;DAYTIME AT START
        JW.WRN==1B2       ;RUN TIME
        JW.WWI==1B3       ;WAIT TIME
        JW.WDR==1B4       ;DISK READS
        JW.WDW==1B5       ;DISK WRITES
        JW.WVR==1B6       ;VERSIONS
        JW.WMT==1B7       ;MTA STATISTICS
        JW.WAL==376B8     ;WATCH ALL
        JW.WMS==7B11      ;/MESSAGE LEVEL
                .JWWPR==1         ;PREFIX
                .JWWOL==2         ;ONE LINE
                .JWWPO==3         ;PREFIX,FIRST
                .JWWLG==6         ;LONG, NO PREFIX
                .JWWPL==7         ;PREFIX AND LONG
        JW.WCN==1B9       ;/MESSAGE:CONTINUATION
        JW.WFL==1B10      ;/MESSAGE:FIRST
        JW.WPR==1B11      ;/MESSAGE:PREFIX
```

```
.GTSPL==36          ;SPOOLING CONTROL
        JS.PRI==7B26      ;DISK PRIORITY
        JS.DFR==1B27      ;DEFERED SPOOLING (MPB-I STYLE)
        JS.PCR==1B31      ;SPOOL CDR
        JS.PCP==1B32      ;SPOOL CDP
        JS.PPT==1B33      ;SPOOL PTP
        JS.PPL==1B34      ;SPOOL PLT
        JS.PLP==1B35      ;SPOOL LPT
        JS.PAL==37        ;SPOOL ALL
.GTRTD==37          ;REAL TIME STATUS
.GTLTM==40          ;TIME AND BATCH STATUS
        JB.LCR==777B9     ;CORE LIMIT
        JB.LBT==1B10      ;BATCH JOB
        JB.LSY==1B11      ;GOTTEN FROM SYS:
        JB.LTM==77777777  ;TIME LIMIT TO GO IN JIFFIES
.GTQQQ==41          ;SCHEDULING QUEUE HEADERS
.GTQJB==42          ;JOB QUEUE LINK
.GTCM2==43          ;MONITOR SET COMMAND NAMES
.GTCRS==44          ;HARDWARE STATUS FROM CRASH
        CR.SAP==0,,44     ;APR CONI
        CR.SPI==1,,44     ;PI CONI
        CR.SSW==2,,44     ;APR DATAI (SWITCHES)
```

```
.GTISC==45          ;SWAP IN SCAN TABLES
.GTOSC==46          ;SWAP OUT SCAN
.GTSSC==47          ;SCHEDULER SCAN
.GTRSP==50          ;RESPONSE COUNTER TABLE
.GTSYS==51          ;SYSTEM WIDE DATA
        %SYERR==0,,51     ;SYSTEM WIDE HARDWARE ERROR COUNT
        %SYCCO==1,,51     ;NUMBER OF TIMES COMCNT WAS OFF
        %SYDEL==2,,51     ;DISABLED HARDWARE ERROR COUNT
        %SYSPC==3,,51     ;LH=3 LETTER CODE OF LAST STOPCD,RH=ADDRESS+1 OF LAST STOPCD
        %SYNDS==4,,51     ;NUMBER OF DEBUG STOPCDS
        %SYNJS==5,,51     ;NUMBER OF JOB STOPCDS (INCLUDING DEBUG
                          ; STOPCD'S IF A JOB IS STOPPED)
        %SYNCP==6,,51     ;NUMBER OF COMMANDS PROCESSED
        %SYSJN==7,,51     ;LAST STOPCD--JOB NUMBER
        %SYSTN==10,,51    ;LAST STOPCD--TTY NAME
        %SYSPN==11,,51    ;LAST STOPCD--PROGRAM NAME
        %SYSUU==12,,51    ;LAST STOPCD--UUO
        %SYSUP==13,,51    ;LAST STOPCD--USER PC
        %SYSPP==14,,51    ;LAST STOPCD--USER PPN
.GTWHY==52          ;OPERATOR WHY COMMENTS IN ASCIZ
.GTTRQ==53          ;TOTAL TIME IN RUN QUEUES WHETHER OR NOT RUNNING
.GTSPS==54          ;SECOND PROCESSOR STATUS
        SP.SC0==1B29      ;SET CPU COMMAND (OK TO USE)
        SP.CR0==1B35      ;SET CPU UUO (OK TO USE)
                    ;OTHERS BY SHIFTING LEFT 1 BIT/PROCESSOR
.GTC0C==55          ;CPU0 CDB CONSTANTS
.GTC0V==56          ;CPU0 CDB VARIABLES
.GTC1C==57          ;CPU1 CDB CONSTANTS
.GTC1V==60          ;CPU1 CDB VARIABLES
.GTC2C==61          ;CPU2 CDB CONSTANTS
.GTC2V==62          ;CPU2 CDB VARIABLES
.GTC3C==63          ;CPU3 CDB CONSTANTS
.GTC3V==64          ;CPU3 CDB VARIABLES
.GTC4C==65          ;CPU4 CDB CONSTANTS
.GTC4V==66          ;CPU4 CDB VARIABLES
.GTC5C==67          ;CPU5 CDB CONSTANTS
.GTC5V==70          ;CPU5 CDB VARIABLES
        %CCPTR==0,,55     ;LH=POINTER TO NEXT CDB
        %CCSER==1,,55     ;APR SERIAL NUMBER
        %CCOKP==2,,55     ;CPU OK IF LE 0, JIFFIES DEAD IF GT 0
        %CCTOS==3,,55     ;TRAP OFFSET FOR KA10 INTERRUPT LOCATIONS
        %CCLOG==4,,55     ;LOGICAL NAME (CPUN)
        %CCPHY==5,,55     ;PHYSICAL NAME (CPXN)
        %CCTYP==6,,55     ;TYPE OF PROCESSOR (LH=DEC, RH=CUST)
                .CC166==1         ;PDP-6
                .CCKAX==2         ;KA-10
                .CCKIX==3         ;KI-10
                .CCKLX==4         ;KL-10
        %CCMPT==7,,55     ;REL. GETTAB POINTER TO BAD ADDRESS TABLE
        %CCRTC==10,,55    ;REAL TIME CLOCK (DK10) DDB
```

```
%CCRTD==11,,55   ;REAL TIME CLOCK DDB IF HI PREC. TIME ACCT.
%CCPAR==12,,55   ;REL. GETTAB POINTER TO PARITY SUMMARY
%CCRSP==13,,55   ;REL. GETTAB POINTER TO RESPONSE SUMMARY
%CCDKX==14,,55   ;NUMBER OF DK10'S ON THIS CPU
%CCEBS==15,,55   ;NUMBER OF EBOX TICKS PER SECOND ON KL10
%CCMBS==16,,55   ;NUMBER OF MBOX TICKS PER SECOND ON KL10
%CVUPT==5,,56    ;UPTIME
%CVLST==12,,56   ;LOST TIME
%CVTPE==14,,56   ;TOTAL MEMORY PARITY ERRORS
%CVSPE==15,,56   ;SPURIOUS MEMORY PARITY ERRORS
%CVMPC==16,,56   ;MULTIPLE MEMORY PARITY ERRORS
%CVMPA==17,,56   ;MEMORY PARITY ADDRESS
%CVMPW==20,,56   ;MEMORY PARITY WORD
%CVMPP==21,,56   ;MEMORY PARITY PC
         ;HOLES ABOVE HERE BECAUSE OF .GTNSW COMPATABILITY
%CVABC==27,,56   ;ADDRESS BREAK COUNT
%CVABA==30,,56   ;ADDRESS BREAK ADDRESS
%CVLJR==31,,56   ;LAST JOB RUN
                 ;(OBSOLETE)
%CVSTS==35,,56   ;STOP TIME-SHARING THIS CPU
%CVRUN==36,,56   ;OPERATOR CONTROLLED SCHEDULING
        CV%RUN==1B0      ;DON'T RUN JOBS
%CVNUL==37,,56   ;NULL TIME
%CVEDI==40,,56   ;LH=PC,RH=COUNT OF EXEC DON'T CARE INTERRUPTS.
%CVJOB==41,,56   ;CURRENT JOB
%CVOHT==42,,56   ;OVERHEAD TIME IN JIFFIES (EXC. UUOS)
%CVEVM==43,,56   ;MAX EVM FOR LOCK UUO MAPPING
%CVEVU==44,,56   ;USED EVM FOR LOCK UUO MAPPING
%CVLLC==45,,56   ;LOCK LOOP COUNT
%CVTUC==46,,56   ;TOTAL UUO COUNT
%CVTJC==47,,56   ;TOTAL JOB CONTEXT SWITCH COUNT
%CVTNE==50,,56   ;TOTAL NXM ERRORS
%CVSNE==51,,56   ;TOTAL NON-REPRODUCIBLE NXM ERRORS
%CVNJA==52,,56   ;NUMBER OF JOBS AFFECTED BY THIS NXM
%CVMNA==53,,56   ;FIRST MEMORY ADDRESS WITH NXM
%CVETJ==54,,56   ;EBOX TICKS PER JIFFY (COMPUTED)
%CVMTJ==54,,56   ;MBOX TICKS PER JIFFY (COMPUTED BY ONCE)
```

```
%CVRSO==0          ;(REL.) SUM TTY OUT UUO RESPONSE
%CVRNO==1          ;(REL.) NUMBER TTY OUT UUO RESPONSE
%CVRHO==2          ;(REL.) HI-SUM SQ TTY OUT UUO RESPONSE
%CVRLO==3          ;(REL.) LO-SUM SQ TTY OUT UUO RESPONSE
%CVRSI==4          ;(REL.) SUM TTY INP UUO RESPONSE
%CVRNI==5          ;(REL.) NUMBER TTY INP UUO RESPONSE
%CVRHI==6          ;(REL.) HI-SUM SQ TTY INP UUO RESPONSE
%CVRLI==7          ;(REL.) LO-SUM SQ TTY INP UUO RESPONSE
%CVRSR==10         ;(REL.) SUM QUANTUM REQ RESPONSE
%CVRNR==11         ;(REL.) NUMBER QUANTUM REQ RESPONSE
%CVRHR==12         ;(REL.) HI-SUM SQ QUANTUM REQ RESPONSE
%CVRLR==13         ;(REL.) LO-SUM SQ QUANTUM REQ RESPONSE
%CVRSX==14         ;(REL.) SUM ONE OF ABOVE RESPONSE
%CVRNX==15         ;(REL.) NUMBER ONE OF ABOVE RESPONSE
%CVRHX==16         ;(REL.) HI-SUM SQ ONE OF ABOVE RESPONSE
%CVRLX==17         ;(REL.) LO-SUM SQ ONE OF ABOVE RESPONSE
%CVRSC==20         ;(REL.) SUM CPU RESPONSE
%CVRNC==21         ;(REL.) NUMBER CPU RESPONSE
%CVRHC==22         ;(REL.) HI-SUM SQ CPU RESPONSE
%CVRLC==23         ;(REL.) LO-SUM SQ CPU RESPONSE
%CVPLA==0          ;(REL.) HIGHEST ADDRESS OF PARITY ERROR
%CVPMR==1          ;(REL.) ADDRESS IN SEGMENT OF PARITY ERROR
%CVPTS==2          ;(REL.) NUMBER OF PARITIES THIS SWEEP
%CVPSC==3          ;(REL.) NUMBER OF PARITY SWEEPS
%CVPUE==4          ;(REL.) NUMBER OF USER ENABLED PARITY ERRORS
%CVPAA==5          ;(REL.) AND OF BAD ADDRESS THIS SWEEP
%CVPAC==6          ;(REL.) AND OF BAD CONTENTS THIS SWEEP
%CVPOA==7          ;(REL.) IOR OF BAD ADDRESS THIS SWEEP
%CVPOC==10         ;(REL.) IOR OF BAD CONTENTS THIS SWEEP
%CVPCS==11         ;(REL.) NUMBER OF SPURIOUS CHANNEL ERRORS
```

```
.GTFET==71          ;FEATURE TEST SETTINGS
        %FTUUO==0,,71    ;UUOS
                F%EQDQ==0,,1B22 ;ENQ./DEQ.
                F%GALA==0,,1B23 ;GALAXY-10 FEATURES
                F%PI==000,,1B24 ;SOFT. PI SYS
                F%IPCF==0,,1B25 ;IPCF
                F%CCIN==0,,1B26 ;CONTROL-C INTERCEPT
                F%PTYU==0,,1B27 ;JOBSTS AND CNTLJOB UUOS
                F%PEEK==0,,1B28 ;PEEK UUO
                F%POKE==0,,1B29 ;POKE. UUO
                F%JCON==0,,1B30 ;JOB CONTINUE
                F%SPL==00,,1B31 ;SPOOLING
                F%PRV==00,,1B32 ;JOB PRIVS
                F%DAEM==0,,1B33 ;DAEMON FUNCTIONS, ETC.
                F%GETT==0,,1B34 ;GETTAB UUO
                F%2REL==0,,1B35 ;2-REGISTER RELOCATION
        %FTRTS==1,,71    ;REAL TIME AND SCHEDULER
                F%NSCH==1,,1B25 ;NEW SCHEDULER
                F%VM==001,,1B26 ;VIRTUAL MEMORY
                F%SWAP==1,,1B27 ;SWAPPER (DEFINED IN S)
                F%SHFL==1,,1B28 ;SHUFFLER
                F%RTC==01,,1B29 ;DK10 SERVICE
                F%LOCK==1,,1B30 ;LOCK UUO
                F%TRPS==1,,1B31 ;TRPSET UUO
                F%RTTR==1,,1B32 ;RTTRAP UUO
                F%SLEE==1,,1B33 ;SLEEP UUO
                F%HIBW==1,,1B34 ;HIBER/WAKE UUOS
                F%HPQ==01,,1B35 ;HIGH PRIORITY RUN QUEUES
        %FTCOM==2,,71    ;COMMANDS
                F%EXE==02,,1B20 ;.EXE FORMAT FILES SUPPORTED
                F%MOFF==2,,1B21 ;SET MEMORY OFF LINE
                F%MONL==2,,1B22 ;SET MEMORY ON LINE
                F%CCL==02,,1B23 ;COMPIL COMMANDS (DEFINED IN S)
                F%CCLX==2,,1B24 ;COMPIL-CLASS
                F%QCOM==2,,1B25 ;QUEUE AND FRIENDS
                F%SET==02,,1B26 ;SET UUO/COMMAND
                F%VERS==2,,1B27 ;VERSION
                F%BCOM==2,,1B28 ;BATCH CONTROL FILE
                F%SEDA==2,,1B29 ;SET DAYTIME AND SET DATE
                F%WATC==2,,1B30 ;WATCH
                F%FINI==2,,1B31 ;FINISH AND CLOSE
                F%REAS==2,,1B32 ;REASSIGN UUO/COMMAND
                F%EXAM==2,,1B33 ;E AND D
                F%TALK==2,,1B34 ;SEND
                F%ATTA==2,,1B35 ;ATTACH COMMAND/UUO
        %FTACC==3,,71    ;ACCOUNTING INFO
                F%TLIM==3,,1B31 ;TIME/CORE LIMITS, ETC.
                F%CNO==03,,1B32 ;CHARGE NUMBER
                F%UNAM==3,,1B33 ;USER NAME
                F%KCT==03,,1B34 ;KILO-CORE-TICKS
                F%TIME==3,,1B35 ;RUN TIME
```

```
%FTERR==4,,71    ;ERROR CONTROL AND OPTIONS
        F%MNXM==4,,1B22 ;NXM ERROR RECOVERY CODE
        F%KL10==4,,1B23 ;THIS IS A KL10
        F%KA10==4,,1B24 ;THIS IS A KA10
        F%22BI==4,,1B25 ;22 BIT CHANNEL (DF10C)
        F%PDBS==4,,1B26 ;SWAPPING PDB
        F%KI10==4,,1B27 ;THIS IS A KI10
        F%METR==4,,1B28 ;METER. UUO
        F%EXON==4,,1B29 ;EXECUTE ONLY FILES (ALWAYS 1 SINCE 5.06)
        F%KII==04,,1B30 ;KI-10 INSTR CHECK ON KA10
        F%BOOT==4,,1B31 ;BOOTS BOOTSTRAP
        F%2SWP==4,,1B32 ;MULT. SWAPPING DEVICES
        F%EL==004,,1B33 ;DAEMON ERROR LOGGING
        F%MS==004,,1B34 ;MULTI-PROCESSORS
        F%MEMP==4,,1B35 ;MEMORY PARITY RECOVERY CODE
%FTDEB==5,,71    ;DEBUGGING FEATURES
        F%2SEG==5,,1B27 ;2 SEGMENT MONITOR
        F%RSP==05,,1B28 ;RESPONSE TIME
        F%WHY==05,,1B29 ;WHY RELOAD
        F%PATT==5,,1B30 ;PATCH SPACE IN TABLES
        F%TRAC==5,,1B31 ;BACK TRACKING FEATURES
        F%HALT==5,,1B32 ;HALTS IN MONITOR
        F%RCHK==5,,1B33 ;INTERNAL REDUNDANCY CHECKS
        F%MONP==5,,1B34 ;MONITOR WRITE PROTECTED
        F%CHEC==5,,1B35 ;MONITOR CHECKSUMMED
%FTSTR==6,,71    ;FILE STRUCTURE PARAMS
        F%DHIA==6,,1B19 ;HIGH AVAIL. FEATURES
        F%DSIM==6,,1B20 ;MULTI. ACCESS UPDATE
        F%NUL==06,,1B21 ;NUL
        F%LIB==06,,1B22 ;LIB/SYS/OLD/NEW ETC.
        F%DPRI==6,,1B23 ;DISK PRIORITY TRANSFERS
        F%APLB==6,,1B24 ;APPEND TO LAST BLOCK
        F%AIR==06,,1B25 ;APPEND IMPLIES READ
        F%GSRC==6,,1B26 ;GENERIC DEVICE SEARCH
        F%DRDR==6,,1B27 ;RENAME ACROSS DIRECTORIES
        F%DSEK==6,,1B28 ;SEEK UUO
        F%DSUP==6,,1B29 ;SUPER USETI/O
        F%DQTA==6,,1B30 ;DISK QUOTAS
        F%STR==06,,1B31 ;MULTIPLE STRUCTURES
        F%5UUO==6,,1B32 ;MISC. 5-SERIES UUOS
        F%PHYO==6,,1B33 ;PHYSICAL ONLY
        F%SFD==06,,1B34 ;SUB FILE DIRECTORIES
        F%MOUN==6,,1B35 ;STPUUO FUNCTIONS
```

```
%FTDSK==7,,71    ;INTERNAL DISK PARAMS
        F%RP04==7,,1B18 ;INCLUDE RP04 SUPPORT
        F%SLCK==7,,1B19 ;DEBUG SEARCH LIST CODE
        F%2ATB==7,,1B20 ;2 PART ACCESS BLOCKS
        F%CBDB==7,,1B21 ;DEBUG CB INTERLOCK
        F%LOGI==7,,1B22 ;LOGIN (DEFINED IN S)
        F%DISK==7,,1B23 ;DISK SYSTEM (DEFINED IN S)
        F%FFRE==7,,1B24 ;PREVENT RACES IN FILFND
        F%SWPE==7,,1B25 ;SWAP READ ERROR RECOVERY
        F%DBBK==7,,1B26 ;BAD BLOCK MARKING
        F%DUFC==7,,1B27 ;UFD COMPRESSOR
        F%DETS==7,,1B28 ;DISK ERROR SIMULATOR
        F%DMRB==7,,1B29 ;MULTI RIBS
        F%DSMC==7,,1B30 ;SMALLER ALLOC. OF DISK CORE BLOCKS
        F%DALC==7,,1B31 ;ALLOCATION OPTIMISATIONS
        F%DSTT==7,,1B32 ;DISK USAGE STATISTICS
        F%DHNG==7,,1B33 ;HUNG DISK RECOVERY
        F%DBAD==7,,1B34 ;DISK OFF-LINE RECOVERY
        F%DOPT==7,,1B35 ;LATENCY OPTIMIZATION
%FTSCN==10,,71   ;SCANNER OPTIONS
        F%DCXH==10,,1B22;DC10-H (2741 ON DC10) SUPPORTED
        F%TVP==010,,1B23;FANCY VERTICAL POSITIONING
        F%TYPE==10,,1B24;TYPESET-10 FEATURES IN DC76
        F%2741==10,,1B25;SUPPORT 2741-LIKE TERMINALS
        F%CAFE==10,,1B26;DC76
        F%TBLK==10,,1B27;TTY BLANK COMMAND
        F%TPAG==10,,1B28;PAGE AND DISPLAY KNOWLEDGE
        F%DIAL==10,,1B29;AUTO DIALER
        F%SCLC==10,,1B30;SPECIAL LINE CONTROL
        F%SCNR==10,,1B31;HARDWARE SCANER
        F%MODM==10,,1B32;MODEM CONTROL
        F%630H==10,,1B33;SINGLE SCANNER 630
        F%GPO2==10,,1B34;U.K. MODEM SUPPORT
        F%HDPX==10,,1B35;TRULY HALF DUPLEX TERMINALS
%FTPER==11,,71   ;I/O PARAMS
        F%RDBA==11,,1B19;READ BACKWARDS ON TU70
        F%TLAB==11,,1B20;TAPE LABEL SUPPORT
        F%TAPO==11,,1B21;TAPOP. UUO
        F%TASK==11,,1B22;TASK TO TASK NETWORK SUPPORT
        F%DAS7==11,,1B23;DAS78 (REMOTE 360/370/2780) SUPPORT
        F%XTC==011,,1B24;DA28-C NETWORK SUPPORT
        F%MSGS==11,,1B25;MSGSER (MPX DEVICE)
        F%HSLN==11,,1B26;HIGH-SPEED LOGICAL DEVICE SEARCH
        F%CPTR==11,,1B27;CDP TROUBLE INTERCEPT
        F%CRTR==11,,1B28;CDR TROUBLE INTERCEPT
        F%CTY1==11,,1B29;SUPPORT CTY1
        F%REM==011,,1B30;REMOTE STATION SOFTWARE
        F%LPTR==11,,1B31;LPT DEVICE ERROR RECOVERY
        F%OPRE==11,,1B32;DEVICE ERRS TO OPER
        F%CDRS==11,,1B33;CDR SUPER IMAGE MODE
        F%MTSE==11,,1B34;MTA DENSITY/BLOCK COMMANDS
        F%TMP==011,,1B35;TMPCOR AREA
```

```
.GTEDN==72        ;ERSATZ DEVICE NAMES
.GTSCN==73        ;SCANNER DATA
        %SCNRI==0,,73    ;NUMBER OF RCV INTERRUPTS
        %SCNXI==1,,73    ;NUMBER OF XMT INTERRUPTS
        %SCNEI==2,,73    ;NUMBER OF ECHO INTERRUPTS (IN XI)
        %SCNMB==3,,73    ;MAX BUFFER SIZE
        %SCNAL==4,,73    ;NUMBER OF ACTIVE LINES
        %SCNPS==5,,73    ;SIZE OF BUFFER FOR PIM MODE
        %SCNRA==6,,73    ;ADDRESS OF RECINT
        %SCNXA==7,,73    ;ADDRESS OF XMTINT
        %SCNTA==10,,73   ;ADDRESS OF TYPE
.GTSNA==74        ;LAST SEND ALL IN 9-BIT
        %SCNAE==0,,74    ;BYTE POINTER TO END BYTE IN MESSAGE
        %SCNAS==1,,74    ;BYTE POINTER TO FIRST-1 BYTE IN MESSAGE
        %SCNAM==2,,74    ;FIRST WORD OF DATA IN MESSAGE
.GTCMT==75        ;SET TTY COMMAND NAMES
.GTPID==76        ;PROCESS COMMUNICATION ID (IPCF)
.GTIPC==77        ;IPCF MISC. DATA
        %IPCML==0,,77    ;MAX. PACKET LENGTH
        %IPCSI==1,,77    ;PID OF SYSTEM-WIDE [SYSTEM]INFO
        %IPCDQ==2,,77    ;DEFAULT QUOTA
        %IPCTS==3,,77    ;TOTAL PACKETS SENT
        %IPCTO==4,,77    ;TOTAL PACKETS OUTSTANDING
        %IPCCP==5,,77    ;PID OF [SYSTEM]IPCC
        %IPCPM==6,,77    ;PID MASK
        %IPCMP==7,,77    ;LENGTH OF PID TABLE
        %IPCNP==10,,77   ;NUMBER OF PID'S NOW DEFINED
        %IPCTP==11,,77   ;TOTAL PID'S DEFINED SINCE RELOAD
.GTUPM==100       ;USER PAGE MAP PAGE
.GTCMW==101       ;SET WATCH COMMAND NAMES
.GTCVL==102       ;CURRENT VIRT LIMIT,,CURRENT PHY LIMIT
.GTMVL==103       ;MAXIMUM VIRT LIMIT,,MAXIMUM PHY LIMIT
.GTIPA==104       ;IPCF STATISTICS PER JOB
        IP.CQD==-1,,0    ;COUNT OF SENDS SINCE LOGIN
        IP.CQC==0,,-1    ;COUNT OF RECEIVES SINCE LOGIN
.GTIPP==105       ;IPCF POINTERS AND COUNTS
        IP.CQP==777B26   ;OUTSTANDING SENDS
        IP.CQQ==777      ;OUTSTANDING RECEIVES
.GTIPI==106       ;PID FOR THIS JOB'S [SYSTEM]INFO
.GTIPQ==107       ;IPCF FLAGS AND QUOTAS PER JOB
        IP.CQX==1B0      ;DISABLED
        IP.CQQ==1B1      ;QUOTA SET
        IP.CQS==777B26   ;SEND QUOTA
        IP.CQR==777      ;RECEIVE QUOTA
.GTDVL==110       ;POINTER TO THIS JOB'S LOGICAL NAME TABLE
.GTABS==111       ;ADDRESS BREAK WORD (DATAC PTR,)
.GTCMP==112       ;RESERVED FOR COMPATABILITY PACKAGES
        %CMPMT==0,,112   ;SIMULATED MONITOR TYPE (%CNMNT)
        %CMPCV==1,,112   ;CUSTOMER VERSION OF COMPAT. (136)
        %CMPDV==2,,112   ;DEC VERSION OF COMPAT. (137)
```

```
.GTVM==113        ;GENERAL VIRTUAL MEMORY DATA
        %VMSWP==0,,113   ;SWAP COUNT
        %VMSCN==1,,113   ;SCAN COUNT
        %VMSIP==2,,113   ;SWAPS IN PROGRESS
        %VMSLE==3,,113   ;SWAP LIST ENTRIES
        %VMTTL==4,,113   ;TOTAL VM IN USE
        %VMCMX==5,,113   ;MAX VALUE OF %VMTTL ALLOWED
        %VMRMX==6,,113   ;PAGING RATE MAX FOR SYSTEM
        %VMCON==7,,113   ;CONSTANT USED IN SWAP RATE COMPUTATION
        %VMQJB==10,,113 ;JOB TO REQUE TO PQV (-1 IF ALL)
        %VMRMJ==11,,113 ;PAGING RATE MAX PER JOB
        %VMTLF==12,,113 ;TIME OF LAST FAULT
        %VMSPF==13,,113 ;SYSTEM PAGE FAULT COUNTS: NOT IN WS,,IN WS
        %VMSW1==14,,113 ;ADDRESS OF SWPLST
        %VMSW2==15,,113 ;ADDRESS OF SW2LST
        %VMSW3==16,,113 ;ADDRESS OF SW3LST
.GTVRT==114       ;PER JOB PAGING RATE
.GTSST==115       ;SCHEDULER STATISTICS
        %SSOSO==0,,115   ;NUMBER OF JOBS RUN OUT OF ORDER TO ALLOW
                ; THEM TO GIVE UP RESOURCE FOR SWAP OUT.
        %SSOPJ==1,,115   ;NUMBER OF JOBS RUN OUT OF ORDER TO ALLOW
                ; THEM TO GIVE UP RESOURCE REQUIRED TO RUN A JOB
        %SSNUL==2,,115   ;SWAPPER NULL TIME
        %SSLOS==3,,115   ;SWAPPER LOST TIME
        %SSRQC==4,,115   ;TOTAL NUMBER OF REQUEUES
        %SSICM==5,,115   ;INTERVAL TO COMPUTE MCU
        %SSMSI==6,,115   ;MEDIUM TERM SCHEDULING INTERVAL
        %SSAJS==7,,115   ;AVERAGE JOB SIZE
        %SSTQT==10,,115 ;TOTAL QUOTA TIME
        %SSEAF==11,,115 ;EXPONENTIAL AVERAGING FACTOR
        %SSEAT==12,,115 ;EXPONENTIALY AVERAGED USER TIME
        %SSRSS==13,,115 ;TOTAL USER RUNTIME SINCE SCHED. UUO
                        ; SET CLASS PARAMETERS
.GTDCF==116       ;DESIRED CHAN. USE FRACTION (INDEX BY CHAN)
.GTST2==117       ;SECOND JOB STATUS WORD
.GTJTC==120       ;JOB TYPE AND SCHEDULER CLASS
.GTCQP==121       ;CLASS QUOTA IN PERCENT (INDEX BY CLASS)
.GTCQJ==122       ;CLASS QUOTA IN JIFFIES (INDEX BY CLASS)
.GTCRT==123       ;CLASS RUNTIME SINCE QUOTAS SET (INDEX BY CLASS)
.GTSQH==124       ;SUB QUEUE HEADERS
.GTSQ==125        ;SUB QUEUE WORD FOR EACH JOB
```

```
.GTSID==126      ;SPECIAL PID TABLE
        %SIIPC==0,,126  ;[SYSTEM]IPCC
        %SIINF==1,,126  ;[SYSTEM]INFO
        %SIQSR==2,,126  ;[SYSTEM]QUASAR
        %SIMDA==3,,126  ;MOUNTABLE DEVICE ALLOCATOR
        %SITLP==4,,126  ;MAGTAPE LABELING PROCESS
.GTENQ==127             ;ENQ./DEQ. STATISTICS
        %EQMSS==0,,127  ;MAXIMUM STRING SIZE
        %EQNAQ==1,,127  ;NUMBER OF ACTIVE QUEUES
        %EQESR==2,,127  ;TOTAL ENQ. SINCE RELOAD
        %EQDSR==3,,127  ;TOTAL DEQ. SINCE RELOAD
        %EQAPR==4,,127  ;ACTIVE POOLED RESOURCES
        %EQDEQ==5,,127  ;DEFAULT ENQ. QUOTA
.GTJLT==130             ;JOB LOGIN TIME IN UNIVERSAL FORMAT
.GTEBT==131             ;JIFFIES OF KL10 EBOX TIME
.GTEBR==132             ;JIFFY REMAINDER MOD RTUPS OF 131
.GTMBT==133             ;JIFFIES OF KL10 MBOX TIME
.GTMBR==134             ;JIFFY REMAINDER MOD RTUPS OF 133
```

```
        SUBTTL  MISC. NON-I/O -- TMPCOR

.TCRFS==0           ;COUNT OF FREE SPACE
.TCRRF==1           ;READ FILE
.TCRDF==2           ;DELETE FILE
.TCRWF==3           ;WRITE FILE
.TCRRD==4           ;READ DIRECTORY
.TCRDD==5           ;DELETE DIRECTORY

        SUBTTL  MISC. NON-I/O -- LOCK

LK.HNP==1B15        ;HI-SEG DON'T LOCK PHYSICALLY CONTIGUOUS
LK.HNE==1B16        ;HI-SEG DON'T MAP IN EXEC VM
LK.HLS==1B17        ;HI-SEG LOCK SEGMENT
LK.LNP==1B33        ;LO-SEG DON'T LOCK PHYSICALLY CONTIGUOUS
LK.LNE==1B34        ;LO-SEG DON'T MAP IN EXEC VM
LK.LLS==1B35        ;LO-SEG LOCK SEGMENT

.LKPPN==0           ;PHYSICAL PAGE NUMBER

;LOCK UUO ERRORS

LKNIS%==0           ;NOT IMPLEMENTED IN THIS SYSTEM
LKNLP%==1           ;NO LOCKING PRIVS
LKNCA%==2           ;NOT ENOUGH CORE TO CONTINUE CURRENT JOBS
LKNCM%==3           ;NOT ENOUGH CORE TO GUARANTEE CORMIN
LKNEM%==4           ;NOT ENOUGH EXEC VIRT MEM
LKNIA%==5           ;ILLEGAL SUB-FUNCTION ARGUMENT
LKNPU%==6           ;PAGE UNAVAILABLE

        SUBTTL  MISC. NON-I/O -- RTTRP
;RTTRP UUO ERROR CODES

RTJNP%==1B24        ;JOB DOESN'T HAVE PRIVS
RTNC0%==1B25        ;NOT RUNNABLE ON CPU0
RTDIU%==1B26        ;DEVICE IN USE BY ANOTHER JOB
RTIAU%==1B27        ;ILLEGAL AC USED DURING RTTRP AT INTERRUPT
RTJNL%==1B28        ;JOB NOT LOCKED (OR NOT PRIVILEGED)
RTSLE%==1B29        ;SYSTEM LIMIT EXCEEDED FOR RT DEVICES
RTILF%==1B30        ;ILLEGAL FORMAT OF I/O INSTRUCTION
RTPWI%==1B31        ;POINTER WORD ILLEGAL
RTEAB%==1B32        ;ERROR ADDRESS OUT OF BOUNDS
RTTAB%==1B33        ;TRAP ADDRESS BAD
RTPNB%==1B34        ;PI CHANNEL NOT CURRENTLY AVAILABLE FOR BLKI/O
RTPNA%==1B35        ;PI CHANNEL NOT AVAILABLE
```

```
        SUBTTL  MISC. NON-I/O -- JOBSTS

JB.UJA==1B0         ;JOB NUMBER ASSIGNED
JB.ULI==1B1         ;JOB IS LOGGED IN
JB.UML==1B2         ;TTY IS AT MONITOR LEVEL
JB.UOA==1B3         ;OUTPUT IS AVAILABLE
JB.UDI==1B4         ;TTY IS DEMANDING INPUT
JB.UJC==1B5         ;JACCT IS SET
JB.UJN==777777      ;JOB NUMBER

        SUBTTL  MISC. NON-I/O -- HIBER

HB.SWP==1B0         ;FORCE IMMEDIATE SWAP OUT
HB.IPC==1B10        ;IPCF
HB.RIO==1B11        ;I/O
HB.RPT==1B12        ;PTY ACTIVITY
HB.RTL==1B13        ;TTY LINE ACTIVITY
HB.RTC==1B14        ;TTY CHARACTER ACTIVITY
HB.RWJ==1B15        ;THIS JOB
HB.RWP==1B16        ;THIS PROGRAMMER
HB.RWT==1B17        ;THIS PROJECT


        SUBTTL  MISC. NON-I/O -- APRENB

AP.REN==1B18        ;REPETITIVE ENABLE
AP.POV==1B19        ;PUSH DOWN OVERFLOW
AP.ABK==1B21        ;(FUTURE)ADDRESS BREAK
AP.ILM==1B22        ;ILLEGAL MEMORY
AP.NXM==1B23        ;NON-EXISTENT MEMORY
AP.PAR==1B24        ;PARITY ERROR FLAG
AP.CLK==1B26        ;CLOCK
AP.FOV==1B29        ;FLOATING OVERFLOW
AP.AOV==1B32        ;ARITHMETIC OVERFLOW
```

```
        SUBTTL  MISC. NON-I/O -- SAVE/GET LOCATIONS

.SGNAM==0           ;FILE NAME FROM RUN UUO
.SGPPN==7           ;DIRECTORY FROM RUN UUO
.SGDEV==11          ;DEVICE FROM RUN UUO
.SGLOW==17          ;EXTENSION OF LOW SEG FROM RUN UUO
.SG41==122          ;LOCATION IN SAVE FILE CONTAINING COPY OF .JB41
.SGDDT==114         ;LOCATION IN SAVE FILE CONTAINING COPY OF .JBDDT

;BLOCK TYPES IN .EXE FILE DIRECTORY
.SVEND==1777        ;END OF DIRECTORY
.SVDIR==1776        ;DIRECTORY BLOCK

;.EXE FILE DIRECTORY ENTRIES
.SVFPF==0           ;FILE PAGE AND FLAGS
        SV%HIS==1B0       ;PAGE IS PART OF HISEG
        SV%SHR==1B1       ;PAGE IS SHARABLE
        SV%WRT==1B2       ;PAGE IS WRITABLE
        SV%CON==1B3       ;PAGE IS CONCEALED
        SV%SYM==1B4       ;PAGE IS PART OF SYMBOL TABLE
        SV%FPN==1777      ;FILE PAGE NUMBER
.SVPPC==1           ;PROCESS PAGE AND REPEAT COUNT
        SV%REP==777B8     ;REPEAT COUNT
        SV%PPN==777       ;PROCESS PAGE NUMBER
```

```
        SUBTTL  MISC. NON-I/O -- SETUUO

.STCMX==0           ;CORE MAX
.STCMN==1           ;CORE MIN
.STDAY==2           ;DAYTIME
.STSCH==3           ;SCHED WORD (SAME AS %CNSTS)
.STCDR==4           ;CDR SPOOL NAME
.STSPL==5           ;SPOOLING BITS (SAME AS .GTSPL)
.STWTC==6           ;WATCH BITS
        ST.WDY==1B19      ;WATCH DAYTIME AT START
        ST.WRN==1B20      ;WATCH RUN TIME
        ST.WWT==1B21      ;WATCH WAIT TIME
        ST.WDR==1B22      ;WATCH DISK READS
        ST.WDW==1B23      ;WATCH DISK WRITES
        ST.WVR==1B24      ;WATCH VERSIONS
        ST.WMT==1B25      ;WATCH MTA STATISTICS
        ST.WAL==376B26    ;WATCH ALL
.STDAT==7           ;DATE
.STOPR==10          ;OPR DEVICE
.STKSY==11          ;KSYS TIMER
.STCLM==12          ;CORE LIMIT
.STTLM==13          ;TIME LIMIT
.STCPU==14          ;CPU SPECIFICATION
.STCRN==15          ;CPU RUNABILITY
        SP.CR5==1B30      ;CPU5
        SP.CR4==1B31      ;CPU4
        SP.CR3==1B32      ;CPU3
        SP.CR2==1B33      ;CPU2
        SP.CR1==1B34      ;CPU1
        ;SP.CR0==1B35     ;CPU0  (SAME BIT DEFINED EARLIER)
.STLMX==16          ;LOGMAX
.STBMX==17          ;BATMAX
.STBMN==20          ;BATMIN
.STDFL==21          ;DSKFUL
        .DFPSE==0         ;PAUSE
        .DFERR==1         ;ERROR
.STMVM==22          ;MAX VM
.STMVR==23          ;MAX VM RATE
.STUVM==24          ;USER VM MAXIMA (VIRT,,PHY)
.STCVM==25          ;USER CURRENT VM MAXIMA (VIRT,,PHY)
        ST.VSG==1B18      ;SET IF PHYS LIMIT IS GUIDELINE
.STTVM==26          ;USER VIRT TIME INTERRUPTS
.STABK==27          ;ADDRESS BREAK (HDWR FORMAT; 1B3 BREAKS UUO REFERENCES)
.STPGM==30          ;SET PROGRAM TO RUN
.STDFR==31          ;SET DEFERRED SPOOLING
```

```
        SUBTTL  MISC. NON-I/O -- SCHED.

;;AC CONTAINS N,,ADDR WHERE ADDR CONTAINS:
;;   !=================================================!
;;   !        FUNCTION 1          !          BLOCK 1   !
;;   !-------------------------------------------------!
;;   !        FUNCTION 2          !          BLOCK 2   !
;;   !-------------------------------------------------!
;;   /                                                 /
;;   /                                                 /
;;   /                                                 /
;;   !-------------------------------------------------!
;;   !        FUNCTION N          !          BLOCK N   !
;;   !=================================================!

;FUNCTION CODES:
.SCRSI==000000  ;READ SCHEDULING INTERVAL
.SCSSI==400000  ;SET   SCHEDULING INTERVAL
        ;BLOCK CONTAINS:
        .SCBSI==0        ;SCHEDULING INTERVAL

.SCRMI==000001  ;READ MCU INTERVAL
.SCSMI==400001  ;SET  MCU INTERVAL
        ;BLOCK CONTAINS:
        .SCBMI==0        ;MCU INTERVAL

.SCRCQ==000002  ;READ CLASS QUOTAS AND FLAGS
.SCSCQ==400002  ;SET  CLASS QUOTAS AND FLAGS
        ;;BLOCK CONTAINS:
        ;;  !=================================!
        ;;  !            SIZE OF BLOCK        !
        ;;  !-----------------------------------!
        ;;  !    BITS+CLASS    !      QUOTA    !
        ;;  !-----------------------------------!
        ;;  /                                   /
        ;;  /                                   /
        ;;  /                                   /
        ;;  !-----------------------------------!
        ;;  !    BITS+CLASS    !      QUOTA    !
        ;;  !=================================!
        .SCBCT==0        ;WORD COUNT
        .SCBCQ==1        ;CLASS QUOTA
                ;******DEFINE FLAGS HERE******

;CONTINUED ON NEXT PAGE
```

```
.SCRTS==000003  ;READ TIME SLICE
.SCSTS==400003  ;SET  TIME SLICE
        ;;BLOCK CONTAINS:
        ;.SCBCT==0       ;WORD COUNT
        .SCBP1==1        ;TIME SLICE FOR PQ1
        .SCBP2==2        ;TIME SLICE FOR PQ2

.SCRUF==000004  ;READ DESIRED CHAN USE FRACTION
.SCSUF==400004  ;SET  DESIRED CHAN USE FRACTION
        ;;BLOCK CONTAINS:
        ;;  !=====================================!
        ;;  !              WORD COUNT             !
        ;;  !-------------------------------------!
        ;;  !      CHAN #       !       DCUF      !
        ;;  !-------------------------------------!
        ;;  /                                     /
        ;;  /                                     /
        ;;  /                                     /
        ;;  !-------------------------------------!
        ;;  !      CHAN #       !       DCUF      !
        ;;  !=====================================!
        ;.SCBCT==0       ;WORD COUNT
        .SCBUF==1        ;CHAN,,USE FRACTION IN %

.SCRJC==000005  ;READ JOB'S CLASS
.SCSJC==400005  ;SET  JOB'S CLASS
        ;;BLOCK CONTAINS:
        ;;  !=====================================!
        ;;  !              WORD COUNT             !
        ;;  !-------------------------------------!
        ;;  !      JOB #        !      CLASS      !
        ;;  !-------------------------------------!
        ;;  /                                     /
        ;;  /                                     /
        ;;  /                                     /
        ;;  !-------------------------------------!
        ;;  !      JOB #        !      CLASS      !
        ;;  !=====================================!
        ;.SCBCT==0       ;WORD COUNT
        .SCBJC==1        ;JOB,,CLASS
```

;CONTINUED ON NEXT PAGE

```
.SCRMC==000006  ;READ MCU CONSTANT
.SCSMC==400006  ;SET  MCU CONSTANT
        ;BLOCK CONTAINS
        .SCBMC==0        ;MCU CONSTANT

.SCRCU==000007  ;READ CLASS USAGE
        ;;BLOCK CONTAINS:
        ;;  !=====================================!
        ;;  !              WORD COUNT             !
        ;;  !-------------------------------------!
        ;;  !            CLASS 0 RUNTIME          !
        ;;  !-------------------------------------!
        ;;  !            CLASS 1 RUNTIME          !
        ;;  !-------------------------------------!
        ;;  /                                     /
        ;;  /                                     /
        ;;  /                                     /
        ;;  !-------------------------------------!
        ;;  !            CLASS N RUNTIME          !
        ;;  !=====================================!
        ;SCBCT==0        ;WORD COUNT
        .SCBCU==1        ;CLASS 0 USED

.SCREF==000010  ;READ EXPONENTIAL FACTOR
.SCSEF==400010  ;SET  EXPONENTIAL FACTOR
        ;BLOCK CONTAINS:
        .SCBEF==0        ;EXPONENTAIL FACTOR

.SCRMM==000011  ;READ MCU MULTIPLIER
.SCSMM==400011  ;SET  MCU MULTIPLIER
        ;BLOCK CONTAINS:
        .SCBMM==0        ;MCU MULTIPLIER

;SCHED. UUO ERROR CODES
SCHAC%==1        ;ADDRESS CHECK
SCHUF%==2        ;UNKNOWN FUNCTION
SCHUJ%==3        ;UNKNOWN JOB
SCHNP%==4        ;NOT PRIVILEGED
SCHUC%==5        ;UNKNOWN CLASS
SCHUQ%==6        ;UNKNOWN QUEUE
SCHNC%==7        ;NON-EXISTANT CHANNEL
SCHEB%==10       ;EXPONENTIAL FACTOR BAD
SCHMI%==11       ;ATTEMPT TO SET PROT WHEN MCUINT NON-ZERO
```

```
        SUBTTL  MISC, NON-I/O -- ATTACH

AT.UMM==180             ;PLACE IN MONITOR MODE
AT.UUM==181             ;PLACE IN USER MODE
```

```
        SUBTTL  UNIVERSAL DEVICE INDEX

.UXCHN==0           ;I/O CHANNEL NUMBER
                    ;001000-077777 ARE PHYSICAL DEVICES
        UX.TYP==77B26    ;DEVICE TYPE (SAME AS DEVTYP)
        UX.UNT==777      ;UNIT WITHIN TYPE
.UXTRM==200000      ;TERMINALS
.UXPRC==300000      ;PROCESS
```

```
        SUBTTL  .JBINT INTERCEPT BLOCK

;;  !=============================================================!
;;  !        BLOCK LENGTH          !            NEW PC            !
;;  !-------------------------------------------------------------!
;;  !                     OLD PC AND FLAGS                        !
;;  !-------------------------------------------------------------!
;;  !       CLASS OF INTERRUPT       !     CHANNEL NUMBER         !
;;  !=============================================================!

.ERNPC==0          ;LH=LENGTH, RH=NEW PC FOR INTERRUPT
.ERCLS==1          ;CLASSES OF ERROR INTERCEPTING
        ER.MSG==1B0       ;SUPPRESS ERROR MESSAGE
        ER.EIJ==1B29      ;ERROR IN JOB
        ER.TLX==1B30      ;TIME LIMIT EXCEEDED
        ER.QEX==1B31      ;QUOTA EXHAUSTED
        ER.FUL==1B32      ;FILE STRUCTURE FULL
        ER.OFL==1B33      ;DISK UNIT OFF-LINE
        ER.ICC==1B34      ;CONTROL-C INTERCEPT
        ER.IDV==1B35      ;"PROBLEM ON DEVICE" ERRORS
.EROPC==2          ;OLD PC
.ERCCL==3          ;RH=CHANNEL, LH=CLASS OF INTERRUPT
```

```
        SUBTTL  PSI SOFTWARE INTERRUPT SYSTEM

;INTERRUPT VECTOR
; SETS OF 4-WORD BLOCKS
;;   !=========================================================!
;;   !                    NEW PC AND FLAGS                     !
;;   !---------------------------------------------------------!
;;   !                    OLD PC AND FLAGS                     !
;;   !---------------------------------------------------------!
;;   ! !O!R!A!D!M!1!              !          I/O REASON        !
;;   !---------------------------------------------------------!
;;   !                    INTERRUPT STATUS                     !
;;   !=========================================================!

.PSVNP==0         ;NEW PC AND FLAGS
.PSVOP==1         ;OLD PC AND FLAGS
.PSVFL==2         ;FLAGS
        PS.VPO==1B1       ;TURN PERMANENTLY OFF, NO RESTORE
        PS.VTO==1B2       ;TURN OFF, RESTORE ON DEBRK.
        PS.VAI==1B3       ;ALLOW ADDITIONAL INTERRUPT
        PS.VDS==1B4       ;DISCARD SUCCESSIVE INTERRUPTS WHILE INTERRUPTED
        PS.VPM==1B5       ;PRINT STANDARD MESSAGE
        PS.VIP==1B6       ;INTERRUPTS IN PROGRESS FOR THIS BLOCK
                          ;  (USED BY PSISER)

        PS.RID==1B19      ;REASON--INPUT DONE
        PS.ROD==1B20      ;REASON--OUTPUT DONE
        PS.REF==1B21      ;REASON--END FILE
        PS.RIE==1B22      ;REASON--INPUT ERROR
        PS.ROE==1B23      ;REASON--OUTPUT ERROR
        PS.RDO==1B24      ;REASON--DEVICE OFF-LINE
        PS.RDF==1B25      ;REASON--DEVICE FULL
        PS.RQE==1B26      ;REASON--QUOTA EXCEEDED
        PS.RWT==1B27      ;REASON--IO WAIT
.PSVIS==3         ;INTERRUPT STATUS (AUX. WORD)
                  ;  I/O DEVICES RETURN   UDX,,GETSTS
```

```
;NON-DEVICE CONDITIONS

.PCTLE==-1          ;TIME LIMIT EXCEEDED (NON-BATCH ONLY)
.PCABT==-2          ;ABORT (^A); RETURNS 1B0=1 IF TI WAIT; (FUTURE)
.PCSTP==-3          ;STOP (^C); RETURNS 1B0=1 IF TI WAIT
.PCUUO==-4          ;ANY MUUO; RETURNS UUO
.PCIUU==-5          ;ILLEGAL UUO; RETURNS UUO
.PCIMR==-6          ;ILLEGAL MEMORY REFERENCE
.PCACK==-7          ;ADDRESS CHECK; RETURNS DEVICE NAME
.PCARI==-10         ;ARITHMETIC EXCEPTION
.PCPDL==-11         ;PDL OVERFLOW
.PCTT3==-12         ;TRAP TYPE 3 (FUTURE)
.PCNXM==-13         ;NON-EXISTENT MEMORY
.PCAPC==-14         ;APR CLOCK; RETURNS MSTIME
.PCUEJ==-15         ;USER INDUCED ERROR IN JOB
.PCXEJ==-16         ;EXTERNAL ERROR IN JOB
.PCKSY==-17         ;KSYS WARNING; RETURNS MINS TO KSYS
.PCDSC==-20         ;DATA-SET CHANGE; RETURNS NEW STATUS
.PCDAT==-21         ;DETACH/ATTACH; RETURNS -1 OR TTY UDX
.PCWAK==-22         ;WAKE UUO; RETURNS JOB NUMBER OF WAKER
.PCABK==-23         ;ADDRESS BREAK
.PCIPC==-24         ;IPCF RECEIVE; RETURNS LENGTH,,FLAGS
.PCRMC==-25         ;REMOTE COMPUTER CONDITION
.PCQUE==-26         ;ENQ/DEQ RESOURCE AVAILABLE
```

```
;INTERRUPT ENABLE REQUEST BLOCK
; SETS OF 3-WORD BLOCKS

.PSECN==0          ;CONDITION OR DEVICE
.PSEOR==1          ;OFFSET,,REASON BITS
.PSEPR==2          ;PRIORITY,,RESERVED

;PISYS. FUNCTION BITS

PS.FOF==1B1        ;TURN OFF
PS.FON==1B2        ;TURN ON
PS.FCP==1B3        ;CLEAR ALL PENDING INTERRUPTS
PS.FCS==1B4        ;CLEAR SELECTED INTERRUPT
PS.FRC==1B5        ;REMOVE CONDITION OR DEVICE
PS.FAC==1B6        ;ADD CONDITION OR DEVICE

;PISAV./PIRST. FLAGS

.PSSFC==0          ;FLAGS,,COUNT
        PS.SON==1B0      ;SYSTEM IS ON
.PSSIV==1          ;ADDRESS OF INTERRUPT VECTOR
.PSSBL==2          ;START OF 3-WORD BLOCKS

;PISYS. ERRORS

PSTMA%==0          ;TOO MANY ARGUMENTS
PSNFS%==1          ;NO FUNCTION SUPPLIED
PSUKF%==2          ;UNKNOWN FUNCTION REQUESTED
PSOOF%==3          ;ON AND OFF IN SAME FUNCTION
PSUKC%==4          ;UNKNOWN CONDITION REQUESTED
PSDNO%==5          ;DEVICE NOT OPEN
PSPRV%==6          ;PRIVILEGE FAILURE
PSIVO%==7          ;INVALID VECTOR OFFSET
PSUKR%==10         ;UNKNOWN REASON ENABLED
PSPIL%==11         ;PRIORITY TOO LARGE
PSNRW%==12         ;NON-ZERO RESERVED WORD
PSPND%==13         ;PIINI. NOT DONE
PSARF%==14         ;ADD AND REMOVE IN SAME FUNCTION

;PISAV. ERRORS

PSBTS%==0          ;BLOCK TOO SMALL

;PIRST. ERRORS

PSNRS%==0          ;NOT RESTORING WHAT WAS SAVED
```

```
        SUBTTL  IPCF INTERPROCESS COMMUNICATION FACILITY

;PACKET FORMAT
;;  !=================================================================!
;;  !BB!IS!R!O!T!                          !P!V!      !   ERROR   !SENDR!RETRN!
;;  !-----------------------------------------------------------------!
;;  !                            SENDER'S PID                         !
;;  !-----------------------------------------------------------------!
;;  !                           RECEIVER'S PID                        !
;;  !-----------------------------------------------------------------!
;;  !        WORD LENGTH OF DATA         !     START OF DATA (WORD/PAGE)  !
;;  !-----------------------------------------------------------------!
;;  !                   SENDER'S PPN (SUPPLIED BY MONITOR)            !
;;  !-----------------------------------------------------------------!
;;  !J !L !X!P!I!                                                     !
;;  !=================================================================!

.IPCFL==0       ;FLAGS
        IP.CFB==1B0     ;DON'T BLOCK READ
        IP.CFS==1B1     ;INDIRECT SENDER'S PID
        IP.CFR==1B2     ;INDIRECT RECEIVER'S PID
        IP.CFO==1B3     ;OVERDRAW SEND
        IP.CFT==1B4     ;TRUNCATE READ
        IP.CFP==1B18    ;SENDER IS PRIVILEGED AND IS INVOKING THEM
        IP.CFV==1B19    ;VM PAGE TRANSFER MODE
        IP.CFE==77B29   ;ERROR FIELD (NOT PRIV.)
        IP.CFC==7B32    ;SYSTEM SENDER CODE (PRIV.)
                .IPCCC==1       ;SENT BY [SYSTEM]IPCC
                .IPCCF==2       ;SENT BY SYSTEM-WIDE [SYSTEM]INFO
                .IPCCP==3       ;SENT BY RECEIVER'S [SYSTEM]INFO
        IP.CFM==7       ;SPECIAL MESSAGE RETURN FIELD (PRIV.)
                .IPCFN==1       ;MESSAGE WAS NOT DELIVERED
.IPCFS==1       ;SENDER'S PID
.IPCFR==2       ;RECEIVER'S PID
.IPCFP==3       ;LENGTH,,START OF DATA IN PACKET
.IPCFU==4       ;SENDER'S PPN (SUPPLIED BY MONITOR)
.IPCFC==5       ;SENDER'S CAPABILITIES WORD. (SUPPLIED BY MONITOR)
        IP.JAC==1B0     ;SENDER HAS JACCT SET
        IP.JLG==1B1     ;SENDER IS  LOGGED-IN
        IP.SXO==1B2     ;SENDER IS  EXECUTE ONLY
        IP.POK==1B3     ;SENDER HAS JS.POK PRIV
        IP.IPC==1B4     ;SENDER HAS IPCF PRIVS
```

```
;IPCC AND INFO ERROR CODES

IPCAC%==1          ;ADDRESS CHECK
IPCNL%==2          ;NOT LONG ENOUGH
IPCNP%==3          ;NO PACKET IN RECEIVE QUEUE
IPCIU%==4          ;(UNUSED)
IPCTL%==5          ;DATA TOO LONG FOR USER'S BUFFER
IPCDU%==6          ;DESTINATION UNKNOWN (RECEIVER'S PID)
IPCDD%==7          ;DESTINATION DISABLED
IPCRS%==10         ;NO ROOM IN SENDER'S QUOTA
IPCRR%==11         ;NO ROOM IN RECEIVER'S QUOTA
IPCRY%==12         ;NO ROOM IN SYSTEM STORAGE
IPCUP%==13         ;UNKNOWN PAGE ON SEND; DUPLICATE PAGE ON RECEIVE (VM)
IPCIS%==14         ;INVALID SEND PID
IPCPI%==15         ;PRIV INSUFFICIENT
IPCUF%==16         ;UNKNOWN FUNCTION
IPCBJ%==17         ;BAD JOB NUMBER
IPCPF%==20         ;PID TABLE FULL
IPCPR%==21         ;PAGE REQUESTED, NORMAL NEXT
IPCIE%==22         ;PAGING I/O ERROR
IPCBI%==23         ;BAD INDEX SPECIFIED FOR SYSTEM PID TABLE
IPCUI%==24         ;UNDEFINED ID IN SYSTEM PID TABLE
IPCFU%==70         ;[SYSTEM]INFO HAS AN UNKNOWN, INTERNAL ERROR
IPCCF%==71         ;[SYSTEM]IPCC REQUEST FROM [SYSTEM]INFO FAILED
IPCFF%==72         ;[SYSTEM]INFO FAILED TO COMPLETE AN ASSIGN
IPCQP%==73         ;PID QUOTA EXCEEDED
IPCBP%==74         ;BAD (UNKNOWN) PID
IPCDN%==75         ;DUPLICATE NAME
IPCNN%==76         ;NO SUCH NAME
IPCBN%==77         ;NAME HAS ILLEGAL CHARACTERS
```

```
;MESSAGES TO AND FROM [SYSTEM]IPCC

.IPCS0==0        ;LH=CALLER'S IDENTIFIER, RH=FUNCTION
        .IPCSE==1        ;ENABLE (ME OR (1)=PID)
        .IPCSD==2        ;DISABLE (ME OR (1)=PID)
        .IPCSI==3        ;TELL PID OF [SYSTEM]INFO FOR (ME OR (1)=PID); (2) GETS PID
        .IPCSF==4        ;MAKE [SYSTEM]INFO OF (1) (2)=FOR WHOM (0=SYSTEM)
        .IPCSZ==5        ;ZAP PID IN (1)
        .IPCSC==6        ;CREATE PID FOR JOB IN (1); (2) GETS PID
        .IPCSQ==7        ;SET QUOTA (2) FOR (1)=PID
        .IPCSO==10       ;CHANGE OWNER OF (1)=PID, (2)=NEW JOB NUMBER
        .IPCSJ==11       ;GIVE JOB OF PID IN (1); (2) GETS JOB NO
        .IPCSP==12       ;GIVE PID LIST FOR JOB (1) STARTING AT (2)
        .IPCSR==13       ;READ QUOTA OF JOB (1); INTO (2)
        .IPCSW==14       ;WAKE JOB (1) SLEEPING FROM .IPCSS
        .IPCSS==15       ;(ANSWER ONLY) IF LH(1)=0, JOB RH(1) IS RESETTING
                         ;              IF LH(1)=-1, JOB RH(1) IS LOGGING OUT
        .IPCWP==24       ;WRITE SYSTEM PID TABLE
        .IPCRP==25       ;READ SYSTEM PID TABLE
        .IPCSU==26       ;SPOOLED FILE CLOSED (SENT TO [SYSTEM]QUASAR)
        .IPCSL==27       ;LOGOUT MESSAGE SEND TO [SYSTEM]QUASAR
.IPCS1==1        ;FIRST ARGUMENT
.IPCS2==2        ;SECOND ARGUMENT
.IPCS3==3        ;THIRD ARGUMENT

;SPECIAL SYSTEM PID TYPES (READ/WRITE .GTSID VIA .IPCRP AND .IPCWP)
.IPCPS==0        ;[SYSTEM]IPCC
.IPCPI==1        ;[SYSTEM]INFO
.IPCPQ==2        ;[SYSTEM]QUASAR
.IPCPM==3        ;MOUNTABLE DEVICE ALLOCATOR
.IPCPT==4        ;TAPE LABEL PROCESS

;MESSAGES TO AND FROM [SYSTEM]INFO

.IPCI0==0        ;LH=CALLER'S IDENTIFIER, RH=FUNCTION
        .IPCIW==1        ;WHAT IS PID, ASCIZ IN (2+)
        .IPCIG==2        ;GET NAME OF (2)=PID
        .IPCII==3        ;ASSIGN NAME UNTIL RESET (FORMAT=.IPCIW)
        .IPCIJ==4        ;ASSIGN NAME UNTIL LOGOUT (FORMAT=.IPCIW)
        .IPCID==5        ;DROP SPECIFIC PID (2)
        .IPCIR==6        ;DROP NAMES SET BY .IPCII, (2)=JOB NUMBER
        .IPCIL==7        ;DROP NAMES SET BY .IPCIJ, (2)=JOB NUMBER
        .IPCIS==15       ;RESET JOB RH(1) IF LH(1)=0, OR LOGOUT IF LH=-1
.IPCI1==1        ;FIRST ARGUMENT (ALWAYS 0 OR PID TO GET DUPLICATE OF ANSWER)
.IPCI2==2        ;SECOND ARGUMENT
```

```
        SUBTTL  PAGE AND VM  VIRTUAL MEMORY FACILITY

;PAGE. UUO FUNCTIONS

.PAGIO==0           ;PAGE IN/OUT (OUT IF 1B0=1 IN LIST)
        PA.GSL==1B1       ;PAGE TO SLOW SWAPPING SPACE
.PAGCD==1           ;PAGE CREATE/DESTROY
        PA.GCD==1B1       ;CREATE PAGE ON DISK
.PAGEM==2           ;PAGE EXCHANGE/MOVE
.PAGAA==3           ;CLEAR/SET ACCESS ALLOWED
.PAGWS==4           ;GET WORKING SET
.PAGGA==5           ;GET ACCESS ALLOWED
.PAGCA==6           ;CHECK ACCESS LEGAL
        PA.GNE==1B0       ;DOES NOT EXIST
        PA.GWP==1B1       ;WRITABLE
        PA.GRD==1B2       ;READABLE
        PA.GAA==1B3       ;ACCESS ALLOWED
        PA.GAZ==1B4       ;ALLOCATED BUT ZERO
        PA.GCP==1B5       ;CAN'T BE PAGED OUT
        PA.GPO==1B6       ;IS PAGED OUT
.PAGCH==7           ;CREATE A HISEG (GENERAL REMAP)

;PAGE. UUO ERRORS

PAGUF%==0           ;UNIMPLEMENTED FUNCTION
PAGIA%==1           ;ILLEGAL ARGUMENT
PAGIP%==2           ;ILLEGAL PAGE NUMBER
PAGCE%==3           ;PAGE CAN'T EXIST BUT DOES
PAGME%==4           ;PAGE MUST EXIST BUT DOESN'T
PAGMI%==5           ;PAGE MUST BE IN CORE BUT ISN'T
PAGCI%==6           ;PAGE CAN'T BE IN CORE BUT IS
PAGSH%==7           ;PAGE IS IN A SHARABLE HI-SEG
PAGIO%==10          ;PAGING I/O ERROR
PAGNS%==11          ;NO SWAPPING SPACE AVAILABLE
PAGLE%==12          ;CORE LIMIT EXCEEDED
PAGIL%==13          ;ILLEGAL IF LOCKED
PAGNX%==14          ;CAN NOT CREATE ALLOCTED BUT ZERO PAGE
                    ; WITH VIRTUAL LIMIT EQUAL TO ZERO.
```

```
;.JBPFH REGION

.PFHNP==0          ;NEW PC AND FLAGS
.PFHOP==1          ;OLD PC AND FLAGS
.PFHFC==2          ;FAULT WORD
        PF.HCB==1B0       ;WORKING SET CHANGED BEHIND BACK
        PF.HPN==777B17    ;PAGE NUMBER
        PF.HFC==0,,-1     ;FAULT CODE
                .PFHNA==1         ;PAGE NOT ACCESSABLE
                .PFHNI==2         ;PAGE NOT IN CORE
                .PFHUU==3         ;PAGE FAULT IN UUO ARGS
                .PFHTI==4         ;VIRTUAL TIMER
                .PFHZI==5         ;ALLOCATED BUT ZERO FROM USER
                .PFHZU==6         ;ALLOCATED BUT ZERO DURING UUO
.PFHVT==3          ;VIRTUAL TIME
.PFHPR==4          ;PAGING RATE
        ;5-10 RESERVED
```

```
        SUBTTL  DAEMON CALLS

;DAEMON UUO FUNCTIONS

.DCORE==1           ;DUMP CORE
.CLOCK==2           ;ENTER A CLOCK REQUEST
.FACT==3            ;MAKE A FACT FILE ENTRY
.DMQUE==4           ;(UNIMPLEMENTED)
.DMERR==5           ;ERROR LOGGING
.DMCTL==6           ;(UNIMPLEMENTED)

;DAEMON UUO ERRORS

DMILF%==1           ;ILLEGAL FUNCTION
DMACK%==2           ;ADDRESS CHECK
DMWNA%==3           ;WRONG NUMBER OF ARGUMENTS
DMSNH%==4           ;IMPOSSIBLE UUO FAILURE (SHOULD NEVER HAPPEN)
DMCWF%==5           ;CAN'T WRITE FILE
DMNPV%==6           ;NO PRIVILEGES
DMFFB%==7           ;FACT FORMAT BAD
DMPTH%==10          ;INVALID PATH SPECIFICATION

;DCORE DUMP CATEGORIES

.CAJOB==1           ;JOB TABLES (SEE BELOW)
.CACNF==2           ;CONFIGURATION TABLES (.GTCNV)
.CADDB==3           ;JOB'S DDBS
.CACOR==4           ;USER'S CORE IMAGE (COMPRESSED)
.CAFET==5           ;FET GETTAB
.CAMAX==5           ;HIGHEST LEGAL CATEGORY NUMBER
```

```
;DCORE JOB TABLE ENTRIES

.DJVER==0          ;DAEMON VERSION (137)
.DJDAT==1          ;DATE (FROM DATE UUO)
.DJMST==2          ;TIME IN MILLISEC. (FROM MSTIME UUO)
.DJJSN==3          ;JOB,,SEGMENT NUMBERS
.DJLIN==4          ;???,,LINE NUMBER (TTY)
.DJSTS==5          ;.GTSTS(JOB)
.DJHTS==6          ;.GTSTS(HISEG)
.DJPPN==7          ;.GTPPN(JOB)
.DJHPN==10         ;.GTPPN(HISEG)
.DJPRG==11         ;.GTPRG(JOB)
.DJHPG==12         ;.GTPRG(HISEG)
.DJTIM==13         ;.GTTIM(JOB)
.DJKCT==14         ;.GTKCT(JOB)
.DJPRV==15         ;.GTPRV(JOB)
.DJSWP==16         ;.GTSWP(JOB)
.DJHWP==17         ;.GTSWP(HISEG)
.DJRCT==20         ;.GTRCT(JOB)
.DJWCT==21         ;.GTWCT(JOB)
.DJTDB==22         ;.GTTDB(JOB)
.DJDEV==23         ;.GTDEV(HISEG)
.DJNM1==24         ;.GTNM1(JOB)
.DJNM2==25         ;.GTNM2(JOB)
.DJCNO==26         ;.GTCNO(JOB)
.DJTMP==27         ;.GTTMP(JOB)
.DJWCH==30         ;.GTWCH(JOB)
.DJSPL==31         ;.GTSPL(JOB)
.DJRTD==32         ;.GTRTD(JOB)
.DJLIM==33         ;.GTLIM(JOB)
.DJSPS==34         ;.GTSPS(JOB)
.DJRSP==35         ;.GTRSP(JOB)
.DJTRQ==36         ;.GTTRQ(JOB)
.DJUPM==37         ;.GTUPM(JOB)
.DJHPM==40         ;.GTUPM(HISEG)
.DJCVL==41         ;.GTCVL(JOB)
.DJMVL==42         ;.GTMVL(JOB)
.DJIPA==43         ;.GTIPA(JOB)
.DJIPC==44         ;.GTIPC(JOB)
.DJIPI==45         ;.GTIPI(JOB)
.DJIPQ==46         ;.GTIPQ(JOB)
.DJDVL==47         ;.GTDVL(JOB)
.DJABS==50         ;.GTABS(JOB)
.DJVRT==51         ;.GTVRT(JOB)
.DJHRT==52         ;.GTVRT(HISEG)
.DJMAX==52         ;HIGHEST LEGAL JOB TABLE
```

```
        SUBTTL  METER UUO

;METER. FUNCTIONS

.MEFCI==0           ;INITIALIZE METER CHANNEL
.MEFCS==1           ;READ METER CHANNEL STATUS
.MEFCR==2           ;RELEASE METER CHANNEL
.MEFPI==3           ;INITIALIZE METER POINT
.MEFPS==4           ;READ METER POINT STATUS
.MEFPR==5           ;RELEASE METER POINT

;METER. ERRORS

MEILF%==1           ;ILLEGAL FUNCTION
MENPV%==2           ;NOT PRIVILEGED USER
MEIMA%==3           ;ILLEGAL MEMORY ADDRESS
MEPDL%==4           ;PDL OVERFLOW
MEIAL%==5           ;ILLEGAL ARG LIST
MEIAV%==6           ;ILLEGAL ARG VALUE
MENFC%==7           ;NOT ENOUGH FREE CORE
MEICT%==10          ;ILLEGAL CHANNEL TYPE
MEIPT%==11          ;ILLEGAL POINT ROUTINE TYPE
MENXP%==12          ;NON-EXISTENT POINT NAME
MENXC%==13          ;NON-EXISTENT CHANNEL
MEPNA%==14          ;POINT NOT AVAILABLE

;STANDARD CHANNEL ARGUMENT BLOCK LOCATIONS

.MCFUN==0           ;UUO FUNCTION CODE
.MCCID==1           ;USER CHANNEL ID
.MCTYP==2           ;CHANNEL TYPE
                    .MCTYN==0         ;NULL CHANNEL
                    .MCTYD==1         ;DISPLAY CHANNEL
                    .MCTYT==2         ;TRACE CHANNEL
.MCSTS==3           ;CHANNEL STATUS
        MC.STS==17777B12 ;STATUS MASK
        MC.USA==1B1      ;USER SEGMENT ADDRESSED
.MCJOB==4           ;CHANNEL JOB NUMBER

;DISPLAY CHANNEL ARGS

.MCTCN==5           ;AVERAGING TIME CONSTANT
.MCPTR==6           ;DEPOSIT BYTE POINTER
```

```
;TRACE CHANNEL ARGS

.MCFLG==5          ;USER ADDRESS OF FLAG AND STATUS WORD
        MC.WAK==1B0      ;ENABLED FOR WAKEUP
.MCBUF==6          ;USER ADDRESS OF TRACE BUFFER
.MCIOX==7          ;USER ADDRESS OF BUFFER INDEX
.MCCNT==10         ;USER ADDRESS OF WAKEUP COUNTER
.MCBFL==11         ;BUFFER LENGTH

;ARGUMENT BLOCK FOR METER, POINT FUNCTIONS

.MPFUN==0          ;UUO FUNCTION CODE
.MPAPP==1          ;NUM ARGS PER POINT IN LIST
.MPNUM==2          ;NUMBER OF POINTS IN LIST
.MPADR==3          ;ADDRESS OF POINT LIST
.MPERR==4          ;ADDRESS OF ERROR POINT

;ARG DISPLACEMENTS PER POINT IN POINT LIST

.MPNAM==0          ;POINT NAME
.MPPID==1          ;USER POINT ID
.MPPAR==2          ;POINT PARAMETER
.MPJOB==3          ;JOB NUMBER
.MPSTS==4          ;POINT STATUS
        MP.STS==17777B12 ;POINT STATUS MASK
        MP.ENB==1B0      ;POINT IS ENABLED
        MP.USA==1B1      ;USER SEGMENT ADDRESSED
.MPPRT==5          ;POINT ROUTINE TYPE
                   .MPRN==00        ;NULL ROUTINE
                   .MPRV==01        ;INTRINSIC VALUE
                   .MPRT==02        ;TIME INTERVAL
                   .MPRVI==3        ;INTRINSIC VALUE+POINT ID
                   .MPRTI==4        ;TIME+POINT ID
.MPPRP==6          ;POINT ROUTINE PARAMETER
.MPCID==7          ;USER CHANNEL ID
```

```
        SUBTTL  ENQUEUE AND DEQUEUE SYMBOLS

;;GENERAL FORMAT FOR ENQ./DEQ./ENQC.
;;!=========================================================!
;;!         # OF LOCKS          !   LENGTH OF THIS BLOCK    !
;;!---------------------------------------------------------!
;;!          RESERVED           !         REQUEST ID        !
;;!=========================================================!
;;!S!B! RESERVED !   LEVEL #    !      CHAN #/-1/-2/-3      !
;;!---------------------------------------------------------!
;;!           BYTE POINTER TO STRING OR USER CODE           !
;;!---------------------------------------------------------!
;;!      # OF RES. IN POOL      !    # WANTED OR GROUP #    !
;;!---------------------------------------------------------!
;;/                                                         /
;;/                   3 WORDS FOR EACH LOCK                 /
;;/                                                         /
;;!---------------------------------------------------------!
;;!S!B! RESERVED !   LEVEL #    !      CHAN #/-1/-2/-3      !
;;!---------------------------------------------------------!
;;!           POINTER TO STRING OR 5B2+USER CODE            !
;;!---------------------------------------------------------!
;;!      # OF RES IN POOL       !    # WANTED OR GROUP #    !
;;!=========================================================!

.ENQLL==0           ;NUMBER OF LOCKS AND LENGTH
        EQ.LNL==777777B17           ;NUMBER OF LOCKS
        EQ.LLB==777777B35           ;LENGTH OF BLOCK
.ENQRI==1           ;REQUEST I.D.

;FOR EACH LOCK:
.ENQFL==0           ;FLAGS, LEVEL, CHAN
        EQ.FSR==1B0         ;SHARED REQUEST
        EQ.FBL==1B1         ;BYPASS LEVEL CHECKING
        EQ.FLV==777B17      ;LEVEL #
        EQ.FCC==777777      ;CHAN. NUMBER OR CODE
            .EQFJB==777777  ;CODE FOR THIS JOB ONLY
            .EQFGL==777776  ;GLOBAL LOCK
            .EQFPL==777775  ;PRIV. GLOBAL LOCK
.ENQBP==1           ;BYTE POINTER OR USER CODE
        EQ.BUC==5B2         ;SET IF 33 BIT USER CODE IS USED
.ENQPS==2           ;POOL SIZE
        EQ.PPS==777777B17           ;TOTAL SIZE OF POOL
        EQ.PPR==777777B35           ;NUMBER REQUESTED FROM POOL
```

```
;ENQ. FUNCTION CODES
.ENQBL==0          ;ENQ. BLOCK TILL AVAILABLE
.ENQAA==1          ;ENQ. ALLOCATE ONLY IF AVAILABLE
.ENQSI==2          ;ENQ. SOFTWARE INTERRUPT WHEN AVAIL.
.ENQMA==3          ;ENQ. MODIFY ACCESS

;DEQ. FUNCTION CODES
.DEQDR==0          ;DEQ. RESOURCE
.DEQDA==1          ;DEQ. ALL
.DEQID==2          ;DEQ. BY REQUEST I.D.

;ENQC. FUNCTION CODES
.ENQCS==0          ;RETURN STATUS
.ENQCG==1          ;GET USER'S QUOTA
.ENQCC==2          ;CHANGE USERS QUOTA
.ENQCD==3          ;DUMP THE DATA BASE
```

```
;;FORMAT OF ENQC. STATUS BLOCKS (FUNCTION 1)
;;!=================================================================!
;;!I !O !Q!X!          !     LEVEL #     !     JOB # OF OWNER OR ERROR #     !
;;!-----------------------------------------------------------------!
;;!                           TIME-STAMP OF LOCK                    !
;;!-----------------------------------------------------------------!
;;!          RESERVED TO DEC            !     REQUEST ID OF CALLER/OWNER     !
;;!-----------------------------------------------------------------!
;;/                                                                 /
;;/                 TWO WORDS FOR EACH LOCK IN ENQC. REQUEST        /
;;/                                                                 /

.ENQCF==0        ;FLAG WORD
        EQ.CFI==1B0      ;LOCK IS INVALID
        EQ.CFO==1B1      ;THIS USER IS THE OWNER
        EQ.CFQ==1B2      ;THIS USER IS IN THE QUEUE
        EQ.CFX==1B3      ;THE OWNER HAS EXCLUSIVE ACCESS
        EQ.CFL==777B17   ;LEVEL NUMBER
        EQ.CFJ==777777   ;JOB # OF OWNER (OR ERROR CODE)
.ENQCT==1        ;TIME-STAMP (TIME LOCK WAS GRANTED TO OWNER
                 ;        IN UNIVERSAL FORMAT)
.ENQCI==2        ;REQUEST ID OF OWNER/CALLER
```

```
;;FORMAT FOR ENQC. DUMP
;;!=========================================================!
;;!                NUMBER OF WORDS IN THIS BLOCK            !
;;!=========================================================!
;;!                                                         !
;;!                                                         !
;;!                  LOCK-BLOCK FOR LOCK # 1                !
;;!                                                         !
;;!---------------------------------------------------------!
;;!           QUEUE BLOCK FOR FIRST ENTRY OF LOCK 1         !
;;!                                                         !
;;!---------------------------------------------------------!
;;!                                                         !
;;!          QUEUE BLOCK FOR SECOND ENTRY OF LOCK 1         !
;;!---------------------------------------------------------!
;;/                                                         /
;;/     TWO WORD QUEUE-BLOCK FOR EACH WAITER FOR LOCK 1     /
;;/                                                         /
;;!=========================================================!
;;!                                                         !
;;!                                                         !
;;!                  LOCK BLOCK FOR LOCK #2                 !
;;!                                                         !
;;!---------------------------------------------------------!
;;!        QUEUE-BLOCK FOR FIRST WAITER FOR LOCK #2         !
;;!                                                         !
;;!---------------------------------------------------------!
;;!                                                         !
;;!        QUEUE-BLOCK FOR SECOND WAITER FOR LOCK #2        !
;;!---------------------------------------------------------!
;;/                                                         /
;;/                                                         /
;;/       LOCK-BLOCKS AND QUEUE BLOCKS FOR THE ENTIRE       /
;;/                   ENQ./DEQ. DATA BASE                   /
;;/                                                         /
;;/                                                         /
```

```
;;FORMAT OF EACH LOCK-BLOCK
;;!=================================================================!
;;! 1! 0!T!0!0!        !     LEVEL #     !             LOCK I.D.    !
;;!-----------------------------------------------------------------!
;;!            # IN POOL OR 0            !          # REMAINING OR 0 !
;;!-----------------------------------------------------------------!
;;!                                TIME-STAMP                       !
;;!-----------------------------------------------------------------!
;;!             ASCIZ STRING (MAY BE SEVERAL WORDS) OR USER CODE    !
;;!=================================================================!

;;FORMAT OF EACH QUEUE-BLOCK
;;!=================================================================!
;;! 0!LO!0!X!B!                          !                JOB #      !
;;!-----------------------------------------------------------------!
;;!        GROUP # OR # REQUESTED        !             REQUEST I.D.  !
;;!=================================================================!

;FLAGS IN FIRST WORD OF EACH BLOCK TYPE:
EQ.DLB==1B0      ;THIS IS A LOCK BLOCK
EQ.DLO==1B1      ;THIS IS THE LOCK OWNER (QUEUE-BLOCK ONLY)
EQ.DLT==1B2      ;THIS LOCK HAS TEXT (LOCK-BLOCK ONLY)
EQ.DXA==1B3      ;EXCLUSIVE ACCESS (QUEUE-BLOCK ONLY)
EQ.DJW==1B4      ;THIS JOB IS BLOCKED WAITING FOR LOCK (QUEUE-BLOCK ONLY)

;FORMAT OF LOCK-BLOCK
.EQDFL==0        ;FLAGS AND LEVEL
        EQ.DFL==777B17  ;LEVEL #
        EQ.DFI==777777  ;LOCK I.D.
.EQDPR==1        ;POOLED REQUEST COUNTS
        EQ.DPS==777777B17 ;SIZE OF POOL
        EQ.DPL==777777B35 ;NUMBER LEFT
.EQDTS==2        ;TIME-STAMP
.EQDSU==3        ;STRING OR USER CODE

;FORMAT OF A QUEUE-BLOCK
.EQDFJ==0        ;FLAGS AND JOB #
        EQ.DJN==777B35  ;JOB NUMBER
.EQDGI==1        ;GROUP # AND REQUEST I.D.
        EQ.DGR==777777B17 ;GROUP OR # REQUESTED
        EQ.DRI==777777B35 ;REQUEST I.D.
```

```
;ENQ./DEQ./ENQC. ERROR CODES

ENQRU%==1          ;SOME RESOURCE(S) REQUEST WERE  UNAVAILABLE
ENQBP%==2          ;ILLEGAL # OF RESOURCES REQUESTED (POOLED RESOURCES)
ENQBJ%==3          ;BAD JOB NUMBER
ENQBB%==4          ;BAD BYTE SIZE IN TEXT STRING
ENQST%==5          ;STRING TOO LONG
ENQBF%==6          ;BAD FUNCTION CODE
ENQBL%==7          ;ILLEGAL ARGUMENT BLOCK LENGTH
ENQIC%==10         ;ILLEGAL NUMBER OF LOCKS SPECIFIED
ENQBC%==11         ;BAD CHANNEL NUMBER
ENQPI%==12         ;OPERATOR/JACCT PRIVILEGE REQUIRED
ENQNC%==13         ;NO CORE AVAILABLE
ENQFN%==14         ;FILE NOT OPEN ON SPECIFIED CHANNEL, OR DEVICE NOT A DISK
ENQIN%==15         ;INDIRECT OR INDEXED BYTE POINTER NOT ALLOWED
ENQNO%==16         ;NO RESOURCES WERE OWNED
ENQLS%==17         ;LEVEL SEQUENCING ERROR (LEVEL # TOO LOW)
ENQCC%==20         ;CAN'T CHANGE ACCESS
ENQQE%==21         ;QUOTA EXCEEDED
ENQPD%==22         ;# OF RESOURCES IN POOL NOT SAME AS IN LOCK
ENQDR%==23         ;DUPLICATE REQUEST FOR RESOURCE (LOCK ALREADY REQUESTED)
ENQNE%==24         ;NOT ENQ'ED ON THIS LOCK
ENQLD%==25         ;LEVEL # IN REQUEST DOES NOT MATCH LOCK
ENQED%==26         ;ENQ/DEQ PRIVILEGES REQUIRED
```

```
        SUBTTL  MISC. I/O -- DEVCHR

DV.DRI==1B0         ;DTA WITH DIRECTORY IN CORE
DV.DSK==1B1         ;DEVICE IS A FILE STRUCTURE
DV.CDR==1B2         ;IF DVOUT=1 DEVICE IS A CDP
                    ; IF DVIN=1 DEVICE IS A CDR
DV.LPT==1B3         ;DEVICE IS A LINE PRINTER
DV.ITA==1B4         ;DEVICE IS A TTY CONTROLING A JOB
DV.TTU==1B5         ;TTY DDB IS IN USE
DV.ITB==1B6         ;FREE BIT LEFT FROM SCNSRF
DV.DIS==1B7         ;DEVICE IS A DISPLAY
DV.LNG==1B8         ;DEVICE HAS A LONG DISPATCH TABLE
DV.PTP==1B9         ;DEVICE IS A PAPER TAPE PUNCH
DV.PTR==1B10        ;DEVICE IS A PAPER TAPE READER
DV.DTA==1B11        ;DEVICE IS A DEC TAPE
DV.AVL==1B12        ;DEVICE IS AVAILABLE TO THIS JOB
DV.MTA==1B13        ;DEVICE IS A MAG TAPE
DV.TTY==1B14        ;DEVICE IS A TTY
DV.DIR==1B15        ;DEVICE HAS A DIRECTORY
DV.IN==1B16         ;DEVICE CAN DO INPUT
DV.OUT==1B17        ;DEVICE CAN DO OUTPUT
DV.ASC==1B18        ;DEVICE ASSIGNED BY ASSIGN COMMAND
DV.ASP==1B19        ;DEVICE ASSIGNED BY INIT OR OPEN UUO
DV.M17==1B20        ;DEVICE CAN DO MODE 17
DV.M16==1B21        ;DEVICE CAN DO MODE 16
DV.M15==1B22        ;DEVICE CAN DO MODE 15
DV.M14==1B23        ;DEVICE CAN DO MODE 14
DV.M13==1B24        ;DEVICE CAN DO MODE 13
DV.M12==1B25        ;DEVICE CAN DO MODE 12
DV.M11==1B26        ;DEVICE CAN DO MODE 11
DV.M10==1B27        ;DEVICE CAN DO MODE 10
DV.M7==1B28         ;DEVICE CAN DO MODE 7
DV.M6==1B29         ;DEVICE CAN DO MODE 6
DV.M5==1B30         ;DEVICE CAN DO MODE 5
DV.M4==1B31         ;DEVICE CAN DO MODE 4
DV.M3==1B32         ;DEVICE CAN DO MODE 3
DV.M2==1B33         ;DEVICE CAN DO MODE 2
DV.M1==1B34         ;DEVICE CAN DO MODE 1
DV.M0==1B35         ;DEVICE CAN DO MODE 0
```

```
        SUBTTL  MISC. I/O -- DEVTYP

;FIRST THE TYPE CODES
.TYDSK==0           ;DEVICE IS A DISK
.TYDTA==1           ;DEVICE IS A DEC TAPE
.TYMTA==2           ;DEVICE IS A MAG TAPE
.TYTTY==3           ;DEVICE IS A TTY
.TYPTR==4           ;DEVICE IS A PTR
.TYPTP==5           ;DEVICE IS A PTP
.TYDIS==6           ;DEVICE IS A DISPLAY
.TYLPT==7           ;DEVICE IS A LINE PRINTER
.TYCDR==10          ;DEVICE IS A CARD READER
.TYCDP==11          ;DEVICE IS A CARD PUNCH
.TYPTY==12          ;DEVICE IS A PTY
.TYPLT==13          ;DEVICE IS A PLOTTER
.TYEXT==14          ;EXTERNAL TASK (DA28C)
.TYMPX==15          ;MULTIPLEXOR
.TYPAR==16          ;PA611R ON DC44
.TYPCR==17          ;PC11(R) ON DC44
.TYPAP==20          ;PA611P ON DC44
.TYLPC==21          ;LPC-11 ON DC44
.TYPCP==22          ;PC-11(P) ON DC44

;NOW THE CHARACTERISTICS
TY.MAN==1B0         ;LOOKUP/ENTER IS REQUIRED
TY.MDA==1B9         ;DEVICE IS CONTROLLED BY MOUNTABLE DEVICE
                    ; ALLOCATOR
TY.EHF==1B10        ;EXTENDED HARDWARE FEATURES:
                    ;IF LPT THEN HAS LOWER CASE
TY.MPX==1B11        ;DEVICE CAN BE USED VIA MPX:
TY.AVL==1B12        ;DEVICE IS FREE
TY.SPL==1B13        ;DEVICE IS SPOOLED
TY.INT==1B14        ;DEVICE IS INTERACTIVE
TY.VAR==1B15        ;DEVICE HAS VARIABLE BUFFER SIZE
TY.IN==1B16         ;DEVICE CAN DO INPUT
TY.OUT==1B17        ;DEVICE CAN DO OUTPUT
TY.JOB==777B26      ;JOB NUMBER OWNING DEVICE
TY.RAS==1B29        ;RESTRICTED DEVICE
TY.DEV==77B35       ;DEVICE TYPE
```

```
        SUBTTL  MISC. I/O -- MTCHR.

MT.AWC==777777B17          ;ACTUAL WORD COUNT
MT.CRC==777B26   ;CRC LAST READ
MT.NCR==7B29     ;NUMBER CHARACTERS READ IN LAST WORD
MT.7TR==1B31     ;7 TRACK
MT.WLK==1B32     ;WRITE LOCKED
MT.DEN==7B35     ;DENSITY
        .MTDN2==1          ;200
        .MTDN5==2          ;556
        .MTDN8==3          ;800
        .MTD16==4          ;1600

.MTRID==1        ;REEL ID
.MTWRD==2        ;WORDS READ (CHARS IN 6.02)
.MTWWT==3        ;WORDS WRITTEN (CHARS IN 6.02)
.MTSRE==4        ;SOFT READ ERRORS
.MTHRE==5        ;HARD READ ERRORS
.MTSWE==6        ;SOFT WRITE ERRORS
.MTHWE==7        ;HARD WRITE ERRORS
.MTTME==10       ;TOTAL MEDIA ERRORS
.MTTDE==11       ;TOTAL DEVICE ERRORS
.MTTUN==12       ;TOTAL UNLOADS
.MTNFB==13       ;NUMBER OF FILES FROM BOT
.MTNRF==14       ;NUMBER OF RECORDS FROM EOF
.MTICC==15       ;INITIAL ERROR CONI MTC
.MTICS==16       ;INITIAL ERROR CONI MTS
.MTFCC==17       ;FINAL ERROR CONI MTC
.MTFCS==20       ;FINAL ERROR CONI MTS
.MTTRY==21       ;RETRIES TO RESOLVE LAST ERROR
```

```
        SUBTTL  MISC. I/O -- TAPOP.

;;TAPOP. UUO TAKES N,,BLOCK IN AC WHERE BLOCK CONTAINS:
;;   !=================================================!
;;   !                  FUNCTION CODE                  !
;;   !-------------------------------------------------!
;;   !             DEVICE NAME, CHAN. OR UDX           !
;;   !-------------------------------------------------!
;;   !                  ARGUMENT 0                     !
;;   !-------------------------------------------------!
;;   !                  ARGUMENT 1                     !
;;   !-------------------------------------------------!
;;   /                                                 /
;;   /                                                 /
;;   /                                                 /
;;   !-------------------------------------------------!
;;   !                  ARGUMENT N-2                   !
;;   !=================================================!

;TAPOP. FUNCTIONS:
.TFWAT==1          ;WAIT FOR I/O TO STOP
.TFREW==2          ;REWIND TO LOAD POINT
.TFUNL==3          ;REWIND AND UNLOAD
.TFFSB==4          ;SKIP FORWARD 1 BLOCK
.TFFSF==5          ;SKIP FORWARD 1 FILE
.TFSLE==6          ;SKIP TO LOGICAL END OF TAPE
.TFBSB==7          ;SKIP BACKWARD 1 BLOCK
.TFBSF==10         ;SKIP BACKWARB 1 FILE
.TFWTM==11         ;WRITE TAPE MARK
.TFWLG==12         ;WRITE 3" OF BLANK TAPE
.TFDSE==13         ;DATA SECURITY ERASE (BLANK WHOLE TAPE) TU70
.TFWLE==14         ;WRITE LOGICAL END OF TAPE (WTM, WTM, BSB)
.TFLBG==15         ;LABEL GET (FOR TAPE LABEL MGR.)
.TFLRL==16         ;LABEL RELEASE (FOR TAPE LABEL MGR.)
.TFLSU==17         ;SWAP UNITS (FOR TAPE LABEL MGR.)
.TFLDD==20         ;DESTROY LABEL DDB (FOR TAPE LABEL MGR.)
.TFFEV==21         ;FORCE END OF VOLUME PROCESSING
.TFURQ==22         ;USER REQUEST
```

```
;READ PARAMETERS. RESULT TO AC.
.TFTRY==1000     ;RETRIES ON LAST ERROR
.TFDEN==1001     ;DENSITY
        .TFD00==0        ;UNIT DEFAULT
        .TFD20==1        ;200 BPI
        .TFD55==2        ;556 BPI
        .TFD80==3        ;800 BPI
        .TFD16==4        ;1600 BPI
        .TFD62==5        ;6250 BPI
.TFKTP==1002     ;CONTROLLER TYPE
        .TFKTA==0        ;TM10A
        .TFKTB==1        ;TM10B
        .TFKTC==2        ;TC10C
        .TFKTX==3        ;TX01
.TFRDB==1003     ;READ BACKWARDS (TU70 ONLY)
.TFLTH==1004     ;LOW THRESHOLD READ (TM10 ONLY)
.TFPAR==1005     ;EVEN PARITY (7TRK ONLY)
.TFBSZ==1006     ;BLOCK SIZE
.TFMOD==1007     ;MODE
        .TFMDD==0        ;DEC COMPAT. CORE DUMP
        .TFMID==1        ;INDUSTRY COMPAT. CORE DUMP
        .TFM8B==2        ;8-BIT MODE (4 BYTES/WORD)
        .TFM6B==3        ;6-BIT MODE (9-TRACK TU70 ONLY)
        .TFM7B==4        ;7-BIT MODE (TU70 ONLY)
        .TFM7I==5        ;7-TRACK CORE DUMP (SIXBIT)
.TFTRK==1010     ;7-TRACK BIT
.TFWLK==1011     ;WRITE LOCK (1=YES, 0=NO)
.TFCNT==1012     ;CHAR. COUNT OF LAST RECORD
.TFRID==1013     ;REELID
.TFCRC==1014     ;LAST CRC (9-TRACK NRZI ONLY)
.TFSTS==1015     ;UNIT STATUS
        TF.UNS==1B18     ;UNIT IS NOT TO BE SCHEDULED
        TF.BOT==1B19     ;BOT
        TF.WLK==1B20     ;WRITE LOCK
        TF.REW==1B21     ;UNIT IS REWINDING
        TF.STA==1B33     ;UNIT IS STARTED
        TF.SEL==1B34     ;UNIT IS SELECTED
        TF.OFL==1B35     ;UNIT IS OFF-LINE
```

```
.TFSTA==1016     ;UNIT STATISTICS TO ARGS 0 TO 12
        .TSFIL==0        ;NUMER OF FILES SINCE BOT (FILE #)
        .TSREC==1        ;NUMBER OF RECORDS SINCE EOF (RECORD #)
        .TSICR==2        ;TOTAL CHARS. READ
        .TSICW==3        ;TOTAL CHARS. WRITTEN
        .TSSRE==4        ;SOFT READ ERRORS
        .TSHRE==5        ;HARD READ ERRORS
        .TSSWE==6        ;SOFT WROTE ERRORS
        .TSHWE==7        ;HARD WRITE ERRORS
        .TSESU==10       ;TOTAL ERRORS SINCE UNLOAD (MOUNT)
        .TSIDE==11       ;TOTAL DEVICE ERRORS SINCE SYSTEM STARTUP
        .TSUNL==12       ;TOTAL UNLOADS
.TFIEP==1017     ;INITIAL ERROR POINTER
.TFFEP==1020     ;FINAL ERROR POINTER
.TFIER==1021     ;INITIAL ERROR STATUS
.TFFER==1022     ;FINAL ERROR STATUS
.TFFRD==1023     ;NUMBER OF RETRIES
.TFLBL==1024     ;TYPE OF LABEL PROCESSING
        .TFLBP==0        ;BYPASS LABEL PROCESSING
        .TFLAL==1        ;ANSI LABELS
        .TFLAU==2        ;ANSI LABELS WITH USER LABELS
        .TFLIL==3        ;IBM LABELS
        .TFLIU==4        ;IBM LABELS WITH USER LABELS
        .TFLTM==5        ;LEADING TAPE MARK
        .TFLNS==6        ;NON-STANDARD LABELS
        .TFLNL==7        ;NO LABELS
.TFPLT==1025     ;SAME AS .TFLBL EXCEPT PRIV SET, USED TO
                 ; SET .TFLBP AND .TFLNL.
.TFLTC==1026     ;LABEL TERMINATION CODE
        .TFTCP==1        ;CONTINUE PROCESSING
        .TFTRE==2        ;RETURN EOF
        .TFTLT==3        ;LABEL TYPE ERROR
        .TFTHL==4        ;HEADER LABEL ERROR
        .TFTTL==5        ;TRAILER LABEL ERROR
        .TFTVL==6        ;VOLUME LABEL ERROR
        .TFTDV==7        ;DEVICE ERROR
        .TFTDE==10       ;DATA ERROR
        .TFTWL==11       ;WRITE LOCK ERROR
.TFDMS==1027     ;DIAGNOSTIC MODE SET IF 1 (TU70 ONLY)
.TFFSO==1030     ;FORCE SENSE OPERATIONS IF 1 (TU70 ONLY)


.TFSET==1000     ;OFFSET FROM READ TO SET
```

```
;TAPOP. ERROR CODES
TPACS%==-1        ;ADDRESS CHECK STORING ANSWER
TPIFC%==0         ;ILLEGAL FUNCTION CODE
TPPRV%==1         ;NOT ENOUGH PRIVS.
TPNMT%==2         ;NOT A MAGTAPE
TPVOR%==3         ;VALUE OUT OF RANGE
TPACR%==4         ;ADDRESS CHECK READING ARGUMENTS
TPCBS%==5         ;PARAMETER CAN NOT BE SET
TPNIA%==6         ;TAPE NOT INITED OR ASSIGNED
```

```
        SUBTTL  MISC. I/O -- WHERE

RM.SUP==17B17   ;STATION UP STATUS
        .RMSUN==1       ;NOT IN CONTACT
        .RMSUD==2       ;DOWN
        .RMSUG==4       ;LOADING
        .RMSUL==10      ;LOADED
RM.SDU==1B13    ;DIAL-UP

        SUBTTL  MISC. I/O -- CAL11.

.C11FC==0       ;FUNCTION WORD
        C1.1NO==777777B17  ;WHICH -11
        C1.1FC==777777  ;WHICH FUNCTION
                .C11DP==0       ;DEPOSIT FUNCTION
                .C11EX==1       ;EXAMINE FUNCTION
                .C11QU==2       ;QUEUE A REQUEST
                .C11NM==3       ;RETURN NAME OF FRONT END PROG
                .C11UP==4       ;RETURN 0 IF DOWN, 1 IF UP
.C11AD==1       ;ADDRESS OF EXAMINE/DEPOSIT
.C11CN==2       ;CONTENTS TO DEPOSIT
.C11EN==1       ;START OF QUEUE ENTRY

C11NP%==1       ;NOT PRIVILEGED
C11UF%==2       ;UNKNOWN FUNCTION
C11ND%==3       ;NOT DC76
C11IU%==4       ;EXAM/DEP IN USE
C11NA%==5       ;NO ANSWER TO EXAM/DEP
C11TS%==6       ;QUEUE ENTRY TOO SHORT
C11NE%==7       ;NOT ENOUGH ARGS
```

```
        SUBTTL  MISC. I/O -- GETLCH AND TRMOP.

;GETLCH BITS

GL.ITY==1B0         ;INVISIBLE TTY (PTY)
GL.CTY==1B1         ;SYSTEM CTY
GL.DSP==1B2         ;DISPLAY CONSOLE
GL.DSL==1B3         ;DATASET DATA LINE
GL.HDP==1B5         ;HALF-DUPLEX
GL.REM==1B6         ;REMOTE TTY
GL.RBS==1B7         ;REMOTE BATCH TTY
GL.LIN==1B11        ;LINE HAS BEEN TYPED
GL.LCM==1B13        ;LOWER CASE MODE
GL.TAB==1B14        ;TABS
GL.LCP==1B15        ;LOCAL COPY
GL.PTM==1B16        ;PAPER TAPE MODE

;TRMOP. ERRORS

TOPRC%==1           ;PROTECTION CHECK
TORGB%==2           ;RANGE BAD
TOADB%==3           ;ADDRESS BAD
TOIMP%==4           ;IMPOSSIBLE
TODIL%==5           ;ERROR IN DIALLER
```

```
;TRMOP. FUNCTIONS

.TOSIP==1          ;SKIP IF INPUT PRESENT
.TOSOP==2          ;SKIP IF OUTPUT PRESENT
.TOCIB==3          ;CLEAR INPUT BUFFER
.TOCOB==4          ;CLEAR OUTPUT BUFFER
.TOOUC==5          ;OUTPUT CHARACTER
.TOOIC==6          ;OUTPUT IMAGE CHARACTER
.TOOUS==7          ;OUTPUT STRING
.TOINC==10         ;INPUT CHARACTER
.TOIIC==11         ;INPUT IMAGE CHARACTER
.TODSE==12         ;DATA SET ENABLE
.TODSC==13         ;DATA SET CALL
.TODSF==14         ;DATA SET OFF
.TORSC==15         ;RESCAN
.TOELE==16         ;SET ELEMENT
.TOEAB==17         ;ENABLE AUTO BAUD DETECT

.TOOIP==1000       ;OUTPUT IN PROGRESS
.TOCOM==1001       ;AT COMMAND LEVEL
.TOXON==1002       ;PAPER TAPE MODE
.TOLCT==1003       ;LOWER CASE TRANSLATE TO UPPER
.TOSLV==1004       ;SLAVE
.TOTAB==1005       ;ACCEPTS TABS
.TOFFM==1006       ;ACCEPTS FF AND LF
.TOLCP==1007       ;LOCAL COPY (NO ECHO)
.TONFC==1010       ;NO FREE CARRIAGE RETURN
.TOHPS==1011       ;HORIZONTAL POSITION
.TOWID==1012       ;WIDTH
.TOSND==1013       ;SEND ALLOWED (NO GAG)
.TOHLF==1014       ;HALF DUPLEX
.TORMT==1015       ;REMOTE NON-DATA SET
.TODIS==1016       ;DISPLAY CONSOLE
.TOFLC==1017       ;FILLER CLASS
.TOTAP==1020       ;PAPER TAPE ENABLED
.TOPAG==1021       ;PAGE COMMAND GIVEN
.TOSTP==1022       ;OUTPUT STOPPED (XOFF OR PAGE LIMIT)
.TOPSZ==1023       ;PAGE SIZE (HEIGHT IN LINES)
.TOPCT==1024       ;LINE COUNT IN PAGE
.TOBLK==1025       ;SUPPRESS BLANK LINES
.TOALT==1026       ;CONVERT ALTMODE (175,176) TO ESCAPE
.TOAPL==1027       ;APL MODE
.TORSP==1030       ;RECIEVE SPEED
.TOTSP==1031       ;TRANSMIT SPEED
.TODBK==1032       ;HAS DEBREAK
.TO274==1033       ;2741
.TOTDY==1034       ;TIDY MODE
.TOACR==1035       ;AUTO CRLF
.TORTC==1036       ;^R ^T COMPATIBLE MODE (DISABLED)
.TOPBS==1037       ;PIM MODE BREAK SET (4 9-BIT BYTES)
.TOSET==1000       ;OFFSET FROM GET TO SET
```

```
        SUBTTL  MISC. I/O -- GETSTS AND SETSTS

IO.IMP==1B18        ;IMPROPER MODE -- SOFTWARE DETECTECTED ERROR
IO.DER==1B19        ;DEVICE ERROR
IO.DTE==1B20        ;DATA ERROR
IO.BKT==1B21        ;BLOCK TOO LARGE
IO.ERR==17B21       ;I/O ERROR BITS
IO.EOF==1B22        ;END OF FILE
IO.ACT==1B23        ;DEVICE IS ACTIVE
IO.D29==1B29        ;DEC029 MODE (CDP ONLY)
IO.SIM==1B29        ;SUPER-IMAGE MODE (CDR ONLY)
IO.WHD==1B29        ;WRITE DISK PACK HEADERS (DSK ONLY)
IO.SSD==1B28        ;SEMI-STANDARD MODE (DTA ONLY)
IO.NSD==1B29        ;NON-STANDARD MODE (DTA ONLY)
IO.SFF==1B29        ;SUPPRESS FORM FEEDS (LPT ONLY)
IO.BOT==1B24        ;BEGINNING OF TAPE (MAG TAPE ONLY)
IO.EOT==1B25        ;END OF TAPE (MAG TAPE ONLY)
IO.PAR==1B26        ;PARITY 1=EVEN 0=ODD (MAG TAPE ONLY)
IO.DEN==3B28        ;DENSITY 0-STD 1-200 2-556 3-800 (MAG TAPE ONLY)
IO.NRC==1B29        ;READ WITH NO REREAD CHECK (MAG TAPE ONLY)
IO.PTI==1B24        ;SUBJOB IN TTY INPUT WAIT (PTY ONLY)
IO.PTO==1B25        ;SUBJOB HAS TTY OUTPUT AVAILABLE (PTY ONLY)
IO.PTM==1B26        ;SUBJOB IS IN MONITOR MODE (PTY ONLY)
IO.TEC==1B27        ;TRUTH IN ECHOING MODE (TTY ONLY)
IO.SUP==1B28        ;SUPPRESS ECHOING (TTY ONLY)
IO.FCS==1B29        ;FULL CHARACTER SET (TTY ONLY) -- OBSOLETE SYMBOL
IO.LEM==1B29        ;LINE EDITOR MODE (TTY ONLY)
IO.SYN==1B30        ;SYNCRONOUS MODE I/O
IO.UWC==1B31        ;USE USER'S WORD COUNT
IO.MOD==17B35       ;DATA MODE

;I/O MODES

.IOASC==0           ;ASCII
.IOASL==1           ;ASCII LINE
.IOPIM==2           ;PACKED IMAGE MODE
.IOIMG==10          ;IMAGE
.IOIBN==13          ;IMAGE BINARY
.IOBIN==14          ;BINARY
.IOIDP==15          ;IMAGE DUMP
.IODPR==16          ;DUMP RECORDS
.IODMP==17          ;DUMP
```

```
        SUBTTL  MISC. I/O -- OPEN AND CLOSE

;CLOSE BITS

CL.DAT==1B29        ;DELETE ACCESS TABLE FROM DISK DATA BASE
CL.RST==1B30        ;INHIBIT CREATING A NEW FILE (OR SUPERSEDING
                    ; AN OLD ONE) ON OUTPUT CLOSE
CL.NMB==1B31        ;INHIBIT DELETING NAME BLOCK ON A
                    ; CLOSE WITH ONLY A LOOKUP DONE
CL.ACS==1B32        ;INHIBIT UPDATING ACCESS DATE
CL.DLL==1B33        ;INHIBIT DEALLOCATION OF ALLOCATED
                    ; BUT UNWRITTEN BLOCKS
CL.IN==1B34         ;INHIBIT CLOSING INPUT
CL.OUT==1B35        ;INHIBIT CLOSING OUTPUT

;OPEN BLOCK

.OPMOD==0           ;MODE, ETC.
.OPDEV==1           ;DEVICE NAME
.OPBUF==2           ;BUFFER HEADER ADDRESSES

;OPEN AND PHYSICAL BITS

UU.PHY==1B19        ;BIT 19 .NE. BIT 18 OF CALLI IMPLIES
                    ; PHYSICAL DEVICE SEARCH

UU.PHS==1B0         ;SIGN BIT IN OPEN BLOCK IMPLIES PHYSICAL
                    ; DEVICE SEARCH
UU.DEL==1B1         ;DISABLE ERROR LOGGING
UU.DER==1B2         ;DISABLE ERROR RETRY
UU.ATO==1B3         ;ASYNCHRONOUS I/O
UU.IBC==1B4         ;ENABLE INHIBITING OF BUFFER CLEAR
UU.SOE==1B5         ;STOP OUTPUT ON ERROR, DISALLOW OUTPUT WITH ANY
                    ; ERROR BITS SET.
```

```
        SUBTTL  MISC. I/O -- FILOP.

;ARGUMENT BLOCK FOR FILOP.
;;  !=================================================================!
;;  !UP!                                !          FUNCTION CODE      !
;;  !-----------------------------------------------------------------!
;;  !                              I/O MODE                           !
;;  !-----------------------------------------------------------------!
;;  !                          DEVICE NAME OR UDX                     !
;;  !-----------------------------------------------------------------!
;;  !        OUTPUT BUFFER HEADER       !       INPUT BUFFER HEADER   !
;;  !-----------------------------------------------------------------!
;;  !      NUMBER OF OUTPUT BUFFERS     !     NUMBER OF INPUT BUFFERS !
;;  !-----------------------------------------------------------------!
;;  !                                   !       PTR TO LOOKUP BLOCK   !
;;  !-----------------------------------------------------------------!
;;  !        LENGTH OF PATH BLOCK       !        PTR TO PATH BLOCK    !
;;  !=================================================================!

;OFFSETS IN ARGUMENT BLOCK
.FOFNC==0       ;FUNCTION (AND FLAGS)
.FOIOS==1       ;I/O STATUS (OPEN MODE)
.FODEV==2       ;DEVICE
.FOBRH==3       ;BUFFER RING HEADER POINTERS
.FONBF==4       ;NUMBER OF BUFFER TO BUILD
.FOLEB==5       ;PTR TO LOOKUP/ENTER BLOCK (SEE .RB??? SYMBOLS)
.FOPAT==6       ;PTR TO PATH BLOCK (SEE .PT??? SYMBOLS)

;FLAGS IN .FOFNC
FO.PRV==1B0     ;JOB IS JACCT OR [1,2] AND WANT TO USE PRIVS

;FUNCTION CODES
.FORED==1       ;READ ONLY
.FOCRE==2       ;CREATE (NEW FILE ONLY)
.FOWRT==3       ;WRITE (CREATE OR SUPERCEDE)
.FOSAU==4       ;SINGLE ACCESS UPDATE
.FOMAU==5       ;MULTI-ACCESS UPDATE
.FOAPP==6       ;APPEND
.FOCLS==7       ;CLOSE (OPTIONAL FLAGS IN .FOIOS, SEE CL.???)
.FOURB==10      ;UPDATE RIB
```

```
        SUBTTL  MISC. I/O -- BUFFER HEADER FORMATS

;BUFFER HEADER FORMATS
;BUFFER RING HEADER
;;  !=========================================================!
;;  !VR!T!                          !ADDRESS OF CURRENT BUFFER !
;;  !---------------------------------------------------------!
;;  !                   BYTE POINTER TO DATA                  !
;;  !---------------------------------------------------------!
;;  !                       BYTE COUNTER                      !
;;  !=========================================================!
;;  !       (MPX: ONLY) UNIVERSAL INDEX OF THIS DEVICE        !
;;  !=========================================================!

.BFADR==0           ;ADDRESS OF BUFFER RING
        BF.VBR==1B0     ;VIRGIN BUFFER RING
        BF.IBC==1B1     ;INHIBIT BUFFER CLEAR
.BFPTR==1           ;BYTE POINTER TO DATA
.BFCTR==2           ;ITEM BYTE COUNT
.BFUDX==3           ;UNIVERSAL DEVICE INDEX (MPX: ONLY)

;INDIVIDUAL BUFFER HEADER
;;  !=========================================================!
;;  !                              !        FILE STATUS      !
;;  !---------------------------------------------------------!
;;  !US!        DATA SIZE          !    NEXT BUFFER ADDRESS  !
;;  !---------------------------------------------------------!
;;  !  (MPX:) UNIV.DEV. INDEX      !        WORD COUNT       !
;;  !=========================================================!
;;  /                                                         /
;;  /                                                         /
;;  !                            DATA                         !
;;  /                                                         /
;;  /                                                         /
;;  !=========================================================!

.BFSTS==0           ;FILE STATUS WORD
        BF.STS==0,,-1   ;FILE STATUS THIS BUFFER
.BFHDR==1           ;BUFFER CONTROL THIS BUFFER
        BF.IDU==1B0     ;BUFFER IN USE
        BF.SIZ==377777B17  ;SIZE OF BUFFER
        BF.NBA==777777  ;NEXT BUFFER ADDRESS
.BFCNT==2           ;WORD COUNT OF DATA (SOMEWHAT DEVICE DEPENDENT)

        SUBTTL  MISC. I/O -- MVHDR.

MVHDR%==1           ;CHANNEL NOT OPEN
```

```
        SUBTTL  MISC. I/O -- CNECT.,SENSE., CLRST.

;CNECT. FUNCTIONS

.CNCCN==1           ;CONNECT DEVICE
.CNCDC==2           ;CLOSE AND DISCONNECT
.CNCDR==3           ;RESET AND DISCONNECT

;CNECT. ERRORS

CNCNM%==1           ;NOT MPX: CHANNEL
CNCUD%==2           ;UNKNOWN DEVICE
CNCCM%==3           ;CAN'T MULTIPLEX THIS DEVICE
CNCNF%==4           ;NO FREE STORAGE
CNCNC%==5           ;NOT CONNECTED
CNCNO%==6           ;CHANNEL NOT OPEN
CNCII%==7           ;INVALID UNIVERSAL DEVICE INDEX
CNCUF%==10          ;UNKNOWN FUNCTION
CNCDU%==11          ;DEVICE UNAVAILABLE
CNCSD%==12          ;SPOOLED DEVICE

;SENSE. SUB-BLOCK

.SNSDV==0           ;DEVICE NAME IN SIXBIT
.SNSST==1           ;GETSTS
.SNSDS==2           ;DEVSTS

;SENSE. ERRORS

SNSBD%==1           ;BAD DEVICE

;ERLST. ERRORS

ERLBC%==1           ;BAD CHANNEL
ERLNM%==2           ;NOT MPX: CHANNEL

;CLRST. BLOCK

.CLRSX==0           ;UDX
.CLRSI==1           ;SETSTS

;CLRST. ERRORS

CLRID%==1           ;ILLEGAL DEVICE
CLRNO%==2           ;NOT OWN DEVICE
```

```
        SUBTTL  MISC. I/O -- DEVLNM

DVLNX%==-1          ;NON-EXISTENT DEVICE
DVLIU%==-2          ;LOGICAL NAME IN USE
DVLNA%==-3          ;DEVICE NOT ASSIGNED OR OPEN

        SUBTTL  MISC. I/O -- DEVSIZ

DVSDM%==0           ;DUMP MODE
DVSNX%==-1          ;NON-EXISTENT DEVICE
DVSIM%==-2          ;ILLEGAL MODE

        SUBTTL  MISC. I/O -- MTAID.

MTINX%==-1          ;DEVICE DOES NOT EXIST OR NOT A MAG TAPE
MTINA%==-2          ;DEVICE IS NOT AVAILABLE TO THIS JOB

;REMOVED (NEED TO SPY TO GET THIS INFO)
;HARDWARE CHANNEL DATA BLOCK WORDS

;.CNBSY==0          ;BUSY IF POSITIVE
;.CNSYS==1          ;LH=ADDRESS OF NEXT BLOCK
;.CNLUE==1          ;RH=ADDRESS OF UNIT WITH LAST ERROR
;.CNICW==2          ;INITIAL C.W. ON LAST ERROR
;.CNFCW==3          ;FINAL C.W. ON LAST ERROR
;.CNCW2==4          ;COMMAND WORD-2 OF ERROR
;.CNCW1==5          ;-1
;.CNCW0==6          ;-0
;.CNDW2==7          ;DATA WORD-2
;.CNDW1==10         ;DATA WORD-1
;.CNDW0==11         ;DATA WORD-0
;.CNMPE==12         ;NO. CHANNEL MEMORY PARITY ERRORS
;.CNDPE==13         ;NO. DATA PARITY ERRORS
;.CNNXM==14         ;NO. CHANNEL NXM
;.CNCSR==15         ;LH=BITS TO REQUEST CPU SWEEP OF CORE
;.CNLDE==15         ;RH=LAST DDB ADDR
;.CNCBL==16         ;LENGTH OF DATA BLOCK
```

```
        SUBTTL  DISK UUOS -- DSKCHR

;DSKCHR STATUS BITS
;;  !=================================================================!
;;  !R!O!H!S!A!Z! !STS!M!N!L!     ! TYP ! DCN !  CNT-TYP  ! CNN ! UNT ! UNN !
;;  !=================================================================!

DC.RHB==1B0      ;READ HOME BLOCK
DC.OFL==1B1      ;UNIT IS OFF-LINE
DC.HWP==1B2      ;HARDWARE WRITE PROTECT
DC.SWP==1B3      ;SOFTWARE WRITE PROTECT
DC.SAF==1B4      ;SINGLE ACCESS FILE STRUCTURE
DC.ZMT==1B5      ;ZERO MOUNT COUNT
DC.STS==3B8      ;UNIT STATUS
        .DCSTP==0        ;PACK IS MOUNTED
        .DCSTN==2        ;NO PACK IS MOUNTED
        .DCSTD==3        ;UNIT IS DOWN
DC.MSB==1B9      ;MULTIPLE SAT BLOCKS
DC.NNA==1B10     ;NO NEW ACCESSES
DC.AWL==1B11     ;WRITE LOCKED FOR ALL JOBS
DC.TYP==7B17     ;TYPE OF ARGUMENT
        .DCTDS==0        ;GENERIC DSK
        .DCTAB==1        ;SUBSET DUE TO ABBREVIATIONS
        .DCTFS==2        ;FILE STRUCTURE NAME
        .DCTUF==3        ;UNIT WITHIN F/S
        .DCTCN==4        ;CONTROLLER CLASS NAME
        .DCTCC==5        ;CONTROLLER CLASS
        .DCTPU==6        ;PHYSICAL UNIT
DC.DCN==7B20     ;DATA CHANNEL NUMBER
DC.CNT==77B26    ;CONTROLLER TYPE
        .DCCFH==1        ;RC-10
        .DCCDP==2        ;RP-10
DC.CNN==7B29     ;CONTROLLER NUMBER
DC.UNT==7B32     ;UNIT TYPE
        .DCUFD==0        ;RD-10
        .DCUFM==1        ;RM-10B
        .DCUD2==1        ;RP02
        .DCUD3==2        ;RP03
DC.UNN==7B35     ;PHYSICAL UNIT NUMBER
```

```
;DSKCHR LOCATIONS

.DCNAM==0           ;ARGUMENT NAME
.DCUFT==1           ;LOGGED IN BLOCKS REMAINING
        DC.NPA==1B0     ;NO PREVIOUS ACCESS
.DCFCT==2           ;PHYSICAL FCFS BLOCKS REMAINING
.DCUNT==3           ;PHYSICAL UNIT BLOCKS REMAINING
.DCSNM==4           ;STRUCTURE NAME
.DCUCH==5           ;CHARACTERISTIC SIZES
        DC.UCC==777B8   ;BLOCKS/CLUSTER
        DC.UCT==777B17  ;BLOCKS/TRACK
        DC.UCY==777777  ;BLOCKS/CYLINDER
.DCUSZ==6           ;UNIT SIZE IN BLOCKS
.DCSMT==7           ;STRUCTURE MOUNT COUNT
.DCWPS==10          ;WORDS/SAT
.DCSPU==11          ;SATS/UNIT
.DCK4S==12          ;K FOR SWAPPING
.DCSAJ==13          ;SINGLE ACCESS JOB
.DCULN==14          ;UNIT LOGICAL NAME
.DCUPN==15          ;UNIT PHYSICAL NAME
.DCUID==16          ;UNIT ID
.DCUFS==17          ;UNIT FIRST BLOCK FOR SWAPPING
.DCBUM==20          ;BLOCKS PER UNIT INCL. MAINT CYLS.
.DCCYL==21          ;CURRENT CYLINER
.DCBUC==22          ;BLOCKS PER UNIT IN PDP-11 COMPAT. MODE
.DCLPQ==23          ;LENGTH OF POSITION WAIT QUEUE
.DCLTQ==24          ;LENGTH OF TRANSFER WAIT QUEUE


        SUBTTL  DISK UUOS -- CHKACC

.ACCPR==0           ;CHANGE PROTECTION
.ACREN==1           ;RENAME
.ACWRI==2           ;WRITE
.ACUPD==3           ;UPDATE
.ACAPP==4           ;APPEND
.ACRED==5           ;READ
.ACEXO==6           ;EXECUTE
.ACCRE==7           ;CREATE
.ACSRC==10          ;SEARCH DIRECTORY
```

```
        SUBTTL  DISK UUOS -- DISK.

.DUPRI==0           ;SET PRIORITY
.DUSEM==1           ;SET PDP-11 (22-SECTOR) MODE ON RP04
.DUSTM==2           ;SET PDP-10 (20-SECTOR) MODE ON RP04
.DUUNL==3           ;UNLOAD RP04
.DUOLS==4           ;CHAN./CONTROLLER WILL BE OFF LINE SOON
.DUOLN==5           ;CHAN./CONTROLLER IS OFF LINE NOW
.DUONL==6           ;CHAN./CONTROLLER IS BACK ON LINE

;DISK. ERRORS

DUILF%==-1          ;ILLEGAL FUNCTION
DUILP%==-2          ;ILLEGAL PRIORITY
;****NEED MORE ERROR CODES*******
```

```
        SUBTTL  DISK UUOS -- JOBSTR

.DFJNM==0           ;STR NAME
.DFJDR==1           ;DIRECTORY
.DFJST==2           ;STATUS
        DF.SWL==1B0         ;WRITE LOCKED
        DF.SNC==1B1         ;NO CREATE

        SUBTTL  DISK UUOS -- GOBSTR

.DFGJN==0           ;JOB NUMBER
.DFGPP==1           ;JOB P,PN
.DFGNM==2           ;STR NAME
.DFGDR==3           ;DIRECTORY
.DFGST==4           ;STATUS (SAME AS .DFJST)

;GOBSTR ERRORS

DFGIF%==3           ;ILLEGAL STR
DFGPP%==6           ;INCORRECT PPN
DFGNP%==10          ;NOT PRIV.
DFGLN%==12          ;INCORRECT LENGTH

        SUBTTL  DISK UUOS -- SUSET.

SU.SOT==1B1         ;OUTPUT
SU.SMN==1B2         ;MAINTENANCE CYLINDER
SU.SCH==17B12       ;CHANNEL
SU.SBL==37,,777777  ;BLOCK NUMBER

;SUSET. ERRORS

SUSNP%==-1          ;NOT PRIVILEGED
```

```
        SUBTTL  DISK UUOS -- PATH.

.PTFCN==0           ;JOB #,,FUNCTION OR ARGUMENT CHANNEL OR DEVICE
                    .PTFRD==-1          ;READ DEFAULT
                    .PTFSD==-2          ;SET DEFAULT PATH
                    .PTFSL==-3          ;SET LIB, NEW, SYS
                    .PTFRL==-4          ;READ LIB, NEW, SYS
.PTSTR==0           ;ANSWER HAS STR NAME
.PTSWT==1           ;SWITCHES AND FLAGS
        PT.SLT==7B29        ;TYPE OF SEARCH LIST
                    .PTSLJ==1           ;JOB
                    .PTSLA==2           ;ALL
                    .PTSLS==3           ;SYS
        PT.IPP==1B30        ;IMPLIED PPN (FORCED)
        PT.LIB==1B31        ;/LIB
        PT.SYS==1B32        ;/SYS
        PT.NEW==1B33        ;/NEW
        PT.SCN==3B35        ;SCAN SWITCH
                    .PTSCN==1           ;NO (OFF)
                    .PTSCY==2           ;YES (ON)
        PT.SNW==1B34        ;/NEW ON .PTFSL/.PTFRL
        PT.SSY==1B35        ;/SYS ON .PTFSL/.PTFRL
.PTPPN==2           ;PPN (UFD) OF PATH
.PTMAX==11          ;LAST POSSIBLE 0 AFTER LAST SFD +1
                    ;(IE, LENGTH OF PATH BLOCK)
```

```
        SUBTTL  DISK UUOS -- STRUUO

.FSSRC==0           ;UPDATE THIS SEARCH LIST (SEE .DFJXX)
.FSDSL==1           ;UPDATE SYSTEM/JOB SEARCH LIST
        .FSDJN==1           ;JOB# (0=SYS)
        .FSDPP==2           ;PPN
        .FSDFL==3           ;FLAGS
                DF.SRM==1B35    ;REMOVE FROM S/L COMPLETELY
.FSDEF==2           ;DEFINE NEW F/S
        .FSNST==1           ;POINTER TO STR PARAMS
        .FSNUN==2           ;FIRST POINTER TO UNIT PARAMS
    ;STR PARAM BLOCK
        .FSSNM==0           ;NAME OF STRUCTURE
        .FSSNU==1           ;NUMBER OF UNITS
        .FSSHL==2           ;HIGHEST LOGICAL BLOCK
        .FSSSZ==3           ;SIZE OF STR
        .FSSRQ==4           ;RESERVED QUOTA
        .FSSRF==5           ;RESERVED FREE
        .FSSTL==6           ;TALLY OF FCFS FREE
        .FSSOD==7           ;BLOCKS FOR OVERDRAW
        .FSSMP==10          ;MFD FIRST RETRIEVAL POINTER
        .FSSML==11          ;-1 IF .FSSMP IS ONLY POINTER
        .FSSUN==12          ;MFD UNIT
        .FSSTR==13          ;NUMBER OF RETRIES ON ERROR
        .FSSBU==14          ;LARGEST BLOCK ON UNIT
        .FSSBC==15          ;BLOCKS PER SUPER-CLUSTER
        .FSSSU==16          ;SUPER-CLUSTERS PER UNIT
        .FSSIG==17          ;(IGNORED)
        .FSSCC==20          ;BYTE POINTER TO CLUSTER COUNT
        .FSSCK==21          ;BYTE POINTER TO CHECKSUM
        .FSSCA==22          ;BYTE POINTER TO CLUSTER ADDRESS
    ;UNIT DATA BLOCK
        .FSUNM==0           ;UNIT NAME
        .FSUID==1           ;PACK ID
        .FSULN==2           ;LOGICAL NAME
        .FSULU==3           ;NUMBER WITHIN STR
        .FSUDS==4           ;STATUS BITS
                FS.UWL==1B0     ;SOFTWARE WRITE-LOCK
                FS.USA==1B1     ;SINGLE ACCESS
        .FSUGP==5           ;NUMBER BLOCKS TO ALLOCATE
        .FSUTL==6           ;FREE BLOCK TALLY
        .FSUBC==7           ;BLOCKS PER CLUSTER
        .FSUCS==10          ;CLUSTERS PER SAT
        .FSUWS==11          ;WORDS PER SAT
        .FSUSC==12          ;SATS IN CORE
        .FSUSU==13          ;SATS PER UNIT
        .FSUSP==14          ;POINTER TO SPT TABLE
```

```
;(CONT.) OF STRUUO FUNCTIONS

.FSRDF==3           ;CHANGE F/S STATUS
        .FSRJN==1           ;JOB NUMBER
        .FSRPP==2           ;JOB P,PN
        .FSRNM==3           ;STR NAME
        .FSRST==4           ;NEW STATUS
                FS.RWL==1B0         ;WRITE LOCK ALL USERS
                FS.RSA==1B1         ;SINGLE ACCESS
.FSLOK==4           ;LOCK F/S
.FSREM==5           ;REMOVE F/S
.FSULK==6           ;TEST/SET UFD INTERLOCK
.FSUCL==7           ;CLEAR UFD INTERLOCK
.FSETS==10          ;SIMULATE ERROR
        .FSEUN==1           ;UNIT
        .FSEGT==2           ;NUMBER OF TRANSFERS BEFORE ERROR
        .FSEDB==3           ;NUMBER OF DATAI'S TO RUIN
        .FSEDO==4           ;OR TO DATAI
        .FSEDA==5           ;ANDCAM TO DATAI
        .FSECB==6           ;NUMBER OF CONI'S TO RUIN
        .FSECO==7           ;OR TO CONI
        .FSECA==10          ;ANDCAM TO CONI
.FSMNW==11          ;MODIFY NOCREATE AND WRITE LOCK
        .FSMFS==1           ;FILE STRUCTURE
        .FSMFL==2           ;FLAGS
                FS.MWL==1B0         ;WRITE LOCK
                FS.MNC==1B1         ;NO CREATE

;STRUUO ERRORS

FSILF%==0           ;ILLEGAL FUNCTION CODE
FSSNF%==1           ;STR NOT FOUND
FSSSA%==2           ;STR IS SINGLE ACCESS
FSILE%==3           ;ILLEGAL ENTRY IN LIST
FSTME%==4           ;TOO MANY ENTRIES IN S/L
FSUNA%==5           ;UNIT NOT AVAILABLE
FSPPN%==6           ;PPN DOES NOT MATCH
FSMCN%==7           ;MOUNT COUNT GREATER THAN ONE
FSNPV%==10          ;NOT PRIVILEGED USER
FSFSA%==11          ;STRUCTURE ALREADY EXISTS
FSILL%==12          ;ILLEGAL ARGUMENT LIST LENGTH
FSUNC%==13          ;UNABLE TO COMPLETE UUO
FSNFS%==14          ;SYSTEM FULL OF STRS
FSNCS%==15          ;INSUFFICIENT FREE CORE FOR DATA BLOCKS
FSUNF%==16          ;ILLEGAL UNIT
FSRSL%==17          ;STR REPEATED IN S/L
```

```
        SUBTTL  LOOKUP/ENTER/RENAME

;DEFINE RIB LOCATIONS (IE, INDEX IN EXTENDED LOOKUP/ENTER BLOCK)

.RBCNT==0           ;COUNT OF ARGS FOLLOWING
        RB.NSE==1B18     ;(ENTER ONLY) NON-SUPERSEDING ENTER
.RBPPN==1           ;DIRECTORY NAME OR POINTER
.RBNAM==2           ;FILENAME
.RBEXT==3           ;EXTENSION, ACCESS DATE, ERROR CODE
        RB.CRX==7B20     ;EXTENSION OF RB.CRD
        RB.ACD==77777    ;ACCESS DATE
.RBPRV==4           ;PRIVILEGE, MODE, CREATION TIME AND DATE
        RB.PRV==777B8    ;PRIVILEGE
        RB.MOD==17B12    ;MODE
        RB.CRT==3777B23 ;CREATION TIME
        RB.CRD==7777B35 ;CREATION DATE
.RBSIZ==5           ;LENGTH
.RBVER==6           ;VERSION
.RBSPL==7           ;SPOOLED FILE NAME
.RBEST==10          ;ESTIMATED LENGTH
.RBALC==11          ;ALLOCATION
.RBPOS==12          ;POSITION TO ALLOCATE
.RBFT1==13          ;DEC NON-PRIV. FUTURE ARG
.RBNCA==14          ;NON-PRIV. CUSTOMER ARG
.RBMTA==15          ;TAPE LABEL
.RBDEV==16          ;LOGICAL UNIT NAME
.RBSTS==17          ;FILE STATUS BITS
        RP.LOG==1B0      ;LOGGED IN
        RP.UCE==1B9      ;CHECKSUM ERROR
        RP.UWE==1B10     ;WRITE ERROR
        RP.URE==1B11     ;READ ERROR
        RP.UER==7B11     ;ALL UFD ERRORS
        RP.DIR==1B18     ;DIRECTORY
        RP.NDL==1B19     ;NO DELETES
        RP.NFS==1B21     ;DON'T FAILSAFE
        RP.ABC==1B22     ;ALWAYS BAD CHECKSUM
        RP.ABU==1B24     ;ALWAYS BACK UP
        RP.NQC==1B25     ;NON-QUOTA CHECKED FILE
        RP.CMP==1B26     ;UFD COMPRESSING
        RP.FCE==1B27     ;CHECKSUM ERROR
        RP.FWE==1B28     ;WRITE ERROR
        RP.FRE==1B29     ;READ ERROR
        RP.BFA==1B32     ;BAD BY FAILSA RESTORE
        RP.CRH==1B33     ;CLOSED AFTER CRASH
        RP.BDA==1B35     ;BAD BY DAMAGE ASSESSMENT
        RP.ERR==715      ;ALL FILE ERRORS
```

```
.RBELB==20        ;ERROR LOGICAL BLOCK
.RBEUN==21        ;ERROR UNIT AND LENGTH
.RBQTF==22        ;FCFS LOGGED-IN QUOTA
.RBQTO==23        ;LOGGED-OUT QUOTA
.RBQTR==24        ;RESERVED QUOTA
.RBUSD==25        ;BLOCK IN USE
.RBAUT==26        ;AUTHOR
.RBNXT==27        ;CONTINUED STR
.RBPRD==30        ;PREDECESSOR STR
.RBPCA==31        ;PRIV. CUSTOMER ARG
.RBUFD==32        ;POINTER BACK TO UFD
.RBFLR==33        ;RELATIVE BLOCK IN FILE COVERED BY THIS RIB
.RBXRA==34        ;POINTER TO NEXT RIB IN CHAIN
.RBTIM==35        ;CREATION DATE,,TIME IN INTERNAL SYSTEM FORMAT
```

```
;LOOKUP/ENTER/RENAME/GETSEG/RUN ERROR CODES

ERFNF%==0          ;FILE NOT FOUND
ERIPP%==1          ;INCORRECT PPN
ERPRT%==2          ;PROTECTION FAILURE
ERFBM%==3          ;FILE BEING MODIFIED
ERAEF%==4          ;ALREADY EXISTING FILE NAME
ERISU%==5          ;ILLEGAL SEQUENCE OF UUOS
ERTRN%==6          ;TRANSMISSION ERROR
ERNSF%==7          ;NOT A SAVE FILE
ERNEC%==10         ;NOT ENOUGH CORE
ERDNA%==11         ;DEVICE NOT AVAILABLE
ERNSD%==12         ;NO SUCH DEVICE
ERILU%==13         ;ILLEGAL UUO
ERNRM%==14         ;NO ROOM
ERWLK%==15         ;WRITE-LOCKED
ERNET%==16         ;NOT ENOUGH TABLE SPACE
ERPOA%==17         ;PATIAL ALLOCATION
ERBNF%==20         ;BLOCK NOT FREE
ERCSD%==21         ;CAN'T SUPERSEDE A DIRECTORY
ERDNE%==22         ;CAN'T DELETE NON-EMPTY DIRECTORY
ERSNF%==23         ;SFD NOT FOUND
ERSLE%==24         ;SEARCH LIST EMPTY
ERLVL%==25         ;SFD NEST LEVEL TOO DEEP
ERNCE%==26         ;NO-CREATE FOR ALL S/L
ERSNS%==27         ;SEGMENT NOT ON SWAP SPACE
ERFCU%==30         ;CAN'T UPDATE FILE
ERLOH%==31         ;LOW SEG OVERLAPS HI SEG (GETSEG)
ERNLI%==32         ;NOT LOGGED IN (RUN)

;FILE PROTECTION CODES

.PTCPR==0          ;CHANGE PROTECTION
.PTREN==1          ;RENAME
.PTWRI==2          ;WRITE
.PTUPD==3          ;UPDATE
.PTAPP==4          ;APPEND
.PTRED==5          ;READ
.PTEXO==6          ;EXECUTE
.PTNON==7          ;NO-ACCESS

;DIRECTORY PROTECTION CODES

PT.LOK==4          ;ALLOW LOOKUPS
PT.CRE==2          ;ALLOW CREATES
PT.SRC==1          ;SEARCH DIRECTORY
```

```
        SUBTTL

        .XCREF

%%UUOS==<VRSN. (UUO)>
        PURGE   UUOWHO,UUOVER,UUOMIN,UUOEDT,%%MACT,CALLI,MTAPE,TTCALL

IFDEF %..C,<    %%C==%%UUOS     >


        IF1,<   ASUPPRESS>
        IFNDEF %..C,<PURGE VRSN.
                     END>       ;END UNIVERSAL OF UUOSYM

        IFDEF %.C,<IFLE %.C+2,<
                        IF2,<PURGE %.C,%..C>
                        END>>   ;BIND OFF TO GET CLEAN LISTING
        .CREF
        LIST
```

APPENDIX K
2741 TERMINALS

The DECsystem-10 provides support for several versions of IBM Model 2741 Terminals. This support is provided through the use of a DC76 Communications Interface. The DC76 converts the 2741 character codes into ASCII character codes.

Model 2741 terminals have upper and lower case capabilities, but they have no control characters. Model 2741's operate in half-duplex mode only. Control characters can be obtained on a 2741 by typing the circumflex key (sometimes the backarrow, the logical not sign, the hook, or the plus/minus sign character must be used in place of the circumflex character). The Circumflex key is held in the down position while typing the desired control character (whether it be C, or T, or whatever).

Figure K-1 illustrates a standard Model 2741 keyboard. Table K-1 lists the characters a user must type to generate the special characters used on other terminals.

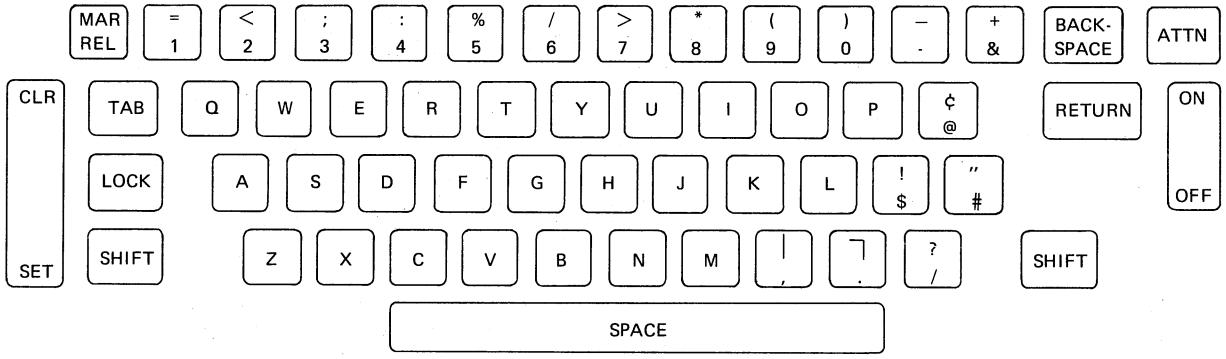


Figure K-1 Model 2741 Keyboard

Table K-1
Conversion of Characters

Type the following on a 2741	To produce the following
^^	^
^$	ESCape
[or ^(or ^<	left square bracket
] or ^] or ^>	right square bracket
< or ^[	left angle bracket
> or ^]	right angle bracket
^/ or c	backslash (\)
^4	034 file separator
^5	035 file separator
^6	036 record separator
^7	037 unit separator
^3	left brace
^1	vertical bar
^2	right brace
^-	tilde
^,	accent grave

NOTE

Each occurrence of the ^character may be substituted with a \ or a plus/minus character. If the circumflex character is followed by any other non-alphabetic character, the circumflex is ignored and the non-alphabetic character is output to the buffer. (These translations do not occur in APL mode.)

It is recommended that you use a 2741 with the Transmit Interrupt Feature (i.e., the ATTN key, IBM feature number 7900). Although this feature is not required, it is strongly recommended. The ATTN key, with this feature, unlocks a locked keyboard, enabling you to type. ATTN also locks an unlocked keyboard enabling output to be printed from the system without user interference. The DC76 recognizes a 2741 through the use of automatic baud detection (a 2741's baud rate is 134.5).

K.1 USER INTERFACE

Alphanumeric characters (ASCII characters 40-176) will appear to be handled in the same manner as other type terminals. For example, when you press the uppercase A key on a 2741, the ASCII character code 101 is received by the system.

A special set of SET TTY commands can be utilized for Model 2741 terminals. Refer to *DECsystem-10 Operating System Commands* for further information. The SET TTY ELEMENT command changes the 2741 typing element number. The usable typing element numbers are listed in Table K-2.

Table K-2
Model 2741 Typing Elements

Element Number	Type Face
987	APL correspondence
087	CALL 360 BASIC
938	BCD
029	Standard Correspondence
963	Extended Binary (EBCD)
988	APL (EBCD)

The SET TTY TIDY command specifies that every character is to occupy one print position. The 2741 terminal normally types characters on the page the way in which they were typed. For example, assume a character on the upper row of the keyboard had the characters : and 4 on it. If you type a 4, a 4 will be printed on the paper. If you type a : (by pressing the SHIFT key and typing the 4/: key), a *4:* will be printed on the paper. In TIDY mode, though, if you press the SHIFT key with the 4/: key, a : will be printed on the paper.

INDEX

2 741 characteristics, 11-8

AC,
  display contents in lights, 4-11
Access protection,
  directory, 8-7
  file, 8-5
Accounting file, 4-12
Accumulated,
  error count, 3-1
  run time, 4-9
Activate dormat file, 43
Address,
  break condition, 4-8
  highest legal, 3-2
  space for program, 2-1
Allocation tables, 8-5
ALT Mode, 11-8
APL Mode, 11-9
Append to file, 8-10
APR,
  clock, 4-10
  traps, 4-2
ASCII
  character codes, 11-9
  Line Mode, 10-2
  Mode, 7-7, 10-2
ASCIZ string,
  output a, 11-2, 11-6
ASSIGN command, 7-2
Associating
  a UDX to a device, 13-2
  Device to channel, 13-2
Automatic baud detection, 11-6
Awaken job, 4-4

Background job, 4-3
Backspace,
  a file, 10-3, 10-4
  a record, 10-3
Batch, 4-7
Binary Mode, 7-7
Block,
  Allocation of, 9-5
  Directory, 9-2
  Format on DECtape, 9-5
  Header, 7-10, 13-1
  Numbers, 9-2
  Size,
    change for magtape, 10-1
    disk, 8-1, 8-5
    magnetic tape, 10-1, 10-8
BATCON, 4-7
BATMAX, 4-7
Baud rate, 11-16
Break characters, 11-9
Buffer,
  Data mode, 7-3
  Header block,
    structure, 7-10, 13-1
    specifying, 7-3
    extension to, 13-1
  Input,
    clear, 11-3, 11-6
    MPX, 13-2
    scan, 11-3
    terminal, 11-6
    general, Chapter 7
  Output, 7-9, 11-3, 11-6
  Ring, 7-10, 13-2
  Size,
    DECtape, 9-1
    Disk, 7-6
    Magentic tape, 10-1, 10-2
    PTYs, 11-14
    Terminals, 11-1

CCONT command, 4-1
CDP spooling, 4-6
CDR spooling, 4-6
Chains, device, 13-1
Change,
  mag tape block size, 10-1
  mag tape density, 10-1
Channel,
  connect devices to, 13-1, 13-2
  MPX, Chapter 13
Character,
  codes,
    ASCII, 11-9
    Break, 11-9
    Mag tape, 10-1

Terminating, 11-9
input a, 11-2
output a, 11-2
Characteristics
line, 11-2
mag tape, 10-1
Clear,
PC flags, 4-2
REELID, 10-12
terminal buffers, 11-3, 11-6
write-protect hit, 2-1
Clock,
APR, 4-10
DK10, 4-10
enter request in queue, 4-12
internal, 4-10
CLOSE monitor call, 9-8
Closing
magnetic tape file, 10-1
DECtape file, 9-8
files, 8-10
CLRST. monitor call, 13-2
Cluster,
af files, 8-2
sizes, 8-5
CNECT. monitor call, 13-2, 13-3
Commands,
ASSIGN, 7-2
CCONT, 4-1
CONT, 4-1
CSTART, 4-1
MOUNT, 7-2
PRESERVE, 8-2
PROTECT, 8-7
REENTER, 4-1
RUN, 4-1
START, 4-1
Configuration information, 4-11
Connect device to MPX channel, 13-1, 13-2
CONT command, 4-1
Continue,
device after error, 13-5
program
after HALT, 4-1
automatically, 4-2
SFD, 8-2
UFD, 8-2
Control,
Flags, 5-7, 5-9
Job, 4-1
Software Interrupt System, 5-9
Controller,
DECtape, 9-1
Mag tape, 10-1, 10-8
Physical class names, 7-5
Terminal, 11-1
Convert ALT modes, 11-8
Core,
guaranteed amount of, 4-5
highest relative location, 3-1
image file, 2-1
maximum amount of, 2-1
CPPC, 2-5
CPPL, 2-5
CPU,
contents of switches, 4-11
runnability, 4-6
specification, 4-6
Create a file, 8-10
CSTART command, 4-1
CTLJOB monitor call, 11-17
CTRL/C, 4-1
CVPC, 2-5
CVPL, 2-5

DAEMON monitor call, 4-12
Data,
Block, 8-5
Modes,
DECtape, 9-1
Disk, 7-6
Dump, 7-7, 10-2
General information on, 7-6
Mag tape, 10-2
MPX devices, 13-2
Terminals, 11-1
Switches, 4-11
Transmission, 8-14
DATE monitor call, 4-11
Date,
algorithm for, 4-10
set, 4-5
Daytime,
algorithm for, 4-10
set, 4-5
DDT, 3-2
Debreak conditions, 11-8
DECtape,
Block allocation, 9-5
Block format, 9-5
Block numbers, 9-2
Buffered data modes, 9-1

Buffer size, 9-1
CLOSE call, 9-8
Controller number, 9-1
Data modes for, 9-1
Dead reckoning, 9-11
DEVSTS call, 9-9
Directory, 9-2, 9-3
ENTER, 9-7
File,
Format, 9-4, 9-5
Status, 9-9
I/O, Chapter 9
INPUT, 9-8
LOOKUP, 9-5, 9-6
Mnemonics, 9-1
MTAPE., 9-9
Number
of Blocks, 9-2
of files, 9-2
OUTPUT, 9-8
Physical name, 9-1
positioning, 9-11
RELEASE, 9-8,
RENAME, 9-8
Unbuffered modes, 9-2
Unit number, 9-1
UGETF, 9-9
USETI, 9-8, 9-9
USETO, 9-9
UTPCLR, 9-9
Default directory path, 8-3
Deferred spooling, 4-8
Delete,
a file, 8-9
empty SFD, 8-10
empty UFD, 8-10
Density, 10-1
Device,
Associate
to channel, 7-2
to MPX channel, 13-1, 13-2
with UDX, 13-2
Chains, 13-1
Connect to MPX channel, 13-1, 13-2
Continue after error, 13-5
Disconnect from MPX channel, 13-2
Errors, 13-3
Generic names, 7-2, 7-4
Identification, 13-2
Initialization, 7-2
Mag tape, 10-1
Names, 7-4
Physical names, 7-2, 7-4
Reassignment, 7-13
Selection, 7-2
Specification, 7-1
Termination, 7-3
Digital compatible mode, 10-3
Directory,
Access protection, 8-7
Block, 9-2
Devices, 7-2
File, 8-1
Format, 9-2, 9-3
Multiple file, 7-2
Path, 8-3
Disk,
Block size, 8-1, 8-5
Data modes, 7-7
Unit names, 7-6
DK10 clock, 4-10
Dormant job, 4-3
Double tape mark, 10-1
DPA, 7-6
DSKFULL, 4-7
Dump data mode, 7-7, 10-2
Dump record data mode, 10-2

EBOX, 4-10
Enable modem, 11-6
ENTER monitor call,
Argument Block, 8-18
General description, 8-10, 8-11
Extended argument block, 8-10
Error recovery for, 8-14
DECtape, 9-7
Parameters for, 9-7
On existent file, 8-14
Enqueue/Dequeue, Chapter 16
ERLST. monitor call, 13-3, 13-4
Error
Continue device after, 13-5
Count, 3-1
Codes,
CLRST., 13-5
CNECT., 13-3
ERLST., 13-5
IPCF, 15-12
PISAV., 5-11
PIRST., 5-12
PISYS., 5-10
TAPOP., 10-10

TRMOP, 11-9
Device, 13-5
File entry, 4-12
Intercepting, 5-2, 5-3, 5-4
Logging, 7-3
Mag tape status, 10-9
Recovery, 8-19
Retry, 7-3
EXE files, 4-1
Execution,
start, 4-1
stop, 4-1
suspend, 4-2
Extended argument block, 8-10
Extension to file name, 8-8
EXIT monitor call, 4-2

FACT file entry, 4-12
FHA, 7-6
File,
access privileges, 8-5
backspace a, 10-3, 10-4
close on mag tape, 10-1
cluster, 8-2
create, 8-10
definition, 8-1
delete, 8-9
directories, 8-1 to 8-4
format,
on DECtape, 9-2
on disk, 8-5
generation, 8-1
length of, 8-1
names, 8-8
number of on DECtape, 9-2
owner, 8-6
status,
PTY, 11-15
Terminal, 11-12
structure,
addressing a, 7-2
names, 7-2, 7-4, 8-3
modifying, 8-4
referring to, 7-2
Filler class codes, 11-7
Foreground jobs, 4-3
Format,
DECtape, 9-2, 9-5
Directory, 9-2, 9-3
Disk files, 8-5
Mag tape, 10-1

FORM Switch, 11-7
FSA, 7-6

Generation files, 8-1
Generic device names, 7-2, 7-4
GETLIN Monitor Call, 11-4
GETPPN Monitor Call, 4-9
GETSEG Monitor Call, 4-1
GPPL, 2-5
GVPL, 2-5

Half-duplex terminals, 11-15
HALT Instruction, 4-1
Header block, 7-10, 13-1
HIBER monitor call, 4-3, 4-4, 11-15
Highest
relative case location, 3-1
Legal address, 3-2
Host computer time, 4-10

I/O
interrupt conditions, 5-6, 5-7
General information, Chapter 7
Non-blocking, 7-4
With DECtape, Chapter 9
With Disk, Chapter 7
With Mag tape, Chapter 10
With Terminals, Chapter 11
Image Mode,
Disk, 7-7
Character, 11-6
Mag tape, 10-2
Binary, 7-7, 10-2
Dump, 7-7
INBUF Operator, 13-1
Industry compatible mode, 10-3
Initialization,
device, 7-2
job, 7-1
software interrupt system, 5-5
IN monitor call, 8-19, 11-16
INIT monitor call, 7-3
Input,
a character, 11-2
a file, 8-10
a line, 11-3
buffers, 11-3
INPUT Operator, 9-8, 8-19, 11-6, 11-16
Internal clocks, 4-10
Interrupt,
allow additional, 5-9

conditions, 5-6
control block, 5-7
definition of, 5-5
disable all, 5-9
dismiss additional, 5-9
granting an, 5-6
I/O, 5-6, 5-7
in progress bit, 5-9
non-I/O, 5-6
Inter-record gap, 10-1
IONDX. monitor call, 13-5
IOWD, 7-6
IPCF,
Address of Packet data block, 15-4
Association between PID and job no., 15-2
Awake on receival, 4-4
Capabilities of sender, 15-5
Default quotas, 15-1
Error codes, 15-12
Flags, 15-1
Indirect sender/receiver PID, 15-3
IPCFQ. monitor call, 15-10
IPCFR. monitor call, 15-11
IPCFS. monitor call, 15-11
Length of packet data block, 15-4
Long form message, 15-4
Name associated with PID, 15-2
Packet,
Data Block, 15-1, 15-5
Descriptor block, 15-1
Flags, 15-3
page, 15-3
privileged, 15-3
PID
definition, 15-1
of SYSTEM [INFO], 15-1
requesting a, 15-4
same as job number, 15-1
Receive packet quota, 15-1
Receiving a packet, 15-6, 15-11
Send packet quota, 15-1
Sender's project-programmer number, 15-5
Sending a packet, 15-5, 15-11
Short form message, 15-4
Status of queues, 15-10
SYSTEM [INFO], 15-1, 15-6, 15-7
SYSTEM [IPCC], 15-1, 15-7, 15-8

.JBOPC, 4-2
Job Data Area, Chapter 3
JOBDAT, Chapter 3
Job,
Activate a dormant, 4-3
Amount of core for, 4-5
Awake when I/O complete, 4-4
Background, 4-3
Control, 4-1
Initialization, 7-1
Information, 4-4
Largest size of, 4-5
Maximum number of, 4-7
Name of, 4-5
Number, 11-17
Obtain,
Number of, 4-9
PPN of, 4-9
Search list, 8-4
Set parameters for, 4-5
Status bits, 11-17
Stop, 4-3
Suspend, 4-2
Swap, 4-4
Time limit for, 4-6
JOBSTS monitor call, 11-16

KSYS word, 4-6

Label processing, 10-9
Length,
logical disk block, 8-1
of files, 8-1
LIGHTS monitor call, 4-11
Line,
Characteristics, 11-2
Numbers, 11-4
Loading,
Core image file, 2-1
Relocatable binaries, 2-1
User programs, 2-1
LOCATE monitor call, 4-8
Logical,
Device names, 7-2
Disk unit, 8-1
End of tape, 10-1
Node, 4-8
Unit names, 7-5
LOGOUT Monitor Call, 4-2
LOOKUP Monitor Call, 8-10, 9-5
Lower case terminals, 11-6
.LOW files, 2-1
LPT spooling, 4-6

Magnetic tape,
Block size, 10-1, 10-8
Buffer size, 10-1, 10-2
Change
Block size, 10-1
Density, 10-1
Characteristics, 10-1
Closing a, 10-1
Controller names, 10-1
Data modes, 10-1, 10-2
Double tape mark, 10-1
Error status, 10-9
File status, 10-12
Format of, 10-1
Logical end of, 10-1
Parity for, 10-1
Unit name, 10-1
Unit status word, 10-8
Master file directory, 8-1
MBOX, 4-10
Modes,
ALT, 11-8
APL, 11-9
ASCII, 7-7, 10-2
ASCII Line, 10-2
Binary, 7-7
DECtape, 9-1
Disk, 7-6
Magnetic tape, 10-2
MPX devices, 13-2
Terminals, 11-1
MOUNT Command, 7-2
MPPL, 2-5
MPX channel,
Connect device to, 13-2
Data modes, 13-2
Disconnect from, 13-2
Status bits, 13-4
MSTIME monitor call, 4-11
MTAID. monitor call, 10-12
MTAPE. monitor call, 10-3
MTCHR. monitor call, 10-4
Multiple file directory, 7-2

Name,
Abbreviation, 7-5, 7-6
Controller, 7-5, 10-1, 11-1
DECtape, 9-1
Device, 7-4
Extension, 8-8
File, 8-8
File structure, 7-5, 8-4
Logical, 7-2, 7-4, 7-5
Physical, 7-2, 7-4
Set program, 4-4
Node, 4-8
Non-directory devices, 7-2
Non-blocking I/O, 7-4
Non-I/O Interrupts, 5-6, 5-8
Normal mode operation, 11-6

OPEN monitor call, 7-3, 7-4
OTHUSR monitor call, 4-10
OUT Operator, 13-1
OUTBUF Operator, 13-1
Output,
ASCIZ string, 11-2, 11-6
Buffer, 11-3, 11-6
Character, 11-2
OUTPUT monitor call, 9-8, 11-16
OUT monitor call, 11-6

Packed image mode, 11-12
Page,
Definition, 2-3
Display mode, 11-7
Counter value, 11-7
Fault rate, 4-7
Fault handler, 3-3
Passive search list, 8-4
PC, 3-1, 4-2
Physical,
DECtape names, 9-1
Device names, 7-2
Disk units, 8-1
Page limits, 2-5, 4-8
Record separation, 10-1
Unit names, 7-5
PID,
association with job number, 15-2
definition, 15-1
indirect sender/receiver, 15-3
name associated with, 15-2
of [SYSTEM] INFO, 15-1
requesting a, 15-1
same as job number, 15-1
PJOB monitor call, 4-9
PIINI. monitor call, 5-5
PISYS. monitor call, 5-5, 5-9
PIRST. monitor call, 5-12
PIM, 11-12
PLT, 4-5
PRESERVE Command, 8-2
Project-programmer number, 4-9, 8-2, 8-9

Program,
Address space, 2-1
Continue, 4-1, 4-2
Counter, 3-3
Loading, 2-1
Set to run, 4-8
Start, 4-1
Stop, 4-1
Suspend, 11-15
Version number, 3-4
PROTECT Command, 8-7
Protection,
Against WAKEP, 4-3
Codes, 8-5 to 8-7
Memory, 2-1
Pseudo-TTYs, 11-14
PTP spooling, 4-5
PTY, 11-14, 11-15

R command, 4-1
.RB symbols, 8-13
Read,
At low threshold, 10-3, 10-8
Backwards, 10-8
Real-time jobs, 4-3
Receive speeds, 11-8, 11-9
Record,
Backspace, 10-3
Skip, 10-3
REEL I.D., 10-6, 10-8, 10-12
REENTER Command, 4-1
Remote status, 11-7
Release,
a device, 7-13
a terminal, 11-16
for DECtape, 9-8
Monitor call, 7-13
RENAME monitor call, 8-8, 9-8
RESET monitor call, 7-1
Retrieval information block, 8-5
Restricted devices, 7-2
RPA, 7-6
Rewind,
magnetic tape, 10-3, 10-7
RIB, 8-5
RUN command, 4-1
RUN monitor call, 4-1
RUN,
set program to, 4-8
time statistics, 4-9
RUNTIME monitor call, 4-9

SAT Blocks, 8-5
.SAV files, 2-1
Scan input buffer
terminal, 11-3
Search list for job, 8-4
Select,
a file, 8-10
SENSE. monitor call, 13-4, 13-5
Set,
block size, 10-1
date, 4-6
density, 10-1, 10-6, 10-7, 10-8
job information, 4-4
Job parameters, 4-5
program name, 4-4
program to run, 4-8
system parameters, 4-5
SETNAM monitor call, 4-4
SETUUO monitor call, 4-5
SFD,
Delete, 8-10
Description, 8-2
Sharable high segment, 2-4
.SHR files, 2-4
Single file directory devices, 7-2
Size,
Block, 8-1, 8-5, 10-1, 10-8
Buffer, 7-1, 9-1, 10-1, 11-1, 12-1
cluster, 8-5
largest size of job, 4-5
Skip,
a file, 10-3
a record, 10-3
backwards, 10-7
forwards, 10-7
SLEEP monitor call, 4-2
Spooling,
Bits, 4-6
set deferred, 4-8
Slave characteristics, 11-6
Software interrupt system,
Allow additional interrupts, 5-9
argument block flags, 5-10
control flags, 5-7, 5-9
Control the, 5-9
DEBRK. monitor call, 5-6, 5-12
definition of interrupt, 5-5
Disable all interrupts, 5-9
Dismiss additional interrupts, 5-9, 5-12
Example of, 5-12
I/O interrupts, 5-6, 5-7

initialization, 5-5, 5-8
Interrupt conditions, 5-6
Interrupt control block, 5-7
Interrupt in progress bit, 5-9
interrupt vector block, 5-9
non-I/O interrupts, 5-6, 5-8
new pc
old pc, 5-7
PIINI. monitor call, 5-5, 5-8
PISYS. monitor call, 5-9, 5-10, 5-11
PIRST. monitor call, 5-12
Save the Interrupt Blocks, 5-11
Turn system on, 5-5
START command, 4-1
Start
program execution, 4-1
STATES words (RH), 4-5
Status,
bits (OPEN), 7-4
for APR clock, 3-3
for parity bit, 10-8
magnetic tape device, 10-5
MPX device, 13-4, 13-5
Terminal, 11-16
Stop,
a program, 4-2
a job, 4-2
a job until event occurs, 4-3
temporarily, 4-2
Storage allocation tables (SAT), 8-5
Structure,
of disk files, 8-1
Organization of, 8-2
Sub-file directory, 8-1, 8-2
Suspend
execution of a job, 4-2
Swap,
job out immediately, 4-4
magnetic tape units, 10-7
SWAP.SYS, 8-5
SWITCH monitor call, 4-11
Switches
contents of data, 4-11
Symbol table,
length of, 3-2
pointer to
Synchronize on I/O error, 7-4
[SYSTEM] INFO,
description of, 15-6
Functions, 15-7
general discussion, 15-1
PID of, 15-1
[SYSTEM] IPCC,
Description of, 15-7
Functions, 15-8
general discussion, 15-1
System,
accounting file, 4-12
error log file, 10-4
set parameters for, 4-5
virtual memory limit, 4-7

TAB capabilities, 11-7
Tables,
storage allocation, 8-5
Tape mark,
description of on magnetic tape, 10-1
write a, 10-3
TAPOP. monitor call,
calling sequence, 10-6
error codes, 10-10
function, 10-6 to 10-10
TC10, 10-8
Terminal,
Buffer size, 11-1
clear buffers for, 11-6
Controller names, 11-1
data modes, 11-9
element for, 11-6
Input buffers, 11-6
line characteristics, 11-4
name of operator's, 4-6
operations, 11-1
SIXBIT physical name for, 11-14
Termination
Device, 7-3
TIDY word, 11-7
Timing information, 4-10
Time,
limit for job, 4-6
of day, 4-11
run statistics, 4-8
TIMER monitor call, 4-11
TM10, 10-8
Track status bit, 10-8
Transmit speed, 11-8, 11-9
Traps,
Set to zero, 4-2
TRMNO. monitor call, 11-5
TRMOP. monitor call,
error codes, 11-9
calling sequence, 11-5
TTCALL monitor call, 11-1
TTY GAG, 11-7

Two segment jobs, 2-1
TX01, 10-8

UFD,
  delete, 8-10
  description, 8-2
Unbuffered I/O, 7-6
UDX,
  general information concerning, 13-1
  return the, 13-5
Unit names,
  DECtape, 9-1
  disk, 7-6
  logical, 7-5
  mag tape, 10-1
  physical, 7-5
  terminals, 11-1
Undefined symbol table, 3-2
Universal date-time standard, 4-1
Universal device index,
  general information concerning, 13-1
  return the, 13-5
Unload
  magnetic tape, 10-3, 10-4
Unrestricted devices, 7-2
Use bit, 7-9
User file directory (UFD), 8-1
  entries in, 8-2
User program,
  Address space of, 2-1
  loading a, 2-1
  start address of, 3-3
User tape servicing,
  APRENB monitor call, 5-1
  APRENB flags, 5-2
USETI/USETO,
  For DECtape, 9-8
UGETF monitor call, 9-9
UTPCLR monitor call, 9-9
UU.PHS, 7-2

Vestigial job data area, Chapter 3
Virtual time traps, 4-8
Virtual memory, 2-4 to 2-6
  page fault rate, 4-7
  system limit, 4-7

WAKE monitor call, 4-3
Watch statistics, 4-6
Wakeup bit, 4-3, 4-4
Write,
  blank tape, 10-3
  dump file, 4-12
  lock bit, 10-8
  on the disk, 8-10
  protect the high segment, 2-1
  tape mark, 10-3, 10-7

READER'S COMMENTS

DECsystem-10 Monitor Calls
DEC-10-OMCMA-B-D

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? _____

What features are most useful? _____

What faults do you find with the manual? _____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

Would you please indicate any factual errors you have found. _____

Please describe your position. _____

Name _____ Organization _____

Street _____ Department _____

City _____ State _____ Zip or Country _____

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

READER'S COMMENTS

DECsystem-10 Monitor Calls
DEC-10-OMCMA-B-D

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? _____

What features are most useful? _____

What faults do you find with the manual? _____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

Would you please indicate any factual errors you have found. _____

Please describe your position. _____

Name _____ Organization _____

Street _____ Department _____

City _____ State _____ Zip or Country _____

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P. O. Box F
Maynard, Massachusetts 01754