TOPS–10
Monitor Calls Manual
Volume 2

AA–K039C–TB

April 1986

This manual describes the monitor calls used by TOPS–10 MACRO programmers to request services that are controlled by the TOPS–10 monitor. The *TOPS–10 Monitor Calls Manual* consists of two volumes. Volume 1 is an overview of the services available to the programmer. Volume 2 is a detailed list of the calls and coding sequences that are used to invoke these services.

This manual replaces the previous manual of the same name, order number AA–K039B–TB.

OPERATING SYSTEM:	TOPS–10 V7.03
SOFTWARE:	GALAXY V5.1

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

Northeast/Mid–Atlantic Region

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

Central Region

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

Western Region

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

digital equipment corporation • marlboro, massachusetts

First Printing, August 1980
Revised, December 1981
Revised, February 1984
Revised, April 1986

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

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

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

The following are trademarks of Digital Equipment Corporation:

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

digital

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

CONTENTS

PREFACE

This is the second volume of the 2-volume TOPS-1Ø Monitor Calls Manual. Volume 1 describes the facilities offered by the monitor for assembly language programs. You can use the information in Volume 1 to learn how to implement these facilities in your programs.

Volume 2 contains a detailed description of each monitor call, its calling sequence, functions, and error codes, if any. It is the definitive list of the monitor call functions. For information on using these calls, you should you read Volume 1 before attempting to use Volume 2.

Not all devices are supported under current versions of TOPS-1Ø. In the interest of providing useful information, this manual includes references to unsupported and obsolete hardware. For support status of hardware and software, please refer to the current TOPS-1Ø Software Product Description.

| Obsolete monitor calls are marked, either in the CALLI UUO, or in the
| chapter in which they were previously described. Appropriate
| substitutes, (if any), for obsolete calls are also indicated. Section
| 23.5 lists the GETTAB Tables, and notes any obsolete tables.

If you find any errors in this manual, please fill out the Reader's Comment Card found at the back of the manual and mail it to DIGITAL Equipment Corporation.

All reported errors will be corrected as soon as possible. These corrections are distributed automatically to notebook subscribers. Availability of updates to this manual will be announced in the Software Dispatch newsletter.

CHAPTER 22

MONITOR CALL DESCRIPTIONS

This chapter describes each of the TOPS-10 monitor calls. For each description the following information is included, if applicable:

- Function: briefly describes the general use of the call.
- Calling Sequence: shows the format for the call. Cases where a word may contain one of a number of types of information are indicated by the presence of braces ({ }) containing the options.
- Restrictions: describes any unusual conditions that might affect the operation of the call or its effect on the calling program.
- Normal Return: describes the result of a normal return from the call and any operational aspects with which you should be concerned.
- Error Return: describes the result of an error on return.
- Example: shows one or more examples of the call.
- Related Calls: lists other, related monitor calls.
- Common Errors: describes common user errors.

In the calling sequences shown, the following definitions apply throughout this section:

- **ac:** an arbitrary accumulator; often used for passing arguments to the call and to store an error code returned from a call.
- **return:** the statement to which control passes on return from a call.
- **normal return:** the statement to which control passes if no error occurs in executing a call.
- **error return:** the statement to which control passes if an error occurs in executing a call.

The monitor call names are defined in the file UUOSYM.MAC; the CALLI, MTAPE, and TTCALL monitor calls offer extensions through parameters passed to the monitor.

ACCLG. [CALLI 204]

Function

Used by the LOGIN system program to increment LOGNUM and ensure that LOGIN does not exceed the maximum number of logged-in jobs. The monitor performs the following functions for the ACCLG. call:

1. Increments LOGNUM (a word containing the number of logged-in jobs).

2. Checks the LOGNUM against the appropriate access maximum (LOGMAX for timesharing jobs or BATMAX for batch jobs).

Calling Sequence

```
        MOVSI   ac,(flags)
        ACCLG.  ac
          error return
        normal return
```

where: flags are:

Bit	Symbol	Meaning
Ø	AC.MAX	Check LOGMAX.
1	AC.BMX	Check BATMAX.
2	AC.DCR	Decrement LOGNUM count.

Normal Return

On a normal return, LOGNUM has been incremented and the maximum is not exceeded. If the LOGIN program is halted before the LOGIN UUO has successfully completed, however, the program should trap the CTRL/C exit and perform another ACCLG. call, setting the AC.DCR flag to decrement the LOGNUM count before allowing the program to exit.

Error Return

When this call takes the error return, one of the following error codes will be returned in the ac.

Error Codes

Code	Symbol	Error
1	ACLMX%	LOGMAX check failed. That is, to log in this job would exceed LOGMAX.
2	ACLBM%	BATMAX check failed.
3	ACLIL%	Invalid argument to ACCLG. call.
4	ACLJL%	ACCLG. with AC.DCR set produced an invalid value after decrementing.
5	ACLDC%	An ACCLG. with AC.DCR had been attempted when the LOGNUM had not been incremented.

ACCT. [CALLI 167]

Function

Reads or changes the account string for a job.

Calling Sequence

```
        MOVE      ac,[XWD fcn-code,addr]
        ACCT.     ac,
          error return
        normal return
        . . .
addr:   EXP       len
        . . .
        last argument
```

where: **fcn-code** is one of the function codes described below.

addr is the address of the argument block.

len is the length of the argument block (not including this word), and the words up through **last argument** are arguments for the given function.

The function codes and their meanings are:

Code	Symbol	Function
Ø	.ACTCH	Changes the account string for the specified job. The first word of the argument block must contain a 1 for the number of arguments; the second word must contain either a byte pointer to the ASCIZ account string or the word [-1,,address], where address is the location of the account string. In the latter case, account strings must be left-justified on a word boundary. You must have JACCT privileges to use function code Ø. Note that [1,2] privileges alone do not provide ability to perform this function.
1	.ACTRD	Reads the account string for the specified job. The first word of the argument block must contain a 2 for the number of arguments; the second word must contain the job number for the desired account string (-1 for the calling job); the third word must contain the address of where to return the account string.

NOTE

The maximum length for account strings is set by the system administrator when the monitor is generated by MONGEN (symbol name MLACTS). This default can be changed if your installation uses fewer than 39 characters in its account strings.

ACCT. [CALLI 167]

Normal Return

If the function was .ACTCH, the account string is changed. If the function was .ACTRD, the account string for the job is in the location pointed to by **addr+2**, and **addr+1** contains the job number.

Error Return

An error code is returned in the **ac**. The error codes and their meanings are:

Code	Symbol	Error
1	ACTTL%	Account string too long for your buffer; the leftmost characters have been stored.
2	ACTAC%	Address check error.
3	ACTIL%	Illegal argument.
4	ACTNJ%	Nonexistent job number.
5	ACTPS%	JACCT privileges required.

Example

```
          MOVE    T1,[XWD .ACTRD,ARGLST]
          ACCT.   T1,
            JRST  ERROR
          JRST    CONTIN
ARGLST:   EXP     2
JOBNO:    EXP     -1
ACCADR:   EXP     ACCSTR
ACCSTR:   BLOCK   ^D8
            .  .  .
ERROR:    error routine
CONTIN:   success routine
```

This code sequence places the ASCIZ account string for the calling job into the locations starting at ACCSTR.

Related Calls

GETPPN, PJOB

APRENB [CALLI 16]

Function

Enables trap servicing for a program. When a condition enabled for trap servicing occurs, control is transferred to the address given by .JBAPR in the job data area. See Chapter 6 for more information about handling traps.

Calling Sequence

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

where: flags is a halfword containing flag bits as follows:

Bit	Symbol	Trap Condition
18	AP.REN	Repetitive enable.
19	AP.POV	Pushdown list overflow.
21	AP.ABK	Reserved.
22	AP.ILM	Memory protection violation.
23	AP.NXM	Nonexistent memory.
24	AP.PAR	Memory parity error.
26	AP.CLK	Clock tick. The clock ticked while your program was actively running; this trap does not occur for every clock tick.
29	AP.FOV	Floating-point overflow.
32	AP.AOV	Arithmetic overflow.

When one of these conditions occurs while the processor is in user mode, the monitor:

1. Stores the PC in location .JBTPC in the Job Data Area. If the PC is equal to the first or second location in your trap servicing routine, the program is terminated.
2. Clears the arithmetic and floating-point overflow flags.
3. Transfers control to your trap-servicing routine; the location is given by the right half of location .JBAPR in the Job Data Area.

Your program must place the address of the trap-servicing routine into .JBAPR before executing the APRENB monitor call.

NOTES

- If your trap-servicing routine contains the instruction

  JRSTF @.JBTPC

  the processor bits are cleared and the state of the CPU is restored; control resumes where the interrupt occurred.

- The APRENB monitor call clears the trap after an occurrence of any selected condition; therefore your program must call APRENB after each trap occurs.

- To enable repeated trap interceptions, your program should set AP.REN (bit 18) when executing the APRENB call; however, clock interrupts must be reenabled after each trap occurs.

- If your program does not enable for traps, overflow conditions and clock ticks are ignored, but the other conditions listed above produce fatal errors.

Example

```
            MOVEI   T1,OVERFL       ;Get routine address
                                    ; to handle overflow
            MOVEM   T1,.JBAPR       ;Put into .JBAPR
            MOVEI   T1,AP.AOV       ;Arithmetic overflow flag
            APRENB  T1,             ;To .JBAPR on arith ovflw
            JRST    CONTIN          ;On to something else
OVERFL:     OUTSTR  [ASCIZ /ARITHMETIC OVERFLOW ERROR/]
            JRSTF   @.JBTPC         ;Resume execution
             .  .  .
CONTIN:                             ;Something else
```

This code sequence sets up an overflow message for the first arithmetic overflow; note that this example will not handle more than one arithmetic overflow.

Common Errors

Not reenabling the interrupt after each trap has occurred.

Failing to set up .JBAPR prior to the APRENB call.

Related Calls

UTRP., PISYS.

ATTACH [CALLI 1Ø4]

Function

Attaches a terminal line to a job. For example, this call is used by the BATCON program to attach and detach jobs from their terminals at system shutdown. This call is more powerful than the ATTACH monitor command.

An unprivileged job can use the ATTACH monitor call only if its terminal is in user mode, and it can only detach from its own controlling terminal.

Calling Sequence

```
MOVE    ac,[EXP <flag>+<lineno>B17+<jobno>B35]
ATTACH  ac,
  error return
normal return
```

where: flag is one of the bits described below.

lineno is a line number (restricted to 16 bits).

jobno is the number of a logged-in job (use -1 for the current job).

If **jobno** is -1, your job is detached from the current line and attached to the specified line; if jobno is Ø, the job attached to the line specified by lineno is detached; if jobno is positive (requires JACCT or [1,2] privileges), the monitor detaches the specified job from its current line and attaches it to the specified line.

The flags and their meanings are:

Bit	Symbol	Terminal Mode
Ø	AT.UMM	Puts terminal in monitor (command) mode. However, .STPGR of the SETUUO may force the terminal into user mode.
1	AT.UUM	Puts terminal in user mode.

If neither flag is set, the terminal mode is not changed. Note that this is the terminal mode, not the job mode.

To attach an arbitrary job to a terminal:

jobno should be the number of the job to be attached.

lineno should be the number of the terminal to which the job is to be attached.

flag is the mode to which the new terminal will be set.

The previous terminal will be left in monitor mode.

To attach your current job to a terminal, follow the above definitions, with the following exception:

jobno should be less than Ø (-1 is recommended).

ATTACH [CALLI 104]

To detach an arbitrary terminal:

jobno — must be zero.

lineno — should be the number of the terminal to be detached.

flags — will be ignored. The terminal will be left in monitor mode.

To detach your job's controlling terminal:

jobno — must be zero.

lineno — should be -1 or 777777. If you explicitly include 777777, the first two bits of the value are ignored, producing 177777 (both flag bits are off).

flags — are ignored. The terminal is placed in monitor mode.

Normal Return

The job is attached or detached as specified, and the terminal is in the specified mode.

Error Return

The ac is cleared. An error return occurs only if you use an illegal line or job number, or if you do not have the required privileges for the call.

Example

```
        MOVSI   T1,-1
        ATTACH  T1,
          JRST  ATTERR
        JRST    CONTIN
        . . .
ATTERR: error routine
CONTIN: success routine
```

This example detaches the current job from its terminal line; the mode is not changed.

CALLI [OPCODE Ø47]

Function

Passes the monitor a code for an extended set of monitor calls, called CALLIs. The negative CALLI codes, except CALLI -1, are reserved for customer-defined monitor calls. All non-negative codes and -1 are reserved for use by DIGITAL. Any CALLIs not listed in the following table are not supported or not implemented. Obsolete CALLIs. are marked as such, and some indicate an appropriate, supported monitor call to substitute for the obsolete call. Obsolete calls are not described in this manual.

The defined CALLIs also have symbolic names; in this chapter they are listed in alphabetical order by symbol name. For example, CALLI -1 is named LIGHTS, and is described in this chapter under LIGHTS.

The CALLIs and their symbolic names are listed in numerical order on the following pages.

CALLI [OPCODE 047]

Symbol	CALLI Function	Symbol	CALLI Function
LIGHTS	(Obsolete)	WHERE	[CALLI 63]
RESET	[CALLI 0]	DEVNAM	[CALLI 64]
DDTIN	(Obsolete; use TTCALL)	CTLJOB	[CALLI 65]
SETDDT	[CALLI 2]	GOBSTR	[CALLI 66]
DDTOUT	(Obsolete; use OUTSTR)	HPQ	[CALLI 71]
DEVCHR	[CALLI 4]	HIBER	[CALLI 72]
GETCHR	(Obsolete; use DEVCHR)	WAKE	[CALLI 73]
WAIT	[CALLI 10]	CHGPPN	[CALLI 74]
CORE	[CALLI 11]	SETUUO	[CALLI 75]
EXIT	[CALLI 12]	OTHUSR	[CALLI 77]
MONRT.	[CALLI 1,12]	CHKACC	[CALLI 100]
UTPCLR	[CALLI 13]	DEVSIZ	[CALLI 101]
DATE	[CALLI 14]	DAEMON	[CALLI 102]
LOGIN	[CALLI 15]	JOBPEK	[CALLI 103]
APRENB	[CALLI 16]	ATTACH	[CALLI 104]
LOGOUT	[CALLI 17]	DAEFIN	[CALLI 105]
SWITCH	(Obsolete)	FRCUUO	[CALLI 106]
REASSI	[CALLI 21]	DEVLNM	[CALLI 107]
TIMER	[CALLI 22]	PATH.	[CALLI 110]
MSTIME	[CALLI 23]	METER.	(Obsolete; use SNOOP.)
GETPPN	[CALLI 24]	MTCHR.	[CALLI 112]
TRPSET	[CALLI 25]	JBSET.	[CALLI 113]
TRPJEN	[CALLI 26]	POKE.	[CALLI 114]
RUNTIM	[CALLI 27]	TRMNO.	[CALLI 115]
PJOB	[CALLI 30]	TRMOP.	[CALLI 116]
SLEEP	[CALLI 31]	RESDV.	[CALLI 117]
SETPOV	(Obsolete; use APRNEB)	UNLOK.	[CALLI 120]
PEEK	[CALLI 33]	DISK.	[CALLI 121]
GETLIN	[CALLI 34]	DVRST.	[CALLI 122]
RUN	[CALLI 35]	DVURS.	[CALLI 123]
SETUWP	[CALLI 36]	XTTSK.	[CALLI 124]
REMAP	[CALLI 37]	CAL11.	[CALLI 125]
GETSEG	[CALLI 40]	MTAID.	[CALLI 126]
GETTAB	[CALLI 41]	IONDX.	[CALLI 127]
SPY	[CALLI 42]	CNECT.	[CALLI 130]
SETNAM	[CALLI 43]	MVHDR.	[CALLI 131]
TMPCOR	[CALLI 44]	ERLST.	[CALLI 132]
DSKCHR	[CALLI 45]	SENSE.	[CALLI 133]
SYSSTR	[CALLI 46]	CLRST.	[CALLI 134]
JOBSTR	[CALLI 47]	PIINI.	[CALLI 135]
STRUUO	[CALLI 50]	PISYS.	[CALLI 136]
SYSPHY	[CALLI 51]	DEBRK.	[CALLI 137]
DEVTYP	[CALLI 53]	PISAV.	[CALLI 140]
DEVSTS	[CALLI 54]	PIRST.	[CALLI 141]
DEVPPN	[CALLI 55]	IPCFR.	[CALLI 142]
SEEK	(Obsolete)	IPCFS.	[CALLI 143]
RTTRP	[CALLI 57]	IPCFQ.	[CALLI 144]
LOCK	[CALLI 60]	PAGE.	[CALLI 145]
JOBSTS	[CALLI 61]	SUSET.	[CALLI 146]
LOCATE	[CALLI 62]	SCHED.	[CALLI 150]

Symbol	CALLI Function
ENQ.	[CALLI 151]
DEQ.	[CALLI 152]
ENQC.	[CALLI 153]
TAPOP.	[CALLI 154]
FILOP.	[CALLI 155]
NODE.	[CALLI 157]
ERRPT.	[CALLI 160]
ALLOC.	[CALLI 161]
PERF.	[CALLI 162]
DIAG.	[CALLI 163]
DVPHY.	[CALLI 164]
GTNTN.	[CALLI 165]
GTXTN.	[CALLI 166]
ACCT.	[CALLI 167]
DTE.	[CALLI 170]
DEVOP.	[CALLI 171]
SPPRM.	[CALLI 172]
MERGE.	[CALLI 173]
UTRP.	[CALLI 174]
PIJBI.	[CALLI 175]
SNOOP.	[CALLI 176]
TSK.	[CALLI 177]
KDP.	[CALLI 200]
QUEUE.	[CALLI 201]
RECON.	[CALLI 202]
PITMR.	[CALLI 203]
ACCLG.	[CALLI 204]
NSP.	[CALLI 205]
NTMAN.	[CALLI 206]
DNET.	[CALLI 207]
SAVE.	[CALLI 210]
CMAND.	[CALLI 211]
PIBLK.	[CALLI 212]
SCS.	[CALLI 213]
SEBLK.	[CALLI 214]
CTX.	[CALLI 215]
PIFLG.	[CALLI 216]
IPCFM.	[CALLI 217]
LLMOP.	[CALLI 220]
LATOP.	[CALLI 221]
KNIBT.	[CALLI 222]
CHTRN.	[CALLI 223]
ETHNT.	[CALLI 224]
ENTVC.	[CALLI 225]
NETOP.	[CALLI 226]

CALl1. [CALLI 125]

Function

Performs front-end testing and debugging functions. Using this call, you can obtain information about PDP-11 based front end nodes, send and receive front-end messages, and examine and deposit into the front-end software.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        CALl1.  ac,
          error return
        normal return
        . . .
addr:   XWD port,fcn-code       ;.C1lFC
        address                 ;.C1lAD
        value                   ;.C1lCN
        qstart                  ;.C1lEN
```

where: len is the length of the argument block.

addr is the address of the argument block. Starting at this address, the call accepts up to four words, depending on the function code.

The argument block for CALl1. UUOs depends on the function specified in the right half of the function word (.C1lFC) at addr. The function word contains a port specification in the left half, and the function code in the right half.

Therefore, the format of the argument list for CALl1. is:

Offset	Symbol	Meaning
Ø	.C1lFC	Function word, containing the port specification and the function code. The left half of this word contains the type of port, in either of the following formats. The first format is old, and may be used by existing programs. However, the new format is recommended for new programs.

The old format for the left half is:

Bits	Symbol	Meaning
9-17	C1.CNO	Type of PDP-11 (see .C1lTY below).
9-14	C1.lNT	Type of port (see .C1lTY below).
14-17	C1.lNN	CPU number.

The new format for the left half of addr is as follows. Note that this format is signified by the setting of bit Ø, the sign bit.

Bits	Symbol	Meaning
Ø	C1.1NF	New format for port specification.
1-8	C1.1XX	Reserved for use by DIGITAL.
9-11	C1.1TY	Type code, one of the following:

Code	Symbol	Meaning
Ø	.C11DL	DL-1Ø
1	.C11DT	DTE-2Ø
2	.C11KD	KMC/DUP
3	.C11DR	DMR-11

11-14	C1.1CN	CPU number.
15-17	C1.1PN	Port number.

The right half of .C11FC (C1.1FC) contains the function code. The arguments following .C11FC depend on the function, so the argument lists are described for each function code listed below.

The function codes and their meanings are:

Code Symbol Function

Ø .C11DP Deposits the specified data in the specified location. The argument block for this function is:

```
addr:   XWD     port,.C11DP     ;.C11FC
        EXP     dpaddr          ;.C11AD
        EXP     value           ;.C11CN
```

where: dpaddr is the address to receive the value.

value is the data to be deposited.

This function requires the JP.POK privilege. This function works for DN6Ø and DN8x front ends only.

1 .C11EX Examines the specified location. The argument list for this function is:

```
addr:   XWD     port,.C11EX     ;.C11FC
        EXP     exaddr          ;.C11AD
```

where: exaddr is the address to be examined.

The data at the specified location is returned in the ac.

This function requires the JP.POK privilege. .C11EX works for DN6Ø and DN8x front ends only.

CALl1. [CALLI 125]

Code	Symbol	Function
2	.C11QU	Queues a request to the front end. The argument list for this function is: addr: XWD port,.C11QU ;.C11FC BLOCK 2 EXP qstart ;.C11EN where: qstart is the first word of a data block containing information regarding the front end function. This function requires the JP.POK privilege. .C11QU works only for DN6Ø front ends.
3	.C11NM	For DL1Ø based ANF-1Ø front ends, returns the name of the program running on the PDP-11. The SIXBIT program name is returned in ac. For all other front ends, .C11NM returns the name of the protocol enabled by the monitor for a given DTE. The argument list for this function is: addr: XWD port,.C11NM ;.C11FC
4	.C11UP	Returns the status of the PDP-11 (Ø for down, 1 for up). The status is returned in the ac. The argument list for this function is: addr: XWD port,.C11UP ;.C11FC
5	.C11SM	Sends a message. The argument list for this function is: addr: XWD port,.C11SM ;.C11FC This function works only for a DN8x front end.
6	.C11RM	Receives a message. The argument list for this function is: addr: XWD port,.C11RM ;.C11FC This function works only for a DN8x front end.
7	.C11TY	Returns the node type and node number. The argument list for this function is: addr: XWD port,.C11TY ;.C11FC The node number is returned in the left half of the ac, and the node type is returned in the right half, as one of the following type codes:

Code	Symbol	Meaning
1	.C1D76	DC76.
2	.C1D75	DC75/DN87.
3	.C1D6Ø	DN6Ø.
4	.C1D8S	DN87S.
5	.C1CFE	Console front end.
6	.C1MCB	DECnet-1Ø MCB front end.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	C11NP%	Job not privileged.
2	C11UF%	Unknown function.
3	C11ND%	Wrong type of PDP-11 specified.
4	C11IU%	Examine/deposit function already in use.
5	C11NA%	No answer to examine/deposit.
6	C11TS%	Queue entry too short.
7	C11NE%	Not enough arguments.
10	C11IA%	Invalid address specified for examine/deposit.
11	C11IQ%	Invalid argument for queue request function.
12	C11IC%	Insufficient core.
13	C11RP%	DTE-reload bit is set, or primary protocol is not running.
14	C11IE%	Insufficient exec virtual memory.
15	C11IL%	Illegal packet length.
16	C11NC%	CPU is not running.
17	C11IT%	Illegal type code specified.
20	C11IP%	Illegal port number specified.
21	C11DL%	No DL10 support in this monitor.
22	C11DT%	No DTE support in this monitor.
23	C11KD%	No KDP support in this monitor.
24	C11DR%	No DMR support in this monitor.

CHGPPN [CALLI 74]

Function

Changes the project-programmer number (PPN) for the current job. This call is reserved for the exclusive use of the LOGIN and INITIA programs.

Calling Sequence

```
MOVE    ac,[XWD projno,progno]
CHGPPN  ac,
  error return
normal return
```

where: projno,progno is the new project-programmer number (PPN).

Normal Return

The PPN for the current job is changed to the given number.

Error Return

Occurs if the calling job is already logged in, or if either the project or programmer number is zero. The ac is unchanged.

Example

```
MOVE   T1,[XWD 27,5Ø31]
CHGPPN T1,
  JRST ERROR
```

This code sequence changes the PPN for the current job to 27,5Ø31.

Related Calls

GETPPN, LOGIN

CHKACC [CALLI 100]

Function

Determines whether a file may be accessed, based on your job's current PPN and the file access protection code. Your programs should not make assumptions concerning the access codes associated with a file; they should use the CHKACC monitor call to determine if access is permitted to that file. This is especially true for privileged programs that are constrained by the access privileges of a non-privileged project-programmer number for which they are performing a task.

The CHKACC call does not function correctly on systems that are running a file daemon program, such as FILDAE. So, if your system is running a FILDAE type program, use the FILOP. call. The FILOP. monitor call allows a privileged program to specify that an operation is to be performed only when the operation would be legal if performed by a specified project-programmer number. In most cases, the FILOP. function eliminates the need for the CHKACC monitor call. New programs should be written using the FILOP. "in your behalf" capability (.FOPPN).

Calling Sequence

```
        MOVEI  ac,addr
        CHKACC ac,
          error return
        normal return
        . . .
addr:   XWD fcn-code,<ufdprot>B26+<filprot>B35
        XWD projno,progno      ;For file
        XWD projno,progno      ;For accessing program
```

where: **addr** is the address of the argument block.

fcn-code is one of the function codes described below.

ufdprot is a directory protection code.

filprot is a file protection code.

projno,progno is a project-programmer number (PPN).

NOTE

When your program specifies function codes 0 through 6, the monitor ignores the directory protection. When your program specifies function codes 7 and 10, the monitor ignores the file protection.

The function codes and their meanings are:

Code	Symbol	Access
0	.ACCPR	Checks whether your job can change the protection for the file.
1	.ACREN	Checks whether your job can rename the file.
2	.ACWRI	Checks whether your job can write the file.
3	.ACUPD	Checks whether your job can update the file (in old update mode).
4	.ACAPP	Checks whether your job can append to the file.
5	.ACRED	Checks whether your job can read the file.
6	.ACEXO	Checks whether your job can execute the file.
7	.ACCRE	Checks whether your job can create the file in the user's UFD.
10	.ACSRC	Checks whether your job can read the directory as a file.

The right to access a file is determined by:

- The type of access desired.
- The project-programmer number of the user desiring access to the file.
- The project-programmer number of the directory containing the file.
- 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 your program is going to perform a LOOKUP.

The owner of a UFD or an SFD can always read the UFD or SFD as a directory.

Normal Return

The monitor returns 0 in the ac if access to the file is allowed, or -1 if access is not allowed.

Error Return

The ac is unchanged; this occurs only if you gave an invalid function code or CHKACC is not implemented on your system.

Example

The following code checks to see if the user logged in as [11,315] can change a file with protection <055> in the directory area [27,5031].

```
        MOVEI   T1,ARGLST
        CHKACC  T1,
          JRST  ERROR
        JRST    CONTIN
ARGLST: XWD     .ACCPR,<775>B26+<055>B35
        XWD     27,5031         ;For files
        XWD     11,315          ;For accessing program
```

Related Calls

FILOP.

Common Errors

Assuming that the CHKACC call grants access to a file. Remember that it only tests the accessibility of the file. FILDAE can still deny access to the file on a LOOKUP, ENTER, RENAME, or FILOP. call. The File Daemon program is described in Appendix C.

CHTRN. [CALLI 223]

Function

CHTRN. is used to translate characters from one representation to another. For instance, CHTRN. may be used to convert 8-bit characters to 7-bit characters.

Calling Sequence

```
        XMOVEI  ac,addr
        CHTRN. ac,
          error return
        normal return

addr:   XWD     flags, source count
        EXP     source byte pointer (word one)
        EXP     source byte pointer (word two)
        XWD     reserved, destination count
        EXP     destination byte pointer (word one)
        EXP     destination byte pointer (word two)
```

where: flags are one or more of the flags listed below.

source count is the number of bytes stored where the source byte pointer indicates.

destination count is the number of bytes available at the location the destination byte pointer indicates.

The flag bits are:

Flag	Symbol	Meaning
Ø	CH.FBR	Fallback representation (translates 8-bit to 7-bit).
1	CH.OVR	Includes overprinting in the fallback representation.
2	CH.RAI	Changes lower case to upper case.
3	CH.6BT	Converts ASCII characters to SIXBIT.
4	CH.IGN	Ignores extra bits; does not range-check characters.
5	CH.ESC	Maps 7-bits ESCape sequences to 8-bit wherever possible.
6	CH.X6B	Expands SIXBIT source to ASCII destination.

Normal Return

The ac is unchanged. The byte pointers returned will have any indirection and indexing resolved. If you specify a one-word global byte pointer, the pointers will be expanded from one-word global format to two-word global format.

Error Return

One of the following codes is returned in the ac:

Code	Symbol	Error
1	CHADC%	Address check while reading or writing arguments.
2	CHBYP%	Illegal byte pointer.
3	CHINV%	Unknown or reserved flag bit specified.
4	CHILC%	Illegal character encountered during translation.
5	CHDCE%	Destination count exhausted prematurely.
6	CHIBC%	Invalid bit combination specified.

CLOSE [OPCODE Ø7Ø]

Function

Terminates transmission of data to or from a file. Closes the file for both input and output. The default functions of the CLOSE call for unbuffered data modes are:

- The output side of the channel is closed. In unbuffered data modes, the effect is to execute a device-dependent function.
- The input side of the channel is closed. The end-of-file flag is cleared. Further actions depend on the data mode. The effect is to execute a device-dependent function.

In buffered data modes, the following operations are performed on the output side of the channel:

- All data in the buffers that have not been transmitted to the the device is written to the device.
- Device-dependent functions are performed.
- The ring use bit is set to 1, indicating that the ring is not in use.
- The buffer byte count, the third word of the buffer header, is set to Ø.
- Control returns to the user program when transmission is complete.

In buffered data modes, if a ring buffer exists, the following operations are performed to close the input side of the channel:

- The monitor waits until the device is inactive.
- The use bit of each buffer is cleared, to indicate that the buffer is empty.
- The use bit of the buffer ring is set to 1, to indicate that the ring is not in use.
- The buffer byte count is set to Ø.
- Control returns to the user program.

If a file is being written to disk at the time of the output CLOSE, the unwritten blocks at the end of the disk file are deallocated. On input CLOSE, the access date of a disk file is updated if any data was actually read. (LOOKUP followed by CLOSE does not change the access date.)

If the file is being output to the card punch, the last card is punched, followed by an end-of-file card. This end-of-file card and the header card contain the file identification punch in column 1, which is ignored by the card reader service routine.

If a file is being output to magtape, two EOF marks are written and the tape position is backspaced over one EOF.

If a file is being output to the line printer, a form-feed character is appended to the last block of data.

Calling Sequence

```
CLOSE     channel,flags
return
```

where: **channel** is the channel number for the file.

flags are one or more of the function flags described below.

The function flags and their meanings are:

Bits	Symbol	Function
29	CL.DAT	Deletes the name block and access tables from the disk data base and the space is returned to monitor free core. For example, this function is used by BACKUP on a RESTORE operation.
3Ø	CL.RST	Inhibits deletion of the original file, if any, for an ENTER call that creates or supersedes the file. The new copy of the file is discarded.
31	CL.NMB	Inhibits deletion of the name block and access tables in monitor memory; this function is effective only if a LOOKUP call was executed for the channel, but no subsequent INPUT call for the channel was executed.
32	CL.ACS	Prevents updating of the file access date. For example, this feature is used by BACKUP, to save files on magtape without changing their access dates.
33	CL.DLL	Inhibits deallocation of any unwritten blocks at the end of a disk file.
34	CL.IN	Inhibits closing of the input side of the channel.
35	CL.OUT	Inhibits closing of the output side of the channel.

Return

The function is performed.

Example

See Chapter 11, Monitor Calls Manual Vol. 1.

Related Calls

LOOKUP, ENTER, RENAME

CLRBFI [TTCALL 11,]

Function

Clears text from the terminal input buffer. This call is often used to clear any further user commands when an error occurs; otherwise, incorrect processing (due to user-type ahead) could follow the error.

Calling Sequence

```
CLRBFI
return
```

Return

All text is cleared from the input buffer.

Related Calls

TTCALLs, TRMOP.

CLRBFO [TTCALL 12,]

Function

Clears the terminal output buffer. This monitor call is normally equivalent to typing CTRL/O.

Calling Sequence

```
CLRBFO
return
```

Return

The terminal output buffer is cleared.

Related Calls

CLRBFI, TTCALLs, TRMOP.

CLRST. [CALLI 134]

Function

Clears or sets the I/O status bits for a device. This enables your program to continue after an I/O error has occurred. The CLRST. UUO functions like SETSTS, taking the list of devices and I/O status bits for each device, with the additional ability to specify devices not explicitly OPENed on an I/O channel.

You can examine the current setting of the I/O status bits by using the SENSE. monitor call.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        CLRST.  ac,
          error return
        normal return
        . . .
addr:  {SIXBIT/device/}
       {EXP     channo}        ;.CLRSX
       {EXP     udx   }
        XWD     Ø,setsts-value ;.CLRST
       {SIXBIT/device/}
       {EXP     channo}        ;.CLRSX
       {EXP     udx   }
        XWD     Ø,setsts-value ;.CLRST
        . . .
```

where: **len** is the length of the argument list.

addr is the address of the argument list, containing one or more 2-word entries.

device is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel.

udx is the universal device index for a device.

setsts-value is the halfword value of the I/O status bits for the given device, channel, or udx; and it specifies the new settings for the I/O status bits.

Your program can clear the I/O status bits for more than one device. The argument block contains a 2-word entry for each device.

For a complete list of I/O status bits, see Chapter 11. Each type of device has a unique set of I/O status bits, which are described in the chapter about the appropriate device.

Normal Return

The I/O status bits for each specified device are cleared.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	CLRID%	Illegal device specified.
2	CLRNO%	Specified device does not belong to your job.

Example

```
        MOVE    T1,[XWD <CONTIN-ARGLST>,ARGLST]
        CLRST.  T1,
          JRST ERROR
        JRST    CONTIN
ARGLST: SIXBIT  /DTAØ/
        EXP     Ø
        EXP     CHANNO
        EXP     Ø
          . . .
CONTIN:
```

This code sequence clears the I/O status bits for DTAØ and the device associated with the channel whose number is the value of CHANNO.

Related Calls

ERLST., GETSTS, SENSE., SETSTS

CMAND. [CALLI 211]

Function

Defines commands that run specified programs, and manipulates the job's user-defined command list. In the argument list to this call, your program defines a command name that, when typed as a monitor command, will run the program specified by the file specification that is also included in the command list. The CMAND. UUO allows you to define multiple command names in the argument list, and allows you to read the command list that is already defined for your job.

Calling Sequence

```
MOVE    ac,[XWD fcn-code,addr]
CMAND.  ac,
  error return
normal return
```

where: **fcn-code** is the function code. The function codes are listed below.

addr is the address of the argument list. The argument list for each function code is described in the following list of function codes.

The function codes and their meanings are:

Code	Symbol	Meaning
Ø	.CMINT	Initializes (clears) any current command definitions and creates a new command list as specified at addr. The argument list stored at addr is formatted as follows:

```
addr:  flag,,length          ;.CMFLA
       command name          ;.CMNAM
       device name           ;.CMDVC
       file name             ;.CMFLE
       file extension        ;.CMEXT
       PPN                   ;.CMPPN
       first SFD             ;.CMSFD
       second SFD
                . . .
       Ø                     ;end of command list
```

Where the flags to be stored in the left half of .CMFLG indicates the number of characters in the command that must be input to define the command uniquely. The flags are:

Mask	Symbol	Meaning
1ØB17	CM.UN1	Command is uniquely identified by the first character of its name.
4B17	CM.UN2	Command is uniquely identified by the first two characters.

Code	Symbol	Meaning

Mask	Symbol	Meaning
2B17	CM.UN3	Command is uniquely identified by the first three characters.
1B17	CM.UN4	Command is uniquely identified by the first four characters.
1B12	CM.AUT	Command is defined as automatically saving the job's current context. The command will create a new context, in which the called program will run. The original context is restored when the program terminates.

And where the length is the number of words in the command block, stored in the right half of .CMFLA, and symbolized by CM.COU.

The rest of the command block contains the command name (in SIXBIT), and the file specification of the program to be run when the command is used. Note that the command block is of variable length, depending on the SFD level of the directory where the file is stored.

You can define more than one command by including a command block for each command, and storing them in contiguous blocks. The last word, where the next .CMFLA might be expected, must be set to zero.

1 .CMADD — Adds one or more command definitions to the current command list. The argument block for this function is identical to that used by function Ø (.CMINT).

2 .CMDEL — Deletes one or more commands from the current list of defined commands. The argument list for this function is:

```
addr:  length                ;.CMSIZ
       first command         ;.CMCMN
                .
                .
                .
```

The length of the argument list is equal to the entire length of the argument list, including .CMSIZ. The commands to be deleted are listed in the following words, and each must be equivalent to the .CMNAM word where the command was defined (see .CMINT argument list). Note that commands in the command list that are not listed in the .CMDEL argument list are not affected by this function.

Code	Symbol	Meaning
3	.CMLST	Lists all the currently defined command names. The argument list for this function is: addr: length BLOCK length-1 where length is the length of the argument block, and length-1 is the number of commands to return. On a successful return, the argument block appears as: addr: length ;.CMSIZ first command ;.CMNAM . . . Where .CMSIZ contains the total number of defined commands. The command names are returned starting at .CMNAM. If the reserved block is not long enough, the list of command names is limited to the reserved space.
4	.CMRET	Returns information about a command. You must include the argument list as: addr: length ;.CMSIZ command name ;.CMCNM In this argument list, specify the length of the block to be returned in .CMSIZ, and the name of the defined command for which information is desired, in .CMCNM. The information is returned in the form of a command block (same as argument list for .CMINT), for the command name.
5	.CMDMP	Dumps the command data base. This function uses the following argument list: addr: length BLOCK length-1 After the call returns successfully, a list of all the command blocks for defined commands will be returned at addr. See function Ø (.CMINT) for the format of the returned command blocks. Note that the last command block will be followed by a zero word to indicate the end of the command list.

Normal Return

The state of a return from CMAND. UUO is described for each function listed above.

Error Return

On an error return, the CMAND. UUO takes the non-skip return and returns the appropriate code from the following list of error codes:

Code	Symbol	Error
1	CMIAL%	Your program specified an illegal argument list. The argument list length was either too long or too short.
2	CMADC%	Address check occurred.
3	CMNER%	Not enough room to define commands in your job's per-process space.
4	CMDNF%	Your program did not finish reading the command list. The buffer size you allowed at addr was not enough to contain all the information to be returned.
5	CMNSN%	No such command name. On a .CMRET or .CMDEL function, you specified a command that is not defined.

Example

```
        MOVE AC,[XWD .CMADD,CMBLK]
        CMAND. AC,
         error return
        success          ;Command has been defined

CMBLK:  CM.UN3!6         ;/UNIQUE:3, and 6 words in block
        SIXBIT /XDDT/    ;Command name
        SIXBIT /DSKA/    ;Device name
        SIXBIT /DDT/     ;File name
        EXP Ø            ;Extension. Might as well leave zero
        XWD 1,4          ;PPN
```

This will define the XDDT command to run DSKA:DDT[1,4].

Common Errors

Assuming that .CMFLA in .CMINT or .CMADD specifies the length of the entire argument list.

CNECT. [CALLI 130]

Function

Connects or disconnects a device associated with an MPX channel. You can use CNECT. only with devices that are MPX-controllable (specifically, terminals, pseudo-terminals, line printers, card readers, paper tape punches, and remote data terminals).

Calling Sequence

```
        MOVEI     ac,addr
        CNECT.    ac,
          error return
        normal return
        . . .
addr:   XWD       fcn-code,channel
       {SIXBIT/device/}
       {EXP     udx   }
```

where: addr is the address of the argument block.

fcn-code is one of the function codes described below.

channel is the number of an initialized MPX channel.

device is the SIXBIT physical, generic, or logical name of a device.

udx is the Universal Device Index for the device.

Your program must initialize an MPX channel for the device using an OPEN call, before using the CNECT. call to connect the device to an MPX channel. The device must be initialized and connected to the MPX channel before it can be used for any I/O.

The function codes and their meanings are:

Code	Symbol	Function
1	.CNCCN	Connects the device to an MPX channel.
2	.CNCDC	Equivalent to CLOSE and disconnect from MPX channel.
3	.CNCDR	Equivalent to RESET and disconnect from MPX channel.
4	.CNOFE	Determines output feasibility.

Normal Return

The specified device is connected, disconnected, reset, and/or closed, as appropriate for the given function code. For the .CNCCN function, the Universal Device Index for the device is returned in the ac.

For the .CNOFE function, two values are returned in the ac. The left half of the ac contains the user address of the current output buffer, or Ø if none. The right half of the ac contains the number of data requests for a network device (except terminals, which return a 1 if output is possible), Ø if there are no data requests for the network device, or -1 if the device is local. The number of data requests indicates the number of buffers that the remote device can accept before your job will block in output wait state.

Your program can perform an output UUO to the device if the left half of the ac contains Ø and the right half is non-zero.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	CNCNM%	MPX channel not initialized.
2	CNCUD%	Nonexistent device.
3	CNCCM%	Illegal device for MPX.
4	CNCNF%	Not enough memory for control blocks.
5	CNCNC%	Device not connected.
6	CNCNO%	Device illegal or not initialized.
7	CNCII%	Invalid Universal Device Index.
1Ø	CNCUF%	Invalid function code.
11	CNCDU%	Device is not available to your job.
12	CNCSD%	Device is spooled; not MPX-controllable.

Example

```
        MOVEI   T1,ARGLST
        CNECT.  T1,
          JRST ERROR
        JRST    CONTIN
ARGLST: XWD     .CNCDC,CHANNO
        SIXBIT  /TTY111/
```

This code sequence disconnects the device TTY111, which is associated with the MPX channel given by CHANNO, from an MPX channel.

CORE [CALLI 11]

Function

Allows your program to dynamically expand or contract its core allocation in either or both segments. Note that neither of the segments may be locked in core.

Calling Sequence

```
MOVE    ac,[XWD hiseg,lowseg]
CORE    ac,flag
  error return
normal return
```

where: **hiseg** is the highest relative address to be used in the program high segment.

lowseg is the highest relative address to be used in the program low segment.

flag is a flag bit (UU.PHY) to indicate that the core assignment applies to physical memory. To set this flag, set bit 19. When bit 19=Ø, virtual memory references are allowed. When bit 19=1, all memory references are accepted as physical addresses.

Note that if the CORE UUO is executed in a non-zero section, all core address arguments will be interpreted as section-relative values. That is, all references are assumed to be relative to the current section.

If **hiseg** = Ø, the core assignment for the high segment is left unchanged; if **lowseg** = Ø, the core assignment for the low segment is left unchanged.

If you give a non-zero hiseg that is less than 4ØØØØØ or the length of the low segment (whichever is greater), the high segment is eliminated. Doing this from the high segment causes an illegal memory reference.

If your program has no high segment, or if you give a CORE call that eliminates the high segment, you can create a new, non-sharable high segment by giving **hiseg** greater than 4ØØØØØ. You can make the new high segment sharable by doing the following:

- Giving it a .EXE extension.
- Writing it onto a storage device.
- Closing the file.
- Using the SSAVE monitor command, or the SAVE. UUO with the SS%SSH flag, to save the entire core image.
- Initializing the program with a GET, R, or RUN monitor command, or with a RUN, MERGE., or GETSEG monitor call.

If you use the CORE monitor call giving a value for lowseg that is less than or equal to .JBREL, the monitor removes any noncontiguous pages from your address space; these pages may include the page fault handler (PFH) or VMDDT. To avoid this, use the PAGE. monitor call to choose only the needed pages.

Before expanding core, you should compare the highest required address with the highest legal address (stored in .JBREL). The example below shows how to expand core only if necessary.

You can specify the beginning of your program's high segment by using the REMAP monitor call, the /NEWPAGE or /SET switches to LINK, or the TWOSEG pseudo-op to MACRO.

Normal Return

The ac contains the current virtual memory limit in 1K blocks. However, if the CORE monitor call is issued from a non-zero section, the virtual memory limit is not returned in the ac.

Error Return

The error return occurs if any of the following conditions occurs:

- You give hiseg < 400001, but you do not have write-access privileges.
- You give both **hiseg** and **lowseg** as zero. In this case, the number of free 1K blocks is returned in the ac.
- The sum of the requested new low segment and the previously existing high segment exceeds your allowed program size. Core assignment is not changed; the maximum allowed program size (in 1K blocks) is returned in the ac.
- The sum of the requested new low and high segments exceeds your allowed program size. Core assignment is not changed; the maximum allowed program size (in 1K blocks) is returned in the ac.
- You give a lowseg argument that would extend the low segment into the high segment.
- One or both segments are locked.

Example

```
        MOVE    T1,NEWSIZ       ;Set up for call
        PUSHJ   P,CHKCOR        ;Call for core
        JRST    CONTIN

;Subroutine to get core only if needed

CHKCOR: CAMG    T1,.JBREL##     ;Core size OK?
        POPJ    P,              ;Yes
        CORE    T1,             ;Get more core
          JRST  ERROR           ;To error routine
        POPJ    P,              ;Core increase OK
```

Related Calls

PAGE.

CTLJOB [CALLI 65]

Function

Obtains the number of the job that is controlling a specified subjob. The subjob must be attached to a pseudo-terminal.

Calling Sequence

```
MOVEI   ac,jobno
CTLJOB  ac,
  error return
normal return
```

where: jobno is the number of the controlled job, or -1 to specify your current job.

Normal Return

The number of the controlling job is returned in the **ac**. If the job given by jobno is not controlled by a pseudo-terminal (PTY), the number returned in the ac is -1.

Error Return

Occurs if the job number is illegal.

Example

```
MOVNI  T1,1
CTLJOB T1,
  JRST ERROR
```

This code sequence returns the number of the controlling job in T1.

Related Calls

PJOB

CTX. [CALLI 215]

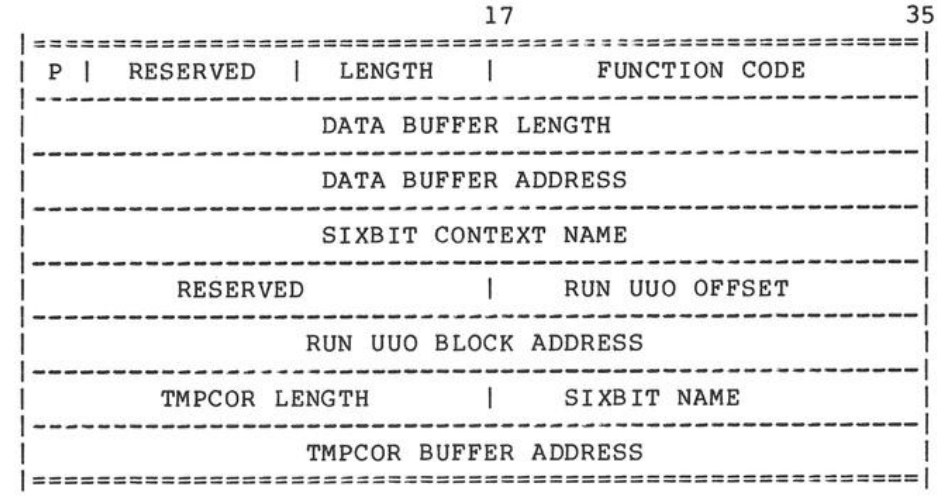
Function

CTX. allows you to manipulate contexts. (For a discussion of contexts, see Volume 1.) Since the argument block of CTX. is never written by the monitor, it may reside in a write-protected page or in a literal.

Calling Sequence

```
        XMOVEI ac,argblk
        CTX.   ac
          error return
        normal return
```

where: argblk is the following argument block whose maximum length is .CTMAX:



The format of the argument block is:

Offset	Symbol	Contents
Ø	.CTFNC	The function code word. It also contains one of the following flags, and the length of the argument block, in the following format:

Bits	Symbol	Meaning
Ø	CT.PHY	Physical-only RUN UUO.
2-8		Reserved for DIGITAL.
9-17	CT.LEN	Length of the argument block, including .CTFNC.
18-35	CT.FNC	One of the function codes listed below.

1	.CTDBL	Holds the data buffer length in words. 510 decimal words is the maximum.
2	.CTDBA	Contains the full 30-bit address of the data buffer. If the IFIW (sign bit) is on, a section local address, relative to the section CTX. is executed in, is referenced.

Offset	Symbol	Contents
3	.CTNAM	Used to hold a context name when creating a new context. When manipulating contexts, this word may contain a context name or context number.
4	.CTRNO	RUN UUO word. This holds the offset that would normally go into the left half of the RUN UUO ac (Ø for terminal input, or 1 for indirect command file input).
5	.CTRNB	Holds the 3Ø-bit block address that would ordinarily go into the right half of the RUN UUO ac.
6	.CTTMN	Contains the TMPCOR length in the left half (bits Ø-17), and its SIXBIT name in the right half (bits 18-35).
7	.CTTMB	The 3Ø-bit TMPCOR buffer address.

Valid function codes you can specify for .CTFNC are:

Code	Symbol	Function
Ø	.CTSVH	Saves the current context and halts the job. This has the effect of a PUSH command (see Operating System Commands Manual, Chapter 1). The context created is inferior. The inferior context is deleted as soon as you switch from it back to the superior one.
1	.CTSVR	Saves the current context, and runs a program. This is the equivalent of doing an auto-save, then a restore, at monitor level.
2	.CTSVT	Creates a parallel context by saving the current one, and creating a new top level context. The new context is different from one formed by a PUSH chain, as it is not inferior, nor is it associated with a chain of PUSHed contexts.
3	.CTSVS	Saves the current context, and switches to another, (already existing) one. For instance, you could use .CTSVR to create a new context running a program, and switch back to the previous context using .CTSVS. You could later return to the context created by .CTSVR, and restart the program in that context, without waiting for it to re-initialize.
4	.CTSVD	Switches to the specified context, deletes it, and returns to the previous (saved) context. A specific deletion is required for parallel contexts only.
5	.CTRDB	Reads the data buffer without changing the information. An inferior context uses this to read data when a superior context passes to it.
6	.CTWDB	Writes the data buffer. An inferior context writes data to its superior using this. Once data has been written, the old data in the superior context is lost.

Code	Symbol	Function
7	.CTRQT	Reads the context quota and saved-page quota for a job. The following offsets in the data buffer are required:

Offset	Symbol	Contents
Ø	.CTJOB	Job number, supplied by program.
1	.CTCTQ	Returned context quota.
2	.CTPGQ	Returned saved-pages quota.

Code	Symbol	Function
1Ø	CTSQT	Sets the context quota and saved-pages quota. The data buffer offsets are the same as for .CTRQT.
11	.CTDIR	Returns a directory map of all contexts. (GETTAB %CTBDM contains the byte pointer to the directory byte-stream.) The data buffer offsets returned are:

Offset	Symbol	Contents
Ø	.CTJOB	Target job number.
1	.CTWCT	Word count of byte-stream data.
2	.CTFDW	First data word of the directory byte-stream.

Code	Symbol	Function
12	.CTINF	Returns information about a particular context. The data buffer offsets returned are:

Offset	Symbol	Contents
Ø	.CTJOB	Target job number.
1	.CTCNO	Number of target context.
2	.CTCNM	Name of target context.
3	.CTSNO	Superior context's number.
4	.CTSNM	Superior context's name.
5	.CTPGM	Program running or saved in target context, if any.
6	.CTITM	Idle time (in clock ticks).

On any return, the ac contains:

Bits	Symbol	Meaning
Ø	CT.DAT	Set if data returned to the buffer.
1	CT.DBT	Returned if the buffer is truncated.
2	CT.ETX	Set if UUO error text in the buffer.
3	CT.RUN	Set for a RUN UUO error.
18-27	CT.RDL	Count of words returned in the buffer.
28-35	CT.ERR	CTX. or RUN UUO error code. Returned regardless of whether or not the data buffer contains error text.

CTX. [CALLI 215]

Normal Return

No errors will be flagged in CT.ETX, CT.RUN, or CT.ERR. Offsets (if any) from requested functions are returned in the data buffer.

Error Return

One of the following error codes is returned in bits 28 through 35 of the ac. If a data buffer is specified, error text is also returned.

Code	Symbol	Error
Ø	CXIFC%	Illegal function code.
1	CXACR%	Address check performed while reading arguments.
2	CXACS%	Address check performed while storing answers.
3	CXNEA%	Insufficient number of arguments.
4	CXNLI%	User not logged in.
5	CXLOK%	Program locked in core.
6	CXDET%	Job detached.
7	CXSCE%	System context quota exceeded.
1Ø	CXSPE%	System page quota exceeded.
11	CXJCE%	Job context quota exceeded.
12	CXJPE%	Job page quota exceeded.
13	CXNCS%	Insufficient core to save context.
14	CXNCD%	Not enough core to return data block.
15	CXICN%	Illegal context number.
16	CXNSC%	No superior context.
17	CXNPV%	No privileges to set quotas.
2Ø	CXIJN%	Illegal job number.
21	CXCSI%	Users cannot switch to an intermediate context.
22	CXCDI%	Users cannot delete an intermediate context.
23	CXCDC%	Users cannot delete the current context.
24	CXCNP%	Context not privileged.
25	CXNDA%	No data block is available.

DAEFIN [CALLI 105]

Function

Indicates that a request to the DAEMON program has been completed. This monitor call is reserved for the exclusive use of the DAEMON program.

If the specified job was in the DAEMON wait state, the monitor requeues the specified job to the run queue.

Calling Sequence

```
        MOVE   ac,[XWD length,addr]
        DAEFIN ac,
          error return
        normal return
        . . .
addr:   jobno
```

where: **length** is the length of the argument block.

addr is the address of the argument block.

jobno is the number of the logged-in job to be restarted.

Normal Return

The monitor leaves the ac unchanged, requeues the specified job, and clears the JDC bit in the job status word .GTSTS.

Error Return

The monitor clears the ac. This occurs if you are not privileged, if the job number is illegal or zero, or if the request could not be completed.

Example

```
        MOVE   T1, [XWD 1,ARGLST]
        DAEFIN T1,
          JRST ERROR
        JRST   CONTIN
ARGLST: EXP    JOBNO
```

Related Calls

DAEMON

DAEMON [CALLI 1Ø2]

Function

Invokes the system program DAEMON. When a job executes the DAEMON monitor call, the monitor puts the job into JD wait (sets the JDC bit in the GETTAB table .GTSTS) and wakes DAEMON. DAEMON examines the status word .GTSTS for each job in the system; for each job in the JD wait state, DAEMON performs the requested function. When the specified function has been completed, DAEMON issues a DAEFIN monitor call to make the job runnable.

Calling Sequence

```
        MOVE   ac,[XWD length,addr]
        DAEMON ac,
          error return
        normal return
        . . .
addr:   EXP    fcn-code
        . . .
        last argument
```

where: **length** is the length of the argument block.

addr is the address of the argument block, which contains **fcn-code** in the first word, followed by an argument list which differs for each function code. The argument lists are described below.

The function codes and their meanings are:

Code	Symbol	Function
1	.DCORE	Obsolete
2	.CLOCK	Enters a request in the clock queue to wake your job after a specified number of seconds has elapsed. As soon as the request has been entered in the queue, you should issue a call to HIBER with no time argument. An argument of zero clears the job's entry in the clock queue and wakes the job.

The argument list for the .CLOCK function is:

```
addr:   .CLOCK
        EXP     seconds
```

where: seconds is the number of seconds before the job DAEMON should wake. The preferred method for awakening the program after a short amount of time is by using the HIBER. call.

Code	Symbol	Function
3	.FACT	Makes an entry in the system accounting file FACT.SYS. Your job must have the JACCT privilege, or be logged in under [1,2]. Note that the USAGE accounting system should be used instead of the FACT file accounting system, as FACT is no longer supported. Refer to the QUEUE. UUO and the TOPS-10/20 Usage Accounting Manual for information about writing USAGE accounting entries. The system accounting file entry must require fewer than 24 words of memory. This entry may be stored (by DAEMON) for as long as 10 minutes before being written. DAEMON closes the current FACT file when its length reaches 2048 blocks, and opens a new FACT file. The argument list for the .FACT function is: addr: .FACT first word of entry . . . last word of entry Your job must have the JACCT privilege, or be logged in under [1,2]. Note that the USAGE accounting system should be used instead of the FACT file accounting system, as FACT is no longer supported. Refer to the QUEUE. UUO and the TOPS-10/20 Usage Accounting Manual for information about writing USAGE accounting entries.
4	.DMQUE	Reserved for use by DIGITAL.
5	.DMERR	Makes an entry in the error file; the third and following words of the argument block are written into the error file. Your job must have JACCT or [1,2] privileges. The argument block for the .DMERR function is: addr: .DMERR EXP ercode first word of message . . . last word of message where: ercode is a code to be entered in the error file.

Code	Symbol	Function

first word of message ... last word of message are words of data to be entered in the error file SYS:ERROR.SYS. The codes are:

Ercode	Symbol	Meaning
1	.ESWHY	Answer to ONCE's Why Reload question, and comment, if any.
2	.ESMSE	Continuable stopcode.
3	.ESMPE	KI memory parity error.
4	.ESNXM	KI non-existent memory error.
5	.ESCIN	Information extracted from a crash.
6	.ESCPE	Channel-detected memory parity error or non-existent memory.
7	.ESDRE	DAEMON restarted.
1Ø	.ESHDE	Hardware-detected device error.
11	.ESMDE	Massbus device error.
12	.ESDXE	DX2Ø device error.
14	.ESSWE	Software event. The events are:

Code	Symbol	Event
1	.SWEPK	POKE. function.
2	.SWESN	SNOOP. function.
3	.SWETP	TRPSET function.
4	.SWERT	RTTRP. function.
5	.SWMS1	Miscellaneous debugging event number 1.
6	.SWMS2	Miscellaneous debugging event number 2.

15	.ESCSE	Configuration status change. The condition change codes are listed below:

Code	Symbol	Status Change
Ø	.CSCAT	Attach function
1	.CSCDT	Detach function.
2	.CSCXC	Exchange function.
3	.CSCTC	Date/time change.
4	.CSCCF	DETACH CPU function.
5	.CSCCO	ATTACH CPU function.
6	.CSCNF	Node off-line.
7	.CSCNO	Node on-line.
1Ø	.CSCMO	Set memory on-line.
11	.CSCMF	Set memory off-line.

16	.ESSLM	System log message.
17	.ESDEB	Software requests data.
21	.ESTAP	Magnetic tape errors (see TAPSER).
3Ø	.ESKLE	KL processor error data from RSX-2ØF front end.
31	.ESFER	Front end reload.
33	.ESHSB	KS processor halt status block.
42	.ESTPS	Magnetic tape performance statistics code (see TAPSER).
43	.ESCFG	Maximum configuration in AVAIL.SYS.

Code	Symbol	Function

Ercode	Symbol	Meaning
44	.ESMRV	Monitor run values in AVAIL.SYS.
45	.ESDSC	Disk statistics (usually from a crash).
46	.ESBAV	Beginning of AVAIL.SYS time stamp.
47	.ESEAV	End of AVAIL.SYS time stamp.
50	.ESDLE	DL10 hardware error.
51	.ESKIP	KI parity/non-existent memory interrupt.
52	.ESKLP	KL parity/non-existent memory interrupt.
54	.ESKSN	KS non-existent memory trap.
55	.ESKPT	KL/KS parity trap.
56	.ESSNX	Non-existent memory scan.
57	.ESSPR	Parity memory scan.
61	.ESKDT	KL data parity trap.
62	.ESMOT	KL data parity interrupt.
63	.ESCSB	CPU status block.
64	.ESDSB	Device status block.
67	.ESKAE	KL addressing failure.
71	.ESLPT	Line printer error.
72	.ESHCC	Hard copy controller entry.
73	.ESULD	Microcode load.
100	.ESDTC	Date/time change (obsolete).
201	.ESNUS	Network utility started.
202	.ESNDL	Network down-line load.
203	.ESNUD	Network up-line dump.
210	.ESNHE	Network hardware error.
211	.ESNSE	Network software error.
220	.ESNOE	Network operator entry.
221	.ESNTC	Network topology change.
230	.ESNLC	Network line counter.
231	.ESNNS	Network node statistic entry.
377	.ESHIA	Hiatus in ERROR.SYS.
775	.ESOFF	Marker for first word of block as pointer to start of first entry.
777	.ESEOF	End-of-file flag.

.DMERR is a privileged function; to use it you must have the JACCT privilege, or be logged in under [1,2].

NOTE

For a complete description of the format of the error file, refer to the SPEAR Reference Manual.

6 .DMCTL Reserved for use by DIGITAL.

Normal Return

The monitor performs the specified function and issues a DAEFIN monitor call to make the job runnable. The ac is cleared.

DAEMON [CALLI 1Ø2]

Error Return

If DAEMON is not running, control returns to the error return, but the ac is unchanged.

If DAEMON is running, an error code is returned in the ac, and control returns to the error return. The error codes and their meanings are:

Code	Symbol	Error
1	DMILF%	Illegal function code.
2	DMACK%	Address check.
3	DMWNA%	Incorrect number of arguments.
4	DMSNH%	Impossible error. If this occurs, please report it to your Software Support Specialist.
5	DMCWF%	File cannot be written.
6	DMNPV%	Not enough privileges.
7	DMFFB%	Incorrect format for FACT file entry.
1Ø	DMPTH%	Invalid path.

Example

```
        MOVE   T1,[2,,ADDR]
        DAEMON T1,
          JRST ERROR
        JRST CONTIN
         . . .
ADDR:   .CLOCK
        EXP    5
```

This code puts the program to sleep for 5 seconds.

Related Calls

DAEFIN

DATE [CALLI 14]

Function

Returns a code giving the system date. The code is an integer given by the formula:

```
code = 31[12(year-1964)+(month-1)]+(day-1)
```

You can obtain the current day, month, and year using the formulas:

```
day = mod(code,31)+1
month = mod(code/31,12)+1
year = (code/372)+1964
```

The DATE call is equivalent to using GETTAB to obtain item %CNDAT. The day, month, and year are stored in GETTAB items %CNDAY, %CNMON, and %CNYER, respectively. Your program can avoid the computations needed to interpret the data returned from the DATE call by GETTABing the specific items, but the efficient program will avoid performing three separate GETTAB calls by GETTABing %CNDAT and then dividing the data into its appropriate components.

Calling Sequence

```
DATE   ac,
return
```

Example

The following macro computes the current day, month, and year.

```
DEFINE    CURDAT(DAY,MONTH,YEAR)<
   DATE   T1,
   IDIVI  T1,^D31
   ADDI   T2,1
   MOVEM  T2,DAY
   IDIVI  T1,^D12
   ADDI   T2,1
   MOVEM  T2,MONTH
   ADDI   T1,^D1964
   MOVEM  T1,YEAR
   >
```

Related Calls

TIMER

DEBRK. [CALLI 137]

Function

Dismisses a PSI software interrupt, reenabling any conditions disabled by the interrupt. See Chapter 6 for a discussion of the software interrupt system.

On a DEBRK. monitor call, the monitor scans the queue of pending interrupts, looking for conditions requiring service by an interrupt routine. If one is found, the interrupt occurs and control passes to the interrupt routine. If no such condition is found, DEBRK. restarts the interrupted process beginning at the point within your job where the interrupt occurred (usually the instruction after the last instruction that was executed).

Calling Sequence

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

Return

The DEBRK. call normally returns to old PC (see Chapter 6). **Return 1** is taken if DEBRK. is not implemented; **return 2** is taken if there is no interrupt in progress.

Related Calls

PIBLK., PIINI., PIRST., PISAV., PISYS.

DEQ. [CALLI 152]

Function

Dequeues one or more requests for enqueued resources, or relinquishes ownership of one or more enqueued resources. See Chapter 8 for a discussion of the ENQ/DEQ facility.

Calling Sequence

```
{MOVE    ac,[XWD .DEQDR,addr]       }
{MOVE    ac,[XWD .DEQDA,Ø]          }
{MOVE    ac,[XWD .DEQID,request-id] }
 DEQ.    ac,
   error return
 normal return
```

where: **function** is one of the function codes listed below.

addr is the address of an argument block, which is the same as the one used for the ENQ. call.

request-id is a request identifier that you specified in the ENQ. argument block.

For a discussion of these parameters and the format of each request block, refer to the ENQ. call.

The function codes and their meanings are:

Code	Symbol	Function
Ø	.DEQDR	This function dequeues a specific request. After a normal return, the monitor has removed the specified request from the specified queue, or the monitor has dissolved the lock between the job and the specified resource. The error return is taken if you set up the call in an incorrect format, or if you have no pending requests and you are not the owner of the specified resource. On an error return, the monitor returns an error code in the **ac**.
1	.DEQDA	This function removes all of your requests for ownership and dissolves all of your resource locks. The error return is taken if you write the call in an incorrect format, or if you do not have any pending requests or locks. On an error return, the monitor returns an error code in the ac. You should perform this function before EXITing; otherwise, when you perform a CLOSE, the function will fail but the nature of the failure will be difficult to determine. The monitor automatically performs the .DEQDA function when you issue a LOGOUT monitor call.

DEQ. [CALLI 152]

Code	Symbol	Function
2	.DEQID	This function requires the request-id in the right half of the ac. The monitor removes all requests of yours with the specified request-id from resource queues, and it dissolves all locks of yours with the specified request-id. You should specify this function when you are dequeueing requests that were made in the same ENQ. argument block. The error return is taken if you have set up the call incorrectly, if you have no pending requests, or if you are not the owner of a resource.

Normal Return

The specified requests are dequeued and the specified locks are dissolved.

Error Return

If an error is found in one of the requests in a multiple request DEQ. monitor call, the error return is taken and the monitor returns an error code in the ac. However, the ENQ/DEQ facility continues processing until all of the dequeue requests have been performed. Therefore, the monitor will have dequeued all valid requests whether or not an error resulted from another request in the same monitor call. If errors are found in several requests of the same monitor call, the error code returned in the ac reflects the last error found.

If you specify that you want to dequeue a request or dissolve a lock associated with a pooled resource, the monitor will return an error code if you attempt to dequeue more resources than you own within the pool. However, you can dequeue a subset of those resources that you own within a pool, still retaining ownership of those you did not dequeue. Therefore, you cannot dequeue more resources than you own, but you do not have to dequeue all that you own in one request.

The error codes for the DEQ. call are identical to those of the ENQ. call. They are listed in the description of the ENQ. call.

Example

DEQ. monitor calls that specify multiple requests are treated as multiple DEQ. monitor calls, each specifying a single request. This is not true for the ENQ. monitor call. For example:

```
            MOVE    T1 [XWD .DEQDR,DEQBLK]
            DEQ.    T1,
              JRST ERROR
            JRST    SUBR

DEQBLK:     2,,^D8
            0,,400000
            0,,2
            POINT 7,[ASCIZ/TEST/]
            ^D10,,1
            0,,4
            POINT 7,[ASCIZ/TESER/]
            ^D10,,1
```

The above code is, in effect, identical to the following, but the following is less efficient:

```
            MOVE    T1,[XWD .DEQDR,DEQ1]
            DEQ.    T1,
              JRST ERROR

DEQ:        MOVE    T1,[XWD .DEQDR,DEQ2]
            DEQ.    T1,
              JRST ERROR
            JRST    SUBR

DEQ1:       1,,^D5
            0,,400000
            0,,2
            POINT 7,[ASCIZ/TEST/]
            ^D10,,1

DEQ2:       1,,^D5
            0,,400000
            0,,4
            POINT 7,[ASCIZ/TESER/]
            ^D10,,1
```

Related Calls

ENQ., ENQC.

DEVCHR [CALLI 4]

Function

Returns the physical characteristics of a specified device.

Calling Sequence

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

where: **device** is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

Return

If the device is not found, or if your program has not initialized the device, the monitor clears the ac. Otherwise, the ac contains flags giving the physical characteristics of the device. The flags and their meanings are:

Bits	Symbol	Device or Mode
Ø	DV.DRI	DECtape whose directory is in memory; you can clear this bit by using the REASSI monitor call for the device.
1	DV.DSK	Disk.
2	DV.CDR	Card device. If DV.IN is set, it is a card reader; if DV.OUT is set, it is a card punch.
3	DV.LPT	Line printer.
4	DV.TTA	Terminal that is currently controlling a job.
5	DV.TTU	Terminal that is in use.
6	DV.2IO	Device can do input and output at the same time.
7	DV.DIS	Special display device. Note that this does not indicate the "display" terminal characteristic.
8	DV.LNG	Device with long dispatch table; this means that monitor calls other than INPUT, OUTPUT, CLOSE, and RELEAS can perform real functions.
9	DV.PTP	Papertape punch.
1Ø	DV.PTR	Papertape reader.
11	DV.DTA	DECtape.

Bits	Symbol	Device or Mode
12	DV.AVL	The device is available or is assigned to your job.
13	DV.MTA	Magnetic tape.
14	DV.TTY	Terminal.
15	DV.DIR	The device is a directory device. You can test this bit to determine whether ENTER/LOOKUP must be done before you can start I/O to the device.
16	DV.IN	Input device.
17	DV.OUT	Output device.
18	DV.ASC	The device has been initialized by the ASSIGN monitor command.
19	DV.ASP	The device has been assigned by the INIT, OPEN, or FILOP. monitor call.

Bits 20-35 specify the modes that are legal for the device.

20	DV.M17	Mode 17, dump. This is the same as IO.MOD = .IODMP returned from a GETSTS monitor call.
21	DV.M16	Mode 16, dump records. This is the same as IO.MOD = .IODPR returned from a GETSTS monitor call.
22	DV.M15	Mode 15, image dump. This is the same as IO.MOD = .IOIDP returned from a GETSTS monitor call.
23	DV.M14	Mode 14, binary. This is the same as IO.MOD = .IOBIN returned from a GETSTS monitor call.
24	DV.M13	Mode 13, image binary. This is the same as IO.MOD = .IOIBN returned from a GETSTS monitor call.
25	DV.M12	Mode 12, reserved for use by DIGITAL.
26	DV.M11	Mode 11, reserved for use by DIGITAL.
27	DV.M10	Mode 10, image. This is the same as IO.MOD = .IOIMG returned from a GETSTS monitor call.
28	DV.M7	Mode 7, reserved for use by customers.
29	DV.M6	Mode 6, reserved for use by customers.
30	DV.M5	Mode 5, reserved for use by DIGITAL.
31	DV.M4	Mode 4, reserved for use by DIGITAL.
32	DV.M3	Mode 3, byte. This is the same as IO.MOD = .IOBYT returned from a GETSTS monitor call.
33	DV.M2	Mode 2, packed image. This is the same as IO.MOD = .IOPIM returned from a GETSTS monitor call.

DEVCHR [CALLI 4]

Bits	Symbol	Device or Mode
34	DV.M1	Mode 1, ASCII line. This is the same as IO.MOD = .IOASL returned from a GETSTS monitor call.
35	DV.MØ	Mode Ø, ASCII. This is the same as IO.MOD = .IOASC returned from a GETSTS monitor call.

NOTE

To check for the NUL device, use DEVCHR to see if both DV.DSK and DV.TTY are set.

Example

```
        MOVE    T1,[SIXBIT/DEV/]
        DEVCHR  T1,
        TLNN    T1,(DV.DSK)
          JRST  NOTDSK
        JRST    ISDSK
```

This example checks to see if device DEV (assumed to be a logical name) is a disk. The call returns to NOTDSK if it is not and returns to ISDSK if it is.

Related Calls

DEVLNM

DEVLNM [CALLI 107]

Function

Assigns (or clears) a logical device name to a device.

Calling Sequence

```
{MOVE     ac,[SIXBIT/device/]}
{MOVEI    ac, channo          }
{MOVEI    ac,udx              }
 MOVE     ac+1,[SIXBIT/name/]
 DEVLNM   ac,
   error return
 normal return
```

where: **device** is the SIXBIT physical or logical name of a device to which you wish to assign a logical name.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

name is the logical name to be assigned to the device. If **name** is binary zero, any existing logical name assignment will be cleared.

Normal Return

The logical name is assigned to the device; the ac and ac+1 are unchanged.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
-3	DVLNA%	Device not assigned to your job.
-2	DVLIU%	Logical name already in use.
-1	DVLNX%	No such device or channel.

Related Calls

DEVCHR, DEVNAM, DEVOP., DEVPPN, DEVSIZ, DEVSTS, DEVTYP, REASSI

Common Errors

Assuming that DEVLNM also causes the device to become associated with your job. Use the REASSI call to actually obtain the device.

DEVNAM [CALLI 64]

Function

Returns the physical name of a device.

Calling Sequence

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

where: device is the logical device name whose physical name is desired.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

Normal Return

The SIXBIT physical name of the device is returned in the ac.

Error Return

If the specified device does not exist or if the specified channel is not initialized, the ac is cleared.

Related Calls

DEVCHR, DEVLNM, DEVOP., DEVPPN, DEVSIZ, DEVSTS, DEVTYP

DEVOP. [CALLI 171]

Function

Performs miscellaneous device functions for devices other than terminals, tapes, disks, or TSKs. Use TRMOP. for terminal functions, TAPOP. for tape functions, DISK. for disk functions, or TSK. for TSK functions.

Calling Sequence

```
        MOVE    ac,[XWD length,addr]
        DEVOP.  ac,
          error return
        normal return
        . . .
addr:   EXP     fcn-code
       {SIXBIT/device/ }
       {EXP     channo }
       {EXP     udx    }
addr+2: arglst
```

where: **length** is the length of the argument block.

addr is the address of the argument block.

fcn-code is one of the function codes described below.

device is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

arglst begins the list of arguments for the given function.

All function codes listed below use the 2-word argument list shown above. Additionally, some function codes accept a longer argument list. For those codes that accept an argument list longer than 2 words, the argument list format is shown with the description of the function code.

The function codes are defined within the following four ranges:

```
0000-0777 Performs a specific action.
1000-1777 Reads a parameter.
2000-2777 Sets a parameter.
3000-3777 Reserved for customer definition.
```

The Read/Set function codes are parallel (for example, function code 1002 reads a parameter and code 2002 sets the same parameter). The symbol .DFSET is equal to 1000, and can be added to the read parameter to establish the offset for the set parameter. Therefore, to read the page counter, use function .DFPCT. To set the page counter, use .DFPCT+.DFSET.

The monitor returns values in the ac for the Read functions.

The function codes and their meanings are:

Code	Symbol	Function
1	.DFLLV	Loads the standard vertical forms control unit.
2	.DFENV	Enables the system to load a non-standard vertical forms control unit.
3	.DFDVL	Disables loading non-standard vertical forms control unit.
4-1Ø		Reserved for use by Digital.
11	.DFLR2	Loads a translation RAM into LP2Ø. This function takes a 4-word argument list of the form: addr: .DFLR2 {device} {channo} {udx} 8-bit byte count for RAM address of RAM buffer
12	.DFLV2	Loads a VFU through LP2Ø. This function takes a 4-word argument list of the form: addr: .DFLV2 {device} {channo} {udx} 7-bit byte count of VFU address of VFU data
13	.DFMDC	Clears DVCMDA. This is the flag indicating whether the device is controlled by MDA (in GALAXY Version 4.1 and later). This function requires privileges.
14	.DFMDS	Sets DVCMDA. This is the flag indicating whether the device is controlled by MDA (in GALAXY Version 4.1 and later). This function requires privileges.
15-777		Reserved for use by Digital.
1ØØØ	.DFPCT	Returns the line printer's page counter in the ac.
2ØØØ		Sets the page counter value in addr+2. The page counter is limited to 12 bits. The argument list for .DFPCT is: addr: .DFPCT {SIXBIT/device/} {EXP channo} {EXP udx} EXP counter

Code	Symbol	Function
1002	.DFHCW	Reads the line printer characteristics. The printer characteristics are returned in the ac in the form:

Bits	Symbol	Meaning
0	DF.LCP	Lowercase capability.
1	DF.PGC	Has page counter.
2		Reserved.
3-5	DF.VFT	Code for type of vertical forms control unit (VFU). The type codes are:

Code	Symbol	Type
0	.DFVTO	Papertape VFU.
1	.DFVTD	DAVFU.
2	.DFVTN	No VFU.

6-8	DF.TYP	Code for character set codes. The set codes are:

Code	Symbol	Character set
0	.DFC64	Set of 64 characters.
1	.DFC95	Set of 95 characters.
2	.DFC28	Set of 128 characters.
3	.DFVAR	Variable size set.

9-11	DF.CLS	Code for line printer class. The class codes are:

Code	Symbol	Class
0	.DFSUK	Unknown.
1	.DFSBX	BA10.
2	.DFSLC	LP100.
3	.DFS20	LP20 (20F).
4	.DFSA1	LP11.
5	.DFSA2	LP20 (ANF DN8X).

12-14	DF.CLU	Line printer class, as the type of unit. The unit codes are:

Code	Symbol	Type
0	.DFUUK	Unknown.
1	.DFUFG	LP05-type.
2	.DFULN	LN01-type.

18-35	DF.CSN	Character set name, in SIXBIT.

Code	Symbol	Function
2002		Sets the line printer characteristics. The argument list for .DFHCW is: addr: .DFHCW {SIXBIT/device/} {EXP channo } {EXP udx } EXP characteristics Defines the characteristics using the definitions listed above for the Read function.
1003	.DFRES	The extended I/O error status for the given device is returned in the ac. The error status is returned as a code. The error codes are:

Code	Symbol	Error
1	IOPLE%	Page limit exceeded (LPT).
2	IOVFE%	VFU format error (LPT).
3	IOLTE%	Label type error (MTA).
4	IOHLE%	Header label error (MTA).
5	IOTLE%	Trailer label error (MTA).
6	IOVLE%	Volume label error (MTA).
7	IODER%	Hard device error.
10	IOPAR%	Parity error.
11	IOWLE%	Write-lock error.
12	IOIPO%	Illegal positioning operation (MTA).
13	IOBOT%	Beginning of tape (MTA).
14	IOIOP%	Illegal operation (MTA).
15	IOFNF%	File not found (MTA).
16	IOCAN%	Operator cancelled volume switch request (MTA).
17	IOTMV%	Too many volumes in volume set (MTA).
20	IONND%	Network node down.
21	IOUNC%	Undefined character interrupt (LP20).
22	IORPE%	RAM parity error (LP20).
23	IOLRA%	Tape labelling request was aborted by a RESET UUO (MTA).
24	IOVPF%	Volume protection failure (MTA).
25	IOFPF%	File protection failure (MTA).
26	IOUEF%	Unexpired file (MTA).
27	IONDD%	Network device disconnected.

Code	Symbol	Function
1004	.DFRDS	A status code for the specified device is returned in the ac.

The status codes and their meanings are:

Bit	Symbol	Status
0	DF.OFL	Device off-line.
34	DF.LLE	DAVFU load-enabled.
35	DF.LVE	A VFU error occurred.

The bits returned in the left half of the ac are device-independent; the bits returned in the right half are device-specific.

Normal Return

The specified function is executed.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
-1	DFACS%	Address check.
0	DFIFC%	Illegal function code.
1	DFPRV%	Not enough privileges.
2	DFIFD%	Function invalid for device.
3	DFNLR%	Value out of range.
4	DFNXD%	Nonexistent device.
5	DFNDV%	No DAVFU (LPT only).
6	DFNIA%	Device not initialized.
7	DFDOL%	Device off-line.
10	DFCNS%	Page counter not set (LPT only).
11	DFNPC%	No page counter (LPT only).
12	DFENI%	Extended error recovery not implemented.
13	DFNVC%	Non-variable character set.

If the monitor call has not been implemented on your system, the error return is taken and the monitor leaves the ac unchanged.

Related Calls

DEVCHR, DEVLNM, DEVNAM, DEVPPN, DEVSIZ, DEVSTS, DEVTYP

DEVPPN [CALLI 55]

Function

Returns the project-programmer number (PPN) associated with a disk device or an ersatz device. Note that the DEVPPN UUO does not return SFD names. It is recommended that programs use the PATH. call to return complete directory names.

Calling Sequence

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

where: **device** is the SIXBIT physical, logical, or ersatz name of a disk device.

channo is a channel number for a disk device.

udx is the Universal Device Index for a disk device.

Normal Return

The PPN for the specified device is returned in the ac. Note that if you have enabled /NEW in your search list, the returned PPN for SYS will be [1,5] instead of [1,4].

Error Return

The error return occurs in two cases. The cause of the error is indicated by the value returned:

- If zero is returned in the ac; the device does not exist, or you have not initialized it.
- If your own PPN is returned; the device is not a disk device.

Related Calls

DEVCHR, DEVLNM, DEVNAM, DEVOP, DEVSIZ, DEVSTS, DEVTYP, PATH.

DEVSIZ [CALLI 101]

Function

Returns the buffer size and standard number of buffers for a device.

Calling Sequence

```
        MOVEI   ac,addr
        DEVSIZ  ac,
          error return
        normal return
        . . .
addr:   EXP     status
       {SIXBIT/device/}
       {EXP     channo }
       {EXP     udx    }
```

where: **addr** is the address of the argument block. Normally, the address points to the OPEN block used to initialize the device.

status is the I/O status word, which must match the information given when the channel was initialized with INIT, OPEN, or FILOP.

device is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

Note that the format for the argument block is identical to the format used for the OPEN monitor call and that the OPEN block is ordinarily used as the DEVSIZ block. The number and sizes of buffers differ among different data modes, and depending on mode modifier bits.

Normal Return

The ac contains the default number of buffers in its left half, and the default buffer size (including a 3-word header) in its right half. If you specify a device that was initialized in dump mode, the monitor clears the ac and takes the normal return.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
-2	DVSIM%	Illegal data mode.
-1	DVSNX%	Nonexistent device.
0	DVSDM%	Dump mode specified; therefore, buffer size is not applicable.

Related Calls

DEVCHR, DEVLNM, DEVNAM, DEVOP., DEVPPN, DEVSTS, DEVTYP

DEVSTS [CALLI 54]

Function

Returns the device status word from the device data block (DDB). This call returns the last CONI performed for the device, which is different for each device type and model. To interpret the device status word, refer to the hardware manual for the specific device.

Calling Sequence

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

where: **device** is the SIXBIT physical or logical name of a device.

channo is the number of a channel.

udx is the Universal Device Index for a device.

You can specify any device on an I/O bus. Where multiple units are on a single controller, the status of the controller is returned.

Normal Return

The device status word is returned in the ac. If the service routine for the device does not store a CONI, the returned word may be useless. Devices having both a controller and data interrupt store the controller CONI.

Error Return

If the device does not exist or is not initialized, the ac is cleared.

Related Calls

DEVCHR, DEVLNM, DEVNAM, DEVOP., DEVPPN, DEVSIZ, DEVTYP

Common Errors

- Confusing "device status" (DEVSTS) with "I/O status" (GETSTS). GETSTS returns the file (I/O) status bits, which are documented in Volume 1. DEVSTS returns the hardware device status.
- Confusing the "device status" returned by DEVSTS with the I/O error status that is returned by the DEVOP. UUO.

DEVTYP [CALLI 53]

Function

Returns the physical properties for a device.

Calling Sequence

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

where: **device** is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

Normal Return

If the ac is Ø, there was no such device; otherwise, the device type bits are returned in the ac as follows:

Bits	Symbol	Characteristic
Ø	TY.MAN	Directory device; a LOOKUP/ENTER is mandatory.
1-7		Reserved.
8	TY.GEN	If the argument is a SIXBIT name, this bit is set if the device is generic.
9	TY.MDA	Controlled by MDA (mountable device allocator).
1Ø	TY.EHF	Extended hardware features; for example, this bit is set for a line printer with lowercase capability.
11	TY.MPX	MPX-controllable.
12	TY.AVL	Available to your job.
13	TY.SPL	Spooled.
14	TY.INT	Interactive; there is output after each break character.
15	TY.VAR	Capable of variable buffer size.
16	TY.IN	Input capability.
17	TY.OUT	Output capability.
18-26	TY.JOB	Job number to which the device is currently assigned.
27-28		Reserved.

Bits	Symbol	Characteristic
29	TY.RAS	Restricted; assigned only to privileged job or by MOUNT command.
30-35	TY.DEV	One of the following device type codes:

Code	Symbol	Device Type
0	.TYDSK	Disk.
1	.TYDTA	DECtape.
2	.TYMTA	Magnetic tape.
3	.TYTTY	Terminal.
4	.TYPTR	Papertape reader.
5	.TYPTP	Papertape punch.
6	.TYDIS	Display unit.
7	.TYLPT	Line printer.
10	.TYCDR	Card reader.
11	.TYCDP	Card punch.
12	.TYPTY	Pseudo-terminal.
13	.TYPLT	Plotter.
14	.TYEXT	External task.
15	.TYMPX	MPX-controlled.
16	.TYPAR	PA611R on a DC44.
17	.TYPCR	PC11(R) on a DC44.
20	.TYPAP	PA611P on a DC44.
21	.TYLPC	LPC-11 on a DC44.
22	.TYPCP	PC-11(P) on a DC11.
23	.TYWTY	WTY device on a DC11.
24	.TYTSK	Network task.
25	.TYD78	DAS78 device.
26	.TYRDA	Remote data entry device.
27	.TYMCR	Monitor command interpreter (MCR) device.
30	.TYDRA	DTR01/DR01 device.
31	.TYKDP	KMC/DUP interface.
32	.TYDTE	DTE interface.
33	.TYDDP	DDP device.
34	.TYDMR	DMR device.
35	.TYRX2	RX02 floppy disk controller.
36-57		Reserved for use by DIGITAL.
60-77		Reserved for use by customers.

Error Return

The DEVTYP monitor call should never take the error return.

Related Calls

DEVCHR, DEVLNM, DEVNAM, DEVOP., DEVPPN, DEVSIZ, DEVSTS

Common Errors

Assuming that a normal return indicates that the device exists.

DIAG. [CALLI 163]

Function

Provides diagnostic functions for devices and CPUs.

Calling Sequence

```
        MOVE   ac,[-len,,addr]
        DIAG.  ac,
          error return
        normal return
          . . .
addr:   fcn-code
        argument
```

where: **len** is the length of the argument list.

addr is the address of the argument list.

fcn-code is one of the function codes listed below.

argument is different for each function code; each type of argument is described with its function code below.

Code	Symbol	Meaning
Ø-6		Reserved for use by DIGITAL.

7	.DIAKU	Returns the controller and unit numbers for a device. The format of the argument list is:
		
		`addr: EXP .DIAKU`
		`SIXBIT/device/`
		
		On a normal return, Bits Ø-8 contain Ø, Bits 9-17 contain the controller device code, Bits 29-32 contain the unit number, and Bits 33-35 contain the slave unit number.

1Ø	.DIACS	Forces a CPU status block read on a CPU and forces DAEMON to make an error entry (code 63) in ERROR.SYS. (The error codes and entry types are listed with the DAEMON call.) This function requires that you have JP.POK, [1,2], or JACCT privileges. The format of the argument list is:
		
		`addr: EXP .DIACS`
		`EXP CPU-number`

11	.DIADS	Reads the device status for all devices on the specified CPU into a GETTAB table in the monitor and forces DAEMON to make an error entry (code 64) in ERROR.SYS. (The error codes and entry types are listed with the DAEMON call.) This function requires that you have JP.POK, [1,2], or JACCT privileges. The format for the argument list is:
		
		`addr: EXP .DIADS`
		`EXP CPU-number`

Code	Symbol	Meaning
13	.DIANL	Enables and disables automatic reload of microcode in DX2Ø when the DX2Ø fails. addr: EXP .DIANL EXP flag,CPUn where the flag specifies the action and CPUn is the CPU number. If flag=Ø, automatic reloading of the DX2Ø is enabled. A non-zero value for the flag disables auto-reload. The CPU number can be -1 to indicate all CPUs.
15	.DIASM	Seizes the specified magnetic tape controller(s). The argument list is formatted as: addr: EXP .DIASM SIXBIT/device/ device may be either a device name, or zero to indicate all units.
16	.DIARM	Releases the specified magnetic tape controllers. The argument list is formatted as: addr: EXP .DIARM SIXBIT/device/ device may be either a device name, or zero to indicate all units.
17	.DIELD	Enables microcode loading. The argument list is formatted as: addr: XWD CPUno, .DIELD
2Ø	.DIDLD	Disables microcode loading. The format of the argument list is: addr: XWD CPUno, .DIDLD
21	.DILOD	Loads the microcode. The format of the argument block is: addr: XWD CPUno, .DILOD
22	.DIISM	Sets IPA channel (CI2Ø or NTA2Ø) maintenance mode. The format of the argument block is: addr: XWD CPUno, .DIISM
23	.DIICM	Clears IPA channel maintenance mode. The format of the argument block is: addr: XWD CPUno, .DIICM

Code	Symbol	Meaning
107	.DIACC	Manipulates the CI port counters. The format of the argument list is:

```
addr:  XWD CPUno, .DIACC
       XWD channo,sub-ftn
```

The only valid channel number is 7. The counters are manipulated in the manner indicated in the sub-function. Valid sub-functions are:

Code	Function
0	Gets counters.
1	Releases counters.
2	Point counters.
3	Reads counters.

Normal Return

The specified function has been performed.

Error Return

The ac is unchanged if the DIAG. monitor call is not implemented on the system. Otherwise, one of the following error codes is returned in the ac:

Code	Symbol	Meaning
1	DIANP%	Not enough privileges.
2	DIAIA%	Illegal number of arguments.
3	DIAIC%	Illegal controller number.
4	DIAIU%	Illegal unit number.
5	DIAAA%	Some units already assigned.
6	DIADM%	Unit not in diagnostic mode.
7	DIAAJ%	Unit assigned to another job.
10	DIAFC%	Not enough free core.
11	DIAAU%	No assigned units.
12	DIACP%	IOWD crosses page boundary.
13	DIAIF%	Illegal function.
14	DIAVC%	Job must not be virtual.
15	DIANC%	No such CPU.
16	DIANR%	CPU not running.
17	DIABA%	Invalid argument list.
20	DIACI%	No CI port on specified CPU.
21	DIATO%	The Read Port Counters function timed out.
22	DIANK%	No NI port on specified CPU.
23	DIARF%	Microcode reload failed.
24	DIANM%	No microcode available.
25	DIAPN%	CI or NI port not running.

DISK. [CALLI 121]

Function

Performs miscellaneous disk functions.

Calling Sequence

```
MOVE   ac,[XWD fcn-code,addr]
DISK.  ac,
  error return
normal return
```

where: **fcn-code** is one of the function codes described below.

addr is the address of the argument list. Each function code requires a different argument list. Therefore, the argument lists are described below, with the appropriate function codes.

Normal Return

On a successful return from the call, the function you specified is accomplished, and neither the ac nor the argument list is affected.

Error Return

Each function can produce its own set of error codes on an error return from the DISK. call. The error code is returned in the ac. A negative error code is one of the following, general-purpose error codes:

Code	Symbol	Meaning
-2	DUINP%	Not enough privileges to perform the function.
-1	DUILF%	Illegal function requested.

A positive error code indicates an error that is specific to the function code. The ac is unchanged if DISK. is not implemented on your system. In the argument lists described below:

device is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

structure is the SIXBIT name of a file structure.

The function codes, their meanings, argument lists, and error codes are:

Code	Symbol	Function
Ø	.DUPRI	Sets the disk priority level. The argument list for .DUPRI is:

```
addr:   XWD     channo,priority
```

where: channo is an initialized channel number. You can use -2 to indicate all channels for the job, or -1 for all explicitly initialized channels for this job.

priority is in the range -3 to +3 (Ø is normal priority and +3 is the highest priority).

If you set the priority for a channel, the setting overrides the setting for the job, and remains in effect until you change it or release the channel.

If you set the priority for the entire job, the setting remains in effect until you change it with another DISK. call or with a SET DSKPRI monitor command.

The maximum priority level you can use for your job is stored in bits 1 and 2 (JP.DPR) of the job privilege table (GETTAB table 6, .GTPRV).

On an error return from this function, one of the following error codes may be stored in the ac:

Code	Symbol	Meaning
1	DUPIP%	Priority higher than JP.DPR.
2	DUPNO%	Channel not initialized.
3	DUPIA%	Illegal channel number or code.

1	.DUSEM	Sets PDP-1Ø/PDP-11 compatibility mode (22-sector mode on the RPØ4/RPØ6) for the channel. .DUSEM is a privileged function. The argument list for .DUSEM is:

```
addr:     EXP   channo
```

On an error return from this function, one of the following error codes may be returned in the ac:

Code	Symbol	Meaning
1	DUSID%	Illegal device.
2	DUSCM%	The device does not support 22-sector mode.

Code	Symbol	Function
2	.DUSTM	Clears PDP-10/PDP-11 compatibility mode. .DUSTM is a privileged function. The argument list for .DUSTM is: addr: EXP channo On an error return from this function, one of the following error codes may be returned in the ac: Code Symbol Meaning 1 DUSID% Illegal device. 2 DUSCM% The device does not support 22-sector mode.
3	.DUUNL	Unloads an RP04 or RP06 drive. .DUUNL is a privileged function. The argument list for .DUUNL is: addr: SIXBIT/device/ where device is the name of a specific unit, such as RPA0. On an error return from this function, one of the following error codes may be returned in the ac: Code Symbol Meaning 1 DUUIU% Illegal unit name. 2 DUUNI% Structure is illegal or not available. 3 DUUNU% Device cannot be unloaded.
4	.DUOLS	Takes a controller/channel off-line soon. The monitor will continue I/O that is in progress, but will not use the controller for new I/O requests. .DUOLS is a privileged function. The argument list for .DUOLS is: addr: SIXBIT/controller/ where controller is the name of a disk controller, such as RPA. On an error return from this function, one of the following error codes may be returned in the ac: Code Symbol Meaning 1 DUOIP% Specified controller/channel is being put off-line. 2 DUOSK% Nonexistent controller. 3 DUOSS% If controller were set off-line, there would not be enough swapping space. 4 DUOIS% Unit in structure cannot be set off-line. 5 DUOES% Not enough space for IOWDs. 6 DUOPI% Obsolete

Code	Symbol	Function
5	.DUOLN	Takes a controller/channel off-line now. The monitor stops current I/O on that controller and will not use the controller for new I/O requests. .DUOLN is a privileged function. The argument list for .DUOLN is: addr: SIXBIT/controller/ On an error return from this function, one of the following error codes may be returned in the ac:

Code	Symbol	Meaning
1	DUOIP%	Specified controller/channel is being put off-line.
2	DUOSK%	Nonexistent controller.
3	DUOSS%	If controller were set off-line, there would not be enough swapping space.
4	DUOIS%	Unit in structure cannot be set off-line.
5	DUOES%	Not enough space for IOWDs.
6	DUOPI%	Obsolete

Code	Symbol	Function
6	.DUONL	Puts a controller/channel on-line. This function makes the controller available for I/O. .DUONL is a privileged function. The argument list for .DUONL is: addr: SIXBIT/controller/ On an error return from this function, one of the following error codes may be returned in the ac:

Code	Symbol	Meaning
1	DUOIP%	Specified controller/channel is being put off-line.
2	DUOSK%	Nonexistent controller.
3	DUOSS%	If controller were set off-line, there would not be enough swapping space.
4	DUOIS%	Unit in structure cannot be set off-line.
5	DUOES%	Not enough space for IOWDs.
6	DUOPI%	Obsolete

Code	Symbol	Function
7	.DUUFD	Sets call for UFD compressor. The argument list for .DUUFD is: addr: EXP channo where channo specifies a channel on which a file is open. The UFD in which the file exists will be compressed. This function does not force the compression to take place immediately, but sets the compression to be performed on the next output CLOSE for a file in this UFD. By default, the compression is performed on an output CLOSE only if the directory contains an empty block.
10	.DUSWP	Removes a disk unit from the active swapping list. .DUSWP is a privileged function. The argument list for .DUSWP is: addr: SIXBIT/device/ On an error return from this function, one of the following error codes may be returned in the ac:

Code	Symbol	Meaning
1	DUOIP%	Specified controller/channel is being put off-line.
2	DUOSK%	Nonexistent controller.
3	DUOSS%	If controller were set off-line, there would not be enough swapping space.
4	DUOIS%	Unit in structure cannot be set off-line.
5	DUOES%	Not enough space for IOWDs.
6	DUOPI%	Obsolete

11	.DUASW	Adds a disk unit to the active swapping list. .DUASW is a privileged function. The argument list for .DUASW is: addr: SIXBIT/device/ On an error return from this function, one of the following error codes may be returned in the ac:

Code	Symbol	Meaning
1	DUANU%	No such unit.
2	DUAAI%	Unit already in active swapping list.
3	DUASF%	SWPTAB is full.
4	DUAN4%	This error code is obsolete.
5	DUANS%	No swapping space (SWAP.SYS) on pack.

Code	Symbol	Function
12	.DUASD	Adds a structure to the system dump list. The argument list for .DUASD is: addr: SIXBIT/structure/ On an error return from this function, one of the following error codes may be returned in the ac:

Code	Symbol	Meaning
1	DUDND%	No such structure.
2	DUDNC%	No crash space on structure.
3	DUDAD%	Structure already on system dump list.
4	DUDDF%	System dump list full.

Code	Symbol	Function
13	.DURSD	Removes a structure from the system dump list. The argument list for .DURSD is: addr: SIXBIT/structure/ On an error return from this function, the following error code may be returned in the ac:

Code	Symbol	Meaning
1	DUDNS%	Structure not in system dump list.

Code	Symbol	Function
14	.DULEN	Returns the number of written blocks in the file in ac. The argument list for .DULEN is: addr: EXP channo
15	.DUCLM	Clears MDA wait for the specified unit. The argument list for .DUCLM is: addr: SIXBIT/device/ This function is used by the GALAXY batch and spooling system and requires [1,2] or JACCT privileges.

DISK. [CALLI 121]

Code	Symbol	Function
16	.DUFRE	Returns the amount of free space in a given UFD before the logged in quota is exhausted. The argument list for .DUFRE is:

```
addr:   SIXBIT/structure/
        XWD p,pn
```

If there is no job logged in with the specified PPN, the normal return is taken with bit Ø set. This bit setting is returned by the DSKCHR call, when DC.NPA is returned in .DCUFT (arg+1). This signifies the fact that the quota is not available.

On an error return from this function, the following error code may be returned in the ac:

Code	Symbol	Meaning
1	DUFND%	No such structure.

Related Calls

DSKCHR

DNET. [CALLI 207]

Function

Obtains information about DECnet network nodes and environment in your network area only. This monitor call is for use in system programs associated with DECnet-10 Versions 3.0 and 4.0.

NOTE

In a multi-area DECnet environment, the DNET.UUO only returns information about nodes in the same area as the DECnet-10 host.

If DECnet-10 is running as an Ethernet endnode, the DNET.UUO only returns information about the DECnet-10 host node.

Calling Sequence

```
        XMOVEI  ac,addr
        DNET.   ac,
          error return
        normal return
addr:   argument list
```

where: addr is the address of the argument list. Each function code requires a different argument list. The first word of every argument list (.DNFFL) contains the following information in the following bit fields:

addr: flags+fcn-code,,len

where flags (DN.FLA) are:

Bit	Symbol	Meaning
0	DN.FLS	Used with functions that return information about single entities (a node or link). Indicates that the function should step through the list, returning information about the next entity in the list.
1	DN.FLK	List information only about known nodes.
2	DN.FLR	List information only about active (reachable) nodes.
3	DN.FLE	List information only about EXECUTOR. nodes. Refer to the DECnet-10 User's Guide for more information.

The function codes and argument lists are described below:

Code	Symbol	Meaning
1	.DNLNN	Lists node names. You specify the following at addr:

```
addr:   flag+<.DNLNN,,length>
        BLOCK length-1
```

You must include one of the following flags:

Bit	Symbol	Meaning
1	DN.FLK	List known nodes.
2	DN.FLR	List active nodes.
3	DN.FLE	List EXECUTOR nodes.

And length is the length of the block to reserve.

The monitor returns the argument list in the following form:

Offset	Symbol	Contents
1	.DNCNT	Number of node names returned in the list.
2	.DNNMS	First node name
3	. . .	Second node name
	. . .	

2	.DNNDI	Returns information about a node. You specify the following at addr:

```
addr:   flag+<.DNNDI,,length>
        node-name
        BLOCK length-2
```

You must include one of the following flags:

Bit	Symbol	Meaning
Ø	DN.FLS	Step through list of nodes. If you set this flag, you must be sure that addr+1 will contain Ø on the first call, to start at the first node in the node list. The nodes are listed in numerical order, by address.
1	DN.FLK	List only known nodes.
2	DN.FLR	List only active nodes.
3	DN.FLE	List only EXECUTOR nodes.

And length is the length of the argument block returned. If you do not specify step mode by setting DN.FLS, you must specify the node-name in addr+1.

Code	Symbol	Meaning

The monitor returns the argument list in the following form:

Offset	Symbol	Contents
1	.DNNAM	Node name.
2	.DNRTR	Router information, in the following format: Bit 0 (DN.RCH) is set if the node is reachable. Bits 1-17 (DN.HOP) contain the number of hops to the specified node. Bits 18-35 (DN.CST) contain the cost of the path to the specified node.
3	.DNLLI	Link information, in the following format: Bit 0 (DN.VLD) is on if the word contains valid information. Bits 1-17 (DN.LNK) contain the number of active links to the node. Bits 18-35 (DN.DLY) contain the message delay time to the node.
4	.DNADR	Node address.
5-10	DNCKT	Circuit name, up to 4 ASCIZ words. This string may contain up to 16 characters.

3 .DNSLS Shows link status. You must specify the following at addr:

```
addr:   DN.FLS+<.DNSLS,,length>
        jobno,,channo
```

where the flag DN.FLS (step through node list) is optional, and length is the number of words reserved for the returned argument list. If you set DN.FLS, be sure that addr+1 is 0 on the first call, so that the information is returned starting at the first node in the node list.

The monitor returns the argument list in the following form:

Offset	Symbol	Contents
1	.DNJCN	Currently displayed job number and link number.
2	.DNNOD	Remote node name, in SIXBIT.
3	.DNOBJ	Object types, where the left half (DN.DOB) contains the destination object type, and the right half (DN.SOB) contains the source object type.
4	.DNSTA	Status word. The left half of this word (DN.LSW) contains the status variable bits and the link status code. The variable bits are:

Bit	Symbol	Meaning
Ø	NS.IDA	Interrupt data is available.
1	NS.IDR	Interrupt data may be sent.
2	NS.NDA	Normal data is available.
3	NS.NDR	Normal data may be sent.

The remainder of the left half contains a numeric code associated with the symbol that is stored in the right half.

The right half of this word (DN.STA) contains a SIXBIT symbol representing the status of the link. The status codes and associated SIXBIT symbols are:

Code	Symbol	State
1	CW	Connect wait.
2	CR	Connect message received.
3	CS	Connect message sent.
4	RJ	Remote task rejected connect initiation message.
5	RN	Link is up and running.
6	DR	Disconnect message received.
7	DS	Disconnect message sent.

Offset	Symbol	Contents		
		Code	Symbol	State
		10	DC	Disconnect message has been confirmed.
		11	CF	No confidence in link.
		12	LK	No link exists.
		13	CM	No communication has taken place.
		14	NR	No resources exist.
5	.DNQUO	Quota word, where the left half (DN.IQT) contains the input quota, and the right half (DN.OQT) contains the output quota.		
6	.DNSEG	Segment size.		
7	.DNFLO	Flow control option, where the left half (DN.XMF) contains the flow control option used for transmission, and the right half (DN.RCF) contains the flow control option used for receiving messages.		
10	.DNMSG	Message count word, where the left half (DN.MRC) contains the number of messages received, and the right half (DN.MXM) contains the number of messages transmitted.		
11	.DNMPR	Monitor process word. If the job number at .DNJCN is -1, this is the terminal number that NRTSER has been given for this particular link. This word is 0 for any job number other than -1.		

Error Return

On an error, one of the following error codes is returned in the ac:

Code	Symbol	Error
1	DNADE%	Address error.
2	DNWNA%	Wrong number of arguments.
3	DNIDN%	Illegal job number.
4	DNFNE%	Illegal function number.
5	DNILF%	Illegal flag set.
6	DNNSN%	No such node name.
7	DNNSC%	No such channel.
10	DNNDA%	Node is in a different DECnet area.

DNET. [CALLI 207]

Normal Return

If DECnet-10 Version 3 or 4 is included in your monitor, the normal return is taken for every DNET. call except when it reaches the end of the node list while returning information about nodes in the list.

Example

The following example shows the programming sequence used to list known nodes, up to the specified length, starting at location DNARG.

```
        MOVE    T1,[DN.FLK+<.DNLNN,,100>]
        MOVEM   T1,DNARG
        MOVEI   T1,DNARG
        DNET.   T1,
          HALT                  ;Error return
DNARG:  BLOCK   100
```

On a normal return, the argument block is filled with the following information:

```
DNARG:  DN.FLK!<.DNLNN,,100>  ;Fcn-code+flags
        20                    ;Number of nodes
        SIXBIT/ONE/           ;Node names
        SIXBIT/TWO/
        SIXBIT/THREE/
        SIXBIT/KL1026/
        SIXBIT/JINX/
        SIXBIT/GNOME/
        .
        .
        .
```

DSKCHR [CALLI 45]

Function

Returns the characteristics of a disk device. These characteristics are needed to allocate storage efficiently on the disk.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        DSKCHR  ac,
          error return
        normal return
        . . .
addr:   SIXBIT/name/
        BLOCK   len-1
```

where: **name** is the SIXBIT name of a file structure, a controller type, a controller, a logical unit, a physical unit, a physical device, or a channel number. The value of **len** is the number of words in the argument list.

If more than one unit was specified, the monitor returns values in the ac and the argument block, pertinent to the first unit specified. If more than one file structure was specified, the monitor returns values in the **ac** and argument block, pertinent to the first unit on the first file structure.

Normal Return

On a successful return, the disk characteristics are returned in **addr+1** through **addr+<length-1>**, and disk status flags are returned in the ac. For a complete list of I/O status bits, refer to Chapter 11.

The contents of the returned argument block are:

Offset	Symbol	Contents
Ø	.DCNAM	The argument supplied for the call. This is the only word in the argument block that the user program supplies. The .DCNAM argument may be a channel number.
1	.DCUFT	The number of blocks left in your job's quota before the UFD is exhausted. If this value is negative (DC.NPA==1BØ), the UFD has not been accessed since the job logged in, and the quota is not available. To obtain this information for jobs other than your own, use the .DUFRE function of the DISK. UUO.
2	.DCFCT	The number of first-come, first-served blocks available to all users.
3	.DCUNT	The number of blocks available to all users on this file structure.
4	.DCSNM	SIXBIT name of the structure to which this unit belongs.

Offset	Symbol	Contents
5	.DCUCH	The size characteristics: Bits — Symbol — Meaning Ø-8 — DC.UCC — Number of blocks per cluster. 9-17 — DC.UCT — Number of blocks per track. 18-35 — DC.UCY — Number of blocks per cylinder.
6	.DCUSZ	Number of 128-word blocks on the unit.
7	.DCSMT	Mount count for the structure. This count is the number of jobs that performed a MOUNT command for this file structure without executing a DISMOUNT command. Note that LOGIN performs an implied MOUNT of all structures in DSK, the default job search list.
1Ø	.DCWPS	Number of words per SAT block.
11	.DCSPU	Number of SAT blocks for each unit.
12	.DCK4S	Space (in K) allocated for swapping.
13	.DCSAJ	Mount word for the structure: Value — Meaning Ø,,Ø — No job or more than one job has the structure mounted. -1,,n — One job (number n) has the structure mounted and the structure is not single-access. Ø,,n — One job (number n) has the structure mounted and the structure is single-access.
14	.DCULN	SIXBIT logical name of the unit.
15	.DCUPN	SIXBIT physical name of the unit.
16	.DCUID	SIXBIT identification of the unit.
17	.DCUFS	First logical block to be used for swapping.
2Ø	.DCBUM	Number of blocks per unit (including maintenance cylinders).
21	.DCCYL	Current cylinder number.
22	.DCBUC	Number of blocks per unit in PDP-11 compatibility mode.
23	.DCLPQ	Length of the position wait queue.
24	.DCLTQ	Length of the transfer wait queue.
25	.DCALT	Unit name for alternate port.
26	.DCOWN	Owner PPN of structure.
27	.DCPAS	Position in active swapping list if argument was a physical unit; -1 if not in list.

Offset	Symbol	Contents
3Ø	.DCPSD	Position in system dump list if argument was a structure; -1 if not in list.
31	.DCBSC	Blocks per super-cluster.
32	.DCXCH	The extended unit characteristics:

Bits	Symbol	Meaning
Ø-8	DC.XCC	Data channel number
9-17	DC.XCK	Unit controller number
18-26	DC.XCU	Physical unit number
27-35	DC.XCA	Bit mask of accessible CPUs (1B35=CPUØ, 1B34=CPU1, etc.)

33	.DCDET	Name of the alternate port. The port does not have to be attached.
34	.DCNUS	The name of the next unit in the specified file structure.
35	.DCBRC	Count of blocks read by buffered I/O.
36	.DCBWC	Count of blocks written by buffered I/O.
37	.DCDRC	Count of blocks read by dump I/O.
4Ø	.DCDWC	Count of blocks written by dump I/O.
41	.DCMRC	Count of blocks read by monitor I/O.
42	.DCMWC	Count of blocks written by monitor I/O.
43	.DCSRC	Count of blocks read by swap I/O.
44	.DCSWC	Count of blocks written by swap I/O.
45	.DCPRC	Count of blocks read by paging I/O.
46	.DCPWC	Count of blocks written by paging I/O.
47	.DCFKS	Remaining swap space.
5Ø	.DCCBK	Count of disk cache blocks in use.
51	.DCCRC	Count of disk cache read calls.
52	.DCCRH	Count of disk cache read hits.
53	.DCCWC	Count of disk cache write calls.
54	.DCCWH	Count of disk cache write hits.
55	.DCSDV	Count of soft device/search errors.
56	.DCSDT	Count of soft data errors.
57	.DCHDV	Count of hard device/search errors.
6Ø	.DCHDT	Count of hard data errors.
61	.DCECT	Count of retries on last error.

Offset	Symbol	Contents
62	.DCSER	Count of SAT errors.
63	.DCRER	Count of RIB errors.
64	.DCCER	Count of software checksum/consistency errors.
65	.DCHBN	Logical block number of last error (within unit).
66	.DCERR	Last error status.
67	.DCSDF	Last error status.
70	.DCHDI	Last error status.
71	.DCSDI	Last error status.
72	.DCNHG	Count of non-recoverable transfer-hung errors.
73	.DCTHG	Count of transfer-hung errors.
74	.DCPHG	Count of position-hung errors.
75	.DCSHG	Count of software-hung errors.
76	.DCMAX	Length of DSKCHR argument block. The value of this symbol may change as words are added to the DSKCHR block.

The flags returned in the ac are as follows:

Bits	Symbol	Meaning
0	DC.RHB	Disk pack off-line; the monitor must reread the home block before the next operation to verify the pack identification.
1	DC.OFL	Unit is off-line.
2	DC.HWP	Hardware write-protected.
3	DC.SWP	Belongs to write-protected file structure.
4	DC.SAF	Belongs to single-access file structure.
5	DC.ZMT	Mount count is zero.
6	DC.PRV	Belongs to private file structure.
7-8	DC.STS	Status code for unit:

Code	Symbol	Status
0	.DCSTP	Has pack mounted.
2	.DCSTN	No pack mounted.
3	.DCSTD	Unit down.

9	DC.MSB	Unit has more than one SAT block.

Bits	Symbol	Meaning
10	DC.NNA	Belongs to a structure that has a lock to prevent further INIT, LOOKUP, ENTER, OPEN, and FILOP. calls (NNA - "no new access"). This lock is set by a privileged STRUUO function.
11	DC.AWL	Write-locked for all jobs.
12-13	DC.CPU	CPU number of the CPU to which the device is connected. DC.XCC in word .DCXCH supersedes DC.CPU.
14	DC.ALT	Dual-ported device.
15-17	DC.TYP	Type of argument passed with the DSKCHR call:

Code	Symbol	Meaning
0	.DCTDS	Generic name, such as DSK.
1	.DCTAB	File structure subset, because of abbreviation, such as D.
2	.DCTFS	File structure name, such as DSKA.
3	.DCTUF	Unit within file structure, such as DSKA0.
4	.DCTCN	Controller class name, such as FH.
5	.DCTCC	Controller name, such as RPA.
6	.DCTPU	Physical unit, such as RPA0.

18-20	DC.DCN	Data channel number that software lists as connected to hardware; first data channel is 0.
21-26	DC.CNT	Controller type:

Code	Symbol	Controller Type
1	.DCCFH	RC10 for RD10 and RM10-B.
2	.DCCDP	RP10 for RP02 and RP03.
4	.DCCFS	RH10 for fixed head disk.
5	.DCCRP	RH10/RH20/RH11 for moving head disk (RP04, RP06, RP07, and RM03).
6	.DCCRN	RH20 for RP20.
7	.DCCRA	HSC50 for CI disks.

27-29	DC.CNN	Controller number; first one of each type is 0.

DSKCHR [CALLI 45]

Bits	Symbol	Meaning
30-32	DC.UNT	Unit type:

Code	Symbol	Meaning	When
Ø	.DCUFD	RD1Ø	(DC.CNT=1)
Ø	.DCUS4	RSØ4	(DC.CNT=4)
Ø	.DCUR4	RPØ4	(DC.CNT=5)
Ø	.DCUNØ	RP2Ø	(DC.CNT=6)
Ø	.DCU8Ø	RA8Ø	(DC.CNT=7)
1	.DCUFM	RM1Ø-B	(DC.CNT=1)
1	.DCUD2	RPØ2	(DC.CNT=2)
1	.DCUR6	RPØ6	(DC.CNT=5)
1	.DCU81	RA81	(DC.CNT=7)
2	.DCUD3	RPØ3	(DC.CNT=2)
2	.DCUR3	RMØ3	(DC.CNT=5)
2	.DCU6Ø	RA6Ø	(DC.CNT=7)
3	.DCUR7	RPØ7	(DC.CNT=5)

33-35	DC.UNN	Physical unit number within the controller; first one is Ø.

Error Return

The error return occurs if any of the following conditions is found:

- The argument at addr is Ø.
- The device does not exist or channel is not initialized.
- The argument is illegal.

Example

The following example checks a user's logged-in quota on structure DSKB:

```
        MOVE   T1,[2,,ADDR]
        DSKCHR T1,
          JRST NOQTA
        SKIPGE ADDR+.DCUFT
          JRST NOQTA
          . . .
ADDR:   SIXBIT /DSKB/
        BLOCK 1
```

This code tests the value returned from the DSKCHR call. When DSKCHR fails, or when no quota is returned at ADDR+1, the program jumps to NOQTA, where it must act on the possibility that the structure is not mounted or there is no quota on the structure.

DTE. [CALLI 17Ø]

Function

Performs functions for the DTE (KL systems only), and is not recommended for customer programs. To use the DTE. monitor call, you must have the JP.POK or JACCT privilege, or be logged in under [1,2].

Calling Sequence

```
        MOVE  ac,[fcn-code,addr]
        DTE.  ac,
          error return
        normal return
        . . .
addr:   argument list
```

where: **fcn-code** is one of the function codes described below.

addr is the address of the argument list. Each function requires a different argument list. These are described below.

In the following discussion of the DTE. functions,

- **cpuno** is the number of a CPU.
- **dteno** is the number of a DTE.
- **fedno** is the unit number of a front-end device.

The function codes and their meanings are:

Code	Symbol	Function
Ø	.DTECL	Clears a PDP-11 on a DTE. The argument list for the .DTECL function is: addr: XWD cpuno,dteno
1	.DTEST	Starts primary protocol on a DTE. The argument list for the .DTEST function is: addr: XWD cpuno,dteno
2	.DTETB	Sets the byte pointer for messages being transferred to the DECsystem-1Ø. The argument list for the .DTETB function is: addr: XWD cpuno,dteno EXP <byte pointer to DECsystem-1Ø>
3	.DTEEB	Sets the byte pointer for messages transmitted to the PDP-11. The argument list for the .DTEEB function is: addr: XWD cpuno,dteno EXP <byte pointer to PDP-11>

DTE. [CALLI 17Ø]

Code	Symbol	Function
4	.DTERW	Returns the PDP-11 reload ROM word in the ac. The argument list for the .DTERW function is: addr: XWD cpuno,dteno If bit 4 (DT.RP4) is set on return, the PDP-11 got code from the disk.
5	.DTEMN	Return (in ac) the master DTE number for the CPU. The argument list for the .DTEMN function is: addr: XWD cpuno,dteno
6	.DTEPR	Presses the PDP-11 reload button. The argument list for the .DTEPR function is: addr: XWD cpuno,dteno
7	.DTEGS	Returns the status word for the DTE. The status word for the specified DTE is returned in ac. The argument list for the .DTEGS function is: addr: XWD cpuno,dteno The status flags that can be returned are: Flag Symbol Meaning 6 DT.DTX DTE exists. 7 DT.DTM DTE is master DTE. 8 DT.PPC DTE is running primary protocol. 9 DT.SPC DTE is running secondary protocol. 1Ø DT.RLD DTE needs reloading.
1Ø	.DTERJ	Sets reload job number. The argument list for the .DTERJ function is: addr: EXP jobno where: jobno is the job number for the reload.
11	.DTEGF	Returns front-end device unit number. The argument list for the .DTEGF function is: addr: XWD cpuno,dteno EXP fedno
12	.DTEIF	Front-end device input. The argument list for the .DTEIF function is: addr: XWD cpuno,dteno EXP fedno XWD byte-count, addr-of-input-buffer
13	.DTEOF	Front-end device output. The argument list for the .DTEOF function is: addr: XWD cpuno,dteno EXP fedno XWD byte-count,addr-of-output-buffer

Code	Symbol	Function
14	.DTEFG	Returns (in ac) the front-end device status. The argument list for the .DTEFG function is:

```
addr:   XWD     cpuno,dteno
        EXP     fedno
```

The returned device status flags are:

Flag	Symbol	Meaning
28	DT.FER	Fatal error.
29		Reserved.
3Ø	DT.EOF	End of file.
31	DT.IOP	I/O in progress.
32	DT.SER	Soft error.
33	DT.HER	Hard error.
34	DT.OFL	Off-line.
35	DT.NXD	Nonexistent device.

15 .DTEFS — Sets front-end device status. The argument list for the .DTEFS function is:

```
addr:   XWD     cpuno,dteno
        EXP     fedno
        EXP     status
```

where: status is the status word for the front-end device.

16 .DTEFR — Releases a front-end device. The argument list for the .DTEFR function is:

```
addr:   XWD     cpuno,dteno
        EXP     fedno
```

17 .DTERC — Releases KL error chunks. The argument list for the .DTERC function is:

```
addr:   XWD     cpuno,Ø
```

2Ø .DTERT — Releases the KL error timer. The argument list for the .DTERT function is:

```
addr:   XWD     cpuno,Ø
```

21 .DTEDT — Returns Universal Device Indexes for terminal lines leading to the DL11s on the specified DTE. The argument list for this function is:

```
addr:   XWD     cpuno,dteno
```

On a successful return, the UDX is returned in the ac. However, for DTE Ø, which is dedicated to the console front end (RSX-2ØF), the ac contains the KLINIK line's UDX in the left half, and the CTY's UDX in the right half.

DTE. [CALLI 170]

Code	Symbol	Function
22	.DTESU	Specifies the type of protocol that will run on the DTE. The argument list for this function is:

```
addr:   XWD     cpuno,dteno
        SIXBIT/user-name/
```

where the user-name is one of the following protocol types:

DECNET	for DECnet-10.
ANF	for ANF-10.
IBM	for IBM communications.
NOBODY	if the DTE is not running a protocol.
PROGRA	if the DTE is dedicated to a job.

23	.DTERU	Reads the protocol type of the protocol that is running on the DTE. The argument list is:

```
addr:   XWD     cpuno,dteno
        BLOCK   2
```

The information is returned in the following format:

```
addr:   XWD     cpuno,dteno
        SIXBIT/user-name/
        EXP     jobn
```

where user-name is the name of the protocol running on the DTE (refer to .DTESU above). The job number (jobn) is returned in addr+2 only if user-name is PROGRA.

24	.DTELS	Loads a secondary bootstrap from your job's memory area, using the PDP-11 bootstrap ROM. This function must be preceded by the .DTECL (clear) and .DTEPR (press reload) functions. You must also use function .DTEDM (dump) before you can load any bootstrap. The argument list for this function is:

```
addr:   XWD     cpuno,dteno
        POINT   16,addr1
        EXP     length
```

where addr+1 contains a byte pointer indicating the location of the secondary loader, and length is the length of the loader, in 16-bit bytes.

Code	Symbol	Function
25	.DTEDM	Dumps PDP-11 memory, using the PDP-11 bootstrap ROM. Before you use this function, be sure to use functions .DTECL (clear) and .DTEPR (press reload). You must always dump the PDP-11 memory before you can load a program into its memory. The argument list for this function is: addr: XWD cpuno,dteno POINT 16,addr1 EXP count where addr+1 contains a byte-pointer to the memory that must be dumped, and where count is the number of 16-bit bytes to dump from the PDP-11.
26	.DTKPS	Not intended for customer use.
27	.DTKPR	Not intended for customer use.

Normal Return

The function is performed, and any requested value is stored in the ac.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	DTENP%	Not enough privileges.
2	DTEUF%	Illegal function code.
3	DTEDC%	Illegal CPU or DTE number.
4	DTEAP%	Primary protocol already running.
5	DTEPT%	Power fail did not come up.
6	DTEDE%	Doorbell did not clear.
7	DTTTE%	To TOPS-10 error during BOOT sequence.
10	DTEDD%	No response from PDP-11 after BOOT sequence.
11	DTEIJ%	Illegal job number.
12	DTEIB%	Illegal byte count.
13	DTENI%	Front-end device not initialized.
14	DTEFB%	Front-end device in use by another job.
15	DTENF%	Nonexistent front-end device.
16	DTEFE%	Fatal error on front-end device.
17	DTESE%	Error starting primary protocol.
20	DTENC%	No free core for front-end device buffers.
21	DTETE%	KL error data timer expired.
22	DTECM%	The FEDSER monitor module was told not to send messages to the PDP-11.
23	DTEIU%	Tried to set line to illegal user value.
24	DTEWU%	Wrong line user for function.
25	DTEEV%	No exec virtual memory to perform function.
26	DTEIP%	Illegal byte pointer.

DVPHY. [CALLI 164]

Function

Returns the physical names of all devices of a given type, or of all devices supported by the system (except pseudo-terminals, terminals, MPX devices, and disks).

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        DVPHY.  ac,
          error return
        normal return
        . . .
addr:  {EXP     device-type}
       {EXP     -1         }
        BLOCK   1
```

where: **len** is the length of the argument block (must be 2).

addr is the address of the argument block.

device-type is one of the device type codes returned from the DEVTYP monitor call, such as .TYLPT for a line printer. A value of -1 means all devices supported by the system.

On the first DVPHY. call, addr+1 should contain Ø; the monitor returns the name of the first device. If you leave this name in addr+1, the next DVPHY. call returns the name of the next device, and so forth. When all devices have been returned (by several calls), the monitor returns Ø in addr+1.

Normal Return

For Ø in addr+1, the monitor returns the name of the first device; for a device name in addr+1, the monitor returns the name of the next device, or, if no more, Ø. The ac is unchanged.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	DVPIA%	Illegal argument length.
2	DVPIT%	Illegal device type.
3	DVPNP%	Nonexistent physical device.
4	DVPNT%	Nonexistent device type.

Example

The following example shows how to obtain the physical names of all line printers on the system:

```
        SETZB   T1,ADDR+1           ;Initialize counter and device name
TAG1:   MOVE    T2,[XWD 2,ADDR]     ;Set up call
        DVPHY.  T2,                 ;Get name
          JRST  ERROR               ;Error
        SKIPN   T3,ADDR+1           ;Get name, skip if not at end
          JRST  TAG2                ;Ø means we're done
        MOVEM   T3,LPTNAM(T1)       ;Save in next block-slot
        AOJA    T1,TAG1             ;Increment count and loop
TAG2:   MOVEM   T1,NLPT             ;Save count
        JRST    CONTIN
NLPT:   BLOCK   1
LPTNAM: BLOCK   1Ø
ADDR:   EXP     .TYLPT              ;Type is LPT
        EXP     Ø                   ;Start with first device
CONTIN:
```

Related Calls

SYSPHY, SYSSTR

Common Errors

Using a SIXBIT name for device type.

DVRST. [CALLI 122]

Function

Restricts the use of a device. Once restricted, the device is then assignable only by the operator; unprivileged users must request assignment through the MOUNT monitor command before using the OPEN/INIT monitor call. (See the Commands Manual.) Privileged users (JACCT or [1,2]) can still use the OPEN or INIT monitor call, or the ASSIGN command, if the device is not controlled by MDA.

The DVRST. monitor call requires the JACCT privilege or that you be logged in under [1,2].

Calling Sequence

```
{MOVE    ac,[SIXBIT/device/]}
{MOVEI   ac,channo          }
{MOVEI   ac,udx             }
 DVRST.  ac,
   error return
 normal return
```

where: **device** is the SIXBIT physical or logical name of a device to be designated as being restricted.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

Normal Return

The device is restricted.

Error Return

The error return occurs if any of the following conditions is found (the ac is unchanged):

- You do not have the JACCT privilege or are not logged in under [1,2].
- The specified device does not exist.
- The device is a disk.

Related Calls

DVURS.

DVURS. [CALLI 123]

Function

Removes the restriction created by a DVRST. monitor call. DVURS. requires the JACCT privilege or that you be logged in under [1,2].

Calling Sequence

```
{MOVE   ac,[SIXBIT/device/]}
{MOVEI  ac,channo           }
{MOVEI  ac,udx              }
 DVURS. ac,
   error return
 normal return
```

where: **device** is the SIXBIT physical or logical name of a device that is to be returned to unrestricted status.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

Normal Return

The restriction is removed. The device is available for public use and returned to the monitor's pool of available devices.

Error Return

The error return occurs if any of the following conditions is found (the ac is unchanged):

- You do not have the JACCT privilege or are not logged in under [1,2].
- The given device does not exist.

Related Calls

DVRST.

ENQ. [CALLI 151]

Function

Requests access to resources that are defined by cooperating user programs. The ENQ. call is one of three monitor calls that provide control over the ENQ/DEQ facility, which provides resource definition, control over access to resources, and deadlock detection for the resources. The ENQ/DEQ facility is described in Chapter 8.

Calling Sequence

```
        MOVE   ac,[XWD fcn-code,addr]
        ENQ.   ac,
          error return
        normal return
        . . .
addr:   EXP     <size>B5+<number>B17+<len>B35     ;header block
        XWD     Ø,request-id
        XWD     time-limit
        <lock block>
        <lock block>
            .
            .
            .
```

where: **fcn-code** is one of the function codes listed below.

addr is the address of the argument block, which consists of a header block followed by one or more lock blocks.

The header block contains 1 to 3 words, in the following order:

Offset	Symbol	Contents
Ø	.ENQLL	The header size, the number of lock requests, and the total length of the argument, including the header and all the words in all the lock blocks. Specifically, the .ENQLL word is formatted as follows:

Bits	Symbol	Value
Ø-5	EQ.BHS	Size of the header block. This value is between 1 and 3, because the second and third words are optional. If you omit this value, the default is 2.
6-17	EQ.LNL	Number of lock blocks following the header block. Include one lock block for each resource requested.

Offset	Symbol	Contents		
		Bits	Symbol	Value
		18-35	EQ.LLB	Total length (in words) of the argument block. All the lock blocks in a single request must be the same length. Thus, the value of EQ.LLB is the header block length (EQ.BHS) plus the length of each lock block times the number of resources requested (EQ.LNL).
1	.ENQRI	An 18-bit request-id identifying this request. This optional value identifies the ENQ. request, enabling you to identify it when it causes a software interrupt. This is useful when you use the ENQ/DEQ facility in conjunction with the software interrupt (PSI) system. After an interrupt is generated, the request-ids of the granted requests are inclusively ORed into the status word of the interrupt block. To receive a software interrupt, use function code 2 (.ENQSI) when you issue the ENQ. monitor call. The request-id can also be used with the DEQ. call to dequeue a specific request.		
2	.ENQTL	Time limit specifying the number of seconds to wait for each request in the call to be granted. If any resource is not available within that time limit, the call takes the error return with the ENQTL% error code in the ac. This word is optional. If you include the time limit in the header block, specify 3 for size in word Ø.		

Each lock block represents a separate ENQ. request. There is no limit to the number of locks that can be requested, but multiple requests in the same ENQ. call must be given level numbers. The locks will be granted in the order of the level numbers.

The format of a lock block is:

```
{EXP      <flags+level>B17+<channo>B35};lock block
{                                     }; byte-pointer
{EXP      flags+user-code             }
{EXP      user-code                   }
{XWD pool-size,number}
{XWD Ø,sharer-group  }
 XWD mask-len,mask-addr
 XWD block-len,block-addr
```

where a lock block is two to five words long, identifying the resource to be locked and describing the characteristics of the lock. The first requestor of a resource defines lock characteristics. Subsequent requests for the same resource must conform to those characteristics or wait until the resource is released by the first requestor.

In the case of multiple-lock requests, all the lock blocks in a single ENQ. request must be the same length. Specifically, a lock block can contain the following words:

Offset	Symbol	Contents
Ø	.ENQFL	Contains the flag bits, level number, and channel number. The flags are:

Bit	Symbol	Meaning
Ø	EQ.FSR	The lock request allows sharers. If you do not set this bit, the monitor assumes that you require exclusive access to the resource. Unless the first requestor for the resource sets this flag, no requests for the same resource (specified in the next word, .ENQBP) can be granted until the first requestor dequeues it (using DEQ. or RESET). If the first requestor sets this bit, other programs with the same sharer group number as that specified in .ENQPS can obtain access to the resource while it is locked for your job.
1	EQ.FLB	Bypass level checking. When multiple request blocks are made in a single ENQ. call, you must assign a level number to each request. When EQ.FLB is not set, lower-level resources will be granted before higher-level resources are considered. The EQ.FLB flag prevents this level-checking, allowing resources to be granted regardless of their order by level number.
2	EQ.FLT	Grant a long-term lock. That is, after the resource is dequeued by all users, the lock data is preserved for about 5 minutes.
3	EQ.FEL	Grant an eternal lock. This prevents the resources from being dequeued automatically when your program performs a RESET function.
4	EQ.FAB	Abort the resource. This prevents the resource from being accessible to any other user. A request for an aborted lock causes error code ENQAB% to be returned in the ac. The resource cannot be granted to another user until it is dequeued.

Bit	Symbol	Meaning
5	EQ.FDD	Set deadlock detection. This flag prevents your request from causing a deadlock among resource users. If this flag is set, and granting your request would cause a deadlock, the ENQ. call takes the error return with error code ENQDD% in the ac.
6	EQ.FCW	Specifies that a 36-bit user code is included in the next word (.ENQBP). This is the preferred method of specifying a user code in the lock block.
7-8		Reserved for use by DIGITAL.
9-17	EQ.FLV	9-bit **level-number** that you assign to each request in a multiple-lock request. In a multiple-lock request (a single ENQ. call containing multiple lock blocks), each lock block must be assigned a level number; the locks will be granted in ascending numerical order according to level number, unless you set EQ.FLB (bypass level checking) in the flag word.
18-35	EQ.FCC	The number of the **channel** on which the resource is being accessed (positive integer), to associate the lock with the file that is open on that channel. Alternatively, you can specify a negative number indicating one of the following conditions:

Code	Symbol	Meaning
-3	.EQFPL	The lock requested is a privileged global lock and the resource is available only to [1,2] or JACCT jobs. This code allows privileged jobs to define locks on resources to prevent access from unprivileged jobs.

Offset	Symbol	Contents

Code	Symbol	Meaning
-2	.EQFGL	The lock you are requesting is a global lock. Specifying this code prevents access to the resource from any other job. To use this code, your job must have JP.ENQ set in its privilege word.
-1	.EQFJB	The lock is a job-wide lock, preventing access to the resource from any other requests by your job.

1 .ENQBP

Specifies the resource to be locked. You can use a pointer to an ASCIZ string, or a user-code in this word (more on this later). You must include this word in every lock block because it defines the resource you are requesting.

When the first program to request the resource is granted a lock, it is said to have ownership of the resource. When a second program makes an ENQ. call with the same value in this word (if a user-code is specified) or the same ASCIZ string (if a byte pointer is used), the request is for the same resource that was granted to the first job. The contents of the word to which the byte pointer refers, or the user code itself, are purely arbitrary values to the monitor. The monitor only checks lock requests for matches, granting or preventing locks on the basis of matching strings. This is the key to the ENQ/DEQ access-checking mechanism.

If the flag EQ.FCW is set in the previous word (.ENQFL), .ENQBP must contain a 36-bit value as a user-code.

If EQ.FCW is not set, and the flag EQ.BUC is set in .ENQBP (that is, a value of 5 is placed in bits Ø-2), the rest of the word must contain a 33-bit user-code.

If neither flag is set, this word (.ENQBP) must contain a pointer to an ASCIZ text string. This may be either a standard byte pointer in the form:

```
POINT 7,address,bit-location
```

Offset	Symbol	Contents
		Or, if the ASCIZ string is stored in 7-bit bytes, starting at the first byte of the location being referenced, the pointer can take the form: XWD -1,address The ASCIZ string at address can be up to 30 (decimal) words. The maximum string length for your system is stored in %EQMSS in GETTAB table .GTENQ. Cooperating programs (those requesting the same resources) must specify exactly the same user code or ASCIZ string.
2	.ENQPS	Specifies either a pool number for a pooled resource, or a sharer group number for a sharable resource. This word is optional and defaults to 0. For a pooled resource, the word contains the pool-size in the left half and the number of resources requested from the pool in the right half. For a sharable resource, the left half is zero, and the right half contains the sharer-group number. Thus, a resource cannot be pooled and also be accessible to a sharer group. A pooled resource is defined by the first requestor of the resource. By specifying the number of resources in the pool, the requestor is defining the number of "copies" of the resource to be made available. Each copy of the resource can be requested for exclusive access by specifying the same resource identifier in word .ENQBP and the same pool-size in word .ENQPS. The requestor must also specify, in the right half of .ENQPS (EQ.PPR), the number of copies of the resource to lock. If the left half of .ENQPS (EQ.PPS) is 0, the right half (EQ.PPR) specifies the sharer-group number, thus defining a group of jobs that can simultaneously share the resource. Any program that sets the flag EQ.FSR and specifies the same resource and the same sharer group number will be granted its request. Therefore, when you share the ownership of a resource, only other jobs in the same sharer group are allowed ownership of the resource. The sharer group number defaults to 0. Therefore, if the first requestor specifies a sharable resource but omits the sharer group number, all subsequent sharable requests for the resource that also omit the sharer group number, or that set it to 0, will be granted immediately.

ENQ. [CALLI 151]

Offset	Symbol	Contents
3	.ENQMS	Contains a pointer to the bit mask representing the portions of a resource to be locked. The pointer consists of the mask-len stored in the left half, and the mask-addr in the right half. The bit mask, describing fields of bits to be locked, is stored at the location specified in mask-addr, and the length of the bit mask (in words) is stored in the mask-len. This provides a facility for partitioning the resource, allowing locks on portions of a resource. This word is optional and defaults to Ø.
4	.ENQTB	Contains the block-length and block-addr of a lock-associated data block. This data block can be used to pass information to subsequent users of a resource. To use this facility, you should set EQ.FLT, thus preserving all lock request data for the resource for at least 5 minutes after you dequeue the resource. This word is optional and defaults to Ø.

The function codes and their meanings are:

Code	Symbol	Function
Ø	.ENQBL	Requests ownership of a resource. Your job will block if the resource is not available. Your request is placed in a queue associated with the specified resource. If more than one request was included in your ENQ. call, your job will block until all the requests have been granted. After all requests have been granted, the normal return is taken and the monitor clears the ac. If you set the flag EQ.FBL in .ENQFL of the request block (to bypass level checking), the monitor could return a nonzero value. A nonzero value indicates that a level number sequencing error occurred, but it was ignored because you specified that level numbers were to be bypassed.
1	.ENQAA	Requests ownership of a resource and returns immediately if the resource is unavailable. If all requests specified in this argument cannot be granted immediately, the system will not enter any requests in the queues associated with those resources and the error return is taken with error code 1 (ENQRU%) in the ac. However, if the system can grant all of your requests immediately, the normal return is taken and the monitor leaves the ac unchanged.

Code	Symbol	Function
2	.ENQSI	Requests ownership of a resource and, if the resource is not immediately available, causes a software interrupt when the resource becomes available. You can use this function when the Programmable Software Interrupt (PSI) system is enabled, to prevent your job from blocking while waiting for the requests to be granted. If all the requests in the call can be granted immediately, this function is equivalent to function code Ø (.ENQBL). If any of the requests are not available, the call takes the error return with error code 1 (ENQRU%) in the ac. In this case, your job can continue processing until it receives a .PCQUE software interrupt. The interrupt control block will contain the request-ids of the requests that are granted, inclusively ORed into the status word. The PSI system is described in Chapter 6.
3	.ENQMA	Modifies an existing request made by your job. If the modification you specify in this request is identical to the request you made earlier, no action is taken and the normal return is taken from the call. If you do not have a request in any queue, the error return is taken and the monitor returns error code 24 (ENQNE%) in the ac. If you specify more than one request with this function code and the error return is taken, you must issue the ENQC. monitor call to determine which (if any) modification request was granted. The error code that the monitor returns in the ac reflects only the last error that occurred as the result of this call. You can modify a request from exclusive ownership to shared ownership, but you will receive an error code if you attempt to modify a request from shared to exclusive ownership if other jobs are also sharing the resource. To modify a request from shared to exclusive ownership when other jobs are sharing the resource, you must first DEQ. the request. Then, enqueue it again as an exclusive ownership request.

Normal Return

All requests in the call are granted, and the resources are locked for your program.

ENQ. [CALLI 151]

Error Return

One of the following error codes is returned in the ac on an error return. These error codes are also returned on an error from a DEQ. or ENQC. call. They are described in more detail in Chapter 8.

Code	Symbol	Error
1	ENQRU%	At least one of the requested resources is not available.
2	ENQBP%	You requested an illegal number of pooled resources.
3	ENQBJ%	You specified an illegal job number.
4	ENQBB%	You specified an illegal byte size for the byte pointer. The byte size must be between 1 and 36, inclusive.
5	ENQST%	The ASCIZ string is too long. It must be less than 3Ø (decimal) words. (Refer to GETTAB table .GTENQ, item %EQMSS for current maximum message length.)
6	ENQBF%	You specified an illegal function code.
7	ENQBL%	You specified an illegal argument list length. The total argument list for the call must be header-block-size plus (request-block-size times number-of-requests).
1Ø	ENQIC%	You specified an illegal number of requests.
11	ENQBC%	You specified an illegal channel number.
12	ENQPI%	Your program does not have enough privileges for the given function.
13	ENQNC%	Not enough core available, or the maximum number of active locks (item %EQMAQ in GETTAB table .GTENQ) has been exceeded.
14	ENQFN%	Device is not initialized or is not a disk.
15	ENQIN%	The address for the byte pointer is indirect or indexed; this is not allowed.
16	ENQNO%	Your program cannot dequeue resources it does not own.
17	ENQLS%	Levels are not specified in ascending order.
2Ø	ENQCC%	Illegal modification of ownership; you cannot change a request from shared to exclusive ownership. You must DEQ. the request, then ENQ. it with EQ.FSR set.
21	ENQQE%	Your ENQ quota has been exceeded; your quota is set by the system administrator.
22	ENQPD%	The number of resources in the pool disagrees with the number in your request.
23	ENQDR%	Duplicate request; identical to a request already in the queue.
24	ENQNE%	Not enqueued on this lock.
25	ENQLD%	Level in request does not match lock.
26	ENQED%	Insufficient privileges for this function; you must have JP.ENQ set.
27	ENQME%	Mask too long or lengths do not match.
3Ø	ENQTE%	Lock-associated table is too long.
31	ENQAB%	A resource that was requested has been marked as aborted.
32	ENQGF%	Attempt to ENQ. with EQ.FEL option on a ghost file.
33	ENQDD%	Deadlock detected.
34	ENQTL%	Time limit exceeded.

Example

```
                .
                .
                .
                MOVE     T2,[XWD 10,ADDR]     ;Initialize channel
                FILOP.   T2,
                  JRST ERROR1                 ;Error return
                JRST     ENQ                  ;Normal return

ADDR:           XWD CHAN,1                    ;Channel and LOOKUP
                EXP MODE                      ;Data mode
                SIXBIT/DSK/                   ;Device
                0,,0                          ;Buffer addresses
                0,,0
                0,,ADDR1                      ;LOOKUP block
                0,,0
                0,,0

ADDR1:          SIXBIT/FILE/                  ;File name
                SIXBIT/EXT/                   ;Extension
                XWD 0,0
                XWD 27,4072                   ;Project-programmer
                                              ;Number

ENQ:            MOVE     T3,[XWD 0,ADDR3]     ;Function and address
                ENQ.     T3,
                  JRST ERROR3                 ;Error return
                JRST NORM                     ;Normal return

ADDR3:          XWD 1,5                       ;Number of locks, length of block
                XWD 0,1                       ;Request identifier
                XWD 0,CHAN                    ;Resource on chan
                POINT 7,[ASCIZ/NAME/]         ;Pointer to resource
                0,,0                          ;Not a pooled resource

                .
                .
                .

ERROR1:         OUTSTR ERROR2
                JRST DONE

ERROR3:         OUTSTR ERROR4
                JRST DONE

ERROR2:         ASCIZ/ERROR WITH FILOP./


ERROR4:         ASCIZ/ERROR WITH ENQ./

                .
                .
                .
DONE:           EXIT
```

Related Calls

DEQ., ENQC.

ENQC. [CALLI 153]

Function

Returns information about the current state of ENQ/DEQ requests and sets access rights for the ENQ/DEQ facility (privileged). Refer to Volume 1 for more information about using the ENQ/DEQ calls. For more information about the contents of the argument block, refer to the ENQ. call.

Calling Sequence

Each function of the ENQC. call requires a different calling sequence. The calling sequence for each ENQC. function is described below, for the appropriate function.

The ENQC. function codes and their meanings are:

Code	Symbol	Function
Ø	.ENQCS	Returns a 3-word status block for each specified lock. The calling sequence for the .ENQCS function is:

```
        MOVE    ac,[XWD .ENQCS,addr]
        MOVEI   ac+1,buffer
        ENQC.   ac,
          error return
        normal return
        . . .
addr:   EXP     parameters
        XWD     Ø,request-id
        EXP     time limit
        first word of first lock block
        . . .
        last word of last lock block
buffer: BLOCK   <locks>*3
        . . .
```

where: addr is the address of the argument list.

buffer is the address of a buffer (of length locks*3) for storing the returned three-word status blocks.

parameters is a word of the form:

<size>B5+<locks>B17+<length>B35

where: size is the size of the header block (1 to 3).

locks is the number of lock blocks in the argument list.

Code Symbol Function

length is the length of each lock block (size plus number of locks times the length of each lock block).

The right half of addr+1 may contain **request-id**, an optional request identifier.

time limit is an optional time limit for the request to be granted.

On a normal return, the monitor returns a three-word status block for each request, at **buffer**. The format of each block is:

Offset	Symbol	Meaning
Ø	.ENQCF	Flags:

Bits	Symbol	Meaning
Ø	EQ.CFI	Invalid lock.
1	EQ.CFO	This user is owner.
2	EQ.CFQ	This user is in queue for specified resource.
3	EQ.CFX	Owner's access is exclusive.
4-8		Reserved.
9-17	EQ.CFL	Level number.
18-26	EQ.CFC	The owner's context number.
27-35	EQ.CFJ	The owner's job number or an error code.

1	.ENQCT	A time stamp, in universal date-time format, indicating the time that the lock was granted; or Ø, indicating that the resource is available.
2	.ENQCI	The left half (EQ.CIQ) contains the number of users sharing the resource. The right half (EQ.CID) contains the request-id of the owner of the lock.

Code	Symbol	Function
1	.ENQCG	Returns user's quota in ac. The calling sequence for the .ENQCG function is:

```
        MOVE    ac,[XWD .ENQCG,addr]
        ENQC.   ac,
          error return
        normal return
        . . .
addr:   XWD     Ø,jobno
```

where: jobno is the number of the job whose ENQ quota is required. If jobno is -1, your own job is assumed.

2 .ENQCC Changes user's quota. The calling sequence for the .ENQCC function is:

```
        MOVE    ac,[XWD .ENQCC,addr]
        ENQC.   ac,
          error return
        normal return
        . . .
addr:   XWD     quota,jobno
```

where: quota is the new ENQ/DEQ quota.

jobno is the number of the job whose quota is to be changed. If jobno is -1, your own job is assumed.

This function sets the lock quota for a specific job. To perform this function, you must have POKE privileges, be a [1,2] job, or be running with the JACCT bit set. The ENQ/DEQ quota for the specified job will be set to the value you specify in the left half of the argument word on a normal return. On a normal return, the ac is cleared. If you attempt to use this function without the required privileges, the error return is taken and the monitor returns an error code in the ac.

3 .ENQCD Dumps the data base. The calling sequence for the .ENQCD function is:

```
        MOVE    ac,[XWD .ENQCD,addr]
        ENQC.   ac,
          error return
        normal return
        . . .
addr:   XWD     Ø,len
```

where: addr is the address of a buffer to receive the returned data.

len is the length of the data block to be returned, minus one word.

Code	Symbol	Function
3	.ENQCD (Cont.)	This function dumps the data base (all lock and queue entries). The entire data base is placed in your area, beginning with addr+1. If this length is not large enough to accommodate the entire data base, the monitor returns as much as possible of the data base. The end of the data base is indicated by a word containing -1. You must have SPY privileges to specify this function code. The format of the returned data is described in Chapter 8.

Normal Return

The requested function is performed.

Error Return

The error codes that can be returned in the ac on an error return are identical to those that can be returned from the ENQ. and DEQ. calls. The error codes are listed in the description of the ENQ. call.

Related Calls

DEQ., ENQ.

ENTER [OPCODE Ø77]

Function

Specifies an output file to create, supersede, or update a file. Use FILOP. to perform an ENTER for an extended I/O channel.

Calling Sequence

The ENTER monitor call has two types of argument lists: one using a four-word argument list and one using an extended argument list. The extended argument list offers many additional options for ENTERing a file. For complete information about the argument lists, refer to Section 11.13.

The calling sequence for the ENTER UUO is:

```
ENTER   channo,addr
  error return
normal return
```

where: addr is the address of the argument list. Refer to Section 11.13 for more information about the argument list.

Normal Return

When you use the short form of the argument block, the monitor returns a four-word argument block at addr.

Refer to Section 11.13.1 for information about the argument block that is returned.

When you use the extended argument list, the monitor returns the information that is listed on Section 11.13.2.

Error Return

On an error return from ENTER, the monitor returns an error code in either of the following:

- For the short-form argument block, the error code is stored in the right half of addr+1 of the 4-word argument block
- For the extended-form argument block, the error code is returned in the right half of addr+3.

It is possible to LOOKUP/RENAME a file after using an ENTER to specify the argument list, referring to the same argument list with subsequent calls. Note, however, that on an error return from the ENTER, the error code overwrites the high-order three bits of the creation date and the entire access date. Because most programs recover from these errors by either aborting or by reinitializing the entire argument block, this overwriting of data normally does not cause any problems. However, a program may attempt to recover from an error by fixing only the incorrect portion of the argument block and then reexecuting the monitor call. These programs should always initialize the contents of these locations before reexecuting the ENTER monitor call.

Error codes are restricted to a maximum of 15 bits to eliminate problems when recovering from an error in a file with a zero creation date. The error codes are described in Section 11.14.

Examples

See Chapter 11.

Related Calls

ENTER, FILOP., INIT, LOOKUP, OPEN, RENAME

ENTVC. [CALLI 225]

Function

Reads or sets an entry vector. An entry vector indicates the entry point for a program. (Refer to the TOPS-1Ø LINK Reference Manual and the TOPS-1Ø MACRO Assembler Manual for more information on entry vectors.)

Calling Sequence

```
        XMOVEI  ac,addr
        ENTVC.  ac,
          error return
        normal return

addr:   flag,,function
        length
        exec-addr
```

where: addr is the address of the argument list.

flag indicates whether the vector is read (Ø) or set (EN.SET).

function is the function code described below.

length is a value returned by the monitor on a read, and supplied by you on a set. A (JRST) in the right half indicates that you are supplying a start address only in exec-addr (if setting), or that the monitor is returning the start address in exec-addr (if reading). Otherwise, you supply a length for the entry vector (Ø-37 words, octal), or the monitor returns the length of the entry vector.

exec-addr contains the 3Ø-bit address of the entry vector or the start address, returned on a read, or to be set for the set function. exec-addr is a user address.

The function code for ENTVC. is:

Code	Symbol	Meaning
Ø	.ENVRS	Reads or sets the entry vector. Set the EN.SET flag in the left half of the first word of the argument block to perform a set. Ø in the left half indicates a read should be performed.

Normal Return

The specified function is performed.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	EVIAL%	Illegal argument list.
2	EVIFC%	Illegal function code.
3	EVADR%	An address check was encountered.

ERLST. [CALLI 132]

Function

Returns data giving the status of each device on an MPX channel that has errors.

Calling Sequence

```
        MOVEI   ac,addr
        ERLST.  ac,
          error return
        normal return
        . . .
addr:   XWD     length,channo
        BLOCK   length-1
```

where: **addr** is the address of the argument block.

length is the length of the argument block; the length should be the number of devices connected to the channel plus two.

channo is the number of an initialized channel.

Normal Return

The monitor returns data at **addr+1** for devices on the channel that have errors. The data at **addr** is in the format:

```
addr:   XWD length,channo
        EXP number of devices
        XWD udx,status
        . . .
        XWD udx,status
```

where: **length** and **channo** were given in the call.

number of devices is the number of devices on the channel that have encountered errors.

udx is the Universal Device Index of a device having errors.

status is a halfword containing I/O status bits for the device. These bits are identical to those returned for a GETSTS monitor call.

The monitor continues to return device error information in the argument block until all space allocated by your program has been filled. Your program should check the value of addr+1. If addr+1 is greater than the length of the argument block minus two, the device error list is incomplete because of lack of space.

For a list of I/O status bits, see the appropriate device in Volume 1.

ERLST. [CALLI 132]

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	ERLBC%	Illegal channel number.
2	ERLNM%	Not an MPX-channel.

Related Calls

CLRST., GETSTS, SENSE.

ERRPT. [CALLI 160]

Function

ERRPT. is a privileged monitor call used only by the DAEMON program to ask the monitor for the next error condition to be logged in the error file.

Calling Sequence

```
        MOVE   ac,[XWD len,addr]
        ERRPT. ac,
          error return
        normal return
        . . .
addr:   BLOCK 4
```

where: len is the length and addr is the address of the argument list that is filled in by the monitor on a normal return.

Normal Return

If an error condition (such as a stopcode or a hardware error) is found, the monitor places values in the locations at addr as follows:

Offset	Symbol	Contents
0	.ERPT0	Address, job number, and error code:

Bits	Symbol	Meaning
0-17	ER.PAD	Address used by DAEMON.
18-26	ER.PJN	Job number.
27-35	ER.PCD	Error code for the error file.

Offset	Symbol	Contents
1	.ERPT1	Two monitor internal addresses:

Bits	Symbol	Meaning
0-17	ER.PDA	Address of DDB for device with error condition.
18-35	ER.PUA	Address of UDB for device with error.

Offset	Symbol	Contents
2	.ERPT2	CPU number.

Bits	Symbol	Meaning
0-14		Reserved for use by DIGITAL.
15-17	ER.CPU	CPU number on which error was detected.
18-35		Reserved for use by DIGITAL.

Offset	Symbol	Contents
3	.ERPT3	Reserved for use by Digital.

ERRPT. [CALLI 160]

Error Return

One of the following conditions occurred:

- The ERRPT. call is not implemented.
- The calling sequence was improper.
- No appropriate error condition was found. In this case, the values of the words at addr are unchanged.

ETHNT. [CALLI 223]

Function

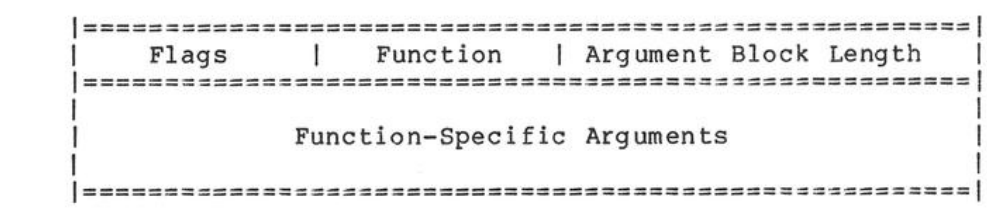
The ETHNT. monitor call accesses the Ethernet. ETHNT. allows you to read the Ethernet configuration, enable and disable protocols, enable and disable multicast addresses, and send and receive datagrams. For an overview of Ethernet, as well as a full description of datagram buffers (addressed using .ETUBL) and function buffers (addressed using .ETBFL and .ETBFA), refer to Chapter 5, Volume 1.

Calling Sequence

```
        XMOVEI  ac,addr
        ETHNT.  ac,
          error return
        normal return
```

where: addr is the address of the ETHNT. argument block.

The argument block is:



The format of the argument block is:

Offset	Symbol	Contents
Ø	.ETFCN	Function code word for the argument block. This word contains the length of the argument block, and may also contain a flag. Its format is:

Bits	Symbol	Meaning
Ø-8	ET.FFL	Function-specific flags.
9-17	ET.FFN	One of the function codes listed at the end of the argument block description.
18-35	ET.FLN	Length of the argument block.

Bit	Symbol	Meaning
1	ET.FZC	Zero counters after they have been read. Use this flag with functions .ETRCC, .ETRPC, and .ETRKC.

Offset	Symbol	Contents
1	.ETPSW	Contains the portal status and the assigned portal ID. ET.PST (Bits Ø-8) may contain one or more of the following flags:

Bit	Symbol	Meaning
Ø	ET.PON	Portal is online.
1	ET.PXB	Transmit buffers available.
2	ET.PRB	Receive buffers available.

The rest of .ETPSW contains the portal ID, .ETPID, assigned by the monitor.

1	.ETCSW	Contains the channel status and channel ID. ET.CST (Bits Ø-8) contains the channel status. If the ET.CON flag in ET.CST is on, the channel is online. ET.CID (Bits 9-35) contains the channel ID.
1	.ETKSW	Contains the status and ID of a controller. ET.KST (Bits Ø-8) contains the controller status. If the ET.KON flag in ET.KST is on, the controller is online. ET.KID (Bits 9-35) contains the controller ID.
2	.ETAR1	Contains the first function-specific argument. Function-specific arguments are described in each of the function codes below.
3	.ETAR2	Contains the second function-specific argument.

Valid function codes for .ETFCN are:

Code	Symbol	Meaning
1	.ETOPN	Opens a user portal. This function requires JP.POK privileges. This function specifies the protocol type to be enabled, and protocol specific flags. The argument block is:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETOPN in Bits 9-17, and the length of the argument block, 4, in the right half.
1	.ETPSW	Contains the portal-id in Bits 9-35. Returns an updated portal status in Bits Ø-8.
2	.ETCIW	Identifies the Ethernet channel on which the protocol should be enabled.
3	.ETPIW	Identifies the protocol type to be enabled on the Ethernet channel. Set the ET.PAD flag in the left half of .ETPIW to enable padding for the protocol.

Code	Symbol	Meaning
2	.ETCLS	Closes a user portal and releases all resources associated with it. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETCLS in Bits 9-17, and the length of the argument block, 2, in the right half.
1	.ETPSW	Contains the portal-id in Bits 9-35. Returns an updated portal status in Bits Ø-8.

Code	Symbol	Meaning
3	.ETQRB	Queues receive datagram buffers. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETQRB in Bits 9-17, and the length of the argument block, 3, in the right half.
1	.ETPSW	Contains the portal-id in Bits 9-35. Returns an updated portal status in Bits Ø-8.
2	.ETUBL	Contains the address of the user buffer descriptor list. Refer to Chapter 5, Volume 1 for the format of the user buffer descriptor list.

Code	Symbol	Meaning
4	.ETRRQ	Reads receive queue. This function fills each block in the buffer descriptor list with data appropriate to a received datagram. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETRRQ in Bits 9-17 and the length of the argument block, 3, in the right half.
1	.ETPSW	Contains the portal-id in Bits 9-35. Returns an updated portal status in Bits Ø-8.
2	.ETUBL	Contains the address of the user buffer descriptor list. Refer to Chapter 5, Volume 1 for the format of the user buffer descriptor list. The status field in .UBSTS of the buffer descriptor contains zero if the datagram was received successfully.

Code	Symbol	Meaning
5	.ETQXB	Transmits datagram buffer to the destination Ethernet address specified in the buffer descriptor block. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETQXB in Bits 9-17, and the length of the argument block, 3, in the right half.
1	.ETPSW	Contains the portal-id in Bits 9-35. Returns an updated portal status in Bits Ø-8.
2	.ETUBL	Contains the address of the user buffer descriptor list. Refer to Chapter 5, Volume 1 for the format of the user buffer descriptor list.

Code	Symbol	Meaning
6	.ETRXQ	Returns data associated with transmitted datagrams. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETRXQ in Bits 9-17, and the length of the argument block, 3, in the right half.
1	.ETPSW	Contains the portal-id in Bits 9-35. Returns an updated portal status in Bits Ø-8.
2	.ETUBL	Contains the address of the user buffer descriptor list. Refer to Chapter 5, Volume 1 for the format of the user buffer descriptor list. On a successful transmission, the returned status is zero.

Code	Symbol	Meaning
7	.ETEMA	Enables a portal to receive datagrams destined for an Ethernet multicast address. .ETEMA may not be used while a promiscuous receiver is active. The argument block for .ETEMA contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETEMA in Bits 9-17, and the length of the argument block, 4, in the right half.
1	.ETPSW	Contains the portal-id in Bits 9-35. Returns an updated portal status in Bits Ø-8.
2	.ETMCA	Contains the two word Ethernet multicast address.

Code	Symbol	Meaning
10	.ETDMA	Disables a portal from receiving datagrams bound for a multicast address. The multicast address you disable must have been previously enabled using the .ETEMA function. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETDMA in Bits 9-17, and the length of the argument block, 4, in the right half.
1	.ETPSW	Contains the portal-id in Bits 9-35. Returns an updated portal status in Bits Ø-8.
2	.ETMCA	Contains the two word Ethernet multicast address.

Code	Symbol	Meaning
11	.ETRCL	Returns a list of all known channels. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETRCL in Bits 9-17, and the length of the argument block, 4, in the right half.
1	.ETCSW	Reserved.
2	.ETBFL	Contains the length of the destination buffer.
3	.ETBFA	Contains the address of the destination buffer.

Code	Symbol	Meaning
12	.ETRCI	Returns information about a specific channel. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETRCI in Bits 9-17, and the length of the argument block, 4, in the right half.
1	.ETCSW	Contains the channel-id in Bits 9-35. Returns an updated channel status in Bits Ø-8. Flag ET.CON indicates whether the channel is on- or off-line.
2	.ETBFL	Contains the length of the destination buffer.
3	.ETBFA	Contains the address of the destination buffer.

Code	Symbol	Meaning
13	.ETRCC	Returns a list of the counters associated with a channel, and (optionally) zeroes them. Zeroing the counters requires JP.POK privileges. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETRCC in Bits 9-17, and the length of the argument block, 4, in the right half. Set the ET.FZC flag of .ETFCN if you want the counters zeroed after information is returned.
1	.ETCSW	Contains the channel-id in Bits 9-35. Returns an updated channel status in Bits Ø-8. Flag ET.CON indicates whether the channel is on- or off-line.
2	.ETBFL	Contains the length of the destination buffer.
3	.ETBFA	Contains the address of the destination buffer.

Code	Symbol	Meaning
14	.ETSCA	Sets the physical address associated with a channel. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETSCA in Bits 9-17, and the length of the argument block, 4, in the right half.
1	.ETCSW	Contains the channel-id in Bits 9-35. Returns an updated channel status in Bits Ø-8. Flag ET.CON indicates whether the channel is on- or off-line.
2	.ETEAD	Specifies the physical address. .ETEAD is two words long. It may not be a multicast address.

Code	Symbol	Meaning
15	.ETRPL	Returns a list of all portals on a channel. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETRPL in Bits 9-17, and the length of the argument block, 4, in the right half.
1	.ETCSW	Contains the channel-id in Bits 9-35. Returns an updated channel status in Bits Ø-8. Flag ET.CON indicates whether the channel is on- or off-line.

Code	Symbol	Meaning

Word	Symbol	Contents
2	.ETBFL	Contains the length of the destination buffer.
3	.ETBFA	Contains the address of the destination buffer.

The list is returned in the specified buffer, with each portal ID occupying a full word, right justified (.ETPSW format).

16 .ETRPI — Returns all information (except counters) about a specific portal. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETRPI in Bits 9-17, and the length of the argument block, 4, in the right half.
1	.ETPSW	Contains the portal-id in Bits 9-35. Returns an updated portal status in Bits Ø-8.
2	.ETBFL	Contains the length of the destination buffer.
3	.ETBFA	Contains the address of the destination buffer.

The information is returned in the specified buffer.

17 .ETRPC — Returns a list of the counters associated with a portal, and (optionally) zeroes them. Zeroing the counters requires JP.POK privileges. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETRPC in Bits 9-17, and the length of the argument block, 4, in the right half. Set the ET.FZC flag if you want the counters zeroed after the information is returned.
1	.ETPSW	Contains the portal-id in Bits 9-35. Returns an updated portal status in Bits Ø-8.
2	.ETBFL	Contains the length of the destination buffer.
3	.ETBFA	Contains the address of the destination buffer.

Code	Symbol	Meaning
20	.ETRKL	Returns a list of all controllers on a channel. The argument block contains:

Word	Symbol	Contents
0	.ETFCN	Contains the function code .ETRKL in Bits 9-17, and the length of the argument block, 4, in the right half.
1	.ETCSW	Contains the channel-id in Bits 9-35. Returns an updated channel status in Bits 0-8. Flag ET.CON indicates whether the channel is on- or off-line.
2	.ETBFL	Contains the length of the destination buffer.
3	.ETBFA	Contains the address of the destination buffer.

The list is returned in the specified buffer, with each portal ID occupying a full word, right justified (.ETKSW format).

Code	Symbol	Meaning
21	.ETRKI	Returns all information (except counters) about a specific controller. The argument block contains:

Word	Symbol	Contents
0	.ETFCN	Contains the function code .ETRKI in Bits 9-17, and the length of the argument block, 4, in the right half.
1	.ETKSW	Contains the controller-id in Bits 9-35. Returns an updated controller status in Bits 0-8. Flag ET.KON indicates whether the controller is on- or off-line.
2	.ETBFL	Contains the length of the destination buffer.
3	.ETBFA	Contains the address of the destination buffer.

The information is returned in the specified buffer.

Code	Symbol	Meaning
22	.ETRKC	Returns a list of the counters associated with a controller, and (optionally) zeroes them. Zeroing the counters requires JP.POK privileges. The argument block contains:

Word	Symbol	Contents
Ø	.ETFCN	Contains the function code .ETRKC in Bits 9-17, and the length of the argument block, 4, in the right half. Set the ET.FZC flag if you want the counters zeroed after the information is returned.
1	.ETKSW	Contains the controller-id in Bits 9-35. Returns an updated controller status in Bits Ø-8. Flag ET.KON indicates whether the controller is on- or off-line.
2	.ETBFL	Contains the length of the destination buffer.
3	.ETBFA	Contains the address of the destination buffer.

Normal Return

The requested function is performed, and information is returned as specified in the description of the function.

Error Return

One of the following codes is returned in the ac:

Code	Symbol	Error
1	ETPRV%	Program has insufficient privileges.
2	ETADC%	Address check attempting to read argument block.
3	ETIAL%	Illegal argument list length.
4	ETILF%	Illegal function code specified.
5	ETUEE%	Unexpected Ethernet error.
6	ETRES%	Insufficient resources.
7	ETIPI%	Invalid portal ID.
1Ø	ETICI%	Invalid channel ID.
11	ETIPT%	Invalid protocol type.
12	ETPIU%	Protocol type already in use.
13	ETPRA%	Promiscuous receiver active.
14	ETBAC%	Buffer address check.
15	ETIBS%	Invalid buffer size.
16	ETIBP%	Invalid byte pointer.
17	ETIEA%	Invalid Ethernet address.
2Ø	ETPQE%	Portal quota exceeded.
21	ETBQE%	Buffer quota exceeded.
22	ETPWS%	Protocol in wrong state.
23	ETIKI%	Invalid controller ID.

EXIT [CALLI 12]

Function

Stops job execution and optionally resets the job.

Calling Sequence

```
EXIT     fcn-code,
continue return
```

where: **fcn-code** is one of the function codes described below.

For either code, when you EXIT from a job in an auto-pushed context, you are returned to the superior context and the inferior one is deleted.

continue return is the instruction to be executed if the user issues a valid CONTINUE monitor command.

The function codes and their meanings are:

Code	Function
Ø	Performs the following: o Releases all I/O devices, closing files if necessary. o Unlocks the job from core. o Sets the user-mode write-protect bit for the high segment. o Resets APR traps to zero. o Clears PC flags. o Performs a RESET and stops the job. If timesharing was stopped by a TRPSET monitor call, the monitor resumes timesharing. A RESET monitor call is executed, and the word EXIT is typed on your terminal, and the terminal is left in monitor mode. You cannot continue with the CONT or CCONT monitor command.
1	Performs the following: o Clears PC flags. o Stops the job. EXIT is not printed on your terminal, and you can continue program execution with the CONT or CCONT monitor command. If you use function code 1, you should first RELEASE all devices and channels; a convenient way to do this is to use the RESET monitor call. The symbol for EXIT 1, is MONRT.
2-17	Reserved for use by DIGITAL.

Related Calls

LOGOUT, MONRT.

FILOP. [CALLI 155]

Function

Performs various file operations, including initializing channels and creating, deleting, writing, reading, renaming, appending to, and superseding files.

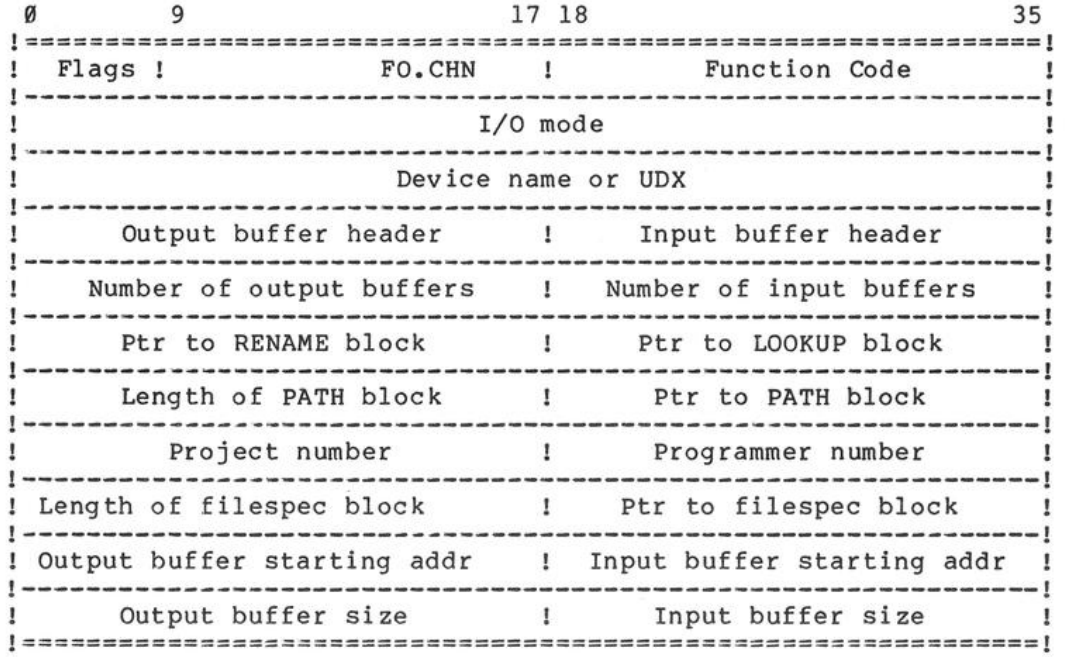
Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        FILOP.  ac,
          error return
        normal return
        . . .
addr:   arglst
```

where: **len** is the length of the argument list.

addr is the address of the argument list.

The argument block for FILOP. functions 1 through 6 and 13 through 15 looks like this:



The format of the argument list (for functions 1-6 and 13-15) is:

Offset	Symbol	Contents
0	.FOFNC	Flags, channel number, and function code:

Bits	Symbol	Meaning
0	FO.PRV	Indicates that the program with appropriate privileges ([1,2] or JACCT) will perform privileged FILOP. functions. You must set this bit to use privileged FILOP. functions.

Offset	Symbol	Contents		
		Bits	Symbol	Meaning
		1	FO.ASC	Assigns an extended channel number, one that is greater than 17. When you set this bit for an OPEN function (function codes 1 through 6) the monitor assigns the next available channel number. It then performs the requested function. On return, the monitor returns the assigned channel number in FO.CHN in the argument block or, if FO.CFW is set, in the address pointed to by the left half of the word at the location specified in FO.FNC. The number will be equal to or greater than 2Ø so that existing channel number allocations will not be affected.
		2	FO.UOC	Specifies the file that is open on the channel that is indicated in the right half of this word (.FOFNC). Normally, a RENAME function is performed on the file specified in the right half of .FOLEB. When you set this bit, however, the right half of .FOLEB is ignored, and the function is performed on the file that is open on the specified channel.
		3	FO.CFW	Indicates that the right half of this function word contains an address. At that address, you must store: channel-addr,,function-code where the channel-addr is the address where the channel number is stored. For this format, the field FO.CHN is ignored.
		4-8		Reserved for use by DIGITAL.
		9-17	FO.CHN	Channel number.
		18-35	FO.FNC	Function code (see below).
1	.FOIOS	I/O status (open mode). Note that any bits appearing here may also be set by OPEN call (see .OPMOD in OPEN call).		
2	.FODEV	SIXBIT device name or Universal Device Index.		

Offset	Symbol	Contents
3	.FOBRH	Buffer ring header pointers: Bits Meaning Ø-17 Address of output buffer ring header. 18-35 Address of input buffer ring header. If the value of this word is Ø, there is no corresponding buffer ring header.
4	.FONBF	Number of buffers needed. The left half is the number of output buffers needed. The right half is the number of input buffers needed. If zero buffers are requested in a FILOP. monitor call, the monitor does not set up any buffers. It also does not clear any buffer ring that is already set up, and does not clear the first word of the buffer ring header. Thus, a FILOP. causing an OPEN allows an old buffer ring to be recycled. This word allows a user program to set up its own buffer ring. If you specify 777777 octal, the monitor sets up a ring of 2 buffers for non-disk devices. If no default has been set for this job, the monitor uses the system default for non-disk devices, or a ring of n buffers for disk devices, where n is specified by the SET DEFAULT BUFFERS monitor command or SETUUO. This argument to FILOP. performs the same action as the INBUF and OUTBUF monitor calls and is needed only for buffered I/O.
5	.FOLEB	Pointers to RENAME and LOOKUP/ENTER blocks: Bits Meaning Ø-17 Address of RENAME block (see RENAME monitor call). 18-35 Address of LOOKUP/ENTER block (see LOOKUP/ENTER monitor call).
6	.FOPAT	Length of, and pointer to PATH. block (see PATH. monitor call). The actual path of the file found or created is returned in this block. A specific path for finding or creating the file must still be specified in the LOOKUP, ENTER, or RENAME argument block.
7	.FOPPN	Project-programmer number. Set the FO.PRV flag if you include this word and want it to take effect. The monitor then performs the file operation as if the current job were logged in under the given PPN. If FO.PRV is set in Word Ø (.FOFNC), and a PPN is supplied in this word, your program acquires the file access rights and restrictions of that PPN. This allows you to do file operations in behalf of the user whose PPN you include here. If you specify [1,2] in this word, you lose full file access. This word is ignored if the job is not logged in under [1,2] or does not have JACCT privileges.

Offset	Symbol	Contents
10	.FOFSP	Length of and pointer to a block in which the full file specification of the new file should be stored. If you include this word, the file specification is returned automatically. Alternatively, you can specify function 33 (.FOFIL) to only return the file specification. Refer to .FOFIL for the format of the returned block.
11	.FOBSA	Buffer starting address. The left half contains the starting address of the output buffer ring, (FO.OSA) the right half holds the starting address of the input buffer ring, (FO.ISA).
12	.FOBSZ	Size of the input and output buffers. The left half, FO.OSZ, contains the output buffer size. FO.ISZ, the right half, holds the size of the input buffer.
13	.FOMAX	Length of the FILOP. argument block.

The function codes and their meanings are:

Code	Symbol	Function
1	.FORED	Opens the file described by the LOOKUP/ENTER block for reading (duplicates LOOKUP call). You must include the LOOKUP/ENTER block pointer for directory devices when you are using this function.
2	.FOCRE	Creates the file described by the LOOKUP/ENTER block. This function strictly requires creation of the file; if a matching file is found in the directory, the error return is taken. The LOOKUP/ENTER block pointer is required for the .FOCRE function.
3	.FOWRT	Writes the file described by the LOOKUP/ENTER block. This function supersedes any matching file in the directory, or creates a new file. The LOOKUP/ENTER block pointer is required for the .FOWRT function.
4	.FOSAU	Updates the file described by the LOOKUP/ENTER block in exclusive access mode. No other user can write to this file until it is closed. The LOOKUP/ENTER block pointer is required for the .FOSAU function. If the specified file does not exist, it will be created automatically for this function.
5	.FOMAU	Updates the file described by the LOOKUP/ENTER block, in multi-access mode. This allows other users to read and write the file. The LOOKUP/ENTER block pointer is required for the .FOMAU function.

Code	Symbol	Function
6	.FOAPP	Appends to the file described in the LOOKUP/ENTER block. Note that if the buffers were built by this FILOP. call, the last block of the file will be read into the first buffer. The byte count and byte pointer are set to write data immediately after the last word of the file. The LOOKUP/ENTER block pointer is required for the .FOAPP function.
7	.FOCLS	Closes the file associated with the channel specified in the word at addr. This function does not accept the standard FILOP. argument list. Instead, you include the CLOSE flags (refer to the CLOSE UUO) in addr+1. The monitor executes a GETSTS call for the file. The I/O status bits are returned in the ac. For a list of I/O status bits, refer to the appropriate device chapter in Volume 1. If the argument block is two or more words long, then the right half of .FOIOS is taken to be the optional CLOSE status bits (for example, CL.DAT or CL.NMB). If there are errors, the monitor takes the error return and returns an error code in the ac.
10	.FOURB	Checkpoints the file associated with the channel specified in the word at addr. Only the function word of the FILOP. argument block is required. The monitor writes all output buffers to disk, updates directories, updates checksums in RIB pointers, and updates the end-of-file pointer. The file remains open for further I/O. The .FOURB function is meaningful only for files that are being written. NOTE If output is not complete, the monitor writes the last partially filled word; this may leave null bytes in the word.
11	.FOUSI	Performs a USETI monitor call (specifies next block number to be input) for a specified block of the file associated with the channel specified at addr, setting that block for next input. The format of the argument list for the .FOUSI function is: addr: XWD channo,.FOUSI EXP blockno where: channo and blockno give the channel number and block number of the file. Refer to the USETI call for more information. On a normal return, the I/O status bits are returned in the ac. The monitor takes the error return if the block number is larger than the specified file or no previous LOOKUP was executed. .FOUSI returns error code %ERILU if the argument block is not exactly two words long.

Code	Symbol	Function
12	.FOUSO	Performs a USETO monitor call for a specified block of the file associated with the channel specified at addr, setting that block for next output. The format of the argument list for the .FOUSO function is: addr: XWD channo,.FOUSO EXP blockno where: channo and blockno give the channel number and block number of the file. Refer to the USETO call for more information. The monitor takes the error return if not enough space is available or no previous ENTER was executed. The I/O status word is returned in the ac for a successful return. .FOUSO returns error code %ERILU if the argument block is not exactly two words long.
13	.FORNM	Renames the file described by the RENAME block. The LOOKUP/ENTER block pointer and the RENAME block pointer are required for the .FORNM function, unless the file is already open on the specified channel. However, if a file is open on the channel specified in .FOFNC, and if you set the flag FO.UOC in the same word, then the right half of .FOLEB is ignored, and the function is performed on the open file.
14	.FODLT	Deletes the file described by the LOOKUP block. Pointers to both LOOKUP and RENAME blocks are required for this function, unless you set the flag FO.UOC, and a file is open on the channel specified in .FOFNC. In this case, the right half of .FOLEB is ignored and the function is performed on the open file.
15	.FOPRE	Preallocates space for the file described by the LOOKUP/ENTER block. This function is most useful for batch jobs. If a preallocated file is entered but not written, the space is still allocated; a CLOSE for the file will not deallocate the space. If the file is entered immediately after being preallocated, it is not superseded; any subsequent ENTER to the file will supersede it. The LOOKUP/ENTER block pointer is required for the .FOPRE function.
16	.FOSIO	Opens a device for super-I/O (refer to the SUSET. UUO). The first four words of the argument list are required for this function. This function does not require .FOLEB.

Code	Symbol	Function
17	.FOINP	Performs INPUT monitor call. Reads data from the file opened on the specified channel. The argument list is: addr: XWD channo,.FOINP addr1 addr2 where: addr1 is the address of the next buffer to be used in non-dump I/O, or the address of the dump mode command list if using dump I/O. This word is optional for non-dump I/O. addr2 is the optional address of a word containing the block number of the file to perform a USETI to before writing. The I/O status bits are returned in the **ac**.
20	.FOOUT	Performs OUTPUT monitor call. Writes data to the file opened on the specified channel. The argument list is: addr: XWD channo,.FOOUT addr1 addr2 where: addr1 is the address of the next buffer to be used in non-dump I/O, or the address of the dump mode command list. addr2 is the optional address of a word that contains the block number of the file to perform a USETO to before reading. .FOIOS is optional for buffered I/O modes. The I/O status bits are returned in the ac.
21	.FOSET	Performs SETSTS monitor call. The format of the argument list is: addr: XWD channo,.FOSET EXP setsts-bits This function returns error code %ERILU if the argument block is not exactly two words long.
22	.FOGET	Performs GETSTS monitor call. The I/O status bits are returned in the ac. The format of the argument list is: addr: XWD channo,.FOGET
23	.FOREL	Performs RELEAS monitor call. The format of the argument list is: addr: XWD channo,.FOREL
24	.FOWAT	Waits for I/O to finish. The format of the argument list is: addr: XWD channo,.FOWAT
25	.FOSEK	Obsolete.

Code	Symbol	Function
26	.FORRC	Rewrites the RIB of a file if it has changed. This function is ignored and the normal return is taken if the channel is not a disk or if the RIB has not changed. The argument list for this function is: addr: XWD channo,.FORRC
27	.FOGTF	Returns the block number of the first file on a DECtape. If the device on the channel is not a DECtape, the ac is not changed. This duplicates the UGETF call, but allows you to use extended channel numbers. The argument list for this function is: addr: XWD channo, .FOGTF
3Ø	.FOMTP	Performs the function of an MTAPE. call, but allows you to use extended channel numbers. The MTAPE. code is included in the FILOP. argument list as shown: addr: XWD channo,.FOMTP EXP n In the argument list shown here, the value of n is equivalent to the MTAPE. code for the function to be employed. For example, EXP 1 would perform the MTREW. function. This function returns error code %ERILU if the argument block is not two or more words long.
31	.FOUTP	Clears a DECtape directory. Duplicates the UTPCLR call. The argument list for this function is: addr: XWD channo, .FOUTP This function returns the ac unchanged if successful.
32	.FORAW	Renames the file with the specified number of words for allocation. Same function as .FORNM, but allocation in words is specified in .RBSIZ of extended RENAME argument block.
33	.FOFIL	Returns the file specification of the file that is open on this channel. To return the file specification as well as to perform another function, include Word 1Ø (.FOFSP) in the argument block instead of using the .FOFIL function. The argument list for this function is: addr: XWD channo,.FOFIL XWD len,addr2 where the second word contains the length (len) and address (addr2) of the block where the file specification should be stored.

Code	Symbol	Function
34	.FOFXI	Performs an IN monitor call, using extended addressing and dump mode I/O. The argument list is: addr: XWD channo, .FOFXI addr1 addr2 where: channo is the channel number from which data is read from the opened file. addr1 is the address of the command list, which has a two-word format. The first word of each command word pair contains the length of the command list. The second word of each command word pair holds the address where I/O should be performed. If the length is zero, the address in the second word is the location of the next command list. When both the length and the address are zero, the end of the list has been encountered. addr2 is the (optional) address of a word containing the block number of the file to perform a USETI to before reading. The I/O status bits are returned in the ac.
35	.FOFXO	Performs an OUT monitor call, using extended addressing and dump-mode I/O. The argument list is: addr: XWD channo, .FOFXO addr1 addr2 where: channo is the channel number on which data is written to the opened file. addr1 is the address of the command list. Command list format is described above in .FOFXI. addr2 is the (optional) address of a word containing the block number of the file to perform a USETO to before writing. The I/O status bits are returned in the ac.

The file specification is returned when you use function .FOFIL or when you specify an address in .FOFSP (offset 1Ø into the argument block). For .FOFSP, the following data block is returned at the address you specify in the right half of the word. For .FOFIL, this data is returned in the argument block at addr2.

Offset	Symbol	Contents
Ø	.FOFND	Reserved for use by DIGITAL.
1	.FOFDV	Device name.
2	.FOFFN	File name.
3	.FOFEX	File extension.
4	.FOFPP	PPN.
5	.FOFSF	First SFD.
6-1Ø		Subsequent levels of SFDs.

Note that words 5 through 1Ø are returned only where appropriate, and that the entire block is ended by a zero word. When you reserve the block for the file specification, be sure to include space for this zero word.

Multiple channels of a single job and/or multiple jobs can update a file simultaneously using FILOP. The monitor imposes no restrictions or interlocks when a file is being simultaneously updated. Therefore, users must ensure that separate jobs do not update the same block of the same file at the same time. The ENQ/DEQ Facility (refer to Chapter 8) may be used to ensure that such interference does not occur, but the monitor does not require its use when simultaneously updating a file.

To update a file simultaneously, your program performs a FILOP. monitor call using function code 5 (.FOMAU). A file can be updated in this manner when the file is idle, when it is being read, or when it is being updated by other jobs. A file cannot be simultaneously updated if the file is in single-access update mode; that is, when a LOOKUP and an ENTER have been performed or a FILOP. has been performed with function code 4 (.FOSAU) or 6 (.FOAPP).

Note that even though an extended LOOKUP/ENTER/RENAME block can be specified by the FILOP. monitor call, your program cannot change the file attributes of a simultaneously updated file. The current file 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. uses the LOOKUP values on the ENTER.

In order to prevent excessive monitor overhead, files that are to be simultaneously updated should be pre-allocated into contiguous blocks, if possible. This will prevent the creation of inefficient retrieval pointers, and will lessen the chance that extended RIBs will be created.

Normal Return

The requested function has been performed.

Error Return

Error codes are returned in the ac for the FILOP. call. If -1 is returned in the ac, an invalid argument list was supplied. Other error codes are identical to those used by LOOKUP/ENTER. These are listed in Section 11.14, in Volume 1. Note that several functions return the I/O status word in the ac.

Examples

See Chapter 11, Monitor Calls Manual, Volume 1.

Related Calls

CLOSE, ENTER, GETSTS, IN/INPUT, LOOKUP, MTAPE, OPEN, OUT/OUTPUT, PATH., RELEAS, RENAME, SETSTS, SUSET., UGETF, USETI/USETO, UTPCLR, WAIT

FRCUUO [CALLI 1Ø6]

Function

Forces a monitor command for a job or a terminal. This monitor call requires JP.POK, JACCT, or [1,2] privileges.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        FRCUUO  ac,
          error return
        normal return
        . . .
addr:   SIXBIT /command/
        {XWD Ø,jobno}    ;optional arguments
        {XWD Ø,udx  }
```

where: **len** is the length of the argument list. If you give a zero len, the default is 1.

addr is the address of the argument list.

command is the name of a command (from the list below).

jobno is the number of a logged-in job. If you omit the jobno, or specify it as zero, the current job is assumed.

udx is the Universal Device Index for the terminal.

The names of the commands that can be forced are:

Command	Meaning
.BPT	Forces a DDT breakpoint trap, simulating <CTRL/D>.
.BYE	Detaches the job, this command is forced when a dataset disconnects.
.FCONT	Continues the job; this command is forced when a job is continued after it was halted by "Waiting for operator action." (Refer to JCONTINUE monitor command in the Commands Manual.)
.HALT	Stops the job; this command is forced when you type CTRL/C.
.HELLO	Connects (greets) the job; this command is forced when a dataset or network connect occurs, and runs INITIA.
.NETLD	Invokes execution of the program which does automatic down-line loading for ANF-1Ø series remote software.
.RESTA	Greets the job but does not run INITIA.
.TYPE	Types the current input buffer; this is equivalent to typing CTRL/R.

FRCUUO [CALLI 1Ø6]

Command	Meaning
HALT	Stops the job (regardless of CTRL/C trapping).
INITIA	This command is forced when the system is initialized and is used to run INITIA for certain terminals (this is controlled by MONGEN).
KJOB	Kills the job; this command is used to force a job to terminate.
USESTA	Types status information; this is equivalent to typing either CTRL/T or the USESTAT command.

Normal Return

The command is executed; the ac is unchanged.

Error Return

The ac is cleared.

Examples

```
        MOVE    T1,[XWD 2,ADDR]
        FRCUUO  T1,
          JRST  FRCERR
        JRST    CONTIN
ADDR:   SIXBIT  /.TYPE/
        XWD     Ø,Ø
```

This code sequence displays the contents of the terminal input buffer for the current job, as though the user had typed <CTRL/R>.

GETLCH [TTCALL 6,]

Function

Returns the line characteristics for a terminal line.

Calling Sequence

```
        GETLCH  addr
        return
        . . .
addr:   XWD     Ø,lineno
```

where: addr is the address of the argument list.

lineno is the line number for the terminal whose characteristics are required. If you use -1 for addr, the monitor returns characteristics for your job's controlling terminal.

Normal Return

If the job is detached and addr contained -1, the monitor returns a Ø word. On a normal return, the monitor returns the terminal's UDX in the right half of addr (.UXTRM + lineno).

The following line characteristics are returned in the left half of the word at addr:

Bit	Symbol	Characteristic
Ø	GL.ITY	Pseudo-terminal (PTY).
1	GL.CTY	Operator's terminal (CTY).
2	GL.DSP	Display console (DIS).
3	GL.DSL	Dataset line.
4	GL.CNE	No characters are echoed.
5	GL.HDP	Half-duplex line.
6	GL.REM	Remote terminal.
7	GL.RBS	Remote batch terminal.
8-9		Reserved for use by DIGITAL.
1Ø	GL.8BM	Terminal is open in 8-bit I/O mode.
11	GL.LIN	User has typed some input.
12	GL.SLV	TTY SLAVE is in effect.
13	GL.LCM	Terminal in lowercase mode.
14	GL.TAB	Terminal has tab capability.
15	GL.LCP	Local copy only (no echo).

GETLCH [TTCALL 6,]

Bit	Symbol	Characteristic
16	GL.PTM	Papertape mode is on (CTRL/Q, CTRL/S, and so forth, control papertape motion instead of terminal output).
17	GL.NEC	Terminal is in no-echo mode. This characteristic is set by setting IO.SUP in the OPEN call, or by SETSTS, or by TRMOP function .TOCOB. This setting is overridden when the job goes to monitor level, and echoing resumes. You can clear this bit using a RESET call.

If you use an invalid line number, the monitor returns Ø in the left half of the word at addr.

Related Calls

GETLIN, SETLCH, SETSTS, TRMOP., TTCALL

Common Errors

Typing a comma after addr.

GETLIN [CALLI 34]

Function

Returns the SIXBIT monitor-assigned name of the terminal attached to your job.

Calling Sequence

```
GETLIN  ac,
return
```

Return

The SIXBIT name of the terminal is in the ac, left-justified, in the form TTYnnn, where nnn is the dynamic terminal number associated with your job's terminal.

If your job is not attached to any terminal, the ac contains:

```
XWD     Ø,'nam'
```

Where: nam is the right half of the name of the terminal to which your job was last attached (that is, nnn in TTYnnn).

Examples

```
GETLIN  T1,             ;Get terminal name
TLNN    T1,-1           ;Job detached?
JRST    NOTTY           ;Yes
  . . .                 ;No
```

This sequence gets the name of the terminal for the job and checks whether the job is currently detached.

GETPPN [CALLI 24]

Function

Returns the project-programmer number (PPN) for your job.

Calling Sequence

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

Where: alternate return is taken if your program has the JACCT bit set and another job is logged in under the same PPN.

The GETPPN monitor call returns the project number in the left half of the ac, and the programmer number in the right half of the ac.

Examples

```
GETPPN  T1,
JFCL
MOVEM   T1,MYPPN
```

This code gets the PPN regardless of whether the program is JACCTed.

Related Calls

OTHUSR

Common Errors

Forgetting the sequence of normal return followed by alternate return.

GETSEG [CALLI 40]

Function

Replaces the current program high segment with a given high segment. Refer to Chapter 2 for specific information about the implementation of this call and the state of memory during the GETSEG operation.

Calling Sequence

```
        MOVEI   ac,addr
        GETSEG  ac,
          error return
        normal return
        . . .
addr:   SIXBIT/device/
        SIXBIT/filename/
        SIXBIT/extension/
        EXP     Ø
        XWD     proj,prog       ;or PATH. pointer
        XWD     Ø,addr2         ;core argument
        XWD     -1,addr3
```

where: **addr** is the address of the 6-word argument block.

addr2 is the address of a PATH block.

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

filename specifies the name of the file containing the new high segment.

extension specifies the extension of the file containing the new high segment.

proj,prog is the project-programmer number for the directory in which the high segment file resides. If you specify this word to be zero, your default directory path is assumed. If you specify this word as [XWD Ø,addr3], addr3 points to a PATH. block containing the full path specification for the file. Refer to the description of the PATH. monitor call for the format of a PATH. block.

The core argument word is optional. If it is zero, the high segment is placed into the current PC section. Otherwise, addr3 is the address containing the section number where the high segment will be placed.

The GETSEG monitor call allows your program to initialize a high segment from a file or from a currently-loaded sharable segment without affecting your program's low segment. This facility can be used for shared data segments, shared program overlays, and runtime routines (such as FORTRAN and COBOL object-time systems). On KL processors, if the high segment obtained by the GETSEG monitor call is an execute-only segment, it is a concealed high segment.

You can give zeros for any argument except the file name or device. The defaults are:

extension	.EXE
PPN	default directory path

Normal Return

The monitor replaces the current high segment with the given high segment.

NOTES

If the given file contains both a high and a low segment, the monitor brings in only the high segment.

The contents of the accumulators are not preserved (this aspect varies from monitor version to monitor version).

The left half of .JBHRL is cleared.

The right half of .JBHRL is set to the new highest legal user address in the high segment.

.JBSA and .JBREN are cleared if they contain addresses in the new high segment. This removes the program's start address, so that an error will occur on a START or REENTER command.

Channel 0 is released by the GETSEG call. Other channels are not released. Refer to the RELEAS UUO.

A GETSEG call made from the current program's high segment can succeed only if the start of the new high segment coincides with the normal return for the call. Program execution returns to the user program at the PC corresponding to the normal return from the GETSEG UUO in the previous segment. It is the user's responsibility to ensure that this PC contains instructions he wishes to be executed.

Error Return

See Section 11.14 a list of GETSEG errors.

Related Calls

MERGE., RELEAS, RUN

Common Errors

- Forgetting to save the acs over the GETSEG.
- Forgetting that channel 0 is destroyed.
- Forgetting that a GETSEG from a high segment returns control to the PC in the new high segment.

GETSTS [OPCODE Ø62]

Function

Returns the I/O status bits for a device. Use FILOP. to perform GETSTS on an extended I/O channel. The specific I/O status bits for each device are listed in Volume 1 in the chapter specific to the device.

Calling Sequence

```
        GETSTS channo,addr
        return
        ...
addr:   BLOCK  1
```

where: **channo** is the channel number of the channel for which the I/O status word is desired.

addr is the address of the word to receive the I/O status word.

Return

The monitor returns the I/O status bits in the right half of the word at **addr**, and the data mode for I/O in the left half of the word at **addr**. The I/O status bits that are possible are:

Bits	Symbol	Meaning
18-21	IO.ERR	Bit mask for device-independent I/O error flags.
18	IO.IMP	Software detected improper data mode, or checksum error occurred.
19	IO.DER	Device error. Refer to specific device for cause of this error.
2Ø	IO.DTE	Data error.
21	IO.BKT	Block too large, quota exceeded, or file structure is full.
22	IO.EOF	End of file was reached.
23	IO.ACT	Device is active.
24-29		Device-dependant error flags. These are listed for each device in the appropriate chapter in Volume 1.
3Ø	IO.SYN	Synchronous mode I/O.
31	IO.UWC	Use user's word count.
32-35		Data mode of the I/O, indicated by one of the codes that are listed in Table 11-2 in Volume 1.

Related Calls

CLRST., ERLST., FILOP., SENSE., SETSTS, STATO, STATZ

Common Errors

- Forgetting that there is only one return from the call.
- If you give a nonexistent or uninitialized channel number, the monitor stops your job and prints the following message on your terminal:

  ?I/O to unassigned channel at user PC address

  where address gives the program counter for your job at the time of the failure.

GETTAB [CALLI 41]

Function

Returns a word from one of the monitor's GETTAB tables, allowing your program to read many types of job and system information. The GETTAB tables are listed in Chapter 23.

Calling Sequence

```
MOVE    ac,[XWD index,table]
GETTAB  ac,
  error return
normal return
```

where: index is an index into the specified table. If the table is indexed by job number, you can use -1 to obtain information about your own job.

If the table is indexed by job number or segment number, you can use -2 to return information about your own high segment.

table gives the number of the GETTAB table.

Normal Return

The requested word from the table is returned in the ac.

Error Return

The index or the table number was invalid.

Examples

See Chapter 23 for examples.

GOBSTR [CALLI 66]

Function

Returns file structure names from the search list for a job or for the system.

To use the GOBSTR call for a job other than your own, you must have either the JP.SPA privilege or the JP.SPM privilege set in your .GTPRV word, or you must have JACCT privileges, or the job must be logged into [1,2].

For a discussion of file structures in a search list, see the SETSRC program in the TOPS-10 User Utilities Manual.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        GOBSTR  ac,
          error return
        normal return
        . . .
addr:   EXP     jobno           ;.DFGJN
        XWD     projno,progno   ;.DFGPP
       {EXP     -1              ;.DFGNM for first in list
       {EXP     0               ;.DFGNM for first after FENCE
       {SIXBIT/structure/       ;.DFGNM for next in list
        EXP     0               ;.DFGDR
        BLOCK   1
```

where: **len** is the length of the argument list.

addr is the address of the argument list.

addr+2 contains the structure name, or 0, or -1. Therefore you can begin with the first name in the list by using -1 at addr+2; then when the monitor returns the first name in the list, you can leave the name in addr+2 to call for the second name, and so forth. If the next item in the list is FENCE, the monitor returns 0. If there are no more items in the list, the monitor returns -1.

jobno is the number of a logged-in job (use -1 for the current job; use 0 for the system search list).

projno,progno is a project-programmer number (PPN).

structure is the SIXBIT name of a file structure.

I/O status bits are returned at addr+4 as follows:

Bits	Symbol	Meaning
0	DF.SWL	If on, software write-protect is set.
1	DF.SNC	If on, creation of files is not allowed on this structure, unless the structure name is explicitly included in the file specification. Refer to Chapter 12 for more information.

Normal Return

The monitor returns the required SIXBIT structure name (or Ø or -1) at addr+2, and the I/O status word at addr+4.

Error Return

The monitor returns one of the following error codes in the ac:

Code	Symbol	Meaning
3	DFGIF%	File structure name is not Ø, -1, or a file structure name in SIXBIT.
6	DFGPP%	The specified job number and project-programmer number do not correspond.
1Ø	DFGNP%	Your job is not privileged.
12	DFGLN%	The specified length of the argument block is invalid.

Examples

The following code reads all the structures in the job search list.

```
        MOVEI   T1,Ø                    ;Initialize counter
LOOP:   MOVE    T2,[.DFGST+1,,ADDR]
        GOBSTR  T2,                     ;Get next structure
          JRST  ERROR
        MOVE    T2,ADDR+.DFGNM          ;Get structure name
        MOVEM   T2,STRTAB(T1)           ;Save in table
        AOJE    T2,CONTIN               ;Last one if -1
        AOJA    T1,LOOP                 ;Bump table pointer and loop
ADDR:   EXP     JOBNO                   ;Job number
        XWD     PROJ,PROG               ;PPN
        EXP     -1                      ;Get first one in list
        EXP     Ø
        EXP     Ø
STRTAB: BLOCK   3Ø                      ;Space to store search list
```

Related Calls

DSKCHR, JOBSTR, STRUUO, SYSSTR

GTNTN. [CALLI 165]

Function

Returns the node number and line number for a terminal. This call is applicable to network systems only.

Calling Sequence

```
{MOVE    ac,[SIXBIT/terminal-name/]}
{MOVEI   ac,udx                    }
{MOVEI   ac,channo                 }
 GTNTN.  ac,
   error return
 normal return
```

where: **channo** is the channel number of the channel to which the terminal is connected.

udx is the Universal Device Index for the terminal.

terminal-name is the monitor-assigned name of the terminal returned when you use the GETLIN monitor call.

Normal Return

The monitor returns the node number and the line number in the **ac** in the form:

```
node-number,,line-number
```

The **node-number** is the number of the node at which the specified terminal is located. The **line-number** on non-network systems is equivalent to the terminal number. On a network system, **line-number** is the physical line number of the terminal on the node to which the terminal is connected.

Networked terminals are assigned logical line numbers from a pool of network terminal numbers when they connect to a host. Therefore, the logical line number will change as the particular node to which the terminal is attached comes on-line, and as the terminal connects to, and disconnects from a host.

Error Return

One of the following error codes is returned in the **ac**:

Code	Symbol	Error
Ø	NTNSD%	Nonexistent device.
1	NTNAT%	Device is not a terminal.
2	NTTNC%	Terminal is not connected.

Related Calls

GTXTN., NETOP.

GTXTN. [CALLI 166]

Function

Returns the physical name of the terminal for a given node and line number. This call applies to network systems only.

Calling Sequence

```
MOVE    ac,[XWD nodeno,lineno]
GTXTN.  ac,
  error return
normal return
```

Where: **nodeno** is the node number for a terminal.

lineno is the physical line number for the terminal at the node.

Normal Return

The physical name of the terminal is returned in **ac** in the form:

```
SIXBIT/name/
```

Where: **name** is the physical name of the terminal (such as TTY427).

Error Return

One of the following error codes is returned in the **ac**:

Code	Symbol	Error
Ø	XTUNT%	Unknown terminal (node number or the line number specified is not known or node or line is not connected to the DECsystem-1Ø).
1	XTNLT%	Not a legal terminal.

Related Calls

GTNTN., NODE.

HIBER [CALLI 72]

Function

Suspends execution of the job until a specified event occurs.

Calling Sequence

```
MOVE    ac,[flags+sleeptime]
HIBER   ac,
  error return
normal return
```

Where: **flags** specify conditions described below.

sleeptime gives the amount of time for the job to sleep. If HB.SEC is set in flags, the sleeptime is specified in seconds; otherwise, it is specified in milliseconds.

The sleeptime is rounded upward to the next larger jiffy, with a maximum of 262 seconds. If you set HB.SEC, the maximum sleeptime is about 72 minutes (at 60 Hz) or 87 minutes (at 50 Hz). If you need a longer sleeptime, use the PITMR. UUO, or the .CLOCK function of the DAEMON UUO. If you give the sleeptime as 0, the job sleeps until awakened by one of the specified events, or by a WAKE monitor call.

If your job is hibernating, it can be woken by another job if that job has sufficient privileges. Refer to the WAKE UUO.

To prevent your job from oversleeping and missing an event, the monitor sets the wakeup bit even if the job is already awake. You can use another HIBER call to clear the bit. You cannot assume that any of the specified events actually occurred to WAKE your job; therefore you should test for all the events that may have caused your job to awaken, and explicitly execute another HIBER call if you were WAKEd unexpectedly.

You can also clear the wake-enable bit for your job by using the RESET monitor call. Note that until the first HIBER call is executed, there is no protection against wakeup commands from other jobs. To guarantee your job's protection, you should execute a WAKE monitor call for your job, followed by a HIBER call giving the protection you want. The HIBER will return immediately, having set the protection codes as desired.

The bits and their meanings are:

Bits	Symbol	Meaning
0	HB.SWP	Clear the in-core protect time, making the job available for swapping out.
1	HB.SEC	The sleeptime is specified in seconds.
9	HB.DIN	When set in conjunction with HB.RTL and/or HB.RTC, enables the JB.UHI bit in JOBSTS, which allows terminal input from programs such as BATCON and OPR. The job is awakened on input to the terminal.

Bits	Symbol	Meaning
1Ø	HB.IPC	Wake the job when an IPCF packet is placed in its input queue.
11	HB.RIO	Wake the job when asynchronous I/O is completed.
12	HB.RPT	Wake the job for PTY activity.
13	HB.RTL	Wake the job when a line of terminal input is typed on any terminal assigned to your job, or if there is a rescanable line available on the job's controlling terminal.
14	HB.RTC	Wake the job when a character of terminal input is ready.
15	HB.RWJ	Wake the job only on a WAKE monitor call from the job itself. Setting this bit prevents other jobs from waking your job, unless the other job is privileged.
16	HB.RWP	Wake the job only on a WAKE monitor call from a job having the same programmer number.
17	HB.RWT	Wake the job only on a WAKE monitor call from a job having the same project number.

Normal Return

When an enabled HIBER condition occurs, execution resumes at the normal return.

Error Return

The HIBER call takes the error return only if it is not implemented on your system.

Examples

```
        MOVSI   T1,(HB.RWP+HB.RWT)
        HIBER   T1,
          JRST  ERROR
```

This code sequence causes the job to sleep until awakened by a WAKE monitor call from another job having the same project-programmer number. See also RTTRP call.

Related Calls

SLEEP, WAKE

Common Errors

- Forgetting to protect against WAKEs from other jobs.
- Assuming a particular event woke your job, without actually checking.

HPQ [CALLI 71]

Function

Places your job in, or removes your job from a high-priority scheduler queue.

You cannot use HPQ unless your system administrator has set the privilege value JP.HPQ to a nonzero value. This value is the highest priority queue you can request. This monitor call is primarily intended for real-time programs where fast response time is critical. Refer to Chapter 9 of the Monitor Calls Manual, Volume 1, for more information.

Calling Sequence

```
        MOVEI   ac,queue
        HPQ     ac,
          error return
        normal return
```

Where: queue is the number of the required high priority queue. The lowest queue number is 1; the highest is a system parameter. If you give queue as Ø, your job returns to the normal scheduler queue.

Normal Return

The monitor places your job in the given queue.

Error Return

The ac contains -1; you gave an illegal value for queue or you are not a privileged user.

Related Calls

RTTRP, TRPSET, UJEN

IN [OPCODE Ø56]

Function

Reads data from an initialized channel into memory. Use FILOP. to perform an IN for an extended I/O channel.

Calling Sequence

```
IN channo,addr
normal return
  error return
```

where: **channo** is the number of an initialized I/O channel.

addr is one of the following:

- If the channel was initialized for dump mode, then **addr** gives the address of an I/O command list.
- If the channel was initialized for buffered mode, then addr gives the address of the second word of the next buffer to be used; if you give Ø (the normal case), the next buffer in the ring is used.

Note that the return locations for this call are in the reverse order from the convention for other calls, because the normal return follows the calling instruction and the error return follows the normal return.

Normal Return

Data is input from the channel.

Error Return

The monitor found an end-of-file mark or errors in the data (reflected in the I/O status word). If using non-blocking I/O mode, the error return could indicate no available data. This is indicated by no error bits set in the I/O status word. Use the GETSTS call to read the I/O status bits.

Examples

See LOOKUP call.

Related Calls

FILOP., INPUT, OUT, OUTPUT

Common Errors

- If the channel was not initialized, the monitor stops the job and prints:

  ?I/O to unassigned channel at user PC xxxxx

- If the specified address is illegal, the monitor stops the job and prints:

  ?Address check for device yyyyyy:UUO at user PC xxxxx

- If the monitor cannot allocate buffers in your address space, the monitor stops the job and prints (see INBUF):

  ?Address check for device yyyyyy:UUO at user PC xxxxx

INBUF [OPCODE Ø64]

Function

Sets up an input buffer ring with the specified number of buffers for a given initialized channel. Use FILOP. to perform an INBUF on an extended I/O channel.

NOTE

Buffers are allocated by the monitor in the user's address space starting at the location pointed to by the contents of .JBFF. This symbol represents a word in the Job Data Area. As the JDA exists only in Section Ø, you cannot initialize a buffer in a non-zero section, unless that section is mapped to section Ø. Use the FILOP. monitor call to specify buffer starting addresses in a non-zero section.

Calling Sequence

```
INBUF channo,buffers
return
```

where: **channo** is the number of an initialized channel.

buffers is the number of buffers to set up in the ring. For disk devices, if you give **buffers** as Ø, the monitor uses the value given in the SET DEFAULT BUFFERS monitor command or SETUUO. If no value has been set, the system default (a MONGEN parameter) is used. For non-disk devices, 2 buffers are assumed.

Normal Return

The buffer ring is set up.

Related Calls

FILOP., OUTBUF

Common Errors

- If the channel was not initialized, the monitor stops the job and prints:

  ?I/O to unassigned channel at user PC xxxxx

- If the monitor cannot allocate buffers in your address space, the monitor stops the job and prints:

  ?Address check for device yyyyyy:UUO at user PC xxxxx

- If your program tries to use INBUF or OUTBUF to create buffers outside the job's core image, the job cannot expand because the system runs out of virtual memory and the monitor stops the job and prints:

  ?Illegal address in UUO at user PC xxxxx

- If you use INBUF or OUTBUF to set up a buffer ring in a non-zero section, the monitor stops the job and displays the following error message:

  ?Illegal INBUF/OUTBUF for device yyyyyy;UUO at user PC xxxxx

INCHRS [TTCALL 2,]

Function

Reads an ASCII character from the job's controlling terminal's input buffer, skipping on return if a character was available. INCHRS also sets "character mode," in which the program will not wait for the end of the line of input from the terminal. Therefore, CTRL/U, DELETE, and other line-editing characters will not function as they do for the monitor. See Chapter 15 for more specific information.

Calling Sequence

```
        INCHRS  addr
        return 1 ;no character in buffer
        return 2 ;character read from buffer
        ...
addr:   BLOCK 1
```

where: addr is the address of a word to contain the input character (in bits 29 to 35); the rest of the word is cleared.

Return

If a character has been input, the monitor copies it to bits 29 to 35 of the word at addr.

Related Calls

TRMOP., TTCALL

Common Errors

Typing a comma after addr.

INCHRW [TTCALL Ø,]

Function

Inputs an ASCII character from the terminal's input buffer. The monitor waits for a character if none is available. INCHRW inputs the character regardless of whether a complete line has been typed. If the program is not prepared to handle every possible control character, you should consider using the INCHWL call instead of INCHRW.

Calling Sequence

```
INCHRW addr
return
```

where: **addr** is the address of the word to receive the ASCII input character. The character is placed in bits 29-35 of the word; the rest of the word is cleared.

If no character has been input, the monitor waits for a character.

Return

If a character has been input, the monitor places the character in bits 29-35 of the word at **addr**.

Related Calls

TRMOP., TTCALL

Common Errors

Typing a comma after **addr**.

INCHSL [TTCALL 5,]

Function

Inputs a character in line mode from the terminal's input buffer, skipping on return if the input was terminated by a line break character such as carriage-return/line-feed.

Calling Sequence

```
INCHSL addr
return 1
return 2
```

where: **addr** is the address of the word to receive the character (in bits 29 to 35); the rest of the word is cleared.

return 1 is the return instruction when a line break has not been input from the terminal

return 2 is the return instruction when a line break character has been input from the terminal.

Return

If a line break has been input from the terminal, the monitor returns at **return 2** with the next character of the line in bits 29-35 of addr; if not, it returns at **return 1**.

Related Calls

TRMOP., TTCALL

Common Errors

Typing a comma after **addr**.

INCHWL [TTCALL 4,]

Function

Inputs a character from the terminal input buffer, waiting until a break character is encountered. With this type of input, the monitor handles line-editing characters like DELETE, CTRL/R, and so forth.

See Chapter 15, Monitor Calls Manual, Vol. 1 for a discussion of break characters.

Calling Sequence

```
INCHWL addr
return
```

where: addr gives the address of the word to contain the input character; the rest of the word is cleared.

Return

The character is right-justified in the word at addr. The rest of the word is cleared.

Related Calls

TRMOP., TTCALL

Common Errors

Typing a comma after addr.

INIT [OPCODE Ø41]

| Obsolete; use OPEN or FILOP..

INPUT [OPCODE Ø66]

Function

Inputs data from an initialized channel to memory. Use FILOP. to perform an INPUT on an extended I/O channel. INPUT is the same as IN, except INPUT does not give an error return if an error or EOF condition occurs. The user must check for such conditions with GETSTS, STATZ, or STATO.

NOTE

Programs doing non-blocking I/O should use the IN monitor call or FILOP. function .FOINP.

Calling Sequence

```
INPUT channo,addr
return
```

where: **channo** is the number of an initialized channel.

addr is one of the following:

- If the channel is initialized for dump mode, then **addr** gives the address of an I/O command list.
- If the channel is initialized for buffered mode, then **addr** gives the address of the second word of the next buffer to be used; if you give Ø (the default), the next buffer in the ring is used.

Return

Data is input from the channel.

Related Calls

FILOP., IN, OUT, OUTPUT

Common Errors

- If the channel was not initialized, the monitor stops the job and prints:

  ?I/O to unassigned channel at user PC xxxxx

- If the specified address is illegal, the monitor stops the job and prints:

  ?Address check for device yyyyyy:UUO at user PC xxxxx

- If the monitor cannot allocate buffers in your address space, the monitor stops the job and prints:

  ?Address check for device yyyyyy:UUO at user PC xxxxx

IONDX. [CALLI 127]

Function

Returns the Universal Device Index (UDX) for a device or channel. For information about terminal names and their UDXs, refer to the TRMNO. UUO.

Calling Sequence

```
{MOVE   ac,[SIXBIT/device/]}
{MOVEI  ac,channo          }
 IONDX. ac,
   error return
 normal return
```

where: **device** is the SIXBIT physical or logical name of a device for which its UDX is desired.

channo is the number of an initialized channel.

Normal Return

The Universal Device Index for the specified device or current device on the specified channel is returned in the ac.

Error Return

If the ac is cleared, you gave a nonexistent device or SIXBIT/MPX/ as a device name.

IONEOU [TTCALL 15,]

Function

Sends an 8-bit image character to the terminal's output buffer.

Calling Sequence

```
IONEOU    addr
return
```

where: addr contains the 8-bit character in bits 28 to 35.

Return

The 8-bit character is output to the terminal.

Related Calls

OUTCHR

Common Errors

Typing a comma after addr.

IPCFM. [CALLI 217]

Function

Communicates with [SYSTEM]INFO and [SYSTEM]IPCC, replacing a message exchange.

Calling Sequence

```
        XMOVEI   ac,addr
        IPCFM.   ac,
          error return
        normal return

addr:   flags dest,,len
        addrl
        optional in-your-behalf process ID (PID)

addrl:  message block
```

where: addr is the address of the packet header block.

flags are one or both of the flags in the packet header block.

dest is the destination PID. len is the length of the packet header block.

The argument block at addr is:

Offset	Symbol	Contents
Ø	.IPCMF	Flags, destination, and length fields, in the following format:

Bits	Symbol	Meaning
Ø	IP.CMP	Invoking privileges. The job must have IPCF privileges to use this bit.
1	IP.CMI	Indirect sender's PID.
2-14		Reserved for DIGITAL.
15-17	IP.CMD	Destination process code, one of the following:

Code	Symbol	Meaning
1	.IPCCC	[SYSTEM]IPCC
2	.IPCCF	System-wide [SYSTEM]INFO
3	.IPCCP	Receiver's [SYSTEM]INFO

18-26		Reserved for DIGITAL.
27-35	IP.CML	Total length of argumnent block, including .IPCMF.

Offset	Symbol	Contents
1	.IPCMP	Pointer to [SYSTEM]IPCC or [SYSTEM]INFO message block detailed below. The pointer may be a 30-bit address or a section address (if an IFIW is given), relative to the section the message block is in. No indexing or indirection is allowed.
2	.IPCMI	In-your-behalf word; the PID on whose behalf to perform this operation, 0 for your own job. If this word is non-zero, IPCF privileges must be enabled or the given PID must belong to your current JCH. If .IPCMI is on, it contains the address (30-bit or IFIW) of the PID.

The message block at addr 1 for [SYSTEM]IPCC is:

Offset	Symbol	Contents
0	.IPCS0	Holds the message length identifier in the left half, and one of the following function codes in the right half:

Code	Symbol	Meaning
1	.IPCSE	Enables IPCF privileges.
2	.IPCSD	Disables IPCF privileges.
3	.IPCSI	Obtains PID of [SYSTEM]INFO for user of PID.
4	.IPCSF	Sets the program specified in .IPCS1 to act as [SYSTEM]INFO for the program specified in .IPCS2. If you do not specify .IPCS2, the system is assumed as the default.
5	.IPCSZ	Deletes PID.
6	.IPCSC	Creates PID for job.
7	.IPCSQ	Sets PID quota.
10	.IPCSO	Changes owner of PID, assign new job number.
11	.IPCSJ	Supplies job number of PID.
12	.IPCSP	Gives PID list for job.
13	.IPCSR	Reads job quota.
16	.IPCQS	Sets PID quota.
17	.IPCQR	Reads PID quota.
23	.IPCLP	Locates PID in system PID table and returns index.
24	.IPCWP	Writes system PID table. (Refer to GETTAB table .GTSID)
25	.IPCRP	Reads system PID table.

1	.IPCS1	First argument.
2	.IPCS2	Second argument.
3	.IPCS3	Third argument.

The message block for [SYSTEM]INFO is:

Offset	Symbol	Contents
Ø	.IPCIØ	The left half holds the message block length; the right half contains one of the following function codes:

Code	Symbol	Meaning
1	.IPCIW	What is PID (ASCIZ).
2	.IPCIG	Gets PID name.
3	.IPCII	Assigns PID name until reset.
4	.IPCIJ	Assigns PID name until logout.
5	.IPCID	Drops specific PID.
6	.IPCIR	Drops names set in .IPCII.
7	.IPCIL	Drops names set in .IPCIJ.
1Ø	.IPCIN	Notifies when the PID is dropped.

1	.IPCI1	First argument.
2	.IPCI2	Second argument.

Normal Return

The system process returns data in a packet to the user's message block.

Error Return

The ac will contain one of the error messages documented under IPCFR. UUO.

Related Calls

.IPCFQ, .IPCFR, .IPCFS

IPCFQ. [CALLI 144]

Function

Returns information about a job's IPCF input queue. The information returned is the packet header block for the next (if any) packet in the queue of packets sent by the inter-process communication facility. The IPCF calls are described in Chapter 7 of Monitor Calls Manual, Volume I.

Calling Sequence

```
        MOVE ac,[XWD len,addr]
        IPCFQ. ac,
          error return
        normal return
        . . .
addr:   BLOCK len
```

where: **len** is the length of the block (4 to 6 words) at **addr** to receive returned data.

addr is the address of the block to receive the data.

Normal Return

The ac is not changed. The packet header block for the next packet in the queue is returned at **addr**. The format of the information returned is described in Chapter 7.

Offset	Symbol	Contents
Ø	.IPCFL	Flag word of the next packet in the queue.
1	.IPCFS	Sender's PID.
2	.IPCFR	Receiver's PID.
3	.IPCFP	Length of next message and number of packets:

Bits	Meaning
Ø-17	Length of next message.
18-35	Number of packets in your input queue.

4	.IPCFU	Sender's PPN.

Offset	Symbol	Contents		
5	.IPCFC	Sender's capability word:		
		Bits	Symbol	Meaning
		Ø	IP.JAC	Sender has JACCT privileges set.
		1	IP.JLG	Sender is logged in.
		2	IP.SXO	Sender is execute-only.
		3	IP.POK	Sender has POKE. privilege (JP.POK).
		4	IP.IPC	Sender has IPCF privilege (JP.IPC).
		5-17		Reserved.
		18-26	IP.SCN	Sender's context number.
		27-35	IP.SJN	Sender's job number.

Error Return

If there is no packet in the input queue, IPCFQ. takes the error return and returns an error code in the ac. The error codes for all IPCF calls are listed under the IPCFR. call.

Related Calls

IPCFM., IPCFR., IPCFS.

IPCFR. [CALLI 142]

Function

Retrieves a packet from the IPCF input queue for the calling process. The IPCF facility and the format of the argument blocks are described in Chapter 7 of Monitor Calls Manual, Volume I.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        IPCFR.  ac,
          error return
        normal return
        . . .
addr:   EXP     flags
        BLOCK   2
        XWD     len2,addr2
        . . .
addr:   BLOCK len2
```

where: len is the length of the packet header block.

addr is the address of the packet header block.

flags is the flag word (.IPCFL) in the packet header block.

len2 is the length of the packet to be retrieved. If the packet is going to be a page of data, this field must contain 512 or an error code is returned in the ac and the IPCFR. monitor call takes the error return.

addr2 is the address of the packet message block.

The retrieving process should check the contents of the flag word (word Ø, .IPCFL). If there is an error condition associated with the packet, it will be indicated in bits 24-29. Error codes received in the ac indicate an error with the monitor call (for example, if the packet was not received).

If the IPCFR. monitor call is issued but there are no packets in the input queue, the job cannot continue executing until a packet is placed in the input queue. To prevent the job from blocking, bit Ø (IP.CFB) should be set in the flag word. When IP.CFB is set and there are no packets in the input queue when the IPCFR. call is issued, the call will take the error return and the monitor will return error code 3 (IPCNT%) in the ac.

When a process is retrieving a page of data, bit 19 (IP.CFV) in the flag word must be set and the length of the data message block (left half of .IPCFP) must contain 1ØØØ or the IPCFR. monitor call fails and the monitor returns error code 21 (IPCPR%) in the ac.

If the retrieved packet is shorter than the number of reserved words (that is, len2), the packet is retrieved and the extra words are left unchanged. If, however, the number of reserved words is not long enough to store the packet, the IPCFR. monitor call takes the error return and the monitor returns error code 5 (IPCTL%) in the ac. To prevent this error, the receiver can set bit 4 (IP.CFT) in the flag word indicating that, if the packet is too long for the reserved space, the message should be truncated.

Normal Return

On a normal return, the monitor returns the associated variable (see Chapter 7) in the ac indicating that there is another packet waiting in the queue. If there are no more packets in the queue, the monitor clears the ac. The packet retrieved from the process' input queue is returned to the address specified in the IPCFR. monitor call (beginning with addr). The packet header block is filled in as follows:

Offset	Symbol	Contents
Ø	.IPCFL	The left half remains the same, the right half contains flags (see Chapter 7).
1	.IPCFS	Sender's PID.
2	.IPCFR	Receiver's PID.
3	.IPCFP	Length and location of data:

Bits	Contents
Ø-17	Message length.
18-35	Address of message, for short-form messages (default), or page number of long-form messages. If the page number refers to a non-existent page, error code IPCAC% is returned.

4	.IPCFU	Sender's PPN. If the argument block length is less than 5, this word is not returned.
5	.IPCFC	Sender's capability word:

Bits	Symbol	Meaning
Ø	IP.JAC	Sending program has JACCT privileges.
1	IP.JLG	Sender is logged in.
2	IP.SXO	Sender is execute-only.
3	IP.POK	Sender has POKE. privilege (JP.POK).
4	IP.IPC	Sender has IPCF privilege (JP.IPC).
5-17		Reserved.
18-26	IP.SCN	Sender's context number.
27-35	IP.SJN	Sender's job number.

If the argument block length is less than 6, this word is not returned.

Error Return

The packet is not retrieved and one of the following error codes is returned in the ac:

Code	Symbol	Error
1	IPCAC%	Address check.
2	IPCNL%	Packet header not long enough.
3	IPCNP%	No packet in receiving queue.
4	IPCIU%	Page is in use (locked in core).
5	IPCTL%	Data too long for user's buffer.
6	IPCDU%	Receiver's PID unknown.
7	IPCDD%	Receiver disabled.
10	IPCRS%	No room in sender's quota.
11	IPCRR%	No room in receiver's quota.
12	IPCRY%	No room in system storage.
13	IPCUP%	Unknown page (send) or duplicate page (receive).
14	IPCIS%	Invalid sender PID.
15	IPCPI%	Not enough privileges.
16	IPCUF%	Unknown function code.
17	IPCBJ%	Illegal job number.
20	IPCPF%	PID table full.
21	IPCPR%	Page requested, normal text.
22	IPCIE%	Paging I/O error.
23	IPCBI%	Bad index for system PID table.
24	IPCUI%	Undefined PID in system table.
25	IPCRU%	Receiver PID unknown or does not match job.
70	IPCFU%	[SYSTEM]INFO has unknown internal error.
71	IPCCF%	[SYSTEM]IPCC request from [SYSTEM]INFO failed.
72	IPCFF%	[SYSTEM]INFO failed to complete an ASSIGN.
73	IPCQP%	PID quota exceeded.
74	IPCBP%	Unknown PID.
75	IPCDN%	Duplicate name.
76	IPCNN%	No such name.
77	IPCBN%	Name has illegal characters.

Examples

An example of the IPCFR. monitor call is shown below.

```
        MOVE   T2,[XWD 6,PHB]      ;Length and address of packet
        IPCFR. T2,                 ; To be retrieved.
          JRST ERR
        JRST NORM

PHB:    EXP 0                      ;No flags
        EXP 0                      ;Sender's PID
        EXP 0                      ;Receiver's PID
        10,,PMB1                   ;Length and address of packet
                                   ;Message block to be retrieved
        EXP 0                      ;PPN of sender
        EXP 0                      ;Capabilities of sender

PMB1:   EXP 0
        EXP 0
        EXP 0
        .
        .
        .
        EXP 0
```

On a normal return from the IPCFR. monitor call, the packet has been retrieved from the input queue. If no packet was in the input queue, the call takes the error return, and error code 3 is returned in the ac. Below is an example of what a response from [SYSTEM]INFO could be after a request for a PID.

```
  PHB   2Ø                ;The packet was sent by [SYSTEM]INFO
PHB+1   2ØØ1              ;[SYSTEM]INFO's PID
PHB+2   31                ;Job number of receiver
PHB+3   4,,PMB1           ;Length and address of packet message
                          ; block
PHB+4   1,,2              ;PPN of sender
PHB+5   26ØØØØ,,1Ø14      ;Capabilities of sender

  PMB1  32,,3             ;User code and function code
PMB1+1  4ØØØØ4,,1ØØ1      ;The requested PID
PMB1+2  ASCIZ/CORP/       ;The symbolic name
PMB1+3  Ø
```

The IPCFR. monitor call can take the normal return and return an error code in the flag word of the packet header block. For example, word Ø of the packet header block could contain the following:

```
    PHB/  Ø52Ø
```

This means that the length of the packet message block specified in the IPCFR. monitor call was not long enough, so the monitor returned error code 5 in the flag word. The 2Ø in the flag word indicates that the message in the receiver's input queue is from [SYSTEM]INFO.

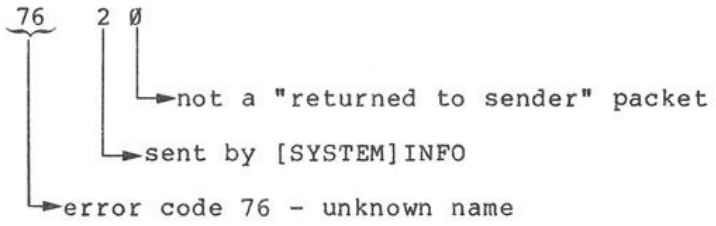
If a process sends a request to [SYSTEM]INFO to obtain the PID associated with the symbolic name "FRED," the following could result:

```
Location  Contents

   AC     Ø                ;indicating a normal return and no
                           ;errors set in the AC; no more packets
                           ;in queue.
  PHB     762Ø             ;the flag word
PHB+1     2,,1ØØ3          ;the sender's PID
PHB+2     164,,1Ø11        ;the receiver's PID
PHB+3     1Ø,,PMB          ;length and addr of message block

  PMB     11,,1            ;user code and function code
PMB+1     Ø                ;no response
PMB+2     ASCIZ/FRED/      ;symbolic name
```

The first word of the packet, PHB, contains 762Ø. This value indicates the following



| IPCFR. [CALLI 142]

The call to [SYSTEM]INFO succeeded, and a normal return was taken. The number of packets still in the queue is stored in the ac.

Error code 76 in the flag word indicates that the symbolic name "FRED" is not associated with any currently assigned PID.

Related Calls

| IPCFM., IPCFQ., IPCFS.

IPCFS. [CALLI 143]

Function

Sends an IPCF packet to the specified process.

By giving the receiver's PID as the PID of [SYSTEM]INFO or [SYSTEM]IPCC, you can obtain information from the IPCF facility itself (see Chapter 7).

Calling Sequence

```
        MOVE   ac,[XWD len,addr]
        IPCFS. ac,
          error return
        normal return
        . . .
addr:   flags
        sender's PID
        receiver's PID
        XWD len,addr2
        . . .
addr2:  message-word-Ø
        .
        .
        .
        message-word-(len-1)
```

where: **len** is the length of the packet header block. The length of this block must be equal to or greater than 4 or the monitor returns error code 2 (IPCNL%) in the **ac**.

addr is the address of the packet header block.

flags is the flag word in the packet header block.

sender's PID is Word 1 of the packet header block.

receiver's PID is Word 2 in the packet header block.

len2 is the length of the packet message block. When sending a short-form message, this value should not exceed 12 octal. The limit may be GETTABed in %IPCML.

addr2 is the address of the packet message block.

message-word-Ø through **message-word-n** are the words making up the packet message block. Refer to Chapter 7 for more information.

Normal Return

On a normal return, the ac is unchanged and the packet described by the packet header block at addr has been placed in the intended receiver's queue.

IPCFS. [CALLI 143]

Offset	Symbol	Contents
Ø	.IPCFL	Flags are the same as those described in Chapter 7.
1	.IPCFS	Sender's PID.
2	.IPCFR	Receiver's PID. If you use the PID for [SYSTEM]INFO or for [SYSTEM]IPCC, you can retrieve information from the IPCF facility itself (see Chapter 7).
3	.IPCFP	Length and location of data:

Bits	Contents
Ø-17	Message length.
18-35	Address of message.

Error Return

On an error return, an error code is returned in the ac and the packet is not sent. The error codes are listed under the IPCFR. call.

Examples

This code fragment sends a packet to [SYSTEM]INFO, asking that a PID be assigned with the symbolic name LJC.

```
        MOVE    T1,[XWD 4,PHB] ;Length and address of packet
        IPCFS.  T1,            ; header block
          JRST ERROR
        JRST NORMAL

PHB:    Ø                      ;This is a packet header
        Ø                      ;Sender's PID
        Ø                      ;Receiver's PID (your [SYSTEM]INFO)
        XWD 3,PMB              ;Length and addr of message block

PMB:    XWD 234,.IPCII         ;Ack code and function (assigns PID)
        Ø                      ;No duplicate PID
        ASCIZ/LJC/             ;Symbolic name
```

Related Calls

IPCFM., IPCFQ., IPCFR.

JBSET. [CALLI 113]

Function

Sets system or job parameters for another job. Your job must have the JACCT bit set, or must be logged in under [1,2]. You can use the SETUUO monitor call to set parameters for your current job.

Calling Sequence

```
        MOVE   ac,[XWD len,addr]
        JBSET. ac,
          error return
        normal return
        . . .
addr:   XWD    Ø,jobno
        XWD    fcn-code,argument
```

where: **len** is the length of the argument list.

addr is the address of the argument list.

jobno is the number of the job for which the SETUUO function is to be performed.

fcn-code is one of the function codes described under SETUUO.

argument is an argument for the given function code.

Refer to the SETUUO description for a list of all function codes and their meanings.

Normal Return

The function has been performed and the **ac** is left unchanged.

Error Return

The error return is taken if the calling job is not privileged, the specified job number is illegal, or the SETUUO function failed.

JOBPEK [CALLI 1Ø3]

Function

Reads or writes another job's core.

To use the JOBPEK call, you must have POKE privileges (JP.POK), your program must have JACCT set, or you must be logged in under [1,2].

Use the Format 1 calling sequence with 18-bit addresses. Use the Format 2 calling sequence if the core being read or written is either in a non-zero section or in a context other than the current one.

Calling Sequences

Format 1:

```
        MOVEI  ac,addr
        JOBPEK ac,
          error return
        normal return
        . . .
addr:   EXP    <flags>+jobnoB17+countB35
        XWD    readaddr,writeaddr
```

where: addr is the address of the argument list.

flags are one or more of the flags listed below.

jobno (JK.JOB) is the number of the logged-in job whose core is to be read or written, stored in Bits 9-17.

count (JK.WCT) is the number of words to be read or written (the maximum can be obtained using GETTAB to read item %CNJPK from table .GTCNF), stored in Bits 18-35.

readaddr is the location of the first word to be read.

writeaddr is the location of the first word to be written.

Format 2:

```
        MOVE   ac,[length,,addr]
        JOBPEK ac,
          error return
        normal return
        . . .
addr:   EXP    <flags>+countB17+JCHB35
        EXP    Ø
        XWD    readaddr
        XWD    writeaddr
```

where: addr is the address of the argument list.

flags are one or more of the flags listed below.

count (JK.EWC) is the number of words to be read or written (refer to GETTAB table .GTCNF, item %CNJPK), stored in Bits 8-17.

JCH (JK.JCH) is the job/context handle of the job whose core is to be read or written, stored in Bits 18-35.

readaddr is the 30-bit address giving the location of the first word to be read.

writeaddr is the 30-bit address giving the location of the first word to be written.

The flags and their meanings are:

Bit	Symbol	Meaning
0	JK.WRT	Write the other job's core; if not set, read the other job's core.
1	JK.UPM	Read the other job's UPMP (user page map page or user page table). JK.WRT must not be set.
2	JK.EVA	Source address is between .MCFV and .UUPMP; treat it as if it were an executive virtual address mapped through the specified job's UPMP. Both JK.WRT and JK.UPM must be off.
3	JK.AIO	Do not block if data is inaccessible (due to the state of cache on SMP systems); set this bit only if you set either JK.UPM or JK.EVA.

Notice that if the other job's core is to be read (JK.WRT is cleared), then **readaddr** is a location in the other job and **writeaddr** is a location in the current program. If the other job's core is to be written (JK.WRT is set), then **readaddr** is a location in the current program and **writeaddr** is a location in the other job.

Normal Return

The specified words are transferred between the other job and the current job.

Error Return

One of the following error codes is returned in the **ac**:

Code	Symbol	Meaning
1	JKNPV%	Job not privileged.
2	JKIJN%	Illegal job number.
3	JKSWP%	Job swapped out or in transit.
4	JKIAD%	Illegal address (source or destination).
5	JKDNA%	Data not addressable (if JK.AIO is set).
6	JKPNC%	Page not in core.
7	JKIOE%	I/O error occurred.
10	JKABZ%	Target address is in an "allocated but zero" page.

Examples

```
        MOVEI     T1,ADDR
        JOBPEK    T1,
          JRST    ERROR
        JRST      CONTIN
        . . .
ADDR:   EXP       14B17+1000B35
        XWD       10000,12000
```

This example reads 1000 (octal) words from the core of job 14 into the current job's core. Reading begins at location 10000 in the other job; writing begins at location 12000 in the current job.

JOBSTR [CALLI 47]

Function

Returns names of file structures in your job's search list. For a discussion of file structures in a search list, see Chapter 11.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        JOBSTR  ac,
          error return
        normal return
        . . .
addr:   SIXBIT/str/             ;.DFJNM
        EXP     Ø               ;reserved
        EXP     Ø               ;.DFJST
```

where: **len** is the length of the argument list (.DFJBL).

addr is the address of the argument list. You can include a structure name (**str**) at addr to obtain the name of the next structure in your job search list, or Ø to obtain the first structure in your active search list, or -1 to obtain the first structure in your job's passive search list (after FENCE in search list returned by SETSRC program).

addr+1 (.DFJDR) is reserved.

At addr+2 (.DFJST), the monitor returns the write-protect flag for the structure. The flags are:

Bits	Symbol	Meaning
Ø	DF.SWL	Software write-protect.
1	DF.SNC	Do not create files on this structure; create only if specified as file structure or a physical device name.

Normal Return

If you give Ø at **addr**, the monitor returns the first structure in the search list after the FENCE.

If you give -1, the monitor returns the first structure in the list.

If you give a SIXBIT structure name (or leave the one the monitor last entered), the monitor returns the next structure name in the search list. When there are no more structures in the list, the monitor returns -1 at **addr**. If the next item in the list is FENCE, the monitor returns Ø.

Therefore you can begin with the first name in the list by using -1 at **addr**. When the monitor returns the first name in the list, you can leave the name in **addr** to call for the second name, and so forth.

JOBSTR [CALLI 47]

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
3	DFGIF%	Illegal file structure name.
12	DFGLN%	Illegal argument length.

Examples

The following example reads all structures in the job's search list:

```
        MOVEI   T1,Ø                    ;Initialize counter
LOOP:   MOVE    T2,[.DFJBL,,ADDR]       ;Pointer to argument block
        JOBSTR  T2,                     ;Get next structure (on Ø or -1)
          JRST  ERROR
        MOVE    T2,ADDR+.DFJNM          ;Get structure
        MOVEM   T2,STRTAB(T1)           ;Save in table
        AOJE    T2,CONTIN               ;All done if -1
        AOJA    T1,LOOP                 ;Bump table pointer and loop
ADDR:   EXP     -1                      ;Start with the first one
        EXP     Ø
        EXP     Ø
STRTAB: BLOCK   3Ø                      ;Where to store search list
CONTIN:   .
          .
          .
```

Related Calls

DVPHY., GOBSTR, SYSPHY, SYSTR

JOBSTS [CALLI 61]

Function

Provides information (including checking statistics) about terminal devices, pseudo-terminals, and software states associated with terminals. For more information about terminals and pseudo-terminals, refer to Chapter 15.

Calling Sequence

```
{MOVEI  ac,udx   }
{MOVEI  ac,channo}
{MOVNI  ac,jobno }
 JOBSTS ac,
   error return
 normal return
```

where: udx is the Universal Device Index of the terminal for which information is desired.

channo is the number of an I/O channel on which a terminal device has been opened.

jobno is the number of a logged-in job associated with the terminal. To obtain status of a pseudo-terminal, provide the job number of the controlled job. Note that the negative of the job number is used because positive values are interpreted as channels or UDXs.

Normal Return

The monitor returns a status word for the job, with the appropriate flags set from the following list.

Bits	Symbol	Meaning
Ø	JB.UJA	The given job number is assigned.
1	JB.ULI	The job is logged in.
2	JB.UML	Terminal is at monitor level.
3	JB.UOA	Terminal output is available.
4	JB.UDI	The terminal is at user level and is in the input wait state, or the terminal is at monitor level and can accept a command. There is no command waiting to be decoded, the job is not running, and the job is not stopped waiting for operator intervention.
5	JB.UJC	JACCT is set for the job. Note that this means that two CTRL/Cs will not stop the job.
6	JB.URN	The job is running. This bit is zero if the job is in a wait state.
7	JB.UFC	The terminal device is in "full character set" mode. This characteristic can be set using the TRMOP. UUO.

Bits	Symbol	Meaning
8	JB.UBK	The terminal device is in "break on all characters" mode. This characteristic can be set using the TRMOP., OPEN, or FILOP. UUOs.
9-1Ø		Reserved for use by DIGITAL.
11	JB.UNE	The terminal device is in "no echo" mode. This characteristic can be set using the TRMOP., OPEN, or FILOP. UUOs.
12	JB.UTO	The terminal is in terminal output state. In other words, the job is blocked waiting for terminal output.
13	JB.UCC	The terminal characteristics have changed since last JOBSTS.
14	JB.UNT	The terminal connected to the pseudo-terminal has used SET HOST to connect to another system.
15	JB.UHI	The terminal is HIBERing for input. If a program such as OPR or BATCON is running under batch, and JB.UHI is set, the job will awaken on input to the terminal. (Refer to the HB.DIN bit in the HIBER monitor call.)
16-26		Reserved for use by DIGITAL.
777B35	JB.UJN	Bit mask to contain job number (Ø if none assigned).

Since JB.UOA will be set if any output is pending, but JB.UTO will be set if the output buffer for the terminal is full, you can make each INPUT UUO transfer more data, by testing for JB.UTO before JB.UOA, then doing on INPUT for a PTY.

Error Return

One of the following occurred:

- The specified job number or channel number is invalid.
- There was no terminal on the specified channel.

KDP. [CALLI 200]

Function:

Loads, dumps, and starts the KMC-11 (KS systems only).

Calling Sequence:

```
        MOVE    ac,[XWD len,addr]
        KDP.    ac,
          error return
        normal return
         ...
addr:   EXP     fcn-code
        argument 1
        argument 2
        argument 3
```

where: len is the length of the argument block.

addr is location of the argument block. At addr, store the function code (fcn-code). The remainder of the argument block depends on the function to be performed.

The function codes are:

Code	Symbol	Meaning
1	.KDPKN	Returns in argument 1 the number of KMC-11s on the system.
2	.KDPDN	Returns in argument 2 the count of DUP-11s on the KMC that you specify in argument 1.
3	.KDPSS	Returns in argument 2 the status of KMC specified in argument 1.
4	.KDPHA	Halts the KMC-11 specified in argument 1.
5	.KDPMC	Master-clears the KMC specified in argument 1.
6	.KDPST	Starts the KMC specified in argument 1.
7	.KDPRE	Reads the CRAM location from the KMC specified in argument 1 and pointed to by the address in argument 2. The CRAM location is stored in argument 3.
10	.KDPWR	Writes in CRAM location from the KMC specified in argument 1, at the address specified in argument 2, from the value stored in argument 3.
101	.KDLRS	Reads line status of KMC specified in argument 1, on line of DUP specified in argument 2. The line status is returned in the address pointed to by argument 3. Argument 3 must be specified as [len,,addr], where len is the length and addr is the address of the block where status is to be stored.

KDP. [CALLI 200]

Code	Symbol	Meaning
102	.KDLHA	Stops DDCMP on a line specified by the KMC in argument 1 and the DUP in argument 2.
103	.KDLST	Starts DDCMP on a line specified by the KMC in argument 1 and the DUP in argument 2.
104	.KDLSU	Sets the line's user. Specify the KMC in argument 1, the DUP in argument 2, and the SIXBIT/user/ in argument 3. Refer to the DTE. call for more information about line users.
105	.KDLRU	Returns the line's user in argument 3. You must specify the KMC in argument 1 and the DUP in argument 2.

Error Return

One of the following error codes may be returned:

Code	Symbol	Meaning
1	KDILF%	Illegal function code.
2	KDILK%	Illegal KMC-11 number.
3	KDALS%	Argument list too short.
4	KDIWR%	Function is illegal when KMC-11 is running.
5	KDICA%	Illegal CRAM address (.KDPRE or .KDPWR).
6	KDILL%	Illegal line (DUP-11) number.
7	KDKNR%	Function is illegal when KMC-11 is not running.
10	KDLNS%	DDCMP was not started on the line.
11	KDLAS%	DDCMP was already started on the line.
13	KDUNP%	User not privileged to perform this function.

KNIBT. [CALLI 222]

Function

Provides functions for starting and stopping the NIA2Ø microprocessor, and reading and writing its control RAM (CRAM). This call requires POKE privileges.

Calling Sequence

```
        XMOVEI   ac,addr
        KNIBT.   ac
          error return
        normal return

addr:   fcncode,,len
        CPUno,,channo
        args
```

where: addr is the address of the argument list.

fcncode is one of the function codes described below.

len is the length of the argument block.

CPUno is the number of the CPU to which the NIA2Ø is attached.

channo is the RH2Ø channel number of the NIA2Ø microprocessor.

args are arguments specific to the functions you want to perform on the NIA2Ø.

The format of the argument block is:

Offset	Symbol	Contents
Ø	.KBFCN	This word holds the function code and the length of the KNIBT. argument block. The left half, KB.FCN, holds one of the function codes listed below. KB.ALN, the right half, holds the length.
1	.KBKID	This is the NIA2Ø identification word. The left half, KB.CPU, holds the CPU number. The right half, KB.RH2, holds the RH2Ø channel number of the NIA2Ø. The only currently valid value of KB.RH2 is 5.
2	.KBCRA	CRAM address, the third word in the argument block for functions .KBSTA, .KBRED, and .KBWRT (see below).
3	.KBCCH	Contains the high order bits (Ø-29) of the CRAM for the NIA-2Ø. This word is used in the .KBRED and .KBWRT argument blocks.
4	.KBCCL	Holds the low order bits (3Ø-59) of the NIA-2Ø CRAM. .KBRED and .KBWRT also use this word.

The function codes for .KBFCN are as follows:

Code	Symbol	Meaning
1	.KBSTS	Returns the status of the NIA2Ø in the user's ac. Status information is returned in the following format:

Bits	Symbol	Meaning
Ø	KS.RUN	NIA2Ø is running.
1	KS.MAI	NIA2Ø is in maintenance mode.
2	KS.RLD	The NIA2Ø needs to be reloaded.
3	KS.ARD	Auto-reload is disabled.
4	KS.RRQ	The system requested an NIA2Ø reload.
5	KS.DRQ	The system requested an NIA2Ø dump.
6-17		Reserved for DIGITAL.
18-35	KS.RJB	Job number of job reloading NIA2Ø.

2	.KBSRJ	Sets the current job as the job to reload the NIA2Ø.
3	.KBSTP	Stops the NIA2Ø. This must be done before .KBSTA, .KBRED, or .KBWRT are performed.
4	.KBSTA	Starts the NIA2Ø microprocessor at a specific CRAM address given in .KBCRA.
5	.KBRED	Reads the contents of the CRAM location given in .KBCRA, placing the CRAM words into .KBCCH, and .KBCCL.
6	.KBWRT	Writes the contents of a specific CRAM location. .KBCRA, .KBCCH, and .KBCCL are the argument words for .KBWRT.

Normal Return

The specified function is performed; status bits are returned in the ac for function .KBSTS.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	KBPRV%	Insufficient privileges.
2	KBADC%	An address check has been performed.
3	KBIAL%	Invalid argument list.
4	KBILF%	Illegal function.
5	KBICS%	Illegal CPU specified.
6	KBCNA%	CPU not available.
7	KBKDE%	NIA2Ø doesn't exist.
1Ø	KBKMM%	NIA2Ø is in maintenance mode.
11	KBDNS%	NIA2Ø microprocessor didn't start.
12	KBDNI%	NIA2Ø microprocessor didn't initialize.

Code	Symbol	Error
13	KBICA%	Invalid CRAM address.
14	KBCRE%	CRAM read error.
15	KBCWE%	CRAM write error.
16	KBNRJ%	Not the reload job.

Related Call

DIAG.

LATOP. [CALLI 221]

Function

Performs Local Area Terminal (LAT) functions. This function is not intended for customer use.

Calling Sequence

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

addr:   EXP     len
        function code
        argument list
        . . .
```

where addr is the address of the argument list, len is the total length of the argument list including this word, and function code is the function code or symbol listed below. The remainder of the argument list differs depending on the function code. The appropriate argument list is described below for each function.

The functions are:

Code	Symbol	Meaning
Ø	.LASET	Sets LAT parameters for the local node. The parameters you set using this function are dynamic parameters stored only in the host software. Your program must have JACCT or [1,2] privileges to set LAT parameters.

The parameters you can set with .LASET are:

Code	Symbol	Meaning
1	.LPMAC	Maximum number of active circuits.
2	.LPMCO	Maximum number of simultaneous connects.
3	.LPNUM	Host number.
4	.LPLAS	LAT access state.
5	.LPRLI	Circuit retransmit limit.
6	.LPTIM	Retransmit initial value.
7	.LPMTI	Multicast timer initial value.
1Ø	.LPCOD	Group codes.
11	.LPNNM	Host node name.
12	.LPNID	Host node identification string.
13	.LPSRV	Service rating and description.

To set parameters with codes 1 through 12, use the following argument list to .LASET:

Word	Symbol	Contents
Ø	.LAACT	Argument list length.
1	.LAFCN	EXP .LASET.
2	.LAPRM	Parameter code identifying the parameter to be set.

Word	Symbol	Contents
3	.LAVAL	Contents depend on the parameter code:

For Codes	.LAVAL Contains
1 through 7	new parameter value
1Ø	address of a bit mask
11 through 13	ASCIZ string pointer

Word	Symbol	Contents
4	.LAQUA	Qualifier (required for parameter code 13 only) contains the following:

Bit	Symbol	Meaning
Ø	LA%RAT	Rating
1	LA%DSC	Service description

Word	Symbol	Contents
5	.LADSC	ASCIZ string pointer to service description string (LA%DSC=1).

Argument Lists for .LASET

The .LASET function accepts the argument list that is appropriate to the parameter code you specify in addr+2, .LAPRM. Parameter codes 1 through 7 accept an argument directly from .LAVAL. On a successful return, the parameter you specify will be set to the value you include in .LAVAL.

Parameter code 1Ø (group codes) requires the address of a bit mask in .LAVAL. The bit mask is 8 words long, representing the group codes of terminals that can access the host. The bit mask is numbered decimally from Ø to 255, signified by bits Ø through 31 of each word. Each bit you set represents a group number that is allowed to access the system.

Each word in the group code bit mask is formatted as:

```
Ø 1 2 3 4.............................31 32-----35
-----------------------------------------------------
| Each bit represents a group number        | Ignored |
-----------------------------------------------------
```

The group numbers that are represented by each word, starting at ADDR:, are:

Word	Group Numbers
ADDR:	Ø through 31
ADDR+1:	32 through 63
ADDR+2:	64 through 95
ADDR+3:	96 through 127
ADDR+4:	128 through 159
ADDR+5:	16Ø through 191
ADDR+6:	192 through 223
ADDR+7:	224 through 255

To specify a group code number, set the corresponding bit in the bit mask. For example, to set group 64, set bit Ø in the second word of the bit mask.

Parameter codes 11 and 12 require an ASCIZ string pointer in .LAVAL. The pointer may be specified as a byte pointer, or in the form -1,,addr, where addr is the address of the ASCIZ string.

Parameter code 13 allows you to set service rating and description. The contents of .LAVAL include flags and service rating. Depending on the flags you set, you may include a pointer in the following word, .LADSC.

The flags you can set in .LAVAL for parameter code 13 are:

Bit	Symbol	Meaning
Ø	LA%RAT	Sets the rating as specified in the right half of this word. If this bit is not set, and no previous rating has been set, then the rating is automatically set to the default value.
1	LA%DSC	Sets the service description as specified by the next word.

If bit 1 is set, then .LADSC must contain an ASCIZ pointer to a service description string. If LA%DSC is set but .LADSC contains Ø, the service description string is cleared.

Code	Symbol	Meaning
1	.LACLR	Clear specified LAT node parameters. This function requires JACCT or [1,2] privileges.

The argument list for .LACLR consists of the following words:

Word	Symbol	Contents
Ø	.LAACT	EXP len
1	.LAFCN	EXP .LACLR
2	.LAPRM	Parameter code (see function code Ø).
3	.LAVAL	Depends on parameter code in .LAPRM. For parameter code 1Ø, contains the address of the group code bit mask. For parameter 13, contains the ASCIZ pointer to service name to clear. This word is ignored for all other parameters.

Code	Symbol	Meaning
2	.LASCH	Shows the LAT characteristics. You can use this function to obtain the values of both permanent and dynamic parameters.

The argument block for this function is:

Word	Symbol	Contents	
Ø	.LAACT	EXP	len
1	.LAFCN	EXP	.LASCH
2	.LABCT	EXP	len1
3	.LABFA	addr1	

where len is the length of the argument block. The value of len1 is the number of words you have reserved to contain the returned information. On the return, the left half of len1 contains the actual number of words returned. The value of addr1 is the location where the information will be stored on the return.

Code	Symbol	Meaning
3	.LASTC	Shows connects. This function returns information about the active LAT connections at the local node.

The argument list is:

Word	Symbol	Contents	
Ø	.LAACT	EXP	len
1	.LAFCN	EXP	.LASTC
2	.LABCT	EXP	len1
3	.LABFN	addr1	

where the value of len is the length of the argument block. The value of len1 is the number of words reserved for the information that will be returned. On the return, the left half of this word contains the actual number of words returned. The value of addr1 is the address where the information will be returned.

Code	Symbol	Meaning
4	.LASAS	Shows adjacent servers. This function returns information about LAT servers that are able to access the local node.

The argument block for this function is:

Word	Symbol	Contents	
Ø	.LAACT	EXP	len
1	.LAFCN	EXP	.LASAS
2	.LABCT	EXP	len1
3	.LABFA	addr1	
4	.LAQUA	pointer	

Code	Symbol	Meaning

where len is the length of the argument list. The value of len1 is the number of words reserved for the information. On the return, the left half of this word contains the actual number of words used. The value of addr1 is the location where the information will be returned.

.LAQUA contains an ASCIZ string pointer to a location containing the server name. You may specify .LAQUA to receive information about a specific LAT server. This returns a Full Format Server Block. The default action, if this word is zero, is to return a summary of all servers (Short Format Server Block).

5 .LASCO Shows counters. The argument list is:

Word	Symbol	Contents	
Ø	.LAACT	EXP	len
1	.LAFCN	EXP	.LASCO
2	.LABCT	EXP	len1
3	.LABFA	addr1	
4	.LAQUA	pointer	

where len is the length of the argument block. The value of len1 is the number of words reserved for information that will be returned. On the return, the left half of this word contains the actual number of words returned. The value of addr1 is the location where the information will be returned.

The last word, .LAQUA, is optional and may contain an ASCIZ string pointer (addr2) to a word containing the server name, to obtain counter information about the specific server. To obtain totals information, leave .LAQUA zero.

6 .LAZRO Zeroes counters. This function requires JACCT or [1,2] privileges. The argument list is:

Word	Symbol	Contents	
Ø	.LAACT	EXP	len
1	.LAFCN	EXP	.LAZCO
2	.LABCT	EXP	len1
3	.LABFA	addr1	
4	.LAQUA	pointer	

where len is the length of the argument block. The value of len1 is the number of words reserved for information that will be returned. On the return, the left half of this word will contain the actual number of words returned. The value of addr1 is the location where the information will be returned.

The last word, .LAQUA, is optional and may contain an ASCIZ string pointer to a word containing the server name, to obtain counter information about the specific server. To zero all totals, zero the contents of .LAQUA.

Normal Return

On a successful completion of the monitor call, the skip return is taken, the requested information is stored in the specified locations, and the ac contains the address of the argument list.

Several LATOP. functions return information in a buffer starting at the address stored in Word 3 of the argument block, .LABFA. The functions and the format of the information returned are listed below.

Function .LASCH (Show Characteristics)

Show Buffer Format

0–17	18–35
MAX-ALLOC-CIRCUITS	N-ALLOC-CIRCUITS
MAX-ACTIVE-CIRCUITS	N-ACTIVE-CIRCUITS
MAX-CONNECTS	N-CONNECTS
HOST-NUMBER	LAT-TERMINAL-ACCESS-STA
HOST-RETRANSMIT-LIMIT	HOST-CIRCUIT-TIMER
HOST-MULTICAST-TIMER	Reserved
HIGH-PROTOCOL-VERSION	LOW-PROTOCOL-VERSION
PROTOCOL-ECO	CURRENT-PROTOCOL-VERSION
MAX-SLOT-SIZE	MAX-SLOTS
FRAME-SIZE	MAX-SERVICES
HOST GROUP CODES (8 words)	
HOST-NAME count	HOST-ID count
HOST NAME (2 words)	
HOST ID (13 words)	
Service Blocks (19 words per service)	

Service Block Format

HOST-SERVICE-NAME-RATING	
SERVICE-NAME count	SERVICE-DESCRIPTION count
SERVICE-NAME (4 words)	
SERVICE-DESCRIPTION (13 words)	

Function .LASTC (Show Connects)

Connect Block Format

Terminal Number	
Server Name count	Indeterminate
Server Name (4 words)	

Function .LASAS (Show Adjacent Servers)

Full Format Block

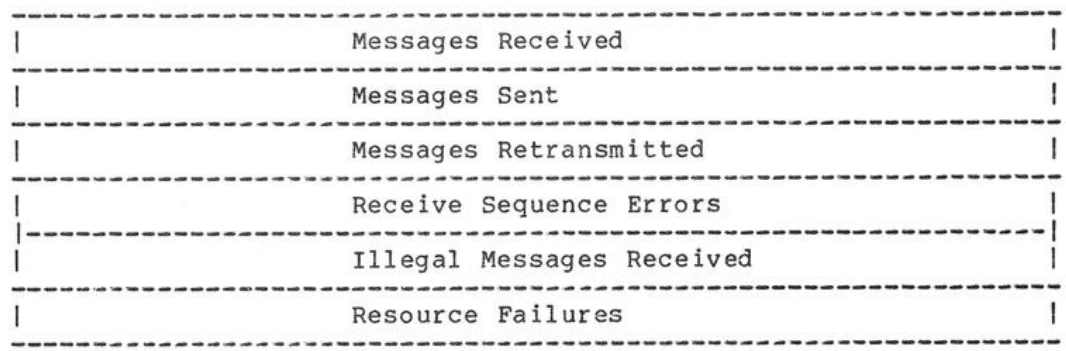
Server Ethernet Address (2 words)	
FRAME-SIZE	SERVER-VERSION
MAX-SLOTS	Indeterminate
CIRCUIT-TIMER	KEEP-ALIVE-TIMER
PRODUCT-TYPE	STATE
SERVER-NUMBER	SERVER-NAME count
SERVER-LOCATION count	Unused
SERVER-NAME (4 words)	
SERVER-LOCATION (4 words)	

Short Format Block

SERVER-NUMBER	SERVER-NAME count
SERVER-NAME (4 words)	
ETHERNET-ADDRESS (2 words)	

Functions .LASCO (Show Counters) and .LAZCO (Zero Counters)

Counter Block Format



Error Return

On an error return, the non-skip return is taken, and the ac contains an error code. The error codes are:

Code	Symbol	Meaning
Ø	LABTS%	The buffer size you allocated was too small for the amount of information available. The actual number of words that are required is stored in the left half of .LABCT.
1	LAVOR%	Value of a parameter is outside the allowed range.
2	LALNO%	LAT is not operational.
3	LASVR%	Invalid or unknown LAT server name.
4	LAIPN%	Invalid LAT parameter.
5	LAIPV%	Invalid LAT parameter value.
6	LASVC%	Invalid or unknown LAT service name.
7	LAILR%	Insufficient LAT resources.
1Ø	LAHAS%	LAT host name already set.
11	LAIVF%	Invalid function code.
12	LAABS%	Argument list too small.
13	LAADC%	Address check for argument list (specified address not in memory)
14	LAPRV%	Not enough privileges.

LLMOP. [CALLI 220]

Function

Performs functions for the network management layer of DECnet. This call is used only by the NML program and is not intended for use in customer programs. The LLMOP. UUO may change at any time without notice. This call requires [1,2], JP.POK, or JACCT privileges.

Calling Sequence

```
MOVE   ac1, fcncode
XMOVEI ac2, addr
LLMOP. ac2,
  error return
normal return
```

where: fcncode is the function code. The argument block found at addr is specific to the function code contained in ac1.

addr is the address of the argument block.

Function codes for LLMOP. are:

Code	Symbol	Function
0	.ELDIR	Builds an Ethernet loopback message from data supplied in the argument block, and transmits it to the destination address. The argument block is:

Offset	Symbol	Contents
0	.LMCID	Channel ID. Bits 34 and 35 (LM.CID) contain the value of the Ethernet port to use.
1-2	.LMDST	Destination address.
3	.LMREQ	Request number, containing:

Bits	Symbol	Meaning
0	LM.AIC	Assigns interrupt channel specified in LM.ICH.
12-17	LM.ICH	Contains PSI channel to interrupt when the message arrives.

Code	Symbol	Function

Bits	Symbol	Meaning
18-35	LM.REQ	Contains the request number returned by LLMOP. This value is used in function .ELRPY.

Offset	Symbol	Contents
4	.LMRBL	Length of the loopback request data buffer. The right half (LM.MBL) contains the length of the data portion of the loopback message.
5	.LMRBP	Pointer to loopback request data buffer.

1 .ELAST — Builds an Ethernet loopback message, and transmits it according to the type of assistance required. The first words in the argument block, .LMCID, .LMDST, .LMREQ, .LMRBL, and .LMRBP, are described in function .ELDIR. The remainder of the argument block is:

Offset	Symbol	Contents
6-7	.LMAST	Address of the node used as the assistant in the loopback request. This may not be a multicast address.
10	.LMHLP	Assistance level. Level 1, .LMXMT, forwards the loopback message to both the destination and local nodes. Level 2, .LMRCV, forwards the loopback message to assistant and local nodes. Level 3, .LMFUL, forwards the message to destination, assistant, and local nodes.

2 .ELRPY — Reads the loopback reply message. The argument block is:

Offset	Symbol	Contents
0	.LMCID	Channel ID. Bits 34 and 35 (LM.CID) contain the value of the Ethernet port to use.
1-2	.LMSRC	Address of the remote system that satisfied a loop assisted operation.

Code	Symbol	Function

Offset	Symbol	Contents
3	.LMREQ	Request number. The right half (LM.REQ) contains the request number of the reply to be read. The caller is blocked until the reply arrives.
4	.LMRBL	Length of the loop response buffer. The left half (LM.RML) contains on return the length of the received loop reply message data. The right half (LM.MBL) holds the maximum length of the loop response message buffer that you supply.
5	.LMRBP	Pointer to loop reply buffer.

3 .ELAIC Assigns interrupt channel for Ethernet loopback reply. The argument block is:

Offset	Symbol	Contents
0	.LMCID	Channel ID, where Bits 34 and 35 (LM.CID) contain the value of the Ethernet port to use.
1	.LMICF	Interrupt channel flags, in the form:

Bits	Symbol	Meaning
0	LM.AIC	Assigns the interrupt channel given in LM.ICH when lit.
12-17	LM.ICH	Contains the PSI channel to interrupt when the loopback message arrives.

4 .ELABT Aborts the loop request. The argument block is:

Offset	Symbol	Contents
0	.LMCID	Channel ID, where bits 34 and 35 contain the value of the Ethernet port to use.

Code	Symbol	Function

Offset	Symbol	Contents
3	.LMREQ	Request number. The right half, LM.REQ, contains the number of the request to be aborted.

5 .ELSTS Obtains status of Ethernet loopback requests. The argument block is:

Offset	Symbol	Contents
0	.LMCID	Channel ID, where bits 34 and 35 contain the value of the Ethernet port to use.
1	.LMSTF	Status code for the request. The right half, LM.RTC, contains one of the following status codes:

Code	Symbol	Status
0	.LMPND	Request pending, incomplete.
1	.LMSUC	Request completed successfully.
2	.LMABT	Request aborted.
3	.LMTXF	Transmit failed.
4	.LMCCE	Channel communication error.

Offset	Symbol	Contents
2	.LMCST	Status returned from the KLNI port driver.
3	.LMREQ	Request number. The right half, LM.REQ, contains the number of the request to be aborted.

6 .RCRID Transmits a Read Identify protocol message to the destination address node on the Ethernet. Use the .RCRPY function to read the System ID reply message. The argument block is identical to that of function .ELDIR. The value returned in LM.REQ of .LMREQ must be used in any subsequent .RCRPY, .RCABT, or .RCSTS calls.

7 .RCRCT Transmits a Read Counters protocol message to the destination address node on the Ethernet. Use the .RCRPY function to read the System ID reply message. The argument block is identical to that of function .ELDIR.

Code	Symbol	Function
10	.RCIDS	Transmits a System ID protocol message to the destination address node on the Ethernet. This function blocks the program until the transmit is completed. The argument block is:

Offset	Symbol	Contents
0	.LMCID	Channel ID, where bits 34 and 35 (LM.CID) contain the value of the Ethernet port to use.
1-2	.LMDST	Destination address.

Code	Symbol	Function
11	.RCRBT	Transmits a Boot protocol message to the destination address node on the Ethernet. .RCRBT blocks the issuing process until the transmit is completed. The argument block is:

Offset	Symbol	Contents
0	.LMCID	Channel ID, where bits 34 and 35 (LM.CID) contain the value of the Ethernet port to use.
1-2	.LMDST	Destination node address.
3-4	.LMPWD	8-byte verification code. The code is transmitted to the remote system, which uses it in deciding whether to allow the boot request. The 8-bit bytes are packed four to a word.
5	.LMCIF	Control information, in the form:

Bits	Symbol	Meaning
26	LM.BDV	Specifies the boot device, where 0 indicates the system default, and 1 represents a specified device.
27	LM.BSV	Specifies the boot server, where 0 is the system default, and 1 indicates requesting a system.

Code	Symbol	Function

Bits	Symbol	Meaning
28-35	LM.PRO	Specifies the processor to boot. Ø indicates the system processor, and 1 represents the communication processor.

Offset	Symbol	Contents
6	.LMDID	Device ID in an 8-bit byte string.
7	.LMSID	Software ID in an 8-bit byte string.

12 .RCRPY Reads the response to a request ID or Read Counters function. The format of the argument block is the same as for .ELRPY. .LMSRC contains the address of the responding node. .LMRBL contains the returned message length, and .LMRBP contains a pointer to the response buffer.

13 .RCRSV Transmits a reserve remote console MOP message. The argument block contains .LMCID, .LMDST, and .LMPWD, as described in function .RCRBT.

14 .RCREL Transmits a release remote console MOP message. The argument block contains .LMCID and .LMDST.

15 .RCSND Sends ASCII console command data to a remote console and polls for response data. If no command data is included, the function only polls for response data. The argument block is:

Offset	Symbol	Contents
Ø	.LMCID	Channel ID, in the form:

Bits	Symbol	Meaning
16	LM.CBF	Command break flag. If this bit is set, a break condition in the serial byte stream precedes the command data buffer.

Code	Symbol	Function

Bits	Symbol	Meaning
17	LM.MNO	Message number, which is a one-bit sequence number, indicating the current Console Requestor command message.
34-35	LM.CID	Channel ID.

Offset	Symbol	Contents
1-2	.LMDST	Destination address.
3	.LMREQ	Request number, as described in .ELDIR.
4	.LMRBL	Length of console request buffer. The right half, LM.MBL, contains the maximum buffer length.
5	.LMRBP	Pointer to the remote console data buffer.

16 .RCPOL Polls for completion of the Send Console Command function. The argument block is:

Offset	Symbol	Contents
Ø	.LMCID	Channel ID and returned flags, in the form:

Bits	Symbol	Meaning
7	LM.RDL	Indicates that received data was lost. The flag is set by the local requestor if the response data buffer was too small to receive the data from the remote node.
15	LM.RDO	Indicates that response data was lost, due to a buffer overrun or error condition.

Code	Symbol	Function

Bits	Symbol	Meaning
16	LM.CDL	Indicates that command data was lost. This flag is set if command data in the Console Command message was lost. The remote server sets this bit.
17	LM.MNO	Message number, which is a one-bit sequence number, indicating the current Console Requestor command message.
34-35	LM.CID	Channel ID.

Offset	Symbol	Contents
1-2	.LMSRC	Source node and physical address of the node that sent this reply.
3	.LMREQ	Request ID, assigned by .RCSND.
4	.LMRBL	Length of console response buffer. The format of the buffer is described in .ELRPY
5	.LMRBP	Pointer to the remote console data buffer.

20	.RCABT	Aborts an outstanding remote console request. The argument block is identical to that of .ELABT.
21	.RCSTS	Obtains status of a remote console request. The argument block is identical to that of .ELSTS.

LLMOP. [CALLI 220]

Code	Symbol	Function
22	.RCADR	Obtains a channel address. The argument block is:

Offset	Symbol	Contents
0	.LMCID	Channel ID, where bits 34 and 35 contain the value of the Ethernet port to use.
1-2	.LMHWA	Hardware address.
3-4	.LMPYA	Physical address.

Normal Return

On a successful completion, the requested functions are performed, and any returns are made as specified in the description of the function code.

Error Return

One of the following codes is returned in the ac:

Code	Symbol	Error
1	LMPRV%	Program has insufficient privileges.
2	LMIIF%	Program specified an illegal function.
3	LMICN%	Program specified an illegal channel number.
4	LMOFF%	LLMOP. is off.
5	LMADC%	An address check was performed.

LOCATE [CALLI 62]

Function

Changes the logical node number for the current job. This call functions in the ANF-1Ø network to allow you to route device I/O to devices at other nodes. Subsequent references to output devices (such as line printers) and input devices (such as card readers), when implicitly requested or generically referenced, will be assumed to refer to devices on the node you specify with this call.

Calling Sequence

```
{MOVE   ac,[SIXBIT/nodename/]}
{MOVEI  ac,nodenumber        }
 LOCATE ac,
   error return
 normal return
```

where: **nodename** is the SIXBIT physical name of a node.

nodenumber is one of the following:

-1	Changes your job's location to the physical node of your terminal.
Ø	Changes your job's location to that of the host computer.
n	Changes your job's location to node number n, where n is a positive integer.

Normal Return

The location of your job is changed as specified. Any subsequent generic device specifications are associated with the new node number and node name.

Error Return

The error return occurs if the LOCATE monitor call is not implemented on your system, or if you specified an invalid node number or node name.

Examples

```
MOVEI   T1,3
LOCATE  T1,
  JRST  ERROR
```

Locates your job at node number 3.

Related Calls

WHERE

LOCK [CALLI 60]

Function

Locks the current job into user memory. Note that there are two calling sequences for LOCK. The standard calling sequence is described under Calling Sequence 1 and the extended calling sequence is described under Calling Sequence 2. The extended calling sequence locks a segment starting at a specified page in physical memory.

The default function of this call locks the segments of the program as set by bits 17 and 35 in the accumulator. Bit 17 must be set to lock the high segment; bit 35 must be set to lock the low segment. The specified segment(s) is locked into physically contiguous memory in contiguous executive virtual memory space, unless you set flags in the accumulator to specify otherwise.

NOTE

Programs using user mode extended addressing cannot use the LOCK monitor call.

Calling Sequence 1

```
MOVE    ac,[flags]
LOCK    ac,
  error return
normal return
```

where: flags include one or more of the following bits:

Bit	Symbol	Function
14	LK.HLC	Locks the high segment in user core and sets its cache bit. If this bit is off, the high segment is locked with its cache bit off. KL10 processors will run your program faster if you use LK.HLC; however, for a real-time program that has direct access to memory, you should not set LK.HLC.
15	LK.HNP	Locks the high segment without forcing the job to be locked into physically contiguous locations. If this bit is not set, physical contiguity for the locked high segment is required.
16	LK.HNE	Locks the high segment without forcing it to reside in executive virtual memory. If this bit is not set, the locked high segment must reside in executive virtual memory.
		NOTE For executive-mode, real-time trapping, your high segment must be locked into contiguous executive virtual memory.
17	LK.HLS	Locks the high segment. Without this bit set, the high segment will not be locked, and bits 14-16 will be ignored.

Bit	Symbol	Function
32	LK.LLC	Locks the low segment in user core and sets its cache bit. If this bit is off, the low segment is locked with its cache bit off. Processors will run your program faster if you use LK.LLC; however, for a real-time program that has direct access to memory, you should not set LK.LLC.
33	LK.LNP	Locks the low segment without requiring physically contiguous locations for the low segment. If this bit is not set, the low segment must be locked into physically contiguous locations.
34	LK.LNE	Locks the low segment without requiring the low segment to reside in executive virtual memory. If this bit is not set, the low segment must be locked into executive virtual memory.

NOTE

For executive-mode, real-time trapping, your low segment must be locked into contiguous executive virtual memory.

35	LK.LLS	Locks the low segment. If this bit is clear, the lowseg will not be locked, and bits 32-35 will be ignored.

Calling Sequence 2

```
        MOVE    ac,[XWD -n,addr]
        LOCK    ac,
          error return
        normal return
        . . .
addr:   fcn-code
        hiseg,,lowseg
```

where: **n** is the number of arguments plus one, expressed as a negative value.

addr is the address of the argument block.

fcn-code is the function code described below.

hiseg is set if the high segment is to be locked. **lowseg** is set if the low segment is to be locked.

Code	Symbol	Meaning
0	.LKPPN	Locks the high and/or the low segment into contiguous physical pages, starting at the physical page number specified in the argument. The left half of addr+1 contains the starting page number of the high segment; if this halfword is 0, the high segment is not locked. The right half of addr+1 contains the starting page number of the low segment; if this halfword is 0, the low segment is not locked.

If you use Calling Sequence 2 when the system is running with KL-paging the low segment is locked into the second higher physical page.

Normal Return

When using Calling Sequence 1, the monitor has locked the program into core. If physical or executive virtual contiguity is required, the following information is stored in the ac:

XWD hiseg,lowseg

where: **hiseg** is the address of the high segment (Ø if no high segment exists).

lowseg is the address of the low segment.

If no contiguity is required, the ac is cleared.

The monitor will lock your program into memory and take the normal return if all of the following conditions are met:

- The lock privilege bit (JP.LCK) is set for your job.
- The locked job would not prevent any other job from expanding to its guaranteed minimum (CORMIN).
- The locked job would not prevent any other current job from running. (Note that unlocked jobs can exceed CORMIN.)
- For executive virtual mapping, the locked job would not exceed the maximum amount of executive virtual memory available for locking.
- The job either has no high segment, or has a sharable high segment.
- The job is not virtual and has a contiguous core image.

When using Calling Sequence 2, the monitor locks the specified segment (contiguously and physically) starting at the page in physical memory specified in your program. If you specify that the low segment is to be locked, the monitor locks your job into the next higher physical page location than the one you specified in the right half of your argument.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
Ø	LKNIS%	The LOCK call, or a feature you requested, is not implemented on your system; or you attempted to lock a nonsharable high segment.
1	LKNLP%	No locking privilege.
2	LKNCA%	Not enough core available; your locked job would prevent running an unlocked job.
3	LKNCM%	Not enough core for CORMIN; your job would prevent maintaining CORMIN for unlocked jobs.

Code	Symbol	Error
4	LKNEM%	Not enough core for executive virtual memory; your locked job would exceed the maximum allowable executive virtual memory. You can obtain the executive virtual memory maximum and in-use values from the GETTAB table .GTCnV, where n is the CPU number. The maximum is in word 43 (%CVEVM) of the table and the in-use value is in word 44 (%CVEVU).
5	LKNIA%	Illegal flags specified.
6	LKNPU%	Specified page not available. You would receive this error on an extended LOCK call if the two segments would overlap, one or both segments would overlap another locked job or the monitor, or one or both segments would be outside the range of on-line memory.
7	LKNAL%	Illegal movement specified. You tried to move a locked segment or place a segment into executive virtual memory.

Examples

See Chapter 9.

LOGIN [CALLI 15]

Function

Informs the monitor that a job has successfully logged in, and passes certain parameters to the monitor (including the project-programmer number). The calling job must not be logged in.

The LOGIN monitor call is used by the LOGIN and INITIA programs and is not intended for customer use.

Calling Sequence

```
        MOVE    ac,[XWD -len,addr]
        LOGIN   ac,
        return
         ...
addr:   proj,,prog              ;JBTPPN  (.GTPPN)
        privilege bits          ;JBTPRV  (.GTPRV)
        user-name               ;first half, .PDNM1  (.GTNM1)
        user-name               ;second half, .PDNM2 (.GTNM2)
        charge #                ;.PDCNØ  (.GTCNO)
```

where: **len** is the length of the argument list.

addr is the address of the argument list. The data in the argument list is to be passed to the monitor.

Return

The job is logged in, if it is not already logged in.

Related Calls

ACCLG., CHGPPN, LOGOUT

LOGOUT [CALLI 17]

Function

Releases all I/O devices associated with the calling job and returns them to the monitor's pool of available devices, along with the job's allocated core and its job number.

To perform this call, the user program should use the RUN UUO to call SYS:LOGOUT.EXE, where SYS is the [1,4] area.

The LOGOUT UUO has no error return. If the calling program has JACCT privileges and is named LOGOUT.EXE, this call logs out the job. Otherwise, the call functions like an EXIT UUO.

Calling Sequence

```
LOGOUT
return
```

Related Calls

EXIT

LOOKUP [OPCODE Ø76]

Function

Selects a file for input. Use FILOP. to perform a LOOKUP for an extended I/O channel. The LOOKUP call is meaningful only for directory devices (disk, DECtape, labelled magnetic tape), and for TSK devices (initiated for task-to-task communication). It is a no-op for other devices, always taking the normal return for these.

Calling Sequence

The LOOKUP monitor call, like the ENTER call, has two calling sequences: one using a 4-word argument list and one using an extended argument list. The argument lists for LOOKUP, ENTER, and RENAME UUOs are identical. These are described in Section 11.13. The four-word argument list is detailed in Section 11.13.1. The extended argument list is described in Section 11.13.2.

Normal Return

For DECtape, the monitor returns a 4-word block at addr in the following form:

Offset	Contents
Ø	The SIXBIT file name.
1	The extension, creation date, and first block number:

Bits	Contents
Ø-17	The SIXBIT file extension.
18-2Ø	The high-order three bits of the file creation date.
21-25	Reserved.
26-35	The first physical (data) block number.

2 Remainder of creation date:

Bits	Contents
Ø-23	Reserved.
24-35	The low-order 12 bits of the file creation date.

3 Length and address of file:

Bits	Contents
Ø-17	Negative of number of words in zero-compressed file.
18-35	Address of word preceding the first word of the file.

For disk files, and labelled magtape files, refer to Sections 11.13.1 and 11.13.2 for the argument blocks returned by LOOKUP, ENTER, and RENAME UUOs.

Error Return

The error codes for LOOKUP are the same as those for ENTER, and are documented in Section 11.14.

Examples

For more information about doing I/O and examples using the LOOKUP call, refer to Chapter 11.

MERGE. [CALLI 173]

Function

Merges an .EXE file or a portion of an .EXE file into the currently loaded low segment in memory, ignoring page Ø from the .EXE file.

Calling Sequence

```
        MOVEI   ac,addr
        MERGE.  ac,
          error return
        normal return
        ...
addr:   SIXBIT/device/
        SIXBIT/filename/
        SIXBIT/extension/
        EXP     Ø
        XWD     {proj,prog}
                {Ø,addr2  }
        XWD     {low-page,hi-page}
                {-n,,addr3       }
                {Ø               }
```

where: **addr** is the address of the 6-word argument block.

device is the name of the device on which the .EXE file resides.

filename specifies the name of the .EXE file.

extension specifies the extension of the .EXE file.

proj,prog is the project-programmer number representing the directory in which the .EXE file resides. If you specify this word as zero, your directory path is assumed. If this word is [XWD Ø,addr], **addr** points to a PATH. block containing the full path specification for the file. Refer to the description of the PATH. monitor call for information pertaining to the format of the PATH. block.

The last word of the argument block has three possible forms: **low-page** and **hi-page** specify the lower-bound virtual page number and the upper-bound virtual page number of the .EXE file to be loaded into your low segment. Specifying -n,,addr3 indicates that each of the n ranges of pages given in addr3 (in low-page, hi-page form) are to be MERGEd. This format saves you from performing multiple MERGEs. Finally, placing zero in the last word causes the low-segment pages in the .EXE file to be merged.

Normal Return

The .EXE file pages are merged into the current low segment in memory. The accumulators are destroyed and channel Ø is released.

Error Return

The error return is taken if any errors are detected; the monitor returns an error code in the **ac**. The possible error codes are listed in Chapter 11.

Related Calls

GETSEG, RUN

Common Errors

- Forgetting to save the acs over the MERGE.
- Forgetting that channel Ø is destroyed.
- Attempting to MERGE high segment data.

MONRT. [CALL 1,12]

Function

Identical to the call:

```
EXIT 1,
```

See the EXIT monitor call. Note that this function does not perform a RESET for your job.

MSTIME [CALLI 23]

Function

Returns the current time of day.

Calling Sequence

```
MSTIME ac,
return
```

Return

The time elapsed (in milliseconds) since midnight is returned in the ac.

Related Calls

DATE, RUNTIM, TIMER

MTAID. [CALLI 126]

Function

Associates a SIXBIT reel identifier with a specified magnetic tape drive. This call requires JACCT or [1,2] privileges.

Calling Sequence

```
{MOVE     ac,[SIXBIT/device/]}
{MOVEI    ac,channo           }
{MOVEI    ac,udx              }
 MOVE     ac+1,[SIXBIT/reelid/]
 MTAID.   ac,
   error return
  normal return
```

where: **device** is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

reelid is the SIXBIT tape reel identifier, or Ø to clear the current reelid.

Note that your program can also clear the reel identifier by using function code 11 (MTUNL.) to the MTAPE monitor call; or by deassigning the drive, using the REASSI UUO. All reel-specific error counts are cleared by the MTAID. call in order that all accumulated data for the specific reel is accurate.

Normal Return

The monitor has associated the tape reel identifier with the specified magtape device; the reel identifier is included in all media reports.

Error Return

One of the following error codes is returned in the **ac**:

Code	Symbol	Error
-2	MTINA%	Specified device is not available to your job or your job is not privileged.
-1	MTINX%	Specified device is nonexistent or not a magtape device.

Examples

```
 MOVE     T1,[SIXBIT/MTAØ/]
 MOVE     T1+1,[SIXBIT/REEL1/]
 MTAID.   T1,
   JRST   ERROR
```

Related Calls

MTAPE, MTCHR., TAPOP.

MTAPE [OPCODE Ø72]

Function

Passes the monitor a code for an extended set of calls; these calls perform functions for magnetic tapes and are usually called MTAPEs. Use FILOP. or TAPOP. to perform magnetic tape or DECtape functions on extended I/O channels.

Each defined MTAPE code also has a symbolic name; in this chapter the MTAPEs are discussed in alphabetical order by their names. For example, MTAPE 3 has the name MTEOF.; its function is discussed under the name MTEOF.

Magtape I/O is described in Volume 1, Chapter 14.

The MTAPEs are:

MTWAT.	[MTAPE Ø]
MTREW.	[MTAPE 1]
MTEOF.	[MTAPE 3]
MTSKR.	[MTAPE 6]
MTBSR.	[MTAPE 7]
MTEOT.	[MTAPE 1Ø]
MTUNL.	[MTAPE 11]
MTBLK.	[MTAPE 13]
MTSKF.	[MTAPE 16]
MTBSF.	[MTAPE 17]
MTDEC.	[MTAPE 1ØØ]
MTIND.	[MTAPE 1Ø1]
MTLTH.	[MTAPE 2ØØ]

After your program issues the MTAPE monitor call, the monitor waits for the magnetic tape to complete any action in progress. The monitor then clears bits 18-25 of the file status word, initiates the indicated MTAPE function, and returns control immediately to your program.

The I/O service routine may be reading several blocks ahead of your program when performing I/O in buffered mode. The execution of the MTAPE monitor call affects only the physical position of the magnetic tape and does not change the data that has already been read into the buffer. Therefore, when your program issues either an IN, INPUT, OUT, or OUTPUT call after the MTAPE monitor call, the monitor may not retrieve the buffer containing the block requested. To guarantee that the requested block will be in the buffer, your program can set up a single buffer ring. With a single buffer ring the monitor is prohibited from reading ahead, and it stops the device after every IN, INPUT, OUT, or OUTPUT monitor call. Alternatively, your program can set bit 3Ø (IO.SYN) in the I/O status word. Setting this bit causes the monitor to stop the device after each buffer is filled on an IN, INPUT, OUT, OUTPUT, or FILOP. monitor call. Note that the FILOP. monitor call provides the functions of the MTAPE calls.

MTBLK. [MTAPE 13]

Function

Writes three inches of blank tape. Use FILOP. to perform an MTBLR. on an extended I/O channel.

Calling Sequence

```
MTBLK.  channo,
return
```

where: channo is the number of a channel initialized for a magtape device.

Return

Three inches of blank tape are written on the device associated with the given channel.

Examples

```
MTBLK.  5,
```

Three inches of blank tape is written to the magtape on the unit associated with channel 5.

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return where none exists.

Forgetting to include a comma after the channel number.

MTBSF. [MTAPE 17]

Function

Backspaces one file on a magtape. Use FILOP. to perform an MTBSF. on an extended I/O channel.

Calling Sequence

```
MTBSF.  channo,
return
```

where: channo is the number of a channel initialized for a magtape device.

Return

The monitor backspaces over one file on the device associated with the given channel. The monitor moves the tape in the reverse direction until the tape has passed a tape mark or reached the beginning of the tape. The backspace operation positions the tape heads either immediately in front of a tape mark or at the beginning of the tape.

In most cases, your program should skip forward over the file mark to the beginning of the file. However, when you have backspaced to the beginning of the tape and when your program issues the MTSKF. call, the monitor skips 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 backspacing a file is:

1. MTBSF. to backspace the file.
2. MTWAT. to wait for completion
3. STATO MT,IO.BOT to determine whether this is the beginning of the tape.
4. MTSKF. to skip over the file mark if it is not the beginning of the tape.

It is necessary to wait after the MTBSF. instruction to ensure that the move is complete before testing to see whether or not this is the beginning of the tape, but your program can use the MTWAT. call to wait for the spacing operation to be completed.

Examples

```
MTBSF.  5,
```

Backspaces over 1 file on the tape associated with channel 5.

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return when none exists.

Forgetting to include a comma after the channel number.

MTBSR. [MTAPE 7]

Function

Backspaces one record on a magtape device. Use FILOP. to perform an MTBSR. on an extended I/O channel.

Calling Sequence

```
MTBSR.  channo,
return
```

where: channo is the number of a channel initialized for a magtape device.

Return

The monitor backspaces over one record on the device associated with the given channel.

Examples

```
MTBSR.  7,
```

This call backspaces over a record on the magtape associated with channel 7.

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return where none exists.

Forgetting to include a comma after the channel number.

MTCHR. [CALLI 112]

Function

Returns information about the state of a magtape drive.

Calling Sequence

```
        {MOVE    ac,[SIXBIT/device/]}
        {MOVEI   ac,channo          }
        {MOVEI   ac,udx             }
        {MOVE    ac,[XWD len,addr]  }
        MTCHR.  ac,
          error return
        normal return
        . . .
addr:   device-identifier     ;applicable only if ac contains
        BLOCK   2Ø            ; len,,addr
```

where: **device** is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

Optionally, you can specify the length and location of the argument list:

len is the length of the argument list.

addr is the address of the argument list, and the words at **addr+1** will be filled in by the monitor on return. The argument list is used only if the length and address are given in the ac instead of a device identifier (device, channel number, or UDX).

device-identifier is set by the user program and contains the device name, UDX, or channel number.

Normal Return

The monitor returns a value in the **ac**, and, if you used the optional argument list, the monitor returns values beginning at **addr+1**.

The word returned in the ac is in the format:

Bits	Symbol	Meaning
Ø-17	MT.AWC	The word count of the last record read or written.
18-26	MT.CRC	If a 9-track NRZI tape, this field contains the last cyclic redundancy character (CRC); otherwise, this field contains Ø.
27-29	MT.NCR	The number of characters not accounted for in MT.AWC, read from the tape into the last addressed location during the last read.
3Ø		Reserved for use by DIGITAL. Should contain Ø.
31	MT.7TR	The unit is a 7-track unit.
32	MT.WLK	The tape transport is write-locked.

Bits	Symbol	Meaning
33-35	MT.DEN	The tape density code:

Code	Symbol	Density
1	.MTDN2	200 bits per inch (8.1 rows per mm).
2	.MTDN5	556 bits per inch (22.5 rows per mm).
3	.MTDN8	800 bits per inch (32.2 rows per mm).
4	.MTD16	1600 bits per inch (65.3 rows per mm).
5	.MTD62	6250 bits per inch (255.5 rows per mm).

When the monitor determines the value of the density indicator to be returned, it examines the I/O status bits you set in the OPEN call. The monitor returns the density identifier you set in the OPEN call. If you did not use OPEN to specify a density indicator, the monitor determines whether or not you issued the SET DENSITY monitor command. If you did, the monitor returns, in the ac, the same value you specified in the monitor command. If neither, the monitor returns the system-default density. (Note that when you issue a GETSTS, the monitor examines only the I/O status bits set by the OPEN. If you did not specify a density indicator with OPEN, the monitor returns a 0 when you issue a GETSTS. Therefore, when you issue a GETSTS, the monitor does not further investigate the density indentifier or supply the system-default density indicator.)

If you use the optional argument list, the monitor returns data at addr+1 and the subsequent locations. The information, starting addr+1, is returned as:

Offset	Symbol	Contents
0	.MTCHN	Channel number.
1	.MTRID	SIXBIT reel identifier of the tape.
2	.MTWRD	Number of files read since the beginning of the tape.
3	.MTREC	Number of records since last end-of-file.
4	.MTCRD	Number of characters read since last tape unload.
5	.MTCWR	Number of characters written since last tape unload.
6	.MTSRE	Number of soft read errors since last unload.
7	.MTHRE	Number of hard read errors since last unload.
10	.MTSWE	Number of soft write errors since last unload.
11	.MTHWE	Number of hard write errors since last unload.
12	.MTTME	Total number of read and write errors since last tape unload.

Offset	Symbol	Contents
13	.MTTDE	Number of device errors since system startup.
14	.MTTUN	Number of unloads since system reload.
15	.MTRTY	Number of retries to resolve last error; if bit 1 is set, the error is a hard error.
16	.MTCCR	Character count of the last record read or written.
17	.MTPBE	Position before error. The file number before last error (right half), and record number before last error (left half).
20	.MTFES	Final error state. Refer to the TOPS-10/TOPS-20 SPEAR Manual.

Error Return

The error return is taken and -1 returned in the ac if the device you specified was not a magnetic tape unit or was nonexistent.

Related Calls

TAPOP.

MTDEC. [MTAPE 100]

Function

Initializes a channel for DIGITAL-compatible mode tape handling. Use FILOP. to perform an MTDEC. on an extended I/O channel.

In DIGITAL compatible mode, the monitor writes or reads 36 bits in 5 frames of a 9-track magnetic tape. The tape can be any density or parity and is not industry-compatible. DIGITAL compatible mode is the default mode that is set when the channel is opened.

The DIGITAL-compatible mode remains in effect until the channel is released, or until you issue the MTIND. monitor call for the channel.

Calling Sequence

```
MTDEC.  channo,
return
```

where: channo is the number of a channel initialized for a magtape device.

Return

The device associated with the given channel is initialized for DIGITAL-compatible mode handling.

Examples

```
MTDEC.  11,
```

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return where none exists.

Forgetting to include a comma after the channel number.

MTEOF. [MTAPE 3]

Function

Writes an end-of-file mark on a magtape. Use FILOP. to perform an MTEOF. on an extended I/O channel.

Calling Sequence

```
MTEOF.  channo,
return
```

where: **channo** is the number of a channel initialized for a magtape device.

Return

The monitor writes an end-of-file mark on the specified device.

Examples

```
MTEOF.  1Ø,
```

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return when none exists.

Forgetting to include a comma after the channel number.

MTEOT. [MTAPE 1Ø]

Function

Advances a magtape device to the logical or physical end-of-tape. Use FILOP. to perform an MTEOT on an extended I/O channel.

The logical end-of-tape is indicated by two consecutive end-of-file marks. The MTEOT. call positions the tape between these two marks, allowing files to be appended to the tape.

Calling Sequence

```
MTEOT.  channo,
return
```

where: **channo** is the number of a channel initialized for a magtape device.

Return

The monitor positions the tape between the two end-of-file marks that indicate the end-of-tape.

Examples

```
MTEOT.  6,
```

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return when none exists.

Forgetting to include a comma after the channel number.

MTIND. [MTAPE 1Ø1]

Function

Initializes a channel for industry-compatible mode tape handling. Use FILOP. to perform an MTIND. on an extended I/O channel.

In industry-compatible mode, the monitor writes or reads 32 bits in 4 frames of a 9-track magnetic tape, ignoring the low-order 4 bits of each PDP-1Ø word. MTIND. will set a default density to 16ØØ BPI, or the highest density allowed on the drive.

The industry-compatible mode remains in effect until the channel is released, or until you issue the MTDEC. monitor call for the channel.

Calling Sequence

```
MTIND.  channo,
return
```

where: channo is the number of a channel initialized for a magtape device.

Return

The device associated with the given channel is initialized for industry-compatible mode handling.

Examples

```
MTIND.  1Ø,
```

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return when none exists.

Forgetting to include a comma after the channel number.

MTLTH. [MTAPE 200]

Function

Sets a flag to read the next record on the given device at low threshold (TM10 only). Use FILOP. to perform MTLTH. on an extended I/O channel.

Calling Sequence

```
MTLTH.  channo,
return
```

where: channo is the number of a channel initialized for a magtape device.

Return

The monitor sets a flag to read the next record from the given device at low threshold.

Examples

```
MTLTH.  5,
```

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return when none exists.

Forgetting to include a comma after the channel number.

MTREW. [MTAPE 1]

Function

Rewinds a magtape. Use FILOP. to perform MTREW. on an extended I/O channel.

Calling Sequence

```
MTREW.  channo,
return
```

where: channo is the number of a channel initialized for a magtape device.

Return

The monitor rewinds the tape on the specified device.

Examples

```
MTREW.  5,
```

Rewind the magtape associated with channel 5.

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return when none exists.

Forgetting to include a comma after the channel number.

MTSKF. [MTAPE 16]

Function

Skips forward one file on a magtape device. Use FILOP. to perform MTSKF on an extended I/O channel.

Calling Sequence

```
MTSKF.  channo,
return
```

where: channo is the number of a channel initialized for a magtape device.

Return

The monitor skips forward one file on the specified device, using a series of skip record operations.

Examples

```
MTSKF.  7,
```

This call skips over a file on the magtape associated with channel 7.

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return when none exists.

Forgetting to include a comma after the channel number.

MTSKR. [MTAPE 6]

Function

Skips forward one record on a magtape device. Use FILOP. to perform MTSKR. on an extended I/O channel.

Calling Sequence

```
MTSKR.  channo,
return
```

where: channo is the number of a channel initialized for a magtape device.

Return

The monitor skips forward one record on the specified device.

Examples

```
MTSKR.  7,
```

This call skips a record on the magtape associated with channel 7.

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return when none exists.

Forgetting to include a comma after the channel number.

MTUNL. [MTAPE 11]

Function

Unloads a magnetic tape. Use FILOP. to perform MTUNL. on an extended I/O channel. If the drive is under the control of MDA (under GALAXY Version 4.1 and later), this call only rewinds the tape.

Calling Sequence

```
MTUNL.  channo,
return
```

where: channo is the number of a channel initialized for a magtape device.

The MTUNL. call 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 TOPS-10/TOPS-20 SPEAR Manual) is written in the following format:

```
Drive number (in the form 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 your terminal (if WATCH MTA is set) in the following format:

```
[MTxn: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 Ø, the information pertaining to READ will not be printed.

When d, e, and f are Ø, the information pertaining to WRITE will not be printed.

To prevent this message from being printed, you can use SETUUO, or type the .STWTC function of the following monitor command:

```
.SET WATCH NO MTA
```

Return

The monitor rewinds the tape on the specified device.

Examples

```
MTUNL.  7,
```

This call rewinds the tape associated with channel 7.

Related Calls

FILOP., TAPOP.

Common Errors

Including an error return when none exists.

Forgetting to include a comma after the channel number.

MTWAT. [MTAPE Ø]

Function

Stops program execution until all spacing and I/O operations for a magnetic tape device are completed. Use FILOP. to perform MTWAT. on an extended I/O channel. Your program should execute this call after all tape-positioning operations.

Calling Sequence

```
MTWAT. channo,
return
```

where: **channo** is the number of a channel initialized for a magtape device.

Return

The monitor resumes execution at **return** after all spacing and I/O operations for the specified device are completed.

Related Calls

TAPOP.

Common Errors

Including an error return.

MVHDR. [CALLI 131]

Function

Allows you to move the buffer ring control block for an initialized channel from one location to another. This move is accomplished by changing the monitor's pointer to the ring control block.

Calling Sequence

```
MOVEI   ac,channo
MOVE    ac+1,[XWD outring,inring]
MVHDR.  ac,
  error return
normal return
```

where: **channo** is the number of an initialized channel.

outring is the new address of the output buffer ring control block, or Ø.

inring is the new address of the input buffer ring control block, or Ø.

outring and **inring** must be in your current low segment.

If you give Ø as the address of either buffer ring control block, the address is not changed.

Normal Return

The pointers to the specified control blocks are changed.

Error Return

The following error code is returned in the ac:

Code	Symbol	Error
1	MVHDR%	Channel not initialized.

If you specify an illegal address, the monitor halts your job and displays the following error message:

```
?Illegal address in UUO at user PC nnnnnn
```

NETOP. [CALLI 226]

Function

Indicates the node name and port name to which a terminal is connected.

Calling Sequence

```
        XMOVEI  ac,arglst
        NETOP.  ac,
          error return
        normal return
```

where: arglst is the address of the argument list described below:

Word	Symbol	Meaning
Ø	.NOFCN	Contains the length of the argument block in the left half, and a function code in the right half. You supply this information. The only valid code is .NOGDI, function 1. .NOGDI obtains information about the specified terminal's connection.
1	.NOFLG	Returns flags that indicate how your terminal is connected. If no flag is returned, the TTY is on a local line. Flags are:
2	.NODEV	The SIXBIT device name, UDX, or open channel number of the TTY. You supply this information.
3	.NODCH	Returns the physical characteristics of the TTY, in the same format as the return from a DEVCHR monitor call. Refer to the DEVCHR description for more information.
4	.NODTY	Returns the physical properties of the TTY, in the same format as the return from a DEVTYP monitor call. Refer to the DEVTYP description for more information.
5	.NONOD	The user-supplied address of a string block that contains the node name string on return. The left half of the string block's first word contains the length of the returned string. The block will contain up to 16 characters of node name.

Flag	Symbol	Meaning
Ø	NO.ANF	The TTY is on an ANF-1Ø node.
1	NO.DCN	The TTY is hosted through DECnet.
2	NO.LAT	The TTY is on a LAT terminal server.

Word	Symbol	Meaning
6	.NOPNM	The user-supplied address of a string block which contains the port name string on return. The left half contains the length of the returned block. ANF port names are returned as TTYnnn, where nnn is the node-local line number in octal. CTERM returns nnnnnn, which is a left-justified (octal) line number. NRT does not report a port name. LAT port names can be a maximum of 16 characters.

Normal Return

Information about the TTY's connection is returned as indicated in the argument block description.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	NOADC%	An address check was encountered.
2	NOILF%	An illegal function code was specified.
3	NOLTS%	The argument list is too short.
4	NONSD%	No such device.
5	NODNC%	The specified device is not connected.
6	NONTY%	Device is not a TTY.

Example

Example of the NETOP.UUO, which indicates where your own job's TTY is connected.

```
          SETO    T1,                     ;Refers to your job
          TRMNO.  T1,                     ;Get the TTY's UDX
            HALT                          ;
          MOVEM   T1,ARGBLK+.NODEV        ;Put into arg block
          MOVEI   T1,ARGBLK               ;Point to arg block
          NETOP.  T1,                     ; or CALLI AC,226
            HALT                          ;

ARGBLK:   XWD     7,.NOGDI    ;Length,,function code
ARGFLG:   BLOCK   1           ;Flags returned from NETOP.
ARGDEV:   BLOCK   1           ;User supplies device specifier
                              ; here
ARGDCH:   BLOCK   1           ;NETOP. returns DEVCHR UUO info here
ARGDTY:   BLOCK   1           ;NETOP. returns DEVTYP UUO info here
ARGNOD:   EXP     NODSPC      ;Address of string block to receive
                              ; node name
ARGPNM:   EXP     PORSPC      ;Address of string block to receive
                              ; port name
NODSPC:   XWD     Ø,5         ;Left half will receive string
                              ; length and 4 words (16 chars)
          BLOCK   4           ; worth of node name
PORSPC:   XWD     Ø,5         ;Left half will receive string length
          BLOCK   4           ; and 4 words worth of port name
```

NODE. [CALLI 157]

Function

Performs miscellaneous functions associated with ANF-1Ø network nodes.

Calling Sequence

```
        MOVE    ac,[XWD fcncode,addr]
        NODE.   ac,
          error return
        normal return
        . . .
addr:   EXP len
        first argument
        . . .
        last argument
```

Where: **fcncode** is one of the function codes described below.

addr is the address of the argument list.

len is the length of the argument list (including this word); and the words up through last argument are arguments for the specified function.

The function codes and their meanings are:

Code	Symbol	Function
1	.NDALN	Reserved for use by DIGITAL.

2	.NDRNN	Given either a node number or a node name, returns the other in the ac. The argument list for .NDRNN is:
		
		`addr:  EXP  2`
		`node-id`
		
		Where node-id is the SIXBIT name or the octal node number of the node. If you specify a node name, the node number is returned in the ac. If you specify the node number, the node name is returned in the ac.

3	.NDSSM	Sends special network station control (maintenance) messages. This function requires that the calling job be logged-in under [1,2] or have POKE. privileges. The argument list for .NDSSM is:
		
		`addr:  XWD  seconds,4`
		`node-id`
		`XWD  send-bytes,send-buffer`
		`XWD  receive-bytes,receive-buffer`
		
		where seconds is the number of seconds to wait before a timeout error (NDTOE%) occurs. This field is ignored if an input buffer is not specified; the default is eight seconds if Ø is specified.

Code	Symbol	Function

node-id is the SIXBIT node name or octal node number of the node.

send-bytes is the number of 8-bit bytes to be sent.

send-buffer is the address of the first byte of the message.

receive-bytes is the number of bytes in the receive buffer. When the receive buffer is filled, the monitor will set receive-bytes to the number of bytes actually stored in the buffer.

receive-buffer is the address of the buffer to store the response.

If the value for both receive-bytes and receive-buffer is Ø, the monitor returns control to your program without waiting for a response from the node.

4 .NDRBM Receives bootstrap messages from a remote node. This function requires that the calling job be logged-in under [1,2] or have POKE. privileges. The argument list for .NDRBM is:

```
addr:   EXP     4
        Ø               ;returned node number
        Ø               ;not used
        XWD     count,addr
```

where count is the number of 8-bit bytes to be received.

addr is the first address of the buffer. If there is a boot message to be read, the following occurs:

1. The second word of the argument block is filled in with the number of the node that sent the boot request station control message.

2. The boot request message is copied into the input buffer.

3. The count field is updated to reflect the actual number of bytes stored.

Code	Symbol	Function
5	.NDRCI	Returns the number of devices at each node for a list of device types. The argument list for .NDRCI is:

```
addr:   EXP     len
        node-id
        EXP     Ø       ;reserved
        BLOCK   buflength
```

where **len** is the length of the argument list (**len**-1 = number of following arguments).

node-id is the SIXBIT name of the node or the octal node number.

The word following the **node-id** must be zero because it is reserved for use by DIGITAL.

buflength is the number of words to reserve for the returned data.

Your program must supply the device types in the right half of each word starting at addr+3. The device types are returned by the DEVTYP call and are documented under that call. The monitor returns, in the left halves of these words, the count of devices whose type is given in the right half. Each word returned will appear as:

```
XWD device-count,device-type
```

6	.NDOUT	Obsolete.
7	.NDIN	Obsolete.
1Ø	.NDTCN	Connects remote terminals to the local system. The argument list for .NDTCN is:

```
addr:   EXP 2
        XWD node,line
```

where **node** is the octal number of the node to which the terminal is connected.

line is the remote line number of the terminal to be connected. On a normal return, the monitor returns the SIXBIT terminal number in the ac. The normal return is taken if the terminal is connected to the system on which the program is running. Therefore, a normal return from this call does not mean that the terminal is connected to your job.

11	.NDTDS	Disconnects a remote terminal from the local system and, optionally, reconnects it to another host system. The argument list is:

```
addr:   EXP m
        SIXBIT/TTYnnu/
        EXP node-number
```

where m is the length of the argument block (either 2 or 3).

Code	Symbol	Function

TTYnnu is the terminal name of the local terminal to be reconnected or disconnected.

node-number is the number of the node to which you wish the terminal to be reconnected. This word is optional.

On a normal return, the terminal specified by **TTYnnu** is disconnected from the local system. If a **node-number** is specified, the terminal will be connected to that host. In this case, the action performed is the same as if you issued a SET HOST monitor command.

12 .NDLND Returns the list of defined nodes. The argument list is:

```
addr:   EXP n
        argl
        ...
        argn
```

where n is the length of the argument list.

On a normal return, the **ac** contains the number of known nodes and **argl** through **argn** contain the node numbers of the known nodes. If the argument block is not long enough to return the complete list of known nodes, the list is truncated.

13 .NDNDB Returns the specified type of information about a specified node. The argument list is:

```
addr:   EXP n
        node-id
        EXP sub-fcn-code
        argl
        ...
        argn
```

where n is the length of the argument block.

node-id is either an octal node number or a SIXBIT node name.

sub-fcn-code is the sub-function code that specifies the type of information to be returned. **argl** through **argn** are words that your program reserves for the information returned by the monitor.

Code	Symbol	Meaning
1	ND.NNM	Returns the number of the node.
2	ND.SNM	Returns the SIXBIT name of the node.
3	ND.SID	Returns the software ID as an ASCIZ string.

Code	Symbol	Meaning
4	ND.DAT	Returns the date the NODE software was generated as an ASCIZ string.
5	ND.LMA	Returns the last NCL (Network Command Language) message assigned (on output).
6	ND.LMS	Returns the last NCL message sent (on output).
7	ND.LAR	Returns the last NCL ACK received (on output).
1Ø	ND.LAP	Returns the last NCL ACK processed (on output).
11	ND.LMR	Returns the last NCL message processed (on input).
12	ND.LMP	Returns the last NCL message received (on input).
13	ND.LAS	Returns the last NCL ACK message sent.
14	ND.MOM	Returns the counter for maximum outstanding messages.
15	ND.TOP	Returns a list of network link descriptors of the form: XWD cost,node where: cost is the line cost and node is the name of a node that is connected to the node specified in addr+1. One descriptor is returned for each neighboring node. A zero word signifies the end of the list.
16	ND.CNF	Returns the device configuration for a node in the following format: XWD obj-type,number where: obj-type is the NCL device type and number is the count of devices. One such descriptor is returned for each type of device on the node.
17	ND.CTJ	Returns the station control job number.
2Ø	ND.OPR	Returns the terminal number of the OPR terminal.

Code	Symbol	Function
14	.NDGNF	Sets or reads the "greeted" node flag. The format of the argument list is: addr: EXP 2 node-number arg where the node-number is the number of the node for which the "greeted" node flag is to be set and/or read. The flag may be specified in arg and is returned in arg. If the node-number is specified as Ø, the node number of the first "ungreeted" node is returned in addr+1. This function is intended to be used by privileged programs that perform a node-specific function to a node when it comes on line.

Normal Return

The function is performed.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	NDIAL%	Illegal argument list.
2	NDINN%	Illegal node name or node number.
3	NDPRV%	Not enough privileges.
4	NDNNA%	Node not available.
5	NDNLC%	Job not locked in core.
6	NDTOE%	Timeout error.
7	NDRNZ%	Reserved word is not zero.
1Ø	NDNND%	Channel not initialized or not a network device.
11	NDIOE%	I/O error occurred. The left half of ac contains I/O status bits. For a list of I/O status bits, see Volume 1.
12	NDNFC%	No free core.
13	NDIAJ%	In use by another job.
14	NDNMA%	No message available.
15	NDTNA%	Terminal not available.
16	NDNLT%	Not a legal terminal.
17	NDISF%	Illegal sub-function.
2Ø	NDRBS%	Receive buffer too small.
21	NDNUG%	No ungreeted nodes.
22	NDILN%	Illegal line number in station-control message.
23	NDADC%	Address check performed while reading or writing arguments.

NSP. [CALLI 205]

Function

The NSP. monitor call enables task-to-task communication between programs running on nodes in DECnet-10 networks. The communicating programs may be on separate nodes or the same node. For information on using this call, refer to Volume 1, Chapter 5.

Calling Sequence

```
        MOVEI   ac,addr
        NSP.    ac,
          error return
        normal return
        . . .
addr:   argument 1
        argument 2
        argument 3
```

Where addr is the address of an argument list appropriate to the function code given in bits 9-17 of argument 1. The function codes are:

Code	Symbol	Meaning
1	.NSFEA	Enter active state.
2	.NSFEP	Enter passive state.
3	.NSFRI	Read connect information.
4	.NSFAC	Accept connection.
5	.NSFRJ	Reject connection.
6	.NSFRC	Read connect confirm information.
7	.NSFSD	Synchronous disconnect.
10	.NSFAB	Abort connection.
11	.NSFRD	Read disconnect data.
12	.NSFRL	Release the channel.
13	.NSFRS	Read the channel status.
14	.NSFIS	Send interrupt data.
15	.NSFIR	Receive interrupt data.
16	.NSFDS	Send normal data.
17	.NSFDR	Receive normal data.
20	.NSFSQ	Set quotas.
21	.NSFRQ	Read quotas.

Code	Symbol	Meaning
22-23		Reserved for use by DIGITAL.
24	.NSFPI	Set PSI reasons for software interrupts.

Normal Return

The specified function has been performed and the ac is not changed.

Error Return

On an error return from NSP., one of the following error codes is returned in the ac:

Code	Symbol	Error
1	NSABE%	The argument block was formatted incorrectly.
2	NSALF%	An allocation attempt failed.
3	NSBCN%	An invalid channel number was specified.
4	NSBFT%	An illegal format type was specified in the process descriptor block.
5	NSCFE%	The connect block was formatted incorrectly.
6	NSIDL%	Interrupt data block pointed to a string block that was too long.
7	NSIFM%	Illegal flow control.
10	NSILF%	Illegal function code specified.
11	NSJQX%	Job quota exhausted.
12	NSLQX%	Link quota exhausted.
13	NSNCD%	No connect data to read.
14	NSPIO%	Percentage input out of bounds.
15	NSPRV%	Insufficient privileges to perform specified function.
16	NSSTB%	Segment size too big.
17	NSUKN%	Unknown node name was specified.
20	NSUXS%	Unexpected or unspecified state.
21	NSWNA%	Wrong number of arguments.
22	NSWRS%	Function call while connected in wrong state.
23	NSCBL%	Wrong length for connect block.
24	NSPBL%	Wrong length for process block.
25	NSSBL%	Wrong length for string block.
26	NSUDS%	Unexpected state: disconnect sent.
27	NSUDC%	Unexpected state: disconnect confirmed.
30	NSUCF%	Unexpected state: no confidence.
31	NSULK%	Unexpected state: no link.
32	NSUCM%	Unexpected state: no communication.
33	NSUNR%	Unexpected state: no resources.

NSP. error codes 34 to 56 correspond to DECnet disconnect codes.

34	NSRBO%	Rejected by object.
35	NSDBO%	Disconnected by object when running.
36	NSRES%	No resources.
37	NSUNN%	Unrecognized node name.
40	NSRNS%	Remote node shut down.
41	NSURO%	Unrecognized object.
42	NSIOF%	Invalid object name format.
43	NSOTB%	Object too busy.
44	NSABM%	Abort by management.
45	NSABO%	Abort by object.
46	NSINF%	Invalid node name format.
47	NSLNS%	Local node shut down.
50	NSACR%	Access control rejection.
51	NSNRO%	No response from object.

Code	Symbol	Error
52	NSNUR%	Node unreachable.
53	NSNLK%	No link.
54	NSDSC%	Disconnect complete.
55	NSIMG%	Image field too long.
56	NSREJ%	Reason for rejection was not specified.
57	NSBCF%	Invalid combination of NS.EOM and NS.WAI flags.
60	NSADE%	Address error.

NTMAN. [CALLI 206]

Function

Performs various functions for the network managment layer of the DECnet-10 network product. This call is used only by the NML program and is not intended for use by customer programs. The NTMAN. call is common to both DECnet-10 and DECnet-20 products, and therefore may change at any time without notice. This call requires JACCT, [1,2], or JP.POK privileges.

Calling Sequence

```
        MOVEI   ac,addr
        NTMAN.  ac,
          error return
        normal return
        . . .
addr:   len
        entity
        ptr to entity-id
        fcn-code
        info-type
        EXP     0
        ptr to data
        len of data
        BLOCK   1
```

where addr is the address of the argument block, which is formatted as:

Word	Symbol	Contents
0	.NTCNT	The number of words in the argument block (len).
1	.NTENT	The entity on which the action is to be performed. The types of entities are:
2	.NTEID	A byte pointer to an entity identification. The byte pointer must point to a node number, line number, or circuit number.
3	.NTFCN	Contains the function code. The function codes are:

Code	Symbol	Meaning
0	.NTNOD	Network node.
1	.NTLIN	Communications line.
2	.NTLOG	Reserved for DIGITAL use.
3	.NTCKT	Circuit.

Code	Symbol	Function
-2	.NTMAP	Returns node number for node name, or node name for node number.
-1	.NTREX	Returns the node-id of the local node.
0	.NTSET	Sets a parameter.

NTMAN. [CALLI 2Ø6]

Word	Symbol	Contents

Code	Symbol	Function
1	.NTCLR	Clears a parameter.
2	.NTZRO	Zeroes counters.
3	.NTSHO	Shows selected items.
4	.NTSZC	Shows and zeroes counters.
5	.NTRET	Returns a list of entities.
6	.NTEVQ	Removes an item from the event queue.

Word	Symbol	Contents
4	.NTSEL	Selection criteria for function. The following allow you to select the item on which the function is to be performed:

Code	Symbol	Meaning
-3 to -1		Used by .NTRET:
-3	.NTLOP	Loop.
-2	.NTACT	Active items.
-1	.NTKNO	Known items.
Ø to 4		Used by .NTSHO:
Ø	.NTSUM	Summary.
1	.NTSTA	Status.
2	.NTCHA	Characteristics.
3	.NTCOU	Counters.
4	.NTEVT	Reserved for DIGITAL use.

Word	Symbol	Contents
5	.NTQUA	Reserved for DIGITAL use.
6	.NTBPT	Byte pointer to data.
7	.NTBYT	Byte count for data.
1Ø	.NTERR	Returned information or error code.
11	.NTMAX	Quantity to place in .NTCNT.

Normal Return

The requested information is returned in the address pointed to by .NTERR, or data is changed according to the function code. On a successful return, error code 1 (NESUC%) is returned in .NTERR.

Error Return

The error code is returned in the ac and in the .NTERR offset into the argument block. Note that a successful return from the call places error code 1 (NESUC%) into the offset .NTERR into the argument block, and error NEADC% is returned only in the ac. The error codes for NTMAN. are defined with decimal values and are:

Code	Symbol	Error
1	NESUC%	Successful call.
-1	NEUFO%	Invalid function or option.
-2	NEIMF%	Invalid message format.
-3	NEPRV%	Insufficient privileges.
-4		Reserved for use by DIGITAL
-5	NEMPE%	Management program error.
-6	NEUPT%	Invalid parameter.
-7		Reserved for use by DIGITAL
-8	NEURC%	Invalid entity.
-9	NTINI%	Invalid entity identifier.
-10	NELCE%	Line communication error.
-11	NECWS%	Component in wrong state.
-12 to -14		Reserved for use by DIGITAL.
-15	NERES%	Resource error.
-16	NEIPV%	Invalid parameter value.
-17 to -19		Reserved for use by DIGITAL.
-20	NENRM%	No room, or slot already taken.
-21		Reserved for use by DIGITAL.
-22	NEPNA%	Parameter not applicable to entity.
-23	NEPVL%	Parameter value too long.
-24		Reserved for use by DIGITAL.
-25	NEOPF%	Operational failure.
-26	NEFNS%	Function not supported.
-27	NEIPG%	Invalid parameter grouping.
-28		Reserved for use by DIGITAL.
-29	NEPAM%	Parameter missing from argument list.
-30 to -46		Reserved for use by DIGITAL.
-47	NEADC%	Address check (returned in ac only).

OPEN [OPCODE Ø5Ø]

Function

Initializes a channel for I/O operation. Use FILOP. to assign an extended I/O channel.

Calling Sequence

```
        OPEN     channo,addr
          error return
        normal return
        . . .
addr:   argument list
```

Where: **channo** is the number of a channel.

addr is the address of the argument list. The argument list is:

Word	Symbol	Contents
Ø	.OPMOD	Flags and status bits. The I/O status bits are a set of 18 bits (right halfword) that reflect the current state of a file transmission. They are initially set by your program with the OPEN monitor call. Thereafter, the monitor sets the bits, but your program can test and reset them using any of several monitor calls.

Bits	Symbol	Meaning
Ø	UU.PHS	Only physical device names will be used in the search. All logical names defined by the job will be ignored.
1	UU.DEL	Disables error logging; only user-mode diagnostic programs may set this bit.
1	UU.FSP	Specifies a full SCNSER PTY (pseudo-terminal). That is, all terminal characteristics that are normally ignored for PTYs will be set and enforced. Refer to Chapter 15.
2	UU.DER	Disables error retry; only user-mode diagnostic programs may set this bit.
2	UU.BJP	If the given device is a PTY, and if the calling program is privileged, this bit specifies that jobs logging on this PTY are to be treated as batch jobs.

Bits	Symbol	Meaning
2	UU.DMR	Disables message re-assembly. This bit is used in ANF-1Ø intertask communication to allow the receipt of messages that do not have the EOM bit set. This flag should be set if UU.AIO is set for TSK devices. (Refer to Chapter 5.)
3	UU.AIO	I/O is nonblocking. This prevents the monitor from stopping your job to wait for I/O to be completed.
4	UU.IBC	Disables clearing of buffers after each output. Your program must also set BF.IBC in the .BFADR word of the buffer ring header. (This is applicable to buffered I/O only. Refer to Chapter 11.)
5	UU.SOE	Enables synchronization on each I/O error. The monitor does not perform more I/O until your program clears the error bits.
6	UU.RRC	Enables automatic rewrite of RIB on change. This bit pertains only to disk devices, causing the monitor to rewrite the file's RIB whenever a change to the file requires it. This is used to ensure file integrity in the event of system failure.
7	UU.LBF	Allows the use of large buffers (multiples of one block) for disk I/O on this channel.
8-14	UU.DEC	Reserved for use by DIGITAL.
15-17	UU.CUS	Reserved for use by customers.
18-21	IO.ERR	Error flags:

Flag	Symbol	Error
18	IO.IMP	Improper mode flag.
19	IO.DER	Error detected by device.
2Ø	IO.DTE	Hard data error.
21	IO.BKT	Block too large.

22	IO.EOF	End-of-file reached.
23	IO.ACT	I/O active.

Bits	Symbol	Meaning
24-29		Device-dependent flags (some flags are repeated for different devices). Refer to the appropriate device chapter in Volume 1 for more information.

Flag	Symbol	Meaning
24	IO.BOT	Beginning of tape encountered (magtape only).
24	IO.PTI	Subjob is in terminal input wait state (PTY only).
25	IO.EOT	End of tape encountered (magtape only).
25	IO.PTO	Subjob is in terminal output wait state (PTY only).
25	IO.ABS	Enable user break mask (terminals only).
26	IO.PAR	Parity of tape, where IO.PAR=1 for even parity. Odd parity (IO.PAR=Ø) is used only for EBCDIC labelled tapes (magtape only).
26	IO.PTM	Subjob is in monitor mode (PTY only).
26	IO.BKA	Break on all characters (terminals only).
27	IO.TEC	Truth in echoing mode (terminals only).
27	IO.MAI	Maintenance DMR mode (KDP and DTE devices only).
27-28	IO.DEN	Bit mask for tape density, where: Ø = standard 1 = 2ØØ 2 = 556 3 = 8ØØ For densities of 16ØØ and 625Ø, IO.DEN=Ø (magtape only). This flag is obsolete. Use TAPOP. to set tape density.
28	IO.SSD	Semi-standard data mode (DECtape devices only).

Bits	Symbol	Meaning

Flag	Symbol	Meaning
28	IO.SUP	Suppress echoing (terminals only).
29	IO.D29	DEC029 mode (card punch only).
29	IO.SIM	Super-image mode (card reader only).
29	IO.WHD	Write disk-pack headers (disk only).
29	IO.NSD	Non-standard data mode (DECtape devices only).
29	IO.SFF	Suppress form feeds (line printers only).
29	IO.NRC	Read with no reread check (magtape only).
29	IO.FCS	Obsolete (terminals only).
29	IO.LEM	Line editor mode (terminals only).

30 IO.SYN Synchronous input.

31 IO.UWC Uses user word count. By default this bit is not set, and the monitor computes the amount of data to be transmitted using the byte pointer in the buffer header. If this flag is set, however, the monitor uses the byte count. Meaningful for output only.

32-35 IO.MOD Data mode. In general, modes 0-14 are considered "buffered I/O modes" and modes 15-17 are "dump I/O modes." The possible values for the data mode (IO.MOD) in .OPMOD are:

Value	Symbol	Meaning
0	.IOASC	ASCII mode (for any device except display).
1	.IOASL	ASCII line mode (for any device except display).
2	.IOPIM	Packed image mode (terminal only).
3	.IOBYT	Byte mode (magtape device only).

Word	Symbol	Contents		
		Value	Symbol	Meaning
		4	.IOAS8	8-bit ASCII mode (terminal, pseudo-terminal, and line printers only).
		5		Reserved for use by DIGITAL.
		6-7		Reserved for use by customers.
		1Ø	.IOIMG	Image mode (for any except display device).
		11-12		Reserved for use by DIGITAL.
		13	.IOIBN	Image binary mode (for disk, DECtape, magtape, plotter, card device, or papertape device).
		14	.IOBIN	Binary mode (for same devices as .IOIBN).
		15	.IOIDP	Image dump mode (for display devices only).
		16	.IODPR	Dump record mode (for disk, DECtape, or magtape devices).
		17	.IODMP	Dump mode (for disk, DECtape, or magtape devices).
1	.OPDEV	SIXBIT physical or logical name or UDX of the device to be initialized on the channel or UDX.		
2	.OPBUF	Buffer addresses (used for buffered I/O only):		

Bits	Meaning
Ø-17	Address of the control block for the output buffers for the given channel. If buffered I/O is not to be used, specify Ø for the buffer control block address.
18-35	Address of the control block for the input buffers for the given channel. If buffered I/O is not to be used, specify Ø for the buffer control block address.

Normal Return

The specified channel is initialized.

Error Return

The monitor takes the error return if the specified device is in use, if the device does not exist, or if the device is restricted. To assign a restricted device, use the MOUNT monitor command before running the program, or use the .QUMNT function of the QUEUE. call.

Examples

```
;Subroutine to OPEN the 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,              ;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 file cannot be created
FILBAD: OUTSTR  [ASCIZ "?CANNOT CREATE DSK:DUMP.BIN
"]                              ;Print an error message
        EXIT                    ;Return to the monitor

OPNBLK: EXP     .IODMP          ;Select dump mode
        SIXBIT  /DSK/           ;Device name
        EXP     Ø               ;No buffers

FILE:   SIXBIT  /DUMP/          ;File name
        SIXBIT  /BIN/           ;File name extension
        EXP     Ø               ;Default protection
        EXP     Ø               ;Default directory

;Subroutine to write data in buffer
;Call with:
;       FILL BUFFER WITH DATA
;       PUSHJ   P,DMPOUT
;       RETURN HERE

DMPOUT: OUT     DSK,OUTLST      ;Write data
          POPJ  P,              ;No errors - Return to caller
        OUTSTR  [ASCIZ "?OUTPUT ERROR FOR DSK:DUMP.BIN
"]                              ;Output error message
        EXIT                    ;Return to monitor
```

OPEN [OPCODE Ø5Ø]

```
     ;Command list for output
     OUTLST: IOWD     BUFSIZ,BUFFER  ;Write BUFSIZ words from buffer
             EXP      Ø              ;End of command list

     BUFFER: BLOCK    BUFSIZ         ;Output buffer

     ;Subroutine to close out file
     ;Call with:
     ;       PUSHJ    P,DMPDON
     ;       RETURN HERE
     ;

     DMPDON: CLOSE    DSK,           ;Write the end of file
             STATO    DSK,IO.ERR     ;Are there any errors?
               POPJ   P,             ;No-return
             OUTSTR   [ASCIZ "?ERROR CLOSING DSK:DUMP.BIN
     "]                              ;Print error message
             EXIT                    ;Return to the monitor
```

Related Calls

FILOP., INIT

OTHUSR [CALLI 77]

Function

Determines whether other jobs are logged in under your project-programmer number (PPN).

Calling Sequence

```
OTHUSR  ac,
  alternate return
normal return
```

Normal Return

The ac contains your project-programmer number; the normal return occurs only if there are other jobs logged in under your project-programmer number.

Alternate Return

The alternate return is taken if no other jobs are logged in under your project-programmer number.

Examples

```
OTHUSR  T1,
  JRST  ONLY1
```

If other jobs are logged in under your project-programmer number, execution continues. If not, control is passed to ONLY1.

OUT [OPCODE Ø57]

Function

Transmits data from your job's physical memory area to the file selected for the given channel. Use FILOP. to perform an OUT UUO on an extended I/O channel.

Calling Sequence

```
OUT     channo,addr
normal return
  error return
```

Where: channo is the number of an initialized channel.

In buffered mode, addr contains the address of the .BFHDR (header) word of the buffer to be used. If you give addr as Ø, the next buffer is used.

In dump mode, addr is the address of the first word of the command list. See IN call.

Normal Return

The data in the buffer at addr+1 is transferred.

Error Return

If an error occurs, you should examine the I/O status bits to determine the cause of the error. Use the GETSTS call to obtain I/O status bits.

NOTE

If, while using non-blocking I/O, your program takes the error return with no error bits set, that indicates you have exhausted all of the output buffers. You must wait until a buffer becomes available. The program at this point should not attempt to store any more data based on the state of the use bit. Instead, keep trying the OUT call until it succeeds.

Examples

See Chapter 11.

Related Calls

FILOP., IN, INPUT, OUTPUT

Common Errors

- If the specified address is illegal, the monitor stops the job and prints:

  ?Address check for device YYYYYY:;UUO at user PC xxxxx

- Failure to supply a command list address in dump mode.
- Forgetting to initialize the channel.

OUTBUF [OPCODE Ø65]

Function

Sets up an output buffer ring with the specified number of buffers for the specified initialized channel. Use FILOP. to perform an OUTBUF for an extended I/O channel.

NOTE

The monitor allocates buffers in the user's address space starting at the location pointed to by the contents of .JBFF. This has no meaning in a non-zero section, unless that section is mapped to section Ø. Use the FILOP. monitor call to specify buffer starting addresses in a non-zero section.

Calling Sequence

```
OUTBUF channo,bufcnt
return
```

Where: **channo** is the number of an initialized channel.

bufcnt is the number of buffers to set up in the ring. If you give **buffers** as Ø, the monitor uses its default number of buffers for the ring. This default varies according to the device. For disks, the number of buffers is a MONGEN parameter that can also be set with the SET DEFAULT BUFFERS monitor command.

Return

The buffer ring is set up.

Related Calls

FILOP., INBUF

Common Errors

See the INBUF call for some common errors and their explanations.

OUTCHR [TTCALL 1,]

Function

Sends an ASCII character to the job's controlling terminal.

Calling Sequence

```
OUTCHR location
return
```

Where: **location** is the address of the word containing the output character; the ASCII code for the character is taken from bits 29-35 of **location**.

Return

The monitor takes an ASCII code from bits 29-35 of **location** and displays the character on the user terminal.

Related Calls

OUTSTR, TRMOP.

Common Errors

- Typing a comma after **location**.
- Assuming OUTCHR takes an immediate value in the effective address field.

OUTPUT [OPCODE Ø67]

Function

Sends data from memory to an initialized channel. Use FILOP. to perform an OUTPUT for an extended I/O channel. The OUTPUT monitor call is the same as the OUT monitor call, except that OUT takes the error return if any error bits are set in the I/O status word, and OUTPUT ignores the error bits and has only one return location.

Calling Sequence

```
OUTPUT channo,addr
return
```

Where: **channo** is the number of an initialized channel.

addr is one of the following:

- If the channel was initialized for dump mode, then **addr** is the address of an I/O command list.
- If the channel was initialized for buffered mode, then **addr** is the address of the second word of the next buffer to be used; if you give Ø (the normal case), the next buffer in the ring is used.

Return

Data is output to the device on the channel.

Related Calls

FILOP., IN, INPUT, OUT

Common Errors

Same as IN call.

OUTSTR [TTCALL 3,]

Function

Sends an ASCIZ string to the user terminal.

Calling Sequence

```
        OUTSTR  addr
        return
        . . .
addr:   ASCIZ/string/
```

Where: addr is the address of the ASCIZ string to be displayed on the terminal and string is the string to be sent.

Return

The string is displayed on the user terminal.

Examples

See OPEN call.

Related Calls

OUTCHR, TRMOP.

Common Errors

Typing a comma after addr.

PAGE. [CALLI 145]

Function

Manipulates pages and the data associated with those pages in your job's address space.

Calling Sequence

```
        MOVE    ac,[XWD fcncode,addr]
        PAGE.   ac,
          error return
        normal return
        . . .
addr:   len
        first argument
        . . .
        last argument
```

where: **fcncode** is one of the function codes described below.

addr is the address of the argument list.

len is the number of words that follow in the argument list; and the words up through **last argument** are arguments for the given function, usually page numbers of memory pages being manipulated. The value of **len** must be greater than 2 or the negative of a value greater than 2.

The **len** can be specified as a negative value. In this case, only one argument follows. That argument is the page number of the first page in a set, where the set contains that page plus the number of consecutive pages indicated by the value. For example, when **len** contains a negative value (such as -3), the argument (for example, page number 401), is the first of 3 consecutive pages (for this example, pages 401, 402, and 403), to be manipulated.

The pages you can specify are restricted by the following attributes:

- Page zero cannot be paged out or destroyed.
- If the high segment is sharable, it cannot be paged out.
- If the page is a SPY page, it cannot be paged out.
- If a page is locked in core, it cannot be paged out.

The function codes and their meanings are:

Code	Symbol	Function
Ø	.PAGIO	Swaps a page in or out. Pages swapped in are added to the working set; pages swapped out are moved to secondary storage.

Use one word in the argument list for each page to be swapped, or specify a negative list length to specify a set of consecutive pages. If you use more than one argument word, the page numbers must be in ascending order. Each argument word is in the form:

```
XWD     flags,pageno
```

where: **pageno** is the number of the page to be swapped (in the range Ø-511 on a KS, or Ø-16383 on a KL) and **flags** is one or both of the following:

Bit	Symbol	Meaning
Ø	PA.GAF	Swap the page out if this bit is set; swap it in if not set.
1	PA.GSL	Swap to slow swapping space if this bit is set; swap to fast space if not set.
2	PA.GDC	Suppresses error codes PAGCE%, PAGME%, PAGSC%, and PAGSM%.

1	.PAGCD	Creates or destroys a specified page. Use one argument word for each page to be created or destroyed. If you use more than one word, the specified pages must be in ascending order. Each argument word is of the form:

```
XWD     flags,pageno
```

where: **pageno** specifies the number of the page (in the range Ø-511 on a KS, or Ø-16383 on a KL) to be created or destroyed and **flags** is one or both of the following:

Bit	Symbol	Meaning
Ø	PA.GAF	Destroy the page if this bit is set; create the page if this bit is not set.
1	PA.GCD	Create the page on disk if this bit is set; create a page in the working set if this bit is not set.

Code	Symbol	Function
2	.PAGEM	Moves or exchanges a page. The page is moved from one virtual address to another, or two pages exchange locations. You cannot move a page to a location that is allocated to another page and you cannot exchange pages unless the source pages are allocated. Use one argument word for each page to be moved or exchanged. If you use more than one argument word, the specified pages must be in ascending order. Each argument word is of the form: <flag>+<source>B17+<destination>B35 where: **source** is the page number of the page to be moved, **destination** is the page number of the location to receive the page, and the following **flag** can be set:

Flag	Symbol	Meaning
Ø	PA.GAF	Exchange the pages if this bit is set; move the source page if this bit is not set.

Code	Symbol	Function
3	.PAGAA	Sets or clears the access-allowed bit for a page. The access-allowed bit may be changed for any page in the working set. If a page is accessed that has this bit off, a page fault occurs. Use one argument word for each page whose access-allowed bit is to be changed. If you use more than one argument, the specified pages must be in ascending order. Each argument word is of the form: XWD flags,pageno where: **pageno** is the page number of the page whose bit is to be changed, and flag is one or more of the following:

Flag	Symbol	Meaning
Ø	PA.GAF	Clear access-allowed for the page if this bit is set; set access-allowed if this bit is not set.
1	PA.GSA	Automatically sets access-allowed on page fault; dispatch to page handler on page fault if this bit is not set.
2	PA.GDC	Ignores fact that page does not exist, suppressing error code PAGME%.

Code	Symbol	Function
4	.PAGWS	This function returns a bit map of those pages in the current working set. In the PAGE. call, you specify the number of words that are to be returned. There is one bit for each possible page. 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 Ø through 35; Word 2 contains the bits associated with pages 36 through 71, and so on. The end of the bit map does not end on an integral word boundary, so the last word in the map is padded with zeroes. The bit map for another section begins on a new word.
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 (.PAGWS). If a bit in the map is set, the page associated with that bit is accessible. In the PAGE. monitor call, you specify the number of words in the bit map that are to be returned.
6	.PAGCA	Determines the type of access allowed for a given page. There is no argument block; instead, you specify the function code in the left half of the ac (bits Ø-17) and the page number in the right half of the ac (bits 18-35): [function,,page-number]. On a normal return, the monitor will set one or more of the bits listed below:

Bits	Symbol	Meaning
Ø	PA.GNE	Page does not exist.
1	PA.GWR	Writable page.
2	PA.GRD	Readable page.
3	PA.GAA	Access allowed.
4	PA.GAZ	Allocated page, but zero.
5	PA.GCP	Page cannot be paged out.
6	PA.GPO	Page is paged out.
7	PA.GHI	Page is in high segment.
8	PA.GSH	Page is sharable.
9	PA.GSP	Page is SPYing (mapped onto running monitor).
1Ø	PA.GLK	Page is locked in memory.
11	PA.GNC	Page is not cached (KL1Ø and KS1Ø only).
12	PA.GSN	Section does not exist.
13	PA.GVR	Page is virtual (SPY page).
14	PA.GIN	Page is in an indirect section, that is, a section mapped onto another section.
15		Reserved for use by DIGITAL.
16-2Ø	PA.GSC	Section is independent, that is, a section that another section is mapped onto. PA.GIN must be set for PA.GSC to be set.
21		Reserved for use by DIGITAL.
22-35	PA.GPN	Page number of the SPY page which the specified user page is SPYing on.

Code	Symbol	Function
7	.PAGCH	Changes the pages in a high segment, or creates a high segment from a contiguous collection of pages. The argument block is written in the following format: addr: Number of words following. addr+1: Number of pages to be remapped. addr+2: Start page number. addr+3: Destination page number. addr+3 is an optional word of the argument block. If not specified, page 400 is assumed. This function waits for all I/O to stop before creating the high segment. On a normal return, the specified pages are REMAPped into the high segment, which begins at destination page number. The error return is taken if all of the pages specified by start page number and number of pages to be remapped do not exist, or if a page included in the list already exists in your program's address space. If the number of pages specified is negative, those pages are remapped from the low segment to the high segment, and appended to the existing hi-seg. Note that a sharable high segment cannot be created or affected with this function code. If only one argument is given, the number of pages specified is deleted from the end of the high segment.
10	.PAGCB	Sets or clears the cache bit for the page. (This flag is automatically set if PA.GAF is on.) This function sets or clears the cache bit on a per-page basis (KL10 and KS10 only). The argument word format is as follows: Bit 0 — If this bit, PA.GAF, is set, the cache bit is set in the corresponding entry in the job's page map. If clear, the cache bit is clear. Bit 1 — Reserved. Bit 2 — This bit, PA.GDC, ignores the fact that a page doesn't exist, suppressing error code PAGME%. Bits 3-26 — Reserved. Bits 27-35 — The page number. If there is more than one argument word in the argument block, the page numbers specified in those words must be in ascending numeric order. The error return is taken if any of the following are true: o The function or call is not implemented. o A high segment page is specified in the argument list. o The argument list is not set up properly. o The job is not locked in core.

Code	Symbol	Function
11	.PAGSP	Allows your program to map an arbitrary set of pages from memory or from the monitor's virtual address space into the program's address space. Use one argument word for each page to be mapped. If you use more than one argument word, you must specify the pages in ascending order. The argument word is formatted as follows:

<flags>+<source>B17+<destination>B35

where **flags** are one or more of the following:

Bit	Symbol	Meaning
Ø	PA.GAF	Remove the page from the user's addressing space. If not set, add the monitor page to the user's addressing space at the specified page number.
2	PA.GDC	On a create, this bit will overlay an already existing page. On a delete, if the page does not exist, it is ignored and error code PAGME% is suppressed.

source is the page number of the source page. If UU.PHY is set in the PAGE. monitor call itself, **source** is a physical page in memory. If UU.PHY is not set, **source** is a monitor virtual address mapped through the executive page map.

destination is the page number of the page to be mapped into your address space.

This function requires that the calling job have PEEK privileges on all of core.

12	.PAGSC	Creates or destroys a specified section. Use one argument word for each section to be created or destroyed. For more than one word, the sections or arguments must be specified in ascending order. Each argument word is of the form:

XWD <flag>+<source>B17+<destination>B35

where flag is one of the following:

Bits	Symbol	Meaning
Ø	PA.GSF	Delete the section if this bit is on, create the section if this bit is off.
1	PA.GMS	On a create, map the sections specified in PA.GSS and PA.GDS together.

Code	Symbol	Function

Bits	Symbol	Meaning
2	PA.GDC	On a create, any existing section is emptied. On a delete, ignore a non-existant section.

If PA.GMS is set, give the source and destination sections using the following format:

Bits	Symbol	Meaning
4-17	PA.GSS	The section number of the source section.
18-22		Reserved for use by DIGITAL.
22-35	PA.GDS	The section number of the destination section.

13 .PAGBM

Returns a bit map that indicates whether specified page accessibility attributes belong to a certain page. If, in the return, the bit map is set on, the page has the specified attributes.

Each argument word is of the form:

```
EXP   count
EXP   attribute-settings
EXP   care-mask
EXP   starting-page-no
```

where:

count is the number of arguments.

attribute-settings is the word indicating the desired state of the given attribute. The page accessibility attribute bits are the same as those given for .PAGCA.

care-mask is the word specifying which bits of the attribute-settings word should be examined. Note that PA.GSC, the independent section bit, is checked only when PA.GIN is turned on in both .PAGCA and in the care mask in .PAGBM. Likewise, PA.GPN, the SPY page number, is checked only when PA.GSP is on in .PAGCA and the care mask in .PAGBM.

starting-page-no specifies the page number of the page that is mapped to bit Ø of the mask.

Code	Symbol	Function
14	.PAGAL	Determines the type of access allowed for a given page. The argument block is: EXP count EXP starting-page where: count is the number of arguments. starting-page is the starting page of the area in which information is to be returned. The bits returned are the same as for .PAGCA.

Normal Return

The specified function has been performed; the ac is unchanged.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
Ø	PAGUF%	Function not implemented.
1	PAGIA%	Illegal argument.
2	PAGIP%	Illegal page number.
3	PAGCE%	Page should not exist, but does.
4	PAGME%	Page should exist, but does not.
5	PAGMI%	Page should be in core, but is not.
6	PAGCI%	Page should not be in core, but is.
7	PAGSH%	Page is in sharable high segment.
1Ø	PAGIO%	Paging I/O error.
11	PAGNS%	No swapping space available.
12	PAGLE%	Core limit exceeded.
13	PAGIL%	Function illegal if page locked.
14	PAGNX%	Cannot allocate zero page with virtual limit zero.
15	PAGNP%	Not enough privileges.
16	PAGSC%	Section should not exist, but does.
17	PAGSM%	Section should exist, but does not.
2Ø	PAGIS%	Illegal section.

PATH. [CALLI 11Ø]

Function

Sets or reads a user's default directory path, reads the default directory path for a device or channel, or sets or reads pathological device name definitions. A pathological device name is a logical name defining a directory search path in the form:

```
dev:file.ext[UFD,SFD1,SFD2,...SFD5].
```

Refer to Section 12.6.5 for more information.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        PATH.   ac,
          error return
        normal return
        . . .
addr:   argument list
```

where: len is the length of the argument list. The value in this word must be at least 3. If you specify Ø, the length defaults to 3.

addr is the address of the argument list.

There are two types of argument blocks for the PATH. monitor call. Type 1 is used for reading and setting default directory paths for users, channels, or devices. Type 2 is used for reading or defining pathological names.

Argument Block Type 1

Argument Block Type 1 is used for reading and setting default directory paths for users, channels, or devices. This argument type is used for all functions (specified in .PTFCN), except functions -6 and -5 (.PTFRN and .PTFSN). The function codes are listed below.

Offset	Symbol	Contents
Ø	.PTFCN	A SIXBIT device name; or a job number in the left half and a function code or channel number in the right half. If you specify a device name, the monitor returns the default path for that device. If you specify a job number and function code, or a job number and channel number, the word is formatted as follows:

Bits	Symbol	Contents
Ø-17	PT.JBN	Job number. This job number defaults to your job number if not in the range of 1 to the highest legal job number.

Bits	Symbol	Contents
18-35	PT.FCN	Function code or channel number. The function codes and their meanings are listed below.

If you specify a channel number instead of a function code, the monitor returns the default path for the device currently open on that channel. If accessing a file that is open on the specified channel, the monitor returns the actual path for the file.

The function codes are:

Code	Symbol	Function
-6	.PTFRN	Reads a pathological name. Refer to Argument Block Type 2.
-5	.PTFSN	Sets a pathological name. Refer to Argument Block Type 2.
-4	.PTFRL	Returns an additional path to be searched when a file is not in your directory path. (For example, the monitor returns SYS, NEW, or LIB, if appropriate. See word .PTSWT below.)
-3	.PTFSL	Sets an additional path to be searched when a file is not found in your directory path. (See .PTFRL.) When you specify this function code, you must supply the following words in the argument block:

.PTFCN
.PTSWT
.PTPPN

Offset	Symbol	Contents

Code	Symbol	Function
-2	.PTFSD	Sets the default path to search if no path is specified.
-1	.PTFRD	Reads the default path that is searched if no path is specified.

1 .PTSWT Flags. The flags are only applicable to specific functions. All flags apply if you specify a device name or channel number in the previous word (.PTFCN).

Bits	Symbol	Meaning
25	PT.EDA	Returned to indicate that the device or channel number was both a pathological name and an ersatz device name. This flag is ignored on all Set functions.
26	PT.DLN	Returned to indicate that the device or channel specified in .PTFCN is a pathological name. Therefore, this flag is applicable only when a device or channel number is specified in .PTFCN.
29	PT.SLT	Returned to indicate the type of search list associated with a device or channel. The search list types are:

Code	Symbol	Type
Ø	.PTSLN	No search list has been associated with the specified device or channel.
1	.PTSLJ	Job search list.
2	.PTSLA	ALL search list.
3	.PTSLS	SYS search list. .PTSLS is applicable only when a device name or channel number is specified in .PTFCN.

Bits	Symbol	Meaning
3Ø	PT.IPP	Returned to indicate that the device specified in .PTFCN has an implied PPN, as in the case of an ersatz device name. The implied PPN is returned in the following word, .PTPPN.
3Ø	PT.DTL	Set to prevent any change to the status of LIB, allowing changes to NEW and SYS without changing LIB. This flag is useful only for function code .PTFSL.
31	PT.LIB	Returned to indicate that LIB (if defined for your job) will be searched on each structure in your job search list, after your default path for each structure in the job search list has been exhausted in the attempt to find a file. The LIB ersatz device name can be defined using function .PTFSL, or by setting flag PT.SEA in the LIB pathological name definition using function .PTFRN.
32	PT.NEW	Returned to indicate that, when SYS is specified or implied, the NEW area [1,5] will be searched before the SYS area [1,4].
33	PT.SYS	Returned to indicate that SYS (ersatz device name for [1,4]) will be searched on each structure in your job search list after your default path to each structure in the search list has been exhausted in the attempt to find a file.
34-35	PT.SCN	Controls searching of higher-level directories. (Similar to /SCAN switch to SETSRC program, but overrides the setting of /SCAN.) The values of this field can be Ø (use same scanning status as before the PATH. call), or one of the following:

Bits	Symbol	Meaning

Value	Symbol	Switch Setting
Ø		Use default setting.
1	.PTSCN	Turns scanning off.
2	.PTSCY	Turns scanning on.

The scanning status is returned if you specify a device or channel number in .PTFCN, or if you use function .PTFRD. The status is set using function .PTFSD. These bits are checked when the path block is given for LOOKUP, GETSEG, RUN, MERGE., and FILOP. calls.

34 PT.SNW — Set to indicate that NEW (ersatz device name for [1,5]) will be searched before [1,4] whenever SYS is specified or implied.

35 PT.SSY — Returned to indicate that SYS (ersatz device name for [1,4]) will be searched on each structure in your job search list after your default path to each structure in the search list has been exhausted in the attempt to find a file.

The following table lists the information that can be stored in .PTFCN and indicates the flags in .PTSWT that apply to each of the functions .PTFRD, .PTFSD, .PTFSL, and .PTFRL:

Table 22-1: PATH. Functions and Flags

Flag	Device or Channel	Functions			
		.PTFRD	.PTFSD	.PTFSL	.PTFRL
PT.EDA	X				
PT.DLN	X				
PT.SLT	X				
PI.IPP	X				
PT.DTL				X	
PT.LIB	X	X			
PT.NEW	X	X			
PT.SYS	X	X			
PT.SCN	X	X	X		
PT.SNW				X	X
PT.SSY				X	X

Offset	Symbol	Contents
2	.PTPPN	The PPN (UFD) of the path. For function .PTFRL, the library PPN, if any, is returned here. For function .PTFSL, the library PPN is set from this word unless PT.DTL is set in .PTSWT.
3	.PTSFD	The first level of SFD, as the SFD name, stored in SIXBIT. Subsequent words contain lower levels of SFDs. TOPS-1Ø allows up to 5 nested levels of SFDs, but MONGEN allows this value to be set at less than 5. You can obtain the maximum number of SFD levels allowed, from the right half of the item %LDSFD in GETTAB table .GTLVD.
4-1Ø	. . .	Name of the following SFD levels.
11	.PTMAX	Maximum length. Contains a Ø to end the PATH. block.

Argument Block Type 2

This argument block is used to read and define pathological names (logical names for directory paths) using functions .PTFSN to set the pathological name and .PTFRN to read the pathological name. The offsets into the argument block are:

Offset	Symbol	Contents
Ø	.PTFCN	Job number and function code in the following format:

Bits	Symbol	Contents
Ø-17	PT.JBN	Ignored for functions .PTFSN and .PTFRN.
18-35	PT.FCN	Function code. The function codes and their meanings are:

Code	Symbol	Function
-6	.PTFRN	Returns information (in .PTLNM) about the current pathological name or returns the next pathological name in the list of defined names.

PATH. [CALLI 11Ø]

Offset	Symbol	Contents

Code	Symbol	Function
-5	.PTFSN	Defines or deletes a pathological name. To delete a name, you must also set flag PT.UDF in .PTLNF and specify the name to be deleted in .PTLNM.

1 .PTLNF Pathological name flags:

Bits	Symbol	Meaning
Ø	PT.SEA	The directories included in the definition of this pathological name will be searched when a file is not found in the default directory path when you issue a LOOKUP monitor call. (This is similar to .PTFSL, but allows more flexibility.) PT.SEA can be set for only one pathological device. However, several directories can be specified for a single pathological name.
1	PT.UDF	Deletes the definition of the pathological name specified in .PTLNM. You must also specify function code -5 (.PTFSN) above.
2	PT.RCN	Returns data about the pathological name specified in .PTLNM when you specify function code -6 (.PTFRN). If this flag is Ø for function .PTFRN, the monitor returns, in .PTLNM, the next pathological name defined in the list.
3		Reserved for use by DIGITAL.

Bits	Symbol	Meaning
4	PT.OVR	Indicates that the file name and extension specified in the definition of the pathological device should override those in the argument block for LOOKUP/ENTER calls. Used with both .PTFRN and .PTFSN. For example, when FOO/OVERRIDE is defined as DSKA:FOO.DAT[1,2], a LOOKUP of FOO:BAR.DAT will not find BAR.DAT; it will find FOO.DAT. When PT.OVR is not set, the pathological name is used to define defaults. For example, when FOO is defined as DSKA:FOO.DAT[1,2], a LOOKUP for FOO:BAR would find BAR.DAT

2 .PTLNM For function .PTFRN, set this word to 0 to return the first pathological name in the list of names defined for your job, or the next path name in this word. For the .PTFSN function, this word contains the path name (in SIXBIT) that you wish to define or delete.

3 .PTLSB First word of the pathological name sub-block.

Each sub-block is in the format shown below. Offsets are from the start of the sub-block. The SFD list for the path begins at Word 5 and is terminated with a zero word. The zero word must not be past Word 12 (.PTLEL).

The entire list of sub-blocks must be terminated by two zero words following the last sub-block. These must be reserved in addition to .PTLEL.

Each path sub-block is formatted as:

Offset	Symbol	Contents
0	.PTNOD	Reserved to DIGITAL for future expansion.
1	.PTLSL	SIXBIT device (such as DSKB, DSK, ALL, or SSL).
2	.PTFIL	File name.
3	.PTEXT	File extension.
4	.PTLPP	PPN.
5	.PTLSF	Start of SFD list.
6-11		Subsequent SFD levels.
12	.PTLEL	Zero word, to terminate SFD list.
13	.PTLZT	First of the two-word zero terminator for the entire list of path blocks.

You can include as many sub-blocks as you wish, up to 77 (octal), except that the length of the entire list of sub-blocks (including the header) may not exceed 144 octal words.

PATH. [CALLI 11Ø]

Normal Return

For Read functions, the argument block is filled in; for Set functions, the function is completed.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
-1	PTNSS%	No such SFD as that specified in search list.
Ø	PTNDD%	Specified channel was not OPEN, or specified device was not a disk device.
1	PTTME%	Too many entries in PATH. block.
2	PTTMN%	Too many pathological names defined.
3	PTNSN%	Attempt to delete nonexistent name.
4	PTNFS%	No per-process free core.
5	PTANE%	Tried to define a pathological name that already exists.
6	PTNEN%	Non-existent pathological name used in argument block for .PTFRN and .PTFSN.
7	PTNSJ%	No such job as the job number you specified in the argument block.
1Ø		Reserved for use by DIGITAL.
11	PTNAI%	Invalid number of arguments specified. You must include 3 words in the block for calling sequence 1; 5 words in the block for calling sequence 2.

Examples

This example defines the following pathological name:

FOO/SEARCH=DSKB:[1Ø,1Ø,MON,NEW],DSKC:[1Ø,11,OLD]

The code to define the pathological name FOO is:

```
        MOVE    T1,[XWD ARGLEN,ARGLST]
        PATH.   T1,
          JRST  ERROR
        JRST    CONTIN
ARGLST: EXP     .PTFSN          ;Function code
        EXP     PT.SEA          ;/SEARCH attribute
        SIXBIT/FOO/             ;Logical path name to define
        EXP     Ø               ;Start of first group
        SIXBIT/DSKB/
        Ø                       ;File name
        Ø                       ;Extension
        XWD     1Ø,1Ø
        SIXBIT/MON/
        SIXBIT/NEW/
        EXP     Ø               ;Word terminating PATH spec
        EXP     Ø               ;Start of second group
        SIXBIT/DSKC/
        Ø                       ;File name
        Ø                       ;Extension
        XWD     1Ø,11
        SIXBIT/OLD/
        EXP     Ø               ;Word terminating PATH spec
        EXP     Ø               ;Two words terminating PATH block
        EXP     Ø
ARGLEN==.-ARGLST                ;Length of arg list
```

PEEK [CALLI 33]

Function

Returns the contents of any location in the monitor. The PEEK monitor call requires that your program have bit 16 (JP.SPA) or bit 17 (JP.SPM) set in the GETTAB table .GTPRV, or your program must have JACCT privileges. If you do not have the proper privileges, the ac is cleared.

Calling Sequence

```
        MOVE    ac,addr
        PEEK    ac,
        return
```

where: addr is the 30 bit address of the word in the monitor virtual address space to be returned.

If you set UU.PHY in this call, using the instruction:

```
PEEK ac,UU.PHY
```

the specified address is assumed to be a physical memory address instead of a virtual address.

Return

On return, the contents of the monitor location given by addr is returned in the ac. If the calling job does not have the required privileges, the monitor clears the ac.

Related Calls

PAGE., POKE, SPY

PERF. [CALLI 162]

Function

Allows privileged programs to perform system measurements over a period of time. The PERF. call works only on the KL1Ø processor. Note that only one job at a time may use the performance meter on each CPU. The PERF. functions are discussed in Chapter 1Ø.

Calling Sequence

```
        MOVE    ac,[XWD n,addr]
        PERF.   ac,
          error return
        normal return
        . . .
addr:   XWD     fcncode,faddr
        . . .
        XWD     fcncode,faddr
```

where: **n** is the number of function words specified in the argument block, which begins at **addr**.

addr is the address of the argument block. The argument block is a list of the functions to be enabled and the address of the argument list that defines each function, allowing you to specify multiple functions in a single monitor call.

fcncode is one of the function codes described on the following pages of this manual.

faddr is the address of the function code argument list. Each function must have a corresponding argument list. These argument lists are described with the function codes.

The function codes and their meanings are:

Code	Symbol	Function
1	.PRSET	Sets up the performance meter.
2	.PRSTR	Starts the performance meter.
3	.PRRED	Reads the performance meter.
4	.PRSTP	Stops the performance meter.
5	.PRRES	Releases the performance meter.
6	.PRBPF	Turns background PERF analysis off.
7	.PRBPN	Turns background PERF analysis on.

The argument list at **faddr** for the .PRSET function is:

Offset	Symbol	Contents
Ø	.PMLEN	Length of the argument list.
1	.PMCPU	CPU type:

Bit	Symbol	CPU Type
Ø	PM.PD6	PDP-6.
1	PM.KA	KA1Ø.
2	PM.KI	KI1Ø.
3	PM.KL	KL1Ø.
4	PM.KS	KS1Ø.

2 .PMMOD CPU number and mode:

Bits	Symbol	Meaning
Ø-17	PM.CPN	CPU number.
18	PM.MOD	Interval mode. If this bit is not set, a count of the enabled events (specified in following words) is kept. If on, the duration of the enabled event (in clock ticks) is kept.
19	PM.CLR	Clears performance meter counts. Resets the counters when the call is issued. If you clear this bit, the meter will be set but any values currently in the counters are left unchanged.

3 .PMCSH Cache enable flags:

Flag	Symbol	Meaning
Ø	PM.CCR	Count references.
1	PM.CCF	Count fills.
2	PM.EWB	Count EBOX writebacks.
3	PM.SWB	Count sweep writebacks.
4	PM.SYN	Synchronize performance and accounting meters.

4 .PMPIE Priority interrupt enable flags:

Flag	Symbol	Meaning
Ø	PM.PIØ	Enable for channel Ø (DTE).
1	PM.PI1	Enable for channel 1.
2	PM.PI2	Enable for channel 2.
3	PM.PI3	Enable for channel 3.
4	PM.PI4	Enable for channel 4.
5	PM.PI5	Enable for channel 5.
6	PM.PI6	Enable for channel 6.
7	PM.PI7	Enable for channel 7.
8	PM.NPI	Enable for no interrupt in progress.

Offset	Symbol	Contents
5	.PMPCE	Program counter enable flags:

Flag	Symbol	Meaning
Ø	PM.UPC	User-mode enable.
1	PM.XPC	Executive-mode enable.

Offset	Symbol	Contents
6	.PMMPE	Microcode probe enable flags:

Flag	Symbol	Meaning
Ø	PM.MPE	Enable microcode probe.

Offset	Symbol	Contents
7	.PMHPE	Hardware probe enable flags:

Flag	Symbol	Meaning
Ø	PM.PØL	Probe zero low.
1	PM.PØH	Probe zero high.
1Ø	.PMJOB	Job enable flag:

Value	Symbol	Meaning
-2	.PMNUL	Enable for null job.
-1	.PMSLF	Enable for calling job.

Offset	Symbol	Contents
11	.PMCHN	Channel enable flags:

Flag	Symbol	Meaning
Ø	PM.ECØ	Enable for channel Ø.
1	PM.EC1	Enable for channel 1.
2	PM.EC2	Enable for channel 2.
3	PM.EC3	Enable for channel 3.
4	PM.EC4	Enable for channel 4.
5	PM.EC5	Enable for channel 5.
6	PM.EC6	Enable for channel 6.
7	PM.EC7	Enable for channel 7.

The arguments at faddr and following for the .PRSTR, .PRRED, .PRSTP, and .PRRES functions are:

Offset	Symbol	Contents
Ø	.PMLEN	Length of the argument list.
1	.PMCPN	CPU number.
2	.PMHTB	High-order word of time-base.
3	.PMLTB	Low-order word of time-base.
4	.PMHPM	High-order word of performance counter.
5	.PMLPM	Low-order word of performance counter.
6	.PMHMC	High-order MBOX reference count.
7	.PMLMC	Low-order MBOX reference count.

The argument offsets for the .PRBPF and .PRBPN functions are:

Offset	Symbol	Contents
Ø	.PMLEN	Length of argument block.
1	.PMCPU	CPU type (same as .PRSET).
2	.PMMOD	CPU and flags (same as .PRSET).
3	.PMBPI	Sample interval in ticks.

Normal Return

For the .PRSET function, the performance meter is set.

For the .PRSTR function, the monitor starts the performance meter.

For the .PRRED function, the monitor has supplied the updated values for faddr+2 through faddr+7.

For the .PRSTP and .PRRES functions, the monitor stops the performance meter or releases the performance meter, respectively.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	PRCPU%	Invalid CPU specified.
2	PRNXC%	Nonexistent CPU specified.
3	PRMOD%	Improper mode specified.
4	PRSET%	Meter not set up.
5	PRUSE%	Meter already in use.
6	PRRUN%	Meter already running.
7	PRJOB%	Invalid job number.
1Ø	PRNRN%	Meter not running.
11	PRNIM%	Function not implemented.
12	PRFUN%	Invalid function code.
13	PRPRV%	Not enough privileges.

PIBLK. [CALLI 212]

Function

Returns the address of the 4-word interrupt control block for the current interrupt in progress on the Programmable Software Interrupt (PSI) system. Refer to Chapter 6 for more information about using the (PSI) system.

This call is used by generic interrupt processes that service multiple interrupt conditions. Note that this call will not generate an interrupt when UUO interrupts are enabled.

Calling Sequence

```
PIBLK.  ac,
  error return
normal return
```

Normal Return

On a successful return from this call, the address of the interrupt control block is stored in the **ac**.

Error Return

The call can take the error return with one of the following error codes stored in the **ac**:

Code	Symbol	Error
Ø	PSNIN%	The PSI system has not been initialized for this job.
1	PSNIP%	No interrupt is in progress.

Related Calls

DEBRK., PIFLG., PIINI., PIJBI., PIRST., PISAV., PISYS., PITMR.

PIFLG. [CALLI 216]

Function

The PIFLG. monitor call allows you to retrieve the PC flags of the highest level pending interrupt that have been stored in the monitor. Flags are stored in the monitor if you are using extended addressing format, set by the PS.IEA bit of the PIINI. monitor call.

Calling Sequence

```
{MOVEI   ac,.PSFRD          }
{MOVE    ac,[flags,,.PSFWT]}
 PIFLG.  ac,
   error return
  normal return
```

where: flags are one or more of the interrupt PC flags. The function codes are:

Code	Symbol	Function
Ø	.PSFRD	Read interrupt flags.
1	.PSFWT	Write interrupt flags.

Normal Return

The requested action is performed. Flags are returned in the ac.

Error Return

One of the following codes in returned in the ac:

Code	Symbol	Error
Ø	PSFNI%	PSI system not initialized.
1	PSFNP%	No interrupt in progress.
2	PSFEA%	Extended addressing format for PI system not in use.
3	PSFIF%	Illegal function code.

Related Calls

PIBLK., PIINI., PIJBI., PISAV., PISYS., PITMR.

PIINI. [CALLI 135]

Function

Initializes the programmable software interrupt (PSI) facility by clearing any old interrupts and storing the base address of the interrupt vector block. Refer to Chapter 6 for more information about using the PSI system.

Calling Sequence

```
        MOVE    ac, [flag + addr]
        PIINI.  ac,
          error return
        normal return
        . . .
addr:
```

where: flag is one or more of the following:

Bits	Symbol	Meaning
Ø	PS.UCS	Use the current (PC) section for the vector section. (IFIW) If PS.UCS is not set, addr is treated as a 3Ø-bit address.
1	PS.IEA	Use extended addressing format. A 3Ø-bit PC word is stored in the old PC location in the PSI block when an interrupt occurs. No flags are stored in the PC. Flags are stored in the monitor, and may be returned using the PIFLG. UUO.

addr is the base address of the first interrupt vector block. The interrupt vector block is a list of one or more sub-blocks, each of which is formatted as follows:

Offset	Symbol	Contents
Ø	.PSVNP	New program counter; this is the address of the interrupt service routine.
1	.PSVOP	Old program counter; this is the address of the next instruction after the instruction that was being executed when the interrupt occurred. If the instruction was a monitor call, .PSVOP contains the return address for the call; however, if the monitor terminated the call, .PSVOP contains the address of the call itself.
2	.PSVFL	Control flags, and either device condition flags or a non-I/O condition code.

The control flags are:

Bits	Symbol	Meaning
1	PS.VPO	Disable all interrupts; they can be reenabled by a PISYS. monitor call.

Offset	Symbol	Contents		
		Bits	Symbol	Meaning
		2	PS.VTO	Disable all interrupts of higher priority until the program gives a DEBRK. monitor call.
		3	PS.VAI	Allow control block to accept additional interrupts. Use DEBRK. to dismiss interrupts.
		4	PS.VDS	Dismiss any additional interrupt requests for this condition or device until this interrupt is dismissed (using DEBRK.).
		5	PS.VPM	Print any standard message that is relevant to this interrupt condition.
		6	PS.VIP	Interrupts are in progress for this interrupt control block. This flag is used by the monitor.
		17-35		The right half of .PSVFL contains the condition (reason) for the interrupt. These are divided into device I/O conditions and non-I/O conditions, and are described under the PISYS. monitor call.
3	.PSVIS	Interrupt status. When this auxiliary word is returned by a device I/O interrupt, it contains: udx,,file-status		

Normal Return

The program can use the PISYS. call to add or delete interrupt conditions.

Error Return

The call can take the error return with one of the following error codes stored in the ac:

Code	Symbol	Meaning
1	PIIBC%	Illegal bit or section number given.
2	PIADC%	Address check occurred.
3	PINFS%	The monitor does not have sufficient free core.

Examples

See Chapter 6.

Related Calls

DEBRK., PIFLG., PIJBI., PIRST., PISAV., PISYS.

PIJBI. [CALLI 175]

Function

The PIJBI. monitor call allows one job to interrupt another with a software interrupt. The interrupted job must be waiting for the interrupt; it cannot be busy handling a previous interrupt. The receiver enables cross-job interrupts by using the PISYS. UUO (non-I/O condition .PCJBI).

Calling Sequence

```
MOVE    ac,[XWD target,status]
PIJBI.  ac,
  error return
normal return
```

where: **target** is either the job context number of the job to interrupt, or the job number of the job to interrupt. **Status** is the status of the interrupt. The status word is described in Volume 1.

A -1 for the job number will interrupt the job that is executing the monitor call.

The job to be interrupted must be enabled for cross-job interrupts or else the call will fail. Note that this situation requires cooperation between two jobs, much like ENQ/DEQ or IPCF. If the target job is processing an interrupt, the sender must try again because requests are not queued.

Normal Return

The interrupted job will receive a word of the following form:

```
[source,,status]
```

where **source** is the job context number of the job that performs the PIJBI. call, and the **status** is the status that the job included in the call.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Meaning
Ø	PSJNE%	Job not enabled or the program specified an invalid job number.
1	PSJOP%	Job has an interrupt in progress. Try to interrupt again.

Related Calls

DEBRK., PIFLG., PIINI., PISAV., PIRST., PISYS.

PIRST. [CALLI 141]

Function

Restores the saved state of the interrupt facility. This does not restore any pending interrupts.

Calling Sequence

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

where: buffer is the address of the data saved by a PISAV. monitor call.

Normal Return

The state of the interrupt facility as saved by PISAV. is restored.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
Ø	PSNRS%	Not restoring what was saved.
1	PSNME%	Not enough monitor core to contain data base.

Related Calls

DEBRK., PIFLG., PIINI., PIJBI., PISAV., PISYS.

PISAV. [CALLI 140]

Function

Returns the monitor's data for the current state of the software interrupt facility. Use PIRST. to restore this data to current state.

Calling Sequence

```
        MOVE    ac,[XWD buflength,buffer]
        PISAV.  ac,
          error return
        normal return
        . . .
buffer: BLOCK   buflength
```

where: buflength is the length of the buffer for returned data (buflength = 2 + (3 * blocks)). The data is returned in a series of 3-word blocks, one block for each interrupt vector.

buffer is the address of the buffer.

Normal Return

The interrupt data is returned at buffer in the format:

Offset	Symbol	Contents
Ø	.PSSFC	Flags and count:

Bits	Symbol	Meaning
Ø	PS.SON	The interrupt facility is on.
1		Reserved.
2	PS.SEA	System using extended addressing.
3-17		Reserved.
18-35		Count of words returned. If error code PSBTS% (Ø) is returned, this count is the number of words required to save the current interrupt system.

Offset	Symbol	Contents
1	.PSSIV	Address of interrupt control block vector.
2	.PSSBL	Address of first 3-word argument block.

Each 3-word argument block is in the form:

Offset	Symbol	Contents
Ø	.PSECN	Condition or device.
1	.PSEOR	Offset,,reasons.
2	.PSEPR	Priority,,Ø.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
Ø	PSBTS%	Block too small.
1	PSBSZ%	Buffer size is Ø words.

Examples

See Chapter 6, Monitor Calls Manual, Volume 1.

Related Calls

DEBRK., PIFLG., PIINI., PIJBI., PIRST., PISYS.

PISYS. [CALLI 136]

Function

Controls the program interrupt facility during execution.

Calling Sequence

```
        MOVE    ac,[EXP flags+addr]
        PISYS.  ac,
          error return
        normal return
        . . .
addr:   SIXBIT/device/
        EXP     channo
        EXP     udx
        EXP     condition
        XWD vectoroffset,reasons
        XWD priority, Ø
```

where: flags are one or more of the function flags described below.

addr is the address of the argument list.

device is the SIXBIT physical or logical name of an initialized device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

condition is one of the non-I/O condition codes described below.

vectoroffset is the relative address of the control block for the interrupt (maximum value for this is stored in GETTAB table .GTCNF, item %CNMVO).

reasons are flags (described below) specifying the device conditions that can cause an interrupt.

priority is the priority level assigned to the interrupt. Priority Ø is the lowest level. Higher values indicate which events may interrupt other events. The maximum priority level is available in GETTAB table .GTCNF, item %CNMIP (normally 3).

The function flags and their meanings are:

Bits	Symbol	Meaning
1	PS.FOF	Turns off the interrupt facility.
2	PS.FON	Turns on the interrupt facility.
3	PS.FCP	Clears all pending interrupts.
4	PS.FCS	Clears all pending interrupts for a given device or condition.

Bits	Symbol	Meaning
5	PS.FRC	Removes the specified device or condition.
6	PS.FAC	Adds the specified device or condition.

The non-I/O condition codes and their meanings are:

Code	Symbol	Meaning
-1	.PCTLE	Time limit exceeded for batch job. The job run time (in milliseconds) is returned in the status word. You can change the job's time limit with the SET TIME monitor command.
-2	.PCTMR	Timer interrupt occurred. This condition is enabled with the PITMR. call.
-3	.PCSTP	CTRL/C received from user terminal. If the job was in terminal input wait state, bit Ø of the status word is set.
-4	.PCUUO	A monitor call is about to be processed; the status word contains the monitor call.
-5	.PCIUU	An illegal monitor call has been processed; the status word contains the monitor call.
-6	.PCIMR	An illegal memory reference occurred; the status word contains the effective address.
-7	.PCACK	An address check occurred; the status word contains the device name.
-1Ø	.PCARI	An exceptional arithmetic condition occurred.
-11	.PCPDL	A pushdown list overflow occurred.
-12	.PCNSP	The DECnet NSP. monitor call interrupt occurred. Refer to the NSP. UUO.
-13	.PCNXM	A reference to nonexistent memory occurred.
-14	.PCAPC	A line-frequency clock tick occurred while the job was running. Note that this does not mean an interrupt occurs on every clock tick, but only on those that occur while the job is being serviced by the CPU. The status word contains the date and time in universal format.
-15	.PCUEJ	A fatal error occurred for the job.
-16	.PCXEJ	An external condition caused a fatal error for the job.
-17	.PCKSY	A KSYS (end of timesharing) warning occurred; the status word contains the number of minutes left until KSYS.
-2Ø	.PCDSC	The dataset status changed.

Code	Symbol	Meaning
-21	.PCDAT	An ATTACH or DETACH monitor call was executed. For ATTACH the status word contains -1; for DETACH the status word contains the Universal Device Index for the terminal.
-22	.PCWAK	A WAKE monitor call was executed; the status word contains the job number of the waker.
-23	.PCABK	An address break condition occurred.
-24	.PCIPC	An IPCF packet is in your job's input queue; the status word contains the associate variable.
-25	.PCDVT	DECnet logging event occurred. Returns DR.xxx conditions, indicating that an event occurred that the DECnet management layer must handle.
-26	.PCQUE	One or more resources requested by an ENQ. monitor call is now available; the status word contains the inclusive OR of the request-ids of the granted requests.
-27	.PCNET	The ANF-10 network topology changed. You can obtain the state of the network using a NODE. monitor call.
-30	.PCJBI	Cross-job interrupt.
-31	.PCDTC	Date/time changed. The offset from the previous UDT is returned in the status word. This offset should be added to a previously stored UDT.
-32	.PCOOB	An out-of-band character was received.
-33	.PCRC1	Reserved for customer use.
-34	.PCRC2	Reserved for customer use.
-35	.PCSCS	SCS event.
-36	.PCETH	ETHERNET event.

The device interrupt reason flags and their meanings are:

Flag	Symbol	Device Condition
19	PS.RID	Input done.
20	PS.ROD	Output done.
21	PS.REF	End-of-file.
22	PS.RIE	Input error.
23	PS.ROE	Output error.
24	PS.RDO	Device off-line.
25	PS.RDF	Device full.
26	PS.RQE	Quota exceeded.

Flag	Symbol	Device Condition
27	PS.RWT	I/O wait.
28	PS.ROL	Device on-line.
29	PS.RRC	RIB has changed.
30	PS.RDH	Device hung.
31	PS.RSW	Reel switch.
32	PS.RIA	Input available.

Normal Return

The specified function is executed or the condition is enabled.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
0	PSTMA%	Although no bits in the left half of the ac require an argument list, the right half is nonzero.
1	PSNFS%	The left half of the ac is zero.
2	PSUKF%	Unknown function flag.
3	PSOOF%	Both the on and off function flags are set.
4	PSUKC%	Address check for addr.
5	PSDNO%	Device not initialized.
6	PSPRV%	Privilege failure.
7	PSIVO%	Invalid vector offset; not a multiple of 4, or too large. This value may not be larger than the limit given in the item %CNMVO in the GETTAB table .GTCNF.
10	PSUKR%	Nonzero value at addr+2.
11	PSPTL%	Priority too large. Highest priority allowed can be obtained from GETTAB table %CNMIP.
12	PSNRW%	Nonzero right halfword in control block.
13	PSPND%	Facility not initialized by PIINI.
14	PSARF%	Function flags for both "add" and "remove" are set.

Examples

See Chapter 6, Monitor Calls Manual, Volume 1.

Related Calls

DEBRK., PIFLG., PIINI., PIJBI., PIRST., PISAV.

PITMR. [CALLI 203]

Function

Enables the PSI system to interrupt after a duration of time. The PSI system must be initialized with the PIINI. call.

Calling Sequence

```
MOVE    ac,[XWD flag,interval]
PITMR.  ac,
  error return
normal return
```

where: **flag** is bit Ø, which can be set to indicate that the interval is specified in milliseconds. If bit Ø is off, the interval is assumed to be the number of seconds.

interval is the number of seconds to wait, then interrupt this job; if bit Ø is set, then the interval is taken as the number of milliseconds. If **interval** is specified as Ø, the default is 1 clock tick.

Restrictions

- The job must enable the timer condition (.PCTMR), using PISYS.
- A second request will override the first, because the job can have only one timer interrupt request pending at a time.
- Specified in milliseconds, the maximum interval is 262.143 seconds.
- In seconds, the maximum interval is 1 hour, 12 minutes, 49 seconds (at 6Ø Hz); or 1 hour, 27 minutes, and 22 seconds (at 5Ø Hz).

Normal Return

The program continues at the skip return and is interrupted with the timer condition after the specified interval.

Error Return

The program receives one of the following error codes in the ac:

Code	Symbol	Error
Ø	PSTNE%	Timer interrupts are not enabled for your job. Use PISYS. call to enable for these types of interrupts.
1	PSUFB%	Unknown function bit. The only bit that may be set in the left half of the ac is bit Ø.

Related Calls

DEBRK., PIFLG., PIINI., PIJBI., PIRST., PISAV., PISYS.

PJOB [CALLI 30]

Function

Returns the job number of your job.

Calling Sequence

```
PJOB    ac,
only return
```

Return

Your job number is returned in the ac.

Related Calls

CTLJOB

POKE. [CALLI 114]

Function

Changes the value of a word in monitor core. Using the POKE. call requires [1,2], JACCT, or JP.POK privileges.

Calling Sequence

```
        MOVE    ac,[XWD 3,addr]
        POKE.   ac,
          error return
        normal return
        . . .
addr:   monitor-addr
        oldvalue
        newvalue
```

where: **addr** is the address of the argument list.

monitor-addr is the address of the monitor word to be changed.

oldvalue is the value of the word before the change.

newvalue is to be the value of the word after the change. If you set UU.PHY using the instruction:

```
POKE. ac,UU.PHY
```

monitor-addr is assumed to be a physical memory address. If you do not set UU.PHY, **monitor-addr** is assumed to be an executive virtual address.

Normal Return

The value of the specified monitor word is changed.

Error Return

One of the following error codes is returned in the **ac**:

Code	Symbol	Error
Ø	PKNPV%	Your job is not properly privileged.
1	PKDIF%	The value of the given word is different from the value of **oldvalue**.
2	PKBAD%	The value of **monitor-addr** is not a valid monitor address.

Related Calls

PAGE., PEEK., SPY

QUEUE. [CALLI 201]

Function

Allows your program to communicate with system components. The actual communication is accomplished by QUEUE., using IPCF in your behalf, but the QUEUE. call allows you to communicate with system components using standard argument block formats. Some functions provided by system components are not accessible through QUEUE., and in these cases you must format your own IPCF. messages. For example, QUEUE. allows you to send messages to the GALAXY batch and spooling system, the accounting system, and site-specific components.

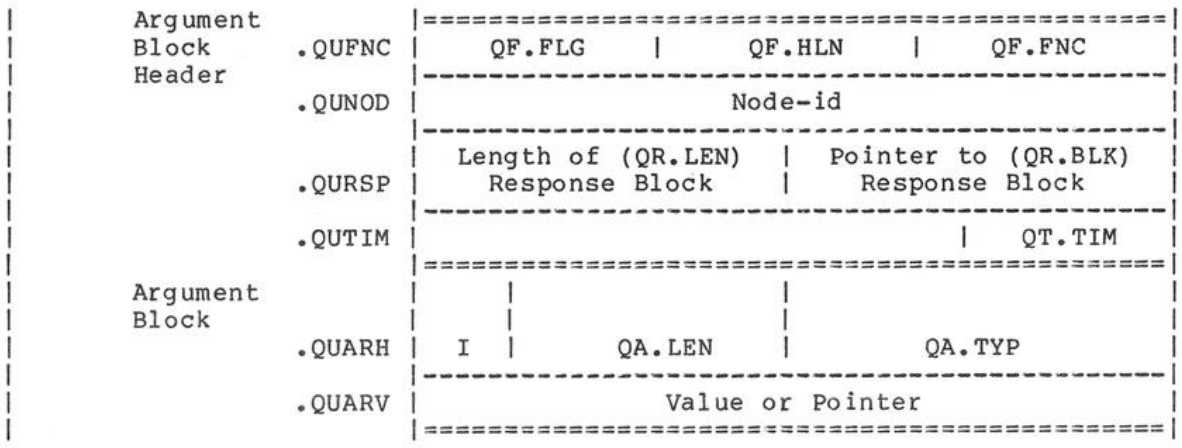
Many of the implemented functions relate directly to monitor commands. Therefore, information about these functions can be obtained from the Commands Manual.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        QUEUE.  ac,
          error return
        normal return
        . . .
addr:
```

where: len is the length of the argument list, and must be 5 or greater.

addr is the address of the argument list, which is formatted as follows:



Each word of the argument block header is described below:

Word	Symbol	Contents
Ø	.QUFNC	Flags, header block length, and function code. The flag field (QB.FLG) is in Bits Ø-11 of the left half of the word, in which you can set any of the following flags:

Bit	Symbol	Meaning
Ø	QF.RSP	User wants acknowledgement text returned at the address specified in Word 2 of this block, .QURSP.
1	QF.PIP	Your privileged job is invoking privileges to perform privileged QUEUE. call functions. You must set this flag to use privileged functions of QUEUE.
2	QF.NBR	Non-blocking return. The call will return automatically, and the function will be performed while your program runs. This flag is useful with function .QUWTO and requires privileges.

The length of the header block is contained in Bits 12-17, defined by the symbol QB.HLN.

The right half of word Ø contains the function code. Function codes are listed at the end of the argument header block description.

1	.QUNOD	ANF-1Ø network node identifier. You must specify whether the function is to be performed at the central site (where the program is running) or at a remote station. If this word is Ø, the central site is assumed. If you set this word to -1, the located node (defined by a LOCATE command or monitor call) is used.
2	.QURSP	Pointer to the first word of a block reserved for acknowlegement response from the system component. Data is read from this word if QF.RSP=1 in the function word. If non-zero, the .QURSP word must contain the length of the response block in the left half (QU.LEN) and the address of the first word of the response block in the right half (QU.BLK).
3	.QUTIM	This optional header word contains the maximum number of seconds to wait for a response. If the time is exceeded, the call returns error code 11. If this word contains Ø, or is non-existent, there is no implied time limit on the request.

The function codes that you can specify in Word 0 (.QUFNC) are:

Code	Symbol	Function
-n		Reserved for use by customers.
1	.QUPRT	Prints a file.
2	.QUCDP	Punches a file on cards.
3	.QUPTP	Punches a file on paper tape.
4	.QUPLT	Plots a file.
5	.QUBAT	Processes the file under BATCON, the batch controller.
6	.QUALC	Allocates a volume set.
7	.QUDAL	Deallocates a volume set.
10	.QUMNT	Mounts a volume set.
11	.QUDIS	Dismounts a volume set.
12	.QUWTO	Writes to operator.
13	.QUWTR	Writes to operator with reply.
14	.QUVAL	Validates an account.
15	.QUMAE	Makes an accounting entry by sending a message to the ACTDAE program. Refer to ACTSYM.MAC for the format of accounting entries.

The argument header block is followed by the one or more argument blocks. These argument blocks each consist of a header word and one or more data words.

The header describes the data word. If Bit 0 of the header word is set, the data words contain the value of the argument. If however, Bit 0 in the header word is clear, the data word contains the address of the data block. The remainder of the header word provides the length and type code.

The argument block, therefore, is formatted as follows:

Word	Symbol	Contents
0	.QUARH	Contains information about the data word(s). Specifically:

Bits	Symbol	Contents
0	QA.IMM	If this bit is set, data for the function starts at .QUARV. If this bit is clear, the data word contains the address of the data block (IFIW or GFIW).

Word	Symbol	Contents

Bits	Symbol	Contents
6-17	QA.LEN	This field contains the number of words in the data block. A value of zero is interpreted as one.
18-35	QA.TYP	This field contains the code for the type of data block that the data word points to.

Word	Symbol	Contents
1	.QUARV	Contains either the first word of the value of an immediate argument for the function, or contains the single-word address of a data block that contains the argument.

Each function can be described by one or more types of data blocks. The data blocks are listed below in the order of their type codes. Include all the data block types that specify information that is needed to perform the function you specified in .QUFNC.

Use the value of QA.IMM to specify the location of the data. If QA.IMM=1, the data words contain the data. If QA.IMM=0, the data word points to a data block.

For the allocation, mounting, dismounting, and deallocation of volume sets (functions 6-11), you must first specify Block Type 37 (.QBVSN) to specify the magtape volume set name or disk structure name. Then list the data blocks that contain or point to data about the request.

The data block types are:

Type	Symbol	Contents
10	.QBFIL	File specification block. You must include this type of data block for any function on a file (such as printing a file).

Offset	Symbol	Contents
0	.QBFSR	SIXBIT structure name.
1	.QBFFL	SIXBIT file name.
2	.QBFEX	SIXBIT extension. The right half of this word must be 0.
3	.QBFPP	UFD number (PPN).
4	.QBFS1	First level of SFD in SIXBIT.
5	.QBFS2	Second level of SFD (SIXBIT).
6	.QBFS3	Third level of SFD (SIXBIT).
7	.QBFS4	Fourth level of SFD (SIXBIT).
10	.QBFS5	Fifth level of SFD (SIXBIT).

Word	Symbol	Contents
11	.QBCOP	Number of copies block:

Offset	Symbol	Contents
0	.QBCNO	Number of copies of the file to be output.

Type	Symbol	Contents
12	.QBFRM	Forms type block:

Offset	Symbol	Contents
Ø	.QBFTY	Forms type in SIXBIT.

13	.QBPTP	Print file type block (for function .QUPRT only):

Offset	Symbol	Contents
Ø	.QBPCD	File format code. Include one of the following codes:

Code	Symbol	Meaning
1	.QBPAS	ASCII.
2	.QBPFR	FORTRAN.
3	.QBPCB	COBOL.
4	.QBPAI	Augmented image.
5	.QBPSA	Stream ASCII.
6	.QBP11	Eleven.
7	.QBPIM	Image.
1Ø	.QBP8B	8-bit ASCII.

14	.QBODP	Output disposition block. Specifies the fate of the file after the file is spooled.

Offset	Symbol	Contents
Ø	.QBODB	Output file disposition, one of the following:

Code	Symbol	Meaning
Ø	.QBODK	Preserve the file after processing it.
1	.QBODD	Delete the file after processing it.
2	.QBODR	Rename the file into the spooling area, effectively deleting it from the original area immediately.

15	.QBUNT	Unit type:

Offset	Symbol	Contents
Ø	.QBUDA	Device attributes in the left half. If .QBUPH is specified in the left half, you must specify the unit number in the right half of this word. The device attribute codes are:

Code	Symbol	Meaning
1	.QBULC	Lowercase printer.
2	.QBUUC	Uppercase printer.
3	.QBUPH	Physical device (specify unit number in right half).
4	.QBUGN	Generic device.

Type	Symbol	Contents
16	.QBAFT	Specifies the date and time at which the request should be processed.

Offset	Symbol	Meaning
0	.QBADT	Time in universal date/time format.

Type	Symbol	Contents
17	.QBLIM	Specifies the maximum number of units to which the job is limited. For printer requests, this is the number of pages. For batch processing, this refers to number of seconds of processing time, and so forth.

Offset	Symbol	Contents
0	.QBLNO	Number of pages, seconds, or appropriate limit.

Type	Symbol	Contents
20	.QBUNI	Specifies whether a batch job can be processed at the same time as others from the same PPN, or if only one batch job from this PPN can run at a time.

Offset	Symbol	Contents
0	.QBNVL	Uniqueness code:

Code	Symbol	Meaning
1	.QBNNO	Does not need to be unique.
2	.QBNYE	Must be unique.

Type	Symbol	Contents
21	.QBRES	Specifies whether a batch job should be restarted by the operator if the job is terminated unexpectedly (by a system failure, for example).

Offset	Symbol	Contents
0	.QBRVL	Restart code:

Code	Symbol	Meaning
1	.QBRNO	Do not restart the job.
2	.QBRYE	Restart the job.

Type	Symbol	Contents
22	.QBLOG	Specifies the circumstances under which to print a log file of the batch job.

Offset	Symbol	Contents
0	.QBLVL	Output type code:

Code	Symbol	Meaning
1	.QBLNL	Never print a log file.
2	.QBLLG	Always print a log file.
3	.QBLLE	Print a log file only when the batch job is terminated with an error.

Type	Symbol	Contents
23	.QBACT	Account string block. Specifies the ASCIZ account string for a batch job. Argument words .QBAC1 through .QBAC8 contain the ASCIZ string.
24	.QBFNC	Reserved for use by DIGITAL.
25	.QBNOD	Specifies the node at which the actual job processing should be done (destination node). Node number must be a remote non-host station in an ANF-10 network.

Offset	Symbol	Contents
0	.QBNND	Destination node number or SIXBIT node name.

26 .QBNAM User's name block:

Offset	Symbol	Contents
0	.QBNM1	First word of SIXBIT user name.
1	.QBNM2	Second word of SIXBIT user name.

27 .QBOID Specifies the owner's PPN:

Offset	Symbol	Contents
0	.QBOPP	Owner's PPN.

30 .QBNOT Specifies whether to notify the job when the request is finished.

Offset	Symbol	Contents
0	.QBNTL	Notify value:

Code	Symbol	Meaning
1	.QBNTY	Notify job when request is complete.
2	.QBNML	Reserved for use by DIGITAL.
3	.QBNJB	Reserved for use by DIGITAL.

31 .QBBLT Specifies the action to take on the batch log file:

Offset	Symbol	Contents
0	.QBBVL	One of the following:

Code	Symbol	Meaning
1	.QBBND	Append output log file to existing log file.
2	.QBBDE	Supersede existing log file.
3	.QBBSP	Spool log file to printer without preserving it in your area.

Type	Symbol	Contents
32	.QBJBN	Specifies the job name:

Offset	Symbol	Contents
Ø	.QBJNM	SIXBIT job name.

Type	Symbol	Contents
33	.QBCDI	Contains the batch job's default path block for batch requests.

Offset	Symbol	Contents
Ø	.QBCPP	PPN.
1	.QBCS1	First word of PATH block.
2	.QBCS2	Second word of PATH block.
3	.QBCS3	Third word of PATH block.
4	.QBCS4	Fourth word of PATH block.
5	.QBCS5	Fifth word of PATH block.

Type	Symbol	Contents
34	.QBNTE	Specifies a note to include on output header pages.

Offset	Symbol	Contents
Ø	.QBNM1	1 to 6 SIXBIT characters.
1	.QBNM2	1 to 6 SIXBIT characters (maximum of 12 characters).

Type	Symbol	Contents
35	.QBBGN	Specifies the page number of the file to begin printing, or the line number or tag in a batch file where processing should begin.

Offset	Symbol	Contents
Ø	.QBBPN	Beginning page number (for printing), line number (for batch jobs), or tag (in SIXBIT) at which to begin processing a batch job.

Type	Symbol	Contents
36	.QBPRI	Specifies the relative priority of the request. Unprivileged users can specify priorities between 1 and 2Ø, and privileged users can specify a priority in the range of 1 to 62. These limits can be changed by GALGEN, the GALAXY generation program. If you specify priority Ø or 63, the default priority is assumed.

Offset	Symbol	Contents
Ø	.QBPVL	Priority value (1 to 62).

Type	Symbol	Contents
37	.QBVSN	Contains the ASCIZ volume set name. This block must precede all other mount-specific blocks when you perform a disk or tape mount.

Offset	Symbol	Contents
Ø	.QBVAS	Beginning of ASCIZ volume set name.

Type	Symbol	Contents
40	.QBMSG	Contains the WTO/WTOR message block:

Offset	Symbol	Contents
0	.QBMAS	Beginning of the ASCIZ message for the operator.

Type	Symbol	Contents
41	.QBTYP	Contains the privileged WTO/WTOR message type block. WTO sends message to operator without requiring response. WTOR requires response from operator.

Offset	Symbol	Contents
0	.QBTAS	Beginning of ASCIZ message for the operator.

Type	Symbol	Contents
42	.QBDEN	Specifies the tape density:

Offset	Symbol	Contents
0	.QBTDN	Tape density code. Refer to the .TFDEN function of the TAPOP. call.

Type	Symbol	Contents
43	.QBTRK	Specified the tape track code:

Offset	Symbol	Contents
0	.QBDRV	Tape track request code:

Code	Symbol	Meaning
1	.QBDR9	9-track tape.
2	.QBDR7	7-track tape.

Type	Symbol	Contents
44	.QBLTP	Specifies the tape label type.

Offset	Symbol	Contents
0	.QBLAB	Label type code. Refer to the .TFLBL function of the TAPOP. call.

Type	Symbol	Contents
45	.QBRMK	Specifies the remark text:

Offset	Symbol	Meaning
0	.QBREM	Start of ASCIZ remark to be sent to operator with request.

Type	Symbol	Contents
46	.QBVOL	Specifies the tape volume list:

Offset	Symbol	Meaning
0	.QBVLS	Start of list of SIXBIT tape volume identifiers. A maximum of 63 volumes is allowed.

Type	Symbol	Contents
47	.QBLNM	Specifies the volume set logical name:

Offset	Symbol	Meaning
Ø	.QBLGN	SIXBIT logical name for this volume set.

Type	Symbol	Contents
5Ø	.QBMFG	Specifies MOUNT/DISMOUNT flags, indicated by setting/clearing bits in the following word:

Offset	Symbol	Meaning
Ø	.QBMDF	Flags to control the MOUNT or DISMOUNT request:

Flag	Symbol	Meaning
Ø	QB.PAS	Add the structure to the passive search list (active search list is default).
1	QB.EXC	Exclusive/sharable access (sharable is default for disk, exclusive is default for tapes).
2	QB.NOC	Prevent files from being created on the volume set. (The default is to allow file creation.)
3	QB.DSK	This is a disk file structure request.
4	QB.TAP	This is a magtape request.
5	QB.WLK	Write-lock the volume set (default for magtapes).
6	QB.WEN	Write-enable the volume set (default for disk).
7	QB.REM	Ask operator to remove the structure when you dismount it.
8	QB.SCR	Ask the operator to mount a scratch tape.
9	QB.ARD	Always recompute disk usage.

Type	Symbol	Contents
51	.QBAFN	Specifies the accounting daemon (ACTDAE) subfunction. .QBAFN is not intended for customer use.
52	.QBAET	Specifies the usage entry type. This type is not intended for customer use.
53	.QBTTY	Contains a SIXBIT terminal name. This block is not intended for customer use. .QBTTY is included in the IPCF message the monitor sends to ORION for a 'SEND OPR' command.
54	.QBFNT	Contains a six-word (maximum) argument block that specifies a font name. LPTSPL uses this ASCIZ string to locate the requested font file. Argument words are .QBFNØ to .QBFN5.

Type	Symbol	Contents
55	.QBEVT	Specifies an event to take place at the interval requested in .QBREP (Type 56). .QBEVØ is the argument word containing one of the following options: Code Symbol Event 1 EV.KSY KSYS 4 EV.ATO Time-of-day 5 EV.USG Usage file closure 6 EV.BIL Billing file closure 7 EV.OPR ORION log file closure
56	.QBREP	Repeats the event requested in .QBEVT at the interval given in the .QBRPØ argument word. .QBRPØ contains one of the following options: Flag Symbol Meaning Ø QB.NOW Event happens now. 1 QB.DLY Event happens daily. 2 QB.WKY Event happens weekly. 3 QB.TIM Event happens at specified time Or, you may specify .QBRPØ as day,,-1, where day is the day of the week when the event occurs (Wednesday=Ø, Thursday=1, and so on.)
57	.QBESW	Contains the event switch block. This block holds two words. .QBESD is the event-dependent switch value word, and .QBESI is the event-independent switch value word. You may specify one of the following flags for .QBESI: Flag Symbol Meaning Ø QB.FSF Failsoft option, which retains the event in the queue after a system reload. 1 QB.NFS No failsoft.
6Ø	.QBAST	Sets the OPR intervention bit to one of the following: Bit Symbol Meaning 1 .QBOIY Enable OPR intervention. 2 .QBOIN Disable OPR intervention.
61	.QBPRC	Sets the IBM /PROCESSING node.
62	.QBOPT	Specifies a SIXBIT batch option name. .QBOPØ is the offset to the option name.
63	.QBDIS	Specifies text to be printed for a DISTRIBUTION: header. .QBDIØ is the offset to the first word of ASCIZ data.

QUEUE. [CALLI 201]

Type	Symbol	Contents
64	.QBUSR	Specifies text to be printed for a USERNAME: header. .QBUS0 is the offset to the first word of ASCIZ data.
65	.QBUTY	Specifies a SIXBIT unit name, such as "LN01" to queue to an LN01 laser printer.

Normal Return

On the return from QUEUE., the IPCF messages have been sent to appropriate components. If you requested a response by setting QF.RSP, the following information is returned in the ac:

Bits	Symbol	Meaning
18	QU.RBT	Response from GALAXY was too long for reserved space (as specified in .QURSP) and had to be truncated.
19	QU.RBR	Response from GALAXY was returned.
23-35	QU.RBL	Contains the length of the returned response from GALAXY.

Error Return

The error codes and their meanings are:

Code	Symbol	Meaning
1	QUIAL%	Illegal argument list.
2	QUILF%	Illegal function.
3	QUNFS%	No monitor free core.
4	QUADC%	Address check.
5	QUCNR%	Component not running or has no system PID.
6	QUFER%	Fatal error returned from ORION.
7	QUSOC%	Invalid message from ORION.
10	QUNPV%	Insufficient privileges.
11	QUTMO%	Timeout limit exceeded.

REASSI [CALLI 21]

Function

Reassigns or deassigns a device for a job. Your program can reassign a device if the device is assigned to your job, or if it is not assigned to any job. Restricted devices cannot be reassigned by unprivileged jobs. The logical name assignment is also cleared, unless the calling job has JACCT privileges or is logged in under [1,2].

Calling Sequence

```
        MOVEI   ac,jobno
       {MOVE    ac+1,[SIXBIT/device/]}
       {MOVEI   ac+1,channo          }
       {MOVEI   ac+1,udx             }
        REASSI  ac,
        return
```

where: **jobno** is the number of a logged-in job to which the device is to be reassigned. Use -1 to indicate the current job or Ø to deassign the device.

device is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

Return

If the ac is unchanged on the return from the call, the device is reassigned as requested.

If the device is restricted and you specified Ø for jobno to deassign the device, it is returned to the system's pool of restricted devices.

A restricted device can be reassigned to an unprivileged job only by a privileged job.

On the return from this call, the monitor performs an implicit RELEAS monitor call for the device, unless you specified Ø or -1 for the **jobno**.

If the **ac** is cleared on the return, the **jobno** was not a valid job number.

If the **ac** contains -1 on the return, the device is not assigned to the specified job. The device is your job's controlling terminal, or the device name given is a duplicate of an existing logical device name.

If ac+1 is cleared on a return from the call, the device is not assigned to your job, or the device you specified was a disk or your job's controlling terminal.

Common Errors

- Forgetting that there is only one return location from the call.
- Attempting to assign a restricted device.

Related Calls

DEVLNM

RECON. [CALLI 202]

Function

Performs tasks to aid system reconfiguration and diagnosis. This call is not recommended for use by customer programs and requires [1,2] or JACCT privileges. It is used by the CONFIG system facility to take system snapshots, suspend the system, and other system-wide functions. To perform the functions offered by the RECON. call, use the CONFIG command level from the OPR program (documented in the TOPS-10 Operator's Command Language Reference Manual), because the functions must be performed in the correct order or the system will fail to continue.

Calling Sequence

```
        MOVE    ac,[fcncode,,addr]
        RECON.  ac
          error return
        normal return
addr:   argument-block
```

where: **fcncode** is one of the functions described below.

addr contains an argument block. The data in the argument block depends on the function code. For functions that do not require an argument, use 0 for **addr**.

The function codes, their meanings, and argument blocks are:

Code	Symbol	Meaning
0	.RCROM	Prevents execution of all jobs on the system except the job number specified in addr+1. Use -1 for current job. If the job number is 0, the function is to resume normal timesharing. The argument block for this call is:

```
        MOVE    ac,[.RCROM,,addr]
        RECON.  ac,
          error return
        normal return
        . . .
addr:   2       ;length of argument block
        jobno   ;contains -1 for current job
```

1	.RCSPN	Causes a suspension of the system. To accomplish this, the monitor writes a copy of memory to CRASH.EXE on disk, and the system is halted. This last action is taken as the result of the CONFIG command SUSPEND. This function does not require an argument block, so the calling sequence is:

```
        MOVE    ac,[.RCSPN,,0]
        RECON.  ac,
          error return
        normal return
```

Code	Symbol	Meaning
2	.RCCI7	Causes the monitor to perform a continuable stopcode (stopcode CI7), take a dump of memory, and continue automatically. This dump is called a "snapshot," and allows you to diagnose problems by obtaining a dump without halting the system. The .RCCI7 function does not require an argument block, so the calling sequence looks like:

```
        MOVE    ac,[.RCCI7,,0]
        RECON.  ac,
          error return
        normal return
```

3	.RCNAR	Clears and sets the DF.NAR bit in the DEBUGF word, which controls whether the system should automatically reload on non-continuable stopcodes. The DEBUGF word is defined in the monitor symbol file S.MAC, and is manipulated during analysis of system errors. By default, DF.NAR is set, and automatic reload is enabled. You can clear the bit to prevent auto-reload when debugging the system, but this function requires that you be logged in under [1,2].

The DF.NAR bit is set/cleared according to the second word in the argument block. If Word 1 of the argument is 0, the DF.NAR is cleared, and automatic reload is enabled (default state). If you place a non-zero value in Word 1, DF.NAR is set, and the system will not automatically reload on a non-continuable stopcode. The calling sequence for this function is:

```
        MOVE    ac,[.RCNAR,,addr]
        RECON.  ac,
          error return
        normal return
        . . .
addr:   2       ;length of argument block
        -1      ;to disable auto-reload
```

4	.RCBTX	Changes the BOOTXT command string to the command string you specify in the argument. The calling sequence for this function is:

```
        MOVE    ac,[.RCBTX,,addr]
        RECON.  ac,
          error return
        normal return
        . . .
addr:   n               ;length of argument block
        command-list    ;first word
```

where n is the length of the command string (in words) + 1. The command string cannot exceed 16 words, and cannot include line-feeds; therefore, the maximum value for n is 17. For information about the BOOTXT command string, refer to BTXLEN in the COMMON monitor source file.

Code	Symbol	Meaning
5	.RCRLD	Causes the system to be reloaded. The monitor will be reloaded from the file specified in BOOTXT, and a crash dump will be taken. You must be logged into [1,2] to use this function (JACCT privileges alone are not sufficient). This function causes an RLD stopcode and does not require an argument list. The calling sequence for this function is: MOVE ac,[.RCRLD,,0] RECON. ac, error return normal return
6	.RCRAC	Causes auto-configuration (AUTCON) to run on the specified CPU(s) to automatically configure disks and tapes into the monitor's data base. You must be logged in as [1,2] to use this function. The calling sequence for this function is: MOVE ac,[.RCRAC,,n] RECON. ac, error return normal return where n specifies the CPU number. If n = -1, AUTCON will run on all CPUs.
7	.RCDET	Detaches a CPU or a device as specified in the argument block. The calling sequence is as follows: MOVE ac,[.RCDET,,addr] RECON. ac, error return normal return addr: 2 ;length of argument block SIXBIT CPU name or SIXBIT device name
10	.RCATT	Attaches a CPU or device specified in the argument block. The calling sequence for .RCATT is: MOVE ac,[.RCDET,,addr] RECON. ac, error return normal return addr: 2 ;length of argument block SIXBIT CPU name or SIXBIT device name

Code	Symbol	Meaning

11	.RCMON	Sets a given range of memory on-line. The range is given in two words of the argument block. The first word specifies the first page of the range, and the second word indicates the first page beyond the range. The calling sequence is:
		
		`MOVE ac,[.RCMON,,addr]`
		`RECON. ac,`
		`  error return`
		`normal return`
		
		`addr: 3 ;length of argument block`
		`first page in range`
		`last page in range +1`

12	.RCMOF	Sets a given range of memory off-line. You specify the range the same way as for .RCMON. The calling sequence is:
		
		`MOVE ac,[.RCMOF,,addr]`
		`RECON. ac`
		`  error return`
		`normal return`
		
		`addr: 3 ;length of argument block`
		`first page in range`
		`last page in range +1`

13	.RCCPU	Returns the CPU accessibility mask. The mask indicates which CPU is using the specified device. The calling sequence for .RCCPU is:
		
		`MOVE ac,[.RCCPU,,addr]`
		`RECON. ac,`
		`  error return`
		`normal return`
		
		`addr: 2 ;length of argument block`
		`SIXBIT device name`
		
		The bit mask returned in the ac indicates which CPU is associated with the device. For example, Bit 35=CPU0, Bit 34=CPU1, and so forth.

14	.RCIOW	Indicates when I/O on the system is complete. The calling sequence is:
		
		`MOVSI ac,.RCIOW`
		`RECON. ac,`
		`  error return`
		`normal return`
		
		The return occurs after active I/O on the system has finished.

Code	Symbol	Meaning
15	.RCSDF	Sets bits in DEBUGF that cause the monitor to reload for a CPU, DEBUG, or JOB stopcode. This function is used by ORION for the CONFIG program. The calling sequence is:

```
        MOVE    ac,[.RCSDF,,addr]
        RECON.  ac,
          error return
        normal return

addr:   2       ;length of argument block
        DEBUGF bits
```

Code	Symbol	Meaning
16	.RCCDF	Clears DEBUGF bits so that the monitor takes a continuable dump on a CPU, DEBUG, or JOB stopcode. This function is not intended for customer use. The calling sequence is:

```
        MOVE    ac,[.RCCDF,,addr]
        RECON.  ac,
        error return
        normal return

addr:   2       ;length of argument block
        DEBUGF bits
```

Normal Return

The specified function is performed.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	RCIAL%	Illegal argument list.
2	RCNPU%	Not privileged.
3	RCNTS%	Timesharing already stopped on some CPU.
4	RCNIJ%	Illegal job number.
5	RCNCD%	This function cannot be performed.
6	RCNAC%	Address check.
7	RCNIS%	Illegal command string for BOOTXT.
10	RCICN%	Illegal CPU number.
11	RCNCR%	CPU still running.
12	RCNCA%	Can't attach disk.
13	RCNDS%	Device is spooled.
14	RCNAA%	Device is already attached.
15	RCNID%	Illegal device.
16	RCNDU%	Device is in use.
17	RCNND%	Can't detach disk.
20	RCNNL%	Can't set memory off-line.
21	RCNMM%	Can't remove monitor memory.
22	RCNTB%	Job or jobs too big.
23	RCNLJ%	Can't move locked job(s).

RELEAS [OPCODE Ø71]

Function

Releases an I/O channel. Use FILOP. to perform a RELEAS on an extended I/O channel.

Calling Sequence

```
RELEAS  channo,
return
```

Where: channo is the number of an initialized channel.

Return

The device is released, the channel is closed, any device-dependent operations are performed, and any enqueue locks are released.

If the given channel is not initialized, the monitor takes no action.

Examples

See Chapter 11, Volume 1.

Common Errors

Forgetting the comma after the channel number.

Related Calls

FILOP.

REMAP [CALLI 37]

Function

Moves the specified portion of a program's low segment into the high segment, discarding the old high segment from the user addressing space. The new low segment will be the previous low segment minus the amount remapped. This monitor call is used by the LINK program when you use the EXECUTE monitor command.

The .PAGCH function of the PAGE UUO has the same capability as REMAP, but is more flexible.

Calling Sequence

```
MOVE  ac,[XWD origin,addr]
REMAP ac,
  error return
normal return
```

where: addr is the highest address in the low segment (that is, the first address of the new high segment).

origin is the origin of the high segment.

The monitor waits until all I/O is completed in the low segment before executing the REMAP monitor call. Then the monitor rounds the address to the nearest core allocation unit (512 decimal words).

Normal Return

- The monitor stores the value of addr in the location .JBREL in the job data area.
- The monitor sets the left half of .JBHRL to zero (it deletes the previous high segment).
- The monitor stores the highest legal user address for the high segment in the right half of .JBHRL.
- The monitor changes the hardware mapping.
- The monitor sets the user-mode write-protect bit (the new high segment is non-sharable).
- The contents of the ac are preserved.

Error Return

The monitor takes the error return if any of the following conditions is found:

- A negative argument is specified.
- The requested remapping would cause the high and the low segments to overlap.
- 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 (that is, greater than or equal to 256K).

- The specified argument 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.
- The segment is locked in memory.

Related Calls

CORE, GETSEG, MERGE., PAGE.

RENAME [OPCODE Ø55]

Function

Performs one or more of the following functions:

- Alters file attributes, including the file name, file extension, and access privilege code of the file.
- Changes an SFD name.
- Deletes the specified file.
- Performs an implicit CLOSE.

Use FILOP. to perform a RENAME on an extended I/O channel.

Calling Sequence

```
RENAME  channo,addr
  error return
normal return
```

where: channo is the number of an initialized channel. If the channel is an extended channel, use FILOP. function .FORNM.

addr is the address of the argument list. The argument list is equivalent to that of LOOKUP and ENTER calls and is described in Section 11.13.

RENAME has two forms of argument block: the four-word block and the extended argument block. The short-form (4-word) argument list is described in Section 11.13.1. The extended argument list is described in Section 11.13.2. For DECtape files, refer to Chapter 13 for descriptions of the arguments.

The only way that your program can RENAME a file into or out of an SFD is to refer to an explicit path using the PATH. argument block (by including an [XWD Ø,addr] instruction as the PPN argument). If a RENAME is given that attempts to move a file into or out of an SFD without specifying an explicit path, it will take the normal return (assuming no other errors), but the file will not change directories.

To delete a file after all read references have been made, your program should specify the value of zero in the address of the file name word in the RENAME block.

A delete function on a channel that is open for output, to supersede a file, simply aborts the creation of the new file. This is equivalent to a CLOSE with CL.RST set.

Although only a privileged job can delete a UFD, an unprivileged job can delete an empty SFD. Note that you must set your path to a different area before you can delete the current SFD. If the directory is not empty or if a job is currently using the directory, the monitor returns the DIRECTORY NOT EMPTY error code.

A CLOSE is optional after a RENAME because a RENAME implicitly performs a CLOSE. A CLOSE should not be issued between a LOOKUP and a RENAME if the file is not in the default directory path, because the CLOSE erases all memory of the path. If RENAME is performed and the file is not in the default path, the monitor returns the FILE NOT FOUND error in the right half of addr+1.

Restriction

If your program attempts to change the extension of an SFD, a protection error results. An error also results if your program attempts to alter the name, extension, or PPN associated with a UFD or the PPN associated with an ersatz device name.

Normal Return

On a normal return, the monitor returns the same information on a RENAME as on a LOOKUP and ENTER. Refer to Section 11.13.

Error Return

The error return is taken under the following conditions:

- No file has been opened on the specified channel.
- The specified file cannot be found.
- The specified file is currently in the process of being written, superseded, or renamed.
- Your program does not have the appropriate privileges to RENAME the file.
- The new file name already exists (occurs when changing file names).

The monitor returns the error code for the RENAME monitor call in the right half of addr+1 of the 4-word argument block, or in the right half of addr+3 in the extended argument block. The error code overwrites the high-order three bits of the creation date and the entire access date.

This overwriting of data does not cause any problems for programs that recover from RENAME errors by aborting or by re-initializing the argument list. However, programs that attempt to recover from an error by fixing only the incorrect portion of the argument block and then reexecuting the monitor call should restore the right half of addr+1 or addr+3 before reexecuting the RENAME monitor call. Error codes are restricted to a maximum of 15 bits to allow programs to recover from an error in a file with a zero creation date. See Section 11.14 for a list of error codes.

RESCAN [TTCALL 10,]

Function

Resets the input buffer pointer to point to the beginning of the previous command. Note that if the RESCAN UUO is issued after the first terminal input or output instruction, the command is no longer in the buffer.

Calling Sequence

```
RESCAN  flag
  return 1
  return 2
```

where: flag controls the action of returning from the call. The flag is bit 35 of the word. If the flag is not set, the call always returns at return 1. If the flag is set, the call returns at return 2 when no command is in the input buffer, otherwise, the call takes return 1.

Example

```
RESCAN 1          ;Read TTY input
SKPINL            ;Is anything there?
  JRST  PROMPT    ;No, must be typeahead
                  ;Read command line
  . . .
```

Common Errors

Placing a comma after the flag.

RESDV. [CALLI 117]

Function

Resets a specified channel. RESDV. is similar to RESET, except that only one channel is reset and any outstanding data is discarded. If RESDV. is performed on a disk device, the file is discarded (refer to the CLOSE function CL.RST).

Calling Sequence

```
MOVEI   ac,channo
RESDV.  ac,
  error return
normal return
```

where: channo is the number of an initialized channel.

Normal Return

The channel is reset. Files that were being created on the channel are deleted; any older files with the same name remain. All I/O for the channel is stopped, and device allocations made on the channel by INIT, OPEN, or FILOP. are closed. If the device was not assigned by ASSIGN, ALLOCATE, REASSI, or MOUNT, it is returned to the monitor's pool of available devices. (See the Commands Manual for descriptions of these three commands.)

Error Return

If the ac contains -1, no device was associated with the channel.

Related Calls

CLOSE, RELEASE, RESET

Common Errors

Placing the channel number in the ac field.

RESET [CALLI Ø]

Function

Initializes a program. Resets the program's runtime environment to its initial state.

Calling Sequence

```
RESET
return
```

Return

The monitor initializes the program. This includes the following functions:

- Clears all device allocations except those for devices assigned by ASSIGN, REASSI, or MOUNT.
- Sets the job's first free location (right half of .JBFF) to its starting value (left half of .JBSA). This allows buffer space to be reclaimed when the program is restarted.
- Clears the left half of .JBFF (the job's first free location).
- Aborts processing of any files that have not been closed to release the associated I/O channels.
- Sets the user-mode write-protect bit for the high segment. This prevents inadvertent data storage in the high segment, and is done even if the segment is nonsharable.
- Unlocks your program, if it is locked in core.
- Releases any realtime devices.
- Resets any high-priority queue values to the value given in the last HPQ command.
- Resumes timesharing if it was stopped by a TRPSET monitor call.
- Resets any actions taken by APRENB, HIBER, or UTRP. monitor calls in your program.
- Clears all program counter flags for your program (except USRMODE and PUBLIC) that may be set.
- Clears any process identifications (PIDs) for your job, except job-wide PIDs.
- Clears the software interrupt facility for your job.
- Releases and dequeues any enqueue locks or requests for your job.

- Clears all of the data mode bits and the noecho bit for a terminal. However, if the RESET is executed for a not-logged-in job, whose program name is LOGIN, the noecho bit will not be cleared. This allows noecho to be set by the LOGIN command.

- Removes and undefines all SNOOP. breakpoints.

- Releases the performance meter.

- Clears any large disk buffers set by a UUO.

- Clears any address breaks set by a UUO.

RTTRP [CALLI 57]

Function

Connects a device to or releases it from the realtime interrupt facility. For a discussion of realtime devices, interrupt modes, and traps, refer to Chapter 9, Volume 1.

To use the RTTRP call, your job must have the JP.RTT privilege. To use an EPT-mode trap, your job must have the JP.TRP privilege. Your job must also have the JP.LCK privilege in order to lock itself in core on the correct CPU.

Calling Sequence

```
        MOVEI ac,addr
        RTTRP ac,
          error return
        normal return
        . . .
addr:   argument list
```

where: addr is the address of the argument list.

The contents of the argument list depend on the interrupt mode your program is setting up.

Normal Return

The device is connected to or released from the realtime interrupt facility.

Error Return

The monitor returns one or more of the following error flags in the ac. Before returning, the monitor scans the entire argument list to discover as many errors as possible.

Bit	Symbol	Error
23	RTNEC%	Nonexistent CPU.
24	RTJNP%	Not enough privileges.
25	RTNCØ%	Not runnable on CPUØ.
26	RTDIU%	Device in use by another job.
27	RTIAU%	Illegal accumulator used during RTTRP at interrupt.
28	RTJNL%	Job not locked (or not privileged).
29	RTSLE%	System limit for realtime devices exceeded.
3Ø	RTILF%	Illegal format for I/O instruction.
31	RTPWI%	Pointer word illegal.
32	RTEAB%	Error address out of bounds.
33	RTTAB%	Trap address bad.
34	RTPNB%	PI channel not currently available for BLKI/BLKO.
35	RTPNA%	PI channel not available.

Related Calls

HPQ, TRPSET, UJEN

RUN [CALLI 35]

Function

Transfers execution control from the current program to another program. The monitor replaces both the high and low segments of your address space with the segments of the called program. The function of the RUN UUO is described in more detail in Chapter 2 Volume 1.

Calling Sequence

```
        MOVSI   ac,start-addr-increment
        HRRI    ac,addr
        RUN     ac,
          error return
        . . .
addr:   SIXBIT/device/
        SIXBIT/filename/
        SIXBIT/extension/           ;or zero
        EXP     Ø
       {XWD     proj,prog}
       {XWD     Ø,addrl  }
       {XWD     Ø,core   }          ;or zero
       {XWD     -1,,addr2}          ;optional
```

where: start-addr-increment is an increment to the starting address of the called program. This increment is used to call indirect command files and should be Ø or 1. If any other value is used, the meddling bit is set for the job, unless the program is execute-only. For an execute-only program, this value can be only Ø or 1.

addr is the address of the 6- or 7-word argument block.

device is the logical or physical name of the device where the program being called resides.

filename is the name of the file being called (containing either or both high and low segments).

extension is the file extension of the low segment file (normally .EXE or blank).

proj,prog specifies the owner of the called program. If this word is zero, the monitor assumes that the called program is in your directory (that is, your PPN is assumed), or the PPN is determined from the directory name (for example, SYS=[1,4]). If this word is [XWD Ø,addrl], addrl points to a PATH. block containing the complete path for the file. Refer to the description of the PATH. monitor call for information pertaining to the format of the PATH. block.

core is the total amount of core to be reserved for the called program. This word must be included, but may be zero. For a program that contains both low and high segments, the amount of core required to load the high segment is subtracted from the core assignment first. The amount reserved for the low segment is the remainder.

addr2 contains a section number indicating where the image should be loaded. Using this argument results in an error if the .EXE file is not a single section program in Section Ø.

Normally your program sets up only the first two words of the argument block, leaving the rest of the words in the argument block zero.

You can omit values and accept defaults for all parameters in the argument list except device and filename. The defaults are:

extension	defaults to .EXE.
PPN	defaults to your PPN.
core	defaults to the amount required by the new program.

Normal Return

The new program is started at its new address plus start-addr-increment. The contents of ac may be changed on the return, and the new contents are unpredictable, because they vary from one monitor release to the next. The RUN call also performs an implicit RESET call.

Error Return

The error return is taken if any errors are detected; the monitor returns an error code in the ac. Your program can attempt to recover from an error and continue the program's execution. If you set the left half of the error return location to a HALT, the monitor will not return to the program but will print an error message. Your terminal will be at monitor level.

If you do not include a HALT in the left half of the error return location, your program can analyze the error code returned in the ac. If the error code indicates an error from which you can recover, your program can issue another RUN monitor call, possibly including a HALT instruction in the error return location.

If your program is using overlays, the monitor will not attempt to return to your program. Therefore, you should place the RUN monitor call in the low segment of your program, in case the error is discovered after the high segment has been released. If the call is issued from the low segment and an error occurs, the high segment of the program is cleared and must be re-initialized.

If the call is issued from the high segment and an error occurs, the monitor may halt the job and print the following message:

```
?Illegal address in UUO at user PC xxxxxx
```

For this reason, the RUN call should be given from the low segment.

Sée Section 11.14 for a list of error codes.

Related Calls

GETSEG, MERGE.

RUNTIM [CALLI 27]

Function

Returns the cumulative runtime (in milliseconds or ten microsecond units) for a specified job.

Calling Sequence

```
MOVEI    ac,jobno
   HRLI  ac,(RN.PCN)        ;optional for high precision
RUNTIM   ac,
return
```

where: **jobno** is the number of a logged-in job (use Ø for your own job). You may optionally set the sign bit 1BØ (RN.PCN), to return the runtime for the specified job in ten microsecond units (high-precision runtime).

Return

The **ac** contains the cumulative runtime (in milliseconds) for the specified job. If no such job exists, the ac contains Ø.

Examples

```
MOVEI   T1,Ø
RUNTIM  T1,
```

This code returns the cumulative runtime for the current job in T1.

SAVE. [CALLI 210]

Function

Saves the program in memory as an executable (.EXE) file on disk. This call is similar to the SAVE monitor command.

Calling Sequence

```
        MOVE    ac,[flag,,addr]
        SAVE.   ac,
          error return
        normal return
        . . .
addr:   SIXBIT/device/
        SIXBIT/filename/
        SIXBIT/extension/       ;or zero
        EXP     Ø
        {XWD    proj,prog}
        {XWD    Ø,,addr1 }
        {XWD    Ø,Ø      }      ;core argument
        {       -n,,addr2}
        {XWD    Ø,core   }
```

where: the **flag** is bit Ø of the **ac**. When set, this bit (SS%SSH) indicates that the program should be saved with a sharable high segment (similar to SSAVE monitor command).

addr is the address of the argument block, which is formatted like the argument blocks of GETSEG and RUN calls. **device** is the disk name or logical name of the disk device where the file should be written. **filename** and **extension** contain the file name and file extension of the new file. Note that **extension** is optional and defaults to .EXE. The extension is optional because the RUN monitor command will not accept file extensions other than .EXE.

proj,prog specifies the owner of the called program. If this word is zero, the monitor assumes that your directory contains the called program (that is, your PPN is assumed), or the PPN is determined from the device name (for example, SYS=[1,4]). If this word is [XWD Ø,,addr1], addr1 points to a PATH. block containing the complete path for the file. Refer to the PATH. monitor call for information pertaining to the format of the PATH. block.

Normal Return

The program in memory is written to disk in executable format. The contents of all accumulators may be changed; the new contents are not reliable and are subject to change from one monitor release to the next. The SAVE. call releases channel Ø implicitly.

Error Return

If an error occurs in the process of executing the SAVE. call, the non-skip return is taken and an error code is returned in the ac. Refer to Section 11.14 for the list of error codes.

SCHED. [CALLI 150]

Function

Reads or sets system scheduling parameters. JACCT or [1,2] privileges are required to issue the SCHED. monitor call. However, the read functions may be used by a user with SPY privileges, and the write functions are available to users with POKE privileges.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        SCHED.  ac,
          error return
        normal return
        . . .
addr:   XWD     fcncode,fcnarg
        . . .
        XWD     fcncode,fcnarg
```

where: len is the length of the argument list.

addr is the address of the argument list.

fcncode is one of the function codes described below.

fcnarg is the address of the argument list for the corresponding function code.

The function codes, their meanings, and their arguments are:

Code	Symbol	Function
Ø	.SCRSI	Reads the micro scheduling interval. The monitor returns the scheduling interval at fcnarg.
4ØØØØØ	.SCSSI	Sets the micro scheduling interval. The word at fcnarg should contain:

Word	Symbol	Contents
Ø	.SCBSI	Scheduling interval.

1	.SCRMI	Reads the minimum core usage function evaluation interval. The monitor returns the interval at fcnarg.
4ØØØØ1	.SCSMI	Sets the minimum core usage evaluation interval. The word at fcnarg should contain:

Word	Symbol	Contents
Ø	.SCBMI	Minimum core usage interval.

Code	Symbol	Function
2	.SCRCQ	Reads class quotas and flags. The monitor returns the quotas and flags at fcnarg and following in the form:

Word	Symbol	Contents
Ø	.SCBCT	Count of following words.
1	.SCBCQ	Class and quota:

Bits	Symbol	Meaning
Ø	SC.FCQ	Set if quota is fixed.
1-17	SC.CLN	Class number.
18-35	SC.CLQ	Class quota.

There is one word of the form of .SCBCQ for each word specified by the word count.

400002	.SCSCQ	Sets class quotas and flags. The data at fcnarg is the same as that returned by the .SCRCQ function.
3	.SCRTS	Reads the base quantum runtime. The monitor returns the time slices at fcnarg in the form:

Word	Symbol	Contents
Ø	.SCBCT	Count of following words.
1	.SCBP1	Base quantum runtime for PQ1.
2	.SCBP2	Base quantum runtime for PQ2.

400003	.SCSTS	Sets the base quantum runtime for one or both queues. The data at fcnarg is the same as that returned by the .SCRTS function.
4	.SCRUF	Reads the desired channel use fraction. This fraction is the swapping channel utilization percentage. The monitor returns the channel use fractions at fcnarg in the form:

Word	Symbol	Contents
Ø	.SCBCT	Count of following words.
1	.SCBUF	Channel number in left half; use fraction in right half.

There is one word of the form of .SCBUF for each word specified by the word count.

400004	.SCSUF	Sets the desired channel use fraction. The data at fcnarg is the same as that returned by the .SCRUF function.
5	.SCRJC	Reads the scheduler class for a job. The arguments at fcnarg are of the form:

Word	Symbol	Contents
Ø	.SCBCT	Count of following words.
1	.SCBJC	Job number in the left half; class in the right half.

Code	Symbol	Function
		There is one word of the form of .SCBJC for each word specified by the word count. The monitor returns the class for each given job.
400005	.SCSJC	Sets the scheduler class for a job. The arguments at fcnarg are the same as those for the .SCRJC function.
6	.SCRMC	Reads the minimum core usage per job. The monitor returns the minimum core usage at fcnarg (.SCBMC).
400006	.SCSMC	Sets the minimum core usage per job. The monitor reads the core usage from fcnarg (.SCBMC).
7	.SCRCU	Reads the class usage since startup. The monitor returns the class runtimes at fcnarg in the form:

Word	Symbol	Contents
0	.SCBCT	Count of following words.
1	.SCBCU	Runtime for class 0.

There is one word of the form of .SCBCU for each word specified by the word count.

10	.SCREF	Obsolete. The offset symbol .SCBEF is also obsolete.
400010	.SCSEF	Obsolete.
11	.SCRMM	Reads the minimum core usage multiplier. The monitor returns the multiplier at fcnarg (.SCBMM).
400011	.SCSMM	Sets the minimum core usage multiplier. The monitor reads the multiplier from fcnarg (.SCBMM).
12	.SCRDC	Reads the default class for new jobs. The monitor returns the default class at fcnarg (.SCBDC).
400012	.SCSDC	Sets the default class for new jobs. The monitor reads the default class from fcnarg (.SCBDC).
13	.SCRRC	Reads the minimum core usage requeue constant. The monitor returns the constant at fcnarg (.SCBRC).
400013	.SCSRC	Sets the minimum core usage requeue constant. The monitor reads the constant from fcnarg (.SCBRC).
14	.SCRPM	Reads the minimum core usage maximum. The monitor returns the maximum (in microseconds) at fcnarg (.SCBPM).

Code	Symbol	Function
400014	.SCSPM	Sets the minimum core usage maximum. The monitor reads the maximum (in microseconds) from fcnarg (.SCBPM).
15	.SCRML	Reads quantum multipliers for PQ1, PQ2, and scale factor. The monitor returns the values at fcnarg in the format:

Word	Symbol	Contents
0	.SCBCT	Count of the following words.
1	.SCBMP	For PQ1, the queue number in left half, quantum multiplier in right half.
2	.SCBMQ	For PQ2, the queue number in left half, quantum multiplier in right half.
3	.SCBMR	3 in left half, scale factor in right half (SC.BMR==3,,0).

400015	.SCSML	Sets quantum multipliers for PQ1, PQ2, and scale factor. The data at fcnarg must the the same as that returned by the .SCRML function.
16	.SCRMX	Reads the maximum quantum run for PQ1 and/or PQ2. The monitor returns the maximum quantum run at fcnarg in the format:

Word	Symbol	Contents
0	.SCBCT	Count of the following words.
1	.SCBMX	Queue number in left half, maximum time slice (in milliseconds) in right half. The returned block contains one word of the form of .SCBMX for each word specified in the word count.

400016	.SCSMX	Sets the maximum quantum run for PQ1 and/or PQ2. The data at fcnarg must be in the same format as that returned by the .SCRMX function.
17	.SCRSQ	Reads secondary class quotas. The monitor returns the quotas at fcnarg in the format:

Word	Symbol	Contents
0	.SCBCT	Count of following words.
1	.SCBSQ	Class in left half, quota in right half.

The data at fcnarg contains one word of the form of .SCBSQ for each word indicated by the word count.

400017	.SCSSQ	Sets secondary class quotas. The data at fcnarg must be in the same form as that returned by the .SCRSQ function.

Code	Symbol	Function
20	.SCRIQ	Reads the time percentage to scan queue just swapped in before subqueues. The monitor returns the time percentage at fcnarg (.SCBIQ).
400020	.SCSIQ	Sets the time percentage to scan queue just swapped in before subqueues. The monitor reads the time percentage from fcnarg (.SCBIQ).
21	.SCRSS	Reads swap scan time. The monitor returns the swap scan time at fcnarg (.SCBSS).
400021	.SCSSS	Sets swap scan time. The monitor reads the swap scan time from fcnarg (.SCBSS).
22	.SCRBB	Reads number for background batch subqueue. The monitor returns the number at fcnarg (.SCBBB).
400022	.SCSBB	Sets number for background batch subqueue. The monitor reads the number from fcnarg (.SCBBB).
23	.SCRBS	Reads background batch swap time interval. The monitor returns the interval at fcnarg (.SCBBS).
400023	.SCSBS	Sets background batch swap time interval. The monitor reads the interval from fcnarg (.SCBBS).
24	.SCRSF	Reads scheduler fairness factor. The monitor returns the fairness factor at fcnarg (.SCBSF).
400024	.SCSSF	Sets scheduler fairness factor. The monitor reads the fairness factor from fcnarg (.SCBSF).
25	.SCRSW	Reads swapper fairness factor. The monitor returns the fairness factor at fcnarg (.SCBSW).
400025	.SCSSW	Sets swapper fairness factor. The monitor reads the fairness factor from fcnarg (.SCBSW).
26	.SCRIO	Reads in-core fairness. The monitor returns the fairness at fcnarg (.SCBIO).
400026	.SCSIO	Sets in-core fairness. The monitor reads the fairness from fcnarg (.SCBIO).
27	.SCRSC	Reads SCDCOR. The monitor returns the value of SCDCOR at fcnarg (.SCBSC).
400027	.SCSSC	Sets SCDCOR. The monitor reads the value for SCDCOR from fcnarg (.SCBSC).
30	.SCRSO	Reads the CPU scan order. The monitor returns the scan order for each CPU at fcnarg+1. The argument block at fcnarg is the same as the information you give to set the scan order in function 400030 (.SCSSO).

Code	Symbol	Function
400030	.SCSSO	Sets the CPU scan order. The argument list at fcnarg should appear as:

Word	Symbol	Contents
0	.SCBCT	Word count.
1	.SCBSO	Scan order for CPU0.
2	.SCBSO	Scan order for CPU1.
. . .		

Where .SCBSO is 0 if the scan order is [HPQ,PQ1,PQ2] or 1 if the order is [HPQ,PQ2,PQ1].

Code	Symbol	Function
31	.SCRRT	Reads dormant segment retention time (in jiffies). The monitor returns retention time at fcnarg (.SCBRT).
400031	.SCSST	Sets dormant segment retention time. The monitor reads retention time from fcnarg (.SCBRT).
32	.SCRFG	Reads the free core goal. The monitor returns the free core goal at fcnarg in the following format:

Word	Symbol	Meaning
0	.SCBFG	Minimum free core size (goal).
1	.SCBFL	Maximum free core size.

Both .SCBFG and .SCBFL are percentages of user core as determined when the system was booted.

Code	Symbol	Function
400032	.SCSFG	Sets the free core goal. The monitor reads the goal from fcnarg in the format given in function .SCRFG.

Normal Return

The function has been performed.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	SCHAC%	Address check.
2	SCHUF%	Unknown function code.
3	SCHUJ%	Unknown job.
4	SCHNP%	Not enough privileges.
5	SCHUC%	Unknown class.
6	SCHUQ%	Unknown queue.
7	SCHNC%	Nonexistent channel.
10	SCHEB%	Bad exponential factor.
11	SCHMI%	Cannot set protection if MCUINT is nonzero.
12		Reserved for use by DIGITAL.
13	SCHNH%	Not 100%.
14	SCHFN%	Fairness not positive.
15	SCHIC%	Illegal CPU number specified in function .SCSSO.
16	SCHUO%	Unknown scan order specified in function .SCSSO.

SCS. [CALLI 213]

Function

Provides the diagnostic interface to the Systems Communications Service layer of the System Communications Architecture, allowing information to be exchanged between jobs on different systems connected over a CI20.

This monitor call is used in DIGITAL-supplied hardware diagnostic programs and is not intended to be used in customer programs. The calling sequences and arguments of SCS. are subject to change without notice. The program must be run under [1,2] or have JACCT privileges to use the SCS. UUO.

Calling Sequence

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

addr:   len,,function
```

where: addr is the starting address of the argument block, len specifies the total length of the argument block, and function is one of the function codes described below. The function word is formatted as follows:

Word Symbol	Bits	Field Symbol	Contents
.SSFNC	1-8	SS.CPU	CPU number.
	9-17	SS.LEN	Length of argument block, including this word.
	18-35	SS.FNC	One of the function codes listed below.

Function codes are:

Code	Symbol	Meaning
0	.SSCON	Requests a connection.
1	.SSLIS	Listens for a connection.
2	.SSREJ	Rejects a connection request.
3	.SSDIS	Disconnects and closes a connection.
4	.SSSDG	Sends a datagram.
5	.SSQRD	Queues buffer(s) to receive datagram.
6	.SSSMG	Sends a message.
7	.SSQRM	Queues buffer(s) to receive message.
10	.SSCSP	Returns information about a status of a connection.
11	.SSRCD	Returns configuration data for a remote system.
12	.SSSTS	Returns information about status of a connection.
13	.SSRMG	Receives a message.
14	.SSMAP	Maps a buffer for DMA transfer.
15	.SSUMP	Unmaps a buffer for DMA transfer.
16	.SSSND	Sends data to remove host.
17	.SSREQ	Requests delivery of data.

Code	Symbol	Meaning
20-21		Reserved.
22	.SSRDG	Receives a datagram.
23	.SSACC	Accepts a connection request.
24	.SSGDE	Returns entry from data request complete queue.
25	.SSEVT	Returns entry from event queue.
26	.SSCRD	Cancels datagram receive.
27	.SSCRM	Cancels message receive.
30	.SSGLN	Gets local node number.
31-34		Reserved.
35	.SSRBS	Returns minimum buffer sizes.
36	.SSRPS	Returns path status.

Normal Return

The function is performed successfully and the program continues at the skip return.

Error Return

The function is not performed, and the error code is returned in the ac. The error codes and their meanings are:

Code	Symbol	Error
0	SSNPV%	Insufficient privileges.
1	SSIFC%	Illegal function code.
2	SSARG%	Bad argument list length.
3	SSACR%	Address check reading argument block.
4	SSACS%	Address check storing data.
5	SSCPN%	CPU number is out of range.
6	SSNPC%	No CI port on specified CPU.
7	SSNNK%	CI node number on specified CPU is not known.
10	SSINN%	Invalid CI node number.
11	SSNFC%	No free core.
12	SSVNO%	Virtual circuit is not open.
13	SSICI%	Invalid connect identification.
14	SSRQE%	Receive queue is empty.
15	SSNBQ%	No buffer queued for packet reception.
16	SSRCF%	Reject connection failed.
17	SSDCF%	Disconnect connection failed.
20	SSNFB%	No free buffers to send packet.
21	SSQBF%	Queue buffers failed.
22	SSCBF%	Cancel buffers failed.
23	SSPSF%	Packet send failed.
24	SSDQE%	Data entry queue empty.
25	SSEQE%	Event queue empty.
26	SSCRB%	Can't remove buffer from database.
27	SSCUB%	Can't unmap buffer.
30	SSNSB%	No such buffer name.
31	SSTMS%	Too many buffer segment descriptions.
32	SSIDM%	Illegal data mode.
33	SSSCP%	Segment crosses page boundary.
34	SSSTL%	Segment is greater than 1 page.

SEBLK. [CALLI 214]

Function

SEBLK. is a privileged monitor call used only by DAEMON. It returns system error block data.

Calling Sequence

```
        MOVE    ac,[arglen,,arglst]
        SEBLK.  ac
          error return
        normal return
```

where: arglen is length of the argument list stored at arglst.

Normal Return

The monitor returns information about system errors in the block starting at arglst. The number of words stored in the monitor's error block is returned in the ac. This tells you whether your block was long enough to hold the information; if the block was not long enough, the monitor truncated the information.

Error Return

One of the following codes is returned in the ac:

Code	Symbol	Error
Ø	SBNPV%	Job not privileged.
1	SBNEQ%	No error blocks on queue.

SENSE. [CALLI 133]

Function

Returns the I/O status bits for a device. I/O status bits can be cleared individually using the CLRST. monitor call.

Calling Sequence

```
        MOVE        ac,[XWD len,addr]
        SENSE.    ac,
          error return
        normal return
        . . .
addr:  {SIXBIT/device/  }
       {EXP       channo}
       {EXP       udx   }
        XWD       length,status
        . . .
status:      SIXBIT/name/
status+1:    XWD Ø,GETSTS-bits
status+2:    DEVSTS-word
```

where: len is the length of the argument list, which must be 2.

addr is the address of the argument list.

device is the SIXBIT physical or logical name of an initialized device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

Device, channo, and UDX are alternate ways of specifying the device for which you desire the status bits.

length specifies the number of words in the status block. This value should equal the number of devices multiplied by 3.

status is the address of the status block.

The status block is returned in the form:

Offset	Symbol	Contents
Ø	.SNSDV	The SIXBIT name of the device.
1	.SNSST	The status bits for the device (GETSTS).
2	.SNSDS	The device status (DEVSTS) bits for the device. DEVSTS bits are from the device DDB, and are different for each device.

Normal Return

The name and status bits for the device are returned at status.

SENSE. [CALLI 133]

Error Return

If the SENSE. monitor call is not implemented on your system, the ac is unchanged; otherwise, the following error code is returned in the ac:

Code	Symbol	Error
1	SNSBD%	Illegal device specified.

Related Calls

CLRST.

SETDDT [CALLI 2]

Function

Sets the value of .JBDDT in the Job Data Area. Note that .JBDDT is protected from a direct MOVEM because the monitor has its own copy of .JBDDT and restores its value at every context switch.

Calling Sequence

```
MOVE    ac,[last-addr,,start-addr]
SETDDT  ac,
return
```

where: **last-addr** is the last address for DDT and **start-addr** is the new start address.

Return

The start address and last address for DDT are set.

SETLCH [TTCALL 7,]

Function

Sets the line characteristics for your job's controlling terminal. The line characteristics can be read using the GETLCH call.

Calling Sequence

```
SETLCH  [XWD flags,lineno]
return
```

where: flags are described below.

lineno is the line number for a terminal that can be a terminal line number (such as 37 for TTY37) or a UDX (such as UXTRM+37 for TTY37).

If you give a negative number for lineno, the current user terminal is assumed. Flags can be changed only for the job's controlling terminal.

The flags are:

Flag	Symbol	Characteristic
13	GL.LCM	Terminal in lowercase mode.
14	GL.TAB	Terminal has tab capability.
15	GL.LCP	Local copy only (no echo).
16	GL.PTM	The CTRL/Q papertape switch is on.
17	GL.NEC	No echo from program.

Return

The line characteristics are set as requested. The argument block is not changed.

Related Calls

GETLCH, TRMOP.

Common Errors

Using an ac in the calling sequence.

SETNAM [CALLI 43]

Function

Changes the name of the current program in the monitor's job table. This name is used by some monitor commands, such as USESTAT (CTRL/T) and SYSTAT.

The SETNAM monitor call also clears the SYS program bit (which is used by GALAXY), clears the execute-only and JACCT bits, and causes a version typeout if a version watch has been set with the SET WATCH VERSION monitor command or with the .STWTC, ST.WVR function of SETUUO.

Calling Sequence

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

where: name is the new program name for the job.

Return

The new program name is entered in the monitor's job table.

Examples

```
MOVE    T1,[SIXBIT/NEWNAM/]
SETNAM  T1,
```

This code changes the program name for the job to NEWNAM.

SETSTS [OPCODE Ø6Ø]

Function

Sets bits in the file status word for a device. Use FILOP. to perform a SETSTS for an extended I/O channel.

Calling Sequence

```
SETSTS channo,bits
return
```

where: channo is the number of an initialized channel.

bits are I/O status bits. For a complete list of I/O status bits, see the appropriate device chapter in Volume 1.

Return

The I/O status bits are set.

Examples

```
GETSTS     CHN,T1        ;get status in T1
TRZ        T1,IO.ERR     ;keep mode and device-
                         ; dependant bits
SETSTS     CHN,(T1)      ;clear errors
```

Related Calls

CLRST., GETSTS

Common Errors

1. If the SETSTS monitor call is done for a channel that has not been initialized, the monitor stops the job and prints:

   ?IO to unassigned channel at user PC nnnnnn

2. If the data mode is illegal for the device, the monitor prints:

   ?Illegal data mode for device xxxnnn; UUO at user PC nnnnnn

SETUUO [CALLI 75]

Function

Sets system or job parameters. To set system parameters, your job must have the JACCT bit set, or must be logged in under [1,2], and may not be a batch job.

Calling Sequence

```
MOVE    ac,[XWD fcncode,argument]
SETUUO  ac,
  error return
normal return
```

where: **fcncode** is one of the function codes described below.

argument is an argument for the given function code.

The function codes and their meanings are:

Code	Symbol	Function
Ø	.STCMX	Sets the maximum core size that a user job may use (the sum of the high and low segments, CORMAX). The minimum value is set by MONGEN. The maximum value is the size of user core in words.
1	.STCMN	Sets the guaranteed amount of contiguous core that a single unlocked job can use (CORMIN). The valid values are in the range Ø to CORMAX. This argument is referred to as CORMIN.
2	.STDAY	Obsolete.
3	.STSCH	Sets parameters in the %CNSTS word in GETTAB Table 11. The argument gives the flags. The flags and their meanings are:

Bit	Symbol	Meaning
26	ST%NDL	No automatic down-line load of DC72, DC71, or DAS8Ø-series remote station.
27	ST%NOP	No operator coverage.
28	ST%NSP	Allow device unspooling.
29	ST%ASS	Allow device assignment and initialization.
32	ST%NRT	No remote TTYs.
33	ST%BON	LOGINs for batch jobs only.
34	ST%NRL	No remote LOGINs.
35	ST%NLG	No LOGINs except CTY.

Code	Symbol	Function
4	.STCDR	Specifies the input name for the card reader job, which is stored in the left half of location .GTSPL. The argument is given in single quotes that contain three SIXBIT characters forming the job name. For example, the MOVE statement in the calling sequence might be: MOVE AC1,[XWD .STCDR,'XYZ'] to specify the input name XYZ.
5	.STSPL	Sets or clears the spooling state for the job's devices. You specify the flag bits in the ac and they are set in bits 31 to 35 of .GTSPL. The flags and their meanings are:

Bit	Symbol	Meaning
31	JS.PCR	Spool card reader.
32	JS.PCP	Spool card punch.
33	JS.PPT	Spool paper tape punch.
34	JS.PPL	Spool plotter.
35	JS.PLP	Spool line printer.

6	.STWTC	Sets flags for SET WATCH routine (refer to monitor command SET WATCH). The argument gives one or more of the flags, which are:

Bit	Symbol	Meaning
9	ST.WCN	/MESSAGE:CONTINUATION.
10	ST.WFL	/MESSAGE:FIRST.
11	ST.WPR	/MESSAGE:PREFIX.
11	ST.WMS	/MESSAGE:ALL.
19	ST.WCX	Watch contexts.
19	ST.WDY	Watch daytime at start.
20	ST.WRN	Watch runtime.
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 statistics for magtapes.
26	ST.WFI	Watch file accessed.
19-26	ST.WAL	Watch all.

7	.STDAT	Sets the system date as the number of days since January 1, 1964, in 15-bit form. The argument gives the number of days as: ((year-1964)12+(month-1))31+(day-1)
10	.STOPR	Sets the SIXBIT name of the terminal to be used as the operator terminal. The argument gives the address of the word containing the name.
11	.STKSY	Sets the decimal number of minutes until timesharing ends; this value is stored in SYSKTM. If SYSKTM is 0, timesharing is continued indefinitely.

Code	Symbol	Function
12	.STCLM	Sets the maximum number of words of memory that the job can use. This value is converted from words to pages (at 512 words/page) and stored in location JB.CLR in table .GTLIM. This value may be set to zero, indicating that there is no maximum limit.
13	.STTLM	Sets the maximum number of seconds the job can run. The argument is the number of seconds permitted. This function cannot be used by batch jobs that already have a time limit. However, this function is allowed for non-batch jobs, batch jobs with no time limit, and privileged batch jobs with or without a time limit.
14	.STCPU	Specifies the CPU on which the job is to run. The argument gives one of the following flags:

Bit	Symbol	Meaning
30	SP.CR5	Run on CPU5.
31	SP.CR4	Run on CPU4.
32	SP.CR3	Run on CPU3.
33	SP.CR2	Run on CPU2.
34	SP.CR1	Run on CPU1.
35	SP.CR0	Run on CPU0.

.STCPU is a privileged function, requiring JP.CCC privileges. The error return is taken if you attempt to change the CPU specification on a single-CPU system.

15 .STCRN Sets runnability for CPUs. The argument gives one or more of the following flags:

Bit	Symbol	Meaning
30	SP.CR5	CPU5 is runnable.
31	SP.CR4	CPU4 is runnable.
32	SP.CR3	CPU3 is runnable.
33	SP.CR2	CPU2 is runnable.
34	SP.CR1	CPU1 is runnable.
35	SP.CR0	CPU0 is runnable.

The error return is taken if you attempt to issue this function on a single-CPU system.

16 .STLMX Sets the maximum number of jobs that can be logged in at any one time; this value is stored in location LOGMAX. The argument gives the maximum number of jobs; this number must be at least 1, but no more than the system maximum, which is defined by the symbol JOBN. JOBN is the system limit defined when the monitor is generated by MONGEN.

If you give a number smaller than the number of jobs currently logged in, no new jobs can log in until the number of jobs falls below LOGMAX.

Code	Symbol	Function
		You can obtain the number of jobs currently logged in from the location %CNLNM in GETTAB table .GTCNF.
17	.STBMX	Sets the maximum number of batch jobs that can be logged in at any one time; this value is stored in location BATMAX. The argument gives the maximum number of batch jobs; this number must be less than the system maximum, which is defined by the symbol JOBN. You can obtain the number of batch jobs currently logged in from the location %CNBNM in GETTAB table .GTCNF.
2Ø	.STBMN	Sets the number of jobs reserved for batch processing (BATMIN). The argument gives the minimum number of jobs reserved. The value must be in the range 1 to the value of BATMAX-1.
21	.STDFL	Sets the action to occur if the user disk space is filled for the job. The argument is one of the following codes: Ø .DFPSE Pause when disk filled for job. 1 .DFERR Error when disk filled for job or the user's quota has been exceeded. Any other value for argument returns the current setting for .STDFL in the ac; the initial default setting is .DFERR.
22	.STMVM	Sets the system-wide virtual memory limit (GVPL). The value returned in ac depends on the given argument: o If the given argument is less than the current virtual memory page count, the value returned is the total amount of virtual memory in use by all virtual memory users. o If the given argument is greater than the current available swapping space, the value returned is the total amount of available swapping space. o If the given argument is greater than the total amount of virtual memory currently in use, the value returned is the given argument.
23	.STMVR	Obsolete. This historical SETUUO function always takes the error return and clears the ac.

Code	Symbol	Function
24	.STUVM	Sets the maximum virtual memory page limit and the maximum physical memory page limit. The argument gives the address of the word whose format is: LH = maximum virtual page limit (MVPL) RH = maximum physical page limit (MPPL) If the left half of the word (MVPL) is Ø, the user cannot use the virtual memory option. When MVPL is set to Ø, MPPL should also be set to Ø. If the right half (MPPL) is Ø, the user can use all of the system's physical memory.
25	.STCVM	Sets the current memory maximum. The argument is the address of a word whose format is: LH = current virtual page limit (CVPL) RH = current physical guideline or limit (CPPL) The left half (CVPL) of the word at the indicated address contains the current virtual page limit. If bit 18 (ST.VSG) is Ø, the right half (CPPL) contains the current physical page guideline; if bit 18 is 1, bits 19 to 35 contain the current physical page limit. A guideline is an approximate physical page limit. The guideline algorithm allows you to set a memory limit that will not be strictly enforced. The page fault handler will attempt to meet the page limit within a window of approximation, allowing slight over-allocation to accommodate the program. If you set the ST.VSG bit, the allocation is taken as a limit, and that limit is strictly enforced.
26	.STTVM	Sets the time interval between virtual time traps in milliseconds. A virtual time trap causes a Code 4 page fault to the page fault handler each time the time interval has elapsed. The argument gives the number of milliseconds between traps.
27	.STABK	Sets the address break condition. On a normal return, the new address break condition and the break address have been set. The address conditions are specified in the word pointed to by the argument. These conditions are: Bits — Symbol — Contents Ø — ST.AEX — Set to break on EXECUTE. 1 — ST.ARD — Set to break on READ. 2 — ST.AWR — Set to break on WRITE.

Code	Symbol	Function

Bits	Symbol	Contents
3	ST.AUU	Set to break on monitor reference.
4-8		Specify the section number for the break address.
9-17	ST.ACT	The number of times the break address is to be referenced before an interrupt occurs.
18-35	ST.ADR	Sets the break address.

To clear the address break, clear bits Ø through 3.

If you have enabled for address break interrupts, the PSI system will interrupt on an address break. If the PSI system is not enabled, the monitor will stop your job and display the following message on your terminal:

```
%Address break at user PC xxxxxx
```

30 .STPGM

Sets the name of a program that will run when the current program session finishes executing. You must run the program executing this SETUUO from SYS:, under [1,2], or with JACCT privilege set. The argument block is:

```
EXP     flag        ;Bits 2-35 reserved
SIXBIT /progra/     ;progra is program name
```

The monitor does an implied RUN UUO on SYS:progra.EXE when the current program session ends. Program session termination occurs under one of the following conditions:

- When Bit Ø of the flag is set, the session terminates whenever the job would otherwise enter monitor mode (for instance, EXIT UUO, ^C, illegal memory reference, or swap read error). If the job becomes detached, issuing the unprivileged ATTACH command (or the ATTACH UUO) does not attach you to the job in monitor mode. You will attach in user mode instead. The DETACH function of the ATTACH UUO allows you to detach, leaving your terminal in monitor mode.

- When Bit Ø of the flag is clear, the session terminates whenever you execute a command that destroys the core image. The RUN UUO executes SYS:progra.EXE instead of whatever command you issued. You can still enter monitor mode when your program terminates, or when you issue an ATTACH command. You may then execute any command that does not change the core image, as well as the KJOB command. Commands that automatically save the current context and push to a new one do not change the core image.

Code	Symbol	Function
		The status of .STPGM remains in effect until it is explicitly cleared by a privileged program, or the job logs out. Whenever .STPGM is in effect, the program may execute RUN UUOs on any file. Control may even transfer to a program that is not privileged to execute this SETUUO. If .STPGM specifies an inaccessible file, SYS:LOGOUT will be run when the program session terminates.
31	.STDFR	Sets deferred spooling. If argument is non-zero, spooled output will not be queued until the job logs out. If argument is zero, spooled output will be queued as each file is closed.
32	.STHST	Sets the host system. This function logically attaches the controlling terminal to the specified host system in an ANF-1Ø network. The job on the previous system becomes detached. The calling sequence is: MOVE ac,[.STHST,,addr] SETUUO ac, error return normal return . . . addr: node number or SIXBIT node name
33	.STDEF	Sets default values for job-wide parameters. The calling sequence for the .STDEF function is: MOVE ac,[XWD .STDEF,addr] SETUUO ac, error return normal return . . . addr: XWD arglen,subfcncode argument where: arglen gives the number of arguments to follow. subfcncode is one of the following subfunction codes:

Code	Symbol	Subfunction
Ø	.STDPC	Set default protection code. addr+1 contains the new default protection code.
1	.STDNB	Set default number of disk buffers. addr+1 contains the new default number of disk buffers.
2	.STDAD	Controls whether LOGIN will ask you about attaching to this job should you previously have detached from it. JD.DAD in .GTDFL contains the value of the flag.

Code	Symbol	Function

Code	Symbol	Subfunction		
		Argument	JD.DFL	Action
		zero	clear	LOGIN will ask about this job.
		non-zero	set	LOGIN does not ask if you want to attach to this job.

Code	Symbol	Subfunction
3	.STDSB	Sets the default size of a disk buffer. Refer to SET BIGBUF monitor command. addr+1 contains the new disk buffer size (value is number of blocks per buffer).

34 .STPRV Sets the privilege and capability words. The calling sequence for the .STPRV function is:

```
        MOVE    ac,[XWD .STPRV,arglst]
        SETUUO  ac,
        . . .
arglst: XWD     Ø,subfcncode
        argument
```

subfcncode is one of the following subfunction codes:

Code	Symbol	Subfunction
Ø	.STCPW	Sets entire privilege word.
1	.STCPS	Sets specified bits of privilege word.
2	.STCPC	Clears specified bits of privilege word.
3	.STCCW	Sets entire capability word.
4	.STCCS	Sets specified bits of capability word.
5	.STCCC	Clears specified bits of capability word.

NOTE

You can always clear bits in the privilege word. However, you can set only those bits in the privilege word that are set in the capability word, unless you are a privileged job.

35 .STBSN Sets batch stream number (settable only once per instance).

36 .STWTO Sets write-to-operator values. Refer to GETTAB table .GTOBI.

Code	Symbol	Function
37	.STCDN	Sets CPU up/down status. To control the CPU up/down status, set the appropriate bits in the argument, from the following list: Bit — Function 18 — Remove CPU from system. 19 — Suspend the CPU. 20 — Add the CPU to the system. The CPU number must be stored in bits 33-35 of the argument.
40	.STCSB	Sets or clears cache bits. The cache can be enabled or disabled for the monitor's low segment by setting the argument to 1 to enable cache, or 0 to disable cache.
41	.STFPS	Obsolete.
42	.STOPP	Allows various levels of operators to run OPR without [1,2] privileges and without having full file access. This value may be read from GETTAB table .GTOBI, and will be be used by LOGIN.
43	.STQST	Sets queue structure. This sets the file structure on which GALAXY queues will be stored.
44	.STCSZ	Sets the size of the software disk cache in blocks. This value can be set with MONGEN symbol M.CBMX. The default value of M.CBMX is the number of jobs on the system. The .STCSZ function is illegal if M.CBMX=0. The argument for this function specifies the number of disk blocks for the cache.
45	.STEBP	Sets the EDDT breakpoint facility. The argument to this function is either 0 (to disable the facility) or 1 (to enable the facility).
46	.STBPT	Sets the DDT breakpoint facility. The argument to this function is either 0 (to disable the facility) or 1 (to enable the facility).
47	.STTMS	Sets the system time of day. Specify the time as the number of seconds past midnight.
50	.STCXP	Sets the maximum core size that a user job may use (CORMAX). The maximum value is the size of the user core in pages.
51	.STCNP	Sets the guaranteed amount of contiguous core that a single unlocked job can use (CORMIN), with the argument in pages.

Normal Return

The function is performed and the ac is unchanged.

SETUUO [CALLI 75]

Error Return

If the ac is cleared, you do not have sufficient privileges or you gave an illegal job number, CPU number, or argument.

If the ac is not changed on an error return, the function you requested is not implemented in the monitor.

Otherwise, one of the following error codes can be returned in the ac:

Code	Symbol	Error
Ø	STNAR%	Specified CPU(s) not running (functions .STCPU and .STCRN).
1	STDHP%	Insufficient privileges to perform specified function.
2	STISN%	Illegal structure name (function .STQST).
3	STITM%	Illegal time (resulting time would be greater than 23:59:59).

Related Calls

JBSET.

SETUWP [CALLI 36]

Function

Sets or clears user-mode write protection for the job's high segment. You must use the SETUWP call to clear write protection before your program can modify its high segment.

Because the previous setting of this bit is returned in **ac**, you can write subroutines that preserve the previous setting and restore them before returning.

Calling Sequence

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

where: **fcncode** is one of the following function codes:

Code	Function
Ø	Write-enables the high segment.
1	Write-protects the high segment.

Normal Return

The user-mode write protection bit is set as specified, and the previous setting is returned in the ac.

Error Return

The error return is taken under the following conditions:

- If the high segment is a SPY segment.
- If the high segment has been meddled.
- If the user does not have the access privileges required to access the specified high segment.

SKPINC [TTCALL 13,]

Function

Skips the next program instruction if at least one character can be input from the job's controlling terminal. The SKPINC call does not input a character. SKPINC clears the CTRL/O output state and sets the terminal to "character mode", preventing the monitor from processing control characters, such as DELETE and CTRL/U, as input line editing commands.

This call is useful in a compute-bound program that should check occasionally for user input.

Calling Sequence

```
SKPINC
  return 1
  return 2
```

where: the call returns to **return** 1 if there is no user input, or to **return** 2 is there is user input.

Related Calls

SKPINL

SKPINL [TTCALL 14,]

Function

Skips the next instruction if at least one line can be input from the job's controlling terminal. SKPINL sets the terminal to "line mode" and clears the CTRL/O output state.

Calling Sequence

```
SKPINL
  return 1
  return 2
```

where: the call returns to **return 1** if a complete line has not been typed, or to **return 2** if a complete line has been typed.

Related Calls

SKPINC

SLEEP [CALLI 31]

Function

Causes your program to become dormant for a specified number of real-time seconds.

Calling Sequence

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

where: seconds gives the number of seconds that the job is to sleep. If you give seconds as Ø, the program will sleep for one clock tick. The maximum sleep time is 68 seconds (or 82 seconds for systems using 5Ø Hz frequency). If you require a longer sleep period, use the HIBER monitor call.

Return

Your job becomes dormant and the monitor sets the JBTST2 bit, JS.SLP. The monitor will clear this bit when the specified time has elapsed and your job becomes runnable again. All potential job-wakers should check this bit and wake a job only if the bit is cleared.

Examples

```
MOVEI T1,1
SLEEP T1,
```

This code puts your job to sleep for 1 second.

Related Calls

HIBER

SNOOP. [CALLI 176]

Function

Allows privileged programs to insert breakpoints in the monitor that trap to a user program. The user program must be locked in core when the trap occurs (refer to LOCK monitor call). This feature is used for fault insertion, performance analysis, and trace functions. Only one job can use SNOOP. at any time.

CAUTION

Improper use of the SNOOP. call can cause the system to fail in a number of ways. User programs may require special code for multiprocessor systems because the monitor may be executing the same code simultaneously on several systems and at different interrupt levels.

Refer to Chapter 1Ø for more information about the SNOOP. monitor call. Do not attempt to use this call until you are familiar with its operation.

Calling Sequence

```
        MOVE    ac,[XWD fcncode,addr]
        SNOOP.  ac,
          error return
        normal return
        . . .
addr:   argument list
```

where: fcncode is one of the function codes described below.

addr is the address of the argument list. The words at addr depend on the given function.

The function codes and their meanings are:

Code	Symbol	Function
Ø	.SODBP	Defines breakpoints. This function is illegal if breakpoints have been inserted.

The argument list for the .SODPB function is:

```
arglst: EXP     arglength
        EXP     symbol checksum
        EXP     address
        instruction
        . . .
        EXP     address
        instruction
```

Code	Symbol	Function

where: arglength is the length of the argument list. This must be 2 + the number of address-instruction pairs in the argument list times 2 .

symbol checksum is the checksum from the current monitor's symbol table.

The checksum is required to ensure that the user is setting breakpoints in the intended monitor. Your program can obtain the version of the monitor that is read in by BOOTS from GETTAB table .GTCNF, where the relevant items are:

Offset	Item	Contents
137	%CNBCP	Bootstrap CPU number.
140	%CNBCL	Bootstrap line number.
141	%CNNCR	Number of CPUs allowed to run.
142	%CNMBS	Bootstrap file structure.
143	%CNMBF	Bootstrap file name.
144	%CNMBX	Bootstrap file extension.
145	%CNMBD	Bootstrap file directory.
155	%CNSF1	Bootstrap first SFD.
156	%CNSF2	Bootstrap second SFD.
157	%CNSF3	Bootstrap third SFD.
160	%CNSF4	Bootstrap fourth SFD.
161	%CNSF5	Bootstrap fifth SFD.

The checksum is followed by a series of word pairs, each of which defines a breakpoint by specifying, in the first word of each pair, the monitor virtual address where the new instruction is to be placed, and, in the second word of the pair, the new instruction to be inserted. Specifically, the argument list for SNOOP. function 0 is:

Word	Symbol	Contents
0	.SOLEN	The length of the argument list, (the number of breakpoints being defined times two, plus two).
1	.SOMSC	The checksum of the monitor symbol table.
2	.SOMVA	Monitor virtual address where new instruction is to be inserted.
3	.SOBPI	New instruction.

.SOMVA and .SOBPI are repeated for each replaced instruction.

Code	Symbol	Function
1	.SOIBP	Inserts all breakpoints that have been defined using function Ø. Your program must be locked in contiguous executive virtual memory to use this function (see the LOCK monitor call).
2	.SORBP	Removes inserted breakpoints from monitor code.
3	.SOUBP	Undefines breakpoints that have been removed using function 2.
4	.SONUL	Null function. This function allows you to execute code inserted at label BP$ØØØ, after ensuring that your job owns the SNOOP resource. This may be used by a program that must execute code in monitor context and wants to ensure that only this program can invoke the inserted code. If you do not own the SNOOP resource, the instruction at BP$ØØØ is not executed, and the error code SOSAS% is returned.

Normal Return

The indicated function has been performed.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	SOIAL%	Illegal argument list.
2	SONPV%	Not enough privileges.
3	SOSAS%	Another program already snooping.
4	SOMBX%	Maximum number of breakpoints exceeded.
5	SOIBI%	Breakpoints already inserted.
6	SONFS%	No monitor free core available.
7	SOADC%	Address check.
1Ø	SOINL%	Program not locked in contiguous executive virtual memory.
11	SOWMS%	Monitor symbol table checksum does not match.

Examples

The monitor computes the symbol table checksum in the following manner:

```
        MOVE    T1,.JBSYM
        SETZM   CHKSUM
LOOP:   MOVE    T2,(T1)
        EXCH    T2,CHKSUM
        ROT     T2,1
        ADD     T2,CHKSUM
        EXCH    T2,CHKSUM
        AOBJN   T1,LOOP

                .
                .
                .
        MOVE    T1,[XWD .SODBP,PUTEM]
        SNOOP.  T1,
          JRST  NOGOOD
        MOVE    T1,[XWD .SOIBP,Ø]
        SNOOP.  T1,
          JRST  NOBTTR
                .
                .
                .
PUTEM:  EXP     6
        EXP     MONITOR-CHECKSUM
        EXP     12345
        JRST    HOOK1
        EXP     12355
        JRST    HOOK2
```

At this point the breakpoints have been inserted. To remove them:

```
                .
                .
                .
        MOVE    T1,[XWD .SORBP,Ø]
        SNOOP.  T1,
          JRST  BUMMER
        MOVE    T1,[XWD .SOUBP,Ø]
        SNOOP.  T1,
          JRST  LOSTIT
                .
                .
                .
```

SPPRM. [CALLI 172]

Function

Sets parameters for spooled files.

Calling Sequence

```
        MOVE    ac,[length,addr]
        SPPRM.  ac,
          error return
        normal return
        . . .
addr:   function code
        device-id
        parameters
        .
        .
        .
```

where: **length** is the length of the argument list.

addr is the address of the argument list; and the data at **addr** is listed below.

function code specifies the type of file.

device-id identifies the device.

parameters describes the characteristics of the file processing to be performed. These parameters are optional.

Specifically, the argument list is:

Word	Symbol	Contents
Ø	.SPPFN	Function code, one of the following:

Code	Symbol	Function
1	.SPSFP	Sets spooled file parameters.
2	.SPSPR	Sets spooled parameters for renamed files.

Word	Symbol	Contents
1	.SPPDN	SIXBIT name of spooled device, channel number of spooled file, or the UDX of the device. The following words are optional
2	.SPPCP	Number of copies.
3	.SPPFM	SIXBIT forms name.
4	.SPPLM	Limit.

SPPRM. [CALLI 172]

Word	Symbol	Contents
5	.SPPSF	Spooling flags:

Bits	Symbol	Meaning
1-2	SP.DFR	Deferred/immediate flag.

Value	Symbol	Setting
1	.SPDFD	Deferred. The spooler should not start processing the file until the user job logs out.
2	.SPDFI	Immediate. Begin spooling immediately.

Bits	Symbol	Meaning
30-35	SP.TYP	Type of device, as reported by DEVTYP monitor call.

Word	Symbol	Contents
6	.SPPDA	Device attributes:

Bits	Symbol	Attribute
Ø	SP.UPC	Uppercase (LPT).
1	SP.LWC	Lowercase (LPT).
18	SP.PHY	Physical unit is given in SP.UNI.
28-35	SP.UNI	Physical unit number (if SP.PHY set).

7	.SPPND	Node at which processing is to be done.
10	.SPPAF	Time at which to begin processing (similar to /AFTER switch).
11	.SPNM1	In-your-behalf user name (word Ø of word pair) in SIXBIT.
12	.SPNM2	Second word of user name, in SIXBIT.
13	.SPMAX	Maximum length of argument block.

Normal Return

The specified parameters are set.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
-1	SPPAC%	Address check.
Ø	SPPIA%	Illegal length for argument list.
1	SPPID%	Illegal device.
2	SPPNA%	Device not assigned or initialized.
3	SPPNS%	Device not spooled.
4	SPPNC%	No free core for spooled parameter block.
5	SPPIF%	Illegal function code.

SPY [CALLI 42]

Function

Maps monitor section zero low-segment address space into your program's high segment. Use of the SPY monitor call requires that bit 16 (JP.SPA) or bit 17 (JP.SPM) in the privilege word .GTPRV be set. The SPY segment cannot be write-enabled.

The SPY monitor call can be used to examine the monitor during timesharing; it allows read-only access to monitor locations.

The SPY segment size cannot be changed by a CORE monitor call; if you attempt to do this, the CORE call will take its error return.

Calling Sequence

```
MOVEI   ac,monitoraddr
SPY     ac,
  error return
normal return
```

where: **monitoraddr** is the highest exec virtual (monitor) address desired. Monitor low-segment core from 0 to **monitoraddr** is mapped into user high-segment core from 400000 to 400000+**monitoraddr**. Therefore the value of **monitoraddr** can be any value between 0 and 377777.

Note that you cannot save this portion of memory with the SAVE. monitor call.

Normal Return

The desired monitor core is mapped into your program's high segment.

Error Return

The error return occurs if you use an invalid value for **monitoraddr**, or if your program does not have the required privileges.

Examples

This code maps some of the monitor's section zero low segment.

```
MOVE     T1,[%CNSIZ]
GETTAB   T1,
  HALT
SPY      T1,
  JRST      ERROR
```

Related Calls

PAGE., PEEK, POKE.

STATO [OPCODE Ø61]

Function

Tests the I/O status word for a device and skips if any of the specified bits are set. Use FILOP. to perform a STATO for an extended I/O channel. The I/O status bits are defined differently for each device. Therefore, the bits appropriate to each device are described in Volume 1 in the chapter on that device.

Calling Sequence

```
STATO channo,mask
  return 1
  return 2
```

where: **channo** is the number of an initialized channel.

mask is a halfword of bits, where each bit sets a corresponding bit in the I/O status word. The I/O status word is described in Volume 1, in each chapter that pertains to a specific device.

The I/O status bits are a set of 18 bits (right half) that reflect the current state of a file transmission. They are initially set by your program with the INIT/OPEN monitor call. Thereafter, the monitor sets the bits, but your program can test and reset them with any of several monitor calls.

Return

The call returns to **return 1** if all of the specified bits are Ø, or to **return 2** if any of the specified bits are set to 1.

Examples

See OPEN call.

Related Calls

STATZ

STATZ [OPCODE Ø63]

Function

Tests the I/O status (also called "file status") word for a device and skips if all of the specified bits are cleared. Use FILOP. to perform a STATZ on an extended I/O channel. For a complete list of I/O status bits, refer to the appropriate device chapter in Volume 1.

Calling Sequence

```
STATZ channo,mask
  return 1
  return 2
```

where: **channo** is the number of an initialized channel.

mask is a halfword in which each bit that you set corresponds to a bit in the I/O status word.

The I/O status bits are a set of 18 bits (right half) that reflect the current state of a file transmission. They are initially set by your program with the INIT/OPEN monitor call. Thereafter, the monitor sets the bits, but your program can test and reset them with any of several monitor calls.

Return

The call returns to **return 1** if one or more of the specified bits is 1, or to **return 2** if all of the specified bits are Ø.

Related Calls

STATO

STRUUO [CALLI 50]

Function

Modifies the search list for a job or for the system. Except for function Ø, the functions and calling sequence for the STRUUO monitor call are subject to change; therefore you should not use anything but function Ø in user programs. All functions except Ø are privileged functions.

Calling Sequence

```
        MOVE   ac,[XWD len,addr]
        STRUUO ac,
          error return
        normal return
        . . .
addr:   fcncode
        first argument
        . . .
        last argument
```

where: **len** is the length of the argument list.

addr is the address of the argument list. The format of the argument block is different depending on the function code specified in the first word of the argument block.

fcncode is one of the function codes described below; the words up through **last argument** are arguments for the given function. The function codes and their meanings are described below:

Code	Symbol	Function
Ø	.FSSRC	Defines a new job search list. The format of the argument block is shown below:

Word	Symbol	Contents
Ø	.FSFCN	The function code. (.FSSRC)
1	.FSCSO	Offset to first word of file structure block, as used for argument list to JOBSTR monitor call.

The first word of the argument block is followed by blocks of three words each. Each three-word block contains, in the first word, the structure name; in the second word, zero; and in the third word, flags that are described for the JOBSTR monitor call.

Code	Symbol	Function
		Your program cannot create files on a file structure unless it has access to the file structure. However, by using the .FSSRC function your program can add a file structure to its search list. If your program attempts to delete a file structure from its search list, the monitor moves the file structure's name from the job's active search list to its passive search list. To remove the file structure from the active or passive search list, issue the DISMOUNT monitor command.
1	.FSDSL	Defines a new search list for a job or for SYS. MOUNT uses this function to complete the mounting or dismounting procedures and to add or delete file structures from another job's search list. The argument block for the .FSDSL function is shown below:

Word	Symbol	Contents
Ø	.FSFCN	Function code. (.FSDSL)
1	.FSDJN	The number of the job whose search list is to be defined.
2	.FSDPP	The project-programmer number of the job.
3	.FSDFL	The flag word. If bit 35 is set, (DF.SRM), the monitor removes all deleted file structures from the job's search list and decrements the file structure's mount count. If bit 35 is not set, the monitor places all deleted file structures in the passive search list. To delete a file structure, it must have been present in the current search list and not listed in the argument block.
4	.FSDSO	Offset to first word of JOBSTR argument block.

The argument block contains as its first four words: the function code, a job number, a project-programmer number, and a flag word. These four words are followed by one or more three-word entries (each is an argument block like that used by the JOBSTR call). The entries specify the file structures to be in the search list. The order in which the file structures appear in the argument block is the order in which they will appear in the search list.

Code	Symbol	Function
		If the job number and the project-programmer number are both -1, the monitor assumes the search list for your job is to be defined. If the job number is Ø, the monitor ignores the project-programmer number and modifies the system search list (SYS). If a value other than -1 or Ø is specified, the monitor defines the search list of the job with the specified job number and project-programmer number. To indicate the FENCE, your program must substitute XWD Ø,Ø for SIXBIT/name/ in the first word of the three-word entry. When your program specifies the FENCE there will be three consecutive zero words in the three-word entry.
2	.FSDEF	Makes a new file structure available to users (for example, defines a new file structure). The file structure name, status, list of drives and their associated units (packs), and information for initializing components of the monitor data base are specified in the argument block. Specifically, the function does the following:

- o Builds a prototype structure data block.
- o Links and initializes all necessary Unit Data Blocks.
- o Allocates core and initializes the SPT tables and SAB rings.
- o Sets the state of the units to PACK MOUNTED.
- o Creates a TABSTR entry (assigns a number to the file structure).

The argument block for the .FSDEF function is shown below:

Word	Symbol	Contents
Ø	.FSFCN	Function code. (.FSDEF)
1	.FSNST	Pointer to the structure parameter block, in the form (length,,address).
2	.FSNUN	Pointer to the unit parameter block for unit Ø, in the form (length,,address).
3		Pointer to the unit parameter block for unit 1, in the form (length,,address).
.		
.		
.		

The structure parameter block is formatted as follows:

Word	Symbol	Contents
Ø	.FSSNM	File structure name in SIXBIT (word HOMSNM in the HOME block).
1	.FSSNU	Number of units in the structure.
2	.FSSHL	Highest logical block number (that is, .FSSBU times .FSSNU - 1).
3	.FSSSZ	Size (in blocks) of the file structure. (That is, the sum of the values of UNIBPU for each unit. The value of UNIBPU can be found in word .DCUSZ returned by the DSKCHR. call).
4	.FSSRQ	Obsolete.
5	.FSSRF	Obsolete.
6	.FSSTL	Number of FCFS (first-come, first-served) blocks left. (That is, the sum of the values of .FSUTL for each unit).
7	.FSSOD	Number of blocks allowed for overdraw (stored as a negative number). See HOMOVR in the HOME block.
1Ø	.FSSMP	First retrieval pointer to Master File Directory. See HOMPT1 in the HOME block.
11	.FSSML	-1 if .FSSMP is the only retrieval pointer to MFD. To set this word, you must read the RIB and test its contents. Do not use COPIPT from HOMUN1 in the HOME block.
12	.FSSUN	Logical unit number within the file structure where MFD begins. (See HOMUN1 from the HOME block.)
13	.FSSTR	Number of retries on an error. The suggested value for this word is 1Ø (decimal).
14	.FSSBU	Largest block on unit. (Largest value of UNIBPU, returned in .DCUSZ by DSKCHR.)

Word	Symbol	Contents
15	.FSSBC	Number of blocks per super-cluster (see HOMBSC in the HOME block).
16	.FSSSU	Number of super-clusters per unit (see HOMSCU in the HOME block).
17	.FSSIG	Obsolete.
2Ø	.FSSCC	Byte pointer to cluster count (see HOMCNP in the HOME block).
21	.FSSCK	Byte pointer to retrieval pointer checksum (see HOMCKP in the HOME block).
22	.FSSCA	Byte pointer to retrieval pointer cluster address (see HOMCLP in the HOME block).
23	.FSPVT	-1 if this is a private structure (see the HOPPVS bit in HOMPVS in the HOME block).
24	.FSPPN	PPN of file structure owner (each half is -1 if wild) (see HOMOPP in the HOME block).
25	.FSSCR	Block in structure containing RIB for CRASH.EXE (see HOMCRS in the HOME block).
26	.FSK4C	Number of K to reserve for CRASH.EXE on disk (see HOMK4C in the HOME block).

The format of the unit parameter block is:

Word	Symbol	Contents
Ø	.FSUNM	Unit name in SIXBIT (such as RPAØ).
1	.FSUID	Pack identifier (that is, the pack serial number in SIXBIT; see HOMHID in the HOME block).
2	.FSULN	Logical name within file structure (such as DSKBØ, DSKB1,...DSKB77; see HOMLOG in the HOME block).

Word	Symbol	Contents
3	.FSULU	Logical unit-number within file structure (0,1,2,....FSSNU-1) (see HOMLUN in the HOME block).
4	.FSUDS	Status bits. These are:

Bit	Symbol	Meaning
0	FS.UWL	Software write-lock.
1	FS.USA	Obsolete.

FS.UWL is meaningful only for the first unit in the structure.

5	.FSUGP	Number of sequential blocks to try for on sequential output (see HOMGRP in the HOME block).
6	.FSUTL	Number of free blocks on unit, minus a safety factor. The suggested safety factor is one block of safety for every 500 (decimal) blocks of disk. Do not allocate safety blocks for the swapping space. Thus, the suggested safety factor is (UNIBPU-HOMK4S*8)/500. This value should be truncated to less than 500 (decimal) blocks per unit.
7	.FSUBC	Number of blocks per cluster (see HOMBPC in the HOME block).
10	.FSUCS	Number of clusters per SAT (that is, (UNIBPU/HOMBPC-1)/HOMSPU+1).
11	.FSUWS	Number of words per SAT (that is, (.FSUCS-1)/36 + 1).
12	.FSUSC	Number of SATs in core (see HOMSIC in the HOME block).
13	.FSUSU	Number of SATs per unit (see HOMSPU in the HOME block).
14	.FSUSP	Pointer to SPT table (length,,address).
15	.FSUSB	First block for swapping (see HOMSLB in the HOME block).
16	.FSUKS	Number of K for SWAP.SYS. (See HOMK4S in the HOME block.)

Code	Symbol	Function

The format of the SPT table is:

Word	Contents
Ø	Pointer to 1st SAT block.
1	Pointer to 2nd SAT block.
.	
.	
.	
n	Pointer to nth SAT block.

Each word in the SPT table is in the form:

Bits	Contain
Ø-12	Number of free clusters in this SAT.
13-35	Address of SAT (as a cluster number).

3 .FSRDF Allows your program to change the status of a file structure if its mount count is Ø or 1. If the mount count is 1, the job number and project-programmer number arguments must be those for the job that has the structure mounted (in its search list). If the job number and project-programmer number are both -1, the search list for your job is assumed.

The argument block for the .FSRDF function is listed below.

Word	Symbol	Contents
Ø	.FSFCN	The function code (.FSRDF).
1	.FSRJN	The job number or -1.
2	.FSRPP	The project-programmer number or -1.
3	.FSRNM	The file structure name.
4	.FSRST	The new status bits to be assigned.

4 .FSLOK Allows your program to place a file structure in a state where no new LOOKUPs or ENTERs are allowed. The monitor will allow current reading and writing to continue until a CLOSE is issued. This function can be used to force a file structure into a dormant state so that it can be removed from the system with minimal damage to its users. For example, this function could be followed by .FSREM. The argument block for this function is described below:

Word	Symbol	Contents
Ø	.FSFCN	The function code (.FSRDF).
1	.FSLNM	The file structure name in SIXBIT.

Code	Symbol	Function
5	.FSREM	Removes a file structure from the system. This removal takes place immediately, with no regard for the users of the file structure. Normally, this function is preceded by the .FSLOK function to prepare the structure for removal. Specifically, the .FSREM function does the following:

- o Takes the non-error return if the file structure does not exist.
- o Removes the file structure name from the search list of all jobs and from the system search list.
- o Unlinks and returns to the free core pool any UFB or access blocks.
- o For every unit, sets the state to NO PACK MOUNTED and returns any core taken from the free core pool.
- o Clears KNOWLEDGE bits in the PPB and NMB blocks.
- o Unlinks STR data blocks and returns its core if taken from the free core pool.
- o Deletes (or marks for deletion) all sharable high segments initialized from the file structure.
- o Clears the TABSTR entry.
- o Takes the non-error return.

The argument block for the .FSREM function is shown below:

Word	Symbol	Contents
Ø	.FSFCN	The function code (.FSREM).
1	.FSMNM	The file structure name in SIXBIT.

6	.FSULK	Tests and sets the software write-lock bit associated with each UFD. This function is used, along with the .FSUCL function, to control programs (such as LOGOUT and LOGIN) attempting to modify a UFD at the same time. With the interlock bit set, only one program can modify the UFD at a time. Other programs should WAIT for the interlock to be cleared.

Code	Symbol	Function

The argument block for the .FSULK function is shown below.

Word	Symbol	Contents
Ø	.FSFCN	The function code (.FSULK).
1	.FSINM	The file structure name in SIXBIT.
2	.FSIPP	The PPN of the UFD.

If the interlock bit is set for a program other than your program, your program takes the error return.

7 .FSUCL — Clears the software interlock associated with a UFD. Once a program has cleared the interlock, another program may set the interlock (function .FSULK) and modify the UFD. The argument block for the .FSUCL function is shown below:

Word	Symbol	Contents
Ø	.FSFCN	The function code (.FSUCL).
1	.FSGNM	The file structure name in SIXBIT.
2	.FSGPP	The PPN of the UFD.

1Ø .FSETS — Tests error recovery procedures for the monitor and your programs by causing hard and soft errors to be simulated on the specified disk unit of the system.

NOTE

This function is obsolete and applies only to RP1Ø and RC1Ø controllers.

All error recovery and reporting procedures are followed through by the monitor as if a real error had occurred. This function causes the monitor to enter the simulated hard errors in the BAT block just as it would enter a real error. Therefore, field service should be notified of any error simulations that are being done.

This function is implemented only for disk packs and should not be attempted for the fixed-head disk because the counts will not be decremented. When a unit is removed from the system, the error test sequence is terminated.

Code Symbol Function

The argument block for the .FSETS function is listed below.

Word	Symbol	Contents
Ø	.FSFCN	The function code (.FSETS).
1	.FSEUN	The disk pack name.
2	.FSEGT	The number of good transfer interrupts before simulation of error.
3	.FSEDB	The number of bad DATAI's before the end of simulated error.
4	.FSEDO	Error DATAI bits ORed with DATAI bits received from the hardware.
5	.FSEDA	Error DATAI bits ANDCMed with DATAI bits received from the hardware.
6	.FSECB	The number of bad CONIs before terminating simulated error sequence.
7	.FSECO	Error CONI bits ORed with CONI bits received from the hardware.
1Ø	.FSECA	Error CONI bits ANDCMed with CONI bits received from the hardware.

Note that the CONI mentioned above is executed after a data transfer interrupt; the DATAI mentioned above is executed before connecting to the unit to initiate a position or transfer operation.

11 .FSNMW Modifies the 'nocreate' and 'write-lock' status of a file structure. The argument block for this function is shown below.

Word	Symbol	Contents
Ø	.FSFCN	The function code (.FSNMW).
1	.FSMFS	The file structure name.
2	.FSMFL	The flag word:

Bit	Symbol	Meaning
Ø	FS.MWL	Write-lock bit.
1	FS.MNC	No create bit.

Code	Symbol	Function
12	.FSCLR	Unlocks a file structure. The argument list is:

Word	Symbol	Contents
Ø	.FSFCN	The function code (.FSCLR).
1	.FSCFS	The file structure name in SIXBIT.

Normal Return

The function is performed.

Error Return

Before the monitor accepts the newly-defined search list by copying it in the PDB, it checks that the number of structures defined is less than the system-defined maximum limit for the job. This limit is stored in GETTAB table %LDMSS. If the number exceeds the maximum, the error return is taken and the ac is cleared.

Otherwise, one of the following error codes is returned in the ac:

Code	Symbol	Error
Ø	FSILF%	An illegal function code was specified.
1	FSSNF%	One or more of the specified file structures were not found.
2	FSSSA%	One or more of the specified file structures are in single-access mode.
3	FSILE%	One or more illegal entries are in the argument block.
4	FSTME%	There are too many entries in the search list.
5	FSUNA%	One or more of the specified units are not available.
6	FSPPN%	The specified job number and project-programmer number do not match.
7	FSMCN%	The mount count is greater than 1.
1Ø	FSNPV%	Your job is not privileged but should be.
11	FSFSA%	The specified file structure already exists.
12	FSILL%	The argument block length has been specified incorrectly.
13	FSUNC%	Unable to complete the call.
14	FSNFS%	The system has reached the maximum number of file structures.
15	FSNCS%	There is not enough free core for the data block.
16	FSUNF%	An illegal unit has been specified.
17	FSRSL%	A file structure name is repeated in a search list
2Ø	FSASL%	Structure contains units in active search list.
21	FSISN%	You specified an illegal structure name.

Related Calls

DSKCHR, GOBSTR, JOBSTR

SUSET. [CALLI 146]

Function

Selects a logical block number to be either read or written on subsequent IN/INPUT or OUT/OUTPUT monitor calls relating to either a file structure or a unit name. This call requires your program to have [1,2] or JACCT privileges.

The block number is relative to a file structure if the channel was initialized with a structure name (such as DSKB) and no file is open on the channel (that is, no LOOKUP or ENTER was performed).

The block number is relative to a unit number if the channel was initialized with a physical or logical unit name (such as RPA4 or DSKB) and no file is open on the channel (that is, no LOOKUP or ENTER was performed).

Refer to Section 11.7.5 for more detailed discussion of SUSET.

Calling Sequence

```
        MOVE    ac,[EXP flags]
        SUSET.  ac,
          error return
        normal return
```

where: flags are as follows:

Bits	Symbol	Meaning
Ø		Reserved.
1	SU.SOT	Output (input if not set).
2	SU.SMN	Maintenance cylinder. You can set this bit only if your job is logged in under [6,6]. SU.SBL (below) must contain the maintenance cylinder.
3		Reserved.
4-12	SU.SCH	Channel number. The channel number may be an extended channel number obtained from the FILOP. monitor call.
13-35	SU.SBL	Block number or maintenance cylinder (if SU.SMN is on).

Normal Return

The specified block will be the next one read/written on a subsequent IN/OUT monitor call. The SUSET. monitor call returns with the I/O status bit IO.BKT set if your job does not have enough privileges, or if the given block number is too large.

SUSET. [CALLI 146]

Error Return

The following error code is returned in the ac:

Code	Symbol	Error
-1	SUSNP%	Not enough privileges.

Related Calls

FILOP., USETI, USETO

SYSPHY [CALLI 51]

Function

Returns the name of a physical disk unit on the system.

Calling Sequence

```
{MOVEI   ac,Ø                }
{MOVE    ac,[SIXBIT/device/]}
 SYSPHY  ac,
   error return
 normal return
```

where: **device** is the physical unit name returned by a previous call (such as SIXBIT/RPAØ/).

Normal Return

If you set **ac** to Ø, the monitor returns the first physical disk name in **ac**. If you gave the name of a disk, the monitor returns the next physical disk name, or, if there are no more disks, the monitor returns Ø in the **ac**.

Error Return

The monitor takes the error return if the **device** you gave was neither Ø nor the name of a disk unit.

Examples

Example to get all unit names in system

```
          SETZB    T1,T2            ;T1=Table pointer, T2=unit
                                    ; name
LOOP:     SYSPHY   T2,              ;Get next one
            JRST   ERROR
          JUMPE    T2,CONTIN        ;Done if zero
          MOVEM    T2,PHYTAB(T1)    ;Save in table
          AOJA     T1,LOOP
PHYTAB:   BLOCK    ^D64
```

Related Calls

DVPHY., SYSSTR

SYSSTR [CALLI 46]

Function

Returns the name of a file structure on the system.

Calling Sequence

```
{MOVEI    ac,Ø                  }
{MOVE     ac,[SIXBIT/device/]   }
 SYSSTR   ac,
   error return
 normal return
```

where: device is the structure name returned by a previous call.

Normal Return

If you set ac to Ø, the monitor returns the first structure name in ac. If you gave the name of a structure, the monitor returns the next structure name in ac, or if there are no more structures, the monitor returns a Ø in the ac.

Error Return

The monitor takes the error return if the device you gave was neither Ø nor the name of a structure.

Examples

Example to get all file structure names on system

```
          SETZB    T1,T2            ;Use T1 as table index, T2 as
                                    ; structure name
LOOP:     SYSSTR   T2,              ;Get next structure
            JRST   ERROR
          JUMPE    T2,CONTIN        ;Done if structure is zero
          MOVEM    T2,STRTAB(T1)    ;Save in table
          AOJA     T1,LOOP          ;Get next one
STRTAB:   BLOCK    ^D64             ;Where to put structures
```

Related Calls

DVPHY., SYSPHY.

TAPOP. [CALLI 154]

Function

Performs various magnetic tape operations. Several TAPOP. functions are identical to or extensions of other monitor calls such as MTAPE and MTCHR. All TAPOP. functions assume that the specified device has been assigned to your job by the ASSIGN monitor command or the OPEN/INIT monitor call or that the calling job has SPY privileges.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        TAPOP.  ac,
          error return
        normal return
        . . .
addr:   EXP     fncode
       {SIXBIT/device/}
       {EXP     channo }
       {EXP     udx    }
        first argument
        . . .
        last argument
```

where: **len** is the length of the argument list.

addr is the address of the argument list.

fcncode is one of the function codes described below.

device is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device and the words up through **last argument** are arguments for the given function.

The function codes fall into four groups:

Codes	Functions
Ø - 777	Perform specific actions.
1ØØØ - 1777	Read parameters.
2ØØØ - 2777	Set parameters. These function codes are not explicitly listed in the descriptions below. To set a parameter, use the corresponding read function name plus the offset .TFSET (=1ØØØ). For example, to set the density indicator, use the read density indicator mnemonic plus .TFSET: .TFDEN+.TFSET
3ØØØ - 3777	Reserved for customer-defined functions.

The function codes and their meanings are:

Code	Symbol	Function
1	.TFWAT	Waits for I/O to be completed.
2	.TFREW	Rewinds tape to load point.
3	.TFUNL	Rewinds and unloads tape.
4	.TFFSB	Skips forward one block.
5	.TFFSF	Skips forward one file.
6	.TFSLE	Skips to logical end-of-tape.
7	.TFBSB	Skips backward one block.
10	.TFBSF	Skips backward one file.
11	.TFWTM	Writes a tape mark.
12	.TFWLG	Writes 3 inches of blank tape.
13	.TFDSE	Erases entire tape (data security, TX01/TX02 on DX10/DX20 only).
14	.TFWLE	Writes logical end-of-tape (two tape marks for unlabeled tapes).
15	.TFLBG	Gets the tape label device data block. Returns the name in ac. This is a privileged function for use by the label processor.
16	.TFLRL	Releases the tape label device data block. This is a privileged function for use by the label processor.
17	.TFLSU	Swaps units. This is a privileged function for use by the label processor.
20	.TFLDD	Destroys the tape label data base. This is a privileged function for use by the label processor.
21	.TFFEV	Forces end-of-volume processing. This allows your program to write the end-of-volume label before PULSAR finds the end-of-tape. The monitor assumes a multivolume file and automatically issues an operator MOUNT request for the next volume.
22	.TFURQ	Requests label processing. Set flag .TFCLE to clear a tape labelling error.
23	.TFSMM	Sets maintenance mode on the tape controller. This is a privileged function.
24	.TFCMM	Clears maintenance mode on the tape controller. This is a privileged function.
25	.TFCEC	Clears error counters. This privileged function is restricted for use by the tape label processor.

Code	Symbol	Function
1000	.TFTRY	Returns in the ac the number of retries on the last error.
1001	.TFDEN	Returns in ac the density code for the tape. To set the density code, use .TFDEN+.TFSET; the monitor reads the new density code from addr+2. Note that in order to set the density with this function code, IO.DEN must be zero. The density codes and their meanings are:

Code	Symbol	Density
0	.TFD00	Unit default.
1	.TFD20	200 bits/inch (8.1 rows/mm).
2	.TFD55	556 bits/inch (22.5 rows/mm).
3	.TFD80	800 bits/inch (32.2 rows/mm).
4	.TFD16	1600 bits/inch (65.3 rows/mm).
5	.TFD62	6250 bits/inch (255.5 rows/mm).
6-17		Reserved for use by DIGITAL.

1002	.TFKTP	Returns in the ac the controller type code for the tape. To set the controller type code, use .TFKTP+.TFSET; the monitor reads the new controller type code from addr+2. The controller type codes and their meanings are:

Code	Symbol	Controller Type
0	.TFKTA	TM10A (TU10/20/30/40/41).
1	.TFKTB	TM10B (TU10/20/30/40/41).
2	.TFKTC	TM10C (TU43).
3	.TFKTX	TX01/TX02 on DX10 (TU70,71,72).
4	.TFKTM	TM02/TM03 on RH10/RH20 (TU16,TU45).
5	.TFKRH	TM02/TM03 on RH11 (TU45).
6	.TFKD2	TX02 on DX20 (TU70,71,72).
7	.TFK78	TU78 on TM78.

1003	.TFRDB	Returns in the ac the read-backwards bit (TM02/TX01/TX02 only). The bit is on if the tape is set for read-backwards, or off for normal read. See Chapter 14. To set the read-backwards bit, use .TFRDB+.TFSET; the monitor reads the bit from addr+2.
1004	.TFLTH	Returns in the ac the bit for read next record at low threshold (TM10A/B/C only). The bit is on if the tape is set for low threshold, or off if not. To set the bit, use .TFLTH+.TFSET; The monitor reads the bit from addr+2.
1005	.TFPAR	Returns in the ac the status of the even parity bit (for 7-track tapes only). To set the status of the even parity bit, use .TFPAR+.TFSET; the monitor reads the status from addr+2.

Code	Symbol	Function
1006	.TFBSZ	Returns in the ac the block size for the tape. The returned value is one greater than the number of data words per record. To set the block size, use .TFBSZ+.TFSET; the monitor reads the block size from addr+2.
1007	.TFMOD	Returns in the ac the data mode code for the tape. To set the data mode code, use .TFMOD+.TFSET; the monitor reads the data mode code from addr+2. The data mode codes and their meanings are:

Code	Symbol	Data Mode
0	.TFMDD	DIGITAL-compatible core dump mode for 7-track and 9-track tapes. The monitor uses the default mode, either code 1 (.TFMID) or code 5 (.TFM7T).
1	.TFMID	DIGITAL-compatible core dump mode for 9-track tapes. The monitor reads and writes 36-bit words in 5 frames. This mode is also settable with MTDEC. monitor call.
2	.TFM8B	Industry-compatible 8-bit mode, with 4 bytes per word. This mode is also settable with the .MTIND monitor call, except that the default density for this mode is 1600 BPI.
3	.TFM6B	6-bit mode, 6 bytes per word (9-track, TU70 only).
4	.TFM7B	ANSI/ASCII 7-bit mode, 5 bytes per word (TU70 only).
5	.TFM7T	DIGITAL-compatible 7-track core dump mode (SIXBIT).

1010	.TFTRK	Returns in the ac the track status bit for the tape (0 for 9-track, 1 for 7-track). To set the track status bit, use .TFTRK+.TFSET; the monitor reads the bit from addr+2.
1011	.TFWLK	Returns in the ac the write-lock bit for the tape (1 if write-locked, 0 if not).
1012	.TFCNT	Returns in the ac the character count of the last record (the actual record length).
1013	.TFRID	Returns in the ac the SIXBIT reel identification for the tape. To set the reel identification, use .TFRID+.TFSET; the monitor reads the SIXBIT reel identification from addr+2.
1014	.TFCRC	Returns in the ac the last cyclic redundancy character (9-track NZRI only).

Code	Symbol	Function
1015	.TFSTS	Returns in the ac the unit status flags for the tape. To set the flags, use .TFSTS+.TFSET; the monitor reads the flags from addr+2. The unit status flags and their meanings are:

Bit	Symbol	Meaning
18	TF.UNS	Unit is not schedulable.
19	TF.BOT	Beginning-of-tape mark.
20	TF.WLK	Write-lock.
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 Returns unit statistics for the tape device. Your program supplies the function code and device at addr and addr+1. (These values are identical to those returned for the MTCHR. monitor call.) The monitor returns the device statistics at addr in the format:

Offset	Symbol	Contents
0	.TSFUN	Function code (user-supplied).
1	.TSDEV	Device (user-supplied).
2	.TSRID	SIXBIT reel identifier.
3	.TSFIL	Number of files read since the beginning of the tape.
4	.TSREC	Number of records since last tape unload.
5	.TSCRD	Number of characters read since last tape unload.
6	.TSCWR	Number of characters written since last tape unload.
7	.TSSRE	Soft read errors since last tape unload.
10	.TSHRE	Hard read errors since last tape unload.
11	.TSSWE	Soft write errors since last tape unload.
12	.TSHWE	Hard write errors since last tape unload.
13	.TSTME	Total number of errors since last tape unload.
14	.TSTDE	Total device errors since system startup.
15	.TSTUN	Total unloads since last system reload.
16	.TSRTY	Number of retries to resolve last error.
17	.TSCCR	Character count of last record read or written.
20	.TSPBE	Position before last error: file number (in left half); record number (in right half).
21	.TSFES	Final error state. See the TOPS-10/TOPS-20 SPEAR Manual.

Code	Symbol	Function
1017	.TFIEP	Returns in the ac the initial error pointer.
1020	.TFFEP	Returns in the ac the final error pointer.

NOTE

Function codes 1021 and 1022 return the blocks pointed to by 1017 and 1020. These blocks are for communication of errors to DAEMON and may change without notice.

1021	.TFIER	Returns in the ac the initial error status.
1022	.TFFER	Returns in the ac the final error status.
1023	.TFFED	Returns in the ac the final error disposition.
1024	.TFLBL	Returns in the ac the label processing type code. To set the label processing type code, use .TFLBL+.TFSET; the monitor reads the new code from addr+2. The label processing type codes and their meanings are:

Code	Symbol	Label Processing Type
0	.TFLBP	Bypass label processing. To set this value, the job must be privileged.
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	Nonstandard labels.
7	.TFLNL	No labels. When tapes are processed with no labels, the label processor is used only to verify that the tape does not contain a tape label. Unlabeled tapes can be copied to create a labeled tape.
10	.TFCBA	DIGITAL COBOL ASCII labels.
11	.TFCBS	DIGITAL COBOL SIXBIT labels.
12	.TFLNV	Same as .TFLNL except that user program is responsible for dealing with an EOT. This type is the default. To switch reels after end-of-tape, use TAPOP. function .TFFEV.

1025	.TFPLT	Performs functions identical to the .TFLBL function 1024 above, except that it allows access to files and tape labels. Using this function, you can examine and modify the contents of a label. The .TFPLT function requires the JP.POK, [1,2], or JACCT privilege.

Code	Symbol	Function
1026	.TFLTC	Returns the last tape label termination code from the tape label processor. It is recommended that you use DEVOP. function .DFRES, because more information can be returned by that function. The return codes are:

Code	Symbol	Error
1	.TFTCP	Continue processing.
2	.TFTRE	Returned 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.
12	.TFPSE	Positioning error.
13	.TFBOT	Beginning of tape.
14	.TFIOP	Illegal operation.
15	.TFFNF	File not found.
16	.TFCAN	Operator cancelled request.
17	.TFTMV	Too many volumes requested.

1027	.TFDMS	Returns in the ac the diagnostic mode set bit (TX01/TX02 on DX10 only). This bit is 1 for diagnostic mode, otherwise 0. To set this bit, use .TFDMS+.TFSET; the monitor reads the new bit from addr+2.
1030	.TFFSO	Returns in the ac the bit showing whether a forced SENSE command will be issued to the controller (TX01/TX02 on DX10/DX20 only) after the completion of every operation. To set the bit, use .TFFSO+.TFSET; the monitor reads the new bit from addr+2. This bit should be set by diagnostic programs only, because it slows down tape operations considerably.
1031	.TFMFC	Returns in the ac the maximum frame count. To set the count, use .TFMFC+.TFSET; the monitor reads the new count from addr+2. Use this function to speed tape throughput for a TU16, or TU45, TU70, TU71, or TU72 that does not have an integral number of bytes per word. The count stays in effect until your program performs a RESET with another TAPOP. monitor call or until the tape is RELEASed (if the device was ASSIGNed). This function allows a TU70 or a TU16 to read and write tapes that do not have an integral number of bytes per word. This function provides tape compatibility with other systems.

Code	Symbol	Function
1Ø32	.TFPDN	Returns in the ac flags showing the possible densities for a tape. The flags and their meanings are:

Bit	Symbol	Density
31	TF.DN5	625Ø bits/inch (255.5 rows/mm)
32	TF.DN4	16ØØ bits/inch (65.3 rows/mm).
33	TF.DN3	8ØØ bits/inch (32.2 rows/mm).
34	TF.DN2	55Ø bits/inch (22.5 rows/mm).
35	TF.DN1	2ØØ bits/inch (8.1 rows/mm).

1Ø33 .TFLPR Returns at **addr+2** the tape label parameters. This function causes the first input label processing if there is no file open for input. To set the parameters, use .TFLPR+.TFSET; the monitor reads the parameters beginning at **addr+2**. This set function is legal only if there is no file open for output on the given channel. The parameters given apply to the next file to be written. The format of the parameters at addr is:

Offset	Symbol	Contents
Ø	.TPFUN	Function code (user-supplied).
1	.TPDEV	Device (user-supplied).
2	.TPREC	Record format and form control:

Bits	Symbol	Meaning
Ø-17	TR.FCT	Forms control byte; one of the following codes:

Code	Symbol	Meaning
1	.TFCNO	Records on tape do not contain form control characters.
2	.TFCAS	First character of each record is a form control character.
3	.TFCAM	Records on tape contain all required form control characters.

Bits	Symbol	Meaning
18-35	TR.RFM	Record format byte; one of the following codes:

Code	Symbol	Meaning
Ø	.TRFDF	Default (Fixed).
1	.TRFFX	Fixed (F).
2	.TRFVR	Variable (D).
3	.TRFSP	Spanned (S).
4	.TRFUN	Undefined (U).

Offset	Symbol	Contents
3	.TPRSZ	Record size in characters.
4	.TPBSZ	Block size in characters.
5	.TPEXP	Expiration date in 15-bit format.

Bits	Symbol	Meaning
18-35	TP.EEX	Expiration date.
Ø-17	TP.ECR	Creation date.

6	.TPPRO	Protection code.
7	.TPSEQ	File sequence number.
1Ø	.TPFNM	File name (17 ASCII characters maximum).
14	.TPGEN	Generation and version numbers.

Bits	Symbol	Meaning
Ø-17	TP.GEN	Generation number.
18-35	TP.VER	Generation version number.

Function code 2Ø33 (to set tape label parameters) is legal only if there is no file currently open for output on the specified channel. The numbers you specify must apply to the next file to be written.

Function code 1Ø33 (to read label parameters) always returns these numbers and causes the first input label processing if there is no file open for input.

Normal Return

The function is performed.

TAPOP. [CALLI 154]

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
-1	TPACS%	Address check while storing answer.
Ø	TPIFC%	Illegal function code.
1	TPPRV%	Not enough privileges.
2	TPNMT%	Not a magtape device.
3	TPVOR%	Specified value out of range.
4	TPACR%	Address check reading arguments.
5	TPCBS%	Parameter cannot be set.
6	TPNIA%	Tape not initialized or assigned.
7	TPNLP%	No label processor.
1Ø	TPETC%	Termination code error.
11	TPIJN%	Illegal job number.
12	TPLRF%	Label release function required.
13	TPLSI%	Set label parameter function illegal after first output.
14	TPLOE%	Attempted to read information from a label DDB owned by someone else.
15	TPDNC%	Drive not capable of specified density.
16	TPWWL%	Write attempted to write-locked tape.

Examples

```
          MOVE    T1,[XWD 2,ARGLST]  ;Pointer to arg list
          TAPOP.  T1,                ;Get controller type
           JRST   ERROR              ;Error
          CAIE    T1,.TFKTX          ;If DX1Ø
          CAIN    T1,.TFKD2          ;Or DX2Ø
          SKIPA                      ;Data security erase
                                     ;  will work
          JRST    NODSE              ;Otherwise don't try it
          MOVE    T1,[XWD 2,DSELSI]  ;Pointer to arg list
          TAPOP.  T1,                ;Erase the entire tape
           JRST   ERROR              ;Error
          JRST    CONTIN             ;Skip argument blocks
ARGLST:   EXP     .TFKTP             ;Function to read
                                     ;  controller types
          SIXBIT  /TAPE/             ;Device is "TAPE"
DSELST:   EXP     .TFDSE             ;Function to do
                                     ;  data security erase
          SIXBIT  /TAPE/             ;Device is "TAPE"
CONTIN:                              ;Continue
```

This example performs a data security erase on the logical device "TAPE" if and only if the controller is capable of doing so. (DX1Ø/DX2Ø).

Related Calls

MTAID., MTAPE, MTCHR.

TIMER [CALLI 22]

Function

Returns the time of day since midnight (00:00) in jiffies. (A jiffy is 1/60 second.)

NOTE

For systems using 50 Hz power, jiffy = 1/50 second. Therefore it is good programming practice to use the MSTIME monitor call on any system, because MTSIME call gives the time of day in milliseconds and is independent of the type of power used.

Calling Sequence

```
TIMER   ac,
only return
```

Return

The number of jiffies since midnight is returned in the ac.

Related Calls

DATE, MSTIME

TMPCOR [CALLI 44]

Function

Creates, reads, writes, or manipulates temporary files left in core from the running of one program to another. Those files are referenced by a three-character file name. All files are deleted when the job is logged out. If the monitor call fails, your program should write DSK:nnnNAM.TMP, where nnn is the job number. This arrangement improves response time and minimizes the number of disk reads.

Calling Sequence

```
        MOVE    ac,[XWD fcncode,addr]
        TMPCOR  ac,
          error return
        normal return
        . . .
addr:   XWD     'nam',0
        IOWD    buflength,buffer
```

where: **fcncode** is one of the function codes described below.

addr gives the address of the argument list.

nam is a 3-character SIXBIT string that is the file name.

buflength is the length of the buffer for the call.

buffer gives the address of the buffer for the call.

The function codes and their meanings are:

Code	Symbol	Function
0	.TCRFS	Obtains free space. For this function, set the ac to 0 before the call; no argument list is required. On a normal return, the ac contains the number of free words available to your program (510 decimal).
1	.TCRRF	Reads a file. The length of the file is returned in the ac, and as much of the file as possible is copied into the buffer for the call. You can check for truncation by comparing the ac to buflength. The error return occurs if the specified file is not found; in this case, the number of free words available to your program is returned in the ac.
2	.TCRDF	Reads and deletes a file. Performs all the same functions as .TCRRF and in addition deletes the file. Note that the file is deleted even if it is too long to fit in the buffer for the call.

Code	Symbol	Function
3	.TCRWF	Writes the contents of the buffer into a file. The requested length of the file is the value of buflength. If there is already a file of the specified name, it is deleted and the space is reclaimed. The requested size of the file is specified by buflength. If there is not enough space to write the entire file, nothing is written, the ac is set to the number of free words of space available to the user, and the error return is taken. If there is enough space, the file is written. The ac is set to the amount of space left after the file has been written and the normal return is taken. If insufficient space is available, none of the file is written, the error return occurs, and the number of free words available to your program is returned in the ac.
4	.TCRRD	Reads a directory. The number of .TMP files in your directory is returned in the ac, and their file names are written into the buffer for the call. You can check for truncation of the directory list by comparing the ac to buflength. Each entry in the buffer is of the form: XWD 'nam',length where: length is the length of the file in words. nam is the file name. The error return occurs only if the call is not implemented.
5	.TCRDD	Reads and deletes from directory. This performs all the same functions as .TCRRD and in addition deletes all files from your directory.

Examples

```
        MOVE    T1,[XWD .TCRWF,ARGLST]
        TMPCOR  T1,
          JRST  TMCERR
        JRST    CONTIN
ARGLST: XWD     'XYZ',Ø
        IOWD    <BUFEND-BUFFER>,BUFFER
BUFFER: ASCIZ /THIS IS THE TEXT FOR THE FILE./
BUFEND:
CONTIN:
```

This example writes the text at BUFFER into the file XYZ if space is available.

TRMNO. [CALLI 115]

Function

Returns the number of the terminal controlling a specified job.

Calling Sequence

```
{MOVEI   ac,jobno}
{MOVNI   ac,1    }
 TRMNO.  ac,
   error return
 normal return
```

where: jobno is the number of a logged-in job (use -1 for the current job).

Normal Return

Returns the UDX for the controlling terminal in the ac. The format of Universal Device Index names is .UXxxx. The range of values is 200000 through 200777 (octal). The symbol .UXTRM (200000) is the offset for the terminal indexes.

Error Return

Zero is returned in the ac and indicates one of the errors listed below:

- The job is currently detached (that is, there is no controlling terminal).
- The job number specified is unassigned.
- The job number specified is illegal.
- The job number specified is a negative number other than -1.

Examples

Your program can determine which of the above error conditions occurred by using the JOBSTS monitor call. An example of a program using this call for this purpose is shown below.

```
MOVE    T1,JOBN
TRMNO.  T1,
  JRST  .+2
JRST    OK              ;No error
MOVNI   T1,JOBN
PJOB    T1,
MOVNS   T1,
JOBSTS  T1,
  JRST  ILLNUM          ;Job number illegal
JUMPL   T1,DETJOB       ;Job is detached
JRST    NOJOB           ;No such job
```

Related Calls

TRMOP.

Common Errors

Using .UXTRM as a mask instead of an offset.

TRMOP. [CALLI 116]

Function

Performs various operations for terminals. Several TRMOP. functions are identical to, or extensions of, TTCALL monitor calls.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        TRMOP.  ac,
          error return
        normal return
        . . .
addr:   EXP     fcncode
        EXP     udx
        first argument
        . . .
        last argument
```

where: **len** is the length of the argument list.

addr is the address of the argument list.

fcncode is one of the function codes described below.

udx is the Universal Device Index for a terminal, or -1 can be used to indicate the program's controlling terminal. The words up through **last argument** are arguments for the given function.

The argument list is formatted as follows:

Word	Symbol	Contents
Ø	.TOFNC	Function code.
1	.TOUDX	Universal Device Index or -1.
2	.TOAR2	Argument word.
3	.TOAR3	Argument word.
4	.TOAR4	Argument word.
5	.TOAR5	Argument word.
6	.TOAR6	Argument word.

The argument words contain values that are read or set by the function code, and they differ for each function. The function codes fall into four groups:

Codes	Symbol	Actions
Ø-777		Perform a specific action.
1ØØØ-1777		Read parameters.
1ØØØ	.TOSET	Add this value (symbol .TOSET) to the read function, thus setting the specified parameter(s).

Codes	Symbol	Actions
2000 - 2777		Set parameters. These functions are not explicitly listed in the descriptions below. To set a parameter, use the corresponding Read function plus the offset .TOSET (=1000). For example, to set a terminal's receive speed, use the receive speed function plus .TOSET: .TORSP + .TOSET
3000 - 3777		Reserved for customer-defined functions.

The function codes and their meanings are:

Code	Symbol	Function
1	.TOSIP	Takes the error (non-skip) return if the terminal's input buffer is empty. The ac is unchanged.
2	.TOSOP	Takes the error (non-skip) return if the terminal's output buffer is empty. The ac is unchanged.
3	.TOCIB	Clears the terminal's input buffer.
4	.TOCOB	Clears the terminal's output buffer.
5	.TOOUC	Outputs a character to the terminal; the character is in bits 29 to 35 of .TOAR2.
6	.TOOIC	Outputs an image-mode character to the terminal; the character is in bits 28 to 35 of .TOAR2.
7	.TOOUS	Outputs an ASCIZ string to the terminal; the address of the string is in .TOAR2. If the job number at the receiving terminal is different from your job number, the character string is limited to 128 characters. If you attempt to send more than 128 characters to a job other than your own, characters may be lost.
10	.TOINC	Inputs a character from the terminal in line mode; the character is stored in bits 29 to 35 of the ac.
11	.TOIIC	Inputs an image-mode character from the terminal; the character is stored in bits 28 to 35 of the ac. This function is not implemented by the monitor.
12	.TODSE	Enables a modem (dataset) for outgoing calls. This function is not implemented by the monitor.
13	.TODSC	Enables and places outgoing calls on a modem with a dialer. A telephone number of up to 17 decimal digits is stored in 4-bit bytes in .TOAR2 and .TOAR3 (terminated by a 17). If the caller must wait for a second dial tone (for example, after dialing 9), a 5-second wait is indicated by a 16 byte.

Code	Symbol	Function
14	.TODSF	Disconnects a call (for example, hangs up a modem).
15	.TORSC	Rescans an input line.
16	.TOELE	Sets the terminal element to the number stored in .TOAR2.
17	.TOEAB	Enables autobaud detection.
20	.TOISC	Inputs a character from the terminal to the ac; waiting in character mode if no input is available.
21	.TOTYP	Puts an ASCIZ string into the terminal's input buffer; the address of the string is in .TOAR2. A string of more than 300 characters results in a range error.
22	.TOGMS	Returns terminal's MIC status bits in addr+2. If MIC is not controlling the job, addr+2 contains 0. The status bits are as follows:

Bits	Symbol	Meaning
0	TO.CHK	Flag bit for word. If this bit is set, there are other set bits in the word.
1	TO.CCT	A CTRL/C was typed.
2	TO.OCS	An operator-sent character was received.
3	TO.ECS	An error character was received.
4	TO.CPT	A CTRL/P was typed.
5	TO.CBT	A CTRL/B was typed.
6	TO.STL	Silence this line.
7	TO.LMM	Line is in monitor mode.
8	TO.LUM	Line is in user mode.
9	TO.C1O	Line is in column 1 on output.
10	TO.CAT	A CTRL/A was typed.
11	TO.RSP	Error response.
12	TO.RSY	Response code sync.
13	TO.LOG	MIC is logging.
14	TO.LUI	Controlling job should do a JOBSTS UUO for the controlled job or terminal.
15-21	TO.AOC	The received operator character (ASCII).
22-28	TO.AEC	The received error character (ASCII).
29-35	TO.MMJ	MIC master job number.

23	.TOSMS	Sets the terminal's MIC status bits to the contents of .TOAR2. The bits are the same as those returned by the .TOGMS function above.
24	.TOCLR	Clears the MIC status bits.
25	.TODSP	Displays an ASCIZ string on the terminal. The address of the string is in .TOAR2.
26	.TOGMR	Returns the MIC response buffer. The address of the 21-word buffer is in .TOAR2.

Code	Symbol	Function
27	.TOLOG	Returns the MIC log buffer. The address of the 21-word buffer is in .TOAR2.
30	.TODSS	A modem is present if bit Ø of the ac is set on return; bits 1 to 35 are reserved. If the line is not a dataset line, the monitor takes the error return and returns the TOIMP% error in ac.
31	.TOSBS	Sets the terminal break character set. Using this function, you can define the characters that, when typed on the terminal, will be interpreted by the monitor as break characters, indicating the end of the input line. In the argument block, you must specify the following:

```
addr:   .TOSBS          ;function code
        udx             ;terminal's UDX
        field-width     ;auto-break
        break mask      ;first word
        . . .           ; of break table
        break mask      ;last word
```

Where the field-width defines the number of characters to be accepted on an input line. After the specified number of characters are typed, a break is automatically made. The field width must be between 1 and 255.

The break mask is a 4-word block indicating the mask of bits (in the left-hand 32 bits of each word from .TOAR3 through .TOAR6) that indicate the octal representation of characters to be defined as break characters. You must enable break set mode by setting flag IO.ABS in the I/O status word. Refer to Volume 1 for more information.

32	.TORBS	Reads the terminal break character set. The field width is returned in .TOAR2 of the argument block, and the break mask is returned in words .TOAR3 through .TOAR6. Refer to .TOSBS.
33	.TOISO	Sets counted image output string mode. This function allows your program to output a specified number of characters in a single sequence. This function allows screen editors and display-oriented programs to update the terminal screen more efficiently. The argument block for this function is:

```
addr:   .TOISO          ;function
        udx             ;terminal's UDX
        byte-size,,byte-count
        string-address
```

Where the size of each byte (7 or 8 bits) is specified in byte-size, and the length of the string is specified in byte-count. The string address is a pointer to the location of the output string.

Codes	Symbol	Function
34	.TOFLM	Returns the carriage to the left margin.
35	.TOGCS	Reads the special character status. The conditions read are set in the TC.VAL field. .TOSCS explains TC.VAL. The argument block for .TOGCS is: addr: .TOGCS udx len2,,addr2 addr2: character to read The number of words given in len at addr2 have the TC.VAL field filled in from the current settings described in .TOSCS.
36	.TOSCS	Sets the special character status. The .TOSCS argument block is: addr: .TOSCS udx len2,,addr2 addr2: mask + values + character The fields in addr2 are:

Bits	Symbol	Meaning
Ø-13	TC.MOD	The mask of the fields to change.
14-27	TC.VAL	The conditions read or set. See below for a list of condition bits.
28-35	TC.CHR	The character which the condition(s) applies to.

Bits which may be selected for the character are:

Bit	Symbol	Meaning
31	TC.CLR	For control characters only, clears the input buffer when the character's interrupt is posted.
32	TC.DFR	Defer the character's interrupt type.
33	TC.OOB	An out-of-band character. This character causes an interrupt when received.
34	TC.NSA	Disable special action.
35	TC.BRK	Line break character.

37 .TOUNR Allows reading of only already echoed characters. No further echoing occurs until an empty buffer is returned, a no input available return is taken, or a null character is returned. The argument block is:

```
addr:   .TOUNR
        udx         ;terminal UDX
```

Code	Symbol	Function
40	.TOASO	Sets counted ASCII output string mode. The argument block is identical to .TOISO, function 33, except for the function code.
41	.TODNT	Disconnects a network terminal, without hanging up the dataset.
1000	.TOOIP	Returns, in the ac, the output-in-progress bit (1 in Bit 35 if output is in progress, otherwise ac=0).
1001	.TOCOM	Returns, in the ac, the monitor-mode bit (1 in bit 35 if terminal is in monitor mode, otherwise ac=0).
1002	.TOXON	Returns, in the ac, the papertape bit (1 in bit 35 if terminal is in papertape mode, otherwise ac=0). To set the bit, use .TOXON+.TOSET; the monitor reads the bit from .TOAR2. When this bit is set, the functions of CTRL/S and CTRL/Q are defined to control the papertape. If CTRL/S and CTRL/Q were defined previously for stopping and continuing terminal output, these functions are temporarily superseded by the papertape function. When you clear .TOXON, the terminal output function is restored.
1003	.TOLCT	Returns, in the ac, the lowercase bit (1 in bit 35 if no lowercase capability, otherwise ac=0). To set the bit, use .TOLCT+.TOSET; the monitor reads the bit from .TOAR2.
1004	.TOSLV	Returns, in the ac, the slave bit (1 in bit 35 if the terminal is slaved, otherwise ac=0). To set the bit, use .TOSLV+.TOSET; the monitor reads the bit from .TOAR2.
1005	.TOTAB	Returns, in the ac, the tab-capability bit for the terminal (1 in bit 35 if the terminal has tab capability, otherwise ac=0). To set the bit, use .TOTAB+.TOSET; the monitor reads the bit from .TOAR2.
1006	.TOFRM	Returns, in the ac, the formfeed-capability bit for the terminal (1 in bit 35 if the terminal performs formfeeds, otherwise ac=0). To set the bit, use .TOFRM+.TOSET; the monitor reads the bit from .TOAR2.
1007	.TOLCP	Returns, in the ac, the local-copy bit for the terminal (1 in bit 35 if characters are not echoed, otherwise ac=0). To set the bit, use .TOLCP+.TOSET; the monitor reads the bit from .TOAR2.

Code	Symbol	Function
1010	.TONFC	Returns, in the ac, the free CRLF bit for the terminal (1 in bit 35 if free CRLFs are not performed, otherwise ac=0). To set the bit, use .TONFC+.TOSET; the monitor reads the bit from .TOAR2. The free CRLF (carriage-return/line-feed) is placed in the terminal output buffer when the maximum width of the line is reached. Set the terminal line width using .TOWID, SET TTY WIDTH monitor command, or by setting the terminal type. The free CRLF function is enabled by default.
1011	.TOHPS	Returns, in the ac, the horizontal position of the carriage or cursor (in the range 0 to octal 377).
1012	.TOWID	Returns, in the ac, the carriage width for the terminal (in the range 16 to 255 decimal). To set this value, use .TOWID+.TOSET; the monitor reads the width from .TOAR2.
1013	.TOSND	Returns, in the ac, the GAG bit for the terminal (1 in bit 35 if NOGAG, otherwise ac=0). To set this bit, use .TOSND+.TOSET; the monitor reads the bit from .TOAR2. Refer to the SET TTY monitor command in the Commands Manual.
1014	.TOHLF	Returns, in the ac, the half-duplex bit for the terminal (1 in bit 35 if the terminal is in half-duplex mode, otherwise ac=0). To set this bit, use .TOHLF+.TOSET; the monitor reads the bit from .TOAR2. Refer to the SET TTY monitor command in the Commands Manual.
1015	.TORMT	Returns, in the ac, the remote bit for the terminal (1 in bit 35 if the terminal is remote, otherwise ac=0). To set this bit, use .TORMT+.TOSET; the monitor reads the bit from .TOAR2. Refer to the SET TTY monitor command in the Commands Manual.
1016	.TODIS	Returns, in the ac, the display bit for the terminal (1 in bit 35 if the terminal is a display device, otherwise ac=0). To set this bit, use .TODIS+.TOSET; the monitor reads the bit from .TOAR2. You may set this bit to indicate that the terminal is a display terminal if any of the following is true: o The terminal can backspace the cursor. o A space character on the terminal erases the character pointed to by the cursor. Refer to the SET TTY monitor command in the Commands Manual.

Code	Symbol	Function
1017	.TOFLC	Returns, in the ac, the filler class code (the number of fillers used for padding) for the terminal (in the range Ø to 3). To set the code, use .TOFLC+.TOSET; the monitor reads the code from .TOAR2. Refer to the SET TTY monitor command in the Commands Manual.
1020	.TOTAP	Returns, in the ac, the papertape-enable bit for the terminal (1 in bit 35 if papertape is enabled, otherwise ac=Ø). To set this bit, use .TOTAP+.TOSET; the monitor reads the bit from .TOAR2. Refer to the SET TTY monitor command in the Commands Manual.
1021	.TOPAG	Returns, in the ac, the bit setting for paged display mode (1 in bit 35 if the terminal is in paged display mode, otherwise ac=Ø). To set this bit, use .TOPAG+.TOSET; the monitor reads the bit from .TOAR2.
1022	.TOSTP	Returns, in the ac, the output-stopped bit for the terminal (1 in bit 35 if output has stopped, otherwise ac=Ø). The output-stopped bit is set when, for example, the terminal reaches its page limit.
1023	.TOPSZ	Returns, in the ac, the number of lines for the current page setting (in the range Ø to 63). To set the number of lines, use .TOPSZ+.TOSET; the monitor reads the number of lines from .TOAR2.
1024	.TOPCT	Returns, in the ac, the value of the page counter (in the range Ø to 63).
1025	.TOBLK	Returns, in the ac, the bit setting for blank line handling (1 in bit 35 if multiple blank lines are to be reduced to one blank line, otherwise ac=Ø). To set the bit, use .TOBLK+.TOSET; the monitor reads the bit setting from .TOAR2.
1026	.TOALT	Returns, in the ac, the bit setting for ESCape (altmode) character handling (1 in bit 35 if no conversion, Ø if the ASCII codes 175 and 176 are converted to Ø33). To set the bit, use .TOALT+.TOSET; the monitor reads the bit setting from .TOAR2.
1027	.TOAPL	Returns, in the ac, the bit setting for APL mode (1 in bit 35 if in APL mode, otherwise ac=Ø). To set the bit, use .TOAPL+.TOSET; the monitor reads the bit setting from .TOAR2.

Code	Symbol	Function
1030	.TORSP	Returns, in the ac, the code for the terminal's receive speed. To set the code, use .TORSP+.TOSET; the monitor reads the code from .TOAR2. (Note that the send and receive speeds cannot both be 0.) The codes and their meanings are:

Code	Symbol	Speed
1	.TO005	50 baud.
2	.TO007	75 baud.
3	.TO011	110 baud.
4	.TO013	134.5 baud.
5	.TO015	150 baud.
6	.TO020	200 baud.
7	.TO030	300 baud.
10	.TO060	600 baud.
11	.TO120	1200 baud.
12	.TO180	1800 baud.
13	.TO240	2400 baud.
14	.TO480	4800 baud.
15	.TO960	9600 baud.
16	.TOEXA	External A.
17	.TOEXB	External B.

1031	.TOTSP	Returns, in the ac, the code for the terminal's transmit speed. To set the code, use .TORSP+.TOSET; the monitor reads the code from .TOAR2. (Note that the send and receive speeds cannot both be 0.) The codes and their meanings are the same as those for the .TORSP function above.
1032	.TODBK	Returns, in the ac, the bit setting for the terminal's debreak capability (1 in bit 35 if debreak is enabled, otherwise ac=0). To set the bit, use .TODBK+.TOSET; the monitor reads the bit setting from .TOAR2. (Meaningful for model 2741 terminals only.)
1033	.TO274	Returns, in the ac, the bit to show whether the terminal is a 2741 (1 in bit 35 if so, otherwise ac=0). To set the bit, use .TO274+.TOSET; the monitor reads the bit from .TOAR2. You must have [1,2] or JACCT privileges to set this bit, because the 2741 terminal is no longer supported.
1034	.TOTDY	Returns, in the ac, the terminal's TIDY setting (1 in bit 35 if TIDY, 0 if NOTIDY). (Meaningful for model 2741 terminals only.)

Code	Symbol	Function
1035	.TOACR	Returns, in the ac, the auto-CRLF column number. If this value is not zero, the first space character received from the terminal, after the specified column, is converted to a carriage-return/line-feed sequence. If the value in the ac is zero, no automatic conversion on input is performed. To set this value, use .TOACR+.TOSET. Include the column number in .TOAR2 as a decimal value from Ø to 255.
1036	.TORTC	Returns, in the ac, the bit for CTRL/R and CTRL/T compatibility (Ø in bit 35 if compatibility is enabled, otherwise ac=1). To set the bit, use .TORTC; the monitor reads the bit from .TOAR2.
1037	.TOPBS	Returns, in the ac, the word containing the PIM (packed image mode) break set (four 9-bit bytes). To set this word, use .TOPBS+.TOSET; the monitor reads the word from .TOAR2. If the ninth bit of the argument is set, the bytes are compared as 7-bit bytes. If the ninth bit is clear, the bytes are compared as 8-bit bytes.
1040	.TODEM	Returns, in the ac, the bit showing the deferred-echo mode (1 in bit 35 if echo is deferred until input is required, otherwise ac=Ø). To set this bit, use .TODEM+.TOSET; the monitor reads the bit from .TOAR2.
1041	.TOTRM	Returns, in the ac, the SIXBIT terminal type. To set the terminal type code, use .TOTRM+.TOSET; the monitor reads the SIXBIT name of the terminal type from .TOAR2. The valid terminal types may be obtained from GETTAB table .GTTNM.
1042	.TOBCT	Returns, in the ac, number of commands processed in the left half, and the number of break characters received in the right half.
1043	.TOICT	Returns, in the ac, number of input characters received.
1044	.TOOCT	Returns, in the ac, number of output characters sent.
1045	.TOOSU	Returns, in the ac, output suppression state (CTRL/O).
1046	.TOFCS	Returns, in the ac, full character set.
1047	.TOBKA	"Break on all characters" mode. If this is off, the break occurs on each line. If it is set, breaks occur on each character.

Code	Symbol	Function
1050		Reserved for use by DIGITAL.
1051		Reserved for use by DIGITAL.
1052	.TOTIC	Returns number of characters in input buffer.
1053		Reserved for use by DIGITAL.
1054	.TOBKC	Returns number of break characters in input buffer.
1055	.TOECC	Returns number of unprocessed (unechoed) characters in input buffer.
1056	.TOTTC	Returns total number of characters in monitor's input buffer.
1057	.TOTOC	Returns total number of characters in monitor's output buffer.
1060	.TOLNB	Returns length of terminal form/page.
1061	.TOLNC	Returns number of lines remaining in page.
1062	.TOSSZ	Returns stop size (number of lines to output) for automatic CTRL/S.
1063	.TOSTC	Returns page stop counter (number of lines remaining on page).
1064-1066		Reserved for use by DIGITAL.
1067	.TOSTO	Specifies that output will stop after the number of lines specified for .TOSSZ.
1070	.TOSST	Does not reset page stop counters after CTRL/S and CTRL/Q.
1071	.TOSBL	Sounds terminal bell on automatic page stop.
1072	.TOFSP	Provides pseudo-terminals with the sceen-editing facilities of a physical terminal. This is a read-only function.
1073	.TOOFL	Returns offline bit. If 0 is returned in the ac, the terminal exists. This is a read-only function.
1074	.TOECH	Returns echo status. If set, echoing is enabled.

Code	Symbol	Function
1075	.TOAPC	Returns asynchronous port characteristics. This is a read-only function.

Code	Symbol	Meaning
0.	.TOUNK	Unknown.
1	.TOHWD	Hard-wired.
2	.TODSD	Dataset line.
3-4		Reserved
5	.TOADL	Auto-dial.
6		Reserved.
7	.TONRT	NRTSER line.
10	.TOLAT	LAT line.
11	.TOCTM	CTERM line.

1076	.TOUNP	Enables unpause character; continues output after CTRL/S. (The enabled character is interpreted, on input, like CTRL/Q.)
1077	.TOESC	Enables ESCape character (behaves like ESC key). In other words, the enabled character is interpreted, on input, as the ESCape character, (ASCII character 033).
1100	.TOSWI	Enables two-character switch sequence.
1101	.TO8BT	Enables 8-bit terminal I/O.
1102	.TO8BI	Enables 8-bit I/O mode on a terminal.
1103	.TOQOT	Enables the terminal quote (^V) character. .TTQOT, when combined with any character, behaves as a single character. A ^V-character combination is deleted by a single rubout, and echoes as one character. ^V suppresses special action on the next character you type. The character is echoed without being processed.
1104	.TOMXT	Returns maximum idle time before an automatic disconnect.
1105	.TOADT	Returns time remaining before automatic disconnect.
1106	.TOCLE	Enables command-level echoing.

Normal Return

The monitor performs the function.

TRMOP. [CALLI 116]

Error Return

One of the following error codes is returned in the **ac**:

Code	Symbol	Error
Ø	TOILF%	Illegal function code.
1	TOPRC%	Not privileged.
2	TORGB%	Illegal range.
3	TOADB%	Illegal argument list address or length.
4	TOIMP%	Line is not a dataset line.
5	TODIL%	Error in dialing routine.
6	TOTNA%	Terminal not available.
7	TONBM%	Terminal is not in break set mode. You must set IO.ABS with the OPEN UUO before you can define and enable break characters.
1Ø	TONIB%	Illegal byte size specified.
11	TONET%	Not a network-based terminal.

TRPSET [CALLI 25]

Function

Prevents jobs other than the calling job from running. You can (if you have the JP.TRP privilege) use this call to guarantee fast response to realtime interrupts.

For a complete discussion of realtime traps and related programming practices, see Chapter 9, Volume 1.

Calling Sequence

```
        MOVE    ac,[XWD len,addr]
        TRPSET  ac,
          error return
        normal return
        . . .
addr:     JSR   address
          BLKI  address
```

where: len is the length of the argument list.

addr is the address of the argument list.

address is the address of a location to be patched to trap directly to your program. This address must be in the range 40 to 57 (octal).

Normal Return

The monitor has suspended execution of other jobs.

Error Return

The error return occurs if the TRPSET call is not implemented, or if your job is not privileged.

Related Calls

HPQ, LOCK, RTTRP, UJEN

TSK. [CALLI 177]

Function

Performs miscellaneous functions for network nodes. This monitor call can be used by applications that wish to perform non-blocking connects and disconnects. Also, it can be used by applications translating ANF-10 protocol into another protocol. These applications usually require more control over the connect message than that provided by the standard LOOKUP/ENTER sequence.

The TSK. monitor call is an alternative to using the LOOKUP/ENTER method for opening/defining network links. Once the link enters the run state (.TKSOK), the normal OUT and IN monitor calls can be used to send or receive data over the network link. The TSK device cannot be designated as an MPX-controlled device, but asynchronous I/O can be performed.

Refer to Chapter 5 for more information about using the TSK. monitor call.

Calling Sequence

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

addr:   EXP     func-code       ;.TKAFN
        EXP     channo          ;.TKACH
        EXP     arg1            ;.TKAA1
        EXP     arg2            ;.TKAA2
        EXP     arg3            ;.TKAA3
```

where: **function** is one of the function codes listed below.

channo is the I/O channel number on which the device TSK has been opened.

Each argument is an argument for the specified function code.

Most arguments will be pointers to Network Process Descriptors (NPDs), having the following format:

```
XWD  length,addr
```

where: **length** is the length of the NPD (must be at least 3).

addr is the location of the NPD.

Associated with each task link are two processes: the local process and remote process. The processes are named by the NPD. The format of the NPD is:

Offset	Symbol	Meaning
0	.TKNND	Node number (-1 implies any node, but is not valid when used in the remote passive NPD).
1	.TKNLN	Length of ASCII process name that follows (number of characters).

Offset	Symbol	Meaning
2	.TKNPN	First word of the ASCII process name.

The following lists the function codes for TSK.:

Code	Symbol	Meaning
1	.TKFRS	Returns the state of the link in **arg1**. The possible states are:

Code	State	Meaning
Ø	.TKSID	The link is idle. The call destroys the contents of **arg2** and stores the reason for the disconnect. **arg3** is unchanged.
1	.TKSCI	The link is waiting for a connect initiate message. It returns the local NPD in the area pointed to by **arg2**. It returns the remote NPD in the area pointed to by **arg3**.
2	.TKSCC	The link is waiting for a connect confirmation message. It returns the local NPD at the location specified in **arg2** and the remote NPD at the location specified in **arg3**.
3	.TKSOK	The link is operational. It returns the local NPD at the location specified in **arg2** and the remote NPD at the location specified in **arg3**.
4	.TKSDC	The link is waiting for a disconnect confirmation message. It returns the local NPD at the location specified in **arg2** and the remote NPD at the location specified in **arg3**.

You may include a pointer (len,,addr) to the local NPD in arg2. You can include the pointer to the remote NPD in arg3. These words, however, are optional.

2 .TKFEP Enters the link into the passive state. The link must be in the .TKSID state. The monitor reads and stores the local and remote NPDs pointed to by arg2 and arg3.

If, at a later time, the monitor receives a Connect Initiate message that "matches" the remote NPD, the following occurs:

- The monitor deletes the remote NPD.
- The monitor builds a new remote NPD from the information given in the connect message. The job can read the new NPD by using the .TKFRS function to determine the process that initiated the connection.

Code	Symbol	Meaning
		o The monitor enters the link into the .TKSOK state. o The monitor issues a device on-line interrupt to the job if the job enabled this condition using the PSI system.
3	.TKFEA	Enters the link into the active state. Before issuing this function the link must be in the .TKSID state. All other states cause an error code (TKILS%) to be returned. When this code is issued, the monitor reads the local NPD pointed to by **arg1** and the remote NPD pointed to by **arg2**. It then sends a Connect Initiate request to the node/task specified in the remote NPD. It puts the link into the TKSCC state and takes the normal return. The link remains in the TKSCC state until a Connect Confirm or Disconnect function is issued. If a Connect Confirm is issued, the monitor discards the remote NPD pointed to by **arg2**. It builds a new remote NPD using the information in the Connect Confirm message (so that it can be read by a .TKFRS function). The link is placed in the .TKSOK state and the controlling job is given a device on-line interrupt (if the condition was enabled using the PSI system). If a Disconnect function is issued, the monitor discards both the local and remote NPD specifications. It places the link into the .TKSID (idle) state and gives the controlling job a device off-line interrupt (if the job enabled this condition using the PSI system).
4	.TKFEI	Enters the link into the idle state. This function is illegal for those tasks in .TKSDC or .TKSCC states and is a no-op for those already in the idle state (.TKSID). The monitor performs the following for those links in .TKSCI and .TKSOK states: .TKSCI Both NPDs are released. The link state is set to .TKSID. .TKSOK A Disconnect Initiate is sent. The link state is set to .TKSDC. When Disconnect Confirmed message is issued at a later time, the monitor frees both NPDs, sets the link state to .TKSID, and issues a "device off-line" interrupt.

Code	Symbol	Meaning
5	.TKFWT	Puts the link into the wait state. If the link is in either the .TKSID or .TKSOK state, the monitor takes the normal return immediately. The monitor performs the following for those links in the other states:

State	Function
.TKSCI	waits for a transition to the .TKSOK state and then returns.
.TKSCC	waits for a transition to either the .TKSOK or .TKSID states, then returns.
.TKSDC	waits for a transition to .TKSID and then returns.

6	.TKFOT	Performs output with control of message disassembly. This function is valid only for links in the .TKSOK state. This function performs an OUT monitor call on the specified channel. If the OUT is successful, the contents of the buffer will be sent without an EOF bit. If unsuccessful, the monitor places error code TKUDW% in the ac and returns the device status word in argl.
7	.TKFIN	Performs input with message reassembly. This function is valid only for those links in the .TKSOK state. It performs an IN monitor call on the specified channel. If the IN is successful (non-skip return), and UU.DMR was not set on the OPEN, the monitor reads the message as one entire buffer and takes a skip return. If UU.DMR was set, the message is read without reassembly. If the IN fails, the monitor places error code TKUDW% in the ac and stores the device status word in ac+1.
10	.TKFRX	Returns the status of the link in argl (see .TKFRS for a list of codes) and the "segment size," or the maximum message size, in arg2. Note that the segment size is only returned if the link is in "OK" state (.TKSOK).

Normal Return

The specified function has been performed.

TSK. [CALLI 177]

Error Return

One of the following error codes is returned in the ac.

Code	Symbol	Meaning
1	TKTNL%	TSKSER not loaded (intertask communication is not supported).
2	TKATS%	Argument list was too short.
3	TKUNP%	Insufficient privileges.
4	TKILF%	Illegal function.
5	TKILC%	Illegal channel (not a TSK device or channel not open).
6	TKILN%	Illegal NPD.
7	TKNTS%	NPD too short.
10	TKILS%	Function is illegal in this state.
11	TKNFC%	Not enough monitor free-core to perform this function.
12	TKNFL%	No free links. (NETLAT is full.)
13	TKNXN%	Attempt to connect to a non-existent node.
14	TKUDW%	IN or OUT UUO (.TKFOT or .TKFIN) did not skip.

Related Calls

NODE.

TTCALL [OPCODE Ø51]

Function

Passes the monitor a code for an extended set of calls; these calls perform terminal functions and are usually called TTCALLs.

Each defined TTCALL code also has a symbolic name; the TTCALLs are discussed in alphabetical order by their symbolic names in this section. For example, TTCALL 1, has the symbolic name OUTCHR; its function is discussed under the name OUTCHR in this section.

The TTCALLs and their symbolic names are:

Symbol	TTCALL Function
INCHRW	[TTCALL Ø,]
OUTCHR	[TTCALL 1,]
INCHRS	[TTCALL 2,]
OUTSTR	[TTCALL 3,]
INCHWL	[TTCALL 4,]
INCHSL	[TTCALL 5,]
GETLCH	[TTCALL 6,]
SETLCH	[TTCALL 7,]
RESCAN	[TTCALL 1Ø,]
CLRBFI	[TTCALL 11,]
CLRBFO	[TTCALL 12,]
SKPINC	[TTCALL 13,]
SKPINL	[TTCALL 14,]
IONEOU	[TTCALL 15,]

Note that TTCALL operations are performed only on physical terminals, not on a device with the logical name TTY.

UGETF [OPCODE Ø73]

Function

Returns the block number of the next free block on a DECtape; the UGETF call is a no-op for other devices. Use FILOP. to perform UGETF for an extended I/O channel.

Calling Sequence

```
        UGETF   channo,addr
        return
        . . .
addr:   BLOCK   1
```

where: channo is the channel number of an initialized device.

addr is the address of the location where the monitor will return a block number at addr.

Return

The block number of the next free block is returned at addr. If this call precedes an ENTER, the inter-block spacing factor used for output is reduced to that used for .SAV files. This function is used to reduce the number of times the tape must reverse direction when reading a large file.

Related Calls

FILOP.

UJEN [OPCODE 1ØØ]

Function

Dismisses a realtime interrupt from a user-supplied service routine, if such a routine is in progress.

Calling Sequence

```
UJEN
return
```

Return

The monitor restores all accumulators and executes the instruction

```
JEN @counter
```

where: **counter** is the address of the program counter stored by a JSR instruction when the interrupt occurred.

Note that you can dismiss a user-mode interrupt with a JRST 12, instruction.

Related Calls

RTTRP, TRPSET

UNLOK. [CALLI 120]

Function

Unlocks one or both segments for the current job. Your job can also be unlocked when the monitor implicitly executes a RESET for your program. This occurs in any of the following cases:

- Your program executes a RUN monitor call.
- You issue any of the monitor commands that invoke a program.

Calling Sequence

```
MOVE   ac,[XWD high,low]
UNLOK. ac,
  error return
normal return
```

where: high can be Ø or 1. If set, the high segment is unlocked. If clear, the status of the high segment remains the same.

low can be Ø or 1. If set, the low segment is unlocked. If clear, the status of the low segment remains the same. If you set both high and low, both segments will be unlocked.

A high segment shared by several jobs cannot be unlocked unless the SN%LOK bit is off for all those jobs. This bit is bit 5 in GETTAB table 14, .GTSGN. This bit will be on for each job that issued the LOCK monitor call for the high segment, but has not issued a subsequent UNLOK. call for the high segment.

Normal Return

The specified segments are unlocked and become eligible for swapping. Any existing meter points (set by the METER. monitor call) are cleared, and any real-time devices are reset. CORMAX is changed to show the newly available pages, if any.

Error Return

The error return occurs only if the UNLOK. monitor call is not implemented on your system. You must use either a RESET or an EXIT monitor call instead.

Related Calls

LOCK

USETI [OPCODE Ø74]

Function

Specifies a block on disk or DECtape to be read, written, or updated. This function can also be performed by SUSET. and FILOP calls. (Use FILOP. to perform USETI on an extended I/O channel.)

The monitor call sequence for reading a file starting at a specific block is listed below:

```
LOOKUP
USETI
INPUT
```

The monitor call sequence for writing a file starting at a specific block is shown below:

```
ENTER
USETO
OUTPUT
```

The monitor call sequence for updating a file is:

```
LOOKUP
ENTER
USETO      ;or USETI
OUTPUT     ;or INPUT
```

If your job is privileged (that is, running with the JACCT bit set or running under [1,2]) and your program does not perform an ENTER before a USETO or a LOOKUP before a USETI, the monitor performs super I/O. If your job is not privileged and your program does not perform an ENTER before a USETO or a LOOKUP before a USETI, the monitor sets IO.BKT in the I/O status word.

The OPEN-ENTER-USETI sequence does not perform super I/O. It returns an IO.IMP error. Likewise, an IO.IMP error results from OPEN-LOOKUP-USETO. Refer to Chapter 11, Volume 1, for more information about using file positioning calls.

Calling Sequence

```
USETI   channo,n
return
```

Where: channo is the channel number for an initialized device.

n is the number of the block to be used for I/O.

For DECtape, the block number is relative to the beginning of the tape.

For disk, n is a block number, when the file is open. When the call is not preceded by a LOOKUP, n is the address of a word where the block number is stored. This is a super-USETI call, because this method allows you to specify a block number, greater than 18 bits, relative to the beginning of the structure.

USETI [OPCODE Ø74]

The action of the USETI call on disk devices is determined by the value of n as follows:

Value	Meaning
-n, n = 2 to 1Ø (octal)	The nth extended RIB is read.
-1	IO.EOF is set in the I/O status word, causing an end-of-file on the next INPUT. On the next OUTPUT, the data is appended to the file.
Ø	The prime RIB is read on the next INPUT.
1 to file size	The block specified is read on next INPUT. If the file size is greater than or equal to 777770, it is recommended that you use FILOP. function .FOUSI.
file size to 777770	The IO.EOF bit is set in the I/O status word, causing an end-of-file on the next INPUT.

Related Calls

FILOP., SUSET., USETO

Common Errors

- Not synchronizing I/O with USETI or USETO.
- Not initializing a device on channo.

USETO [OPCODE Ø75]

Function

Selects a block on disk or DECtape to be written by an OUT monitor call. This function can be performed by SUSET. and FILOP. (Use FILOP. to perform USETO on an extended I/O channel.) Refer to the USETI UUO and Chapter 11, Volume 1, for more information.

Calling Sequence

```
USETO   channo,n
return
```

Where: channo is the channel number for an initialized device.

n is the number of the I/O block.

For DECtape, n is the block number relative to the beginning of the tape.

For disk, n is a block number if a previous ENTER has been used to open a file. Otherwise, n is the address of a word that contains the block number relative to the beginning of the structure or unit (super-USETO), allowing you to specify a block number greater than 18 bits.

The action of the USETO call for disk is determined by the value of n as follows:

Value	Meaning
-n for n = 2 to 1Ø (octal)	Equivalent To a USETO 777776 to 777770; you may not write to the RIBs of a file.
-1	The most recently input or output block is re-written on the next OUTPUT.
Ø	The IO.BKT is set in the file status word.
1 to file size	The specified block is written on the next OUTPUT. If the file size is greater than or equal to 777777, it is recommended that you use FILOP. function .FOUSO.
file size to 777776	The monitor allocates all blocks from the block after file size to the block before the one specified. Each block allocated is written with zeros. The block specified is the next block written in the next OUTPUT.

USETO [OPCODE Ø75]

Related Calls

FILOP., SUSET., USETI

Common Errors

- Not synchronizing I/O with USETI or USETO.
- Not initializing a device on channo.

UTPCLR [CALLI 13]

Function

Clears a DECtape directory. Use FILOP. to perform UTPCLR on an extended I/O channel. The UTPCLR monitor call is a no-op for other devices.

Calling Sequence

```
UTPCLR  channo,
return
```

where: **channo** is the channel number for an initialized device.

Normal Return

The monitor clears the directory by clearing the first 83 words (except those 7-bit bytes describing blocks 0, 1, 2, 100, and 1102 through 1105 octal).

Common Errors

- I/O to unassigned channel at user PC xxxxxx.
- Forgetting to place the channel in the ac or forgetting the comma after **channo**.
- Not initializing a DECtape on **channo**.

UTRP. [CALLI 174]

Function

Sets or reads user trap instructions. This UUO allows a user to handle non-zero section LUUOs, arithmetic overflows, or pushdown list overflows by depositing instructions in locations 420, 421 and 422 in the UPMP. Usually these instructions are calls to user-supplied subroutines.

Calling Sequence

```
        MOVE   ac,[XWD fcncode,addr]
        UTRP. ac,
          error return
        normal return
        . . .
addr:   length
        trapno
        trapinstr
        . . .
        trapno
        trapinstr
```

where: **fcncode** is one of the function codes described below.

addr is the address of the argument list.

length is the number of words in the argument list.

trapno is the number of a trap. Trap numbers are listed below.

trapinstr is the instruction to call the trap routine.

The function codes and their meanings are:

Code	Symbol	Meaning
0	.UTRED	Reads the contents of the trap location. A zero opcode cannot be used as a trap instruction, but a zero fullword here will restore monitor handling of the specific condition.
1	.UTSET	Sets the contents of the trap location.

The trap numbers are:

Code	Symbol	Meaning
0	.UTLUU	Stores the contents of the 4 word LUUO trap block in the UPT.
1	.UTAOF	Arithmetic overflow trap (location 421 in the UPMP).
2	.UTPOV	Pushdown list overflow trap (location 422 in the UPMP).

Normal Return

The specified traps are cleared or set.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
1	UTIAD%	Illegal address.
2	UTUKF%	Unknown function.
3	UTITN%	Illegal trap number.
4	UTIUT%	Illegal user trap instruction.

Related Calls

ABRENB, .JBINT trapping, PSI system

WAIT [CALLI 1Ø]

Function

Causes program execution to wait until all data transmissions on a given channel are completed. Use FILOP. to perform WAIT on an extended I/O channel.

Calling Sequence

```
WAIT    channo,
return
```

where: channo is the channel number for an initialized device.

Normal Return

The monitor stops your program's execution until transmissions on the channel are completed.

Common Errors

- Using WAIT on a tape which is spacing (see MTWAT.).
- Not initializing a device on channo.
- Omitting the comma after channo.

WAKE [CALLI 73]

Function

Sets the wake bit for a specified job.

Calling Sequence

```
MOVEI   ac,jobno
WAKE    ac,
  error return
normal return
```

Where: jobno is the number of a logged-in job (use -1 for the current job).

You can design a real-time process control job to run other process control jobs when specific alarm conditions occur. WAKE can be called from an RTTRP job running at interrupt level; this allows the real-time job to wake its background quickly when necessary. See the RTTRP monitor call for restrictions on accumulators when calling from the interrupt level.

If your job does not have the required privileges, the error return occurs and the monitor clears the ac. A JACCT or [1,2] job may WAKE any job. If any condition enabled in the last HIBER call occurs, the wake bit for the job is set. At the next HIBER call, the wake bit is cleared and the monitor returns at the normal return immediately. The wake bit prevents the job from oversleeping a wake condition.

Normal Return

The specified job is awake and resumes execution at the normal return for the HIBER call that made the job dormant.

Error Return

Your job did not have the required privileges. The ac is cleared.

Related Calls

Refer to the HIBER monitor call.

WHERE [CALLI 63]

Function

Returns the node number for a device.

Calling Sequence

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

where: **device** is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel.

udx is the Universal Device Index for a device.

When your program specifies OPR as the device, the monitor returns the node number at which your job is logically located. Refer to the LOCATE command description in the Commands Manual.

When your program specifies CTY as the device, the monitor returns the node number of your job's host system.

When your program specifies TTY as the device, the monitor returns the node number to which your terminal is physically located.

Normal Return

The monitor returns the status flags for the node and the node number for the given device in the **ac**. The format of the returned word is:

Bits	Symbol	Meaning
Ø-12		Reserved for use by DIGITAL.
13	RM.SDU	Dial-up node.
14-17	RM.SUP	Status of the node:

Value	Symbol	Status
1	.RMSUN	Not in contact with central site.
2	.RMSUD	Down.
4	.RMSUG	Loading.
1Ø	.RMSUL	Loaded.

18-35		Node number for device.

Error Return

A nonexistent device was specified.

Related Calls

LOCATE, NODE., NETOP.

CHAPTER 23

GETTAB TABLES

The TOPS-10 monitor maintains tables that contain system and job information. Values from some of these tables can be retrieved by a program using the GETTAB monitor call.

23.1 HOW TO USE GETTAB TABLES

The calling sequence for GETTAB is:

```
MOVE    ac,[XWD index,table]
GETTAB  ac,
  error return
normal return
```

where: ac is an accumulator.

index is the index into the table, which may be a job number, a table item number, a high-segment number, or a class code, depending on the table's organization.

table is the symbolic name of the desired table.

For those tables indexed by table item numbers, it is easier and more reliable to use the calling sequence:

```
MOVE   ac,[item]
GETTAB ac,
  error return
normal return
```

where: item is the symbolic name of the desired item as defined in UUOSYM. Using this calling sequence eliminates the need to name both the table and the item desired. In the following table descriptions, items marked by a * are described in more detail at the end of the table.

For tables indexed by job numbers, use the index -1 to specify the current job; use the index -2 to specify the job's current high segment.

23.2 HOW TO USE GETTAB SUBTABLES

Some GETTAB tables have subtables. In this section, each subtable is described after the description of the table containing it. Each subtable description includes the calling sequence for accessing items in the subtable.

Each subtable has a single entry in the GETTAB table. This entry returns <length>B8+offset, where length is the length of the subtable and offset is the first entry in the GETTAB table that corresponds to the subtable.

Contents	Offset	GETTAB Table
	Ø	entry Ø
	1	1
5B8+1Ø	2	subtable 1 pointers
	3	entry 3
4B8+15	4	subtable 2 pointers
	5	entry 5
	6	entry 6
	7	entry 7
	1Ø	subtable 1 (entry Ø)
	11	
subtable 1	12	
	13	s
	14	c
	15	
	16	
subtable 2	17	
	2Ø	

An example of the use of GETTAB subtables follows below:

```
          TITLE   GTRSP - Example of CPU response sub-GETTAB
          SUBTTL  Hanley A. Strappman 13-June-80 /HAS

          SEARCH  UUOSYM              ;Use standard symbols

T1=1                                  ;Accumulators
T2=T1+1
P1=5
P=17

PLN=100                               ;Length of program stack
          ARRAY   PDL[PLN]            ;The program stack

NCPUS==6                              ;How many CPUs this program allows

GTRSP:    JFCL                        ;In case of CCL RUN
          RESET                       ;Reset the world
          MOVE    P,[IOWD PLN,PDL]    ;Set up stack
          MOVSI   P1,-NCPUS           ;For loop control
LOOP:     HRRZ    T1,P1               ;Get number of next CPU
          PUSHJ   P,GETIT             ;Get the data
            JRST  NEXT                ;No such table

;Insert here the code to process the data, then

NEXT:     AOBJN   P1,LOOP             ;Go on to next CPU
          EXIT                        ;Done

;Subroutine to return in T2 the number of TTY input-to-input
;UUO responses for the CPU specified by T1.

GETIT:    LSH     T1,1                ;Variables table numbers go up by twos
                                      ; (for example, .GTC1V=.GTC0V+2)
          MOVE    T2,[%CCRSP]         ;CPU0's subtable pointer
          ADD     T2,T1               ;CPUn's subtable pointer
          GETTAB  T2,                 ;Get base index of subtable
            POPJ  P,                  ; ??? Must be an old monitor
          JUMPE   T2,CPOPJ            ;No such subtable
          ADDI    T2,%CVRNI           ;Add entry offset to subtable base

          HRL     T2,T2               ;This becomes the item number
          HRRI    T2,.GTCOV           ;Right half is table number which
          ADD     T2,T1               ; equals <CPU0 index> + 2*<CPU number>
          GETTAB  T2,                 ;Finally get the subtable entry
            POPJ  P,                  ; ???Must be an old monitor
          AOS     (P)                 ;Skip return means good data in T2
CPOPJ:    POPJ    P,                  ;All done
          END     GTRSP
```

23.3 ADDING ITEMS TO THE MONITOR'S GETTAB TABLES

System programmers can add words (items) to the monitor's GETTAB tables. The items added must have negative indexes, and must be added at the top of the table in the order -n, -(n-1), -(n-2), ... -2, -1. When the monitor is assembled, the range of valid indexes for each table must begin with the lowest (most negative) index, and proceed to the highest index.

It is good programming practice to use a .UNV file containing symbols for these items for use in programs. This usage is similar to that of searching UUOSYM.UNV.

23.4 ADDING NEW GETTAB TABLES TO THE MONITOR

System programmers can add completely new GETTAB tables to the monitor. These tables must have negative table numbers, and must be added at the beginning of COMMON.MAC in the order -n, -(n-1), -(n-2), ... -2, -1. When the monitor is assembled, the range of valid table numbers must begin with the lowest (most negative) table number, and proceed to the highest table number. Within these added tables, items must be indexed sequentially.

For example, the system programmer can add a new table with the number -1. This table must be added to the source code in the monitor module UUOCON. The items in this table could begin and end with negative indexes; for example, the indexes could begin with -14 and end with -1. The items could begin with a negative index and end with a positive index; for example, the indexes could begin with -10 and end with 27. Or the items could begin and end with nonnegative indexes; for example the indexes could begin with 0 and end with 15.

23.5 ALPHABETIC LISTING

Because GETTAB tables are often referred to by symbolic name, the following list of GETTAB tables is provided in alphabetical order:

Symbol	Table No.	Description
.GTABS	111	Address Break Word
.GTADR	1	Job Relocation and Protection
.GTC0C	55	CPU0 CPU Data Block Constants
.GTC0V	56	CPU0 CPU Data Block Variables
.GTC1C	57	CPU1 CPU Data Block Constants
.GTC1V	60	CPU1 CPU Data Block Variables
.GTC2C	61	CPU2 CPU Data Block Constants
.GTC2V	62	CPU2 CPU Data Block Variables
.GTC3C	63	CPU3 CPU Data Block Constants
.GTC3V	64	CPU3 CPU Data Block Variables
.GTC4C	65	CPU4 CPU Data Block Constants
.GTC4V	66	CPU4 CPU Data Block Variables
.GTC5C	67	CPU5 CPU Data Block Constants
.GTC5V	70	CPU5 CPU Data Block Variables
.GTCAP	153	Job Capability Word
.GTCM2	43	SET Command Names
.GTCMP	112	Obsolete
.GTCMT	75	SET TTY Command Names
.GTCMW	101	SET WATCH Command Names
.GTCNF	11	System Configuration Table
.GTCNO	33	Charge Number
.GTCOJ	122	Obsolete
.GTCOM	30	Monitor Command Names
.GTCOR	27	Core Table (Obsolete)
.GTCQP	121	Scheduler Class Quota
.GTCRS	44	Hardware Status After Crash
.GTCRT	123	Class Runtime
.GTCTX	175	Context Table
.GTCVL	102	Current Page Limits
.GTDBS	21	Disk Block Seconds (Obsolete)

Symbol	Table No.	Description
.GTDCD	160	CONI/DATAI To Device Status Block
.GTDCF	116	Obsolete
.GTDCN	164	SET DEFAULT Command Argument(s)
.GTDDB	200	I/O Wait DDB
.GTDEV	24	Segment Device or Structure
.GTDFL	140	User Defaults for Job
.GTDVL	110	Pointer to Logical Name Table
.GTEBR	132	EBOX Jiffy Remainder
.GTEBT	131	KL10 EBOX Time
.GTEDN	72	Ersatz Device Names
.GTENQ	127	ENQ/DEQ Statistics
.GTEQJ	163	ENQ/DEQ Queue Header
.GTETH	202	Ethernet Information
.GTFET	71	Feature Test Settings
.GTGTB	155	GETTAB Immediate
.GTIDX	154	Range of GETTAB Tables
.GTIMI	176	Job Page Count
.GTIMO	177	Swapped-Out Page Count
.GTIPA	104	IPCF Statistics
.GTIPC	77	IPCF Miscellaneous Data
.GTIPI	106	PID for [SYSTEM]INFO
.GTIPP	105	IPCF Pointers and Counts
.GTIPQ	107	IPCF Flags and Quotas
.GTISC	45	Swap-In Scan Tables
.GTJLT	130	LOGIN Time for Job
.GTJTC	120	Job Type and Scheduler Class
.GTKCT	5	Job Kilo-Core Ticks
.GTLIM	40	Time Limit and Batch Status
.GTLOC	26	Remote Station Number
.GTLBS	165	Large Buffer Size
.GTLVD	16	Level D Disk Parameters
.GTMBR	134	MBOX Jiffy Remainder
.GTMBT	133	KL10 MBOX Time
.GTMVL	103	Maximum Page Limits
.GTNDB	161	Byte Pointers to Node Data Block
.GTNM1	31	User Name (first 6 characters)
.GTNM2	32	User Name (last 6 characters)
.GTNSW	12	Nonswapping Data Table
.GTNSP	172	DECnet session control (Obsolete)
.GTNTP	141	Network Performance Data
.GTOBI	157	WTO and Batch Data
.GTODP	15	ONCE-only Disk Parameters
.GTOSC	46	Swap-Out Scan Tables
.GTPC	152	Wait DDB and User PC
.GTPDB	162	Job PDB Word
.GTPID	76	Process Communication ID (IPCF)
.GTPPN	2	Job's PPN
.GTPRG	3	User Program Name
.GTPRV	6	Job Privilege Flags
.GTPTR	166	Program To Run
.GTQJB	42	Scheduling Queue Table (Obsolete)
.GTQQQ	41	Scheduling Queue Headers (Obsolete)
.GTRCT	17	Disk Blocks Read
.GTRDI	136	Program Run Directory
.GTRDV	135	Program Run Device
.GTRFN	137	Program Run File name
.GTRS0	145	First SFD in Run Path
.GTRS1	146	Second SFD in Run Path
.GTRS2	147	Third SFD in Run Path
.GTRS3	150	Fourth SFD in Run Path
.GTRS4	151	Fifth SFD in Run Path

Symbol	Table No.	Description
.GTRSP	5Ø	Response Counter Table
.GTRTD	37	Realtime Status Words
.GTSCN	73	Scanner Data
.GTSDT	13	Swapping Data Table
.GTSGN	14	High Segment Parameters
.GTSG2	2Ø3	High Segment Section Number
.GTSID	126	Special PID Table
.GTSLF	23	GETTAB Table Data
.GTSNA	74	Last SEND ALL in 9-bit
.GTSPA	142	Scheduler Performance Data
.GTSPL	36	Spooling Control Flags
.GTSPS	54	Status of Subsequent Processors
.GTSQ	125	Obsolete
.GTSQH	124	Obsolete
.GTSSC	47	Scheduler Scan Tables
.GTSST	115	Scheduler Statistics
.GTST2	117	Second Job Status Word
.GTSTS	Ø	Job Status Word
.GTSWP	7	Job Swapping Parameters
.GTSYS	51	System-Wide Data
.GTTDB	22	Last Allocation (Obsolete)
.GTTIM	4	User Runtime
.GTTMP	34	TMPCOR Pointers (Obsolete)
.GTTNM	156	Terminal Type Names
.GTTRQ	53	Time in Run Queue
.GTTTY	1Ø	Job's Controlling Terminal
.GTUPM	1ØØ	Physical Page Number of UPMP
.GTUUC	144	Monitor Calls Executed
.GTVIR	2Ø1	Job's Virtual Size
.GTVKS	143	Virtual Kilo-Core Ticks
.GTVM	113	Virtual Memory Data
.GTVRT	114	Paging Rage
.GTWCH	35	Watch Bits
.GTWCT	2Ø	Disk Blocks Written
.GTWHY	52	Operator Reload Comments
.GTWSN	25	Names of Wait States

23.6 TOPS-1Ø GETTAB TABLES

The remainder of this chapter describes the TOPS-1Ø GETTAB tables. For each table, the contents of the table, the indexing scheme, the GETTAB calling sequence, and a word map for the table are described. The description of each GETTAB table also includes the associated monitor table. The monitor tables are described in the TOPS-1Ø Monitor Tables descriptions, available in the TOPS-1Ø Software Notebook Set.

.GTSTS - Job Status Word
GETTAB Table Ø

Contents One word for each job running on the system, giving the status word for the job. (There is a second job status word in GETTAB table .GTST2, Number 117.) The bits in the job status word may vary from monitor to monitor; therefore you should not reference .GTSTS in a program that is monitor-independent.

Indexed by Job number or segment number.

Monitor table JBTSTS

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTSTS]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job (use -1 for the current job) or a high-segment number (use -2 for the current high segment).

Job status

The only documented bit in this word is JS.XO (bit 25), which indicates a program that is execute-only. Refer to the Monitor Tables descriptions for more information about the job status bits.

.GTADR - Job Relocation and Protection
GETTAB Table 1

Contents

One word for each job running on the system, giving the relocation addresses and protection for the job. The bits in this word may vary from monitor to monitor; therefore you should not reference .GTADR in a program that is monitor-independent.

Indexed by

Job number or segment number.

Monitor Table

JBTADR

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTADR]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job (use -1 for the current job) or a high-segment number (use -2 for the current high segment).

Job relocation and protection

.GTPPN - Project-programmer Number
GETTAB Table 2

Contents One word for each job running on the system, giving the project-programmer number (PPN) for the job.

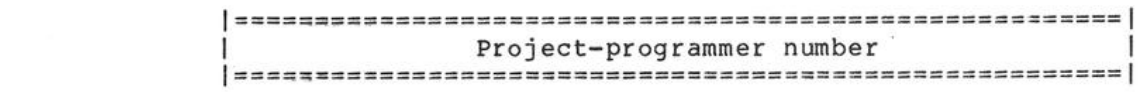
Indexed by Job number or segment number.

Monitor Table JBTPPN

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTPPN]
        GETTAB  ac,
          error return
        normal return
```

where **jobno** is the number of a logged-in job (use -1 for the current job) or a high-segment number (use -2 for the current high segment).



where project-programmer number is the job's PPN or the segment owner's PPN.

If the high segment's file is in an SFD, this word is returned as Ø,,path-pointer, where path-pointer is a pointer to a path block in monitor memory. PEEK privileges are required to read the monitor's path block.

.GTPRG - User Program Name
GETTAB Table 3

Contents — One word for each job running on the system, giving the SIXBIT name of the current user program.

Indexed by — Job number or segment number.

Monitor Table — JBTNAM

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTPRG]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job (use -1 for the current job) or a high-segment number (use -2 for the current high segment).

SIXBIT program name

For jobs, this is the program name that can be set by SETNAM. The default is the file name of the runnable program (.EXE file). For segments, this is the segment name. If the segment is not sharable, Ø is returned. If sharable, it is the file name of the .EXE file.

.GTTIM - User Runtime
GETTAB Table 4

Contents — One word for each job running on the system, giving the total runtime (in jiffies) for the job.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTTIM]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job (use -1 for the current job).

Total runtime in ticks

.GTKCT - Job Kilo-Core Ticks
GETTAB Table 5

Contents

One word for each job running on the system, giving the total kilo-core ticks for the job. This value is equivalent to the product of the number of clock ticks in the job's runtime and the average number of Ks of core used in each tick.

Indexed by

Job number.

Monitor Table

PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTKCT]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Kilo-core ticks

.GTPRV - Job Privilege Flags
GETTAB Table 6

Contents One word for each job running on the system, giving the privilege bits for the job.

Indexed by Job number.

Monitor Table JBTPRV

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTPRV]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Job privilege bits

Privilege bits for each job are as follows:

Bits	Symbol	Privilege
Ø	JP.IPC	IPCF privilege.
1-2	JP.DPR	Highest disk priority for the job (a value in the range Ø to 3).
3	JP.MET	METER. privilege.
4	JP.POK	POKE. privilege.
5	JP.CCC	Privilege to change CPU specification with either a command or a monitor call.
6-9	JP.HPQ	Highest high-priority queue available to the job (a value in the range Ø to 17 octal).
1Ø	JP.NSP	Device unspooling privilege.
11	JP.ENQ	ENQ/DEQ privilege (allows you to use -2 (.EQFGL) in the ENQ. block to set global privileges).
12	JP.ADM	System Administrator privilege.
13	JP.RTT	RTTRP privilege.
14	JP.LCK	LOCK privilege.
15	JP.TRP	TRPSET privilege.
16	JP.SPA	PEEK and SPY privilege for any core.
17	JP.SPM	PEEK and SPY privilege for monitor core.

Bits in the right half are reserved for customer definition.

.GTSWP - Job Swapping Parameters
GETTAB Table 7

Contents

One word for each job running on the system, giving the swapping parameters for the job. The definitions of the bits in this word may vary from monitor release to monitor release; therefore you should not reference .GTSWP in a program that is monitor-independent.

Indexed by

Job number.

Monitor Table

JBTSWP

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTSWP]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Swapping parameters

.GTTTY - Job's Controlling Terminal
GETTAB Table 1Ø

Contents

One word for each job running on the system, giving the controlling terminal for the job. The definitions of the bits in this word may vary from monitor release to monitor release; therefore you should not reference .GTTTY in a program that is monitor-independent.

Indexed by

Job number.

Monitor Table

TTYTAB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTTTY]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Controlling terminal

The right half contains the address of the DDB for the job's controlling TTY. This is always the address of the DDB, even if the job is detached. If Ø, there is no such job number.

.GTCNF - System Configuration Table
GETTAB Table 11

Contents — Data describing the current configuration of the system.

Indexed by — Item number.

Monitor Table — CNFTBL

Calling Sequence

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below. When an item name is followed by a *, that item is more fully described at the end of the table.

Word	Symbol	Map
		|==========|
Ø,,11	%CNFGØ	| System name (1st of 5 ASCIZ words) |
1,,11	%CNFG1	| System name (2nd of 5 ASCIZ words) |
2,,11	%CNFG2	| System name (3rd of 5 ASCIZ words) |
3,,11	%CNFG3	| System name (4th of 5 ASCIZ words) |
4,,11	%CNFG4	| System name (5th of 5 ASCIZ words) |
5,,11	%CNDTØ	| System creation date (1st of 2 ASCIZ words) |
6,,11	%CNDT1	| System creation date (2nd of 2 ASCIZ words) |
7,,11	%CNTAP	| SIXBIT name of system device |
1Ø,,11	%CNTIM	| Encoded time of day in jiffies |
11,,11	%CNDAT	| Encoded date (15-bit binary format) |
12,,11	%CNSIZ	| System memory size (in words) |
13,,11	%CNOPR	| SIXBIT name of operator TTY |
14,,11	%CNDEV	| Start of DDB chain | Reserved |
15,,11	%CNSJN	|-(Max number of high segs) |Max number of current jobs |
16,,11	%CNTWR	| Two-register hardware and software flag |
17,,11	%CNSTS*	| Feature test switches | Switch states |
2Ø,,11	%CNSER	| CPUØ serial number |
21,,11	%CNNSM	| Number of nanoseconds per memory cycle |
22,,11	%CNPTY	| Number of 1st PTY (CTY+1) | Number of PTYs in system |
23,,11	%CNFRE	| Pointer to bit map of core blocks |
		|==========|

24,,11	%CNLOC	Address of low-segment core blocks
25,,11	%CNSTB	Obsolete
26,,11	%CNOPL	Pointer to line data block (LDB) of OPR TTY
27,,11	%CNTTF	Pointer to TTY free chunks
30,,11	%CNTTC	Number of TTY chunks | Address of 1st TTY chunk
31,,11	%CNTTN	Number of free TTY chunks
32,,11	%CNLNS	Pointer to current command TTY
33,,11	%CNLNP	-Number of TTY+PTY+CTY lines,,addr of LINTAB
34,,11	%CNVER	Monitor version number
35,,11	%CNDSC	-Len of dataset ctrl tbl,,addr of ctrl tbl
36,,11	%CNDLS	Obsolete
37,,11	%CNCCI	Obsolete
40,,11	%CNSGT	Ptr to last dormant seg deleted to free a seg number
41,,11	%CNPOK	Last location changed by a POKE. monitor call
42,,11	%CNPUC	Job that made last POKE. | Number of POKE.s made
43,,11	%CNWHY	SIXBIT reason for last reload (operator input)
44,,11	%CNTIC	Number of clock ticks per second
45,,11	%CNPDB	Pointer to process data block (PDB) pointer tables
46,,11	%CNRTC	Resolution of runtime clock (units/sec)
47,,11	%CNCHN	Ptr to channel data block | Reserved
50,,11	%CNLMX	Maximum number of logged-in jobs allowed
51,,11	%CNBMX	Maximum number of batch jobs allowed
52,,11	%CNBMN	Minimum number of jobs reserved to batch
53,,11	%CNDTM	Date/time in universal date/time format
54,,11	%CNLNM	Number of jobs logged in
55,,11	%CNBNM	Number of batch jobs logged in
56,,11	%CNYER	Current year
57,,11	%CNMON	Current month
60,,11	%CNDAY	Current day of the month
61,,11	%CNHOR	Current hour (0-23)
62,,11	%CNMIN	Current minute (0-59)
63,,11	%CNSEC	Current second (0-59)

64,,11	%CNGMT	Offset from Greenwich Mean Time (such that %CNGMT +%CNDTM = GMT)
65,,11	%CNDBG	Debug status
66,,11	%CNFRU	Number of free core blocks in use by monitor
67,,11	%CNTCM	Addr of last TTY chunk
70,,11	%CNCVN	Customer version number
71,,11	%CNDVN	DIGITAL version number
72,,11	%CNDFC	Number of data channels on the system
73,,11	%CNRTD	Number of realtime devices
74,,11	%CNHPQ	Number of high-priority queues
75,,11	%CNLDB	TTY device data blk wrd pointing to line data blk
76,,11	%CNMVO	Maximum vector offset for PISYS.
77,,11	%CNMIP	Maximum priority for PISYS.
100,,11	%CNMER	Offset of MTA err rep word | Address of lst MTA DDB
101,,11	%CNET1	User address of exec AC T1 (for DAEMON)
102,,11	%CNLSD	Length of short device data block
103,,11	%CNLLD	Length of long device data block
104,,11	%CNLDD	Length of disk device data block
105,,11	%CNEXM	Addr in JOBDAT of last Examine or Deposit command
106,,11	%CNST2*	Software configuration flags
107,,11	%CNPIM	Minumum condition in PISYS.
110,,11	%CNPIL	Length of internal PITs
111,,11	%CNPIA	Address of JBTPIA
112,,11	%CNMNT*	Monitor type
113,,11	%CNOCR	Addr of lst CDR DDB | Offset to card count
114,,11	%CNOCP	Addr of lst CDP DDB | Offset to card count
115,,11	%CNPGS	Unit of core allocation (in words)
116,,11	%CNMMX	Maximum allowable CORMAX (total phys mem for all jobs
117,,11	%CNNSC	Number of scheduler classes
120,,11	%CNUTF	Exponential user time factor
121,,11	%CNHSO	Address of start of monitor high segment

122,,11	%CNHSL	Length of monitor high segment
123,,11	%CNNWC	Number of words in core (highest addr of on-line mem)
124,,11	%CNNXM	AOBJN pointer to NXMTAB used to scan for zeros
125,,11	%CNNDB	Addr of 1st network node data block
126,,11	%CNTKB	Offset in MTA KDB of addr of CDB
127,,11	%CNDDC	Offset into TTY DDB of character counts
130,,11	%CNHDL	Potentially hung device list
131,,11	%CNBTX	Address of reload .CCL text for BOOTS
132,,11	%CNTDB	Offset in MTA UDB of addr of DDBs
133,,11	%CNMTK	Addr of 1st MTA KDB in system
134,,11	%CNCPU	Number of CPUs monitor was built for
135,,11	%CNDJB	Byte pointer to jobno in DDB
136,,11	%CNSUP	System uptime
137,,11	%CNBCP	Bootstrap CPU number
140,,11	%CNBCL	Bootstrap CTY line number
141,,11	%CNNCR	Number of CPUs allowed to run
142,,11	%CNMBS	Monitor bootstrap file structure (from BOOTS)
143,,11	%CNMBF	Monitor bootstrap file name
144,,11	%CNMBX	Monitor bootstrap file extension
145,,11	%CNMBD	Monitor bootstrap file directory
146,,11	%CNBPM	Maximum number of SNOOP. breakpoints allowed
147,,11	%CNMXF	First free virtual address about the monitor
150,,11	%CNLVO	Virtual address where LDBs start
151,,11	%CNHXC	Maximum number of FILOP. extended channels
152,,11	%CNVSH	Monitor virtual start address of high segment
153,,11	%CNRST	Universal date/time of last role switch on multiple CPU systems
154,,11	%CNDCH	Offset into LDB of LDBDCH
155,,11	%CNSF1	Monitor bootstrap 1st SFD
156,,11	%CNSF2	Monitor bootstrap 2nd SFD
157,,11	%CNSF3	Monitor bootstrap 3rd SFD
160,,11	%CNSF4	Monitor bootstrap 4th SFD

161,,11	%CNSF5	Monitor bootstrap 5th SFD
162,,11	%CNFLN	TTY number of FRCLIN
163,,11	%CNPNP	Pointer to PTY table
164,,11	%CNCAT	Pointer to network link address table
165,,11	%CNLPD	Length of PDB
166,,11	%CNJPK*	Max. size of JOBPEK transfers
167,,11	%CNDAE*	Previous and current monitor versions
170,,11	%CNHSH	AOBJN pointer to ENQ. HSHTAB
171,,11	%CNACS	Offset to PDB for account string
172,,11	%CNTOP	Pointer to TRMOP. dispatch table
173,,11	%CNSFD	Pointer to JBTSFD
174,,11	%CNCIP	Pointer to CIPWT
175,,11	%CNPRV	Privilege word to be used by privileged jobs (FRCLIN, INITIA)
176,,11	%CNCV1	First word of CTERM version string (8-bit)
177,11	%CNCV2	Second word of CTERM version string (8-bit)
200,11	%CNLHN	Pointer to LAT host node data base
201,,11	%CNIVM	"AND" mask for .GTIMI/.GTIMO/.GTVIR
202,,11	%CNACB	Address of first Allocation Control Block
203,,11	%CNAHB	Address of first Allocation Header Block

The items in the configuration table are defined below:

Item	Symbol	Contains
17	**%CNSTS**	Feature test switch flags and switch state flags are as follows:

Bits	Symbol	Feature or State
0	ST%DSK	Disk system.
1	ST%SWP	Swapping system.
2	ST%LOG	LOGIN system.
3	ST%FTT	Full duplex TTY software.
4	ST%PRV	Privileged features exist.
5	ST%TWR	Software is two-segment (reentrant).
6	ST%CYC	System clock runs at 50 Hz.
9	ST%TDS	Type of disk system: 0 = 4-series 1 = 5-series 2 = spooled disk

Item	Symbol	Contains		
		Bits	Symbol	Feature or State
		10	ST%IND	Independent PPNs on disk.
		11	ST%IMG	Image mode supported on TTYs.
		12	ST%DUL	Dual-processor system.
		13	ST%MRB	Multiple RIBs supported.
		14	ST%HPT	High-precision time accounting supported.
		15	ST%EMO	Monitor overhead excluded from accounting.
		16	ST%RTC	System has realtime clock (DK10).
		17	ST%MBF	System supports FOROTS.
		18-25		Reserved.
		26	ST%NDL	No automatic down-line load of DC72, DC71, and DAS80 series remote stations.
		27	ST%NOP	No operator coverage.
		28	ST%NSP	Device unspooling allowed without privilege.
		29	ST%ASS	System assigning/initializing restricted devices allowed.
		32	ST%NRT	No remote TTYs.
		33	ST%BON	Batch jobs only.
		34	ST%NRL	No remote logging-in.
		35	ST%NLG	No logging-in except operator CTY.
65	%CNDBG	Debugging flags are as follows:		
		Bits	Symbol	Meaning
		0	ST%DBG	System debugging allowed.
		1	ST%RDC	Reload system on DEBUG stopcode.
		2	ST%RJE	Reload system on JOB stopcode.
		3	ST%NAR	No automatic reloading.
		4	ST%CP1	Stop CPU0 on CPU1 halt.
		18	ST%BP0	CPU0 can enter EDDT mode using XCT .C0DDT.
		19	ST%BP1	CPU1 can enter EDDT mode using XCT .C1DDT.
		20	ST%BP2	CPU2 can enter EDDT mode using XCT .C2DDT.
		21	ST%BP3	CPU3 can enter EDDT mode using XCT .C3DDT.
		22	ST%BP4	CPU4 can enter EDDT mode using XCT .C4DDT.
		23	ST%BP5	CPU5 can enter EDDT mode using XCT .C5DDT.
		77B23	ST%BPT	Mask for all CPU breakpoint bits.

Item	Symbol	Contains
1Ø6	%CNST2	Configuration feature flags are as follows:

Bits	Symbol	Feature
6	ST%END	DECnet is running as an Ethernet end node.
7	ST%NPP	Disabled starting primary protocol on DTEs.
8	ST%RCM	Restricted commands monitor.
9	ST%EXA	Extended addressing in effect.
1Ø	ST%D36	Monitor has DECnet Phase III code.
11	ST%KLP	Monitor uses KL-paging.
12	ST%MDA	Mountable device allocation is in effect.
13	ST%LSC	Low segment of monitor is cached.
14	ST%ACV	Account validation.
15	ST%NER	Version 6.Ø3 error reporting.
16	ST%NCS	Scheduler is not a class system scheduler.
17	ST%ITA	Interval timer available.
18	ST%NDN	Network devices have names of the form gggnnu, where ggg is a generic device name (such as TTY), nn is the last two digits of the node number, and u is the unit number.
19	ST%XPI	PI time excluded from runtime.
2Ø	ST%ERT	EBOX/MBOX runtime (KL1Ø only).
21	ST%EXE	.EXE files written by SAVE and SSAVE.
22	ST%NJN	System uses 9-bit job numbers.
23	ST%EER	Extended error reporting.
24	ST%TAP	TAPSER included in monitor.
25	ST%MBE	Massbus error reporting.
26	ST%GAL	GALAXY supported.
27	ST%ENQ	ENQ./DEQ. monitor calls included.
28	ST%SHC	Scheduler is a class type scheduler.
29	ST%NSE	Nonsuperseding ENTER call.
3Ø	ST%MSG	MPX channels supported.
31	ST%PSI	Software interrupt supported.
32	ST%IPC	IPCF supported.
33	ST%VMS	VMSER included in monitor.
34	ST%MER	Magtape error reporting.
35	ST%SSP	Swapping done in page units.

Item	Symbol	Contains
112	**%CNMNT**	Monitor type flags are as follows:

Bits	Symbol	Monitor Type
Ø	CN%MNX	Unknown monitor.
77B23	CN%MNT	Monitor type:

Value	Symbol	Type
1	.CNT1Ø	TOPS-1Ø
2	.CNITS	ITS
3	.CNTNX	TENEX
4	.CNT2Ø	TOPS-2Ø
5	.CNTCX	TYMCOM-X

24-29	CN%MNS	DIGITAL monitor subtype.
3Ø-35	CN%MNC	Customer monitor subtype.

This word is used by operating systems that have TOPS-2Ø UUO compatibility packages.

For example, a calling sequence to read this word may be:

```
MOVE ac,[112,,11]
GETTAB ac,
   MOVEI ac,Ø
LDB ac,[POINT ac,CN/MNT]
CAIN ac,1
   JRST TOPS1Ø
CAIN ac,4
   JRST TOPS2Ø
JRST UNKNOWN
```

166	**%CNJPK**	Bit Ø of this word is a flag. If not set, this flag indicates that JOBPEK transfers cannot cross page boundaries.
167	**%CNDAE**	In the left half is the previous version of the monitor in SIXBIT. In the right half is the current version of the monitor in binary.

.GTNSW - Nonswapping Data Table
GETTAB Table 12

Contents: Data about nonswapping memory utilization.

Indexed by: Item number.

Monitor Table: NSWTBL

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

Some of this data is CPU-specific and exists in CDBs.

Word	Symbol	Map
10,,12	%NSCMX	System CORMAX (largest user job + 1)
11,,12	%NSCLS	Byte pointer to last free core area
12,,12	%NSCTL	Virtual core tally
13,,12	%NSSHW	Obsolete
14,,12	%NSHLF	Obsolete
15,,12	%NSUPT	System uptime (in ticks)
16,,12	%NSSHF	Obsolete
17,,12	%NSSTU	Obsolete
20,,12	%NSHJB	Highest job number in use
21,,12	%NSCLW	Words cleared by system
22,,12	%NSLST	Lost time
23,,12	%NSMMS	Memory size in words
24,,12	%NSTPE	Total memory parity errors
25,,12	%NSSPE	Spurious memory parity errors
26,,12	%NSMPC	Multiple memory parity errors
27,,12	%NSMPA	Absolute addr of last memory parity error
30,,12	%NSMPW	Contents of 1st bad wd on parity sweep
31,,12	%NSMPP	PC where last MEM PAR error was detected
32,,12	%NSEPO	Number of exec PDL overflows not recovered
33,,12	%NSEPR	Number of exec PDL overflows recovered

34,,12	%NSMXM	Maximum value of CORMAX
35,,12	%NSKTM	KSYS timer
36,,12	%NSCMN	Amt of memory guaranteed to non-locked jobs (CORMIN)
37,,12	%NSABC	Count of address breaks
40,,12	%NSABA	Address break addresses
41,,12	%NSLJR	Last job run
42,,12	%NSACR	Obsolete
43,,12	%NSNCR	Obsolete
44,,12	%NSSCR	Obsolete

.GTSDT - Swapping Data Table
GETTAB Table 13

Contents: Contains data pertinent to swapping.

Indexed by: Item number.

Monitor Table: SWPTBL

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

Word	Symbol	Map
Ø,,13	%SWBGH	Size (in pages) of biggest hole in core
1,,13	%SWFIN	Job number of job Being swapped (Positive if swapping in, negative if swapping out)
2,,13	%SWFRC	Job number of job being forced to swap out
3,,13	%SWFIT	Job number of job waiting to swap in
4,,13	%SWVRT	Number of 1P blocks of virtual core left in system
5,,13	%SWERC*	Swap error count and flags
6,,13	%SWPIN*	PDB swapping flag
7,,13	%SWEUJ	Segment,,UDB-address

%SWERC contains the count of swap read or write errors in its left half; bits 18 to 21 of the right half are the same as status bits returned by a GETSTS monitor call for the disk; bits 22 to 35 contain the count of bad 1K blocks.

%SWPIN is -1 if the monitor swaps Process Data Blocks and a swap-in is in progress.

.GTSGN - High Segment Parameters
GETTAB Table 14

Contents — One word for each job running on the system, giving parameters for the job's high segment. The definitions of these bits in this word may vary from monitor release to monitor release; therefore you should not reference .GTSGN in a program that is monitor-independent.

Indexed by — Job number.

Monitor Table — JBTSGN

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTSGN]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Word	Symbol	Map
Ø,,14	.GTSGN*	High segment parameters

.GTSGN parameters are as follows:

Bits	Symbol	Meaning
Ø	SN%SPY	Job is spying. See bits 18-35 below.
1	SN%SHR	Job high segment is sharable.
2	SN%UWP	Job high segment is write-enabled.
3	SN%MDL	Job high segment has been meddled.
4		Reserved.
5	SN%LOK	Job high segment is locked into memory.
6	SN%NCS	Job high segment is not cached.
7-9		Reserved.
1Ø	SN%GTS	Job high segment was obtained by GETSEG.
18-35		Defined as described below:

The right half of .GTSGN contains one of the following:

- If bit Ø = Ø, then the right half contains the job high segment index number.
- If bit Ø = 1, then the job is spying and the right half contains the maximum exec virtual address that can be spied upon.

.GTODP - ONCE-Only Disk Parameters
GETTAB Table 15

Contents: Disk parameters that are established at monitor generation time.

Indexed by: Item number.

Monitor Table: ODPTBL

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

		Obsolete
1,,15	%ODK4S	Number of K of disk words available for swapping
2,,15	%ODPRT	In-core protect time multiplier
3,,15	%ODPRA	In-core protect time offset
4,,15	%ODPMN	Minimum ICPT after requeue to back of PQ2
5,,15	%ODPMX	Maximum value of ICPT

.GTLVD - Level D Disk Parameters
GETTAB Table 16

Contents Project-programmer numbers for libraries, file data, and other data. These PPNs are established at monitor generation time.

Indexed by Item number.

Monitor Table LVDTBL

Calling Sequence

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

0,,16	%LDMFD	MFD PPN [1,1]
1,,16	%LDSYS	SYS PPN [1,4]
2,,16	%LDFFA	Full file access PPN [1,2]
3,,16	%LDHLP	Not-logged-in PPN [2,5]
4,,16	%LDQUE	Queue area PPN [3,3]
5,,16	%LDSPB	Addr of first PPB | Addr of next PPB to scan
6,,16	%LDSTR	First structure data block |Offset to next str data blk
7,,16	%LDUNI	First unit data block |Offset to nxt unit data blk
10,,16	%LDSWP	First swap unit | Offset to next swap unit
11,,16	%LDCRN	Number of 4-wd blks allocated at ONCE-only
12,,16	%LDSTP	Standard file protection
13,,16	%LDUFP	Standard UFD protection
14,,16	%LDMBN	Obsolete
15,,16	%LDQUS	SIXBIT queue structure name
16,,16	%LDCRP	CRASH PPN [10,1]
17,,16	%LDSFD	Maximum depth of SFDs to write
20,,16	%LDSPP	Spooled file protection
21,,16	%LDSYP	Standard SYS: protection
22,,16	%LDSSP	Standard SYS:filename.SYS protection
23,,16	%LDMNU	Maximum negative USETI that reads extended RIBs
24,,16	%LDMXT	Maximum blocks to transfer with 1 I/O operation
25,,16	%LDNEW	Experimental SYS PPN [1,5]

26,,16	%LDOLD	Old SYS PPN [1,3]
27,,16	%LDUMD	User-mode diagnostics PPN [6,10]
30,,16	%LDNDB	Default disk buffers in ring
31,,16	%LDMSL	Maximum units in active swapping list
32,,16	%LDALG	ALGOL library PPN [5,4]
33,,16	%LDBLI	BLISS library PPN [5,5]
34,,16	%LDFOR	FORTRAN library PPN [5,6]
35,,16	%LDMAC	MACRO source library PPN [5,7]
36,,16	%LDUNV	UNIVERSAL file library PPN [5,17]
37,,16	%LDPUB	Public user library PPN [1,6]
40,,16	%LDTED	Text editor library PPN [5,10]
41,,16	%LDREL	.REL file library PPN [5,11]
42,,16	%LDRNO	RUNOFF library PPN [5,12]
43,,16	%LDSNO	SNOBOL library PPN [5,13]
44,,16	%LDDOC	.DOC file library PPN [5,14]
45,,16	%LDFAI	FAIL library PPN [5,15]
46,,16	%LDMUS	Music library PPN [5,16]
47,,16	%LDDEC	Standard DIGITAL software [10,7]
50,,16	%LDSLP	AOBJN pointer to active swap list
51,,16	%LDBAS	BASIC library PPN [5,1]
52,,16	%LDCOB	COBOL library PPN [5,2]
53,,16	%LDMXI	PDP-11 library PPN [5,3]
54,,16	%LDNEL	NELIAC library PPN [5,20]
55,,16	%LDDMP	Dump PPN [5,21]
56,,16	%LDPOP	POP2 library PPN [5,22]
57,,16	%LDTST	TEST library PPN [5,23]
60,,16	%LDLSO*	If nonzero, call DAEMON to log soft overruns
61,,16	%LDMBR*	Massbus register pointers
62,,16	%LDBBP*	Pointer to BAT pointer | Channel terminal fail count
63,,16	%LDDBS	DBMS library PPN [5,24]
64,,16	%LDEXP*	Offset of expected channel term word in CDB
65,,16	%LDMIC	MIC macro library PPN [5,25]

66,16	%LDTPS	Text processing system library PPN [5,26]
67,,16	%LDCTL	.CTL file library PPN [5,27]
70,,16	%LDGAM	Games library PPN [5,30]
71,,16	%LDACT	System accounting PPN [1,7]
72,,16	%LDAPL	APL library PPN [5,31]
73,,16	%LDECT	RIB error threshold
74,,16	%LDTOT	Total RIB errors
75,,16	%LDDOR	Addr of first dormant acc table,,addr of last dormant acc table
76,,16	%LDCOR	Addr first free 4-wd core blk,,0
77,16	%LDINT	Disk interference count
100,16	%LDD60	D60 library PPN [5,32]
101,,16	%LDERT	Address of queue table for DAEMON error reporting
102,,16	%LDPT1	Pointer to extract entries for DAEMON queue table
103,,16	%LDPT2	Pointer to insert entries for DAEMON queue table
104,,16	%LDLTH	Length of DAEMON queue table
105,,16	%LDCDA	Offset of UNICDA in UDB
106,,16	%LDDES	Offset of UNIDES in UDB
107,,16	%LDPTR	Pointer to in-core copies of retrieval pointers
110,,16	%LDMSS	Max strs in sys search list | Max strs in job search list
111,,16	%LDSLB	Offset of UNISLB in UDB
112,,16	%LDUTP	Define ersatz device UTP
113,,16	%LDINI	INI PPN
114,,16	%LDESZ	Size of 1 entry in ERPTBK
115,,16	%LDKON	Pointer to first controller's data block
116,,16	%LDLBF	Default number of disk buffers
117,,16	%LDDVU	Offset to device unit number
120,,16	%LDCSZ	Size of disk cache, in blocks
121,,16	%LDRDC	Monitor cache block read calls
122,,16	%LDRDH	Monitor cache block read hits
123,,16	%LDWRC	Monitor cache block write calls
124,,16	%LDWRH	Monitor cache block write hits

125,,16	%LDHSF	CSHFND calls
126,,16	%LDHSC	CSHFND collisions in hash table
127,,16	%LDHSL	Length of cache hash table
130,,16	%LDHST	Address of cache hash table
131,,16	%LDCHD	Address of cache list header
132,,16	%LDSPN	DDB offset for spooled file name
133,,16	%LDSPM	DDB offset for spooled parameter block pointer
134,,16	%LDBLK	DDB offset for I/O block number
135,,16	%LDRSU	DDB offset to retrieval/acc blocks
136,,16	%LDNMB	DDB offset for NMB of father SFD
137,,16	%LDUPS	PPN for use by mail programs (UPS device)

Item	Contains
60	**%LDLSO** contains a flag for DAEMON. If %LDLSO is nonzero, and if an overrun is recovered on the first retry, then DAEMON is called.
61	**%LDMBR** contains massbus register pointers. The left half contains the offset into KBD of the number of registers. The right half contains the offset into UBD of the number of registers.
62	**%LDBBP** contains pointers. The left half contains the address of a byte pointer to the number remaining in the block access table. The right half contains the offset into the UDB of the channel terminal fail count.
64	**%LDEXP** contains the offset of the expected channel terminal word in the channel data block.

.GTRCT - Disk Blocks Read
GETTAB Table 17

Contents One word for each job running on the system, giving the number of disk blocks read by the job.

Indexed by Job number.

Monitor Table JBTRCT

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTRCT]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Disk blocks read

The word contains the number of incremental disk block reads in bits Ø to 11 (RC.INC==7777B11 in UUOSYM), and the total number of disk block reads in bits 12 to 35 (RC.TTL==77777777 in UUOSYM).

.GTWCT - Disk Blocks Written
GETTAB Table 2Ø

Contents — One word for each job running on the system, giving the number of disk blocks written by the job.

Indexed by — Job number.

Monitor Table — JBTWCT

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTWCT]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Disk blocks written

The word contains the number of incremental disk block writes in bits Ø to 11 (WC.INC==7777B11 in UUOSYM), and the total number of disk block writes in bits 12 to 35 (WC.TTL==77777777 in UUOSYM).

.GTSLF - GETTAB Table Data
GETTAB Table 23

Contents: Data for each GETTAB table.

Indexed by: GETTAB table number.

Monitor Table: NUMTAB

Calling Sequence:

```
MOVE    ac,[XWD table,.GTSLF]
GETTAB  ac,
  error return
normal return
```

where table is the symbolic name of the table whose data is required.

Word	Symbol	Map
Ø,,23	.GTSLF	GETTAB table data

The word contains one word for each GETTAB table. The word gives the following information:

Bits	Symbol	Meaning
Ø-8	SL.MAX	If SL.TYP is 1, 2, 3, or 4, this field is the largest item number in the table. If SL.TYP is 5, this field is the index into the range table.
9-11	SL.TYP	Type of table.

Value	Symbol	Type
Ø	.SLNIC	Not included in this system.
1	.SLIXI	Indexed by item number.
2	.SLIXJ	Indexed by job number.
3	.SLIXS	Indexed by job number or segment number.
4	.SLIXP	Indexed by job number; data in PDB.
5	.SLIXR	Indexed by item number. Range may not be Ø to length -1.

12-13		Reserved for DIGITAL.
14-17	SL.MAC	A monitor accumulator number.

Bits	Symbol	Meaning
18-35	SL.ADR	If SL.TYP=1, 2, or 3, this halfword contains the executive mode address of the table. If SL.TYP=4, this halfword contains the offset in job's PDB. If SL.TYP is 5, this halfword is the executive mode address that would correspond to index Ø in the table.

.GTDEV - Segment Device or Structure
GETTAB Table 24

Contents

One word for each high segment running on the system, giving the device or file structure for the sharable high segment.

Indexed by

Segment number.

Monitor Table

JBTDEV

Calling Sequence

```
MOVE    ac,[XWD segno,.GTDEV]
GETTAB  ac,
  error return
normal return
```

where segno is a high-segment number. Use -2 for the current high segment. For the high segment of a different job, obtain the segment number using a GETTAB to .GTSGN.

Device or structure (segments only)

This returns Ø if there is no such segment or if the segment is not sharable.

.GTWSN - Names of Wait States
GETTAB Table 25

Contents — Names of wait states used in job queues. The names of wait states may vary from monitor release to monitor release; therefore you should not reference .GTWSN in a program that is monitor-independent.

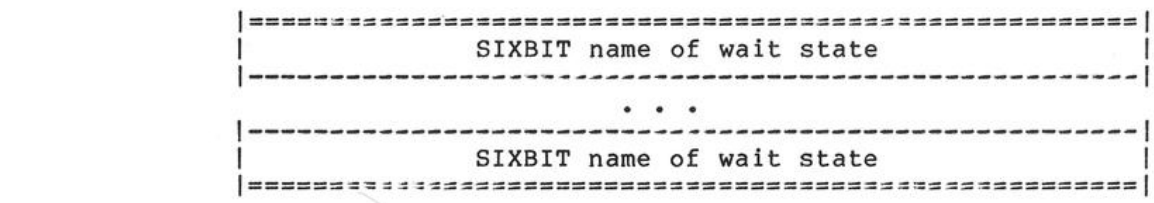
Indexed by — Item number.

Monitor Table — STSTBL

Calling Sequence

```
        MOVE    ac,[item]
        GETTAB  ac,
          error return
        normal return
```

where **item** is one of the symbols given in the word map below.



There are three 2-character SIXBIT names in each word returned. These states are defined at label STSTBL: in COMMON.

.GTLOC - Remote Station Number
GETTAB Table 26

Contents One word for each job running on the system, giving the station (ANF-1Ø network node) number for the job. This is the node at which the job is logically located (LOCATE call and monitor command).

Indexed by Job number.

Monitor Tables JBTLOC

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTLOC]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job or use Ø for the ANF-1Ø node number of the central station.

Network node number

.GTCOM - Monitor Command Names
GETTAB Table 30

Contents — Monitor command names as SIXBIT words.

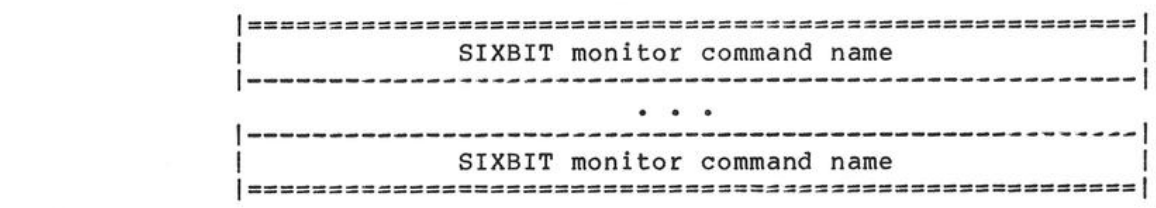
Indexed by — Command number.

Monitor Table — COMTAB

Calling Sequence

```
        MOVE    ac,[item]
        GETTAB  ac,
          error return
        normal return
```

where **item** is the position in the table of the required command name.



The HELP * command displays a list of these command names.

.GTNM1 and .GTNM2 - User Name
GETTAB Tables 31 and 32

Contents Two words for each job running on the system, giving the user's name in SIXBIT (up to 12 characters).

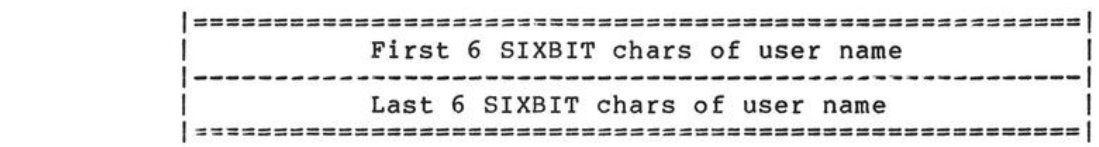
Indexed by Job number.

Monitor Table PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTNM1]
        GETTAB  ac,
          error return
        MOVEM   ac,uname1
        MOVE    ac,[XWD jobno,.GTNM2]
        GETTAB  ac,
          error return
        MOVEM   ac,uname2
        . . .
uname1: block 1
uname2: block 1
```

where **jobno** is the number of a logged-in job (use -1 for the current job); and **uname1** and **uname1** are locations for storing the user name.



.GTCNO - Charge Number
GETTAB Table 33

Contents — One word for each job running on the system, giving the accounting charge number for the job.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTCNO]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

User charge number

.GTWCH - Watch Bits
GETTAB Table 35

Contents One word for each job running on the system, giving the watch bits for the job. These bits are set by defaults, by the SETUUO or JBSET. monitor call, or by the SET WATCH monitor command.

Indexed by Job number.

Monitor Table JBTWCH

CALLING SEQUENCE

```
MOVE    ac,[XWD jobno,.GTWCH]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Watch flags

Watch flags are as follows:

Flag	Symbol	Meaning
Ø	JW.WCX	Watch context changes.
1	JW.WDY	Watch daytime at start.
2	JW.WRN	Watch runtime.
3	JW.WWT	Watch wait time.
4	JW.WDR	Watch disk reads.
5	JW.WDW	Watch disk writes.
6	JW.WVR	Watch versions.
7	JW.WMT	Watch magtape statistics.
8	JW.WFI	Watch file.
377B8	JW.WAL	Watch all.
9	JW.WCN	Message continuations.
1Ø	JW.WFL	First line of message.
11	JW.WPR	Message prefixes.
7B11	JW.WMS	Message level:

Value	Symbol	Message level
1	.JWWPR	Prefix only.
2	.JWWOL	One line.
3	.JWWPO	Prefix and first line.
6	.JWWLG	Long without prefix.
7	.JWWPL	Prefix and long.

12-35	Reserved.

.GTSPL - Spooling Control Flags
GETTAB Table 36

Contents — One word for each job running on the system, giving the file spooling flags for the job. These flags are set by the SETUUO or JBSET. monitor call, or by the SET SPOOL monitor command.

Indexed by — Job number.

Monitor Table — JBTSPL

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTSPL]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Spooling control flags

Spooling control flags are as follows:

Flag	Symbol	Meaning
26	JS.PRI	Disk priority spooling.
27	JS.DFR	Deferred spooling (old MPB style).
3Ø		Reserved.
31	JS.PCR	Spool card reader.
32	JS.PCP	Spool card punch.
33	JS.PPT	Spool papertape punch.
34	JS.PPL	Spool plotter.
35	JS.PLP	Spool line printer.
31-35	JS.PAL	Spool all devices.

.GTRTD - Realtime Status Word
GETTAB Table 37

Contents

One word for each job running on the system, giving the realtime status word for the job. The definitions of the bits in the realtime status word may vary from monitor release to monitor release; therefore you should not reference .GTRTD in a program that is monitor-independent.

Indexed by

Job number.

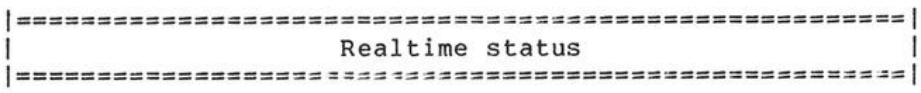
Monitor Table

JBTRTD

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTRTD]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.



.GTLIM - Time Limit and Batch Status
GETTAB Table 40

Contents — One word for each job running on the system, giving the time limit and batch status for the job.

Indexed by — Job number.

Monitor Table — JBTLIM

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTLIM]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Time and batch status

Time and batch status flags are as follows:

Bits	Symbol	Meaning
0	JB.LTL	Time limit set from forced DETACH command.
7B9	JB.LCR	Obsolete
10	JB.LBT	Batch job.
11	JB.LSY	Job using program from SYS.
12-35	JB.LTM	Time to go, in jiffies

.GTCM2 - SET Command Names
GETTAB Table 43

Contents

The SIXBIT names of all SET monitor commands. The contents of the .GTCM2 table may vary from monitor release to monitor release; therefore you should not reference .GTCM2 in a program that is monitor-independent.

Indexed by

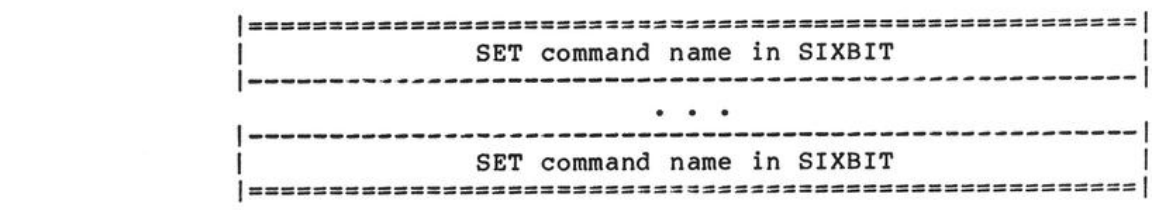
Command number.

Monitor Table

COMTB2

Calling Sequence

```
        MOVE    ac,[XWD index,.GTCM2]
        GETTAB  ac,
          error return
        normal return
```



These names are defined by the SNAMES macro in COMCON and will be displayed if you type the monitor command HELP *.

.GTCRS - Hardware Status After Crash
GETTAB Table 44

Contents	Hardware status words after a crash. (See also %CCCSB and %CCDSB.)
Monitor Table	CPU Status Block
Indexed by	Item number.
Calling Sequence	MOVE ac,[item] GETTAB ac, error return normal return

where item is one of the symbols given in the word map below.

Ø,,44	CR.SAP	APR CONI
1,,44	CR.SPI	PI CONI
2,,44	CR.SSW	APR DATAI switches

.GTISC - Swap In Scan Tables
GETTAB Table 45

Contents

Swapper input scan list of queues. The definitions of the bits in this table may vary from monitor release to monitor release; therefore you should not reference .GTISC in a program that is monitor-independent.

Indexed by

Queue number.

Monitor Table

ISCAN

Calling Sequence

```
MOVE    ac,[XWD index,.GTISC]
GETTAB  ac,
  error return
normal return
```

Swap-in scan tables

.GTOSC - Swap Out Scan Tables
GETTAB Table 46

Contents

Swapper output scan list of queues. The definitions of the bits in this table may vary from monitor release to monitor release; therefore you should not reference .GTOSC in a program that is monitor-independent.

Indexed by

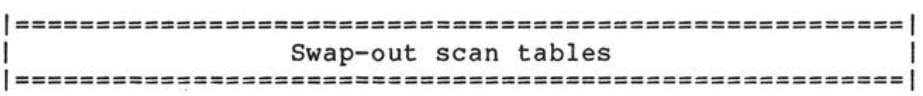
Queue number.

Monitor Table

OSCAN

Calling Sequence

```
        MOVE    ac,[XWD index,.GTOSC]
        GETTAB  ac,
          error return
        normal return
```



.GTSSC - Scheduler Scan Tables
GETTAB Table 47

Contents	Scheduler scan list of queues. The definitions of the bits in this table may vary from monitor release to monitor release; therefore you should not reference .GTSSC in a program that is monitor-independent.
Indexed by	Queue number.
Monitor Table	SSCAN
Calling Sequence	MOVE ac,[item] GETTAB ac, error return normal return

Scheduler scan tables

.GTRSP - Response Counter Table
GETTAB Table 5Ø

Contents — One word for each job running on the system, giving the time (in jiffies) when the job began its wait for running. This time is cleared when the job is placed in the run queue by the scheduler.

Indexed by — Job number.

Monitor Table — JBTRSP

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTRSP]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Response counter table

.GTSYS - System-Wide Data
GETTAB Table 51

Contents: System-wide data concerning errors and stopcodes.

Indexed by: Item number.

Monitor Table: SYSTBL

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

0,,51	%SYERR	System-wide hardware error count
1,,51	%SYCCO	Number of times COMCNT was wrong
2,,51	%SYDEL	Disabled hardware error count
3,,51	%SYSPC	Last 3-char stopcode | Last stopcode addr+1
4,,51	%SYNDS	Number of DEBUG stopcodes
5,,51	%SYNJS	Number of JOB stopcodes (+ DEBUGs if stopped)
6,,51	%SYNCP	Number of commands processed
7,,51	%SYSJN	Last stopcode job number
10,,51	%SYSTN	Last stopcode TTY name
11,,51	%SYSPN	Last stopcode program name
12,,51	%SYSUU	Last stopcode monitor call
13,,51	%SYSUP	Last stopcode user PC
14,,51	%SYSPP	Last stopcode user PPN
15,,51	%SYSCD	Last stopcode stopcode name
16,,51	%SYNCS	Total number of CPU stopcodes
17,,51	%SYNIS	Number of no dump stopcodes
20,,51	%SYSTY	Type of last stopcode
21,,51	%SYSUD	UDT of last stopcode
22,,51	%SYSCP	CPU number of last stopcode

.GTWHY - Operator Reload Comments
GETTAB Table 52

Contents	ASCIZ string giving the operator's reason for reloading.
Indexed by	Word of ASCIZ string.
Monitor Table	CRSHAC
Calling Sequence	MOVE ac,[XWD word,.GTWHY] GETTAB ac, error return normal return

Operator why comments in ASCIZ

.GTTRQ - Time in Run Queues
GETTAB Table 53

Contents

One word for each job running on the system, giving the total time the job was in the run queues (even if not running all the time).

NOTE

This table is usually set to Ø, because it is expensive for the CPU to maintain.

Indexed by

Job number.

Monitor Table

JBTRQT

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTTRQ]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Total time in RUN queues (whether or not running)

.GTSPS - Status Word for Subsequent Processors
GETTAB Table 54

Contents

Status bits for processors other than CPU0. The definitions of the bits in this word may vary from monitor release to monitor release; therefore you should not reference .GTSPS in a program that is monitor-independent.

Indexed by

Job number.

Monitor Table

JBTSPS

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTSPS]
GETTAB  ac,
  error return
normal return
```

Second processor status

Status flags are as follows:

Bits	Symbol	Meaning
29	SP.SC0	Can use SET CPU monitor command for CPU0. Bits 28 through 24 are similar for CPU1 through CPU5.
35	SP.CR0	Can use JBSET. or SETUUO monitor call for CPU0. Bits 30 through 34 are similar for CPU1 through CPU5.

.GTCnC - CPUn CPU Data Block Constants
GETTAB Tables 55, 57, 61, 63, 65, 67

Contents CPU data block constants for CPUn, where n is a CPU number from Ø to 5. For CPUs Ø to 5, respectively, these tables are called .GTCØC, .GTC1C, .GTC2C, .GTC3C, .GTC4C, and .GTC5C.

Indexed by Item number.

Monitor Table .CnCDB

Calling Sequence

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below; and n is the number of a CPU (from Ø to 5).

Word	Symbol	Map
Ø,,55	%CCPTR	Pointer to next CDB | Reserved
1,,55	%CCSER	APR serial number
2,,55	%CCOKP	Jiffies CPU has been down (<=Ø if OK)
3,,55	%CCTOS	EPT address for this CPU
4,,55	%CCLOG	Logical name (CPUn)
5,,55	%CCPHY	Physical name (CPXn) (X,I,L, or S)
6,,55	%CCTYP*	Customer processor code | DIGITAL processor code
7,,55	%CCMPT*	Pointer to bad address subtable in variable area
1Ø,,55	%CCRTC	Addr of realtime clock (DK1Ø) DDB
11,,55	%CCRTD	Addr of realtime clock DDB (if precision accounting)
12,,55	%CCPAR*	Pointer to parity subtable in variable area
13,,55	%CCRSP*	Pointer to response subtable in variable area
14,,55	%CCDKX	Number of DK1Øs on this CPU
15,,55	%CCEBS	Number of EBOX ticks/second on KL1Ø
16,,55	%CCMBS	Number of MBOX ticks/second on KL1Ø
17,,55	%CCNXT*	Pointer to NXM subtable in variable area
2Ø,,55	%CCCSB*	Pointer to CPU status block subtable in variable area
21,,55	%CCDSB*	Ptr to device status block subtable in variable area

22,,55	%CCSDP*	Ptr to SBDIAG subtable in variable area
23,,55	%CCBPA	Pointer to PERF. counts in variable subtable
24,,55	%CCCIP	CI port control block
25,,55	%CCNIP	NI port control block

Word	Contains
6	**%CCTYP** processor types are as follows:

Value	Symbol	Processor Type
1	.CC166	PDP-6
2	.CCKAX	KA1Ø
3	.CCKIX	KI1Ø
4	.CCKLX	KL1Ø
5	.CCKSX	KS1Ø

The pointers **%CCMPT, %CCPAR, %CCRSP, %CCNXT,** %CCCSB, and %CCDSB are pointers to GETTAB subtables for CPUn; these subtables follow table .GTCnV, and are described in the next few pages.

7 **%CCMPT** is of the form:

Bits	Symbol	Meaning
Ø-8	CC%BLN	Length-1 of bad address subtable.
9-17		Reserved.
18-35	CC%BRA	Offset into .GTCnV of bad address subtable.

12 **%CCPAR** is of the form:

Bits	Symbol	Meaning
Ø-8	CC%PLN	Length-1 of parity subtable.
9-17		Reserved.
18-35	CC%PRA	Offset into .GTCnV of parity subtable.

13 **%CCRSP** is of the form:

Bits	Symbol	Meaning
Ø-8	CC%RLN	Length-1 of response subtable.
9-17		Reserved.
18-35	CC%RRA	Offset into .GTCnV of response subtable.

17 **%CCNXT** is of the form:

Bits	Symbol	Meaning
Ø-8	CC%NLN	Length-1 of NXM subtable.
9-17		Reserved.
18-35	CC%NRA	Offset into .GTCnV of NXM subtable.

Word	Contains
20	%CCCSB is of the form:

Bits	Symbol	Meaning
0-8	CC%CLN	Length-1 of CPU status block subtable.
9-17		Reserved.
18-35	CC%CRA	Offset into .GTCnV of CPU status block subtable.

Word	Contains
21	%CCDSB is of the form:

Bits	Symbol	Meaning
0-8	CC%DLN	Length-1 of device status block subtable.
9-17		Reserved.
18-35	CC%DRA	Offset into .GTCnV of device status block subtable.

Word	Contains
22	%CCSDP is of the form:

Bits	Symbol	Meaning
0-8	CC%SLN	Length-1 of SBDIAG subtable.
9-17		Reserved.
18-35	CC%SRA	Offset into .GTCnV of SBDIAG subtable.

.GTCnV - CPUn CPU Data Block Variables
GETTAB Tables 56, 60, 62, 64, 66, 70

Contents — CPU data block variables for CPUn, where n is a CPU number (from 0 to 5). For CPUs 0 to 5, respectively, these tables are called .GTC0V, .GTC1V, .GTC2V, .GTC3V, .GTC4V and .GTC5V.

Indexed by — Item number.

Monitor Table — .CnVBG

Calling Sequence

```
MOVE    ac,[XWD item,<2*n>+.GTC0V]
GETTAB  ac,
  error return
normal return
```

where **item** is one of the symbols given in the word map below and **n** is a CPU number (from 0 to 5).

Word	Symbol	Map
		Reserved
		Reserved
		Reserved
		Reserved
		Reserved
5,,56	%CVUPT	Uptime in jiffies
		Reserved
		Reserved
		Reserved
		Reserved
12,,56	%CVLST	Lost time
		Reserved
14,,56	%CVTPE	Total memory parity errors
15,,56	%CVSPE	Spurious memory parity errors
16,,56	%CVMPC	Multiple memory parity errors
17,,56	%CVMPA	Absolute addr of last MEM PAR error
20,,56	%CVMPW	Contents of 1st bad wd on parity sweep

21,,56	%CVMPP	PC where last MEM PAR was found
		Reserved
		Reserved
		Reserved
		Reserved
		Reserved
27,,56	%CVABC	Address break count
30,,56	%CVABA	Address break address
31,,56	%CVLJR	Last job run (obsolete)
		Reserved
		Reserved
		Reserved
35,,56	%CVSTS	Number of job that stopped timesharing on this CPU
36,,56	%CVRUN*	Operator controlled scheduling
37,,56	%CVNUL	Null time
40,,56	%CVEDI	PC |Exec don't care interrupts
41,,56	%CVJOB	Current job
42,,56	%CVOHT	Overhead time in jiffies (exec UUOs)
43,,56	%CVEVM	Max exec virtual memory for LOCK mapping
44,,56	%CVEVU	Exec virtual memory used for LOCK mapping
45,,56	%CVLLC	No. of times CPU has looped waiting for interlock
46,,56	%CVTUC	Total monitor call count
47,,56	%CVTJC	Total job context switch count
50,,56	%CVTNE	Total nonexistent memory errors
51,,56	%CVSNE	Total nonreproducible NXM errors
52,,56	%CVNJA	Number of jobs affected by this NXM
53,,56	%CVMNA	First memory address with NXM
54,,56	%CVETJ	EBOX ticks/jiffy (computed)
55,,56	%CVNTJ	MBOX ticks/jiffy (computed)
56,,56	%CVBPA	Phys addr of bad parity word on last AR/ARX trap (KL)

57,,56	%CVTBD	Bad data on last AR/ARX trap
60,,56	%CVTGD	Good data after recovery from AR/ARX trap
61,,56	%CVNPT	Number of AR/ARX traps since reload
62,,56	%CVAER	RDERA results after unusual APR interrupt
63,,56	%CVPCN	CONI APR after parity interrupt
64,,56	%CVSB0	SBUS diagnostic function 0, word 0
65,,56	%CVS0A	SBUS diagnostic function 0, word 1
66,,56	%CVSB1	SBUS diagnostic function 1, word 0
67,,56	%CVS1A	SBUS diagnostic function 1, word 1
70,,56	%CVPPC	PC on AR/ARX trap
71,,56	%CVPFW	Page fail word on last AR/ARX trap
72,,56	%CVHPT	Number of hard AR/ARX traps
73,,56	%CVSPT	Number of soft AR/ARX traps
74,,56	%CVPTP	Number of page table parity errors
75,,56	%CVCSN	Number of cache sweeps since reload (cache sweep serial number)
76,,56	%CVCLN	Number of times a job couldn't run due to cache state
77,,56	%CVCLT	Lost time accrued due to cache state
100,,56	%CVCSD	Incr on swapper wait for cache sweep by another CPU
101,,56	%CVCRN	Cache sweep request sweep count (see COMMON.MAC)
102,,56	%CVCEC	Count nonrecoverable AR/ARX prty errs involving cache
103,,56	%CVPTR	Retry word for AR/ARX parity error trap routine
104,,56	%CVTSD	AR/ARX trap routine has saved APR ERA.SB Diags
105,,56	%CVREP	Used by NXM/parity recovery routines
106,,56	%CVNDB	Number of times this CPU's doorbell was rung
107,,56	%CVSBR	Status blocks read on this CPU
110,,56	%CVBPF	0 if performance counts being kept (%CCBPA)
111,,56	%CVFBI	Number of file blocks input (read)
112,,56	%CVFBO	Number of file blocks output (written)

113,,56	%CVSBI	Number of swapping blocks input (read)
114,,56	%CVSBO	Number of swapping blocks output (written)
115,,56	%CVSNC	Number of CPU stopcodes on this CPU
116,,56	%CVSND	Number of DEBUG stopcodes on this CPU
117,,56	%CVSNJ	Number of job stopcodes on this CPU
120,,56	%CVSJN	Last stopcode on this CPU - job number
121,,56	%CVSNM	Stopcode name | Stopcode PC+1
122,,56	%CVSPN	Program running at last stopcode
123,,56	%CVSPP	PPN of user running at last stopcode
124,,56	%CVSTN	TTY name of user running at last stopcode
125,,56	%CVSUP	User PC at last stopcode
126,,56	%CVSUU	UUO at last stopcode
127,,56	%CVEJN	Last parity/NXM error on this CPU - job number
130,,56	%CVEPN	Last parity/NXM error on this CPU - job name
131,,56	%CVPPI	CONI PI, at last parity/NXM interrupt
132,,56	%CVTPI	CONI PI, at last error trap
133,,56	%CVRQS	Requests for scheduler interlock
134,,56	%CVTFI	Number of magnetic tape frames read
135,,56	%CVTFO	Number of magnetic tape frames written
136,,56	%CVSNI	Number of no dump stopcodes
137,,56	%CVSTY	Type of last stopcode on this CPU
140,,56	%CVSUD	UDT of last stopcode on this CPU

Subtable: Bad Addresses
Subtable of .GTCnV

Contents — Addresses generating parity errors on CPUn.

Indexed by — Sequence of bad addresses found.

Calling Sequence

```
        MOVE    ac,[%CCMPT+<2*n>]
        GETTAB  ac,
          error return
        ADDI    ac,item
        HRLZS   ac
        HRRI    ac,.GTCØV+<2*n>
        GETTAB  ac,
          error return
        normal return
```

where n is the number of the required CPU (from Ø to 5); and item is the item number of the required item in the subtable.

Word	Symbol	Map
Ø		First bad address found
		. . .
last		Last bad address found

where last is the sequence number of the last bad address found. The value of last+1 is stored in location %CVPTS in the parity subtable for the CPU. The maximum length for the bad address subtable is stored in bits Ø-8 of %CCMPT in table .GTCnC.

Subtable: Parity
Subtable of .GTCnV

Contents — Parity error data for CPUn, where n is a CPU number (from Ø to 5).

Indexed by — Item number.

Calling Sequence

```
MOVE    ac,[%CCPAR+<2*n>]
GETTAB  ac,
  error return
ADDI    ac,item
HRLZS   ac
HRRI    ac,.GTCØV+<2*n>
GETTAB  ac,
  error return
normal return
```

where n is the number of the required CPU (from Ø to 5); and item is one of the symbols given in the word map below.

Word	Symbol	Map
Ø	%CVPLA	Last (highest) address of parity error
1	%CVPMR	Relative (not virtual) addr in low or high segment of last PAR ERR
2	%CVPTS	Number of PAR errors found on last sweep
3	%CVPSC	Number of parity sweeps by monitor
4	%CVPUE	Number of user-enabled parity errors
5	%CVPAA	AND of bad address on last parity sweep
6	%CVPAC	AND of bad contents last sweep
7	%CVPOA	IOR of bad address on last parity sweep
1Ø	%CVPOC	IOR of bad contents last sweep
11	%CVPCS	Number of spurious channel errors
12	%CVMET	MOS errors this minute
13	%CVMEC	MOS errors sent to TGHA
14	%CVTME	Total MOS errors

Subtable: Responses
Subtable of .GTCnV

Contents — Response data.

Indexed by — Item number.

Calling Sequence

```
MOVE    ac,[%CCRSP+<2*n>]
GETTAB  ac,
  error return
ADDI    ac,item
HRLZS   ac
HRRI    ac,.GTCØV+<2*n>
GETTAB  ac,
  error return
normal return
```

where n is the number of the required CPU (from Ø to 5); and item is one of the symbols given in the word map below.

Word	Symbol	Map
Ø	%CVRSO*	Sum of all terminal OUT UUO responses
1	%CVRNO	Number of terminal OUT UUO responses
2	%CVRHO	High-sum square of terminal OUT UUO responses
3	%CVRLO	Low-sum square of terminal OUT UUO responses
4	%CVRSI*	Sum of terminal IN UUO responses
5	%CVRNI	Number of terminal IN UUO responses
6	%CVRHI	High-sum square of terminal IN UUO responses
7	%CVRLI	Low-sum square of terminal IN UUO responses
1Ø	%CVRSR	Sum of CPU quantum requeue responses
11	%CVRNR	Number of quantum requeue responses
12	%CVRHR	High-sum square of quantum requeue responses
13	%CVRLR	Low-sum square of quantum requeue responses
14	%CVRSX*	Sum of one of responses terminated by 1 of 3 above
15	%CVRNX	Number of responses reflected in %CVRSX
16	%CVRHX	High-sum square of responses in %CVRNX
17	%CVRLX	Low-sum square of responses in %CVRNX
2Ø	%CVRSC*	Sum of CPU responses
21	%CVRNC	Number of CPU responses
22	%CVRHC	High-sum square of CPU responses

23	%CVRLC	Low-Sum Square of CPU Responses
32	%CVNRI	Number of characters received
33	%CVNXI	Number of characters sent
34	%CVNEI	Number of characters echoed

The Responses Subtable contains information concerning the response times calculated for user jobs, on a per-CPU basis. The responses subtable is made up of blocks of four words each. Each block contains information pertaining to a type of response that is measured. The following format is used for each block:

Word 1: Sum of responses, where response time is measured in ticks

Word 2: Number of responses

Words 3 - 4: A double-word integer containing the sum of squares of response times.

%CVRSI is the input response time. This value is increased every time a job runs a program whose first event (of those measured and stored in %CVRSI, %CVRSO, and %CVRSR) is a terminal input operation. Note that the input response time is calculated only once for this program, the first time it does input from the terminal, and is measured from the time that the monitor receives the command to run the program, to the time the program does its first terminal input UUO.

%CVRSO is the output response time and is similar to the input response time. Only the first terminal output done by the program is calculated and added to this word.

%CVRSR is the quantum requeue response time. This time is measured for compute-bound jobs (jobs that finish a CPU quantum without performing a terminal input or output operation), and measures the amount of time from the time the monitor receives the command to run the program, to the time the program must be rescheduled for more CPU time. Again, this time is calculated only once for each program execution.

%CVRSX is the response time for the first of the above three events to occur for the job. If a program does an input operation before an output and before a quantum expires, the input response time (also calculated in %CVRSI) is stored in %CVRSX. If a second job runs a program that does an output operation first, the response time for the output is stored in %CVRSO and added to %CVRSX.

Before the values in these locations can be used, however, it is important to understand the way that response time is calculated. The intention of counting response time is to understand the amount of time it takes to reach one of the three measured events (input operation, output operation, or quantum expiration). The user may, for example, type a command. The response time is the lag between the time the monitor accepts the command and the time it takes to reach one of the three events. Note that this does not include the time the user spends typing the command. The response time is counted in jiffies (ticks).

The Responses Subtable collects response times for the first event for a job when it begins running a program. If the first event for a program is to output a * as a command prompt, the amount of time between the time the "RUN program" command is accepted by the monitor and the time the monitor sends the * to the user's terminal is measured as the response time for that program. If a job runs a program that immediately begins input from the terminal, the response time for that action is the amount of time between the time the "RUN program" command is accepted by the monitor and the time the terminal input operation is attempted. The response times are accumulated for all the jobs as they are scheduled to run.

Subtable: Nonexistent Memory
Subtable of .GTCnV

Contents — Nonexistent memory data.

Indexed by — Item number.

Calling Sequence

```
MOVE    ac,[%CCNXT+<2*n>]
GETTAB  ac,
  error return
ADDI    ac,item
HRLZS   ac
HRRI    ac,.GTCØV+<2*n>
GETTAB  ac,
  error return
normal return
```

where n is the number of the required CPU (from Ø to 5); and item is one of the symbols given in the word map below.

Word	Symbol	Map
Ø	%CVNLA	Last NXM address
1	%CVNMR	Last NXM relative address
2	%CVNTS	Number of NXMs found this sweep
3	%CVNSC	Number of NXM sweeps done
4	%CVNUE	Number of user-enabled NXMs
5	%CVNAA	AND of blad addresses
6	%CVNOA	IOR of bad addresses
7	%CVNCS	Number of spurious channel NXMs

Subtable: CPU Status Block
Subtable of .GTCnV

Contents — CPU status block data.

Indexed by — Item number.

Calling Sequence

```
MOVE    ac,[%CCCSB+<2*n>]
GETTAB  ac,
  error return
ADDI    ac,item
HRLZS   ac
HRRI    ac,.GTCØV+<2*n>
GETTAB  ac,
  error return
normal return
```

where n is the number of the required CPU (from Ø to 5); and item is one of the symbols given in the word map below.

Word	Symbol	Map
Ø	%CVSAI	APRID
1	%CVSAP	CONI APR,
2	%CVSPI	CONI PI,
3	%CVSPD	DATAI PAG,
4	%CVSPC	CONI PAG,
		Reserved
		Reserved
		Reserved
		Reserved
11	%CVSER	RDERA
12	%CVSRD	CONI RH2Ø for First RH
		. . .
		CONI RH2Ø for eighth RH
22	%CVSDT	Four words for the 4 DTEs on a CPU

			EPT location 0
			. . .
			EPT location 37
66		%CVSE1	EPT location 140
			. . .
			EPT location 177
126		%CVSU1	UPT location 500
			. . .
			UPT location 503
132		%CVSA6	AC block 6, register 0
			. . .
			AC block 6, register 3
136		%CVSA7	AC block 7, register 0
			. . .
			AC block 7, register 2
141	(1)	%CVSSB	First word of SBDIAG data
			. . .
			50th word of SBDIAG data

Note (1) Each SBDIAG Block has the format:

Number of sub-block blocks,,offset to first

Each sub-block has the format:

Number of words,,logical
controller#
function 0 word 1
function 1 word 1

Subtable: Device Status Block
Subtable of .GTCnV

Contents — Device status block data.

Indexed by — Item number.

Calling Sequence

```
MOVE    ac,[%CCDSB+<2*n>]
GETTAB  ac,
  error return
ADDI    ac,item
HRLZS   ac
HRRI    ac,.GTCØV+<2*n>
GETTAB  ac,
  error return
normal return
```

where n is the number of the required CPU (from Ø to 5); and item is the item number of the required item in the subtable.

The table of device status for devices on this CPU contains the results of executing the instructions in the table obtained with the .GTDCD GETTAB. This table and .GTDCD are parallel tables with a one-for-one mapping of instructions in .GTDCD and resumes here. Intentionally, there is no order specified. It is intended that a program (DAEMON/SPEAR) should get one instruction from .GTDCD, print its symbol and device code in octal, and then print the value from this table.

Subtable: Background Performance Analysis
Subtable of .GTCnV

Contents: Disk/PI usage.

Indexed by: Item number.

Calling Sequence:

```
MOVE    ac,[%CCSDP+<2*n>]
GETTAB ac,
  error return
ADDI    ac,item
HRLZS   ac
HRRI    ac,.GTCØV+<2*n>
GETTAB  ac,
  error return
normal return
```

Word	Symbol	Map
Ø	%CVCHØ	RH2Ø #Ø usage
4	%CVCH1	RH2Ø #1 usage
1Ø	%CVCH2	RH2Ø #2 usage
14	%CVCH3	RH2Ø #3 usage
2Ø	%CVCH4	RH2Ø #4 usage
24	%CVCH5	RH2Ø #5 usage
3Ø	%CVCH6	RH2Ø #6 usage
34	%CVCH7	RH2Ø #7 usage
4Ø	%CVPIØ	PI level Ø (DTE, KLIPA, and KLINI) usage
44	%CVPI1	PI level 1 usage
5Ø	%CVPI2	PI level 2 usage
54	%CVPI3	PI level 3 usage
6Ø	%CVPI4	PI level 4 usage
64	%CVPI5	PI level 5 usage
7Ø	%CVPI6	PI level 6 usage
74	%CVPI7	PI level 7 usage

.GTFET - Feature Test Settings
GETTAB Table 71

Contents — Feature test settings that describe the features included in the current monitor.

Indexed by — Item number.

Monitor Table — FETTBL

Calling Sequence

```
MOVE    ac,[item]
GETTAB  ac,
  error return
TLNN    ac,<bit+777777>
  JRST  not-available-address
TRNN    ac,<bit+777777>
  JRST  feature-test-off-address
JRST    feature-test-on-address
```

where item is one of the symbols given in the word map below; bit is one of the feature test bits given below the word map.

Word	Symbol	Map
0,,71	%FTUUO*	Monitor call features
1,,71	%FTRTS*	Realtime and scheduler features
2,,71	%FTCOM*	Command features
3,,71	%FTACC*	Accounting features
4,,71	%FTERR*	Error control and option features
5,,71	%FTDEB*	Debugging features
6,,71	%FTSTR*	File structure features and parameters
7,,71	%FTDSK*	Internal disk features and parameters
10,,71	%FTSCN*	Scanner option features
11,,71	%FTPER*	I/O features and parameters
12,,71	%FTPE2*	More I/O features and parameters
13,,71	%FTDS2*	More internal disk features and parameters
14,,71	%FTST2*	More file structure features and parameters
15,,71	%FTUU2*	More monitor call features

%FTUUO monitor call feature test flags (more at %FTUU2 below) are as follows:

Bits	Symbol	Feature
19	F%MDA	Mountable device allocator.
20	F%MLOG	MIC log file support.
21	F%MIC	MACRO command processor.
22	F%EQDQ	ENQ./DEQ. monitor calls.
23	F%GALA	GALAXY (always set).
24	F%PI	Software PI system.
25	F%IPCF	IPCF.
26	F%CCIN	CTRL/C intercept.
27	F%PTYU	JOBSTS and CTLJOB monitor calls.
28	F%PEEK	PEEK monitor call.
29	F%POKE	POKE. monitor call.
30	F%JCON	Job continuation.
31	F%SPL	Spooling.
32	F%PRV	Job privileges.
33	F%DAEM	DAEMON monitor call.
34	F%GETT	GETTAB monitor call.
35	F%2REL	2-register relocation (obsolete).

%FTRTS realtime and scheduler feature test flags are as follows:

Bits	Symbol	Feature
23	F%CMSR	Communication measurement.
24	F%PSCD	Scheduler performance gathering.
25	F%NSCH	New scheduler.
26	F%VM	Virtual memory (always on).
27	F%SWAP	Swapper (defined in S, always on).
28	F%SHFL	Shuffler (obsolete).
29	F%RTC	DK10 service.
30	F%LOCK	LOCK monitor call.
31	F%TRPS	TRPSET monitor call.
32	F%RTTR	RTTRP monitor call.
33	F%SLEE	SLEEP monitor call.
34	F%HIBW	HIBER and WAKE monitor calls.
35	F%HPQ	HPQ monitor call.

%FTCOM command feature test flags are as follows:

Bits	Symbol	Feature
19	F%PJOB	Extended PJOB command.
20	F%EXE	.EXE file format.
21	F%MOFF	Set memory off-line.
22	F%MONL	Set memory on-line.
23	F%CCL	COMPILE commands (defined in S, always on).
24	F%CCLX	COMPILE-class commands.
25	F%QCOM	QUEUE and related commands.
26	F%SET	SET command and SETUUO monitor call.
27	F%VERS	Version.
28	F%BCOM	Batch control files.
29	F%SEDA	Set daytime and date.
30	F%WATC	SET WATCH command and monitor call.
31	F%FINI	FINISH and CLOSE commands.
32	F%REAS	REASSIGN command and monitor call.
33	F%EXAM	E and D commands.
34	F%TALK	SEND command.
35	F%ATTA	ATTACH command and monitor call.

%FTACC accounting feature test flags are as follows:

Bits	Symbol	Feature
28	F%ACCT	Accounting support.
29	F%EMRT	KL1Ø EBOX/MBOX user runtime capability.
3Ø	F%FDAE	File DAEMON.
31	F%TLIM	Limits for time, core, and so forth.
32	F%CNO	Accounting charge numbers.
33	F%UNAM	User names.
34	F%KCT	Kilo-core ticks.
35	F%TIME	Run time.

%FTERR error control and option feature test flags are as follows:

Bits	Symbol	Feature
19	F%SCA	Systems Communication Architecture is supported.
2Ø	F%KLP	KL-paging is enabled.
21	F%KS1Ø	KS1Ø processor.
22	F%MNXM	Nonexistent memory error recovery.
23	F%KL1Ø	KL1Ø processor.
24	F%KA1Ø	KA1Ø processor (obsolete, always off).
25	F%22BI	22-bit channel (DF1ØC).
26	F%PDBS	Swapping PDB (always off).
27	F%KI1Ø	KI1Ø processor.
28	F%METR	METER. monitor call.
29	F%EXON	Execute-only files (always on since 6.Ø1).
3Ø	F%KII	KI1Ø instruction check on KA1Ø.
31	F%BOOT	BOOTS bootstrap.
32	F%2SWP	Multi-swapping devices.
33	F%EL	DAEMON error logging.
34	F%MS	Multi-processors.
35	F%MEMP	Memory parity error recovery.

%FTDEB debugging feature test flags are as follows:

Bits	Symbol	Feature
27	F%2SEG	2-segment monitor (always on).
28	F%RSP	Response time.
29	F%WHY	Why reload logging.
3Ø	F%PATT	Patch space in tables.
31	F%TRAC	Back-tracking features.
32	F%HALT	Halts in monitor.
33	F%RCHK	Internal redundancy checks.
34	F%MONP	Monitor write-protected (always off).
35	F%CHEC	Monitor check-summed (always off).

%FTSTR file structure feature test flags (more at %FTST2 below) are as follows:

Bits	Symbol	Feature
19	F%DHIA	High availability features.
20	F%DSIM	Simultaneous file update.
21	F%NUL	Null.
22	F%LIB	LIB/SYS/OLD/NEW, and other structures.
23	F%DPRI	Disk priority transfers.

Bits	Symbol	Meaning
24	F%APLB	Append to last file block.
25	F%AIR	Append implies read.
26	F%GSRC	Generic device searching.
27	F%DRDR	Rename across directories.
28	F%DSEK	SEEK monitor call.
29	F%DSUP	Super USETI/USETO monitor calls.
30	F%DQTA	Disk quotas.
31	F%STR	Multiple structures.
32	F%5UUO	Miscellaneous 5-series monitor calls.
33	F%PHYO	Physical devices only.
34	F%SFD	Subfile directories (SFDs).
35	F%MOUN	STRUUO monitor call functions.

%FTDSK internal disk parameter flags (more at %FTDS2 below) are as follows:

Bits	Symbol	Feature
19	F%SLCK	Debug search list code.
20	F%2ATB	2-part access blocks.
21	F%CBDB	Debug CB interlock.
22	F%LOGI	LOGIN (defined in S, always on).
23	F%DISK	Disk system (defined in S, always on).
24	F%FFRE	Prevent races in FILFND.
25	F%SWPE	Swap read error recovery.
26	F%DBBK	Bad block marking.
27	F%DUFC	UFD compressing.
28	F%DETS	Disk error simulator.
29	F%DMRB	Multi-RIBs.
30	F%DSMC	Smaller allocation of disk core blocks.
31	F%DALC	Allocation optimization.
32	F%DSTT	Disk-usage statistics.
33	F%DHNG	Hung disk recovery.
34	F%DBAD	Disk offline recovery.
35	F%DOPT	Latency optimization.

%FTSCN scanner option feature test flags are as follows:

Bits	Symbol	Feature
20	F%RP04	RP04 support.
21	F%RDX	Remote data-entry service.
22	F%DCXH	DC10-H (2741 on DC10) support.
23	F%TVP	Fancy vertical positioning.
24	F%TYPE	TYPESET-10 features on DC76.
25	F%2741	Support for 2741-like terminals.
26	F%CAFE	DC76 support.
27	F%TBLK	TTY blank command.
28	F%TPAG	PAGE and display knowledge.
29	F%DIAL	Auto-dialer.
30	F%SCLC	Special line control.
31	F%SCNR	Hardware scanner.
32	F%MODM	Modem control.
33	F%630H	Single-scanner 630.
34	F%GPO2	Modem support (Great Britain).
35	F%HDPX	Truly half-duplex terminals.

%FTPER I/O parameter feature test flags (more at %FTPE2 below) are as follows:

Bits	Symbol	Feature
19	F%RDBA	Read backwards on TU70.
20	F%TAPO	TAPOP. monitor call.
21	F%TLAB	Tape label support.
22	F%TASK	Task-to-task network support.
23	F%DAS7	DAS78 (remote 360/370/2780) support.
24	F%XTC	DA28-C network support.
25	F%MSGS	MSGSER (MPX device) monitor module.
26	F%HSLN	High-speed logical device search.
27	F%CPTR	CDP trouble intercept.
28	F%CRTR	CDR trouble intercept.
29	F%CTY1	Support device CTY1 (always on).
30	F%NET	Network software.
30	F%REM	Remote-station software.
31	F%LPTR	LPT-device error recovery.
32	F%OPRE	Device errors to operator.
33	F%CDRS	CDR superimage mode.
34	F%MTSE	Magtape density/block commands.
35	F%TMP	TMPCOR area (always on).

%FTPE2 I/O parameter feature test flags (more at %FTPER above) are as follows:

Bits	Symbol	Feature
35	F%DX10	DX10 device-chaining (magtapes).

%FTDS2 internal disk parameter feature test flags (more at %FTDSK above) are as follows:

Bits	Symbol	Feature
35	F%DUAL	Dual-ported disks (RP04, RP06).

%FTST2 file structure parameter feature test flags (more at %FTSTR above) are as follows:

Bits	Symbol	Feature
35	F%PSTR	Private file structures.

%FTUU2 monitor call feature test flags (more at %FTUUO above) are as follows:

Bits	Symbol	Feature
35	F%MPB	MPB batch code.

.GTEDN - Ersatz Device Names
GETTAB Table 72

Contents: SIXBIT names of ersatz devices. The first blank name is the end of the table.

Indexed by: Item number.

Monitor Table: SDVTBL

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is the item number of the required name.

Word	Symbol	Map
Ø		First ersatz device name
		. . .
last		Last ersatz device name

where the entry following last is blank (Ø).

The ersatz device names are defined in the EDEVS macro in COMMOD.

.GTSCN - Scanner Data
GETTAB Table 73

Contents: Scanner data.

Indexed by: Item number.

Monitor Table: .GTSCN

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

Word	Symbol	Map
Ø,,73	%SCNRI	Number of receive interrupts
1,,73	%SCNXI	Number of transmit interrupts
2,,73	%SCNEI	Number of echo interrupts (in %SCNXI)
3,,73	%SCNMB	Maximum buffer size
4,,73	%SCNAL	Number of active lines
5,,73	%SCNPS	Size of buffer for PIM mode
6,,73	%SCNRA	Address of receive interrupt routine
7,,73	%SCNXA	Address of transmit interrupt routine
1Ø,,73	%SCNTA	Address of type
11,,73	%SCTFT	Address of first TTY chunk on free list
12,,73	%SCTFP	Address of last TTY chunk on free list
13,,73	%SCRCQ	Number of characters queued or deferred
14,,73	%SCRQF	Number of characters lost for queue overflow

.GTSNA - Last SEND ALL in 9-Bit
GETTAB Table 74

Contents: Data for last send-all message.

Indexed by: Item number.

Monitor Table: SNDTMP

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where **item** is one of the symbols given in the word map below.

Word	Symbol	Map
		Byte pointer to end byte in message
0,,74	%SCNAE	
		Byte pointer to first-1 byte in message
1,,74	%SCNAS	
		First word of data in message
2,,74	%SCNAM	
		.
		.
		.
		Last word of data in message

.GTCMT - SET TTY Command Names
GETTAB Table 75

Contents The SIXBIT names of the SET TTY monitor commands. The last name is followed by a blank word.

Indexed by Not indexed.

Monitor Table TTCWDT

Calling Sequence

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is the number of the name to be returned.

Word	Map
ø	First SET TTY command name
	. . .
last	Last SET TTY command name

where last is the number-1 of SET TTY commands in the table.

The SET TTY command names are defined with the TTNAME macro in COMCON and will be displayed if you type the HELP * monitor command.

.GTPID - Process Communication ID (IPCF)
GETTAB Table 76

Contents All process communication identifiers (PIDs) that have been assigned by the system. The default length of the table is twice the number of jobs that can run.

Indexed by Item number.

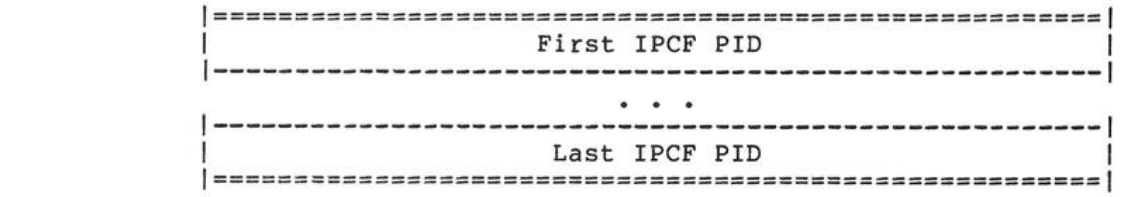
Monitor Table PIDTAB

Calling Sequence

```
        MOVE    ac,[item]
        GETTAB  ac,
          error return
        normal return
```

where item is the number of the desired item in the table. The entry after the last PID in the table is Ø.

Word **Map**



where last is the number-1 of PIDs in the table.

.GTIPC - IPCF Miscellaneous Data
GETTAB Table 77

Contents: Miscellaneous IPCF data.

Indexed by: Item number.

Monitor Table: IPCTAB

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

Word	Symbol	Map
Ø,,77	%IPCML	Maximum IPCF packet length
1,,77	%IPCSI	PID of system-wide [SYSTEM]INFO
2,,77	%IPCDQ	Default data
3,,77	%IPCTS	Total packets sent
4,,77	%IPCTO	Total packets outstanding
5,,77	%IPCCP	PID of [SYSTEM]IPCC
6,,77	%IPCPM	PID mask
7,,77	%IPCMP	Length of PID table
1Ø,,77	%IPCNP	Number of PIDs now defined
11,,77	%IPCTP	Total PIDs defined since reload
12,,77	%IPCIC	Number of IPCF pages currently in core
13,,77	%IPCSP	PID of [SYSTEM]GOPHER

.GTUPM - Physical Page of User Page Map
GETTAB Table 1ØØ

Contents One word for each job running on the system, giving the physical page number of the job's page map.

Indexed by Job number or segment number.

Monitor Table JBTUPM

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTUPM]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job (use -1 for the current job) or a high-segment number (use -2 for the current high segment).

For high segment entries, this is the page number of the start of the high segment. Note that only the left half contains the page number. The right half contains other data and should be ignored.

User page map page number

.GTCMW - SET WATCH Command Names
GETTAB Table 1Ø1

Contents The SIXBIT names of the SET WATCH monitor commands.

Indexed by Item number.

Monitor Table WATTAB

Calling Sequence

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is the number of the desired item in the table.

Word	Map
Ø	First SET WATCH command name
	. . .
last	Last SET WATCH command name

where last is the number-1 of command names in the table.

The SET WATCH command names are defined with the WATTAB macro in COMCON and will be displayed if you type the HELP * monitor command.

.GTCVL - Current Virtual and Physical Limits
GETTAB Table 1Ø2

Contents

One word for each job running on the system, giving the current virtual and physical page limits for the job.

Indexed by

Job number.

Monitor Table

PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTCVL]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Current virtual limit	Current physical limit

.GTMVL - Maximum Virtual and Physical Limits
GETTAB Table 1Ø3

Contents

One word for each job running on the system, giving the maximum virtual and physical page limits for the job.

Indexed by

Job number.

Monitor Table

PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTMVL]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Maximum virtual limit	Maximum physical limit

.GTIPA - IPCF Statistics for Job
GETTAB Table 104

Contents — One word for each job running on the system, giving the IPCF statistics for the job.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTIPA]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

IPCF statistics for job

IPCF statistics bits are as follows:

Bits	Symbol	Meaning
0-17	IP.CQD	Count of sends since the job logged in.
18-35	IP.CQC	Count of receives since the job logged in.

.GTIPP - IPCF Pointers and Counts
GETTAB Table 1Ø5

Contents — IPCF pointers and counts for the system.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTIPP]
        GETTAB  ac,
          error return
        normal return
```

IPCF pointers and counts

IPCF pointer and count bits are as follows:

Bits	Symbol	Meaning
Ø-17		Reserved.
18-26	IP.CQP	Outstanding sends.
27-35	IP.CQO	Outstanding receives.

.GTIPI - PID for Job's [SYSTEM]INFO
GETTAB Table 1Ø6

Contents One word for each job running on the system, giving the process communication identifier for the job's [SYSTEM]INFO.

Indexed by Job number.

Monitor Table PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTIPI]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

PID for [SYSTEM]INFO for job

.GTIPQ - IPCF Flags and Quotas for Job
GETTAB Table 1Ø7

Contents One word for each job running on the system, giving the IPCF quotas and flags for the job.

Indexed by Job number.

Monitor Table PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTIPQ]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

IPCF flags and quotas for job

Quota and flag bits are as follows:

Bits	Symbol	Meaning
Ø	IP.CQX	Disabled.
1	IP.CQQ	Quota set.
777B17		PID quota.
777B26	IP.CQS	Send quota.
777B35	IP.CQR	Receive quota.

.GTDVL - Pointer to Job Logical Name Table
GETTAB Table 110

Contents

One word for each job running on the system, giving a pointer to the table of addresses of DDBs owned by this job which have logical names.

Indexed by

Job number.

Monitor Table

PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTDVL]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Pointer to logical name table for job

.GTABS - Address Break Word
GETTAB Table 111

Contents

The address break word. The definitions of the bits in this word may vary from monitor release to monitor release and processor to processor; therefore you should not reference .GTABS in a program that is monitor-independent.

Indexed by

Job number.

Monitor Table

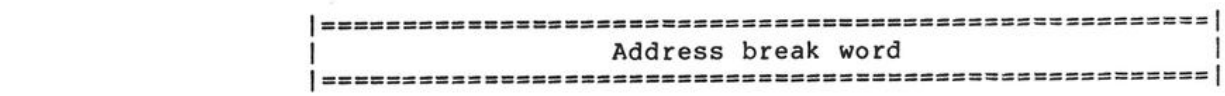
PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTABS]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Word Map



.GTVM - General Virtual Memory Data
GETTAB Table 113

Contents: Data about virtual page handling.

Indexed by: Item number.

Monitor Table: .GTVM

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

Word	Symbol	Map
0,,113	%VMSWP	Swap count
1,,113	%VMSCN	Scan count
2,,113	%VMSIP	Swaps + page operations in progress
3,,113	%VMSLE	Number of SWPLST entries
4,,113	%VMTTL	Total virtual memory in use
5,,113	%VMCMX	Maximum value of %VMTTL allowed
6,,113	%VMRMX	Obsolete
7,,113	%VMCON	Constant used in swap rate computation
10,,113	%VMQJB	Obsolete
11,,113	%VMRMJ	Obsolete
12,,113	%VMTLF	Time of last fault
13,,113	%VMSPF	System page fault counts
14,,113	%VMSW1	Address of SWPLST
15,,113	%VMSW2	Address of SW2LST
16,,113	%VMSW3	Address of SW3LST
17,,113	%VMEXP	Time constant exponent
20,,113	%VMDIF	%VMEXP - %VMCON
21,,113	%VMMXI	Maximum interval for fault-rate computation
22,,113	%VMIPC	Count of IPCF pages being swapped out

23,,113	%VMUPJ	Offset of job number in UPMP
24,,113	%VMUPR	Offset of end of low segment in UPMP
25,,113	%VMLST	Offset of pointer to swappable DDBs in UPMP
26,,113	%VMUPM	Virtual address of UPMP
27,,113	%VMLNM	Offset of pointer to logical names in UPMP
30,,113	%VMIC1	Number of swap input requests in SWPLST
31,,113	%VMHUA	Highest unmapped EXEC address
32,,113	%VMPPB	Address of beginning of per-process space
33,,113	%VMPPE	Address of end + 1 of per-process space
34,,113	%VMPPJ	Address of per-process user JOBDAT
35,,113	%VMFCC	Offset in UPMP for TMPCOR
36,,113	%VMCTA	Offset in UPMP for extended channel table pointer
37,,113	%VMJDA	EXEC virtual address of USRJDA
40,,113	%VMRMC	Real maximum CORMAX

.GTVRT - Paging Rate for Job
GETTAB Table 114

Contents One word for each job running on the system, giving the page rate for the job.

Indexed by Job number.

Monitor Table JBTVRT

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTVRT]
        GETTAB  ac,
          error return
        normal return
```

where **jobno** is the number of a logged-in job. Use -1 for the current job.

Paging rate for job	Reserved

.GTSST - Scheduler Statistics
GETTAB Table 115

Contents: Statistics kept by and for the job scheduler.

Indexed by: Item number.

Monitor Table: .GTSST

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

Word	Symbol	Map
Ø,,115	%SSOSO	Number of jobs run out-of-order to allow them to give up resources for swap-out
1,,115	%SSORJ	Number of jobs run out-of-order to allow them to give up resources required to run a job
2,,115	%SSNUL	Swapper null time
3,,115	%SSLOS	Swapper lost time
4,,115	%SSRQC	Total number of requeues
5,,115	%SSICM	Obsolete
6,,115	%SSMSI	Medium-term scheduling interval
7,,115	%SSAJS	Average job size
1Ø,,115	%SSTQT	Total runtime given to all subclasses
11,,115	%SSEAF	Obsolete
12,,115	%SSEAT	Obsolete
13,,115	%SSRSS	Total user time since SCHED. Set class parameters
14,,115	%SSCLS	Default class for new jobs
15,,115	%SSJIL	% of time scheduler scans just-swapped in list before subqueues
16,,115	%SSSWP	Min no. of ticks swapper scans same primary subqueue
17,,115	%SSBBQ	Background batch subqueue
2Ø,,115	%SSBBS	No. of ticks between background batch swaps
21,,115	%SSIOF	% of time swapper scans PQ2 incore chain before outcore
22,,115	%SSSET	Ø If round-robin scheduling; Date/time when class runtime scheduling initiated if class scheduling

23,,115	%SSFLG	Ø if round-robin scheduling; Count of CPU classes with nonzero quota if class scheduling
24,,115	%SSCOR	Seconds to wait after swapping out a runnable job before ignoring incore protect time

.GTST2 - Second Job Status Word
GETTAB Table 117

Contents One word for each job running on the system, giving the second job status word for the job. (The first job status word is in GETTAB table Ø (.GTSTS).) The definitions of the bits in the job status word may vary from monitor release to monitor release; therefore you should not reference .GTST2 in a program that is monitor-independent.

Indexed by Job number.

Monitor Table JBTST2

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTST2]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job (use -1 for the current job).

Second job status word

.GTJTC - Job Type and Scheduler Class
GETTAB Table 120

Contents

One word for each job running on the system, giving the job type and scheduler class for the job. The bits in this word may vary from monitor to monitor; therefore you should not reference .GTJTC in a program that is monitor-independent.

Indexed by

Job number.

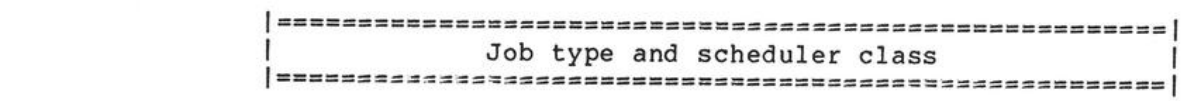
Monitor Table

JBTSCD

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTJTC]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.



.GTCQP - Scheduler Quota Percent for Class
GETTAB Table 121

Contents The scheduler class quota in percent for each class.

Indexed by Scheduler class.

Monitor Table CLSSTS

Calling Sequence

```
MOVE    ac,[XWD class,.GTCQP]
GETTAB  ac,
  error return
normal return
```

where class is the class number of the class whose quota percentage is required.

Class quota in percent for class

.GTCRT - Class Runtime Since Quota Set
GETTAB Table 123

Contents

The runtime for each class since the class quotas were set.

Indexed by

Scheduler class.

Monitor Table

SIDOFS

Calling Sequence

```
        MOVE    ac,[XWD class,.GTCRT]
        GETTAB  ac,
          error return
        normal return
```

where class is the class number of the class whose runtime is required.

Class runtime since quotas set for class

.GTSID - Special PID Table
GETTAB Table 126

Contents: A list of the defined system process identifiers (PIDs) used by the IPCF facility.

Indexed by: Item number.

Monitor Table: .GTSID

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

Word	Symbol	Map
0,,126	%SIIPC	[SYSTEM]IPCC
1,,126	%SIINF	[SYSTEM]INFO
2,,126	%SIQSR	[SYSTEM]QUASAR
3,,126	%SIMDA	Mountable device allocator
4,,126	%SITLP	Magtape labeling process
5,,126	%SIFDA	File Daemon
6,,126	%SIMDC	Mountable device coordinator (historical)
6,,126	%SITOL	Tape AVR process
7,,126	%SIACT	[SYSTEM]ACCOUNTING
10,,126	%SIOPR	Operator interface
11,,126	%SISEL	System error logger
12,,126	%SIDOL	Disk AVR process
13,,126	%SITGH	[SYSTEM]TGHA
14,,126	%SINML	DECnet NML listener
15,,126	%SIGFR	[SYSTEM]GOPHER
16,,126	%SICAT	[SYSTEM]CATALOG
17,,126	%SIMAI	[SYSTEM]MAILER

.GTENQ - ENQ./DEQ. Statistics
GETTAB Table 127

Contents — Statistics and quotas for the ENQ. and DEQ. monitor calls.

Indexed by — Item number.

Monitor Table — .EQTAB

Calling Sequence

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where **item** is one of the symbols given in the word map below.

Word	Symbol	Map
Ø,,127	%EQMSS	Maximum string size
1,,127	%EQNAQ	Number of active queues
2,,127	%EQESR	Total ENQ. since reload
3,,127	%EQDSR	Total DEQ. since reload
4,,127	%EQAPR	Number of active pooled resources
5,,127	%EQDEQ	Default ENQ. quota
6,,127	%EQMMS	Maximum pie-slice lock mask block size
7,,127	%EQMTS	Maximum lock-associated table size
1Ø,,127	%EQLTL	Minutes that unused lock data is kept
11,,127	%EQNDD	Number of deadlocks detected
12,,127	%EQNTO	Number of timeouts
13,,127	%EQMAQ	Maximum number of active queues

.GTJLT - LOGIN Time for Job
GETTAB Table 130

Contents — One word for each job running on the system, giving the date/time (in universal format) that the job logged in.

Indexed by — Job number.

Monitor Table — JBTJLT

CALLING SEQUENCE

```
MOVE    ac,[XWD jobno,.GTJLT]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

LOGIN date/time in universal format

NOTE

This table is adjusted retroactively whenever the current system date/time is changed with the SET DATE monitor command, the SET DAYTIME monitor command, or the appropriate SETUUO function. Subtracting values in this table from %CNDTM will result in the elapsed time since the job logged in.

.GTEBT - KL1Ø EBOX Time in Jiffies
GETTAB Table 131

Contents — The number of jiffies of KL1Ø EBOX time used. The definitions of the bits in this word may vary from monitor release to monitor release; therefore, you should not reference .GTEBT in a program that is monitor-independent.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTEBT]
        GETTAB  ac,
          error return
        normal return
```

Jiffies of KL1Ø EBOX time

.GTEBR - Jiffy Remainder: Mod(.GTEBT,RTUPS)
GETTAB Table 132

Contents	The remainder resulting from dividing the contents of .GTEBT by RTUPS. The definitions of the bits in this word may vary from monitor release to monitor release; therefore, you should not reference .GTEBR in a program that is monitor-independent.
Indexed by	Job number.
Monitor Table	PDB
Calling Sequence	MOVE ac,[XWD jobno,.GTEBR] GETTAB ac, error return normal return

Jiffy remainder - mod(.GTEBT,RTUPS)

.GTMBT - KL1Ø MBOX Time in Jiffies
GETTAB Table 133

Contents — The number of jiffies of KL1Ø MBOX time used. The definitions of the bits in this word may vary from monitor release to monitor release; therefore, you should not reference .GTMBT in a program that is monitor-independent.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTMBT]
        GETTAB  ac,
          error return
        normal return
```

Ø,,133	.GTMBT	Jiffies of KL1Ø MBOX time

.GTMBR - Jiffy Remainder: Mod(.GTMBT,RTUPS)
GETTAB Table 134

Contents The remainder resulting from dividing the Contents of .GTMBT by RTUPS. The definitions of the bits in this word may vary from monitor release to monitor release; therefore, you should not reference .GTMBR in a program that is monitor-independent.

Indexed by Job number.

Monitor Table PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTMBR]
GETTAB  ac,
  error return
normal return
```

Jiffy remainder - mod(.GTMBT,RTUPS)

.GTRDV - Program Run Device
GETTAB Table 135

Contents — One word for each job running on the system, giving the device the program is run from.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTRDV]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Device program is run from

NOTE

GETTAB Tables 135, 136, and 137 store the file specification of the last program run by the job. Therefore, these words change when you issue a RUN, GET, SAVE, or CORE Ø command.

.GTRDI - Program Run Directory
GETTAB Table 136

Contents — One word for each job running on the system, giving the project-programmer number (PPN) of the directory from which the job's program is being run.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTRDI]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Directory program is run from

NOTE

GETTAB Tables 135, 136, and 137 store the file specification of the last program run by the job. Therefore, these words change when you issue a RUN, GET, SAVE, or CORE Ø command.

.GTRFN - Program Run File name
GETTAB Table 137

Contents — One word for each job running on the system, giving the SIXBIT name of the file from which the job's program is being run.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTRFN]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

File program is run from

NOTE

GETTAB Tables 135, 136, and 137 store the file specification of the last program run by the job. Therefore, these words are changed when you issue a RUN, GET, SAVE or CORE Ø command.

.GTDFL - User Defaults for Job
GETTAB Table 140

Contents — One word for each job running on the system, giving the user defaults for the job as set by the SET DEFAULT command or SETUUO call.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTDFL]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

User defaults

User default bits are as follows:

Bits	Symbol	Meaning
777B8	JD.PRT	Default file protection.
9	JD.SDP	Set if user has set default protection.
12	JD.DAD	Set if LOGIN shouldn't ask about detached jobs.
777B35	JD.BUF	Number of default disk buffers.
20077B26	JD.MSK	Reserved for use by DIGITAL.

.GTNTP - Network Performance Data
GETTAB Table 141

Contents: Data for network performance analysis.

Indexed by: Item number.

Monitor Table: NETGTT

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

Word	Symbol	Map
0,,141	%NTCOR	Number of words of free space now in use
1,,141	%NTMAX	Maximum value %NTCOR has reached
2,,141	%NTAVG	Exponential average of %NTCOR (in K words)
3,,141	%NTBAD	Number of bad messages received and ignored
4,,141	%NTRTP*	Ptr to received NCL message type subtable
5,,141	%NTRMT*	Ptr to received NCL numbered message type subtable
6,,141	%NTRDL*	Ptr to received NCL data message lengths subtable
7,,141	%NTXTP*	Ptr to transmitted NCL message type subtable
10,,141	%NTXMT*	Ptr to transmitted NCL numbered message type subtable
11,,141	%NTXDL*	Ptr to transmitted NCL data message lengths subtable
12,,141	%NTBLC	PC of detection | PCB adr of last bad message
13,,141	%NTBYI	Number of input bytes processed
14,,141	%NTBYO	Number of output bytes processed

The pointers **%NTRTP, %NTRMT, %NTRDL, %NTXTP, %NTXMT,** and **%NTXDL** are of the form:

<length-1>B8+<offset>B35

where length is the maximum length of the subtable; and offset is the offset into .GTNTP of the start of the subtable.

Subtable: Received NCL Message Types
Subtable of .GTNTP

Contents — Received NCL message types.

Indexed by — Item number.

Calling Sequence

```
MOVE    ac,[%NTRTP]
GETTAB  ac,
  error return
ADDI    ac,item
HRLZS   ac
ADDI    ac,.GTNTP
GETTAB  ac,
  error return
normal return
```

where item is the number (starting with Ø) of the required entry in the subtable.

Word	Map
Ø	First received NCL message type
	. . .
last	Last received NCL message type

Subtable: Received NCL Numbered Message Types
Subtable of .GTNTP

Contents — Received NCL numbered message types.

Indexed by — Item number.

Calling Sequence

```
MOVE    ac,[%NTRMT]
GETTAB  ac,
  error return
ADDI    ac,item
HRLZS   ac
ADDI    ac,.GTNTP
GETTAB  ac,
  error return
normal return
```

where item is the number (starting with Ø) of the required entry in the subtable.

Word	Map
Ø	First received NCL numbered message type
	. . .
last	Last received NCL numbered message type

Subtable: Received NCL Data Message Lengths
Subtable of .GTNTP

Contents — Received NCL message lengths by powers of 2.

```
Ø = Ø bytes, and message too long
1 = 1 byte
2 = 2 to 3 bytes
3 = 4 to 7 bytes
. . .
n = 2(n-1) to (2*n)-1 bytes
```

Indexed by — Item number.

Calling Sequence

```
MOVE    ac,[%NTRDL]
GETTAB  ac,
  error return
ADDI    ac,item
HRLZS   ac
ADDI    ac,.GTNTP
GETTAB  ac,
  error return
normal return
```

where **item** is the number (starting with Ø) of the required entry in the subtable.

Word	Map
Ø	First received NCL data message length
	. . .
last	Last received NCL data message length

Subtable: Transmitted NCL Message Types
Subtable of .GTNTP

Contents — Transmitted NCL message types.

Indexed by — Item number.

Calling Sequence

```
MOVE    ac,[%NTXTP]
GETTAB  ac,
  error return
ADDI    ac,item
HRLZS   ac
ADDI    ac,.GTNTP
GETTAB  ac,
  error return
normal return
```

where item is the number (starting with Ø) of the required entry in the subtable.

Word	Map
Ø	First transmitted NCL message type
	. . .
last	Last transmitted NCL message type

Subtable: Transmitted NCL Numbered Message Types
Subtable of .GTNTP

Contents — Transmitted NCL numbered message types.

Indexed by — Item number.

Calling Sequence

```
        MOVE    ac,[%NTXMT]
        GETTAB  ac,
          error return
        ADDI    ac,item
        HRLZS   ac
        ADDI    ac,.GTNTP
        GETTAB  ac,
          error return
        normal return
```

where item is the number (starting with Ø) of the required entry in the subtable.

Word	Map
Ø	First transmitted NCL numbered message type
	. . .
last	Last transmitted NCL numbered message type

Subtable: Transmitted NCL Data Message Lengths
Subtable of .GTNTP

Contents

Transmitted NCL message lengths by powers of 2.

```
Ø = Ø bytes, and message too long
1 = 1 byte
2 = 2 to 3 bytes
3 = 4 to 7 bytes
. . .
n = 2**(n-1) to (2**n)-1 bytes
```

Indexed by

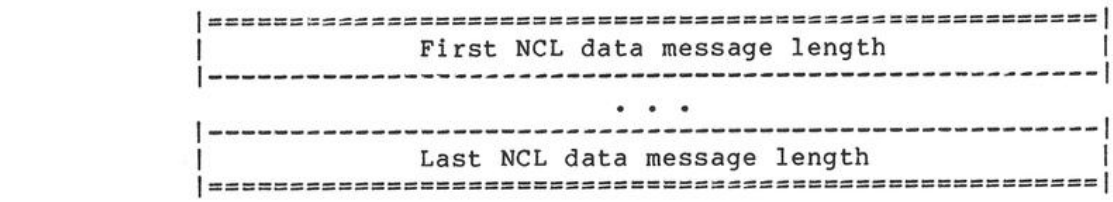
Item number.

Calling Sequence

```
MOVE    ac,[%NTXDL]
GETTAB  ac,
  error return
ADDI    ac,item
HRLZS   ac
ADDI    ac,.GTNTP
GETTAB  ac,
  error return
normal return
```

where item is the number (starting with Ø) of the required entry in the subtable.

Word — Map



.GTSPA - Scheduler Performance Data
GETTAB Table 142

Contents: Data for analysis of scheduler performance.

Indexed by: Item number.

Monitor Table: SCDPER

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where **item** is one of the symbols given in the word map below.

Word	Symbol	Map
0,,142	%SPDGS	DECtape-generated sleeps
1,,142	%SPMGS	Magtape-generated sleeps
2,,142	%SPEWS	Event-wait satisfied
3,,142	%SPTIS	Terminal input satisfied
4,,142	%SPTOS	Terminal output satisfied
5,,142	%SPPIS	Pseudo-terminal input satisfied
6,,142	%SPPOS	Pseudo-terminal output satisfied
7,,142	%SPRS1	Requeues from SS into PQ1
10,,142	%SPRW1	Requeues from wake into PQ1
11,,142	%SPRD1	Requeues from DAEMON-satisfied into PQ1
12,,142	%SPRO1	Other requeues into PQ1
13,,142	%SPQR1	PQ1 jobs that expired quantum runtime
14,,142	%SPQR2	PQ2 jobs that expired quantum runtime
15,,142	%SPQRH	HPQ jobs that expired quantum runtime
16,,142	%SPIP1	PQ1 jobs that expired in-core protect time
17,,142	%SPIP2	PQ2 jobs that expired in-core protect time
20,,142	%SPIPH	HPQ jobs that expired in-core protect time
21,,142	%SPKS1	Number of swapped in for PQ1 jobs
22,,142	%SPKS2	Number of swapped in for PQ2 jobs
23,,142	%SPKSH	Number of swapped in for HPQ jobs
24,,142	%SPNJ1	Number of PQ1 jobs swapped in

25,,142	%SPNJ2	Number of PQ2 jobs swapped in
26,,142	%SPNJH	Number of HPQ jobs swapped in
27,,142	%SPTC1	Clock ticks charged to PQ1
30,,142	%SPTC2	Clock ticks charged to PQ2
31,,142	%SPTCH	Clock ticks charged to HPQ
32,,142	%SPNRS	Number of responses for PQ1/CMQ swap-in
33,,142	%SPNTS	Total ticks of response for PQ1/CMQ swap-in
34,,142	%SPSSS	Sum of squares of PQ1/PQ2 swap-in (2-Word integer)
35,,142		Reserved
36,,142	%SPMWC	Number of measurements of wasted core
37,,142	%SPSWC	Sum of wasted core (pages)
40,,142	%SPSSC	Sum of squares of wasted core (2-word integer)

.GTVKS - Virtual Kilo-Core Ticks for Job
GETTAB Table 143

Contents One word for each job running on the system, giving the number of virtual kilo-core ticks for the job.

Indexed by Job number.

Monitor Table PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTVKS]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Virtual kilo-core ticks

.GTUUC - Monitor Calls Executed for Job
GETTAB Table 144

Contents One word for each job running on the system, giving the number of monitor calls executed for the job.

Indexed by Job number.

Monitor Table PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTUUC]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Count of monitor calls done for job

.GTRSØ - First SFD in Job Run Path
GETTAB Table 145

Contents — One word for each job running on the system, giving the first SFD in the run path for the job.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTRSØ]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

First SFD in path program was run from

.GTRS1 - Second SFD in Job Run Path
GETTAB Table 146

Contents One word for each job running on the system, giving the second SFD in the run path for the job.

Indexed by Job number.

Monitor Table PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTRS1]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Second SFD in path program was run from

.GTRS2 - Third SFD in Job Run Path
GETTAB Table 147

Contents One word for each job running on the system, giving the third SFD in the run path for the job.

Indexed by Job number.

Monitor Table PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTRS2]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Third SFD in path program was run from

.GTRS3 - Fourth SFD in Job Run Path
GETTAB Table 15Ø

Contents — One word for each job running on the system, giving the fourth SFD in the run path for the job.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTRS3]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Fourth SFD in path program was run from

.GTRS4 - Fifth SFD in Job Run Path
GETTAB Table 151

Contents One word for each job running on the system, giving the fifth SFD in the run path for the job.

Indexed by Job number.

Monitor Table PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTRS4]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Fifth SFD in path program was run from

.GTPC - User PC
GETTAB Table 152

Contents One word for each job running on the system, giving its user program counter.

Indexed by Job number.

Monitor Table JBTPC

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTPC]
        GETTAB  ac,
          error return
        normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

User PC

.GTCAP - Job Capability Word
GETTAB Table 153

Contents — One word for each job running on the system, giving the maximum privileges that can be enabled for the job.

Indexed by — Job number.

Monitor Table — PDB

Calling SEQUENCE

```
MOVE    ac,[XWD jobno,.GTCAP]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Job capability word (maximum privileges)

Capability bits are as follows:

Bits	Symbol	Privilege
Ø	JP.IPC	IPCF privilege.
1-2	JP.DPR	Highest disk priority for the job (n is in the range Ø to 3).
3	JP.MET	METER. privilege.
4	JP.POK	POKE. privilege.
5	JP.CCC	Privilege to change CPU specification with a command or a monitor call.
6-9	JP.HPQ	Highest high-priority queue available to the job (n is in the range Ø to 17 octal).
1Ø	JP.NSP	Device unspooling privilege.
11	JP.ENQ	ENQ/DEQ privilege.
12	JP.ADM	System administrator privileges.
13	JP.RTT	RTTRP privilege.
14	JP.LCK	LOCK privilege.
15	JP.TRP	TRPSET privilege.
16	JP.SPA	PEEK and SPY privilege for any core.
17	JP.SPM	PEEK and SPY privilege for monitor core.
18-35		Reserved for users.

.GTIDX - Range of Each GETTAB Table
GETTAB Table 154

Contents — The entry numbers of the minimum and maximum entries for each GETTAB table.

Indexed by — GETTAB table number.

Monitor Table — RNGTAB

Calling Sequence

```
        MOVE    ac,[XWD table,.GTIDX]
        GETTAB  ac,
          error return
        HLREM   ac,minent
        HRREM   ac,maxent
```

where **table** is the symbol for the table whose range is required; **minent** is a memory location for the minimum entry number; and **maxent** is a memory location for the maximum entry number.

Min table index (ID.MIN)	Max table index (ID.MAX)

.GTIDX contains one word for each GETTAB table. The word gives the following information:

Bit	Symbol	Meaning
17	ID.MIN	Minimum programs should do a HLRE in case negative
35	ID.MAX	Maximum programs should do a HRRE in case negative.

.GTGTB - GETTAB Immediate Using Range Table
GETTAB table 155

Contents — Data for each GETTAB table.

Indexed by — GETTAB table number.

Monitor Table — NUMTAB

Calling Sequence

```
MOVE    ac,[XWD table,.GTGTB]
GETTAB  ac,
  error return
normal return
```

where table is the mnemonic name of the table whose data is required.

GETTAB table data

The word gives the following information for each GETTAB table:

Bits	Symbol	Meaning
Ø-8	SL.MAX	If SL.TYP is 1, 2, 3, or 4, this field is the largest item number in the table. If SL.TYP is 5, this field is the index into the range table.
9-11	SL.TYP	Type of table:

Value	Symbol	Type
Ø	.SLNIC	Not included in this system.
1	.SLIXI	Indexed by item number.
2	.SLIXJ	Indexed by job number.
3	.SLIXS	Indexed by job number or segment number.
4	.SLIXP	Indexed by job number; data in PDB.
5	.SLIXR	Indexed by item number. Range may not be Ø to length -1.

12-13		Reserved for use by DIGITAL.
14-17	SL.MAC	A monitor accumulator number.
18-35	SL.ADR	If SL.TYP=1,2,3,5, this halfword contains the executive mode address of the table; if SL.TYP=4, this halfword contains the offset to PDB. If SL.TYP is 5, this halfword is the executive mode address of offset Ø into the table.

.GTTNM - Terminal Type Names
GETTAB Table 156

Contents — The SIXBIT names of those terminals that may be specifed in the SET TTY TYPE monitor command or in TRMOP. monitor call functions .TOTRM + .TOSET.

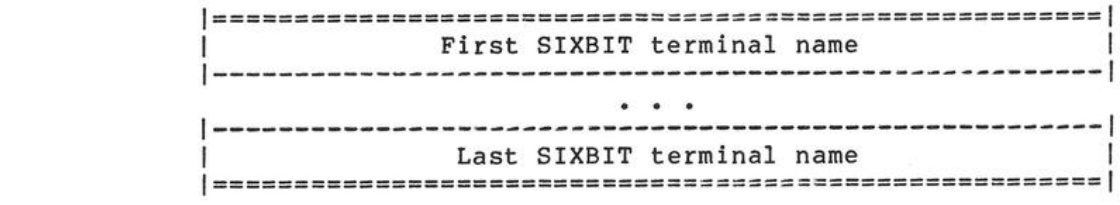
Indexed by — Terminal type number.

Monitor Table — TTTWDT

Calling Sequence

```
        MOVE    ac,[item]
        GETTAB  ac,
          error return
        normal return
```

where item is the number of the desired entry in the table.



The terminal names are defined with the TTTWDT macro in COMCON.

.GTOBI - Write-to-Operator and Batch Data
GETTAB Table 157

Contents One word for each job running on the system, giving flags defining the write-to-operator capabilities and batch stream numbers and sets for the job.

Indexed by Job number.

Monitor Table PDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTOBI]
GETTAB  ac,
  error return
normal return
```

where jobno is the number of a logged-in job. Use -1 for the current job.

Operator batch data

Operator and batch flags are as follows:

Flag	Symbol	Meaning
3B1	OB.WTO	Write-to-operator capabilities:

Value	Symbol	Meaning
Ø	.OBALL	WTO and WTOR allowed.
1	.OBNWR	No WTOR allowed; WTO only.
2	.OBNOM	No messages to operator allowed.

Flag	Symbol	Meaning
7B4	OB.OPR	Operator privileges:

Value	Symbol	Meaning
Ø	.OBNOP	No operator privileges
1	.OBSOP	System-wide privileges
2	.OBHOP	Host system privileges
3	.OBROP	Remote operator privileges

1Ø	OB.BSS	Batch stream number set.
177B17	OB.BSN	Batch stream number.

.GTDCD - CONI/DATAI Corresponding to DSB
GETTAB Table 160

Contents: Device status block subtable.

Indexed by: Item number.

Monitor Table: DVCSTS

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

CONI/DATAI corresp to device status block (see %CCDSB)

.GTNDB - Byte Pointers Into Node Data Block
GETTAB Table 161

Contents — Pointers into an NDB to facilitate the retrieval of data.

Indexed by — Item number.

Monitor Table — NDBTBL

Calling Sequence

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is one of the symbols given in the word map below.

Word	Symbol	Map
Ø,,161	%NDLEN	Length of NDB (not a byte pointer)
1,,161	%NDNXT	Address of next NDB
2,,161	%NDNNM	Node number
3,,161	%NDSNM	Address of SIXBIT station name
4,,161	%NDTIM	Timer
5,,161	%NDNGH	First neighbor entry
6,,161	%NDNGL	Last neighbor entry
7,,161	%NDNGN	Node number from %NDNGH (address = Ø)
1Ø,,161	%NDOPR	Address of OPR LDB
11,,161	%NDCTJ	Station control job number
12,,161	%NDLAR	Last ACK received
13,,161	%NDLAP	Last output message acknowledged
14,,161	%NDLMS	Last message sent
15,,161	%NDLMA	Last message number assigned
16,,161	%NDLAS	Last ACK sent
17,,161	%NDLMR	Last message received
2Ø,,161	%NDLMP	Last message processed
21,,161	%NDSDT	Address of system build date
22,,161	%NDSID	Address of system identification
23,,161	%NDMOM	Maximum number of outstanding messages allowed
24,,161	%NDDEV	First device

.GTPDB - Job PDB Word
GETTAB Table 162

Contents	Number of monitor per process pages, and monitor address of the job's PDB.
Indexed by	Job number.
Monitor Table	JBTPDB

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTPDB]
GETTAB  ac
  error return
normal return
```

Number of monitor "per process" pages	Address (monitor) of job's PDB

Where the left half of this word is divided into 6-bit fields. Bits Ø-5 are reserved for use by DIGITAL. Bits 6-11 contain the number of per-process pages to be swapped in, and bits 12-17 contain the total number of per-process pages. This word does not include section maps for non-zero sections.

.GTEQJ - ENQ./DEQ. Queue Header
GETTAB Table 163

Contents — ENQ/DEQ queue header.

Indexed by — Job number.

Monitor Table — PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTEQJ]
        GETTAB  ac
          error return
        normal return
```

ENQ/DEQ queue header

.GTDCN - Default Command Arguments
GETTAB Table 164

Contents — One word for each job running on the system, giving the SET DEFAULT monitor command argument for the job.

Indexed by — Item number.

Monitor Table — DFLTTB

Calling Sequence

```
MOVE    ac,[item]
GETTAB  ac
  error return
normal return
```

Where item is the number of the command in the list of SET DEFAULT commands. Use the HELP * monitor command to obtain the list of SET DEFAULT commands for your system.

Set default command arguments

.GTLBS - Large Buffer Size
GETTAB Table 165

Contents

Size of disk buffers as adjusted by program and SET BIGBUF monitor command. The program sets the buffer size with the SETUUO; this setting overrides any that might have been set with the monitor command. The monitor command setting takes precedence when the program is halted.

Indexed by

Job number.

Monitor table

PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTLBS]
        GETTAB  ac
          error return
        normal return
```

Set by program	Set by user command

Where the data in the left half is the buffer size as set by the program. The right half contains the buffer size as set by the monitor command SET BIGBUF.

.GTPTR - Program To Run
GETTAB Table 166

Contents

The name of the program to run. Refer to the .STPGM function of the SETUUO UUO.

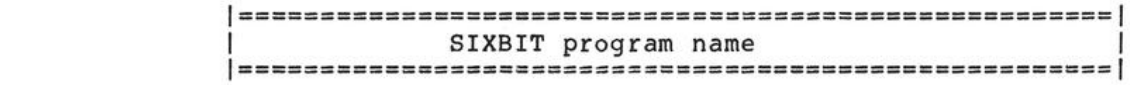
Indexed by

Job number.

Monitor Table

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTPTR]
GETTAB  ac
  error return
normal return
```



.GTSTM - Time of Last Reset
GETTAB Table 167

Contents	Time the program was last RESET.
Indexed by	Job number.
Monitor Table	PDB
Calling Sequence	MOVE ac,[XWD jobno,.GTSTM] GETTAB ac error return normal return

Universal date/time of last RESET

.GTDNT - DECnet Queue Headers
GETTAB Table 170

Contents: Pointers to tables and information about DECnet-10 Version 3.

Indexed by: Item number.

Monitor table: DCNGTB

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac
  error return
normal return
```

Word	Symbol	Map
0,,170	%DNRCH*	Queue header for circuit blocks
1,,170	%DNNPH*	Queue header for port blocks
2,,170	%DNETH*	DTE control block table
3,,170	%DNNSJ*	Pointer to session control job block for NRTSER
4,,170	%DNNCH*	Pointer to NRTSER's internal channel table
5,,170	%DNNDQ*	Queue block header for LLINK's node blocks
6,,170	%DNLOC	Obsolete
7,,170	%DNPTR	Obsolete
10,,170	%DNCHB*	Pointer to start of blocks describing DECnet's fixed-size freecore
11,,170	%DNKON*	Pointer to table of DECnet device names
12,,170	%DNNRV*	Pointer to current routing vector (indexed by node number)
13,,170	%DNOFS*	Pointer to offset from routing vector to secondary vector
14,,170	%DNRMX	Pointer to address of router maximum node number
15,,170	%DNCST	Address of byte pointer to cost
16,,170	%DNHOP	Address of byte pointer to hops
17,,170	%DNLCL*	Address of byte pointer to LOCAL bit
20,,170	%DNACT	Address of byte pointer to ACTIVE bit
21,,170	%DNNDT	Obsolete
22,,170	%DNSMX	Obsolete
23,,170	%DNACB	Address of DECnet Allocation Control Block

For additional information (format of blocks pointed to by this table), refer to code as follows:

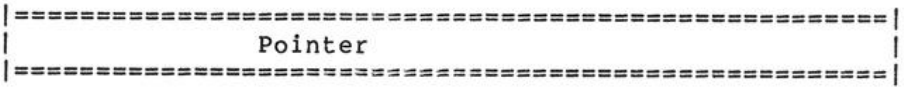
Item	Module	Label
Ø	D36PAR	BEGSTR RC
1	D36PAR	BEGSTR EL
2	DTEPRM	DTEGEN
3	Format returned is similar to that returned by GETTAB .GTSJB.	
4	NRTSER	BEGSTR NR
5	D36PAR	BEGSTR NN
1Ø	D36COM	BEGSTR CH
11	Device names are listed in ASCII.	
12	ROUTER	BEGSTR RN
13	Contains pointers to output adjacency block for this node.	
17	Set only for executor (local host) node.	

.GTSJB - DECnet Session Control Block Pointer
GETTAB Table 171

Contents	Pointer to DECnet session control job block.
Indexed by	Job number.
Monitor table	PDB

Calling Sequence

```
        MOVE    ac,[XWD jobno,.GTSJB]
        GETTAB  ac
          error return
        normal return
```



.GTNDA - NDB Address
GETTAB Table 173

Contents

Pointers into the DECnet-10 node data block address table. This table contains the address of the NDB for each node.

Indexed by

Node number.

Monitor table

.GTNDA

Calling Sequence

```
        MOVE    ac,[XWD nodeno,.GTNDA]
        GETTAB  ac
          error return
        normal return
```

Node data block address

.GTAOT - ANF-1Ø Object Translation Table
GETTAB Table 174

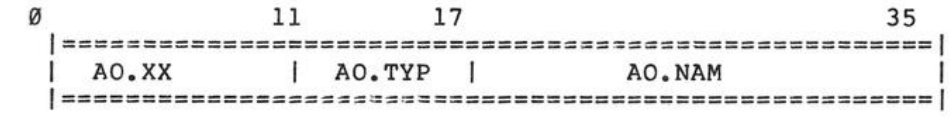
Contents — The ANF-1Ø object translation table for each DECnet object type.

Indexed by — Item number: NCL object type.

Monitor table — OBJTAB

Calling Sequence

```
MOVE    ac,[item]
GETTAB  ac
  error return
normal return
```



AO.XX, in bits Ø through 11, is reserved for use by DIGITAL.

AO.TYPE contains the device type (refer to the DEVTYP UUO).

AO.NAM contains the device name in SIXBIT.

.GTCTX - Context Table
GETTAB Table 175

Contents: Information about contexts.

Indexed by: Item number.

Monitor table

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac
  error return
normal return
```

where item is one of the symbols given in the word map below.

Word	Symbol	Map
Ø,,175	%CTJCQ	Default job context quota
1,,175	%CTJPQ	Default job saved-pages quota
2,,175	%CTSCQ	System-wide context quota
3,,175	%CTSPQ	System-wide saved-pages quota
4,,175	%CTSCU	System-wide count of contexts in use
5,,175	%CTSPU	System-wide count of currently saved pages
6,,175	%CTTCS	Total context saves done
7,,175	%CTACE	No. of times auto-push exceeded context quota
1Ø,,175	%CTAPE	No. of times auto-push exceeded saved-pages quota
11,,175	%CTPCE	No. of times a privileged program exceeded context quota
12,,175	%CTPPE	No. of times a privileged program exceeded pages-saved quota
13,,175	%CPBDM	Byte pointer to returned context directory map

.GTIMI - Job Page Count
GETTAB Table 176

Contents Number of memory pages in use by each job.

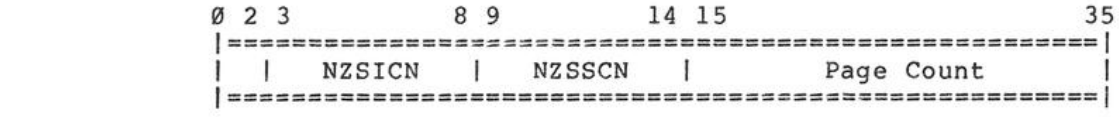
Indexed by Job number.

Monitor Table JBTIMI

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTIMI]
GETTAB  ac,
  error return
 normal return
```

where jobno is the job number of a logged-in job, or -1 for your current job.



Bit definitions:

Bits	Symbol	Contents
Ø-2		Reserved
3-8	NZSICN	Number of pages to allocate on swap-in for non-zero section maps.
9-14	NZSSCN	Number of pages currently allocated to non-zero section maps.
15-35	IMGIN	Number of physical pages in user portion of job.

.GTIMO - Swapped-Out Page Count
GETTAB Table 177

Contents Number of physical pages in swapped-out job, on disk.

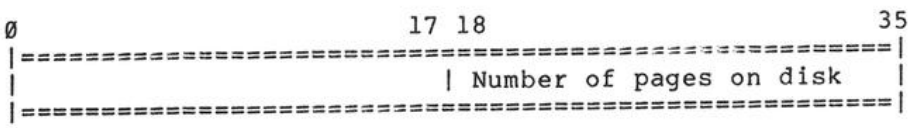
Indexed by Job number.

Monitor Table JBTIMO

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTIMO]
GETTAB  ac,
  error return
normal return
```

where jobno is the job number of a logged-in job, or -1 for the current job.



.GTDDB - I/O Wait DDB
GETTAB Table 200

Contents	Device data block for devices in I/O wait state.
Indexed by	Job number.
Monitor Table	JBTDDB
Calling Sequence	MOVE ac,[XWD jobno,.GTDDB] GETTAB ac, error return normal return

Pointer to DDB

.GTVIR - Job's Virtual Size
GETTAB Table 201

Contents Virtual size of program.

Indexed by Job number.

Monitor Table

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTVIR]
GETTAB  ac,
  error return
normal return
```

where jobno is the job number, or -1 for current job.

The virtual size is returned with bits 6-14 containing the high segment, and bits 15-35 containing the low segment.

.GTETH - Ethernet Information
GETTAB Table 202

Contents: Data about Ethernet configuration.

Indexed by: Item number.

Monitor Table

Calling Sequence:

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is the symbol representing one of the words in the following word map.

Word	Symbol	Map
0,,202	%EINEC	Number of Ethernet channels on system
1,,202	%EICHN	Address of first Ethernet channel block
2,,202	%EINEK	Number of Ethernet controllers on system
3,,202	%EIKON	Address of first Ethernet controller block
4,,202	%EISYS	Values of .ECBSYS,,.EKBSYS
5,,202	%EISTS	Values of .ECBSTS,,.EKBSTS
6,,202	%EIBYR	Total bytes received
7,,202	%EIBYX	Total bytes transmitted
10,,202	%EIDGR	Total datagrams received
11,,202	%EIDGX	Total datagrams transmitted

.GTSG2 - High Segment Section Number
GETTAB Table 203

Contents The section number of the job's high segment.

Indexed by Job number.

Monitor Table JBTSG2

Calling Sequence

```
MOVE    ac,[XWD jobno,.GTSG2]
GETTAB  ac,
  error return
normal return
```

where jobno is the job number, or -1 for the current job.

S#	Reserved

Bits	Symbol	Contents
Ø-4	SG%SCN	Section number where the high segment for the job is stored.
5-35		Reserved.

GTCCM - Site-specific Commands
GETTAB Table 204

Contents Site-specific commands, defined using the MONGEN dialog. (See the Software Installation Guide for information on defining commands with MONGEN.)

Indexed by Command number.

Monitor Table CSTTAB

Calling Sequence

```
MOVE    ac,[item]
GETTAB  ac,
  error return
normal return
```

where item is the table position of the command name.

MONGEN-defined command name
. . .
MONGEN-defined command name

The table of commands defined using the DECLARE command is searched first when you issue a command. If an exact match is not found, this table is searched next. The HELP * command displays a list of these command names.

APPENDIX A

GLOSSARY

Absolute virtual address

A fixed location in user virtual address space that cannot be relocated by the software. However, it can be translated to a physical address by the hardware. For example, locations Ø - 17 are mapped into the current AC block by the hardware. The corresponding locations in physical memory are never referenced.

AC

Refer to accumulator.

ACCESS.USR

Each user can create his own ACCESS.USR file to specify who can and cannot access his files. See Appendix C.

Access date

The date on which a file on disk was last read or written. If a file has not been read or written since it was created, the creation date and the access date are the same. The access date is kept in the Retrieval Information Block (RIB) for the file.

Access privileges

Attributes of a file that specify the class of users allowed to access the file and the type of access they are allowed.

Access table

A table stored in the monitor that reflects the status of a file. There is one access table for each file that is open for reading or writing, in addition to those files that were recently closed. This information is kept in the monitor in order to decrease the time needed to access the files.

Accumulator

One of the registers and associated equipment in the arithmetic unit in which data can be placed while it is being examined or manipulated (for example, the 16 high-speed registers at address locations Ø through 17).

Active search list

An ordered list of file structures established for each job running on the system. This list is used to translate references to the generic device DSK into the actual file structures to be used. This means if a user reads a file on device DSK, the system will look for the file on structures contained in the active search list. The active search list is separated from the passive search list by the FENCE. The SETSRC program can be used to alter the contents of the job's active search list.

Address

1. An identification represented by a name, label, or number for a register, a location in storage, or any other data source or destination in memory or on an addressable storage device.

2. The part of an instruction that specifies the location of an operand of the instruction (also called "effective address").

?ADDRESS CHECK Error

This error can occur when a dump mode I/O command list or LOOKUP/ENTER/RENAME block is not in your low segment. It can also occur when an invalid address is encountered during any I/O UUO processing.

Address mapping

The assignment of user virtual address space to the physical address space in computer memory. This is automatically performed by the TOPS-1Ø monitor and is transparent to user programs.

ALL search list

The list of all structures currently known to the system and physically mounted. This list is the output from the SYSSTR monitor call.

Alphanumeric

The set of characters that includes the letters of the alphabet (A through Z), and the numerals (Ø through 9).

Arithmetic unit

The portion of the central processing unit in which arithmetic and logical operations are performed.

ASCII code

American Standard Code for Information Interchange. A 7-bit code in which textual information is recorded. The ASCII code can represent 128 distinct characters. These characters are the upper and lower case letters, numbers, common punctuation marks, and special control characters.

Assembly language

The machine-oriented symbolic programming language specific to a given computing system. The assembly language for TOPS-1Ø is MACRO.

ASCIZ

A 7-bit ASCII string terminated by a zero byte. The string is word aligned (left justified) unless specified by a byte pointer. The zero byte is not included in the string length, but must be present.

Assigning a device

Associating an I/O device to the user's job either for the duration of the job or until the user relinquishes it.

Associated variable

Returned in the AC on a normal return for the IPCFR monitor call and returned to the status word when a IPCF-related software interrupt is generated.

Associative memory

High-speed, 32-word memory that is used by the KI1Ø processor to provide address mapping information for the operating system and user programs.

Asynchronous

1. Pertaining to the procedure by which the hardware can begin a second operation before waiting for the first operation to be completed.

2. Pertaining to the method of data transmission in which each character is sent with its own synchronizing information and no fixed time between consecutive characters.

Backspace

To move back the logical position in a file or on a line according to a prescribed format. For example, magnetic tape units can be backspaced over a file or a record. Some terminals allow backspacing in order to permit over-printing.

Bad Allocation Table (BAT) block

A block written on every disk unit to enumerate the bad regions of consecutive bad blocks on that unit so that they are not reused. The BAT blocks appear in the HOME.SYS file.

BADBLK.SYS

The file that contains all bad blocks. It may be read but not deleted and is useful for testing error recovery.

Baud

A unit of signalling speed equal to the number of discrete conditions or signal events per second.

Binary code

A code that uses two distinct characters only; usually these characters are Ø and 1.

Bit

A binary digit (that is, Ø or 1). Usually refers to the smallest unit of information storage, which can be on or off. A word on TOPS-1Ø has 36 bits.

Block

A set of records, words, characters, or digits handled as a unit. On the TOPS-1Ø, a 128-word unit of disk storage allocated by hardware and software; 128 words are always written, adding zeroes as necessary, although fewer than 128 words can be read.

Bootstrap

A routine designed to bring itself into a desired state by means of its own action (for example, a machine routine whose first instructions are sufficient to bring the rest of itself into the computer from an input device).

Breakpoint

A location at which program operation is suspended in order to examine partial results. Breakpoints are used in the debugging process.

Break Set

The set of characters used by the monitor to determine the end of a command line typed on the terminal. The default terminal break set includes <ESC> and <RET>, but your program can be enabled to recognize any set of characters as break characters.

Buffer

A device or area used to temporarily hold information being transmitted between two processes, such as external and internal storage devices or I/O devices and internal high-speed storage. A buffer is often a special register or a designated area of internal storage.

Buffer pointer

A position indicator that is located between two characters in a buffer, before the first character in the buffer, or after the last character in the buffer, to indicate the position at which the next operation will begin.

Buffer Ring

A ring of buffers used to allow a program to perform I/O efficiently. In a buffer ring, the program can execute instructions while the monitor is filling buffers.

Bug

A deficiency in a program that causes it to execute incorrectly, or a mistake made by a person when writing a program or designing hardware.

Byte

Any contiguous set of bits within a word.

Call

(verb) To transfer control to a specified closed subroutine.

Call

(noun) An instruction used to pass control to another program, such as a "monitor call."

Caller

The program or routine which calls another program or routine. The person who invoked the caller is referred to as the user. As an example, the user types commands to SCAN which stores them in core. The caller then calls WILD to study this block and select files. Thus the user has specified the request but the caller actually invoked WILD.

Calling sequence

A specified arrangement of instructions, pointers, and data necessary to pass parameters and control to, and return from, a given subroutine.

Card

A punched card with 80 vertical columns, each containing 12 vertical rows. Also, a unit of computer circuitry.

Card column

One of the vertical lines of 12 punching positions on a punched card.

Card field

A fixed number of consecutive card columns assigned to a unit of information.

Card hopper

The tray on a card processing machine that holds the cards to be processed and makes them available to the card feed mechanism.

Card row

One of the horizontal lines of punching positions on a punched card. A row is 80 columns long.

Card stacker

The tray on a card processing machine that receives processed cards.

Carriage return

1. The operation that prepares for the next character to be printed or displayed at the first position on the same (current) line on a terminal or line printer.

2. The ASCII character with the octal code Ø15.

CDP

The generic device name for the card punch device.

CDR

The generic device name for the card reader device.

Central processing unit (CPU)

The portion of the computer that contains the arithmetic and logical facilities, control circuits, and basic I/O and memory interfaces. There can be more than one CPU in a computing system.

Central site

The location of the central computer in a computer network. This term is used in conjunction with remote communications to mean the location of the TOPS-1Ø central processor as distinguished from the location of the remote station. Refer to "Host."

CFP

See Compressed File Pointer.

Channel

1. A path along which signals can be sent, such as an output channel.

2. A portion of TOPS-1Ø that can overlap I/O transmission while computations proceed simultaneously (such as a data channel).

Character

One symbol of a set of elementary symbols such as those corresponding to the keys on a typewriter. The characters usually include the decimal digits Ø through 9, the letters A through Z, punctuation marks, operation symbols, and any other special symbols which a computer may read, store, or write.

Clear

To erase the contents of a location, a block of memory, or a mass storage device by replacing the contents with blanks or zeroes.

Cluster

A single- or multi-block unit of disk storage assignment. The number of blocks per cluster is a parameter of each file structure.

Command

The part of an instruction that causes the computer to execute a specified operation.

Command List

Specifies the memory area to be read or written when performing dump I/O.

Communication link

The physical means of connecting one device to another for the purpose of transmitting and receiving data.

Compressed file pointer

An 18-bit pointer to the unit within the file structure and to the first super-cluster of the file. This pointer is stored in the UFD for each file in that UFD. It points to the retrieval information block, which contains the information necessary to access the desired file.

Computer operator

A person who manipulates the controls of a computer and performs all functions that are required to maintain and operate the system, such as adjusting parameters which affect the operation of the system, loading tape and disk drives, placing cards in the input hopper, and removing listings from the line printer.

Computer program

A series of instructions or statements prepared in order to achieve a specific result and intended for execution by a computer. A program can be in either the binary form in which it can be directly executed by a computer or a symbolic form that must be compiled and/or assembled before it can be executed.

Concatenation

The joining of two strings of characters to produce a longer string, often used to create symbols in macro definitions, or combining two or more files into one larger file.

Concealed mode

The user submode on the KI or KL processor that may contain proprietary coding. Sections of proprietary code are hardware-protected from access by public mode programs except through predefined entry points (PORTAL instructions).

Console

The part of a computer used by the operator to determine the status of, and to control the operation of, the computer (CTY). Also informally used to refer to the user's terminal.

Context switching

The saving of sufficient hardware and software information for a process so that it may be continued at a later time, and the restoring of similar information relevant to another process. A common use of context switching is the temporary suspension of a user program so that the monitor can execute a function.

Continued directory

The collection of all directories with a particular name and path on all file structures in the job's search list.

Continued MFD

The MFDs on all file structures in the job's search list.

Continued SFD

The SFDs on all file structures in the job's search list which have the same name and path.

Continued UFD

The UFDs for the same project-programmer number on all file structures in the job's search list.

Control character

A character whose purpose is to control an action, such as line spacing on the line printer, rather than to pass data to a program. An ASCII control character has an octal representation of Ø-37. It is typed by holding down the CTRL key on the terminal while striking a character key. It can be punched on a card using the multi-punch key.

Controller

The device or portion of a device which controls the operation of connected units. Some controllers can initiate simultaneous positioning commands to some of its units and can then perform a transfer for one of its units.

Controller class name

All file structures residing on all controllers of a given type.

Controller name

All file structures residing on a specific controller.

Core

Physical memory.

Core storage

A storage device normally used for main memory in a computer.

CORMAX

The maximum amount of core memory that a single job can use at one time. This value can range from MINMAX to total user core.

CORMIN

The amount of core guaranteed to unlocked jobs. Locked jobs are limited to total user core minus CORMIN.

Counter

A device, such as a register or storage location, used to represent the number of occurrences of a certain event. Refer to program counter.

CPU

See central processing unit.

Create

To open, write, and close a file for the first time. Only one user at a time can create a file with a given name and extension in the same directory or subdirectory of a file structure.

CRLF

Carriage-return/line-feed sequence. A "free CRLF" can be enabled for terminal output. An "automatic CRLF" can be enabled for terminal input.

CTY

Console terminal used to load, control, and debug the system.

CUSP

A Commonly Used System Program, such as LOGIN, that works closely with the monitor to perform system functions.

Customer

A customer of Digital Equipment Corporation who has purchased a DECsystem-10 as distinguished from a user at a terminal who may be purchasing time from a customer.

Cylinder

The hardware-defined region of consecutive logical disk blocks which can be read or written without repositioning. A cylinder usually consists of tracks in the same physical position on different disk surfaces.

DAEMON

A program for writing all or parts of a job's core area and associated monitor tables onto disk.

Data

A general term used to denote any or all information (facts, numbers, letters, and symbols that refer to or describe an object, idea, condition, or situation). It represents basic elements of information which can be processed by a computer.

Data channel

The device which passes data between the memory system and a controller.

DDB

A device data block.

DDT

The Dynamic Debugging Technique program used for on-line checkout, testing, examination, modification, and program composition of object programs. Various types of DDT programs are available, such as DDT11 for debugging PDP-11 remote stations and the RSX-2ØF front end, and EDDT for debugging the monitor.

Deadlock

The situation where two or more jobs are waiting for each other to complete use of a resource, but neither of the jobs can obtain a lock on the resource it needs for completion.

Debug

To detect, locate, and correct any mistakes in a computer program.

DECtape

A convenient, pocket-sized reel of random access magnetic tape developed by Digital Equipment Corporation. A standard reel consists of 578 (decimal) blocks, each capable of storing 128 (decimal) words of data.

Default directory

The directory that monitor searches when a disk directory has not been specified by the user. Typically, this is the UFD (user-file directory) corresponding to the user's project-programmer number but it may be another UFD or a SFD (sub-file directory).

Demand paging

The operation in which all pages of a program are not resident in core during execution. References to non-resident pages initiate the actions of moving in additional pages or replacing inactive pages.

Dequeue

The function of releasing or relinquishing ownership of a resource. Refer to "Enqueue."

Device substitution

Your program can be written for one device, and, before your program is executed, you can substitute another device by using the ASSIGN command.

Device routines

Routines that perform I/O for specific hardware devices. They usually translate logical block numbers to physical addresses for those devices that associate addresses with data. These routines also handle error recovery and ensure ease of programming through device independence.

Diagnostic

Pertaining to the detection and isolation of a hardware malfunction or bug. A program which tests the hardware and isolates any faults.

Digit

A symbol that represents one of the nonnegative integers smaller than the base of the system. For example, in the decimal system, a digit is one of the characters from Ø to 9.

Direct address

An address that specifies the location of an operand. Contrast with indirect address.

Directory

A file that contains the names and pointers to other files on the device. The MFD, UFDs, and SFDs are directory files. The MFD is the directory containing all the UFDs. The UFD is the directory containing the files existing in a given project-programmer number area. The SFD is a directory pointed to by a UFD or a higher-level SFD. The SFDs exist as files under the UFD. The DIRECT monitor command lists a directory.

Directory device

A storage retrieval device, such as disk, DECtape, or labelled magnetic tape, that contains information describing the names of files and the layout of stored data (programs and other files). A directory device is randomly accessible.

Directory path

The ordered list of directory names, starting with a UFD name, which uniquely specifies a directory without regard to a file structure. A file structure name, a path, and a file name and extension are needed to uniquely identify a file in the system.

Directory specification

The way that the user specifies the directory to the SCAN program. It is always typed within square brackets. Fields are separated by commas. The first two fields are the project and programmer numbers which are octal. They specify the particular UFD. Additional fields are SFDs in order from the UFD down. The following notations are allowed:

[PPN]	UFD
[PPN,sfd,...sfd]	full path to directory
[-]	default directory
[,]	your UFD

DIS

Display light pen.

Disk

A form of mass storage device in which information is stored on rotating magnetic platters. A disk is a directory device.

Disk address

All references to disk addresses refer to a logical or relative address; they do not refer to any physical addressing scheme. The basic addressable unit is a 200 (octal) 36-bit word block.

Dismounting a file structure

Deleting a file structure from a user's active search list by using the DISMOUNT command. It does not necessarily imply physical removal of the file structure from the system.

Dormant file structure

A file structure that is physically mounted but has no current users; that is, when the mount count is zero.

Dormant segment

A sharable high segment kept on a swapping space, and possibly in core, which is in no user's addressing space.

Double precision

The use of two computer words to represent a number.

DSK

The generic device name for disk-like devices. The generic device DSK is translated by the system into actual file structure names which are defined for each job by the file structure search list.

DSKLST

A program that gives the status and statistics of all user disk files at a given time.

DSKRAT

A damage assessment program that scans a file structure and reports any inconsistencies detected.

DTA

The generic device name for DECtape.

Dump

A listing of variables and their values, or a listing of the values of locations in core.

Echoing

A method of data transmission in which the received data is returned to the sending end. Usually used in discussions of terminal I/O.

Edit

To modify the content or format of a program or data file (as to insert or delete characters).

Effective address

The actual address used; that is, the specified address as modified by any indexing or indirect addressing rules.

ENQ/DEQ

A facility that ensures that resources such as files are shared correctly.

Enqueue

The function of storing requests for ownership of some limited resource in lists or queues until the requests can be granted.

Entry point

A point in a subroutine to which control is transferred when that subroutine is called.

Error Interception

When an error occurs, the monitor intercepts control of the program, examines location .JBINT, and transfers control to an error intercepting routine.

Ersatz device name

A device name that may not refer to an actual device, but represents a UFD. Ersatz device names are a specific set of such logical names, recognized by the monitor.

Execute

To interpret an instruction or command and perform the indicated operation(s).

Executive mode

A central processor mode characterized by the lack of memory protection and relocation and by the normal execution of all defined operation codes.

Extended file

A file that has more than one RIB in which to record the retrieval pointers.

Extended argument block for LOOKUP, ENTER, and RENAME

A detailed argument block for each of these calls that describes information from the file's RIB.

Extended RIBs

Additional retrieval information blocks (RIBs) required when the retrieval pointers in a file overflow the prime RIB.

External symbol

A global symbol which is referenced in one module but defined in another module. The EXTERN statement in MACRO-1Ø is used to declare a symbol to be external. A subroutine name referenced in a CALL statement in a FORTRAN module is automatically declared external.

FENCE

The boundary between the active and passive search lists. This distinction is maintained by the SETSRC program.

File

An ordered collection of characters or 36-bit words containing computer instructions and/or data. A file is stored on a device, such as disk or magnetic tape, and can be of any length, limited only by the available space on the device and the user's maximum space allotment on that device. A file is uniquely identified in the system by a file structure name or directory name, a directory path, and a file name and extension.

File Daemon

The monitor calls the File Daemon (if F%FDAE=1) every time that someone tries to access a file that has a 4, 5, 6, or 7 code in the owner's protection field and the access fails due to a protection error. Refer to Appendix C.

File directory

See Directory.

File extension

One to three alphanumeric characters usually chosen by the program to describe the class of information in a file. The extension is separated from the file name by a period.

File name

One to six alphanumeric characters chosen by the user to identify a file.

File specification

A list of identifiers which uniquely specifies a particular file. A complete file specification consists of: the name of the device on which the file is stored, the name of the file including its extension, and the name of the directory in which the file is contained. File specifications are ignored for non-file-oriented devices, such as cards and paper tape. Your program specifies a file name and directory name in a LOOKUP, ENTER, RENAME, or FILOP. monitor call.

File specification area

The area of core in which SCAN stores the result of scanning the user's file specification. This instructs WILD as to the files to select.

File status word

See I/O status word.

File structure

The logical arrangement of blocks (which are normally 128 words long) on one or more disk units of the same type to form a collection of files and directories.

File structure abbreviation

An abbreviation of one or more file structures. This refers to all those structures in the ALL search list whose names match the abbreviation. For example, if there were structures "PRIV" and "PACK," "P" would refer to both structures but "PR" would mean just "PRIV."

File structure owner

The user whose project-programmer number is associated with the file structure in the administrative file STRLST.SYS. The CATLOG program is used to enter or delete this project-programmer number or any of the other information that is contained in an STRLST.SYS entry.

Flag

An indicator that signals the occurrence of some condition, such as the end of a word.

Fragmentation

The state existing when swapped segments cannot be allocated in one contiguous set of blocks on the swapping space and therefore must be allocated in separate sections.

Full-SCNSER PTY

A pseudo-terminal (PTY) that contains the full terminal characteristic set, allowing echoing to the controlled job and a full break set. Refer to "PTY."

Fullword

A contiguous sequence of bits or characters that comprises a single computer word, or describing a word that can be referred to as a single unit. On TOPS-1Ø, a word is 36 bits long.

Funny space

Refer to "Per-process space."

Generic device name

The name of a class of physical units. This abbreviation is usually three characters. As an example, DTA is the generic name for DECtapes, and DTAØ, DTA1, and so forth, are specific unit names.

Global symbol

A symbol that is accessible to modules other than the one in which it is defined. The value of a global symbol is placed in the loader's global symbol table when the module containing the symbol definition is loaded.

Group

A contiguous set of disk clusters allocated as a single unit of storage and described by a single retrieval pointer.

Halfword

A contiguous sequence of bits or characters which comprises half of a computer word and may be addressed as a unit. On TOPS-10, bits 0 through 17 comprise the left half word and bits 18 through 35, the right half word. Each half word is 18 bits long.

Hardware

Physical equipment of the computer (such as magnetic, mechanical, and electronic devices), as contrasted with the computer program (software) or method of use.

High segment

That portion of the user's addressing space, usually beginning at virtual address 400000, which generally is used to contain pure code that can be shared by other users. This segment is usually write-protected in order to protect its contents. The user can place information into a high segment with the TWOSEG pseudo-op in MACRO-10. Higher-level languages, such as COBOL and FORTRAN, also have provisions for loading code into the high segment.

HOME.SYS

The file that contains a number of special blocks for system use. These blocks are the home blocks, the BAT blocks, the ISW blocks, and block zero.

HOME block

The block written twice on every unit that identifies the file structure the unit belongs to and its position on the file structure. This block specifies all the parameters of the file structure along with the location of the MFD. The home block appears in the HOME.SYS file.

Host

A processor or system in a computer network that processes and executes user commands and programs. For example, this term is used to distinguish a DECsystem-10 from a PDP-11 remote station.

I/O

An abbreviation for input or output, or both. Pertaining to all equipment and activity that transfers information into or out of a computer.

I/O status word

Sometimes called "file status word," this word contains I/O error bits and data modes for the device that is OPEN for I/O.

ICPT

In-core protect time. Minimum amount of time that a job is guaranteed to reside in core.

Idle segment

A sharable segment that is referenced by one or more swapped-out jobs, but not by any jobs currently in core.

Idle time

That part of uptime in which no job could run because all jobs were HALTed or waiting for some external action such as I/O.

Immediate mode addressing

The interpretation of certain instructions in which the effective address of the instruction is used as the value of an operand (rather than the address of an operand).

Impure code

The code that is modified by a program.

Indexed address

An address that is formed by adding the content of an index register to the content of an address field prior to or during the execution of a computer instruction.

Index register

A register whose contents may be added to the operand address prior to or during the execution of a computer instruction. Accumulators 1 through 17 (octal) may be used as index registers. (Accumulator Ø may not be used as an index register.)

Indirect address

An address which indicates a storage location where the address of the referenced operand (or another indirect address) is to be found. Contrast with direct address.

Initialize

To set counters, switches, or addresses to zero or other starting values at prescribed points in the execution of a computer routine, particularly in preparation for reexecution of a sequence of code.

Initialize a device

A device must be initialized on a software I/O channel to do I/O.

Input

1. Pertaining to a device, process, or channel involved in the acquisition of data.

2. Information that is read by a computer.

Instruction

A bit pattern which, when interpreted by the computer, directs the computer to execute a specific operation. An instruction generally contains the values or locations of its operands.

Interleaving

The process of configuring the memory addressing so that consecutive addresses are not stored in the same memory module. This allows the possibility of increasing memory speed by overlapping part of the operation of different memory modules.

Internal date-time format

The format for storing a combined date and time internally. This format is used by SCAN and other programs. It has the property that it is one 35-bit ("integer") quantity such that the difference between two points in time in internal format is constant if they are a constant time apart. The format is:

In the left halfword, the number of days since November 17, 1858.

In the right half a fraction of the day since midnight.

This results in a resolution of approximately one third of a second. The date field will not be exceeded until 2217 A.D. (November 17, 1858, is the origin date used by the Smithsonian calendar. This calendar is in use by several computer systems and many astrophysics programs. Its origin was selected because November 18, 1858 was the date of the first "Harvard Plates," which were the first accurate astronomical photographs. Hence, this date standard minimizes the date field while leaving all astrophysical measurements as positive dates).

Internal storage

Addressable high speed storage directly controlled by the central processing unit.

Internal symbol

A global symbol located in the module in which it is defined. In a MACRO-10 program, a symbol is declared internal with the INTERN or ENTRY pseudo-op. These pseudo-ops generate a global definition which is used to satisfy all global requests for the symbol.

Interrupt

A signal which, when activated, causes a transfer of control to a specific location in memory thereby breaking the normal flow of control of the routine being executed. An interrupt is caused by an external event such as a done condition in a peripheral. It is distinguished from a trap which is caused by the execution of a processor instruction.

IPCF

The Inter-Process Communications Facility, which allows communication among jobs and system processes.

JACCT program

A program running with the JACCT privilege bit. This is set by the monitor for special system programs such as LOGIN. This bit gives the caller full file access; that is, it allows the caller to LOOKUP and read any file in the system regardless of the file's protection code.

Jiffy

A period of time equal to 1/60 of a second (for 60 Hz power) or 1/50 of a second (for 50 Hz power), used to count CPU cycles. Synonym for "tick."

Job

The entire sequence of steps from beginning to end, that the user initiates from his interactive terminal or batch control file or that the operator initiates from his operator's console. Thus, it is a specific group of steps presented as a unit of work for the computer. A job usually includes all necessary computer programs, files, linkages and instructions to the operating system.

Job Data Area (JOBDAT)

The first 140 octal locations of a user's virtual address space. This area provides storage for certain data items used by both the monitor and the user's program.

Job search list

The ordered list of file structures for your job that are searched automatically when the generic device name DSK is specified or implied in the file specification.

K

A symbol used to represent 1024 (2000 octal); for example, 32K is equivalent to 32,768.

Kernel mode

The executive submode in the processor under which I/O and system-wide functions operate. Code executed in kernel mode can access and alter all of memory.

KL-paging

The method of paging memory used by the hardware of the KL processor to extend the virtual memory space of the program to a multiple of 256K. Refer to the Processor Reference Manual.

Label

A symbolic name used to identify a statement or an item of data in a program.

Leader

A blank section of tape at the beginning of a reel of magnetic tape or the beginning or end of a stack of paper tape.

Library

A file containing one or more relocatable binary modules which may be loaded in Library Search Mode. MAKLIB is a system utility program which enables users to merge and edit a collection of relocatable binary modules into a library file. PIP can also be used to merge relocatable binary modules into a library, but it has no facilities for editing libraries.

Library search mode

The mode in which a module (one of many in a library) is loaded only if one or more of its declared entry points satisfy an unresolved global request.

Library search symbol (entry symbol)

A list of symbols that are matched against unresolved symbols in order to load the appropriate modules. This list is used only in library search mode. A library search symbol is defined by an ENTRY statement in MACRO-1Ø.

Line

A string of characters terminated with a vertical tab, form feed, or line feed. The terminator belongs to the line that it terminates.

Line feed

1. The operation that prepares for the next character to be printed or displayed at the same (current) position on the next line on a terminal or line printer.

2. The ASCII character with the octal code Ø12.

Line printer

An electro-mechanical computer peripheral which accepts a line of characters from the computer at a high speed and then prints the entire line in one operation.

Line

To combine independently--translated modules into one module in which all relocation of addresses has been performed relative to that module and all external references to symbols have been resolved based on the definition of internal symbols.

Load

To produce a core image and/or a saved file from one or more relocatable binary files (REL files) by transforming relocatable addresses to absolute addresses. This operation is not to be confused with the GET operation, which initializes a core image from a saved file (refer to GET).

Local peripherals

The I/O devices and other data processing equipment and memory, excluding the central processor and memory, located at the central site.

Local symbol

A symbol known only to the module in which it is defined. Because it is not accessible to other modules, the same symbol name with different values can appear in more than one module. These modules can be loaded and executed together without conflict. Local symbols are primarily used when debugging modules; symbol conflicts between different modules are resolved by mechanisms in the debugging program.

Lock

An association between a job and a resource.

Locked job

A job in core that is never a candidate for swapping or shuffling.

Logged-in UFD

The UFD that corresponds to the project-programmer number under which the user is logged in.

Logical device name

An alphanumeric name you choose to represent a physical device. This name can be used synonymously with the physical device name in all references to the device. Logical device names allow device independence in that the most convenient physical device can then be associated with the logical name at run time. Logical names take precedence over physical names. With the exception of disks, only one logical name can be associated with a physical name.

Logical record

A collection of related items stored together. It is possible to have:

1. Several logical records stored in a single physical record.
2. Each logical record stored in a single physical record.
3. Each logical record occupy one or more physical records.
4. Logical records span several physical records, and at the same time, have more than one logical record in a single physical record.

LOGIN

The system program by which the system users gain access to the computing system.

Lost time

The time that the null job was running, while at least one other job wanted to run (was not waiting for a device) but could not because one of the following was true:

1. The job was being swapped out.
2. The job was being swapped in.
3. The job was on disk waiting to be swapped in.
4. The job was momentarily stopped so devices could become inactive in order to shuffle jobs in core.

Low segment

The segment of user virtual address space beginning at zero. It contains the Job Data Area and I/O buffers. The length of the low segment is stored in location .JBREL of the Job Data Area. When writing two-segment programs, it is advisable to place data locations and impure code in the low segment.

LPT

The generic device name for line printers.

MACRO

The symbolic assembly program on the TOPS-1Ø.

Macro

A portion of code that is substituted for its name whenever its name is invoked.

Magnetic tape

A tape with a magnetic surface on which data can be stored by magnetizing selective portions of the surface.

MAINT.SYS

The area of the disk reserved for maintenance use only.

Mask

1. A combination of bits that is used to control the retention or elimination of portions of any word, character, or byte in memory.
2. On half-duplex circuits, the characters typed on the terminal to make the password unreadable.

Master/slave system

A specific type of multiprocessing system involving two processors where one processor has a more important role than the other.

Master file directory (MFD)

The file created when the disk is refreshed, which contains the names of all user file directories including itself.

Meddling

The action of attempting to modify code in a sharable high segment.

Memory cycle overlap

The hardware feature that allows a second memory reference to be made before data from the first reference has been received by the processor.

Memory protection

A scheme for preventing read and/or write access to certain areas of storage.

Metering

A technique used to perform performance analysis.

MINMAX

The minimum value for CORMAX.

Mnemonic symbol

A symbolic representation for a computer instruction or other numeric item. All defined monitor symbols are listed in UUOSYM.MAC.

Modes

The data modes that can be used when performing I/O.

Module

The smallest entity that can be loaded by the loader. It is composed of a collection of control sections. In MACRO-1Ø, the code between the TITLE and END statements represents a module. In FORTRAN, the code between the first statement and the END statement is a module. In COBOL, the code between the IDENTIFICATION DIVISION statement and the last statement is a module.

Module origin

The first location occupied by the module in user virtual address space.

MONGEN

The monitor generator dialogue program that enables the system programmer to define the hardware configuration of his individual installation and the set of software options that he wishes to select for his system.

Monitor

The collection of programs which schedules and controls the operation of user and system programs, performs overlapped I/O, provides context switching, and allocates resources so that the computer's time is efficiently used. Also called the operating system.

Monitor command

An instruction to the monitor to perform an operation.

Mount count

The count of the number of jobs that have a certain file structure in their active or passive search lists (plus 1 if the structure is in the system search list).

Mounting a device

To request both the system to assign an I/O unit and the operator to physically place the specified medium (for example, magnetic tape reel or disk pack) on that unit.

Mounting a file structure

The process of adding a file structure to a search list. If the file structure is not already defined and mounted, this action is requested of the operator.

MPX-controlled device

A device connected to a multiplexed channel.

MPXable Devices

The devices that can be connected to an MPX channel are line printers, terminals, paper-tape punches, remote data entry devices, and pseudo-TTYs.

MTA

The generic device name referring to a magnetic tape unit.

Multiprocessing

Simultaneous execution of two or more computer programs by two or more processors.

Multiprogramming

A technique that allows scheduling in such a way that more than one job is in an executable state at any one time. TOPS-10 is a multiprogramming operating system in which there are two or more independent instruction streams that are simultaneously active but are not necessarily simultaneously executed.

Nesting

To include a loop, a macro definition, a routine, or a block of data within a larger loop, macro definition, routine, or block of data.

No-op

An instruction that specifically instructs the computer to do nothing. The next instruction in sequence is then executed.

Non-blocking I/O

In buffered modes, the program does not block while waiting for a buffer to be filled or emptied.

Non-directory device

A device, such as unlabelled magnetic tape or paper tape, that does not contain a file describing the names and layout of data files.

Non-sharable segment

A segment for which each user has his own copy. This segment can be created by a CORE or REMAP UUO or initialized from a file.

Octal

1. Pertaining to a characteristic or property in which there are eight possibilities.

2. Pertaining to the number system with a radix of eight.

Offset

The number of locations or bytes relative to the base of an array, string, or block. For example, the number of locations relative to zero that a Control Section must be moved before it can be executed.

ONCE-only time

The time at which the operator can change a number of monitor parameters when the monitor is started up. This is done prior to scheduling any jobs, when the ONCE program is run at system startup.

Operand

1. The data that is accessed when an operation (either a machine instruction or a higher level operation) is executed.

2. The symbolic expression representing that data or the location in which that data is stored, for example, the input data or arguments of a pseudo-op or macro instruction.

Operating system

The collection of programs that administer the operation of the computing system by scheduling and controlling the operation of user and system programs, performing I/O and various utility functions, and allocating resources for efficient use of the hardware.

OPR

The operator's control program to monitor and maintain the GALAXY batch and spooling system (Version 4.1 and later), and the DECnet network environment (Version 3 or later).

OPSER

The OPerator SERvice program that allows multiple job control from a single terminal.

Output

1. Pertaining to a device, process, or channel involved in an output process (that is, the process of transferring data from memory to a peripheral device).

2. The data that has been transferred from memory to a medium readable by a person (such as line printer listings).

Pack

1. To compress data in memory or on a peripheral storage device by taking advantage of known characteristics of the data so that the original data can be recovered.

2. A disk pack (that is, a removable set of disks mounted on a common shaft).

Packet

A group of words or block of data passed from one program to anothers cooperating program. This is accomplished through use of the IPCF facility or through task-to-task network communication.

Pack-ID

A 6-character SIXBIT name or number used to uniquely identify a disk pack.

Page

1. Any number of lines terminated with a form feed character.

2. The smallest mappable unit of core storage. On the KL10 processor, a page is 512 continuous words in core starting on boundaries which are even multiples of 512. It is also the smallest allocatable unit of memory. KL10 operations allow programs to be composed of up to 512 pages scattered within core.

3. To selectively remove parts of a user's program from core memory.

Paper tape

A tape on which data is represented by specific patterns of punched holes.

Parameter

A variable that is given a constant value for a specific purpose or process, for example, an input argument to a subroutine or command, or a value specifically assigned to a symbol in an assembly in order to control exactly what code is assembled.

Parity bit

A binary digit attached to a group of bits to make the sum of all the bits always odd (for odd parity) or always even (for even parity).

Parity check

A check that tests whether the number of ones or zeros in an array of binary digits is correct. This check helps ensure that the data read has not been unintentionally altered.

Passive search list

An unordered list of the file structures that have been in the job's active search list but have been removed without ever having been dismounted. This list is maintained by the SETSRC program and is use for accounting purposed when you log out.

Password

The character string assigned to a user; it is known only to the user, the installation administration, and the monitor system. The password is used to verify that a user is entitled to run a job under a specific project-programmer number.

Path

See directory path.

Pathological name

The logical name associated with a directory path. The pathological name refers to the list of structures and directories (STR:[UFD,SFD1,SFD2,...,SFD5]) to be searched any time the pathological name is specified as the device in the file specification.

PC

See "program counter."

Peripheral equipment

Any unit of equipment, distinct from the central processing unit, the console, and the memory, that can provide input to, or accept output from, the computer.

Per-process space

The portion of monitor memory used to store data specific to user jobs. Also called "funny space."

PHB

Packet Header Block used to store information when using IPCF.

Physical address space

A set of physical memory locations where information is actually stored for the purpose of program execution. (As opposed to virtual memory addresses, which may be mapped, relocated, or translated to produce a physical memory address in the hardware memory units. This physical address is 22 bits long on the DECsystem-10.

Physical device name

The name of a specific peripheral unit. It is a SIXBIT name consisting of 3 to 6 characters. Examples: FHA0, FHA1, DPA0, DPA7, LPT0, DTA3.

PI

See "priority interrupt."

PID

A Process IDentifier is used to identify a system program that is the target of communication using the IPCF facility.

PIT

PSI system's internal data base.

PLT

The generic device name for plotter.

PMB

The Packet Message Block where user data is stored to be sent to another program, using IPCF.

Pointer

1. A location or register containing an address rather than data. A pointer may be used in indirect addressing or in indexing.

2. An instruction indicating the address, position, and length of a byte of information (such as a byte pointer).

Policy CPU

In a symmetric multi-processing system, the CPU that provides system initialization and other overhead functions for the rest of the central processors.

Pool

One or more logically complete file structures that provide file storage for the users and that require no special action on the part of the user.

Pooled Resource

A pooled resource occurs when multiple copies of a resource exist. You specify that a resource is to be a pooled resource with the ENQ. monitor call.

Positioning operation

On the TOPS-10, the operation of moving the read-write heads of a disk to the proper cylinder prior to a data transfer. This operation requires the control for several micro-seconds to initiate activity, but does not require the channel or memory system.

Prime RIB

The first retrieval information block (RIB) of a file. This block contains all file attributes and pointers to data blocks on disk. Refer to RIB definition.

Priority interrupt

An interrupt that usurps control of the computer from the program or monitor and jumps to an interrupt service routine if its priority is higher than the interrupt currently being serviced.

Privileged program

1. Any program running under project number 1, programmer number 2.

2. A monitor support program executed by a monitor command which has the JACCT (job status) bit set, for example, LOGOUT.

Process

A collection of segments that perform a particular task. Usually synonymous with "job," "program," or "task."

Program

1. The complete plan for the solution of a problem, more specifically the complete sequence of machine instructions and routines necessary to solve a problem.

2. A collection of routines which have been linked and loaded to produce a saved file or a core image. These routines typically consist of a main program and a set of subroutines, some of which may have come from a library.

Program counter (PC)

A register that contains the address from which the next instruction to be executed is fetched. At the beginning of each instruction on a PDP-10, the PC normally contains an address that is one greater than the location of the previous instruction.

Programmed operators

Instructions which, instead of performing a hardware operation, cause a jump into the monitor system or the user area at a predetermined point and perform a software operation. The monitor (or special user code) interprets these entries as commands from the user program to perform specified operations.

Program origin

The location assigned by LINK to relocatable zero of a program.

Program trap

One of the software-defined operation codes which, when decoded by the processor, causes the next instruction to be executed from a specified address.

Project-programmer number

Two octal numbers, separated by commas, which, when considered as a unit, identify the user and his file storage area on a file structure.

Protected location

1. A storage location which cannot be accessed in a certain context. For example, a write-protected location cannot be written into.

2. A storage location reserved for special purposes in which data cannot be stored without undergoing a screening procedure to establish suitability for storage therein.

Protection address

The maximum relative address that the user can reference.

Protection code

Each file has a protection code that indicates who may or may not access the file, in the form <opa>, where o is an octal digit representing accessability to the owner of the file, p is the digit for members of the same project (possessing the same project number in their PPN), and a is the digit for all other users. Each octal digit represents the level of access allowed to the appropriate type of user, from Ø, allowing any type of access, to 7, allowing no access to other users. Note that the owner can always change the protection code associated with the file.

PSI System

Programmable software interrupt system.

Pseudo-op

An operation that is not part of the computer's operation repertoire as realized by hardware; hence, an extension of the set of machine operations. In MACRO, pseudo-ops are directions for assembly operations.

Pseudo-terminal

A simulation of a terminal device generated by the software to accept commands from a data base rather than a physical input device.

PTP

The generic device name used to refer to the paper tape punch.

PTR

The generic device name used to refer to the paper tape reader.

PTY

The generic device name used to refer to a pseudo-terminal.

Public disk pack

A disk pack belonging to the storage pool and whose storage is available to all users who have quotas on it.

Public mode

The user submode on the processor.

Pure code

Code which is never modified in the process of execution. Therefore, it is possible to let many users share the same copy of a program.

Pushdown list

A list that is constructed and maintained so that the next to be retrieved is the most recently stored item in the list. Also called "stack" and first-in/last-out (FILO) list.

Pushup list

A list that is constructed and maintained so that the next item to be retrieved and removed is the oldest item in the list. Also known as a first-in/first-out (FIFO) list.

Quantum time

The processor time given to each job when it is assigned to run.

Queue

A list of items waiting to be scheduled or processed according to system, operator, or user-assigned priorities. Examples: batch input queue, spooling queues, monitor scheduling queues.

QUOTA.SYS

The file that contains a list of users and their quotas for the private file structure on which the file resides. Created using PULSAR.

Random access

A process having the characteristic that the access time is effectively independent of the location of the data.

RDA

The generic device name used to refer to a Remote Data terminal.

Read

Input data from a file.

Record

A collection of adjacent related items of data treated as a logical unit.

Record gap

An area on a data medium between consecutive records. It is sometimes used to indicate the end of a block or record.

Recursive

A repetitive process in which the result of each process is dependent upon the result of the previous one.

Reentrant program

A program consisting of sharable code which can have several simultaneously independent users.

Refresh

To remove all files from a file structure and to build the initial set of files based on information in the HOME block.

Relative address

The address before hardware or software relocation is added.

REL file

A file containing one or more relocatable object modules.

Relocatable address

An address within a module which is specified as an offset from the first location in that module.

Relocate

1. To move a routine from one portion of storage to another and to adjust the necessary address references so that the routine can be executed in its new location.

2. To convert a relocatable binary module to an absolute binary module.

Relocation counter

1. The number assigned by LINK-10 as the beginning address of a Control Section. This number is assigned in the process of loading specific Control Sections into a saved file or a core image and is transformed from a relocatable quantity to an absolute quantity.

2. The address counter that is used during the assembly of relocatable code.

Relocation factor

The contents of the relocation counter for a control section. This number is added to every relocatable reference within the Control Section. The relocation factor is determined from the relocatable base address for the control section (usually Ø and 4ØØØØØ) and the actual address in user virtual address space at which the module is being loaded.

Remote access

Pertaining to communication with a data processing facility by one or more stations that are distant from that facility.

Remote peripherals

The I/O devices and other data processing equipment that are located at the site of the remote station.

Removing a file structure

The process of physically removing a file structure from the system. This is requested with the REMOVE switch in the DISMOUNT command string and requires the operator's approval.

Resource

Any entity within the system. The actual definition of a resource is defined by the job(s) using that resource. Refer to ENQ/DEQ.

Response time

The time between the generation of an inquiry or request and the receipt of the response or the accomplishment of the requested action.

Restricted device

Your program can use a restricted device only if it is assigned to you by a privileged job. You ask for this assignment by issuing the MOUNT command.

Retrieval Information Block (RIB)

The block that contains pointers to all the groups in a specific file. Each file has two copies of the RIB, one in the first block of the first group and the second in the block following the last data block in the last group of the file.

Return

1. The set of instructions at the end of a subroutine that transfers control to the proper point in the calling program.

2. The point in the calling program to which control is returned.

3. Informally, the carriage-return/line-feed sequence. Refer to CRLF.

RIB

See Retrieval Information Block.

Routine

A set of instructions and data for performing one or more specific functions.

Run

To transfer a save file from a device into core and to begin execution.

SAT.SYS

The Storage Allocation Table file which contains a bit for each cluster in the file structure. Clusters which are free are indicated by zero and clusters which are bad, allocated, or nonexistent are indicated by one.

Save

To produce a file from a core image of a program in memory. This results in an executable file that can be loaded and run without relocation.

Scan

The process of examining and parsing a text string. The SCAN program parses commands for the monitor.

Search

1. The process of locating an object by examining each object in the set to determine if it is the desired object or if the desired object exists.

2. The process by which the disk controller reads sector heads to find the correct sector. The second step in the transfer operation.

Search List

A list of the file structures that may be searched when files are referenced. (One of the ALL Search List, Job Search List, or SYS Search List.)

Secondary storage

Low speed magnetic storage such as disks or drums.

Sector

A physical portion of a mass storage device.

Segment

An absolute Control Section. A logical collection of data, either program data or code, that is the building block of a program. The monitor keeps a segment in core and/or on the swapping device.

Segment resident block

A block that contains all the information that the monitor requires for a particular segment.

Service routine

A routine in general support of the operation of a computer.

SETSRC

A program that allows the user to list or change his search list.

SFD (sub-file directory)

A directory pointed to by a UFD or a higher-level SFD. Each user has a UFD. Within that, he may have as many SFDs as he wishes.

Sharable segment

A high segment that can be used by several programs at a time.

Shared code

Pure code residing in a shared segment.

Sharer's Group

A subset of those jobs desiring shared ownership of a particular resource.

Simultaneous Update

Allowing more than one cooperating job to update a file.

Single access

The status of a file structure that allows only one particular job to access the file structure. This job is the one whose project number matches the project number of the owner of the file structure.

SIXBIT code

A 6-bit code in which textual information is recorded. It is a compressed form of the ASCII character set, and thus not all of the characters in ASCII are available in SIXBIT, notably the nonprinting characters and the lower case letters are omitted. The range of SIXBIT code is ØØ to 77 (octal) which is equal to 4Ø through 137 (octal) in ASCII.

Skip

The process by which an instruction, macro or subroutine causes control to bypass one instruction and proceed to the next instruction.

Software Interrupt System

Interrupts the sequential flow of program execution under a variety of conditions. Also called PSI system.

Spooling

The technique by which output to slow-speed devices is placed into queues on faster devices (such as disk) to await transmission to the slower devices; this allows more efficient use of the computer.

Storage Allocation Table

A file reflecting the status of every addressable block on the disk (SAT.SYS).

String

A set of contiguous items of a similar type. Generally strings are sequences, of variable or arbitrary length; of bits, digits, or characters.

STRLST.SYS

The administrative file that describes each file structure in the system. This file is used by the MOUNT command.

Structure

A File Structure.

Sub-File Directory

An extension of the user-file directory that allows the user to categorize his files into sub-groups.

Subroutine

A routine designed to be used by other routines to accomplish a specific task.

Super-cluster

A contiguous set of one or more clusters introduced to compress the file pointer for large units into 18 bits. Refer to compressed file pointer.

Super-USETI

A style of reading a disk unit or file structure by giving absolute addresses rather than locations within a file.

Supersede

To open a file for writing, write the file, and close the file when an older copy of the same name already exists. Only one user at a time may supersede a given file at any one time. The older copy of the file is deleted when all users are finished reading it.

Supervisor mode

The executive submode of the processor. Similar to public mode; however, code executed in supervisor mode is able to access, but not alter, concealed mode code.

SWAP.SYS

The file containing the swapping area on a file structure.

Swapping

1. The technique in multiprogramming of running more jobs than there is physical memory for, by storing some of the jobs on secondary storage when they are not executing.

2. The action of moving user programs between core and secondary storage.

Swapping class

A category of swapping units distinguished from other categories of swapping units according to speed. Class Ø contains the fastest swapping units.

Swapping device

Secondary storage that is suitable for swapping, usually a high-speed disk or drum.

Switch

1. The part of a file specification which is preceded by a slash.

2. One of several physical controls on the operator's console.

3. A flag used to control the path of execution within programs.

Symbol

Any identifier used to represent a value that may or may not be known at the time of its original use in a source language program. Symbols appear in source language statements as lables, addresses, operators, and operands.

Symbolic address

An address used to specify a storage location in the context of a particular program. Symbolic addresses must then be translated into relocatable (or absolute) addresses by the assembler.

Symbol table

A table containing entries and binary values for each symbol defined or used within a module. This table generally contains additional information about the way in which the symbol was defined in the module.

SYS

A system-wide logical name for the system library. This is the area where the standard programs of the system are maintained.

SYS search list

The file structure search list for device SYS. This is also used for several of the ersatz devices because it is a constant, well-ordered list.

SYSTAT

A program that outputs to the user's terminal status information on the system as a whole, on selected aspects of the system, or on a selected job or set of jobs.

Terminal

A device, normally consisting of both a keyboard and printing (or display) mechanism, that is used to enter information into a computer and to accept output from a computer. When it is used as a timesharing terminal, the computer to which it is connected can be very close or many miles away.

Tick

See Jiffy.

Total user core

The amount of physical core which can be used for locked and unlocked jobs. This is all of the physical core minus the core size of the monitor.

Track

The portion of a moving storage medium, such as disk, drum, or tape, that is accessible to a given reading head position.

Transfer operation

The hardware operation of connecting a channel to a controller and a controller to a unit for passing data between the memory and the unit. The transfer operation involves verification, search, and actual transfer.

Translate

To compile or assemble a source program into a machine language program, usually in the form of a (relocatable) object module.

Trap

An unprogrammed conditional jump to a known location, automatically activated by a side effect of executing a processor instruction. The location from which the jump occurred is then recorded. It is distinguished from an interrupt which is caused by an external event.

Trap Servicing Routines

Allow programs to handle errors while a program is running. Some of the errors that can be handled in this manner are illegal memory references, and pushdown list overflows.

TSK

The generic device name for the device used to refer to one program involved in inter-task communication with another program.

Two's complement

A number used to represent the negative of a given value. This number is obtained by substituting a zero for each one and a one for each zero in the bit configuration of the binary number and adding one to the result.

UFD

1. A file whose entries are the names of files existing in a given project-programmer number area within a file structure.

2. The top-level directory for each user. Also, the top-level directory for the ersatz devices which appear as one directory.

Unconditional transfer

An instruction which transfers control to a specified location.

Unit

The smallest portion of a device that can be positioned independently from all other units. Several examples of units are: a disk, a disk pack, and a drum.

Universal Device Index (UDX)

A number used to identify any device on the system. The monitor assigns the device a UDX when your program issues an IONDX. monitor call.

Update

To open a file for reading and writing simultaneously on the same software channel, rewrite one or more blocks in place, and close the file.

User's program

All of the data and code running in a user virtual address space.

User file directory

See UFD.

User I/O mode

1. The central processor mode that allows a user program to be run with automatic protection and relocation in effect, as well as the normal execution of all defined operation codes (including I/O instructions).

2. The monitor mode which allows a job to run with the I/O mode hardware on.

User library

Any user file containing one or more relocatable binary modules of which some or all can be loaded in library search mode.

User mode

A central processor mode during which instructions are executed normally except for all I/O and HALT instructions, which return control to the monitor. This makes it possible to prevent the user from interfering with other users or with the operation of the monitor. Memory protection and relocation are in effect so that the user can modify only his area of core.

User virtual address space

A set of memory addresses within the range of Ø to 256K-1 words. These addresses are mapped into physical core addresses by the paging or relocation-protection hardware when a program is executed.

UUO

Refer to programmed operators (Unimplemented User Operations).

Variable

Any entity that can assume any of a given set of values. When stored in core, a variable can occupy part of a core location, exactly one core location, or more than one core location.

Vestigial job data area

The first 1Ø (octal) locations of the high segment used to contain data for initializing certain locations in the job data area. These locations are usually 4ØØØØØ-4ØØØØ7 (octal) inclusive.

Wildcard construction

A technique used to designate a group of files without enumerating each file separately. The file name, extension, or project-programmer number in a file specification can be replaced totally with an asterisk or partially with a question mark to represent the group of files desired.

Word

An ordered set of bits which occupies one storage location and is treated by the computer circuits as a unit. The word length of the DECsystem-1Ø is 36 bits. This means that it is possible to store 36 bits of information at each memory address and to transfer all 36 bits between memory and the CPU at the same time.

Working set

The collection of pages in physical core immediately accessible to a job. Pages in core, but with the accessiblity bit off are also included in the working set.

Zero compression

The technique of compressing a core image by eliminating consecutive blocks of zeros and replacing them with an indication of the number of words of zeros that were removed.

Zero length module

A module containing symbol definitions but no instruction or data words (for example, JOBDAT). Note that the word "length" in this context refers to the program length of the module after loading.

APPENDIX B

.EXE FILES

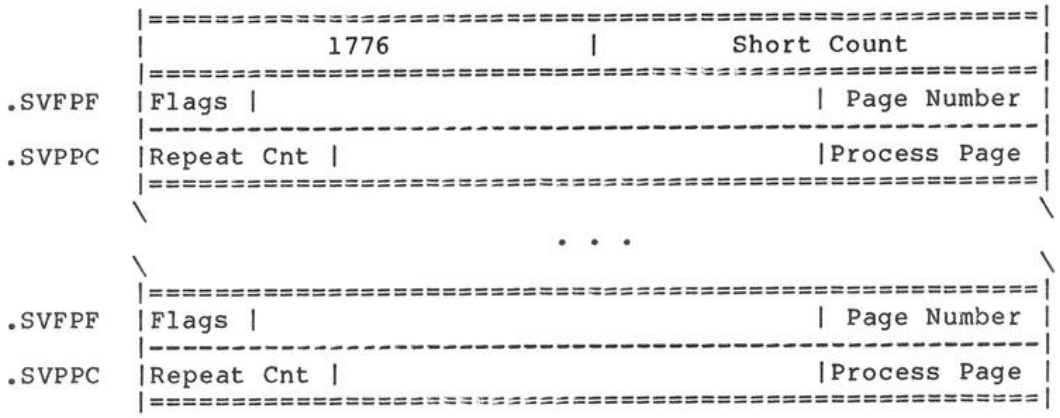
An .EXE file consists of a directory page followed by one or more pages of data.

The data in the directory page consists of a variable number of chunks. Each chunk starts with a word containing a code in the left half and a count of the number of words in the chunk in the right half. The following codes are defined:

1775	.SVSTA	Entry vector block.
1776	.SVDIR	Directory.
1777	.SVEND	End of directory.

B.1 THE DIRECTORY

The directory for an .EXE file starts with a .SVDIR header word and contains one or more 2-word entries that map the pages of the .EXE file into a process' address space. This format is represented in the following diagram.



The format of each .SVFPF word is as follows:

Bits	Symbol	Meaning
BØ	SV%HIS	Page is in high segment.
B1	SV%SHR	Page is sharable.
B2	SV%WRT	Page is writable.
B3	SV%CON	Page is concealed.
B4	SV%SYM	Page is part of symbol table.
5-22		Reserved.
23-35		The page number in the .EXE file at which the page starts.

The format of each .SVPPC word is as follows:

Bits	Symbol	Meaning
Ø-8	SV%REP	Repeat count (the number of contiguous pages minus 1 that are described by this entry).
9-22		Reserved.
23-35	SV%PPN	The process page number into which the page should be loaded.

APPENDIX C

FILE DAEMON

The File Daemon provides extended file protection. The File Daemon described in this appendix is a prototype that you may use to help you in understanding the monitor support for this feature. The File Daemon is supplied only to serve as a prototype for the File Daemon you may desire at your installation.

Each installation will have varying types of accounting and file security measures. Therefore, each installation's File Daemon may be written to account for these differences and varying requirements. The DIGITAL-supplied, prototype File Daemon supports access lists and access logging that is performed on a user's or a system administrator's request.

C.1 USER INTERFACE

The File Daemon allows any user to specify who can and who cannot access his files. Each user may create a file named ACCESS.USR (which is described in Section C.3). This file optionally lists the names of some or all of that user's files and specifies, on an individual file basis, the users who can and cannot access those files. Under specific conditions, the File Daemon examines the user's ACCESS.USR file and may record information, in a separate file called ACCESS.LOG, regarding specific access requests to the listed files. Note that ACCESS.USR can be created only by the owner of the particular directory or by a job logged in under [1,2].

C.2 THE FILE DAEMON

The monitor calls the File Daemon (only if the monitor feature test switch F%FDAE = -1) each time that someone tries to access a file that has a 4, 5, 6, or 7 protection code in the owner's protection code field and the access fails due to a file protection error or due to a directory protection error.

For example, if you protect a file against a specific user and that user attempts to access your file (with a LOOKUP, ENTER, RENAME, or FILOP. monitor call), the monitor suspends the execution of the accessing user's program and it sends a message to the File Daemon. This message includes the type of access the user is attempting and that user's project-programmer number. The monitor gives control to the File Daemon, which looks for your file called ACCESS.USR. ACCESS.USR must be on the same file structure and in the same directory area as the file being accessed.

After examining ACCESS.USR, the File Daemon returns to the monitor the highest type of access you have specified that the user attempting access to your file may have. Then, the File Daemon logs the access request in ACCESS.LOG (if you set the /LOG switch in your ACCESS.USR file; refer to Table C-1).

All of this occurs, even when you attempt to access your own files, if a file has a 4, 5, 6, or 7 protection code in the owner's protection code field. However, as the file's owner, you can read your file and change the file's protection code without having the File Daemon called. Depending on the information you specified in your ACCESS.USR file, the File Daemon either grants or denies access to the accessing user.

If the monitor attempts to pass control to the File Daemon, but the File Daemon is not running, the monitor denies access to the file unless the program attempting access has full file access rights ([1,2] or JACCT). The same result occurs when one of the following conditions occurs:

1. The File Daemon cannot find ACCESS.USR in the same path as the file being accessed.

2. The File Daemon cannot find ACCESS.USR in a higher-level directory, when it scans up the directory structure.

If the File Daemon finds ACCESS.USR but cannot find the name of the accessed file in ACCESS.USR, the File Daemon denies file access to the accessing user. The File Daemon also denies access to the accessing user if the File Daemon finds he specified filename in ACCESS.USR but the project-programmer number does not match any of the project-programmer numbers you have specified that may access your file.

All files listed in your ACCESS.USR are assumed to be in the same User File Directory (UFD) as the file named ACCESS.USR. However, if your ACCESS.USR is in your UFD and it describes the type of accesses to be allowed to files contained in the SFDs, the accessing user must specify the full path to the file in the SFD before the File Daemon will consider the file specification to match.

The File Daemon treats all file accessors the same. All accesses to a file having a 4, 5, 6, or 7 protection code in the owner's protection code field cause the File Daemon to be called when a protection error results. The File Daemon is always called when a protection error occurs as a result of the directory protection code. Because of this equal treatment, you should not do the following:

1. If a [1,2] job attempts to access a file that is protected such that the File Daemon is called, that job may be denied access to the file. This is a possible problem, for example, if the [1,2] job is BACKUP and you have denied (either implicitly or explicitly) these programs access to your files. When you do this, your file will not be backed up on magnetic tape. Therefore, you must accept the responsibility of backing up your own files.

2. In general, full file access programs will not be allowed to read your files. Therefore, under most circumstances, QUEUE would not be allowed to queue a file that was protected such that the File Daemon was called.

3. If the file's owner protection code field is such that the File Daemon is called and the owner has neglected to include his own project-programmer number in ACCESS.USR for this file, the File Daemon grants the owner the same type of access as if a 7 were in the owner's protection code field (that is, the owner can only read the file or change the file's protection code.)

4. ACCESS.USR files may be restored at arbitrary times. Therefore, operators should not perform a full restore of the disk using BACKUP when the File Daemon is running. If such a full restore is done, the action may not allow BACKUP to restore files that ACCESS.USR allows them to BACKUP.

5. The CHKACC monitor call tells a program what a user's file access privileges are. Therefore, by using CHKACC, a program can tell if the File Daemon will be called, but it will not know the access privileges returned by the File Daemon.

C.3 ACCESS.USR

Every user can create his own ACCESS.USR file. Note that ACCESS.USR files can be created only by the owner of the specific directory or a [1,2] job. ACCESS.USR is made up of one or more 'command lines'. You must write each command line in the following format:

```
file-spec/switches=[ppn]/switches,...,[ppn]/switches
```

The file-spec is a full file specification (that is, device: filename.extension [path]). The File Daemon scans each line in ACCESS.USR until it matches a file specification on the left of the equal sign and a project-programmer number on the right. All access rights will then be determined by that line (there will be no continued scan). The user should minimally specify one of the switches synonymous with protection codes (such as, READ, EXECUTE, ALL,...) for that file specification; refer to Table C-1. If you do not specify a switch, a default of /NONE is provided. The possible switches are listed in Table C-1.

Table C-1: ACCESS.USR Switches

Switch	Meaning
/LOG /NOLOG	This switch causes the File Daemon to log any access attempt in the file named ACCESS.LOG. If you specify this switch, the File Daemon appends a LOG entry to the end of ACCESS.LOG, which is found in the same directory as your ACCESS.USR. The log entry includes the following: o the date of the access o the time of the access o the job number of the accessing job o the project-programmer number and name associated with the accessing job o the name of the accessing program o the type of access attempted o the full file specification of the access file o the access permitted, detailing whether access was permitted to the file If you also specify the /EXIT or /CLOSE switch, the File Daemon includes the following information in the LOG entry (both the initial entry and when the file is closed: o the accessing job's run time o kilo-core-seconds o disk reads o disk writes If the File Daemon cannot find ACCESS.LOG in your area, it creates one, giving it the same protection code as your ACCESS.USR. Note that the File Daemon can always access ACCESS.USR and ACCESS.LOG.

Table C-1: ACCESS.USR Switches (Cont.)

Switch	Meaning
/LOG:n	This switch allows the File Daemon to log access attempts based on the switch value. The following are the legal switch values: ALL — Log all accesses attempted (same as /LOG). NONE — Do not log any accesses (same as /NOLOG). SUCCESSES — Log only those accesses that were permitted. FAILURES — Log only those accesses that were not permitted.
/CLOSE /NOCLOSE	If you specify the /LOG switch and the /CLOSE switch, the File Daemon makes the log entry when the file is closed.
/EXIT /NOEXIT	If the accessing program is executing and you have specified the /LOG and /EXIT switches, the File Daemon makes the log entry when the program has finished execution.
/CREATE /NOCREATE	This switch allows a user who would ordinarily not be allowed to create files in your directory to do so. This switch is used in conjunction with one of the ACCESS.USR switches that are synonymous with protection codes (such as /RENAME). This switch can appear on either side of the equal sign. An example of a command line with the /CREATE switch is as follows: WONDER.TST=[10,3333]/CREATE/NONE This command line allows any user to create a file called WONDER.TST in your directory, but none of these users may have any other access to that file. Another example is WOND.TST=[10,10]/CREATE/READ,[*,*]/NONE This command line prevents all users from accessing the file WOND.TST, but allows user [10,10] to create a file called WOND.TST.

Table C-1: ACCESS.USR Switches (Cont.)

Switch	Meaning
/PROT:nnn	This switch specifies the protection code with which a file will be created. This switch is allowed only on the left side of the equal sign. The value nnn must be an octal number in the range Ø-777. The file is created with the specified protection code if the following conditions occur: 1. You specify the /PROTECTION switch. 2. The File Daemon is called because a user attempted to create a file in a directory protected against that user. 3. The File Daemon allows the user to create the file (determined by the contents of ACCESS.USR).
/PROG:file	This switch allows the specified program to have the desired type of access to the file. This switch can appear only on the right side of the equal sign in the command line. For example: ONE.TST/READ=[1Ø,1Ø],[1Ø,65]/WRITE,[1,2]- #/PROGRAM:SYS:BACKUP This command line specifies that [1Ø,1Ø] jobs can read ONE.TST, and [1Ø,65] jobs can read and write ONE.TST, a job logged in under [1,2] running BACKUP can read the file. No one else can access ONE.TST. You may omit the device specification or you may specify DSK: or ALL: in the filespec argument to the /PROGRAM switch. However, this is not a recommended procedure because there may be potential security violations. The File Daemon has no knowledge of your search list; therefore, the File Daemon treats DSK: identically to ALL:. It is recommended that the device name be either a file structure name or an ersatz device name (LIB: is not allowed, however).
/XONLY	This switch, when it appears in conjunction with the PROGRAM switch, considers the specified program to match the program doing the accessing, only if the accessing program is Execute-only.

Table C-1: ACCESS.USR Switches (Cont.)

Switch	Meaning
/ALL	This switch specifies that ALL access to the file is allowed. Specified accessors of this file can change the protection code for the file, rename, write, execute, update, and append to the file. (This is equal to protection code Ø.)
/RENAME	This switch specifies that rename access is allowed. Specified accessors of this file can rename, execute, write, read, update, or append to the file. (This is equal to protection code 1.)
/WRITE	This switch specifies that write access is allowed. Desired accessors of this file can write, read, execute, update, and append to the file. (This is the same as protection code 2.)
/UPDATE	This switch specifies that update access is allowed. Specified accessors of the file can update, append, read, and execute the file. (This is equal to protection code 3.)
/APPEND	This switch specifies that append access is allowed. Specified accessors of this file can append, read, or execute the file. (This is the same as protection code 4.)
/READ	This switch specifies that read access is allowed. Specified accessors of this file can read or execute the file. (This is the same as protection code 5.)
/EXECUTE	This switch specifies that execute access is allowed. Specified accessors of this file can only execute the file. (This is the same as protection code 6.)
/NONE	This switch specifies that no access is allowed to the file. (This is the same as protection code 7.)

You create an ACCESS.USR file to specify for each file which project-programmer numbers can access the file and what type of access those accessors can have. The switches indicate the type of access allowed.

Switches appearing on the left side of the equal sign affect all project-programmer numbers appearing on the right side of the equal sign. However, with the exception of the /PROTECTION switch, the switch on the left side can be overridden for one or more project-programmer numbers specified on the right side of the equal sign. You can override the switches by explicitly specifying another switch. For example, if the following line appeared in your ACCESS.USR file:

```
TST.TST/ALL=[10,*],[11,*],[27,*],[17,*]/NONE
```

The File Daemon would allow all members of projects 10, 11, and 27 complete access to the file TST.TST. However, the File Daemon would not allow members of project 17 to access TST.TST. For project-programmer numbers other than 10, 11, 27, 17, the File Daemon will search for a later TST.TST that contains the accessing job's project-programmer number. If no match is found, the File Daemon denies the accessing user's request.

Full wildcard specifications are allowed both on the left and right sides of the equal sign. Comments and continuation lines are allowed in ACCESS.USR. A comment must begin with a semicolon or an exclamation point. A continuation line is indicated by inserting a hyphen (minus sign) immediately proceeding the carriage return that terminates a line. If there is a syntax error in a line in ACCESS.USR, the File Daemon ignores that line. You should insure the accuracy of your own ACCESS.USR files by proofing carefully. If the following line were in your ACCESS.USR file:

```
FOO.BAR+[*,*]
```

The File Daemon would ignore the line because a + sign appears where an = sign should appear. The File Daemon would deny access to all users desiring access to FOO.BAR, since the File Daemon denies access to all files whose names do not appear in ACCESS.USR. Since the File Daemon ignores the line, it does not know that FOO.BAR is listed in the file.

The following is an example of an ACCESS.USR file that uses most of the features of the File Daemon.

```
Directory user = [13,675]

Directory protection = <700>

File            Protection

ACCESS.USR      <777>
ACCESS.LOG      <777>
F1.TST          <077>           File Daemon will not be called.
F2.TST          <457>           Project members may READ, otherwise
                                call File Daemon.
F3.TST          <477>           Only owner may access without File
                                Daemon.
F4.TST          <777>           Call File Daemon on all accesses.

ACCESS.USR

ACCESS.*/NONE=[*,*]             ;No one can touch ACCESS.USR and
                                ACCESS.LOG including [1,2] and
                                JACCT users. Note that these files
                                cannot be backed up if the File
                                Daemon is running.
```

```
ALL:*.*/READ/LOG=[1,2]/PROGRAM:SYS:BACKUP/XONLY

                                ;Allow access from BACKUP (from
                                SYS, execute only, and running
                                under [1,2] to read the file and to
                                make LOG entry.

F?.TST/LOG=[10,11]/NONE,[10,*]/EXECUTE/EXIT

                                ;Log all access attempts. No
                                access allowed to [10,11], but
                                other project members [10,*] can
                                execute the file. Log entries are
                                made when the accessing program
                                exits.

*.*/CREATE/PROTECTION:055=[12,21]/ALL,[12,17]

                                ;[12,21] has privileges for all
                                files (except ACCESS.*) and may
                                create files that have a protection
                                of 055. [12,17] cannot access any
                                file (/NONE is a default) but may
                                create files. No log entries will
                                be made.

*.*/CREATE/PROTECTION:777/LOG=[123,456]/NONE

                                ;[123,456] may create files at will
                                but may not access them (such as a
                                student turning in homework).

*.*[13,675,A]/ALL/PROTECTION:057/CREATE=[1,2]/LOG

                                ;[1/2] has all privileges in this
                                SFD and may create files with a
                                protection code of 057.

[13,675].UFD/LOG/READ=[*,*]     ;Anyone may read this directory as
                                a file.

F3.TST/LOG=[12,3]/EXECUTE
*.*/LOG=[12,3]/NONE             ;[12,3] can only execute F3.TST.

*.*=[*,*]/NONE                  ;No other access is granted and no
                                LOG entry is made.
```

Note that entries are scanned from left to right and top to bottom. The scan stops on the first match of a file name on the left side of the equal sign and a project-programmer number on the right side of the equal sign.

When you create your ACCESS.USR file, you should take care to see that a wild card specification will not match in a line earlier than a specific specification in a later line. As a general rule, place specific statements first in the ACCESS.USR file, followed by more general "catch all" statements. If you want to log entries, you must use the /LOG switch (and any of the other switches) on every line for which that switch applies.

C.4 MONITOR INTERFACE TO A FILE DAEMON

A File Daemon is a privileged program that can be used for the following purposes:

1. overseeing file accesses
2. aiding in proprietary billing
3. tracking program usage

The interface between the monitor and the File Daemon that is described in this section is supplied and supported by Digital.

There is a privileged program called the File Daemon. Digital supplies one unsupported version of a File Daemon, which is described in the preceding sections of this appendix. But, each installation should write its own File Daemon, because each installation will vary on its requirements for such a program.

When a File Daemon is running, the monitor calls it every time someone tries to access a file or a directory that has a 4, 5, 6, or 7 code in the owner's protection code field and the access fails due to a protection error. So that the monitor knows there is a File Daemon, the following must occur:

1. The feature test switch F%FDAE must be set to -1, to enable the condition.
2. The program that will be the File Daemon must be privileged (that is, it must be running under [1,2] or running with the JACCT bit set).
3. This program must send an IPCF request to [SYSTEM] IPCC (code 6, .IPCSC) requesting a special PID.
4. This program must then send a request to [SYSTEM]IPCC specifying code 24 (.IPCWP). This code requests that the File Daemon's PID be entered in the Special PID table.

After each request to [SYSTEM]IPCC, the File Daemon receives verification that the function occurred. After the verification resulting from the File Daemon specifying code 24, the monitor sends an IPCF packet to the File Daemon each time that a protection failure occurs on a file or a directory.

The message portion of the IPCF packet that the monitor sends to the File Daemon when a protection failure occurs has the following format:

Type of access	Code
File structure name	
File name	
File name extension	
Project number	Programmer num.
Sub-file directory 1 or Ø	
Sub-file directory 2 or Ø	
Sub-file directory 3 or Ø	
Sub-file directory 4 or Ø	
Sub-file directory 5 or Ø	

Where: **type of access** is the type of access being attempted to the file. The Access Type Codes are listed in Table C-2.

code is a File Daemon Code, all of which are listed in Table C-3.

The remaining words in the IPCF packet message are the full file specification for the file being accessed.

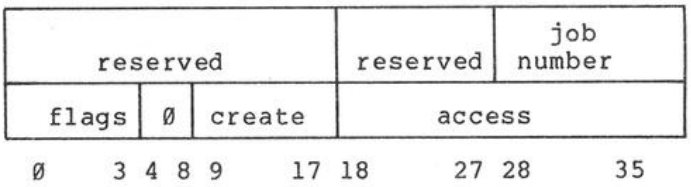
Table C-2: Access Codes

Code	Mnemonic	Meaning
Ø	FNCNAA	No access is allowed.
1	FNCEXE	Execute.
2	FNCRED	Read.
3	FNCALL	Allocate.
4	FNCDLL	Deallocate.
5	FNCAPP	Append.
6	FNCUPD	Update.
7	FNCCRE	Create.
1Ø	FNCSUP	Supersede.
11	FNCTRN	Truncate.
12	FNCCAT	Change attributes.
13	FNCDEL	Delete.
14	FNCCNM	Change name.
15	FNCCPR	Change protection.

Table C-3: File Daemon Codes

Code	Mnemonic	Meaning
1	.FLDCA	This code is set when the accessing program has performed a LOOKUP, ENTER, RENAME, or FILOP monitor call and a protection failure occurred.
2	.FLDIC	This code is set as a result of a previous call to the File Daemon, the File Daemon requested that it be called when the program issues a CLOSE. This code is set as the result of the program issuing an input CLOSE. Refer to Table C-5, flag bit 1.
3	.FLDOC	As a result of a previous call to the File Daemon, the File Daemon requested that it be called when the program issues a CLOSE. This code is set as the result of the program issuing an output CLOSE. Refer to Table C-5, flag bit 1.
4	.FLDXT	This code is set as a result of a previous call to the File Daemon, which occurred because a job tried to issue a R, RUN, or GET command or a RUN monitor call and a protection error resulted. The File Daemon requested that the monitor call it when the accessing program terminates execution. The termination of a program's execution is defined by the terminal user or by the batch .CTL file, either of which may type something that logically supersedes the core image. The program may also terminate its own execution by performing a RUN monitor call. Refer to Table C-5, flag bit 2.
5	.FLDPG	This code is set because a job tried to execute a protected program by issuing a R, RUN, or GET command or a RUN monitor call.
6	.FLDDA	This code is set because a directory protection failure occurred.
7	.FLDPS	This code is set when a PUSH occurs from a program that has /EXIT specified.
10		This code is set when a suspended program with /EXIT specified resumes due to a POP.

The File Daemon responds to the monitor by sending the monitor an IPCF packet. The packet's message is in the following format:



Where:

job number is the number of the job attempting to access a file.

flags are bits Ø through 3 which are described in Table C-4.

create is the protection code at which the file will be created if the specified job is creating a file.

access is the highest access this job is allowed to this file. Refer to Table C-3.

The monitor grants or denies the job's access to the file based on the access value and the type of access specified by the accessing job. If the access value in the packet from the File Daemon to the monitor is greater than or equal to the type of access the accessing job desired, the monitor grants the job access to the file.

Table C-4: File Daemon Flags

Code	Mnemonic	Meaning
Ø	FL.DAA	The monitor is to call the File Daemon every time this file is accessed. For example, if this bit is not set and the program did a RENAME before a LOOKUP, the File Daemon would get called only on the LOOKUP.
1	FL.DCL	The File Daemon is called when the file is CLOSED.
3	FL.DXT	The File Daemon is called when this program terminates execution.
3	FL.DSP	If the program is attempting to create a file and this bit is set, the monitor assumes that the protection code for the file is in bits 9 through 17 of this word.
777B17	FL.DPT	Protection code supplied by File Daemon.
777777B35	FL.DHA	Highest access allowed.